Pengujian Sistem Pengendalian Temperatur pada Prototipe *Heat exchanger* Berbasis PID

[1]Hana Amelinda Azizah, [2]Asepta Surya Wardhana*), [3]Chalidia Nurin Hamdani

[1,2,3]Teknik Instrumentasi Kilang, Politeknik Energi dan Mineral Akamigas,

Jl. Gajah Mada No. 38 Cepu, Kabupaten Blora, 58315

aseptasw@esdm.go.id)*

Abstrak

Rancang bangun sistem pengendalian temperatur *heat exchanger* berbasis PID artinya mengendalikan temperatur air dingin keluaran *heat exchanger* yang telah dirancang agar sesuai dengan nilai *set point* melalui pengendalian *mode controller* PID (*Proportional Integral Derivative*). Pengendalian temperatur dilakukan dengan melibatkan pengendalian laju aliran air panas. Ketika set point temperatur air dingin *heat exchanger* dinaikkan maka *servo valve* semakin membuka sehingga laju aliran air panas bertambah. Software Delphi 7 digunakan untuk menampilkan hasil berupa angka dan grafik dari data proses sistem pengendalian temperatur. Nilai parameter mode kontroler PID didapat melalui metode trial and error. Sebagai pembanding, dilakukan perhitungan PID ideal menggunakan metode *direct synthesis* untuk diterapkan pada kalkulasi pengendalian sistem di dalam mikrokontroler Arduino. Perhitungan PID ideal memerlukan hasil *bump test* dari masing–masing variabel proses. Hasil analisis data dan grafik *interface* menunjukkan bahwa kenaikan *set point* dari 25°C menjadi 32°C pada temperatur air keluaran *heat exchanger* mengakibatkan bukaan *servo valve* mendekati bukaan penuh dan berada pada kondisi *hunting* system saat temperatur telah mencapai set point. Dari hasil pengujian diperoleh *Time Constant* 89,744 detik, *Settling Time* 127,232 detik dan *Rise Time* 127,8 detik.

Kata Kunci: Temperatur, *PID*, *Cascade*, Perpindahan Panas, Delphi

1 Pendahuluan

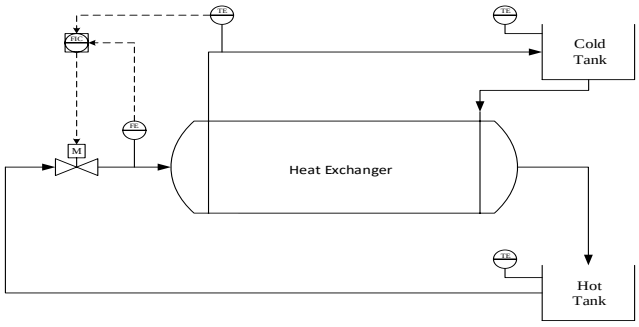
Perpindahan panas dan massa adalah ilmu dasar yang berhubungan dengan laju *transfer* energi panas. Panas adalah bentuk dari energi yang dapat berpindah dari satu sistem ke sistem lain dengan temperatur yang berbeda [1]. Beberapa alat transfer panas yang sering dijumpai diantaranya, *heat exchanger*, *boiler, condenser, furnace, radiator, heater* dengan desain berbasis analisis transfer panas. *Heat exchanger* dideskripsikan sebagai suatu unit yang dapat memindahkan panas dari aliran fluida satu ke aliran fluida yang lain melalui perpindahan energi antar fluida yang memiliki temperatur yang berbeda [2]. Untuk meningkatkan area perpindahan panas atau energi, permukaan sekunder harus melekat dengan permukaan primernya [3]. Tetapi penerapan kontrol temperatur keluaran *heat exchanger* dalam industri tidak semudah prinsip kerjanya dalam teori, masih ada kemungkinan adanya deviasi antara variabel yang dikendalikan (*controlled variable*) dan *set point* ataupun karena adanya gangguan berupa perubahan temperatur luar yang terikut masuk ke dalam *heat exchanger* [4]. Meminimalisir sistem kesalahan dan menjaga kondisi sistem agar tetap berada pada kondisi yang diinginkan dapat digunakan teknik pengendalian PID [5], [6]. *Cascade control* adalah metode terbaik pernah diterapkan untuk memperbaiki performa *control single loop* dengan melakukan stabilitas, mengurangi osilasi dan error dari *disturbance* pada variabel manipulasi. Sehingga efektivitas kontrol sistem *cascade* lebih efisien dibandingkan dengan sistem *control single loop* [7], [8]. *Controller Proportional-Integral-Derivative* (PID) merupakan *controller* yang paling banyak digunakan di berbagai bidang pengendalian otomatis. Metode Fuzzy dan kecerdasan buatan merupakan sistem pengendalian handal dan cocok untuk sistem yang rumit [9], [10]. Meskipun pengendalian lain lebih unggul dari *controller* PID, tetapi kesederhanaan dan bukti kehandalan dari *controller* PID membuat *controller* ini menjadi pilihan yang paling banyak diminati untuk menyelesaikan masalah–masalah dalam bidang pengendalian *heat exchanger* [11] dan pengendalian level [12], [13]. Dari berbagai studi literatur mengenai *cascade control* menggunakan PID, terbentuklah penelitian mengenai perancangan sistem pengendalian temperatur pada prototipe *heat exchanger* berbasis PID. Penelitian ini bertujuan untuk mengetahui respon pengendalian temperatur keluaran *heat exchanger* berdasarkan nilai parameter PID yang diberikan menggunakan metode *trial and error*.

2 Metode

2.1 Desain Perancangan

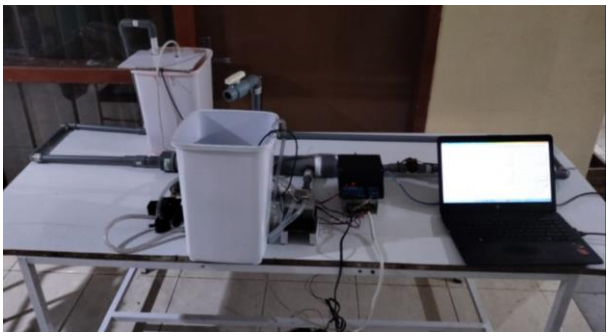
Penelitian ini berkaitan erat dengan temperatur dan laju aliran suatu fluida, dimana nilai kedua variabel proses itu saling berhubungan di dalam satu unit prototipe yang telah dirancang dan dibuat hingga sedemikian

rupa, dapat dilihat pada Gambar 1. Penelitian ini dilaksanakan berdasarkan pada prinsip kerja *heat exchanger* yang dilakukan menggunakan dua fluida dengan temperatur yang berbeda. Media atau fluida yang digunakan untuk melakukan proses penukaran panas adalah air dengan temperatur tinggi dan air temperatur rendah. Secara garis besar, cara kerja kontrol temperatur pada *heat exchanger* bergantung pada kecepatan laju aliran proses yang memiliki temperatur lebih tinggi. Semakin besar laju aliran yang masuk ke dalam *heat exchanger tube* maka semakin tinggi atau panas nilai temperatur air keluaran *heat exchanger*.



Gambar 1. Alur proses *heat exchanger*

Peralatan utama yang diperlukan pada penelitian ini diantaranya, Arduino Nano, sensor, sumber daya, s*ervo valve*, pendingin dan pemanas air, mekanik (pipa, tangki, pompa, tempat penyangga project) dan *interface*. Arduino yang memiliki beberapa pin *input / output* digital dan analog, berfungsi untuk mendeteksi serta memberi aksi perangkat eksternal [14]. Sensor yang memiliki fungsi utama mendeteksi nilai proses yang diukur secara akurat karena dalam implementasinya di lapangan, tidak ada sistem pengukuran yang sempurna, tetapi dibutuhkan sistem pengukuran dengan ketepatan akurasi yang dicapai melalui peralatan dan strategi sederhana [15]. *Servo valve* yang menggunakan sinyal umpan balik untuk mengatur kecepatan dan arah putar dari motor tersebut untuk mencapai kondisi yang diinginkan [8]. Berikut pada gambar 2 merupakan desain prototipe yang digunakan dalam penelitian perpindahan panas.

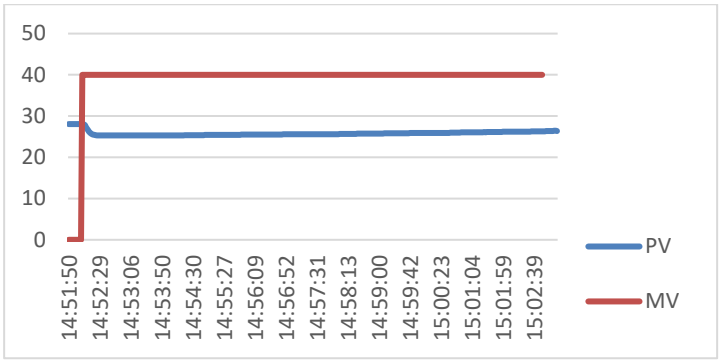


Gambar 2. Prototipe *heat exchanger*

Interface yang digunakan pada penelitian ini adalah Delphi 7 karena *software* ini memiliki bahasa pemrograman tingkat tinggi yang sederhana dibandingkan dengan kompleksitas kode yang digunakan pemrograman dengan tingkat tinggi lainnya dan beberapa penelitian Delphi dapat digunakan untuk pemrograman optimasi kecerdasan buatan [16]. Data primer dari Arduino kemudian ditampilkan pada kolom memo “Data Arduino”. Empat kolom memo di samping kiri “Data Arduino” adalah kolom memo yang akan menampilkan hasil konversi data mentah Arduino. Arti dari konversi di sini merujuk pada pemisahan berbagai jenis data dari Arduino, yang dikelompokkan menjadi satu pada satu kolom memo sesuai dengan jenis data masing–masing.

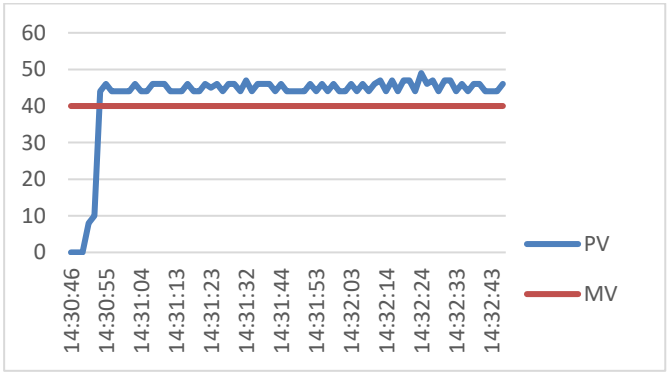
2.2 Fungsi Alih Proses

Langkah awal untuk menentukkan fungsi alih proses temperatur dan laju alir sistem *cascade* pada prototipe *heat exchanger* adalah dengan melakukan *bump test*. Pada uji coba sistem, *bump test* dilakukan dengan mengatur bukaan *servo valve* secara manual. Bukaan *valve* dinaikkan secara manual dari 0% menjadi 40%, didapatkan hasil grafik seperti pada Gambar 3. yang menunjukkan nilai MV dan PV dari proses temperatur. Kemudian dilakukan pemantauan dan pencatatan terhadap nilai temperatur dan laju alir yang dihasilkan sistem selama tujuh menit. Temperatur yang tercatat mengalami kestabilan pada $\pm$ 25.31°C.



Gambar 3. Grafik hasil *bump test* dari proses temperatur

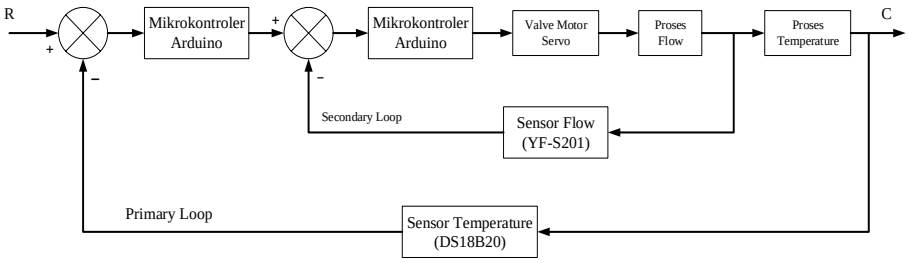
Bump test juga dilakukan pada proses laju aliran untuk mendapatkan respon proses *open loop*-nya. Respon ini dapat dilihat pada Gambar 4. yang menunjukkan nilai MV dan PV dari proses laju alir. Proses perubahan PV pada laju alir mengalami osilasi disebabkan proses buka tutup pada control valve. Sama halnya dengan proses *bump test* temperatur, *bump test* proses laju alir juga dilakukan selama tujuh menit. Dibandingkan dengan *bump test* pada temperatur maka *bump test* laju alir mempunyai perfoma osilasi mulai 2 sampai 5 ml/detik.



Gambar 4. Grafik hasil *bump test* dari proses flow

2.3 Pengendalian

Cascade control merupakan sistem pengendali yang melibatkan penggunaan dua buah pengontrol dengan keluaran dari pengontrol pertama adalah acuan pengaturan untuk pengontrol kedua seperti ditunjukkan pada gambar 5. Secara garis besar, terdapat dua fungsi utama dari *cascade control*, yaitu untuk menghilangkan efek dari gangguan, dan untuk meningkatkan kerja dinamis dari loop kontrol sistem [8]. Ketika muncul suatu *disturbance* pada *loop* sekunder (bagian dalam) maka *controller* dari loop sekunder akan segera melakukan aksi koreksi sebelum terjadi perubahan pada keluaran sistem, sedangkan kontrol *loop* primer (bagian luar) akan melakukan aksi koreksi sendiri apabila muncul *disturbance* pada *loop* bagian luar [17].

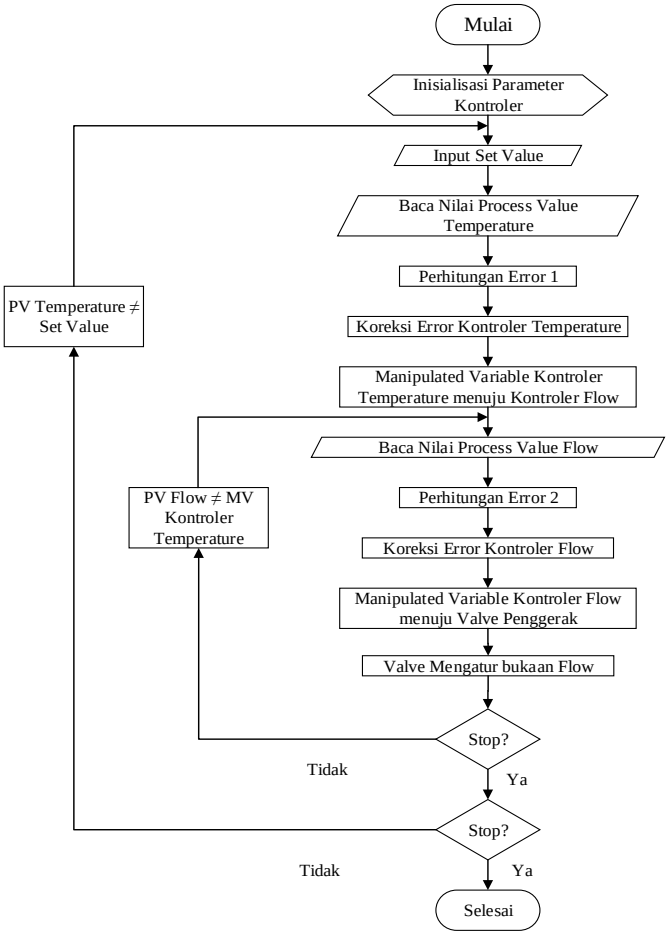


Gambar 5. Blok Diagram Cascade Control Prototipe *Heat exchanger*

Persamaan 1. merupakan representasi matematis dari PID *Controller* [7].

$$\text{MV(s)} = K_p e(s)\left[1 + \frac{1}{T_i s} + T_D s\right] + bias \quad (1)$$

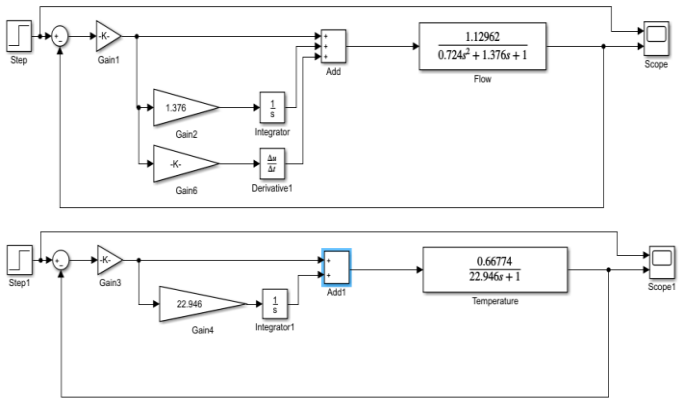
Gambar 6 merupakan proses alur dari perpindahan panas. Diawali dengan inisialisasi parameter *controller* untuk memberikan nilai awal pada parameter dari masing–masing komponen *controller proportional*, *integral*, dan *derivative*. Inisialisasi ini dapat diperoleh melalui percobaan *trial and error* untuk mengetahui keberhasilan kerja kalkulasi dari parameter *controller*. Kemudian dilanjutkan dengan memberi nilai input berupa *set point* yang berfungsi sebagai nilai acuan dalam kalkulasi mikrokontroler. Nilai *set point* yang telah ditetapkan akan dibandingkan dengan nilai aktual proses (temperatur) hasil deteksi sensor temperatur. Perbandingan nilai ini masuk ke dalam tahap perhitungan *error* untuk menghitung deviasi yang muncul akibat adanya perbedaan nilai antara *set point* dan nilai aktual sensor temperatur.



Gambar 6. *Flowchart* Proses Prototipe *Heat exchanger*

Hasil perhitungan *error* temperatur kemudian dikoreksi oleh perhitungan *controller* temperatur pada pemrograman Arduino IDE. Hasil koreksi ini menjadi nilai MV yang akan dibandingkan dengan nilai aktual laju aliran dari sensor YF–S201. Seperti halnya proses temperatur, data aktual sensor *flow* akan dibandingkan terlebih dahulu dengan nilai MV yang didapatkan dari hasil kalkulasi *controller* temperatur. Hasil perbandingan ini memunculkan nilai *error* yang akan dimasukkan ke dalam kalkulasi perhitungan *controller* laju aliran pada pemrograman Arduino IDE. Selanjutnya hasil kalkulasi tersebut menjadi nilai MV yang dapat menggerakkan bukaan *servo valve*. Selama terjadi proses kalkulasi perhitungan *controller* untuk menghasilkan nilai MV pada masing – masing variabel proses maka *servo valve* berada pada kondisi *floating* atau *hunting* sebelum sensor temperatur keluaran HE mencapai nilai *set point* yang telah ditetapkan. Kalkulasi pada masing–masing *controller* variabel proses akan terus berulang hingga *error* satu bernilai 0 (nol) atau PV temperatur keluaran HE telah sesuai dengan SP. Apabila nilai PV sukar mencapai nilai SP atau sensor temperatur keluaran HE telah mencapai nilai SP tetapi *servo valve* tetap bergerak pada kondisi *floating*, artinya perlu dilakukan inisialisasi kembali terhadap parameter *controller* masing – masing variabel proses. Respon dari perhitungan yang telah dilakukan di dalam mikrokontroler Arduino dapat dimonitor melalui *interface* Delphi 7. *Tuning controller* digunakan untuk mencari nilai–nilai parameter PID ideal dari masing–masing variabel proses. Metode ini sebagai acuan nilai PID dari masing–masing variabel proses sebelum dilakukan *trial and error*. *Tuning* berfungsi untuk menetapkan nilai PID awal sebelum prototipe diaktifkan.

Dari hasil *bump test*, fungsi alih proses dari temperatur dan laju alir dapat diperoleh melalui sistem identifikasi pada MatLab *software*. Sistem identifikasi mendapatkan persamaan fungsi alih melalui identifikasi data *bump test* yang diinputkan pada lembar kerja MatLab seperti gambar 7 dan 8. Dari sistem identifikasi tersebut, didapatkan fungsi alih temperatur keluaran *heat exchanger*.



Gambar 7. Diagram *Block* Simulink dari Proses Laju Alir Proses Temperatur

- Laju Alir

$$G_{PF} = \frac{1{,}12962}{0{,}724^2 + 1{,}376s + 1} \quad (2)$$

Diketahui:

K= 1,56 , ω_n = 1,17525, $\frac{2\zeta}{\omega_n}$ = 1,376 , ζ = 0,80857

Time Constant yang dikehendaki: $\tau^* \approx$ 2 detik

Maka diperoleh nilai P, I, dan D:

- *Integral :* $\tau_i = \frac{2\zeta}{\omega_n}$=1,376
- *Derivative :* $\tau_d = \frac{1}{2\zeta\omega_n} = \frac{1}{2x0{,}80857x1{,}17525}$ = 0,526163
- *Proportional :* $K_p = \frac{\tau_i}{\tau^* x\, K} = \frac{1{,}376}{2\, x\, 1{,}12962}$ = 0,60905

- Temperatur

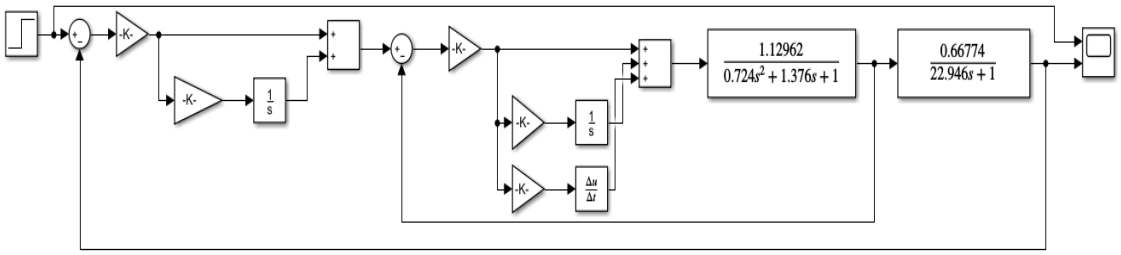
$$G_{PT} = \frac{0{,}66774}{22{,}946\, s + 1} \quad (3)$$

Diketahui

- K = 0,66774
- τ = 22,946

Maka diperoleh nilai P dan I:

- *Integral :* $\tau_i = \tau$; τ_i = 22,946
- *Proportional :* $K_p = \frac{\tau_i}{\tau^* x K} = \frac{22{,}946}{2x0{,}66774}$ = 17,18183724



Gambar 8. Diagram *Block* Simulink *Closed Loop*

Tabel 1. merupakan hasil *tuning* PID *Controller* dan nilai *trial and error*. Tuning *controller* dilakukan menggunakan metode *direct synthesis*. Tetapi apabila hasil tuning *controller* tidak dapat membuat proses stabil maka nilai parameter mode kontroller dilakukan dengan metode *trial and error.*

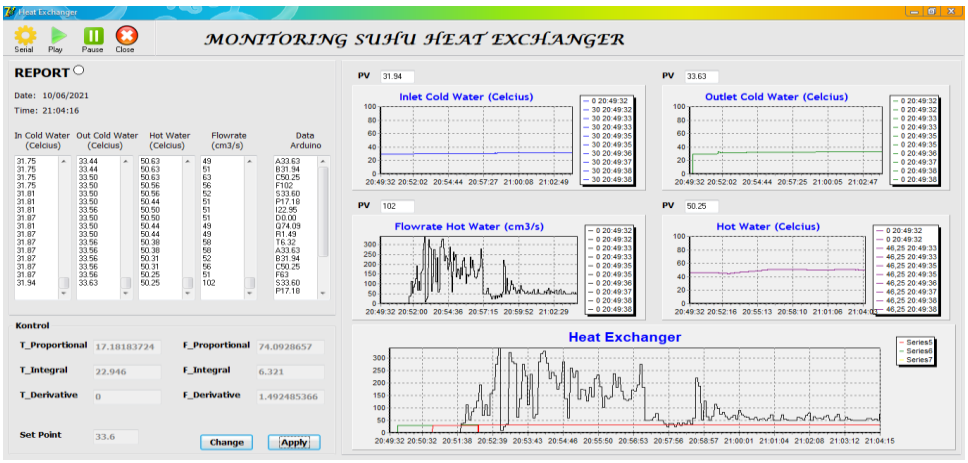
Tabel 1. Hasil tuning dan *trial and error* parameter PID *controller*

No.	T_P	T_I	T_D	F_P	F_I	F_D	SP_0	SP_1
				Hasil *Tuning*				
1.	17, 1818	22, 946	0	74, 09287	6, 321	1, 4925	32, 6	33, 6
2.	17, 1818	22, 946	10	74, 09287	6, 321	1, 4925	30, 5	31, 5
3.	17, 1818	22, 946	10	1, 290365	2, 925	1	34,25	35,25
				Hasil *Trial and error*				
4.	60	30	20	450	4, 5	1000	31, 6	32, 6
5.	17, 1818	22, 946	10	5	4	1	35,25	36,25
6.	17, 1818	22, 946	10	5	4	100	36,25	37,25
7.	17, 1818	22, 946	50	5	4	100	37, 5	38, 5
8.	17, 1818	22, 946	50	150	4	100	38, 5	39,5
9.	17	22	15	10	4, 5	2	40	41
10.	17, 1818	7	10	5	2	1	41, 1	42, 1
11.	17, 1818	7	10	2	1	0	34,94	35 9
12.	5	4	3.5	2	1	0	36, 4	37, 4
13.	5	2	1	2	1	0	37, 5	38, 5
14.	2	1	2	2	1	0	38, 7	39, 7
15.	2	1	2	10	5	5	37, 5	38, 5
16.	2	1	2	10	5	15	38, 5	39, 5
17.	2	1	2	2	1	3	39, 5	40, 5
18.	2	1	2	2	1	4	40, 5	41,5
19.	2	1	2	2	1	5	41, 5	42, 5
20.	2	1	2	10	2	0	39, 8	40, 8
21.	2	1	2	10	5	20	41, 6	42, 6
22.	2	1	2	100	5	150	32, 5	33, 5
23.	2	1	2	2	1	1	33, 9	34, 9
24.	2	1	2	10	5	2	35, 2	36, 2
25.	2	1	2	10	5	10	36, 4	37, 4

3 Hasil

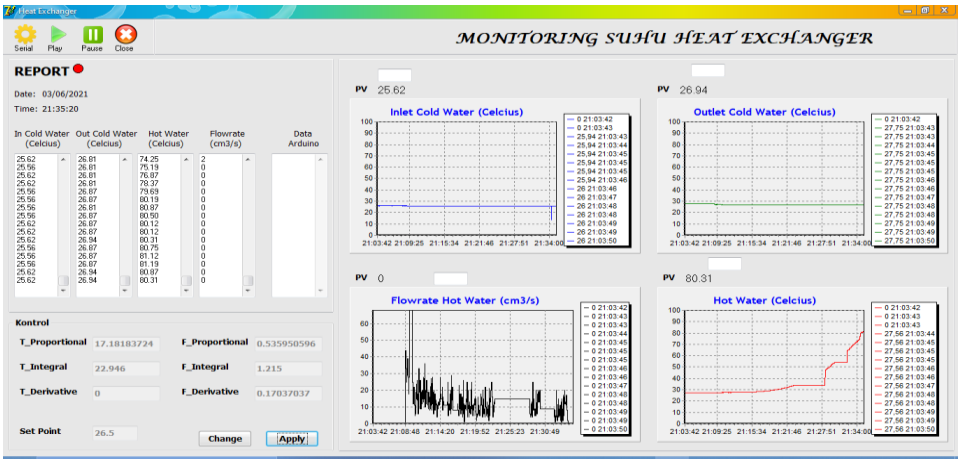
3.1 Hasil Pengujian

Uji coba pertama dilakukan menggunakan nilai parameter PID ideal yang dicari menggunakan metode *tuning direct synthesis* dengan menaikkan nilai *set point* dari 32,6℃ hingga 33,6℃. Hasil yang didapatkan ditampilkan pada Gambar 9.



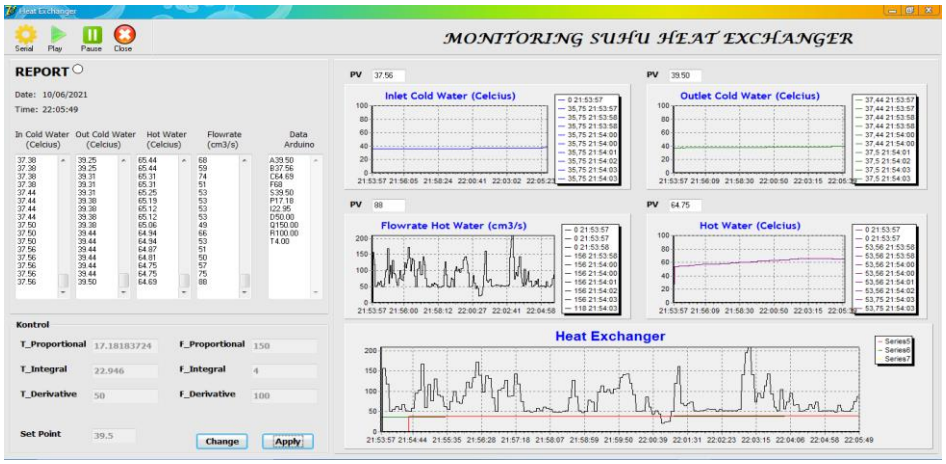
Gambar 9. Hasil *Interface* PID Ideal saat Temperatur *set point* Dinaikkan

Hasil *tuning* lainnya diterapkan pada uji coba proses seperti Gambar 10. Pengujian dilakukan dengan menurunkan nilai SP dari temperatur keluaran *heat exchanger* sebesar 1℃. Perubahan nilai SP dilakukan dari temperatur 27,5℃ menjadi 26,5℃.



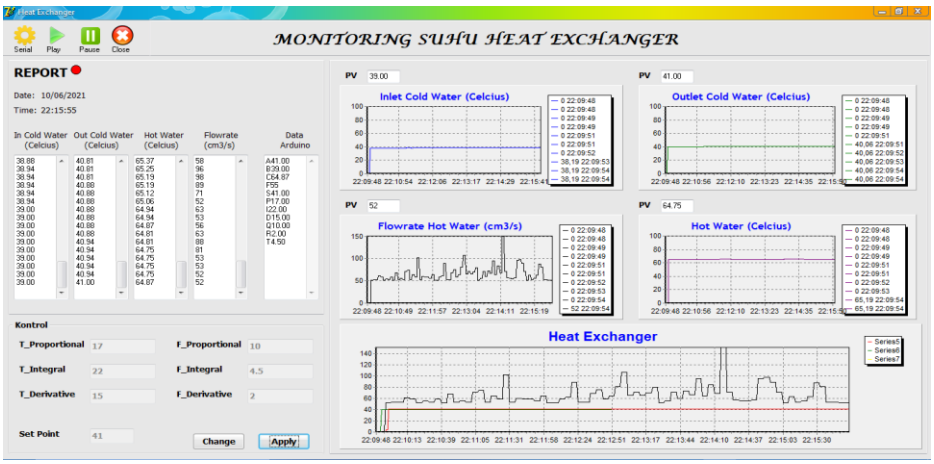
Gambar 10. Hasil *Interface* PID Ideal saat Temperatur Diturunkan

Uji coba selanjutnya dilakukan dengan menggunakan nilai parameter PID *trial and error*. Pengubahan nilai PID tidak jauh dari hasil *tuning direct synthesis*. Penerapan pengubahan PID yaitu membuat nilai *proportional* yang lebih besar dari *derivative* bersamaan dengan pengubahan nilai SP dari 38,5℃ menjadi 39,5℃. Hasil respon yang didapatkan ditampilkan pada Gambar 11.



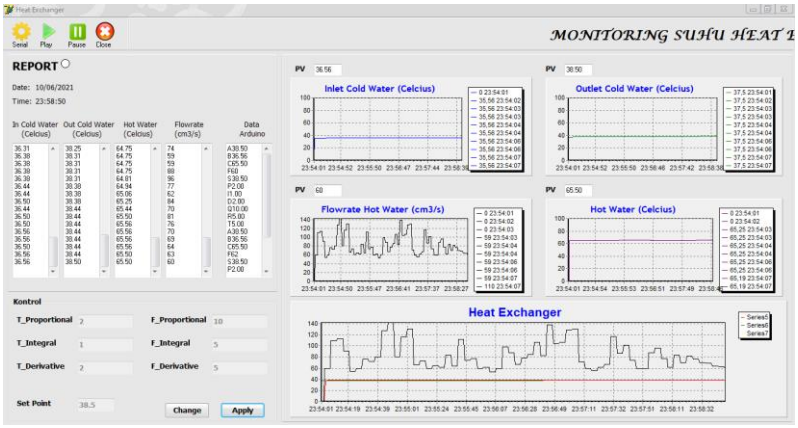
Gambar 11. Hasil *Trial and error* Saat *Proportional* Lebih Besar dari *Derivative*

Perubahan lain terhadap nilai parameter PID dilakukan dengan membuat *proportional* lebih kecil dari *derivative*. Bersamaan dengan itu, nilai SP dinaikkan dari 40℃ hingga 41℃. Respon yang dihasilkan dapat dilihat pada Gambar 12.



Gambar 12. Hasil *Trial and error* Saat *Proportional* Lebih Kecil dari *Derivative*

Uji coba lainnya, membuat nilai *integral* sama dengan *derivative*nya bersamaan dengan melakukan pengubahan terhadap nilai SP dari 37,5℃ hingga 38,5℃. Respon yang dihasilkan pada percobaan ini ditampilkan pada Gambar 13. Percobaan lain dilakukan dengan *trial and error* melalui pengubahan terhadap nilai parameter PID dari masing – masing variabel proses. Parameter – parameter ini diubah menggunakan beberapa tahapan, diantaranya membuat nilai *proportional* lebih besar dari nilai *derivative* yang hasilnya dapat dilihat pada Gambar 13. Membuat nilai *proportional* lebih kecil dari nilai *derivative*, membuat *proportional* dan *derivative* memiliki nilai yang sama, membuat *integral* dan *derivative* memiliki nilai yang sama.



Gambar 13. Hasil *trial and error* saat *Integral* Sama Dengan *Derivative*

4 Analisis

Pada salah satu metode *tuning* PID yang telah dilakukan, didapatkan hasil mode *controller* ideal untuk proses temperatur adalah PI sedangkan mode *controller* ideal untuk proses laju aliran adalah PID. Pada salah satu metode *trial and error*, parameter PID diubah dengan membuat nilai *proportional* dari laju aliran lebih besar dari nilai *derivative*-nya. Secara keseluruhan waktu yang dibutuhkan beberapa percobaan yang telah ditampilkan melalui beberapa gambar sebelumnya untuk melakukan pertukaran panas di dalam *heat exchanger* hingga mencapai nilai *set point*-nya yang ditunjukkan pada Tabel 2.

Tabel 2. Waktu yang Dibutuhkan untuk masing – masing Implementasi PID

Gambar	Kondisi PID	Waktu
Gambar 9	Nilai D Temp. = 0	12 menit (21.19 – 21.31)
Gambar 10	Nilai D Temp. = 0	13 menit (21.10 – 21.23)
Gambar 11	Nilai P *Flow* > Nilai D *Flow*	5 menit (22.00 – 22.05)
Gambar 12	Nilai P *Flow* < Nilai D *Flow*	5 menit (22.10 – 22.15)

Gambar 13 Nilai D Temp. ≠ 0 dan nilai D *Flow* = 0 4 menit (22.33 – 22.37)

Pemberian nilai nol pada *derivative* dari proses temperature akan mengakibatkan lambatnya respon temperatur keluaran *heat exchanger* untuk mencapai *set point*. Dan respon sistem berubah lebih cepat saat nilai *derivative* dari proses temperatur dinaikkan (tidak bernilai nol). Pada percobaan lain, nilai *proportional* dari proses laju alir dibuat lebih besar dibanding nilai *derivative*-nya. Akibatnya, *servo valve* lebih cepat bergerak membuka dan menutup (kondisi *floating*) apabila temperatur keluaran *heat exchanger* belum mencapai *set point*. Hal ini dapat terjadi karena proses koreksi pada kalkulasi pengendalian laju alir menjadi lebih cepat. Dampak lainnya yaitu respon laju alir mengalami fluktuasi yang tinggi selama proses pencapaian *set point*, tetapi dampak positifnya adalah *servo valve* dapat menutup penuh sehingga tidak ada aliran air panas yang mengalir menuju *heat exchanger* dan temperatur keluaran *heat exchanger* dapat mempertahankan temperatur sesuai *set point*-nya. Sebaliknya, apabila nilai *proportional* laju alir lebih kecil dibandingkan nilai *derivative*-nya, hal ini berdampak pada *servo valve* yang tidak menutup penuh saat temperatur keluaran *heat exchanger* telah mencapai *set point*.

Setelah melakukan beberapa percobaan, ditemukan nilai parameter PID untuk menghasilkan respon sistem yang sesuai yaitu memiliki respon proses laju alir yang stabil dan temperatur keluaran *heat exchanger* dapat mencapai nilai *set point* lebih cepat. Nilai parameter PID tersebut:

Temperatur:

Proportional : 17, 18183724

Integral : 7

Derivative : 10

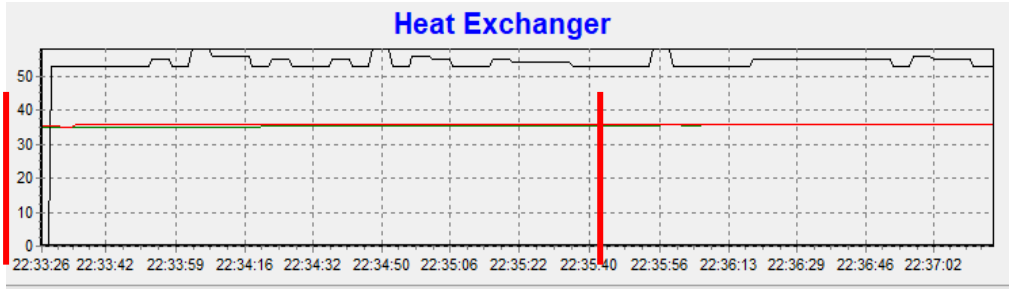
Laju alir:

Proportional : 2

Integral : 1

Derivative : 0

Dari beberapa percobaan pengubahan nilai parameter PID, dilakukan analisis terhadap waktu yang dibutuhkan untuk mencapai *Settling Time*, *Rise Time*, dan *Time Constant* dari salah satu respon yang dihasilkan. Apabila gambar grafik diperbesar maka nilai – nilai parameter tersebut dapat dilihat pada Gambar 14 secara jelas.



Gambar 14. Hasil Respon Proses dari Awal Pengubahan SP hingga Steady

Waktu yang diperlukan untuk mencapai keadaan *steady* 142 detik, dihitung dari awal terjadinya perubahan hingga respon mencapai keadaan *steady*.

Time Constant (τ)

τ = 63, 2% x waktu saat mencapai keadaan *steady*

τ = 63, 2% x 142 = 89, 744 detik (4)

Settling Time (τ_s)

$$\tau_s\ (\pm 5\%) = \ 3\ \tau \quad (5)$$

= 3 x 89, 744

= 269, 232

τ_s = 269, 232 – 142 = 127, 232 detik

Rise Time (τ_R)

τ_R (5%) = 5% x waktu saat mencapai keadaan *steady* (6)

τ_R (5%) = 5% x 142 = 7,1 detik

τ_R (95%) = 95% x 142 = 134, 9 detik

τ_R = 134, 9 – 7,1 = = 127, 8 detik

5 Kesimpulan

Berdasarkan hasil pengujian prototipe *heat exchanger* maka sistem pengendalian yang semula menggunakan mode tuning temperatur PI dan laju alir PID maka dirubah menjadi mode *trial and error* karena proses menjadi lebih baik dengan mode temperatur PID dan mode laju alir menggunakan PI. Metode *trial and error* PID yang sesuai untuk pengendalian temperatur berupa *Proportional* = 17,18, *Integral* = 7, *Derivative* = 10 dan laju alir mempunyai *Proportional* = 2, *Integral* = 1, *Derivative* = 0. Menghilangkan mode *derivative* pada laju alir dan menambahkan mode *derivative* pada temperature membuat proses lebih cepat $\pm$ 8 menit. Pada prinsipnya, kedua variabel proses berjalan secara linear, apabila *servo valve* berada pada kondisi bukaan penuh maka temperatur keluaran *heat exchanger* yang dihasilkan akan semakin panas, begitu pula sebaliknya pada saat kondisi *servo valve* mendekati kondisi tutup *valve*.

6 Ucapan Terima Kasih

Terima kasih diucapkan kepada Politeknik Energi dan Mineral Akamigas yang telah memberikan fasilitas dan biaya dalam penelitian ini.

7 Referensi

[1] Y. A. Cengel, *Heat and Mass Transfer*. 2015.
[2] S. Sujono, A. K. Dewi, and T. S. Soegiarto, “Evaluating and Optimizing Performance of Shell and Tube Heat Exchanger Using Excel-solver”, *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 830, no. 4, 2020, doi: 10.1088/1757-899X/830/4/042029.
[3] K. Thulukkanam, *Heat Exchanger Design Handbook, Second Edition*, 2nd ed. New York: CRC Press, 2013.
[4] Y. Efendi, N. A. Mardiyah, and Z. Has, “Desain dan Verifikasi Kontrol Cascade Pengendali Suhu Berbasis Fuzzy-PID dan PI pada Heat Exchanger”, pp. 107–113, 2019.
[5] R. Sandy Montolalu, F. Yosef Suratman, and P. Pangaribuan, “Rancang Bangun Sistem Kontrol Level dan Temperatur Boiler Dengan Metode PID dan Kontrol Dua Posisi (Design and Implementation for Controlling Boiler Water Level and Temperature Using Pid Method and On- Off Control)", *e-Proceeding Eng.*, vol. 2, no. 2, pp. 2262–2269, 2015.
[6] C. Yusuf and D. I. Saputra, “Optimasi Kendali Suhu pada Sistem Nirkabel Penetasan Telur Berbasis PI dan PI Anti Windup", *J. Otomasi Kontrol dan Instrumentasi*, vol. 12, no. 2, pp. 79–101, 2020, doi: 10.5614/joki.2020.12.2.3.
[7] Y. V. P. K. Y V Pavan Kumar, “Cascaded PID Controller Design for Heating Furnace Temperature Control", *IOSR J. Electron. Commun. Eng.*, vol. 5, no. 3, pp. 76–83, 2013, doi: 10.9790/2834-0537683.
[8] A. S. Wardhana, M. Ashari, and H. Suryoatmojo, “Optimal Control of Robotic Arm System to Improve Flux Distribution on Dual Parabola Dish Concentrator", *Int. J. Intell. Eng. Syst.*, vol. 13, no. 1, pp. 364–378, 2020, doi: 10.22266/ijies2020.0229.34.
[9] A. Jamal and R. Syahputra, “Heat Exchanger Control Based on Artificial Intelligence Approach", *Int. J. Appl. Eng. Res.*, vol. 11, no. 16, pp. 9063–9069, 2016.
[10] A. Vasičkaninová and M. Bakošová, “Control of a Heat Exchanger Using Neural Network Predictive Controller Combined with Auxiliary Fuzzy Controller”, *Appl. Therm. Eng.*, vol. 89, no. January 2015, pp. 1046–1053, 2015, doi: 10.1016/j.applthermaleng.2015.02.063.
[11] S. Padhee, “Controller Design for Temperature Control of Heat Exchanger System : Simulation Studies", vol. 9, pp. 485–491, 2014.

[12] J. U. Ravy, N. A. Septiani, A. K. Dewi, and A. S. Wardhana, “Evaluasi Kinerja Controller Design PI Sistem Pengendalian Level pada Centrifugal Preparation Tank", in *Prosiding Seminar Nasional Teknologi Energi dan Mineral*, 2021, vol. 1, pp. 1–11.
[13] D. Hendrawati and Suwarti, “Perbaikan Karakteristik Kontroller Temperatur pada Model Boiler", in *Prosiding SNST Fakultas Teknik*, 2010, pp. 13–17.
[14] I. P. A. W. Widyatmika, N. P. A. W. Indrawati, I. W. W. A. Prastya, I. K. Darminta, I. G. N. Sangka, and A. A. N. G. Sapteka, “Perbandingan Kinerja Arduino Uno dan ESP32 Terhadap Pengukuran Arus dan Tegangan", *J. Otomasi Kontrol dan Instrumentasi*, vol. 13, no. 1, pp. 35–47, 2021, doi: 10.5614/joki.2021.13.1.4.
[15] P. F. Dunn, *Fundamentals of Sensors for Engineering and Science*. 2012.
[16] A. S. Wardhana, M. Ashari, and H. Suryoatmojo, “Designing and Modeling a Novel Dual Parabolic Concentrator With Three Degrees of Freedom (DOF) Robotic Arm", *Sol. Energy*, vol. 194, 2019, doi: 10.1016/j.solener.2019.10.057.
[17] Kharagpur, “Structures : Cascade , Override and Split Range,” pp. 1–9, 2008.