**LAPORAN PRAKTIKUM REST API BASIC**

**MENAMPILKAN DATA CUACA DENGAN ESP32 PADA IOT**

****

**Dosen Pengampu :**

Ir. Subairi, ST., MT., IPM

**Disusun Oleh :**

Cakra Pradipta Kuntara (233140707111056)

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI UNIVERSITAS BRAWIJAYA**

**MALANG 2025**

**Abstract**

This experiment aims to create a weather monitoring system using the ESP32 microcontroller and LCD I2C display. It connects to an open weather API to fetch real-time weather data, including temperature, windspeed, and weather condition codes. The system supports a multi-page LCD interface navigated by buttons to switch between data categories. The result shows that ESP32 is capable of fetching, parsing JSON, and displaying data efficiently on a limited display.

***Keywords****—ESP32, LCD I2C, Weather API, Internet of Things, JSON, Push Button*

1. **Introduction (Pendahuluan)**

Internet of Things (IoT) memungkinkan berbagai perangkat fisik untuk berkomunikasi dan bertukar data melalui jaringan. Dalam eksperimen ini, ESP32 digunakan untuk mengakses API cuaca dan menampilkannya ke layar LCD, sehingga dapat digunakan sebagai sistem monitoring cuaca sederhana berbasis jaringan.

* 1. **Latar Belakang**

Pemantauan kondisi cuaca secara real-time sangat penting dalam banyak bidang seperti pertanian, transportasi, bahkan perencanaan aktivitas sehari-hari. Sebagian besar sistem informasi cuaca yang tersedia saat ini membutuhkan perangkat kompleks dan mahal. Namun, dengan hadirnya mikrokontroler ESP32, kini pemantauan sederhana dapat dilakukan dengan biaya rendah dan konfigurasi ringan. Penggunaan API cuaca online juga menambah fleksibilitas, karena data yang ditampilkan akan selalu diperbarui secara otomatis tanpa perlu sensor tambahan. Dengan begitu, pengguna dapat memperoleh informasi cuaca akurat hanya dengan koneksi internet dan LCD.

* 1. **Tujuan Eksperimen**

Adapun tujuan dari praktikum ini adalah:

* Membangun sistem informasi cuaca sederhana menggunakan ESP32 dan LCD I2C.
* Mengambil data cuaca real-time dari open API dan menampilkannya pada perangkat fisik.
* Mengimplementasikan navigasi antar informasi menggunakan tombol Next dan Previous.
* Melatih penggunaan HTTP request dan parsing data JSON di mikrokontroler.

1. **Methodology (Metodologi)**
   1. **Tools & Materials**

* ESP32
* LCD 1602 dengan modul l2C
* 2 Push Button (Next dan Previous)
* Kabel Jumper dan Breadboard
* Software Arduino IDE
  1. **Implementation Steps**
* Persiapkan Komponen ESP32 board, 2 buah push button (merah & hijau) dan LCD 16x2 dengan modul I2C.
* Rangkai Komponen ESP32 board, 2 buah push button (merah & hijau) dan LCD di platform wokwi
* Menulis kode program untuk menampilkan temperature pada LED.
* Memindahkan kode ESP32 ke Visual Studio Code, dan mengamati hasilnya

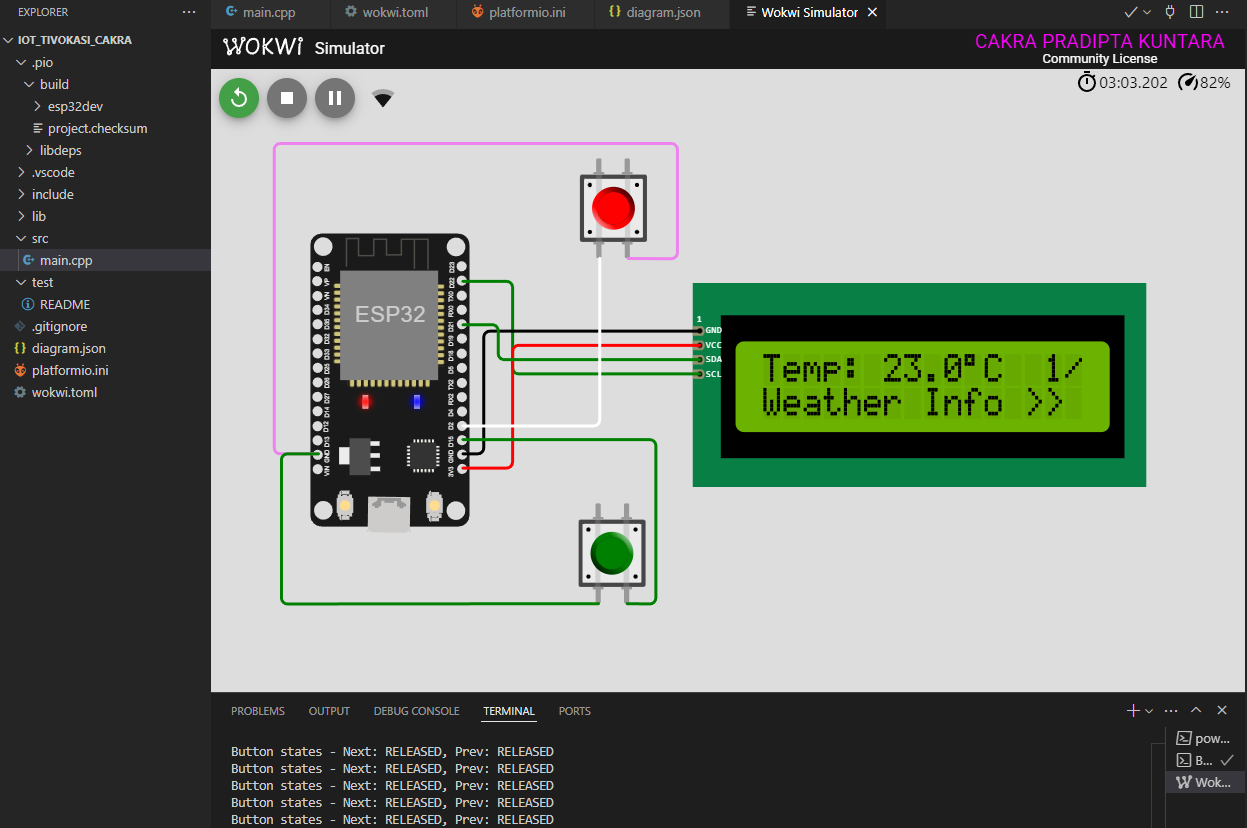
1. **Results and Discussion (Hasil dan pembahasan)**
   1. **Experimental Results**

Hasil eksperimen menunjukkan sistem berjalan sesuai ekspektasi:

* LCD berhasil menampilkan data suhu, angin, dan kondisi cuaca dalam tampilan yang berbeda.
* Tombol Next dan Previous berfungsi baik setelah ditambahkan logika debounce.
* Koneksi ke WiFi berjalan lancar dan API berhasil merespon data dengan cepat.

Berikut adalah dokumentasi eksperimen meliputi screenshoot simulasi ESP32 di Visual Studio Code:

* Menampilkan Data Temperatur



* Menampilkan Data Kecepatan Angin

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Appendix**

Berikut adalah kode program yang digunakan untuk simulasi:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

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

const char\* password = "";

String server = "https://api.open-meteo.com/v1/forecast?latitude=-7.9797&longitude=112.6304&current\_weather=true&timezone=Asia%2FJakarta";

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

const int buttonNext = 2;

const int buttonPrev = 15;

int currentPage = 0;

const int totalPages = 2;

String temperature = "N/A";

String windspeed = "N/A";

String winddirection = "N/A";

unsigned long lastButtonTimeNext = 0;

unsigned long lastButtonTimePrev = 0;

const unsigned long debounceDelay = 200;

unsigned long lastRefreshTime = 0;

const unsigned long refreshInterval = 3000;

unsigned long lastWeatherFetchTime = 0;

const unsigned long weatherFetchInterval = 60000;

void fetchWeather();

void handleButtons();

void displayPage();

void setup() {

    Serial.begin(115200);

    lcd.init();

    lcd.backlight();

    lcd.setCursor(0, 0);

    lcd.print("Weather Info:");

    pinMode(buttonNext, INPUT\_PULLUP);

    pinMode(buttonPrev, INPUT\_PULLUP);

    WiFi.begin(ssid, password);

    lcd.setCursor(0, 1);

    lcd.print("Connecting...");

    unsigned long startTime = millis();

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

        if (millis() - startTime >= 10000) {

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("WiFi Failed");

            Serial.println("WiFi connection failed");

            return;

        }

        delay(1000);

        Serial.println("Connecting to WiFi...");

    }

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("Connected!");

    delay(2000);

    lcd.clear();

    fetchWeather();

    displayPage();

}

void loop() {

    handleButtons();

    unsigned long currentTime = millis();

    if (currentTime - lastWeatherFetchTime >= weatherFetchInterval) {

        fetchWeather();

        lastWeatherFetchTime = currentTime;

    }

    if (currentTime - lastRefreshTime >= refreshInterval) {

        displayPage();

        lastRefreshTime = currentTime;

    }

    delay(50);

}

void fetchWeather() {

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

        HTTPClient http;

        http.begin(server);

        int httpCode = http.GET();

        if (httpCode > 0) {

            String payload = http.getString();

            Serial.println(payload);

            StaticJsonDocument<1024> doc;

            DeserializationError error = deserializeJson(doc, payload);

            if (!error) {

                float temp = doc["current\_weather"]["temperature"];

                float wind = doc["current\_weather"]["windspeed"];

                int windDir = doc["current\_weather"]["winddirection"];

                temperature = String(temp, 1);

                windspeed = String(wind, 1);

                winddirection = String(windDir);

                Serial.println("Parsed values from JSON:");

                Serial.println("Temperature: " + temperature + "°C");

                Serial.println("Wind Speed: " + windspeed + " km/h");

                Serial.println("Wind Direction: " + winddirection + "°");

            } else {

                Serial.println("Failed to parse JSON");

                temperature = "Error";

                windspeed = "Error";

                winddirection = "Error";

            }

        } else {

            Serial.println("HTTP request failed");

            temperature = "Error";

            windspeed = "Error";

            winddirection = "Error";

        }

        http.end();

    }

}

void handleButtons() {

    unsigned long now = millis();

    int nextState = digitalRead(buttonNext);

    if (nextState == LOW && now - lastButtonTimeNext > debounceDelay) {

        currentPage = (currentPage + 1) % totalPages;

        lastButtonTimeNext = now;

        Serial.println("Button NEXT pressed - Page: " + String(currentPage + 1) + "/" + String(totalPages));

        displayPage();

        lastRefreshTime = now;

    }

    int prevState = digitalRead(buttonPrev);

    if (prevState == LOW && now - lastButtonTimePrev > debounceDelay) {

        currentPage = (currentPage - 1 + totalPages) % totalPages;

        lastButtonTimePrev = now;

        Serial.println("Button PREV pressed - Page: " + String(currentPage + 1) + "/" + String(totalPages));

        displayPage();

        lastRefreshTime = now;

    }

    static unsigned long lastDebugTime = 0;

    if (now - lastDebugTime > 1000) {

        Serial.print("Button states - Next: ");

        Serial.print(nextState == LOW ? "PRESSED" : "RELEASED");

        Serial.print(", Prev: ");

        Serial.println(prevState == LOW ? "PRESSED" : "RELEASED");

        lastDebugTime = now;

    }

}

void displayPage() {

    lcd.clear();

    lcd.setCursor(14, 0);

    lcd.print(currentPage + 1);

    lcd.print("/");

    lcd.print(totalPages);

    if (currentPage == 0) {

        lcd.setCursor(0, 0);

        lcd.print("Temp: ");

        lcd.print(temperature);

        lcd.print((char)223);

        lcd.print("C");

        lcd.setCursor(0, 1);

        lcd.print("Weather Info >>");

    } else if (currentPage == 1) {

        lcd.setCursor(0, 0);

        lcd.print("Wind: ");

        lcd.print(windspeed);

        lcd.print(" km/h");

        lcd.setCursor(0, 1);

        lcd.print("Dir: ");

        lcd.print(winddirection);

        lcd.print((char)248);

    }

}