Pengendalian Robot DDMR (*Differentially Driven Mobile Robot*) Dengan Metode PWM Berbasis PLC SIEMENS S7-200

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

MUHAMMAD SYAIFUDIN EFENDI
NIM. 021 063 3056 – 63

DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG

2009

Pengendalian Robot DDMR (*Differentially Driven Mobile Robot*) Dengan Metode PWM Berbasis PLC SIEMENS S7-200

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

MUHAMMAD SYAIFUDIN EFENDI
NIM. 021 063 3056 – 63

Telah diperiksa dan disetujui oleh
DOSEN PEMBIMBING :

Pembimbing 1	Pembimbing 2
Ir. Bambang Siswojo, MT.	Ir. Purwanto, MT.
NIP. 131 759 588	NIP. 131 574 847

Pengendalian Robot DDMR (*Differentially Driven Mobile Robot*) Dengan Metode PWM Berbasis PLC SIEMENS S7-200

Disusun oleh :

MUHAMMAD SYAIFUDIN EFENDI
NIM. 021 063 3056 – 63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal **9 Februari 2009**

Majelis Penguji :

Dr. M. Azis Muslim, ST., MT.
NIP. 132 281 763

Goegoes Dwi N., ST, MT.
NIP. 132 318 316

Fitriana Suhartati ST. MT.
NIP. 132 206 527

Mengetahui :
Ketua Jurusan Teknik Elektro

Ir.Heru Nurwasito, M.Kom.
NIP. 131 879 033

RINGKASAN

Muhammad Syaifudin Efendi, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijiya, Februari 2009, "Pengendalian Robot DDMR(Differentially Driven Mobile Robot) Dengan Metode PWM Berbasis PLC SIEMENS S7-200", Dosen Pembimbing : Bambang Siswojo Ir, MT. dan Purwanto, Ir, MT.

Perkembangan dunia industri dewasa ini menuntut untuk dapat menghasilkan jumlah produksi yang maksimal tanpa melupakan kualitas produk yang dihasilkan. Sehingga teknologi robot banyak digunakan untuk mengatasinya. Merupakan tantangan yang besar bagi SDM Indonesia untuk menguasai teknologi tersebut.

Melihat keadaan ini penulis terpikir merancang robot yang dikendalikan dengan menggunakan PLC. Hal ini dilandasi oleh keunggulan yang dimiliki oleh PLC. PLC relatif handal terhadap gangguan, kita ketahui bersama PLC dibuat berdasarkan standar industri. Sehingga diharapkan robot yang dikontrol dengan PLC memiliki performa yang lebih baik.

Dari hasil pengujian terlihat bahwa robot yang dikontrol dengan PLC lebih stabil, hal ini terlihat dari rata-rata waktu tempuh yang diperlukan dari beberapa kali percobaan memiliki selisih waktu dan error yang relatif kecil.

Kata kunci : PLC, DDMR, SIEMENS

PENGANTAR

Assalamualaikum Wr.Wb.

Puji syukur penulis panjatkan yang setinggi-tingginya kehadirat ALLAH SWT, Dzat Yang Maha Tinggi Kemuliaan-Nya dan Maha Agung Kedudukan-Nya, karena hanya dengan Karunia-Nyalah laporan Skripsi ini terselesaikan. Penulis hanya bisa berserah diri, dan mengucap Alhamdulillaahirabbilaalamiin. Tidak lupa shalawat dan salam selalu tercurah kepada junjungan kita Nabi Muhammad SAW.

Melewati masa-masa pengerjaan laporan skripsi ini merupakan saat yang penuh perjuangan bagi penulis, menggali dan menggali materi dari berbagai sumber, dan tentu saja dari guru-guru penulis. Tak ada orang yang begitu saja menguasai ilmu tanpa tempaan dan pengajaran dari sosok guru.

Laporan Skripsi penulis yang berjudul "Pengendalian Robot DDMR (Differentially Driven Mobile Robot) Dengan Metode PWM Berbasis PLC SIEMENS S7-200" adalah salah satu syarat yang harus dipenuhi dalam kelulusan pendidikan yang penulis tempuh sebagai mahasiswa di Universitas Brawijaya.

Penulis mengucapkan banyak terimakasih yang tidak terhingga kepada berbagai pihak yang telah banyak membantu terutama kepada:

1. **Bapak Heru Nurwarsito, Ir, M Kom**, selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya dan **Bapak Rudy Yuwono, Ir,** selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
2. **Bapak Purwanto, Ir, MT**, selaku KKDK Teknik Kontrol Jurusan Teknik Elektro Universitas Brawijaya sekaligus selaku dosen pembimbing II, terimakasih atas banyak pengarahan dan diskusi untuk penyelesaian Skripsi ini.
3. **Bapak Bambang Siswojo Ir,** selaku dosen pembimbing I, atas banyak pengarahan dan bimbingannya dalam menyelesaikan Skripsi ini.
4. **Ayah dan Bunda** yang telah memberikan segala yang penulis butuhkan sampai hari ini termasuk cinta dan kasih sayang yang tiada

pernah habisnya, semoga ALLAH SWT selalu melimpahkan rahmat, karunia serta ampunan-Nya kepada mereka. Amin.

5. **Seluruh Dosen Teknik Elektro** yang telah banyak memberikan ilmunya sebagai bekal dalam menjalani perjalanan hidup yang merupakan proses belajar yang tiada akhir ini, semoga ilmu yang telah kalian berikan dapat bermanfaat bagi penulis dan bagi masyarakat pada umumnya.
6. **Seluruh staf**, petugas Laboratorium Teknik Kontrol dan seluruh karyawan Teknik Elektro Universitas Brawijaya yang telah banyak membantu selama penulis kuliah di universitas ini.
7. **Adikku Dhuduy,** yang selalu memberi spirit semangat dan do'anya. Semoga selalu diberi kesuksesan dalam hidupnya, Mas doakan juga kamu diberi kesehatan dan selalu dalam lindungan-Nya.

   Fathi Ismail Hawari, ponakanku yang lucu, canda tawamu selalu memberikan semangat baru bagiku, semoga kelak kamu jadi anak yang sholeh, berbakti sama orang tua dan selalu dalam lindungan-Nya.
8. **Seluruh keluarga besar di Gresik**: Eyang, seluruh Bude, Pakde, tante, om dan seluruh sepupuku yang selalu mendoakanku.
9. **Agus Setiono, Ahmad dan Djoko Santoso** yang selalu memberikan dorongan, sumbangsih sebagian ide dan pikirannya dalam terselesaikannya Skripsi ini. Makasih atas bantuannya.
10. **Seluruh teman Kontrakan,** *thanks* atas kebersamaan dan kekompakan kita selama ini. *Semoga kalian semua sukses.*
11. **Teman-teman Teknik Elektro UB** seluruh angkatan Ekstensi atau reguler, Bograh, Epep, Rosyid, Eric, Lookman, Mbah Edy, Farid, Adit, Anggas, Naben, dan teman-teman Precboth lainnya, *Terimakasih dan maaf atas segala kesalahan!, semoga persahabatan kita selalu abadi.*
12. Dan temen-temen lainnya yang telah begitu banyak memberikan saran dan waktunya serta berbagai pihak yang tidak mungkin bagi penulis untuk menyebutkan satu persatu, namun jasa mereka tetap melekat di sanubari penulis. Hanya Allah SWT yang paling pantas membalas jasa-jasa mereka.

Sebagai manusia biasa penulis merasa memiliki banyak keterbatasan, keterbatasan ilmu, keterbatasan tenaga, keterbatasan pikiran sehingga tidak menutup kemungkinan banyak melakukan kesalahan. Seperti kata pepatah bahwa "tidak ada gading yang tak retak", penulis pun menyadari bahwa pada Skripsi ini terdapat kekurangan-kekurangan baik disengaja maupun tidak. Oleh karena itu, kritikan dan saran yang membangun, sangat penulis harapkan demi kesempurnaan dari Skripsi ini atau untuk laporan-laporan berikutnya. Besar harapan penulis, Skripsi ini dapat membawa manfaat bagi semua pihak di kemudian hari.

Wassalamualaikum Wr.Wb.

Malang, Februari 2009

Penulis

DAFTAR ISI

Hal

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Memperhatikan perkembangan dunia industri dewasa ini, dapat kita lihat bahwa sebuah industri harus dapat bersaing untuk dapat menghasilkan jumlah produksi yang maksimal dengan kualitas produk yang terjaga. Teknologi robot merupakan solusi paling masuk akal untuk mengatasinya. Kondisi ini menuntut SDM Indonesia untuk segera meningkatkan penguasaan teknologi perekayasaan robot yang diharapkan dapat memberi nilai positif dalam upaya peningkatan daya saing nasional.

Sebagai mahasiswa Teknik Elektro kita wajib untuk meningkatkan kemampuan teori dan praktik dalam pengembangan dan penguasaan teknologi robot. Sementara ini teknologi yang digunakan untuk mengendalikan robot adalah dengan menggunakan mikrokontroller. Dengan alasan itulah sehingga dalam skripsi ini dipilih penggunaan PLC sebagai pengendali robot, alasan yang lain dikarenakan oleh keterbatasan yang dimiliki oleh mikrokontroler itu sendiri, pada mikrokontroler instalasi pengkabelan relatif lebih rumit, mikrokontroler memiliki sensitifitas yang tinggi terhadap gangguan terutama drop catu daya, pengaruh medan magnet motor dan guncangan dari pergerakan robot.

Dipilihnya PLC sebagai pengendali dengan alasan, pada PLC program berupa ladder diagram sehingga I/O bisa ditampilkan, pada PLC instalasi pengkabelan relatif lebih mudah dibanding pada penggunaan mikrokontroler, pada PLC kalibrasi sensor dapat dilihat langsung pada LED indikator PLC, hal ini sangat memudahkan *trouble shoting* bila terjadi permasalahan *hardware* dan *software* dan keuntungan yang paling utama pada penggunaan PLC adalah kehandalannya terhadap gangguan, karena kita ketahui bersama PLC dibuat berdasarkan standar industri.

Selain kekurangan dari segi pengendali, dalam perancangan sebuah robot agar diperoleh kemampuan pergerakan yang baik sangat tergantung pula pada kualitas motor DC yang digunakan, motor DC berperan sangat vital terhadap

kemampuan sistem pergerakan sebuah robot. Adakalanya motor DC dengan merek dan spesifikasi yang sama pada kenyataan di lapangan memberikan karakteristik yang berbeda dalam hal responnya terhadap aksi kontrol. Atau mungkin karena alasan tertentu kita terpaksa menggunakan dua motor DC yang berbeda putaran maksimumnya, sehingga menyebabkan pergerakan roda robot tidak stabil, untuk itulah digunakan metode PWM berbasis PLC Siemens S7-200 untuk mengendalikan kecepatan putar motor DC tersebut agar diperoleh kualitas mobilitas robot yang baik.

1.2. Rumusan Masalah

Dari latar belakang masalah di atas dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana menggunakan PLC sebagai sistem pengendali pada robot DDMR (Differentially Driven Mobile Robot) otomatis.
2. Bagaimana merancang program (ladder diagram) kendali pada PLC sehingga dapat melakukan sistem mobilitas (pergerakan) yang diinginkan pada robot DDMR.

1.3 Ruang Lingkup

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan alat yang akan dibuat, diberi batasan sebagai berikut:

1. Gerak pada robot DDMR mengikuti lintasan garis hitam. Lintasan yang digunakan berupa lintasan lurus sejauh 120 cm kemudian berbelok 90^0 ke kiri. Setelah itu kembali ke lintasan lurus lagi sejauh 350 cm dan berbelok 90^0 ke kanan.
2. PLC yang digunakan *SIEMENS S7-200* dengan fasilitas 14 input dan 10 output yang terdapat pada Laboratorium Sistem Kontrol Jurusan Teknik Elektro Universitas Brawijaya.
3. Software yang digunakan adalah STEP 7-MicroWIN 32 v3.1.1.6.

1.4 Tujuan

Tujuan penyusunan skripsi ini adalah untuk merancang dan membuat sebuah sistem pengendalian robot DDMR otomatis dengan menggunakan PLC untuk meningkatkan mobilitas robot.

1.5. Sistematika Penulisan

Skripsi terdiri dari enam bab dengan sistematika penulisan sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika pembahasan.

BAB II Teori Penunjang

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat, yang meliputi dasar teori PLC (*Programmable Logic Controller*), dasar teori PWM, dasar teori robot, motor DC, sensor pengikut garis (*Line Follower*), dan sensor putaran motor (Encoder).

BAB III Metodologi

Berisi tentang metode penelitian dan perencanaan alat.

BAB IV Perancangan Sistem

Memuat perancangan sistem pengendalian robot dengan kontroler PLC dan bagaimana penerapannya pada sistem secara keseluruhan.

BAB V Pengujian Sistem

Memuat hasil pengujian terhadap alat yang telah direalisasikan.

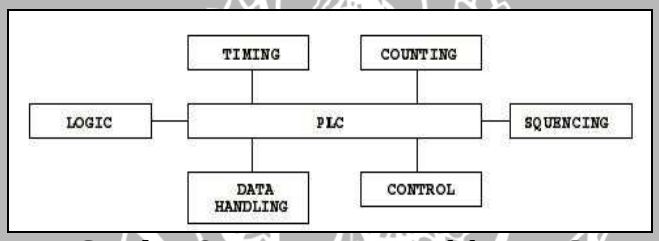
BAB VI Kesimpulan dan Saran

Memuat kesimpulan dari hasil perancangan yang dibuat dan saran-saran yang diperlukan untuk melakukan pengembangan aplikasi selanjutnya.

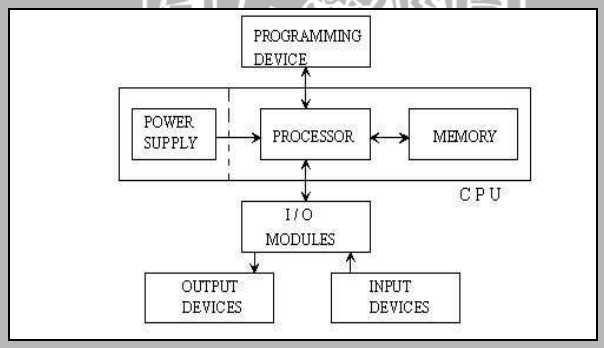
BAB II
DASAR TEORI

2.1 Programmable Logic Controller (PLC)

PLC merupakan pengontrol berbasis mikroprosesor yang dapat diprogram untuk menyimpan instruksi-instruksi dan untuk mengimplementasikan fungsi-fungsi logika, perurutan (sequencing), pewaktuan (timing), pencacahan (counting), dan aritmatika guna mengontrol mesin-mesin dan proses-proses. PLC memantau input-input dan output-output sesuai dengan instruksi-instruksi di dalam program dan membuat keputusan berdasarkan program. [PUT-04]



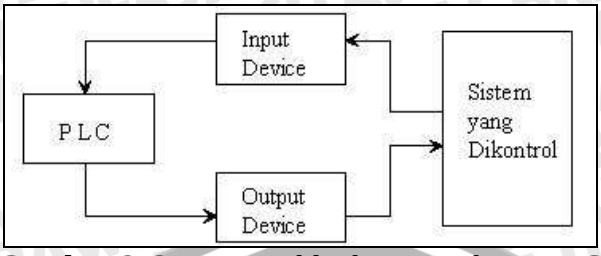
Gambar 2.1 Fungsi-Fungsi dalam PLC
Sumber: A Beginners Guide to PLC, 1993:13



Gambar 2.2 Arsitektur PLC
Sumber: A Beginners Guide to PLC, 1993:15

2.1.1 Prinsip Kerja PLC

Pada prinsipnya, sebuah PLC bekerja dengan cara menerima data-data dari peralatan input luar atau "*Input Device*", seperti yang dijelaskan pada gambar 2.3.

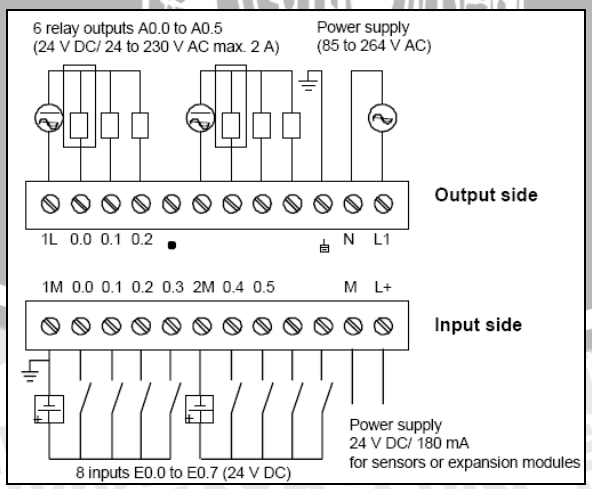


Gambar 2. 3 Diagram block prinsip kerja PLC
Sumber: A Beginners Guide to PLC, 1993:22

Peralatan input dapat berupa sakelar, tombol, sensor, dan peralatan lainnya. Data-data yang masuk dari peralatan input ini berupa sinyal-sinyal analog. Oleh modul input sinyal-sinyal yang masuk akan diubah menjadi sinyal-sinyal digital. Kemudian, oleh unit pemroses pusat atau "*Centrall Processing Unit*" (CPU) yang ada didalam PLC ditetapkan di dalam memorinya. Selanjutnya, CPU akan mengambil keputusan-keputusan tersebut kemudian dipindahkan ke modul output masih dalam bentuk digital. Oleh modul output sinyal-sinyal ini akan diubah kembali menjadi sinyal-sinyal analog. Sinyal-sinyal analog inilah yang nantinya akan menggerakkan peralatan output atau "*Output Device*" yang dapat berupa kontaktor-kontaktor ataupun relay-relay. "*Output Device*" inilah yang nantinya akan mengoperasikan sistem atau proses yang akan dikontrol.

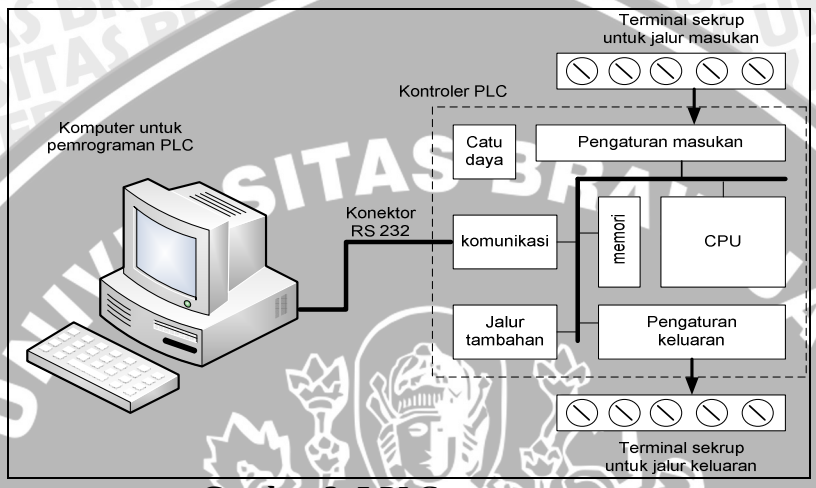
2.1.2 Pengawatan PLC dan Bagian-Bagian PLC

Pengawatan standar CPU PLC SIEMENS S7-200 dapat dilihat pada gambar berikut :



Gambar 2. 4 Pengawatan PLC
Sumber: SIEMENS Edition 7, 1999:9

Pada gambar diatas diperlihatkan bahwa pada busbar(jalur) input diberikan tegangan 24 V DC/180mA, dengan 6 relay output 24 V DC/ 24 sampai 230 V AC dengan arus maksimal 2 A. Tegangan input dan output tersebut dapat diganti besaran tegangannya atau jenis tegangannya (AC/DC). [SIM-99:9]



Gambar 2. 5 PLC secara umum
Sumber: Modul Laboratorium Kontrol, 2008:1

Bagian PLC pada prinsipnya tidak jauh berbeda dari perangkat keras yang dimiliki oleh komputer, yaitu terdiri atas *Central Processing Unit* atau CPU, *Programming Device*, Modul Input/Output, Unit Power Supply.

- ***Central Processing Unit* (CPU)**

  Central Processing Unit berfungsi untuk mengambil instruksi dari memori, mendekodenya kemudian mengeksekusi instruksi tersebut. Selama proses tersebut, CPU akan menghasilkan sinyal kontrol, memindahkan data ke I/O port atau sebaliknya, melakukan fungsi aritmatik dan logika juga mendeteksi sinyal dari luar CPU. CPU, pada umumnya terdiri atas 3 (tiga) unsur utama, yaitu *processor*, sistem memori dan catu daya. Arsitektur CPU dapat berbeda-beda untuk setiap merek, misalnya saja catu dayanya berada di luar CPU.

- **Unit Catu Daya**

Catu daya listrik digunakan untuk memberikan pasokan catu daya ke seluruh bagian PLC (termasuk CPU, memori, dan lain-lain). Kebanyakan PLC bekerja dengan catu daya 24V DC atau 220 AC. Ada beberapa PLC yang memiliki catu daya yang terpisah, biasanya PLC yang besar, sedangkan yang jenis PLC medium atau yang kecil catu dayanya menjadi satu.

- **Unit Pemrograman**

Unit pemrograman digunakan untuk memasukkan program yang dibutuhkan ke dalam memori. Program tersebut dibuat kemudian dipindahkan ke dalam unit memori PLC. Unit pemrograman dapat berbentuk perangkat genggam, panel meja (*desktop console*), atau sebuah komputer.

- **Unit Memori**

Memori digunakan oleh PLC untuk sistem control proses, menyimpan sistem operasi, dan menyimpan program yang harus dijalankan, penyimpanan dalam bentuk biner hasil terjemahan diagram tangga yang dibuat oleh pemrogram(pengguna). Memori pengguna dibagi menjadi beberapa blok yang memiliki fungsi khusus. Beberapa bagian memori digunakan untuk menyimpan status masukan dan keluaran. Sedangkan bagian lain dari memori digunakan untuk menyimpan isi variabel-variabel yang digunakan dalam pemrograman yang dituliskan, misalnya nilai Timer (pewaktu) atau nilai Counter (pencacah).

- **Unit Input**

Unit Input Digital

Modul antarmuka masukan ini berfungsi untuk mengkonversi atau mengubah sinyal-sinyal yang sesuai dengan tegangan kerja CPU, misalnya masukan dari sensor dengan tegangan kerja 24V DC harus dikonversikan menjadi tegangan 5V DC agar sesuai dengan tegangan kerja CPU. Input digital atau input diskrit hanya mengenal kondisi on atau off, ia hanya mempunyai dua kemungkinan kondisi yaitu “0” dan “1”.

Unit Input Analog

Unit input analog berfungsi untuk menangani sinyal analog dan mengkonversikannya ke bentuk digital dengan menggunakan konverter analog ke digital sehingga dapat diproses oleh prosesor. Kisaran input analog adalah sebagai berikut 0-10V DC, 0-10V AC, -10V DC hingga +10V DC, 4-20mA DC.

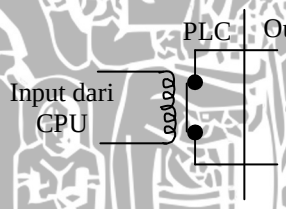
- **Unit Output**

Unit Output Digital

Output digital seringkali digolongkan ke dalam tipe relai, tipe transistor, dan tipe triac.

- Output tipe relai

Sinyal dari output PLC mengaktifkan sebuah relai sehingga mampu menyambungkan arus beberapa ampere ke rangkaian eksternal. Output tipe relai dapat menangani pensaklaran DC maupun AC. Output tipe relai ditunjukkan gada gambar.



Gambar 2. 6 Gambar Output tipe relai
Sumber: Modul Laboratorium Kontrol, 2008:3

- Output tipe transistor

Output tipe transistor menggunakan transistor sebagai sakelar untuk menyambung arus ke rangkaian eksternal. Waktu pensaklaran transistor lebih cepat dibandingkan dengan waktu pensaklaran dengan tipe relai, akan tetapi piranti ini hanya mampu menangani pensaklaran arus DC, dan akan rusak oleh arus lebih maupun tegangan balik yang tinggi.

- Output tipe triac

Output tipe triac digunakan untuk pensaklaran beban-beban eksternal yang disambungkan ke catu daya AC. Output ini hanya bisa digunakan untuk beban yang menggunakan catu daya AC.

Unit Output Analog

Unit output analog berfungsi untuk merubah sinyal digital dari CPU menjadi sinyal analog pada keluaran PLC. Prinsip kerja modul output analog berlawanan dengan prinsip kerja modul input analog. Kisaran output analog adalah 4 sampai 20mA, 0 sampai 5V DC, dan -10V sampai +10 V DC.

2.1.3 Dasar-Dasar Pemrograman PLC

Dasar-dasar dari pemrograman dari *Programmable Logic Controller* (PLC) dapat dilakukan langkah-langkah sebagai berikut :

1. Menentukan diskripsi kerja sistem yang akan dikontrol.
2. Menentukan peralatan input/output yang dipakai kedalam PLC I/O bit yaitu peralatan eksternal yang akan mengirim/menerima sinyal dari PLC.
3. Menentukan simbol-simbol ladder diagram untuk menggambarkan rangkaiannya.
4. Menggunakan program Micro/Win / LSS / programming console, untuk mengubah ladder diagram kedalam kode mnemonic agar CPU PLC dapat mengerjakannya.
5. Memindahkan program yang telah ditulis/ digambar kedalam memori PLC.
6. Memperbaiki kesalahan pemrograman jika terjadi kesalahan pada program yang telah dibuat, sehingga menjadi benar.
7. Menjalankan program pada PLC dan mengetes kesalahan program execution.

Hubungan kontak-kontak diagram tangga yang ada dalam CPU PLC terangkai secara elektronik, sehingga tidak memerlukan kawat penghubung seperti pada rangkaian kontrol secara konvensional. Adapun ketentuan-ketentuan dalam penyusunan rangkaian ke diagram tangga adalah sebagai berikut :

1. Pembuatan rangkaian kontrol diusahakan menggunakan kontak seminimum mungkin, sehingga efisiensi kerja dari PLC dapat ditingkatkan. Alamat-alamat serta data-data dalam register digunakan sehemat mungkin agar tidak melebihi kapasitas memori yang ditetapkan.
2. Kondisi sinyal yang mengalir pada rangkaian logika PLC selalu datang dari arah kiri menuju ke arah kanan.
3. Tidak ada satu koil atau relay output yang dapat dihubungkan langsung pada busbar bagian kiri. Jika diperlukan relay output bekerja terus menerus, maka di antara busbar kiri dengan output diberi kontak NC dari internal *Auxilary Relay* yang tidak digunakan.
4. Busbar sebelah kanan dari diagram tangga boleh tidak digambar, karena hubungan busbar tersebut telah tersambung secara otomatis pada PLC.
5. Semua output dilengkapi dengan kontak-kontak bantu yang dapat digunakan secara seri maupun paralel.
6. Jumlah kontak-kontak NO dan NC dapat dihubungkan secara seri maupun paralel dengan tak terbatas sesuai dengan kebutuhan.
7. Tidak ada kontak yang dapat diprogram atau disisipkan setelah output atau dengan kata lain antara busbar sebelah kanan dan hasil output tidak boleh disisipi kontak.
8. Pengkodean nomor-nomor kontak dan nomor-nomor koil output, termasuk *timer*, *counter* dan lain-lain disesuaikan dengan spesifikasi yang telah ditetapkan oleh pabriknya.
9. Sebuah output koil, termasuk *timer*, *counter* tidak dapat digunakan untuk lebih dari dari satu kali.
10. Dua atau lebih koil output, termasuk *timer*, *counter* dapat dihubungkan secara paralel.
11. Program rangkaian dieksekusi oleh CPU secara berurutan, mulai dari alamat yang pertama sampai dengan alamat yang terakhir pada program.

2.1.4 Menggambar Ladder Diagram

Dalam setiap bahasa pemrogramana kita mengenal unit terkecil yang dapat diproses yaitu yang biasa disebut bit, bit dapat diasumsikan sebagai dua keadaan:

1. ”1” memiliki arti ”bit set” atau keadaan ”benar”
2. ”0” memiliki arti ”bit not set” atau keadaan ”untrue”

Metode yang paling familiar bagi kita dalam mengartikan dua keadaan biner ”1” dan ”0” ialah dengan mengumpamakan keadaan tersebut sebagai suatu sirkuit elektronik yang umum disebut saklar. Keadaan pada saat saklar tertutup menyebabkan ada arus yang mengalir yang berarti bit dalam keadaan = ”1”. Pada saat saklar dalam keadaan terbuka (open) maka tidak ada arus yang mengalir sehingga mewakiki keadaan bit = ”0”

Representasi yang dapat dipahami dari aplikasi tersebut pada pembuatan Ladder diagram adalah kita mengenal dua keadaan logika:

- Aplikasi sebagai logika positif

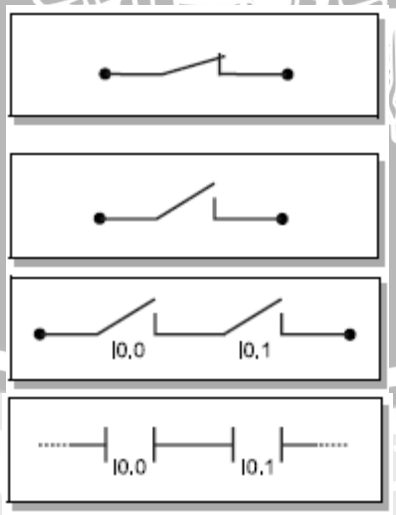
  24V = level-high = ”1”

  0 V = level-low = ”0”

- Aplikasi sebagai logika negatif

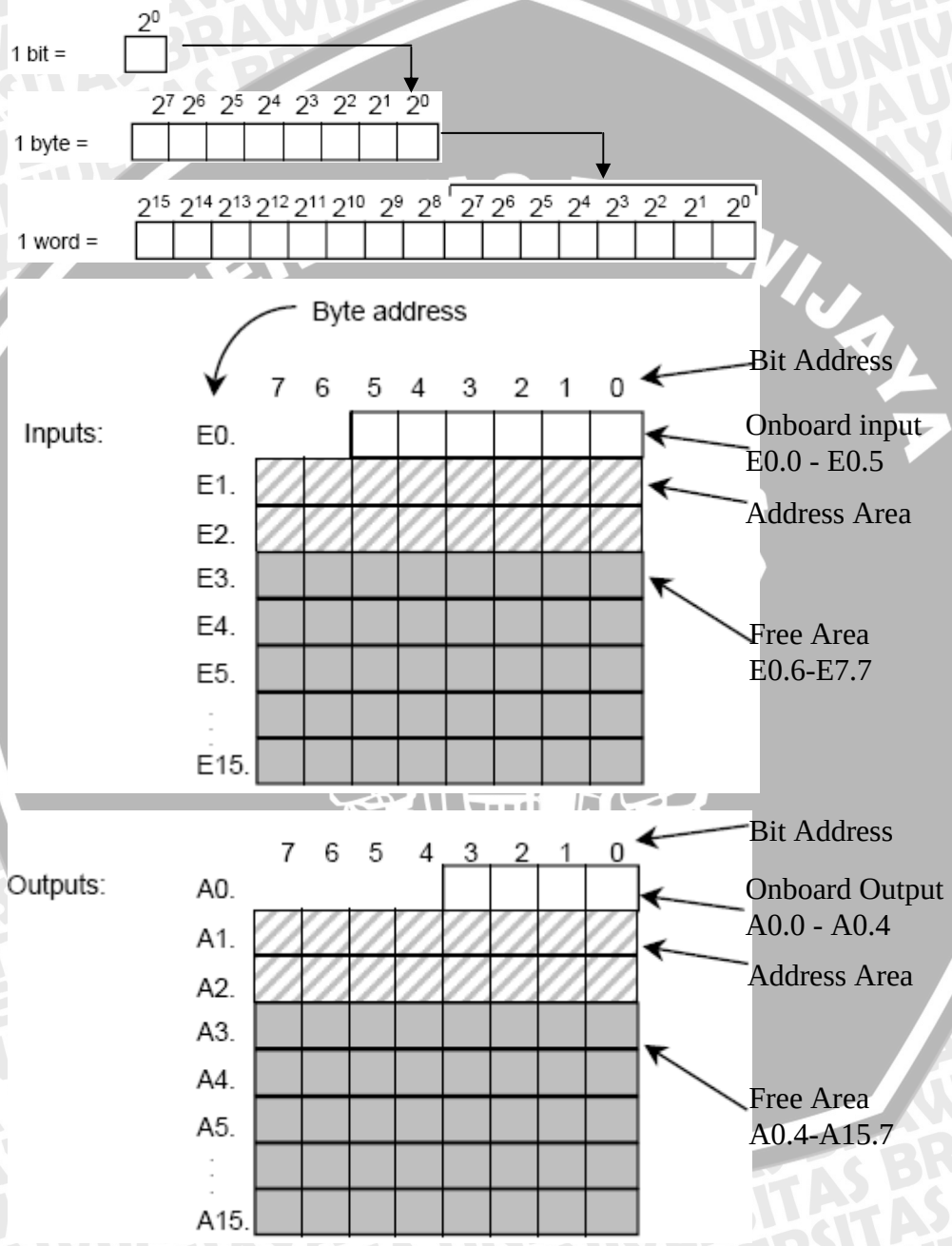
  0 V = level-high = ”1”

  24V = level-low = ”0”



Gambar 2. 7 Analogi Saklar
Sumber: SIEMENS Edition 1, 2000:6

Pada PLC, bits terorganisir ke dalam suatu group, group yang terdiri dari 8 bit disebut *byte*. Setiap bit didalam group menempati posisi pada alamatnya masing-masing. Setiap *byte* memiliki alamat *byte* dan bit alamat dari 0 sampai 7, gabungan dari dua *byte* disebut *word*. [SIM-99:49]

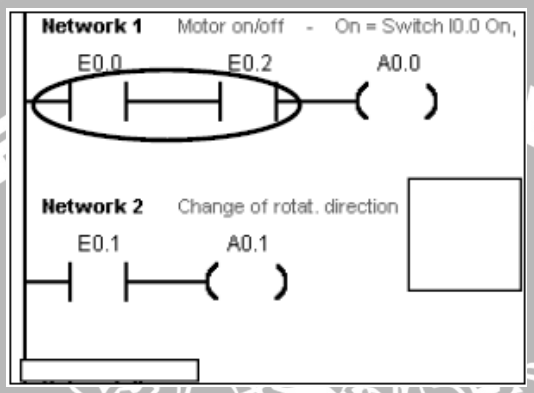


Gambar 2. 8 Area alamat pada PLC Siemens S7-200
Sumber: SIEMENS Edition 7, 1999:49

Pada pemrograman PLC terdapat komponen-komponen dasar yaitu :

- **Operasi AND**

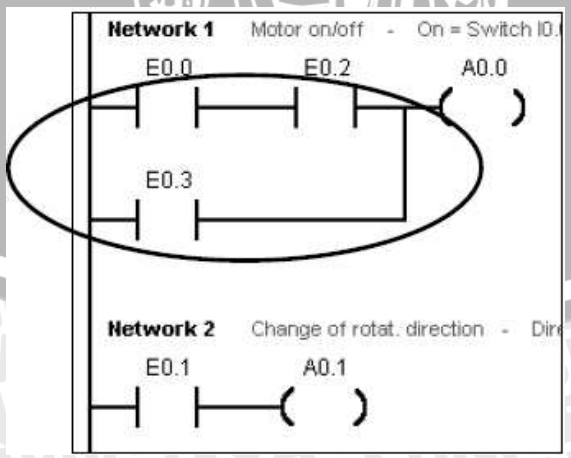
  Apabila dua buah atau lebih kontak NO (Normally Open) dihubungkan secara seri, akan dihasilkan operasi logika AND. Diagram rangkaian pada gambar menunjukkan koil A0.0 tidak akan aktif selama E0.0 dan E0.2 tidak dioperasikan secara bersamaan.



Gambar 2. 9 Ladder diagram operasi AND
Sumber: SIEMENS Edition 7, 1999:29

- **Operasi OR**

  Apabila dua atau lebih kontak NO (Normally open) dihubungkan secara paralel maka akan tercipta sebuah operasi logika OR. Mode operasi ini ditunjukkan pada gambar yang berarti saat (E0.0 AND E0.2) OR E0.3 tertutup maka aliran arus dari *power rail* akan mengalir ke koil A0.0. [SIM-99:33]



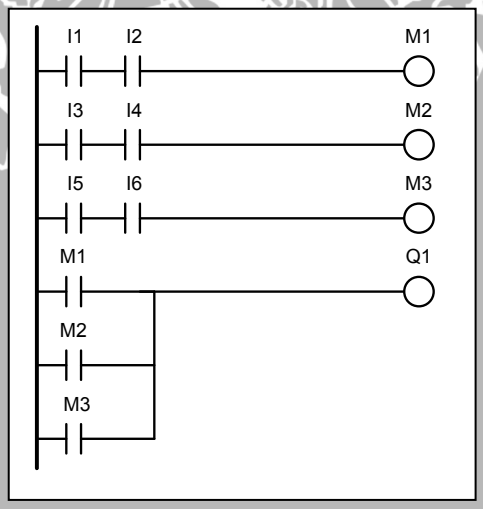
Gambar 2. 10 Ladder diagram operasi OR
Sumber: SIEMENS Edition 7, 1999:33

- **Operasi Gabungan**

  Jika instruksi AND dan OR digabung atau dikombinasikan dalam suatu diagram tangga yang lebih kompleks, maka bisa dipandang satu persatu, artinya bisa dilihat masing-masing hasil gabungan dua kondisi menggunakan instruksi AND atau OR secara sendiri-sendiri kemudian menggabungkannya menjadi satu kondisi menggunakan instruksi AND atau OR terakhir.

- **Marke**

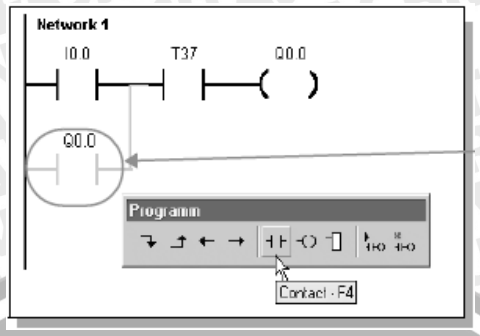
  Marke merupakan fasilitas khusus pada PLC yang digunakan untuk mengelompokkan program-program PLC menjadi beberapa subprogram. Pada pemrograman PLC, Marke dapat berfungsi sebagai masukan maupun keluaran. Operand yang mewakili Marke adalah M. Jumlah Marke yang dapat digunakan dalam sebuah PLC tergantung pada kemampuan masing-masing PLC.



Gambar 2. 11 Marke
Sumber: Modul Laboratorium Kontrol, 2008:11

- **Latching**

  Konsep latching (penguncian) dapat diwakili oleh susunan elemen logika standar yang keluarannya diumpan balikkan ke masukannya seperti terlihat pada gambar 2.12.



Gambar 2. 12 Latching
Sumber: SIEMENS Edition 1, 2000:13

Pada gambar 2.12 diatas menunjukkan adanya suatu hubungan OR. Dari sini dapat dipahami ketika sinyal masukan I0.0 berubah dari 0 ke 1, sinyal keluaran Q0.0 akan berkondisi 1, yang berarti siyal keluaran tidak akan pernah bernilai 0 kembali sekalipun sinyal masukan I0.0 bernilai 1 hanya dalam waktu singkat.

- **R-S Memori**

  Fungsi R-S merupakan element penyimpanan yang telah tersedia dalam pemrograman PLC.

Gambar 2. 13 R-S Memori
Sumber: Modul Laboratorium Kontrol, 2008:12

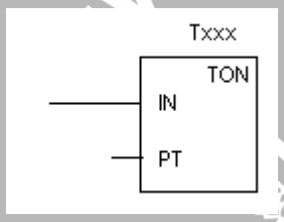
Masukan I_1 akan mengubah status bit operan menjadi ON, sedangkan masukan I_2 adalah sebaliknya. Oleh karena itu kedua masukan I_1 : masukan set (S) dan I_2 : masukan reset (R). Instruksi Set akan meng-ON-kan bit operan saat kondisi eksekusinya ON, namun tidak seperti *Latching*, Set tidak akan meng-OFF-kan bit operan saat kondisi eksekusinya menjadi OFF (setelah ON).

- **Timer**

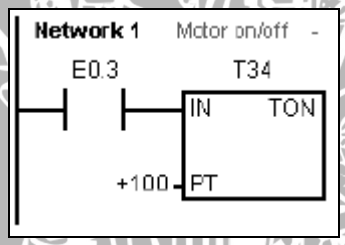
  Pada PLC Siemens S7-200 menyediakan tiga macam tipe Timer yaitu, On-Delay Timer (TON), Retentive On-Delay Timer (TONR) dan Off-Delay Timer (TOF) dengan penjelasan dan kegunaannya sebagai berikut:

- **On-Delay Timer**

  Timer merupakan instruksi penundaan waktu yang membutuhkan nomor timer dan nilai set (PT = present value) dimana nomor timer (T_{XXX}) dalam suatu program tidak boleh digunakan lebih dari satu kali. Pada siemens S7-200 ini memiliki 256 nomer timer yaitu T0 sampai T255 yang dibagi dalam beberapa timebase seperti terlihat pada tabel dibawah ini[SIM-99:36]

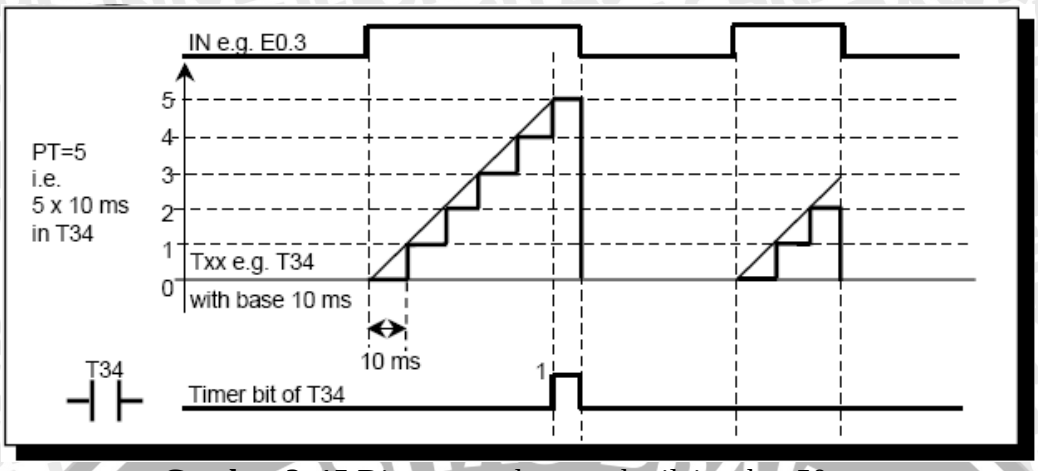


Txxx : nomor timer
timebase 1ms:T0,T32,T64,T96
timebase10ms:T1-T4,T33-T36,T65-T68,T97-T100
timebase 100ms:T5-T31,T37-T63, T69-T95, T101-T255
IN : start signal (high signal)
PT : present value



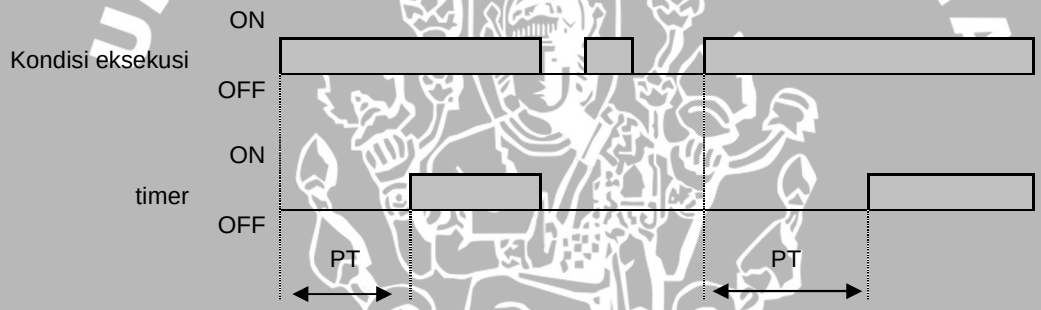
Gambar 2. 14 On-Delay Timer
Sumber: SIEMENS Edition 7, 1999:36

Setiap timer pada S7-200 memiliki status bit (waktu terpenuhi/tidak terpenuhi) bit ini disebut bit timer. Pada contoh pada gambar 2.14 dapat kita pahami bahwa sebelum bit timer T34 bernilai ”1”, kondisi masukan E0.3 harus bernilai ”1” yang berarti 1s. Apabila waktu delay belum terpenuhi pada saat sinyal enable ”IN” dibatalkan maka timer set pada keadaan ”0” sehingga bit timer juga dalam keadaan tidak set (lihat diagram waktu pada gambar). Waktu tunda diperoleh dari perkalian antara faktor PT dengan timebase dari timer dalam hal ini PT = 100 sedangkan T34 memiliki timebase 10ms, jadi 100 x 10ms = 1s. [SIM-99:36]



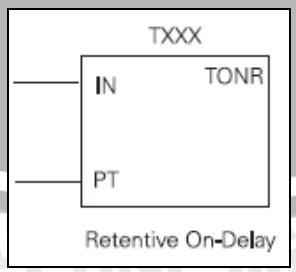
Gambar 2. 15 Diagram waktu untuk nilai waktu 50 ms
Sumber: SIEMENS Edition 7, 1999:36

Prinsip kerja sebuah timer dapat diilustrasikan oleh gambar berikut ini

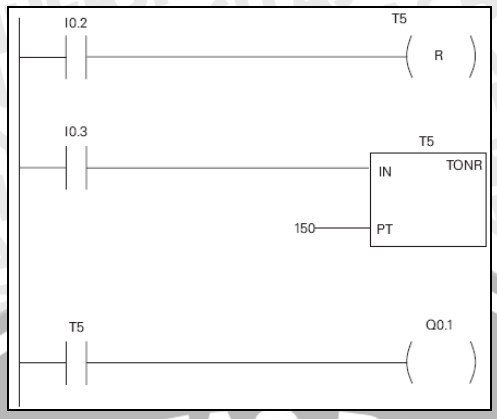


- **Retentive ON-Delay Timer**

  Pada Retentive On-Delay Timer (TONR) sebenarnya memiliki cara kerja yang sama dengan On-Delay Timer (TON), perbedaannya hanya terletak pada saat kondisi sinyal masukan *enable* ”IN” tidak aktif/off kondisi keluaran TONR tidak akan reset sebelum timer direset dengan instruksi RESET (R).



Gambar 2. 16 Retentive On-Delay Timer
Sumber: SIEMENS Edition 7, 1999:36

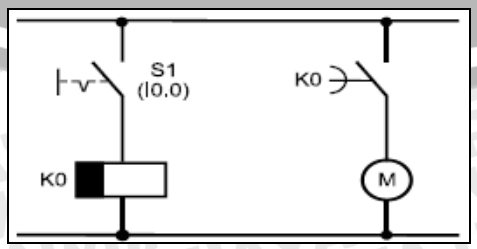


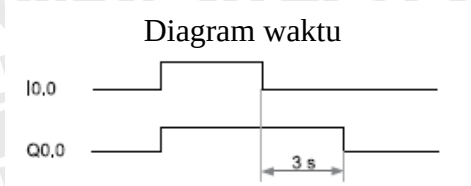
Gambar 2. 17 Contoh aplikasi TONR
Sumber: SIEMENS Edition 7, 1999:36

Untuk lebih memahami prinsip kerja TONR kita perhatikan contoh kasus pada gambar 2.17. pada saat I0.3 tertutup Timer mulai aktif, sebagai contoh bila setelah 10 detik I0.3 dimatikan maka timer berhenti aktif, pada saat I0.3 aktif kembali timer akan meneruskan hitungannya dimulai dari 10 detik, lampu akan menyala selama 5 detik dari saat I0.3 tertutup untuk yang kedua kali. Apabila setelah detik ke 10 *pushbutton* yang terhubung dengan I0.2 ditekan maka Timer akan reset dan akan menghitung mulai dari 0 kembali bila I0.3 dihubungkan.

- **Off-Delay Timer**

Kita telah mengetahui bersama dengan apa yang disebut On-Delay Timer, tetapi bagaimana halnya dengan Off-Delay Timer. Untuk lebih mudahnya kita memahami tentang hal ini ada baiknya kita mengambil sebuah contoh kasus seperti terlihat pada gambar 2.18 pada saat S1 (I0.0) beroperasi motor kipas pada output Q0.0 diaktifkan. Apabila S1 (I0.0) dimatikan, kipas masih terus menyala selama 3 detik baru kemudian berhenti.



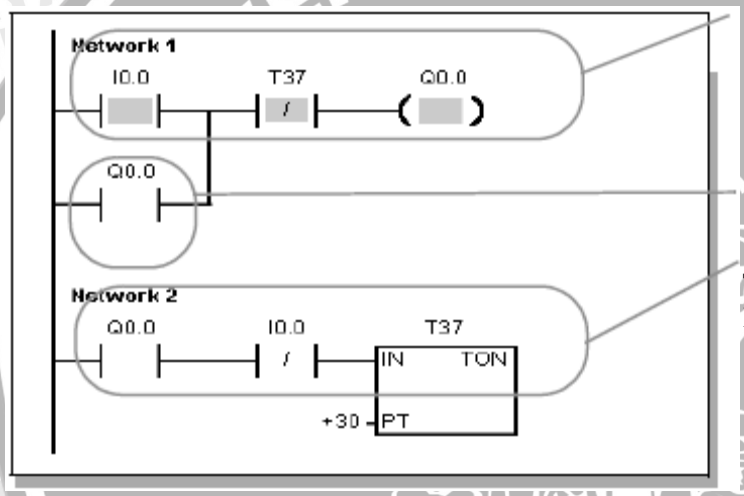


Gambar 2. 18 Contoh kasus Off-Delay Timer
Sumber: SIEMENS Edition 1, 2000:29

Dari contoh kasus diatas kita memperoleh solusi sebagai berikut:

Phase 1:

Pengaktifan latching circuit, I0.0 berlogika ”1”, pada phase ini kita asumsikan Q0.0 tidak aktif.

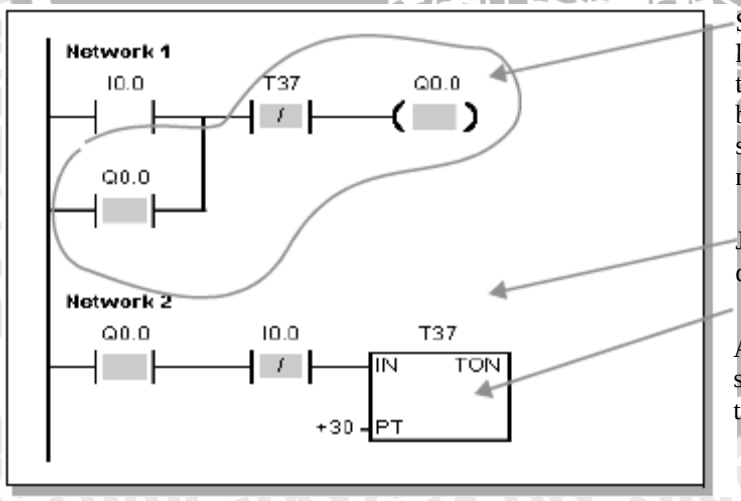


Apabila I0.0 beroperasi AND T37 belum terpenuhi maka Q0.0 diaktifkan (=”1”)

Q0.0 terkunci (latching)

T37 belum bekerja karena I0.0 masih bernilai “1”

Phase 2:

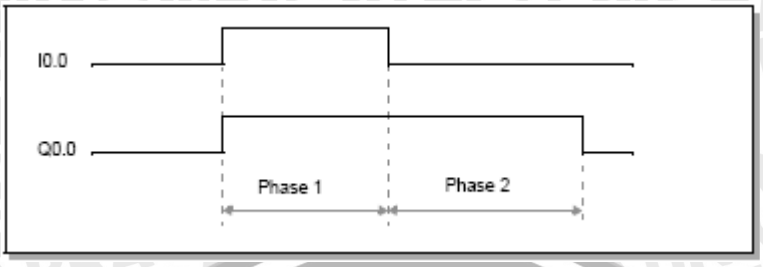


Setelah I0.0 tidak beroperasi latch tetap bekerja hingga T37 terpenuhi. Sementara timer bekerja T37 kondisi “0” sehingga kontak NC melewatkan arus

Jalannya timer dapat diamati degan test mode

Apabila Q0.0 aktif AND I0.0 sudah tidak beroperasi, maka timer bekerja

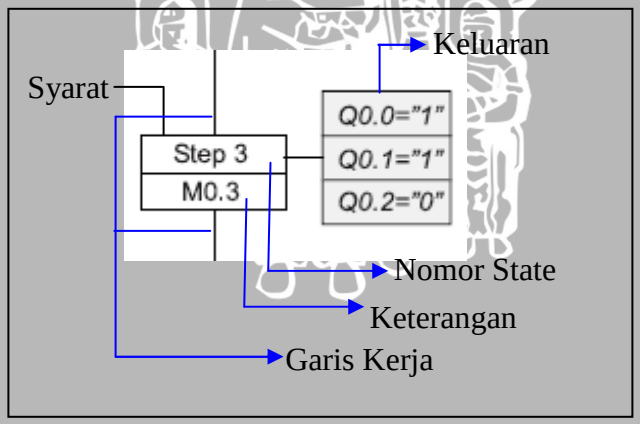
Sehingga diperoleh hasil keluaran seperti yang diharapkan hal ini ditunjukkan pada gambar 2.19



Gambar 2. 19 Keluaran Off-Delay Timer
Sumber: SIEMENS Edition 1, 2000:35

- **Diagram State**

Untuk memudahkan dalam pembuatan program, maka sebelum mengaplikasikannya ke dalam Ladder diagram kita dapat terlebih dahulu menyusun diagram state-nya. Diagram state disimbolkan dengan sebuah kotak persegi panjang yang diberi garis lurus untuk memisahkan antara nomor state dan keterangan. Setiap state memiliki dua garis kerja yang menyatakan aliran kerja, sebelum state dan sesudah state tersebut. Setiap state dimungkinkan dilengkapi dengan keluaran dan syarat agar state tersebut dapat dieksekusi atau dilaksanakan. Bila dilihat dari gambar, state no 3 akan dilaksanakan apabila syarat dipenuhi dan state sebelumnya telah dilaksanakan.

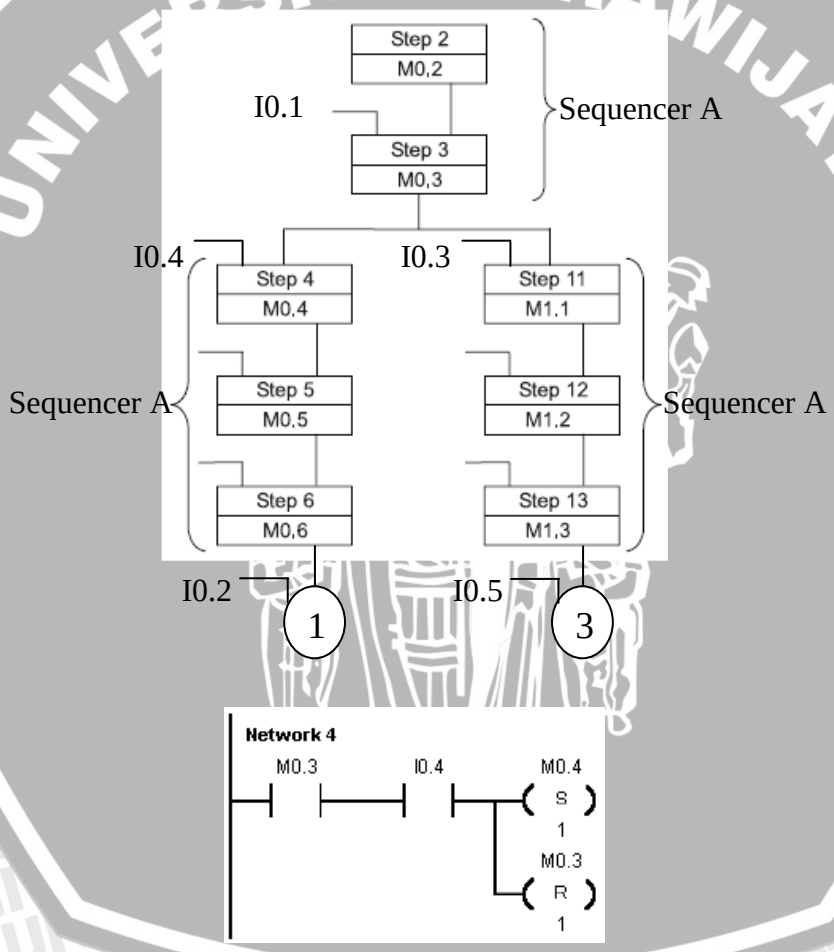


Gambar 2. 20 diagram state
Sumber: Siemens,2000:43

Pada kotak keluaran dimungkinkan terdapat beberapa prasyarat yang harus dipenuhi. Penulisan prasyarat itu diletakkan di atas kanan kotak keluaran. Pada diagram state dimungkinkan pula memiliki state percabangan. Pada gambar dapat kita

lihat apabila M0.3 berlogika 1 maka dua sequencer A dan B akan dimulai, apabila syarat I0.4 dipenuhi maka M0.4 yang akan dieksekusi, apabila syarat I0.3 yang dipenuhi maka M1.1 yang akan dieksekusi.

Apabila state berikutnya terletak diatas state yang bersangkutan maka proses berikutnya dituliskan dengan lingkaran kecil yang dilengkapi dengan persyaratannya. Pada contoh pada gambar menunjukkan bahwa setelah state 6 terpenuhi, state 1 akan dieksekusi apabila syarat I0.2 terpenuhi. Demikian pula yang terjadi pada state 13, setelah state 13 terpenuhi state 3 akan dieksekusi apabila syarat I0.5 dipenuhi.



Gambar 2. 21 Contoh aplikasi diagram state
Sumber: Siemens,2000:45

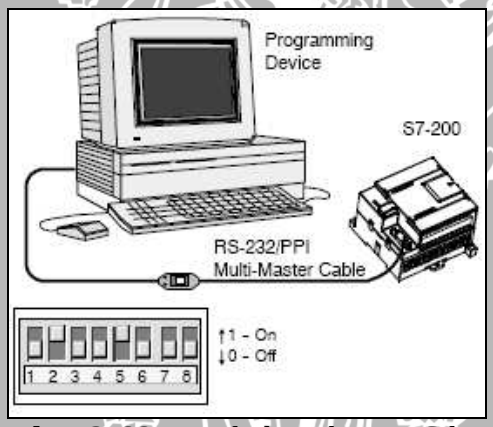
Pada perubahan diagram state ke program PLC SIEMENS setiap state dinyatakan sebagai marke, setiap state diwakilkan dengan komponen set reset.

2.2 Penggunaan Program Micro/WIN

Micro/WIN adalah sebuah paket program yang memberikan fasilitas yang memudahkan untuk menuliskan, memperbaiki dan mengawasi program ladder pada sistem operasi windows, Micro/WIN menyediakan tiga program editor yang sangat memudahkan serta efisien dalam mengembangkan program aplikasi yang kita buat.

2.2.1 Menghubungkan dengan PC

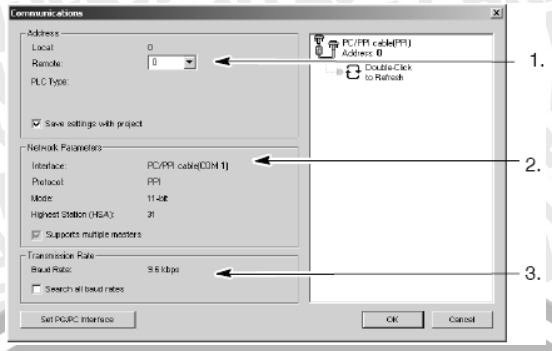
Siemens menyediakan dua menu pilihan untuk menghubungkan PLC SIEMENS S7-200 dengan komputer: yaitu koneksi secara langsung menggunakan kabel PPI Multi-Master dalam hal ini RS-232, atau dengan cara komunikasi Processor card menggunkan kabel MPI. Untuk menghubungkan RS-232 ke PC kita tinggal menghubungkan konektor RS-232 ke communications ports (sebagai contoh, hubungkan ke COM 1.), dengan memastikan switch DIP di set seperti yang ditunjukkan pada Gambar 2.22. [SIM-03:07]



Gambar 2. 22 menghubungkan PLC ke PC
Sumber: Siemens,2003:7

Untuk verifikasi parameter komunikasi gunakan default settings dengan langkah sebagai berikut:

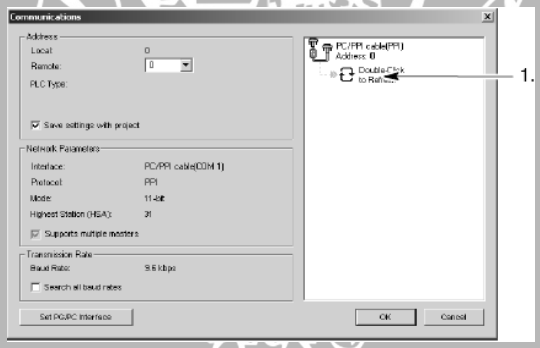
1. Pastikan alamat (address) kabel PC/PPI pada kotak dialog *communications* diset ke 0.
2. Pastikan interface untuk *network parameter* diset pada kabel PC/PPI(COM1).
3. Pastikan transmission rate di set pada 9.6 kbps.



Gambar 2.23 Verifikasi parameter komunikasi program Micro/WIN
Sumber: Siemens,2003:8

Untuk menghubungkan CPU anda dengan S7-200 kita dapat menggunakan kotak dialog *communications* dengan langkah-langkah sebagai berikut:

1. Klik dobel pada ikon *refresh* pada kotak dialog *communications,* Micro/WIN akan mencari perangkat S7-200 dan menampilkannya pada layar ikon untuk menghubungkan CPU dengan perangkat S7-200
2. Pilih S7-200 kemudian klik OK.



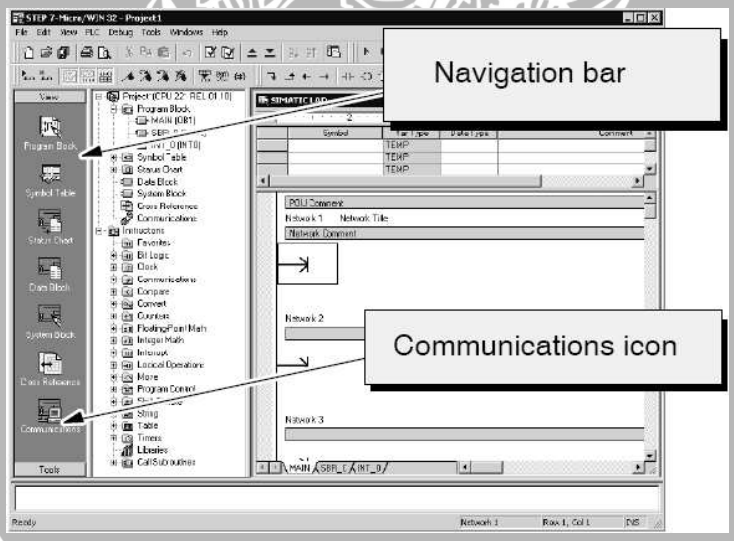
Gambar 2. 24 Membangun komunikasi dengan S7-200
Sumber: Siemens,2003:8

Apabila Micro/WIN tidak menemukan perangkat S7-200 CPU, maka cek kembali pengaturan pada *communications parameters* dan ulangi kembali langkah diatas. Setelah perangkat S7-200 telah terhubung, anda sudah bisa membuat dan download program.

2.2.2 Memulai Program Micro/WIN

Sebelum dapat digunakan terlebih dahulu kita harus menginstal Program kedalam komputer yaitu dengan langkah-langkah masukkan CD instaler Micro/WIN ke dalam drive CD-ROM pada komputer anda. Program penginstal (installation wizard) akan run secara otomatis dan akan memberikan petunjuk kepada anda untuk proses instalasi selanjutnya. Ada baiknya sebelum anda menginstal Micro/WIN terlebih dahulu anda membaca *readme file* untuk mendapatkan informasi yang lebih jelas. [SIM-03:07]

Untuk memulai proses pembuatan program pada Micro/WIN klik ikon *new project* seperti terlihat pada Gambar 2.25 perhatikan *navigation bar,* anda dapat menggunakan ikon-ikon yang terdapat pada navigation bar untuk membuka element dari program Micro/WIN, sebagai contoh klik pada ikon *communications* anda dapat menggunakan kotak dialog tersebut untuk mengatur komunikasi data pada Micro/WIN.

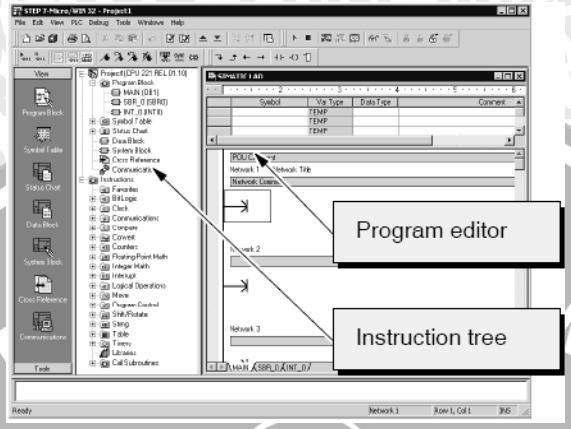


Gambar 2. 25 Tampilan program Micro/WIN
Sumber: Siemens,2003:7

- Membuka Program Editor

Klik pada ikon program block untuk membuka program editor, dapat dilihat pada gambar 2.26. Perhatikan *instruction tree* dan *program editor*. Anda dapat

menggunakan pohon instruksi (*instruction tree*) untuk memasukkan Ladder ke dalam *networks* dari program editor dengan cara *dragging* dan *dropping* instruksi.



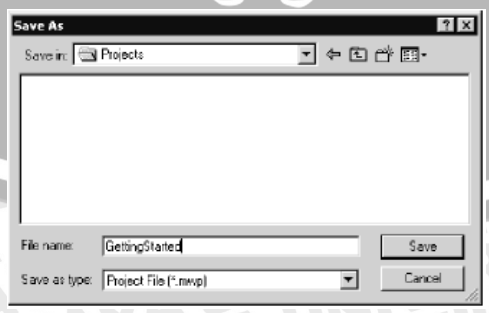
Gambar 2.26 Membuka program editor pada Micro/WIN
Sumber: Siemens,2003:9

- Menyimpan Program

Setelah selesai membuat program dengan cara memasukkan instruksi-instruksi ke dalam *networks*, anda dapat menyimpan program tersebut dengan langkah sebagai berikut:

1. Pilih menu perintah **File > Save As** dari menu bar
2. Masukkan judul dari program anda pada kotak dialog Save As
3. Klik OK untuk menyimpan program

Setelah proses penyimpanan selesai anda dapat mendownload program tersebut ke dalam PLC S7-200

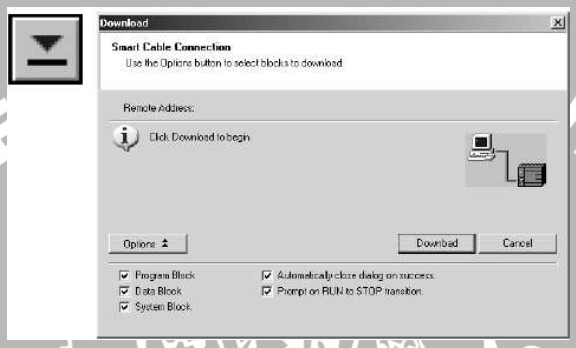


Gambar 2.27 Menyimpan program pada Micro/WIN
Sumber: Siemens,2003:11

- Download Program

Untuk mendownload program PLC yang telah ada kita dapat mengikuti prosedur sebagai berikut:

1. Klik ikon *download* pada *toolbar* atau pilih dari menu perintah **File > Download**, seperti ditunjukkan pada Gambar2.28.
2. Klik OK untuk download elemen program ke dalam PLC S7-200.



Gambar 2.28 Download program pada Micro/WIN
Sumber: Siemens,2003:12

2.2.3 Menempatkan PLC S7-200 ke dalam Mode RUN

Cara menempatkan S7-200 CPU ke dalam mode RUN adalah dengan terlebih dahulu switch mode pada S7-200 harus di set ke TERM atau RUN. Apabila S7-200 sudah berada dalam mode RUN maka program akan di eksekusi. [SIM-03:12] Langkah-langkahnya dapat ditunjukkan sebagai berikut:

1. Klik ikon RUN pada toolbar atau pilih pada menu perintah **PLC > RUN.**
2. Klik OK untuk mengganti mode operasi dari S7-200

Gambar 2.29 Menempatkan S7-200 ke mode RUN
Sumber: Siemens,2003:12

Pada saat S7-200 berada pada mode RUN, LED output untuk Q0.0 akan menyala dan mati sesuai dengan eksekusi program yang dilakukan oleh PLC. Kita dapat mengamati program dengan memilih **Debug > Program Status** pada menu perintah. Micro/WIN akan menampilkan nilai dari instruksi. Untuk menghentikan program kembalikan S7-200 ke dalam mode STOP dengan mengklik ikon STOP atau memilih **PLC > STOP** pada menu perintah. [SIM-03:12]

Untuk keluar dari program Micro/WIN adalah klik File pada menu windows Micro/WIN kemudian Exit.

2.3 Teknik Pulse Width Modulation

Pada dasarnya putaran motor DC secara umum berbanding lurus dengan tegangan supplay pada terminalnya. Untuk mendapatkan putaran rendah maka diberi tegangan rendah. Untuk putaran tinggi maka tegangan harus tinggi. Dengan demikian, masalah yang harus diselesaikan pada rangkaian pengemudi motor adalah bagaimana membuat tegangan output dapat bervariasi (dapat diatur mulai dari 0 volt hingga tegangan maksimum secara linier). [PIT-06:88]

Secara teori, spesifikasi ini dapat diperoleh dengan memanfaatkan rangkaian penguat transistor yang tegangan / arus basisnya dapat diatur untuk mendapatkan tegangan kolektor yang variatif. Akan tetapi, cara ini tidak disarankan, karena dapat menimbulkan panas yang berlebihan pada transitor. Hal ini disebabkan transistor bekerja pada daerah linier sehingga disipasi daya berupa panas yang setara dengan hasil perkalian arus kolektor dengan resistansi kolektor emitor adalah relatif besar. Seperti yang diketahui, resistansi kolektor emitor akan mendekati tak terhingga (atau hubungan terbuka) bila transistor berada dalam kondisi cut-off, dan resistansi menjadi minimum bila transistor berada dalam kondisi saturasi. Jika cut-off maka disipasi daya adalah mendekati nol sehingga tidak terjadi panas, dan jika saturasi maka resistansi mendekati nol sehingga disipasi daya pada sisi transistor juga mendekati nol sehingga panas juga tidak terjadi.[PIT-06:89]

Pulse Width Modulation (PWM) adalah suatu teknik manipulasi dalam pengemudian motor (atau perangkat elektronik berarus besar lainnya) yang

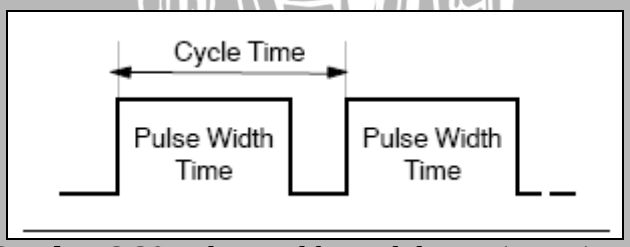
menggunakan prinsip cut-off dan saturasi. Transistor atau komponen switching didisain bekerja dengan karakteristik mirip ”linier” namun sebenarnya menggunakan teknik ON-OFF.

Lebar pulsa PWM dinyatakan dalam Duty Cycle. Misal duty cycle 10 %, berarti lebar pulsa adalah 1/10 bagian dari satu periode penuh, makin sempit pulsa PWM, tegangan ekivalen liniernya makin kecil. Jika duty-cyle 100% maka tegangan ekivalen linier sama dengan tegangan maksimum pada motor dikurangi tegangan saturasi pada kolektor-emitor (atau drain-source pada komponen MOSFET).

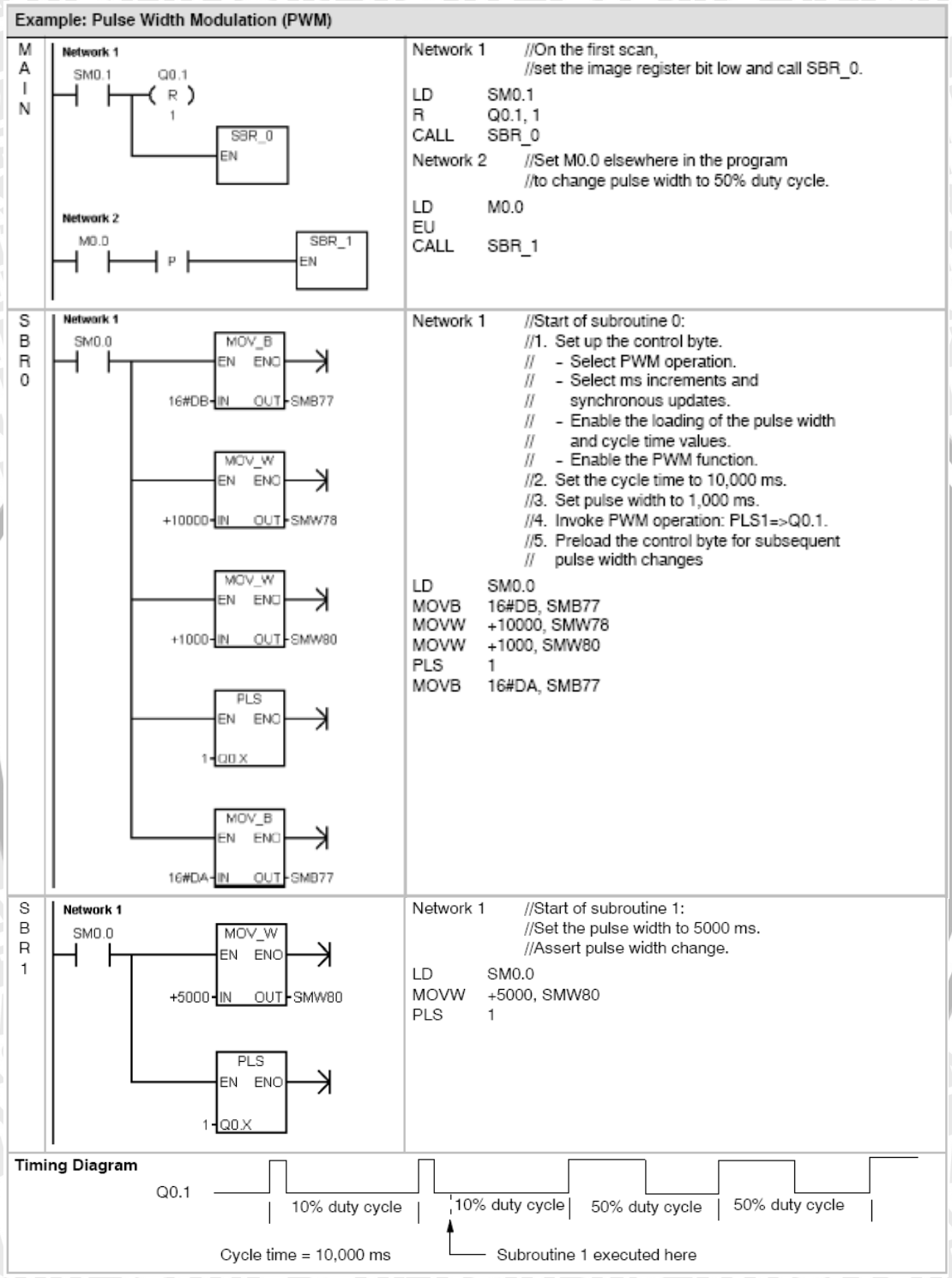
Dalam aplikasi untuk driver motor DC- MP secara umum, ferkuensi PWM dapat ditentukan mulai dari 60 hingga 2000 Hz sesuai dengan kemampuan switching komponen. Pembangkitan pulsa PWM dapat diperoleh melalui berbagai rangkaian timer yang bisa didapat di pasaran. [PIT-06:90]

Fasilitas PWM ini pada PLC SIEMENS S7-200 disediakan pada fungsi *high-speed outputs* (Q0.0 dan Q0.1). PWM dapat menyediakan berupa output dengan cycle time tetap dan duty cycle yang bisa diubah-ubah, anda dapat mengatur cycle time dan pulse width dengan memilih salah satu dari dua ordo yaitu microsecond atau millisecond.

- Cycle time: 50 μs sampai 65.535 μs atau 2 ms sampai 65.535 ms
- Pulse width time: 0 μs sampai 65.535 μs atau 0 ms sampai 65.535 ms



Gambar 2.30 Pulse Width Modulation (PWM)
Sumber: Siemens,2003:128



Gambar 2.31 Contoh Ladder (PWM)
Sumber: Siemens,2003:134

2.4 Robot

Robot adalah sebuah alat mekanik yang dapat melakukan tugas fisik, baik menggunakan pengawasan dan dikontrol manusia, ataupun menggunakan program yang telah didefinisikan terlebih dulu (kecerdasan buatan).

2.4.1 Sejarah Robot

Kata robot berasal dari kata dalam bahasa Czech, *robota*, yang berarti pekerja, mulai populer ketika seorang penulis berkebangsaan Chzech (Ceko), Karl Capek, membuat pertunjukan dari lakon komedi yang ditulisnya pada tahun 1921 yang berjudul RUR (Rossum's Universal Robot). Ia bercerita tentang mesin yang menyerupai manusia tapi mampu bekerja terus-menerus tanpa lelah. Gaung popularitas istilah robot ini kemudian memperoleh sambutan dengan diperkenalkannya robot Jerman dalam film Metropolis tahun 1926 yang sempat dipamerkan dalam New York World's Fair 1939. film ini mengisahkan tentang robot berjalan mirip manusia beserta hewan peliharaannya. Kembali atas jasa insan film, istilah robot ini makin populer dengan lahirnya robot C3PO dalam film Star Wars pertama pada tahun 1977. [PIT-06:01] Adapun robot yang pertama kali dibuat, besar kemungkinan adalah *clepsydra* (jam air). Ctesibius, seorang fisikawan Yunani, dikabarkan telah bisa membuat jam seperti ini pada tahun 250 sebelum Masehi. Berabad-abad kemudian Pierre dan Henri-Louis Jacquet-Droz serta Henri Maillardet membuat otomaton (boneka peniru tingkah laku makhluk hidup) yang dapat menulis, menggambar, dan memainkan alat musik.

Selama ini robot digunakan untuk tugas-tugas yang dirasa berat, berbahaya, atau pekerjaan yang dilakukan berulang. Sehingga sering kita temui robot digunakan dalam bidang produksi yang membutuhkan ketelitian dan terjadi secara berulang seperti pada industri perakitan. Penggunaan lainnya adalah untuk menangani material radioaktif atau limbah berbahaya, penjelajahan bawah air, penjelajahan luar angkasa, pertambangan, pekerjaan penyelamatan, dan untuk pencarian tambang. Adapun kecenderungan yang terjadi belakangan ini robot mulai memasuki pasaran konsumen di bidang hiburan, dan peralatan rumah tangga.

Menurut Fu, et al. (1987) penelitian dan pengembangan pertama yang berbuah produk robotik dapat dilacak mulai dari tahun 1940-an ketika Argonne National Laboratories di Oak Ridge, Amerika, memperkenalkan sebuah mekanisme robotik yang dinamai master-slave manipulator. Robot ini digunakan untuk menangani material radioaktif. Kemudian produk robot komersial pertama diperkenalkan oleh Unimation Incorporated, Amerika, pada tahun 1950-an. Hingga belasan tahun kemudian langkah komersial ini telah diikuti oleh perusahaan-perusahaan di belahan dunia lain. [PIT-06:02]

2.4.2 Perkembangan Robot (sekarang)

Pada awalnya, aplikasi robot hampir tak dapat dipisahkan dengan industri sehingga muncul istilah *industrial robot* dan *robot manipulator*, Definisi yang populer ketika itu. robot industri adalah suatu robot tangan (robot arm) yang diciptakan untuk berbagai keperluan dalam meningkatkan produksi, memiliki bentuk lengan-lengan kaku yang terhubung secara seri dan memiliki sendi yang dapat bergerak berputar (rotasi) atau memanjang/memendek (translasi atau prismatik). Dewasa ini mungkin definisi robot industri itu sudah tidak sesuai lagi karena teknologi mobile robot juga sudah dipakai meluas sejak awal tahun 80-an. Seiring itu pula kemudian muncul istilah robot humanoid (konstruksi mirip manusia), animaloid (mirip binatang), dan sebagainya. [PIT-06]

Ketika para pencipta robot pertama kali mencoba meniru manusia dan hewan, mereka menemukan bahwa hal tersebut sangatlah sulit, membutuhkan sumberdaya yang jauh lebih banyak dari yang ada pada masa itu. Jadi, penekanan perkembangan diubah ke bidang riset lainnya. Robot sederhana beroda digunakan untuk melakukan eksperimen dalam tingkah laku, navigasi, dan perencanaan jalur. Teknik navigasi tersebut telah berkembang menjadi sistem kontrol robot otonom yang tersedia secara komersial, contoh paling mutakhir dari sistem kontrol navigasi otonom yang tersedia sekarang ini adalah sistem navigasi berdasarkan-laser dan VSLAM (Visual Simultaneous Localization and Mapping) yang dikembangkan oleh ActivMedia Robotics dan Evolution Robotics.

Ketika para teknisi siap untuk mencoba robot berjalan kembali, mereka mulai dengan heksapoda dan platform berkaki banyak lainnya. Robot-robot tersebut meniru serangga dan arthropoda dalam bentuk dan fungsi. Dengan lebih dari empat kaki, robot-robot ini stabil secara statis yang membuat mereka bekerja lebih mudah. Tujuan dari riset robot berkaki dua adalah mencapai gerakan berjalan menggunakan gerakan pasif-dinamik yang meniru gerakan manusia.

Dengan didukung perkembangan dunia ilmu pengetahuan yaitu dengan dikenalnya cabang disiplin ilmu baru berupa *virtual reality*. Sebelum suatu robot betul-betul diputuskan untuk dibuat, pakar robot cukup mencobanya dahulu secara virtual, secara imajinasi. Baik dalam hal bentuk fisik maupun dalam hal pergerakan yang sesungguhnya. Bahkan apakah program kemudi yang dikembangkan dapat berfungsi dengan baik dalam mengendalikan robot itu, cukup diujicoba secara virtual dulu. Hal ini erat kaitannya dengan kecanggihan komputer-komputer era baru dan teknologi pemrograman yang terus-menerus dikembangkan. [PIT-06:03]

2.4.3 Jenis-jenis Robot

• Humanoid

Robot yang memiliki fizikal asas manusia yaitu kepala, sepasang kaki dan tangan, sebagai contoh robot Asimo ciptaan Honda.

• Android

Robot yang memiliki sifat dan persamaan anggota tubuh menyerupai lelaki tetapi tidak kelihatan hidup.

• Cyborg

Robot yang tidak menyerupai manusia, tetapi mempunyai ciri fizikal dan memiliki fungsi manusia.

• Gynoid

Berasal dari perkataan ‘Gyne’ (Greek) yang berarti wanita, robot yang memiliki sifat dan persamaan anggota badan menyerupai wanita.

• A.I (Artificial Intelligent)

Robot yang mempunyai perasaan seperti manusia atau hewan.

• Disabled Robotik

Robot yang dicipta untuk membantu pergerakan orang yang cacat fisik. Lebih kepada bentuk penggunaan mesin beroda.

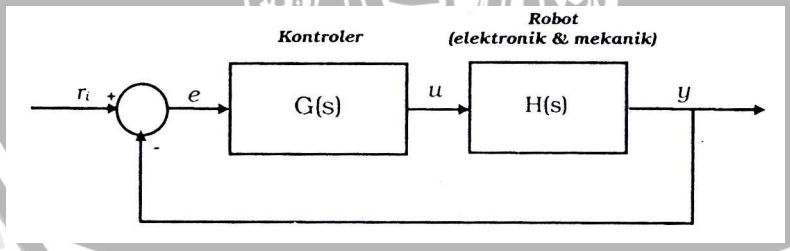
• Domotics

Robot yang sering digunakan dalam sektor perindustrian. Dikendalikan oleh seperangkat komputer.

2.4.4 Differentially Driven Mobile Robot

Sebuah robot dapat dianalisa dalam dua domain kajian, yaitu analisa kinematik dan analisa dinamik. Dalam skripsi ini hanya dilakukan analisa kinematik, sedangkan untuk analisa dinamik hanya dibahas sebatas rumusan matematisnya saja. hal ini dikarenakan kepresisian gerakan robot bukan merupakan tujuan utama. Yang lebih diutamakan, robot dapat dikontrol (*controlable*) selama dan sepanjang referensi trajektori yang diberikan. Maka hal-hal yang berkaitan dengan unsur dinamik yang dipengaruhi faktor-faktor seperti gravitasi, backslash pada gearbox, sampai dengan interaksi kopel antar bagian robot diabaikan. Analisa kinematik hanya berkaitan dengan gerakan robot tanpa memandang efek inersia / kelembaman yang terjadi ketika robot melakukan gerakan.

Sistem robotik secara garis besar terdiri dari sistem kontroler, elektronik dan mekanik robot. Dalam bentuk diagram dapat dinyatakan seperti pada gambar berikut ini.

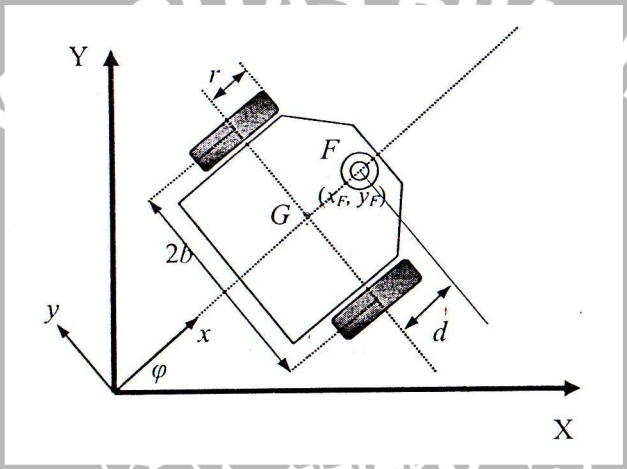


Gambar 2.32 Diagram Sistem Robotik
Sumber: Robotika, Buku I : 190

G(s) adalah persamaan matematik kontroler, sedang H(s) adalah persamaan untuk sistem robot secara fisik termasuk aktuator dan sistem elektroniknya.

Komponen r_i adalah referensi input yang dalam aplikasinya dapat berupa referensi posisi, kecepatan dan akselerasi. Dalam fungsi waktu, nilai input ini dapat bervariasi dan kontinyu yang membentuk suatu konfigurasi trajektori. Komponen *e* adalah error dan komponen *u* adalah output dari kontroler. Output *y* adalah fungsi gerak robot yang diharapkan selalu sama dengan referensi (gerak) yang didefinisikan pada input r_i

Pada skripsi ini Mobile robot yang dimaksud ialah mobile robot berpenggerak dua roda kiri-kanan yang dikemudikan terpisah (differentially driven mobile robot, disingkat DDMR), seperti yang ditunjukkan pada Gambar 2.33 berikut ini:



Gambar 2.33 DDMR pada medan 2D cartesian
Sumber: Robotika, Buku I : 224

Robot diasumsikan berada dalam kawasan 2D pada koordinat cartesian XY. Parameter-parameter dalam gambar adalah:

φ sudut arah hadap robot
2b lebar robot yang diukur dari garis tengah roda ke roda
r jari-jari roda (roda kiri dan kanan adalah sama dan sebangun)
d jarak antara titik tengah antara 2 roda, G dengan titik acuan F
(x,y) koordinat acuan di tubuh robot terhadap sumbu XY

Dalam kajian kinematik ini robot diasumsikan bergerak relatif pelan dan roda tidak slip terhadap jalan. Maka komponen x dan y dapat diekspresikan dalam suatu persamaan nonholonomic sebagai berikut,

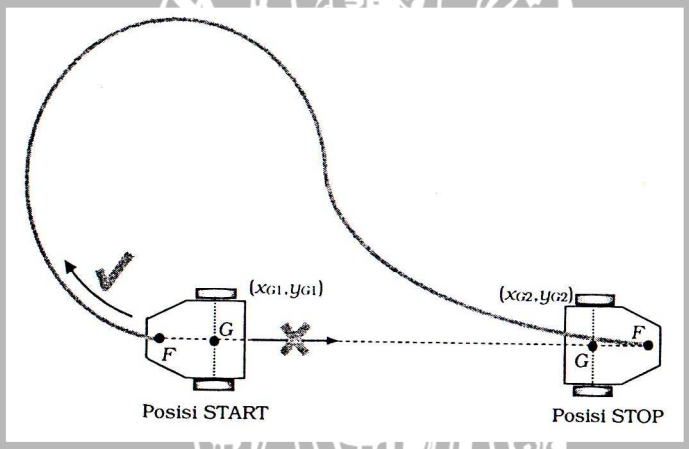
$$\dot{x}_G \sin \varphi - \dot{y}_G \cos \varphi = 0 \qquad (2.1)$$

Untuk titik F sebagai acuan analisa, persamaan di atas dapat ditulis,

$$\dot{x}_F \sin \varphi - \dot{y}_F \cos \varphi + \dot{\varphi} d = 0 \qquad (2.2)$$

Masalah klasik dalam kontrol kinematik DDMR ini adalah bahwa ia memiliki dua aktuator, namun parameter kontrolnya lebih dari dua, yaitu x untuk gerakan ke arah X dan y untuk gerakan ke arah Y yang diukur relatif terhadap perpindahan titik G, dan gerakan sudut hadap φ yang diukur dari garis hubung titik G dan F terhadap sumbu X. Inilah ciri khas dari sistem nonholonomic.

Dari persamaan (2.2) nampak bahwa derajat kebebasan dalam kontrol kinematiknya berjumlah tiga, yaitu (x, y, φ) karena ketiga parameter ini perlu dikontrol secara simultan untuk mendapatkan gerakan nonholonomic. Untuk lebih jelasnya dapat dilihat pada gambar 2.34 berikut.



Gambar 2.34 Contoh manuver DDMR
Sumber: Robotika, Buku I : 225

Perpindahan kedudukan robot dari START ke STOP bila dipandang pada titik G adalah perpindahan dari koordinat (x_{G1}, y_{G1}) ke (x_{G2}, y_{G2}) secara translasi. Namun hal ini tidak dapat dilakukan sebab robot harus dikontrol agar bergerak maju, sehingga ia harus membuat manuver belok membentuk lingkaran terlebih dahulu hingga pada posisi yang memungkinkan untuk mengarahkannya ke koordinat (x_{G2}, y_{G2}). Oleh karena itu diperlukan titik acuan F yang berada di luar garis yang menghubungkan kedua roda agar sudut hadap dapat dihitung.

- **Persamaan gerak kinematik pada DDMR**

Kinematik dalam robotic adalah suatu bentuk pernyataan yang berisi tentang deskripsi matematik geometri dari suatu struktur robot. Dari persamaan kinematik dapat diperoleh hubungan antara konsep geometri ruang sendi pada robot dengan konsep koordinat yang biasa dipakai untuk menentukan kedudukan dari suatu objek. bentuk umum persamaan kinematik untuk DDMR ini dapat dinyatakan dalam persamaan kecepatan sebagai berikut.

$$\begin{pmatrix} \dot{x}_F \\ \dot{y}_F \\ \dot{\varphi}_F \end{pmatrix} = \mathbf{T_{NH}} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix} \text{ atau } \dot{q}(t) = \mathbf{T_{NH}}(q)\dot{\theta}(t) \tag{2.3}$$

T_{NH} adalah matriks transformasi nonholonomic, $\dot{\theta}_L$ dan $\dot{\theta}_R$ adalah kecepatan radial roda kiri dan kanan, dan q adalah system koordinat umum robot,

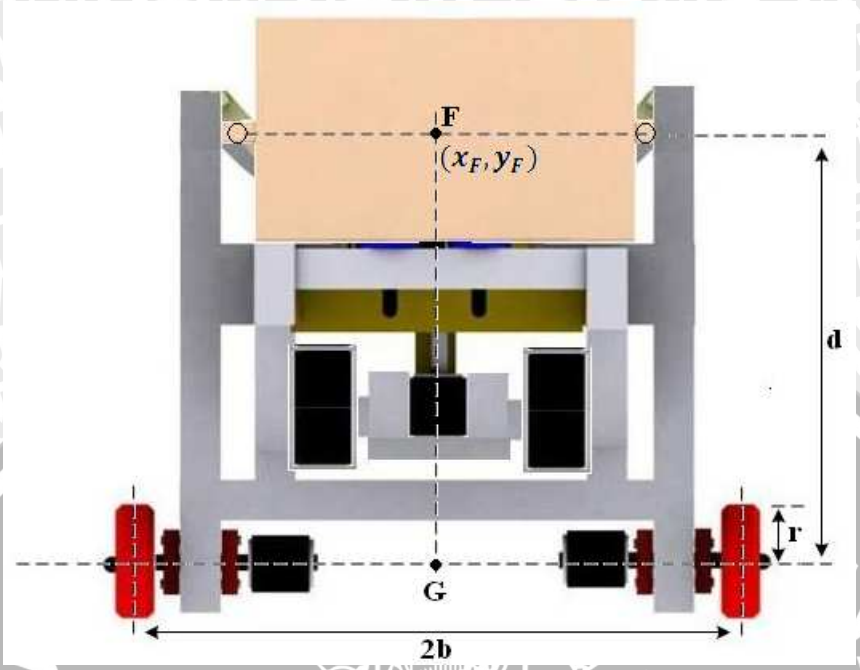
$$q = [x_F, y_F, \varphi]^T \text{ atau } q = \begin{bmatrix} x_F \\ y_F \\ \varphi \end{bmatrix} \tag{2.4}$$

Jika T_{NH} diuraikan dari persamaan (2.2) dengan memperhatikan gambar 2.33 maka dapat ditentukan,

$$\mathbf{T_{NH}}(q) = \begin{bmatrix} \frac{r}{2}\cos\varphi + \frac{d \cdot r}{2b}\sin\varphi & \frac{r}{2}\cos\varphi + \frac{d \cdot r}{2b}\sin\varphi \\ \frac{-r}{2}\sin\varphi - \frac{d \cdot r}{2b}\cos\varphi & \frac{r}{2}\sin\varphi + \frac{d \cdot r}{2b}\cos\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{bmatrix} \tag{2.5}$$

Kinematik inversnya dapat ditulis,

$$\dot{\theta}(t) = T^{-1}_{NH}(q)\dot{q}(t) \tag{2.6}$$

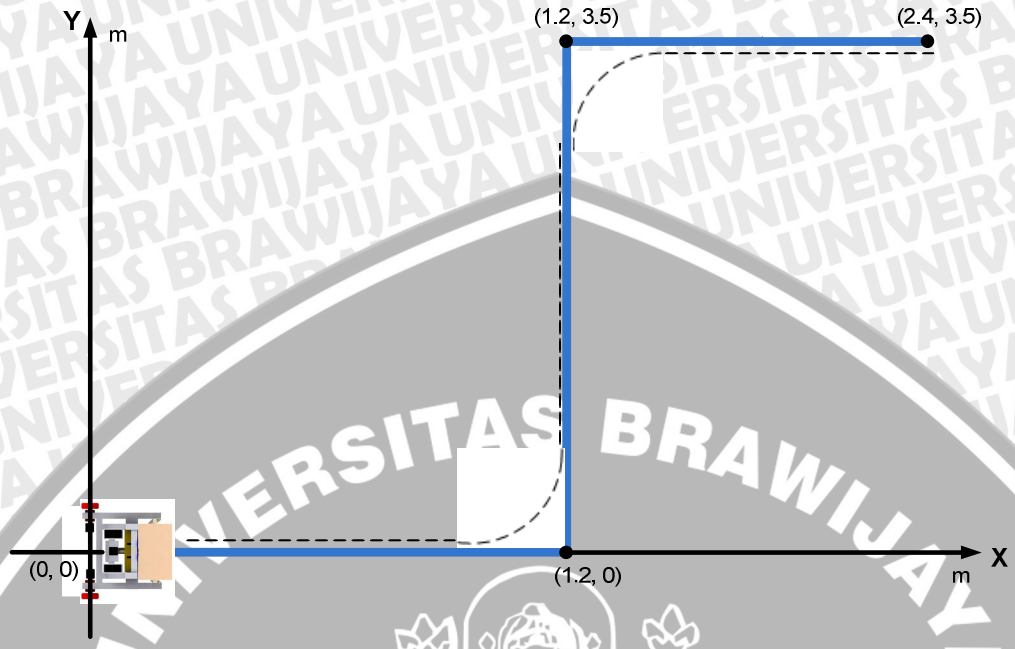


Gambar 2.35 Parameter-parameter pada robot
Sumber: Perancangan

Dari hasil pengukuran pada robot maka diperoleh besaran-besaran sebagai berikut : 2b dengan nilai 48 cm, r dengan nilai 4 cm, dan d dengan nilai 33 cm. Apabila robot dibawa ke dalam kawasan 2D pada koordinat cartesian dan titik G diasumsikan berada pada koordinat (0,0) maka akan diperoleh titik F akan berada pada koordinat (0.33, 0). Dengan demikian matriks posisi dan matriks transformasi nonholonomic robot dapat diketahui untuk menentukan persamaan kecepatan.

$$q = [x_F, y_F, \varphi]^T \text{ atau } q = \begin{bmatrix} 0.33 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{pmatrix} \dot{x}_F \\ \dot{y}_F \end{pmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ -\frac{d.r}{2b} & \frac{d.r}{2b} \end{pmatrix} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$



Gambar 2.36 Simulasi gerakan robot dalam kawasan cartesian
Sumber: Perancangan

$$\begin{pmatrix} \dot{x}_F \\ \dot{y}_F \\ \dot{\varphi}_F \end{pmatrix} = \begin{bmatrix} \frac{r}{2}\cos\varphi + \frac{d\cdot r}{2b}\sin\varphi & \frac{r}{2}\cos\varphi + \frac{d\cdot r}{2b}\sin\varphi \\ \frac{-r}{2}\sin\varphi - \frac{d\cdot r}{2b}\cos\varphi & \frac{r}{2}\sin\varphi + \frac{d\cdot r}{2b}\cos\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{bmatrix} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$

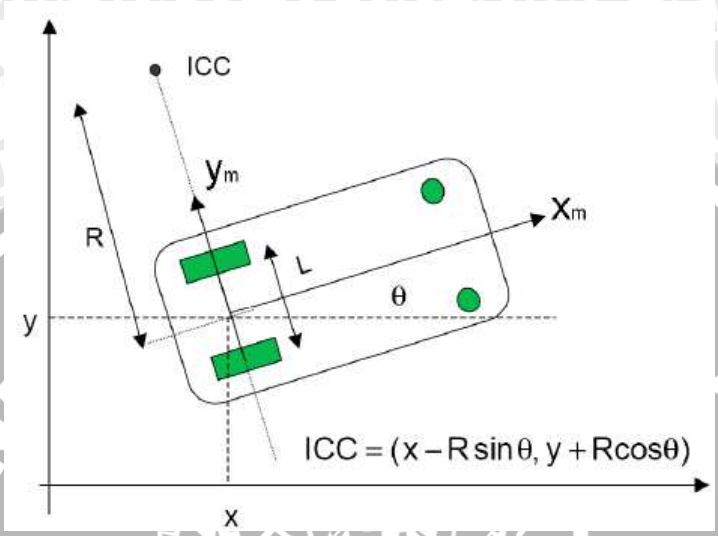
$$= \begin{bmatrix} \frac{0.04}{2}\cos 0^0 + \frac{0.33\cdot 0.04}{0.48}\sin 0^0 & \frac{0.04}{2}\cos 0^0 + \frac{0.33\cdot 0.04}{0.48}\sin 0^0 \\ \frac{-0.04}{2}\sin 0^0 - \frac{0.33\cdot 0.04}{0.48}\cos 0^0 & \frac{0.04}{2}\sin 0^0 + \frac{0.33\cdot 0.04}{0.48}\cos 0^0 \\ -\frac{0.04}{0.48} & \frac{0.04}{0.48} \end{bmatrix} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$

$$= \begin{bmatrix} 0.02 & 0.02 \\ -0.0275 & 0.0275 \\ -0.083 & 0.083 \end{bmatrix} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$

Sehingga kinematik inversnya dapat ditulis,

$$\dot{\theta}(t) = T_{NH}^{-1}(q)\dot{q}(t)$$

$$= \begin{bmatrix} 0.02 & 0.02 \\ -0.0275 & 0.0275 \\ -0.083 & 0.083 \end{bmatrix}^{-1} \begin{bmatrix} 0.33 \\ 0 \\ 0 \end{bmatrix} \begin{pmatrix} \dot{x}_F \\ \dot{y}_F \end{pmatrix}$$



Gambar 2.37 Parameter-parameter robot dalam kawasan cartesian
Sumber: Autonomous System : 16

Keterangan :

- 2 buah roda belakang, yang difungsikan sebagai pengemudi.
- 2 buah roda depan, berupa castor yang dapat bergerak ke segala arah.
- r = jari-jari roda
- R = jari-jari sesaat robot pada lintasan, relative terhadap poros.

  Untuk roda kiri $R - \frac{L}{2}$

  Untuk roda kanan $R + \frac{L}{2}$

Parameter-parameter yang ada pada roda :

- r = jari-jari roda
- v = kecepatan linear roda
- w = kecepatan anguler roda

$$w(t)=\frac{v_r(t)}{R+L/2} \qquad w(t)=\frac{v_r(t)-v_l(t)}{L}$$

$$w(t)=\frac{v_l(t)}{R-L/2} \Rightarrow R=\frac{L}{2}\frac{(v_l(t)+v_r(t))}{(v_l(t)-v_r(t))} \Rightarrow v(t)=w(t)R=\frac{1}{2}(v_r(t)+v_l(t))$$

$$\dot{x}(t)=v(t)\cos\theta(t)$$
$$\dot{y}(t)=v(t)\sin\theta(t)$$
$$\dot{\theta}(t)=w(t)$$

$$\Leftrightarrow$$

$$x(t)=\int_0^t v(\sigma)\cos(\theta(\sigma))d\sigma$$
$$y(t)=\int_0^t v(\sigma)\sin(\theta(\sigma))d\sigma$$
$$\theta(t)=\int_0^t w(\sigma)d\sigma$$

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} w_l(t) \\ w_r(t) \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 0 \\ -1/L & 1/L \end{bmatrix} \begin{bmatrix} v_l(t) \\ v_r(t) \end{bmatrix}$$

$W_r(t)$ = kecepatan angular roda kanan

$W_l(t)$ = kecepatan angular roda kiri

$V_r(t)$ = kecepatan linear roda kanan

$V_l(t)$ = kecepatan linear roda kiri

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} \cos\theta \cdot v(t) \\ \sin\theta \cdot v(t) \\ w(t) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}\cos\theta \cdot (v_r + v_l) \\ \frac{1}{2}\sin\theta \cdot (v_r + v_l) \\ (v_r - v_l)/L \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}\cos\theta & \frac{1}{2}\cos\theta \\ \frac{1}{2}\sin\theta & \frac{1}{2}\sin\theta \\ -1/L & 1/L \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix}$$

Dari $\dot{x}(t) = \cos\theta \cdot v(t)$ dan $\dot{y}(t) = \sin\theta \cdot v(t)$,

$-\dot{x}(t)\sin\theta + \dot{y}(t)\cos\theta = 0$ (nonholonomic constraint)

Pada kondisi bergerak lurus, robot berada dalam kecepatan maksimum

$$V(t) = V_r = V_l$$

$$W(t) = 0 \Rightarrow \dot{\theta}(t) = 0 \Rightarrow \theta(t) = kons\tan$$

Kecepatan sudut roda kanan dan kiri robot pada kecepatan maksimum, yaitu pada pembacaan sensor 000100, 001100, dan 001000 adalah 148 rpm = 15,52 rad/s. dari data tersebut kecepatan linear robot dapat dihitung dengan rumus:

$V = \omega R \Rightarrow$ R= jari-jari roda = 4 cm

V(t) = 15,52rad/s x 4 cm

= 15,52rad/s x 4.10^{-2}m

= 0,6208 m/s

Pada pembacaan sensor 001000, 011000, 010000 diasumsikan robot berada pada posisi sedikit ke arah kiri terhadap trayektori yang diberikan, maka aksi kontrol yang diberikan adalah roda kanan dipercepat

V_r = 152 rpm = 15,92 rad/s, sehingga kecepatan linearnya = 0,6368 m/s

V_l = 148 rpm = 15,52 rad/s, sehingga kecepatan linearnya = 0,6208 m/s

Sehingga kecepatan robot dapat dihitung dengan rumus:

$V(t) = \frac{1}{2} (V_r(t) + V_l(t))$

$= \frac{1}{2} (0,6368 + 0,6208)$

= 0,6288 m/s

Dengan diketahui nilai L = 48 cm, yaitu jarak antara roda kiri dan roda kanan, sudut penyesuaian untuk memperbaiki posisi robot dapat dihitung dengan rumus:

$$W(t) = \frac{V_r(t) - V_l(t)}{L}$$

$$W(t) = \frac{0{,}6368 - 0{,}6208}{0{,}48}$$

$$W(t) = 0{,}034^{o}$$

Pada pembacaan sensor 110000, 100000, diasumsikan robot berada pada posisi terlalu ke arah kiri terhadap trayektori yang diberikan, maka aksi kontrol yang diberikan adalah roda kanan dipercepat

V_r = 155 rpm = 16,23 rad/s, sehingga kecepatan linearnya = 0,6492 m/s

V_l = 148 rpm = 15,52 rad/s, sehingga kecepatan linearnya = 0,6208 m/s

Sehingga kecepatan robot dapat dihitung dengan rumus:

$V(t) = ½ (V_r(t) + V_l(t))$

$= ½ (0{,}6492 + 0{,}6208)$

$= 0{,}6350$ m/s

Sudut penyesuaian

$$W(t) = \frac{V_r(t) - V_l(t)}{L}$$

$$W(t) = \frac{0{,}6492 - 0{,}6208}{0{,}48}$$

$$W(t) = 0{,}059^{o}$$

Pada pembacaan sensor 000011, 000010, 000110, diasumsikan robot berada pada posisi sedikit ke arah kanan terhadap trayektori yang diberikan, maka aksi kontrol yang diberikan adalah roda kanan diperlambat

V_r = 138 rpm = 14,45 rad/s, sehingga kecepatan linearnya = 0,5780 m/s

V_l = 148 rpm = 15,52 rad/s, sehingga kecepatan linearnya = 0,6208 m/s

Sehingga kecepatan robot dapat dihitung dengan rumus:

$V(t) = ½ (V_r(t) + V_l(t))$

$= ½ (0{,}5780 + 0{,}6208)$

$= 0{,}5999$ m/s

Sudut penyesuaian

$$W(t) = \frac{V_r(t) - V_l(t)}{L}$$

$$W(t) = \frac{0{,}5780 - 0{,}6208}{0{,}48}$$

$$W(t) = -0{,}089^{o}$$

Pada pembacaan sensor 000001 diasumsikan robot berada pada posisi terlalu ke arah kanan terhadap trayektori yang diberikan, maka aksi kontrol yang diberikan adalah roda kanan diperlambat

V_r = 122 rpm = 12,78 rad/s, sehingga kecepatan linearnya = 0,5112 m/s

V_l = 148 rpm = 15,52 rad/s, sehingga kecepatan linearnya = 0,6208 m/s

Sehingga kecepatan robot dapat dihitung dengan rumus:

$V(t) = ½ (V_r(t) + V_l(t))$

$= ½ (0{,}5112 + 0{,}6208)$

$= 0{,}5660$ m/s

Sudut penyesuaian

$$W(t) = \frac{V_r(t) - V_l(t)}{L}$$

$$W(t) = \frac{0{,}5112 - 0{,}6208}{0{,}48}$$

$$W(t) = -0{,}228^{o}$$

Untuk sudut penyesuaian yang bernilai positif memiliki arti robot bergerak kearah kiri, sedangkan untuk sudut penyesuaian yang bernilai negative memiliki arti robot bergerak kearah kanan.

- **Persamaan gerak dinamik pada DDMR**

Persamaan Gerak Dinamik (*dynamics motion equation*) mobile robot secara umum dapat diekspresikan dalam bentuk terminology torsi dinamik sebagai berikut,

$$B(q)\tau = M(q)\ddot{q} + V\left(q,\dot{q}\right)\dot{q} + G(q) - A^{T}(q)\lambda \qquad (2.7)$$

dengan

$q \in \Re^{n}$ koordinat umum robot

$\tau \in \Re^{n}$ vektor torsi actuator (input)

$B(q) \in \Re^{n\times(n-m)}$ vektor matriks transformasi input

$\lambda \in \Re^{n}$ vektor torsi gangguan

$M(q) \in \Re^{n\times n}$ vektor transformasi (2X2) matriks inersia

$V\left(q,\dot{q}\right) \in \Re^{n\times n}$ vektor torsi efek coriolis dan gaya sentrifugal

$G(q) \in \Re^{n}$ vektor torsi efek gravitasi

$A(q) \in \Re^{m\times n}$ matriks transformasi yang berhubungan dengan Gangguan atau hambatan struktur kinematik nonholonomic, yang memenuhi

$A(q)\dot{q} = 0$

Perhatikan kembali gambar 2.33. sebelumnya telah diterangkan tentang persamaan kinematik untuk mobile robot DDMR secara umum yaitu $\dot{q}(t) = \mathbf{T_{NH}}(q)\dot{\theta}(t)$ dengan $\mathbf{T_{NH}}$ adalah matriks transformasi yang dapat menyelesaikan masalah nonholonomic pada system koordinat umum robot, $q = [x_F, y_F, \varphi]^T$, dengan

$$(q) = \begin{bmatrix} \frac{r}{2}\cos\varphi + \frac{d\cdot r}{2b}\sin\varphi & \frac{r}{2}\cos\varphi + \frac{d\cdot r}{2b}\sin\varphi \\ \frac{-r}{2}\sin\varphi - \frac{d\cdot r}{2b}\cos\varphi & \frac{r}{2}\sin\varphi + \frac{d\cdot r}{2b}\cos\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{bmatrix} \qquad (2.8)$$

Dengan memasukkan persamaan (2.7) ke dalam persamaan (2.8) didapat persamaan gerak dinamik DDMR sebagai berikut,

$$T_{NH}^{T}(q)B(q)\tau = T_{NH}^{T}(q)M(q)T_{NH}(q)\ddot{q} + T_{NH}^{T}(q)M(q)T_{NH} + V\left(q,\dot{q}\right)T_{NH}^{T}(q)\dot{q} + \tau_{Q} \qquad (2.9)$$

Atau

$$B(q)\tau = M(q)\ddot{\theta} + V\left(q,\dot{q}\right)\dot{q} + \tau_{Q} \qquad (2.10)$$

Dengan $M(q)$ matriks transformasi (2X2) yang terkait dengan gerak Dinamik percepatan,

$V\left(q,\dot{q}\right)$ matriks transformasi (2X2) yang terkait dengan efek coriolis dan gaya sentrifugal

$B(q)$ matriks determinan untuk torsi motor kiri dan kanan,

τ_{Q} gangguan luar.

Matriks-matriks ini adalah,

$$M(q) = \begin{bmatrix} \frac{r^2}{4b^2}\left(mb^2 + I\right) + I_w & \frac{r^2}{4b^2} - \left(mb^2 - I\right) \\ \frac{r^2}{4b^2}\left(mb^2 - I\right) & \frac{r^2}{4b^2}\left(mb^2 + I\right) + I_w \end{bmatrix} \qquad (2.11)$$

$$V\left(q,\dot{q}\right) = \begin{bmatrix} 0 & \frac{r^2}{2b} m_c d\,\dot{\varphi} \\ -\frac{r^2}{2b} m_c d\,\dot{\varphi} & 0 \end{bmatrix} \qquad (2.12)$$

$$B(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (2.13)$$

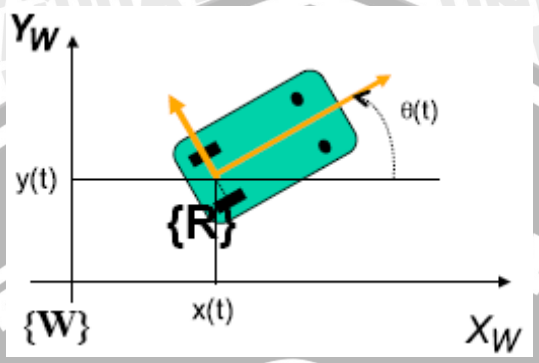
Dengan

$m = m_c + 2m_w$; m_c = massa robot (kg), m_w = massa roda (kg)

$I = m_c d^2 + 2m_w b^2 + I_c + 2I_m$; I_c adalah inersia momen tubuh robot terhadap

sumbu vertical melalui titik G,

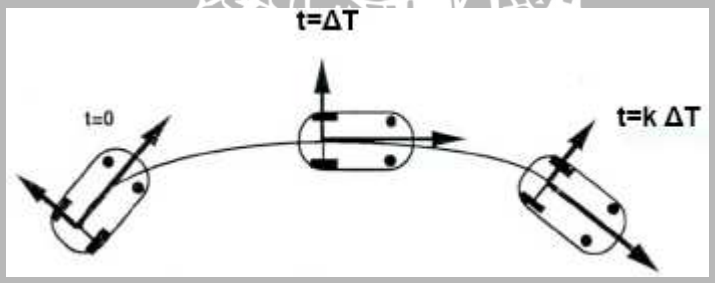
I_w adalah inersia momen roda dan rotor motor terhadap diameter roda pada sumbunya.



Gambar 2.38 Gambaran DDMR
Sumber: Autonomous System : 8

Dari model kinematik robot didapatkan persamaan untuk satu roda, yaitu

$$\begin{cases} \dot{x}(t) = \dfrac{V_d(t)+V_e(t)}{2}\cos(\theta(t)) \\ \dot{y}(t) = \dfrac{V_d(t)+V_e(t)}{2}\sin(\theta(t)) \\ \dot{\theta}(t) = \dfrac{V_d(t)+V_e(t)}{2} \end{cases}$$



Gambar 2.39 Simulasi gerakan robot dalam fungsi waktu
Sumber: Autonomous System : 8

Berasumsi bahwa

$$\begin{bmatrix} x(t=k\Delta T) \\ y(t=k\Delta T) \\ \theta(t=k\Delta T) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix}$$

Diketahui $\Delta D_l(k)$, $\Delta D_r(k)$ dan jarak yang ditempuh oleh masing-masing roda pada interval tertentu adalah $[k\Delta T,(k+1)\Delta T]$ dapat diperkirakan, yaitu :

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = ?$$

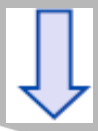
Persamaan kinematik secara diskrit

$$\begin{cases} \dot{x}(t) = \dfrac{v_d(t)+v_e(t)}{2}\cos(\theta(t)) \\ \dot{y}(t) = \dfrac{v_d(t)+v_e(t)}{2}\sin(\theta(t)) \\ \dot{\theta}(t) = \dfrac{v_d(t)-v_e(t)}{L} \end{cases}$$

untuk $t \geq t_k$

$$\begin{aligned} x(t) &= x(t_k) + \int_{t_k}^{t} \frac{v_d(\tau)+v_e(\tau)}{2}\cos(\theta(\tau))d\tau \\ y(t) &= y(t_k) + \int_{t_k}^{t} \frac{v_d(\tau)+v_e(\tau)}{2}\sin(\theta(\tau))d\tau \\ \theta(t) &= \theta(t_k) + \int_{t_k}^{t} \frac{v_d(\tau)-v_e(\tau)}{L}d\tau \end{aligned}$$

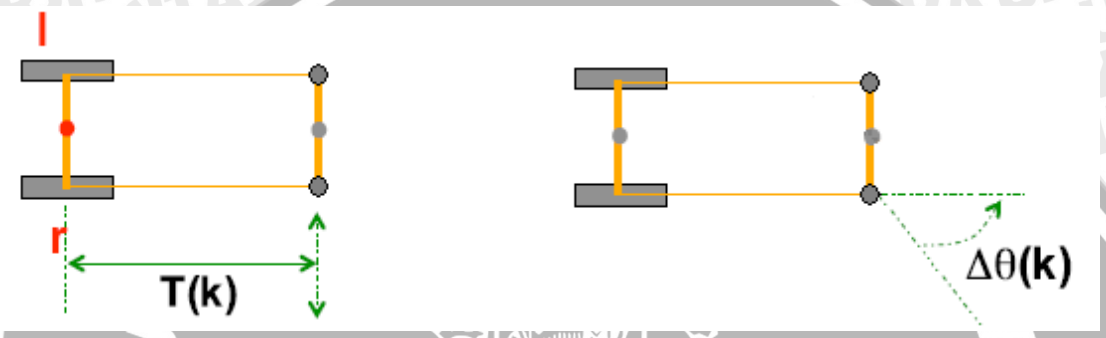
untuk $t \geq t_{k+1}$



$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} \frac{v_d(\tau)+v_e(\tau)}{2}\cos(\theta(\tau))d\tau$$

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} \frac{v_d(\tau)+v_e(\tau)}{2}\sin(\theta(\tau))d\tau$$

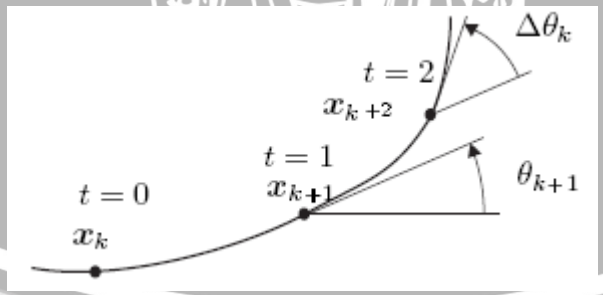
$$\theta(t_{k+1}) = \theta(t_k) + \int_{t_k}^{t_{k+1}} \frac{v_d(\tau)-v_e(\tau)}{L}d\tau$$



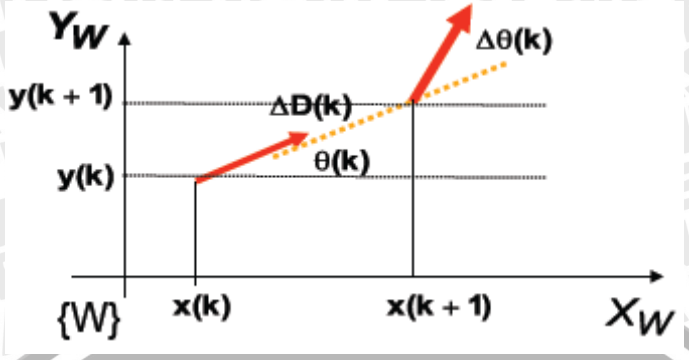
Gambar 2.40 Skema gerakan belok robot
Sumber: Perancangan

$$\begin{cases} \Delta D_e(k) = T(k) - \Delta\theta(k)\frac{L}{2} \\ \Delta D_d(k) = T(k) + \Delta\theta(k)\frac{L}{2} \end{cases} \Rightarrow \begin{cases} \Delta D(k) = \frac{\Delta D_d(k) + \Delta D_e(k)}{2} = T(k) \\ \Delta\theta(k) = \frac{\Delta D_d(k) - \Delta D_e(k)}{L} = \Delta\theta(k) \end{cases}$$

$$\Delta D_l(k),\ \Delta D_r(k) \quad ? \quad \Delta D(k),\ \Delta\theta(k)$$



Gambar 2.41 Grafik posisi robot dalam fungsi waktu
Sumber: Perancangan



Gambar 2.42 Grafik aproksimasi
Sumber: Autonomous System : 11

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} x(k) + \Delta D(k)\cos(\theta(k)) \\ y(k) + \Delta D(k)\sin(\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{bmatrix}}_{f(X(k),\Delta D(k),\Delta\theta(k))} + v(k) \quad \text{Noise}$$

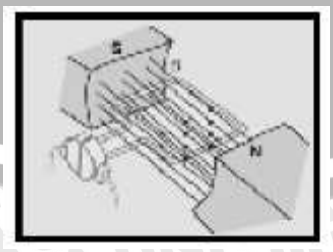
2.5 Motor DC

2.5.1 Prinsip Kerja Motor DC

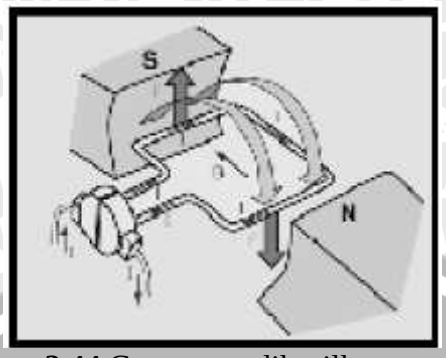
Segulung kawat yang dialiri arus listrik dan ditempatkan di dalam suatu medan magnet akan mengalami gaya yang sebanding dengan arus dan kekuatan medan magnetnya. Gaya yang ditimbulkan disebut dengan Gaya *Lorentz* [SOE-97] yang dapat dirumuskan sebagai berikut:

$$F = B\,I\,L \qquad (2.14)$$

Dalam hal ini B adalah kerapatan *fluks* magnet, I adalah arus yang mengalir dan L adalah panjang kawat.



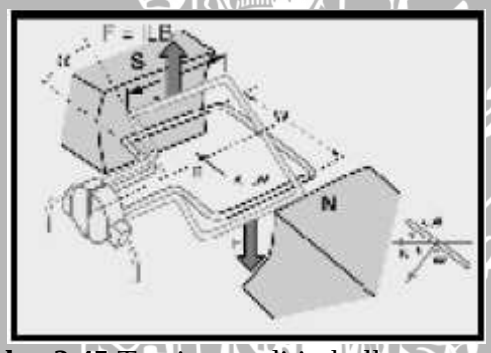
Gambar 2.43 Garis-garis medan magnet
Sumber: Sumanto,MA : 65



Gambar 2.44 Gaya yang dihasilkan motor DC
Sumber: Sumanto,MA : 65

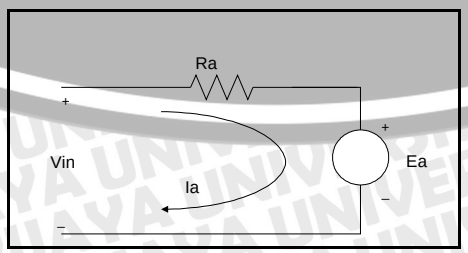
Persamaan 2.14 merupakan prinsip dasar sebuah motor, dimana terjadinya proses perubahan energi listrik (*I*) akan menimbulkan gaya mekanik (*F*). Jika motor mempunyai jari-jari sebesar *r*, maka akan menimbulkan torsi sebesar:

$$\Gamma = F.r = B.I.L.r \qquad (2.15)$$



Gambar2.45 Torsi yang ditimbulkan motor DC
Sumber: Sumanto,MA : 66

Pada saat dibangkitkan, konduktor akan bergerak di dalam medan magnet dan akan menimbulkan gaya gerak listrik (ggl) yang merupakan reaksi lawan terhadap tegangan penyebabnya. Proses konversi energi listrik menjadi energi mekanik dapat berlangsung jika tegangan sumber lebih besar dari gaya gerak listrik lawan.



Gambar 2.46 Skema Rangkaian Motor Secara Umum
Sumber: Fitzgerald, 1992 : 269

Motor dapat berputar jika tegangan masukan motor lebih besar dari ggl yang timbul. Hubungan antara tegangan sumber dan ggl lawan seperti ditunjukkan dalam Gambar 2.46, dapat dirumuskan sebagai berikut:

$$Ea = Vin - Ia.Ra \quad (2.16)$$

Dalam hal ini *Ea* adalah tegangan pada jangkar, *Vin* adalah tegangan masukan, *Ia* adalah arus jangkar dan *Ra* adalah tahanan jangkar, sedangkan induksi yang timbul adalah:

$$Ea = C\, n\, \Phi \quad (2.17)$$

Dengan *C* adalah konsatanta, *n* adalah kecepatan motor, dan Φ adalah *fluks* magnetik yang besarnya sebanding dengan arus penguatan torsi. Torsi pada motor juga sebanding dengan *fluks* magnetik dan arus. Hal ini ditunjukkan pada Persamaan 2.18 berikut:

$$\Gamma = C\, \Phi\, Ia \quad (2.18)$$

Jika diketahui kecepatan sudut ω adalah:

$$\omega = 2\,\pi\,\frac{n}{60} \quad (2.19)$$

Maka hubungan torsi dan kecepatan motor adalah:

$$\Gamma = \frac{P}{\omega}$$

$$\Gamma = \frac{P}{2\pi\frac{n}{60}} \quad (2.20)$$

Dengan *P* adalah daya motor.

2.5.2 Pengaturan Motor DC

Apabila Persamaan 2.16 disubstitusikan dengan Persamaan 2.17 akan didapatkan rumus kecepatan motor (n) sebagai berikut:

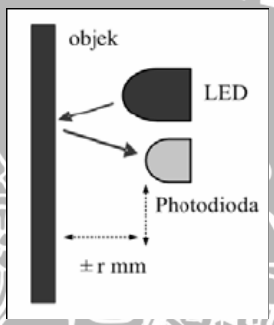
$$n = \frac{Vin - Ia\,Ra}{C\Phi} \quad (2.21)$$

Dari Persamaan 2.21 terlihat bahwa kecepatan motor sebanding dengan tegangan masukan (*Vin*) [ZUH-93: 91]. Jadi apabila tegangan masukan besar maka kecepatan

motor akan cepat, demikian pula sebaliknya, jika tegangan masukan kecil maka kecepatan motor akan lambat.

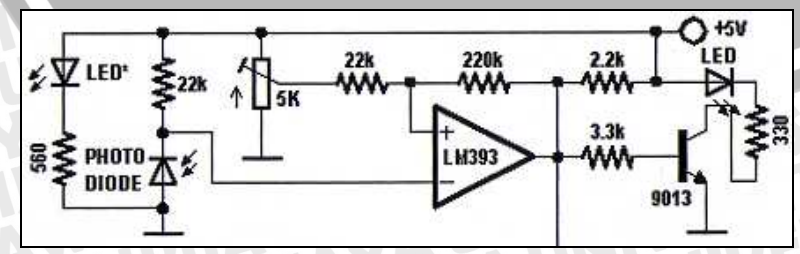
2.6 Sensor Pengikut Garis (line follower).

Gerakan robot otomatis ini dipandu oleh LED *superbright* sebagai *transmitter* dan *photodioda* sebagai *receiver* yang akan membedakan antara garis hitam dengan lapangan putih. LED diberi sinyal masukan dari sumber DC 5 volt. Sinar LED ini diarahkan menghadap ke permukaan lapangan, begitu juga *photodioda*, seperti ditunjukkan dalam gambar berikut ini



Gambar 2.47 Ilustrasi prinsip kerja sensor pengikut garis

Saat sinar dari LED mengenai warna putih, intensitas sinar yang terpantul adalah besar, sedangkan saat sinar LED mengenai warna yang lebih gelap, intensitas sinar yang terpantul menjadi lebih kecil daripada saat terkena warna putih tadi. Pantulan cahaya ini ditangkap oleh *photodiode*. Perbedaan intensitas cahaya terpantul (antara putih dan hitam) mengakibatkan suatu perubahan resistansi pada *photodiode*, yang nilainya berbanding terbalik dengan nilai intensitas cahaya pantul tersebut. Oleh karena itu, bisa didapatkan tegangan keluaran yang berbeda pada sensor saat terpantul warna putih dengan saat terpantul warna yang lebih gelap.

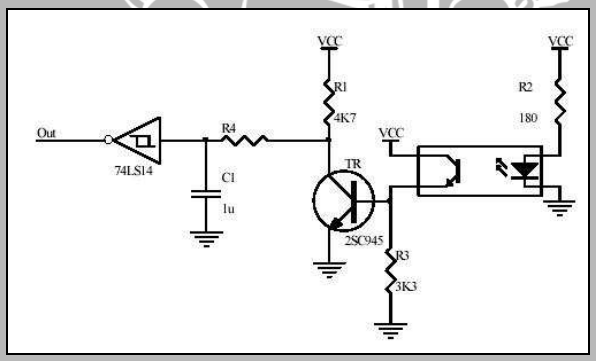


Gambar 2.48. Rangkaian sensor pengikut garis

Perbedaan tegangan ini diumpankan pada sebuah komparator di mana komparator itu akan membandingkan kedua tegangan yang berbeda tersebut. Kedua tegangan yang berbeda tersebut akan direpresentasikan sebagai logika *high* atau *low*. Logika tersebut dimasukkan ke dalam sebuah kontroler (PLC) sehingga robot dapat membedakan warna gelap dengan warna yang lebih putih (dalam hal ini warna lapangan).

2.7 Sensor Putaran (encoder)

Sensor putaran menggunakan optoswitch model U (860D) untuk membaca putaran dari piringan berlubang yang berputar sinkron dengan putaran motor mobilitas robot. Piringan dibuat berlubang banyak dan rapat agar pembacaan bisa presisi. Sensor putaran ini terdapat pada roda kiri dan kanan robot. Tujuan penggunaan sensor putaran ini adalah untuk menyamakan putaran motor mobilitas kiri dan kanan, sekaligus juga menghitung berapa jumlah putaran yang dihasilkan. Hal ini dimaksudkan untuk menunjang gerak lurus robot agar benar-benar stabil, selain nantinya juga akan dikombinasikan dengan sensor-sensor lain. Rangkaian dari sensor putaran ditunjukkan dalam Gambar 2.49.



Gambar 2.49 Rangkaian sensor putaran

LED pada optoswitch akan menyala saat catu Vcc diberikan padanya. Tergantung dari putaran piringan, bila piringan yang berputar di dalam optoswitch model U ini tepat pada bagian yang berlubang, maka cahaya dari LED akan diterima oleh phototransistor sehingga phototransistor saturasi. Karena phototransistor saturasi, tegangan di kolektor akan sangat kecil dan tegangan pada R3 mendekati nilai

Vcc. Tegangan pada R3 akan menghasilkan arus basis untuk transistor TR hingga saturasi. Karena transistor TR saturasi, tegangan di kolektor akan sangat kecil dan tegangan pada R1 akan mendekati nilai Vcc. Tegangan pada R1 ini merupakan *input* logika rendah bagi mikrokontroler. *Input* logika tinggi terjadi bila piringan yang berputar di dalam optoswitch model U ini tepat pada bagian yang tidak berlubang. Yang berarti pula nantinya *input* pada mikrokontroler berlogika tinggi. Untuk menghilangkan *spike* yang muncul pada saat terjadi perubahan logika dari rendah ke tinggi atau sebaliknya, maka sebelum masuk ke kaki mikrokontroler sebagai *input*, sinyal perlu diumpankan dulu pada inverter *Schmitt trigger* (74LS14). Schmitt trigger ini membuat sinyal yang masuk ke mikrokontroler benar-benar berupa sinyal kotak yang setara logika rendah dan tinggi.

BAB III
METODOLOGI

Kajian dalam skripsi ini merupakan penelitian yang bersifat aplikatif, yaitu perencanaan dan pembuatan robot otomatis yang dikendalikan oleh PLC dalam sistem pergerakannya. Jenis data yang digunakan dalam desain skripsi ini meliputi data sekunder dan data primer. Data dan spesifikasi komponen yang digunakan dalam perencanaan merupakan data sekunder yang diambil dari buku data komponen elektronika dan literatur terkait. Pemilihan komponen berdasarkan perencanaan dan disesuaikan dengan komponen yang ada di pasaran. Sedangkan data primer merupakan data spesifikasi alat dari hasil pengujian. Langkah-langkah yang perlu dilakukan untuk merealisasikan sistem yang akan dibuat adalah sebagai berikut:

3.1 Studi Literatur

Pada tahap ini, dilakukan pengumpulan data-data dan literatur yang menunjang dalam perencanaan dan pembuatan sistem secara keseluruhan, dengan harapan dapat banyak membantu dalam proses-proses selanjutnya. Adapun data-data yang dibutuhkan dalam penulisan skripsi ini adalah antara lain:

a. Sistem instalasi PLC
b. Pembuatan program PLC melalui software Micro/WIN
c. Sensor pengikut garis (*line follower*)
d. Sensor putaran (*encoder*)

3.2 Perancangan Sistem

Perancangan sistem dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem beserta rangkaian elektronik pendukungnya, hal ini dimaksudkan agar sistem pergerakan pada robot DDMR otomatis tersebut dapat berjalan sesuai dengan yang deskripsi awal yang telah direncanakan.

Perancangan sistem yang dilakukan meliputi:

- Penentuan spesifikasi sistem yang akan dibuat, meliputi:

a. Penentuan deskripsi kerja sistem secara keseluruhan.
b. PLC dan rangkaian elektronik pendukung.

- Penentuan rangkaian yang digunakan.
- Perancangan diagram tangga melalui compiler Micro/WIN.

3.3 Realisasi Pembuatan Sistem

Setelah melalui proses perancangan sistem, langkah selanjutnya adalah merealisasikan hasil dari rancangan yang telah dibuat menjadi *Hardware* yang sesungguhnya serta memasukkan program yang telah dibuat tersebut ke dalam sistem secara keseluruhan. Bahasa pemrograman yang digunakan dalam perangkat ini adalah bahasa *Ladder Diagram* atau dalam bentuk *mnemonic code* dari *Ladder Diagram* yang digunakan. Pembuatan program disesuaikan dengan PLC yang digunakan, dalam penulisan skripsi ini digunakan PLC produksi Siemens seri S7-200 yang dapat diprogram melalui komputer dengan *Software* Micro/WIN.

Realisasi pembuatan sistem yang dilakukan meliputi:

- Pembuatan mekanik robot otomatis secara keseluruhan yang meliputi penempatan PLC, sistem pergerakan, motor DC dan komponen pendukung lainya.
- Pengisian program yang telah dirancang sebelumnya pada PLC.

3.4 Pengujian Sistem

Pengujian ini dilakukan untuk mengetahui apakah sistem pergerakan secara keseluruhan telah bekerja sesuai dengan yang telah direncanakan sebelumnya atau tidak. Pengujian juga digunakan untuk mengetahui waktu yang dibutuhkan untuk melakukan pergerakan bila menggunakan PLC.

3.5 Pengambilan Kesimpulan

Kesimpulan diambil berdasarkan data yang diperoleh dari pengujian sistem secara keseluruhan. Jika hasil yang didapatkan telah sesuai dengan yang direncanakan sebelumnya, maka sistem kendali tersebut telah berhasil memenuhi harapan dan tentunya memerlukan pengembangan lebih lanjut untuk penyempurnaan.

BAB IV
PERANCANGAN SISTEM

4.1 Deskripsi Sistem Secara Umum

Perancangan secara umum dari sistem yang akan dibuat dijelaskan untuk memudahkan dalam proses pembuatan program *ladder diagram* yang akan digunakan sebagai pengendali sistem pergerakan robot. Sistem kendali tersebut dibuat berdasarkan bahasa pemrograman *ladder diagram* yang berbasis pada PLC SIEMENS dengan tipe S7-200. Sistem yang akan dibuat secara umum memiliki perencanaan sebagai berikut :

1) Robot DDMR otomatis ini tersusun atas beberapa material bahan, antara lain : Alumunium (kerangka utama, dll), Karet (roda, pelindung PLC, belt, dll), Besi (pengunci, *pulley*, *bearing*, *roller*, *casters*, dll), Baja (ulir, *cube*/rumah ulir, dll).
2) Robot DDMR otomatis ini memiliki empat buah roda, yaitu dua buah *cluster* sebagai penyangga keseimbangan robot dan dua buah roda penggerak utama yang digerakkan oleh 2 buah motor DC 24V sebagai keluaran sistem dan akan difungsikan sebagai sistem pergerakan robot DDMR (maju, mundur, belok kanan / belok kiri).
3) Robot DDMR otomatis ini dirancang untuk bisa bergerak mengikuti lintasan yang telah ditentukan dan melakukan manuver sesuai dengan belokan-belokan yang ada pada lintasan tersebut.
4) Saat robot DDMR bergerak melintasi lintasan yang telah ditentukan pergerakannya dibantu oleh sensor, ada dua jenis sensor yang digunakan. Sensor-sensor itu antara lain :

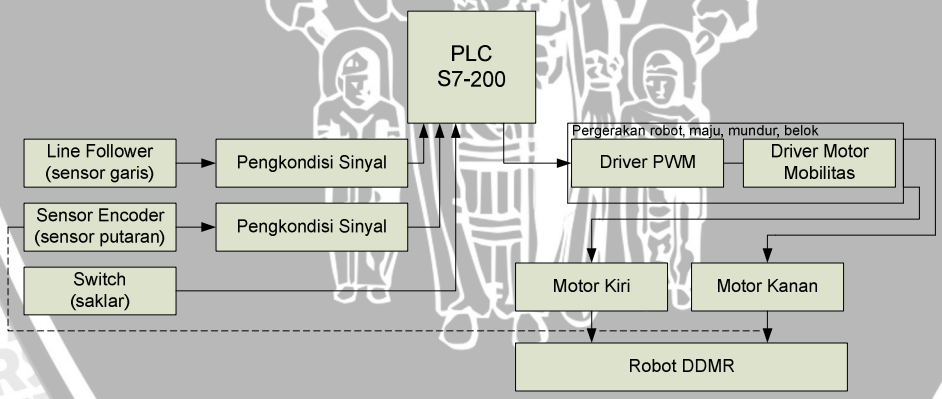
   - Sensor cahaya, yang diletakkan pada bagian bawah robot DDMR. Selain difungsikan sebagai sensor pengikut garis (*line follower*), sensor ini juga difungsikan untuk mendeteksi serta menghitung jumlah garis perempatan yang telah dilewati oleh robot pada lintasan.
   - Sensor putaran (*encoder*), yang diletakkan pada bagian samping dari robot DDMR. selain digunakan untuk mengatur putaran kedua

motor mobilitas agar diperoleh putaran yang sinkron antar keduanya, sensor ini juga digunakan untuk menghitung jumlah putaran dari motor mobilitas tersebut.

5) PLC SIEMENS S7-200 pada sistem ini difungsikan sebagai pengendali (*controller*) yang menerima masukan berupa posisi robot terhadap lintasan, masukan tersebut didapatkan dari sensor pengikut garis (*line follower*) dan sensor putaraan (*encoder*). Keluaran dari PLC berupa aksi kontrol yang mengatur kecepatan putaran motor mobilitas dengan perantara driver PWM.
6) METODE PWM digunakan untuk pengendalian kecepatan putaran motor mobilitas dengan mengatur arus yang melewati motor melalui driver motor. Motor kanan dan Motor kiri difungsikan sebagai kemudi yang menggerakkan robot ketika harus maju, belok kanan atau belok kiri.

4.2 Perancangan Sistem Robot DDMR

4.2.1 Perancangan Blok Diagram Sistem



Gambar 4.1 Blok Diagram Kerja Sistem
Sumber: Perancangan

4.2.2 Deskripsi Kerja Sistem

Dalam pembuatan suatu sistem yang kompleks agar lebih memudahkan perencanaan dan menentukan langkah-langkah selanjutnya untuk menyelesaikan sistem yang akan dibuat, maka terlebih dahulu perlu dijabarkan deskripsi kerja

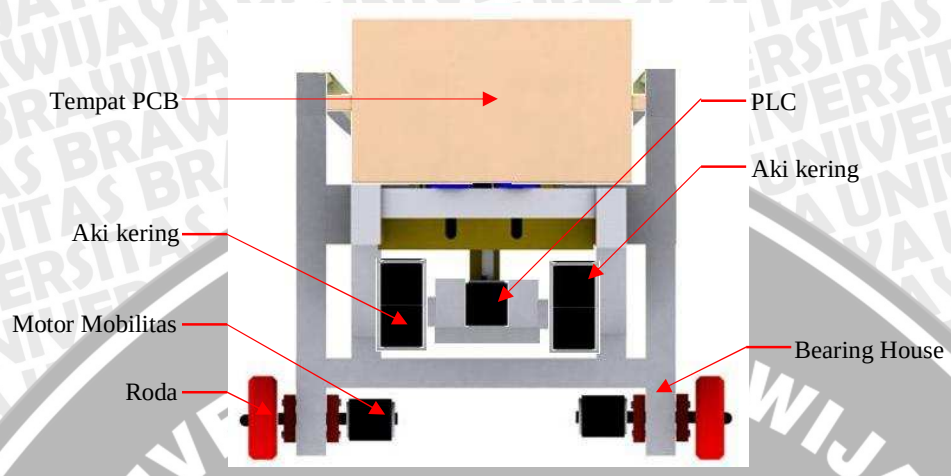
sistem secara keseluruhan. Deskripsi kerja sistem yang akan dibuat adalah sebagai berikut:

- Pada saat awal sistem mulai bekerja (saklar ON), PLC akan menentukan kecepatan kedua motor DC berdasarkan keadaan sensor *line follower* terhadap garis lintasan pada saat itu. Untuk mobilitas robot DDMR, mode pergerakan dibagi menjadi beberapa keadaan berdasarkan sarat tersebut.
- Terdapat 9 keadaan sensor *line follower* untuk menentukan mode pergerakan robot DDMR saat berjalan pada lintasan garis hitam. Penentuan mode pergerakan tersebut dimaksudkan agar robot DDMR dapat menyesuaikan diri (*self conditioning*) saat berjalan lurus maupun pada saat berbelok pada lintasan yang telah ditentukan. Sehingga dalam setiap melakukan mobilitas selalu ada penyesuaian percepatan (akselerasi) dan penyesuaian perlambatan (deselerasi) pada robot DDMR otomatis.
- Sensor putaran (encoder) yang digunakan untuk mengatur putaran kedua motor mobilitas memiliki fungsi lain yaitu untuk menghitung jumlah putaran dari motor mobilitas tersebut. Hasil penghitungan jumlah putaran motor mobilitas dimanfaatkan untuk penentuan saat yang tepat saat robot DDMR melakukan mode belokan.
- Jika jumlah perempatan yang dihitung terpenuhi, maka robot akan segera melakukan proses berbelok. Selanjutnya dibantu oleh line follower robot robot akan segera memposisikan diri untuk menyesuaikan diri pada lintasan dan melanjutkan pergerakannya.

Gambar 4.2 *Flowchart* Deskripsi Kerja Sistem
Sumber: Perancangan

4.3 Perancangan Perangkat Keras

4.3.1 Rancangan *Prototype* Robot DDMR Otomatis



Gambar 4.3 Sketsa desain tampak atas Robot DDMR
Sumber: Perancangan

→Ukuran Robot DDMR Otomatis

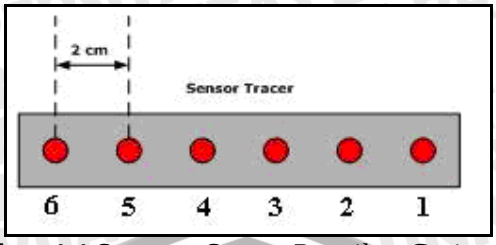
Tinggi : 150 mm

Lebar : 500 mm

Panjang : 420 mm

4.3.2 Rancangan Sensor Pengikut Garis (*line follower*)

Sensor yang digunakan adalah 6 buah sensor tracer dengan kemampuan membaca 6 titik dimana jarak masing-masing titik adalah sebesar 2 cm. Garis yang akan dibaca adalah garis berwarna hitam selebar 3 cm yang dibuat dengan menggunakan karet perekat (plester) yang membentuk jalur lintasan tertentu sesuai dengan ruang lingkup permasalahan pada Bab pendahuluan. Keluaran sensor tracer adalah berupa 6 pin logika berdasarkan pembacaan sensor tracer terhadap garis. Apabila membaca/terkena garis, maka nilai keluaran logikanya adalah satu, apabila tidak bernilai nol. Enam pin tersebut dihubungkan dengan 6 pin I/O PLC. Kombinasi logika dari 6 pin inilah yang akan diolah oleh PLC SIEMENS S7-200. Data-data mengenai prinsip kerja dan rangkaian dari sensor pengikut garis ini telah dibahas pada Bab Dasar Teori.



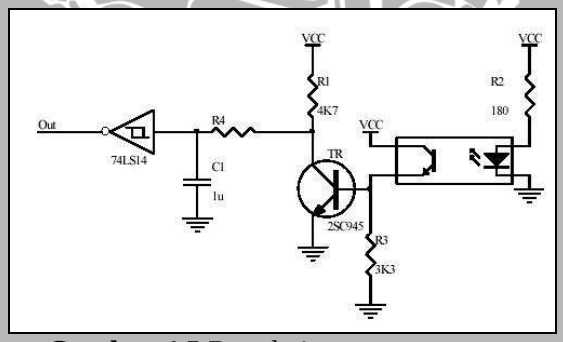
Gambar 4.4 Susunan Sensor Pengikut Garis
Sumber: Perancangan

Dalam perancangan ini, digunakan enam buah sensor pengikut garis (*line follower*) yang disusun seperti terlihat pada gambar diatas yaitu dengan jarak 2 cm antar tiap sensor. Rangkaian sensor tersebut diletakkan pada bagian bawah robot.

4.3.3 Rancangan Sensor Putaran (*encoder*)

Sensor putaran menggunakan Optoswitch model U (860D),yang digunakan untuk membaca putaran dari piringan berlubang yang berputar sinkron dengan putaran motor mobilitas, lubang pada piringan dibuat banyak dan rapat agar pembacaan bisa presisi. Data-data mengenai prinsip kerja dari sensor putaran ini telah dibahas pada Bab Dasar Teori.

Pada perancangan ini, digunakan dua buah sensor putaran (*encoder*) yang diletakkan pada roda kiri dan roda kanan robot DDMR.



Gambar 4.5 Rangkaian sensor putaran
Sumber: Datasheet

4.3.4 PLC SIEMENS S7-200

Bentuk fisik dari PLC SIEMENS S7-200 adalah seperti yang ditunjukkan pada gambar berikut :

Gambar 4.6 PLC SIEMENS S7-200

4.4 Perancangan Sistem Pengendalian Robot

4.4.1 Peralatan Masukan

- Saklar ON/OFF

  Saklar on/off berjumlah 1 buah dan berfungsi untuk menghidupkan dan mematikan sistem. Saklar yang pertama terhubung dengan alamat I0.0 dan diberi simbol SAKLAR_ON.

- Sensor *line follower*

  Sensor ini memiliki 6 masukan,masing-masing memiliki simbol S_TRACE_1,S_TRACE_2,S_TRACE_3, S_TRACE_4, S_TRACE_5, S_TRACE_6, dan beralamat I 01, I 02, I 03, I 04, I 05, I 06.

- Sensor putaran (*encoder*)

  Masing-masing sensor yang terletak di kanan-kiri *chasis* ini memiliki simbol EN_KANAN, dan EN_KIRI dan beralamatkan I 1.0 , I 1.1.

4.4.2 Peralatan Keluaran

- Motor DC untuk pergerakan *mobile robot.*

  Pada peralatan keluaran terdapat 2 keluaran berupa 2 buah motor yang terdapat pada masing-masing motor penggerak mobilitas robot yang terdapat di kanan dan kiri robot, untuk motor kanan bersimbol MTR_KANAN, dan PWM_KANAN, yang beralamat Q 0.5, dan Q 0.7 Sedangkan keluaran motor kiri bersimbol MTR_KIRI, dan PWM_KIRI, yang beralamat Q 0.0, dan Q 0.2.

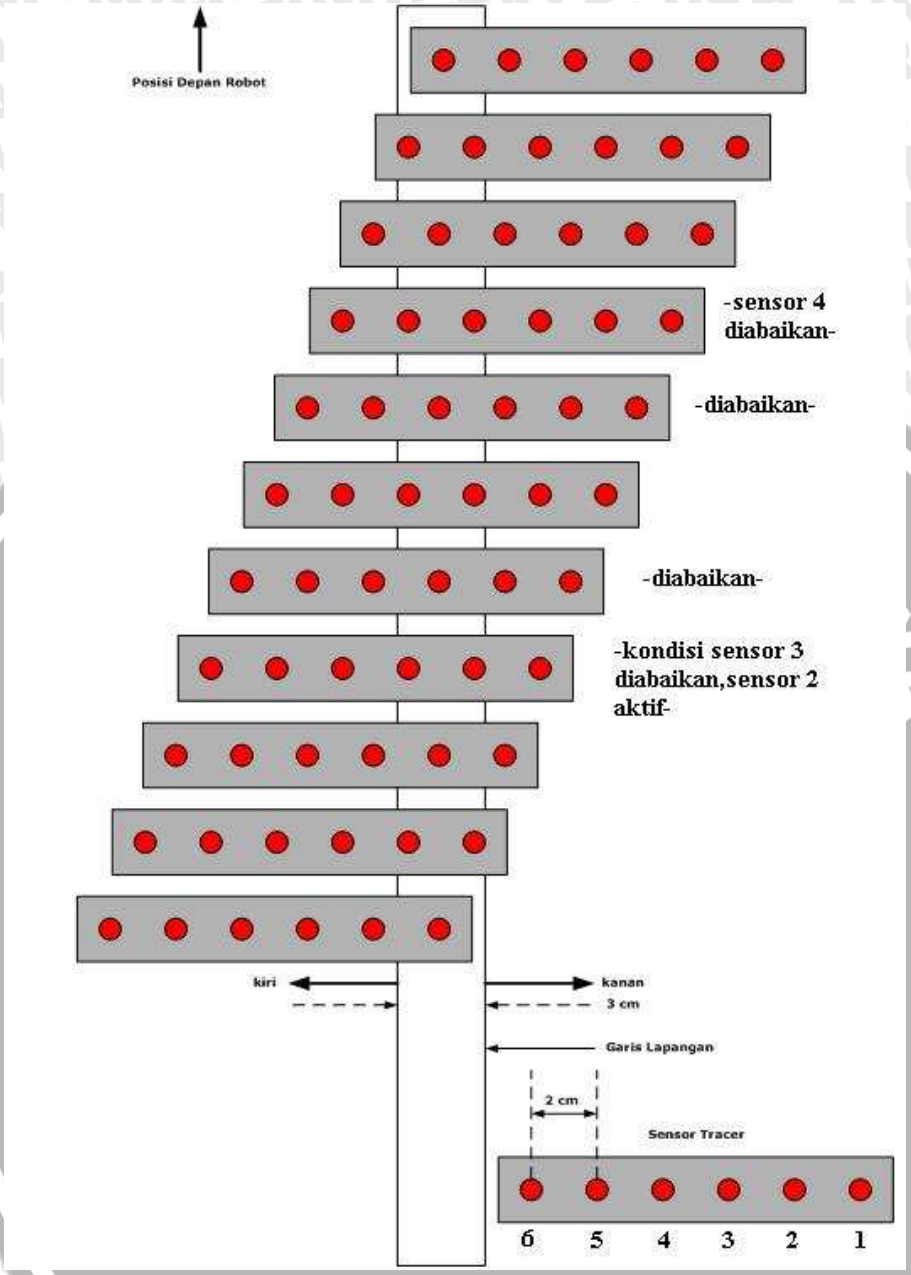
4.4.3 Perancangan PWM pada Motor DC

Perancangan PWM yang dimaksud di sini ialah menentukan prosentase PWM yang tepat untuk mengendalikan motor DC sebagai faktor utama

pergerakan robot DDMR. Dengan perancangan PWM ini diharapkan dapat mempertahankan posisi robot tetap berada pada lintasan yang telah ditentukan. Dengan adanya pemberian prosentase PWM yang tepat pada motor kanan ataupun kiri maka diharapkan bisa diperoleh hasil *tracing* yang baik, yaitu dengan semakin baiknya mobilitas robot yang dapat diketahui dari semakin cepatnya waktu tempuh yang diperlukan untuk menyelesaikan lintasan. Prosentase PWM didapatkan berdasarkan pengujian karakteristik kecepatan putaran pada masing-masing motor kanan dan kiri dengan kondisi robot tidak sedang dijalankan.

4.4.4 Perancangan Kondisi Tracing dengan Bantuan Sensor Pengikut garis (*line follower*)

Dalam perancangan ini hubungan yang ditentukan antara sensor pengikut garis (*line follower*) dengan *ladder diagram* adalah *ladder diagram* dibuat agar dapat memetakan hasil pembacaan sensor *tracer* yang kemudian akan ditentukan kecepatan motor kiri dan motor kanan agar dapat memiliki kecepatan yang disesuaikan dengan pembacaan sensor pengikut garis terhadap garis hitam pada lintasan. Visualisasi dari pemetaan sensor *tracer* dapat dilihat dalam Gambar 4.7.



Gambar 4.7 Pemetaan sensor *tracer* berdasarkan posisi robot
Sumber: Dokumentasi Teknis Tim Robot KRI TEUB 2007 + perancangan

Penentuan kondisi pergerakan robot dapat dilihat sebagai berikut :

Tabel 4.1. Aksi kontrol terhadap kondisi pembacaan tracer

Pembacaan Tracer Aktif	Aksi Kontrol Yang Diharapkan
000001	robot bergerak ke kanan (hampir di luar jalur), PWM kanan mengurangi kecepatan motor kanan

000011	robot bergerak ke kanan, PWM kanan mengurangi kecepatan motor kanan
000010	robot bergerak ke kanan, PWM kanan mengurangi kecepatan motor kanan
000110	kondisi sensor 3 diabaikan, kondisi sama dengan aktifnya sensor 2.
000100	kondisi diabaikan
001100	kecepatan 100%
001000	kondisi diabaikan
011000	kondisi diabaikan
010000	robot bergerak ke kiri, PWM kanan menambah kecepatan motor kanan
110000	robot bergerak ke kiri, PWM kanan menambah kecepatan motor kanan
100000	robot bergerak ke kiri (hampir di luar jalur), PWM kanan menambah kecepatan motor kanan

Adanya kondisi sensor yang diabaikan dimaksudkan untuk menyederhanakan program tanpa mengurangi nilai efektifnya.

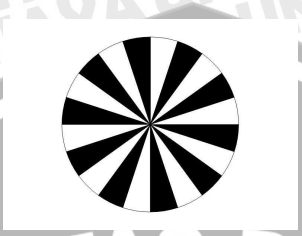
4.4.5 Perancangan Kondisi Belok Saat Tracing dengan Bantuan Sensor Putaran (*encoder*)

Sensor putaran menggunakan *optoswitch* model U (860D) untuk membaca putaran dari piringan berlubang yang berputar sinkron dengan putaran motor mobilitas robot.

Pada perancangan ini proses pembacaan pulsa dari *encoder* ini dilakukan dengan memanfaatkan fasilitas *counter* pada PLC. Pada tiap motor dapat diketahui jumlah pulsa setiap satuan waktu tertentu berikut arah dari putarannya. Dari *encoder* yang ada digunakan pin keluaran satu fase, sehingga untuk satu buah *encoder* akan membutuhkan satu buah pin input pada PLC. Dengan menggunakan *encoder* ini maka kecepatan motor dapat diketahui melalui proses pembacaan pulsa pada *counter*.

- **Perancangan Aktifasi Belok**

Pada gambar 4.8 terlihat bentuk piringan dari sensor putaran. Sehingga tiap motor dapat diketahui jumlah pulsa setiap satuan waktu tertentu berikut arah dari putarannya.



Gambar 4.8 Bentuk piringan dalam sensor putaran
Sumber: Perancangan

Dalam perancangan aktifasi belok ini ada beberapa hal yang akan diutamakan yaitu sebagai berikut :

1. Menentukan pada perempatan ke berapa robot akan berputar (panjang lintasan).
2. Menghitung keliling dari roda.
3. Menghitung jumlah lubang dalam piringan *encoder*.
4. Menghitung banyaknya hitungan *counter* dengan rumus sebagai berikut :

$$x = \frac{a}{b} \quad \text{.....(1)}$$

Keterangan :

a = keliling roda (cm)

b = jumlah lubang piringan

x = jarak roda berputar setiap satu lubang piringan (cm)

$$y = \frac{c - d}{x} \quad \text{.....(2)}$$

Keterangan :

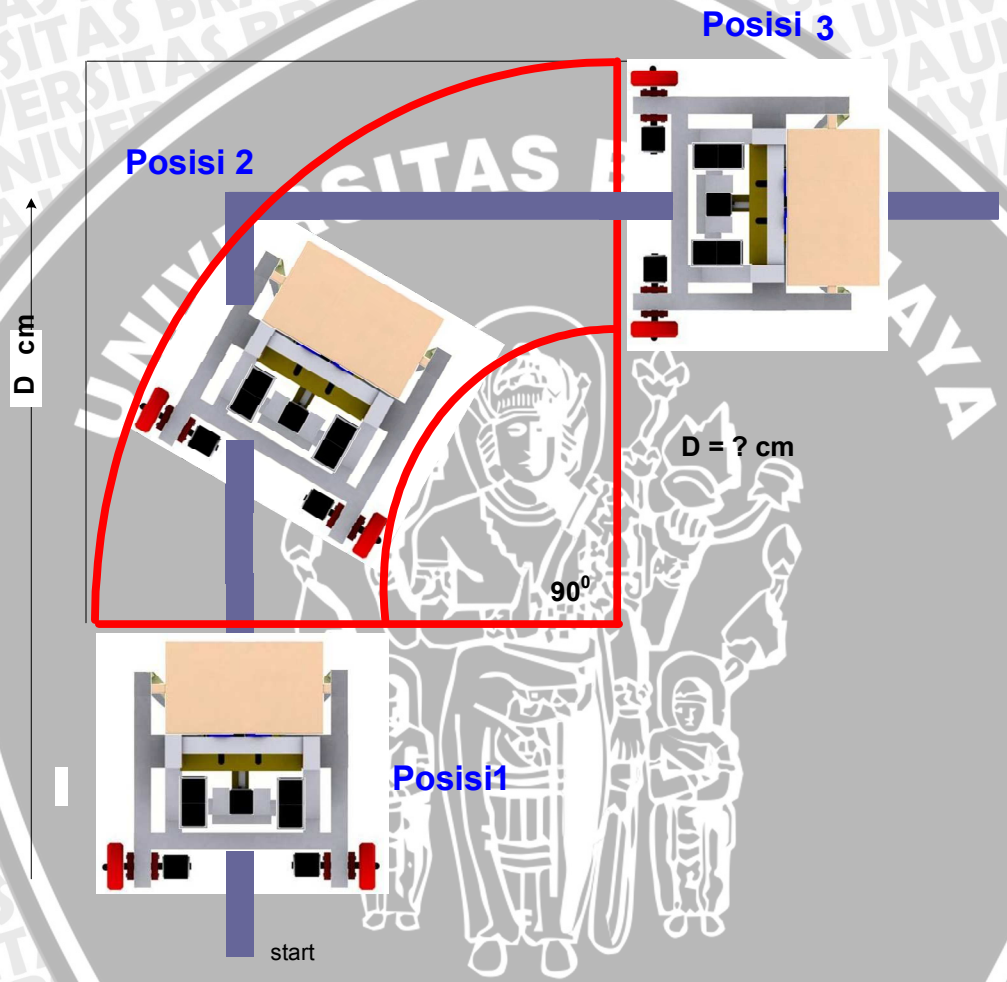
c = jarak pada perempatan yang diinginkan untuk berbelok (cm).

d = jarak batas dari jarak maksimum agar robot saat berputar berada tepat di lintasannya kembali (cm).

y = jumlah hitungan (masukan) pada counter agar robot dapat berputar.

5. Menentukan jarak z untuk posisi start.

Langkah-langkah diatas dapat diilustrasikan oleh gambar 4.9 apabila kita telah mendapatkan nilai D yang tepat maka kita dapat menentukan jumlah hitungan pada counter untuk menentukan putaran masing-masing roda. Sehingga perancangan aktifasi belok dapat dilihat pada gambar di bawah ini.



Gambar 4.9 Ilustrasi Perancangan Aktifasi Belok
Sumber: Perancangan

4.4.6 Perancangan Diagram Tangga (Program) Dengan Menggunakan Micro/WIN

Sesuai dengan deskripsi kerja sistem yang diinginkan, selanjutnya adalah merancang program *ladder diagram*, serta memberi nama dan nomor pada masing-masing ke dalam tabel-tabel berikut :

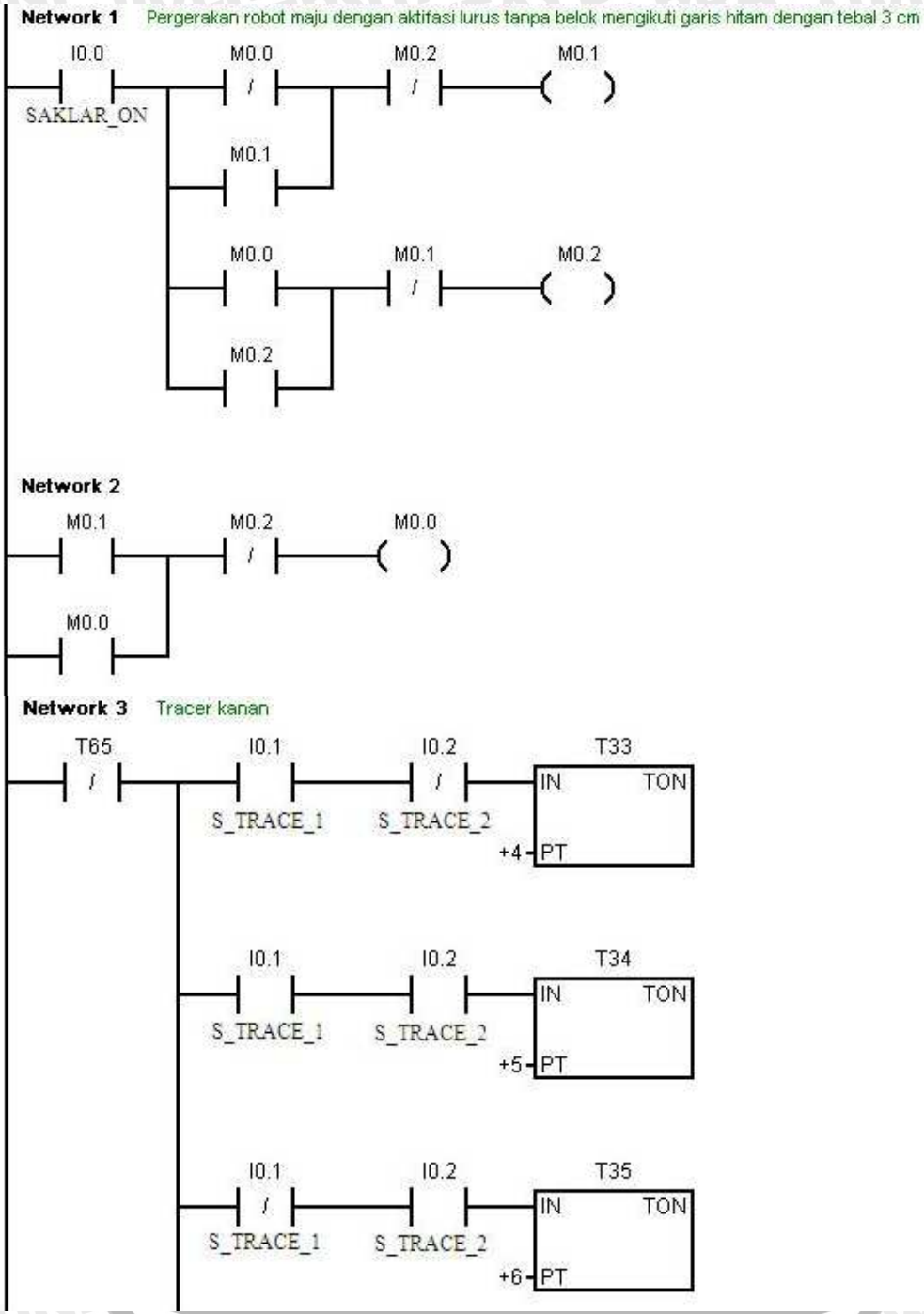
Tabel 4.2. Alamat Masukan *Ladder Diagram* Pada PLC

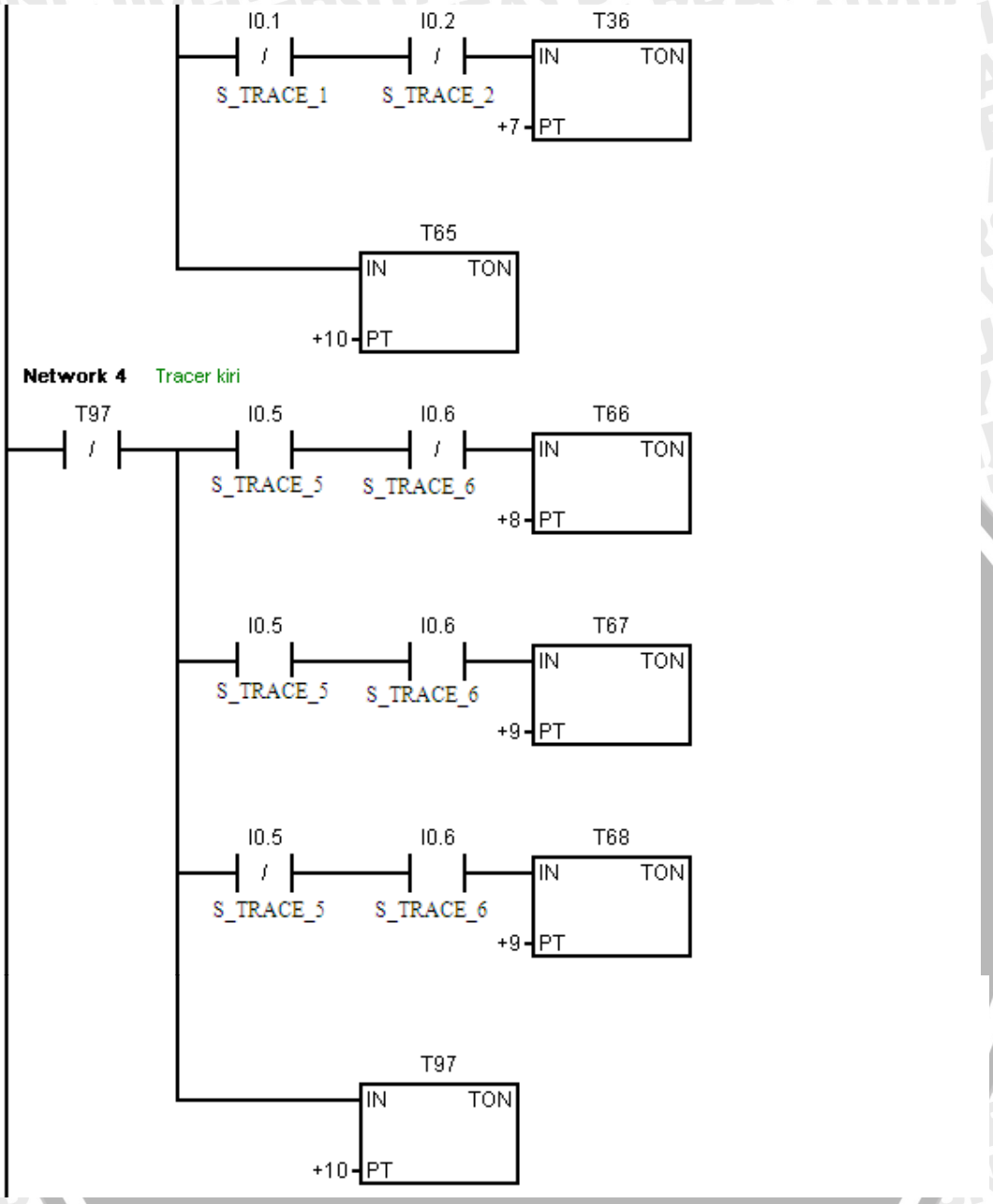
Alamat	Keterangan	Simbol
I 00	Saklar ON/OFF	SAKLAR_ON
I 01	Sensor *Line follower* 1	S_TRACE_1
I 02	Sensor *Line follower* 2	S_TRACE_2
I 03	Sensor *Line follower* 3	S_TRACE_3
I 04	Sensor *Line follower* 4	S_TRACE_4
I 05	Sensor *Line follower* 5	S_TRACE_5
I 06	Sensor *Line follower* 6	S_TRACE_6
I 1.0	Sensor *Encoder* kanan	E_KANAN
I 1.1	Sensor *Encoder* kiri	E_KIRI

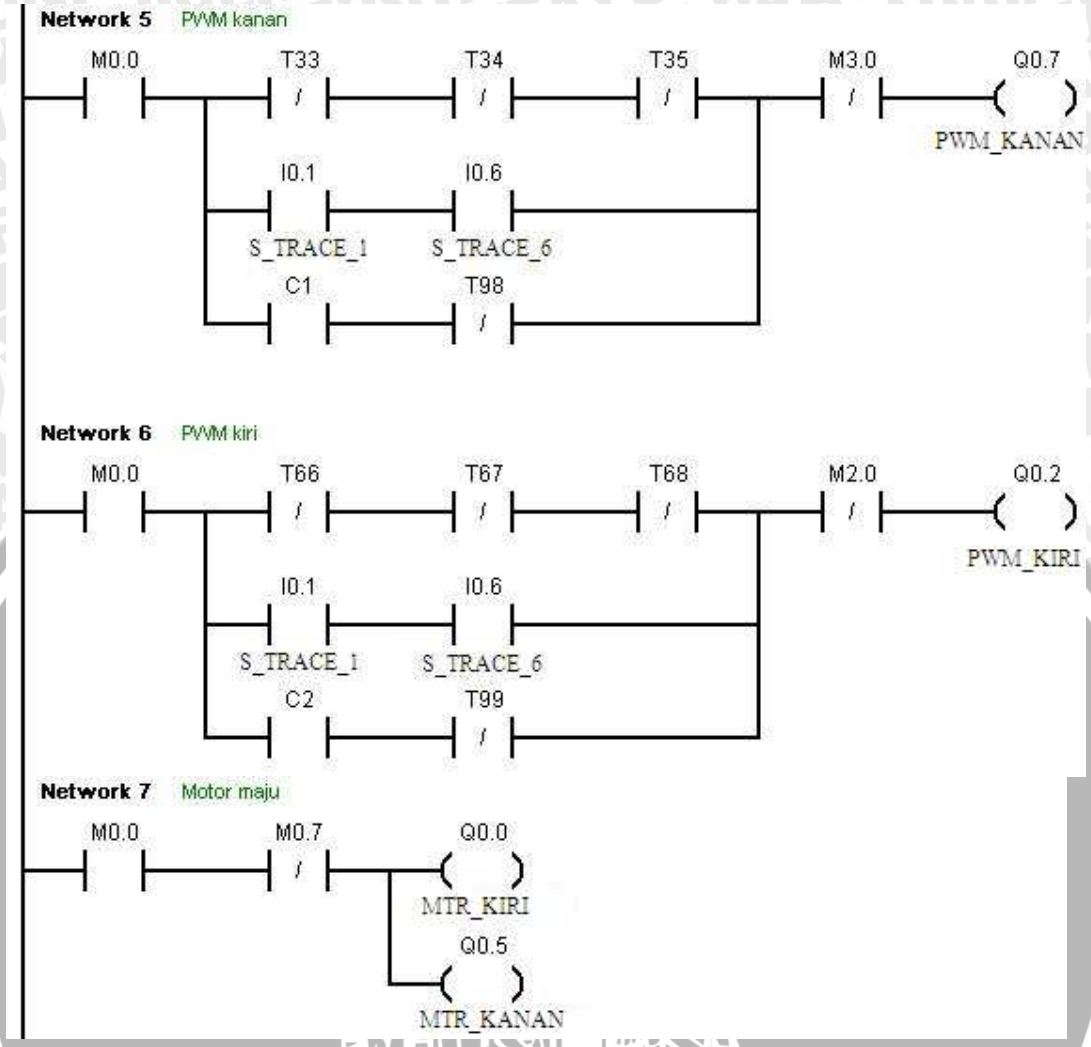
Tabel 4.3. Alamat Keluaran *Ladder* Diagram Pada PLC

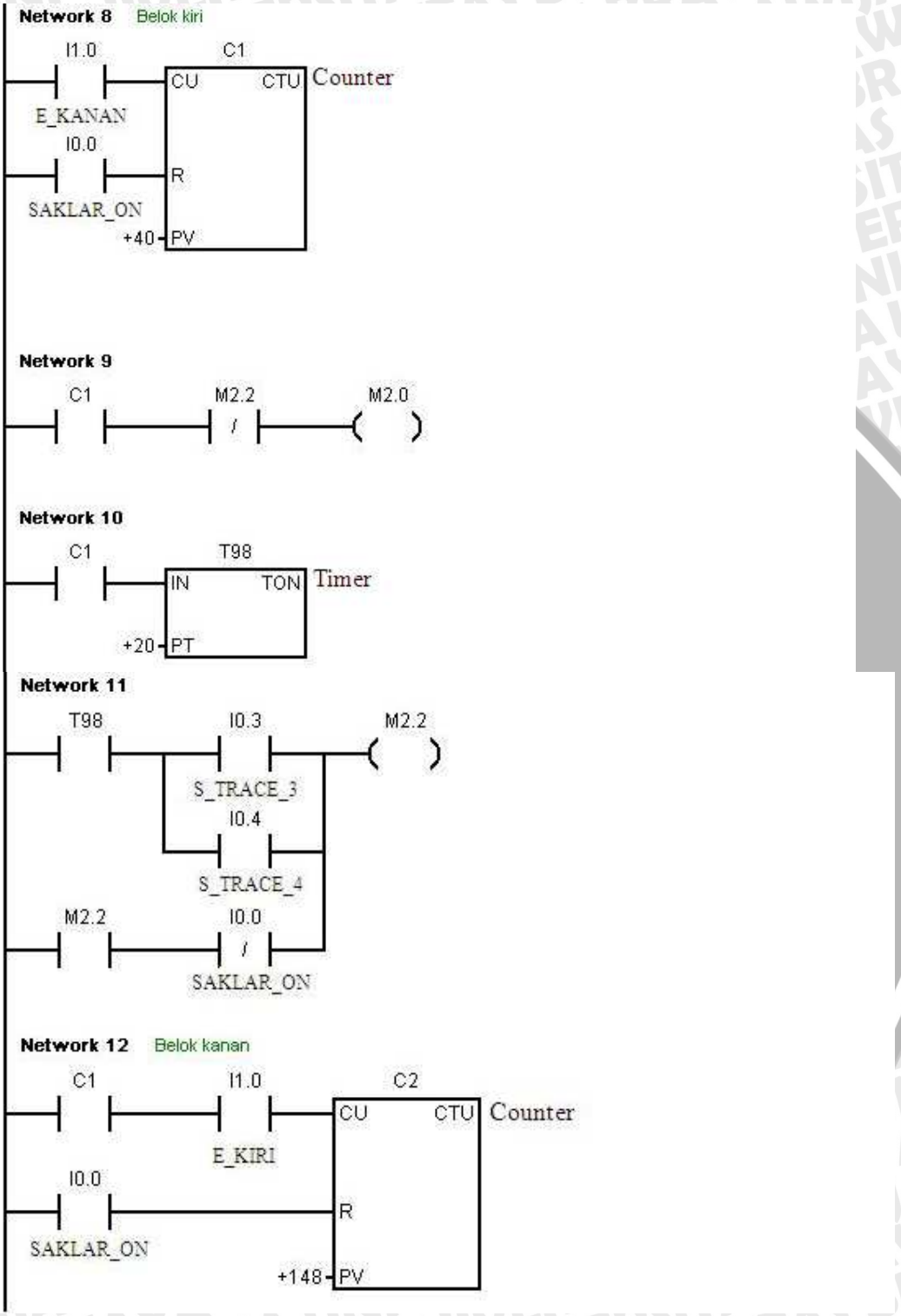
Alamat	Keterangan	Simbol
Q 00	Relai 1 penggerak motor kiri maju	MTR_KIRI
Q 02	PWM kiri	PWM_KIRI
Q 05	Relai 2 penggerak motor kanan maju	MTR_KANAN
Q 07	PWM kanan	PWM_KANAN

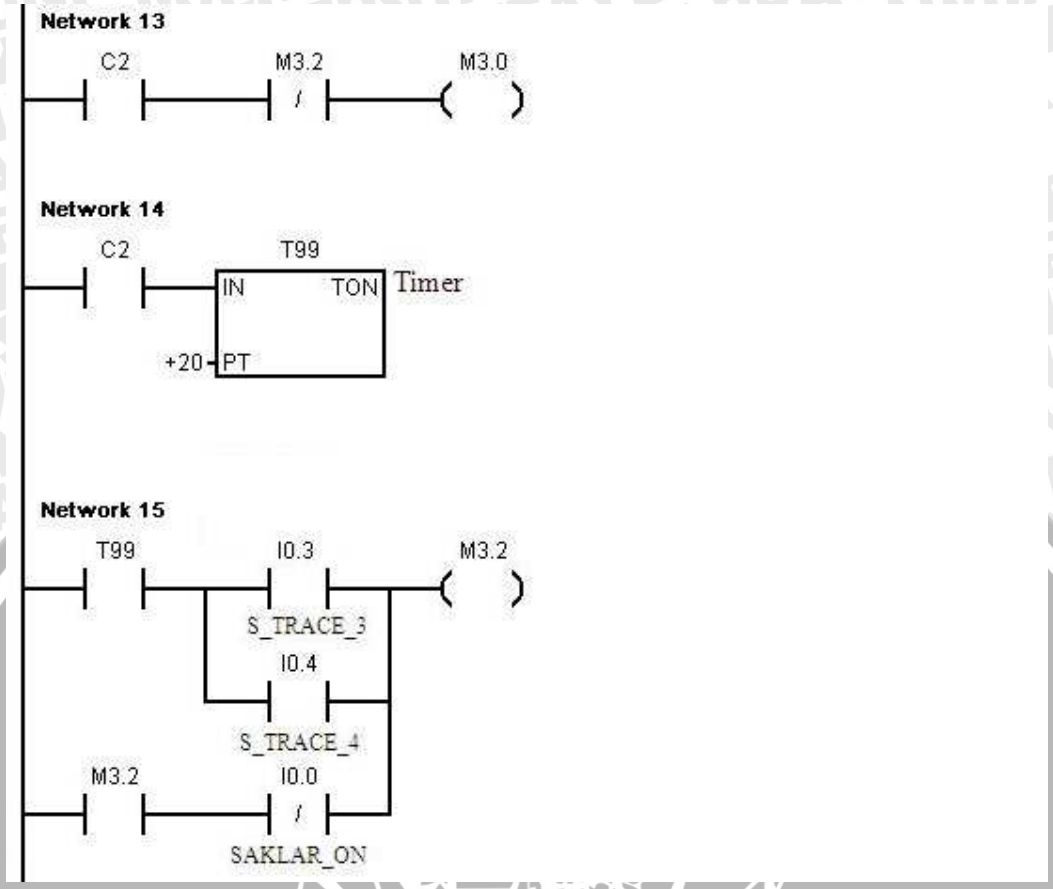
Dari data-data yang telah ditentukan dalam tabel, maka dapat dibuat program *ladder diagram* yang menggambarkan proses kerja sistem secara keseluruhan, yaitu sebagai berikut :


Network 1 Pergerakan robot maju dengan aktifasi lurus tanpa belok mengikuti garis hitam dengan tebal 3 cm
I0.0
SAKLAR_ON
M0.0
M0.2
M0.1
M0.1
M0.0
M0.1
M0.2
M0.2
Network 2
M0.1
M0.2
M0.0
M0.0
Network 3 Tracer kanan
T65
I0.1
S_TRACE_1
I0.2
S_TRACE_2
T33
IN
TON
+4
PT
I0.1
S_TRACE_1
I0.2
S_TRACE_2
T34
IN
TON
+5
PT
I0.1
S_TRACE_1
I0.2
S_TRACE_2
T35
IN
TON
+6
PT


I0.1
I0.2
T36
IN
TON
S_TRACE_1
S_TRACE_2
+7
PT
T65
IN
TON
+10
PT
Network 4
Tracer kiri
T97
I0.5
I0.6
T66
IN
TON
S_TRACE_5
S_TRACE_6
+8
PT
I0.5
I0.6
T67
IN
TON
S_TRACE_5
S_TRACE_6
+9
PT
I0.5
I0.6
T68
IN
TON
S_TRACE_5
S_TRACE_6
+9
PT
T97
IN
TON
+10
PT


Network 5 PWM kanan
M0.0
T33
T34
T35
M3.0
Q0.7
PWM_KANAN
I0.1
I0.6
S_TRACE_1
S_TRACE_6
C1
T98
Network 6 PWM kiri
M0.0
T66
T67
T68
M2.0
Q0.2
PWM_KIRI
I0.1
I0.6
S_TRACE_1
S_TRACE_6
C2
T99
Network 7 Motor maju
M0.0
M0.7
Q0.0
MTR_KIRI
Q0.5
MTR_KANAN


Network 8 Belok kiri
I1.0
E_KANAN
C1
CU
CTU
Counter
I0.0
SAKLAR_ON
R
+40
PV
Network 9
C1
M2.2
/
M2.0
Network 10
C1
T98
IN
TON
Timer
+20
PT
Network 11
T98
I0.3
S_TRACE_3
M2.2
I0.4
S_TRACE_4
M2.2
I0.0
/
SAKLAR_ON
Network 12 Belok kanan
C1
I1.0
E_KIRI
C2
CU
CTU
Counter
I0.0
SAKLAR_ON
R
+148
PV


Network 13
C2
M3.2
/
M3.0
Network 14
C2
T99
IN
TON
Timer
+20
PT
Network 15
T99
I0.3
M3.2
S_TRACE_3
I0.4
S_TRACE_4
M3.2
I0.0
/
SAKLAR_ON

BAB V
PENGUJIAN DAN ANALISIS SISTEM

Setelah menentukan piranti *input*/*output* dari sistem robot DDMR dan memprogram PLC dengan piranti pemrograman, maka dapat disajikan hasilnya dengan langsung mengaplikasikan melalui *hardware* yang telah dibuat atau dapat dilihat secara langsung dari output PLC yang berupa LED dan juga pergerakan robot pada lintasan yang telah ditentukan.

Pengujian perangkat lunak dari proses pergerakan robot ini bertujuan agar diketahui perangkat lunak (*software*) yang telah dibuat sesuai dengan perencanaan awal. Disamping itu pengujian ini bertujuan untuk mengetahui kelemahan / kekurangan dari *software* tersebut sehingga dapat disempurnakan.

Pengujian yang dilakukan terhadap sistem yang telah dibuat dapat dibagi menjadi beberapa tahapan, diantaranya:

- Pengujian karakteristik kecepatan putaran motor (PWM)
- Pengujian sensor pengikut garis (*line follower)* dan sensor putaran (*encoder*)
- Pengujian robot saat melakukan tracing dengan menggunakan sensor pengikut garis (*line follower)*
- Pengujian robot saat melakukan tracing lurus dengan aktifasi belok menggunakan sensor *line follower* dan sensor putaran (*encoder*)
- Pengujian sistem secara keseluruhan

5.1 Pengujian Karakteristik Kecepatan Putaran Motor (PWM)

5.1.1 Tujuan Pengujian

Tujuan dari pengujian karakteristik kecepatan putaran motor (PWM) adalah untuk mengetahui karakteristik masing-masing motor mobilitas sehingga hasilnya dapat digunakan untuk menentukan prosentase PWM yang akan dipakai pada tiap-tiap motor dalam upaya mendapatkan sistem pergerakan yang maksimal.

5.1.2 Prosedur Pengujian

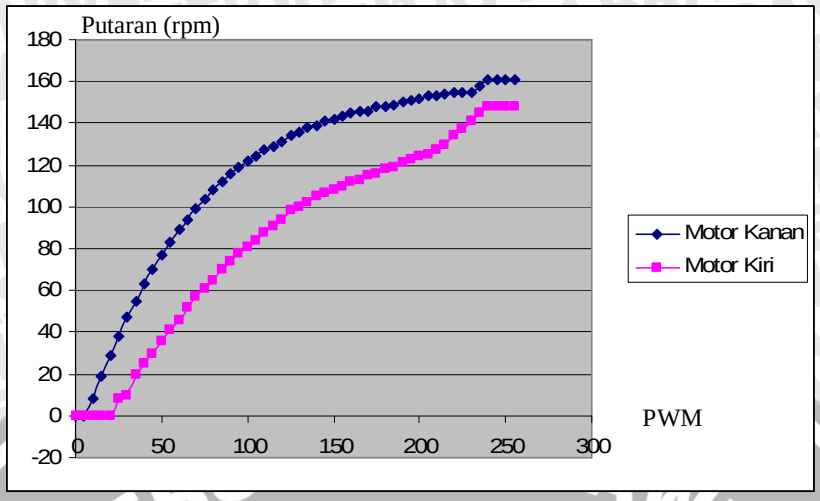
Prosentase PWM didapatkan berdasarkan pengujian karakteristik kecepatan putaran pada motor kanan dan kiri dengan keadaan robot tidak sedang

dijalankan. Pengujian dilakukan dengan memberikan nilai PWM secara bertahap kelipatan 5 dimulai dari 0 sampai 255 kemudian dilihat hasil putaran (rpm) motor tersebut.

5.1.3 Hasil Pengujian dan Analisis

Tabel 5.1. Hasil Pengujian Karakteristik Kecepatan Putaran Motor (PWM)

NILAI PWM	M. Kanan (rpm)	M. Kiri (rpm)	NILAI PWM	M. Kanan (rpm)	M. Kiri (rpm)
0	0	0	130	136	100
5	0	0	135	138	102
10	8	0	140	139	105
15	19	0	145	141	107
20	29	0	150	142	108
25	38	8	155	143	110
30	47	10	160	145	112
35	55	20	165	146	113
40	63	25	170	146	115
45	70	30	175	148	116
50	77	36	180	148	118
55	83	41	185	149	119
60	89	46	190	150	121
65	94	52	195	151	123
70	99	57	200	152	124
75	104	61	205	153	125
80	108	65	210	153	127
85	112	70	215	154	130
90	116	74	220	155	134
95	119	78	225	155	137
100	122	81	230	155	141
105	124	84	235	158	145
110	127	88	240	161	148
115	129	91	245	161	148
120	131	94	250	161	148
125	134	98	255	161	148



Gambar 5.1 Grafik Pengujian Karakteristik Kecepatan Putaran Motor (PWM)

Dari hasil pengujian terlihat bahwa kecepatan motor kanan dan motor kiri tidak sama. Saat diberikan PWM 255 (100%) motor kanan berputar sebesar 161 rpm sedangkan motor kiri 148 rpm. Bahkan ketika diberi PWM 0-20 motor kiri belum aktif untuk berputar. Maka untuk menentukan prosentase PWM pada masing-masing motor kanan dan kiri diambil rpm kedua motor yang memiliki karakteristik putaran yang sama. Atau untuk menentukan prosentase PWM dapat dicari berdasarkan rumus berikut ini :

$$\%PWM = \frac{a}{b} x100\%$$

Keterangan :

a = PWM yang digunakan

b = jumlah PWM

Maka berdasarkan pada pengujian karakteristik kecepatan putaran motor diatas, dapat ditentukan prosentase PWM yang digunakan pada motor, yang dapat dilihat pada tabel 5.2 dibawah ini.

Tabel 5.2. PWM yang digunakan pada motor DC kanan-kiri

Posisi sensor yang terkena garis	PWM Motor DC kanan	Prosentase PWM kanan	PWM Motor DC kiri	Prosentase PWM Kiri
Semua sensor tidak terkena	180	70,59 %	255	100 %
3 dan 4	180	70,59 %	255	100 %
4 dan 5	200	78,43 %	255	100 %
5 dan 6	220	86,27 %	255	100 %

6	245	94,12 %	255	100 %
2 dan 3	135	52,94 %	255	100 %
1 dan 2	115	45,10 %	255	100 %
1	100	39,22 %	255	100 %
Semua sensor terkena	180	70,59 %	255	100 %

Data-data pada tabel 5.2 merupakan acuan yang dijadikan referensi untuk menentukan PWM di dalam pemrograman PLC menggunakan *ladder diagram*. Di dalam pemrograman *ladder diagram* menggunakan PLC SIEMENS S-7200 untuk memfasilitasi PWM digunakan fasilitas *Timer* yang mempunyai ordo *time value* 10ms (T33-36 dan T97-100). Semakin kecilnya nilai dari ordo *time value* diharapkan dapat menghasilkan kualitas PWM yang baik. Fasilitas inilah yang nantinya akan digunakan untuk menentukan pengaturan nilai PWM. Maka berdasarkan data tersebut diatas, dapat ditentukan prosentase PWM yang digunakan di dalam pemrograman PLC (*ladder diagram*). yang dapat dilihat pada tabel 5.3 di bawah ini.

Tabel 5.3. PWM yang digunakan pada motor DC kanan-kiri di dalam pemrograman PLC (*ladder diagram*)

Posisi sensor yang terkena garis	Prosentase PWM Motor DC kanan	Prosentase PWM Motor DC kiri
Semua sensor tidak terkena	70%	100 %
3 dan 4	70 %	100 %
4 dan 5	80 %	100 %
5 dan 6	90 %	100 %
6	90 %	100 %
2 dan 3	50 %	100 %
1 dan 2	50 %	100 %
1	40 %	100 %
Semua sensor terkena	70 %	100 %

Dalam tabel 5.3 terlihat bahwa prosentase PWM yang digunakan tidak sama persis dengan referensi karakteristik kecepatan putaran motor (PWM) pada tabel 5.2. Hal ini dikarenakan untuk fasilitas pada PLC tidak dapat digunakan untuk pengisian bilangan dibelakang koma (desimal). Maka di dalam

pemrograman *ladder diagram* pada PLC PWM yang diberikan dibulatkan menjadi bilangan bulat yang paling mendekati. Tentunya hal ini akan mempermudah didalam pemrograman pada PLC (*ladder diagram*) dengan tetap tidak mengurangi efektifitas pergerakan robot.

5.2 Pengujian sensor pengikut garis (*line follower)* dan sensor putaran (*encoder*)

5.2.1 Tujuan Pengujian

Pengujian ini dimaksudkan untuk mengetahui apakah semua sensor pengikut garis (line follower) dan sensor putaran (*encoder*) dapat berfungsi dengan baik. Sehingga sistem dapat bekerja sesuai dengan yang direncanakan.

5.2.2 Prosedur Pengujian

Pengujian ini dilakukan dengan mengukur tegangan keluaran dari masing-masing sensor pengikut garis (*line follower*) ketika mendeteksi garis hitam pada lintasan dan ketika mendeteksi lantai. Untuk sensor *line follower* dapat dikatakan dalam kondisi baik untuk keluaran sinyalnya jika memiliki selisih tegangan konstan kurang lebih sebesar 5 volt antara logika Low dan logika Highnya. Pada aplikasi pemrograman sensor dianggap aktif saat logika Low. Sedangkan pada sensor encoder dengan mengukur keluaran sensor saat mendeteksi lubang pada piringan dan pada saat tidak mendeteksi lubang.

5.2.3 Hasil Pengujian dan Analisis

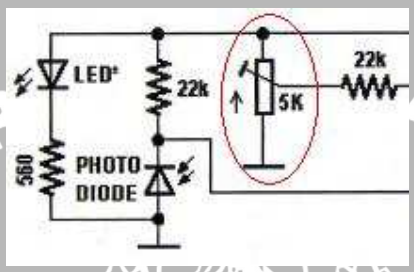
Tabel 5.4. Tegangan Output yang dihasilkan oleh sensor *Line Follower* saat mendeteksi lantai dan garis hitam

Sensor	Tegangan Output Sensor (V)	
	Lantai*	Garis*
Sensor tracer 1	15,5	10,4
Sensor tracer 2	15,7	10,2
Sensor tracer 3	15,4	10.5
Sensor tracer 4	14,7	10.7
Sensor tracer 5	14,9	9,8

Sensor tracer 6	15,2	9,9
Kondisi Logika	High	Low

* Setelah melalui rangkaian *Hysterisis*

Dari hasil pengujian di atas terlihat bahwa sensor memiliki rata-rata tegangan keluaran yang konstan. Hal tersebut tidak terlepas dari adanya rangkaian pembanding atau yang disebut rangkaian *Hysterisis* yang dapat dilihat pada gambar berikut ini :



Gambar 5.2. Rangkaian Hysterisis

Rangkaian tersebut dapat mengatur kepekaan sensor terhadap gelap atau putihnya garis, dengan mengatur VR 5kΩ. Dengan mengatur VR ini maka akan didapatkan tegangan masukan menuju photodioda yang sama dengan tegangan masukan Vcc.

Sensor Putaran (*encoder*)

Tabel 5.5. Tegangan Output yang Dihasilkan oleh Sensor Encoder kanan

Kondisi	Vout sensor (Volt)	Vout sensor (Volt)*
Tidak terkena lubang piringan	0,9	9,7
Terkena lubang piringan	4,9	15,6

* Keterangan : Setelah melalui rangkaian pengkondisi sinyal

Tabel 5.6. Tegangan Output yang Dihasilkan oleh Sensor Encoder kiri

Kondisi	Vout sensor (Volt)	Vout sensor (Volt)*
Tidak terkena lubang piringan	0,9	9,7
Terkena lubang piringan	4,8	15,4

* Keterangan : Setelah melalui rangkaian pengkondisi sinyal

5.3 Pengujian robot saat melakukan tracing dengan menggunakan sensor pengikut garis (*line follower)*

5.3.1 Tujuan Pengujian

Maksud dari pengujian ini adalah untuk mengetahui kemampuan mobilitas robot untuk melewati lintasan-lintasan dengan sudut-sudut tertentu yang sudah ditentukan sebelumnya.

5.3.2 Prosedur Pengujian

Pengujian ini dilakukan dengan menjalankan program (*ladder diagram*) yang telah dirancang sebelumnya untuk melakukan *tracing* dengan didasari pemberian prosentase PWM pada motor DC kanan dan kiri. Pengujian tersebut dilakukan dengan menjalankan robot pada lintasan lurus kemudian lintasan tersebut diubah berdasar sudut lintasan (30^0, 40^0, 50^0, dan 60^0).

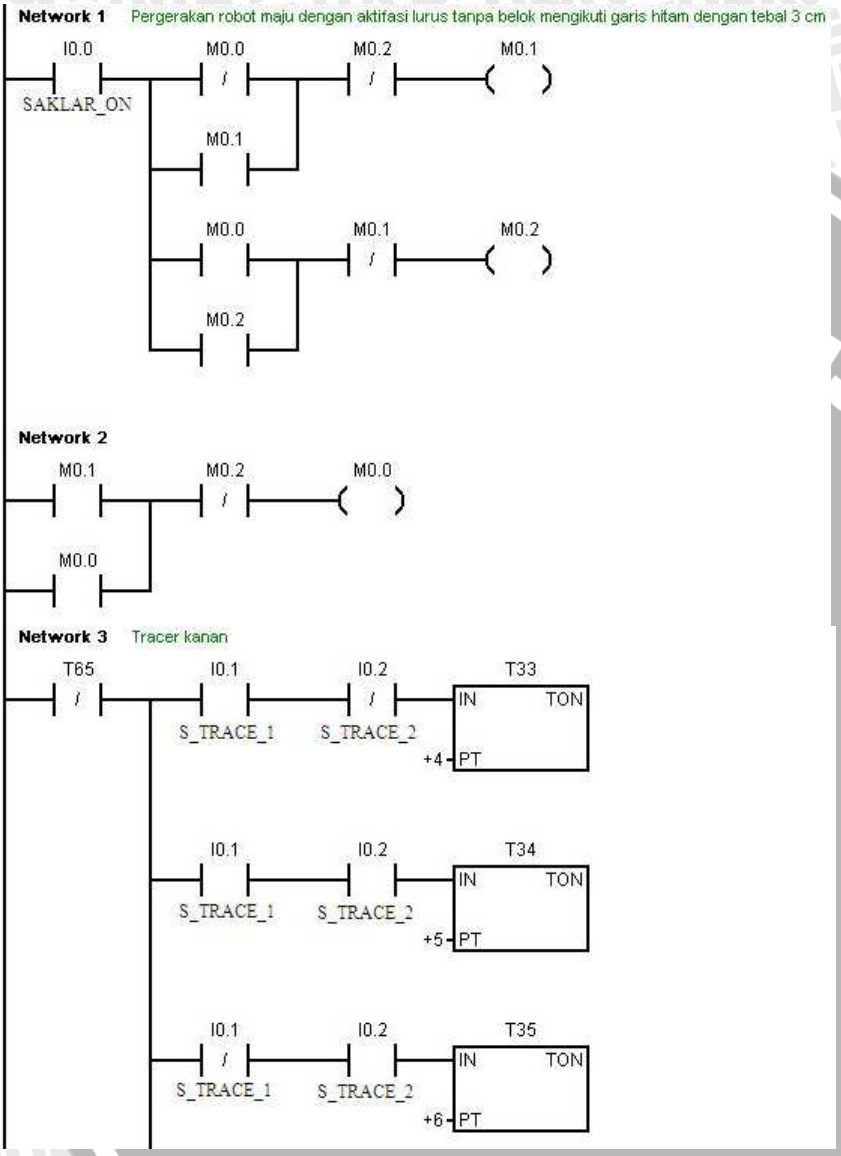
5.3.3 Hasil Pengujian dan Analisis

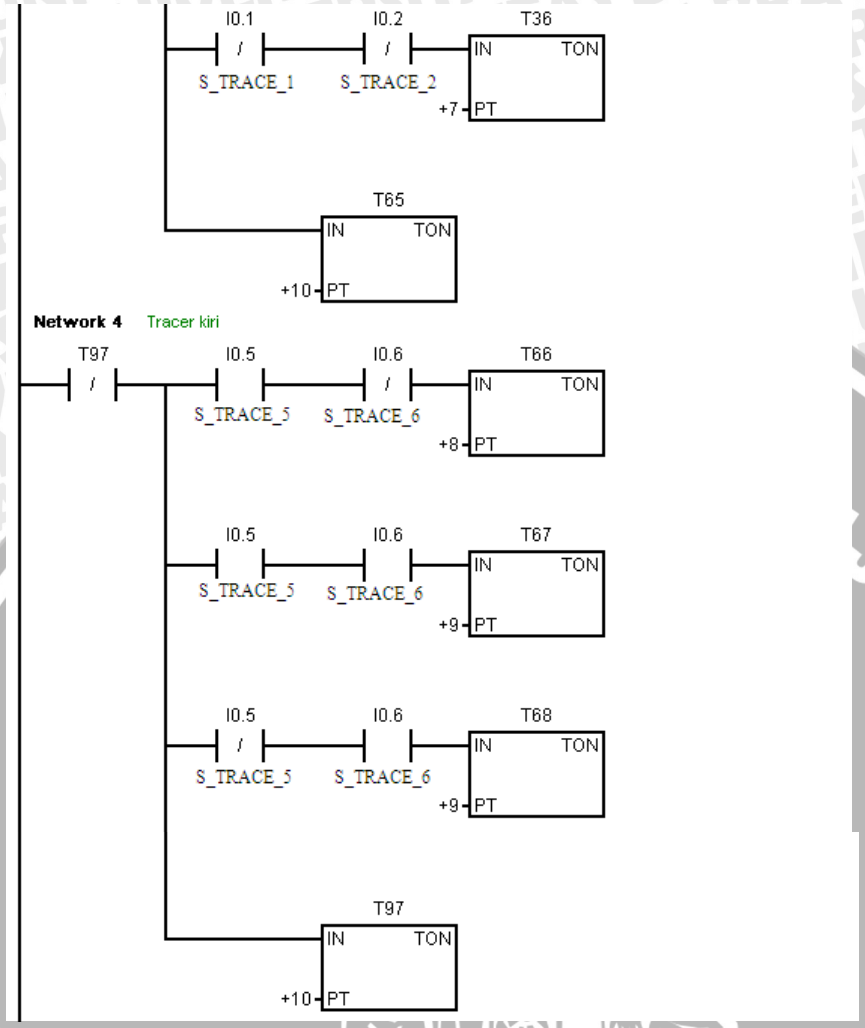
Tabel 5.7. Hasil Pengujian *Tracing* Robot berdasarkan sudut

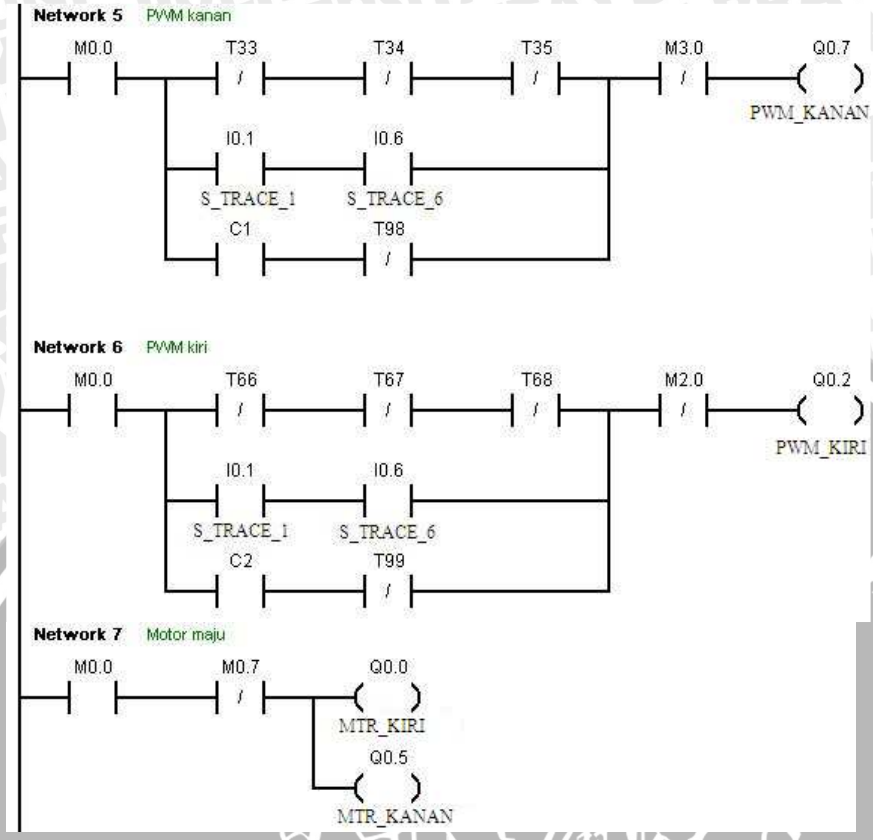
Sudut lintasan	Kondisi
0^0	Berhasil
30^0	Berhasil
40^0	Berhasil
50^0	Berhasil
60^0	Berhasil
$>60^0$	Gagal

Dari hasil pengujian pada tabel 5.5 terlihat bahwa robot dapat melewati lintasan-lintasan yang telah diatur sesuai besaran sudut lebih kecil samadengan 60^0, hal ini menunjukkan program yang telah dibuat dapat berjalan sesuai dengan yang diharapkan. Sedangkan kegagalannya melewati sudut $>60^0$ dikarenakan sensor garis (*line follower*) yang berjumlah 6 buah ini tidak dapat membaca sudut $>60^0$ sehingga tidak dapat mengkondisikan motor kanan dan kiri untuk mengikuti lintasan tersebut. Agar dapat melewati sudut $>60^0$ maka akan diatasi dengan mode belok menggunakan sensor garis (*line follower*) dan sensor putaran (*encoder*).

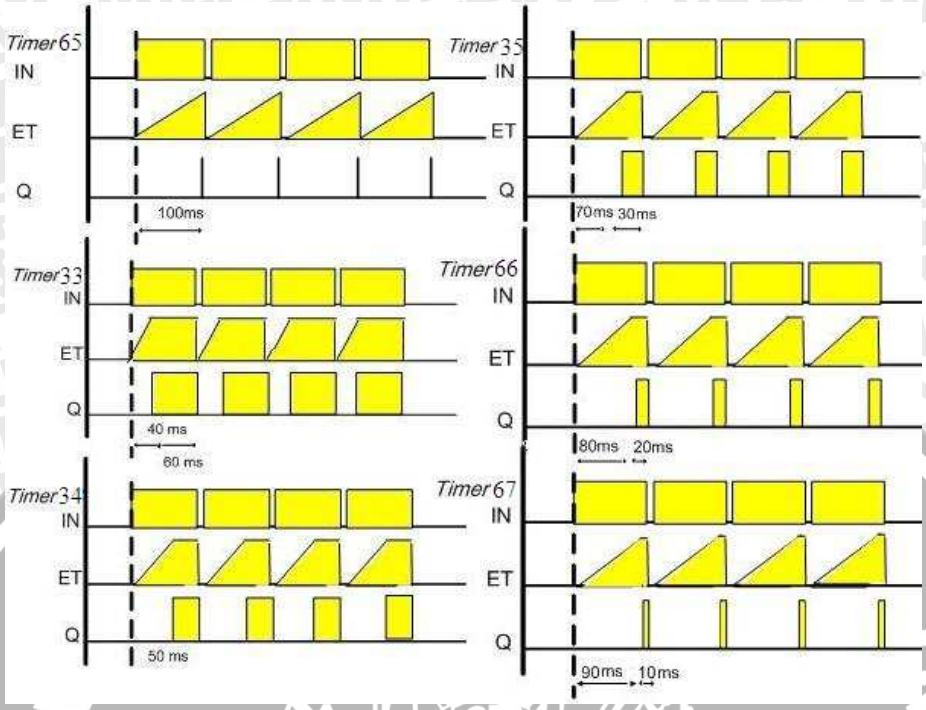
Program *Ladder diagram* sensor *line follower* untuk kedua motor (motor kanan dan motor kiri) saat *tracing* adalah sebagai berikut:




I0.1
/
S_TRACE_1
I0.2
/
S_TRACE_2
T36
IN
TON
+7
PT
T65
IN
TON
+10
PT
Network 4
Tracer kiri
T97
/
I0.5
/
S_TRACE_5
I0.6
/
S_TRACE_6
T66
IN
TON
+8
PT
I0.5
S_TRACE_5
I0.6
S_TRACE_6
T67
IN
TON
+9
PT
I0.5
/
S_TRACE_5
I0.6
S_TRACE_6
T68
IN
TON
+9
PT
T97
IN
TON
+10
PT



Pada *ladder diagram* di atas , masing-masing sensor untuk *tracer* kanan dan kiri (Network 3-4) lebih tepatnya yaitu sensor tracer 1,2,5, dan 6 diberikan PWM dalam pemrograman melalui TON33, TON34, TON35, TON36, TON66, TON67, dan TON68. Sedangkan TON65 dan TON97 digunakan sebagai referensi frekuensi yang digunakan dalam penggunaan PWM Berdasarkan pemberian prosentase pada masing-masing posisi sensor bila membaca garis jalur (lihat pada Tabel 5.3) sehingga didapatkan hasil tracing yang baik. Hal ini dikarenakan program yang dirancang dikondisikan untuk menyamakan kondisi motor DC kanan dan kiri agar memiliki kecepatan putar yang tidak berbeda terlalu jauh. *Timing diagram* dari pemberian prosentase PWM (40%, 50%, 70%, 80%, dan 90%) berdasarkan *ladder diagram* di atas dapat ditunjukkan pada gambar 5.3. *Timer* 65 berfungsi sebagai penentu frekuensi yang digunakan.

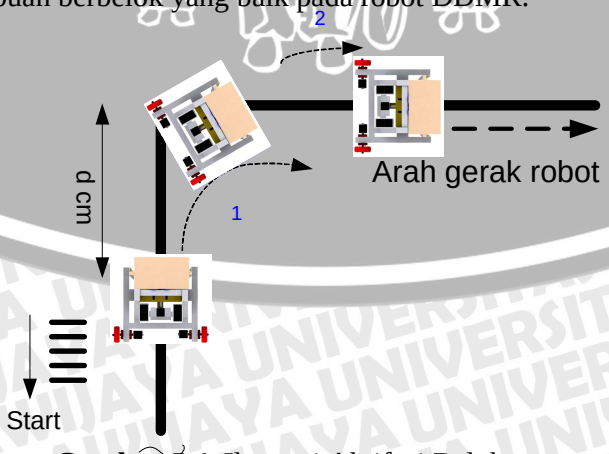


Gambar 5.3. *Timing Diagram* Robot *Tracing*

5.4 Pengujian robot saat melakukan tracing pada aktifasi belok menggunakan sensor *line follower* dan sensor putaran (*encoder*)

5.4.1 Tujuan Pengujian

Pada Bab perancangan telah ditentukan beberapa nilai d untuk aktifasi belok, dimana ”d” adalah jarak batas dari jarak maksimum agar robot saat berputar berada tepat di lintasannya kembali (cm). Tujuan pengujian ini adalah untuk mengetahui dengan nilai ”d” berapa saja yang masih mendukung agar diperoleh kemampuan berbelok yang baik pada robot DDMR.



Gambar 5.4. Ilustrasi Aktifasi Belok

5.4.2 Prosedur Pengujian

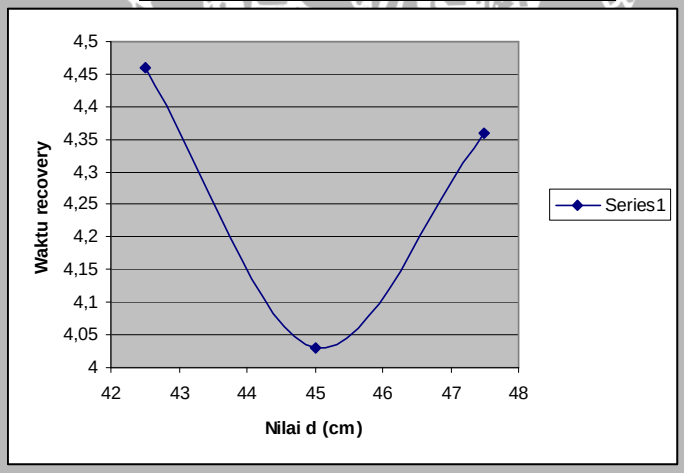
Pengujian dilakukan dengan mengubah-ubah posisi robot terhadap belokan kemudian robot dijalankan untuk dapat melewati belokan tersebut. Tolok ukur keberhasilan dari pengujian ini meliputi waktu tempuh dan apakah robot dapat melewati belokan yang telah ditentukan, dalam percobaan ini digunakan belokan dengan sudut 90^0.

5.4.3 Hasil Pengujian dan Analisis

Berikut tabel hasil pengujian untuk mendapatkan nilai d (berdasar perancangan) :

Tabel 5.8. Hasil Pengujian Nilai d

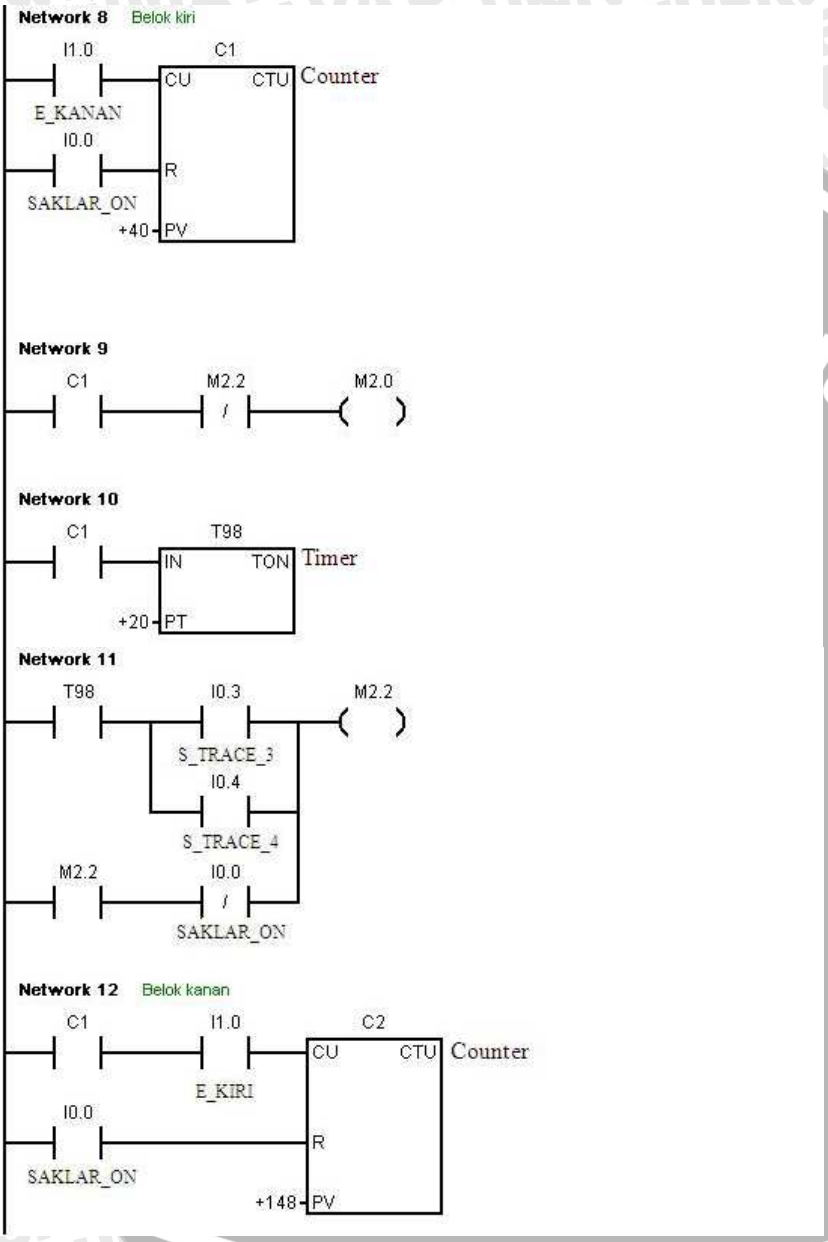
Nilai d (cm)	Waktu untuk kembali ke lintasan (detik)
40	gagal
42,5	4,46
45	4,03
47,5	4,36
50	gagal

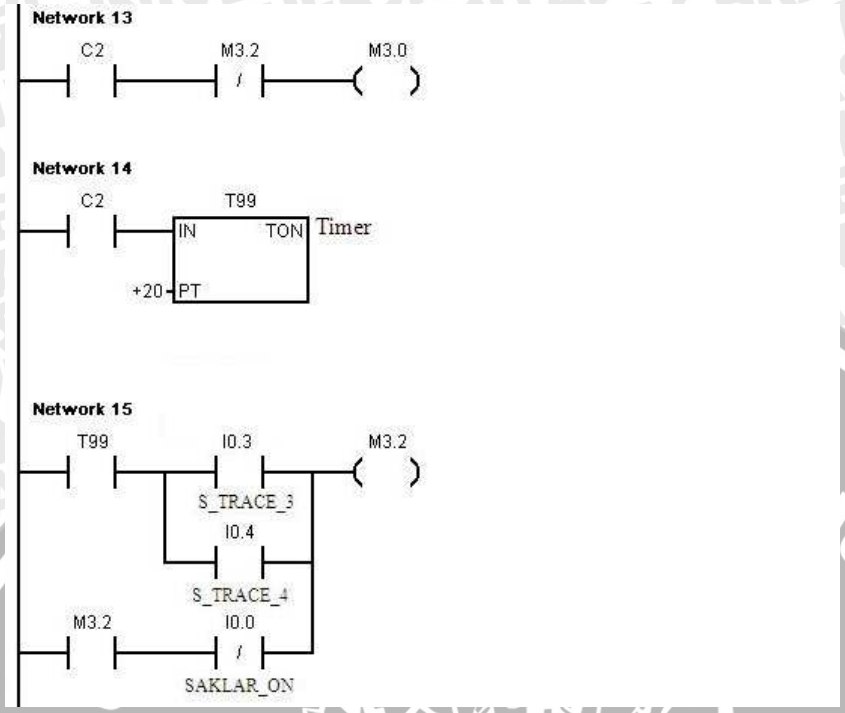


Gambar 5.5. Grafik Pengujian Nilai d

Dari grafik pengujian nilai d terlihat bahwa ada nilai d tertentu yang memiliki waktu respon yang cepat untuk kembali ke lintasan setelah berbelok. Dengan demikian maka nilai d inilah yang akan dijadikan referensi untuk pergerakan robot pada saat berbelok di daerah perempatan lintasan. Dengan menggunakan instruksi *counter* untuk memfungsikan *encoder* maka akan mempermudah untuk proses robot belok kanan ataupun kiri.

Program *Ladder diagram encoder* untuk kedua motor (motor kanan dan motor kiri) saat *tracing* adalah sebagai berikut:





Pada *Ladder diagram* di atas terlihat bahwa saat sensor *encoder* kanan aktif maka akan mengaktifkan *counter* C1 untuk menghitung pulsa. Perhitungan pulsa ini berdasar pada perancangan aktifasi belok kiri yang telah dilakukan sebelumnya. Saat *counter* aktif maka akan mengaktifkan *Marke* M2.0 yang akan dijadikan kondisi untuk menentukan kondisi PWM kanan untuk aktif atau tidak. Aktifnya *counter* C1 ini juga akan mengaktifkan *timer* 98 yang akan berfungsi untuk menghentikan motor kanan dan kiri selama 2 detik setelah melakukan proses belok. Hal ini ditujukan agar setelah proses belok sensor (3 dan 4) bisa tepat berada di lintasan garis yang bila itu terpenuhi maka robot akan dapat melakukan *tracing* dengan baik. Hal ini juga berlaku untuk proses belok kanan.

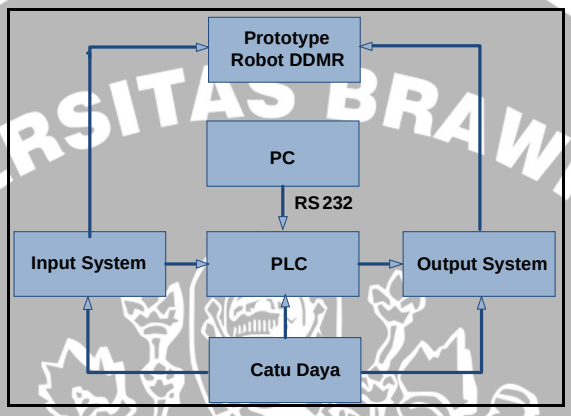
5.5 Pengujian dan Analisis Sistem Secara Keseluruhan

5.5.1 Tujuan Pengujian

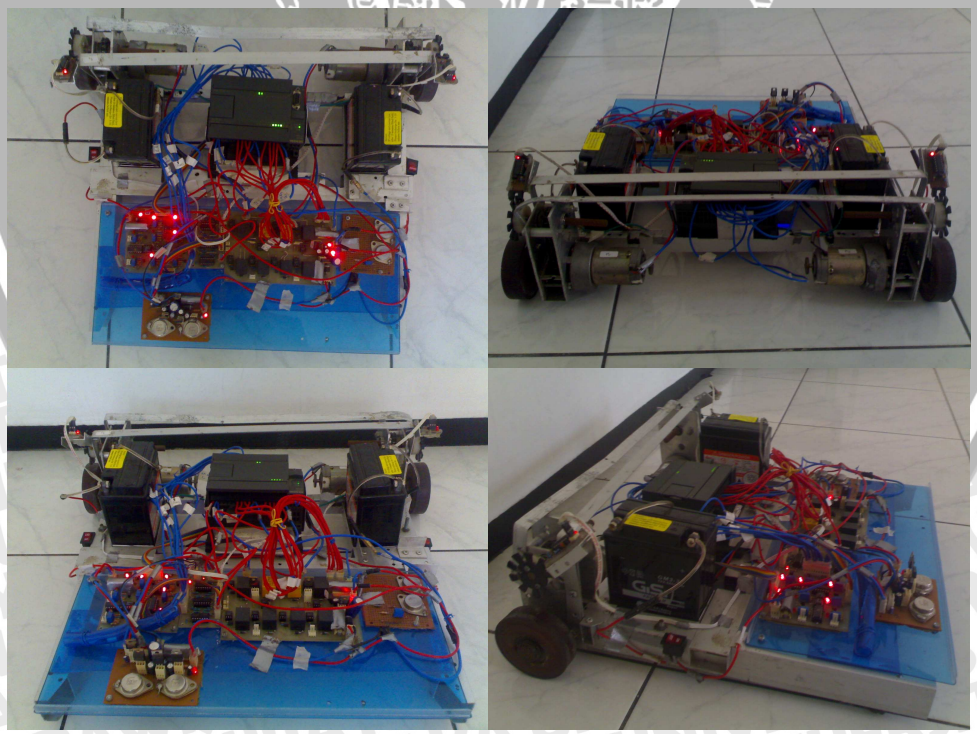
Tujuan pengujian sistem secara keseluruhan adalah untuk melihat apakah sistem yang telah dibuat baik *hardware* ataupun *software* yang berupa *ladder diagram* dapat berjalan dengan baik sesuai dengan tujuan awal.

5.5.2 Prosedur Pengujian

Parameter keberhasilan pengujian ini adalah jika waktu tempuh robot melewati lintasan yang telah ditentukan adalah konstan atau semakin cepat. Dengan alasan, jika waktu tempuh relatif konstan maka sistem dapat berjalan dengan baik sesuai dengan program *ladder diagram* yang telah dirancang. Pengujian dilakukan secara menyeluruh pada blok sistem dengan menguji robot pada lintasan. Adapun blok sistem secara keseluruhan adalah sebagai berikut :

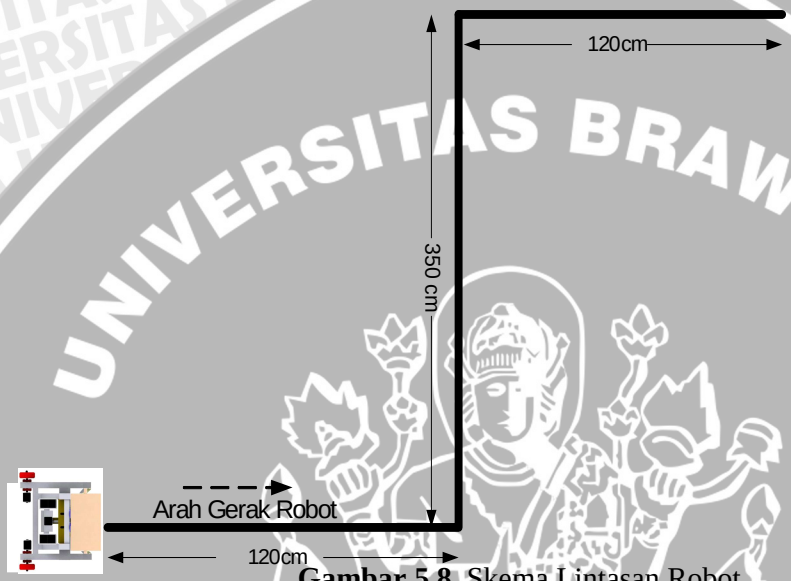


Gambar 5.6. Blok Pengujian Sistem Secara Keseluruhan



Gambar 5.7. Realisasi Prototype Robot

Pengujian dilakukan dengan menjalankan program *ladder diagram* untuk sistem secara keseluruhan berdasar perancangan pada sistem dan mengamati pergerakan robot pada lintasan kemudian menghitung simpangan robot saat tracing pada lintasan. Kemudian menghitung waktu tempuh robot dari start sampai menyelesaikan lintasan. Skema lintasan robot dapat dilihat pada gambar 5.8.



Gambar 5.8. Skema Lintasan Robot

5.5.3 Hasil Pengujian dan Analisis

Dari pengujian untuk mengetahui waktu tempuh dan simpangan robot saat *tracing* didapatkan data waktu sebagai berikut :

- Lintasan Pertama Pada Jarak 120 cm

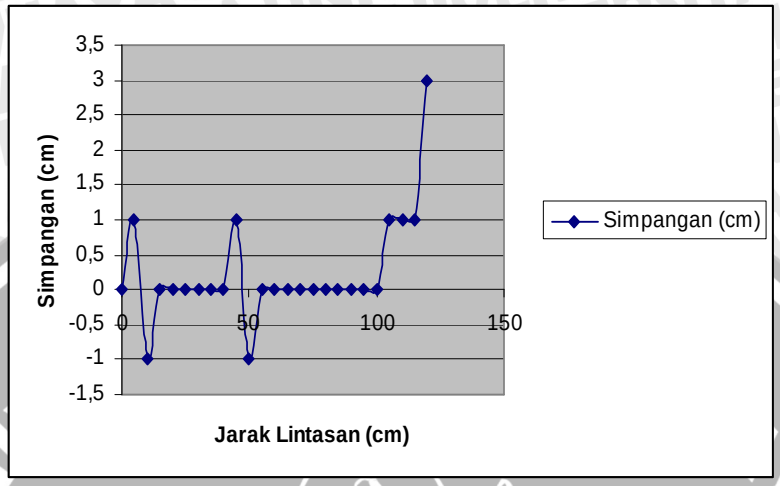
Tabel 5.9. Hasil Pengujian Untuk Mengetahui Simpangan Pada Lintasan Lurus Pertama Jarak 120 cm.

Jarak Lintasan (cm)	Simpangan (cm)
0	0
5	1
10	-1
15	0
20	0
25	0
30	0
35	0
40	0
45	0

60	0
65	0
70	0
75	0
80	0
85	0
90	0
95	0
100	0
105	1
110	1

50	-1	115	1
55	0	120	3

*ket : nilai simpangan (-) diberlakukan bila robot menyimpang ke kanan lintasan.



Gambar 5.9. Grafik Simpangan Tracing Robot Pada Lintasan Lurus Pertama Jarak 120 cm.

Dari pengujian terlihat bahwa pada awal pergerakan terjadi simpangan. Hal ini dikarenakan saat robot mulai bergerak di lintasan sensor akan mulai membaca data masukan yang berupa posisi robot terhadap lintasan. Karena motor DC yang digunakan ini tidak memiliki kecepatan yang sama maka saat awal-awal pergerakan akan terjadi simpangan. Saat Robot berada pada jarak 105 cm akan terjadi simpangan antara 1 dan 3 cm sampai pada jarak 120 cm. Hal ini dikarenakan pada jarak tersebut robot akan mulai melakukan aktifasi belok kiri.

Dari data pengujian dapat diketahui simpangan rata-rata yaitu :

$$\frac{\sum simpangan}{Nsampel} = \frac{\sum |simpangan|}{n} = \frac{9}{25} = 0.36cm$$

Error dapat dihitung dengan rumus ”simpangan rata-rata dibagi panjang lintasan dikalikan 100%” sehingga diperoleh hasil :

$$\frac{0.36}{120} \times 100\,\% = 0.3\,\% \text{.....error 1}$$

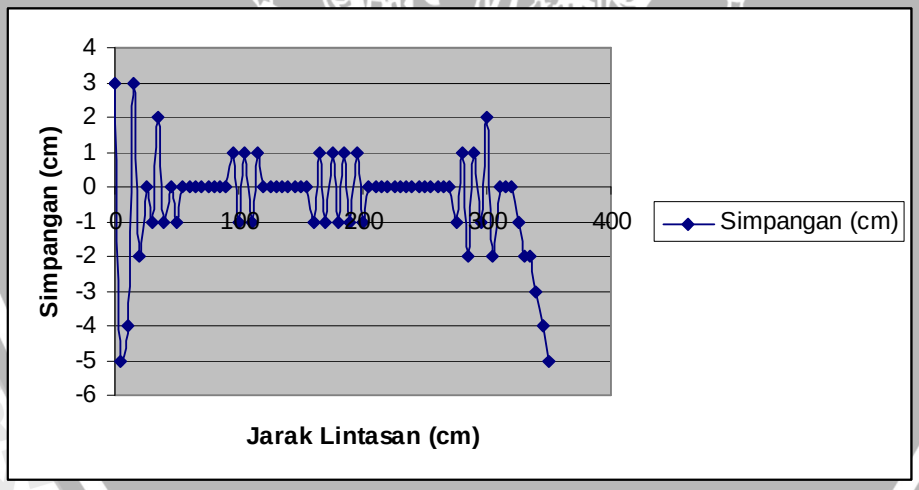
- Lintasan Kedua Pada Jarak 350 cm

Tabel 5.10. Hasil Pengujian Simpangan Pada Lintasan Lurus Jarak 350 cm.

Jarak Lintasan (cm)	Simpangan (cm)				
0	3	115	1	235	0
		120	0	240	0

5	-5	125	0	245	0
10	-4	130	0	250	0
15	3	135	0	255	0
20	-2	140	0	260	0
25	0	145	0	265	0
30	-1	150	0	270	0
35	2	155	0	275	-1
40	-1	160	-1	280	1
45	0	165	1	285	-2
50	-1	170	-1	290	1
55	0	175	1	295	-1
60	0	180	-1	300	2
65	0	185	1	305	-2
70	0	190	-1	310	0
75	0	195	1	315	0
80	0	200	-1	320	0
85	0	205	0	325	-1
90	0	210	0	330	-2
95	1	215	0	335	-2
100	-1	220	0	340	-3
105	1	225	0	345	-4
110	-1	230	0	350	-5

*ket : nilai simpangan (-) diberlakukan bila robot menyimpang ke kanan lintasan.



Gambar 5.10. Grafik Simpangan Tracing Robot Pada Lintasan Lurus Jarak 350 cm.

Dari pengujian didapatkan bahwa simpangan terbesar terjadi di jarak awal lintasan. Hal ini dikarenakan inilah jarak robot saat melakukan belok kiri sehingga posisi robot saat berbelok akan sedikit keluar lintasan. Robot pada jarak 325 cm

akan terjadi simpangan antara -1 dan -5 sampai pada jarak 350 cm. Hal ini dikarenakan pada jarak tersebut robot akan mulai melakukan aktifasi belok kanan.

Dari data pengujian dapat diketahui simpangan rata-rata yaitu :

$$\frac{\sum simpangan}{Nsampel} = \frac{\sum |simpangan|}{n} = \frac{63}{71} = 0.89cm$$

Error dapat dihitung dengan rumus ”simpangan rata-rata dibagi panjang lintasan dikalikan 100%” sehingga diperoleh hasil :

$$\frac{0.89}{350} \times 100\,\% = 0.254\,\% \text{.....error 2}$$

- Lintasan ketiga Pada Jarak 120 cm

Tabel 5.11. Hasil Pengujian Untuk Mengetahui Simpangan Pada Lintasan Lurus Jarak 120 cm Yang kedua.

Jarak Lintasan (cm)	Simpangan (cm)		
0	-5	60	0
5	-5	65	0
10	4	70	0
15	3	75	0
20	-2	80	0
25	0	85	0
30	-1	90	0
35	2	95	0
40	-1	100	0
45	0	105	0
50	-1	110	0
55	0	115	0
		120	0

*ket : nilai simpangan (-) diberlakukan bila robot menyimpang ke kanan lintasan.

Dari data pengujian dapat diketahui simpangan rata-rata yaitu :

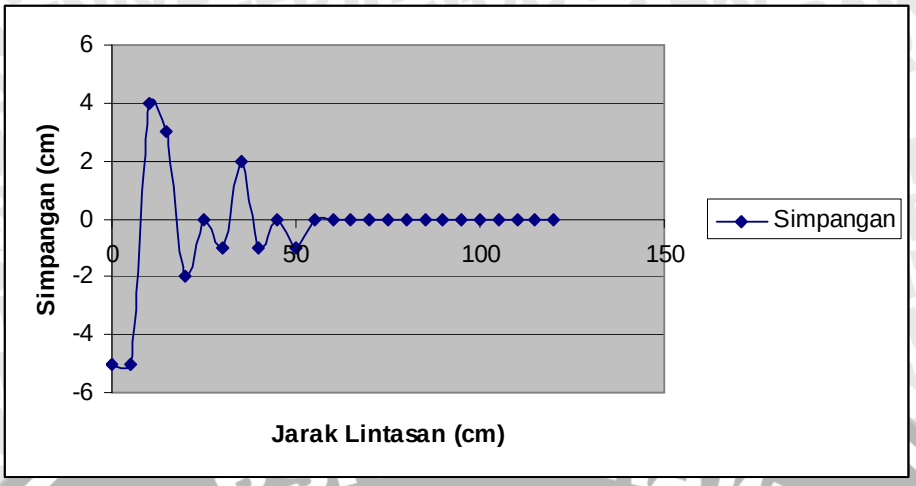
$$\frac{\sum simpangan}{Nsampel} = \frac{\sum |simpangan|}{n} = \frac{24}{25} = 0.96cm$$

Error dapat dihitung dengan rumus ”simpangan rata-rata dibagi panjang lintasan dikalikan 100%” sehingga diperoleh hasil :

$$\frac{0.96}{120} \times 100\,\% = 0.8\,\% \text{.....error 3}$$

Sehingga error total sama dengan error 1 + error 2 + error 3 yaitu :

0.3% + 0.254% + 0.8% = 1.354%



Gambar 5.11. Grafik Simpangan Tracing Robot Pada Lintasan Lurus Jarak 120 cm Kedua

- Waktu Tempuh Robot dari *start* sampai akhir

Tabel 5.12. Hasil Pengujian Waktu Tempuh

Percobaan Ke- n	Waktu Tempuh (detik)
1	37,35
2	38,48
3	39,50
4	36,40
5	37,74
6	36,20
7	37,35

$$t = \frac{37,35 + 38,48 + 39,50 + 36,40 + 37,74 + 36,20 + 37,35}{7}$$

$$= 37,15$$

Dari table 5.12 dapat diketahui bahwa waktu tempuh yang dibutuhkan robot untuk menyelesaikan lintasan tidak mengalami *fluktuasi* yang terlalu besar. Adanya *fruktuasi* tersebut dapat dipengaruhi oleh banyak faktor, yaitu *disturbance* dari luar maupun dari dalam sistem (catu daya, lintasan, mekanik,dll)). Waktu tempuh rata-rata dari pengujian tersebut adalah 37,15 detik.

Dari seluruh pengujian dapat disimpulkan bahwa sistem dapat berjalan dengan baik. Dengan ditandai waktu tempuh dari beberapa percobaan yang tidak

mengalami perubahan yang cukup besar (relatif konstan) dan berfungsinya masing-masing input terutama sensor sehingga sistem kontroller dapat berjalan dengan semestinya.

Dari hasil pengujian kerja sistem untuk I/O dapat dianalisa dalam bentuk tabel data input dan output dari PLC adalah sebagai berikut :

Tabel 5.13. Data Pengujian Input Output Sistem Secara Keseluruhan

INPUT	STATUS	INDIKASI	OUTPUT	STATUS	INDIKASI
I0.0	ON	Sistem Jalan			
I0.1 I0.2	ON	Terkena garis	Q0.0 Q0.7 Q0.5	ON	PWM kanan aktif mengurangi kecepatan motor kanan agar robot berjalan lurus
	OFF	Tidak terkena	-	-	-
I0.1	OFF	Tidak terkena	-	-	-
I0.2	ON	Terkena garis	Q0.5 Q0.7 Q0.0	ON	PWM kanan aktif mengurangi kecepatan motor kanan agar robot berjalan lurus
I0.5 I0.6	ON	Terkena garis	Q0.0 Q0.2 Q0.5	ON	PWM kiri aktif mengurangi kecepatan motor kiri agar robot berjalan lurus
	OFF	Tidak terkena	-	-	-
I0.5	OFF	Tidak terkena	-	-	-
I0.6	ON	Terkena garis	Q0.0 Q0.2 Q0.5	ON	PWM kiri aktif mengurangi kecepatan motor kiri agar robot berjalan lurus
I1.1	ON	Aktif krn indikasi dr sensor enkoder kanan	Q0.5 Q0.7 Q0.0	ON	PWM kanan aktif mengurangi kecepatan sehingga robot belok kanan
I1.0	ON	Aktif krn indikasi dr sensor enkoder kiri	Q0.5 Q0.7 Q0.0	ON	PWM kiri aktif mengurangi kecepatan sehingga robot belok kiri

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

1. Ada beberapa keadaan sensor line follower yang diabaikan yaitu pada pembacaan 000100, 001000, kondisi tersebut diasumsikan sama dengan kondisi 001100 yaitu pada saat robot tepat berada ditengah jalur, langkah ini membuat robot lebih stabil pada saat kecepatan maksimum.
2. Dalam aktifasi belok pengujian yang dilakukan mendapatkan nilai d sebesar ±45cm dimana dengan nilai tersebut, robot dapat berbelok dengan baik sehingga bisa kembali lagi ke lintasan lurus.
3. Waktu tempuh robot dari posisi start sampai menyelesaikan melewati lintasan memiliki waktu rata-rata 37,15 detik dari 7 kali percobaan dengan selisih waktu cukup kecil serta didukung oleh error yang kecil pula, sekitar 1,354%. Hal ini menunjukkan program *ladder* yang dibuat telah berhasil melakukan mobilitas yang baik pada robot DDMR.

6.2 Saran

1. Dalam pengembangan sistem pengendalian robot ini diperlukan PLC yang memiliki dimensi mikro dan memiliki fasilitas PWM dengan eksekusi waktu dalam *mikrosekon*. Dengan ukuran dimensi PLC yang mikro maka diharap akan mengurangi beban robot sehingga pergerakan robot mampu berjalan lebih cepat. Bila memiliki fasilitas PWM dalam mikrosekon maka pengaturan kecepatan motor dapat memperoleh hasil yang lebih baik dengan respon yang cepat.
2. Dalam pembuatan hardware pendukung (sensor) diperlukan tingkat ketelitian yang tinggi dalam menerima dan memberikan data. Hal ini diharapkan akan mempermudah kerja PLC dalam melakukan pengendalian.
3. Diperlukan suatu pengamanan terhadap kinerja sensor *line follower* terhadap pengaruh cahaya luar. Ada beberapa hal untuk mengatasi hal tersebut yaitu sebagai berikut :

- Dengan meng*cover* sensor *line follower* menggunakan pelindung dari kertas *carton* atau bahan yang memantulkan cahaya.
- Menambahkan tegangan offset pada rangkaian *hysterisis*.

4. Dengan menambah jumlah sensor *line follower* dan memposisikannya semakin rapat diharapkan dapat memperbaiki kinerja sensor tersebut, terutama untuk mendeteksi sudut lintasan yang lebar.

DAFTAR PUSTAKA

[ACH-07] Achmad, Balza. 2007. *"Pemrograman PLC Menggunakan Simulator"*. Yogyakarta: Andi.

[DAV-95] David, R. 1995. *"Grafcet: A Powerful Tool for Specification of Logic Controller*

[PIT-06] Pitowarno, Endro. 2006. *"Robotika, Disain, Kontrol, dan Kecerdasan Buatan*". Yogyakarta: Andi.

[PUT-04] Putra, Agfianto. 2004. *"PLC Konsep, Pemrograman, dan Aplikasi"*. Yogyakarta: Gaya Media.

[SET-06] Setiawan, I. 2006. *"Programmable Logic Controller (PLC) dan Teknik Perancangan Sistem Kontrol"*. Yogyakarta: Andi.

[SOE-97] Soemarwanto. 1997. *"Diktat Kuliah Dasar Konversi Energi Elektrik"*. Malang: Universitas Brawijaya

[SIM-03] _____. 2003. *"S7-200 Programmable Controller System Manual Edition 05"*.. _____: Siemens

[SIM-99] _____. 1999. *"Starter Kit SIEMENS S7-200 Edition 07"*.. _____: Siemens

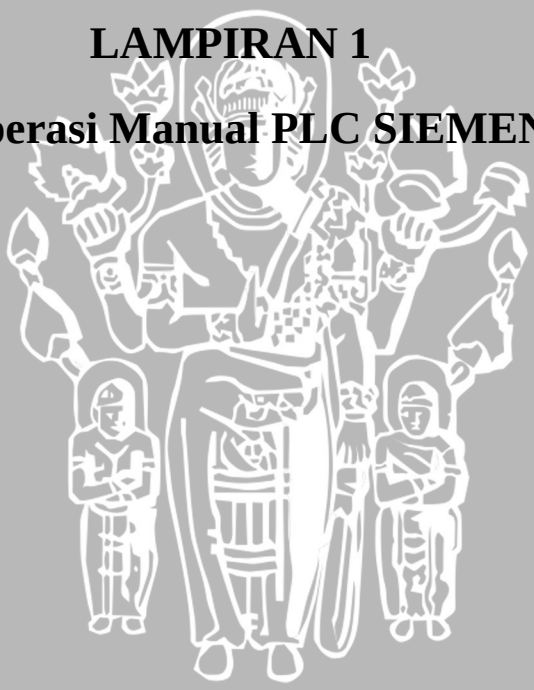
[SIM-00] _____. 2000. *"Micro System SIMATIC S7-200 Edition 01"*.. _____: Siemens

[WAS-04] Wasito, S. 2004. *"Kamus Elektronika"*. Jakarta: Gramedia Pustaka Utama

[ZUH-93] Zuhal. 1993. *"Dasar tenaga Listrik"*. Bandung: ITB.

LAMPIRAN 1

Datasheet Operasi Manual PLC SIEMENS S7-200



1

S7-200 CPU

The S7-200 CPU combines a microprocessor, an integrated power supply, input circuits, and output circuits in a compact housing to create a powerful Micro PLC. See Figure 1-1. After you have downloaded your program, the S7-200 contains the logic required to monitor and control the input and output devices in your application.

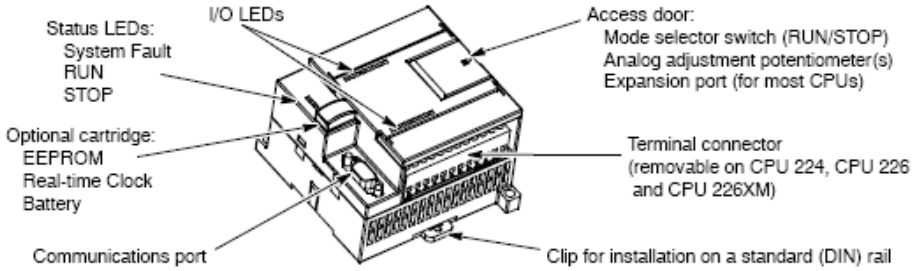


Figure 1-1 S7-200 Micro PLC

Siemens provides different S7-200 CPU models with a diversity of features and capabilities that help you create effective solutions for your varied applications. Table 1-1 briefly compares some of the features of the CPU. For detailed information about a specific CPU, see Appendix A.

Table 1-1 Comparison of the S7-200 CPU Models

Feature	CPU 221	CPU 222	CPU 224	CPU 226	CPU 226XM
Physical size (mm)	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	190 x 80 x 62	190 x 80 x 62
Program memory	4096 bytes	4096 bytes	8192 bytes	8192 bytes	16384 bytes
Data memory	2048 bytes	2048 bytes	5120 bytes	5120 bytes	10240 bytes
Memory backup	50 hours typical	50 hours typical	190 hours typical	190 hours typical	190 hours typical
Local on-board I/O	6 In/4 Out	8 In/6 Out	14 In/10 Out	24 In/16 Out	24 In/16 Out
Expansion modules	0 modules[1]	2 modules[1]	7 modules[1]	7 modules[1]	7 modules[1]
High-speed counters Single phase Two phase	 4 at 30 kHz 2 at 20 kHz	 4 at 30 kHz 2 at 20 kHz	 6 at 30 kHz 4 at 20 kHz	 6 at 30 kHz 4 at 20 kHz	 6 at 30 kHz 4 at 20 kHz
Pulse outputs (DC)	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz
Analog adjustments	1	1	2	2	2
Real-time clock	Cartridge	Cartridge	Built-in	Built-in	Built-in
Communications ports	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
Floating-point math	Yes				
Digital I/O image size	256 (128 in, 128 out)				
Boolean execution speed	0.37 microseconds/instruction				

[1] You must calculate your power budget to determine how much power (or current) the S7-200 CPU can provide for your configuration. If the CPU power budget is exceeded, you may not be able to connect the maximum number of modules. See Appendix A for CPU and expansion module power requirements, and Appendix B to calculate your power budget.

S7-200 Memory Ranges and Features

Table 6-1 Memory Ranges and Features for the S7-200 CPUs

Description		CPU 221	CPU 222	CPU 224	CPU 226	CPU 226XM
User program size		4096 bytes	4096 bytes	8192 bytes	8192 bytes	16384 bytes
User data size		2048 bytes	2048 bytes	5120 bytes	5120 bytes	10240 bytes
Process-image input register		I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7
Process-image output register		Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7
Analog inputs (read only)		--	AIW0 to AIW30	AIW0 to AIW62	AIW0 to AIW62	AIW0 to AIW62
Analog outputs (write only)		--	AQW0 to AQW30	AQW0 to AQW62	AQW0 to AQW62	AQW0 to AQW62
Variable memory (V)		VB0 to VB2047	VB0 to VB2047	VB0 to VB5119	VB0 to VB5119	VB0 to VB10239
Local memory (L)[1]		LB0 to LB63	LB0 to LB63	LB0 to LB63	LB0 to LB63	LB0 to LB63
Bit memory (M)		M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7
Special Memory (SM)		SM0.0 to SM179.7	SM0.0 to SM299.7	SM0.0 to SM549.7	SM0.0 to SM549.7	SM0.0 to SM549.7
Read only		SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7
Timers		256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)
Retentive on-delay	1 ms	T0, T64	T0, T64	T0, T64	T0, T64	T0, T64
	10 ms	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68
	100 ms	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95
On/Off delay	1 ms	T32, T96	T32, T96	T32, T96	T32, T96	T32, T96
	10 ms	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100
	100 ms	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255
Counters		C0 to C255	C0 to C255	C0 to C255	C0 to C255	C0 to C255
High-speed counters		HC0, HC3, HC4, and HC5	HC0, HC3, HC4, and HC5	HC0 to HC5	HC0 to HC5	HC0 to HC5
Sequential control relays (S)		S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7
Accumulator registers		AC0 to AC3	AC0 to AC3	AC0 to AC3	AC0 to AC3	AC0 to AC3
Jumps/Labels		0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
Call/Subroutine		0 to 63	0 to 63	0 to 63	0 to 63	0 to 127
Interrupt routines		0 to 127	0 to 127	0 to 127	0 to 127	0 to 127
Positive/negative transitions		256	256	256	256	256
PID loops		0 to 7	0 to 7	0 to 7	0 to 7	0 to 7
Ports		Port 0	Port 0	Port 0	Port 0, Port 1	Port 0, Port 1

[1] LB60 to LB63 are reserved by STEP 7-Micro/WIN, version 3.0 or later.

Table 6-2 Operand Ranges for the S7-200 CPUs

Access Method		CPU 221	CPU 222	CPU 224, CPU 226	CPU 226XM
Bit access (byte.bit)	I	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7
	Q	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7
	V	0.0 to 2047.7	0.0 to 2047.7	0.0 to 5119.7	0.0 to 10239.7
	M	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7
	SM	0.0 to 179.7	0.0 to 299.7	0.0 to 549.7	0.0 to 549.7
	S	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7
	T	0 to 255	0 to 255	0 to 255	0 to 255
	C	0 to 255	0 to 255	0 to 255	0 to 255
	L	0.0 to 59.7	0.0 to 59.7	0.0 to 59.7	0.0 to 59.7
Byte access	IB	0 to 15	0 to 15	0 to 15	0 to 15
	QB	0 to 15	0 to 15	0 to 15	0 to 15
	VB	0 to 2047	0 to 2047	0 to 5119	0 to 10239
	MB	0 to 31	0 to 31	0 to 31	0 to 31
	SMB	0 to 179	0 to 299	0 to 549	0 to 549
	SB	0 to 31	0 to 31	0 to 31	0 to 31
	L	0 to 63	0 to 63	0 to 63	0 to 255
	AC	0 to 3	0 to 3	0 to 3	0 to 255
Word access	IW	0 to 14	0 to 14	0 to 14	0 to 14
	QW	0 to 14	0 to 14	0 to 14	0 to 14
	VW	0 to 2046	0 to 2046	0 to 5118	0 to 10238
	MW	0 to 30	0 to 30	0 to 30	0 to 30
	SMW	0 to 178	0 to 298	0 to 548	0 to 548
	SW	0 to 30	0 to 30	0 to 30	0 to 30
	T	0 to 255	0 to 255	0 to 255	0 to 255
	C	0 to 255	0 to 255	0 to 255	0 to 255
	LW	0 to 58	0 to 58	0 to 58	0 to 58
	AC	0 to 3	0 to 3	0 to 3	0 to 3
	AIW	*None*	0 to 30	0 to 62	0 to 62
	AQW	*None*	0 to 30	0 to 62	0 to 62
Double word access	ID	0 to 12	0 to 12	0 to 12	0 to 12
	QD	0 to 12	0 to 12	0 to 12	0 to 12
	VD	0 to 2044	0 to 2044	0 to 5116	0 to 10236
	MD	0 to 28	0 to 28	0 to 28	0 to 28
	SMD	0 to 176	0 to 296	0 to 546	0 to 546
	SD	0 to 28	0 to 28	0 to 28	0 to 28
	LD	0 to 56	0 to 56	0 to 56	0 to 56
	AC	0 to 3	0 to 3	0 to 3	0 to 3
	HC	0, 3, 4, 5	0, 3, 4, 5	0 to 5	0 to 5

Pulse Train Operation (PTO)

PTO provides a square wave (50% duty cycle) output for a specified number of pulses and a specified cycle time. (See Figure 6-29.) PTO can produce either a single train of pulses or multiple trains of pulses (using a pulse profile). You specify the number of pulses and the cycle time (in either microsecond or millisecond increments):

- Number of pulses: 1 to 4,294,967,295
- Cycle time: 50 µs to 65,535 µs or 2 ms to 65,535 ms.

Specifying an odd number of microseconds or milliseconds for the cycle time (such as 75 ms), causes some distortion in the duty cycle.

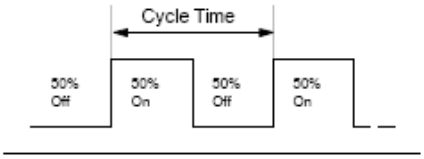


Figure 6-29 Pulse Train Output (PTO)

See Table 6-32 for pulse count and cycle time limitations.

Table 6-32 Pulse Count and Cycle Time in the PTO function

Pulse Count/Cycle TIme	Reaction
Cycle time < 2 time units	Cycle time defaults to 2 time units.
Pulse count = 0	Pulse count defaults to 1 pulse.

The PTO function allows the "chaining" or "pipelining" of pulse trains. When the active pulse train is complete, the output of a new pulse train begins immediately. This allows continuity between subsequent output pulse trains.

Single-Segment Pipelining of PTO Pulses

In single-segment pipelining, you are responsible for updating the SM locations for the next pulse train. After the initial PTO segment has been started, you must modify immediately the SM locations as required for the second waveform and execute the PLS instruction again. The attributes of the second pulse train are held in a pipeline until the first pulse train is completed. Only one entry at a time can be stored in the pipeline. When the first pulse train completes, the output of the second waveform begins, and the pipeline is made available for a new pulse train specification. You can then repeat this process to set up the characteristics of the next pulse train.

Smooth transitions between pulse trains occur unless there is a change in the time base or the active pulse train completes before a new pulse train setup is captured by the execution of the PLS instruction.

Multiple-Segment Pipelining of PTO Pulses

In multiple-segment pipelining, the S7-200 automatically reads the characteristics of each pulse train segment from a profile table located in V memory. The SM locations used in this mode are the control byte, the status byte, and the starting V memory offset of the profile table (SMW168 or SMW178). The time base can be either microseconds or milliseconds, but the selection applies to all cycle time values in the profile table, and cannot be changed while the profile is running. Execution on the PLS instruction starts multiple segment operation.

Each segment entry is 8 bytes in length, and is composed of a 16-bit cycle time value, a 16-bit cycle time delta value, and a 32-bit pulse count value. Table 6-33 shows the format of the profile table. You can increase or decrease the cycle time automatically by programming a specified amount for each pulse. A positive value in the cycle time delta field increases cycle time, a negative value in the cycle time delta field decreases cycle time, and 0 results in an unchanging cycle time.

While the PTO profile is operating, the number of the currently active segment is available in SMB166 (or SMB176).

Table 6-33 Profile Table Format for Multiple-Segment PTO Operation

Byte Offset	Segment	Description of Table Entries
0		Number of segments: 1 to 255[1]
1	#1	Initial cycle time (2 to 65,535 units of the time base)
3		Cycle time delta per pulse (signed value) (-32,768 to 32,767 units of the time base)
5		Pulse count (1 to 4,294,967,295)
9	#2	Initial cycle time (2 to 65,535 units of the time base)
11		Cycle time delta per pulse (signed value) (-32,768 to 32,767 units of the time base)
13		Pulse count (1 to 4,294,967,295)
(Continues)	#3	(Continues)

1 Entering a value of 0 for the number of segments generates a non-fatal error. No PTO output is generated.

6

Pulse Width Modulation (PWM)

PWM provides a fixed cycle time output with a variable duty cycle. (See Figure 6-30.) You can specify the cycle time and the pulse width in either microsecond or millisecond increments:

- Cycle time: 50 μs to 65,535 μs or 2 ms to 65,535 ms
- Pulse width time: 0 μs to 65,535 μs or 0 ms to 65,535 ms

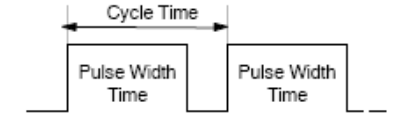


Figure 6-30 Pulse Width Modulation (PWM)

As shown in Table 6-34, setting the pulse width equal to the cycle time (which makes the duty cycle 100 percent) turns the output on continuously. Setting the pulse width to 0 (which makes the duty cycle 0 percent) turns the output off.

Table 6-34 Pulse Width Time and Cycle Time and Reactions in the PWM Function

Pulse Width Time/ Cycle Time	Reaction
Pulse width time >= Cycle time value	The duty cycle is 100%: the output is turned on continuously.
Pulse width time = 0	The duty cycle is 0%: the output is turned off.
Cycle time < 2 time units	The cycle time defaults to two time units.

There are two different ways to change the characteristics of a PWM waveform:

- Synchronous Update: If no time base changes are required, you can use a synchronous update. With a synchronous update, the change in the waveform characteristics occurs on a cycle boundary, providing a smooth transition.
- Asynchronous Update: Typically with PWM operation, the pulse width is varied while the cycle time remains constant so time base changes are not required. However, if a change in the time base of the PTO/PWM generator is required, an asynchronous update is used. An asynchronous update causes the PTO/PWM generator to be disabled momentarily, asynchronous to the PWM waveform. This can cause undesirable jitter in the controlled device. For that reason, synchronous PWM updates are recommended. Choose a time base that you expect to work for all of your anticipated cycle time values.

Tip

The PWM Update Method bit (SM67.4 or SM77.4) in the control byte specifies the update type used when the PLS instruction is executed to invoke changes.

If the time base is changed, an asynchronous update occurs regardless of the state of the PWM Update Method bit.

Using SM Locations to Configure and Control the PTO/PWM Operation

The PLS instruction reads the data stored in the specified SM memory locations and programs the PTO/PWM generator accordingly. SMB67 controls PTO 0 or PWM 0, and SMB77 controls PTO 1 or PWM 1. Table 6-35 describes the registers used to control the PTO/PWM operation. You can use Table 6-36 as a quick reference to determine the value to place in the PTO/PWM control register to invoke the desired operation.

You can change the characteristics of a PTO or PWM waveform by modifying the locations in the SM area (including the control byte) and then executing the PLS instruction. You can disable the generation of a PTO or PWM waveform at any time by writing 0 to the PTO/PWM enable bit of the control byte (SM67.7 or SM77.7) and then executing the PLS instruction.

The PTO Idle bit in the status byte (SM66.7 or SM76.7) is provided to indicate the completion of the programmed pulse train. In addition, an interrupt routine can be invoked upon the completion of a pulse train. (Refer to the descriptions of the Interrupt instructions and the Communications instructions.) If you are using the multiple segment operation, the interrupt routine is invoked upon completion of the profile table.

The following conditions set SM66.4 (or SM76.4) and SM66.5 (or SM76.5):

- Specifying a cycle time delta value that results in an illegal cycle time after a number of pulses generates a mathematical overflow condition that terminates the PTO function and sets the Delta Calculation Error bit (SM66.4 or SM76.4) to 1. The output reverts to image register control.
- Manually aborting (disabling) a PTO profile in progress sets the User Abort bit (SM66.5 or SM76.5) to 1.
- Attempting to load the pipeline while it is full sets the PTO overflow bit (SM66.6 or SM76.6) to 1. You must clear this bit manually after an overflow is detected if you want to detect subsequent overflows. The transition to RUN mode initializes this bit to 0.

Tip

When you load a new pulse count (SMD72 or SMD82), pulse width (SMW70 or SMW80), or cycle time (SMW68 or SMW78), also set the appropriate update bits in the control register before you execute the PLS instruction. For a multiple segment pulse train operation, you must also load the starting offset (SMW168 or SMW178) of the profile table and the profile table values before you execute the PLS instruction.

Table 6-35 SM Locations of the PTO / PWM Control Registers

Q0.0	Q0.1	Status Bits		
SM66.4	SM76.4	PTO profile aborted (delta calculation error):	0 = no error	1 = aborted
SM66.5	SM76.5	PTO profile aborted due to user command:	0 = no abort	1 = aborted
SM66.6	SM76.6	PTO pipeline overflow/underflow: overflow/underflow	0 = no overflow	1 =
SM66.7	SM76.7	PTO idle:	0 = in progress	1 = PTO idle
Q0.0	**Q0.1**	**Control Bits**		
SM67.0	SM77.0	PTO/PWM update the cycle time:	0 = no update	1 = update cycle time
SM67.1	SM77.1	PWM update the pulse width time: width	0 = no update	1 = update pulse
SM67.2	SM77.2	PTO update the pulse count value: count	0 = no update	1 = update pulse
SM67.3	SM77.3	PTO/PWM time base:	0 = 1 µs/tick	1 = 1 ms/tick
SM67.4	SM77.4	PWM update method:	0 = asynchronous	1 = synchronous
SM67.5	SM77.5	PTO single/multiple segment operation:	0 = single	1 = multiple
SM67.6	SM77.6	PTO/PWM mode select:	0 = PTO	1 = PWM
SM67.7	SM77.7	PTO/PWM enable:	0 = disable	1 = enable
Q0.0	**Q0.1**	**Other PTO/PWM Registers**		
SMW68	SMW78	PTO/PWM cycle time value	range: 2 to 65,535	
SMW70	SMW80	PWM pulse width value	range: 0 to 65,535	
SMD72	SMD82	PTO pulse count value	range: 1 to 4,294,967,295	
SMB166	SMB176	Number of the segment in progress	Multiple-segment PTO operation only	
SMW168	SMW178	Starting location of the profile table (byte offset from V0)	Multiple-segment PTO operation only	

Table 6-36 PTO/PWM Control Byte Reference

Control Register (Hex Value)		Result of Executing the PLS Instruction						
	Enable	Select Mode	PTO Segment Operation	PWM Update Method	Time Base	Pulse Count	Pulse Width	Cycle Time
16#81	Yes	PTO	Single		1 µs/cycle			Load
16#84	Yes	PTO	Single		1 µs/cycle	Load		
16#85	Yes	PTO	Single		1 µs/cycle	Load		Load
16#89	Yes	PTO	Single		1 ms/cycle			Load
16#8C	Yes	PTO	Single		1 ms/cycle	Load		
16#8D	Yes	PTO	Single		1 ms/cycle	Load		Load
16#A0	Yes	PTO	Multiple		1 µs/cycle			
16#A8	Yes	PTO	Multiple		1 ms/cycle			
16#D1	Yes	PWM		Synchronous	1 µs/cycle			Load
16#D2	Yes	PWM		Synchronous	1 µs/cycle		Load	
16#D3	Yes	PWM		Synchronous	1 µs/cycle		Load	Load
16#D9	Yes	PWM		Synchronous	1 ms/cycle			Load
16#DA	Yes	PWM		Synchronous	1 ms/cycle		Load	
16#DB	Yes	PWM		Synchronous	1 ms/cycle		Load	Load

Calculating Profile Table Values

The multiple-segment pipelining capability of the PTO/PWM generators can be useful in many applications, particularly in stepper motor control.

For example, you can use PTO with a pulse profile to control a stepper motor through a simple ramp up, run, and ramp down sequence or more complicated sequences by defining a pulse profile that consists of up to 255 segments, with each segment corresponding to a ramp up, run, or ramp down operation.

Figure 6-31 illustrates sample profile table values required to generate an output waveform that accelerates a stepper motor (segment 1), operates the motor at a constant speed (segment 2), and then decelerates the motor (segment 3).

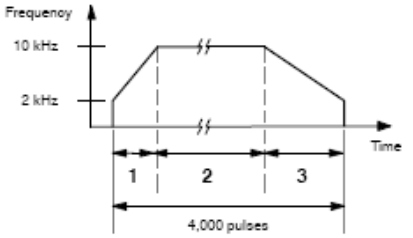


Figure 6-31 Frequency/Time Diagram

For this example: The starting and final pulse frequency is 2 kHz, the maximum pulse frequency is 10 kHz, and 4000 pulses are required to achieve the desired number of motor revolutions. Since the values for the profile table are expressed in terms of period (cycle time) instead of frequency, you must convert the given frequency values into cycle time values. Therefore, the starting (initial) and final (ending) cycle time is 500 µs, and the cycle time corresponding to the maximum frequency is 100 µs. During the acceleration portion of the output profile, the maximum pulse frequency should be reached in approximately 200 pulses. The deceleration portion of the profile should be completed in approximately 400 pulses.

You can use the following formula to determine the delta cycle time value for a given segment that the PTO/PWM generator uses to adjust the cycle time of each pulse:

Delta cycle time for a segment = | End_CT_{seg} - $Init\_CT_{seg}$ | / $Quantity_{seg}$

where: End_CT_{seg} = Ending cycle time for this segment
$Init\_CT_{seg}$ = Initial cycle time for this segment
$Quantity_{seg}$ = Quantity of pulses in this segment

Using this formula to calculate the delta cycle time values for the sample application:

Segment 1 (acceleration):
Delta cycle time = -2

Segment 2 (constant speed):
Delta cycle time = 0

Segment 3 (deceleration):
Delta cycle time = 1

Table 6-37 lists the values for generating the example waveform (assumes that the profile table is located in V memory, starting at V500). You can include instructions in your program to load these values into V memory, or you can define the values of the profile in the data block.

Table 6-37 Profile Table Values

Address	Value	Description	
VB500	3	Total number of segments	
VW501	500	Initial cycle time	Segment 1
VW503	-2	Initial delta cycle time	
VD505	200	Number of pulses	
VW509	100	Initial cycle time	Segment 2
VW511	0	Delta cycle time	
VD513	3400	Number of pulses	
VW517	100	Initial cycle time	Segment 3
VW519	1	Delta cycle time	
VD521	400	Number of pulses	

In order to determine if the transitions between waveform segments are acceptable, you need to determine the cycle time of the last pulse in a segment. Unless the delta cycle time is 0, you must calculate the cycle time of the last pulse of a segment, because this value is not specified in the profile. Use the following formula to calculate the cycle time of the last pulse:

Cycle time of the last pulse for a segment = $Init\_CT_{seg}$ + ($Delta_{seg}$ * ($Quantity_{seg}$ – 1))

where: $Init\_CT_{seg}$ = Initial cycle time for this segment

$Delta_{seg}$ = Delta cycle time for this segment

$Quantity_{seg}$ = Quantity of pulses in this segment

While the simplified example above is useful as an introduction, real applications can require more complicated waveform profiles. Remember that the delta cycle time can be specified only as an integer number of microseconds or milliseconds, and the cycle time modification is performed on each pulse.

The effect of these two items is that calculation of the delta cycle time value for a given segment could require an iterative approach. Some flexibility in the value of the ending cycle time or the number of pulses for a given segment might be required.

The duration of a given profile segment can be useful in the process of determining correct profile table values. Use the following formula to calculate the length of time for completing a given profile segment:

Duration of segment = $Quantity_{seg}$ * ($Init\_CT$ + (($Delta_{seg}$/2) * ($Quantity_{seg}$ – 1)))

where: $Quantity_{seg}$ = Quantity of pulses in this segment

$Init\_CT_{seg}$ = Initial cycle time for this segment

$Delta_{seg}$ = Delta cycle time for this segment

Sample Operation of a PWM Output

Tip

The following description of the PWM initialization and operation sequences recommends using the First Scan bit (SM0.1) to initialize the pulse output. Using the First Scan bit to call an initialization subroutine reduces the scan time because subsequent scans do not call this subroutine. (The First Scan bit is set only on the first scan following a transition to RUN mode.) However, your application could have other constraints that require you to initialize (or re-initialize) the pulse output. In that case, you can use another condition to call the initialization routine.

Initializing the PWM Output

Typically, you use a subroutine to initialize the PWM for the pulse output. You call the initialization subroutine from the main program. Use the first scan memory bit (SM0.1) to initialize the output used by the PWM to 0, and call a subroutine to perform the initialization operations. When you use the subroutine call, subsequent scans do not make the call to the subroutine, which reduces the scan time execution and provides a more structured program.

After creating the call to the initialization subroutine from the main program, use the following steps to create the control logic for configuring pulse output Q0.0 within the initialization subroutine:

1. Configure the control byte by loading one of the following values to SMB67: 16#D3 (to select microsecond increments) or 16#DB (to select millisecond increments).

   Both of these values enable the PTO/PWM function, select PWM operation, set the update pulse width and cycle time values, and select the time base (microseconds or milliseconds).
2. Load a word-sized value for the cycle time in SMW68.
3. Load a word-sized value for the pulse width in SMW70.
4. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator).
5. To preload a new control byte value for subsequent pulse width changes (optional), load one of the following values in SMB67: 16#D2 (microseconds) or 16#DA (milliseconds).
6. Exit the subroutine.

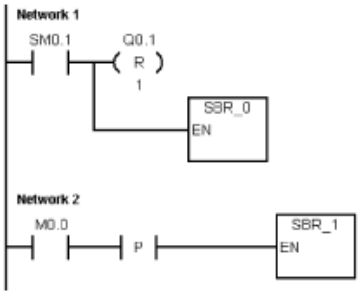
Changing the Pulse Width for the PWM Output

If you preloaded SMB67 with 16#D2 or 16#DA (see step 5. above), you can use a subroutine that changes the pulse width for the pulse output (Q0.0). After creating the call to this subroutine, use the following steps to create the control logic for changing the pulse width:

1. Load a word-sized value for the new pulse width in SMW70.
2. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator).
3. Exit the subroutine.

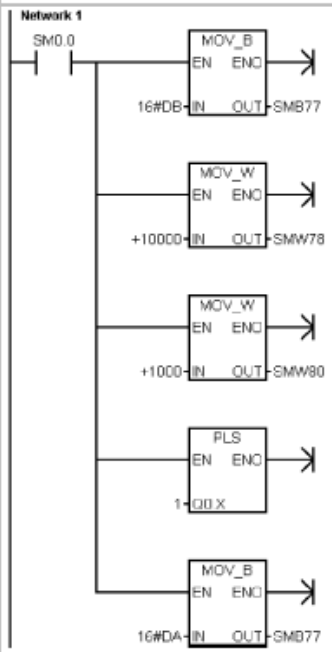
Example: Pulse Width Modulation (PWM)

MAIN



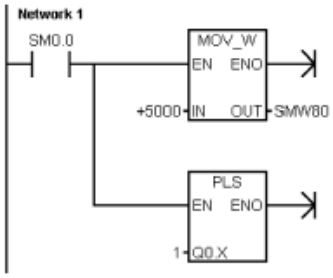

```
Network 1     //On the first scan,
              //set the image register bit low and call SBR_0.
LD      SM0.1
R       Q0.1, 1
CALL    SBR_0
Network 2     //Set M0.0 elsewhere in the program
              //to change pulse width to 50% duty cycle.
LD      M0.0
EU
CALL    SBR_1
```

SBR0



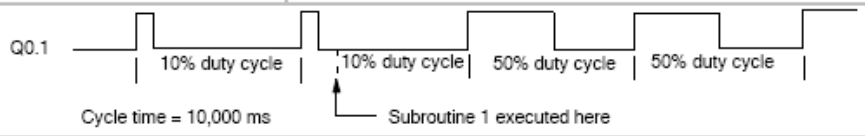

```
Network 1     //Start of subroutine 0:
              //1. Set up the control byte.
              //   - Select PWM operation.
              //   - Select ms increments and
              //     synchronous updates.
              //   - Enable the loading of the pulse width
              //     and cycle time values.
              //   - Enable the PWM function.
              //2. Set the cycle time to 10,000 ms.
              //3. Set pulse width to 1,000 ms.
              //4. Invoke PWM operation: PLS1=>Q0.1.
              //5. Preload the control byte for subsequent
              //   pulse width changes
LD      SM0.0
MOVB    16#DB, SMB77
MOVW    +10000, SMW78
MOVW    +1000, SMW80
PLS     1
MOVB    16#DA, SMB77
```

SBR1




```
Network 1     //Start of subroutine 1:
              //Set the pulse width to 5000 ms.
              //Assert pulse width change.
LD      SM0.0
MOVW    +5000, SMW80
PLS     1
```

Timing Diagram



Sample Operation of a PTO Output

Tip

The following description of the PTO initialization and operation sequences recommends using the First Scan memory bit (SM0.1) to initialize the pulse output. Using the First Scan bit to call an initialization subroutine reduces the scan time because subsequent scans do not call this subroutine. (The First Scan bit is set only on the first scan following a transition to RUN mode.) However, your application could have other constraints that require you to initialize (or re-initialize) the pulse output. In that case, you can use another condition to call the initialization routine.

Initializing the PTO Output for a Single-Segment Operation

Typically, you use a subroutine to configure and initialize the PTO for the pulse output. You call the initialization subroutine from the main program. Use the first scan memory bit (SM0.1) to initialize the output used by the PTO to 0, and call a subroutine to perform the initialization operations. When you use the subroutine call, subsequent scans do not make the call to the subroutine, which reduces the scan time execution and provides a more structured program.

After creating the call to the initialization subroutine from the main program, use the following steps to create the control logic for configuring pulse output Q0.0 within the initialization subroutine:

1. Configure the control byte by loading one of the following values in SMB67: 16#85 (to select microsecond increments) or 16#8D (to select millisecond increments).

   Both of these values enable the PTO/PWM function, select PTO operation, set the update pulse count and cycle time values, and select the time base (microseconds or milliseconds).

2. In SMW68, load a word-sized value for the cycle time.
3. In SMD72, load a double-word-sized value for the pulse count.
4. (Optional) To perform a related function as soon as the pulse train output is complete, you can program an interrupt by attaching the pulse train complete event (interrupt event 19) to an interrupt subroutine. Use the ATCH instruction and execute the global interrupt enable instruction ENI.
5. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator).
6. Exit the subroutine.

Changing the PTO Cycle Time (Single-Segment Operation)

For a single-segment PTO operation, you can use an interrupt routine or a subroutine to change the cycle time. To change the PTO cycle time in an interrupt routine or subroutine when using a single-segment PTO operation, follow these steps:

1. Set the control byte (to enable the PTO/PWM function, to select PTO operation, to select the time base, and to set the update cycle time value) by loading one of the following values in SMB67: 16#81 (for microseconds) or 16#89 (for milliseconds).
2. In SMW68, load a word-sized value for the new cycle time.
3. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator). The S7-200 completes any PTO that is in process before starting to generate the PTO waveform with the updated cycle time.
4. Exit the interrupt routine or the subroutine.

Changing the PTO Pulse Count (Single-Segment Operation)

For a single-segment PTO operation, you can use an interrupt routine or a subroutine to change the pulse count. To change the PTO pulse count in an interrupt routine or a subroutine when using a single-segment PTO operation, follow these steps:

1. Set the control byte (to enable the PTO/PWM function, to select PTO operation, to select the time base, and to set the update pulse count value) by loading either of the following values in SMB67: 16#84 (for microseconds) or 16#8C (for milliseconds).
2. In SMD72, load a double-word-sized value for the new pulse count.
3. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator). The S7-200 completes any PTO that is in process before starting to generate the waveform with the updated pulse count.
4. Exit the interrupt routine or the subroutine.

Changing the PTO Cycle Time and the Pulse Count (Single-Segment Operation)

For a single-segment PTO operation, you can use an interrupt routine or a subroutine to change the cycle time and pulse count. To change the PTO cycle time and pulse count in an interrupt routine or a subroutine when using a single-segment PTO operation, follow these steps:

1. Set the control byte (to enable the PTO/PWM function, to select PTO operation, to select the time base, and to set the update cycle time and pulse count values) by loading either of the following values in SMB67: 16#85 (for microseconds) or 16#8D (for milliseconds).
2. In SMW68, load a word-sized value for the new cycle time.
3. In SMC72, load a double-word-sized value for the new pulse count.
4. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator). The S7-200 completes any PTO that is in process before starting to generate the waveform with the updated pulse count and cycle time.
5. Exit the interrupt routine or the subroutine.

Initializing the PTO Output for a Multiple-Segment Operation

Typically, you use a subroutine to configure and initialize the PTO for the pulse output for multiple-segment operation. You call the initialization subroutine from the main program. Use the first scan memory bit (SM0.1) to initialize the output used by the PTO to 0, and call a subroutine to perform the initialization operations. When you use the First Scan bit to call the initialization subroutine, the subsequent scans do not make the call to the subroutine, which reduces scan time execution.

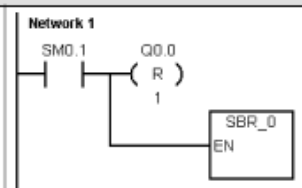
After creating the call to the initialization subroutine from the main program, use the following steps to create the control logic for configuring pulse output Q0.0 within the initialization subroutine:

1. Configure the control byte by loading one of the following values in SMB67: 16#A0 (to select microsecond increments) or 16#A8 (to select millisecond increments).

   Both of these values enable the PTO/PWM function, select PTO operation, select multiple-segment operation, and select the time base (microseconds or milliseconds).
2. In SMW168, load a word-sized value for the starting V memory offset of the profile table.
3. Use V memory to set up the segment values in the profile table. Ensure that the Number of Segment field (the first byte of the table) is correct.
4. (Optional) To perform a related function as soon as the PTO profile is complete, you can program an interrupt by attaching the pulse train complete event (interrupt event 19) to an interrupt subroutine. Use the ATCH instruction and execute the global interrupt enable instruction ENI.
5. Execute the PLS instruction (so that the S7-200 programs the PTO/PWM generator).
6. Exit the subroutine.

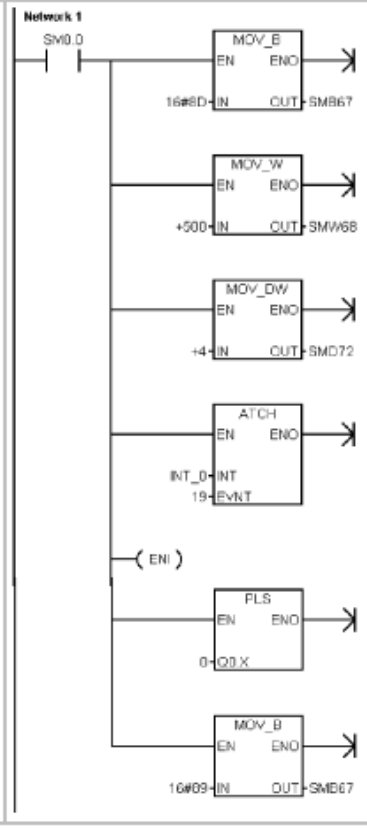
Example: Single-Segment Pulse Train Operation (PTO)

MAIN



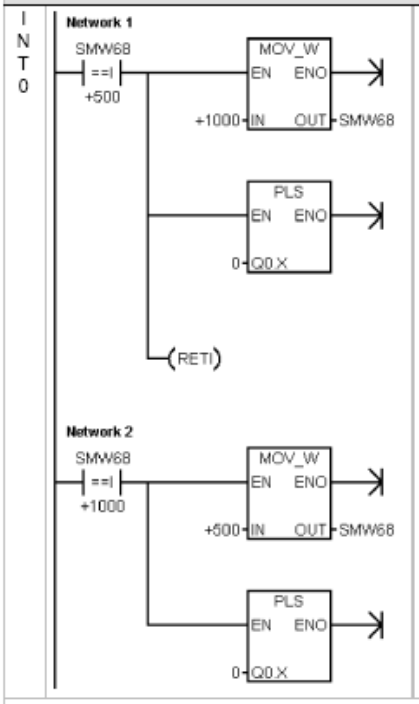

```
Network 1     //On the first scan,
              //set the image register bit low and call subroutine 0.
LD      SM0.1
R       Q0.0, 1
CALL    SBR_0
```

SBR 0



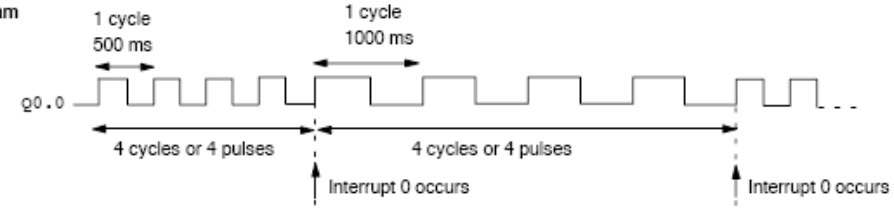

```
Network 1     //Start of subroutine 0: Configure PTO
              //1. Set up the control byte:
              //    - Select PTO operation.
              //    - Select single segment operation.
              //    - Select ms increments.
              //    - Enable the loading of the pulse count
              //      and cycle time value.
              //    - Enable the PTO function.
              //2. Set cycle time to 500ms.
              //3. Set pulse count to 4 pulses.
              //4. Define interrupt routine 0 to be the interrupt
              //   for processing PTO complete interrupts.
              //5. Global interrupt enable.
              //6. Invoke PTO operation, PLS0 => Q0.0.
              //7. Preload the control byte for subsequent cycle time
              //changes.
LD      SM0.0
MOVB    16#8D, SMB67
MOVW    +500, SMW68
MOVD    +4, SMD72
ATCH    INT_0, 19
ENI
PLS     0
MOVB    16#89, SMB67
```

Example: Single-Segment Pulse Train Operation (PTO), continued



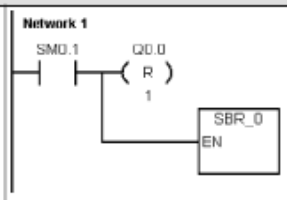

```
Network 1       //If current cycle time is 500 ms:
                //Set the cycle time to 1000 ms and generate 4 pulses.
LDW=    SMW68, +500
MOVW    +1000, SMW68
PLS     0
CRETI

Network 2       //If current cycle time is 1000 ms:
                //Set the cycle time to 500 ms and generate 4 pulses.
LDW=    SMW68, +1000
MOVW    +500, SMW68
PLS     0
```

Timing Diagram



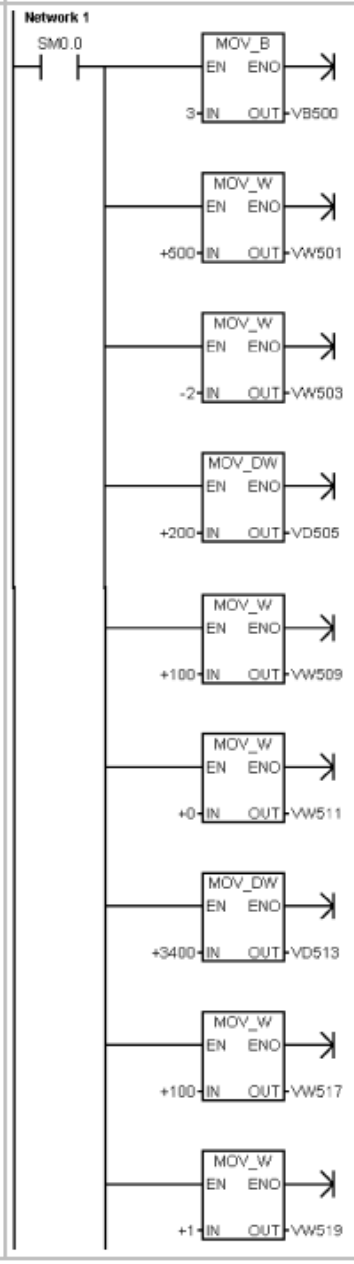
Example: Multiple-Segment Pulse Train Operation (PTO)

MAIN




```
Network 1       //On the first scan,
                //set the image register bit low and call subroutine 0
LD      SM0.1
R       Q0.0, 1
CALL    SBR_0
```

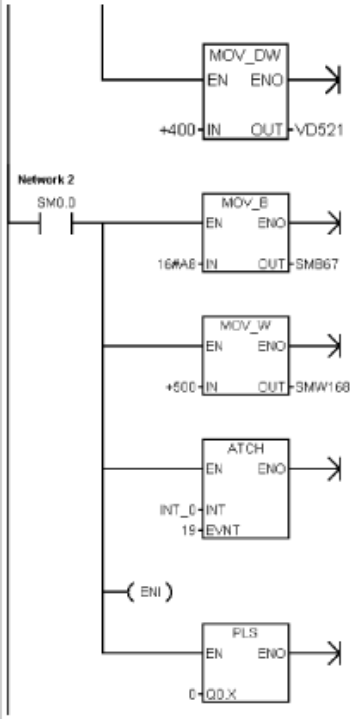
SBR0




```
Network 1       //Preload the PTO profile table:
                //Set number of profile table segments to 3.
                //Configure each of the 3 segments.
                //
                //1. Configure segment 1:
                //    - Set the initial cycle time = 500 ms.
                //    - Set the delta cycle time to -2 ms.
                //    - Set the number of pulses to 200.
                //2. Configure segment 2:
                //    - Set the initial cycle time to 100 ms.
                //    - Set the delta cycle time to 0 ms.
                //    - Set the number of pulses to 3400.
                //3. Configure segment 3:
                //    - Set the initial cycle time to 100 ms.
                //    - Set the delta cycle time to 1 ms.
                //    - Set the number of pulses to 400.

LD      SM0.0
MOVB    3, VB500
MOVW    +500, VW501   //Segment 1
MOVW    -2, VW503
MOVD    +200, VD505
MOVW    +100, VW509   //Segment 2
MOVW    +0, VW511
MOVD    +3400, VD513
MOVW    +100, VW517   //Segment 3
MOVW    +1, VW519
MOVD    +400, VD521
```

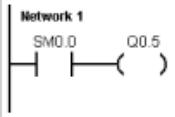
Example: Multiple-Segment Pulse Train Operation (PTO) , continued

SBR0 continued




```
Network 2     //1. Set up the control byte:
              //    - Select PTO operation
              //    - Select multiple segment operation
              //    - Select ms increments
              //    - Enable the PTO function
              //2. Set the start address of the profile table to V500.
              //3. Define interrupt routine 0 to be the interrupt
              //   for processing PTO complete interrupts.
              //4. Global interrupt enable
              //5. Invoke PTO operation, PLS0 => Q0.0.
LD     SM0.0
MOVB   16#A8, SMB67
MOVW   +500, SMW168
ATCH   INT_0, 19
ENI
PLS    0
```

INT 0




```
Network 1     //When the PTO output profile is complete,
              //Turn on output Q0.5
LD     SM0.0
=      Q0.5
```

LAMPIRAN 2

Datasheet Manual Micro/Win

S7-200 Expansion Modules

To better solve your application requirements, the S7-200 family includes a wide variety of expansion modules. You can use these expansion modules to add additional functionality to the S7-200 CPU. Table 1-2 provides a list of the expansion modules that are currently available. For detailed information about a specific module, see Appendix A.

Table 1-2 S7-200 Expansion Modules

Expansion Modules		Types		
Discrete modules	Input	8 x DC In	8 x AC In	16 x DC In
	Output	4 x DC 8 x DC Out	4 x Relays 8 x AC Out	 8 x Relay
	Combination	4 x DC In / 4 x DC Out 4 x DC In / 4 x Relay	8 x DC In / 8 x DC Out 8 x DC In / 8 x Relay	16 x DC In / 16 x DC Out 16 x DC In / 16 x Relay
Analog modules	Input	4 x Analog In	4 x Thermocouple In	2 x RTD In
	Output	2 x Analog Out		
	Combination	4 x Analog In / 1 Analog Out		
Intelligent modules		Position Ethernet	Modem Internet	PROFIBUS-DP
Other modules		AS-Interface		

STEP 7-Micro/WIN Programming Package

The STEP 7-Micro/WIN programming package provides a user-friendly environment to develop, edit, and monitor the logic needed to control your application. STEP 7-Micro/WIN provides three program editors for convenience and efficiency in developing the control program for your application. To help you find the information you need, STEP 7-Micro/WIN provides an extensive online help system and a documentation CD that contains an electronic version of this manual, application tips, and other useful information.

Computer Requirements

STEP 7-Micro/WIN runs on either a personal computer or a Siemens programming device, such as a PG 760. Your computer or programming device should meet the following minimum requirements:

- Operating system:
  Windows 95, Windows 98,
  Windows 2000, Windows Me (Millennium Edition), Windows NT 4.0 (or later version),
  Windows XP Professional
- At least 100M bytes of free hard disk space
- Mouse (recommended)

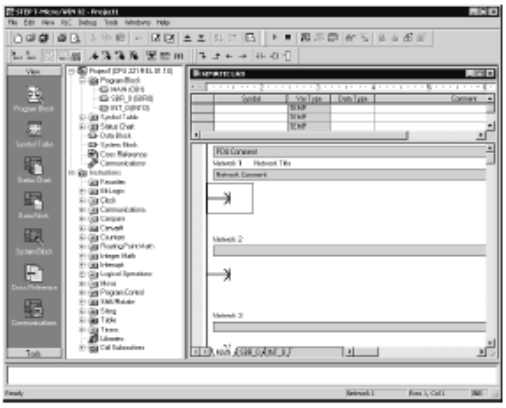


Figure 1-2 STEP 7-Micro/WIN

1

Installing STEP 7-Micro/WIN

Insert the STEP 7-Micro/WIN CD into the CD-ROM drive of your computer. The installation wizard starts automatically and prompts you through the installation process. Refer to the Readme file for more information about installing STEP 7-Micro/WIN.

Tip
To install STEP 7-Micro/WIN on a Windows NT, Windows 2000, or Windows XP Professional operating system, you must log in with Administrator privileges.

Communications Options

Siemens provides two programming options for connecting your computer to your S7-200: a direct connection with a PPI Multi-Master cable, or a Communications Processor (CP) card with an MPI cable.

The PPI Multi-Master programming cable is the most common and economical method of connecting your computer to the S7-200. This cable connects the communications port of the S7-200 to the serial communications of your computer. The PPI Multi-Master programming cable can also be used to connect other communications devices to the S7-200.

Display Panels

TD 200 Text Display Unit

The TD 200 is a 2-line, 20-character, text display device that can be connected to the S7-200. Using the TD 200 wizard, you can easily program your S7-200 to display text messages and other data pertaining to your application.

The TD 200 provides a low cost interface to your application by allowing you to view, monitor, and change the process variables pertaining to your application.

A separate manual describes the detailed functionality and specifications of the TD 200.

TD 200

The TD 200 Configuration Wizard in STEP 7-MicroWIN helps you configure TD 200 messages quickly and easily. To start the TD 200 Wizard, select the **Tools > TD 200 Wizard** menu command.

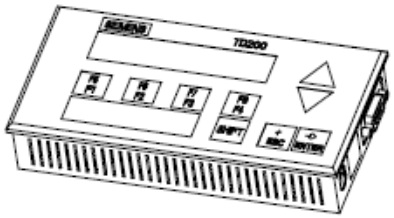


Figure 1-3 TD 200 Text Display Unit

TP070 Touch Panel Display

The TP070 is a touch panel display device that can be connected to the S7-200. This touch panel provides you with a means to customize your operator interface.

The TP070 can display custom graphics, slider bars, application variables, custom user buttons, and so forth, by means of a user-friendly touch panel.

The optional TP-Designer for TP070, Version 1.0 CD provides the TP Designer software, which is required for programming your TP070.

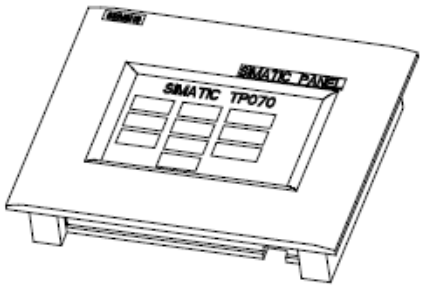


Figure 1-4 TP070 Touch Panel Unit

Getting Started

2

STEP 7-Micro/WIN makes it easy for you to program your S7-200. In just a few short steps using a simple example, you can learn how to connect, program, and run your S7-200.

All you need for this example is a PPI Multi-Master cable, an S7-200 CPU, and a programming device running the STEP 7-Micro/WIN programming software.

In This Chapter

Connecting the S7-200 CPU

2

Connecting your S7-200 is easy. For this example, you only need to connect power to your S7-200 CPU and then connect the communications cable between your programming device and the S7-200 CPU.

Connecting Power to the S7-200 CPU

The first step is to connect the S7-200 to a power source. Figure 2-1 shows the wiring connections for either a DC or an AC model of the S7-200 CPU.

Before you install or remove any electrical device, ensure that the power to that equipment has been turned off. Always follow appropriate safety precautions and ensure that power to the S7-200 is disabled before attempting to install or remove the S7-200.

Warning

Attempts to install or wire the S7-200 or related equipment with power applied could cause electric shock or faulty operation of equipment. Failure to disable all power to the S7-200 and related equipment during installation or removal procedures could result in death or serious injury to personnel, and/or damage to equipment.

Always follow appropriate safety precautions and ensure that power to the S7-200 is disabled before attempting to install or remove the S7-200 or related equipment.

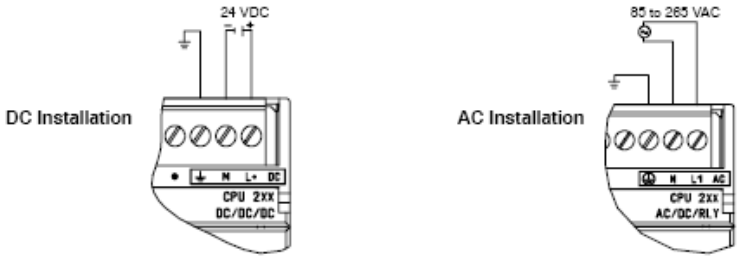


Figure 2-1 Connecting Power to the S7-200 CPU

Tip

Examples in this manual use the RS-232/PPI Multi-Master cable. The RS-232/PPI Multi-Master cable replaces the previous PC/PPI cable. A USB/PPI Multi-Master cable is also available. Refer to Appendix E for order numbers.

Connecting the RS-232/PPI Multi-Master Cable

Figure 2-2 shows an RS-232/PPI Multi-Master cable connecting the S7-200 to the programming device. To connect the cable:

1. Connect the RS-232 connector (marked "PC") of the RS-232/PPI Multi-Master cable to the communications port of the programming device. (For this example, connect to COM 1.)
2. Connect the RS-485 connector (marked "PPI") of the RS-232/PPI Multi-Master cable to Port 0 or Port 1 of the S7-200.
3. Ensure that the DIP switches of the RS-232/PPI Multi-Master cable are set as shown in Figure 2-2.

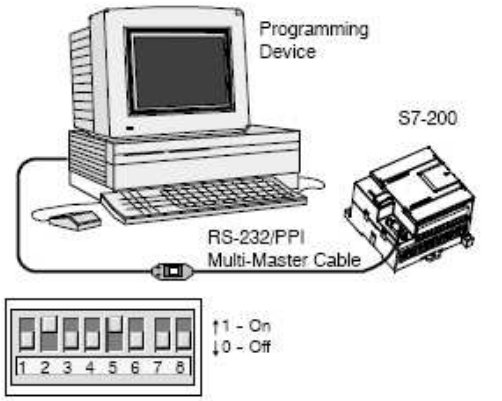


Figure 2-2 Connecting the RS-232/PPI Multi-Master Cable

Starting STEP 7-Micro/WIN

Click on the STEP 7-Micro/WIN icon to open a new project. Figure 2-3 shows a new project.

Notice the navigation bar. You can use the icons on the navigation bar to open elements of the STEP 7-Micro/WIN project.

Click on the Communications icon in the navigation bar to display the Communications dialog box. You use this dialog box to set up the communications for STEP 7-Micro/WIN.

Figure 2-3 New STEP 7-Micro/WIN Project

Verifying the Communications Parameters for STEP 7-Micro/WIN

The example project uses the default settings for STEP 7-Micro/WIN and the RS-232/PPI Multi-Master cable. To verify these settings:

1. Verify that the address of the PC/PPI cable in the Communications dialog box is set to 0.
2. Verify that the interface for the network parameter is set for PC/PPI cable(COM1).
3. Verify that the transmission rate is set to 9.6 kbps.

If you need to change your communications parameter settings, see Chapter 7.

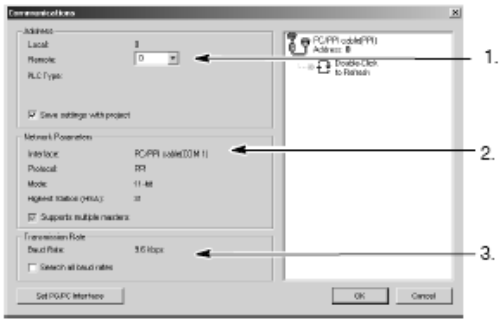


Figure 2-4 Verifying the Communications Parameters

Establishing Communications with the S7-200

Use the Communications dialog box to connect with your S7-200 CPU:

1. Double-click the refresh icon in the Communications dialog box.
   STEP 7-Micro/WIN searches for the S7-200 station and displays a CPU icon for the connected S7-200 station.
2. Select the S7-200 and click OK.

If STEP 7-Micro/WIN does not find your S7-200 CPU, check the settings for the communications parameters and repeat these steps.

After you have established communications with the S7-200, you are ready to create and download the example program.

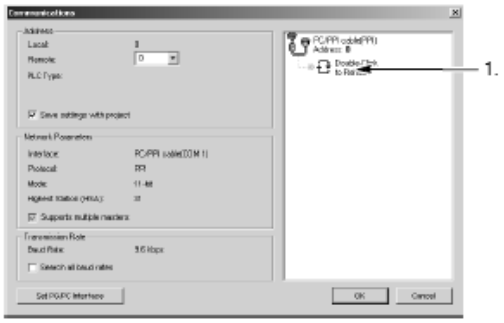


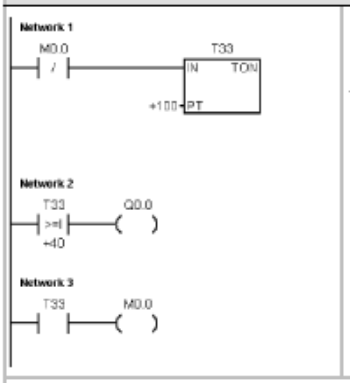
Figure 2-5 Establishing Communications to the S7-200

Creating a Sample Program

Entering this example of a control program will help you understand how easy it is to use STEP 7-Micro/WIN. This program uses six instructions in three networks to create a very simple, self-starting timer that resets itself.

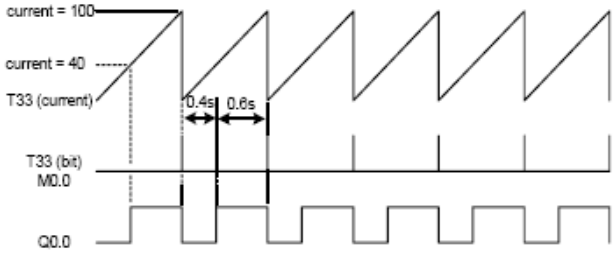
For this example, you use the Ladder (LAD) editor to enter the instructions for the program. The following example shows the complete program in both LAD and Statement List (STL). The network comments in the STL program explain the logic for each network. The timing diagram shows the operation of the program.

Example: Sample Program for getting started with STEP 7-Micro/WIN




```
Network 1     //10 ms timer T33 times out after (100 x 10 ms = 1 s)
              //M0.0 pulse is too fast to monitor with Status view.
LDN     M0.0
TON     T33, +100
Network 2     //Comparison becomes true at a rate that is visible with
              //Status view. Turn on Q0.0 after (40 x 10 ms = 0.4 s),
              //for a 40% OFF/60% ON waveform.
LDW>=   T33, +40
=       Q0.0
Network 3     //T33 (bit) pulse too fast to monitor with Status view.
              //Reset the timer through M0.0 after the
              ///(100 x 10 ms  = 1 s) period.
LD      T33
=       M0.0
```

Timing Diagram



Opening the Program Editor

Click on the Program Block icon to open the program editor. See Figure 2-6.

Notice the instruction tree and the program editor. You use the instruction tree to insert the LAD instructions into the networks of the program editor by dragging and dropping the instructions from the instruction tree to the networks.

The toolbar icons provide shortcuts to the menu commands.

After you enter and save the program, you can download the program to the S7-200.

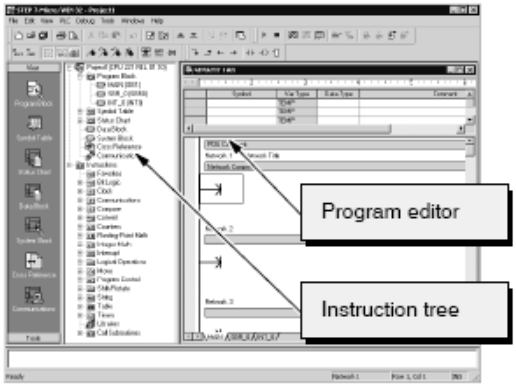


Figure 2-6 STEP 7-Micro/WIN Window

Entering Network 1: Starting the Timer

When M0.0 is off (0), this contact turns on and provides power flow to start the timer. To enter the contact for M0.0:

1. Either double-click the Bit Logic icon or click on the plus sign (+) to display the bit logic instructions.
2. Select the Normally Closed contact.
3. Hold down the left mouse button and drag the contact onto the first network.
4. Click on the "???" above the contact and enter the following address: M0.0
5. Press the Return key to enter the address for the contact.

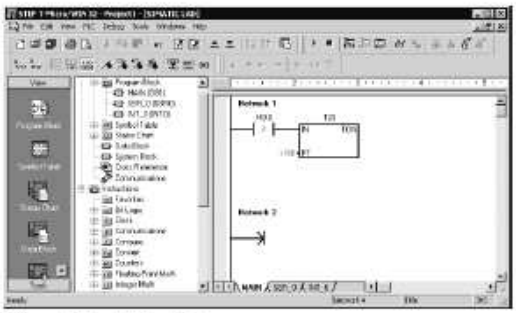


Figure 2-7 Network 1

To enter the timer instruction for T33:

1. Double-click the Timers icon to display the timer instructions.
2. Select the TON (On-Delay Timer).
3. Hold down the left mouse button and drag the timer onto the first network.
4. Click on the "???" above the timer box and enter the following timer number: T33
5. Press the Return key to enter the timer number and to move the focus to the preset time (PT) parameter.
6. Enter the following value for the preset time: 100
7. Press the Return key to enter the value.

Entering Network 2: Turning the Output On

When the timer value for T33 is greater than or equal to 40 (40 times 10 milliseconds, or 0.4 seconds), the contact provides power flow to turn on output Q0.0 of the S7-200. To enter the Compare instruction:

1. Double-click the Compare icon to display the compare instructions. Select the >=I instruction (Greater-Than-Or-Equal-To-Integer).
2. Hold down the left mouse button and drag the compare instruction onto the second network.
3. Click on the "???" above the contact and enter the address for the timer value: T33
4. Press the Return key to enter the timer number and to move the focus to the other value to be compared with the timer value.
5. Enter the following value to be compared with the timer value: 40
6. Press the Return key to enter the value.

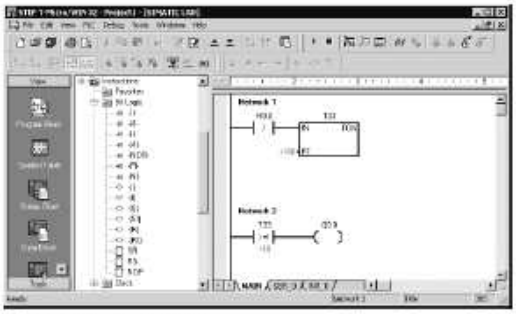


Figure 2-8 Network 2

To enter the instruction for turning on output Q0.0:

1. Double-click the Bit Logic icon to display the bit logic instructions and select the output coil.
2. Hold down the left mouse button and drag the coil onto the second network.
3. Click on the "???" above the coil and enter the following address: Q0.0
4. Press the Return key to enter the address for the coil.

Entering Network 3: Resetting the Timer

When the timer reaches the preset value (100) and turns the timer bit on, the contact for T33 turns on. Power flow from this contact turns on the M0.0 memory location. Because the timer is enabled by a Normally Closed contact for M0.0, changing the state of M0.0 from off (0) to on (1) resets the timer.

To enter the contact for the timer bit of T33:

1. Select the Normally Open contact from the bit logic instructions.
2. Hold down the left mouse button and drag the contact onto the third network.
3. Click on the "???" above the contact and enter the address of the timer bit: T33
4. Press the Return key to enter the address for the contact.

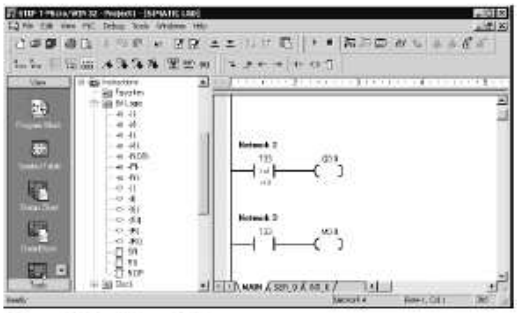


Figure 2-9 Network 3

To enter the coil for turning on M0.0:

1. Select the output coil from the bit logic instructions.
2. Hold down the left mouse button and drag the output coil onto the third network.
3. Double-click the "???" above the coil and enter the following address: M0.0
4. Press the Return key to enter the address for the coil.

Saving the Sample Project

After entering the three networks of instructions, you have finished entering the program. When you save the program, you create a project that includes the S7-200 CPU type and other parameters. To save the project:

1. Select the **File** > **Save As** menu command from the menu bar.
2. Enter a name for the project in the Save As dialog box.
3. Click OK to save the project.

After saving the project, you can download the program to the S7-200.

Figure 2-10 Saving the Example Program

Downloading the Sample Program

Tip

Each STEP 7-Micro/WIN project is associated with a CPU type (CPU 221, CPU 222, CPU 224, CPU 226, or CPU 226XM). If the project type does not match the CPU to which you are connected, STEP 7-Micro/WIN indicates a mismatch and prompts you to take an action. If this occurs, choose "Continue Download" for this example.

1. Click the Download icon on the toolbar or select the **File > Download** menu command to download the program. See Figure 2-11.
2. Click OK to download the elements of the program to the S7-200.

If your S7-200 is in RUN mode, a dialog box prompts you to place the S7-200 in STOP mode. Click Yes to place the S7-200 into STOP mode.

Figure 2-11 Downloading the Program

Placing the S7-200 in RUN Mode

For STEP 7-Micro/WIN to place the S7-200 CPU in RUN mode, the mode switch of the S7-200 must be set to TERM or RUN. When you place the S7-200 in RUN mode, the S7-200 executes the program:

1. Click the RUN icon on the toolbar or select the **PLC > RUN** menu command.
2. Click OK to change the operating mode of the S7-200.

When the S7-200 goes to RUN mode, the output LED for Q0.0 turns on and off as the S7-200 executes the program.

Figure 2-12 Placing the S7-200 in RUN Mode

Congratulations! You have just completed your first S7-200 program.

You can monitor the program by selecting the **Debug > Program Status** menu command. STEP 7-Micro/WIN displays the values for the instructions. To stop the program, place the S7-200 in STOP mode by clicking the STOP icon or by selecting the **PLC > STOP** menu command.

Using STEP 7-Micro/WIN to Create Your Programs

To open STEP 7-Micro/WIN, double-click on the STEP 7-Micro/WIN icon, or select the **Start > SIMATIC > STEP 7 MicroWIN 3.2** menu command. As shown in Figure 5-1, the STEP 7-Micro/WIN project window provides a convenient working space for creating your control program.

The toolbars provide buttons for shortcuts to frequently used menu commands. You can view or hide any of the toolbars.

The navigation bar presents groups of icons for accessing different programming features of STEP 7-Micro/WIN.

The instruction tree displays all of the project objects and the instructions for creating your control program. You can drag and drop individual instructions from the tree into your program, or you can double-click an instruction to insert it at the current location of the cursor in the program editor.

The program editor contains the program logic and a local variable table where you can assign symbolic names for temporary local variables. Subroutines and interrupt routines appear as tabs at the bottom of the program editor window. Click on the tabs to move between the subroutines, interrupts, and the main program.

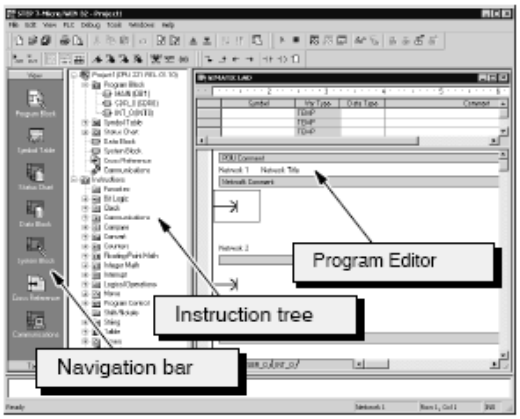


Figure 5-1 STEP 7-Micro/WIN

STEP 7-Micro/WIN provides three editors for creating your program: Ladder Logic (LAD), Statement List (STL), and Function Block Diagram (FBD). With some restrictions, programs written in any of these program editors can be viewed and edited with the other program editors.

Features of the STL Editor

The STL editor displays the program as a text-based language. The STL editor allows you to create control programs by entering the instruction mnemonics. The STL editor also allows you to create programs that you could not otherwise create with the LAD or FBD editors. This is because you are programming in the native language of the S7-200, rather than in a graphical editor where some restrictions must be applied in order to draw the diagrams correctly. As shown in Figure 5-2, this text-based concept is very similar to assembly language programming.

The S7-200 executes each instruction in the order dictated by the program, from top to bottom, and then restarts at the top.

STL uses a logic stack to resolve the control logic. You insert the STL instructions for handling the stack operations.

```
LD    I0.0    //Read one input
A     I0.1    //AND with another input
=     Q1.0    //Write value to output 1
```

Figure 5-2 Sample STL Program

Consider these main points when you select the STL editor:

- STL is most appropriate for experienced programmers.
- STL sometimes allows you to solve problems that you cannot solve very easily with the LAD or FBD editor.
- You can only use the STL editor with the SIMATIC instruction set.
- While you can always use the STL editor to view or edit a program that was created with the LAD or FBD editors, the reverse is not always true. You cannot always use the LAD or FBD editors to display a program that was written with the STL editor.