BAB IV
PEMBUATAN ALAT

4.1 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 ESP8266, sensor deteksi rangkaian Gelombang Amplitude Modulation, sensor suhu dan tekanan BME280, penentu koordinat lokasi alat modul GPS NEO6MV2. LCD 16 x 2 sebagai antarmuka visual disetiap alat. Pada sitstem supply alat menggunakan Baterai 18650 2000mAh, dengan dilengkapi modul LM2596 untuk pengaturan tegangan. Sebagai pegendali dan bertukar mengirim dan menerima data ESP8266 dirangakai dengan keseluruhan sistem, kemudian untuk *monitoring* platform ThingSpeak 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 Tabel 4 -1 dan Tabel 4 – 2 berisi bahan yang digunakan dalam perancangan alat.

Tabel 4 - 1 Alat

No	Nama Alat	Jumlah
1	Multimeter	1 Buah
2	Obeng Set	1 Set
3	Gunting	1 Buah
4	Tang Kupas	1 Buah
5	Tang Potong	1 Buah

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

Tabel 4 - 2 Bahan

No	Nama Bahan	Jumlah
1	ESP8266	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	1 Buah
9	Buckconverter LM2596	1 Buah
10	Buzzer 5V	1 Buah
11	Kabel Data USB to Micro B	2 Buat
12	Kapasitor 270pF	1 Buah
13	Kapasitorr 100nF	2 Buah
14	Kapasitor 100uF	2 Buah
15	Resistor 100KOhm	1 Buah
16	Resistor 1KOhm	3 Buah

No	Nama Bahan	Jumlah
17	Dioda 1N4001	1 Buah
18	Antena Telescopic	1 Buah
19	IC TA7642	1 Buah
20	BOX	2 Buah
21	Timah Solder	2 Rol

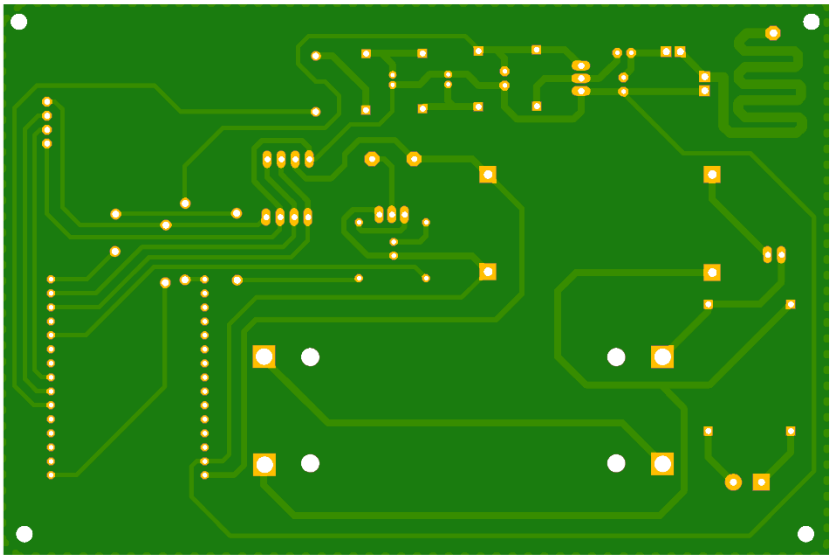
Keeluruhan alat dan bahan yang digunakan dipilih berdasarkan ketersediaan, keandalan, serta kesesuaian dengan kebutuhan sistem.

4.1.2 Pembuatan Perangkat Elektrik

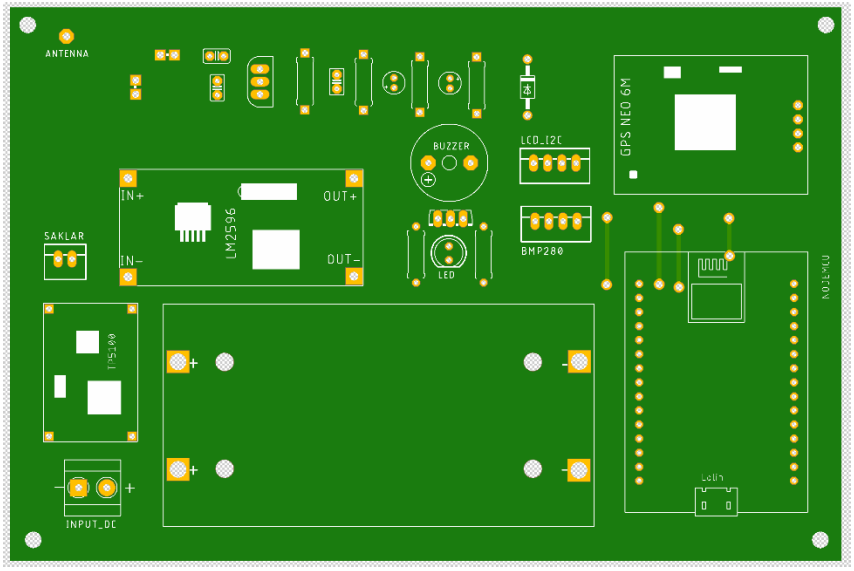
Pembuatan perangkat elektrik, dilakukan perancangan dan perakitan seluruh komponen elektronik yang bertugas mengendalikan sistem. Seluruh komponen elektronik ini dirangkai dalam satu sistem berbasis mikrokontroler ESP8266. Adapun langkah-langkah pembuatan perangkat elektrik adalah sebagai berikut.

1. Perancangan Skematik dan Layout PCB

Perancangan skematik dan layout PCB menggunakan *software* EasyEDA. Skematik ini terdapat ESP8266, Sensor BME280, Modul GPS NEO6, modul step-down LM2596, Baterai Management System (BMS) TP5100, Baterai 18650 2000mAh, dan rangakaian penerima sinyal Amplitude Modulation (AM) seperti pada gambar berikut. Pada proses ini komponen disusun dengan terstruktur, melakukan penempatan jalur, pengukuran papan pcb, pemberian lubang dan label konektor.



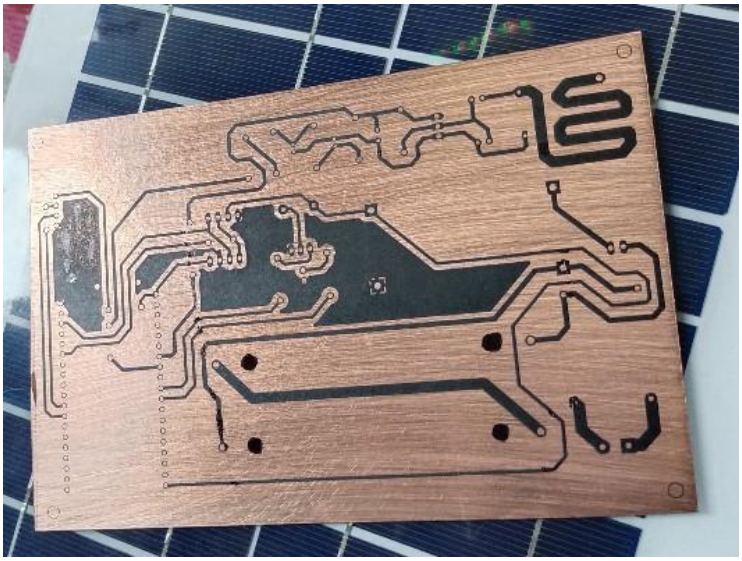
Gambar 4. 1 Layout PCB bawah



Gambar 4. 2 Layout PCB atas

2. Pencetakan PCB

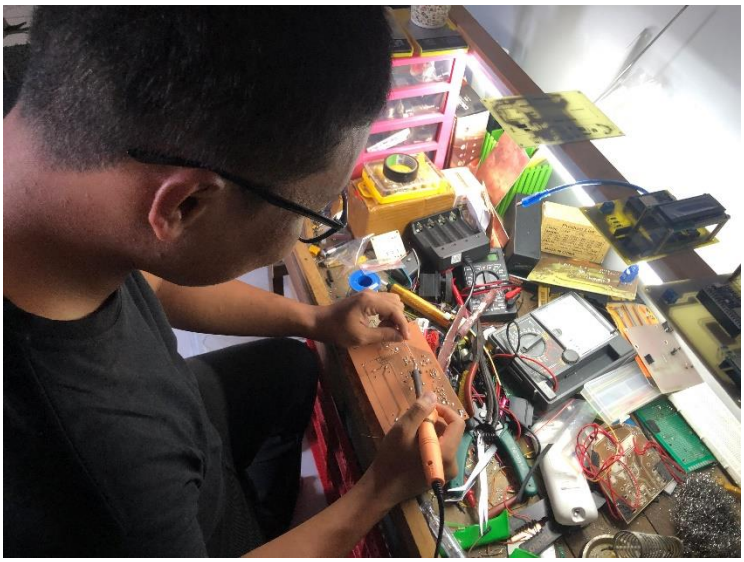
Proses pencetakan PCB dimulai dari cetak file desain menggunakan printer laser. Melakukan pemindahan jalur ke papan tembaga untuk mentup yang akan dijadikan jalur. Berikutnya proses *Etching* perendaman papan ke larutan *ferric chloride* (FeCL) untuk melarutkan tembaga yang tidak tertutup sablon. Setelan jalur terbentuk dengan rapi, dilakukan pembersihan papan dan pelungan papan pada bagian yang telah ditentukan.



Gambar 4. 3 Pencetakan PCB atas

3. Penyolderan Komponen ke PCB

Komponen yang didesain dipasang dan disolder pada papan PCB seperti ESP8266 dan dipasangkan pin header dan pin JST untuk menghubungkan komponen yang peletakannya tidak dalam PCB, hal tersebut bertujuan dapat memudahkan penyesuaian kebutuhan sistem.



Gambar 4. 4 Penyolderan PCB

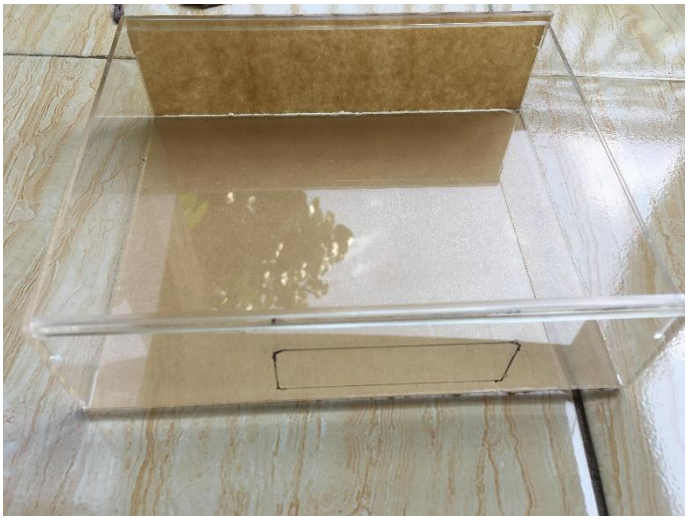
4.1.3 Pembuatan Perangkat Mekanik

Pembuatan perangkat mekanik pada sistem deteksi sambaran petir dilakukan untuk melindungi dan menopang seluruh rangkaian elektronik agar dapat beroperasi secara optimal di lingkungan luar ruangan. Tahapan ini melibatkan pemilihan serta desain mekanik dan perakitan wadah, pemasangan antena sebagai komponen penangkap sinyal elektromagnetik, serta penempatan sensor dan modul ESP8266 secara ergonomis agar terhindar dari gangguan fisik dan interferensi. Antena ditempatkan di bagian atas kotak agar memperoleh jangkauan sinyal maksimal.

1. Pemilihan Box dan Desain Mekanik

   Dalam proses perancangan perangkat deteksi sambaran petir ini, pemilihan komponen mekanik menjadi aspek penting guna memastikan perangkat memiliki ketahanan dan kemudahan dalam operasional di lapangan. Pemilihan Akrilik transparan digunakan dengan mempertimbangkan proses pemeliharaan dari komponen pada alat.

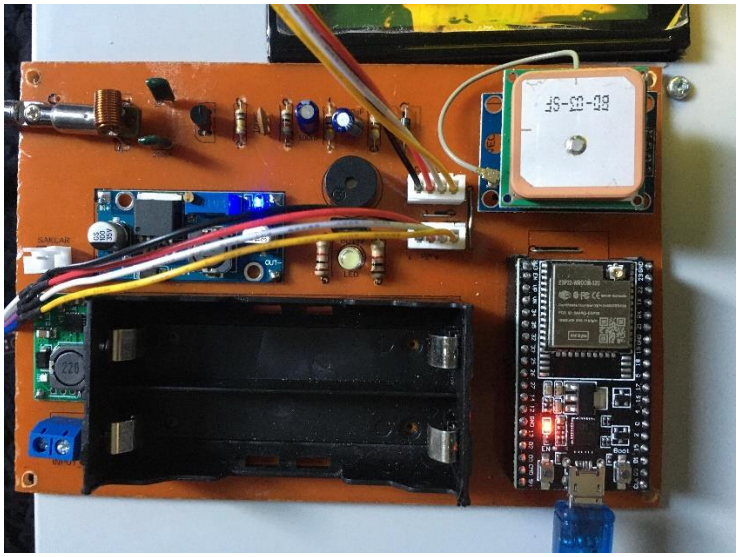
   Secara desain, bagian depan box dirancang untuk menempatkan LCD 16x2 yang berfungsi menampilkan data seperti suhu, tekanan, kelembapan, dan status deteksi petir. Tepat di sebelah kanan LCD, dipasang sensor BME280 yang ditonjolkan pada permukaan penutup box agar dapat langsung mengakses udara lingkungan sekitar. Di sisi kiri LCD, disematkan sebuah jack bulat sebagai port pengisian daya. Sementara itu, pada bagian samping kiri box, terdapat sebuah saklar utama untuk menyalakan atau mematikan perangkat secara manual. Adapun antena teleskopik sebagai bagian dari sistem penerima sinyal penerima sinyal amplitude modulation (AM), diposisikan pada **sisi** belakang bagian atas box**,** mengarah ke luar untuk mendapatkan penerimaan sinyal yang optimal. Desain ini dirancang dengan mempertimbangkan aspek estetika, fungsionalitas, serta kemudahan dalam perawatan dan pemasangan.



Gambar 4. 5 Pemilihan Kotak

2. Integrasi Sistem

Perakitan sensor atau komponen seperti Antena, BME280 tidak dalam PCB, tetapi dipasang menggunakan kabel JST. Bertujuan saat pembacaan sensor lebih baik dan memudahkan proses penggantian komponen saat terjadi trouble.

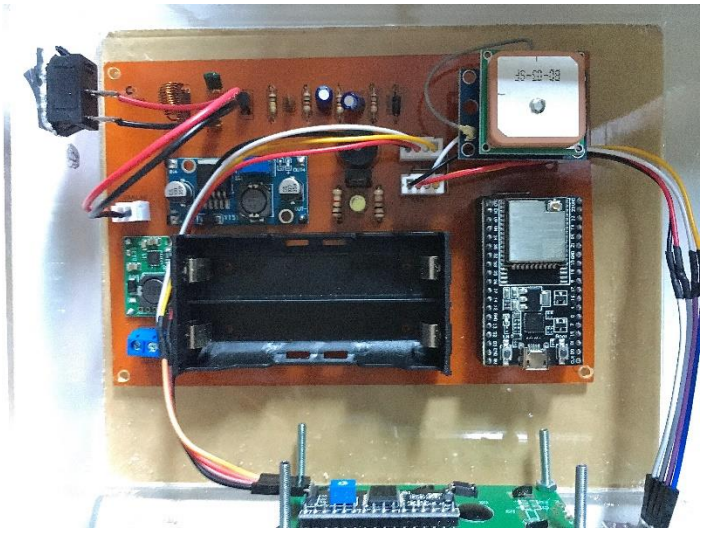


Gambar 4. 6 Integrasi Sistem

3. Perakitan

Tahapan perakitan perangkat dilakukan setelah seluruh komponen elektronik dan mekanik tersedia. Proses dimulai dengan pemasangan modul-modul utama seperti ESP8266, LCD 16x2, sensor BME280, modul GPS NEO-6M, serta rangkaian deteksi petir berbasis IC TA7642 ke dalam box akrilik menggunakan dudukan (*holder*) dan perekat khusus elektronik.

Setelah itu, LCD diposisikan dan dikencangkan pada bagian depan box sesuai lubang yang telah dipotong sesuai ukuran, diikuti oleh pemasangan sensor BME280 dan port charger. Saklar dipasang di sisi kiri, sedangkan antena teleskopik dirakit dan ditempatkan di sisi belakang menyesuaikan desain pada PCB dipasangkan dengan pengencang khusus agar tetap stabil saat digunakan.



Gambar 4. 7 Perakitan

4.2 Pembuatan Perangkat Lunak

Perangkat lunak dirancang untuk mengatur komunikasi antara modul ESP8266, sensor, serta *platform* pemantauan ThingSpeak. Pembuatan program dilakukan menggunakan bahasa pemrograman C++ pada lingkungan Arduino IDE. Dalam implementasinya, program diawali dengan inisialisasi koneksi WiFi agar ESP8266 dapat terhubung ke jaringan internet.

Selanjutnya, perangkat lunak mengatur pembacaan data dari sensor suhu, tekanan, serta deteksi sinyal petir dari rangkaian detektor. Data yang diperoleh akan dikemas dalam format yang sesuai dan dikirimkan secara berkala ke server ThingSpeak menggunakan metode HTTP POST. Selain itu, logika program juga mencakup mekanisme pembacaan waktu sambaran serta penanganan error apabila terjadi kegagalan koneksi internet. Dalam pengiriman data ke ThingSpeak, perangkat lunak menggunakan *API key* yang telah disesuaikan dengan channel yang ditentukan sebelumnya. Dengan demikian, sistem dapat melakukan monitoring melalui *dashboard* ThingSpeak.

4.2.1 Pembuatan Program Pada Arduino IDE

Pembuatan program dalam proyek ini dilakukan menggunakan platform Arduino IDE, yang berfungsi sebagai lingkungan pemrograman untuk mikrokontroler ESP8266. Program yang dibuat memiliki beberapa fungsi utama, yaitu membaca data dari rangkaian deteksi sambaran petir, sensor suhu dan tekanan, serta mengirimkan data tersebut ke platform pemantauan berbasis cloud, yakni ThingSpeak.

4.2.2.1 Pembuatan Program Koneksi ESP8266 dengan Wifi dan ThingSpeak

Tahap awal konektivitas antara ESP8266 dengan Wifi dan Platform ThingSpeak adalah memanggil pustaka yang dibutuhkan. Tahap ini merupakan penghubung antara keseluruhan daripada sitem alat yaitu *Hardware* dan Server. Pemanggilan pustaka ini bertujuan untuk menyediakan seluruh fungsi komunikasi internet dan protokol HTTP yang dibutuhkan dalam sistem. Tahap ini dilakukan inisialisasi *library*.

WiFi.h adalah Library bawaan ESP8266 untuk mengakses jaringan WiFi (mode Station atau Access Point). Dan ThingSpeak.h Digunakan untuk koneksi dan pengiriman data ke server ThingSpeak.

```
#include <ESP8266WiFi.h>
#include <LiquidCrystal_I2C.h>
```

Gambar 4. 8 Inisialisasi library untuk Wifi dan ThingSpeak

Langkah selanjutnya adalah melakukan konfigurasi parameter jaringan dan platform ThingSpeak. Variabel `ssid` dan `pass` menyimpan nama dan kata sandi jaringan WiFi yang digunakan oleh ESP8266 untuk terhubung ke internet. Sementara itu, `myChannelNumber` dan `myWriteAPIKey` digunakan untuk mengarahkan data yang dikirim ke channel yang benar di server ThingSpeak. Kedua parameter ini bersifat krusial untuk verifikasi akses sistem terhadap akun cloud pengguna.

```
// Konfigurasi Thingspeak & WiFi
const char *ssid = "HGU-KOS-FASINDO";
const char *pass = "F4S1ND00";

long myChannelNumber = 2991071;
const char myWriteAPIKey[] = "IM7Y5ZSV99M0AM4T";
```

Gambar 4. 9 Konfigurasi Wifi dan ThingSpeak

Setelah konfigurasi, program membuat objek WiFiClient sebagai jalur komunikasi data TCP/IP antara ESP8266 dan server ThingSpeak. Objek ini kemudian digunakan sebagai parameter dalam inisialisasi ThingSpeak melalui fungsi ThingSpeak.begin(client), dibuat hanya satu kali sebagai *instance* dasar.

```
ThingSpeak.begin(client);
```

Gambar 4. 10 Membuat Objek WiFiClient

Untuk memastikan konektivitas, dibuat satu fungsi khusus bernama cekKoneksiWiFi(). Fungsi ini bertanggung jawab untuk menjalankan perintah WiFi.begin(ssid, pass) yang memulai proses koneksi ke jaringan WiFi yang telah dikonfigurasi sebelumnya. Selanjutnya, koneksi WiFi dipantau dengan memeriksa

status koneksi menggunakan perintah WiFi.status(). Jika koneksi tidak berhasil dalam waktu lebih dari 30 detik, maka sistem secara otomatis akan me-*restart* mikrokontroler dengan perintah ESP.restart(). Hal ini merupakan bentuk *self-recovery* atau pemulihan mandiri agar sistem tetap andal ketika dioperasikan di lapangan.

```
Serial.println("Menyambungkan ke WiFi...");
WiFi.begin(ssid, pass);


while (WiFi.status() != WL_CONNECTED) {
```

Gambar 4. 11 Fungsi khusus Wifi

Setelah berhasil terhubung ke jaringan WiFi dan ThingSpeak, sistem selanjutnya dirancang untuk mengirimkan data ke cloud secara periodik. Hal ini dilakukan dengan menggunakan metode ThingSpeak.setField(channel, fieldNumber, value, apiKey), di mana data sensor seperti jumlah sambaran petir, suhu, tekanan udara, serta koordinat GPS dikirimkan satu per satu ke masing-masing field pada channel yang tersedia.

```
void thingspeakData() {
  ThingSpeak.setField(1, pulseCount);
  ThingSpeak.setField(2, dataLed);
  ThingSpeak.setField(3, suhuGlobal);
  ThingSpeak.setField(4, tekananGlobal);
  ThingSpeak.setField(5, latGlobal);
  ThingSpeak.setField(6, lngGlobal);
  ThingSpeak.setField(7, kelembabanGlobal);
  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
```

Gambar 4. 12 Pengiriman Data ke ThingSpeak

Setelah perangkat ESP8266 berhasil terhubung dengan jaringan WiFi dan platform ThingSpeak, pengguna dapat mengakses data yang dikirim. Data dapat diakses oleh pengguna sudah melaksanakan tahapan penggunaan website ThingSpeak. Tahapan tersebut dimana pengguna sudah membuat akun dan

membuat sebuah kanal channel baru, kemudian mengatur nama kanal, deskripsi, dan mengaktifkan hingga delapan *field* sesuai kebutuhan. Masing-masing *field* pada kanal ini berfungsi untuk menampilkan jenis data yang berbeda, seperti jumlah sambaran petir, suhu udara, tekanan atmosfer, kelembaban udara, serta informasi lokasi berupa lintang (latitude) dan bujur (longitude) yang dikirim dari modul GPS.

Apabila sudah dikonfigurasi sistem ESP8266 akan secara berkala mengirimkan data sensor ke kanal tersebut melalui protokol HTTP menggunakan API key yang telah diberikan oleh ThingSpeak kemudian disimpan pada program Arduin IDE, maka data yang diterima akan langsung ditampilkan dalam bentuk grafik pada setiap *field*, sehingga memudahkan pengguna dalam memantau perubahan lingkungan secara visual.

4.2.2.2 Pembuatan Program Rangkaian Skematik

Sistem deteksi petir ini bekerja dengan prinsip membaca gangguan elektromagnetik yang ditangkap oleh rangkaian berbasis modul deteksi sinyal AM, yang terdiri dari antena, amplifier, dan IC TA7642 sebagai detektor utama. Saat sistem pertama kali dinyalakan, ESP8266 akan melakukan inisialisasi pin input dan output, termasuk pin yang digunakan untuk membaca sinyal analog dari rangkaian detektor, serta pin PWM yang akan mengirimkan sinyal pemicu ke basis transistor. Rangkaian IC TA7642 diberi catu daya melalui regulator 3V atau langsung dari pin 3.3V ESP8266.

```
#define PWM_pin D7
#define AnalogIn A0
```

Gambar 4. 13 Definisi serta Inisialisasi pin Analog dan PWM

Dalam fungsi bacaSensor(), sinyal dari pin analog dibaca menggunakan analogRead(PIN_ANALOG) dan hasilnya disimpan sebagai nilai dataPulse. Tahapan pertama dalam proses ini dimulai dari pemberian sinyal PWM (Pulse Width Modulation) oleh ESP8266 pada pin keluaran yang terhubung ke rangkaian deteksi berbasis IC TA7642. Pin tersebut ditentukan sebagai `PWM_PIN` dengan nilai

logika `analogWrite(PWM_PIN, PWM_DutyCycle)`. ESP8266 mengirimkan sinyal PWM.

```
analogWrite(PWM_pin, PWM_DutyCycle);

int PWM_DutyCycle = 500;
```

Gambar 4. 14 Pengiriman sinyal PWM

Nilai Setelah sinyal analog dari TA7642 diterima oleh pin A0 pada ESP8266, nilai tersebut disimpan dalam variabel ActValue. Selanjutnya, sistem membandingkan nilai ini dengan pembacaan sebelumnya yang disimpan pada PrevValue, kemudian menghitung selisih mutlaknya melalui perintah `Delta = abs(ActValue - PrevValue)`. Nilai selisih ini disebut Delta dan berperan penting dalam menentukan adanya gangguan mendadak pada sinyal. Jika nilai Delta melebihi ambang batas yang telah ditentukan misalnya lima satuan ADC, maka dianggap terjadi lonjakan signifikan yang diasosiasikan dengan sambaran petir. Pendekatan ini dipilih karena sambaran petir menimbulkan impuls elektromagnetik singkat yang menyebabkan perubahan tegangan secara tiba-tiba. Dengan memantau perubahan antar-pengambilan sampel, sistem dapat membedakan antara noise biasa dan peristiwa petir..

```
void deteksiPetir() {
  unsigned long now = millis();
  ActValue = analogRead(AnalogIn);
  Delta = abs(ActValue - PrevValue);
  PrevValue = ActValue;

  // Trigger petir
  if (Delta >= deltaThreshold && !isStriking && (now - lastPulseTime > cooldownDuration)) {
    digitalWrite(LED_strike, HIGH);
    digitalWrite(Buzzer, HIGH);
    isStriking = true;
    lastTriggerTime = now;
    lastPulseTime = now;
    pulse++;
    dataLed = 1;

    tampilDeteksi = true;
    waktuDeteksiLCD = millis();
  }

  // Matikan LED dan buzzer setelah LED_DURATION
  if (isStriking && (now - lastTriggerTime >= LED_DURATION)) {
    digitalWrite(LED_strike, LOW);
    digitalWrite(Buzzer, LOW);
    isStriking = false;
  }
}
```

Gambar 4. 15 Fungsi deteksi nilai petir

Rumus ini bertujuan untuk memberikan bobot lebih besar terhadap data-data sebelumnya agar perubahan mendadak dapat lebih mudah dikenali sebagai anomali. Sambaran petir menyebabkan perubahan mendadak pada sinyal analog (Δ). Namun, noise kecil juga bisa memengaruhi nilai. Ambang ini (5 satuan ADC) digunakan agar sistem tidak mendeteksi noise sebagai petir.

```
const int deltaThreshold = 5;
```

Gambar 4. 16 Minimum Treshold

Setelah dinyalakan, sistem tidak langsung mendeteksi. Diberikan jeda 2 detik (variabel durasiWarmup) agar semua komponen stabil terlebih dahulu. Ini mencegah deteksi palsu akibat lonjakan awal yang biasa terjadi saat boot. Variabel dataLed merepresentasikan status indikator yang kemudian dikirim ke platform ThingSpeak, sedangkan isStriking menjadi penanda kondisi deteksi petir saat ini.

Waktu penyalaan indikator dan jeda antar deteksi dicatat oleh lastTriggerTime dan lastPulseTime, sedangkan PreviousMillis digunakan untuk mengatur pencatatan data serial agar tetap efisien. Dengan pengaturan ini, sistem mampu mendeteksi sambaran petir secara akurat tanpa dipengaruhi oleh noise atau impuls berulang.

```
const unsigned long cooldownDuration = 200;  // Jeda antar trigger petir (ms)
int dataLed;

bool isStriking = false;
unsigned long lastTriggerTime = 0;
unsigned long lastPulseTime = 0;
unsigned long PreviousMillis;
```

Gambar 4. 17 Jeda pembacaan.

4.2.2.3 Pembuatan Program Modul Sensor BME280

Sensor BME280 adalah sensor multifungsi berbasis I2C yang digunakan untuk mengukur tiga parameter utama lingkungan, yaitu suhu (temperature), kelembaban relatif (humidity), dan tekanan udara (pressure). Dalam program ini, sensor BME280 diakses melalui library Adafruit BME280, dan proses kerjanya dibagi dalam beberapa tahap dengan Insialisasi Library terlebih dahulu kemudian Inisialisasi Objek.

```
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

Gambar 4. 18 Insialisasi library untuk BME280

```
Adafruit_BME280 bme;  // gunakan I2C
```

Gambar 4. 19 Insialisasi objek untuk BME280

Selanjutnya adalah proses definisi pin sensor pada sensor BME280 inisialisasi sensor dilakukan pada fungsi setup() dengan pemanggilan bme.begin(0x76), yang menyatakan bahwa sensor terhubung melalui alamat I2C

0x76. Apabila sensor tidak terdeteksi pada alamat ini, sistem akan menghentikan operasinya dan menampilkan pesan kesalahan pada Serial Monitor.

```
if (!bme.begin(0x76)) {  // atau 0x77 tergantung modul
  Serial.println("Sensor BMP280 tidak ditemukan!");
  while (1)
    ;
}
```

Gambar 4. 20 Insialisasi pada Fungsi BME280

Kemudian memasuki proses pembacaan sensor BME280. Pada fungsi bacaSensor(), dilakukan proses pembacaan tiga parameter utama dari BME280. Terbagi tiga pembacaan yaitu Suhu, Tekanan Udara dan Kelembaban. Data suhu diperoleh dari fungsi `bme.readTemperature()` yang menghasilkan nilai dalam satuan derajat Celcius, sedangkan tekanan dibaca melalui fungsi `bme.readPressure()` kemudian dikonversi dari Pascal menjadi hektopascal dengan membaginya seratus. Untuk kelembaban, pembacaan dilakukan dengan fungsi `bme.readHumidity()` yang memberikan nilai persentase relatif (%).

```
// Baca BME280 hanya setiap 1 detik
if (now - lastBMERead >= intervalBME) {
  suhu = bme.readTemperature();
  tekanan = bme.readPressure() / 100.0F;
  kelembaban = bme.readHumidity();
  lastBMERead = now;
}
```

Gambar 4. 21 Fungsi pembacaan BME280

Agar pembacaan sensor lebih efisien dan tidak membebani mikrokontroler, digunakan mekanisme pengendalian interval melalui variabel lastBMERead dan intervalBME. Variabel lastBMERead menyimpan waktu terakhir pembacaan dilakukan, sementara intervalBME yang bernilai 1000 milidetik menentukan jeda

antar pembacaan. Proses ini memastikan sensor hanya dibaca setiap satu detik, sehingga konsumsi daya dan beban pemrosesan dapat diminimalkan.

```
unsigned long lastBMERead = 0;
const unsigned long intervalBME = 1000;
```

Gambar 4. 22 Pengendaian interval

Nilai ini kemudian disimpan ke variabel global suhuGlobal agar dapat diakses oleh fungsi lain seperti tampilanLCD() dan thingspeakData().

```
int suhuGlobal = 0;
int kelembabanGlobal = 0;
int tekananGlobal = 0;
```

Gambar 4. 23 Menyimpan data BME280 ke Global

4.2.2.4 Pembuatan Program Modul GPS

Modul GPS yang digunakan dalam sistem ini merupakan modul NEO-6M, yang bekerja dengan protokol komunikasi serial UART. Oleh karena itu, tahap pertama adalah membuat konfigurasi jalur komunikasi UART kedua (UART2) pada ESP8266, penggunaan lebih dari satu UART memungkinkan karena ESP8266 memiliki kemampuan multi serial.

Komunikasi antara GPS dan mikrokontroler ESP8266 dilakukan melalui antarmuka serial menggunakan library `SoftwareSerial`, karena ESP8266 hanya memiliki satu port UART hardware yang sudah digunakan untuk komunikasi dengan komputer.

```
#include <NMEAGPS.h>
#include <SoftwareSerial.h>
```

Gambar 4. 24 Insialisasi library GPS

Selanjutnya tahap deklarasi pin pin untuk komunikasi serial GPS. Pin RX_GPS (D5) jalur masuk data GPS ke ESP8266 dan pin TX_GPS (D6) → jalur keluar data ke GPS. Library ini mampu memparse kalimat NMEA format data standar GPS seperti $GPRMC, $GPGGA. gps_fix fix dignakan untuk menyimpan hasil parsing data GPS yang berisi data seperti fix.latitude(), fix.longitude(). Validitas data lokasi.

```
#define RX_GPS D5  // GPIO12 ← TX GPS
#define TX_GPS D6  // GPIO14 → RX GPS

SoftwareSerial gpsPort(RX_GPS, TX_GPS);  // RX, TX
NMEAGPS gps;
gps_fix fix;
```

Gambar 4. 25 Deklarasi dan insialisasi objek UART

Pada fungsi `setup()`, komunikasi serial dengan GPS diinisialisasi dengan perintah `Serial.begin(9600` Baudrate sebesar 9600 bps digunakan sesuai dengan standar modul NEO-6M. Dengan konfigurasi ini, sistem siap menerima data berupa string NMEA dari modul GPS yang berisi informasi lokasi, kecepatan, waktu UTC, dan parameter navigasi lainnya.

```
void setup() {
  Serial.begin(9600);

  gpsPort.begin(9600);  // GPS Baudrate
```

Gambar 4. 26 Insialisasi Serial GPS

Pemrosesan data GPS dilakukan melalui NMEAGPS. Melalui library ini merupakan tahapan membaca stream data NMEA dari gpsPort. Selajutya mem-parse data menjadi struktur yang mudah digunakan (latitude, longitude, validitas).

Fungsi bacaGPS() bertugas mengambil dan memperbarui data koordinat dari modul GPS secara terstruktur. Proses dimulai dengan pemeriksaan ketersediaan data melalui gps.available(gpsPort), yang menunjukkan bahwa modul GPS telah

mengirimkan kalimat NMEA melalui jalur serial. Setiap kali data tersedia, fungsi gps.read() dipanggil untuk mem-parsing kalimat NMEA tersebut menjadi objek gps_fix yang lebih mudah digunakan. Validitas lokasi diuji melalui fix.valid.location guna memastikan bahwa koordinat yang diterima berasal dari perhitungan posisi yang sah (GPS sudah memperoleh *fix*). Apabila data valid, nilai latitude dan longitude diambil dari objek fix, dikonversi ke format string dengan presisi enam digit desimal, lalu dibandingkan dengan data sebelumnya (prevLatitude, prevLongitude). Pembaruan variabel global latitude dan longitude hanya dilakukan bila terjadi perubahan posisi, sehingga sistem lebih efisien dan tidak menampilkan atau mengirim data yang sama berulang kali. Nilai koordinat terbaru ini kemudian dimanfaatkan oleh fungsi lain untuk tampilan LCD, pencatatan dan pengiriman ke ThingSpeak.

```
void bacaGPS() {
  while (gps.available(gpsPort)) {
    fix = gps.read();
    if (fix.valid.location) {
      String newLat = String(fix.latitude(), 6);
      String newLon = String(fix.longitude(), 6);

      if (newLat != prevLatitude || newLon != prevLongitude) {
        latitude = newLat;
        longitude = newLon;
        prevLatitude = newLat;
        prevLongitude = newLon;
      }
    }
  }
}
```

Gambar 4. 27 Fungsi pembacaan GPS

Data lokasi ini kemudian disimpan ke dalam variabel global yang selanjutnya dapat ditampilkan di LCD ataupun dikirim ke platform ThingSpeak bersama parameter sensor lainnya.

```
String latitude = "-";
String longitude = "-";
String prevLatitude = "-";
String prevLongitude = "-";
String gps_location;
```

Gambar 4. 28 Menyimpan Data GPS ke Global

4.2.2.5 Pembuatan Program Tampilan LCD

Tampilan data pada sistem ini dilakukan menggunakan modul LCD 16x2 berbasis komunikasi I2C, yang memiliki keunggulan dalam penggunaan jumlah pin yang lebih sedikit dibanding antarmuka paralel. Tahap pertama dalam pembuatan program LCD adalah pemanggilan pustaka LiquidCrystal_I2C.h, yang memungkinkan antarmuka I2C dapat mengendalikan tampilan teks pada layar LCD. Yang selanjutnya masuk tahap deklarasi inisialisasi objek I2C.

```
#include <LiquidCrystal_I2C.h>
```

Gambar 4. 29 Insialisasi library LCD I2C

```
LiquidCrystal_I2C lcd(0x27, 16, 2);  //LCD address = 0x27
```

Gambar 4. 30 Deklarasi dan insialisasi objek LCD I2C

Pada fungsi `setup()`, LCD diaktifkan dengan perintah `lcd.begin()` dan `lcd.backlight()` untuk menyalakan pencahayaan belakang. Dimana pada proses tersebut setelah *backlight* menyala dilanjutkan saat mulai proses koneksi Wifi tampilan “Menyambungkan….” dan “Wifi” setelah itu proses penyambungan koneksi Wifi tampil “Menyambung ke Wifi….”. Apabila koneksi berhasil terdapat *delay* selama 1 detik.

```
lcd.begin();      // Inisialisasi LCD
lcd.backlight();  // Nyalakan backlight
lcd.clear();

lcd.setCursor(0, 0);
lcd.print("Menghubungkan...");
lcd.setCursor(0, 1);
lcd.print("WiFi");
```

Gambar 4. 31 Fungsi awal LCD

Terdapat animasi LCD pada baris kedua, dengan animasi titk berjalan dengan program seperti pada gambar berikut.

```
// Animasi titik berjalan di baris kedua LCD
lcd.setCursor(0, 1);
lcd.print(loadingDots[dotIndex]);
dotIndex++;
if (dotIndex >= 4) dotIndex = 0;
```

Gambar 4. 32 Animasi LCD

Tahap selanjutnya hasil daripada koneksi Wifi, yaitu tampilan apabila waktu koneksi habis atau Timeout Koneksi. Jika proses lebih dari 30 detik diasumsikan proses koneksi gagal dengan dengan program pada gambar berikut.

```
    // 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. 33 Hasil koneksi LCD

Jika koneksi gagal tahapanya dimulai dari "\nGagal konek WiFi. Restart ESP..." apabila sudah valid proses koneksi gagal, maka "Gagal Konek WiFi" dan selanjutnya proses restart ESP8266 secara otomatis. Sedangkan apabila koneksi berhasil akan tampil "\nWiFi Terhubung!" dan IP Address: ". Setelah proses koneksi selesai, dilanjutkan proses fungsi daripada LCD untuk parameter parameter yang dibutuhkan.

Proses tampilan utama dalam fungsi `tampilanLCD()`, yang dirancang untuk menampilkan data suhu dan jumlah pulsa deteksi petir pada baris pertama. Sedangkan pada baris kedua, ditampilkan data tambahan secara bergantian berupa tekanan udara, kelembaban, latitude, dan longitude. Peralihan tampilan dilakukan dengan bantuan variabel `lcdPage` yang akan berganti setiap pemanggilan fungsi, sehingga seluruh data tetap bisa disajikan secara periodik meskipun layar terbatas.

```
// Baris 1: suhu dan kelembaban
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(suhu);
lcd.print((char)223);
lcd.print("C H:");
lcd.print(kelembaban);
lcd.print("%");

// Baris 2: rotasi info
lcd.setCursor(0, 1);
if (lcdPage == 0) {
  lcd.print("Tek:");
  lcd.print(tekanan);
  lcd.print("hPa ");
} else if (lcdPage == 1) {
  lcd.print("Lat:");
  lcd.print(latitude.substring(0, 8));
} else if (lcdPage == 2) {
  lcd.print("Lng:");
  lcd.print(longitude.substring(0, 8));
}

lcdPage++;
if (lcdPage > 2) lcdPage = 0;
}
```

Gambar 4. 34 Fungsi tampilan LCD

Untuk menampilkan deteksi petir, diatur hanya dapat tampil selama satu detik menggunakan logika waktu, kemudian kembali ke tampilan data sensor normal. Pendekatan ini memberikan umpan balik langsung kepada pengguna bahwa aktivitas elektromagnetik telah terdeteksi.

```
lcd.clear();

// Jika baru saja deteksi petir, tampilkan 2 detik pesan khusus
if (tampilDeteksi) {
  if (millis() - waktuDeteksiLCD <= 2000) {
    lcd.setCursor(0, 0);
    lcd.print("!PULSE DETECTED!");
    lcd.setCursor(0, 1);
    lcd.print("!DETEKSI PETIR!");
    return;
  } else {
    tampilDeteksi = false;
  }
}
```

Gambar 4. 35 Fungsi tampilan LCD Deteksi Petir

Pada sistem menggunakan juga penjadawalan tampilan bergantian. Fungsi millis() untuk memperbarui LCD setiap dua detik. Ini jauh lebih efisien dibandingkan delay() karena tidak menghentikan proses utama sistem.

```
static unsigned long lastLCD = 0;
if (millis() - lastLCD >= 2000) {
  lastLCD = millis();
  tampilanLCD();
}
```

Gambar 4. 36 Penjadwalan tampilan LCD

Dengan pendekatan ini, sistem mampu menyediakan informasi cepat dan ringkas secara lokal tanpa harus bergantung pada koneksi internet.

4.2.2.6 Pembuatan Program Buzzer

Tahapan awal dalam perancangan program buzzer adalah mendeklarasikan pin digital yang digunakan, yaitu D0, yang didefinisikan dengan `#define BUZZER D0`.

```
#define Buzzer D0
```

Gambar 4. 37 Deklarasi Pin Buzzer

Selanjutnya, pada bagian fungsi setup(), pin tersebut diatur sebagai output menggunakan perintah pinMode(BUZZER, OUTPUT), dan diatur dalam kondisi awal LOW atau tidak menyala. Hal ini dilakukan untuk memastikan buzzer tidak aktif secara tidak sengaja ketika sistem pertama kali dinyalakan atau saat melakukan reset.

```
pinMode(BUZZER, OUTPUT);
pinMode(LED, OUTPUT);
```

Gambar 4. 38 Insialisasi Buzzer

Dalam fungsi setup pin buzzer diatur ke kondisi logika rendah (LOW). Ini bertujuan untuk memastikan bahwa saat sistem baru dinyalakan, buzzer tidak akan berbunyi atau secara default dalam keadaan nonaktif.

```
digitalWrite(LED, LOW);
digitalWrite(BUZZER, LOW);
```

Gambar 4. 39 Fungsi setup Buzzer

Pada fungsi deteksi petir, buzzer diaktifkan ketika sistem mendeteksi adanya lonjakan sinyal dari pembacaan analog sensor TA7642 yang melampaui ambang batas. Nilai selisih antar-sampel disimpan dalam variabel `Delta`, dan jika melebihi ambang yang ditentukan (`deltaThreshold`), sistem menganggap terjadi impuls terkait aktivitas petir. Selama tidak berada dalam masa jeda (*cooldown*) dan indikator belum aktif, LED dan buzzer akan dinyalakan secara bersamaan, variabel `pulse` dinaikkan sebagai penghitung kejadian, dan variabel `dataLed` diset sebagai penanda status untuk pengiriman data ke ThingSpeak.

```
// Trigger petir
if (Delta >= deltaThreshold && !isStriking && (now - lastPulseTime > cooldownDuration)) {
  digitalWrite(LED_strike, HIGH);
  digitalWrite(Buzzer, HIGH);
  isStriking = true;
  lastTriggerTime = now;
  lastPulseTime = now;
  pulse++;
  dataLed = 1;

  tampilDeteksi = true;
  waktuDeteksiLCD = millis();
}

// Matikan LED dan buzzer setelah LED_DURATION
if (isStriking && (now - lastTriggerTime >= LED_DURATION)) {
  digitalWrite(LED_strike, LOW);
  digitalWrite(Buzzer, LOW);
  isStriking = false;
}
}
```

Gambar 4. 40 Fungsi Buzzer Deteksi Petir

Dalam fungsi deteksi petir juga terdapat logika Non Blocking Delay yang berfungsi untuk mematikan buzzer. Logika ini memastikan bahwa buzzer hanya menyala selama 100 milidetik setelah deteksi petir, tanpa menghentikan program utama. Pendekatan non-blocking ini sangat penting agar sistem tetap dapat merespons input lain.

```
const int deltaThreshold = 5;                // Ambang perubahan ADC
const unsigned long LED_DURATION = 100;      // Lama LED menyala (ms)
const unsigned long cooldownDuration = 200;  // Jeda antar trigger petir (ms)
int dataLed;
```

Gambar 4. 41 Fungsi mematikan Buzzer

4.2.2.7 Pembuatan Program LED

LED pada sistem ini berperan sebagai indikator visual lokal untuk menampilkan status aktivitas petir. Perancangan awal dimulai dengan mendefinisikan pin digital yang digunakan, yaitu D4, yang kemudian disebut sebagai LED. Pin ini diatur sebagai pin output karena LED hanya perlu menerima logika tegangan untuk menyala (HIGH) atau mati (LOW).

```
#define LED_strike D4
```

Gambar 4. 42 Deklarasi LED

Selanjutnya Inisialisasi dilakukan di dalam fungsi setup() dengan menggunakan perintah pinMode(LED, OUTPUT) dan nilai awal digitalWrite(LED, LOW) untuk memastikan LED tidak aktif ketika alat baru menyala.

```
pinMode(LED, OUTPUT);
digitalWrite(LED, LOW);
```

Gambar 4. 43 Inisialisasi LED

Dalam fungsi deteksi petir LED diaktifka sebagai tanda bahwa adanya deteksi lonjakan sinyal analog dari sensor TA7642. LED dan buzzer akan dinyalakan secara bersamaan, variabel `pulse` dinaikkan sebagai penghitung kejadian, dan variabel `dataLed` diset sebagai penanda status untuk pengiriman data ke ThingSpeak. Waktu penyalaan dicatat dalam `lastTriggerTime`, dan LED dimatikan kembali setelah melewati durasi pendek (`LED_DURATION`) guna menghemat daya dan mencegah alarm berkepanjangan. Pada saat yang sama, flag `tampilDeteksi` diaktifkan agar LCD menampilkan pesan peringatan “DETEKSI PETIR”. Dengan pola ini, indikator LED dan buzzer menyediakan umpan balik instan kepada pengguna, sekaligus terintegrasi dengan sistem pencatatan daring melalui pengiriman status event dan jumlah kejadian ke platform ThingSpeak.

```
// Trigger petir
if (Delta >= deltaThreshold && !isStriking && (now - lastPulseTime > cooldownDuration)) {
  digitalWrite(LED_strike, HIGH);
  digitalWrite(Buzzer, HIGH);
  isStriking = true;
  lastTriggerTime = now;
  lastPulseTime = now;
  pulse++;
  dataLed = 1;

  tampilDeteksi = true;
  waktuDeteksiLCD = millis();
}

// Matikan LED dan buzzer setelah LED_DURATION
if (isStriking && (now - lastTriggerTime >= LED_DURATION)) {
  digitalWrite(LED_strike, LOW);
  digitalWrite(Buzzer, LOW);
  isStriking = false;
}
}
```

Gambar 4. 44 Fungsi LED Deteksi Petir

Bagian ini berfungsi sebagai simulasi dan dapat dengan mudah diubah agar LED merespons kondisi sebenarnya dari jumlah pulsa deteksi petir. Pada LED juga menggunakan Non Blocking Delay yang berfungsi untuk mematikan LED setelah menyala selama 100 milidetik. Durasi yang singkat namun efektif ini memberikan sinyal visual secara cepat tanpa menghambat alur eksekusi utama..

```
const int deltaThreshold = 5;                // Ambang perubahan ADC
const unsigned long LED_DURATION = 100;      // Lama LED menyala (ms)
const unsigned long cooldownDuration = 200;  // Jeda antar trigger petir (ms)
int dataLed;
```

Gambar 4. 45 Fungsi mematikan LED