Simulator Kontrol Manual Dan Swa-tala Untuk Dehydration Glycol Unit Berbasis JavaScript

Muhammad Putra Rasuanta[1], Eko Mursito Budi [2,3], Fitria Hidayanti[1,4] & Fitri Rahmah[1,5]

[1]Teknik Fisika UNAS

[2]Teknik Fisika ITB

[3]*mursito@tf.itb.ac.id*

[4]*fitriahidayanti@gmail.com*

[5]*fitrirahmahtf@gmail.com*

Abstrak

Salah satu cara memurnikan gas alam adalah melalui proses distilasi menggunakan *Dehydration glycol unit*. Pemurnian cara klasik ini masih digunakan hingga sekarang sehingga hampir di setiap kilang gas menggunakan *Dehydration glycol unit* dalam prosesnya. Oleh karena itu memahami cara kerja dan cara mengendalikan *Dehydration glycol unit* penting bagi seorang mahasiswa yang akan atau sedang bekerja dalam bidang *oil and gas*. Makalah ini membahas sebuah simulator untuk memahami cara kerja dan mengendalikan plant *Dehydration glycol unit* dengan pengontrol PID yang ditala berdasar metode manual *Ziegler-Nichols* dan swa-tala *relay*. Simulator dibuat berdasar algoritme Euler dan diimplementasikan dengan JavaScript agar bisa dijalankan sebagai aplikasi berbasis web. Simulasi dengan JavaScript relatif mudah dilakukan dikarenakan penulisan kode yang ringkas dan terdapat pustaka-pustaka pendukung. Sementara itu dari hasil simulasi, didapati bahwa respons penalaan manual menghasilkan *overshoot* dan memiliki waktu lebih cepat dalam mencapai *setpoint*. Sebaliknya penalaan swa-tala tidak menghasilkan *overshoot* walau lebih lama untuk mencapai nilai *setpoint*.
Keywords: *Dehydration glycol unit; PID; Ziegler-Nichols; Relay; JavaScript;* simulator

1 Pendahuluan

Uap air adalah zat yang tercampur pada gas alam ketika baru keluar dari sumur sumber gas tersebut. Hal itu umum ditemukan dan menjadikan gas harus dimurnikan terlebih dahulu sebelum didistribusikan nantinya. Cara memurnikan gas alam ialah dengan proses distilasi menggunakan *Dehydration glycol unit*. Pemurnian cara klasik ini masih digunakan hingga saat ini sehingga hampir di setiap kilang gas menggunakan *Dehydration glycol unit* dalam prosesnya. Oleh karena itu memahami cara kerja dan cara mengendalikan *Dehydration glycol unit* begitu penting.

PID adalah metode kontrol paling populer digunakan pada industri. Survey menunjukkan 90% loop kontrol menggunakan struktur kontrol PI atau PID [8]. Agar parameter kontrol PID bekerja dengan baik, dibutuhkan sebuah penalaan.

Metode penalaan Ziegler-Nichols (ZN) ditemukan oleh John Ziegler dan Nathaniel Nichols pada tahun 1942 [1]. Ada dua cara dalam penalaan PID, pertama dari respons *step* sistem tersebut dan kedua dari respons osilasi sistem. Untuk melakukan penalaan degan cara pertama, hal yang pertama kali dilakukan adalah dengan memberikan input *step* dan sistem akan merespons input tersebut hingga mencapai keadaan tunak [2]. Saat sistem sudah mencapai keadaan tunak, dapat ditentukan nilai t_{10} dan t_{90} dimana keduanya adalah waktu ketika sistem mencapai nilai 10% dan 20% sebelum keadaan tunak. Dari waktu tersebut akan digunakan untuk mencari gain, *time constant* (τ), dan *dead time* (θ). Tiga nilai tersebut merupakan nilai penting untuk mendapatkan parameter PID pada metode ini.

Penalaan pengamatan respons osilasi sistem menggunakan aksi kontrol Proporsional (P) [1]. Untuk membuat sistem mengalami osilasi sempurna yaitu dengan mencari nilai K_P[1]. Nilai K_P yang dapat membuat sistem mengalami osilasi dinamakan gain *ultimate* K_u. Hasil respons osilasi akan diamati jarak antara dua puncak terdekat (P_{cr}) dengan begitu parameter PID dapat ditentukan.

Swatala degan metode *relay* dapat dikatakan hampir sama dengan pengamatan respons osilasi pada metode ZN [3]. Perbedaannya adalah, untuk membuat respons osilasi pada sistem menggunakan histeresis. Metode ini ditemukan oleh Åström dan Hägglund untuk memperbaiki metode ZN respons osilasi [4]. Gain *ultimate* akan tetap dicari dengan menggunakan nilai histeresis dan amplitude respons [5].

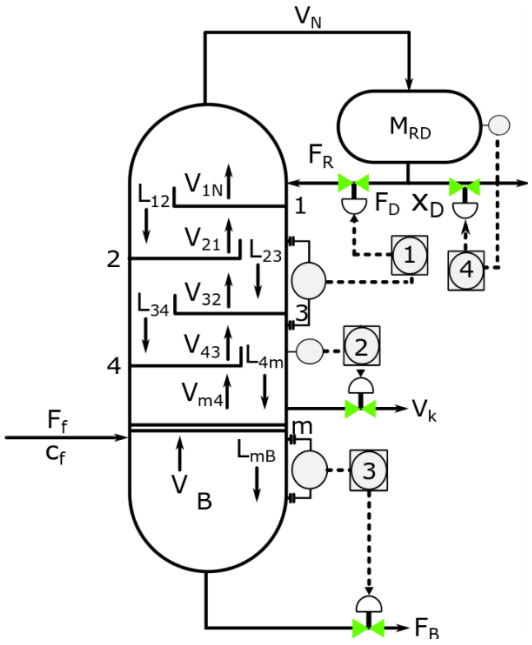
Pada paper ini akan dibahas mengenai komparasi antara hasil respons penalaan manual dengan swatala pada plant *Dehydration Glycol Unit*. Dari perbandingan tersebut akan dipaparkan hasil respons sehingga menjadi

wawasan dalam melakukan penalaan pada *plant* tersebut. Kontrol hanya akan digunakan adalah PID dan besaran fisis level, sedangkan perhitungan akan menggunakan JavaScript sebagai bahasa pemrograman.

2 *Dehydration Glycol Unit*

Cara kerja dari *Dehydration Glycol Unit* adalah dengan mengalirkan gas alam (mengandung air dan zat campuran seperti hidrokarbon dll.) melalui bawah *contractor tower*. Kemudian gas akan menuju bagian atas dari *contractor tower* melewati *mist extractor* sehingga sebagian dari air yang tercampur berkurang, setelah itu gas alam akan melewati *bubble cap trays.* Pada saat yang bersamaan, glikol dialirkan menuju bagian atas *contractor tower*. Glikol akan menuju bagian bawah *contractor tower* melalui *bubble cap trays*. Gas dan glikol akan bertemu, kemudian kandungan air yang tercampur pada gas akan terserap oleh glikol sehingga kandungan air akan berkurang, fenomena tersebut terjadi di setiap *bubble cap trays*.

Glycol yang sudah berikatan dengan air akan terkumpul pada dasar *contractor tower* dan dialirkan melewati *reflux coil* menuju *heat exchanger* untuk diproses di *ashgas separator*. Proses yang terjadi di *ashgas separator* adalah pemisahan glikol dari air beserta zat lain yang terikat dengan glikol. Zat yang sudah terpisah dari glikol akan dialirkan menuju proses selanjutnya dan glikol yang sudah bersih akan dialirkan kembali menuju bagian atas *contractor tower*. Selanjutnya gas yang sudah terpisah dari air dialirkan menuju *reflux drum*.



B = Bagian terbawah kontraktor (*Base*)

C_f = Komposisi gas masuk kedalam kontraktor [fraksi mol]

F_B = Aliran glikol keluar dari *base* [kmol/min]

F_D = Aliran glikol keluar dari *reflux drum* [kmol/min]

F_f = Aliran gas masuk kedalam kontraktor [kmol/min]

F_R = Aliran glikol berasal dari *reflux drum* [kmol/min]

$L_{i,i+/-1}$ = Aliran glikol dari *tray* i menuju *tray* ke i [kmol/min]

L_n = Aliran glikol pada *tray* ke n [kmol/min]

L_{0n} = Aliran glikol mula-mula pada *tray* ke n [kmol/min]

m = *Mist extractor*

M_{RD} = Massa pada *reflux drum* [kmol]

M_n = Massa pada *tray* ke n [kmol]

M_{0n} = Massa mula-mula pada *tray* ke n [kmol]

V = Aliran uap masuk kedalam kontraktor menuju *mist extractor* [kmol/min]

$V_{i,i+/-1}$ = Aliran uap dari *tray* i menuju *tray* ke i [kmol/min]

V_K = Aliran glikol keluar dari *mist extractor* [kmol/min]

V_N = Aliran uap menuju *reflux drum* [kmol/min]

X_D = Komposisi glikol keluar dari *reflux drum* [fraksi mol]

Gambar 1 P&ID Dehydration Glycol Unit

Penentuan model *Dehydration Glycol Unit* direpresentasikan dari gambar diatas dimana model, di asumsikan bahwa uap pada setiap *tray* bernilai sama, tekanan dalam kontraktor konstan, konstanta *volatilize* dua komponen konstan uap pada setiap *tray* sehingga persamaan matematis dituliskan sebagai berikut [6] :

a. Bagian atas kontraktor (*tray 1*)

$$\frac{dM}{dt} = F_R + V_{21} - L_{12} - V_{1N} \quad (1)$$

b. Bagian setiap *tray*

$$\frac{dM_i}{dt} = L_{i-1,i} + V_{i+1,i} - L_{i,i+1} - V_{i,i-1} \quad (2)$$

c. Bagian *mist extractor*

$$\frac{dM_m}{dt} = L_{4m} + V - L_{mB} - V_{m4} - V_K \quad (3)$$

d. Bagian bawah kontraktor (*base*)

$$\frac{dM_B}{dt} = F_f + L_{mB} - F_B \quad (4)$$

e. Bagian *reflux drum*

$$\frac{dM_{RD}}{dt} = V_N + F_R - F_D \quad (5)$$

Selanjutnya, untuk mendeskripsikan jatuhnya glycol yang terbendung pada *tray* dituliskan menggunakan persamaan Francis Weir sebagai berikut [7][8] :

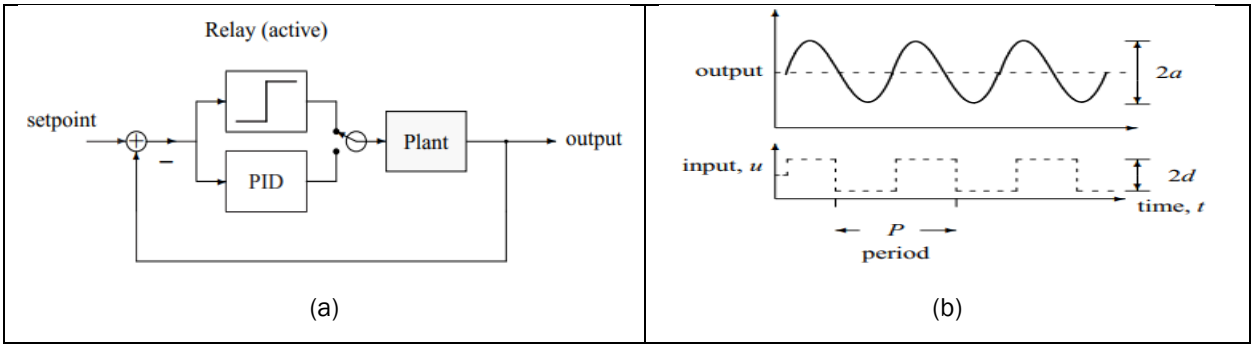
$$L_n = \frac{L_{0n} + (M_n - M_{0n})}{\tau} \quad (6)$$

3 Rancangan Arsitektur Kontrol Loop

Kontrol PID umum digunakan karena strukturnya, efektifitas yang bagus dan mudahnya dalam implementasi. Dinamakan PID karena keluaran dari kontrol ini adalah penjumlahan dari ketiga komponen kontrol tersebut. Alasan mengapa kontrol PID umum digunakan karena persamaan kontrol ini tidaklah rumit seperti matematika kompleks. Berikut adalah rumus dari kontrol PID :

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t) + T_d\frac{de(t)}{dt}\right) \quad (7)$$

Dimana *u*(*t*) adalah keluaran sinyal kontrol, *e*(*t*) adalah selisih hasil keluaran dengan SP (galat), Sedangkan K_p, K_i, dan K_d adalah gain proporsional, gain integral dan gain derivatif [9]. Ada 3 loop kontrol pada *Dehydration glycol unit* yang telah tertera pada P&ID Gambar 1. Masing-masing loop tersebut dipasangi pengontrol PID. Pada kesempatan pertama, PID tersebut ditala secara manual. Selanjutnya PID diberi swa-tala metode relay, dengan diagram blok seperti Gambar 2.a.



Gambar 2 Arsitektur Kontrol dan keluaran Swa-tala [10]

Prinsip kerja dari swa-tala ini berdasarkan pada informasi periode dan magnitude osilasi dari respons keluaran kontrol [10]. Prinsip ini mirip dengan kontrol on-off atau kontrol histeresis yang hanya memiliki dua keadaan saja, yaitu maksimum dan minimum. Pada Gambar 2.b, Nampak bahwa ketika variabel proses melebihi nilai *set point* dan menyentuh nilai maksimum histeresis maka keluaran relay akan bernilai negatif, sedangkan apabila variabel proses dibawah nilai *set point* dan menyentuh nilai maksimum histeresis maka nilai relay akan menjadi positif. Hal tersebut menyebabkan respons keluaran dari sistem mengalami osilasi dengan sempurna.

Informasi dari nilai magnitude tersebut akan dijadikan dasar untuk mencari nilai K_u danP_{cr}. Nilai K_u dapat dicari dengan persamaan sebagai berikut:

$$K_u = \frac{4d}{\pi A} \tag{8}$$

Keterangan :

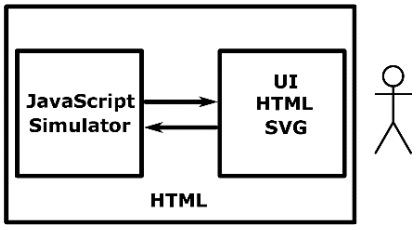
A = Amplitudo

d = Keluaran *relay* puncak ke puncak (histeresis)

K_u = Gain *ultimate*

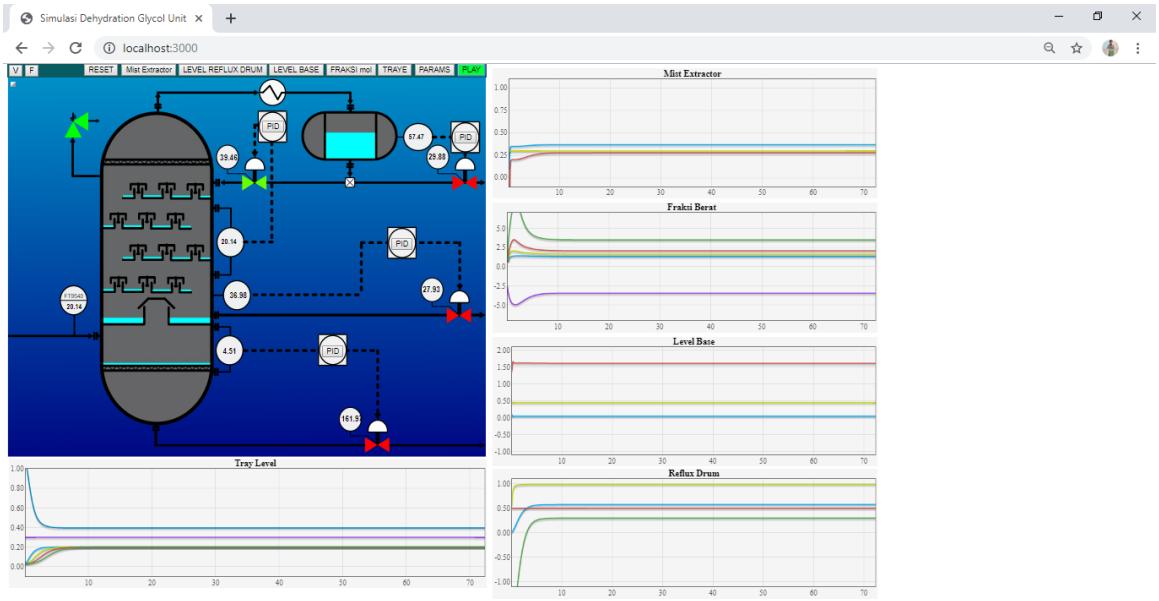
4 Simulator Berbasis JavaScript

Sistem ini dirancang sebagai aplikasi berbasis web yang dijalankan menggunakan browser. Seperti diberikan pada Gambar 3, sistem terdiri atas user interface (UI) berbasis hypertext markup language (HTML) dan scalable vector graphics (SVG), serta mesin simulator JavaScript (JS). HTML berperan sebagai wadah JavaScript dan SVG. Kemudian JavaScript berperan melakukan perhitungan matematis yang nantinya akan digunakan untuk memanipulasi objek SVG. Sedangkan SVG digunakan untuk antarmuka simulator.



Gambar 3 Arsitektur Simulator dan User Interface

Antarmuka pengguna berfungsi untuk menampilkan berbagai informasi proses secara dinamis, serta menerima masukan dari pengguna sistem. Untuk itu, seperti Nampak pada Gambar 4, ditampilkan grafik P&ID plant berikut grafik berbagai variabel proses. Pada antarmuka juga terdapat tombol dan menu untuk mengubah parameter proses (misalnya menala PID). Selama simulasi, berbagai variabel proses akan direkam dan dapat diunduh dalam format CSV (comma separated values).



Gambar 4 User Interface

Sementara itu, simulator dibuat menggunakan metode Euler agar mudah dan cepat dalam mengalkulasi. Dibawah ini adalah *pseudocode* simulator :

```
Set(waktu_cuplik, parameter, state, waktu)
Tampilkan(state, waktu)
While running do
  Read(input)
  delta_state = perubahan_state(parameter, state, input, waktu)
  state = state + (delta_state * waktu_cuplik)
  waktu = waktu + waktu_cuplik
  Tampilkan(state, waktu)
End
```

Selanjutnya pemrograman antarmuka maupun simulator tersebut menggunakan JavaScript dengan pustaka-pustaka bantu seperti dijabarkan pada Tabel 1.

Tabel 1 Pustaka-pustaka penunjang.

No	Pustaka	Fungsi	File yang digunakan
1	Node.js	Sebagai server untuk memuat simulator.	node.js
2	Flotr2	Membuat dan menampilkan animasi grafik.	flotr2.js
3	FileSaver	Membuat dan menampilkan animasi grafik.	FileSaver.js
4	Snap.svg	Menampilkan gambar SVG sebagai UI.	Snap.svg.js
5	Sprintf	Memanipulasi teks angka sebagai animasi.	

5 Uji Coba Simulator

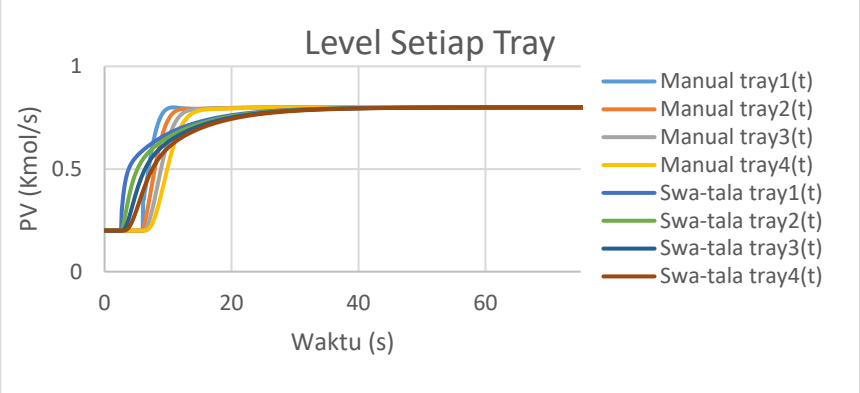
Saat uji coba, akan dibandingkan kinerja sistem berdasar penalaan manual dan swa-tala. Pengguna akan diberi dua pilihan metode penalaan ketika hendak menggunakan simulator. Pada penalaan manual, sistem akan diberikan respons *step* dalam kondisi loop terbuka. Ketika sudah mencapai keadaan tunak maka akan dicari nilai gain, *time constant* (τ), dan *dead time* (θ) dengan menggunakan waktu ketika sistem mencapai 10% dan 90% dari nilai keadaan tunak. Dari hasil tersebut, akan dihitung parameter PID memakai metode Ziegler Nichols. Setelah itu dilakukan uji respons step kembali, dengan memasukkan PID yang telah ditala dengan parameter hasil perhitungan tersebut. Sementara itu pada metode swa-tala, pengguna hanya perlu memasukan nilai histeresis dan sistem akan menghitung nilai PID mandiri. Berikut ini hasil perbandingan antara penalaan manual dan swa-tala.

a. Bagian setiap *tray* (Loop 1)

Nilai parameter PID yang didapat pada masing-masing penalaan adalah sebagai berikut:

Tabel 2 Parameter PID pada *tray*.

No	Penalaan	P	I	D	h
1	Manual	1,08	2,00	0,5	
2	Swa-tala	2,15	0,55	0,13	5



Gambar 5 Hasil Perbandingan Penalaan Level Setiap *Tray*

Tabel 3 Analisa Respons Setiap *Tray*

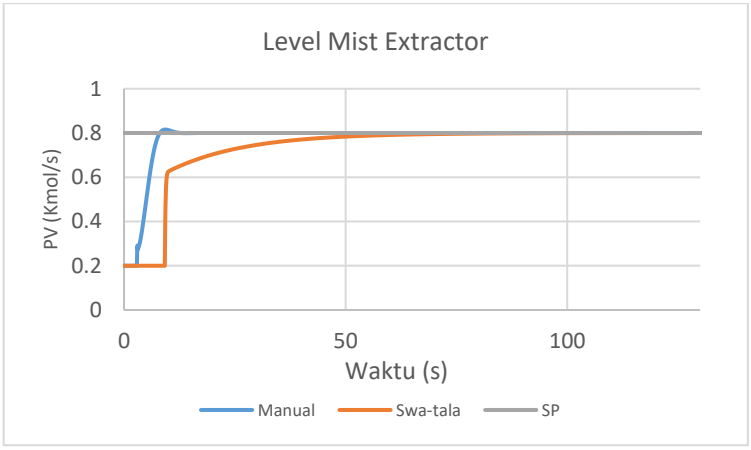
No	Analisa	Manual				Swa-tala			
		Tray 1	Tray 2	Tray 3	Tray 4	Tray 1	Tray 2	Tray 3	Tray 4
1	*Delay Time* (s)	6,7	7,5	8,4	9,4	3,7	5,1	6,3	7,6
2	*Rise Time*(s)	6,5	8,8	9,9	11	15,3	15,8	18	19,2
3	*Peak Time*(s)	-	-	-	-	-	-	-	-
4	*Maximum Overshoot*	-	-	-	-	-	-	-	-
5	*Time Constant* (s)	6,9	7,9	8,9	9,8	5,4	6,8	8	5,6
6	*Settling Time* (s)	15,7	15,9	16,4	19,3	30,2	38,1	40,2	48

Respons *rise time tray* tercepat ada pada urutan pertama sedangkan terlama di urutan terakhir dan ini terjadi pada kedua penalaan. Hal tersebut dapat terjadi dikarenakan *tray* memiliki ketergantungan pada *tray* lainnya berdasarkan urutan. *Tray* pada urutan kedua baru akan terisi jika *tray* pada urutan pertama sudah membendung penuh, begitu pula seterusnya. Kemudian, pada bagian *tray* penalaan manual memiliki respons waktu tunda dan *time constant* lebih lama masing-masing 3 s dan 1.5 s dari swa-tala dengan referensi *tray 1* hal tersebut dikarenakan nilai kontrol Proporsional pada swa-tala lebih besar dari penalaan manual, namun memiliki nilai waktu lebih cepat pada *rise time dan settling time* masing-masing 8.8 s dan 22.4 s dengan referensi *tray 1.*

b. Bagian *mist extractor* (Loop 2)

Tabel 4 Parameter PID pada *mist extractor*.

No	Penalaan	P	I	D	h
1	Manual	0,84	2,00	0,50	
2	Swa-tala	2,1211	0,15	0,0375	5



Gambar 6 Hasil Perbandingan Penalaan Level *Mist extractor*

Tabel 5 Analisa Respons *Mist extractor*

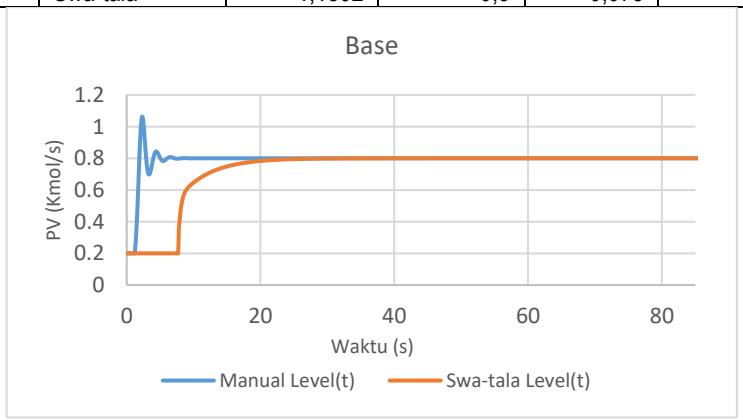
No	Analisa	Manual	Swa-tala
1	*Delay Time* (s)	5,1	9,4
2	*Rise Time* (s)	7	27,6
3	*Peak Time* (s)	9,3	-
4	*Maximum Overshoot*	0,81	-
5	*Time Constant* (s)	5,5	9,6
6	*Settling Time* (s)	12,9	83

Pada penalaan manual, sistem mengalami *overshoot* sebesar 0,81 pada s ke 9,3 akan tetapi hal tersebut tidak terjadi pada swa-tala. Tidak terjadinya *overshoot* pada swa-tala menyebabkan lebih besarnya nilai *settling time* yaitu selama 83 s bila dibandingkan dengan penalaan manual. Untuk *delay time, time rise,* dan *time constant* penalaan manual masing-masing selama 5,1 s, 7 s dan 5,5 s. Waktu tersebut lebih lama dari swa-tala.

c. Bagian bawah kontraktor (*base*) (Loop 3)

Tabel 6 Parameter PID pada *base*.

No	Penalaan	P	I	D	h
1	Manual	0,36	3,33	-	
2	Swa-tala	1,1802	0,3	0,075	10



Gambar 7 Hasil Perbandingan Penalaan Level *Base*

Tabel 7 Analisa Respons *Base*

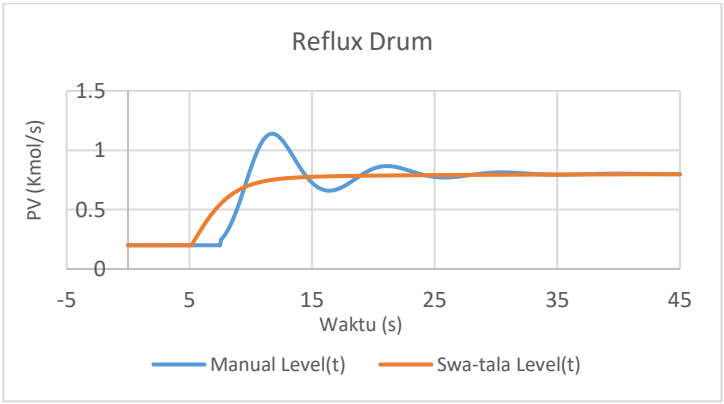
No	Analisa	Manual	Swa-tala
1	*Delay Time* (s)	1,6	8,1
2	*Rise Time* (s)	1,8	14
3	*Peak Time* (s)	2,2	-
4	*Maximum Overshoot*	1,06	-
5	*Time Constant* (s)	1,7	8,5
6	*Settling Time* (s)	6,2	26,6

Penalaan manual pada bagian *base* digunakan kontrol PI, hal tersebut dikarenakan respons kontrol PI dapat mencapai nilai *setpoint*. Pada penalaan manual sistem mengalami *overshoot* sedangkan pada swa-tala tidak. Namun, *settling time* pada swa-tala lebih lama dengan waktu 26,6 s jika dibandingkan dengan penalaan manual.

d. Bagian *reflux drum* (Loop 4)

Tabel 8 Parameter PID pada *base*.

No	Penalaan	P	I	D	h
1	Manual	1,2	2	0,5	
2	Swa-tala	2,1634	0,15	0,0375	5



Gambar 8 Hasil Perbandingan Penalaan Reflux Drum

Tabel 9 Analisa Respons *Reflux Drum*

No	Analisa	Manual	Swa-tala
1	*Delay Time* (s)	8,9	7,1
2	*Rise Time* (s)	9,7	11
3	*Peak Time* (s)	11,8	-
4	*Maximum Overshoot*	1,13	-
5	*Time Constant* (s)	9,2	7,8
6	*Settling Time* (s)	38,8	30,2

Untuk penalaan manual pada *reflux drum* respons mengalami overshoot sebesar 1,29 pada s ke 7,1. Nilai *delay time, rise time,* dan *time constant* swa-tala masing-masing selama 8,9 s, 9,7 s, dan 9,2 s lebih lama dari pada penalaan manual dikarenakan nilai kontrol Proporsional swa-tala lebih besar dan pada swa-tala tidak memiliki *overshoot.*

6 Kesimpulan

1. Bagian *tray* dengan penalaan manual, *tray* pada urutan pertama memiliki nilai *settling time* tercepat yaitu selama 15,7 s. sedangkan yang terlama adalah *tray* pada urutan ke empat dengan waktu selama 19,3 s. Untuk swa-tala *settling time* tercepat ada pada *tray* urutan pertama dengan lama 30,2 s dan terlama pada urutan ke empat selama 48 s. Penalaan manual dan swa-tala pada *tray* tidak mengalami *overshoot*.

   Pada *mist extractor* respons penalaan manual mengalami *overshoot* sebesar 0,81 pada s ke 9,3 sedangkan respons swa-tala tidak menghasilkan *overshoot. Settling time* penalaan manual dan swa-tala masing-masing selama 12,9 s dan 83 s. Respons swa-tala membutuhkan waktu lebih lama untuk mencapai keadaan tunak.

   Untuk bagian *base* kontrol yang digunakan pada penalaan manual adalah PI, dikarenakan pada kontrol lain sistem tidak mencapai nilai *setpoint.* Respons penalaan manual mengalami *overshoot* sebesar 1,06 pada s ke 2,2 dengan *settling time* selama 6,2 s. pada swa-tala respons tidak mengalami *overshoot* namun waktu untuk mencapai keadaan tunak lebih lama yaitu 26,6 s.

   Terakhir pada bagian *reflux drum*, respons penalaan manual mengalami *overshoot* sebesar 1,29 pada s ke 7,1 sedangkan pada swa-tala tidak. Untuk *settling time*, respons penalaan manual mem butuhkan waktu lebih lama untuk mencapai keadaan tunak. *Settling time* penalaan manual dan swa-tala masing-masing adalah 4,31 s dan 30,2 s.

2. Pemrograman simulator menggunakan JavaScript cukup mudah dilakukan karena telah tersedia berbagai pustaka penunjang. Perhitungan dinamika sistem simulator menggunakan algoritme Euler dapat dituliskan dalam 47 *Line of Code*. Untuk penalaan manual dan swa-tala dituliskan masing-masing 12 dan 41 *Line of Code.*

7 Referensi

[1] Ogata, K. 1970. *Modern Control Engineering*. 5th edition. Prentice Hall.

[2] Liptak B.G, Instrument Engineers Handb ook Process Control and Optimization, Vol 2., CRC Press. 1995.

[3] Wu, A., Weihua, S. & Zhiguo, L., *PID Controllers: design and tuning method*., IEEE, pp 808 – 813 , 2014.

[4] Prokop, R., Ji, K., S. & Radek, M., *Relay Feedback Identification of Dynamical SISO System Analysis and Setting*s., Latest Trends on System, pp 450 – 454.

[5] Ho, W, K., Y, Hong., A, Hanson., H, Hjalmarsson & J, W, Deng., *Relay Auto-tuning of PID Controllers Using Iterative Feedback Tuning*., Automatica, pp 149 – 157 , 2003.

[6] S, Skogestad., *Dynamics and Control of Distillation Columns: A Critical Survey*., Modelling, Identification and Control, pp 177 – 217, 1997.

[7] S, Skogestad., *Dynamics and Control of Distillation Columns: A Tutorial Introduction*., Institution of Chemical Engineering, pp 539 – 562, 1997.

[8] Ekawati, E., Megarini, H. & Tua, A, T,. *Process and Controller Configuration of High Purity Binary Distillation Columns in the Purification Stage of Petrochemical Plant*. IEEE, pp 82 – 88, 2017.

[9] Chopral, V., Sunil, K., Sigla, L, D., *Comparative Analysis of Tuning a PID Controller using Intelligent Methods*. Acta Polytechnica Hungarica, pp 235 – 249, 2014.

[10] Utomo, S, D., Djoko, P & Tria, A, S., *Implementasi Metode Auto Tuning PID Pada Balancing Robot*. Teknik Elektro ITS, pp 1 – 9.