Praktik Monitoring Cuaca Menggunakan OpenWeatherMap API

Ranindya Dwi Qintari
Fakultas Vokasi, Universitas Brawijaya
Email: ranindyadq@student.ub.ac.id

Abstrak: Praktikum ini membahas implementasi sistem Internet of Things (IoT) sederhana yang menggunakan mikrokontroler ESP32 untuk mengambil data cuaca secara real-time dari layanan OpenWeatherMap melalui koneksi Wi-Fi dan menampilkannya pada LCD I2C 16x2. Tujuan dari praktikum ini adalah untuk mengenalkan konsep integrasi antara perangkat IoT dengan layanan cloud publik. Metode yang digunakan meliputi koneksi ESP32 ke jaringan internet, permintaan data melalui HTTP dari API OpenWeatherMap, parsing data JSON, serta penampilan informasi suhu dan deskripsi cuaca pada LCD. Hasil praktikum menunjukkan bahwa sistem mampu menampilkan informasi cuaca secara real-time dan stabil selama koneksi internet tersedia. Praktikum ini memperlihatkan bagaimana perangkat IoT dapat digunakan untuk monitoring data berbasis cloud secara efisien.

Abstract: This practicum discusses the implementation of a simple Internet of Things (IoT) system using the ESP32 microcontroller to retrieve real-time weather data from the OpenWeatherMap service via Wi-Fi and display it on a 16x2 I2C LCD. The purpose of this practicum is to introduce the concept of integrating IoT devices with public cloud services. The methods used include connecting the ESP32 to the internet, fetching data via HTTP from the OpenWeatherMap API, parsing JSON data, and displaying temperature and weather descriptions on the LCD. The results show that the system is capable of displaying real-time weather information reliably as long as an internet connection is available. This practicum demonstrates how IoT devices can efficiently be used for cloud-based data monitoring.

Keywords- Internet of Things, ESP32, OpenWeatherMap, LCD I2C, HTTP API

1. PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) merupakan teknologi yang memungkinkan perangkat elektronik terhubung dan berkomunikasi melalui jaringan internet. IoT memainkan peran penting dalam pengumpulan dan penyajian data secara real-time dari berbagai sumber, baik lokal melalui sensor maupun eksternal melalui layanan berbasis cloud. Salah satu contoh penerapan sederhana dari IoT adalah pengambilan dan penampilan data cuaca dari layanan web seperti OpenWeatherMap.

Dalam praktik ini, digunakan mikrokontroler ESP32 yang memiliki konektivitas Wi-Fi bawaan untuk mengambil data cuaca seperti suhu dan deskripsi cuaca dari OpenWeatherMap API. Data tersebut kemudian ditampilkan pada LCD I2C 16x2, sehingga pengguna dapat melihat informasi cuaca terkini secara langsung dari perangkat IoT tanpa perlu membuka aplikasi atau situs web. Praktik ini tidak hanya melatih keterampilan teknis seperti pemrograman mikrokontroler dan pengambilan data dari API, tetapi juga memperkenalkan konsep integrasi perangkat IoT dengan layanan cloud publik.

1.2 Tujuan Eksperimen

- Menghubungkan ESP32 ke internet menggunakan jaringan Wi-Fi.
- Mengakses data cuaca (suhu dan deskripsi cuaca) dari layanan OpenWeatherMap melalui HTTP API.
- Menampilkan informasi cuaca secara real-time ke LCD I2C 16x2 menggunakan ESP32.

2. METODOLOGI

2.1 Alat dan Bahan

- Laptop
- Visual Studio Code
- PlatformIO Extension
- Wokwi Simulator Extension
- Mikrokontroler ESP32 (simulasi)
- LCD I2C 16x2
- Kabel jumper (simulasi)
- Internet/Wi-Fi
- API Key dari OpenWeatherMap

2.2 Langkah Implementasi

2.2.1 Simulasi Rangkaian di Wokwi

1. Membuka situs Wokwi Simulator dan memilih template ESP32.
2. Menambahkan komponen LCD I2C 16x2 dengan alamat I2C default 0x27.
3. Menghubungkan pin I2C LCD ke ESP32:
   - SDA → GPIO21
   - SCL → GPIO22
   - VCC dan GND sesuai power supply ESP32
4. Menghubungkan ESP32 ke jaringan Wi-Fi simulator (SSID: "Wokwi-GUEST", password: kosong).
5. Menulis kode pada sketch.ino untuk:
   - Koneksi ke Wi-Fi.
   - Mengakses data cuaca dari API OpenWeatherMap menggunakan HTTPClient.
   - Penguraian data suhu dan deskripsi cuaca dari respon JSON.
   - Menampilkan informasi suhu dan deskripsi pada LCD I2C.

2.2.2 Implementasi di Visual Studio Code + PlatformIO

1. Membuka Visual Studio Code dan membuat project baru menggunakan PlatformIO untuk board esp32doit-devkit-v1.
2. Menambahkan file main.cpp dan menyalin kode program dari sketch.ino di Wokwi ke dalam vscode.
3. Menambahkan diagram.json dan menyalin kode program dari diagram.json di Wokwi ke dalam vscode.
4. Menambahkan file platformio.ini dengan konfigurasi board dan library seperti:
   ```
   [env:esp32doit-devkit-v1]
   platform = espressif32
   board = esp32doit-devkit-v1
   framework = arduino
   lib_deps =
     marcoschwartz/LiquidCrystal_I2C@^1.1.4
   ```
5. Menambahkan file wokwi.toml untuk simulasi lokal:
   ```
   [wokwi]
   version = 1
   firmware = ".pio/build/esp32doit-devkit-v1/firmware.bin"
   elf = ".pio/build/esp32doit-devkit-v1/firmware.elf"
   ```
6. Melakukan proses build dan menjalankan simulasi menggunakan Wokwi di dalam VS Code.
7. Mengamati hasil di LCD berupa:
   - Baris 1: Temp: 27.5 C
   - Baris 2: clear sky, rain, atau deskripsi cuaca lain sesuai data dari API.

2.2.3 Penggunaan API OpenWeatherMap

1. Membuat akun di situs https://openweathermap.org untuk mendapatkan API Key.
2. Mengatur endpoint API untuk mengambil data cuaca berdasarkan kota, format:
   http://api.openweathermap.org/data/2.5/weather?q=Malang&units=metric&appid=API_KEY

3. Memastikan koneksi HTTP berhasil (kondisi httpCode > 0), dan memproses JSON untuk mendapatkan:
   - "temp" dari bagian main
   - "description" dari bagian weather

3. HASIL DAN PEMBAHASAN

3.1 Hasil Praktik

Praktikum ini berhasil menampilkan data cuaca berupa suhu, dan deskripsi cuaca secara real-time pada LCD I2C 16x2 dengan menggunakan mikrokontroler ESP32 yang terkoneksi ke internet melalui jaringan Wi-Fi. Data diperoleh dari API OpenWeatherMap berdasarkan nama kota yang dikonfigurasi dalam program (contoh: Malang).

Setelah ESP32 berhasil terhubung ke Wi-Fi, ESP32 mengirimkan permintaan (HTTP Request) ke server OpenWeatherMap dan menerima respons berupa data cuaca dalam format JSON. Kemudian, data suhu ditampilkan pada baris pertama LCD, sedangkan deskripsi cuaca ditampilkan pada baris kedua.

Tabel berikut menunjukkan contoh hasil data yang berhasil ditampilkan:

Parameter	Nilai Sebelum Perubahan
Suhu	24.01 C
Deskripsi	Overcast clouds

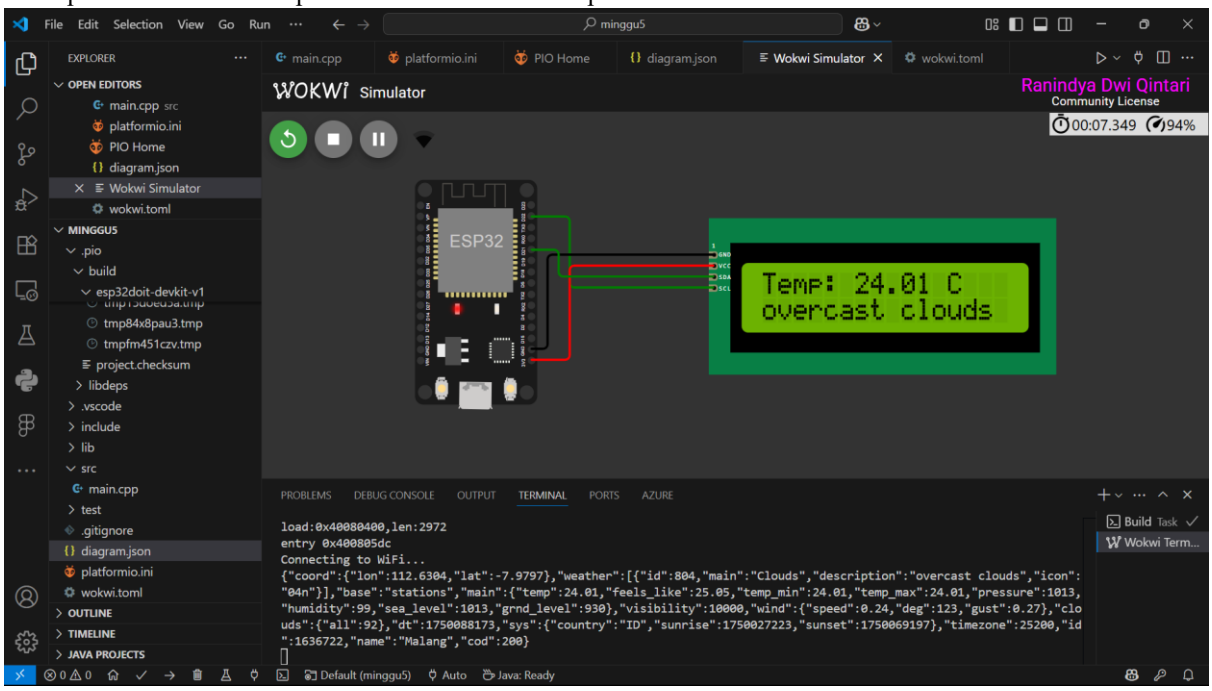
Tabel berikut menunjukkan contoh hasil data setelah ditambahkan untuk menampilan kelembapan:

Parameter	Nilai Sebelum Perubahan
Suhu	24.01 C
Kelembapan	99%
Deskripsi	Overcast clouds

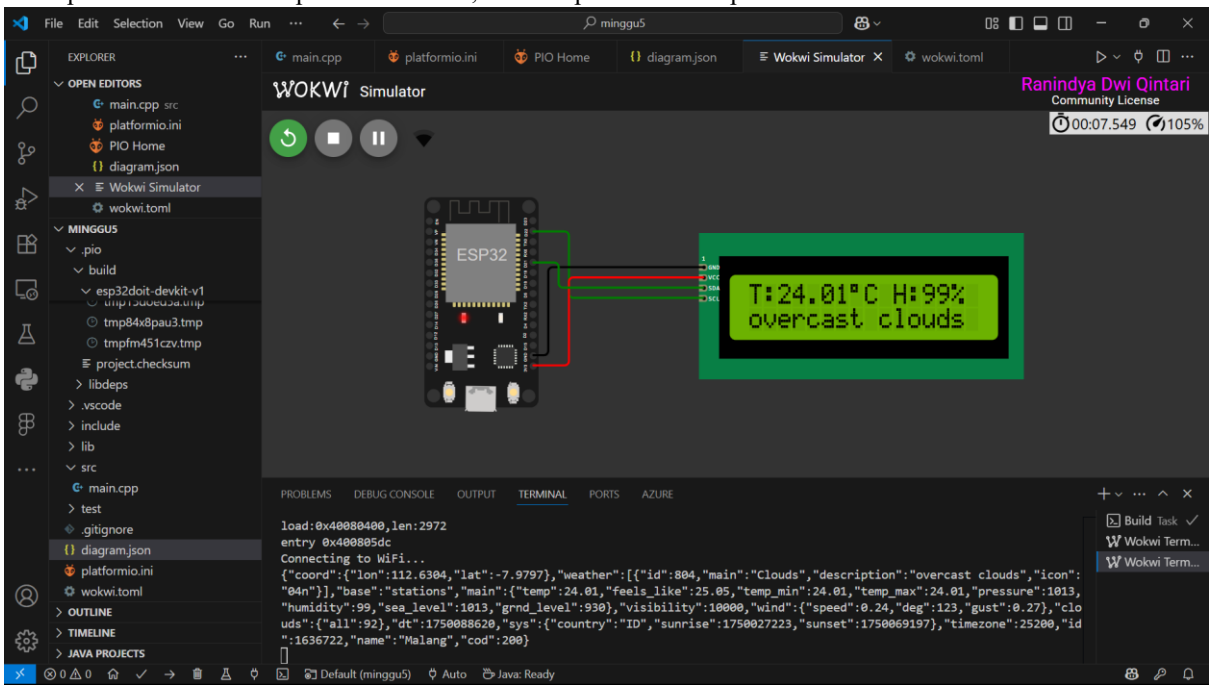
Data tersebut diperoleh langsung dari server OpenWeatherMap secara dinamis setiap 1 menit.

3.2 Tampilan Hasil Praktik

Hasil praktik untuk menampilkan data suhu dan deskripsi:



Hasil praktik untuk menampilkan data suhu, kelembapan dan deskripsi:



4. LAMPIRAN

4.1 Kode Program

- **main.cpp**

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>          // Ganti dengan ESP8266WiFi.h jika menggunakan ESP8266
#include <HTTPClient.h>

const char* ssid = "Wokwi-GUEST";        // Ganti dengan SSID Wi-Fi kamu
const char* password = ""; // Ganti dengan password Wi-Fi kamu
String apiKey = "20ca0ff523294dcdeb424dfc5802e21b";       // API Key dari OpenWeatherMap
String city = "Malang";              // Kota yang ingin ditampilkan
String units = "metric";              // Untuk Celsius gunakan "metric", untuk Fahrenheit
"imperial"
String server = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=" +
units + "&appid=" + apiKey;

LiquidCrystal_I2C lcd(0x27, 16, 2);   // Inisialisasi LCD dengan alamat I2C 0x27

void setup() {
  Serial.begin(115200);

  // Inisialisasi LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Weather Info:");

  // Inisialisasi Wi-Fi
```

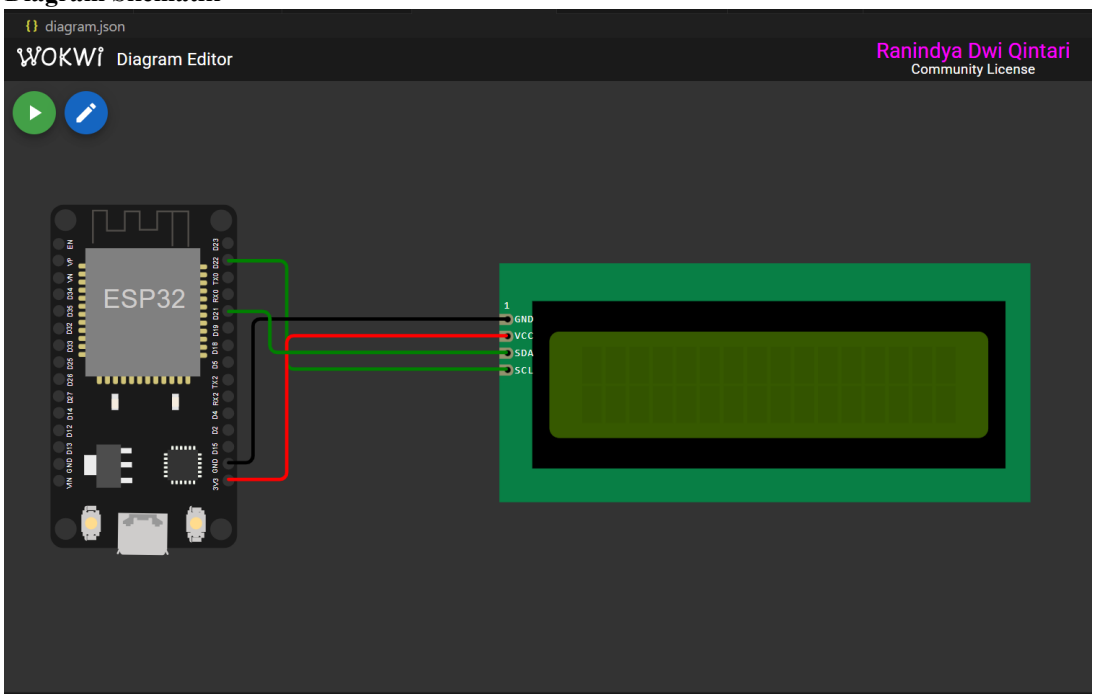
```
  WiFi.begin(ssid, password);
  lcd.setCursor(0, 1);
  lcd.print("Connecting...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connected!");
  delay(2000);
  lcd.clear();
}

void loop() {
  if ((WiFi.status() == WL_CONNECTED)) {  // Check WiFi connection status

    HTTPClient http;
    http.begin(server);     // Specify the URL
    int httpCode = http.GET();  // Make the request

    if (httpCode > 0) { // Check for the returning code

      String payload = http.getString();
      Serial.println(payload);   // Print the response payload

      // Parse data (extract temperature)
      int tempIndex = payload.indexOf("temp");
      String temp = payload.substring(tempIndex + 6, payload.indexOf(",", tempIndex));

      // Extract humidity
      int humIndex = payload.indexOf("humidity");
      String humidity = payload.substring(humIndex + 10, payload.indexOf(",", humIndex));

      // Display temperature and humidity on LCD (baris 0)
      lcd.setCursor(0, 0);
      lcd.print("T:");
      lcd.print(temp);
      lcd.print((char)223); // simbol derajat
      lcd.print("C H:");
      lcd.print(humidity);
      lcd.print("%");

      // Extract weather description
      int descIndex = payload.indexOf("description");
      String desc = payload.substring(descIndex + 14, payload.indexOf("\"", descIndex + 14));

      // Display description on LCD (baris 1)
      lcd.setCursor(0, 1);
```

```
      lcd.print("                "); // Clear baris kedua
      lcd.setCursor(0, 1);
      lcd.print(desc);

    } else {
      Serial.println("Error on HTTP request");
    }

    http.end();  // Free the resources
  }

  delay(60000);  // Update every minute
}
```

- **diagram.json**

```
{
  "version": 1,
  "author": "Ranindya Dwi Qintari",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    {
      "type": "wokwi-lcd1602",
      "id": "lcd1",
      "top": 35.2,
      "left": 255.2,
      "attrs": { "pins": "i2c" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "lcd1:SCL", "esp:D22", "green", [ "h-124.8", "v-66.9" ] ],
    [ "lcd1:VCC", "esp:3V3", "red", [ "h-124.8", "v86.5" ] ],
    [ "lcd1:GND", "esp:GND.1", "black", [ "h-144", "v57.6" ] ],
    [ "lcd1:SDA", "esp:D21", "green", [ "h-134.4", "v-28.6" ] ]
  ],
  "dependencies": {}
}
```

4.2 Diagram Skematik



5. PENUTUP

5.1 Kesimpulan

Dari hasil eksperimen, dapat disimpulkan bahwa ESP32 mampu terhubung ke jaringan internet dan mengambil data cuaca secara real-time dari OpenWeatherMap API. Data yang berhasil ditampilkan mencakup suhu, kelembapan, dan deskripsi cuaca, yang kemudian ditampilkan pada LCD I2C 16x2. Sistem bekerja secara otomatis untuk memperbarui data setiap satu menit tanpa perlu interaksi manual, menunjukkan keberhasilan integrasi antara perangkat keras (ESP32 dan LCD) dan layanan cloud (OpenWeatherMap). Praktik ini membuktikan bahwa ESP32 dapat digunakan dalam proyek IoT sederhana untuk melakukan monitoring kondisi lingkungan berdasarkan data eksternal melalui API, dan menampilkannya langsung melalui perangkat fisik.