KONTROL KECEPATAN MOTOR DC DENGAN METODE PID MENGGUNAKAN VISUAL BASIC 6.0 DAN MIKROKONTROLER ATMEGA 16

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
MUHAMMAD RIZKI SETIAWAN
NIM. 0710630066

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK ELEKTRO

2013

LEMBAR PERSETUJUAN

KONTROL KECEPATAN MOTOR DC DENGAN METODE PID MENGGUNAKAN VISUAL BASIC 6.0 DAN MIKROKONTROLER ATMEGA 16

SKRIPSI
KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
MUHAMMAD RIZKI SETIAWAN
NIM. 0710630066

Telah diperikas dan disetujui oleh:

Pembimbing I	Pembimbing II
M. Aziz Muslim, ST., MT., Ph.D.	**Goegoes Dwi Nusantoro, ST., MT.**
NIP. 19741203 200012 1 001	**NIP. 19711013 200604 1 001**

LEMBAR PENGESAHAN

KONTROL KECEPATAN MOTOR DC DENGAN METODE PID MENGGUNAKAN VISUAL BASIC 6.0 DAN MIKROKONTROLER ATMEGA 16

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

MUHAMMAD RIZKI SETIAWAN
NIM. 0710630066

Skripsi ini telah diuji dan dinyatakan lulus pada
Tanggal 13 Mei 2013

DOSEN PENGUJI

Dr. Ir. Erni Yudaningtyas, MT.	**Zainul Abidin, ST., MT., MEng.**
NIP. 19650913 199002 2 001	**NIK. 86012306110279**

Ir. Purwanto, MT.
NIP. 19540424 198601 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS.
NIP. 19580728 198701 1 001

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur kehadirat Allah SWT yang telah melimpahkan rahmad dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "KONTROL KECEPATAN MOTOR DC DENGAN METODE PID MENGGUNAKAN VISUAL BASIC 6.0 DAN MIKROKONTROLER ATMEGA 16". Skripsi ini disusun sebagai salah satu syarat mendapatkan gelar Sarjana Teknik di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.

Pada kesempatan ini penulis mengucapkan terima kasih kepada Bapak M. Aziz Muslim, ST., MT., Ph.D. dan Bapak Goegoes Dwi Nusantoro, ST., MT. sebagai dosen pembimbing skipsi yang telah memberikan bimbingan,saran dan motivasi sehingga skripsi ini dapat terselesaikan. Tidak lupa, ucapan terima kasih penulis ucapkan kepada:

1. Keluarga. Ayah,Ibuk, mbak Ita, mbak Arie dan dek Lia yang telah memberi dukungan serta doa selama ini.
2. Bapak Sholeh Hadi Pramono,ST.,MT.,PhD. Selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
3. Bapak Ir. Purwanto MT. selaku Ketua Kelompok Dosen dan Keahlian Teknik Kontrol.
4. Bapak Ali Mustofa. ST,.MT selaku dosen pembimbing akademik.
5. Bapak, Ibu dosen serta segenap staf dan karyawan Jurusan Teknik Elektro baik secara langsung maupun tidak langsung telah membantu dalam menyelesaian skripsi ini.
6. Anin, Taufik, Bhasori, Agung Adiyatma, Bagus Ilyas, Doni, Firda, Rizqi Rahmawan, Afriono Subekti, Khasbullah Huda, Kadir, Gio, Sofyan teman-teman elektro yang telah memberikan ilmu, informasi dan fasilitas yang dimiliki untuk kelancaran skripsi ini.
7. Mushbikin Spirit, Taufik Sanjaya, Rohmadi, Ian Westbury, Wahyu Rahmaniar, Andi Dermawan, Toopayz DE, Iful Jericho. Teman-teman dunia

8. maya yang mau membagikan ilmu nya meskipun hanya berkomunikasi lewat dunia maya.
9. Fordi Mapelar UB, yang telah mendidik, membina dan meningkatkan penalaran untuk menjadi manusia yang lebih bermanfaat melalui pengalaman dan kesempatan yang diberi, terima kasih kawan-kawan, Salam Penalaran.
10. Teman-teman intra dan ekstra kampus yang telah banyak memberikan warna memalui gesekan-gesekan yang ada, terima kasih pengalamannya.
11. Mbak Nurul Hasanah, SPsi, terima kasih atas segala bentuk dukungannya selama ini dan selanjutnya.

Penulis menyadari bahwa skripsi yang telah tersusun ini masih banyak kekurangan karena keterbatasan ilmu dan pengalaman dari penulis, maka dari itu segala bentuk kritik dan saran sangat diharapkan demi kemajuan ilmu pengetahuan manusia. Akhir kata, penulis berharap akan ada kebaikan yang terjadi dari skripsi ini.

Malang, 15 April 2013

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Dalam penelitian ini telah diimplementasikan suatu sistem kontrol motor DC dengan kontroler PID menggunakan mikrokontroler ATmega16. *Interface* sistem kontrol ini menggunakan Visual Basic 6.0 sehingga memudahkan untuk pengembangan lebih lanjut. Proses identifikasi sistem dilakukan dengan menggunakan sinyal *input Pseudo Random Binary Sequence* (PRBS) 10 bit dengan variasi sinyal acak sebanyak 1024 data. Dari hasil identifikasi diperoleh fungsi alih model system adalah $\boldsymbol{F(s)} = \frac{3{,}492\ s^2+24{,}4\ s+0{,}5313}{s^3+9{,}173\ s^2+32{,}05+0{,}5376}$ Selanjutnya dilakukan proses penentuan parameter PID dengan menggunakan metode *root locus*, yang hasilnya menunjukkan bahwa semua akar berada disebelah kiri bidang *s*. Sehingga respon yang didapat dari semua pole stabil. Hasil perhitungan parameter PID dengan pole s = -9.4 + j3.2 didapatkan nilai parameter PID terbaik yaitu K_P=4.6805, K_I=10 dan K_D=0.1026.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kebutuhan akan sistem kontrol yang lebih efektif dan efisien di era modern ini semakin meningkat, mengingat bahwa jumlah *plant* yang akan dikontrol semakin banyak dan memiliki struktur yang semakin kompleks. Maka dari itu sistem kontrol yang dapat dikendalikan melalui perangkat komputer mutlak diperlukan. Selain dapat melakukan sistem *monitoring* secara *real time* dengan menggunakan penyajian data yang lebih bagus, perangkat komputer dapat dengan mudah melakukan proses pengiriman *set point* yang dikehendaki.

Salah satu jenis kontroler yang banyak digunakan saat ini adalah kontroler PID (Proporsional Integral Diferensial) karena kontroler ini sederhana dan relatif mudah dalam pengaplikasiannya. Pada umumnya PID diimplementasikan menggunakan rangkaian analog bahkan ada yang menggunakan komponen mekanis dalam penentuan *set point, setting parameter.*

Berdasarkan permasalahan tersebut maka dalam skripsi ini akan dirancang sebuah alat dengan piranti lunak yang mampu mengendalikan kecepatan motor DC dengan sistem *monitoring* secara *real time* sehingga memudahkan untuk mengetahui perubahan data yang terjadi. Data yang disajikan langsung dirubah menjadi bentuk grafik supaya dapat dimengerti dengan baik, hal tersebut dapat dilakukan mengingat pembuatan perangkat lunak dalam skripsi ini dilakukan menggunakan program Visual Basic 6.0 yang dapat menyajikan data dalam bentuk grafik secara *real time.* Sedangkan untuk perangkat kerasnya menggunakan Mikrokontroler ATMega16 dan perantara port serial dalam komunikasi datanya.

1.2 Rumusan Masalah

Mengacu pada permasalahan yang telah diuraikan pada latar belakang, maka rumusan masalah dapat ditekankan pada beberapa point berikut:

1. Bagaimana merancang perangkat keras yang mampu mengontrol motor DC dan mengirim data kecepatan motor DC ke komputer?
2. Bagaimana merancang perangkat lunak menggunakan Visual Basic yang mampu melakukan *monitoring* kecepatan motor DC dan mengirim *set point, setting parameter* PID pada Mikrokontroler?
3. Bagaimana memanfaatkan program MATLAB untuk mendapatkan fungsi alih *plant* dan menentukan nilai PID?

1.3 Batasan Masalah

Mengacu pada permasalahan yang ada, maka skripsi ini akan dibatas pada:

1. Perangkat lunak dibuat dengan menggunkan Visual Basic 6.0.
2. Pembahasan ditekankan pada proses *monitoring*, pengiriman *set point, setting paramter* PID. Kinerja driver dan rangkaian elektronika tidak dibahas mendalam
3. Metode kontrol menggunakan PID (Proporsional Integral Diferensial).
4. Menggunakan sensor *rotary encoder*.
5. Menggunakan Mikrokontroler ATmega16
6. Menggunakan serial *port*.

1.4 Tujuan

Merancang perangkat lunak yang mampu melakukan monitoring, pengiriman *set point* dan parameter PID pada Mikrokontroler untuk menjalankan driver.

1.5 Sistematika Pembahasan

Sistematika penulisan yang digunakan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

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

BAB II Tinjauan Pustaka

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

BAB III Metodologi

Berisi tentang metode penelitian dan perencanaan alat serta pengujian.

BAB IV Perancangan dan Pembuatan Alat

Perancangan alat yang merupakan spesifikasi, perencanaan blok diagram, prinsip kerja dan pembuatan alat.

BAB V Pengujian dan Analisis

Memuat hasil pengujian terhadap alat yang telah dibuat.

BAB VI Kesimpulan dan Saran

Memuat kesimpulan dan saran-saran.

BAB II
TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai teori-teori yang akan menunjang perancangan Kontrol Kecepatan Motor DC dengan metode PID menggunakan Visual Basic 6.0 dan Mikrokontroler ATmega16.

2.1 Pengenalan Visual Basic 6.0

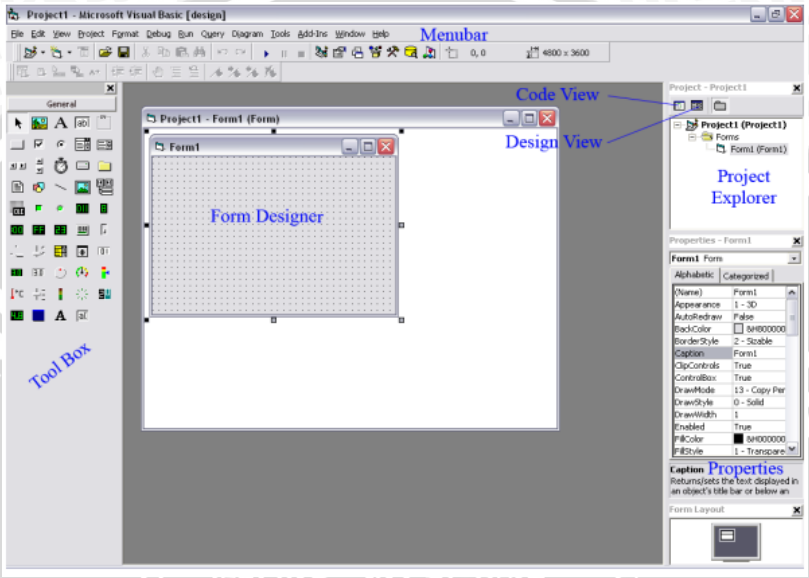
Visual Basic adalah salah satu bahasa pemrograman komputer. Bahasa pemrograman adalah perintah yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Bahasa pemrograman Visual Basic, yang dikembangkan oleh Microsoft sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu bahasa pemrograman BASIC (*Beginner's All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. Visual Basic merupakan salah satu *Development Tool* yaitu alat bantu untuk membuat berbagai macam program komputer, khususnya yang menggunakan sistem operasi Windows. Visual Basic merupakan salah satu bahasa pemrograman komputer yang mendukung object (*Object Oriented Programming* = OOP).

Dengan menggunakan Visual Basic 6.0 dapat menghasilkan berbagai macam jenis program. Aplikasi yang dibuat dapat diintegrasikan dengan *database*, *hardware* lain (*interface*) dan sebagainya.

Gambar 2.1 Tampilan Awal Visual Basic 6.0

Pada layar awal akan muncul tampilan seperti di atas. Visual Basic 6.0 menyediakan banyak jenis modul aplikasi. Untuk memulai program standar pilihlah **Standar EXE**, kemudian klik *open*.

Setelah itu akan muncul tampilan seperti berikut ini, yang menunjukan bagian-bagian dari IDE (*Integrated Development Environment*) yang akan digunakan.



Gambar 2.2 Tampilan Lembar Kerja Visual Basic 6.0

Form Designer

Form design merupakan tempat perancangan *user interface* (antar muka pemakai). Untuk menampilkan layar ini, klik **Design View** atau dengan menekan **shift + F7**. Sedangkan untuk menampilkan layar *coding* dapat menekan **F7** atau klik pada ***Coding View***.

Menu

Merupakan menu standar pada *Windows*, dapat digunakan untuk menyimpan *project*, menjalankan *project*, membuka *projec*t baru dan sebagainya.

Toolbox

Toolbox merupakan tempat komponen-komponen yang disediakan untuk merancang user *interface*. Setiap komponen memiliki ciri dan kegunaan yang berbeda, disesuaikan dengan kebutuhan pengguna.

Project Explorer

Merupakan struktur *project* yang sedang dikerjakan, suatu *project* dapat terdiri

dari beberapa *form.*

Properties

Menampilkan bagian-bagian dari komponen yang sedang aktif. Setiap komponen mempunyai karakteristik yang berbeda, bergantung pada kegunaan.

2.2 Pengkodean Pada Visual Basic 6.0

2.2.1 Tipe Data

Tipe data memiliki ciri – ciri tersendiri. Berikut bentuk dan ukuran dari tipe data:

Tabel 2.1 Ukuran Dari Tipe Data

Tipe Data	Ukuran Storage	Jangkauan
Byte	1 Byte	0 s/d 255
Boolean	1 Byte	True atau False
Integer	2 Byte	-32768 s/d 32767
Long	2 Byte	-2.147.483.648 s/d 2.147.483.647
Single	4 Byte	-3,40282e38 s/d -1,401296e-45 (-)
		1,401296e-45 s/d 3,402823e38 (+)
Double	8 Byte	-1,797691348623e308 s/d -4,9406564844127
Currency	8 Byte	-922.337.203.685.477,5808 s/d 922.337.203.685.477,5807
Decimal	14 Byte	7,92E+028
Date	8 Byte	1 Januari 100 s/d 31 desember 9999
Object	4 Byte	Mangacu pada objek tertentu
String	Panjang dari string	1 sampai ± 65400
Variant	16 Byte	Sembarang angka sampai jangkauan jenis double atau string

2.2.2 Variabel

Variabel digunakan untuk menampung nilai sementara di memori. Untuk membuat sebuah variabel terdapat ketentuan sebagai berikut:

a. Harus dimulai dengan suatu huruf
b. Tidak dapat mengandung titik atau spesial karakter
c. Tidak dapat lebih dari 255 huruf
d. Tidak dapat sama dengan *keyword* dari visual basic
e. Tidak membedakan huruf besar dan huruf kecil (*no case sensitive*)

Cara mendeklarasi variabel adalah sebagai berikut :

```
Dim [nama variabel] As [tipe data] atau
Public [nama variabel] As [tipe data] atau
Private [nama variabel] As [tipe data]
```

Public akan membuat suatu variabel dapat diakses dari segala tempat di dalam *project*, sedangkan Dim dan *Private* akan membuat suatu variabel yang hanya dapat diakses di dalam module dimana variabel tersebut dideklarasikan.

2.2.3 Operator

Visual basic menyediakan operator aritmatika, komparasi dan logika, salah satu hal yang harus dipahami adalah tata urut dari masing–masing operator, sehingga mampu membuat ekspresi yang akan menghasilkan nilai yang benar.

a. Operator aritmatika

Tabel 2.2 Operator Aritmatika

Nama Operator	Tanda Operator
Pangkat	^
Negatif	-
Kali dan Bagi	*, /
Pembagian Bulat	\
Sisa Bagi	Mod

Tambah dan Kurang	+, -
Penggabungan string	&

b. Operator Komparasi

Tabel 2.3 Operator Komparasi

Nama Operator	Tanda Operator
Sama	=
Tidak Sama	<>
Kurang dari	<
Lebih dari	>
Pembagian Bulat	\
Kurang dari sama	<=
Lebih dari sama	>=
like	like

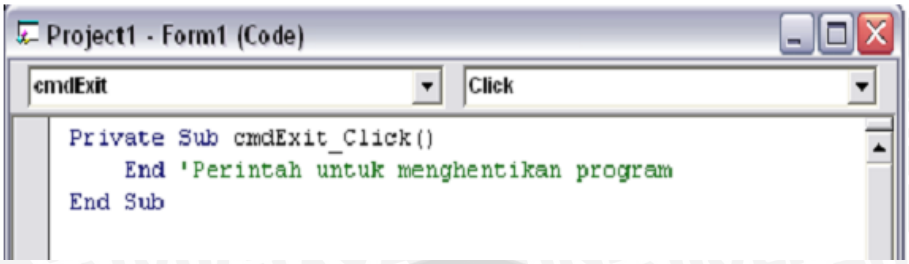
c. Operator Logika

Tabels 2.4 Operator Logika

Nama Operator	Tanda Operator
Not	Not
And	And
Or	Or
Xor	Xor

2.2.4 *Event-Driven*

Visual Basic 6.0 bersifat *event-driven*, artinya program bekerja berdasarkan *event* yang terjadi pada objek di dalam program tersebut. Misalkan *event* klik pada tombol *exit*, pada saat tombol *exit* diklik maka program akan berhenti/keluar. Caranya klik dua kali pada tombol *exit*, maka akan muncul layar *coding* sebagai berikut.



Gambar 2.3 Tampilan Layar Pengkodean

2.2.5 Struktur Pemilihan

Struktur pemilihan biasanya menggunakan *if-else*. Contoh penggunaan perintah *if-else*, sebagai berikut:

```
If kondisi Then
        ....
        ....
End If
```

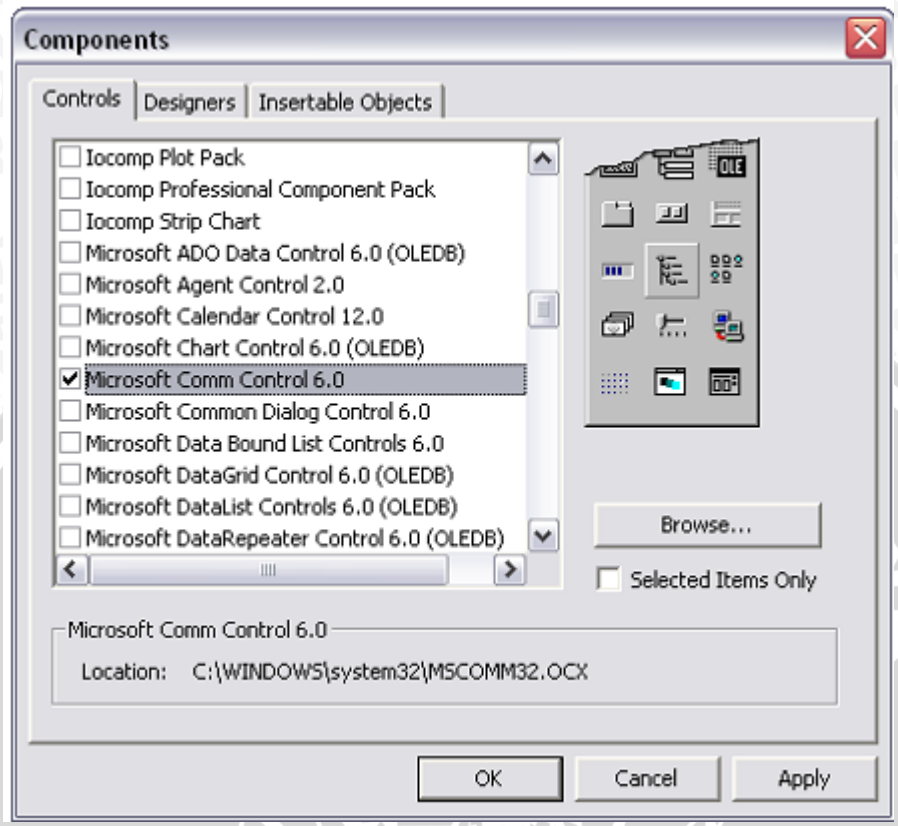
```
If kondisi Then
        ....
        ....
Else
        ....
        ....
End If
```

```
If kondisi Then
        ....
        ....
ElseIf kondisi Then
        ....
        ....
End If
```

2.3 Komunikasi Serial Visual Basic 6.0

Dalam berkomunikasi dengan "dunia luar" atau perangkat lain di luar komputer, Visual Basic 6.0 menyediakan komponen *MS Comm Control 6.0*, sebagai media komunikasi. Untuk menambahkan komponen ini pada Visual

Basic, pilih *Project* -> *Components*. Setelah itu akan muncul tampilan sebagai berikut:



Gambar 2.4 Penambahan Komponen Pada Visual Basic 6.0

Fungsi dari komponen ini adalah sebagai berikut :

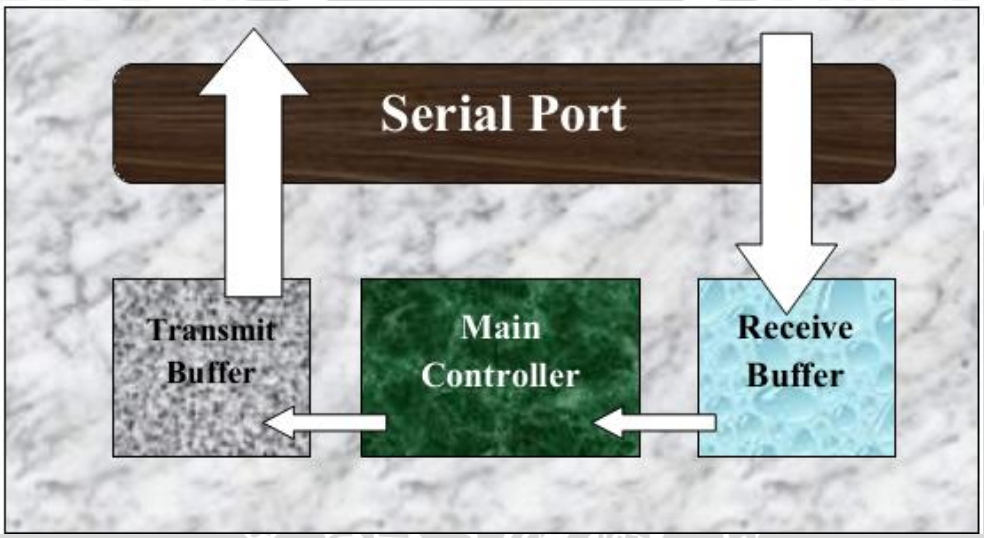
a. Mengadakan hubungan dengan serial *port* PC
b. Berhubungan dengan alat komunikasi lain (contoh: modem)
c. Melakukan pertukaran data
d. Memonitor dan merespon *event* dan *error* yang terjadi pada hubungan serial

Untuk mengadakan suatu komunikasi serial antara 2 peralatan, kita harus melakukan beberapa langkah.

1. Membuka serial *port*
2. Mengatur serial *device*
3. *Setting Receive* and *Transmit Buffer Properties*
4. *Managing Receive* and *Transmit Buffer*

2.3.1 Membuka Serial *Port*

Pada komunikasi serial, bit-bit data yang masuk dari dunia luar ke dalam komputer melalui serial *port* akan ditampung dulu di *receive buffer* sebelum akan dieksekusi oleh *main controller*. Demikian pula sebelum dikirimkan ke luar, data akan ditampung dulu di *transmit buffer*. Gambar skema lengkapnya dapat dilihat pada gambar di bawah ini.



Gambar 2.5 Skema Jalur Penerimaan dan Pengiriman Data

Sebelum membuka serial *port*, dilakukan pengaturan protokol komunikasi serial dengan *property MSComm*. Menentukan nomor *port* komunikasi menggunakan *CommPort* dan menentukan *baud rate, parity, data bits, stop bits* mengguankan *property setting*. Sedangkan untuk membuka serial *port* cukup menggunakan perintah *PortOpen*. Sehingga kode program akan tertulis sebagai berikut :

```
MSComm1.ComPort = 2
MSComm1.Settings = "9600,N,8,1"
MSComm1.PortOpen = True
```

2.3.2 Mengatur Serial *Device*

Pada tahap ini kita perlu memastikan bahwa pengaturan protokol komunikasi serial yang digunakan pada peralatan lain yang kita akses, sesuai dengan pengaturan pada komputer yang kita pakai.

2.3.3 *Setting Receive* dan *Transmit Buffer Properties*

Ada beberapa *property* dari *receive buffer* dan *transmit buffer (porperty* dari *MSComm)* yang perlu kita atur antar lain:

a. *InBufferSize* : mengatur ukuran *receive buffer*.

b. *OutBuffer Size* : mengatur ukuran *transmit buffer*.

c. *Rthreshold* : menentukan jumlah karakter yang diterima oleh *receive buffer* sebelum *OnComm event* dipicu.

d. *Sthreshold* : menentukan jumlah karakter yang diterima oleh *transmit buffer* sebelum *OnComm e*vent dipicu. Nilai 0 berarti tidak pernah dipicu, sedangkan nilai 1 berarti dipicu setiap satu karakter.

e. *InputLen* : menentukan jumlah karakter yang dibaca CPU dari *receive buffer*.

f. *InputMode* : menentukan tipe data input yang akan dibaca CPU . *comInputModeTex*t untuk data *string*/teks dan *comInputModeBinary* untuk data biner.

2.3.4 *Managing Receive* dan *Transmit Buffer*

Untuk menampilkan data dari peralatan lain ke dalam aplikasi VB, digunakan *properti Input*, sedangkan untuk mengirim data dari aplikasi VB ke peralatan lain digunakan poperti *Output*. Contoh struktur kode untuk *Input* dan *Output*.

```
TxtDisplay.Text = MSComm1.Input   ( Contoh Input )

MSComm1.Output = "Data String"   ( Contoh Output )
```

2.4 Studi Teknik Pengontrolan

Dalam Skripsi ini perancangan perangkat keras menggunakan Mikrokontroler ATMega16 dengan kontrol PID, masukan dari kontroler ini adalah nilai *set point*, *setting parameter* dan *output* nya merupakan kecepatan motor (dalam rpm) yang akan dikirm ke komputer melalui komunikasi serial.

2.4.1 Kontroler PID (Proporsional Integral Diferensial)

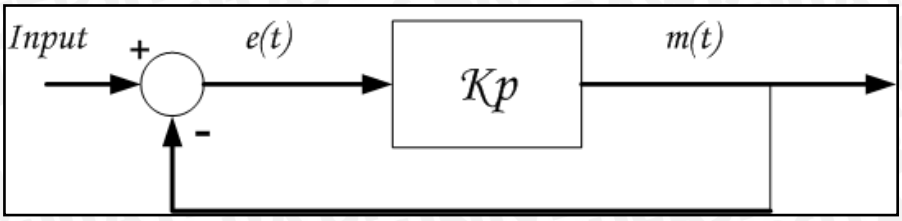
Didalam suatu sistem kontrol kita mengenal adanya beberapa macam aksi kontrol, diantaranya yaitu aksi kontrol proporsional, aksi kontrol integral dan aksi kontrol diferensial. Masing-masing aksi kontrol ini mempunyai keunggulan-keunggulan tertentu, dimana aksi kontrol proporsional mempunyai keunggulan *rise time* yang cepat, aksi kontrol integral mempunyai keunggulan untuk memperkecil *error* ,dan aksi kontrol diferensial mempunyai keunggulan untuk memperkecil *error* atau meredam *overshot/undershot*. Untuk itu agar kita dapat menghasilkan output dengan *rise time* yang cepat dan *error* yang kecil kita dapat menggabungkan ketiga aksi kontrol ini menjadi aksi kontrol PID.

Parameter pengontrol proporsional integral diferensial (PID) selalu didasari atas tinjauan terhadap karakteristik yang diatur (*plant*). Dengan demikian bagaimanapun rumitnya suatu *plant,* prilaku plant tersebut harus diketahui terlebih dahulu sebelum pencarian parameter PID itu dilakukan.

2.4.1.1 Kontroler Proporsional

Kontroler proporsional memiliki keluaran yang sebanding/proporsional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan besaran aktualnya). Secara lebih sederhana dapat dikatakan bahwa keluaran kontroler proporsional merupakan perkalian antara konstanta proporsional dengan masukannya. Perubahan pada sinyal masukan akan segera menyebabkan sistem secara langsung mengubah keluarannya sebesar konstanta pengali.

Pada gambar 2.6 menunjukkan blok diagram yang menggambarkan hubungan antara *input* (besaran referensi yang diinginkan), besaran aktual, besaran keluaran kontroler proporsional, dan besaran kesalahan (*error*). Sinyal kesalahan (*error*) merupakan selisih antara besaran *setting* dengan besaran aktualnya.



Gambar 2.6 Diagram Blok Kontroler Proporsional

Sumber: Philips, 1996: 220

Pada pengendali proporsional hubungan antara keluaran kontroler m(t) dan sinyal kesalahan e(t) adalah:

$$m(t) = Kp\ e(t) \tag{2.1}$$

dengan Kp adalah penguatan proporsional. Keluaran m(t) hanya tergantung pada Kp dan *error*, semakin besar *error* maka semakin besar koreksi yang dilakukan. Penambahan Kp akan menaikkan penguatan sistem sehingga dapat digunakan untuk memperbesar kecepatan respon dan mengurangi kesalahan keadaan mantap. Akan tetapi penambahan Kp juga membuat sistem lebih sensitif, yang akan mengakibatkan ketidakstabilan atau osilasi.

Maka ciri-ciri kontroler proporsional harus diperhatikan ketika kontroler tersebut ditetapkan pada suatu sistem, yaitu:

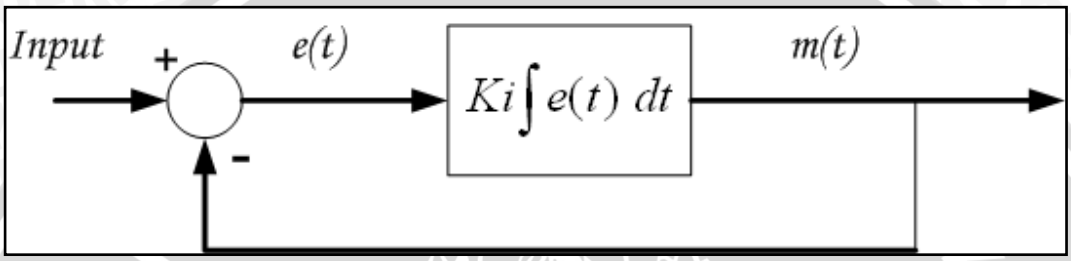
1. Nilai *Kp* kecil, kontroler proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat.
2. Nilai *Kp* dinaikkan, respon sistem menunjukkan semakin cepat mencapai keadaan mantapnya dan kesalahan (*error*) semakin kecil.
3. Namun jika nilai *Kp* diperbesar sehingga mencapai harga yang berlebihan (di luar penguatan stabil sistem), akan mengakibatkan sistem bekerja tidak stabil, atau respon sistem akan berosilasi.

2.4.1.2 Kontroler Integral

Kontroler integral berfungsi menghasilkan respon sistem yang memiliki kesalahan keadaan mantap nol. Kalau sebuah *plant* tidak memiliki unsur integrator (1/s), kontroler proporsional tidak mampu menjamin keluaran sistem dengan kesalahan mantapnya nol. Dengan kontroler integral respon sistem dapat diperbaiki, yaitu menjadikan kesalahan mantapnya nol.

Kontroler integral memiliki karakteristik seperti halnya sebuah integral. Keluaran kontroler sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan. Keluaran kontroler ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Bila sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan.

Gambar 2.7 menunjukkan blok diagram antara besaran kesalahan dengan keluaran suatu kontroler integral.



Gambar 2.7 Diagram Blok Kontroler Integral
Sumber: Philips, 1996: 220

Nilai keluaran kontroler m(t) sebanding dengan integral sinyal kesalahan e(t), sehingga

$$\frac{dm(t)}{dt} = Ki.e(t) \qquad (2.2)$$

$$m(t) = Ki \int_0^t e(t)\ dt \qquad (2.3)$$

dengan *Ki* adalah konstanta integral. Jika sinyal kesalahan *e(t)* = 0, maka laju perubahan sinyal kendali integral $\frac{dm(t)}{dt} = 0$, atau sinyal keluaran kendali akan tetap berada pada nilai yang dicapai sebelumnya. Aksi kontrol integral digunakan untuk menghilangkan kesalahan posisi dalam keadaan mantap (*error steady state*) tanpa memperhitungkan kecepatan respon.

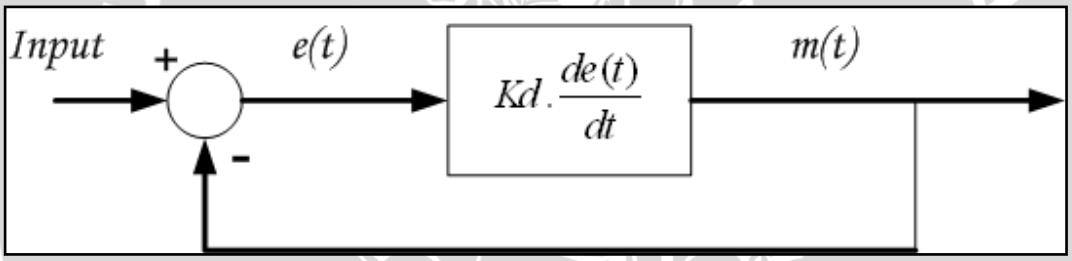
Pada penggunaan kontroler integral harus diperhatikan ciri-cirinya ketika kontroler tersebut diterapkan pada suatu sistem, yaitu:

1. Keluaran kontroler membutuhkan selang waktu tertentu, sehingga kontroler integral cenderung memperlambat respon.

2. Ketika sinyal kesalahan berharga nol, keluaran kontroler akan bertahan pada nilai sebelumnya.
3. Konstanta integral Ki yang berharga besar akan mempercepat hilangnya *offset*, tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran kontroler (Gunterus, 1994, 7-4).

2.4.1.3 Kontroler Differensial

Keluaran kontroler differensial memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan kontroler, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.8 Menunjukkan blok diagram yang menggambarkan hubungan antara sinyal kesalahan dengan keluaran kontroler.



Gambar 2.8 Diagram Blok Kontroler Differensial

Sumber: Philips, 1996: 220

Nilai keluaran kontroler *m(t)* sebanding dengan laju kesalahan $\frac{de(t)}{dt}$, hubungan ini dapat ditulis sebagai:

$$m(t) = Kd\,\frac{de(t)}{dt} \qquad (2.4)$$

Kontroler diffrensial akan memberikan sinyal kendali keluaran m(t) = 0, untuk sinyal kesalahan yang konstan sehingga kontroler differensial tidak mempengaruhi keadaan mantap. Kontroler differensial digunakan untuk memperbaiki atau mempercepat respon *transient* sebuah sistem serta dapat meredam osilasi.

Pada penggunaan kontroler diferensial harus diperhatikan karakteristiknya, yaitu ketika kontroler tersebut diterapkan pada suatu sistem. Karakteristik kontroler differensial yang harus diperhatikan yaitu:

1. Kontroler ini tidak dapat menghasilkan keluaran bila tidak ada perubahan pada masukannya.
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan kontroler tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
3. Kontroler differensial mempunyai suatu karakter untuk mendahului, sehingga kontroler ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi kontroler differensial dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat korektif, dan cenderung meningkatkan stabilitas sistem (Ogata, 1997: 240).

Berdasarkan karakteristik kontroler tersebut, kontroler differnsial umumnya dipakai untuk mempercepat respon awal suatu sistem, tetapi tidak memperkecil kesalahan pada keadaan tunaknya. Kerja kontroler differensial hanyalah efektif pada lingkup yang sempit, yaitu pada periode peralihan. Oleh sebab itu kontroler differensial diferensial tidak pernah digunakan tanpa ada kontroler lain pada sebuah sistem.

Dari ketiga aksi kontrol dasar diatas, dapat dibuat kombinasi dari ketiganya, yaitu:

1. Kontroler Proporsional Integral (PI)

Aksi kontrolnya dinyatakan dalam persamaan:

$$m(t) = Kp\ e(t) + \frac{Kp}{Ti}\int_0^t e(t)\ dt \qquad (2.5)$$

Kontroler ini menghasilkan olahan sinyal kesalahan $\int_0^t e(t)\ dt$ kemudian ditambahkan dengan olahan sinyal kesalahan *e(t)*.

2. Kontroler Proporsional Differensial (PD)

Aksi kontrolernya dapat dinyatakan sebagai:

$$m(t) = Kp\ e(t) + Kp\ Td \frac{de(t)}{dt} \qquad (2.6)$$

Kontroler PD selalu mengukur kemiringan (*slope*) sinyal kesalahan $\frac{de(t)}{dt}$ dan memperkirakan besar *Mp* yang akan terjadi serta memberikan koreksi sebelum terjadi lawatan sebenarnya, sehingga diperoleh *maximum overshoot* yang kecil.

Jika kesalahan keadaan mantap tidak berubah terhadap waktu maka turunannya terhadap waktu sama dengan nol, sehingga kontroler PD tidak mempunyai pengaruh terhadap kesalahan keadaan mantap, tetapi jika terdapat perubahan kesalahan, kontroler PD akan mengurangi besar kesalahan keadaan mantap. Jadi kontroler PD digunakan untuk memperbaiki suatu sistem pengendalian yang tanggapan peralihannya mempunyai *maximum overshoot* yang berlebihan, tanpa memperhitungkan kecepatan responnya.

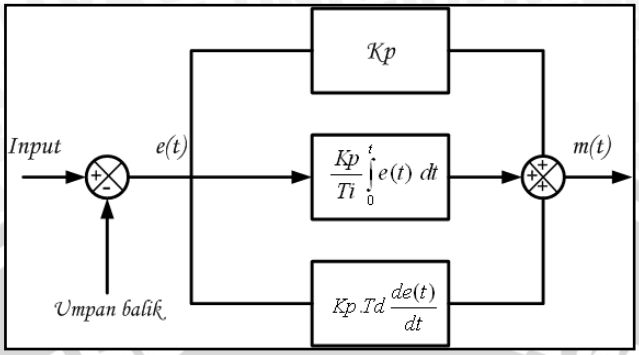
3. Kontroler Proporsional Integral Differensial (PID)

Setiap kekurangan dan kelebihan dari masing-masing kontroler P, I, dan D dapat saling menutupi dengan menggabungkan ketiganya secara pararel menjadi kontroler Proporsional Integral Differensial (PID). Elemen-elemen kontroler P, I, dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar (Guterus, 1994: 8-10)

Aksi kontrolnya dinyatakan sebagai:

$$m(t) = Kp\ e(t) + \frac{Kp}{Ti} \int_0^t e(t)\ dt + Kp\ Td \frac{de(t)}{dt} \qquad (2.7)$$

Jenis kontroler ini digunakan untuk memperbaiki kecepatan respon, mencegah terjadinya kesalahan keadaan mantap, serta mempertahankan kestabilan.



Gambar 2.9 Diagram Blok Kontroler Proporsional, Integral, Differensial
Sumber: Philips, 1996: 220

Keluaran kontroler PID merupakan jumlahan dari keluaran kontroler proporsional, keluaran kontroler integral, dan kontroler differensial. Gambar 2.11 menunjukkan hubungan tersebut. Karakteristik kontroler PID sangat dipengaruhi oleh kontibusi dari ketiga parameter P, I, dan D. Penyetelan konstanta Kp, Ti, dan Td akan mengakibatkan penonjolan sifat dari masing-masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat disetel lebih menonjol dibanding yang lainnya. Konstanta yang menonjol itulah akan memberi konstribusi pengaruh pada respon sistem secara keseluruhan (Gunterus, 1994: 8-10)

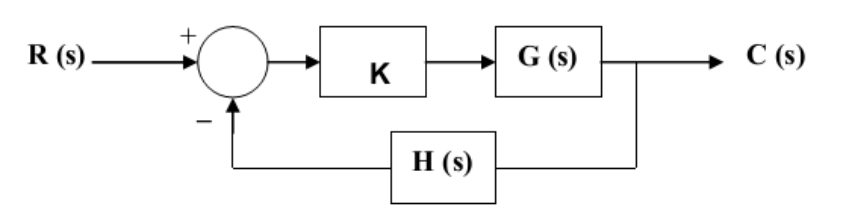
Persamaan (2.19) merupakan persamaan dalam kawasan waktu *continuous* (analog). Sedangkan agar persamaan tersebut dapat direalisasikan dalam bentuk pemrograman, maka persamaan dalam kawasan waktu *continuous* tersebut harus didiskritasi terlebih dahulu (kawasan *digital*). Berikut adalah diskritisasi menggunakan metode numerik *rectangular* mundur (*backward rectangular*) dengan Tc merupakan waktu sampling atau waktu cuplik (*Sampling Time)*.

$$m(t) = Kp\ e(t) + KiTc[e(t-1) + e(t)] + Kd\ \frac{e(t-1) + e(t)}{Tc} \qquad (2.8)$$

2.4.2 *Root Locus*

Root Locus (tempat kedudukan akar) merupakan suatu analisis yang menggambarkan pergeseran letak pole-pole suatu sistem close loop dari perubahan besarnya penguatan open loop dengan gain adjustment. Analisis ini digunakan sebagai salah satu dasar untuk mendesain suatu sistem kendali sesuai dengan karakteristik dan spesifikasi yang diinginkan. Analisis root locus ini dapat menentukan apakah suatu system stabil atau tidak. Selain itu dapat menentukan besarnya rentang penguatan open loop, agar suatu system masih dapat dikatakan stabil (tetapi tidak bisa menstabilkan suatu system tidak stabil secara utuh menjadi system yang stabil). Plot kurva root locus berada pada bidang-s (domain frekuensi).

Tempat Kedudukan Akar sebuah sistem merupakan kurva atau tempat kedudukan dari akar-akar persamaan karakteristik (pole–pole dari fungsi alih lingkar tertutup) dengan parameter gain (K) yang berubah – ubah.



Gambar 2.10 Blok diagram *root locus*

Dari gambar 2.10, persamaan karakteristik sistem dinyatakan dengan

$$1 + KG(s)\,H(s) = 0 \qquad (2.9)$$

Nilai s berada pada TKA jika s memenuhi persamaan di atas. Karena s dapat merupakan bilangan kompleks, maka dari persamaan tersebut, s adalah sebuah titik pada TKA jika memenuhi syarat magnitude.

$$|K| = \frac{1}{|G(s)\,H(s)|} \qquad (2.10)$$

Dengan syarat sudut

$$\angle G(s)\,H(s) = r\,180^{o}\text{, dengan } r = \pm 1, \pm 3, \pm 5, ... \qquad (2.11)$$

2.4.2.1 Menggambar *Root Locus* dengan Manual

Dari kedua syarat tersebut, diturunkan aturan-aturan menggambarkan *root locus* sebagai berikut :

1. *Root locus* mempunyai sifat simetri terhadap sumbu nyata.
2. Menentukan pole-pole dan zero-zero dari fungsi alih lingkar terbuka sistem KG(s)H(s). TKA bermula dari pole-pole (untuk K=0) dan berakhir zero-zero (untuk K → tak hingga) termasuk zero-zero pada titik tak hingga.
3. Menentukan asimptot θ dan titik potongnya dengan sumbu nyata σ dapat dihitung dengan rumus :

   $$\theta = \frac{r\,180}{\alpha} \qquad (2.12)$$

   dimana ±1, ±3, ±5 … dan α banyaknya zero pada titik tak terhingga dan

   $$\sigma = \frac{\sum(\text{letak pole berhingga}) - \sum(\text{letak zero berhingga})}{\sum(\text{pole berhingga}) - \sum(\text{zero berhingga})} \qquad (2.13)$$

4. Menentukan daerah cakupan *root locus* pada sumbu nyata. Tempat Kedudukan Akar mencakup titik-titik pada sumbu nyata di sebelah kiri frekuensi kritis (pole-pole dan zero-zero) nyata yang berjumlah ganjil.
5. Menentukan titik pencar (titik pisah atau titik temu), yang terdapat diantara akar-akar

   $$N(s)D'(s) - N'(s)D(s) = 0 \qquad (2.14)$$

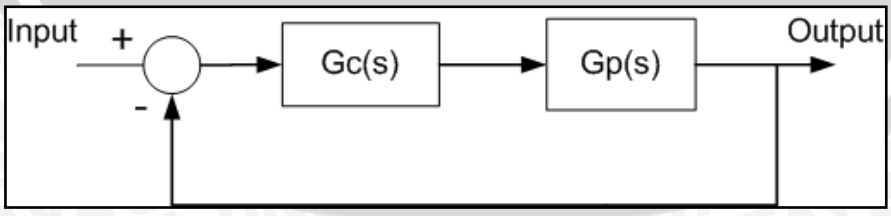
   dengan N(s) dan D(s) masing-masing merupakan numerator dan denumerator G(s)H(s).

Prosedur untuk mendapatkan kurva root locus:

a. menentukan open loop transfer function (OLTF) dan meletakkan pole-pole dan zero-zeronya pada bidang –s.
b. Menentukan interval terdapatnya root locus pada sumbu real. Bila interval daerah sumbu real mempunyai jumlah pole dan zero di sebelah kanannya bernilai ganjil, maka daerah tersebut terdapat root locus.
c. Menentukan jumlah asimtot, sudut asimtot, dan perpotongan asimtot dengan sumbu real.
d. Menentukan titik pencar dan temu pole-pole (break away point dan break in point)
e. Menentukan titik potong kurva root locus dengan sumbu imajiner (jika ada)
f. Menentukan sudut datang (untuk zero) dan berangkat (untuk pole)
g. Sketsa root locus dari data-data yang telah didapatkan.

2.4.3 Persamaan Perancangan PID dengan *Root Locus*

Rancangan sistem kendali loop tertutup menggunakan *root locus* memungkinkan untuk mengatur sekurang-kurangnya beberapa letak pole sistem loop tertutup sehingga dapat mengatur tanggapan transient pada tingkat tertentu dan pengaruhnya terhadap tanggapan keadaan mantap (Philips,1996: 209). Prosedur analitis perancangan kontroler PID menggunakan metode *root locus* yang dijelaskan dalam *Feedback Control System* oleh Charles L. Philips dan Royce D. Harbour dapat dilihat dalam gambar berikut.



Gambar 2.11 Sistem Kendali

Untuk sistem tersebut, persamaan karakteristik diberikan oleh

$$1 + Gc(s)Gp(s) = 0 \qquad (2.15)$$

Misalkan diinginkan lokus akar melalui $s = s_1$, maka

$$Gc(s_1)Gp(s_1) = -1$$
$$Gc(s_1)\left|Gp(s_1)\right|e^{j\psi} = 1e^{j\Pi}$$

Fungsi alih kontroler PID setelah ditransformasi laplace dinyatakan oleh

$$Gc(s) = Kp + \frac{Ki}{s} + Kd\, s$$

Sehingga dari persamaan diatas didapatkan

$$Gc(s_1) = \frac{1}{\left|Gp(s_1)\right|}e^{j(\Pi-\psi)}$$

atau

$$Kd\, s_1^2 + Kp\, s_1 + Ki = \frac{e^{j(\Pi-\psi)}}{\left|Gp(s_1)\right|} \qquad (2.16)$$

Dengan

$$s_1 = \left|s_1\right|e^{j\beta}$$

Maka

$$Kd\left|s_1\right|^2(\cos 2\beta + j\sin 2\beta) + Kp\left|s_1\right|(\cos\beta + j\sin\beta) + Ki$$

$$= \frac{\left|s_1\right|}{\left|Gp(s_1)\right|}\left[\cos(\beta + \Pi - \psi) + j\sin(\beta + \Pi - \psi)\right]$$

Menyamakan real dengan real dan imajiner dengan imajiner, didapat

$$\begin{bmatrix} |s_1|^2 & |s_1|\cos\beta \\ |s_1|^2 & |s_1|\sin\beta \end{bmatrix} \begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} \frac{|s_1|}{Gp(s_1)}\cos(\beta + \Pi + \psi) - Ki \\ \frac{|s_1|}{Gp(s_1)}\sin(\beta + \Pi + \psi) \end{bmatrix}$$

atau

$$\begin{bmatrix} |s_1|^2 & |s_1|\cos\beta \\ |s_1|^2 & |s_1|\sin\beta \end{bmatrix} \begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} -\frac{|s_1|}{Gp(s_1)}\cos(\psi - \beta) - Ki \\ \frac{|s_1|}{Gp(s_1)}\sin(\psi - \beta) \end{bmatrix} \quad (2.17)$$

Dari persamaan di atas dapat dilihat bahwa untuk perancangan kontroler PID, satu dari tiga penguatan Kp. Ki, Kd, harus ditentukan dahulu. Sedangkan untuk perancangan PI atau PD, penguatan yang sesuai pada persamaan diatas dibuat sama dengan nol.

Untuk kasus s_1 adalah imajiner, persamaan di atas akan menghasilkan dua persamaan dalam Kp dan Kd serta besar Ki harus ditentukan terlebih dahulu. (Philips, 1996:223)

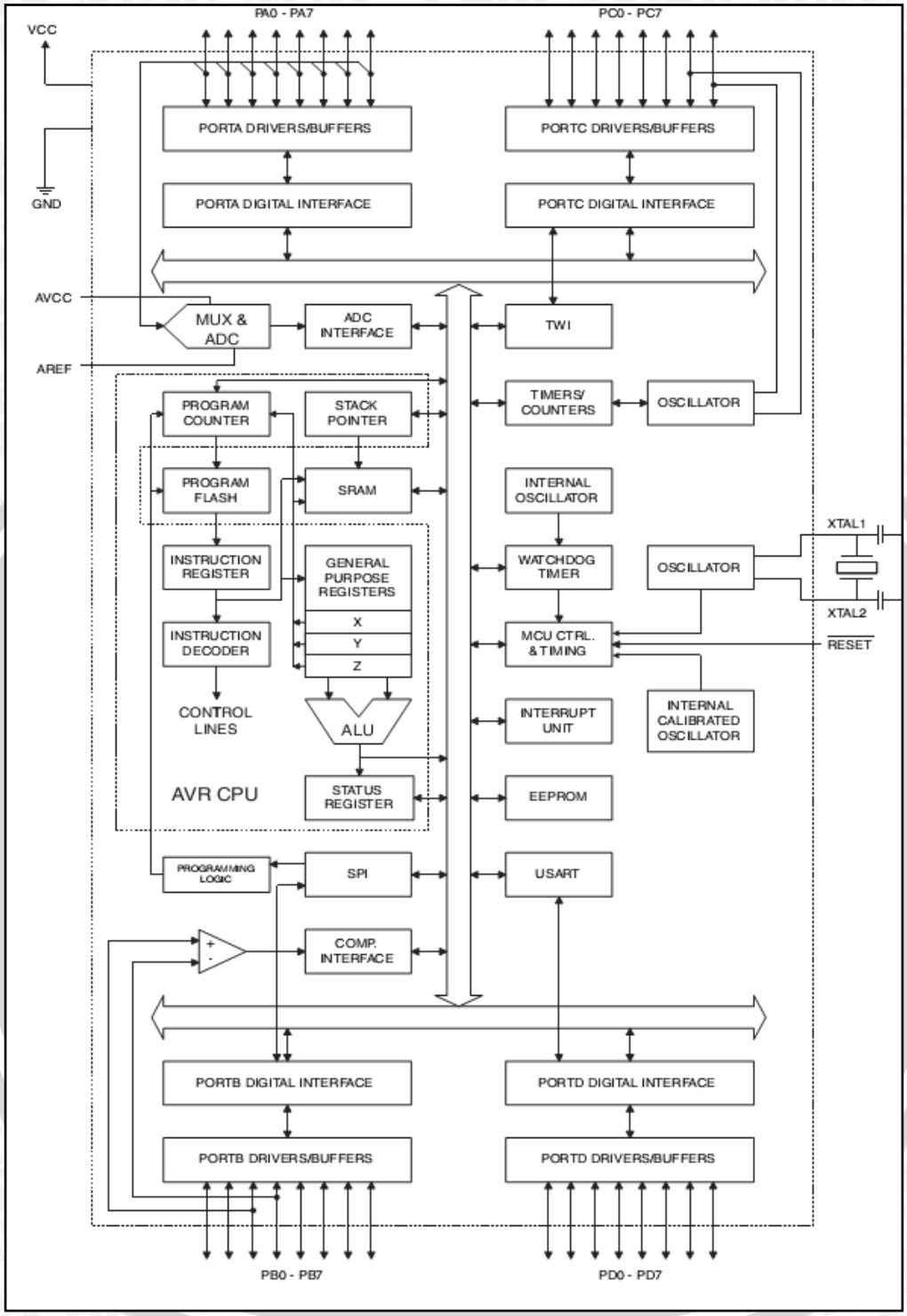
2.4.3 Mikrokontroler ATMega16

Mikrokontroler berfungsi sebagai pusat pengolahan data dan pengendali bagi perangkat lain seperti sensor SHT-11 dan modul GPS EG-T10. Untuk memenuhi kebutuhan memori program yang cukup besar, maka digunakan mikrokontroler ATmega16. Fitur-fitur yang yang dimiliki ATmega16 sebagai berikut:

- Mikrokontroler AVR 8 Bit yang memiliki kemampuan tinggi, dengan daya rendah.
- Memiliki kapasitas Flash memori 16 KByte, EEPROM 512 Byte dan SRAM 1KByte
- Saluran I/O sebanyak 32 buah, yaitu *Port* A, *Port* B, *Port* C dan *Port* D.
- CPU terdiri atas 32 register.

- Unit Interupsi internal dan eksternal.
- ADC internal dengan fidelitas 10 bit 8 channel.
- Sistem mikroprosesor 8 bit berbasis RISC dengan kecepatan maksimal 16 MHz.
- *Port* USART untuk komunikasi serial

Dengan fitur-fitur seperti diatas, pembuatan alat menggunakan ATmega16 menjadi lebih sederhana dan tidak memerlukan IC pendukung yang banyak. Agar lebih jelasnya dapat dilihat pada gambar dibawah ini:

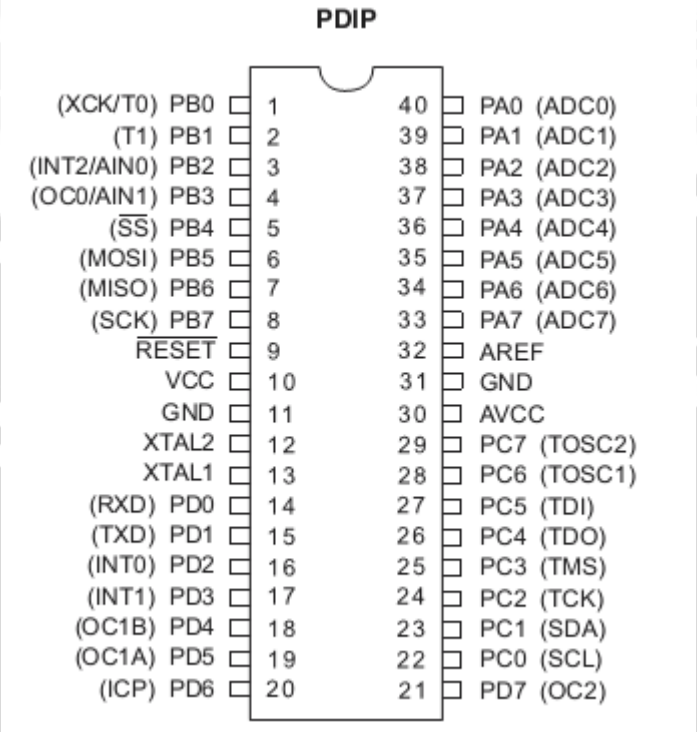


Gambar 2.12 Diagram Blok Mikrokontroler ATmega16

2.4.3.1 Deskripsi Pin ATMega16

ATmega16 memiliki 32 pin yang digunakan untuk *input/output*, pin-pin tersebut terdiri dari 8 pin sebagai *port* A. 8 pin sebagai *port* B. 8 pin sebagai *port*

C. 8 pin sebagai *port* D. Dalam komunikasi serial, maka hanya *port* D yang dapat digunakan kerena fungsi khusus yang dimilikinya. Untuk lebih jelas akan ditunjukan pada tabel-tabel fungsi khusus *port*. Susunan pin Mikrokontroler ATmega16 diperlihatkan pada gambar dibawah ini:



Gambar 2.13 Konfigurasi Pin ATmega16

- Pin 1 sampai 8 (*Port* B) merupakan *port* pararel 8 bit dua arah *(input/output)* dan pin fungsi khusus.

Tabel 2. 2 Fungsi Khusus *Port* B ATmega16

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	SS (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

- Pin 9 (Reset) merupakan pin yang digunakan untuk mereset mikrokontroler.

- Pin 10 (VCC) merupakan pin masukan catu daya.
- Pin 11 & 31 (GND) merupakan pin ground.
- Pin 12 (XTAL2) & Pin 13 (XTAL1) merupakan pin masukan clock ekstenal.
- Pin 14 sampai 21 (*Port* D) merupakan *port* pararel 8 bit dua arah *(input/output)* dan pin fungsi khusus.

Tabel 2. 3 Fungsi Khusus *Port* D ATmega16

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

- Pin 22 sampai 29 (*Port* C) merupakan *port* pararel 8 bit dua arah *(input/output)* dan pin fungsi khusus

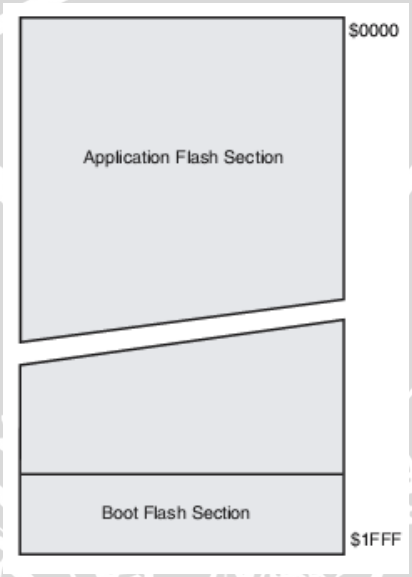
Tabel 2. 4 Fungsi Khusus *Port* C ATmega16

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

- Pin 30 (AVCC) merupakan pin masukan tegangan untuk ADC.
- Pin 32 (AREF) merupakan pin masukan tegangan referensi ADC.
- Pin 33 sampai 40 (*Port* A) merupakan pin I/O dua arah dan pin masukan ADC.

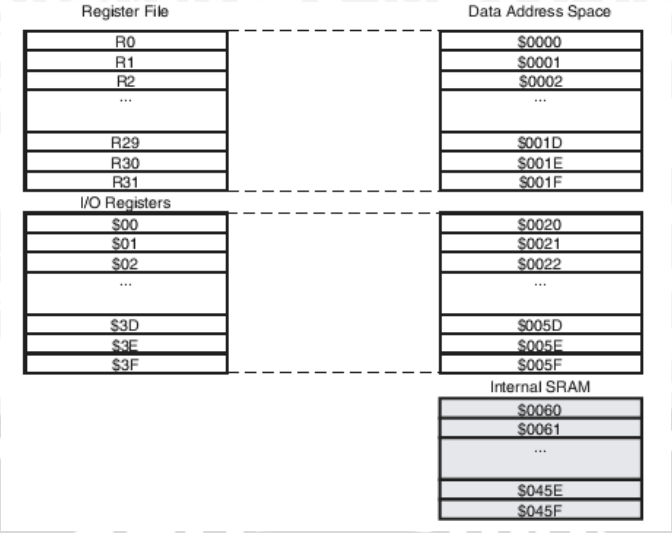
2.4.3.2 Peta Memori ATMega16

Arsitektur AVR mempunyai dua memori utama, yaitu memori data dan memori program. Untuk penyimpanan data program, ATmega16 memiliki memori EEPROM sebesar 512 Byte. Selain itu, ATmega16 terdiri atas 16 Kbyte *Onchip In-System Reprogrammable Flash memor*y untuk menyimpan program. Berikut gambar peta memori ATmega16.



Gambar 2.14 Peta Memori Program AVR ATmega16

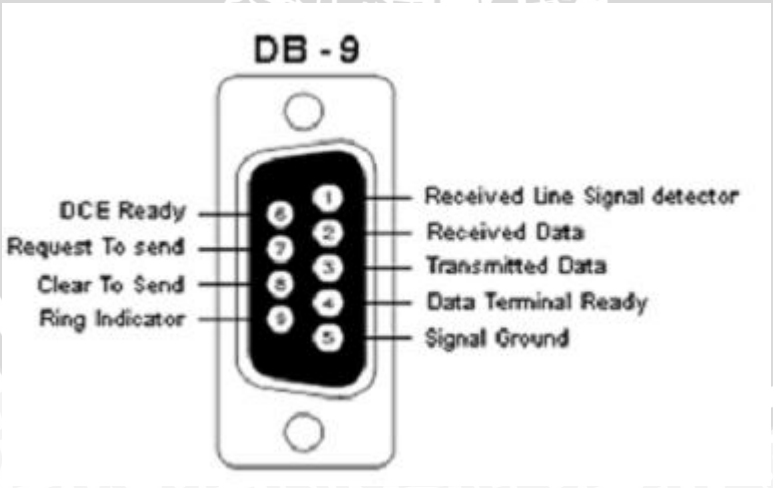
AVR ATmega16 mempunyai memori data yang terbagi menjadi 3 bagian, yaitu 32 register umum, 64 register I/O dan 1 Kbyte SRAM internal. General Purpose register menempati alamat data paling bawah, yaitu $00 sampai $1F. sedangkan memory I/O menempati 64 alamat berikutnya mulai dari $20 hingga $5F. 1024 memory berikutnya digunakan untuk SRAM yaitu pada alamat $60 hingga $45. Berikut gambar konfigurasi memory data ATmega16.



Gambar 2.15 Konfigurasi Memori Data AVR ATmega 16

2.4.3.3 Komunikasi Serial USART

Pada dasarnya *port* serial komputer menggunakan level RS-232, RS (Recommended Standard) dikeluarkan oleh EIA (*Electronics Industry Association*). Dalam logika RS-232 logika 1 dinyatakan sebagai Mark dengan level tegangan antara -3V dan -25V (negatif), sedangkan logika 0 dinyatakan sebagai *Space* dengan level tegangan antara 3V dan 25V (positif). Konektor DB-9 pada bagian belakang komputer PC adalah *port* serial RS-232 yang biasa dinamai COM1 atau COM2.



Gambar 2.16 Konektor DB-9 Pada PC

Tabel 2. 5 Konfigurasi Pin RS-232

Nomer Pin	Nama Sinyal	Direction	Keterangan
1	DCD	IN	Receive Signal Ditect
2	RXD	IN	Receive Data
3	TXD	OUT	Transmit Data
4	DTR	OUT	Data Terminal Ready
5	GND	-	Ground
6	DSR	IN	Data set Ready
7	RST	OUT	Request to Send
8	CTS	IN	Clear to Send
9	RI	IN	Ring Indicator

Keterangan mengenai saluran RS-232 pada konektor DB9 adalah sebagai Berikut:

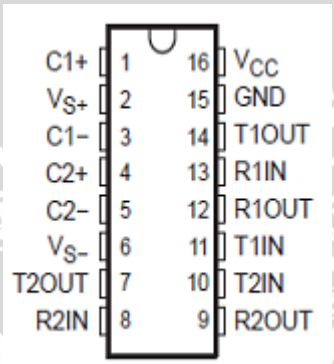
1. *Received Line Signal Detect*, dengan saluran ini DCE memberitahukan ke DTE bahwa terminal masukan ada data masukan.
2. *Received Data*, digunakan DTE menerima data dari DCE.
3. *Transmite Data*, digunakan DTE mengirimkan data ke DCE.
4. *Data Terminal Ready*, pada saluran ini DTE memberitahukan kesiapan sinyalnya.
5. *Signal Ground*, saluran Ground.
6. *Ring Indicator*, pada saluran ini DCE memberitahukan ke DTE bahwa stasiun menghendaki hubungan dengannya.
7. *Clear to Send*, dengan saluran ini DCE memberitahukan ke DTE boleh mengirimkan data.
8. *Request to Send*, dengan saluran ini DCE diminta mengirimkan data oleh DTE.
9. *DCE Ready*, sinyal aktif pada saluran ini menunjukan bahwa DCE sudah siap.

Dengan adanya perbedaan level tegangan TTL mikrokontroler dengan level tegangan RS-232 pada serial *port* komputer, maka *port* serial mikrokontroler

tidak bisa secara langsung dihubungkan dengan serial *port* komputer, oleh sebab itu diperlukan sebuah pengubah level logika dari TTL ke RS-232.

IC MAX232 dari *Maxim Incoporation* adalah IC pengubah level TTL menjadi RS-232 atau sebaliknya, yang memiliki sebuah *charge pump* yang bisa menghasilkan tegangan +10V dan -10V dari tegangan catu daya 5V. Tegangan-tegangan ini dihasilkan dengan proses pengisian dan pembuangan empat kapasitor luar yang dihubungkan dengan rangkaian pengganda tegangan internal yang dimiliki IC ini.

MAX232 mempunyai 2 penerima (RS-232 ke TTL) dan 2 pengirim (TTL ke RS232), cukup untuk menghubungkan pin TXD dan RXD mikrokontroler dengan serial *port* komputer. Konfigurasi Pin MAX232 sebagai berikut:



Gambar 2.17 Konfigurasi IC MAX232

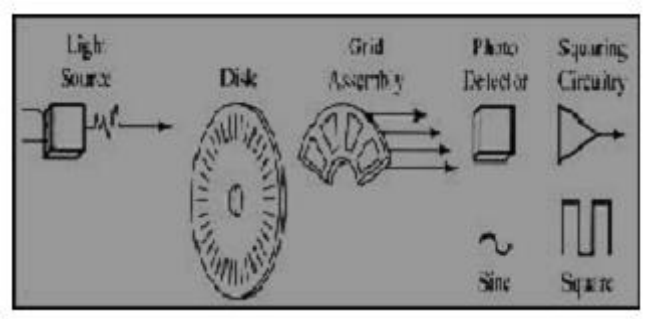
Pada umumnya hanya 1 pengirim dan 1 penerima yang dipakai, baik untuk level RS232 atau TTL. Pin yang digunakan diantaranya Pin 11 (T1IN) sebagai *Input* (TTL), Pin 12 (R1OUT) sebagai *Output* (TTL), Pin 14 (T1OUT) sebagai Input (RS-232) dan Pin 13 (R1IN) digunakan sebagai *Output* (RS-232).

2.5 Studi Sensor

Sensor *Rotary encoder*

Rotary encoder adalah elektromekanik yang dapat mendeteksi atau memonitor gerakan dan posisi. *Rotary encoder* biasanya menggunakan sensor optik untuk menghasilkan serial pulsa yang dapat dijadikan gerakan, posisi, dan arah. Sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh *rotary encoder*.

Rotary encoder tersusun dari suatu piringan tipis yang memiliki lubang-lubang pada bagian lingkaran piringan. LED ditempatkan pada salah satu sisi piringan sehingga cahaya akan menuju ke piringan. Di sisi yanglain suatu photo-transistor diletakkan sehingga photo-transistor ini dapat mendeteksi cahaya dari LED yang berseberangan. Piringan tipis tadi dikopel dengan poros motor berputar lainnya yang ingin kita ketahui posisinya, sehingga ketika motor berputar piringan juga akan ikut berputar. Apabila posisi piringan mengakibatkan cahaya dari LED dapat mencapai photo-transistor melalui lubang-lubang yang ada, maka photo-transistor akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi. Semakin banyak deretan pulsa yang dihasilkan pada satu putaran menentukan akurasi *rotary encoder* tersebut, akibatnya semakin banyak jumlah lubang yang dapat dibuat pada piringanmenentukan akurasi *rotary encoder* tersebut.



Gambar 2.18 Blok Penyusun Rotary Encoder

Rangkaian penghasil pulsa (gambar di atas) yang digunakan umumnya memiliki output yang berubah dari +5V menjadi 0.5V ketika cahaya diblok oleh piringan dan ketika diteruskan ke photo-transistor. Karena *device* ini umumnya bekerja dekat dengan motor DC maka banyak *noise* yang timbul sehingga biasanya output akan dimasukkan ke *low-pass filter* dahulu. Apabila *low-pass filter* digunakan, frekuensi *cut-off* yang dipakai umumnya ditentukan oleh jumlah slot yang ada pada piringan dan seberapa cepat piringan tersebut berputas, dinyatakan dengan :

$$f_c = \frac{s_w n}{60} \qquad (2.12)$$

f_c = Frekuensi *cut-off filter*, s_w adalah kecepatan piringan dan n adalah jumlah *slot* pada piringan.

2.6 Motor DC

Motor DC yagn digunakan dalam penelitian adalah motor DC yang menggunakan motor magnet permanen. Alasan pemilihan Motor DC ini adalah kemudahan dalam pengontrolan dengan menggunakan pengatur tegangan DC. Medan stator motor jenis ini dihasilkan oleh magnet permanen bukan electromagnet. Motor DC mempunyai kurva kecepatan torsi yang linier dalam janga yang lebar. Penggunakan magnet permanen tidak membutuhkan daya listrik untuk menghasilkan medan stator, sehingga daya dan pendinginan yang diperlukan lebih rendah dibandingkan motor yang menggunakan electromagnet. Perubahan kecepatan motor dapat diatur dengan cara mengubah-ubah besarnya tegangan DC yang diberikan. Motor DC memiliki beberapa keunggulan, yaitu:

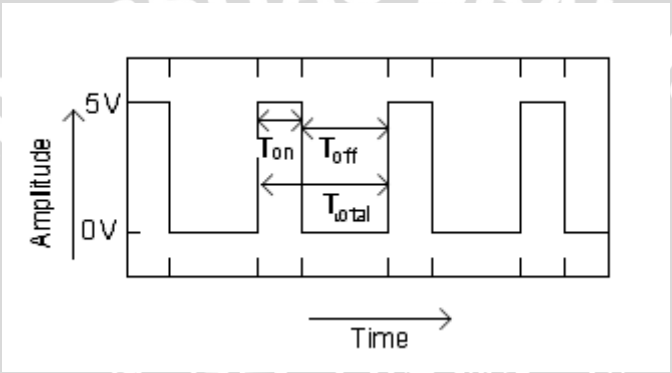
1. Bentuknya kompak, ringan dan berdaya tinggi.
2. Dapat bekerja pada daerah atau tempat yang mempunyai lingkungan ekstrem
3. Biaya perawatan mudah

Motor DC hampir sama konstruksinya dengan motor AC, perbedaanya terletak pada sikat dan cincin belah (komutator). Saat siklus pertama, arus mengalir dari kutub positif ke negative. Aliran arus yang melewati bagian kabel yang berada di dekat kutub N magnet akan menimbulkan gaya Lorentz ke bawah. Sementara itu aliran arus yang melewati kabel yang berada di dekat kutub S magnet akan menyebabkan gaya Lorenzt ke atas. Kedua perpaduan gaya Lorenzt tersebut akan menyebabkan kawat berputar. Pada siklus berikutnya terjadi hal yang serupa seperti siklus sebelumnya. Apabila arus terus-menerus dialirkan, maka kawat akan berputar secara terus menerus pula. Pada aplikasi sesungguhnya kawat adalah sebuah rotor yang akan dikopel dengan sebuah as dan akan memutar as tersebut terus-menerus seiring perputaran motor.

2.7 PWM (*Pulse Width Modulation)*

PWM (*Pulse Width Modulation*) digunakan untuk mengatur kecepatan dari motor DC. Dimana 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%-10%. Dengan mengatur *duty cycle* akan diperoleh keluaran yang diinginkan. Pada AVR ATMega16 *duty cycle* ditentukan oleh *Output Compare Match A (*OC1A). Sinyal PWM (*pulse width modulation*) secara umum dapat dilihat dalam gambar di bawah ini.



Gambar 2.19 Sinyal PWM Secara Umum

$$Duty\ Cycle = \frac{T_{on}}{T} x\ 100\% \quad (2.13)$$

$$V_{DC} = Duty\ Cycle\ x\ Vcc \quad (2.14)$$

Sedangkan frekuensinya dapat ditentukan dengan rumus sebagai berikut:

$$fOCn = \frac{fclk\ \ I/O}{N.256} \quad (2.15)$$

Timer/counter yang digunakan pada PWM ini yaitu *Timer/Counter0* (8 bit) dengan metode Fast PWM dan *Prescaler factor (N)* yaitu 256.

2.8 Pengambilan Data Input-Output

Langkah awal dalam melaksakan identifikasi sistem adalah pengambilan data *input-output*. Pengujian ini tentu memerlukan sinyal uji tertentu yang akan diberikan kepada sistem fisik yang akan diidentifikasi. Agar diperoleh model yang tepat maka dalam pemilihan sinyal uji ini tidak boleh sembarangan. Syarat

pemilihannya adalah suatu sinyal uji harus memiliki cakupan frekuensi yang lebar dan standard yang digunakan adalah sinyal *Pseudo Random Binary Sequences* (PRBS). (Landau, 2006)

Pseudo Random Binary Sequence (PRBS) adalah sinyal kotak yang termodulasi pada lebarnya dan berlangsung secara sekuensial. Sinyal ini biasanya dibangkitkan menggunakan *Linear Feedback Shift Register* (LFSR). Pada LFSR memiliki 2 parameter dasar yang menentukan sifat sekuensial yang dihasilkan, yaitu: panjang dari *shift register* dan susunan umpan balik. PRBS memiliki variasi panjang sekuensialnya, tergantung dari panjangnya *shift register* seperti ditunjukkan Tabel 2.6

Tabel 2. 6 Tabel Variasi Panjang Sekuensial PRBS

Panjang Register (N)	Panjang Sekuensial $L=2^N-1$	Posisi Tap Umpan Balik
2	3	1 dan 2
3	7	1 dan 3
4	15	3 dan 4
5	31	3 dan 5
6	63	5 dan 6
7	127	4 dan 7
8	255	2, 3, 4, dan 8
9	511	5 dan 9
10	1023	7 dan 10

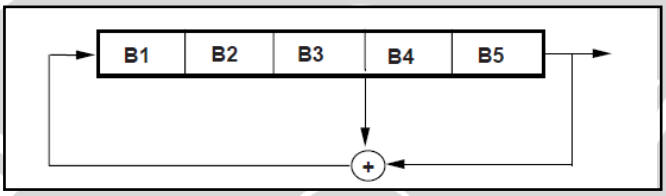
Sumber: Landau, 2006

Panjang dari *shift register* menentukan periode maksimum yang dapat dihasilkan dari sekuensial PRBS yan tidak berulang dan dapat dinyatakan dengan persamaan:

$$L_{PRBS}=2^n-1 \quad \text{(2.16)}$$

Dimana n adalah panjang dari register LFSR (jumlah bit). Panjang maksimum dari PRBS disebut M-*sequence*.

Panjang maksimum yang dapat dihasilkan PRBS untuk panjang tertentu dari *shift register* dapat dicapai dengan mempersiapkan konfigurasi dari *feedback*. Pada dasarnya ada 2 kemungkinan untuk realisasi LFSR, yaitu: Fibonacci (*many to one*) dan Galois (*one to many*). Keduanya dapat didasarkan pada gerbang XOR atau XNOR menggunakan bermacam-macam angka dan kombinasi dari *feedback*-taps-keluaran dari *shift register* khusus. Dengan mengubah konfigurasi umpanbalik (jumlah tap dan posisinya) memungkinkan untuk menemukan M-*sequence* yang berbeda untuk panjang tertentu dari *shift register*.



Gambar 2.20 Register Geser 5 Bit dengan Umpan Balik
Sumber: Landau, 2006

Prinsip pengujian proses dengan sinyal PRBS adalah membuat perubahan input kecil secara acak untuk membangkitkan gangguan (*perturbation*) yang kontinyu pada variabel *output*. Salah satu keuntungan penggunaan dari pendekatan ini adalah amplitudo perubahan input yang dibutuhkan dapat lebih kecil jika dibandingkan dengan perubahan *step* pada *step testing*. Selain itu, proses pengujian dapat dilakukan tanpa harus menunggu proses dalam keadaan tunak (*steadystate*). Jika pengujian dengan sinyal PRBS dilakukan, sinyal input secara teoritis disebut *white* (tidak berkorelasi) dan akan menghasilkan parameter model estimasi yang lebih baik. Frekuensi dari PRBS dapat dipilih untuk putaran (*flips*) cepat (*fast*) atau lambat (*slow*). Pemilihan frekuensi ini dapat menentukan jenis informasi terbaik yang akan didapat, misalnya untuk *fast* akan memberikan informasi yang akurat mengenai *deadtime*, *slow* akan memberikan informasi *steadystate gain* yang tepat sedangkan *medium* memberikan informasi *time constant* lebih baik. Dalam merancang PRBS perlu diperhatikan beberapa hal:

1. Menentukan Ukuran PRBS

Untuk mengidentifikasi secara tepat penguatan *steady state* dari model dinamis plant, durasi paling tidak, 1 dari pulsa harus lebih besar dari *rise time* t_R

dari plant (termasuk *time delay*). Durasi maksimum (T_{im}) dari pulsa adalah $N.T_s$, kondisi tersebut menyebabkan:

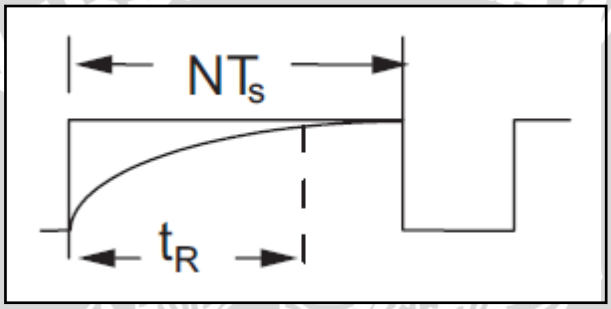
$$T_{im}=NT_s > t_R \quad \text{..... (2.17)}$$

Dengan: T_{im} = Durasi maksimum

N = Panjang register

T_s = Periode sampling

Ditunjukkan oleh Gambar 2.4:



Gambar 2.21 Lebar Pulsa Sinyal PRBS

Sumber: Landau, 2006

Agar seluruh jangkauan spektrum frekuensi yang dihasilkan oleh sinyal PRBS dapat terpakai, maka lamanya percobaan minimal kurang atau sama dengan panjang sekuensialnya. Pada kebanyakan kasus, durasi dari percobaan (L) yang dipilih adalah sama dengan panjangnya sekuensial, jika tidak maka harus dipastikan bahwa

$$2^{N-1}T_s<L \text{ ; L=durasi percobaan} \quad \text{..... (2.18)}$$

Jika proses mengalami *noise* yang tinggi dan pergerakan *input* terlalu kecil sehingga berada dalam daerah *bandwith noise*, maka periode pengujian harus lebih lama untuk mendapatkan kualitas model yang lebih baik

2. Memilih Magnitude dari PRBS

Magnitude yang dihasilkan oleh PRBS mungkin akan sangat kecil, namun dia harus lebih besar daripada amplitudo dari gangguan (*noise*). Jika perbandingan magnitudo sinyal dengan *noise* terlalu kecil maka penting untuk menambah waktu pengujian agar diperoleh estimasi parameter yang lebih baik. (Landau, 2006)

Dengan memilih amplituda yang cukup besar dapat dilihat efek dari pergerakan data, tetapi juga jangan terlalu besar sehingga dapat menyebabkan proses *upset* (di atas *setpoint*) dan operator harus mengkompensasi variabel yang lain.

BAB III
METODOLOGI PENELITIAN

Dalam pencapaian tujuan skripsi ini dibutuhkan metodologi penelitian dalam pelaksanaannya, berikut ini adalah metodologi yang digunakan dalam penelitian:

3.1 Studi Literatur

Literatur yang dibutuhkan adalah dasar teori yang berhubungan dengan alat yang akan dirancang, yaitu sebagai berikut:

1. Visual Basic 6.0
2. Motor DC
3. Sensor *Rotary Encoder*
4. Mikrokontroler ATMega16

3.2 Perancangan Alat

Pada tahap perancangan ini dibuat terlebih dahulu blok diagram fungsional dari rangkaian yang direncanakan baik itu dari perancangan perangkat keras dan perangakat lunak. Hal ini dilakukan supaya memudahkan dalam proses pembuatan alat. Perancangan terbagi menjadi 2 bagian besar perancangan perangkat lunak dan perancangan perangkat keras.

Adapun secara garis besar perancangan alat dilakukan sebagai berikut:

1. penentuan spesifikasi alat.
2. pembuatan diagram blok sistem keseluruhan.
3. perancangan mekanik.
4. perancangan perangkat elektrik yang terdiri dari mikrokontroler dan rangkaian sensor.
5. Perancangan perangkat lunak menggunakan Visual Basic 6.0

3.2.1 Pembuatan Perangkat Keras

Pembuatan alat dengan menggunakan komponen elektronika yang telah direncanakan, perancangan perangkat keras terbagi menjadi tiga bagian besar diantaranya:

1. pembuatan *power supply*.
2. pembuatan minimum sistem dan driver motor.
3. pembuatan rangkaian motor beserta sensor *rotary encoder*.

3.2.2 Pembuatan Perangkat Lunak

Pembuatan perangkat lunak terbagi menjadi dua bagian yaitu:

1. pembuatan perangkat lunak untuk Mikrokontroler. Menggunakan CV-AVR.
2. Pembuatan perangkat lunak untuk komputer menggunakan Visual Basic 6.0.

3.3 Pengujian Alat

Untuk mengetahui apakah alat yang telah dibuat sesuai dengan yang direncanakan maka perlu dilakukan pengujian rangkaian. Pengujian yang dilakukan terbagi menjadi tiga, yaitu:

1. Pengujian perangkat keras.

   Perangkat keras yang telah dibuat tahap demi tahap akan diuji satu persatu sesuai blok diagram. Pengujian perangkat keras ini meliputi pengujian rangkaian *power supply* menggunakan multimeter. Pengujian sensor *rotary encoder*, Pengujian driver motor dan pengujian motor DC.
2. Pengujian perangkat lunak

   Perangkat lunak dalam hal ini yang telah dibuat dengan Visual Basic 6.0 terlebih dahulu apakah komunikasi serial bisa dilakukan dan dapat menerima serta mengiri data melalui port serial yang telah ditentukan.
3. Pengujian secara keseluruhan

   Pengujian secara keseluruahn dilakukan dengan cara menghubungkan tiap tiap perangkat keras sesuai dengan blok diagram dan menjalankan

perangkat lunaknya. Kemudian di dalam prangkat lunak yang sudah berhasil terhubung dengan minimum sistem dilakukan pengesetan *set point, setting parameter* . Hasil pengesetan tersebut dapat dilihat secara langsung di perangkat lunak yang telah terbuat beserta kecepatan saat ini dan nilai error.

3.4 Pengambilan Kesimpulan dan Saran

Tahap terakhir dari penelitian ini adalah pengambilan kesimpulan dan saran. Kesimpulan diperoleh dari hasil pengujian alat dan kesesuaiannya dengan teori yang telah dipelajar. Sedangkan saran diberikan untuk memperbaiki kesalahan dan pengembangan alat supaya lebih baik untuk penelitian selanjutnya.

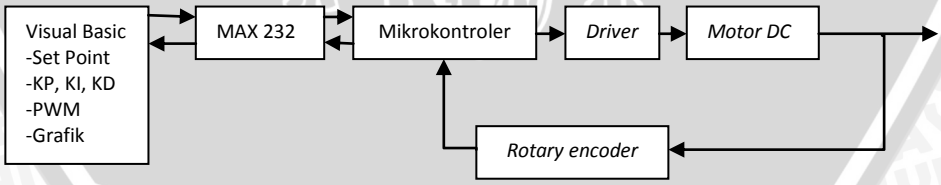
BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Dalam bab ini akan menjelasakan spesifikasi alat, perancangan perangkat keras dan perangkat lunak dari pengontrol kecepatan motor DC dengan metode PID menggunakan mikrokontroler ATmega16 dan Visual Basic 6.0. Tahap-tahap perancangan adalah sebagi berikut:

- Perancangan blok diagram sistem secara keseluruhan.
- Perancangan perangkat keras
  - Catu daya sistem.
  - Rangkaian *driver* motor.
  - Rangkaian *minimum system* ATmega16.
  - Rangkaian Max 232.
  - Rangkaian *rotary encoder*.
- Perancangan perangkat lunak
  - Pemrograman mikrokontroler ATmega16 menggunakan CVAVR.
  - Pemrograman Visual Basic 6.0.

4.1 Perancangan Keseluruhan Sistem.

Dalam Skripsi ini alat yang akan dibuat sesuai dengan blog diagram yang ditunjukkan oleh gambar 4.1.



Gambar 4.1 Blok Diagram Keseluruhan Sistem.

Program Visual Basic 6.0 yang dibuat berfungsi sebagai *interface* pengontrolan pada pengguna, dilengkapi dengan pembacaan RPM dalam bentuk angka dan grafik sehingga bisa dilihat dengan jelas respon dari motor DC. Selain itu pengguna bisa memberikan nilai parameter PID dari hasil tuning atau pun trial

error, dalam skripsi ini tuning motor dilakukan menggunakan sinyal *input Pseudo Random Binary Sequence* (PRBS). Hasilnya dianalisa menggunakan pada MATLAB 7.0.4.

Penghubung antara program Visual Basic 6.0 menggunakan rangkaian Serial MAX 232, rangaian ini berfungsi untuk menngkonversi sinyal dari Visual Basic 6.0 supaya sesuai dengan sinyal yang dibutuhkan Mikrokontroler.

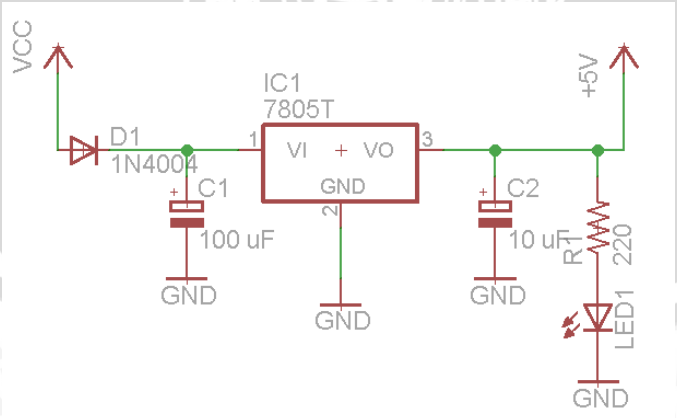
Mikrokontroler ATmega16 berfungsi sebagai penghitung nilai RPM motor, mengirimkan sinyal PWM pada driver dan mengirimkan informasi RPM kepada program Visual Basic 6.0.

4.2 Perancangan Perangakat Keras

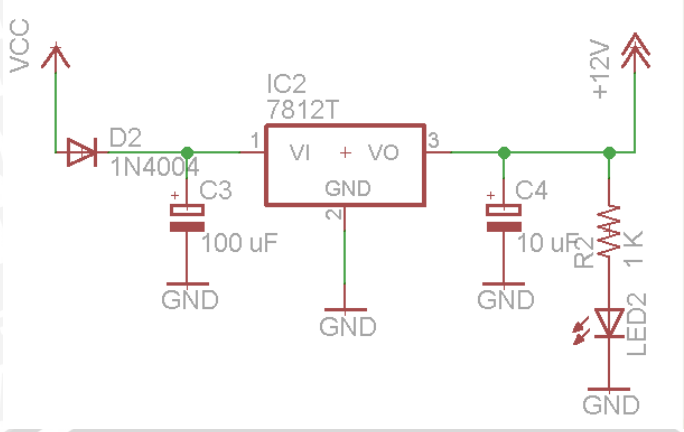
4.2.1 Perancangan Catu Daya Sistem

Pembuatan alat dalam skripsi ini membutuhkan 2 jenis besaran tegangan yaitu 5 volt untuk mengaktifkan rangkaian mikrokontroler dan sensor *rotary encoder,* sedangkan sebagai penggerak motor DC menggunakan catu daya sebesar 12 volt.

Dalam pembuatan catu daya yang diperlukan, digunakan rangkaian penyearah gelombang penuh, menggunakan 2 diode dan tranformator 1 amper jenis *Center Tap (CT).*



Gambar 4.2 Rangkaian Catu daya 5 volt

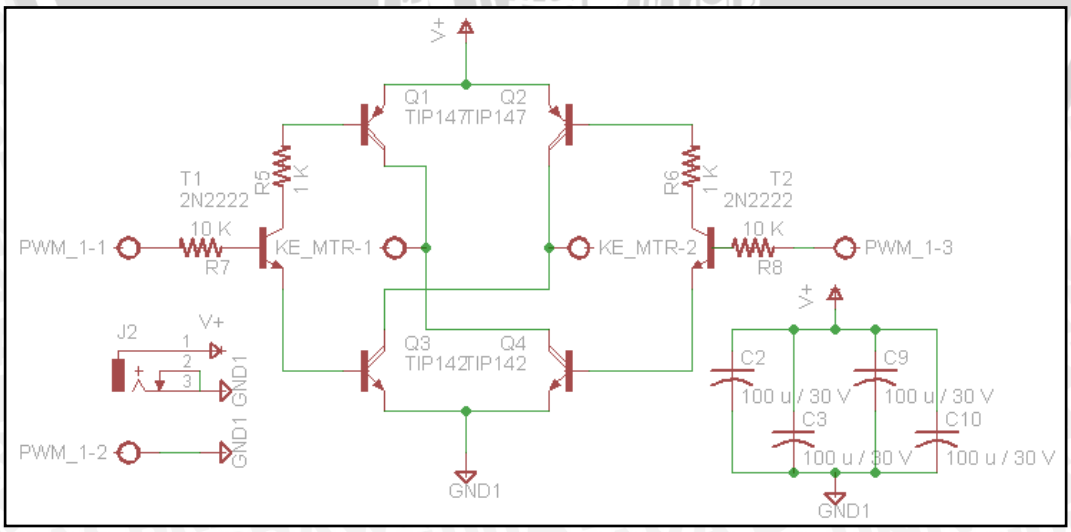


Gambar 4.3 Rangkaian Catu daya 12 volt

Rangkaian catu daya diatas menggunakan *Fixed Output Regulator*, sesuai dengan *datasheet* LM78XX. Regulator jenis LM7805 memiliki tegangan keluaran minimal 4,8 volt dan maksismal 5,2 volt sedangakan untuk LM7812 memiliki keluaran minimal 11,5 volt dan maksimal 12,5 volt.

4.2.2 Perancangan *Driver* Motor

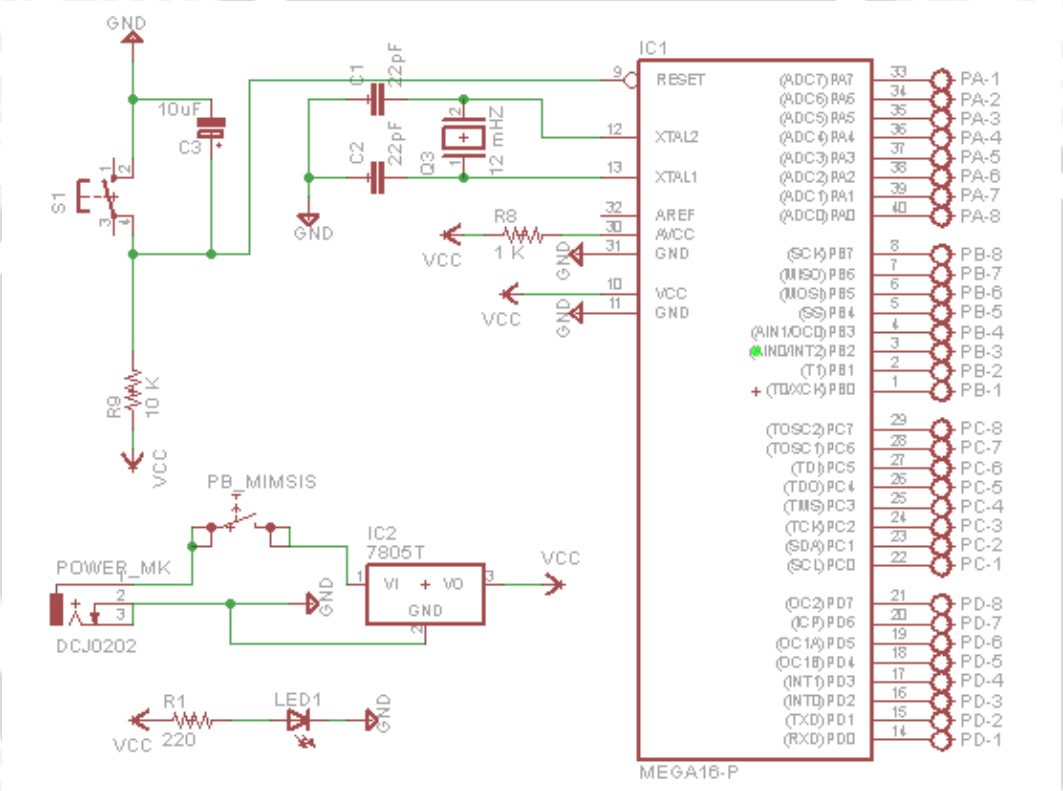
Driver motor yang digunakan dalam skripsi ini merupakan implementasi dari prinsip H-Bridge (jembatan H), menggunakan 2 Transisor TIP 147 PNP dan TIP 142 NPN sebagai penyusun jembatan H karena memiliki ketahahan yang tinggi, daya sampai 125 watt dan dapat mengalirkan alur sampai dengan 10 ampere. Untuk pencacah PWM nya menggunakan transistor 2n222 NPN.



Gambar 4.4 Rangkaian Driver Motor DC

4.2.3 Perancangan Minimum Sistem ATmega16

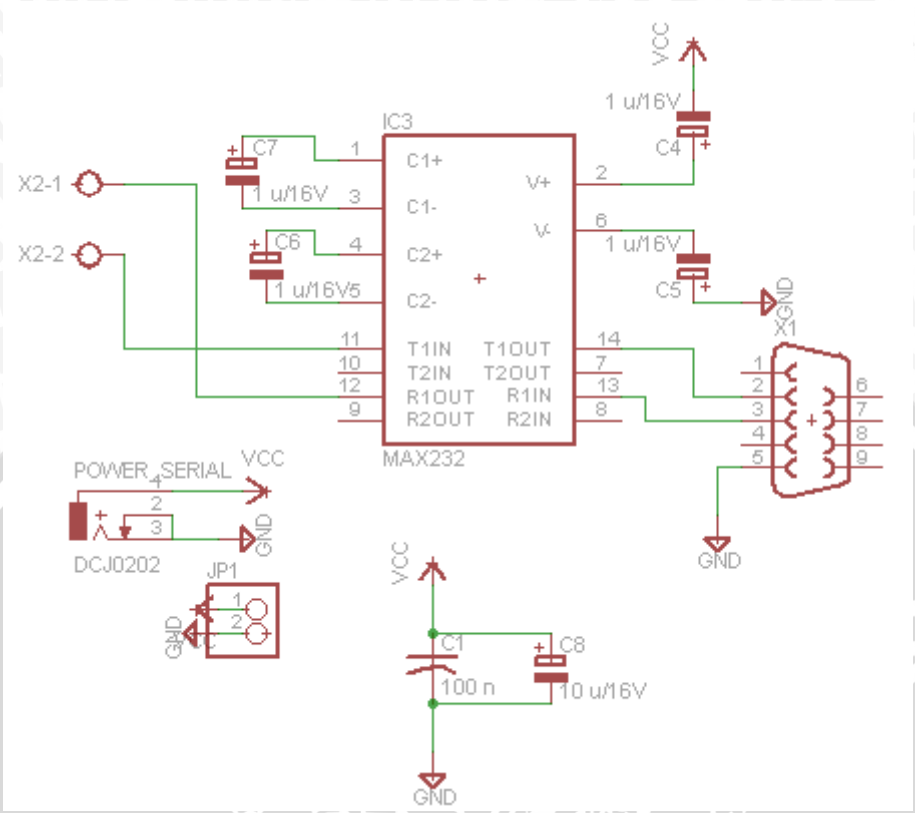
Skirpsi ini menggunakan mikrokontroler ATmega16 untuk menghitung RPM motor sekaligus mengirimkannya kepada program Visual Basic 6.0 dan mengirim sinyal PWM kepada driver. Kristal yang digunakan sebesar 11.0592 MHz.



Gambar 4.5 Rangkaian Minimum Sistem ATmega16

4.2.4 Perancangan Max 232

IC MAX232 dari Maxim Incoporation adalah IC pengubah level TTL menjadi RS-232 atau sebaliknya, yang memiliki sebuah *charge pump* yang bisa menghasilkan tegangan +10V dan -10V dari tegangan catu daya 5V. Tegangan-tegangan ini dihasilkan dengan proses pengisian dan pembuangan empat kapasitor luar yang dihubungkan dengan rangkaian pengganda tegangan internal yang dimiliki IC ini.



Gambar 4.6 Rangkaian Max 232

4.2.5 Perancangan *Rotary Encoder*

Sensor *rotary encoder* yang digunakan merupakan sensor yang sudah *build in* pada motor DC sehingga hanya perlu memberi catu daya sebesar 5 volt dan menghubungkan ground sensor pada ground sumber. Sensor yang digunakan memiliki resolusi 200 pulsa setiap satu kali putaran.

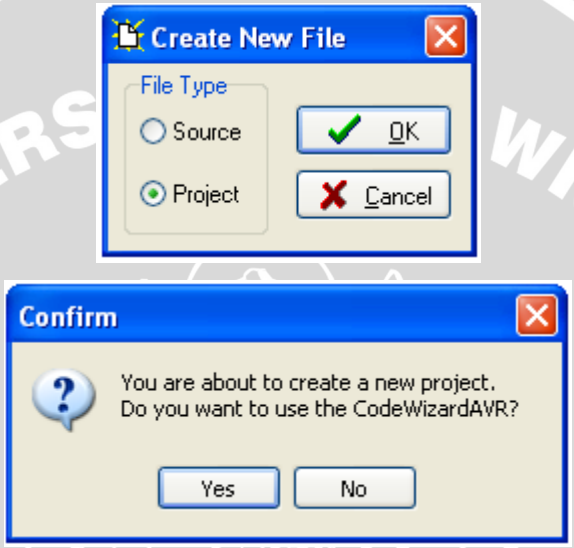
Gambar 4.7 Sensor Rotary Encoder

4.3 Perancangan Perangkat Lunak

4.3.1 Pemrograman Mikrokontroler ATmega16 menggunakan CVAVR.

Pemrograman miktrokontroler ATmega 16 menggunakan fasilitas CodeWizardAVR, langkah-langkahnya sebagai berikut:

1. ketika program CVAVR sudah terbuka, klik *file->new*, kemudian pilih *Project* Klik OK. Dalam kotak *box Confirm* Klik Yes.



Gambar 4.8 Membuat Project Baru di CVAVR

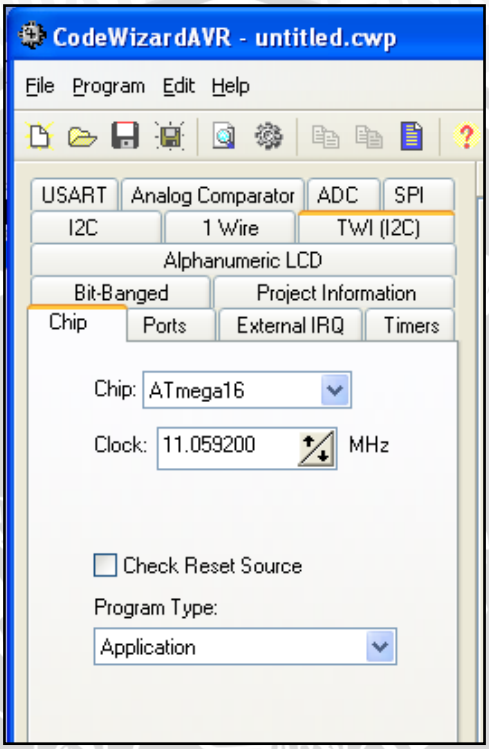
2. Dalam pilihan AVR *Chip Type* pilih opsi pertama kemudian Klik OK

Gambar 4.9 Pemilihan Jenis Chip

3. Ketika jendela *Code|wizardAVR* sudah terbuka beberapa hal yang perlu di atur sebagai berikut

   Tab Chip:

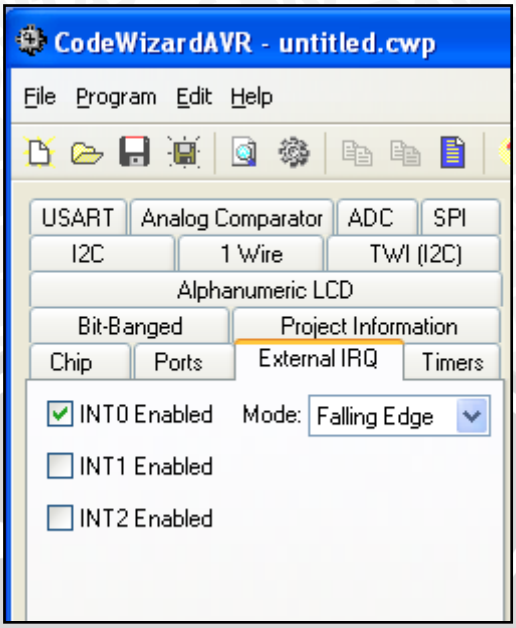
   pilih ATmega16 dengan Clock 11.059200 MHz



Gambar 4.10 Pengaturan Tab Chip

Tab External IRQ:

Centang INT0 Enabled dengan mode *falling edge*, bagian ini berfungsi untuk mencacah pulsa yang yang berasal dari sensor *rotary encoder*.



Gambar 4.11 Pengaturan Tab External IRQ

Tab Timer:

dalam skripsi ini menggunakan 2 *timer* yang ada yaitu *timer* 0 dan *timer* 1. Timer 0 berfungsi sebagai penghitung nilai RPM berdasarkan pulsa yang dibaca oleh INT0.

Untuk pengaturan Timer 0 menggunakan rumusan sebagai berikut:

Ttimer0 = Tosc*(256-TCNT0)*N → (8 bit = 256)

Tosc = 1/Fosc → pada aplikasi ini saya menggunakan kristal 11.0592 MHz, sehingga:

Tosc = 1/11.0592 = 0.00000009042245370 detik

Dimana:

Ttimer0 = lamanya periode *Timer* 0

TCNT0 = Register Timer0

N = Skala clock (mempunyai nilai 1, 8, 64, 256 dan 1024)

Tosc = periode clock

Fosc = frekuensi clock Kristal

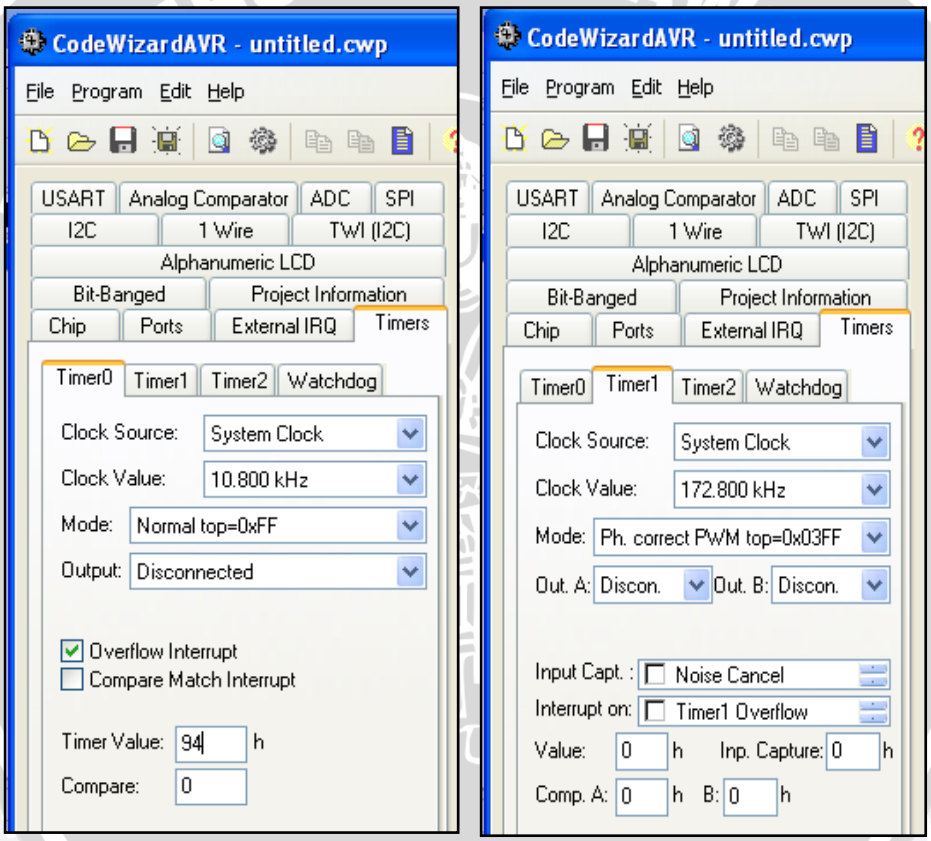
Periode *timer* 0 diatur sebesar 0,01 detik, jadi *timer* 0 akan melakukan interupsi tiap 0,01 detik dan skala *clock* (N) yang tetapkan 1024 Sehingga nilai TCNT0 dapat diketahui dengan rumus

Ttimer0 = Tosc*(256-TCNT0)*N

0,01=0.00000009042245370(256-TCNT0)*1024

TCNT0= 148 = 94 dalam Hexadesimal

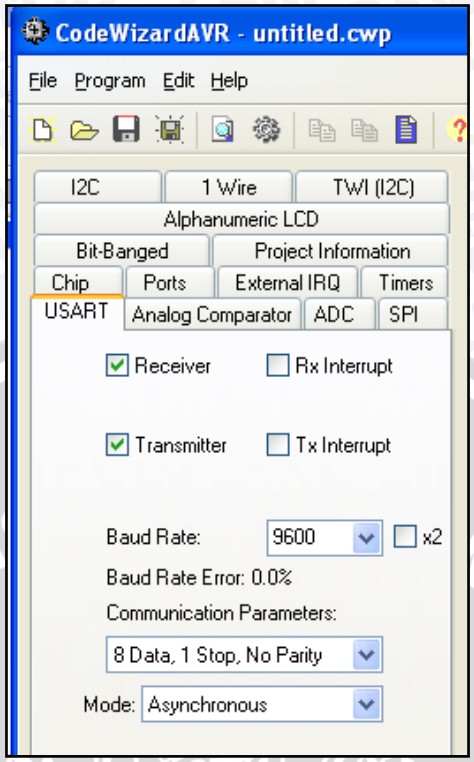
Sedangkan *timer* 1 berfungsi sebagi pengirim sinyal PWM kepada driver, resolusi yang ditentukan adalah sebesar 1023 cacah mewakili 100 persen Duty Cycle PWM. PWM yang digunakan adalah Fast PWM.



Gambar 4.12 Pengaturan Timer0 dan Timer1

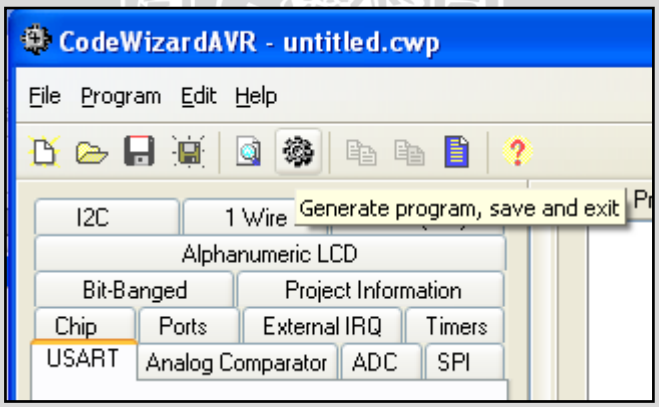
Tab USART:

Centang *Receiver* dan *Transmitter* untuk mengaktifkan Pin Rx dan Tx.



Gambar 4.13 Pengaturan USART

4. Setelah semua pengaturan sudah selesai tekan tombol berbentuk roda gigi untuk menggenerate *source code*. Kemudian *save* nama *file* sesuai keinginan dalam hal ini nama filenya final_mk.

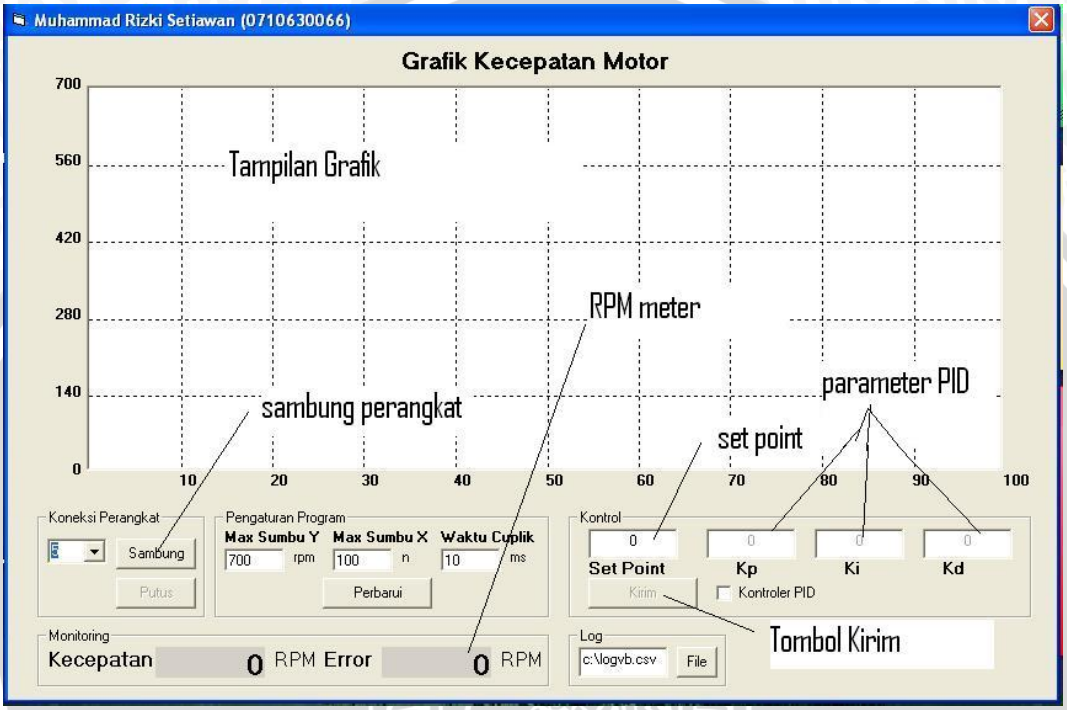


Gambar 4.14 Generate Program

5. Program lengkap mikrokontroler terdapat pada lampiran skripsi ini dengan nama final_mk.

4.3.2 Pemrograman Visual Basic 6.0.

Program yang dibuat menggunakan Visual Basic 6.0 memiliki kemampuan mengirim dan menerima data dari mikrokontroler ATmega16 ditambah kemampuan membuat grafik kecepatan secara *real time*. Data yang dikirim berupa *ouput* kontroler dan yang diterima berupa RPM motor.



Gambar 4.15 Tampilan Program Pengontrol Kecepatan

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai dengan perancangan yang telah dilakukan. Pengujian dilakukan per blok sistem untuk lebih mudah mengetahui kesalahan yang timbul, setelah itu dilakukan pengujian secara keseluruhan sistem. Adapun pengujian yang dilakukan sebagai berikut:

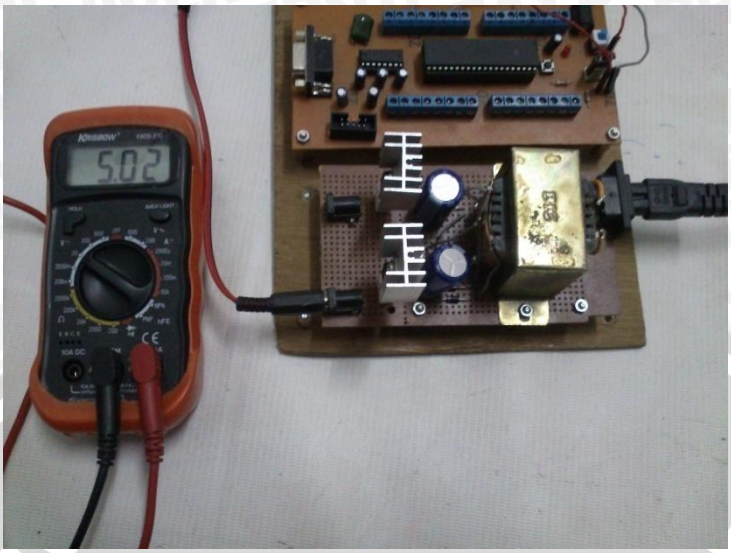
a. pengujian perangkat keras
   - pengujian catu daya
   - pengujian komunikasi serial
   - pengujian sensor *rotary encoder*
   - pengujian driver motor

b. pengujian keseluruhan sistem
   - pengambilan data *input-output*
   - penentuan fungsi alih *plant*
   - penentuan parameter PID menggunakan *root locus*
   - implementasi parameter PID pada *plant*

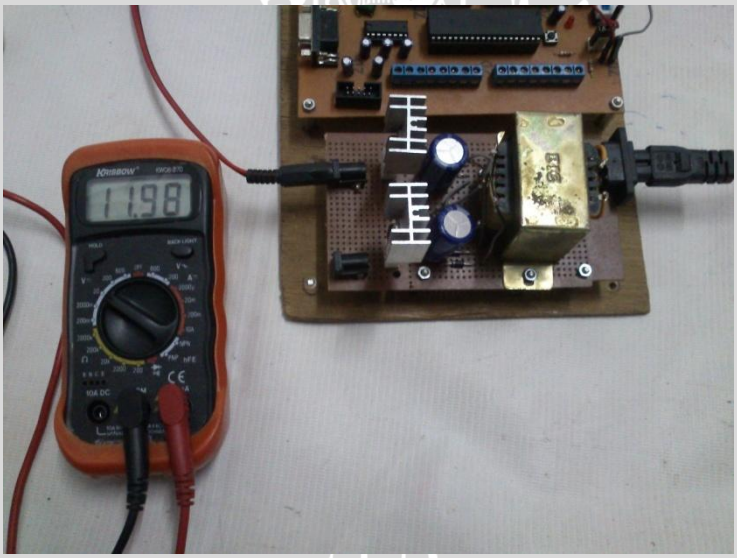
5.1 Pengujian Perangkat Keras

5.1.1 Pengujian Catu Daya

Pengujian rangkaian catu daya sistem bertujuan untuk mengetahui apakah catu daya dari sumber sudah sesuai dengan kriteria yang dibutuhkan dalam hal ini motor dan minimum sistem. Pengujian dilakukan dengan cara menghubungkan rangkaian catu daya dengan *power supply* dan keluarannya diukur dengan multimeter yang difungsikan sebagai *voltmeter*. Hasil pengujian ditunjukkan oleh Gambar 5.1 dan Gambar 5.3



Gambar 5.1 Pengujian Catu Daya Minimum Sistem



Gambar 5.2 Pengujian Catu Daya Motor DC

Pada gambar 5.1 dapat dilihat hasil pengukuran catu daya minimum sistem sebesar 5.02 V tegangan tersebut sudah mencukupi untuk menghidupkan minimum sistem. Sedangkan pada gambar 5.2 hasil pengukuran menujukkan angka 11.98 V, tegangan tersebut sudah mencukupi untuk menggerakkan motor DC.

5.1.2 Pengujian Komunikasi Serial

a. Tujuan

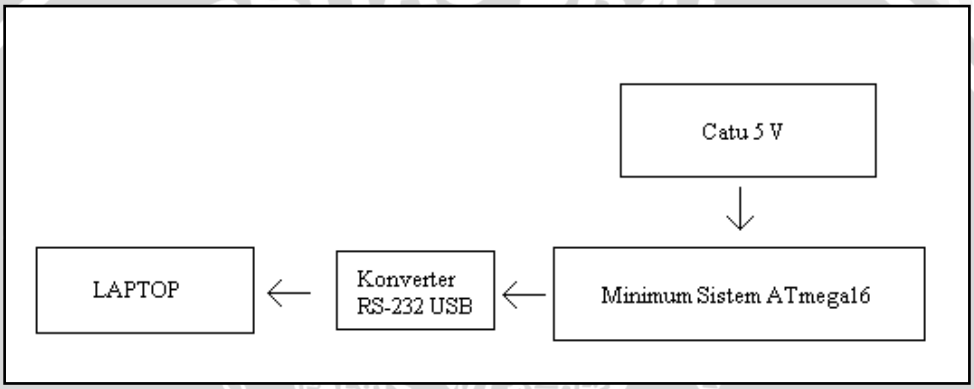
Mengetahui kinerja dari konverter RS-232 USB

b. Peralatan

1. minimum sistem Atmega16
2. konverter RS-232 USB
3. laptop

c. Langkah Pengujian

1. merangkai peralatan seperti pada gambar 5.3



Gambar 5.3 Blok Diagram Pengujian Komunikasi Serial

2. Mengisi mikrokontroler ATmega16 dengan program sederhana untuk mengirim beberapa karakter ke laptop.
3. Menghubungkan konektor DB9 dengan Konverter RS-232 USB kemudian mancapkannya ke salah satu port USB pada laptop.
4. Mengaktifkan minimum sistem dan menjalankan program Tera Term.
5. Mengamati perubahan pada termimal program *Tera Term*.

d. Hasil Pengujian

Tabel 5.1 Hasil Pengujian Komunikasi Serial

Data yang dikirim	Data yang diterima
1	1
2	2

3	3
A	A
B	B
Test Serial	Test Serial

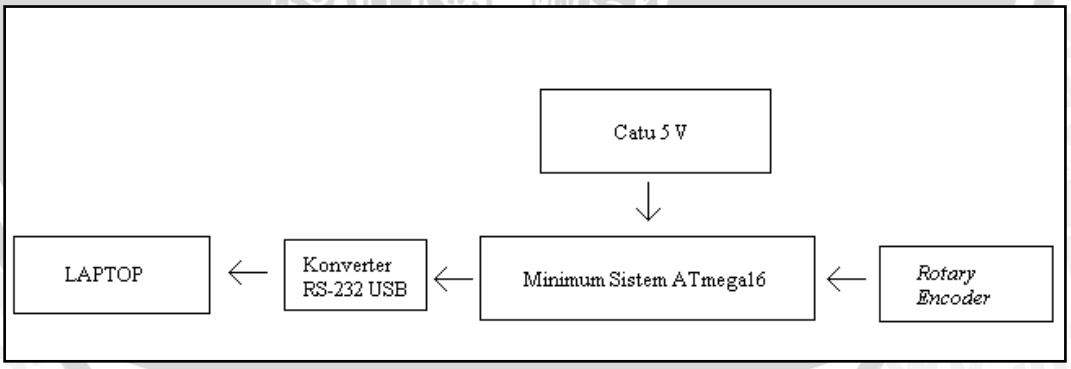
Berdasarkan hasil pengujian pada tabel 5.1 dapat diketahui bahwa rangkaian yang dibuat dapat mengirimkan 1 karakter dan beberapa karakter dari mikrokontroler ke laptop.

5.1.3 Pengujian Sensor *Rotary Encoder*

a. Tujuan
   - Mengetahui kinerja sensor *rotary encoder*
   - Mengetahui resolusi dari sensor *rotary encoder*

b. Peralatan
   1. Sensor *rotary encoder*
   2. Minimum sistem ATmega16
   3. Konverter RS-232 USB

c. Langkah Pengujian
   1. Rangkai rangkaian seperti pada gambar 5.4 berikut

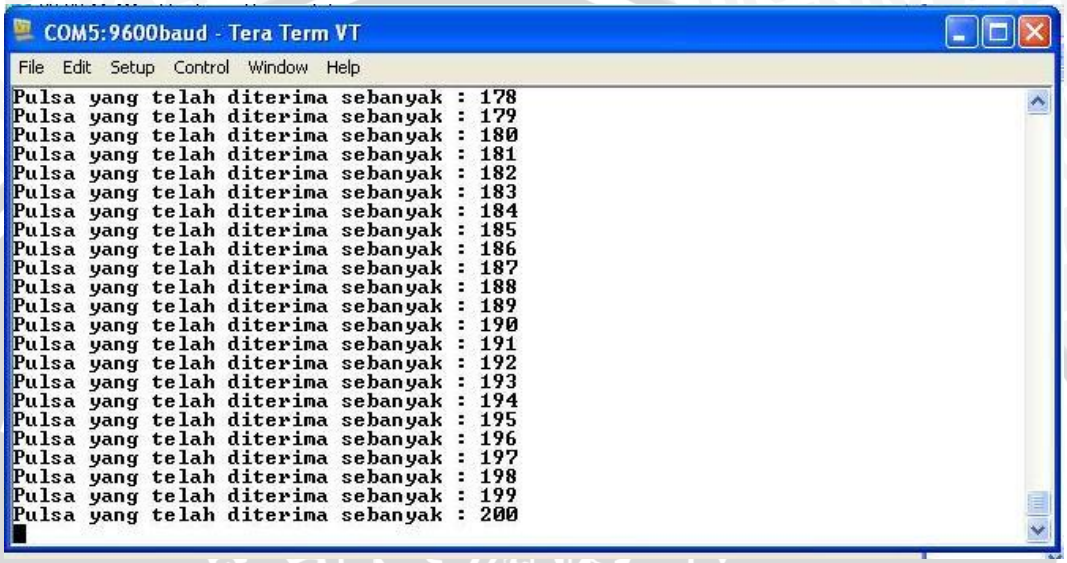


Gambar 5.4 Blok Diagram Pengujian Rotary Encoder

   2. Buat program sederhana pada mikrokontroler untuk menghitung jumlah pulsa yang dikirim oleh sensor *rotary encoder.*
   3. Menghubungkan konektor DB9 dengan Konverter RS-232 USB kemudian mancapkannya ke salah satu port USB pada laptop.

4. Mengaktifkan minimum sistem ATmega16, sensor *rotary encoder* dan menjalankan program tera term.
5. Memutar motor 1 kali putaran penuh.
6. Mengamati perubahan pada terminal program *tera term.*

d. Hasil Pengujian



Gambar 5.5 Hasil Pengujian Sensor Rotary Encoder

Dari hasil pengujian seperti yang tampak pada gambar 5.5 diketahui bahwa sensor *rotary encoder* dapat bekerja dengan baik dan resolusi sensor sebesar 200 pulsa setiap satu putaran, resolusi ini berguna untuk menghitung kecepatan motor.

5.1.4 Pengujian Driver

a. Tujuan

- Mengetahui hubungan antara PWM dengan tegangan keluaran driver.
- Mengetahui hubungan antara PWM dengan kecepatan motor.

b. Peralatan

1. Motor DC
2. Sensor *rotary encoder*
3. Minimum sistem ATmega16

4. Driver Motor

c. Langkah Pengujian Driver Tanpa Motor
   1. Menghubungkan driver dengan minimum sistem ATmega16.
   2. Menghubungkan tegangan 12 V pada masukan driver.
   3. Membuat program sederhana yang untuk mengeluarkan PWM dari 0 sampai 1020 (dengan skala PWM 1023).
   4. Mengukur tegangan keluaran pada keluaran driver menggunakan multimeter.

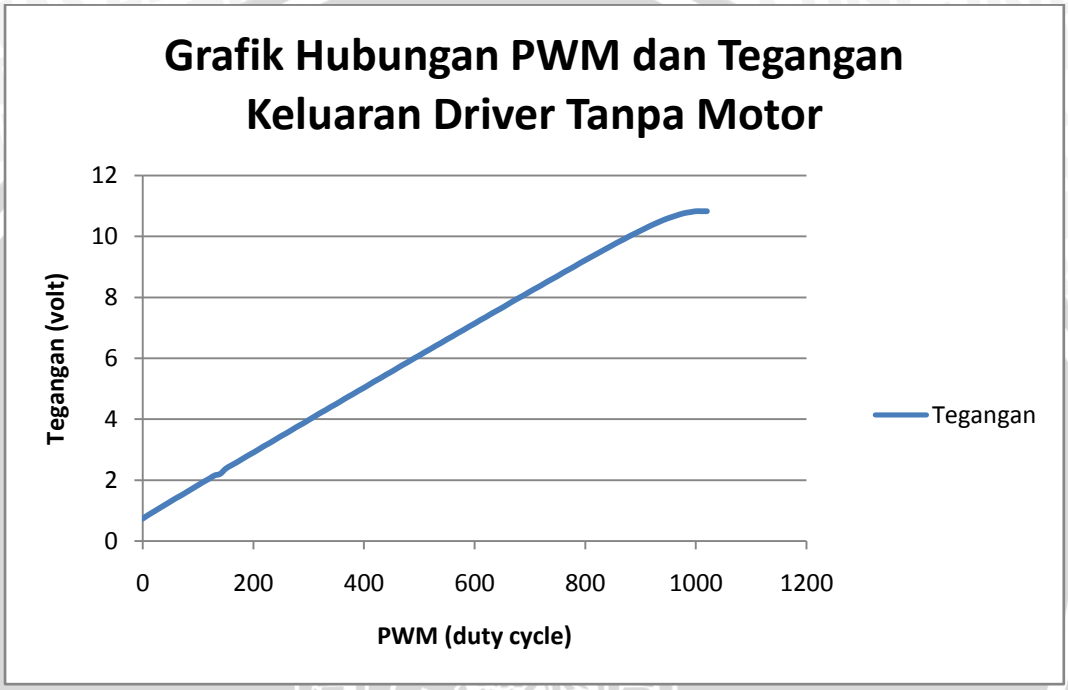
d. Langkah Pengujian Driver Menggunakan Motor
   1. Menghubungkan driver dengan minimum sistem dan motor DC.
   2. Menghubungkan tegangan 12 V pada masukan driver
   3. Membuat program sederhana yang untuk mengeluarkan PWM dari 0 sampai 1023 (dengan skala PWM 1023) dan mengitung nilai kecepatan pada saat itu juga.
   4. Menghubungkan minimum sistem dengan Laptop menggunakan koverter RS232 USB dan menjalan program tera term.
   5. Mengamati perubahan pada program tera term.

e. Hasil pengujian Driver tanpa motor

Tabel 5.2 Hasil Pengujian Driver Tanpa Motor

No	PWM	volt	No	PWM	volt	No	PWM	volt	No	PWM	volt
1	0	1.07	**27**	260	3.54	**53**	520	6.3	**79**	780	9.01
2	10	1.18	**28**	270	3.65	**54**	530	6.41	**80**	790	9.12
3	20	1.29	**29**	280	3.76	**55**	540	6.51	**81**	800	9.22
4	30	1.4	**30**	290	3.86	**56**	550	6.62	**82**	810	9.32
5	40	1.5	**31**	300	3.97	**57**	560	6.72	**83**	820	9.42
6	50	1.61	**32**	310	4.08	**58**	570	6.83	**84**	830	9.52
7	60	1.72	**33**	320	4.19	**59**	580	6.93	**85**	840	9.62
8	70	1.83	**34**	330	4.29	**60**	590	7.04	**86**	850	9.72
9	80	1.94	**35**	340	4.4	**61**	600	7.14	**87**	860	9.82
10	90	2.04	**36**	350	4.5	**62**	610	7.25	**88**	870	9.91
11	100	2.15	**37**	360	4.61	**63**	620	7.35	**89**	880	10.01
12	110	2.2	**38**	370	4.72	**64**	630	7.46	**90**	890	10.1
13	120	2.37	**39**	380	4.82	**65**	640	7.56	**91**	900	10.19
14	130	2.48	**40**	390	4.93	**66**	650	7.66	**92**	910	10.28
15	140	2.58	**41**	400	5.03	**67**	660	7.77	**93**	920	10.37
16	150	2.69	**42**	410	5.14	**68**	670	7.88	**94**	930	10.45
17	160	2.8	**43**	420	5.25	**69**	680	7.98	**95**	940	10.53
18	170	2.9	**44**	430	5.35	**70**	690	8.08	**96**	950	10.6

19	180	3.01	**45**	440	5.46	**71**	700	8.19	**97**	960	10.66
20	190	3.12	**46**	450	5.56	**72**	710	8.29	**98**	970	10.72
21	200	3.22	**47**	460	5.67	**73**	720	8.39	**99**	980	10.77
22	210	3.33	**48**	470	5.78	**74**	730	8.5	**100**	990	10.8
23	220	3.44	**49**	480	5.88	**75**	740	8.6	**101**	1000	10.83
24	230	1.07	**50**	490	5.99	**76**	750	8.7	**102**	1010	10.83
25	240	1.18	**51**	500	6.09	**77**	760	8.81	**103**	1020	10.83
26	250	1.29	**52**	510	6.2	**78**	770	8.91			



Gambar 5.6 Grafik Hubungan PWM dengan Tegangan Keluaran Driver Tanpa Motor

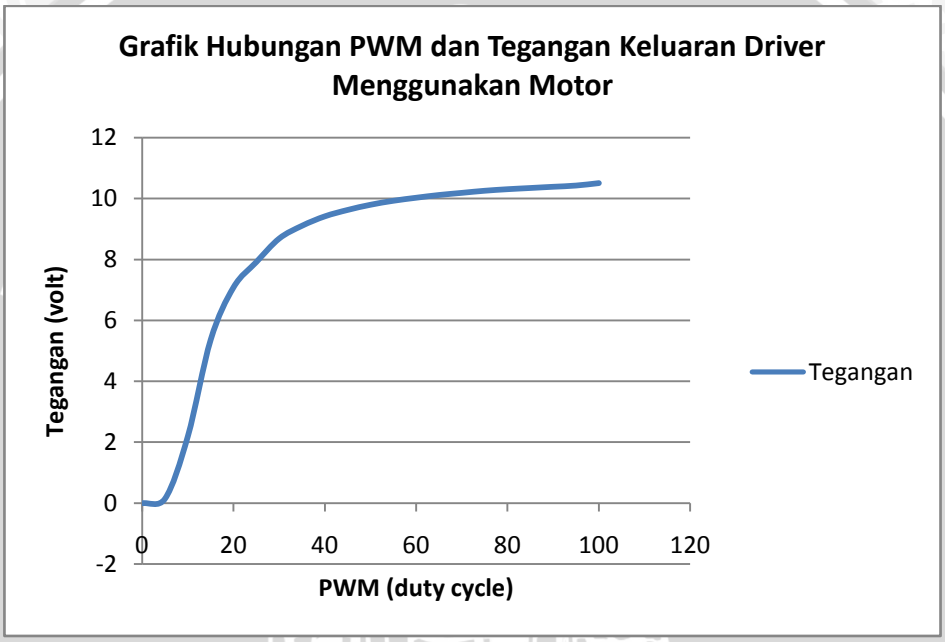
Dari hasil pengujian yang telah dilakukan didapatkan hasil seperti pada tabel 5.2 dan gambar 5.6 diketahui bahwa driver sudah mengeluarkan tegangan meskipun nilai PWM masih 0, hal ini disebabkan karena dalam rangkaian ini tidak digunakan *optocoupler* sehingga terjadi kebocoran tegangan dari minimum sistem tetapi masih dalam jumlah yang kecil.

f. Hasil Pengujian Driver menggunakan motor

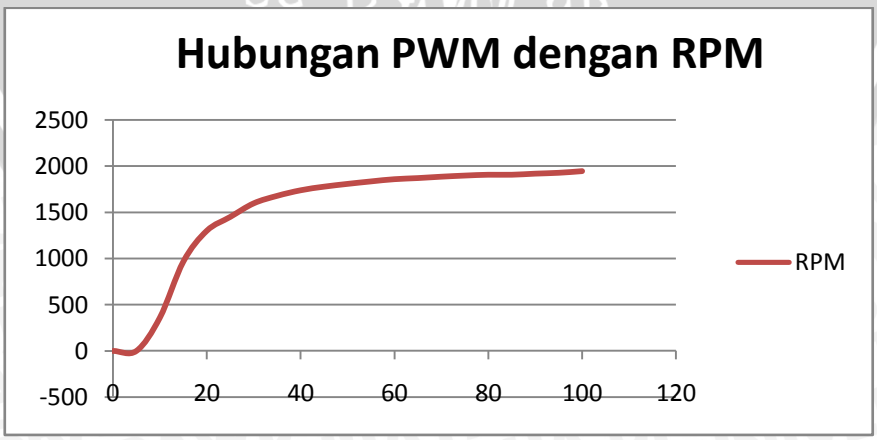
Tabel 5.3 Hasil Pengujian Driver Menggunakan Motor

No	duty cycle (%)	volt	RPM	No	duty cycle (%)	volt	RPM
1	0	0	0	**12**	55	9.93	1833

2	5	0.15	0	**13**	60	10.03	1857
3	10	2.2	360	**14**	65	10.12	1869
4	15	5.37	966	**15**	70	10.19	1884
5	20	7.09	1299	**16**	75	10.26	1896
6	25	7.92	1449	**17**	80	10.31	1905
7	30	8.69	1596	**18**	85	10.35	1905
8	35	9.11	1677	**19**	90	10.39	1917
9	40	9.42	1737	**20**	95	10.43	1926
10	45	9.63	1776	**21**	100	10.51	1944
11	50	9.8	1806				



Gambar 5.7 Grafik Hubungan PWM dengan Tegangan Keluaran Driver Menggunakan Motor



Gambar 5.8 Grafik Hubungan PWM Kecepatan Motor(RPM)

Dari grafik di atas motor mulai berputar dalam kisaran PWM 5%-10% *duty cycle*, kecepatan motor mengalami perubahan yang besar dalam kisaran 10%-30% *duty cycle* sedangkan 30%-100% *duty cycle* perubahan kecepatan tidak begitu besar.

5.2 Pengujian Keseluruhan Sistem

5.2.1 Pengambilan Data *Input-Output*

a. Tujuan

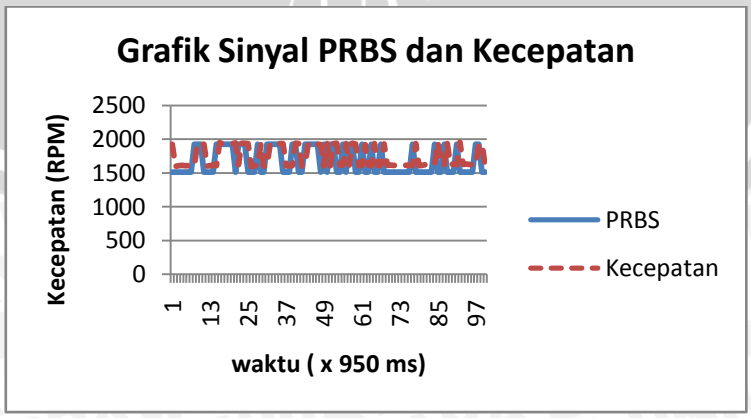
b. Peralatan

1. Motor DC
2. Sensor *rotary encoder*
3. Minimum sistem ATmega16
4. Driver Motor

c. Langkah Pengujian

1. Menghubungkan driver dengan minimum sistem dan motor DC.
2. Menghubungkan tegangan 12 V pada masukan driver.
3. Membuat program PRBS 10 bit yang menghasilkan 1024 jenis sinyal acak.
4. Menghubungkan minimum sistem dengan laptop menggunakan koverter RS232 USB dan menjalan program *tera term*.
5. Mengamati perubahan pada terminal program *tera term*.

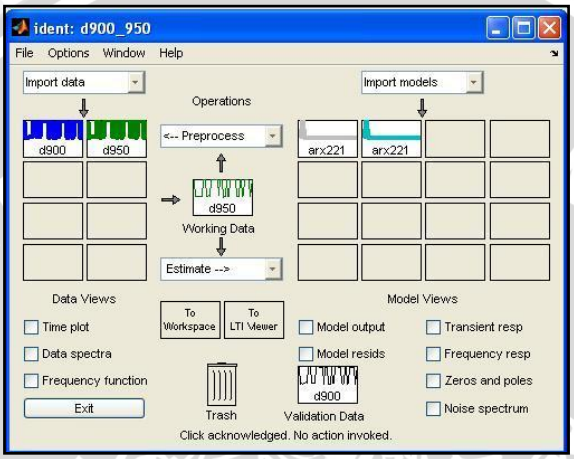
d. Hasil pengujian



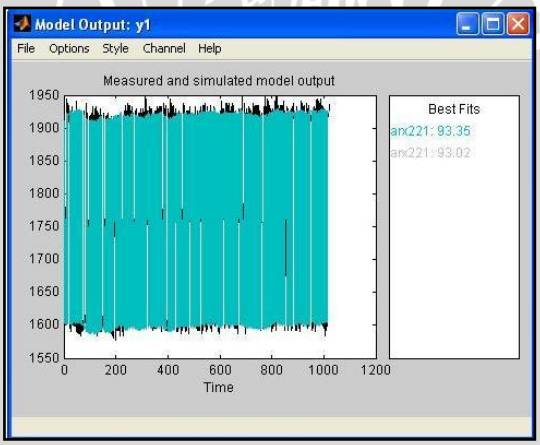
Gambar 5.9 Grafik Sinyal PRBS dan Kecepatan

5.2.2 Penentuan Fungsi Alih Plant

Identifikasi sistem bertujuan didapatkannya fungsi alih dari keseluruhan sistem, pengujian dilakukan menggunakan software MATLAB 7 dengan fasilitas *ident* yang dimilikinya. Data identifikasi yang digunakan, PRBS sebagai input sedangkan kecepatan sebagai output. Struktur model yang digunakan *adalah Auto Regresive with Exogenous input* (ARX) dengan estimasi parameter 2 2 1.



Gambar 5.10 Identifikasi Menggunakan Ident MATLAB



Gambar 5.11 Best Fit Sistem

Dari hasil di atas didapatkan best fit terbaik yaitu 93.35 % dengan fungsi alih diskrit:

A(q) = 1 - 0.9717 (+-0.009162) q^-1 - 0.01271 (+-0.00163) q^-2 (5.1)

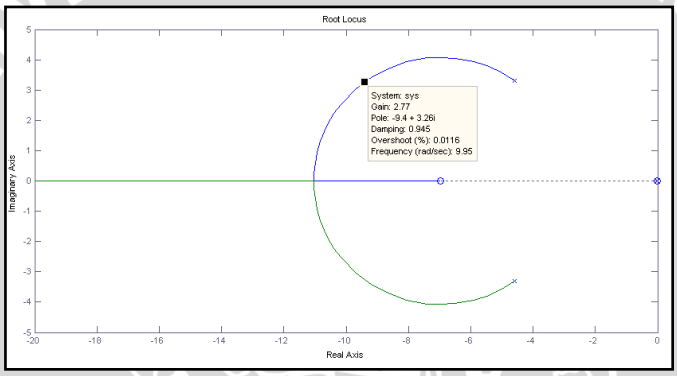
B(q) = 0.7754 (+-0.001338) q^-1 - 0.7595 (+-0.007813) q^-2 (5.2)

Dari data di atas bisa didapatakan fungsi alih dalam bentuk s nya:

$$F(s) = \frac{3.492\,s^2 + 24.4\,s + 0.5313}{s^3 + 9.173\,s^2 + 32.05\,s + 0.5376} \qquad (5.3)$$

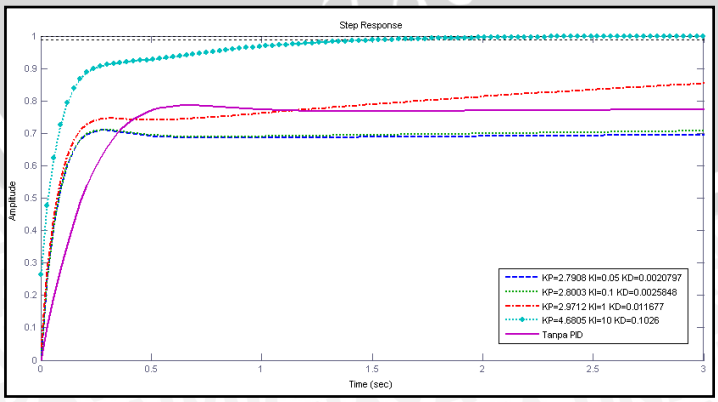
5.2.3 Penentuan Parameter PID menggunakan *root locus*

Untuk mendapatkan parameter PID yang yang diinginkan terlebih dahulu ditentukan pole yang dinginkan berdasarkan grafik *root locus* dari sistem. Dari grafik *root locus* dapat dilihat bahwa semua akar berada pada sisi kira bidang s, dapat disimpulkan bahwa sistem stabil dalam nilai manapun, dalam penelitian ini dipilih pole s = -9.4 + j3.26.



Gambar 5.12 *Root Locus* Fungsi Alih sistem dan Pemilihan Pole

selanjutnya dilakukan perhitungan untuk menetukan parameter PID sesuai dengan kriteria yang diinginkan dalam hal ini menggunakan metode *root locus* yang diimplementasikan dalam program MATLAB. Sesuai dengan persamaan 2.17



Gambar 5.13 Grafik Respond sistem tanpa PID dan Dengan PID

Dari 4 jenis parameter PID yang didapat dipilih nilai PID yang memiliki respond terbaik yaitu :

KP = 4.6805

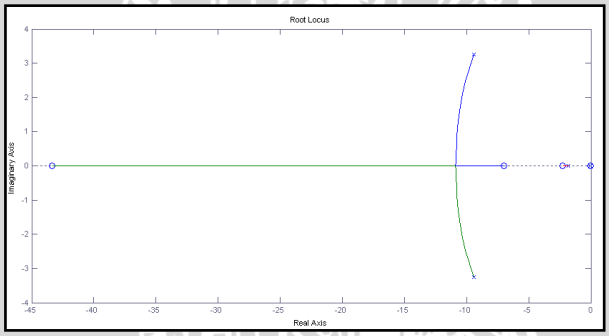
KI = 10

KD = 0.1026

Dengan memasukkan nilai parameter PID ke dalam fungsi alih sistem, maka fungsi alih sistem berubah menjadi:

$$F(s) = \frac{0.3583s^4 + 18.8478s^3 + 149.1795s^2 + 246.4868s + 5.313}{1.3583s^4 + 28.021s^3 + 181.23s^2 + 247.02s + 5.313} \quad (5.4)$$

Gambar *root locus* dari fungsi alih plant ditambah PID adalah sebagai berikut:

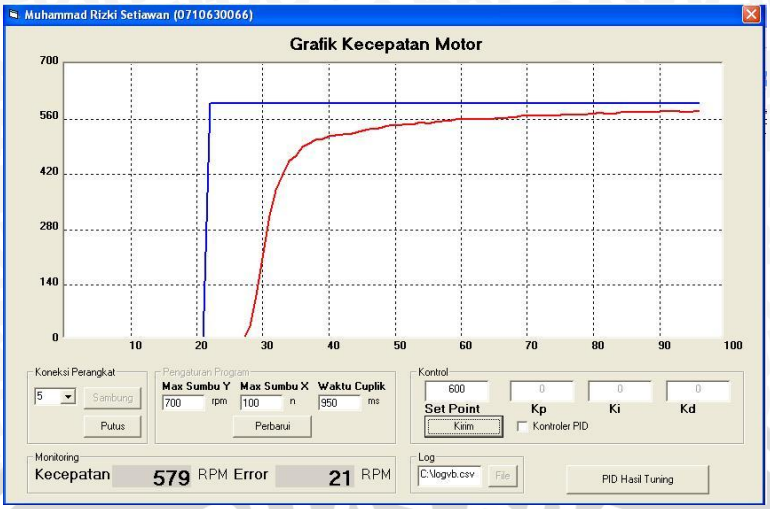


Gambar 5.14 *Root Locus* Sistem Keseluruhan

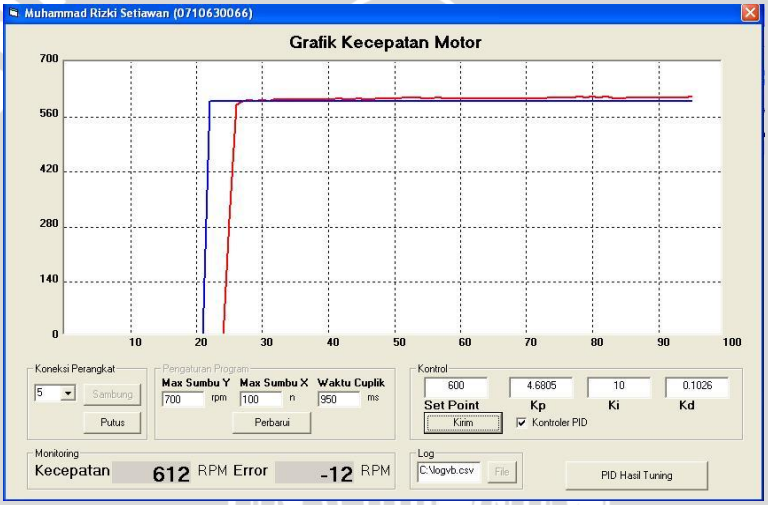
5.2.4 Implementasi Parameter PID pada Plant

Pengujian sistem secara keselurahan ini dilakukan untuk mengetahui kinerja perangkat keras dan perangkat lunak serta mengetahui respon motor tanpa PID dan dengan PID.

Dengan dilakukannya implementasi nilai parameter PID yang telah dihitung yaitu KP=4.680, KI=10 dan KD=0.1026 ke dalam rangkaian keseluruhan sistem. Dari proses implentasi tersebut dihasilkan respon seperti pada Gambar 5.15 dan Gambar 5.14



Gambar 5.15 Grafik Respon Motor Tanpa PID dengan Visual Basic



Gambar 5.16 Grafik Respon Motor Menggunakan PID dengan Visual Basic

Garis warna biru menunjukkan *set point* dan warna merah menunjukkan kecepatan, diketahui bahwa dengan menggunakan parameter PID hasil tuning didapatkan respon yang lebih cepat dari pada respon tanpa menggunakan PID.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan dari hasil pengujian keseluruhan yang telah dilakukan dapat disimpulakan beberapa hal sebagai berikut:

1. *Driver* motor dapat menjalankan motor DC sesuai dengan sinyal PWM yang diberikan dengan nilai kecepatan maksimal 1962 RPM.
2. Minimum sistem ATmega16 dapat menghitung dan mengirim kecepatan motor DC sesuai dengan masukan dari sensor *rotary encoder*.
3. Program yang dibuat dari Visual Basic 6.0 dapat melakukan pembacaan nilai kecepatan dari miktrokontroler dan menampilkannya dengan grafik secara *real time*.
4. Dari identifikasi data *input-output* sinyal PRBS didapatkan *Best Fit* sebesar 93.35 % dan fungsi alih sistem:

$$F(s) = \frac{3.492\,s^2 + 24.4\,s + 0.5313}{s^3 + 9.173\,s^2 + 32.05\,s + 0.5376}$$

5. Dari penentuan parameter PID menggunakan metode *root locus* dipilih pole s = -9.4 + j3.26 dengan kriteria *damping ratio* = 0.945, *overshoot* = 0.0116 dan frekuensi = 9.95 rad/sec didapat nilai PID terbaik yaitu KP = 4.6805, KI = 10, KD = 0.1026.

6.2 Saran

Untuk penelitian selanjutanya mengenai kontrol kecepatan motor DC disarankan:

1. Menggunakan perantara USB sebagai penghubung dengan komputer.
2. Mencari parameter PID dengan metode lain.
3. Proses pencarian parameter PID dapat langsung dilakukan dengan program yang dibuat, jadi pengguna tinggal menentukan karakteristik respon yang diinginkan.
4. Melakukan modifikasi *hardware* dan *software* sehingga dapat diketahui karakteristik ketika motor diberi beban.

DAFTAR PUSTAKA

ATMEL.2007.ATmega16/ATmega16L, 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash.

Wibowo, Teguh Budi. 2008. Pemodelan Sistem Plant Suhu Dengan Metode Identifikasi Recursive Least Square. Laporan Skripsi, Teknik Elektro Brawijaya

Landau, Ioan dan Gianluca Zito. 2006. Digital Control Systems Design, Identification and Implementation. Germany: Springer-Verlag London Limited

Ikrom, Hassanal. 2008. Perancangan Kontroler Pid-Kaskade Dengan Metode *Root Locus* Untuk Kontrol Temperatur Dan Tekanan Pada Proses Evaporator. Laporan Skripsi, Teknik Elektro Brawijaya

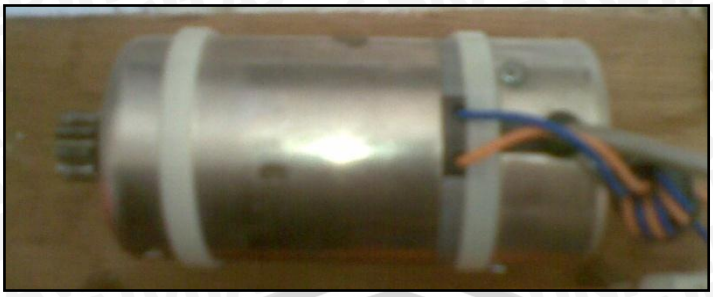
Philip, C. L. & Harbor, R. D. 1996. Feedback Control System. Prentice Hall. New Jersey

Prasetia, Retna.dkk.2004. interfacing port pararel dan port serial computer dengan Visual Basic 6.0.Andi Yogyakarta

LAMPIRAN

LAMPIRAN I

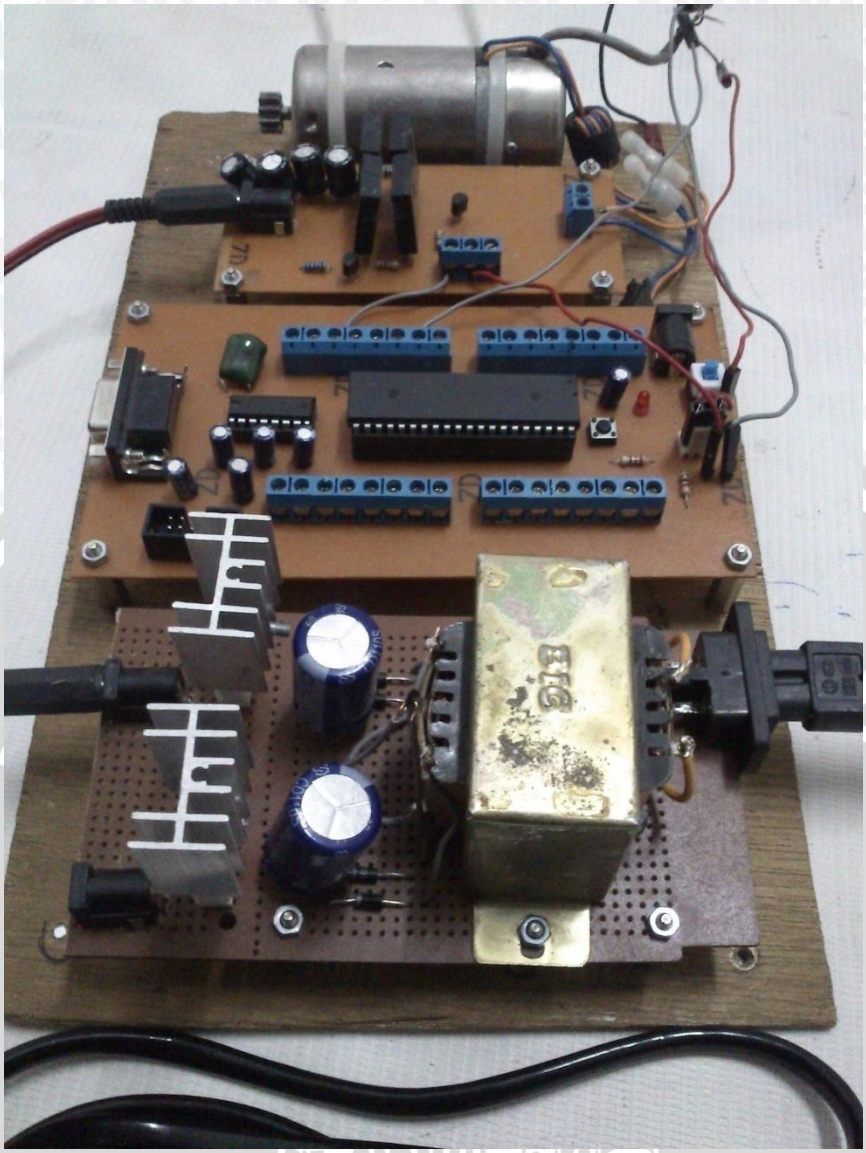
FOTO ALAT



Gambar Motor DC dan Sensor *rotary encoder*

Gambar Driver Motor

Gambar Minimum Sistem ATmega16



Gambar Keseluruhan Sistem

LAMPIRAN II

LISTING PROGRAM

MIKROKONTROLER ATMEGA 16

```
#include <mega16.h>
#include <stdlib.h>
#include <delay.h>

int rotari=0;
int hitung=0;
int pwm,rpm;

// External Interrupt 0 service routine
interrupt [EXT_INT0] void
ext_int0_isr(void)
{
// Place your code here
  rotari++;
}

// Standard Input/Output functions
#include <stdio.h>
void clearStringBuffer();
void getString();

int ref;

char dataString[16];
void clearStringBuffer(){
      unsigned char i=0;
      for (i=0;i<16;i++){
            dataString[i]='';
      }
}
void getString(){
    unsigned char i=0,temp;
    clearStringBuffer();
    temp = getchar();
    while (temp!='a'){
          dataString[i]=temp;
          i++;
          temp = getchar();
    }
}

// Timer 0 overflow interrupt service
routine
interrupt [TIM0_OVF] void
timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x94;
// Place your code here
hitung++;
if (hitung==10)
 {
 rpm=(float)rotari*3;
 rotari=0;
 hitung=0;
 }
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// 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=0x00;

// 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=In Func3=In Func2=In
Func1=In Func0=In
// State7=T State6=T State5=0
State4=T State3=T State2=T
State1=T State0=T
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 10.800 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x05;
TCNT0=0x94;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 11059.200 kHz
// Mode: Ph. correct PWM
top=0x03FF
// OC1A output: Non-Inv.
// OC1B output: Discon.
// 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=0x83;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;

// Timer(s)/Counter(s) Interrupt(s)
initialization
TIMSK=0x01;

// USART initialization
// Communication Parameters: 8
Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

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

// ADC initialization
// ADC disabled
ADCSRA=0x00;

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

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

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

while (1)
      {
      // Place your code here
       printf("s%dp",rpm);
    clearStringBuffer();
    getString();


 //ref=atof(dataString);
    ref=atoi(dataString);
    pwm=ref;
    if (pwm>1023)
    {pwm=1023;}
    if (pwm<0)
    {pwm=0;};
    OCR1A=pwm;
      }

}
```

LAMPIRAN III

LISTING PROGRAM

VISUAL BASIC 6.0

```
Option Explicit
Dim buffer, filename, kiri, pvs, folder As String
Dim sv, pp, pv, pv2, sv2, kp, ki, kd, ts, t, t2, i As Integer
Dim error, error_i, error_d, error_sblm_i, error_sblm_d, tc As Integer
Dim outp, outi, outd  As Integer
Dim SkalaX, SkalaY, UX1, UY1, YX1, YY1, UX2, UY2, YX2, YY2 As Double
Dim skala_rpm As Integer 'nilai maksimal rpm
Dim skala_error As Integer
Dim t_rpm As Integer 'nilai maksimal t rpm
Dim t_error As Integer
Dim n As Byte
Dim pwm As String
Dim outpidpwm As String
Dim outpid As String



Private Sub Check1_Click()
If Check1.Value = 1 Then
'enable input ke pid default
Text2.Enabled = True
Text3.Enabled = True
Text4.Enabled = True
Timer2.Enabled = False
Timer3.Enabled = True
Else
'disable input ke pid default
Text2.Enabled = False
Text3.Enabled = False
Text4.Enabled = False
Timer2.Enabled = True
Timer3.Enabled = False

End If
End Sub

Private Sub cmdKirim_Click()
If ((IsNumeric(Text1.Text) And IsNumeric(Text2.Text) And IsNumeric(Text3.Text) And IsNumeric(Text4.Text))) = False Then
MsgBox "Data yang dimasukkan tidak valid, Hanya menerima data Numerik"
Else
If Check1.Value = 0 Then
sv = Text1.Text
Else
If Check1.Value = 1 Then
'reset pid
error_sblm_i = 0
error_sblm_d = 0
outp = 0
outi = 0
outd = 0
outpid = 0
Timer3.Enabled = False
Timer3.Enabled = True


sv = Val(Text1.Text)
kp = Val(Text2.Text)
ki = Val(Text3.Text)
kd = Val(Text4.Text)
End If
End If
End If

End Sub

Private Sub cmdLog_Click()
' Set CancelError is True
 CommonDialog1.CancelError = True
 On Error GoTo ErrHandler
 ' Set flags
 CommonDialog1.Flags = cdlOFNHideReadOnly


 ' Set filters
 CommonDialog1.Filter = "All Files (*.*)|*.*|Text Files" & _
 "(*.txt)|*.txt|csv (*.csv)|*.csv"
 ' Specify default filter
 CommonDialog1.FilterIndex = 3
 ' Display the Open dialog box
```

```
  CommonDialog1.ShowOpen
  ' Display name of selected file
   filename                    =
CommonDialog1.filename
   Text5.Text = filename
   Exit Sub

ErrHandler:
  'User pressed the Cancel button
  Exit Sub
End Sub

Private Sub cmdPutus_Click()
MSComm1.PortOpen = False
cmdSambung.Enabled = True
cmdPutus.Enabled = False
Label1.Caption = ""
Label4.Caption = ""
Timer1.Enabled = False
Frame5.Enabled = True

'membuat settingan grafik default

t2 = 0
Picture1.Cls
i = 0
UX1 = 0

SkalaX = (Picture1.Width / t_rpm)
SkalaY     =     (Picture1.Height     /
skala_rpm)

skala_sumbu_y
waktu_rpm
Frame3.Enabled = True
cmdLog.Enabled = True

End Sub

Private Sub cmdSambung_Click()
MSComm1.CommPort = port.Text
MSComm1.PortOpen = True
cmdSambung.Enabled = False
cmdPutus.Enabled = True
cmdLog.Enabled = False
Timer1.Enabled = True
Timer2.Enabled = True
Frame5.Enabled = False
cmdKirim.Enabled = True


End Sub


Private Sub Command1_Click()
If    ((IsNumeric(Text8.Text)    And
IsNumeric(Text9.Text)          And
IsNumeric(Text10.Text)))   =   False
Then
MsgBox "Data yang dimasukkan
tidak valid, Hanya menerima data
Numerik"
Else
skala_rpm = Text8.Text
t_rpm = Text9.Text
Timer1.Interval = Text10.Text
Timer2.Interval = Text10.Text
Timer3.Interval = Text10.Text
t = t_rpm
t2 = 0

SkalaX = (Picture1.Width / t_rpm)
SkalaY     =     (Picture1.Height     /
skala_rpm)

skala_sumbu_y
waktu_rpm
End If

End Sub


Private Sub Command2_Click()
Text2.Text = 4.6805
Text3.Text = 10
Text4.Text = 0.1026

kp = Val(Text2.Text)
ki = Val(Text3.Text)
kd = Val(Text4.Text)

End Sub
```

```
Private Sub Form_Load()
'nilai awal pid

outpid = 0
sv = 0
error_sblm_i = 0
error_sblm_d = 0

tc = 0.01 '0.95 '0.01

skala_rpm = 2500
t_rpm = 100

t = t_rpm
t2 = 0
UX1 = 0
UY1 = Picture1.Height
YX1 = 0
YY1 = Picture1.Height

SkalaX = (Picture1.Width / t_rpm)
SkalaY = (Picture1.Height / skala_rpm)

skala_sumbu_y
grid
waktu_rpm

port.AddItem "1"
port.AddItem "2"
port.AddItem "3"
port.AddItem "4"
port.AddItem "5"
port.AddItem "6"
port.AddItem "7"
port.AddItem "8"
port.AddItem "9"
port.AddItem "10"
port.AddItem "11"
port.AddItem "12"
port.AddItem "13"
port.AddItem "14"
port.AddItem "15"
port.AddItem "16"
port.AddItem "17"
port.AddItem "18"
port.AddItem "19"
port.AddItem "20"
port.AddItem "21"
port.AddItem "22"
port.AddItem "23"
port.AddItem "24"

port.ListIndex = 4
cmdPutus.Enabled = False
cmdKirim.Enabled = False


'disable input ke pid default
Text2.Enabled = False
Text3.Enabled = False
Text4.Enabled = False

filename = Text5.Text
End Sub
Private Sub MSComm1_OnComm()

If MSComm1.CommEvent = comEvReceive Then
On Error Resume Next
buffer = MSComm1.Input
kiri = buffer
If Left(kiri, 1) = "s" Then
pp = InStr(1, kiri, "p")
pvs = Mid(kiri, 2, pp - 2)
pv = Val(pvs)
Label1.Caption = pv
Label4.Caption = sv - pv
End If
End If

End Sub

Private Sub Timer1_Timer()
MSComm1_OnComm
Open filename For Append As #1
Print #1, sv; pv
Close #1

'gambar grafik kecepatan
Picture1.DrawWidth = 2
UX2 = i * SkalaX
UY2 = Picture1.Height - (SkalaY * pv)
If UY2 < 0 Then UY2 = 0
```

```
If UY2 > Picture1.Height Then UY2
= Picture1.Height

YX2 = i * SkalaX
YY2 = Picture1.Height - (SkalaY *
sv)
If YY2 < 0 Then UY2 = 0
If YY2 > Picture1.Height Then YY2
= Picture1.Height

Picture1.Line (UX1, UY1)-(UX2,
UY2), vbRed
Picture1.Line (YX1, YY1)-(YX2,
YY2), vbBlue

'gambar data berikutnya
UX1 = UX2
UY1 = UY2
YX1 = YX2
YY1 = YY2
i = i + 1

If i >= t_rpm Then
Picture1.Cls
i = 0
UX1 = 0
YX1 = 0

t2 = t2 + t 'menambah waktu di
gambar berikutnya
waktu_rpm
End If
End Sub

Private Sub skala_sumbu_y()
'rata kanan
Label11.Alignment = 1
Label12.Alignment = 1
Label13.Alignment = 1
Label14.Alignment = 1
Label15.Alignment = 1
Label16.Alignment = 1

'menentukan sisi kiri dari sumbu y
rpm
Label11.Left = Picture1.Left - 800
Label12.Left = Picture1.Left - 800
Label13.Left = Picture1.Left - 800
Label14.Left = Picture1.Left - 800
Label15.Left = Picture1.Left - 800
Label16.Left = Picture1.Left - 800
'menentukan sisi atas dari sumbu y
rpm
Label11.Top = Picture1.Top + (5 *
(Picture1.Height / 5)) - 125
Label12.Top = Picture1.Top + (4 *
(Picture1.Height / 5)) - 125
Label13.Top = Picture1.Top + (3 *
(Picture1.Height / 5)) - 125
Label14.Top = Picture1.Top + (2 *
(Picture1.Height / 5)) - 125
Label15.Top = Picture1.Top + (1 *
(Picture1.Height / 5)) - 125
Label16.Top = Picture1.Top + (0 *
(Picture1.Height / 5)) - 125
'memberikan nilai dari sumber y rpm
Label11.Caption = skala_rpm / 5 * 0
Label12.Caption = skala_rpm / 5 * 1
Label13.Caption = skala_rpm / 5 * 2
Label14.Caption = skala_rpm / 5 * 3
Label15.Caption = skala_rpm / 5 * 4
Label16.Caption = skala_rpm / 5 * 5

'menentukan sisi kiri dari sumbu x
rpm
Label17.Left = Picture1.Left + (1 *
Picture1.Width / 10)
Label18.Left = Picture1.Left + (2 *
Picture1.Width / 10)
Label19.Left = Picture1.Left + (3 *
Picture1.Width / 10)
Label20.Left = Picture1.Left + (4 *
Picture1.Width / 10)
Label21.Left = Picture1.Left + (5 *
Picture1.Width / 10)
Label22.Left = Picture1.Left + (6 *
Picture1.Width / 10)
Label23.Left = Picture1.Left + (7 *
Picture1.Width / 10)
Label24.Left = Picture1.Left + (8 *
Picture1.Width / 10)
Label25.Left = Picture1.Left + (9 *
Picture1.Width / 10)
Label26.Left = Picture1.Left + (10 *
Picture1.Width / 10)
```

```
'menentukan sisi atas dari sumbu x rpm
Label17.Top = Picture1.Top + (Picture1.Height + 10)
Label18.Top = Picture1.Top + (Picture1.Height + 10)
Label19.Top = Picture1.Top + (Picture1.Height + 10)
Label20.Top = Picture1.Top + (Picture1.Height + 10)
Label21.Top = Picture1.Top + (Picture1.Height + 10)
Label22.Top = Picture1.Top + (Picture1.Height + 10)
Label23.Top = Picture1.Top + (Picture1.Height + 10)
Label24.Top = Picture1.Top + (Picture1.Height + 10)
Label25.Top = Picture1.Top + (Picture1.Height + 10)
Label26.Top = Picture1.Top + (Picture1.Height + 10)



End Sub

Private Sub grid()
'horisontal rpm
Line1.X1 = 0
Line1.X2 = 0 + Picture1.Width
Line1.Y1 = 0 + (1 * (Picture1.Height / 5))
Line1.Y2 = 0 + (1 * (Picture1.Height / 5))
Line2.X1 = 0
Line2.X2 = 0 + Picture1.Width
Line2.Y1 = 0 + (2 * (Picture1.Height / 5))
Line2.Y2 = 0 + (2 * (Picture1.Height / 5))
Line3.X1 = 0
Line3.X2 = 0 + Picture1.Width
Line3.Y1 = 0 + (3 * (Picture1.Height / 5))
Line3.Y2 = 0 + (3 * (Picture1.Height / 5))
Line4.X1 = 0
Line4.X2 = 0 + Picture1.Width
Line4.Y1 = 0 + (4 * (Picture1.Height / 5))
Line4.Y2 = 0 + (4 * (Picture1.Height / 5))

'vertikal rpm
Line5.X1 = 0 + (1 * (Picture1.Width / 10))
Line5.X2 = 0 + (1 * (Picture1.Width / 10))
Line5.Y1 = 0 + Picture1.Height
Line5.Y2 = 0
Line6.X1 = 0 + (2 * (Picture1.Width / 10))
Line6.X2 = 0 + (2 * (Picture1.Width / 10))
Line6.Y1 = 0 + Picture1.Height
Line6.Y2 = 0
Line7.X1 = 0 + (3 * (Picture1.Width / 10))
Line7.X2 = 0 + (3 * (Picture1.Width / 10))
Line7.Y1 = 0 + Picture1.Height
Line7.Y2 = 0
Line8.X1 = 0 + (4 * (Picture1.Width / 10))
Line8.X2 = 0 + (4 * (Picture1.Width / 10))
Line8.Y1 = 0 + Picture1.Height
Line8.Y2 = 0
Line9.X1 = 0 + (5 * (Picture1.Width / 10))
Line9.X2 = 0 + (5 * (Picture1.Width / 10))
Line9.Y1 = 0 + Picture1.Height
Line9.Y2 = 0
Line10.X1 = 0 + (6 * (Picture1.Width / 10))
Line10.X2 = 0 + (6 * (Picture1.Width / 10))
Line10.Y1 = 0 + Picture1.Height
Line10.Y2 = 0
Line11.X1 = 0 + (7 * (Picture1.Width / 10))
Line11.X2 = 0 + (7 * (Picture1.Width / 10))
Line11.Y1 = 0 + Picture1.Height
```

```
Line11.Y2 = 0
Line12.X1 = 0 + (8 * (Picture1.Width / 10))
Line12.X2 = 0 + (8 * (Picture1.Width / 10))
Line12.Y1 = 0 + Picture1.Height
Line12.Y2 = 0
Line13.X1 = 0 + (9 * (Picture1.Width / 10))
Line13.X2 = 0 + (9 * (Picture1.Width / 10))
Line13.Y1 = 0 + Picture1.Height
Line13.Y2 = 0
End Sub

Private Sub waktu_rpm()
Label17.Caption = t_rpm / 10 * 1 + t2
Label18.Caption = t_rpm / 10 * 2 + t2
Label19.Caption = t_rpm / 10 * 3 + t2
Label20.Caption = t_rpm / 10 * 4 + t2
Label21.Caption = t_rpm / 10 * 5 + t2
Label22.Caption = t_rpm / 10 * 6 + t2
Label23.Caption = t_rpm / 10 * 7 + t2
Label24.Caption = t_rpm / 10 * 8 + t2
Label25.Caption = t_rpm / 10 * 9 + t2
Label26.Caption = t_rpm / 10 * 10 + t2
End Sub

Private Sub pid()
On Error Resume Next

sv2 = sv * 1023 / 1962
pv2 = pv * 1023 / 1962

error = sv2 - pv2

outp = kp * error 'nilai P

error_i = error + error_sblm_i
outi = ki * error_i * tc 'nilai I
error_sblm_i = error_i 'geser nilai untuk selanjutnya

error_d = error + error_sblm_d
outd = (kd * error_d) / tc 'nilai D
error_sblm_d = error_d 'geser nilai untuk selanjutnya

outpid = Round(outp + outi + outd) 'nilai pid siap kirim ke mk

End Sub

Private Sub kontrol_sederhana()

If (sv - pv > 0) Then
pwm = Val(pwm) + 1
Else
pwm = Val(pwm) - 1
End If

If Val(pwm) > 1023 Then
pwm = 1023
If Val(pwm) < 0 Then
pwm = 0
End If
End If

End Sub

Private Sub Timer2_Timer()
If MSComm1.PortOpen = True Then

kontrol_sederhana
MSComm1.Output = "a" + pwm
End If
End Sub
Private Sub Timer3_Timer()
If MSComm1.PortOpen = True Then
pid
MSComm1.Output = "a" + outpid 'Chr$(outpid)
End If

End Sub
```

LAMPIRAN IV

LISTING PROGRAM

MATLAB

```
%nilai pole yang ditentukan dari gambar root locus

%s1=-9.4+3.26i
s1=-4.58+3.31i



KI=[0.05 0.1 1 10]


plant_num=[0 3.492 24.4 0.5313];
plant_den=[1 9.173 32.05 0.5376];


s1mag = abs(s1)
beta = angle(s1)
plant_al = polyval(plant_num,s1)/polyval(plant_den,s1);
plants1mag = abs(plant_al)
psi = angle(plant_al)
%t=0:2:20:300;
t=10
for k =1:4

KP = -sin(beta+psi)/(plants1mag*sin(beta))-2*KI(k)*cos(beta)/s1mag
nilai_KI = KI(k)
KD = sin(psi)/(s1mag*plants1mag*sin(beta))+KI(k)/s1mag^2

Gcnum = [KD KP KI(k)];
Gcden = [0 1 0];

Tnum = conv(plant_num,Gcnum);
Tden = conv(plant_den,Gcden) + conv(plant_num,Gcnum);

r = roots(Tden)

nilai_pid=['KP=',num2str(KP),' KI=',num2str(KI(k)),' KD=',num2str(KD)]
cell_pid(k)=cellstr(nilai_pid)

step (Tnum,Tden,t)

if k==4
cell_pid(k+1)=cellstr('Tanpa PID')
step (plant_num,plant_den,t)
end

legend(cell_pid,'Location','Best')
hold on
end


hold off

figure, rlocus(Tnum,Tden)
```