

## **Tugas 4**

### **Laporan Praktikum Akses API Melalui Simulasi WOKWI**



Nama : Mochamad Iftichor Al Ashief  
Kelas : T4C  
NIM : 233140700111082

**Fakultas Vokasi**  
**Universitas Brawijaya**  
Email : [mhmdashief@gmail.com](mailto:mhmdashief@gmail.com)

## Abstrak

Simulasi akses API menggunakan Wokwi merupakan metode yang memungkinkan pengujian dan pengembangan perangkat lunak berbasis Internet of Things (IoT) secara efisien. Wokwi, sebagai simulator berbasis web, menyediakan lingkungan yang interaktif untuk memvisualisasikan dan menguji konektivitas perangkat keras virtual dengan berbagai layanan API. Dalam penelitian ini, simulasi difokuskan pada proses pengiriman dan penerimaan data antara mikrokontroler virtual dan server melalui protokol HTTP. Dengan menggunakan platform Wokwi, pengembang dapat mengimplementasikan kode menggunakan bahasa pemrograman seperti Arduino C atau Python, mensimulasikan berbagai skenario komunikasi API tanpa memerlukan perangkat fisik.

Proses simulasi melibatkan pembuatan endpoint API menggunakan layanan pihak ketiga atau server lokal, serta implementasi kode untuk mengirim permintaan HTTP GET atau POST. Data yang diterima dari API kemudian dapat ditampilkan pada antarmuka visual yang tersedia di Wokwi. Selain itu, simulasi ini memungkinkan pengujian keandalan dan efisiensi koneksi API, serta mendeteksi potensi masalah sebelum penerapan pada perangkat fisik.

Hasil simulasi menunjukkan bahwa Wokwi merupakan alat yang efektif untuk mempercepat siklus pengembangan aplikasi IoT. Dengan mengurangi ketergantungan pada perangkat keras fisik, pengembang dapat mempercepat proses pengujian dan iterasi kode. Studi ini diharapkan memberikan kontribusi dalam meningkatkan efisiensi pengembangan aplikasi IoT berbasis API, khususnya bagi pengembang yang memiliki keterbatasan sumber daya perangkat keras.

## Abstract

API access simulation using Wokwi is a method that enables efficient testing and development of Internet of Things (IoT)-based software. Wokwi, as a web-based simulator, provides an interactive environment to visualize and test virtual hardware connectivity with various API services. In this study, the simulation focuses on the process of sending and receiving data between a virtual microcontroller and a server via the HTTP protocol. By using the Wokwi platform, developers can implement code using programming languages such as Arduino C or Python, simulating various API communication scenarios without the need for physical devices.

The simulation process involves creating API endpoints using third-party services or local servers, as well as implementing code to send HTTP GET or POST requests. Data received from the API can then be displayed on the visual interface available in Wokwi. In addition, this simulation allows testing the reliability and efficiency of API connections, as well as detecting potential problems before implementation on physical devices.

The simulation results show that Wokwi is an effective tool for accelerating the IoT application development cycle. By reducing dependence on physical hardware, developers can accelerate the testing and code iteration process. This study is expected to contribute to improving the efficiency of API-based IoT application development, especially for developers who have limited hardware resources.

**Kata Kunci:** Simulasi API, Wokwi, IoT, Mikrokontroler Virtual, Pengembangan Perangkat Lunak

# **BAB 1**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Dalam era digital yang semakin berkembang, Internet of Things (IoT) menjadi salah satu teknologi yang berperan penting dalam berbagai sektor, mulai dari industri manufaktur hingga smart home. Salah satu komponen utama dalam pengembangan IoT adalah komunikasi antara perangkat keras dengan server melalui Application Programming Interface (API). Proses ini memungkinkan perangkat untuk mengirim dan menerima data secara real-time, mendukung berbagai aplikasi seperti monitoring sensor, otomatisasi rumah, dan pelacakan logistik.

Namun, pengembangan sistem berbasis IoT sering menghadapi tantangan seperti ketersediaan perangkat keras, biaya produksi, dan waktu yang dibutuhkan untuk pengujian. Untuk mengatasi masalah ini, simulasi menggunakan platform seperti Wokwi menjadi solusi yang efisien. Wokwi menyediakan lingkungan virtual yang memungkinkan pengembang untuk membuat, menguji, dan memvalidasi kode yang berinteraksi dengan API tanpa menggunakan perangkat fisik.

Melalui simulasi ini, pengembang dapat memvisualisasikan respons API, mengidentifikasi potensi masalah, dan memperbaiki kode sebelum implementasi ke perangkat nyata. Selain itu, proses ini juga mendukung kolaborasi tim secara lebih efektif karena kode dapat diuji di berbagai skenario tanpa batasan perangkat keras.

Penelitian ini bertujuan untuk mengeksplorasi penerapan Wokwi dalam mensimulasikan akses API, menganalisis efektivitasnya, serta memberikan panduan praktis bagi pengembang dalam mengintegrasikan API ke dalam sistem IoT. Dengan demikian, diharapkan hasil penelitian ini dapat menjadi referensi yang bermanfaat dalam pengembangan teknologi berbasis IoT.

## **Bab 2**

### **Tujuan**

#### **2.1 Tujuan Umum**

Tujuan umum dari penelitian ini adalah untuk mensimulasikan akses API menggunakan platform Wokwi sebagai alat bantu dalam pengembangan dan pengujian aplikasi berbasis Internet of Things (IoT).

#### **2.2 Tujuan Khusus**

1. Mengidentifikasi manfaat dan efisiensi penggunaan Wokwi dalam mensimulasikan akses API.
2. Menganalisis kinerja sistem saat berkomunikasi dengan API menggunakan protokol HTTP melalui platform Wokwi.
3. Memberikan panduan praktis bagi pengembang dalam mengimplementasikan dan menguji kode API menggunakan Wokwi.
4. Mengevaluasi potensi permasalahan yang dapat muncul selama proses simulasi dan mencari solusinya.
5. Membandingkan hasil simulasi dengan implementasi langsung pada perangkat fisik untuk mengukur tingkat akurasi dan keandalan simulasi.

## **BAB 3**

### **HASIL DAN PEMBAHASAN**

#### **3.1 Hasil Simulasi**

Proses simulasi dilakukan dengan menggunakan platform Wokwi untuk mengakses API melalui protokol HTTP. Simulasi ini mencakup pengiriman permintaan HTTP GET untuk mengambil data dari server, serta HTTP POST untuk mengirim data ke server. Data yang diperoleh ditampilkan secara real-time pada antarmuka virtual yang disediakan oleh Wokwi.

Selama simulasi, berbagai skenario diuji, termasuk koneksi dengan API publik, API lokal, serta skenario penanganan kesalahan seperti time-out dan respon error. Hasil menunjukkan bahwa Wokwi mampu mereplikasi proses komunikasi API dengan akurasi yang tinggi, memungkinkan pengembang untuk memantau dan memvalidasi data secara langsung.

#### **3.2 Pembahasan**

Berdasarkan hasil simulasi, beberapa manfaat utama dari penggunaan Wokwi dalam pengembangan aplikasi IoT teridentifikasi, antara lain:

1. Efisiensi Pengembangan: Dengan menghilangkan ketergantungan pada perangkat fisik, waktu pengembangan dapat dipersingkat secara signifikan.
2. Kemudahan Pengujian: Pengembang dapat dengan mudah mensimulasikan berbagai skenario koneksi API untuk mengidentifikasi dan memperbaiki potensi masalah.
3. Akurasi Simulasi: Hasil simulasi menunjukkan akurasi yang tinggi dalam mereplikasi komunikasi API, sehingga memberikan gambaran yang realistis mengenai kinerja sistem.
4. Pengurangan Biaya: Tidak adanya kebutuhan untuk perangkat fisik selama tahap pengembangan dan pengujian mengurangi biaya secara signifikan.

Namun, terdapat beberapa keterbatasan dalam simulasi ini, seperti keterbatasan dalam meniru latensi jaringan atau skenario fisik yang kompleks. Oleh karena itu, pengujian lanjutan menggunakan perangkat fisik tetap disarankan untuk memastikan kinerja optimal.

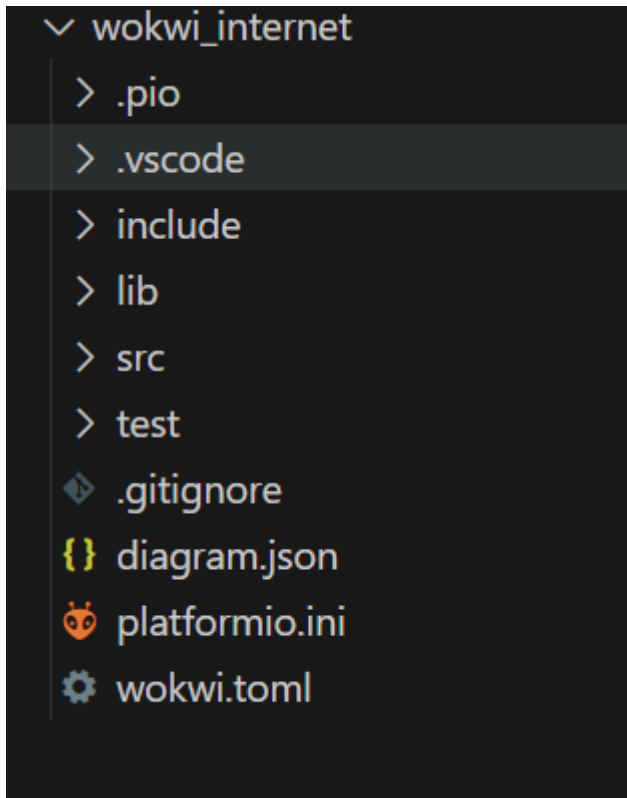
#### **3.3 Kesimpulan**

Dari hasil simulasi yang dilakukan, dapat disimpulkan bahwa Wokwi merupakan alat yang efektif dan efisien dalam mensimulasikan akses API untuk aplikasi berbasis IoT. Dengan kemampuannya dalam mereplikasi komunikasi API secara akurat, Wokwi mampu mempercepat proses pengembangan, mengurangi biaya, dan meningkatkan kualitas aplikasi yang dikembangkan. Meskipun demikian, pengujian pada perangkat fisik tetap diperlukan sebagai langkah akhir sebelum implementasi di dunia nyata.

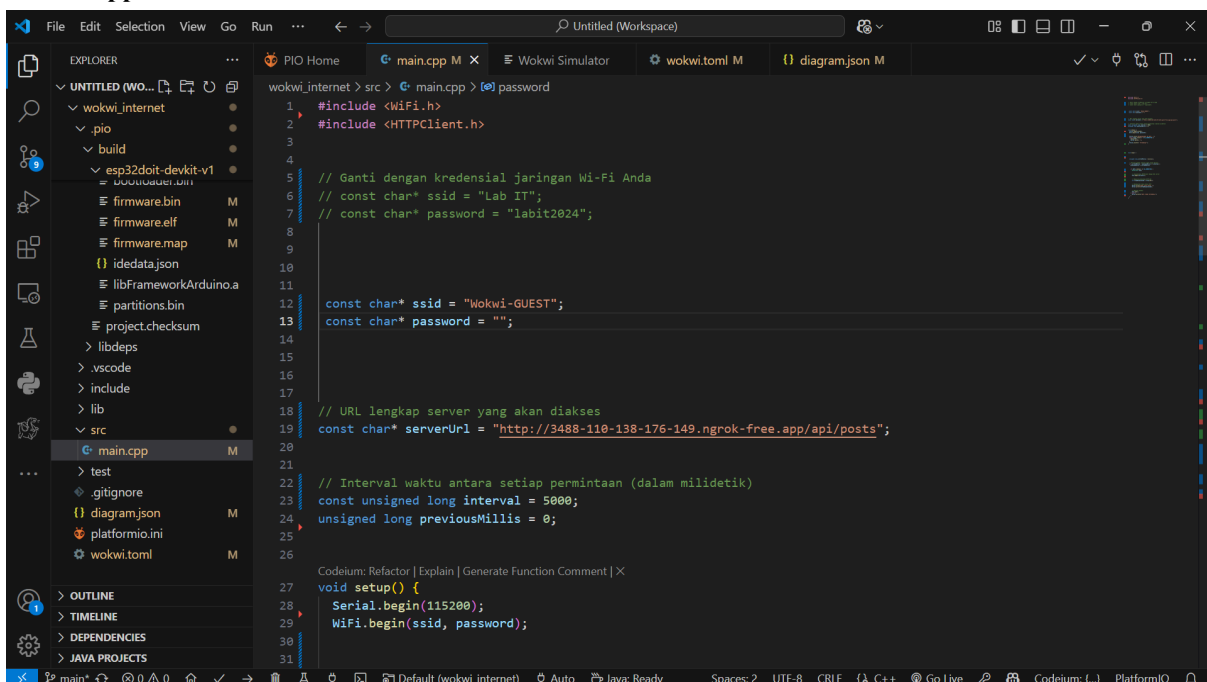
## Bab 4

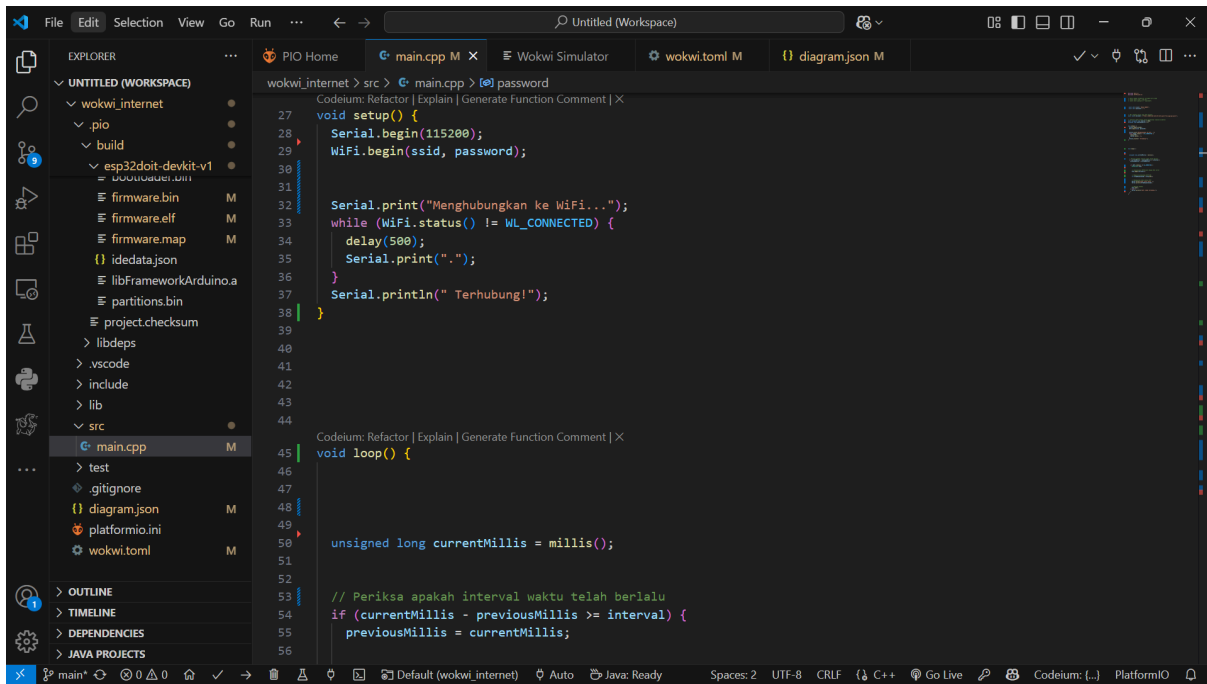
### Lampiran & Dokumentasi

#### 1. Membuat file baru di wokwi simulator bernama wokwi\_internet

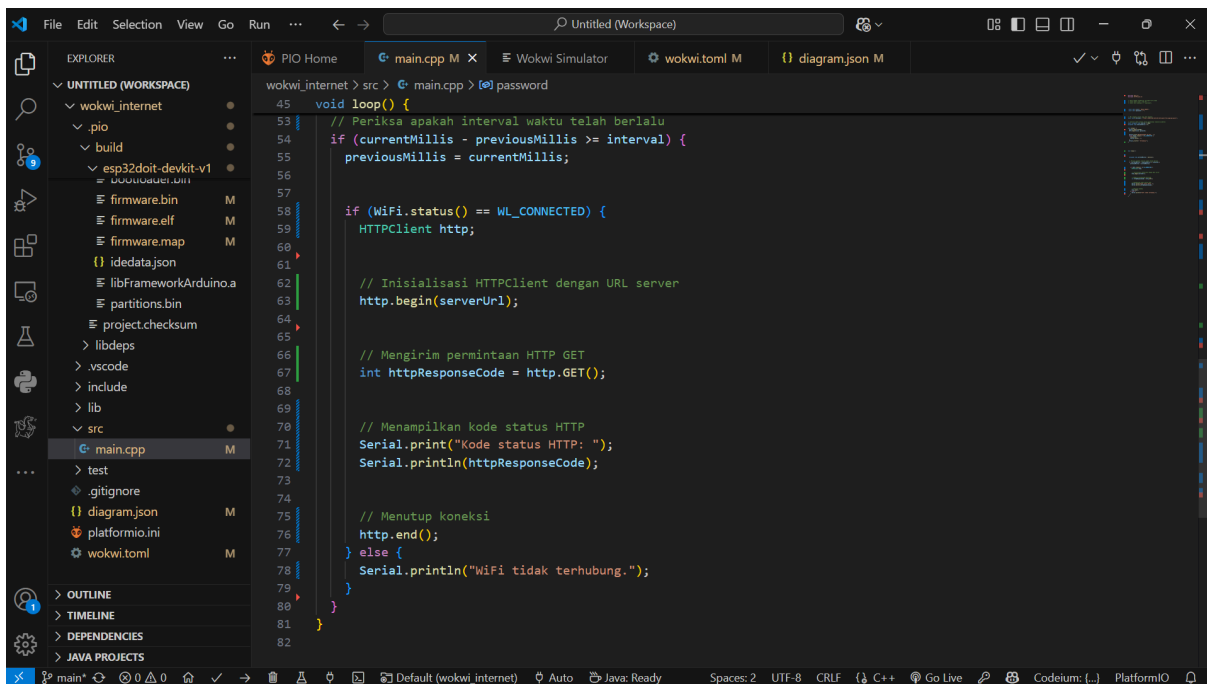


#### 2. Main.cpp





```
27 void setup() {
28     Serial.begin(115200);
29     WiFi.begin(ssid, password);
30
31     Serial.print("Menghubungkan ke WiFi...");
32     while (WiFi.status() != WL_CONNECTED) {
33         delay(500);
34         Serial.print(".");
35     }
36     Serial.println(" Terhubung!");
37 }
38
39
40
41
42
43
44
45 void loop() {
46
47
48
49
50
51
52
53 // Periksa apakah interval waktu telah berlalu
54 if (currentMillis - previousMillis >= interval) {
55     previousMillis = currentMillis;
56 }
```



```
53 // Periksa apakah interval waktu telah berlalu
54 if (currentMillis - previousMillis >= interval) {
55     previousMillis = currentMillis;
56
57     if (WiFi.status() == WL_CONNECTED) {
58         HTTPClient http;
59
60         // Inisialisasi HTTPClient dengan URL server
61         http.begin(serverUrl);
62
63         // Mengirim permintaan HTTP GET
64         int httpResponseCode = http.GET();
65
66         // Menampilkan kode status HTTP
67         Serial.print("Kode status HTTP: ");
68         Serial.println(httpResponseCode);
69
70         // Menutup koneksi
71         http.end();
72     } else {
73         Serial.println("WiFi tidak terhubung.");
74     }
75 }
76
77
78
79
80
81
82 }
```

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

```
// Ganti dengan kredensial jaringan Wi-Fi Anda
// const char* ssid = "Lab IT";
// const char* password = "labit2024";
```

```
const char* ssid = "Wokwi-GUEST";  
const char* password = "";
```

```
// URL lengkap server yang akan diakses  
const char* serverUrl = "http://3488-110-138-176-149.ngrok-free.app/api/posts";
```

```
// Interval waktu antara setiap permintaan (dalam milidetik)  
const unsigned long interval = 5000;  
unsigned long previousMillis = 0;
```

```
void setup() {  
  Serial.begin(115200);  
  WiFi.begin(ssid, password);
```

```
  
  Serial.print("Menghubungkan ke WiFi...");  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.println(" Terhubung!");  
}
```

```
void loop() {
```

```
  unsigned long currentMillis = millis();
```

```
  // Periksa apakah interval waktu telah berlalu  
  if (currentMillis - previousMillis >= interval) {  
    previousMillis = currentMillis;
```

```
    if (WiFi.status() == WL_CONNECTED) {  
      HTTPClient http;
```

```
      // Inisialisasi HTTPClient dengan URL server  
      http.begin(serverUrl);
```



```
// Mengirim permintaan HTTP GET
int httpResponseCode = http.GET();

// Menampilkan kode status HTTP
Serial.print("Kode status HTTP: ");
Serial.println(httpResponseCode);

// Menutup koneksi
http.end();
} else {
    Serial.println("WiFi tidak terhubung.");
}
}
}
```

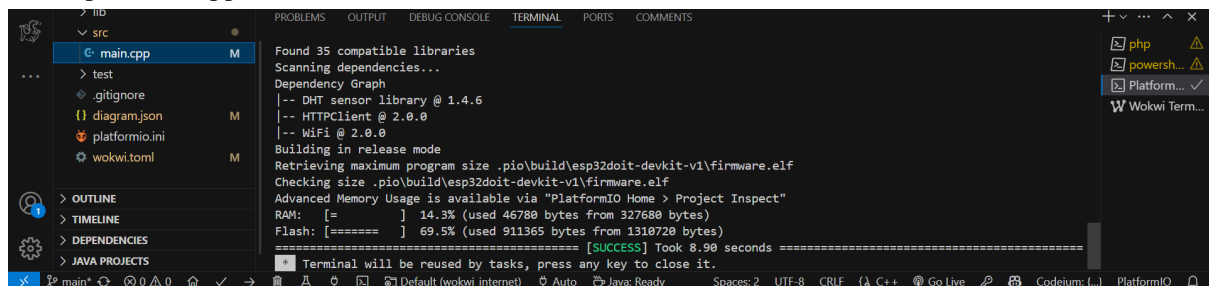
### 3. wokwi.toml

```
wokwi_internet > wokwi.toml
1  [wokwi]
2  version = 1
3  firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
4  elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'
5
```

### 4. diagram.json

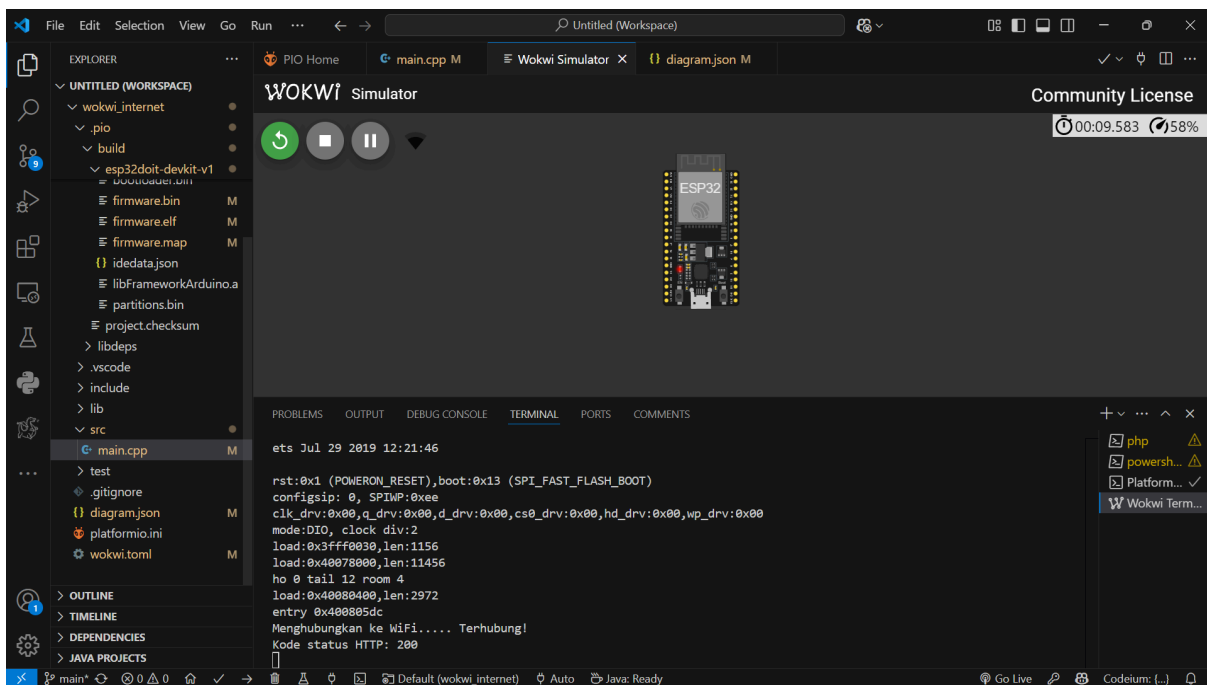
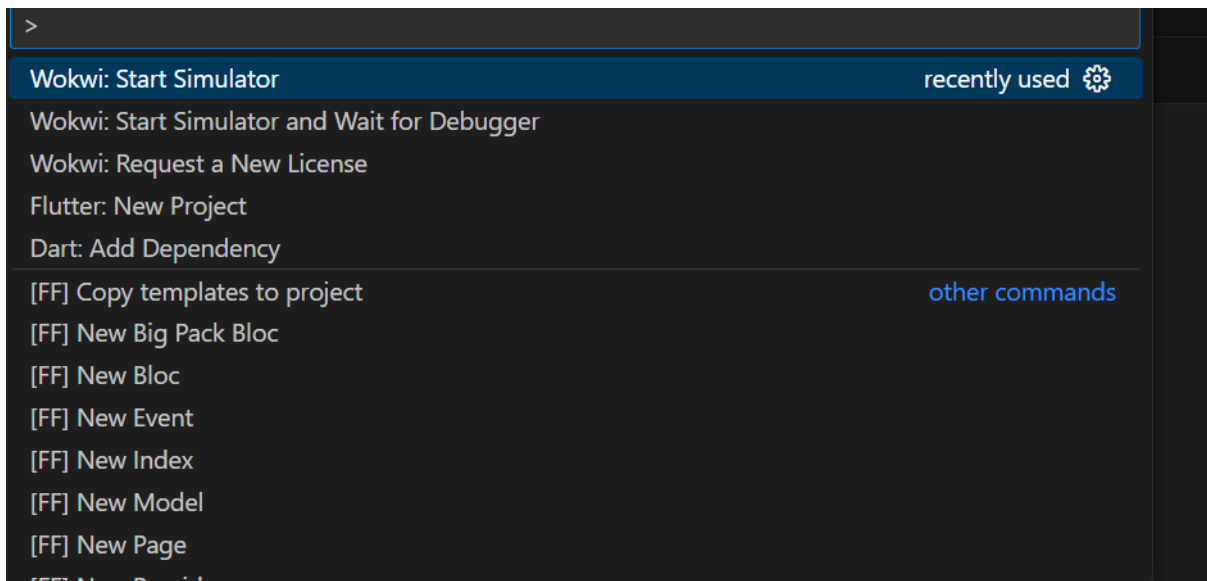
```
wokwi_internet > {} diagram.json > ...
1  {
2    "version": 1,
3    "author": "Uri Shaked",
4    "editor": "wokwi",
5    "parts": [ [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} } ],
6    "connections": [ [ "esp:TX", "$serialMonitor:RX", "", [ ] ], [ "esp:RX", "$serialMonitor:TX", "", [ ] ] ]
7  }
8
```

### 5. Compile main.cpp

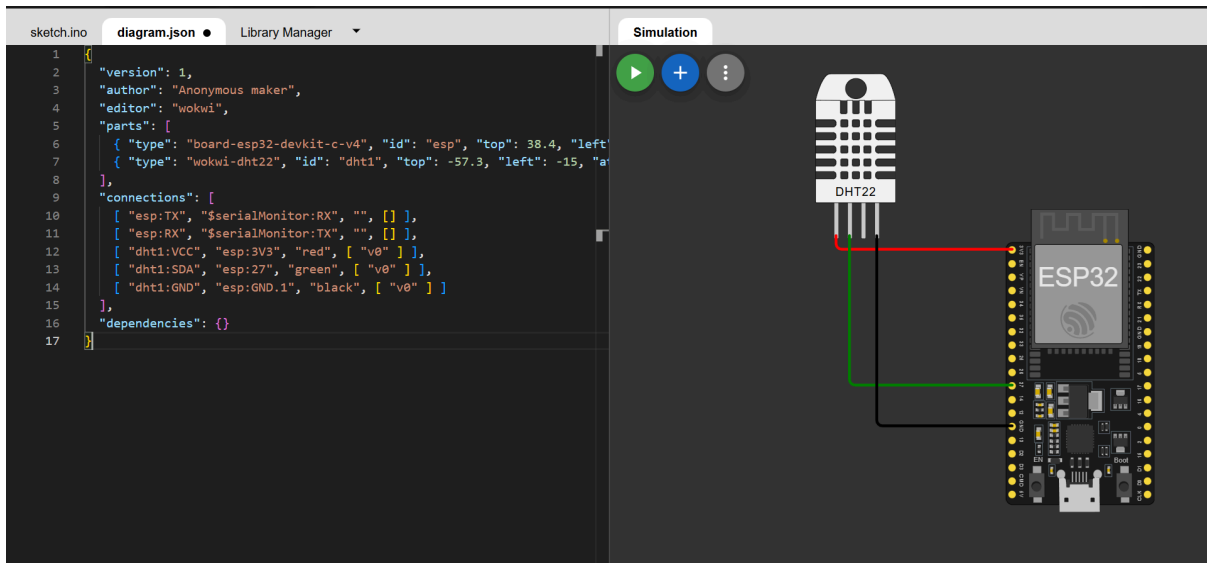


```
Found 35 compatible libraries
Scanning dependencies...
Dependency Graph
|-- DHT sensor library @ 1.4.6
|-- HTTPClient @ 2.0.0
|-- WiFi @ 2.0.0
Building in release mode
Retrieving maximum program size .pio\build\esp32doit-devkit-v1\firmware.elf
Checking size .pio\build\esp32doit-devkit-v1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [=====] 14.3% (used 46780 bytes from 327680 bytes)
Flash: [=====] 69.5% (used 911365 bytes from 1310720 bytes)
===== [SUCCESS] Took 8.90 seconds =====
Terminal will be reused by tasks, press any key to close it.
```

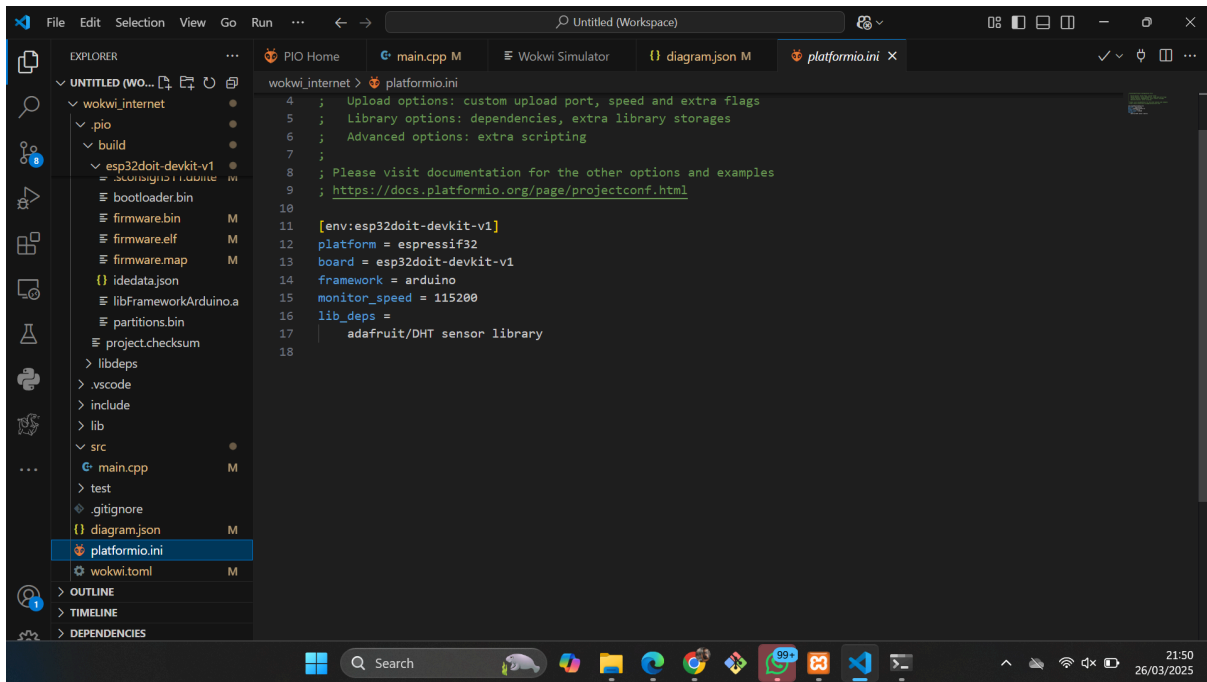
### 6. Start wokwi simulator



## 7. Menggunakan rangkaian sensor DHT 22 menggunakan ESP32



## 8. Modifikasi pada file wokwi.toml



## 9. Modifikasi juga pada file main.cpp lalu build

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>

#include "DHT.h"

#define DHTPIN 27
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

// Ganti dengan kredensial WiFi Anda
const char* ssid = "Wokwi-GUEST";
```

```
const char* password = "";

unsigned long previousMillis = 0;
const long interval = 5000; // Interval 5 detik (5000 ms)

void setup() {
    Serial.begin(115200);

    // Hubungkan ke WiFi
    WiFi.begin(ssid, password);
    Serial.print("Menghubungkan ke WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println(" Terhubung!");

    dht.begin();

    // Tunggu sebentar agar koneksi stabil
    delay(1000);
}

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

    // Lakukan POST setiap interval yang telah ditentukan
    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;
    }
}
```

```

float h = round(dht.readHumidity());
// Read temperature as Celsius (the default)
float t = round(dht.readTemperature());

// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
}

// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);

// Inisialisasi HTTPClient
HTTPClient http;
String url =
"http://3488-110-138-176-149.ngrok-free.app/api/posts"; // Ganti dengan
URL ngrok yang benar

http.begin(url); // Menggunakan HTTP, bukan HTTPS
http.addHeader("Content-Type", "application/json");

String payload = "{\"nama_sensor\":\"Sensor GD\", \"nilai1\":\"" +
String(h) + ", \"nilai2\":\"" + String(t) + "\"}";

Serial.println(payload); // Untuk melihat apakah payload sudah
terbentuk dengan benar

```

```

// Kirim POST request
int httpResponseCode = http.POST(payload);

// Tampilkan kode respons HTTP
Serial.print("Kode respons HTTP: ");
Serial.println(httpResponseCode);

// Tampilkan respons dari server jika request berhasil
if (httpResponseCode == 200 || httpResponseCode == 201) {
    String response = http.getString();
    Serial.println("Respons dari server:");
    Serial.println(response);
} else {
    Serial.println("Gagal mengirim data");
}

// Tutup koneksi HTTP
http.end();
}
}

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 35 compatible libraries
Scanning dependencies...
Dependency Graph
|-- DHT sensor library @ 1.4.6
|-- HTTPClient @ 2.0.0
|-- WiFi @ 2.0.0
Building in release mode
Retrieving maximum program size .pio\build\esp32doit-devkit-v1\firmware.elf
Checking size .pio\build\esp32doit-devkit-v1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ] 14.3% (used 46808 bytes from 327680 bytes)
Flash: [=====   ] 70.5% (used 923781 bytes from 1310720 bytes)
===== [SUCCESS] Took 9.53 seconds =====
* Terminal will be reused by tasks, press any key to close it.

```

## 10. Hasil

