SKRIPSI

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

Disusun oleh :

FIPIT PIBRIANTO

NIM 02.12.024

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
OKTOBER 2008

OKTOBER 2008
INSTITUT TEKNOLOGI MARDIGUNA
FAKULTAS TEKNOLOGI INDUSTRI
KONSENTRASI TEKNIK ENERGI LISTRIK
JURUSAN TEKNIK ELEKTRO S-1

NIM 03.41.019
RIKI PRASTIYATO
Diajukan oleh :

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR KECEPATAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

SKRIPSI

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi
Persyaratan Guna Mencapai Gelar Sarjana Teknik Elektro (S-1)

Disusun Oleh :

FIPIT PIBRIANTO

02.12.024

Malang, Oktober 2008

Diperiksa dan disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. Widodo Pudji M.,MT)	**(Ir. Eko Nurcahyo)**
NIP. Y. 1028700171	**NIP. Y. 1028700172**

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

(Ir. F. Yudi Limpraptono, MT)
NIP. Y. 1039500274

KONSENTRASI TEKNIK ENERGI LISTRIK
JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2008

ABSTRAKSI

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

(Fipit Pibrianto, Nim : 02.12.024, T. Elektro Energi Listrik S-1)
(Dosen Pembimbing : Ir. Widodo Pudji M, MT dan Ir. Eko Nurcahyo)
E-mail : Fp_che@yahoo.com

Salah satu jenis kontrol yang sering digunakan dalam sistem otomatisasi industri adalah kontrol PID. Jenis kontrol ini memiliki algoritma sederhana, namun telah terbukti memiliki performa yang baik. Namun dalam pengembangannya, penerapan algoritma jenis kontrol ini masih sangat jarang diterapkan di mikrokontroler. Padahal, secara teoritis IC ini telah dilengkapi oleh beberapa fasilitas yang mampu mendukung algoritma tersebut diterapkan hanya dengan menggunakan satu ragam tunggal IC mikrokontroler.

Salah satu contoh aplikasinya yaitu alat pengatur putaran motor DC berbasis mikrokontroler Atmega8535. Dengan adanya fasilitas internal yang berupa ADC (Analog to Digital Converter) dan PWM (Pulse Width Modulation) di dalam ATmega8535, maka IC ini dapat secara langsung membaca sensor putaran di ADC kemudian melakukan algoritma PID di mikroprosessornya dan menyimpan data hasil perhitungannya di memori data, serta menentukan besarnya tegangan yang diterima motor DC agar putarannya sesuai dengan yang diinginkan dengan menggunakan metode PWM.

Dari hasil rancangan alat yang telah dibuat dan diuji, diketahui bahwa karakteristik motor uji sebelum menggunakan PID kontrol yaitu memiliki waktu mati (L) sebesar 40ms dan waktu tunda (T) sebesar 120ms. Sedangkan setelah digunakannya kontrol PID, waktu tunda tersebut menjadi lebih cepat yaitu sebesar 40ms. Hal ini membuktikan bahwa kontrol PID mampu meningkatkan respon motor untuk mencapai kondisi steady state yang lebih cepat.

Kata kunci : pengatur putaran motor DC, algoritma PID kontrol, mikrokontroller ATmega8535.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat rahmat dan hidayah-Nya penulis dapat menyelesaikan Laporan Skripsi ini. Laporan Skripsi ini penulis susun dari hasil selama menempuh studi/kuliah dan mengikuti praktikum di ITN Malang.

Laporan Skripsi ini merupakan salah satu syarat wajib bagi mahasiswa Jurusan Teknik Energi Listrik S-1 dalam memperoleh gelar Sarjana Teknik.

Dalam penyelesaian Laporan Skripsi ini, banyak pihak yang telah membantu penulis baik dalam melakukan penelitian maupun dalam penulisan, sehingga lapopran skripsi ini dapat tersusun dan terselesaikan. Untuk itu atas segala bimbingan dan fasilitas yang telah diberikan kepada penulis, tak lupa penulis menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Ir. F YUDI LIMPRAPTONO, MT, selaku Ketua Jurusan Teknik Elektro S-1 Institut Teknologi Nasional Malang.
2. Bapak Ir. YUSUF ISMAIL NAKHODA, MT, selaku Sekretaris Jurusan Teknik Elektro S-1 Institut Teknologi Nasional Malang.
3. Bapak Ir. WIDODO PUDJI M, MT selaku Dosen Pembimbing I selama penulis menyelesaikan Skripsi.
4. Bapak Ir. EKO NURCAHYO selaku Dosen Pembimbing II selama penulis menyelesaikan Skripsi.
5. Seluruh Dosen pengajar di ITN Malang khususnya Jurusan Teknil Elektro S-1.

6. Seluruh karyawan di ITN Malang khususnya Jurusan Teknik Elektro S-1.
7. Kepada Orang Tua saya yang selama ini memberikan semangat dan dorongan yang tak ternilai harganya baik dalam bentuk material maupun spirituil, serta kasih sayang selama ini yang tak terbatas kepada penulis.
8. Spesial To My Soulmate (Isteriku Tercinta) atas semua dorongan, semangat dan pengertiannya yang tak henti-hentinya, dan kasih sayang yang tak terbatas.
9. Kepada seluruh saudara baik kakak, adik, paman, bibi, pakde, bude dan keluarga besar penulis yang telah memberikan semangat dan dorongan serta pinjaman komputer, jas, kemeja, dasi dan masih banyak lagi kapada penulis selama ini.
10. Thanx My Bro Alfan Sujatmiko yang selama ini telah banyak membantu, entah kapan penulis dapat membalasnya, karena terlalu besar and banyak. Dan teman penulis yang lain seperti Mansyur Isfahan, Mas Roby, Henda, Roby, Jupiter and all my frend yang gak bisa disebut namanya satu persatu yang selama ini memberikan semangat dan dorongan kepada penulis.

Akhirnya penulis berharap, semoga Laporan Skripsi ini dapat bermanfaat bagi semua pembaca, khususnya bagi mahasiswa Jurusan teknik Energi Listrik S-1.

Malang, September 2008

Penulis.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR GRAFIK

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi khususnya di bidang industri dewasa ini telah membawa perubahan dan kemajuan bagi peradaban kehidupan umat manusia, dimana perkembangan teknologi tersebut telah mendorong manusia untuk membuat inovasi baru. Salah satu perkembangan teknologi yang bisa kita temukan dibidang industri saat ini adalah peralatan yang mampu beroperasi secara otomatis dengan kinerja yang maksimal. Adanya satu system control yang baik akan dapat menunjang proses berjalannya industri tersebut dan untuk meningkatkan efisiensi dalam hal industri.

Dalam suatu system control dikenal pula istilah kontrol PID. Kontrol PID merupakan umpan balik yang paling popular didalam dunia industri saat ini. Kontroler PID terbukti telah dapat memberikan performasi kontrol yang baik meskipun memiliki algoritma yang sederhana dan sangat mudah dipahami. Kontroler PID pun dikenal sebagai kontroler klasik yang dapat menghasilkan performasi yang paling baik.

Oleh sebab itu skripsi ini menguraikan desain kontrol PID (Proporsional Integral Differensial) dengan menggunakan perangkat keras Mikrokontroler ATmega8535 suatu system otomatisasi yang bekerja secara kontinyu. Dalam hal ini dirancang suatu simulator system yang dapat mengendalikan putaran motor DC agar tetap stabil (konstan).

1.2. Rumusan Masalah

Dalam penulisan judul ini penulis mencoba merumuskan masalahnya sebagai berikut:

- Bagaimana merancang hardware dan software suatu alat yang dapat mengatur putaran Motor DC dengan pendekatan PID kontroller menggunakan mikrokontroler ATmega8535?

1.3. Tujuan Penulisan

Tujuan dari penulisan skripsi ini adalah bagaimana merancang dan merealisasi sistem pengatur putaran Motor DC tersebut dengan pendekatan PID kontroller yang berbasis Mikrokontroler ATmega8535.

1.4. Batasan Masalah

Penulis membatasi ruang lingkup dalam skripsi ini dalam batasan masalah. Adapun batasan masalah tersebut adalah sebagai berikut :

a) Perancangan pengontrolan Algoritma PID.

b) Motor DC yang digunakan :
   - Tegangan 24 V.
   - Arus 1 A.
   - Putaran 1800 rpm.

c) Kontroler menggunakan mikrokontroler ATmega8535.

d) Variabel yang diteliti yaitu putaran motor DC dengan mengatur tegangan masukan motor.

1.5. Metodologi Penelitian

Desain proyek berupa perangkat keras (hard ware) dan perangkat lunak (software):

a. Perancangan Perangkat Keras Dan Perangkat Lunak.

   Perancangan perangkat keras dibutuhkan Mikrokontroler, Driver Motor, Motor DC, Sensor Infrared dan Photodioda. Untuk mengaktifkan hubungan perangkat keras tersebut maka dibutuhkan perangkat lunak didalam IC Mikrokontroler Atmega8535 yang mana perangkat lunaknya ditulis dalam bahasa pemrograman asembler.

b. Pengukuran dan pengujian perangkat keras dan lunak

   Dari hasil perancangan dilakukan realisasi pembuatan baik perangkat keras maupun pada perangkat lunak, dan dilakukan pengukuran/ pengujian masing-masing bagian (sub-sistem) dari perangkat-perangkat tersebut.

c. Pengujian Sistem

   Sistem yang sudah dibangun, dianggap memadahi sehingga dapat digunakan untuk mengendalikan putaran pada Motor DC.

1.6. Sistematika Pembahasan

Agar pembaca lebih cepat memahami alur dari laporan ini maka sistematika pembahasan dalam laporan akhir perlu penulis kemukakan. Adapun sistematikanya adalah sebagai berikut:

- **BAB I PENDAHULUAN**

  Bab ini berisikan uraian latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi dan sistematika pembahasan.

- **BAB II LANDASAN TEORI**

  Mengulas tentang teori dasar system mikrokontroler ATmega8535, Motor DC, dan Kontrol PID.

- **BAB III PERANCANGAN DAN PEMBUATAN ALAT**

  Bab ini berisikan tentang tahap-tahap perencanaan, pembuatan perangkat keras dan lunak.

- **BAB IV PENGUJIAN DAN ANALISA ALAT**

  Bab ini berisikan tentang laporan hasil pengujian alat di labolatorium.

- **BAB V KESIMPULAN DAN SARAN**

  Bab ini berisi kesimpulan dari pembahasan sebelumnya, serta saran-saran yang memungkinkan untuk pengembangan dari skripsi lebih lanjut.

BAB II

DASAR TEORI

2.1. Mikrokontroler AVR ATmega8535

2.1.1. Pendahuluan AVR

AVR : Alf and Vegard RISC atau

AVR : Advanced Virtual RISC

RISC: Reduced Instruction Set Computer

Arsitektur mikrokontroler jenis AVR pertamakali dikembangkan pada tahun 1996 oleh dua orang mahasiswa Norwegian Institute of Technology yaitu Alf-Egil Bogen dan Vegard Wollan. Mikrokontroler AVR kemudian dikembangkan lebih lanjut oleh Atmel. Seri pertama AVR yang dikeluarkan adalah mikrokontroler 8 bit AT90S8515, dengan konfigurasi pin yang sama dengan mikrokontroler 8051, termasuk address dan data bus yang termultipleksi. Mikrokontroler AVR menggunakan teknologi RISC dimana set instruksinya dikurangi dari segi ukurannya dan kompleksitas mode pengalamatannya. Pada awal era industri komputer, bahasa pemrograman masih menggunakan kode mesin dan bahasa assembly. Untuk mempermudah dalam pemrograman para desainer komputer kemudian mengembangkan bahasa pemrograman tingkat tinggi yang mudah dipahami manusia. Namun akibatnya, instruksi yang ada menjadi semakin komplek dan membutuhkan lebih banyak memori. Dan tentu saja siklus eksekusi instruksinya menjadi semakin lama. Dalam AVR dengan arsitektur RISC 8 bit, semua instruksi berukuran 16 bit dan sebagian besar dieksekusi dalam 1

siklus clock. Berbeda dengan mikrokontroler MCS-51 yang instruksinya bervariasi antara 8 bit sampai 32 bit dan dieksekusi selama 1 sampai 4 siklus mesin, dimana 1 siklus mesin membutuhkan 12 periode clock.

Dalam perkembangannya, AVR dibagi menjadi beberapa varian yaitu AT90Sxx, ATMega, AT86RFxx dan ATTiny. Pada dasarnya yang membedakan masing-masing varian adalah kapasitas memori dan beberapa fitur tambahan saja.

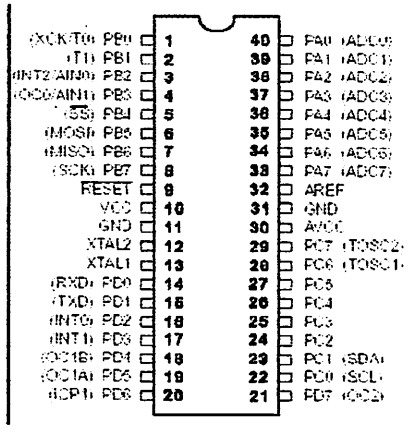
2.1.2. Karakteristik mikrokontroler AVR seri ATMega8535

2.1.2.1. Fitur ATMega8535

Fitur yang tersedia pada ATMega 8535 adalah :

- Frekuensi clock maksimum 16 MHz
- Jalur I/O 32 buah, yang terbagi dalam PortA, PortB, PortC dan PortD
- Analog to Digital Converter 10 bit sebanyak 8 input
- Timer/Counter sebanyak 3 buah
- CPU 8 bit yang terdiri dari 32 register
- Watchdog Timer dengan osilator internal
- SRAM sebesar 512 byte
- Memori Flash sebesar 8 Kbyte dengan kemampuan *read while write*
- Interrupt internal maupun eksternal
- Port komunikasi SPI
- EEPROM sebesar 512 byte yang dapat diprogram saat operasi
- Analog Comparator
- Komunikasi serial standar USART dengan kecepatan maksimal 2,5 Mbps

2.1.2.2. Konfigurasi pin ATMega8535



Gambar 2.1
Konfigurasi Pin ATmega8535[2]

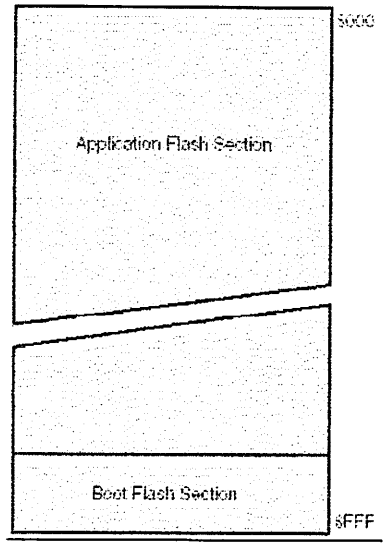
2.1.2.3. Peta memori ATMega8535

ATMega8535 memiliki dua jenis memori yaitu Data Memory dan Program Memory ditambah satu fitur tambahan yaitu EEPROM Memory untuk penyimpan data.

- *Program Memory*

  ATMega8535 memiliki On-Chip In-System Reprogrammable Flash Memory untuk menyimpan program. Untuk alasan keamanan, program memory dibagi menjadi dua bagian yaitu Boot Flash Section dan Application Flash Section. Boot Flash Section digunakan untuk menyimpan program Boot Loader, yaitu program yang harus dijalankan pada saat AVR reset atau pertamakali diaktifkan. Application Flash Section digunakan untuk menyimpan program aplikasi yang dibuat user. AVR tidak dapat menjalankan program aplikasi ini sebelum menjalankan program Boot Loader. Besarnya memori Boot Flash Section dapat diprogram dari 128 word sampai 1024 word tergantung setting

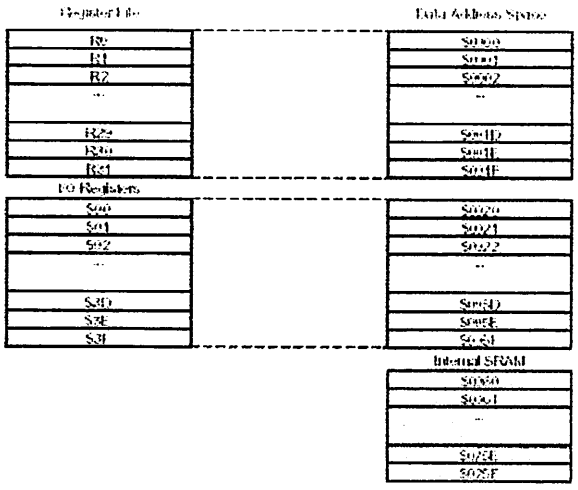
pada konfigurasi bit di register BOOTSZ. Jika Boot Loader diproteksi, maka program pada Application Flash Section juga sudah aman.



Gambar 2.2
Peta Program Memori ATmega8535[2]

- *Data Memory*

Gambar berikut menunjukkan peta memori SRAM pada ATMega8535. Terdapat 608 lokasi address data memori. 96 lokasi address digunakan untuk Register File dan I/O Memory sementara 512 lokasi address lainnya digunakan untuk internal data SRAM. Register File terdiri dari 32 general purpose working register, I/O register terdiri dari 64 register.



Gambar 2.3
Peta Data Memori ATmega8535[2]

- *EEPROM Data Memory*

  ATMega8535 memiliki EEPROM sebesar 512 byte untuk menyimpan data. Lokasinya terpisah dengan sistem address register, data register dan control register yang dibuat khusus untuk EEPROM.

2.1.3. Status Register (SREG)

Status Register adalah register yang memberikan informasi yang dihasilkan dari eksekusi instuksi aritmatika. Informasi ini berguna untuk mencari alternatif alur program sesuai dengan kondisi yang dihadapi.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – I : Global Interrupt Enable

Jika bit Global Interrupt Enable diset, maka fasilitas interupsi dapat dijalankan. Bit ini akan clear ketika ada interrupt yang dipicu dari hardware,

setelah program interrupt dieksekusi, maka bit ini harus di set kembali dengan instruksi SEI.

Bit 6 – T : Bit Copy Storage

Instruksi bit copy BLD dan BST menggunakan bit T sebagai sumber atau tujuan dalam operasi bit.

Bit 5 – H : Half Carry Flag

Bit 4 – S : Sign Bit

Bit S merupakan hasil exlusive or dari Negative Flag N dan Two's Complement Overflow Flag V.

Bit 3 – V : Two's Complement Overflow Flag

Digunakan dalam operasi aritmatika

Bit 2 – N : Negative Flag

Jika operasi aritmatika menghasilkan bilangan negatif, maka bit ini akan set.

Bit 1 – Z : Zero Flag

Jika operasi aritmatika menghaslkan bilangan nol, maka bit ini akan set.

2.1.4 Register I/O

Setiap port ATMega8535 terdiri dari 3 register I/O yaitu DDRx, Portx dan PINx.

- DDRx (Data Direction Register)

  Register DDRx digunakan untuk memilih arah pin. Jika DDRx = "1" maka Pxn sebagai pin output Jika DDRx = "0" maka Pxn sebagai input.

- Portx (Port Data Register)

Register Portx digunakan untuk 2 keperluan yaitu untuk jalur output atau untuk mengaktifkan resistor pullup.

1. Portx berfungsi sebagai output jika DDRx = "1" maka :

   Portxn = "1" maka pin Pxn akan berlogika high.

   Portxn = "0" maka pin Pxn akan berlogika low.

2. Portx berfungsi untuk mengaktifkan resistor pullup jika DDRx = "0" maka:

   Portxn = "1" maka pin Pxn sebagai pin input dengan resistor pull up.

   Portxn = "0" maka pin Pxn sebagai output tanpa resistor pull up.

Tabel 2.1
Konfigurasi Port ATmega8535[2]

DDRxn	Portxn	I/O	Pull up	Comment
0	0	Input	No	Tri state (Hi-Z)
0	1	Input	Yes	Pull up aktif
1	0	Output	No	Output Low
1	1	Output	No	Output High

Catatan :

x menunjukkan nama port (A,B,C,D)

n menunjukkan nomor bit (0,1,2,3,4,5,6,7)

Nilai awal *(initial value)* seluruh register I/O adalah 00h.

- PINx (Port Input Pin Address)

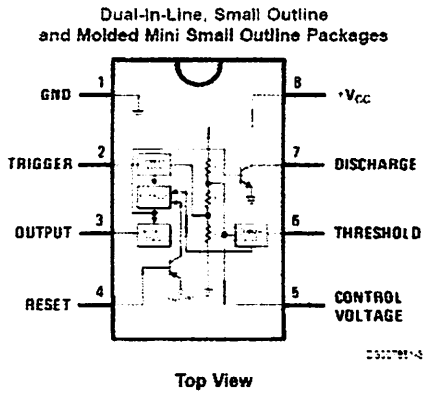
  Digunakan sebagai register input.

2.2. Tachometer

Tachometer merupakan peralatan pengukur putaran motor. Alat ini terdiri dari komponen pengirim sinyal yang dibentuk dari IC Timer LM555 dan komponen penerima sinyal dari IC LM567.

2.2.1. IC Timer LM555

IC LM555 merupakan komponen yang memiliki kestabilan tinggi sebagai pembangkit waktu dan osilator. Pada mode pembangkit waktu, waktu dapat diatur secara presisi hanya dengan menggunakan satu buah reistor dan kapasitor. Sedangkan pada mode pembangkit gelombang (osilator), frekuensi dan *duty cycle* secara akurat dapat diatur dengan hanya menggunakan 2 buah resistor dan 1 buah kapasitor. Adapun susunan kaki-kaki dari IC LM555 sesuai dengan Gambar 2.4.



Gambar 2.4
Susunan kaki-kaki IC LM555[8]

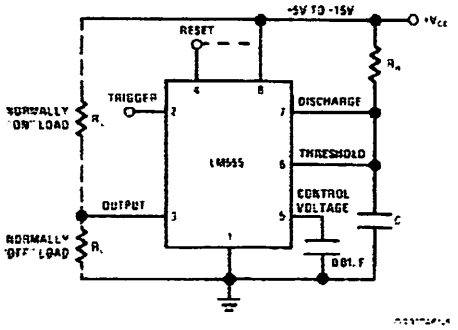
2.2.1.1. *One-Shoot Monostable Operation*

Pada jenis pengoperasian ini, IC LM555 difungsikan dengan cara memberikan pulsa negatif atau tegangan maksimal 1/3 dari Vcc (logika "0") ke kaki *trigger* (pin 2). Ketika IC ini diaktifkan, rangkaian flip-flop di dalam

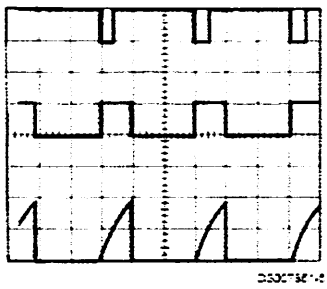
IC akan terhubung dengan kapasitor dan menghasilkan logika "high" di keluarannya. Tegangan yang melewati kapasitor akan meningkat selama :

$$t = 1.1xR_A.C$$

Sampai tegangan tersebut sebesar 2/3Vcc. Setelah itu, komparator akan melakukan reset terhadap rangkaian flip-flop sehingga keluarannya berlogika "low". Rangkaian ini sesuai dengan Gambar 2.2., sedangkan gambar gelombangnya ditunjukkan pada Gambar 2.3.



Gambar 2.5
Monostable operation IC LM555[8]



V_{CC} = 5V
TIME = 0.1 ms/Div.
R_A = 9.1kΩ
C = 0.01uF

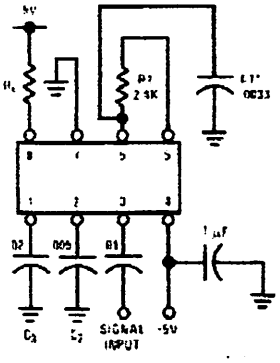
Top Trace: Input 5V/Div.
Middle Trace: Output 5V/Div.
Bottom Trace: Capacitor Voltage 2V/Div.

Gambar 2.6
Bentuk gelombang *monostable* IC LM555[8]

2.2.1.2. Free-Running Monostable Operation

Gambar rangkaian dari jenis operasi ini hampir sama dengan operasi *one-shoot monostable*, perbedaannya hanya di jalur pin 2 (*trigger*). Pada jenis pengoperasian ini, pin 2 terhubung ke pin 6. Dengan adanya hubungan ini, maka jalur *trigger* dapat berosilasi secara terus-menerus.

2.2.2. IC LM567



Gambar 2.7
Rangkaian IC LM567 sebagai detektor frekuensi[8]

IC ini akan mengeluarkan logika "0" pada pin 8 ketika frekuensi masukkan (pin 3) melebihi frekuensi acuannya (frekuensi tengah). Adapun besarnya frekuensi tengah (f_o) ditentukan berdasarkan rumusan sebagai berikut :

$$f_o = \frac{1}{1.1 x R_1 x C_1}$$

2.3. Motor DC

Motor arus searah, sebagaimana namanya, mengunakan arus langsung yang tidak *langsung/direct-undiretional.* Motor DC digunakan pada penggunaan khusus dimana diperlukan penyalaan *torque* yang tinggi atau percepatan yang tetap untuk kisaran kecepatan yang luas.

Sebuah motor DC yang memiliki tiga komponen utama:

- *Kutub medan,* secara sederhana digambarkan bahwa interaksi dua kutub magnet akan menyebabkan perputaran pada motor DC. Motor DC memiliki kutub medan yang stasioner dan dinamo yang menggerakan *bearing* pada ruang diantara kutub medan. Motor DC sederhana memiliki dua kutub medan, kutub utara dan kutub selatan. Garis magnetik energi membesar melintasi bukaan diantara kutub-kutub dari utara ke selatan. Untuk motor yang lebih besar atau lebih komplek terdapat satu atau lebih elektromagnet. Elektromagnet menerima listrik dari sumber daya dari luar sebagai penyedia struktur medan.
- *Dinamo,* bila arus masuk menuju dinamo, maka arus ini akan menjadi elektromagnet. Dinamo yang berbentuk silinder, dihubungkan ke as penggerak untuk menggerakan beban. Untuk kasus motor DC yang kecil, dinamo berputar dalam medan magnet yang dibentuk oleh kutub-kutub, sampai kutub utara dan selatan magnet berganti lokasi. Jika hal ini terjadi, arusnya berbalik untuk merubah kutub-kutub utara dan selatan dinamo.
- *Commutator,* komponen ini terutama ditemukan dalam motor DC. Kegunaannya adalah untuk membalikan arah arus listrik dalam dinamo.

Commutator juga membantu dalam transmisi arus antara dinamo dan sumber daya.

2.3.1. Prinsip Kerja Motor DC

Pada motor DC, kumparan medan yang dialiri arus listrik akan menghasilkan medan magnet yang melingkupi kumparan jangkar dengan arah tertentu. Converter energi baik energi listrik menjadi energi mekanik (motor) maupun sebaliknya dari energi mekanik menjadi energi listrik (generator) berlangsung melalui medium medan magnet. Energi yang akan diubah dari suatu sistem ke sistem yang lain, sementara akan tersimpan pada medium medan magnet untuk kemudian dilepaskan menjadi energi system lainnya. Dengan demikian, medan disini selain berfungsi sebagai tempat penyimpan energi juga sekaligus proses perubahan energi, dimana proses perubahan energi pada motor arus searah dapat digambarkan pada gambar.

Gambar 2.8
Proses Konversi Energi Pada Motor DC

Dengan mengingat hukum kekekalan energi, proses konversi energi listrik menjadi energi mekanik dapat dinyatakan sebagai berikut :

Energi listrik sebagai *input* = Energi mekanik sebagai *output* + energi yang diubah menjadi panas + Energi ang tersimpan dalam medan magnet.

2.3.2. Model Matematik Motor DC

Prinsip kerja motor DC berdasarkan pada penghantar yang dialiri arus ditempatkan dalam suatu medan magnet sehingga penghantar tersebut akan mengalami gaya. Gaya menimbulkan torsi sehingga menghasilkan putaran. Penghantar yang berputar akan menimbulkan tegangan AC sehingga diubah menjadi tegangan DC oleh komutator dan sikat.

Gaya yang dihasilkan sebesar:

$$F = B^* i^* l \tag{2.1}$$

Bila jari-jari rotor adalah r, maka torsi yang akan dibangkitkan adalah:

$$T = F^* r = B^* i^* l^* r \tag{2.2}$$

Keterangan:

F = *Gaya (N)*

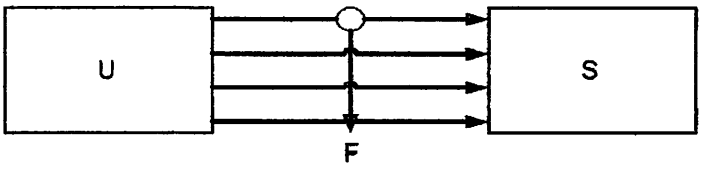
B = *Rapat Flux (T)*

i = *Arus yang mengalir pada penghantar (A)*

l = *Panjang penghantar (m)*

r = *Jari-jari inti jangkar (m)*

T = *Torsi (Nm)*



Gambar 2.9
Arah Gaya Pada Motor DC

Pada saat gaya F dibangkitkan, konduktor bergerak didalam medan magnet dan akan menimbulkan gaya gerak listrik (GGL) yang merupakan reaksi (lawan) terhadap tegangan penyebabnya. Agar proses konversi energi listrik menjadi energi mekanik (motor) dapat berlangsung, tegangan sumber harus lebih besar dari gaya gerak listrik lawan. Torsi akan memutar rotor bila yang terbangkit telah memiliki torsi lawan dari motor dan beban.

Telah diketahui bahwa untuk motor arus searah dapat diturunkan rumus sebagai berikut:

$$Vt = Ea + Ia.Ra \tag{2.3}$$

$$Ea = k.n.\phi \tag{2.4}$$

Keterangan:

Vt = Tegangan jangkar (V)

Ea = Gaya gerak listrik lawan (V0

Ia = Arus jangkar (A)

Ra = Tahanan Jangkar (Ω)

n = Putaran (Rpm)

ϕ = Fluks/kutub

k = Konstanta

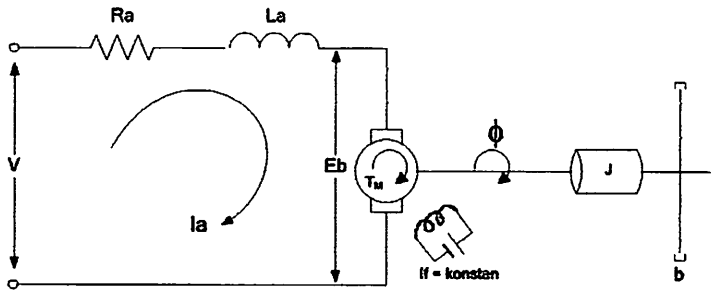
Berdasarkan rumus diatas diturunkan rumus kecepatan putar (n), yaitu:

$$n = \frac{Vt - Ia.Ra}{k.\phi} \tag{2.5}$$

Dari persamaan diatas, dapat dilihat bahwa kecepatan putaran (n) motor DC dapat diatur dengan mengubah-ubah besarnya nilai Vt (tegangan jangkar), Ra (tahanan jangkar), dan ϕ (fluks magnet).

Banyak terdapat jenis motor yang digunakan sebagai *plant* untuk sistem kontrol industri. Salah satu diantaranya adalah Motor DC magnet permanen. Motor DC magnet permanen adalah motor yang medan magnet utamanya berasal dari magnet permanen. Dan kumparan medan elektromagnetik digunakan untuk medan jangkar. Sistem operasi dari motor DC magnet permanen yaitu arus yang mengalir melalui kumparan jangkar dari sumber tegangan DC, menyebabkan jangkar berfungsi sebagai magnet. Kutub pada kumparan jangkar akan ditarik oleh kutub medan utama dari polaritas yang berbeda, sehingga jangkar berputar. Apabila kutub jangkar segaris dengan kutub medan, sikat-sikat berada pada celah dikomutator, sehingga tidak ada arus yang mengalir pada jangkar. Jadi, gaya tarik atau tolak dari magnet akan berhenti. Kemudian kelembaman membawa jangkar melewati titik netral. Komutator akan membalik arus jangkar ketika kutub yang tidak sama dari jangkar dan medan saling berhadapan satu sama lain, sehingga membalik polaritas medan jangkar. Kutub-kutub yang sama dari jangkar dan medan kemudian menjadi saling tolak menolak, sehingga jangkar akan berputar terus menerus. Arah putaran dari motor DC magnet permanen ditentukan oleh arah arus yang mengalir pada jangkar. Pembalikan ujung-ujung jangkar tidak akan membalik arah putaran. Kecepatan motor DC magnet permanent berbanding langsung dengan besar tegangan yang diberikan dijangkar. Semakin besar tegangan jangkar maka semakin tinggi kecepatan putaran motor.

Dengan mengatur tegangan dan fluks magnetnya tetap, diharapkan dapat menghasilkan torsi yang diinginkan agar dapat menghasilkan output putaran motor mendekati *setting point*. Untuk mengetahui rangkaian ekivalen motor DC dengan pengontrolan jangkar dapat dilihat pada gambar dibawah ini.



Gambar 2.10
Rangkaian Ekivalen Motor DC Pengotrolan Jangkar[7]

Keterangan:

V = Tegangan jangkar (V)

Eb = Gaya gerak listrik (V)

Ia = Arus jangkar (A)

Ra = Tahanan kumparan jangkar (Ω)

La = Induksi kumparan jangkar (*henry*)

n = Putaran jangkar (Rpm)

ϕ = Fluks magnet per kutub (*weber*)

ω_m = Kecepatan jangkar (*rad/sec*)

J = Momen inersia total dari motor dan beban pada poros motor ($Kg.m^2$)

b = Koefisien gesekan viskos ekivalen dari motor dan beban pada poros motor ($N.m(rad/sec)^{-1}$)

T_M = Torsi yang diberikan oleh motor (N.m)

Besarnya GGL lawan (Eb) yang dibngkitkan oleh motor adalah:

$$Eb = Ke.\ \phi.\ \omega_m \tag{2.6}$$

Dan tegangan jangkar (Vt) adalah:

$$V = Eb + Ra.\ Ia \tag{2.7}$$

Serta torsi motor (T_M) yang diberikan adalah:

$$T_M = Ke.\ \phi.\ Ia \tag{2.8}$$

Dari persamaan (2.6), (2.7) dan (2.8) dapat ditentukan persamaan kecepatan jangkar (ω_m), yaitu:

$$\omega_m = \frac{V}{Ke.\phi} - \frac{Ra.\,Ia}{Ke.\phi} = \frac{1}{Ke.\varphi}\left(V - Ra.Ia\right) \tag{2.9}$$

$$\omega_m = \frac{V}{Ke.\phi} - \frac{Ra.\,T_M}{(Ke.\phi)^2} \tag{2.10}$$

Persamaan tegangan (V) dari rangkaian jangkar pada kondisi dinamik diberikan dalam bentuk:

$$V = Ra.\ Ia + La\ \frac{dla}{dt} + Eb \tag{2.11}$$

Karena,

$$Eb = Ke.\ \phi.\ \omega_m \tag{2.12}$$

Dan,

$$Ke.\ \phi = K_B. \tag{2.13}$$

Dimana, K_B konstanta gaya gerak listrik balik, maka:

$$Eb = K_B.\ \omega_m \tag{2.14}$$

Sehingga,

$$V = Ra.\ Ia + La\ \frac{dIa}{dt} + K_{B.}\ \omega_m \tag{2.15}$$

Dalam bentuk *Transformasi Laplace* dapat ditulis:

$$V\ (s) = Ra.\ Ia\ (s) + La.\ S\ Ia\ (s) + K_{B.}\ \omega_m\ (s) \tag{2.16}$$

Berdasarkan persamaan (2.16) dapat diturunkan persamaan arus jangkar (Ia), yaitu:

$$Ia(s) = \frac{1}{Ra + LaS}[V(s) - K_B.\omega_m(s)] \tag{2.17}$$

Jadi time konstan rangkaian jangkar (rotor) adalah:

$$\tau_a = \frac{La}{Ra} \tag{2.18}$$

Maka,

$$Ia(s) = \frac{1}{Ra(1 + S\tau_a)} V(s) - K_B.\omega_m(s) \tag{2.19}$$

Sehingga,

$$Ia(s) = \frac{1/Ra}{Ra(1 + S\tau_a)} V(s) - K_B.\omega_m(s) \tag{2.20}$$

Persamaan sistem motor berbeban dalam kondisi dinamik diberikan dalam bentuk:

$$J\frac{d\omega_m}{dt} = T_M - T_L - B.\omega_m \tag{2.21}$$

Maka torsi:

$$T_M = J\frac{d\omega_m}{dt} + B.\omega_m + T_L \tag{2.22}$$

$$T_M - T_L = J\frac{d\omega_m}{dt} + B.\omega_m \tag{2.23}$$

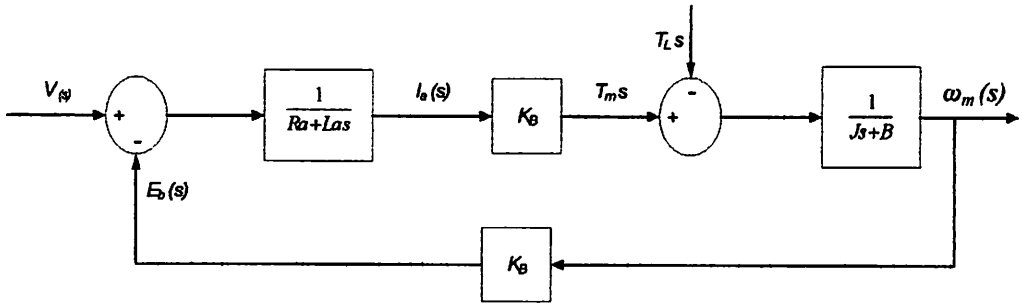
Dalam bentuk transformasi *Laplace* dapat ditulis:

$$T_M(s) - T_L(s) = J.S(s) + B.\omega_m(s) \tag{2.24}$$

$$T_M(s) - T_L(s) = (J.S + B).\omega_m(s) \tag{2.25}$$

Berdasarkan persamaan (2.25) dapat diturunkan persamaan kecepatan jangkar (ω_m), yaitu:

$$\omega_m(s) = \frac{1}{J.S+B}[T_M(s) - T_L(s)] \tag{2.26}$$



Gambar 2.11
Diagram Blok Motor DC Penguat Terpisah Dengan
Pengontrolan Jangkar[7]

Dari rangkaian *loop* diatas dapat diketahui bahwa motor berkarateristik seperti sebuah sistem *close loop*. Dengan demikian dapat dikatakan bahwa pengontrolan motor DC dengan pengaturan tegangan jangkar menjadikan motor tersebut sistemnya *close loop*.

2.4. Metode Kontrol

Keberadaan kontroller dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadap perilaku sistem. Pada prinsipnya hal itu disebabkan oleh tidak dapat diubahnya komponen penyusun sistem tersebut. Artinya, karakteristik *plant* harus diterima sebagaiamana adanya, sehingga perubahan perilaku sistem hanya dapat dilakukan melalui penambahan suatu sub sistem, yaitu kontroller.

Ada banyak macam pengendali akhir *(final control element)*. Secara umum mereka terbagi menjadi dua bagian, yaitu *final control element* untuk pengendalian on-off dan *final control element* untuk pengendalian kontinyu. Unit kontrol pada sistem pengendalian on-off hanya akan bekerja pada posisi terbuka penuh dan tertutup penuh. sedangkan pada sistem pengendalian kontinyu, unit kontrol akan bekerja dari titik 0% sampai ketitik 100%. Oleh karena itu jenis pengendali ini lebih banyak digunakan dalam sistem pengendalian proses.

Ada tiga jenis pengendali kontinyu, yaitu pengendali proporsional disingkat P, pengendali Integral disingkat I, dan pengendali Diferensial disingkat D. Karena ketiga pengendali tersebut memiliki kekurangan dan kelebihan, maka seringkali dipakai dalam bentuk kombinasi P+I+D disingkat PID.

Pada dasarnya tugas dari komponen kontroller adalah mereduksi sinyal kesalahan, yaitu perbedaan antara sinyal setting dan sinyal aktual. Hal ini sesuai dengan tujuan sistem kontrol adalah mendapatkan sinyal aktual (diinginkan) senantiasa sama dengan sinyal setting. Semakin cepat reaksi sistem mengikuti sinyal aktual dan semakin kecil kesalahan yang terjadi, maka semakin baiklah kinerja sistem kontrol yang diterapkan.

Apabila perbedaan antara nilai setting dengan nilai keluaran relative besar, maka kita harus mengamati perbedaan ini untuk segera menghasilkan sinyal keluaran untuk mempengaruhi *plant*. Dengan demikian sistem secara cepat mengubah keluaran *plant* sampai diperoleh selisih antara setting dengan besaran yang diatur sekecil mungkin.

2.4.1. Kontroller Proporsional

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

Untuk kontroller dengan aksi kontrol proporsional, hubungan antara masukan kontroller u(t) dan sinyal kesalahan c(t) adalah:

$u(t) = Kp.\ e(t)$

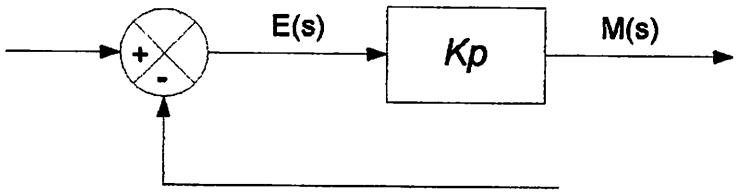
keterangan : Kp = Penguatan Proporsional

Sedangkan persamaan dalam transformasi Laplace:

$$\frac{U(s)}{E(s)} = Kp$$

Gambar 2.12. menunjukkan blok diagram yang menggambarkan hubungan antara besaran setting, besaran aktual dengan besaran keluaran kontroller proporsioanal. Sinyal kesalahan (error) merupakan selisih antara besaran setting dengan besaran aktualnya. Selisih ini akan mempengaruhi kontroller, untuk

mengeluarkan sinyal positif (mempercepat pencapaian harga setting) atau negative (memperlambat tercapainya harga yang diinginkan).



Gambar 2.12
Diagram Blok Kontroller Proporsional[7]

Kontroller proporsional memiliki 2 parameter, yaitu pita proporsional *(proporsional band)* dan konstanta proporsional. Daerah kerja kontroller efektif dicerminkan oleh pita proporsioanl, sedangkan konstanta proporsioanl menunjukkan nilai faktor penguatan terhadap sinyal kesalahan.

Hubungan antara pita proporsional (PB) dengan konstanta proporsional (Kp) ditunjukkan secara prosentase oleh persamaan berikut:

$$PB = \frac{1}{Kp} x\,100\%$$

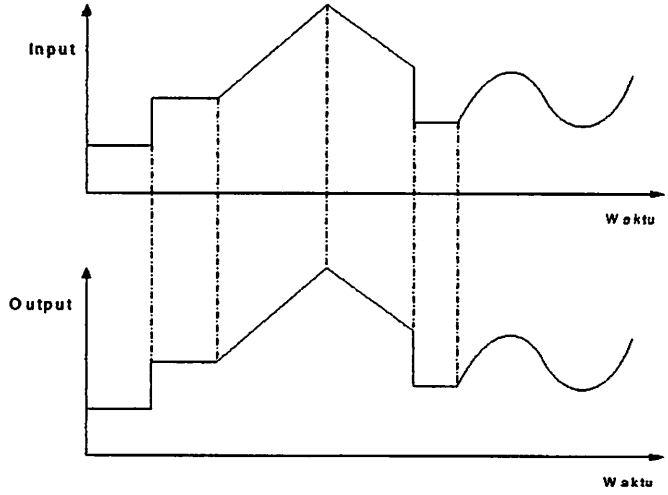
Dimana :

PB = Proporsional Band

Kp = Konstanta Proporsional

Walaupun hubungan input-output unit kontrol proporsional bukan merupakan funsi waktu, untuk bahan perbandingan dengan unit kontrol lain, ada baiknya kalau hubungan itu dinyatakan dalam bentuk kurva fungsi waktu. Dari gambar 2.13. jelas terlihat bahwa output selalu mengikuti input secara proporsional. Naik turunnya input diikuti secara langsung oleh output, dan besarnya selalu sama dengan input kali gain. Karena unit control proporsional ini

bukan fungsi waktu, dinamik gain pengendali ini sama dengan *steady state* gainnya. Dengan kata lain, besarnya gain tidak tergantung pada besarnya frekuensi loop.



Gambar 2.13
Response Sebuah Pengendali Proporsional[6]

Ciri-ciri kontroller proporsional harus diperhatikan ketika kontroller tersebut diterapkan pada suatu sistem. Secara eksperimen, pengguna kontroller proporsional harus memperhatikan ketentuan-ketentuan berikut ini:

1. Kalau nilai Kp kecil, kontroller proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat.
2. Kalau nilai Kp dinaikkan, respon sistem menunjukkan semakin cepat mencapai keadaan mantapnya.
3. Namun jika nilai Kp diperbesar sehingga mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil, atau respon sistem akan berosilasi.

Pengendali proporsional ini memiliki kekurangan yaitu, dimana dalam proses pengedalian ini meninggalkan error yang lazim disebut offset. Offset yang ditinggalkan oleh pengendali proporsional ini tidak dapat diperkecil begitu saja dengan memperbesar gain, karena dengan bertambahnya gain, kemungkinan besar sistem menjadi tidak stabil. Untuk menghilangkan adanya error pada pengendali proporsional ini, maka besaran bias diperlukan untuk mempertahankan output pada waktu error sama dengan nol. Salah satu cara untuk menghilangkan offset adalah dengan menyetel bias agar sedekat mungkin dengan load, namun cara ini tidak dapat direkomendasikan begitu saja, karena tidak semua kontroller memiliki fasilitas *adjustable bias*.

2.4.2 Kontroller Integral

Untuk menghilangkan adanya offset pada pengendali proporsional, diperlukannya pengendali lain yang dapat mengeluarkan output walaupun padanya tidak diperlukan input. Dengan kata lain diperlukan pengendali yang dapat menghasilkan output lebih besar atau lebih kecil dari bias, pada saat input (error) sama dengan nol. Pengendali yang memiliki kriteria ini adalah pengendali integral disingkat I.

Sifat dasar pengendali integral yang dapat mengeluarkan output pada saat input sama dengan nol, adalah sifat sebuah unit integrator.

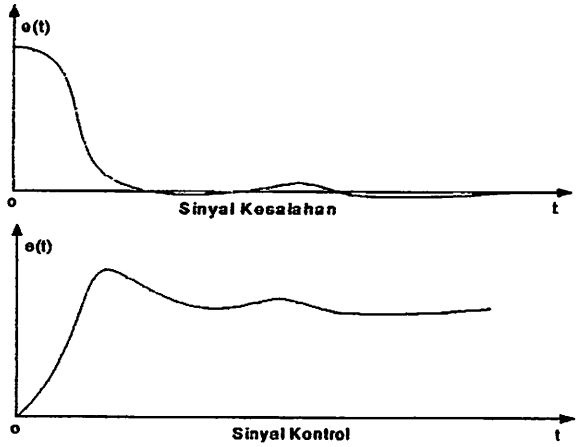
Kontroller integral berfungsi untuk menghasilkan respon sistem yang memiliki kesalahan keadaan mantapnya nol. Kalau sebuah plant tidak memiliki unsur integrator (1/s), kontroller proporsional tidak akan mampu menjamin

keluaran sistem dengan kesalahan keadaan mantapnya nol. Dengan kontroller integral, respon sistem dapat diperbaiki, yaitu mempunyai keadaan mantapnya nol.

Kontroller integral memiliki karakteristik seperti halnya sebuah integaral. Keluaran kontroller sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan.

Keluaran kontroller ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaa seperti sebelum terjadinya perubahan masukan.

Sinyal keluaran kontroller integral merupakan luas bidang yang dibentuk oleh kurva ksalahan penggerak (lihat konsep numerik). Sinyal keluaran akan berharga sama dengan harga sebelumnya ketika sinyal kesalahan berharga nol. Gambar 2.14. menujukkan contoh sinyal kesalahan yang disulutkan kedalam kontroller integral dan keluaran kontroller integral terhadap perubahan sinyal kesalahan tersebut.



Gambar 2.14
Kurva Sinyal Kesalahan e(t) Terhadap t dan Kurva u(t) Terhadap t Pada Pembangkit Kesalahan nol[6]

Pada kendali integral, nilai masukan kontroller u(t) diubah pada laju proporsional dari sinyal pembangkit kesalahan e(t), sehingga:

$$\frac{du(t)}{dt} = K_i\, e(t)$$

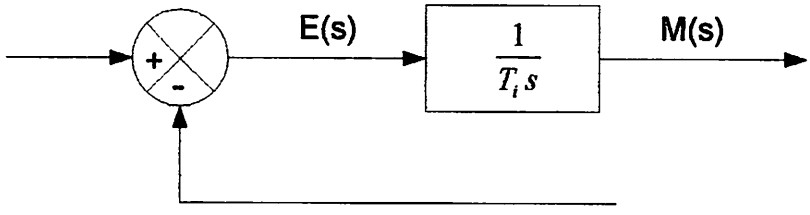
Atau

$$u(t) = K_i \int_0^t e(t)\,dt$$

Fungsi alih dari kontroller integral, adalah:

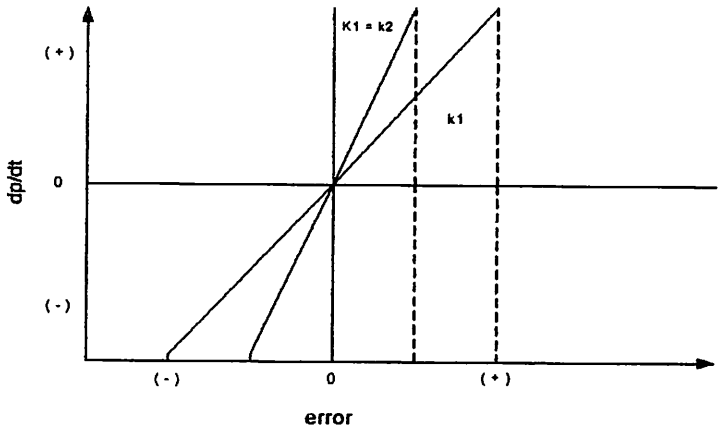
$$\frac{U(s)}{E(s)} = \frac{K_i}{s}$$

Gambar 2.15. menunjukkan blok diagram antara besaran kesalahan dengan keluaran suatu kontroller integral.



Gambar 2.15
Blok Diagram Hubungan Antara Besaran Kesalahan Dengan Kontroller Integral[7]

Pengaruh perubahan konstanta integral terhadap keluaran integral ditunjukan oleh Gambar 2.16. Ketika sinyal kesalahan berlipat ganda, maka nilai laju perubahan keluaran kontroller berubah menjadi dua kali dari semula. Jika nilai konstanta integrator berubah menjadi lebih besar, sinyal kesalahan yang relative kecil dapat mengakibatkan laju keluaran menjadi besar.



Gambar 2.16
Perubahan Keluaran Sebagai Akibat
Penguatan Dan Kesalahan[7]

Ketika digunakan, kontroller integral mempunyai beberapa karakteristik berikut ini:

1. Keluaran kontroller membutuhkan selang waktu tertentu, sehingga kontroler integral cenderung memperlambat respon.
2. Ketika sinyal kesalahan berharga nol, keluaran kontroller akan bertahan pada nilai sebelumnya.
3. Jika siinyal tidak berharga nol, keluaran akan menunjukan kenaikan atau penurunan yang dipengaruhioleh besarnya sinyal kesalahan dan nilai Ki.
4. Konstanta integral Ki yang berharga besar akan mempercepat hilangnya *offset.* Tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran kontroler.

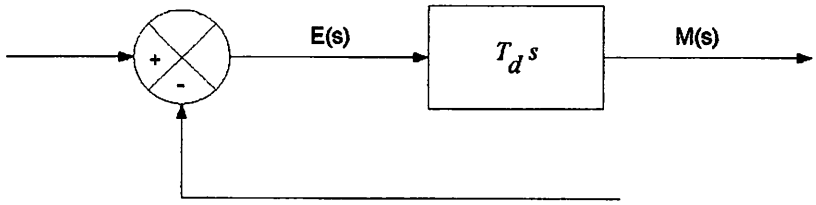
2.4.3. Kontroller Differensial

Aksi kendali diferensial atau kendali kecepatan dipakai untuk mempercepat tanggapan dinamik sistem kendali, yaitu mengantisipasi sinyal kesalahan yang

akan terjadi dengan memperhatikan laju perubahan kesalahan. Dengan memberikan aksi kendali antisipator, maka proses terkontrol cenderung stabil, sehingga sering digunakan untuk melawan pengaruh osilasi yang disebabkan oleh aksi integral.

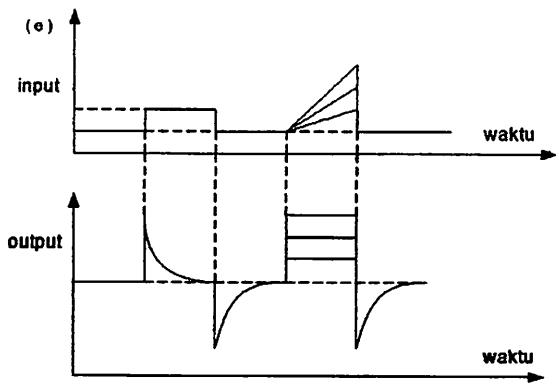
Perubahan yang mendadak pada masukan kontroler, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.17. menunjukan blok diagram yang menggambarkan hubungan antara sinyal kesalahan dengan keluaran kontroller.

Keluaran kontroller diferensial memiliki sifat seperti halnya suatu operasi derivatitif.



Gambar 2.17
Blok Diagram Kontroller Diferensial[7]

Gambar 2.17. menyatakan hubungan antara sinyal masukan dengan sinyal keluaran kontroller diferensial. Ketika masukannya tidak mengalami perubahan, keluaran kontroler juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk impuls. Jika sinyal masukan berubah naik secara perlahan (fungsi *ramp*), keluarannya justru merupakan funsi step yang besar magnitudnya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan faktor konstanta diferensialnya Td.



Gambar 2.18
Kurva Waktu Hubungan Input-Output
Kontroller Diferensial[7]

Karakteristik kontroller diferensial adalah sebagai berikut:

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

Berdasarkan karateristik kontroller tersebut, kontroller diferensial umumnya dipakai untuk mempercepat respon awal suatu sistem, tetapi tidak memperkecil kesalahan pada keadaan tunaknya. Kerja kontroller diferensial hanyalah efektif pada lingkup yang sempit, yaitu pada periode peralihan. Oleh

sebab itu kontroller diferensial tidak pernah digunakan tanpa ada kontroller lain pada sebuah sistem.

2.4.4. Kontroller PID

Sistem kontrol PID memiliki umpan balik yang merupakan gabungan dari tiga mode kontroller yaitu kontroller proporsional, kontroller integral, kontroller diferensial. Kontroller proporsional dalam hal ini berfungsi untuk mempercepat tanggapan awal sistem (*rise time/waktu bangkit*), kontroller derivative untuk mempercepat tanggapan dinamik dan controller integral untuk menghilangkan offset. Kontroller integral pada sistem kontrol PID digunakan, jika sistem harus menjaga proses variable pada nilai pengoperasian nominal, dimana perubahan variable proses hanya akan terjdi sebagai akibat perubahan beban.

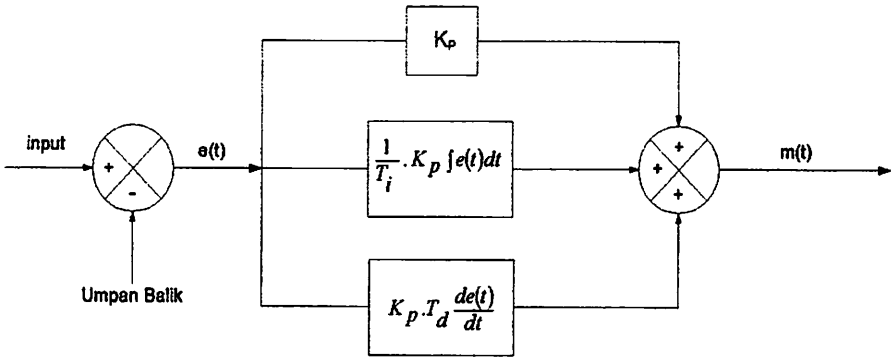
Setiap kekurangan dan kelebihan dari masing-mabsing kontroller P, I dan D dapat saling menutupi dengan menggabungkan ketiganya secara parallel menjadi kontroller proporsional plus integral plus deferensial (kontroller PID). Elemen-elemen kontroller P, I dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan offset dan menghasilkan perubahan awal yang besar. Persamaan dengan tiga kombinasi ini diberikan:

$$u(t) = K_P\, e(t) + \frac{K_P}{T_i}\int_0^t e(t)\, dt + K_P\, T_d\, \frac{de\, e(t)}{dt}$$

Sehingga fungsi alihnya adalah:

$$\frac{U(s)}{E(s)} = K_P\left[1 + T_d s + \frac{1}{T_i s}\right]$$

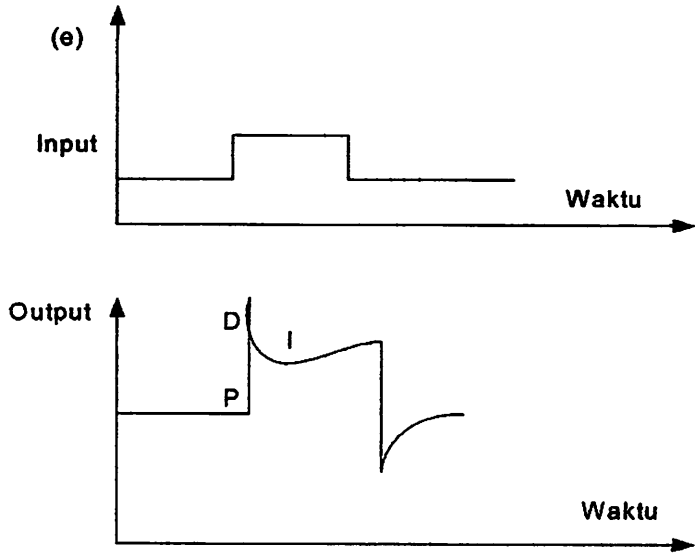
Gambar 2.19. menunjukkan blok diagram kontroller PID.



Gambar 2.19
Blok Diagram Kontroller PID[7]

Keluaran kontroller PID merupakan jumlahan dari keluaran kontroller proporsional, keluaran kontroller integral dan keluaran kontroller diferensial. Gambar 2.20. menunjukkan hubungan tersebut.

Dalam mendesain sustu sistem kontrol yang terpenting adalah spesifikasi atau kriteria performasi yang ditampilkan, dapat kita lihat pada gambar berikut :



Gambar 2.20
Hubungan Fungsi Waktu Antara Sinyal Input-Output Untuk Kontroller PID[6]

Karakteristik kontroller PID sangat dipengaruhi oleh kontribusi besar dari ketiga parameter P, Idan 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 disbanding yang lain. Konstanta yang menonjol itulah akan memberikan kontribusi pengaruh pada respon sistem secara keseluruhan.

Mode proporsional dalam kontroller PID menghasilkan sinyal keluaran yang proporsional terhadap perbedaan antara nilai aktual dan nilai set point. Kontrol integral berfungsi untuk menghasilkan keluaran yang proporsional dengan penjumlahan waktu dan lamanya waktu sinyal kesalahan timbul. Pengendali diferensial menghasilkan sinyal keluaran yang sebanding dengan laju perubahan sinyal kesalahan. Kontroller PID tepat digunakan pada situasi dimana kontroller harus mampu merespon perubahan sinyal analog yang sangat cepat, dan pada kasus yang terjadi kelambanan waktu yang panjang antara pemakaian aksi koreksi, munculnya perubahan yang nyata sebagai akibat dari pemberian aksi kendali.

Pengaturan dan penalaan kontroller PID untuk mendapatkan performasi yang optimal dilakukan dengan mencari kombinasi dari ketiga metode tersebut, ini dilakukan saat proses benar-benar berjalan. Penetapan awal sistem kendali diperkirakan, dan tiga mode ditala dengan baik untuk akhirnya dicapai penetapan kendali yang optimal.

BAB III

PERENCANAAN ALAT

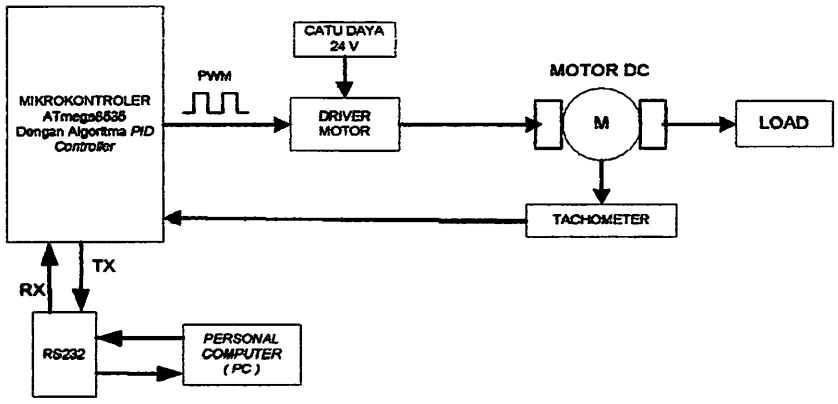
3.1. Pendahuluan

Dalam bab ini akan dibahas pembuatan seluruh sistem perangkat yang ada pada alat pengatur putaran motor DC dengan menggunakan algoritma PID kontroller, secara garis besar terdapat dua bagian perangkat yang ada yaitu:

1. Perencanaan perangkat keras
2. Perencanaan perangkat lunak

Pada perencanaan perangkat keras akan meliputi penjelasan dari perencanaan diagram blok sistem dan juga perencanaan minimum sistem mikrokontroller Atmega8535 yang digunakan pada perencanaan perangkat lunak. Akan tetapi perangkat tersebut dalam kerjanya akan saling mendukung satu dengan lainnya sehingga alat yang direncanakan dapat berjalan sesuai dengan perencanaannya.

3.2. Diagram Blok



Gambar 3.1
Diagram Blok Alat

3.2.1. Cara kerja alat

Putaran yang akan dihasilkan oleh motor DC ditentukan di PC, yang kemudian dikirim ke mikrokontroler ATmega8535. Besaran putaran dari PC tersebut merupakan nilai putaran acuan (*setting point*), sedangkan putaran yang terbaca oleh *tachometer* merupakan putaran aktual yang terjadi akibat ada atau tidak adanya *load.*

Agar nilai putaran aktual stabil dengan *setting point*, maka data putaran aktual dari *tachometer* diatur dengan metode PID kontrol yang telah dimasukkan secara program ke dalam IC mikrokontroler ATmega8535. Hasil perhitungan dengan PID kontrol tersebut berupa nilai PWM yang mempengaruhi cepat atau lambatnya motor berputar.

PWM (*Pulse Width Modulation*) adalah salah satu metode dalam mengendalikan tegangan yang masuk ke motor DC dengan cara mengatur durasi mati dan hidupnya motor tersebut. Mikrokontroler Atmega8535 telah dilengkapi oleh fasilitas PWM, sehingga durasi mati dan hidupnya motor tersebut cukup ditentukan hanya berdasarkan lamanya pulsa "high" untuk mengaktifkan motor dan pulsa "low" untuk mematikan motor DC.

Keluaran dari rangkaian *tachometer* merupakan sinyal analog sehingga jalur ini akan dihubungkan ke jalur ADC dari mikrokontroler Atmega8535. Dari data analog tersebut, oleh Atmega8535 akan dikonversi ke data putaran. Data putaran aktual yang terbaca oleh Atmega8535 akan dikirimkan secara terus menerus ke PC dan akan ditampilkan secara grafik sehingga kestabilan putaran motor tersebut dapat langsung terlihat secara jelas.

3.2.2. Jenis dan fungsi komponen alat

1. PERSONAL COMPUTER (PC)

   Berfungsi sebagai pengendali utama sistem dari alat dan menampilkan hasil pengukuran putaran motor DC dalam bentuk grafik.

2. RS232

   Berfungsi sebagai pengkondisi sinyal dari PC ke mikrokontroler Atmega8535 dengan menggunakan komunikasi serial.

3. MIKROKONTROLER ATmega8535

   Berfungsi sebagai pengendali besarnya putaran motor dengan menggunakan algoritma PID kontroller.

4. DRIVER MOTOR DC

   Berfungsi sebagai pengkondisi tegangan keluran dari mikrokontroller ATmega8535 sebelum digunakan untuk mengggerakkan motor DC.

5. CATU DAYA

   Catu daya berfungsi sebagai sumber energi bagi motor DC sebesar 24 Volt.

6. MOTOR DC

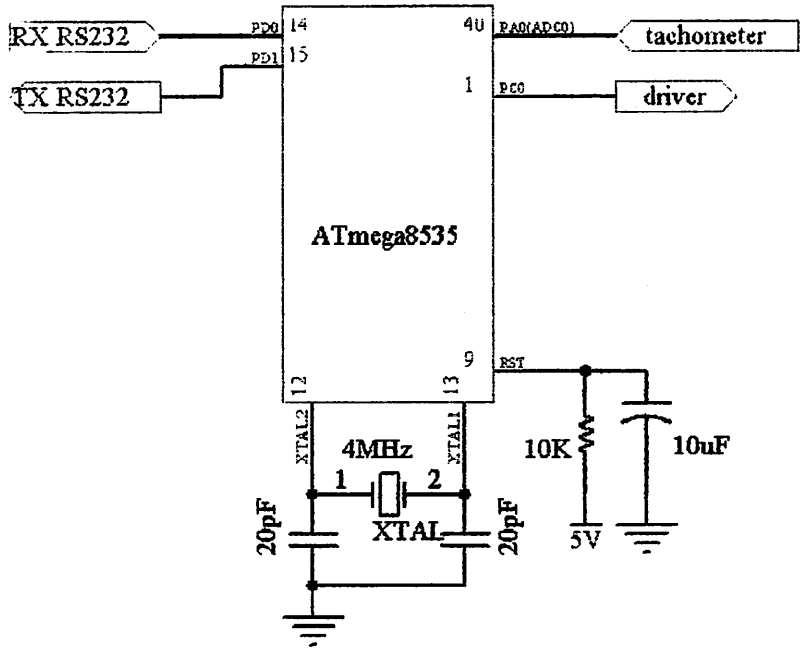
   Motor DC merupakan komponen yang dikendalikan.

7. TACHOMETER

   Tachometer merupakan komponen pendeteksi putaran motor DC. Komponen ini terdiri dari rangkaian pemancar dan penerima. Rangkaian pemancar merupakan rangkaian penghasil frekuensi yang melalui *infrared* sedangkan penerimanya yaitu menggunakan *photodioda*.

3.3. Perencanaan Perangkat Keras

3.3.1. Minimum Sistem Mikrokontroler Atmega8535



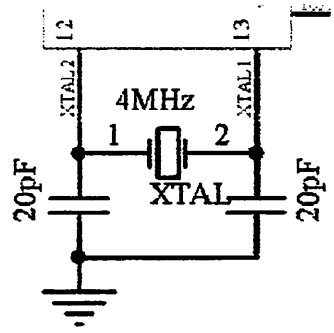
Gambar 3.2
Skematik Minimum Sistem Atmega8535[2]

Sub-sistem pembentuk sistem mikrokontroler ini adalah sebagai berikut :

- **Rangkaian *clock* (pin 12 dan 13)**

  Kecepatan proses yang dilakukan oleh mikrokontroler ditentukan oleh sumber *clock* (pewaktuan) yang mengendalikan mikrokontroler tersebut. Sistem yang dirancang akan menggunakan osilator internal yang sudah tersedia di dalam chip ATmega8535. Frekuensi osilator dibentuk dari komponen kristal dan kapasitor, berdasarkan *data sheet* mikrokontroler ATmega8535. Pada alat ini, *crystal* yang digunakan adalah 4MHz dan kapasitornya sebesar 20pF.

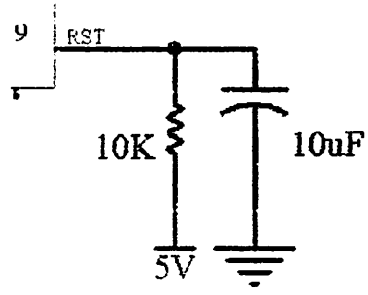
  Rangkaian *clock* (pewaktu) yang digunakan sesuai dengan Gambar 3.3.



Gambar 3.3

Rangkaian clock Mikrokontroler ATmega8535[2]

- **Rangkaian *Reset* (pin 9)**



Gambar 3.4

Rangkaian reset Mikrokontroler ATmega8535[2]

Rangkaian reset bertujuan agar mikrokontroler dapat menjalankan proses dari awal. Rangkaian *reset* untuk mikrokontroler dirancang agar mempunyai kemampuan *power on reset*, yaitu *reset* yang terjadi pada saat sistem dinyalakan untuk pertama kalinya.

Rangkaian *Reset* terbentuk oleh komponen R dan C. Nilai R yang dipakai adalah 10 kΩ dan C 10 µF.

Sedangkan untuk mencari frekuensi dari reset tersebut dengan menggunakan rumus sebagai berikut :

$$fo = \frac{1}{1{,}1R.C}$$

Sehingga dengan komponen resistor 10 Kohm dan kapasitor 10 uF akan dihasilkan frekuensi (f_o) sebagai berikut :

$$fo = \frac{1}{1{,}1.RC}$$

$$= \frac{1}{1{,}1x10.10^{3}x10.10^{-6}} \qquad = 9.09Hz$$

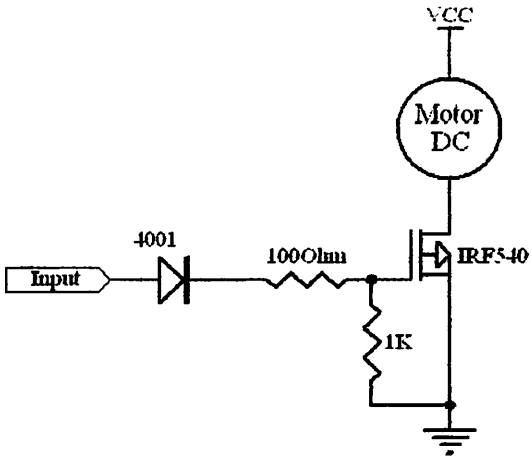
Maka Periode Clock $T = \frac{1}{fo}$

$$T = \frac{1}{9.09Hz} \quad = 0{,}11 \text{ detik} = 110\text{ms}$$

Sesuai dengan datasheet ATmega8535 telah dicantumkan bahwa $t_{reset(min)}$ adalah sebesar 1,5µs. Berdasarkan hasil perhitungan tersebut, maka komponen resistor 10 [Kohm] dan kapasitor 10 [uF] dapat membentuk waktu reset lebih besar daripada $t_{reset(min)}$ datasheet.

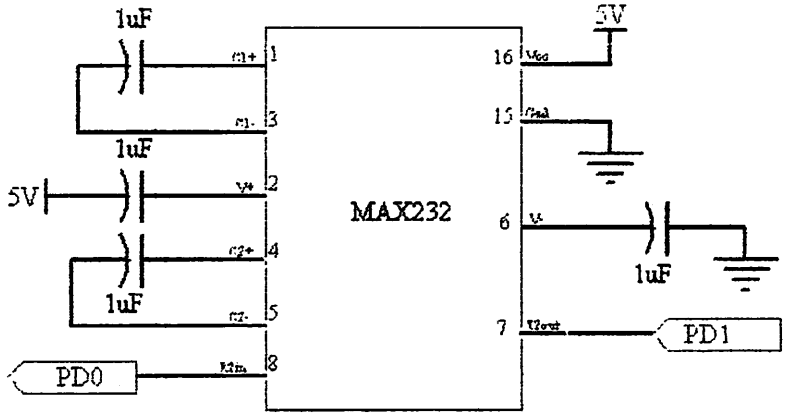
3.3.2. Rangkaian Driver Motor DC

Pada rangkaian driver motor disini menggunakan komponen MOSFETs type IRF540, adapun keunggulan komponen ini yaitu mampu mengaliri arus secara terus menerus sampai 28A dan tegangan maksimalnya 100V.



Gambar 3.5
Rangkaian Driver Motor DC

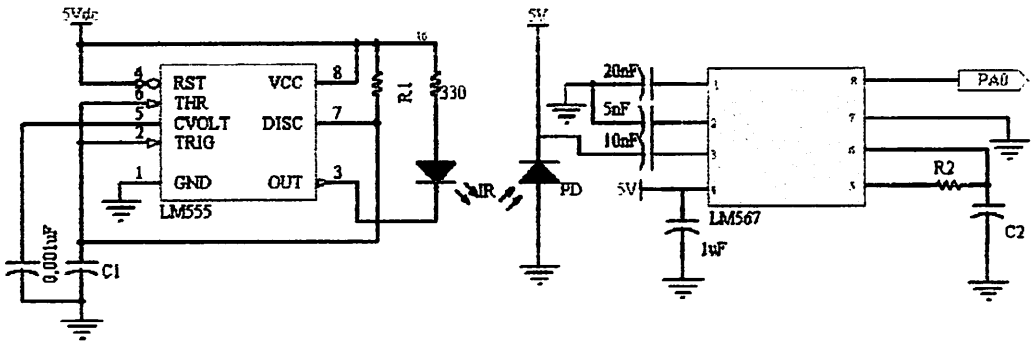
3.3.3. Rangkaian Serial RS232



Gambar 3.6
Rangkaian RS232[8]

Rangkaian ini berfungsi sebagai pengkondisi sinyal dari mikrokontroler ke PC, hal ini dikarenakan tegangan dari PC lebih besar daripada tegangan mikrokontroler. Nilai kapasitor yang digunakan dalam rangkaian ini sesuai dengan datasheet dari IC Max232.

3.3.4. Rangkaian Sensor Putaran



Gambar 3.7
Skematik Tachometer[8]

Rangkaian ini terdiri dari 2 bagian yaitu bagian pengirim dan penerima. Prinsip kerjanya yaitu bagian pengirim mengeluarkan sinyal infrared secara digital pada frekuensi 40KHz. Sedangkan bagian penerimanya yaitu berupa komponen photodioda akan menerima sinyal tersebut apabila tidak adanya penghalang antara infrared (IR) dengan photodioda (PD). Apabila frekuensi tersebut melebihi nilai frekuensi tengah yang telah ditentukan di IC LM567 maka, IC ini akan memberikan logika "0" ke mikrokontroler pada jalur PA0. Penentuan besarnya frekuensi pemancar didasarkan pada nilai R1 dan C1 pada IC Timer LM555. Adapun perhitungannya adalah sebagai berikut :

$$f_o = 40.000Hz$$
$$T = 1/f$$
$$T = 1/40.000$$
$$T = 0{,}000025s$$

Dari perioda tersebut (T), maka penentuan nilai R1 dan C1 pada IC LM555 dihitung hanya berdasarkan pulsa "1" saja dengan rumusan sebagai berikut :

$$t = T/2$$
$$t = 0{,}0000125s$$
$$t = 1{,}1xR1xC1$$

Maka :

$$R1xC1 = t/1{,}1$$
$$R1xC1 = 0{,}0000125/1{,}1$$
$$R1xC1 = 0{,}0000113$$

Dengan komponen C1 sebesar 10nF, maka nilai R1 adalah :

$$R1 = 0{,}0000113/10.10^{-9}$$
$$R1 = 1.130\Omega$$

Berdasarkan nilai yang ada di pasaran, maka resistor yang digunakan adalah sebesar 1KΩ.

Sedangkan penentuan frekuensi tengah komponen penerima IC LM567 dengan rumusan sebagai berikut :

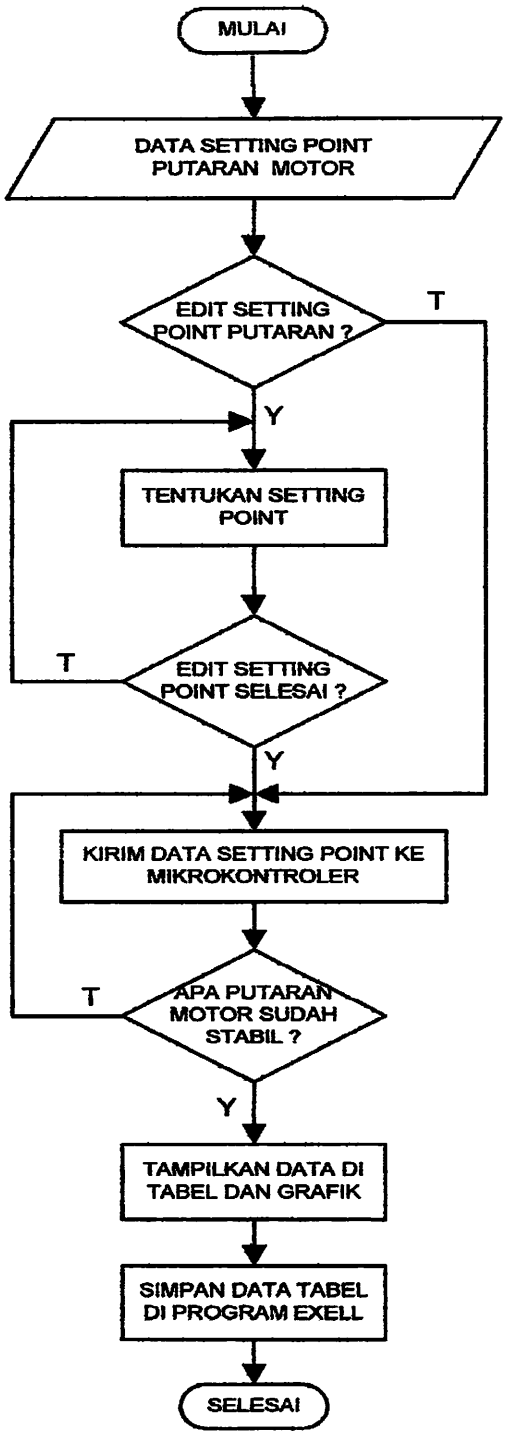
$$f_O = \frac{1}{1{,}1.R2.C2}$$

Berdasarkan nilai yang ada dipasaran, maka nilai C2 ditentukan terlebih dahulu sebesar 10nF. Sedangkan f_o sebesar $\frac{f_{IR}}{2} = \frac{40000}{2} = 20.000Hz$ dan nilai R2 adalah :

$$R2 = \frac{1}{1{,}1.f_O.C2}$$
$$R2 = \frac{1}{1{,}1x20.000x10x10^{-9}}$$
$$R2 = 4.545\Omega$$

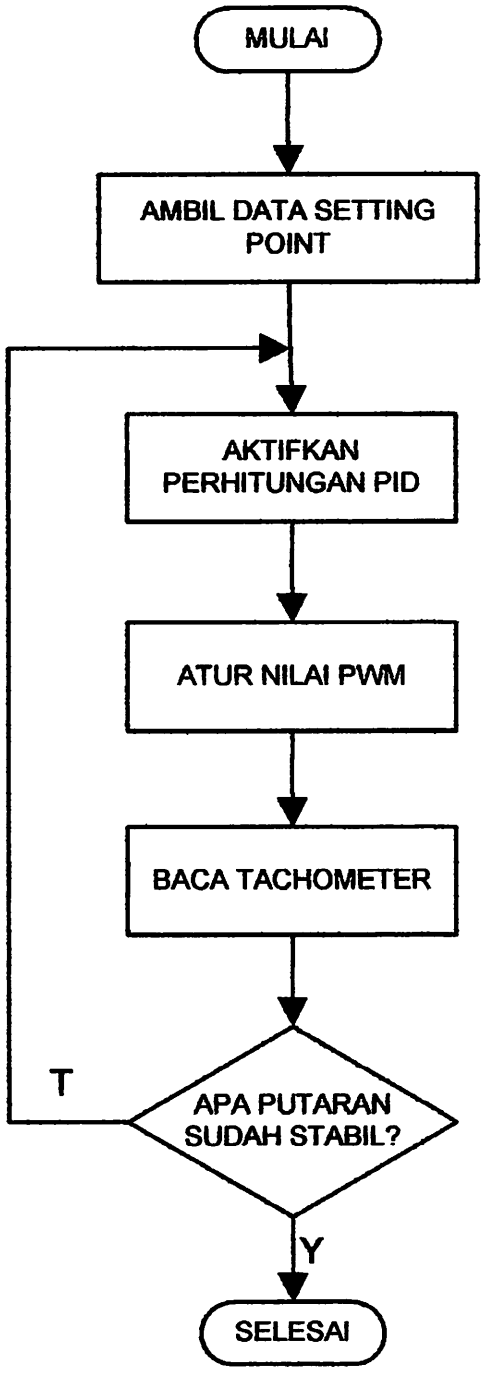
Dari perhitungan tersebut dan nilai yang ada dipasaran, maka resistor yang digunakan adalah sebesar 4K7.

3.4. Perencanaan Perangkat Lunak PC



Gambar 3.8
Diagram Alir PC

3.5. Perencanaan Perangkat Lunak Mikrokontroler ATmega8535



Gambar 3.9
Diagram Alir Mikrokontroler Atmega8535

BAB IV

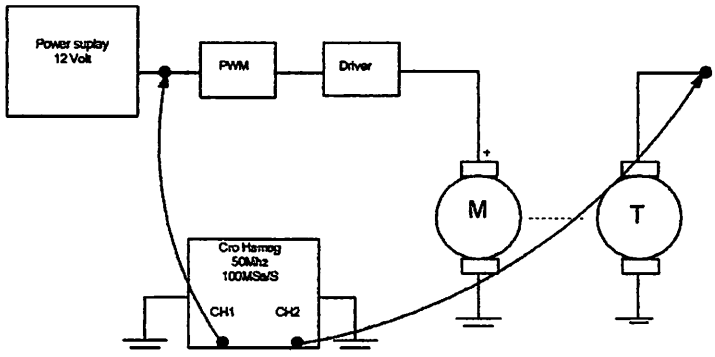
PENGUJIAN DAN ANALISA ALAT

4.1.Pengujian Identifikasi Motor DC

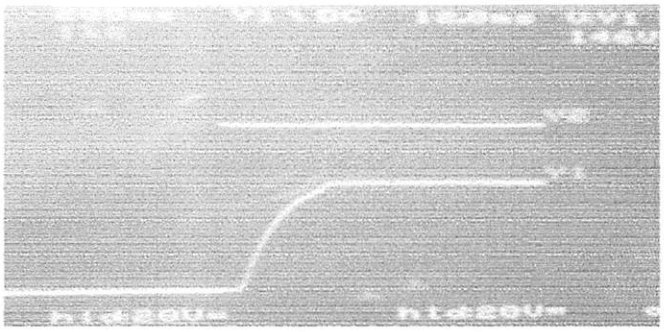
Kurva respon berbentuk S dapat dihasilkan secara eksperimen, oleh karena itu perlu sekali dilakukan proses identifikasi motor DC yang nantinya akan saling bersangkut paut dengan Metode Zighler-Nichols.

Karakteristik kurva yang dihasilkan dari proses identifikasi dapat diberikan dua konstanta yakni waktu tunda dan waktu mati yang ditentukan dengan menggambarkan garis singgung pada titik oerubahan kurva berbentuk S.

Dari proses identifikasi yang telah dilakukan maka didapat kurva S dari motor DC magnit permanent yang dipergunakan dalam laporan skripsi ini dengan data V/D = 5 Volt, T/D =200 ms. Dari identifikasi yang telah dilakukan, maka dapat diketahui bahwa respon motor DC (kurva S) yang digunakan sebagai *plant* adalah seperti gambar 4.2.

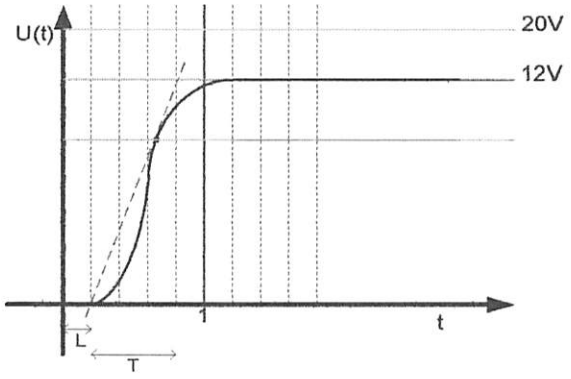


Gambar 4.1.
Pengujian Identifikasi Motor DC



Gambar 4.2.
Kurva S Identifikasi Motor DC

Sehingga dari respon motor DC, dapat dilakukan pendekatan melalui metode Ziegler-Nichols untuk menentukan waktu mati(L) dan waktu tunda(T). Dengan mengacu pada kurva S metode Ziegler_Nichols maka didapatkan data sebagai berikut :



Gambar 4.3.
Kurva S dengan metode Ziegler-Nichols

Dengan mengacu kepada kurva S metode ziegler nichols seperti gambar dan table dibawah ini maka didapatlah parameter-parameter Kp, Ki, Kd dari sistem kontroler PID yang dipergunakan

Tabel 4.1.
Parameter PID pada Kurva Reaksi

Tipe Kontroler	K_p	T_i	T_d
P	T/L	-	0
PI	0,9. T/L	L/0,3	0
PID	1,2.T/L	2.L	0,5.L

Dari gambar 4.3 dapat dijabarkan sebagai berikut:

L = 0,2 x 200ms = 40ms

T = 0,6 x 200ms = 120ms

Maka:

$K_p = 1{,}2 \cdot \frac{T}{L}$

$= 1{,}2 \cdot \frac{120}{40}$

$= 1{,}2 \cdot 3$

$K_p = 3{,}6$

$T_i = 2 \cdot L$

$= 2 \cdot 40ms$

$T_i = 80$ ms

$T_d = 0{,}5 \cdot L$

$= 0{,}5 \cdot 40ms$

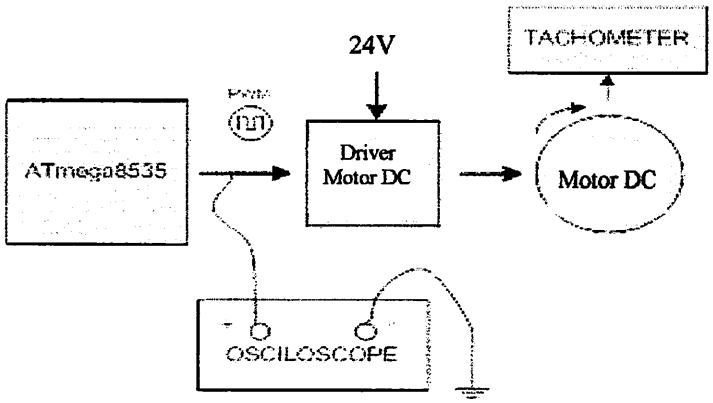
$T_d = 20$ ms

Tabel 4.2.
Hasil Perhitungan Parameter PID

Tipe kontroler	K_p	T_i	T_d
PID	3.6	80ms	20ms

4.2.Pengujian PWM

Peralatan pengujian sesuai dengan Gambar 4.4.



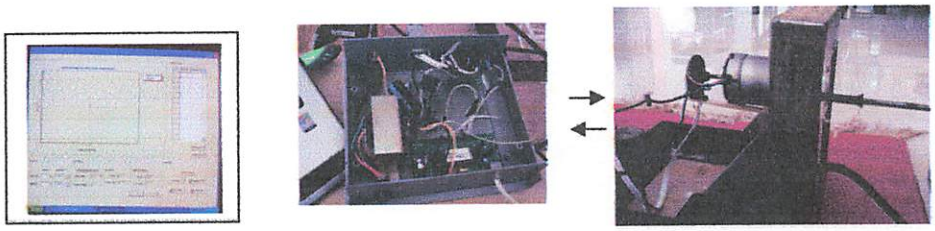
Gambar 4.4.
Pengujian PWM

Hasil pengujian ditampilkan sesuai dengan Gambar 4.5.

Gambar 4.5.
Hasil Pengujian PWM

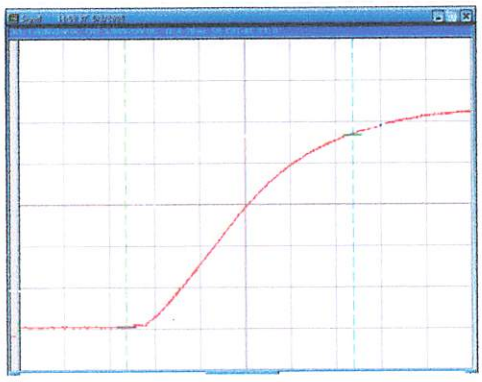
Dari hasil pengujian PWM, diketahui bahwa semakin lama durasi pulsa "high" (*on*), maka semakin cepat motor DC berputar, hal ini dikarenakan tegangan yang terjadi semakin besar.

4.3. Pengujian Keseluruhan



PC Atmega8535 Motor DC

Gambar 4.6.
Pengujian Keseluruhan



Gambar 4.7.
Kurva Respon Motor dengan PID Kontrol

Dari gambar 4.7. dengan parameter V/D = 5V dan T/d = 200ms maka didapat :

L = 0,1 x 200ms = 20ms

T = 0,2 x 200ms = 40ms

Dari perhitungan ini membuktikan bahwa kontrol PID mampu menguarngi waktu tunda (T) sehingga tanggapan respon motor semakin cepat dibandingkan dengan T tanpa kontrol PID yaitu sebesar 120ms.

Tabel 4.3.
Data Putaran Motor Tanpa Beban Tanpa PID
Dengan Set Point 1300 Rpm

Data	Set Point	Terbaca	Error
1	1300	1768	400
2	1300	872	428
3	1300	928	372
4	1300	936	364
5	1300	936	364
6	1300	936	364
7	1300	936	364
8	1300	944	356
9	1300	944	356
10	1300	936	364
11	1300	944	356
12	1300	944	356
13	1300	944	356
14	1300	944	356
15	1300	944	356
16	1300	944	356
17	1300	944	356
18	1300	952	348
19	1300	944	356
20	1300	952	348

Tabel 4.4.
Data Putaran Motor Tanpa Beban Tanpa PID
Dengan Set Point 1100 Rpm

Data	Set Point	Terbaca	Error
1	1100	1608	508
2	1100	632	468
3	1100	896	204
4	1100	904	196
5	1100	904	196
6	1100	904	196
7	1100	904	196
8	1100	912	188
9	1100	912	188
10	1100	912	188
11	1100	904	196
12	1100	904	196
13	1100	888	212
14	1100	904	196

15	1100	904	196
16	1100	888	212
17	1100	888	212
18	1100	904	196
19	1100	912	188
20	1100	904	196

Tabel 4.5.
Data Putaran Motor Tanpa Beban Tanpa PID
Dengan Set Point 1000 Rpm

Data	Set Point	Terbaca	Error
1	1000	1392	392
2	1000	440	560
3	1000	768	232
4	1000	808	192
5	1000	808	192
6	1000	816	184
7	1000	808	192
8	1000	808	192
9	1000	776	224
10	1000	776	224
11	1000	752	248
12	1000	752	248
13	1000	760	240
14	1000	784	216
15	1000	784	216
16	1000	792	208
17	1000	784	216
18	1000	784	216
19	1000	784	216
20	1000	784	216

Tabel 4.6.
Data Putaran Motor Tanpa Beban Dengan PID
Dengan Set Point 1300 Rpm

Data	Set Point	Terbaca	Error
1	1300	0	1300
2	1300	0	1300
3	1300	0	1300
4	1300	400	9000
5	1300	400	900
6	1300	400	900

7	1300	400	900
8	1300	1024	276
9	1300	1024	276
10	1300	1024	276
11	1300	1024	276
12	1300	1168	132
13	1300	1168	132
14	1300	1168	132
15	1300	1168	132
16	1300	1336	36
17	1300	1336	36
18	1300	1336	36
19	1300	1336	36
20	1300	1336	36

Tabel 4.7.
Data Putaran Motor Tanpa Beban Dengan PID
Dengan Set Point 1100 Rpm

Data	Set Point	Terbaca	Error
1	1100	0	1100
2	1100	0	1100
3	1100	1336	236
4	1100	1336	236
5	1100	1336	236
6	1100	1336	236
7	1100	472	628
8	1100	472	628
9	1100	472	628
10	1100	472	628
11	1100	968	132
12	1100	968	132
13	1100	968	132
14	1100	968	132
15	1100	1024	76
16	1100	1024	76
17	1100	1024	76
18	1100	1024	76
19	1100	1112	12
20	1100	1112	12

Tabel 4.8.
Data Putaran Motor Tanpa Beban Dengan PID
Dengan Set Point 1000 Rpm

Data	Set Point	Terbaca	Error
1	1000	0	
2	1000	0	1000
3	1000	1096	96
4	1000	1096	96
5	1000	1096	96
6	1000	1096	96
7	1000	464	536
8	1000	464	536
9	1000	464	536
10	1000	464	536
11	1000	880	120
12	1000	880	120
13	1000	880	120
14	1000	880	120
15	1000	992	8
16	1000	992	8
17	1000	992	8
18	1000	992	8
19	1000	1000	0
20	1000	1000	0

Tabel 4.9.
Data Putaran Motor Dengan Beban Tanpa PID
Dengan Set Point 1100 Rpm

Data	Set Point	Terbaca	Error
1	1100	64	1036
2	1100	1864	764
3	1100	1084	16
4	1100	880	220
5	1100	912	188
6	1100	904	196
7	1100	904	196
8	1100	904	196
9	1100	904	196
10	1100	904	196
11	1100	928	172
12	1100	912	188
13	1100	904	196
14	1100	888	212

15	1100	856	244
16	1100	848	252
17	1100	832	268
18	1100	840	260
19	1100	456	456
20	1100	856	244

Tabel 4.10.
Data Putaran Motor Dengan Beban Tanpa PID
Dengan Set Point 900 Rpm

Data	Set Point	Terbaca	Error
1	900	1344	444
2	900	648	252
3	900	1008	108
4	900	784	116
5	900	808	92
6	900	800	100
7	900	808	92
8	900	764	136
9	900	712	188
10	900	728	172
11	900	712	188
12	900	720	180
13	900	712	172
14	900	720	180
15	900	712	188
16	900	728	172
17	900	712	188
18	900	712	188
19	900	712	188
20	900	720	180

Tabel 4.11.
Data Putaran Motor Dengan Beban Dengan PID
Dengan Set Point 1100 Rpm

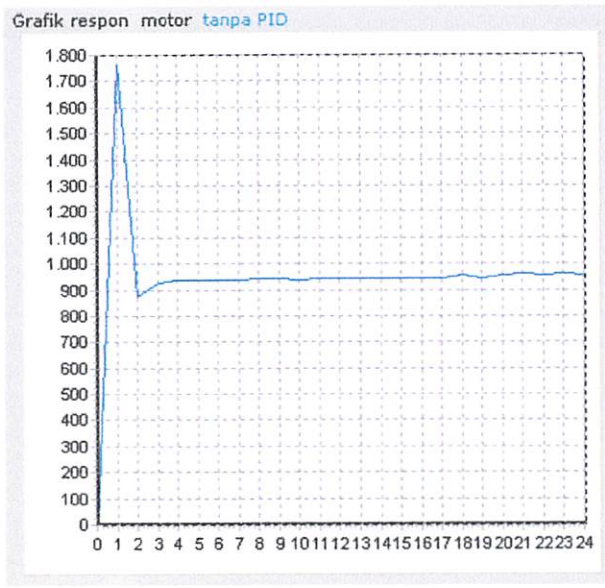
Data	Set Point	Terbaca	Error
1	1100	0	1100
2	1100	968	132
3	1100	744	356
4	1100	872	228
5	1100	992	108
6	1100	1128	28
7	1100	1224	124

8	1100	1200	100
9	1100	1208	108
10	1100	1232	132
11	1100	1216	116
12	1100	1184	84
13	1100	1120	20
14	1100	1104	4
15	1100	1144	44
16	1100	1064	36
17	1100	1000	100
18	1100	1008	92
19	1100	1112	12
20	1100	1120	20

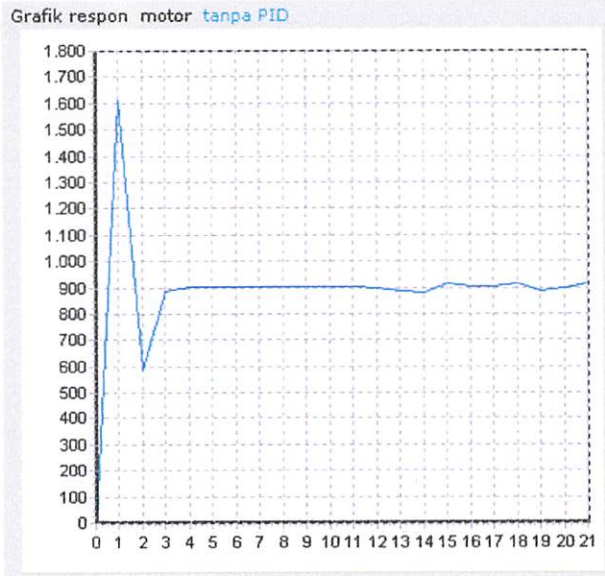
Tabel 4.12.
Data Putaran Motor Dengan Beban Dengan PID
Dengan Set Point 900 Rpm

Data	Set Point	Terbaca	Error
1	900	0	900
2	900	448	452
3	900	704	196
4	900	912	12
5	900	920	20
6	900	984	84
7	900	848	52
8	900	432	468
9	900	808	92
10	900	792	108
11	900	832	68
12	900	944	44
13	900	928	28
14	900	904	4
15	900	928	28
16	900	936	36
17	900	904	4
18	900	936	36
19	900	912	12
20	900	936	36

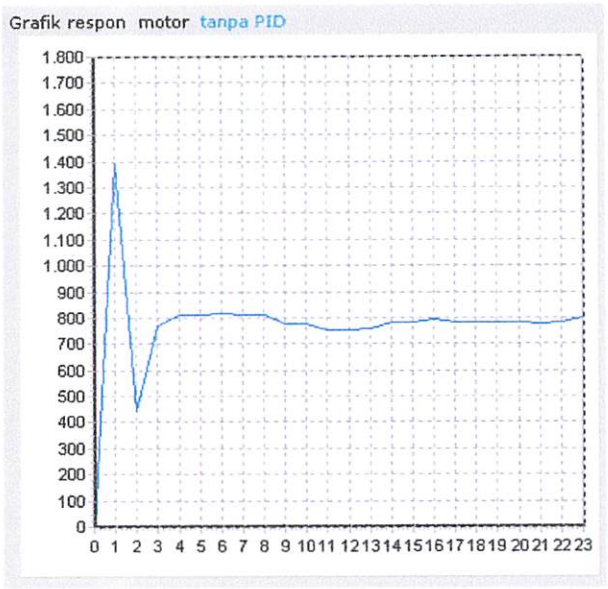
Grafik 4.1.
Respon Motor Tanpa Beban Tanpa PID
Dengan Set Point 1300 Rpm



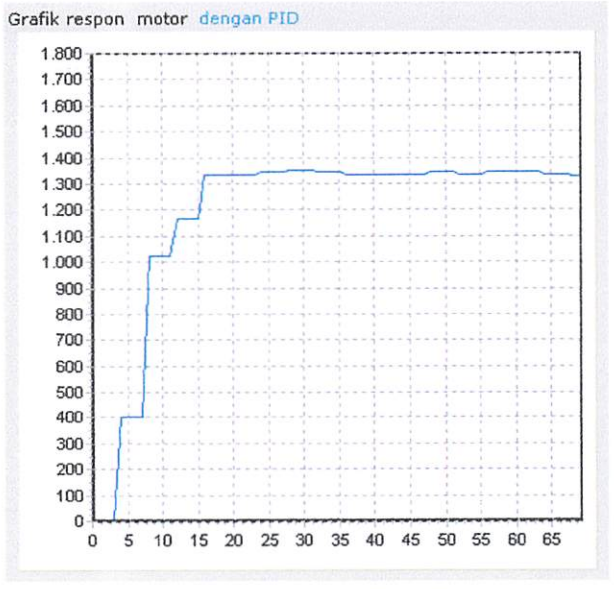
Grafik 4.2.
Respon Motor Tanpa Beban Tanpa PID
Dengan Set Point 1100 Rpm



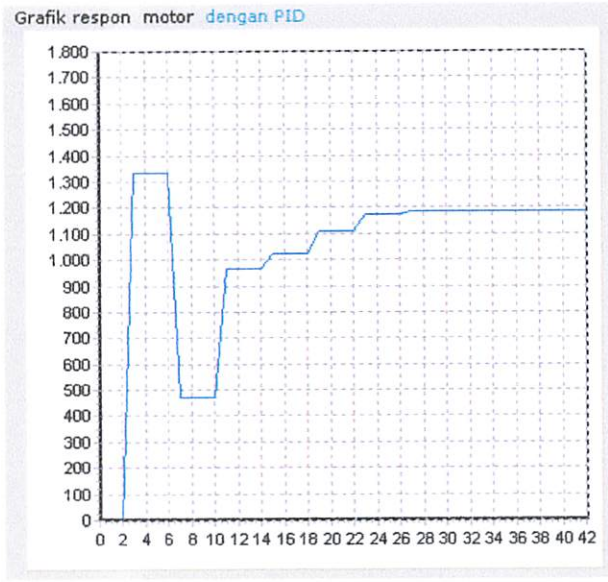
Grafik 4.3.
Respon Motor Tanpa Beban Tanpa PID
Dengan Set Point 1000 Rpm



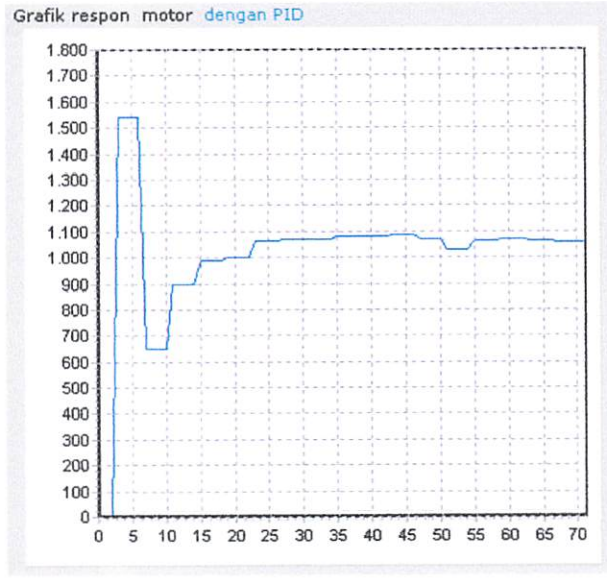
Grafik 4.4.
Respon Motor Tanpa Beban Dengan PID
Dengan Set Point 1300 Rpm



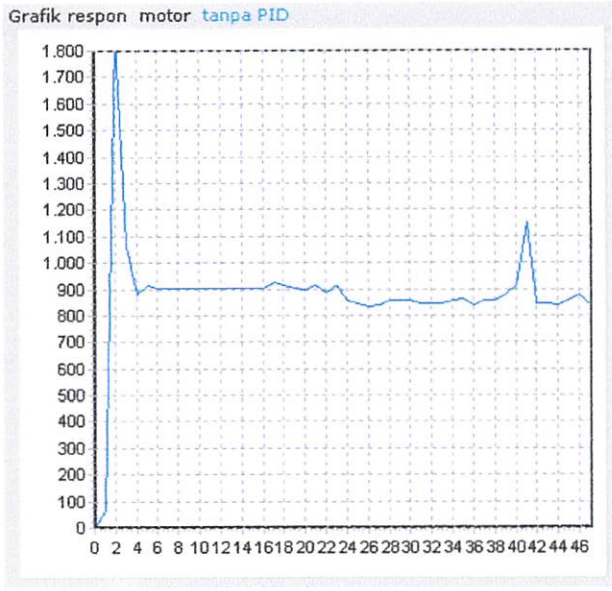
Grafik 4.5.
Respon Motor Tanpa Beban Dengan PID
Dengan Set Point 1100 Rpm



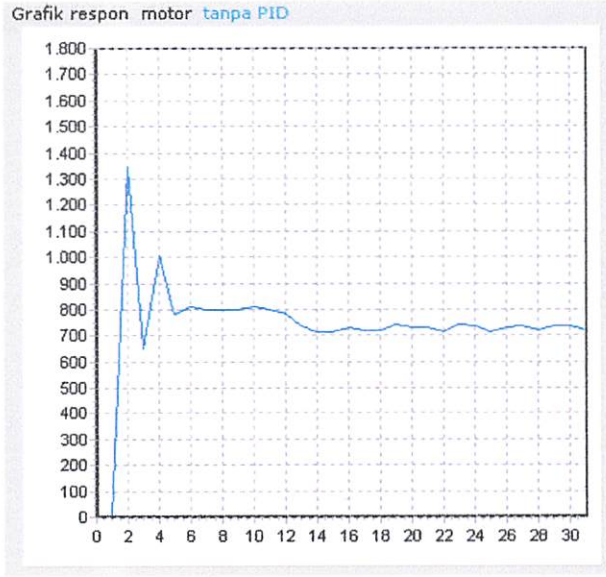
Grafik 4.6.
Respon Motor Tanpa Beban Dengan PID
Dengan Set Point 1000 Rpm



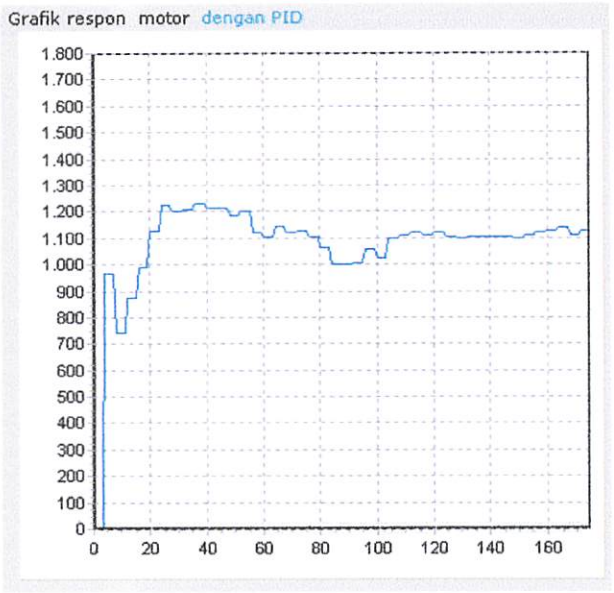
Grafik 4.7.
Respon Motor Dengan Beban Tanpa PID
Dengan Set Point 1100 Rpm



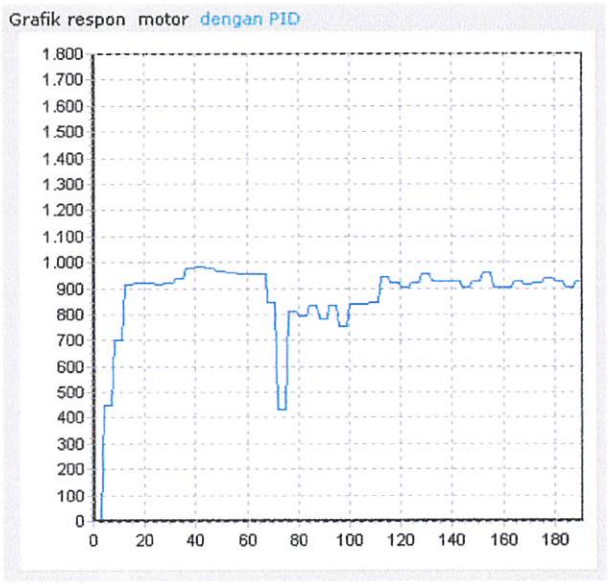
Grafik 4.8.
Respon Motor Dengan Beban Tanpa PID
Dengan Set Point 900 Rpm



Grafik 4.9.
Respon Motor Dengan Beban Dengan PID
Dengan Set Point 1100 Rpm



Grafik 4.10.
Respon Motor Dengan Beban Dengan PID
Dengan Set Point 900 Rpm



- Untuk pengujian tanpa beban
  - Dengan set point 1300 Rpm pada saat motor dijalankan tanpa PID terjadi over shoot hingga 1768 Rpm dan dengan diterapkannya kontrol PID dapat mengurangi over shoot dan mengembalikan kecepatan motor mendekati set point dalam waktu 16 detik dengan kecepatan yang terbaca 1336 Rpm.
  - Dengan set point 1100 Rpm pada saat motor dijalankan tanpa PID terjadi over shoot hingga 1608 Rpm dan dengan diterapkannya kontrol PID dapat mengurangi over shoot dan mengembalikan kecepatan motor mendekati set point dalam waktu 15 detik dengan kecepatan yang terbaca 1024 Rpm.
  - Dengan set point 1000 Rpm pada saat motor dijalankan tanpa PID terjadi over shoot hingga 1392 Rpm dengan diterapkannya kontrol PID dapat mengurangi over shoot dan mengembalikan kecepatan motor mendekati set point dalam waktu 16 detik dengan kecepatan yang terbaca 992 Rpm.

- Untuk pengujian dengan beban
  - Dengan set point 1100 Rpm pada saat motor dijalankan tanpa PID terjadi over shoot hingga 1864 Rpm dan mengalami penurunan kecepatan hingga 832 Rpm dan dengan diterapkannya kontrol PID dapat mengurangi over shoot dan mengembalikan kecepatan motor mendekati set point dalam waktu 19 detik dengan kecepatan yang terbaca 1112 Rpm.
  - Dengan set point 900 Rpm pada saat motor dijalankan tanpa PID terjadi over shoot hingga 1344 Rpm dan mengalami penurunan kecepatan hingga 712 Rpm dengan diterapkannya kontrol PID dapat mengurangi over shoot dan mengembalikan kecepatan motor mendekati set point dalam waktu 12 detik dengan kecepatan yang terbaca 944 Rpm.

BAB V

PENUTUP

5.1. Kesimpulan

Setelah dilakukannya perancangan dan pengujian alat, maka dapat ditarik kesimpulan sebagai berikut :

a. Untuk merancang dan merealisasi sistem pengatur putaran Motor DC tersebut dengan pendekatan PID kontroller yang berbasis Mikrokontroler ATmega8535, maka dapat menggunakan rotary encoder sebagai pembaca data aktual sekaligus feedback bagi sistem, mikrokontroller sebagai pemroses algoritma PID dan PC sebagai pemberi data setting point dan penampil hasil..
b. Dari hasil pengujian bahwa dengan diterapkannya kontrol PID maka, dapat memperbaiki respon system pada putaran motor DC. Serta dalam kondisi pembebanan, penggunaan kontrol PID dapat mengembalikan kecepatan motor mendekati set point.

5.2. Saran

Dengan melihat hasil yang telah dicapai dalam perancangan dan pembuatan alat, maka untuk pengembangan alat selanjutnya menggunakan sistem pengaturan kecepatan secara wireless (kontrol jarak jauh), sehingga pengontrol tidak harus berada ditempat.

DAFTAR PUSTAKA

[1] Alexander Mangkulo, Hengky. Membuat Aplikasi Database dengan Delphi 8.0. PT Gramedia : Jakarta, 2005.

[2] Atmel. "8-bit microcontroller with 8Kbytes Flash Atmega8535". http://www.atmel.com didownload tanggal 7 Juli 2008.

[4] Soebhakti, Hendrawan. *Basic AVR Microcontroller Tutorial*. Politeknik Batam : Batam, 2007.

[5] Zuhal. Dasar Tenaga Listrik. ITB : Bandung, 1997.

[6] Gunterus Frans, Falsafah Dasar Sistem Pengendalian Proses, Jakarta: PT. Elex Media Komputindo, 1997.

[7] Ogata Katsuiko, Teknik Kontrol Automatik, Jakarta: Erlangga, 1991.

[8] http://www.data sheet.com didownload tanggal 7 Juli 2008.

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa : FIPIT PIBRIANTO
NIM : 02.12.024
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Energi Listrik
Judul Skripsi : PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Kamis
Tanggal : 25 September 2008
Dengan Nilai : 76,95 (B+)

Panitia Ujian Skripsi

Ketua

(Ir. Mochtar Asroni, MSME)
NIP.Y. 1018100036

Sekretaris

(Ir. F. Yudi Limpraptono, MT)
NIP.Y. 1039500274

Anggota Penguji

Penguji Pertama

(Ir. Abdul Hamid, MT)
NIP.Y. 1018800188

Penguji Kedua

(Irrine Budi S, ST, MT)
NIP. 132314400

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Kamis
Tanggal : 25 September 2008

Telah dilakukan perbaikan skripsi oleh :

1. Nama : FIPIT PIBRIANTO
2. NIM : 02.12.024
3. Jurusan : Teknik Elektro S-1
4. Konsentrasi : Teknik Energi Listrik
5. Judul Skripsi : PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

Perbaikan meliputi :

No	Materi Perbaikan	Ket.
1.	Nilai error disesuaikan dengan rumus yang umum	
2.	Apa yang dimaksud waktu mati ?	
3.	Pengujian yang dilakukan dan kesimpulan tidak sesuai dengan penjelasan	

Anggota Penguji

(Ir. M. Abdul Hamid, MT)
NIP.Y. 1018800188

Dosen Pembimbing I

(Ir. Widodo Pudji M, MT)
NIP.Y. 1028700171

Dosen Pembimbing II

(Ir. Eko Nurcahyo)
NIP.Y. 1028700172

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Kamis
Tanggal : 25 September 2008

Telah dilakukan perbaikan skripsi oleh :

1. Nama : FIPIT PIBRIANTO
2. NIM : 02.12.024
3. Jurusan : Teknik Elektro S-1
4. Konsentrasi : Teknik Energi Listrik
5. Judul Skripsi : PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

Perbaikan meliputi :

No	Materi Perbaikan	Ket.
1.	Penulisan tabel diatas tabelnya	
2.	Error perhitungan seharusnya dalam %	
3.	Koordinat membingungkan	

Anggota Penguji

(Irrine Budi S, ST, MT)
NIP. 132314400

Dosen Pembimbing I

(Ir. Widodo Pudji M, MT)
NIP.Y. 1028700171

Dosen Pembimbing II

(Ir. Eko Nurcahyo)
NIP.Y. 1028700172

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Lampiran : 1 (satu) berkas
Pembimbing Skripsi

Kepada : Yth. Bapak Ir. Eko Nurcahyo
Dosen Jurusan Elektro
Institut Teknologi Nasional
Malang

Yang bertandatangan di bawah ini :

Nama	: Fipit Pibrianto
Nim	: 02.12.024
Jurursan	: Teknik Elektro S-1
Konsentrasi	: Teknik Energi Listrik

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing ~~Utama~~ / Pendamping *), untuk penyusunan Skripsi dengan judul (proposal terlampir):

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLER ATmega8535

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh ujian Akhir Sarjana Teknik.
Demikian permohonan kami, atas kesediaan Bapak/Ibu kami ucapkan terimakasih.

Malang, Mei 2008

Hormat kami

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F.Yudi Limpraptono, MT
NIP. Y. 103 950 0274

Fipit Pibrianto
Nim : 02.12.024

*) **Coret yang tidak perlu**

Form S-3a

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Lampiran : 1 (satu) berkas
Pembimbing Skripsi

Kepada : Yth. Bapak Ir. Widodo Pudji Mulyanto, MT
Dosen Jurusan Elektro
Institut Teknologi Nasional
Malang

Yang bertandatangan di bawah ini :

Nama	: Fipit Pibrianto
Nim	: 02.12.024
Jurursan	: Teknik Elektro S-1
Konsentrasi	: Teknik Energi Listrik

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing Utama / ~~Pendamping~~ *), untuk penyusunan Skripsi dengan judul (proposal terlampir):

PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLER ATmega8535

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh ujian Akhir Sarjana Teknik.
Demikian permohonan kami, atas kesediaan Bapak/Ibu kami ucapkan terimakasih.

Malang, Mei 2008

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F.Yudi Limpraptono, MT
NIP. Y. 103 950 0274

Hormat kami,

Fipit Pibrianto
Nim : 02.12.024

*) **Coret yang tidak perlu**

Form S-3a

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

PERNYATAAN KESEDIAAN SEBAGAI DOSEN PEMBIMBING SKRIPSI

ai dengan Permohonan Mahasiswa :

a : Fipit Pibrianto

: 02.12.024

ıan : Teknik Elektro S-1

ıentrasi : Teknik Energi Listrik

ıan ini menyatakan **bersedia / tidak bersedia** *) Membimbing Skripsi dari ısiswa tersebut dengan judul:

ıANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLER ATmega8535

ıikian pernyataan ini kami buat untuk dapat dipergunakan sebagaimana mestinya.

Malang, Juni 2008
Kami yang membuat pernyataan,

Ir. Widodo Pudji Mulyanto, MT
NIP. Y. 102 870 071

ıtan:
ılah disetujui agar formulir ini
ırahkan mahasiswa/i yang bersangkutan
ada Jurusan untuk diproses lebih lanjut
oret yang tidak perlu

Form S-3 b

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

PERNYATAAN KESEDIAAN SEBAGAI DOSEN PEMBIMBING SKRIPSI

uai dengan Permohonan Mahasiswa :

ma : Fipit Pibrianto

n : 02.12.024

usan : Teknik Elektro S-1

nsentrasi : Teknik Energi Listrik

ngan ini menyatakan **bersedia / tidak bersedia** *) Membimbing Skripsi dari hasiswa tersebut dengan judul:

RANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLER ATmega8535

mikian pernyataan ini kami buat untuk dapat dipergunakan sebagaimana mestinya.

Malang, Juni 2008
Kami yang membuat pernyataan,

Ir. Eko Nurcahyo
NIP. Y. 102 870 072

tatan:
telah disetujui agar formulir ini
serahkan mahasiswa/i yang bersangkutan
pada Jurusan untuk diproses lebih lanjut
Coret yang tidak perlu

Form S-3 b

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

SERO) MALANG
GA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 28 Juni 2008

or : ITN-109/I.TA/2/08
iran : -
al : BIMBINGAN SKRIPSI

da : Yth. Sdr. **Ir. WIDODO PUDJI M, MT**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: FIPIT PIBRIANTO
Nim	: 0212024
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Energi Listrik

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

23 Juni 2008 s/d 23 Nopember 2008

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih

Ketua Jurusan
Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
Nip. Y. 1039500274

Tembusan Kepada Yth :
1. Mahasiswa Yang Bersangkutan
2. Arsip

Form. S 4a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

ERSERO) MALANG
NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 28 Juni 2008

nor : ITN-110/I.TA/2/08
npiran : -
hal : BIMBINGAN SKRIPSI

ada : Ytn. Sdr. **Ir. EKO NURCAHYO**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: FIPIT PIBRIANTO
Nim	: 0212024
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Energi Listrik

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

23 Juni 2008 s/d 23 Nopember 2008

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih

Ketua Jurusan
Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
Nip. Y. 1039500274

Tembusan Kepada Yth :
1. Mahasiswa Yang Bersangkutan
2. Arsip

Form. S 4a

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : FIPIT PIBRIANTO
Nim : 02.12.024
Masa Bimbingan : 23 Juni 2008 s/d 23 Desember 2008
Judul Skripsi : **PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTOLLER ATmega8535**

No.	Tanggal	Uraian	Paraf Pembimbing
	26-06-2008	Bab I, Bab II (revisi)	
	28-06-2008	Bab II (Acc), Bab III (revisi)	
	10-07-2008	Bab III (Acc), Bab IV (revisi)	
	28-08-2008	Bab IV (revisi), Bab V (revisi)	
	16-09-208	Bab IV (revisi), Bab V (Acc)	
	20-9-2008	Bab I, II, III, IV, V (Acc)	

Malang,
Dosen Pembimbing,

Ir. Widodo Budji Muljanto, MT
Nip. Y. 102 870 071

Form.S-4b

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Fipit Pibrianto
Nim : 02.12.024
Masa Bimbingan : 23 Juni 2008 s/d 23 Desember 2008
Judul Skripsi : PERANCANGAN DAN PEMBIAUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535

D	Tanggal	Uraian	Paraf Pembimbing
	26 – 06-2008	Revisi Bab I dan Bab II	
	03 – 07-2008	ACC Bab I dan Bab II	
	20 – 07-2008	Revisi Bab III	
	03 – 08-2008	Revisi Bab III dan Bab IV	
	25 – 08-2008	ACC Bab III dan Revisi Bab IV	
	30 – 08-2008	Revisi Bab IV dan Bab V	
	05– 09-2008	Revisi Bab IV dan Bab V	
	17– 09-2008	ACC Bab IV dan Bab V	
	24-09-2008	ACC Ujian Skripsi	

Malang,
Dosen Pembimbing

Ir. Eko Nurcahyo
Nip. Y 102 870 072

Form.S-4b

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika*)

1.	Nama Mahasiswa: FIPIT PIBRIANTO			Nim: 02.12.024
2.	Keterangan	Tanggal	Waktu	Tempat
	Pelaksanaan	23-06-2008		Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**)			
	a. Sistem Tenaga Elektrik b Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran (d) Sistem Kendali Industri		e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya	
4.	Judul Proposal yang diseminarkan Mahasiswa	PERANCANGAN DAN PEMBUATAN ALAT PENGATUR PUTARAN MOTOR DC MENGGUNAKAN METODE ALGORITMA PID KONTROLLER BERBASIS MIKROKONTROLLER ATmega8535		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	.....		
6.	Catatan:			
7.	Catatan:			
	Persetujuan Judul Skripsi			
	Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II		
	Mengetahui, Ketua Jurusan. Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274	Disetujui, Calon Dosen Pembimbing ybs Pembimbing I Ir. Widodo Budi Mulyanto, MT NIP.Y. 102 870 071	Pembimbing II Ir. Eko Nurcahyo NIP.Y. 1028700?2	

Perhatian:

1. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-3c

```
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#include <string.h>
#define t0                  1//1=0,99 ms baca data serial dan tentukan penanda
#define t1                  250//250=0,248s//25=25ms
#define t2              50//proses penanda
#define t3                250//kirim data serial
#define pin_input           PIND.2
#define RXC                 UCSRA.7

/*
karakteristik pembacaan motor
waktu sampling untuk pembacaan encoder dipilih 25 ms untuk mempercepat
update
pulsa maksimal untuk OCR=255, t=25ms, adalah 62 sehingga setting point adalah
max-2=60
nilai terendah OCR yang masih dapat direspon adalah 20 yakni 2 pulsa/25ms
*/

//konstanta yang digunakan
unsigned char time0,time1, time2, time3;
unsigned char jml_pulsa;

char temp_rpm[60];

char data,penanda;
unsigned int rpm, nilai_proses;

unsigned int set_point,set_point1;
int i,j;//variable untuk tes

interrupt [EXT_INT0] void ext_int0_isr(void)
{
        if(pin_input==1)
        jml_pulsa++;
}

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
 TCNT1L=0x00;
 ICR1H=0x00;

 if(time0>0)
 --time0;
```

```
if(time1>0)
--time1;

if(time2>0)
--time2;

if(time3>0)
--time3;
}

unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & (1<<RXC)) )
;
/* Get and return received data from buffer */
return UDR;
}

void kirim_serial(void)
{
}

struct parameter_PID
{
 unsigned int KP;
 float KI;
 unsigned int KD;
 unsigned int error;
 unsigned int max_error;
 unsigned int error_terakhir;
 unsigned int akumulasi_error;
 unsigned int max_akumulasi_error;
 char out_pwm_max;
 float out_pwm;
}pid;

char proses_pid(void)
{
   unsigned int bagian_p;
   float bagian_i;
   unsigned int bagian_d;
   pid.KP=3.2;
   pid.KI=0.0001;
   pid.KD=5;
   //menghitung bagian proportional
```

```
pid.error=set_point-nilai_proses;//berdasarkan pembacaan pulsa
pid.max_error=set_point;
   if (pid.max_error>=set_point)
   {
   bagian_p=set_point;
   }
   else if(pid.max_error<=0)
   {
   bagian_p=0;
   }
   else
   {
   bagian_p=pid.KP*pid.error;
   }
//menghitung bagian integral
pid.akumulasi_error=pid.akumulasi_error+pid.error;

bagian_i=pid.KI*pid.akumulasi_error;

//menghitung bagian deferensial
bagian_d=pid.KD*(pid.error-pid.error_terakhir);
pid.error_terakhir=pid.error;

//menghitung keluaran proses PID
pid.out_pwm+=bagian_p+bagian_i+bagian_d;

   //mengatasi overflow
   if(pid.out_pwm>=255)
   {
        OCR0=pid.out_pwm_max;
   }
   else if(pid.out_pwm<=0)
   {
        OCR0=0;
   }
   else
   {
        OCR0=(unsigned char)pid.out_pwm;
   }
     putchar('$');

sprintf(temp_rpm,"a%ib%ic%id%ie",i,set_point1,nilai_proses*8,pid.error*8);
     puts(temp_rpm);
     putchar('*');
```

```
        i++;
        return OCR0;
}


void main(void)
{

PORTA=0x00;
DDRA=0x00;

PORTB=0x00;
DDRB=0x08;

PORTC=0x00;
DDRC=0x00;

PORTD=0x00;
DDRD=0x00;

TCCR0=0x6B;
TCNT0=0x00;
OCR0=0;

TCCR1A=0x00;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x1F;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

GICR|=0x40;
MCUCR=0x03;
MCUCSR=0x00;
GIFR=0x40;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=0x10;

UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;

ACSR=0x80;
SFIOR=0x00;

time0=t0;
time1=t1;
time2=t2;
time3=t3;

#asm("sei")

//forever
for (;;)
    {
        //baca serial
        if(time0==0)
        {
         data=UDR;
         switch(data)
         {
              case 'a'     : penanda=1;break;
              case 'b'     : penanda=2;break;
              case 'c'     : penanda=3;break;
              case 'd'     : penanda=4;break;
              case 'e'     : penanda=5;break;
              case 'f'     : penanda=6;break;
              case 'g'     : penanda=7;break;
              case 'h'     : penanda=8;break;
              case 'i'     : penanda=9;break;
              case 'j'     : penanda=10;break;
              case 'k'     : penanda=11;break;
              case 'l'     : penanda=12;break;
              case 'm'     : penanda=13;break;
              case 'n'     : penanda=14;break;
              case 'o'     : penanda=15;break;
              case 'p'     : penanda=16;break;
              case 'q'     : penanda=17;break;
              case 'r'     : penanda=18;break;
              case 's'     : penanda=19;break;
```

```
            case 't'     : penanda=20;break;
            case 'u'     : penanda=21;break;
            case 'v'     : penanda=22;break;
          }

          time0=t0;
          }

          //baca rpm
          if(time1==0)
          {
          #asm("cli")
          rpm=(unsigned int)jml_pulsa;
          nilai_proses=rpm;
          jml_pulsa=0;
          time1=t1;
          #asm("sei")
          }

          //proses PID
          if(time2==0)
          {
                    switch(penanda)
          {
            case 1     : set_point=1300;OCR0=255;break;//128
            case 2     : set_point=1100;OCR0=240;break;//122
            case 3     : set_point=1000;OCR0=225;break;//108
            case 4     : set_point=900;OCR0=210;break;//104
            case 5     : set_point=800;OCR0=195;break;//98
            case 6     : set_point=700;OCR0=170;break;//86
            case 7     : set_point=600;OCR0=155;break;//84
            case 8     : set_point=500;OCR0=140;break;//70
            case 9     : set_point=400;OCR0=125;break;//58
            case 10    : set_point=300;OCR0=110;break;//54
            case 11    :
set_point1=1300;set_point=128;pid.out_pwm_max=255;OCR0=proses_pid();brea
k;
            case 12    :
set_point1=1100;set_point=122;pid.out_pwm_max=240;OCR0=proses_pid();brea
k;
            case 13    :
set_point1=1000;set_point=108;pid.out_pwm_max=225;OCR0=proses_pid();brea
k;
            case 14    :
set_point1=900;set_point=104;pid.out_pwm_max=210;OCR0=proses_pid();break
;
```

```
                case 15     :
set_point1=800;set_point=98;pid.out_pwm_max=195;OCR0=proscs_pid();break;
                case 16     :
set_point1=700;set_point=86;pid.out_pwm_max=170;OCR0=proses_pid();break;
                case 17     :
set_point1=600;set_point=84;pid.out_pwm_max=155;OCR0=proses_pid();break;
                case 18     :
set_point1=500;set_point=70;pid.out_pwm_max=140;OCR0=proses_pid();break;
                case 19     :
set_point1=400;set_point=58;pid.out_pwm_max=125;OCR0=proses_pid();break;
                case 20     :
set_point1=300;set_point=54;pid.out_pwm_max=110;OCR0=proses_pid();break;
                case 21     :
OCR0=0;OCR0=0;OCR0=0;set_point=0;j=0;i=0;penanda=0;break;

             }

            time2=t2;
            }

            //kirim data ke serial
            if(time3==0)
            {
            if((penanda<11)&&(penanda>0))
             {
              putchar('$');

sprintf(temp_rpm,"a%ib%ic%id%ic",j,set_point,nilai_proses*8,(unsigned
int)(set_point-nilai_proses));
              puts(temp_rpm);
              putchar('*');
              j++;
              time3=t3;
             }
            }
    };
}
```

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, CPort, StdCtrls, ExtCtrls, TeEngine, Series, TeeProcs, Chart,
Grids,Comobj;

type
TForm1 = class(TForm)
Button1: TButton;
ComPort1: TComPort;
Button3: TButton;
Button5: TButton;
GroupBox1: TGroupBox;
Chart1: TChart;
Series1: TLineSeries;
Label4: TLabel;
GroupBox3: TGroupBox;
Button2: TButton;
GroupBox4: TGroupBox;
GroupBox5: TGroupBox;
StringGrid1: TStringGrid;
Label1: TLabel;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
GroupBox2: TGroupBox;
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
RadioButton5: TRadioButton;
RadioButton6: TRadioButton;
RadioButton7: TRadioButton;
RadioButton8: TRadioButton;
RadioButton9: TRadioButton;
RadioButton10: TRadioButton;
Label2: TLabel;
Label3: TLabel;
Label5: TLabel;
GroupBox6: TGroupBox;
Memo1: TMemo;
GroupBox7: TGroupBox;
Button4: TButton;
Button7: TButton;
Label6: TLabel;
Series2: TLineSeries;
```

```
Button6: TButton;
Label7: TLabel;
procedure Button3Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure ComPort1AfterClose(Sender: TObject);
procedure ComPort1AfterOpen(Sender: TObject);
procedure ComPort1RxChar(Sender: TObject; Count: Integer);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;
i:integer;

implementation

{$R *.dfm}

procedure TForm1.Button3Click(Sender: TObject);
begin
Comport1.ShowSetupDialog;
end;

procedure TForm1.Button1Click(Sender: TObject);


begin
Label7.Caption:='dengan PID';
Label6.Caption:='dengan PID';
StringGrid1.Visible:=true;
if (Radiobutton1.Checked) then
begin
Comport1.WriteStr('k');
Comport1.WriteStr('k');
end
else if (Radiobutton2.Checked) then
begin
```

```
Comport1.WriteStr('l');
Comport1.WriteStr('l');
end
else if (Radiobutton3.Checked) then
begin
Comport1.WriteStr('m');
Comport1.WriteStr('m');
end
else if (Radiobutton4.Checked) then
begin
Comport1.WriteStr('n');
Comport1.WriteStr('n');
end
else if (Radiobutton5.Checked) then
begin
Comport1.WriteStr('o');
Comport1.WriteStr('o');
end
else if (Radiobutton6.Checked) then
begin
Comport1.WriteStr('p');
Comport1.WriteStr('p');
end
else if (Radiobutton7.Checked) then
begin
Comport1.WriteStr('q');
Comport1.WriteStr('q');
end
else if (Radiobutton8.Checked) then
begin
Comport1.WriteStr('r');
Comport1.WriteStr('r');
end
else if (Radiobutton9.Checked) then
begin
Comport1.WriteStr('s');
Comport1.WriteStr('s');
end
else if (Radiobutton10.Checked) then
begin
Comport1.WriteStr('t');
Comport1.WriteStr('t');
end

end;
```

```
procedure TForm1.ComPort1AfterClose(Sender: TObject);
begin
if label5<>nil then
Label5.Caption:='terputus';
end;

procedure TForm1.ComPort1AfterOpen(Sender: TObject);
begin
Label5.Caption:='terhubung';
end;

procedure TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
var
baca_serial:string;
a,b,c,d,e,f,g:integer;
data_x,data_y:integer;
temp1,temp2,temp3,temp4:string;
//$a1b100c120d*
begin
ComPort1.ReadStr(baca_serial,count);
//temp2:=temp2+baca_serial;
Memo1.Text:=Memo1.Text+baca_serial;

if baca_serial<>' ' then
begin
if copy(Memo1.Text,length(Memo1.Text),1)='*' then
begin
a:=pos('$',Memo1.Text);
b:=pos('*',Memo1.Text);
temp1:=copy(Memo1.Text,a+1,b-a-1);
Memo1.Text:=' ';
c:=pos('a',temp1);
d:=pos('b',temp1);
StringGrid1.Cells[0,i]:=copy(temp1,c+1,d-c-1);
data_x:=strToInt(StringGrid1.Cells[0,i]);
e:=pos('c',temp1);
StringGrid1.Cells[1,i]:=copy(temp1,d+1,e-d-1);

f:=pos('d',temp1);
StringGrid1.Cells[2,i]:=(copy(temp1,e+1,f-e-1));
data_y:=strToInt(StringGrid1.Cells[2,i]);
g:=pos('e',temp1);
StringGrid1.Cells[3,i]:=copy(temp1,f+1,g-f-1);


with Chart1 do
```

```
with Series1 do
Addxy(data_x,data_y,'',clBlue);
i:=i+1;
stringGrid1.RowCount:=i+1;
end
end
end;

procedure TForm1.Button5Click(Sender: TObject);

begin
Label7.Caption:='tanpa PID';
Label6.Caption:='tanpa PID';
//StringGrid1.Visible:=false;
if (Radiobutton1.Checked) then
begin
Comport1.WriteStr('a');
Comport1.WriteStr('a');
end
else if (Radiobutton2.Checked) then
begin
Comport1.WriteStr('b');
Comport1.WriteStr('b');
end
else if (Radiobutton3.Checked) then
begin
Comport1.WriteStr('c');
Comport1.WriteStr('c');
end
else if (Radiobutton4.Checked) then
begin
Comport1.WriteStr('d');
Comport1.WriteStr('d');
end
else if (Radiobutton5.Checked) then
begin
Comport1.WriteStr('e');
Comport1.WriteStr('e');
end
else if (Radiobutton6.Checked) then
begin
Comport1.WriteStr('f');
Comport1.WriteStr('f');
end
else if (Radiobutton7.Checked) then
begin
```

```
Comport1.WriteStr('g');
Comport1.WriteStr('g');
end
else if (Radiobutton8.Checked) then
begin
Comport1.WriteStr('h');
Comport1.WriteStr('h');
end
else if (Radiobutton9.Checked) then
begin
Comport1.WriteStr('i');
Comport1.WriteStr('i');
end
else if (Radiobutton10.Checked) then
begin
Comport1.WriteStr('j');
Comport1.WriteStr('j');
end

end;

procedure TForm1.Button6Click(Sender: TObject);
var
Trans_Excel:variant;
new_name:variant;
cnt_trans:integer;
begin
Trans_Excel:=CreateOLEObject('Excel.Application');
Trans_Excel.Visible:=True;
Trans_Excel.Workbooks[1].Worksheets[1].name:='backup_data';
New_Name:=Trans_Excel.workbooks[1].Worksheet['backup_data'];
New_Name.Cells[0,1]:='Data';
New_Name.Cells[1,2]:='Set point';
New_Name.Cells[2,3]:='Proses Vlue';
for cnt_trans:=i to (i+1)do
begin
New_name.cells[cnt_trans,0]:=StringGrid1.Cells[0,cnt_trans-1];
New_name.cells[cnt_trans,1]:=StringGrid1.Cells[1,cnt_trans-1];
New_name.cells[cnt_trans,2]:=StringGrid1.Cells[2,cnt_trans-1];
New_name.cells[cnt_trans,3]:=StringGrid1.Cells[3,cnt_trans-1];
end

end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
i:=1;
button6.Enabled:=false;
StringGrid1.Cells[0,0]:='Data';
stringgrid1.Cells[1,0]:='Set Point';
stringgrid1.Cells[2,0]:='Terbaca';
stringgrid1.Cells[3,0]:='Error';
if ComPort1.Connected then
ComPort1.Close
else
ComPort1.Open;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin

Comport1.WriteStr('u');
Comport1.WriteStr('u');
Comport1.WriteStr('u');

end;

procedure TForm1.Button4Click(Sender: TObject);
begin
Series1.clear;
repeat

stringGrid1.Cells[0,i]:='';
stringGrid1.Cells[1,i]:='';
stringGrid1.Cells[2,i]:='';
stringGrid1.Cells[3,i]:='';
i:=i-1;

until (i=0);
i:=1;

end;
procedure TForm1.Button7Click(Sender: TObject);
begin
Chart1.Print ;
end;

end.
```

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, CPort, StdCtrls, ExtCtrls, TeEngine, Series, TeeProcs, Chart,
Grids,Comobj;

type
TForm1 = class(TForm)
Button1: TButton;
ComPort1: TComPort;
Button3: TButton;
Button5: TButton;
GroupBox1: TGroupBox;
Chart1: TChart;
Series1: TLineSeries;
Label4: TLabel;
GroupBox3: TGroupBox;
Button2: TButton;
GroupBox4: TGroupBox;
GroupBox5: TGroupBox;
StringGrid1: TStringGrid;
Label1: TLabel;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
GroupBox2: TGroupBox;
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
RadioButton5: TRadioButton;
RadioButton6: TRadioButton;
RadioButton7: TRadioButton;
RadioButton8: TRadioButton;
RadioButton9: TRadioButton;
RadioButton10: TRadioButton;
Label2: TLabel;
Label3: TLabel;
Label5: TLabel;
GroupBox6: TGroupBox;
Memo1: TMemo;
GroupBox7: TGroupBox;
Button4: TButton;
Button7: TButton;
Label6: TLabel;
Series2: TLineSeries;
```

```
Button6: TButton;
Label7: TLabel;
procedure Button3Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure ComPort1AfterClose(Sender: TObject);
procedure ComPort1AfterOpen(Sender: TObject);
procedure ComPort1RxChar(Sender: TObject; Count: Integer);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;
i:integer;

implementation

{$R *.dfm}

procedure TForm1.Button3Click(Sender: TObject);
begin
Comport1.ShowSetupDialog;
end;

procedure TForm1.Button1Click(Sender: TObject);


begin
Label7.Caption:='dengan PID';
Label6.Caption:='dengan PID';
StringGrid1.Visible:=true;
if (Radiobutton1.Checked) then
begin
Comport1.WriteStr('k');
Comport1.WriteStr('k');
end
else if (Radiobutton2.Checked) then
begin
```

```
Comport1.WriteStr('l');
Comport1.WriteStr('l');
end
else if (Radiobutton3.Checked) then
begin
Comport1.WriteStr('m');
Comport1.WriteStr('m');
end
else if (Radiobutton4.Checked) then
begin
Comport1.WriteStr('n');
Comport1.WriteStr('n');
end
else if (Radiobutton5.Checked) then
begin
Comport1.WriteStr('o');
Comport1.WriteStr('o');
end
else if (Radiobutton6.Checked) then
begin
Comport1.WriteStr('p');
Comport1.WriteStr('p');
end
else if (Radiobutton7.Checked) then
begin
Comport1.WriteStr('q');
Comport1.WriteStr('q');
end
else if (Radiobutton8.Checked) then
begin
Comport1.WriteStr('r');
Comport1.WriteStr('r');
end
else if (Radiobutton9.Checked) then
begin
Comport1.WriteStr('s');
Comport1.WriteStr('s');
end
else if (Radiobutton10.Checked) then
begin
Comport1.WriteStr('t');
Comport1.WriteStr('t');
end

end;
```

```
procedure TForm1.ComPort1AfterClose(Sender: TObject);
begin
if label5<>nil then
Label5.Caption:='terputus';
end;

procedure TForm1.ComPort1AfterOpen(Sender: TObject);
begin
Label5.Caption:='terhubung';
end;

procedure TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
var
baca_serial:string;
a,b,c,d,e,f,g:integer;
data_x,data_y:integer;
temp1,temp2,temp3,temp4:string;
//$a1b100c120d*
begin
ComPort1.ReadStr(baca_serial,count);
//temp2:=temp2+baca_serial;
Memo1.Text:=Memo1.Text+baca_serial;

if baca_serial<>' ' then
begin
if copy(Memo1.Text,length(Memo1.Text),1)='*' then
begin
a:=pos('$',Memo1.Text);
b:=pos('*',Memo1.Text);
temp1:=copy(Memo1.Text,a+1,b-a-1);
Memo1.Text:=' ';
c:=pos('a',temp1);
d:=pos('b',temp1);
StringGrid1.Cells[0,i]:=copy(temp1,c+1,d-c-1);
data_x:=strToInt(StringGrid1.Cells[0,i]);
e:=pos('c',temp1);
StringGrid1.Cells[1,i]:=copy(temp1,d+1,e-d-1);

f:=pos('d',temp1);
StringGrid1.Cells[2,i]:=(copy(temp1,e+1,f-e-1));
data_y:=strToInt(StringGrid1.Cells[2,i]);
g:=pos('e',temp1);
StringGrid1.Cells[3,i]:=copy(temp1,f+1,g-f-1);


with Chart1 do
```

```
with Series1 do
Addxy(data_x,data_y,'',clBlue);
i:=i+1;
stringGrid1.RowCount:=i+1;
end
end
end;

procedure TForm1.Button5Click(Sender: TObject);

begin
Label7.Caption:='tanpa PID';
Label6.Caption:='tanpa PID';
//StringGrid1.Visible:=false;
if (Radiobutton1.Checked) then
begin
Comport1.WriteStr('a');
Comport1.WriteStr('a');
end
else if (Radiobutton2.Checked) then
begin
Comport1.WriteStr('b');
Comport1.WriteStr('b');
end
else if (Radiobutton3.Checked) then
begin
Comport1.WriteStr('c');
Comport1.WriteStr('c');
end
else if (Radiobutton4.Checked) then
begin
Comport1.WriteStr('d');
Comport1.WriteStr('d');
end
else if (Radiobutton5.Checked) then
begin
Comport1.WriteStr('e');
Comport1.WriteStr('e');
end
else if (Radiobutton6.Checked) then
begin
Comport1.WriteStr('f');
Comport1.WriteStr('f');
end
else if (Radiobutton7.Checked) then
begin
```

```
Comport1.WriteStr('g');
Comport1.WriteStr('g');
end
else if (Radiobutton8.Checked) then
begin
Comport1.WriteStr('h');
Comport1.WriteStr('h');
end
else if (Radiobutton9.Checked) then
begin
Comport1.WriteStr('i');
Comport1.WriteStr('i');
end
else if (Radiobutton10.Checked) then
begin
Comport1.WriteStr('j');
Comport1.WriteStr('j');
end

end;

procedure TForm1.Button6Click(Sender: TObject);
var
Trans_Excel:variant;
new_name:variant;
cnt_trans:integer;
begin
Trans_Excel:=CreateOLEObject('Excel.Application');
Trans_Excel.Visible:=True;
Trans_Excel.Workbooks[1].Worksheets[1].name:='backup_data';
New_Name:=Trans_Excel.workbooks[1].Worksheet['backup_data'];
New_Name.Cells[0,1]:='Data';
New_Name.Cells[1,2]:='Set point';
New_Name.Cells[2,3]:='Proses Vlue';
for cnt_trans:=i to (i+1)do
begin
New_name.cells[cnt_trans,0]:=StringGrid1.Cells[0,cnt_trans-1];
New_name.cells[cnt_trans,1]:=StringGrid1.Cells[1,cnt_trans-1];
New_name.cells[cnt_trans,2]:=StringGrid1.Cells[2,cnt_trans-1];
New_name.cells[cnt_trans,3]:=StringGrid1.Cells[3,cnt_trans-1];
end

end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
i:=1;
button6.Enabled:=false;
StringGrid1.Cells[0,0]:='Data';
stringgrid1.Cells[1,0]:='Set Point';
stringgrid1.Cells[2,0]:='Terbaca';
stringgrid1.Cells[3,0]:='Error';
if ComPort1.Connected then
ComPort1.Close
else
ComPort1.Open;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin

Comport1.WriteStr('u');
Comport1.WriteStr('u');
Comport1.WriteStr('u');

end;

procedure TForm1.Button4Click(Sender: TObject);
begin
Series1.clear;
repeat

stringGrid1.Cells[0,i]:='';
stringGrid1.Cells[1,i]:='';
stringGrid1.Cells[2,i]:='';
stringGrid1.Cells[3,i]:='';
i:=i-1;

until (i=0);
i:=1;

end;
procedure TForm1.Button7Click(Sender: TObject);
begin
Chart1.Print ;
end;

end.
```