PRAKTIKUM SIMULASI SISTEM MONITORING CUACA DENGAN SCROLL DATA LCD BERBASIS API OPENWEATHER

TB Rangga Gilang Yanuari
Fakultas Vokasi, Universitas Brawijaya
gilangyanuarirangga@gmail.com

ABSTRAK

Dalam era Internet of Things (IoT), monitoring data lingkungan secara real-time menjadi sangat penting. Praktikum ini mensimulasikan sistem pemantauan cuaca berbasis ESP32 dengan mengambil data dari API OpenWeather dan menampilkannya pada LCD 16x2 melalui komunikasi I2C. Data yang ditampilkan mencakup suhu, suhu terasa (feels like), curah hujan (precipitation), kecepatan angin, arah angin, kelembapan, persentase tutupan awan (clouds), tekanan udara, waktu matahari terbit (sunrise), dan waktu matahari terbenam (sunset). Sistem ini mendukung navigasi antar informasi menggunakan tombol Next dan Prev untuk scrolling data. Simulasi dilakukan menggunakan platform Wokwi, yang memungkinkan pengujian tanpa perangkat keras fisik. Hasil simulasi menunjukkan bahwa sistem mampu menampilkan informasi cuaca secara dinamis, responsif, dan sesuai dengan urutan data yang ditentukan.
Kata Kunci: IoT, ESP32, LCD I2C, API OpenWeather, Wokwi

ABSTRACT

In the era of the Internet of Things (IoT), real-time environmental data monitoring is crucial. This practicum simulates a weather monitoring system based on ESP32 by utilizing data from the OpenWeather API and displaying it on a 16x2 LCD via I2C communication. The displayed data includes temperature, feels like temperature, precipitation, wind speed, wind direction, humidity, cloud coverage percentage, air pressure, sunrise time, and sunset time. The system supports navigation between information points using Next and Prev buttons for data scrolling. The simulation is conducted using the Wokwi platform, enabling testing without physical hardware. Simulation results show that the system is able to dynamically and responsively display weather information in an orderly manner.
Keywords: IoT, ESP32, LCD I2C, API OpenWeather, Wokwi

1. PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) memungkinkan berbagai aplikasi inovatif dalam pemantauan kondisi lingkungan. Salah satu contohnya adalah memanfaatkan data cuaca dari layanan publik seperti OpenWeather untuk keperluan monitoring. Dengan ESP32 sebagai mikrokontroler yang mendukung Wi-Fi, sistem ini dapat terhubung ke API secara real-time untuk memperoleh data cuaca. Data yang diperoleh kemudian ditampilkan pada LCD 16x2, dengan kemampuan scrolling menggunakan tombol untuk navigasi antar informasi.

1.2 Tujuan Praktikum

Praktikum ini bertujuan untuk:

1) Memahami konsep dasar pemanfaatan API dalam sistem IoT.
2) Mempelajari cara membaca data cuaca dari API OpenWeather.
3) Mengimplementasikan tampilan data cuaca pada LCD 16x2 berbasis I2C.
4) Mengaplikasikan navigasi antar data menggunakan tombol Next dan Prev.
5) Menggunakan Wokwi sebagai alat simulasi.

2. METODOLOGI

2.1 Alat dan Bahan

- **Alat:**
  a) Laptop/computer
  b) Wokwi Simulator
  c) VSCode + PlatformIO
- **Bahan:**
  a) ESP32 Devkit V1.
  b) LCD 16x2 I2C
  c) Dua Pushbutton (Next dan Prev).
  d) Koneksi Wi-Fi (Wokwi Guest)

2.2 Langkah Implementasi

Berikut adalah langkah-langkah implementasi secara rinci dalam melakukan simulasi sistem monitoring Cuaca dengan scroll data LCD berbasis API OpenWeather ESP32 di Wokwi dan PlatformIO:

1. **Simulasi di Wokwi Web**
   1) Buka https://wokwi.com.
   2) Klik **New Project** → Pilih **ESP32**.
   3) Tambahkan komponen:
      a) ESP32 Devkit V1
         - SDA ke pin D21
         - SCL ke pin D22
         - VCC ke 3V3
         - GND ke GND
      b) Tombol Next:
         - Salah satu kaki ke D12
         - Kaki lainnya ke GND
      c) Tombol Prev:

- Salah satu kaki ke D14
- Kaki lainnya ke GND

4) Tulis dan upload kode program di Wokwi Web.
5) Klik **Start Simulation** untuk menjalankan dan menguji apakah:
   a) LCD menampilkan data cuaca.
   b) Tombol Next dan Prev berfungsi untuk scroll data.
6) Setelah berhasil, Lanjut melakukan simulasi di Wokwi Vscode (opsional)

2. Simulasi di Wokwi VSCode (PlatformIO)

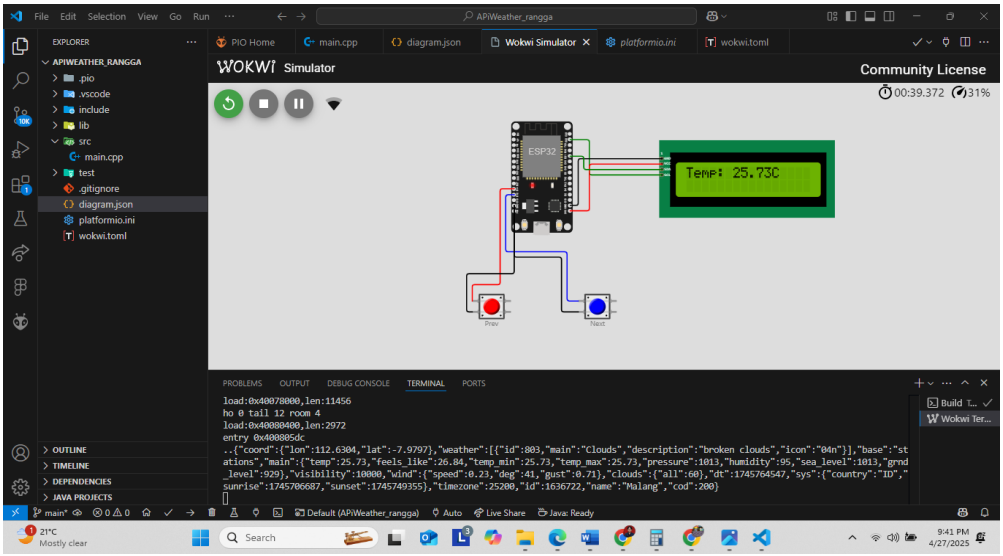
1) Buka **Visual Studio Code**.
2) Pastikan sudah menginstall **PlatformIO IDE** Extension.
3) Buat **New Project** di PlatformIO:
   a) Board: ESP32 Devkit V1 (esp32doit-devkit-v1)
   b) Framework: Arduino.
4) Di dalam project:
   a) Masukkan kode program ke src/main.cpp.
   b) Copy diagram.json hasil export ke folder project.
5) Buat file baru bernama wokwi.toml di root project, isi dengan:

```
[wokwi]
version = 1
firmware = ".pio/build/esp32doit-devkit-v1/firmware.bin"
elf = ".pio/build/esp32doit-devkit-v1/firmware.elf"
```

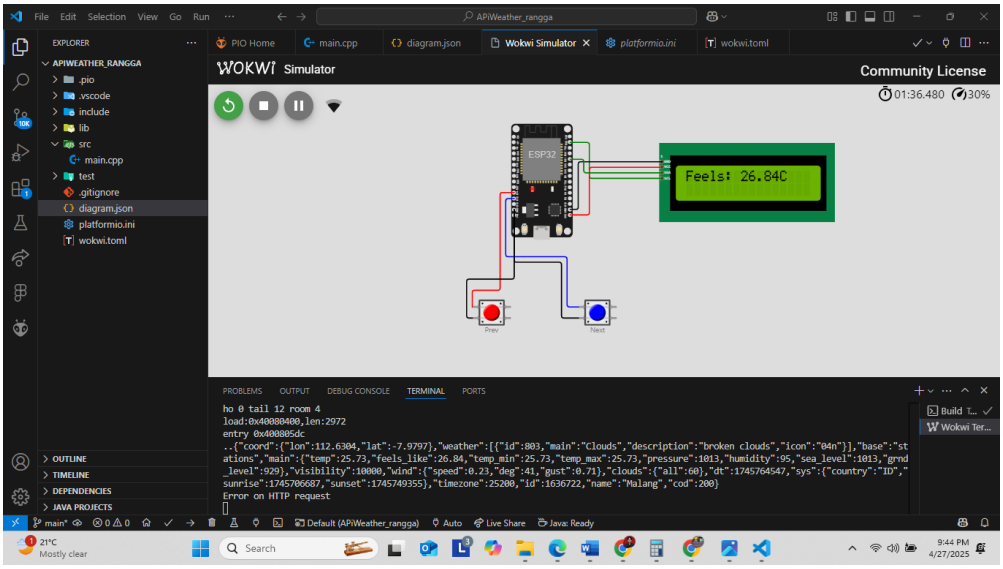
6) Klik **Build** (tanda centang di pojok kiri bawah) untuk compile program.
7) Setelah build sukses:
   a) Buka diagram.json.
   b) Klik tombol **Start Simulation** (ikon di kiri atas).
8) Simulasi berjalan di Wokwi langsung dari VSCode

3. HASIL DAN PEMBAHASAN

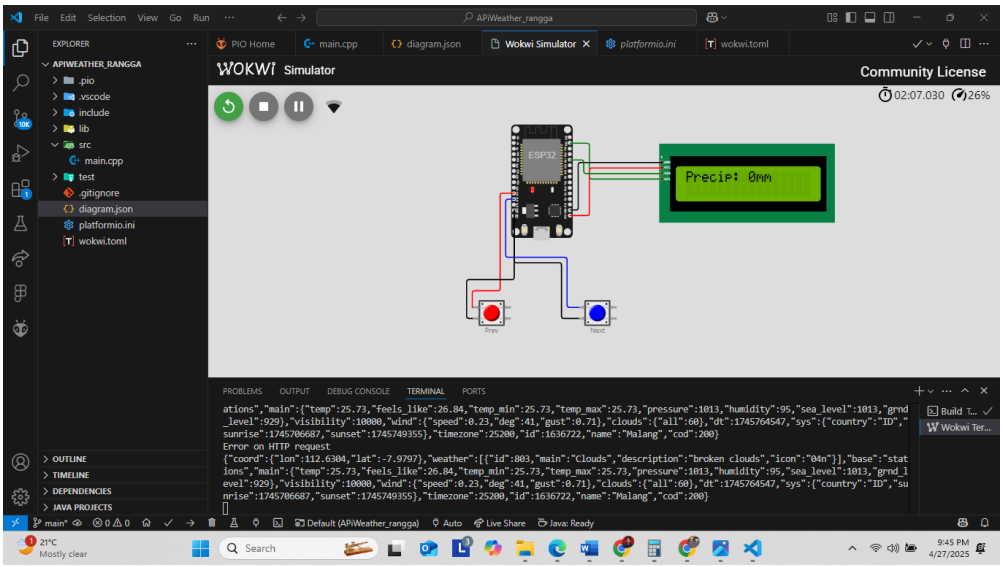
3.1 Hasil Tampilan Temperature



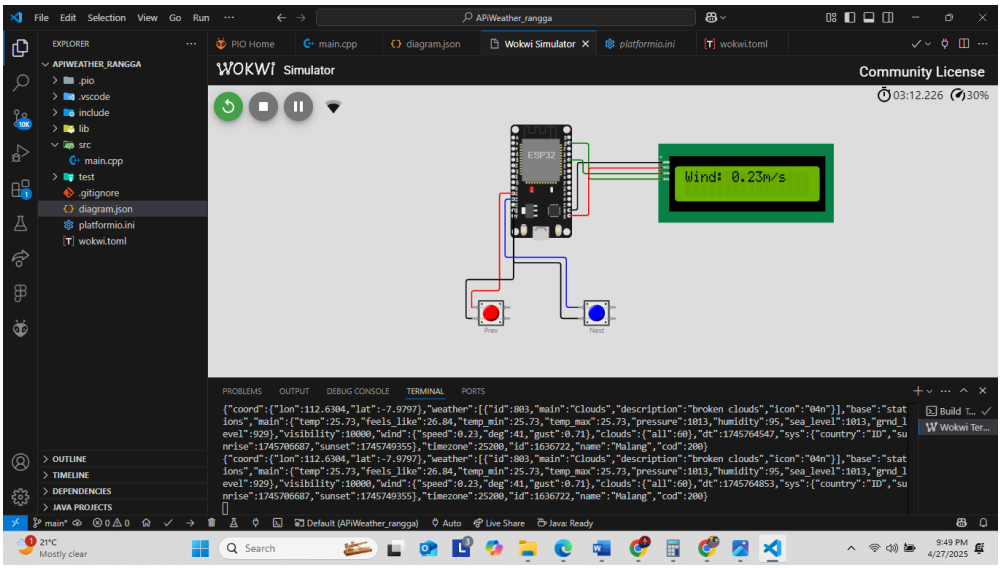
3.2 Hasil Tampilan Feels Like



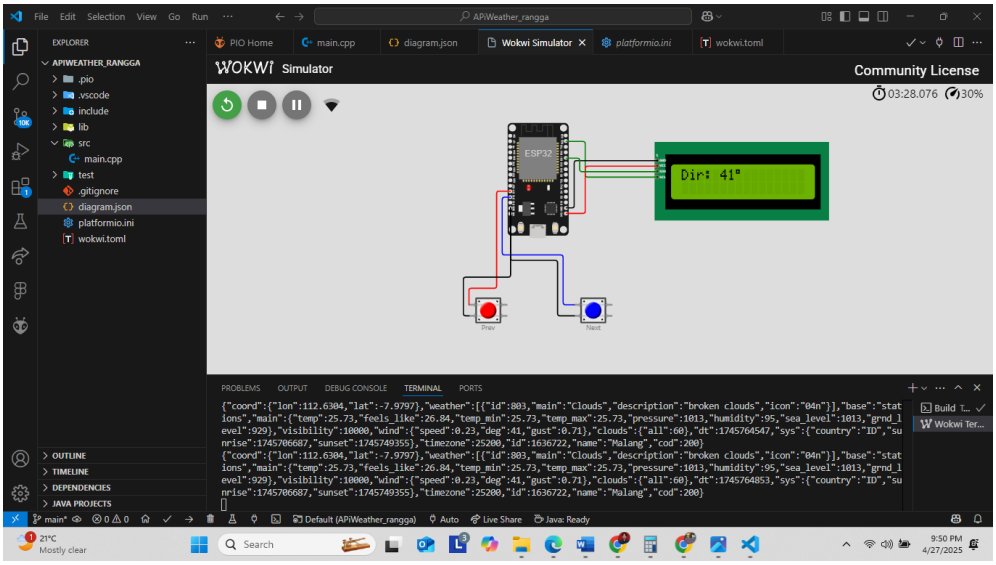
3.3 Hasil Tampilan Precipitation



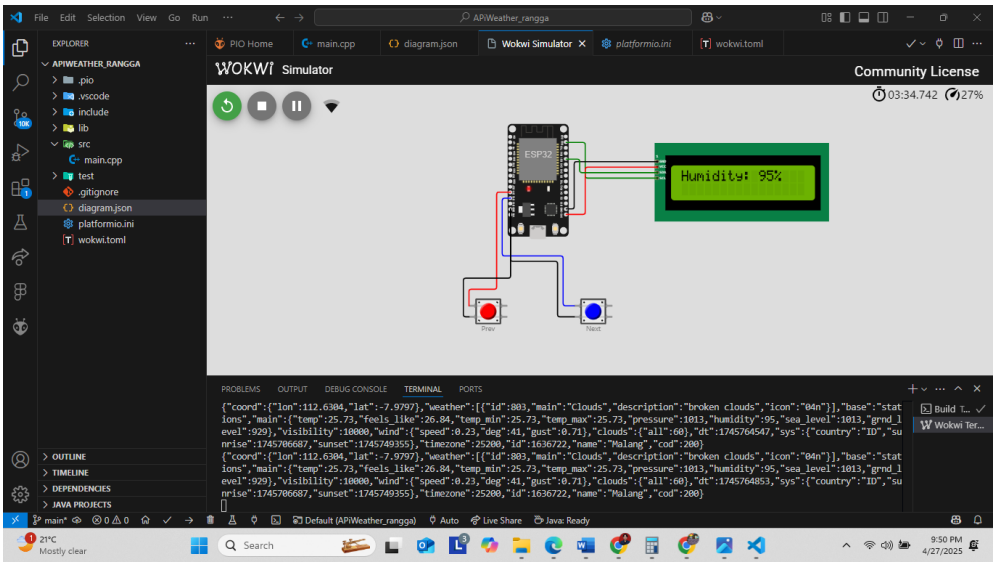
3.4 Hasil Tampilan Wind Speed



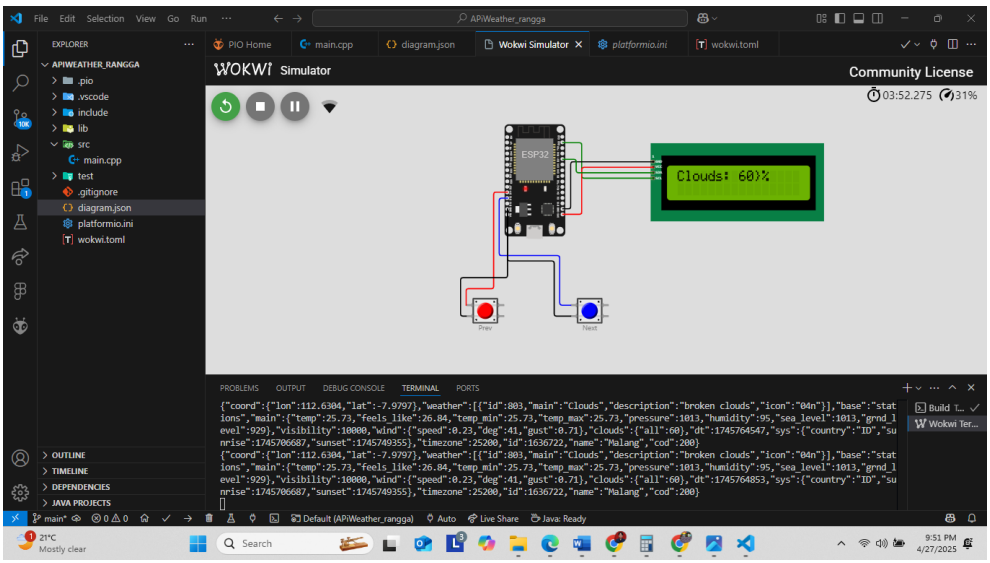
3.5 Hasil Tampilan Wind Direction



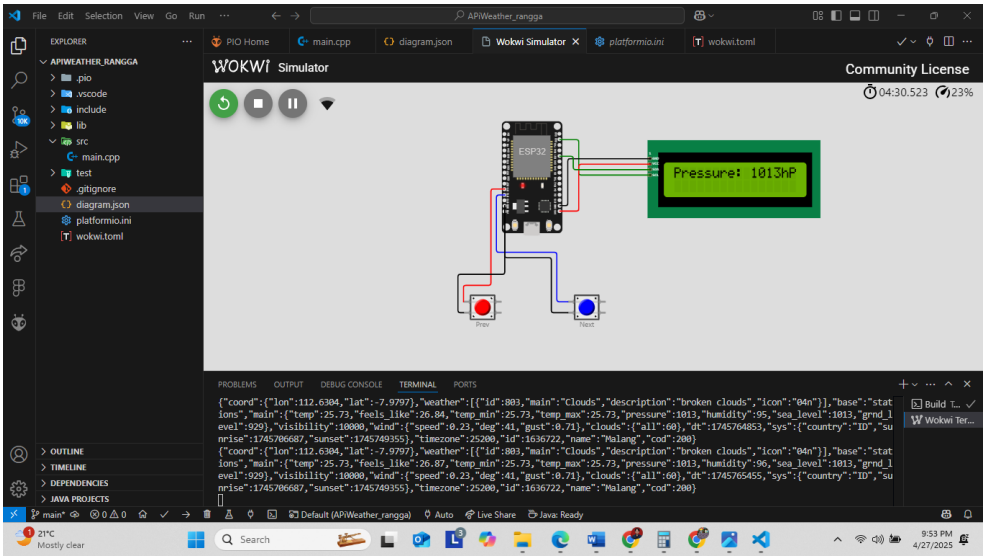
3.5 Hasil Tampilan Humadity



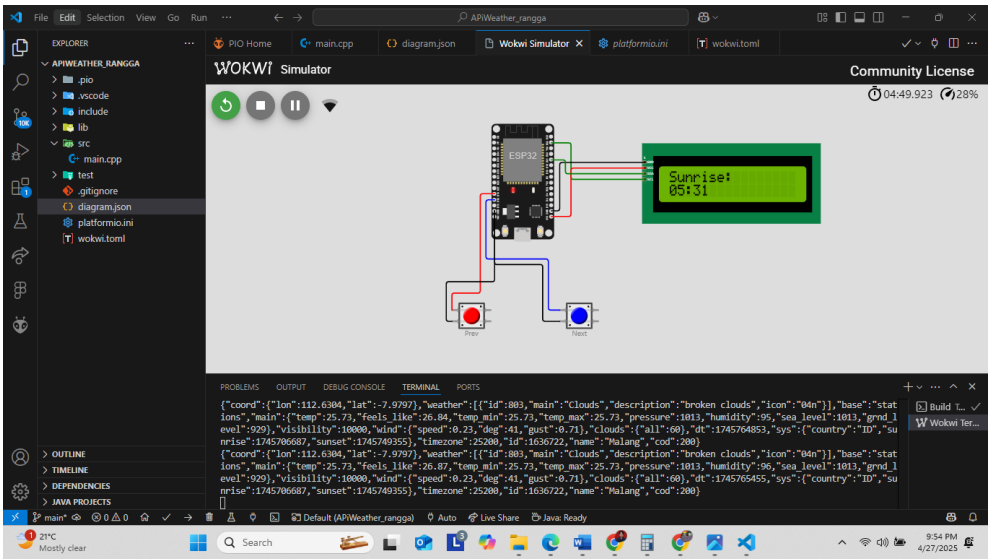
3.6 Hasil Tampilan Clouds



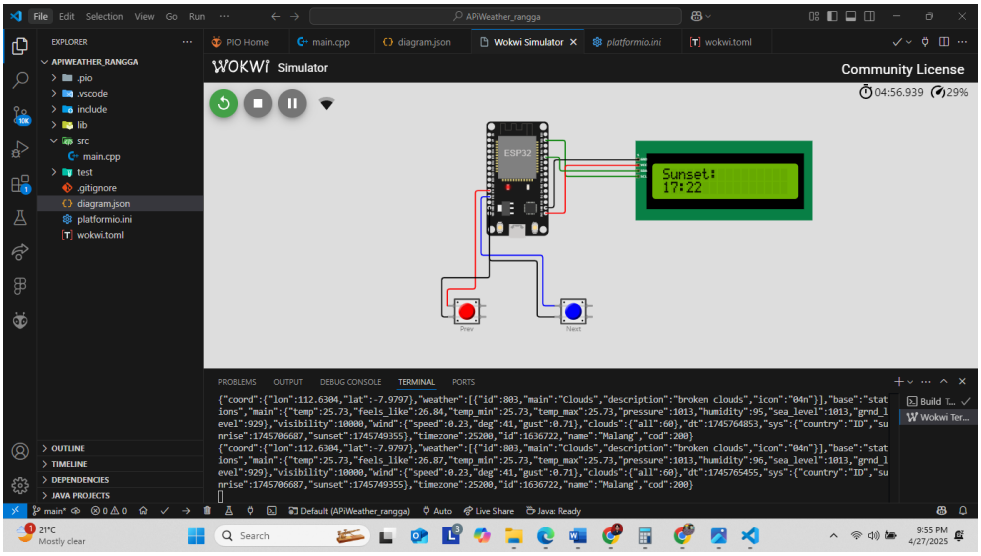
3.7 Hasil Tampilan Pressure



3.8 Hasil Tampilan Sunrise



3.9 Hasil Tampialn Sunset



3.10 Pembahasan

Pada praktikum ini, sistem monitoring cuaca berbasis ESP32 berhasil dibuat dan disimulasikan menggunakan platform Wokwi. Sistem ini mengambil data cuaca real-time dari API OpenWeather, yang kemudian ditampilkan secara bergilir (scroll) di layar LCD 16x2 I2C.

Di awal sistem dihidupkan, LCD menampilkan informasi identitas pembuat, yaitu tulisan "Created Rangga" sebagai salam pembuka. Setelah itu, sistem berusaha menghubungkan ke jaringan Wi-Fi dengan SSID "Wokwi-GUEST". Setelah koneksi berhasil, sistem akan mengambil data cuaca dari server OpenWeather menggunakan API Key yang sudah ditentukan.

Data cuaca yang berhasil diambil mencakup:

- Temperatur (Temp)
- Temperatur terasa (Feels Like)
- Curah hujan (Precipitation)
- Kecepatan angin (Wind Speed)
- Arah angin (Wind Direction)
- Kelembapan udara (Humidity)
- Persentase tutupan awan (Clouds)
- Tekanan udara (Pressure)
- Waktu matahari terbit (Sunrise)
- Waktu matahari terbenam (Sunset)

Tiap data ditampilkan satu per satu pada LCD, dengan navigasi menggunakan tombol Next (D12) untuk maju dan tombol Prev (D14) untuk mundur. Ketika tombol Next ditekan, LCD akan menampilkan informasi berikutnya. Sebaliknya, jika tombol Prev ditekan, LCD akan menampilkan informasi sebelumnya. Sistem ini mampu menampilkan semua informasi secara berurutan sesuai dengan struktur data API OpenWeather.

Pada bagian waktu (Sunrise dan Sunset), dilakukan konversi dari UNIX Timestamp ke waktu lokal Indonesia Barat (WIB) dengan menambahkan offset +7 jam dari waktu UTC. Ini dilakukan agar waktu yang ditampilkan sesuai dengan zona waktu tempat pengamatan (Malang).

Selain itu, fungsi parsing data JSON dari API tidak menggunakan library tambahan seperti ArduinoJson. Parsing dilakukan secara manual menggunakan fungsi getValue() dan getOptionalValue(), sehingga program menjadi lebih ringan dan sederhana.

Namun, karena simulasi dilakukan pada platform Wokwi Web, data yang diterima adalah dummy data (data tiruan) yang tidak berubah-ubah. Oleh karena itu, nilai suhu, kelembapan, atau waktu Sunrise dan Sunset mungkin sedikit berbeda dibandingkan jika program dijalankan di ESP32 asli dengan koneksi internet sebenarnya.

Secara keseluruhan, praktikum ini menunjukkan bahwa sistem monitoring cuaca berbasis IoT sederhana dapat dibuat dan disimulasikan dengan baik. Semua komponen seperti ESP32, Wi-Fi, HTTP Client, parsing data API, LCD display, dan navigasi tombol berfungsi sesuai harapan.

4. LAMPIRAN

4.1 Kode program untuk Menampilkan data dari API OpenWeather

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "Wokwi-GUEST";
const char* password = "";
String apiKey = "0c854b30810ba8e3e1f55b2eeabe6da6";
String city = "Malang";
String units = "metric";
String server = "http://api.openweathermap.org/data/2.5/weather?q=" + city +
"&units=" + units + "&appid=" + apiKey;

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int buttonNext = 12;
const int buttonPrev = 14;

int scrollIndex = 0;
unsigned long lastUpdate = 0;
unsigned long updateInterval = 60000;

// Data weather
String temp = "N/A";
String feels_like = "N/A";
String precipitation = "0"; // Default 0 mm
String windSpeed = "N/A";
String windDirection = "N/A";
String humidity = "N/A";
String clouds = "N/A";
String pressure = "N/A";
String sunrise = "N/A";
String sunset = "N/A";

// Function prototypes
void fetchWeather();
void displayData(int index);
String getValue(String data, String key);
String getOptionalValue(String data, String key);
String formatTime(String unixTimeStr);

void setup() {
```

```
  Serial.begin(115200);

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

  pinMode(buttonNext, INPUT_PULLUP);
  pinMode(buttonPrev, INPUT_PULLUP);

  // Tampilan awal nama pembuat
  lcd.setCursor(0, 0);
  lcd.print("Created by");
  lcd.setCursor(0, 1);
  lcd.print("TB Rangga Gilang");
  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Yanuari");
  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connecting...");

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connected!");
  delay(1000);
  lcd.clear();

  fetchWeather(); // Fetch pertama
  displayData(scrollIndex);
}

void loop() {
  unsigned long currentMillis = millis();

  // Update data tiap 60 detik
  if (currentMillis - lastUpdate > updateInterval) {
```

```
    lastUpdate = currentMillis;
    fetchWeather();
    displayData(scrollIndex);
  }

  // Tombol Next
  if (digitalRead(buttonNext) == LOW) {
    scrollIndex++;
    if (scrollIndex > 9) scrollIndex = 0;
    displayData(scrollIndex);
    delay(300); // Debounce
  }

  // Tombol Prev
  if (digitalRead(buttonPrev) == LOW) {
    scrollIndex--;
    if (scrollIndex < 0) scrollIndex = 9;
    displayData(scrollIndex);
    delay(300); // Debounce
  }
}

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);

      temp = getValue(payload, "temp");
      feels_like = getValue(payload, "feels_like");
      humidity = getValue(payload, "humidity");
      windSpeed = getValue(payload, "speed");
      windDirection = getValue(payload, "deg");
      pressure = getValue(payload, "pressure");
      clouds = getValue(payload, "all");
      precipitation = getOptionalValue(payload, "\"rain\":{\"1h\":");
      sunrise = formatTime(getValue(payload, "sunrise"));
      sunset = formatTime(getValue(payload, "sunset"));

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

```
    }
    http.end();
  }
}

void displayData(int index) {
  lcd.clear();
  lcd.setCursor(0, 0);

  switch(index) {
    case 0:
      lcd.print("Temp: " + temp + "C");
      break;
    case 1:
      lcd.print("Feels: " + feels_like + "C");
      break;
    case 2:
      lcd.print("Precip: " + precipitation + "mm");
      break;
    case 3:
      lcd.print("Wind: " + windSpeed + "m/s");
      break;
    case 4:
      lcd.print("Dir: " + windDirection + (char)223);
      break;
    case 5:
      lcd.print("Humidity: " + humidity + "%");
      break;
    case 6:
      lcd.print("Clouds: " + clouds + "%");
      break;
    case 7:
      lcd.print("Pressure: " + pressure + "hPa");
      break;
    case 8:
      lcd.print("Sunrise:");
      lcd.setCursor(0, 1);
      lcd.print(sunrise);
      break;
    case 9:
      lcd.print("Sunset:");
      lcd.setCursor(0, 1);
      lcd.print(sunset);
      break;
  }
```

```
}

// Fungsi parsing data biasa
String getValue(String data, String key) {
  int startIndex = data.indexOf(key);
  if (startIndex == -1) return "N/A";
  startIndex = data.indexOf(":", startIndex) + 1;
  int endIndex = data.indexOf(",", startIndex);
  if (endIndex == -1) endIndex = data.indexOf("}", startIndex);
  String value = data.substring(startIndex, endIndex);
  value.trim();
  return value;
}

// Fungsi parsing rain opsional
String getOptionalValue(String data, String key) {
  int start = data.indexOf(key);
  if (start == -1) return "0"; // kalau tidak ada hujan, return 0
  start += key.length();
  int end = data.indexOf(",", start);
  if (end == -1) end = data.indexOf("}", start);
  String value = data.substring(start, end);
  value.trim();
  return value;
}

// Format UNIX Time ke WIB jam:menit
String formatTime(String unixTimeStr) {
  unsigned long unixTime = unixTimeStr.toInt();

  // Tambah offset 7 jam untuk WIB
  unixTime += 7 * 3600;

  int hours = (unixTime % 86400L) / 3600;
  int minutes = (unixTime % 3600) / 60;

  char buffer[6];
  sprintf(buffer, "%02d:%02d", hours, minutes);
  return String(buffer);
}
5.
```

5.1 Kode diagram.json

```
{
    "version": 1,
    "author": "Rangga",
    "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" }
      },
      {
        "type": "wokwi-pushbutton",
        "id": "btn1",
        "top": 303.8,
        "left": -67.2,
        "attrs": { "color": "red", "xray": "1", "label": "Prev" }
      },
      {
        "type": "wokwi-pushbutton",
        "id": "btn2",
        "top": 303.8,
        "left": 115.2,
        "attrs": { "color": "blue", "xray": "1", "label": "Next" }
      }
    ],
    "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" ] ],
      [ "btn2:2.l", "esp:GND.2", "black", [ "h-28.8", "v-96", "h-172.8" ] ],
      [ "btn2:1.l", "esp:D12", "blue", [ "h-19.2", "v-86.4", "h-105.6", "v-
105.6" ] ],
      [ "btn1:1.l", "esp:D14", "red", [ "h0", "v-28.8", "h48", "v-163.2" ] ],
      [ "btn1:2.l", "esp:GND.2", "black", [ "h-9.6", "v-67", "h86.4" ] ]
    ],
    "dependencies": {}
  }
```