Pengembangan Kontroler PID Digital Menggunakan Matlab dan Proteus

Development of Digital PID Controller Using Matlab dan Proteus

Sidik Nurcahyo[1*], Sungkono[2], Yulianto[3], Hairus Shandy[4]
[1,2,3,4]Jurusan Teknik Elektro, Politeknik Negeri Malang
Jl. Soekarno Hatta No. 9, Kota Malang, Jawa timur, Indonesia
sidik.nurcahyo@polinema.ac.id[1*], sungkono@polinema.ac.id[2], yulianto@polinema.ac.id[3],
hairus@polinema.ac.id[4]

Abstrak – *Artikel ini membahas aspek teknis terkait desain dan implementasi kontroler PID digital pada mikrokontroler menggunakan Matlab dan Proteus. Matlab digunakan untuk mencari parameter PID digital yang tepat sesuai dengan beberapa kriteria perancangan, seperti gain margin, frekuensi crossover, dan amplitudo maksimal aktuator. Sedangkan Proteus digunakan untuk mengimplementasi dan menguji PID digital hasil rancangan. PID digital tersebut diimplementasikan pada mikrokontroler AVR 8-bit yang dilengkapi dengan program PID berupa kode C WinAVR. Fungsi alih plant yang akan dikontrol dianggap telah diketahui dan parameter PID ditala menggunakan tool Matlab bernama pidtune. Tahap implementasi dan pengujian dimulai dengan membuat diagram simulasi Proteus yang memuat plant, sensor, aktuator dan kontroler PID. Kode program PID pada mikrokontroler dibuat berdasarkan persamaan beda yang diturunkan dari rumus PID digital hasil diskritisasi Euler Backward. Untuk menjamin bahwa program PID dijalankan sekali setiap waktu sampling yang ditentukan maka kode program PID diletakkan pada vektor interupsi timer. Hasil pengujian dengan waktu sampling 10ms menunjukkan bahwa respon PID digital didalam Proteus sangat mendekati respon PID analog didalam Matlab, dimana selisih output pada waktu mantap hanya 0.0134 atau 0.089% dari setpoint. Hal ini membuktikan bahwa PID yang dirancang menggunakan Matlab telah berhasil diimplementasikan secara digital pada mikrokontroler AVR 8-bit.*

Kata Kunci*: pengembangan, PID digital, mikrokontroler, Matlab, Proteus.*

Abstract – *This article discusses technical aspects related to design and implementation of digital PID controller on microcontroller using Matlab and Proteus. Matlab is used to find the right digital PID parameters according to some design criterias, such as gain margin, crossover frequency, and maximum actuator amplitude. Meanwhile, Proteus is used to implement and test the designed digital PID. This controller is implemented on an 8-bit AVR microcontroller which is equipped with PID program in the form of WinAVR C code. The plant transfer function to be controlled is assumed to be known and the PID parameters are tuned using Matlab tool named pidtune. The implementation and testing begin by creating Proteus simulation diagram containing plant, sensor, actuator and PID controller. PID program code on the microcontroller is created based on difference equation derived from the digital PID formula result of Euler Backward Discreatization. To guarantee that this program is executed once every specified sampling time, this code is placed in timer interrupt vector. The test results with sampling time 10ms show that Digital PID response in Proteus response is very close to analog PID in Matlab response, where otput difference at setting time is only 0.0134 or 0.089% of setpoint. This proves that the digital PID designed with Matlab has been successfully implemented on 8-bit AVR microcontroller.*

Keywords*: development, digital PID, microcontroller, Matlab, Proteus.*

1. Pendahuluan

Kontroler PID termasuk kontroler yang paling banyak digunakan dalam aplikasi sistem kendali karena unjuk kerjanya cukup baik dan mudah untuk diterapkan [1],[2]. Kontroler PID dapat diwujudkan dalam dua bentuk, yaitu analog [3] dan digital. Kontroler PID analog

membutuhkan beberapa komponen analog untuk mewujudkan persamaan matematik PID, meliputi: resistor, kapasitor, transistor dan operational amplifier [4]. Komponen-komponen ini rentan terhadap derau (*noise*) dan unjuk kerjanya mudah terganggu akibat dari nilai komponen yang berubah karena suhu, kelembaban dan usia. Adapun kontroler PID digital yang berupa kode program di dalam mikrokontroler lebih tahan terhadap gangguan eksternal tersebut sehingga lebih terjamin unjuk kerjanya [5]. Kenyataan ini menjadikan kontroler PID digital lebih disukai dan lebih banyak digunakan dibandingkan dengan kontroler PID analog.

Terdapat banyak artikel yang telah membahas kontroler PID digital dan aplikasinya, sebagai contoh: kontrol suhu udara [6], kontrol level air [7], kontrol HVAC [8], kontrol kecepatan BLDC [9], kontrol kecepatan motor DC [10], kontrol pemanas air [11], kontrol manipulator robot [12], kontrol posisi angular [13] masih banyak lagi. Namun artikel-artikel ini belum membahas aspek-aspek teknis terkait implementasi pada mikrokontroler. Contoh aspek teknis yang dimaksud diantaranya ialah bagaimana cara menjamin supaya waktu *sampling* bernilai tetap atau tidak berubah-ubah karena perubahan aliran program didalam mikrokontroler dan bagaimana memastikan bahwa waktu *sampling* itu benar-benar akurat atau bernilai sama dengan waktu *sampling* saat perancangan. Selain itu, bagaimana cara mengakomodasi fungsi alih atau gain sensor dan aktuator di dalam program.

Artikel ini akan membahas aspek teknis perancangan PID digital tersebut menggunakan Matlab dan Proteus. Disini Matlab digunakan untuk mencari parameter PID digital, mencakup $K_p, K_i,$ dan K_d. Parameter-parameter PID ini dicari berdasarkan fungsi alih plant yang telah diketahui, waktu *sampling* yang telah ditetapkan, dan kriteria perancangan, seperti: frekuensi *cossover*, gain margin dan misi kontroler. Terdapat tiga misi kontroler yang dapat dipilih, diantaranya: *setpoint tracer*, *disturbance rejection* atau keseimbangan diantara keduanya (*balanced*). Selain itu, Matlab juga digunakan untuk mengevaluasi apakah parameter PID yang didapatkan menghasilkan sinyal kontrol yang amplitudonya melebihi kemampuan fisik dari aktuator yang akan digunakan. Adapun simulasi Proteus akan digunakan untuk mengimplementasi dan menguji PID digital hasil rancangan tersebut. Meskipun pengujian dilakukan secara simulasi, adanya komponen mikrokontroler didalam Proteus yang dapat diisi program PID [14], menjadikan pengujian lebih mendekati kondisi real dibandingkan dengan pengujian yang hanya dilakukan melalui Matlab. Pengujian secara simulasi juga dapat mengurangi kerumitan yang tidak perlu yang justru dapat mengganggu fokus penelitian [15], [16], [17]. Adanya komponen Graph didalam Proteus juga akan mempermudah evaluasi karakteristik respon, tidak serumit seperti saat pengujian dilakukan secara langsung pada hardware yang tentu akan membutuhkan perangkat pencatat respon transien, seperti *storage osciloscope* atau *datalogger*.

Artikel ini diharapkan dapat memberikan gambaran lebih jelas terhadap PID digital, tidak hanya dari sisi teori namun juga dari sisi implementasinya. Hal-hal penting yang ingin dikemukanan dalam artikel ini mencakup: bagaimana cara mendapatkan parameter PID yang tepat menggunakan skrip Matlab dengan mempertimbangkan kemampuan maksimal aktuator; bagaimana menyusun kode program PID berdasarkan rumus PID digital; bagaimana cara menyusun program PID tersebut supaya waktu *sampling* bernilai tetap dan akurat, tidak terpengaruh oleh perubahan aliran program yaitu nilainya sama dengan yang digunakan dalam perancangan sehingga respon yang dihasilkan sama dengan respon saat perancangan.

2. Metode Penelitian

Penelitian ditujukan untuk mengevaluasi apakah implementasi PID digital telah dilakukan dengan benar, yaitu dengan membandingkan respon PID di dalam Matlab dengan respon PID digital di dalam Proteus. Jadi penelitian ini tidak membahas cara menemukan parameer PID yang baik untuk plant tertentu. Karena itu tahapan penelitian mencakup: menyusun skrip Matlab pencari parameter PID digital metode standart Matlab; menurunkan persamaan beda dari rumus PID digital dan menyusun kode program PID untuk mikrokontroler menggunakan bahasa C WinAVR; membuat rangkaian simulasi sistem kendali PID digital didalam Proteus dengan

ISSN (e): 2540-9123
ISSN (p): 2502-1982

kontroler berupa mikrokontoler yang diisi program PID hasil rancangan; menguji parameter PID digital hasil skrip Matlab pada Proteus dan membandingkan respon Proteus tersebut dengan respon Matlab.

2.1. Skrip Matlab Pencari Parameter PID Digital

Skrip Matlab pencari parameter PID digital (K_p, K_i, dan K_d) disusun menggunakan fungsi yang disediakan Matlab bernama pidtune. Fungsi ini menerima fungsi alih plant (G), opsi kriteria tuning (opts), dan frekuensi crossover (ω_c). Opsi kriteria yang dapat diberikan berupa gain margin (g_m) dan fokus tuning (*setpoint tracking*, *disturance rejection* atau *balanced*). Mengingat PID yang akan dibuat berupa PID digital maka fungsi alih yang dimasukkan kedalam pidtune juga harus berupa fungsi alih digital yang dihasilkan dari fungsi alih kontinyu yang di-*sampling* dengan waktu *sampling T*. Dengan cara ini akan dihasilkan parameter PID yang langsung dapat digunakan pada program PID digital. Untuk mengonversi fungsi alih kontinyu menjadi digital, digunakan fungsi c2d. Hal lain yang penting untuk dipertimbangkan ialah amplitudo maksimal sinyal kontrol atau output kontroler hasil rancangan tidak boleh melebihi kemampuan aktuator. Untuk mengetahui kondisi ini, skrip Matlab perlu dilengkapi instruksi yang akan menunjukkan nilai amplitudo maksimal sinyal kontrol (u_{max}). Dengan demikian skrip Matlab pencari parameter PID digital dapat disusun seperti dalam Gambar 1.

```
1  uc = 15;    % setpoint
2  orde = 2;   % set orde=1 jika plant adalah orde 1
3  gain = 0.5; % gain plant
4  tau = 1;    % konstanta waktu orde-1
5  si = 1.707; % konstanta redaman orde-2
6  wn = 0.8;   % freq natural orde-2
7
8  T  = 0.05;  % waktu sampling
9  wc = 0.4;   % frekuensi crossover (rad/s)
10 gm = 40.0;  % gain margin
11
12 focus = 1;  % (1)tracking setpoint, (2) penolak gangguan, (3) antara 1&2
13 if orde==1 G = tf(gain, [1/tau 1]);
14 else G = tf(gain*wn*wn, [1 2*si*wn wn*wn]); end
15 focuses = ["reference-tracking","balanced", "disturbance-rejection"];
16 opts = pidtuneOptions('PhaseMargin', gmargin, 'DesignFocus', focuses(focus));
17 C = pidtune(G,'pid', wc, opts);
18 Gcs = feedback(C*G,1);
19 Gz = c2d(G,T,'zoh');
20 Cz = pidtune(Gz, 'pid', wc, opts);
21 Gcz = feedback(Cz*Gz,1);
22 M = uc*feedback(C*tf(1,[1e-5 1]),G);
23 step(M,'r-'), grid on, title('Output PID')
24 [m,t]=step(M); m_max=max(m)
25 step(uc*G,'b-',uc*Gcs,'k-.', uc*Gcz,'k-'), grid on, legend('plant','PIDs','PIDz')
26 stepinfo([uc*Gz uc*Gcz])
27 display(Cz)
```

Gambar 1. Skrip Matlab pencari parameter PID.

Baris 1 menentukan nilai *setpoint*. Baris 2 memilih orde plant yang akan dikontrol sebagai orde satu atau dua. Jika plant yang akan dikontrol merupakan sistem orde satu yang memiliki penguatan sebesar *gain* dan konstanta waktu τ, seperti berikut.

$$G = \frac{gain}{\tau s + 1} \tag{1}$$

atau merupakan sistem orde dua yang memiliki konstanta redaman ζ dan frekuensi natural ω_n, seperti berikut.

$$G = \frac{gain \cdot \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2}$$

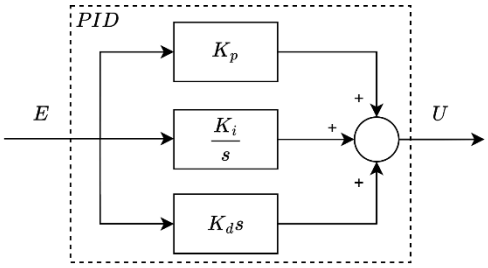
ISSN (e): 2540-9123
ISSN (p): 2502-1982

Maka nilai $gain, \tau, \zeta,$ dan ω_n masing-masing dapat diseting menggunakan instruksi baris 3-6. Adapun waktu *sampling*, frekuensi *crossover*, dan *gain margin* masing-masing diseting oleh pengguna melalui instruksi baris 8-10. Nilai-nilai ini harus dipilih secara bijak, misalnya waktu *sampling* tidak layak bila dipilih terlalu besar namun juga tidak bisa dipilih terlalu kecil karena dibatasi oleh kecepatan pemrosesan mikrokontroler. Baris 11 digunakan untuk memilih fokus tuning. Matlab menyediakan tiga pilihan fokus *tuning* seperti disebutkan pada baris 16. Jika membutuhkan tuning sebagai *reference tracking* maka varibel fokus harus diseting 1. Baris 17 dan 18 masing-masing untuk menghitung parameter PID kontinyu dan menghitung fungsi alih loop tertutup kontinyu. Meskipun yang sedang dibahas adalah PID digital, fungsi alih kontinyu tetap digunakan karena akan digunakan sebagai pembanding. Rancangan versi digital yang benar ialah yang grafik responnya tidak menyimpang terlalu jauh dari versi kontinyu. Baris 19-21 masing-masing untuk mengonversi fungsi alih kontinyu menjadi digital, menghitung parameter PID digital dan menghitung fungsi alih loop tertutup digital. Baris 22-24 untuk menghitung rumus output kontroler, menampilkan grafik output kontroler dan menghitung nilai maksimalnya. Baris 25 menampilkan respon sistem loop terbuka, sistem loop tertutup kontinyu dan sistem loop tertutup digital. Baris 26 mentabelkan karakteristik waktu sistem loop terbuka dan sistem loop tertutup. Baris 27 menampilkan parameter kontroler PID digital hasil tuning.

2.2. Kode Program PID Digital

Untuk mendapatkan kode program PID digital yang akan dijalankan pada mikrokontroler perlu melakukan tahapan: 1) mengonversi persamaan PID analog menjadi bentuk digital, 2) mengonversi persamaan PID digital menjadi persamaan beda (*difference equation*); dan 3) menulis kode program berdasarkan persamaan beda yang dihasilkan sebelumnya.

Terdapat beberapa macam rumus PID kontinyu, namun yang paling umum digunakan adalah bentuk paralel seperti ditunjukkan dalam Gambar 2 [18].



Gambar 2. Diagram blok kontroler PID paralel.

Fungsi alih kontroler ini dapat dinyatakan sebagai berikut.

$$PID = \frac{U}{E} = K_p + \frac{K_i}{s} + K_d s \tag{3}$$

PID ini memiliki input E, output U, dan parameter PID, masing-masing $K_p, K_i,$ danK_d. Konversi bentuk kontinyu ke bentuk digital dapat dilakukan melalui beberapa pendekatan, seperti: Euler Backward, Euler Forward, Tustin/Bilinier, dan Matched Pole-Zero [7]. Jika konversi dilakukan menggunakan Euler Backward dengan waktu *sampling* T maka semua *s* dalam (3) dapat diganti dengan persamaan berikut.

$$\mathrm{s} = \frac{1 - z^{-1}}{T} \tag{4}$$

sehingga (3) berubah menjadi seperti berikut.

$$PID = \frac{U}{E} = K_p + \frac{K_i T}{1 - z^{-1}} + K_d \left(\frac{1 - z^{-1}}{T} \right) \tag{5}$$

atau

$$\mathrm{U} = K_p \mathrm{E} + \frac{K_i T E}{1 - z^{-1}} + K_d \mathrm{E} \left(\frac{1 - z^{-1}}{T} \right) \tag{6}$$

ISSN (e): 2540-9123
ISSN (p): 2502-1982

Jika setiap suku (6) masing-masing diuraikan menjadi P, I dan D, maka didapatkan:

$$P = K_p\mathrm{E};\quad \mathrm{I} = \frac{K_i TE}{1 - z^{-1}};\quad D = K_d\mathrm{E}\left(\frac{1 - z^{-1}}{T}\right) \tag{7}$$

Sehingga bagian P dapat dikonversi menjadi persamaan beda berikut.

$$p(k) = K_p e(k) \tag{8}$$

Bagian I dapat diubah menjadi bentuknya menjadi seperti berikut.

$$\begin{aligned} \mathrm{I} &= \frac{K_i TE}{1 - z^{-1}} \\ I(1 - z^{-1}) &= K_i TE \\ I &= Iz^{-1} + K_i TE \end{aligned} \tag{9}$$

Sehingga bagian I dapat dikonversi menjadi persamaan beda berikut.

$$i(k) = i(k-1) + K_i Te(k) \tag{10}$$

Terakhir, bagian D dapat dikonversi menjadi persamaan beda berikut.

$$\mathrm{d}(k) = K_d\big(e(k) - e(k-1)\big)/T \tag{11}$$

Persamaan beda tersebut selanjutnya diwujudkan menjadi kode program PID, seperti ditunjukkan dalam Gambar 3.

```
1  void pid(){
2      static int i = N_PI;
3      if(++i<N_PI)
4          return;
5      i = 0;
6
7      const float T = tick*N_PI;
8      static float e1 = 0.0;
9      static float I = 0.0;
10     const float UMAX = 255.0;
11     const float UMIN = 0.0;
12
13     float y = 5.0*(float)adc_read(0)/1023.0;
14     y /= sensor_gain;
15
16     float e = uc - y;
17     float P = Kp*e;
18     I += Ki*T*e1;
19     float D = Kd*(e-e1)/T;
20     float u = P+I+D;
21     e1 = e;
22
23     u *= 255.0/(5.0*actuator_gain);
24
25     if(I>UMAX) I = UMAX;
26     else if(I<UMIN) I=UMIN;
27     if(u>UMAX) u = UMAX;
28     else if(u<UMIN) u=UMIN;
29
30     pwm_writeA(u);
31 }
```

Gambar 3. Kode program PID.

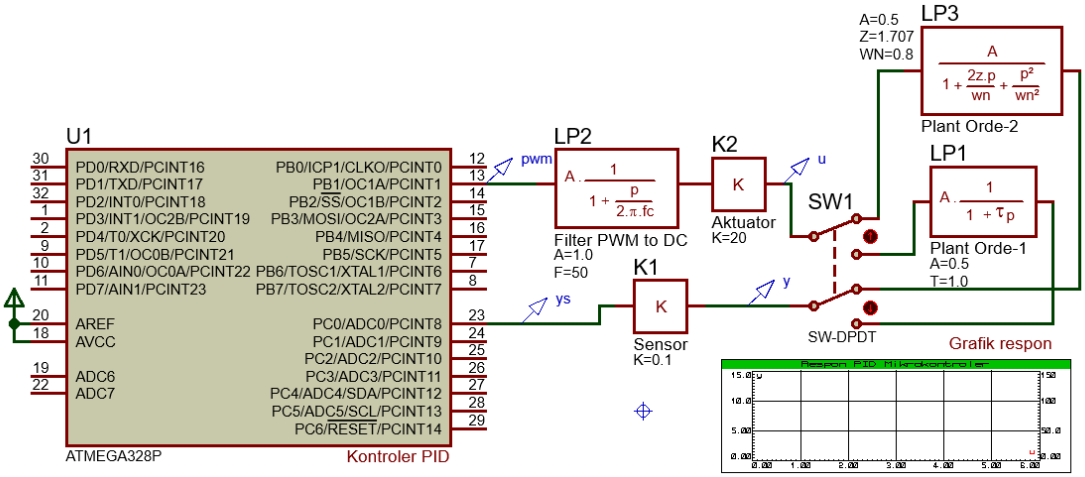
Didalam program tersebut, *e(k)* dinyatakan oleh variabel *e* dan dihitung menggunakan instruksi baris 16, yaitu setpoint u_c dikurangi output *y*. Nilai setpoint merupakan nilai yang diseting secara tetap didalam program sedangkan nilai *y* dihitung menggunakan instruksi baris 13-14. Baris 13 membaca ADC channel 0 dan mengonversinya menjadi tegangan maksimal 5V. Baris 14 membagi nilai tegangan tersebut dengan sensor_gain sehingga didapatkan nilai output plant. Dengan demikian (8), (10) dan (11) masing-masing dapat dinyatakan menggunakan instruksi baris 17-19 dan output PID u dinyatakan menggunakan instruksi baris 20. Waktu *sampling T* dinyatakan oleh instruksi baris 7, yaitu tick dikalikan *N_PI*, dimana tick bernilai $0{,}99ms \approx 1ms$ (dibuat menggunakan interupsi timer), sedangkan *N_PI* dapat diseting sesuai keperluan. Jika diinginkan waktu *sampling* PID digital bernilai 50ms maka *N_PI* harus diseting 50. Baris 21 diperlukan untuk mencatat error sekarang e(k) sebagai error sebelumnya *e(k-1)* sehingga *e(k-1)*

ISSN (e): 2540-9123
ISSN (p): 2502-1982

dapat digunakan pada siklus penghitungan PID berikutnya. Terlihat bahwa program menggunakan variabel *e1* untuk mencatat *e(k-1)*. Supaya nilai *e1* selalu diingat atau tidak hilang dari satu pemanggilan fungsi pid() ke pemanggilan fungsi pid() beriktunya maka *e1* harus didefinisikan sebagai variabel statik oleh instruksi baris 8. Hal yang sama berlaku untuk variabel I seperti ditunjukkan oleh instruksi baris 9. Output PID *u* hasil baris 20 masih harus diolah lebih lanjut supaya terkirim dengan benar pada aktuator. Jika aktuator memiliki penguatan sebesar actuator_gain, dan sinyal *u* akan dikirim dalam bentuk PWM 8bit maksimal 5V maka sinyal u harus dikalikan dengan 255/(5.0 * actuator_gain) seperti ditunjukkan instruksi baris 23. Karena instruksi ini dapat menghasilkan nilai di luar jangkauan 0-255 sedangkan pembangkit PWM hanya menerima nilai 0-255 maka diperlukan instruksi baris 27-28 untuk membatasi nilai u supaya berada dalam jangkauan tersebut. Demikian juga supaya nilai integral tidak terus bertambah atau terus berkurang melebihi batas pembangkit PWM maka nilai integral juga harus dibatasi seperti ditunjukkan instruksi baris 25-26. Sinyal *u* dikirm ke pembangkit PWM menggunakan instruksi baris 30.

2.3. Diagram Simulasi Kontrol PID

Diagram simulasi Kontrol PID digital dapat disusun seperti dalam Gambar 4. Tujuan utama simulasi ini bukan untuk membahas cara menala PID namun untuk memverifikasi hasil implementasi PID digital, apakah sudah benar yang dibuktikan dengan cara membandingkan respon simulasi ini dengan respon simulasi Matlab. Jika kedua respon sudah sama maka dapat disimpulkan bahwa implementasi PID digital telah benar. Plant atau bagian yang akan dikontrol diwujudkan menggunakan blok LP1 untuk plant orde-1 dan LP3 untuk plant orde-2. Blok-blok ini terbuat dari komponen *low pass filter* dengan parameter yang dapat diatur melalui Property kedua blok tersebut sehingga membentuk (1) dan (2). Plant mana yang akan digunakan dalam simulasi dapat dipilih melalui switch SW1. Sensor dan aktuator masing-masing diwujudkan menggunakan blok K1 dan K2 yang terbuat dari komponen gain. Nilai penguatan kedua blok ini dapat diatur melalui property kedua blok tersebut. Adapun blok LP2 digunakan untuk memfilter sinyal PWM yang dihasilkan mikrokontroler (U1) supaya menjadi sinyal DC 0-5V. Sinyal PWM itu merupakan sinyal output algoritma PID didalam mikrokontroler. Simulasi ini dilengkapi dengan Grafik Respon yang terbuat dari komponen Graph. Ia digunakan untuk menampilkan sinyal output plant *y* dan output kontroler *u*. Terlihat bahwa mikrokontroler U1 yang bekerja sebagai kontroler PID membaca output plant melalui ADC channel 0 (pin ADC0) dan mengeluarkan hasil PID melalui pin OC1A.



Gambar 4. Diagram simulasi PID digital didalam Proteus.

Output plant *y* dibaca dan diolah oleh mikrokontroler sekali setiap waktu *sampling T* sesuai dengan program dalam Gambar 3. Salah satu cara untuk menjamin bahwa kode program PID tersebut benar-benar dijalankan sekali setiap waktu *sampling T* ialah dengan menstart ADC

ISSN (e): 2540-9123
ISSN (p): 2502-1982

melalui counter dan mengonfigurasi ADC supaya mengalami interupsi pada akhir konversi. Kemudian program PID dan pengiriman sinyal kontrol hasil PID ke aktuator dilakukan dari vektor interupsi ADC tersebut [19]. Artikel ini menggunakan cara lain yaitu menjalankan proses PID melalui vektor interupsi timer seperti ditunjukkan dalam Gambar 5.

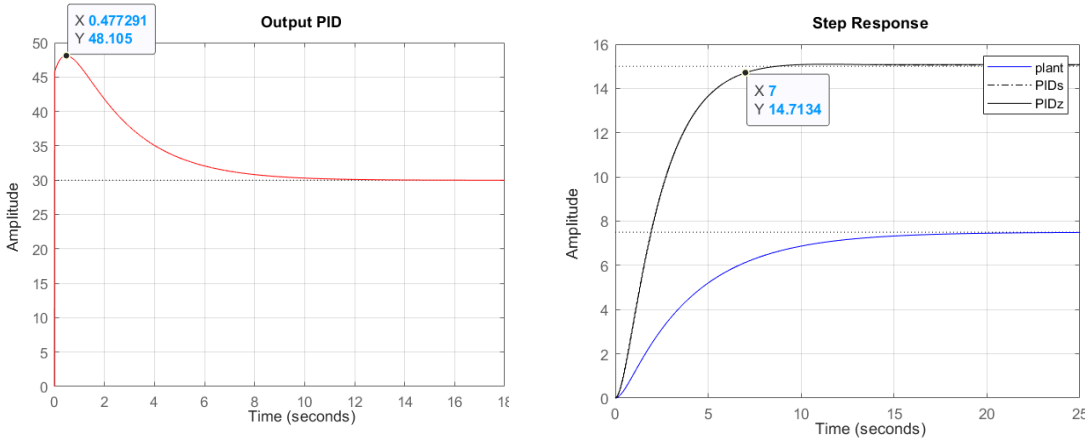
```
33 #define TCNT0_PRELOAD   62
34 #define TCNT0_PRESCALE  256
35 #define TCNT0_CS(PS)    (PS==1?1:(PS==8?2:\
36                          (PS==64?3:(PS==256?4:\
37                          (PS==1024?5:(0))))))
38 //prescale 256--->16MHz/256/62=1008Hz -->tick=0.992
39 const float tick        = 1.0/(F_CPU/TCNT0_PRESCALE/TCNT0_PRELOAD);
40
41 ISR(TIMER0_OVF_vect){
42     TCNT0 = 255+1-TCNT0_PRELOAD;
43     pid();
44 }
45
46 //overflow setiap 1ms
47 void timer0ovf_init(){
48     TCNT0 = 255+1-TCNT0_PRELOAD;
49     TCCR0A = 0;
50     TCCR0B = (TCNT0_CS(TCNT0_PRESCALE)<<CS00);
51     TIMSK0 =(1<<TOIE0);
52     sei();
53 }
```

Gambar 5. Kode program pengatur waktu *sampling*.

Baris 48-50 digunakan untuk mengonfigurasi timer0 supaya mengalami overflow setiap 0.992ms ≈1ms. Sedangkan instruksi 51-52 mengijinkan interupsi timer0 overflow. Dengan cara ini maka instruksi 42-43 (termasuk fungsi pid() Gambar 3) akan dijalankan sekali setiap 1ms. Sementara itu adanya instruksi baris 2-5 Gambar 3 menjadikan instruksi terkait kontroler PID (baris 7-30 Gambar 3) akan dijalankan sekali setiap N_PI.

3. Hasil dan Pembahasan

Hasil eksekusi skrip Matlab Gambar 1 ditunjukkan dalam Gambar 6. Grafik Output PID (sebelah kiri) menampilkan sinyal u yang memiliki nilai maksimal 48,1. Nilai ini masih dibawah nilai maksimal aktuator yang digunakan dalam simulasi Proteus yang bernilai 5V x actuator_gain = 5V x 20 = 100V sehingga parameter PID yang dihasilkan layak untuk digunakan. Grafik Step Respon (sebelah kanan) menampilkan output plant, output PID kontinyu (PIDs) dan output PID digital (PIDz). Terlihat bahwa output keadaan mantap PID sama antara kontinyu dan digital dan bernilai sama dengan setpoint, yaitu 15.




```
m_max = 48.1052
```

ans = 1×2 struct

Fields	RiseTime	TransientTime	SettlingTime	SettlingMin	SettlingMax	Overshoot	Undershoot	Peak	PeakTime
1	8.5500	**15.5500**	15.5500	6.7554	7.4983	0	0	7.4983	32.9500
2	4.3000	**7**	7	13.5439	15.0937	0.6247	0	15.0937	11.3500

```
Cz =
                Ts             z-1
  Kp + Ki * ------ + Kd * ------
              z-1              Ts
  with Kp = 3.08, Ki = 0.865, Kd = 0.0765, Ts = 0.05

Sample time: 0.05 seconds
Discrete-time PID controller in parallel form.
```

Gambar 6. Hasil eksekusi skrip Matlab.

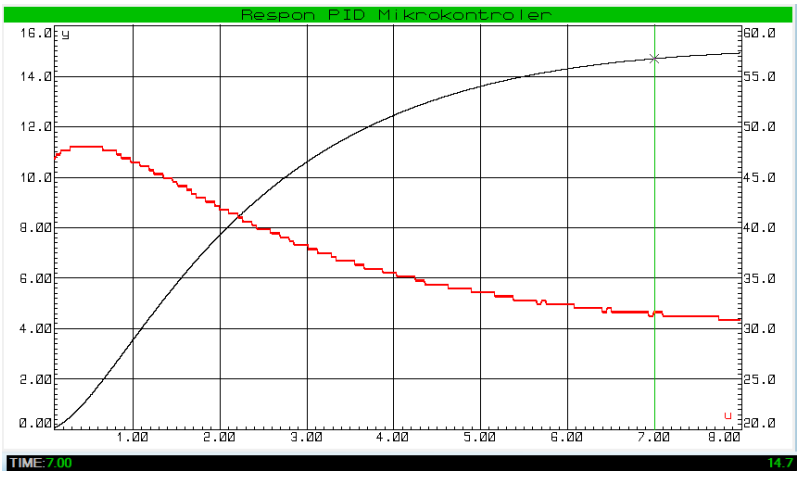
Berdasarkan tabel dan data dibawah grafik tersebut terlihat bahwa waktu mantap (Transient Time) plant 15,55s sedangkan waktu mantap (Transient Time) sistem kendali PID 7s. Adapun nilai *Kp=3.08, Ki=0,865, Kd=0,0765* dan *T=0,05s*, dapat mempercepat waktu mantap lebih dari 2 kali lipat. Pada saat mencapai waktu mantap, output sistem kendali bernilai 14,7 (98% y_{ss}).

Untuk mendapatkan respon Proteus Gambar 4, perlu memasukkan nilai *Kp*, *Ki*, *Kd*, dan *Ts* hasil skrip Matlab Gambar 6 kedalam kode program PID, seperti ditunjukkan dalam Gambar 7.

```
20 //PID controller
21 volatile float uc = 15.0;    //setpoint
22 volatile float Kp=3.08;      //Kp
23 volatile float Ki=0.865;     //Ki
24 volatile float Kd=0.0765;    //Kd
25 const int N_PI = 50;         //waktu sampling dalam milidetik
```

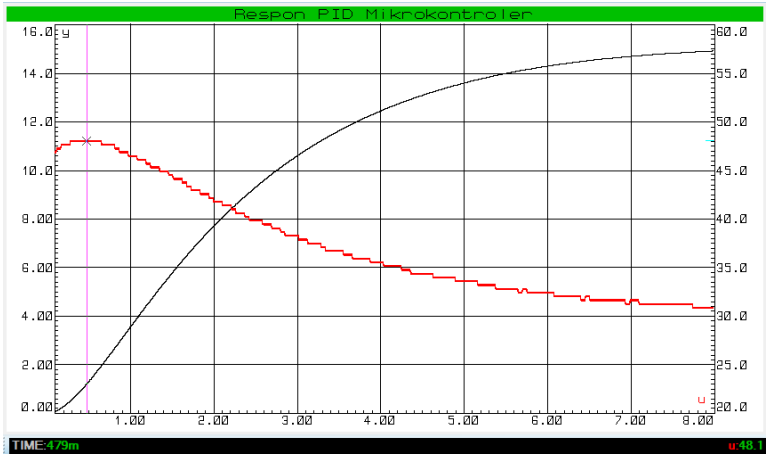
Gambar 7. Instruksi untuk menyeting parameter PID.

Grafik respon Proteus dapat diperoleh dengan menjalankan simulasi terlebih dahulu melalui menu Graph submenu Simulate Graph sehingga didapatkan hasil seperti dalam Gambar 8.



Gambar 8. Hasil eksekusi simulasi PID didalam Proteus.

Terlihat bahwa respon ini sama dengan respon Matlab Gambar 6 grafik Step Respon, yaitu pada t=7s output y=14,7 (seperti ditunjukkan angka pada pojok kanan bawah Gambar 8). Adapun nilai puncak sinyal kontrol u dapat diperoleh dengan cara klik grafik merah pada puncak tertingginya dan hasilnya ditunjukkan dalam Gambar 9.



Gambar 9. Nilai puncak sinyal kontrol dalam Proteus.

Terlihat pada pojok kanan bawah bahwa sinyal kontrol memiliki nilai puncak 48,1 sama seperti hasil simulasi Matlab Gambar 6 grafik Output PID. Hal ini membuktikan bahwa PID digital hasil rancangan skrip Matlab berhasil diimplementasikan pada mikrokontoler AVR.

4. Kesimpulan

Perangkat pengembangan kontroler PID digital menggunakan skrip Matlab dan Proteus telah berhasil dikembangkan. Hasil pengujian menunjukkan bahwa skrip Matlab berhasil menghitung nilai Kp, Ki dan Kd untuk fungsi alih plant dan kriteria perancangan yang ditentukan, seperti: waktu *sampling*, frekuensi *crossover*, gain margin dan fokus tuning. Skrip Matlab itu juga dapat menunjukkan nilai maksimal sinyal kontrol. Fitur ini dapat digunakan untuk mengevaluasi layak tidaknya parameter PID yang dihasilkan. Jika nilai maksimal tersebut lebih kecil dari kemampuan aktuator maka parameter PID tersebut dapat digunakan. Jika tidak, maka skrip Matlab perlu dijalankan kembali dengan frekuensi *cossover* yang lebih kecil hingga didapatkan amplutudo sinyal kontrol yang lebih kecil daripada kemampuan maksimal aktuator. Hasil pengujian juga menunjukkan bahwa program PID telah bekerja seperti yang diharapkan yaitu memberikan respon sama dengan respon Matlab. Dibuktikan oleh selisih output pada waktu mantap menggunakan waktu *sampling* 10ms hanya 0.0134 atau 0.089% dari setpoint. Hal ini disebabkan oleh kesesuaian antara parameter PID dan waktu *sampling* yang ada didalam Matlab dengan yang ada didalam Proteus. Penggunaan interupsi timer terbukti dapat menghasilkan waktu *sampling* PID yang tetap dan akurat, tidak bertambah atau berkurang karena perubahan aliran program.

Ucapan Terima Kasih

Tim peneliti menyampaikan terima kasih kepada P3M Politeknik Negeri Malang atas dana Riset Terapan Inovasi 2024 melalui DIPA Nomor SP DIPA 023.18.2.677606/2024 dengan surat perjanjian nomor 5575/PL2.1/HK/2024 sehingga penelitian berjalan sesuai rencana.

Referensi

[1] T. Alamirew, V. Balaji, and N. Gabbeye, “Comparison of PID controller with model predictive controller for milk pasteurization process,” *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 1, pp. 24–35, 2017, doi: 10.11591/eei.v6i1.575.

[2] H. Toar, E. Purwanto, H. Oktavianto, R. Ridwan, and M. R. Rusli, “Penala Parameter Pid Otomatis Pada Pengatur Kecepatan Motor Induksi Tiga Fasa,” *Jurnal Integrasi*, vol. 12, no. 1, pp. 1–12, 2020, doi: 10.30871/ji.v12i1.1372.

[3] P. H. G. Coelho, J. F. M. Amaral, Y. C. Bacelar, E. N. Rocha, M. Bentes, and T. S. Souza, “Tuning Analog PID Controllers by Multi-Objective Genetic Algorithms with Fuzzy Aggregation,” *International Conference on Enterprise Information Systems, ICEIS - Proceedings*, vol. 1, no. Iceis, pp. 563–571, 2023, doi: 10.5220/0011976900003467.

[4] N. Roongmuanpha, J. Satansup, T. Pukkalanun, and W. Tangsrirat, "Design of Mixed-Mode Analog PID Controller with CFOAs," *Sensors*, vol. 24, no. 10, p. 3125, 2024, doi: 10.3390/s24103125.
[5] P. Podržaj, "Contionuous VS discrete PID controller," *2018 IEEE 9th International Conference on Mechanical and Intelligent Manufacturing Technologies, ICMIMT 2018*, vol. 2018-Janua, pp. 177–181, 2018, doi: 10.1109/ICMIMT.2018.8340444.
[6] A. H. Akbar, A. Ma, C. Rekik, A. J. Abougarair, and A. Molla, "Implementing PID Control on Arduino Uno for Air Temperature Optimization," vol. 6, no. 1, pp. 1–13, 2024, doi: 10.12928/biste.v6i1.9725.
[7] M. F. V. Isdaryani, Feni Hesya and F. Feriyonika, "Sintesis Kendali PID Digital dengan Diskritisasi Langsung dan Backward Difference," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 9, no. 2, p. 467, 2021, doi: 10.26760/elkomika.v9i2.467.
[8] V. Ballerini, C. Biserni, G. Fabbri, P. Guidorzi, E. R. di Schio, and P. Valdiserri, "The Use of Arduino and PID Control Approach for the Experimental Setup of HVAC Temperature Testing," *Journal of Robotics and Control (JRC)*, vol. 5, no. 2, pp. 482–489, 2024, doi: 10.18196/jrc.v5i2.20915.
[9] B. Setiawan, M. K. Firdaus, S. Wibowo, I. Agafta, and F. Zain, "Identifiying the PID-N method performance for speed control of BLDC motor propeller on catamaran ships model," *Eastern-European Journal of Enterprise Technologies*, vol. 3, no. 2 (123), pp. 35–43, Jun. 2023, doi: 10.15587/1729-4061.2023.274713.
[10] H. Supriyono, F. F. Alanro, and A. Supardi, "Development of DC Motor Speed Control Using PID Based on Arduino and Matlab For Laboratory Trainer," *Jurnal Nasional Teknik Elektro*, vol. 1, pp. 36–41, 2024, doi: 10.25077/jnte.v13n1.1155.2024.
[11] V. F. Rahmadini, A. Ma'arif, and N. S. Abu, "Design of Water Heater Temperature Control System using PID Control," *Control Systems and Optimization Letters*, vol. 1, no. 2, pp. 111–117, Aug. 2023, doi: 10.59247/csol.v1i2.41.
[12] P. Chotikunnan and R. Chotikunnan, "Dual Design PID Controller for Robotic Manipulator Application," *Journal of Robotics and Control (JRC)*, vol. 4, no. 1, pp. 23–34, 2023, doi: 10.18196/jrc.v4i1.16990.
[13] N. Ramadhani, A. Ma'arif, and A. Çakan, "Implementation of PID Control for Angular Position Control of Dynamixel Servo Motor," *Control Systems and Optimization Letters*, vol. 2, no. 1, pp. 8–14, 2024, doi: 10.59247/csol.v2i1.40.
[14] A. Rahman and T. Taali, "Simulator Rangkaian Mikrokontroler Arduino Uno sebagai Media Pembelajaran menggunakan Proteus," *Jurnal Pendidikan Teknik Elektro*, vol. 04, no. 01, pp. 125–132, 2023.
[15] S. Buwarda, "Development of a Closed-Loop Controller for an Arduino-Based DC Chopper using Proteus and LabVIEW Research Methodology," vol. 21, no. 1, pp. 24–27, 2024.
[16] S. Syahminan and C. W. Hidayat, "Development of digital engineering learning with proteus software media and emulators department of informatics engineering Kanjuruhan University," *J Phys Conf Ser*, vol. 1869, no. 1, 2021, doi: 10.1088/1742-6596/1869/1/012076.
[17] A. R. Suharso, A. Nuriyanis, A. Hendartono, E. Sirait, and F. S. Kurniawan, "Analysis of The use of Proteus Software as a Practical Learning Support," vol. 7, no. 1, pp. 30–39, 2024.
[18] P. Mohindru, "Review on PID, fuzzy and hybrid fuzzy PID controllers for controlling non-linear dynamic behaviour of chemical plants," *Artif Intell Rev*, vol. 57, no. 4, 2024, doi: 10.1007/s10462-024-10743-0.
[19] K. Sozański, "Low Cost PID Controller for Student Digital Control Laboratory Based on Arduino or STM32 Modules," *Electronics (Switzerland)*, vol. 12, no. 15, 2023, doi: 10.3390/electronics12153235.