LAPORAN PRAKTIKUM INTERNET OF THINGS (IOT) SISTEM INFORMASI CUACA *REALTIME* MENGGUNAKAN API *OPENWEATHERMAP*

FAKULTAS VOKASI UNIVERSITAS BRAWIJAYA

Rifcha Sya'bani Fatullah

Fakultas Vokasi Universitas Brawijaya

Email: rifchasyabani30@gmail.com

ABSTRAK

Tujuan dari praktik ini adalah untuk membuat sistem informasi cuaca menggunakan Arduino yang terhubung ke WiFi dan menampilkan data cuaca secara real-time pada layar LCD berbasis I2C. Sistem ini memanfaatkan API OpenWeatherMap untuk mengumpulkan data seperti tekanan udara, kecepatan angin, suhu, kelembapan, dan waktu terbit dan terbenam matahari. Dengan menggunakan dua tombol, pengguna dapat menavigasi antar berbagai tampilan cuaca dan melihat informasi cuaca yang berbeda. Setiap menit, data cuaca akan diperbarui untuk memberikan informasi terbaru. Simulator Wokwi dan VSCode sebagai editor teks digunakan untuk menjalankan proyek ini. Hasil pengujian menunjukkan bahwa sistem menampilkan dan memperbarui data cuaca secara akurat secara berkala.

Kata Kunci : ESP32, Wokwi, *OpenWeatherMap*, *Internet Of Things*, *API*

ABTRACT

The goal of this project is to create a weather information system that uses an Arduino, WiFi, and an I2C LCD screen to display current weather data. The system retrieves meteorological information, including temperature, humidity, air pressure, wind speed, and dawn and sunset times, via the OpenWeatherMap API. Two buttons allow users to view different weather information and switch between different weather displays. To give you the most recent information, the weather data is updated every minute. This project was implemented using VSCode as a text editor and the Wokwi simulator. The system properly shows correct weather information and periodically refreshes the data, according to the test findings.

Keywords: ESP32, Wokwi, *OpenWeatherMap*, *Internet Of Things*, *API*

1. PENDAHULUAN

1.1 Latar Belakang

Cuaca adalah salah satu komponen yang mempengaruhi aktivitas sehari-hari, sangat penting untuk mendapatkan informasi cuaca yang akurat dan terkini. Arduino adalah platform open-source yang mudah diakses yang dipilih untuk membuat alat informasi cuaca yang dapat menampilkan suhu, kelembapan, tekanan udara, kecepatan angin, waktu terbit dan terbenam matahari. Dengan menggunakan WiFi pada board ESP32, alat ini dapat mengakses data cuaca langsung dari internet melalui *API OpenWeatherMap*. Dengan layar LCD berbasis I2C yang menampilkan data yang berubah-ubah, pengguna dapat melihat informasi cuaca. Praktik ini dilakukan dengan simulator Wokwi dan editor teks Visual Studio Code.

1.2 Tujuan

Adapun tujuan dari praktikum ini ialah :

1. Mengembangkan sistem informasi cuaca berbasis Arduino yang dapat terhubung ke internet dengan menggunakan WiFi.
2. Menampilkan data cuaca secara *real-time* pada LCD melalui antarmuka I2C.
3. Memahami cara mendapatkan data cuaca menggunakan API dan mengolahnya menggunakan Arduino.

2. METODOLOGI

2.1 Alat dan Bahan

Alat :

a. Laptop
b. Platform Wokwi
c. Platform Visual Studio Code (VSCode)
d. Platform OpenWeatherMap

Bahan :

a. ESP32 (Virtual dalam platform wokwi)
b. I2C LCD 16x2
c. Push Button
d. Wifi
e. Bahasa Pemrograman C++ Pustaka Arduino
f. Library
    - LiquidCrystal_I2C.h
    - Wifi.h
    - Wire.h
    - HTTPClient.h
    - ArduinoJson.h

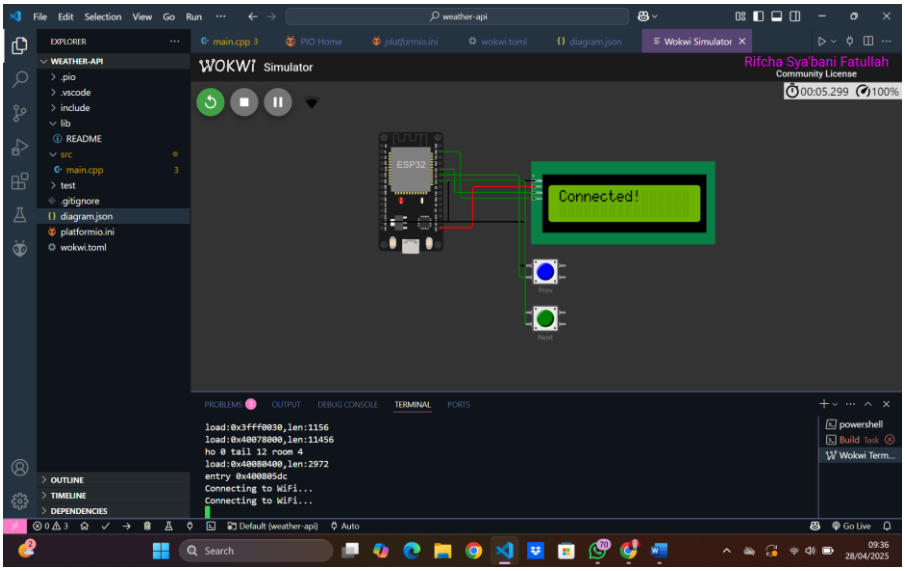
2.2 Langkah Implementasi

1. Buka Visual Studio Code (VSCode)
2. Unduh dan install ekstensi PlatformIO dan Wokwi pada VSCode
3. Buat proyek baru di Platform untuk ESP32
4. Buka situs Wokwi
5. Buat proyek baru
6. Tambahkan sensor LCD 16x2 dan dua *push button*
7. Gunakan library LiquidCrystal_I2C untuk mengontrol LCD dan WIFI untuk menghubungkan ESP32 ke jaringan wifi
8. HTTPClient untuk meminta data cuaca dari API OpenWeatherMap
9. Membuat kode program
10. Akses platform OpenWeatherMap untuk mendapatkan API key
11. Pilih kota atau negara yang ingin ditampilkan data cuacanya
12. Buat agar data cuaca di perbarui setiap satu menit untuk memperoleh data cuaca terbaru
13. Jika sudah merancang diagram di wokwi, selanjutnya salin kode diagram.json ke vscode
14. Jalankan Program

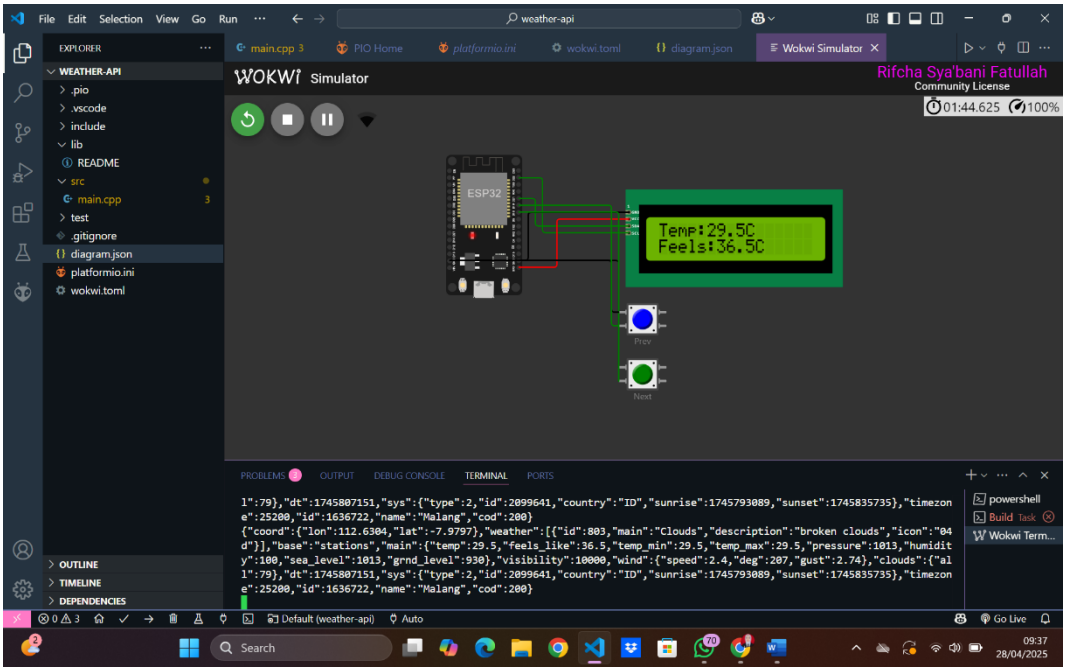
3. Hasil dan Pembahasan

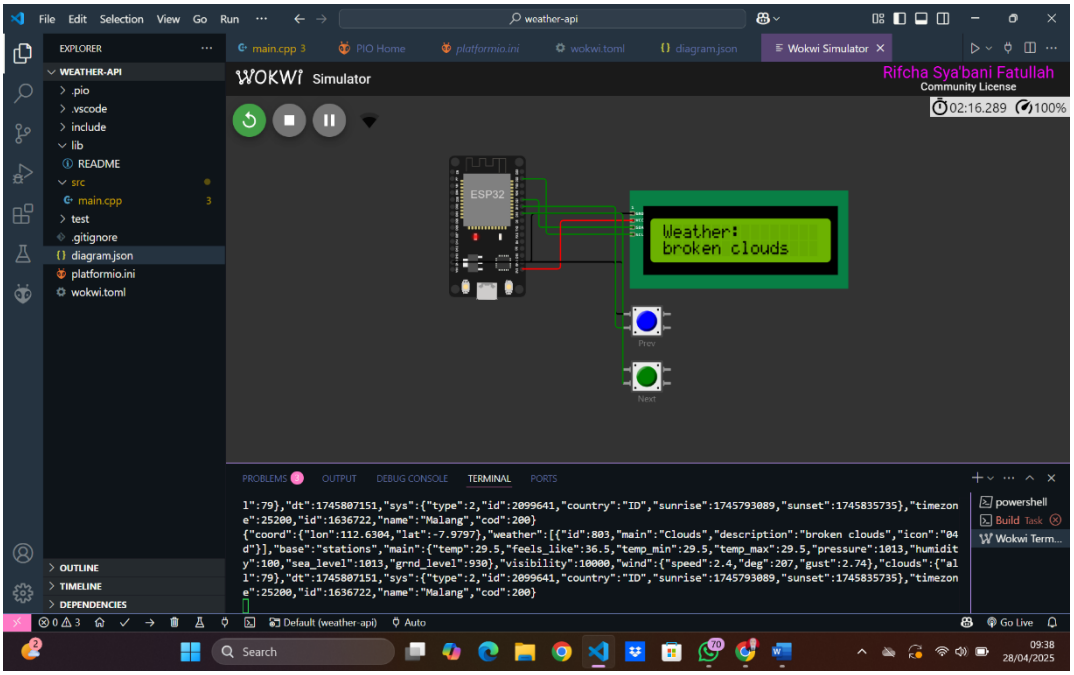
3.1 Hasil Eksperimen

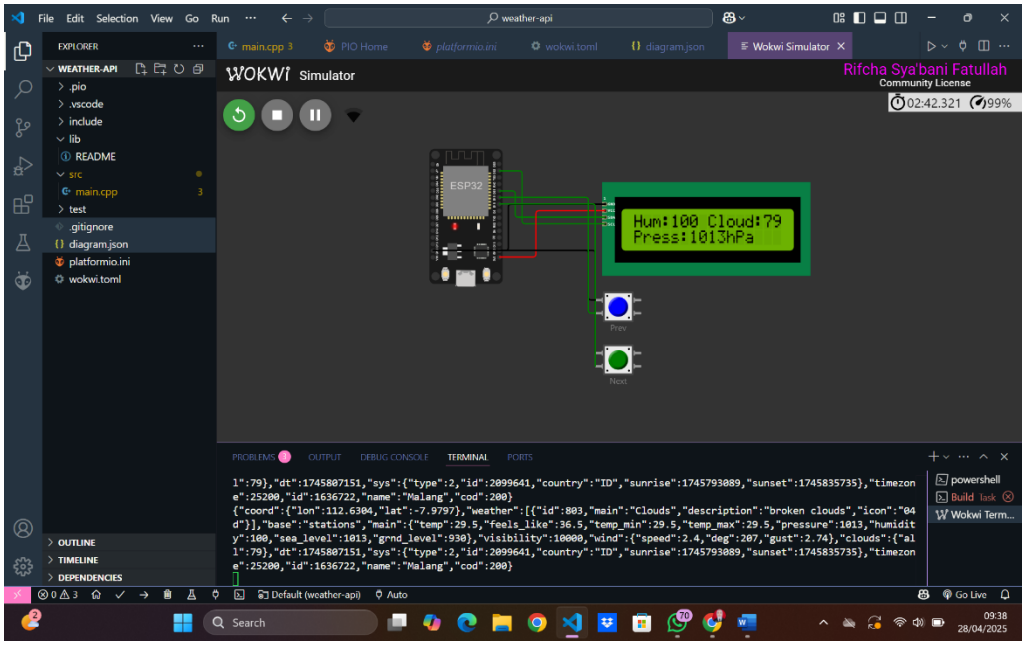
a. Sistem berhasil terhubung ke WIFI dan menerima data cuaca secara realtime

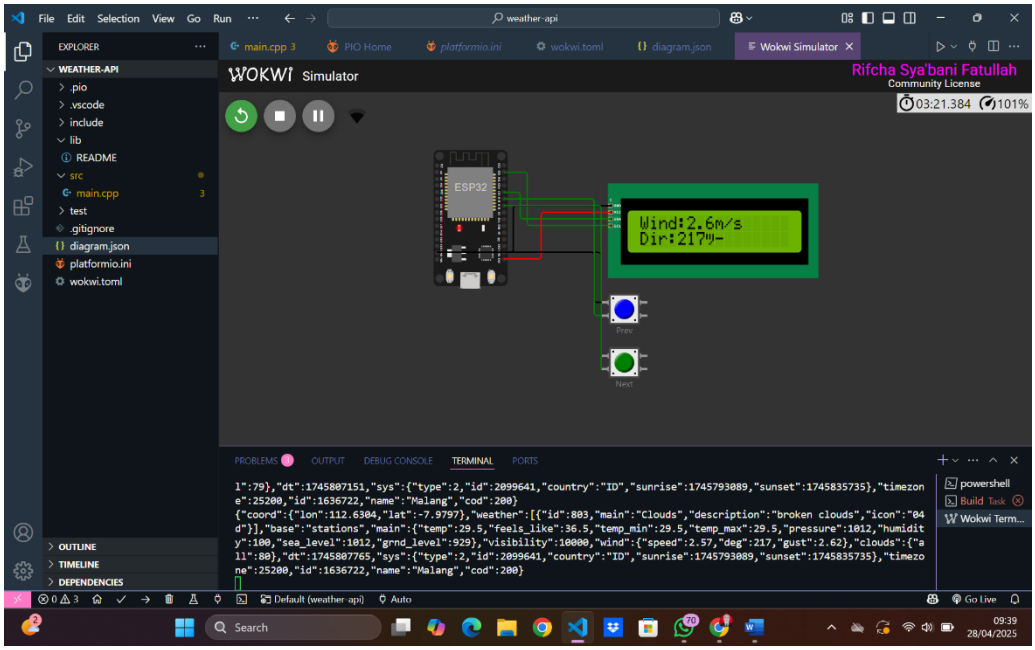


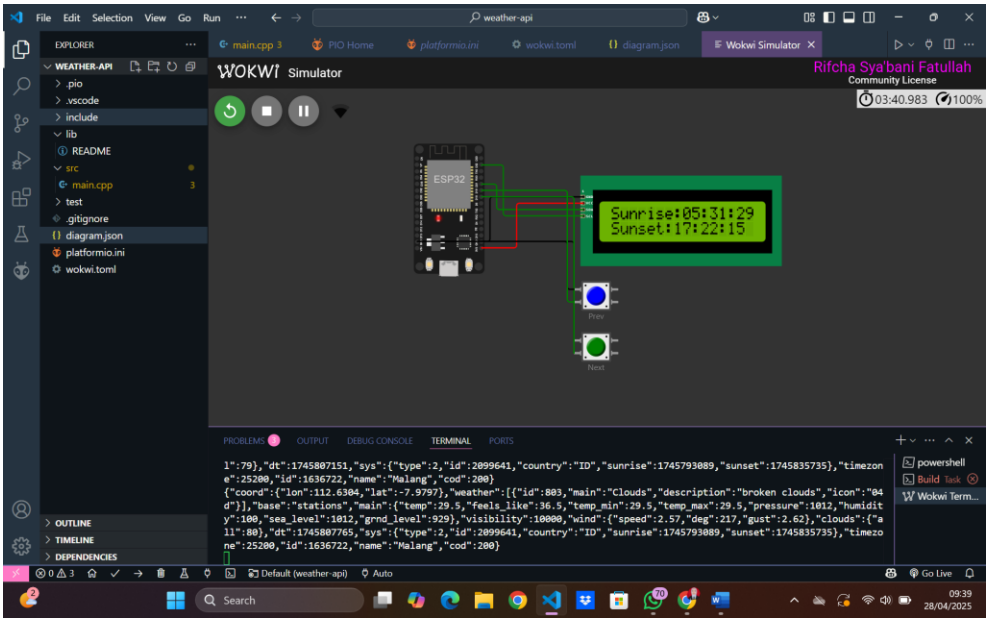
b. **Data yang ditampilkan pada lcd meliputi suhu, kondisi cuaca, kelembapan, tekanan udara, kecepatan angin, arah nagin, waktu terbit dan terbenam matahari**






File Edit Selection View Go Run
weather-api
EXPLORER
WEATHER-API
.pio
.vscode
include
lib
README
src
main.cpp
test
.gitignore
diagram.json
platformio.ini
wokwi.toml
main.cpp 3
PIO Home
platformio.ini
wokwi.toml
diagram.json
Wokwi Simulator
WOKWI Simulator
Rifcha Sya'bani Fatullah
Community License
02:42.321
99%
ESP32
Hum:100 Cloud:79
Press:1013hPa
Prev
Next
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
Build Task
Wokwi Term...
OUTLINE
TIMELINE
DEPENDENCIES
Default (weather-api)
Auto
Go Live
Search
09:38
28/04/2025


File Edit Selection View Go Run
weather-api
EXPLORER
WEATHER-API
.pio
.vscode
include
lib
README
src
main.cpp
test
.gitignore
diagram.json
platformio.ini
wokwi.toml
main.cpp 3
PIO Home
platformio.ini
wokwi.toml
diagram.json
Wokwi Simulator
WOKWI Simulator
Rifcha Sya'bani Fatullah
Community License
03:21.384
101%
ESP32
Wind:2.6m/s
Dir:217°
Prev
Next
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
Build Task
Wokwi Term...
OUTLINE
TIMELINE
DEPENDENCIES
Default (weather-api)
Auto
Go Live
Search
09:39
28/04/2025



3.2 Pembahasan

Projek ini menunjukkan bagaimana Arduino dapat digunakan untuk mengakses data cuaca *real-time* dan menampilkannya pada layar LCD dengan jelas. Koneksi ke *API OpenWeatherMap* dan pengolahan data JSON berjalan dengan lancar, tetapi ada beberapa masalah dengan menangani data besar yang diterima dari API.

1. Keterbatasan LCD
   Karena ukuran LCD 16x2, hanya satu halaman dapat menampilkan informasi. Oleh karena itu, Anda harus menggunakan tombol untuk beralih antara halaman informasi.
2. Koneksi WiFi
   Simulator Wokwi kadang-kadang memerlukan waktu lebih lama untuk stabil untuk koneksi WiFi, yang dapat mempengaruhi pengambilan data cuaca.
3. Pengguna dapat beralih antara halaman informasi cuaca dengan menekan tombol.
4. Data cuaca diperbarui setiap satu menit untuk memberikan informasi terbaru.

4. Lampiran

a. 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 API
const 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 setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Buttons
const int nextButtonPin = 18;
const int prevButtonPin = 19;
int currentPage = 0;

// Data dari server
String temp = "-";
String feelsLike = "-";
String desc = "-";
String humidity = "-";
String clouds = "-";
String pressure = "-";
String precipitation = "-";
String windSpeed = "-";
String windDeg = "-";
String sunrise = "-";
String sunset = "-";

// Timer
unsigned long lastUpdateTime = 0;
const long updateInterval = 60000; // 1 menit

// Fungsi Forward
void 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);
}
```

b. Kode Program diagram.json

```
{
  "version": 1,
  "author": "Rifcha Syabani Fatullah",
  "editor": "wokwi",
  "wifi": {
    "ssid": "Wokwi-GUEST",
    "password": ""
  },
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "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": {}
}
```