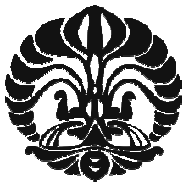


UNIVERSITAS INDONESIA

RANCANG BANGUN PROTOTIPE SISTEM AKTUATOR KENDALI SIRIP MENGGUNAKAN LABVIEW

SKRIPSI

ELLAN S. SIREGAR
0906603291

FAKULTAS TEKNIK
PROGRAM S1 EKSTENSI TEKNIK ELEKTRO
DEPOK
JUNI

UNIVERSITAS INDONESIA

RANCANG BANGUN PROTOTIPE SISTEM AKTUATOR KENDALI SIRIP ROKET BERBASIS LABVIEW

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

ELLAN S. SIREGAR
0906603291

FAKULTAS TEKNIK
PROGRAM S1 EKSTENSI TEKNIK ELEKTRO
DEPOK
JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Tugas akhir ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Ellan S. Siregar**
NPM	**: 0906 603 291**
Tanda Tangan	**:** 
Tanggal	**: 19 Juni 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	:	Ellan S. Siregar
NPM	:	0906 603 291
Program Studi	:	Ekstensi Teknik Elektro
Judul Tugas Akhir	:	Rancang Bangun Prototipe Sistem Aktuator Kendali Sirip Menggunakan LabVIEW

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pemimbing	: Dr. Abdul Halim M.Eng	( )
Penguji	: Dr. Abdul Muis S.T., M.Eng	( )
Penguji	: Dr. Ir. Feri Yusivar M.Eng	( )

Ditetapkan di : Depok
Tanggal : 19 Juni 2012

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur penulis panjatkan kepada ALLAH SWT yang telah memberikan rahmat, hidayah, karunia, serta rezeki kepada penulis sehingga dapat menyelesaikan Tugas Akhir ini dengan baik.

Laporan Tugas Akhir dengan judul "**Rancang Bangun Prototipe Sistem Aktuator Kendali Sirip Menggunakan LabVIEW**" ini disusun sebagai salah satu syarat untuk menyelesaikan masa studi dan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.Selain itu, saya juga ingin mengucapkan banyak terima kasih pada beberapa pihak yang telah membantu penulis baik selama masa studi maupun dalam penyusunan Tugas Akhir, antara lain:

1. ALLAH SWT yang telah melimpahkan rahmat serta hidayah-Nya.
2. Papah dan almarhumah Ibu yang telah mendidik, membesarkan, merawat, dan membiayai pendidikan saya hingga saat ini.
3. Mba nana dan mba tita, serta keluarga baruku bang kiky - cesa & bang asep – zira, nurul makasih buat perhatian dan semuanya.
4. Dr. Abdul Halim, selaku dosen pembimbing yang telah memberikan petunjuk, kemudahan dalam berpikir dan bimbingan dalam penyelesaian tugas akhir ini.
5. Dosen – dosen pengajar FT UI.
6. Anak – anak Ekstensi Elektro angkatan 2009 dan juga rekan skripsi saya nugroho nandar dyto.
7. Dan semua pihak yang tidak dapat saya sebutkan satu – per – satu.

Semoga penulisan ilmiah ini benar-benar dapat memberikan kontribusi positif dan menimbulkan sikap kritis kepada para pembaca khususnya dan masyarakat pada umumnya untuk senantiasa terus memperoleh wawasan dan ilmu pengetahuan di bidang teknologi dan sain.

Penulis menyadari keterbatasan pengalaman dan kemampuan yang tentu terdapat kekurangan serta kemungkinan jauh dari sempurna, untuk itu saya tidak

menutup diri dan mengharapkan adanya saran serta kritik dari berbagai pihak yang sifatnya membangun guna menyempurnakan penulisan ilmiah ini.

Akhir kata semoga penulisan ilmiah ini dapat memberikan manfaat bagi semua pihak yang bersangkutan, khususnya bagi saya dan umumnya bagi para pembaca.

Depok, Juni 2012

Ellan S. Siregar

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Ellan S. Siregar
NPM	: 0906 603 291
Program Studi	: Ekstensi Teknik Elektro
Departemen	: Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

"Rancang Bangun Prototipe Sistem Aktuator Kendali Sirip Menggunakan LabVIEW"

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 19 Juni 2012

Yang menyatakan

(Ellan S. Siregar)

ABSTRAK

Nama : Ellan S. Siregar
Program Studi : Ekstensi Teknik Elektro
Judul : Rancang Bangun Prototipe Sistem Aktuator Kendali Sirip Menggunakan LabVIEW

Prototipe sistem aktuator kendali sirip berbasis LabVIEW telah didesain dan dibuat. Sistem ini terdiri dari *brushed DC motor, planetary gear, bevel gear,* sensor rotasi dan perangkat lunak LabVIEW yang dipasang di komputer. Sistem ini dipergunakan untuk mengendalikan sudut putaran sirip. Kendali PID dipergunakan dalam sistem ini yang ditanamkan dalam mikrokontroler ATmega8538 dengan nilai Kp = 0.0037, Ki = 0.000022, dan Kd = 0.14985. Sudut referensi diberikan melalui LabVIEW dan diumpankan ke mikrokontroler melalui komunikasi serial. Dari hasil pengujian sistem diperoleh Tr = 0.42, Tp = 0.675, Ts = 0.8125, %OS = 5.375% dan *steady state error* = 14.75%.

Kata kunci :
Kendali sirip, LabVIEW, aktuator, ATmega8535, motor DC, PID, sensor rotasi, mikrokontroler, komunikasi serial, ADC

ABSTRACT

Name : Ellan S. Siregar
Study Program : Extensionof Electrical Engineering
Title : Development of Fin Control Actuator System Using LabVIEW

Prototype of fin control actuator system based on LabVIEW has been designed and built. System consist of brushed DC motor, planetary gear, bevel gear, fin, electronic driver circuit, microcontroller, rotary sensor and software LabVIEW that installed in computers. The system is used to regulate fin angular position. PID control has been explored and embedded in microcontroller Atmega8535 with the value of Kp = 0.0037, Ki = 0.000022, and Kd = 0.14985. Angular position reference has been set in LabVIEW and fed to microcontroller via serial communication. From system testing result, it has shown Tr = 0.42, Tp = 0.675, Ts = 0.8125, %OS = 5.375% and *steady state error* = 14.75%.

Keywords :
Fin rocket controller, LabVIEW, actuators, ATmega8535, DCmotors, PID, rotationsensors, microcontrollers, serialcommunication, ADC

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perubahan lingkungan merupakan tantangan sekaligus peluang bagi dunia sains dan teknologi untuk dapat memanfaatkannya secara maksimal. Kesadaran masyarakat akan kebutuhan, berkembang diakibatkan perubahan lingkungan sesuai kemajuan ekonomi, tingkat pendidikan, lingkungan sosial dan kemajuan teknologi.

Dalam bidang sistem informasi dan kemiliteran sangat dibutuhkan suatu alat yang canggih dan praktis dalam menunjang suatu pekerjaan itu sendiri. Mahasiswa sebagai salah satu Sumber Daya Manusia (SDM) yang berpotensi untuk mengembangkan dunia sains dan teknologi diharapkan dapat lebih mamahami berbagai macam peralatan serta sistem yang digunakan di dalamnya.

Salah satu contoh yang dirasa perlu untuk dikembangkan adalah roket kendali. Seperti yang kita ketahui perkembangan roket kendali di dunia sudah sangat pesat kemajuannya, begitu pula dengan dampaknya di Indonesia. perkembangan roket di Indonesia sendiri bisa di katakan berjalan secara dinamis. Jika dibandingkan dengan negara – negara di Asia Tenggara, kualitas penciptaan roket kendali di Indonesia tidak kalah canggih. Akan tetapi, jika di bandingkan dengan negara – negara di Asia seperti Jepang, Cina, Iran dan India, teknologi roket kendali di Indonesia masih kalah jauh. Di India misalnya, mereka sudah memiliki roket kendali yang mampu menjelajah luar angkasa.

Dalam penelitian kali ini, kami hanya memfokuskan pada pergerakan sirip roket kendali yang merupakan komponen penting dalam roket yang berfungsi sebagai pengatur arah gerak roket kendali. Sirip dikendalikan oleh aktuator untuk mendapatkan sudut pergerakan yang diinginkan. Aktuator yang digunakan adalah aktuator elektromekanik, yaitu aktuator yang menggunakan energi listrik sebagai sumber energi, disini motor DC sebagai komponen penggerak sirip roket kendali.

Motor DC yang berfungsi sebagai aktuator, akan mengatur pergerakan sirip roket kendali sebesar 10^0 sampai -10^0. Aktuator akan dikendalikan melalui LabVIEW untuk menentukan pergerakan sudut yang diinginkan. Pergerakan

aktuator dikendalikan melalui metode PWM (*pulse width modulation*) oleh LabVIEW. Untuk mengetahui apakah sirip bergerak sesuai dengan sudut yang diinginkan, maka digunakan sebuah sensor sudut. Sensor sudut yang digunakan adalah sensor sudut yang memiliki ketelitian hingga 300^0. Keluaran dari sensor sudut ini berupa tegangan. tegangan dapat berubah – ubah sesuai dengan sudut yang kita inginkan. Karena keluaran dari sensor sudut ini masih berupa keluaran analog, maka dibutuhkan sebuah konverter yang dapat mengubah keluaran analog menjadi keluaran digital agar sinyal dapat diproses oleh LabVIEW. Sinyal dari sensor akan diubah oleh mikrokontroler ATmega8535 menggunakan ADC (*analog to digital converter*). Sinyal kemudian akan dikirim melalui komunikasi serial ke LabVIEW sebagai umpan balik untuk mengetahui apakah sirip telah bergerak sesuai dengan sudut yang kita inginkan. Mudah-mudahan dengan gagasan dan rancangan sederhana yang kami buat menjadi suatu hal yang dapat diaplikasikan secara langsung dengan berbagai pengembangan teknologi yang lebih moderen khususnya di Indonesia.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dituliskan di atas, maka perumusan masalah akan ditekankan pada:

1. Bagaimana mendapatkan pergerakan sirip roket dengan ketelitian sebesar 1^0 dan dengan pergerakan sebesar 10^0 sampai -10^0.
2. Bagaimana mencari nilai ADC (*analog to digital converter*) pada sensor rotasi agar mendapatkan persamaan yang linier terhadap perubahan sudut.
3. Bagaimana mengendalikan kecepatan motor DC dengan menggunakan metode PWM (*pulse width modulation*).
4. Bagaimana membuat program pada mikrokontroler ATmega8535 untuk komunikasi serial antara mikrokontroler dan LabVIEW untuk mengendalikan pergerakan aktuator.
5. Bagaimana membuat program LabVIEW untuk pengendalian pergerakan aktuator agar bergerak sesuai dengan sudut yang diinginkan.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini antara lain:

1. Mempelajari, merancang, dan membuat prototipe sistem aktuator kendali sirip menggunakan *brushed DC motor, planetary gear,* dan *bevel gear*.
2. Mempelajari, merancang, dan membuat metode pengendalian pada prototipe sistem aktuator kendali sirip menggunakan PID dengan metode Ziegler – Nichols yang ditanamkan pada mikrokontroler ATmega8535.
3. Mempelajari, merancang, dan membuat sistem komunikasi menggunakan USB – *serial communication* antara mikrokontroler ATmega8535 dan LabVIEW sebagai piranti lunak untuk masukan dan tampilan pergerakan aktuator.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Hasil dari penelitian ini diharapkan dapat menjadi pedoman dan referensi untuk pengembangan pada penelitian berikutnya.
2. Pengaturan aktuator sirip pada roket diharapkan dapat diaplikasikan pada bidang lainnya, seperti aero modeling dan transportasi contohnya pada pergerakan sirip kapal.

1.5 Batasan Penelitian

Pembahasan penelitian ini dibatasi oleh beberapa hal, antara lain:

1. Perancangan driver motor DC
2. Pengendalian kecepatan motor DC dengan metode PWM
3. Pemrograman komunikasi serial pada mikrokontroler
4. Pemrograman kendali aktuator pada mikrokontroler ATmega8535
5. Perancangan LabVIEW sebagai piranti lunak untuk masukan dan tampilan pada aktuator
6. Penggunaan metode PID dalam pengendalian dengan kekhususan pada metode Ziegler – Nichols

1.6 Metode Penelitian

Metode yang digunakan untuk pengerjaan dan penulisan Tugas Akhir antara lain:

1. Studi Literatur

   Metode ini digunakan untuk memperoleh informasi tentang teori – teori dasar sebagai sumber penulisan skripsi. Informasi dan pustaka yang berkaitan dengan masalah ini diperoleh dari literatur, penjelasan yang diberikan dosen pembimbing, rekan – rekan mahasiswa, internet, *data sheet*, dan buku – buku yang berhubungan dengan tugas akhir penulis.

2. Perancangan dan Pembuatan Alat

   Perancangan alat merupakan tahap awal penulis untuk mencoba memahami, menerapkan, dan menggabungkan semua literatur yang diperoleh maupun yang telah dipelajari untuk melengkapi sistem serupa yang pernah dikembangkan, dan selanjutnya penulis dapat merealisasikan sistem sesuai dengan tujuan.

3. Uji Sistem

   Uji sistem ini berkaitan dengan pengujian alat serta hasil pembentukan dan pengendalian dari mesin tersebut.

4. Metode Analisis

   Metode ini merupakan pengamatan terhadap pengendalian posisi yang diperoleh dari pengujian alat tersebut. Setelah itu dilakukan penganalisisan sehingga dapat ditarik kesimpulan dan saran – saran untuk pengembangan lebih lanjut.

1.7 Sistematika Penulisan

Pada penulisan laporan Tugas Akhir ini, dapat dibuat urutan bab serta isinya secara garis besar. Diuraikan sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini berisi tentang latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian, metode penelitian dan sistematika penulisan.

BAB 2 TEORI DASAR

Teori dasar berisi landasan-landasan teori sebagai hasil dari studi literatur yang berhubungan dalam perancangan dan pembuatan program (*software*).

BAB 3 PERANCANGAN DAN CARA KERJA SISTEM

Pada bab ini akan dijelaskan secara keseluruhan sistem kerja dari semua program penghubung (*software*) yang terlibat.

BAB 4 PENGUJIAN ALAT DAN ANALISA HASIL

Bab ini berisi tentang unjuk kerja alat sebagai hasil dari perancangan sistem. Pengujian akhir dilakukan dengan menyatukan seluruh bagian-bagian kecil dari sistem untuk memastikan bahwa sistem dapat berfungsi sesuai dengan tujuan awal. Setelah sistem berfungsi dengan baik maka dilanjutkan dengan pengambilan data untuk memastikan kapabilitas dari sistem yang dibangun.

BAB 5 KESIMPULAN DAN SARAN

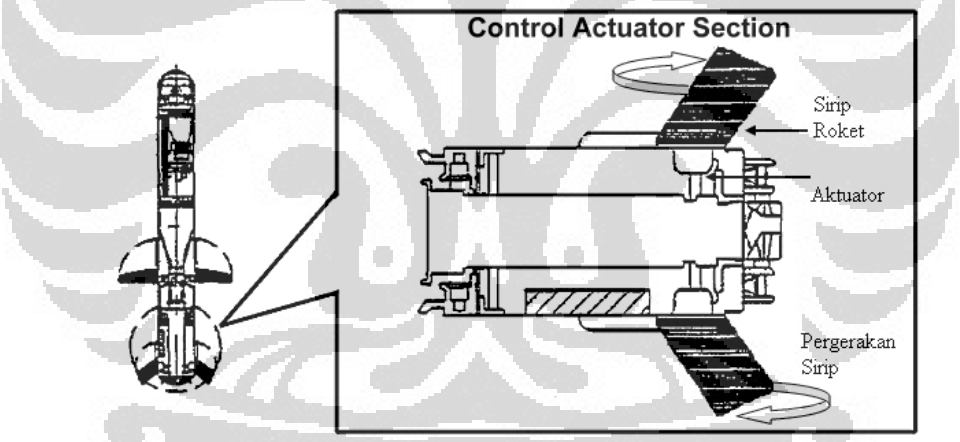
Penutup berisi kesimpulan yang diperoleh dari pengujian sistem dan pengambilan data selama penelitian berlangsung, selain itu juga penutup memuat saran untuk pengembangan lebih lanjut dari penelitian ini baik dari segi perangkat keras (*hardware*) dan program (*software*).

BAB 2
TEORI DASAR

Pada bab ini akan dijelaskan beberapa teori dasar yang menjadi acuan atau dasar penelitian pembuatan prototipe sistem aktuator kendali sirip. Hal-hal tersebut antara lain adalah teori dasar tentang motor DC, LabVIEW, Mikrokontroler ATmega8535, dan komponen-komponen penunjang, baik pasif maupun aktif.

2.1 Sistem Gerak Sirip Roket

Sirip roket merupakan komponen penting dalam sistem gerak sebuah roket atau misil. Sirip roket berfungsi untuk mengendalikan arah gerak roket dan juga keseimbangan dari roket.



Gambar 2.1 Sistem Sirip Roket

Sirip roket akan digerakkan oleh sebuah aktuator dan akan di kendalikan oleh sistem pengendali roket. Pergerakan sisip roket akan di rancang bergerak sebesar 10 derajat ke kanan, dan 10 derajat ke kiri dengan ketelitian 1 derajat pada setiap masukan (Gambar 2.1). Sensor yang digunakan untuk pergerakan sirip roket adalah komponen yang sangat penting dari pergerakan sirip roket, sensor harus memiliki kestabilan dan ketelitian tinggi agar pergerakan sirip yang dihasilkan sesuai dengan yang diinginkan.Pergerakan sirip roket juga harus kuat

terhadap gangguan luar seperti beban angin akibat dari kecepatan yang dihasilkan oleh pendorong roket.

2.2 Motor DC

Arus listrik adalah kumpulan muatan-muatan yang bergerak. Apabila seutas kawat mengalirkan arus listrik dan ditempatkan tegak lurus pada sebuah medan magnetik, maka timbul gaya pada kawat tesebut, gaya ini dinamakan gaya magnetik.

Komponen utama yang diperlukan dalam pembuatan prototipe sistem aktuator kendali sirip salah satunya adalah motor DC, Prinsip kerja motor DC dapat dijelaskan dengan teori elektromagnetikyaitu jika sebuah penghantar kawat berarus listrik ditempatkan di daerah medan magnet maka akan mengalami gaya magnet yang besarnya sebanding dengan arus yang melewati penghantar, besar medan magnet, panjang kawat penghantar dan sudut antara medan magnet dengan arus, jika ditulis dalam sebuah persamaan adalah:

$$F = B \cdot i \cdot L \sin \alpha \qquad (2.1)$$

Dimana:

F = gaya magnet (Newton)

B = medan magnet luar (Wb/m2)

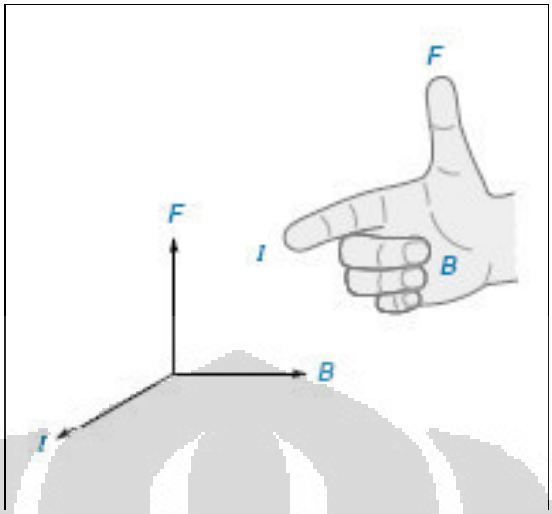
i = kuat arus (Ampere)

L = panjang kawat (Meter

α = sudut yang dibentuk medan magnetik dengan arus

Dari Persamaan 2.1 dapat dilihat bahwa F akan bernilai nol bila medan magnetik sejajar dengan arus listrik, dan akan bernilai maksimum ketika medan megnetik tegak lurus dengan arah arus listrik.

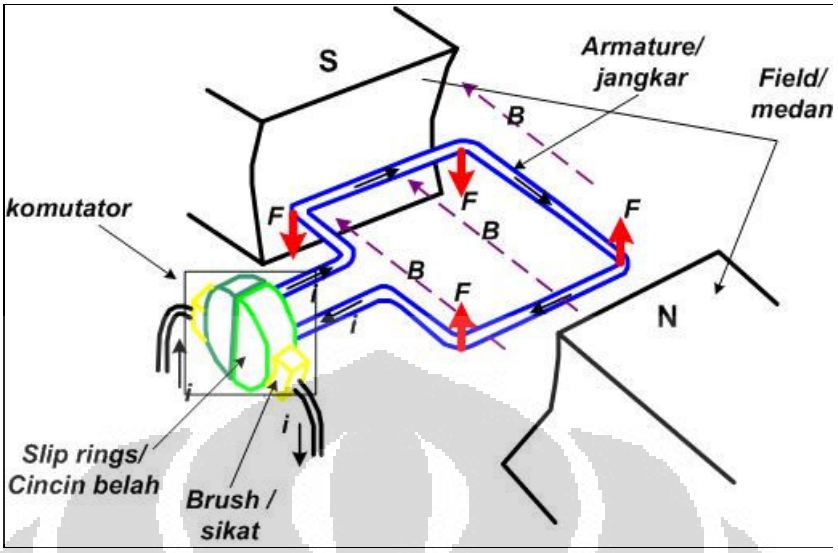
Arah gaya magnet, medan magnet, dan kuat arus dapat ditentukan dengan menggunakan kaedah tangan kanan (Gambar 2.2).



Gambar 2.2 Kaidah Tangan Kanan

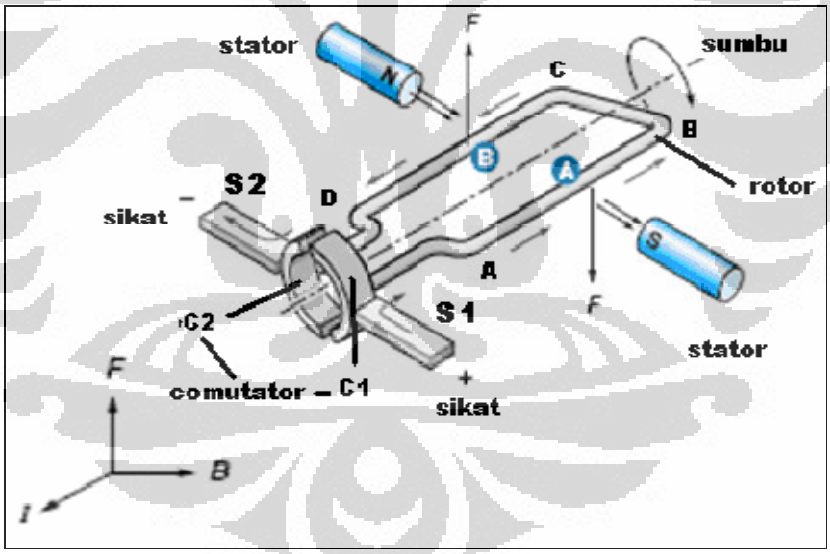
Motor dcterdiri dari bagian-bagian yang dapat menggerakkan motor tersebut (Gambar 2.3), yaitu:

- Rotor, yaitu bagian yang berputar pada motor berupa kumparan kawat. Bagian ini berupa inti besi yang memiliki kumparan (*coil*) berupa lilitan kawat dan sebuah komutatoryang menjadi penghubung antara lilitan kawat dengan sumber tegangan DC yang akan diberikan melalui sikat arang pada bagian stator.
- Stator, yaitu bagian yang diam pada motor berupa magnet.
  Statorini menghasilkan medan magnet, biasanya berasal dari sebuah magnet permanen walaupun terdapat juga motor DC yang memiliki medan magnet yang dihasilkan oleh lilitan kumparan (*coil*) pada sebuah inti besi seperti halnya pada bagian rotor.
- Komutator, yaitu cincin belah yang berfungsi sebagai penukar arus.
- Sikat, yaitu sepasang batang grafit yang menempel pada komutator tetapi tidak berputar.



Gambar 2.3 Penampang Motor DC

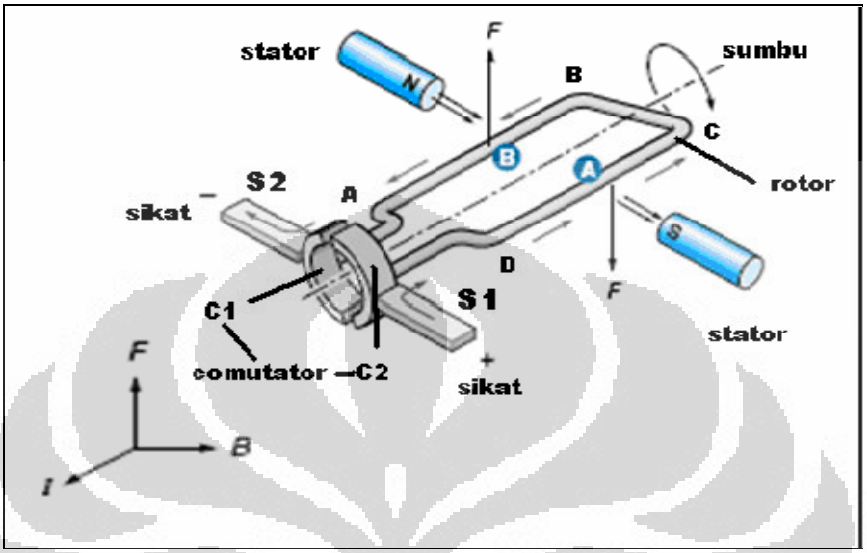
Cara kerja motor DC dapat dijelaskan dengan melihat gambar dibawah ini:



Gambar 2.4 Posisi Awal Gerakan Motor

Misalkan kedudukan mula-mula seperti pada gambar 2.4 arus listrik mengalir dari kutub (+) baterai melalui sikat S1 – cincin C1- rotor ABCD – cincin C2 – sikat S2 – kembali ke kutub (-) baterai. Ketika rotor CD yang dekat dengan kutub utara mengalami gaya ke atas dan sisi rotor AB yang dekat dengan kutub

selatan mengalami gaya ke bawah.akiibatnya rotor ABCD berputar searah jarum jam.



Gambar 2.5 Posisi Motor Setelah 180^0

Setelah setengah putaran (180^0), terjadi pertukaran posisi antara sikat dan comutator. Sekarang, C2 menyentuh sikat S1 dan C1 menyentuh sikat S2. Sehingga arus mengalir dari kutub (+) baterai menuju kutub (-) melalui sikat 1 (S1), komutator 2 (C2), Rotor DCBA, komutator 2 (C2), dan sikat 2 (S2) yang terlihat pada Gambar 2.5. Pertukaran posisi antara sikat dan komutator mengakibatkan motor terus berputar.

Selama motor berputar menghasilkan torka (τ = *torque*) seperti yang terlihat pada Gambar 2.6. Torka merupakan analogi gaya dari gerak translasi untuk gerak rotasi. Karena torka ini dihasilkan oleh sistem elektromagnetik, maka disebut torka elektromagnetik (*electromagnetic torque*). Torka yang dihasilkan motor mempunyai nilai yang besarnya ditunjukkan pada Persamaan 2.2 berikut:

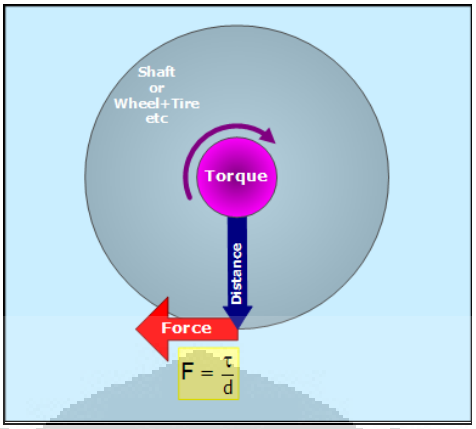
$$\tau = rF \qquad (2.2)$$

Dimana:

τ = torka (Nm)

r = jarak dari pusat rotasi ke titik beban (m)

F = gaya yang ditimbulkan medan magnet(N)



Gambar 2.6 Prinsip Torka

Sedangkan ketika terjadi putaran persamaan torka menjadi seperti yang diuraikan pada Persamaan 2.3:

$$\tau = I\alpha \quad (2.3)$$

Dimana:

τ = torka putaran

I = momen inersia

α = kecepatan sudut (rad/s)

Adapun cara mengubah putaran motor DC adalah salah satunya dengan mengubah arah aliran arus yang melewati rotor. Sedangkan kecepatan putaran motor dc dapat ditingkatkan dengan memperbesar tegangan yang masuk ke motor, sehingga dapat mengakibatkan arus yang masuk ke motor menjadi besar pula. Hal ini sesuai dengan hukum Kirchoff berikut ini:

$$V = i.R \quad (2.4)$$

Dimana:

V = Tegangan (Volt)

i = Besar arus (Amepere)

R = Hambataan (Ohm)

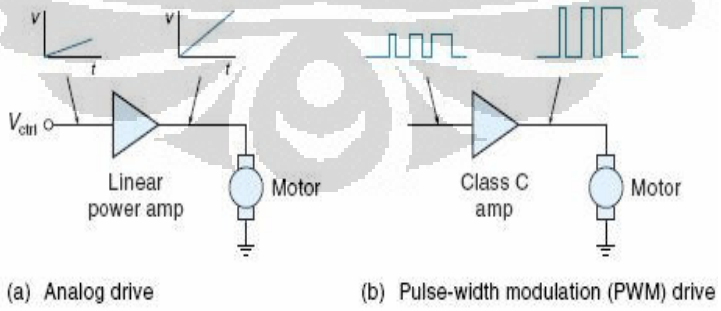
Dengan hambatan yang tetap dan tegangan diperbesar akan mengakibatkan arus menjadi besar pula. Dengan arus yang diperbesar maka akan menyebabkan gaya (F) menjadi besar pula sesuai dengan Persamaan 2.1. Dan

apabila F semakin besar maka kekuatan rotor akan semakin besar dan berdampak pada makin cepatnya putaran motor. Dari Persamaan 2.1 dapat diambil kesimpulan bahwa semua unsur yang mempengaruhi nilai F dapat mempercepat putaran motor, yaitu dengan memperpanjang lilitan, memperbesar i , dan memperbesar medan magnet (B).

2.2.1 Cara Pengendalian Motor DC

Ada 2 cara untuk mengontrol kecepatan dari motor DC. Kontrol kecepatan merupakan sesuatu yang tidak akurasi dikarenakan motor yang dari energi listrik diubah menjadi torsi dan tidak diubah menjadi kecepatan. Kecepatan yang presisi ditentukan oleh torsi motor dan beban mekanik. Untuk mengendalikan motor diperlukan rangkaian *interface* yang dapat mengubah sinyal motor level rendah dari *controller* menjadi sebuah sinyal yang cukup besar untuk menggerakkan motor dan cara ini disebut *analog drive*. Pada metode ini, sebuah penguatan *power amplifier* adalah linier dari *controller* dan tegangan analog diberikan ke motor. Teknik yang lain untuk mengontrol sebuah motor dc adalah *pulse-width modulation* (PWM) seperti yang terlihat pada Gambar 2.7.

Dalam sistem ini, power di suplai ke motor dalam bentuk pulsa dc pada tegangan tertentu. Lebar pulsa bervariasi untuk mengontrol kecepatan motor. Pulsa yang lebih lebar, tegangan dc rata-rata yang lebih tinggi diperbolehkan untuk motor. Frekuensi dari pulsa yang cukup tinggi dapat menginduktansi rata-rata motor sehingga dapat menggerakkan motor secara baik.

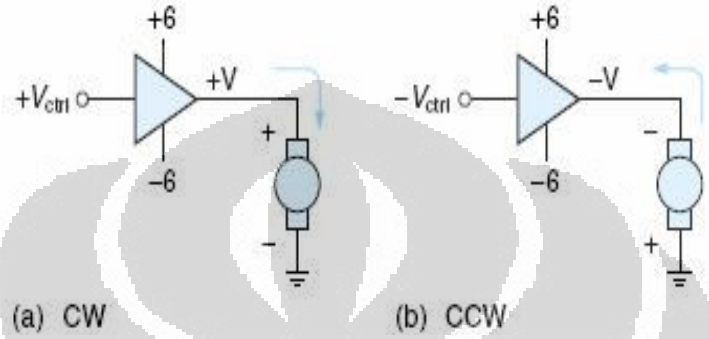


Gambar 2.7 Cara Pengendalian Motor

Untuk mengubah arah rotasi dari motor DC, polaritas dari tegangan yang digunakan adalah berlawanan. untuk dapat melakukannya sehingga sebuah motor

driver mampu mengeluarkan tegangan positif dan tegangan negatif. Ketika tegangannya positif dan *ground* maka motor akan bergerak searah jarum jam.

Ketika tegangannya negatif dan *ground* maka polaritas tegangan pada terminal motor berlawanan sehingga motor bergerak berlawanan arah jarum jam (Gambar 2.8).

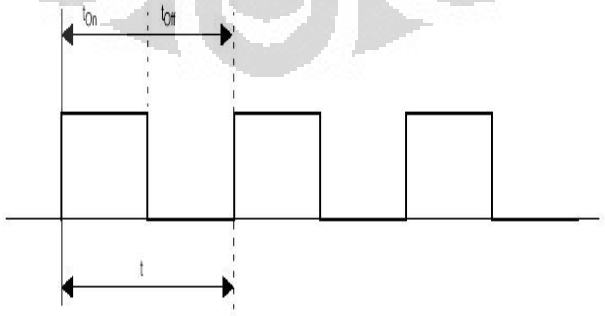


Gambar 2.8 Arah Putaran Motor DC

2.2.2 PWM (*Pulse Width Modulation*)

Suatu teknik yang digunakan untuk mengontrol kerja dari suatu alat atau menghasilkan suatu tegangan DC yang variabel adalah PWM (*Pulse Width Modulation*). Rangkaian PWM adalah rangkaian yang lebar pulsa tegangan keluarannya dapat diatur atau dimodulasi oleh sinyal tegangan modulasi.

Disamping itu kita dapat menghasilkan suatu sinyal PWM dengan menentukan frekuensi dan waktu dari variabel ON dan OFF. Pemodulasian sinyal yang beragam dapat menghasilkan *duty cycle* yang diinginkan sehingga memperlihatkan sinyal kotak dengan *duty cycle* 50% (Gambar 2.9).



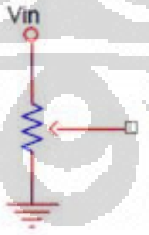
Gambar 2.9 Sinyal PWM Dengan *Duty Cycle* 50 %

2.3 Sensor Sudut Putar (*Rotation Sensor*)

Sensor sudut putar merupakan sensor analog yang digunakan untuk mendeteksi sudut putar suatu benda (Gambar 2.10). Sensor ini memiliki sudut putar maksimum 300^0. Sensor ini memiliki 3 buah pin konektor, pin pertama sabagai *output* tegangan (Vout), pin kedua sebagai *grounding,* dan pin ketiga sebagai inputan tegangan (Vin).

Gambar 2.10 Sensor Sudut Putar

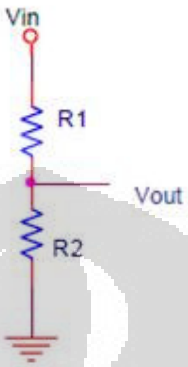
Prinsip kerja dari sensor rotasi analog ini cukup sederhana, bagian sensor yang dapat berputar merupakan sebuah potensiometer atau variabel resistor (Gambar 2.11), variabel resistor adalah resistor yang nilai hambatannya dapat di ubah – ubah.



Gambar 2.11 Skematik Variabel Resistor

Prinsip kerja alat ini adalah membagi tegangan (Gambar 2.12). Tegangan pada vout akan berubah ketika variabel resistor kita putar. Vin merupakan tegangan maksimum yang dapat keluar melalui output vout. Jika vin sebesar 5 volt, maka tegangan output yang dapat kita atur adalah 0-5 volt, jika vin sebesar

10 volt, maka tegangan output yang dapat kita atur sebesar 0-10 volt, begitu seterusnya. Untuk mendapatkan nilai tegangan pada *output* dapat kita peroleh melalui Persamaan 2.5 berikut:



Gambar 2.12 Rangkaian Pembagi Tegangan

$$Vout = \frac{R2}{R1 + R2} Vin \tag{2.5}$$

Dimana:

Vout : Tegangan keluaran dari variabel resistor

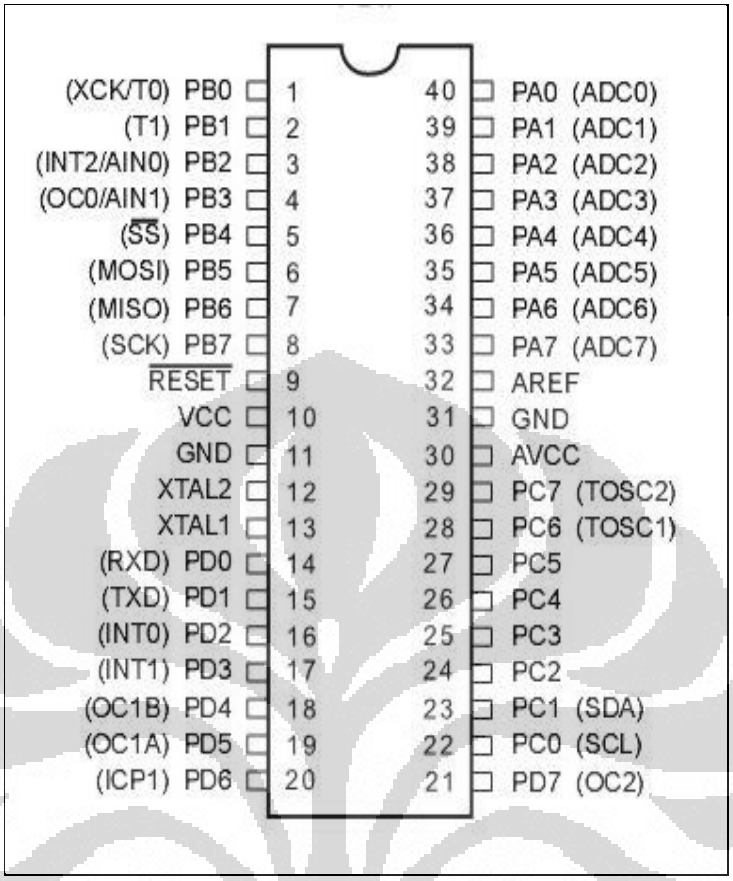
Vin : Tegangan masukan yang diberikan pada variabel resistor

R1 : Nilai tahanan 1

R2 : Nilai tahanan 2

2.4 Mikrokontroler ATmega8535

Mikrokontroler ATmega8535 (keluarga Atmel) mempunyai 40 kaki, 32 kaki diantaranya digunakan sebagai port paralel. Satu port paralel terdiri dari 8 kaki, sehingga 32 kaki tersebut membentuk 4 buah port paralel, yang masing-masing dikenal sebagai Port 0/Port a, Port 1/Port b, Port 2/Port c, dan Port 3/Port d. nomor dari masing-masing jalur (kaki) pertama Port a disebut sebagai Porta.0 dan jalur (kaki) terakhir untuk Port d adalah Portd.7. Perhatikan gambar 2.13 untuk diagram pin ATmega8535.



Gambar 2.13 Diagram Pin ATmega8535

ATmega8535 adalah suatu low-power CMOS 8-bit microcontroller berbasiskan pada AVR tingkatkan RISC arsitektur. Dengan instruksi *single clock* siklus, ATmega8535 hampir mendekati 1 MIPS (tekanan induksi rata – rata) per MHZ sehingga dapat membantu perancang sistem untuk mengoptimalkan konsumsi tanaga dan kecepatan proses.

Fungsi-fungsi kaki (pin):

- Vcc

  Suplai tegangan (5 volt dc).

- GND

  Grounding atau pentanahan

- RESET

Masukan reset. Kondisi logika ‘1’ selama siklus mesin saat osilator bekerja akan mereset mikrokontroler yang bersangkutan.

- XTAL1

Inputan (masukan) untuk inverting osilator amplifier dan masukan untuk *internal clock operation circuit.*

- XTAL2

Output (keluaran) dari inverting osilator amplifier.

- AVCC

AVCC adalah voltase pin untuk PortA dan A/D konverter. Haruslah dikoneksikan secara eksternal pada Vcc, sekalipun ADC sedang tidak digunakan.

- AREF

AREF adalah referensi pin analog untuk A/D converter.

- PORT A

Port A berfungsi sebagai analog input (masukan) pada A/D converter. Port A juga berfungsi sebagai 8-bit bi-directional I/O port jika A/D converter tidak digunakan. Pin pada port dapat menyediakan internal pull-up resistor. Port suatu output buffers mempunyai karakteristik pengarah simetris.

Ketika pin PA0 sampai PA7 digunakan sebagai input, dan semuanya memiliki pull-up eksternal yang rendah, maka internal pull-up resistor yang terdapat pada port harus diaktifkan. Pin pada port A akan bersifat *tri-stated* ketika kondisi *reset* menjadi aktif, sekalipun *clock* sedang tidak jalan. Untuk lebih jelas mengenai fungsi port A dapat dilihat pada Tabel 2.1.

Tabel 2.1 Alternatif Fungsi Pin pada Port A

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

- PORT B

Port B berfungsi sebagai 8-bit bi-directional I/O. Pin pada port dapat menyediakan internal pull-up resistor. Port suatu output buffers mempunyai karakteristik pengarah simetris.

Ketika pin PB0 sampai PB7 digunakan sebagai input, dan semuanya memiliki pull-up eksternal yang rendah, maka internal pull-up resistor yang terdapat pada port harus diaktifkan. Pin pada Port B akan bersifat *tri-stated* ketika kondisi *reset* menjadi aktif, sekalipun *clock* sedang tidak jalan.

Port B juga memiliki beserapa fungsi khusus yang terdapat pada ATmega8535 (Tabel 2.2).

Tabel 2.2 Alternatif Fungsi Pin pada Port B

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

- PORT C

Port C berfungsi sebagai 8-bit bi-directional I/O. Pin pada port dapat menyediakan internal pull-up resistor. Port suatu output buffers mempunyai karakteristik pengarah simetris.

Ketika pin PC0 sampai PC7 digunakan sebagai input, dan semuanya memiliki pull-up eksternal yang rendah, maka internal pull-up resistor yang terdapat pada port harus diaktifkan. Pin pada Port C akan bersifat *tri-stated* ketika kondisi *reset* menjadi aktif, sekalipun *clock* sedang tidak jalan. Untuk lebih jelas mengenai fungsi port C dapat dilihat pada Tabel 2.3.

Tabel 2.3 Alternatif Fungsi Pin Pada Port C

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

- PORT D

Port D juga berfungsi sebagai 8-bit bi-directional I/O. Pin pada port dapat menyediakan internal pull-up resistor. Port suatu output buffers mempunyai karakteristik pengarah simetris.

Ketika pin PD0 sampai PD7 digunakan sebagai input, dan semuanya memiliki pull-up eksternal yang rendah, maka internal pull-up resistor yang terdapat pada port harus diaktifkan. Pin pada PortD akan bersifat *tri-stated* ketika kondisi *reset* menjadi aktif, sekalipun *clock* sedang tidak jalan.

Port D juga memiliki beberapa fungsi khusus yang terdapat pada ATmega8535 (Tabel 2.4).

Tabel 2.4 Alternatif Fungsi Pin Pada Port D

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

2.4.1 Instruksi dalam BASCOM AVR

Alv and Vegard's Risc Processor atau biasa di singkat AVR merupakan mikrokontroler Reduced Instruction Set Computer (RISC) 8 bit, instruksi yang dimiliki terbatas, tetapi memiliki fasilitas yang lebih banyak. Karena RISC inilah sebagian besar kode instruksinya dikemas dalam satu siklus clock. AVR adalah jenis mikrokontroler yang paling sering digunakan dalam bidang elektronika dan instrumentasi.

Secara umum, AVR dapat dikelompokkan dalam 4 kelas. Pada dasarnya yang membedakan masing – masing kelas adalah memori, peripheral dan fungsinya. Keempat kelas tersebut adalah keluarga ATtiny, keluarga AT90Sxx, keluarga ATmega dan AT86RFxx.

Secara keseluruhan BASCOM AVR mempunyai beberapa macam instruksi, yang dibentuk dengan menkombinasikan dan operan. Penjelasan berikut adalah beberapa instruksi yang dipakai dalam pemrograman pada pembuatan tugas akhir ini :

- DO – LOOP

Instruksi Do – Loop berfungsi untuk mengulangi suatu blok statemen sampai kondisi adalah benar atau kita juga dapat keluar dari suatu Do - Loop dengan statemen Exit Do. Do - Loop selalu dilakukan sedikitnya sekali. Sintaksis instruksi Do - Loop adalah sebagai berikut :

```
Do
        Statements
```

Loop [Until expression]

Instruksi Do – Loop akan terus menerus mengerjakan suatu blok statemen yang berada di dalam suatu Do – Loop sampai kondisi terpenuhi. misalnya jika kita ingin menggerakan sebuah motor dc sampai motor dc itu berhenti ketika terkena sebuah sensor Limit Switch (aktif *high*). Contoh programnya adalah :

```
Do
        M1 = 1              ; variable M1 berlogika 1 atau high
        M2 = 0              ; variable M2 berlogika 0 atau low
Loop Until Ls = 1     ; mengulang blok statemen sampai variable Ls = 1
```

Maksud variabel nilai M1 dan M2 diberi logika 1/0 adalah untuk mengkondisikan gerak motor ke kanan atau ke kiri.

- IF – THEN

Instruksi If – Then digunakan jika hanya terdapat satu syarat yang digunakan untuk mrnguji apakah suatu instruksi akan diproses atau tidak, jika syarat terpenuhi maka instruksi tersebut akan diproses terlebih dahulu sebelum mekanjutkan ke eksekusi baris instruksi selanjutnya, jika syarat tidak terpenuhi maka program akan langsung melanjutkan dengan mengeksekusi baris instruksi selanjutnya. Sintaksis instruksi If – Then adalah sebagai berikut :

```
If expression (syarat) Then
        statements (instruksi)
End If
```

- IF – THEN – ELSE

Instruksi If – Then – Else digunakan jika terdapat satu syarat yang digunakan untuk memutuskan alternatif instruksi mana yang akan diproses, jika syarat terpenuhi maka instruksi akan diproses lebih dahulu sebelum mengeksekusi baris instruksi yang selanjutnya, jika syarat tidak terpenuhi maka program akan mengeksekusi program instruksi dibawahnya sebelum melanjutkannya dengan mengeksekusi baris instruksi selanjutnya. Sintaksis instruksi If – Then - Else adalah sebagai berikut :

```
If expression (syarat) Then
        statements (instruksi)
```

```
Else
        Statements-2 (instruksi-2)
End If
```

contoh penggunaan instruksi If – Then – Else :

```
If x > y then
        Lcd " X Lebih besar dari Y"
Else
        Lcd " X Lebih kecil dari Y"
End If
```

- IF – THEN – ELSEIF

Instruksi If – Then – Elseif digunakan jika terdapat beberapa syarat yang digunakan untuk memutuskan alternatif instruksi mana yang akan diproses, misalkan terdapat beberapa pilihan huruf, "a", "b", dan "c". jika syarat pada "a" terpenuhi, maka instruksi akan diproses. Namun jika sayarat pada "a" tidak terpenuhi, maka instruksi yang akan diproses adalah instruksi pada syarat "b" atau "c". begitu pula dengan "b", jika syarat pada "b" terpenuhi, maka instruksi akan diproses. Namun jika sayarat pada "b" tidak terpenuhi, maka instruksi yang akan diproses adalah instruksi pada syarat "a" atau "c". begitu pula sebaliknya. Sintaksis instruksi If – Then - Elseif adalah sebagai berikut :

```
If expression (syarat) Then
        statements (instruksi)
Elseif expression (syarat) Then
        Statements-2 (instruksi-2)
Elseif expression (syarat) Then
        Statements-3 (instruksi-3)
        .
        .
        .
        .
Elseif expression (syarat) Then
        Statements-n (instruksi-n)
End If
```

contoh penggunaan instruksi If – Then – Elseif :

```
If A = 0 then
        Lcd " Nilai A adalah 0"
Elseif A = 1 then
        Lcd " Nilai A adalah 1"
Elseif A = 2 then
        Lcd " Nilai A adalah 2"
End If
```

- FOR – NEXT

Instruksi for – next adalah sebuah instruksi untuk perulangan yang hanya menggunakan sebuah variable counter. Untuk perulangan yang bersifat menambahkan, kita mesti menggunakan To, sedangkan untuk perulangan yang bersifat mengurangkan, kita harus menggunakan Step yang nilainya negatif. Sintaksis instruksi for - next adalah sebagai berikut :

```
For var = start To end [Step value]
        statements (instruksi)
Next var
```

contoh penggunaan instruksi For - Next :

```
For A = 1 To 10 Step 1
        Lcd "This is A " ; A
Next A
```

Dari contoh program di atas, maka instruksi untuk menampilkan tulisan pada LCD dalah sebanyak 10 kali.

2.4.2 ADC ATmega8535

ATmega8535 menyediakan fasilitas ADC dengan resolusi 10 bit. ADC ini dihubungkan dengan 8 *channel analog multiplexer* yang memungkinkan terbentuk 8 input tegangan *single ended* yang masuk melalui pin pada PortA. ADC memiliki pin supply tegangan analog yang terpisah yaitu AVCC. Besarnya tegangan AVCC adalah ±0.3V dari VCC. Tegangan referensi ADC dapat dipilih menggunakan tegangan referensi internal maupun eksternal. Jika menggunakan tegangan

referensi internal, bisa dipilih on-chip internal reference voltage yaitu sebesar 2.56V atau sebesar AVCC. Jika menggunakan tegangan referensi eksternal, dapat dihubungkan melalui pin AREF. ADC mengkonversi tegangan input analog menjadi data digital 8 bit atau 10 bit. Untuk mencari nilai resolusi ADC, digunakan Persamaan 2.6 berikut:

$$\text{Resolusi} = \left(\frac{\text{Tegangan Referensi}}{2^n - 1} \right) \quad (2.6)$$

Dimana:

n = menyatakan jumlah bit keluaran biner IC ADC

berikut merupakan listing program yang ditanamkan pada chip untuk membaca nilai ADC:

```
Dim A As Byte
Do
        A = Inkey()
        If A > 0 Then
                Print "ASCII code " ; A ; " from serial"
        End If
Loop Until A = 27
End
```

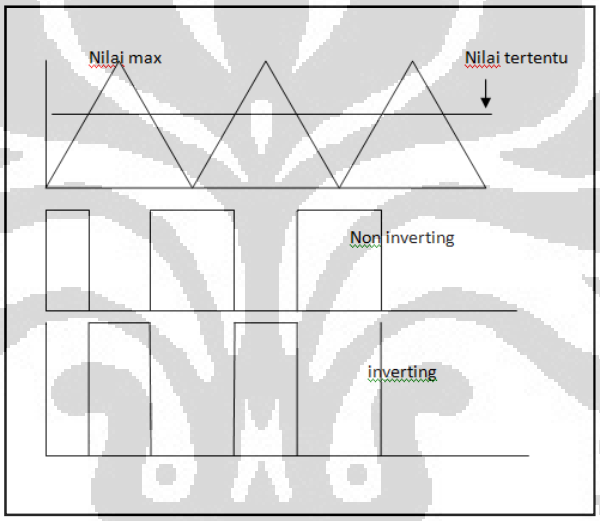
ADC akan berhenti membaca data ketika nilai veriabel A=27, 27 merupakan nilai ASCII dari tombol Esc (*escape*) pada *keyboard* komputer

2.4.3 PWM Atmega8535

PWM (*Pulse Width Modulation*) atau modulasi lebar pulsa adalah salah satu keunggulan *Timer/Counter* yangterdapat pada ATmega8535. Ketiga jenis *Timer/Counter* pada Atmega8535 dapat menghasilkan pulsa PWM. Pulsa PWM adalah sederetan pulsa yang lebar pulsanya dapat diatur. Pulsa PWM berfungsi mengatur kecepatan motor DC, mengatur gelap terang LED dan aplikasi lainnya. PWM adalah *Timer mode Output Compare* yang canggih. Mode PWM *Timer* juga dapat mencacah turun yang berlawanan dengan mode *Timer* lainnya yang hanya mencacah naik. Pada mode PWM tersebut, Timer mencacah naik hingga mencapai nilai TOP, yaitu 0xFF (255) untuk PWM 8 bit dan 0x3FF (1023) untuk

PWM 10 bit. *Timer/Counter* 0 hanya memiliki PWM 8 bit, sedangkan pada *Timer/Counter* 1 memiliki 9 bit dan PWM 10 bit, dan *Timer/Counter* 2 memiliki PWM 8 bit.

Pemilihan mode PWM diseting melalui bit COM01 dan bit COM00 pada register TCCR. Saat COM00 *clear* dan COM01 set, pin OC0 *clear* sat timer mencacah diatas *Compare Match* dan pin OC0 set saat timer mencacah dibawah *Compare Match* atau *non-inverting* PWM. Kebalikannya, saat COM00 set dan COM01 juga set, maka pin OC0 set saat timer mencacah dibawah *Compare Match* atau disebut juga *inverting* PWM (Gambar 2.14).



Gambar 2.14 Modulasi Lebar Pulsa

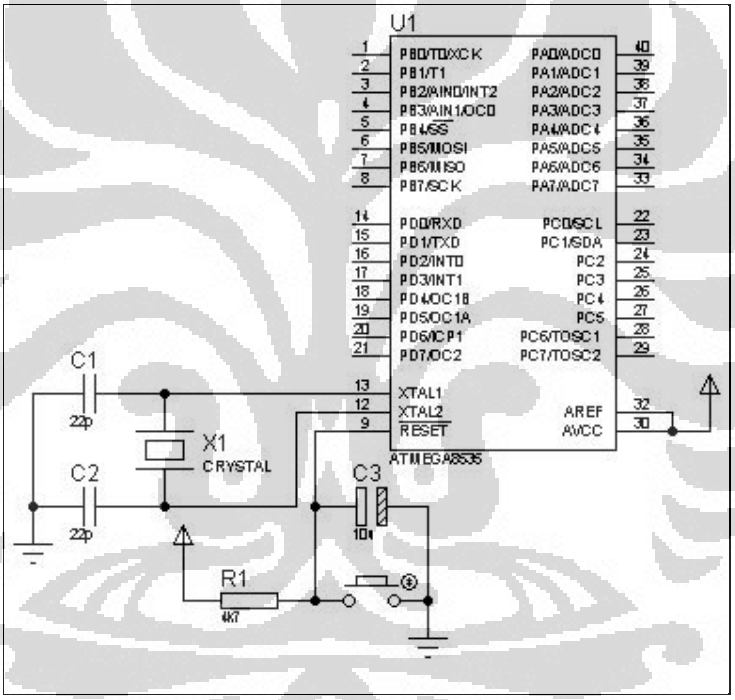
Berikut merupakan program pada ATmega8535 untuk menggunakan PWM:

```
Config Timer1 = Pwm , Pwm = 10 , Prescale = 64 , Compare A Pwm = Clear
Down , Compare B Pwm = Clear Down
Do
        Pwm1a = 512 ‘pin OC1A
        Pwm1b = 512 ‘pin OC1B
Loop
End
```

Program PWM di atas dengan setingan *fast* PWM sebesar 10 bit, dan memiliki mode inverting.

2.4.4 Pemberian *Clock* pada Mikrokontroler

Kecepatan suatu prosesor menunjukkan kemampuan processor tersebut untuk dapat mengeksekusi suatu perintah (*command*). Pada mikrokontroler kecepatan *central processing unit* untuk dapat mengeksekusi suatu perintah sangat tergantung pada clock mikrokontroler itu sendiri. Untuk menggunakannya, hubungkan sebuah resonator kristal atau keramik diantara kaki-kaki XTAL1 dan XTAL2 pada mikrokontroler dan hubungkan kapasitornya ke ground. Sedangkan contoh bagaimana mengaktifkan *clock* menggunakan osilator eksternal ditunjukkan pada gambar di bawah ini.



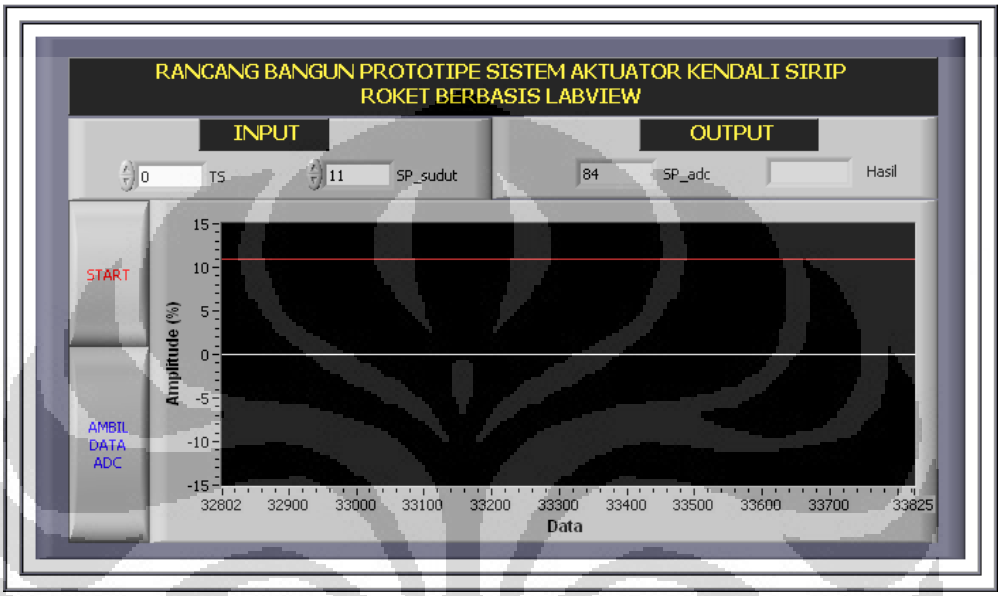
Gambar 2.15 Pemberian *Clock* Pada Mikrokontroler

Siklus mesin mikrokontroler dapat diatur menggunakan kristal 12 MHz sehingga kecepatan siklus mesin menjadi 12 MHz/12 = 1 MHz, yang artinya periode detak waktunya 1 mikrodetik.

2.5 LabVIEW (*Laboratory Virtual Instrumentation Engineering Workbench*)

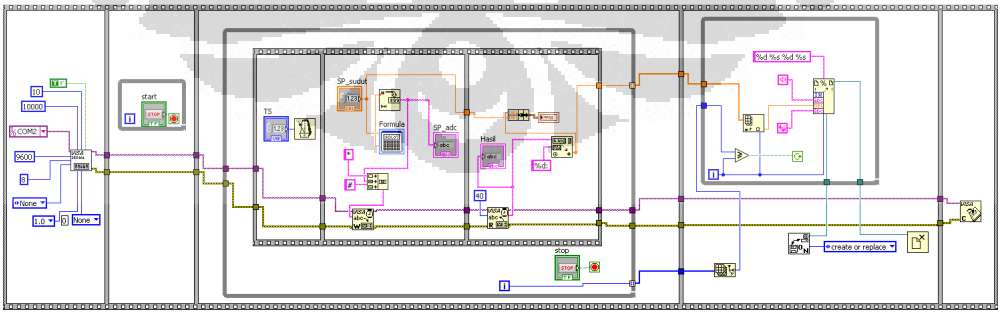
LabVIEW (singkatan dari *Laboratory Virtual Instrumentation Engineering Workbench*) adalah perangkat lunak komputer untuk pemrosesan dan

visualisasi data dalam bidang akuisisi data, kendaliinstrumentasi serta automasi industri yang pertama kali dikembangkan oleh perusahaan National Instruments pada tahun 1986. Perangkat lunak ini dapat dijalankan pada sistem operasiLinux, Unix, Mac OS X dan Windows.



Gambar 2.16 *Front Panel* Pada LabVIEW

Program LabVIEW disebut dengan Virtual Instrument (VI) karena beberapa tampilan dan operasi pada program LabVIEW menyerupai suatu instrument seperti multimeter dan osiloskop.



Gambar 2.17 *Block Diagram* Pada LabVIEW

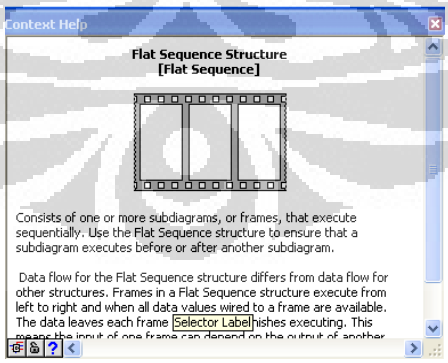
Setiap VI menggunakan fungsi – fungsi yang memanipulasi input dari *user interface* atau sumber lain dan menampilkan informasi tersebut atau memindahkan informasi tersebut ke file/komputer lain. LabVIEW terdiri dari tiga komponen utama, yaitu:

1. *Front panel*, merupakan *user interface* (Gambar 2.16)
2. *Block diagram*, merupakan tempat memprogram LabVIEW dan mendefinisikan fungsi – fungsi dari VI (Gambar 2.17)
3. *Icon* dan *connector panel*

Sama halnya dengan bahasa pemrograman yang lainnya yaitu C++, MatLab, dan Visual Basic, bahasa pemrogramman pada LabVIEW juga memiliki fungsi dan peranan yang sama. Pemrograman pada LabVIEW menggunakan bahasa pemrograman berupa blok diagram atau grafis yang dikenal dengan sebutan VI (*Virtual Instrument*), sementara pada bahasa pemrograman yang lain menggunakan format teks. Berikut merupakan beberapa contoh program pada LabView yang digunakan pada penelitian kali ini:

a. Flat Sequence Structure

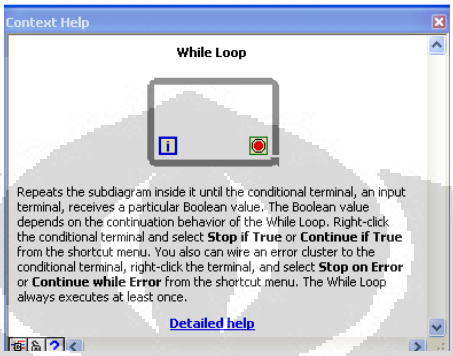
Merupaka program pada LabVIEW yang berfungsi untuk mengurutkan jalannya program, sehingga program terstruktur dengan baik. Kolom – kolom pada *flat sequence structure* (Gambar 2.18) merupakan langkah – langkah dimana program akan dijalankan satu persatu mulai dari kiri ke kanan.



Gambar 2.18 *Flat Sequence Structure*

b. While Loop

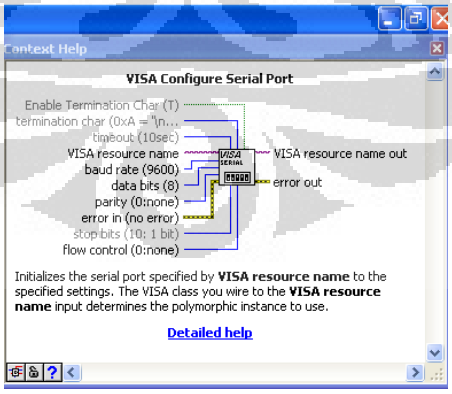
Merupakan program pada LabVIEW yang berfungsi untuk memutar jalannya program (Gambar 2.19). Program akan terus berputar (*looping*) dan baru akan berhenti jika kondisi pada fungsi while loop telah terpenuhi.



Gambar 2.19 *While Loop*

c. VISA Configure Serial Port

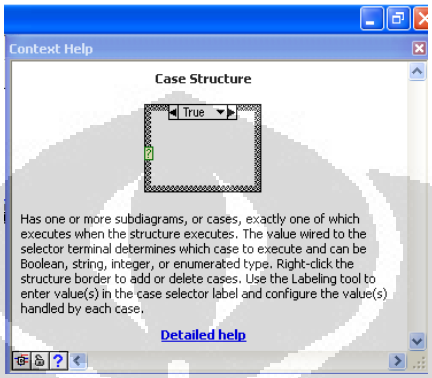
Merupakan fasilitas pada LabVIEW untuk berkomunikasi melalui komunikasi serial (Gambar 2.20). *VISA configure serial port* digunakan untuk menginisialisasi port komunikasi pada komputer, baik itu melalui port RS232 atau pun port USB. Baud rate dan port input/output juga dapat di atur melalui *VISA configure serial port*.



Gambar 2.20 *VISA Configure Serial Port*

d. Case Structure

Case Structure merupakan program pada LabVIEW yang berfungsi sebagai pilihan (*case*). Terdapat 2 kondisi yang dapat dipergunakan yaitu *True* dan *False* (Gambar 2.21).



Gambar 2.21 *Case Structure*

e. VISA Write

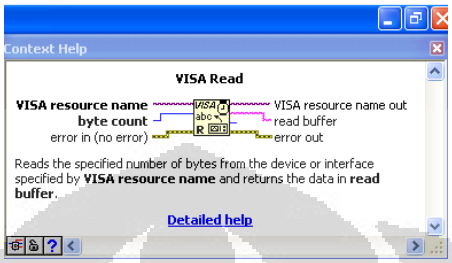
Merupakan program pada LabVIEW yang berfungsi untuk memasukan data/perintah yang hasilnya akan dibaca oleh *VISA Read.* Misalnya kita ingin membaca nilai ADC yang telah diterima oleh *VISA configure serial port,* maka kita memerlukan perintah pembacaan ADC agar bisa ditampilkan di *Visa Read,* perintah tersebut kita tuliskan di *VISA Write* di *Write Buffer* dengan format *string* (Gambar 2.22).

Gambar 2.22 *VISA Write*

f. VISA Read

Seperti yang telah dijelaskan di atas, *VISA Read* (Gambar 2.23) pada LabVIEW berfungsi untuk membaca data masukan yang telah masuk melalui

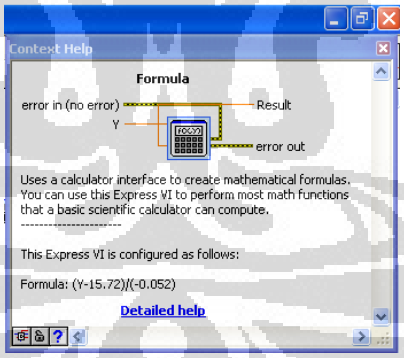
VISA configure serial port, VISA Read akan mulai membaca setelah ada instruksi dari *VISA write.* Panjang/banyak data yang diukur dapat diatur melalui *Byte Count.*



Gambar 2.23 *VISA Read*

g. *Formula*

Formula merupakan Program LabVIEW yang berfungsi untuk memasukkan sebuah persamaan matematika (Gambar 2.24). Selain membuat program menjadi lebih mudah, *Formula* juga sangat mudah untuk digunakan. Memiliki hingga delapan buah masukan dan satu buah keluaran.



Gambar 2.24 *Formula*

2.6 Teori Kontrol Proporsional Integral Diferensial(PID)

Sistem pengendali merupakan suatu sistem yang difungsikan untuk mengendalikan suatu sistem yang lain. Sistem pengendali digunakan agar kinerja suatu sistem kendali menjadi lebih baik atau pasti. Secara umum sistem pengendalian terbagi menjadi dua jenis yaitu *Open Loop Control System* dan

Closed Loop Control System. Pada sistem pengendali dikenal beberapa istilah, antara lain SP, *error*, MV, PV, dan *Plant*, yaitu adalah:

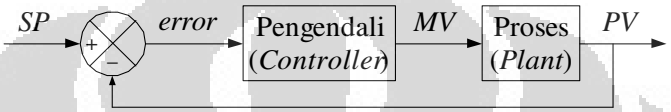
- SP (*Set Point*) adalah harga atau nilai dari keadaan yang ingin dicapai pada proses.
- *Error* adalah selisih antara *Set Point* dan *Process Variable*.
- MV (*Manipulated Variable*) adalah harga atau nilai yang diatur agar proses menjadi stabil. *Manipulated Variable* biasanya dihubungkan dengan input aktuator (contoh: *control valve*).
- PV (*Process Variable*) adalah sinyal hasil pemantauan terhadap proses atau *plant*. *Process Variable* umumnya adalah hasil pembacaan dari suatu sensor (contoh: *thermocouple*).
- *Plant* adalah objek yang akan dikendalikan (contoh: temperatur).

Open Loop Control System atau sistem pengendali *loop* terbuka merupakan sistem pengendalian dimana objek yang dikontrol tidak di-*feedback* ke pengendali, sehingga pengendali hanya akan memberikan *output* jika diberikan suatu sinyal input. Pengendali jenis ini masih bersifat manual karena tidak akan terlepas dari interfensi atau campur tangan manusia. Pengendali ini tidak akan bekerja secara otomatis, karena masih adanya interfensi manusia dan hasil dari suatu proses yang dikendalikan tidak dibandingkan oleh pengendali itu sendiri. Gambar 2.25 menggambarkan sistem pengendali *loop* terbuka (*Open Loop Control System*).

Gambar 2.25 Sistem Pengendali Loop Terbuka

Sistem pengendali yang kedua adalah *Closed Loop Control System* atau sistem pengendali *loop* tetutup, yaitu sistem pengendalian dimana objek yang dikontrol di-*feedback* ke input pengendali. Input yang diberikan ke pengendali merupakan selisih antara besaran (PV) dan besaran (SP). Nilai selisih ini sering disebut dengan *error*. Tujuan dari pengendali adalah membuat nilai *Process Variable* (PV) sama dengan nilai *Set Point* (SP), atau nilai *error* = 0. Sinyal

*error*akan diolah oleh pengendali agar nilai (PV) sama dengan nilai (SP). Pengendali jenis ini bersifat otomatis karena objek yang akan dikendalikan dibandingkan lagi dengan input keadaan yang diinginkan, sehingga interfensi manusia dapat dihilangkan. Kinerja dari suatu pengendali ditentukan oleh semakin cepatnya respon pengendali untuk mengubah MV terhadap perubahan sinyal *error*, dan semakin kecilnya kesalahan yang terjadi. Gambar 2.26 menggambarkan sistem pengendali *loop* tertutup (*Closed Loop Control System*).

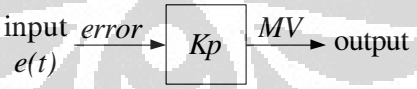


Gambar 2.26 Sistem Pengendali Loop Tertutup

Pengendali PID terdiri dari tiga macam pengendali yaitu pengendali *Proportional* (P), pengendali *Integral* (I), dan pengendali *Differential* (D). Masing-masing pengendali ini saling dikombinasikan sehingga didapatkan bentuk atau struktur dari PID, yaitu struktur paralel atau struktur *mix*. Berikut ini adalah penjelasan dari masing-masing pengendali.

a. Pengendali *Proportional* (P)

Pengendali *proportional* berfungsi untuk mengalikan sinyal input dengan suatu besaran atau konstanta dengan nilai tertentu (Gambar 2.27).

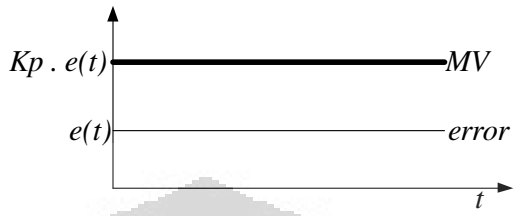


Gambar 2.27 Blok Diagram Pengendali Proporsional

Persamaan 2.7 memperlihatkan hubungan antara *input* (*error*) dan *output* (*MV*) pada pengendali:

$$MV = Kp \cdot e(t) \tag{2.7}$$

Karena pengendali *proportional* hanya menguatkan sinyal input saja, maka hubungan antara sinyal *error* dan sinyal *MV* dapat digambarkan seperti grafik respon pada Gambar 2.28.



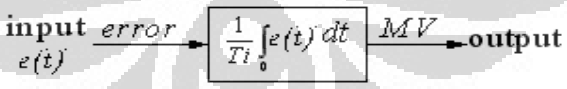
Gambar 2.28 Grafik Respon Pengendali Proporsional

Pengendali *proportional* berfungsi untuk mempercepat proses yang dikendalikan menuju ke keadaan *set-point*. Kecepatan proses ini sangat bergantung dari besarnya nilai Kp pada pengendali *proportional*.

Semakin besar nilai Kp maka semakin besar juga penguatannya sehingga respon dari pengendali akan semakin cepat juga dan akan mengurangi besarnya *steady-state error*. Tetapi jika nilai Kp terlalu besar maka sistem akan mengalami *over shoot* yang besar sehingga proses yang dikendalikan menjadi tidak stabil bahkan akan mengalami osilasi.

b. Pengendali *Integral* (I)

Pengendali *integral* berfungsi untuk meng-*integral*-kan sinyal input lalu dibagi dengan suatu besaran atau konstanta dengan nilai tertentu (Gambar 2.29).

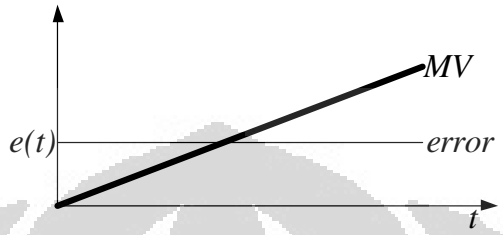


Gambar 2.29 Blok Diagram Pengendali Integral

Persamaan 2.8 memperlihatkan hubungan antara input (*error*) dan output (*MV*) pada pengendali:

$$MV = \frac{1}{Ti}\int_0^t e(t)\,dt \qquad (2.8)$$

Karena pengendali *integral* hanya meng-*integral*-kan sinyal input saja, maka hubungan antara sinyal *error* dan sinyal *MV* dapat digambarkan seperti grafik respon pada Gambar 2.30.



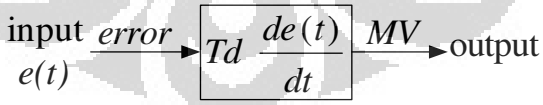
Gambar 2.30 Grafik Respon Pengendali Integral

Pengendali *integral* berfungsi untuk mengurangi dan menghilangkan *steady-state error* yang timbul setelah respon *plant* dari pengendali *proportional* sudah stabil. Semakin kecil nilai *steady-state error*, maka respon dari *plant* akan semakin mendekati keadaan *steady-state*. Semakin kecil nilai *error* maka semakin kecil juga nilai *timing integral*-nya, sehingga kurva MV akan semakin landai.

Pengendali *integral* sangat optimal bekerja pada daerah di sekitar titik *set-point*, yaitu antara *steady-state error* dan *set point*.

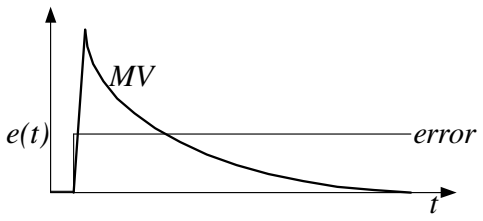
c. Pengendali *Differential* (D)

Pengendali *differential* berfungsi untuk men-*differential*-kan sinyal input lalu dikalikan dengan suatu besaran atau konstanta dengan nilai tertentu (Gambar 2.31).



Gambar 2.31 Blok Diagram Pengendali Diferensial

Persamaan hubungan antara input (*error*) dan output (*MV*) pada pengendali ini adalah karena pengendali *differensial* hanya meng- *differensial*-kan sinyal input saja, maka hubungan antara sinyal *error* dan sinyal *MV* dapat digambarkan seperti grafik respon pada Gambar 2.32.

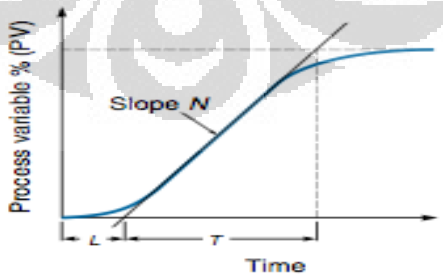


Gambar 2.32 Grafik Respon Pengendali Diferensial

Pengendali *differential* berfungsi untuk mengurangi respon yang terlalu berlebih yang dapat mengakibatkan *over shoot* pada proses *plant* karena nilai Kp yang terlalu besar pada pengendali *proportional*. Output dari pengendali *differential* akan bernilai sangat besar jika perubahan *error* sangat besar. Perubahan *error* yang sangat besar ini terjadi ketika proses *plant* bergerak menuju ke titik *set-point* dalam waktu yang sangat singkat (nilai *dt* sangat kecil). Hal ini disebabkan karena respon pengendali yang terlalu cepat akibat terlalu besarnya nilai *Kp* pada pengendali *proportional*. Pengendali *differential* hanya akan bekerja ketika terjadi perubahan *error*, sehingga ketika proses yang dikendalikan sudah stabil maka pengendali *differential* sudah tidak bekerja lagi.

2.6.1 Teori Dasar Kurva Reaksi dengan Metode Ziegler- Nichols

Metode kurva reaksi adalah salah satu cara pengendalian parameter PID. Metode ini tidak membutuhkan sistem untuk berosilasi. Sebagai alternatif, *loop* pada *feedback* terbuka dan pengendalian terhadap *output* dilakukan manual secara langsung (Gambar 2.33).



Gambar 2.33 Contoh Respon Pengendalian Terhadap Waktu

Dengan :

L = *Lag Time*

T = *Rise time*

Dalam melakukan pengendalian dengan kurva reaksi metode Ziegler-Nichols dapat dilakukan melalui beberapa langkah, yaitu:

- mencari persamaan tangen pada bagian *rising* dari kurva respon. Garis ini akan menghasilkan nilai L (*delay*) dan T (*rise time*). L merupakan *delay* waktu antara *output* pengendali dengan respon yang dikendalikan.
- Menghitung *slope* dari kurva dengan cara:

  $N = S / \Delta$ (2.9)

  Dimana :

  N = Kemiringan garis singgung yang melalui titik infleksi/ perubahan

  ΔP = Besarnya perubahan dalam pengendali *output*
- Menghitung Konstanta PID:

  Kc = 1.2 / L N

  Ti = 2 L

  Td = 0.5 L

2.6.1.1 Keuntungan Metode Ziegler – Nichols (Z - N)

1. Hanya membutuhkan 1 kali eksperimen.
2. Tidak membutuhkan prosedur *trial* dan *error*.
3. Pengaturan kontrolnya lebih mudah diperhitungkan.

2.6.1.2 Kerugian Metode Ziegler – Nichols (Z - N)

1. Eksperimennya dijalankan berdasarkan kondisi *open- loop*, gangguan yang terjadi harus dieliminasi selama eksperimen.
2. Titik perubahannya tidak mudah didapat jika pengukuran mengalami gangguan dan grafik pencatatannya tidak cukup besar.
3. Metode ini cenderung sensitif terhadap kesalahan kalibrasi *controller*. Sedangkan metode Z - N tidak begitu sensitif terhadap kesalahan kalibrasi karena daya *controller* disesuaikan selama eksperimen.
4. Metode ini tidak direkomendasikan untuk proses yang memiliki respon osilasi.

2.7 Roda Gigi (*Gear*)

Roda gigi adalah bagian dari mesin yang berputar yang berguna untuk mentransmisikan daya. Roda gigi memiliki gigi-gigi yang saling bersinggungan dengan gigi dari roda gigi yang lain. Dua atau lebih roda gigi yang bersinggungan dan bekerja bersama-sama disebut sebagai transmisi roda gigi, dan bisa menghasilkan keuntungan mekanis melalui rasio jumlah gigi. Roda gigi mampu mengubah kecepatan putar, torsi, dan arah daya terhadap sumber daya.

2.7.1 Jenis – Jenis Roda Gigi

Jenis-jenis Roda gigi dapat dibedakan pula dari keadaan konstruksi alur bentuk gigi serta berdasarkan bentuk serta fungsi konstruksinya.

2.7.1.1 Roda Gigi Lurus

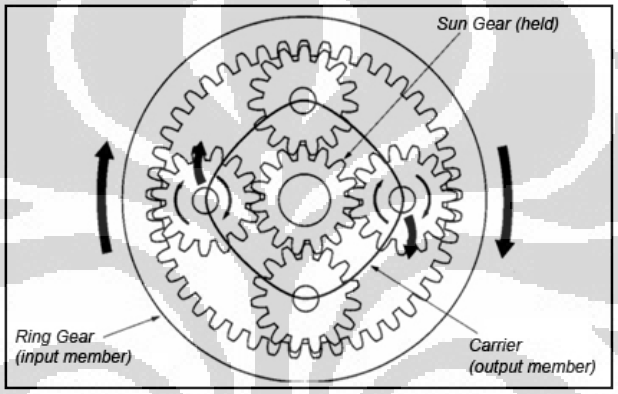
Adalah roda gigi dengan bentuk profil gigi beralur lurus dengan kondisi penggunaan untuk sumbu sejajar. Pada konstruksi berpasangan, penggunaannya terdapat dalam tiga keadaan, yaitu :

a. Roda Gigi lurus eksternal (*spur gear*)
b. Roda Gigi lurus internal (*planetary gear*)
c. Roda Gigi lurus *rack* dan *pinion*.

Penggunaan roda gigi lurus ini cukup luas terutama *spur gear* pada konstruksi general mekanik yang sederhana sampai sedang putaran dan beban relatif sedang. Dari ketiga jenis roda gigi ini, rnaka *internal gear* memiliki tingkat kesulitan pemasangan yang agak sulit, sehubungan dalam menentukan ketepatan pemasangan sumbu. Sedangkan untuk jenis *rack* dan *pinion gear*, mempunyai kekhususan dalam penggunaannya, yaitu untuk pengubah gerak putar ke gerak lurus atau sebaliknya, sedangkan pada *rack gear* mempunyai sumbu *pitch* yang lurus. Pembebanan pada gigi-giginya mempunyai distribusi beban yang paling sederhana, yaitu gaya normal yang terurai menjadi gaya keliling (gaya tangensial) dan gaya radial.

Planetary gear unit dipakai untuk menaikan dan menurunkan momen mesin, menaikan dan menurunkan kecepatan kendaraan, di pakai untuk memundurkan kendaraan dan dipakai untuk bergerak maju. Pada dasarnya *planetary gearunit* dipakai mesin untuk menghasilkan tenaga dan menggerakan kendaraan dengan beban yang berat dengan tenaga yang ringan. *Planetary*

*gear*memiliki tiga tipe gigi cincin, gigi *pinion*, *sun gear* dan *planetary carrier*(Gambar 2.34).*Planetary carrier* dihubungkan dengan poros tengah tiap gigi *pinion* dan membuat gigi *pinion* berputar. Gigi-gigi pada *planetary carrier* berhubungan satu sama lainnya.Gigi *pinion* mempunyai prinsip kerja menyerupai planet yang berputar di sekeliling matahari. Oleh karena itu, disebut *planetary carrier*. Biasanya, *planetary carrier* dikombinasikan dalam unit *planetary carrier*.Penggantian *input* pada *planetary carrier*, *output*, dan elemen tetap, memungkinkan untuk deselerasi, mundur, hubungan langsung dan akselerasi.



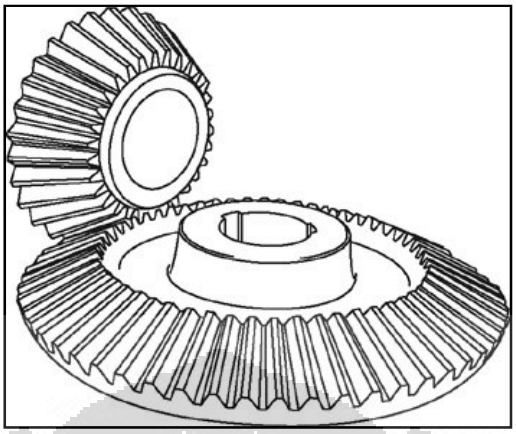
Gambar 2.34 *Planetary Gear Unit*

2.7.1.2 Roda Gigi Payung

Roda gigi payung sering disebut juga roda gigi kerucut atau *bevel gear*. Penggunaannya secara umum untuk pentransmisian putaran dan beban dengan posisi sumbu menyudut berpotongan dimana kebanyakan bersudut 90^0. Khusus jenis roda gigi payung hypoid, posisi sumbunya bersilangan. Pada pemasangan roda gigi payung umumnya salah satu dipasang dengan konstruksi tumpuan melayang, terutama pada roda gigi penggerak. Dari bentuk serta arah alur giginya, terdapat beberapa jenis roda gigi payung, diantaranya :

a. Roda Gigi Payung Gigi Lurus

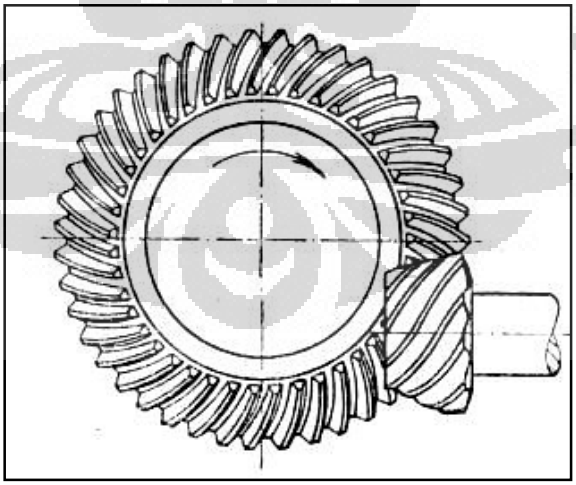
   Untuk jenis ini mempunyai konstruksi yang sederhana dibanding jenis roda gigi payung lainnya. Pembuatannya relatif mudah dan penggunaannya untuk konstruksi umum yang sederhana sampai sedang, baik dalam menerima beban maupun putaran (Gambar 2.35).



Gambar 2.35 Roda Gigi Payung Gigi Lurus

b. Roda Gigi Payung Gigi Miring

Disebut juga *spiral bevel gear*. Perbedaan antara bentuk gigi lurus dengan bentuk gigi miring pada roda gigi payung ini, kurang lebih seperti perbedaan yang terdapat pada roda gigi lurus (*spur gear*) dengan roda gigi miring, dimana dengan adanya kemiringan tersebut akan meningkatkan kemampuan menerima beban, mengurangi kebisingan, sehingga dapat digunakan pada putaran yang lebih tinggi dibanding dengan roda gigi payung gigi lurus pada ukuran geometris yang sama (Gambar 2.36).



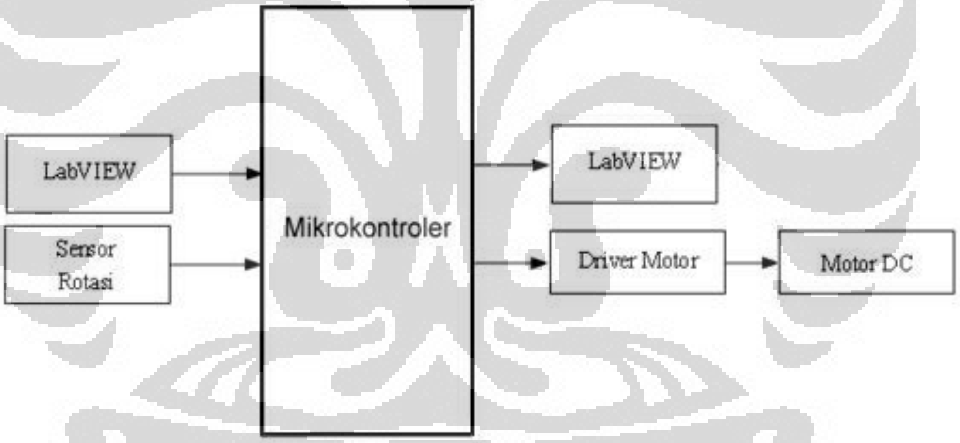
Gambar 2.36 Roda Gigi Payung Gigi Miring

BAB 3
PERANCANGAN DAN CARA KERJA SISTEM

Pada bab ini akan dibahas mengenai perancangan sistem dan cara kerja dari masing-masing *hardware* dan *software* yang digunakan penulis dalam pembuatan alat "Rancang Bangun Prototipe Sistem Aktuator Kendali Sirip Menggunakan LabVIEW".

3.1 Perancangan Kerja Sistem

Untuk mempermudah dalam perancangan dan pemahaman system rangkaian, maka perancangan dibuat berdasarkan blok. Dimana tiap blok mempunyai fungsi dan kerja tertentu, blok diagram yang satu dengan yang lain berhubungan dan saling mendukung hingga terbentuk suatu sistem rangkaian yang mempunyai satu fungsi dan kerja khusus. Diagram blok selengkapnya ditampilkan pada Gambar 3.1.



Gambar 3.1 *Block Diagram* Cara Kerja Alat

Dari blok diagram diatas terdapat LabVIEW sebagai data *tranceiver* untuk mengirim nilai *Set Point* (SP) yang berupa bilangan ke dalam *microcontroller* dan menampilkan nilai *Process Variable* (PV) pada LabVIEW lagi. Proses pengendalian arah gerak motor akan terjadi di *microcontroller*. Kemudian motor dan sensor rotasi akan bekerja sesuai dengan nilai *Set Point* yang telah diatur. Hasil pembacaan dari sensor akan dikirimkan ke *microcontroller*. Dan kemudian ditampilkan pada LabVIEW.

Pada dasarnya, alat ini dikendalikan oleh sistem pengendali yaitu pengendali kecepatan motor (Persamaan 3.1 dan 3.2).

$$E = SP - PV \quad (3.1)$$

Dengan:

E = *Error*

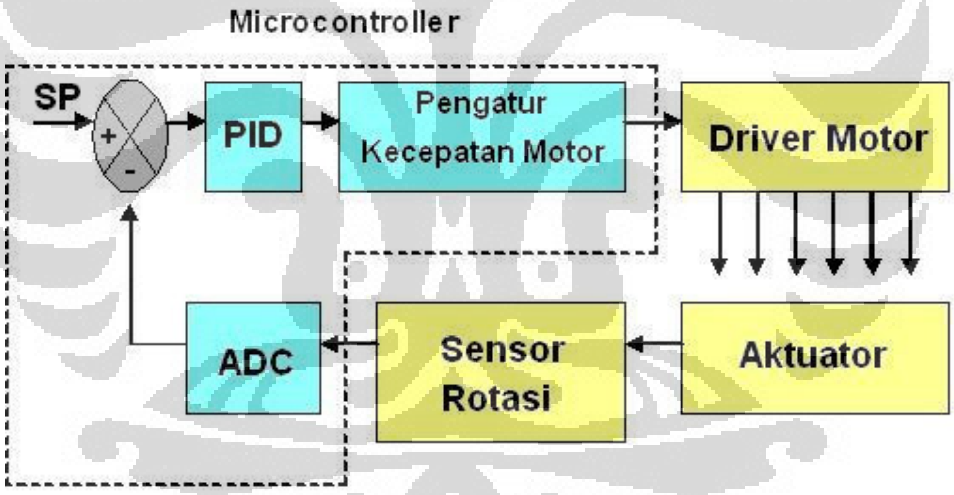
SP = *Set Point*

PV = *Process Variable*

Lalu ke dalam PID yang akan dihitung nilai *Manipulated Variable* (MV) dengan Persamaan 3.2

$$MV = Kp\left(E + Ki\int Edt + Kd\frac{dE}{dt}\right) \quad (3.2)$$

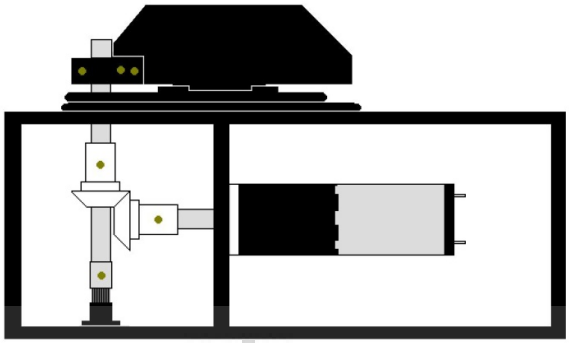
Nilai MV tersebut akan mengatur kecepatan putaran motor agar kecepatan motor dapat dikendalikan ketika motor hampir mencapai *set point* (Gambar 3.2).



Gambar 3.2 *Block Diagram* Pengendalian Kecepatan

3.2 Rancang Bangun Prototipe

Rancang bangun prototipe sistem kendali sirip roket berbasis LabVIEW ini menggunakan material plat besi setebal 1 milimeter. Digunakan plat besi agar rancangan kokoh dan mengurangi gangguan luar, seperti tersenggol yang dapat merubah posisi dari sirip.



Gambar 3.3 Prototipe Sirip Roket

Dibagi oleh tiga bagian yaitu, ruang motor DC, ruang *gear* dan sensor rotasi, dan yang paling atas adalah tempat simulasi pergerakan sirip yang disertai dengan *pitch gauge* (Gambar 3.3).

3.3 *Pitch Gauge*

Pitch gauge merupakan alat ukur yang berfungsi untuk menghitung besar sudut yang didapat oleh sebuah benda (Gambar 3.4). *Pitch gauge* yang kami pakai adalah *pitch gauge* yang memiliki ketelitian sebesar 1 derajat. *Pitch gauge* ini mampu mengukur besar sudut dari 10 derajat sampai -10 derajat. *Pitch gauge* akan dihubungkan langsung dengan sirip roket untuk mengetahui besar sudut yang diperoleh oleh sirip roket.

Gambar 3.4 *Pitch Gauge*

3.4 Motor DC dan *Planetary Gear*

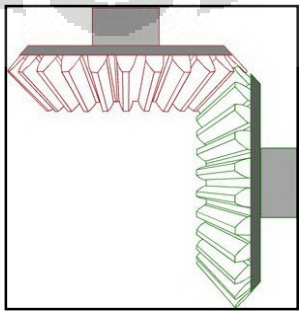
Motor DC menggunakan tipe maxon 242472 jenis *Brushed*. Motor ini memiliki tegangan nominal 24 volt, daya yang dihasilkan 15 watt, kecepatan 5930 rpm, dan torsi 3.67 Ncm.

Planetary gear yang digunakan juga merupakanproduksi Maxon dengan nomor seri manufaktur 110375 yang mempunyai diameter 32 mm, panjang badan 49.8 mm, panjang shaft 21 mm, diameter shaft 5.5 mm, reduksi kecepatan 246:1. *Gear* ini digunakan karena mempunyai torsi yang besar yaitu 6.75 Nm dan mampu mereduksi kecepatan putaran motor dari 246 putaran menjadi 1 putaran (Gambar 3.5) .

Gambar 3.5 Motor DC dan *Planetary Gear*

3.5 Roda Gigi

Untuk menghubungkan antara motor DC dan sirip roket, maka diperlukan sebuah sistem roda gigiyang mampu mentransmisikan dua buah poros yang saling berpotongan. Digunakan jenis roda gigi kerucut (*Bevel*) dengan perbandingan 1:1 (Gambar 3.6).

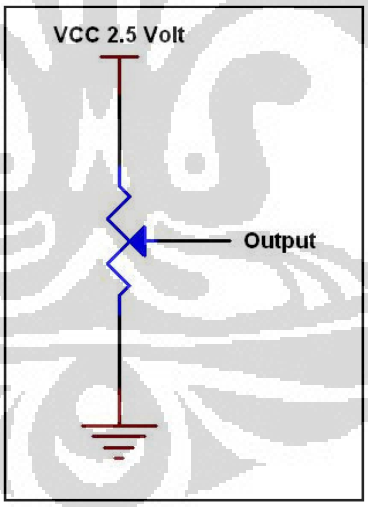


Gambar 3.6 Roda Gigi Kerucut

3.6 Sensor Rotasi

Sensor rotasi yang kami gunakan pada penelitian kali ini adalah sensor rotasi dengan sudut putar maksimum sebesar 300^0. Sensor rotasi ini diletakan di bawah sirip untuk memantau pergerakan sirip, apakah sirip bergerak sudah sesuai dengan masukan (*input*) atau tidak. Sensor ini memiliki tiga buah pin konektor, konektor pertama berfungsi sebagai keluaran (*output*), konektor yang kedua berfungsi sebagai pertanahan (*grounding*), dan konektor yang ketiga merupakan konektor sumber tegangan (Gambar 3.7). Keluaran (*Output*) dari sensor ini berupa tegangan yang nantinya akan dikonversi menjadi data digital oleh ADC. Untuk mendapatkan ketelitian sudut yang tinggi pada sensor rotasi, kami menggunakan tegangan masukan sebesar 2.5 volt. Dari tegangan tersebut dapat diperoleh nilai resolusi sebesar: 2.44 (Persamaan 3.3)

$$\text{Resolusi} = \left(\frac{\text{Tegangan Referensi}}{2^n - 1} \right) = \frac{2{,}5 \text{ volt}}{1023} = 2.44 \text{ mV} \quad (3.3)$$



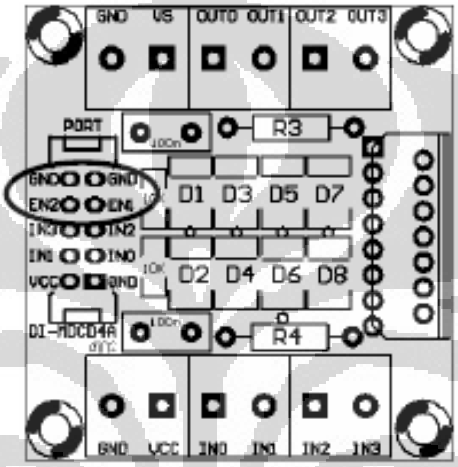
Gambar 3.7 Sensor Rotasi

3.7 Penggerak Motor

Penggerak motor (*driver motor*) yang kami gunakan adalah tipe DI-M.D.C.D.4.A *series* (Gambar 3.8). Merupakan salah satu produk dari Depok Instrumen atau biasa disingkat DI.Penggerak motor iniadalah seri modul Motor

DC *Driver* yang digunakan dalam suatu sistem untuk menguatkan arus dan atau tegangan keluaran (*output*) pengendali (misalkan mikrokontroler) agar pengendali dapat mengendalikan Motor DC.

Pengaplikasian penggerak motor ini bisa digunakan untuk rangkaian penguat untuk mengendalikan motor-motor berarus DC (*Direct Current*) sepert Motor DC, *Gearbox, Motor Stepper, Motor DC Tape*, dll.



Gambar 3.8 Penggerak Motor (*Driver Motor*)

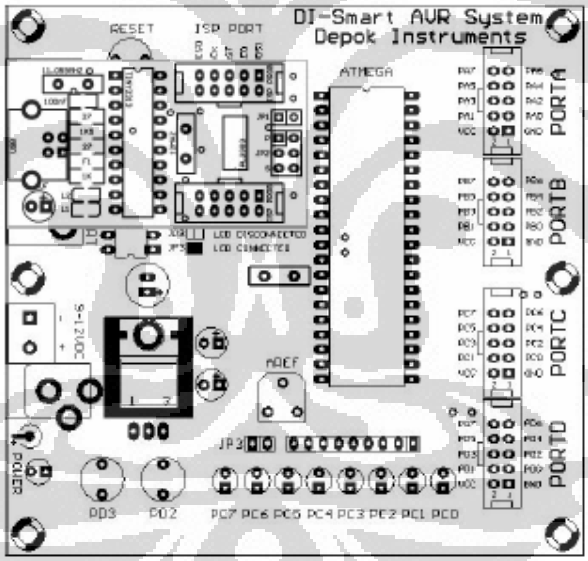
Spresifikasi dari penggerak motor (*driver motor*) ini adalah sebagai berikut:

1. Menggunakan komponen penguat *dual full bridge drive*:
    a. Tegangan *supply* operasi sampai dengan 46 VDC.
    b. Total arus DC yang mampu dilewatkan sampai dengan 4 Ampere.
    c. Terdiri dari 2 bagian yang independen.
2. Memiliki 4 output (dapat dikoneksikan dengan 1 *Motor Stepper*, atau 2 Motor DC 2 arah, atau 4 Motor DC 1 arah).
3. Dapat langsung dihubungkan pada *DI-Smart AVR System*atau *DI-Smart 51 System*atau *DI-Basic AVR System*:
    a. Pin-pin kendali berada di posisi D0-D3.
    b. *Enable* IC ada di posisi D4 dan D5. *Default* dari pin ini adalah *high* atau IC dalam kondisi *Enabled.*

c. Pin D6 dan D7 dihubungkan ke GND, untuk me-non-aktifkan (Disable) IC, yaitu dengan menghubungkan pin EN1 atau EN2 dengan GND yang berdampingan (lihat bagian yang dilingkari pada Gambar 3.8).

3.8 Mikrokontroler

DI-Super Smart AVRadalah sebuah modul elektronika yang berdasar pada rangkaian sistem minimum mikrokontroler AVR (sismin AVR) Atmega8535 yang telah dilengkapi dengan modul *downloader* yang juga dapat berfungsi sebagai antarmuka komunikasinya dengan komputer melalui PORT USB. Modul ini dibangun dari dua modul Depok Instrumen, yaitu *DI-Smart AVR System* dan DI-USB AVR ISP V2 (Gambar 3.9).



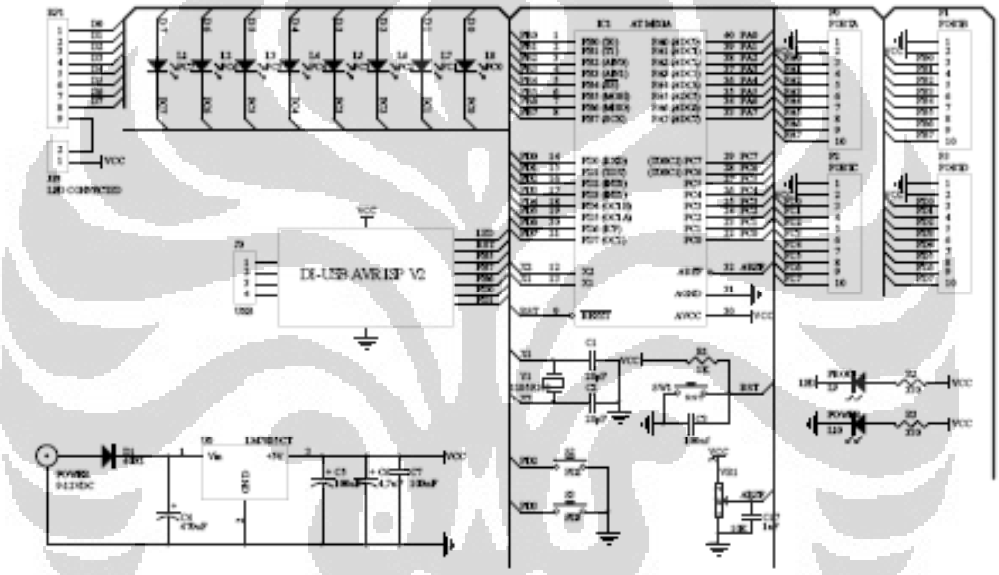
Gambar 3.9 Mikrokontroler

Peran dari mikrokontroler sendiri adalah sebagai CPU (*Central Processing Unit*) atau pengendali dalam berbagai macam sistem, baik itu sistem instrumentasi, sistem robotika, dan otomasi-otomasi yang lainnya. Spesifikasi dari mikrokontroler ini adalah sebagai berikut:

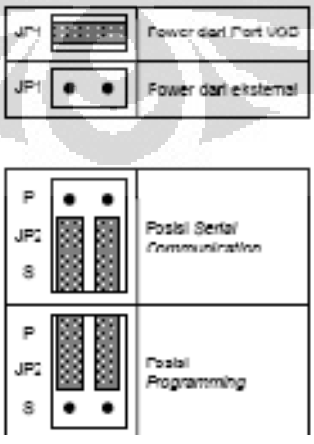
- ✓ Dapat digunakan untuk jenis AVR ATmega8535(L), ATmega16(L), ATmega32(L), ATmega163(L), ATmega323(L).

- ✓ Memiliki USB *converte*r sehingga dapat langsung dihubungkan pada PORT USB komputer (Gambar 3.11).
- ✓ Koneksi ADC sudah disiapkan (AVCC, AGND, dan AREF) sehingga sistem sudah siap untuk menerima *input* analog pada PORTA.
- ✓ Tersedia *Array* LED pada PORTC, dan Push-ON pada PORTD.2 dan PORTD.3 sehingga cocok untuk latihan atau pengecekkan program.

Berikut merupakan gambar dari skematik mikrokontroler AVR ATmega8538 beserta perangkat pendukungnya (Gambar 3.10).



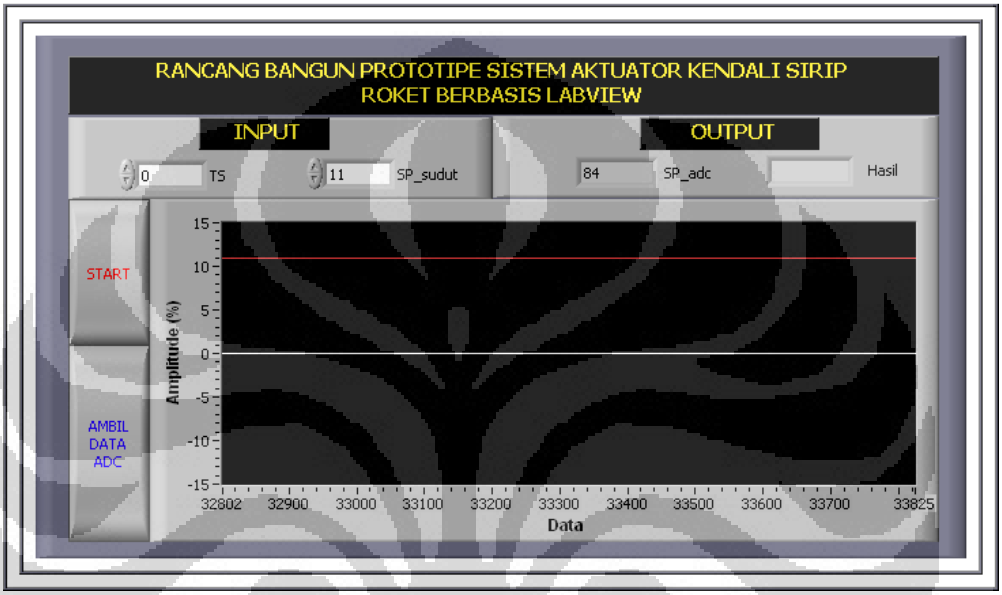
Gambar 3.10 Skematik Rangkaian Mikrokontroler



Gambar 3.11 Pengaturan *Jumper*

3.9 LabVIEW

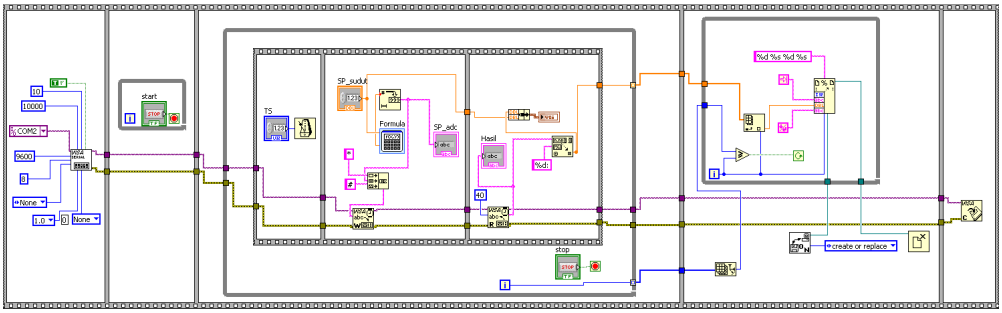
LabVIEW disini berfungsi sebagai masukan dan tampilan, atau bisa juga dibilang sebagai *output*. Untuk mengatur besarnnya sudut yang ingin digerakkan, kami menggunakan *control - string* sebagai masukan sudut yang diatur agar dapat berputar sebesar 10 derajat hingga -10 derajat (Gambar 3.12).



Gambar 3.12 *Front Panel*

Selain itu, terdapat juga tombol *start* yang berfungsi untuk memulai jalannya program LabVIEW dan tombol Ambil Data ADC untuk mengakhiri jalannya program LabVIEW. Nilai ADC yang didapat, kami tampilkan melalui *VISA read* yang sebelumnya telah dikonversi menjadi angka sehingga dapat dibaca. Pergerakan sirip akan divisualisasikan melalui *waveform chart. waveform chart* akan menampilkan nilai *set point* yang di beri warna garis merah, sedangkan untuk pergerakan sirip, akan ditampilkan oleh garis berwarna putih.

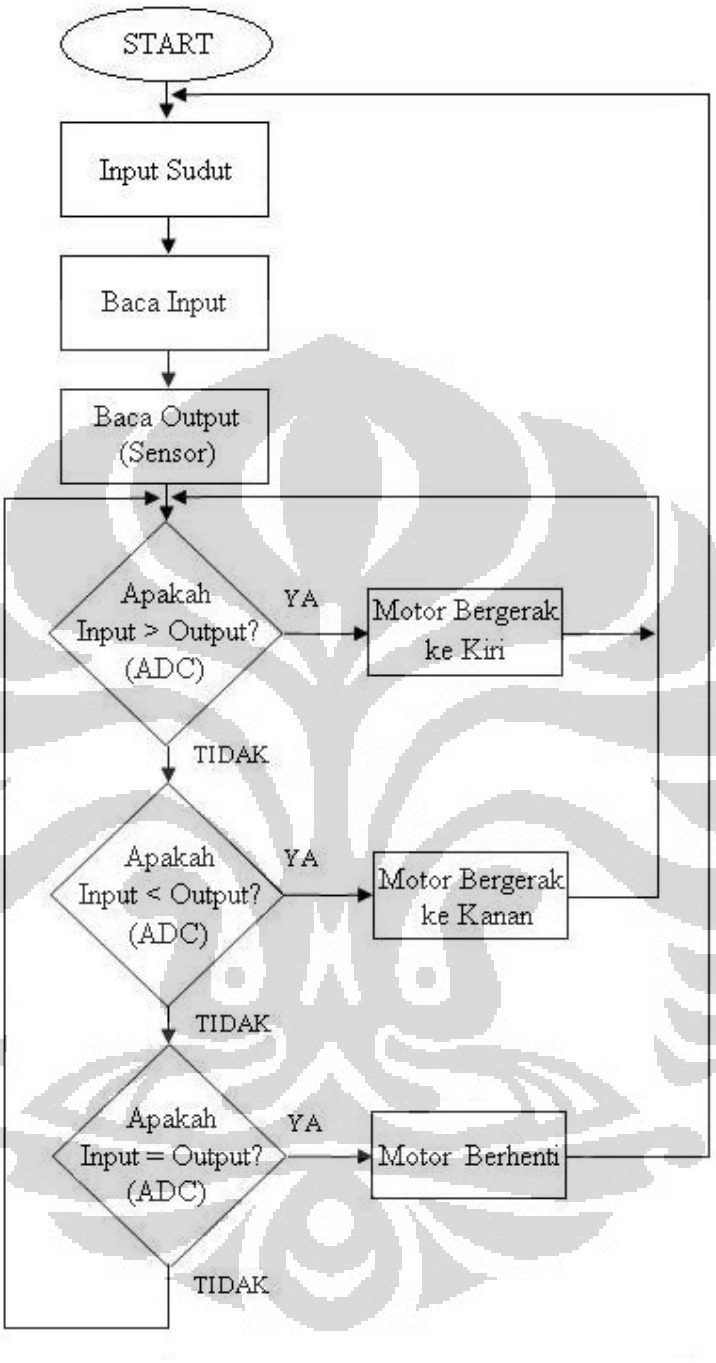
Pergerakan sirip atau garis yang berwarna putih akan bergerak mengikuti nilai *set point* yang telah dimasukan melalui *control-string* atau garis yang berwarna merah. Garis putih akan berhenti jika sudah sama posisinya dengan garis merah, atau dengan kata lain pergerakan sirip roket telah sesuai dengan nilai *set point* yang telah kita inputkan sebelumnya.



Gambar 3.13 *Block Diagram*

Bisa kita liat pada blok diagram di atas (Gambar 3.13), program bermula dari sebelah kiri dan berakhir di sebelah kanan. Mula – mula *VISA read* membaca data ADC dari mikrokontroler melalui komunikasi serial USB. Setelah itu, ADC masuk ke *VISA read* dan kemudian ditampilkan pada *front pan*el. Setelah ditampilkan, nilai ADC kemudian dibandingkan, apakah nilai ADC yang terbaca dengan nilai sudut yang telah dikonversi menjadi ADC oleh *formula* sudah sama atau belum. Jika belum, maka motor akan terus bergerak hingga set poin yang telah ditentukan melalui *control-string* yang tersedia di *front panel.* Jika nilai ADC yang terbaca, dengan nilai ADC pada set poin telah sama, maka motor akan berhenti bergerak.

Untuk mempermudah pendesainan program maka penulis membuat *flowchart* terlebih dahulu sebelum membuat program pada chip ataupun pada LabVIEW. *Flowchart* di desain mengikuti tahap – tahap sistem yang telah dirancang sebelumnya. Tiap bagian *Flowchart* memiliki fungsinya masing – masing, bentuk bulat dan lonjong adalah awal dan akhir dari suatu proses. Persegi panjang berfungsi untuk memberi sebuah perintah atau menyatakan sebuah kondisi. Sedangkan yang berbentuk belah ketupat berfungsi untuk membuat sebuah pertanyaan. Adapun *flowchart* yang dibuat penulis sebagai berikut (Gambar 3.14).



Gambar 3.14 *Flow Chart*

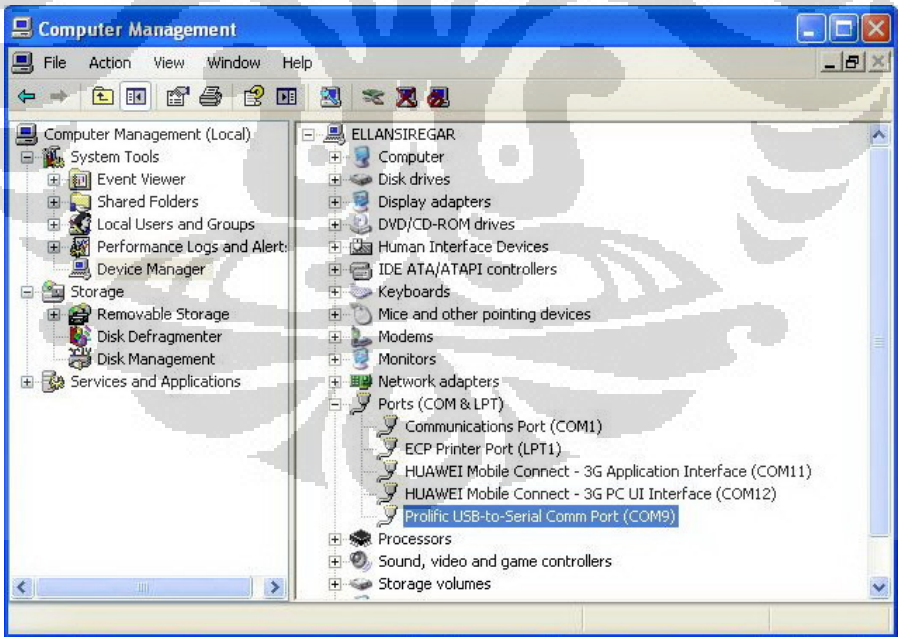
Dengan *flow chart* di atas, maka dapat mempermudah penulis dalam pengerjaan program pada ATmega8535 dan pemrograman berupa blok diagram pada LabVIEW.

BAB 4
PENGUJIAN ALAT DAN ANALISA HASIL

Setelah keseluruhan sistem dibuat, maka perlu dilakukan uji coba dan analisa sistem, apakah sistem dapat bekerja dengan baik dan benar. Bab ini akan membahas pengujian ADC, komunikasi serial, program pengendali dan respon motorbeserta komponen-komponen pendukung dari sistem pengendali sirip roket.Pengujian dan pengukuran dilakukan terhadap komponen yang penulis merasa perlu untuk dilakukan.

4.1 Komunikasi serial

Pada pengujian kamunikasi serial, ada beberapa hal yang perlu di perhatikan, yang pertama adalah port komunikasi yang akan digunakan. Cek port yang digunakan di *computer management* (Gambar 4.1) pada komputer atau laptop yang kita gunakan.



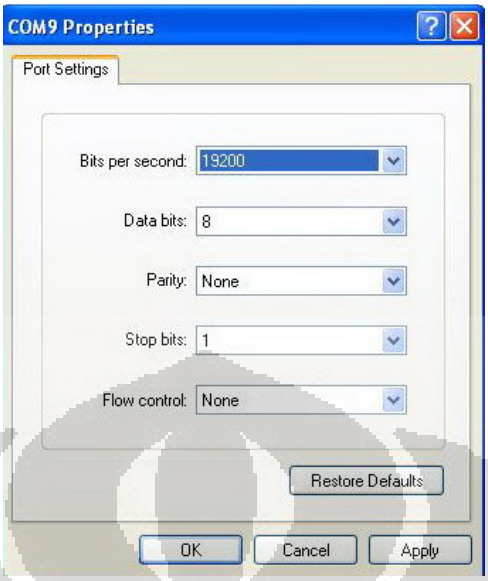
Gambar 4.1 Pengaturan Komunikasi USB-Serial Pada *Computer Management*

Port yang kami gunakan pada USB-serial komunikasi ini adalah port COM9, yang merupakan port USB pada komputer.

Langkah yang kedua adalah mengatur komunikasi pada program *hyper terminal* yang telah tersedia pada komputer dengan sistem operasi windows XP, untuk windows vista dan windows 7, program *hyper terminal* harus di unduh terlebih dahulu. Pastikan koneksi menggunakan port COM9 (Gambar 4.2) seperti pada *computer management.*

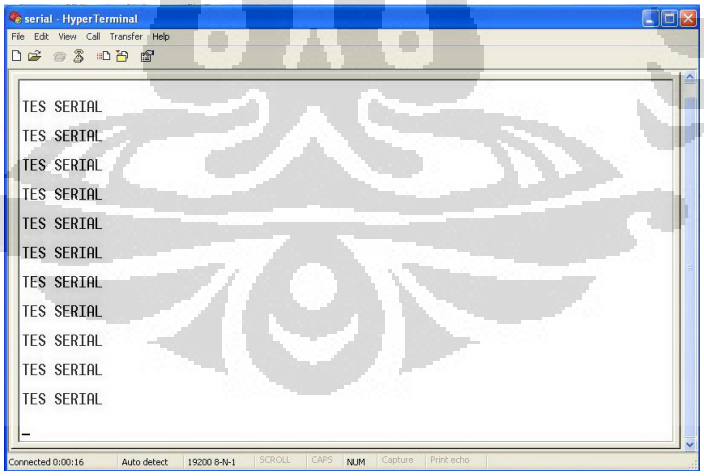
Gambar 4.2 Pemilihan Koneksi Pada *Hyper Terminal*

Setelah klik ok, maka akan muncul menu seperti yang di tunjukkan pada Gambar 4.3, yaitu menu untuk mengatur port. Ubah nilai *bits per second* pada *port setting* menjadi 19200 (Gambar 4.3). 19200 merupakan nilai *baud rate* yang digunakan untuk komunikasi serial dari mikrokontroler ke komputer atau laptop. Nilai *baud rate* bebas di pilih, tapi dengan syarat harus sinkron dengan *baud rate* pada mikrokontroller. *Data bits, Parity, Stop bits,* dan *Flow control* tidak perlu dirubah, biarkan saja mengikuti *default setting* dari pengaturan port. Selanjutnya klik ok untuk ke tahap berikutnya.



Gambar 4.3 Pengaturan Koneksi Pada *Hyper Terminal*

Setelah mengatur port pada aplikasi *hyper terminal,* selanjutnya program siap digunakan, cara mengaktifkan komunikasi tergantung dari program yang tertulis di mikrokontroler. Setelah aktif maka bisa dilihat komunikasi antara mikrokontroler dan komputer telah berhasil (Gambar 4.4).

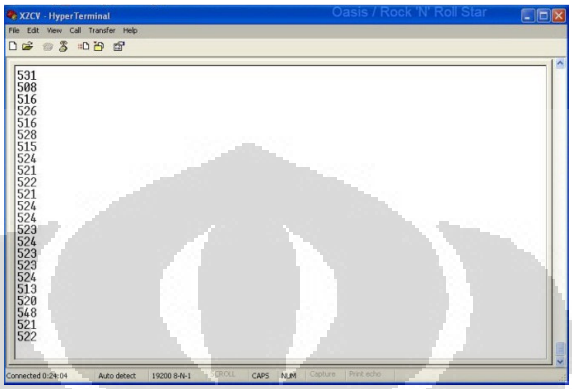


Gambar 4.4 Tampilan Komunikasi Program *Hyper Terminal*

4.2 ADC (*Analog to Digital Converter*)

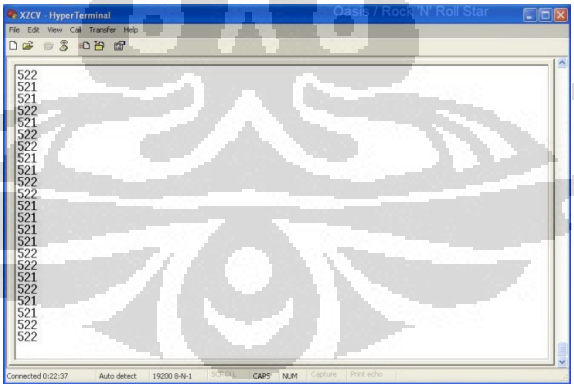
Pada pengendali sirip roket ini, ADC digunakan sebagai pengubah besaran analog menjadi besaran digital pada tegangan keluaran sensor rotasi. Pada awal

pengambilan data sensor, nilai ADC yang di dapat sangat fluktuatif (Gambar 4.5), ini diakibatkan oleh tegangan keluaran sensor yang tidak stabil. Untuk mengatasi tidak stabilnya nilai ADC yang di dapatkan, kami menggunakan kapasitor 100µF.



Gambar 4.5 Nilai ADC Sebelum Diberi Kapasitor

Kapasitor ini berfungsi sebagai *filter ripple*, disini sifat dasar kapasitor yaitu dapat menyimpan muatan listrik yang berfungsi untuk memotong tegangan ripple, sehingga di dapatkan nilai ADC yang cukup stabil (Gambar 4.6), yang membantu pengambilan data sensor menjadi lebih baik dari sebelumnya.



Gambar 4.6 Nilai ADC Setelah Diberi Kapasitor

Dengan nilai tegangan referensi sebesar 2.5 volt, berikut merupakan data ADC yang di dapat pada Tabel 4.1.

Tabel 4.1 Nilai ADC Terhadap Sudut

Sudut	Nilai ADC	Nilai Tegangan
10^0	101	0.97 Volt
9^0	123	1.02 Volt
8^0	145	1.08 Volt
7^0	161	1.13 Volt
6^0	180	1.17 Volt
5^0	195	1.20 Volt
4^0	212	1.23 Volt
3^0	135	1.28 Volt
2^0	248	1.32 Volt
1^0	272	1.38 Volt
0^0	291	1.42 Volt
-1^0	303	1.46 Volt
-2^0	320	1.50 Volt
-3^0	340	1.56 Volt
-4^0	364	1.64 Volt
-5^0	380	1.69 Volt
-6^0	400	1.72 Volt
-7^0	422	1.76 Volt
-8^0	444	1.79 Volt
-9^0	466	1.87 Volt
-10^0	478	1.92 Volt

Dari data di atas di dapat persamaan sebagai berikut:

$$y = -0.053x + 15.44$$

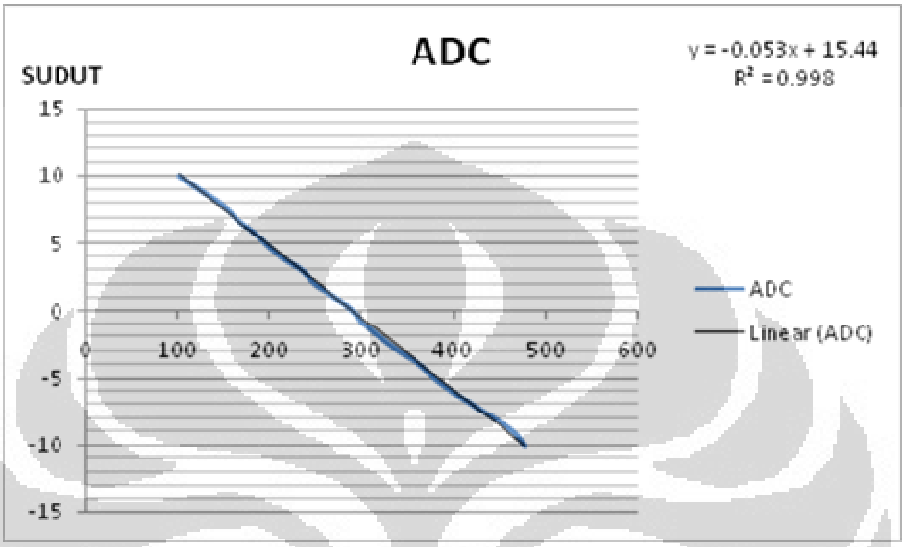
$$x = \frac{y - 15.44}{-0.053} \quad (4.1)$$

Dimana:

x = Nilai ADC

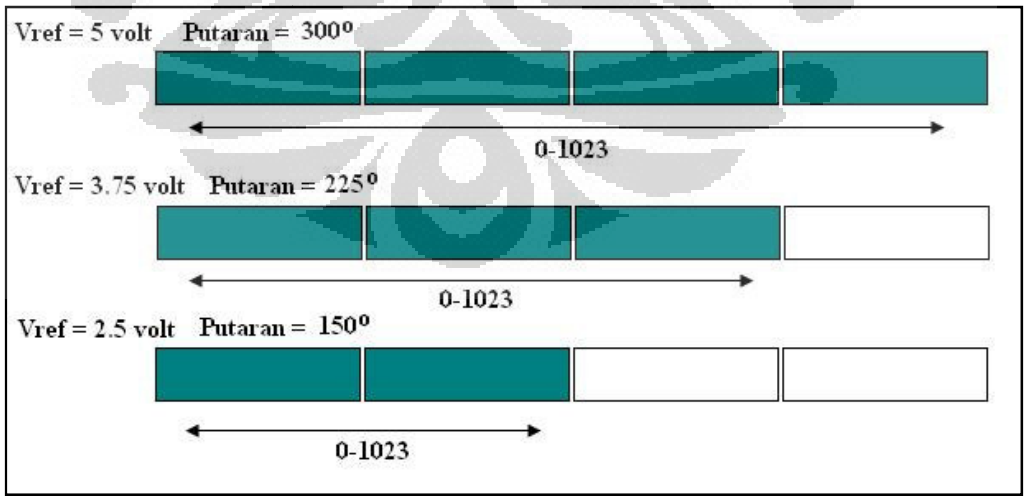
y = Sudut

Dari persamaan diatas (Persamaan 4.1), maka akan dimasukan ke dalam formula pada LabVIEW untuk mendapatkan nilai sudut dari hasil perhitungan (Gambar 4.7).



Gambar 4.7 Nilai ADC Terhadap Sudut

Pada sensor, kami menggunakan tegangan referensi sebesar 2.5 volt. Tegangan ini diatur melalui potensiometer yang tersedia di mikrokontroler. Fungsi tegangan ini adalah sebagai tegangan referensi untuk penggunaan sensor.



Gambar 4.8 Perbandingan Tegangan Referensi Pada Sensor

Bisa dilihat dari Gambar 4.8, dengan menggunakan tegangan referensi yang semakin kecil, maka ketelitian pada nilai ADC semakin besar. Sebagai contoh, ketika menggunakan tegangan referensi 5 volt, maka nilai ADC/derajat adalah 1023/300 = 3.41. ini berarti nilai ADC yang terbaca sekitar 3 poin/derajat. Dengan kata lain, sensor memiliki ketelitian sebesar 1/3.41 = 0.29^0/ nilai ADC. Sedangkan untuk tegangan referensi sebesar 2.5 volt, maka nilai ADC/derajat adalah 1023/150 = 6.82. ini berarti nilai ADC yang terbaca sekitar 6-7 poin/derajat. Sehingga, sensor memiliki ketelitian sebesar 1/6.82 = 0.14^0/ nilai ADC.

4.3 Rangkaian *Driver* Motor DC

Pengujian pada *driver* motor DC dilakukan dengan cara memberikan logika pada driver melalui mikrokontroler. Pengujian dilakukan dengan cara mengukur tegangan keluaran pada rangkaian pada saat dibebani oleh motor DC dan pada saat tidak dibebani oleh motor DC (Tabel 4.2 dan Tabel 4.3). Pemberian logika dilakukan dengan cara memberikan logika 1 – 0 , 0 – 1, 0 – 0 dan 1 – 1 pada dua buah masukan *driver* motor DC yang dilakukan melalui mikrokontroler. Mokrokontroler akan memberikan tegangan 5 volt pada logika 1 dan tegangan 0 volt pada logika 0. Tegangan ini yang nantinya akan men-*drive* transistor pada *driver* untuk menggerakan motor DC.

Tabel 4.2 Pengujian Rangkaian *Driver* Motor Tanpa Beban

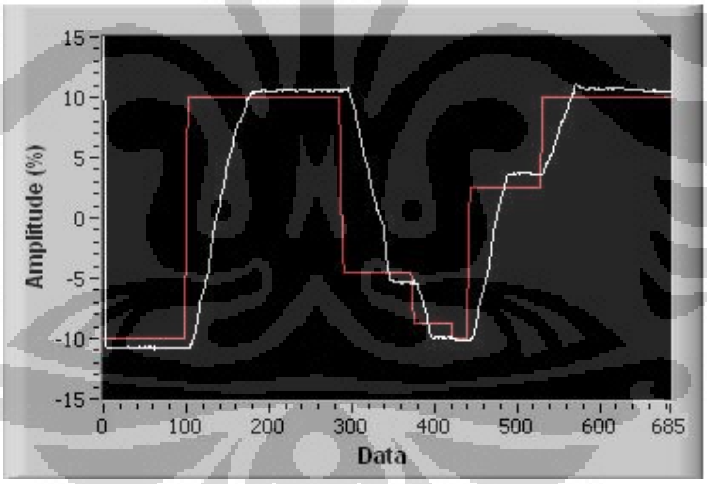
Enable	**Masukan 1**	**Masukan 2**	**Keluaran 1**	**Keluaran 2**
Ven = low	low	low	low	low
	low	high	low	low
	high	low	low	low
	high	high	low	low
Enable	**Masukan 1**	**Masukan 2**	**Keluaran 1**	**Keluaran 2**
Ven = high	low	low	low	low
	low	high	low	high
	high	low	high	low
	high	high	high	high

Tabel 4.3 Pengujian Rangkaian *Driver* Motor Dengan Beban

Enable	Masukan 1	Masukan 2	Kondisi Motor
Ven = 0 volt	0 volt	0 volt	diam
	0 volt	5 volt	diam
	5 volt	0 volt	diam
	5 volt	5 volt	diam
Ven = 5 volt	0 volt	0 volt	diam
	0 volt	5 volt	putar kiri
	5 volt	0 volt	putar kanan
	5 volt	5 volt	diam

4.4 Pengujian Dengan Sistem Loop Terbuka

Pengambilan data ini menggunakan sistem loop terbuka, dimana tidak ada *feedback* dari sensor kepada sistem. Sistem akan berhenti jika sirip telah mencapai *set point* tanpa memperhitungkan nilai *overshoot* atau *error* yang dihasilkan (Gambar 4.9).

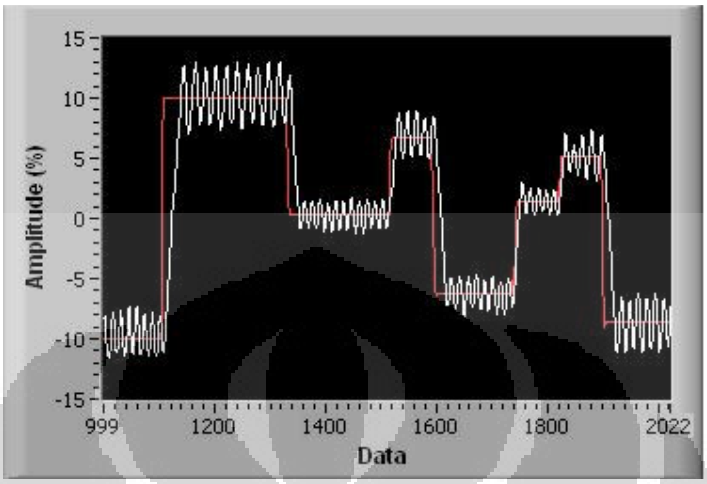


Gambar 4.9 Sistem Loop Terbuka

4.5 Pengujian Dengan Sistem Loop Tertutup

Pengambilan data ini menggunakan sistem loop tertutup, dimana sensor berfungsi sebagai *feedback* terhadap sistem. Sistem akan terus menuju titik *set poin* sampai nilai *set point* sama dengan nilai proses variabel. Namun dalam pengujian ini belum ada sistem pengendali yang berfungsi mengurangi kecepatan

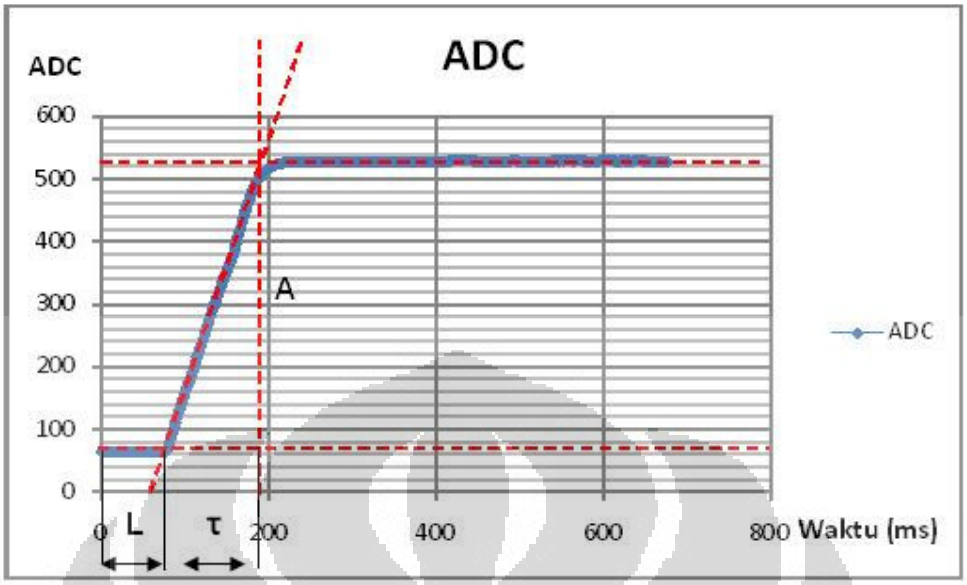
motor pada saat menuju titik *set point* dan juga terdapatnya nilai ADC yang fluktuatif sehingga sistem tidak dapat mencapai titik *set point* (Gambar 4.10).



Gambar 4.10 Sistem Loop Tertutup

4.6 Pengujian Sistem Pengendalian Dengan Metode Ziegler – Nichols

Pengujian sistem pengendali dengan metode Ziegler - Nichols berfungsi mencari fungsi tranters dengan mencoba sistem tersebut. Dari Grafik daya 100% pada pengujian respon motor merupakan langkah awal yang lakukan untuk menentukan nilai matematik dari respon sistem. Untuk mendapatkan fungsi tranfersnya harus mencari nilai keterlambatan transportasi (L) dan nilai konstanta waktu proses (τ). Untuk mendapatkan nilai L, yakni dengan menarik garis lurus pada saat grafik mengalami kenaikan yang secara konstan. Kemudian ditarik garis lurus horisontal untuk mendapatkan titik temu dengan garis lurus pada grafik. Pada titik temu tersebut ditarik garis vertikal hingga diketahui posisi garis vertikal tersebut. Nilai L adalah nilai yang ditunjukkan garis tersebut.



Gambar 4.11 Mencari Nilai L dan τ

Sedangkan untuk Mendapatkan nilai τ dan A, yaitu dengan menarik garis vertikal yang melewati titik awal proses hingga titik *set point* (Gambar 4.11).

Bila dilihat dari gambar diatas, dapat diketahui fungsi transfer dari system lalu dengan menggunakan metode Zigler - Nicholsdapat diperoleh nilai Kp, Ti dan Td. Sebelum itu, dari gambar diatas diperoleh nilai L= 81 ms, nilai τ = 194-81 = 113, dan A = 510-64 = 446. Setelah diperoleh nilai L, τ, dan A, maka akan didapat nilai R (Persamaan 4.2), Kp (Persamaan 4.3), Ti (Persamaan 4.4), dan Td (Persamaan 4.5). setelah mendapatkan nilai Kp, Ti, dan Td, maka akan diperoleh nilai Ki (Persamaan 4.6) dan Kd (Persamaan 4.7).

1. $R = A / \tau$ (4.2)

   $R = 446 / 113$

   $R = 3.947$

2. $Kp = 1.2 / RL$ (4.3)

   $Kp = 1.2 / 3.947 * 81$

   $Kp = 1.2 / 319.7$

   $Kp = 0.0037$

3. Ti = 2L (4.4)

   Ti = 2 * 81

   Ti = 162

4. Td = 0.5L (4.5)

   Td = 0.5 * 81

   Td = 40.5

5. Ki = Kp / Ti (4.6)

   Ki = 0.0037 / 162

   Ki = 0.000022

6. Kd = Kp * Td (4.7)

   Kd = 0.0037 * 40.5

   Kd = 0.14985

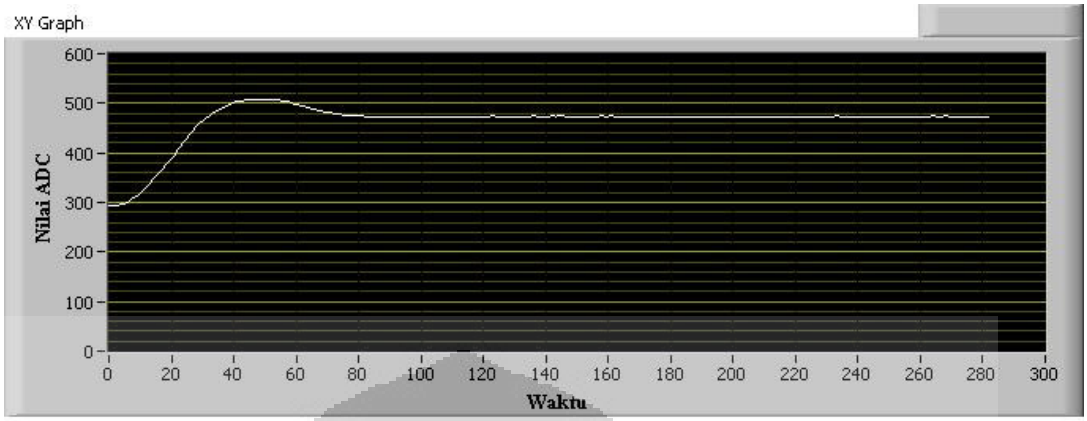
Setelah itu, nilai dari persamaan-persamaan diatas yang telah didapatkan kemudian dimasukkan kedalam persamaan dibawah ini, yang diprogram dalam mikrokontroler (Persamaan 4.8).

$$MV = Kp\left(E + \frac{1}{Ti}\int E\,dt + Td\,\frac{dE}{dt}\right)$$

$$Mv = Kp * E + Ki\int E\,dt + Kd\,\frac{dE}{dt} \quad (4.8)$$

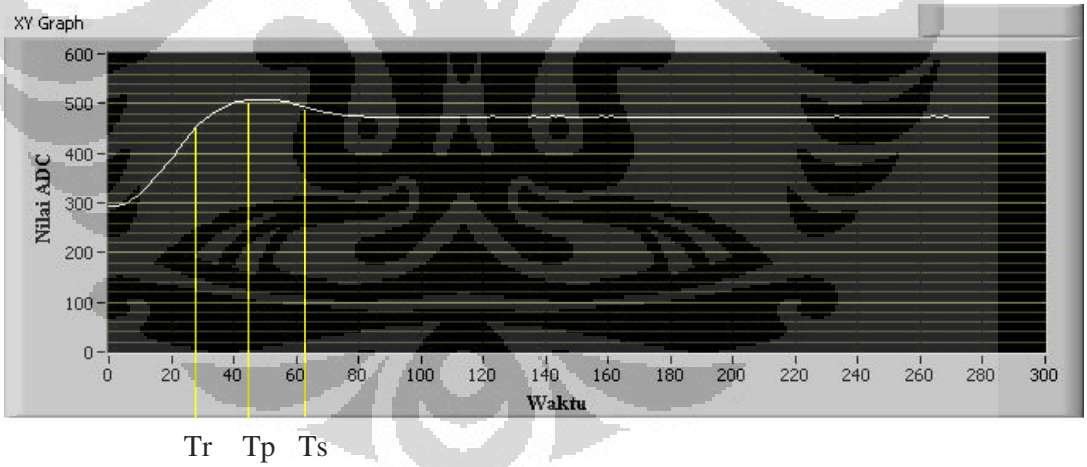
Dimana:

MV = (*Manipulated Variable*)



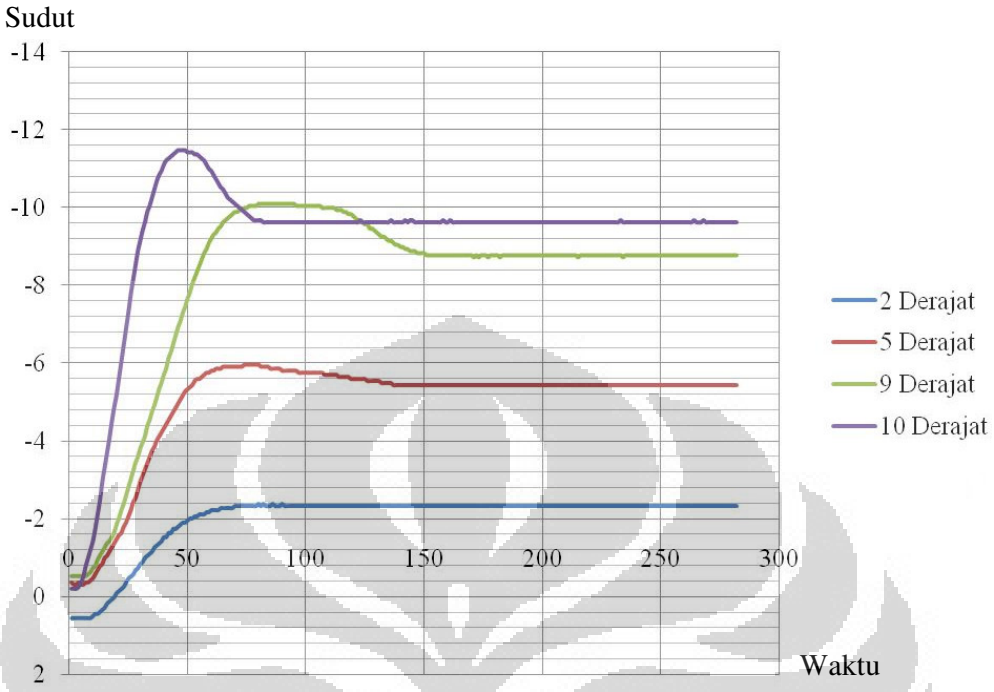
Gambar 4.12 Sistem Ziegler – Nichols

Grafik diatas (Gambar 4.12) merupakan pergerakan aktuator dari 0 sampai -10 derajat, terlihat ada sedikit *overshoot* dan kemudian stabil. Namun pada saat stabil, data terlihat sedikit mengalami fluktuasi yang disebabkan tidak stabilnya data ADC pada keluaran sensor.



Gambar 4.13 Mencari Nilai Tr, Tp, Ts, dan %OS

Dari grfaik di atas (Gambar 4.13) didapat nilai Tr (waktu naik) = 290 milisekon, Tp (waktu puncak) = 450 milisekon, Ts (waktu penetapan) = 630 milisekon, dan OS = 1.4^0 atau %OS = 7%



Gambar 4.14 Grafik Nilai Sudut Terhadap Waktu

Dari grafik di atas (Gambar 4.14), didapatkan nilai Tr (*rising time*), Tp (*peak time*), Ts (*settling time*), %OS (*overshoot*), dan SSE (*steady state error*) pada masing – masing sudut yang ditampilkan pada Tabel 4.4.

Tabel 4.4 Respon Transien

Sudut (derajat)	Respon Transien				
	Tr (detik)	Tp (detik)	Ts (detik)	%OS	%SSE
2	0.36	0.7	0.7	4%	37.50%
5	0.47	0.74	0.74	6%	14%
9	0.56	0.8	1.18	4%	5%
10	0.42	0.46	0.63	7.5%	2.50%
Rata - Rata	**0.42**	**0.675**	**0.8125**	**5.375%**	**14.75%**

Tabel 4.5 Perbandingan PID dan Logika Fuzzy

Metode Pengendalian	Respon Transien				
	Tr (detik)	Tp (detik)	Ts (detik)	%OS	%SSE
PID	0.42	0.675	0.8125	5.375%	14.75%
Logika Fuzzy	0.32	0.47	0.72	21.57%	20%

Bisa dilihat dari Tabel 4.5, pengendalian menggunakan PID mempunyai nilai Tr (*time rising*), nilai Tp (*time peak*) dan nilai Ts (*settling time*) lebih lambat dari pada logika fuzzy, namun dilihat dari nilai persentase *overshoot* dan nilai *steady state error*, PID jauh lebih baik daripada logika fuzzy.

Dari data Tr, Tp, dan Ts di atas, dapat disimpulkan bahwa pengendalian menggunakan sistem PID, sedikit lebih lambat untuk mencapai nilai/titik *set point* dibandingkan menggunakan logika fuzzy. namun dari nilai persentase *overshoot* dan *steady state error*, menggunakan pengendalian PID jauh lebih baik dibandingkan dengan logika fuzzy, dengan kata lain, pengendalian menggunakan PID lebih kecil kesalahannya pada saat pencapaian nilai *set point* dari pada menggunakan logika fuzzy. Kesimpulan dan analisa berdasarkan pada studi kasus diatas yaitu, sistem aktuator kendali sirip.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan perancangan sistem serta pengujian terhadap sistem tersebut, maka penulis dapat mengambil kesimpulan bahwa:

1. Penggunaan sensor rotasi dengan *output* tegangan yang fluktuatif sangat mempengaruhi performa dari pergerakan sirip.
2. Didapatkan bahwa respon terbaik sistem diperoleh pada saat Kp = 0.037 , Ki = 0.000022, dan Kd = 0.14985.
3. Respon Transien rata – rata Tr (waktu naik) = 420 milisekon, Tp (waktu puncak) = 675 milisekon, Ts (wakt penetapan) = 812.5 milisekon, %OS = 5.375% dan *steady state error* = 14.75%.
4. Penggunaan *planetary gear* sangat membantu dalam mengurangi kecepatan motor dan juga menambah torsi motor yang sangat bagus digunakan untuk aplikasi gerak sirip roket.
5. Penggunaan tegangan referensi untuk ADC sebesar 2.5 volt, lebih baik dari penggunaan tegangan referensi 5 volt.

5.2 Saran

Untuk pengembangan peneltian ini pada masa yang akan datang penulis memiliki beberapa saran sebagai berikut:

1. Menggunakan sensor rotasi dengan tegangan *output* yang stabil
2. Penggunaan motor DC sebagai penggerak sirip kurang bagus, karena motor DC sangat cepat putarannya, disarankan menggunakan motor *stepper* atau motor servo sebagai penggerak.
3. Sebaiknya menggunakan rodagigi dengan bahan yang kuat, seperti besi / baja, agar rodagigi tidak cepat aus yang dapat menimbulkan *loss* pada roda gigi.
4. Gunakan roda gigi kerucut dengan perbandingan yang lebih besar, agar *loss* pada saat motor berhenti, tidak mempengaruhi pergerakan sirip.

DAFTAR ACUAN

[1] Rahman, M. Aulia. (2011). Rancang bangun hotplate stirrer magnetik terkendali temperatur dan kecepatan. Dalam laporan skripsi. Depok: Departemen Fisika. Universitas Indonesia.

[2] Kurniawan, Arief. (2012). Rancang bangun prototipe sistem aktuator sirip roket kendali menggunakan *brushedDC motor* dan *planetary gear.* Dalam laporan skripsi. Depok: Departemen Elektro. Universitas Indonesia.

[3] Malvino, Prinsip-Prinsip Elektronika, Erlangga, Jakarta, 1995.

[4] Clayton, George. *Operational Amplifier*. Erlangga, 2002.

[5] Dale E., Seborg. *Practical Dynamics And Control*. John Wiley & Sons Inc, 1989.

[6] Dfrobot. (2012). *Rotation sensor*. Diambil tanggal 30 april 2011 dari http://www.robotshop.com/dfrobot-rotation-sensor-v1.html

[7] http://etekno.blogspot.com/2012/02/berkenalan-dengan-labview. akses tanggal 3 mei 2012 jam 11.30 WIB.

[8] http://elib.unikom.ac.id/files/disk1/398/jbptunikompp-gdl-setiawanar-19859-9-babii. akses tanggal 3 mei 2012 jam 11.30 WIB.

[9] Nurijah. (2009). Sistem Pengendali Temperatur dan Kecepatan *shaking water bath* Berbasis Mikrokontroler. Dalam laporan tugas akhit. Depok: Departemen Fisika. Universitas Indonesia.

[10] Wikipedia. (2012). PID. Diambil tanggal 10 mei 2012 dari http://id.wikipedia.org/wiki/PID.

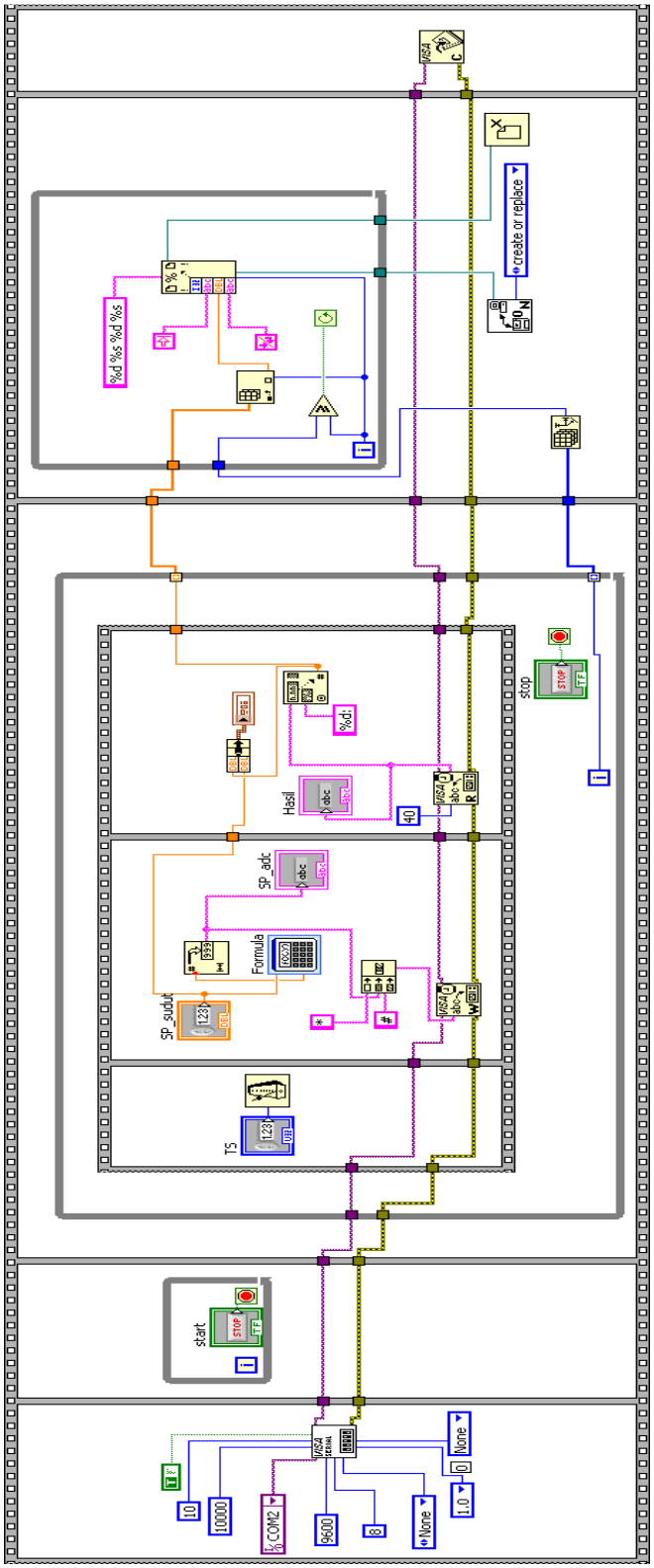
[11] Motor, Maxon. (2012). *Planetary gearhead*. Diambil tanggal 10 mei 2012 dari http://uk.farnell.com/maxon-motors/166174/gearhead planetary-246-1-ratio/dp/1761297

[12] http://catatan-elektro.blogspot.com/2011/11/pengertian-kendali-pid.html. akses tanggal 18 juni 2012.

[13] http://manufakturpolman.blogspot.com/2010/12/roda-gigi-gear.html. akses tanggal 18 juni 2012.

[14] http://m-edukasi.net/online/2008/transmisiotomatis/materi02b.html. akses tanggal 18 juni 2012.

Lampiran 1. Blok Diagram LabVIEW



Lampiran 2. Listing Program

```
$regfile = "m8535.dat"
$crystal = 11059200
$baud = 9600

'=========================' ADC
Config Adc = Single , Prescaler = Auto
Start Adc
Dim Dataadc As Word
'=========================' ADC

'=========================' INTERRUPTS SERIAL
On Urxc , Getdata
Enable Urxc
Enable Interrupts

Dim Dataserial As Byte
'=========================' INTERRUPTS SERIAL

'=========================' PWM MOTOR
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Compare B
Pwm = Clear Down , Prescale = 1
Config Portd.6 = Output
Config Portd.7 = Output

Dim Nilaipwm As Single
Dim Nilaipwm_w As Word
'=========================' PWM MOTOR

Dim Kp As Single
Dim Ki As Single
Dim Kd As Single
Dim Sp_s As String * 6
Dim Set_point As Single
Dim E As Single
Dim De As Single
Dim Ebefore As Single
Dim Pterm As Single
Dim Iterm As Single
Dim Dterm As Single
Dim Flagproses As Byte
Dim Vout As Single
Dim Voutabs As Single

Portd.6 = 0
Portd.5 = 0
```

Lampiran 2. Program (Lanjutan)

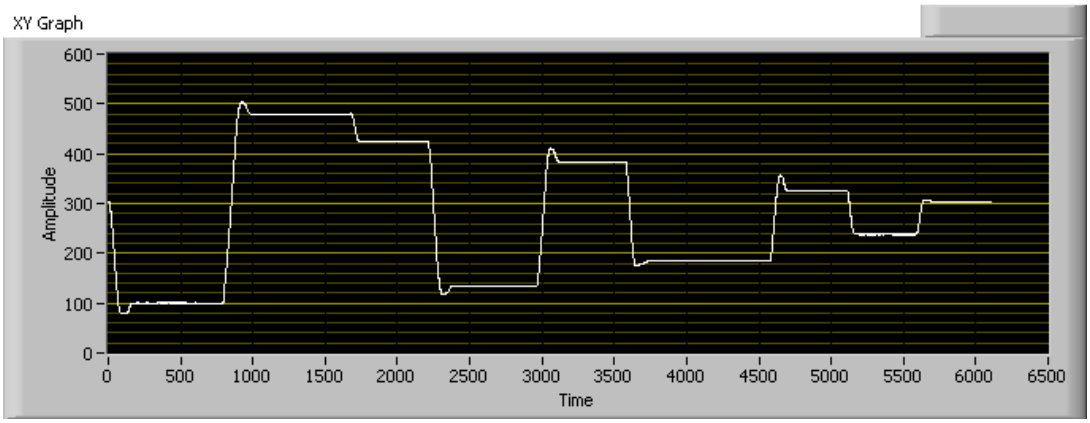
```
'========================== PID
    Kp = 0.0037
    Ki = 0.000022
    Kd = 0.14985
    Dataadc = Getadc(0)
Do
  If Flagproses = 4 Then
    Flagproses = 0
    Set_point = Val(sp_s)
    E = Set_point - Dataadc
    Pterm = Kp * E
    Iterm = Ki * E
    Iterm = Iterm + Ebefore
    De = E - Ebefore
    Dterm = De
    Dterm = Kd * De
    Vout = Pterm + Iterm
    Vout = Vout + Dterm
    Voutabs = Abs(vout)
    Nilaipwm = Voutabs * 1023
    Nilaipwm = Nilaipwm / 100
    If Vout > 0 Then
      Portd.6 = 1
      Nilaipwm = 1023 - Nilaipwm
    Else
      Portd.6 = 0
    End If
    If Nilaipwm > 1023 Then
      Nilaipwm = 1023
    Elseif Nilaipwm < 0 Then
      Nilaipwm = 0
    End If
    Nilaipwm_w = Nilaipwm
    Pwm1a = Nilaipwm_w
    Pwm1b = Nilaipwm_w

    Ebefore = E
    Dataadc = Getadc(0)
    Print Dataadc ; ":"
  End If
Loop
'========================== PID
```

Lampiran 2. Program (Lanjutan)

```
'========================== AMBIL DATA SERIAL
Getdata:
Dataserial = Waitkey()
If Dataserial = "*" Then
   Sp_s = ""
Elseif Dataserial = "#" Then
   Flagproses = 4
Else
   Sp_s = Sp_s + Chr(dataserial)
End If
Return
'========================== AMBIL DATA SERIAL
```

Lampiran 3. Grafik Hasil Pengujian Sudut



Grafik sudut dengan masukan secara acak

Lampiran 4. Gambar Prototipe Aktuator