DT-51 *Application Note*

AN116 – DC Motor Speed Control using PID

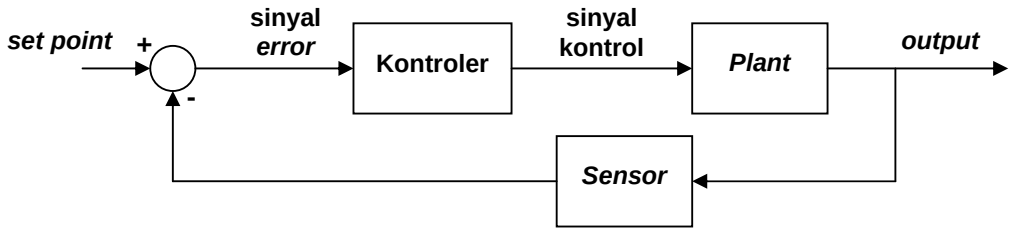
Oleh: Tim IE, Yosef S. Tobing, dan Welly Purnomo
(Institut Teknologi Sepuluh Nopember)

Sistem kontrol dengan metode PID (*Proportional Integral Derivative*) merupakan salah satu metode kontrol yang banyak digunakan dalam dunia industri. Dewasa ini banyak sistem kontrol yang menggunakan kontroler digital sebagai pengganti kontroler analog. Hal ini disebabkan kecepatan kontroler digital yang semakin tinggi dan kemudahan dalam perbaikan performansi sistem kontrol, yaitu dengan mengubah *software*, dibandingkan dengan mengubah *hardware*. Kali ini metode PID akan diimplementasikan secara digital dalam sistem pengaturan kecepatan motor DC dengan menggunakan modul DT-51™ Minimum System Ver 3.3 dan SPC DC Motor. Adapun perlengkapan yang dibutuhkan adalah sebagai berikut:

- DT-51™ Minimum System Ver 3.3,
- SPC DC Motor,
- Motor DC yang dilengkapi rangkaian *speed encoder,*
- DT-I/O 3x4 Keypad Module,
- LCD karakter 24x2 (dengan *driver* HD44780 / kompatibel).

PID

Diagram blok sebuah sistem kontrol dengan umpan balik secara umum dapat digambarkan seperti pada gambar di bawah ini.



Gambar 1
Diagram Blok Sistem Kontrol dengan Umpan Balik

Set point merupakan harga atau keadaan tertentu yang ingin dicapai, misalnya kecepatan motor DC sebesar 2000 rpm (*rotation per minute*). Sensor berfungsi mendeteksi keluaran *plant* dan mengkonversikannya menjadi besaran dengan satuan yang sama seperti satuan *set point*. Sebagai contoh, kecepatan motor DC dapat dideteksi dengan menggunakan rangkaian *encoder* atau *tachogenerator*. Keluaran dari *encoder* yang berupa pulsa atau dari *tachogenerator* yang berupa tegangan harus dikonversikan menjadi satuan yang sama seperti satuan *set point*, misalnya rpm atau rps (*rotation per second*). Keluaran sensor merupakan sinyal umpan balik (*feedback*) yang akan dikurangkan dengan *set point* menghasilkan sinyal *error*. Jika satuan dari keluaran sudah sama dengan *set point* maka blok sensor dapat dihilangkan sehingga sekarang sinyal *feedback* adalah keluaran *plant*. Secara umum sinyal *error* merupakan selisih antara *set point* dengan keluaran *plant*. Secara matematis dituliskan :

$$e(t) = r(t) - y(t) \quad \text{.....(a)}$$

dengan :

$e(t)$ = sinyal *error*
$r(t)$ = *reference* atau *set point*
$y(t)$ = keluaran *plant*

Sinyal *error* diproses oleh kontroler menghasilkan sinyal kontrol yang diumpankan ke *plant*, dengan tujuan akhir agar keluaran dari *plant* sama dengan *set point* yang berarti sinyal *error* bernilai (atau mendekati) nol.

Dalam metode kontrol PID, sinyal kontrol dihasilkan dengan cara memperkuat sinyal *error* (*proportional*), mengintegralkan sinyal *error* (*integral*), dan membuatnya sebanding dengan laju perubahan sinyal *error* itu sendiri (*derivative*). Kontroler yang melakukan mekanisme tersebut disebut dengan *PID controller.* Secara umum sinyal kontrol dituliskan :

$$u(t) = Kp\left(e(t) + Ki\int e(t)dt + Kd\,\frac{de(t)}{dt} \right) \quad \text{.....(b)}$$

dengan :

$u(t)$ = sinyal kontrol
Kp = konstanta proporsional
Ki = konstanta integral
Kd = konstanta diferensial

Dalam bentuk digital rumus di atas dituliskan :

$$u[n] = Kp\left(e[n] + Ki\sum e[n]T + Kd\,\frac{e[n]-e[n-1]}{T} \right) \quad \text{.....(c)}$$

dengan T adalah periode *sampling*.

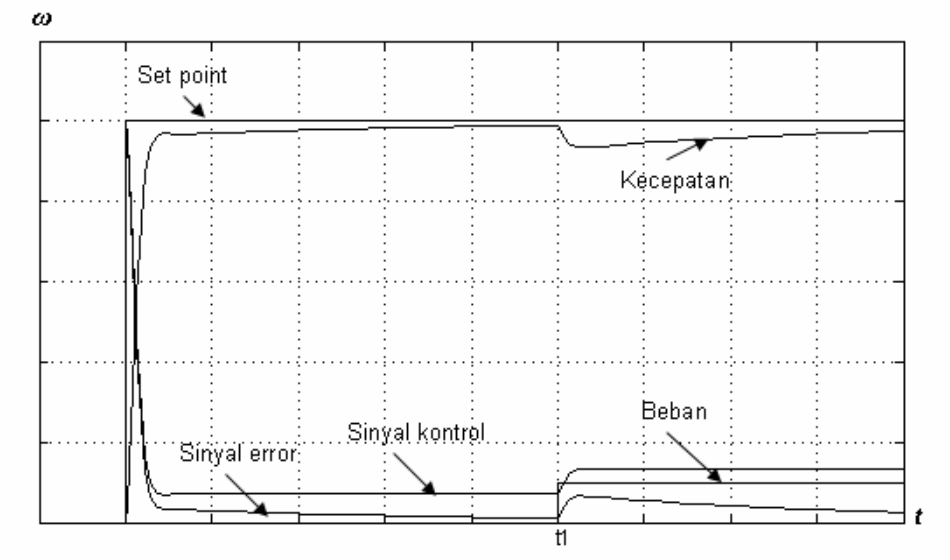
Pembahasan selanjutnya difokuskan pada sistem pengaturan kecepatan motor DC untuk memahami kinerja komponen proporsional, integral dan diferensial terhadap kecepatan motor. Motor DC pada umumnya diinginkan untuk berputar pada kecepatan yang konstan meskipun beban yang diberikan padanya berubah-ubah. Contoh kasus ini adalah pada ban berjalan (konveyor) dalam industri dimana beban yang diletakkan di atas ban berjalan tersebut berubah-ubah. Ketika beban yang diberikan pada motor bertambah, putaran motor cenderung melambat, sedangkan ketika beban yang diberikan pada motor berkurang, putaran motor cenderung bertambah cepat. Untuk itu perlu dibuat sebuah kontroler yang bertugas untuk menjaga kecepatan motor sehingga sesuai dengan kecepatan yang diinginkan.

Pengaruh komponen proporsional terhadap kecepatan motor dijelaskan sebagai berikut. *Error* positif, yang dihasilkan ketika kecepatan motor kurang dari *set point*, diperkuat oleh kontroler dengan nilai penguatan tertentu (umumnya dinotasikan Kp) untuk menghasilkan sinyal kontrol yang lebih besar, sehingga kecepatan motor bertambah. Ketika kecepatan motor bertambah maka sinyal error akan bertambah kecil yang berarti sinyal kontrol juga bertambah kecil. Pada akhirnya, kecepatan motor akan stabil pada kecepatan tertentu di bawah *set point* dimana sinyal kontrol seimbang dengan beban yang diberikan pada motor. Jika penguatan kontroler sangat tinggi maka kemungkinan terjadi osilasi yang disebabkan oleh koreksi berlebihan terhadap sinyal error secara terus menerus. Sinyal kontrol yang terlalu besar mengakibatkan motor berputar di atas *set point*, yang artinya dihasilkan sinyal *error* negatif. Sinyal *error* ini diperkuat oleh kontroler sehingga dihasilkan sinyal kontrol negatif yang berarti ada usaha untuk menghentikan putaran motor. Kecepatan motor akan berkurang drastis dan dihasilkan sinyal *error* positif yang memerlukan koreksi lebih lanjut. *Error* positif diperkuat oleh kontroler menghasilkan sinyal kontrol berlebihan yang membuat motor berputar pada kecepatan di atas *set point*. Demikian seterusnya sehingga terjadi osilasi dan dikatakan bahwa sistem tidak stabil. Komponen proporsional memiliki kegunaan terbatas sebab tidak dapat membuat motor untuk berputar tepat (mendekati) *set point*, namun mampu menghasilkan respon yang cepat terhadap sinyal *error*.

Pengembangan lebih lanjut dilakukan dengan menambahkan atau mengurangkan nilai tertentu pada sinyal kontrol sampai motor mencapai *set point*, dimana tidak terjadi perubahan lebih lanjut. Nilai ini dapat diperoleh dengan mengakumulasi sinyal *error*. Secara efektif nilai ini adalah integral dari sinyal *error*. Nilai ini diperkuat dengan nilai penguatan tertentu (umumnya dinotasikan Ki) membentuk suku integral, sebelum ditambahkan pada sinyal kontrol. Suku integral bekerja lebih lambat untuk mengkoreksi error, namun mampu menghilangkan *steady state error*.

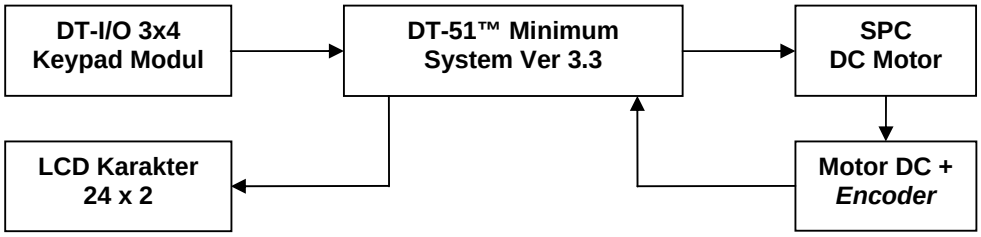
Perbaikan selanjutnya adalah dengan menggunakan laju perubahan sinyal *error* untuk ditambahkan pada sinyal kontrol. Ketika motor mengalami perlambatan yang sangat cepat, misalnya akibat penambahan beban secara tiba-tiba, sinyal *error* juga akan meningkat secara cepat. Laju perubahan (diferensial) sinyal *error* diperkuat dengan nilai penguatan tertentu (umumnya dinotasikan Kd) membentuk suku diferensial, sebelum ditambahkan pada sinyal kontrol. Suku diferensial bersifat antisipatif karena merespon terhadap laju perubahan sinyal *error*. Artinya semakin besar laju perubahan sinyal *error*, semakin besar pula suku diferensial. Akibatnya sinyal kontrol juga meningkat secara cepat akibat adanya suku diferensial ini. Suku diferensial juga dapat meredam osilasi yang disebabkan tingginya penguatan pada komponen proporsional. Namun demikian, banyak kontroler tidak menggunakan komponen diferensial, sebab dengan hanya menggunakan komponen proporsional dan integral, aksi kontrol yang dihasilkan sudah cukup bagus.

Set point, sinyal *error*, sinyal kontrol, dan pengaruh pemberian beban terhadap kecepatan motor DC diilustrasikan pada Gambar 2. Saat *t* = t1 motor diberi beban sehingga kecepatan motor berkurang. Terlihat bahwa kontroler menghasilkan sinyal kontrol yang lebih besar untuk mengembalikan kecepatan motor ke *set point*.



Gambar 2
Ilustrasi Pengaruh Pemberian Beban terhadap Kecepatan Motor

Implementasi sistem pengaturan kecepatan motor DC berbasis DT-51™ Minimum System Ver 3.3 tampak pada gambar di bawah.



Gambar 3
Diagram Blok Hubungan Antar Modul

Hubungan antara modul-modul tersebut adalah sebagai berikut:

DT-51™ Minimum System Ver 3.3 (Port LCD)	LCD Karakter 24 x 2
GND	GND (Pin 1)
VCC	VCC (Pin 2)
CON	Vo (Pin 3)
P10	RS (Pin 4)
P12	E (Pin 6)
P14	DB4 (Pin 11)
P15	DB5 (Pin 12)
P16	DB6 (Pin 13)
P17	DB7 (Pin 14)
VCC	A (Pin 15)
GND	K (Pin 16)

Tabel 1
Hubungan DT-51™ Minimum System Ver 3.3 dengan LCD Karakter 24 x 2

DT-51™ Minimum System Ver 3.3	SPC DC Motor
P11 (Port C & Port 1)	SCL (J2)
P13 (Port C & Port 1)	SDA (J2)
VCC (Control)	+5V (J1)
GND (Control)	GND (J1)

Tabel 2
Hubungan DT-51™ Minimum System Ver 3.3 dengan SPC DC Motor

Port LCD pada DT-51™ Minimum System Ver 3.3 dirancang agar dapat dihubungkan dengan LCD secara cepat (LCD dengan *driver* HD44870 / sejenis). P11 pada Port LCD seharusnya terhubung ke pin 5 LCD ($R/\overline{W}$). Perlu diingat bahwa P11 telah dihubungkan dengan SCL SPC DC Motor sehingga P11 tidak boleh digunakan lagi untuk keperluan pembacaan dari atau penulisan ke LCD. Karena itu hubungkan pin 5 LCD ($R/\overline{W}$) ke *ground* LCD (pin 1 atau pin 16) sehingga kini hanya dapat dilakukan proses penulisan ke LCD.

SPC DC Motor (J6)	Motor DC
M1 +	Kutub +
M1 -	Kutub -

Tabel 3
Hubungan SPC DC Motor dengan Motor DC

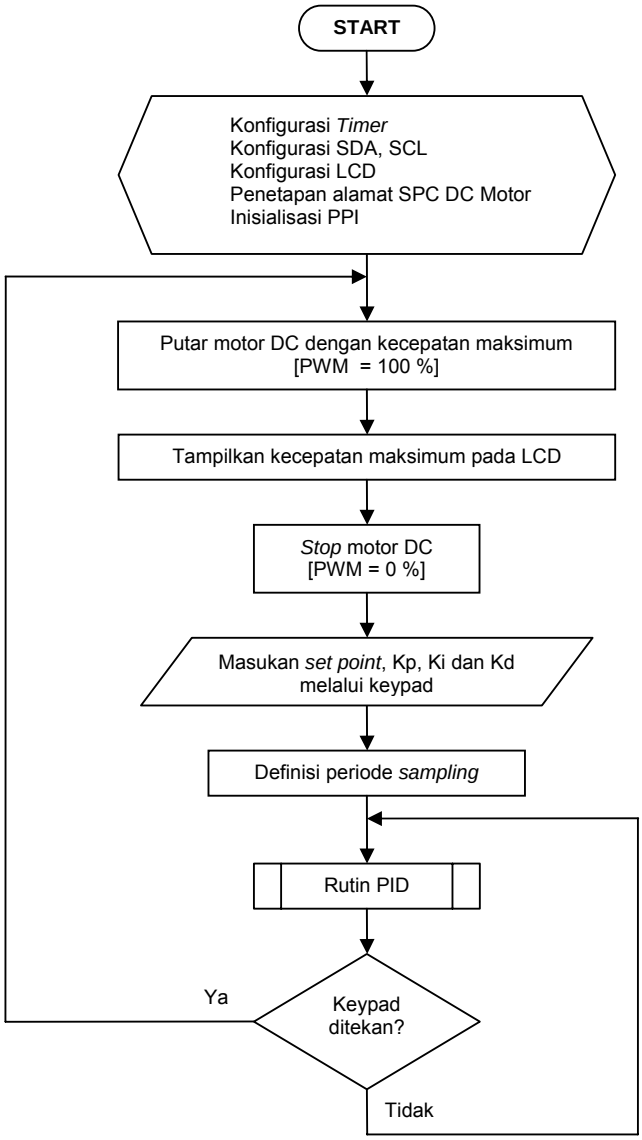
Pada modul SPC DC Motor, hubungkan VMotor (J6) dengan sumber tegangan yang sama dengan tegangan motor yang akan digunakan, misalnya motor DC yang digunakan 12 volt maka hubungkan VMotor (J6) dengan sumber tegangan 12 volt. Kemudian hubungkan kutub positif motor DC pada M1+ dan kutub negatif motor DC pada M1–. Hubungkan +5V (J1, SPC DC Motor) dengan tegangan rangkaian *speed encoder* serta hubungkan *output* rangkaian *speed encoder* ke pin T1 (Control, DT-51™ Minimum System Ver 3.3). Jika Anda menggunakan motor DC dengan *encoder* yang sudah *built-in*, Anda perlu memberi resistor *pull-up* pada *output* rangkaian *encoder*, gunakan nilai resistor 10K ohm. T1 terhubung ke pin T1 dari mikrokontroler AT89C51 yang difungsikan sebagai *counter*. *Counter* ini berfungsi mencacah pulsa yang dihasilkan rangkaian *speed encoder*. Anda dapat menghubungkannya dengan pin T0 dengan mengubah program. *Jumper* Address A0-A2 (J4) pada SPC DC Motor tidak dipasang sedangkan semua *jumper* pada SDA dan SCL (J3) dipasang. Sehingga alamat I^2C untuk modul SPC DC Motor adalah EEH.

DT-51™ Minimum System Ver 3.3	DT-I/O 3 x 4 Keypad Module
VCC (Control)	VCC (J1)
PC0 (Port C & Port1)	C1 (J3)
PC1 (Port C & Port1)	C2 (J3)
PC2 (Port C & Port1)	C3 (J3)
PC3 (Port C & Port1)	-
PC4 (Port C & Port1)	R1 (J4)
PC5 (Port C & Port1)	R2 (J4)
PC6 (Port C & Port1)	R3 (J4)
PC7 (Port C & Port1)	R4 (J4)

Tabel 4
Hubungan DT-51™ Minimum System Ver 3.3 dengan DT-I/O 3 x 4 Keypad Module

Hubungkan *serial port* DT-51™ Minimum System Ver 3.3 ke COM1 atau COM2 komputer menggunakan kabel serial DT-51™ Minimum System Ver 3.3. Setelah semua rangkaian dan sumber tegangan terhubung secara tepat, *download*-lah **PID.HEX** ke DT-51™ Minimum System Ver 3.3. Anda dapat menggunakan **DT51L.exe** (versi Ms-DOS®) atau **DT51Win.exe** (versi Windows®) untuk melakukan *download* program.

Algoritma program utama dapat dilihat pada *flowchart* berikut:



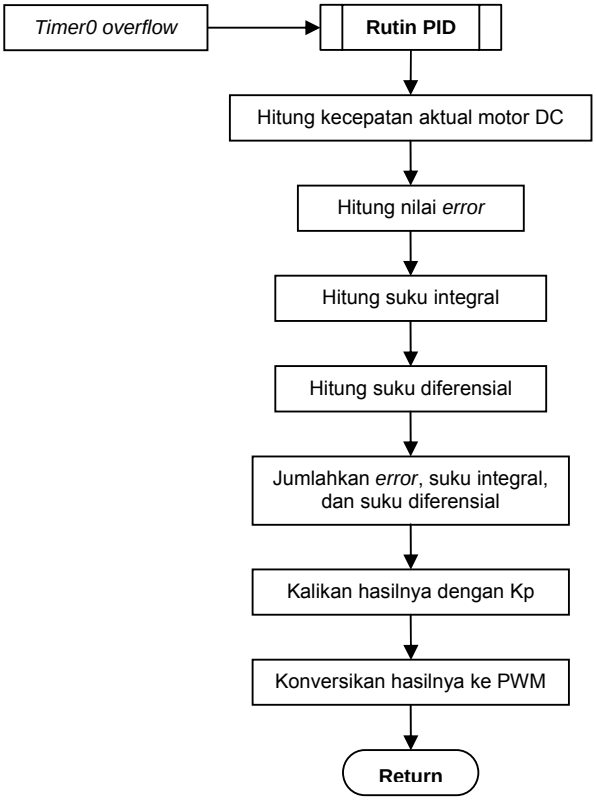
Gambar 4
***Flowchart* Program Utama**

Program utama bekerja sebagai berikut :

1. Program akan mengkonfigurasi *Timer0* sebagai *timer* dan *Timer1* sebagai *counter*. Baik *Timer0* maupun *Timer1* dioperasikan pada mode 1, yaitu *timer* atau *counter* 16 bit, serta interupsi *Timer0* diaktifkan. *Timer0* berfungsi sebagai pewaktu periode *sampling* T sehingga ketika *Timer0 overflow*, maka akan terjadi interupsi dan rutin perhitungan sinyal kontrol dengan metode PID akan dipanggil. Pada *listing* program rutin ini dinamakan rutin PID dan akan dijelaskan secara tersendiri. *Timer1* bertugas untuk mencacah pulsa keluaran rangkaian *speed encoder*. Program juga mengkonfigurasi pin SDA & SCL untuk komunikasi I^2C serta pin port LCD. Alamat SPC DC Motor juga ditetapkan, yaitu EEh. Port A PPI difungsikan sebagai *output*, Port B sebagai *output*, Port C *upper* (PC7-PC4) sebagai *output*, dan Port C *lower* (PC3-PC0) sebagai *input*. Port C inilah yang digunakan untuk keperluan pembacaan keypad.
2. Motor DC diputar pada kecepatan penuh (PWM 100 %) dengan mengirimkan data PWM = 0 ke SPC DC Motor melalui komunikasi I^2C.
3. Kecepatan maksimum motor DC ditampilkan pada LCD dalam satuan putaran per detik.
4. Setelah *user* menekan sembarang tombol pada keypad, maka motor DC dihentikan dengan mengirimkan data PWM = 255 (PWM = 0%) ke SPC DC Motor.

5. *User* diminta memasukkan nilai *Set point*, Kp, Ki, dan Kd. *User* hanya dapat mengisi bilangan bulat positif maksimal 3 digit untuk masing-masing variabel. Tombol bintang (*) dan pagar (#) berfungsi sebagai ENTER. *User* diharapkan tidak mengisi *Set point* lebih besar daripada kecepatan maksimum yang sebelumnya telah ditampilkan pada LCD. Jika *Set point* lebih besar daripada kecepatan maksimum, maka *error* tidak akan pernah bernilai nol.
6. Periode *sampling* T merupakan selang waktu dari diaktifkannya *Timer0* (start *Timer0*) sampai *Timer0 overflow*. Lihat kembali persamaan (c). Frekuensi *crystal oscillator* yang dipakai pada DT-51™ Minimum System adalah 11,0592 MHz. Frekuensi ini secara internal dibagi dengan 12 (1/12 x 11.0592 MHz = 0.9216 MHz), sehingga isi *Timer0* akan bertambah 1 setiap 1/0.9216 MHz = 1.08507 µs. Jika diinginkan periode *sampling* selama 50 ms, maka *Timer0* harus mencacah sebanyak 50 ms / 1.08507µs ≈ 46080 kali sebelum *overflow*. Ini berarti *Timer0* harus diisi 65536 – 46080 = 19456 sebagai nilai awal jika diinginkan periode *sampling* 50 ms. Definisi periode *sampling* diletakkan pada bagian ini semata-mata untuk kemudahan pemahaman *listing* program. Definisi periode *sampling* dapat diletakkan pada bagian awal program.
7. Setelah *Timer0* diisi dengan nilai awal, *Timer0* dan *Timer1* diaktifkan sehingga *Timer0* mulai menghitung dan *Timer1* mulai mencacah pulsa keluaran rangkaian *speed encoder*.
8. Ketika *Timer0 overflow*, maka rutin PID dipanggil dan motor akan mulai berputar di akhir rutin PID. Hal ini akan dijelaskan lebih lanjut pada rutin PID.
9. Jika sembarang tombol pada keypad ditekan maka program akan kembali ke langkah nomor 5.

Algoritma rutin PID dapat dilihat pada *flowchart* di bawah.



Gambar 5
***Flowchart* Rutin PID**

Program rutin PID bekerja sebagai berikut :

1. Rutin PID akan dikerjakan setiap kali *Timer0 overflow*. Laju *overflow Timer0* inilah yang merupakan periode *sampling* T sistem kontrol. Selama *Timer0* belum *overflow*, interupsi tidak akan terjadi dan rutin PID tidak akan dikerjakan. Lihat kembali penjelasan program utama pada langkah 6. Ketika terjadi interupsi, maka program akan mengerjakan hal berikut :
    - Isi *Timer1* yang difungsikan sebagai *counter* 16 bit disimpan dalam variabel *pulse* (pada *listing* program).

- *Timer0* diisi ulang dengan nilai awal.
- *Timer1* di-*reset* (isi dari *Timer1* dinolkan), sebab *Timer1* dipakai untuk mencacah pulsa keluaran rangkaian *speed encoder*.
- *Timer0* diaktifkan (start *Timer0*), periode *sampling* dimulai kembali.
- *Timer1* diaktifkan (start *Timer1*), penghitungan pulsa keluaran rangkaian *speed encoder* dimulai dari nol.

2. Kecepatan aktual motor DC diperoleh dari isi *Timer1* yang difungsikan sebagai *counter* 16 bit. Pulsa yang dicacah *Timer1* adalah pulsa yang dicacah selama 50 ms. Jika putaran motor DC diasumsikan konstan, berarti pulsa yang dicacah selama 1 s adalah 1 s / 50 ms = 20 kali isi register *Timer1*. Jadi isi register *Timer1* harus dikalikan 20 dan hasilnya harus dibagi dengan jumlah lubang yang terdapat pada piringan *speed encoder* untuk mendapatkan jumlah putaran motor DC setiap detik. Sebagai contoh, jika isi register *Timer1* = 175 ketika terjadi interupsi, dan piringan *encoder* memiliki 116 lubang, maka jumlah putaran motor DC setiap detik adalah (175 x 20) / 116 ≈ 30 putaran/detik. Jadi kecepatan aktual motor DC adalah 30 putaran/detik.
3. *Error* diperoleh dengan mengurangkan kecepatan aktual dari *set point*. Pada *listing* program nilai *error* disimpan dalam variabel error. Lihat kembali persamaan (a).
4. Yang dimaksud suku integral adalah suku $Ki\sum e[n]T$ pada persamaan (c). Periode *sampling* T dapat dikeluarkan dari notasi sigma sebab nilainya konstan, sehingga suku integral (pada *listing* program disimpan dalam variabel intgrl_term) menjadi :

   $$\text{intgrl\_term} = KiT\sum e[n]$$

   Pada program nilai Ki langsung dikalikan dengan periode *sampling* T segera setelah *user* memasukkan nilai Ki. Akumulasi *error* $\sum e[n]$ disimpan dalam variabel acc_error dan dihitung sebagai berikut :

   acc_error = acc_error + error

   Pada akhirnya suku integral menjadi :

   intgrl_term = Ki x acc_error

   dimana Ki = Ki x T. Alasan penggunaan variabel yang sama untuk nilai Ki yang baru adalah untuk menghemat pemakaian memori internal mikrokontroler AT89C51.
5. Yang dimaksud suku diferensial adalah suku $Kd\,\frac{e[n]-e[n-1]}{T}$ pada persamaan (c). Pada program nilai Kd langsung dibagi dengan periode *sampling* T segera setelah *user* memasukkan nilai Kd. $e[n]$ adalah nilai *error* saat ini dan $e[n\text{-}1]$ adalah nilai *error* sebelumnya. Pada program $e[n\text{-}1]$ disimpan dalam variabel prev_error sehingga suku diferensial (disimpan dalam variabel derv_term) menjadi :

   derv_term = Kd x (error – prev_error)

   dimana Kd = Kd / T.
6. Nilai *error*, suku integral dan suku diferensial dijumlahkan membentuk suku yang berada dalam tanda kurung pada persamaan (c) dan disimpan dalam variabel sum. Lihat kembali persamaan (c)

   sum = error + intgrl_term + derv_term
7. Variabel sum dikalikan dengan Kp membentuk sinyal kontrol.

   control = Kp x sum
8. Beberapa variabel di atas nilainya dibatasi oleh suatu angka maksimum (batas saturasi) agar hasil perhitungan tidak melebihi nilai maksimum yang dapat diakomodasi oleh suatu tipe data. Sebagai contoh variabel control dideklarasikan dengan tipe data *long*. Tipe data *long* dapat mengakomodasi nilai -2147483648 sampai dengan 2147483647. Nilai maksimum beberapa variabel dalam program dibatasi agar nilainya terletak diantara -10000 sampai 10000. Misalkan variabel sum bernilai 12000, maka nilainya akan dijadikan 10000. Jika nilai Kp = 999 (maksimal) maka variabel control bernilai 9990000. Nilai ini masih di bawah nilai maksimum tipe data *long* sehingga perhitungan dapat dilanjutkan dengan benar. Nilai maksimum ini dapat diubah asalkan *user* memperhatikan tipe-tipe data yang digunakan sehingga tidak terjadi kesalahan perhitungan. Nilai maksimum ini dipakai untuk menentukan nilai PWM yang harus dikirimkan ke SPC DC Motor. Perhitungan untuk PWM adalah sebagai berikut :

   control = (control / nilai maksimum) x 255

   PWM = 255 – control

   Nilai variabel control dikalibrasikan ke rentang nilai 0 – 255. Sekali lagi nilai baru variabel control disimpan dalam variabel yang sama dengan tujuan menghemat pemakaian memori internal mikrokontroler. Jika nilai control = nilai maksimum, maka PWM akan bernilai 0, yang berarti putar motor DC pada kecepatan penuh. Nilai maksimum variabel mempengaruhi performansi sistem kontrol dan dapat diubah sesuai keperluan. Anda dapat mencoba mengganti nilai maksimum ini dengan memperhatikan tipe-tipe data yang dipakai agar perhitungan tetap dapat dikerjakan dengan benar.
9. Nilai *error* dan kecepatan aktual ditampilkan pada LCD.

Catatan

Semua variabel pada *listing* program tidak ada yang menggunakan tipe data *single* (*floating* point 32 bit, sesuai standar IEEE) sehingga semua hasil perhitungan dibulatkan ke bawah. Tipe data *single* memungkinkan implementasi sistem kontrol dengan perhitungan yang lebih akurat dan respon yang lebih halus. Akan tetapi hasil kompilasi (file berekstensi HEX) perhitungan dengan tipe data *single* jauh lebih besar daripada hasil kompilasi perhitungan dengan tipe data yang lain. Memori internal yang dipakai juga lebih banyak. Tipe data *single* tidak dapat dikonversikan menjadi tipe data lain secara benar dengan tujuan menghemat pemakaian memori internal. Akibatnya semua variabel harus dideklarasikan dengan tipe *single* jika diinginkan perhitungan yang benar. BASCOM-8051© versi 2.0.12.0 (edisi demo) hanya memungkinkan hasil kompilasi sebesar 4K byte dan ini tidak cukup untuk mengkompilasi program PID jika semua variabel dideklarasikan dengan tipe *single*. Namun demikian, sistem pengaturan kecepatan motor DC berbasis DT-51™ Minimum System dalam aplikasi ini cukup representatif untuk melihat performansi metode kontrol PID dengan mengubah-ubah Kp, Ki dan Kd. Sebagai contoh, nilai Kp yang terlalu besar akan menyebabkan kecepatan motor tidak stabil seperti yang telah dijelaskan pada teori sebelumnya.

Listing program ditulis menggunakan BASCOM-8051© versi 2.0.12.0 (edisi demo) dan dapat dilihat pada **AN116.ZIP**.

Selamat berinovasi!