

BAB IV

PENGUJIAN DAN ANALISIS

4.1 Pengujian Komunikasi *Pin to pin* Antar ESP32

Pengujian komunikasi *pin to pin* dilakukan untuk mengetahui apakah data dari satu ESP32 dapat diteruskan dan dibaca dengan baik oleh ESP32 lainnya. Pada sistem ini, data dikirim dalam bentuk sinyal PWM, kemudian dilewatkan melalui rangkaian *RC filter* agar dapat dibaca kembali sebagai tegangan analog oleh ADC pada sisi penerima.

Pengujian dilakukan dengan memberikan beberapa nilai PWM tetap, kemudian diamati hasil tegangan setelah *RC filter*, nilai ADC yang diterima, serta hasil *scaling* pada *OpenPLC*. Selain melihat kesesuaian nilai yang dikirim dan diterima, pengujian ini juga digunakan untuk melihat kestabilan data, terutama karena pada pengujian temperatur sebelumnya masih terlihat perubahan nilai pada *OpenPLC* walaupun kondisi temperatur air relatif stabil.

Dari pengujian ini dapat diketahui apakah perubahan nilai yang dikirim dari ESP32 pengirim dapat diteruskan dengan baik sampai ke ESP32 penerima, sekaligus melihat seberapa besar pengaruh jalur komunikasi, *RC filter*, dan pembacaan ADC terhadap kestabilan nilai yang diterima.

4.1.1 Pengujian Pengiriman Nilai 0 pada *OpenPLC*

Pada pengujian pertama, *OpenPLC* diberikan nilai PWM sebesar 0. Kondisi ini digunakan sebagai titik awal untuk melihat keadaan komunikasi ketika tidak ada nilai kendali yang dikirim. Secara kondisi kerja, keluaran seharusnya berada mendekati nol sehingga tegangan hasil *RC filter* juga berada pada nilai yang sangat rendah.

Pengukuran dilakukan pada keluaran *RC filter* menggunakan multimeter dalam mode tegangan DC. Hasil pengukuran tersebut digunakan untuk melihat tegangan yang nantinya masuk ke ADC ESP32 Arduino IDE.

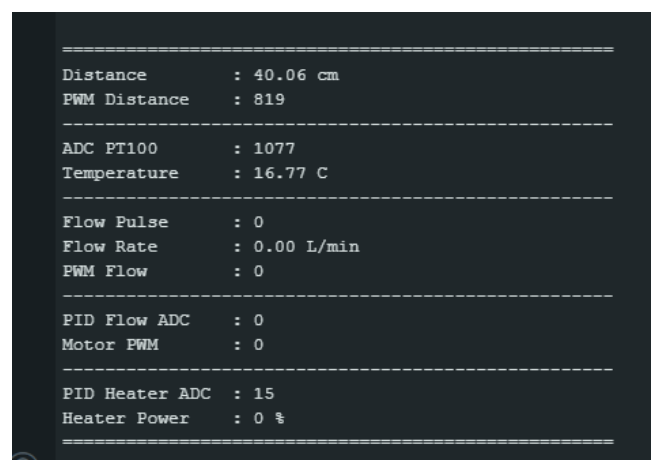


Gambar 4.1 Pembacaan Arus PWM 0 *OpenPLC*

Berdasarkan Gambar 4.1, ketika nilai PWM yang diberikan sebesar 0, tegangan yang masih terbaca pada keluaran *RC filter* adalah sekitar 98,0 mV atau 0,098 V. Nilai tersebut sudah sangat kecil dan mendekati kondisi nol, tetapi belum benar-benar 0 V.

Munculnya tegangan kecil pada kondisi PWM 0 masih memungkinkan terjadi karena keluaran rangkaian analog tidak selalu benar-benar berada tepat pada 0 V. Masih dapat terdapat tegangan sisa pada rangkaian, noise, maupun toleransi pembacaan alat ukur. Yang lebih penting dari hasil ini adalah tegangan yang diperoleh masih berada sangat dekat dengan kondisi nol.

Setelah tegangan tersebut masuk ke ADC Arduino, hasil pembacaannya diamati melalui Serial Monitor. Hal ini dilakukan untuk melihat apakah tegangan kecil yang masih terdapat pada keluaran *RC filter* ikut terbaca oleh ADC dan bagaimana hasil akhirnya setelah diproses oleh program.



Gambar 4.2 Pembacaan Arduino IDE

Berdasarkan Gambar 4.2, Arduino membaca PID *Heater* ADC sebesar 15, sedangkan *Heater Power* yang dihasilkan tetap sebesar 0%. Hasil tersebut sesuai dengan kondisi awal karena nilai ADC yang terbaca masih sangat rendah.

Nilai ADC 15 juga menjelaskan mengapa tegangan pada hasil pengukuran tidak benar-benar nol. Masih terdapat sedikit nilai yang masuk ke pin ADC, tetapi setelah diolah oleh program nilainya tetap masuk ke kondisi *Heater Power* 0%. Jadi pada titik ini komunikasi dapat membedakan kondisi tidak ada perintah daya dengan kondisi ketika nilai PWM mulai dinaikkan.

4.1.2 Pengujian Pengiriman Nilai 20000 pada *OpenPLC*

Pada pengujian kedua, nilai PWM dari *OpenPLC* dinaikkan menjadi 20000. Pengujian ini dilakukan untuk melihat perubahan tegangan hasil *RC filter* ketika nilai kendali mulai diberikan dan melihat apakah perubahan tersebut juga terbaca pada ADC Arduino.

Nilai 20000 yang dikirim dari *OpenPLC* keluar dalam bentuk sinyal PWM. Setelah melewati *RC filter*, sinyal tersebut berubah menjadi tegangan analog yang kemudian diukur secara langsung menggunakan multimeter.



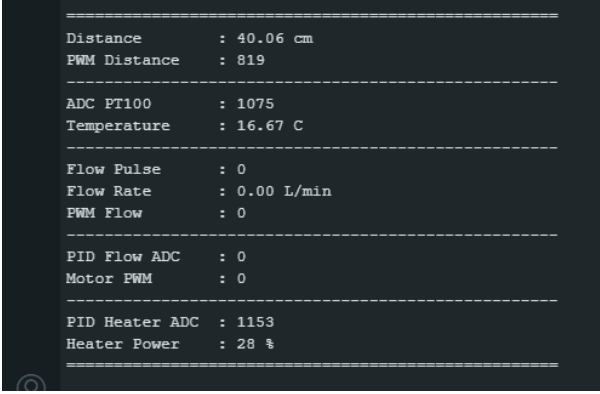
Gambar 4.3 Pembacaan Arus PWM 20000 *OpenPLC*

Berdasarkan Gambar 4.3, pada saat *OpenPLC* mengirim nilai PWM 20000, tegangan setelah *RC filter* terbaca sebesar 1,030 V. Jika dibandingkan dengan kondisi PWM 0 yang hanya menghasilkan sekitar 0,098 V, terlihat bahwa kenaikan nilai PWM dari *OpenPLC* langsung diikuti oleh kenaikan tegangan pada keluaran *RC filter*.

Hasil ini menunjukkan bahwa *RC filter* sudah bekerja mengubah perubahan nilai PWM menjadi perubahan tegangan analog. Nilai PWM yang lebih besar

menghasilkan tegangan rata-rata yang lebih besar, dan tegangan inilah yang kemudian diterima oleh ADC Arduino.

Setelah tegangan hasil *RC filter* masuk ke Arduino, nilai ADC dan hasil pengolahannya dilihat melalui *Serial Monitor*.



Distance	: 40.06 cm
PWM Distance	: 819
ADC PT100	: 1075
Temperature	: 16.67 C
Flow Pulse	: 0
Flow Rate	: 0.00 L/min
PWM Flow	: 0
PID Flow ADC	: 0
Motor PWM	: 0
PID Heater ADC	: 1153
Heater Power	: 28 %

Gambar 4.4 Pembacaan Arduino IDE

Berdasarkan Gambar 4.4, tegangan sekitar 1,030 V menghasilkan pembacaan *PID Heater ADC* sebesar 1153. Setelah nilai ADC tersebut diolah oleh program Arduino, diperoleh *heater power* sebesar 28%.

Dari hasil ini terlihat hubungan yang jelas antara ketiga bagian pengujian. *OpenPLC* mengirim nilai 20000, sinyal tersebut menghasilkan tegangan sekitar 1,030 V setelah melewati *RC filter*, kemudian Arduino membacanya sebagai ADC 1153 dan mengolahnya menjadi *heater power* 28%.

Nilai 28% tidak harus sama persis apabila nilai 20000 langsung dibandingkan dengan batas maksimum 65535. Hal tersebut karena nilai tidak berpindah secara langsung sebagai angka digital, melainkan melewati proses PWM, *RC filter*, pembacaan ADC, dan filtering pada Arduino. Namun, arah perubahan nilainya sudah sesuai dan menunjukkan bahwa perintah yang diberikan dari *OpenPLC* berhasil diterima pada sisi Arduino.

4.1.3 Pengujian Pengiriman Nilai 40000 pada *OpenPLC*

Pada pengujian ketiga, nilai PWM kembali dinaikkan menjadi 40000. Nilai ini hampir dua kali lebih besar dibandingkan pengujian sebelumnya, sehingga hasil pengukuran tegangan dan ADC seharusnya juga mengalami peningkatan apabila komunikasi berjalan sesuai.

Seperti pada pengujian sebelumnya, nilai dari *OpenPLC* dikirim melalui PWM kemudian dilewatkan pada *RC filter*. Tegangan hasil filter diukur terlebih dahulu sebelum masuk ke ADC Arduino.

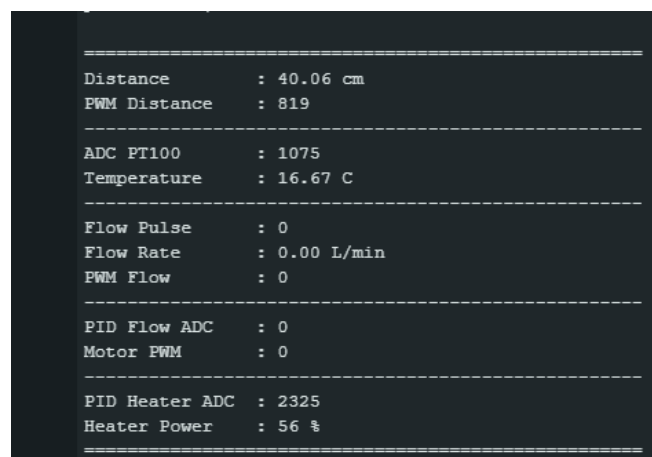


Gambar 4.5 Pembacaan Arus PWM 40000 *OpenPLC*

Berdasarkan Gambar 4.5, ketika nilai PWM dinaikkan menjadi 40000, tegangan hasil *RC filter* terbaca sebesar 1,970 V. Nilai ini hampir dua kali lebih besar dibandingkan tegangan pada pengiriman PWM 20000 sebesar 1,030 V.

Hasil tersebut memperlihatkan bahwa kenaikan nilai PWM yang diberikan dari *OpenPLC* masih diikuti oleh kenaikan tegangan keluaran *RC filter*. Hal ini penting karena Arduino tidak membaca nilai 40000 tersebut secara langsung, tetapi membaca tegangan yang terbentuk dari hasil proses PWM dan *RC filter*.

Tegangan sebesar 1,970 V kemudian masuk ke ADC Arduino dan hasil pembacaannya diamati pada Serial Monitor.



Gambar 4.6 Pembacaan PWM Arduino IDE

Berdasarkan Gambar 4.6, pada kondisi tersebut Arduino membaca PID *Heater* ADC sebesar 2325 dan menghasilkan *heater power* sebesar 56%. Jika dibandingkan

dengan pengujian sebelumnya, ADC meningkat dari 1153 menjadi 2325, sedangkan *heater power* meningkat dari 28% menjadi 56%. Kenaikannya terlihat cukup konsisten. Nilai PWM dari *OpenPLC* dinaikkan dari 20000 menjadi 40000, lalu tegangan hasil filter naik dari 1,030 V menjadi 1,970 V dan nilai ADC juga naik hampir dua kali lipat.

Hasil ini menunjukkan bahwa komunikasi *pin to pin* tidak hanya dapat membedakan kondisi rendah dan tinggi, tetapi juga dapat meneruskan perubahan nilai secara bertahap. Nilai yang dikirim semakin besar, tegangan hasil *RC filter* semakin tinggi, dan ADC yang diterima Arduino ikut meningkat.

4.1.4 Pengujian Pengiriman Nilai 65535 pada *OpenPLC*

Pengujian terakhir dilakukan dengan memberikan nilai maksimum sebesar 65535 pada *OpenPLC*. Kondisi ini digunakan untuk melihat batas atas komunikasi dan memastikan apakah nilai maksimum dari *OpenPLC* dapat diterima sebagai kondisi maksimum pada Arduino.

Pada kondisi ini sinyal PWM berada pada keluaran tertinggi. Tegangan hasil *RC filter* kemudian diukur untuk melihat tegangan maksimum yang diterima pada sisi Arduino.

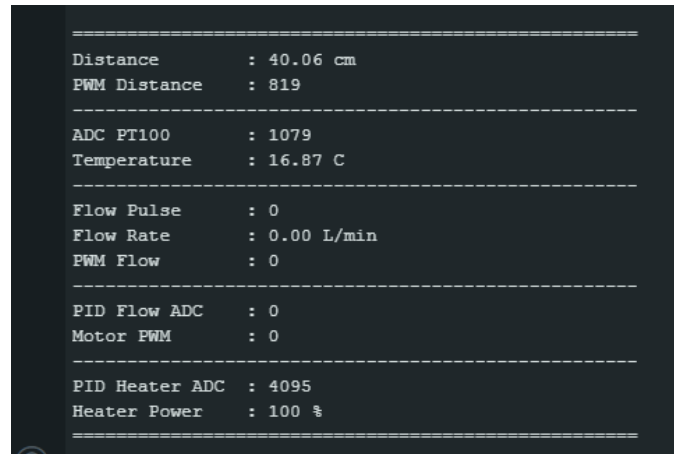


Gambar 4.7 Pembacaan Arus PWM 65535 *OpenPLC*

Berdasarkan Gambar 4.7, pada saat nilai PWM maksimum 65535 dikirim, tegangan setelah *RC filter* terbaca sebesar 3,151 V. Nilai ini menjadi tegangan tertinggi dari seluruh kondisi pengujian.

Jika dilihat dari keseluruhan pengukuran, tegangan meningkat dari sekitar 0,098 V pada PWM 0, menjadi 1,030 V pada PWM 20000, 1,970 V pada PWM

40000, dan akhirnya 3,151 V pada PWM maksimum. Pola tersebut menunjukkan bahwa keluaran *RC filter* mengikuti perubahan nilai PWM yang diberikan. Setelah tegangan maksimum tersebut masuk ke ADC Arduino, hasil pembacaannya diamati melalui *Serial Monitor*.



Gambar 4.8 Pembacaan PWM Arduino IDE

Berdasarkan Gambar 4.8, Arduino membaca PID *Heater* ADC sebesar 4095 dan *heater power* sebesar 100%. Nilai ADC 4095 merupakan batas maksimum pembacaan ADC 12 bit yang digunakan pada ESP32.

Hasil ini menunjukkan bahwa nilai maksimum yang dikirim dari *OpenPLC* berhasil diterima sebagai nilai maksimum pada sisi Arduino. Tegangan sebesar 3,151 V sudah menghasilkan pembacaan ADC penuh sehingga program Arduino menghasilkan *heater power* sebesar 100%.

Pada kondisi maksimum, hasil antara sisi pengirim dan penerima terlihat paling jelas. *OpenPLC* mengirim nilai maksimum 65535 dan Arduino menerima kondisi maksimum ADC 4095 dengan hasil akhir 100%. Artinya jalur komunikasi dapat digunakan mulai dari kondisi nol sampai kondisi maksimum.

4.1.5 Analisis Keseluruhan Komunikasi *Pin to pin*

Setelah seluruh kondisi pengujian selesai dilakukan, hasil dari masing-masing nilai PWM kemudian dibandingkan untuk melihat perubahan yang terjadi pada setiap bagian komunikasi. Perbandingan dilakukan mulai dari nilai PWM yang dikirim dari *OpenPLC*, tegangan yang terbentuk setelah melewati rangkaian RC Low Pass Filter, nilai ADC yang diterima pada Arduino IDE, hingga hasil akhir *Heater Power* yang ditampilkan pada program.

Dari hasil pengujian tersebut terlihat bahwa setiap kenaikan nilai PWM yang diberikan dari *OpenPLC* diikuti oleh kenaikan tegangan keluaran *RC filter*. Kenaikan tegangan tersebut kemudian terbaca sebagai nilai ADC yang semakin besar pada Arduino IDE dan akhirnya menghasilkan nilai *heater power* yang juga meningkat. Untuk mempermudah melihat hubungan antar nilai tersebut, seluruh hasil pengujian dirangkum pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian Komunikasi *Pin to pin*

Nilai PWM <i>OpenPLC</i>	Duty Cycle Teoritis	Tegangan Ideal	Tegangan setelah <i>RC</i> <i>filter</i>	PID <i>Heater</i> ADC	<i>Heater</i> Power
0	0%	0 V	0,098 V	15	0%
20000	30,52%	1,007 V	1,030 V	1153	28%
40000	61,04%	2,014 V	1,970 V	2325	56%
65535	100%	3,300 V	3,151 V	4095	100%

Berdasarkan Tabel 4.1 tersebut, kenaikan nilai PWM *OpenPLC* diikuti oleh kenaikan *duty cycle* teoritis, tegangan ideal, tegangan aktual setelah *RC filter*, dan nilai ADC Arduino. Pola perubahan yang searah tersebut menunjukkan bahwa jalur komunikasi mampu meneruskan perubahan nilai kendali dari sisi *OpenPLC* ke ESP32 Arduino IDE.

Pada nilai 0 masih terdapat tegangan sisa sekitar 0,098 V dan ADC sebesar 15, tetapi hasil akhirnya tetap dibaca sebagai 0%. Ketika nilai dinaikkan menjadi 20000, tegangan menjadi 1,030 V dan Arduino membaca ADC 1153 dengan hasil 28%. Pada nilai 40000, tegangan naik menjadi 1,970 V, ADC menjadi 2325, dan hasilnya 56%. Pada nilai maksimum 65535, tegangan mencapai 3,151 V dan ADC mencapai 4095 sehingga *heater power* terbaca 100%.

Nilai *OpenPLC* menentukan *duty cycle* PWM. Karena rentang keluaran yang digunakan adalah 0–65535, nilai *duty cycle* secara ekuivalen dapat dihitung sebagai $D = \text{PWM}/65535$. Mengacu pada Persamaan (2.5), PWM 0, 20000, 40000, dan 65535 setara dengan *duty cycle* sekitar 0%, 30,52%, 61,04%, dan 100%.

Selanjutnya, dengan tegangan logika HIGH ESP32 sekitar 3,3 V dan mengacu pada Persamaan (2.6), tegangan rata-rata ideal menjadi sekitar 0 V, 1,007 V, 2,014 V, dan 3,300 V. Hasil pengukuran aktual masing-masing adalah 0,098 V, 1,030 V,

1,970 V, dan 3,151 V. Perbedaan tersebut menunjukkan pengaruh tegangan HIGH aktual, *RC filter*, beban ADC, sambungan, noise, dan toleransi alat ukur.

Pada kondisi minimum masih terdapat tegangan 0,098 V dan ADC 15. Nilai sisa ini tetap menghasilkan *heater power* 0% sehingga sistem masih dapat membedakan kondisi tanpa perintah daya.

Pada kondisi maksimum, nilai teoritis berdasarkan Persamaan (2.6) adalah sekitar 3,3 V, sedangkan hasil pengukuran 3,151 V. Nilai tersebut tetap menghasilkan ADC maksimum 4095 dan *heater power* 100%, sehingga rentang komunikasi minimum–maksimum dapat diterima oleh sistem.

Hasil tersebut menunjukkan bahwa jalur *OpenPLC*, PWM, *RC filter*, ADC Arduino bekerja secara monoton. Kenaikan nilai PWM diikuti kenaikan tegangan dan pembacaan ADC. Angka pada sisi penerima tidak harus sama dengan angka mentah *OpenPLC* karena bentuk dan skala datanya berubah selama proses komunikasi.

Dengan demikian, Persamaan (2.5) digunakan untuk menilai duty cycle teoritis, sedangkan Persamaan (2.6) digunakan untuk memperoleh tegangan rata-rata ideal sebagai pembanding terhadap hasil pengukuran. Kedua persamaan ini digunakan untuk analisis komunikasi *pin to pin*, bukan untuk menghitung daya *heater* AC secara langsung.

Secara keseluruhan, komunikasi *pin to pin* dapat meneruskan perubahan nilai kendali dari kondisi minimum sampai maksimum. Selisih antara nilai teoritis dan aktual menjadi karakteristik jalur konversi PWM–RC–ADC dan perlu dipertimbangkan pada proses *scaling*.

4.2 Pengujian Pembacaan Sensor PT100

Pengujian sensor PT100 dilakukan untuk melihat hasil pembacaan temperatur mulai dari bagian sensor sampai nilai tersebut ditampilkan pada sistem. Pada pengujian ini, PT100 dipasang pada air di dalam tangki sehingga perubahan temperatur air dapat dibaca oleh sensor. Sinyal dari PT100 diproses oleh RTD transmitter, kemudian dibaca oleh ADC ESP32 yang diprogram menggunakan Arduino IDE.

Data hasil pembacaan tersebut tidak langsung masuk ke *OpenPLC* dalam bentuk angka temperatur. ESP32 Arduino mengirimkan data menuju ESP32

OpenPLC melalui komunikasi *pin to pin* menggunakan sinyal PWM. Pada sisi penerima, sinyal PWM dilewatkan melalui rangkaian *RC Filter* sehingga diperoleh tegangan analog yang dapat dibaca kembali oleh ADC ESP32 *OpenPLC*. Nilai ADC tersebut kemudian dilakukan *scaling* menjadi temperatur dan nilai akhirnya ditampilkan pada HMI.

4.2.1 Hasil Pembacaan PT100 pada Arduino IDE

Pengamatan pertama dilakukan pada sisi ESP32 Arduino IDE untuk melihat hasil pembacaan awal sensor PT100. Bagian ini digunakan untuk mengetahui nilai ADC yang diperoleh dari sensor dan temperatur hasil pengolahannya sebelum data dikirim menuju *OpenPLC*. Dengan melihat hasil pada *Serial Monitor*, dapat diketahui kondisi data pada sisi pengirim sebelum melewati komunikasi *pin to pin*.

Pada saat pengujian, temperatur air masih berada di sekitar temperatur lingkungan. *Serial Monitor* digunakan untuk melihat nilai ADC PT100 dan temperatur hasil konversinya secara langsung. Hasil pembacaan sensor PT100 pada Arduino IDE ditunjukkan pada Gambar 4.9.

```
=====
Distance      : 29.74 cm
PWM Distance  : 593
=====
ADC PT100     : 1350
Temperature   : 30.51 C
=====
Flow Pulse    : 0
Flow Rate     : 0.00 L/min
PWM Flow      : 0
=====
PID Flow ADC  : 0
Motor PWM     : 0
=====
PID Heater ADC : 26
Heater Power  : 0 %
=====
```

Gambar 4.9 Pembacaan Temperatur Arduino IDE

Berdasarkan Gambar 4.9, nilai ADC PT100 yang terbaca pada salah satu kondisi pengujian adalah sebesar 1350, sedangkan temperatur hasil konversinya sebesar 30,51 °C. Hasil tersebut menunjukkan bahwa sinyal yang berasal dari PT100 sudah dapat dibaca oleh ADC ESP32 dan diolah menjadi nilai temperatur.

Nilai ADC menjadi bagian penting dalam pembacaan ini karena temperatur yang ditampilkan berasal dari hasil pengolahan nilai tersebut. Apabila nilai ADC mengalami perubahan, nilai temperatur hasil konversi juga akan ikut berubah. Pada pembacaan sinyal analog, nilai ADC memang tidak selalu berada pada angka yang benar-benar tetap. Perubahan kecil masih dapat muncul akibat perubahan sinyal

keluaran RTD *transmitter*, *noise* pada jalur analog, maupun karakteristik pembacaan ADC ESP32.

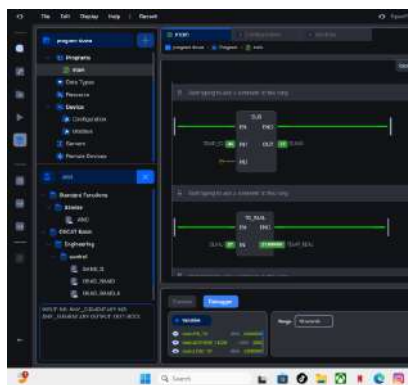
Untuk mengurangi perubahan tersebut, pembacaan ADC pada Arduino dilakukan secara berulang, kemudian hasilnya dirata-ratakan dan diproses kembali menggunakan *filtering*. Dengan proses tersebut, perubahan kecil pada pembacaan sensor dapat dikurangi sehingga nilai yang akan dikirim menuju *OpenPLC* menjadi lebih stabil dibandingkan apabila hanya menggunakan satu kali pembacaan ADC.

Setelah nilai temperatur diperoleh, data tersebut kemudian diubah menjadi sinyal PWM untuk dikirim menuju ESP32 *OpenPLC*. Dengan demikian, pembacaan pada Arduino IDE dapat digunakan sebagai acuan untuk melihat kondisi data pada sisi awal sebelum melewati rangkaian komunikasi.

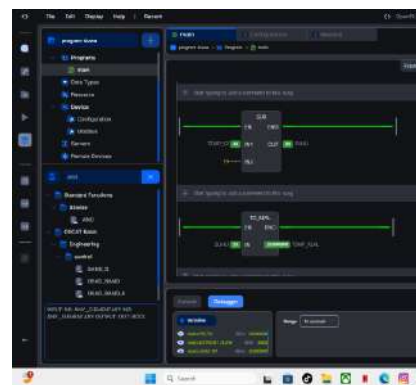
4.2.2 Hasil Pembacaan Temperatur pada *OpenPLC*

Setelah data dikirim dari ESP32 Arduino, pengamatan berikutnya dilakukan pada ESP32 yang menjalankan *OpenPLC*. Pada bagian ini data sudah melewati beberapa tahapan, yaitu pengubahan data menjadi sinyal PWM, pengiriman melalui komunikasi *pin to pin*, proses pada *RC Filter*, pembacaan ADC kembali, dan proses *scaling* untuk memperoleh nilai temperatur.

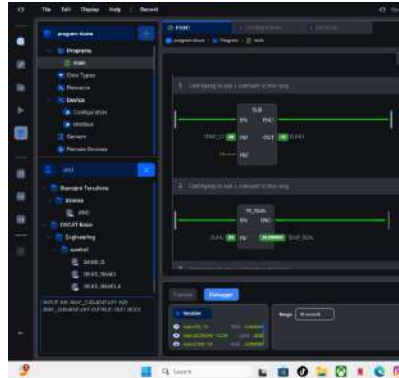
Pengamatan pada *OpenPLC* dilakukan melalui debugger untuk melihat nilai yang diterima secara langsung selama sistem bekerja. Berbeda dengan nilai pada Arduino IDE yang berasal dari pembacaan awal sensor, nilai pada *OpenPLC* merupakan hasil setelah data melewati seluruh jalur komunikasi tersebut. Hasil pembacaan temperatur pada *OpenPLC* ditunjukkan pada Gambar 4.9.



Gambar 4.10 Pembacaan Temperatur *OpenPLC*



Gambar 4.11 Pembacaan Temperatur *OpenPLC*



Gambar 4.12 Pembacaan Temperatur *OpenPLC*

Berdasarkan Gambar 4.10 – 4.12, nilai yang diterima pada *OpenPLC* terlihat mengalami perubahan selama proses pengamatan. Pada salah satu kondisi, nilai TEMP_S2 sebesar 46 menghasilkan nilai TEMPERATUR sebesar 27 °C. Pada pembacaan berikutnya, nilai TEMP_S2 dapat berubah menjadi 48 dan 49 sehingga nilai temperatur yang diperoleh berubah menjadi sekitar 29 °C dan 30 °C.

Perubahan nilai tersebut lebih terlihat pada sisi *OpenPLC* karena data yang diterima telah melewati jalur komunikasi tambahan. Data dari Arduino terlebih dahulu dikirim dalam bentuk sinyal PWM. Sinyal PWM tersebut kemudian masuk ke rangkaian *RC Filter*. Rangkaian ini digunakan untuk meratakan sinyal PWM sehingga menghasilkan tegangan analog yang dapat dibaca kembali oleh ADC pada ESP32 *OpenPLC*.

Walaupun sudah melewati *RC filter*, tegangan keluarannya tidak selalu benar-benar berada pada satu nilai yang tetap. Masih dapat terjadi perubahan kecil pada tegangan hasil *filter*. Perubahan tersebut kemudian ikut terbaca oleh ADC ESP32 *OpenPLC*.

Selain itu, pembacaan ADC pada ESP32 juga tidak selalu menghasilkan angka yang sama pada setiap siklus pembacaan. Walaupun tegangan masukan relatif tetap, nilai ADC masih dapat berubah beberapa satuan. Perubahan inilah yang selanjutnya masuk ke proses *scaling*.

Setelah nilai ADC diperoleh, *OpenPLC* melakukan *scaling* untuk mengubah nilai tersebut menjadi temperatur. Karena hasil temperatur berasal dari nilai ADC, perubahan kecil pada ADC dapat menyebabkan nilai hasil *scaling* ikut berubah.

Hal tersebut dapat dilihat langsung dari hasil pengujian. Ketika nilai TEMP_S2 berubah dari 46 menjadi 48 atau 49, hasil temperatur ikut berubah dari 27 °C

menjadi 29 °C atau 30 °C. Padahal secara fisik temperatur air tidak berubah beberapa derajat dalam waktu yang sangat singkat.

Dari hasil tersebut dapat dilihat bahwa perubahan nilai temperatur pada *OpenPLC* tidak seluruhnya menunjukkan perubahan temperatur air yang sebenarnya. Sebagian perubahan tersebut berasal dari proses pengiriman sinyal PWM, tegangan hasil *RC filter*, pembacaan ADC, dan proses *scaling* sebelum nilai akhirnya menjadi temperatur.

4.2.3 Hasil Pembacaan Temperatur pada HMI

Nilai temperatur yang telah diperoleh pada *OpenPLC* selanjutnya digunakan sebagai process variable dan ditampilkan pada HMI. Pada bagian ini HMI tidak membaca PT100 secara langsung. Nilai yang ditampilkan HMI berasal dari nilai temperatur yang sudah diproses pada *OpenPLC*.

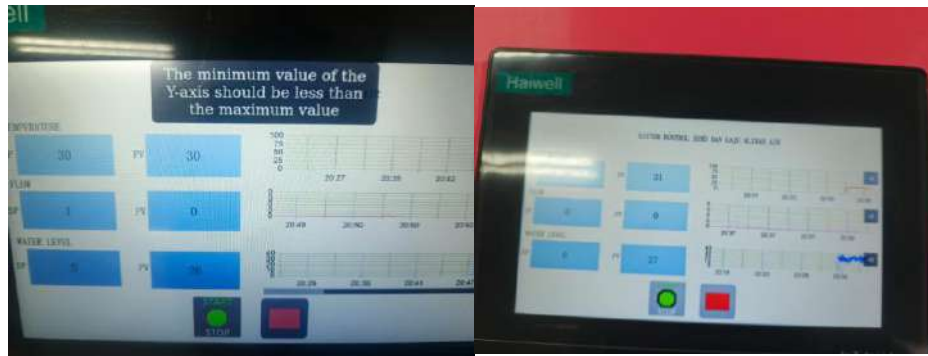
Pengamatan pada HMI dilakukan pada waktu yang sama dengan pengamatan pada *OpenPLC*. Hal ini dilakukan untuk melihat apakah perubahan nilai yang muncul pada *OpenPLC* juga ikut terlihat pada HMI.



Gambar 4.13 Tampilan Temperatur Pada HMI



Gambar 4.14 Tampilan Temperatur Pada HMI



Gambar 4.15 Tampilan Temperatur Pada HMI

Berdasarkan Gambar 4.13 - 4.15, nilai temperatur yang ditampilkan pada HMI terlihat berubah pada kisaran sekitar 26 °C sampai 31 °C selama proses pengamatan. Nilai tersebut berubah ketika sistem terus melakukan pembaruan data temperatur.

Perubahan pada HMI mengikuti nilai yang berasal dari *OpenPLC*. Ketika hasil pengolahan pada *OpenPLC* berubah, nilai yang ditampilkan pada HMI juga ikut berubah. Oleh karena itu, perubahan yang terlihat pada HMI bukan berasal dari proses pembacaan sensor baru pada HMI, melainkan merupakan hasil akhir dari nilai temperatur yang sebelumnya sudah diproses oleh *OpenPLC*.

Hal tersebut juga terlihat pada grafik temperatur HMI. Grafik mengalami kenaikan dan penurunan dalam waktu yang relatif berdekatan. Perubahan tersebut tidak berarti temperatur air benar-benar mengalami kenaikan dan penurunan sebesar angka yang terlihat pada tampilan.

Sebagai contoh, apabila nilai pada HMI berubah dari 28 °C menjadi 30 °C kemudian kembali menjadi 29 °C dalam waktu yang singkat, kondisi tersebut lebih memungkinkan berasal dari perubahan nilai yang diterima *OpenPLC*. Temperatur air memiliki perubahan yang lebih lambat dibandingkan perubahan angka hasil pembacaan elektronik.

Kondisi ini berhubungan langsung dengan hasil pengamatan pada *OpenPLC*. Ketika nilai TEMP_S2 berubah, nilai hasil *scalling* temperatur juga berubah. Nilai inilah yang kemudian diteruskan ke HMI. Dengan demikian, fluktuasi yang terlihat pada HMI merupakan lanjutan dari fluktuasi yang sudah muncul pada sisi *OpenPLC*..

4.2.4 Perbandingan Pembacaan Sistem dengan Temperatur Aktual

Untuk mengetahui apakah hasil pembacaan sistem masih berada di sekitar kondisi temperatur sebenarnya, dilakukan pengukuran temperatur air secara langsung menggunakan *Thermometer*. Pengukuran ini digunakan sebagai pembandingan terhadap hasil pembacaan yang diperoleh pada Arduino IDE, *OpenPLC*, dan HMI.

Pengukuran dilakukan pada air di dalam tangki yang sama dengan tempat sensor PT100 melakukan pembacaan. Dengan begitu, hasil dari *Thermometer* dapat digunakan untuk melihat seberapa dekat temperatur yang dibaca sistem terhadap kondisi aktual air.



Gambar 4.16 Temperatur Aktual Air Pada *Thermometer*

Berdasarkan Gambar 4.16, temperatur aktual air yang terbaca menggunakan *Thermometer* berada di sekitar 29,5 °C. Pada kondisi pengujian tersebut, Arduino IDE menunjukkan nilai temperatur sekitar 30,51 °C, sedangkan nilai pada *OpenPLC* dan HMI bergerak di sekitar temperatur aktual tersebut.

Jika dibandingkan, pembacaan Arduino IDE sebesar 30,51 °C memiliki selisih sekitar 1,01 °C terhadap temperatur aktual sebesar 29,5 °C. Selisih tersebut menunjukkan bahwa pembacaan awal PT100 masih berada cukup dekat dengan kondisi temperatur sebenarnya.

Hasil tersebut juga memberikan gambaran bahwa perubahan nilai yang cukup besar pada *OpenPLC* dan HMI bukan sepenuhnya berasal dari sensor PT100. Pada sisi Arduino, pembacaan awal sudah berada di sekitar 30 °C dan cukup dekat

dengan temperatur aktual. Fluktuasi yang lebih jelas mulai terlihat setelah data melewati jalur komunikasi menuju *OpenPLC*.

Dengan demikian, hasil pengukuran temperatur aktual digunakan bukan hanya untuk melihat selisih pembacaan sensor, tetapi juga untuk membantu menentukan bagian mana dari sistem yang memberikan pengaruh lebih besar terhadap perubahan nilai temperatur yang ditampilkan.

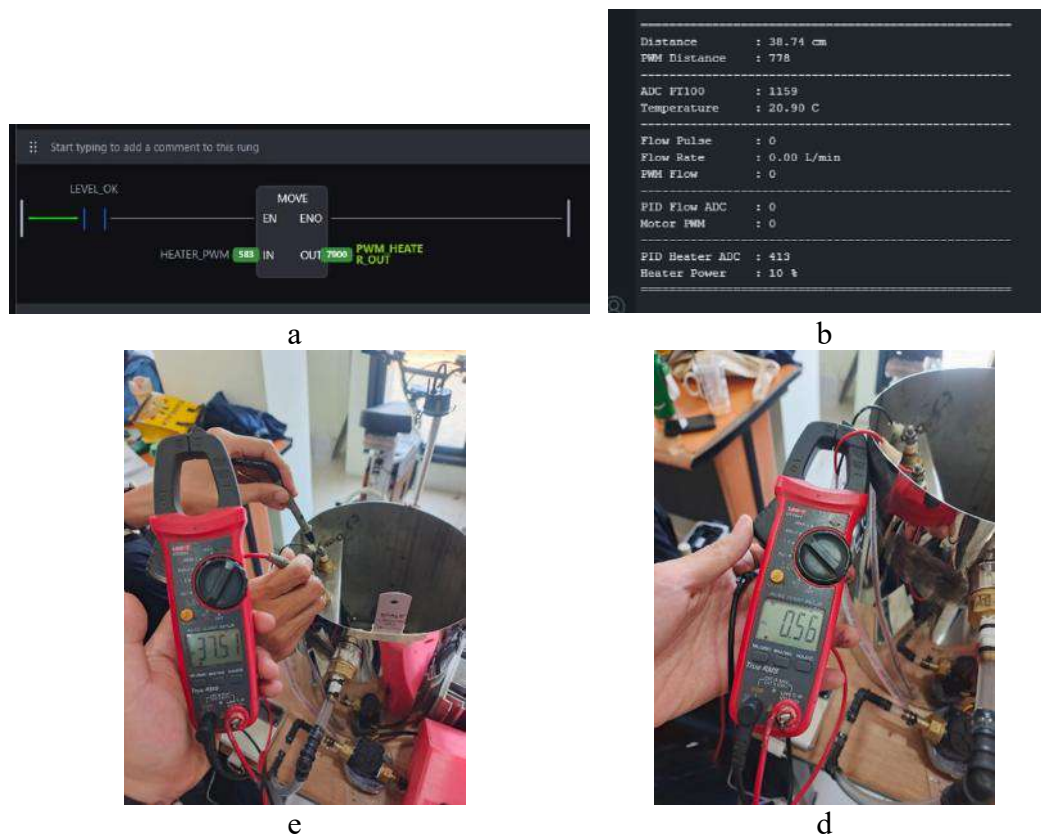
4.3 Pengujian Sistem Pengaturan Daya Heater

Pengujian keluaran *Heater* dilakukan untuk mengetahui apakah nilai kendali yang diberikan dari *OpenPLC* dapat diterima dengan baik oleh ESP32 Arduino IDE dan menghasilkan perubahan daya pada *heater*. Pengujian ini dilakukan dengan memberikan beberapa nilai PWM yang berbeda dari *OpenPLC*, kemudian mengamati nilai yang diterima pada Arduino IDE serta melakukan pengukuran arus secara langsung pada *Heater*.

Pengujian dilakukan sebanyak tiga kali dengan nilai keluaran *OpenPLC* yang berbeda. Pada setiap percobaan dilakukan tiga tahap pengamatan, yaitu melihat nilai PWM yang dikirim dari *OpenPLC*, melihat nilai ADC dan persentase *heater Power* yang diterima pada *Serial Monitor* Arduino IDE, kemudian mengukur arus aktual yang mengalir pada *heater* menggunakan *clamp meter*. Dari pengujian ini dapat dilihat hubungan antara kenaikan nilai kendali dari *OpenPLC* dengan kenaikan persentase daya *heater* dan arus yang mengalir pada *heater*.

4.3.1 Pengujian Daya Heater 10%

Pengujian *Heater Power* 10% dilakukan sebagai tahap awal untuk mengetahui respons AC *Dimmer* ketika menerima perintah kendali pada *level* rendah. Pada pengujian ini, nilai *Heater Power* diatur melalui *OpenPLC* kemudian diteruskan menuju ESP32 Arduino IDE sebagai antarmuka perangkat keras yang mengendalikan AC *Dimmer*. Tegangan AC diukur pada sisi keluaran *dimmer* yang terhubung ke *Heater*, sedangkan arus diukur pada jalur beban *Heater*. Pengujian pada *level* 10% digunakan sebagai kondisi acuan awal sebelum nilai *Heater Power* dinaikkan secara bertahap pada pengujian berikutnya. Data yang diperoleh selanjutnya digunakan untuk melihat hubungan antara perintah kendali, tegangan keluaran *dimmer*, dan arus yang mengalir pada *Heater*.



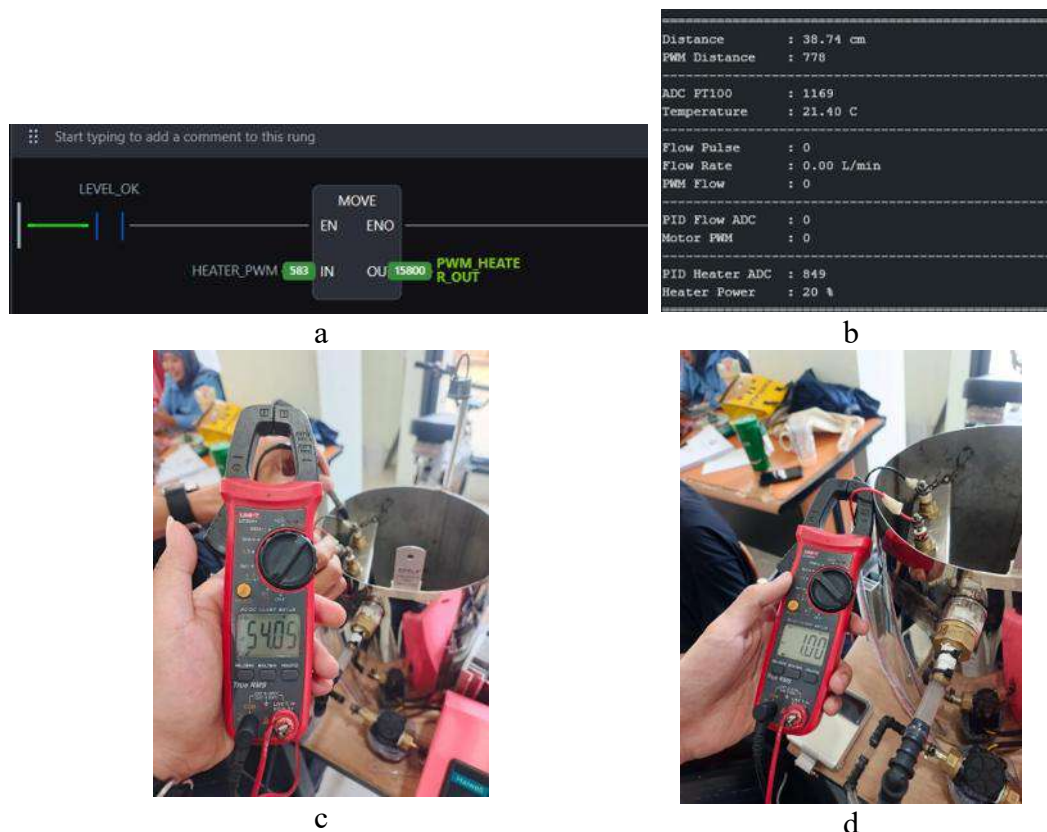
Gambar 4.17 a.) Input *OpenPLC* b.) Keluar Arduino IDE 10% c.) Pengukuran Tegangan 10% d.) Pengukuran Arus 10%

Berdasarkan Gambar 4.17, ketika *Heater Power* ditetapkan sebesar 10%, tegangan AC yang terukur pada keluaran AC *Dimmer* sebesar 37,6 Vac RMS dan arus yang mengalir pada *Heater* sebesar 0,56 A. Kondisi ini merupakan tingkat keluaran terendah pada rangkaian pengujian. Pada perintah yang masih kecil, TRIAC dipicu lebih lambat setelah titik *zero crossing* sehingga hanya sebagian akhir dari setiap setengah gelombang AC yang diteruskan menuju *Heater*. Akibatnya, nilai tegangan efektif yang diterima beban masih rendah dan arus yang mengalir juga masih kecil. Hasil ini menunjukkan bahwa AC *Dimmer* sudah dapat memberikan keluaran rendah ketika sistem memerintahkan *Heater* bekerja pada *level* 10%. Dengan demikian, *level* 10% dapat digunakan sebagai titik dasar untuk membaca perubahan keluaran ketika perintah *Heater Power* dinaikkan pada level berikutnya.

4.3.2 Pengujian Daya *Heater* 20%

Pengujian *Heater Power* 20% dilakukan dengan menaikkan perintah kendali satu tingkat dari kondisi sebelumnya. Tujuan pengujian ini adalah melihat apakah perubahan nilai *Heater Power* pada *OpenPLC* diikuti oleh perubahan keluaran

listrik AC *Dimmer* yang menuju *Heater*. Pengukuran tetap dibuat sama, yaitu mengamati nilai perintah pada sistem, kemudian mengukur tegangan AC pada keluaran *dimmer* serta arus yang mengalir pada *Heater*. Dengan menggunakan prosedur yang sama pada setiap *level*, hasil pengukuran dapat dibandingkan secara langsung dengan kondisi 10% tanpa mengubah rangkaian maupun beban yang digunakan.



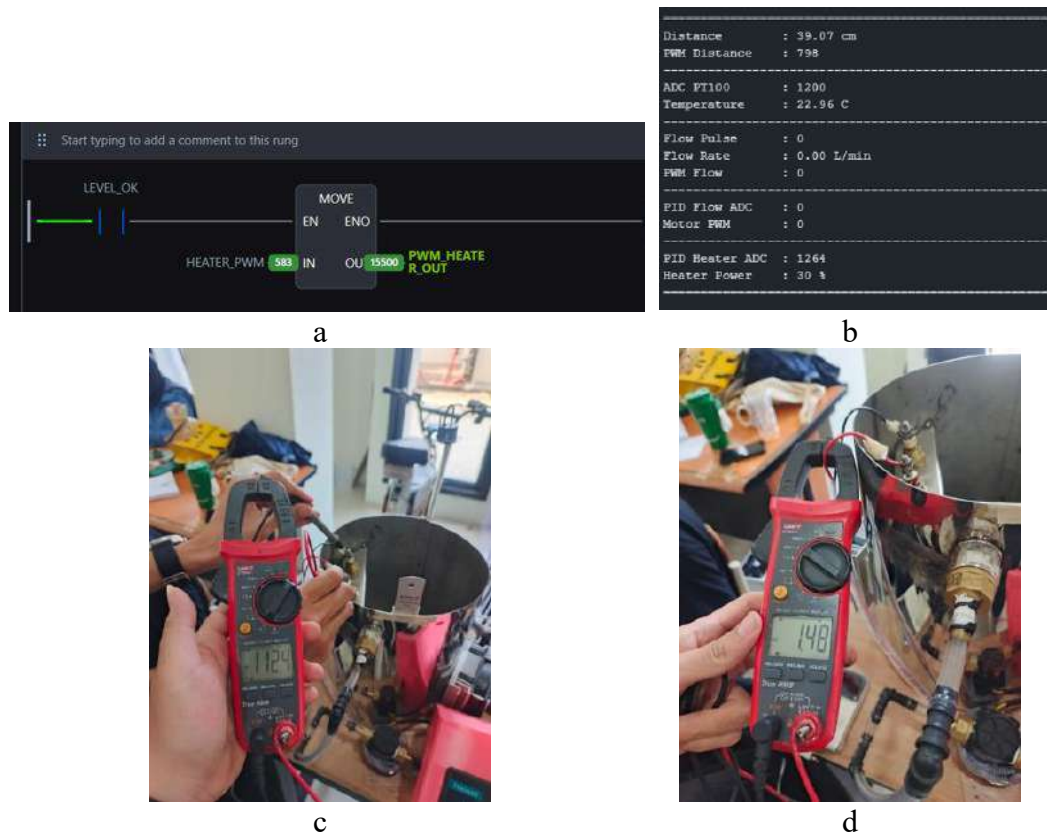
Gambar 4.18 a.) Input *OpenPLC* b.) Keluar *Arduino IDE* 20% c.) Pengukuran Tegangan 20% d.) Pengukuran Arus 20%

Berdasarkan Gambar 4.18, pada *Heater Power* 20% diperoleh tegangan keluaran sebesar 54,05 Vac RMS dan arus *Heater* sebesar 1,00 A. Jika dibandingkan dengan pengujian 10%, tegangan meningkat sebesar 16,45 V dan arus meningkat sebesar 0,44 A. Kenaikan tersebut menunjukkan bahwa penambahan perintah *Heater Power* membuat waktu penyalaan TRIAC menjadi lebih awal sehingga bagian gelombang AC yang diteruskan ke *Heater* bertambah. Walaupun nilai perintah berubah dari 10% menjadi 20%, tegangan keluaran tidak menjadi dua kali lipat. Hal tersebut menegaskan bahwa *Heater Power* merupakan perintah *phase angle control* dan bukan skala linier langsung terhadap tegangan AC keluaran. Data

ini memperlihatkan bahwa perubahan pertama dari 10% ke 20% sudah menghasilkan perubahan listrik yang dapat terukur pada beban *Heater*.

4.3.3 Pengujian Daya *Heater* 30%

Pengujian *Heater Power* 30% dilakukan untuk melanjutkan pengamatan karakteristik AC *Dimmer* pada daerah kendali rendah menuju menengah. Nilai perintah ditetapkan melalui *OpenPLC*, kemudian sinyal kendali diteruskan ke sisi Arduino IDE untuk menentukan kerja AC Dimmer. Pengukuran difokuskan pada tegangan AC yang diterima *Heater* dan arus yang mengalir pada beban dengan susunan rangkaian yang sama seperti pengujian sebelumnya. Data pada level ini digunakan untuk menilai apakah kenaikan perintah *Heater Power* menghasilkan perubahan keluaran yang tetap berurutan serta untuk melihat kecenderungan hubungan antara persentase kendali dan besaran listrik aktual pada *Heater*.



Gambar 4.19 a.) Input *OpenPLC* b.) Keluar Arduino IDE 30% c.) Pengukuran Tegangan 30% d.) Pengukuran Arus 30%

Berdasarkan Gambar 4.19, pada *Heater Power* 30% tegangan keluaran meningkat menjadi 112,4 Vac RMS dengan arus *Heater* sebesar 1,48 A. Dibandingkan pengujian 20%, terjadi kenaikan tegangan sebesar 58,35 V dan kenaikan arus sebesar 0,48 A. Kenaikan tegangan pada interval ini terlihat lebih

besar dibandingkan interval 10% ke 20%. Perbedaan besar kenaikan antar *level* merupakan karakteristik yang dapat muncul pada pengaturan phase angle karena perubahan sudut penyalan memengaruhi bentuk gelombang AC dan nilai efektifnya secara tidak linier. Hasil pengukuran menunjukkan bahwa pada *level* 30% energi listrik yang diteruskan menuju Heater sudah meningkat secara nyata dibandingkan dua kondisi sebelumnya. Kondisi tersebut memperlihatkan bahwa respons *dimmer* pada daerah ini mulai memberikan kenaikan energi pemanasan yang lebih besar dibandingkan level awal.

4.3.4 Pengujian Daya Heater 40%

Pengujian *Heater Power* 40% dilakukan untuk mengamati keluaran AC *Dimmer* ketika perintah kendali mulai memasuki daerah menengah. Sistem tetap menggunakan *Heater* yang sama sebagai beban sehingga perubahan yang diamati berasal dari perubahan perintah pengendalian pada *dimmer*. Nilai *Heater Power* diberikan melalui *OpenPLC* dan diteruskan menuju pengendali *dimmer*, kemudian dilakukan pengukuran tegangan AC pada keluaran dan arus *Heater*. Hasil pengukuran pada kondisi ini selanjutnya dibandingkan dengan level 30% untuk melihat kecenderungan perubahan tegangan dan arus ketika sudut penyalan TRIAC diubah oleh perintah kendali yang lebih besar.



a



Distance	: 27.92 cm
PWM Distance	: 552
ADC PI100	: 1220
Temperature	: 23.97 C
Flow Pulse	: 0
Flow Rate	: 0.00 L/min
PWM Flow	: 0
PID Flow ADC	: 0
Motor PWM	: 0
PID Heater ADC	: 1649
Heater Power	: 40 %

b



c d

Gambar 4.20 a.) Input OpenPLC b.) Keluar Arduino IDE 40% c.) Pengukuran Tegangan 40% d.) Pengukuran Arus 40%

Berdasarkan Gambar 4.20, pada *Heater Power* 40% diperoleh tegangan sebesar 141,1 Vac RMS dan arus *Heater* sebesar 1,78 A. Nilai tersebut lebih tinggi daripada kondisi 30%, yaitu bertambah 28,7 V pada tegangan dan 0,30 A pada arus. Kenaikan ini menunjukkan bahwa peningkatan perintah kendali masih diikuti oleh bertambahnya bagian gelombang AC yang dihantarkan ke *Heater*. Namun, besar perubahan tegangan tidak sama dengan besar perubahan persentase perintah. Dengan demikian, pada level 40% sistem telah memberikan keluaran pemanasan yang lebih besar daripada level sebelumnya, tetapi hubungan antara *Heater Power* dan Vac RMS tetap bersifat nonlinier. Hasil tersebut memperkuat bahwa perintah kendali mampu menggeser kondisi penghantaran dimmer secara bertahap menuju keluaran yang lebih tinggi.

4.3.5 Pengujian Daya *Heater* 50%

Pengujian *Heater Power* 50% dilakukan pada titik tengah rentang perintah kendali yang digunakan dalam penelitian. Pengujian ini bertujuan mengetahui karakteristik keluaran AC *Dimmer* ketika perintah sistem berada pada setengah skala *Heater Power*, tanpa mengasumsikan bahwa tegangan keluarannya juga harus bernilai setengah dari tegangan sumber. Hal tersebut karena *Heater Power* digunakan sebagai perintah untuk pengaturan sudut penyalaan TRIAC pada metode phase angle control. Tegangan keluaran *dimmer* dan arus yang mengalir pada *Heater* tetap diukur secara langsung agar kondisi listrik aktual pada beban dapat diketahui dan dibandingkan dengan level 40% maupun level berikutnya.



a

Distance	: 38.48 cm
PWM Distance	: 778
ADC PT100	: 1260
Temperature	: 25.98 C
Flow Pulse	: 0
Flow Rate	: 0.00 L/min
PWM Flow	: 0
FID Flow ADC	: 0
Motor PWM	: 0
FID Heater ADC	: 2064
Heater Power	: 50 %

b



c



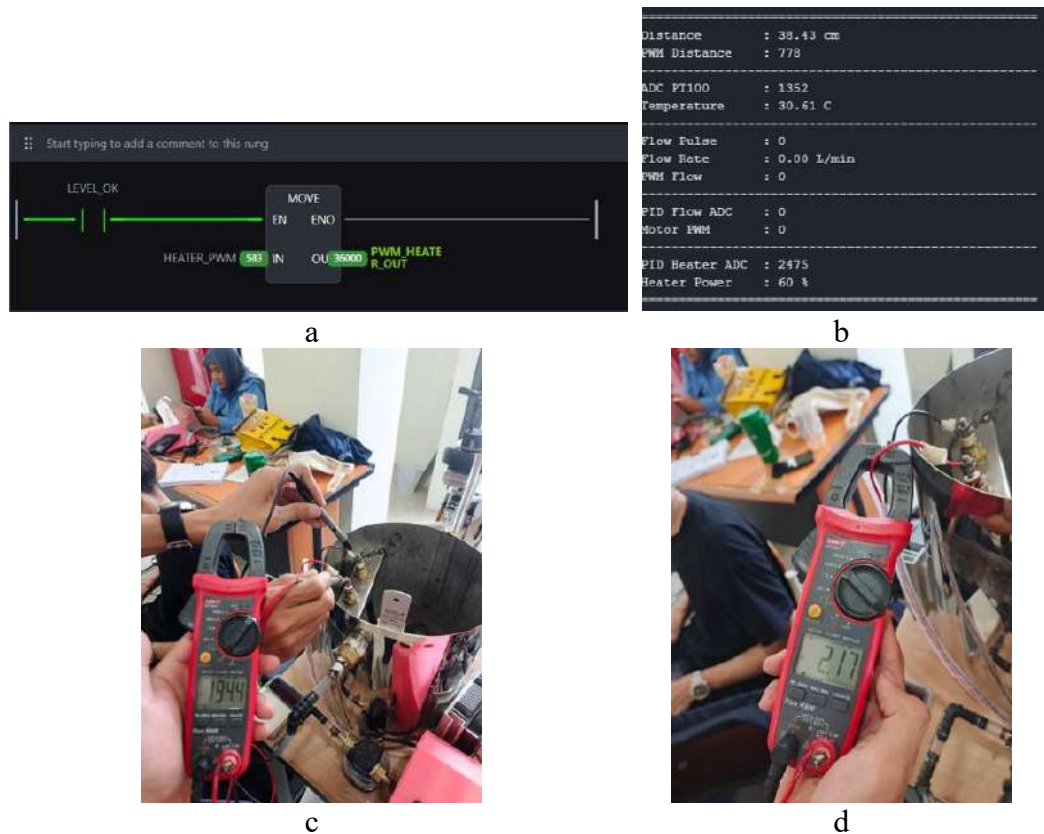
d

Gambar 4.21a.) Input *OpenPLC* b.) Keluar Arduino IDE 50% c.) Pengukuran Tegangan 50% d.) Pengukuran Arus 50%

Berdasarkan Gambar 4.21, pada *Heater Power* 50% tegangan keluaran yang terukur sebesar 171,9 Vac RMS dan arus *Heater* sebesar 1,99 A. Jika dibandingkan dengan kondisi 40%, tegangan meningkat sebesar 30,8 V dan arus meningkat sebesar 0,21 A. Nilai 50% pada sisi perintah tidak menghasilkan tegangan sebesar setengah dari tegangan sumber. Hal ini terjadi karena perintah 50% digunakan untuk menentukan sudut penyalaan TRIAC, bukan untuk membagi amplitudo tegangan sumber menjadi 50%. Hasil pengujian ini penting karena menunjukkan bahwa keluaran kendali PID yang nantinya diteruskan sebagai *Heater Power* harus dipahami sebagai level kendali *dimmer*, sedangkan tegangan dan arus aktual pada *Heater* tetap mengikuti karakteristik *phase angle control*. Dengan demikian, level 50% berfungsi sebagai titik pengamatan penting untuk menunjukkan perbedaan antara persentase perintah kontrol dan besaran listrik aktual yang diterima *Heater*.

4.3.6 Pengujian Daya *Heater* 60%

Pengujian *Heater Power* 60% dilakukan untuk melihat respons AC *Dimmer* ketika perintah kendali dinaikkan dari daerah menengah menuju daerah yang lebih tinggi. Pada kondisi ini, sistem tetap menggunakan konfigurasi perangkat dan beban yang sama agar hasil antar level dapat dibandingkan secara konsisten. Nilai *Heater Power* diatur melalui *OpenPLC*, kemudian dilakukan pengukuran tegangan AC pada sisi keluaran *dimmer* dan arus yang mengalir pada *Heater*. Pengujian ini juga digunakan untuk mengamati apakah perubahan keluaran mulai menunjukkan kecenderungan mendekati tegangan sumber ketika TRIAC menghantarkan bagian gelombang AC yang semakin besar.



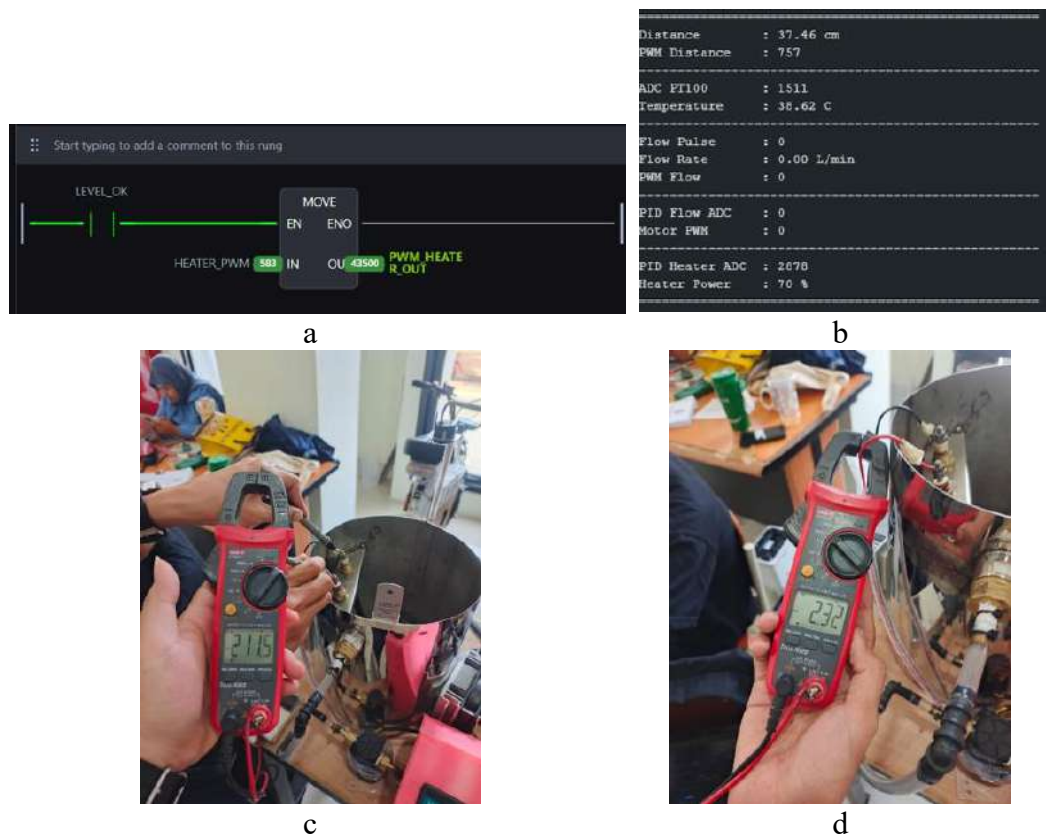
Gambar 4.22 a.) Input *OpenPLC* b.) Keluar Arduino IDE 60% c.) Pengukuran Tegangan 60% d.) Pengukuran Arus 60%

Berdasarkan Gambar 4.22, pada *Heater Power* 60% tegangan keluaran mencapai 194,4 Vac RMS dan arus *Heater* sebesar 2,17 A. Dibandingkan level 50%, tegangan bertambah 22,5 V dan arus bertambah 0,18 A. Pada kondisi ini tegangan keluaran sudah berada semakin dekat dengan daerah tegangan sumber sehingga tambahan perintah 10% tidak lagi menimbulkan kenaikan sebesar yang terlihat pada beberapa *level* sebelumnya. Hasil tersebut menunjukkan bahwa semakin besar *Heater Power*, semakin awal TRIAC menghantarkan gelombang AC dan semakin besar energi yang diteruskan ke *Heater*, tetapi sensitivitas perubahan Vac RMS mulai mengecil ketika level kendali memasuki daerah menengah ke tinggi. Kondisi ini menjadi transisi menuju daerah keluaran tinggi, yaitu saat perubahan perintah mulai menghasilkan kenaikan tegangan yang lebih kecil dibandingkan daerah rendah-menengah.

4.3.7 Pengujian Daya *Heater* 70%

Pengujian *Heater Power* 70% dilakukan ketika perintah kendali telah memasuki daerah keluaran tinggi. Tahap ini digunakan untuk mengetahui bagaimana AC *Dimmer* merespons kenaikan perintah setelah keluaran sebelumnya

mulai mendekati kondisi penghantaran yang lebih besar. Pengukuran tegangan dan arus dilakukan dengan metode yang sama seperti level sebelumnya agar perubahan yang terlihat tidak dipengaruhi oleh perbedaan percobaan. Data pada kondisi 70% kemudian dibandingkan dengan level 60% dan 80% untuk melihat apakah kenaikan *Heater Power* masih menghasilkan perubahan tegangan dan arus yang sebanding atau mulai menunjukkan karakteristik nonlinier pada daerah kendali tinggi.



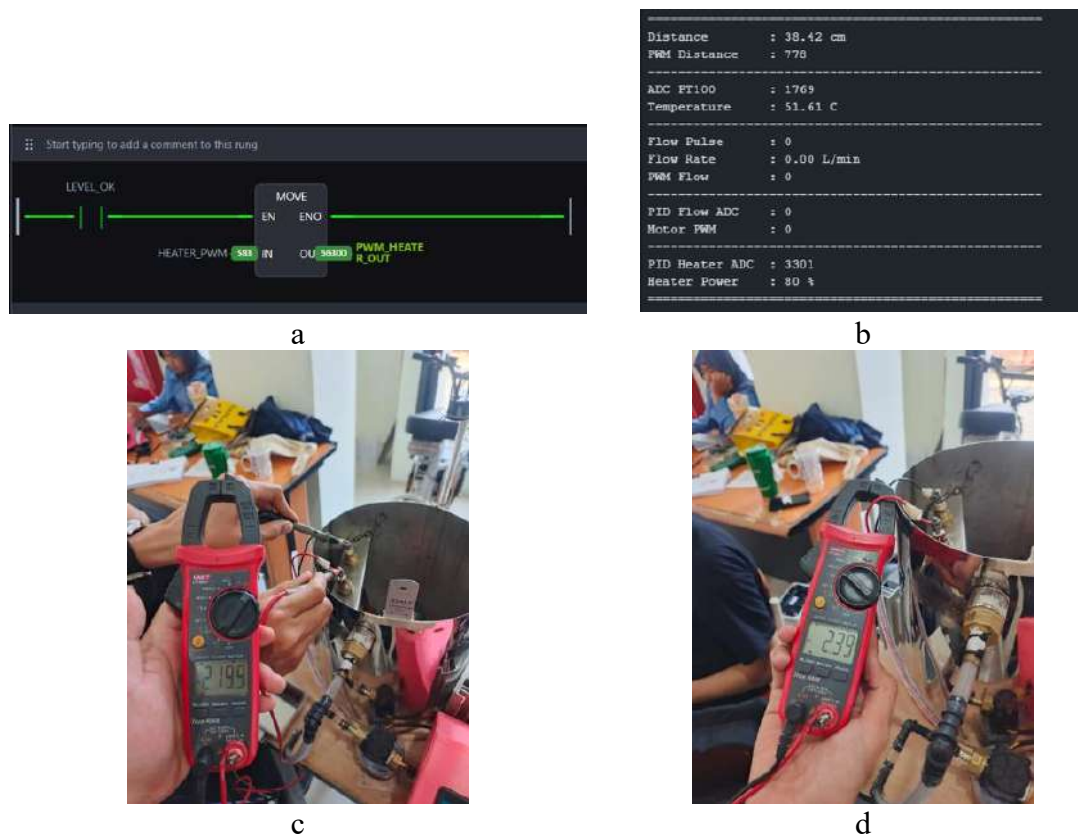
Gambar 4.23 a.) Input *OpenPLC* b.) Keluar *Arduino IDE* 70% c.) Pengukuran Tegangan 70% d.) Pengukuran Arus 70%

Berdasarkan Gambar 4.23, pada *Heater Power* 70% arus *Heater* yang terukur sebesar 2,32 A, sedangkan tegangan keluaran AC *Dimmer* sebesar 211,5 Vac RMS. Jika dibandingkan dengan kondisi 60%, tegangan meningkat dari 194,4 V menjadi 211,5 V atau bertambah 17,1 V, sedangkan arus meningkat dari 2,17 A menjadi 2,32 A atau bertambah 0,15 A. Hasil ini menunjukkan bahwa kenaikan *Heater Power* dari 60% menjadi 70% masih diikuti oleh peningkatan tegangan efektif dan arus yang diterima *Heater*. Pada level ini keluaran AC *Dimmer* sudah berada pada daerah penghantaran tinggi, sehingga perubahan selanjutnya tidak harus menghasilkan kenaikan Vac RMS dan arus yang linier terhadap persentase perintah.

Oleh karena itu, nilai 211,5 Vac RMS dan 2,32 A digunakan sebagai hasil pengukuran aktual pada kondisi *Heater Power* 70%.

4.3.8 Pengujian Daya *Heater* 80%

Pengujian *Heater Power* 80% dilakukan untuk mengamati kondisi AC *Dimmer* pada perintah kendali tinggi, ketika sebagian besar gelombang AC diperkirakan telah diteruskan menuju *Heater*. Perintah *Heater Power* tetap diberikan melalui *OpenPLC*, sedangkan keluaran aktual pada beban diperiksa melalui pengukuran tegangan AC dan arus. Pengujian dilakukan tanpa mengubah *Heater* maupun konfigurasi rangkaian sehingga hasilnya dapat dibandingkan langsung dengan level sebelumnya. Data yang diperoleh digunakan untuk melihat apakah keluaran listrik masih meningkat secara nyata atau mulai berada pada daerah yang mendekati batas tegangan sumber.



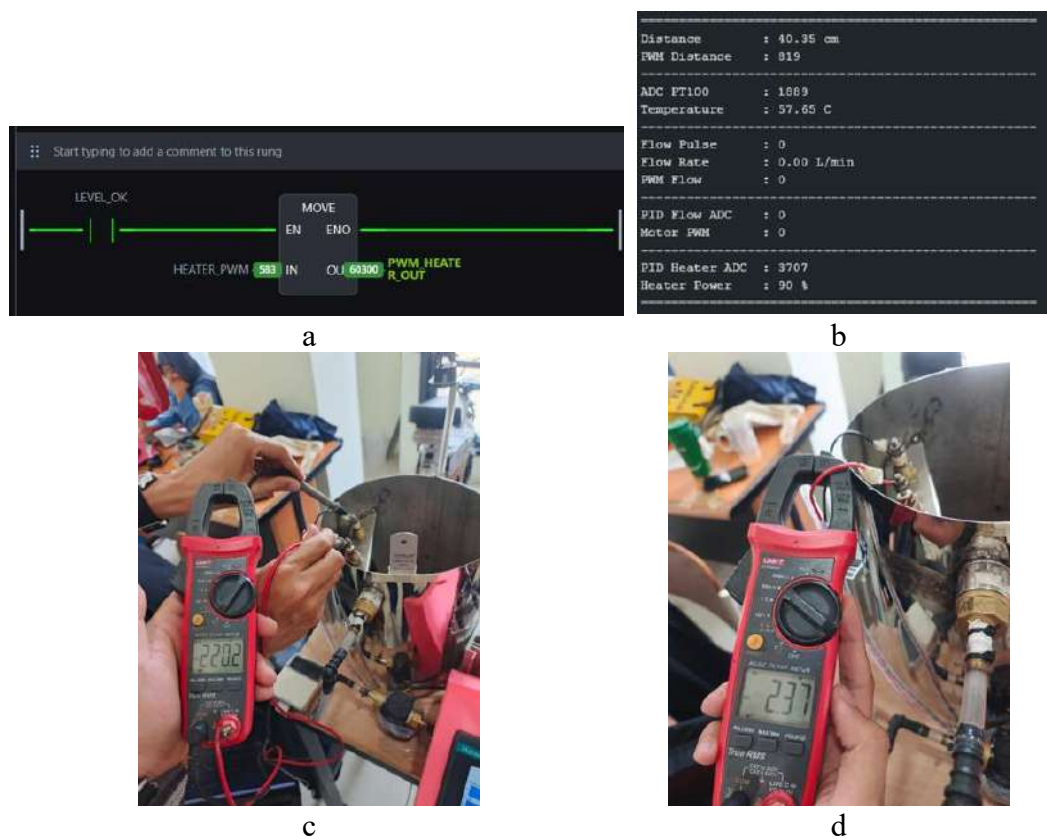
Gambar 4.24 a.) Input *OpenPLC* b.) Keluar Arduino IDE 80% c.) Pengukuran Tegangan 80% d.) Pengukuran Arus 80%

Berdasarkan Gambar 4.24, pada *Heater Power* 80% diperoleh tegangan sebesar 219,9 Vac RMS dan arus *Heater* sebesar 2,39 A. Tegangan yang terbaca sedikit lebih rendah dibandingkan hasil 70%, sedangkan arus meningkat menjadi 2,39 A. Kondisi ini kembali memperlihatkan bahwa pada daerah kendali tinggi,

hubungan antara *Heater Power*, tegangan terukur, dan arus tidak dapat diasumsikan linier. Secara fungsi, *AC Dimmer* sudah meneruskan sebagian besar gelombang AC menuju *Heater*, sehingga *Vac RMS* berada di sekitar tegangan sumber. Perbedaan kecil antarlevel lebih tepat dibaca sebagai variasi hasil pengukuran aktual daripada sebagai kegagalan perubahan perintah kendali. Hasil ini juga menunjukkan bahwa interpretasi level 80% harus tetap didasarkan pada pembacaan aktual, karena perintah kendali tidak sama dengan persentase langsung tegangan maupun arus.

4.3.9 Pengujian Daya Heater 90%

Pengujian *Heater Power* 90% dilakukan sebagai kondisi terakhir sebelum *AC Dimmer* diberikan perintah maksimum. Pada tahap ini, pengukuran digunakan untuk melihat karakteristik keluaran ketika *TRIAC* bekerja pada daerah penghantaran yang sangat tinggi. Tegangan keluaran dan arus *Heater* diukur menggunakan prosedur yang sama dengan level 10% sampai 80%, sehingga hasil dapat dibandingkan secara berurutan. Pengujian 90% juga menjadi acuan penting untuk menilai perubahan yang terjadi ketika sistem selanjutnya dinaikkan ke kondisi 100%.



Gambar 4.25 a.) Input *OpenPLC* b.) Keluar *Arduino IDE* 90% c.) Pengukuran Tegangan 90% d.) Pengukuran Arus 90%

Berdasarkan Gambar 4.25, pada Heater Power 90% tegangan keluaran tercatat sebesar 220,2 Vac RMS dan arus *Heater* sebesar 2,37 A. Dibandingkan level 80%, tegangan hanya berubah sebesar 0,3 V dan arus berubah sebesar 0,02 A. Perubahan yang sangat kecil tersebut menunjukkan bahwa pada daerah 80–90% keluaran dimmer sudah berada pada daerah penghantaran tinggi. TRIAC menghantarkan sebagian besar gelombang AC sehingga peningkatan perintah berikutnya tidak memberikan perubahan Vac RMS sebesar pada daerah kendali rendah dan menengah. Dengan demikian, level 90% dapat dipahami sebagai kondisi ketika keluaran listrik menuju *Heater* sudah mendekati batas yang diberikan oleh sumber dan karakteristik beban. Level 90% kemudian digunakan sebagai pembanding langsung terhadap kondisi 100% untuk melihat apakah perintah maksimum masih menghasilkan perubahan keluaran yang konsisten.

4.3.10 Pengujian Daya *Heater* 100%

Pengujian *Heater Power* 100% dilakukan untuk mengetahui kondisi keluaran AC *Dimmer* ketika *OpenPLC* memberikan perintah kendali maksimum. Pada level ini, tegangan keluaran dan arus *Heater* tetap diukur pada titik yang sama seperti pengujian sebelumnya. Pengukuran dilakukan lebih dari satu kali agar kondisi maksimum tidak hanya diwakili oleh satu pembacaan sesaat, terutama karena pembacaan alat ukur dapat berubah selama pengamatan. Dengan cara tersebut, data yang dicatat dapat menunjukkan rentang kondisi aktual yang terjadi pada keluaran *dimmer* ketika perintah sistem berada pada 100%. Hasil beberapa kali pembacaan kemudian dibahas setelah dokumentasi pengujian ditampilkan.



a

```
pidADC = 0 | motorPWM = 0

-----
Distance      : 24.51 cm
PWM Distance  : 532
-----
ADC PT100    : 2065
Temperature   : 66.52 C
-----
Flow Pulse    : 0
Flow Rate    : 0.00 L/min
PWM Flow      : 0
-----
PID Flow ADC  : 0
Motor PWM     : 0
-----
PID Heater ADC : 4090
Heater Power  : 100 %
```

b



c



d



e



f

Gambar 4.26 a.) Input *OpenPLC* b.) Keluar Arduino IDE 100% c.) Pengukuran Tegangan 100% d.) Pengukuran Tegangan 100% e.) Pengukuran Arus 100% f.) Pengukuran Arus 100%

Berdasarkan Gambar 4.26, Gambar 4.27, dan Gambar 4.28, *Heater Power* 100% merupakan kondisi ketika *OpenPLC* memberikan perintah maksimum ke AC Light Dimmer. Berbeda dengan pengujian 10–90% yang dicatat menggunakan satu pembacaan utama, pada kondisi 100% tegangan dan arus diamati beberapa kali karena nilai pada alat ukur berubah selama pengukuran. Tegangan AC yang terbaca adalah 183,1 V, 191,8 V, dan 206,6 V, sedangkan arus Heater berada pada rentang 2,07–2,20 A. Data tegangan dan arus tersebut tidak dipasangkan menjadi satu titik tetap karena diperoleh sebagai variasi pembacaan selama kondisi perintah 100%. Hasil ini menunjukkan bahwa perintah maksimum pada sisi kontrol tidak selalu menghasilkan satu nilai Vac RMS yang konstan pada alat ukur. Oleh karena itu, kondisi 100% dilaporkan sebagai rentang/variasi hasil aktual, bukan diasumsikan sebagai 220 V tetap ataupun sebagai nilai arus tertentu. Variasi tersebut perlu dipertimbangkan ketika keluaran dimmer digunakan sebagai input *Heater* pada

sistem pengendalian temperatur. Dengan demikian, evaluasi kondisi 100% lebih tepat dinyatakan sebagai rentang hasil pengukuran daripada satu angka tunggal, karena tujuan pengujian adalah mencatat kondisi listrik aktual yang diterima *Heater* pada saat perintah maksimum diberikan.

4.3.11 Analisis Perbandingan Tegangan dan Arus Keluaran AC *Dimmer*

Setelah seluruh *level Heater Power* 10% sampai 100% diuji, hasil tegangan dan arus dirangkum untuk melihat kecenderungan keluaran AC *Dimmer* secara keseluruhan. Perbandingan ini digunakan untuk mengetahui daerah keluaran rendah, menengah, dan tinggi, serta untuk melihat apakah perubahan perintah kendali menghasilkan perubahan listrik yang konsisten menuju *Heater*. Data 10–90% ditampilkan menggunakan pembacaan utama pada masing-masing kondisi, sedangkan kondisi 100% tetap ditampilkan sebagai beberapa nilai tegangan dan rentang arus karena pembacaannya berubah selama pengamatan.

Tabel 4.2 Hasil Pengukuran daya *Heater*

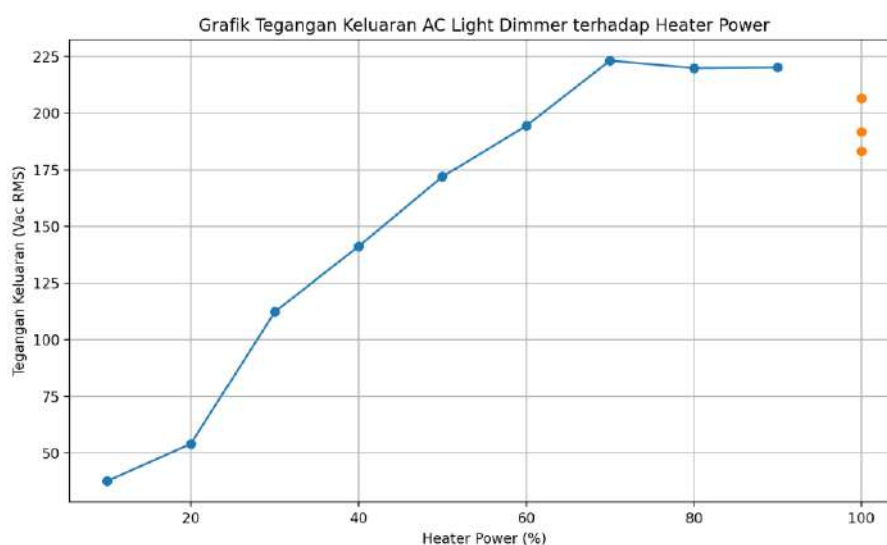
Heater Power (%)	Tegangan Vac RMS (V)	Arus Heater (A)	Keterangan
10	37,6	0,56	Pembacaan aktual
20	54,05	1,0	Pembacaan aktual
30	112,4	1,48	Pembacaan aktual
40	141,1	1,78	Pembacaan aktual
50	171,9	1,99	Pembacaan aktual
60	194,4	2,17	Pembacaan aktual
70	211,5	2,32	Pembacaan aktual
80	219,9	2,39	Pembacaan aktual
90	220,2	2,37	Pembacaan aktual
			Beberapa
100	183,1. 191,8. 206,6	2,07–2,20	pembacaan, nilai berubah-ubah

Berdasarkan Tabel 4.2, karakteristik keluaran AC *Dimmer* dapat dibagi menjadi beberapa daerah pengamatan. Pada *Heater Power* 10–60%, kenaikan nilai perintah secara umum diikuti oleh kenaikan Vac RMS dari 37,6 V menjadi 194,4 V dan kenaikan arus dari 0,56 A menjadi 2,17 A. Pada *Heater Power* 70–90%,

tegangan berada pada kisaran 211,5–220,2 V dan arus berada pada kisaran 2,32–2,39 A. Kondisi tersebut menunjukkan bahwa keluaran sudah memasuki daerah penghantaran tinggi sehingga perubahan perintah tidak lagi menghasilkan kenaikan tegangan dan arus yang besar secara linier. Khusus pada kondisi 100%, tegangan berubah pada 183,1 V, 191,8 V, dan 206,6 V, sedangkan arus berada pada 2,07–2,20 A, sehingga data ditampilkan sebagai beberapa pembacaan dan bukan satu pasangan nilai tetap.

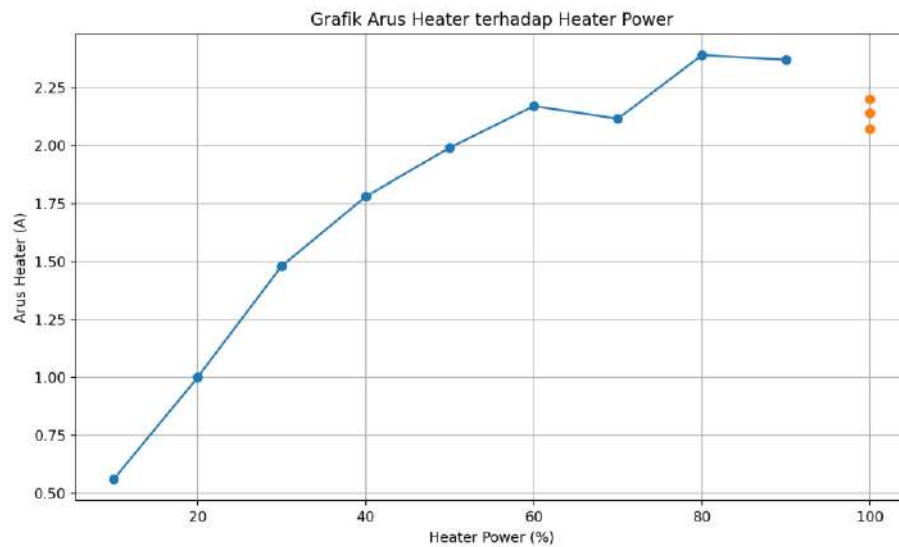
Pada rentang 70% sampai 90%, Vac RMS berada pada kisaran 211,5–220,2 V, sedangkan arus berada pada kisaran 2,32–2,39 A. Perubahan pada daerah ini relatif kecil dibandingkan daerah kendali rendah hingga menengah dan tidak membentuk hubungan linier terhadap kenaikan persentase perintah. Kondisi tersebut sesuai dengan karakteristik phase angle control, yaitu persentase perintah berhubungan dengan sudut penyalan TRIAC dan bukan langsung dengan persentase tegangan RMS maupun arus.

Pada kondisi 100%, hasil pengukuran berbeda dengan 90% karena pembacaan tegangan dan arus berubah-ubah selama pengamatan. Tegangan tercatat sebesar 183,1 V, 191,8 V, dan 206,6 V, sedangkan arus berada pada rentang 2,07–2,20 A. Oleh sebab itu, data 100% diperlakukan sebagai variasi hasil pengukuran aktual dan tidak digunakan sebagai satu pasangan tegangan-arus tetap. Perbedaan tersebut perlu dicatat sebagai karakteristik hasil pengujian nyata pada sistem dan tidak boleh ditukar antara besaran arus dan tegangan.



Gambar 4.27 Grafik Tegangan Percobaan Daya Heater

Berdasarkan Gambar 4.27, kurva tegangan menunjukkan kecenderungan meningkat pada daerah Heater Power rendah sampai menengah, kemudian mulai berada di sekitar tegangan sumber pada daerah kendali tinggi. Pada Heater Power 70% tegangan yang digunakan adalah 211,5 Vac RMS, kemudian 219,9 Vac RMS pada 80% dan 220,2 Vac RMS pada 90%. Bentuk kurva tersebut menegaskan bahwa nilai Heater Power bukan persentase langsung dari Vac RMS. Pada kondisi 100%, tiga pembacaan tegangan ditampilkan secara terpisah, yaitu 183,1 V, 191,8 V, dan 206,6 V, karena nilai berubah selama pengamatan.



Gambar 4.28 Grafik Arus Percobaan Daya Heater

Berdasarkan Gambar 4.28, arus *Heater* secara umum meningkat dari 0,56 A pada Heater Power 10% hingga berada di kisaran lebih dari 2 A pada daerah kendali tinggi. Setelah 60%, perubahan arus tidak lagi menunjukkan pola kenaikan yang tetap; pada 70% arus terbaca 2,32 A, pada 80% 2,39 A, dan pada 90% 2,37 A. Pada kondisi 100%, arus berubah pada rentang 2,07–2,20 A. Hasil tersebut menunjukkan bahwa hubungan perintah dimmer terhadap arus aktual tidak dapat dianggap linier. Secara keseluruhan, perubahan *Heater Power* dari *OpenPLC* dapat mengubah kondisi listrik yang diterima *Heater*, sedangkan besar tegangan dan arus aktual tetap ditentukan oleh karakteristik AC *Dimmer*, sumber, dan beban *Heater*.

4.4 Pengujian Sistem Pendingin *Peltier*

Pengujian sistem pendingin *peltier* dilakukan untuk melihat bagaimana *peltier* bekerja sebagai bagian penstabil temperatur pada sistem. Pada penelitian ini, *peltier* tidak dipakai sebagai pendingin utama yang bekerja terus-menerus, tetapi

digunakan sebagai elemen bantu untuk menjaga kestabilan temperatur air. Dari hasil perancangan sistem, *Peltier* dibagi menjadi dua bagian fungsi, yaitu *Peltier* penampungan dan *peltier final*. *Peltier* penampungan ditempatkan pada tangki penampungan, sedangkan *peltier final* digunakan pada tahap akhir sebagai penstabil temperatur ketika terjadi kenaikan temperatur melebihi batas yang diinginkan.

Secara umum, logika kerja *peltier* pada sistem ini dibuat agar pendinginan hanya aktif pada kondisi tertentu. Artinya, *peltier* tidak langsung menyala setiap saat, melainkan menunggu kondisi sistem terpenuhi lebih dulu. Dengan cara ini, kerja pendinginan menjadi lebih terarah, sehingga *Peltier* hanya aktif ketika memang dibutuhkan, terutama saat temperatur air sudah mendekati atau bahkan melewati *setpoint*. Dari pengujian ini, yang diamati bukan hanya kondisi ON dan OFF saja, tetapi juga bagaimana urutan logikanya bekerja dari pembacaan temperatur, validasi kondisi sistem, sampai keluaran akhir yang membuat *peltier* aktif.

4.4.1 Pengujian *Peltier* Penampungan

Pada bagian ini dilakukan pengujian terhadap *peltier* penampungan. *Peltier* penampungan berfungsi untuk membantu menjaga kondisi air pada tangki penampungan agar temperatur air yang akan masuk ke proses berikutnya tidak terlalu jauh dari kondisi yang diinginkan. Jadi, fungsi utamanya bukan untuk mendinginkan secara ekstrem, tetapi lebih ke arah membantu menstabilkan kondisi awal air sebelum masuk ke bagian proses berikutnya.

Pengujian dilakukan dengan melihat dua kondisi utama, yaitu saat *Peltier* penampungan OFF dan saat *Peltier* penampungan ON. Kedua kondisi ini penting untuk menunjukkan bahwa *peltier* benar-benar bekerja sesuai logika sistem. Dari sini bisa dilihat kapan *peltier* belum diperlukan, dan kapan *Peltier* mulai diaktifkan karena syarat sistem sudah terpenuhi.



Gambar 4.29 Kondisi Ladder dan *Peltier* penampungan OFF



Gambar 4.30 Kondisi Ladder dan *Peltier* Penampungan ON

Berdasarkan Gambar 4.29, terlihat kondisi saat *Peltier* penampungan OFF. Pada kondisi ini, sistem belum memberikan perintah pendinginan ke *Peltier* penampungan, sehingga keluaran masih dalam keadaan tidak aktif. Hal ini menunjukkan bahwa meskipun sistem sudah membaca beberapa kondisi dasar, pendinginan belum dijalankan karena belum semua syarat terpenuhi atau karena kondisi temperatur air masih belum memerlukan bantuan pendinginan dari *peltier* penampungan.

Selanjutnya, pada Gambar 4.30 terlihat kondisi saat *peltier* penampungan ON. Pada kondisi ini, logika sistem sudah memenuhi syarat untuk mengaktifkan *Peltier* penampungan. Secara alur kerja, kondisi ini terjadi ketika sistem utama sudah aktif, level air berada pada kondisi aman atau sesuai, dan proses berjalan dalam kondisi yang mengizinkan *peltier* membantu menstabilkan temperatur. Jadi, *peltier* penampungan bekerja bukan secara acak, tetapi benar-benar menunggu kondisi sistem siap terlebih dahulu.

Jika temperatur air pada tangki penampungan terlalu tinggi, maka temperatur awal air yang masuk ke sistem juga akan tinggi. Kondisi tersebut bisa membuat proses pengendalian temperatur menjadi lebih berat. Karena itu, *peltier* penampungan dipakai untuk membantu menjaga agar temperatur air penampungan tetap lebih stabil sebelum air digunakan pada percobaan berikutnya.

4.4.2 Pengujian *Peltier Final*

Berbeda dengan *peltier* penampungan, *peltier final* bekerja pada tahap akhir sebagai pendingin penstabil ketika temperatur air sudah melewati batas yang ditentukan. Jadi, fungsi *peltier final* lebih spesifik, yaitu untuk membantu menurunkan temperatur ketika terjadi *overshoot* atau ketika temperatur aktual lebih tinggi dari *setpoint* yang telah ditentukan.

Pengujian pada bagian ini dilakukan dengan melihat alur logika kerja *peltier final* dari awal sampai keluaran akhir. Yang ingin ditunjukkan bukan hanya kondisi ON atau OFF saja, tetapi bagaimana sistem memutuskan bahwa pendinginan akhir memang perlu diaktifkan. Dari gambar-gambar hasil pengujian, terlihat bahwa keputusan mengaktifkan *peltier final* dilakukan secara bertahap mulai dari validasi *setpoint*, pembentukan batas temperatur, perbandingan temperatur aktual, hingga keluaran akhir.



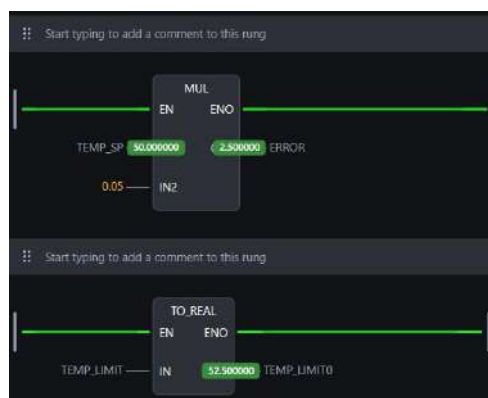
Gambar 4.31 Kondisi Ladder Validasi *setpoint*



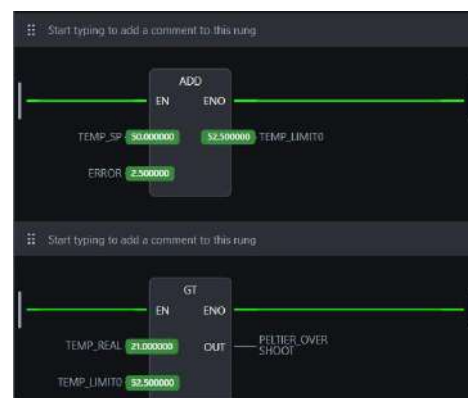
Gambar 4.32 Kondisi Ladder Validasi dan Perhitungan *Error*

Berdasarkan Gambar 4.31, terlihat bahwa sistem terlebih dahulu melakukan validasi terhadap *setpoint* yang digunakan. Pada bagian ini, sistem memeriksa apakah *setpoint* yang dibutuhkan untuk proses sudah terisi dengan benar dan sudah siap digunakan. Tahap ini penting, karena *peltier final* tidak mungkin bekerja dengan benar jika data dasar seperti *setpoint* temperatur belum valid.

Kemudian pada Gambar 4.32, sistem mulai masuk ke tahap pembacaan temperatur dan pembentukan logika *error*. Pada bagian ini, temperatur aktual yang terbaca dibandingkan dengan *setpoint* temperatur yang telah ditentukan. Dari sinilah nanti sistem dapat mengetahui apakah temperatur masih aman, sudah mendekati batas, atau justru sudah melewati batas yang membuat pendinginan perlu diaktifkan.



Gambar 4.33 Kondisi Ladder Perhitungan *Error*



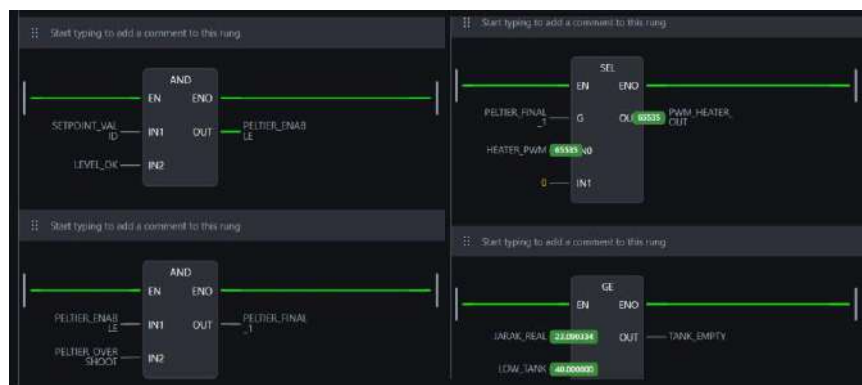
Gambar 4.34 Kondisi Ladder Perhitungan *Error*

Berdasarkan Gambar 4.33, terlihat bahwa sistem membentuk batas temperatur tambahan dari *setpoint* yang telah ditentukan. Dari logika yang digunakan, sistem tidak langsung menjadikan *setpoint* sebagai batas nyala *peltier final*, tetapi

menambahkan toleransi tertentu. Artinya, *peltier final* baru akan aktif ketika temperatur aktual benar-benar melewati batas atas yang sudah ditentukan, bukan saat temperatur baru menyentuh *setpoint*.

Pada Gambar 4.34, terlihat proses perbandingan antara temperatur aktual dengan batas temperatur tersebut. Jika temperatur aktual lebih tinggi dari batas itu, maka sistem menganggap telah terjadi kondisi *Overshoot*. Dari sinilah muncul keputusan bahwa *Peltier final* perlu diaktifkan. Jadi, secara logika, *Peltier final* berfungsi sebagai penstabil akhir yang hanya bekerja saat memang ada indikasi temperatur melewati batas.

Cara kerja seperti ini cukup penting, karena jika *peltier* langsung aktif tepat saat temperatur mencapai *setpoint*, maka sistem bisa terlalu sensitif dan pendinginan bisa bekerja terlalu cepat. Akibatnya, temperatur justru bisa turun terlalu jauh. Dengan memberi batas toleransi, sistem jadi lebih masuk akal: pendingin akhir baru bekerja saat benar-benar dibutuhkan.



Gambar 4.35 Kondisi Ladder Validasi *Peltier* Tangki *Final* ON/OFF

Berdasarkan Gambar 4.35, setelah kondisi valid dan *Overshoot* terdeteksi, sistem membentuk kondisi *enable* untuk *peltier final*. Tahap ini bisa dipahami sebagai tahap izin kerja. Jadi meskipun temperatur tinggi, *Peltier final* tetap harus menunggu syarat sistem lain terpenuhi, seperti kondisi level air dan validasi proses secara keseluruhan.

Selanjutnya, pada Gambar 4.35, terlihat keluaran akhir yang menunjukkan bahwa *peltier final* sudah siap bekerja saat seluruh syarat sudah terpenuhi. Artinya, pendinginan akhir tidak aktif hanya karena satu kondisi saja, tetapi benar-benar merupakan hasil keputusan logika dari beberapa tahapan sebelumnya. Dari alur ini

bisa dipahami bahwa sistem pendinginan *peltier final* bekerja secara bertahap dan lebih terkontrol.

Dari keseluruhan logika tersebut, dapat dijelaskan bahwa *Peltier final* pada sistem ini memang dirancang untuk menangani kondisi *Overshoot*. Jadi ketika temperatur air hasil pengaturan temperatur masih naik dan melewati *setpoint* yang ditentukan, *peltier final* akan membantu menurunkan temperatur agar kembali mendekati nilai yang diinginkan. Ketika temperatur sudah turun kembali dan kondisi *overshoot* hilang, maka *peltier final* juga akan berhenti bekerja. Dengan kata lain, *peltier final* berperan sebagai penyeimbang akhir temperatur.

4.4.3 Pengujian Arus *Peltier* Saat ON

Selain mengamati logika kerja pada program, pengujian juga dilakukan secara langsung pada sisi perangkat untuk memastikan bahwa saat logika menyatakan *Peltier* aktif, perangkat memang benar-benar bekerja. Salah satu cara yang dilakukan adalah dengan mengukur arus *peltier* saat kondisi ON.

Pengukuran ini penting karena dari sisi program saja belum cukup. Meskipun pada logika sistem *peltier* sudah dinyatakan aktif, perlu dibuktikan juga bahwa arus benar-benar mengalir ke modul *peltier*. Dengan begitu, hasil pengujian tidak hanya berhenti pada tampilan program, tetapi juga terbukti secara fisik pada perangkat.



Gambar 4.36 Pengukuran Arus *Peltier*

Berdasarkan Gambar 4.36, terlihat bahwa saat *Peltier* dalam kondisi aktif, arus yang terukur sebesar 3,17 A. Nilai ini menunjukkan bahwa pada saat logika sistem mengaktifkan *peltier final*, arus benar-benar mengalir ke modul pendingin. Dengan adanya arus tersebut, dapat dipastikan bahwa *peltier* memang bekerja secara aktual, bukan hanya aktif pada tampilan program saja.

Hasil arus ini juga menunjukkan bahwa *peltier* termasuk beban yang cukup besar dibanding sinyal logika kontrolnya. Karena itu, keberadaan rangkaian driver atau jalur catu daya yang sesuai menjadi penting agar modul *Peltier* dapat bekerja dengan baik. Jika arus yang mengalir terlalu kecil atau tidak stabil, maka efek pendinginan yang dihasilkan juga tidak akan optimal.

Dari sisi pengujian, nilai arus sebesar 3,17 A dapat dijadikan bukti bahwa sistem kontrol benar-benar mampu mengaktifkan beban *peltier*. Jadi hubungan antara logika kontrol dan kerja aktual perangkat sudah terlihat: saat kondisi *overshoot* terdeteksi dan logika mengizinkan pendinginan, *peltier* aktif dan arus mengalir pada modul.

4.5 Implementasi Sistem Kontrol

Implementasi sistem kontrol dilakukan untuk mengetahui kinerja sistem yang telah dirancang pada proses pengendalian temperatur air. Sistem kontrol menggunakan *OpenPLC* yang dijalankan pada ESP32 sebagai pengendali utama. Program kontrol dibuat mmenggubakan *Ladder Diagram* untuk mengatur proses kerja sistem serta menjalankan pengendalian temperatur meggunakan metode PID.

Temperatur air dibaca menggunakan PT100 sebagai *Process Variable* (PV). Pada pengujian *closed loop*, nilai tersebut dibandingkan dengan *setpoint* sehingga *Error* dihitung sebagai selisih *setpoint* dan PV mengacu pada Persamaan (2.10). *Error* kemudian diproses oleh Function Block CTRL_PID; secara teoritis aksi PID mengacu pada Persamaan (2.11), sedangkan implementasinya dilakukan secara internal oleh *OpenPLC* untuk menentukan keluaran *heater*.

Heater digunakan sebagai aktuator utama untuk mengatur temperatur air, sedangkan *peltier* digunakan sebagai aktuator pendingin untuk membantu menstabilkan temperatur jika melebihi kondisi yang diinginkan. Sebelum sistem diuji menggunakan PID secara *closed loop*, dilakukan pengujian *open loop* terlebih dahulu untuk mengetahui karkteristik dasar dari *plant* pemanas.

Pada sistem yang dirnacang, ESP32 yang menjalankan *OpenPLC* berfungsi sebagai pengendali utama, sedangkan ESP32 satunya sebagai antar mula untuk pembacaan sensor dan pengendali aktuator. Pertukaran data antar kedua ESP32 dilakukan melalui komunikasi *pin to pin* berbasis PWM.

4.6 Pengujian Karakteristik Sistem Pemanas

Pengujian karakteristik sistem pemanas dilakukan untuk mengetahui respon temperatur air terhadap *input heater*. Pengujian dilakukan secara *open loop*, yaitu *Heater* diberikan *input* tetap tanpa adanya koreksi berdasarkan temperatur aktual yang terbaca oleh sensor. Pengujian *Open Loop* diperlukan untuk mengetahui karakteristik alami *plant* sebelum diterapkan pengendalian PID. Data hasil pengujian selanjutnya digunakan untuk memperoleh parameter karakteristik *plant* berupa *gain*, *dead time* dan *time constant* yang diperlukan dalam proses tuning PID menggunakan metode *Ziegler Nichols*.

Pada pengujian ini digunakan volume awal air sebesar 4 Liter. *Heater* yang digunakan memiliki daya maksimum 500watt dan diberikan *input* sebesar 69%. Temperatur air dibaca menggunakan PT100.

4.7 Pengujian Sistem Kontrol *Open Loop*

Pengujian sistem kontrol *open loop* dilakukan untuk mengetahui karakteristik awal sistem pemanasan sebelum diterapkan pengendalian PID. Pada pengujian ini, *heater* diberikan masukan secara tetap tanpa adanya proses koreksi berdasarkan nilai temperatur yang terbaca oleh sensor. Pengujian dilakukan dengan tujuan untuk mengetahui respon temperatur air terhadap pemberian daya *heater* sehingga dapat diperoleh karakteristik dari *plant* yang akan digunakan sebagai dasar dalam proses tuning PID.

Pada sistem yang dirancang, temperatur air pada tangki *final* dibaca menggunakan sensor PT100. Nilai temperatur tersebut nantinya digunakan sebagai *Process Variable* (PV) pada sistem pengendalian PID. *Heater* digunakan sebagai aktuator utama untuk menaikkan temperatur air, sedangkan *peltier* digunakan sebagai aktuator tambahan untuk membantu menstabilkan temperatur apabila terjadi *overshoot*. Sistem kontrol temperatur tersebut diimplementasikan menggunakan *OpenPLC* pada ESP32 sesuai dengan rancangan sistem yang telah dibuat.

Pengujian *open loop* dilakukan terlebih dahulu karena karakteristik respon *plant* perlu diketahui sebelum parameter PID ditentukan. Hasil pengujian berupa perubahan temperatur terhadap waktu selanjutnya digunakan untuk menentukan

parameter karakteristik sistem yang diperlukan dalam proses tuning menggunakan metode *Ziegler Nichols*.

4.7.1 Pengujian *Open Loop* PWM 69% 4 Liter

Sebelum pengujian dimulai, kondisi awal air pada tangki *final* terlebih dahulu ditentukan. Pada pengujian ini digunakan air sebanyak 4 liter dengan ketinggian permukaan air awal sebesar 10,8 cm.



Gambar 4.37 Aktual Air Pengujian

Berdasarkan Gambar 4.37, dapat dilihat bahwa kondisi awal air pada tangki *final* berada pada ketinggian sekitar 10,8 cm dengan volume air sebesar 4 liter. Kondisi awal tersebut digunakan sebagai acuan untuk mengetahui perubahan kondisi air selama proses pemanasan berlangsung. Pada awal pengujian, temperatur air yang terbaca oleh sensor PT100 adalah sekitar 23,06 °C.

Penentuan kondisi awal diperlukan agar perubahan temperatur yang terjadi selama pengujian dapat dianalisis berdasarkan kondisi awal yang telah diketahui. Selain itu, ketinggian air awal juga dicatat karena selama proses pemanasan terdapat kemungkinan perubahan volume air akibat penguapan.

Pengujian dilakukan tanpa memberikan *setpoint* temperatur karena tujuan pengujian ini bukan untuk melihat kemampuan sistem dalam mengikuti *setpoint*, melainkan untuk mengetahui karakteristik respon temperatur terhadap masukan *Heater*. Oleh karena itu, *Heater* tetap diberikan masukan sebesar 69% selama proses pengujian berlangsung. Data temperatur dicatat setiap satu menit menggunakan pembacaan sensor PT100. Pencatatan dilakukan mulai dari kondisi

awal hingga temperatur mengalami perubahan yang relatif kecil dan mendekati kondisi steady state. Kondisi pengujian dapat dilihat pada Tabel 4.3.

Tabel 4.3 Kondisi Pengujian

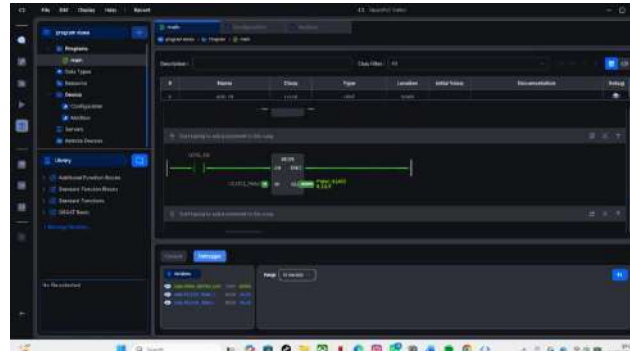
Parameter	Nilai
Volume Air	4 Liter
Daya <i>Heater</i> Maksimum	500 Watt
<i>Input Heater</i>	69%
Temperatur Awal	23,06 Celcius
Motode Pengujian	<i>Open Loop</i>
Interval Pengujian	1 Menit
Sensor Temperatur	PT100
<i>Mixing</i>	Tidak Ada

Kondisi pengujian pada Tabel 4.3 menunjukkan bahwa pengujian dilakukan menggunakan volume air sebesar 4 liter dengan *Heater* yang memiliki daya maksimum 500 W. *Heater* diberikan masukan sebesar 69% dan nilai tersebut dijaga tetap selama pengujian berlangsung. Temperatur awal air yang digunakan adalah 23,06°C, kemudian perubahan temperatur diamati menggunakan sensor PT100 dengan interval pencatatan setiap satu menit.

Pengujian dilakukan pada kondisi tangki terbuka dan tanpa menggunakan pengaduk. Kondisi tersebut merupakan kondisi aktual pada saat pengujian sehingga panas yang dihasilkan *Heater* tidak hanya digunakan untuk menaikkan temperatur air, tetapi sebagian juga dilepaskan ke lingkungan. Tidak digunakannya pengaduk juga menyebabkan distribusi panas di dalam air berlangsung secara alami, sehingga temperatur pada bagian yang berbeda di dalam tangki dapat mengalami perbedaan selama proses pemanasan. Kondisi ini menjadi salah satu faktor yang perlu diperhatikan ketika menganalisis respon temperatur yang diperoleh.

Pada pengujian sistem *Open Loop*, nilai masukan *Heater* terlebih dahulu diberikan melalui program Ladder pada *OpenPLC*. Pada sisi *OpenPLC*, nilai keluaran *Heater* ditampilkan sebesar 49.500. Nilai tersebut merupakan nilai keluaran PWM yang digunakan untuk merepresentasikan besar sinyal kendali *Heater*. Pada rancangan sistem, ESP32 yang menjalankan *OpenPLC* berfungsi sebagai pengendali utama, sedangkan ESP32 yang diprogram menggunakan

Arduino IDE berfungsi sebagai antarmuka perangkat keras untuk menerima keluaran tersebut dan mengatur *Heater* melalui modul *AC Dimmer RobotDyn*. Rancangan ini memang menggunakan komunikasi *pin to pin* berbasis PWM dari ESP32 *OpenPLC* menuju ESP32 Arduino.



Gambar 4.38 *Input PWM OpenPLC*

Berdasarkan Gambar 4.38, pengaturan *Heater* diawali dari program Ladder pada *OpenPLC*. Nilai keluaran yang diberikan untuk pengaturan *Heater* adalah sebesar 49.500. Nilai tersebut merupakan keluaran PWM dari ESP32 yang menjalankan *OpenPLC*. Keluaran PWM ini digunakan sebagai sinyal kendali untuk menentukan besar masukan yang diberikan kepada *Heater*.

Sinyal PWM dari *OpenPLC* kemudian diteruskan menuju ESP32 yang menjalankan program Arduino IDE melalui koneksi *pin to pin*. Karena keluaran dari *OpenPLC* berupa sinyal PWM, sinyal tersebut tidak langsung dibaca oleh Arduino sebagai nilai digital sebesar 49.500. Sinyal terlebih dahulu dilewatkan melalui rangkaian RC Low Pass Filter yang terdiri dari resistor dan kapasitor.

Rangkaian RC digunakan untuk mengubah sinyal PWM menjadi tegangan analog yang lebih stabil. Resistor berfungsi membatasi arus pada proses pengisian dan pengosongan kapasitor, sedangkan kapasitor berfungsi menyimpan muatan sehingga perubahan tegangan akibat pulsa PWM dapat diratakan. Dengan proses tersebut, keluaran PWM yang sebelumnya berupa pulsa berubah menjadi tegangan rata-rata yang kemudian dapat dibaca oleh ADC ESP32.

Pada tahap ini, nilai 49.500 pada *OpenPLC* tidak langsung menjadi nilai ADC 49.500 pada Arduino. Nilai tersebut telah melalui proses fisik berupa penyaringan PWM oleh rangkaian RC. Oleh karena itu, Arduino membaca tegangan hasil keluaran rangkaian RC, kemudian mengubahnya menjadi nilai ADC sesuai dengan kemampuan ADC ESP32.

```

// PID HEATER DARI OPENPLC
//-----
heaterADC=analogRead(PID_HEATER_PIN);
heaterADC=constrain(heaterADC,0,4095);

//-----
// FILTER ADC HEATER
//-----
heaterFiltered=(heaterFiltered*9+heaterADC)/10;

//-----
// KONVERSI KE PERSEN
//-----
heaterPower=map(heaterFiltered,0,4095,0,100);
heaterPower=constrain(heaterPower,0,100);

```

Gambar 4.39 Perhitungan penerimaan PWW *Heater* Arduino IDE

Pada Gambar 4.39 program Arduino, pembacaan tegangan tersebut dilakukan menggunakan fungsi *analogRead()* pada pin yang digunakan sebagai masukan *Heater*. ADC ESP32 menggunakan resolusi 12-bit sehingga nilai pembacaan berada pada rentang 0–4095.

Nilai ADC tersebut kemudian tidak langsung digunakan sebagai nilai akhir. Program Arduino melakukan proses penyaringan terhadap pembacaan ADC untuk mengurangi perubahan nilai yang terlalu cepat akibat fluktuasi tegangan dan noise. Proses penyaringan dilakukan dengan mempertahankan sebagian besar nilai pembacaan sebelumnya dan memasukkan sebagian nilai pembacaan terbaru. Dengan cara tersebut, perubahan nilai ADC menjadi lebih halus dan tidak langsung menyebabkan perubahan besar pada pengaturan *Heater*.

Setelah proses penyaringan selesai, nilai ADC kemudian dikonversikan oleh program menjadi nilai persentase dengan rentang 0–100%. Hasil pengolahan tersebut ditampilkan pada Serial Monitor sebagai *Heater Power* sebesar 69%. Nilai persentase inilah yang kemudian digunakan oleh program Arduino sebagai acuan pengaturan daya pada modul *AC Dimmer* RobotDyn. Pada Gambar 4.39 program Arduino, pembacaan ADC *Heater* terlebih dahulu dibatasi pada rentang 0–4095. Setelah itu, nilai pembacaan dilakukan penyaringan menggunakan metode pembobotan. Proses tersebut dilakukan agar perubahan kecil atau noise pada pembacaan ADC tidak langsung menyebabkan perubahan besar pada nilai pengaturan *Heater*. Pada program yang digunakan, proses penyaringan dilakukan dengan memberikan bobot sembilan bagian terhadap nilai sebelumnya dan satu bagian terhadap nilai ADC terbaru.

```

=====
Distance      : 29.34 cm
PWM Distance   : 593
=====
ADC PT100     : 1202
Temperature    : 23.06 C
=====
Flow Pulse    : 0
Flow Rate     : 0.00 L/min
PWM Flow      : 0
=====
PID Flow ADC   : 0
Motor PWM      : 0
=====
PID Heater ADC : 2883
Heater Power   : 69 %
=====

```

Gambar 4.40 Tampilan Pembacaan PWM *Heater* Pada Arduino IDE

Berdasarkan Gambar 4.40 tersebut, Dengan demikian, hasil pengujian menunjukkan bahwa nilai keluaran 49.500 pada *OpenPLC* mengalami beberapa tahapan pengolahan sebelum menjadi nilai pengaturan *Heater* pada Arduino. Tahapan tersebut dimulai dari keluaran PWM *OpenPLC*, kemudian melewati rangkaian RC Low Pass Filter untuk menghasilkan tegangan analog, dibaca oleh ADC ESP32 pada rentang 0–4095, dilakukan penyaringan secara perangkat lunak, dan akhirnya dikonversikan menjadi persentase daya *Heater*. Pada kondisi pengujian yang dilakukan, hasil pembacaan ADC berada pada 2883 dan hasil pengolahan program ditampilkan sebagai *Heater Power* 69%.

Perbedaan nilai antara 49.500 pada *OpenPLC* dan 2883 pada Arduino IDE tidak menunjukkan adanya kesalahan pembacaan, karena kedua nilai tersebut merupakan nilai pada tahapan sistem yang berbeda. Nilai 49.500 merupakan keluaran PWM pada sisi *OpenPLC*, sedangkan nilai 2883 merupakan hasil pembacaan ADC terhadap tegangan yang telah melewati rangkaian RC. Dengan demikian, nilai yang ditampilkan pada Arduino merupakan hasil dari proses pengolahan sinyal secara keseluruhan, bukan hasil pemindahan langsung nilai PWM dari *OpenPLC*.

Pada kondisi tersebut, temperatur awal air yang terbaca oleh sensor PT100 adalah 23,06 °C. Nilai ini menjadi kondisi awal sebelum proses pemanasan berlangsung. Setelah *Heater* bekerja dengan pengaturan 69%, temperatur air kemudian mengalami peningkatan secara bertahap. Perubahan temperatur terhadap waktu tersebut selanjutnya digunakan untuk menganalisis karakteristik respon sistem pemanasan pada pengujian *Open Loop*.



Gambar 4.41 Pembacaan Tegangan *Heater*



Gambar 4.42 Pembacaan Arus *Heater*

Berdasarkan Gambar 4.41 dan Gambar 4.42, pada saat *Heater* diberikan *input* sebesar 69%, dilakukan pengukuran terhadap tegangan dan arus listrik yang digunakan oleh *Heater*. Hasil pengukuran menunjukkan bahwa tegangan AC yang diterima *Heater* sebesar 194,1 V, sedangkan arus yang terukur sebesar 1,59 A. Pengukuran tersebut dilakukan untuk mengetahui kondisi kelistrikan *Heater* ketika beroperasi selama pengujian *Open Loop*.

Nilai tegangan sebesar 194,1 V menunjukkan tegangan AC aktual yang terukur pada saat *Heater* diberikan *input* 69%. Sementara itu, arus sebesar 1,59 A menunjukkan besarnya arus yang mengalir pada rangkaian *Heater* pada kondisi pengujian tersebut. Kedua nilai tersebut merupakan hasil pengukuran secara langsung sehingga digunakan sebagai data pendukung untuk menunjukkan kondisi kerja *Heater* selama proses pemanasan.

Berdasarkan hasil pengukuran tersebut, estimasi daya listrik *Heater* dihitung dari tegangan dan arus terukur menggunakan Persamaan (2.25).

Tabel 4.4 Hasil Pengukuran

Parameter	Hasil Pengukuran
<i>Input Heater</i>	69%
Tegangan AC	194,1 V
Arus AC	1.59 A
Daya Hasil Perhitungan	308,62 W
Daya Maksimal <i>Heater</i>	500 W

Berdasarkan Tabel 4.3, *Heater* pada saat pengujian diberikan *input* sebesar 69% dengan tegangan AC terukur sebesar 194,1 V dan arus sebesar 1,59 A. Dari hasil perkalian antara tegangan dan arus diperoleh daya sekitar 308,62 W. Nilai tersebut menunjukkan bahwa *Heater* bekerja pada kondisi daya yang lebih rendah dibandingkan daya maksimalnya sebesar 500 W.

Kondisi kelistrikan tersebut menjadi salah satu faktor yang memengaruhi respon temperatur air. Semakin besar daya yang diberikan kepada *Heater*, semakin besar energi panas yang dapat ditransfer ke air. Pada pengujian ini, *input Heater* sebesar 69% menghasilkan respon temperatur yang kemudian digunakan untuk menentukan karakteristik plant dan melakukan tuning PID menggunakan metode Ziegler–Nichols.

4.7.2 Data Hasil Pengujian *Open Loop* Pemanasan Air 4 Liter

Setelah kondisi awal sistem ditentukan dan *Heater* diberikan masukan tetap sebesar 69%, dilakukan pengambilan data temperatur selama proses pemanasan. Pengambilan data dilakukan secara berkala untuk mengetahui perubahan temperatur air terhadap waktu. Data yang diperoleh meliputi waktu pengamatan, temperatur air, nilai ADC sensor PT100, nilai *Heater Power*, serta kondisi *Heater* selama proses pengujian.

Pengujian ini dilakukan menggunakan air sebanyak 4liter dengan temperatur awal sebesar 23,06 °C. Nilai *Heater Power* dipertahankan sebesar 69% selama proses pemanasan sehingga respon temperatur yang diperoleh dapat digunakan untuk mengetahui karakteristik plant secara *Open Loop*. Hasil pengambilan data selama proses pemanasan ditunjukkan pada Tabel 4.4.

Tabel 4 5 Data hasil Pengujian PWM *Heater* 69%

Waktu (Menit)	Temperatur (°C)	ADC PT100	<i>Heater Power</i> (%)
1	23.06	2847	69
2	23.87	2847	69
3	24.22	2849	69
4	25.18	2849	69
5	25.78	2847	69
6	26.18	2846	69
7	26.94	2853	69
8	27.34	2846	69
9	27.74	2843	69
10	28.70	2851	69

Waktu (Menit)	Temperatur (°C)	ADC PT100	Heater Power (%)
11	28.95	2843	69
12	29.86	2845	69
13	30.77	2848	69
14	31.07	2849	69
15	31.82	2832	69
16	32.83	2841	69
17	33.33	2853	69
18	34.14	2848	69
19	34.79	2841	69
20	36.56	2849	69
21	36.81	2842	69
22	37.71	2852	69
23	37.92	2849	69
24	38.67	2849	69
25	39.12	2847	69
26	40.13	2852	69
27	40.74	2845	69
28	41.79	2858	69
29	42.70	2839	69
30	43.66	2841	69
31	44.16	2849	69
32	44.81	2849	69
33	45.77	2855	69
34	46.48	2841	69
35	47.18	2851	69
36	47.83	2847	69
37	48.49	2839	69
38	49.24	2848	69
39	49.90	2849	69
40	50.70	2855	69
41	51.21	2851	69
42	52.01	2837	69
43	52.32	2849	69
44	53.83	2853	69
45	53.73	2844	69
46	54.48	2853	69
47	54.98	2843	69
48	55.69	2835	69
49	56.14	2837	69
50	57.00	2839	69
51	57.30	2854	69

Waktu (Menit)	Temperatur (°C)	ADC PT100	Heater Power (%)
52	57.50	2847	69
53	58.56	2849	69
54	58.61	2847	69
55	59.11	2846	69
56	59.72	2855	69
57	60.27	2837	69
58	60.88	2843	69
59	61.08	2844	69
60	61.53	2854	69
61	61.68	2847	69
62	62.64	2838	69
63	62.29	2850	69
64	63.14	2839	69
65	63.54	2850	69
66	64.20	2839	69
67	64.45	2843	69
68	64.30	2853	69
69	64.95	2839	69
70	65.31	2849	69
71	65.31	2849	69
72	66.21	2847	69
73	65.96	2845	69
74	66.82	2838	69
75	66.82	2846	69
76	67.12	2856	69
77	67.57	2843	69
78	67.77	2851	69
79	67.98	2845	69
80	68.23	2847	69
81	68.38	2839	69
82	68.28	2839	69
83	68.98	2836	69
84	68.88	2846	69
85	69.18	2846	69
86	69.08	2846	69
87	69.34	2851	69
88	69.49	2845	69
89	69.44	2858	69
90	70.04	2849	69
91	69.84	2842	69
92	70.14	2847	69

Waktu (Menit)	Temperatur (°C)	ADC PT100	Heater Power (%)
93	70.29	2848	69
94	70.24	2846	69
95	70.54	2853	69
96	70.34	2839	69
97	70.59	2850	69
98	70.95	2843	69
99	70.75	2851	69
100	70.80	2852	69
101	70.85	2847	69
102	71.00	2848	69
103	71.10	2847	69
104	71.25	2842	69
105	71.35	2847	69
106	70.95	2847	69
107	71.25	2847	69
108	71.65	2849	69
109	71.95	2847	69
110	72.05	2845	69
111	72.16	2851	69
112	71.90	2861	69
113	72.26	2847	69
114	72.36	2849	69
115	72.05	2845	69
116	72.31	2843	69
117	72.36	2847	69
118	72.51	2842	69
119	72.21	2847	69
120	72.26	2839	69
121	72.10	2850	69
122	72.41	2846	69
123	72.26	2839	69
124	72.16	2843	69
125	72.41	2845	69
126	72.41	2850	69
127	72.31	2851	69
128	72.51	2845	69
129	72.41	2851	69
130	72.46	2847	69
131	72.41	2847	69
132	72.31	2846	69
133	72.31	2853	69

Waktu (Menit)	Temperatur (°C)	ADC PT100	Heater Power (%)
134	72.31	2836	69
135	72.46	2850	69
136	72.71	2837	69
137	72.51	2855	69

Berdasarkan Tabel 4.4, dapat dilihat bahwa temperatur air mengalami peningkatan secara bertahap selama proses pemanasan. Pada awal pengujian, temperatur air sebesar 23,06 °C. Setelah *Heater* mulai bekerja, temperatur terus meningkat hingga mencapai kisaran 70 °C. Pada bagian akhir pengujian, temperatur berada pada kisaran 72 °C dengan nilai tertinggi sebesar 72,71 °C dan pada data terakhir sebesar 72,51 °C.

Pada tahap awal pemanasan, perubahan temperatur berlangsung lebih cepat karena perbedaan temperatur antara air dan lingkungan masih cukup besar. Seiring meningkatnya temperatur air, kenaikan temperatur menjadi semakin kecil. Kondisi tersebut terlihat dari data pada bagian akhir pengujian yang menunjukkan perubahan temperatur relatif kecil dan berada di sekitar 72 °C. Hal ini menunjukkan bahwa sistem mulai mendekati kondisi steady state.

Nilai ADC PT100 selama pengujian juga mengalami perubahan kecil di sekitar nilai 2840 sampai 2850. Perubahan tersebut merupakan bagian dari proses pembacaan sensor selama temperatur air meningkat. Sementara itu, *Heater Power* dipertahankan sebesar 69% selama proses pemanasan sesuai dengan kondisi pengujian *Open Loop*.

Data tersebut selanjutnya digunakan untuk membuat grafik respon temperatur terhadap waktu. Grafik kemudian dianalisis untuk memperoleh karakteristik plant, yaitu steady state, *Rise Time*, time constant, *dead time*, dan gain plant. Hasil karakteristik tersebut selanjutnya digunakan sebagai dasar dalam menentukan parameter PID menggunakan metode Ziegler–Nichols.

4.7.3 Grafik Hasil Percobaan

Setelah seluruh data temperatur diperoleh seperti yang ditunjukkan pada Tabel 4.4, data tersebut selanjutnya diolah dalam bentuk grafik untuk melihat karakteristik respon temperatur terhadap waktu selama proses pemanasan. Grafik dibuat dengan waktu pengujian sebagai sumbu horizontal dan temperatur air sebagai sumbu vertikal. Penyajian data dalam bentuk grafik bertujuan untuk mempermudah

pengamatan terhadap pola kenaikan temperatur, perubahan laju pemanasan, serta kondisi ketika temperatur mulai mendekati keadaan stabil.

Pada pengujian ini, temperatur awal air sebesar 23,06 °C, sedangkan *Heater* diberikan masukan tetap sebesar 69%. Selama proses berlangsung, temperatur dicatat secara berkala hingga mencapai kondisi mendekati steady state. Oleh karena itu, grafik respon temperatur digunakan sebagai dasar untuk menentukan karakteristik plant yang meliputi *Rise Time*, time constant, *dead time*, dan steady state.

Berdasarkan Gambar 3.14, terlihat bahwa temperatur awal sistem berada pada 23,06 °C. Setelah *Heater* diberikan masukan sebesar 69%, temperatur mulai mengalami kenaikan secara bertahap. Pada bagian awal grafik, kenaikan temperatur berlangsung secara relatif cepat. Seiring bertambahnya waktu, kenaikan temperatur mulai melambat hingga akhirnya temperatur mendekati kondisi stabil.

Pada akhir pengujian, temperatur mencapai nilai sekitar 72,51 °C. Nilai tersebut digunakan sebagai kondisi steady state dari respon *Open Loop*. Walaupun pada bagian akhir grafik masih terdapat perubahan kecil pada temperatur, perubahan tersebut relatif kecil dan berada di sekitar nilai 72,51 °C sehingga dapat dianggap sebagai kondisi stabil sistem pada pengujian ini.

Dari kondisi awal sebesar 23,06 °C hingga steady state sebesar 72,51 °C, terjadi perubahan temperatur sebesar 49,45 °C. Perubahan tersebut menunjukkan bahwa pemberian daya *Heater* sebesar 69% mampu menghasilkan kenaikan temperatur yang cukup besar pada air yang diuji.

Pada grafik juga terlihat beberapa titik karakteristik respon, yaitu nilai 10%, 63,2%, dan 90% dari perubahan temperatur. Titik 10% berada pada temperatur sekitar 28,01 °C dan dicapai pada waktu sekitar 9,28 menit. Titik ini menunjukkan bagian awal dari respon pemanasan setelah temperatur mulai mengalami perubahan.

Selanjutnya, titik 63,2% berada pada temperatur sekitar 54,31 °C dan dicapai pada waktu sekitar 45,78 menit. Titik 63,2% digunakan karena merupakan salah satu titik karakteristik yang digunakan untuk menentukan time constant pada pendekatan model proses orde satu dengan *dead time*.

Titik 90% berada pada temperatur sekitar 67,57 °C dan dicapai pada waktu sekitar 76,99 menit. Setelah mencapai titik tersebut, kurva semakin mendekati kondisi steady state. Hal ini dapat dilihat dari bentuk kurva pada bagian akhir yang semakin landai.

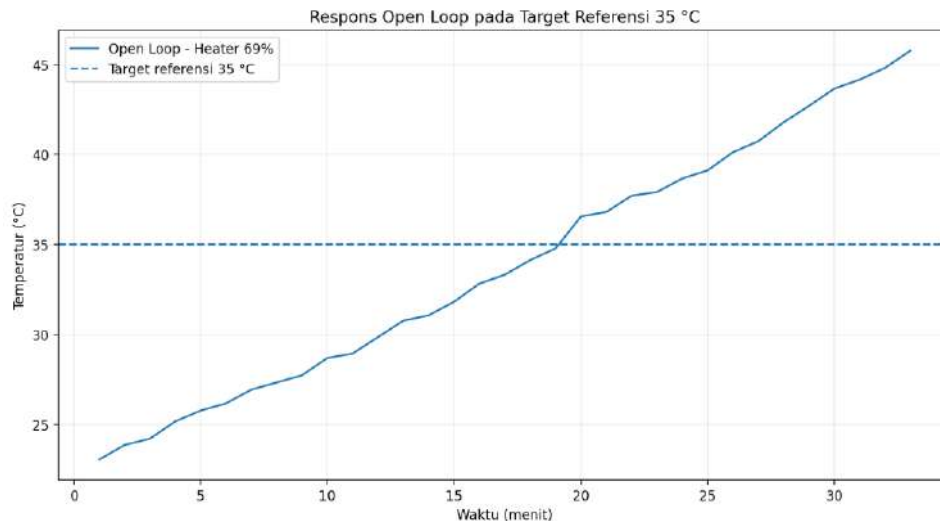
Pada grafik juga ditunjukkan titik infleksi pada sekitar 18,00 menit. Titik infleksi menunjukkan daerah perubahan karakteristik kemiringan kurva temperatur. Setelah melewati daerah tersebut, respon masih mengalami kenaikan, tetapi secara bertahap mulai menuju kondisi yang semakin landai.

Berdasarkan bentuk keseluruhan grafik, sistem pemanasan menunjukkan karakteristik respon yang relatif lambat. Hal tersebut terlihat dari waktu yang diperlukan temperatur untuk meningkat dari kondisi awal hingga mendekati *steady state*. Karakteristik tersebut dipengaruhi oleh proses perpindahan panas antara *Heater* dan air serta kehilangan panas ke lingkungan.

Hasil pengujian juga menunjukkan bahwa *Heater* yang diberikan daya tetap tidak menyebabkan temperatur langsung mencapai nilai *steady state*. Temperatur membutuhkan waktu yang cukup panjang untuk mencapai kondisi mendekati stabil. Oleh karena itu, sistem memerlukan pengendalian untuk menjaga temperatur pada nilai yang diinginkan dan mengurangi perbedaan antara temperatur aktual dengan *setpoint*.

4.7.4 Pengujian *Open Loop Setpoint* 35°C dengan Input *Heater* 69%

Analisis *Open Loop* pada target *setpoint* 35 °C menggunakan data pengujian yang sama, yaitu volume air 4 liter, temperatur awal 23,06 °C, dan *input Heater* tetap 69%. Target 35 °C tidak digunakan sebagai umpan balik, sehingga ketika temperatur mencapai atau melewati nilai tersebut, *Heater* tetap bekerja pada masukan yang sama. Kondisi ini digunakan untuk menunjukkan karakteristik *Open Loop* sebelum dibandingkan dengan pengendalian PID.



Gambar 4. 43 Grafik *Open Loop Setpoint 35°C*

Berdasarkan Gambar 4.43, temperatur meningkat secara bertahap dari kondisi awal 23,06 °C menuju target setpoint 35 °C. Setelah target tercapai, temperatur masih terus meningkat karena sistem *Open Loop* tidak mempunyai mekanisme untuk mengurangi keluaran *Heater* berdasarkan temperatur aktual. *Heater* tetap bekerja dengan input 69%.

Mengacu pada Persamaan (2.1), waktu pencapaian titik 10% dan 90% terhadap perubahan dari temperatur awal menuju target 35 °C masing-masing diperoleh sekitar 3,04 menit dan 17,59 menit. Dengan demikian, *Rise Time Open Loop* pada target 35 °C sebesar 14,55 menit.

Pada data pencatatan, target 35 °C pertama kali terlewati pada sekitar menit ke-20. Karena *input Heater* tidak dikurangi, 10 menit setelah titik tersebut temperatur mencapai 43,66 °C. Mengacu pada Persamaan (2.4), selisih temperatur terhadap target pada titik evaluasi tersebut sebesar 8,66 °C. Jika dibandingkan terhadap target 35 °C, penyimpangan tersebut setara dengan sekitar 24,74%, sehingga kedekatan respons terhadap target pada titik evaluasi adalah sekitar 75,26%.

Tabel 4. 6 Hasil Percobaan *Open Loop Setpoint 35°C*

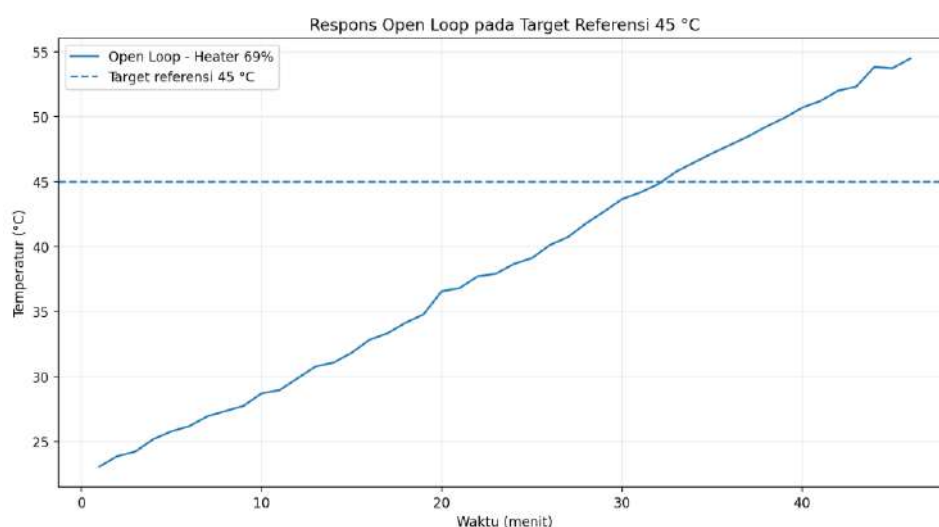
Parameter	Hasil
<i>Setpoint</i>	35 °C
<i>Input Heater</i>	69% tetap
Temperatur Awal	23,06 °C
Waktu 10%	3,04 menit

Parameter	Hasil
Waktu 90%	17,59 menit
<i>Rise Time</i>	14,55 menit
Target pertama kali terlewati	Menit ke-20
Temperatur 10 menit setelah target	43,66 °C
<i>Error</i> 10 menit setelah target	8,66 °C
<i>Error</i> terhadap target	24,74%
Akurasi terhadap target	75,26%

Berdasarkan Tabel 4.6, sistem *Open Loop* dapat mencapai target *setpoint* 35 °C, tetapi tidak dapat mempertahankan temperatur pada nilai tersebut karena *Heater* tetap bekerja 69%. Oleh karena itu, hasil ini digunakan sebagai kondisi pembanding terhadap pengujian *Closed Loop setpoint* 35 °C

4.7.5 Pengujian *Open Loop Setpoint* 45°C dengan Input Heater 69%

Analisis *Open Loop* pada target *setpoint* 45 °C menggunakan data pengujian yang sama, yaitu volume air 4 liter, temperatur awal 23,06 °C, dan input *Heater* tetap 69%. Target 45 °C tidak digunakan sebagai umpan balik, sehingga ketika temperatur mencapai atau melewati nilai tersebut, *Heater* tetap bekerja pada masukan yang sama. Kondisi ini digunakan untuk menunjukkan karakteristik *Open Loop* sebelum dibandingkan dengan pengendalian PID.



Gambar 4. 44 Grafik *Open Loop Setpoint* 45°C

Berdasarkan Gambar 4.44, temperatur meningkat secara bertahap dari kondisi awal 23,06 °C menuju target *setpoint* 45 °C. Setelah target tercapai, temperatur

masih terus meningkat karena sistem *Open Loop* tidak mempunyai mekanisme untuk mengurangi keluaran *Heater* berdasarkan temperatur aktual. *Heater* tetap bekerja dengan input 69%.

Mengacu pada Persamaan (2.1), waktu pencapaian titik 10% dan 90% terhadap perubahan dari temperatur awal menuju target 45 °C masing-masing diperoleh sekitar 4,12 menit dan 29,11 menit. Dengan demikian, *Rise Time Open Loop* pada target 45 °C sebesar 24,99 menit.

Pada data pencatatan, target 45 °C pertama kali terlewati pada sekitar menit ke-33. Karena *input Heater* tidak dikurangi, 10 menit setelah titik tersebut temperatur mencapai 52,32 °C. Mengacu pada Persamaan (2.4), selisih temperatur terhadap target pada titik evaluasi tersebut sebesar 7,32 °C. Jika dibandingkan terhadap target 45 °C, penyimpangan tersebut setara dengan sekitar 16,27%, sehingga kedekatan respons terhadap target pada titik evaluasi adalah sekitar 83,73%.

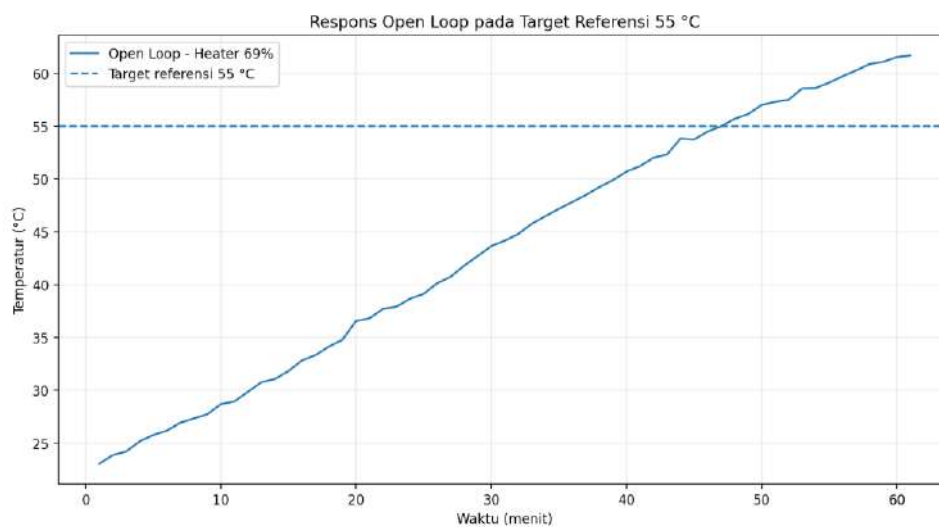
Tabel 4. 7 Hasil Percobaan *Open Loop Setpoint 45°C*

Parameter	Hasil
<i>Setpoint</i>	45 °C
<i>Input Heater</i>	69% tetap
Temperatur Awal	23,06 °C
Waktu 10%	4,12 menit
Waktu 90%	29,11 menit
<i>Rise Time</i>	24,99 menit
Target pertama kali terlewati	Menit ke-33
Temperatur 10 menit setelah target	52,32 °C
Error 10 menit setelah target	7,32 °C
<i>Error terhadap target</i>	16,27%
Akurasi terhadap target	83,73%

Berdasarkan Tabel 4.7, sistem *Open Loop* dapat mencapai target *setpoint* 45 °C, tetapi tidak dapat mempertahankan temperatur pada nilai tersebut karena *Heater* tetap bekerja 69%. Oleh karena itu, hasil ini digunakan sebagai kondisi pembanding terhadap pengujian *Closed Loop setpoint 45 °C*.

4.7.6 Pengujian *Open Loop Setpoint 55°C* dengan *Input Heater 69%*

Analisis *Open Loop* pada target *setpoint 55 °C* menggunakan data pengujian yang sama, yaitu volume air 4 liter, temperatur awal 23,06 °C, dan *input Heater* tetap 69%. Target 55 °C tidak digunakan sebagai umpan balik, sehingga ketika temperatur mencapai atau melewati nilai tersebut, *Heater* tetap bekerja pada masukan yang sama. Kondisi ini digunakan untuk menunjukkan karakteristik *Open Loop* sebelum dibandingkan dengan pengendalian PID.



Gambar 4. 45 Grafik *Open Loop Setpoint 55°C*

Berdasarkan Gambar 4.45, temperatur meningkat secara bertahap dari kondisi awal 23,06 °C menuju target *setpoint 55 °C*. Setelah target tercapai, temperatur masih terus meningkat karena sistem *Open Loop* tidak mempunyai mekanisme untuk mengurangi keluaran *Heater* berdasarkan temperatur aktual. *Heater* tetap bekerja dengan *input 69%*.

Mengacu pada Persamaan (2.1), waktu pencapaian titik 10% dan 90% terhadap perubahan dari temperatur awal menuju target 55 °C masing-masing diperoleh sekitar 6,10 menit dan 41,75 menit. Dengan demikian, *Rise Time Open Loop* pada target 55 °C sebesar 35,65 menit.

Pada data pencatatan, target 55 °C pertama kali terlewati pada sekitar menit ke-48. Karena *input Heater* tidak dikurangi, 10 menit setelah titik tersebut temperatur mencapai 60,88 °C. Mengacu pada Persamaan (2.4), selisih temperatur terhadap target pada titik evaluasi tersebut sebesar 5,88 °C. Jika dibandingkan terhadap target

55 °C, penyimpangan tersebut setara dengan sekitar 10,69%, sehingga kedekatan respons terhadap target pada titik evaluasi adalah sekitar 89,31%.

Tabel 4. 8 Hasil Percobaan Open Loop Setpoint 55°C

Parameter	Hasil
<i>Setpoint</i>	55 °C
<i>Input Heater</i>	69% tetap
Temperatur Awal	23,06 °C
Waktu 10%	6,10 menit
Waktu 90%	41,75 menit
<i>Rise Time</i>	35,65 menit
Target pertama kali terlewati	Menit ke-48
Temperatur 10 menit setelah target	60,88 °C
<i>Error</i> 10 menit setelah target	5,88 °C
<i>Error</i> terhadap target	10,69%
Akurasi terhadap target	89,31%

Berdasarkan Tabel 4.8, sistem *Open Loop* dapat mencapai target *setpoint* 55 °C, tetapi tidak dapat mempertahankan temperatur pada nilai tersebut karena Heater tetap bekerja 69%. Oleh karena itu, hasil ini digunakan sebagai kondisi pembanding terhadap pengujian *Closed Loop setpoint* 55 °C. Parameter utama yang dibandingkan pada Subbab 4.9 adalah *Rise Time* dan besar *Error* temperatur setelah target tercapai.

4.7.7 Perbandingan Respons Open Loop

Untuk melihat keseluruhan respons *Open Loop* terhadap tiga target temperatur yang digunakan pada pengujian *Closed Loop*, data pemanasan dengan input Heater tetap 69% ditampilkan dalam satu grafik. Karena ketiga target *setpoint* dianalisis dari percobaan Open Loop yang sama, kurva temperatur yang digunakan tetap satu kurva respons. Nilai 35 °C, 45 °C, dan 55 °C ditampilkan sebagai garis target *setpoint* untuk menunjukkan kapan respons *Open Loop* melewati masing-masing temperatur.

Tabel 4. 9 Perbandingan Respon *Open Loop*

Target <i>Setpoint</i>	Rise Time (menit)	Target Pertama Dilewati (menit)	Temperatur +10 Menit (°C)	Error +10 Menit (°C)	Error +10 Menit (%)
35 °C	14,55	20	43,66	8,66	24,74
45 °C	24,99	33	52,32	7,32	16,27
55 °C	35,65	48	60,88	5,88	10,69

Berdasarkan Tabel 4.9, semakin tinggi target *setpoint* maka waktu yang dibutuhkan respons *Open Loop* untuk mencapainya semakin lama. Namun, pada seluruh target terlihat bahwa temperatur terus meningkat setelah target terlewati. Hal tersebut menjadi karakteristik utama sistem *Open Loop* pada pengujian ini karena input *Heater* 69% tidak dikoreksi berdasarkan temperatur aktual.

4.8 Pengujian Sistem Kontrol *Closed Loop*

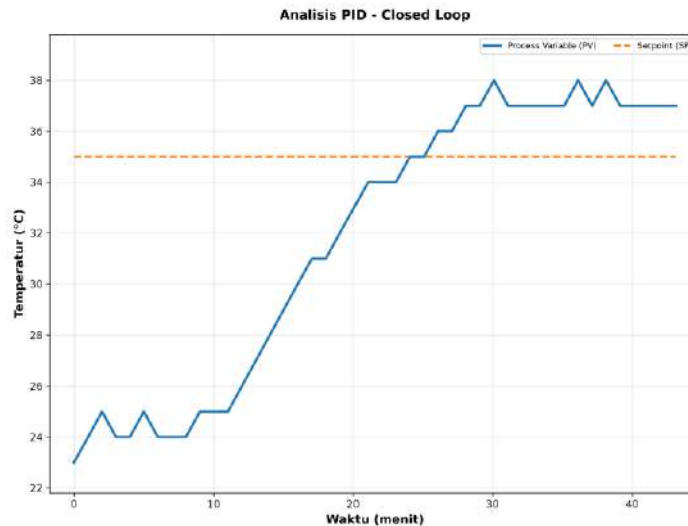
Pengujian *Closed Loop* dilakukan setelah parameter PID diperoleh dari pengujian *Open Loop*. Parameter yang digunakan pada Function Block CTRL_PID di *OpenPLC* adalah $K_P = 40,09$, $T_N = 5,59$ menit, dan $T_V = 1,40$ menit. Nilai temperatur yang diterima *OpenPLC* digunakan sebagai process variable dan dibandingkan dengan *setpoint* untuk menghasilkan keluaran PID.

Pengujian dilakukan pada *setpoint* 35 °C, 45 °C, dan 55 °C. Data temperatur dan keluaran PID dicatat untuk melihat *Rise Time*, *Overshoot* terhadap *setpoint*, pemenuhan pita *settling* $\pm 5\%$, serta *Error* akhir terhadap *setpoint*. Pembacaan *Arduino IDE* dan *Thermometer* digunakan sebagai pembanding terhadap nilai temperatur yang diterima *OpenPLC*.

4.8.1 Pengujian *Closed Loop Setpoint 35 °C*

Pada percobaan ini sistem diberikan *setpoint* temperatur sebesar 35 °C. Kondisi air pada tangki tidak menggunakan proses *mixing*, sehingga selama proses pemanasan air hanya menerima panas dari *Heater* tanpa adanya pengadukan untuk meratakan temperatur di seluruh bagian tangki. Setelah sistem dijalankan, data temperatur dicatat setiap satu menit sampai kondisi temperatur berada di atas *setpoint* dan perubahan temperatur mulai mengecil.

Data yang diperoleh kemudian dibuat menjadi grafik untuk melihat perubahan temperatur terhadap waktu. Garis *setpoint* digunakan sebagai pembanding untuk melihat kapan temperatur mulai mencapai nilai 35 °C dan bagaimana kondisi temperatur setelah *setpoint* tersebut tercapai. Hasil respon temperatur ditunjukkan pada Gambar 4.46.



Gambar 4.46 Hasil grafik Percobaan *Setpoint* 35 °C

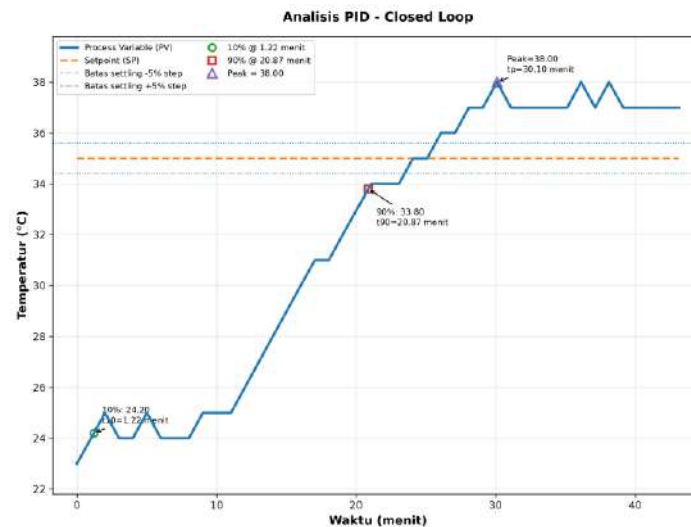
Berdasarkan Gambar 4.46, temperatur awal yang terbaca berada di sekitar 23 °C. Pada beberapa menit pertama temperatur masih berubah pada kisaran 24–25 °C, kemudian mulai mengalami kenaikan yang lebih jelas setelah sekitar menit ke-11. Temperatur selanjutnya naik secara bertahap dari 25 °C, 31 °C, 34 °C, hingga mencapai *setpoint* 35 °C pada sekitar menit ke-25.

Setelah mencapai *setpoint*, temperatur tidak langsung berhenti pada 35 °C. Temperatur masih mengalami kenaikan menjadi 36 °C, 37 °C, kemudian mencapai nilai maksimum sebesar 38 °C pada sekitar menit ke-30. Setelah nilai maksimum tersebut tercapai, temperatur turun menjadi sekitar 37 °C dan selanjutnya bergerak pada kisaran 37–38 °C sampai akhir pengujian.

Dari bentuk kurva tersebut terlihat bahwa sistem sudah mampu membawa temperatur menuju *setpoint*, tetapi temperatur masih meningkat setelah 35 °C tercapai. Pada bagian akhir pengujian pembacaan *OpenPLC* lebih banyak berada di sekitar 37–38 °C dan masih belum menunjukkan kondisi *steady-state* yang benar-benar stabil.

Untuk melihat lebih jelas karakteristik respon yang dihasilkan, grafik temperatur kemudian dianalisis dengan menentukan titik 10%, 90%, temperatur

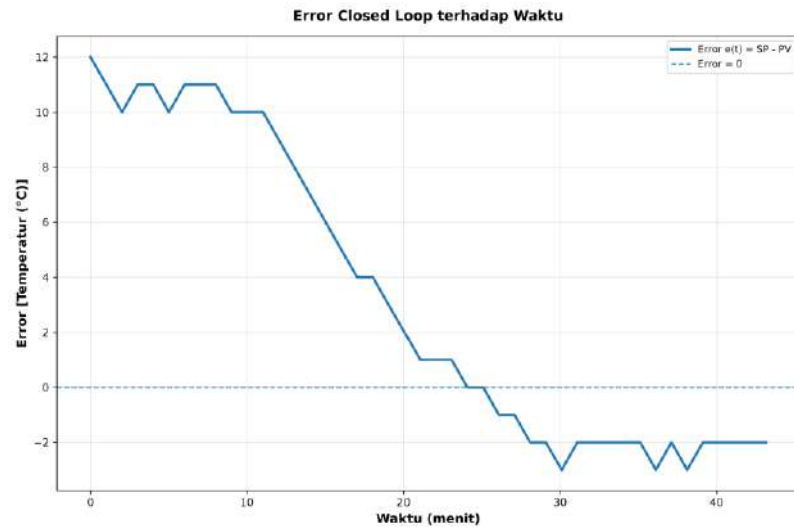
maksimum, dan daerah settling. Hasil analisis tersebut ditunjukkan pada Gambar 4.47.



Gambar 4.47 Hasil Grafik Keterangan *Setpoint* 35 °C

Berdasarkan Gambar 4.47, temperatur awal berada di sekitar 23 °C dengan *setpoint* 35 °C. Berdasarkan perhitungan Persamaan (2.12), perubahan temperatur acuan diperoleh sebesar 12 °C. Selanjutnya, berdasarkan Persamaan (2.13) dan Persamaan (2.15), titik 10% dan 90% masing-masing diperoleh pada temperatur 24,20 °C dan 33,80 °C. Berdasarkan interpolasi yang mengacu pada Persamaan (2.16), waktu pencapaian titik 10% diperoleh sekitar 1,22 menit dan titik 90% sekitar 20,87 menit.

Selain respons temperatur, perubahan *Error* selama pengujian juga diamati. *Error* dihitung sebagai selisih *setpoint* 35 °C dan temperatur yang terbaca sesuai Persamaan (2.10), kemudian ditampilkan pada grafik untuk melihat perubahan tanda dan besar penyimpangan selama proses.

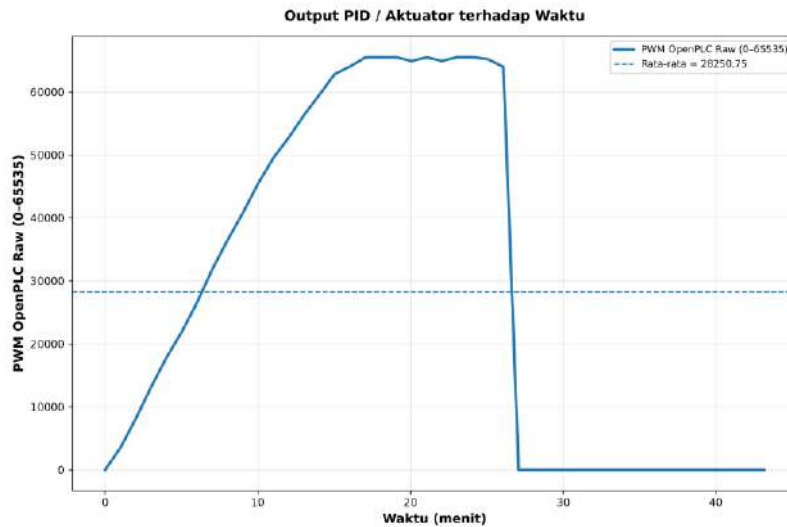


Gambar 4.48 Hasil Grafik *Error Setpoint* Terhadap Aktual Temperatur

Berdasarkan Gambar 4.48, pada awal pengujian *Error* berada di sekitar 12 °C karena temperatur awal masih berada jauh di bawah *setpoint*. Seiring temperatur meningkat, nilai *Error* semakin kecil sampai mencapai sekitar 0 °C ketika temperatur mulai mencapai 35 °C.

Setelah temperatur melewati *setpoint*, *Error* berubah menjadi negatif. Pada bagian akhir pengujian *Error* lebih banyak berada di sekitar -2 °C, sedangkan pada beberapa titik mencapai sekitar -3 °C ketika temperatur terbaca sebesar 38 °C. Kondisi ini menunjukkan bahwa setelah *setpoint* tercapai, temperatur yang dibaca sistem masih berada di atas nilai yang diinginkan.

Perubahan temperatur tersebut juga dapat dibandingkan dengan keluaran PID yang diberikan selama pengujian. Grafik keluaran PID ditunjukkan pada Gambar 4.46.



Gambar 4.49 Grafik Laju PWM percobaan *Setpoint* 35°C

Berdasarkan Gambar 4.49, keluaran PID pada awal pengujian meningkat secara bertahap seiring dengan kebutuhan pemanasan. Nilai keluaran kemudian mencapai daerah mendekati maksimum, yaitu sekitar 65000 pada skala keluaran *OpenPLC*, dan bertahan pada nilai yang cukup tinggi selama temperatur masih bergerak menuju *setpoint*.

Pada sekitar menit ke-27, keluaran PID turun menjadi 0 dan tetap berada pada kondisi tersebut sampai akhir data yang tercatat pada *OpenPLC*. Walaupun keluaran sudah dihentikan, temperatur masih mengalami kenaikan dan mencapai 38 °C beberapa menit kemudian. Hal ini menunjukkan bahwa panas yang sudah masuk ke air dan elemen *Heater* masih memberikan pengaruh setelah keluaran kendali dihentikan. Karena tangki juga tidak menggunakan *mixing*, panas di dalam air tidak langsung tersebar merata pada seluruh bagian tangki.

Untuk melihat karakteristik respon waktu dari pengujian *setpoint* 35 °C, dilakukan perhitungan parameter sebagai berikut.

a. *Rise Time*

Berdasarkan perhitungan menggunakan Persamaan (2.1), *rise time* pada pengujian *closed loop setpoint* 35 °C diperoleh sebesar 19,65 menit. Nilai tersebut menunjukkan waktu yang diperlukan respons untuk bergerak dari titik 10% sampai 90% perubahan temperatur menuju *setpoint*.

b. *Overshoot*

Temperatur maksimum yang tercatat sebesar 38 °C dengan *setpoint* 35 °C. Berdasarkan Persamaan (2.2), *Overshoot* terhadap *setpoint* diperoleh sebesar 8,57%..

c. *Settling time*

Dengan menggunakan pita toleransi $\pm 5\%$ terhadap *setpoint* 35 °C, batas *settling* berada pada 33,25–36,75 °C. Pada bagian akhir pengujian temperatur masih berada di sekitar 37–38 °C, sehingga respons tidak dapat tetap berada di dalam pita tersebut. Berdasarkan Persamaan (2.3), *Settling time* pada pengujian *setpoint* 35 °C dinyatakan belum tercapai sampai akhir pengujian.

d. *Error setpoint*

Pada bagian akhir pengujian, temperatur representatif pada *OpenPLC* berada di sekitar 37 °C, sedangkan *setpoint* sebesar 35 °C. Berdasarkan Persamaan (2.4), *Error* akhir terhadap *setpoint* adalah 2 °C.

Setelah dilakukan analisis dan perhitungan berdasarkan kurva respon, hasil parameter pengujian *setpoint* 35 °C disajikan pada Tabel 4.10.

Tabel 4.10 Parameter Respon *Closed Loop Setpoint* 35 °C

<i>Setpoint</i>	<i>Rise Time</i>	<i>Settling time</i>	<i>Overshoot</i>	<i>Error setpoint</i>
35 °C	19,65 menit	Belum tercapai	8,57%	2 °C

Berdasarkan Gambar 4.47 dan Tabel 4.10, sistem mampu mencapai *setpoint* 35 °C, tetapi temperatur masih naik hingga 38 °C dan pada bagian akhir pembacaan *OpenPLC* berada di sekitar 37–38 °C. *Settling time* belum tercapai pada pita $\pm 5\%$ *setpoint*, sedangkan *Error* akhir terhadap *setpoint* sebesar sekitar 2 °C.

Selain data yang tercatat pada *OpenPLC*, dilakukan juga pengecekan pada sisi Arduino IDE untuk melihat pembacaan PT100 dan kondisi keluaran *Heater* secara langsung. Hasil pembacaan Arduino IDE pada saat pengujian ditunjukkan pada Gambar 4.50.

```

=====
JSN Raw      : 32.88 cm
JSN Filtered : 32.88 cm
PWM Distance  : 673
=====
ADC PT100    : 1439
Temperature   : 40.58 C
=====
Flow Pulse    : 0
Flow Rate     : 0.00 L/min
PWM Flow      : 0
=====
PID Flow ADC  : 0
Motor PWM     : 0
=====
PID Heater ADC : 2354
Heater Power  : 57 %
=====

```

Gambar 4.50 Pembacaan Temperatur Pada Akhir Pengujian di Arduino IDE

Berdasarkan Gambar 4.50, nilai ADC PT100 sebesar 1439 dan temperatur yang terbaca pada Arduino IDE sebesar 40,58 °C. Pada kondisi tersebut nilai PID *Heater* ADC sebesar 2354 dan *Heater* Power terbaca sebesar 57%.

Hasil pembacaan tersebut memperlihatkan adanya perbedaan dengan nilai temperatur yang tercatat pada *OpenPLC*. Pada grafik *OpenPLC* temperatur bagian akhir berada di sekitar 37 °C, sedangkan Arduino membaca 40,58 °C. Nilai *Heater* Power yang masih terbaca 57% pada Arduino juga menunjukkan bahwa nilai yang diterima pada sisi Arduino tidak selalu sama dengan kondisi keluaran yang tercatat pada grafik *OpenPLC*. Kondisi ini sejalan dengan hasil pengujian komunikasi *pin to pin* sebelumnya, di mana nilai yang dikirim harus melewati sinyal PWM, *RC filter*, dan pembacaan ADC sebelum diterima oleh perangkat berikutnya.

Untuk melihat kondisi air secara langsung, temperatur di dalam tangki kemudian diukur menggunakan *Thermometer* digital. Pengukuran dilakukan pada kondisi tangki yang sama dan tanpa menggunakan proses *mixing*. Hasil pengukuran ditunjukkan pada Gambar 4.51.



Gambar 4.51 Pengukuran Akutual di Akhir Pengujian

Berdasarkan Gambar 4.51, temperatur yang terbaca menggunakan *Thermometer* sebesar 42,1 °C. Nilai tersebut lebih tinggi dibandingkan temperatur pada Arduino IDE sebesar 40,58 °C dan pembacaan *OpenPLC* yang berada di sekitar 37 °C.

Perbedaan pembacaan tersebut menunjukkan bahwa temperatur air pada tangki belum merata. Pada pengujian ini tidak digunakan *mixing*, sehingga air yang berada dekat dengan *Heater* dapat mempunyai temperatur yang lebih tinggi dibandingkan bagian lain. Posisi PT100 dan *Thermometer* yang tidak berada pada titik yang sama juga membuat kedua alat dapat membaca kondisi temperatur yang berbeda.

Selain itu, perbedaan antara pembacaan Arduino dan *OpenPLC* kembali menunjukkan adanya pengaruh dari jalur komunikasi dan proses *scalling*. Karena PID menggunakan temperatur pada *OpenPLC* sebagai process variable, keputusan keluaran kontrol mengikuti nilai yang diterima pada *OpenPLC*, sedangkan temperatur aktual pada bagian lain tangki dapat sudah berada pada nilai yang lebih tinggi.

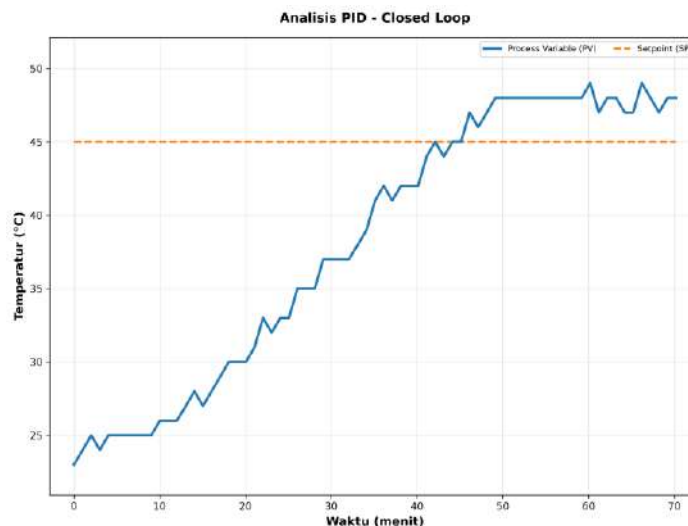
Secara keseluruhan, pada pengujian *setpoint* 35 °C, sistem sudah mampu menaikkan temperatur sampai mencapai target. Namun hasil akhirnya belum mampu mempertahankan temperatur air tepat pada 35 °C. Berdasarkan data *OpenPLC* temperatur berada di sekitar 37 °C, Arduino IDE membaca 40,58 °C, sedangkan hasil pengukuran langsung menunjukkan 42,1 °C. Perbedaan tersebut menunjukkan bahwa hasil pengendalian masih dipengaruhi oleh pembacaan

temperatur pada jalur komunikasi serta kondisi distribusi panas di dalam tangki yang belum merata karena tidak terdapat proses *mixing*.

4.8.2 Pengujian *Closed Loop Setpoint* 45 °C

Pada percobaan selanjutnya sistem diberikan *setpoint* temperatur sebesar 45 °C. Kondisi tangki masih sama seperti pada percobaan sebelumnya, yaitu air tidak menggunakan proses *mixing*. Selama proses berlangsung, perubahan temperatur dan keluaran PID dicatat setiap satu menit sampai temperatur melewati *setpoint* dan mulai berada pada kondisi yang lebih tetap.

Data hasil pengujian kemudian digunakan untuk melihat perubahan temperatur terhadap waktu. Dari grafik dapat dilihat bagaimana temperatur bergerak dari kondisi awal menuju *setpoint* 45 °C dan bagaimana kondisi temperatur setelah *setpoint* tersebut tercapai. Hasil respon temperatur ditunjukkan pada Gambar 4.52.



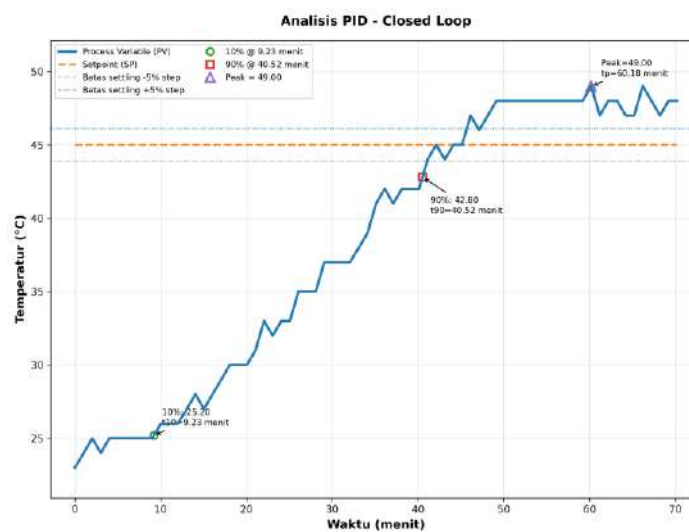
Gambar 4.52 Hasil Grafik Percobaan 45°C

Berdasarkan Gambar 4.52, temperatur awal yang terbaca berada di sekitar 23 °C. Pada bagian awal pengujian temperatur masih berada pada kisaran 24–25 °C, kemudian mulai naik secara bertahap. Setelah itu temperatur terus meningkat melewati 30 °C, 35 °C, 40 °C, sampai akhirnya mencapai *setpoint* 45 °C pada sekitar menit ke-42.

Setelah mencapai *setpoint*, temperatur tidak langsung berhenti pada 45 °C. Temperatur masih mengalami kenaikan menjadi 46 °C, 47 °C, kemudian berada di sekitar 48 °C. Nilai tertinggi yang tercatat pada pengujian ini sebesar 49 °C pada sekitar menit ke-60.

Setelah mencapai nilai tersebut, temperatur kembali turun ke sekitar 47–48 °C dan pada bagian akhir pengujian lebih banyak berada di sekitar 48 °C. Dari hasil tersebut terlihat bahwa sistem sudah mampu membawa temperatur sampai melewati *setpoint*, tetapi temperatur akhir masih berada beberapa derajat di atas nilai yang diinginkan.

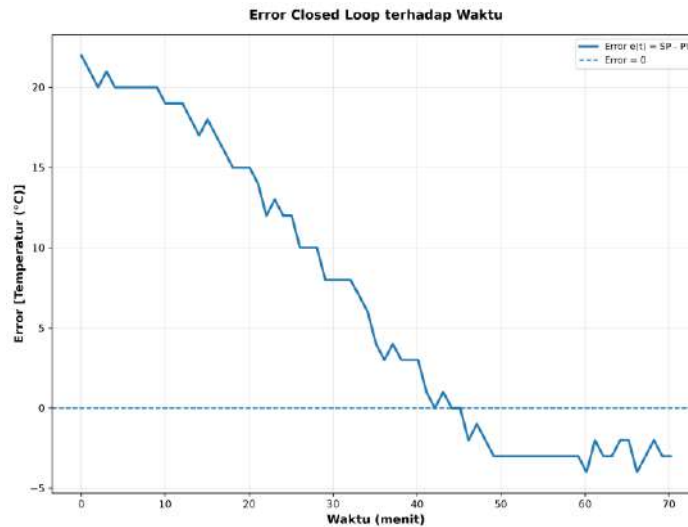
Untuk melihat karakteristik respon lebih jelas, data temperatur kemudian dianalisis menggunakan titik 10%, 90%, temperatur maksimum, dan batas settling seperti ditunjukkan pada Gambar 4.53.



Gambar 4.53 Hasil Grafik Keterangan *Setpoint* 45 °C

Berdasarkan Gambar 4.53, temperatur awal berada di sekitar 23 °C dengan *setpoint* 45 °C. Berdasarkan perhitungan Persamaan (2.12), perubahan temperatur acuan diperoleh sebesar 22 °C. Selanjutnya, berdasarkan Persamaan (2.13) dan Persamaan (2.15), titik 10% dan 90% masing-masing diperoleh pada temperatur 25,20 °C dan 42,80 °C. Berdasarkan interpolasi yang mengacu pada Persamaan (2.16), waktu pencapaian titik 10% diperoleh sekitar 9,23 menit dan titik 90% sekitar 40,52 menit.

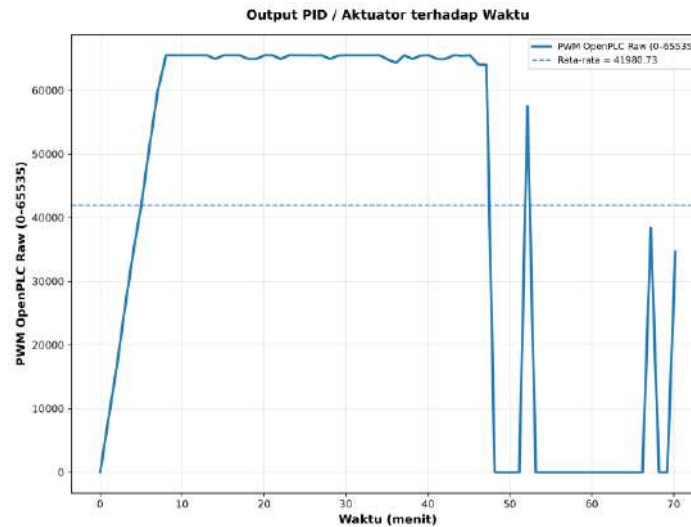
Selain respons temperatur, perubahan *Error* diamati dengan menghitung selisih *setpoint* 45 °C terhadap temperatur yang terbaca sesuai Persamaan (2.10). Hasil perubahan *Error* ditunjukkan pada Gambar 4.54.



Gambar 4.54 Hasil Grafik *Error Setpoint* Terhadap Aktual Temperatur

Berdasarkan Gambar 4.54, pada awal pengujian nilai *Error* berada di sekitar 22 °C karena temperatur awal masih jauh di bawah *setpoint* 45 °C. Seiring dengan kenaikan temperatur, nilai *Error* terus mengecil sampai mendekati 0 °C ketika temperatur mulai mencapai *setpoint*.

Setelah temperatur melewati 45 °C, nilai *Error* berubah menjadi negatif. Pada bagian akhir pengujian *Error* lebih banyak berada di sekitar -3 °C, sedangkan pada beberapa titik mencapai sekitar -4 °C ketika temperatur berada di sekitar 49 °C. Perubahan *Error* tersebut sesuai dengan grafik temperatur. Ketika temperatur masih berada di bawah *setpoint*, *Error* bernilai positif. Setelah temperatur melewati *setpoint*, *Error* berubah menjadi negatif karena temperatur sistem sudah berada di atas nilai yang diinginkan.



Gambar 4.55 Grafik Laju PWM percobaan *Setpoint* 45°C

Berdasarkan Gambar 4.55, keluaran PID pada awal pengujian meningkat cukup cepat sampai mendekati batas maksimum *OpenPLC*. Setelah itu nilai keluaran bertahan cukup tinggi selama proses pemanasan karena temperatur masih berada jauh di bawah *setpoint*.

Ketika temperatur mulai mendekati dan melewati *setpoint*, keluaran PID kemudian turun sampai 0 pada sekitar menit ke-48. Setelah itu nilai keluaran sebagian besar berada pada kondisi 0, tetapi masih terlihat beberapa kali naik kembali sebelum turun lagi.

Perubahan tersebut menunjukkan bahwa PID masih merespons perubahan temperatur yang diterima *OpenPLC*. Ketika pembacaan temperatur turun atau mendekati *setpoint*, keluaran dapat naik kembali. Ketika temperatur kembali terbaca di atas *setpoint*, keluaran kembali turun.

Kondisi ini juga berkaitan dengan pembacaan temperatur pada *OpenPLC* yang masih dapat berubah akibat proses komunikasi *pin to pin* dan *scaling*. Karena itu, setelah proses pemanasan utama selesai, keluaran PID masih terlihat beberapa kali berubah mengikuti nilai temperatur yang diterima sistem.

Untuk menilai respon sistem pada *setpoint* 45 °C, dilakukan perhitungan parameter sebagai berikut.

a. *Rise Time*

Berdasarkan hasil pengolahan data dan mengacu pada Persamaan (2.1), titik 10% diperoleh pada waktu sekitar 9,23 menit, sedangkan titik 90% dicapai pada waktu sekitar 40,52 menit. Dari kedua waktu tersebut diperoleh *Rise Time* sebesar 31,29 menit. Nilai tersebut menunjukkan bahwa sistem membutuhkan waktu sekitar 31,29 menit untuk bergerak dari titik 10% sampai titik 90% perubahan temperatur.

b. *Overshoot*

Temperatur maksimum yang tercatat sebesar 49 °C dengan *setpoint* 45 °C. Berdasarkan Persamaan (2.2), *Overshoot* terhadap *setpoint* diperoleh sebesar 8,89%.

c. *Settling time*

Berdasarkan Persamaan (2.3), *Settling time* pada pengujian *setpoint* 45 °C dinyatakan belum tercapai sampai akhir pengujian.

d. *Error Setpoint*

Pada bagian akhir pengujian, temperatur representatif pada *OpenPLC* berada di sekitar 48 °C, sedangkan *setpoint* sebesar 45 °C. Berdasarkan Persamaan (2.4), *Error* akhir terhadap *setpoint* adalah $|45 - 48| = 3$ °C.

Setelah dilakukan analisis berdasarkan kurva respon, hasil parameter pengujian *setpoint* 45 °C dirangkum pada Tabel 4.11.

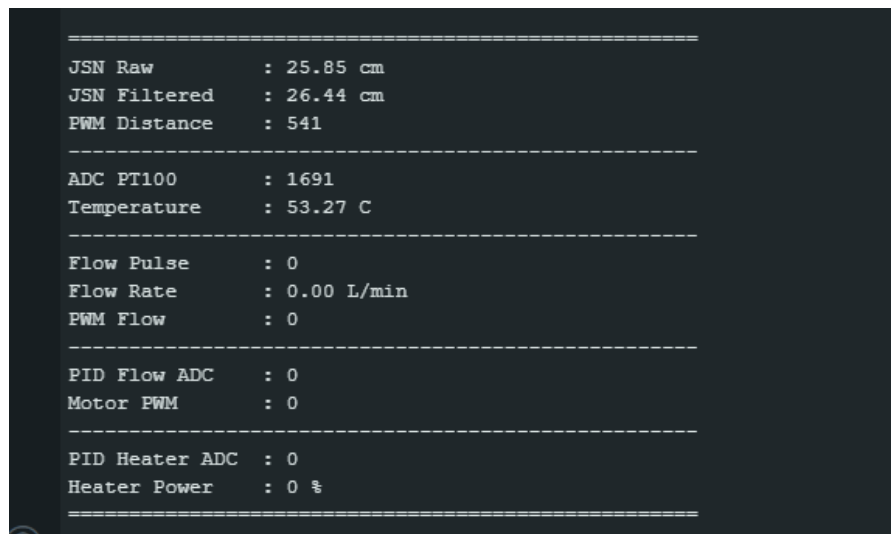
Tabel 4.11 Parameter Respon *Closed Loop Setpoint* 45 °C

<i>Setpoint</i>	<i>Rise Time</i>	<i>Settling time</i>	<i>Overshoot</i>	<i>Error setpoint</i>
45 °C	31,29 menit	Belum tercapai	8,89%	3 °C

Berdasarkan Gambar 4.53 dan Tabel 4.11, sistem mampu menaikkan temperatur sampai mencapai dan melewati *setpoint* 45 °C. Temperatur maksimum mencapai 49 °C dan bagian akhir berada sekitar 48 °C. *Settling time* belum tercapai pada pita $\pm 5\%$ *setpoint*, sedangkan *Error* akhir terhadap *setpoint* sebesar sekitar 3 °C.

Selain data yang diterima *OpenPLC*, pada akhir pengujian dilakukan pengecekan melalui Arduino IDE. Pengecekan ini dilakukan untuk melihat

temperatur yang terbaca langsung pada sisi PT100 dan kondisi *Heater* pada saat tersebut. Hasil pembacaan Arduino IDE ditunjukkan pada Gambar 4.56.



```

=====
JSN Raw      : 25.85 cm
JSN Filtered : 26.44 cm
PWM Distance : 541
=====
ADC PT100    : 1691
Temperature   : 53.27 C
=====
Flow Pulse    : 0
Flow Rate     : 0.00 L/min
PWM Flow      : 0
=====
PID Flow ADC  : 0
Motor PWM     : 0
=====
PID Heater ADC : 0
Heater Power  : 0 %
=====

```

Gambar 4.56 Pembacaan Temperatur Pada Akhir Pengujian di Arduino IDE

Berdasarkan Gambar 4.56, nilai ADC PT100 sebesar 1691 dan temperatur yang terbaca pada Arduino IDE sebesar 53,27 °C. Pada kondisi tersebut nilai PID *Heater* ADC sebesar 0 dan *Heater* Power juga sudah berada pada 0%.

Hasil tersebut menunjukkan bahwa pada saat pengecekan dilakukan *Heater* sudah tidak diberikan daya. Namun temperatur yang terbaca pada Arduino sebesar 53,27 °C masih lebih tinggi dibandingkan temperatur pada *OpenPLC* yang berada di sekitar 48 °C.

Perbedaan tersebut kembali menunjukkan adanya selisih antara nilai yang terbaca pada sisi Arduino dengan nilai yang diterima pada *OpenPLC* setelah melalui komunikasi *pin to pin* dan proses *scalling*.

Untuk melihat kondisi air secara langsung, dilakukan pengukuran menggunakan *Thermometer* digital pada tangki yang sama. Karena kondisi tangki masih tanpa proses *mixing*, temperatur pada beberapa bagian air dapat berbeda. Hasil pengukuran aktual ditunjukkan pada Gambar 4.54.



Gambar 4.57 Pengukuran Akatual di Akhir Pengujian

Berdasarkan Gambar 4.57, temperatur yang terbaca menggunakan *Thermometer* sebesar 53,7 °C. Nilai tersebut sangat dekat dengan pembacaan *Arduino IDE* sebesar 53,27 °C.

Selisih kedua pembacaan hanya sekitar 0,43 °C, sehingga masih berada di dalam rentang akurasi alat ukur pembanding sebesar ± 1 °C. Dari hasil ini terlihat bahwa pembacaan PT100 pada sisi *Arduino* cukup dekat dengan temperatur aktual air.

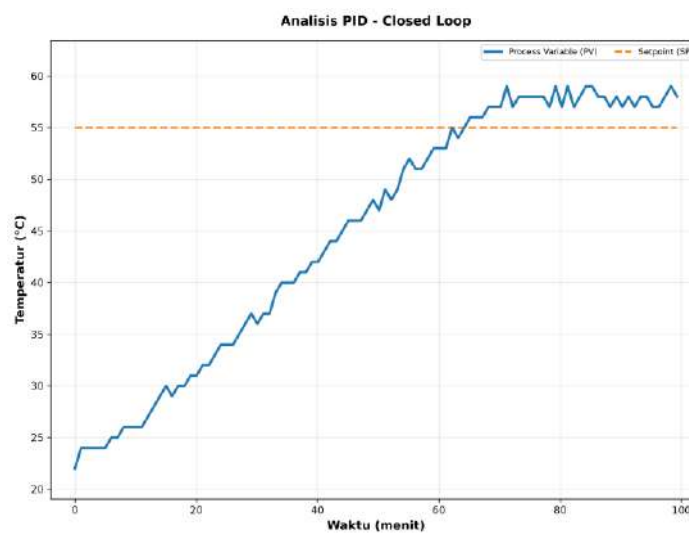
Perbedaan yang lebih besar justru terlihat pada nilai yang diterima *OpenPLC*. Pada bagian akhir grafik, *OpenPLC* menunjukkan temperatur sekitar 48 °C, sedangkan *Arduino* dan *Thermometer* sudah menunjukkan nilai di atas 53 °C. Kondisi ini menunjukkan bahwa perubahan nilai setelah data dikirim ke *OpenPLC* masih memberikan pengaruh terhadap temperatur yang digunakan sebagai process variable.

Secara keseluruhan, pada pengujian *setpoint* 45 °C, sistem sudah mampu menaikkan temperatur sampai mencapai dan melewati *setpoint*. Namun temperatur aktual air masih berada lebih tinggi dari target. *OpenPLC* menunjukkan kondisi akhir sekitar 48 °C, *Arduino IDE* membaca 53,27 °C, sedangkan *Thermometer* menunjukkan 53,7 °C. Perbedaan tersebut menunjukkan bahwa hasil pengendalian masih dipengaruhi oleh pembacaan temperatur pada jalur komunikasi dan proses *scalling*, sedangkan kondisi tangki tanpa *mixing* juga membuat distribusi temperatur belum sepenuhnya merata.

4.8.3 Pengujian Closed Loop Setpoint 55 °C

Pada percobaan ketiga sistem diberikan *setpoint* temperatur sebesar 55 °C. Kondisi tangki masih sama seperti percobaan sebelumnya, yaitu air tidak menggunakan proses *mixing*. Selama proses berlangsung, perubahan temperatur dan keluaran PID dicatat setiap satu menit sampai temperatur melewati *setpoint* dan mulai berada pada kondisi yang lebih tetap.

Data hasil pengujian kemudian digunakan untuk melihat perubahan temperatur terhadap waktu. Dari grafik dapat dilihat bagaimana temperatur bergerak dari kondisi awal menuju *setpoint* 55 °C dan bagaimana kondisi temperatur setelah nilai tersebut tercapai.

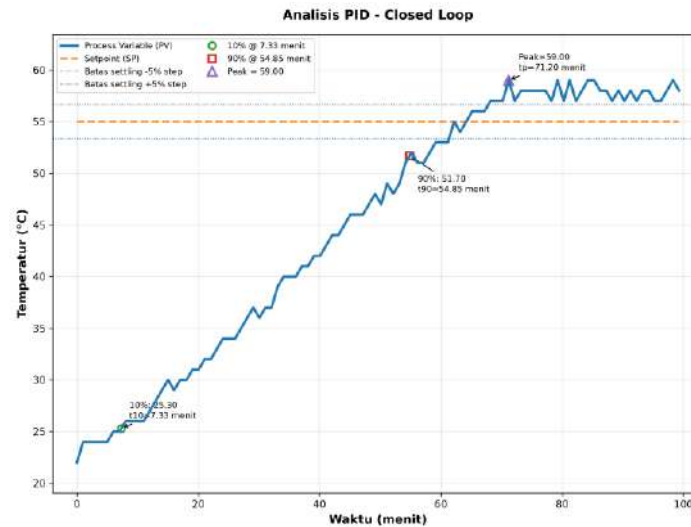


Gambar 4.58 Hasil grafik Percobaan *Setpoint* 55 °C

Berdasarkan Gambar 4.58, temperatur awal yang terbaca berada di sekitar 22 °C. Pada bagian awal pengujian temperatur meningkat secara bertahap dari sekitar 22 °C menuju 30 °C, kemudian terus naik melewati 35 °C, 40 °C, 45 °C, sampai mendekati *setpoint* 55 °C.

Temperatur mencapai daerah *setpoint* pada sekitar menit ke-62. Setelah mencapai 55 °C, temperatur masih mengalami kenaikan menjadi 56 °C, 57 °C, kemudian berada pada kisaran 58 °C. Nilai tertinggi yang tercatat selama pengujian sebesar 59 °C.

Pada bagian akhir pengujian temperatur masih berubah pada kisaran sekitar 57–59 °C, tetapi lebih banyak berada di sekitar 58 °C. Dari hasil tersebut terlihat bahwa sistem sudah mampu membawa temperatur sampai melewati *setpoint* 55 °C, tetapi temperatur akhir masih berada beberapa derajat di atas nilai yang diberikan.

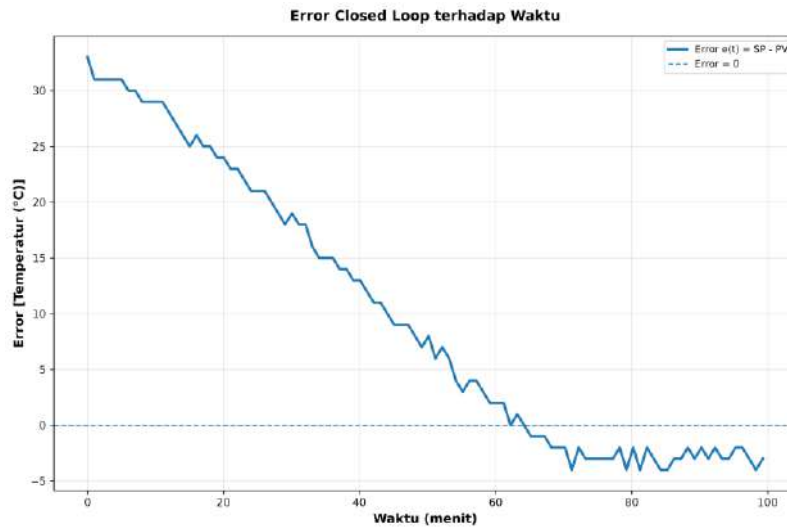


Gambar 4.59 Hasil Grafik Keterangan *Setpoint* 55 °C

Berdasarkan Gambar 4.59, temperatur awal berada di sekitar 22 °C dengan *setpoint* 55 °C. Berdasarkan perhitungan Persamaan (2.12), perubahan temperatur acuan diperoleh sebesar 33 °C. Selanjutnya, berdasarkan Persamaan (2.13) dan Persamaan (2.15), titik 10% dan 90% masing-masing diperoleh pada temperatur 25,30 °C dan 51,70 °C. Berdasarkan interpolasi yang mengacu pada Persamaan (2.16), waktu pencapaian titik 10% diperoleh sekitar 7,33 menit dan titik 90% sekitar 54,85 menit..

Pada grafik juga terlihat temperatur maksimum sebesar 59 °C yang terjadi pada sekitar 71,20 menit. Setelah temperatur mencapai nilai maksimum tersebut, pembacaan turun kembali dan selanjutnya bergerak pada kisaran sekitar 57–59 °C sampai akhir pengujian.

Selain respons temperatur, perubahan *Error* diamati dengan menghitung selisih *setpoint* 55 °C terhadap temperatur yang terbaca sesuai Persamaan (2.10). Hasil perubahan *Error* ditunjukkan pada Gambar 4.60.



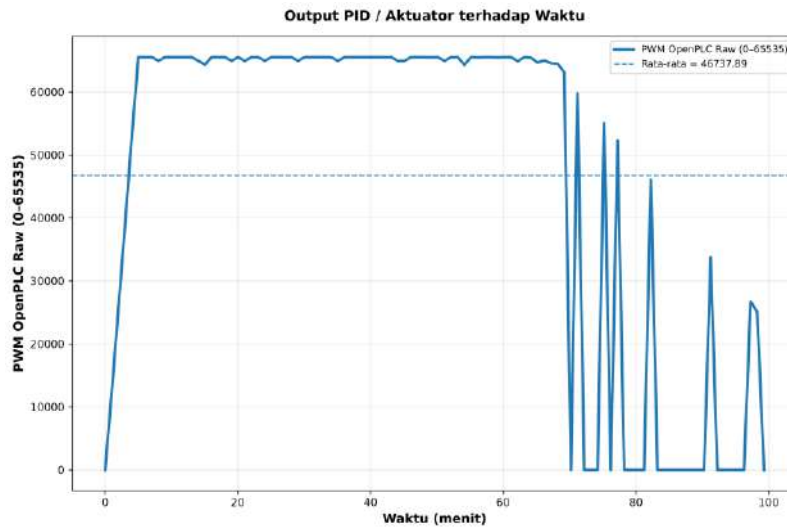
Gambar 4.60 Hasil Grafik *Error Setpoint* Terhadap Aktual Temperatur

Berdasarkan Gambar 4.60, pada awal pengujian nilai *Error* berada di sekitar 33 °C karena temperatur awal masih berada jauh di bawah *setpoint*. Seiring proses pemanasan berlangsung, nilai *Error* terus menurun karena temperatur semakin mendekati 55 °C.

Ketika temperatur sudah mencapai *setpoint*, nilai *Error* berada di sekitar 0 °C. Setelah temperatur kembali meningkat melewati 55 °C, *Error* berubah menjadi negatif. Pada bagian akhir pengujian *Error* lebih banyak berada di sekitar -3 °C, sedangkan pada beberapa titik mencapai sekitar -4 °C ketika temperatur berada pada nilai tertinggi.

Perubahan *Error* tersebut sesuai dengan grafik temperatur. Pada saat temperatur masih di bawah 55 °C, *Error* bernilai positif. Setelah temperatur berada di atas *setpoint*, nilai *Error* berubah menjadi negatif karena temperatur yang dibaca sistem sudah melebihi nilai yang diinginkan.

Perubahan temperatur dan *Error* tersebut kemudian dibandingkan dengan keluaran PID selama pengujian. Grafik keluaran PID ditunjukkan pada Gambar 4.61.



Gambar 4.61 Grafik Laju PWM percobaan *Setpoint* 55°C

Berdasarkan Gambar 4.61, pada awal pengujian keluaran PID meningkat dengan cepat hingga mendekati batas maksimum *OpenPLC*. Setelah itu keluaran bertahan pada nilai yang cukup tinggi selama sebagian besar proses pemanasan karena temperatur masih berada di bawah *setpoint*.

Ketika temperatur mulai mendekati dan melewati *setpoint*, keluaran PID turun hingga mendekati 0. Setelah bagian tersebut terlihat beberapa kali keluaran kembali naik dengan nilai yang berbeda-beda, kemudian kembali turun ke 0.

Perubahan keluaran tersebut menunjukkan bahwa PID masih terus merespons temperatur yang diterima *OpenPLC*. Ketika nilai temperatur yang terbaca turun atau kembali mendekati *setpoint*, keluaran *Heater* dapat diberikan kembali. Sebaliknya, ketika temperatur kembali terbaca di atas *setpoint*, keluaran kembali diturunkan.

Kondisi tersebut juga berkaitan dengan pembacaan temperatur pada *OpenPLC* yang masih mengalami perubahan akibat proses komunikasi *pin to pin* dan *scalling*. Karena nilai tersebut digunakan sebagai process variable, perubahan pembacaan ikut terlihat pada perubahan keluaran PID setelah temperatur mulai berada di sekitar *setpoint*.

Untuk menilai respon sistem pada *setpoint* 55 °C, dilakukan perhitungan parameter sebagai berikut.

a. *Rise Time*

Berdasarkan hasil pengolahan data dan mengacu pada Persamaan (2.1), titik 10% diperoleh pada waktu sekitar 7,33 menit, sedangkan titik 90% dicapai pada

waktu sekitar 54,85 menit. Dari kedua waktu tersebut diperoleh *Rise Time* sebesar 47,52 menit. Nilai tersebut menunjukkan bahwa sistem membutuhkan waktu sekitar 47,52 menit untuk bergerak dari titik 10% sampai titik 90% perubahan temperatur.

b. *Overshoot*

Temperatur maksimum yang tercatat sebesar 59 °C dengan *setpoint* 55 °C. Berdasarkan Persamaan (2.2), *Overshoot* terhadap *setpoint* diperoleh sebesar 7,27%..

c. *Settling time*

Dengan pita toleransi $\pm 5\%$ terhadap *setpoint* 55 °C, batas *settling* berada pada 52,25–57,75 °C. Pada bagian akhir pengujian temperatur masih berubah sekitar 57–59 °C dan beberapa kali berada di atas batas atas, sehingga respons belum dapat dinyatakan menetap di dalam pita toleransi. Berdasarkan Persamaan (2.3), *Settling time* pada pengujian *setpoint* 55 °C dinyatakan belum tercapai sampai akhir pengujian.

d. *Error Akhir*

Pada kondisi akhir, temperatur yang terbaca berada di sekitar 58 °C, sedangkan *setpoint* yang diberikan sebesar 55 °C. Berdasarkan Persamaan (2.4), diperoleh *Error* sebesar 3 °C. Nilai tersebut menunjukkan bahwa pada kondisi akhir temperatur yang terbaca sistem masih berada sekitar 3 °C di atas *setpoint*.

Setelah dilakukan analisis berdasarkan kurva respon, hasil parameter pengujian *setpoint* 55 °C dirangkum pada Tabel 4.12.

Tabel 4.12 Parameter Respon *Closed Loop Setpoint* 55 °C

<i>Setpoint</i>	<i>Rise Time</i>	<i>Settling time</i>	<i>Overshoot</i>	<i>Error Setpoint</i>
55 °C	47,52 menit	Belum Tercapai	7,27%	3 °C

Berdasarkan Gambar 4.59 dan Tabel 4.12, sistem mampu menaikkan temperatur sampai mencapai dan melewati *setpoint* 55 °C. Temperatur maksimum mencapai 59 °C dan bagian akhir berada sekitar 58 °C. *Settling time* belum tercapai pada pita $\pm 5\%$ *setpoint*, sedangkan *Error* akhir terhadap *setpoint* sebesar sekitar 3 °C.

Selain data yang diterima *OpenPLC*, pada akhir pengujian dilakukan pengecekan melalui Arduino IDE. Pengecekan ini dilakukan untuk melihat

temperatur yang terbaca langsung pada sisi PT100 dan kondisi *Heater* pada saat tersebut. Hasil pembacaan Arduino IDE ditunjukkan pada Gambar 4.62.

```

=====
JSN Raw      : 21.97 cm
JSN Filtered : 22.07 cm
PWM Distance : 451
=====
ADC PT100    : 1912
Temperature   : 64.40 C
=====
Flow Pulse   : 0
Flow Rate    : 0.00 L/min
PWM Flow     : 0
=====
PID Flow ADC : 0
Motor PWM    : 0
=====
PID Heater ADC : 13
Heater Power  : 0 %
=====

```

Gambar 4.62 Pembacaan Temperatur Pada Akhir Pengujian di Arduino IDE

Berdasarkan Gambar 4.62, nilai ADC PT100 sebesar 1912 dan temperatur yang terbaca pada Arduino IDE sebesar 64,40 °C. Pada kondisi tersebut nilai PID *Heater* ADC sebesar 13, sedangkan *Heater* Power sudah terbaca 0%.

Hasil tersebut menunjukkan bahwa pada saat pengecekan dilakukan *Heater* sudah tidak memberikan daya pemanasan. Namun temperatur yang terbaca pada Arduino sebesar 64,40 °C masih lebih tinggi dibandingkan temperatur yang terlihat pada grafik *OpenPLC* yang berada di sekitar 58 °C pada bagian akhir. Perbedaan tersebut kembali menunjukkan adanya selisih antara pembacaan temperatur pada Arduino dengan nilai yang diterima pada *OpenPLC* setelah melalui komunikasi *pin to pin* dan proses *scalling*.

Untuk melihat kondisi air secara langsung, dilakukan pengukuran menggunakan *Thermometer* digital pada tangki yang sama. Kondisi tangki masih tanpa proses *mixing*, sehingga temperatur pada setiap bagian air belum tentu sama. Hasil pengukuran aktual ditunjukkan pada Gambar 4.63.



Gambar 4.63 Pengukuran Aktual *Thermometer*

Berdasarkan Gambar 4.63, temperatur air yang terbaca menggunakan *Thermometer* sebesar 66,4 °C. Nilai tersebut lebih tinggi dibandingkan pembacaan *Arduino IDE* sebesar 64,40 °C dan nilai pada *OpenPLC* yang berada di sekitar 58 °C.

Selisih antara *Thermometer* dan *Arduino* berada di sekitar 2 °C. Nilai tersebut sudah lebih besar dibandingkan akurasi alat ukur pembanding sebesar ± 1 °C. Pada percobaan ini kondisi tanpa *mixing* mempunyai pengaruh yang cukup terlihat karena panas di dalam tangki tidak tersebar secara merata. Posisi PT100 dan *Thermometer* yang membaca bagian air yang berbeda dapat menghasilkan nilai temperatur yang berbeda.

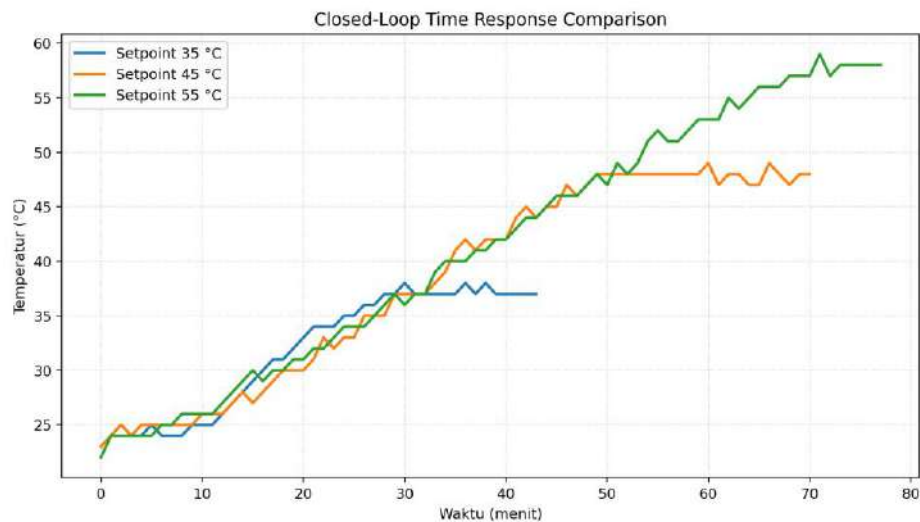
Perbedaan yang lebih besar terlihat ketika hasil tersebut dibandingkan dengan *OpenPLC*. Pada kondisi akhir *OpenPLC* masih menunjukkan temperatur sekitar 58 °C, sedangkan *Arduino* sudah membaca 64,40 °C dan *Thermometer* menunjukkan 66,4 °C. Kondisi ini menunjukkan bahwa nilai temperatur yang digunakan oleh PID masih belum sama dengan temperatur aktual air yang terbaca pada sisi pengukuran langsung.

Secara keseluruhan, pada pengujian *setpoint* 55 °C, sistem sudah mampu menaikkan temperatur sampai mencapai dan melewati *setpoint*. Namun hasil akhir masih berada di atas target. Berdasarkan pembacaan *OpenPLC* temperatur berada di sekitar 58 °C, *Arduino IDE* membaca 64,40 °C, sedangkan hasil pengukuran langsung menunjukkan sekitar 66,4 °C. Seperti pada percobaan sebelumnya, perbedaan tersebut dipengaruhi oleh pembacaan temperatur setelah melalui

komunikasi dan proses *scalling*, sedangkan kondisi tangki tanpa *mixing* juga menyebabkan temperatur air belum tersebar secara merata.

4.8.4 Perbandingan Respon Closed Loop

Setelah masing-masing pengujian Closed Loop dianalisis, ketiga kurva respons temperatur dibandingkan dalam satu grafik untuk melihat pengaruh perubahan setpoint terhadap respons sistem. Ketiga pengujian menggunakan parameter CTRL_PID yang sama, yaitu $K_P = 40,09$, $T_N = 5,59$ menit, dan $T_V = 1,40$ menit, sedangkan setpoint diubah menjadi $35\text{ }^{\circ}\text{C}$, $45\text{ }^{\circ}\text{C}$, dan $55\text{ }^{\circ}\text{C}$.



Gambar 4.64 Perbandingan Grafik Hasil Percobaan *Closed Loop*

Berdasarkan Gambar 4.64, kenaikan setpoint menyebabkan waktu respons menuju target menjadi lebih panjang. Kurva setpoint $35\text{ }^{\circ}\text{C}$ mencapai target lebih dahulu, diikuti $45\text{ }^{\circ}\text{C}$ dan $55\text{ }^{\circ}\text{C}$. Setelah target tercapai, ketiga kurva masih mengalami kenaikan dan fluktuasi sehingga Settling Time belum tercapai pada seluruh percobaan. Meskipun demikian, keluaran sistem tidak terus meningkat seperti pada Open Loop karena PID mulai mengurangi keluaran Heater ketika temperatur mendekati atau melewati setpoint.

Tabel 4.13 Perbandingan Pengujian *Closed Loop*

Setpoint	Rise Time (menit)	Temperatur +10 Menit (°C)	Error +10 Menit (%)	Overshoot (%)	Settling Time
35 °C	19,65	37	5,71	8,57	Belum tercapai
45 °C	31,29	48	6,67	8,89	Belum tercapai
55 °C	47,52	57	3,64	7,27	Belum tercapa

Berdasarkan Tabel 4.XX, Rise Time bertambah ketika setpoint dinaikkan karena sistem membutuhkan energi dan waktu pemanasan yang lebih besar untuk mencapai temperatur yang lebih tinggi. Nilai Overshoot pada ketiga percobaan berada pada rentang 7,27–8,89%, sedangkan Settling Time belum tercapai. Hasil ini menunjukkan bahwa PID telah memberikan koreksi terhadap keluaran Heater, tetapi kestabilan temperatur masih dipengaruhi oleh pembacaan process variable dan kondisi distribusi panas di dalam tangki.

4.8.5 Analisis Keseluruhan Pengujian *Open Loop*

Berdasarkan hasil pengujian *Closed Loop* pada setpoint 35 °C, 45 °C, dan 55 °C, sistem mampu menaikkan temperatur hingga mencapai dan melewati setpoint. Namun, pembacaan pada bagian akhir masih berfluktuasi dan belum menunjukkan kondisi *steady-state* yang benar-benar stabil.

Hal tersebut terutama dipengaruhi oleh pembacaan temperatur pada *OpenPLC* yang masih mengalami fluktuasi setelah data dari Arduino diteruskan melalui komunikasi *pin to pin* dan masuk ke proses *scaling*. Nilai temperatur yang diterima *OpenPLC* dapat berubah naik dan turun walaupun perubahan temperatur air secara fisik tidak secepat perubahan tersebut. Karena nilai inilah yang digunakan sebagai *process variable* oleh PID, maka keluaran kendali juga ikut merespons perubahan pembacaan tersebut.

Kondisi ini terlihat pada ketiga percobaan. *Rise Time* dan *Overshoot* masih dapat dihitung dari data yang diterima *OpenPLC*, tetapi *Settling time* tidak tercapai karena respons tidak dapat tetap berada di dalam pita $\pm 5\%$ terhadap setpoint. Oleh

karena itu, penyimpangan pada bagian akhir dinyatakan sebagai *Error* akhir terhadap *setpoint*, bukan galat kondisi tunak.

Error akhir terhadap *setpoint* berdasarkan pembacaan OpenPLC adalah 2 °C pada *setpoint* 35 °C, serta 3 °C pada *setpoint* 45 °C dan 55 °C. Jika dinyatakan relatif terhadap masing-masing *setpoint*, persentase Error akhir sebesar 5,71%, 6,67%, dan 5,45%, dengan rata-rata 5,95%. Nilai ini bukan steady-state error karena Settling time pada ketiga pengujian belum tercapai. Ringkasan hasil ditunjukkan pada Tabel 4.14.

Tabel 4.14 Tabel Error Akhir Percobaan

Setpoint	Temperatur Akhir OpenPLC	Error Akhir	Error Akhir (%)
35 °C	37 °C	2 °C	5,71%
45 °C	48 °C	3 °C	6,67%
55 °C	58 °C	3 °C	5,45%
Rata-rata	–	–	5,95%

Berdasarkan Tabel 4.14, persentase *Error* akhir *Closed Loop* terhadap *setpoint* berada pada rentang 5,45–6,67% dengan rata-rata 5,95%. Nilai ini digunakan untuk menjawab performa PID terhadap *setpoint* pada akhir masing-masing pengujian. Nilai tersebut berbeda dengan Error +10 menit, karena kondisi 10 menit setelah target pertama kali tercapai sebagai titik pembandingan yang sama antara *Open Loop* dan *Closed Loop*.

Fluktuasi pembacaan pada *OpenPLC* juga terlihat langsung pada grafik *Error*. Ketika nilai temperatur hasil *scalling* naik, *Error* akan mengecil atau bahkan berubah menjadi negatif. Sebaliknya, ketika pembacaan temperatur turun, nilai *Error* dapat naik kembali. Oleh karena itu, bentuk grafik *Error* pada bagian akhir tidak benar-benar datar, tetapi masih terlihat naik turun mengikuti perubahan nilai temperatur yang diterima sistem.

Kondisi yang sama juga terlihat pada grafik keluaran PID atau PWM. Pada saat temperatur masih jauh di bawah *setpoint*, keluaran PID berada pada nilai tinggi karena sistem masih membutuhkan pemanasan. Ketika temperatur mulai mencapai atau melewati batas atas, keluaran *Heater* turun bahkan dapat mencapai 0.

Namun pada beberapa bagian grafik terlihat nilai PWM kembali naik setelah sebelumnya turun ke 0. Perubahan tersebut terjadi karena nilai temperatur yang terbaca pada *OpenPLC* ikut naik dan turun. Ketika pembacaan sesaat turun mendekati *setpoint*, PID kembali memberikan keluaran *Heater*. Pada pembacaan berikutnya, ketika temperatur kembali terbaca lebih tinggi, keluaran diturunkan lagi.

Selain itu, pada sistem terdapat logika *Peltier* yang bekerja ketika temperatur melewati batas atas atau terjadi kondisi *Overshoot*. Karena keputusan ON dan OFF *Peltier* juga menggunakan nilai temperatur dari *OpenPLC*, pembacaan hasil *scalling* yang fluktuatif dapat menyebabkan kondisi *Peltier* ikut berubah. Pada saat temperatur terbaca melewati batas atas, *Peltier* dapat aktif. Kemudian ketika nilai pembacaan kembali turun, *Peltier* dapat kembali mati. Kondisi ini membuat respon aktuator pada bagian akhir terlihat lebih aktif walaupun perubahan temperatur air sebenarnya berlangsung lebih lambat.

Dari hasil ketiga percobaan juga terlihat adanya perbedaan antara temperatur yang terbaca oleh *OpenPLC*, temperatur pada Arduino IDE, dan temperatur aktual menggunakan *Thermometer*. Perbedaan tersebut tidak hanya dipengaruhi oleh proses komunikasi dan *scalling*, tetapi juga oleh kondisi tangki *final* yang tidak menggunakan *mixing*.

Tanpa adanya *mixing*, air di dalam tangki tidak mengalami pengadukan sehingga panas dari *Heater* tidak langsung tersebar secara merata. Bagian air yang berada dekat *Heater* dapat memiliki temperatur lebih tinggi dibandingkan bagian lain. Karena posisi PT100 dan *Thermometer* tidak berada tepat pada titik yang sama, hasil pengukurannya juga dapat berbeda.

Hal ini terlihat pada pengujian *setpoint* 45 °C dan 55 °C, di mana pembacaan Arduino IDE lebih dekat dengan hasil *Thermometer* dibandingkan nilai pada *OpenPLC*. Kondisi tersebut menunjukkan bahwa pembacaan awal PT100 sebenarnya masih cukup mendekati kondisi aktual pada titik pengukuran, sedangkan perbedaan yang lebih besar mulai terlihat setelah data diteruskan menuju *OpenPLC*.

Secara keseluruhan, hasil pengujian menunjukkan bahwa sistem PID sudah dapat memberikan respon terhadap perubahan *setpoint*, tetapi kestabilan temperatur

belum dapat dicapai dengan baik. Penyebab utamanya bukan hanya pada parameter PID, tetapi juga pada kestabilan nilai temperatur yang digunakan sebagai process variable. Selama pembacaan hasil *scalling* pada *OpenPLC* masih naik turun, PID akan terus merespons perubahan tersebut sehingga keluaran *Heater* dan *Peltier* juga ikut berubah.

Selain itu, kondisi tangki tanpa *mixing* menyebabkan temperatur aktual air tidak merata, sehingga hasil pengukuran di beberapa titik dapat menunjukkan nilai yang berbeda. Dengan demikian, hasil *Closed Loop* pada penelitian ini lebih menunjukkan bahwa sistem sudah mampu melakukan pemanasan dan merespons *setpoint*, tetapi kondisi steady-state yang benar-benar stabil belum dapat diperoleh karena pembacaan temperatur pada *OpenPLC* masih fluktuatif dan distribusi panas di dalam tangki belum merata.

4.9 Perbandingan Open loop dan Closed Loop

Perbandingan sistem Open Loop dan Closed Loop dilakukan untuk mengetahui perubahan karakteristik respons sebelum dan setelah pengendali PID diterapkan. Pada pengujian Open Loop, Heater diberikan masukan tetap sebesar 69% tanpa mekanisme koreksi berdasarkan temperatur yang telah dicapai. Data Open Loop tersebut sebelumnya digunakan untuk memperoleh karakteristik plant dan parameter awal PID. Pada pengujian Closed Loop, *OpenPLC* menerima temperatur sebagai process variable, membandingkannya dengan *setpoint*, kemudian Function Block CTRL_PID menentukan keluaran kendali menggunakan parameter $K_P = 40,09$, $T_N = 5,59$ menit, dan $T_V = 1,40$ menit.

Agar perbandingan dilakukan pada rentang temperatur yang sama, nilai 35 °C, 45 °C, dan 55 °C digunakan sebagai target *setpoint* pada kurva Open Loop dan sebagai *setpoint* pada pengujian Closed Loop. Pada Open Loop, ketiga nilai tersebut hanya menjadi acuan analisis dan tidak digunakan untuk mengubah masukan Heater. Dengan demikian, pengujian tetap bersifat Open Loop karena Heater tetap bekerja pada masukan 69%. Hasil analisis Open Loop pada masing-masing target telah disajikan pada Subbab 4.7.4, 4.7.5, dan 4.7.6, kemudian dibandingkan dengan hasil Closed Loop pada target yang sama.

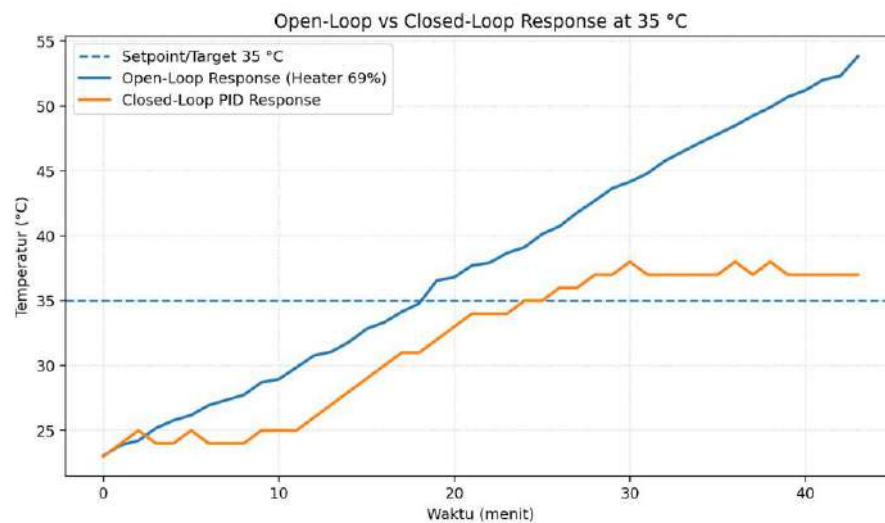
Penggunaan PID pada penelitian ini ditujukan untuk memberikan mekanisme koreksi berdasarkan selisih antara *setpoint* dan process variable. Oleh karena itu,

evaluasi tidak hanya dilakukan berdasarkan kecepatan kenaikan temperatur, tetapi juga berdasarkan besar penyimpangan temperatur setelah target tercapai. Rise Time dihitung menggunakan Persamaan (2.1), sedangkan Error temperatur dihitung menggunakan Persamaan (2.4). Nilai Error kemudian dinyatakan juga dalam bentuk persentase terhadap target untuk mempermudah perbandingan performa pada setiap temperatur.

Perbandingan berikut disajikan untuk masing-masing target temperatur dengan format grafik Open Loop dan Closed Loop pada satu bidang yang sama. Garis target digunakan sebagai acuan untuk melihat perbedaan kecepatan respons dan penyimpangan temperatur setelah target tercapai.

4.9.1 Perbandingan Setpoint 35°C

Perbandingan pertama dilakukan pada target temperatur 35 °C. Pada kurva *Open Loop*, respons dianalisis dari temperatur awal 23,06 °C menuju target 35 °C dengan masukan *Heater* tetap 69%. Pada pengujian *Closed Loop*, sistem diberikan setpoint 35 °C dengan temperatur awal sekitar 23 °C. Setelah *setpoint* tercapai, keluaran PID mulai berkurang mengikuti perubahan *process variable*.



Gambar 4.65 Grafik Perbandingan *Open Loop* dengan *Closed Loop* Setpoint 35°C

Berdasarkan Gambar 4.65, respons *Open Loop* mencapai target 35 °C lebih cepat karena *Heater* terus menerima masukan tetap sebesar 69%. Setelah temperatur mencapai target, pemanasan tetap berlangsung sehingga temperatur terus meningkat dan menjauhi target. Pada *Closed Loop*, temperatur mencapai

setpoint lebih lambat, tetapi keluaran Heater dapat dikurangi oleh PID ketika process variable mendekati atau melewati setpoint.

Tabel 4.15 Perbandingan Akurasi

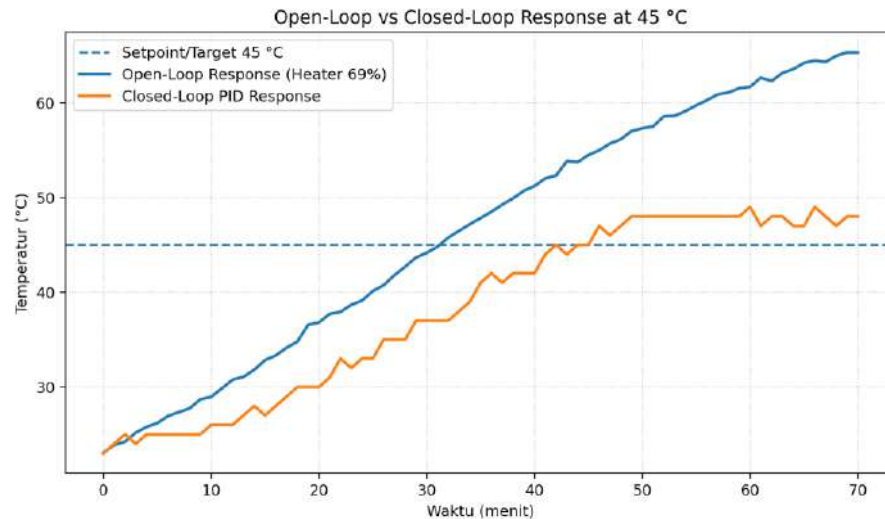
Sistem	Rise Time (menit)	Temperatur		Error (%)	Akurasi terhadap target (%)
		10 menit setelah target (°C)	Error (°C)		
<i>Open Loop</i>	14,55	43,66	8,66	24,74	75,26
<i>Closed Loop</i>	19,65	37,00	2,00	5,71	94,29
PID					
Selisih	+5,10	-6,66	-6,66	Error turun 19,03 poin	Akurasi naik 19,03 poin

Berdasarkan Tabel 4.15, *Rise Time Open Loop* pada target 35 °C sebesar 14,55 menit, sedangkan *Closed Loop* sebesar 19,65 menit. Dengan demikian, *Closed Loop* membutuhkan waktu sekitar 5,10 menit lebih lama atau 35,03% lebih besar dibandingkan *Open Loop*. Hasil ini menunjukkan bahwa penerapan PID pada kondisi pengujian ini tidak mempercepat proses pemanasan menuju target.

Perbedaan utama terlihat setelah target tercapai. 10 menit setelah target 35 °C pertama kali dicapai, temperatur *Open Loop* telah meningkat menjadi 43,66 °C sehingga Error terhadap target berdasarkan Persamaan (2.4) sebesar 8,66 °C atau 24,74%. Pada *Closed Loop*, temperatur berada sekitar 37 °C sehingga Error sebesar 2 °C atau 5,71%. Dengan acuan kedekatan terhadap target, nilai tersebut setara dengan 75,26% pada *Open Loop* dan 94,29% pada *Closed Loop*. Dengan demikian, PID mengurangi persentase *Error* pada kondisi pembanding tersebut sekitar 76,91%.

4.9.2 Perbandingan Setpoint 45°C

Perbandingan kedua dilakukan pada target temperatur 45 °C. Pada data *Open Loop*, temperatur melewati target 45 °C pada sekitar menit ke-33 dan terus meningkat karena *Heater* tetap menerima masukan 69%. Pada *Closed Loop*, temperatur awal berada sekitar 23 °C dan mencapai *setpoint* 45 °C pada sekitar menit ke-42, kemudian keluaran PID menyesuaikan berdasarkan *Error* temperatur.



Gambar 4.66 Grafik Perbandingan *Open Loop* dengan *Closed Loop* Setpoint 45°C

Berdasarkan Gambar 4.66, *Open Loop* kembali menunjukkan respons menuju target yang lebih cepat. Namun, setelah target terlewati, temperatur *Open Loop* terus meningkat. Pada *Closed Loop*, sistem memerlukan waktu lebih panjang untuk mencapai setpoint, tetapi keluaran Heater tidak lagi diberikan secara tetap karena CTRL_PID menyesuaikan keluaran terhadap perubahan *process variable*.

Tabel 4.16 Perbandingan Akurasi

Sistem	Rise Time (menit)	Temperatur			Akurasi terhadap target (%)
		10 menit setelah target (°C)	Error (°C)	Error (%)	
<i>Open Loop</i>	24,99	52,32	7,32	16,27	83,73
<i>Closed Loop</i> PID	31,29	48,00	3,00	6,67	93,33
Selisih	+6,30	-4,32	-4,32	Error turun 9,60 poin	Akurasi naik 9,60 poin

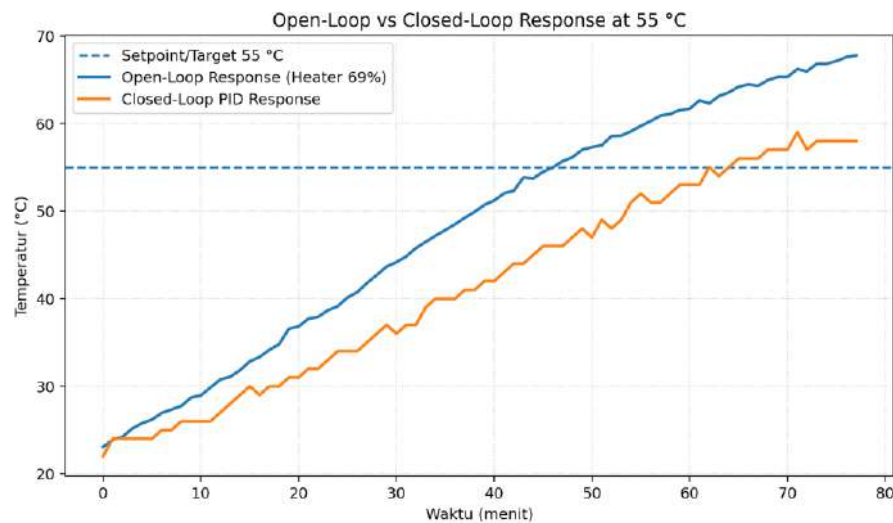
Berdasarkan Tabel 4.16, *Rise Time Open Loop* pada target 45 °C sebesar 24,99 menit dan *Closed Loop* sebesar 31,29 menit. *Closed Loop* membutuhkan waktu sekitar 6,30 menit lebih lama atau 25,22% lebih besar dibandingkan *Open Loop*.

10 menit setelah target 45 °C pertama kali dicapai, temperatur *Open Loop* telah mencapai 52,32 °C sehingga *Error* terhadap target sebesar 7,32 °C atau 16,27%. Pada *Closed Loop*, temperatur berada sekitar 48 °C sehingga *Error* sebesar 3 °C

atau 6,67%. Kedekatan respons terhadap target menjadi 83,73% pada *Open Loop* dan 93,33% pada *Closed Loop*. Persentase *Error* pada kondisi pembandingan tersebut berkurang sekitar 59,02% setelah PID diterapkan.

4.9.3 Perbandingan Setpoint 55°C

Perbandingan ketiga dilakukan pada target temperatur 55 °C. Pada data *Open Loop*, temperatur pertama kali melewati 55 °C pada sekitar menit ke-48 dan masih terus meningkat setelah target tercapai. Pada *Closed Loop*, temperatur awal berada sekitar 22 °C dan mencapai setpoint 55 °C pada sekitar menit ke-62. Setelah *setpoint* tercapai, keluaran PID mulai berkurang dan *Heater* dapat dimatikan ketika temperatur berada di atas target.



Gambar 4.67 Grafik Perbandingan *Open Loop* dengan *Closed Loop Setpoint 55°C*

Berdasarkan Gambar 4.67, pola respons masih sama dengan dua target sebelumnya. *Open Loop* mencapai target lebih cepat, tetapi tidak mempunyai mekanisme koreksi untuk membatasi kenaikan temperatur setelah target tercapai. *Closed Loop* membutuhkan waktu lebih panjang, tetapi PID dapat mengurangi keluaran *Heater* berdasarkan kondisi temperatur yang diterima *OpenPLC*.

Tabel 4.17 Perbandingan Akurasi

Sistem	Rise Time (menit)	Temperatur		Error (%)	Akurasi terhadap target (%)
		10 menit setelah target (°C)	Error (°C)		
<i>Open Loop</i>	35,65	60,88	5,88	10,69	89,31
<i>Closed Loop</i> PID	47,52	57,00	2,00	3,64	96,36
Selisih	+11,87	-3,88	-3,88	Error turun 7,05 poin	Akurasi naik 7,05 poin

Berdasarkan Tabel 4.17, *Rise Time Open Loop* pada target 55 °C sebesar 35,65 menit, sedangkan *Closed Loop* sebesar 47,52 menit. *Closed Loop* membutuhkan waktu sekitar 11,87 menit lebih lama atau 33,30% lebih besar dibandingkan *Open Loop*.

10 menit setelah target 55 °C pertama kali dicapai, temperatur *Open Loop* sebesar 60,88 °C sehingga *Error* terhadap target sebesar 5,88 °C atau 10,69%. Pada *Closed Loop*, temperatur berada sekitar 57 °C sehingga *Error* sebesar 2 °C atau 3,64%. Kedekatan respons terhadap target menjadi 89,31% pada *Open Loop* dan 96,36% pada *Closed Loop*. Persentase *Error* berkurang sekitar 65,99% pada kondisi pembandingan tersebut setelah PID diterapkan.

4.9.4 Analisis Perbandingan Open Loop dan Closed Loop

Untuk melihat perubahan kecepatan respons secara keseluruhan, nilai Rise Time pada ketiga target dibandingkan seperti ditunjukkan pada Gambar 4.64. Perhitungan Rise Time tetap menggunakan Persamaan (2.1) yang telah digunakan pada analisis respons sebelumnya.

Tabel 4.18 Analisis Hasil Keseluruhan

Target	Rise Time Open Loop (menit)	Rise Time Closed Loop (menit)	Selisih (menit)	Perubahan terhadap Open Loop (%)	Target
35 °C	14,55	19,65	+5,10	+35,03	35 °C
45 °C	24,99	31,29	+6,30	+25,22	45 °C
55 °C	35,65	47,52	+11,87	+33,30	55 °C

Berdasarkan Tabel 4.11, Rise Time Closed Loop lebih besar pada seluruh target temperatur. Hal tersebut disebabkan Open Loop memberikan Heater 69% secara tetap sehingga energi pemanasan terus diberikan tanpa koreksi. Pada Closed Loop, keluaran Heater berubah mengikuti Error dan process variable sehingga proses menuju target tidak selalu berlangsung dengan masukan tetap.

Selanjutnya, kemampuan kedua sistem dalam menjaga temperatur di sekitar target dibandingkan menggunakan Error pada 10 menit setelah target pertama kali tercapai. Nilai Error absolut tetap ditentukan menggunakan Persamaan (2.4), kemudian hasilnya dinyatakan dalam persentase terhadap target untuk memudahkan perbandingan antar setpoint.

Tabel 4.19 Error dan Akurasi Keseluruhan

Target	Error Open Loop (%)	Error Closed Loop (%)	Penurunan Error (%)	Akurasi Open Loop (%)
35 °C	24,74	5,71	76,91	75,26
45 °C	16,27	6,67	59,02	83,73
55 °C	10,69	3,64	65,99	89,31

Berdasarkan Tabel 4.12, rata-rata persentase Error Open Loop pada kondisi 10 menit setelah target tercapai sebesar 17,23%, sedangkan Closed Loop sebesar 5,34%. Dengan demikian, penerapan PID menurunkan rata-rata persentase Error sekitar 69,02%. Nilai kedekatan respons terhadap target meningkat dari rata-rata 82,77% pada Open Loop menjadi 94,66% pada Closed Loop.

Hasil keseluruhan menunjukkan adanya perbedaan antara kecepatan respons dan kemampuan pengendalian terhadap target. Open Loop mempunyai Rise Time

yang lebih cepat karena Heater terus bekerja pada masukan tetap 69%, tetapi temperatur terus meningkat setelah target tercapai. Closed Loop mempunyai Rise Time yang lebih panjang, tetapi mampu menghasilkan penyimpangan temperatur yang lebih kecil setelah target tercapai karena keluaran Heater dapat dikoreksi oleh PID.

Dengan demikian, alasan penggunaan PID pada penelitian ini bukan semata-mata untuk mempercepat proses pemanasan, melainkan untuk memberikan mekanisme umpan balik sehingga keluaran Heater dapat disesuaikan berdasarkan kondisi temperatur. Walaupun Settling Time pada ketiga pengujian Closed Loop masih belum tercapai, hasil perbandingan menunjukkan bahwa respons setelah PID diterapkan lebih dekat terhadap target dibandingkan kondisi Open Loop. Kinerja tersebut masih dipengaruhi oleh fluktuasi process variable akibat komunikasi pin to pin, proses scalling, serta distribusi temperatur di dalam tangki yang belum merata karena tidak menggunakan mixing.

4.10 Evaluasi Implementasi dan Keterbatasan *OpenPLC* ESP32

Berdasarkan seluruh proses perancangan dan pengujian yang telah dilakukan, *OpenPLC* berbasis ESP32 dapat digunakan untuk menjalankan logika pengendalian pada sistem pengaturan temperatur yang dibuat. Program yang dirancang dapat dijalankan pada ESP32, menerima nilai *process variable*, membandingkannya dengan *setpoint*, menjalankan *Function Block* CTRL_PID, dan menghasilkan nilai keluaran pengendalian. Pada pengujian yang dilakukan, keluaran tersebut dapat digunakan untuk memberikan perubahan perintah pada Heater sesuai kondisi temperatur yang diterima *OpenPLC*. Dengan demikian, dari sisi eksekusi program, logika pengendalian yang dibuat dapat dijalankan pada perangkat keras ESP32.

Meskipun demikian, selama proses implementasi ditemukan beberapa keterbatasan yang memengaruhi rancangan akhir sistem. Pada konfigurasi *OpenPLC* ESP32 yang digunakan dalam penelitian ini, pembacaan sensor PT100 melalui RTD *Transmitter* belum dapat diterapkan secara langsung sesuai kebutuhan sistem. Kondisi tersebut menyebabkan pembacaan sensor dilakukan menggunakan ESP32 tambahan yang diprogram melalui Arduino IDE. ESP32 Arduino IDE digunakan untuk membaca dan mengolah data awal dari PT100, sedangkan ESP32 *OpenPLC* difokuskan untuk menjalankan logika pengendalian. Pembagian tersebut

membuat sistem memerlukan jalur pertukaran data antara kedua ESP32 sebelum nilai temperatur dapat digunakan sebagai *process variable* pada *OpenPLC*.

Komunikasi antar ESP32 pada penelitian ini dilakukan menggunakan metode *pin to pin*. Data direpresentasikan dalam bentuk PWM, kemudian dilewatkan melalui RC *low pass filter* untuk menghasilkan tegangan yang dapat dibaca kembali menggunakan ADC. Setelah itu dilakukan proses *scaling* agar nilai yang diterima dapat digunakan oleh sistem. Hasil pengujian menunjukkan bahwa jalur tersebut mampu meneruskan perubahan nilai dari kondisi minimum sampai maksimum. Namun, nilai pada sisi penerima tidak selalu sama dengan nilai pada sisi pengirim karena data mengalami beberapa tahap konversi. Kondisi tersebut menjadi salah satu sumber selisih pembacaan yang perlu diperhatikan dalam implementasi sistem.

Pengaruh jalur komunikasi tersebut juga terlihat pada pengujian temperatur. Nilai temperatur yang terbaca pada Arduino IDE, OpenPLC, dan termometer tidak selalu sama. Sebagai contoh, pada pengujian setpoint 35 °C, OpenPLC menunjukkan temperatur sekitar 37 °C, Arduino IDE membaca 40,58 °C, sedangkan termometer menunjukkan 42,1 °C. Selain dipengaruhi oleh komunikasi dan proses *scaling*, perbedaan tersebut juga dipengaruhi oleh kondisi tangki yang tidak menggunakan *mixing*, sehingga temperatur air belum merata pada seluruh bagian tangki. Karena PID menggunakan temperatur pada OpenPLC sebagai *process variable*, perbedaan pembacaan tersebut secara langsung memengaruhi keluaran pengendalian.

Selama proses penelitian juga ditemukan adanya perbedaan penggunaan beberapa fungsi pada versi *OpenPLC* yang digunakan. Beberapa fungsi yang dapat digunakan pada versi tertentu mengalami perubahan setelah dilakukan pembaruan, sedangkan beberapa fitur yang tersedia sebelumnya tidak selalu ditemukan atau dapat digunakan dengan cara yang sama pada versi berikutnya. Kondisi tersebut membuat beberapa bagian program dan konfigurasi perlu disesuaikan kembali selama proses pengembangan. Oleh karena itu, versi *OpenPLC* yang digunakan perlu diperhatikan dan didokumentasikan ketika sistem akan direplikasi atau dikembangkan kembali.

Dari hasil tersebut dapat diketahui bahwa penerapan *OpenPLC* berbasis ESP32 pada sistem pengaturan temperatur tidak hanya bergantung pada keberhasilan

logika PID yang dibuat. Integrasi pembacaan sensor, metode komunikasi antarperangkat, proses *scaling*, kompatibilitas versi perangkat lunak, dan kondisi fisik plant juga memberikan pengaruh terhadap hasil akhir pengendalian. Pada penelitian ini, *OpenPLC* sudah dapat menjalankan logika pengendalian yang dirancang, tetapi hasil *closed-loop* belum mampu mempertahankan temperatur secara stabil pada *setpoint*. Dengan demikian, *OpenPLC* berbasis ESP32 dapat digunakan sebagai perangkat untuk menjalankan logika kontrol pada sistem yang dibuat, tetapi masih memerlukan penyesuaian pada sisi integrasi perangkat keras dan pertukaran data agar hasil pengendalian dapat lebih konsisten.