BAB IV
PEMBUATAN ALAT

4.1.1 Pembuatan Perangkat Keras

Pembuatan perangkat keras merupakan tahap penting dalam mewujudkan alat pada sistem deteksi sambaran petir, guna dapat berfungsi sesuai rencana perancangan. Tahap ini dilaksanankan perakitan seluruh komponen, diawali dengan penyusunan skematik rangkaian pendeteksi elektronika berdasarkan komponen utama yang telah ditentukan, yakni mikrokontroler ESP32, sensor deteksi petir AS3935, sensor suhu, kelembaban dan tekanan BME280, penentu koordinat lokasi alat modul GPS NEO6MV. LCD 16 x 2 sebagai antarmuka visual disetiap alat. Pada sitstem supply alat menggunakan Baterai 18650mAh, dengan dilengkapi modul LM2596 untuk pengaturan tegangan. Sebagai pegendali dan bertukar mengirim dan menerima data ESP32 dirangakai dengan keseluruhan sistem, kemudian untuk *Monitoring* Google Spreadsheet yang digunakan untuk menyajikan data pembacaan alat, selain untuk menyajikan data dapat berperan juga sebagai *database*.

4.1.1 Alat dan Bahan

Tahap pertama pembuatan sistem adalah mempersiapkan alat dan bahan. Daftar alat yang akan digunakan dapat dilihat pada Table 4 -1 dan Table 4 – 2 berisi bahan yang digunakan dalam perancangan alat.

Table 4- 1 Alat

No	Nama Alat	Jumlah
1	Multimeter	1 Buah
2	Obeng Set	1 Buah
3	Gunting	1 Buah
4	Tang Kombinasi	1 Buah
5	Bor Tangan	1 Buah

No	Nama Alat	Jumlah
6	Solder	1 Buah
7	Cutter	1 Buah
8	Alat Tulis	1 Set
9	Gerinda	1 Buah

Table 4- 2 Bahan

No	Nama Bahan	Jumlah
1	ESP WROOM Devkit V1	1 Buah
2	Baterai Li-Ion 18650 2000 mAh	2 Buah
3	PCB	1 Buah
4	Sensor BME280	1 Buah
5	Modul GPS NEO6MV2	1 Buah
6	LCD 16 x 2	1 Buah
7	Modul I2C	1 Buah
8	Modul BMS TP5100 2A	1 Buah
9	Buckconverter LM2596	1 Buah
10	Buzzer 3,5V	1 Buah
11	Kabel Data USB to Micro B	1 Buah
12	Kapasitor Keramik 10pF	1 Buah
13	Kapasitor Milar 1nF	1 Buah
14	Kapasitor Keramik 0,1nF	1 Buah
15	Kapasitor Elektrolit 10uF	1 Buah
16	Resistor 270KOhm	2 Buah

No	Nama Bahan	Jumlah
17	Resistor 1MOhm	1 Buah
18	Resistor 10Kohm	1 Buah
19	Resistor 82Kohm	1 Buah
20	Resistor 3.9Kohm	1 Buah
21	Resistor 100Kohm	1 Buah
22	Dioda 1N4148	1 Buah
23	Antena *Telescopic*	1 Buah
24	Transistor PNP 2N4401	1 Buah
25	Transistor NPN 2N4403	2 Buah
26	Induktor 3,3mH	1 Buah
27	Induktor 1mH	1 Buah
28	BOX 18x15x5 cm	1 Buah
29	Timah Solder	1 Rol

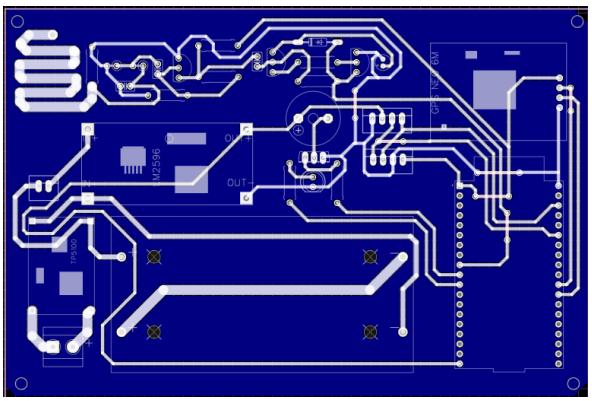
Pada keseluruhan alat dan bahan yang akan digunakan dipilih berdasarkan ketersediaan, keandalan, serta kesesuaian dengan kebutuhan sistem.

4.1.2 Pembuatan Perangkat Elektrik

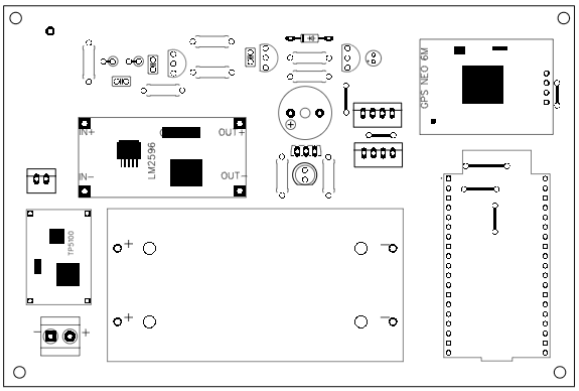
Dalam proses pembuatan perangkat elektronik, dilakukan tahapan perancangan dan perakitan seluruh komponen yang berfungsi sebagai pengendali sistem secara keseluruhan. Seluruh komponen tersebut disusun dalam satu sistem terintegrasi yang dikendalikan oleh mikrokontroller berbasis ESP32. Adapun tahapan dalam proses perangkat ini terdiri dari beberapa bagian penting.

1. Tahap perancangan skematik dan layout (PCB)

Perancangan skematik mencakup dua konfigurasi alat yang masing-masing menggunakan ESP32, sensor BME280, modul navigasi GPS NEO6, modul step-down LM2596, sistem manajemen baterai (*Battery Management System* atau BMS), serta baterai tipe 18650 berkapasitas 2000 mAh. Pada tahap ini, seluruh komponen ditempatkan secara sistematis, mencakup penataan jalur, pengukuran dimensi PCB, serta penambahan lubang dan label konektor sesuai kebutuhan.



Gambar 4. 1 Layout PCB bawah



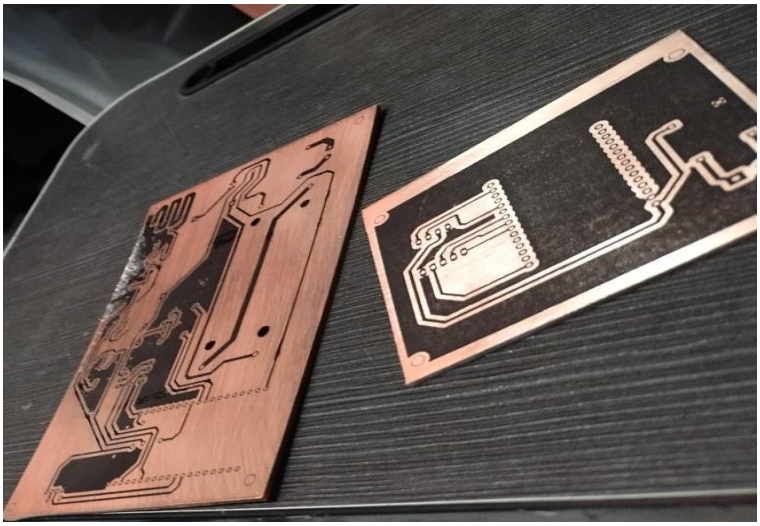
Gambar 4. 2 Layout PCB atas

Pada gambar 4.1 dan 4.2 diatas adalah selesainya tahap perancangan skematik dan layout PCB, diperoleh gambaran lengkap mengenai susunan komponen dan jalur koneksi antar komponen dalam sistem. Desain ini menjadi pedoman utama dalam proses pencetakan papan sirkuit dan perakitan perangkat keras. Tahapan ini

sangat penting untuk memastikan bahwa sistem bekerja secara terstruktur, efisien, dan minim kesalahan pada saat implementasi fisik.

2. Pencetakan PCB.

Proses ini diawali dengan pencetakan desain jalur PCB, kemudian dilakukan pemindahan pola jalur ke papan tembaga sebagai dasar sirkuit. Selanjutnya, papan tersebut direndam dalam larutan *ferri klorida* ($FeCl_3$) dalam proses etching untuk menghilangkan bagian tembaga yang tidak tertutup oleh pola. Setelah jalur terbentuk secara rapi, dilakukan pembersihan dan pelubangan sesuai tata letak komponen.



Gambar 4. 3 Pencetakan PCB

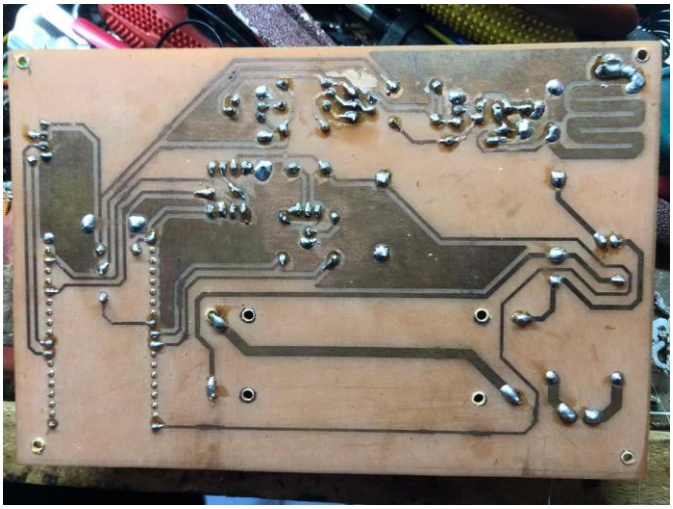
Dengan demikian gambar 4.3 diatas merupakan Percetakan PCB, proses percetakan PCB telah dilakukan sesuai dengan rancangan jalur yang telah dibuat sebelumnya. Hasil cetakan menunjukkan layout jalur yang presisi dan sesuai dengan desain skematik, yang menjadi dasar utama dalam proses pemasangan komponen. Tahap ini merupakan fondasi penting dalam memastikan kualitas koneksi antar komponen dan kestabilan sistem secara keseluruhan.

3. Penyolderan komponen ke permukaan PCB.

Komponen utama seperti ESP32 disolder secara permanen, dan untuk komponen yang letaknya tidak terintegrasi langsung di papan, digunakan pin header serta konektor JST untuk mempermudah proses penyambungan dan perawatan sistem. Tujuan dari penggunaan konektor ini adalah untuk meningkatkan fleksibilitas dan memudahkan proses penyesuaian terhadap kebutuhan sistem. Berikut gambar 4.4 dibawah ini saat melakukan penyolderan di PCB.

Gambar 4. 4 Penyolderan PCB

Gambar 4. 5 Tampilan PCB bagian Atas

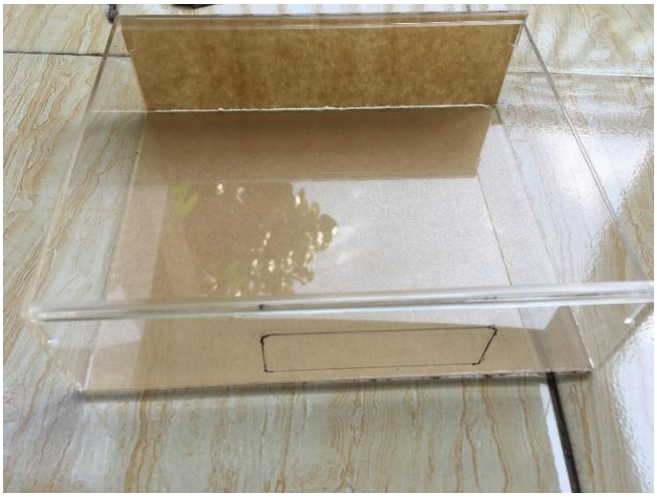


Gambar 4. 6 Tampilan PCB bagian Bawah

Pada gambar 4.5 dan 4.6 adalah hasil dari proses penyolderan komponen ke permukaan PCB, seluruh rangkaian elektronik telah terintegrasi secara menyeluruh dan siap digunakan. Penyolderan dilakukan dengan teliti agar setiap jalur koneksi antar komponen terhubung dengan baik dan tidak terjadi hubung singkat. Proses ini menjadi tahap penting dalam menjamin keandalan sistem secara keseluruhan, sebelum perangkat masuk ke tahap pengujian dan implementasi lebih lanjut.

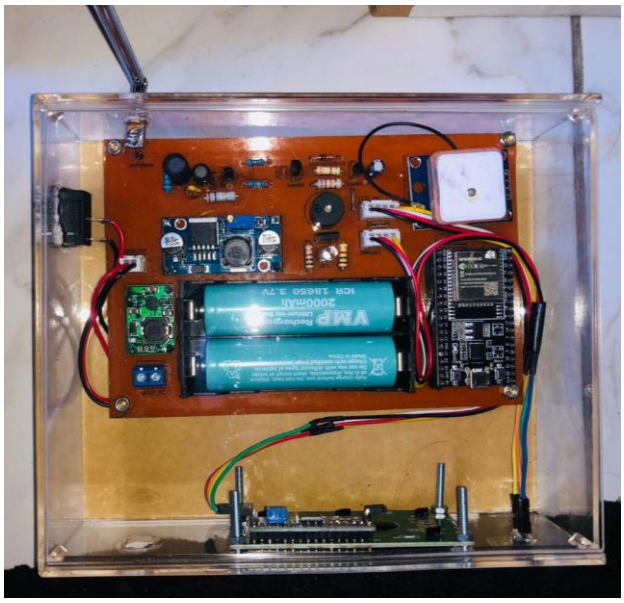
4.1.3 Pembuatan Perangkat Mekanik

Proses pembuatan bagian mekanik pada sistem deteksi sambaran petir bertujuan untuk melindungi serta menopang komponen elektronik agar sistem dapat berfungsi secara maksimal di lingkungan luar ruangan. Tahapan ini mencakup kegiatan pemilihan material, perancangan struktur mekanik, dan perakitan wadah yang memiliki ketahanan terhadap pengaruh cuaca. Pada gambar 4.7 dibawah ini adalah tampilan box yang akan digunakan.



Gambar 4. 7 Tampilan Box

Komponen-komponen seperti antena, sensor, dan modul ESP32 dirancang untuk dipasang secara strategis guna meminimalkan risiko gangguan fisik maupun interference dari lingkungan sekitar. Dalam proses ini, digunakan wadah berbahan plastik ABS yang bersifat tahan air dan memiliki tingkat perlindungan minimal IP65, sehingga mampu memberikan perlindungan terhadap cipratan air dan kelembapan tinggi. Antena diletakkan pada bagian atas enclosure guna memastikan jangkauan penerimaan sinyal elektromagnetik secara optimal.



Gambar 4. 8 Tampilan Perancangan Mekanik pada Box

Dengan demikian gambar 4.8 diatas adalah hasil dari perancangan mekanik pada sistem pendeteksi sambaran petir telah dirancang sedemikian rupa agar seluruh komponen elektronik tersusun secara rapi dan aman di dalam sebuah box pelindung. Tata letak komponen mempertimbangkan kemudahan perawatan, sirkulasi udara, serta kestabilan sistem saat digunakan di lapangan. Desain ini diharapkan mampu melindungi rangkaian dari gangguan luar seperti debu, kelembapan, dan getaran, sehingga menunjang keandalan alat dalam proses pendeteksian petir secara *real-time*.

4.2 Pembuatan Perangkat Lunak

Perangkat lunak pada sistem ini dirancang untuk mengelola komunikasi antara modul ESP32, sensor-sensor, serta pemantauan Google Spreadsheet. Pembuatan program dilakukan menggunakan bahasa pemrograman C++ dalam lingkungan pengembangan Arduino IDE. Pada tahap implementasinya, program diawali dengan proses inisialisasi koneksi Wi-Fi agar modul ESP32 dapat terhubung ke jaringan internet.

Selanjutnya, perangkat lunak mengatur proses pembacaan data dari berbagai sensor, termasuk sensor suhu, tekanan, serta sinyal elektromagnetik yang mengindikasikan adanya sambaran petir. Data yang diperoleh dari sensor kemudian diolah dan dikirimkan secara berkala ke Google Spreadsheet menggunakan protokol komunikasi *Firebase JSON*. Pengiriman data ini dilakukan dengan memanfaatkan *endpoint* dari Google Apps Script yang telah terhubung ke Spreadsheet.

Sebagai langkah autentikasi dan keamanan, sistem memanfaatkan *private key* yang telah ditentukan sebelumnya dalam kode program. Selain itu, perangkat lunak juga dilengkapi dengan logika untuk mencatat waktu sambaran petir dan menangani kemungkinan kegagalan jaringan atau pengiriman data. Dengan demikian, sistem ini memungkinkan pemantauan secara *real-time* melalui Google Spreadsheet sebagai antarmuka visual, sekaligus merekam riwayat data sambaran petir secara otomatis.

4.2.1 Pembuatan Program Pada Arduino IDE

Penyusunan program pada proyek ini dilakukan menggunakan platform Arduino IDE, yang berfungsi sebagai lingkungan pemrograman utama untuk mikrokontroler ESP32. Program yang dikembangkan memiliki beberapa fungsi utama, antara lain melakukan pembacaan data dari rangkaian pendeteksi sambaran petir, sensor suhu, serta sensor tekanan. Data yang diperoleh dari sensor-sensor tersebut selanjutnya dikirimkan ke platform pemantauan berbasis cloud, yaitu Google Spreadsheed, agar dapat diakses dan ditampilkan secara *real-time* melalui dashboard tersebut. Berikut ini pembagian dari program yang di kembangkan meliputi:

1. Program untuk koneksi ESP32 dengan jaringan WiFi dan Google Spreadsheet
2. Program untuk koneksi ke Rangkaian Skematik
3. Program untuk koneksi ke Modul Sensor BME 280
4. Program untuk koneksi ke Modul GPS
5. Program untuk koneksi ke Tampilan LCD
6. Program untuk koneksi ke Buzzer dan LED

4.2.2.1 Program untuk Koneksi ESP32 ke jaringan WiFi dan G. Spreadsheet

Pada Bab ini membahas tahapan pembuatan program yang bertujuan untuk menghubungkan modul ESP32 dengan jaringan WiFi serta platform *Monitoring* Google Spreadsheet. Koneksi ini menjadi komponen penting dalam sistem karena memungkinkan data hasil pembacaan sensor dikirim dan direkam secara otomatis melalui jaringan internet. Dalam prosesnya, digunakan protokol komunikasi berbasis *Firebase JSON* yang didukung oleh endpoint dari Google Apps Script, sehingga ESP32 dapat melakukan transmisi data secara langsung ke Google Spreadsheet.

```
//Inisialisasi Autentikasi Google Spreadsheet
#define PROJECT_ID "deteksi-petir-001"
#define CLIENT_EMAIL "deteksi-petir-001@deteksi-petir-001.iam.gserviceaccount.com"
/ Private key
const char PRIVATE_KEY[] PROGMEM = R"KEY(
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCp5sQKOEK4JOBC
94Gq7TWQc7Q2OM5iTbtUvPCnwRtkMS2t2D822LjyWhUGKlZQOdpGgzTZ/WeMqX6h
/K0M5ihxRXh09HL6q7R6Zlj0mz1TLUvhEnWsavXfmzht1gtKynK5pv7DZHb2LZLH
73vaXhL26ZHd25FKx7OHXh/LtgJhPaGehW1iVXhnhbJBEgh2mCiQhdmU3ai6sHTO
bFskTpWMYTMldN6zHNoQ2Z9Wgo9sanEGDQmRQgnNCzZCgmN3CqK44XKhIm2ZQhMV
ZbBAA8GKeUF2s4OGyCQf7706ryNXpHOF/o6GlRecOeHB99raqhROW01XccJMgNcK
hkC6pJvbAgMBAAECggEAGsjKJQD45lt3OKIY61+umbwz2ncayifjnW3qfne/SrdE
spWsZDLG9/s4OfBDX9hkTx4bdIuHXzHP8yzeAceS8/B2F0dQ4lsJU7Z+Kums7Eea
y7uuQ2Uv/R//mwkrCeUKhba247TF2vS/oWIMDhaAj4kDjuHbcNo0tOV4xmjwzRq4
1LE7KcPmmM6SbNQMS+86HuA3S5Ni0TdCh6F4yksdOTkTUXIqUUZDtw/wdbJv1JyA
/Cwr1l98OM0h6rO4ifJ3asdjRkgv+9OkF8KO+BKmRj3xNyneG+BRcW6NW794Tm+u
/XhgNxi5MBq1tXVmXEsQA5gebxjB4Fn+q8hRssTJ2QKBgQDve+qTkIqOie5rueqg
BfZA9jkHHt0/hoYHwRhmWkjOgJTzDLuX7I0j9Mnkdfbmcy8vA9E5zOSCAzwLhzCb
sJqv6slBzRmujF7UmCP5PiNOje5ZRoaDYqSVTaGtWvWDJyIU32Wiklexcsxn PLLB
3h0vpVHZzUwE5MpNKAYitqdB9QKBgQC1nl7TPKJ1UxGCQhw0g3nFW1J1g819g+Zo
a9cWpnXA/eonConj0TF9CY+KGyxd2FnAgu+aFYGEtW7LhYB/cHcROgIjBtfA02Ui
JDAQIIlthX6bDRI/98Z9yd9IrtE0eQbqCnHDcDLJdedoj3vtjtasz31HyMYQTU+Z
577Twu00jwKBgGvMX57k5TmcJNIg8dfuVxaOvsTRw82ghBxhctY/tZ4eiu2MwbR1
KmMRAP/xiZ3aHLNcCIX4ELPlONYHmH28VSTyjn0sEW9ciCMyCJ9ctQ/bG3rcIBhs
AqGiPDCITRMoPlMe6+Tt8sNRZWjjDaPukzyrMdjUbG+XfOiWJ4zgoxExAoGBAIzZ
19ym569pzuhgcRNg661dV6P/pfwzsRtGlzbfve8AfPnuOGZQNLsuEP4geNYHFgQx
B7VN/HZ5ZwW78Z3kENNxU3rqsaS/zW0e949qTGp932RMTN0DQhMH3ny151V0jyIv
G29gsmJ/aAo1TP09i8vV1uk3EjROYO+RRPNh9yiNAoGAT8NseeNCjwECJ/SeObn0
qgiRl4S0qwOpfPtV8iCvPga/kz8G9bW5qwR104Z+cSkrucaFxnCpYMd7WN+xDpJZ
Q4jHsVCMl9SO6W93ilYviMcknISwE1RSqTi6mjoYO+j05lUJLVwfMNgy6P0uhdXd
U3OtkMTH17+Or5CKbykjf5A=
-----END PRIVATE KEY-----
```

Gambar 4. 9 Inisialisasi Autentikasi Google Spreadsheet

Pada gambar 4.9 menunjukkan bagian program yang digunakan untuk melakukan inisialisasi autentikasi terhadap layanan Google Spreadsheet dengan menggunakan akun layanan (*service account*) yang telah didaftarkan melalui Google Cloud Platform. Dalam potongan program tersebut, terdapat tiga elemen utama yang didefinisikan, yaitu `PROJECT_ID` yang merepresentasikan identitas proyek, `CLIENT_EMAIL` yang berfungsi sebagai alamat email dari akun layanan yang digunakan, serta `PRIVATE_KEY` yang merupakan kunci privat dalam format PEM. Ketiga elemen ini digunakan untuk membentuk token autentikasi berbasis JSON Web Token (JWT**)** guna menjamin keamanan dan validitas komunikasi antara mikrokontroler ESP32 dan layanan Google Spreadsheet. Proses autentikasi ini bersifat otomatis dan tidak memerlukan interaksi langsung dari pengguna (*userless authentication*), sehingga mendukung prinsip kerja sistem berbasis *Internet Of Things* (IoT) yang menekankan pada efisiensi, keamanan, dan keterhubungan data secara *real-time*.

```
//Pemilihan Library
#include <WiFi.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "time.h"
#include <ESP_Google_Sheet_Client.h>
#include <GS_SDHelper.h>
```

Gambar 4. 10 Pemilihan Library

Pada Gambar 4.10 memperlihatkan pemilihan pustaka (*library*) yang digunakan dalam pengembangan sistem pendeteksi sambaran petir berbasis ESP32 dengan *Monitoring* melalui Google Spreadsheet. Beberapa pustaka standar seperti `WiFi.h` dan `Wire.h` digunakan untuk menginisialisasi koneksi jaringan nirkabel serta komunikasi antarperangkat melalui protokol I2C. Selain itu, pustaka `LiquidCrystal_I2C.h` digunakan untuk mengatur tampilan informasi pada modul LCD 16x2 berbasis antarmuka I2C, sedangkan pustaka `time.h` memungkinkan sistem memperoleh dan mengelola data waktu secara tepat.

Untuk integrasi dengan Google Spreadsheet, digunakan dua pustaka utama, yaitu `ESP_Google_Sheet_Client.h` dan `GS_SDHelper.h`. Pustaka `ESP_Google_Sheet_Client.h` berperan sebagai antarmuka utama untuk melakukan autentikasi dan pertukaran data antara ESP32 dan Google Spreadsheet, sedangkan `GS_SDHelper.h` berfungsi mendukung pengelolaan penyimpanan data jika dibutuhkan, seperti dalam proses *logging* atau *debugging*. Pemilihan pustaka-pustaka ini dirancang secara selektif agar mendukung fungsi utama sistem dan memastikan efisiensi dalam proses pemrograman serta pengolahan data secara *real-time*.

```
//Konfigurasi WiFi dan Spreadsheet
char ssid[] = "Ramadhan__";
char pass[] = "00000000";

GSheet.printf "(ESP Google Sheet Client v%s\n\n"
              ESP_GOOGLE_SHEET_CLIENT_VERSION);

void setup() {
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();

  GSheet.begin(CLIENT_EMAIL, PROJECT_ID, PRIVATE_KEY);

 configTime(7 * 3600, 0, ntpServer);  // GMT+7 untuk WIB
}
```

Gambar 4. 11 Konfigurasi Wifi dan Spreadsheet

Pada Gambar 4.11 menunjukkan proses konfigurasi awal untuk konektivitas WiFi dan integrasi dengan Google Spreadsheet. Pada bagian awal, variabel `ssid` dan `pass` didefinisikan untuk menyimpan nama jaringan WiFi (*Service Set Identifier*) dan kata sandi yang digunakan oleh modul ESP32 untuk terhubung ke jaringan internet. Koneksi internet ini merupakan prasyarat utama agar ESP32 dapat mengirimkan data ke layanan Google Spreadsheet secara daring (*online*).

Selanjutnya, fungsi `GSheet.printf()` digunakan untuk menampilkan versi dari pustaka ESP Google Sheet Client yang digunakan, berguna untuk kepentingan *debugging* dan dokumentasi. Pada bagian fungsi `setup()`, beberapa inisialisasi dilakukan seperti komunikasi serial melalui `Serial.begin()`, inisialisasi tampilan LCD (`lcd.init()`), serta pengaktifan lampu latar (`lcd.backlight()`).

Tahap krusial lainnya adalah pemanggilan fungsi `GSheet.begin()` yang memerlukan tiga parameter utama, yaitu `CLIENT_EMAIL`, `PROJECT_ID`, dan `PRIVATE_KEY`. Ketiga parameter ini merupakan bagian dari sistem autentikasi yang memastikan bahwa hanya perangkat yang terotorisasi yang dapat mengakses dan memodifikasi dokumen spreadsheet terkait. Terakhir, fungsi `configTime()` digunakan untuk mengatur zona waktu sesuai dengan wilayah Indonesia Barat

(GMT+7), yang penting untuk mencatat waktu kejadian secara akurat saat data dikirim ke Google Spreadsheet.

```
//Fungsi Cek Koneksi Wi-fi
void cekKoneksiWiFi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);  // Delay 1 detik per percobaan
    Serial.print(".");
    // Timeout koneksi: jika sudah lebih dari 30 detik, restart ESP
    if (retryCount > 30) {
      Serial.println("\nGagal konek WiFi. Restart ESP...");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Gagal Konek WiFi");
      lcd.setCursor(0, 1);
      lcd.print("Restart ESP...");
      delay(3000);
      ESP.restart();  // Restart otomatis
    }
  }

  // Jika konek berhasil
  Serial.println("\nWiFi Terhubung!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi Terhubung!");
  lcd.setCursor(0, 1);
  lcd.print(WiFi.localIP());
  delay(2000);
}
```

Gambar 4. 12 Fungsi Cek Koneksi Wi-Fi

Gambar 4.12 menunjukkan fungsi `cekKoneksiWiFi()` yang bertanggung jawab untuk memverifikasi koneksi WiFi pada modul ESP32. Fungsi ini mencoba menghubungkan ESP32 ke jaringan WiFi dengan memanggil `WiFi.begin(ssid, password)`, menggunakan parameter `ssid` dan `password` yang telah didefinisikan sebelumnya. Selanjutnya, dilakukan pengecekan status koneksi dengan `WiFi.status()`. Apabila belum terhubung (`!= WL_CONNECTED`), maka sistem akan menunggu selama satu detik sebelum mencoba kembali, ditandai dengan mencetak

titik (.) pada *Serial Monitor* sebagai indikator proses pengulangan. Jika jumlah percobaan melebihi 30 kali (sekitar 30 detik), maka sistem menganggap koneksi gagal. Sebagai respon, LCD akan menampilkan pesan "Gagal Konek WiFi" dan "Restart ESP...", kemudian ESP32 secara otomatis di-*restart* menggunakan perintah `ESP.restart()` untuk menginisiasi ulang koneksi.

Apabila koneksi berhasil terjalin, maka akan ditampilkan pesan “WiFi Terhubung!” beserta alamat IP lokal dari perangkat, baik pada *Serial Monitor* maupun pada tampilan LCD. Penambahan jeda selama dua detik (`delay(2000)`) memungkinkan pengguna untuk melihat informasi tersebut sebelum sistem melanjutkan proses berikutnya. Pendekatan ini menunjukkan ketahanan sistem dalam menjaga kestabilan koneksi, sekaligus memberikan umpan balik visual kepada pengguna.

```
//Sinkronisasi Waktu dengan NTP
String getFormattedTime() {
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    return "Failed to get time";
  }
  strftime(formattedTime, sizeof(formattedTime), "%d/%m/%Y %H:%M:%S", &timeinfo);
  return String(formattedTime);
}
```

Gambar 4. 13 Sinkronisasi Waktu dengan NTP

Program pada gambar 4.13 menunjukkan fungsi `getFormattedTime()` yang digunakan untuk melakukan sinkronisasi waktu dengan server NTP (Network Time Protocol) dan mengembalikan format waktu yang telah dirapikan sebagai string. Fungsi ini sangat penting dalam sistem *Monitoring* berbasis waktu, terutama ketika data yang dikirim ke *Google Spreadsheet* memerlukan penanda waktu (*timestamp*) yang akurat.

Di dalam fungsi tersebut, struktur `tm` digunakan untuk menyimpan informasi waktu lokal. Proses sinkronisasi dilakukan dengan memanggil fungsi `getLocalTime()`, yang secara otomatis mengambil waktu dari server NTP yang telah dikonfigurasikan sebelumnya. Apabila proses pengambilan waktu gagal,

maka fungsi akan mengembalikan pesan kesalahan `"Failed to get time"`. Jika waktu berhasil diperoleh, maka fungsi `strftime()` akan digunakan untuk memformat waktu dalam bentuk `dd/mm/yyyy HH:MM:SS`. Format ini kemudian dikonversi menjadi objek `String` dan dikembalikan sebagai output fungsi. Dengan pendekatan ini, sistem memperoleh keandalan dalam pencatatan waktu, yang krusial untuk keperluan dokumentasi, pelacakan, maupun analisis historis data yang terekam.

```
//Pengiriman Data ke Spreadsheet
if (ready && millis() - lastTime > timerDelay) {
  lastTime = millis();

  FirebaseJson response;
  FirebaseJson valueRange;

  // Get timestamp
  String waktuSekarang = getFormattedTime();

  valueRange.add("majorDimension", "COLUMNS");
  valueRange.set("values/[0]/[0]", waktuSekarang);
  valueRange.set("values/[1]/[0]", pulseGlobal);
  valueRange.set("values/[2]/[0]", String(suhuGlobal) + "°C");
  valueRange.set("values/[3]/[0]", String(kelembabanGlobal) + "%");
  valueRange.set("values/[4]/[0]", String(tekananGlobal) + "hPa");
  valueRange.set("values/[5]/[0]", latGlobal + "," + lngGlobal);

 bool success = GSheet.values.append(&response /* returned response */, spreadsheetId /* spreadsheet Id to append */,
                                     "Sheet1!A1" /* range to append */, &valueRange /* data range to append */);
  if (success) {
    response.toString(Serial, true);
    valueRange.clear();
  } else {
    Serial.println(GSheet.errorReason());
  }
  Serial.println();
  Serial.println(ESP.getFreeHeap());
}
```

Gambar 4. 14 Pengiriman Data ke Spreadsheet

Gambar 4.14 menunjukkan proses pengiriman data dari mikrokontroler ESP32 ke Google Spreadsheet menggunakan protokol komunikasi Firebase dengan format FirebaseJson. Pengiriman data dilakukan secara berkala dengan memanfaatkan fungsi `millis()` sebagai timer delay.

Langkah pertama dalam blok program ini adalah memperoleh waktu terkini melalui pemanggilan fungsi `getFormattedTime()`, yang telah dijelaskan sebelumnya. Data waktu tersebut kemudian dimasukkan sebagai baris pertama (indeks `[0][0]`) dari objek JSON `valueRange`, bersama dengan data dari variabel

lain seperti *pulse*Global, suhuGlobal, kelembabanGlobal, tekananGlobal, serta informasi koordinat lokasi latGlobal dan lngGlobal.

Objek valueRange dikonfigurasi dengan majorDimension sebagai "COLUMNS", yang berarti data akan ditambahkan per kolom. Setelah semua nilai disusun, data dikirimkan ke Google Spreadsheet menggunakan fungsi GSheet.values.append, yang menerima parameter berupa ID spreadsheet, range penempatan data ("Sheet1!A1"), dan objek valueRange.

Apabila proses pengiriman berhasil (success == true), data respon dicetak ke serial monitor dan memori JSON dikosongkan menggunakan valueRange.clear(). Jika gagal, maka akan ditampilkan pesan kesalahan dari fungsi GSheet.errorReason(). Selain itu, sisa memori heap juga ditampilkan untuk keperluan debugging dan *Monitoring* sistem. Proses ini menjadi komponen penting dalam sistem karena memungkinkan data sensor terekam secara *real-time* dan terdokumentasi dalam format tabular secara daring.

Dengan demikian, proses pengiriman data ke Google Spreadsheet telah berhasil diimplementasikan secara *real-time* menggunakan mikrokontroler ESP32 dan protokol FirebaseJson. Melalui integrasi ini, sistem mampu mencatat data lingkungan secara otomatis dan tersimpan dalam format yang dapat diakses serta dianalisis secara daring, sehingga meningkatkan efisiensi pemantauan dan keandalan dokumentasi data.

4.2.2.2 Program untuk Koneksi ke Rangkaian Skematik

Subbab ini menjelaskan perancangan program yang digunakan untuk menghubungkan mikrokontroler ESP32 dengan komponen-komponen yang terdapat pada rangkaian skematik sistem. Perancangan ini bertujuan untuk memastikan setiap perangkat keras, seperti sensor, modul tampilan, dan aktuator, dapat berkomunikasi secara tepat melalui konfigurasi pin yang sesuai. Program ini menjadi dasar utama dalam membentuk integrasi antara perangkat lunak dan perangkat keras sehingga sistem dapat berfungsi secara optimal.

```
//Pemilihan Libary
#include <Wire.h>
#include <WiFi.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_BME280.h>
#include <TinyGPS++.h>
#include <HardwareSerial.h>
#include <ESP_Google_Sheet_Client.h>
```

Gambar 4. 15 Pemilihan Library

Pada gambar 4.15 terdapat pemilihan library merupakan langkah penting untuk memastikan kompatibilitas dan fungsionalitas setiap komponen yang digunakan. Library `Wire.h` digunakan untuk mendukung komunikasi I2C antara ESP32 dengan perangkat lain, seperti sensor dan modul tampilan. `WiFi.h` berfungsi untuk mengatur koneksi ESP32 ke jaringan nirkabel, yang diperlukan untuk mengirim data ke layanan cloud seperti Google Spreadsheet.

Library `LiquidCrystal_I2C.h` digunakan untuk mengontrol tampilan LCD 16x2 yang menggunakan protokol I2C, sehingga mempermudah proses pengiriman data visual kepada pengguna. Selanjutnya, `Adafruit_BME280.h` merupakan library yang menyediakan fungsi untuk membaca data suhu, kelembaban, dan tekanan dari sensor BME280 secara akurat. Untuk memperoleh data lokasi, digunakan library `TinyGPS++.h`, yang memungkinkan ESP32 untuk memproses data dari modul GPS.

Selain itu, *library* `HardwareSerial.h` diperlukan untuk mendefinisikan port komunikasi serial tambahan yang digunakan untuk membaca data dari modul GPS melalui pin UART. Terakhir, `ESP_Google_Sheet_Client.h` adalah library yang memungkinkan integrasi antara ESP32 dan Google Spreadsheet melalui protokol Firebase JSON dan autentikasi menggunakan *private key*, sehingga memungkinkan penyimpanan data secara otomatis ke dalam lembar kerja secara daring.

```
//Deklarasi Objek dan Pin
LiquidCrystal_I2C lcd(0x27, 16, 2); // LCD
Adafruit_BME280 bme; // BME280
TinyGPSPlus gps; // GPS
HardwareSerial neogps(1); // Gunakan UART1
#define PIN_BUZZER  12 // Buzzer dan LED
#define PIN_LED     13 // LED
```

Gambar 4. 16 Deklarasi Objek dan Pin

Pada gambar 4.16 diatas adalah bagian yang dilakukan deklarasi objek dan penentuan pin yang akan digunakan dalam sistem. Objek `lcd` dideklarasikan dengan menggunakan library `LiquidCrystal_I2C`, yang berfungsi untuk menginisialisasi LCD 16x2 dengan alamat I2C `0x27`. Inisialisasi ini memungkinkan perangkat untuk menampilkan informasi dalam dua baris dengan masing-masing 16 karakter. Selanjutnya, objek `bme` dideklarasikan dari kelas `Adafruit_BME280`, yang digunakan untuk membaca parameter lingkungan seperti suhu, kelembaban, dan tekanan udara dari sensor BME280.

Untuk memperoleh data lokasi, digunakan objek `gps` dari kelas `TinyGPSPlus` yang berfungsi untuk memproses data yang diterima dari modul GPS. Komunikasi antara modul GPS dan ESP32 dilakukan melalui port UART1, yang didefinisikan melalui objek `neogps(1)` menggunakan library `HardwareSerial`.

Selain itu, dua pin juga didefinisikan untuk komponen output, yaitu `PIN_BUZZER` pada pin GPIO 12 dan `PIN_LED` pada pin GPIO 13. Pin ini masing-masing digunakan untuk mengaktifkan buzzer dan LED sebagai indikator terjadinya sambaran petir. Deklarasi ini bertujuan untuk memudahkan pengendalian perangkat keras melalui program yang dijalankan pada mikrokontroler ESP32.

```
//Variabel untuk WiFi dan Google Spreadsheet
char ssid[] = "Ramadhan__";
char pass[] = "00000000";
const char PRIVATE_KEY[] PROGMEM = R"KEY(
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCp5sQKOEK4JOBC
94Gq7TWQc7Q2OM5iTbtUvPCnwRtkMS2t2D822LjyWhUGKlZQOdpGgzTZ/WeMqX6h
/K0M5ihxRXh09HL6q7R6Zlj0mz1TLUvhEnWsavXfmzht1gtKynK5pv7DZHb2LZLH
73vaXhL26ZHd25FKx7OHXh/LtgJhPaGehW1iVXhnhbJBEgh2mCiQhdmU3ai6sHTO
bFskTpWMYTMldN6zHNoQ2Z9Wgo9sanEGDQmRQgnNCzZCgmN3CqK44XKhIm2ZQhMV
ZbBAA8GKeUF2s4OGyCQf7706ryNXpHOF/o6GlRecOeHB99raqhROW01XccJMgNcK
hkC6pJvbAgMBAAECggEAGsjKJQD45lt3OKIY61+umbwz2ncayifjnW3qfne/SrdE
spWsZDLG9/s4OfBDX9hkTx4bdIuHXzHP8yzeAceS8/B2F0dQ4lsJU7Z+Kums7Eea
y7uuQ2Uv/R//mwkrCeUKhba247TF2vS/oWIMDhaAj4kDjuHbcNo0tOV4xmjwzRq4
1LE7KcPmmM6SbNQMS+86HuA3S5Ni0TdCh6F4yksdOTkTUXIqUUZDtw/wdbJv1JyA
/Cwr1l98OM0h6rO4ifJ3asdjRkgv+9OkF8KO+BKmRj3xNyneG+BRcW6NW794Tm+u
/XhgNxi5MBq1tXVmXEsQA5gebxjB4Fn+q8hRssTJ2QKBgQDve+qTkIqOie5rueqg
BfZA9jkHHt0/hoYHwRhmWkjOgJTzDLuX7I0j9MnkdfbmcyBvA9E5zOSCAzwLhzCb
sJqv6slBzRmujF7UmCP5PiNOje5ZRoaDYqSVTaGtWvWDJyIU32WiklexcsxnPLLB
3h0vpVHZzUwE5MpNKAYitqdB9QKBgQC1nl7TPKJ1UxGCQhw0g3nFW1J1g819g+Zo
a9cWpnXA/eonConj0TF9CY+KGyxd2FnAgu+aFYGEtW7LhYB/cHcROgIjBtfA02Ui
JDAQIIlthX6bDRI/98Z9yd9IrtE0eQbqCnHDcDLJdedoj3vtjtasz31HyMYQTU+Z
577Twu00jwKBgGvMX57k5TmcJNIg8dfuVxaOvsTRw82ghBxhctY/tZ4eiu2MwbR1
KmMRAP/xiZ3aHLNcCIX4ELPlONYHmH28VSTyjn0sEW9ciCMyCJ9ctQ/bG3rcIBhs
AqGiPDCITRMoPlMe6+Tt8sNRZWjjDaPukzyrMdjUbG+XfOiWJ4zgoxExAoGBAIzZ
19ym569pzuhgcRNg661dV6P/pfwzsRtGlzbfve8AfPnuOGZQNLsuEP4geNYHFgQx
B7VN/HZ5ZwW78Z3kENNxU3rqsaS/zW0e949qTGp932RMTN0DQhMH3ny151V0jyIv
G29gsmJ/aAo1TP09i8vV1uk3EjROYO+RRPNh9yiNAoGAT8NseeNCjwECJ/SeObn0
qgiRl4S0qwOpfPtV8iCvPga/kz8G9bW5qwR104Z+cSkrucaFxnCpYMd7WN+xDpJZ
Q4jHsVCMl9SO6W93ilYviMcknISwE1RSqTi6mjoYO+j05lUJLVwfMNgy6P0uhdXd
U3OtkMTH17+Or5CKbykjf5A=
-----END PRIVATE KEY-----
```

Gambar 4. 17 Variabel Untuk Wi-Fi dan Google Spreadsheet

Pada gambar 4.17 dilakukan deklarasi variabel yang digunakan untuk koneksi jaringan Wi-Fi serta autentikasi ke layanan Google Spreadsheet. Variabel `ssid` dan `pass` masing-masing berisi nama SSID dan kata sandi dari jaringan WiFi yang akan digunakan oleh mikrokontroler ESP32 untuk mengakses internet. Koneksi ini diperlukan agar data hasil pemantauan dari sistem dapat dikirim dan disimpan secara otomatis ke dalam Google Spreadsheet.

Selanjutnya, variabel `PRIVATE_KEY` didefinisikan menggunakan perintah `PROGMEM` untuk menyimpan data secara permanen di memori program (*flash memory*) ESP32. Variabel ini berisi *private key* yang digunakan dalam proses

autentikasi OAuth 2.0, agar perangkat dapat terhubung secara aman ke layanan Google API. Format penyimpanan kunci pribadi tersebut menggunakan metode *raw string literal* (`R"KEY(...)"KEY"`), yang memungkinkan pemrosesan teks dalam bentuk multiline tanpa perlu escape karakter tambahan. Keberadaan private key ini bersifat krusial karena menjadi bagian dari kredensial digital yang menjamin otentikasi dan integritas dalam proses pengiriman data sensor ke spreadsheet secara otomatis.

```
//Struktur Setup Awal
void setup() {
  Serial.begin(115200);
  Wire.begin();

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

  if (!bme.begin(0x76)) {  // atau 0x77 tergantung modul
    Serial.println("Sensor BME280 tidak ditemukan!");
    while (1)
      ;
  }

  pinMode(BUZZER, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(pulsePin, INPUT);

  SerialGPS.begin(9600, SERIAL_8N1, TX_GPS, RX_GPS);  //GPS NEO 6M menggunakan baudrate 9600
  // Attach interrupt ke pin PULSA_OUT, trig saat RISING (pulsa naik)

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi Connected");
  lcd.setCursor(0, 0);
  lcd.print("WiFi Connected");
}
```

Gambar 4. 18 Struktur Setup Awal

Fungsi `setup()` merupakan bagian penting dalam program mikrokontroler yang dieksekusi satu kali saat sistem pertama kali dinyalakan atau di-*reset*. Pada

bagian awal fungsi ini, antarmuka serial diinisialisasi dengan kecepatan baudrate 115200 bps menggunakan `Serial.begin()`, yang berfungsi untuk komunikasi antara ESP32 dan komputer melalui USB. Kemudian, antarmuka I2C diaktifkan melalui `Wire.begin()`, untuk memungkinkan komunikasi dengan perangkat I2C seperti LCD dan sensor BME280.

Selanjutnya, LCD diinisialisasi dengan perintah `lcd.init()` dan lampu latar (*backlight*) diaktifkan agar tampilan dapat terbaca dengan baik. Inisialisasi sensor BME280 dilakukan menggunakan `bme.begin(0x76)`; apabila sensor tidak terdeteksi, maka sistem akan menampilkan pesan kesalahan pada antarmuka serial dan masuk ke dalam loop tak hingga, sebagai bentuk penanganan kesalahan (*error handling*).

Pin I/O kemudian dikonfigurasi, yaitu `BUZZER` dan `LED` sebagai keluaran (*output*), dan *`pulse`*`Pin` sebagai masukan (*input*). Modul GPS NEO-6M diinisialisasi melalui port UART1 dengan baudrate 9600 bps menggunakan `SerialGPS.begin()`. Setelah itu, ESP32 mencoba untuk terhubung ke jaringan WiFi dengan parameter SSID dan kata sandi yang telah dideklarasikan sebelumnya. Sistem akan menunggu hingga koneksi berhasil, ditandai dengan perubahan status `WiFi.status()` menjadi `WL_CONNECTED`. Setelah berhasil terhubung, sistem mencetak pesan “Wi-Fi Connected” baik melalui serial monitor maupun pada tampilan LCD sebagai konfirmasi keberhasilan koneksi jaringan.

Dengan demikian, proses inisialisasi pada fungsi `setup()` memastikan bahwa seluruh komponen perangkat keras, termasuk sensor, modul komunikasi, serta elemen output seperti LCD dan buzzer, telah siap beroperasi secara optimal sebelum sistem memasuki loop utama. Langkah-langkah ini menjadi fondasi penting dalam menjamin stabilitas dan keandalan sistem *Monitoring* deteksi sambaran petir berbasis ESP32.

4.2.2.3 Program untuk Koneksi ke Modul Sensor BME 280

Pada subbab ini membahasa proses perancangan program untuk menghubungkan mikrokontroler ESP32 dengan modul sensor BME280. Sensor BME280 digunakan untuk mengukur parameter lingkungan seperti suhu, kelembapan, dan tekanan udara secara *real-time*. Koneksi antara ESP32 dan sensor BME280 dilakukan melalui antarmuka komunikasi I2C, yang memungkinkan pertukaran data secara efisien dan stabil. Pemrograman pada bagian ini bertujuan untuk memastikan bahwa data yang diperoleh dari sensor dapat diakses dan diolah dengan akurat oleh sistem.

```
//Inisialisasi Sensor BME280
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
Adafruit_BME280 bme;
```

Gambar 4. 19 Inisialisasi Sensor BME 280

Pada gambar 4.19 dilakukan inisialisasi sensor BME280 dengan menggunakan pustaka *Adafruit_BME280* dan *Adafruit_Sensor* yang telah tersedia dalam Arduino IDE. Pustaka *Adafruit_BME280* digunakan untuk memudahkan proses komunikasi antara mikrokontroler ESP32 dengan sensor BME280 melalui antarmuka I2C. Baris kode `Adafruit_BME280 bme;` berfungsi untuk mendeklarasikan objek `bme` yang mewakili sensor, sehingga memungkinkan pemanggilan fungsi-fungsi pembacaan data seperti suhu, kelembapan, dan tekanan udara. Inisialisasi ini merupakan tahapan awal yang penting agar sensor dapat diakses dan digunakan dalam keseluruhan sistem *Monitoring*.

```
//Konfigurasi dalam Fungsi setup
if (!bme.begin(0x76)) {
  Serial.println("Sensor BME280 tidak ditemukan!");
  while (1); // Hentikan program jika sensor tidak terdeteksi
}
```

Gambar 4. 20 Konfigurasi dalam fungsi Setup Sensor BME 280

Pada gambar 4.20 tersebut merupakan bagian dari fungsi `setup()` yang berfungsi untuk melakukan konfigurasi awal sensor BME280 sebelum digunakan. Baris `if (!bme.begin(0x76))` digunakan untuk menginisialisasi sensor BME280 dengan alamat I2C `0x76`, yang merupakan alamat default dari sebagian besar modul BME280. Jika sensor tidak terdeteksi pada alamat tersebut, maka akan ditampilkan pesan "Sensor BME280 tidak ditemukan!" melalui komunikasi serial, dan program akan dihentikan secara permanen menggunakan `while (1);`. Langkah ini penting untuk memastikan bahwa sistem tidak melanjutkan proses selanjutnya apabila sensor utama tidak berfungsi, sehingga dapat mencegah kesalahan pembacaan data lingkungan dan menjaga keandalan sistem secara keseluruhan.

```
//Pembacaan Data Sensor BME280
void bacaSensor() {
  int suhu = bme.readTemperature();
  float tekanan = bme.readPressure() / 100.0F;
  int kelembaban = bme.readHumidity();

lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(suhuGlobal);
  lcd.print((char)223);  // Simbol derajat
  lcd.print("C H:");
  lcd.print(kelembabanGlobal);
  lcd.print("%");
```

Gambar 4. 21 Pembacaan Data Sensor BME 280

Pada gambar 4.21 tersebut menunjukkan proses pembacaan data lingkungan menggunakan sensor BME280 yang dilakukan dalam fungsi `bacaSensor()`. Sensor ini mengukur tiga parameter utama, yaitu suhu (`bme.readTemperature()`), tekanan udara (`bme.readPressure()`), dan kelembaban relatif (`bme.readHumidity()`). Nilai tekanan dikonversi ke satuan hPa dengan membagi hasil pembacaan dalam Pascal dengan 100.0. Selanjutnya, data suhu dan kelembaban ditampilkan secara *real-time* pada layar LCD 16x2. Format tampilan mencakup suhu dalam derajat Celsius (`°C`) dan kelembaban dalam persen (`%`).

Simbol derajat ditampilkan menggunakan kode karakter ASCII `(char)223`, yang merupakan karakter khusus untuk simbol tersebut pada LCD berbasis HD44780. Tampilan ini memberikan umpan balik visual langsung kepada pengguna mengenai kondisi lingkungan yang terpantau oleh sistem.

```
//Tampilan Data ke LCD dan Serial Monitor
lcd.setCursor(0, 1);
  if (lcdPage == 0) {
    lcd.print("Tek:");
    lcd.print(tekananGlobal, 0);
    lcd.print("hPa");

  Serial.print(" | T: ");
  Serial.print(suhu);
  Serial.print(" | H: ");
  Serial.print(kelembaban);
  Serial.print(" | P: ");
  Serial.print(tekanan);
```

Gambar 4. 22 Tampilan Data ke LCD dan Serial Monitor

Pada gambar 4.22 ini bertanggung jawab untuk menampilkan data hasil pembacaan sensor BME280 ke LCD dan Serial Monitor. Baris `lcd.setCursor(0, 1)` mengatur posisi kursor LCD pada baris kedua. Jika variabel `lcdPage` bernilai 0, maka sistem menampilkan nilai tekanan udara (`tekananGlobal`) dalam satuan hPa pada LCD. Nilai tersebut ditampilkan tanpa angka di belakang koma (`0` digit desimal) untuk menjaga keterbacaan di layar berukuran kecil.

Selain itu, data suhu, kelembaban, dan tekanan juga dikirimkan ke Serial Monitor melalui perintah `Serial.print`, yang berguna untuk debugging atau pemantauan secara langsung melalui perangkat komputer. Format tampilan pada Serial Monitor dirancang agar informatif dan mudah dibaca, dengan label seperti “T” untuk suhu, “H” untuk kelembaban, dan “P” untuk tekanan udara. Pendekatan ini memungkinkan verifikasi data secara paralel antara tampilan fisik (LCD) dan tampilan digital (Serial Monitor).

Dengan demikian, proses pembacaan dan penampilan data lingkungan seperti suhu, kelembaban, dan tekanan udara dari sensor BME280 dapat dilakukan secara *real-time* melalui tampilan LCD dan Serial Monitor. Hal ini memungkinkan sistem untuk memberikan informasi kondisi cuaca secara langsung dan memudahkan pengguna dalam melakukan pemantauan serta analisis data yang dibutuhkan.

4.2.2.4 Program untuk Koneksi ke Modul GPS

Pada subbab ini membahas perancangan program untuk menghubungkan mikrokontroler ESP32 dengan modul GPS, yang bertujuan untuk memperoleh data lokasi berupa koordinat lintang dan bujur secara *real-time*. Modul GPS yang digunakan dalam sistem ini adalah NEO-6M, yang terhubung melalui komunikasi serial dan akan digunakan sebagai referensi lokasi pada saat terjadinya sambaran petir.

```
//Inisialisasi Modul GPS
#include <TinyGPS++.h>
#include <HardwareSerial.h>

TinyGPSPlus gps;
HardwareSerial neogps(1); // UART1 (RX=GPIO9, TX=GPIO10 misalnya)
```

Gambar 4. 23 Inisialisasi Modul GPS

Pada gambar 4.23 tersebut Untuk menginisialisasi modul GPS, digunakan pustaka *TinyGPS++* yang berfungsi untuk memproses dan menguraikan data NMEA dari modul GPS. Selain itu, digunakan pustaka *HardwareSerial* untuk mengaktifkan port komunikasi serial tambahan pada ESP32. Objek `gps` dideklarasikan sebagai instance dari kelas `TinyGPSPlus` untuk menangani pengolahan data GPS. Sementara itu, `neogps` dideklarasikan sebagai objek dari `HardwareSerial` yang menggunakan UART1 sebagai saluran komunikasi dengan modul GPS NEO-6M. Pada konfigurasi ini, pin RX dan TX dipetakan ke GPIO tertentu sesuai kebutuhan perangkat keras. Pendekatan ini memungkinkan ESP32

untuk berkomunikasi secara paralel dengan modul GPS tanpa mengganggu komunikasi serial utama (*Serial Monitor*).

```
//Konfigurasi Pin
void setup() {
  neogps.begin(9600, SERIAL_8N1, 16, 17); // RX=GPIO16, TX=GPIO17
  Serial.begin(115200);
}
```

Gambar 4. 24 Konfigurasi Pin

Pada gambar 4.24 ini adalah bagian fungsi `setup()`, dilakukan konfigurasi awal komunikasi serial antara mikrokontroler ESP32 dan modul GPS NEO-6M. Objek `neogps` diinisialisasi menggunakan fungsi `begin()` dengan kecepatan baud rate sebesar 9600 bps, yang merupakan nilai standar komunikasi untuk modul GPS tersebut. Parameter `SERIAL_8N1` menunjukkan bahwa format komunikasi menggunakan 8 bit data, tanpa parity, dan 1 stop bit. Selain itu, pin GPIO16 dan GPIO17 digunakan sebagai jalur komunikasi serial RX dan TX masing-masing. Di samping itu, komunikasi serial utama (melalui port USB) diaktifkan dengan kecepatan 115200 bps menggunakan `Serial.begin()`, yang berfungsi untuk keperluan debugging dan pemantauan data melalui Serial Monitor. Konfigurasi ini memastikan bahwa ESP32 dapat secara efektif membaca dan menampilkan data dari modul GPS.

```
//Fungsi Pembacaan Data GPS
  dataGPS(latitude, longitude);  // Data GPS
  String koordinatGlobal = latGlobal + "," + lngGlobal; // Gabungan koordinat
```

Gambar 4. 25 Fungsi Pembacaan Data GPS

Pada gambar 4.25 ini adalah Bagian program ini menunjukkan pemanggilan fungsi `dataGPS(latitude, longitude)` yang bertujuan untuk mengambil informasi koordinat dari modul GPS, berupa nilai lintang (*latitude*) dan bujur

(*longitude*). Setelah data koordinat berhasil diperoleh, nilai lintang dan bujur yang telah disimpan ke dalam variabel `latGlobal` dan `lngGlobal` kemudian digabungkan ke dalam satu string menggunakan format teks "latitude,longitude".

Gabungan ini disimpan dalam variabel `koordinatGlobal`, yang dapat digunakan lebih lanjut, misalnya untuk ditampilkan pada antarmuka pengguna, dikirim ke layanan penyimpanan daring seperti Google Spreadsheet, atau digunakan sebagai data lokasi dalam sistem *Monitoring*. Pendekatan ini memudahkan penyajian dan pengolahan informasi lokasi dalam satu kesatuan format.

```
//Menampilkan Data GPS
void loop() {
  while (neogps.available() > 0) {
    gps.encode(neogps.read());
  }

  if (gps.location.isUpdated()) {
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();
    String waktuUTC = String(gps.time.hour()) + ":" +
                      String(gps.time.minute()) + ":" +
                      String(gps.time.second());

    Serial.print("Lat: ");
    Serial.println(latitude, 6);
    Serial.print("Lng: ");
    Serial.println(longitude, 6);
    Serial.print("Time (UTC): ");
    Serial.println(waktuUTC);

    // Menampilkan di LCD
    lcd.setCursor(0, 0);
    lcd.print("Lat:");
    lcd.print(latitude, 2);
    lcd.setCursor(0, 1);
    lcd.print("Lng:");
    lcd.print(longitude, 2);
  }
}
```

Gambar 4. 26 Menampilkan Data GPS

Pada gambar 4.26 adalah bagian yang berfungsi untuk membaca dan menampilkan data lokasi dari modul GPS ke Serial Monitor dan LCD. Fungsi `loop()` dimulai dengan memeriksa ketersediaan data dari modul GPS melalui objek `neogps`. Data yang diterima kemudian dienkode menggunakan fungsi `gps.encode()` dari pustaka *TinyGPS++* untuk diolah lebih lanjut. Apabila data lokasi terkini berhasil diperbarui, maka nilai lintang (*latitude*) dan bujur (*longitude*) disimpan dalam variabel `latitude` dan `longitude`. Selain itu, waktu dalam format UTC juga diambil dan disusun dalam bentuk string. Nilai-nilai tersebut kemudian ditampilkan melalui *Serial Monitor* untuk keperluan pemantauan, dan secara bersamaan ditampilkan pula pada LCD. Koordinat lintang ditampilkan pada baris pertama, sedangkan bujur ditampilkan pada baris kedua LCD, masing-masing dengan dua angka di belakang koma untuk meningkatkan keterbacaan. Bagian ini memastikan bahwa sistem mampu menyajikan data lokasi secara *real-time* secara visual dan digital.

Dengan demikian, program koneksi dan pembacaan data dari modul GPS telah berhasil diimplementasikan dengan baik, memungkinkan sistem untuk memperoleh serta menampilkan informasi lokasi secara *real-time* melalui Serial Monitor maupun layar LCD. Hal ini menjadi salah satu komponen penting dalam mendukung fungsi *Monitoring* dan pelacakan pada sistem deteksi sambaran petir secara akurat.

4.2.2.5 Program untuk Koneksi ke Tampilan LCD

Pada bagian ini dijelaskan mengenai proses pemrograman untuk menghubungkan mikrokontroler ESP32 dengan modul tampilan LCD 16x2 yang menggunakan protokol komunikasi I2C. Penggunaan tampilan LCD bertujuan untuk menyajikan informasi hasil pembacaan sensor, seperti suhu, kelembaban, tekanan udara, serta status sistem secara *real-time*. Dengan implementasi ini, pengguna dapat dengan mudah memantau kondisi lingkungan langsung melalui tampilan fisik tanpa memerlukan perangkat tambahan lainnya.

```
//Inisialisasi Libary dan Objek LCD
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // Alamat I2C, jumlah kolom dan baris
```

Gambar 4. 27 Inisialisasi LCD

Pada gambar 4.27 merupakan bagian ini dilakukan inisialisasi library dan objek yang diperlukan untuk mengoperasikan modul LCD 16x2 berbasis antarmuka I2C. Baris pertama memuat `#include <Wire.h>` yang berfungsi untuk mengaktifkan komunikasi I2C pada mikrokontroler ESP32. Selanjutnya, `#include <LiquidCrystal_I2C.h>` digunakan untuk memanggil pustaka khusus yang mendukung pengoperasian LCD melalui jalur I2C. Deklarasi objek `LiquidCrystal_I2C lcd(0x27, 16, 2);` menunjukkan bahwa LCD berada pada alamat I2C `0x27`, serta memiliki konfigurasi 16 kolom dan 2 baris. Inisialisasi ini merupakan langkah penting agar tampilan dapat difungsikan secara optimal dalam menampilkan data sensor.

```
// Fungsi Setup di  LCD
void setup() {
  lcd.init();           // Inisialisasi LCD
  lcd.backlight();      // Menyalakan backlight LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("M.Indra R");
  lcd.setCursor(0, 1);
  lcd.print("40040621650061");
  delay(2000);
}
```

Gambar 4. 28 Fungsi Setup di LCD

Pada gambar 4.28 ini merupakan bagian Fungsi `setup()` pada bagian ini digunakan untuk menginisialisasi dan menyiapkan LCD agar siap menampilkan informasi. Perintah `lcd.init()` berfungsi untuk menginisialisasi perangkat keras LCD dan memastikan komunikasi I2C telah berjalan dengan baik. Selanjutnya,

`lcd.backlight()` digunakan untuk menyalakan lampu latar LCD agar tampilan dapat terlihat dengan jelas. Fungsi `lcd.clear()` akan menghapus seluruh tampilan sebelumnya pada layar, memastikan bahwa data baru ditampilkan dari awal. Melalui perintah `lcd.setCursor(0, 0)` dan `lcd.setCursor(0, 1)`, kursor LCD diarahkan ke baris pertama dan kedua sebelum mencetak teks identitas berupa nama dan nomor induk mahasiswa. Penundaan selama dua detik dengan `delay(2000)` diberikan agar informasi tersebut dapat terbaca sebelum tampilan diperbarui. Tahapan ini merupakan bagian penting dalam proses inisialisasi visual sistem.

```
//Menampilkan Informasi ke LCD
void tampilkanData(float suhu, float kelembaban, float tekanan, float lat, float lng) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("S:");
  lcd.print(suhu, 1);
  lcd.print("C H:");
  lcd.print(kelembaban, 0);
  lcd.print("%");

  lcd.setCursor(0, 1);
  lcd.print("P:");
  lcd.print(tekanan, 0);
  lcd.print(" hPa");
  delay(3000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Lat:");
  lcd.print(lat, 2);
  lcd.setCursor(0, 1);
  lcd.print("Lng:");
  lcd.print(lng, 2);
  delay(3000);
}
```

Gambar 4. 29 Menampilkan Informasi ke LCD

Pada gambar 4.29 diatas merupakan Fungsi `tampilkanData()` bertugas untuk menampilkan informasi sensor dan koordinat GPS ke layar LCD secara bergantian. Parameter fungsi ini meliputi data suhu, kelembaban, tekanan udara, serta koordinat lintang (latitude) dan bujur (longitude). Tahap pertama dalam fungsi ini adalah membersihkan layar dengan `lcd.clear()` guna memastikan tidak ada tampilan sebelumnya yang tertinggal. Kemudian, data suhu dan kelembaban ditampilkan di baris pertama dengan format "S: suhu°C H: kelembaban%",

sedangkan data tekanan udara ditampilkan di baris kedua dalam satuan hPa. Setelah ditampilkan selama tiga detik menggunakan `delay(3000)`, layar kembali dibersihkan dan dilanjutkan dengan menampilkan data GPS. Baris pertama menampilkan nilai latitude dengan awalan "Lat:", sementara baris kedua menampilkan longitude dengan awalan "Lng:". Masing-masing ditampilkan dalam dua angka di belakang koma sebagai representasi akurasi posisi. Proses ini memungkinkan pengguna untuk memantau data lingkungan dan lokasi secara bergantian melalui tampilan LCD.

Dengan adanya implementasi program tampilan LCD, sistem mampu menampilkan informasi parameter lingkungan dan data lokasi secara *real-time* dengan format yang mudah dibaca. Hal ini memudahkan pengguna dalam memantau kondisi lingkungan serta posisi koordinat secara langsung melalui antarmuka LCD, sehingga mendukung efektivitas sistem dalam memberikan informasi secara cepat dan akurat.

4.2.2.6 Program untuk Koneksi ke Buzzer dan LED

Pada sistem pendeteksi sambaran petir ini, buzzer dan LED digunakan sebagai indikator peringatan dini terhadap potensi bahaya yang terdeteksi oleh sensor. Oleh karena itu, diperlukan perancangan program khusus yang mengatur konfigurasi pin, logika aktivasi, serta respon sistem terhadap sinyal yang dihasilkan oleh sensor. Subbab ini membahas proses pemrograman yang digunakan untuk menghubungkan dan mengendalikan buzzer serta LED sebagai komponen output visual dan auditori dari sistem.

```
//Inisialisasi Pin Buzzer dan LED
#define BUZZER_PIN 26
#define LED_PIN 25

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
}
```

Gambar 4. 30 Inisialisasi Pin dan Buzzer

Pada gambar 4.30 merupakan bagian yang dilakukan proses inisialisasi pin output untuk buzzer dan LED yang masing-masing terhubung pada pin GPIO 26 dan GPIO 25 pada mikrokontroler ESP32. Pendeklarasian `#define` digunakan untuk memberikan penamaan simbolik terhadap pin tersebut guna meningkatkan keterbacaan dan kemudahan pemeliharaan kode. Selanjutnya, fungsi `pinMode()` digunakan dalam blok `setup()` untuk mengatur kedua pin tersebut sebagai output, sehingga dapat memberikan sinyal tegangan ke buzzer dan LED sesuai kondisi logika yang diatur pada program utama. Inisialisasi ini merupakan langkah awal yang penting dalam memastikan bahwa perangkat output siap untuk diaktifkan ketika sistem mendeteksi adanya potensi sambaran petir.

```
//Fungsi Aktifkan Peringatan
void peringatanPetir() {
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(LED_PIN, HIGH);
  delay(1000); // buzzer dan LED menyala selama 1 detik
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
}
```

Gambar 4. 31 Fungsi Aktifkan Peringatan

Pada gambar 4.31 ini merupakan Fungsi `peringatanPetir()` bertujuan untuk mengaktifkan peringatan visual dan suara ketika terjadi deteksi sambaran petir. Fungsi ini memanfaatkan dua komponen output, yaitu buzzer dan LED. Ketika fungsi ini dipanggil, pertama-tama pin untuk buzzer dan LED diaktifkan dengan memberikan sinyal HIGH menggunakan fungsi `digitalWrite()`, yang mengindikasikan bahwa buzzer dan LED akan menyala. Selanjutnya, terdapat penundaan selama 1 detik menggunakan `delay(1000)` untuk memastikan buzzer dan LED menyala dalam jangka waktu yang cukup, sebelum keduanya dimatikan dengan mengirimkan sinyal LOW ke kedua pin tersebut. Proses ini memberikan notifikasi fisik kepada pengguna terkait potensi bahaya akibat sambaran petir yang terdeteksi oleh sistem.

Dengan demikian, program untuk koneksi dan pengendalian buzzer serta LED ini berperan penting dalam memberikan peringatan dini secara langsung kepada pengguna ketika sistem mendeteksi adanya potensi sambaran petir. Dengan respon visual dan auditori yang cepat dan jelas, sistem ini mampu meningkatkan kewaspadaan serta mendukung keselamatan di area pemantauan.