BAB II

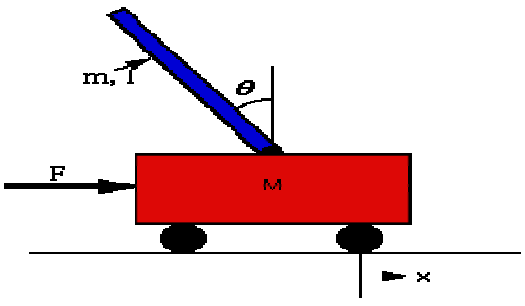
TINJAUAN PUSTAKA

2.1 Teori Pendulum Terbalik

Pendulum terbalik pada dasarnya hanya suatu batang stick berdiri akan jatuh bebas akibat gaya tarik bumi jika tidak ada gaya luar lain yang mengimbanginya. Secara nyata dapat dilihat akibat ketidakstabilan sistem. Pendulum akan jatuh ke kiri atau kanan.

Dalam kehidupan nyata, pendulum terbalik ini dapat dilihat penggunaanya dalam pengendalian posisi antena radar dan pengendalian lengan robot pada robot yang bergerak (misalnya membawa kamera pada daerah yang tidak dapat dijangkau manusia).

Kereta dengan pendulum terbalik, ditampilkan dibawah ini dengan gaya impuls F. Persamaan gerak untuk dinamika sistem dan linearitas tentang sudut pendulum adalah $\theta = P$ atau dengan kata lain, pendulum diasumsikan tidak bergerak lebih dari beberapa derajat dari sumbu vertikal yang berada pada sudut θ.



Gambar 2.1 Desain pendulum terbalik

(http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html)

Keterangan:

M = Massa kereta

m = Massa pendulum

b = Gesekan kereta

l = Panjang pendulum pusat massa

I = Inersia pendulum

F = Gaya yang diberikan pada kereta

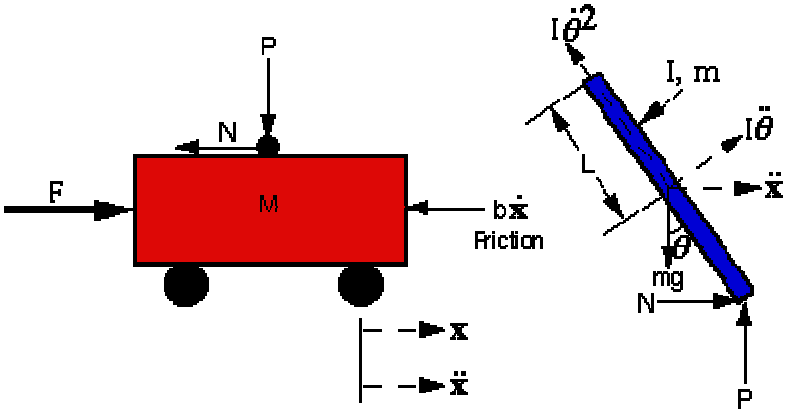
x = Posisi koordinat kereta

θ = Sudut vertikal pendulum

Untuk PID, *root locus* dan bagian respon frekuensi dari masalah ini kita hanya akan tertarik pada kontrol posisi pendulum saja. Karena teknik yang digunakan dalam tutorial ini hanya dapat diterapkan untuk sistem *Single Input Single Output* (SISO).

2.1.1 Analisis Gaya dan Sistem Persamaan

Berikut adalah dua bangun diagram sistem.



Gambar 2.2 Dua bangun diagram sistem kereta pendulum

(http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html)

Dengan menjumlahkan badan kereta kearah horizontal, maka persamaan gerak kereta adalah:

$$M\ddot{x} + b\dot{x} + N = F \quad \textbf{(2.1)}$$

Dengan menjumlahkan batang pendulum kearah horizontal, maka diperoleh persamaan untuk N:

$$N = m\ddot{x} + ml\ddot{\theta}cos\theta - m\dot{\theta}^2 l sin\theta \quad \textbf{(2.2)}$$

Jika persamaan **(2.1)** dan **(2.2)** disubtitusi, maka persamaan gerak untuk sistem ini adalah:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - m\dot{\theta}^2 l sin\theta = F$$

Untuk mendapatkan persamaan gerak kedua, dengan menjumlahkan garis tegak lurus pada pendulum. Persamaanya:

$$Psin\theta + Ncos\theta - mgsin\theta = m\ddot{x}cos\theta \quad \textbf{(2.3)}$$

Untuk menghilangkan P dan N dalam persamaan diatas, untuk mendapatkan persamaan berikut maka, keliling pusat massa pendulum dijumlahkan.

$$-Plsin\theta - Nlcos\theta = I\ddot{\theta} \quad \textbf{(2.4)}$$

Menggabungkan persamaan **(2.3)** dan **(2.4)** maka akan diperoleh persamaan dinamis

$$(I + ml^2)\ddot{\theta} + mgl sin\theta = -ml\ddot{x}cos\theta$$

Jika kita menganggap bahwa sistemnya linear maka diasumsikan bahwa θ = $Pi + \emptyset\emptyset$ merupakan sudut terkecil dari arah vertikal) oleh karena itu $\cos\theta = -1$, $\sin\theta = -\emptyset$ dan $(\frac{d\theta}{dt})^2$. Setelah linearisasi dua persamaan gerak menjadi seperti berikut (dimana u merupakan masukan)

$$(I + ml^2)\emptyset - mg\ddot{l}\emptyset = m\ddot{l}x$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\emptyset} = u$$

2.1.2 Fungsi Transfer

Untuk mendapatkan fungsi transfer dari sistem linearisasi persamaan analitis, kita harus terlebih dahulu mengambil transformasi *Laplace* dari persamaan sistem. Transformasi *Laplace*nya adalah:

$$(I + m^2 l)\varphi(s)s^2 - m\ g\ l(\varphi) = m\ l\ X\ (s^2)s \qquad \textbf{(2.5)}$$

$$(M + m\ )X(s)s^s + b\ X\ ()ss - m\ l(\varphi)s^2 = U\ (s\ ) \qquad \textbf{(2.6)}$$

Catatan: Saat fungsi transfer inisial kondisinya diasumsikan jadi nol (0).

Untuk mencari sudut φ sebagai output (keluaran) maka persamaan pertama untuk X(s):

$$X(s) = \left[\left(\frac{I + m^2 l}{m\ l} - \frac{g}{s^2} \right) \right] \varphi\ (s\ ) \qquad \textbf{(2.7)}$$

kemudian persamaan **(2.7)** disubtitusi ke persamaan **(2.6)**

$$(M + m)\left[\left(\frac{I + m^2 l}{m\ l} + \frac{g}{s} \right) \right] \varphi(s)s^2 + b\ \left[\left(\frac{I + m^2 l}{m\ l} + \frac{g}{s} \right) \right] \varphi(s)s - m\ l(\varphi)s^2 = U\ (s\ )$$

dimana

$$q = [(M + m)(I + m^2 l - (m\ l)^2]$$

Fungsi transfernya adalah:

$$\frac{\varphi\ (s\ )}{U\ (s\ )} = \frac{\frac{m\ l}{q} s^2}{s^4 + \frac{b\ (I + m^2 l}{q} s^3 - \frac{(M + m)m\ g\ l}{q} s^2 - \frac{b\ m\ g}{q} s l}$$

2.1.3 Kestabilan Sistem

Suatu sistem dinamik dikatakan stabil apabila sistem tersebut dapat kembali ke posisi setimbangnya semula, apabila diberikan input lalu input tersebut dihilangkan. Secara matematik, hal ini dapat dilihat dari posisi akar-akar karakteristik sistem tersebut. Apabila semua akar karakteristiknya negatif, maka

sistemnya stabil dan apabila minimal terdapat satu akar karakteristik yang positif, maka sistemnya tidak stabil. Selanjutnya istilah akar karakteristik digantikan dengan *pole*.

Pada sistem dinamik yang direpresentasikan oleh sebuah fungsi transfer, akan memiliki *zero* dan *pole*. Sebagai contoh jiika suatu sistem memiliki fungsi transfer

$$H(s) = \frac{s\ +1}{s^2 + 5s\ +6}$$

Maka sistem ini akan memiliki *zero*, yaitu dengan membuat numeratornya sama dengan nol.

$$s + 1 = 0, \quad s = -1$$

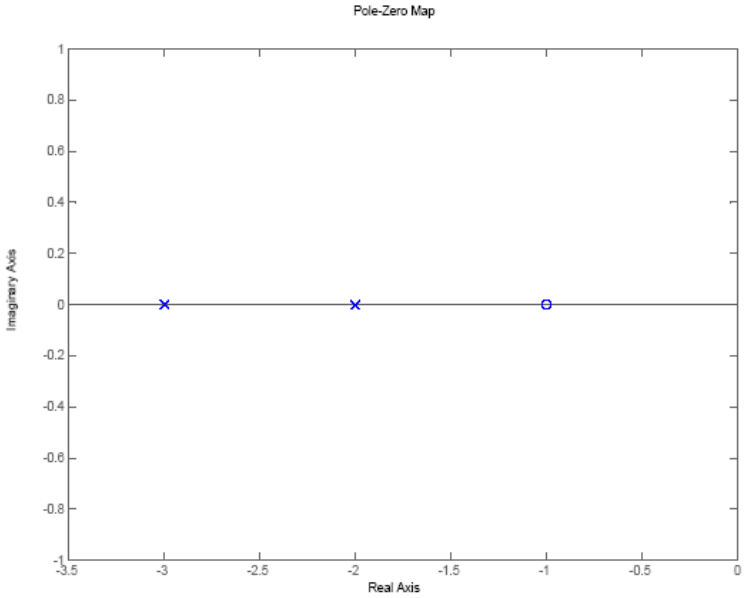
Jadi, *zero*-nya adalah -1.

Untuk memperoleh *pole*-nya, maka denumeretornya dibuat sama dengan nol, sehingga menghasilkan

$$s^2 + 5s\ + 6 = 0, s\ 1 = -2\ d\ a\ n\ \ s\ 2 = -3$$

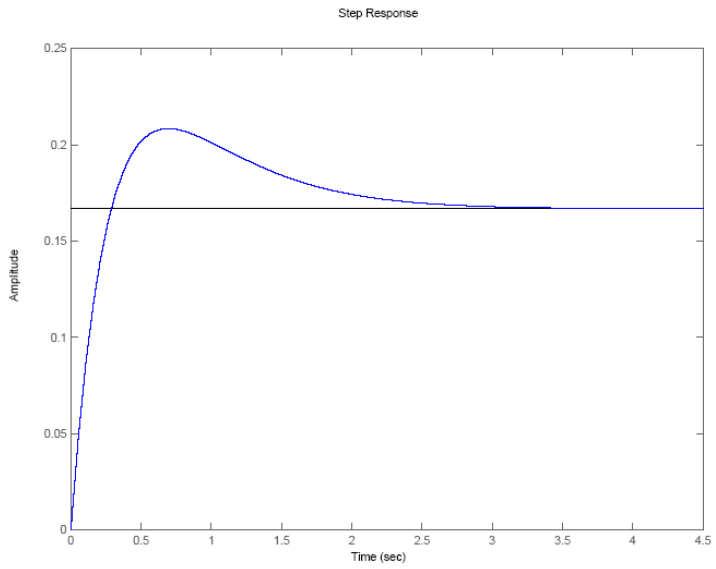
Jadi, *pole-pole* yang diperoleh adalah -2 dan -3.

Dari persyaratan kestabilan, maka sistem tersebut merupakan sistem yang stabil karena memiliki semua pole yang bagian riilnya negatif. Letak *pole-pole* dan *zero* sistem ini diperlihatkan pada gambar berikut ini.



Gambar 2.3 Respon letak pole dan zero sistem

Jika sistem ini diberikan masukan step, maka sistem akan menghasilkan keluaran yang mencapai *steady-state* (konstan) pada suatu waktu tertentu. Berikut adalah respon waktu dari sistem ini.



Gambar 2.4 Respon sistem saat mencapai steady state

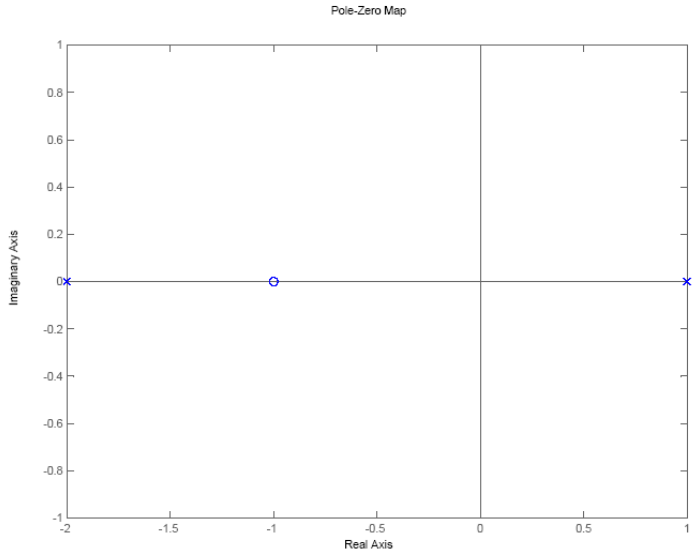
Bandingkan dengan sistem berikut ini

$$H(s) = \frac{s\ +1}{s^2 + s\ +2}$$

Pole-pole sistem ini adalah

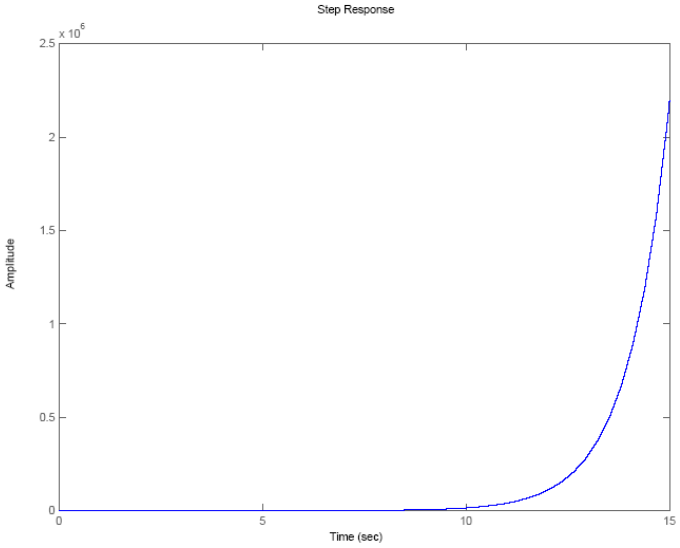
$$s^2 + s - 2 = 0, \quad sehingga$$
$$s1 = 1 \quad dan \quad s2 = -2$$

Letak *pole* dan *zero*-nya pada bidang kompleks adalah sebagai berikut



Gambar 2.5 Respon letak pole dan zero pada bidang kompleks

Respon waktu sistem ini terhadap masukan step adalah keluarannya akan terus bertambah besar (tidak stabil). Berikut ini adalah gambar respon waktu step sistem.



Gambar 2.6 Respon waktu step sistem

2.2 Pengontrol *Proportional Integral Derivative* (PID)

Pengontrol PID adalah pengendali umpan balik (*controller*) yang banyak digunakan dalam industri sistem kontrol dan paling umum. Sebuah kontroller PID menghitung nilai error sebagai perbedaan antara variabel proses dengan *set point* yang diinginkan. Kontroller berupaya untuk meminimalkan kesalahan dengan menyesuaikan proses input kontrol.

Persamaan pengontrol PID adalah:

$$m v(t) = K p \left(e(t) + \frac{1}{T i} \int_0^1 e(t) d t + T d \frac{d e(t)}{d t} \right) \quad \textbf{(2.8)}$$

Keterangan:

$m v(t) = output dari pengontrol PID atau Manipulated$

$K p = konstanta Proporsional$

$T i = Konstanta Integral$

$T d = Konstanta Derivatif$

$e(t) = error (Selisihantara set point dengan level akt$

Persamaan pengontrol PID diatas dapat juga dituliskan sebagai berikut:

$$m v(t) = K p (e(t) + K i \int_0^1 e(t) d t + K d \frac{d e(t)}{d t} \quad \textbf{(2.9)}$$

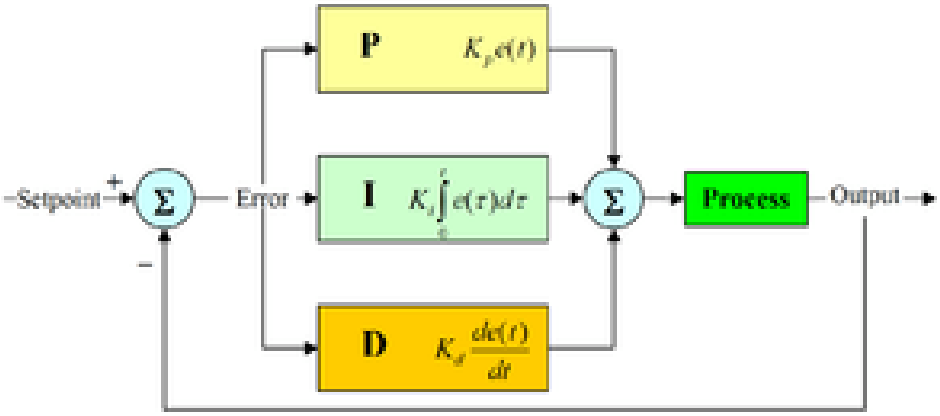
dengan

$$K i = K p \frac{1}{T i} dan \; K d = K p \times T d$$

Untuk lebih memaksimalkan kerja pengontrol diperlukan nilai batas minimum dan maksimum yang akan membatasi nilai manipulated variabel yang dihasilkan.

2.2.1 Teori Pengontrol *Proportional Integral Derivative* (PID)

Pengontrol PID melibatkan tiga parameter yang terpisah yaitu *proportional, integral,* dan *derivative*, parameter tersebut dinotasikan P, I, dan D. Nilai *proportional* menentukan reaksi terhadap kesalahan saat ini, nilai *integral* menentukan reaksi berdasarkan jumlah kesalahan baru-baru ini dan nilai *derivative* merupakan reaksi berdasarkan tingkat di mana kesalahan telah berubah. Jumlah kesetimbangan dari ketiga tindakan ini digunakan untuk mengatur proses tersebut melalui elemen kontrol seperti posisi katup kontrol atau catu daya dari elemen pemanas.

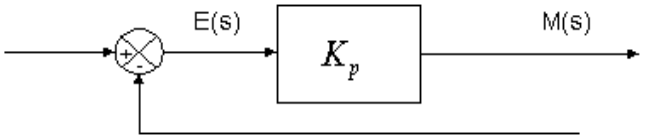


Gambar 2.7 Diagram blok pengontrol PID

1. Pengontrol *Proportional*

   Kontrol P, jika G(s) = kp dengan k adalah konstanta. Jika u = G(s)*e maka u = Kp*e dengan Kp adalah konstanta *proportional*. Kp berlaku sebagai *gain* (penguat) saja tanpa memberikan efek dinamik kepada kinerja *controller*. Penggunaan kontrol P memiliki berbagai keterbatasan karena sifat kontrol yang tidak dinamik. Walaupun demikian dalam aplikasi-aplikasi dasar yang sederhana kontrol P ini cukup mampu untuk memperbaiki respon transien khususnya *rise time* dan *settling time*.

Gambar 2.7 menunjukkan diagram blok yang menggambarkan hubungan antara besaran *setting*, besaran aktual dengan besaran keluaran *proportional controller*. Sinyal kesalahan (*error*) merupakan selisih antara besaran *setting* dengan besaran aktualmya. Selisih ini akan mempengaruhi *controller*, untuk mengeluarkan sinyal positif (mempercepat pencapaian harga *setting*) atau negatif (memperlambat tercapainya harga yang diinginkan).



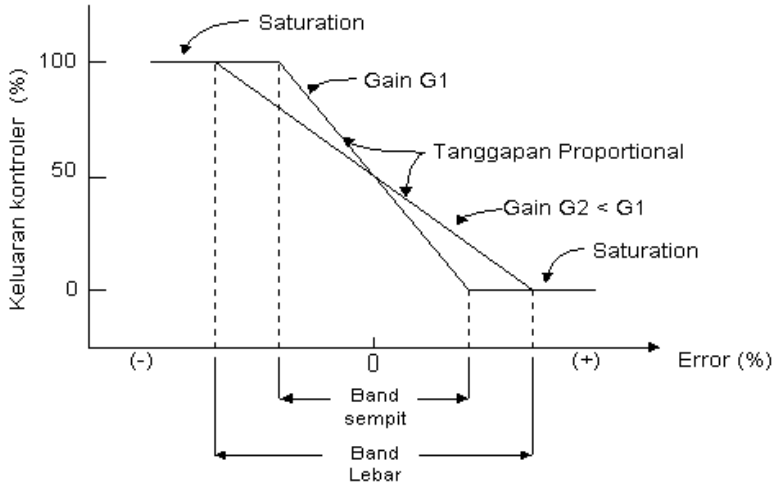
Gambar 2.8 Diagram blok pengendali proportional

Pengendali *proportional* memiliki 2 parameter, pita *proportional* (*proportional band*) dan konstanta *proportional*. Daerah kerja kontroller efektif dicerminkan oleh pita proportional, sedangkan konstanta *proportional* menunjukkan nilai faktor penguatan terhadap sinyal kesalahan, K_p.

Hubungan antara Pita *Proportional* (PB) dengan konstanta *proportional* (K_p) ditunjukkan secara presentasi oleh persamaan berikut:

$$PB = \frac{1}{Kp} x100\%$$

Gambar 2.9 menunjukkan grafik hubungan antara PB, keluaran *controller* dan kesalahan yang merupakan masukan *controller*. Ketika konstanta *proportional* bertambah semakin tinggi, PB menunjukkan penurunan yang semakin kecil, sehingga lingkup kerja yang dikuatkan akan semakin sempit.



Gambar 2.9 *Proportional band* dari *proportional controller* tergantung pada penguatan

Ciri-ciri pengontrol *proportional*:

a. Jika nilai Kp kecil, pengontrol *proportional* hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat.
b. Jika nilai Kp dinaikkan, respon/tanggapan sistem akan semakin cepat mencapai keadaanya (mengurangi *rise time*).
c. Jika harga Kp diperbesar sehingga mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil atau respon sistem akan berosilasi.
d. Nilai Kp dapat diset sedemikian sehingga mengurangi *steady state error*, tetapi tidak menghilangkannya.

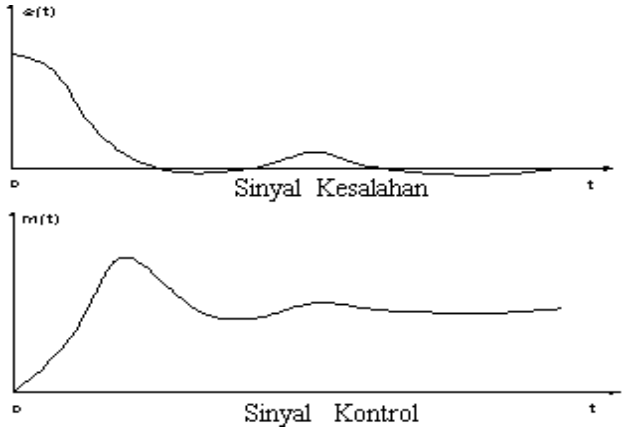
2. Pengontrol *Integral*

Jika *G(s)* adalah kontrol I maka *u* dapat dinyatakan sebagai *u(t) = [∫(t)dt]* Ki. Dengan Ki adalah konstanta *integral*, dan dari persamaan diatas, *G(s)* dapat dinyatakan sebagai $u = Kd \times \Delta e/\Delta t$. Jika *e(t)*

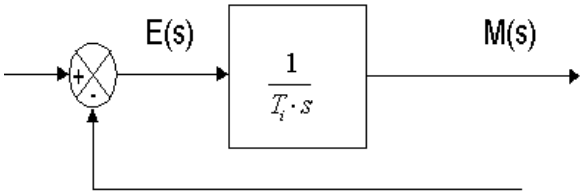
mendekati konstan (bukan nol) maka *u(t)* akan menjadi sangat besar sehingga diharapkan dapat memperbaiki error. Jika *e(t)* mendekati nol maka efek kontrol I ini semakin kecil. Kontrol I dapat memperbaiki sekaligus menghilangkan respon *steady state*, namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi sehingga dapat menyebabkan ketidakstabilan sistem. Pemilihan Ki yang sangat tinggi justru dapat menyebabkan output berosilasi karena menambah orde sistem.

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

Sinyal keluaran *integral controller* merupakan luas bidang yang dibentuk oleh kurva kesalahan penggerak. Sinyal keluaran akan berharga sama dengan harga sebelumnya ketika sinyal kesalahan berharga nol. Gambar 2.10 menunjukkan contoh sinyal kesalahan yang disulutkan ke dalam pengontrol *integral* dan keluaran pengontrol *integral* terhadap perubahan sinyal kesalahan tersebut.

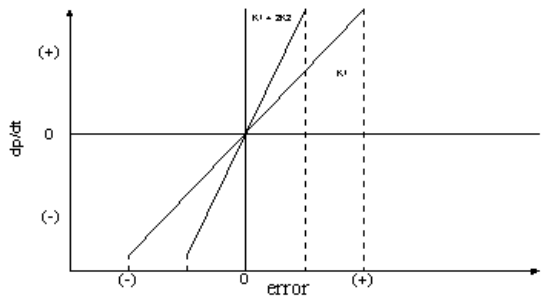


Gambar 2.10 Kurva sinyal kesalahan *e(t)* terhadap *t* dan kurva *u(t)* terhadap *t* pada pembangkit kesalahan nol



Gambar 2.11 Blok diagram hubungan antara besaran kesalahan dengan keluaran *integral controller*

Pengaruh perubahan konstanta *integral* terhadap keluaran *integral* ditunjukkan oleh Gambar 2.12. Ketika sinyal kesalahan berlipat ganda, maka nilai laju perubahan keluaran *controller* berubah menjadi dua kali dari semula. Jika nilai konstanta integrator berubah menjadi lebih besar, sinyal kesalahan yang relatif kecil dapat mengakibatkan laju keluaran menjadi besar.



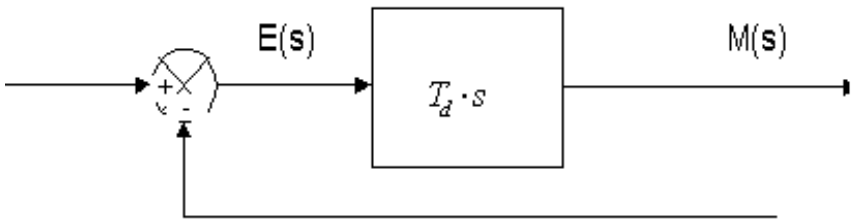
Gambar 2.12 Perubahan keluaran sebagai akibat penguatan dan kesalahan

Ciri-ciri pengontrol *Integral*:

a. Keluaran pengontrol *integral* membutuhkan selang waktu tertentu, sehingga pengontrol *integral* cenderung memperlambat respon.
b. Ketika sinyal kesalahan bernilai nol, keluaran pengontrol akan bertahan pada nilai sebelumnya.
c. Jika sinyal kesalahan tidak bernilai nol, keluaran akan menunjukkan kenaikan atau penurunan yang dipengaruhi oleh besarnya sinyal kesalahan dan nilai Ki.
d. Konstanta integral Ki yang berharga besar akan mempercepat hilangnya *offset*. Tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengontrol.

3. Pengontrol *Derivative*

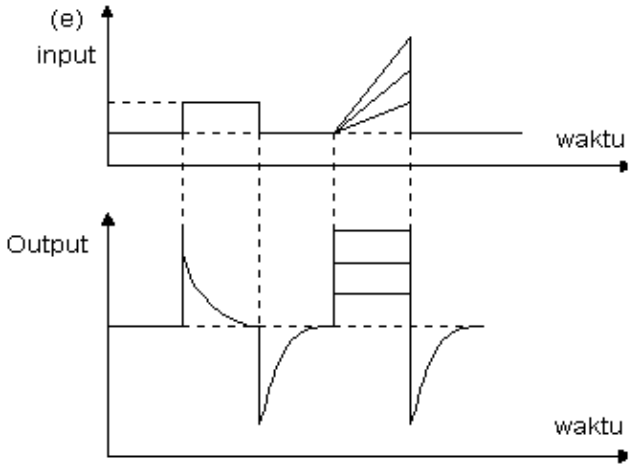
Pengontrol *derivative* memiliki sifat seperti halnya suatu operasi *derivative*. Perubahan yang mendadak pada masukan *controller*, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.13 menunjukkan blok diagram yang menggambarkan hubungan antara sinyal kesalahan dengan keluaran *controller*.



Gambar 2.13 Blok diagram kontroller *derivative*

Gambar 2.14 menyatakan hubungan antara sinyal masukan dengan sinyal keluaran kontroller *derivative*. Ketika masukannya tidak

mengalami perubahan, keluaran kontroller juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk impuls. Jika sinyal masukan berubah naik secara perlahan (fungsi *ramp*), keluarannya justru merupakan fungsi *step* yang besar magnitudnya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan faktor konstanta *derivative* T_d .



Gambar 2.14 Kurva waktu hubungan input-output kontroller *derivative*

Ciri-ciri pengontrol *derivative*:

a. Pengontrol ini tidak dapat menghasilkan keluaran jika tidak ada perubahan pada masukannya (berupa perubahan sinyal kesalahan).
b. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengontrol tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
c. Pengontrol *derivative* mempunyai suatu karakter untuk mendahului, sehingga pengontrol ini dapat menghasilkan koreksi yang signifikan

sebelum pembangkit kesalahan menjadi besar. Jadi pengontrol diferensial dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat korektif dan cenderung meningkatkan stabilitas sistem.

d. Dengan meningkatkan nilai Kd, dapat meningkatkan stabilitas sistem dan mengurangi *overshoot*.

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

Efek dari setiap pengontrol *Proportional* (P), *Integral* (I) dan *Derivative* (D) pada setiap loop tertutup disimpulkan pada Tabel 2.1 yang merupakan efek setiap pengontrol.

Tabel 2.1 Efek setiap pengontrol pada loop tertutup

Respon Loop Tertutup	Rise Time	Overshoot	Settling Time	Steady-State Error
P	Menurunkan	Meningkatkan	Perubahan kecil	Menurunkan
I	Menurunkan	Meningkatkan	Meningkatkan	Mengeliminasi
D	Perubahan kecil	Menurunkan	Menurunkan	Perubahan kecil

Elemen-elemen pengontrol P, I dan D masing-masing secara keseluruhan bertujuan:

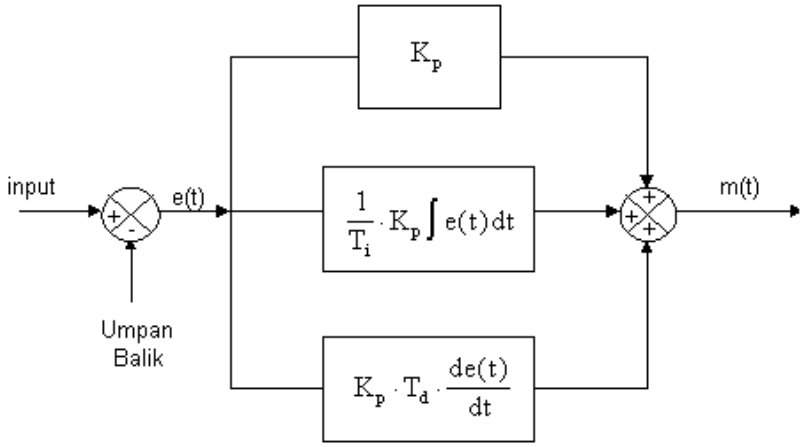
a. Mempercepat reaksi sebuah sistem mencapai *set point*-nya.

b. Menghilangkan *offset*.

c. Menghasilkan perubahan awal yang besar dan mengurangi *overshoot*.

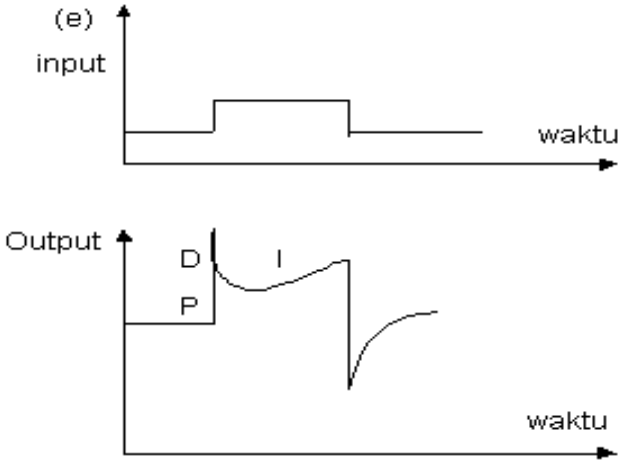
2.2.2 Kontroller PID

Setiap kekurangan dan kelebihan dari masing-masing kontroller P, I dan D dapat saling menutupi dengan menggabungkan ketiganya secara paralel menjadi kontroller *Proposional, Integral, Diferensial* (kontroller PID). Elemen-elemen kontroller P, I, dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar.



Gambar 2.15 Blok diagram kontroller PID analog

Keluaran kontroller PID merupakan penjumlahan dari keluaran kontroller *proportional* dan keluaran kontroller *integral*. Gambar 2.16 menunjukkan hubungan tersebut.



Gambar 2.16 Hubungan dalam fungsi waktu antara sinyal keluaran dengan masukan untuk kontroller PID

Karakteristik kontroller PID sangat dipengaruhi oleh kontribusi besar dari ketiga parameter P, I, dan D. Penyetelan konstanta Kp, Ti, dan Td akan mengakibatkan penonjolan sifat dari masing-masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat disetel lebih besar dibanding yang lain. Konstanta yang besar itulah akan memberikan kontribusi pengaruh pada respon sistem secara keseluruhan.

2.3 Mikrokontroller AVR ATmega8

Atmel AVR adalah jenis mikrokontroller yang paling sering dipakai dalam bidang elektronika dan instrumentasi. Mikrokontroller AVR ini memiliki arsitektur RISC (*Reduce Instruction Set Computing*) 8-bit, di mana semua instruksi dikemas dalam kode 16-bit (16 bits word) dan sebagian besar instruksi dieksekusi dalam satu siklus clock.

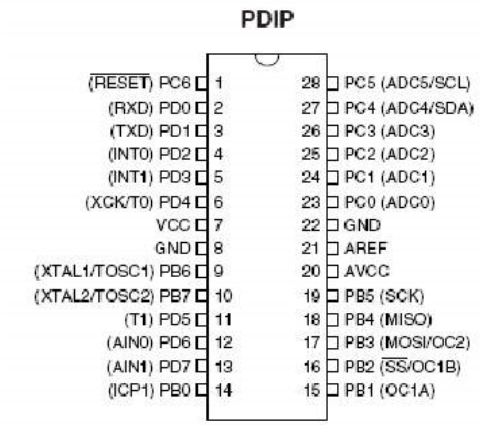
Nama AVR sendiri berasal dari *"Alf (Egil Bogen) and Vegard (Wollan) 's Risc processor"* dimana Alf Egil Bogen dan Vegard Wollan adalah dua penemu berkebangsaan Norwegia yang menemukan mikrokontroller AVR yang kemudian diproduksi oleh Atmel.



Gambar 2.17 Atmel AVR jenis ATmega8

2.3.1 Konfigurasi Pin AVR ATmega8

Dalam hal ini yang digunakan adalah AVR ATmega8, perbedaannya dengan AVR ATmega8L hanyalah terletak pada besarnya tegangan yang diperlukan untuk bekerja. Untuk ATmega8 tipe L dapat bekerja pada tegangan antara 2,7V – 5,5V sedangkan untuk ATmega8 hanya dapat bekerja pada tegangan 4,5V – 5,5V. Berikut adalah Gambar 2.18 konfigurasi pin ATmega8.



Gambar 2.18 Konfigurasi pin

Sumber : *8-bit Microcontroller with 8K Bytes In-Sistem Programmable Flash ATMega8.(ATMega8.pdf).* San Jose: Atmel Corporation, 2001. p.2.

2.3.2 Status Register AVR ATmega8

ATmega8 memiliki 28 pin yang masing-masing *pin*-nya memiliki fungsi yang berbeda-beda baik sebagai port ataupun sebagai fungsi yang lain. Berikut akan dijelaskan tentang kegunaan dari masing-masing kaki pada ATmega8.

- VCC

  Merupakan supply tegangan untuk digital.

- GND

  Merupakan ground untuk semua komponen yang membutuhkan grounding.

- Port B

  Di dalam Port B terdapat XTAL1, XTAL2, TOSC1, TOSC2. Jumlah Port B adalah 8 buah pin mulai dari pin B.0 sampai dengan pin B.7. Tiap pin dapat digunakan sebagai input dan juga output. Port B merupakan sebuah 8-bit *bidirectional* I/O port dengan internal *pull up* resistor. Sebagai input, pin-pin yang terdapat pada port B yang secara eksternal diturunkan, maka akan mengeluarkan arus jika *pull up* resistor diaktifkan. Jika ingin menggunakan tambahan kristal, maka cukup menghubungkan kaki dari kristal ke kaki pada pin port B. Namun jika tidak digunakan, maka cukup dibiarkan saja. Kegunaan dari masing-masing kaki ditentukan dari *clock fuse setting*-nya.

- Port C

  Port C merupakan sebuah 7-bit bi-directional I/O port yang di dalam masing-masing pin terdapat *pull up* resistor. Jumlah pinnya hanya 7 buah mulai dari pin C.0 sampai dengan pin C.6. Sebagai keluaran, port C

memiliki karakteristik yang sama dalam hal kemampuan menyerap arus (*sink*) ataupun mengeluarkan arus (*source*).

- Reset/PC6

  Jika RSTDISBL Fuse diprogram, maka PC6 akan berfungsi sebagai pin I/O. Untuk diperhatikan juga bahwa pin ini memiliki karakteristik yang berbeda dengan pin-pin yang terdapat pada port C. Namun jika RSTDISBL *fuse* tidak diprogram, maka pin ini akan berfungsi sebagai input reset. Dan jika level tegangan yang masuk ke pin ini rendah dan pulsa yang ada lebih pendek dari pulsa minimum, maka akan menghasilkan suatu kondisi reset meskipun *clock*nya tidak bekerja.

- Port D

  Port D merupakan 8-bit bi-directional I/O dengan internal *pull up* resistor. Fungsi dari port ini sama dengan port-port yang lain. Hanya saja pada port ini tidak terdapat kegunaan-kegunaan yang lain. Pada port ini hanya berfungsi sebagai masukan dan keluaran saja atau biasa disebut dengan I/O.

- AVCC

  Pada pin ini memiliki fungsi sebagai *supply* tegangan untuk ADC. Untuk pin ini harus dihubungkan secara terpisah dengan VCC karena pin ini digunakan untuk analog saja. Bahkan jika ADC pada AVR tidak digunakan, tetap saja disarankan untuk menghubungkan secara terpisah dengan VCC. Cara menghubungkan AVCC adalah melewati *low pass filter* setelah itu dihubungkan dengan VCC.

- AREF

  Merupakan pin referensi analog jika menggunakan ADC. Pada AVR status register mengandung beberapa informasi mengenai hasil dari kebanyakan hasil eksekusi instruksi aritmatik. Informasi ini dapat digunakan untuk altering arus program sebagai kegunaan untuk meningkatkan performa pengoperasian. Perlu diketahui bahwa register ini di *update* setelah semua operasi *Arithmetic Logic Unit* (ALU). Hal tersebut seperti yang telah tertulis dalam datasheet khususnya pada bagian *Instruction Set Reference*.

Dalam hal ini untuk beberapa kasus dapat membuang kebutuhan penggunaan instruksi perbandingan yang telah didedikasikan serta dapat menghasilkan peningkatan dalam hal kecepatan dan kode yang lebih sederhana dan singkat. Register ini tidak secara otomatis tersimpan ketika memasuki sebuah rutin interupsi dan juga ketika menjalankan sebuah perintah setelah kembali dari interupsi. Namun hal tersebut harus dilakukan melalui s*oftware*. Berikut adalah Gambar 2.19 Status register.

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

Gambar 2.19 Status register ATmega8

Sumber: *8-bit Microcontroller with 8K Bytes In-Sistem Programmable Flash ATMega8. (ATMega8.pdf)*. San Jose: Atmel Corporation, 2001. p.11.

Kegunaan dari masing-masing bit yang terlihat pada Gambar 2.19 :

- Bit 7 (I)

  Merupakan bit *Global Interrupt Enable*. Bit ini harus diset supaya semua perintah interupsi dapat dijalankan. Untuk fungsi interupsi individual akan dijelaskan pada bagian yang lain. Jika bit ini di*reset*, maka semua perintah interupsi baik yang individual maupun yang secara umum akan diabaikan. Bit ini akan dibersihkan atau cleared oleh *hardware* setelah sebuah interupsi dijalankan dan akan diset kembali oleh perintah RETI. Bit ini juga dapat diset dan direset melalui aplikasi dengan instruksi SEI dan CLI.

- Bit 6 (T)

  Merupakan bit *copy storage*. Instruksi bit *Copy Instructions Bit LoaD* (BLD) *and Bit STore* (BST) menggunakan bit ini sebagai asal atau tujuan untuk bit yang telah dioperasikan. Sebuah bit dari sebuah register dalam register file dapat disalin ke dalam bit ini dengan menggunakan instruksi BST, dan sebuah bit di dalam bit ini dapat disalin ke dalam sebuah bit di dalam register pada register file dengan menggunakan perintah BLD.

- Bit 5 (H)

  Merupakan bit *Half Carry Flag*. Bit ini menandakan sebuah *Half Carry* dalam beberapa operasi aritmatika. Bit ini berfungsi dalam aritmatik BCD.

- Bit 4 (S)

  Merupakan *sign bit*.

- Bit 3 (V)

  Merupakan bit *Two's Complement Overflow Flag*. Bit ini menyediakan fungsi aritmatika dua komplemen.

- Bit 2 (N)

  Merupakan bit *Negative Flag*. Bit ini mengindikasikan sebuah hasil negatif di dalam sebuah fungsi logika atau aritmatika.

- Bit 1 (Z)

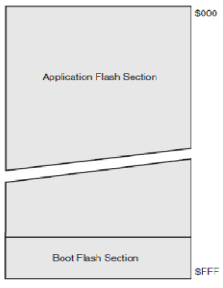
  Merupakan bit *Zero Flag*. Bit ini mengindikasikan sebuah hasil nol "0" dalam sebuah fungsi aritmatika atau logika.

- Bit 0 (C)

  Merupakan bit *Carry Flag*. Bit ini mengindikasikan sebuah *carry* atau sisa dalam sebuah fungsi aritmatika atau logika.

2.3.3 Memori Program ATmega8

Memori program yang terletak pada *Flash Perom* tersusun dalam *word* atau 2 byte karena setiap instruksi memiliki lebar 16 bit atau 32 bit. AVR ATmega8 memiliki 4KByte x 16 Bit *Flash Perom* dengan alamat mulai dari $000 sampai $FFF. AVR tersebut memiliki 12 bit *Program Counter* (PC) sehingga mampu mengalamati isi flash.

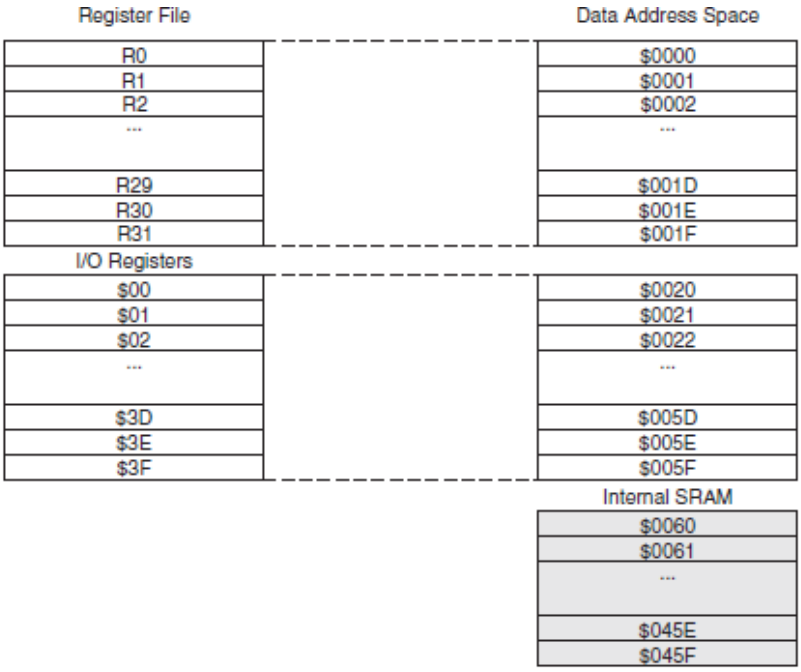


Gambar 2.20 Memori program AVR ATmega8

2.3.4 SRAM Data Memori

ATmega8 memiliki ruang pengalamatan memori data dan memori program yang terpisah. Memori data terbagi menjadi 3 bagian yaitu: 32 buah register umum, 64 buah register I/O, dan 512 byte SRAM internal.

Register untuk keperluan umum menempati *space* data pada alamat terbawah yaitu $00 sampai $1F. Sementara itu register khusus untuk menangani I/O dan kontrol terhadap mikrokontroller menempati 64 alamat berikutnya, yaitu mulai dari $20 sampai $5F. Register tersebut merupakan register yang khusus digunakan untuk mengatur fungsi terhadap berbagai peripheral mikrokontroller, seperti kontrol *register, timer/counter*, fungsi-fungsi I/O, dan sebagainya. Register khusus alamat memori secara lengkap dapat dilihat pada gambar dibawah ini. Alamat memori berikutnya digunakan untuk SRAM 512 byte, yaitu pada lokasi $60 sampai dengan $25F.



Gambar 2.21 Memori data AVR ATmega8

2.4 Motor Servo

Motor servo adalah motor yang mampu bekerja dua arah (CW dan CCW) dimana arah dan sudut pergerakan rotornya dapat dikendalikan hanya dengan memberikan pengaturan *duty cycle* sinyal PWM pada bagian pin kontrolnya. Motor servo adalah jenis motor yang digunakan sebagai penggerak pada sistem servo seperti pada penggerak pada kontrol posisi lengan robot. Motor servo terdiri dari motor DC, *gear box*, dan *driver control* yang terpadu menjadi satu.



Gambar 2.22 Motor servo

2.4.1 Jenis-jenis Motor Servo

Di pasaran ada berbagai jenis atau tipe servo, namun berdasarkan dari putaran sudutnya yang umum dijumpai sebagai berikut:

1. Motor servo standar 180˚

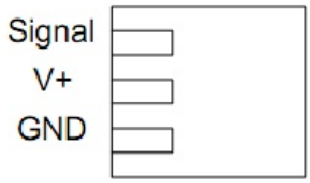
   Motor servo jenis ini hanya mampu bergerak dua arah (CW dan CCW) dengan defleksi masing-masing sudut mencapai 90° sehingga total defleksi sudut dari kanan, tengah, dan kiri adalah180°.

2. Motor servo 360˚ (*continues rotation*)

   Motor servo jenis ini mampu bergerak dua arah (CW dan CCW) tanpa batasan defleksi sudut putar (dapat berputar secara kontinyu).

Sedangkan berdasarkan dari tipe *signal* yang digunakan, terdapat servo analog dan servo digital. Dalam tugas akhir ini, motor servo yang digunakan

adalah motor servo analog dengan putaran sudut 180˚. Yang perlu diperhatikan pada bagian output terdapat 3 kabel, yang masing-masing berfungsi sebagai:



Gambar 2.23 Pin out pada motor servo

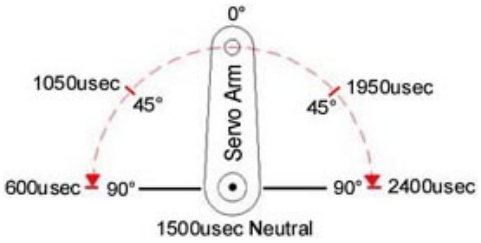
a. Input data PWM (Signal)

b. Tegangan 5 - 6V

c. Ground

Dan warna dari masing-masing kabel bergantung pada merek servo tersebut. Untuk menentukan posisi kabel *signal* adalah dengan mengingat kabel merah adalah (+), kabel hitam adalah (-), warna lain selain merah dan hitam adalah kabel *signal*.

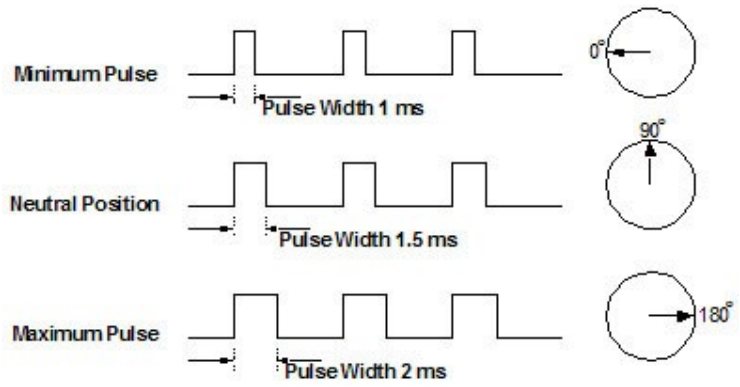
Untuk dapat melakukan *controling* pada servo, kabel *signal* di sambung langsung pada salah satu port *microcontroller* dan diset sebagai output. kemudian servo diberi suplay 5 - 6V. Sedangkan nilai sinyal yang dikirim dan sudut yang di hasilkan terlihat seperti berikut:



Gambar 2.24 Sudut pada motor servo

Motor Servo akan bekerja dengan baik jika pada bagian pin kontrolnya diberikan sinyal PWM dengan frekuensi 50Hz. Dimana pada saat sinyal dengan frekuensi 50Hz tersebut dicapai, dengan *ton duty cycle* 1,5ms rotor dari motor

akan berhenti tepat ditengah-tengah (sudut 90°). Pada saat *Ton duty cycle* yang diberikan kurang dari 1,5ms, maka rotor akan berputar kearah kiri dengan membentuk sudut yang besarnya linier terhadap besarnya *Ton duty cycle* dan akan bertahan diposisi tersebut (sudut 0˚). Dan sebaliknya, jika *ton duty cycle* yang diberikan lebih dari 1,5ms, maka rotor akan berputar kearah kanan dengan membentuk sudut yang linier pula terhadap besarnya *Ton duty cycle* dan bertahan diposisi tersebut (sudut 180˚). Penjelasan ini dapat terlihat pada Gambar 2.25.



Gambar 2.25 Pensinyalan motor servo

Untuk motor servo berbeda dengan motor DC dan Stepper. Pada motor DC dan stepper rangkaiannya searah tanpa ada feedback. Sedangkan dalam motor servo di gunakan sistem umpan balik. Servo sendiri merupakan suatu motor yang didesain dengan sistem *feedback* dimana posisi dari motor akan diinformasikan kembali ke dalam servo tersebut.

2.4.2 Prinsip Kerja Motor Servo

Prinsip kerja motor didasarkan pada peletakan suatu konduktor dalam suatu medan magnet. Jika suatu konduktor dililitkan dengan kawat berarus maka akan dibangkitkan medan magnet berputar. Kontribusi dari setiap putaran akan

merubah intensitas medan magnet yang ada dalam bidang yang tertutup kumparan. Dengan cara inilah medan magnet yang kuat terbentuk. Tenaga yang digunakan untuk mendorong fluks magnet tersebut disebut *Manetomotive Force* (MMF).

Fluks magnet digunakan untuk mengetahui seberapa banyak fluks pada daerah disekitar koil atau magnet permanen. Medan magnet pada motor DC servo dibangkitkan oleh magnet permanen, jadi tidak perlu tenaga untuk membuat medan magnet. Fluks madan magnet pada stator tidak dipengaruhi oleh arus *armature*. Oleh karena itu, kurva perbandingan antara kecepatan dengan torsi adalah linier.

Pada prinsipnya jika sebuah penghantar dilalui arus listrik *I* maka akan menghasilkan medan magnet disekelilingnya. Kemudian jika penghantar ini ditempatkan dalam induksi magnetik *B*, akan memperoleh gaya *FB*. Besarnya gaya yang ditimbulkan sebanding dengan arus listrik *Ia* dan panjang penghantar *L* yang memotong induksi magnetik *B*. Atau biasa dinyatakan dengan persamaan, induksi magnetik:

$$F\ B\ = B\ .I\ .L$$

Motor servo biasanya hanya bergerak mencapai sudut tertentu saja dan tidak kontinyu seperti motor DC maupun motor stepper. Walau demikian, untuk beberapa keperluan tertentu, motor servo dapat dimodifikasi agar bergerak kontinyu.

2.5 Potensiometer

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

Potensiometer jarang digunakan untuk mengendalikan daya tinggi (lebih dari 1 Watt) secara langsung. Potensiometer digunakan untuk menyetel taraf isyarat analog (misalnya pengendali suara pada peranti audio), dan sebagai pengendali masukan untuk sirkuit elektronik. Sebagai contoh, sebuah peredup lampu menggunakan potensiometer untuk menendalikan pensaklaran sebuah TRIAC, jadi secara tidak langsung mengendalikan kecerahan lampu.

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



Gambar 2.26 Potensiometer satu putaran secara umum

2.5.1 Konstruksi Potensiometer

Sebuah potensiometer biasanya dibuat dari sebuah unsur resistif semi lingkar dengan sambungan geser (penyapu). Unsur resistif, dengan terminal pada salah satu ataupun kedua ujungnya, berbentuk datar atau menyudut, dan biasanya dibuat dari grafit. Untuk potensiometer putaran tunggal, penyapu biasanya bergerak kurang dari satu putaran penuh sepanjang kontak.

Potensiometer putaran ganda, elemen resistifnya berupa pilinan dan penyapu bergerak 10, 20, atau lebih banyak putaran untuk menyelesaikan siklus. Dibandingkan putaran tunggal, potensiometer putaran ganda biasanya murah karena dibuat dari unsur resistif konvensional yang sama dengan resistor putaran tunggal, sedangkan penyapu digerakkan melalui gir cacing. Disamping grafit, bahan yang digunakan untuk membuat unsur resistif adalah kawat resistansi, plastik partikel karbon dan campuran keramik-logam yang disebut cermet.

2.5.2 Jenis-jenis Potensiometer

1. Potensiometer Logaritmik.

   Potensiometer logaritmik mempunyai unsur resistif yang semakin menyempit atau dibuat dari bahan yang memiliki resistivitas bervariasi. Ini memberikan piranti yang resistansinya merupakan fungsi logaritmik terhadap sudut poros potensiometer.

   Sebagian besar potensiometer *log* (terutama yang murah) sebenarnya tidak benar-benar logaritmik, tetapi menggunakan dua jalur resistif linier untuk meniru hukum logaritma. Potensiometer log juga dapat dibuat dengan menggunakan potensiometer linier dan resistor

eksternal. Potensiometer yang benar-benar logaritmik relatif sangat mahal. Potensiometer logaritmik sering digunakan pada peranti audio, terutama sebagai pengendali volume.

2. Potensiometer Linier.

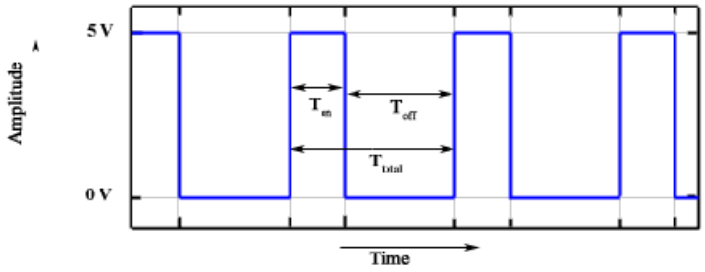
Potensiometer linier mempunyai unsur resistif dengan penampang konstan, menghasilkan peranti dengan resistansi antara penyapu dengan salah satu terminal *proportional* dengan jarak antara keduanya. Potensiometer linier digunakan jika relasi *proportional* diinginkan antara putaran sumbu dengan rasio pembagian dari potensiometer, misalnya pengendali yang digunakan untuk menyetel titik pusat layar osiloskop.



Gambar 2.27 Potensiometer Linier

2.6 *Pulse Width Modulation* (PWM)

Secara umum PWM adalah sebuah cara memanipulasi lebar sinyal atau tegangan yang dinyatakan dengan pulsa dalam suatu perioda, yang akan digunakan untuk mentransfer data pada telekomunikasi ataupun mengatur tegangan sumber yang konstan untuk mendapatkan tegangan rata-rata yang berbeda. Penggunaan PWM sangat banyak, mulai dari pemodulasian data untuk telekomunikasi, pengontrolan daya atau tegangan yang masuk ke beban, regulator tegangan, *audio effect* dan penguatan, serta aplikasi-aplikasi lainnya.



Gambar 2.28 Sinyal PWM

Terlihat pada gambar, bahwa sinyal PWM adalah sinyal digital yang amplitudonya tetap, namun lebar pulsa yang aktif (*duty cycle*) per periodenya dapat diubah-ubah. Dimana periodenya adalah waktu pulsa *high* (1) T*on* ditambah waktu pulsa low (0) T*off*.

$$T\ t\ o\ t\ a\ l\ = T\ o\ n\ + T\ o\ f\ f$$

Duty cycle adalah lamanya pulsa *high* (1) T*on* dalam satu perioda. Jika f(t) adalah sinyal PWM, maka besar *duty cycle*-nya adalah:

$$\bar{y} = \frac{1}{T} \int_0^T f(t) d\ t$$

atau bisa diulis dengan

$$D = \frac{T\ o\ n}{(T\ o\ n\ + T\ o\ f\ f} = \frac{T\ o\ n}{T)t\ o\ t\ a\ l}$$

sehingga

$$V\ o\ u\ t\ = D\ x\ V\ i\ n$$

$$V\ o\ u\ t = \frac{T\ o\ n}{T\ t\ o\ t\ a\ l}\ x\ V\ i\ n$$

Grafik dibawah ini, menggambarkan beberapa PWM dalam *duty cycle* yang berbeda.



(a)



(b)



(c)

Gambar 2.29 (a), (b), dan (c) beberapa PWM dalam *duty cycle* yang berbeda

Pada Gambar 2.29 (a) terlihat bahwa sinyal *high* per periodenya, sangat kecil (hanya 10%). Pada Gambar 2.29 (b) terlihat sinyal *high*-nya hampir sama dengan sinyal low (50%). Dan pada Gambar 2.29 (c) terlihat bahwa sinyal *high*-nya lebih besar dari sinyal *low*-nya (90%).

Maka jika dimisalkan tegangan input yang melalui rangkaian tersebut sebesar 10V. Maka jika digunakan PWM pada Gambar 2.29 (a), nilai tegangan output rata ratanya sebesar 1V (10% dari V*source*), jika digunakan PWM Gambar 2.29 (b), maka tegangan output rata-ratanya sebesar 5V (50%). Begitu pula jika menggunakan PWM Gambar 2.29 (c), maka tegangan output rata-ratanya sebesar 9V (90%).

2.7 Bahasa Pemrograman

Dewasa ini penggunanaan bahasa pemrograman tingkat tinggi (seperti C, Basic, Pascal, Forth dan sebagainya) semakin popular dan banyak digunakan untuk memprogram sistem mikrokontroller. Berdasarkan sifatnya yang sangat *fleksibel* dalam hal kelulusan pemrogram untuk mengakses perangkat keras, bahasa C merupakan bahasa pemrograman yang paling cocok dibandingkan bahsa-bahasa pemrograman tinggi lainnya.

Dikembangkan pertama kali oleh Dennis Ritchie dan Ken Thomson pada tahun 1972, bahasa C merupakan salah satu pemrograman yang paling populer untuk pengembangan program-program aplikasi yang berjalan pada system *microprocessor* (komputer). Karena kepopulerannya, vendor-vendor perangkat lunak kemudian mengembangkan compiler C sehingga menjadi beberapa varian berikut: Turbo C, Borland C, Microsoft C, Power C, Zortech C dan lain sebagainya. Untuk menjaga portabilitas, *compiler-compiler* C tersebut menerapkan ANSI C *American National Standards Institute* (ANSI) sebagai standar bakunya. Perbedaan antara *compiler-compiler* tersebut umumnya hanya terletak pada pengembangan fungsi-fungsi *library* serta fasilitas *Integrated Development Environment* (IDE) saja.

Dibandingkan dengan bahasa assembler, penggunaan bahsa C dalam pemrograman memiliki beberapa kelebihan berikut, memepercepat waktu pengembangan, bersifat modular dan terstriktur, sedangkan kelemahannya adalah kode program hasil kompilasi akan relative lebih besar (dan sebagai konsekuensinya hal ini terkadang akan mengurangi kecepatan eksekusi). Khusus pada mikrokontroller AVR, untuk mereduksi konsekuensi negatif diatas,

perusahaan atmel merancang sedemikian sehingga arsitektur AVR ini efisien dalam mendekode serta mengeksekusi instruksi-instruksi yang umum dibangkitkan oleh *compiler* C. Dalam kenyataannya, pengembangan arsitektur AVR ini tidak dilakukan sendiri oleh perusahaan atmel tetapi ada kerja sama dengan salah satu vendor pemasok *compiler* C untuk mikrokontroller tersebut, yaitu IAR C *software* atau pemrograman. Bagian tersebut digunakan dalam pengembangan robot dengan komponen *Programmable Logic Control* (PLC).

Komponen *programmable* adalah komponen yang terbuat dari bahan semikonduktor dan merupakan integrasi dari berbagai komponen elektronika didalamnya dan dapat deprogram ulang demi tujuan pengendalian robot. Komponen *programmable* digunakan oleh robot-robot yang dibuat untuk melakukan tugas-tugas beragam dan biasanya bekerja secara otomatis. Dengan demikian, robot-robot yang bersifat *unprogrammable* (sederhana), yang bisa bergerak hanya dengan rangkaian elektronika analog biasa, tidak memerlukan program atau *software* tersebut. Adapun instruksi-instruksi pada bahasa C di CodeVision AVR:

1. Header File

   Adalah berkas yang berisi *prototype* fungsi definisi konstanta dan definisi *variable*. Fungsi adalah kumpulan *code* C yang diberi nama dan ketika nama tersebut dipanggil maka kumpulan kode tersebut dijalankan.

   Contoh:

   stdio.h

   math.h

2. *#include*

   Preprosesor directive adalah bagian yang berisi pengikutsertaan file atau berkasberkas fungsi maupun pendefinisian konstanta.

   Contoh:

   #include <stdio.h>

   #include phi 3.14

3. Void

   artinya fungsi yang mengikutinya tidak memiliki nilai kembalian *(return)*.

4. Main ( )

   Fungsi main ( ) adalah fungsi yang pertama kali dijalankan ketika program dieksekusi tanpa fungsi main suatu program tidak dapat dieksekusi namun dapat dikompilasi.

5. Statement

   Statement adalah instruksi atau perintah kepada suatu program ketika program itu dieksekusi untuk menjalankan suatu aksi. Setiap statement diakhiri dengan titik-koma (;).

6. Tipe Data Dasar

   Tipe data yang dapat digunakan pada software code vision AVR

Tabel 2.2 Tipe Data

Tipe Data	Ukuran (Byte)	Range	Keterangan
char	1	-128 s/d +127	Karakter/string
Int	2	-32768 s/d +32127	Integer/bilangan bulat

Float	4	3.4E-38 s/d 3.4E38	Float/pecahan
Long	4	-2.147.438.648 s/d +2.147.438.647	-
Double	8	1.7E-308 s/d 1.7E308	Pecahan presisi ganda
Long Double	10	3.4E-4932 s/d 3.4E4932	-

7. Operator

Tabel 2.3 Daftar Operator Aritmetika

Operator	Deskripsi
+	Penjumlahan (*add*)
-	Pengurangan (*substract*)
*	Perkalian (*multiply*)
/	Pembagian (*divide*)
%	Sisa pembagian integer (modulus)
~	Negasi (*negate*)

Tabel 2.4 Daftar Operator Kondisi

Operator	Keterangan
==	Sama dengan (bukan assignment)
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan

<=	Lebih kecil atau sama dengan

Tabel 2.5 Daftar Operator Logika

Opearator	Keterangan
&&	Logic AND
||	Logic OR
!	Logic NOT

Tabel 2.6 Daftar Operator Bitwise

Operator	Keterangan
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT

8. Pernyataan *if*

   Sebuah pernyataan yang dapat dipakai untuk mengambil keputusan berdasarkan suatu kondisi. Bentuk pernyataan ini ada dua macam:

   - *If* saja
   - *Else*

2.8 RS232

Nama resmi dari *standart interface* ini adalah *interface between data terminal equipment and data communication employing serial binary data*

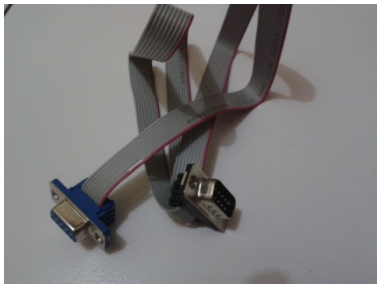
interchange, yaitu suatu terminal yang menghubungkan antara terminal data dari suatu peralatan dan peralatan komunikasi data yang yang menjalankan pertukaran data biner secara serial, oleh industri komunikasi data disebut RS-232. Sejak dipublikasikan pertama kali pada tahun 1962 , *interface* ini telah mengalami beberapa revisi dan sekarang yang banyak dipergunakan adalah RS-232 revisi kelima yang digunakan pada tahun 1991 dan disebut RS-232E atau EIA-232-E. Awalan "RS" pada RS-232 merupakan singkatan dari *recommended standart* yang berarti standar yang dianjurkan, karena selama ini dalam publikasi *Electronic Industries Association* (EIA) tidak pernah memiliki ketetapan hukum yang mengharuskan untuk digunakan dalam dunia komunikasi data. Ada beberapa standar sederhana yang dikembangkan sendiri oleh pembuat peralatan elektronik yang mungkin dapat diikuti, agar secara umum ada kecocokan. Standar lain yang menyangkut referensi aspek fungsi dan prosedur *interface* secara mekanikal dan elektrikal yang dipublikasikan oleh ITU-T pada tahun 1993 adalah standar V.25 dan V.28, sehingga RS-232E sebenarnya mencakup empat aspek, yaitu :

a. Mechanical : ISO21110
b. Electrical : V.28
c. Functional : V.24
d. Prosedural : V.24

Terdapat beberapa macam cara untuk menerapkan *interface* data biner pada komunikasi secara serial, salah satunya adalah RS-232 yang merupakan salah satu dari standart yang dipilih dan sekarang telah dipakai secara luas dan dalam komunikasi data umumnya digunakan untuk menghubungkan *Data Terminal Equipment* (DTE) ke *Data Communication Equipment* (DCE) yang berupa

peralatan sistem komunikasi analog. Protokol standar yang mengatur komunikasi melalui *serial port* disebut *Recommended Standard-232* (RS-232) yang dikembangkan oleh EIA.

Interfacing RS-232 menggunakan komunikasi *asyncronous* di mana sinyal *clock* tidak dikirimkan bersamaan dengan data. Setiap *word* data disinkronisasikan menggunakan sebuah *start* bit dan sebuah *stop* bit. Jadi, sebuah *frame* data terdiri dari sebuah *start* bit, diikuti bit-bit data dan diakhiri dengan *stop* bit. Jumlah bit data yang digunakan dalam komunikasi serial adalah 8 bit.



Gambar 2.30 Konektor RS232 atau DB9

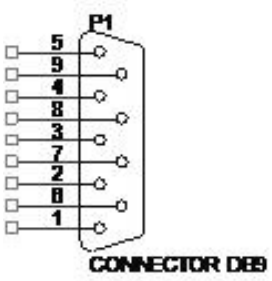
2.8.1 Karakteristik Sinyal Serial Port

Standar sinyal serial RS232 memiliki ketentuan level tegangan sebagai berikut:

1. Logika ‘1’ disebut ‘*mark*’ terletak antara -3 volt hingga -25 volt.
2. Logika ‘0’ disebut ‘*space*’ terletak antara +3 volt hingga +25 volt.
3. Daerah tegangan antara -3 volt hingga +3 volt adalah invalid level, yaitu daerah tegangan yang tidak memiliki level logika pasti sehingga harus dihindari. Demikian juga level tegangan lebih negatif dari -25 volt atau lebih positif dari +25 volt juga harus dihindari karena dapat merusak *line driver* pada saluran RS232.

2.8.2 Konfigurasi Serial Port

Gambar 2.31 adalah gambar konektor port serial DB9. Pada komputer IBM PC *Compatibel* biasanya kita dapat menemukan dua konektor DB9 yang biasanya dinamakan COM1 dan COM2.



Gambar 2.31 Konektor port serial DB9

Tabel 2.7 Konfigurasi pin dan nama sinyal konektor serial DB9

No. Pin	Nama Sinyal	Direction	Keterangan
1	DCD	In	*Data Carrier Detect/Received Line Signal Detect*
2	RxD	In	*Receive Data*
3	TxD	Out	*Transmit Data*
4	DTR	Out	*Data Terminal Ready*
5	GND	-	*Ground*
6	DSR	In	*Data Set Ready*
7	RTS	Out	*Request To Send*
8	CTS	In	*Clear To Send*
9	RI	In	*Ring Indicator*

Keterangan mengenai fungsi saluran RS232 pada konektor DB9 adalah sebagai berikut:

- *Receive Line Signal Detect*, dengan saluran ini DCE memberitahukan ke DTE bahwa pada terminal masukkan ada data masuk.
- *Receive Data*, digunakan DTE menerima data dari DCE.
- *Transmit Data*, digunakan DTE mengirimkan data ke DCE.
- *Data Terminal Ready*, pada saluran ini DTE memberitahukan kesiapan terminalnya.
- *Signal Ground*, saluran ground.
- *Ring Indicator*, pada saluran ini DCE memberitahukan ke DTE bahwa sebuah stasiun menghendaki berhubungan dengannya.
- *Clear To Send*, dengan saluran ini DCE memberitahukan bahwa DTE boleh mulai mengirim data.
- *Request To Send*, dengan saluran ini DCE diminta mengirim data oleh DTE.
- *DCE Ready*, sinyal aktif pada saluran ini menunjukkan bahwa DCE sudah siap.