**Assalamualaikum Wr.Wb.**

# Pada kesempatan ini saya akan menjelaskan mengenai proyek akhir mata kuliah Sistem Mikrokontroler dengan dosen pengampu Bpk. Muchamad Rusdan, ST., MT. dengan judul/topik “SISTEM MONITORING SUHU KAMAR JENAZAH”.

Perkenalkan nama saya Devis Pringga NPM 23552012004 Kelas TIF RM 22 CNS.

## 1. Latar Belakang

Ruang pendingin (cold storage) memiliki peranan penting dalam menjaga kualitas dan kesegaran produk yang mudah rusak, seperti bahan makanan, obat-obatan, dan juga jenazah pada kamar Jenazah. Fluktuasi suhu yang tidak terpantau dapat menimbulkan kerusakan, kerugian finansial, serta risiko terhadap kesehatan masyarakat. Oleh karena itu, dibutuhkan sistem pemantauan suhu yang mampu bekerja secara real-time, akurat, serta dapat memberikan notifikasi peringatan saat terjadi anomali suhu.

## 2. Tujuan

* Membangun sistem pemantauan suhu ruang pendingin secara real-time.
* Memberikan peringatan otomatis apabila suhu melebihi atau kurang dari ambang batas yang telah ditentukan.
* Menyediakan visualisasi data suhu melalui dashboard atau aplikasi mobile.

**struktur dan logika** kode ESP32 pada **monitoring suhu kamar jenazah, menampilkan ke LCD, kirim data ke server (webhook.php), dan menerima perintah dari dashboard (command.json)**: menggunakan Wokwi Simulator

* Komponen:
  + Breadboard
  + ESP32
  + Sensor DHT22 (suhu & kelembaban) / 1 buah resistor 10Kohm
  + LCD I2C
  + LED Merah & Hijau / 2 buah resistor 220 ohm
  + Buzzer
  + Kabel Jumper

## 🔰 ****1. Bagian**** #include ****dan**** define

#include <WiFi.h> // Untuk koneksi WiFi

#include <HTTPClient.h> // Untuk kirim request HTTP

#include <Wire.h> // Untuk komunikasi I2C (LCD)

#include <LiquidCrystal\_I2C.h> // Untuk LCD 16x2

#include "DHTesp.h" // Untuk sensor DHT22

#include <ArduinoJson.h> // Untuk parsing JSON dari server

#define DHT\_PIN 15

#define LED\_MERAH 2

#define LED\_HIJAU 27

#define BUZZER\_PIN 14

* Menyiapkan semua library dan pin yang digunakan.

## 🔧 ****2. Variabel Konstanta dan Global****

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

const char\* webhookUrl = ".../webhook.php";

const char\* commandUrl = ".../command.json";

* ssid, password: info WiFi.
* webhookUrl: tempat mengirim data suhu ke server.
* commandUrl: tempat mengambil perintah (mode dan LED) dari dashboard.

DHTesp dht;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

float suhuGlobal = 0;

float humGlobal = 0;

String modeGlobal = "SENSOR";

bool sudahKirim = false;

unsigned long lastWebhookTime = 0;

const unsigned long intervalWebhook = 10000;

* Menyimpan nilai sensor dan kontrol waktu kirim data tiap 10 detik.

## 🚀 ****3. Fungsi**** setup()

void setup() {

Serial.begin(115200);

dht.setup(DHT\_PIN, DHTesp::DHT22);

lcd.init();

lcd.backlight();

pinMode(LED\_MERAH, OUTPUT);

pinMode(LED\_HIJAU, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

// Koneksi WiFi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println(" Terhubung!");

}

* Menghubungkan ke WiFi.
* Menyiapkan LCD dan DHT22.
* Set pin LED dan buzzer sebagai output.

## 🔁 ****4. Fungsi**** loop()

TempAndHumidity data = dht.getTempAndHumidity();

suhuGlobal = data.temperature;

humGlobal = data.humidity;

* Baca data suhu & kelembaban dari sensor DHT22.
* Simpan ke variabel global.

// Tampilkan ke LCD

lcd.setCursor(0, 0); lcd.print("Suhu: "); lcd.print(suhuGlobal, 1);

lcd.setCursor(0, 1); lcd.print("Hum : "); lcd.print(humGlobal, 1);

* Tampilkan suhu & kelembaban di layar LCD.

### 🔁 Kirim Data ke Server

if (millis() - lastWebhookTime > intervalWebhook) {

sendWebhook(suhuGlobal, humGlobal);

lastWebhookTime = millis();

}

* Setiap 10 detik, data suhu & kelembaban dikirim ke webhook.php.

### 📦 Mode SENSOR: Kontrol Otomatis

if (modeGlobal == "SENSOR") {

if (suhuGlobal > 8) {

digitalWrite(LED\_MERAH, HIGH); // Bahaya

digitalWrite(LED\_HIJAU, LOW);

digitalWrite(BUZZER\_PIN, HIGH); delay(500);

digitalWrite(BUZZER\_PIN, LOW); delay(500);

} else {

digitalWrite(LED\_MERAH, LOW); // Aman

digitalWrite(LED\_HIJAU, HIGH);

}

}

* Suhu > 8°C: nyalakan LED merah + buzzer
* Suhu aman: LED hijau menyala

## 🌐 ****5. Fungsi**** sendWebhook()

void sendWebhook(float suhu, float hum) {

HTTPClient http;

String url = String(webhookUrl) + "?suhu=" + String(suhu, 1) + "&humidity=" + String(hum, 1);

http.begin(url);

int response = http.GET();

// Log hasil

http.end();

}

* Mengirim suhu dan kelembaban ke webhook.php lewat HTTP GET.
* Digunakan untuk menyimpan data ke file data.json dan trigger WhatsApp alert.

## 🔁 ****6. Fungsi**** checkCommand()

void checkCommand() {

HTTPClient http;

http.begin(commandUrl);

int httpResponseCode = http.GET();

if (httpResponseCode == 200) {

String payload = http.getString();

StaticJsonDocument<256> doc;

DeserializationError error = deserializeJson(doc, payload);

* Mengambil isi file command.json dari server.
* File ini berisi:

json

CopyEdit

{

"mode": "DASHBOARD",

"led": "ON"

}

modeGlobal = doc["mode"] | "SENSOR";

String ledStatus = doc["led"] | "OFF";

* modeGlobal dan status LED disesuaikan dengan perintah dari dashboard.

### 🧠 Mode DASHBOARD: Kontrol Manual

if (modeGlobal == "DASHBOARD") {

if (ledStatus == "ON") {

if (suhuGlobal <= 8) {

digitalWrite(LED\_HIJAU, HIGH);

digitalWrite(LED\_MERAH, LOW);

} else {

digitalWrite(LED\_HIJAU, LOW);

digitalWrite(LED\_MERAH, HIGH);

}

} else {

digitalWrite(LED\_HIJAU, LOW);

digitalWrite(LED\_MERAH, LOW);

}

}

* Saat **mode = DASHBOARD**:
  + LED mengikuti perintah dari dashboard (ON/OFF)
  + Tidak mengikuti sensor secara otomatis.

## ✅ ****Rangkuman Logika Kerja****

| **Bagian** | **Fungsi** |
| --- | --- |
| setup() | Inisialisasi WiFi, sensor, LCD, dan pin output |
| loop() | Baca sensor, tampilkan LCD, kirim ke server, kontrol LED |
| sendWebhook() | Kirim suhu & kelembaban ke webhook.php setiap 10 detik |
| checkCommand() | Ambil perintah dari command.json (mode & LED) |
| Mode SENSOR | ESP32 kontrol otomatis LED & buzzer sesuai suhu |
| Mode DASHBOARD | ESP32 mengikuti perintah dari dashboard web |

**DEMO PROYEK**

**skema alur kerja proyek IoT Monitoring Suhu kamar jenazah**, mulai dari simulasi di **Wokwi**, proses di **ESP32**, hingga tampil di **dashboard web** dan terkirim ke **WhatsApp via Twilio**.

## 🔧 ****1. Simulasi Sensor di Wokwi****

* Komponen:
  + ESP32
  + Sensor DHT22 (suhu & kelembaban)
  + LCD I2C
  + LED Merah & Hijau
  + Buzzer

ESP32 membaca suhu & kelembaban dari DHT22 setiap ~0.5–1 detik.

## 📡 ****2. Koneksi ke WiFi (Wokwi Virtual)****

* ESP32 tersambung ke Wokwi-GUEST (simulasi).
* Setelah tersambung, ESP32 bisa:
  + Kirim data ke server (webhook.php)
  + Ambil perintah terbaru dari command.json

## 🌐 ****3. Kirim Data ke Server (Webhook)****

sendWebhook(suhu, humidity)

→ webhook.php?suhu=...&humidity=...

* Data dikirim setiap 10 detik via HTTP GET.
* webhook.php akan:
  + Simpan data ke data.json
  + Jika suhu > 8°C:
    - Kirim pesan WhatsApp via Twilio

## 💬 ****4. Kirim WhatsApp via Twilio****

* webhook.php memanggil API Twilio:

arduino

CopyEdit

https://api.twilio.com/.../Messages.json

* Isi pesan:

yaml

CopyEdit

🚨 ALERT: Suhu Tinggi!

🌡 Suhu: 9.2°C

📍 Lokasi: Kamar Jenazah

* Dikirim ke WhatsApp penerima (misalnya: kamu sendiri).

## 🧠 ****5. ESP32 Ambil Perintah Dashboard****

checkCommand()

→ Baca file command.json

* File ini diubah oleh pengguna melalui dashboard/index.php saat memilih:
  + Mode: SENSOR atau DASHBOARD
  + LED: ON atau OFF
* Berdasarkan isi command.json, ESP32:
  + Mengatur LED merah/hijau
  + Mengabaikan sensor jika mode DASHBOARD

## 📊 ****6. Dashboard Menampilkan Data****

* dashboard/index.php:
  + Membaca isi data.json untuk tampilkan:
    - Suhu, kelembaban
    - Status (Aman / Bahaya)
    - Grafik real-time (Chart.js)
    - Log terakhir (opsional)
  + Bisa memilih mode kontrol (DASHBOARD/SENSOR)
  + Bisa nyalakan/matikan LED secara manual saat DASHBOARD

## 🔁 ****Alur Lengkap Proyek****

graph TD

A1[Wokwi (Simulasi ESP32 + Sensor DHT22)] --> A2[ESP32 Baca Suhu & Kelembaban]

A2 -->|tiap 10 detik| B1[sendWebhook() ➜ webhook.php]

B1 -->|simpan JSON| C1[data.json (dashboard)]

B1 -->|jika suhu > 8| B2[🔔 Kirim WhatsApp via Twilio]

C1 --> D1[Dashboard Web Tampil: Suhu, Status, Grafik]

D1 -->|ubah mode/LED| D2[command.php update command.json]

D2 --> E1[ESP32 baca command.json via HTTP]

E1 -->|mode SENSOR| E2[Kontrol LED + Buzzer otomatis]

E1 -->|mode DASHBOARD| E3[LED dikontrol dari dashboard]

## 💡 ****Penjelasan Mode****

| **Mode** | **Fungsi** |
| --- | --- |
| SENSOR | ESP32 otomatis nyalakan LED/buzzer jika suhu > 8 |
| DASHBOARD | LED dikendalikan manual dari dashboard (tidak membaca suhu) |

## ✅ Komponen Utama Sistem

| **Komponen** | **Fungsi** |
| --- | --- |
| ESP32 | Otak sistem, baca sensor dan ambil/kirim data |
| DHT22 | Sensor suhu dan kelembaban |
| LCD | Tampilkan nilai suhu/humidity lokal |
| webhook.php | Simpan data ke file + kirim notifikasi WhatsApp jika suhu tinggi |
| command.php | Menerima perintah dari dashboard dan simpan ke command.json |
| Twilio | Layanan API untuk kirim WhatsApp alert otomatis |
| index.php | Dashboard monitoring dan kontrol mode/LED |

**kode .ino**

**webhook.php**

**command.php**

**command.json**

penjelasan mengenai kode yang digunakan dalam proyek **Monitoring Suhu dan Kelembaban Kamar Jenazah** menggunakan **ESP32**, **DHT22**, **LCD I2C**, dan **koneksi internet (WiFi)** ke **dashboard web**.

**🧾 1. Library yang Digunakan**

#include <WiFi.h>

#include <HTTPClient.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include "DHTesp.h"

#include <ArduinoJson.h>

* WiFi.h & HTTPClient.h: Untuk koneksi WiFi & mengirim HTTP request ke server.
* Wire.h & LiquidCrystal\_I2C.h: Untuk komunikasi I2C dengan LCD 16x2.
* DHTesp.h: Untuk pembacaan sensor suhu dan kelembaban DHT22.
* ArduinoJson.h: Untuk parsing JSON dari command server (dashboard).

**🧩 2. Pin dan Konfigurasi**

#define DHT\_PIN 15

#define LED\_MERAH 2

#define LED\_HIJAU 27

#define BUZZER\_PIN 14

* **DHT\_PIN**: Pin data dari sensor DHT22.
* **LED\_MERAH & LED\_HIJAU**: Indikator status suhu.
* **BUZZER\_PIN**: Bunyi alarm saat suhu bahaya.

**🌐 3. Koneksi WiFi dan URL Webhook**

const char\* ssid = "Wokwi-GUEST";

const char\* webhookUrl = ".../webhook.php";

const char\* commandUrl = ".../command.json";

* ESP32 terhubung ke WiFi.
* **webhookUrl**: Tempat mengirim suhu/kelembaban ke server PHP.
* **commandUrl**: Tempat mengambil perintah dari dashboard (mode & LED).

**📟 4. Setup Awal (void setup)**

WiFi.begin(ssid, password);

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

pinMode(...);

* Inisialisasi WiFi, LCD, dan semua pin.
* Menunggu hingga WiFi tersambung.

**🔄 5. Loop Utama (void loop)**

TempAndHumidity data = dht.getTempAndHumidity();

lcd.setCursor(0, 0);

**a. Baca Sensor & Tampilkan ke LCD**

* Membaca suhu dan kelembaban setiap loop.
* Menampilkan ke LCD baris 1 dan 2.

**b. Kirim Webhook Setiap 10 Detik**

if (millis() - lastWebhookTime > intervalWebhook) {

sendWebhook(...);

}

* Mengirim data suhu dan kelembaban ke server setiap 10 detik agar dashboard tetap update.

**c. Kontrol Otomatis (jika mode = SENSOR)**

if (modeGlobal == "SENSOR") {

if (suhuGlobal > 8) {

// LED Merah & Buzzer aktif

} else {

// LED Hijau menyala

}

}

* Bila suhu > 8°C → LED Merah menyala & buzzer bunyi.
* Bila suhu aman → LED Hijau menyala.

**d. Ambil Perintah dari Dashboard**

checkCommand();

* Fungsi checkCommand() mengecek mode dan status LED dari command.json.

**📬 6. Fungsi Kirim Data ke Webhook (sendWebhook)**

HTTPClient http;

http.begin(...?suhu=...);

* Mengirim data ke URL PHP webhook.php.
* Server menyimpan ke database dan JSON untuk dashboard.

**📥 7. Fungsi Ambil Perintah Dashboard (checkCommand)**

if (WiFi.status() == WL\_CONNECTED) {

http.begin(commandUrl);

}

**a. Ambil JSON dari Server**

* File command.json berisi:

json

CopyEdit

{

"mode": "DASHBOARD",

"led": "ON"

}

**b. Jika Mode = DASHBOARD**

if (modeGlobal == "DASHBOARD") {

if (ledStatus == "ON") { ... }

else { matikan LED }

}

* Bila led = ON → nyalakan LED sesuai suhu:
  + Suhu < 8°C → LED hijau
  + Suhu > 8°C → LED merah
* Bila led = OFF → semua LED dimatikan.

**✅ Kesimpulan Fungsi Utama**

| **Komponen** | **Fungsi** |
| --- | --- |
| DHT22 | Membaca suhu dan kelembaban ruangan. |
| LCD I2C | Menampilkan data suhu dan kelembaban real-time. |
| LED & Buzzer | Memberi notifikasi visual dan suara jika suhu > 8°C. |
| WiFi & Server | Mengirim data ke dashboard dan menerima kontrol dari dashboard. |
| Mode SENSOR | ESP32 otomatis kontrol LED & buzzer berdasarkan suhu. |
| Mode DASHBOARD | LED dikontrol manual oleh admin melalui dashboard web. |

**penjelasan** dari kode webhook.php yang digunakan dalam proyek **IoT Monitoring Suhu + Notifikasi WhatsApp**:

**🔧 Tujuan webhook.php**

File ini **menerima data suhu & kelembaban** yang dikirim oleh **ESP32**, lalu:

1. Menyimpan data ke file data.json untuk ditampilkan di **dashboard**.
2. Mengirim **notifikasi WhatsApp via Twilio** jika suhu lebih dari 8°C.
3. Menampilkan hasil pemrosesan (untuk debug & uji coba).

**🧠 Penjelasan Per Baris**

**Bagian 1: Menampilkan isi parameter GET**

echo "<pre>";

print\_r($\_GET);

echo "</pre>";

* Untuk keperluan debugging.
* Menampilkan semua parameter yang dikirim melalui URL (misalnya ?suhu=9.3&humidity=65).

**Bagian 2: Ambil dan validasi data suhu & kelembaban**

$suhu = $\_GET['suhu'] ?? null;

$humidity = $\_GET['humidity'] ?? null;

if ($suhu !== null && is\_numeric($suhu)) {

* Ambil nilai suhu dan kelembaban dari parameter GET.
* Cek apakah suhu valid (ada dan berupa angka).

**Bagian 3: Notifikasi jika suhu > 8°C**

if ($suhu > 8) {

// Kirim WhatsApp

$message = "🚨 ALERT: Suhu Tinggi Terdeteksi!\n..." ;

...

$result = file\_get\_contents($url, false, $context);

* Jika suhu lebih dari 8°C, sistem:
  + Menyiapkan pesan alert.
  + Menggunakan API Twilio untuk mengirim pesan WhatsApp ke nomor yang ditentukan.

$sid = "...";

$token = "...";

$twilio\_number = "whatsapp:+14155238886";

$target\_number = "whatsapp:+62813...";

* Ini adalah data kredensial Twilio dan nomor tujuan.

**Bagian 4: Jika suhu aman**

} else {

echo "✅ Suhu aman ($suhu°C), tidak perlu notifikasi.";

}

* Jika suhu tidak melebihi ambang batas, maka tidak dikirim WA, hanya tampilkan pesan aman.

**Bagian 5: Simpan ke data.json**

$jsonData = [

'suhu' => $suhu,

'humidity' => $humidity ?? 'N/A',

'timestamp' => date('d-m-Y H:i:s')

];

file\_put\_contents("data.json", json\_encode($jsonData, JSON\_PRETTY\_PRINT));

* Menyimpan suhu, kelembaban, dan waktu saat data diterima.
* File ini akan **dibaca oleh dashboard web** agar tampilannya selalu update.

**Bagian 6: Error handling jika suhu tidak valid**

} else {

echo "❌ Parameter suhu tidak ditemukan atau tidak valid.";

}

* Jika tidak ada suhu dikirim atau nilainya tidak valid (misal teks), tampilkan pesan error.

**Bagian 7: Tampilkan respons dari Twilio (opsional)**

if (isset($result)) {

echo "<pre>Response dari Twilio:\n";

print\_r($result);

echo "</pre>";

}

* Untuk debugging: menampilkan respon dari Twilio apakah berhasil mengirim WA atau tidak.

**🔁 Skema Alur Singkat**

ESP32 ➜ webhook.php ➜

✔️ Simpan suhu & kelembaban

✔️ Jika suhu > 8 ➜ kirim WhatsApp

✔️ Dashboard baca data.json

**📄 Tujuan File: command.php**

File ini digunakan untuk:

* **Menerima kontrol dari dashboard** (dari form di index.php).
* **Memperbarui file command.json**, yang akan dibaca oleh ESP32.
* Mendukung 2 aksi utama:
  1. Mengganti **mode** (SENSOR atau DASHBOARD).
  2. Mengatur status **LED** (ON atau OFF).

**🔍 Penjelasan Baris per Baris**

**✅ 1. Aktifkan Error Reporting (Debugging)**

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

* Menampilkan pesan kesalahan jika ada bug (penting saat development).

**📥 2. Ambil Data dari Form (POST)**

$mode = $\_POST['mode'] ?? null;

$led = $\_POST['led'] ?? null;

* Mengambil nilai dari form yang dikirim dengan method POST:
  + mode → "SENSOR" atau "DASHBOARD"
  + led → "ON" atau "OFF"

**🧠 3. Jika Ada Data yang Dikirim**

if ($mode || $led) {

...

}

* Program akan lanjut **hanya jika ada perubahan yang dikirim dari form**.

**📂 4. Baca File JSON Lama (Jika Ada)**

$command = file\_exists('command.json') ? json\_decode(file\_get\_contents('command.json'), true) : [

'mode' => 'SENSOR',

'led' => 'OFF'

];

* Membaca file command.json untuk mendapatkan **status mode & LED sebelumnya**.
* Jika file belum ada, digunakan **default**: mode=SENSOR, led=OFF.

**🔄 5. Perbarui Nilai Mode dan LED**

if ($mode !== null) {

$command['mode'] = strtoupper($mode);

}

if ($led !== null) {

$command['led'] = strtoupper($led);

}

* Jika user memilih **mode baru**, akan di-update.
* Jika user menekan tombol **LED**, status LED diubah.
* strtoupper() digunakan agar selalu tersimpan dalam huruf kapital (mis. ON bukan on).

**💾 6. Simpan ke command.json**

file\_put\_contents('command.json', json\_encode($command, JSON\_PRETTY\_PRINT));

* Perintah terbaru disimpan kembali ke file command.json.
* File ini akan dibaca oleh ESP32 setiap detik (via checkCommand()).

**🔁 7. Redirect ke Dashboard**

header("Location: index.php");

exit;

* Setelah menyimpan, langsung **redirect ke halaman dashboard** agar:
  + Tampilan ter-refresh.
  + Tidak mengirim ulang form saat di-refresh manual.

**🚫 8. Jika Tidak Ada Data Dikirim**

echo "❌ Tidak ada parameter 'mode' atau 'led' yang dikirim.";

* Muncul jika file command.php diakses langsung tanpa data POST.

**🔚 Kesimpulan**

| **Aksi dari Dashboard** | **Efek pada File JSON** | **Efek di ESP32** |
| --- | --- | --- |
| Pilih Mode: DASHBOARD | mode = DASHBOARD | LED dikontrol manual |
| Pilih Mode: SENSOR | mode = SENSOR | ESP32 kontrol otomatis |
| Tekan LED ON/OFF | led = ON/OFF | Mengatur nyala LED di mode DASHBOARD |

penjelasan untuk isi file **command.json**:

{

"mode": "DASHBOARD",

"led": "OFF"

}

**🔍 Apa Itu command.json?**

command.json adalah **file konfigurasi** yang digunakan untuk menyimpan **perintah kontrol terbaru** dari dashboard web, yang nantinya akan dibaca secara berkala oleh **ESP32**.

File ini bersifat **dinamis**, diubah oleh:

* **User melalui dashboard** (via command.php).
* **ESP32 tidak mengubah file ini**, hanya membaca.

**🧠 Penjelasan Isi**

| **Key** | **Nilai** | **Arti** |
| --- | --- | --- |
| mode | DASHBOARD | Artinya kontrol perangkat (seperti LED) dilakukan **dari dashboard** |
| led | OFF | Artinya **LED dalam keadaan mati**, karena tombol terakhir yang ditekan adalah **Matikan LED** |

**🔁 Hubungan dengan ESP32**

Saat ESP32 menjalankan fungsi checkCommand():

1. ESP32 membaca isi command.json.
2. Jika mode == "DASHBOARD", maka ESP32 **tidak menggunakan suhu sensor**, melainkan mengikuti perintah led:
   * led = ON → Nyalakan LED
   * led = OFF → Matikan LED
3. Jika mode == "SENSOR", maka ESP32 akan **mengabaikan nilai led** dan bekerja secara otomatis berdasarkan suhu.

**✅ Contoh Kasus Penggunaan**

| **Kondisi di Dashboard** | **Isi command.json** | **Efek di ESP32** |
| --- | --- | --- |
| User pilih "Mode SENSOR" | { "mode": "SENSOR", "led": "OFF" } | Kontrol otomatis berdasarkan suhu |
| User pilih "Mode DASHBOARD" + Nyalakan LED | { "mode": "DASHBOARD", "led": "ON" } | LED nyala manual, abaikan sensor suhu |
| User tekan "Matikan LED" di mode dashboard | { "mode": "DASHBOARD", "led": "OFF" } | LED langsung dimatikan, tidak otomatis |

**🧩 Letak File**

File command.json biasanya disimpan di root direktori yang sama dengan index.php dan command.php.

ESP32 mengambil file ini secara real-time melalui:

HTTPClient http;

http.begin("https://namadomainmu.com/iot/command.json");