**Praktik Implementasi Penggunaan API OpenWeather Menggunakan Wokwi**

**oleh**

*Nadia Alya Paramitha Erwanto1*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:*[*nadiaalya2729@student.ub.ac.id*](https://d.docs.live.net/18b7c2b78e49b7a2/Dokumen/nadiaalya2729@student.ub.ac.id)

**Abstrak**

Pada praktik ini, dilakukan implementasi penggunaan API OpenWeather untuk mendapatkan data cuaca secara real-time menggunakan simulasi perangkat ESP32 melalui platform Wokwi. Data yang diambil meliputi informasi kota, negara, kondisi cuaca, deskripsi cuaca, suhu, kelembapan, tekanan udara, kecepatan dan arah angin, serta waktu terbit dan terbenam matahari. Setelah berhasil mengambil data dari API, informasi tersebut ditampilkan secara rapi di Serial Monitor dan OLED Display. Proses parsing data dilakukan menggunakan library ArduinoJson untuk memudahkan pengolahan data JSON dari API. Selain itu, kondisi tertentu seperti suhu panas, suhu dingin, dan cuaca hujan dikendalikan melalui LED indikator dan buzzer. Praktik ini bertujuan untuk melatih pemahaman dalam integrasi mikrokontroler dengan layanan berbasis internet (IoT) serta penerapan API dalam proyek embedded system. Hasil praktik menunjukkan bahwa ESP32 dapat secara efektif mengambil, memproses, dan menampilkan data dari API OpenWeather dengan koneksi WiFi melalui simulasi Wokwi.

***Kata kunci****: OpenWeather API, ESP32, IoT, Wokwi,Cuaca.*

**Abstract**

This practice involves the implementation of the OpenWeather API to retrieve real-time weather data using the ESP32 microcontroller simulated on the Wokwi platform. The retrieved data includes city name, country, weather conditions, weather description, temperature, humidity, atmospheric pressure, wind speed and direction, as well as sunrise and sunset times. After successfully fetching the data, the information is neatly displayed on the Serial Monitor and OLED Display. JSON data parsing is performed using the ArduinoJson library to simplify the extraction of relevant values from the API response. Furthermore, specific weather conditions such as hot, cold, and rainy weather are indicated using LED indicators and a buzzer. This practice aims to enhance the understanding of integrating microcontrollers with internet-based services (IoT) and applying APIs in embedded system projects. The results show that the ESP32 is capable of effectively fetching, processing, and displaying weather data from the OpenWeather API over WiFi through Wokwi simulation.

***Keywords:*** *OpenWeather API, ESP32, IoT, Wokwi,Weather Monitoring*

**Pendahuluan**

Perkembangan teknologi Internet of Things (IoT) memungkinkan berbagai perangkat untuk saling terhubung dan bertukar data melalui internet. Salah satu penerapan IoT yang banyak dikembangkan adalah sistem monitoring cuaca berbasis API. Dengan memanfaatkan Application Programming Interface (API) dari layanan penyedia data cuaca seperti OpenWeather, pengguna dapat memperoleh data cuaca secara real-time tanpa harus memiliki sensor fisik untuk mengukur berbagai parameter atmosfer.

Pada praktik ini, dilakukan implementasi penggunaan OpenWeather API untuk mengambil data cuaca menggunakan mikrokontroler ESP32 yang disimulasikan melalui platform Wokwi. Data yang diambil meliputi nama kota, negara, kondisi cuaca, deskripsi cuaca, suhu, kelembapan, tekanan udara, kecepatan dan arah angin, serta waktu matahari terbit dan terbenam. Data yang diperoleh dari API diproses dan ditampilkan secara rapi pada Serial Monitor dan OLED Display. Selain itu, digunakan juga indikator LED dan buzzer untuk memberikan peringatan berdasarkan kondisi suhu atau cuaca tertentu.

Melalui praktik ini, diharapkan peserta dapat memahami proses pengambilan data dari API, parsing data dalam format JSON, serta bagaimana mengintegrasikan data tersebut ke dalam sistem berbasis mikrokontroler. Penerapan ini menjadi contoh nyata bagaimana sistem IoT dapat berinteraksi dengan layanan berbasis cloud untuk memperkaya fungsionalitas perangkat.

**Metodologi**

Dalam praktik ini, metodologi yang digunakan mencakup beberapa tahapan mulai dari perancangan sistem, koneksi perangkat, konfigurasi jaringan, hingga pengambilan dan penampilan data cuaca. Setiap tahap dilakukan secara sistematis untuk memastikan integrasi antara perangkat keras dan perangkat lunak berjalan dengan baik. 1. Perancangan Sistem

1. Perancangan Sistem

Langkah pertama adalah merancang sistem berbasis IoT menggunakan ESP32 pada platform simulasi Wokwi. Perancangan ini melibatkan integrasi beberapa komponen, yaitu:

* ESP32 DevKit V1 sebagai mikrokontroler utama.
* DHT22 untuk pengukuran suhu dan kelembapan lokal.
* OLED Display (SSD1306) untuk menampilkan data cuaca.
* LED (tiga buah) untuk indikator suhu (dingin, normal, panas).
* Buzzer untuk memberikan peringatan suara saat cuaca ekstrem (seperti hujan atau badai).

2. Koneksi Perangkat

* DHT22 dihubungkan ke pin digital ESP32 (misal D15).
* OLED Display menggunakan komunikasi I2C melalui pin D21 (SDA) dan D22 (SCL).
* LED dihubungkan ke pin D2, D4, dan D5 untuk masing-masing indikator suhu.
* Buzzer dihubungkan ke pin D18.

3. Pengaturan Koneksi WiFi

ESP32 dikonfigurasi untuk terhubung ke jaringan WiFi publik "Wokwi-GUEST" tanpa password, agar dapat mengakses layanan internet.

4. Pengambilan Data dari API

ESP32 menggunakan HTTPClient library untuk melakukan request ke OpenWeather API. Data yang diperoleh berupa format JSON, yang kemudian diparsing untuk mengambil informasi cuaca seperti:

* Nama kota dan negara
* Kondisi cuaca dan deskripsi
* Suhu, suhu minimum dan maksimum
* Kelembapan, tekanan udara
* Kecepatan dan arah angin
* Waktu sunrise dan sunset

5. Pengolahan dan Penampilan Data

Data hasil parsing kemudian:

* Ditampilkan di Serial Monitor dalam format rapi.
* Ditampilkan di OLED Display dalam beberapa baris.
* Mengaktifkan LED sesuai kondisi suhu:
  + LED merah untuk suhu panas
  + LED biru untuk suhu dingin
  + LED hijau untuk suhu normal
* Mengaktifkan buzzer jika cuaca hujan atau badai terdeteksi.

**Hasil dan Pembahasan**

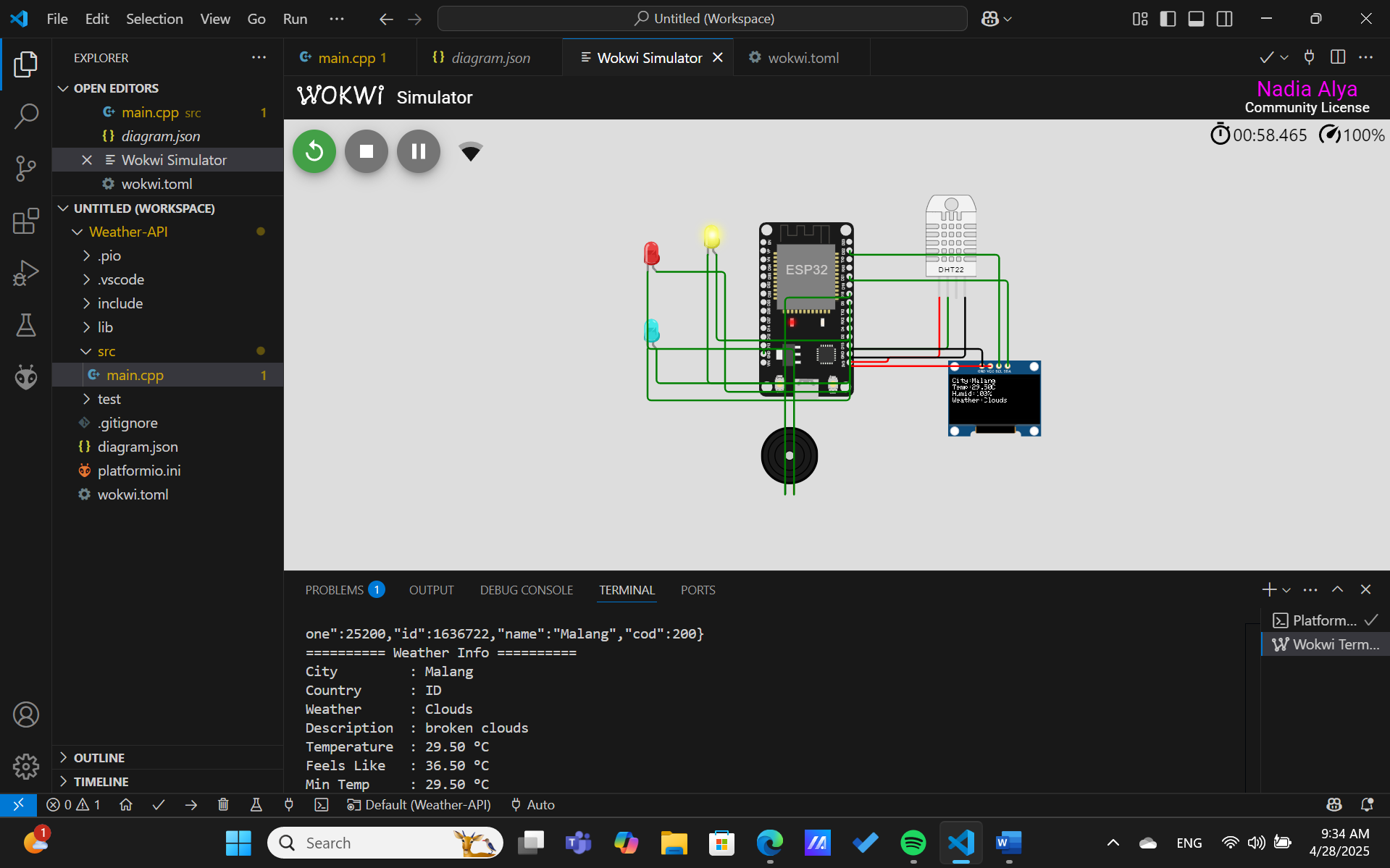
Setelah melakukan konfigurasi dan pengembangan sistem berbasis ESP32, proyek berhasil mengambil data cuaca secara real-time dari API OpenWeather dan menampilkannya melalui Serial Monitor dan layar OLED. Data yang ditampilkan meliputi informasi lengkap seperti nama kota, negara, kondisi cuaca, deskripsi cuaca, suhu, suhu terasa, suhu minimum dan maksimum, tekanan udara, kelembapan, kecepatan angin, arah angin, waktu matahari terbit, dan waktu matahari terbenam.

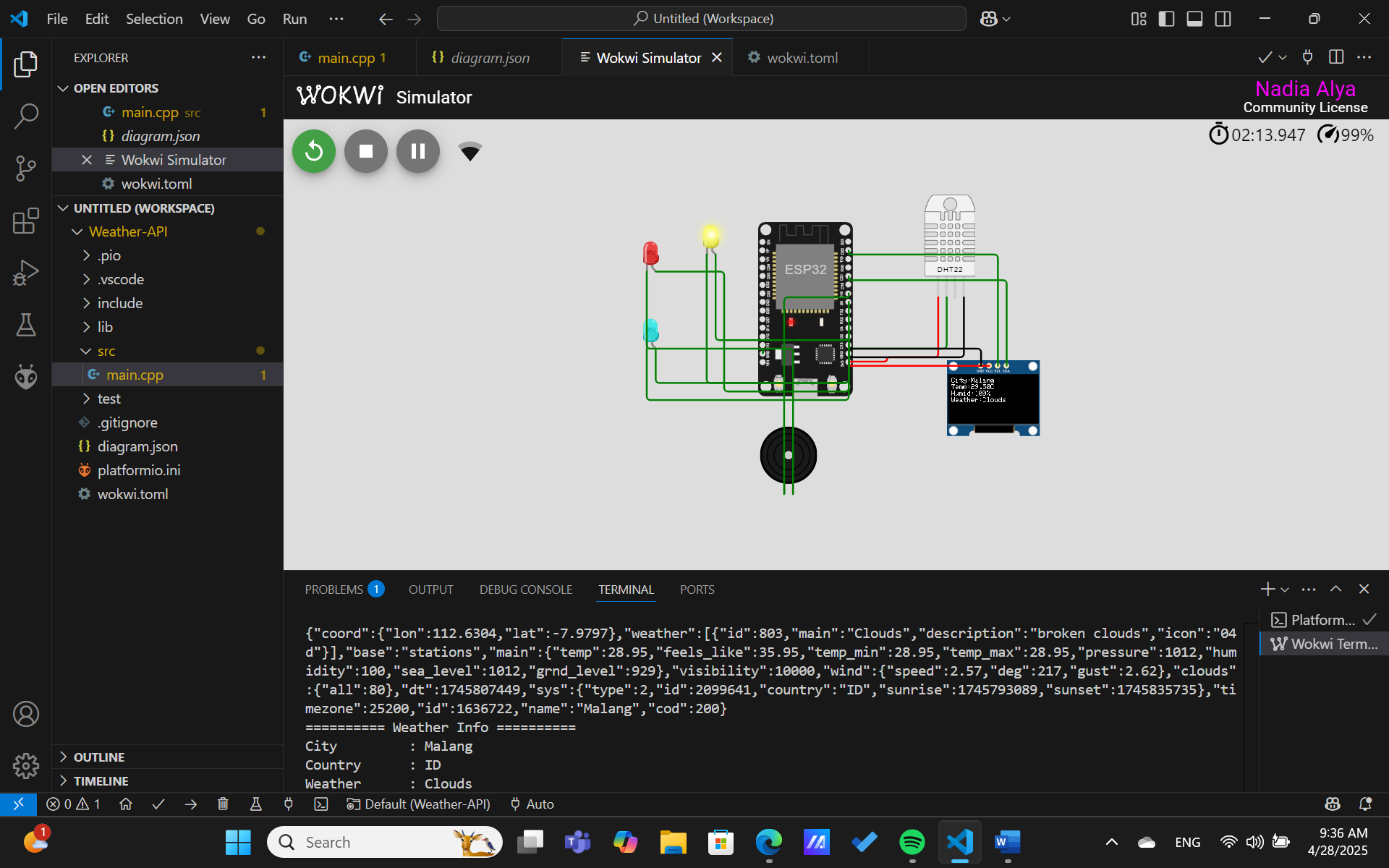
Pada Serial Monitor, informasi dicetak dengan format rapi yang memudahkan pengguna membaca setiap parameter cuaca. Tampilan diawali dan diakhiri dengan garis pembatas agar lebih terstruktur.  
Sementara itu, pada OLED display, data penting seperti suhu, kelembapan, dan kondisi cuaca juga ditampilkan bergiliran untuk menyesuaikan dengan ukuran layar yang terbatas.

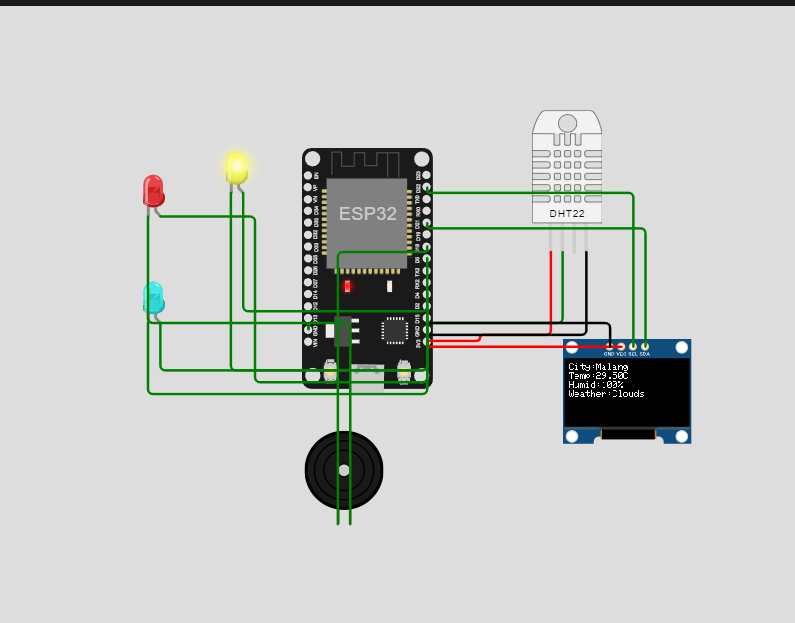
LED indikator bekerja sesuai suhu dari data API:

* **LED Merah** menyala jika suhu sangat panas (≥30°C).
* **LED Biru** menyala jika suhu dingin (≤20°C).
* **LED Hijau** menyala untuk suhu normal di antara keduanya.

Selain itu, buzzer aktif memberikan peringatan bunyi singkat jika terdeteksi cuaca ekstrem seperti hujan deras atau badai, berdasarkan deskripsi cuaca dari API. Seluruh sistem berjalan stabil dan pembacaan data dari OpenWeather secara berkala diperbarui setiap 60 detik, sesuai kebutuhan monitoring cuaca real-time.







**Lampiran**

Source code **main.cpp :**

#include <Wire.h>

#include <Adafruit\_SSD1306.h>

#include <Adafruit\_GFX.h>

#include <DHT.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

// WiFi Setup

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

const char\* password = "";

// OpenWeatherMap

String apiKey = "6375be75703e403d828c83238072ae30";

String city = "Malang";

String units = "metric";

String server = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=" + units + "&appid=" + apiKey;

// OLED

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// DHT Sensor

#define DHTPIN 15

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

// LED Pins

#define LED\_HOT 2

#define LED\_COLD 4

#define LED\_NORMAL 5

// Buzzer Pin

#define BUZZER\_PIN 18

void setup() {

  Serial.begin(115200);

  // Start OLED

  if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

    Serial.println(F("SSD1306 allocation failed"));

    for(;;);

  }

  display.display();

  delay(1000);

  display.clearDisplay();

  // Start DHT

  dht.begin();

  // LED and Buzzer setup

  pinMode(LED\_HOT, OUTPUT);

  pinMode(LED\_COLD, OUTPUT);

  pinMode(LED\_NORMAL, OUTPUT);

  pinMode(BUZZER\_PIN, OUTPUT);

  // Connect WiFi

  WiFi.begin(ssid, password);

  display.setTextSize(1);

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(0, 10);

  display.print("Connecting WiFi...");

  display.display();

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

    delay(500);

    Serial.print(".");

  }

  display.clearDisplay();

  display.setCursor(0, 10);

  display.print("WiFi Connected!");

  display.display();

  delay(1000);

}

void loop() {

  display.clearDisplay();

  // DHT Local Sensor Read

  float h = dht.readHumidity();

  float t = dht.readTemperature();

  // Check WiFi

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

    HTTPClient http;

    http.begin(server);

    int httpCode = http.GET();

    if (httpCode > 0) {

      String payload = http.getString();

      Serial.println(payload);

      // Parsing JSON

      StaticJsonDocument<1024> doc;

      DeserializationError error = deserializeJson(doc, payload);

      if (!error) {

        String cityName = doc["name"];

        String country = doc["sys"]["country"];

        String weatherMain = doc["weather"][0]["main"];

        String weatherDesc = doc["weather"][0]["description"];

        float temp = doc["main"]["temp"];

        float feelsLike = doc["main"]["feels\_like"];

        float tempMin = doc["main"]["temp\_min"];

        float tempMax = doc["main"]["temp\_max"];

        int pressure = doc["main"]["pressure"];

        int humidity = doc["main"]["humidity"];

        float windSpeed = doc["wind"]["speed"];

        int windDeg = doc["wind"]["deg"];

        long sunrise = doc["sys"]["sunrise"];

        long sunset = doc["sys"]["sunset"];

        // Print ke Serial Monitor

        Serial.println("========== Weather Info ==========");

        Serial.println("City         : " + cityName);

        Serial.println("Country      : " + country);

        Serial.println("Weather      : " + weatherMain);

        Serial.println("Description  : " + weatherDesc);

        Serial.print("Temperature  : "); Serial.print(temp); Serial.println(" °C");

        Serial.print("Feels Like   : "); Serial.print(feelsLike); Serial.println(" °C");

        Serial.print("Min Temp     : "); Serial.print(tempMin); Serial.println(" °C");

        Serial.print("Max Temp     : "); Serial.print(tempMax); Serial.println(" °C");

        Serial.print("Pressure     : "); Serial.print(pressure); Serial.println(" hPa");

        Serial.print("Humidity     : "); Serial.print(humidity); Serial.println(" %");

        Serial.print("Wind Speed   : "); Serial.print(windSpeed); Serial.println(" m/s");

        Serial.print("Wind Degree  : "); Serial.print(windDeg); Serial.println(" °");

        // Konversi sunrise & sunset dari epoch ke jam:menit:detik

        time\_t rawSunrise = sunrise;

        struct tm \* sunriseTime = gmtime(&rawSunrise);

        Serial.print("Sunrise      : ");

        Serial.printf("%02d:%02d:%02d\n", sunriseTime->tm\_hour, sunriseTime->tm\_min, sunriseTime->tm\_sec);

        time\_t rawSunset = sunset;

        struct tm \* sunsetTime = gmtime(&rawSunset);

        Serial.print("Sunset       : ");

        Serial.printf("%02d:%02d:%02d\n", sunsetTime->tm\_hour, sunsetTime->tm\_min, sunsetTime->tm\_sec);

        Serial.println("===================================");

        // OLED Tampilan sederhana

        display.setTextSize(1);

        display.setCursor(0,0);

        display.print("City:");

        display.print(cityName);

        display.setCursor(0,10);

        display.print("Temp:");

        display.print(temp);

        display.print("C");

        display.setCursor(0,20);

        display.print("Humid:");

        display.print(humidity);

        display.print("%");

        display.setCursor(0,30);

        display.print("Weather:");

        display.print(weatherMain);

        display.display();

        // LED & Buzzer Logic

        digitalWrite(LED\_HOT, LOW);

        digitalWrite(LED\_COLD, LOW);

        digitalWrite(LED\_NORMAL, LOW);

        digitalWrite(BUZZER\_PIN, LOW);

        if (temp >= 30.0) {

          digitalWrite(LED\_HOT, HIGH);

        } else if (temp <= 20.0) {

          digitalWrite(LED\_COLD, HIGH);

        } else {

          digitalWrite(LED\_NORMAL, HIGH);

        }

        if (weatherDesc.indexOf("rain") >= 0 || weatherDesc.indexOf("storm") >= 0) {

          tone(BUZZER\_PIN, 1000);

          delay(500);

          noTone(BUZZER\_PIN);

        }

      } else {

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

      }

    } else {

      Serial.println("Error on HTTP request");

    }

    http.end();

  }

  delay(60000); // Update per 1 menit

}

**diagram.json :**

{

  "version": 1,

  "author": "VIOLA ROSA EMAWARDANI 2020",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": {} },

    {

      "type": "board-ssd1306",

      "id": "oled1",

      "top": 156.74,

      "left": 211.43,

      "attrs": { "i2cAddress": "0x3c" }

    },

    { "type": "wokwi-dht22", "id": "dht1", "top": -28.5, "left": 186.6, "attrs": {} },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": 102,

      "left": -140.2,

      "attrs": { "color": "cyan" }

    },

    {

      "type": "wokwi-led",

      "id": "led2",

      "top": 15.6,

      "left": -140.2,

      "attrs": { "color": "red" }

    },

    {

      "type": "wokwi-led",

      "id": "led3",

      "top": -3.6,

      "left": -73,

      "attrs": { "color": "yellow" }

    },

    {

      "type": "wokwi-buzzer",

      "id": "bz1",

      "top": 223.2,

      "left": 1.8,

      "attrs": { "volume": "0.1" }

    }

  ],

  "connections": [

    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

    [ "esp:RX0", "$serialMonitor:TX", "", [] ],

    [ "dht1:VCC", "esp:3V3", "red", [ "v67.2", "h-57.6", "v4.9" ] ],

    [ "dht1:GND", "esp:GND.1", "black", [ "v67.2", "h-124.8" ] ],

    [ "dht1:SDA", "esp:D15", "green", [ "v57.6", "h-105.5" ] ],

    [ "oled1:VCC", "esp:3V3", "red", [ "v0", "h-153.45" ] ],

    [ "oled1:GND", "esp:GND.1", "black", [ "v-19.2", "h-144" ] ],

    [ "oled1:SCL", "esp:D22", "green", [ "v-124.8", "h-162.9" ] ],

    [ "oled1:SDA", "esp:D21", "green", [ "v-96", "h-172.73" ] ],

    [ "led2:A", "esp:D2", "green", [ "h76.8", "v134.4", "h-349.9" ] ],

    [ "led1:A", "esp:D4", "green", [ "v38.4", "h-268.8" ] ],

    [ "led3:A", "esp:D5", "green", [ "v96", "h-211.2" ] ],

    [ "led1:C", "esp:GND.1", "green", [ "v57.6", "h-258.8" ] ],

    [ "led3:C", "esp:GND.1", "green", [ "v144", "h-201.2" ] ],

    [ "led2:C", "esp:GND.2", "green", [ "v86.4", "h125.2" ] ],

    [ "bz1:1", "esp:D18", "green", [ "v-220.8", "h67.2" ] ],

    [ "bz1:2", "esp:GND.2", "green", [ "v-163.2", "h-38.8" ] ]

  ],

  "dependencies": {}

}