**Laporan Pratikum Internet of Things (IOT) Sistem Informasi Cuaca Realtime Menggunakan API *OpenWeaterMap***

**Oleh**

*Maynanda Elisa Pasya T*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*maynandatbg@student.ub.ac.id*](mailto:maynandatbg@student.ub.ac.id)

**ABSTRAK**

Perkembangan teknologi Internet of Things (IoT) memungkinkan perangkat fisik terhubung ke internet dan saling bertukar data secara realtime. Dalam praktikum ini, dirancang dan diimplementasikan sebuah sistem informasi cuaca realtime berbasis ESP32 dengan memanfaatkan API dari OpenWeatherMap. Sistem ini mampu menampilkan informasi cuaca seperti suhu, kelembapan, dan kondisi langit pada layar LCD 16x2, serta memungkinkan pengguna untuk menavigasi antar data menggunakan tombol. Proyek dikembangkan menggunakan simulator Wokwi untuk perancangan awal dan Visual Studio Code dengan PlatformIO untuk implementasi akhir. ESP32 terhubung ke jaringan WiFi dan secara berkala mengirim permintaan HTTP ke server OpenWeatherMap, kemudian memproses data JSON yang diterima. Hasil pengujian menunjukkan bahwa sistem bekerja dengan baik dalam menampilkan data cuaca secara realtime dan dapat dengan mudah disesuaikan untuk berbagai lokasi. Sistem ini diharapkan dapat menjadi solusi sederhana dan efektif untuk kebutuhan monitoring cuaca berbasis IoT.

*Kata Kunci*: *Internet of Things, ESP32, OpenWeatherMap, real-time weather, LCD display, HTTP request, JSON parsing*

***ABTRACT***

*The development of Internet of Things (IoT) technology enables physical devices to connect to the internet and exchange data in real time. In this practical project, a real-time weather information system was designed and implemented using an ESP32 microcontroller and the OpenWeatherMap API. The system is capable of displaying weather data such as temperature, humidity, and sky conditions on a 16x2 LCD screen, with navigation between data points using push buttons. The project was initially developed using the Wokwi simulator and later implemented in Visual Studio Code with PlatformIO. The ESP32 connects to a WiFi network and periodically sends HTTP requests to the OpenWeatherMap server, then parses the received JSON data. Test results show that the system functions properly in displaying real-time weather information and can be easily adapted for different locations. This system offers a simple and effective solution for weather monitoring using IoT technology*.

*Keywords: Internet of Things, ESP32, OpenWeatherMap, real-time weather, LCD display, HTTP request, JSON parsing*

1. **Pendahuluan**
   1. Latar Belakang

Internet of Things (IoT) telah menjadi teknologi yang memungkinkan perangkat elektronik berkomunikasi melalui internet. Salah satu implementasinya adalah pemantauan cuaca secara real-time yang sangat penting untuk berbagai sektor seperti pertanian, pariwisata, dan transportasi. Dengan memanfaatkan layanan API cuaca dari OpenWeatherMap dan mikrokontroler seperti ESP32, kita dapat membuat sistem informasi cuaca sederhana namun fungsional.

* 1. Tujuan
* Membangun sistem yang dapat menampilkan informasi cuaca real-time.
* Mengakses data cuaca dari OpenWeatherMap melalui koneksi WiFi.
* Menampilkan data seperti suhu, kelembapan, dan kondisi cuaca pada layar LCD I2C.
* Melatih penggunaan simulator Wokwi dan IDE PlatformIO (Visual Studio Code) dalam pengembangan sistem IoT.

1. **Metodologi**
   1. Alat dan Bahan

Alat:

* Laptop
* Platform Wokwi
* Platform Visual Studio Code (VSCode)
* Platform OpenWeatherMap

Bahan:

* ESP32 (Virtual dalam platform wokwi)
* I2C LCD 16x2
* Push Button
* Wifi
* Bahasa Pemrograman C++ Pustaka Arduino
* Library
* LiquidCrystal\_I2C.h
* Wifi.h
* Wire.h
* HTTPClient.h
* ArduinoJson.h
  1. Langkah Implementasi

1. Buka Visual Studio Code (VSCode).
2. Unduh dan install ekstensi **PlatformIO** dan **Wokwi** pada VSCode.
3. Buat proyek baru di PlatformIO dengan memilih board **ESP32** (misalnya: DOIT ESP32 DEVKIT V1).
4. Buka situs **Wokwi** (<https://wokwi.com>).
5. Buat proyek baru dan pilih board **ESP32**.
6. Tambahkan komponen **LCD 16x2** (dengan modul I2C) dan dua **push button** untuk navigasi.
7. Gunakan **library LiquidCrystal\_I2C** untuk mengontrol LCD dan **library WiFi.h** untuk menghubungkan ESP32 ke jaringan WiFi.
8. Tambahkan **HTTPClient.h** untuk mengambil data cuaca dari **API OpenWeatherMap.**
9. Buat kode program untuk membaca data dari API, parsing JSON, dan menampilkannya di LCD.
10. Daftarkan akun dan akses platform **OpenWeatherMap** (<https://openweathermap.org>) untuk mendapatkan **API key**.
11. Tentukan kota atau negara yang ingin ditampilkan datanya dengan menyisipkan nama kota ke dalam URL API.
12. Tambahkan mekanisme agar data cuaca diperbarui secara periodik, misalnya setiap satu menit.
13. Jika simulasi di Wokwi sudah lengkap, salin file **diagram.json** dari Wokwi ke proyek di VSCode agar bisa diintegrasikan.
14. Jalankan program melalui **PlatformIO** dan pastikan semua dependensi telah terpasang, lalu pantau hasilnya di serial monitor atau LCD.
15. **Hasil dan Pembahasan**

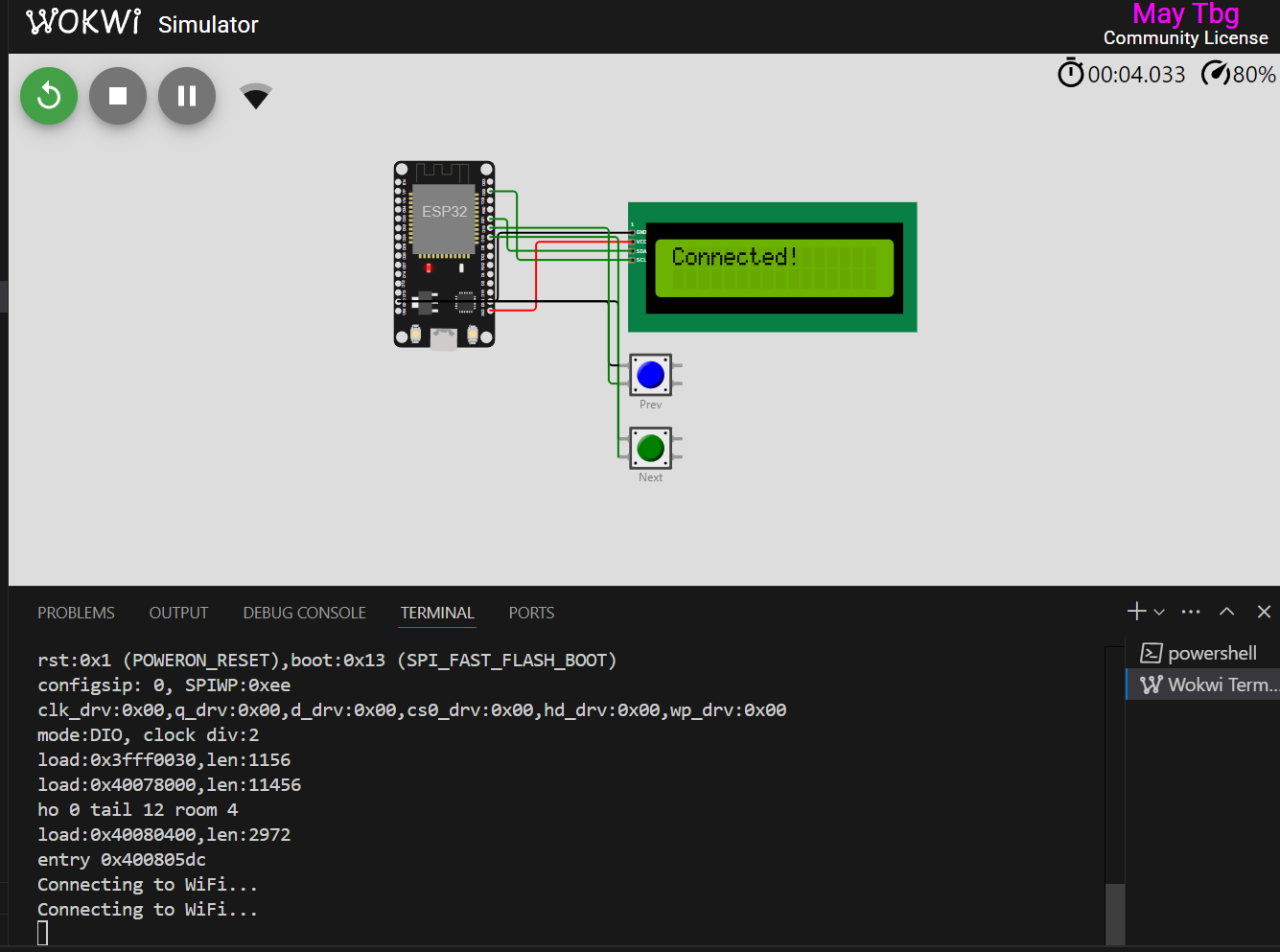
Setelah menyelesaikan proses implementasi baik di Wokwi maupun PlatformIO, sistem berhasil menjalankan fungsinya dengan baik. Ketika simulasi dijalankan, ESP32 berhasil melakukan koneksi ke jaringan WiFi dan mengirimkan permintaan HTTP ke server OpenWeatherMap. Server merespon dengan data cuaca dalam format JSON, yang kemudian berhasil diuraikan menggunakan library ArduinoJson.

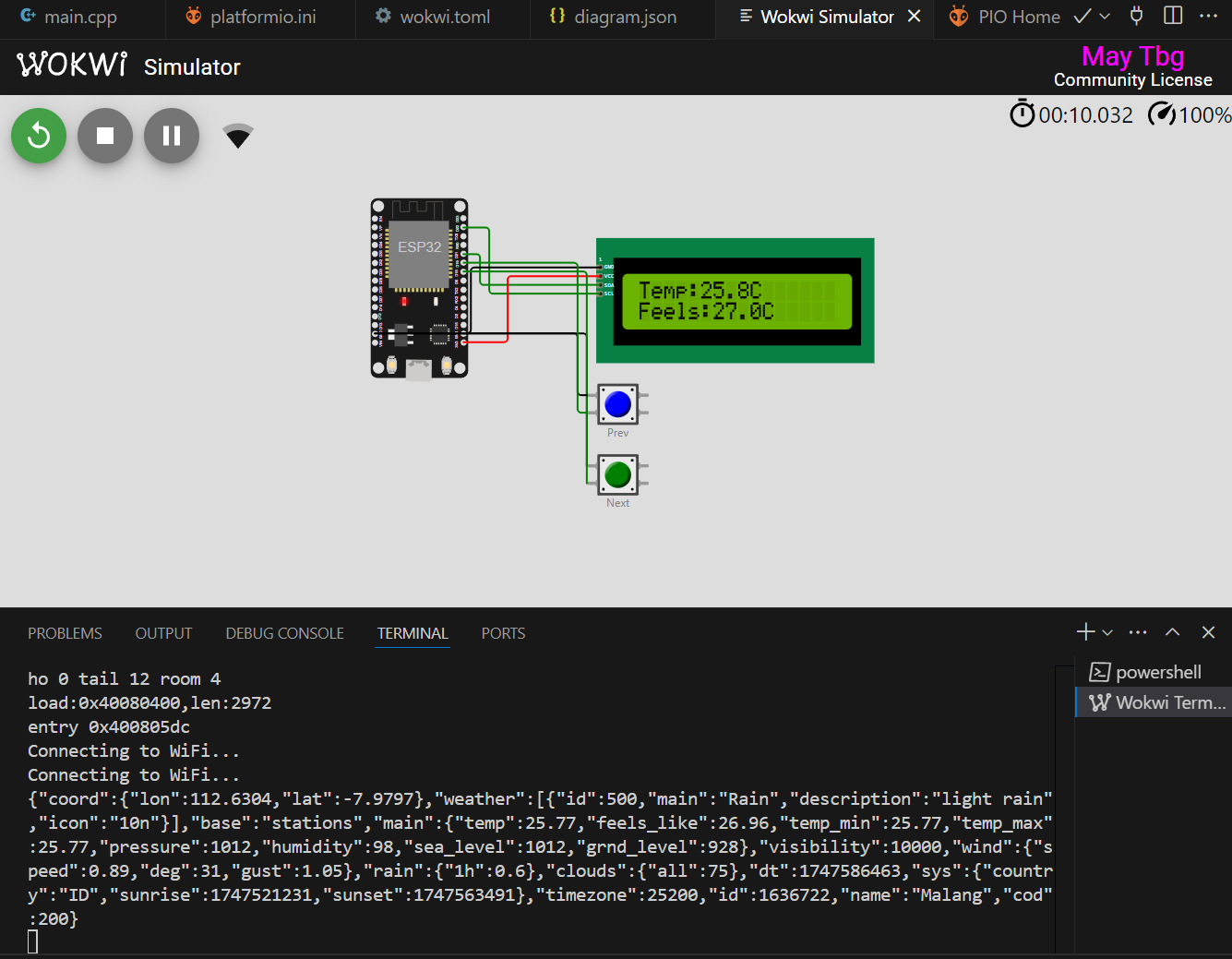
Informasi yang berhasil ditampilkan di layar LCD antara lain suhu udara saat ini, suhu terasa (feels like), kelembapan udara, dan deskripsi kondisi cuaca seperti “broken clouds” atau “clear sky”. Tampilan data ini dapat dinavigasi menggunakan dua tombol yang berfungsi sebagai "Next" dan "Prev", sehingga pengguna dapat berpindah antar informasi yang berbeda secara interaktif.

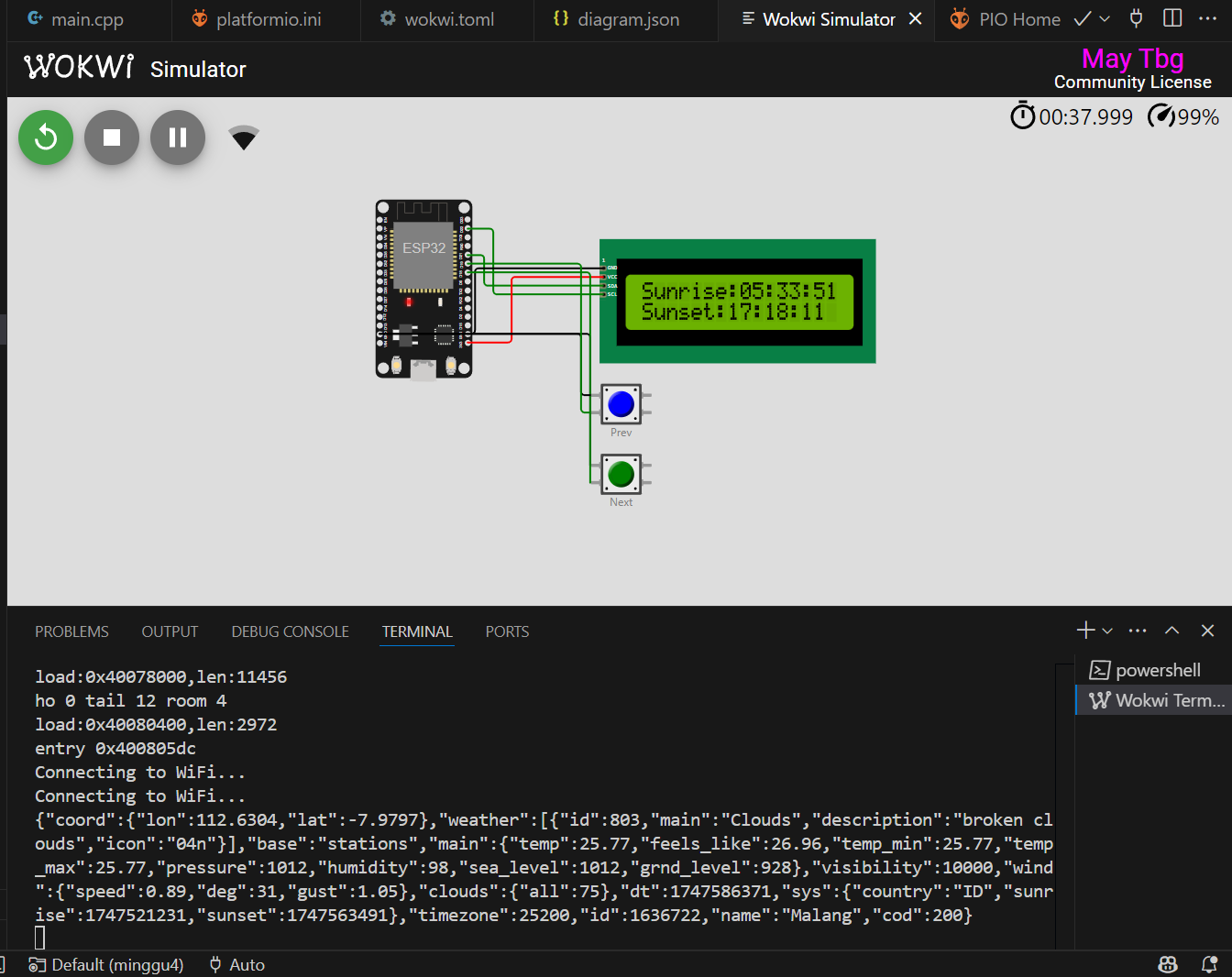
Penggunaan LCD I2C sangat membantu dalam menampilkan data secara sederhana dan jelas, serta memudahkan integrasi dengan ESP32 karena hanya menggunakan dua jalur komunikasi (SDA dan SCL). Selama pengujian di Wokwi, sistem berhasil menampilkan data dengan jeda yang cukup untuk terbaca dengan baik oleh pengguna, serta merespon tombol dengan stabil.

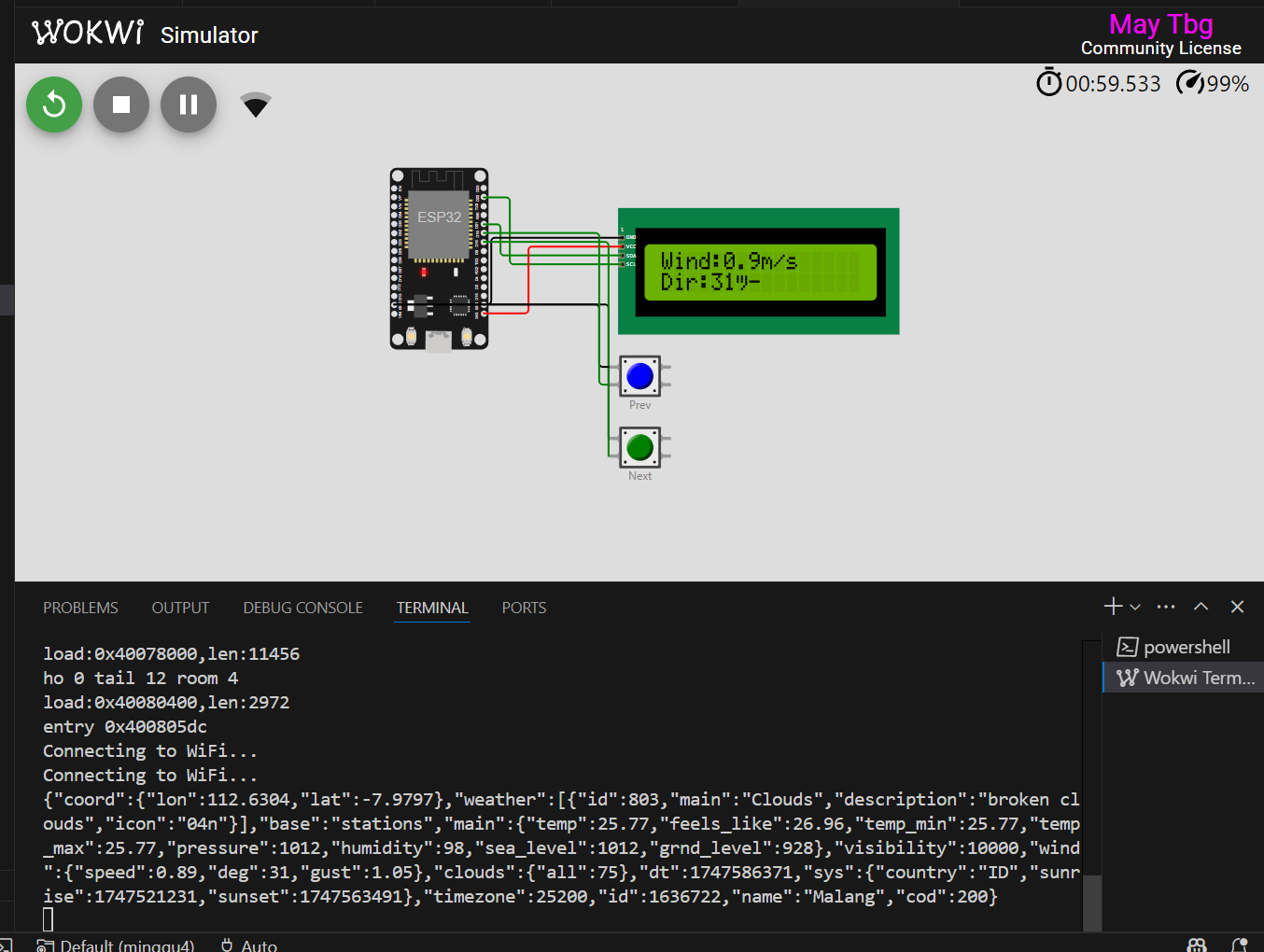
Dari sisi perangkat lunak, penggunaan PlatformIO memberikan lingkungan kerja yang terstruktur dan mendukung manajemen dependensi dengan mudah melalui file platformio.ini. Namun, tantangan sempat muncul ketika memilih library yang kompatibel, khususnya untuk LiquidCrystal\_I2C. Beberapa versi library tidak dapat digunakan, sehingga perlu dicoba beberapa opsi hingga menemukan versi yang sesuai dan berhasil dikompilasi tanpa error.

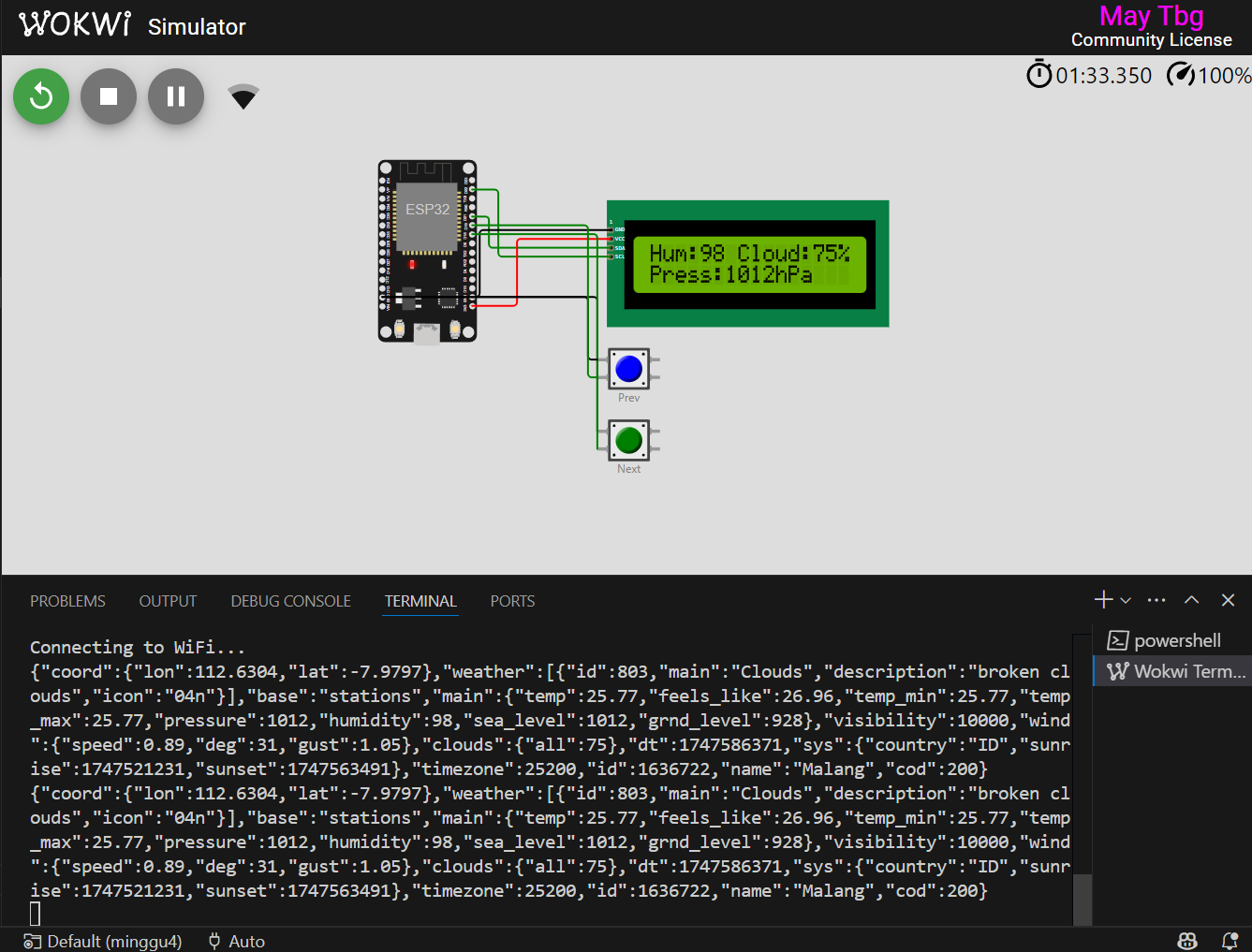
Secara keseluruhan, sistem ini berjalan sesuai harapan. Meskipun simulasi di Wokwi tidak benar-benar mengakses internet, struktur kode dan alur logika telah mewakili implementasi nyata secara akurat. Dengan demikian, sistem ini sudah siap untuk diunggah ke perangkat fisik dan digunakan dalam aplikasi monitoring cuaca berbasis IoT.











1. **Lampiran Kode**

* Kode Program Main.cpp

#include <Arduino.h>#include <Wire.h>#include <LiquidCrystal\_I2C.h>#include <WiFi.h>#include <HTTPClient.h>#include <ArduinoJson.h>// WiFi and APIconst char\* ssid = "Wokwi-GUEST";const char\* password = "";String apiKey = "4e3ff9f85087a3196c469c725af5f9f3"; String city = "Malang"; String units = "metric"; String server = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=" + units + "&appid=" + apiKey;// LCD setupLiquidCrystal\_I2C lcd(0x27, 16, 2);// Buttonsconst int nextButtonPin = 18;const int prevButtonPin = 19;int currentPage = 0;// Data dari serverString temp = "-";String feelsLike = "-";String desc = "-";String humidity = "-";String clouds = "-";String pressure = "-";String precipitation = "-";String windSpeed = "-";String windDeg = "-";String sunrise = "-";String sunset = "-";// Timerunsigned long lastUpdateTime = 0;const long updateInterval = 60000; // 1 menit// Fungsi Forwardvoid getWeatherData();void displayPage();String convertUnixTime(long unixTime);void setup() { Serial.begin(115200); pinMode(nextButtonPin, INPUT\_PULLUP); pinMode(prevButtonPin, INPUT\_PULLUP); lcd.init(); lcd.backlight(); lcd.setCursor(0, 0); lcd.print("Weather Info:"); WiFi.begin(ssid, password); lcd.setCursor(0, 1); lcd.print("Connecting..."); while (WiFi.status() != WL\_CONNECTED) { delay(500); Serial.println("Connecting to WiFi..."); } lcd.clear(); lcd.setCursor(0, 0); lcd.print("Connected!"); delay(2000); lcd.clear(); getWeatherData(); // Fetch pertama displayPage();}void loop() { if (digitalRead(nextButtonPin) == LOW) { currentPage++; if (currentPage > 4) currentPage = 0; // Update sesuai jumlah page displayPage(); delay(300); } if (digitalRead(prevButtonPin) == LOW) { currentPage--; if (currentPage < 0) currentPage = 4; // Update sesuai jumlah page displayPage(); delay(300); } if (millis() - lastUpdateTime >= updateInterval) { getWeatherData(); lastUpdateTime = millis(); } delay(100);}void getWeatherData() { if (WiFi.status() == WL\_CONNECTED) { HTTPClient http; http.begin(server); int httpCode = http.GET(); if (httpCode > 0) { String payload = http.getString(); Serial.println(payload); const size\_t capacity = 2048; DynamicJsonDocument doc(capacity); DeserializationError error = deserializeJson(doc, payload); if (!error) { temp = String(doc["main"]["temp"].as<float>(), 1); feelsLike = String(doc["main"]["feels\_like"].as<float>(), 1); desc = doc["weather"][0]["description"].as<String>(); humidity = String(doc["main"]["humidity"].as<int>()); clouds = String(doc["clouds"]["all"].as<int>()) + "%"; pressure = String(doc["main"]["pressure"].as<int>()) + "hPa"; windSpeed = String(doc["wind"]["speed"].as<float>(), 1) + "m/s"; windDeg = String(doc["wind"]["deg"].as<int>()) + "°"; sunrise = convertUnixTime(doc["sys"]["sunrise"].as<long>()); sunset = convertUnixTime(doc["sys"]["sunset"].as<long>()); // Periksa apakah ada hujan if (doc.containsKey("rain") && doc["rain"].containsKey("1h")) { precipitation = String(doc["rain"]["1h"].as<float>()) + "mm"; } else { precipitation = "0mm"; } } else { Serial.println("Failed to parse JSON"); } displayPage(); } else { Serial.println("HTTP Request error"); } http.end(); } else { Serial.println("WiFi disconnected"); }}void displayPage() { lcd.clear(); switch (currentPage) { case 0: lcd.setCursor(0, 0); lcd.print("Temp:" + temp + "C"); lcd.setCursor(0, 1); lcd.print("Feels:" + feelsLike + "C"); break; case 1: lcd.setCursor(0, 0); lcd.print("Weather:"); lcd.setCursor(0, 1); lcd.print(desc); break; case 2: lcd.setCursor(0, 0); lcd.print("Hum:" + humidity + " "); lcd.print("Cloud:" + clouds); lcd.setCursor(0, 1); lcd.print("Press:" + pressure); break; case 3: lcd.setCursor(0, 0); lcd.print("Wind:" + windSpeed); lcd.setCursor(0, 1); lcd.print("Dir:" + windDeg); break; case 4: lcd.setCursor(0, 0); lcd.print("Sunrise:" + sunrise); lcd.setCursor(0, 1); lcd.print("Sunset:" + sunset); break; }}String convertUnixTime(long unixTime) { const int timeOffset = 7 \* 3600; // WIB (UTC+7) unixTime += timeOffset; time\_t rawtime = unixTime; struct tm \* ti; ti = gmtime(&rawtime); char buffer[9]; sprintf(buffer, "%02d:%02d:%02d", ti->tm\_hour, ti->tm\_min, ti->tm\_sec); return String(buffer);}

* Kode Program diagram.json

{ "version": 1, "author": "Maynanda", "editor": "wokwi", "parts": [ { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 4.7, "left": 4.6, "attrs": {} }, { "type": "wokwi-lcd1602", "id": "lcd", "top": 50, "left": 250, "attrs": { "pins": "i2c" } }, { "type": "wokwi-pushbutton", "id": "btnNext", "top": 284.6, "left": 240, "attrs": { "color": "green", "label": "Next" } }, { "type": "wokwi-pushbutton", "id": "btnPrev", "top": 207.8, "left": 240, "attrs": { "color": "blue", "label": "Prev" } } ], "connections": [ [ "esp:TX0", "$serialMonitor:RX", "", [] ], [ "esp:RX0", "$serialMonitor:TX", "", [] ], [ "lcd:SCL", "esp:D22", "green", [ "h-120", "v-50" ] ], [ "lcd:SDA", "esp:D21", "green", [ "h-130", "v-20" ] ], [ "lcd:VCC", "esp:3V3", "red", [ "h-100", "v30" ] ], [ "lcd:GND", "esp:GND.1", "black", [ "h-140", "v50" ] ], [ "btnNext:1.l", "esp:GND.2", "black", [ "h0", "v-30" ] ], [ "btnNext:2.l", "esp:D18", "green", [ "h0", "v-70" ] ], [ "btnPrev:1.l", "esp:GND.2", "black", [ "h-10", "v-40" ] ], [ "btnPrev:2.l", "esp:D19", "green", [ "h-10", "v-80" ] ] ], "dependencies": {}}