PENGENDALIAN KECEPATAN MOTOR DC PADA MINIATUR MOBIL LISTRIK TERHADAP LINTASAN LENGKUNG

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

DisusunOleh:

R. PRAJA KUSUMANUGRAHA
NIM. 0910633015 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK ELEKTRO
MALANG
2015

LEMBAR PERSETUJUAN

PENGENDALIAN KECEPATAN MOTOR DC PADA MINIATUR MOBIL LISTRIK TERHADAP LINTASAN LENGKUNG

SKRIPSI

Diajukan untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

DisusunOleh:

R. PRAJA KUSUMANUGRAHA
NIM. 0910633015 - 63

Telah diperiksa dan disetujui oleh :

Pembimbing 1	**Pembimbing 2**
Ir. Purwanto, MT.	**Ir. Moch. Rusli, Dipl.-Ing,**
NIP. 19540424 198601 1 001	**NIP. 19630104 198701 1 001**

LEMBAR PENGESAHAN

PENGENDALIAN KECEPATAN MOTOR DC PADA MINIATUR MOBIL LISTRIK TERHADAP LINTASAN LENGKUNG

SKRIPSI

Disusun oleh;

R. PRAJA KUSUMANUGRAHA

NIM. 0910633015 - 63

Skripsi ini telah diuji dan dinyatakan lulus
pada tanggal 20 Agustus 2015

Majelis Penguji :

Dr., Ir. Erni Yudaningtyas.
NIP.19650913 199002 2 001

Goegoes Dwi N., ST., MT.
NIP. 19630104 198701 1 001

Dr., Ir. Bambang Siswoyo., MT.
NIP. 19630104 198701 1 001

Mengetahui

ketua jurusan Teknik Elektro

M. Aziz Muslim, ST., MT., Ph.D
NIP. 19741203 200012 1 001

KATA PENGANTAR

Alhamdulillah, segala puji hanya bagi Allah Subhanahu Wa Ta'ala, Rabb alam semesta. Dialah Allah, Tuhan Yang Maha Esa, Yang Maha Pengasih lagi Maha Penyayang. Sebaik baik Penolong dan Sebaik baik Pelindung. Shalawat serta salam kepada Nabi Muhammad Rasulullah Shallallahu Alaihi Wa Salam, Sang pembawa kabar gembira dan sebaik baik suri tauladan bagi yang mengharap Rahmat dan Hidayah-Nya.

Sungguh hanya melalui Pertolongan dan Perlindungan Allah SWT semata sehingga saya dapat menyelesaikan skripsi ini. Dengan seizin Allah SWT, di kesempatan yang baik ini saya ingin menghaturkan rasa terima kasih dan penghargaan yang sebesar besarnya atas bantuan sehingga terselesainya skripsi ini kepada:

- Keluarga tercinta, kedua orang tua R. Poernomo (alm) dan Siti Harini yang memberikan semangat dan inspirasi dalam penulisan penelitian ini. Serta kakak R. Sonny Yanupraja dan Swasti Komala yang telah memberikan dukungan dan saran.
- Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Hadi Suyono, ST.,MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Ir. Purwanto MT selaku KKDK Teknik Kontrol dan dosen pembimbing yang telah banyak memberikan pengarahan, bimbingan, nasehat, saran dan motivasinya.
- Ir. Moch. Rusli, Dipl.-Ing,. selaku dosen pembimbingyang telah banyak memberikan bimbingan, nasehat, saran, masukan dan motivasinya.
- Laboratorium Sistem Kontrol Teknik Elektro Universitas Brawijaya atas segala alat serta sarana dan prasarana yang dimanfaatkan penulis dalam melakukan penelitian.
- Teman-teman yang tidak bisa saya sebutkan satu-persatu, yang telah membantu semangat serta doa sehingga penelitian ini bisa diselesaikan.

Sekiranya Allah SWT mencatat amalan ikhlas kami dan semua pihak yang turut membantu sehingga skripsi ini terselesaikan. Akhirnya, kami menyadari bahwa skripsi ini masih jauh dari sempurna namun semoga skripsi ini dapat memberikan manfaat bagi kita semua. Allahumma Amîn.

Malang, 10 Agustus 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Abstrak - Pengendalian kecepatan motor DC terhadap lintasan yang dilakukan dalam penelitian ini bertujuan untuk merancang dan membuat sistem kontrol yang diterapkan kedalam miniatur mobil listrik secara diferensial supaya dapat mengikuti garis lengkung. Hasil identifikasi motor dengan menggunakan Matlab R2014B didapatkan fungsi alih sistem $\frac{1,102}{0,009425s^2+0.1243s+1,45}$.

Mikrokontroler menggunakan AVR ATmega16 kemudian sensor *photodiode* digunakan sebagai pengendali sensor garis atau lintasan, kemudian *rotary encoder* pada motor DC sebagai sensor kecepatan diberi kontroler berupa PI. Untuk mencari parameter PI menggunakan metode *direct* dan didapatkan hasil dari perhitungan dengan nilai PI dengan nilai Kp= 0,0084 dan Ti= 29,73

Kata kunci: sensor *Photodiode,*Motor DC, kontrol PI *direct*

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dengan semakin meningkatnya jumlah kendaraan bermotor terutama mobil, maka konsumsi bahan bakar minyak akan semakin meningkat. Bahan bakar minyak termasuk jenis energi yang tidak dapat diperbaruhi. Apabila tingkat konsumsi masyarakat pada energi berjenis bahan bakar minyak ini semakin tinggi maka akan terjadi kelangkaan energi bahan bakar minyak itu sendiri yang mengakibatkan kenaikan harga dari bahan bakar minyak karena kebutuhan semakin meningkat tidak berbanding lurus dengan jumlah produksinya. Cadangan minyak bumi di dunia untuk saat ini diperkirakan akan bertahan sampai 300 tahun lagi (esdm, 2014).

Penggunaan mobil listrik adalah salah satu cara untuk menekan jumlah konsumsi pemakaian bahan bakar minyak, karena mobil listrik ini menggunakan energi listrik. Sehingga para pengguna bisa menekan biaya pemakaian apabila menggunakan mobil listrik. Pemerintah sendiri menargetkan pada tahun 2014 mobil listrik akan diproduksi secara masal dengan *road map* industri yang sedang berjalan. Dalam pembahasan ini penulis merancang mobil listrik menggunakan sistem pengendali secara diferensial. Pengertian diferensial adalah roda kanan dan roda kiri dikontrol secara terpisah, sehingga dapat berbelok sesuai lintasan garis yang telah ditentukan terlebih dahulu. Perancangan mobil listrik ini menggunakan sistem kerja salah satu roda diperlambat dan dipercepat. Sistem diferensial ini berada pada penggerak roda belakang.

Maka dari itu penulisan ini, akan dibahas bagaimana merancang mobil diferensial yang mengikuti garis menggunakan dua motor DC untuk masing-masing roda. Pada penulisan ini penulis menggunakan dua roda bebas yang terletak didepan agar memudahkan sistem diferensial pada saat berada di lintasan lengkung, karena miniatur mobil listrik ini berjalan dengan otomatis. Untuk sistem pengendali menggunakan metode *Proporsional Integral Deferensial* (PID).

Dalam penulisan ini diharapkan dapat merancang kecepatan pada kedua motor DC supaya miniatur mobil listrik dapat mengikuti lintasan lengkung sesuai

dengan yang ditentukan terlebih dahulu. Pada penulisan ini penulis menggunakan kontrol PI pada kecepatan motor DC. Dan menggunakan sensor *photodiode* sebagai sensor garis.

1.2. Rumusan Masalah

Berdasarkan uraian dalam latar belakang, dapat disusun rumusan masalah sebagai berikut:

1. Bagaimana merancang sistem *hardware* dan *software* untuk mengatur kecepatan motor DC supaya miniatur mobil listrik dapat mengikuti garis lengkung.

1.3. Batasan Masalah

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

1. Pengontrolan digunakan untuk kecepatan motor DC.
2. Sensor yang digunakan adalah *rotary encoder*
3. Sensor garis yang digunakan adalah sensor *photodiode* tidak menggunakan sensor sudut.
4. Pemograman mikrokontroler (*slave*) menggunakan *softwareCodeVisionAVR Compiler* untuk pengolahan data.
5. Dimensi perancangan mobil listrik ini memiliki panjang 100cm, lebar 80cm dan tinggi 45cm.
6. Perhitungan fisika yang terjadi pada sistem tidak dibahas secara detail.
7. Sistem kerja pada driver motor DC dan sistem elektronika tidak dibahas secara mendalam.

1.4. Tujuan

Penelitian ini bertujuan untuk merancang dan membuat sistem yang dapat menggerakan mobil dengan sistem diferensial menggunakan metode *Proporsional Intergra* (PI).

1.5 Sistematika Penulisan

Agar penyusunan laporan skripsi ini dapat mencapai sasaran dan tidak menyimpang dari judul yang telah ditentukan, maka diperlukan sistematika

pembahasan yang jelas. Pembahasan dalam skripsi ini secara garis besar adalah sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Dasar Teori

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat.

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Perancangan dan perealisasian alat yang meliputi spesifikasi, perencanaan blok diagram, prinsip kerja dan realisasi alat.

BAB V Pengujian dan Analisis

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian di masa yang akan datang.

BAB II
DASAR TEORI

Dalam skripsi berjudul pengendalian posisi miniatur mobil listrik terhadap lintasan lengkung ini, untuk memudahkan dalam memahami cara kerja pada rangkaian ataupun dasar perancangan alat ini maka diperlukan penjalasan dan uraian teori penunjang yang digunakan dalam penulisan ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah sebagai berikut:

- Motor DC
- Mikrokontroler AVR ATMega 16
- PWM
- Program CodeVision AVR
- Potensio
- Rotary encoder
- Sensor *photodiode*
- Kontroler

2.1. Motor DC

Prinsip kerja motor DC sesuai dengan hukum kemagnetan Lorentz, yaitu membangkitkan fungsi magnet pada suatu konduktor berarus dalam medan magnet sehingga timbul ggl induksi. Setiap arus yang mengalir melalui sebuah konduktor akan menimbulkan medan magnet. Arah medan magnet dapat ditentukan dengan kaidah tangan kiri. Kaidah tangan kiri untuk motor menunjukkan arah arus yang mengalir didalam sebuah konduktor yang berada dalam medan magnet. Jari tengah menunjukan arah arus yang mengalir pada konduktor, jari telunjuk menunjukkan arah medan magnet dan ibu jari menunjukkan arah medan putar. Adapun besarnya gaya (f) yang bekerja pada konduktor tersebut dapat dirumuskan dengan (Soemarwanto,1999):

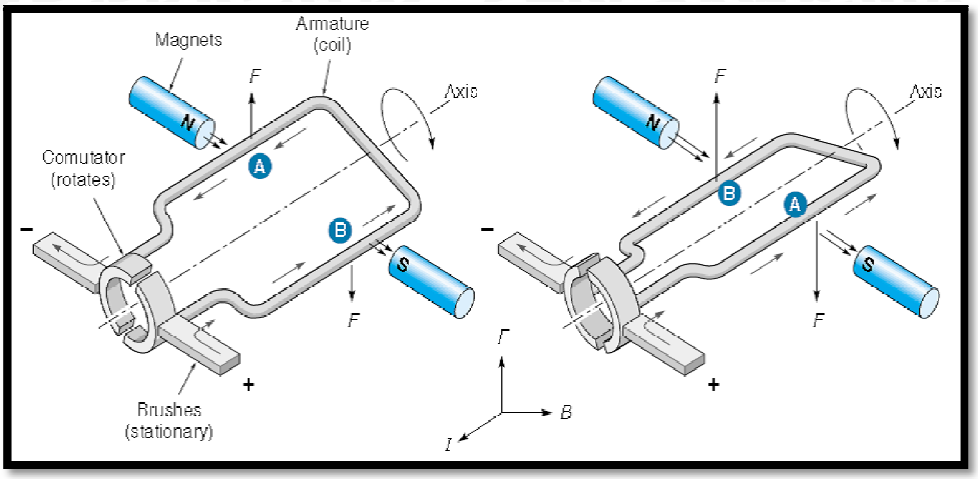
F = B.I.L (Newton)

Dimana :

B = kerapatan fluks magnet (weber)

L = panjang konduktor (meter)

I = arus listrik (ampere)



Gambar 2.1 Cara Kerja Motor DC
Sumber : Kilian, 1996

Gambar 2.1 adalah ilustrasi cara kerja motor DC yang mempunyai satu lilit kawat a dan b berada didalam medan magnet. Lilitan ini dapat berputar dengan bebas, lilitan ini biasa disebut dengan jangkar (*armature*).

Pada jangkar diberikan arus yang berasal dari sumber yang terhubung dengan sikat (*brushes*). Sikat-sikat ini terpasang pada sebuah cincin yang terbelah dua, yang disebut cincin belah (*commutator*). Adapun tujuan dari konstruksi ini adalah agar lilitan kawat dapat berputar apabila ada arus listrik yang melewatinya.

Pada kawat yang berada dikanan arus mengalir dari depan kebelakang. Pada kawat yang berada di bagian kiri, arus mengalir dari belakang kedepan kawat a dan b secara bergantian berada di kiri dan kanan. Karena itu arah arus di a dan arah arus di b selalu bersifat bolak-balik. Pembalikan arah arus itu terjadi pada saat lilitan kawat melintasi posisi vertikal.

Bagian *commutator* berfungsi sebagai penyearah mekanik. Fluksi magnet yang ditimbulkan magnet permanen disebut medan magnet motor. Dalam gambar 2.1 arah *fluks magnetic* adalah dari kiri ke kanan. Adapun gaya yang bekerja pada penghantar b adalah ke atas, sementara gaya yang bekerja pada penghantar a adalah ke bawah. Gaya-gaya yang bekerja sama kuatnya, sehingga terdapat kopel yang bekerja pada kawat sehingga lilitan jangkar dapat berputar. Setelah berputar 180° arah arus berbalik, pada saat itu penghantar a dan b bertukar tempat. Akibatnya arah gerak putaran tidak berubah.

2.1.1 Menentukan Parameter Motor DC

Untuk menentukan fungsi alih dari motor DC metode yang digunakan adalah pemodelan fisik, metode pemodelan fisik adalah identifikasi paramater fisik dan elektrik pada Motor DC. Parameter yang digunakan dalam penentuan fungsi alih adalah resistansi, induktansi, konstanta motor, koefisien vricsi dan Momen inersia.

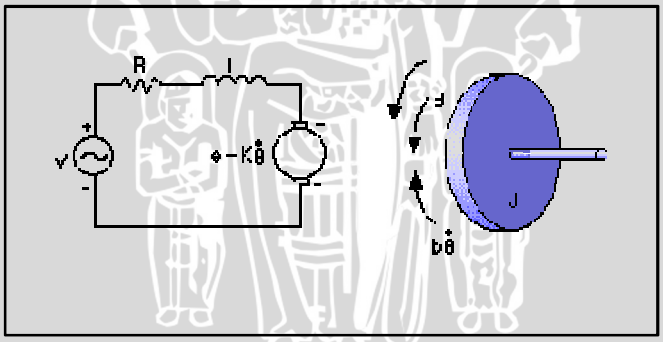
Karakteristik motor DC adalah nilai torsi sebaanding dengan arus jangkari (i) dengan faktor konstanta (K_t), seperti yang ditunjukan pada persamaan 4.1

$$T = K_T i \quad \text{.....(2.1)}$$

Back-emf (e) adalah jika sebuah konduktor listrik memotong garis medan magnet maka timbul gaya gerak listrik pada konduktor.

$$e = K_e \quad \text{.....(2.2)}$$

Dalam SI unit, torsi motor dengan *back-emf* adalah sebanding maka nilai dari $K_t = K_e$, oleh karena itu untuk mempermudah dapat digunakan K untuk mewakili dari nilai K_t dan K_e.



Gambar 4.8 Rangkaian motor DC

Dari gambar 4.8 maka dapat diuraikan menggunakan hukum Newton II dan Hukum tegangan Kirchoff.

$$\Sigma F = ma$$

$$K_i - b = J$$

$$J + b = K_i \quad \text{.....(2.3)}$$

$$\Sigma V = 0$$

$$L\frac{di}{dt} + R_i + K - V = 0$$

$$L\frac{di}{dt} + R_i = V - K \quad \text{.....(2.4)}$$

Dari persamaan (3) dan (4) dapat diturunkan menggunakan transformasi laplace maka didapatkan persamaan:

$$s(Js + b)(s) = KI(s) \quad \text{.....(2.5)}$$

$$(Ls + R)I(s) = V(s) - Ks(s) \quad \text{.....(2.6)}$$

Untuk mendapatkan fungsi alih maka $I(s)$ dihilangkan pada persamaan (5) dan (6), dimana kecepatan rotasi dianggap sebagai keluaran dan masukan dianggap sebagai masukan.

$$P(s) = \frac{(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2}\frac{rad/s}{V} \quad \text{.....(2.7)}$$

Apabila persamaan fungsi alih sudah didapatkan, untuk nilai resistansi (R) dan nilai induktansi (L) dapat dicari dengan menggunakan alat ukur. Selanjutnya untuk mencari nilai dari *back-emf* didapatkan dengan persamaan:

$$e = V - IR \quad \text{.....(2.8)}$$

Untuk mencari nilai dari K didapatkan dengan mengukur kecepatan motor ω (dalam rad/s) pada saat diberi sumber tegangan (V) dari persamaan (4.2) dan (4.8). Sehingga didapatkan persamaan:

$$K = \frac{e}{\omega} \quad \text{.....(2.9)}$$

Selanjutnya untuk mengestimasi nilai koefisien dari nilai friksi (b) motor menggunakan persamaan (3). Motor dinyalakan hingga mencapai kecepatan konstan sehingga $\frac{d\theta}{dt} = 0$ atau bisa diasumsikan bahwa nilai percepatanya adalah 0.

$$b = \frac{Ki}{\omega} \quad \text{.....(2.10)}$$

Yang terakhir adalah menentukan nilai dari momen inersia dari motor. Motor dinyalakan sampai mencapai kecepatan yang konstan kemudian tidak diberi sumber tegangan atau V=0, sehingga nilai arus sama dengan 0.

$J + b = 0$ .....(2.11)

Pada persamaan (4.11) diperoleh dari dengan cara pada saat kecepatan *steady state* sebagai awal kecepatan dalam persamaan.

$\omega = \frac{Te}{B} e^{-\left(\frac{B}{\tau}\right)t}$ .....(2.12)

Ketika nilai (*t)* sama dengan $\tau = \frac{J}{B}$ adalah waktu konstan pada saat di beri sumber tegangan (V). Kemudian (V) dilepas hingga kondisi motor berhenti. Karena nilai b sudah diketahui dari persamaan (10), maka akan didapatkan nilai momen inersia dari motor DC.

2.2. Mikrokontroler AVR ATMega 16

AVR merupakan seri mikrokontroler *Complementary Metal Oxide Semiconductor* (CMOS) 8-*bit* buatan Atmel berbasis arsitektur RISC *(Reduced Instruction Set Computer*).

Fitur AVR ATMEGA16 antara lain:

1. Mikrokontroler AVR 8 *bit* yang memilliki kemampuan tinggi dengan konsumsi daya rendah
2. Arsitektur RISC dengan *throughput* mencapai 16 MIPS pada frekuensi 16MHz
3. Memiliki kapasitas *Flash* memori 16 Kbyte, EEPROM 512 Byte dan SRAM 1 Kbyte
4. Saluran I/O sebanyak 32 buah, yaitu *Port* A, *Port* B, *Port* C dan *Port* D
5. CPU yang terdiri dari 32 buah *register*
6. Unit interupsi dan eksternal
7. *Port* USART untuk komunikasi serial
8. Fitur *peripheral*

   -Tiga buah *Timer/Counter* dengan kemampuan perbandingan (*compare*)

   -Dua buah *Timer/Counter* 8 *bit* dengan *Prescaler* terpisah dan *Mode Compare*

-Satu buah *Timer/Counter* 16 *bit* dengan *Prescaler* terpisah, *Mode Compare* dan *Mode Capture*

-*Real Time Counter* dengan *Oscillator* tersendiri

-Empat kanal PWM

-8 kanal ADC

-8 *Single-ended Channel* dengan keluaran hasil konversi 8 dan 10 resolusi (register ADCH dan ADCL)

-7 *Diferrential Channel* hanya pada kemasan *Thin Quad Flat Pack* (TQFP)

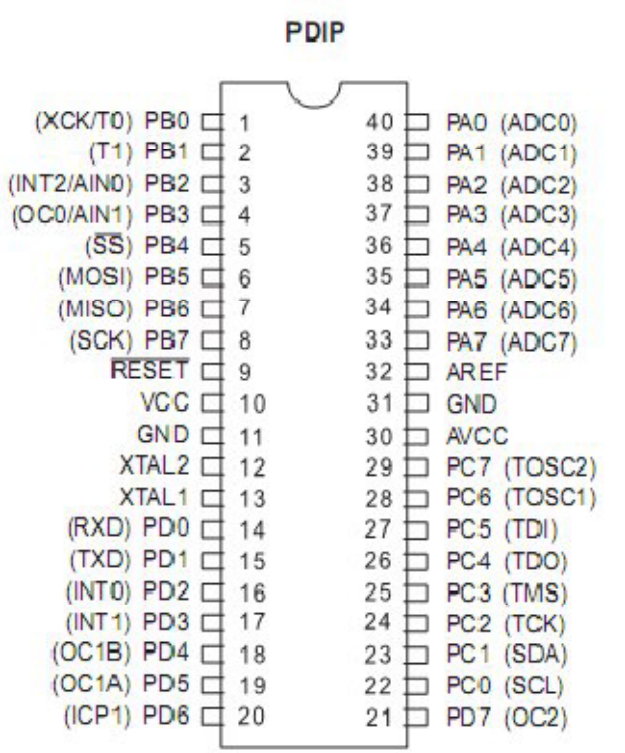
-2 *Differential Channel* dengan *Programmable Gain*

-Antarmuka *Serial Peripheral Interface* (SPI) *Bus*

-*Watchdog Timer* dengan *Oscillator Internal*

-*On-chip Analog Comparator*

9. *Non-volatile program memory*



Gambar 2.2 Konfigurasi Pin ATMega16
Sumber: ATMEL, 2010

Konfigurasi pin mikrokontroler ATMega 16 meliputi :

- VCC : Suplai tegangan.
- GND : Ground.
- Port A (PA7..PA0)

Bandar A berfungsi sebagai *input* analog pada konverter A/D. Bandar A juga sebagai suatu bandar I/O 8-bit dua arah, jika A/D konverter tidak digunakan. Pena -Pena Bandar dapat menyediakan resistor *internal pull-up* (yang dipilih untuk masing-masing bit). Bandar A *output* buffer mempunyai karakteristik gerakan simetris dengan keduanya sink tinggi dan kemampuan sumber. Ketika pena PA0 ke PA7 digunakan sebagai input dan secara eksternal ditarik rendah, pena–pena akan memungkinkan arus sumber jika resistor *internal pull-up*diaktifkan. Pena Bandar A adalah *tri-stated* manakala suatu kondisi *reset* menjadi aktif, sekalipun waktu habis.

- Port B (PB7..PB0)

Bandar B adalah suatu bandar I/O 8-bit dua arah dengan resistor *internalpull-up*(yang dipilih untuk beberapa bit). Bandar B output *buffer*

mempunyaikarakteristik gerakan simetris dengan keduanya sink tinggi dan kemampuan sumber. Sebagai input, pena Bandar B yang secara eksternal ditarik rendah akanarus sumber jika resistor *pull-up* diaktifkan. Pena Bandar B adalah *tri-stated* manakala suatu kondisi reset menjadi aktif, sekalipun waktu habis.

- Port C (PC7..PC0)

Bandar C adalah suatu bandar I/O 8-bit dua arah dengan resistor *internal pull-up* (yang dipilih untuk beberapa bit). Bandar C *outputbuffer* mempunyai karakteristik gerakan simetris dengan keduanya sink tinggi dan kemampuan sumber. Sebagai input, pena bandar C yang secara *eksternal* ditarik rendah akan arus sumber jika resistor *pull-up* diaktifkan. Pena bandar C adalah *tri-stated* manakala suatu kondisi *reset* menjadi aktif, sekalipun waktu habis.

- Port D (PD7..PD0)

Bandar D adalah suatu bandar I/O 8-bit dua arah dengan resistor *internal pull-up*(yang dipilih untuk beberapa bit). Bandar D *output buffer* mempunyai karakteristik gerakan simetris dengan keduanya sink tinggi dan kemampuan sumber. Sebagai input, pena bandar D yang secara eksternal ditarik rendah akan arus sumber jika resistor *pull-up* diaktifkan. Pena Bandar D adalah tri-stated manakala suatu kondisi reset menjadi aktif, sekalipun waktu habis.

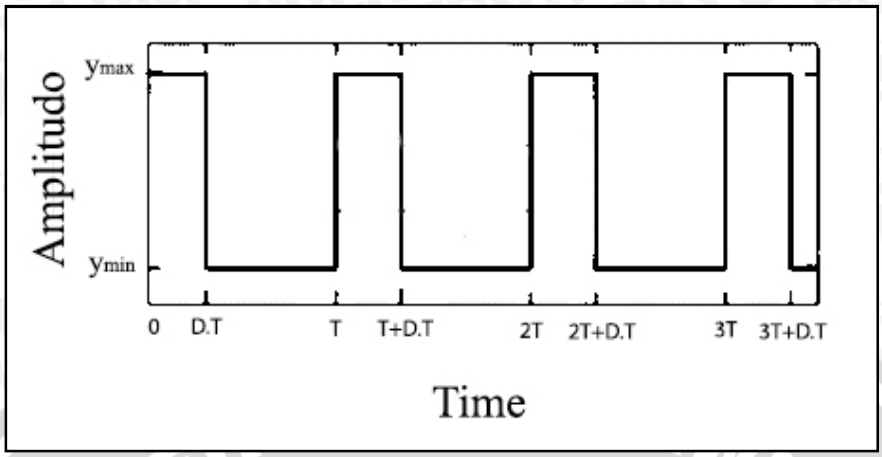
- RESET (*Reset input*)
- XTAL1 (*Input Oscillator*)
- XTAL2 (*Output Oscillator*)
- AVCC adalah pena penyedia tegangan untuk bandar A dan Konverter A/D.
- AREF adalah pena referensi analog untuk konverter A/D

2.3. PWM

PWM (*Pulse Width Modulation*) digunakan untuk mengatur kecepatan dari motor DC. Kecepatan motor DC tergantung pada besarnya *duty cycle* yang diberikan pada motor DC tersebut.

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *duty cycle* bervariasi dari 0% - 100%. Dengan mengatur *duty cycle* akan diperoleh keluaran

yang diinginkan. Sinyal PWM (*Pulse Width Modulation*) secara umum dapat dilihat dalam gambar 6.3 berikut:



Gambar 6.3 Gelombang kotak yang memiliki y_{max}, y_{min} dan *D*
Sumber: Heri Andrianto, 2008 : 137

$$Dutycycle = \frac{Ton}{T} \times 100(\%) \quad \text{.....(2.12)}$$

Dengan:

Ton = Periode logika tinggi

T = Periode keseluruhan

$$Vdc = Dutycycle \times Vcc(\text{V}) \quad \text{.....(2.13)}$$

Sedangkan frekuensi sinyal dapat ditentukan dengan rumus berikut:

$$fOn = \frac{fclkI/O}{N.256}(\text{Hz}) \quad \text{.....(2.14)}$$

2.4. Program CodeVision AVR

CodeVisionAVR merupakan sebuah cross-compiler C, *Integrated Development Environtment* (IDE), dan *Automatic Program Generator* yang didesain untuk mikrokontroler buatan Atmel seri AVR. CodeVisionAVR dapat dijalankan pada sistem operasi Windows 95, 98, Me, NT4, 2000, dan XP.

Cross-compiler C mampu menerjemahkan hampir semua perintah dari bahasa ANSI C, sejauh yang diijinkan oleh arsitektur dari AVR, dengan tambahan beberapa fitur untuk mengambil kelebihan khusus dari arsitektur AVR dan kebutuhan pada sistem *embedded*.

File object COFF hasil kompilasi dapat digunakan untuk keperluan *debugging* pada tingkatan C, dengan pengamatan variabel, menggunakan *debugger* Atmel AVR Studio.

IDE mempunyai fasilitas internal berupa software AVR Chip In-System Programmer yang memungkinkan pengguna untuk melakukan transfer program kedalam chip mikrokontroler setelah sukses melakukan kompilasi/asembli secara otomatis. *Software* In-System Programmer didesain untuk bekerja dengan Atmel STK500/AVRISP/AVRProg, Kanda Systems STK200+/300, Dontronics DT006, Vogel Elektronik VTEC-ISP, Futurlec JRAVR dan MicroTronics ATCPU/Mega2000 *programmers/development boards.*

Untuk keperluan *debugging* sistem *embedded*, yang menggunakan komunikasi serial, IDE mempunyai fasilitas internal berupa sebuah Terminal.

Selain *library* standar C, CodeVisionAVR juga mempunyai *library* tertentu untuk:

a) Modul LCD *alphanumeric*
b) Bus I2C dari Philips
c) Sensor Suhu LM75 dari *National Semiconductor*
d) *Real-Time Clock*: PCF8563, PCF8583 dari Philips, DS1302 dan DS1307 dari Maxim/Dallas *Semiconductor*
e) Protokol 1-Wire dari Maxim/Dallas *Semiconductor*
f) Sensor Suhu DS1820, DS18S20, dan DS18B20 dari Maxim/Dallas *Semiconductor*
g) Termometer/Termostat DS1621 dari Maxim/Dallas *Semiconductor*
h) EEPROM DS2430 dan DS2433 dari Maxim/Dallas *Semiconductor*
i) SPI
j) *Power Management*
k) Delay
l) Konversi ke Kode Gray

CodeVisionAVR juga mempunyai *Automatic Program Generator* bernama CodeWizardAVR, yang mengijinkan pengguna untuk menulis, dalam hitungan menit, semua instruksi yang diperlukan untuk membuat fungsi-fungsi berikut:

a. *Set-up* akses memori eksternal
b. Identifikasi sumber *reset* untuk *chip*
c. Inisialisasi port *input/output*
d. Inisialisasi interupsi eksternal
e. Inisialisasi *Timer/Counter*
f. Inisialisasi *Watchdog-Timer*
g. Inisialisasi UART (USART) dan komunikasi *serial* berbasis *buffer* yang digerakkan oleh interupsi
h. Inisialisasi Pembanding Analog
i. Inisialisasi ADC
j. Inisialisasi Antarmuka SPI
k. Inisialisasi Antarmuka Two-Wire
l. Inisialisasi Antarmuka CAN
m. Inisialisasi Bus I2C, Sensor Suhu LM75, *Thermometer/Thermostat* DS1621 dan *Real-Time Clock* PCF8563, PCF8583, DS1302, dan DS1307
n. Inisialisasi *Bus 1-Wire* dan Sensor Suhu DS1820, DS18S20
o. Inisialisasi modul LCD

2.5. Potensio

Potensio adalah resistor tiga terminal dengan sambungan geser yang membentuk pembagi tegangan dapat disetel. Jika hanya dua terminal yang digunakan (salah satu terminal tetap dan terminal geser), potensiometer berperan sebagai resistor variabel atau *Rheostat*. Potensiometer biasanya digunakan untuk mengendalikan peranti elektronik seperti pengendali suara pada penguat. Potensiometer yang dioperasikan oleh suatu mekanisme dapat digunakan sebagai transduser, misalnya sebagai sensor *joystick*.

Potensiometer biasanya digunakan untuk mengendalikan peranti elektronik seperti pengendali suara pada penguat. Potensiometer yang dioperasikan oleh suatu mekanisme dapat digunakan sebagai *transduser*, misalnya sebagai sensor *joystick*.

Potensiometer jarang digunakan untuk mengendalikan daya tinggi (lebih dari 1 Watt) secara langsung. Potensiometer digunakan untuk menyetel taraf

isyarat analog (misalnya pengendali suara pada peranti audio), dan sebagai pengendali masukan untuk sirkuit elektronik. Sebagai contoh, sebuah peredup lampu menggunakan potensiometer untuk menendalikan pensakelaran sebuah TRIAC, jadi secara tidak langsung mengendalikan kecerahan lampu.

Potensiometer yang digunakan sebagai pengendali volume kadang-kadang dilengkapi dengan sakelar yang terintegrasi, sehingga potensiometer membuka sakelar saat penyapu berada pada posisi terendah.

Potensiometer kadang-kadang dilengkapi dengan satu atau lebih tombol menjulang pada batang yang sama. Sebagai contoh, ketika berkait dengan suatu pengatur volume, tombol dapat juga berfungsi sebagai suatu on/off tombol di volume yang paling rendah.

2.6. Rotary Encoder

Sensor *rotary encoder* yang digunakan pada penulisan ini merupakan sensor yang diproduksi oleh Autonics dengan tipe E40H8-500P/R-6-L-5 seperti pada Gambar 6.13. Sensorini memiliki lubang berukuran 80mm sebagai tempat untuk menghubungkan antara sensor dengan roda. Keluaran dari sensor ini berupa pulsa dengan tegangan 5v ±5% saat pulsa *high* dan 0v saat pulsa *low.* Sensor ini dapat menyediakan 500 pulsa /rotasi. Pulsa keluaran dari sensor ini dapat didenfinisikan menjadi gerakan, posisi dan arah. Sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh *rotary encoder* untuk diteruskan oleh sistem rangkaian kendali.

Gambar 6.4 Rotary Encoder
Sumber: autonics 2013

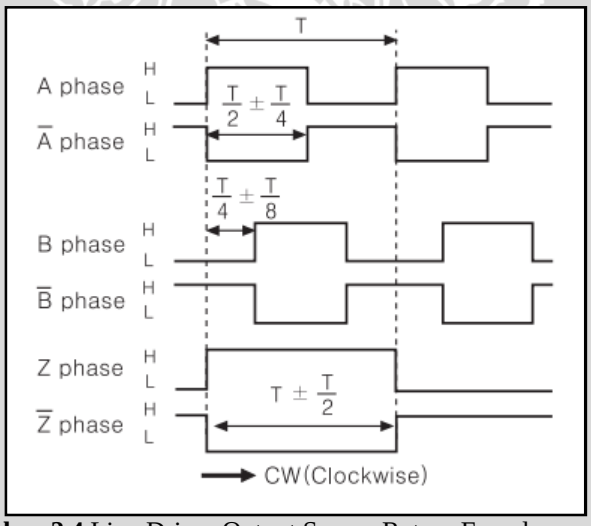
Sensor ini memiliki 9 kaki atau pin. Konfigurasi dari masing-masing pin ditunjukan pada Tabel 2.1

Tabel 2.1 Konfigurasi pin dengan warna kabel

No pin	Fungsi	Warna Kabel
1	OUT A	Hitam
2	OUT A'	Merah
3	+ V	Coklat
4	GND	Biru
5	OUT B	Putih
6	OUT B'	Abu-abu
7	OUT Z	Jingga
8	OUT Z'	Kuning
9	F.G	Pelindung kabel

sumber: *Datasheet Autonics Rotary Encoder*

Perbedaan antara masing-masing keluaran dari sensor *rotary encoder* ini dapat kita lihat pada Gambar 2.4



Gambar 2.4 Line Driver Output Sensor Rotary Encoder

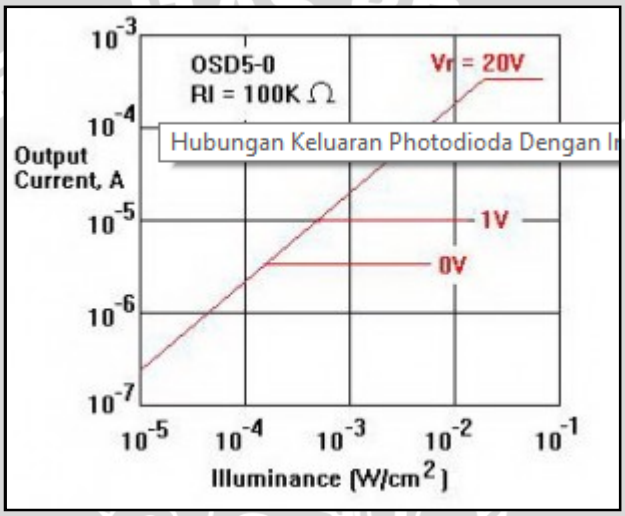
Pada Gambar 2.4 dapat diketahui bahwa terdapat pergeseran fasa antara OUT A dan OUT B dengan periode pulsa yang sama, sedangkan OUT Z memiliki periode 2 kali lebih lama. Fasa yang berkebalikan juga disediakan oleh rotaru ini pada OUT A', OUT B' dan OUT Z'.

2.7. Sensor Photodiode

Sensor *photodiode* adalah salah satu jenis sensor yang peka terhadap cahaya (*photodetector). Photodiode* akan mengalirkan arus yang membentuk

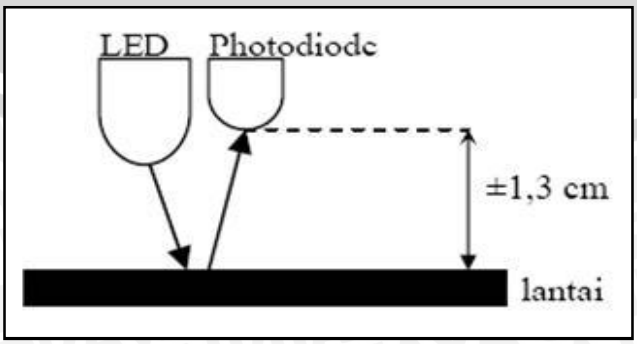
fungsi linier terhadap insensitas cahaya yang diterima. Arus ini pada umumnya teratur terhadap *power density* (Dp). Perbandingan antara arus keluaran dengan *power density* disebu sebagai *current responsitivity.* Arus yang dimaksud adalah arus bocor ketika *photodiode* tersebut disinari dan dalam keadaan dipanjar mundur.

Hubungan antara keluaran sensor *photodiode* dengan isensitas cahaya yang diterimanya ketika dipanjar mundur adalah membentuk suatu fungsi yang linier. Gambar 2.5 menunjukan grafik hubungan antara keluaran sensor *photodiode* dengan isensitas cahaya.



Gambar 2.5 *Hubungan keluaran photodiode dengan isensitas cahaya*
(sumber: http://elektronika-dasar.web.id/)

LED *superbright* berfungsi sebagai pengirim cahaya ke garis untuk dipantulkan kemudian dibaca oleh sensor *photodiode.* Sifat dari warna putih (permukaan terang) yang memantulkan cahaya berwarna hitam (permukaan gelap) yang tidak memantulkan cahaya digunakan dalam aplikasi ini. Gambar 2.6 adalah ilustrasi sistem kerja sensor garis.



Gambar 2.6 Ilustrasi sistem kerja sensor garis
(sumber: https://fahmizaleeits.wordpress.com)

2.8 Kontroler

Keberadaan kontroler pada sebuah sistem kontrol mempunyai kontribusi yang besar perilaku suatu sistem. Hal itu disebabkan oleh karena tidak dapat diubahnya suatu komponen penyusun pada sistem rangkaian tersebut. Ini dapat diartikan, karakteristik plant harus diterima sebagai adanya, sehingga perubahan perilaku suatu sistem hanya dapat dilakukan oleh suatu sub sistem, yaitu kontroler.

Salah satu tugas dari komponen disuatu kontroler adalah mereduksi sinyal *error*, yaitu perbedaan antara sinyal *input* dan *output*. Hal ini sesuai dengan tujuan sistem kontrol yaitu mendapatkan sinyal *output* yang sesuai dengan sinyal *input*. Semakin cepat reaksi sistem mengikuti sinyal *output* dan semakin kecil *error* yang terjadi, maka semakin baik kinerja sistem kontrol yang diterapkan.

Apabila terjadi perbedaan yang signifikan antara sinyal input dengan sinyal output relatif besar, maka kontroler yang baik seharusnya mampu menganalisis perbedaan yang terjadi, hal ini dapat dikatakan kinerja sistem sesuai dengan yang diharapkan.

Sistem kontrol kaskade adalah sistem kontrol yang banyak digunakan di bidang proses industri. Kaskade kontrol mempunyai dua *feedback* kontrol dimana output utama (primer) menjadi setpoint dari pengontrol sekunder (*slave*). *Output* dari kontroler sekunder (*slave*) yang akan menggerakan *final control* (*valve*).

1. Pengendalian dengan *loop* terbuka

Sistem kontrol *loop* terbuka adalah sistem kontrol yang keluarannya tidak berpengaruh pada aksi pengontrolan. Jadi pada sistem kontrol *loop* terbuka, keluaran tidak diukur atau diumpan balikan untuk dibandingkan dengan masukan.

2. Pengendalian dengan *loop* tertutup

Sistem kontrol *loop* tertutup adalah sistem kontrol yang keluarannya mempunyai pengaruh langsung pada aksi pengontrolan. Disebut juga sistem kontrol yang menggunakan umpan balik untuk memperkecil kesalahan sistem.

2.8.1. Kontroler Proporsional

Kontroler proporsional memiliki keluaran yang sebanding dengan sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya). Untuk kontroler dengan aksi kontrol proporsional, hubungan antara keluaran kontroler *m(t)* dan sinyal kesalahan penggerak *e(t)* adalah:

$m(t)=K_p.e(t)$……(2.15)

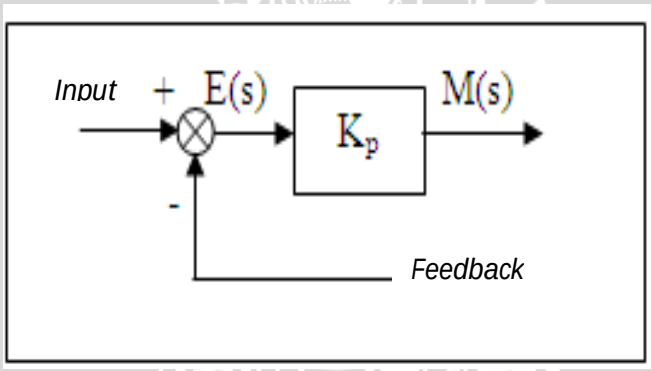
atau, dalam besaran transformasi Laplace,

$\frac{M(s)}{E(s)} = Kp$……(2.16)

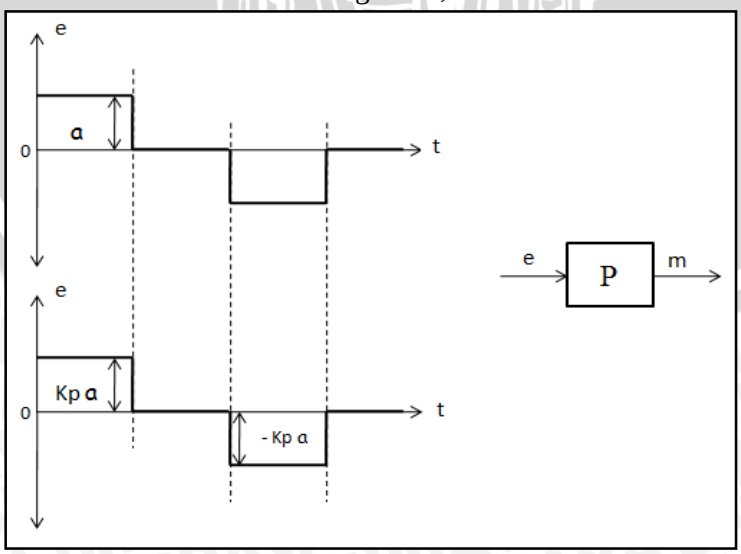
Di mana K_p adalah kepekaan proporsional atau penguatan.

Apapun wujud mekanisme yang sebenarnya dan apapun bentuk daya penggeraknya, kontroler proporsional pada dasarnya merupakan penguat dengan penguatan yang dapat diatur (Ogata K.,1997).

Diagram blok kontroler proporsional ditunjukkan dalam Gambar 2.7



Gambar 2.7 Diagram Blok Kontroler Proposional
Sumber: Ogata K., 1997



Gambar 2.8 Kurva waktu hubungan input-output pada kontrol proposional

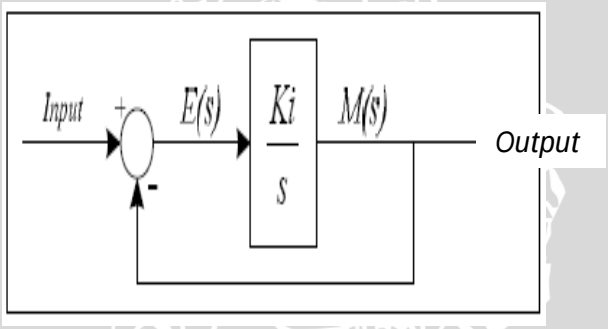
2.8.2. Kontroler Integral

Kontroler integral berfungsi mengurangi kesalahan keadaan mantap pada kontroler proporsional sebelumnya. Pada kontroler dengan aksi integral, harga. keluaran kontroler *m(t)* diubah dengan laju yang sebanding dengan sinyal kesalahan penggerak *e (t).*

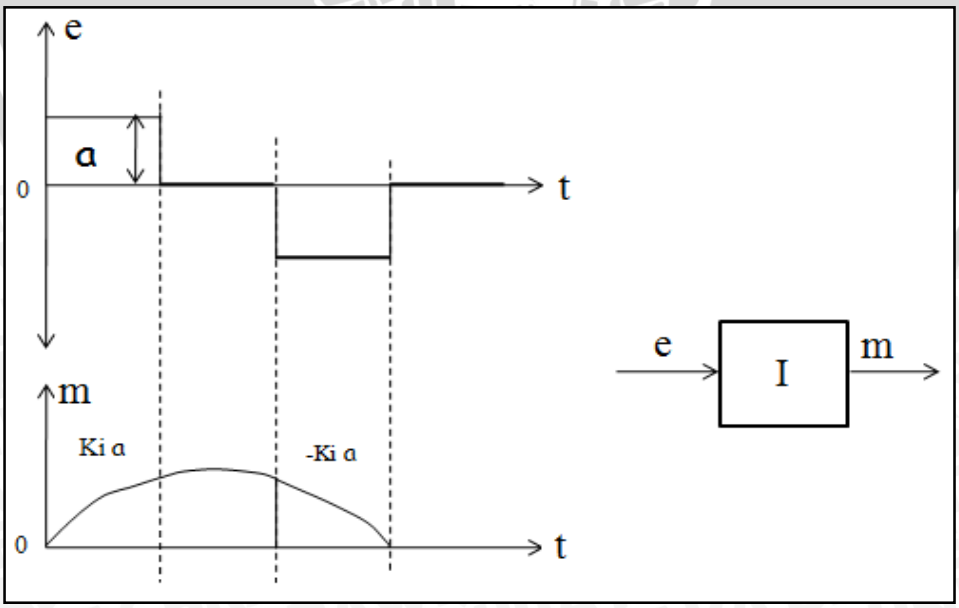
Jadi,

$$\frac{dm(t)}{dt} = Ki.e(t) \quad \text{.....(2.17)}$$

Jika harga *e(t)* diduakalikan, maka harga *m(t)* berubah dengan laju perubahan menjadi dua kali semula. Jika kesalahan penggerak nol, maka harga *m(t)* tetap stasioner. Aksi kontrol integral seringkali disebut *controlreset* (Ogata K.,1997). Gambar 2.8 menunjukkan diagram blok kontroler integral.



Gambar 2.8 Diagram Blok Kontroler Integral
Sumber: Ogata K., 1997



Gambar 2.8 Kurva waktu hubungan input-output pada kontrol integral

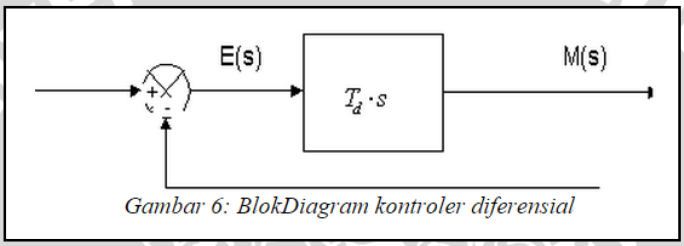
2.8.3. Kontroler Diferensial

Kontroler ini digunakan untuk memperbaiki atau mempercepat respons transien sebuah sistem kontrol dengan cara memperbesar *phase lead* terhadap penguatan kontrol dan mengurangi *phase lag* penguatan tersebut (Ogata K.,1997).

Kontroler diferensial tidak dapat mengeluarkan *output* bila tidak ada perubahan *input*, selain itu kontroler differensial tidak dapat digunakan untuk proses yang mengandung *noise*. Hubungan antara keluaran kontroler *m(t)* dan sinyal kesalahan penggerak *e(t)* adalah :

$$\frac{M(s)}{E(s)} = Kd.s \quad \text{.....(2.18)}$$

Gambar 2.9 menunjukkan diagram blok kontroler diferensial.



Gambar 2.9 Diagram Blok Kontroler Diferensial
Sumber: Teknik Kontrol Automatik, Katsuhiko Ogata, 1997

2.8.4. Kontroler Proporsional Integral Diferensial (PID)

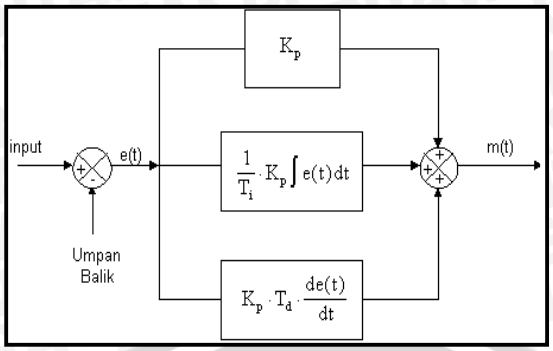
Gabungan aksi kontrol proporsional, integral, dan diferensial mempunyai keunggulan dapat saling menutupi kekurangan dan kelebihan dari masing-masing kontroler. Elemen-elemen P, I, dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar. Persamaan kontroler PID ini dapat dinyatakan sebagai berikut :

$$m(t) = Kp\,.e(t) + \frac{Kp}{Ti}.e(t)dt + Kp.Td\frac{de(t)}{dt} \quad \text{.....(2.19)}$$

Dalam transformasi Laplace dinyatakan sebagai berikut :

$$\frac{M(s)}{E(s)} = Kp\,(1 + \frac{1}{Ti.s} + Td.s) \quad \text{.....(2.20)}$$

Gambar 2.10 menunjukkan diagram blok kontroler PID

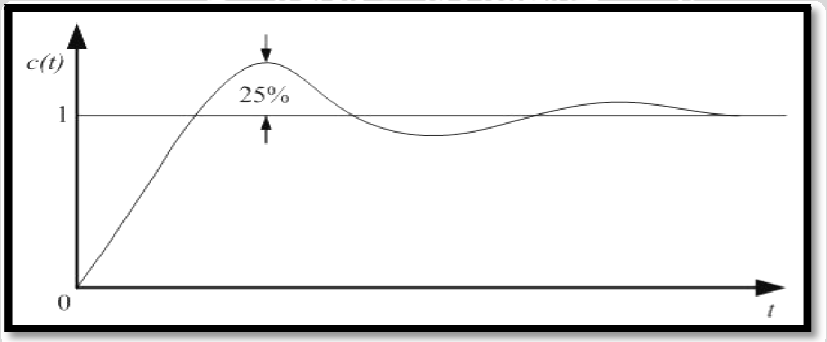


Gambar 2.10 Diagram Blok Kontroler PID
Sumber: Ogata K., 1997

2.8.5. Metode Kontroler Proporsional Integral Diferensial (PID)

Ziegler dan Nichols mengemukakan aturan-aturan untuk menentukan nilai dari gain proporsional Kp, waktu integral Ti, dan waktu derivatif Td berdasarkan karakteristik respon transien dari *plant* yang diberikan. Penentuan parameter kontroler PID atau penalaan kontroler PID tersebut dapat dilakukan dengan bereksperimen dengan plan.(Ogata, K., 1997)

Terdapat dua metode yang disebut dengan aturan penalaan Ziegler-Nichols, pada kedua metode tersebut memiliki tujuan yang sama yaitu untuk mencapai 25% *maximum overshoot* pada respon unit step, ditunjukkan dengan Gambar 2.11



Gambar 2.11 Kurva Respon Unit Step yang Menunjukkan 25% *Maximum Overshoot*
Sumber: Ogata, K., 1997

BAB III
METODOLOGI PENELITIAN

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat di bab pendahuluan maka diperlukan metode untuk menyelesaikan masalah tersebut.

3.1. Studi Literatur

Studi literatur dilakukan agar dapat memahami komponen penyusun sistem. Pentuan kontroler PI *direct*, motor DC, *rotary encoder*, ATMega16 dan *software CodeVisionAVR C Compiler*.

3.2. Penentuan Spesifkasi Sistem Kontrol

Spesifikasi sistem kontrol ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut:

a. Jari-jari roda mobil sebesar 7,5 cm dan diameter 15 cm.
b. Miniatur mobil listrik berbahan dasar alumunium.
c. Dimensi mobil yaitu panjang 80 cm, lebar 100 cm dan tinggi 20 cm.
d. Menggunakan sensor garis untuk melakukan pengujian miniatur mobil listrik terhadap lintasan garis yang telah ditentukan.
e. Dua buah *Rotary Encoder* sebagai sensor kecepatan yang dipasang pada masing-masing motor DC.
f. Menggunakan 1 buah Mikrokontroler ATMega 16 sebagai pengendali utama *mobile robot*.
g. Dua roda depan merupakan roda bebas dan dua roda belakang yang dikopel motor DC sebagai penggerak dengan sistem differensial (terpisah).

3.3. Perancangan dan Perealisasian Alat

3.3.1. Perancangan Perangkat Keras dan Realisasi Tiap Blok

1. Penentuan spesifikasi alat
2. Pembuatan blok diagram lengkap sistem

3. Pembuatan mekanik robot
4. Penentuan dan perhitungan komponen yang akan digunakan
5. Merakit perangkat keras masing-masing blok

3.3.2. Perancangan dan Penyusunan Perangkat Lunak

Untuk perancangan perangkat lunak hal pertama yang dilakukan adalah mengetahui karakteristik motor DC dengan cara mencari fungsi alih dari motor DC, setelah itu dilakukan perhitungan menggunkan metode *direct* untuk menentukan algoritma kontroler dengan kontrol PI *direct* sebagai kontrol kecepatan motor DC. Setelah didapatkan algoritma, kemudian dibuat program untuk mikrokontroler dengan *software CodeVisionAVR C Compiler*.

3.4. Pengujian Alat

Pengujian alat dilakukan untuk memastikan bahwa sistem ini berjalan sesuai yang direncanakan. Pengujian alat meliputi pengujian perangkat keras yang dilakukan baik per blok maupun keseluruhan sistem.

3.4.1. Pengujian Tiap Blok

Pengujian per blok dilakukan dengan tujuan untuk menyesuaikan nilai masukan dan nilai keluaran tiap-tiap blok sesuai dengan perancangan yang dilakukan sebelumnya.

3.4.2. Pengujian Keseluruhan Sistem

Pengujian sistem secara keseluruhan dilakukan dengan tujuan untuk mengetahui untuk kerja alat setelah perangkat keras dan perangkat lunak diintegrasikan bersama.

3.4.3. Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah didapatkan hasil dari pengujian. Jika hasil yang diperoleh telah sesuai dengan spesifikasi yang direncanakan maka alat tersebut telah memenuhi harapan dan memerlukan pengembangan.

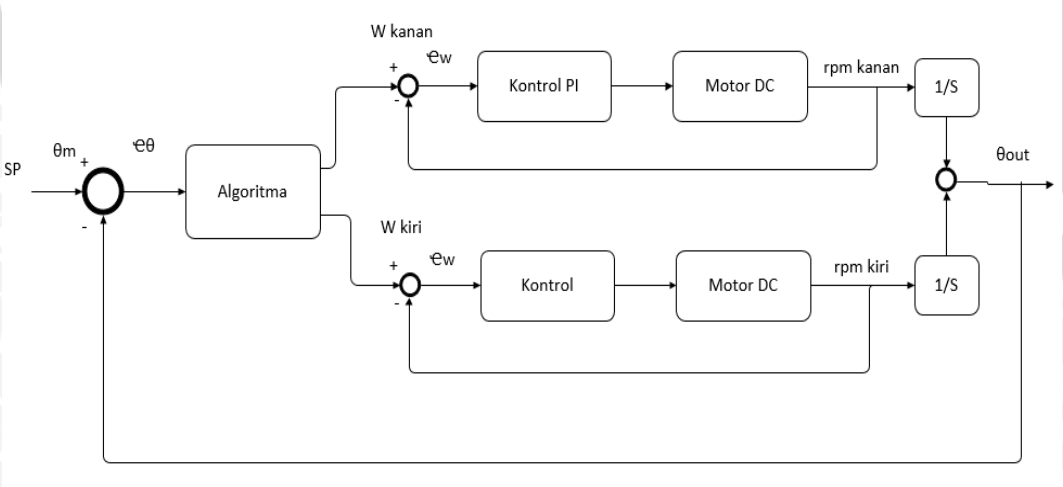
BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Perancangan miniatur mobil listrik ini dilakukan secara bertahap sehingga akan memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. Sistem kerja perangkat tersebut meliputi perangkat keras dan perangkat lunak sedangkan pembuatan bertujuan untuk menghasilkan semua perangkat pendukung maupun alat secara keseluruhan yang terdiri dari:

- Diagram blok sistem.
- Pemodelan perangkat keras (*hardware*).
- Pemodelan sistem kontroler kaskade.
- Pemodelan perangkat lunak.
- Penentuan fungsi alih motor DC.

4.1. Diagram Blok Sistem

Pada perancangan sistem ini diperlukan blok diagram sistem untuk menjelaskan sebagian besar sistem kerja miniatur mobil listrik dan diharapkan sistem kerja miniatur mobil listrik sesuai dengan yang direncanakan. Blok diagram dapat dilihat pada Gambar 4.1



Gambar 4.1 Diagram blok sistem

Berikut merupakan komponen-komponen dari diagram blok 4.1:

- *Set Point* : Lintasan lengkung yang telah ditentukan.
- *Error detector* : Selisih antara *input* dan *respons* melalui umpan

balik.

- Kontroler : Kontroler yang digunakan adalah Proposional. Integral. Pengolahan PI menggunakan miktrontroler dan nilai PI digunakan sebagai referensi referensi pemanggilan nilai kontrol kecepatan motor DC untuk menyetabilkan posisi miniatur mobil listrik terhadap lintasan lengkung.
- Aktuator : motor DC
- Sensor : - Sensor *photodiode* sebagai pendeteksi terhadap lintasan.
  -Sinyal umpan balik dari *rotary encoder* berupa pembacaan putaran motor DC.
- *Output* (αm) : Kecepatan motor DC .

Miniatur mobil listrik ini dirancang untuk dapat berbelok secara diferensial sesuai dengan lintasan garis yang telah ditentukan. Pada miniatur mobil listrik ini menggunakan dua buah motor DC secara terpisah yang masing-masing mempunyai karakteristrik tersendiri, oleh karena itu tidak menutup kemungkinan perhitungan motor DC ini berbeda satu dengan yang lainya. Untuk memperoleh kecepatan yang sama pada masing-masing motor DC ini penulis menggunakan perhitungan karakteristik motor DC kemudian memasukan ke program dengan mengatur PWM sebagai masukan supaya mendapatkan keluaran yang diinginkan berupa kecepatan motor DC.

4.2. Prinsip Kerja Alat

Miniatur mobil listrik ini akan berjalan setelah diberi sumber tegangan, sistem dari alat ini akan bekerja setelah sensor *photodiode* yang mendeteksi garis hitam yang telah ditentukan jalurnya terlebih dahulu. Saat tombol *start* diaktifkan maka seluruh sistem akan aktif, miniatur mobil listrik ini menggunakan dua buah motor DC, sensor *photodiode* dan *rotary encoder*.

Sensor *photodiode* digunakan untuk membaca garis hitam dan mengirimkan sinyal input ke mikrokontroler ATmega16, kemudian

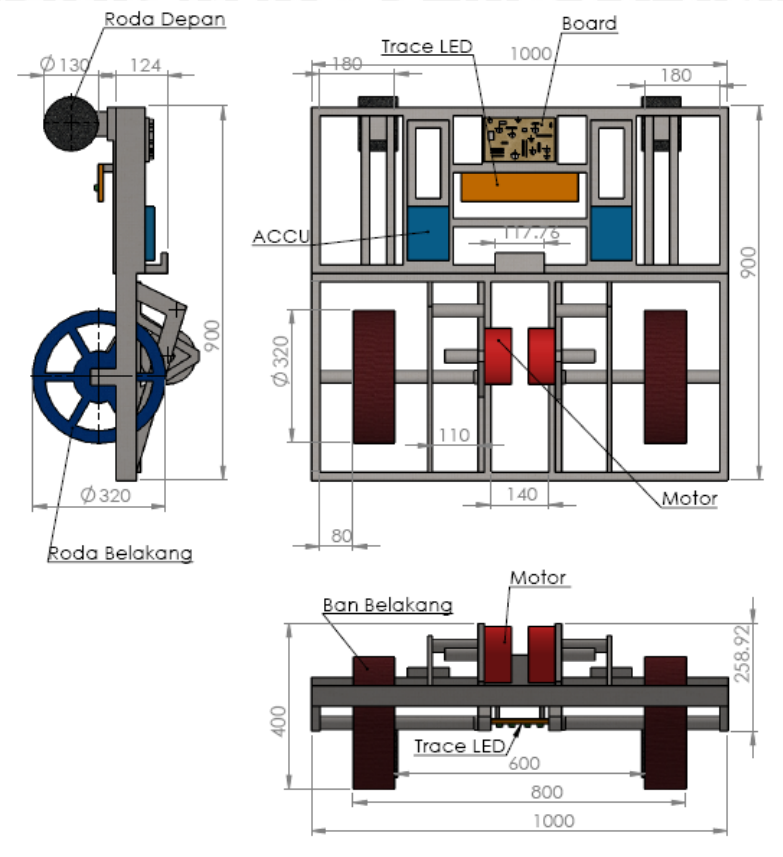
mikrokontroler ATmega16 akan memperoses dan mengirimkan sinyal ke dua buah motor DC untuk mengikuti garis yang ditentukan. *Rotary encoder* berfungsi sebagai pembaca kecepatan pada masing-masing motor DC dan mengirimkan sinyal intput ke mikrokontroler ATmega16.

4.3. Pemodelan Perangkat Keras

4.3.1. Mekanik Robot

1. Miniatur mobil listrik menggunakan rangka dengan spesifikasi secara keseluruhan yaitu panjang 100cm, lebar 80cm dan tinggi 45cm.
2. Pergerakan miniatur mobil listrik menggunakan dua buah motor DC.
3. Dua roda beban menggunakan roda bebas dan dua roda belakang yang dikopel dengan motorDC sebagai penggerak dengan sistem diferensial (terpisah).
4. Dua buah *Rotary Encoder* sebagai sensor kecepatan yang dipasang pada masing-masing motor DC.
5. Sensor *Photodiode* berfungsi sebagai sensor garis.
6. Mikrokontroler menggunakan ATmega16
7. LCD diletakan didepan miniatur mobil listrik yang berfungsi untuk menampilkan konstanta-konstanta yang akan ditentukan dalam sistem *tuning* kontrol PID.
8. Catu tegangan untuk motor DC menggunakan dua buah *accu* 12V.

Sistem mekanik yang baik, mendukung pergerakan robot menjadi lebih baik, oleh karena itu pembuatan mekanik dalam hal ini bodi dan rangka robot haruslah proporsional dengan panjang dan lebar serta tinggi dari robot. Dimensi robot ditunjukan pada gambar 4.2



Gambar 4.2 Mekanik Miniatur Mobil Listrik

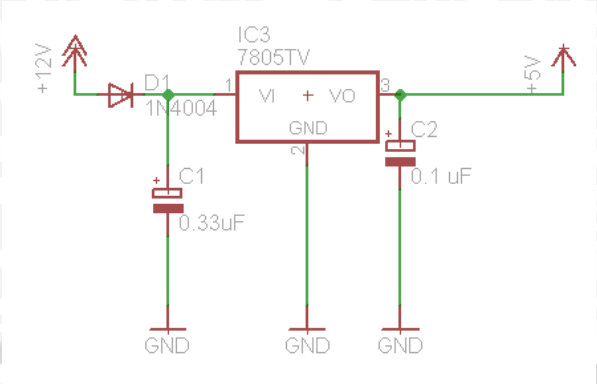
Keterangan gambar 4.2:

- Sensor diletakan di depan.
- Dua buah motor DC diletakan dibelakang.
- Baterai dan komponen elektrik diletakan di depan.
- Dua roda depan merupakan roda bebas.

4.3.2. Catu Daya Sistem

Miniatur mobil listrik ini menggunakan dua jenis catu daya. Yang pertama menggunakan dua buah *accu* 12V, kemudian yang kedua menggunakan catu daya 11V untuk rangkaian minimum sistem mikrokontroler ATmega16 bersumber dari baterai lipo (*lithium polimer*).

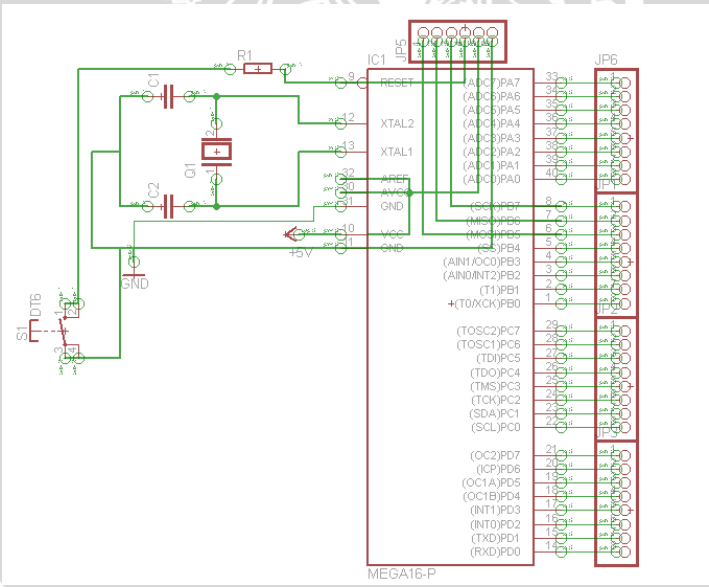
Pada pemodelan catu daya sistem digunakan catu daya sebesar 5V yang diperoleh dari rangkaian *Fixed Output Regulator* pada datasheet LM7805. IC78xx mempunyai tiga kaki, satu V_{in}, satu untuk V_{out} dan satu untuk GND (Blocher, 2003). Skema rangkaian catu daya ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Rangkaian Catu daya 5V

4.3.3. Minimum Sistem Mikrokontroler Atmega16

Robot ini menggunakan 1 mikrokontroler ATmega16. Fungsi dari mikrokontroler tersebut antara lain sebagai pengolah data sensor *photodiode*, pengatur arah dan kecepatan putaran motor kanandan pengatur arah dan kecepatan putaran motor kiri. Konfigurasi kaki I/O dari mikrokontroler(ATmega16) ditunjukkan dalam Gambar 4.4.



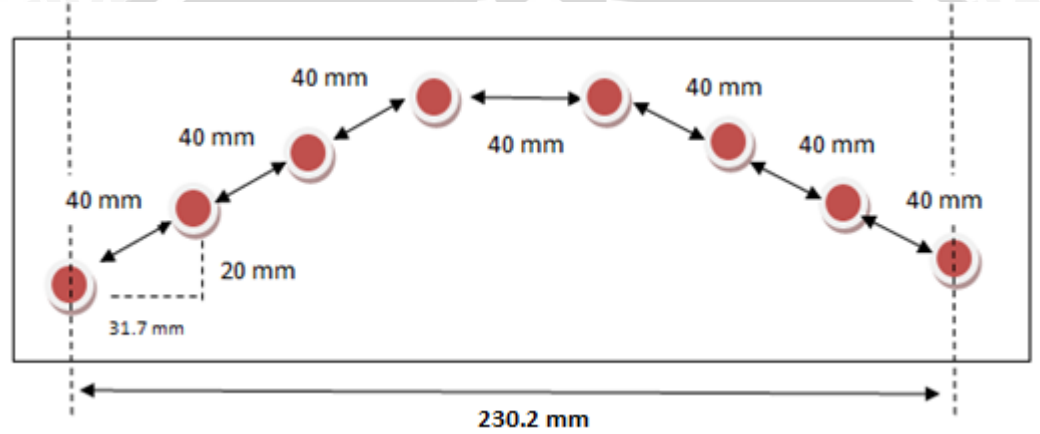
Gambar 4.4 Rangkaian Minimum Sistem Mikrokontroler Atmega16

Pin ADC mikrokontroler ATMega16 digunakan untuk membaca sensor *photodiode* melalui multiplekser analog. Pembacaan 8 sensor *photodiode* tersebut menggunakan metode *scanning/switching* dengan multiplekser analog tersebut. pemilihan sensor mana yang akan dibaca dapat diset melalui pin selector pada multiplekser. Seting inisialisasi ADC pada ATMega8 adalah:

- Resolusi yang digunakan yaitu 8-bit (nilai maksimal 255).
- Menggunakan Vref AVCC.
- Pin AVCC dihubungkan ke VCC yang tegangannya bernilai 5V.

4.3.4. Desain Sensor Garis

Pada miniatur mobil listrik ini menggunakan *photodiode* sebagai sensor garis. *Photodiode* yang digunakan sebagai sensor berjumlah delapan buah dan jarak antar sensor satu dengan yang lain ditentukan sebesar 3cm dengan desain penepatan posisi pada Gambar 4.5

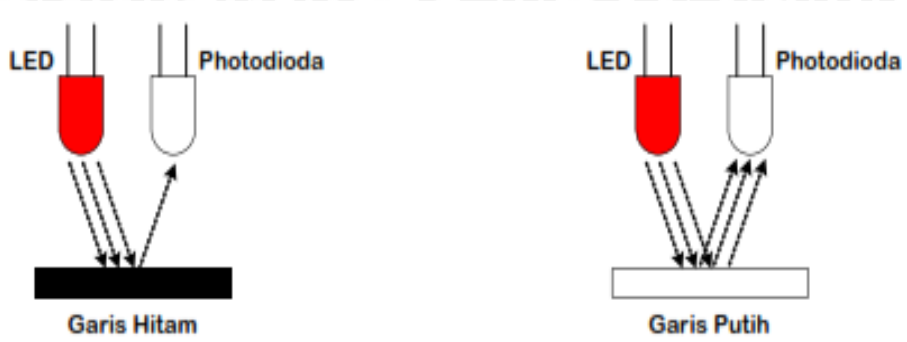


Gambar 4.5 Sensor Garis

Spesifikasi dari sensor *photodiode*, yaitu :

- Panjang sensor 230,2 mm.
- Mempunyai jarak antar sensor 40 mm.

Prinsip kerja sensor garis hanya memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya kita gunakan LED (Light Emiting Diode) yang akan memancarkan cahaya merah. Dan untuk menangkap pantulan cahaya LED, kita gunakan *photodiode*. Jika sensor berada diatas garis hitam maka *photodiode* akan menerima sedikit sekali cahaya pantulan. Tetapi jika sensor berada diatas garis putih maka *photodiode* akan menerima banyak cahaya pantulan. Ilustrasi prinsip kerja sensor garis ditunjukkan dalam Gambar 4.6



Gambar 4.6 Prinsip Kerja Sensor Garis

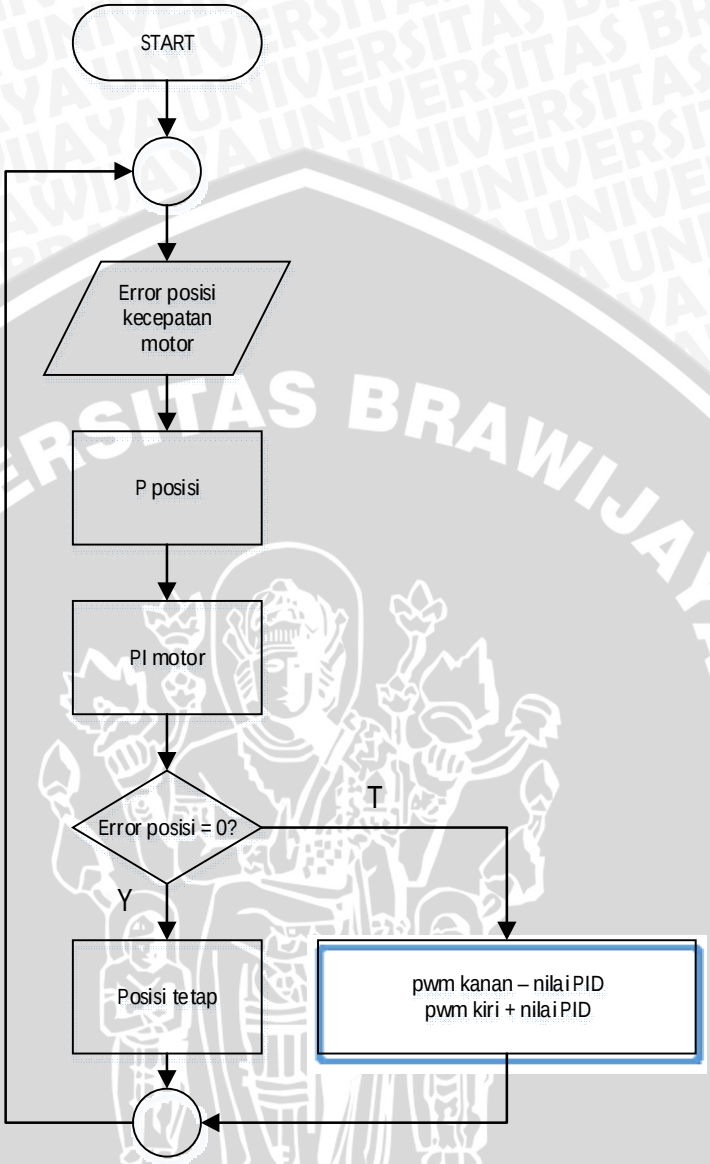
Sifat dari *photodiode* adalah jika semakin banyak cahaya yang diterima, maka nilai resistansi dioda semakin kecil. Dengan melakukan sedikit modifikasi, maka besaran resistansi tersebut dapat diubah menjadi tegangan. Sehingga jika sensor berada diatas garis hitam, maka tegangan keluaran sensor akan kecil, demikian pula sebaliknya.

4.3.5. Rotary Encoder

Dibutuhkan dua Rotary Encoder yang di*couple* langsung dengan masing-masing motor DC. *Rotary Encoder* berfungsi sebagai sensor kecepatan. Kemudian sinyal keluaran dari *Rotary Encoder* akan dikirim ke mikrokontroler sebagai data untuk mengetatur kecepatan Motor DC. *Rotary encoder* yang digunakan ditunjukkan dalam Gambar 4.7.

Gambar 4.7 Rotary Encoder

4.4. Pemodelan Perangkat lunak



4.5. Identifikasi Motor DC

Pada pengujian identifikasi driver motor DC menggunakan Matlab ini akan didapatkan fungsi alih. Fungsi alih sendiri digunakan untuk mengetahui karakteristik dari motor DC yang digunakan. Setelah melakukan pengujian rangkaian driver DC maka didapatkan data dari hasil pengujian tersebut. Data hasil pengujian kemudian disimulasikan menggunakan Matlab.

4.5.1. Menentukan Parameter Motor DC

Untuk mendapatkan fungsi alih dari motor DC diperlukan beberapa paramater data hasil pengujian. Data yang diperoleh dari pengujian fisik motor DC antara lain:

- $V = 2\ volt$
- $i = 0{,}25\ A$
- $\omega = 1{,}52\ rad/s$
- $R = 1{,}3\ \Omega$
- $L = 0{,}01\ H$
- $T_m = 0{,}52\ s$

Kemudian hasil dari pengujian fisik motor DC didapatkan beberapa parameter pada persamaan (2.8) sampai (2.10) sebagai berikut:

nilai dari *back - emf* didapatkan menggunakan persamaan (2.8) dengan memasukkan nilai parameter diatas dan menghasilkan nilai sebagai berikut :

$$e = V - i \times R$$

$$e = 2 - 0{,}25 \times 1{,}3 = 1{,}675\ volt$$

Nilai dari konstanta motor dapat diperoleh dari persamaan (2.9), setelah memasukan nilai parameter yang diperoleh diatas akan didapatkan:

$$K = e : \omega = 1{,}102$$

$$K = 1{,}675 : 1{,}52 = 1{,}102$$

Nilai koofisien *friction* dapat diperoleh dari persamaan (2.10), setelah memasukan nilai parameter yang diperoleh diatas akan didapatkan:

$$b = K \times i : \omega = 0{,}181$$

b = 1,102 × 0,25 : 1,25 = 0,181

Dari nilai-nilai diatas akan didapatkan nilai momen inersia yang dihitung sebagai berikut ini:

$J = b \times Tm$ = 0,0941

J = 0,181 × 0,52 = 0,0941

4.5.2. Menentukan Fungsi Alih Motor DC

Setelah mendapatkan parameter motor DC, kemudian akan ditentukan fungsi alih dengan cara memasukan ke dalam simulasi menggunakan Matlab. Dengan memasukan paramter motor DC kedalam program Matlab seperti Gambar 4.9

```
V=2
i=0.25
w=1.52
R=1.3
L=0.01
Tm=0.52

E=V-i*R
K=E/w
B=K*i/w
J=B*Tm

s=tf('s')
P=K/((J*s+B)*(L*s+R)+K^2)
step(P)

pole(P)
```

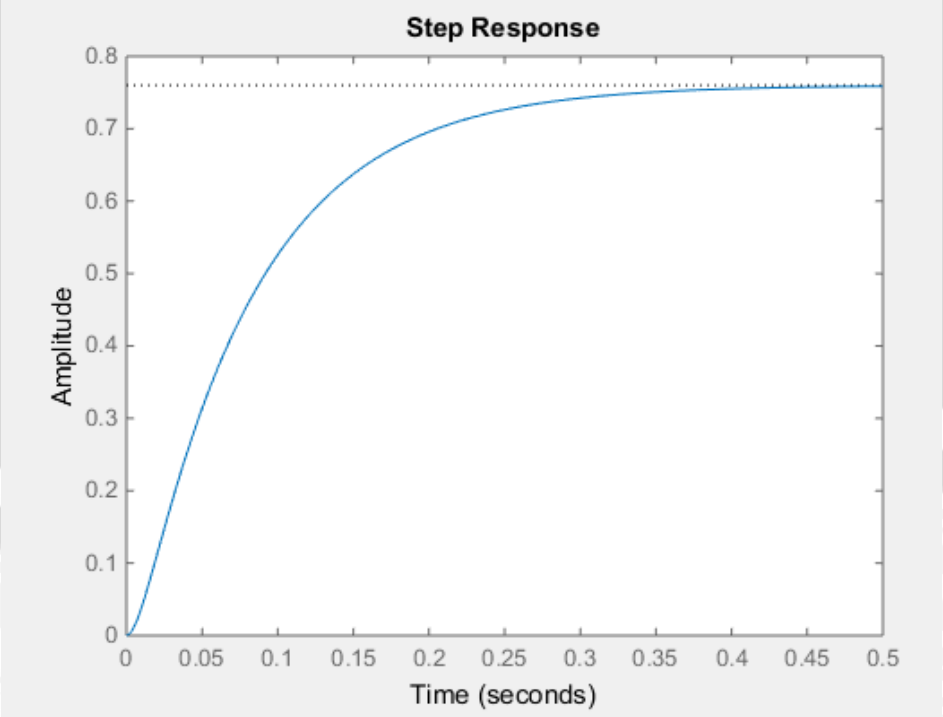
Gambar 4.9 Program parameter motor DC pada Matlab

Kemudian pada Matlab akan mensimulasikan untuk mendapatakan fungsi alih dari hasil paramater motor DC dan menunjukan pole dari fungsi alih. Hasil dari fungsi alih dan pole ditunjukan pada gambar 4.10. Setelah itu akan di plot grafik respon dari karakteristik motor DC yamg ditunjukan pada Gambar 4.11.

```
P =

                1.102
  -------------------------------
  0.0009425 s^2 + 0.1243 s + 1.45

Continuous-time transfer function.


ans =

 -118.9942
  -12.9289
```

Gambar 4.10 Fungsi alih dan pole motor DC

Setelah mendapatkan fungsi alih kemudian diberi masukan step pada simulasi Matlab bisa dilihat grafik respon dari motor DC *time constan* dari motor 0,25 detik pada gambar 4.11



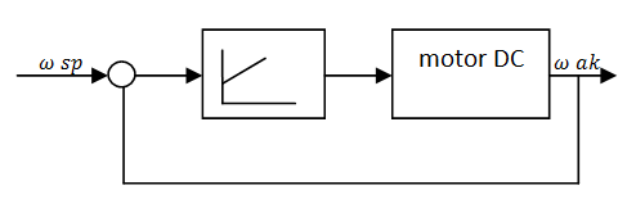
Gambar 4.11 Respon motor DC dari hasil simulasi Matlab

4.5.3. Perancangan Kontroler PI Direct

Perancangan kontroler PI untuk kecepatan motor DC, Sehingga sistem dapat bekerja dengan yang diharapkan. Untuk mencari nilai dari PI, menggunakan fungsi alih dari motor.

4.5.3.1. Kontrol PI

Yang pertama dicari kontroler PI dengan menggunakan metode kontroler *direct.*



Gambar 5.12 Blok diagram loop dalam

Berikut ini adalah perhitungan direct dari fungsi alih untuk mencari parameter kontroler PI sebagai berikut:

Setelah melakukan pengujian fisik dari motor maka didapatkan fungsi alih seperti persamaan 4.11

$$\frac{\omega}{V_a} = \frac{1{,}102}{0{,}00232s^2 + 0{,}3034s + 1{,}45} \quad \ldots\ldots (4.1)$$

Persamaan fungsi alih motor kedalam persamaan 4.2

$$\frac{V_R}{(T_1 s + 1)(T_2 s + 1)} \quad \ldots\ldots (4.2)$$

$$\frac{\omega}{V_a} = \frac{1{,}102}{(s + 118{,}9942)(s + 12{,}9289)}$$

$$\frac{1{,}102}{(118{,}9942)(12{,}9289)\left[\frac{1}{118{,}9942}s + 1\right]\left[\frac{1}{12{,}9289}s + 1\right]}$$

Maka didapatkan nilai V_R T_1 dan T_2 sebagai berikut;

$$V_R = \frac{1{,}102}{(118{,}9942\, s + 1)(12{,}9289\, s + 1)}$$

Nilai T_1 dan T_2 masing-masing didapatkan dari koefisien s. Nilai T_1 kemudian diasumsikan menjadi T_i.

$T_i = 0{,}0084$ s

$T_2 = 0{,}077\ s$

diasumsikan nilai time integral untuk kontroler PI (T_i) sama dengan T_1

$$\frac{K_i}{s} = \frac{1}{\left(\frac{1}{118{,}9942}s + 1\right) K_i} \quad \text{(4.3)}$$

Dari presamaan 4.14 maka didapatkan fungsi close loop dalam:

$$F_{cl} = \frac{K_i}{s(0{,}0084\ s + 1) + K_i} \quad \text{(4.4)}$$

$$= \frac{K_i}{0{,}0084\ s^2 + s + K_i}$$

Fungsi close loop dalam yang sudah diketahui, kemudian digunakan untuk mencari nilai α dan β dengan membagi penyebut dengan 0,0084. Setelah dibagi, didapatkan perhitungan:

$$\frac{K_i / 0{,}0084}{s^2 + \frac{1}{0{,}0084}s + \frac{K_i}{0{,}0084}} \quad \text{(4.5)}$$

Dengan persamaan fungsi close loop tersebut Maka nilai α dan β dapat diperoleh:

$$2\alpha = \frac{1}{0{,}0084} \rightarrow \alpha = \frac{1}{0{,}0168} \quad \text{(4.6)}$$

$$\beta = \sqrt{\frac{K_i}{0{,}0084}} \quad \text{(4.7)}$$

Nilai α dan β kemudian digunakan untuk mendapatkan nilai D:

$$D = \frac{\alpha}{\beta} = \frac{\frac{1}{0{,}0168}}{\sqrt{\frac{K_i}{0{,}0084}}} \quad \text{(4.8)}$$

nilai D diasumsikan menjadi 1, sehingga perhitungan menjadi berikut:

$$1 = \frac{59{,}5}{\sqrt{\frac{K_i}{0{,}0084}}}$$

Dari perhitungan dengan mengasumsikan nilai D menjadi 1, maka nilai $K_i$dapat diperoleh:

$$\frac{K_i}{0{,}0084} = (59{,}5)^2$$

$K_i = 29{,}73$

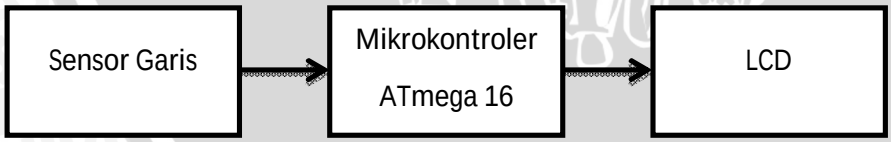
BAB V
PENGUJIAN DAN ANALISIS

Tujuan pengujian sistem ini adalah untuk menentukan apakah alat yang telah dibuat berfungsi dengan baik dan sesuai dengan perancangan. Pengujian pada sistem ini meliputi pengujian setiap blok maupun pengujian secara keseluruhan. Pengujian setiap blok ini dilakukan untuk menemukan letak kesalahan dan mempermudah analisis pada sistem apabila alat tidak bekerja sesuai dengan perancangan. Pengujian dibagi menjadi beberapa bagian, yaitu:

1. Pengujian data sensor garis.
2. Pengujian kecepatan putaran motor DC.
3. Simulasi MatLab.
4. Pengujian keseluruhan sistem meliputi pengujian miniatur mobil listrik terhadap lintasan lurus dan lintasan lengkung.

5.1. Pengujian Data Sensor Garis

Pengujian ini dilakukan dengan tujuan untuk mengetahui perbedaan warna dari lintasan miniatur mobil listrik. Pada lintasan berwarna putih dan garis lintasan miniatur mobil listrik berwarna hitam. Sehingga miniatur mobil listrik dapat mengikuti garis hitam yang telah ditentukan terlebih dahulu. Langkah-langkah pengujian dilakukan dengan menghubungkan sensor garis, mikrokontroler utama ATmega 16, dan LCD yang sesuai pada Gambar 5.1.



***Gambar 5.1** Diagram blok pengujian data sensor garis*

5.1.1. Peralatan pengujian terdiri dari :

1. Laptop
2. Mikrokontroler ATmega 16
3. Sensor sensor garis

5.1.2. Prosedur Pengujian Sensor Garis

Setelah dilakukan pengujian terhadap sensor garis didapattkan nilai hasil berupa keluaran dari sensor *photodiode.*Untuk pembacaan dari warna hitam di asumsikan menjadi logika tinggi dan untuk pembacaan dari warna putih berlogika rendah.

5.1.3. Hasil Pengujian Sensor Garis

Untuk inialisai ADC dari sensor garis maka dibutuhkan pengujian dari nilai ADC sensor. Sehingga dapat dibedakan untuk nilai tinggi sebagai warna hitam dan nilai yang rendah untuk warna putih.

Tabel 5.1. Tabel hasil pengujian sensor warna putih

Sensor	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10	Rata-Rata
1	226	211	216	215	207	214	221	214	222	221	215
2	186	219	182	154	150	176	183	163	174	212	178,2
3	195	196	195	195	223	191	192	200	177	178	200,8
4	226	230	192	223	224	233	225	216	218	200	219,8
5	221	218	235	238	200	219	211	207	208	217	221,8
6	226	235	217	222	195	220	213	216	226	212	219
7	220	226	210	224	232	227	213	221	218	215	222,4
8	211	203	225	214	224	235	225	220	210	204	215,4

Pada tabel 5.1 hasil pengujian sensor dengan warna putih nilai tertinggi dari pembacaan sensor adalah 106 pada sensor ke-8 hasil percobaan kedua. Maka dalam penentuan logika rendah adalah dengan nilai tertinggi dari pembacaan sensor.

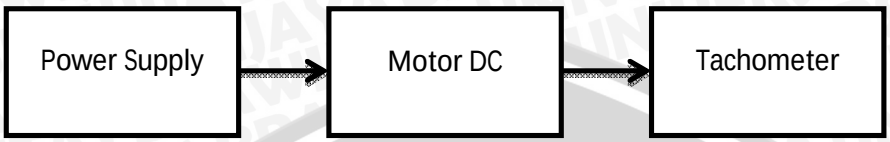
Tabel 5.2. Tabel hasil pengujian sensor warna hitam

Sensor	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10	Rata-Rata
1	226	211	216	215	207	214	221	214	222	221	215
2	186	219	182	154	150	176	183	163	174	212	178,2
3	195	196	195	195	223	191	192	200	177	178	200,8
4	226	230	192	223	224	233	225	216	218	200	219,8
5	221	218	235	238	200	219	211	207	208	217	221,8
6	226	235	217	222	195	220	213	216	226	212	219
7	220	226	210	224	232	227	213	221	218	215	222,4
8	211	203	225	214	224	235	225	220	210	204	215,4

Pada tabel 5.2 hasil pengujian sensor dengan warna hitam nilai terendah dari pembacaan sensor adalah 150 pada sensor ke-2 hasil percobaan kelima. Maka dalam penentuan logika rendah adalah dengan nilai terendah dari pembacaan sensor.

5.2. Pengujian Kecepetan Motor DC

Pada pengujian motor DC bertujuan untuk mengetahui hubungan antara perubahan tegangan terhadap kecepatan motor. Langkah-langkah percobaan pengujian dilakukan seperti pada diagram blok pengujian kecepatan motor DC yang ditunjukan seperti pada gambar 5.3



Gambar 5.3 Diagram blok pengujian kecepatan motor

5.2.1. Peralatan Yang Digunakan

1. Power supply
2. Motor DC
3. Tachometer

5.2.2. Prosedur Pengujian Kecepetan Motor DC

Peosedur pengujian dilakukan dengan menghubungkan motor DC dengan power supply, kemudian poros motor DC dihubungkan dengan tachometer, setelah itu tegangan dari power supply diubah-ubah mulai 0-24 V dengan kenaikan 1 V, catat hasil setiap perubahan kecepatan motor yang ditunjukan di tachometer.

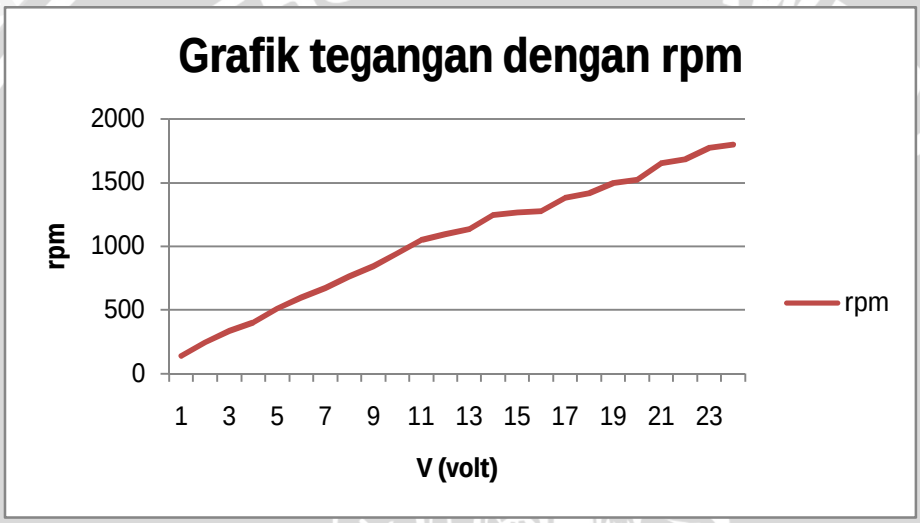
5.2.3. Hasil Pengujian Kecepetan Motor DC

Hasil pengujian kecepatan motor terhadap tegangan tertentu yang diperoleh melalui beberapa kali percobaan ditunjukan pada tabel 5.2

Tabel 5.2 Data tegangan dengan kecepatan dan arus motor

Tegangan	Motor 1		Motor 2	
	Kecepatan	Arus	Kecepatan	Arus
(V)	(RPM)	(A)	(RPM)	(A)
1	142	0,11	145	0,11
2	247	0,19	250	0,19
3	338	0,21	340	0,21
4	402	0,22	412	0,22
5	512	0,23	512	0,23
6	598	0,25	598	0,25
7	672	0,26	672	0,26
8	762	0,27	762	0,27
9	846	0,28	846	0,28
10	946	0,29	946	0,29
11	1048	0,3	1048	0,3

12	1097	0,3	1097	0,3
13	1138	0,31	1138	0,31
14	1247	0,32	1247	0,32
15	1268	0,34	1268	0,34
16	1278	0,35	1278	0,35
17	1380	0,36	1380	0,36
18	1418	0,37	1418	0,37
19	1497	1,38	1497	1,38
20	1522	0,38	1522	0,38
21	1652	0,39	1652	0,39
22	1682	0,4	1682	0,4
23	1772	0,41	1772	0,41
24	1798	0,42	1798	0,42



Gambar 5.4 Grafik hubungan antara tegangan dengan kecepaatan

5.3. Simulasi Matlab

Pada simulasi menggunakan *software* Matlab ini bertujuan untuk menerapkan algoritma yang telah ditetapkan sebelumnya dan untuk mengetahui performasi yang ideal secara simulasi dan dapat dijadikan tolok ukur dengan pengujian secara keseluruhan yang akan dilakukan selanjutnya.

5.3.1. Peralatan Yang Digunakan

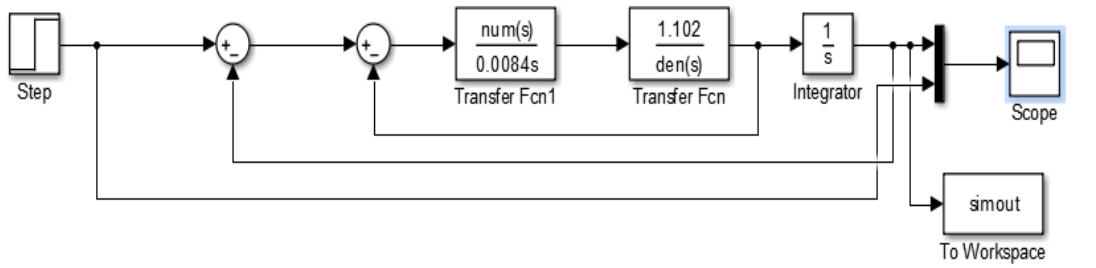
1. program Matlab dengan *toolbox simulink.*

5.3.2. Prosedur Percobaan

1. Menyusun diagram blok pada Matlab.
2. Memasukan nilai kontroler P dan PI .

3. Memasukan fungsi alih motor DC yang telah didapatkan.
4. Menampilkan hasil dari simulasi melalui scope.
5. Menggunakan sinyal masukan step.

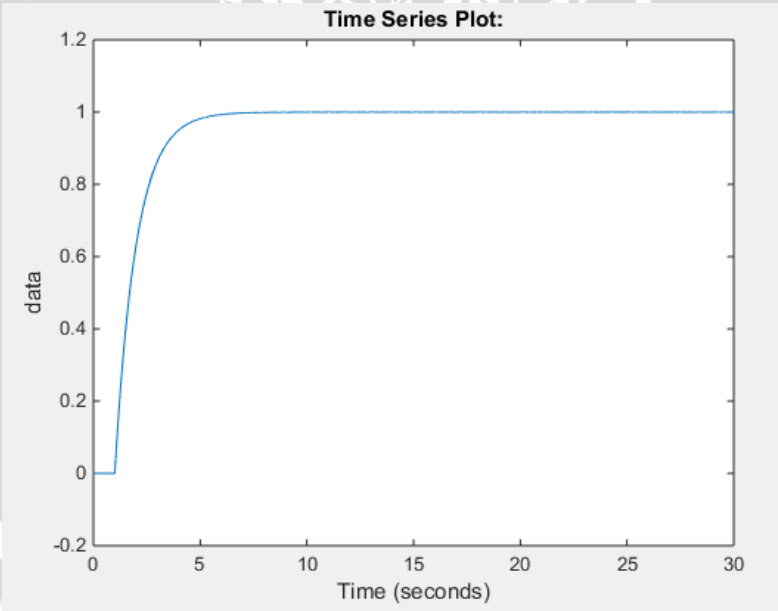
Setelah melakuan prosedur diatas maka akan didapatkan gambar seperti yang ditunjukan gambar 5.4



Gambar 5.5 Diagram blok simulasi Matlab

5.3.3. Hasil Simulasi Matlab

Hasil simulasi sistem menggunakan Matlab ditunjukan pada gambar 5.5



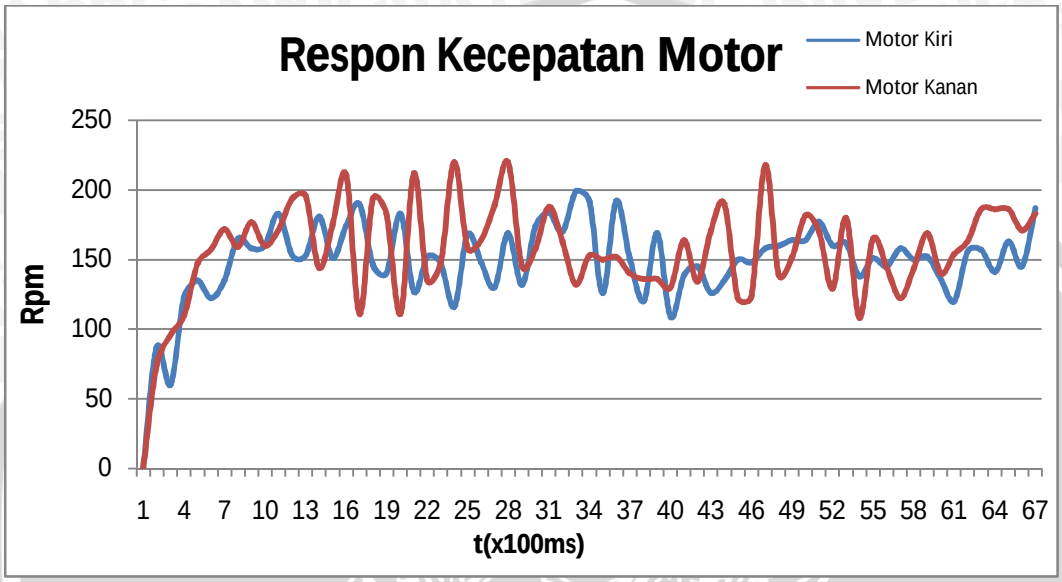
Gambar 5.5 Grafik simulasi respon sistem menggunakan Matlab

5.4. Pengujian Keseluruhan Sistem

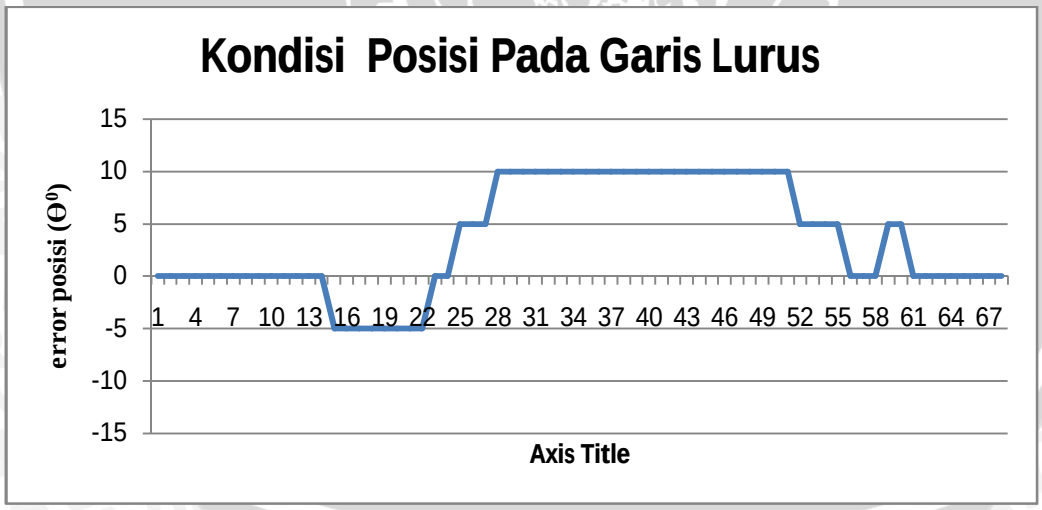
Pada pengujian keseluruhan sistem ini, miniatur mobil listrik diletakan di lintasan yang telah ditentukan. Ada dua lintasan yang digunakan yaitu yang pertama lintasan lurus dengan pancang 300cm dan lintasan lengkung dengan panjang 500cm.

Prosedur pengujian dengan meletakan miniatur mobil listrik pada posisi awal dalam lintasan dan mengaktifkanya agar bergerak mengikuti lintasan. Miniatur mobil listrik akan berhenti dengan sendirinya setelah melewati ujung garis.

5.4.1 Pengujian Keseluruhan Pada Lintasan Lurus



Gambar 5.6 Grafik kecepatan motor pada lintasan lurus

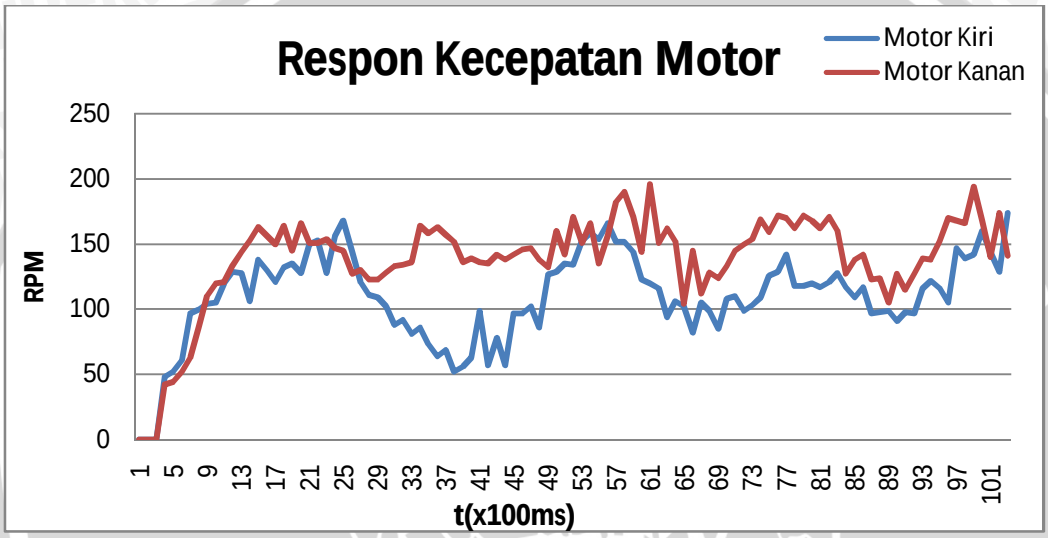


Gambar 5.7 Grafik posisi pada lintasan lurus

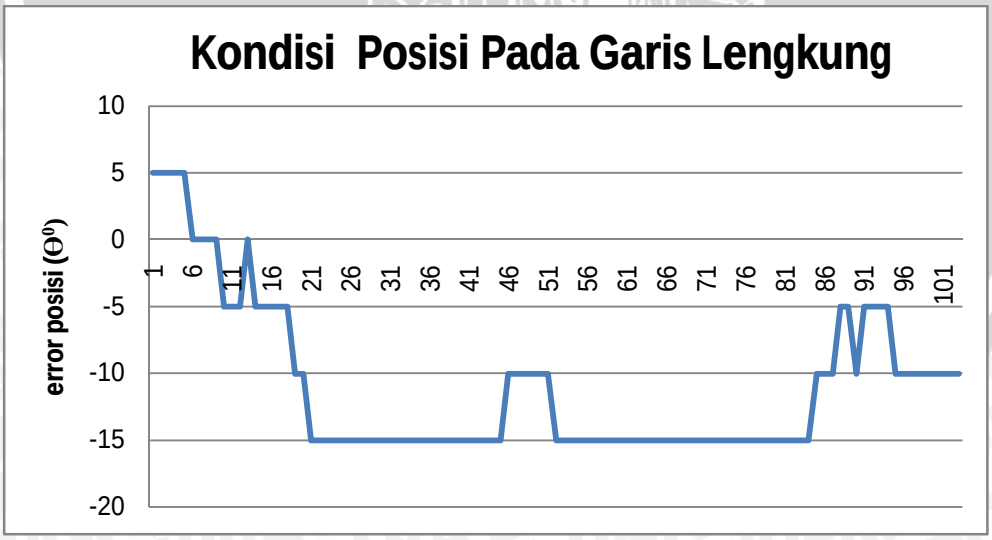
Set point untuk posisi alat adalah pada 0^o dengan kecepatan motor sebesar 180 Rpm. Pada Gambar 5.6 menunjukkan perubahan kecepatan masing-masing motor. Pada saat t kurang dari 130ms alat mampu berada pada set point(ditunjukkan pada Gambar 5.7). Apabila posisi bernilai minus maka berarti alat menghadap ke kanan, sehingga untuk memposisikan

lagi menuju setpoint maka kecepatan motor kanan harus lebih cepat dari motor kiri, maka karena hal tersebut kecepatan motor kanan akan lebih besar dari motor kiri (ditunjukkan pada gambar 5.6 saat t=1400ms hingga t=2200ms). pada t=2400ms hingga t=5400ms posisi alat menghadap ke kiri, sehingga untuk ke posisi tengah kecepatan motor kiri harus lebih besar dari motor kanan, saat t yang sama pada gambar 5.6 kecepatan motor kiri lebih besar dari motor kanan. demikin pula seterusnya hingga posisi alat berada pada 0^o lagi kecepatan motor dan motor kiri akan menjadi sama atau hampir sama.

5.4.2 Pengujian Keseluruhan Pada Lintasan Lengkung



Gambar 5.8 Grafik kecepatan motor pada lintasan lengkung



Gambar 5.9 Grafik posisi pada lintasan lengkung

Setpoint untuk posisi alat adalah pada 0^0 dengan *setpoint* motor sebesar 180 rpm. Pada Gambar 5.8 menunjukkan perubahan kecepatan masing-masing motor. Pada saat t= 0 ms sampai kurang dari 600 ms motor sebelah kiri kecepatannya lebih besar dibanding motor sebelah kanan, agar motor tetap berada pada *setpoint* maka sebelah motor kanan dipercepat dibanding motor sebelah kiri (ditunjukkan pada gambar 5.9. t < 600 ms sampai dengan t= 600 ms) motor cenderung kembali pada *setpoint*, karena bahwa garis lengkung arahnya ke kanan ketika t= 1300 ms hingga t= 1600 ms (seperti yang ditunjukkan pada gambar 5.8), dikarenakan garis lintasan cenderung kekanan maka motor sebelah kanan cenderung lebih cepat dibanding motor sebelah kiri seperti yang ditunjukkan pada gambar 5.8 pada t =2700 ms hingga t= 10.100 ms.

BAB VI
KESIMPULAN

6.1 Kesimpulan

Setelah dilakukan beberapa pengujian dari sistem, serta pengujian keseluruhan sistem maka penulis dapat menyimpulkan bahwa dapat disimpulkan bahwa hasil pengendalian kecepatan motor DC pada garis lengkung cenderung mengarah kekanan dibandingkan pada lintasan garis lurus. miniatur mobil listrik mampu memposisikan untuk kembali ke *setpoint* (posisi tengah). Dengan nilai Kp= 0,0084 ,Ki= 29,73 dan *setpoint* kecepatan motor 180 rpm mampu kembali ke *setpoin* dalam waktu 7s.

DAFTAR PUSTAKA

Akbar, Arnas Elmiawan. 2013. *Implementasi Sistem Navigasi Wall Following Menggunakan Kontroler PID dengan Metode Tuning pada Robot Kontes Robot Cerdas Indonesia (KRCI) Divisi Senior Beroda*. Malang: Skripsi Jurusan Teknik Elektro FT-UB.

Andrianto, Heri. 2008. *Pemrograman Mikrokontroler AVR Atmega16*. Bandung: Informatika.

Atmel. 2007b. *8-bit AVR with 8K Bytes In-System Programmable Flash Atmega8, Atmega8L*. San Jose: Atmel.

Ogata, Katsuhiko. 1997. *Teknik Kontrol Automatik (Sistem Pengaturan)*. Jakarta: Erlangga.

.Soebhakti, Hendawan. 2007. *AVR Application Note*. Jakarta: Erlangga.

Winoto, Ardi. 2008. *Mikrokontroller AVR Atmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR*. Bandung: Informatika.

Cadangan Minyak dunia (http://www.esdm.go.id/berita/migas/40-migas/6487-cadangan-minyak-kita-cuma-1100-venezuela.htm)

Cadangan minyak dunia perkembangan mobil listrik. 2014. (http://www.kemenperin.go.id /artikel/4260/Mulai-2014,-Proyek-Mobil-Listrik-Diproduksi-Massal). Diakses tanggal 12 desember 2014

LAMPIRAN I

Foto Alat

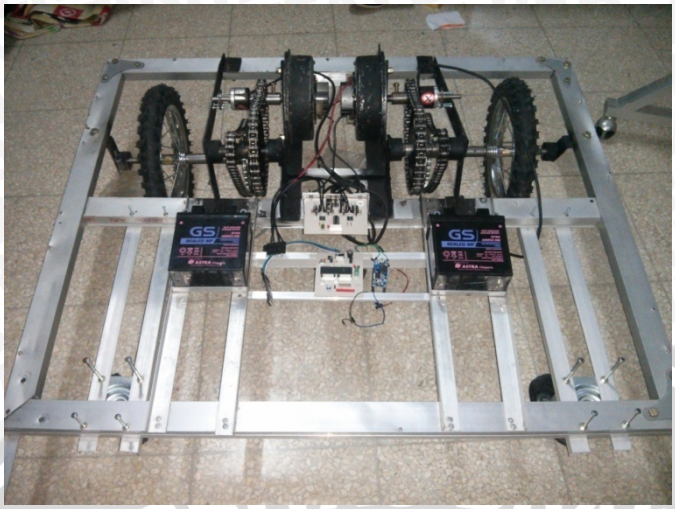


Foto proses pembuatan miniatur mobil listrik

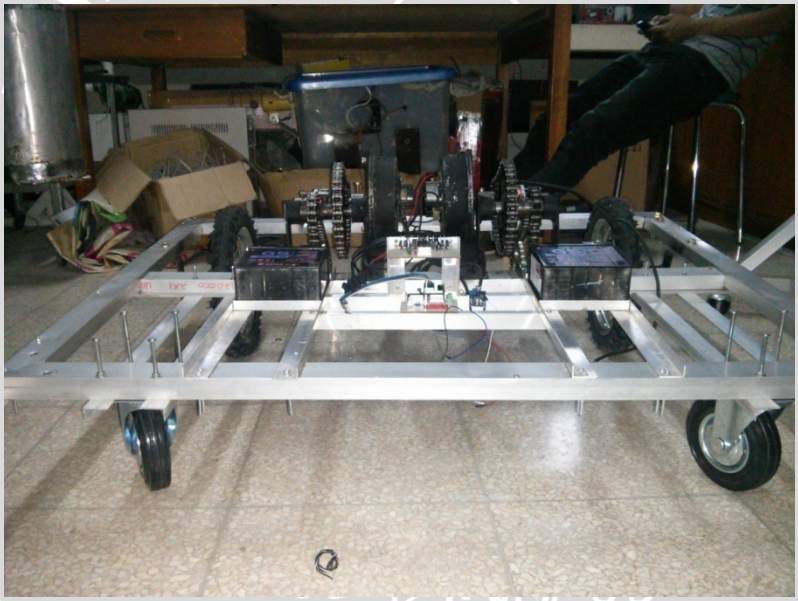


Foto mekanik miniatur mobil listrik

LAMPIRAN II

Data Sheet

EMS 30A H-Bridge

EMS (Embedded Module Series) 30 A H-Bridge merupakan driver H-Bridge berbasis VNH3SP30 yang didisain untuk menghasilkan drive 2 arah dengan arus kontinu hingga 30 A pada tegangan 5,5 Volt sampai 36 Volt (IC VNH2SP30 hanya sampai 16V). Dilengkapi dengan rangkaian sensor arus beban yang dapat digunakan sebagai umpan balik ke pengendali. Modul ini dapat men-drive beban-beban induktif seperti misalnya motor DC, motor stepper, koil relay, selenoida, dan beban-beban lainnya.

Spesifikasi :

- Catu daya : 5VDC.
- Terdiri dari 1 driver full H-Bridge. (IC VNH2SP30 memiliki fitur current sense).
- Driver mampu melewatkan arus kontinu 30 A dan mendukung tegangan beban dari 5,5 Vsampai 36 V (IC VNH2SP30 hanya sampai 16V).
- Input kompatibel dengan level tegangan TTL dan CMOS.
- MOSFET output dengan resistansi drain-source rendah (typ. 0,034 ohm).
- Mendukung kontrol PWM dengan frekuensi sampai 20 Khz.
- Proteksi hubungan singkat dan proteksi overtemperature.
- Undervoltage and overvoltage shutdown.
- Reverse battery protection.
- Jalur catu daya input (logika) terpisah dari jalur catu daya untuk beban.
- Desain PCB standar industri dengan bahan 2 layer FR4 dan plated through hole (PTH).
- Kompatibel dengan SPC Gamepad Interface serta mendukung sistem mikrokontroler.

E40 Series

Diameter ø40mm Shaft type/Hollow type/Built-in type Incremental Rotary Encoder

■ Features

- **12-24VDC power supply of line driver output(Line-up)**
- Easy installation at narrow space
- Small moment of inertia
- Power supply : 5VDC, 12-24VDC ±5%
- Various output types

⚠ Please read "Caution for your safety" in operation manual before using.

CE

■ Ordering information

E40 H — 8 — 5000 — 3 — N — 24 — ☐

Series	Shaft type	Hollow type	Pulse/1Revolution	Output phase	Output	Power supply	Cable
S: Shaft type H: Hollow type HB: Hollow built-in type	External 6: ø6mm 8: ø8mm	Inner 6: ø6mm 8: ø8mm 10: ø10mm 12: ø12mm	Series	2: A, B 3: A, B, Z 4: A, $\overline{A}$, B, $\overline{B}$ 6: A, $\overline{A}$, B, $\overline{B}$, Z, $\overline{Z}$	T: Totem pole output N: NPN open collector output V: Voltage output L: Line driver output	5 : 5VDC ±5% 24: 12-24VDC ±5%	No mark: Cable type C: Connector cable type(※)

※Standard: E40S6-PULSE-3-N-24, E40H8-PULSE-3-N-24
E40HB8-PULSE-3-N-24

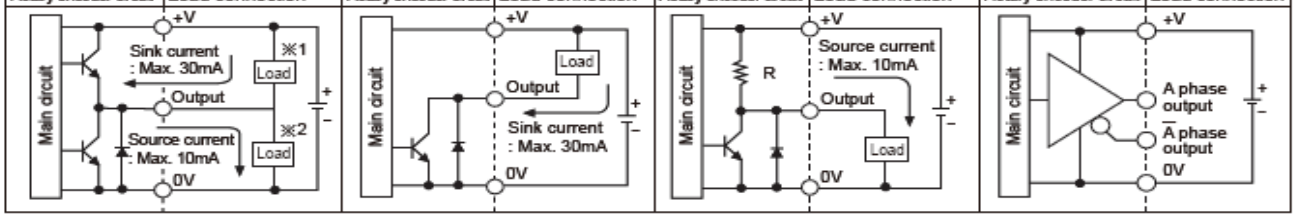
※Standard: A, B, Z

※Cable length : 250mm

Incremental ø40mm Shaft/Hollow Shaft/Built-in type

▣ Control output diagram

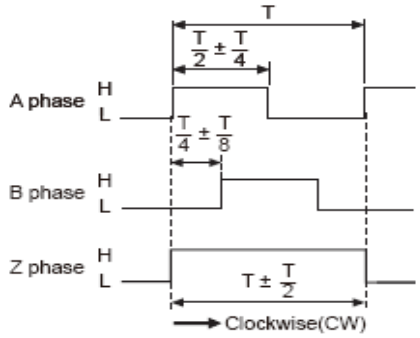
Totem pole output		NPN open collector output		Voltage output		Line driver output	
Rotary encoder circuit	Load connection	Rotary encoder circuit	Load connection	Rotary encoder circuit	Load connection	Rotary encoder circuit	Load connection



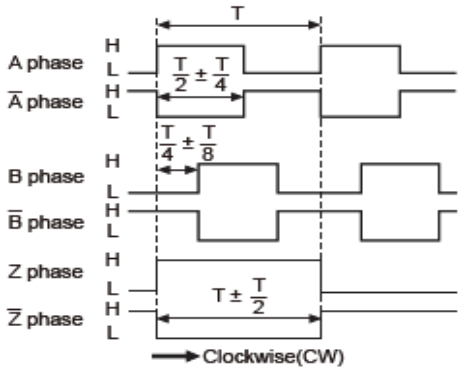
- Totem pole output type can be used for NPN open collector output type(※1) or Voltage output type(※2).
- All output circuits of A, B, Z phase are the same. (Line driver output is A, Ā, B, B̄, Z, Z̄)

▣ Output waveform

• Totem pole output / NPN open collector output / Voltage output



• Line driver output



※Z phase output is option.
※CW : Right turn as from the shaft.

▣ Specifications

Item			Diameter ø40mm shaft/hollow shaft/hollow built-in type of incremental rotary encoder	
Resolution(P/R)※1			⋆1, ⋆2 ,⋆5, 10, ⋆12, 15, 20, 23, 25, 30, 35, 40, 45, 50, 60, 75, 100, 120, 150, 192, 200, 240, 250, 256, 300, 360, 400, 500, 512, 600, 800, 1000, 1024, 1200, 1500, 1800, 2000, 2048, 2500, 3000, 3600, 5000 (Not indicated resolution is customizable.)	
Electrical specification	Output phase		A, B, Z phase(Line driver A, $\overline{A}$, B, $\overline{B}$, Z, $\overline{Z}$ phase)	
	Phase difference of output		Phase difference between A and B : $\frac{T}{4} \pm \frac{T}{8}$ (T=1cycle of A phase)	
	Control output	Totem pole output	• Low - Load current:Max. 30mA, Residual voltage : Max. 0.4VDC • High - Load current:Max. 10mA, Output voltage(Power voltage 5VDC): Min. (Power voltage-2.0)VDC, Output voltage(Power voltage 12-24VDC):Min. (Power voltage-3.0)VDC	
		NPN open collector output	Load current : Max. 30mA, Residual voltage : Max. 0.4VDC	
		Voltage output	Load current : Max. 10mA, Residual voltage : Max. 0.4VDC	
		Line driver output	• Low - Load current : Max. 20mA, Residual voltage : Max. 0.5VDC • High - Load current : -20mA, Output voltage(Power voltage 5VDC): Min. 2.5VDC, Output voltage(Power voltage 12-24VDC): Min. (Power voltage-3.0)VDC	
	Response time (Rise/Fall)	Totem pole output	Max. 1μs	• Measuring condition - Cable length : 2m, I sink = 20mA
		NPN open collector output		
		Voltage output		
		Line driver output	Max. 0.5μs	
	Max. Response frequency		300kHz	
	Power supply		• 5VDC ±5%(Ripple P-P : Max. 5%) • 12-24VDC ±5%(Ripple P-P : Max. 5%)	
	Current consumption		Max. 80mA(disconnection of the load), Line driver output : Max. 50mA(disconnection of the load)	
	Insulation resistance		Min. 100MΩ(at 500VDC megger between all terminals and case)	
	Dielectric strength		750VAC 50/60Hz for 1 minute(Between all terminals and case)	
	Connection		Cable type, 250mm connector cable type	
Mechanical specification	Starting torque		Shaft type : Max. 40gf·cm(0.004N·m), Hole type : Max. 50gf·cm(0.005N·m)	
	Moment of inertia		Max. 40g·cm²(4×10^{-6} kg·m²)	
	Shaft loading		Radial : Max. 2kgf, Thrust : Max. 1kgf	
	Max. allowable revolution ※2		5000rpm	
Vibration			1.5mm amplitude or 300m/s² at frequency of 10 to 55Hz(for 1 min.) in each of X, Y, Z directions for 2 hours	
Shock			Approx. Max. 50G	
Environment	Ambient temperature		-10 to 70℃, storage : -25 to 85℃	
	Ambient humidity		35 to 85%RH, storage : 35 to 90%RH	
Protection			IP50(IEC standard) ※Option type is available for IP64 (IEC standard).	
Cable			ø5, 5-wire, Length : 2m, Shield cable(Line driver output : ø5, 8-wire) (AWG24, Core diameter: 0.08, Number of cores: 40, Insulator out diameter: ø1)	
Accessory			• Shaft type :ø6mm coupling standard, ø8mm coupling(Sold separately) • Hole type : Bracket	
Approval			CE (Except line driver output)	
Unit weight			Approx. 160g	

※1: 'ㅅ' pulse is only for A, B phase(Line Driver output is for A, $\overline{A}$, B, $\overline{B}$ phase)

※2: Make sure that. Max response revolution should be lower than or equal to max. allowable revolution when selecting the resolution.

[Max. response revolution(rpm)= $\frac{\text{Max. response frequency}}{\text{Resolution}}$ x 60 sec] ※Environment resistance is rated at no freezing or condensation.

Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL

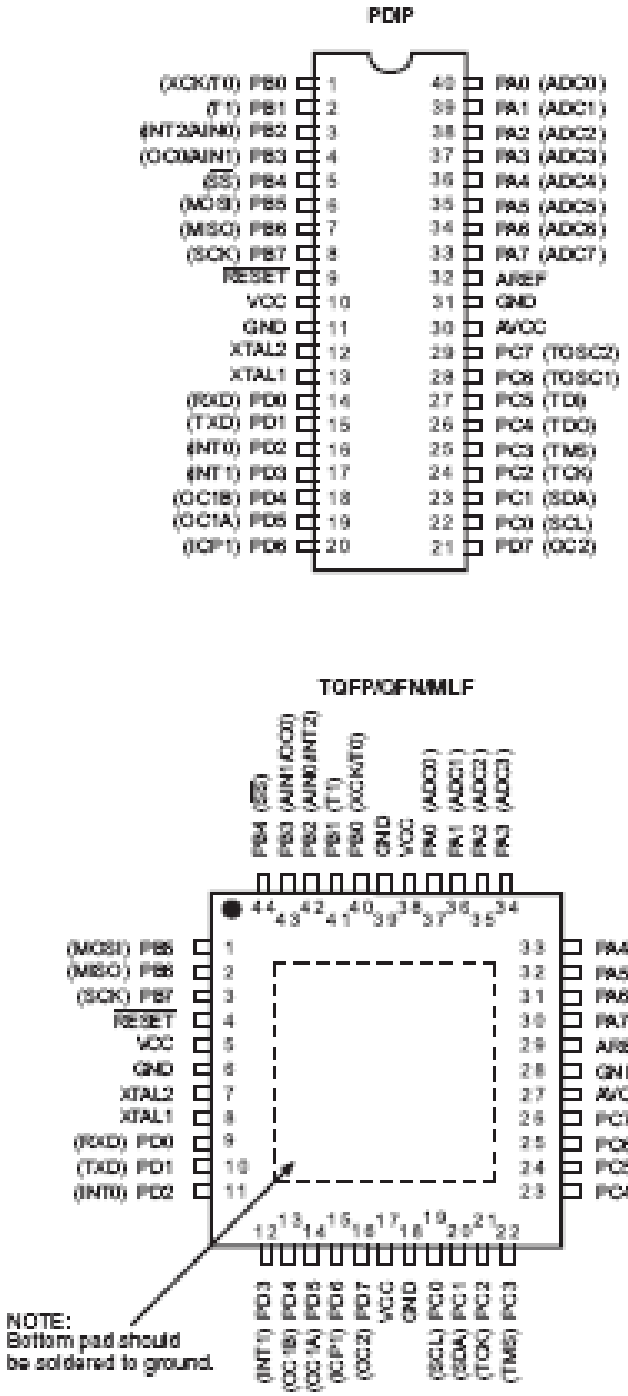
8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

Rev. 2466T–AVR–07/10

Pin Configurations

Figure 1. Pinout ATmega16



Disclaimer

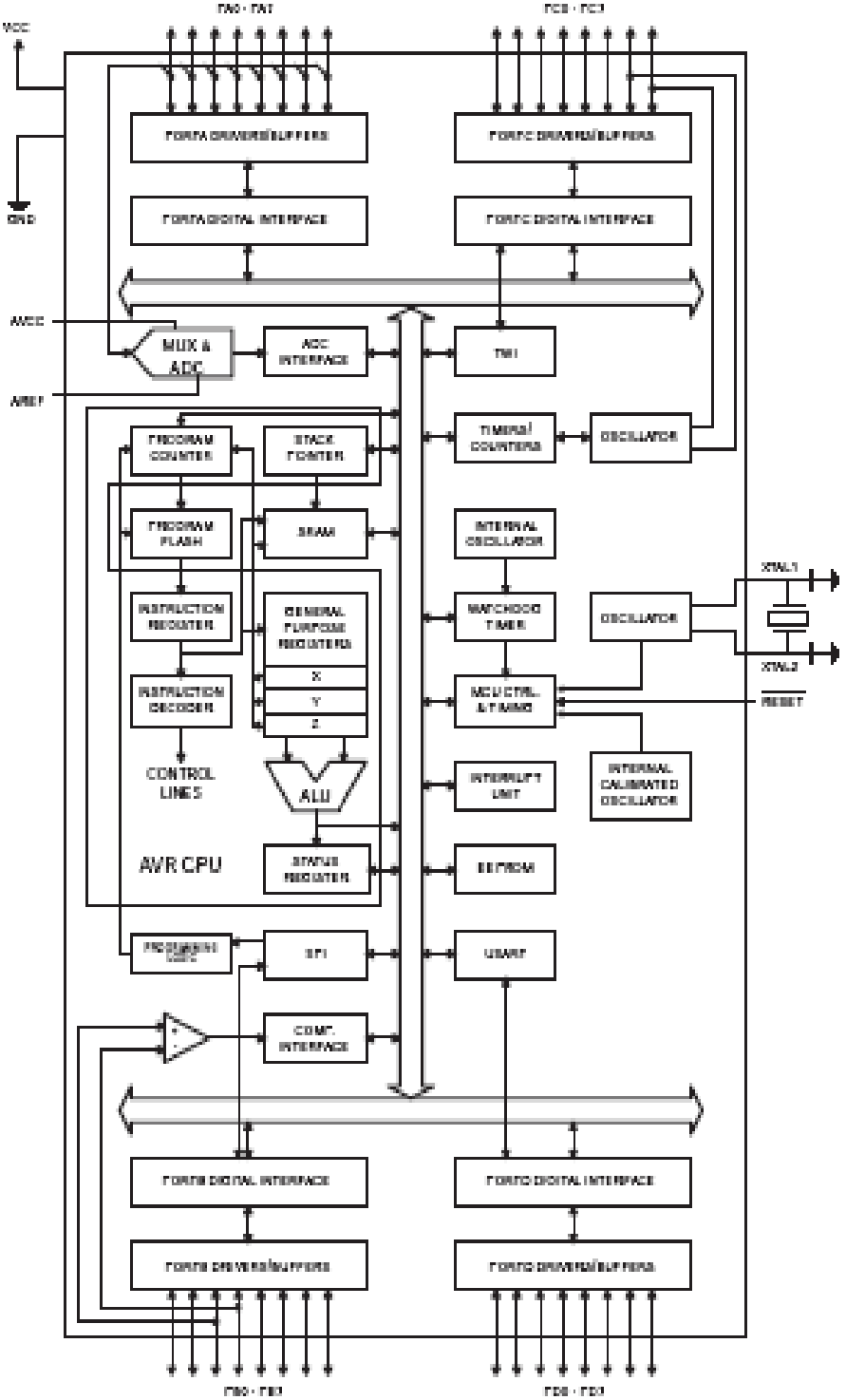
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-circuit emulators, and evaluation kits.

Pin Descriptions

VCC — Digital supply voltage.

GND — Ground.

Port A (PA7..PA0) — Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega16 as listed on page 58.
Port C (PC7..PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 61.
Port D (PD7..PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega16 as listed on page 63.
$\overline{\text{RESET}}$	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.

LAMPIRAN III

Listing Program

```
*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : PID cascade Motor DC
Version :
Date    : 12/02/2015
Author  : R.Praja
Company : Toshiba
Comments:

Chip type           : ATmega16
Program type        : Application
AVR Core Clock frequency: 12,000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
#include <stdio.h>
#include <delay.h>
#include <alcd.h>

#define pwm_righ OCR1A
#define pwm_left OCR1B

#define selA PORTB.5    //misal selektor A terhubung ke PB0
```

```
#define selB PORTB.6    //misal selektor B terhubung ke PB1
#define selC PORTB.7    //misal selektor C terhubung ke PB2
#define pinADC_sensor 0         //misal keluaran multiplekser terhubung ke PA0

#define m1a PORTB.0    //motor kanan in 1
#define m1b PORTB.1    //motor kanan in 2
#define m2a PORTB.3    //motor kiri in 1
#define m2b PORTB.4    //motor kiri in 2

#define hole_kanan 500
#define hole_kiri 500

#define batas 145 // batas adc

#define ADC_VREF_TYPE 0x60
unsigned char bufpwm1[20],bufpwm2[20],bufvel1[20],bufvel2[20],;
unsigned char adc1[20],
        adc2[20],
        adc3[20],
        adc4[20],
        adc5[20],
        adc6[20],
        adc7[20],
        adc8[20];

int frekuensi_ka,frekuensi_ki;
int kecepatan_kanan,kecepatan_kiri;
int error_kanan,
   error_kiri,
   error_pos,
   error1_kanan,
```

```
  error1_kiri,
  error1_pos,
  SP;
long int nil_pid_kanan,
   nil_pid_kiri,
   nil_pid_pos;

int s1,
  s2,
  s3,
  s4,
  s5,
  s6,
  s7,
  s8;

int pwm_kanan_sekarang,pwm_kiri_sekarang;
int a;
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
```

```
}

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
frekuensi_ka++;
}

// External Interrupt 2 service routine
interrupt [EXT_INT2] void ext_int2_isr(void)
{
frekuensi_ki++;
}

// Timer 0 overflow interrupt service routine
//interrupt [TIM0_OVF] void timer0_ovf_isr(void)
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
  kecepatan_kanan=(float)frekuensi_ka*60/hole_kanan;//formula perhitungan kecepatan
  frekuensi_ka=0;
  kecepatan_kiri=(float)frekuensi_ki*60/hole_kiri;//formula perhitungan kecepatan
  frekuensi_ki=0;
}

//// Timer2 overflow interrupt service routine
//interrupt [TIM2_OVF] void timer2_ovf_isr(void)
//{
//
//   kecepatan_kiri=(float)frekuensi_ki*60/hole_kiri;//formula perhitungan kecepatan
//   frekuensi_ki=0;
//}
```

```
void inisialisasi();
unsigned char baca_sensor(unsigned char channel);
void maju();
void rem_kanan();
void rem_kiri();
void berhenti();
void PID_kanan(float Kp, float Ki, float Kd, float Ts);
void PID_kiri(float Kp, float Ki, float Kd, float Ts);
void e_pos(float Kp_pos,float Ki_pos,float Kd_pos,float Ts_pos);
void uji_adc();
void uji_motor();
void uji_rotary();
void uji_rotary2();
void uji_logika();

void main(void)
{
inisialisasi();

while (1)
    {
    //uji_rotary2();
    PID_kanan(10.52,0.2,0,0.18);
    PID_kiri(10.52,0.2,0,0.18);
    e_pos(1.25,0,0,0.18);
    }
}

void inisialisasi()
{
```

```
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xFB;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x30;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 11,719 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
//TCCR0=0x05;
```

```
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 11,719 kHz
// Mode: Fast PWM top=0x00FF
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off

TCCR1A=0xA2;
TCCR1B=0x1D;

//TCCR1A=0xA1;
////TCCR1B=0x1D;
//TCCR1B=0x01;

TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x2D;
ICR1L=0xC7;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 11,719 kHz
// Mode: Normal top=0xFF
// OC2 output: Disconnected
//ASSR=0x00;
//TCCR2=0x07;
//TCNT2=0x00;
//OCR2=0x8A;
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: On
// INT2 Mode: Falling Edge
GICR|=0x60;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x60;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x45;
```

```
// USART initialization
// USART disabled
UCSRB=0x00;

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

// ADC initialization
// ADC Clock frequency: 750,000 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: ADC Stopped
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
```

```
// RD - PORTC Bit 1
// EN - PORTC Bit 2
// D4 - PORTC Bit 4
// D5 - PORTC Bit 5
// D6 - PORTC Bit 6
// D7 - PORTC Bit 7
// Characters/line: 16
lcd_init(16);

// Global enable interrupts
#asm("sei")
}

unsigned char baca_sensor(unsigned char channel)
 {
        unsigned char hasil;
        switch(channel)
 {
        case 0 : selC=0; selB=0; selA=0; break;
        case 1 : selC=0; selB=0; selA=1; break;
        case 2 : selC=0; selB=1; selA=0; break;
        case 3 : selC=0; selB=1; selA=1; break;
        case 4 : selC=1; selB=0; selA=0; break;
        case 5 : selC=1; selB=0; selA=1; break;
        case 6 : selC=1; selB=1; selA=0; break;
        case 7 : selC=1; selB=1; selA=1; break;
        }

        delay_us(1); //untuk propagration delay dan rising/falling time
        hasil = read_adc(pinADC_sensor);
        return hasil;
```

```
}

void maju ()
{
m1a=1;m1b=0;
m2a=1;m2b=0;
}

void rem_kanan ()
{
m1a=0;m1b=1;
m2a=0;m2b=1;
}
void rem_kiri ()
{
m1a=1;m1b=0;
m2a=1;m2b=0;
}

void berhenti()
{
m1a=1;m1b=1;
m2a=1;m2b=1;
}

void PID_kanan(float Kp, float Ki, float Kd, float Ts)
{
SP=(read_adc(1)/255)*1700;
error_kanan=SP-kecepatan_kanan;
nil_pid_kanan=((Kp*error_kanan)+((Ki/10)*(error_kanan+error1_kanan)*Ts)+((Kd/Ts)*(error_kanan-
error1_kanan)));
```

```
error1_kanan=error_kanan;

  if(error_kanan==0)
  {
   pwm_righ=pwm_righ*46;
  }

  else if(error_kanan<0)//kondisi terlalu cepat maka error<0
  {
   pwm_righ=(10-nil_pid_kanan)*46;
  }

  else //terlalu lambat erorr>0
  {
   pwm_righ=(10+nil_pid_kanan)*46;
  }
delay_ms(Ts);
}

void PID_kiri(float Kp, float Ki, float Kd, float Ts)
{
SP=(read_adc(1)/(255))*1700;
error_kiri=SP-kecepatan_kiri;
nil_pid_kiri=((Kp*error_kiri)+((Ki/10)*(error_kiri+error1_kiri)*Ts)+((Kd/Ts)*(error_kiri-error1_kiri)));
error1_kiri=error_kiri;
pwm_left=10+nil_pid_kiri;

  if(error_kiri==0)
  {
   pwm_left=pwm_left*46;
  }
```

```
  else if(error_kiri<0)//kondisi terlalu cepat maka error<0
  {
   pwm_left=(10-nil_pid_kiri)*46;
  }

  else //terlalu lambat erorr>0
  {
   pwm_left=(10+nil_pid_kiri)*46;
  }
delay_ms(Ts);
}

void uji_adc()
{
  sprintf(adc1,"%d",baca_sensor(3));
  lcd_gotoxy(0,0);
  lcd_puts(adc1);

  sprintf(adc2,"%d",baca_sensor(0));
  lcd_gotoxy(4,0);
  lcd_puts(adc2);

  sprintf(adc3,"%d",baca_sensor(1));
  lcd_gotoxy(8,0);
  lcd_puts(adc3);

  sprintf(adc4,"%d",baca_sensor(2));
  lcd_gotoxy(12,0);
  lcd_puts(adc4);
```

```
    sprintf(adc5,"%d",baca_sensor(4));
    lcd_gotoxy(0,1);
    lcd_puts(adc5);

    sprintf(adc6,"%d",baca_sensor(6));
    lcd_gotoxy(4,1);
    lcd_puts(adc6);

    sprintf(adc7,"%d",baca_sensor(7));
    lcd_gotoxy(8,1);
    lcd_puts(adc7);

    sprintf(adc8,"%d",baca_sensor(5));
    lcd_gotoxy(12,1);
    lcd_puts(adc8);
    delay_ms(150);
    lcd_clear();
}

void rule()
{
    //Hitam= 0 Putih=1
    if (baca_sensor(3)>batas)
    s1=1;
    else
    s1=0;

    if (baca_sensor(0)>batas)
    s2=1;
    else
    s2=0;
```

```
    if (baca_sensor(1)>batas)
    s3=1;
    else
    s3=0;

    if (baca_sensor(2)>batas)
    s4=1;
    else
    s4=0;

    if (baca_sensor(4)>batas)
    s5=1;
    else
    s5=0;

    if (baca_sensor(6)>batas)
    s6=1;
    else
    s6=0;

    if (baca_sensor(7)>batas)
    s7=1;
    else
    s7=0;

    if (baca_sensor(5)>batas)
    s8=1;
    else
    s8=0;
}
```

```
void e_pos(float Kp_pos,float Ki_pos,float Kd_pos,float Ts_pos)
{
  /* Kp_pos=0;
   Ki_pos=0;
   Kd_pos=0;
   Ts_pos=0;
  */
   rule();
        /*
        //Hitam= 0 Putih=1
        11111110 (PV=-7)
        11111000 (PV=-6)
        11111100 (PV=-6)
        11111101 (PV=-5)
        11110001 (PV=-4)
        11111001 (PV=-4)
        11111011 (PV=-3)
        11100011 (PV=-2)
        11110011 (PV=-2)
        11110111 (PV=-1)
        11100111 (PV=0)
        11101111 (PV=1)
        11000111 (PV=2)
        11001111 (PV=2)
        11011111 (PV=3)
        10001111 (PV=4)
        10011111 (PV=4)
        10111111 (PV=5)
        00011111 (PV=6)
        00111111 (PV=6)
```

```
    01111111 (PV=7)
    11111111 (stop)
    */
//sensor mengarah ke kiri
if (s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==1 && s7==1 && s8==0)
error_pos=-7;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==0 && s7==0 && s8==0)
error_pos=-6;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==1 && s7==0 && s8==0)
error_pos=-6;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==1 && s7==0 && s8==1)
error_pos=-5;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==0 && s6==0 && s7==0 && s8==1)
error_pos=-4;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==0 && s7==0 && s8==1)
error_pos=-4;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==0 && s7==1 && s8==1)
error_pos=-3;
else if(s1==1 && s2==1 && s3==1 && s4==0 && s5==0 && s6==0 && s7==1 && s8==1)
error_pos=-2;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==0 && s6==0 && s7==1 && s8==1)
error_pos=-2;
else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==0 && s6==1 && s7==1 && s8==1)
error_pos=-1;

else if(s1==1 && s2==1 && s3==1 && s4==0 && s5==0 && s6==1 && s7==1 && s8==1)
error_pos=0;

//sensor mengarah ke kanan
else if(s1==1 && s2==1 && s3==1 && s4==0 && s5==1 && s6==1 && s7==1 && s8==1)
error_pos=1;
```

```
  else if(s1==1 && s2==1 && s3==0 && s4==0 && s5==0 && s6==1 && s7==1 && s8==1)
  error_pos=2;
  else if(s1==1 && s2==1 && s3==0 && s4==0 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=2;
  else if(s1==1 && s2==1 && s3==0 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=3;
  else if(s1==1 && s2==0 && s3==0 && s4==0 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=4;
  else if(s1==1 && s2==0 && s3==0 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=4;
  else if(s1==1 && s2==0 && s3==1 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=5;
  else if(s1==0 && s2==0 && s3==0 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=6;
  else if(s1==0 && s2==0 && s3==1 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=6;
  else if(s1==0 && s2==1 && s3==1 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  error_pos=7;
  else if(s1==1 && s2==1 && s3==1 && s4==1 && s5==1 && s6==1 && s7==1 && s8==1)
  berhenti();


nil_pid_pos=((Kp_pos*error_pos)+((Ki_pos/10)*(error_pos+error1_pos)*Ts_pos)+((Kd_pos/Ts_pos)*
(error_pos-error1_pos)));
  error1_pos=error_pos;


  pwm_kanan_sekarang=pwm_righ+nil_pid_pos;
  pwm_kiri_sekarang=pwm_left-nil_pid_pos;


  delay_ms(Ts_pos);


  }
```

```
void uji_motor()
{
    maju();
    pwm_righ=read_adc(1)*46;
    pwm_left=read_adc(1)*46;

    sprintf(bufpwm2,"pwm ki= %d",pwm_left/46);
    lcd_gotoxy(1,1);
    lcd_puts(bufpwm2);

    sprintf(bufpwm1,"pwm ka= %d",pwm_righ/46);
    lcd_gotoxy(1,0);
    lcd_puts(bufpwm1);
    delay_ms(50);
    lcd_clear();
}

void uji_rotary()
{
    maju();
//    pwm_righ=200; //read_adc(1);
//    pwm_left=20; //read_adc(1);

   pwm_righ=read_adc(1)*46;
   pwm_left=read_adc(1)*46;

//    sprintf(bufpwm2,"ki=%d",kecepatan_kiri);
//    lcd_gotoxy(0,0);
//    lcd_puts(bufpwm2);
```

```
    sprintf(bufvel2,"fi=%d",frekuensi_ki);
    lcd_gotoxy(0,0);
    lcd_puts(bufvel2);


//    sprintf(bufpwm1,"ka=%d",kecepatan_kanan);
//    lcd_gotoxy(0,1);
//    lcd_puts(bufpwm1);

    sprintf(bufvel1,"fa=%d",frekuensi_ka);
    lcd_gotoxy(0,1);
    lcd_puts(bufvel1);

    sprintf(bufpwm2,"pi=%d",pwm_left/46);
    lcd_gotoxy(8,0);
    lcd_puts(bufpwm2);

    sprintf(bufpwm1,"pa=%d",pwm_righ/46);
    lcd_gotoxy(8,1);
    lcd_puts(bufpwm1);
    delay_ms(100);
    lcd_clear();
}

void uji_logika()
{
 rule();

   sprintf(adc1,"%d",s1);
   lcd_gotoxy(0,0);
   lcd_puts(adc1);
```

```
sprintf(adc2,"%d",s2);
lcd_gotoxy(2,0);
lcd_puts(adc2);

sprintf(adc3,"%d",s3);
lcd_gotoxy(4,0);
lcd_puts(adc3);

sprintf(adc4,"%d",s4);
lcd_gotoxy(6,0);
lcd_puts(adc4);

sprintf(adc5,"%d",s5);
lcd_gotoxy(8,0);
lcd_puts(adc5);

sprintf(adc6,"%d",s6);
lcd_gotoxy(10,0);
lcd_puts(adc6);

sprintf(adc7,"%d",s7);
lcd_gotoxy(12,0);
lcd_puts(adc7);

sprintf(adc8,"%d",s8);
lcd_gotoxy(14,0);
lcd_puts(adc8);

lcd_gotoxy(0,1);
lcd_puts("Sensor 1-8 ki-ka");
```

```
   delay_ms(150);
   lcd_clear();
}
void uji_rotary2()
{
    maju();
//    pwm_righ=200; //read_adc(1);
//    pwm_left=20; //read_adc(1);

   pwm_righ=read_adc(1)*46;
   pwm_left=read_adc(1)*46;

    sprintf(bufvel2,"ki=%d",kecepatan_kiri);
    lcd_gotoxy(0,0);
    lcd_puts(bufvel2);

    sprintf(bufvel1,"ka=%d",kecepatan_kanan);
    lcd_gotoxy(0,1);
    lcd_puts(bufvel1);

    sprintf(bufpwm2,"pi=%d",pwm_left/46);
    lcd_gotoxy(8,0);
    lcd_puts(bufpwm2);

    sprintf(bufpwm1,"pa=%d",pwm_righ/46);
    lcd_gotoxy(8,1);
    lcd_puts(bufpwm1);
    delay_ms(100);
    lcd_clear();
}
```