MONITORING CUACA TERKINI DARI API OPENWEATHERMAP MELALUI KONEKSI WIFI BERBASIS ESP 32

Feby Kurnia Putri

Teknologi Informasi, Fakultas Vokasi, Universitas Brawijaya Jl. Veteran No.10-11, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145

febystudentub@gmail.com

Abstrack -- This practicum focuses on the development of a weather monitoring system based on the Internet of Things (IoT), utilizing the ESP32 microcontroller as the main controller. The system is designed to access and display real-time weather data retrieved from the OpenWeatherMap API via a Wi-Fi connection. The displayed information includes temperature, weather description, humidity, and wind speed. All data is shown on a 16x2 LCD screen using an I2C interface. User interaction is facilitated through navigation buttons that allow easy switching between information pages. The monitoring area in this practicum is centered on Malang, East Java, with each page displaying specific data: location, temperature, scrollable weather description, and a combination of wind speed and humidity. To overcome the display limitations, a scrolling feature is implemented for the description page. Weather data is automatically updated every 60 seconds to ensure accurate and up-to-date information. Additionally, button debounce is applied to prevent unintended multiple inputs due to mechanical bouncing. Overall, this system offers a simple and efficient solution for real-time weather monitoring with a responsive and user-friendly experience.

Keyword : ***Internet of Things, ESP32, OpenWeatherMap, I2C LCD, button navigation, real-time.***

Abstrak **--** Dalam praktikum ini dikembangkan sebuah sistem pemantauan cuaca berbasis *Internet of Things (IoT)* dengan memanfaatkan mikrokontroler *ESP32* sebagai pusat kendali. Sistem dirancang agar mampu mengakses dan menampilkan data cuaca terkini dari layanan *OpenWeatherMap* melalui jaringan *Wi-Fi.* Informasi yang ditampilkan mencakup suhu, kondisi cuaca, kelembaban, serta kecepatan angin. Seluruh data disajikan melalui layar *LCD* 16x2 yang terhubung menggunakan antarmuka I2C. Interaksi pengguna difasilitasi melalui tombol navigasi, yang memungkinkan peralihan antar halaman informasi dengan mudah. Fokus wilayah pemantauan diarahkan pada daerah Malang, Jawa Timur, dengan masing-masing halaman menyajikan data lokasi, suhu, deskripsi cuaca yang dapat digulir, serta kombinasi data kecepatan angin dan kelembaban. Untuk menyesuaikan keterbatasan tampilan layar, fitur *scroll* ditambahkan pada halaman deskripsi. Pembaruan data dilakukan otomatis setiap 60 detik, menjamin informasi yang tersaji tetap relevan. Selain itu, pengaturan *debounce* diterapkan pada tombol navigasi guna menghindari respon ganda akibat getaran mekanik. Secara keseluruhan, sistem ini menawarkan solusi sederhana dan efisien untuk memantau kondisi cuaca secara *real-time* dengan pengalaman pengguna yang responsif dan informatif.

Kata Kunci : ***Internet of Things IoT)*, ESP32, *OpenWeatherMap*, *LCD* I2C, navigasi tombol, *real-time.***

I. PENDAHULUAN

Perkembangan teknologi *Internet of Things (IoT)* telah memberikan banyak kemudahan dalam berbagai aspek kehidupan, Salah satu aplikasi *IoT* yang populer adalah sistem monitoring cuaca, yang memungkinkan pengguna untuk memantau kondisi cuaca secara *real-time* menggunakan perangkat yang terhubung ke internet. Sistem ini memiliki berbagai manfaat, terutama dalam konteks prediksi cuaca, perencanaan kegiatan luar ruangan, dan manajemen sumber daya alam. Dalam praktikum ini, dibangun sebuah sistem monitoring cuaca berbasis *IoT* menggunakan *platform ESP32. ESP32* adalah salah satu mikrokontroler yang populer karena memiliki kemampuan *Wi-Fi* dan *Bluetooth,* sehingga sangat cocok untuk aplikasi berbasis *IoT.* Sistem ini mengakses data cuaca dari API *OpenWeatherMap,* yang menyediakan informasi cuaca secara global melalui jaringan *internet. OpenWeatherMap* adalah layanan cuaca online yang menyediakan data cuaca terkini, perkiraan cuaca, dan informasi atmosfer lainnya. *API* yang disediakan oleh *OpenWeatherMap* memungkinkan pengguna untuk mengakses data cuaca secara bebas dengan batasan tertentu, dan memberikan informasi seperti suhu, kelembaban, deskripsi cuaca, kecepatan angin, dan banyak lagi, yang dapat digunakan untuk aplikasi berbasis *IoT.* Dengan menggunakan *ESP32,* data yang diperoleh dari *API*

OpenWeatherMap akan ditampilkan secara langsung pada layar LCD 16x2 untuk memudahkan pengguna dalam memantau kondisi cuaca. Sistem ini tidak hanya menampilkan informasi cuaca secara statis, tetapi juga menyediakan fitur navigasi antar halaman dan *scrolling* teks, sehingga pengguna dapat memperoleh informasi cuaca secara lebih interaktif. Selain itu, sistem ini mampu memperbarui data cuaca setiap 60 detik sekali untuk memastikan informasi yang ditampilkan selalu terbaru. Dengan mengintegrasikan sensor dan *API* cuaca dalam satu sistem berbasis *IoT*, praktikum ini menunjukkan potensi besar teknologi *IoT* dalam mempermudah akses informasi secara *real-time.*

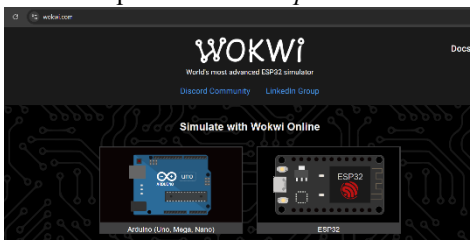
II. METODOLOGI

A. ALAT & BAHAN

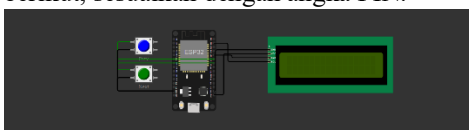
Mikrokontroler *(ESP32), LCD* 16x2 modul I2C, *Push Button, API OpenWeatherMap, Visual Studio Code* (C++), *PlatformIO IDE.*

B. LANGKAH IMPLEMENTASI

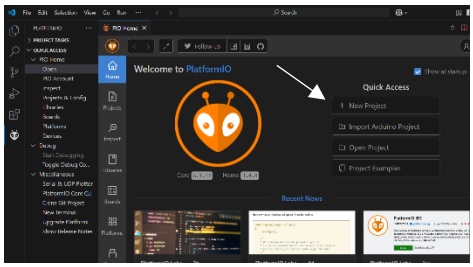
1. Buka https://wokwi.com/ pilih ESP32, kemudian pilih *Starter Templates.*



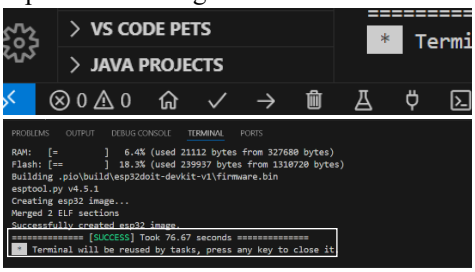
2. Atur rangkain diagram seperti gambar berikut, sesuaikan dengan angka PIN.



3. Buka VScode, kemudian *install PlatformIO IDE.* Buat proyek baru seperti gambar berikut:



4. *Build Wokwi* dengan cara klik centang pada bagian bawah dari *Visual Studio Code,* seperti gambar berikut.



5. Buat *file* baru pada *Visual Studio Code*, kemudian buka *wokwi.toml* pada *google*, kemudian *copy* ke *file* baru yang diberi nama *wokwi.toml* di *VScode.*

wokwi.toml

A basic `wokwi.toml` file looks like this:

```
[wokwi]
version = 1
firmware = 'path-to-your-firmware.hex'
elf = 'path-to-your-firmware.elf'
```

6. Setelah itu ganti menjadi *firmware bin*, dan *esp32 firmware elf* pada *Visual Studio Code* masing-masing.

7. Buat *file* baru bernama *diagram.json* pada *Visual Studio Code.* Kemudian pergi ke *wokwi* untuk meng *copy code diagram.json.*

```
{
  "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": {}
}
```

8. *Paste* ke *Visual Studio Code,* pada *file diagram.json. Jika diagram* tidak berfungsi, maka lakukan edit pada *diagram.json*, kemudian *paste code* yang sudah di *copy* pada *wokwi.*

9. Kemudian *wokwi* akan menampilkan hasil dari *diagram.json* pada simulasi *Temperature dan Humidity.*

10. Buat code C++ pada main

```
#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;
  }
}
```

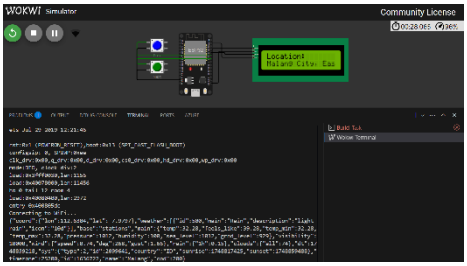
```
void scrollDisplay() {
  if (displayText.length() > 0) {
    lcd.setCursor(0, 1);
    if (scrollIndex + 16 <= displayText.length()) {

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;
    }
  }
}
```

III. HASIL DAN PEMBAHASAN

1. *ESP32* berhasil terhubung ke jaringan Wi-Fi bernama Wokwi-GUEST tanpa memerlukan password, dan status koneksi yang berhasil ditampilkan pada layar *LCD* dengan pesan *"Connected!"*. Setelah koneksi terjalin, sistem mampu mengambil data cuaca secara *real-time* dari *server OpenWeatherMap.*
2. Informasi yang diperoleh mencakup suhu udara dalam satuan *Celcius,* deskripsi kondisi cuaca seperti *"clear sky"* atau *"overcast clouds",* kelembaban udara dalam persen, serta kecepatan angin dalam meter per detik. Data tersebut diperbarui secara otomatis setiap 60 detik untuk memastikan informasi yang ditampilkan tetap akurat dan terkini. Seluruh informasi cuaca ini ditampilkan secara bergantian dalam empat halaman berbeda pada layar *LCD*. Halaman pertama (halaman 0) menampilkan nama kota, yaitu "Malang", halaman kedua menampilkan suhu udara terkini, halaman ketiga menyajikan deskripsi kondisi cuaca yang dilengkapi dengan fitur *scrolling* apabila teks melebihi 16 karakter, dan halaman keempat memperlihatkan data kecepatan angin serta kelembaban udara.



3. Navigasi sistem dilakukan menggunakan dua buah tombol dengan mekanisme debouncing selama 200 milidetik, sehingga perpindahan antar halaman terasa lebih responsif dan stabil. Pengujian tombol menunjukkan hasil yang sesuai dengan fungsinya, yaitu tombol *"Next"* digunakan untuk berpindah ke halaman berikutnya, sementara tombol *"Prev"* digunakan untuk kembali ke halaman sebelumnya. Seluruh proses eksperimen dan pengujian sistem dilakukan menggunakan platform simulasi *Wokwi* dan *Visual Studio Code,* dengan konfigurasi *ESP32,* layar *LCD* I2C, serta dua push button. Hasil simulasi menunjukkan bahwa sistem berfungsi sesuai dengan rancangan dan mampu menjalankan seluruh fitur yang telah ditetapkan secara optimal.

IV. KESIMPULAN

Setelah melakukan perancangan dan pengujian terhadap alat secara keseluruhan. Maka dapat diambil kesimpulan :

1. Perangkat yang digunakan oleh penulis dapat bekerja dengan baik sesuai dengan yang diharapkan.
2. Mikrokontroler *ESP32* yang digunakan sebagai pengendali utama dapat bekerja dalam menjalankan perintah yang diberikan

V. DAFTAR PUSTAKA

Espressif Systems. (2023). *ESP32 technical reference manual*. https://www.espressif.com/en/products/socs/esp32/resources

I2C LCD Tutorial. (n.d.). *How to use an I2C LCD with ESP32 on Arduino IDE*. https://randomnerdtutorials.com/esp32-i2c-lcd-arduino-ide/