Bab III
Metode Penelitian

3.1 Waktu dan Tempat

Penelitian ini dilakukan di Laboratorium Program Studi Teknologi Rekayasa Otomasi, Sekolah Vokasi, Universitas Diponegoro. Waktu yang diperlukan dalam penelitian yaitu selama 3 bulan dengan estimasi waktu dimulai dari bulan Januari 2026 s.d. Maret 2026.

3.2 Alat dan Bahan

Pelaksanaan penelitian ini didukung dengan berbagai alat dan bahan pendukung. Alat dan bahan yang digunakan berguna untuk memperlancar proses kerja alat yang dirancang. Alat dan bahan yang digunakan adalah sebagai berikut.

a. Mikrokontroler

   Perangkat yang digunakan yaitu ESP32-S3 Super Mini pabrikan Espressif untuk menjalankan sistem kendali pada kesatuan perangkat yang dirancang.

b. Perangkat Lunak

   Perangkat Lunak yang digunakan adalah Arduino IDE untuk membuat dan mengunggah program pada Mikrokontroler.

c. Catu Daya

   Perangkat ini berfungsi menyuplai tenaga untuk mengaktifkan dan menjalankan komponen yang digunakan.

d. Sensor

   Sensor Gas MQ-2 digunakan untuk mengubah pembacaan Gas menjadi sinyal yang dapat dibaca oleh Mikrokontroler.

e. Printed Circuit Board

   PCB digunakan untuk menyatukan komponen yang diperlukan dan untuk mengurangi noise.

f. Solder

   Solder digunakan untuk menyatukan seluruh komponen dalam PCB.

3.3 Deskripsi Sistem dan Cara Kerja

Analisis sistem dari perancangan alat Pendeteksi Gas yaitu sebagai alat yang mampu membaca keberadaan Gas LPG didalam ruangan dan berfungsi untuk memberikan peringatan dini. Hal ini dapat bermanfaat untuk mencegah terjadinya kebakaran.

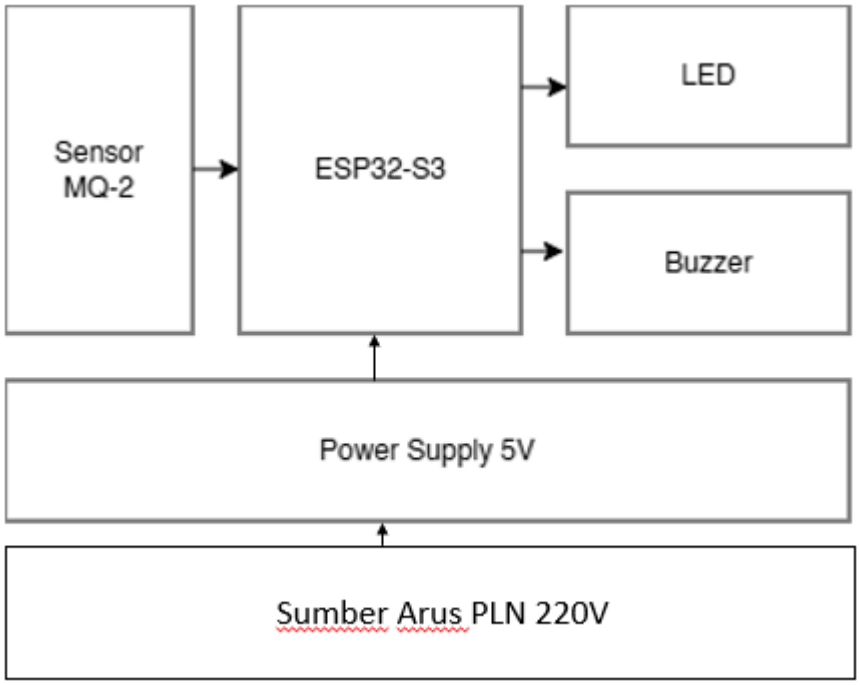
Alur kerja sistem dimulai dari proses pemantauan kondisi lingkungan yang dilakukan secara terus-menerus oleh sensor gas. Sensor MQ-2 berfungsi mendeteksi keberadaan gas mudah terbakar di udara dengan cara membandingkan perubahan resistansi material sensitif didalam sensor terhadap gas. Ketika terdapat gas di lingkungan sekitar, sensor akan menghasilkan sinyal analog yang nilainya sebanding dengan konsentrasi gas yang terdeteksi. Sinyal analog dari sensor kemudian dikirimkan ke mikrokontroler untuk diproses. Mikrokontroler membaca nilai tegangan dari sensor melalui pin input analog, lalu mengolah data tersebut dengan membandingkannya terhadap nilai ambang batas yang telah ditentukan dalam program. Proses ini dilakukan secara berulang sehingga sistem mampu melakukan pemantauan secara real time.

Apabila nilai pembacaan sensor berada di bawah ambang batas, sistem akan menganggap kondisi lingkungan masih aman. Dalam kondisi ini, mikrokontroler hanya melanjutkan proses pemantauan tanpa mengaktifkan indikator peringatan. Sebaliknya, jika nilai sensor melebihi ambang batas, sistem akan mengidentifikasi kondisi sebagai potensi bahaya. Ketika kondisi bahaya terdeteksi, mikrokontroler akan mengaktifkan indikator lokal (LED) sebagai tanda peringatan langsung di lokasi alat. Aktivasi indikator ini bertujuan memberikan respons cepat kepada pengguna yang berada di sekitar alat. Selanjutnya, mikrokontroler memanfaatkan koneksi Wi-Fi untuk mengirimkan data peringatan melalui jaringan internet. Sistem akan secara otomatis mengirimkan pesan notifikasi ke aplikasi Telegram yang berisi informasi mengenai kondisi terdeteksi, sehingga pengguna dapat segera mengetahui situasi dan mengambil tindakan penanganan yang diperlukan.

Proses tersebut berlangsung secara berkelanjutan selama alat dalam kondisi aktif. Dengan demikian, sistem mampu memberikan pemantauan lingkungan secara otomatis, mendeteksi potensi kebocoran gas atau kebakaran sejak dini, serta mengirimkan peringatan secara cepat dan tepat waktu kepada pengguna.

3.4 Blok Diagram Perancangan Alat

Sensor Gas MQ-2 sebagai penghasil sinyal analog dan diolah oleh mikrokontroler sebagai pusat pengolahan data serta meneruskannya menjadi output berupa indikator lokal dan notifikasi eksternal. Untuk memahami dengan mudah struktur kerja pada sistem pendeteksi kebocoran gas, maka disediakan blok diagram pada Gambar 3.1 berikut.



Gambar 3. 1 Blok Diagram

Bagian-bagian dari blok diagram tersebut meliputi:

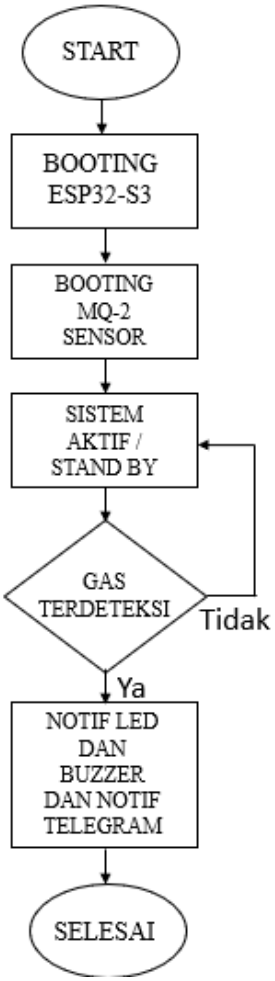
1. ESP32-S3 berfungsi sebagai mikrokontroler utama yang mengendalikan seluruh proses pada sistem. Komponen ini bertugas untuk membaca data dari sensor MQ2, memproses data tersebut, serta menentukan tindakan yang akan

dilakukan oleh sistem. Berdasarkan hasil pembacaan sensor, ESP32-S3 akan mengaktifkan indikator seperti buzzer dan LED sebagai bentuk peringatan. Selain itu, ESP32-S3 juga mengatur komunikasi antar komponen dalam sistem.

2. Sensor MQ-2 digunakan sebagai sensor pendeteksi gas dan asap. Sensor ini mampu mendeteksi keberadaan gas yang mudah terbakar seperti LPG, propana, metana, serta asap. Sensor akan menghasilkan sinyal analog yang menunjukkan konsentrasi gas di lingkungan sekitar. Nilai tersebut kemudian dikirimkan ke ESP32-S3 untuk diproses lebih lanjut.
3. Buzzer berfungsi sebagai indikator peringatan suara. Ketika sistem mendeteksi adanya konsentrasi gas atau asap yang melebihi batas tertentu, ESP32-S3 akan mengaktifkan buzzer sehingga menghasilkan bunyi alarm. Tujuan dari buzzer adalah memberikan peringatan kepada pengguna secara cepat melalui sinyal suara.
4. LED digunakan sebagai indikator visual pada sistem. LED akan menyala untuk memberikan tanda kondisi tertentu, misalnya kondisi normal atau kondisi bahaya ketika gas terdeteksi. Dengan adanya LED, pengguna dapat mengetahui status sistem secara visual.

3.5 Diagram Alir Sistem

Pada perancangan alat deteksi gas ini terdapat diagram alir untuk menentukan urutan proses kerja dari sistem kerja alat tersebut. Diagram alir dapat dilihat pada Gambar 3.2 berikut.



Gambar 3. 2 Diagram Alir

3.6 Desain Alat

Perencanaan pembuatan alat pendeteksi gas membahas tentang desain yang akan penulis rancang. Penulis merancang gambar 3D alat menggunakan *Software* Autodesk Fusion 360. Perancangan gambar 3D alat pendeteksi gas dapat dilihat pada Gambar 3.3 dan Gambar 3.4 berikut.

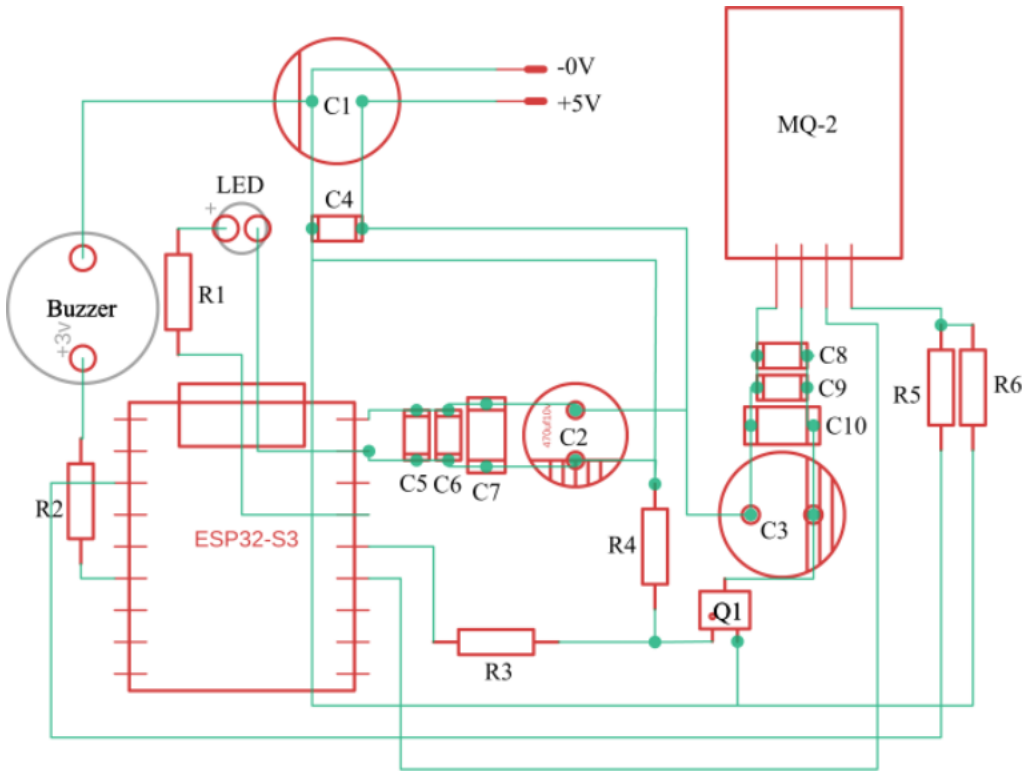
Gambar 3. 3 Desain Alat Tampak Depan Serong Kanan

Gambar 3. 4 Desain Alat Tampak Depan Serong Kiri

Untuk penempatan sensor MQ-2 tepat berada di tengah cover berbentuk tabung 8x3cm, komponen lain seperti ESP32-S3, Buzzer, Kapasitor, Transistor dan LED menyesuaikan pada ruang kosong di dalam cover alat tersebut.

3.7 Perancangan Rangkaian Sistem Alat

Perancangan rangkaian sistem alat merupakan tahapan yang dilakukan untuk menghubungkan seluruh komponen perangkat keras sehingga dapat bekerja secara terintegrasi sesuai dengan fungsi yang telah dirancang. Pada penelitian ini, sistem dirancang menggunakan beberapa komponen utama yang terdiri dari mikrokontroler ESP32-S3, sensor MQ-2 sebagai pendeteksi gas, buzzer sebagai indikator peringatan suara, LED sebagai indikator visual, serta *power supply* sebagai sumber daya listrik dapat dilihat pada Gambar 3.5.



Gambar 3. 5 Rangkaian Sistem Alat

Sensor MQ-2 digunakan untuk mendeteksi keberadaan gas LPG di lingkungan sekitar. Sensor ini akan menghasilkan sinyal berupa tegangan analog yang merepresentasikan konsentrasi gas yang terdeteksi di udara. Sinyal tersebut kemudian dikirimkan ke mikrokontroler ESP32-S3 melalui pin input untuk dilakukan proses pembacaan dan pengolahan data.

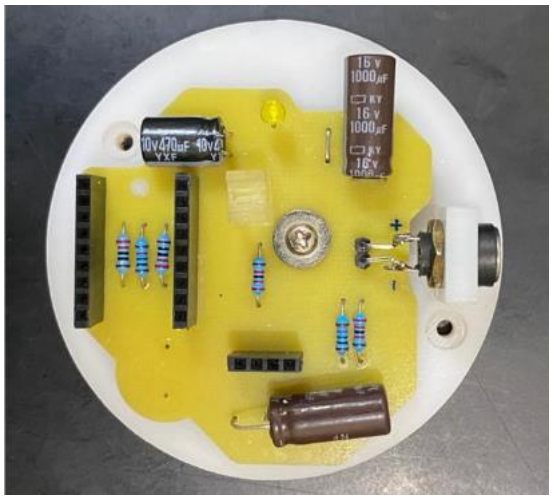
Mikrokontroler ESP32-S3 berperan sebagai pusat pengendali sistem yang bertugas untuk memproses data yang diterima dari sensor MQ-2. Berdasarkan hasil pengolahan data tersebut, mikrokontroler akan menentukan kondisi sistem apakah berada dalam keadaan normal atau terdeteksi adanya kebocoran gas. Apabila nilai konsentrasi gas melebihi ambang batas yang telah ditentukan, maka mikrokontroler akan mengaktifkan buzzer sebagai alarm peringatan serta menyalakan LED sebagai indikator visual.

Selain itu, sistem juga dirancang untuk terhubung dengan jaringan internet sehingga memungkinkan pengiriman notifikasi kepada pengguna melalui aplikasi Telegram secara real-time. Hal ini dilakukan dengan memanfaatkan kemampuan komunikasi jaringan yang dimiliki oleh ESP32-S3.

Seluruh komponen dalam rangkaian sistem memperoleh sumber daya listrik dari *power supply* yang dirancang untuk memberikan tegangan yang stabil agar sistem dapat bekerja secara optimal. Dengan adanya perancangan rangkaian ini, diharapkan seluruh komponen dapat saling berinteraksi dengan baik sehingga sistem pendeteksi kebocoran gas dapat bekerja secara efektif dan responsif.

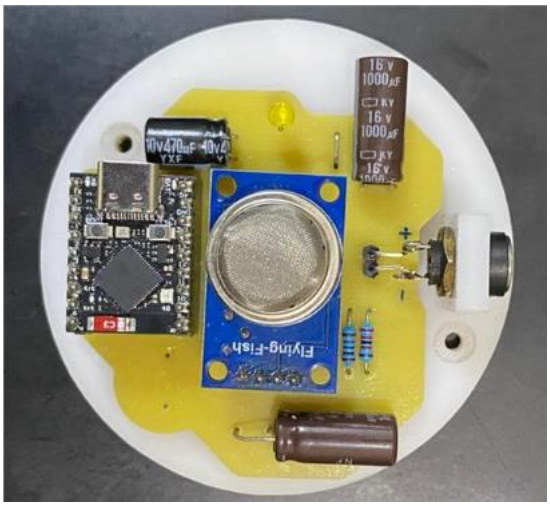
3.8 Perancangan Alat

Perancangan alat merupakan tahap implementasi dari perancangan rangkaian sistem yang telah dibuat sebelumnya. Pada tahap ini dilakukan proses penyusunan dan integrasi seluruh komponen elektronik ke dalam sebuah papan rangkaian tercetak atau Printed Circuit Board (PCB) sehingga membentuk sebuah perangkat yang dapat berfungsi sebagai sistem pendeteksi kebocoran gas.



Gambar 3. 6 Rangakaian Komponen dalam PCB

Penggunaan PCB dalam perancangan alat bertujuan untuk menghasilkan rangkaian yang lebih rapi, stabil, serta meminimalkan kesalahan koneksi yang umumnya terjadi pada penggunaan kabel atau breadboard. Dengan menggunakan PCB, seluruh jalur koneksi antar komponen telah dirancang secara permanen sehingga meningkatkan keandalan sistem.



Gambar 3. 7 Rangkaian Keseluruhan Komponen dalam PCB

Pada alat yang dirancang dalam penelitian ini, komponen utama seperti mikrokontroler ESP32-S3, sensor gas MQ-2, buzzer, LED indikator, resistor, dan kapasitor dipasang secara langsung pada PCB sesuai dengan jalur

rangkaian yang telah dirancang sebelumnya. Sensor MQ-2 ditempatkan pada bagian atas papan rangkaian agar dapat mendeteksi keberadaan gas di lingkungan sekitar secara optimal.

Gambar 3. 8 Cover Alat Tampak Bawah

Sebagai tahap akhir dari purwarupa perangkat keras, PCB yang telah dirakit diintegrasikan ke dalam casing pelindung difabrikasi menggunakan teknologi 3D printing. Papan PCB juga difiksasi menggunakan pilar dudukan (standoff) di dalam casing untuk memastikan stabilitas mekanis alat saat dipasang di dinding atau area rawan getaran.

Gambar 3. 9 Cover Alat Tampak Atas

Desain casing dilengkapi dengan kisi-kisi ventilasi yang diposisikan secara presisi di area sensor MQ-2. Perancangan casing ini difokuskan pada perlindungan komponen dari kontak fisik langsung dan optimasi sirkulasi udara. Penempatan ventilasi ini sangat krusial agar gas dari lingkungan sekitar dapat masuk dan berinteraksi dengan elemen sensing secara optimal tanpa hambatan sekaligus memfasilitasi penyebaran energi panas dari heater sensor.

3.9 Perancangan Perangkat Lunak dan Integrasi Sistem

Perancangan perangkat lunak pada sistem ini difokuskan pada keandalan (*reliability*) dan kemampuan pemrosesan data secara *real-time*. Perangkat lunak diimplementasikan pada mikrokontroler ESP32-S3 Super Mini menggunakan bahasa pemrograman C/C++ dengan *framework* Arduino. Arsitektur program dirancang secara modular sehingga proses pembacaan sensor, pengolahan data, pengendalian aktuator, serta komunikasi jaringan dapat berjalan secara terstruktur dan mudah dikembangkan.

Pada saat sistem dinyalakan, ESP32-S3 Super Mini melakukan proses inisialisasi seluruh perangkat yang digunakan, meliputi sensor gas MQ-2 sebagai masukan (*input*), LED dan buzzer sebagai perangkat keluaran (*output*), serta modul Wi-Fi bawaan untuk komunikasi jaringan. Setelah proses inisialisasi selesai, perangkat akan melakukan koneksi ke jaringan Wi-Fi yang telah dikonfigurasi. Koneksi internet diperlukan agar sistem dapat mengirimkan notifikasi secara otomatis kepada pengguna melalui aplikasi Telegram.

Setelah berhasil terhubung ke jaringan, program memasuki proses utama (*main loop*) yang berjalan secara terus-menerus. Pada setiap siklus, ESP32-S3 membaca nilai analog yang dihasilkan oleh sensor MQ-2. Nilai tersebut kemudian dibandingkan dengan batas ambang (*threshold*) yang telah ditentukan untuk mengidentifikasi apakah terdapat indikasi kebocoran gas.

Apabila nilai sensor melebihi ambang batas, sistem akan mengaktifkan buzzer sebagai alarm suara dan LED sebagai indikator visual bahwa telah terdeteksi kebocoran gas. Sebaliknya, apabila nilai sensor berada di bawah batas

ambang, buzzer dan LED akan tetap berada dalam kondisi tidak aktif sehingga menunjukkan kondisi lingkungan yang masih aman.

Selain memberikan peringatan secara lokal, sistem juga diintegrasikan dengan layanan Telegram agar pengguna dapat menerima notifikasi secara jarak jauh. Integrasi dilakukan menggunakan Telegram Bot API. Sebelum sistem dijalankan, bot Telegram dibuat melalui layanan BotFather untuk memperoleh Bot Token yang berfungsi sebagai identitas autentikasi bot. Selain itu, sistem juga menggunakan Chat ID sebagai tujuan pengiriman pesan kepada pengguna.

ESP32-S3 memanfaatkan pustaka WiFiClientSecure dan HTTPClient untuk membangun komunikasi menggunakan protokol HTTPS dengan server Telegram. Ketika kondisi kebocoran gas terdeteksi, mikrokontroler mengirimkan permintaan HTTP (*HTTP Request*) ke endpoint Telegram Bot API dengan menyertakan *Bot Token*, *Chat ID*, serta isi pesan notifikasi. Format endpoint yang digunakan adalah:

https://api.telegram.org/bot${token}/sendMessage

dengan parameter *chat_id* sebagai tujuan pengiriman pesan dan *text* sebagai isi notifikasi. Penggunaan protokol HTTPS memastikan proses komunikasi berlangsung secara terenkripsi sehingga data yang dikirimkan lebih aman. Untuk implementasi pada pemograman Arduino dapat dilihat pada Gambar 3.10.

```
void executeSend(int gasValue, bool isHwTriggered, unsigned long uptimeSec) {
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("[TELEGRAM] Aborted: WiFi Disconnected.");
        return;
    }

    HTTPClient https;
    char url[128];
    snprintf(url, sizeof(url), "https://api.telegram.org/bot%s/sendMessage", _token);

    String message;
    message.reserve(256);
    message = "⚠️ *BAHAYA KEBOCORAN GAS* ⚠️\n\n";
    message += "📈 Level Sensor: " + String(gasValue) + "\n";
    message += "🎛️ HW Status: " + String(isHwTriggered ? "BAHAYA (HW Trigger)" : "Tidak (SW Only)") + "\n";
    message += "📍 Lokasi: Dapur\n";
    message += "⏱️ Uptime: " + String(uptimeSec) + "s";

    String jsonPayload = "{\"chat_id\": \"" + String(_chatId) + "\", \"text\": \"" + message + "\", \"parse_mode\": \"Markdown\"}";

    if (https.begin(_client, url)) {
        https.addHeader("Content-Type", "application/json");
        int httpCode = https.POST(jsonPayload);

        if (httpCode == 200 || httpCode == 201) {
            Serial.println("[TELEGRAM] Notifikasi berhasil dikirim!");
        } else {
            Serial.printf("[TELEGRAM] Gagal kirim. HTTP Code: %d\n", httpCode);
        }
        https.end();
    }
}
```

Gambar 3. 10 Block Code Arduino Kirim Notifikasi Telegram

Untuk menghindari pengiriman pesan secara berulang ketika kondisi kebocoran masih berlangsung, perangkat lunak menerapkan mekanisme *state management*. Notifikasi hanya dikirim ketika terjadi perubahan kondisi dari aman menjadi bahaya. Setelah konsentrasi gas kembali berada di bawah nilai ambang, status sistem akan diatur kembali sehingga notifikasi dapat dikirim apabila kebocoran terjadi kembali di kemudian hari.

Dengan rancangan tersebut, sistem mampu melakukan pemantauan kadar gas secara terus-menerus, memberikan peringatan lokal melalui LED dan buzzer, serta mengirimkan notifikasi secara otomatis kepada pengguna melalui Telegram tanpa mengganggu proses pembacaan sensor yang berlangsung secara *real-time*.

3.9.1 Inisialisasi dan Manajemen State

Pada saat perangkat dinyalakan, ESP32-S3 Super Mini menjalankan proses inisialisasi seluruh komponen perangkat keras dan perangkat lunak. Tahap ini meliputi konfigurasi komunikasi serial (*Serial Communication*) sebagai media *debugging*, inisialisasi pin *General Purpose Input/Output* (GPIO), sensor gas MQ-2, LED indikator, buzzer, serta layanan notifikasi Telegram. Selain itu, sistem juga menginisialisasi modul Wi-Fi menggunakan pustaka WiFiManager untuk mempermudah proses konfigurasi jaringan.

Berbeda dengan metode *hardcode* SSID dan *password*, WiFiManager memungkinkan pengguna melakukan konfigurasi jaringan melalui *captive portal* dengan nama akses poin ESP32-S3-GAS. Apabila perangkat belum memiliki konfigurasi Wi-Fi atau gagal terhubung ke jaringan yang tersimpan, ESP32-S3 secara otomatis membuat *access point* sehingga pengguna dapat memasukkan kredensial Wi-Fi melalui *web browser*. Setelah konfigurasi berhasil dilakukan, data jaringan akan disimpan pada memori internal ESP32 sehingga tidak perlu dikonfigurasi kembali pada proses *booting* berikutnya.

Setelah perangkat berhasil terhubung ke jaringan Wi-Fi, catu daya sensor MQ-2 diaktifkan melalui pin kontrol (PIN_SENSOR_POWER). Sistem kemudian memasuki *state* WARMUP, yaitu tahap pemanasan sensor selama ±60 detik. Tahap ini diperlukan karena sensor MQ-2 menggunakan elemen pemanas (*heater*) yang membutuhkan waktu untuk mencapai kondisi kerja yang stabil sehingga hasil pembacaan menjadi lebih konsisten.

Selama *state* WARMUP, fungsi isWarmedUp() digunakan untuk memeriksa apakah waktu pemanasan telah terpenuhi. Selama proses tersebut berlangsung, sistem tetap melakukan pembacaan nilai analog sensor untuk keperluan pemantauan, namun nilai tersebut belum digunakan sebagai dasar pengambilan keputusan. LED indikator berkedip setiap 500 ms sebagai penanda visual bahwa proses inisialisasi masih berlangsung, sedangkan buzzer memberikan bunyi singkat selama sekitar 100 ms setiap 5 detik sebagai indikasi bahwa perangkat masih berada pada tahap pemanasan. Seluruh pola indikator dijalankan menggunakan fungsi millis() sehingga bersifat *non-blocking* dan tidak menghentikan proses utama sistem.

Apabila waktu pemanasan telah selesai, sistem secara otomatis berpindah dari *state* WARMUP menuju IDLE. Pada kondisi IDLE, sistem berada dalam keadaan siaga dengan LED dan buzzer dalam kondisi tidak aktif, kemudian mulai melakukan pemantauan kadar gas secara berkala setiap 1 detik.

Manajemen kondisi sistem diimplementasikan menggunakan konsep Finite State Machine (FSM) yang terdiri atas lima *state*, yaitu BOOTING,

WIFI_SETUP, WARMUP, IDLE, dan DANGER. Pendekatan FSM dipilih agar setiap kondisi operasi memiliki fungsi dan perilaku yang jelas sehingga perpindahan antar kondisi dapat dikendalikan secara terstruktur. Konsep *Finite State* pada sistem ditunjukkan pada Gambar 3.11.

```
if (currentState == SystemState::WARMUP) {
    if (!sensor.isWarmedUp()) {
        Serial.printf("[WARMUP] %lu s remaining | Val: %d\n", sensor.getRemainingWarmup(), analogVal);
        return; // Tunggu sampai panas
    } else {
        Serial.println("[SYSTEM] Warm-up Selesai. Masuk mode Siaga (IDLE).");
        currentState = SystemState::IDLE;
        alertSystem.setMode(currentState);
    }
}
```

Gambar 3. 11 Konsep *Finite State* pada Pemograman Arduino

3.9.2 Akuisisi dan Pemfilteran Data Sensor

Sensor MQ-2 menghasilkan sinyal analog yang merepresentasikan perubahan resistansi akibat keberadaan gas di lingkungan. Nilai analog tersebut dibaca oleh modul *Analog-to-Digital Converter* (ADC) pada ESP32-S3 secara periodik setiap 1 detik sesuai nilai konstanta READ_INTERVAL_MS. Interval pembacaan yang tetap bertujuan menjaga konsistensi proses pemantauan tanpa membebani mikrokontroler dengan pembacaan yang terlalu sering.

Untuk mengurangi fluktuasi pembacaan akibat *noise* listrik maupun karakteristik alami sensor, sistem tidak menggunakan hasil pembacaan ADC tunggal. Setiap proses akuisisi dilakukan sebanyak 10 kali, kemudian dihitung nilai rata-ratanya (*moving average sederhana*) sehingga menghasilkan nilai sensor yang lebih stabil sebelum digunakan pada proses pengambilan keputusan.

Nilai ADC hasil pemfilteran selanjutnya dibandingkan dengan nilai ambang (*threshold*) GAS_THRESHOLD_SOFT. Apabila nilai pembacaan melebihi ambang batas tersebut, sistem menganggap telah terjadi indikasi kebocoran gas dan mengubah kondisi sistem ke *state* DANGER. Sebaliknya, apabila nilai sensor kembali berada di bawah ambang batas bawah yang ditentukan menggunakan algoritma histeresis, sistem akan kembali ke kondisi IDLE.

Pendekatan ini menghasilkan proses deteksi yang lebih stabil dibandingkan penggunaan pembacaan tunggal karena mampu mengurangi pengaruh fluktuasi sesaat (*transient noise*) tanpa mengurangi respons sistem terhadap perubahan konsentrasi gas.

Nilai ambang (*threshold*) pada penelitian ini ditetapkan sebesar **850** berdasarkan hasil pengujian awal (*preliminary testing*) terhadap karakteristik keluaran sensor MQ-2. Pengamatan dilakukan dengan membandingkan nilai ADC saat kondisi lingkungan normal dan saat sensor diberikan paparan gas LPG. Berdasarkan hasil pengujian tersebut, nilai 850 dipilih karena mampu membedakan kondisi normal dan kondisi terpapar gas dengan baik serta meminimalkan terjadinya *false alarm*. Nilai ambang ini kemudian dikombinasikan dengan algoritma histeresis untuk meningkatkan kestabilan sistem.

3.9.3 Respon Keamanan Lokal (Aktuator)

Sebagai sistem peringatan dini, respon lokal memiliki prioritas utama sehingga tetap dapat berfungsi meskipun koneksi internet tidak tersedia. Pengendalian aktuator dilakukan secara independen melalui modul AlertController yang bertanggung jawab mengatur pola kerja LED dan buzzer berdasarkan *state* sistem.

Selama proses WARMUP, LED berkedip setiap 500 ms sebagai indikator bahwa sensor masih berada pada tahap pemanasan, sedangkan buzzer memberikan bunyi singkat selama sekitar 100 ms setiap 5 detik. Pola ini memberikan informasi kepada pengguna bahwa perangkat sedang melakukan proses inisialisasi dan belum siap melakukan deteksi kebocoran gas.

Ketika sistem berada pada kondisi IDLE, LED dan buzzer berada dalam kondisi tidak aktif sebagai indikasi bahwa lingkungan berada dalam keadaan aman.

Apabila nilai sensor melampaui nilai ambang batas sehingga sistem memasuki *state* DANGER, LED dan buzzer diaktifkan secara bersamaan dengan pola kedip cepat setiap 150 ms. Pola ini dirancang agar mudah dikenali sebagai kondisi darurat sehingga pengguna dapat segera mengambil tindakan.

Seluruh pengendalian aktuator diimplementasikan menggunakan fungsi millis() sehingga bersifat *non-blocking*. Dengan pendekatan ini, proses pembacaan sensor dan komunikasi jaringan tetap dapat berjalan secara bersamaan tanpa saling menghambat.

3.9.4 Integrasi Telemetri via Telegram API

Integrasi sistem dengan Telegram Bot API dirancang untuk memberikan notifikasi secara real-time kepada pengguna ketika terdeteksi kebocoran gas. Struktur Program Jaringan (TelegramNotifier) ditunjukkan pada Gambar 3.12.

```
enum class SystemState {
    BOOTING,
    WIFI_SETUP,
    WARMUP,
    IDLE,
    DANGER
};

class TelegramNotifier {
private:
    const char* _token;
    const char* _chatId;
    WiFiClientSecure _client;

    struct AlertPayload {
        TelegramNotifier* instance;
        int gasValue;
        bool isHwTriggered;
        unsigned long uptimeSec;
    };

    static void asyncTask(void* parameter) {
        AlertPayload* payload = static_cast<AlertPayload*>(parameter);
        payload->instance->executeSend(payload->gasValue, payload->isHwTriggered, payload->uptimeSec);
        delete payload;
        vTaskDelete(NULL);
    }

public:
    TelegramNotifier(const char* token, const char* chatId)
        : _token(token), _chatId(chatId) {
        _client.setInsecure();
    }

    void sendAlertAsync(int gasValue, bool isHwTriggered) {
        AlertPayload* payload = new AlertPayload{this, gasValue, isHwTriggered, millis() / 1000};
        xTaskCreatePinnedToCore(asyncTask, "TeleTask", 8192, payload, 1, NULL, 0);
    }
};
```

Gambar 3. 12 Struktur Program Jaringan *(TelegramNotifier)*

Perancangan perangkat lunak menerapkan pendekatan modular dengan memisahkan proses komunikasi jaringan ke dalam kelas TelegramNotifier, sehingga logika pengiriman notifikasi tidak bercampur dengan proses akuisisi

data sensor maupun pengendalian aktuator. Pendekatan ini menerapkan prinsip Single Responsibility Principle (SRP), di mana setiap kelas memiliki tanggung jawab yang spesifik sehingga struktur program menjadi lebih terorganisasi, mudah dipelihara, dan lebih mudah dikembangkan.

Pada saat kondisi bahaya terdeteksi, sistem tidak secara langsung melakukan komunikasi HTTP pada *main loop*. Sebaliknya, metode sendAlertAsync() dipanggil untuk membuat sebuah *task* baru menggunakan fungsi xTaskCreatePinnedToCore() yang disediakan oleh sistem operasi FreeRTOS pada ESP32-S3. Melalui mekanisme ini, proses pengiriman notifikasi dijalankan secara asinkron pada inti prosesor (*core*) yang terpisah, sehingga waktu tunggu (*latency*) akibat komunikasi jaringan tidak menghambat proses pembacaan sensor maupun pengendalian alarm lokal. Pendekatan ini memungkinkan sistem tetap mempertahankan karakteristik *real-time* meskipun sedang melakukan komunikasi dengan server Telegram.

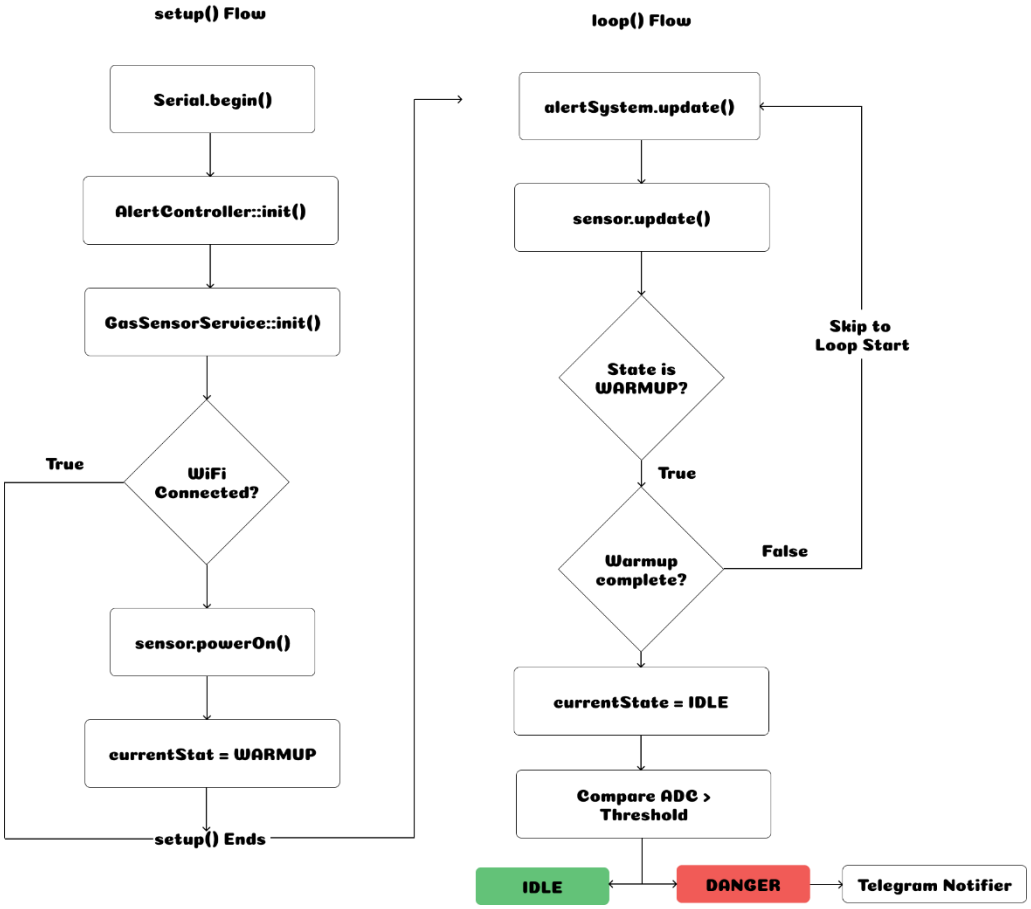
Proses komunikasi jaringan menggunakan pustaka WiFiClientSecure dan HTTPClient sehingga pertukaran data dilakukan melalui protokol HTTPS. Endpoint Telegram yang digunakan mengikuti format:

https://api.telegram.org/bot<TOKEN>/sendMessage

Data notifikasi disusun dalam format JavaScript Object Notation (JSON) yang berisi parameter chat_id sebagai tujuan pengiriman pesan, text sebagai isi notifikasi, serta parse_mode untuk mendukung format *Markdown*. Informasi yang dikirimkan meliputi nilai pembacaan sensor MQ-2 (*gas value*), status keluaran digital sensor (*hardware trigger*), lokasi perangkat, dan waktu aktif sistem (*uptime*). Setelah *payload* selesai dibentuk, data dikirim ke server Telegram menggunakan metode HTTP POST.

Pada implementasi ini, objek WiFiClientSecure dikonfigurasi menggunakan fungsi _client.setInsecure(). Pengaturan tersebut membuat ESP32-S3 tetap menggunakan koneksi HTTPS melalui port 443, namun tidak melakukan proses verifikasi sertifikat Root Certificate Authority (Root CA) milik server Telegram. Pendekatan ini dipilih untuk menyederhanakan

implementasi prototipe dan mengurangi kompleksitas pengelolaan sertifikat pada perangkat dengan sumber daya terbatas. Meskipun demikian, konfigurasi tersebut menyebabkan identitas server tidak diverifikasi sehingga secara teoritis masih memiliki risiko terhadap serangan *Man-in-the-Middle* (MITM). Oleh karena itu, pada implementasi sistem yang akan digunakan di lingkungan operasional, disarankan menggunakan sertifikat Root CA resmi Telegram agar proses autentikasi server dapat dilakukan sebelum pertukaran data berlangsung.



Gambar 3. 13 Diagram Struktur Utama

Pada program utama, pengiriman notifikasi diintegrasikan dengan mekanisme *Finite State Machine* (FSM). Setelah sensor MQ-2 menyelesaikan proses *warm-up*, sistem secara berkala melakukan pembacaan sensor setiap satu detik. Nilai hasil pembacaan kemudian dibandingkan dengan nilai ambang GAS_THRESHOLD_SOFT. Ketika nilai sensor pertama kali melampaui

ambang batas dan sistem berpindah dari *state* IDLE ke DANGER, metode sendAlertAsync() dipanggil untuk mengirimkan notifikasi Telegram. Karena pemanggilan fungsi ini hanya dilakukan saat terjadi perubahan *state*, sistem secara otomatis hanya mengirimkan satu notifikasi untuk setiap kejadian kebocoran gas.

Untuk mencegah perubahan kondisi yang terlalu sering akibat fluktuasi nilai sensor di sekitar ambang batas, sistem menerapkan algoritma histeresis menggunakan parameter HYSTERESIS. Dengan mekanisme ini, *state* DANGER tidak akan langsung dinonaktifkan ketika nilai sensor sedikit menurun, melainkan baru kembali ke *state* IDLE setelah nilai pembacaan berada di bawah batas GAS_THRESHOLD_SOFT − HYSTERESIS. Pendekatan tersebut menghasilkan proses deteksi yang lebih stabil, mengurangi perpindahan *state* yang berulang, serta mencegah pengiriman notifikasi Telegram secara berlebihan akibat perubahan nilai sensor yang bersifat sesaat.

3.9.5 Alur Kerja Sistem (Finite State Machine)

Secara keseluruhan, alur operasional perangkat lunak dirancang menggunakan pendekatan Finite State Machine (FSM). Pendekatan ini membagi proses kerja sistem ke dalam beberapa *state* yang saling terhubung sehingga setiap kondisi operasi memiliki fungsi dan transisi yang jelas. Dengan penerapan FSM, logika program menjadi lebih terstruktur, mudah dipelihara, serta mampu menghindari perpindahan kondisi yang tidak diinginkan akibat fluktuasi pembacaan sensor.

Diagram *Finite State Machine* yang digunakan pada sistem ditunjukkan pada Gambar 3.12. FSM terdiri atas lima *state* utama, yaitu BOOTING, WIFI_SETUP, WARMUP, IDLE, dan DANGER.

1. State BOOTING

   Tahap ini merupakan kondisi awal ketika ESP32-S3 pertama kali dinyalakan. Sistem melakukan inisialisasi komunikasi serial, konfigurasi pin GPIO, inisialisasi sensor MQ-2, LED, buzzer, serta seluruh layanan

perangkat lunak yang digunakan. Setelah proses inisialisasi selesai, sistem berpindah ke *state* WIFI_SETUP.

2. State WIFI_SETUP

   Pada tahap ini sistem melakukan konfigurasi dan koneksi jaringan menggunakan pustaka WiFiManager. Apabila kredensial Wi-Fi telah tersedia dan koneksi berhasil dilakukan, sistem akan melanjutkan proses dengan mengaktifkan catu daya sensor MQ-2 melalui fungsi powerOn(). Sebaliknya, apabila koneksi gagal, ESP32-S3 secara otomatis membentuk *Access Point* (AP) ESP32-S3-GAS sehingga pengguna dapat memasukkan SSID dan *password* melalui *captive portal*. Setelah koneksi berhasil diperoleh, sistem berpindah menuju *state* WARMUP.

3. State WARMUP

   Setelah sensor memperoleh catu daya, sistem melakukan proses pemanasan (*warm-up*) selama sekitar 60 detik agar elemen pemanas pada sensor MQ-2 mencapai kondisi kerja yang stabil. Selama tahap ini, pembacaan sensor tetap dilakukan sebagai proses pemantauan, namun belum digunakan untuk menentukan kondisi bahaya. LED berkedip setiap 500 ms dan buzzer berbunyi singkat setiap 5 detik sebagai indikator bahwa proses inisialisasi masih berlangsung. Setelah waktu pemanasan selesai, sistem berpindah ke *state* IDLE.

4. State IDLE

   Pada kondisi ini sistem berada dalam keadaan siaga dan mulai melakukan pembacaan sensor secara periodik setiap satu detik. Nilai ADC hasil pembacaan dibandingkan dengan nilai ambang GAS_THRESHOLD_SOFT untuk menentukan ada atau tidaknya indikasi kebocoran gas. Selama kondisi masih aman, LED dan buzzer berada dalam keadaan tidak aktif. Apabila nilai sensor melampaui ambang batas, sistem berpindah ke *state* DANGER.

5. State DANGER

   State ini menunjukkan bahwa sistem telah mendeteksi indikasi kebocoran gas. ESP32-S3 segera mengaktifkan LED dan buzzer dengan pola kedip cepat sebagai peringatan lokal. Pada saat yang sama, sistem membuat sebuah *task* baru menggunakan FreeRTOS melalui fungsi xTaskCreatePinnedToCore() untuk mengirimkan notifikasi ke Telegram secara asinkron tanpa menghambat proses pembacaan sensor. Sistem akan tetap berada pada kondisi DANGER hingga nilai sensor turun di bawah batas GAS_THRESHOLD_SOFT - HYSTERESIS. Penerapan algoritma histeresis ini bertujuan mencegah perpindahan *state* yang terlalu sering akibat fluktuasi nilai sensor di sekitar ambang batas. Setelah kondisi kembali aman, sistem berpindah kembali ke *state* IDLE dan melanjutkan proses pemantauan.

   Dengan penerapan *Finite State Machine*, setiap tahapan operasi sistem memiliki alur yang jelas mulai dari proses inisialisasi, konfigurasi jaringan, pemanasan sensor, pemantauan kondisi lingkungan, hingga penanganan keadaan darurat. Pendekatan ini membuat sistem lebih mudah dipelihara, meningkatkan keandalan proses deteksi, serta memastikan proses pemantauan sensor, pengendalian aktuator, dan pengiriman notifikasi Telegram dapat berjalan secara konsisten tanpa saling mengganggu.