BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Robot

Robot adalah sebuah alat mekanik yang dapat melakukan tugas fisik, baik menggunakan pengawasan dan kontrol manusia, ataupun menggunakan program yang telah didefinisikan terlebih dahulu (kecerdasan buatan). Istilah robot berawal dari bahasa Cheko *"robota"* yang berarti pekerja atau kuli yang tidak mengenal lelah atau bosan. Robot biasanya digunakan untuk tugas yang berat, berbahaya, pekerjaan yang berulang dan kotor. Biasanya kebanyakan robot industri digunakan dalam bidang produksi. Penggunaan robot lainnya termasuk untuk pembersihan limbah beracun, penjelajahan bawah air dan luar angkasa, pertambangan, pekerjaan "cari dan tolong" (*search and rescue*), dan untuk pencarian tambang. Belakangan ini robot mulai memasuki pasaran konsumen di bidang hiburan, dan alat pembantu rumah tangga.

Saat ini hampir tidak ada orang yang tidak mengenal robot, namun pengertian robot tidaklah dipahami secara sama oleh setiap orang. Sebagian membayangkan robot adalah suatu mesin tiruan manusia (*humanoid*), meski demikian *humanoid* bukanlah satu-satunya jenis robot, jenis robot yang lain yaitu robot *mobile*, robot *manipulator* (tangan), robot berkaki, robot *flying*, robot *underwater*.

Pada kamus *Webster* pengertian robot adalah: *An automatic device that performs function ordinarily ascribed to human beings* (sebuah alat otomatis yang melakukan fungsi berdasarkan kebutuhan manusia). Dari kamus *Oxford* diperoleh pengertian robot adalah: *A machine capable of carrying out a complex series of actions automatically, especially one programmed by a computer.* (Sebuah mesin yang mampu melakukan serangkaian tugas rumit secara otomatis, terutama yang diprogram oleh komputer).

Pengertian dari *Webster* mengacu pada pemahaman banyak orang bahwa robot melakukan tugas manusia, sedangkan pengertian dari *Oxford* lebih umum, beberapa organisasi di bidang robot membuat definisi tersendiri. *Robot Institute of America* memberikan definisi robot sebagai: *A reprogammable multifunctional*

manipulator designed to move materials, parts, tools or other specialized devices through variable programmed motions for the performance of a variety of tasks (Sebuah *manipulator* multifungsi yang mampu diprogram, didesain untuk memindahkan material, komponen, alat, atau benda khusus lainnya melalui serangkaian gerakan terprogram untuk melakukan berbagai tugas)

International Organization for Standardization (ISO 8373) mendefinisikan robot sebagai: *An automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either Fixed in place or mobile for use in industrial automation applications* (Sebuah *manipulator* yang terkendali, multifungsi, dan mampu diprogram untuk bergerak dalam tiga aksis atau lebih, yang tetap berada di tempat atau bergerak untuk digunakan dalam aplikasi otomasi industri). Dari beberapa definisi di atas, kata kunci yang dapat menerangkan pengertian robot adalah:

1. Dapat memperoleh informasi dari lingkungan (melalui sensor).
2. Dapat diprogram.
3. Dapat melaksanakan beberapa tugas yang berbeda.
4. Bekerja secara otomatis
5. Cerdas (*intelligent*)
6. Digunakan di industri. (*Membuat Robot Itu Gampang,* hal 5, 2006)

(*Sumber: http://eprints.polsri.ac.id/2780/2/BAB%20II.pdf*)

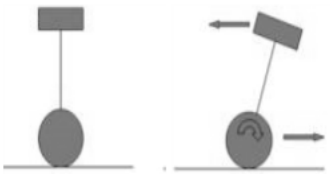
2.2 Robot Keseimbangan (*Balancing Robot*)

Robot keseimbangan (*balancing robot*) beroda dua merupakan suatu robot *mobile* yang memiliki dua buah roda disisi kanan dan kirinya yang tidak akan seimbang apabila tanpa adanya kontroler. Robot keseimbangan (*balancing robot*) ini merupakan pengembangan dari model pendulum terbalik (*inverted pendulum*).

Pendulum terbalik (*inverted pendulum*) adalah pendulum yang terengsel ke kereta beroda yang dapat bergerak maju dan mundur pada bidang horizontal di sepanjang lintasan. Penerapan konsep pendulum terbalik (*inverted pendulum*) dalam dunia robotika bisa dilihat pada robot keseimbangan (*balancing robot*), yaitu robot dengan dua roda. Roda tersebut diasumsikan sebagai kereta beroda dan badan robot

diasumsikan sebagai pendulum. Sistem ini tidak stabil karena ketika kereta beroda diberi gangguan dari luar maka pendulum akan jatuh. Oleh karena itu, untuk mempertahankan agar pendulum atau badan robot tidak jatuh dibutuhkan suatu kendali khusus (Stang, 2005).

Berikut adalah gambar dari pendulum terbalik (*inverted pendulum*) ditunjukkan pada gambar 2.1

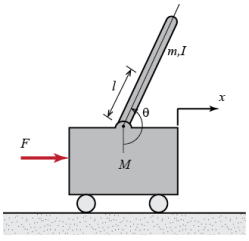


Gambar 2.1 Gambar Pendulum terbalik

(Sumber: https://core.ac.uk/download/pdf/11731567.pdf)

Pendulum terbalik pada dasarnya hanya suatu batang stik 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 $\theta = P\ i$ atau dengan kata lain, pendulum diasumsikan tidak bergerak lebih dari beberapa derajat dari sumbu vertikal yang berada pada sudut θ .



Gambar 2.2 Desain pendulum terbalik

(Sumber:http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=System Modeling)

Keterangan:

M = Massa kereta (0,5 Kg)

m = Massa pendulum (0,2 Kg)

b = Gesekan kereta (0,1 N/m/sec)

l = Panjang pendulum pusat massa (0,3 m)

I = Inersia pendulum (0,006 Kg.m^2)

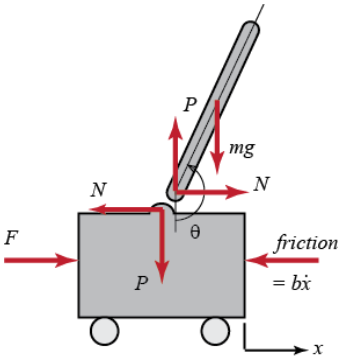
F = Gaya yang diberikan pada kereta

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).

2.2.1 Analisis Gaya dan Sistem Persamaan



Gambar 2.3 diagram sistem kereta pendulum

(Sumber:http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=System Modeling)

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

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

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

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

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

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \quad \text{(2.3)}$$

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

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \quad \text{(2.4)}$$

Untuk menghilangkan P dan N dalam persamaan (2.4) untuk mendapatkan persamaan berikut maka keliling pusat massa pendulum dijumlahkan.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \quad \text{(2.5)}$$

Menggabungkan persamaan (2.4) dan (2.5) maka akan diperoleh persamaan dinamis

$$(I+ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \quad \text{(2.6)}$$

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

$$(I+ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \quad \text{(2.7)}$$

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \quad \text{(2.8)}$$

2.2.2 Fungsi Transfer

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

$$(I+ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \quad \text{(2.9)}$$

$$(M+m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s) \quad \text{(2.10)}$$

Catatan:

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

Untuk mencari sudut $\hat{}$ sebagai output (keluaran) maka persamaan pertama untuk X(s):

$$X(s) = \left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s) \quad \text{.....(2.11)}$$

kemudian persamaan (2.11) disubtitusi ke persamaan (2.10)

$$(M+m)\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s) \quad \text{.....(2.12)}$$

Dimana

$$q = [(M+m)(I+ml^2) - (ml)^2] \quad \text{.....(2.13)}$$

Fungsi transfernya adalah:

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \quad \text{.....(2.14)}$$

2.3 Aplikasi Robot Keseimbangan (*Balancing Robot*)

Dalam aplikasinya, Robot Keseimbangan (*Balancing Robot*) dapat digunakan untuk teknologi *Single Human Transporter*. *Single human transporter* adalah kendaraan yang beroperasi menggunakan konsep *balancing* robot atau populer disebut dengan *segway*. Berikut adalah gambar dari *segway* ditunjukkan pada gambar 2.4

Gambar 2.4 *Segway*

(Sumber: digilib.mercubuana.ac.id/manager/n!@file_skripsi/Isi2424390255394.pdf)

Model alat transportasi ini akan berkembang pada masa depan dengan tidak menggunakan elemen bahan bakar. Berdiri stabil diatas dua roda merupakan sistem transportasi yang menggunakan konsep *balancing* robot.

Contoh lain terdapat pada robot penelitian bawah laut, seperti pada gambar 2.5

Gambar 2.5 Robot Penelitian Bawah Laut

(Sumber: digilib.mercubuana.ac.id/manager/n!@file_skripsi/Isi2424390255394.pdf)

Robot tersebut bekerja untuk menunjang penelitian bawah laut yang membawa kamera, sonar, magnetometer, lengan pemotong, contoh air, mengukur intesitas kelembapan, cahaya dan suhu pada air laut.

Sumber: digilib.mercubuana.ac.id/manager/n!@file_skripsi/Isi2424390255394.pdf

2.4 Modul MPU-6050

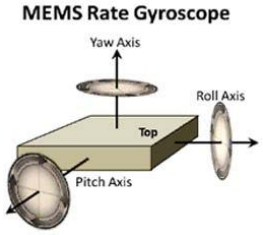
Modul MPU-6050 adalah sebuah modul yang pertama mengintegrasikan 6 *axis* alat pendeteksian gerak yang dikombinasikan dari 3 *axis* sensor *gyroscope* dan 3 *axis* sensor *accelerometer*. Modul ini sangat akurat dengan fasilitas *hardware* internal 16 bit ADC untuk setiap kanalnya. Modul ini akan menangkap nilai kanal axis X, Y dan Z bersamaan dalam satu waktu. Dengan penambahan regulator tegangan dan beberapa komponen pelengkap lainnya membuat modul ini siap dipakai dengan tegangan supply sebesar 3,3VDC dan 5VDC. Modul ini memiliki i*nterface* I2C dengan menyambungkan pin SDA (Serial Data) dan SCL (Serial Clock) pada pin mikrokontroller. Berikut adalah gambar dari sensor modul MPU-6050 ditunjukkan pada gambar 2.6

Gambar 2.6 Sensor Modul MPU-6050

(Sumber: Datasheet Sensor MPU6050)

2.4.1 Sensor *Gyroscope*

Gyroscope adalah alat yang berguna untuk menentukan orientasi gerak yang berputar pada poros sumbu. *Gyroscope* memiliki *output* terhadap kecepatan sudut dari arah sumbu x yang nantinya akan menjadi sudut *phi (roll),* dari sumbu y nantinya menjadi sudut *theta (pitch),* dan sumbu z nantinya menjadi sudut *psi* (yaw). Berikut adalah gambar dari sumbu sensor *gyroscope* ditunjukkan pada gambar 2.7



Gambar 2.7 Sumbu Sensor *Gyroscope*

(Sumber: digilib.mercubuana.ac.id/manager/n!@file_skripsi/Isi2424390255394.pdf)

Gyroscope adalah perangkat untuk mengukur atau mempertahankan orientasi, dengan prinsip ketetapan momentum sudut. Mekanismenya adalah sebuah roda berputar dengan piringan didalamya yang tetap stabil. *Gyroscope* sering digunakan pada robot atau heli dan alat-alat cangggih lainnya. *Gyroscope* berupa sensor untuk menentukan orientasi gerak dengan bertumpu pada roda atau cakram yang berotasi dengan cepat pada sumbu.

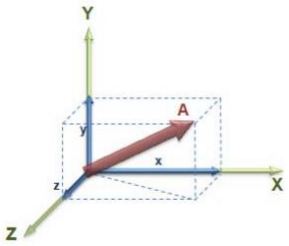
Prinsip kerja dari *Gyroscope* ini adalah pada saat *Gyroscope* berotasi maka *Gyroscope* akan memiliki nilai keluaran. Apabila *Gyroscope* berotasi searah dengan jarum jam pada sumbu Z maka tegangan ouput yang dihasilkan akan mengecil sedangkan jika *Gyroscope* berotasi berlawan arah dengan jarum jam pada sumbu Z maka tegangan *output* yang dihasilkan akan membesar. Pada saat *Gyroscope* tidak

sedang berotasi atau berada pada keadaan diam maka tegangan *output*nya akan sesuai dengan nilai *offset gyrosensor* tersebut. Nilai keluaran pada sensor diubah menjadi radian/second (rad/s) lalu diubah kembali menjadi degree/second (deg/s).

Sumber: digilib.mercubuana.ac.id/manager/n!@file_skripsi/Isi2424390255394.pdf

2.4.2 Sensor *Accelerometer*

Accelerometer adalah sebuah tranduser yang berfungsi untuk mengukur percepatan, mendeteksi dan mengukur getaran, ataupun untuk mengukur percepatan akibat gravitasi bumi. *Accelerometer* juga dapat digunakan untuk mengukur getaran yang terjadi pada kendaraan, bangunan, mesin, dan juga bisa digunakan untuk mengukur getaran yang terjadi di dalam bumi, getaran mesin, jarak yang dinamis, dan kecepatan dengan ataupun tanpa pengaruh gravitasi bumi. Berikut adalah gambar dari prinsip kerja *accelerometer* ditunjukkan pada gambar 2.8



Gambar 2.8 Prinsip Kerja *Accelerometer*

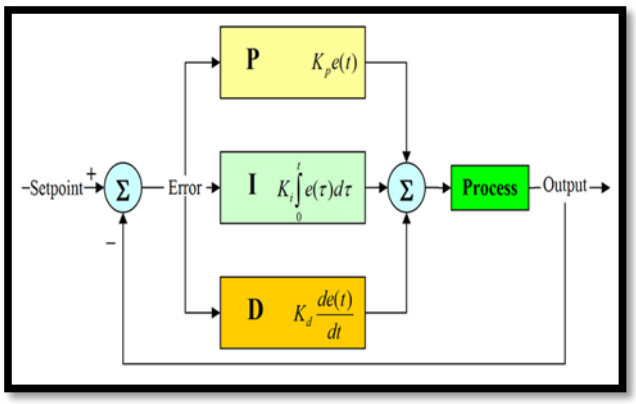
(Sumber:http://repository.usu.ac.id/bitstream/123456789/47742/3/Chapter%20II.pdf)

Prinsip kerja dari *accelerometer* ini berdasarkan hukum fisika bahwa apabila suatu konduktor digerakkan melalui suatu medan magnet, atau jika suatu medan magnet digerakkan melalui suatu konduktor, maka akan timbul suatu tegangan induksi pada konduktor tersebut. *Accelerometer* yang diletakan di permukaan bumi dapat mendeteksi percepatan 1g (ukuran gravitasi bumi) pada titik vertikalnya, untuk percepatan yang dikarenakan oleh pergerakan horizontal maka *accelerometer* akan mengukur percepatannya secara langsung ketika bergerak secara horizontal.

2.5 Kontrol PID

Kontrol PID *(Proportional Integral Derivative)* merupakan kontroler untuk menentukan kepresisian suatu sistem instrumentasi dengan karakteristik adanya

umpan balik / *feed back* pada sistem tersebut. Komponen PID terdiri dari 3 jenis, yaitu *Proportional, Integral,* dan *Derivative*. Ketiganya dapat dipakai bersamaan maupun sendiri-sendiri, tergantung dari respon yang kita inginkan terhadap suatu *plant*. Berikut adalah gambar dari diagram blok PID ditunjukkan pada gambar 2.9



Gambar 2.9 Diagram Blok Sistem Kontrol PID

(Sumber : http://id.wikipedia.org/wiki/PID)

Adapun persamaan control PID adalah :

$y(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$ .....(2.15)

Persamaan Pengontrol PID diatas dapat juga dituliskan sebagai berikut :

$y(t) = K_P \left(e(t) + \frac{1}{Ti} \int_0^t e(t)\,dt + T_d \frac{de(t)}{dt} \right)$ .....(2.16)

dengan :

$K_i = K_p \times \frac{1}{Ti}$ dan $K_d = K_P \times T_d$ .....(2.17)

Keterangan :

y(t) = *output* dari pengontrol PID

Kp = konstanta Proporsional

Ki = konstanta *Integral*

Kd= konstanta Detivatif

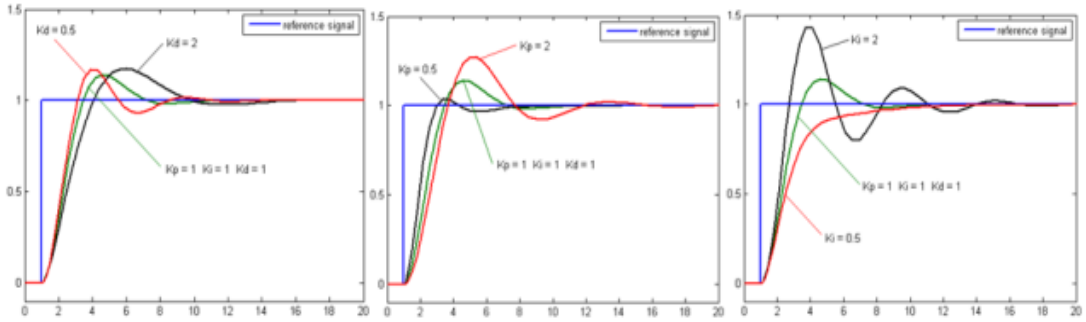
Ti= Waktu Integral

Td= Waktu derivatif

e(t) = error (selisih antara set point dengan level aktual)

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

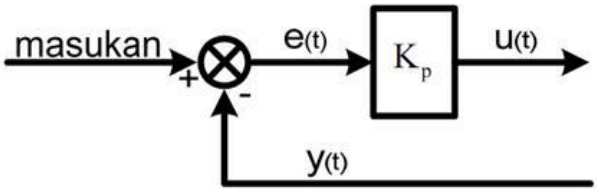
Adapun beberapa grafik dapat menunjukkan bagaimana respon dari sitem terhadap perubahan Kp, Ki dan Kd ditunjukkan pada gambar 2.10



Gambar 2.10 Respon PID

(Sumber: http://elib.unikom.ac.id/download.php?id=92558)

2.5.1 Kontrol *Proportional*



Gambar 2.11 Diagram Blok Kp

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

Kp berlaku sebagai G(s) = *gain* (penguat) saja tanpa memberikan efek dinamik kepada kinerja kontroler. Jika,

$$G(s) = kp \quad ....(2.18)$$

dengan k adalah konstanta. Jika,

$$u(t) = G(s) * e \text{ maka } u(t) = Kp * e(t) \quad ....(2.19)$$

dengan Kp adalah Konstanta *Proportional*.

Penggunaan kontrol P memiliki berbagai keterbatasan karena sifat kontrol yang tidak dinamik. Walaupun demikian dalam aplikasi-aplikasi dasar yang sederhana kontrol P cukup mampu untuk memperbaiki respon transien khususnya *rise time* dan *settling time*. *Output proportional* adalah hasil pekalian antara konstata proposional dengan nilai *error* nya. Perubahan yang terjadi pada sinyal *input* akan menyebabkan sistem secara langsung mengubah *output* sebesar konstata pengalinya.

Ciri-ciri pengontrol *proportional*:

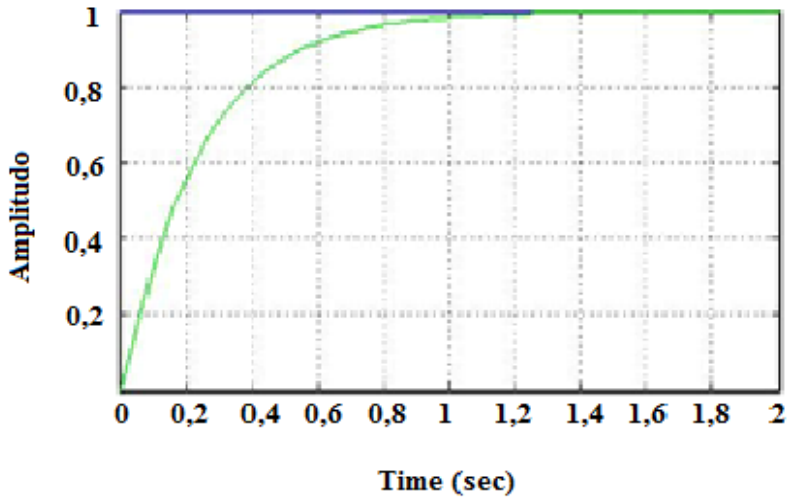
1. Jika nilai Kp kecil, pengontrol proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat (menambah *rise time*).
2. Jika nilai Kp dinaikkan, respon/tanggapan sistem akan semakin cepat mencapai keadaan mantapnya (mengurangi *rise time*).
3. Namun jika nilai Kp diperbesar sehingga mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil atau respon sistem akan berosilasi.
4. Nilai Kp dapat diset sedemikian sehingga mengurangi steady state *error*, tetapi tidak menghilangkannya.

Pengaruh kontrol *Proportional* pada sistem:

1. Menambah atau mengurangi kestabilan.
2. Dapat memperbaiki respon transien khususnya : *rise time, settling time*

3. Mengurangi (bukan menghilangkan) *Error steady state*. Untuk menghilangkan Ess, dibutuhkan Kp besar, yang akan membuat sistem lebih tidak stabil
4. Kontroler *Proportional* memberi pengaruh langsung (sebanding) pada *error*.Semakin besar *error*, semakin besar sinyal kendali yang dihasilkan kontroler.

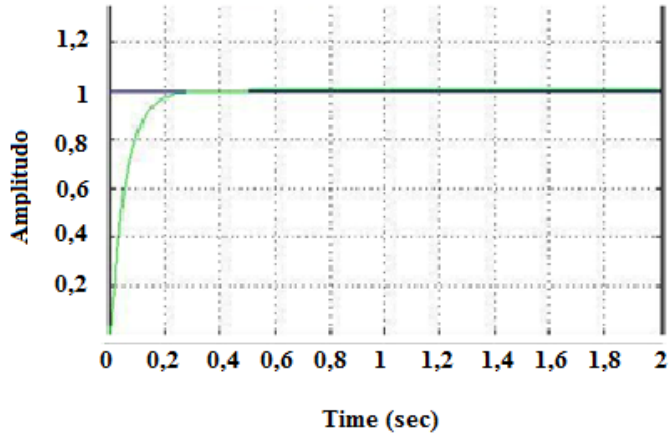
Jika nilai Kp kecil, kontroller proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat. Berikut adalah grafik jika Nilai Kp kecil ditunjukkan pada gambar 2.12



Gambar 2.12 Nilai Kp kecil

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

Jika nilai Kp besar, respon sistem menunjukan semakin cepat mencapai keadaan yang stabil, tetapi juga memungkinkan motor berputar diatas *set point*. Berikut adalah grafik jika Nilai Kp besar ditunjukkan pada gambar 2.13

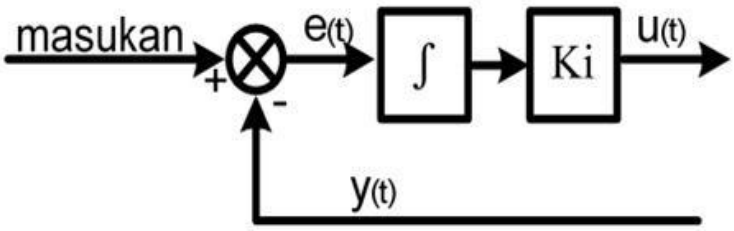


Gambar 2.13 Nilai Kp besar

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

2.5.2 Kontrol *Integral*

Kontrol *proporsional* tidak akan mampu menjamin *output* dari sistem akan menuju ke keadaan yang diinginkan kalau sebuah *plant* tidak memiliki unsur *integrator*. Pada control *Integral*, respon kepada sistem akan meningkat secara kontinu terus-menerus kecuali nilai *error* yang diIntegralkan dengan batasan atas t dan batasan bawah 0 (nol). Berikut adalah diagram blok Kontrol Integral ditunjukkan pada gambar 2.14



Gambar 2.14 Diagram Blok Ki

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/20 11-2-00660-%20SK%20Bab%202.pdf)

Hubungan antara output control integral u(t) engan sinyal error e(t) terlihat pada persamaan (2.6)

$u(t) = K_i \int_0^t e(t)\,dt$ .....(2.20)

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.

Keluaran pengontrol ini merupakan hasil penjumlahan yang terus menerus dari perubahan masukannya. Jika sinyal kesalahan tidak mengalami perubahan, maka keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan. Sinyal keluaran pengontrol *Integral* merupakan luas bidang yang dibentuk oleh kurva kesalahan / *error*.

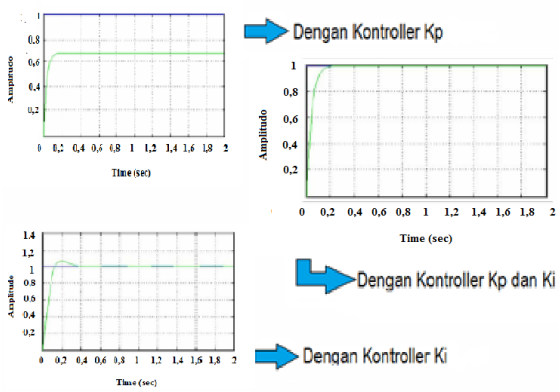
Ciri-ciri pengontrol *Integral* :

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

Pengaruh control *Integral* pada sistem :

1. Menghilangkan *Error* Steady State
2. Respon lebih lambat (dibandingkan dengan P)
3. Dapat Menambah Ketidakstabilan (karena menambah orde pada sistem)

Perubahan sinyal kontrol sebanding dengan perubahan *error*. Semakin besar *error*, semakin cepat sinyal kontrol bertambah/berubah. Berikut adalah grafik yang menunjukkan penggunaan Kp dan Ki yang sesuai ditunjukkan pada gambar 2.15



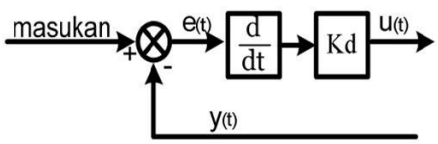
Gambar 2.15 Penggunaan Kp dan Ki

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

2.5.3 Kontrol *Derivative*

Keluaran pengontrol diferensial memiliki sifat seperti halnya suatu operasi *Derivative*. Perubahan yang mendadak pada masukan pengontrol akan mengakibatkan perubahan yang sangat besar dan cepat. Ketika masukannya tidak mengalami perubahan, keluaran pengontrol 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 magnitudenya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan factor konstanta Kd.

Berikut adalah diagram blok konrol *derivative* ditunjukkan pada gambar 2.16



Gambar 2.16 Diagram Blok Kd

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

Sinyal kontrol u yang dihasilkan oleh kontrol D dapat dinyatakan sebagai

$$u(t) = K_d \frac{de(t)}{dt} \quad \text{.....(2.21)}$$

Dari persamaan di atas, nampak bahwa sifat dari kontrol D ini dalam konteks "kecepatan" atau *rate* dari *error*. Dengan sifat ini dia dapat digunakan untuk memperbaiki respon transien dengan memprediksi *error* yang akan terjadi. Kontrol *Derivative* hanya berubah saat ada perubahan *error* sehingga saat *error* statis kontrol ini tidak akan bereaksi, hal ini pula yang menyebabkan kontroler *Derivative* tidak dapat dipakai sendiri.

Ciri-ciri pengontrol *Derivative* :

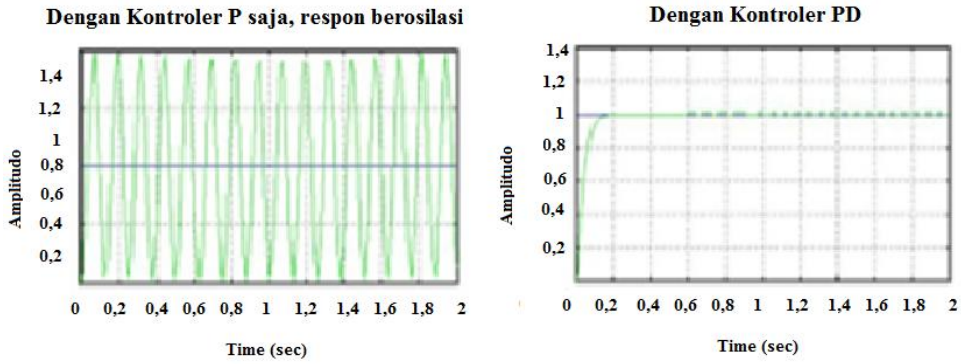
1. Pengontrol tidak dapat menghasilkan keluaran jika tidak ada perubahan pada masukannya (berupa perubahan sinyal kesalahan)
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengontrol tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
3. Pengontrol diferensial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

Pengaruh control *Derivative* pada sistem :

1. Memberikan efek redaman pada sistem yang berosilasi sehingga bisa memperbesar pemberian nilai Kp

2. Memperbaiki respon transien, karena memberikan aksi saat ada perubahan *error*
3. D hanya berubah saat ada perubahan *error*, sehingga saat ada *error* statis D tidak beraksi. Sehingga D tidak boleh digunakan sendiri.

Besarnya sinyal kontrol sebanding dengan perubahan *error* (e)Semakin cepat *error* berubah, semakin besar aksi kontrol yang ditimbulkan. Berikut adalah grafik penggunaan kontrol *proportional* dan kontrol *derivative* ditunjukkan pada gambar 2.17



Gambar 2.17 Penggunaan Kp dan Kd

(Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf)

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

Sumber:http://thesis.binus.ac.id/Asli/Bab2/2011-2-00660-%20SK%20Bab%202.pdf

2.5.4 Realisasi Kontrol PID Pada Bahasa Pemrograman

Kontrol PID umumnya diimplementasikan dengan menggunakan rangkaian elektronika analog. Bahkan banyak diantaranya direalisasikan dalam komponen mekanis dan pneumatis murni. Seiring dengan berkembangnya dunia digital (terutama mikroprosessor dan mikrokontroler) maka algoritma kontrol PID dapat direalisasikan kedalam bentuk persamaan PID digital. Yang jika diimplementasikan

hanya berupa sebuah program saja yang ditanamkan kedalam *embedded system* (mikroprosessor atau mikrokontroler).

Pada persamaan (2.15) merupakan PID bentuk *independent* dan persamaan (2.16) merupakan PID bentuk *dependent*. Istilah tersebut mengacu kepada ketergantungan setiap suku persamaan terhadap nilai Kp. Untuk persamaan (2.15), jika kita melakukan perubahan nilai pada konstanta proporsional (Kp) maka tidak akan mempengaruhi konstanta parameter lainnya. Sedangkan untuk persamaan (2.16), dengan merubah nilai Kp maka akan merubah nilai dari parameter-parameter lainnya. Jika ingin menggunakan persamaan *dependent*, maka hanya memasukan nilai dari K_p x $\frac{1}{Ti}$ dan $K_d = K_P$ x T_d

Pada persamaan (2.19), persamaan (2.20) dan persamaan (2.21) diatas merupakan persamaan dalam kawasan waktu *continuous* (analog). Sedangkan agar persamaan persamaan tersebut dapat direalisasikan dalam bentuk pemrograman, maka persamaan dalam kawasan waktu *continuous* tersebut harus didiskretisasi terlebih dahulu (kawasan digital). Berikut adalah diskretisasi menggunakan metode *numeric rectangular* mundur (*backward rectangular*).

Kontrol Proporsional

Jika dari persamaan (2.19) didiskretisasi maka akan menjadi:

$u(k) = Kp * e(t)$ .....(2.22)

Kontrol Integral

Jika dari persamaan (2.20) didiskretisasi maka akan menjadi:

$$u(k) = K_i \sum_{i=0}^{k} e(i)\, T_c$$

$$u(k) = K_i T_c \sum_{i=0}^{k} e(i) = K_i T_c [e(0) + e(1) + \cdots + e(k-1) + e(k)]$$

$$u(k) = K_i T_c [e(k-1) + e(k)] \quad \text{.....(2.23)}$$

Tc = waktu sapling atau waktu cuplik (sampling time)

"Integral (∫) adalah suatu operator matematis dalam kawasan kontinyu jika didiskretisasi maka akan menjadi sigma (∑), yang merupakan operator matematis dalam kawasan diskret. Dimana fungsi dari operator sigma adalah menjumlahkan nilai ke i sampai dengan nilai ke k. Berdasarkan perhitungan diatas variabel error (e) yang di integralkan sehingga dalam kawasan diskret menjadi e(0)+e(1)+…+e(k-1)+e(k), atau dengan kata lain **error yang sebelumnya dijumlahkan dengan error-error yang sebelumnya hingga error yang sekarang**."

Kontrol Derivatif

Jika dari persamaan (2.21) didiskretisai dengan menggunakan cara yang sama seperti kontrol integral maka akan menjadi:

$$u(k) = K_d \frac{e(k) - e(k-1)}{T_c} \quad \text{.....(2.24)}$$

Tc= waktu sampling atau waktu cuplik (Sampling time)

"Derivatif (de/dt) adalah suatu operator matematis dalam kawasan kontinyu jika didiskretisasi maka akan menjadi limit, yang merupakan operator matematis dalam kawasan diskret. Dimana fungsi dari operator limit adalah mengurangi nilai ke k dengan nilai ke k-1. Berdasarkan perhitungan diatas variabel error (e) yang di derivatifkan, atau dengan kata lain **error yang sekarang dikurangi error yang sebelumnya**."

Waktu sampling adalah lamanya waktu yang digunakan untuk mencuplik atau mensampling nilai dari sensor. Nilai dari sensor ini berguna untuk mendapatkan sinyal error (error(e)=set point-nilai sensor).

$$error = SP - PV$$

Dimana:

1. SP = Set point, secara simple maksudnya ialah suatu prameter nilai acuan atau nilai yang kita inginkan.

2. PV= Present Value, maksudnya ialah nilai bobot pembacaan sensor saat itu atau variabel terukur yang di umpan balikan oleh sensor (sinyal feedback dari sensor).
3. Error = nilai kesalahan, maksudnya ialah Deviasi atau simpangan antar variabel terukur atau bobot sensor (PV) dengan nilai acuan (SP)

Dari persamaan-persamaan (2.22), (2.23) dan (2.24) sudah berupa persamaan digital yang telah didiskretisasi, sehingga bisa langsung direalisasikan kedalam bahasa pemrograman.

2.6 Mikrokontroller ATmega32

Mikrokontroller Alf and Vegard's Risc processor (AVR) memiliki arsitektur 8 bit, dimana semua instruksi dikemas dalam kode 16-bit dan sebagian besar instruksi dieksekusi dalam 1 siklus clock atau dikenal dengan teknologi Reduced Instruction Set Computing (RISC). Secara umum, AVR dapat dikelompokan ke dalam 4 kelas, yaitu keluarga AT90Sxx, keluarga ATMega dan AT86RFxx. Pada dasarnya yang membedakan masing-masing adalah kapasitas memori, peripheral dan fungsinya (Heryanto, dkk, 2008:1).

ATMega32 adalah mikrokontroler CMOS 8 *bit* daya rendah berbasis arsitektur RISC. Instruksi dikerjakan pada satu siklus *clock*, ATMega32 mempunyai *throughput* mendekati 1 MIPS per MHz, hal ini membuat ATMega32 dapat bekerja dengan kecepatan tinggi walaupun dengan penggunaan daya rendah.

Mikrokontroler ATMega32 memiliki beberapa fitur atau spesifikasi yang menjadikannya sebuah solusi pengendali yang efektif untuk berbagai keperluan. Fitur-fitur tersebut antara lain:

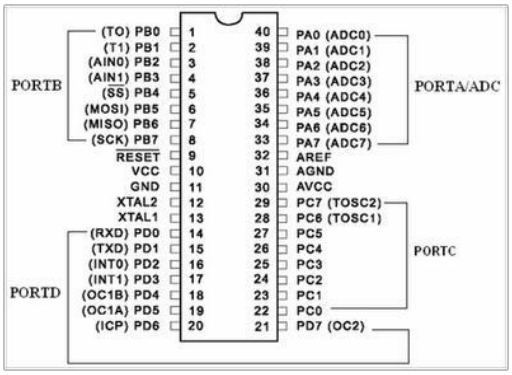
- Saluran I/O sebanyak 32 buah, yang terdiri atas *Port* A, B, C dan D
- ADC (*Analog to Digital Converter*)
- Memori *Flash* sebesar *8kb* dengan kemampuan *read while write*
- Unit Interupsi *Internal* dan *External*
- *Port* antarmuka SPI untuk men-*download* program ke *flas*
- EEP*ROM* sebesar 512 *byte* yang dapat diprogram saat operasi

- Antarmuka komparator *analog*
- *Port* USART untuk komunikasi serial.

2.6.1 Konfigurasi *Pin* ATMega32

Mikrokontroler AVR ATMega memiliki 40 *pin* dengan 32 *pin* diantaranya digunakan sebagai *port paralel*. Satu *port paralel* terdiri dari 8 *pin*, sehingga jumlah *port* pada mikrokontroler adalah 4 *port*, yaitu *port* A, *port* B, *port* C dan *port* D. Sebagai contoh adalah *port* A memiliki *pin* antara *port* A.0 sampai dengan *port* A.7, demikian selanjutnya untuk *port* B, *port* C, *port* D.

Diagram *pin* mikrokontroler dapat dilihat pada gambar 2.18



Gambar 2.18 Konfugirasi *pin* ATMega32

(Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam)

Berikut ini adalah tabel penjelasan mengenai *pin* yang terdapat pada mikrokontroler ATMega32:

Tabel 2.1 Tabel Penjelasan *pin* pada mikrokontroler ATMega32

Vcc	Tegangan suplai (*5 volt*)
GND	*Ground*
RESET	*Input reset level* rendah, pada *pin* ini selama lebih dari panjang pulsa *minimum* akan menghasilkan *reset* walaupun *clock* sedang berjalan. RST pada *pin* 9 merupakan *reset* dari AVR. Jika pada *pin* ini diberi masukan *low* selama minimal 2 *machine cycle* maka sistem akan di-*reset*

XTAL 1	*Input* penguat *osilator inverting* dan *Input* pada rangkaian operasi
XTAL 2	*Output* dari penguat *osilator inverting*
Avcc	*Pin* tegangan suplai untuk *port* A dan ADC. *Pin* ini harus dihubungkan ke Vcc walaupun ADC tidak digunakan, maka *pin* ini harus dihubungkan ke Vcc melalui *low pass filter*
Aref	*pin* referensi tegangan *analog* untuk ADC
AGND	*pin* untuk *analog ground*. Hubungkan kaki ini ke GND, kecuali jika *board* memiliki *analog ground* yang terpisah

Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam

Berikut ini adalah penjelasan dari *pin* mikrokontroler ATMega32 menurut *port*-nya masing-masing:

A. *Port* A

Pin 33 sampai dengan *pin* 40 merupakan *pin* dari *port* A. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* A dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* A (DDRA) harus di-*setting* terlebih dahulu sebelum *port* A digunakan. *Bit-bit* DDRA diisi 0 jika ingin memfungsikan *pin-pin port* A yang disesuaikan sebagai *Input*, atau diisi 1 jika sebagai *Output*. Selain itu, *pin-pin* pada *port* A juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel 2.2

Tabel 2.2 Tabel Penjelasan *pin* pada *port* A

Pin	Keterangan
PA.7	ADC7 (ADC *Input Channel* 7)
PA.6	ADC6 (ADC *Input Channel* 6)

PA.5	ADC7 (ADC *Input Channel* 5)
PA.5	ADC4 (ADC *Input Channel* 4)
PA.3	ADC3 (ADC *Input Channel* 3)
PA.2	ADC2 (ADC *Input Channel* 2)
PA.1	ADC1 (ADC *Input Channel* 1)
PA.0	ADC0 (ADC *Input Channel* 0)

Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam

B. *Port* B

Pin 1 sampai dengan *pin* 8 merupakan *pin* dari *port* B. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* B dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* B (DDRB) harus di-*setting* terlebih dahulu sebelum *port* B digunakan. *Bit-bit* DDRB diisi 0 jika ingin memfungsikan *pin-pin port* B yang disesuaikan sebagai *Input*, atau diisi 1 jika sebagai *Output*. Selain itu, *pin-pin port* B juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel 2.3

Tabel 2.3 Tabel Penjelasan *pin* pada *port* B

Pin	Keterangan
PB.7	SCK (SPI *Bus Serial Clock*)
PB.6	VISO (SPI *Bus Master Input/Slave Output*)
PB.5	VOSI (SPI *Bus Master Output/Slave Input*)
PB.4	SS (SPI *Slave Select Input*)
PB.3	AIN1 (*Analog Comparator Negative Input*)OCC (*Timer/Counter0 Output Compare Match Output*)
PB.2	AIN0 (*Analog Comparator Positive Input*)INT2 (*External Interrupt 2 Input*

PB.1	T1 (*Timer/Counter1 External Counter Input*)
PB.0	T0 (*Timer/Counter0 External Counter Input*)XCK (JSART *External Clock Input/Output*)

Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam

C. *Port* C

Pin 22 sampai dengan *pin* 29 merupakan *pin* dari *port* C. *Port* C sendiri merupakan *port Input* atau *Output*. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* C dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* C (DDRC) harus di-*setting* terlebih dahulu sebelum *port* C digunakan. *Bit-bit* DDRC diisi 0 jika ingin memfungsikan *pin-pin port* C yang disesuaikan sebagai *Input*, atau diisi 1 jika sebagai *Output*. Selain itu, *pin-pin port* C juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel 2.4.

Tabel 2.4 Tabel Penjelasan *pin* pada *port* C

Pin	Keterangan
PC.7	TOSC2 (*Timer Oscillator Pin* 2)
PC.6	TOSC1 (*Timer Oscillator Pin* 1)
PC.1	SDA (*Two-Wire Serial Bus Data Input/Output Line*)
PC.0	SCL (*Two-Wire Serial Bus Clock Line*)

Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam

D. *Port D*

Pin 14 sampai dengan *pin* 20 merupakan *pin* dari *port* D. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* D dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* D (DDRD) harus di-*setting* terlebih dahulu sebelum *port* D digunakan. *Bit-bit*

DDRD diisi 0 jika ingin memfungsikan *pin-pin port* D yang disesuaikan sebagai *Input*, atau diisi 1 jika sebagai *Output*. Selain itu, *pin-pin port* D juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel 2.5

Tabel 2.5 Tabel Penjelasan *pin* pada *port* D

Pin	Keterangan
PD.0	RDX (*UART Input line*)
PD.1	TDX (*UART Output line*)
PD.2	INT0 (*external interrupt 0 Input*)
PD.3	INT1 (*external interrupt 1 Input*)
PD.4	OC1B (*Timer/Counter1 Output compareB match Output*)
PD.5	OC1A (*Timer/Counter1 Output compareA match Output*)
PD.6	ICP (*Timer/Counter1 Input capture pin*)
PD.7	OC2 (*Timer/Counter2 Output compare match Output*)

Sumber: Hendrawan (2007). Sistem Mikrokontroler.Makalah Teknik Elektro Politeknik Batam

2.7 Motor DC

Motor DC (*Direct Current*) adalah peralatan elektromagnetik dasar yang berfungsi untuk mengubah tenaga listrik menjadi tenaga mekanik yang desain awalnya diperkenalkan oleh Michael faraday lebih dari seabad yang lalu (E. Pitowarno, 2006). Motor DC dikendalikan dengan menentukan arah dan kecepatan putarnya. Arah putaran motor DC adalah searah dengan arah putaran jarum jam (*Clock Wise/CW*) atau berlawanan arah dengan arah putaran jarum jam (*Counter Clock Wise/CCW*), yang bergantung dari hubungan kutub yang diberikan pada motor DC. Kecepatan putar motor DC diatur dengan besarnya arus yang diberikan. Berikut adalah gambar motor DC yang ditunjukkan pada gambar 2.19

Gambar 2.19 motor DC

(Sumber:http://library.binus.ac.id/eColls/eThesisdoc/Bab2/2012-1-00631-sk%202.pdf)

Motor DC merupakan jenis motor yang menggunakan tegangan searah sebagai sumber tenaganya. Dengan memberikan beda tegangan pada kedua terminal tersebut, motor akan berputar pada satu arah, dan bila polaritas dari tegangan tersebut dibalik maka arah putaran motor akan terbalik pula. Polaritas dari tegangan yang diberikan pada dua terminal menentukan arah putaran motor sedangkan besar dari beda tegangan pada kedua terminal menentukan kecepatan motor.

2.8 IC L298 (Motor *Driver*)

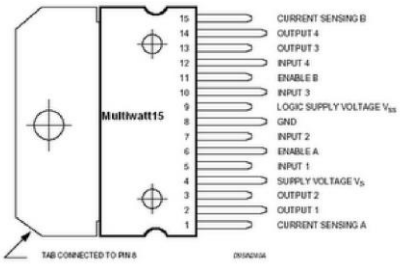
Motor *gear* DC tidak dapat dikendalikan langsung oleh mikrokontroler, karena kebutuhan arus yang besar sedangkan keluaran arus dari mikrokontroler sangat kecil. Oleh karena itu, dibutuhkan suatu *driver motor* sebagai alternatif yang dapat digunakan untuk menggerakkan motor DC.

IC L298 adalah IC yang dapat digunakan sebagai *driver motor* DC. L298 dapat mengontrol 2 buah motor DC. Tegangan yang dapat digunakan untuk mengendalikan robot bisa mencapai tegangan 46 VDC dan arus 2 A untuk setiap kanalnya. Berikut ini bentuk IC L298 yang digunakan sebagai *motor driver.* Berikut adalah gambar IC L298 yang ditunjukkan pada gambar 2.20

Gambar 2.20 *IC driver motor L298*

(Sumber: http://elib.unikom.ac.id/download.php?id=92558)

Pengaturan kecepatan kedua motor dilakukan dengan cara pengontrolan lama pulsa aktif/*Pulse width modulation* (PWM) yang dikirimkan ke rangkaian *driver motor* oleh pengendali (mikrokontroler). Berikut adalah penjelasan pin-pin pada IC L298 yang ditunjukkan pada gamba 2.21

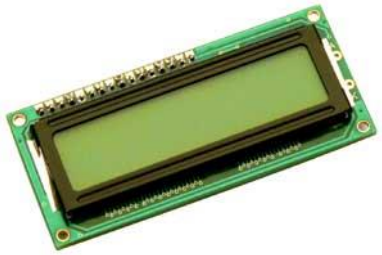


Gambar 2.21 Konfigurasi *Pin* IC L298

(Sumber:http://repository.usu.ac.id/bitstream/123456789/42918/4/Chapter%20II.pdf)

2.9 *Liquid Crystal Display* (LCD)

Liquid Crystal Display (LCD) berfungsi sebagai penampil data baik dalam bentuk karakter, huruf, angka ataupun grafik. Material LCD adalah lapisan dari campuran organik antara lapisan kaca bening dengan elektroda transparan indium oksida dalam bentuk tampilan *seven-segment* dan lapisan elektroda pada kaca belakang. Ketika elektroda diaktifkan dengan medan listrik (tegangan), molekul organik yang panjang dan silindris menyesuaikan diri dengan elektroda dari segmen. Lapisan *sandwich* memiliki *polarizer* cahaya *vertikal* depan dan *polarizer* cahaya *horizontal* belakang yang diikuti dengan lapisan reflektor. Cahaya yang dipantulkan tidak dapat melewati molekul-molekul yang telah menyesuaikan diri dan *segment* yang diaktifkan terlihat menjadi gelap dan membentuk karakter data yang ingin ditampilkan. Berikut adalah gambar LCD 16x2 yang ditunjukkan pada gambar 2.22



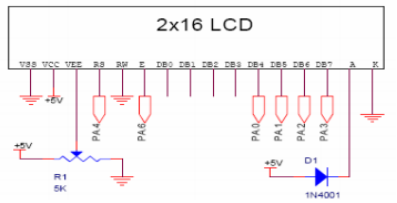
Gambar 2.22 Foto LCD (*Liquid Crystal Dislay*)

(Sumber:http://repository.usu.ac.id/bitstream/123456789/42918/4/Chapter%20II.pdf)

Pin, kaki atau jalur *input* dan kontrol dalam suatu LCD diantaranya adalah :

- Pin data adalah jalur untuk memberikan data karakter yang ingin ditampilkan menggunakan LCD dapat dihubungkan dengan bus data dari rangkaian lain seperti mikrokontroler dengan lebar data 8 bit.

- Pin RS (*Register Select*) berfungsi sebagai indikator atau yang menentukan jenis data yang masuk, apakah data atau perintah. Logika low menunjukan yang masuk adalah perintah, sedangkan logika high menunjukan data.
- Pin R/W (*Read Write*) berfungsi sebagai instruksi pada modul jika low tulis data, sedangkan high baca data. Pin E (*Enable*) digunakan untuk memegang data baik masuk atau keluar. Pin VLCD berfungsi mengatur kecerahan tampilan (kontras) dimana pin ini dihubungkan dengan trimpot 5 Kohm, jika tidak digunakan dihubungkan ke ground, sedangkan tegangan catu daya ke LCD sebesar 5 Volt. Berikut ini adalah skematik LCD 16x2 yang ditunjukkan pada gambar 2.23



Gambar 2.23 Skematik Rangkaian LCD 16x2

(Sumber : Elektronika-dasar.web.id "LCD 16x2", 2012)

Spesifikasi pada LCD 16x2 adalah sebagai berikut :

- Terdiri dari 16 kolom dan 2 baris
- Mempunyai 192 karakter yang tersimpan
- Tegangan kerja 5V
- Memiliki ukuran yang praktis

Sumber:http://repository.usu.ac.id/bitstream/123456789/42918/4/Chapter%20II.pdf

2.10 Catu Daya

Catu daya memegang peranan yang sangat penting dalam hal perancangan sebuah robot. Tanpa bagian ini robot tidak akan berfungsi. Begitu juga bila pemilihan catu daya tidak tepat, maka robot tidak akan bekerja dengan baik.

Penentuan sistem catu daya yang akan digunakan ditentukan oleh banyak faktor, diantaranya:

- Tegangan

Setiap aktuator tidak memiliki tegangan yang sama. Hal ini akan berpengaruh terhadap disain catu daya. Tegangan tertinggi dari salah satu aktuator akan menentukan nilai tegangan catu daya.

- Arus

Arus memiliki satuan Ah (*Ampere-hour*).Semakin besar Ah, semakin lama daya tahan baterai bila digunakan pada beban yang sama.

- Teknologi Baterai

Baterai isi ulang ada yang dapat diisi hanya apabila benar-benar kosong, dan ada pula yang dapat diisi ulang kapan saja tanpa harus menunggu baterai tersebut benar-benar kosong.

Banyak jenis baterai yang bisa dipakai untuk mensuplai tegangan untuk robot antara lain (*Alkaline, Fuel Cell, Lead Acid, Lithium, NiCad, NiMH,* baterai *Lithium Polymer* (LIPO)). Namun catu daya yang digunakan dalam robot ini menggunakan baterai *Lithium Polymer* (LIPO). Pada baterai jenis NiCad atau NiMH tiap sel memiliki 1,2 volt sedangkan pada baterai Li-Po memiliki *rating* 3,7 volt per sel. Keuntungannya adalah tegangan baterai yang tinggi dapat dicapai dengan menggunakan jumlah sel yang lebih sedikit.

Pada setiap paket baterai Li-Po selain tegangan ada label yang disimbolkan dengan “S”. Disini “S” berarti sel(*cell*) yang dimiliki sebuah paket baterai (*battery pack*). Sementara bilangan yang berada didepan simbol menandakan jumlah sel dan biasanya berkisar antar 2-6S (meskipun kadang ada yang mencapai 10S). Berikut adalah beberapa contoh notasi baterai Li-Po.

- 3.7 volt *battery* = 1 cell x 3.7 volts
- 7.4 volt *battery* = 2 cells x 3.7 volts (2S)
- 11.1 volt *battery* = 3 cells x 3.7 volts (3S)
- 14.8 volt *battery* = 4 cells x 3.7 volts (4S)
- 18.5 volt *battery* = 5 cells x 3.7 volts (5S)
- 22.2 volt *battery* = 6 cells x 3.7 volts (6S)

Ada tiga kelebihan utama yang ditawarkan oleh baterai berjenis Li-Po dibandingkan baterai jenis lain seperti NiCad atau NiMH yaitu :

- Baterai Li-Po memiliki bobot yang ringan dan tersedia dalam berbagai macam bentuk dan ukuran.
- Baterai Li-Po memiliki kapasitas penyimpanan energi listrik yang besar.
- Baterai Li-Po memiliki tingkat *discharge rate* energi yang tinggi, dimana hal ini sangat berguna sekali dalam bidang RC selain keuntungan lain yang dimilikinya.

Baterai jenis ini juga memiliki beberapa kelemahan yaitu:

- Harga baterai Li-Po masih tergolong mahal jika dibandingkan dengan baterai jenis NiCad dan NiMH.
- Performa yang tinggi dari baterai Li-Po harus dibayar dengan umur yang lebih pendek. Usia baterai Li-Po sekitar 300-400 kali siklus pengisian ulang. Sesuai dengan perlakuan yang diberikan pada beterai.
- Baterai Li-Po menggunakan bahan elektrolit yang mudah terbakar.
- Baterai Li-Po membutuhkan penanganan khusus agar dapat bertahan lama. *Charging, Discharging*, maupuan penyimpanan dapat mempengaruhi usia dari baterai jenis ini.

Gambar 2.24 Baterai Li-Po

(Sumber: http://repository.maranatha.edu/3861/4/0522013_Chapter2.pdf)

2.11 Basic Compiler AVR

BASCOM-AVR adalah program basic compiler berbasis windows untuk mikrokontroler keluarga AVR merupakan pemrograman dengan bahasa tingkat tinggi ” BASIC ” yang dikembangkan dan dikeluarkan oleh MCS elektronika sehingga dapat dengan mudah dimengerti atau diterjemahkan.

Dalam program BASCOM-AVR terdapat beberapa kemudahan, untuk membuat program software ATMEGA 32, seperti program simulasi yang sangat berguna untuk melihat, simulasi hasil program yang telah kita buat, sebelum program tersebut kita download ke IC atau ke mikrokontroler. Berikut ini adalah tampilan Bascom AVR versi 1.11.9.8 yang ditunjukkan pada gambar 2.25

Gambar 2.25 Logo Bascom-AVR

(Sumber http://www.robotics-university.com/2013/04/cara-menggunakan- bascom-avr.html)

Ketika program BASCOM-AVR dijalankan dengan mengklik icon BASCOM-AVR, maka jendela berikut akan tampil seperti pada gambar 2.26

Gambar 2.26 Tampilan Jendela Program BASCOM-AVR

(Sumber http://www.robotics-university.com/2013/04/cara-menggunakan- bascom-avr.html)

BASCOM-AVR menyediakan pilihan yang dapat mensimulasikan program. Program simulasi ini bertujuan untuk menguji suatu aplikasi yang dibuat dengan pergerakan LED yang ada pada layar simulasi dan dapat juga langsung dilihat pada LCD, jika kita membuat aplikasi yang berhubungan dengan LCD.

2.11.1 Kontruksi bahasa BASIC pada BASCOM-AVR

Setiap bahasa pemprograman mempunyai standar penulisan program. Konstruksi dari program bahasa BASIC harus mengikuti aturan sebagai berikut:

```
$regfile = "header"
'inisialisasi
'deklarasi variabel
'deklarasi konstanta
Do
'pernyataan-pernyataan
Loop end
```

2.11.2 Pengarah preprosesor

$regfile = "m32def.dat" merupakan pengarah pengarah preprosesor bahasa BASIC yang memerintahkan untuk meyisipkan file lain, dalam hal ini adalah *file* m32def.dat yang berisi deklarasi register dari mikrokonroller ATmega 32, pengarah preprosesor lainnya yang sering digunakan ialah sebagai berikut:

```
$crystal = 12000000  'menggunakan crystal clock 12 MHz
$baud = 9600         'komunikasi serial dengan baudrate 9600
$eeprom              'menggunakan fasilitas eeprom
```

2.11.3 Tipe Data

Tipe data merupakan bagian program yang paling penting karena sangat berpengaruh pada program. Pemilihan tipe data yang tepat maka operasi data menjadi lebih efisien dan efektif.

Tabel 2.6 Tipe Data pada BASCOM AVR

TIPE DATA	UKURAN (BYTE)	RENTANG
Bit	1/8	0 atau 1
Byte	1	0 sampai 255
Integer	2	-32768 sampai +32767
Word	2	0 sampai 65535

Long	4	-2147483648 sampai +2147483647
Single	4	
String	Maksimum 254 byte	

(Sumber : Byron (1991). *Pemrograman dengan Basic.*Penerbit Erlangga:Jakarta)

2.11.4 Konstanta dan Variabel

Konstanta adalah suatu nilai dengan tipe data tertentu yang tidak dapat diubah-ubah selama proses program berlangsung. Konstanta harus didefinisikan terlebih dahulu diawal program. Contoh : *Kp = 35, Ki=15, Kd=40*

Variabel adalah suatu pengenal (*identifier)* yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program yang dapat diubah-ubah sesuai dengan kebutuhan. Nama dari variable terserah sesuai dengan yang diinginkan namun hal yang terpenting adalah setiap variabel diharuskan :

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf, max 32 karakter.
2. Tidak boleh mengandung spasi atau symbol-simbol khusus seperti : *$, ?,%, #, !, &, *, (,), -, +, =* dan lain sebagainya kecuali *underscore*.
3. Deklarasi sangat diperlukan bila akan menggunakan pengenal (identifier) dalam suatu program.

2.11.5 Deklarasi Variabel, Konstanta, Fungsi dan Buatan

Bentuk umum pendeklarasian suatu variable adalah Dim *nama_variabel* AS *tipe_data.* Contoh : *Dim x* As *Integer* '*deklarasi x bertipe integer*. Sedangkan dalam konstanta di deklarasikan langsung. Contohnya : *S = "Hello world"* '*Assign string.*

Deklarasi fungsi adalah bagian yang terpisah dari program dan dapat dipanggil di manapun di dalam program. Fungsi dalam Bahasa Basic ada yang sudah disediakan sebagai fungsi pustaka seperti print, input data dan untuk menggunakannya tidak perlu dideklarasikan.

Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah Sub *Test* (byval *variabel* As *type*). Contohnya: Sub *Pwm (*byval *Kiri* As *Integer ,* Byval *Kanan As Integer)*

2.11.6 Operator

Operator digunakan untuk melakukan operasi terhadap bilangan. Pada Bascom operator dibedakan menjadi operator aritmetik, operator relasional, dan operator logika. Operator aritmatik adalah operator yang digunakan dalam kalkulasi, yaitu + (penjumlahan), - (pengurangan), * (perkalian, / (pembagian), \ (pembagian integer), MOD (modulo = sisa dari pembagian). Operator relasional digunakan untuk membandingkan dua nilai, yang memberikan hasil benar (1) atau salah (0) dan dapat digunakan untuk membuat keputusan.

Tabel 2.7 Operator-Operator Relasi Pada BASCOM AVR

Operator	**Relasi**	**Pernyataan**
=	Sama dengan	X = Y
< >	Tidak sama dengan	X < > Y
<	Lebih kecil	X < Y
>	Lebih besar	X > Y
<=	Lebih kecil atau sama dengan	X <= Y
>=	Lebih besar atau sama dengan	X >= Y

(Sumber : Byron (1991). *Pemrograman dengan Basic.*Penerbit Erlangga:Jakarta)

Operator logika digunakan untuk menguji suatu pola bit tertentu, manipulasi bit atau operator Boolean. Misal operator AND dapat digunakan untuk mengabaikan semua bit dalam suatu byte kecuali satu bit untuk memantau status bit tersebut.

Tabel 2.8 Operator-Operator Logika Pada Bascom AVR

Operator	**Makna**
NOT	Komplemen/Inverter
AND	Konjungsi (dan)
OR	Disjungsi (atau)
EXOR	Exclusive OR

(Sumber : Byron (1991). *Pemrograman dengan Basic.*Penerbit Erlangga:Jakarta)