**LAPORAN PRAKTIKUM**

**IOT MINGGU KE-5**

**Memantau Cuaca Terkini Dari Open Weather Map API**

**Melalui Koneksi Wifi Bebasis Esp32**

****

**Dosen Pengampu :**

**Ir. Subairi, ST., MT., IPM**

**Disusun Oleh:**

**Qaila Salsabila**

**(233140707111089)**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI**

**UNIVERSITAS BRAWIJAYA**

**2025**

**Abstrak**

Tujuan proyek ini adalah untuk membuat sistem pemantauan cuaca berbasis ESP32 yang terhubung ke OpenWeatherMap API melalui koneksi Wi-Fi. Sistem ini akan memungkinkan pengguna memantau informasi seperti suhu, kelembapan, dan kondisi cuaca secara real-time menggunakan perangkat mobile atau layar LCD. Data cuaca diperoleh melalui pengiriman permintaan ke server OpenWeatherMap, dan mikrokontroler ESP32 menangani data tersebut. Untuk menguji logika program tanpa perangkat fisik, simulasi dilakukan menggunakan platform Wokwi. Hasil eksperimen menunjukkan bahwa sistem dapat membaca data cuaca secara akurat dan menampilkan informasi dalam berbagai format, seperti tampilan gulir pada layar LCD. Ini meletakkan dasar untuk pengembangan aplikasi IoT lainnya, seperti dashboard cuaca portabel atau sistem peringatan cuaca otomatis.

*Keywords - ESP32, OpenWeatherMap API, IoT, pemantauan cuaca, koneksi Wi-Fi, sistem real-time, simulasi Wokwi, layar LCD, perangkat pintar, data lingkungan..*

**BAB I**

**PENDAHULUAN**

**1.1 Latar Belakang**

Dengan kemajuan teknologi Internet of Things (IoT), pengolahan data lingkungan dapat dilakukan secara real-time. Pememantauan cuaca adalah salah satu aplikasi penting dalam bidang ini, yang membantu kegiatan sehari-hari, pengelolaan agrikultur, dan mitigasi bencana. Open Weather Map API menjadi sumber data cuaca yang handal dan dapat diakses di seluruh dunia. Dengan menggunakan mikrokontroler ESP32, sistem pemantauan cuaca dapat dikembangkan dengan konektivitas Wi-Fi untuk mengambil data dari API. Sistem ini memungkinkan pengguna memantau parameter cuaca seperti suhu, kelembapan, dan prakiraan cuaca melalui perangkat mobile atau dashboard. Solusi yang efektif dan murah dapat dibuat dengan menggabungkan Open Weather Map API dan ESP32. Selain itu, proyek ini mendiskusikan integrasi Internet of Things (IoT) dan pengelolaan data cloud untuk aplikasi pemantauan berbasis teknologi.

**1.2 Tujuan Eksperimen**

1. Mengembangkan sistem berbasis ESP32 yang memungkinkan Anda mengakses dan menampilkan data cuaca terbaru dari Open Weather Map API dengan menggunakan koneksi internet.
2. Meningkatkan pemahaman tentang cara mengintegrasikan layanan API dengan perangkat IoT untuk keperluan pemantauan lingkungan.
3. Merancang prototipe yang dapat secara efektif memberikan informasi cuaca secara real-time dan dapat diakses melalui perangkat pintar.

**BAB II**

**METODOLOGI**

**2.1 Alat dan Bahan**

* ESP32 DevKit v1
* Resistor (1kΩ dan 10kΩ)
* LED (opsional)
* Kabel Penghubung Virtual
* Platform Simulasi Wokwi
* Komputer/Laptop dengan Internet
* OpenWeatherMap API Key
* Sensor Virtual

**2.3 Implementasi Sistem**

1. Sambungkan ESP32 DevKit v1 dengan kabel virtual di Wokwi.
2. Gunakan koneksi Wi-Fi untuk mengakses data cuaca melalui OpenWeatherMap API.
3. Masukkan API Key OpenWeatherMap dalam kode program untuk autentikasi.
4. Konfigurasikan Serial Monitor untuk menampilkan data cuaca yang diperoleh dari API secara real-time.

**BAB III**

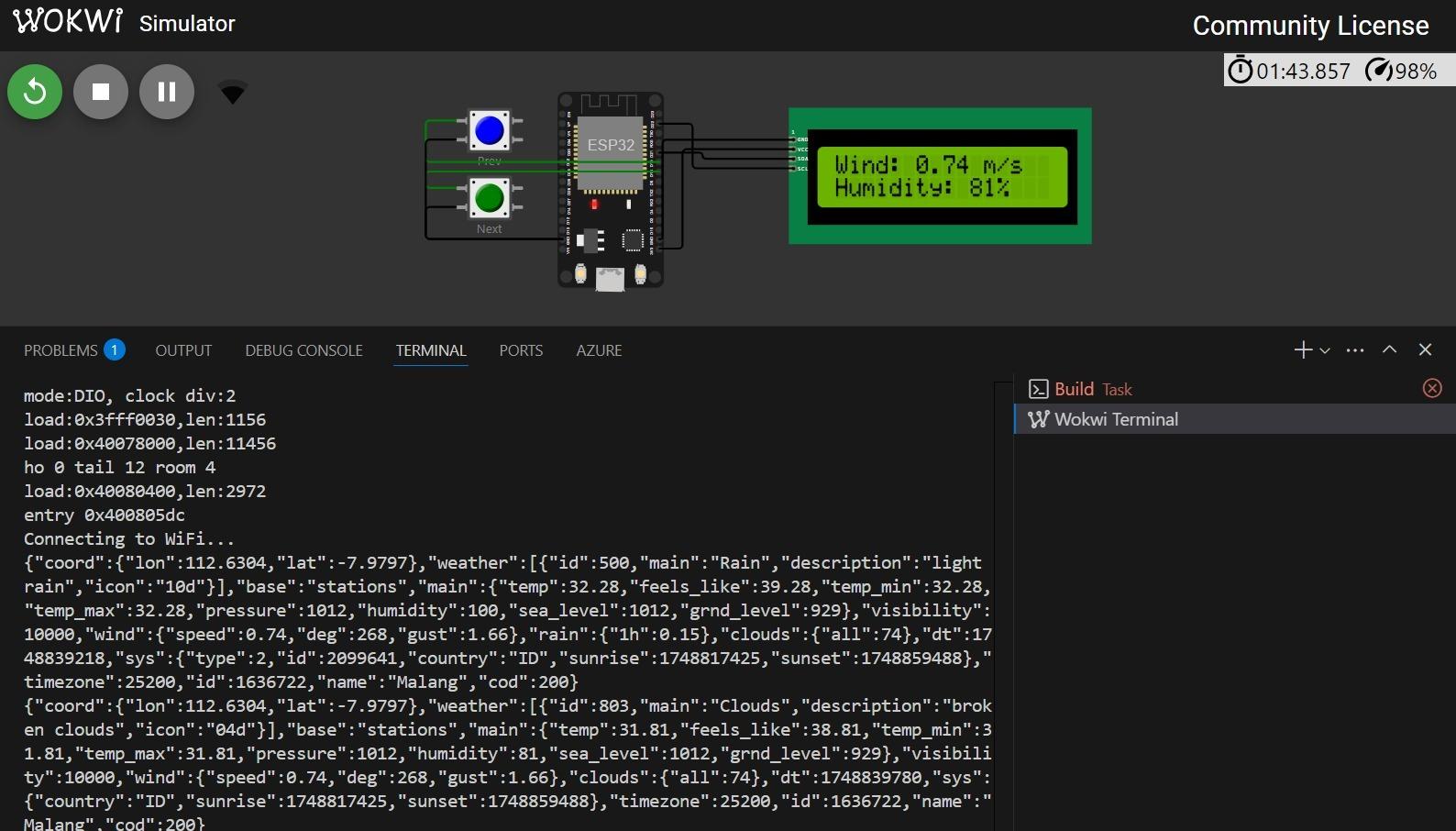
**HASIL DAN PEMBAHASAN**

**3.1 Hasil Eksperimen**

Hasil eksperimen menunjukkan bahwa sistem berbasis ESP32 dapat menggunakan API OpenWeatherMap untuk mengambil dan menampilkan data cuaca secara real-time. Selama koneksi Wi-Fi diatur dengan benar, ESP32 dapat mengakses data dari API dan menampilkan informasi seperti suhu, kelembapan, dan kondisi cuaca melalui Serial Monitor. Eksperimen berjalan lancar tanpa masalah dalam pengambilan data atau pengiriman permintaan ke server API. Dengan menggunakan platform Wokwi untuk simulasi, Anda dapat menguji dan memverifikasi logika program tanpa menggunakan perangkat fisik.

Selain itu, hasil pengujian menunjukkan bahwa program memiliki kemampuan untuk mengumpulkan data dengan jeda waktu yang sesuai. Ketika koneksi Wi-Fi terganggu, sistem dapat mengirimkan pesan kesalahan dan secara otomatis mencoba kembali menghubungkan. Hasil ini menunjukkan bahwa ESP32 dapat berfungsi dengan baik untuk aplikasi IoT sederhana seperti pemantauan cuaca.

**3.2 Dokumentasi eksperimen meliputi screenshoot simulasi :**

****

**Lampiran**

**Kode Program 1 :**

{

"version": 1,

"author": "Cagita Dian A'yunin",

"editor": "wokwi",

"parts": [

{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -50, "left": -100, "attrs": {} },

{

"type": "wokwi-lcd1602",

"id": "lcd",

"top": -32,

"left": 130.4,

"attrs": { "pins": "i2c" }

},

{

"type": "wokwi-pushbutton",

"id": "btnNext",

"top": 35,

"left": -201.6,

"attrs": { "color": "green", "label": "Next" }

},

{

"type": "wokwi-pushbutton",

"id": "btnPrev",

"top": -32.2,

"left": -201.6,

"attrs": { "color": "blue", "label": "Prev" }

}

],

"connections": [

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

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

[ "lcd:SCL", "esp:D22", "black", [ "h-100", "v-40" ] ],

[ "lcd:SDA", "esp:D21", "black", [ "h-90", "v-20" ] ],

[ "lcd:VCC", "esp:3V3", "black", [ "h-110", "v60" ] ],

[ "lcd:GND", "esp:GND.1", "black", [ "h-130", "v40" ] ],

[ "btnNext:1.l", "esp:D18", "green", [ "h-30", "v-20" ] ],

[ "btnNext:2.l", "esp:GND.2", "black", [ "h-30", "v20" ] ],

[ "btnPrev:1.l", "esp:D19", "green", [ "h-30", "v-20" ] ],

[ "btnPrev:2.l", "esp:GND.2", "black", [ "h-30", "v20" ] ]

],

"dependencies": {}

}

**Kode Program 2 :**

#include <Arduino.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

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

const char\* password = "";

String apiKey = "aa183d09b9792e4ef1d50d33a2222699";

String city = "Malang City, East Java";

String units = "metric";

String server = "http://api.openweathermap.org/data/2.5/weather?q=Malang&units=metric&appid=aa183d09b9792e4ef1d50d33a2222699";

LiquidCrystal\_I2C lcd(0x27, 16, 2);

String displayText = "";

int scrollIndex = 0;

unsigned long lastUpdateTime = 0;

unsigned long lastScrollTime = 0;

const long updateInterval = 60000;

const long scrollInterval = 300;

const int buttonNextPin = 18;

const int buttonPrevPin = 19;

int currentPage = 0;

const int totalPages = 4;

unsigned long lastDebounceTime = 0;

const long debounceDelay = 200;

String temp = "";

String desc = "";

String humidity = "";

String wind = "";

void updateWeather();

void showPage(int page);

void scrollDisplay();

void setup() {

Serial.begin(115200);

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Weather Info:");

pinMode(buttonNextPin, INPUT\_PULLUP);

pinMode(buttonPrevPin, INPUT\_PULLUP);

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();

updateWeather();

showPage(currentPage);

}

void loop() {

unsigned long currentMillis = millis();

if (currentMillis - lastUpdateTime >= updateInterval) {

updateWeather();

lastUpdateTime = currentMillis;

showPage(currentPage);

}

if (currentPage == 2) {

if (currentMillis - lastScrollTime >= scrollInterval) {

scrollDisplay();

lastScrollTime = currentMillis;

}

}

if (digitalRead(buttonNextPin) == LOW) {

if (millis() - lastDebounceTime > debounceDelay) {

currentPage++;

if (currentPage >= totalPages) currentPage = 0;

showPage(currentPage);

lastDebounceTime = millis();

}

}

if (digitalRead(buttonPrevPin) == LOW) {

if (millis() - lastDebounceTime > debounceDelay) {

if (currentPage == 0) currentPage = totalPages - 1;

else currentPage--;

showPage(currentPage);

lastDebounceTime = millis();

}

}

}

void updateWeather() {

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

HTTPClient http;

http.begin(server);

int httpCode = http.GET();

if (httpCode > 0) {

String payload = http.getString();

Serial.println(payload);

StaticJsonDocument<1024> doc;

deserializeJson(doc, payload);

temp = String(doc["main"]["temp"].as<float>());

desc = doc["weather"][0]["description"].as<String>();

humidity = String(doc["main"]["humidity"].as<int>());

wind = String(doc["wind"]["speed"].as<float>());

displayText = "| Temp: " + temp + " C | " + desc;

scrollIndex = 0;

} else {

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

}

http.end();

}

}

void showPage(int page) {

lcd.clear();

switch (page) {

case 0:

lcd.setCursor(0, 0);

lcd.print("Location:");

lcd.setCursor(0, 1);

lcd.print(city);

break;

case 1:

lcd.setCursor(0, 0);

lcd.print("Temperature:");

lcd.setCursor(0, 1);

lcd.print(temp + " C");

break;

case 2:

lcd.setCursor(0, 0);

lcd.print("Weather:");

lcd.setCursor(0, 1);

if (desc.length() > 16) {

lcd.print(desc.substring(0, 16));

} else {

lcd.print(desc);

}

break;

case 3:

lcd.setCursor(0, 0);

lcd.print("Wind: " + wind + " m/s");

lcd.setCursor(0, 1);

lcd.print("Humidity: " + humidity + "%");

break;

}

}

void scrollDisplay() {

if (displayText.length() > 0) {

lcd.setCursor(0, 1);

if (scrollIndex + 16 <= displayText.length()) {

lcd.print(displayText.substring(scrollIndex, scrollIndex + 16));

} else {

String part1 = displayText.substring(scrollIndex);

String part2 = displayText.substring(0, 16 - part1.length());

lcd.print(part1 + part2);

}

scrollIndex++;

if (scrollIndex >= displayText.length()) {

scrollIndex = 0;

}

}

}