# BAB IV
# PEMBUATAN ALAT

## 4.1 Pembuatan Prototype

Pada pembuatan Tugas Akhir ini memerlukan metode perancangan yang terstuktur secara baik untuk menghasilkan sistem yang berfungsi sesuai dengan tujuan serta dapat bekerja secara optimal. Bab ini membahas mengenai pembuatan alat secara keseluruhan, mulai dari proses perencanaan, pemilihan komponen, perancangan perangkat keras dan perangkat lunak, hingga konfigurasi dan integrasi pada setiap komponen untuk menjadi suatu sistem. Setiap tahapan yang dijalankan memiliki hubungan erat dalam pengembangan sistem pendingin panel surya berbasis metode fuzzy dengan ESP32 ini. Secara umum, proses pembuatan alat dibagi menjadi dua tahapan utama, yaitu pembuatan perangkat keras (*Hardware*) dan pembuatan perangkat lunak (*Software*).

Pada pembuatan perangkat kerat (*Hardware*) diawali dengan proses mengidentifikasi komponen-komponen fisik yang akan digunakan pada sistem yang akan dibuat, seperti mikrokontroler ESP32, sensor suhu DS18B20, sensor intensitas cahaya BH1750, sensor *water flow* YF-S201, relay optocoupler, pompa air 12 VDC, *Nozzle* 180°, PSU 12 VDC, dan komponen lainnya. Kemudian melakukan proses perancangan dan perakitan perangkat keras berdasarkan rancangan sistem yang telah disusun. Pembuatan perangkat keras ini menjadi dasar dari realiasi alat karena seluruh proses pendinginan panel surya bergantung pada kerja dari komponen-komponen yang terintegrasi dengan baik.

Tahapan selanjutnya yaitu pembuatan perangkat lunak (*Software*). Pada tahap ini merupakan proses pengembangan program untuk logika kendali pada alat yang akan diimplementasikan pada mikrokontroler ESP32. Perangkat lunak ini membuat sistem dapat bekerja sesuai fungsi yang dirancang dimana untuk mengatur kerja sensor suhu dan sensor intensitas cahaya sebagai parameter input, dan mengendalikan relay optocoupler serta pompa air sebagai output. Hubungan antara parameter masukan dan keluaran tersebut ditentukan menggunakan metode logika fuzzy Mamdani, yang berfungsi mengolah data suhu dan intensitas cahaya menjadi keputusan berupa volume air penyiraman yang sesuai dengan kondisi

operasi panel surya. Selain itu, sistem juga dilengkapi dengan konektivitas WiFi sehingga dapat mengirimkan data ke Google *Spreaadsheet* sebagai monitoring dan data logger. Pengembangan program dilakukan menggunakan Arduino IDE dengan struktur pemrograman yang mendukung efisiensi dan kemudahan dalan pengujian dan pengembangan lebih lanjut.

## 4.2 Alat dan Bahan

Dalam pelaksanaan pembuatan sistem Tugas akhir ini, diperlukan berbagai alat dan bahan yang berfungsi sebagai komponen utama maupun pendukung dalam proses perancangan dan pengembangan sistem sehingga sistem dapat berjalan secara berhasil dan sesuai dengan fungsi. Identifikasi terhadap kebutuhan tersebut dilakukan sebagai langkah awal untuk memastikan seluruh komponen yang digunakan sesuai dengan kebutuhan sistem pendingin panel surya berbasis logika fuzzy dengan ESP32. Kegiatan ini bertujuan untuk menjamin bahwa perangkat keras dan perangkat lunak yang digunakan dapat bekerja secara optimal sesuai dengan spesifikasi yang telah ditentukan. Selain itu, hasil identifikasi kebutuhan juga menjadi dasar dalam proses pengadaan, perakitan, serta pengujian sistem sehingga dapat mendukung terciptanya sistem yang terintegrasi, andal, dan mampu menjalankan fungsinya sesuai dengan tujuan penelitian. Untuk bahan yang digunakan untuk pembuatan tugas akhir ini ditunjukkan pada tabel 4.1.

**Tabel 4.1** Bahan Pembuatan Tugas Akhir

| No | Nama | Jumlah |
|---|---|---|
| 1 | Panel Surya 30 Wp | 1 |
| 2 | ESP32 | 1 |
| 3 | Sensor Intensitas Cahaya BH1750 | 1 |
| 4 | Sensor Suhu DS18B20 | 2 |
| 5 | Sensor *Water Flow* YF-S201 | 1 |
| 6 | PCB | 1 |
| 7 | Pompa Air 12 VDC 100 PSI | 1 |
| 8 | Relay Optocoupler | 1 |
| 9 | LCD 16x2 | 1 |

| No | Nama | Jumlah |
|---|---|---|
| 10 | PSU 12 VDC | 1 |
| 11 | Lampu Indikator | 3 |
| 12 | *Nozzle* 180° | 1 |
| 13 | IC LM7805 | 2 |
| 14 | Box Kontrol 215 x 145 x 85 mm | 1 |
| 15 | Pipa Air 1/2 inch | 1 |
| 16 | Selang Air | 1 |
| 17 | Rangka Panel Surya (Besi) | 1 |
| 18 | Kapasitor 1000 $\mu F$ | 1 |
| 19 | Kapasitor 100 $\mu F$ | 2 |
| 20 | Resistor 1K | 3 |
| 21 | Resistor 10K | 3 |
| 22 | Resistor 4.7K | 1 |
| 23 | Terminal Blok 2 Pin | 4 |
| 24 | JST 2 Pin | 2 |
| 25 | JST 3 Pin | 4 |
| 26 | JST 4 Pin | 1 |
| 27 | Transistor 2N2222 | 3 |
| 28 | Push Button | 1 |
| 29 | Baut 10 mm | 2 |
| 30 | Jack DC 1 set | 1 |
| 31 | Power Socket AC-027 | 1 |

Selanjutnya yaitu alat yang digunakan dalam proses perakitan tugas akhir ditunjukkan pada tabel 4.2.

| No | Nama | Jumlah |
|---|---|---|
| 1 | Obeng | 1 |
| 2 | Solder | 1 |
| 3 | Multimeter Digital | 1 |

| No | Nama | Jumlah |
|---|---|---|
| 4 | Double Tip | 1 |
| 5 | Kabel Ties | 15 |
| 6 | Gelas Ukur | 1 |
| 7 | Tang Potong | 1 |
| 8 | Laptop +Arduino IDE | 1 |
| 9 | Microsoft Excel | 1 |
| 10 | MATLAB | 1 |

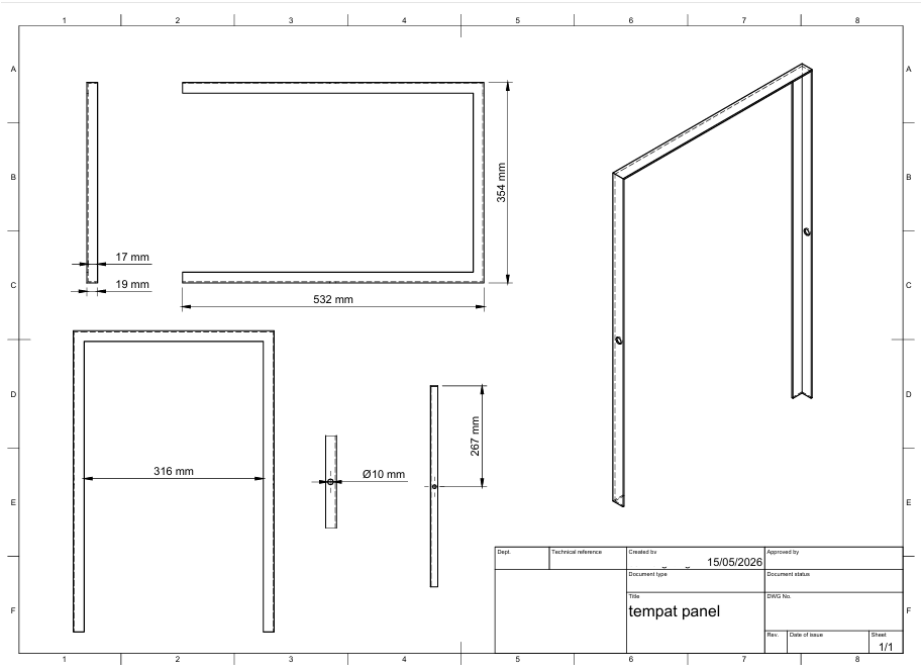
## 4.3 Pembuatan Perangkat Keras (*Hardware*)

Pembuatan perangkat keras merupakan tahap awal proses realisasi sistem dimana yang terdiri dari pembuatan dan perakitan sisi elektrikal maupun mekanikal. Tahap ini diawali dengan proses pengujian pada masing-masing komponen untuk memastikan seluruh komponen dapat berfungsi dengan baik dan tidak ada kerusakan. Kemudian untuk menampatkan komponen-komponen tersebut dengan merancang dan membuat struktur mekanik dari alat. Struktur ini berfungsi untuk penopang panel surya, tempat pipa air dan *nozzle* untuk proses pendinginan panel surya, serta tempat bagi seluruh komponen pendukung lainnya seperti box kontrol, pompa air, sensor, dan lainnya. Adapun perancangan dari rangka mekanik dengan mempertimbangkan dimensi panel surya dan peletakan komponen seperti pipa air dan box kontrol.

Selanjutnya, pada sisi elektrikal dilakukan adalah merancang layout PCB berdasarkan skema rangkaian yang telah ditentukan. Layout PCB dirancang untuk menentukan tata letak komponen serta jalur koneksi yang menghubungkan pin input, output, dan catu daya pada setiap komponen. Setelah proses desain selesai, dilakukan proses perakitan dengan memasang dan menyolder seluruh komponen, seperti ESP32, relay optocoupler, sensor, serta modul pendukung lainnya pada PCB. Melalui integrasi tersebut, terbentuk sebuah sistem yang memungkinkan komunikasi antara ESP32 dengan sensor dan aktuator sehingga fungsi monitoring dan pengendalian pada sistem pendingin panel surya dapat dijalankan sesuai dengan rancangan.

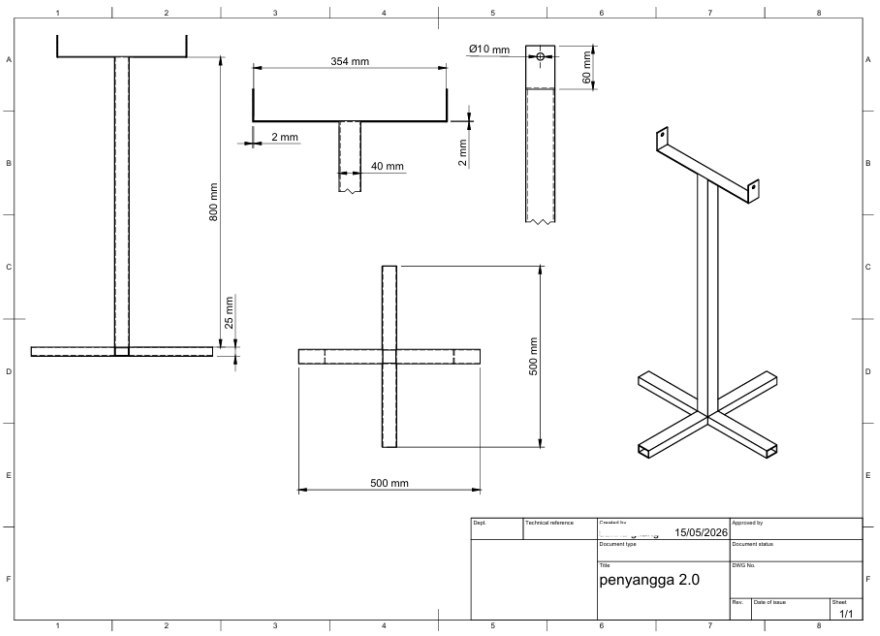
### 4.3.1 Perancangan Mekanik

Pada sisi mekanik, perancangan diawali dengan melakukan pengukuran dimensi fisik panel surya sebagai dasar dalam menentukan kebutuhan ukuran konstruksi penyangga. Parameter yang diukur meliputi panjang, lebar, serta posisi titik pemasangan panel. Berdasarkan hasil pengukuran tersebut, dilakukan pembuatan desain mekanik menggunakan perangkat lunak CAD untuk menentukan dimensi penyangga panel dan tiang penopang yang sesuai. Desain yang telah dibuat kemudian digunakan sebagai acuan dalam proses pembuatan rangka sehingga diperoleh konstruksi yang memiliki kekuatan mekanik yang memadai, stabil, serta mampu mendukung pemasangan panel surya dan komponen pendukung lainnya.




**Gambar 4.1** Kerangka Penyangga Panel Surya

Gambar 4.1 merupakan desain dan ukuran dari kerangka panel surya, dimana desain tersebut disesuaikan dengan bentuk panel surya secara asli. Kerangka tersebut digunakan untuk menopang panel surya dan menjaga posisi panel surya agar tetap stabil selama proses pengoperasian. Desain tersebut dirancang dengan ukuran rangka utama sebesar 532 mm × 354 mm. Selain itu, pada bagian tengah penyangga disediakan lubang pemasangan berdiameter 10 mm yang berfungsi sebagai titik pengikat menggunakan baut.




**Gambar 4.2** Tiang Penyangga Panel Surya

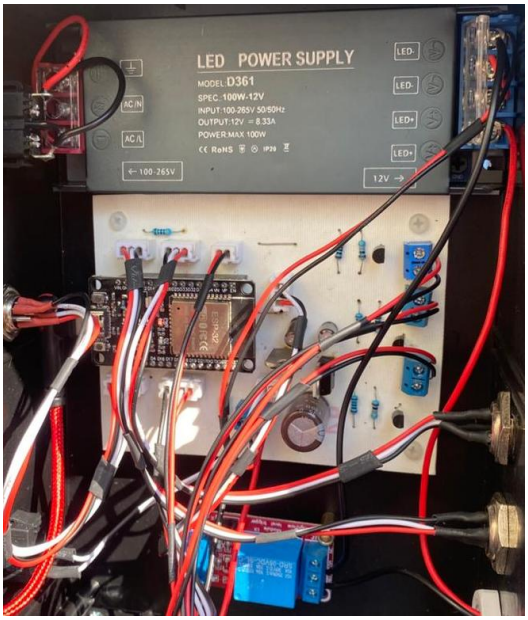
Gambar 4.2 merupakan tiang penyangga panel surya yang berfungsi untuk menopang rangka panel surya agar dapat dipasang pada posisi yang stabil selama proses pengoperasian dan pengujian sistem. Tiang penyangga memiliki tinggi 800 mm yang terdiri dari tiang vertikal yang menghubungkan bagian dudukan panel dengan alas penyangga. Adapun ukuran dari material tiang tersebut 40 mm × 40 mm dengan ketebalan 2 mm. Pada bagian atas terdapat dudukan panel sepanjang 354 mm yang dilengkapi lubang berdiameter 10 mm sebagai titik pemasangan rangka panel surya. Sementara itu, bagian alas dirancang berbentuk silang (cross base) dengan dimensi 500 mm × 500 mm sehingga mampu menjaga keseimbangan sistem dan mengurangi risiko terguling akibat beban panel.

**Gambar 4.3** Konstruksi Kerangka Penyangga Panel Surya

Gambar 4.3 menunjukkan konstruksi kerangka penyangga panel surya yang dibuat berdasarkan desain mekanik yang telah dirancang sebelumnya. Kerangka menggunakan material besi siku dan besi hollow sebagai struktur utama penopang panel surya. Penggunaan material jenis ini karena memiliki kekuatan yang dapat menopang beban panel surya dan mudah untuk difabrikasi. Bagian atas dirancang sesuai dengan dimensi panel sehingga panel dapat terpasang dengan stabil, sedangkan bagian tiang dan alas penyangga berfungsi meningkatkan kestabilan konstruksi dengan mendistribusikan beban secara merata. Selain itu, kerangka juga menyediakan ruang untuk pemasangan komponen pendukung sehingga seluruh sistem dapat terintegrasi dalam satu konstruksi yang kokoh.

### 4.3.2 Perancangan Elektrik

Pada perancangan elektrik sistem pendingin panel surya berbasis logika fuzzy dilakukan dengan menggunakan papan PCB (*Printed Circuit Board*) yang dibuat secara khusus untuk menampung dan mengintegrasikan seluruh komponen elektronik utama ke dalam satu rangkaian yang rapi dan efisien.




**Gambar 4.4** Penempatan Komponen

Gambar 4.4 menunjukkan tata letak komponen pada box kontrol dan juga PCB yang dirancang dengan mempertimbangkan aspek kemudahan perakitan dan optimalisasi ruang yang tersedia. Dengan penempatan komponen yang terstruktur, jalur koneksi antar komponen dapat dibuat lebih efisien sehingga menghasilkan rangkaian yang lebih ringkas dan mudah dalam proses implementasi.

Pada PCB sistem pendingin panel surya berbasis logika fuzzy, mikrokontroler ESP32 diletakkan pada tengah sisi kanan PCB, pemilihan posisi ini untuk mempersingkat jalur koneksi, meningkatkan efisiensi tata letak PCB, memudahkan proses pengkabelan, serta mempermudah akses saat proses pemrograman dan pemeliharaan sistem. Pada sisi atas PCB terdapat 2 buah konektor JST 3 pin yang dimana pin tersebut digunakan untuk menghubungkan sensor DS18B20 dan sensor YF-S201. Kemudian, pada sisi atas juga terdapat JST 2 pin yang terhubung dengan tombol reset sistem. Kemudian, pada sisi bawah ESP32 terdapat 2 buah konektor JST 4 pin yang digunakan untuk menghubungkan dengan sensor BH1750 serta LCD sebagai media tampilan data. Pada sisi samping kiri ESP 32 terdapat JST 3 pin untuk menghubungkan ke relay optocoupler sebagai aktuator yang berfungsi mengendalikan pompa air berdasarkan hasil perhitungan logika fuzzy. Penggunaan konektor JST dipilih karena memiliki ukuran yang ringkas, pemasangan yang

mudah, serta memungkinkan proses pelepasan dan penggantian komponen dilakukan tanpa perlu menyolder ulang rangkaian.

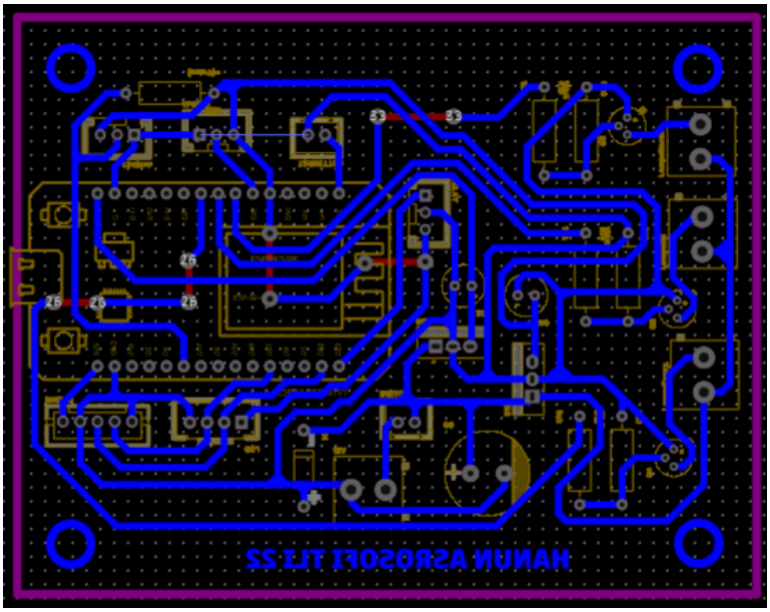
Selain konektor JST, PCB juga dilengkapi dengan terminal block yang berfungsi sebagai titik koneksi untuk komponen yang menggunakan kabel berukuran lebih besar dan memerlukan sambungan yang lebih kuat. Terminal block digunakan untuk menghubungkan sumber catu daya 12 VDC dari power supply serta lampu indikator yang dipasang pada panel box kontrol. Penggunaan terminal block dipilih karena memiliki sistem penjepitan kabel yang kuat sehingga dapat meningkatkan keamanan dan keandalan sambungan listrik selama sistem beroperasi.

Pada beberapa bagian papan PCB juga terdapat sejumlah resistor, kapasitor, dan komponen pasif lainnya yang dipasang pada sekitar jalur rangkaian dan terdapat IC untuk menunjang kestabilan sinyal dan sistem secara keseluruhan. Selain itu pada perancangan elektrik, terdapat komponen lain seperti power supply dan juga relay optoucpler yang terpasang dipisah tidak pada PCB. Pemisahan ini berfungsi untuk untuk mengurangi potensi gangguan elektromagnetik (*electromagnetic interference*) dan lonjakan tegangan yang dapat memengaruhi kinerja mikrokontroler maupun sensor.

Proses perakitan komponen dipasang pada box kontrol yang dimana pada box tersebut terdapat PCB, relay optocoupler, serta power supply 12 VDC. Adapun proses perakitan diawali dengan pemasangan komponen pasif seperti resistor, IC dan komponen kecil lainnya, dilanjutkan dengan pemasangan soket, terminal, dan modul. Setelah semua komponen terpasang, dilakukan pengujian awal dengan program sederhana untuk memastikan semua jalur bekerja dan berfungsi dengan baik.

## 4.4 Perancangan PCB

Perancangan PCB (*Printed Circuit Board*) dilakukan untuk mengintegrasikan seluruh rangkaian elektronik sistem pendingin panel surya berbasis logika fuzzy ke dalam satu papan rangkaian yang lebih ringkas, rapi, dan mudah diimplementasikan.



**Gambar 4.5** Desain PCB

Gambar 4.5 merupakan desain PCB dari sistem pendingin panel surya berbasis logika fuzzy dengan ESP32. PCB tersebut memiliki fungsi sebagai tempat untuk seluruh komponen sistem saling terhubung dan terintegrasi yang dikendalikan oleh mikrokontroler ESP32 sehingga dapat menghasilkan sistem yang berfungsi dengan baik.

Pada papan PCB, terdapat dudukan khusus untuk mikrokontroler ESP32 yang ditempatkan pada bagian tengah rangkaian. Penempatan tersebut dilakukan untuk mempermudah proses routing jalur serta mempersingkat koneksi antara mikrokontroler dengan sensor maupun perangkat keluaran. Selain itu, tersedia beberapa konektor JST yang digunakan untuk menghubungkan sensor suhu DS18B20, sensor intensitas cahaya BH1750, sensor debit air YF-S201, LCD, tombol reset, serta modul relay. Penggunaan konektor JST bertujuan untuk memudahkan proses pemasangan dan pelepasan komponen tanpa perlu melakukan penyolderan ulang sehingga proses perawatan dan penggantian komponen dapat dilakukan dengan lebih mudah. Selain konektor JST, PCB juga dilengkapi dengan terminal block yang digunakan sebagai titik koneksi untuk sumber catu daya dan lampu indikator yang dipasang pada panel box kontrol.

Pada PCB juga terdapat rangkaian catu daya yang terdiri dari dua buah regulator tegangan LM7805 yang berfungsi menurunkan tegangan 12 VDC dari power supply menjadi 5 VDC. Selain regulator, dipasang pula kapasitor elektrolit berfungsi sebagai filter tegangan untuk mengurangi riak (*ripple*) dan menjaga kestabilan suplai daya. Beberapa resistor digunakan sebagai pembatas arus, *pull-up resistor*, maupun penstabil logika pada jalur rangkaian tertentu. Selain itu, transistor digunakan sebagai penguat arus dan sakelar elektronik untuk mengendalikan perangkat keluaran yang membutuhkan arus lebih besar. Jalur-jalur PCB dirancang menggunakan dua lapisan (*double layer*) sehingga proses routing dapat dilakukan secara lebih efisien dan mengurangi persilangan jalur. Keseluruhan desain PCB dibuat dengan mempertimbangkan kemudahan perakitan, kestabilan distribusi daya, serta kemudahan pemeliharaan sistem.

## 4.5 Pembuatan Perangkat Lunak

Setelah sistem pada tahap pembuatan dan perakitan pada perangkat keras, tahap selanjutnya yaitu pembuatan perangkat lunak yang merupakan inti dari pengendalian sistem secara otomatis. Program pada mikrokontroler dilakukan dengan menggunakan *software* Arduino IDE dengan board ESP32 Dev Module yang dipilih karena menyediakan dukungan yang sesuai terhadap fitur dan spesifikasi mikrokontroler ESP32 sehingga proses pemrograman, kompilasi, dan pengunggahan program dapat dilakukan dengan mudah. Program dibuat dengan logika utama yang menggunakan beberapa library penting seperti WiFi.h untuk konektivitas jaringan, HTTPClient.h untuk pengiriman dan monitoring data melalui Google *Spreadsheet*, Wire.h untuk komunikasi protokol I2C, sehingga seluruh proses pengolahan data sensor hingga pengaktifan aktuator dapat berjalan dengan baik dan sesuai dengan yang dirancang.

### 4.5.1 Inisialisasi Sistem

Tahap awal dari pembuatan perangkat lunak yaitu proses inisialisasi sistem, tahap ini dilakukan agar mikrokontroler (ESP32) mengetahui tugasnya, seperti pada sisi perangkat lunak yaitu konfigurasi pin, pengaturan komunikasi data, sedangkan pada perangkat keras seperti proses penyaluran daya dan pengecekan status awal

sensor dan aktuator sehingga memastikan semua koneksi siap operasi sebelum menjalankan logika utama.

```
hprogram_tes2_Spreadd_FIX.ino
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>

//============== WIFI ==================
char ssid[] = "OPPO A74";
char pass[] = "123456789";

//========== GOOGLE SHEET URL ==========
String GAS_ID = "AKfycbzw6X4YbZUbg-s45X2EZpz3IhWvspTcoCKQXxzIzVEccqBYnSW06kjj8_TP41J2E3Zb";

//============== PIN ==================
#define ONE_WIRE_BUS 16
#define FLOW_SENSOR_PIN 32
#define RELAY_PIN 23
#define LAMP_ON 25
#define LAMP_FUZZY 26
#define LAMP_NORMAL 33

//============ SENSOR ==================
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
BH1750 lightMeter;
LiquidCrystal_I2C lcd(0x27,16,2);

//========== FLOW SENSOR ==============
volatile int pulseCount = 0;
float flowRate;
float flowMilliLitres;
```

```
hprogram_tes2_Spreadd_FIX.ino
//=====================================
void setup()
{
  Serial.begin(115200);
  lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x23);

  pinMode(RELAY_PIN, OUTPUT);
  pinMode(LAMP_ON, OUTPUT);
  pinMode(LAMP_FUZZY, OUTPUT);
  pinMode(LAMP_NORMAL, OUTPUT);

  digitalWrite(RELAY_PIN, LOW);

  sensors.begin();

  Wire.begin(21,22);
  lightMeter.begin();

  lcd.init();
  lcd.backlight();

  pinMode(FLOW_SENSOR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), pulseCounter, FALLING);

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    digitalWrite(LAMP_ON, LOW);
    delay(500);
    Serial.print(".");
  }
```

**Gambar 4.6** Inisialisasi Sistem

Pada gambar 4.5 ditunjukkan proses inisialisasi sistem yang dilakukan sebelum program utama dijalankan. Tahap ini diawali dengan pendefinisian pin-pin yang digunakan pada sistem menggunakan perintah #define, meliputi pin sensor suhu DS18B20, sensor debit air YF-S201, relay pompa, dan lampu indikator. Selanjutnya dilakukan pembuatan objek untuk sensor DS18B20, sensor intensitas cahaya BH1750, dan LCD I2C. Proses inisialisasi kemudian dilaksanakan pada fungsi setup(), yang mencakup aktivasi komunikasi serial, konfigurasi pin input dan output. Setelah seluruh komponen berhasil diinisialisasi dan sistem siap beroperasi, program akan masuk ke fungsi loop() yang berfungsi menjalankan proses pembacaan sensor, pengolahan logika fuzzy, pengendalian pompa, penampilan data pada LCD, dan pengiriman data ke Google *Spreadsheet* secara berulang selama sistem memperoleh catu daya.

### 4.5.2 Konfigurasi Sensor DS18B20

Salah satu komponen yang digunakan pada sistem yaitu sensor DS18B20 dimana sensor ini berfungsi sebagai parameter input dari logika fuzzy untuk mengukur suhu pada panel surya. Pada sistem menggunakan 2 buah sensor DS18B20. Untuk inisialisasi sensor ini meliputi penggunaan library DallasTemperature.h untuk pembacaan data suhu dari sensor, selanjutnya sensor ini

menggunakan komunikasi one wire yang menggunakan library OneWire.h. Langkah ini menjadi krusial untuk menentukan output dari logika fuzzy dan sinkronisasi dengan proses lainnya. Penjelasan lebih rinci mengenai inisialisasi perangkat lunak DS18B20 disampaikan pada bagian berikut.

```
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>

//================ WIFI & SHEET ==================
char ssid[] = "Din's Kost 1";
char pass[] = "Dilupeaz60";
String GAS_ID = "AKfycbzw6X4YbZUbg-s45X2EZpz3IhWvspTcoCKQXxzIzVEccqBYnSW06kjj8_TP41

//================ PIN ==================
#define ONE_WIRE_BUS 16
#define FLOW_SENSOR_PIN 32
#define RELAY_PIN 23
#define LAMP_ON 25
#define LAMP_FUZZY 26
#define LAMP_NORMAL 33

void bacaSensor() {
  sensors.requestTemperatures();
  suhuRata = (sensors.getTempCByIndex(0) + sensors.getTempCByIndex(1)) / 2.0;
  lux = lightMeter.readLightLevel();
```

**Gambar 4.7** Konfigurasi Sensor DS18B20

Pada gambar 4.6 merupakan konfigurasi koding yang digunakan untuk inisialisasi sensor. Proses ini diawali dari penggunaan library untuk menyediakan kumpulan fungsi, perintah, dan kode yang sudah dibuat sebelumnya sehingga tidak perlu membuat semuanya dari awal. Selanjutnya yaitu pendefinisian pin yang digunakan sebagai jalur komunikasi data sensor yaitu dengan GPIO 16. Kemudian pembuatan objek komunikasi OneWire pada pin yang telah ditentukan, sedangkan baris kedua membuat objek sensor DS18B20 yang akan digunakan untuk membaca data suhu. Untuk mengaktifkan sensor suhu menggunakan fungsi sensor.begin () yang dilanjutkan dengan fungsi sensors.requestTemperatures() untuk meminta data suhu dari sensor pertama dan kedua. Kemudian, perhitungan suhu rata-rata panel dengan menggunakan fungsi suhuRata = (sensor1+sensor2)/2 untuk menghasilkan suhu akhir yang digunakan pada input logika fuzzy.

### 4.5.3 Konfigurasi Sensor BH1750

Sensor yang kedua yang dipakai yaitu sensor BH1750 yang berfungsi mengukur intensitas cahaya yang ditangkap pada panel surya. Sensor ini sebagai parameter input pendukung yang digunakan untuk parameter logika fuzzy selain sensor DS18B20. Inisialisasi sensor ini dimulai dari menggunakan library Wire.h yang digunakan untuk komunikasi I2C antara ESP32 dan sensor BH1750. Kemudian menggunakan library BH1750.h digunakan untuk mengakses fungsi-fungsi pembacaan intensitas cahaya dari sensor BH1750. Penjelasan lebih rinci mengenai inisialisasi perangkat lunak BH1750 disampaikan pada bagian berikut.

```
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>

//================ WIFI & SHEET ==================
char ssid[] = "Din's Kost 1";
char pass[] = "Dilupeaz60";
String GAS_ID = "AKfycbzw6X4YbZUbg-s45X2EZpz3IhWvspTcoCKQXxzIzVEccqBYnSW06kjj8_TP41

//================ PIN ===================
#define ONE_WIRE_BUS 16
#define FLOW_SENSOR_PIN 32
#define RELAY_PIN 23
#define LAMP_ON 25
#define LAMP_FUZZY 26
#define LAMP_NORMAL 33

void setup() {
  Serial.begin(115200);

  Wire.begin(21, 22);
  lightMeter.begin();

 void bacaSensor() {
   sensors.requestTemperatures();
   suhuRata = (sensors.getTempCByIndex(0) + sensors.getTempCByIndex(1)) / 2.0;
   lux = lightMeter.readLightLevel();
```

**Gambar 4.8** Konfigurasi Sensor BH1750

Konfigurasi sensor BH1750 pada perangkat lunak diawali dengan pemanggilan library. Setelah library dipanggil, dilakukan pembuatan objek sensor menggunakan perintah BH1750 lightMeter; yang berfungsi sebagai media komunikasi antara program dan sensor BH1750. Selanjutnya, pada fungsi setup(), komunikasi I2C diinisialisasi menggunakan perintah Wire.begin(21,22) dengan

GPIO 21 sebagai jalur SDA (*Serial Data*) dan GPIO 22 sebagai jalur SCL (*Serial Clock*). Setelah komunikasi I2C aktif, sensor BH1750 diinisialisasi menggunakan perintah lightMeter.begin() sehingga sensor siap melakukan pengukuran intensitas cahaya. Pada proses pembacaan data, digunakan perintah lightMeter.readLightLevel() yang menghasilkan nilai intensitas cahaya dalam satuan lux (lx). Nilai lux yang diperoleh kemudian disimpan ke dalam variabel lux dan digunakan sebagai salah satu parameter masukan pada sistem logika fuzzy untuk menentukan volume air yang akan dikeluarkan pada proses pendinginan panel surya. Dengan adanya sensor BH1750, sistem dapat mengetahui kondisi intensitas cahaya lingkungan secara real-time sehingga keputusan yang dihasilkan oleh logika fuzzy menjadi lebih sesuai dengan kondisi operasi panel surya yang sebenarnya.

### 4.5.4 Konfigurasi Sensor YF-S201

Sensor debit air YF-S201 digunakan untuk mengukur jumlah volume air yang dialirkan oleh pompa pada proses pendinginan panel surya. Sensor ini bekerja dengan menghasilkan pulsa yang frekuensinya sebanding dengan laju aliran air yang melewati sensor. Inisialisasi perangkat lunak dilakukan dengan mendefinisikan pin input sensor pada ESP32 serta mengaktifkan fitur interrupt untuk menghitung jumlah pulsa yang dihasilkan secara real-time. Jumlah pulsa yang diperoleh kemudian diolah menggunakan faktor kalibrasi untuk mendapatkan nilai volume air yang telah dikeluarkan. Hasil perhitungan tersebut disimpan dalam variabel program dan digunakan sebagai acuan untuk mengendalikan durasi penyiraman sesuai volume target yang dihasilkan oleh sistem logika fuzzy.

```
//================ PIN ==================
#define ONE_WIRE_BUS 16
#define FLOW_SENSOR_PIN 32
```

```
//=========== VARIABEL GLOBAL ================
volatile int pulseCount = 0;
float flowRate;
float totalMilliLitres = 0;
float volKeluarSpreadsheet = 0.0;
float targetVolumeSpreadsheet = 0.0;
```

```
void loop() {
  // 1. KONTROL FLOW SENSOR REAL-TIME
  if (pulseCount > 0) {
    detachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN));
    int currentPulses = pulseCount;
    pulseCount = 0;
    attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), pulseCounter, FALLING);

    if (isCoolingCycle) {
      totalMilliLitres += (currentPulses / 0.45);

      volKeluarSpreadsheet = totalMilliLitres;

      float batasMatikan = targetVolumeSpreadsheet - offsetVolume;
      if (batasMatikan < 0) batasMatikan = 0;

      if (totalMilliLitres >= batasMatikan && targetVolumeSpreadsheet > 0) {
        digitalWrite(RELAY_PIN, LOW);
        isCoolingCycle = false;

        totalMilliLitres = targetVolumeSpreadsheet;
        volKeluarSpreadsheet = targetVolumeSpreadsheet;

        sedangJeda = true;
        waktuSelesaiSiklus = millis();
      }
    }
  }
```

**Gambar 4.9** Konfigurasi Sensor YF-S201

Proses inisialisasi pada sensor ini diawali dengan mendefisikan pin yang terhubung dengan ESP32 yaitu dengan menggunakan pin GPIO 32 sebagai pin input yang terhubung dengan keluaran pulsa dari sensor debit air YF-S201. Selanjutnya, Variabel pulseCount digunakan untuk menyimpan jumlah pulsa yang dihasilkan oleh sensor YF-S201. Variabel totalMilliLitres digunakan untuk menyimpan akumulasi volume air yang telah mengalir, sedangkan volKeluar*Spreadsheet* digunakan untuk menyimpan nilai volume yang akan dikirim ke Google *Spreadsheet*. Fungsi pulseCounter() merupakan *Interrupt Service Routine* (ISR) yang akan dijalankan secara otomatis setiap kali sensor menghasilkan pulsa. Penggunaan interrupt memungkinkan proses pencacahan pulsa dilakukan secara akurat tanpa terganggu oleh proses lain yang sedang berjalan pada ESP32. Perintah pinMode() digunakan untuk mengatur pin sensor sebagai input dengan resistor *pull-up* internal. Selanjutnya, fungsi attachInterrupt() digunakan untuk mengaktifkan interrupt pada pin sensor. Parameter FALLING

menunjukkan bahwa interrupt akan dipicu ketika terjadi perubahan sinyal dari logika HIGH ke LOW pada keluaran sensor YF-S201.

Selanjutnya pembacaan jumlah pulsa yang dihitung oleh sensor dengan fungsi interrupt. Interrupt dinonaktifkan sementara menggunakan detachInterrupt() agar nilai pulsa tidak berubah selama proses pembacaan berlangsung. Setelah nilai pulsa disimpan ke dalam variabel currentPulses, nilai pulseCount direset menjadi nol dan interrupt diaktifkan kembali menggunakan attachInterrupt(). Untuk mengonversi jumlah pulsa yang dihasilkan sensor menjadi volume air. Volume air dapat dihitung menggunakan persamaan:

$$Volume\ (mL) = \frac{Jumlah\ Pulsa}{0.45}$$

Nilai volume yang diperoleh kemudian dijumlahkan dengan volume sebelumnya dan disimpan pada variabel totalMilliLitres. Data volume tersebut selanjutnya dimanfaatkan sebagai umpan balik untuk menghentikan proses penyiraman ketika volume target yang dihasilkan oleh sistem logika fuzzy telah tercapai.

### 4.5.5 Konfigurasi Logika Fuzzy

Konfigurasi logika fuzzy pada perangkat lunak dilakukan melalui fungsi fuzzyLogic(). Fungsi ini merupakan inti dari sistem pengendalian karena bertugas mengolah data hasil pembacaan sensor suhu DS18B20 dan sensor intensitas cahaya BH1750 menjadi keputusan berupa volume air pendinginan yang harus dikeluarkan oleh pompa. Metode yang digunakan adalah logika fuzzy Mamdani yang terdiri atas 3 tahapan, yaitu fuzzifikasi, inferensi, dan defuzzifikasi.

```
// FUNGSI KEANGGOTAAN INPUT (SUHU)
//=====================================
float dingin(float t) {
  if (t <= 34.0) return 1.0;
  else if (t > 34.0 && t < 35.0) return (35.0 - t) / 1.0;
  else return 0.0;
}
float normal(float t) {
  if (t <= 34.0 || t >= 40.0) return 0.0;
  else if (t > 34.0 && t < 35.0) return (t - 34.0) / 1.0;
  else if (t >= 35.0 && t <= 38.0) return 1.0;
  else return (40.0 - t) / 2.0;
}
float panas(float t) {
  if (t <= 38.0 || t >= 55.0) return 0.0;
  else if (t > 38.0 && t < 40.0) return (t - 38.0) / 2.0;
  else if (t >= 40.0 && t <= 53.0) return 1.0;
  else return (55.0 - t) / 2.0;
}
float sangat_panas(float t) {
  if (t <= 53.0) return 0.0;
  else if (t > 53.0 && t < 55.0) return (t - 53.0) / 2.0;
  else return 1.0;
}

//=====================================
// FUNGSI KEANGGOTAAN INPUT (CAHAYA)
//=====================================
float cerah(float l) {
  if (l <= 20000.0) return 1.0;
  else if (l > 20000.0 && l < 22000.0) return (22000.0 - l) / (22000.0 - 20000.0);
  else return 0.0;

}
float sangat_cerah(float l) {
  if (l <= 20000.0) return 0.0;
  else if (l > 20000.0 && l < 22000.0) return (l - 20000.0) / (22000.0 - 20000.0);
  else return 1.0;
}

//=====================================
// FUNGSI KEANGGOTAAN OUTPUT (VOLUME AIR)
//=====================================
float calc_off(float z) {
  if (z == 0.0) return 1.0;
  else return 0.0;
}

float calc_sedikit(float z) {
  if (z < 100.0 || z >= 400.0) return 0.0;
  else if (z >= 100.0 && z <= 350.0) return 1.0;
  else return (400.0 - z) / 50.0;
}

float calc_sedang(float z) {
  if (z <= 350.0 || z >= 700.0) return 0.0;
  else if (z > 350.0 && z < 400.0) return (z - 350.0) / 50.0;
  else if (z >= 400.0 && z <= 650.0) return 1.0;
  else return (700.0 - z) / 50.0;
}

float calc_banyak(float z) {
  if (z <= 650.0) return 0.0;
  else if (z > 650.0 && z < 700.0) return (z - 650.0) / 50.0;
  else return 1.0;
```

```
void fuzzyLogic()
{
  float muDingin       = dingin(suhuRata);
  float muNormal       = normal(suhuRata);
  float muPanas        = panas(suhuRata);
  float muSangat_Panas = sangat_panas(suhuRata);

  float muCerah        = cerah(lux);
  float muSangat_Cerah = sangat_cerah(lux);

  float rule1 = min(muDingin, muCerah);               // R1 -> OFF
  float rule2 = min(muDingin, muSangat_Cerah);        // R2 -> OFF
  float rule3 = min(muNormal, muCerah);               // R3 -> Sedikit
  float rule4 = min(muNormal, muSangat_Cerah);        // R4 -> Sedikit
  float rule5 = min(muPanas, muCerah);                // R5 -> Sedikit
  float rule6 = min(muPanas, muSangat_Cerah);         // R6 -> Sedang
  float rule7 = min(muSangat_Panas, muCerah);         // R7 -> Sedang
  float rule8 = min(muSangat_Panas, muSangat_Cerah);  // R8 -> Banyak

  float maxVal = rule1; ruleAktif = "R1";
  if (rule2 > maxVal) { maxVal = rule2; ruleAktif = "R2"; }
  if (rule3 > maxVal) { maxVal = rule3; ruleAktif = "R3"; }
  if (rule4 > maxVal) { maxVal = rule4; ruleAktif = "R4"; }
  if (rule5 > maxVal) { maxVal = rule5; ruleAktif = "R5"; }
  if (rule6 > maxVal) { maxVal = rule6; ruleAktif = "R6"; }
  if (rule7 > maxVal) { maxVal = rule7; ruleAktif = "R7"; }
  if (rule8 > maxVal) { maxVal = rule8; ruleAktif = "R8"; }
  if (maxVal == 0.0)  { ruleAktif = "--"; }

float muOff     = max(rule1, rule2);
float muSedikit = max(rule3, max(rule4, rule5));
float muSedang  = max(rule6, rule7);
float muBanyak  = rule8;

float sumMuZ = 0;
float sumMu = 0;

for(int z = 0; z <= 1000; z += 10) {
  float mZ_off     = min(muOff, calc_off(z));
  float mZ_sedikit = min(muSedikit, calc_sedikit(z));
  float mZ_sedang  = min(muSedang, calc_sedang(z));
  float mZ_banyak  = min(muBanyak, calc_banyak(z));

  float max_mZ = max(mZ_off, max(mZ_sedikit, max(mZ_sedang, mZ_banyak)));

  sumMuZ += (max_mZ * z);
  sumMu  += max_mZ;
}

if (sumMu != 0) targetVolume = sumMuZ / sumMu;
else targetVolume = 0;
```

**Gambar 4.10** Konfigurasi Logika Fuzzy

Tahap pertama pada fungsi fuzzyLogic() adalah melakukan proses fuzzifikasi. Tahap ini merupakan proses mengubah nilai tegas (*crisp*) hasil pembacaan sensor menjadi nilai derajat keanggotaan (*membership value*) dengan rentang 0 sampai 1. Nilai suhu rata-rata (suhuRata) dan intensitas cahaya (lux) yang sebelumnya diperoleh dari fungsi bacaSensor() diubah menjadi derajat keanggotaan fuzzy. Pada variabel suhu, sistem membagi kondisi suhu menjadi 4 himpunan fuzzy, yaitu Dingin, Normal, Panas, dan Sangat Panas. Keempat himpunan tersebut

direalisasikan melalui fungsi dingin(), normal(), panas(), dan sangat_panas(). Untuk variabel intensitas cahaya terbagi menjadi 2 kondisi yaitu cerah dan sangat cerah yang di realisasikan melalui fungsi cerah() dan sangat_cerah(). Hasil dari proses fuzzifikasi disimpan pada variabel muDingin, muNormal, muPanas, muSangat_Panas, muCerah, dan muSangat_Cerah dengan rentang nilai antara 0 hingga 1. Nilai tersebut menunjukkan tingkat keanggotaan suatu data terhadap masing-masing himpunan fuzzy.

Setelah proses fuzzifikasi selesai, tahap berikutnya adalah inferensi fuzzy. Pada tahap ini dilakukan evaluasi terhadap delapan aturan (rule base) yang telah dirancang berdasarkan hubungan antara suhu panel surya dan intensitas cahaya terhadap kebutuhan pendinginan. Masing-masing aturan menggunakan operator logika AND yang direalisasikan dengan fungsi min(). Fungsi min() dipilih karena pada metode Mamdani nilai aktivasi suatu aturan ditentukan oleh derajat keanggotaan terkecil dari kedua variabel masukan. Dengan demikian, apabila salah satu kondisi memiliki nilai keanggotaan yang rendah, maka tingkat aktivasi aturan tersebut juga akan rendah.

Setelah seluruh nilai aktivasi aturan (*firing strength*) diperoleh melalui proses inferensi, program selanjutnya menentukan aturan yang memiliki nilai aktivasi terbesar. Proses ini dilakukan untuk mengetahui aturan (*rule*) yang paling dominan atau paling berpengaruh terhadap keputusan sistem pada kondisi masukan tertentu. Penentuan rule dominan diawali dengan menginisialisasi variabel maxVal menggunakan nilai dari rule1 serta memberikan identitas awal ruleAktif sebagai "R1". Selanjutnya program membandingkan nilai aktivasi setiap aturan secara berurutan menggunakan struktur percabangan if. Pada setiap proses perbandingan, apabila nilai aktivasi suatu aturan lebih besar dibandingkan nilai yang tersimpan pada variabel maxVal, maka nilai maxVal akan diperbarui dengan nilai aktivasi tersebut. Bersamaan dengan itu, variabel ruleAktif juga diperbarui sesuai dengan nama aturan yang sedang memiliki nilai aktivasi terbesar. Proses ini berlangsung secara berurutan mulai dari rule2 hingga rule8, sehingga pada akhir proses perbandingan, variabel maxVal akan berisi nilai aktivasi tertinggi dari seluruh

aturan, sedangkan ruleAktif akan menunjukkan identitas aturan yang memiliki pengaruh paling besar terhadap keputusan sistem.

Tahap selanjutnya adalah agregasi. Proses agregasi bertujuan menggabungkan beberapa aturan yang memiliki konsekuen (output) yang sama menjadi satu nilai keanggotaan keluaran. Pada tahap ini digunakan fungsi max() sebagai operator OR. Operator ini bekerja dengan memilih nilai aktivasi terbesar dari beberapa aturan yang menghasilkan keluaran yang sama. Nilai terbesar tersebut dianggap paling mewakili tingkat keanggotaan himpunan keluaran, karena menunjukkan aturan yang memiliki pengaruh paling kuat terhadap keputusan sistem. Hasil dari proses agregasi berupa empat nilai derajat keanggotaan, yaitu muOff, muSedikit, muSedang, dan muBanyak. Keempat nilai tersebut belum merupakan keputusan akhir sistem, melainkan masih berupa himpunan fuzzy keluaran yang menunjukkan tingkat keanggotaan masing-masing kategori volume air. Nilai-nilai inilah yang selanjutnya digunakan pada tahap defuzzifikasi untuk membentuk kurva keluaran gabungan dan menghitung nilai volume air secara tegas (*crisp*) menggunakan metode Centroid.

Tahap terakhir adalah defuzzifikasi menggunakan metode Centroid (Center of Area). Pada tahap ini, kurva keluaran hasil agregasi dihitung pada rentang volume 0–1000 mL dengan interval 10 mL. Proses ini direalisasikan menggunakan perulangan for, sehingga setiap nilai volume dalam rentang tersebut akan dihitung nilai derajat keanggotaannya terhadap masing-masing himpunan keluaran. Pada setiap nilai volume (z), program terlebih dahulu melakukan proses pemotongan kurva (*clipping*) terhadap fungsi keanggotaan keluaran menggunakan operator MIN. Proses clipping dilakukan dengan membandingkan nilai hasil agregasi (muOff, muSedikit, muSedang, dan muBanyak) dengan fungsi keanggotaan masing-masing keluaran (calc_off(), calc_sedikit(), calc_sedang(), dan calc_banyak()). Operator min() digunakan untuk membatasi tinggi kurva keluaran agar tidak melebihi nilai hasil inferensi. Dengan kata lain, nilai aktivasi hasil inferensi digunakan sebagai batas maksimum tinggi kurva keanggotaan pada setiap kategori volume air.

Setelah seluruh kurva keluaran dipotong, program melakukan penggabungan seluruh kurva keluaran menggunakan operator MAX. Penggabungan ini dilakukan untuk membentuk satu kurva fuzzy gabungan yang merepresentasikan hasil keseluruhan dari sistem inferensi. Pada setiap titik volume, program memilih nilai keanggotaan terbesar dari seluruh himpunan keluaran sehingga diperoleh satu nilai keanggotaan akhir. Hasil dari proses tersebut berupa satu kurva fuzzy keluaran yang merupakan gabungan dari seluruh kategori volume air, yaitu OFF, Sedikit, Sedang, dan Banyak. Kurva gabungan inilah yang selanjutnya digunakan sebagai dasar perhitungan metode centroid. Setelah diperoleh kurva gabungan, program menghitung momen dan luas daerah dari kurva tersebut. Perhitungan momen dilakukan dengan mengalikan setiap nilai volume (z) dengan nilai derajat keanggotaannya (max_mZ), kemudian hasilnya dijumlahkan dan disimpan pada variabel sumMuZ.

Variabel sumMuZ merepresentasikan jumlah momen dari seluruh titik pada kurva keluaran. Semakin besar nilai keanggotaan pada suatu volume, maka kontribusi volume tersebut terhadap hasil akhir akan semakin besar. Variabel sumMu merepresentasikan luas daerah di bawah kurva fuzzy hasil agregasi. Nilai ini digunakan sebagai penyebut pada perhitungan metode centroid sehingga hasil yang diperoleh benar-benar menunjukkan titik pusat dari kurva keluaran. Setelah seluruh proses iterasi selesai dilakukan, nilai volume akhir dihitung menggunakan persamaan metode Centroid, yaitu dengan membagi jumlah momen (sumMuZ) dengan jumlah luas daerah (sumMu). Apabila nilai sumMu tidak sama dengan nol, maka program menghitung nilai targetVolume Sebaliknya, apabila nilai sumMu sama dengan nol, maka program memberikan nilai targetVolume sebesar 0 mL.

Nilai targetVolume yang diperoleh dari proses defuzzifikasi selanjutnya digunakan sebagai acuan utama dalam pengendalian pompa air. Program akan membandingkan volume air yang telah dikeluarkan, yang diukur menggunakan sensor debit air YF-S201, dengan nilai targetVolume. Selama volume air yang keluar masih lebih kecil dari nilai target, relay tetap berada pada kondisi aktif sehingga pompa terus bekerja. Ketika volume air telah mencapai nilai

targetVolume, relay akan dimatikan secara otomatis sehingga proses pendinginan berhenti.

### 4.5.6 Monitoring dan Pengiriman Data ke Google *Spreadsheet*

Sistem monitoring dan pengiriman data ke Google *Spreadsheet* pada perangkat ini digunakan untuk merekam seluruh parameter kerja sistem secara real-time berbasis Internet of Things (IoT). Proses ini memanfaatkan konektivitas WiFi dengan menggunakan library WiFi.h pada ESP32 untuk mengirimkan data hasil pembacaan sensor dan keluaran logika fuzzy. Inisialisasi perangkat lunak dilakukan dengan mendefinisikan SSID dan password jaringan WiFi, serta ID Google Apps Script sebagai endpoint pengiriman data. Data tersebut dikirim secara berkala menggunakan protokol HTTP GET melalui library HTTPClient, sehingga seluruh parameter seperti suhu awal, suhu akhir, intensitas cahaya, volume air, dan rule fuzzy dapat terdokumentasi pada Google *Spreadsheet* untuk keperluan monitoring dan analisis sistem.

```
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>

//================ WIFI & SHEET ==================
char ssid[] = "Din's Kost 1";
char pass[] = "Dilupeaz60";
String GAS_ID = "AKfycbzw6X4YbZUbg-s45X2EZpz3IhWvspTcoCKQXxzIzVEccqBYnSW06kjj8_TP41J
```

```
// Timer Mutlak Spreadsheet 20 Detik
unsigned long lastSpreadsheetTime = 0;
const long intervalSpreadsheet = 20000;
```

```
// 4. TIMER SPREADSHEET MUTLAK TIAP 20 DETIK
if (millis() - lastSpreadsheetTime >= intervalSpreadsheet) {
  suhuAkhir = suhuRata;
  kirimSpreadsheet();
```

```
void kirimSpreadsheet()
{
  if(WiFi.status() == WL_CONNECTED)
  {
    HTTPClient http;
    String url = "https://script.google.com/macros/s/" + GAS_ID + "/exec?";

    url += "suhuAwal=" + String(suhuAwal);
    url += "&suhuAkhir=" + String(suhuAkhir);
    url += "&lux=" + String(lux);
    url += "&volKeluar=" + String(volKeluarSpreadsheet);
    url += "&volTarget=" + String(targetVolumeSpreadsheet);
    url += "&rule=" + ruleAktif;

    http.begin(url.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

    int httpCode = http.GET();

    Serial.println("========================================");
    Serial.print(">>> MENGIRIM DATA KE SPREADSHEET... Status: ");
    Serial.println(httpCode);
    Serial.println("========================================");

    http.end();
  }
}
```

**Gambar 4.11** Konfigurasi Monitoring dan Pengiriman Data ke Google *Spreadsheet*

Selain inisialisasi jaringan, sistem juga mendefinisikan beberapa variabel penting yang digunakan untuk proses monitoring dan pengiriman data seperti yang ditunjukkan pada gambar 4.11. Proses ini diawali dengan pendefinisian data utama yang dikirim ke monitor dan *Spreadsheet* yang ditunjukkan pada variabel seperti suhuAwal dan suhuAkhir digunakan untuk merekam kondisi suhu pada awal dan akhir siklus pengukuran, sedangkan suhuRata menyimpan nilai suhu rata-rata hasil pembacaan sensor DS18B20. Variabel lux digunakan untuk menyimpan nilai intensitas cahaya dari sensor BH1750. Selanjutnya, variabel volKeluar*Spreadsheet* dan targetVolume*Spreadsheet* digunakan untuk mencatat volume air aktual yang dikeluarkan serta volume target hasil perhitungan logika fuzzy yang akan dikirim ke *Spreadsheet*. Selain itu, variabel ruleAktif digunakan untuk menyimpan aturan fuzzy yang sedang dominan pada proses pengambilan keputusan. Seluruh variabel ini berfungsi sebagai data utama yang akan diproses, dimonitor, dan dikirimkan ke Google *Spreadsheet* sehingga sistem dapat melakukan pencatatan data secara real-time dan terstruktur untuk keperluan analisis performa sistem.

Selanjutnya, yaitu fungsi const long inteval*Spreadsheet* = 20000 digunakan untuk mengatur interval pengiriman data ke Google *Spreadsheet* setiap 20 detik. Fungsi if (millis() - last*Spreadsheet*Time >= interval*Spreadsheet*) berada pada fungsi loop() yan berfungsi sebagai "pemicu utama" pengiriman data. Jika selisih waktu sudah mencapai 20 detik, maka sistem akan masuk ke proses kirim data ke *Spreadsheet*. Kemudian terdapat fungsi void kirim*Spreadsheet* () yang berfungsi inti komunikasi cloud yang digunakan untuk mengirim data monitoring ke Google *Spreadsheet* Terdapat fungsi string url dimana bagian ini menyusun data dalam bentuk URL parameter (HTTP GET). Semua variabel sensor dan hasil fuzzy dimasukkan ke dalam URL agar dapat diterima oleh Google Apps Script dan disimpan ke Google *Spreadsheet*. Fungsi http.begin() digunakan untuk memulai koneksi ke server. Fungsi setFollowRedirects() berfungsi memastikan redirect Google Script tetap diikuti, dan fungsi http.GET() untuk mengirim data ke server. Terakhir fungsi http.end() Digunakan untuk mengakhiri koneksi agar memori ESP32 tidak penuh dan sistem tetap stabil.

### 4.5.7 Konfigurasi LCD

LCD (Liquid Crystal Display) pada penelitian ini digunakan sebagai media untuk menampilkan parameter kerja sistem secara *real-time*, sehingga pengguna dapat memantau kondisi panel surya dan proses pendinginan secara langsung. Informasi yang ditampilkan meliputi suhu rata-rata panel surya, intensitas cahaya, volume air yang telah dikeluarkan, serta volume target yang dihasilkan dari proses logika fuzzy. Konfigurasi perangkat lunak diawali dengan memanggil library LiquidCrystal_I2C.h sebagai library komunikasi LCD berbasis I2C, kemudian dilakukan inisialisasi komunikasi I2C menggunakan library Wire.h, pembuatan objek LCD sesuai alamat perangkat, serta inisialisasi LCD sehingga mampu menampilkan hasil pembacaan sensor dan keluaran logika fuzzy secara terus-menerus selama sistem beroperasi.

```
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <BH1750.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
 Wire.begin(21, 22);
 lightMeter.begin();
 lcd.init(); lcd.backlight();
  //============ LCD ==================
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T:"); lcd.print(suhuRata, 1); lcd.print("C ");
  lcd.print("L:"); lcd.print((int)lux);

  lcd.setCursor(0,1);
  lcd.print("V: "); lcd.print(totalMilliLitres, 0); lcd.print("/"); lcd.print(targetVolume, 0); lcd.print("mL");
```

**Gambar 4.12** Konfigurasi LCD

Proses konfigurasi diawali dengan memanggil library, kemudian membuat objek LCD menggunakan alamat I2C 0x27 dengan ukuran tampilan 16 kolom dan 2 baris melalui perintah LiquidCrystal_I2C lcd(0x27, 16, 2). Selanjutnya komunikasi I2C diinisialisasi menggunakan Wire.begin(21,22) dengan GPIO 21 sebagai jalur SDA dan GPIO 22 sebagai jalur SCL. Setelah komunikasi berhasil dilakukan, LCD diinisialisasi menggunakan fungsi lcd.init() dan lampu latar diaktifkan menggunakan lcd.backlight() sehingga tampilan dapat terbaca dengan jelas. Pada saat sistem berjalan, tampilan LCD diperbarui secara berkala menggunakan fungsi lcd.clear(). Baris pertama LCD digunakan untuk menampilkan nilai suhu rata-rata panel surya yang diperoleh dari sensor DS18B20 serta intensitas cahaya yang diukur oleh sensor BH1750. Sementara itu, baris kedua digunakan untuk menampilkan volume air yang telah dikeluarkan oleh pompa (totalMilliLitres) dan volume target (targetVolume) yang dihasilkan dari proses logika fuzzy. Informasi tersebut berfungsi sebagai media monitoring sehingga pengguna dapat mengetahui kondisi panel surya dan proses pendinginan yang sedang berlangsung secara langsung tanpa harus membuka Serial Monitor pada Arduino IDE.

### 4.5.8 Konfigurasi Lampu Indikator

Lampu indikator pada sistem berfungsi sebagai media visual untuk menunjukkan status operasi perangkat secara *real-time*. Penggunaan lampu indikator bertujuan untuk memudahkan pengguna dalam mengetahui kondisi sistem tanpa harus melihat tampilan LCD maupun Serial Monitor pada Arduino IDE. Pada penelitian ini digunakan tiga buah lampu indikator, yaitu lampu indikator koneksi

WiFi, lampu indikator proses pendinginan (*fuzzy*), dan lampu indikator kondisi normal.

```
//================ PIN ===================
#define ONE_WIRE_BUS 16
#define FLOW_SENSOR_PIN 32
#define RELAY_PIN 23
#define LAMP_ON 25
#define LAMP_FUZZY 26
#define LAMP_NORMAL 33
pinMode(RELAY_PIN, OUTPUT);
pinMode(LAMP_ON, OUTPUT);
pinMode(LAMP_FUZZY, OUTPUT);
pinMode(LAMP_NORMAL, OUTPUT);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED) {
  digitalWrite(LAMP_ON, LOW);
  delay(500);
  Serial.print(".");
}
digitalWrite(LAMP_ON, HIGH);
if (isCoolingCycle) {
  digitalWrite(RELAY_PIN, HIGH);
  digitalWrite(LAMP_FUZZY, HIGH);
  digitalWrite(LAMP_NORMAL, LOW);
  kondisiFuzzy = "PENDINGIN ON";
  Serial.println("Kondisi Pompa            : MENYALA (MENYIRAM)");
}
else {
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(LAMP_FUZZY, LOW);
  digitalWrite(LAMP_NORMAL, HIGH);
  kondisiFuzzy = "NORMAL";
```

**Gambar 4.13** Konfigurasi Lampu Indikator

Kode program diawali dengan mendefinisikan pin lampu indikator menggunakan perintah #define, yaitu LAMP_ON pada GPIO 25, LAMP_FUZZY pada GPIO 26, dan LAMP_NORMAL pada GPIO 33. Selanjutnya, pada fungsi setup(), ketiga pin tersebut dikonfigurasi sebagai pin keluaran (*output*) menggunakan fungsi pinMode(). Lampu LAMP_ON digunakan sebagai indikator status koneksi WiFi, dimana selama proses koneksi berlangsung lampu berada pada kondisi mati (LOW) dan akan menyala (HIGH) setelah ESP32 berhasil terhubung ke jaringan WiFi melalui pengecekan WiFi.status() == WL_CONNECTED. Sementara itu, lampu LAMP_FUZZY dan LAMP_NORMAL dikendalikan pada fungsi fuzzyLogic() sesuai dengan kondisi operasi sistem. Ketika variabel isCoolingCycle bernilai true, program mengaktifkan relay untuk menyalakan pompa sekaligus memberikan logika HIGH pada LAMP_FUZZY dan logika LOW pada LAMP_NORMAL, sehingga lampu indikator menunjukkan bahwa sistem

sedang melakukan proses pendinginan panel surya berdasarkan hasil logika fuzzy. Sebaliknya, apabila isCoolingCycle bernilai false, relay dimatikan sehingga pompa berhenti bekerja, lampu LAMP_FUZZY dipadamkan, dan lampu LAMP_NORMAL dinyalakan sebagai indikator bahwa sistem berada pada kondisi normal atau sedang tidak melakukan proses pendinginan. Dengan konfigurasi tersebut, ketiga lampu indikator mampu memberikan informasi visual mengenai status koneksi jaringan maupun kondisi operasi sistem secara real-time, sehingga pengguna dapat mengetahui kondisi perangkat tanpa perlu melihat tampilan LCD ataupun Serial Monitor.