

BAB II DASAR TEORI

2.1 Tinjauan Pustaka

Pada bab ini akan dijelaskan tinjauan pustaka dari hasil studi literatur untuk sebagai acuan dan dasar teori untuk merancang sistem pengaturan temperatur air menggunakan *Heater* dan *Peltier* menggunakan *OpenPLC* ke ESP32 menggunakan metode PID.

Tabel 2.1 Tinjauan Pustaka

No	Judul Penelitian	Keterangan	Target Pembaharuan
1.	<i>Conceptual Design of Heating Power Control System Using PID</i> (Valentina et al., 2025)	Penelitian ini membahas sistem kontrol temperatur menggunakan metode PID untuk mengatur daya pemanas agar temperatur air tetap stabil pada nilai <i>setpoint</i> .	Penelitian ini menerapkan pengendalian temperatur menggunakan OpenPLC berbasis ESP32 sebagai perangkat yang menjalankan logika kontrol PID. Berbeda dengan penelitian sebelumnya yang menggunakan PID untuk mengatur daya pemanas, penelitian ini juga mengevaluasi proses implementasi OpenPLC pada sistem temperatur secara langsung, termasuk integrasi sensor, aktuator, dan pertukaran data antarperangkat.

No	Judul Penelitian	Keterangan	Target Pembaharuan
2.	<i>PID Temperature Control of Demineralized Water Tank</i> (Mahmood et al., 2018)	Penelitian ini membahas pengendalian temperatur pada tangki air <i>demineralisasi</i> menggunakan metode PID. Parameter PID ditentukan menggunakan metode tuning <i>Ziegler-Nichols</i> dan <i>Tyreus-Luyben</i> untuk mendapatkan performa sistem yang optimal. Hasil penelitian menunjukkan bahwa kontrol PID mampu menghasilkan respon sistem yang lebih baik dibandingkan kontrol PI dalam menjaga kestabilan temperatur air pada tangki.	Penelitian ini menggunakan metode Ziegler–Nichols Process-Reaction Curve berdasarkan respons <i>open-loop</i> sistem pemanas untuk memperoleh parameter awal PID, kemudian parameter tersebut diterapkan pada Function Block CTRL_PID di OpenPLC dan diuji pada kondisi <i>closed-loop</i> . Jadi fokus pembaruannya bukan menjamin sistem lebih stabil, tetapi menerapkan hasil tuning pada OpenPLC dan mengevaluasi respons yang dihasilkan
3.	<i>Design of Water Heater Temperature Control System using PID Control</i> (Rahmadini et al., 2023)	Penelitian ini membahas sistem kontrol temperatur pada pemanas air listrik menggunakan metode PID. Sistem dirancang untuk menjaga temperatur air agar tetap stabil pada nilai <i>setpoint</i> dengan respon sistem yang baik. Hasil penelitian menunjukkan bahwa penggunaan PID mampu mengurangi <i>Overshoot</i> serta menghasilkan <i>steady state Error</i> yang kecil.	Penelitian ini mengembangkan sistem pengaturan temperatur dengan Heater sebagai aktuator utama dan Peltier sebagai aktuator pendukung ketika temperatur melewati batas yang ditentukan. Peltier digunakan untuk membantu menurunkan temperatur, sedangkan logika PID dijalankan menggunakan OpenPLC berbasis ESP32.

No	Judul Penelitian	Keterangan	Target Pembaharuan
4.	Sistem Kontrol Temperatur Air Menggunakan Pengendali PID dan Volume Air pada Tangki Pemanas Air Berbasis Arduino UNO (Megido & Ariyanto, 2016)	Penelitian ini membahas sistem kontrol temperatur air pada tangki pemanas menggunakan metode PID berbasis mikrokontroler Arduino. Sensor temperatur digunakan untuk membaca temperatur air dan mengontrol elemen pemanas agar temperatur sesuai dengan <i>setpoint</i> yang diinginkan.	Penelitian ini mengembangkan sistem kontrol temperatur yang sebelumnya berbasis Arduino menjadi sistem dengan pembagian fungsi dua ESP32, yaitu ESP32 Arduino IDE untuk pembacaan sensor dan antarmuka perangkat keras, serta ESP32 OpenPLC untuk menjalankan logika PID. Pertukaran data antara keduanya dilakukan menggunakan komunikasi pin-to-pin berbasis PWM, RC Filter, ADC, dan scaling.
5.	Rancang Bangun Kendali Temperatur Pemanas Tangki Berpengaduk Kontinyu Menggunakan PID Berbasis NI ELVIS II LabVIEW (Febiawan et al., 2020)	Penelitian ini membahas sistem pengendalian temperatur pada tangki pemanas menggunakan metode PID dengan sensor temperatur LM35. Sistem dimonitor menggunakan antarmuka GUI berbasis <i>LabVIEW</i> sehingga respon sistem dapat diamati secara <i>real time</i> .	Penelitian ini menerapkan dan mengevaluasi OpenPLC berbasis ESP32 sebagai perangkat pengendali temperatur, berbeda dengan penelitian sebelumnya yang menggunakan NI ELVIS II dan LabVIEW. Evaluasi tidak hanya dilakukan terhadap respons temperatur, tetapi juga terhadap kemampuan OpenPLC menjalankan logika PID serta keterbatasan yang ditemukan pada integrasi sensor, komunikasi antar-ESP32, dan proses <i>scaling</i>

Pada penelitian ini, sistem dirancang untuk melakukan pengaturan temperatur air menggunakan metode PID berbasis *OpenPLC* pada ESP32. Sistem menggunakan dua ESP32 dengan fungsi yang berbeda. ESP32 yang diprogram menggunakan Arduino IDE digunakan untuk membaca sensor PT100 melalui RTD *Transmitter* dan melakukan pengolahan awal data temperatur, sedangkan ESP32 yang menjalankan *OpenPLC* digunakan untuk menjalankan logika pengendalian. Penggunaan dua ESP32 dilakukan karena pada konfigurasi *OpenPLC* yang digunakan dalam penelitian ini, pembacaan sensor PT100 belum dapat diterapkan secara langsung sesuai kebutuhan sistem. Data temperatur dari ESP32 Arduino IDE kemudian dikirim menuju ESP32 *OpenPLC* melalui komunikasi *pin to pin* menggunakan sinyal PWM, RC *filter*, ADC, dan proses *scaling*.

Pada sisi *OpenPLC*, nilai temperatur yang diterima digunakan sebagai *process variable* dan dibandingkan dengan nilai *setpoint* yang diberikan melalui HMI. Selisih antara *setpoint* dan *process variable* kemudian diproses menggunakan Function Block CTRL_PID untuk menghasilkan keluaran pengendalian. Nilai keluaran tersebut diteruskan kembali ke ESP32 Arduino IDE dan digunakan untuk mengatur daya *Heater* melalui modul AC *Dimmer*. Parameter awal PID diperoleh menggunakan metode *Ziegler Nichols Process Reaction Curve* berdasarkan karakteristik respons *open loop*. Selain *Heater* sebagai aktuator utama, modul *Peltier* digunakan sebagai aktuator pendukung ketika temperatur berada di atas kondisi yang telah ditentukan. Dengan rancangan tersebut, penelitian tidak hanya melihat respons temperatur yang dihasilkan, tetapi juga bagaimana *OpenPLC* diterapkan untuk menjalankan logika pengendalian pada sistem yang menggunakan sensor dan aktuator secara langsung.

Berdasarkan studi literatur yang digunakan dalam penelitian ini, pembahasan mengenai pengendalian temperatur menggunakan metode PID sudah cukup banyak ditemukan. Namun, literatur yang secara khusus membahas penerapan *OpenPLC* berbasis ESP32 pada sistem pengendalian temperatur secara langsung masih terbatas. Sebagian penelitian terdahulu lebih banyak membahas penerapan PID menggunakan mikrokontroler, PLC konvensional, atau perangkat kontrol lainnya. Oleh karena itu, penelitian ini tidak hanya melihat hasil pengendalian temperatur, tetapi juga membahas proses penerapan *OpenPLC* pada ESP32, mulai dari integrasi

sensor, pertukaran data, pelaksanaan logika kontrol, sampai keterbatasan yang ditemukan selama sistem digunakan.

2.2 Sistem Kontrol

Sistem kendali merupakan suatu sistem yang dirancang untuk mengatur atau mempertahankan keluaran agar sesuai dengan nilai yang diinginkan. Berdasarkan ada atau tidaknya penggunaan informasi keluaran sebagai umpan balik, sistem kendali dapat dibedakan menjadi sistem kendali *open loop* dan sistem kendali *closed loop*. Kedua konfigurasi tersebut memiliki karakteristik yang berbeda dalam proses pengendalian dan penerapannya pada suatu plant. Pada penelitian ini, kedua konsep tersebut digunakan dalam tahapan pengujian sistem pengendalian temperatur air menggunakan *heater*, yaitu pengujian *open loop* untuk mengetahui karakteristik *plant* dan pengujian *closed loop* untuk mengevaluasi kinerja pengendalian temperatur[7].

2.2.1 Sistem Kontrol Loop Terbuka (*Open Loop*)

Sistem kendali *open loop* merupakan sistem pengendalian yang tindakan pengendaliannya tidak dipengaruhi oleh kondisi keluaran sistem. Pada sistem ini, keluaran tidak diukur dan tidak dikembalikan sebagai sinyal umpan balik untuk dibandingkan dengan nilai masukan atau *setpoint*. Oleh karena itu, pengendalian dilakukan berdasarkan masukan yang telah ditentukan sebelumnya tanpa melakukan koreksi berdasarkan kondisi keluaran aktual.

Pada penelitian pengendalian temperatur air menggunakan *heater*, prinsip *open loop* dapat diterapkan dengan memberikan nilai masukan tertentu kepada *heater*, misalnya melalui nilai PWM atau daya *heater* tertentu, kemudian mengamati perubahan temperatur air terhadap waktu. Selama pengujian, temperatur air hanya diamati sebagai keluaran dan belum digunakan untuk mengoreksi masukan *heater*.

Pengujian *open loop* dapat digunakan untuk mengetahui karakteristik awal plant, seperti hubungan antara masukan *heater* dengan kenaikan temperatur, waktu yang diperlukan air untuk mencapai kondisi tertentu, serta kecenderungan sistem menuju kondisi stabil. Ogata juga menjelaskan bahwa dalam proses perancangan sistem kendali, pengujian terhadap sistem *open loop* dapat dilakukan terlebih dahulu sebelum sistem dioperasikan dalam konfigurasi *closed loop*. Dalam penelitian ini, hasil pengujian *open loop* temperatur dapat digunakan sebagai dasar

untuk memahami karakteristik *plant* pemanas air sebelum diterapkan pengendalian PID[7].

2.2.2 Sistem Kontrol Loop Tertutup (*Closed Loop*)

Sistem kendali *closed loop* atau sistem kendali umpan balik merupakan sistem yang menggunakan informasi keluaran untuk menentukan tindakan pengendalian. Nilai keluaran sistem diukur oleh sensor kemudian dibandingkan dengan nilai referensi atau *setpoint*. Selisih antara keduanya menghasilkan sinyal kesalahan (*Error*) yang digunakan oleh pengendali untuk menentukan tindakan koreksi terhadap *plant*.

Pada sistem pengendalian temperatur air yang digunakan dalam penelitian, temperatur air merupakan *controlled variable*, sedangkan *heater* merupakan aktuator yang menerima sinyal kendali. Sensor temperatur mengukur temperatur aktual air, kemudian nilai tersebut dibandingkan dengan temperatur yang ditetapkan sebagai *setpoint*. Selanjutnya, *error* temperatur diproses oleh pengendali PID untuk menentukan besar sinyal kendali *heater*.

Konsep tersebut sesuai dengan contoh pengendalian temperatur yang diberikan Ogata. Pada sistem temperatur, temperatur aktual diukur, dikonversikan menjadi sinyal yang dapat diproses oleh pengendali, kemudian dibandingkan dengan temperatur yang diinginkan. Jika terdapat perbedaan antara keduanya, pengendali menghasilkan sinyal untuk mengatur *heater* sehingga temperatur bergerak menuju nilai yang diinginkan.

Keunggulan utama *closed loop* adalah adanya mekanisme umpan balik sehingga perubahan kondisi *plant* atau gangguan yang memengaruhi keluaran dapat dikoreksi oleh pengendali. Namun, sistem *closed loop* juga memiliki permasalahan kestabilan yang perlu diperhatikan. Penguatan pengendali yang terlalu besar dapat menyebabkan sistem melakukan koreksi berlebihan sehingga muncul osilasi[7].

2.3 Karakteristik Respons Sistem Kendali

Karakteristik respons sistem digunakan untuk menggambarkan perilaku keluaran ketika suatu sistem diberikan masukan. Analisis respons menjadi penting untuk mengetahui seberapa cepat sistem mencapai kondisi yang diinginkan, seberapa besar penyimpangan yang terjadi selama proses, serta seberapa dekat keluaran sistem terhadap nilai referensi setelah mencapai keadaan tunak. Ogata

membahas respons waktu sistem melalui karakteristik transien dan steady-state, yang selanjutnya dapat dinilai menggunakan beberapa parameter kinerja.

Pada penelitian ini, karakteristik respons digunakan untuk mengevaluasi kinerja sistem pengendalian temperatur air setelah diberikan *setpoint*. Parameter yang digunakan meliputi *Rise Time*, *maximum Overshoot*, *Settling time*, dan *Error* akhir terhadap *setpoint*. *Rise Time* dan *Overshoot* dihitung dari data temperatur yang diterima *OpenPLC*, sedangkan *Settling time* hanya dinyatakan tercapai apabila respons dapat masuk dan tetap berada di dalam pita toleransi yang ditentukan [7].

2.3.1 *Rise Time*

Rise Time merupakan waktu yang dibutuhkan respons untuk bergerak dari 10% hingga 90% perubahan menuju nilai referensi. Pada pengujian *open loop*, nilai referensi dapat menggunakan kondisi akhir respons, sedangkan pada pengujian *closed loop* nilai referensi menggunakan *setpoint*. Parameter ini digunakan untuk melihat kecepatan respons sistem terhadap perubahan masukan.

Nilai *rise time* dihitung menggunakan Persamaan:

$$tr = t_{90\%} - t_{10\%} \quad (2.1)$$

Keterangan:

t_r = rise time (menit)

t_{90} = waktu saat respons mencapai 90% perubahan (menit)

t_{10} = waktu saat respons mencapai 10% perubahan (menit)

Dengan t_{90} merupakan waktu ketika respons mencapai 90% perubahan dan t_{10} merupakan waktu ketika respons mencapai 10% perubahan. Semakin kecil *rise time*, semakin cepat respons bergerak menuju nilai referensi [7].

2.3.2 *Overshoot*

Overshoot merupakan kondisi ketika nilai puncak respons melebihi nilai akhir yang telah dicapai sistem. Parameter ini digunakan untuk mengetahui besarnya penyimpangan maksimum respons sebelum sistem mencapai kondisi tunak. Nilai *overshoot* juga dapat digunakan sebagai salah satu indikator kestabilan relatif dari suatu sistem.

Pada sistem pengendalian temperatur air, *overshoot* terjadi ketika temperatur air mencapai nilai yang lebih tinggi dari temperatur pada kondisi *steady state*

sebelum akhirnya kembali menuju kondisi stabil. Besarnya *overshoot* dapat dihitung menggunakan persamaan:

$$\%OS = \frac{(T_{max} - T_{ss})}{T_{ss}} \times 100\% \quad (2.2)$$

Keterangan :

%OS= persentase *overshoot* (%)

T_{max} = temperatur maksimum (°C)

TSP= temperatur *setpoint* (°C)

Dengan T_{max} merupakan temperatur maksimum yang dicapai sistem dan TSP merupakan temperatur *setpoint*. Semakin besar nilai *overshoot*, semakin jauh temperatur maksimum melewati nilai target [7].

2.3.3 *Settling time*

Settling time merupakan waktu yang dibutuhkan respons untuk masuk dan tetap berada di dalam pita toleransi terhadap nilai referensi. Pada pengujian *closed loop* penelitian ini, pita toleransi ditetapkan sebesar $\pm 5\%$ terhadap *setpoint*. Apabila respons kembali keluar dari pita tersebut, *settling time* belum dinyatakan tercapai.

Pada pengendalian temperatur air, *Settling time* ditentukan dari saat temperatur mulai masuk ke pita toleransi $\pm 5\%$ terhadap *setpoint* dan setelah itu tidak keluar lagi sampai akhir pengamatan.

Untuk batas toleransi $\pm 5\%$ terhadap *setpoint*, kondisi *settling* dievaluasi menggunakan Persamaan :

$$ts = t \text{ saat } |T(t) - T_{ss}| \leq 5\% \times T_{ss} \quad (2.3)$$

Keterangan:

t_s = *settling time* (menit)

T_t = temperatur pada waktu t (°C)

TSP= temperatur *setpoint* (°C)

5%= batas toleransi *settling time*

Dengan T_t merupakan temperatur pada waktu t dan TSP merupakan temperatur *setpoint*. *Settling time* adalah waktu paling awal ketika Persamaan (2.3) terpenuhi dan tetap terpenuhi sampai akhir pengujian. Jika syarat tersebut tidak terpenuhi, *settling time* dinyatakan belum tercapai [7].

2.3.4 Error Terhadap Setpoint

Error akhir terhadap *setpoint* digunakan untuk melihat selisih antara *setpoint* dengan temperatur yang digunakan sebagai nilai representatif pada bagian akhir pengujian. Parameter ini digunakan karena pada hasil *closed loop* temperatur masih mengalami fluktuasi sehingga belum dapat dinyatakan mencapai kondisi *steady state* yang benar-benar stabil.

Pada sistem pengendalian temperatur air, *error* akhir dihitung dari selisih absolut antara *setpoint* dan temperatur bagian akhir pengujian. Nilai ini tidak dinyatakan sebagai galat kondisi tunak, tetapi hanya digunakan untuk menunjukkan besar penyimpangan pembacaan *OpenPLC* terhadap target pada akhir pengamatan persamaan:

$$e_{akhir} = |TSP - T_{akhir}| \quad (2.4)$$

Keterangan :

e_{akhir} = *Error* akhir terhadap *setpoint* (°C)

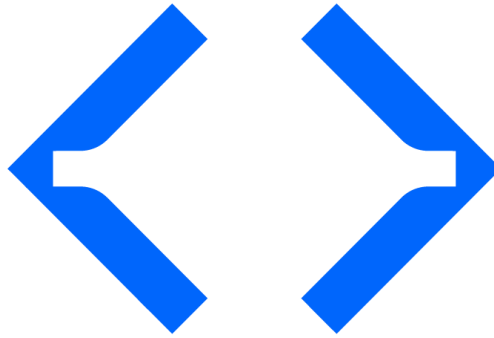
TSP = temperatur *setpoint* (°C)

T_{akhir} = temperatur representatif pada bagian akhir pengujian (°C)

Dengan TSP merupakan temperatur *setpoint* dan T_{akhir} merupakan temperatur representatif pada bagian akhir pengujian. Semakin kecil *error* akhir, semakin dekat pembacaan akhir terhadap *setpoint* [7].

2.4 OpenPLC (Open Programmable Logic Controller)

Programmable Logic Controller (PLC) merupakan perangkat kontrol yang banyak digunakan dalam sistem otomasi industri otomasi untuk mengontrol berbagai macam proses produksi, seperti dalam mengontrol temperatur, tekanan, *level* cairan dan juga mesin industri. PLC dirancang untuk menggantikan sistem kontrol berbasis *relay*, sehingga sistem kontrol dapat diprogram secara fleksible menggunakan bahasa pemrograman tertentu. Namun penggunaan PLC industri umumnya memiliki biaya yang cukup tinggi sehingga kurang efisiensi untuk digunakan dalam penelitian atau dalam pembelajaran laboratorium[8].



Gambar 2.1 *Software OpenPLC* [9]

Seiring dengan perkembangan teknologi sistem kontrol, muncul konsep *Soft PLC* atau *OpenPLC*, yaitu sistem PLC yang dijalankan pada perangkat komputer atau mikrokontroller sehingga mampu menjalankan fungsi PLC tanpa menggunakan perangkat PLC industri secara langsung. Salah satu platform *Soft PLC* yang banyak digunakan dalam penelitian maupun pembelajaran yaitu *OpenPLC*. *OpenPLC* memungkinkan implementasi sistem kontrol berbasis PLC yang terintegrasi dengan berbagai *hardware* seperti komputer, *rasberry pi*, maupun mikrokontroller lainnya. Dikarenakan sifatnya yang fleksibel dan mudah diimplementasikan dalam berbagai aplikasi sistem kontrol karena mendukung bahasa pemrograman standar PLC[10].

OpenPLC dikembangkan berdasarkan standar IEC 61131-3, standar IEC 61131-3 adalah standar internasional yang digunakan dalam pemrograman *Programmable Logic Controller* (PLC). Standar ini menentukan beberapa bahasa pemrograman yang digunakan untuk menerapkan logika kontrol dalam sistem otomasi industri. Bahasa pemrograman ini dibuat supaya sistem PLC bisa digunakan untuk mengendalikan berbagai proses di industri, seperti pengaturan temperatur, tekanan, tingkat cairan, dan pengoperasian mesin produksi. Bahasa pemrograman yang umum digunakan dalam PLC diantaranya lain *Ladder Diagram* (LD); *Function Diagram* (FBD), *Structured Text* (ST), dan *Sequential Function Chart* (SFC). Dari semuanya memiliki karakteristik yang berbeda beda, serta kegunaan yang berbeda disesuaikan dengan kebutuhan sistem yang dirancang[11].

2.4.1 *OpenPLC Runtime*

OpenPLC Runtime merupakan perangkat lunak yang berfungsi mengeksekusi program PLC yang telah dibuat menggunakan *OpenPLC Editor*. *Runtime* bertugas menjalankan siklus kerja PLC yang meliputi pembacaan data masukan (*input*

scanning), eksekusi program (*program execution*), dan pembaruan data keluaran (*output update*) secara berulang (*scan cycle*). Melalui proses tersebut, logika pengendalian yang telah dirancang dapat dijalankan pada perangkat keras yang digunakan sebagai PLC.

OpenPLC Runtime mendukung berbagai platform perangkat keras, salah satunya adalah mikrokontroler ESP32. Pada implementasi ini, program yang telah selesai dirancang dan dikompilasi menggunakan *OpenPLC Editor* selanjutnya diunggah (*upload*) secara langsung ke mikrokontroler ESP32. Dengan demikian, ESP32 menjalankan fungsi *OpenPLC Runtime* secara langsung sehingga mampu mengeksekusi program PLC tanpa memerlukan PLC komersial.

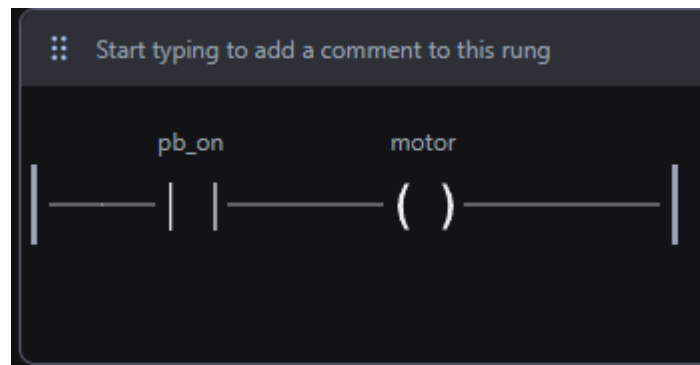
Berbeda dengan implementasi *OpenPLC* yang dijalankan pada komputer menggunakan antarmuka berbasis web (*localhost*), penelitian ini tidak menggunakan *OpenPLC Runtime* berbasis web sebagai media eksekusi program. Seluruh proses eksekusi program dilakukan langsung pada ESP32 setelah proses *upload* dari *OpenPLC Editor* selesai dilakukan. Dengan metode tersebut, ESP32 dapat menjalankan logika PLC secara mandiri, mulai dari pembacaan variabel masukan, eksekusi *Function Block Diagram* (FBD), perhitungan algoritma PID, hingga menghasilkan sinyal keluaran untuk pengendalian sistem.

Pada penelitian ini, *OpenPLC Runtime* pada ESP32 digunakan untuk menjalankan logika pengendalian laju aliran air berbasis PID. *Runtime* menerima nilai umpan balik (*feedback*) dari ESP32 antarmuka perangkat keras melalui komunikasi *pin to pin*, menghitung aksi kendali menggunakan *Function Block PID CTRL*, kemudian menghasilkan keluaran berupa nilai PID yang telah melalui proses *scaling* untuk dikirim kembali ke ESP32 antarmuka perangkat keras. Selain itu, *OpenPLC Runtime* juga mengendalikan *relay Pompa Out* dan *relay Peltier* sesuai dengan logika pengendalian yang telah dirancang.

2.4.2 Ladder Diagram (LD)

Ladder Diagram merupakan bahasa pemrograman yang paling banyak digunakan dalam sistem otomasi industri, memiliki bentuk yang menyerupai rangkaian *relay* listrik sehingga mudah untuk dipahami oleh teknisi maupun *engginer* yang terbiasa dengan sistem kontrol berbasis *relay*. *Ladder Diagram*

pertama kali digunakan pada generasi awal untuk menggantikan sistem kontrol menggunakan *relay* di industri manufaktur[12].



Gambar 2.2 *Ladder Diagram*

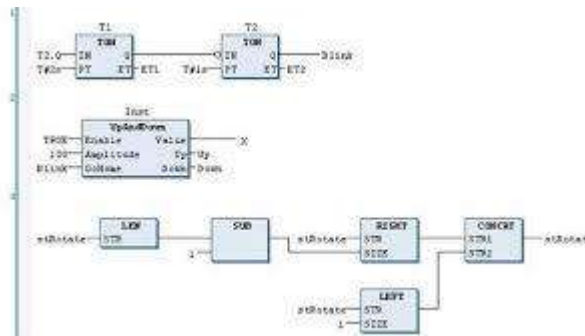
(Dokumentasi Pribadi)

Pada *Ladder Diagram*, program dituliskan dalam bentuk rung yang terdiri dari beberapa elemen logika seperti *contact* dan *coil*. *Contact* digunakan sebagai *input* logika yang merepresentasikan kondisi sensor atau saklar, sedangkan *coil* digunakan sebagai *output* yang mengaktifkan perangkat seperti motor, lampu indikator, atau aktuator lainnya. Struktur dasar *Ladder Diagram* terdiri dari dua garis vertikal yang disebut *power rail* dan beberapa garis horizontal yang disebut rung, dirancang terutama untuk memproses sinyal *Boolean* menggunakan simbol grafis yang dibatasi oleh jalur daya, yang secara keseluruhan menyerupai tangga pada diagram logika tangga[13].

Menurut penelitian oleh (Alves et al., 2014), *Ladder Diagram* merupakan bahasa pemrograman yang paling populer dalam sistem PLC karena memiliki struktur yang sederhana serta mudah dipahami oleh teknisi industri. Bahasa ini juga sangat cocok digunakan dalam sistem kontrol berbasis logika seperti sistem *start stop* motor, kontrol pompa, dan sistem pengendalian proses sederhana.

2.4.3 Function Block Diagram (FBD)

FBD merupakan bahasa pemrograman PLC yang menggunakan blok fungsi untuk merepresentasikan operasi atau proses kontrol. Pada bahasa pemrograman ini, setiap fungsi digambarkan dalam bentuk blok yang saling terhubung melalui garis sinyal yang menunjukkan aliran data antara satu blok dengan blok lainnya. FBD biasanya digunakan dalam sistem kontrol yang memiliki struktur logika yang lebih kompleks dibandingkan dengan *Ladder Diagram*.



Gambar 2.3 Function Block Diagram [14]

Pada *Function Block* Diagram, setiap blok fungsi memiliki *input* dan *output* yang dapat dihubungkan dengan blok lainnya. Blok fungsi ini dapat berupa operasi logika seperti AND dan OR, maupun fungsi kontrol seperti *timer*, *counter*, dan *PID controller*. Elemen-elemen ini memungkinkan pembentukan struktur yang terdiri dari kotak-kotak persegi panjang dengan *input*, *output*, dan pernyataan alur kontrol, yang dapat digunakan untuk melakukan ekspresi logika, ekspresi aritmatika, atau memanggil blok fungsi, sehingga setelah dihubungkan menggunakan garis alur sinyal, elemen-elemen tersebut dapat menghubungkan *output* sistem[11].

Menurut penelitian oleh (Roomi et al., 2022), penggunaan *Function Block Diagram* banyak diterapkan pada sistem kontrol proses karena memungkinkan implementasi fungsi kontrol yang lebih kompleks seperti sistem kontrol temperatur, kontrol *level* cairan, serta sistem kontrol industri lainnya.

2.4.4 Structured Text (ST)

Bahasa ST merupakan cabang dari bahasa pemrograman *Pascal*. menurut standar yang berlaku, bahasa ini tergolong sebagai salah satu bahasa tingkat tinggi. Bahasa ST dapat mengendalikan alur perintah melalui variabel, ekspresi, atau deklarasi pemilihan tipe, iterasi, rumus matematika kompleks, atau perhitungan bahasa ini mudah diterapkan dan dipahami oleh seorang *programmer*[11].

```

13 void setup() {
14   // initialize the following pin as an output:
15   pinMode(OUTPin, OUTPUT);
16   // initialize the following pin as an input:
17   pinMode(ProgramPin, INPUT);
18   // set up the LCD's number of rows and columns:
19   lcd.begin(16, 2);
20   // set cursor for messages and print Program select messages on the LCD.
21   lcd.setCursor(0,0);
22   lcd.print( ">1.Closed(ON)");
23   lcd.setCursor(0, 1);
24   lcd.print ( ">2.Open(OFF)");
25 }
26
27 void loop(){
28   // read the status of the Program Switch value:
29   ProgramStatus = digitalRead(ProgramPin);
30   // check if Program select choice is 1.ON.
31   if(ProgramStatus == HIGH) {
32     digitalWrite(OUTPin, HIGH);
33     lcd.setCursor(0,0);
34     lcd.print( ">1.Closed(ON)");
35     lcd.setCursor(0,1);
36     lcd.print ( ">2.Open(OFF) ");
37   }
38   else{
39     digitalWrite(OUTPin, LOW);
40     lcd.setCursor(0,1);
41     lcd.print ( ">2.Open(OFF)");
42     lcd.setCursor(0,0);
43     lcd.print ( ">1.Closed(ON) ");
44   }
45 }

```

Gambar 2.4 Structured Text [15]

Structured Text memungkinkan programmer untuk menuliskan logika kontrol dalam bentuk kode program yang lebih fleksibel dibandingkan dengan *Ladder Diagram* atau *Function Block Diagram*. Bahasa ini sangat cocok digunakan untuk implementasi algoritma kontrol seperti PID, perhitungan matematika, serta proses pengolahan data dalam sistem kontrol industri.

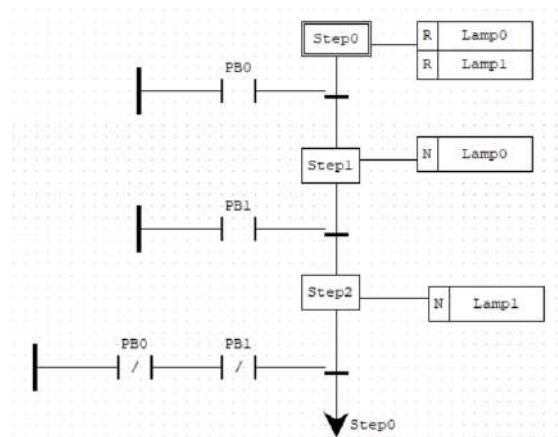
Dalam sistem kontrol temperatur air seperti pada penelitian ini, *Structured Text* dapat digunakan untuk mengimplementasikan algoritma kontrol PID yang memerlukan perhitungan matematis seperti perhitungan *Error*, integral *Error*, serta turunan *Error*. Dengan menggunakan *Structured Text*, program kontrol dapat dibuat lebih fleksibel sehingga memungkinkan pengembangan sistem kontrol yang lebih kompleks[16].

Menurut Penelitian oleh (Wang et al., 2022) menunjukkan bahwa *Structured Text* merupakan salah satu bahasa pemrograman PLC yang memiliki kemampuan tinggi dalam mengimplementasikan algoritma kontrol kompleks. Bahasa ini juga banyak digunakan dalam sistem kontrol modern karena mampu menangani perhitungan logika dan operasi matematis dengan lebih efisien.

2.4.5 Sequential Function Chart (SFC)

Sequential Function Chart dibuat untuk membagi program yang lebih kompleks menjadi unit-unit kecil yang lebih mudah dikelola. SFC secara grafis menggambarkan alur kendali antar sistem menggunakan tahap dan transisi untuk merancang proses sekuensial dan paralel. Bahasa SFC memungkinkan visualisasi

ketergantungan atau saling ketergantungan antar tahap sesuai dengan lokasi proses[11].



Gambar 2.5 *Sequential Function Chart* [17]

Pada *Sequential Function Chart*, proses kontrol dibagi menjadi beberapa langkah yang disebut step, sedangkan perpindahan antar langkah disebut transition. Setiap step merepresentasikan kondisi tertentu dalam proses kontrol, sedangkan transition merupakan kondisi yang harus terpenuhi agar sistem dapat berpindah ke langkah berikutnya.

Dalam sistem kontrol temperatur seperti penelitian ini, *Sequential Function Chart* dapat digunakan untuk menggambarkan urutan proses seperti proses pengisian tangki, proses pemanasan air, serta proses pendinginan air hingga temperatur mencapai nilai *setpoint* yang diinginkan. Dengan menggunakan SFC, alur kerja sistem kontrol dapat digambarkan secara lebih sistematis sehingga mempermudah proses perancangan dan analisis sistem kontrol.

2.5 Function Block Yang Digunakan

Pada sub bab dibawah adalah *Function Block* yang di gunakan pada penelitian saya untuk melakukan pengolahan data sensor, perhitungan algoritma PID, konversi tipe data, proses *scalling*, serta pengambilan keputusan dalam sistem pengendalian laju aliran air berbasis *OpenPLC*. *Function Block* yang digunakan meliputi PID CTRL, blok operasi aritmatika, blok pembanding (*comparator*), blok konversi tipe data, serta blok pewaktuan (*timer*).

2.5.1 Function Block PID CTRL

Function Block PID CTRL merupakan blok fungsi yang digunakan untuk mengimplementasikan algoritma *Proportional-Integral-Derivative* (PID) pada

sistem kendli berbasis *Function Block Diagram* (FBD). Blok ini menghasilkan sinyal kendali berdasarkan selisih antara nilai acuan (*setpoint*) dan nilai aktual sistem (*process variable*). Tujuan utama penggunaan PID CTRL adalah memperkecil nilai *Error* sehingga variabel proses dapat mencapai dan mempertahankan nilai *setpoint* dengan respons yang cepat serta kesalahan tunak (*steady state error*) yang kecil.

Function Block CTRL_PID menerima nilai *Setpoint* (SET), *Process Variable/actual value* (ACT), serta parameter pengendali berupa KP, TN, dan TV. Dalam penelitian ini, KP digunakan sebagai gain proporsional, TN sebagai waktu integral, dan TV sebagai waktu derivatif. Blok kemudian menghitung keluaran kendali secara internal.

Karena perhitungan PID dilakukan di dalam Function Block CTRL_PID, komponen proportional, integral, dan derivative tidak dihitung secara terpisah pada program utama. Persamaan (2.10) sampai Persamaan (2.14) tetap digunakan sebagai dasar teori untuk menjelaskan mekanisme PID, sedangkan implementasi praktis menggunakan parameter KP, TN, dan TV pada *OpenPLC*..

Pada penelitian ini, *Function Block* PID CTRL digunakan pada dua sistem pengendalian, yaitu pengendalian laju aliran air dan pengendalian temperatur air. Pada pengendalian laju aliran air, nilai *process variable* diperoleh dari pembacaan Flow Sensor YF-S201, sedangkan keluarannya digunakan sebagai referensi pengaturan kecepatan Pompa *In* melalui driver BTS7960. Pada pengendalian temperatur, nilai *process variable* diperoleh dari pembacaan sensor PT100 melalui *transmitter* 4–20 mA, sedangkan keluaran PID digunakan sebagai referensi pengaturan daya *Heater* menggunakan modul *AC Dimmer*. Dengan penggunaan dua buah *Function Block* PID CTRL tersebut, sistem mampu mengendalikan laju aliran dan temperatur secara independen sesuai dengan nilai *setpoint* yang diberikan melalui *Human Machine Interface* (HMI).

2.5.2 *Function Block* ADD

Function Block ADD merupakan blok operasi aritmatika yang digunakan untuk menjumlahkan dua atau lebih nilai masukan sehingga menghasilkan satu nilai keluaran berupa hasil penjumlahan. Blok ini mendukung berbagai tipe data numerik, baik bilangan bulat maupun bilangan pecahan.

Pada penelitian ini, *Function Block* ADD digunakan pada beberapa proses perhitungan, seperti penyesuaian nilai hasil konversi sensor maupun proses *scalling* sebelum nilai dikirim atau diproses lebih lanjut oleh sistem pengendalian.

2.5.3 *Function Block* SUB

Function Block SUB digunakan untuk melakukan operasi pengurangan antara dua nilai masukan. Blok ini menghasilkan selisih antara nilai pertama dan nilai kedua sehingga banyak digunakan pada proses perhitungan matematis maupun penyesuaian nilai hasil pembacaan sensor.

Pada penelitian ini, blok SUB digunakan dalam proses konversi data sensor serta penyesuaian rentang nilai sebelum digunakan pada proses pengendalian.

2.5.4 *Function Block* MUL

Function Block MUL merupakan blok operasi aritmatika yang digunakan untuk mengalikan dua atau lebih nilai masukan sehingga menghasilkan nilai keluaran berupa hasil perkalian.

Pada penelitian ini, *Function Block* MUL digunakan dalam proses *scalling* nilai keluaran PID serta konversi data sensor sehingga diperoleh rentang nilai yang sesuai dengan kebutuhan sistem.

2.5.5 *Function Block* DIV

Function Block DIV merupakan blok operasi aritmatika yang digunakan untuk membagi suatu nilai dengan nilai pembagi tertentu sehingga menghasilkan nilai keluaran berupa hasil pembagian.

Pada penelitian ini, *Function Block* DIV digunakan bersama *Function Block* MUL dan ADD untuk membentuk proses *scalling*, baik pada pengolahan data sensor maupun pada keluaran PID sebelum dikirimkan ke ESP32 kedua.

2.5.6 *Function Block* MOVE

Function Block MOVE digunakan untuk menyalin nilai dari suatu variabel menuju variabel lain tanpa mengubah besar nilainya. Blok ini banyak digunakan untuk memindahkan data antar variabel sehingga memudahkan proses pengolahan data pada program PLC. Pada penelitian ini, *Function Block* MOVE digunakan untuk memperbarui nilai variabel yang akan digunakan pada proses berikutnya maupun untuk meneruskan hasil perhitungan ke blok fungsi selanjutnya.

2.5.7 *Function Block Comparator*

Function Block Comparator merupakan blok logika yang digunakan untuk membandingkan dua buah nilai masukan. Hasil perbandingan berupa logika *TRUE* atau *FALSE* yang selanjutnya digunakan sebagai dasar pengambilan keputusan pada sistem kontrol.

Beberapa digunakan diantaranya GT (*Greater Than*), untuk mengetahui apakah nilai pertama lebih besar dari nilai kedua. GE (*Greater Than or Equal*), untuk mengetahui apakah nilai pertama lebih besar atau sama dengan nilai kedua. LT (*Less Than*), untuk mengetahui apakah nilai pertama lebih kecil dari nilai kedua. LE (*Less Than or Equal*), untuk mengetahui apakah nilai pertama lebih kecil atau sama dengan nilai kedua. EQ (*Equal*), untuk mengetahui apakah kedua nilai memiliki besar yang sama.

Pada penelitian ini, *Function Block Comparator* digunakan untuk menentukan kondisi pengendalian, seperti aktivasi pompa *Out*, *Peltier*, maupun logika perpindahan proses berdasarkan nilai sensor dan *setpoint*.

2.5.8 *Function Block REAL_TO_UINT*

Function Block REAL_TO_UINT digunakan untuk mengubah tipe data *REAL* menjadi *Unsigned Integer* (UINT). Proses konversi ini diperlukan ketika hasil perhitungan berupa bilangan pecahan akan digunakan oleh *Function Block* atau perangkat yang hanya menerima data bertipe bilangan bulat.

Pada penelitian ini, blok *REAL_TO_UINT* digunakan untuk mengubah hasil keluaran PID maupun hasil perhitungan lainnya sebelum diteruskan menuju proses pengiriman data antar ESP32.

2.5.9 *Function Block UINT_TO_REAL*

Function Block UINT_TO_REAL merupakan blok konversi tipe data yang digunakan untuk mengubah data bertipe *Unsigned Integer* (UINT) menjadi *REAL*. Konversi ini diperlukan ketika data masukan berupa bilangan bulat akan digunakan pada proses perhitungan yang memerlukan ketelitian dalam bentuk bilangan pecahan.

Pada penelitian ini, *Function Block* *UINT_TO_REAL* digunakan pada proses pengolahan data sensor sebelum dilakukan perhitungan menggunakan *Function Block* PID CTRL.

2.5.10 *Function Block TON (On Delay Timer)*

Function Block TON (On Delay Timer) merupakan blok pewaktuan yang menghasilkan keluaran setelah masukan aktif selama waktu tertentu yang telah ditentukan. Selama waktu yang ditetapkan belum tercapai, keluaran tetap berada pada kondisi tidak aktif. Setelah waktu tunda terpenuhi, keluaran akan berubah menjadi aktif.

Blok TON memiliki beberapa parameter utama, yaitu IN sebagai masukan logika, PT (*Preset Time*) sebagai waktu tunda yang diinginkan, ET (*Elapsed Time*) sebagai waktu yang telah berlalu, dan Q sebagai keluaran logika.

Pada penelitian ini, *Function Block TON* digunakan untuk memberikan jeda waktu pada proses pengendalian sehingga perpindahan kondisi antar proses berlangsung secara lebih teratur sesuai dengan urutan operasi sistem.

2.6 **Arduino IDE**

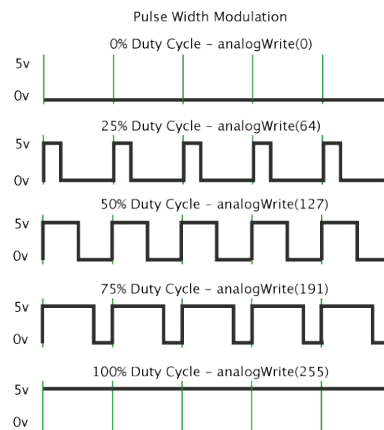
Arduino IDE (*Integrated Development Environment*) merupakan perangkat lunak yang digunakan untuk menulis, mengompilasi, dan mengunggah program ke papan mikrokontroler. Perangkat lunak ini dapat dijalankan pada beberapa sistem operasi seperti *Windows*, *Linux*, dan *macOS* serta menggunakan bahasa pemrograman berbasis C/C++. Dalam pengembangan sistem berbasis mikrokontroler, Arduino IDE digunakan untuk membuat program sesuai fungsi yang akan dijalankan oleh perangkat, kemudian program tersebut diunggah ke papan mikrokontroler agar dapat melakukan proses pembacaan masukan maupun pengendalian keluaran. Taiwo dan Ezugwu menggunakan Arduino IDE untuk menulis dan mengunggah program pada perangkat ESP8266 dan ESP32-CAM yang digunakan untuk membaca sensor serta menjalankan fungsi kendali pada sistem yang dikembangkan.

Penggunaan Arduino IDE pada ESP32 memberikan kemudahan dalam pengembangan program karena proses konfigurasi perangkat, pembacaan sensor, pengolahan data, dan pengaturan keluaran dapat dilakukan dalam satu program. Program yang telah selesai disusun terlebih dahulu melalui proses kompilasi untuk mengetahui adanya kesalahan penulisan atau konfigurasi sebelum diunggah ke mikrokontroler. Pada penelitian Taiwo dan Ezugwu, program yang dibuat melalui Arduino IDE digunakan untuk mengatur kerja sensor dan perangkat keluaran,

kemudian program tersebut diunggah ke papan mikrokontroler yang digunakan pada sistem[18].

2.7 Pulse Width Modulation

Pulse Width Modulation (PWM) merupakan teknik pengaturan rasio waktu kondisi *HIGH* terhadap satu periode sinyal. Pada penelitian ini, PWM terutama digunakan sebagai representasi data pada komunikasi *pin to pin* antar ESP32 dan pada pengaturan kecepatan pompa. Untuk pengendalian *heater* AC, PWM tidak diberikan langsung ke elemen *heater*, nilai kendali dari *OpenPLC* terlebih dahulu dikirim antar ESP32, kemudian ESP32 Arduino menerapkannya pada *AC Light Dimmer* yang bekerja dengan metode *phase angle contro*[19].



Gambar 2.6 Pulse Width Modulation [20]

Pada jalur komunikasi *pin to pin*, nilai digital keluaran *OpenPLC* direpresentasikan sebagai *duty cycle* PWM. Setelah melewati *RC Low Pass Filter*, sinyal PWM menghasilkan tegangan rata-rata yang dapat dibaca oleh ADC ESP32 penerima. Hubungan ini digunakan untuk menganalisis kesesuaian nilai yang dikirim dan diterima.

$$Duty\ Cycle = \frac{T_{on}}{T_{total}} \times 100\% \quad (2.5)$$

Keterangan :

D= duty cycle PWM (%)

Ton= lama sinyal HIGH (detik)

Ttotal= periode PWM (detik)

Hubungan antara *duty cycle* dan daya *Heater*:

$$V_{out} = D \times V_{in} \quad (2.6)$$

Keterangan :

D = duty cycle

T_{on} = lama sinyal HIGH

T_{total} = periode PWM

V_{in} = tegangan masuk

Semakin besar *duty cycle*, maka daya yang diberikan ke *heater* semakin besar sehingga temperatur air akan meningkat lebih cepat.

2.8 Komunikasi *Pin to pin*

Komunikasi *pin to pin* merupakan metode pertukaran data antar dua mikrokontroler menggunakan hubungan langsung melalui pin *input* dan *output* (GPIO) tanpa memanfaatkan media komunikasi jaringan seperti *Wi-Fi*, *Ethernet*, maupun protokol internet. Pada metode ini, sebuah mikrokontroler bertindak sebagai pengirim (*transmitter*) yang menghasilkan sinyal *digital* atau *analog* melalui pin *output*, sedangkan mikrokontroler lainnya bertindak sebagai penerima (*reciver*) yang membaca sinyal tersebut melalui pin *input*.

Pada penelitian ini, komunikasi *pin to pin* digunakan untuk menyebut pertukaran sinyal secara langsung melalui pin masukan dan keluaran antara dua ESP32. Jalur ini digunakan karena kedua ESP32 mempunyai tugas yang berbeda. ESP32 Arduino menangani pembacaan sensor dan antarmuka perangkat keras, sedangkan ESP32 *OpenPLC* menjalankan logika pengendalian. Dengan hubungan tersebut, data yang dihasilkan oleh salah satu ESP32 dapat diteruskan ke ESP32 lainnya tanpa melalui jaringan *Wi-Fi* atau *Ethernet*.

ESP32 memiliki beberapa periferal yang dapat digunakan untuk proses masukan dan keluaran, seperti GPIO, PWM, ADC, *timer*, serta DAC pada tipe tertentu. Rohman dkk. melakukan pengujian terhadap beberapa periferal mikrokontroler, termasuk ESP32, dan menunjukkan bahwa kemampuan GPIO, PWM, dan ADC menjadi bagian penting dalam pemilihan mikrokontroler untuk suatu sistem tertanam. Kemampuan tersebut memungkinkan ESP32 menghasilkan sinyal melalui pin keluaran dan membaca kembali sinyal melalui bagian ADC sesuai kebutuhan aplikasi[21].

2.8.1 Komunikasi PWM Antar ESP32

Pada penelitian ini, komunikasi antar ESP32 memanfaatkan sinyal PWM yang dikirim melalui pin GPIO. ESP32 pertama menghasilkan sinyal PWM sebagai representasi keluaran logika kontrol *OpenPLC*, sedangkan ESP32 kedua membaca *duty cycle* PWM tersebut untuk diterjemahkan menjadi nilai kontrol. Karena PWM hanya berupa sinyal digital dengan perubahan *duty cycle*, komunikasi ini memiliki beberapa keunggulan implementasi sederhana, tidak memerlukan konfigurasi protokol komunikasi, latensi sangat rendah, tidak bergantung pada jaringan internet dan mudah diintegrasikan dengan *OpenPLC*.

Namun demikian, pembacaan *duty cycle* harus dilakukan secara presisi agar nilai yang diterima sesuai dengan nilai yang dikirim. Gangguan *noise* maupun kesalahan sinkronisasi dapat menyebabkan perbedaan nilai pembacaan.

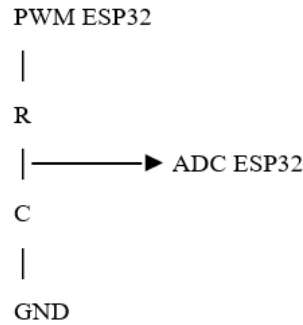
2.9 Resistor Capacitor Filter (RC filter)

RC filter merupakan rangkaian elektronika pasif yang tersusun atas resistor (R) dan kapasitor (C). Salah satu jenis *RC filter* yang paling banyak digunakan adalah *Low Pass Filter* (LPF), yaitu rangkaian yang berfungsi melewatkan sinyal berfrekuensi rendah dan meredam komponen frekuensi tinggi.

Pada sistem kendali berbasis mikrokontroler, *RC Low Pass Filter* sering dimanfaatkan untuk mengubah sinyal *Pulse Width Modulation* (PWM) menjadi tegangan analog rata-rata (*average voltage*). Dengan demikian, perangkat yang tidak memiliki keluaran DAC (*Digital to Analog Converter*) tetap dapat menghasilkan sinyal yang menyerupai tegangan analog.

Prinsip kerja *RC Low Pass Filter* didasarkan pada proses pengisian (*charging*) dan pengosongan (*discharging*) kapasitor melalui resistor. Kapasitor akan menyimpan muatan ketika sinyal berada pada kondisi HIGH dan melepaskan muatan ketika sinyal berubah menjadi LOW. Apabila frekuensi PWM cukup tinggi, perubahan tegangan pada kapasitor menjadi sangat halus sehingga menghasilkan tegangan rata-rata yang sebanding dengan nilai *duty cycle* PWM[19].

Rangkaian *RC Low Pass Filter* terdiri atas sebuah resistor yang dipasang seri dengan sumber sinyal dan sebuah kapasitor yang dihubungkan ke ground.



Pada penelitian ini, *RC low pass filter* ditempatkan pada jalur komunikasi antar ESP32. Sinyal PWM dari ESP32 pengirim dilewatkan melalui *RC filter* sebelum masuk ke ADC ESP32 penerima. Rangkaian tersebut digunakan agar PWM dapat dibaca sebagai perubahan tegangan oleh ADC. Selanjutnya, nilai hasil pembacaan ADC dikonversi kembali melalui proses *scaling* sesuai variabel yang dikirimkan.

2.9.1 Konstanta Waktu RC

Karakteristik *RC filter* ditentukan oleh konstanta waktu (τ) yang dinyatakan sebagai:

$$\tau = R \times C \quad (2.7)$$

Keterangan :

τ = konstanta waktu (detik)

R = resistansi (Ω)

C = kapasitansi (F)

Konstanta waktu menunjukkan kecepatan kapasitor dalam mengisi maupun melepaskan muatan. Nilai konstanta waktu yang terlalu kecil menyebabkan riak (*ripple*) tegangan masih cukup besar, sedangkan nilai yang terlalu besar akan memperlambat perubahan tegangan sehingga respons sistem menjadi lambat.

2.9.2 Frekuensi Cut-Off

Frekuensi Cut Off RC Low Pass Filter dihitung menggunakan persamaan:

$$f_c = \frac{1}{2\pi RC} \quad (2.8)$$

Keterangan :

f_c = *Frekuensi Cut-Off* (Hz)

R = resistor (Ω)

C = kapasitor (F)

Frekuensi Cut Off harus dipilih lebih rendah daripada frekuensi PWM sehingga komponen frekuensi tinggi dari sinyal PWM dapat diredam secara efektif dan hanya nilai rata-ratanya yang diteruskan.

2.10 Digital Low Pass Filter (Exponential Moving Average)

Selain menggunakan *RC Low Pass Filter* pada sisi perangkat keras, penelitian ini juga menerapkan penyaringan secara perangkat lunak (*software filtering*) menggunakan metode *Exponential Moving Average* (EMA). Metode ini digunakan untuk mengurangi fluktuasi nilai ADC akibat noise pada proses pembacaan sensor PT100 sebelum data dikirim menuju ESP32 *OpenPLC*.

Metode EMA bekerja dengan menggabungkan data pembacaan terbaru dengan data sebelumnya menggunakan faktor pembobot (*weighting factor*). Dengan cara tersebut, perubahan data berlangsung lebih halus tanpa menghilangkan karakteristik utama sinyal yang diukur.

Persamaan EMA dinyatakan sebagai

$$y(k) = \alpha x(k) + (1 - \alpha)y(k - 1) \quad (2.9)$$

Keterangan :

$y(k)$ = keluaran filter saat ini

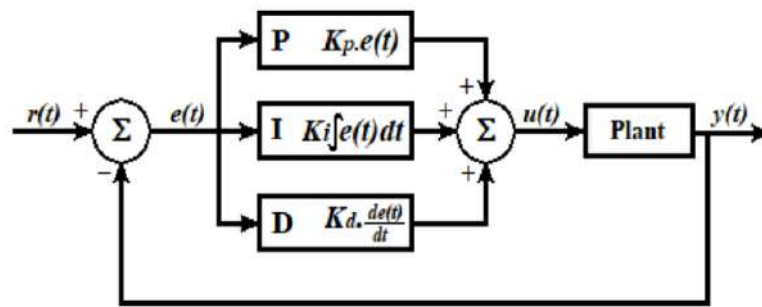
$y(k - 1)$ = keluaran filter sebelumnya

$x(k)$ = data pembacaan terbaru

α = konstanta filter ($0 < \alpha < 1$)

2.11 Metode PID (Proportional Integral Derivative)

Proportional Integral Derivative (PID) merupakan salah satu metode kontrol yang paling banyak digunakan dalam sistem kontrol industri karena memiliki struktur yang sederhana, mudah diimplementasikan, serta mampu menghasilkan performa kontrol yang stabil pada berbagai sistem proses. Metode PID bekerja dengan meminimalkan nilai *error* antara nilai keluaran sistem dengan nilai referensi atau *setpoint* melalui tiga komponen kontrol yaitu *proportional*, *integral*, dan *derivative*. PID controller banyak diterapkan pada berbagai sistem kontrol seperti kontrol temperatur, *level* cairan, tekanan, dan kecepatan motor karena kemampuannya dalam memberikan respon sistem yang cepat serta menjaga kestabilan sistem pada kondisi *steady state*[22].



Gambar 2.7 *Proportional Integral Derivative* [23]

Penggunaan PID juga banyak diterapkan dalam sistem pengaturan temperatur pada berbagai aplikasi industri seperti sistem pemanas air, sistem penukar panas, serta sistem fermentasi. Penelitian lain menunjukkan bahwa PID controller mampu mengontrol temperatur sistem secara efektif dengan menjaga temperatur tetap berada pada nilai referensi meskipun terjadi gangguan pada sistem[24].

Secara umum PID controller terdiri dari tiga komponen utama yaitu *proportional* (P), *integral* (I), dan *derivative* (D). Ketiga komponen tersebut bekerja secara bersamaan untuk menghasilkan sinyal kontrol yang digunakan untuk mengendalikan aktuator pada sistem kontrol.

2.11.1 Persamaan Umum PID

Pengendali PID terdiri dari tiga aksi pengendalian, yaitu *proportional*, *integral*, dan *derivative*. Aksi *proportional* memberikan respons yang sebanding dengan besar *error*, aksi integral mengakumulasi *error* terhadap waktu untuk mengurangi penyimpangan yang berlangsung terus menerus, sedangkan aksi *derivative* merespons laju perubahan *error* untuk membantu meredam perubahan respons yang terlalu cepat dan *overshoot*. Ketiga aksi tersebut bekerja secara bersamaan dalam menghasilkan sinyal kendali.

Nilai *Error* antara *setpoint* dan nilai aktual atau process variable dihitung menggunakan Persamaan (2.10):

$$e(t) = r(t) - y(t) \quad (2.10)$$

Keterangan:

$e(t)$ = *Error* sistem

$r(t)$ = nilai *setpoint*

$y(t)$ = nilai keluaran sistem

Karena *Function Block* CTRL_PID pada *OpenPLC* menggunakan parameter gain proporsional, waktu integral, dan waktu derivatif, persamaan PID pada penelitian ini dituliskan dalam bentuk K_p , T_i , dan T_d . Bentuk persamaan umum PID ditunjukkan pada Persamaan (2.11):

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int e(t) dt + K_p T_d \frac{de(t)}{dt} \quad (2.11)$$

Keterangan :

$u(t)$ = sinyal kontrol

K_p = konstanta *proportional*

T_i = waktu integral (menit)

T_d = waktu derivatif (menit)

$e(t)$ = *Error* sistem

Dalam implementasinya pada sistem kontrol temperatur air, nilai keluaran sistem merupakan temperatur air yang diukur oleh sensor temperatur. Nilai temperatur tersebut kemudian dibandingkan dengan nilai *setpoint* sehingga diperoleh nilai *Error* yang digunakan oleh pengendali PID untuk menghasilkan sinyal kontrol yang mengatur kerja aktuator seperti *heater* atau sistem pendingin. Penggunaan metode PID dalam sistem kontrol temperatur terbukti mampu menghasilkan respon sistem yang stabil dengan *error* yang kecil serta waktu respon yang lebih cepat dibandingkan metode kontrol sederhana[25]. Oleh karena itu metode PID banyak digunakan dalam berbagai sistem kontrol proses baik pada skala industri maupun sistem eksperimen laboratorium.

Dalam penelitian ini metode PID digunakan untuk mengontrol temperatur air pada sistem tangki dengan menggunakan *heater* sebagai elemen pemanas dan modul *peltier* sebagai elemen pendingin sehingga temperatur air dapat dijaga sesuai dengan nilai *setpoint* yang diinginkan.

2.11.2 Metode *Tuning Ziegler Nichols*

Metode *Ziegler Nichols* merupakan salah satu metode penalaan untuk menentukan parameter awal pengendali PID dari karakteristik respons plant. Pada pendekatan *process reaction curve*, plant diberi perubahan masukan pada kondisi *Open Loop* dan respons keluarannya diamati terhadap waktu. Dalen dan Di Ruscio menjelaskan bahwa metode *Ziegler Nichols Process Reaction Curve* dapat

menggunakan kemiringan maksimum respons dan waktu tunda sebagai parameter utama penalaan[26].

Pada respons proses yang dapat didekati sebagai sistem orde satu dengan waktu tunda, karakteristik tambahan dapat diperoleh dari respons step. Niemann dan Miklos menunjukkan bahwa respons orde satu mencapai sekitar 63,2% dari perubahan total setelah satu konstanta waktu, dan pendekatan garis singgung dapat digunakan untuk mengestimasi parameter dinamik respons [27].

Pada penelitian ini, titik 63,2% digunakan untuk mengidentifikasi konstanta waktu plant sebagai informasi karakteristik dinamika sistem pemanasan. Nilai konstanta waktu tersebut tidak digunakan secara langsung dalam perhitungan K_p , T_i , dan T_d pada bentuk *Ziegler–Nichols unit reaction rate* yang digunakan. Penalaan PID tetap menggunakan slope R , perubahan masukan Δu , *dead time* L , dan unit reaction rate R_1 .

Temperatur pada titik 63,2% ditentukan dari temperatur awal T_0 dan temperatur *steady state* T_{ss} . Setelah waktu ketika temperatur mencapai titik tersebut diperoleh, konstanta waktu plant τ untuk model dengan *dead time* dihitung sebagai selisih antara waktu pencapaian 63,2% dan *dead time* L . Hubungan ini digunakan untuk menjelaskan seberapa lambat respons termal plant.

Kemiringan respons atau slope R dan *dead time* L pada penelitian ini diperoleh dari hasil pengolahan grafik respons *open loop*. Nilai R ditentukan dari kemiringan bagian kurva yang relatif linear, sedangkan L ditentukan dari titik perpotongan perpanjangan garis pendekatan terhadap kondisi awal sistem. Nilai R dan L tersebut selanjutnya digunakan sebagai parameter pada metode *Ziegler Nichols Process Reaction Curve*.

Dengan demikian, identifikasi karakteristik *Open Loop* pada penelitian ini dilakukan dalam dua bagian. Pertama, titik 63,2% dan konstanta waktu τ digunakan untuk mendeskripsikan dinamika plant. Kedua, slope R dan *dead time* L digunakan untuk membentuk unit reaction rate R_1 dan menghitung parameter awal PID menggunakan aturan *Ziegler Nichols Process Reaction Curve*.

Perubahan total temperatur dari kondisi awal T_0 menuju temperatur steady-state T_{ss} dihitung menggunakan Persamaan berikut:

$$\Delta T = T_{ss} - T_0 \quad (2.12)$$

Keterangan :

ΔT = perubahan total temperatur ($^{\circ}\text{C}$)

T_{ss} = temperatur *steady state* atau temperatur akhir respons *open loop* ($^{\circ}\text{C}$)

T_0 = temperatur awal pengujian ($^{\circ}\text{C}$)

Temperatur pada titik 10% dihitung dari 10% perubahan total respons menggunakan Persamaan berikut:

$$T_{10} = T_0 + 0,10\Delta T \quad (2.13)$$

Keterangan :

T_{10} = temperatur pada 10% perubahan total respons ($^{\circ}\text{C}$)

T_0 = temperatur awal pengujian ($^{\circ}\text{C}$)

ΔT = perubahan total temperatur ($^{\circ}\text{C}$)

Temperatur pada titik 63,2% dihitung menggunakan Persamaan berikut:

$$T_{63,2} = T_0 + 0,632\Delta T \quad (2.14)$$

Keterangan :

$T_{63,2}$ = temperatur pada 63,2% perubahan total respons ($^{\circ}\text{C}$)

T_0 = temperatur awal pengujian ($^{\circ}\text{C}$)

ΔT = perubahan total temperatur ($^{\circ}\text{C}$)

Temperatur pada titik 90% dihitung menggunakan Persamaan berikut:

$$T_{90} = T_0 + 0,90\Delta T \quad (2.15)$$

Keterangan :

T_{90} = temperatur pada 90% perubahan total respons ($^{\circ}\text{C}$)

T_0 = temperatur awal pengujian ($^{\circ}\text{C}$)

ΔT = perubahan total temperatur ($^{\circ}\text{C}$)

Karena data temperatur dicatat pada interval waktu diskrit, waktu ketika respons tepat mencapai T_{10} , $T_{63,2}$, atau T_{90} dapat berada di antara dua sampel. Waktu pencapaian titik persentase tersebut ditentukan dengan interpolasi linier antara dua data terdekat menggunakan Persamaan berikut:

$$t_p = t_1 + \left(\frac{T_p - T_1}{T_2 - T_1} \right) (t_2 - t_1) \quad (2.16)$$

Keterangan :

t_p = waktu saat temperatur mencapai target T_p (menit)

t_1 = waktu pada sampel pertama (menit)

t_2 = waktu pada sampel kedua (menit)

T_p = temperatur target ($^{\circ}\text{C}$)

T_1 = temperatur pada waktu t_1 ($^{\circ}\text{C}$)

T_2 = temperatur pada waktu t_2 ($^{\circ}\text{C}$)

Dengan T_p adalah temperatur target persentase, sedangkan (t_1, T_1) dan (t_2, T_2) merupakan dua sampel terdekat yang mengapit nilai T_p . Setelah waktu pencapaian 63,2% diketahui, konstanta waktu plant untuk model dengan *dead time* dihitung menggunakan Persamaan berikut:

$$\tau = t_{63,2} - L \quad (2.17)$$

Keterangan :

τ = konstanta waktu plant (menit)

$t_{63,2}$ = waktu saat respons mencapai 63,2% perubahan total (menit)

L = *dead time* atau waktu tunda plant (menit)

Nilai slope R yang diperoleh dari pengolahan grafik selanjutnya dinormalisasi terhadap besar perubahan masukan Δu untuk memperoleh unit reaction rate R_1 sesuai Persamaan (2.18). Nilai R_1 dan *dead time* L yang diperoleh dari grafik digunakan langsung pada aturan *Ziegler Nichols* untuk memperoleh K_p , T_i , dan T_d .

Pada penelitian ini, penalaan *Ziegler Nichols* dilakukan dari kurva reaksi proses (process-reaction curve) hasil pengujian *open loop*. Nilai slope R dan *dead time* L diperoleh dari hasil pengolahan grafik respons, sedangkan besar perubahan masukan Δu ditentukan dari masukan *heater* pada pengujian. Ketiga parameter tersebut digunakan untuk memperoleh unit reaction rate R_1 dan parameter awal PID

$$R_1 = \frac{R}{\Delta u} \quad (2.18)$$

Tabel 2.2 Penalaan Rumus

Tipe Kontrol	K_p	T_i	T_d
P	$\frac{T}{R_1 L}$	0	0
PI	$\frac{0,9T}{R_1 L}$	3,33L	0
PID	$\frac{1,2}{R_1 L}$	2L	0,5L

Berdasarkan karakteristik tersebut, aturan penalaan Ziegler–Nichols untuk pengendali PID dapat dinyatakan sebagai:

$$K_p = \frac{1,2}{R_1 L} \quad (2.19)$$

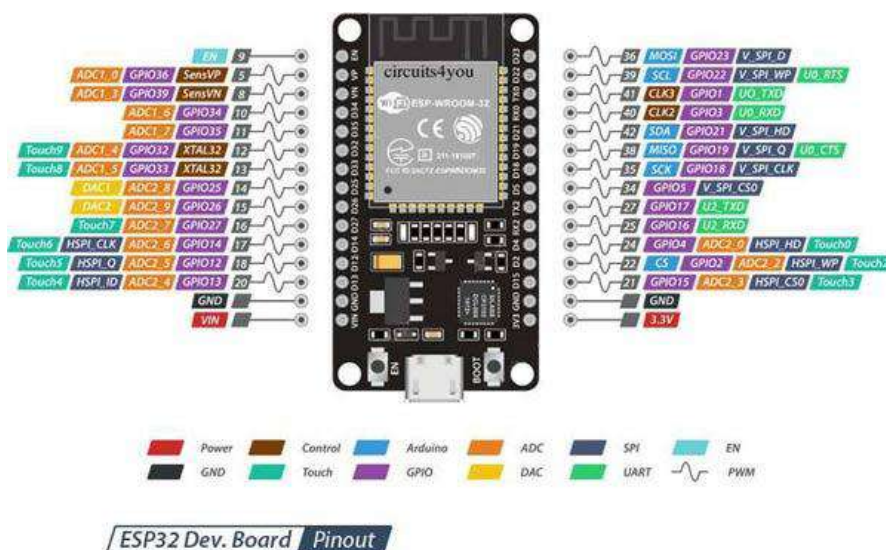
$$T_i = 2L \quad (2.20)$$

$$T_d = 0,5L \quad (2.21)$$

Parameter hasil tuning *Ziegler Nichols* digunakan sebagai parameter awal, sehingga kinerja pengendali masih perlu dievaluasi melalui pengujian *closed loop*. Hasil pengujian kemudian dapat dianalisis berdasarkan kemampuan sistem mencapai *setpoint*, besar *overshoot*, waktu penyelesaian, serta *error* pada kondisi stabil.

2.12 Mikrokontroler ESP32

ESP32 merupakan mikrokontroler yang banyak digunakan dalam sistem *embedded* dan *Internet of Things* (IoT) karena memiliki kemampuan pemrosesan yang tinggi serta dilengkapi dengan berbagai fitur komunikasi dan antarmuka perangkat keras. Selain modul *Wi-Fi*, mikrokontroler ini juga memiliki modul *Bluetooth* dan disertai dengan kapasitas RAM cukup besar untuk menyimpan data, hal tersebut membuat ESP32 menjadi salah satu mikrokontroler yang populer untuk membuat sebuah proyek *Internet of Things* (IoT). ESP32 ini memiliki 30 pin GPIO (15 pin channel ADC, 25 pin PWM, dan 2 pin channel DAC). Pada ESP32 ini memiliki tegangan operasi 3,3V dan 5V.



Gambar 2.8 Mikrokontroler ESP32 [28]

Pada Tabel 2.3 merupakan spesifikasi Mikrokontroller ESP32 yang akan dipakai pada penelitian ini.

Tabel 2.3 Parameter dan Spesifikasi

Parameter	Spesifikasi
GPIO	15 pin channel ADC, 25 pin PWM, dan 2 pin channel DAC
Tegangan	3,3V dan 5V
Output	3,3 v

2.13 Heater

Prinsip kerja *heater* air listrik adalah mengubah energi listrik menjadi energi panas melalui elemen pemanas yang memiliki hambatan listrik tinggi. Ketika arus listrik mengalir pada elemen tersebut, terjadi proses pemanasan akibat efek *Joule*, sehingga elemen menjadi panas. Panas yang dihasilkan kemudian ditransfer ke air melalui proses konduksi dari elemen ke dinding *Heater* dan dilanjutkan dengan konveksi di dalam air, sehingga temperatur air meningkat secara bertahap. Elemen Panas Listrik (*Electrical Heating Element*) pada *water heater* yaitu sebuah alat elektrik yang dapat memanaskan air dengan mudah dan dalam waktu yang singkat[29].



Gambar 2.9 Heater [30]

Pada Tabel 2.4 merupakan spesifikasi *heater* yang akan dipakai pada penelitian ini.

Tabel 2.4 Parameter dan Spesifikasi

Parameter	Spesifikasi
Panjang Elemen	195 mm
Tegangan	220 Volt
Daya	500 Watt
Diameter Pipa	8 mm
Ukuran Drat	1 Inch (ulir)

2.14 Persamaan Kebutuhan Daya Heater

Dalam sistem pemanas air, perhitungan kebutuhan daya *heater* sangat penting untuk menentukan kapasitas *heater* yang digunakan agar mampu menaikkan temperatur air sesuai dengan kebutuhan sistem. Energi panas yang diperlukan untuk menaikkan temperatur suatu zat dapat dihitung menggunakan persamaan kalor.

$$Q = mc\Delta T \quad (2.22)$$

Keterangan :

Q = energi panas (*Joule*)

m = massa air (kg)

c = kalor jenis air (4186 J/kg°C)

ΔT = perubahan temperatur (°C)

Jika energi panas tersebut diberikan oleh *heater* dengan daya tertentu, maka waktu pemanasan dapat dihitung menggunakan persamaan berikut.

$$P = \frac{Q}{t} \quad (2.23)$$

Keterangan :

P = daya *Heater* (*Watt*)

Q = energi panas (*Joule*)

t = waktu pemanasan (detik)

Perhitungan Arus Sisi AC untuk penggunaan MCB

$$I = \frac{P}{V} \quad (2.24)$$

Keterangan :

I = arus listrik (A)

P = daya listrik (*Watt*)

V = tegangan listrik (*Volt*)

Perhitungan Daya.

$$P = V \cdot I \quad (2.25)$$

Keterangan :

P= daya listrik (Watt)

V= tegangan listrik (Volt)

I= arus listrik (A)

2.15 Peltier TEC1-12706

Modul *Peltier* merupakan perangkat *termoelektrik* yang dapat menghasilkan efek pendinginan atau pemanasan ketika dialiri arus listrik. Modul ini banyak digunakan pada sistem pendingin skala kecil seperti pendingin elektronik, pendingin minuman, serta sistem kontrol temperatur.

Pada modul TEC1-12706, terdapat banyak pasangan elemen *semikonduktor* yang disusun seri secara listrik dan paralel secara termal untuk meningkatkan efek perpindahan panas. Ketika tegangan 12V diberikan, modul akan mulai memindahkan panas dari sisi dingin ke sisi panas, sehingga jika sisi panas tidak didinginkan menggunakan *heatsink* atau kipas, kinerja pendinginan akan menurun. Awal mula dari *Peltier* ialah kebalikan dari fenomena *Seebeck* yaitu ketika arus listrik mengalir pada suatu rangkaian dari material logam yang berbeda terjadi penyerapan panas pada sambungan kedua logam tersebut dan pelepasan panas pada sambungan yang lainnya. Pelepasan dan penyerapan panas bersesuaian dengan arah arus listrik pada logam, hal ini dikenal dengan efek *Peltier*[31].



Gambar 2.10 *Peltier TEC1-12706* [32]

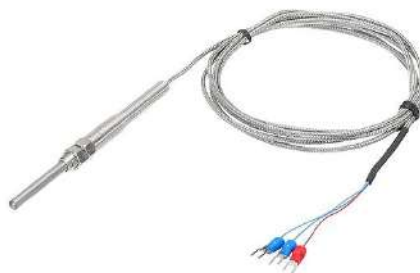
Pada Tabel 2.5 merupakan spesifikasi *Peltier* TEC1-12706 yang akan dipakai pada penelitian ini.

Tabel 2.5 Parameter dan Spesifikasi

Parameter	Spesifikasi
Tegangan	12 Volt
Electric current	5.8 A
Power	Qcmax 50-60W
Ukuran	40x40x2.6mm

2.16 PT100

PT100, yang merupakan perangkat penginderaan temperatur canggih, beroperasi dengan mendeteksi variasi temperatur dengan mengubah besarnya resistansinya sesuai dengan representasi matematis yang disediakan dalam Persamaan. Akibatnya, menjadi penting untuk membangun sirkuit listrik yang rumit yang mampu menerjemahkan *fluktuasi* nilai *resistansi* yang ditunjukkan oleh PT100 ke tingkat tegangan yang sesuai. Proses konversi ini sangat penting, karena memungkinkan pemantauan variasi temperatur yang tepat, yang dimanifestasikan sebagai pergeseran *output level* tegangan dari sensor. Jenis sirkuit spesifik yang diperlukan untuk berintegrasi secara efektif dengan sensor temperatur PT100 dicirikan sebagai konfigurasi jembatan *Wheatstone*, yang secara luas dikenal karena keefektifannya dalam mengukur perubahan resistansi[33].

**Gambar 2.11** Sensor PT100

Pada Tabel 2.6 merupakan spesifikasi Sensor PT100 yang akan dipakai pada penelitian ini.

Tabel 2.6 Parameter dan Spesifikasi

Parameter	Spesifikasi
Tegangan	220 Volt
Temperature Range	-50°C to 400°C
Akurasi	+/- 2.2°C
Panjang Kabel	2 Meter
Drat Baut	1.25mm (M8)
Ukuran Kepala	50mm, 100mm

2.17 *Transmitter* RTD PT100

Transmitter RTD (*Resistance Temperature Detector*) adalah perangkat konversi sinyal yang mengubah resistansi variabel dari sensor PT100 menjadi sinyal arus *analog*. Nilai resistansi tersebut perlu diolah agar dapat diterima oleh perangkat akuisisi atau pengendali. Salah satu cara yang digunakan pada sistem instrumentasi adalah menggunakan transmitter temperatur yang mengubah karakteristik resistansi PT100 menjadi sinyal standar, salah satunya 4–20 mA.

Penggunaan transmitter memudahkan sinyal temperatur diteruskan menuju perangkat pengendali, terutama ketika posisi sensor dan sistem akuisisi berada pada jarak yang berbeda. Pada penelitian tersebut juga digunakan konfigurasi PT100 tiga kawat untuk mengurangi pengaruh resistansi kabel dan gangguan pada proses pengukuran.

Sinyal 4–20 mA memberikan rentang keluaran yang mewakili batas bawah hingga batas atas pengukuran temperatur yang telah ditentukan pada *transmitter*. Dengan demikian, perubahan temperatur yang dibaca PT100 akan menghasilkan perubahan arus pada keluaran transmitter. Nilai tersebut kemudian dapat dikonversi menjadi tegangan atau dibaca oleh perangkat akuisisi yang sesuai sebelum diproses menjadi nilai temperatur[34].



Gambar 2.12 Transmitter RTD

Pada Tabel 2.7 merupakan spesifikasi *Transmitter* RTD yang akan dipakai pada penelitian ini.

Tabel 2.7 Parameter dan Spesifikasi

Parameter	Spesifikasi
Tipe <i>Input</i>	RTD PT100 (3 wire)
<i>Output</i>	4–20 mA
Tegangan Operasi	24 VDC
Rentang Temperature	0°C ~100°C (Tergantung model)
Akurasi Konversi	0.2% FS
Ukuran	44(D) x 18(H) mm

2.18 *Power Supply*

Power Supply adalah perangkat yang menyediakan energi listrik untuk satu atau beberapa beban listrik. *Power Supply* memiliki peran penting dalam dunia elektronika sebagai sumber tenaga, contohnya seperti baterai atau aki. Secara umum, konstruksi *Power Supply* memiliki struktur yang mirip, yang biasanya terdiri dari transformator, penyearah, dan perata tegangan. Istilah ini biasanya digunakan untuk perangkat yang mengubah satu jenis energi listrik menjadi jenis energi listrik lain, namun juga dapat merujuk pada perangkat yang mengubah energi dari bentuk lain-misalnya mekanik, kimia, atau energi surya menjadi listrik[35].



Gambar 2.13 *Power Supply*

Pada Tabel 2.8 merupakan spesifikasi *Power Supply* yang akan dipakai pada penelitian ini.

Tabel 2.8 Parameter dan Spesifikasi

Parameter	Spesifikasi
Tegangan	220 Volt
<i>Output</i> Tegangan	24 Volt
<i>Output</i> Arus	4,5 Ampere
Dimensi	21,5 x 11,5 x 5,5 mm

2.19 *Buck Converter*

Buck Converter adalah salah satu topologi dasar dari *DC to DC converter* yang berfungsi untuk menurunkan tegangan *input* (V_i) dari sumber DC yang lebih tinggi. *Buck Converter* dikenal memiliki efisiensi tinggi serta desain yang relatif sederhana. Rancangannya terdiri dari komponen *switching* semikonduktor, seperti MOSFET dan dioda, serta komponen pasif, yaitu induktor dan kapasitor, ditambah resistor sebagai beban. Kehadiran komponen *switching* membuat *Buck Converter* menjadi sistem orde dua, sehingga keluaran yang dihasilkan bersifat nonlinier dalam domain waktu[36].



Gambar 2.14 *Buck Converter 5V*

(Dokumentasi Pribadi)

Pada Tabel 2.9 merupakan spesifikasi *Buck Converter* yang akan dipakai pada penelitian ini.

Tabel 2.9 Parameter dan Spesifikasi

Parameter	Spesifikasi
Tegangan	<i>Input</i> 12/24V
<i>Output</i> Tegangan	5V
<i>Output</i> Arus	30A
Keunggulan	Efisiensi tinggi (85-95%)

Dibawah ini adalah gambar *Buck Converter 12V* yang akan dipakai pada penelitian ini.



Gambar 2.15 *Buck Converter 12V*

Pada Tabel 2.10 merupakan spesifikasi *Buck Converter* yang akan dipakai pada penelitian ini.

Tabel 2.10 Parameter dan Spesifikasi

Parameter	Spesifikasi
Model Number	DY-20A
Tegangan	<i>Input</i> 15-30V
<i>Output</i> Tegangan	12V,
<i>Output</i> Arus	20 Ampere
Dimensions	117.5x83x40mm

2.20 AC Light Dimmer Module

AC Light Dimmer merupakan modul yang digunakan untuk mengatur besar daya AC yang diberikan ke beban seperti lampu maupun *heater* dengan menggunakan metode *phase angle control*. Modul ini bekerja menggunakan komponen TRIAC dan *zero crossing detector* untuk mengatur sudut penyalan gelombang AC sehingga besar tegangan dan arus yang diterima beban dapat dikendalikan, sistem *dimmer* AC memanfaatkan *zero crossing* untuk sinkronisasi pemotongan gelombang AC agar proses pengaturan daya menjadi lebih stabil dan mengurangi noise pada sistem[37].

Prinsip kerja modul *dimmer* AC menggunakan komponen utama berupa TRIAC dan rangkaian *zero crossing detector*. *Zero crossing detector* berfungsi mendeteksi titik nol pada gelombang AC sinusoidal. Setelah titik nol terdeteksi, mikrokontroler memberikan *delay* tertentu sebelum memicu TRIAC untuk menghantarkan arus. Metode ini disebut *phase angle control*, yaitu pengaturan sudut fasa penyalan TRIAC pada setiap setengah gelombang AC. Besarnya *delay* penyalan menentukan besar tegangan efektif (RMS) yang diterima oleh beban. Jika TRIAC dipicu lebih cepat, maka daya yang diterima *heater* semakin besar. Jika dipicu lebih lambat, maka daya yang diterima *heater* menjadi kecil.

Pada sistem kontrol temperatur, modul *dimmer* digunakan untuk mengatur daya *heater* secara bertahap berdasarkan sinyal kontrol dari sistem PID. Semakin besar sudut penyalan TRIAC, maka semakin besar daya yang diberikan ke *heater* sehingga temperatur meningkat. Sebaliknya, jika sudut penyalan diperkecil maka daya *Heater* menurun dan temperatur menjadi lebih stabil.



Gambar 2.16 *AC Light Dimmer Module* [38]

Pada gambar 2.16 merupakan bentuk dari *AC Light Dimmer Module* dan untuk Tabel 2.11 dibawah merupakan parameter dan spesifikasi dari *AC Light Dimmer Module* yang akan digunakan.

Tabel 2.11 Parameter dan Spesifikasi

Parameter	Spesifikasi
Nama Komponen	<i>AC Light Dimmer 2 channel</i>
<i>Power</i>	<i>Power Up to 400V AC</i>
Arus	8A (<i>normal</i>) - 16A (<i>peak</i>)
Tegangan Mikrokontroler	3.3V/5V DC