

REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA



AIR QUALITY AND MACHINE

SAFETY MONITORING

SYSTEM FOR FACTORIES

GROUP 3

Muhammad Nadhif Fasichul Ilmi	2206813416
Bintang Siahaan	2206024322
Hanif Nur Ilham Sanjaya	2206059692
Zikri Zulfa Azhim	2206028390

PREFACE

Segala puji dan syukur kami panjatkan ke hadirat Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan laporan proyek akhir praktikum *Sistem Waktu Nyata dan IoT 2024* ini dengan baik. Proyek ini dirancang sebagai respons terhadap kebutuhan mendesak akan solusi yang dapat meningkatkan keselamatan kerja dan kualitas udara di lingkungan pabrik, yang menjadi tantangan utama dalam era industrialisasi modern.

Proyek "*Air Quality and Machine Safety Monitoring System for Factories*" bertujuan untuk menciptakan sistem pemantauan real-time yang mengintegrasikan sensor gas berbahaya seperti CO, CO₂, NH₃, dan CH₄ dengan sensor jarak untuk mendeteksi keberadaan karyawan di zona risiko. Data yang diperoleh diolah melalui mikrokontroler ESP32 dan dikirimkan ke platform Blynk untuk memantau kondisi secara langsung serta memberikan fitur pengaturan ambang batas gas berbahaya. Sistem ini juga dilengkapi dengan mekanisme otomatis seperti peringatan melalui buzzer dan LED, hingga mematikan mesin saat kondisi berbahaya terdeteksi.

Kami berharap laporan ini memberikan gambaran komprehensif mengenai pengembangan sistem, mulai dari teknologi yang diterapkan hingga manfaatnya dalam menciptakan lingkungan kerja yang lebih aman dan efisien. Semoga proyek ini dapat menjadi langkah awal yang mendorong penerapan teknologi serupa untuk mendukung keselamatan dan keberlanjutan di berbagai sektor industri.

Depok, December 09, 2024

Group 03

TABLE OF CONTENTS

CHAPTER 1.....	3
INTRODUCTION.....	3
1.1 PROBLEM STATEMENT.....	3
1.2 PROPOSED SOLUTION.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	6
1.5 TIMELINE AND MILESTONES.....	7
CHAPTER 2.....	8
IMPLEMENTATION.....	8
2.1 HARDWARE DESIGN AND SCHEMATIC.....	8
2.2 SOFTWARE DEVELOPMENT.....	9
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	25
CHAPTER 3.....	26
TESTING AND EVALUATION.....	26
3.1 TESTING.....	26
3.2 RESULT.....	27
3.3 EVALUATION.....	32
CHAPTER 4.....	34
CONCLUSION.....	34

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Lingkungan pabrik merupakan salah satu area kerja yang memiliki risiko tinggi terhadap kesehatan dan keselamatan karyawan. Salah satu ancaman utama adalah meningkatnya polusi udara yang dihasilkan dari proses produksi, seperti emisi gas berbahaya, termasuk karbon monoksida (CO), karbon dioksida (CO₂), ammonia (NH₃), dan metana (CH₄). Paparan berlebihan terhadap gas-gas tersebut dapat menyebabkan berbagai masalah kesehatan, mulai dari gangguan pernapasan hingga keracunan akut. Dalam jangka panjang, hal ini dapat menurunkan produktivitas karyawan dan mengakibatkan absensi yang tinggi, sehingga berdampak negatif pada efisiensi operasional pabrik.

Selain risiko kesehatan, kedekatan karyawan dengan mesin berbahaya di area produksi juga menjadi salah satu faktor yang meningkatkan kemungkinan terjadinya kecelakaan kerja. Mesin yang beroperasi pada kecepatan tinggi atau suhu ekstrem dapat menimbulkan risiko cedera serius jika karyawan tidak berhati-hati atau tidak memiliki peringatan yang memadai. Kejadian seperti ini tidak hanya membahayakan keselamatan individu, tetapi juga dapat mengganggu alur produksi secara keseluruhan, yang pada akhirnya mempengaruhi target produksi dan kualitas output pabrik.

Dalam konteks tersebut, diperlukan solusi teknologi yang mampu mendeteksi tingkat polusi udara serta memantau jarak karyawan dari mesin berbahaya secara real-time. Sistem ini harus mampu memberikan peringatan dini ketika ambang batas gas berbahaya terlampaui atau ketika karyawan memasuki zona risiko. Dengan demikian, perusahaan dapat memitigasi potensi bahaya, menjaga kesehatan dan keselamatan kerja, serta memastikan efisiensi dan kontinuitas operasional tetap terjaga.

Penerapan sistem ini juga akan mendukung perusahaan dalam mematuhi regulasi keselamatan kerja dan lingkungan, sekaligus meningkatkan citra perusahaan sebagai entitas yang peduli terhadap kesejahteraan karyawannya. Oleh karena itu, integrasi teknologi deteksi kualitas udara dan pemantauan jarak karyawan merupakan langkah penting dalam menciptakan lingkungan kerja yang aman dan produktif.

1.2 PROPOSED SOLUTION

Sistem ini dirancang untuk memantau kualitas udara dan keamanan mesin di lingkungan pabrik dengan menggunakan mikrokontroler **ESP32** sebagai pusat pengendali. Sistem ini dilengkapi dengan berbagai sensor yang memungkinkan deteksi real-time dan respons otomatis untuk meningkatkan keselamatan kerja.

Sensor **MQ-135** digunakan untuk mendeteksi konsentrasi gas berbahaya seperti karbon monoksida (CO), karbon dioksida (CO₂), ammonia (NH₃), dan metana (CH₄). Sensor ini terhubung ke ESP32 untuk membaca data kualitas udara yang kemudian dikirimkan secara real-time ke aplikasi **Blynk** melalui koneksi Wi-Fi. Data ini ditampilkan dalam antarmuka interaktif berupa grafik yang mudah dipahami pengguna. Selain itu, sistem ini dilengkapi dengan sensor ultrasonik **SR04** yang digunakan untuk memantau jarak karyawan dari mesin berbahaya. Sensor ini mengidentifikasi apakah karyawan berada di zona aman atau zona risiko. Jika karyawan mendekati area berbahaya, sistem akan memberikan peringatan visual melalui **LED** dan peringatan audio melalui **buzzer**.

Sistem ini juga memiliki fitur otomatisasi yang responsif. Ketika konsentrasi gas melebihi ambang batas yang telah ditentukan, mesin akan dimatikan secara otomatis untuk mencegah potensi bahaya. Pengguna dapat mengatur ambang batas gas berbahaya dan durasi mode **deep sleep** untuk menghemat daya melalui aplikasi Blynk.

Tujuan utama dari proyek ini adalah menciptakan solusi inovatif yang meningkatkan keselamatan dan efisiensi operasional pabrik. Dengan mengintegrasikan teknologi IoT berbasis ESP32, sistem ini memberikan kemampuan pemantauan dan pengendalian jarak jauh yang andal dan user-friendly. Proyek ini dirancang untuk tidak hanya meningkatkan keselamatan kerja, tetapi juga mempromosikan efisiensi energi dan penerapan teknologi berbasis IoT di lingkungan industri. Dengan pengembangan lebih lanjut, sistem ini diharapkan dapat menjadi solusi standar untuk meningkatkan kualitas lingkungan kerja di pabrik.

1.3 ACCEPTANCE CRITERIA

Sistem yang dirancang dianggap berhasil jika memenuhi kriteria berikut :

1. Pembacaan dan Visualisasi Data Gas Berbahaya Secara Real-Time

- Sistem harus mampu membaca data konsentrasi gas berbahaya, yaitu CO, CO₂, NH₃, dan CH₄, menggunakan sensor MQ-135.
- Data yang diperoleh harus dikirim secara real-time ke aplikasi Blynk melalui koneksi Wi-Fi dan ditampilkan dalam format yang mudah dipahami, seperti grafik.
- Pembaruan data di aplikasi Blynk harus terjadi dalam interval waktu yang singkat (misalnya, setiap 5 detik) untuk memastikan data yang ditampilkan selalu terkini.

2. Otomatisasi Penghentian Operasi Mesin

- Sistem harus memonitor konsentrasi gas berbahaya dan membandingkannya dengan ambang batas yang telah ditentukan.
- Jika kadar gas berbahaya melebihi ambang batas, sistem harus secara otomatis mematikan mesin produksi untuk mencegah bahaya lebih lanjut.
- Proses pematiian mesin harus dilakukan dengan cepat, tidak lebih dari 1 detik setelah deteksi melebihi ambang batas.

3. Peringatan untuk Zona Risiko Karyawan

- Sistem harus menggunakan sensor ultrasonik SR04 untuk mendeteksi jarak karyawan dari mesin berbahaya.
- Jika karyawan berada di area 1 (jarak aman), sistem akan mengaktifkan buzzer dan LED sebagai peringatan visual dan audio.
- Jika karyawan memasuki area 2 (jarak berbahaya), sistem harus secara otomatis mematikan mesin untuk menghindari kecelakaan.

4. Pengaturan Ambang Batas dan Mode Deep Sleep Melalui Aplikasi Blynk

- Aplikasi Blynk harus menyediakan fitur untuk mengatur ambang batas konsentrasi gas berbahaya secara manual oleh pengguna.
- Blynk juga harus memungkinkan pengguna mengatur durasi mode deep sleep, yang digunakan untuk menghemat daya sistem saat kondisi lingkungan berada dalam batas aman.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Role 1	Membuat code, menyusun Blynk, membuat rangkaian.	Zikri Zulfa Azhim
Role 2	Membuat code, menyusun Blynk, menyusun PPT, menyusun laporan.	Bintang Siahaan
Role 3	Membuat code, membuat skema rangkaian, menyusun Blynk.	Hanif Nur Ilham Sanjaya
Role 4	Menyusun laporan, menyusun PPT, membuat README.	Muhammad Nadhif Fasichul Ilmi

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

	Week 1 (26 Nov - 2 Des 2023)							Week 2 (3 - 10 Des 2023)							
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T
Hardware Design															
Software Development															
Integration and Testing															
Final Product Testing															
Report Paper and Slides															

CHAPTER 2

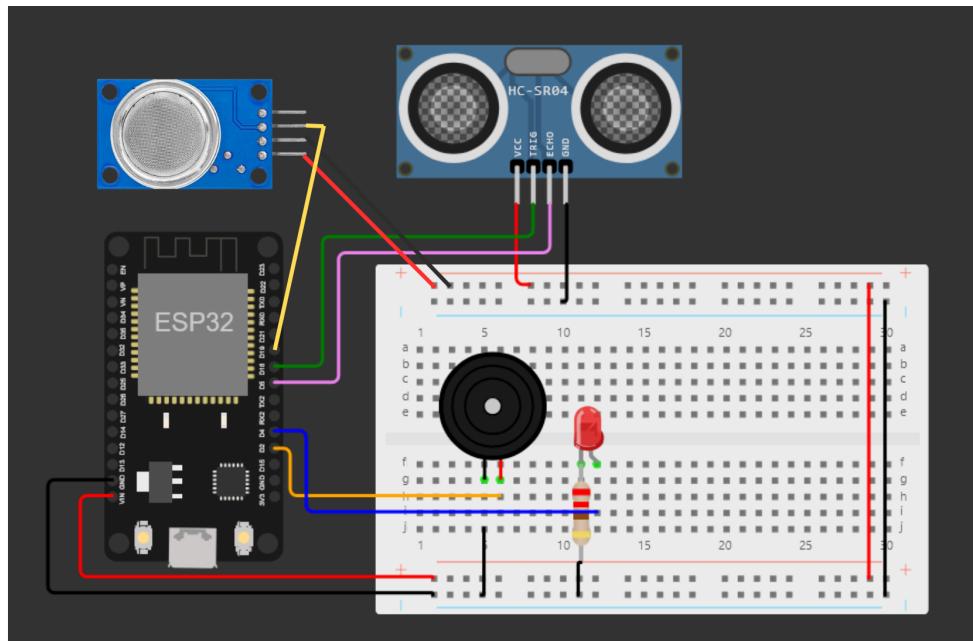
IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Sistem ini dirancang dengan menggunakan mikrokontroler ESP32 sebagai pusat kendali utama. ESP32 bertugas menerima data dari sensor serta memberikan respon otomatis terhadap situasi tertentu. Untuk mendeteksi keberadaan gas berbahaya di lingkungan pabrik, sistem menggunakan sensor MQ-135, yang mampu mengukur konsentrasi berbagai gas seperti karbon monoksida (CO), karbon dioksida (CO₂), ammonia (NH₃), dan metana (CH₄). Data dari sensor ini diproses oleh ESP32 dan divisualisasikan secara real-time melalui aplikasi Blynk, sehingga pengguna dapat memantau kondisi udara secara langsung.

Untuk meningkatkan keselamatan pekerja, sistem juga dilengkapi dengan sensor ultrasonik SR04, yang dirancang untuk memantau jarak antara karyawan dan mesin. Berdasarkan data jarak ini, sistem membagi area kerja menjadi dua zona: zona aman (area 1) dan zona bahaya (area 2). Jika karyawan memasuki zona bahaya, sistem secara otomatis memberikan peringatan berupa nyala LED dan suara buzzer. Jika kondisi semakin berbahaya, ESP32 akan memerintahkan mesin untuk masuk ke mode deep sleep.

Sistem ini mengandalkan mode deep sleep yang diaktifkan langsung oleh ESP32. Saat gas berbahaya melebihi ambang batas atau karyawan berada di area berbahaya, mesin akan masuk ke mode deep sleep berdasarkan perintah ESP32. Durasi mode ini dapat disesuaikan melalui aplikasi Blynk, sehingga pengguna memiliki fleksibilitas untuk mengatur waktu penghentian mesin sesuai kebutuhan. Desain ini tidak hanya sederhana tetapi juga hemat daya, memastikan keberlanjutan operasional di lingkungan pabrik yang sibuk.



Gambar 2.1.1. Skema Rangkaian

2.2 SOFTWARE DEVELOPMENT

Perangkat lunak untuk sistem ini dirancang agar mudah digunakan dan dapat diakses oleh pengguna dengan memanfaatkan aplikasi Blynk sebagai platform utama. Aplikasi Blynk digunakan untuk memvisualisasikan data gas secara real-time, memberikan pengguna gambaran langsung tentang kondisi kualitas udara di dalam pabrik. Data yang diperoleh dari sensor MQ-135, seperti tingkat konsentrasi CO, CO₂, NH₃, dan CH₄, diolah oleh mikrokontroler ESP32 dan kemudian dikirimkan ke server Blynk melalui koneksi Wi-Fi yang stabil. Hal ini memungkinkan pengguna untuk memantau kondisi udara dari jarak jauh menggunakan perangkat pintar seperti ponsel atau tablet.

Selain memvisualisasikan data, aplikasi Blynk juga memberikan kontrol penuh kepada pengguna untuk menyesuaikan parameter operasional sistem. Pengguna dapat dengan mudah mengatur ambang batas gas berbahaya sesuai dengan kebutuhan spesifik pabrik, sehingga sistem dapat memberikan peringatan atau mengambil tindakan otomatis hanya jika ambang batas tersebut dilampaui. Fungsi ini dirancang agar fleksibel dan responsif terhadap kebutuhan lingkungan kerja yang dinamis.

Aplikasi ini juga memungkinkan pengguna untuk mengontrol durasi mode deep sleep, yang digunakan untuk menghentikan sementara operasi mesin ketika konsentrasi gas

berbahaya melebihi ambang batas maksimal kadar zat atau pekerja berada di area bahaya. Dengan adanya fitur ini, pengguna memiliki fleksibilitas untuk menentukan waktu pemulihan operasi mesin, memastikan sistem bekerja secara efisien tanpa mengganggu produktivitas secara signifikan. Implementasi konektivitas Wi-Fi pada ESP32 menjadi komponen penting dalam perangkat lunak ini, karena memastikan data dapat dikirimkan ke server secara real-time dan memungkinkan kontrol jarak jauh yang handal. Kombinasi fitur ini menjadikan sistem lebih pintar, aman, dan efisien dalam menjaga kualitas udara serta keselamatan kerja di pabrik.

Modul-modul yang Diimplementasikan :

Module 2: Memory Management and Queue

Dalam proyek ini, manajemen memori dan penggunaan queue sangat penting untuk memastikan bahwa data sensor yang dikumpulkan dapat diproses dan dikirimkan secara efisien tanpa kehilangan informasi. Queue digunakan untuk mengirimkan data sensor dari tugas pengumpulan sensor (readSensors) ke tugas pengiriman data (blynkTask), sehingga menjaga alur data yang terstruktur dan terorganisir.

Implementasi Queue :

```
// Struktur untuk menyimpan data sensor
typedef struct {
    float co2_ppm;
    float co_ppm;
    float methane_ppm;
    float ammonia_ppm;
    float distanceCm;
} SensorData_t;

// Deklarasi Queue Handle
QueueHandle_t xSensorQueue;

// Inisialisasi Queue di setup()
void setup() {
    // ...
    // Inisialisasi Queue
    xSensorQueue = xQueueCreate(10, sizeof(SensorData_t)); // Queue dengan kapasitas 10 item
    if (xSensorQueue == NULL) {
        Serial.println("Gagal membuat queue!");
        while (1); // Hentikan eksekusi jika queue gagal dibuat
```

```

    }
    // ...
}

// Mengirim data ke queue dalam fungsi readSensors
void readSensors(void *pvParameters) {
    SensorData_t sensorData;

    for(;;){
        readMQ135();
        readSR04();

        // Mengemas data sensor ke dalam struct
        sensorData.co2_ppm = co2_ppm;
        sensorData.co_ppm = co_ppm;
        sensorData.methane_ppm = methane_ppm;
        sensorData.ammonia_ppm = ammonia_ppm;
        sensorData.distanceCm = distanceCm;

        // Mengirim data ke queue
        if (xQueueSend(xSensorQueue, &sensorData, portMAX_DELAY)
!= pdPASS) {
            Serial.println("Gagal mengirim data ke queue!");
        }

        // Delay non-blocking selama 1 detik
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}

// Menerima data dari queue dalam fungsi blynkTask
void blynkTask(void *pvParameters) {
    SensorData_t receivedData;

    for (;;) {
        // Tunggu hingga data tersedia di queue
        if (xQueueReceive(xSensorQueue, &receivedData,
portMAX_DELAY) == pdPASS) {
            // Mengirim data ke Blynk
            Blynk.virtualWrite(VIRTUAL_CO2_PIN,
receivedData.co2_ppm);
            Blynk.virtualWrite(VIRTUAL_CO_PIN, receivedData.co_ppm);

            Blynk.virtualWrite(VIRTUAL_CH4_PIN,
receivedData.methane_ppm);
            Blynk.virtualWrite(VIRTUAL_NH3_PIN,
receivedData.ammonia_ppm);
            Blynk.virtualWrite(VIRTUAL_DISTANCE_PIN,
receivedData.distanceCm);
        }
    }
}

```

```

// Cek ambang batas gas dan jarak
bool gasExceeded = false;
String warningMessage = "";

if (receivedData.co2_ppm > threshold_co2) {
    warningMessage += "Peringatan: Kadar CO2 melebihi batas aman!\n";
    gasExceeded = true;
}
if (receivedData.co_ppm > threshold_co) {
    warningMessage += "Peringatan: Kadar CO melebihi batas aman!\n";
    gasExceeded = true;
}
if (receivedData.methane_ppm > threshold_ch4) {
    warningMessage += "Peringatan: Kadar Methane (CH4) melebihi batas aman!\n";
    gasExceeded = true;
}
if (receivedData.ammonia_ppm > threshold_nh3) {
    warningMessage += "Peringatan: Kadar Ammonia (NH3) melebihi batas aman!\n";
    gasExceeded = true;
}

if (receivedData.distanceCm <= safe_distance) {
    warningMessage += "Peringatan: Pegawai mendekat ke mesin!\n";
    gasExceeded = true;
}

if (gasExceeded) {
    Serial.println(warningMessage);
    Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, warningMessage);
    // Aksi: Mematikan mesin melalui deep sleep
    enterDeepSleep();
} else {
    Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, "Semua parameter dalam batas aman.");
}
}
// Tambahkan delay kecil untuk menghindari penggunaan CPU yang berlebihan
vTaskDelay(pdMS_TO_TICKS(100));
}
}

```

Cara Kerja:

Pada modul ini, queue digunakan untuk mengelola pengiriman data sensor dari tugas readSensors ke tugas blynkTask. Berikut adalah langkah-langkah detailnya:

- Definisi Struktur Data (SensorData_t): Struktur ini mengemas semua data sensor yang diperlukan, termasuk konsentrasi gas (CO₂, CO, CH₄, NH₃) dan jarak yang diukur oleh sensor SR04.
- Deklarasi dan Inisialisasi Queue (xSensorQueue):
 - Queue dideklarasikan secara global dan diinisialisasi di dalam fungsi setup(). Queue dibuat dengan kapasitas 10 item, memungkinkan penyimpanan beberapa paket data sensor sekaligus.
 - Jika pembuatan queue gagal, program akan berhenti dengan menampilkan pesan error di Serial Monitor.
- Mengirim Data ke Queue dalam readSensors:
 - Tugas readSensors membaca data dari sensor MQ-135 dan SR04.
 - Data yang dibaca dikemas ke dalam struktur SensorData_t dan dikirimkan ke queue menggunakan xQueueSend.
 - Setelah mengirim data, tugas ini menunda eksekusi selama 1 detik menggunakan vTaskDelay untuk delay non-blocking.
- Menerima Data dari Queue di blynkTask:
 - Tugas blynkTask menunggu data tersedia di queue menggunakan xQueueReceive.
 - Setelah menerima data, tugas ini mengirimkan data tersebut ke platform Blynk melalui virtual pins yang telah ditentukan.
 - Selain itu, tugas ini memeriksa apakah ada ambang batas gas atau jarak yang terlampaui. Jika ada, pesan peringatan dikirimkan ke Blynk dan mesin dimatikan melalui fungsi enterDeepSleep.
 - Tugas ini juga menambahkan delay kecil untuk menghindari penggunaan CPU yang berlebihan.

Module 4: Software Timer & Hardware Interrupts

Pada proyek ini, pengelolaan waktu dan interupsi perangkat keras sangat penting untuk mengoptimalkan konsumsi daya dan memastikan responsivitas sistem. Implementasi

deep sleep menggunakan software timer memungkinkan perangkat untuk beristirahat dalam jangka waktu tertentu dan bangun secara otomatis, sedangkan interupsi perangkat keras memastikan bahwa perangkat dapat merespons peristiwa eksternal secara cepat.

Implementasi Deep Sleep:

```
// Fungsi untuk memasuki mode deep sleep
void enterDeepSleep() {
    // Jalankan timer countdown
    for (int remainingTime = sleep_time; remainingTime >= 0;
remainingTime--) {
        Blynk.virtualWrite(VIRTUAL_SLEEP_COUNTDOWN,
remainingTime); // Kirim waktu tersisa ke Virtual Pin V3
        Serial.print("Waktu deep sleep tersisa: ");
        Serial.println(remainingTime);

        vTaskDelay(pdMS_TO_TICKS(1000)); // Tunggu 1 detik
sebelum memperbarui waktu
    }

    // Masuk ke mode deep sleep setelah timer selesai
    Serial.println("Memasuki deep sleep...");
    Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, "Memasuki deep
sleep...");
    esp_sleep_enable_timer_wakeup(sleep_time * 1000000); // Set
deep sleep timer
    esp_deep_sleep_start(); // Masuk ke deep sleep
}
```

Cara Kerja:

Pada modul ini, deep sleep diimplementasikan untuk menghemat daya dan mengelola waktu tidur perangkat secara efisien. Berikut adalah penjelasan detailnya:

- Countdown sebelum Deep Sleep:
 - Fungsi enterDeepSleep menjalankan loop countdown yang menampilkan waktu tersisa sebelum perangkat memasuki mode deep sleep.
 - Setiap detik, nilai remainingTime dikirimkan ke Virtual Pin VIRTUAL_SLEEP_COUNTDOWN dan ditampilkan di Serial Monitor.
 - vTaskDelay(pdMS_TO_TICKS(1000)); digunakan untuk delay non-blocking selama 1 detik, memastikan scheduler FreeRTOS tidak terblokir.
- Mengaktifkan dan Memasuki Deep Sleep:

- Setelah countdown selesai, perangkat mengaktifkan timer wakeup menggunakan esp_sleep_enable_timer_wakeup, yang akan membangunkan perangkat setelah sleep_time detik.
- Pesan "Memasuki deep sleep..." dikirimkan ke Virtual Pin VIRTUAL_MESSAGE_PIN dan ditampilkan di Serial Monitor.
- Perangkat kemudian memasuki mode deep sleep dengan memanggil esp_deep_sleep_start(), yang menghentikan operasi perangkat hingga timer wakeup terpenuhi.

Module 5: Deadlock & Priority Inversion

Dalam sistem multitasking seperti proyek IoT ini, manajemen sinkronisasi antar tugas sangat penting untuk menghindari kondisi balapan, deadlock, dan priority inversion. Implementasi semaphore memastikan bahwa tugas-tugas saling berkoordinasi dengan baik tanpa mengganggu eksekusi satu sama lain, menjaga stabilitas dan keandalan sistem.

Implementasi Semaphore:

```
// Deklarasi Binary Semaphore
SemaphoreHandle_t xSemaphoreDataReady;

// Inisialisasi Semaphore di setup()
void setup() {
    // ...
    // Inisialisasi Semaphore
    xSemaphoreDataReady = xSemaphoreCreateBinary();
    if (xSemaphoreDataReady == NULL) {
        Serial.println("Gagal membuat binary semaphore!");
        while (1); // Hentikan eksekusi jika semaphore gagal
dibuat
    }
    // ...
}

// Memberi sinyal dari fungsi readSensors
void readSensors(void *pvParameters) {
    SensorData_t sensorData;

    for(;;){
        readMQ135();
        readSR04();
    }
}
```

```

    // Mengemas data sensor ke dalam struct
    sensorData.co2_ppm = co2_ppm;
    sensorData.co_ppm = co_ppm;
    sensorData.methane_ppm = methane_ppm;
    sensorData.ammonia_ppm = ammonia_ppm;
    sensorData.distanceCm = distanceCm;

    // Mengirim data ke queue
    if (xQueueSend(xSensorQueue, &sensorData, portMAX_DELAY)
!= pdPASS) {
        Serial.println("Gagal mengirim data ke queue!");
    }

    // Memberi sinyal bahwa data sensor telah diperbarui
    xSemaphoreGive(xSemaphoreDataReady);

    // Delay non-blocking selama 1 detik
    vTaskDelay(pdMS_TO_TICKS(1000));
}

// Menunggu sinyal dalam fungsi blynkTask
void blynkTask(void *pvParameters) {
    SensorData_t receivedData;

    for (;;) {
        // Tunggu hingga semaphore tersedia
        if (xSemaphoreTake(xSemaphoreDataReady, portMAX_DELAY))
{
            // Menerima data dari queue dan memprosesnya
            if (xQueueReceive(xSensorQueue, &receivedData, 0) ==
pdPASS) {
                // Mengirim data ke Blynk
                Blynk.virtualWrite(VIRTUAL_CO2_PIN,
receivedData.co2_ppm);
                Blynk.virtualWrite(VIRTUAL_CO_PIN, receivedData.co_ppm);

                Blynk.virtualWrite(VIRTUAL_CH4_PIN,
receivedData.methane_ppm);
                Blynk.virtualWrite(VIRTUAL_NH3_PIN,
receivedData.ammonia_ppm);
                Blynk.virtualWrite(VIRTUAL_DISTANCE_PIN,
receivedData.distanceCm);

                // Cek ambang batas gas dan jarak
                bool gasExceeded = false;
                String warningMessage = "";

                if (receivedData.co2_ppm > threshold_co2) {

```

```

        warningMessage += "Peringatan: Kadar CO2 melebihi
batas aman!\n";
        gasExceeded = true;
    }
    if (receivedData.co_ppm > threshold_co) {
        warningMessage += "Peringatan: Kadar CO melebihi
batas aman!\n";
        gasExceeded = true;
    }
    if (receivedData.methane_ppm > threshold_ch4) {
        warningMessage += "Peringatan: Kadar Methane (CH4)
melebihi batas aman!\n";
        gasExceeded = true;
    }
    if (receivedData.ammonia_ppm > threshold_nh3) {
        warningMessage += "Peringatan: Kadar Ammonia (NH3)
melebihi batas aman!\n";
        gasExceeded = true;
    }

    if (receivedData.distanceCm <= safe_distance) {
        warningMessage += "Peringatan: Pegawai mendekat ke
mesin!\n";
        gasExceeded = true;
    }

    if (gasExceeded) {
        Serial.println(warningMessage);
        Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN,
warningMessage);
        // Aksi: Mematikan mesin melalui deep sleep
        enterDeepSleep();
    } else {
        Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, "Semua
parameter dalam batas aman.");
    }
}
}
}
// Tambahkan delay kecil untuk menghindari penggunaan
CPU yang berlebihan
vTaskDelay(pdMS_TO_TICKS(100));
}
}
}

```

Cara Kerja:

Pada modul ini, semaphore digunakan untuk mengelola sinkronisasi antara tugas `readSensors` dan `blynkTask`, serta untuk menghindari deadlock dan priority inversion. Berikut adalah penjelasan detailnya:

- Deklarasi dan Inisialisasi Semaphore (xSemaphoreDataReady):
 - Semaphore dideklarasikan secara global menggunakan SemaphoreHandle_t.
 - Di dalam fungsi setup(), semaphore diinisialisasi menggunakan xSemaphoreCreateBinary(). Jika inisialisasi gagal, program akan berhenti dengan menampilkan pesan error di Serial Monitor.
- Memberi Sinyal dari readSensors:
 - Setelah tugas readSensors membaca data dari sensor MQ-135 dan SR04 serta mengirimkannya ke queue, tugas ini memberi sinyal bahwa data siap dengan memanggil xSemaphoreGive(xSemaphoreDataReady).
 - Semaphore ini berfungsi sebagai tanda bahwa blynkTask dapat memproses data yang telah diperbarui.
- Menunggu Sinyal dalam blynkTask:
 - Tugas blynkTask menunggu hingga semaphore tersedia menggunakan xSemaphoreTake(xSemaphoreDataReady, portMAX_DELAY).
 - Setelah semaphore diambil, tugas ini menerima data dari queue dan memprosesnya untuk dikirim ke platform Blynk.
 - Dengan menggunakan semaphore, kita memastikan bahwa blynkTask hanya memproses data setelah readSensors telah selesai memperbarui data, sehingga menghindari kondisi balapan (race condition).
- Menghindari Deadlock dan Priority Inversion:
 - Dengan menggunakan semaphore binary, kita mengelola sinkronisasi antar tugas secara efisien tanpa menyebabkan deadlock.
 - Semaphore ini memastikan bahwa blynkTask hanya berjalan setelah data siap, sehingga menghindari masalah priority inversion di mana tugas dengan prioritas rendah dapat menghalangi tugas dengan prioritas tinggi.

Module 7: WiFi, HTTP, & MQTT

Koneksi jaringan adalah komponen krusial dalam proyek IoT, memungkinkan perangkat untuk terhubung ke internet dan berkomunikasi dengan platform monitoring. Implementasi Wi-Fi dalam kode ini memungkinkan perangkat untuk terhubung ke jaringan lokal dan mengirim data sensor secara real-time ke platform Blynk melalui protokol HTTP.

Implementasi Koneksi Wi-Fi:

```
// Kredensial Wi-Fi
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Hotspot"; // Ganti dengan SSID Wi-Fi Anda
char pass[] = "n0t33421"; // Ganti dengan Password Wi-Fi
Anda

// Inisialisasi Wi-Fi di setup()
void setup() {
    // Inisialisasi Serial dengan baud rate yang lebih tinggi
    Serial.begin(115200);

    // Inisialisasi PIN
    pinMode(MQ135_PIN, INPUT);
    pinMode(LED_PIN, OUTPUT);
    pinMode(BUZZER_PIN, OUTPUT);
    pinMode(SR04_TRIGGER_PIN, OUTPUT);
    pinMode(SR04_ECHO_PIN, INPUT);

    // Matikan LED dan Buzzer pada awalnya
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);

    // Inisialisasi Queue
    xSensorQueue = xQueueCreate(10, sizeof(SensorData_t)); // Queue dengan kapasitas 10 item
    if (xSensorQueue == NULL) {
        Serial.println("Gagal membuat queue!");
        while (1); // Hentikan eksekusi jika queue gagal dibuat
    }

    // Koneksi ke Wi-Fi
    Serial.println("Menghubungkan ke Wi-Fi...");
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nTerhubung ke Wi-Fi");

    // Koneksi ke Blynk
    Serial.println("Menghubungkan ke Blynk...");
    Blynk.begin(auth, ssid, pass);

    if (Blynk.connected()) {
        Serial.println("Terhubung ke Blynk");
    } else {
        Serial.println("Gagal terhubung ke Blynk");
    }
}
```

```

// Menampilkan alasan wakeup setelah deep sleep
if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_TIMER)
{
    Serial.println("Bangun dari deep sleep");
    Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, "Bangun dari
deep sleep");
} else {
    Serial.println("Power-on atau reset");
}

// Membuat task
xTaskCreate(readSensors, "Sensor Task", 4096, NULL, 1,
NULL);
xTaskCreate(blynkTask, "Blynk Task", 4096, NULL, 1, NULL);
}

```

Cara Kerja:

Pada modul ini, koneksi Wi-Fi diimplementasikan untuk memungkinkan perangkat IoT terhubung ke jaringan lokal dan berkomunikasi dengan platform Blynk. Berikut adalah penjelasan detailnya:

- Deklarasi Kredensial Wi-Fi:
 - Kredensial Wi-Fi (SSID dan password) didefinisikan secara global.
 - Token otentikasi Blynk (auth) juga didefinisikan untuk menghubungkan perangkat ke aplikasi Blynk.
- Inisialisasi Wi-Fi di setup():
 - Fungsi WiFi.begin(ssid, pass) dipanggil untuk memulai proses koneksi ke jaringan Wi-Fi yang ditentukan.
 - Perangkat akan terus mencoba untuk terhubung hingga status Wi-Fi menunjukkan koneksi berhasil (WL_CONNECTED).
 - Selama proses koneksi, titik-titik (.) akan ditampilkan di Serial Monitor setiap setengah detik sebagai indikator progres.
 - Setelah terhubung, pesan "Terhubung ke Wi-Fi" akan ditampilkan di Serial Monitor.
- Koneksi ke Blynk:

- Setelah terhubung ke Wi-Fi, perangkat mencoba untuk terhubung ke platform Blynk menggunakan Blynk.begin(auth, ssid, pass).
- Jika koneksi berhasil, pesan "Terhubung ke Blynk" akan ditampilkan; jika gagal, pesan "Gagal terhubung ke Blynk" akan muncul.
- Koneksi ke Blynk memungkinkan perangkat untuk mengirim dan menerima data melalui aplikasi Blynk secara real-time.
- Menangani Wakeup dari Deep Sleep:
 - Setelah perangkat bangun dari mode deep sleep, fungsi esp_sleep_get_wakeup_cause() digunakan untuk menentukan alasan bangun.
 - Jika bangun disebabkan oleh timer wakeup (ESP_SLEEP_WAKEUP_TIMER), pesan "Bangun dari deep sleep" dikirimkan ke Blynk dan ditampilkan di Serial Monitor.
 - Jika bangun disebabkan oleh power-on atau reset, pesan "Power-on atau reset" akan ditampilkan.

Module 9: IoT Platform - Blynk

Platform Blynk digunakan sebagai antarmuka IoT untuk memonitor dan mengendalikan perangkat secara real-time melalui aplikasi mobile. Implementasi Blynk dalam kode ini memungkinkan pengiriman data sensor ke aplikasi serta pengiriman pesan peringatan dan kontrol perangkat dari jarak jauh.

Implementasi Blynk:

```
// Mengirim data ke Blynk dalam fungsi blynkTask
void blynkTask(void *pvParameters) {
    SensorData_t receivedData;

    for (;;) {
        // Tunggu hingga data tersedia di queue
        if (xQueueReceive(xSensorQueue, &receivedData,
portMAX_DELAY) == pdPASS) {
            // Mengirim data ke Blynk
            Blynk.virtualWrite(VIRTUAL_CO2_PIN,
receivedData.co2_ppm);
            Blynk.virtualWrite(VIRTUAL_CO_PIN, receivedData.co_ppm);

            Blynk.virtualWrite(VIRTUAL_CH4_PIN,
```

```

receivedData.methane_ppm);
    Blynk.virtualWrite(VIRTUAL_NH3_PIN,
receivedData.ammonia_ppm);
    Blynk.virtualWrite(VIRTUAL_DISTANCE_PIN,
receivedData.distanceCm);

    // Cek ambang batas gas dan jarak
    bool gasExceeded = false;
    String warningMessage = "";

    if (receivedData.co2_ppm > threshold_co2) {
        warningMessage += "Peringatan: Kadar CO2 melebihi batas
aman!\n";
        gasExceeded = true;
    }
    if (receivedData.co_ppm > threshold_co) {
        warningMessage += "Peringatan: Kadar CO melebihi batas
aman!\n";
        gasExceeded = true;
    }
    if (receivedData.methane_ppm > threshold_ch4) {
        warningMessage += "Peringatan: Kadar Methane (CH4)
melebihi batas aman!\n";
        gasExceeded = true;
    }
    if (receivedData.ammonia_ppm > threshold_nh3) {
        warningMessage += "Peringatan: Kadar Ammonia (NH3)
melebihi batas aman!\n";
        gasExceeded = true;
    }

    if (receivedData.distanceCm <= safe_distance) {
        warningMessage += "Peringatan: Pegawai mendekat ke
mesin!\n";
        gasExceeded = true;
    }

    if (gasExceeded) {
        Serial.println(warningMessage);
        Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, warningMessage);
        // Aksi: Mematikan mesin melalui deep sleep
        enterDeepSleep();
    } else {
        Blynk.virtualWrite(VIRTUAL_MESSAGE_PIN, "Semua parameter
dalam batas aman.");
    }
}
// Tambahkan delay kecil untuk menghindari penggunaan
CPU yang berlebihan
vTaskDelay(pdMS_TO_TICKS(100));

```

```
}
```

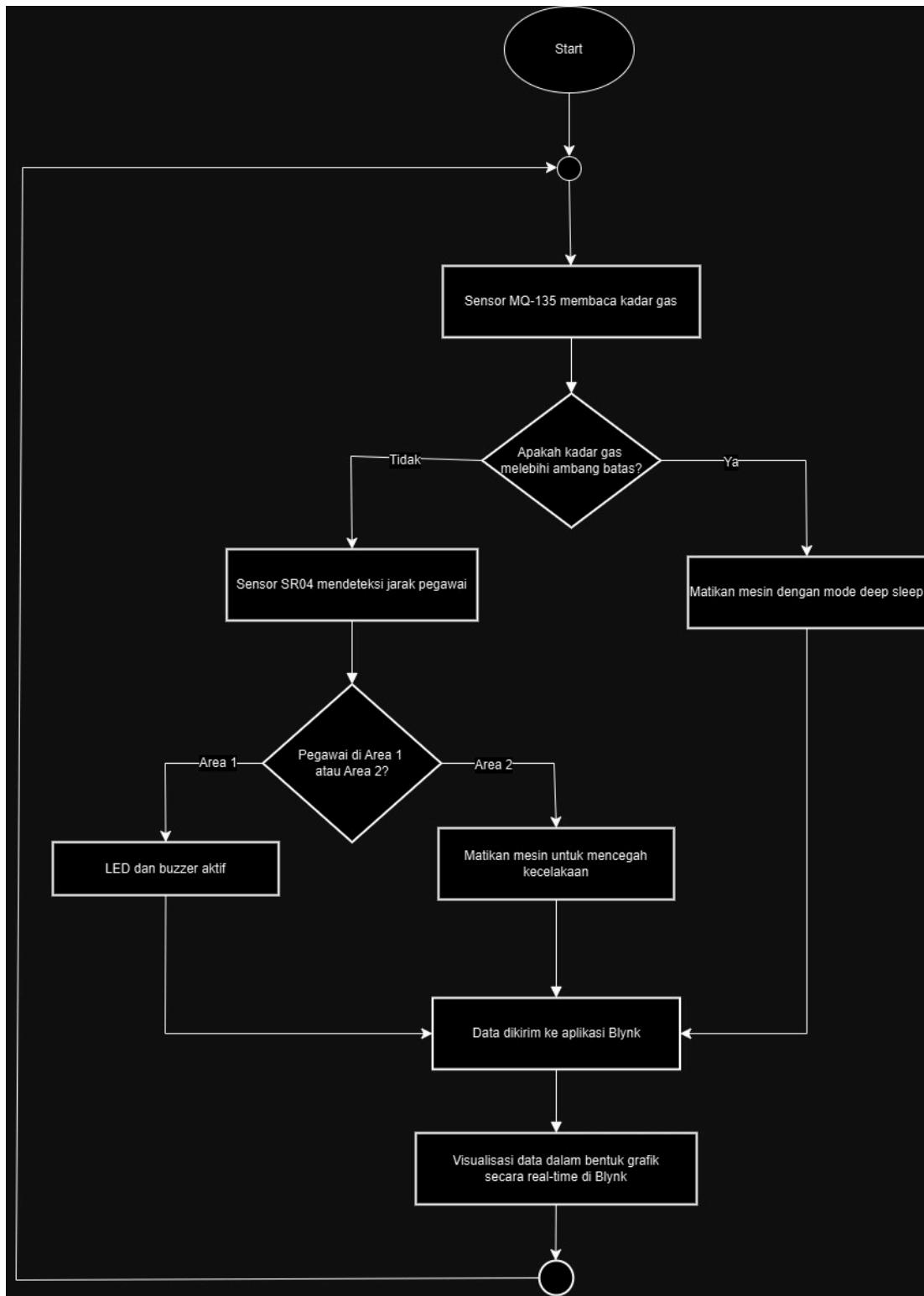
Cara Kerja :

Pada modul ini, platform Blynk digunakan untuk memonitor dan mengendalikan perangkat IoT secara real-time melalui aplikasi mobile. Berikut adalah penjelasan detailnya:

- Inisialisasi Blynk di setup():
 - Fungsi Blynk.begin(auth, ssid, pass) digunakan untuk memulai koneksi ke platform Blynk setelah perangkat terhubung ke Wi-Fi.
 - Jika koneksi berhasil, pesan "Terhubung ke Blynk" akan ditampilkan di Serial Monitor; jika gagal, pesan "Gagal terhubung ke Blynk" akan muncul.
 - Setelah perangkat bangun dari deep sleep, alasan bangun (wakeup cause) dikirimkan ke Blynk melalui Virtual Pin VIRTUAL_MESSAGE_PIN.
- Mengirim Data ke Blynk dalam blynkTask:
 - Tugas blynkTask menerima data sensor dari queue dan mengirimkannya ke aplikasi Blynk melalui Virtual Pins yang telah ditentukan.
 - Data yang dikirim mencakup konsentrasi gas (CO₂, CO, CH₄, NH₃) dan jarak yang diukur oleh sensor SR04.
 - Selain mengirim data sensor, tugas ini juga memeriksa apakah ada ambang batas yang terlampaui. Jika ada, pesan peringatan dikirimkan ke Blynk dan perangkat memasuki mode deep sleep.
 - Jika semua parameter dalam batas aman, pesan "Semua parameter dalam batas aman." dikirimkan ke Blynk.
- Interaksi dengan Aplikasi Blynk:
 - Virtual Pins: Data sensor dikirimkan ke aplikasi Blynk melalui Virtual Pins (V6, V7, V8, V9, V1, V2). Aplikasi Blynk dapat dikonfigurasi untuk menampilkan data ini secara real-time.
 - Peringatan dan Pesan: Aplikasi Blynk menerima pesan peringatan yang muncul ketika ada parameter yang melebihi ambang batas, memungkinkan pengguna untuk mengambil tindakan yang diperlukan.
 - Remote Control: Melalui aplikasi Blynk, pengguna dapat mengatur parameter seperti sleep_time dan safe_distance, yang kemudian diperbarui di perangkat

- melalui fungsi BLYNK_WRITE

Flowchart :



2.3 HARDWARE AND SOFTWARE INTEGRATION

Integrasi perangkat keras dan perangkat lunak merupakan aspek penting dalam sistem ini, menghubungkan sensor, perangkat output, dan mikrokontroler ESP32 dengan aplikasi Blynk secara mulus. Sistem menggunakan sensor MQ-135 untuk mendeteksi konsentrasi gas berbahaya seperti CO, CO₂, NH₃, dan CH₄, serta sensor ultrasonik SR04 untuk memantau jarak karyawan dari mesin. Semua perangkat keras ini terhubung langsung ke ESP32, yang berfungsi sebagai pusat kendali untuk membaca data, memproses informasi, dan mengirimkan hasilnya ke aplikasi Blynk melalui koneksi Wi-Fi.

Pada sisi perangkat lunak, program yang diimplementasikan pada ESP32 dirancang untuk mengatur proses pembacaan data sensor secara real-time. Data mentah yang diperoleh dari sensor akan diolah oleh ESP32 menggunakan logika yang telah ditentukan untuk mendeteksi kondisi bahaya. Misalnya, ketika konsentrasi gas melebihi ambang batas yang ditentukan atau pekerja terdeteksi berada di area berisiko, ESP32 akan segera mengaktifkan perangkat output, seperti LED dan buzzer, untuk memberikan peringatan. Program ini juga dirancang untuk mengontrol aktivasi mode deep sleep secara otomatis sesuai dengan parameter yang telah diatur pengguna melalui aplikasi Blynk.

Selain memproses logika lokal, ESP32 juga bertugas menjaga komunikasi dua arah dengan server Blynk. Data yang telah diproses akan dikirimkan ke aplikasi Blynk untuk divisualisasikan secara real-time. Sebaliknya, perintah dari pengguna, seperti perubahan ambang batas gas atau durasi deep sleep, akan diterima ESP32 dari aplikasi dan diterapkan pada sistem. Integrasi ini memastikan bahwa perangkat keras dan perangkat lunak bekerja secara harmonis, memberikan pengalaman pengguna yang intuitif sekaligus menjamin respons sistem yang cepat terhadap kondisi lingkungan pabrik. Kombinasi dari kedua elemen ini menjadikan sistem lebih efektif dalam menjaga kualitas udara dan keselamatan pekerja di pabrik.

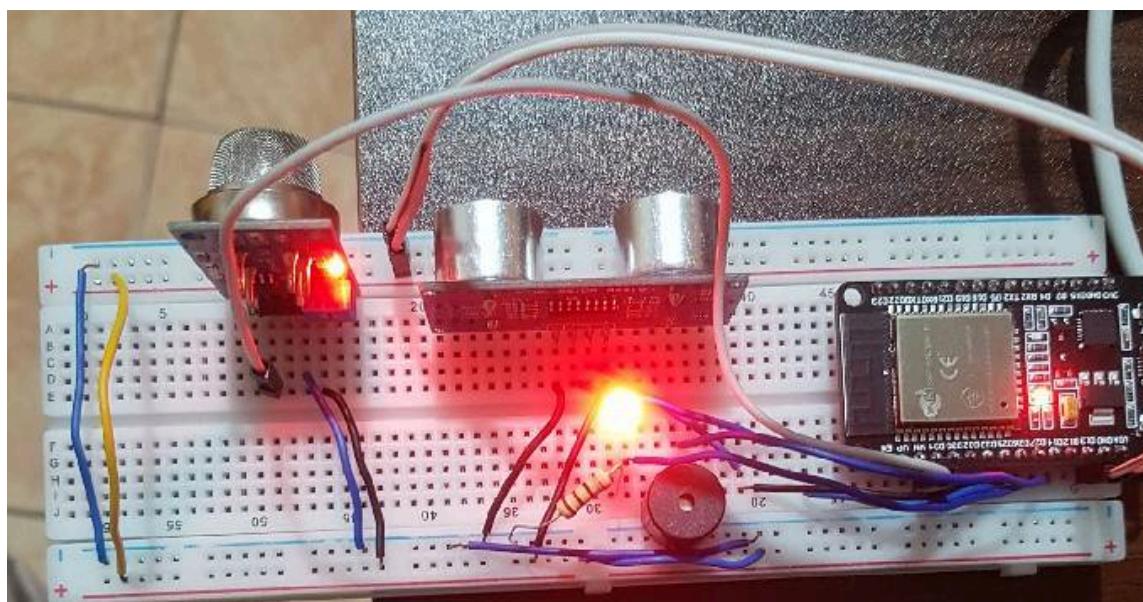
CHAPTER 3

TESTING AND EVALUATION

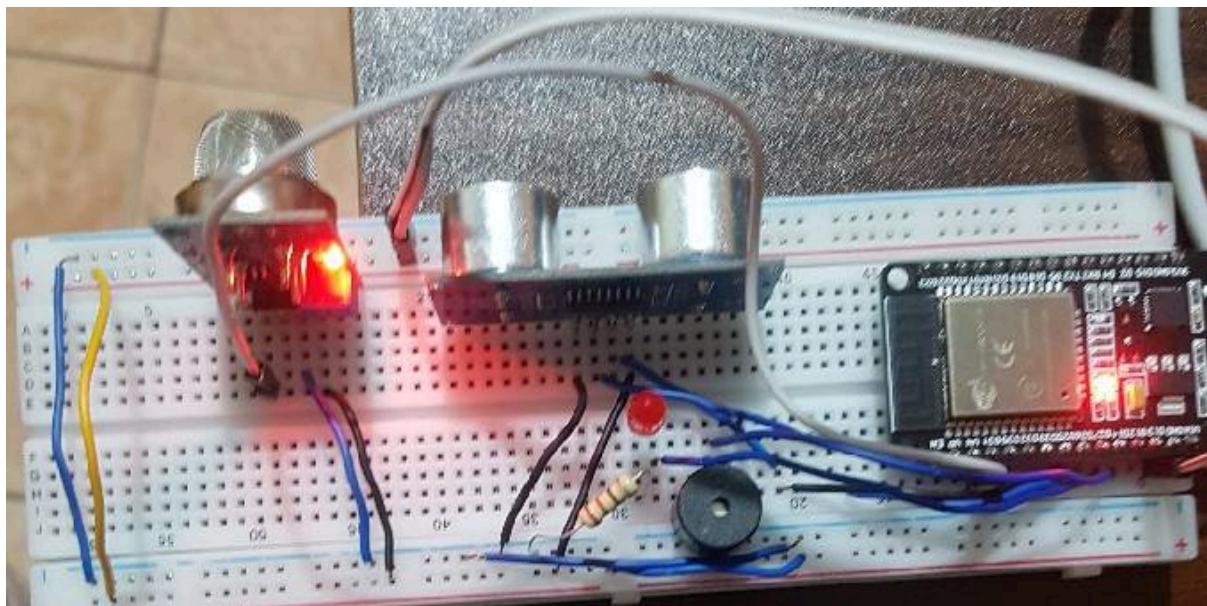
3.1 TESTING

Pada tahap pengujian, dilakukan serangkaian uji untuk memastikan sistem bekerja secara akurat, andal, dan sesuai dengan spesifikasi yang telah ditentukan. Pengujian deteksi gas dilakukan dengan mengekspos sensor MQ-135 pada berbagai konsentrasi gas seperti karbon monoksida (CO), karbon dioksida (CO₂), ammonia (NH₃), dan metana (CH₄). Hasil pengukuran dari sensor dibandingkan dengan nilai referensi untuk memverifikasi keakuratannya dan memastikan bahwa sistem mampu mendeteksi perubahan konsentrasi gas secara real-time. Selain itu, pengujian ini juga membantu mengidentifikasi respons waktu sensor dalam mendeteksi peningkatan atau penurunan konsentrasi gas.

Untuk pengujian pemantauan jarak, sensor HC-SR04 diuji dengan menempatkan objek pada berbagai jarak tertentu dari sensor untuk memastikan sistem dapat mendeteksi zona keselamatan dengan benar. Pengujian ini melibatkan simulasi dua zona keselamatan, yaitu area 1 (jauh) yang mengaktifkan peringatan berupa buzzer dan LED, serta area 2 (dekat) yang secara otomatis mematikan mesin untuk mencegah potensi kecelakaan. Proses ini memastikan sensor ultrasonik mampu mengukur jarak dengan akurat dan memicu respon sistem sesuai dengan logika yang telah dirancang.



Gambar 3.1.1. Area 1



Gambar 3.1.2. Area 2

Pengujian integrasi keseluruhan sistem dilakukan dengan menghubungkan seluruh komponen, termasuk sensor, ESP32, buzzer, LED, dan aplikasi Blynk, pada sebuah rangkaian yang dirancang untuk mensimulasikan kondisi lingkungan industri. Rangkaian ini digunakan untuk menguji komunikasi antar komponen, kecepatan pemrosesan data, serta fungsi keseluruhan sistem secara real-time. Pengujian ini mencakup situasi darurat, seperti peningkatan konsentrasi gas di atas ambang batas atau objek yang terlalu dekat dengan mesin, untuk memastikan sistem merespon sesuai harapan.

3.2 RESULT

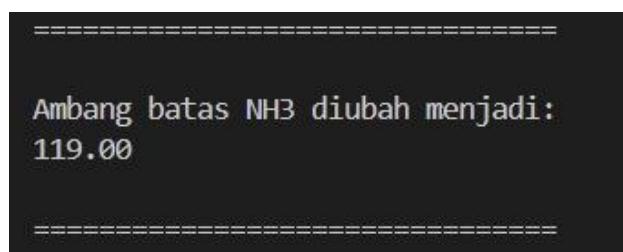
Dari pengujian yang telah dilakukan, sistem menunjukkan hasil yang sangat memuaskan dalam mengintegrasikan pemantauan kualitas udara dan keselamatan mesin. Sensor MQ-135 berhasil mengukur konsentrasi gas secara real-time dengan tingkat akurasi yang tinggi. Hasil pengukuran konsentrasi gas tersebut ditampilkan secara langsung melalui aplikasi Blynk. Fitur ini memberikan kemudahan bagi pengguna untuk memantau kualitas udara secara konsisten dan mengambil langkah preventif jika konsentrasi gas berbahaya

melampaui ambang batas yang telah ditentukan. Sistem ini diharapkan mampu membantu industri dalam menciptakan lingkungan kerja yang lebih aman dan sehat.

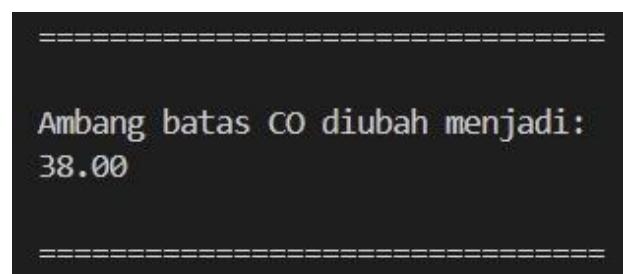
Mekanisme pemantauan jarak berbasis sensor HC-SR04 juga berfungsi sesuai dengan yang diharapkan. Sistem berhasil mendeteksi keberadaan objek di dua area berbeda, yaitu Area 1 dan Area 2. Jika terdeteksi keberadaan objek di salah satu zona tersebut, sistem secara otomatis mengirimkan peringatan kepada pengguna melalui aplikasi dan melakukan pemutusan daya pada mesin. Hal ini bertujuan untuk memastikan keselamatan pekerja di sekitar mesin, mengurangi risiko kecelakaan, serta menjaga efisiensi proses operasional pabrik.

Aplikasi Blynk yang digunakan memberikan antarmuka pengguna yang intuitif dan interaktif. Dengan aplikasi ini, pengguna dapat :

- Memantau data sensor secara real-time, termasuk konsentrasi gas dan keberadaan objek di Area 1 maupun Area 2.
- Mengatur ambang batas konsentrasi gas secara dinamis sesuai dengan kebutuhan.
- Menyesuaikan durasi deep-sleep sistem untuk mengoptimalkan penggunaan daya.



Gambar 3.2.1. Mengubah Ambang batas NH3 dari Blynk



Gambar 3.2.2. Mengubah Ambang batas CO dari Blynk

```
=====
Ambang batas CO2 diubah menjadi:  
14.00  
=====
```

Gambar 3.2.3. Mengubah Ambang batas CO2 dari Blynk

```
=====
Ambang batas CH4 diubah menjadi:  
17.00  
=====
```

Gambar 3.2.4. Mengubah Ambang batas CO2 dari Blynk

```
=====
Sleep time diupdate menjadi:  
20  
=====
```

Gambar 3.2.5. Mengubah durasi deep-sleep dari Blynk

Menunjukkan pengaturan ambang batas konsentrasi gas dan menyesuaikan durasi deep-sleep sistem yang dapat disesuaikan oleh pengguna melalui aplikasi Blynk.

```
MQ-135 - Raw Analog Value: 717
Voltage: 0.58
Sensor Resistance: 4.71
CO2 PPM: 3.51
CO PPM: 1.60
Methane (CH4) PPM: 1.30
Ammonia (NH3) PPM: 0.98
SR04 - Distance (cm): 27.86
Peringatan: Pegawai mendekat ke mesin!
```

Message

Peringatan: Pegawai mendekat ke mesin!

Distance

29 cm

Gambar 3.2.5. Hasil Area 1

Area 1: Memperlihatkan deteksi keberadaan objek di Zona 1 dan tindakan yang diambil oleh sistem.

```
MQ-135 - Raw Analog Value: 659
Voltage: 0.53
Sensor Resistance: 5.21
CO2 PPM: 3.07
CO PPM: 1.40
Methane (CH4) PPM: 1.14
Ammonia (NH3) PPM: 0.85
SR04 - Distance (cm): 15.96
Aksi: Mematikan mesin untuk mencegah kecelakaan.
Waktu deep sleep tersisa: 10
Waktu deep sleep tersisa: 9
Waktu deep sleep tersisa: 8
Waktu deep sleep tersisa: 7
Waktu deep sleep tersisa: 6
```

Message

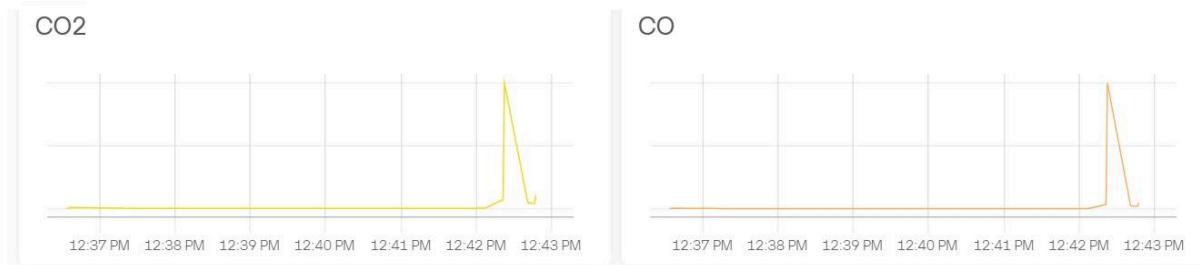
Mematikan mesin untuk mencegah kecelakaan.

Distance

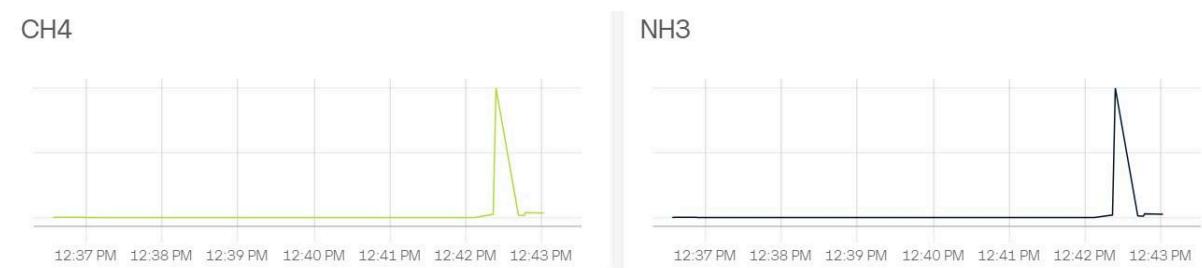
3 cm

Gambar 3.2.6. Hasil Area 2

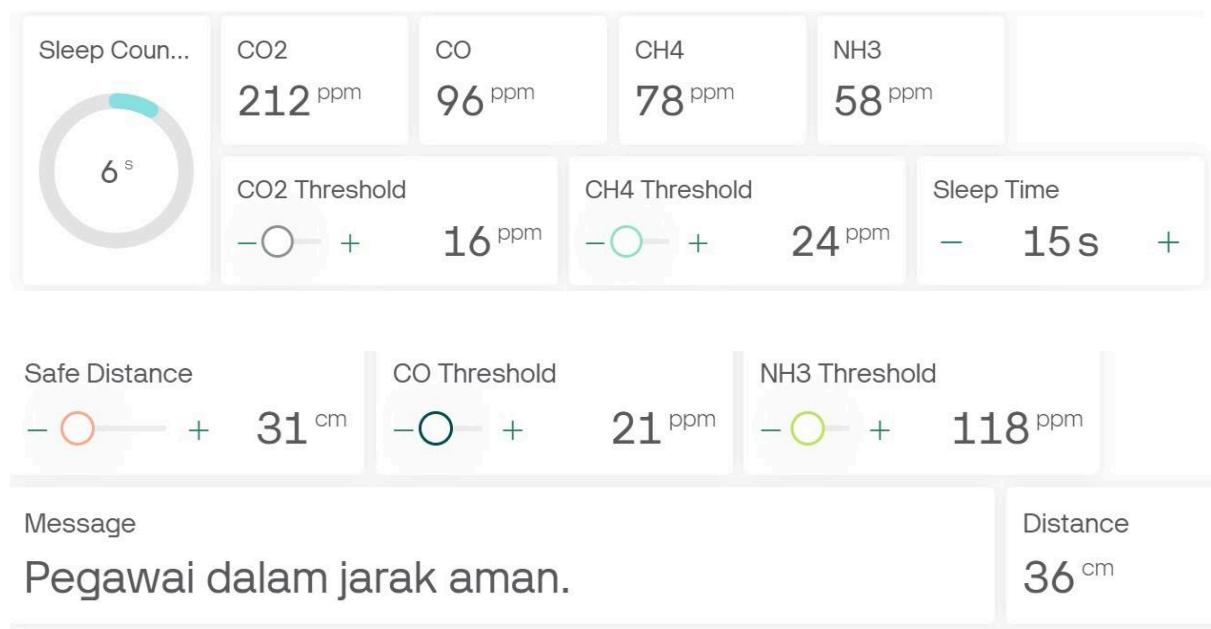
Area 2: Menunjukkan aktivitas sensor di Zona 2 ketika mendeteksi objek, termasuk pengaktifan peringatan dan pemutusan daya mesin.



Gambar 3.2.7. Grafik CO2 dan CO



Gambar 3.2.8. Grafik CH4 dan NH3



Gambar 3.2.8. Data real-time di Blynk

3.3 EVALUATION

Sistem yang dikembangkan berhasil memenuhi semua kriteria penerimaan yang telah ditentukan. Dari hasil pengujian, sistem menunjukkan tingkat keandalan, akurasi, dan kemudahan penggunaan yang tinggi. Fitur yang dapat disesuaikan, seperti pengaturan ambang batas gas dan durasi deep-sleep, menjadikan sistem lebih fleksibel dan mampu beradaptasi dengan berbagai kebutuhan di lingkungan industri yang berbeda.

Beberapa penyesuaian kecil dilakukan selama tahap pengujian untuk meningkatkan performa sistem, termasuk peningkatan sensitivitas sensor ultrasonik agar lebih responsif terhadap jarak objek serta optimasi waktu respons sensor MQ-135 untuk mendeteksi perubahan konsentrasi gas dengan lebih cepat.

Untuk pengembangan di masa depan, sistem ini dapat dilengkapi dengan sensor tambahan untuk memperluas kemampuan deteksi gas berbahaya lainnya, seperti sulfur dioksida (SO₂) atau ozon (O₃). Selain itu, peningkatan skalabilitas sistem juga dapat dilakukan agar dapat diterapkan pada fasilitas industri yang lebih besar dengan jumlah sensor yang lebih banyak dan kemampuan pemantauan terpusat melalui dashboard berbasis web.

Implementasi teknologi tambahan, seperti edge computing, juga dapat meningkatkan efisiensi pemrosesan data dan mengurangi ketergantungan pada koneksi internet.

Sistem ini memiliki potensi besar untuk meningkatkan keselamatan dan efisiensi operasional di berbagai sektor industri, sekaligus memberikan kontribusi yang signifikan terhadap upaya menciptakan lingkungan kerja yang lebih aman dan ramah bagi pekerja.

CHAPTER 4

CONCLUSION

Air Quality and Machine Safety Monitoring System for Factories yang dirancang telah membuktikan kemampuannya dalam menjawab tantangan keselamatan yang sering dihadapi di lingkungan industri. Dengan memanfaatkan teknologi Internet of Things (IoT), sistem ini memungkinkan pemantauan secara real-time terhadap konsentrasi gas berbahaya seperti CO, CO₂, NH₃, dan CH₄, serta pengawasan jarak menggunakan sensor ultrasonik. Data yang diperoleh dari sensor ditampilkan melalui aplikasi Blynk, yang menyediakan antarmuka yang intuitif untuk memantau parameter secara langsung serta mengatur ambang batas yang sesuai dengan kebutuhan lingkungan kerja.

Fitur-fitur unggulan yang ditanamkan pada sistem ini, seperti ambang batas gas yang dapat disesuaikan dan mekanisme keselamatan otomatis berupa pematiian mesin jika parameter tertentu terdeteksi, memberikan kontribusi signifikan terhadap keselamatan pekerja. Dengan adanya mekanisme ini, sistem mampu mencegah terjadinya insiden akibat paparan gas berbahaya atau keberadaan objek yang mendekati zona berbahaya di sekitar mesin. Hal ini secara langsung meningkatkan efisiensi operasional serta menciptakan lingkungan kerja yang lebih aman dan kondusif. Selain itu, hasil pengujian menunjukkan bahwa sistem bekerja dengan tingkat keandalan dan akurasi yang baik. Sensor MQ-135 berhasil mendeteksi berbagai konsentrasi gas dengan respons yang cepat, sementara sensor HC-SR04 mampu mengidentifikasi jarak dengan presisi tinggi. Antarmuka aplikasi Blynk juga memberikan kemudahan bagi pengguna dalam memantau data sensor dan mengontrol parameter sistem, membuatnya lebih praktis untuk diterapkan dalam berbagai skenario industri.

Meskipun sistem ini sudah memenuhi semua kriteria penerimaan, beberapa peluang perbaikan masih dapat dilakukan untuk pengembangan di masa depan. Penambahan sensor tambahan, misalnya untuk mendeteksi gas berbahaya lainnya, dapat memperluas cakupan pemantauan kualitas udara. Selain itu, peningkatan skalabilitas sistem untuk diterapkan di fasilitas industri yang lebih besar, seperti pabrik dengan banyak area kritis, juga menjadi salah satu langkah yang dapat dilakukan agar sistem ini dapat melayani kebutuhan yang lebih kompleks.

Secara keseluruhan, sistem ini telah menunjukkan potensi besar dalam mendukung keamanan dan efisiensi operasional di lingkungan industri. Dengan pemanfaatan teknologi

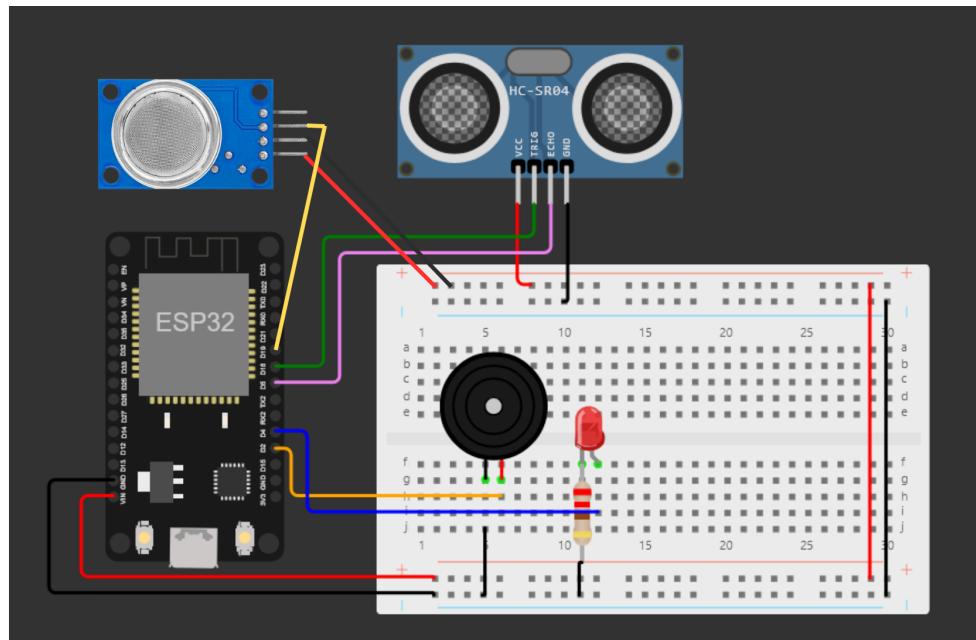
IoT yang inovatif, sistem ini diharapkan dapat terus dikembangkan untuk menjawab berbagai tantangan industri masa depan dan mendukung terciptanya tempat kerja yang lebih aman dan ramah lingkungan.

REFERENCES

- [1] "Memory Management and Queue in IoT Real-Time Systems," Digilabdt.com, 2024. [Online]. Available: <https://learn.digilabdt.com/books/iot-real-time-system/page/module-2-memory-management-queue>. [Accessed: Dec. 10, 2024].
- [2] "Integrated Hardware Garbage Collection for Real-Time Embedded Systems," Andrés Amaya García, University of Bristol, August 2021. [Online]. Available: https://sourcecodeartisan.com/download/phd_thesis.pdf. [Accessed: Dec. 10, 2024].
- [3] Michael Harditya, Muhammad Naufal Faza, "MODUL 4: SOFTWARE TIMER & HARDWARE INTERRUPTS," Digilab, [Online]. Available : <https://emas2.ui.ac.id/mod/resource/view.php?id=2673870> [Accessed:Dec. 10, 2024].
- [4] "Introduction to FreeRTOS Solution to Part 8: Software Timers," Digikey, 2021. [Online]. Available: <https://www.digikey.com/en/maker/projects/introduction-to-rtos-solution-to-part-8-software-timers/0f64cf758da440a29476165a5b2e577e>. [Accessed: Dec. 10, 2024].
- [5] "Deadlock in Multicore Systems," Digilab DTE, 2024. [Online]. Available: <https://learn.digilabdt.com/books/realtime-system-iot/page/module-5-deadlock-multicore-systems>. [Accessed: Dec. 10, 2024].
- [6] "Deadlock and Starvation," Idraya, [Online]. Available: <https://idraya.com/networking/deadlock-and-starvation.html>. [Accessed: Dec. 10, 2024].
- [7] AM, "Module 7: WiFi, HTTP(S), & MQTT(S)| Digilab UI," Digilabdt.com, 2024. <https://learn.digilabdt.com/books/realtime-system-iot/page/module-7-wifi-https-mqtt-s> (accessed Dec. 10, 2024).
- [8] "Module 9: IOT Platform - Blynk" Digilabdt.com, 2024. <https://learn.digilabdt.com/books/realtime-system-iot/page/module-9-iot-platform-blynk> [accessed Dec. 10, 2024].
- [9] Blynk, "Introduction | Blynk Documentation," Blynk.io, Mar. 07, 2024. [Online]. Available: <https://docs.blynk.io/en>. [Accessed: Dec. 10, 2024].
- [10] "ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE | Random Nerd Tutorials," Random Nerd Tutorials, Jul. 20, 2021. <https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/> (accessed Dec. 07, 2024).

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation

