**Uji Coba Integrasi API Cuaca ke dalam ESP32 dengan Wokwi dan Arduino di Visual Studio Code**

*Muhammad Fau Zan Sabani1*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*mfauzan18@student.ub.ac.id*](mailto:mfauzan18@student.ub.ac.id)

# Abstrak:

Abstrak ini membahas **uji coba integrasi API cuaca** dengan mikrokontroler **ESP32** untuk mengambil data cuaca secara real-time dari internet. Tujuan utama dari praktikum ini adalah untuk membuktikan kemampuan ESP32 dalam melakukan permintaan HTTP ke layanan web eksternal, mem-parsing data JSON yang diterima, dan menampilkannya sebagai informasi yang berguna, di mana keseluruhan prosesnya disimulasikan menggunakan **Wokwi**.

Eksperimen dimulai dengan pengembangan kode dalam **Visual Studio Code** menggunakan kerangka kerja Arduino. Kode dirancang untuk menghubungkan ESP32 ke jaringan Wi-Fi, kemudian membuat permintaan HTTP GET ke sebuah endpoint API cuaca. Data respons dalam format JSON kemudian di-parsing untuk mengekstrak informasi spesifik seperti suhu, kelembapan, dan deskripsi cuaca, yang hasilnya ditampilkan melalui Serial Monitor.

Hasil dari eksperimen ini menunjukkan bahwa **integrasi antara simulasi ESP32 di Wokwi dengan API cuaca eksternal berhasil dilakukan**. ESP32 mampu terhubung ke internet, mengambil, dan mem-parsing data JSON dengan benar. Uji coba ini membuktikan bahwa Wokwi adalah platform yang andal untuk mengembangkan dan memvalidasi proyek IoT yang terkoneksi dengan internet, memungkinkan pengujian interaksi dengan layanan web sebelum implementasi pada perangkat keras fisik.

# Kata Kunci *Cuaca, IoT, Arduino, ESP32, Wokwi, Laporan Praktikum*

# Abstract:

This This abstract discusses the trial integration of the Arduino system with the Blynk IoT platform for monitoring and controlling devices via a web browser. The main objective of this practicum is to implement a system capable of sending sensor data (temperature and humidity) from the Wokwi simulation to the Blynk dashboard, as well as receiving commands to manually control light switches from the dashboard.

The experiment started with designing the system in Visual Studio Code using the Arduino framework. Program code was created to read data from the DHT22 sensors (temperature and humidity) inside the Wokwi simulation. In addition to sending the data to the Blynk server, the system was also designed to listen to commands from a switch widget on the Blynk dashboard that serves to manually turn the lights on and off.

The results of this experiment show that the two-way integration between the Wokwi simulation and the Blynk platform is successful. Temperature and humidity data can be accurately displayed on the web dashboard, and commands from the switch on the Blynk dashboard successfully control the lights in the Wokwi simulation responsively. This trial proves the effectiveness of Wokwi for validating IoT systems that involve simultaneous monitoring of sensors and actuators before physical implementation.

Keywords *Weather, IoT, ESP32, DHT, LDR, Wokwi, Practical Report*

# 1. Pendahuluan

* 1. **Latar Belakang**

Perangkat Internet of Things (IoT) pada dasarnya dirancang untuk berinteraksi dengan lingkungan sekitarnya melalui sensor dan aktuator. Namun, banyak sistem IoT sederhana yang hanya beroperasi dalam lingkup data lokalnya sendiri; artinya, mereka hanya mengetahui apa yang bisa diukur oleh sensor yang terhubung langsung. Keterbatasan ini membuat perangkat menjadi kurang 'cerdas' karena tidak memiliki kesadaran akan konteks dunia luar yang lebih luas, seperti kondisi cuaca terkini, berita, atau data publik lainnya.

Untuk mengatasi keterbatasan tersebut, integrasi dengan Application Programming Interface (API) menjadi solusi yang sangat kuat. API berfungsi sebagai jembatan yang memungkinkan perangkat IoT untuk 'berkomunikasi' dengan layanan web eksternal dan mengambil data dinamis dari internet. Dengan mengakses API, sebuah mikrokontroler sederhana sekalipun dapat memperoleh informasi yang kaya dan relevan—seperti data prakiraan cuaca—sehingga fungsinya menjadi jauh lebih canggih dan aplikasinya lebih bermanfaat.

Oleh karena itu, praktikum yang menguji coba integrasi dengan API cuaca menjadi langkah fundamental untuk memahami cara kerja konektivitas IoT modern. Dalam proyek ini, mikrokontroler ESP32 dipilih karena kemampuannya untuk terhubung ke jaringan Wi-Fi. Platform simulasi Wokwi digunakan untuk memvalidasi seluruh proses, mulai dari membuat permintaan HTTP ke API hingga mem-parsing data JSON yang diterima, yang semuanya dilakukan dalam lingkungan virtual di Visual Studio Code sebelum implementasi fisik.

* 1. **Tujuan Eksperimen**

# Tujuan dari eksperimen integrasi sistem Arduino dengan platform Blynk ini adalah sebagai berikut:

# Merancang dan mengimplementasikan sistem IoT dua arah yang mampu mengirimkan data sensor (suhu dan kelembapan) ke dasbor Blynk dan menerima perintah untuk mengendalikan saklar lampu secara real-time.

# Menguji integrasi antara simulasi Wokwi dan platform cloud Blynk untuk memvalidasi fungsionalitas pengiriman data (uplink) dan penerimaan perintah (downlink) tanpa memerlukan perangkat keras fisik.

# Memahami arsitektur sistem kontrol IoT secara menyeluruh, mulai dari pembacaan data sensor di perangkat, komunikasi dengan layanan cloud, hingga interaksi pengguna melalui antarmuka dasbor web.

# Mengembangkan kemampuan teknis dalam pemrograman mikrokontroler untuk berinteraksi dengan layanan IoT serta mengonfigurasi dasbor kontrol yang fungsional dan interaktif.

# Menyediakan dasar untuk pengembangan aplikasi Smart Home atau otomasi sederhana, yang membuktikan kelayakan konsep sebelum diimplementasikan dengan biaya dan perangkat keras yang sebenarnya.

# 2. Metodologi

* 1. **Alat dan Bahan**

Dalam perancangannya, untuk membuat simulasi menampilkan indikator suhu, kelembapan dan mengatur saklar pada IoT menggunakan dashboard Blynk dan Arduino dibutuhkan beberapa alat dan bahan, diantaranya:

* 1 buah mikrokontroller ESP32,
* 1 buah LCD 16x2 I2C,
* Platform Wokwi, yang digunakan untuk simulasi rangkaian dan pemrograman secara virtual.
* Software Arduino IDE/Extension PlatformIo (jika menggunakan Visual Studio Code).
  1. **Langkah Implementasi**

Berikut langkah-langkah implementasi simulasi lampu lalu lintas dengan menggunakan Mikrokontroler ESP32 di Wokwi/Arduino:

* Install Arduino IDE atau jika menggunakan VsCode install Extension PlatfromIO di bagian ekstensinya. Buka juga website wokwi untuk tampilan simulasi nya secara virtual (jangan lupa untuk install ekstensi Wokwi juga di VsCode).



* Buat proyek baru di Wokwi dengan memilih ESP32 sebagai mikrokontroler. Tambahkan juga komponen yang dibutuhkan seperti LCD 16x2 I2C. Smbungkan tiap pin LCD ke tiap pin yang sesuai di mikrokontroller ESP32 menggunakan kabel jumper virtual. Seperti dibawah:

A computer screen shot of a computer

Description automatically generated

* Untuk pemogramannya menggunakan bahasa Arduino (C/C++) di Arduino IDE/PlatformIO (bisa juga langsung dimasukkan ke file ‘main.c’ di website Wokwi). Kodenya kita atur agar output dari pemograman ini nantinya dapat menampilkan hasil indikator suhu, tingkat kelembapan dan intensitas cahaya pada OLED Display. Seperti berikut kode pemogramannya:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <WiFi.h> // Ganti dengan ESP8266WiFi.h jika menggunakan ESP8266

#include <HTTPClient.h>

const char\* ssid = "Wokwi-GUEST"; // Ganti dengan SSID Wi-Fi kamu

const char\* password = ""; // Ganti dengan password Wi-Fi kamu

String apiKey = "20ca0ff523294dcdeb424dfc5802e21b"; // API Key dari OpenWeatherMap

String city = "Malang"; // Kota yang ingin ditampilkan

String units = "metric"; // Untuk Celsius gunakan "metric", untuk Fahrenheit "imperial"

String server = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=" + units + "&appid=" + apiKey;

//String server= "https://api.openweathermap.org/data/2.5/weather?q=Malang&appid=20ca0ff523294dcdeb424dfc5802e21b";

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Inisialisasi LCD dengan alamat I2C 0x27

void setup() {

**Serial**.begin(115200);

// Inisialisasi LCD

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Weather Info:");

// Inisialisasi Wi-Fi

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();

}

void loop() {

if ((WiFi.status() == WL\_CONNECTED)) { // Check WiFi connection status

HTTPClient http;

http.begin(server); // Specify the URL

int httpCode = http.GET(); // Make the request

if (httpCode > 0) { // Check for the returning code

String payload = http.getString();

**Serial**.println(payload); // Print the response payload

// Parse data (extract temperature)

int tempIndex = payload.indexOf("temp");

String temp = payload.substring(tempIndex + 6, payload.indexOf(",", tempIndex));

// Display temperature on LCD

lcd.setCursor(0, 0);

lcd.print("Temp:");

lcd.setCursor(6, 0);

lcd.print(temp);

lcd.print(" C");

// Extract weather description

int descIndex = payload.indexOf("description");

String desc = payload.substring(descIndex + 14, payload.indexOf("\"", descIndex + 14));

// Display description on LCD

lcd.setCursor(0, 1);

lcd.print(desc);

} else {

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

}

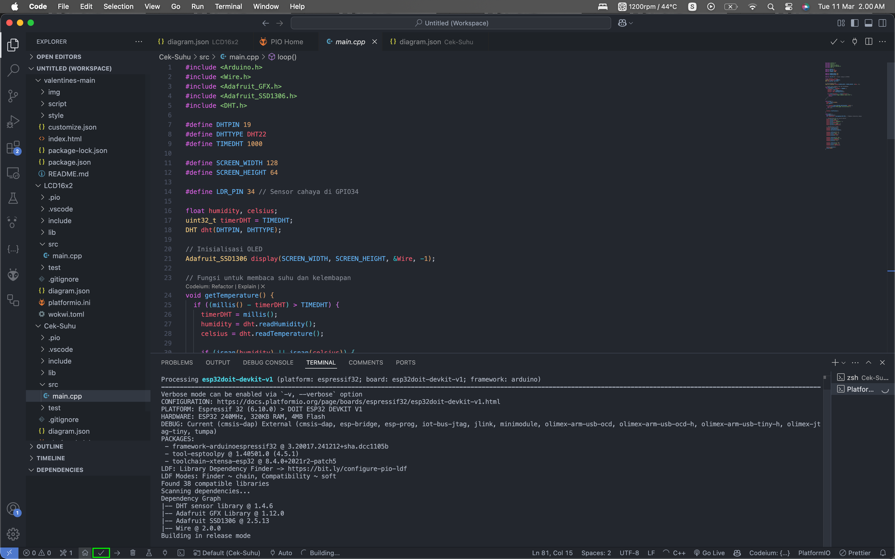
http.end(); // Free the resources

}

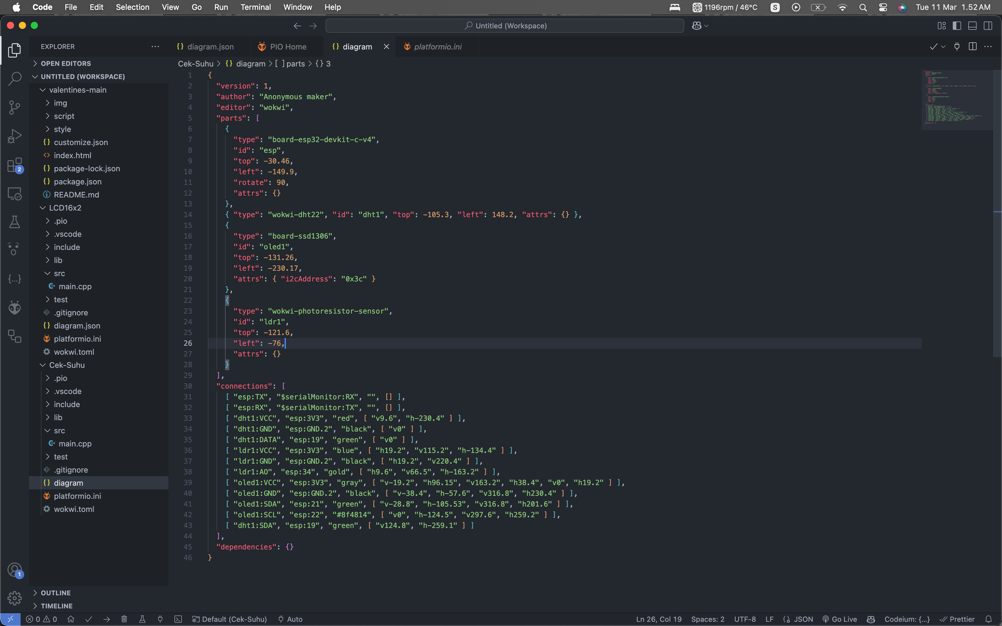
delay(60000); // Update every minute

}

* Karena di dalam eksperimen kali ini menggunakan beberapa part yang memerlukan library untuk menjalankannya seperti library untuk LCD maka dari itu perlu penginstalannya dari library yang sudah disediakan di Library Manager. Berikut library yang dibutukan pada eksperimen kali ini:
* Langkah selanjutnya jalankan semua kodenya agar bisa menampilkan outputnya.
* Jika di platform Wokwi file ‘main.c’ tidak bisa ter-compile/dijalankan maka kita bisa menguji apakah sukses atau tidak program kode tersebut di PlatformIO dengan menggunakan tombol centang dibawah.



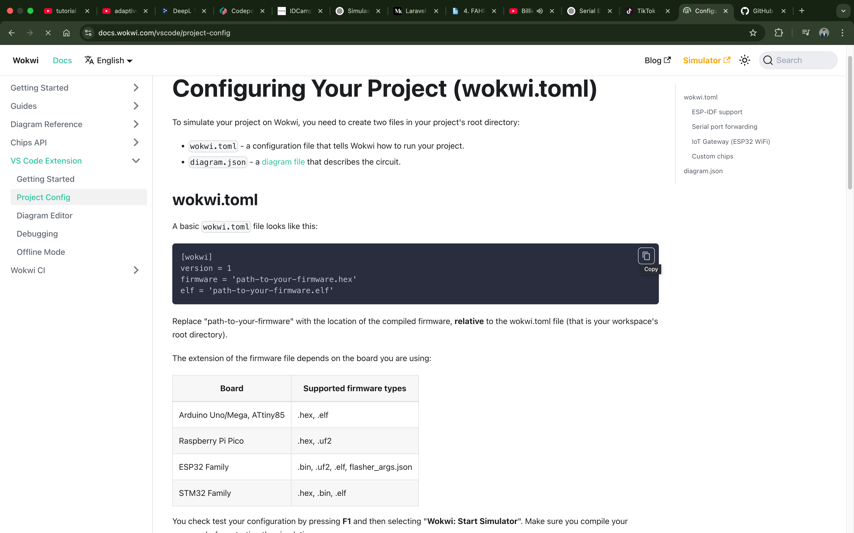
* Jika kode di file ‘main.cpp’ sukses di *Build* maka langkah selanjutnya untuk dapat melakukan virtualisasi tampilannya dengan membuat file bernama ‘diagram’ yang nanti diisi oleh kode dari file ‘diagram.json’ di projek Wokwi sebelumnya. Lalu kalau sudah, ubah nama file ‘diagram’ sbelumnya menjadi ‘diagram.json’ seperti di platform Wokwi sebelumnya.

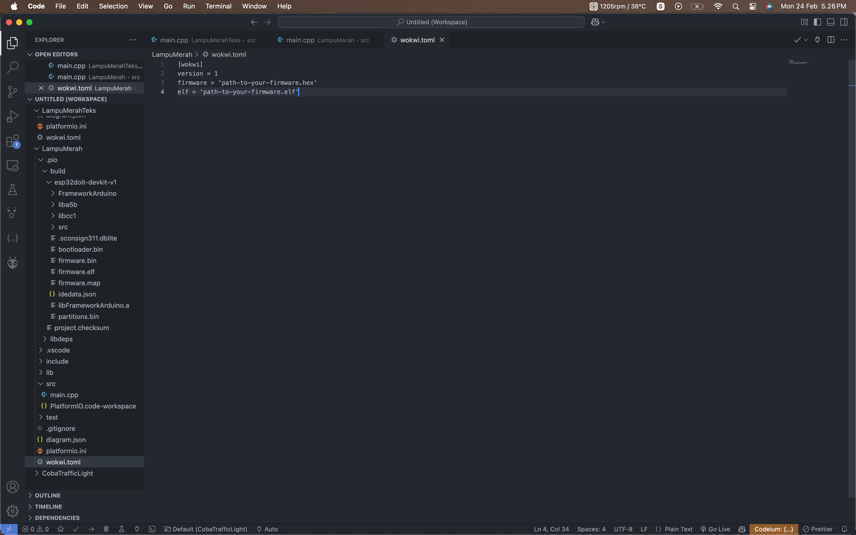


A computer screen shot of a computer

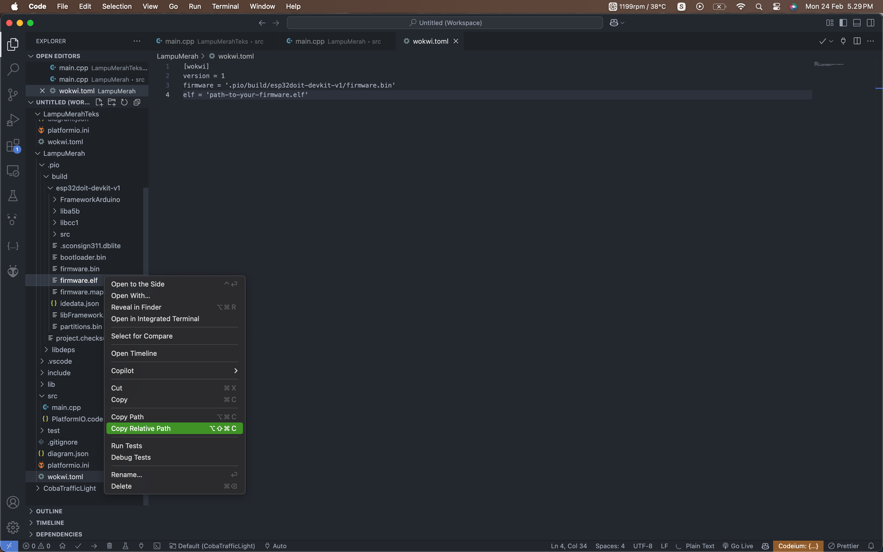
Description automatically generated

* Untuk dapat menjalankan simulasi dari file ‘diagram.json’ tersebut maka kita harus menambahkan file yang kita namakan ‘wokwi.toml’. Langkah berikutnya cari di web browser dengan keyword ‘Wokwi.toml’ dan salin kode konfigurasinya yang berasal dari halaman web dokumentasi wokwi tersebut ke file ‘wokwi.toml’ yang sudah kita buat.





* Langkah selanjutnya untuk dapat mengkonfigurasi file ‘wokwi.toml’ tersebut agar dapat membantu menjalankan file diagramnya maka kita perlu untuk mengganti path ‘firmware’ dan ‘elf’ path file yang berada di dalam rute folder ‘pio/build/esp32doit-devkit-v1’. Salin tiap file dengan format file yang bertuliskan ‘.hex’ dan ‘.elf’ dengan menggunakan klik kanan dan klik ‘Copy relative path’ lalu tempel ke path yang harus diisi di file ‘wokwi.toml’.



* Kemudian pergi[*https://openweathermap.org/api*](https://openweathermap.org/api)dan lakukan sesi login untuk bisa mendapatkan API KEY agar bisa mengintegrasikan informasi cuaca di kota tertentu ke dalam ESP32.
* Tambahkan API Key yang sudah dibuat tadi ke dalam kode ESP32 di main

A screenshot of a computer

Description automatically generated

* Jika sudah mendapatkan token, ganti token yang ada di kode main untuk bisa mengintegrasikan hasil output dari wokwi ke dalam dashboard blynk.
* Langkah terakhir lakukan testing atau uji coba dengan menjalankan file diagram.json dan cek apakah data dari hasil output terintegrasikan ke dalam platform ESP32.

A computer screen shot of a computer

Description automatically generated

# 3. Hasil dan Pembahasan

1. **Hasil Eksperimen**

Dari simulasi yang sudah dibuat, di dapatkan hasil yang sudah sesuai yang kita inginkan. Berikut hasil simulasinya:

* Tampilan Wokwi:

A computer screen shot of a computer

Description automatically generated

* Hasil uji coba:

A computer screen shot of a computer

Description automatically generated

# 4. Lampiran

* Kode Program diagram.json:

{

"version": 1,

"author": "VIOLA ROSA EMAWARDANI 2020",

"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" }

}

],

"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" ] ]

],

"dependencies": {}

}