

**RANCANG BANGUN SISTEM PENGENDALIAN SUHU DAN  
LEVEL AIR MENGGUNAKAN *RATIO CONTROLLER* DAN  
*FUZZY LOGIC***

**TUGAS AKHIR**

Laporan ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan  
Program Studi Diploma Empat Teknik Otomasi Industri  
di Jurusan Teknik Elektro

**Oleh:**

**MUAMMAR FATHURRAHMAN SAHIB**

**NIM : 211364015**



**POLBAN**

POLBAN

**POLITEKNIK NEGERI BANDUNG**

**2025**

## TUGAS AKHIR

# RANCANG BANGUN SISTEM PENGENDALIAN SUHU DAN LEVEL AIR MENGGUNAKAN *RATIO CONTROLLER* DAN *FUZZY LOGIC*

Disusun oleh:

**Muammar Fathurrahman Sahib**

**NIM: 211364015**

Tugas Akhir ini telah disidangkan pada tanggal 20 Juni 2025 dan disahkan sesuai dengan ketentuan.

Tim Penguji:

Ketua : Sofyan Muhammad Ilman S.ST., MT  
NIP. 199403302022041001


Anggota 1 : Giga Verian Pratama, S.Si., M.T.  
NIP. 199508212024061001

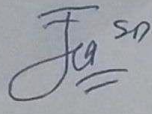
Menyetujui,

Bandung, Juli 2025

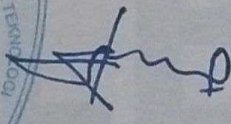
Pembimbing Utama

Pembimbing Pendamping

  
Nanang Mulyono, ST., MT.  
NIP. 197707182008121001

  
Febi Ariefka Septian Putra, S.ST., M.T.  
NIP. 199302102022031006

Ketua Jurusan Teknik Elektro

  
Dr. Hesti Ludyati, S.T., M.T.  
NIP. 197204262001122001



## PERNYATAAN PENULIS

Dengan ini menyatakan bahwa laporan Tugas Akhir dengan judul Rancang Bangun Sistem Pengendalian Suhu dan Level Air Menggunakan *Ratio Controller* dan *Fuzzy Logic* adalah karya ilmiah yang bebas dari unsur tindakan plagiarisme, dan sesuai dengan ketentuan tata tulis yang berlaku.

Apabila dikemudian hari ditemukan adanya unsur plagiarisme, maka hasil penilaian dari Tugas Akhir ini dicabut dan bersedia menerima sanksi sesuai dengan ketentuan yang berlaku.

Demikian pernyataan ini dibuat dengan sesungguhnya dalam keadaan sadar sepenuhnya.

Bandung, 14 Juli 2025



Muammar Fathurrahman Sahib

NIM: 211364015

POLBAN

## ABSTRAK

Pengendalian suhu dan level air sangat penting dalam menjaga kualitas produk pada proses industri. Untuk menjawab kebutuhan tersebut, perancangan dan implementasi sistem pengendalian suhu dan level air otomatis berbasis Raspberry Pi Pico dengan pendekatan *fuzzy logic* dan *ratio controller* pada *mini plant* berskala laboratorium dengan konfigurasi *Multi-Input Multi-Output* (MIMO) dilakukan. Pengujian sistem terhadap berbagai setpoint suhu menunjukkan bahwa sistem mampu memenuhi spesifikasi performa, yakni *overshoot* <10%, *rise time* <2 menit, *settling time* <5 menit, dan *steady-state error* <5%. Hasil ini menunjukkan bahwa kombinasi metode *fuzzy* dan *ratio controller* berhasil memberikan hasil respon yang stabil terhadap setpoint dalam pengendalian suhu dan level air secara simultan pada sistem prototipe berbasis mikrokontroler.

**Kata Kunci:** *Fuzzy Logic*, *Ratio Controller*, MIMO, Pengendalian Suhu, Level Air

POLBAN

## ***ABSTRACT***

*Temperature and water level control are very important in maintaining product quality in industrial processes. To answer these needs, the design and implementation of an automatic temperature and water level control system based on Raspberry Pi Pico with a fuzzy logic and ratio controller approach on a laboratory-scale mini plant with a Multi-Input Multi-Output (MIMO) configuration were carried out. System testing of various temperature setpoints showed that the system was able to meet performance specifications, namely overshoot <10%, rise time <2 minutes, settling time <5 minutes, and steady-state error <5%. These results indicate that the combination of fuzzy and ratio controller methods successfully provided stable response results to setpoints in controlling temperature and water levels simultaneously on a microcontroller-based prototype system.*

**Keywords:** *Fuzzy Logic, Ratio Controller, MIMO, Temperature Control, Water Level*

POLBAN

## KATA PENGANTAR

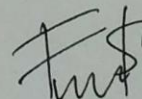
Puji syukur penulis panjatkan kepada Allah Subhanahu wa Ta'ala atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir berjudul **“Rancang Bangun Sistem Pengendalian Suhu dan Level Air Menggunakan *Ratio Controller* dan *Fuzzy Logic*”** tepat waktu. Laporan ini disusun sebagai salah satu syarat untuk menyelesaikan pendidikan pada Program Diploma Empat Teknik Otomasi Industri di Jurusan Teknik Elektro.

Penulis mengucapkan terima kasih sebesar-besarnya kepada:

1. Mamah dan Abah yang selalu mendoakan, memberikan dukungan, serta menjadi sumber kekuatan dalam setiap langkah penulis.
2. Ibu Dr. Hapi Ludyati, S.T., M.T. selaku ketua Jurusan Teknik Elektro Politeknik Negeri Bandung.
3. Bapak Nanang Mulyono, S.T., M.T. selaku pembimbing utama tugas akhir yang telah meluangkan waktu, tenaga dan pikiran untuk selalu memberikan pengarahan, petunjuk dan saran dalam pengerjaan tugas akhir ini.
4. Bapak Febi Ariefka Septian Putra, S.ST., MT. sebagai pembimbing pendamping tugas akhir yang selalu memberikan pengarahan dan bantuan untuk menyelesaikan setiap kendala dan permasalahan.
5. Pemilik NIM 211364007, yang telah menjadi rekan kerja sama yang penuh dedikasi, memberikan kontribusi, semangat, dan komitmen dalam setiap tahap penyusunan tugas akhir ini.

Penulis juga menyadari bahwa masih terdapat kekurangan dalam laporan ini, sehingga saran dan kritik yang membangun sangat diharapkan untuk perbaikan di masa mendatang. Semoga tugas akhir ini dapat memberikan manfaat, dan penulis menghargai setiap dukungan serta perhatian yang telah diberikan.

Bandung, 14 Juli ..... 2025



Muammar Fathurrahman Sahib

NIM. 211364015



## DAFTAR ISI

|   |      |
|---|------|
| PERNYATAAN PENULIS .....                      | ii   |
| ABSTRAK .....                                 | iv   |
| <i>ABSTRACT</i> .....                         | v    |
| KATA PENGANTAR.....                           | vi   |
| DAFTAR ISI .....                              | vii  |
| DAFTAR LAMPIRAN .....                         | x    |
| DAFTAR GAMBAR .....                           | xi   |
| DAFTAR TABEL.....                             | xii  |
| DAFTAR ISTILAH .....                          | xiii |
| DAFTAR SINGKATAN DAN LAMBANG.....             | xv   |
| DAFTAR RUMUS.....                             | xvii |
| BAB I PENDAHULUAN .....                       | 1    |
| I.1 Latar Belakang .....                      | 1    |
| I.2 Rumusan Masalah .....                     | 2    |
| I.3 Tujuan.....                               | 2    |
| I.4 Ruang Lingkup.....                        | 2    |
| I.5 Sistematika Penulisan.....                | 3    |
| BAB II TINJAUAN PUSTAKA DAN DASAR TEORI ..... | 4    |
| II.1 Tinjauan Pustaka .....                   | 4    |
| II.2 Dasar Teori .....                        | 5    |
| II.2.1 <i>Mini Plant</i> Pengkondisi Air..... | 5    |
| II.2.1.1 Modul Pemanas .....                  | 6    |
| II.2.1.2 Modul Pendingin .....                | 7    |
| II.2.1.3 Sensor dan Aktuator .....            | 8    |
| II.2.1.3.1 Modul Sensor Suhu dan Level .....  | 8    |
| II.2.1.3.2 Aktuator dan Penunjang .....       | 10   |
| II.2.1.4 Modul Antarmuka .....                | 12   |
| II.2.2 Raspberry Pi Pico .....                | 13   |

|                                     |   |    |
|-------------------------------------|---|----|
| II.2.3                              | <i>Fuzzy Logic</i> .....                                    | 14 |
| II.2.3.1                            | Inferensi <i>Fuzzy</i> .....                                | 14 |
| II.2.3.2                            | Fungsi Keanggotaan .....                                    | 16 |
| II.2.4                              | <i>Ratio Controller</i> .....                               | 19 |
| BAB III METODE RANCANG BANGUN ..... |   | 20 |
| III.1                               | Prosedur Pelaksanaan .....                                  | 20 |
| III.2                               | Perancangan <i>Mini Plant</i> .....                         | 21 |
| III.3                               | Diagram Alir Kerja Sistem .....                             | 22 |
| III.3.1                             | Diagram Blok <i>Mini Plant</i> .....                        | 24 |
| III.3.2                             | Diagram Blok Sistem Kendali .....                           | 25 |
| III.4                               | Perancangan Sistem Kendali .....                            | 27 |
| III.4.1                             | Fungsi Keanggotaan dan Basis Aturan <i>Fuzzy</i> .....      | 27 |
| III.4.2                             | Perancangan Kontrol .....                                   | 30 |
| III.4.2.1                           | Pengendalian <i>Fuzzy</i> dan <i>Ratio Controller</i> ..... | 31 |
| III.4.2.1.1                         | Prinsip Kerja Kontrol <i>Fuzzy</i> .....                    | 31 |
| III.4.2.1.2                         | Prinsip Kerja <i>Ratio Controller</i> .....                 | 35 |
| III.4.2.2                           | Pengendalian Level .....                                    | 35 |
| BAB IV HASIL DAN PEMBAHASAN .....   |   | 37 |
| IV.1                                | Hasil Pengujian Sistem .....                                | 37 |
| IV.1.1                              | Respon <i>Open Loop</i> .....                               | 37 |
| IV.1.2                              | Respon <i>Close Loop</i> .....                              | 38 |
| IV.1.3                              | Validasi Sensor Suhu DS18B20 .....                          | 41 |
| IV.2                                | Evaluasi Kinerja Sistem Kendali Suhu .....                  | 41 |
| BAB V KESIMPULAN DAN SARAN .....    |   | 43 |
| V.1                                 | Kesimpulan .....  | 43 |
| V.2                                 | Saran .....   | 43 |
| DAFTAR PUSTAKA .....                |   | 44 |



|               |    |
|---------------|----|
| LAMPIRAN..... | 46 |
|---------------|----|



**POLBAN**

## DAFTAR LAMPIRAN

|   |    |
|---|----|
| Lampiran 1. Panduan Penggunaan Alat .....                     | 46 |
| Lampiran 2. Skema Rangkaian Kendali.....                      | 52 |
| Lampiran 3. Skema Rangkaian <i>Mini Plant</i> .....           | 53 |
| Lampiran 4. Desain 3D <i>Mini Plant</i> Pengkondisi Air ..... | 54 |
| Lampiran 5. Realisasi <i>Mini Plant</i> Pengkondisi Air.....  | 55 |
| Lampiran 6. Program Raspberry Pi Pico.....                    | 56 |

POLBAN

## DAFTAR GAMBAR

|              |   |    |
|--------------|---|----|
| Gambar II.1  | Diagram Blok Sistem MIMO .....  | 6  |
| Gambar II.2  | Mug Listrik .....   | 6  |
| Gambar II.3  | Thermostat Digital HX W3001.....  | 7  |
| Gambar II.4  | Peltier TEC-12706 .....   | 7  |
| Gambar II.5  | Sensor Suhu DS18B20 .....   | 9  |
| Gambar II.6  | Modul ADC ADS1115 .....   | 9  |
| Gambar II.7  | <i>Base Plate</i> Sensor .....  | 10 |
| Gambar II.8  | Katup Servo .....   | 11 |
| Gambar II.9  | Pompa Air .....   | 11 |
| Gambar II.10 | Katup Solenoid.....   | 12 |
| Gambar II.11 | Relay 2 <i>Channel</i> .....  | 12 |
| Gambar II.12 | LCD 20x4.....   | 13 |
| Gambar II.13 | Keypad Matriks 4x4 .....  | 13 |
| Gambar II.14 | Raspberry Pi Pico .....   | 14 |
| Gambar II.15 | Grafik Fungsi Keanggotaan Segitiga .....                                | 17 |
| Gambar II.16 | Grafik Fungsi Keanggotaan Trapesium .....                               | 17 |
| Gambar III.1 | Diagram Alir Tahapan Pelaksanaan .....                                  | 20 |
| Gambar III.2 | Rancangan <i>Mini Plant</i> .....                                       | 22 |
| Gambar III.3 | Diagram Sistem <i>Mini Plant</i> MIMO .....                             | 23 |
| Gambar III.4 | Diagram Blok <i>Mini Plant</i> .....                                    | 24 |
| Gambar III.5 | Diagram Blok Sistem Kendali .....                                       | 25 |
| Gambar III.6 | Fungsi Keanggotaan <i>Error</i> .....                                   | 28 |
| Gambar III.7 | Fungsi Keanggotaan $\Delta Error$ .....                                 | 29 |
| Gambar III.8 | <i>Flowchart</i> Kendali <i>Fuzzy</i> dan <i>Ratio Controller</i> ..... | 31 |
| Gambar IV.1  | Respon Suhu terhadap Bukaannya Katup Pemanas Tetap.....                 | 37 |
| Gambar IV.2  | Respon Suhu terhadap Bukaannya Katup Pendingin Tetap.....               | 38 |
| Gambar IV.3  | Grafik Respon Suhu terhadap Setpoint 20°C.....                          | 39 |
| Gambar IV.4  | Grafik Respon Suhu terhadap Setpoint 40°C.....                          | 39 |
| Gambar IV.5  | Grafik Respon Suhu terhadap Setpoint Dinamis .....                      | 40 |
| Gambar IV.6  | Perbandingan Suhu DS18B20 terhadap Suhu Termometer .....                | 41 |

## DAFTAR TABEL

|             |   |    |
|-------------|---|----|
| Tabel II.1  | Perbandingan Metode <i>Fuzzy</i> Mamdani dan Sugeno .....           | 15 |
| Tabel III.1 | <i>Input</i> Sistem MIMO .....                                      | 23 |
| Tabel III.2 | <i>Output</i> Sistem MIMO.....                                      | 23 |
| Tabel III.3 | Batas Fungsi Keanggotaan <i>Fuzzy</i> Variabel <i>Error</i> .....   | 28 |
| Tabel III.4 | Batas Fungsi Keanggotaan <i>Fuzzy</i> Variabel $\Delta Error$ ..... | 29 |
| Tabel III.5 | Nilai <i>Crisp Output</i> Berdasarkan Kategori <i>Fuzzy</i> .....   | 29 |
| Tabel III.6 | <i>Fuzzy Rule</i> .....   | 30 |
| Tabel IV.1  | Evaluasi Kinerja Sistem Kendali Suhu .....                          | 41 |

POLBAN



## DAFTAR ISTILAH

|                             |  |
|-----------------------------|--|
| <i>Overshoot</i>            | : Perubahan <i>output</i> sistem yang melebihi nilai setpoint sebelum akhirnya kembali menuju kondisi <i>steady-state</i> .  |
| Osilasi                     | : Perubahan <i>output</i> sistem yang berulang dengan respon naik-turun terhadap nilai setpoint.   |
| Histerisis                  | : Perbedaan respon sistem terhadap <i>input</i> yang meningkat dan menurun, sehingga nilai <i>output</i> tidak langsung mengikuti perubahan <i>input</i> .   |
| <i>Mini plant</i>           | : Replika berskala kecil dari sistem industri yang sebenarnya.   |
| <i>Hardware</i>             | : Perangkat keras.   |
| <i>Software</i>             | : Perangkat lunak.   |
| <i>Rise time</i>            | : Waktu yang dibutuhkan <i>output</i> sistem untuk naik menuju setpoint.   |
| <i>Settling time</i>        | : Waktu yang dibutuhkan oleh <i>output</i> sistem untuk tetap berada batas toleransi tertentu, setelah mengalami <i>overshoot</i> dan osilasi.   |
| <i>Steady-state error</i>   | : Selisih antara setpoint dan <i>output</i> sistem setelah mencapai kondisi stabil.  |
| <i>Matrix</i>               | : Bahasa pemrograman tingkat tinggi dan lingkungan komputasi numerik yang digunakan untuk analisis data, pemodelan system, simulasi, perhitungan matematis, visualisasi, dan <i>control system</i> . |
| <i>Laboratory</i>           |  |
| <i>Wireless</i>             | : Teknologi komunikasi tanpa kabel.  |
| <i>Multiplexing spasial</i> | : Teknik transmisi data dalam MIMO dengan pemisahan spasial.   |
| <i>Nichrome</i>             | : Paduan logam nikel-kromium untuk elemen pemanas.   |
| Peltier                     | : Perangkat efek termoelektrik untuk pendinginan.  |
| <i>solid-state</i>          | : Teknologi atau perangkat berbasis material padat tanpa bagian bergerak.  |
| <i>Waterblock</i>           | : Komponen pendingin berbasis air untuk perangkat elektronik.  |

|                   |  |
|-------------------|--|
| <i>Heatsink</i>   | : Komponen pendingin yang menyerap dan melepas panas.                                |
| Semikonduktor     | : Material dengan konduktivitas listrik yang dapat dikontrol.                        |
| <i>Plunger</i>    | : Inti logam.  |
| <i>Interface</i>  | : Titik penghubung antara dua sistem atau perangkat untuk komunikasi atau interaksi. |
| Matriks           | : Susunan angka atau simbol dalam baris dan kolom untuk operasi matematika.          |
| Inferensi         | : Proses menarik kesimpulan dari data atau informasi.                                |
| Fuzzifikasi       | : Proses mengubah nilai tegas menjadi nilai kabur dalam <i>fuzzy logic</i> .         |
| Implikasi         | : Hubungan sebab-akibat antara dua pernyataan dalam logika.                          |
| Agregasi          | : Penggabungan beberapa elemen menjadi satu kesatuan.                                |
| Defuzzifikasi     | : Proses mengubah nilai kabur menjadi nilai tegas dalam <i>fuzzy logic</i> .         |
| <i>Antecedent</i> | : Bagian kondisi dalam aturan “IF...THEN...”.  |
| <i>Consequent</i> | : Bagian hasil dalam aturan “IF...THEN...”.  |
| <i>Close Loop</i> | : Sistem kendali dengan umpan balik.   |
| <i>Open Loop</i>  | : Sistem kendali tanpa umpan balik.  |

POLBAN

## DAFTAR SINGKATAN DAN LAMBANG

| SINGKATAN | NAMA   | PEMAKAIAN PERTAMA<br>KALI PADA HALAMAN |
|-----------|--|--|
| MIMO      | <i>Multi-Input Multi-Output</i>                        | 1                                      |
| SISO      | <i>Single-Input Single-Output</i>                      | 4                                      |
| DCS       | <i>Distributed Control System</i>                      | 4                                      |
| MATLAB    | <i>Matrix Laboratory</i>                               | 4                                      |
| RTD       | <i>Resistance Temperature<br/>Detector</i>             | 9                                      |
| NTC       | <i>Negative Temperature<br/>Coefficient</i>            | 9                                      |
| PLA       | <i>Polylactic Acid</i>                                 | 11                                     |
| I2C       | <i>Inter-Integrated Circuit</i>                        | 12                                     |
| SDA       | <i>Serial Data</i>                                     | 13                                     |
| SCL       | <i>Serial Clock</i>                                    | 13                                     |
| RAM       | <i>Random Access Memory</i>                            | 14                                     |
| GPIO      | <i>General Purpose<br/>Input/Output</i>                | 14                                     |
| UART      | <i>Universal Asynchronous<br/>Receiver Transmitter</i> | 14                                     |
| PWM       | <i>Pulse Width Modulation</i>                          | 14                                     |

POLBAN

| LAMBANG  | NAMA                   | PEMAKAIAN PERTAMA<br>KALI PADA HALAMAN |
|----------|------------------------|--|
| $\mu$    | <i>Mu</i>              | 16                                     |
| $\alpha$ | <i>Alpha</i>           | 16                                     |
| $\int$   | <i>Integral</i>        | 16                                     |
| $\Sigma$ | <i>Sigma</i>           | 16                                     |
| $\sigma$ | <i>Standar Deviasi</i> | 18                                     |
| $e$      | Bilangan Euler         | 18                                     |
| $\Delta$ | <i>Delta</i>           | 29                                     |

POLBAN



## DAFTAR RUMUS

|   |    |
|---|----|
| Persamaan Implikasi Fuzzy Mamdani .....                 | 16 |
| Persamaan Implikasi Fuzzy Sugeno .....                  | 16 |
| Persamaan Agregasi Fuzzy Mamdani .....                  | 16 |
| Persamaan Defuzzifikasi Fuzzy Mamdani .....             | 16 |
| Persamaan Defuzzifikasi Fuzzy Sugeno .....              | 16 |
| Persamaan Rules IF THEN .....                           | 16 |
| Persamaan Matematis Fungsi Keanggotaan Segitiga .....   | 18 |
| Persamaan Matematis Fungsi Keanggotaan Trapesium .....  | 18 |
| Persamaan Matematis Fungsi Keanggotaan Gassium .....    | 18 |
| Persamaan <i>Ratio Controller</i> .....                 | 19 |
| Rumus <i>Error</i> .....                                | 32 |
| Rumus $\Delta Error$ .....                              | 32 |
| Rumus Derajat Keanggotaan Variabel <i>Error</i> .....   | 32 |
| Rumus Derajat Keanggotaan Variabel $\Delta Error$ ..... | 32 |
| Rumus Aturan <i>Fuzzy</i> .....                         | 32 |
| Rumus Derajat Aktivasi .....                            | 33 |
| Rumus <i>Output Fuzzy</i> .....                         | 33 |
| Rumus Bukan Katup Air Panas .....                       | 35 |
| Rumus Bukan Katup Air Dingin .....                      | 35 |

POLBAN

# BAB I

## PENDAHULUAN

### I.1 Latar Belakang

Berbagai sektor industri seperti makanan dan kimia sangat bergantung pada akurasi pengendalian suhu dan level air dalam tangki, karena parameter ini berpengaruh langsung terhadap kualitas produk dan efisiensi proses produksi. Ketidaktepatan dalam pengendalian dapat menyebabkan fluktuasi kualitas, pemborosan energi, hingga ketidaksesuaian spesifikasi produk. Oleh karena itu, dibutuhkan sistem pengendalian yang tidak hanya akurat, tetapi juga otomatis dan responsif, guna memastikan parameter dalam tangki tetap berada pada nilai yang diinginkan secara konsisten [1].

Pada penelitian sebelumnya, metode kontrol *On-Off* banyak digunakan karena implementasinya yang sederhana. Namun, metode ini memiliki sejumlah keterbatasan, seperti kecenderungan munculnya *overshoot* dan histeresis, yang merupakan karakteristik khas dari jenis kontrol ini [2]. Kondisi ini menjadi hambatan dalam menciptakan sistem pengendalian yang efisien, terutama pada proses yang memerlukan respon cepat dan akurat.

Untuk menjawab permasalahan tersebut, digunakan pendekatan pengendalian dengan mengombinasikan *fuzzy logic* dan *ratio controller*. *Ratio controller* berfungsi untuk mengatur perbandingan air panas dan air dingin, sedangkan *fuzzy logic* memberikan fleksibilitas dalam menghadapi ketidakpastian dan dinamika sistem pada parameter *ratio controller*. Implementasi metode ini diharapkan mampu meningkatkan akurasi dan stabilitas dalam pengendalian suhu dan level air secara otomatis.

Penelitian ini akan merancang dan membangun sistem pengendalian berbasis mikrokontroler pada *mini plant* pengkondisian air yang memiliki konfigurasi MIMO. Sistem terdiri dari tiga tangki pendukung dan satu tangki utama, di mana variabel yang dimanipulasi adalah katup pemanas dan katup pendingin, sedangkan variabel yang dikontrol adalah suhu dan level air pada tangki utama. Fokus penelitian ini mencakup perancangan *hardware*, implementasi *software*, serta

pengujian dan analisis performa sistem untuk memastikan bahwa metode yang diterapkan dapat meningkatkan stabilitas dan akurasi pengendalian suhu dan level air secara otomatis.

## **I.2 Rumusan Masalah**

1. Bagaimana merancang dan membangun *mini plant* pengkondisian air yang dapat mendukung sistem pengendalian suhu dan level air secara otomatis dalam konfigurasi MIMO?
2. Bagaimana merancang dan merealisasikan sistem pengendalian suhu dan level air otomatis berbasis Raspberry Pi Pico yang sesuai untuk konfigurasi MIMO pada *mini plant*?
3. Bagaimana mengimplementasikan metode *fuzzy logic* dengan *ratio controller* untuk menghasilkan pencampuran air panas dan air dingin yang mampu menjaga suhu sesuai setpoint?

## **I.3 Tujuan**

Penelitian ini bertujuan untuk merancang dan membangun *mini plant* berskala laboratorium sebagai media pengujian sistem kendali suhu dan level air otomatis. Sistem dikendalikan menggunakan mikrokontroler Raspberry Pi Pico dengan konfigurasi MIMO, yang mengintegrasikan metode *fuzzy logic* tipe Sugeno dan *ratio controller* untuk mengatur rasio pencampuran air panas dan dingin, serta logika digital untuk kontrol level air. Tujuan akhir dari penelitian ini adalah mengevaluasi kinerja sistem terhadap berbagai setpoint suhu untuk memastikan tercapainya kriteria teknis, yaitu *overshoot* <10%, *rise time* <2 menit, *settling time* <5 menit, dan *steady-state error* <5%.

## **I.4 Ruang Lingkup**

Adapun ruang lingkup pada tugas akhir ini yaitu:

1. Fokus pada perancangan dan implementasi sistem kendali suhu dan level air menggunakan *ratio controller* dan *fuzzy logic*.
2. Sistem dikendalikan menggunakan mikrokontroler Raspberry Pi Pico.
3. *Mini plant* dibangun menggunakan pendekatan eksperimen prototipe, tanpa perhitungan teknis seperti debit, tekanan, dan dimensi pipa.

4. Lingkup terbatas pada *hardware mini plant*, pengembangan *software* kendali, dan pengujian performa berdasarkan parameter yang telah ditentukan.
5. Sistem dikembangkan dengan kisaran suhu air dingin sekitar 15°C–20°C dan air panas antara 40°C–50°C, serta setting target suhu *output* pada tangki utama antara 20°C–40°C.

### **I.5 Sistematika Penulisan**

Sistematika penulisan laporan Tugas Akhir ini disusun berdasarkan pedoman yang berlaku, dan secara garis besar terdiri dari beberapa bab utama sebagai berikut:

1. Bab I Pendahuluan, bab ini menjelaskan latar belakang permasalahan, rumusan masalah, tujuan, ruang lingkup, serta sistematika penulisan laporan tugas akhir secara keseluruhan.
2. Bab II Tinjauan Pustaka, berisi kajian teori dan literatur yang menjadi landasan dalam pelaksanaan penelitian. Bab ini mencakup konsep-konsep dasar dan referensi ilmiah yang relevan untuk mendukung perancangan dan implementasi sistem.
3. Bab III Metode Rancang Bangun, berisikan penguraian metode yang digunakan dalam penelitian, mulai dari tahapan perancangan sistem, alat dan bahan yang digunakan, proses realisasi sistem, hingga metode pengujian dan analisis data.
4. Bab IV Hasil dan Pembahasan, menyajikan hasil dari setiap tahapan yang telah dilakukan, termasuk pengujian sistem dan analisis terhadap performa yang dihasilkan berdasarkan parameter yang telah ditentukan.
5. Bab V Kesimpulan dan Saran, menyimpulkan temuan-temuan utama dari penelitian serta memberikan saran yang dapat dijadikan pertimbangan untuk pengembangan atau penelitian lanjutan di masa mendatang.
6. Daftar Pustaka, berisi daftar referensi yang digunakan selama penyusunan laporan tugas akhir, baik dari jurnal, buku, maupun sumber ilmiah lainnya yang relevan.
7. Lampiran, memuat dokumen pendukung, data tambahan, serta informasi lain yang berkaitan dengan pelaksanaan dan hasil tugas akhir.



## BAB II

### TINJAUAN PUSTAKA DAN DASAR TEORI

#### II.1 Tinjauan Pustaka

Sistem pengkondisi air pada proses produksi makanan, minuman, kimia, farmasi dan proses produksi lainnya, masih sering kali menghadapi tantangan, seperti fluktuasi atau perubahan suhu dan level air yang dapat menyebabkan ketidaksesuaian terhadap nilai setpoint, sehingga berdampak pada kualitas produk akhir dan efisiensi proses. Sehingga, berbagai penelitian terkait dengan sistem pengkondisi air telah dilakukan dengan tujuan untuk menghasilkan sistem yang cepat dan stabil dalam menghadapi perubahan kondisi operasional. Penelitian mengenai sistem pengkondisian air dengan kendali MIMO, yang mengatur suhu dan level air secara terintegrasi masih terbatas. Sebagian besar studi sebelumnya lebih berfokus pada pengendalian *Single-Input Single-Output* (SISO).

Penelitian oleh Satrio Sarwo Mumpuni dan Rini Puji Astutik [3] menunjukkan bahwa mikrokontroler dapat digunakan untuk memproses sinyal *input* dari berbagai sensor dan menghasilkan sinyal kontrol yang sesuai untuk mengatur aktuator dimana pada penelitian tersebut jika suhu di atas 28°C akan mengaktifkan pompa air dan akan mati jika suhu turun di bawah 28°C, demikian juga dengan kelembapan pasir. Jika kelembapan kurang dari 80%, pompa air yang mengairi pasir akan diaktifkan.

Merujuk pada hasil penelitian I Putu Ari Kristiawan, Muhamad Rifa, dan Denda Dewatama [4] serta hasil penelitian Yogi Sulistyopambudi, Muhamad Rifa'i, dan Ratna Ika Putri [5] metode *ratio control* dapat diimplementasikan pada sistem *Distributed Control System* (DCS) dengan cara merasiokan komposisi 2 komponen atau lebih dengan suatu perbandingan tertentu yang terdapat di prosesnya. Sedangkan pada penelitian Nadya Sari Nastiti [6] berhasil mensimulasikan rasio kontrol di MATLAB dengan variabel yang dirasiokan adalah laju aliran biogas dan laju aliran yang dimanipulasi.

Merujuk pada hasil penelitian Puput Wanarti R, Miftahur Rohman, dan Lilik Anifah Afiliati [7] dan penelitian Fani Kurniawan, Yulian Zetta Maulana, dan Risa

Farrid Christianti [8], kedua penelitian tersebut berhasil mensimulasikan kendali suhu dan level menggunakan metode *fuzzy* pada *software* MATLAB.

Pengendalian suhu dan level pada *mini plant* ini merupakan inovasi terkini yang belum pernah dilakukan dalam penelitian sebelumnya, mengingat *mini plant* ini dirancang dan dibangun oleh penulis sendiri. Oleh karena itu, kriteria pengendalian yang ingin dicapai pun berbeda dibandingkan dengan penelitian-penelitian sebelumnya. Sebagian besar penelitian terdahulu fokus pada *plant* dengan sistem SISO, sementara sistem yang diterapkan di *plant* ini lebih kompleks.

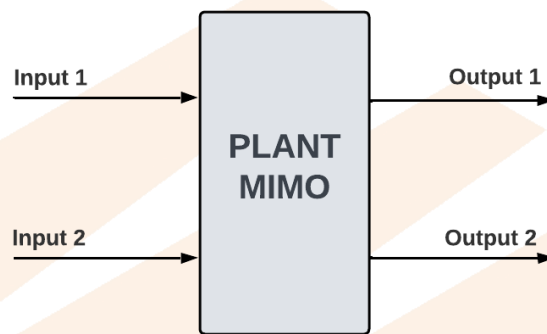
## II.2 Dasar Teori

Dasar teori merupakan landasan ilmiah yang digunakan dalam proses perancangan dan pembangunan sistem pengkondisian air otomatis berbasis Raspberry Pi Pico dengan metode pengendalian *fuzzy logic* dan *ratio controller*. Pemahaman terhadap konsep dan prinsip kerja dari masing-masing komponen dan metode pengendalian sangat diperlukan agar sistem dapat dibangun dengan tepat dan sesuai fungsi yang diharapkan.

Dalam bab ini dibahas berbagai teori yang menjadi dasar dari penelitian, antara lain teori mengenai sistem pengkondisian air, komponen-komponen seperti pemanas, pendingin, sensor suhu dan level, aktuator berupa pompa dan katup, serta pengendali *fuzzy logic* dan *ratio controller* yang digunakan untuk mengatur pencampuran air panas dan air dingin. Selain itu, dibahas pula mengenai platform mikrokontroler Raspberry Pi Pico yang menjadi otak dari sistem.

### II.2.1 Mini Plant Pengkondisi Air

*Mini plant* pengkondisi air ini menggunakan pendekatan kendali MIMO untuk mengatur suhu dan level air secara simultan, meningkatkan kinerja dan stabilitas sistem dibandingkan kendali SISO. Konsep ini mengadopsi prinsip sistem *wireless* MIMO yang memanfaatkan banyak *input* dan *output* untuk meningkatkan kapasitas dan keandalan sistem melalui keragaman dan *multiplexing* spasial. Pendekatan ini memungkinkan pengendalian interaksi antar variabel secara lebih terkoordinasi dan selaras dengan karakteristik sistem industri yang kompleks, sebagaimana dijelaskan oleh Roza dan Mujirudin [9].



Gambar II.1 Diagram Blok Sistem MIMO

*Mini plant* ini merupakan suatu sistem berskala kecil yang dirancang untuk mengatur dan menjaga parameter penting dari air, seperti suhu dan level air, agar tetap stabil dan sesuai dengan nilai setpoint yang telah ditetapkan. Sistem ini dirancang menyerupai proses yang terjadi pada sistem nyata di industri, namun dalam skala laboratorium atau prototipe. Fungsinya tidak hanya sebagai representasi sistem kendali di dunia nyata, tetapi juga sebagai media pembelajaran dan pengujian konsep kendali secara langsung.

#### II.2.1.1 Modul Pemanas



Gambar II.2 Mug Listrik

Mug listrik merupakan perangkat listrik yang dirancang untuk meningkatkan suhu air dengan memanfaatkan elemen pemanas, seperti kawat *nichrome* atau pelat berbahan logam resistif, yang terpasang pada bagian dasar mug. Prinsip kerja dari modul ini adalah mengubah energi listrik menjadi energi panas melalui elemen pemanas tersebut.

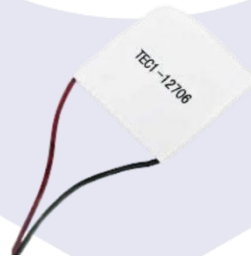


Gambar II.3 Thermostat Digital HX W3001

Pada sistem *mini plant* yang dirancang dalam penelitian ini, penggabungan mug listrik dengan thermostat digital digunakan sebagai tangki sumber air panas yang memiliki tinggi 10 cm dan diameter 12 cm dengan kapasitas 900ml. Fungsinya adalah untuk menaikkan suhu air hingga mencapai suhu 40-50°C

#### II.2.1.2 Modul Pendingin

Modul pendingin ini merupakan gabungan beberapa komponen yang digunakan untuk menurunkan suhu air dengan memanfaatkan prinsip perpindahan panas. Modul ini bekerja dengan cara menyerap panas dari air di satu sisi dan melepaskannya ke lingkungan di sisi lainnya. Salah satu komponen utama yang digunakan dalam sistem ini adalah modul termoelektrik Peltier TEC-12706.



Gambar II.4 Peltier TEC-12706

Peltier TEC1-12706 adalah modul pendingin *solid-state* yang bekerja berdasarkan efek Peltier, di mana aliran arus listrik menyebabkan perbedaan suhu antara dua permukaan keramikanya. Saat tegangan diberikan, satu sisi modul menjadi dingin dan menyerap panas dari *waterblock* yang ditempelkan, sementara sisi lainnya menjadi panas dan harus didinginkan menggunakan sistem pembuangan panas seperti *heatsink* aluminium besar dan kipas DC.

Pada sistem *mini plant* dalam penelitian ini, modul pendingin dihubungkan dengan tangki sumber air dingin yang berdiameter 11 cm serta tinggi 10 cm dan



memiliki volume  $\pm 700$  ml. Target suhu air dingin yang dihasilkan berkisar pada 15-20°C.

### II.2.1.3 Sensor dan Aktuator

Sensor dan aktuator merupakan komponen penting dalam sistem otomatisasi, termasuk pada *mini plant* pengkondisian air. Sensor berfungsi sebagai perangkat *input* yang mendeteksi perubahan fisik pada lingkungan, seperti suhu dan ketinggian air, lalu mengubahnya menjadi sinyal listrik yang dapat diproses oleh mikrokontroler. Sementara itu, aktuator bertugas sebagai perangkat *output* yang menerima sinyal kontrol untuk menggerakkan atau mengatur perangkat mekanis, seperti katup dan pompa, dalam rangka merealisasikan aksi pengendalian. Kedua komponen ini menjadi elemen kunci dalam menciptakan sistem kendali *close loop* yang mampu merespon perubahan kondisi secara cepat dan sesuai dengan kebutuhan sistem.

#### II.2.1.3.1 Modul Sensor Suhu dan Level

Sensor suhu dan sensor level merupakan komponen penting dalam sistem *mini plant* pengkondisian air, karena keduanya memberikan umpan balik utama untuk proses pengendalian suhu dan level air dalam tangki. Modul ini bertugas menyediakan informasi *real-time* yang digunakan dalam logika kontrol untuk memastikan suhu dan tinggi air berada dalam batas yang diinginkan.

- Sensor DS18B20

Sensor ini digunakan untuk memantau suhu air pada tangki pemanas, tangki pendingin, maupun tangki utama. Sensor ini menggunakan protokol komunikasi *1-Wire* dan tersedia dalam versi tahan air, menjadikannya ideal untuk aplikasi pengukuran suhu langsung dalam cairan. DS18B20 memiliki resolusi hingga 12-bit dan akurasi  $\pm 0,5^{\circ}\text{C}$ , serta mampu bekerja dalam rentang suhu  $-55^{\circ}\text{C}$  hingga  $+125^{\circ}\text{C}$  [10], menjadikannya cocok untuk aplikasi pemantauan suhu air dalam sistem ini. Kelebihan lainnya adalah sifatnya yang tahan terhadap kelembapan dan dapat direndam langsung ke dalam air menggunakan pembungkus tahan air.



Gambar II.5 Sensor Suhu DS18B20

Sensor ini termasuk kategori termometer digital berbasis semikonduktor, dan bukan termasuk tipe RTD (*Resistance Temperature Detector*) ataupun NTC (*Negative Temperature Coefficient*). Artinya, sensor ini tidak mengandalkan perubahan resistansi seperti RTD atau NTC, melainkan langsung mengeluarkan data digital hasil konversi suhu dari sensor internalnya [10].

- Sensor Level

Sensor Level digunakan untuk mendeteksi level air dalam tangki, baik tangki sumber maupun tangki utama. Sensor ini bekerja berdasarkan prinsip konduktivitas air, dengan tiga kabel utama: kabel catu daya 5V, sinyal level tinggi (*high level*), dan sinyal level rendah (*low level*). Saat permukaan air menyentuh titik sensor, konduktivitas antara kabel sinyal dan kabel 5V akan terbentuk, sehingga level air dapat dikenali secara digital sebagai kondisi "*HIGH*" atau "*LOW*". Dengan desain yang sederhana, sensor ini cukup andal untuk digunakan dalam aplikasi monitoring level air pada sistem skala kecil hingga menengah.

- Modul ADC Eksternal



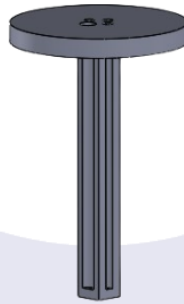
Gambar II.6 Modul ADC ADS1115

Untuk meningkatkan akurasi pembacaan dan memudahkan integrasi pembacaan level dengan mikrokontroler, pada sistem ini juga digunakan modul

ADC eksternal ADS1115. ADS1115 berfungsi mengkonversi sinyal analog dari sensor level menjadi sinyal digital dengan resolusi tinggi, sehingga mampu memberikan data level air yang lebih presisi dan stabil. Penambahan ADS1115 membuat pemantauan level air menjadi lebih andal, terutama dalam kondisi sinyal yang lemah atau bising pada sistem.

- *Base Plate Sensor*

Untuk mendukung efisiensi integrasi dan pemasangan, kedua jenis sensor ini disusun dalam satu modul berbasis *base plate* yang didesain secara modular agar mudah dipasang pada tutup tangki. Dengan desain ini, sensor suhu dan level dapat diposisikan secara strategis dan stabil pada ketinggian tertentu sesuai kebutuhan. Integrasi dalam satu modul memungkinkan penghematan ruang dan kabel, serta mempermudah proses kalibrasi dan perawatan.



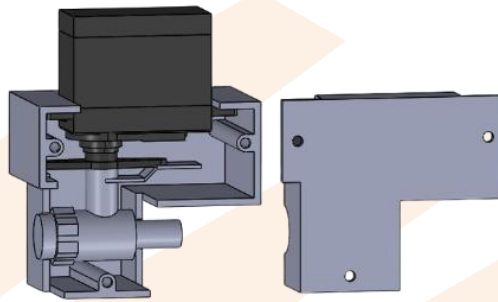
Gambar II.7 *Base Plate Sensor*

#### II.2.1.3.2 Aktuator dan Penunjang

- Katup Servo

Katup servo merupakan salah satu aktuator penting dalam sistem mini plant pengkondisian air yang berfungsi untuk mengatur bukaan air panas dan air dingin. Berbeda dengan katup konvensional yang hanya memiliki dua kondisi (buka dan tutup), katup servo mampu melakukan pengaturan secara proporsional terhadap tingkat bukaan katup, sehingga aliran fluida dapat disesuaikan secara lebih presisi sesuai kebutuhan sistem.

Servo motor yang digunakan dalam sistem ini adalah MG995, yaitu servo tipe standar dengan gear berbahan logam yang tangguh dan tahan lama. MG995 memiliki desain *double ball bearing* yang memberikan kestabilan, ketahanan terhadap getaran, serta presisi putaran yang baik untuk aplikasi kontrol fluida.



Gambar II.8 Katup Servo

Untuk mengintegrasikan motor servo dengan valve secara mekanis, digunakan braket atauudukan khusus yang dirancang untuk menyatukan posisi servo dan valve agar sumbu putar sejajar seperti pada gambar II.8. Braket ini dibuat menggunakan 3D printing berbahan *Polylactic Acid* (PLA). Desain braket disesuaikan agar servo terpasang kokoh dan tidak goyah saat beroperasi.

- Pompa Air



Gambar II.9 Pompa Air

Pompa air adalah perangkat elektromekanik yang berfungsi mengubah energi listrik menjadi energi mekanik berupa gaya hisap dan dorong, yang digunakan untuk mengalirkan air dari satu tempat ke tempat lainnya. Dalam sistem *mini plant* pengkondisi air, pompa yang digunakan bekerja dengan tegangan DC 12V, yang umum digunakan karena efisien dan aman.

- Modul Katup Solenoid

Katup kendali selenoid merupakan jenis aktuator yang berfungsi untuk mengatur bukaan katup secara otomatis berdasarkan prinsip kerja elektromagnetik. Katup ini memiliki mekanisme berupa kumparan elektromagnetik yang, saat diberi tegangan, menghasilkan medan magnet untuk menarik *plunger* ke dalam, sehingga

membuka jalur aliran air. Ketika tegangan dilepaskan, pegas akan mengembalikan *plunger* ke posisi semula dan menutup jalur tersebut.



Gambar II.10 Katup Solenoid

Karena katup solenoid umumnya membutuhkan tegangan dan arus yang relatif besar yaitu 12V DC dengan arus sekitar 1A, pengendaliannya tidak dapat dilakukan langsung oleh mikrokontroler seperti Raspberry Pi Pico, yang hanya menyediakan arus kecil pada pin digital (sekitar 3.3V dengan arus <50 mA). Untuk mengatasi kendala ini, modul relay digunakan sebagai penghubung antara sinyal mikrokontroler dan katup solenoid.

- Relay 2 Channel



Gambar II.11 Relay 2 Channel

Relay adalah saklar elektromagnetik yang dikendalikan oleh sinyal mikrokontroler untuk menghubungkan atau memutuskan sirkuit bertegangan lebih tinggi. Dalam sistem ini, Raspberry Pi Pico akan mengirimkan sinyal digital ke modul relay 2 channel 5V, yang kemudian akan mengaktifkan atau mematikan arus utama menuju katup solenoid dan pompa.

#### II.2.1.4 Modul Antarmuka

- Display

Pada sistem mini plant ini digunakan layar LCD karakter 20x4 dengan *interface* I2C (*Inter-Integrated Circuit*). LCD I2C 20x4 ini mampu menampilkan 20 karakter dalam 4 baris dan menggunakan protokol komunikasi I2C yang hanya

membutuhkan dua kabel data (SDA dan SCL) untuk menghubungkan dengan mikrokontroler. Penggunaan *interface* I2C pada LCD ini mempermudah proses wiring dan mengurangi jumlah pin yang digunakan pada mikrokontroler, sehingga lebih efisien. LCD I2C menampilkan informasi penting seperti status sistem, nilai suhu, level air, dan parameter lain secara *real-time* untuk memudahkan monitoring dan kontrol sistem *mini plant*.



Gambar II.12 LCD 20x4

- Keypad

Sebagai media input, digunakan keypad matriks 4x4 yang terdiri dari 16 tombol yang tersusun dalam 4 baris dan 4 kolom. Keypad ini memungkinkan penulis untuk memasukkan perintah atau pengaturan seperti nilai setpoint suhu, pembacaan sensor, atau kontrol manual sistem.



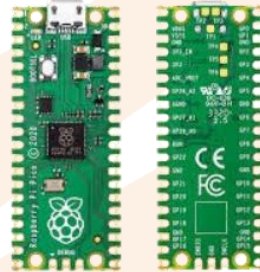
Gambar II.13 Keypad Matriks 4x4

## II.2.2 Raspberry Pi Pico

Raspberry Pi Pico adalah papan mikrokontroler berbiaya rendah dan berperforma tinggi yang dikembangkan oleh Raspberry Pi Foundation. Papan ini menggunakan chip RP2040, yaitu mikrokontroler berbasis dual-core Arm Cortex-



M0+ yang berjalan hingga 133 MHz, dengan RAM sebesar 264 KB dan memori flash internal sebesar 2 MB. Mikrokontroler ini dirancang untuk mendukung berbagai aplikasi sistem tertanam dengan efisiensi tinggi dan konsumsi daya yang rendah.



Gambar II.14 Raspberry Pi Pico

Papan ini dipilih karena memiliki jumlah GPIO yang cukup untuk menghubungkan seluruh sensor dan aktuator dalam sistem, serta mendukung komunikasi digital seperti I2C, UART, dan 1-Wire. Kemampuan PWM (*Pulse Width Modulation*) pada Raspberry Pi Pico juga memungkinkan pengaturan sudut bukaan katup servo secara presisi. Selain itu, ukuran Pico yang ringkas dan konsumsi daya yang efisien menjadikannya ideal untuk sistem *mini plant* berskala laboratorium seperti yang dirancang dalam penelitian ini.

### II.2.3 Fuzzy Logic

*Fuzzy logic* merupakan salah satu metode pengendalian cerdas yang dirancang untuk menangani ketidakpastian dan ketidaktepatan data pada sistem kendali. Berbeda dengan sistem logika biner konvensional yang hanya mengenal nilai 0 dan 1, *fuzzy logic* memungkinkan nilai keanggotaan di antara 0 dan 1 [11].

#### II.2.3.1 Inferensi Fuzzy

Metode inferensi yang umum digunakan dalam sistem *fuzzy logic* terbagi menjadi dua metode utama, yaitu metode Mamdani dan Sugeno. Berdasarkan penelitian [11], diperoleh kesimpulan mengenai persamaan dan perbedaan antara kedua metode tersebut:

Tabel II.1 Perbandingan Metode *Fuzzy* Mamdani dan Sugeno

| Aspek                   | Persamaan  | Perbedaan   |
|-------------------------|--|---|
| <b>Fuzzifikasi</b>      | Keduanya menerapkan proses fuzzifikasi untuk mengubah input numerik menjadi representasi <i>fuzzy</i> dengan menggunakan fungsi keanggotaan. | -   |
| <b>Basis Aturan</b>     | Menggunakan basis aturan berbentuk IF-THEN untuk menggambarkan hubungan antara <i>input fuzzy</i> dan <i>output fuzzy</i> .                  | -   |
| <b>Proses Inferensi</b> | Menggunakan inferensi logika berdasarkan aturan IF-THEN.   | <p>Mamdani: Setiap aturan menghasilkan himpunan <i>fuzzy</i>, menggunakan fungsi implikasi MIN dan agregasi MAX.</p> <p>Sugeno: Setiap aturan menghasilkan nilai numerik langsung berdasarkan fungsi matematis.</p>           |
| <b>Output</b>           | -  | <p>Mamdani: <i>Output</i> akhir diperoleh melalui proses defuzzifikasi (biasanya metode <i>centroid</i>).</p> <p>Sugeno: <i>Output</i> akhir berupa nilai numerik tanpa perlu proses defuzzifikasi himpunan <i>fuzzy</i>.</p> |

Tabel II.2 Rumus Perhitungan Inferensi Fuzzy

| Aspek                | Mamdani   | Sugeno  |
|----------------------|---|---|
| <b>Implikasi</b>     | $\mu_{output} = \min (\alpha - x_i)$ (II.1)                                     | $w_i$ (II.2)<br>$= \min (\alpha - x_i)$                           |
| <b>Agregasi</b>      | $\mu_{agregat}(x)$<br>$= \max (\mu_{rule_1}(x), \mu_{rule_2}(x), \dots)$ (II.3) | Tidak diperlukan<br>karena <i>output</i> langsung<br>berupa angka |
| <b>Defuzzifikasi</b> | $z = \frac{\int x * \mu_{agregat}(x) dx}{\int \mu_{agregat}(x) dx}$ (II.4)      | $z = \frac{\sum w_i * z_i}{\sum w_i}$ (II.5)                      |

Dalam sistem inferensi *fuzzy*, aturan dengan bentuk IF-THEN memiliki *output* berupa konstanta atau himpunan *fuzzy*. Secara umum, model *fuzzy* dapat ditulis sebagai:

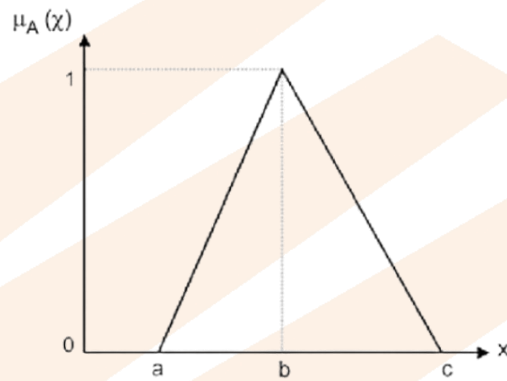
$$IF (x_1 \text{ is } A_1). (x_2 \text{ is } A_2) \dots (x_n \text{ is } A_n) THEN z = k \quad (II.6)$$

Dengan  $A_i$  adalah himpunan *fuzzy* sebagai *antecedent* dan  $k$  adalah konstanta sebagai *consequent* [12]. Dalam penelitian ini, metode Sugeno dipilih karena lebih efisien dan mudah diterapkan pada mikrokontroler dengan keterbatasan daya komputasi.

### II.2.3.2 Fungsi Keanggotaan

Dalam sistem *fuzzy logic*, fungsi keanggotaan berperan penting untuk memetakan *input* numerik ke dalam derajat keanggotaan dalam himpunan *fuzzy*. Menurut Arpit Jain dan Abhinav Sharma [13] beberapa bentuk fungsi keanggotaan yang umum digunakan adalah sebagai berikut:

## 1. Fungsi Keanggotaan Segitiga



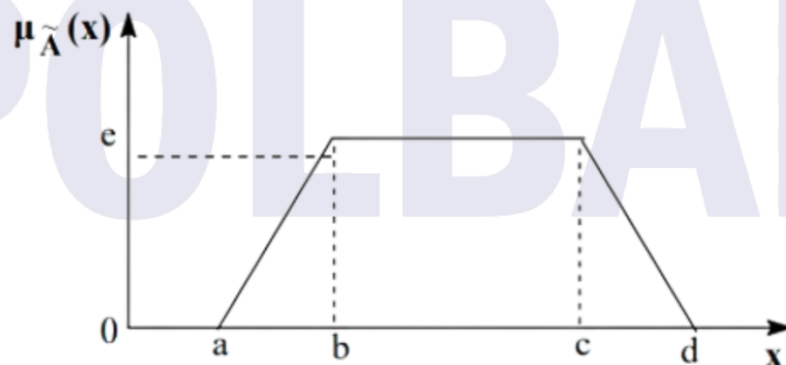
Gambar II.15 Grafik Fungsi Keanggotaan Segitiga

Merupakan bentuk paling sederhana yang terdiri dari tiga parameter utama: titik awal (a), puncak (b), dan titik akhir (c). Fungsi ini membentuk segitiga, di mana keanggotaan meningkat linier dari a ke b, dan menurun linier dari b ke c. Dengan persamaan matematis sebagai berikut.

$$\mu_A(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & x \geq c \end{cases} \quad (\text{II.7})$$

Fungsi ini sering digunakan karena komputasinya ringan dan cocok untuk sistem dengan keterbatasan daya pemrosesan seperti mikrokontroler.

## 2. Fungsi Keanggotaan Trapesium

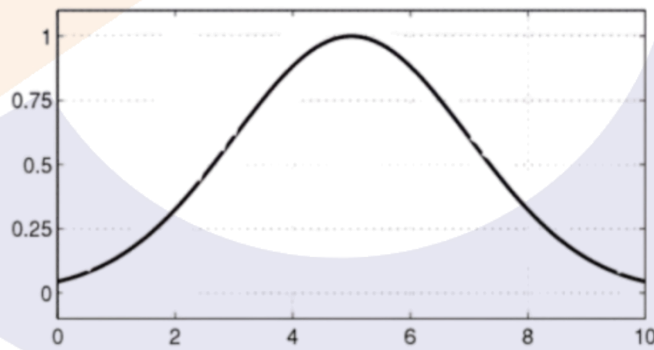


Gambar II.16 Grafik Fungsi Keanggotaan Trapesium

Memiliki empat parameter: a, b, c, dan d, yang membentuk kurva trapesium. Fungsi ini memiliki bagian puncak datar yang menunjukkan rentang nilai dengan keanggotaan penuh (nilai 1). Bentuk ini sangat berguna ketika nilai input dianggap sepenuhnya termasuk dalam satu kategori dalam suatu rentang tertentu. Berikut adalah persamaan matematisnya:

$$\mu_A(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c \leq x < d \\ 0, & x \geq d \end{cases} \quad (\text{II.8})$$

### 3. Fungsi Keanggotaan Gaussian



Gambar II.17 Grafik Fungsi Keanggotaan Gaussian

Berbentuk kurva lonceng yang halus, ditentukan oleh dua parameter: pusat (c) dan deviasi standar ( $\sigma$ ). Dengan persamaan eksponensial:

$$\mu_A(x) = e^{\frac{-(x-c)^2}{2\sigma^2}} \quad (\text{II.9})$$

Sehingga fungsi ini tidak memiliki batas tegas, karena nilainya menurun secara eksponensial namun tidak pernah benar-benar nol. Cocok untuk data alami dan aplikasi yang membutuhkan transisi halus.

Dalam sistem *fuzzy logic* yang diimplementasikan pada *mini plant* pengkondisian air, digunakan fungsi keanggotaan jenis segitiga. Fungsi ini dipilih karena memiliki struktur yang sederhana dan ringan secara komputasi. Selain itu,

bentuk segitiga memberikan cukup fleksibilitas untuk membagi rentang nilai suhu menjadi beberapa kategori linguistik.

#### II.2.4 *Ratio Controller*

*Ratio controller* adalah metode pengendalian dalam sistem industri yang digunakan untuk mempertahankan perbandingan tetap antara dua variabel atau parameter proses, meskipun nilai absolut dari salah satu atau kedua variabel tersebut mengalami perubahan, seperti yang diimplementasikan pada eksperimen menggunakan *Process Control Trainer* dari Apex Innovations [14]. Tujuan utamanya adalah memastikan bahwa rasio antara dua aliran atau parameter selalu konsisten demi menjaga kualitas hasil akhir [15].

Dari deskripsi diatas maka didapatkan persamaan matematis sebagai berikut:

$$R = \frac{K_2}{K_1} \quad (\text{II.10})$$

Dimana,

R = Rasio yang ditentukan,

K<sub>1</sub> = Variabel utama, dan

K<sub>2</sub> = Variabel yang dikontrol agar sesuai dengan rasio terhadap variabel utama.

Penggunaan *ratio controller* dalam tugas akhir ini sangat relevan karena sistem mini plant yang dirancang berfokus pada pencampuran air panas dan air dingin untuk mencapai suhu target secara otomatis. Dengan mengatur rasio bukaan air panas dan dingin, suhu hasil campuran di tangki utama dapat dikendalikan secara presisi.

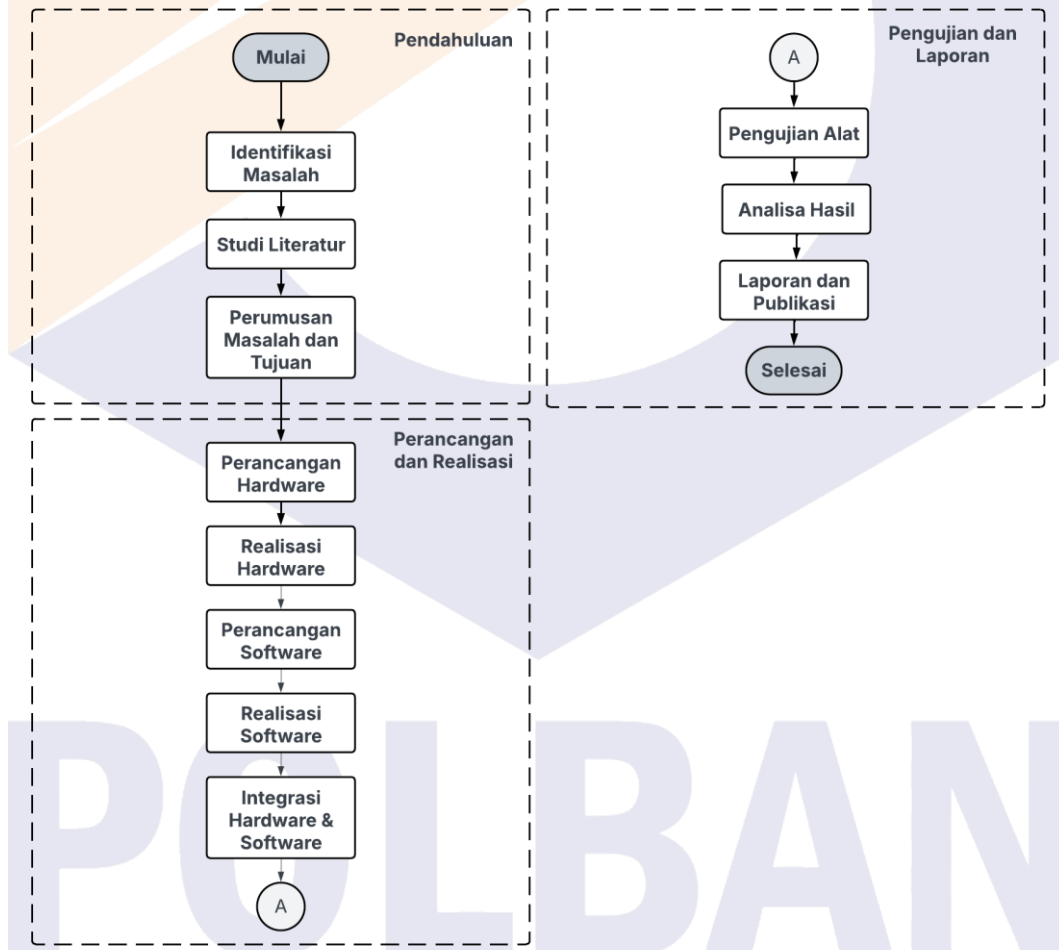


## BAB III

### METODE RANCANG BANGUN

#### III.1 Prosedur Pelaksanaan

Prosedur pelaksanaan dalam tugas akhir ini dibagi ke dalam tiga tahap utama yang saling berkesinambungan, yaitu tahap pendahuluan, perancangan dan realisasi sistem, serta pengujian dan pelaporan. Setiap tahapan disusun secara sistematis agar proyek rancang bangun dapat dilaksanakan secara terstruktur dan efisien. Diagram alir pada Gambar III.1 menggambarkan urutan logis kegiatan yang dilakukan selama pengerjaan tugas akhir ini.



Gambar III.1 Diagram Alir Tahapan Pelaksanaan

Tahap pendahuluan dimulai dengan identifikasi permasalahan di industri terkait pengendalian suhu dan level air. Setelah itu dilakukan studi literatur untuk memahami berbagai metode pengendalian yang telah ada, seperti kontrol *on-off*,

*ratio controller*, dan *fuzzy logic*. Berdasarkan hasil kajian tersebut, dirumuskan masalah dan tujuan dari tugas akhir ini.

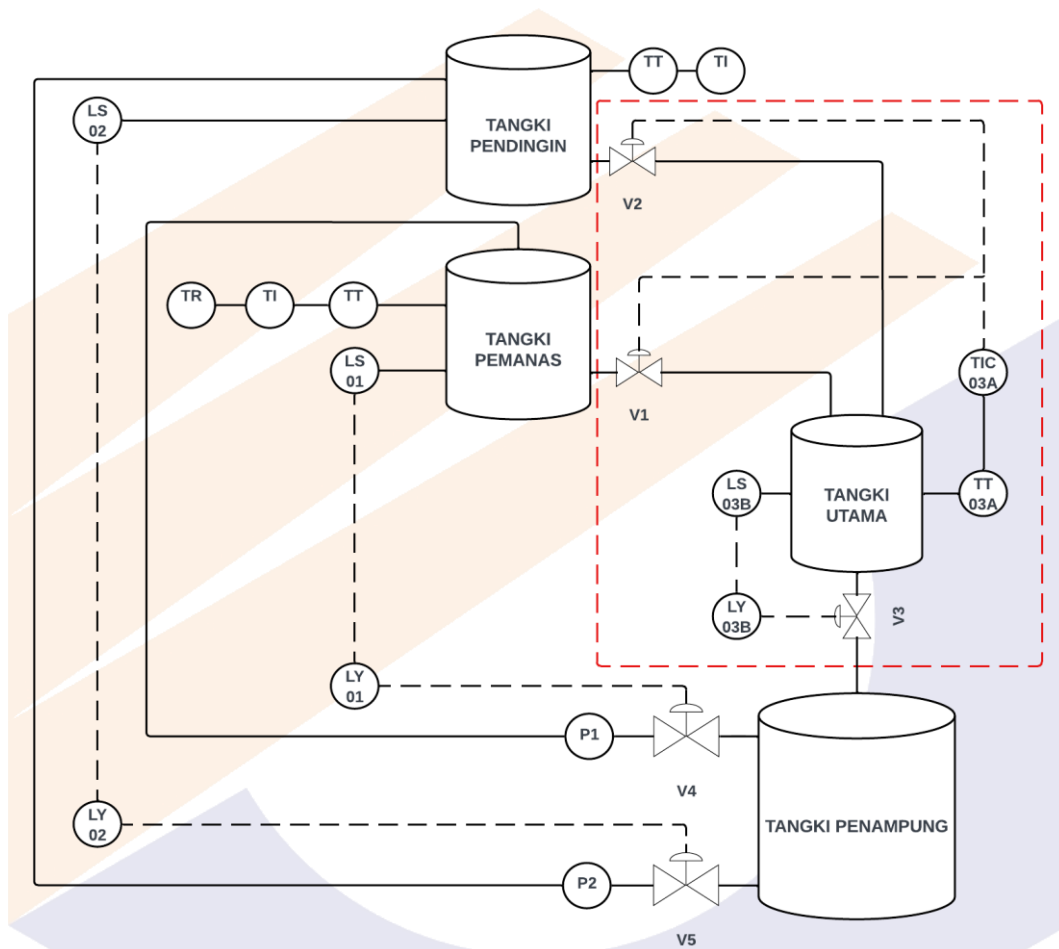
Selanjutnya, pada tahap perancangan dan realisasi, dilakukan perancangan *mini plant* sebagai media simulasi pengendalian suhu dan level air. Perancangan meliputi desain mekanis, seperti penempatan tangki dan komponen fisik, serta desain elektrik seperti sistem wiring dan kendali aktuator. Setelah perancangan selesai, dilakukan realisasi atau perakitan *mini plant* sesuai desain. *Software* sistem kendali juga dirancang menggunakan *fuzzy logic* dan *ratio controller*, lalu diimplementasikan ke dalam mikrokontroler Raspberry Pi Pico. Tahap ini diakhiri dengan proses integrasi antara *hardware* dan *software* untuk membentuk satu sistem kendali yang utuh dan siap diuji.

Tahap terakhir adalah pengujian dan pelaporan. Sistem yang telah selesai dirakit akan diuji performanya untuk mengetahui apakah telah memenuhi spesifikasi yang ditetapkan, seperti *overshoot* <10%, *rise time* <2 menit, *settling time* <5 menit, dan *steady-state error* <5%.

### **III.2 Perancangan *Mini Plant***

Perancangan *mini plant* dilakukan dengan mempertimbangkan aspek fungsionalitas, kemudahan perakitan, serta efisiensi ruang dan biaya. Sistem *mini plant* terdiri dari tiga tangki utama, yaitu tangki air panas, tangki air dingin, dan tangki utama sebagai tempat pencampuran. Ketiga tangki dipasang pada platform berbahan kayu untuk menjaga kestabilan struktur dan memudahkan proses instalasi.

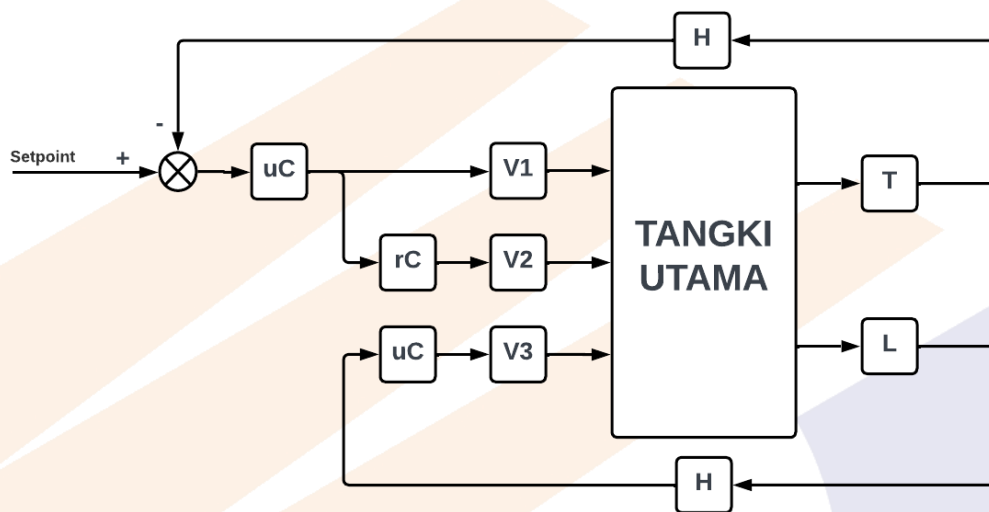
POLBAN



Gambar III.2 Rancangan *Mini Plant*

### III.3 Diagram Alir Kerja Sistem

Diagram alir kerja sistem merupakan representasi visual dari alur proses yang terjadi dalam sistem mini plant ini. Diagram ini membantu dalam memahami tahapan-tahapan operasional yang berlangsung.



Gambar III.3 Diagram Sistem *Mini Plant* MIMO

Diagram blok yang ditampilkan pada Gambar III.3 merupakan representasi dari arsitektur sistem MIMO yang diimplementasikan dalam *mini plant* pengkondisian air. Sistem ini memiliki tiga variabel *input* yang dapat dimanipulasi dan dua variabel *output* yang dikendalikan. Dengan deskripsi *input* dan *output* seperti pada tabel berikut.

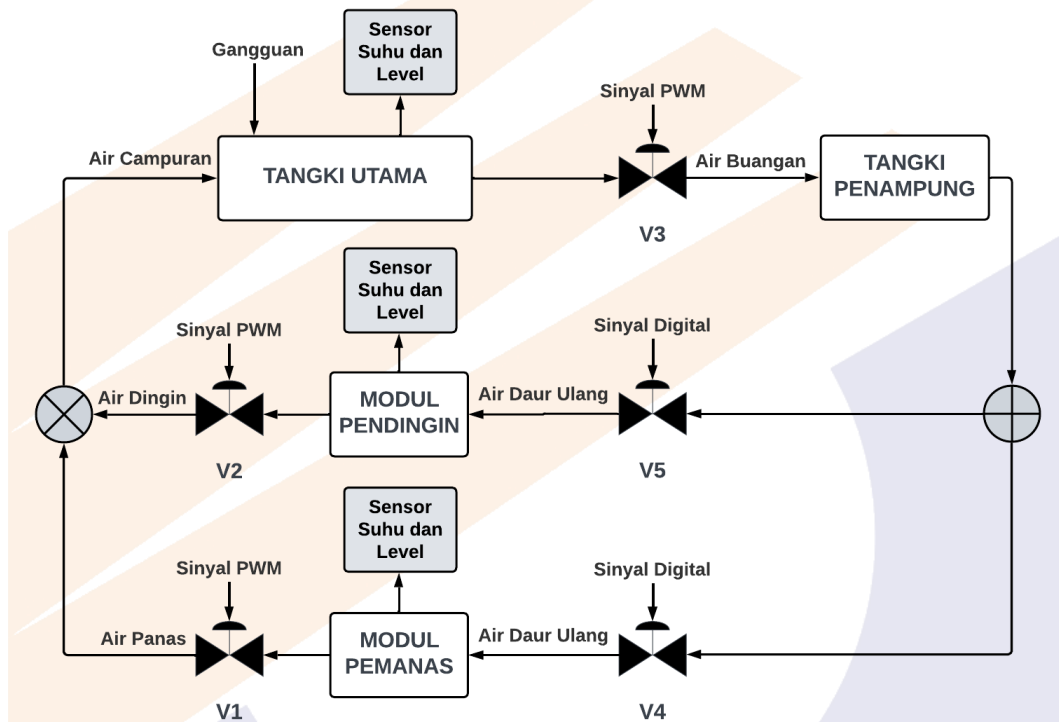
Tabel III.1 *Input* Sistem MIMO

| Input Sistem | Deskripsi                           |
|--------------|-------------------------------------|
| V1           | Bukaan katup servo tangki pemanas   |
| V2           | Bukaan katup servo tangki pendingin |
| V3           | Bukaan katup servo tangki utama     |

Tabel III.2 *Output* Sistem MIMO

| Output Sistem | Deskripsi   |
|---------------|---|
| T             | Suhu tangki, diukur dengan sensor DS18B20.          |
| L             | Ketinggian air, diukur dengan sensor konduktivitas. |

### III.3.1 Diagram Blok *Mini Plant*



Gambar III.4 Diagram Blok *Mini Plant*

Gambar III.4 menunjukkan arsitektur sistem *mini plant* pengkondisian air . Berikut adalah penjelasan tiap bagian:

#### 1. Modul Pemanas

Modul ini berfungsi sebagai sumber air panas. Dilengkapi dengan sensor suhu dan level, serta katup V1 yang dikendalikan oleh sinyal PWM untuk mengatur bukaan air panas menuju tangki utama. Katup V4 berfungsi untuk mengatur sirkulasi ulang air dari tangki penampung ke pemanas, dikendalikan oleh sinyal digital.

#### 2. Modul Pendingin

Modul ini menyediakan air dingin untuk pencampuran. Sama seperti pemanas, dilengkapi sensor suhu dan level, dan katup V2 dengan kontrol PWM untuk mengatur bukaan air dingin ke tangki utama. Katup V5 memungkinkan air dari tangki penampung dipompa kembali ke pendingin, juga dikontrol secara digital.

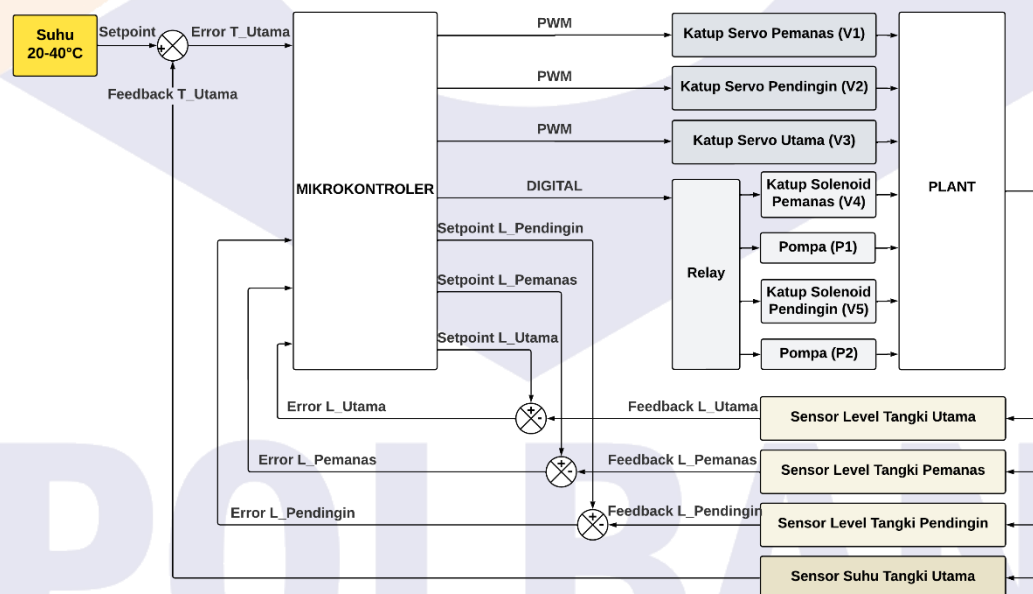
### 3. Tangki Utama

Berfungsi sebagai tempat pencampuran akhir antara air panas dan dingin. Dilengkapi dengan sensor suhu dan level utama untuk memantau kondisi aktual. Katup V3, dikendalikan oleh sinyal PWM, mengatur pembuangan air ke tangki penampung saat level melebihi batas atas. Tangki ini juga memiliki jalur gangguan yang secara eksplisit ditandai pada diagram. Jalur ini digunakan untuk mensimulasikan gangguan eksternal berupa masuknya air dari luar sistem (misalnya kebocoran saluran katup sumber maupun suhu ruang), yang mempengaruhi suhu atau level air. Fungsi ini penting untuk menguji respons sistem terhadap perubahan tak terduga atau kondisi dinamis *non-ideal*.

#### 4. Tangki Penampung

Berfungsi menampung air buangan dari proses di tangki utama dan menyuplai ulang air ke modul pemanas dan pendingin melalui katup V4 dan V5.

### III.3.2 Diagram Blok Sistem Kendali



Gambar III.5 Diagram Blok Sistem Kendali

Diagram blok di atas menunjukkan arsitektur sistem kontrol suhu dan level air yang dikendalikan oleh mikrokontroler. Sistem ini menerima *input* dari pengguna dan sensor, kemudian memproses data tersebut untuk mengendalikan berbagai aktuator seperti katup servo dan relay. Sistem ini dirancang untuk menjaga



kestabilan suhu dan level air dalam plant, yaitu sistem fisik yang terdiri dari tangki pemanas, tangki pendingin, dan tangki utama.

a. *Input Setpoint*

Pada sisi kiri diagram, terdapat *input* suhu yang digunakan untuk memasukkan nilai setpoint suhu air yang diinginkan dengan rentang yang telah ditentukan. Nilai ini menjadi referensi utama bagi mikrokontroler dalam menentukan aksi kontrol terhadap sistem.

b. Mikrokontroler

Mikrokontroler berfungsi sebagai pusat pengendali utama sistem. Perangkat ini menerima *input* dari nilai setpoint dan data sensor, lalu mengeluarkan perintah ke aktuator menggunakan sinyal digital maupun PWM. Tiga jalur PWM digunakan untuk:

- PWM1: Mengontrol katup servo pemanas (V1) untuk mengatur bukaan air panas.
- PWM2: Mengontrol katup servo pendingin (V2) untuk mengatur bukaan air dingin.
- PWM3: Mengontrol katup servo utama (V3) untuk mengatur buangan dari tangki utama.

Mikrokontroler juga mengontrol relay yang mengaktifkan beberapa aktuator.

- Katup solenoid pemanas (V4) dan pendingin (V5) yang berfungsi membuka dan menutup aliran air ke modul pemanas atau pendingin.
- Pompa P1 dan P2 yang masing-masing berfungsi memompa air dari tangki penampung kembali ke tangki sumber.

c. Sensor

Sensor-sensor yang terhubung ke mikrokontroler berfungsi sebagai *feedback* sistem untuk menjamin kontrol yang presisi. Terdapat empat sensor utama:

- Sensor Level Tangki Heater
- Sensor Level Tangki Cooler

- Sensor Level Tangki Utama
- Sensor Suhu Tangki Utama

Sensor-sensor ini memberikan informasi suhu dan ketinggian air secara *real-time* yang digunakan oleh mikrokontroler dalam proses pengambilan keputusan.

#### d. Plant

Bagian plant mencakup seluruh sistem fisik tempat pengolahan air berlangsung, yaitu tangki pemanas, pendingin, serta tangki utama. *Output* dari sistem berupa suhu dan level air.

### III.4 Perancangan Sistem Kendali

Perancangan sistem kendali dilakukan untuk mengatur suhu air pada tangki utama melalui pengaturan bukaan katup pemanas dan pendingin secara proporsional. Sistem kendali ini dibangun menggunakan kombinasi *fuzzy logic* dan *ratio controller* dengan keluaran berupa bukaan katup servo. Sistem dikendalikan secara otomatis dan dijalankan pada mikrokontroler Raspberry Pi Pico.

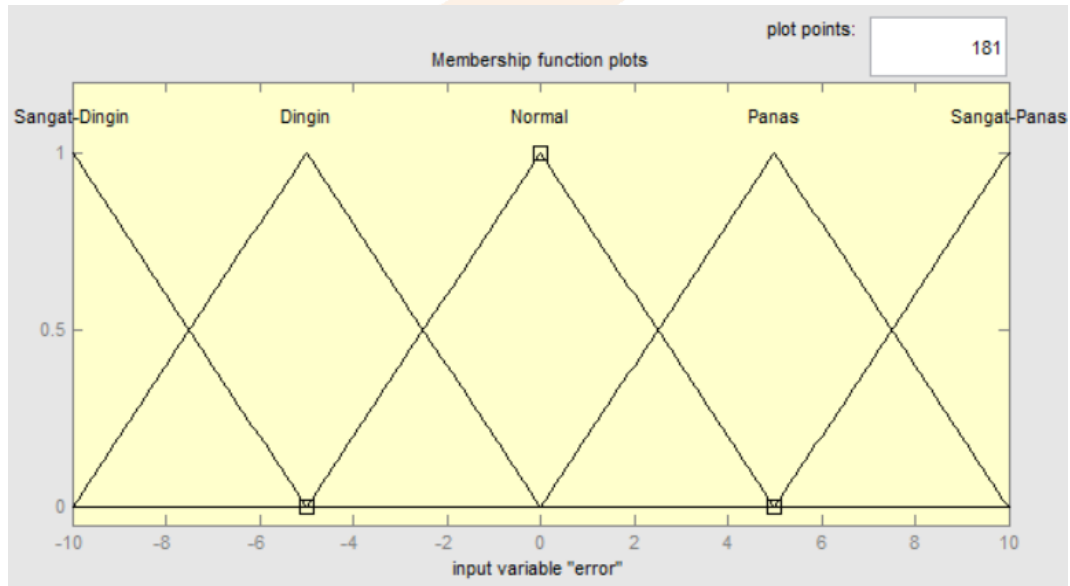
#### III.4.1 Fungsi Keanggotaan dan Basis Aturan *Fuzzy*

Sistem *fuzzy logic* dirancang dengan tipe sugeno karena memiliki komputasi yang lebih efisien dibandingkan metode mamdani, sehingga lebih sesuai untuk platform mikrokontroler seperti Raspberry Pi Pico. Selain itu, sistem sugeno dapat berupa nilai numerik langsung (*crisp value*) yang memudahkan untuk diolah menjadi sinyal PWM.

##### a. Fungsi Keanggotaan

Fungsi keanggotaan yang digunakan adalah segitiga karena bentuk ini sederhana, mudah dihitung, dan sesuai untuk sistem kontrol dasar. Fungsi keanggotaan diterapkan pada dua input, yaitu:

## 1. *Error*



Gambar III.6 Fungsi Keanggotaan *Error*

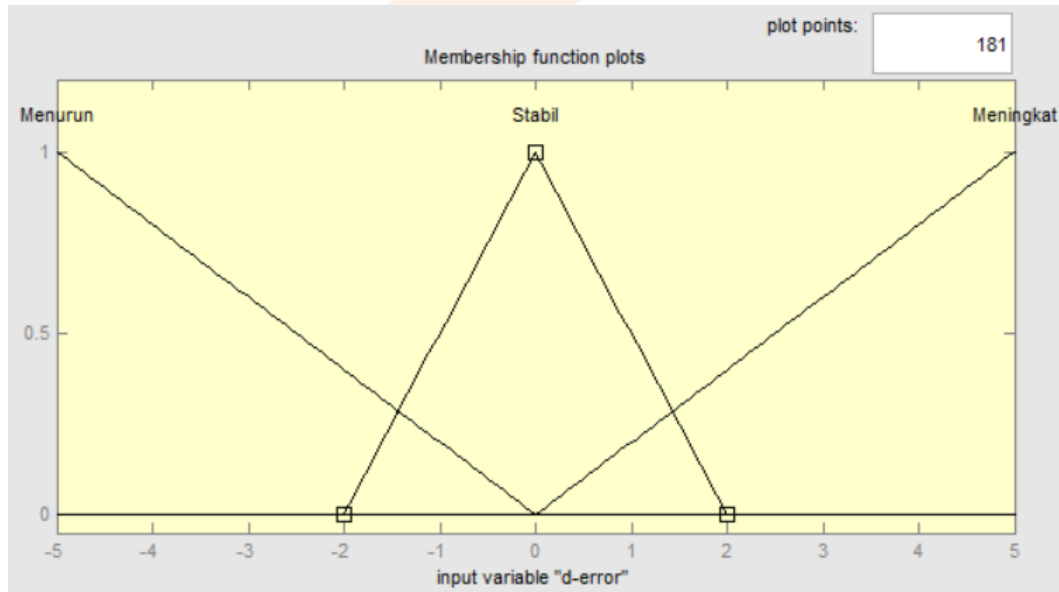
Tabel III.3 Batas Fungsi Keanggotaan *Fuzzy Variabel Error*

| Error         | Titik Bawah | Titik Tengah | Titik Atas |
|---------------|-------------|--------------|------------|
| Sangat Dingin | -15         | -10          | -5         |
| Dingin        | -10         | -5           | 0          |
| Normal        | -5          | 0            | 5          |
| Panas         | 0           | 5            | 10         |
| Sangat Panas  | 5           | 10           | 15         |

Nilai *error* ( $e$ ) dihitung berdasarkan selisih antara suhu setpoint dan suhu aktual di dalam tangki. Fungsi keanggotaan pada variabel *error* ini digunakan untuk mengetahui sejauh mana sistem sedang berada dalam kondisi terlalu dingin, normal, atau terlalu panas.

Sebagai pelengkap dari analisis penyimpangan suhu, sistem juga memperhatikan arah dan laju perubahan *error* melalui variabel  $\Delta error$  ( $\Delta e$ ). Dengan demikian, pengendali dapat lebih responsif terhadap dinamika perubahan suhu, tidak hanya pada nilai *error* saat ini.

## 2. $\Delta Error$



Gambar III.7 Fungsi Keanggotaan  $\Delta Error$

Tabel III.4 Batas Fungsi Keanggotaan *Fuzzy* Variabel  $\Delta Error$

| $\Delta Error$ | Titik Bawah | Titik Tengah | Titik Atas |
|----------------|-------------|--------------|------------|
| Menurun        | -10         | -5           | 0          |
| Stabil         | -2          | 0            | 2          |
| Meningkat      | 0           | 5            | 10         |

Dengan memanfaatkan fungsi keanggotaan untuk  $\Delta error$ , pengendali *fuzzy* dapat mengetahui apakah suhu sedang naik, tetap, atau menurun.

Adapun *output* dari sistem *fuzzy* ini berupa nilai bukaan air panas terhadap bukaan air dingin yang berkisar antara 0 hingga 1, di mana nilai mendekati 1 berarti dominan air panas, dan mendekati 0 berarti dominan air dingin.

Tabel III.5 Nilai *Crisp Output* Berdasarkan Kategori *Fuzzy*

| <i>Output</i> | Value |
|---------------|-------|
| Maksimum      | 0.00  |
| Sangat Tinggi | 0.10  |
| Tinggi        | 0.25  |
| Agak Tinggi   | 0.35  |
| Sedang        | 0.50  |

|               |      |
|---------------|------|
| Sedang Rendah | 0.65 |
| Rendah        | 0.75 |
| Sangat Rendah | 0.90 |
| Minimum       | 1.00 |

a. Basis Aturan (*Rule Base*)

Rule base disusun berdasarkan pemahaman logis terhadap bagaimana sistem harus merespons perubahan suhu. Berikut adalah aturan yang telah dibuat:

Tabel III.6 *Fuzzy Rule*

| <i>Error</i>  | <i><math>\Delta Error</math></i> | <i>Output</i> |
|---------------|----------------------------------|---------------|
| Sangat Dingin | Menurun                          | Maksimum      |
| Sangat Dingin | Stabil                           | Maksimum      |
| Sangat Dingin | Meningkat                        | Sangat Tinggi |
| Dingin        | Menurun                          | Sangat Tinggi |
| Dingin        | Stabil                           | Tinggi        |
| Dingin        | Meningkat                        | Cukup Tinggi  |
| Normal        | Menurun                          | Cukup Tinggi  |
| Normal        | Stabil                           | Sedang        |
| Normal        | Meningkat                        | Rendah        |
| Panas         | Menurun                          | Rendah        |
| Panas         | Stabil                           | Cukup Rendah  |
| Panas         | Meningkat                        | Sangat Rendah |
| Sangat Panas  | Menurun                          | Sangat Rendah |
| Sangat Panas  | Stabil                           | Minimum       |
| Sangat Panas  | Meningkat                        | Minimum       |

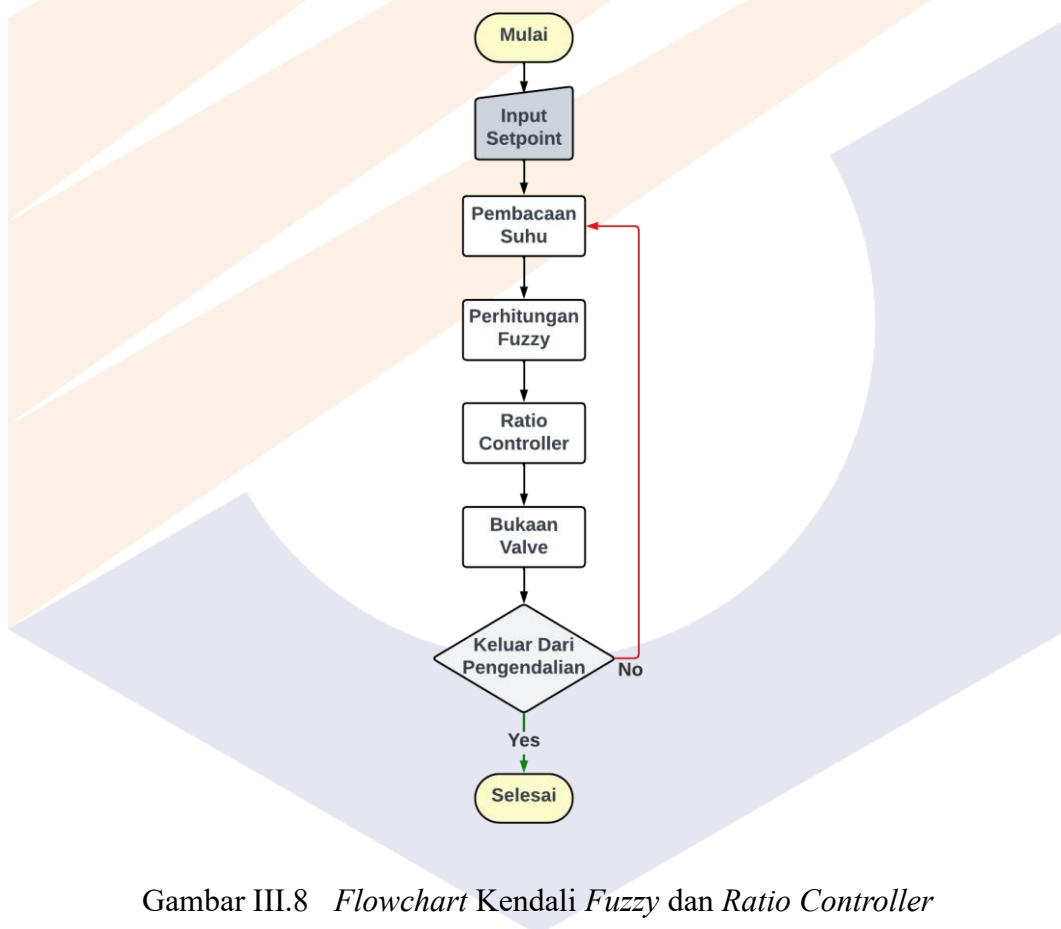
Aturan-aturan ini digunakan untuk membentuk sistem inferensi *fuzzy* yang akan menghasilkan nilai *output* sesuai dengan kondisi suhu aktual.

#### III.4.2 Perancangan Kontrol

Perancangan kontrol dalam sistem ini dibagi menjadi dua bagian utama, yaitu pengendalian suhu menggunakan kombinasi *fuzzy logic* dan *ratio controller*, serta pengendalian level air secara digital berdasarkan kondisi ketinggian dalam tangki utama. Kedua sistem kontrol ini bekerja secara paralel namun terkoordinasi dalam satu sistem MIMO yang saling memengaruhi.

#### III.4.2.1 Pengendalian *Fuzzy* dan *Ratio Controller*

Sistem pengendalian suhu pada *mini plant* ini dirancang untuk mengatur proporsi pencampuran antara air panas dan air dingin guna mencapai suhu target di dalam tangki utama. Pengendalian diimplementasikan secara terintegrasi dalam mikrokontroler Raspberry Pi Pico.



Gambar III.8 *Flowchart* Kendali *Fuzzy* dan *Ratio Controller*

Gambar III.8 menunjukkan *flowchart* alur kerja pengendalian *fuzzy* dan *ratio*, dimulai dari proses *input* setpoint oleh pengguna, pembacaan suhu aktual, perhitungan *fuzzy logic*, pemrosesan *ratio*, pengaturan bukaan katup, hingga evaluasi kondisi untuk melanjutkan atau keluar dari mode pengendalian.

##### III.4.2.1.1 Prinsip Kerja Kontrol *Fuzzy*

Pada sistem pengendalian suhu berbasis *fuzzy logic* ini, proses dimulai dari pembacaan dua variabel *input* utama, yaitu:

- *Error* ( $e$ ): selisih antara nilai setpoint suhu dengan suhu aktual



$$e = T_{setpoint} - T_{aktual} \quad (III.1)$$

- $\Delta Error (\Delta e)$ : perubahan *error* dari waktu ke waktu

$$\Delta e = e(t) - e(t - 1) \quad (III.2)$$

Kedua variabel ini mencerminkan kondisi dinamis sistem secara *real-time* dan menjadi dasar dalam proses pengambilan keputusan kontrol.

Nilai *input* numerik dari *error* dan  $\Delta error$  diproses melalui tahap fuzzifikasi, yaitu dikonversi menjadi derajat keanggotaan dalam himpunan *fuzzy*. Fungsi keanggotaan yang digunakan adalah fungsi segitiga dengan parameter titik bawah (a), titik tengah (b) dan titik atas (c), dan dirumuskan sebagai:

- Parameter *error* (e)

$$\mu_{e_i}(e) = \begin{cases} 0, & e \leq a \\ \frac{e - a}{b - a}, & a < e \leq b \\ \frac{c - e}{c - b}, & b < e < c \\ 0, & e \geq c \end{cases} \quad (III.3)$$

- Parameter  $\Delta error (\Delta e)$

$$\mu_{\Delta e_j}(\Delta e) = \begin{cases} 0, & \Delta e \leq a \\ \frac{\Delta e - a}{b - a}, & a < \Delta e \leq b \\ \frac{c - \Delta e}{c - b}, & b < \Delta e < c \\ 0, & e \geq c \end{cases} \quad (III.4)$$

Setelah itu, dilakukan evaluasi aturan (*rule evaluation*) berdasarkan *rule base* yang telah dirancang. Setiap kombinasi nilai *error* dan  $\Delta error$  akan mengaktifkan satu atau lebih aturan *fuzzy* dengan derajat kepercayaan tertentu.

Setiap aturan *fuzzy* berbentuk:

$$IF \ e \text{ is } A_i \text{ AND } \Delta e \text{ is } B_j \text{ THEN output is } z_{i,j} \quad (III.5)$$

Derajat aktivasi dari setiap aturan dihitung menggunakan operator AND (fungsi minimum):

$$output_{i,j} = \min (\mu_{e_i}(e), \mu_{\Delta e_j}(\Delta e)) \quad (III.6)$$

Tahap akhir defuzzifikasi menggunakan metode *weighted average* untuk menghasilkan satu *value* yang menentukan sudut bukaan katup servo air panas melalui sinyal PWM.

$$percentage = \frac{\sum output_{i,j} * value_{i,j}}{\sum output_{i,j}} \quad (III.7)$$

- Contoh Perhitungan *Fuzzy*

Diberikan kondisi sebagai berikut:

- *Setpoint suhu* = 30°C
- *Suhu Aktual* = 27°C
- *Error sebelumnya* = 2°C

Maka, nilai *error* saat ini dihitung sebagai:

$$e = T_{setpoint} - T_{aktual} = 30 - 27 = 3^\circ\text{C}$$

Selanjutnya, nilai  $\Delta error$  dihitung sebagai perubahan nilai *error* terhadap waktu sebelumnya:

$$\Delta e = e(t) - e(t - 1) = 3 - 2 = 1^\circ\text{C}$$

Langkah berikutnya adalah mentransformasikan nilai numerik dari *error* dan  $\Delta error$  ke dalam bentuk linguistik menggunakan fungsi keanggotaan segitiga berdasarkan tabel *fuzzy* yang telah ditentukan.

a) *Fuzzifikasi Error* = 3°C:

Dari tabel keanggotaan:

- *Normal*: (-5, 0, 5)
- *Panas*: (0, 5, 10)\

Maka derajat keanggotaannya adalah:

$$\mu_{Normal}(3) = \frac{5-3}{5-0} = \frac{2}{5} = 0.4$$

$$\mu_{Panas}(3) = \frac{3-0}{5-0} = \frac{3}{5} = 0.6$$

b)  $\Delta Error = 1^{\circ}C$

Dari tabel keanggotaan:

- *Stabil*:  $(-2, 0, 2)$
- *Meningkat*:  $(0, 5, 10)$

Maka derajat keanggotaannya:

$$\mu_{Stabil}(1) = \frac{2-1}{2-0} = \frac{1}{2} = 0.5$$

$$\mu_{Meningkat}(1) = \frac{1-0}{5-0} = \frac{1}{5} = 0.2$$

Selanjutnya, nilai derajat keanggotaan digunakan untuk mengaktifkan aturan-aturan dalam *rule base fuzzy*. Aktivasi aturan dihitung dengan menggunakan operator logika AND yang diimplementasikan menggunakan fungsi minimum (min).

Tabel di bawah ini menunjukkan kombinasi aturan yang aktif berdasarkan nilai fuzzifikasi:

Tabel III.7 Contoh Perhitungan Evaluasi Aturan

| <i>Error</i> | <i>ΔError</i> | <i>Output</i> | <i>z (Value)</i> | <i>Bobot Aktivasi</i>  |
|--------------|---------------|---------------|------------------|------------------------|
| Normal       | Stabil        | Sedang        | 0.50             | $\min(0.4, 0.5) = 0.4$ |
| Normal       | Meningkat     | Rendah        | 0.75             | $\min(0.4, 0.2) = 0.2$ |
| Panas        | Stabil        | Cukup Rendah  | 0.65             | $\min(0.6, 0.5) = 0.5$ |
| Panas        | Meningkat     | Sangat Rendah | 0.90             | $\min(0.6, 0.2) = 0.2$ |

Untuk memperoleh nilai *output* akhir dari sistem *fuzzy*, dilakukan proses defuzzifikasi dengan metode *weighted average*. *Output crisp* dihitung dengan rumus:

$$z = \frac{(0.4 \cdot 0.5) + (0.75 \cdot 0.2) + (0.65 \cdot 0.5) + (0.9 \cdot 0.2)}{0.4 + 0.2 + 0.5 + 0.2}$$

$$z = 0.6577$$

Dengan menggunakan metode defuzzifikasi ini, nilai *output crisp* yang diperoleh adalah sebesar 0.6577. Nilai ini merepresentasikan hasil akhir dari proses pengambilan keputusan sistem *fuzzy* berdasarkan kombinasi derajat keanggotaan dan nilai konsekuen pada setiap aturan yang aktif.

#### III.4.2.1.2 Prinsip Kerja *Ratio Controller*

*Ratio controller* merupakan metode pengendalian yang digunakan untuk mengatur perbandingan dua aliran dalam suatu sistem pencampuran, dalam hal ini antara bukaan katup servo air panas dan air dingin. Prinsip utama dari *ratio control* adalah menjaga total aliran konstan, tetapi membagi besarnya secara proporsional berdasarkan nilai referensi dari *fuzzy logic controller*.

$$Bukaan_{pemanas} = z \quad (III.8)$$

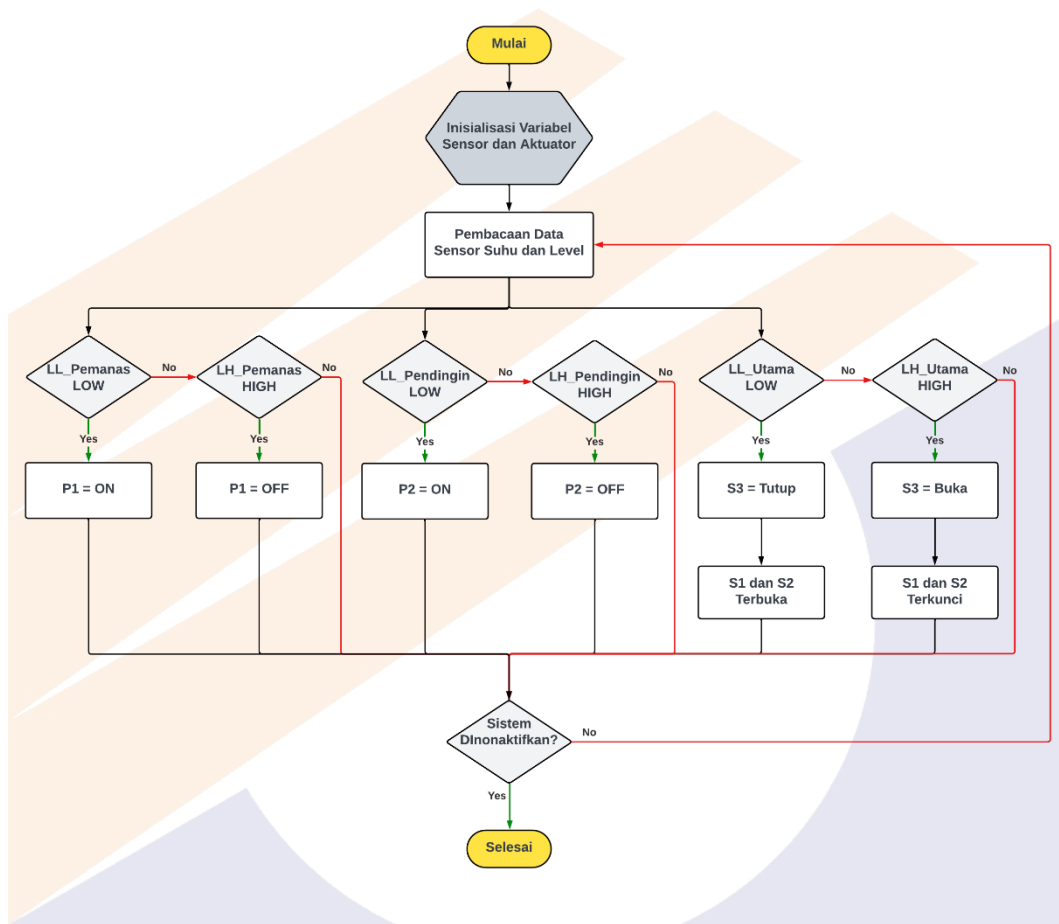
$$Bukaan_{pendingin} = 1 - z \quad (III.9)$$

Dengan:

- $z \in [0 \ 1]$
- $z = 0 \rightarrow 100\%$  air dingin
- $z = 1 \rightarrow 100\%$  air panas
- $z = 0.5 \rightarrow$  campuran seimbang 50% air panas dan air dingin

#### III.4.2.2 Pengendalian Level

Sistem pengendalian level pada *mini plant* ini bertujuan untuk menjaga ketinggian air di masing-masing tangki agar tetap berada dalam batas operasional yang aman. Pengendalian dilakukan secara otomatis berdasarkan pembacaan sensor level konduktivitas dua titik, yaitu *Low Level* (LL) dan *High Level* (LH), yang dipasang pada setiap tangki.



Gambar III.9 Flowchart Kendali Level

Sistem dirancang untuk menjamin ketersediaan air di tangki sumber melalui kontrol pompa otomatis, menjaga volume air dalam tangki utama tetap stabil selama pencampuran, mencegah *overflow* dengan pembuangan otomatis saat level tinggi terdeteksi, serta melindungi proses pencampuran dari kekosongan tangki yang dapat mengganggu kinerja kontrol suhu *fuzzy*.

POLBAN

## BAB IV

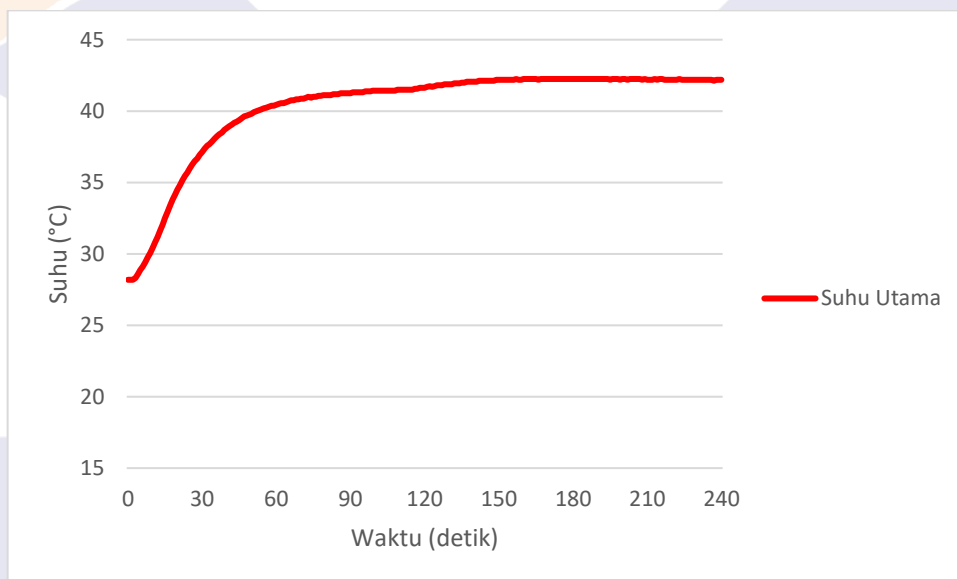
### HASIL DAN PEMBAHASAN

#### IV.1 Hasil Pengujian Sistem

Pengujian dilakukan untuk mengevaluasi performa sistem kendali suhu berbasis *fuzzy logic* dan *ratio controller* terhadap berbagai skenario operasional. Respon yang diamati meliputi kestabilan suhu terhadap setpoint, waktu pencapaian setpoint (*settling time*), tingkat *overshoot*, dan kemampuan sistem merespons perubahan setpoint saat kontrol berlangsung.

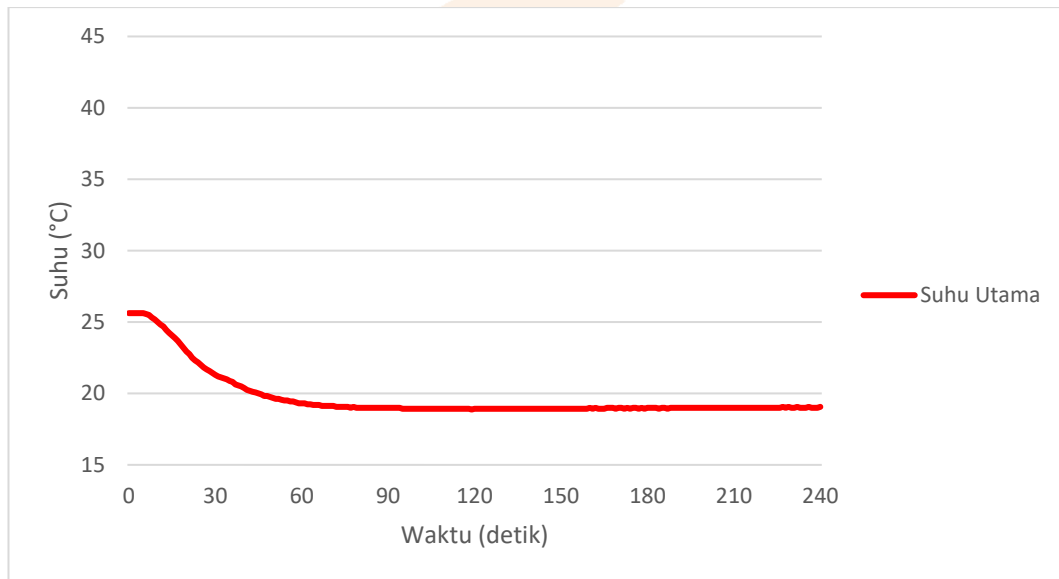
##### IV.1.1 Respon *Open Loop*

Pengujian open loop dilakukan untuk mengevaluasi karakteristik dasar sistem tanpa adanya umpan balik kendali. Dalam kondisi ini, bukaan katup air panas dan dingin diatur secara manual tanpa intervensi dari logika kontrol. Tujuannya adalah untuk memahami dinamika sistem fisik, seperti waktu tunda, laju perubahan suhu, serta pengaruh proporsi bukaan katup servo terhadap suhu akhir di tangki utama.



Gambar IV.1 Respon Suhu terhadap Bukaan Katup Pemanas Tetap

Grafik menunjukkan kenaikan suhu dari 28.2°C hingga 42.2°C ketika katup pemanas dibuka secara tetap tanpa kontrol. Laju kenaikan suhu tinggi di awal, lalu melambat mendekati kondisi *steady-state* pada detik ke-150. Tidak terdapat *overshoot* atau koreksi karena sistem tidak memiliki umpan balik.



Gambar IV.2 Respon Suhu terhadap Buka-an Katup Pendingin Tetap

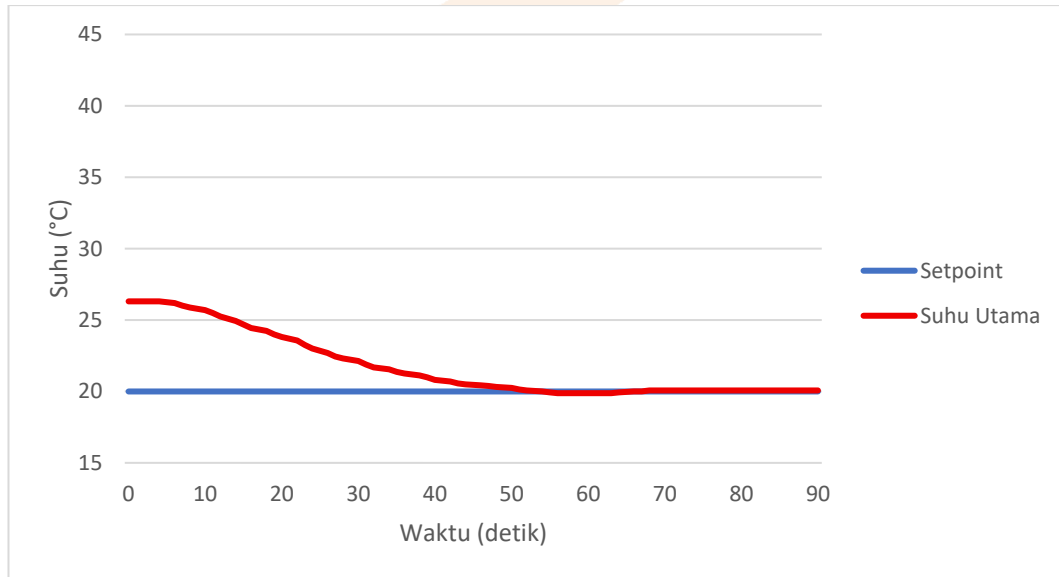
Grafik menunjukkan penurunan suhu dari sekitar 25.6°C menuju 19°C saat katup pendingin dibuka secara manual. Suhu mencapai kestabilan pada detik ke-80. Respon ini menunjukkan sifat pendinginan pasif tanpa adaptasi terhadap kondisi dinamis sistem.

#### IV.1.2 Respon *Close Loop*

Pengujian respon *close loop* dilakukan untuk mengevaluasi kinerja sistem kendali dalam mengikuti perubahan setpoint suhu secara otomatis. Pada pengujian ini, sistem diuji terhadap tiga kondisi: setpoint tetap pada 20°C, setpoint tetap pada 40°C, dan setpoint dengan variasi tertentu. Tujuan dari pengujian ini adalah untuk melihat seberapa cepat dan stabil sistem merespons perubahan setpoint, serta mengevaluasi parameter performa seperti *overshoot*, *rise time*, *settling time*, dan *steady-state error*. Hasil dari masing-masing skenario pengujian akan dijelaskan pada bagian berikut:



- Setpoint 20°C

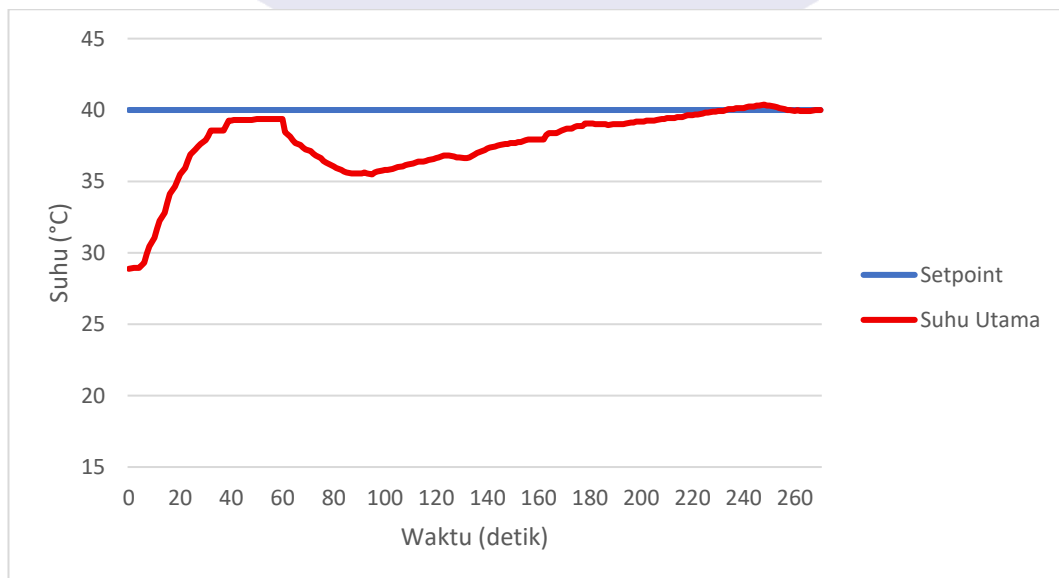


Gambar IV.3 Grafik Respon Suhu terhadap Setpoint 20°C

Analisis:

1. Pengujian dilakukan dengan setpoint 20°C dan suhu awal 26.31°C.
2. Sistem menunjukkan penurunan suhu secara bertahap hingga mendekati nilai setpoint pada detik ke-54.
3. Tidak terjadi overshoot, dan suhu stabil setelah waktu tersebut.

- Setpoint 40°C



Gambar IV.4 Grafik Respon Suhu terhadap Setpoint 40°C

Analisis:

1. Uji dilakukan dengan setpoint  $40^{\circ}\text{C}$  dan suhu awal  $28.88^{\circ}\text{C}$ .
2. Suhu meningkat secara cepat hingga mendekati  $40^{\circ}\text{C}$  dalam waktu 40 detik,
3. Setelah itu, suhu sempat turun hingga  $35.5^{\circ}\text{C}$ , lalu perlahan naik kembali dan stabil di  $40^{\circ}\text{C}$ .
4. Respon sistem menunjukkan osilasi dan *overshoot*, tetapi mampu kembali stabil setelah 240 detik.

- Setpoint Dinamis

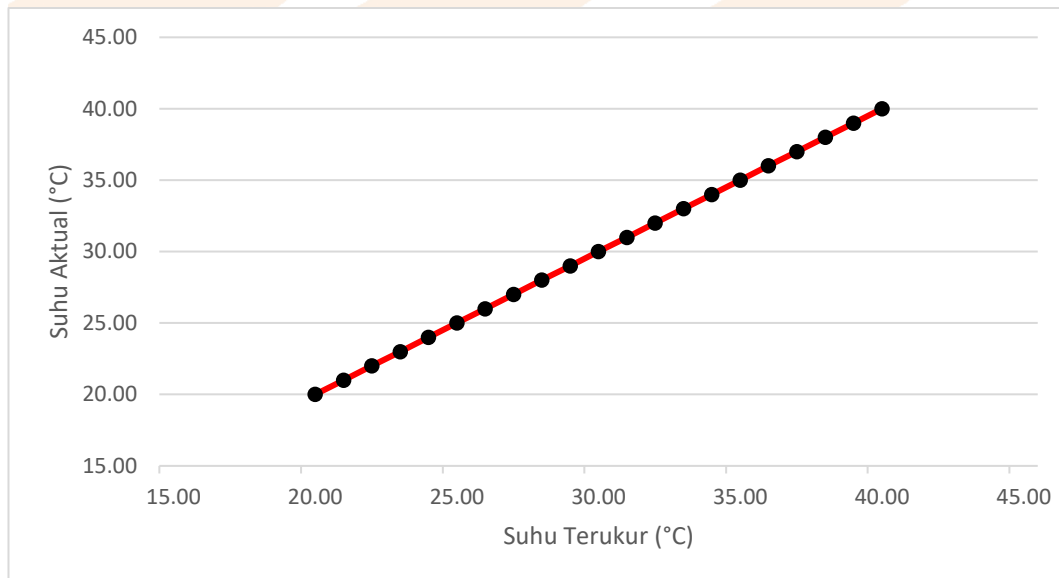


Gambar IV.5 Grafik Respon Suhu terhadap Setpoint Dinamis

1. Dalam uji ini, sistem dijalankan dengan beberapa perubahan setpoint secara *real-time*, dimulai dari  $30^{\circ}\text{C}$  kemudian  $36^{\circ}\text{C}$  lalu  $28^{\circ}\text{C}$  dan berakhir pada  $32^{\circ}\text{C}$ .
2. Suhu aktual berhasil menyesuaikan terhadap setiap perubahan setpoint dengan respon transien yang halus, meskipun terdapat sedikit lag dan osilasi pada penurunan suhu drastis ( $36^{\circ}\text{C} \rightarrow 28^{\circ}\text{C}$ ).
3. Sistem menunjukkan fleksibilitas dan kemampuan adaptasi yang baik terhadap variasi *input*.
4. Waktu penyesuaian untuk tiap perubahan setpoint berkisar antara 30–50 detik, tergantung pada arah perubahan dan perbedaan suhu.

#### IV.1.3 Validasi Sensor Suhu DS18B20

Untuk memastikan keandalan sistem pengukuran suhu dalam pengendalian ini, dilakukan validasi akurasi sensor DS18B20 terhadap termometer analog sebagai alat ukur referensi. Pengujian dilakukan dengan membandingkan pembacaan sensor dan termometer pada beberapa titik suhu representatif.



Gambar IV.6 Perbandingan Suhu DS18B20 terhadap Suhu Termometer

Hasil pengujian menunjukkan bahwa pembacaan sensor DS18B20 menunjukkan tren linier yang konsisten dengan pembacaan termometer analog pada rentang suhu 20°C hingga 40°C dengan kenaikan 1°C.

Validasi ini memperkuat bahwa hasil kendali suhu yang ditampilkan dalam pengujian *close loop* benar-benar merefleksikan kondisi aktual di dalam tangki utama dan bukan merupakan hasil pengukuran yang bias atau tidak reliabel.

#### IV.2 Evaluasi Kinerja Sistem Kendali Suhu

Tabel IV.1 Evaluasi Kinerja Sistem Kendali Suhu

| No | Suhu awal (°C) | Setpoint (°C) | Overshoot (%)  | Rise time (s)     | Settling time (s)    | Steady-state error (%) |
|----|----------------|---------------|----------------|-------------------|----------------------|------------------------|
| 1  | 26.31          | 20            | 0              | 54                | 54                   | 0.3                    |
| 2  | 28.88          | 40            | 0.95           | 40                | 240                  | 0                      |
| 3  | Dinamis        | 30→36→28→32   | 3.2 (maksimal) | 30-50 (rata-rata) | 60-90 (per transisi) | 0.97 (rata-rata)       |

Penjelasan Parameter:

- *Rise time*: Waktu yang dibutuhkan suhu untuk pertama kali mencapai nilai setpoint dari kondisi awal (tanpa mempertimbangkan *overshoot*).
- *Overshoot*: Persentase kenaikan maksimum suhu yang melebihi setpoint (hanya terjadi pada uji 2 & 3).
- *Settling time*: Waktu yang dibutuhkan suhu untuk stabil dalam rentang toleransi  $\pm 5\%$  dari setpoint.
- *Steady-state error*: Selisih rata-rata antara suhu akhir sistem dengan setpoint setelah sistem stabil.

POLBAN

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **V.1 Kesimpulan**

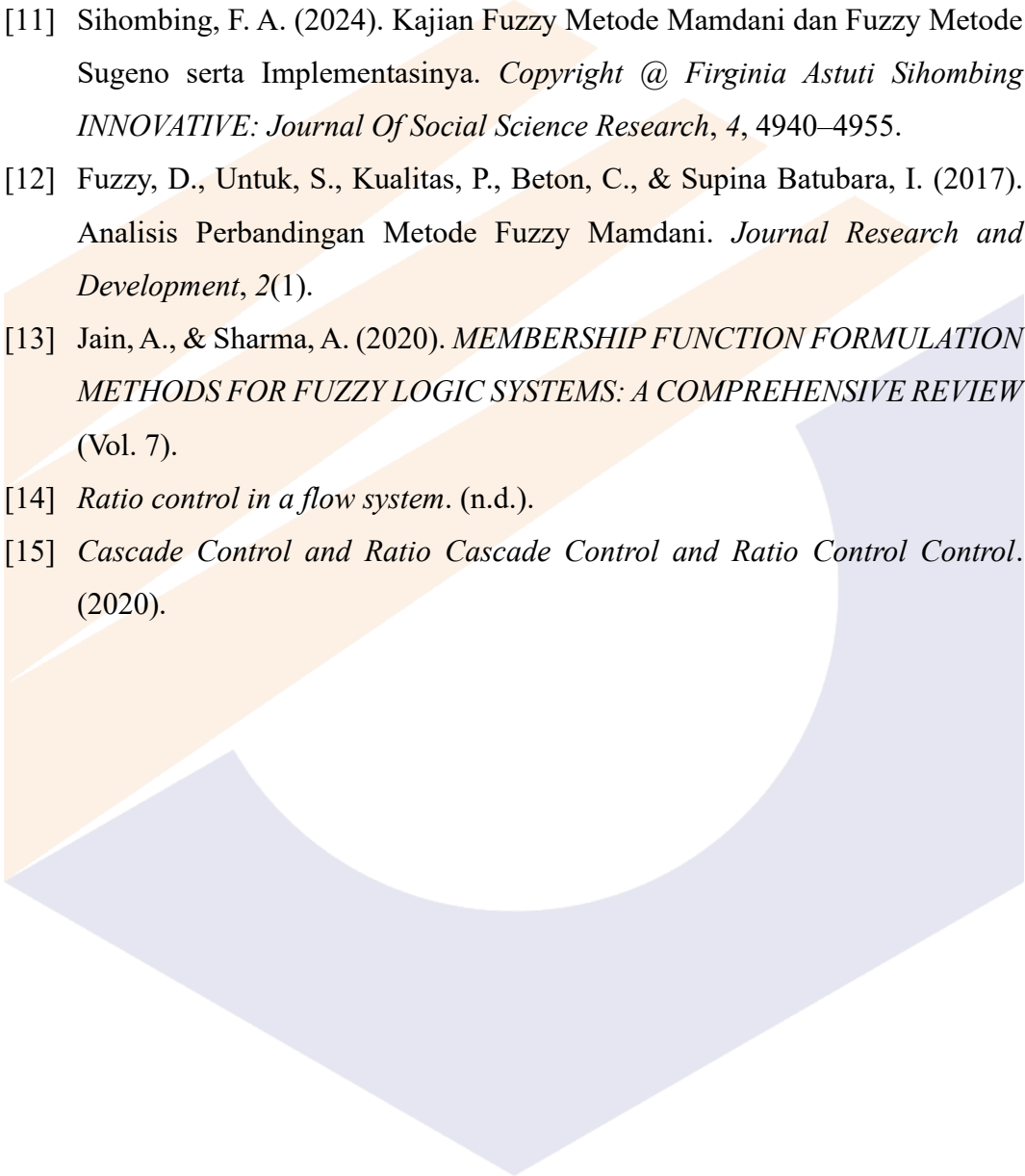
Berdasarkan hasil perancangan, implementasi, dan pengujian, dapat disimpulkan bahwa seluruh tujuan penelitian tercapai. *Mini plant* skala laboratorium berhasil dibangun secara fungsional, terdiri dari sistem pemanas, pendingin, tangki utama, sensor suhu dan level, serta aktuator servo dan solenoid valve dalam struktur modular. Sistem pengendalian otomatis suhu dan level direalisasikan menggunakan mikrokontroler Raspberry Pi Pico dengan konfigurasi MIMO, menggabungkan pengendalian suhu berbasis *fuzzy logic* dan *ratio controller*, serta pengendalian level berbasis sensor konduktivitas dua titik. Hasil pengujian menunjukkan sistem mampu merespons perubahan setpoint suhu dengan cepat dan stabil, dengan *overshoot* <10%, *rise time* <2 menit, *settling time* <5 menit, dan *steady-state error* <5%.

#### **V.2 Saran**

- Integrasi IoT untuk memungkinkan pemantauan dan pengendalian suhu secara *real-time* dari jarak jauh melalui aplikasi atau web.
- Penambahan kontrol suhu pada tangki sumber air panas dan dingin agar suhu air yang masuk stabil, meningkatkan konsistensi pencampuran dan akurasi pengendalian suhu di tangki utama.
- Memasang sensor level pada tangki penampung untuk memantau volume air otomatis, mencegah kekurangan atau kelebihan air, dan menjaga kestabilan operasi.

## DAFTAR PUSTAKA

- [1] Febiawan, I., Lilansa, N., & Salam, A. (2020). *Rancang Bangun Kendali Suhu Pemanas Tangki Berpengaduk Kontinyu menggunakan Pengendali PID berbasis NI ELVIS II dan LabVIEW*. <https://journal.uny.ac.id/index.php/jee>
- [2] Zulrahma, Y., & Triyanto, R. H. (2024). Pengendalian Temperatur Fluida dengan Menggunakan Kontrol On-Off Berbasis IoT. *INHARDWARE: Journal of Instrumentation and Hardware*, 2(1), 1–7. <https://doi.org/10.53026/inhardware.v2i1.31>
- [3] Mumpuni, S. S., & Astutik, R. P. (2025). Implementation of Raspberry Pi PICO as PLC for Monitoring and Control in Swiftlet Cultivation Based on Weintek HMI. *G-Tech: Jurnal Teknologi Terapan*, 9(1), 266–274. <https://doi.org/10.70609/gtech.v9i1.6054>
- [4] Kristiawan, I. P. A., Rifa'i, M., & Dewatama, D. (2021). Penerapan Rasio Kontrol Untuk Pewarnaan Susu Pada Mini Plant Pengolahan Susu Menggunakan DCS (Distributed Control System) PCS 7 Siemens. *Jurnal Elektronika Dan Otomasi Industri*, 8(1), 83. <https://doi.org/10.33795/elk.v8i1.231>
- [5] Sulistyopambudi, Y., Rifa'i, M., & Putri, R. I. (2024). *IMPLEMENTASI RASIO KONTROL PROSES BAHAN CAIR PADA MINI-PLANT MULTIPRODUK MULTI-JALUR BAGIAN PENGISIAN*.
- [6] Nastiti, N. S. (2021). *PERANCANGAN RASIO KONTROL PADA RUANG BAKAR GENSET BERBAHAN BAKAR BIOGAS*.
- [7] Pramudito, A., Wanarti, P. R., Rohman, M., & Anifah Afiliati, L. (2025). *Analisis dan Simulasi Sistem Kontrol Suhu Otomatis Berbasis Fuzzy Logic*.
- [8] Kurniawan, F., Zetta Maulana, Y., & Farrid Christianti, R. (2022). *Sistem Kendali Level Ketinggian Air Berbasis Fuzzy Control Menggunakan Simulink*.
- [9] Roza, E., Mujirudin, M., Kunci, K., Tinggi, K., & Kanal, K. (2013). *Sistem Mimo dan Aplikasi Penggunaannya* (Vol. 6, Issue 2).
- [10] Maxim Integrated Products, I. (2019). *DS18B20*.



- 
- [11] Sihombing, F. A. (2024). Kajian Fuzzy Metode Mamdani dan Fuzzy Metode Sugeno serta Implementasinya. *Copyright @ Firginia Astuti Sihombing INNOVATIVE: Journal Of Social Science Research*, 4, 4940–4955.
- [12] Fuzzy, D., Untuk, S., Kualitas, P., Beton, C., & Supina Batubara, I. (2017). Analisis Perbandingan Metode Fuzzy Mamdani. *Journal Research and Development*, 2(1).
- [13] Jain, A., & Sharma, A. (2020). *MEMBERSHIP FUNCTION FORMULATION METHODS FOR FUZZY LOGIC SYSTEMS: A COMPREHENSIVE REVIEW* (Vol. 7).
- [14] *Ratio control in a flow system*. (n.d.).
- [15] *Cascade Control and Ratio Cascade Control and Ratio Control Control*. (2020).



POLBAN









## LAMPIRAN




### Lampiran 1. Panduan Penggunaan Alat

| <b>Deskripsi:</b><br><br><i>Mini plant</i> sistem pengkondisi air merupakan sistem yang terintegrasi secara otomatis untuk mengendalikan variabel suhu dan ketinggian air agar tetap stabil sesuai dengan nilai setpoint. |   |   |
|---|---|---|
| Proses  | Langkah Kerja   |   |
| <b>Pengisian Air pada <i>Mini Plant</i></b>   | <ol style="list-style-type: none"> <li>1. Siapkan air bersih dalam botol.</li> <li>2. Tuangkan air dari botol ke dalam tangki penampung.</li> </ol>   |   |
| <b>Prosedur Pengaktifan Sistem</b>  | <ol style="list-style-type: none"> <li>1. Pastikan kabel sumber AC sudah terhubung dengan benar ke sistem.</li> <li>2. Aktifkan sistem dengan menekan saklar daya pada .</li> <li>3. Hubungkan adaptor 5V ke stop kontak pada plant pengkondisi air untuk memberikan catu daya pada panel kontrol dan katup servo.</li> </ol> | <ol style="list-style-type: none"> <li>1. </li> <li>2. </li> </ol> |

|  |   |   |
|--|---|---|
| <p><b>Navigasi Menu</b></p>                    | <p>Gunakan tombol “A” untuk berpindah ke menu sebelumnya (atas).</p> <p>Gunakan tombol “B” untuk berpindah ke menu berikutnya (bawah).</p>  |              |
| <p><b>Pengisian Air pada Tangki Sumber</b></p> | <ol style="list-style-type: none"> <li>1. Saat sistem aktif, air dari tangki penampung akan dialirkan secara otomatis oleh pompa ke tangki sumber.</li> <li>2. Jika pompa masih menyala namun air di tangki penampung sudah habis, segera isi kembali air ke tangki penampung hingga tangki sumber penuh.</li> <li>3. Pompa akan mati secara otomatis setelah kondisi tersebut tercapai.</li> </ol>       |   |
| <p><b>Pengendalian Otomatis</b></p>            | <ol style="list-style-type: none"> <li>1. Pilih menu “<b>Setpoint Utama</b>” pada panel kontrol, lalu tekan tombol “#” untuk melanjutkan.</li> <li>2. Masukkan nilai setpoint yang diinginkan, lalu tekan tombol “#” untuk konfirmasi.</li> <li>3. Tekan tombol “#” sekali lagi untuk memulai proses pengendalian otomatis.</li> <li>4. Sistem akan mengendalikan bukaan katup secara otomatis</li> </ol> | <p>1.</p>  |

|  |   |   |
|--|---|---|
|  | <p>hingga suhu air pada tangki utama mencapai nilai <i>setpoint</i>.</p> <p>5. Jika ingin mengubah <i>setpoint</i> selama proses berlangsung, tekan tombol “#”, lalu masukkan nilai <i>setpoint</i> baru.</p> <p>6. Tekan tombol “#” untuk menjalankan sistem dengan <i>setpoint</i> yang telah diperbarui.</p> <p>7. Untuk menghentikan pengendalian otomatis, tekan tombol “*”.</p> |   |
|  |   | <p>2.</p>    |
|  |   | <p>3.</p>   |
|  |   | <p>4.</p>  |

|                                    |  |  |
|------------------------------------|--|--|
| <p><b>Monitoring Suhu Air</b></p>  | <ol style="list-style-type: none"> <li>1. Pilih menu “<b>Sensor Suhu</b>” pada panel kontrol, lalu tekan tombol “#” untuk melanjutkan.</li> <li>2. Hasil pembacaan sensor akan langsung ditampilkan pada <i>display</i> dan dapat dipantau secara <i>real-time</i> dengan pembaruan setiap 1 detik.</li> <li>3. Untuk keluar dari menu monitoring suhu air, tekan tombol “*”.</li> </ol> | <p>1.</p>  <p>2.</p>  |
| <p><b>Monitoring Level Air</b></p> | <ol style="list-style-type: none"> <li>1. Pilih menu “<b>Sensor Level</b>” pada panel kontrol, lalu tekan tombol “#” untuk melanjutkan.</li> <li>2. Hasil pembacaan sensor akan langsung ditampilkan pada <i>display</i> dan dapat dipantau secara <i>real-</i></li> </ol>   | <p>1.</p>   |

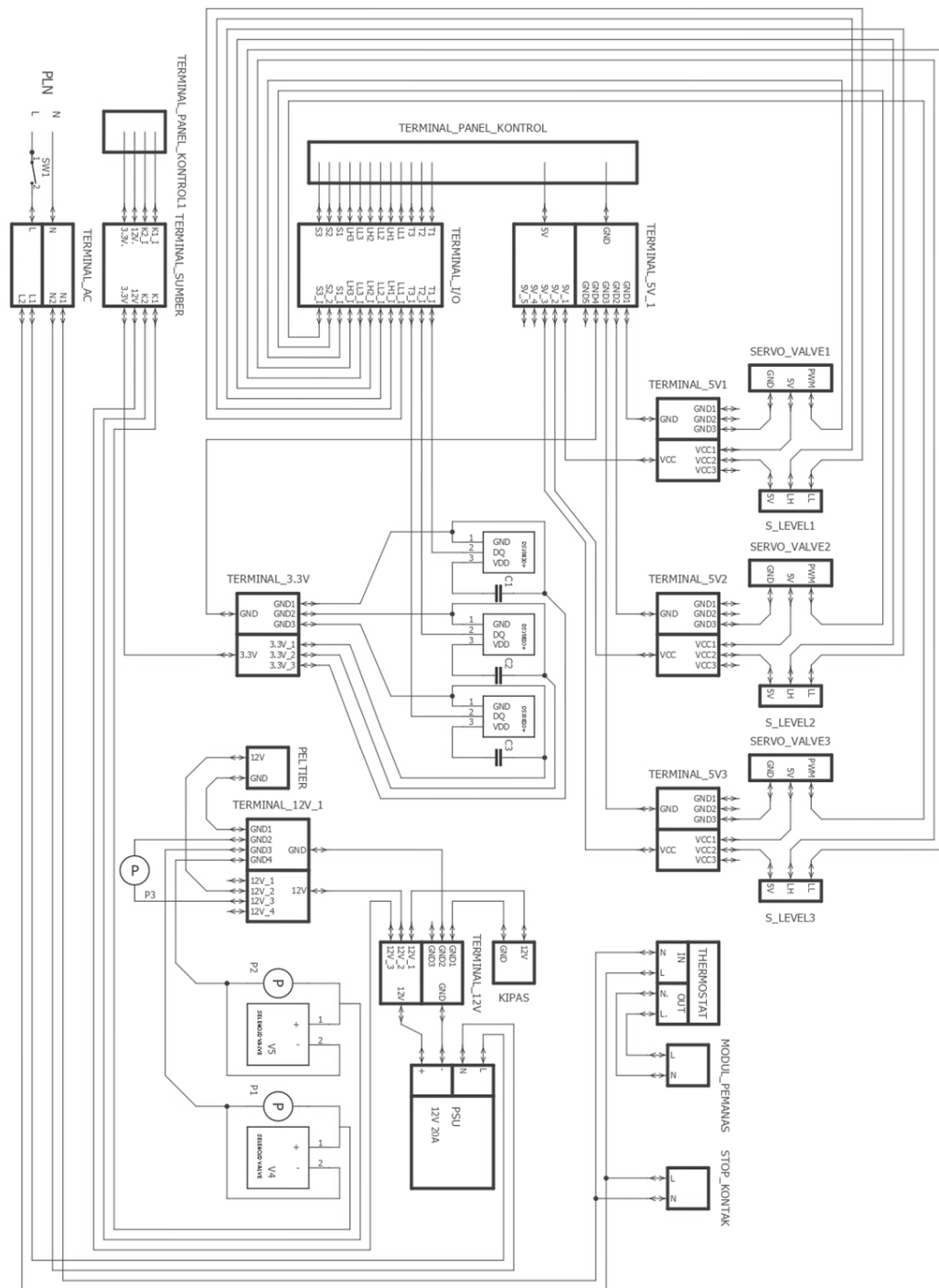
|                              |  |  |
|------------------------------|--|--|
|                              | <p><i>time</i> dengan pembaruan setiap 1 detik.</p> <p>3. Untuk keluar dari menu monitoring suhu air, tekan tombol “*”.</p>  | <p>2.</p>   |
| <p><b>Kontrol Manual</b></p> | <p>1. Pilih menu “<b>Kontrol Manual</b>” pada panel kontrol, lalu tekan tombol “#” untuk melanjutkan.</p> <p>2. Tekan tombol “1” untuk membuka atau katup servo pemanas.</p> <p>3. Tekan tombol “2” untuk membuka atau menutup katup servo pendingin.</p> <p>4. Tekan tombol “3” untuk membuka atau menutup katup servo utama.</p> <p>5. Tekan tombol “C” untuk mengubah status pompa pemanas.</p> <p>6. Tekan tombol “D” untuk mengubah status pompa pendingin.</p> | <p>1.</p>  <p>2.</p>  |

|  |   |
|--|---|
| <b>Prosedur<br/>Menonaktifkan<br/>Sistem</b> | <ol style="list-style-type: none"> <li>1. Tekan saklar catu daya pada plant untuk menonaktifkan sistem.</li> <li>2. Cabut kabel AC dari sumber listrik.</li> <li>3. Cabut adaptor 5V dari stop kontak pada plant.</li> <li>4. Buang sisa air pada tangki penampung menggunakan selang dengan cara disedot hingga kosong.</li> <li>5. Rapihan kembali <i>mini plant</i> seperti semula.</li> </ol> |
|--|---|



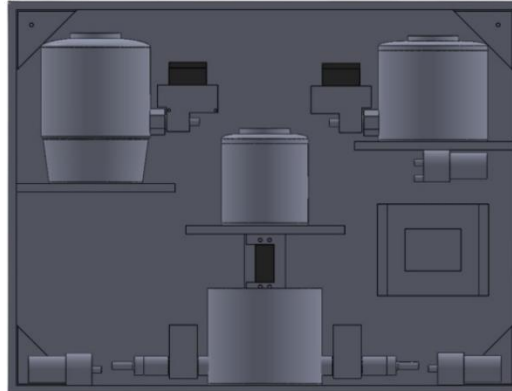


### Lampiran 3. Skema Rangkaian *Mini Plant*



Lampiran 4. Desain 3D *Mini Plant* Pengkondisi Air

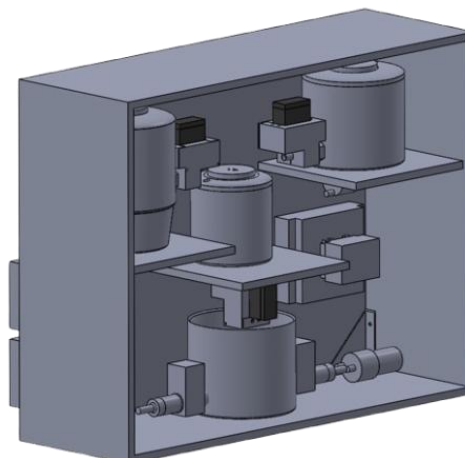
- Tampak Depan



- Tampak Belakang



- Tampak Samping

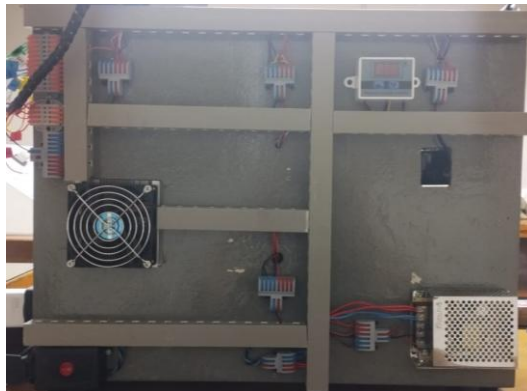


## Lampiran 5. Realisasi *Mini Plant* Pengkondisi Air

- Tampak Depan



- Tampak Belakang



- Tampak Samping



## Lampiran 6. Program Raspberry Pi Pico

### - adc\_levels.py

```
from machine import I2C, Pin
from ads1x15 import ADS1115

ads_i2c_bus = I2C(1, scl=Pin(19), sda=Pin(18), freq=400000)
adc_level1 = ADS1115(ads_i2c_bus, address=0x48, gain=1)
adc_level2 = ADS1115(ads_i2c_bus, address=0x49, gain=1)

def read_adc_level1():
    raw_LH_Heater = adc_level1.read(channel1=0, channel2=None)
    raw_LL_Heater = adc_level1.read(channel1=1, channel2=None)
    raw_LH_Cooler = adc_level1.read(channel1=2, channel2=None)
    raw_LL_Cooler = adc_level1.read(channel1=3, channel2=None)
    return raw_LH_Heater, raw_LL_Heater, raw_LH_Cooler, raw_LL_Cooler

def read_adc_level2():
    raw_LH_Utama = adc_level2.read(channel1=0, channel2=None)
    raw_LL_Utama = adc_level2.read(channel1=1, channel2=None)
    return raw_LL_Utama, raw_LH_Utama

def raw_to_voltage_level1(raw):
    return adc_level1.raw_to_v(raw)

def raw_to_voltage_level2(raw):
    return adc_level2.raw_to_v(raw)
```

### - ads1x15.py

```
# The MIT License (MIT)
#
# Copyright (c) 2016 Radomir Dopieralski (@deshipu),
#       2017 Robert Hammelrath (@robert-hh)
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
#
import utime as time

_REGISTER_MASK = const(0x03)
_REGISTER_CONVERT = const(0x00)
_REGISTER_CONFIG = const(0x01)
_REGISTER_LOWTHRESH = const(0x02)
_REGISTER_HITHRESH = const(0x03)

_OS_MASK = const(0x8000)
_OS_SINGLE = const(0x8000) # Write: Set to start a single-conversion
_OS_BUSY = const(0x0000) # Read: Bit=0 when conversion is in progress
_OS_NOTBUSY = const(0x8000) # Read: Bit=1 when no conversion is in progress

_MUX_MASK = const(0x7000)
_MUX_DIFF_0_1 = const(0x0000) # Differential P = AIN0, N = AIN1 (default)
_MUX_DIFF_0_3 = const(0x1000) # Differential P = AIN0, N = AIN3
```

```

_MUX_DIFF_1_3 = const(0x2000) # Differential P = AIN1, N = AIN3
_MUX_DIFF_2_3 = const(0x3000) # Differential P = AIN2, N = AIN3
_MUX_SINGLE_0 = const(0x4000) # Single-ended AIN0
_MUX_SINGLE_1 = const(0x5000) # Single-ended AIN1
_MUX_SINGLE_2 = const(0x6000) # Single-ended AIN2
_MUX_SINGLE_3 = const(0x7000) # Single-ended AIN3

_PGA_MASK = const(0x0E00)
_PGA_6_144V = const(0x0000) # +/-6.144V range = Gain 2/3
_PGA_4_096V = const(0x0200) # +/-4.096V range = Gain 1
_PGA_2_048V = const(0x0400) # +/-2.048V range = Gain 2 (default)
_PGA_1_024V = const(0x0600) # +/-1.024V range = Gain 4
_PGA_0_512V = const(0x0800) # +/-0.512V range = Gain 8
_PGA_0_256V = const(0x0A00) # +/-0.256V range = Gain 16

_MODE_MASK = const(0x0100)
_MODE_CONTIN = const(0x0000) # Continuous conversion mode
_MODE_SINGLE = const(0x0100) # Power-down single-shot mode (default)

_DR_MASK = const(0x00E0) # Values ADS1015/ADS1115
_DR_128SPS = const(0x0000) # 128 /8 samples per second
_DR_250SPS = const(0x0020) # 250 /16 samples per second
_DR_490SPS = const(0x0040) # 490 /32 samples per second
_DR_920SPS = const(0x0060) # 920 /64 samples per second
_DR_1600SPS = const(0x0080) # 1600/128 samples per second (default)
_DR_2400SPS = const(0x00A0) # 2400/250 samples per second
_DR_3300SPS = const(0x00C0) # 3300/475 samples per second
_DR_860SPS = const(0x00E0) # - /860 samples per Second

_CMODE_MASK = const(0x0010)
_CMODE_TRAD = const(0x0000) # Traditional comparator with hysteresis (default)
_CMODE_WINDOW = const(0x0010) # Window comparator

_CPOL_MASK = const(0x0008)
_CPOL_ACTVLOW = const(0x0000) # ALERT/RDY pin is low when active (default)
_CPOL_ACTVHI = const(0x0008) # ALERT/RDY pin is high when active

_CLAT_MASK = const(0x0004) # Determines if ALERT/RDY pin latches once asserted
_CLAT_NONLAT = const(0x0000) # Non-latching comparator (default)
_CLAT_LATCH = const(0x0004) # Latching comparator

_CQUE_MASK = const(0x0003)
_CQUE_1CONV = const(0x0000) # Assert ALERT/RDY after one conversions
_CQUE_2CONV = const(0x0001) # Assert ALERT/RDY after two conversions
_CQUE_4CONV = const(0x0002) # Assert ALERT/RDY after four conversions
# Disable the comparator and put ALERT/RDY in high state (default)
_CQUE_NONE = const(0x0003)

_GAINS = (
    _PGA_6_144V, # 2/3x
    _PGA_4_096V, # 1x
    _PGA_2_048V, # 2x
    _PGA_1_024V, # 4x
    _PGA_0_512V, # 8x
    _PGA_0_256V # 16x
)

_GAINS_V = (
    6.144, # 2/3x
    4.096, # 1x
    2.048, # 2x
    1.024, # 4x
    0.512, # 8x
    0.256 # 16x
)

_CHANNELS = {
    (0, None): _MUX_SINGLE_0,
    (1, None): _MUX_SINGLE_1,
    (2, None): _MUX_SINGLE_2,
    (3, None): _MUX_SINGLE_3,
    (0, 1): MUX_DIFF_0_1,

```

```

(0, 3): _MUX_DIFF_0_3,
(1, 3): _MUX_DIFF_1_3,
(2, 3): _MUX_DIFF_2_3,
}

_RATES = (
    _DR_128SPS, # 128/8 samples per second
    _DR_250SPS, # 250/16 samples per second
    _DR_490SPS, # 490/32 samples per second
    _DR_920SPS, # 920/64 samples per second
    _DR_1600SPS, # 1600/128 samples per second (default)
    _DR_2400SPS, # 2400/250 samples per second
    _DR_3300SPS, # 3300/475 samples per second
    _DR_860SPS # - /860 samples per Second
)

class ADS1115:
    def __init__(self, i2c, address=0x48, gain=1):
        self.i2c = i2c
        self.address = address
        self.gain = gain
        self.temp2 = bytearray(2)

    def _write_register(self, register, value):
        self.temp2[0] = value >> 8
        self.temp2[1] = value & 0xff
        self.i2c.writeto_mem(self.address, register, self.temp2)

    def _read_register(self, register):
        self.i2c.readfrom_mem_into(self.address, register, self.temp2)
        return (self.temp2[0] << 8) | self.temp2[1]

    def raw_to_v(self, raw):
        v_p_b = _GAINS_V[self.gain] / 32768
        return raw * v_p_b

    def set_conv(self, rate=4, channel1=0, channel2=None):
        """Set mode for read_rev"""
        self.mode = (_CQUE_NONE | _CLAT_NONLAT |
                     _CPOL_ACTVLOW | _CMODE_TRAD | _RATES[rate] |
                     _MODE_SINGLE | _OS_SINGLE | _GAINS[self.gain] |
                     _CHANNELS[(channel1, channel2)])

    def read(self, rate=4, channel1=0, channel2=None):
        """Read voltage between a channel and GND.
        Time depends on conversion rate."""
        self._write_register(_REGISTER_CONFIG, (_CQUE_NONE | _CLAT_NONLAT |
                                                _CPOL_ACTVLOW | _CMODE_TRAD | _RATES[rate] |
                                                _MODE_SINGLE | _OS_SINGLE | _GAINS[self.gain] |
                                                _CHANNELS[(channel1, channel2)]))
        while not self._read_register(_REGISTER_CONFIG) & _OS_NOTBUSY:
            time.sleep_ms(1)
        res = self._read_register(_REGISTER_CONVERT)
        return res if res < 32768 else res - 65536

    def read_rev(self):
        """Read voltage between a channel and GND. and then start
        the next conversion."""
        res = self._read_register(_REGISTER_CONVERT)
        self._write_register(_REGISTER_CONFIG, self.mode)
        return res if res < 32768 else res - 65536

    def alert_start(self, rate=4, channel1=0, channel2=None,
                    threshold_high=0x4000, threshold_low=0, latched=False):
        """Start continuous measurement, set ALERT pin on threshold."""
        self._write_register(_REGISTER_LOWTHRESH, threshold_low)
        self._write_register(_REGISTER_HITHRESH, threshold_high)
        self._write_register(_REGISTER_CONFIG, _CQUE_1CONV |
                             _CLAT_LATCH if latched else _CLAT_NONLAT |
                             _CPOL_ACTVLOW | _CMODE_TRAD | _RATES[rate] |
                             _MODE_CONTIN | _GAINS[self.gain])

```

```

        _CHANNELS[(channel1, channel2)])

def conversion_start(self, rate=4, channel1=0, channel2=None):
    """Start continuous measurement, trigger on ALERT/RDY pin."""
    self._write_register(_REGISTER_LOWTHRESH, 0)
    self._write_register(_REGISTER_HITHRESH, 0x8000)
    self._write_register(_REGISTER_CONFIG, _CQUE_1CONV | _CLAT_NONLAT |
        _CPOL_ACTVLOW | _CMODE_TRAD | _RATES[rate] |
        _MODE_CONTIN | _GAINS[self.gain] |
        _CHANNELS[(channel1, channel2)])

def alert_read(self):
    """Get the last reading from the continuous measurement."""
    res = self._read_register(_REGISTER_CONVERT)
    return res if res < 32768 else res - 65536

class ADS1113(ADS1115):
    def __init__(self, i2c, address=0x48):
        super().__init__(i2c, address, 1)

    def raw_to_v(self, raw):
        return super().raw_to_v(raw)

    def read(self, rate=4):
        return super().read(rate, 0, 1)

    def alert_start(self, rate=4, threshold_high=0x4000, threshold_low=0, latched=False):
        return super().alert_start(rate, 0, 1, threshold_high, threshold_low, latched)

    def alert_read(self):
        return super().alert_read()

class ADS1114(ADS1115):
    def __init__(self, i2c, address=0x48, gain=1):
        super().__init__(i2c, address, gain)

    def raw_to_v(self, raw):
        return super().raw_to_v(raw)

    def read(self, rate=4):
        return super().read(rate, 0, 1)

    def alert_start(self, rate=4, threshold_high=0x4000, threshold_low=0, latched=False):
        return super().alert_start(rate, 0, 1, threshold_high,
            threshold_low, latched)

    def alert_read(self):
        return super().alert_read()

class ADS1015(ADS1115):
    def __init__(self, i2c, address=0x48, gain=1):
        super().__init__(i2c, address, gain)

    def raw_to_v(self, raw):
        return super().raw_to_v(raw << 4)

    def read(self, rate=4, channel1=0, channel2=None):
        return super().read(rate, channel1, channel2) >> 4

    def alert_start(self, rate=4, channel1=0, channel2=None, threshold_high=0x400,
        threshold_low=0, latched=False):
        return super().alert_start(rate, channel1, channel2, threshold_high << 4,
            threshold_low << 4, latched)

    def alert_read(self):
        return super().alert_read() >> 4

```



- fuzzy\_control.py

```
# fuzzy_control.py

# Membership function boundaries (triangle MF)
error_mf_bounds = {
    'Very Cold': (-100, -10, -5),
    'Cold': (-10, -5, 0),
    'Normal': (-5, 0, 5),
    'Hot': (0, 5, 10),
    'Very Hot': (5, 10, 100)
}

delta_error_mf_bounds = {
    'Decreasing': (-100, -5, 0),
    'Stable': (-2, 0, 2),
    'Increasing': (0, 5, 100)
}

# Output crisp values for Rasio (%)
output_values = {
    'Sangat Tinggi': 0,
    'Tinggi': 10,
    'Agak Tinggi': 25,
    'Sedang Tinggi': 35,
    'Sedang': 50,
    'Sedang Rendah': 65,
    'Rendah': 75,
    'Sangat Rendah': 90,
    'Minimum': 100
}

# Fuzzy rules as per your specification
rules = {
    ('Very Cold', 'Decreasing'): 'Sangat Tinggi',
    ('Very Cold', 'Stable'): 'Sangat Tinggi',
    ('Very Cold', 'Increasing'): 'Tinggi',
    ('Cold', 'Decreasing'): 'Tinggi',
    ('Cold', 'Stable'): 'Agak Tinggi',
    ('Cold', 'Increasing'): 'Sedang Tinggi',
    ('Normal', 'Decreasing'): 'Sedang Tinggi',
    ('Normal', 'Stable'): 'Sedang',
    ('Normal', 'Increasing'): 'Sedang Rendah',
    ('Hot', 'Decreasing'): 'Sedang Rendah',
    ('Hot', 'Stable'): 'Rendah',
    ('Hot', 'Increasing'): 'Sangat Rendah',
    ('Very Hot', 'Decreasing'): 'Sangat Rendah',
    ('Very Hot', 'Stable'): 'Minimum',
    ('Very Hot', 'Increasing'): 'Minimum'
}

def triangle_mf(x, a, b, c):
    if x <= a or x >= c:
        return 0.0
    elif a < x < b:
        return (x - a) / (b - a)
    elif b <= x < c:
        return (c - x) / (c - b)
    else:
        return 0.0

def mf_error(x):
    memberships = {}
    for label, (a, b, c) in error_mf_bounds.items():
        memberships[label] = triangle_mf(x, a, b, c)
    return memberships

def mf_delta_error(x):
    memberships = {}
    for label, (a, b, c) in delta_error_mf_bounds.items():
        memberships[label] = triangle_mf(x, a, b, c)
    return memberships
```

```

def fuzzy_sugeno(error, delta_error):
    error_memberships = mf_error(error)
    delta_error_memberships = mf_delta_error(delta_error)

    numerator = 0.0
    denominator = 0.0

    for e_label, e_val in error_memberships.items():
        if e_val == 0:
            continue
        for d_label, d_val in delta_error_memberships.items():
            if d_val == 0:
                continue
            weight = min(e_val, d_val)
            output_label = rules.get((e_label, d_label))
            if output_label:
                z = output_values[output_label]
                numerator += weight * z
                denominator += weight

    if denominator == 0:
        # No rule fired, return 0 or some default safe value
        return 0
    return numerator / denominator

```

## - i2c\_lcd.py

```

import utime
import gc

from lcd_api import LcdApi
from machine import I2C

# PCF8574 pin definitions
MASK_RS = 0x01    # P0
MASK_RW = 0x02    # P1
MASK_E = 0x04     # P2

SHIFT_BACKLIGHT = 3 # P3
SHIFT_DATA = 4 # P4-P7

class I2cLcd(LcdApi):

    #Implements a HD44780 character LCD connected via PCF8574 on I2C

    def __init__(self, i2c, i2c_addr, num_lines, num_columns):
        self.i2c = i2c
        self.i2c_addr = i2c_addr
        self.i2c.writeto(self.i2c_addr, bytes([0]))
        utime.sleep_ms(20) # Allow LCD time to powerup
        # Send reset 3 times
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(5) # Need to delay at least 4.1 msec
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(1)
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        utime.sleep_ms(1)
        # Put LCD into 4-bit mode
        self.hal_write_init_nibble(self.LCD_FUNCTION)
        utime.sleep_ms(1)
        LcdApi.__init__(self, num_lines, num_columns)
        cmd = self.LCD_FUNCTION
        if num_lines > 1:
            cmd |= self.LCD_FUNCTION_2LINES
        self.hal_write_command(cmd)
        gc.collect()

    def hal_write_init_nibble(self, nibble):
        # Writes an initialization nibble to the LCD.
        # This particular function is only used during initialization.

```

```

byte = ((nibble >> 4) & 0x0f) << SHIFT_DATA
self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytes([byte]))
gc.collect()

def hal_backlight_on(self):
    # Allows the hal layer to turn the backlight on
    self.i2c.writeto(self.i2c_addr, bytes([1 << SHIFT_BACKLIGHT]))
    gc.collect()

def hal_backlight_off(self):
    # Allows the hal layer to turn the backlight off
    self.i2c.writeto(self.i2c_addr, bytes([0]))
    gc.collect()

def hal_write_command(self, cmd):
    # Write a command to the LCD. Data is latched on the falling edge of E.
    byte = ((self.backlight << SHIFT_BACKLIGHT) |
            ((cmd >> 4) & 0x0f) << SHIFT_DATA))
    self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
    self.i2c.writeto(self.i2c_addr, bytes([byte]))
    byte = ((self.backlight << SHIFT_BACKLIGHT) |
            (cmd & 0x0f) << SHIFT_DATA))
    self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
    self.i2c.writeto(self.i2c_addr, bytes([byte]))
    if cmd <= 3:
        # The home and clear commands require a worst case delay of 4.1 msec
        utime.sleep_ms(5)
    gc.collect()

def hal_write_data(self, data):
    # Write data to the LCD. Data is latched on the falling edge of E.
    byte = (MASK_RS |
            (self.backlight << SHIFT_BACKLIGHT) |
            (((data >> 4) & 0x0f) << SHIFT_DATA))
    self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
    self.i2c.writeto(self.i2c_addr, bytes([byte]))
    byte = (MASK_RS |
            (self.backlight << SHIFT_BACKLIGHT) |
            (data & 0x0f) << SHIFT_DATA))
    self.i2c.writeto(self.i2c_addr, bytes([byte | MASK_E]))
    self.i2c.writeto(self.i2c_addr, bytes([byte]))
    gc.collect()

```

## - interlocks.py

```

import adc_levels
import relays
from servo_control import servo_heater, servo_cooler, servo_utama, percentage_to_duty

# Initialize lock flags
relay1_locked = False
relay2_locked = False
servo_locked = False

# Initialize sensor states for interlock sensors
sensor_states = {
    "LH_Heater": False,
    "LL_Heater": False,
    "LH_Cooler": False,
    "LL_Cooler": False,
    "LH_Utama": False,
    "LL_Utama": False,
}

def check_sensor_state(name, voltage, threshold=3.5):
    """
    Track sensor HIGH/LOW state and print on state changes.
    Returns True if HIGH, False if LOW.
    """
    global sensor_states

```

```

if voltage > threshold and not sensor_states[name]:
    sensor_states[name] = True
    #print(f'{name} sensor is HIGH')
elif voltage <= threshold and sensor_states[name]:
    sensor_states[name] = False
    #print(f'{name} sensor is LOW')
return sensor_states[name]

def verify_sensor(read_voltage_func, *args, threshold=3.5, max_attempts=3):
    """
    Verify the sensor reading by checking it multiple times.
    Returns the voltage if readings agree, else None.
    """
    readings = []
    for _ in range(max_attempts):
        voltage = read_voltage_func(*args)
        readings.append(voltage)
    # Check if all readings are consistent (all above or all below threshold)
    all_high = all(v > threshold for v in readings)
    all_low = all(v <= threshold for v in readings)

    if all_high:
        return readings[-1] # return last reading if all high
    elif all_low:
        return readings[-1] # return last reading if all low
    else:
        # Inconsistent readings, return None to avoid state change
        return None

def apply_interlocks():
    global relay1_locked, relay2_locked, servo_locked

    high_voltage_threshold = 3.3

    # Read raw ADC values once for each sensor group
    raw_LH_Heater, raw_LL_Heater, raw_LH_Cooler, raw_LL_Cooler = adc_levels.read_adc_level1()
    raw_LL_Utama, raw_LH_Utama = adc_levels.read_adc_level2()

    # Verify and convert voltages with double check

    val_LH_Heater = verify_sensor(adc_levels.raw_to_voltage_level1, raw_LH_Heater,
threshold=high_voltage_threshold)
    val_LL_Heater = verify_sensor(adc_levels.raw_to_voltage_level1, raw_LL_Heater,
threshold=high_voltage_threshold)
    val_LH_Cooler = verify_sensor(adc_levels.raw_to_voltage_level1, raw_LH_Cooler,
threshold=high_voltage_threshold)
    val_LL_Cooler = verify_sensor(adc_levels.raw_to_voltage_level1, raw_LL_Cooler,
threshold=high_voltage_threshold)
    val_LH_Utama = verify_sensor(adc_levels.raw_to_voltage_level2, raw_LH_Utama,
threshold=high_voltage_threshold)
    val_LL_Utama = verify_sensor(adc_levels.raw_to_voltage_level2, raw_LL_Utama,
threshold=high_voltage_threshold)

    # If verification failed (None), keep previous states to avoid false triggering
    def safe_check(name, val):
        if val is None:
            # Return previous state without change
            return sensor_states[name]
        else:
            return check_sensor_state(name, val, high_voltage_threshold)

    # Check sensors and track states with printouts
    # Check sensors and track states with printouts
    LH_Heater_high = safe_check("LH_Heater", val_LH_Heater)
    LL_Heater_high = safe_check("LL_Heater", val_LL_Heater)
    LH_Cooler_high = safe_check("LH_Cooler", val_LH_Cooler)
    LL_Cooler_high = safe_check("LL_Cooler", val_LL_Cooler)
    LH_Utama_high = safe_check("LH_Utama", val_LH_Utama)
    LL_Utama_high = safe_check("LL_Utama", val_LL_Utama)

```

```

# Enforce rule: If LL is LOW, then LH must be LOW (regardless of voltage)
if not LL_Heater_high:
    if LH_Heater_high:
        #print("LL_Heater LOW forces LH_Heater LOW")
        LH_Heater_high = False

if not LL_Cooler_high:
    if LH_Cooler_high:
        #print("LL_Cooler LOW forces LH_Cooler LOW")
        LH_Cooler_high = False

if not LL_Utama_high:
    if LH_Utama_high:
        #print("LL_Utama LOW forces LH_Utama LOW")
        LH_Utama_high = False

# Enforce rule: If LH is HIGH, then LL must be HIGH (force LL HIGH if LH HIGH)
if LH_Heater_high and not LL_Heater_high:
    LL_Heater_high = True

if LH_Cooler_high and not LL_Cooler_high:
    LL_Cooler_high = True

if LH_Utama_high and not LL_Utama_high:
    LL_Utama_high = True

# Relay 1 interlock
if LH_Heater_high:
    relay1_locked = True
    relays.relay_1_off()
elif not LL_Heater_high:
    relay1_locked = False
    relays.relay_1_on()

# Relay 2 interlock
if LH_Cooler_high:
    relay2_locked = True
    relays.relay_2_off()
elif not LL_Cooler_high:
    relay2_locked = False
    relays.relay_2_on()

# Servo interlock
if LH_Utama_high:
    servo_locked = True
    servo_heater.duty_u16(percentage_to_duty(0, "heater")) # Servo Heater OFF
    servo_cooler.duty_u16(percentage_to_duty(0, "cooler")) # Servo Cooler OFF
    servo_utama.duty_u16(percentage_to_duty(100, "utama")) # Servo Utama ON
elif not LL_Utama_high:
    servo_locked = False
    servo_utama.duty_u16(percentage_to_duty(0, "utama")) # Servo Utama OFF

return relay1_locked, relay2_locked, servo_locked

```

## - keypad.py

```

from machine import Pin
import time

rows = [Pin(2, Pin.OUT), Pin(3, Pin.OUT), Pin(4, Pin.OUT), Pin(5, Pin.OUT)]
cols = [Pin(6, Pin.IN, Pin.PULL_DOWN), Pin(7, Pin.IN, Pin.PULL_DOWN),
        Pin(8, Pin.IN, Pin.PULL_DOWN), Pin(9, Pin.IN, Pin.PULL_DOWN)]

keymap = [
    ['D', 'C', 'B', 'A'],
    ['#', '9', '6', '3'],
    ['0', '8', '5', '2'],
    ['*', '7', '4', '1']
]

def scan_keypad():

```

```

for r in range(4):
    rows[r].on()
    for c in range(4):
        if cols[c].value() == 1:
            while cols[c].value() == 1:
                time.sleep_ms(10)
            rows[r].off()
            return keymap[r][c]
    rows[r].off()
return None

```

## - lcd\_api.py

```

import time

class LcdApi:

    # Implements the API for talking with HD44780 compatible character LCDs.
    # This class only knows what commands to send to the LCD, and not how to get
    # them to the LCD.
    #
    # It is expected that a derived class will implement the hal_xxx functions.
    #
    # The following constant names were lifted from the avrlib lcd.h header file,
    # with bit numbers changed to bit masks.

    # HD44780 LCD controller command set
    LCD_CLR          = 0x01 # DB0: clear display
    LCD_HOME         = 0x02 # DB1: return to home position

    LCD_ENTRY_MODE   = 0x04 # DB2: set entry mode
    LCD_ENTRY_INC    = 0x02 # DB1: increment
    LCD_ENTRY_SHIFT  = 0x01 # DB0: shift

    LCD_ON_CTRL      = 0x08 # DB3: turn lcd/cursor on
    LCD_ON_DISPLAY   = 0x04 # DB2: turn display on
    LCD_ON_CURSOR    = 0x02 # DB1: turn cursor on
    LCD_ON_BLINK     = 0x01 # DB0: blinking cursor

    LCD_MOVE         = 0x10 # DB4: move cursor/display
    LCD_MOVE_DISP    = 0x08 # DB3: move display (0-> move cursor)
    LCD_MOVE_RIGHT   = 0x04 # DB2: move right (0-> left)

    LCD_FUNCTION     = 0x20 # DB5: function set
    LCD_FUNCTION_8BIT = 0x10 # DB4: set 8BIT mode (0->4BIT mode)
    LCD_FUNCTION_2LINES = 0x08 # DB3: two lines (0->one line)
    LCD_FUNCTION_10DOTS = 0x04 # DB2: 5x10 font (0->5x7 font)
    LCD_FUNCTION_RESET = 0x30 # See "Initializing by Instruction" section

    LCD_CGRAM        = 0x40 # DB6: set CG RAM address
    LCD_DDRAM        = 0x80 # DB7: set DD RAM address

    LCD_RS_CMD       = 0
    LCD_RS_DATA      = 1

    LCD_RW_WRITE     = 0
    LCD_RW_READ      = 1

    def __init__(self, num_lines, num_columns):
        self.num_lines = num_lines
        if self.num_lines > 4:
            self.num_lines = 4
        self.num_columns = num_columns
        if self.num_columns > 40:
            self.num_columns = 40
        self.cursor_x = 0
        self.cursor_y = 0
        self.implied_newline = False
        self.backlight = True
        self.display_off()
        self.backlight_on()

```

```

self.clear()
self.hal_write_command(self.LCD_ENTRY_MODE | self.LCD_ENTRY_INC)
self.hide_cursor()
self.display_on()

def clear(self):
    # Clears the LCD display and moves the cursor to the top left corner
    self.hal_write_command(self.LCD_CLR)
    self.hal_write_command(self.LCD_HOME)
    self.cursor_x = 0
    self.cursor_y = 0

def show_cursor(self):
    # Causes the cursor to be made visible
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR)

def hide_cursor(self):
    # Causes the cursor to be hidden
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY)

def blink_cursor_on(self):
    # Turns on the cursor, and makes it blink
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR | self.LCD_ON_BLINK)

def blink_cursor_off(self):
    # Turns on the cursor, and makes it no blink (i.e. be solid)
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR)

def display_on(self):
    # Turns on (i.e. unblanks) the LCD
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY)

def display_off(self):
    # Turns off (i.e. blanks) the LCD
    self.hal_write_command(self.LCD_ON_CTRL)

def backlight_on(self):
    # Turns the backlight on.

    # This isn't really an LCD command, but some modules have backlight
    # controls, so this allows the hal to pass through the command.
    self.backlight = True
    self.hal_backlight_on()

def backlight_off(self):
    # Turns the backlight off.

    # This isn't really an LCD command, but some modules have backlight
    # controls, so this allows the hal to pass through the command.
    self.backlight = False
    self.hal_backlight_off()

def move_to(self, cursor_x, cursor_y):
    # Moves the cursor position to the indicated position. The cursor
    # position is zero based (i.e. cursor_x == 0 indicates first column).
    self.cursor_x = cursor_x
    self.cursor_y = cursor_y
    addr = cursor_x & 0x3f
    if cursor_y & 1:
        addr += 0x40 # Lines 1 & 3 add 0x40
    if cursor_y & 2: # Lines 2 & 3 add number of columns
        addr += self.num_columns
    self.hal_write_command(self.LCD_DDRAM | addr)

def putchar(self, char):
    # Writes the indicated character to the LCD at the current cursor
    # position, and advances the cursor by one position.
    if char == '\n':
        if self.implied_newline:

```

```

        # self.implied_newline means we advanced due to a wraparound,
        # so if we get a newline right after that we ignore it.
        pass
    else:
        self.cursor_x = self.num_columns
    else:
        self.hal_write_data(ord(char))
        self.cursor_x += 1
    if self.cursor_x >= self.num_columns:
        self.cursor_x = 0
        self.cursor_y += 1
        self.implied_newline = (char != '\n')
    if self.cursor_y >= self.num_lines:
        self.cursor_y = 0
    self.move_to(self.cursor_x, self.cursor_y)

def putstr(self, string):
    # Write the indicated string to the LCD at the current cursor
    # position and advances the cursor position appropriately.
    for char in string:
        self.putchar(char)

def custom_char(self, location, charmap):
    # Write a character to one of the 8 CGRAM locations, available
    # as chr(0) through chr(7).
    location &= 0x7
    self.hal_write_command(self.LCD_CGRAM | (location << 3))
    self.hal_sleep_us(40)
    for i in range(8):
        self.hal_write_data(charmap[i])
        self.hal_sleep_us(40)
    self.move_to(self.cursor_x, self.cursor_y)

def hal_backlight_on(self):
    # Allows the hal layer to turn the backlight on.
    # If desired, a derived HAL class will implement this function.
    pass

def hal_backlight_off(self):
    # Allows the hal layer to turn the backlight off.
    # If desired, a derived HAL class will implement this function.
    pass

def hal_write_command(self, cmd):
    # Write a command to the LCD.
    # It is expected that a derived HAL class will implement this function.
    raise NotImplementedError

def hal_write_data(self, data):
    # Write data to the LCD.
    # It is expected that a derived HAL class will implement this function.
    raise NotImplementedError

def hal_sleep_us(self, usecs):
    # Sleep for some time (given in microseconds)
    time.sleep_us(usecs)

```

## - lcd\_display.py

```

from machine import I2C, Pin
from lcd_api import LcdApi
from i2c_lcd import I2cLcd

i2c = I2C(0, scl=Pin(1), sda=Pin(0), freq=400000)
lcd_addr = i2c.scan()[0]
lcd = I2cLcd(i2c, lcd_addr, 4, 20)

def clear():
    lcd.clear()

def putstr(text):

```



```

        lcd.putstr(text)

def move_to(x, y):
    lcd.move_to(x, y)

```

## - main.py

```

import time
from servo_control import initialize_servos, percentage_to_duty, servo_heater, servo_cooler, servo_utama, servo_states
from keypad import scan_keypad
import lcd_display
import sensors
import interlocks
import relays
import adc_levels

from fuzzy_control import fuzzy_sugeno

# Variables for fuzzy control error tracking
last_error = 0
last_time = time.time()

# Menu and mode variables
menu_items = [
    "Setpoint Utama",
    "Sensor Suhu",
    "Sensor Level",
    "Kontrol Manual"
]

current_selection = 0
mode = "menu"
setpoint = ""

def show_menu(selected):
    lcd_display.clear()
    for i, item in enumerate(menu_items):
        lcd_display.move_to(0, i)
        if i == selected:
            lcd_display.putstr(">")
        else:
            lcd_display.putstr(" ")
        lcd_display.putstr(item[:19])

def handle_menu():
    global current_selection, mode, setpoint
    show_menu(current_selection)

    while True:
        key = scan_keypad()
        if key:
            if mode == "menu":
                if key == 'A':
                    current_selection = (current_selection - 1) % len(menu_items)
                    show_menu(current_selection)
                elif key == 'B':
                    current_selection = (current_selection + 1) % len(menu_items)
                    show_menu(current_selection)
                elif key == '#':
                    selected_item = menu_items[current_selection]
                    if selected_item == "Setpoint Utama":
                        mode = "input_setpoint"
                    elif selected_item == "Sensor Suhu":
                        mode = "baca_suhu"
                    elif selected_item == "Sensor Level":
                        mode = "baca_level"
                    elif selected_item == "Kontrol Manual":
                        mode = "kontrol_manual"
                    break
            time.sleep(0.1)
    lcd_display.clear()

```

```

def run_fuzzy_control():
    global mode, setpoint
    inputting_setpoint = False
    new_setpoint = None
    last_error = 0
    last_time = time.time()
    last_valid_temp = None # Store last valid temperature here
    start_time = time.time() # Start time recorded here!

    sensor_detected = False # Flag to track if sensor has been detected
    main_rom_id = None # Store main ROM ID when detected
    target_temp = float(setpoint)

    while True:
        relay1_locked, relay2_locked, servo_locked = interlocks.apply_interlocks()
        if servo_locked:
            # Interlock active: pause PID control
            servo_heater.duty_u16(percentage_to_duty(0, "heater"))
            servo_cooler.duty_u16(percentage_to_duty(0, "cooler"))

            # Just wait and keep checking, don't proceed further
            time.sleep(0.5)
            continue

        temp_main = None

        if not sensor_detected:
            # Retry loop for detecting main_roms only if not detected yet
            retry_count = 0
            while retry_count < 10:
                heater_roms, cooler_roms, main_roms = sensors.scan_sensors()
                if main_roms:
                    main_rom_id = main_roms[0]
                    sensor_detected = True
                    temp_main = sensors.read_temp(sensors.ds_main, main_rom_id)
                    break # Exit retry loop
                else:
                    retry_count += 1
                    time.sleep(0.2) # Small delay between retries

            if not sensor_detected:
                lcd_display.clear()
                lcd_display.move_to(0, 0)
                lcd_display.putstr("Sensor Error")
                mode = "menu"
                time.sleep(2)
                break # Exit run_fuzzy_control loop

        else:
            # Sensor already detected, just try to read temp directly
            try:
                temp_main = sensors.read_temp(sensors.ds_main, main_rom_id)
            except Exception:
                # Reading failed, keep temp_main as None to use last valid temp
                temp_main = None

        try:
            if temp_main is not None:
                last_valid_temp = temp_main # Update last valid temp
            else:
                temp_main = last_valid_temp # Use last valid temp if no new reading
                time.sleep(0.1)

            if temp_main is not None and setpoint:
                target_temp = float(setpoint)
                current_temp = temp_main

                error = target_temp - current_temp
                now = time.time()
                dt = time.time_diff(now, last_time) / 1000
                delta_error = 0
                if dt > 0:

```

```

        delta_error = (error - last_error) / dt

        last_error = error
        last_time = now

        value = fuzzy_sugeno(error, delta_error)
        percentage_value = value / 100

        elapsed_time = time.time() - start_time

        # PRINT TO TERMINAL:
        print(f"Time: {elapsed_time / 1000:.2f}s, Setpoint: {float(setpoint):.2f} C, Temp: {current_temp:.2f} C, Output
: {percentage_value:.2f}%")

        # Control heater servo with fuzzy output
        duty_heater = percentage_to_duty(percentage_value * 100, "heater")
        duty_cooler = percentage_to_duty((1 - percentage_value) * 100, "cooler")

        servo_heater.duty_u16(duty_heater)
        servo_cooler.duty_u16(duty_cooler)

        # If we're inputting new setpoint, show that UI and capture input

        key = scan_keypad()
        if key == '*':
            mode = "menu"
            # Turn off servo when exiting fuzzy control
            servo_heater.duty_u16(percentage_to_duty(0, "heater"))
            servo_cooler.duty_u16(percentage_to_duty(0, "cooler"))
            break
        elif key == '#' and not inputting_setpoint:
            inputting_setpoint = True
            new_setpoint = None
            lcd_display.clear()

        if inputting_setpoint:
            result = input_new_setpoint(new_setpoint or setpoint)
            if result == "cancel":
                inputting_setpoint = False
                lcd_display.clear()
            elif result is not None: # New setpoint confirmed
                setpoint = result
                inputting_setpoint = False
                lcd_display.clear()
            else:
                # Still entering, just continue showing input
                pass

        if not inputting_setpoint:
            lcd_display.clear()
            lcd_display.putstr("Temp: {:.2f} C".format(current_temp))
            lcd_display.move_to(0, 1)
            lcd_display.putstr("Setpt: {:.2f} C".format(target_temp))
            lcd_display.move_to(0, 2)
            lcd_display.putstr("Output: {:.2f}%".format(percentage_value))
            lcd_display.move_to(0, 3)
            lcd_display.putstr("*:EXIT #: Set New SP")
        else:
            # No valid temp or setpoint - turn off heater
            servo_heater.duty_u16(percentage_to_duty(0, "heater"))
            servo_cooler.duty_u16(percentage_to_duty(0, "cooler"))

    except Exception as e:
        print("Exception occurred:", e)
        import sys
        import traceback
        traceback.print_exc()
        lcd_display.clear()
        lcd_display.putstr("Error")
        time.sleep(2)
        break

```

```

time.sleep(0.1)

def input_new_setpoint(current_setpoint):

    lcd_display.move_to(0, 0)
    lcd_display.putstr("Setpoint (20-40 C):")
    lcd_display.move_to(0, 1)

    entered = ""
    # Use a loop that runs for a short time, collecting input, but returns quickly.
    start_time = time.time()

    while True:
        key = scan_keypad()
        if key:
            if key.isdigit():
                if len(entered) < 2: # limit input length to 2 digits
                    entered += key
                    lcd_display.clear()
                    lcd_display.putstr("Setpoint (20-40 C):")
                    lcd_display.move_to(0, 1)
                    lcd_display.putstr(entered + " C")
                    time.sleep(0.2) # debounce
            elif key == '#': # Confirm
                try:
                    val = int(entered)
                    if 20 <= val <= 40:
                        return str(val) # valid new setpoint
                    else:
                        lcd_display.clear()
                        lcd_display.putstr("Out of range")
                        time.sleep(1)
                        lcd_display.clear()
                        lcd_display.putstr("Setpoint (20-40 C):")
                        lcd_display.move_to(0, 1)
                        entered = ""
                except:
                    lcd_display.clear()
                    lcd_display.putstr("Invalid input")
                    time.sleep(1)
                    lcd_display.clear()
                    lcd_display.putstr("Setpoint (20-40 C):")
                    lcd_display.move_to(0, 1)
                    entered = ""
            elif key == '*': # Cancel
                return "cancel"
        # Return None to indicate still entering or no key
        # But add a timeout so this doesn't hang too long
        if time.time() - start_time > 5.0: # 5 sec timeout
            return None
        time.sleep(0.1)

def input_setpoint_mode():
    global mode, setpoint
    lcd_display.clear()
    lcd_display.putstr("Setpoint (20-40 C):") # changed here
    setpoint = ""
    lcd_display.move_to(0, 1)

    while True:
        interlocks.apply_interlocks()
        key = scan_keypad()
        if key:
            if key.isdigit():
                setpoint += key
                lcd_display.clear()
                lcd_display.putstr("Setpoint (20-40 C):") # changed here
                lcd_display.move_to(0, 1)
                lcd_display.putstr(setpoint + " C")
            elif key == '#': # Confirm
                try:
                    val = int(setpoint)

```

```

if 20 <= val <= 40: # changed range here
    lcd_display.clear()
    lcd_display.putstr("SP saved: {} C".format(val))
    setpoint = str(val)
    time.sleep(2)

    # Option after saving setpoint
    lcd_display.clear()
    lcd_display.putstr("#: Mulai Fuzzy")
    lcd_display.move_to(0, 1)
    lcd_display.putstr("*: Kembali")

    while True:
        key2 = scan_keypad()
        if key2 == "#":
            run_fuzzy_control()
            break
        elif key2 == "*":
            mode = "menu"
            break
        time.sleep(0.1)
    break # exit input_setpoint_mode()
else:
    lcd_display.clear()
    lcd_display.putstr("Out of range")
    time.sleep(2)
    lcd_display.clear()
    lcd_display.putstr("Setpoint (20-40 C):") # changed here
    setpoint = ""
except:
    lcd_display.clear()
    lcd_display.putstr("Invalid input")
    time.sleep(2)
    lcd_display.clear()
    lcd_display.putstr("Setpoint (20-40 C):") # changed here
    setpoint = ""
elif key == "*": # Cancel
    mode = "menu"
    break
time.sleep(0.1)

def baca_suhu_mode():
    global mode
    lcd_display.clear()
    lcd_display.putstr("Baca Sensor Suhu")
    time.sleep(1)

    while True:
        interlocks.apply_interlocks()
        try:
            heater_roms, cooler_roms, main_roms = sensors.scan_sensors()

            temp_heater = None
            temp_cooler = None
            temp_main = None

            if heater_roms:
                try:
                    temp_heater = sensors.read_temp(sensors.ds_heater, heater_roms[0])
                except Exception:
                    temp_heater = None

            if cooler_roms:
                try:
                    temp_cooler = sensors.read_temp(sensors.ds_cooler, cooler_roms[0])
                except Exception:
                    temp_cooler = None

            if main_roms:
                try:
                    temp_main = sensors.read_temp(sensors.ds_main, main_roms[0])

```

```

except Exception:
    temp_main = None

    lcd_display.clear()
    lcd_display.putstr("Heater: {} C".format(f"{temp_heater:.2f}" if temp_heater is not None else "N/A"))
    lcd_display.move_to(0, 1)
    lcd_display.putstr("Cooler: {} C".format(f"{temp_cooler:.2f}" if temp_cooler is not None else "N/A"))
    lcd_display.move_to(0, 2)
    lcd_display.putstr("Utama: {} C".format(f"{temp_main:.2f}" if temp_main is not None else "N/A"))
    lcd_display.move_to(0, 3)
    lcd_display.putstr("Press '*' to exit")

    key = scan_keypad()
    if key == '*':
        mode = "menu"
        break

    time.sleep(0.5)

except Exception as e:
    lcd_display.clear()
    lcd_display.putstr("Sensor error")
    time.sleep(2)
    break

def print_level_readings():
    raw_LH_Heater, raw_LL_Heater, raw_LH_Cooler, raw_LL_Cooler = adc_levels.read_adc_level1()
    raw_LL_Utama, raw_LH_Utama = adc_levels.read_adc_level2()

    val_LL_Heater = adc_levels.raw_to_voltage_level1(raw_LL_Heater)
    val_LH_Heater = adc_levels.raw_to_voltage_level1(raw_LH_Heater)
    val_LL_Cooler = adc_levels.raw_to_voltage_level1(raw_LL_Cooler)
    val_LH_Cooler = adc_levels.raw_to_voltage_level1(raw_LH_Cooler)
    val_LL_Utama = adc_levels.raw_to_voltage_level2(raw_LL_Utama)
    val_LH_Utama = adc_levels.raw_to_voltage_level2(raw_LH_Utama)

def baca_level_mode():
    global mode
    lcd_display.clear()
    lcd_display.putstr("Baca Sensor Level")
    time.sleep(1)

    while True:
        # Read all raw ADC values at once
        interlocks.apply_interlocks()
        raw_LH_Heater, raw_LL_Heater, raw_LH_Cooler, raw_LL_Cooler = adc_levels.read_adc_level1()
        raw_LL_Utama, raw_LH_Utama = adc_levels.read_adc_level2()

        # Convert raw to voltage
        val_LL_Heater = adc_levels.raw_to_voltage_level1(raw_LL_Heater)
        val_LH_Heater = adc_levels.raw_to_voltage_level1(raw_LH_Heater)
        val_LL_Cooler = adc_levels.raw_to_voltage_level1(raw_LL_Cooler)
        val_LH_Cooler = adc_levels.raw_to_voltage_level1(raw_LH_Cooler)
        val_LL_Utama = adc_levels.raw_to_voltage_level2(raw_LL_Utama)
        val_LH_Utama = adc_levels.raw_to_voltage_level2(raw_LH_Utama)

        # Update LCD with readings
        lcd_display.move_to(0, 0)
        lcd_display.putstr("Htr L: {:.2f} H: {:.2f} ".format(val_LL_Heater, val_LH_Heater))
        lcd_display.move_to(0, 1)
        lcd_display.putstr("Clr L: {:.2f} H: {:.2f} ".format(val_LL_Cooler, val_LH_Cooler))
        lcd_display.move_to(0, 2)
        lcd_display.putstr("Utma L: {:.2f} H: {:.2f} ".format(val_LL_Utama, val_LH_Utama))
        lcd_display.move_to(0, 3)
        lcd_display.putstr("(*) Kembali ")

        # Check for exit key
        key = scan_keypad()
        if key == '*':
            mode = "menu"
            break

```

```

time.sleep(0.5)

def kontrol_manual_mode():
    global mode, relay_1_status, relay_2_status

    relay_1_status = False
    relay_2_status = False
    lcd_display.clear()
    lcd_display.putstr("B:Servo; C:S1; D:S2")
    lcd_display.move_to(0, 3)
    lcd_display.putstr("(*) Kembali")
    max_retries = 3
    retry_delay = 0.5 # seconds

    while True:
        # Try to read interlocks with retries
        for attempt in range(max_retries):
            try:
                relay1_locked, relay2_locked, servo_locked = interlocks.apply_interlocks()
                break # success, exit retry loop
            except Exception as e:
                time.sleep(retry_delay)
        else:
            time.sleep(2)
            mode = "menu"
            break

    key = scan_keypad()
    if key:
        if key == "*":
            mode = "menu"
            break
        elif key == "1":
            if servo_locked:
                lcd_display.move_to(0, 1)
                lcd_display.putstr("Servo Heater Locked!")
            else:
                servo_states["heater"] = not servo_states["heater"]
                pos = 100 if servo_states["heater"] else 0
                servo_heater.duty_u16(percentage_to_duty(pos, "heater"))

                lcd_display.move_to(0, 1)
                if servo_states["utama"]:
                    lcd_display.putstr("Servo Heater Opened ")
                else:
                    lcd_display.putstr("Servo Heater Closed ")

        elif key == "2":
            if servo_locked:
                lcd_display.move_to(0, 1)
                lcd_display.putstr("Servo Cooler Locked!")
            else:
                servo_states["cooler"] = not servo_states["cooler"]
                pos = 100 if servo_states["cooler"] else 0
                servo_cooler.duty_u16(percentage_to_duty(pos, "cooler"))

                lcd_display.move_to(0, 1)
                if servo_states["utama"]:
                    lcd_display.putstr("Servo Cooler Opened ")
                else:
                    lcd_display.putstr("Servo Cooler Closed ")

        elif key == "3":
            lcd_display.move_to(0, 1)
            lcd_display.putstr(" ")
            servo_states["utama"] = not servo_states["utama"]
            pos = 100 if servo_states["utama"] else 0
            servo_utama.duty_u16(percentage_to_duty(pos, "utama"))

            lcd_display.move_to(0, 1)
            if servo_states["utama"]:

```

```

        lcd_display.putstr("Servo Utama Opened ")
    else:
        lcd_display.putstr("Servo Utama Closed ")

elif key == "C":
    if relay1_locked:
        lcd_display.move_to(0, 1)
        lcd_display.putstr("Relay Heater Locked!")
    else:
        if relay_1_status:
            relays.relay_1_off()
            relay_1_status = False
            lcd_display.move_to(0, 1)
            lcd_display.putstr("Relay Heater OFF ")
        else:
            relays.relay_1_on()
            relay_1_status = True
            lcd_display.move_to(0, 1)
            lcd_display.putstr("Relay Heater ON ")

elif key == "D":
    if relay2_locked:
        lcd_display.move_to(0, 1)
        lcd_display.putstr("Relay Cooler Locked!")
    else:
        if relay_2_status:
            relays.relay_2_off()
            relay_2_status = False
            lcd_display.move_to(0, 1)
            lcd_display.putstr("Relay Cooler OFF ")
        else:
            relays.relay_2_on()
            relay_2_status = True
            lcd_display.move_to(0, 1)
            lcd_display.putstr("Relay Cooler ON ")

time.sleep(0.1)

initialize_servos()

while True:
    interlocks.apply_interlocks()
    if mode == "menu":
        handle_menu()
    elif mode == "input_setpoint":
        input_setpoint_mode()
    elif mode == "baca_suhu":
        baca_suhu_mode()
    elif mode == "baca_level":
        baca_level_mode()
    elif mode == "kontrol_manual":
        kontrol_manual_mode()
    else:
        mode = "menu"

time.sleep(0.1)

```

## - relays.py

```

from machine import Pin

relay_1 = Pin(11, Pin.OUT)
relay_2 = Pin(10, Pin.OUT)

# True = OFF, False = ON (inverted logic)
relay_1_status = True
relay_2_status = True

relay_1.high()

```



```

relay_2.high()

def relay_1_on():
    global relay_1_status
    relay_1.low()
    relay_1_status = False

def relay_1_off():
    global relay_1_status
    relay_1.high()
    relay_1_status = True

def relay_2_on():
    global relay_2_status
    relay_2.low()
    relay_2_status = False

def relay_2_off():
    global relay_2_status
    relay_2.high()
    relay_2_status = True

```

#### - sensors.py

```

import onewire, ds18x20
from machine import Pin
import time

ds_heater = ds18x20.DS18X20(onewire.OneWire(Pin(26)))
ds_cooler = ds18x20.DS18X20(onewire.OneWire(Pin(27)))
ds_main = ds18x20.DS18X20(onewire.OneWire(Pin(28)))

def scan_sensors():
    heater_roms = ds_heater.scan()
    cooler_roms = ds_cooler.scan()
    main_roms = ds_main.scan()
    return heater_roms, cooler_roms, main_roms

def read_temp(ds_sensor, rom):
    try:
        ds_sensor.convert_temp()
        time.sleep(0.75)
        temp = ds_sensor.read_temp(rom)
    except Exception as e:
        temp = None
    return temp

```

#### - servo\_control.py

```

from machine import Pin, PWM
from time import sleep

# Duty cycle ranges
max_duty_heater = 5000
min_duty_heater = 2900

max_duty_cooler = 5500
min_duty_cooler = 2900

max_duty_utama = 4600
min_duty_utama = 1400

# Setup servo pins and PWM
pin_servo_heater = Pin(12, Pin.OUT)
pin_servo_cooler = Pin(13, Pin.OUT)
pin_servo_utama = Pin(14, Pin.OUT)

servo_heater = PWM(pin_servo_heater)
servo_cooler = PWM(pin_servo_cooler)

```

```

servo_utama = PWM(pin_servo_utama)

servo_heater.freq(50)
servo_cooler.freq(50)
servo_utama.freq(50)

servo_states = {
    "heater": False,
    "cooler": False,
    "utama": False
}

def percentage_to_duty(percentage, device):
    # Clamp percentage between 0 and 100
    if percentage < 0:
        percentage = 0
    elif percentage > 100:
        percentage = 100

    if device == "heater":
        duty = max_duty_heater - (percentage / 100) * (max_duty_heater - min_duty_heater)
    elif device == "cooler":
        duty = min_duty_cooler + (percentage / 100) * (max_duty_cooler - min_duty_cooler)
    elif device == "utama":
        duty = max_duty_utama - (percentage / 100) * (max_duty_utama - min_duty_utama)
    else:
        duty = 0
    return int(duty)

def initialize_servos():
    servo_heater.duty_u16(percentage_to_duty(0, "heater"))
    servo_cooler.duty_u16(percentage_to_duty(0, "cooler"))
    servo_utama.duty_u16(percentage_to_duty(0, "utama"))

```