

**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)
PRAKTIK AKSES API MELALUI SIMULASI WOKWI**



Nadia Aulia Zahra
Fakultas Vokasi, Universitas Brawijaya
Email: nadiaaulia@student.ub.ac.id

**Fakultas Vokasi
Universitas Brawijaya
Tahun Ajaran 2025**

ABSTRAK

Dalam pengembangan perangkat IoT (Internet of Things), akses API menjadi salah satu aspek penting untuk memungkinkan komunikasi antara perangkat keras dan layanan cloud. Penelitian ini membahas praktik akses API melalui simulasi menggunakan wokwi, sebuah platform emulator berbasis web untuk mikrokontroler seperti ESP32 dan Arduino. Dengan menggunakan wokwi, pengujian dan simulasi dapat dilakukan tanpa memerlukan perangkat fisik, sehingga mempercepat proses pengembangan serta mengurangi biaya. Studi ini mengeksplorasi bagaimana perangkat virtual dalam wokwi dapat mengakses API untuk mengirim dan menerima data dari server eksternal. Hasil simulasi menunjukkan bahwa wokwi mampu mereplikasi perilaku perangkat IoT dalam berkomunikasi dengan layanan berbasis API, termasuk pengiriman data sensor dan penerimaan perintah dari server. Implementasi ini memberikan solusi efisien bagi pengembang dalam menguji akses API sebelum diterapkan pada perangkat fisik.

Kata kunci: *IoT, API, wokwi, simulasi, ESP32, Arduino*

1. Pendahuluan

1.1. Latar Belakang

Dalam pengembangan aplikasi berbasis web, API (Application Programming Interface) memiliki peran penting dalam menghubungkan berbagai layanan dan perangkat. Laravel 11 sebagai framework PHP modern menyediakan fitur yang memudahkan pembuatan API yang efisien, aman, dan scalable. Namun, dalam pengujian API yang berjalan di server lokal, sering kali terdapat keterbatasan dalam mengaksesnya dari jaringan eksternal. Untuk mengatasi masalah ini, digunakan Ngrok, sebuah layanan tunneling yang memungkinkan server lokal diakses melalui internet tanpa konfigurasi jaringan yang kompleks. Dengan menggunakan Ngrok, API yang dibuat dapat diuji dan digunakan dari mana saja tanpa perlu deploy ke hosting terlebih dahulu.

1.2. Tujuan Eksperimen

Tujuan dari eksperimen ini adalah:

1. Mengembangkan API menggunakan Laravel 11 untuk menangani permintaan dan respons data.
2. Menggunakan Ngrok untuk membuat API yang berjalan di server lokal dapat diakses dari internet.
3. Menguji aksesibilitas API melalui berbagai perangkat dan jaringan eksternal.
4. Mengevaluasi performa dan keamanan API saat diakses melalui Ngrok.

2. Metodologi

2.1. Tools & Materials (Alat dan Bahan)

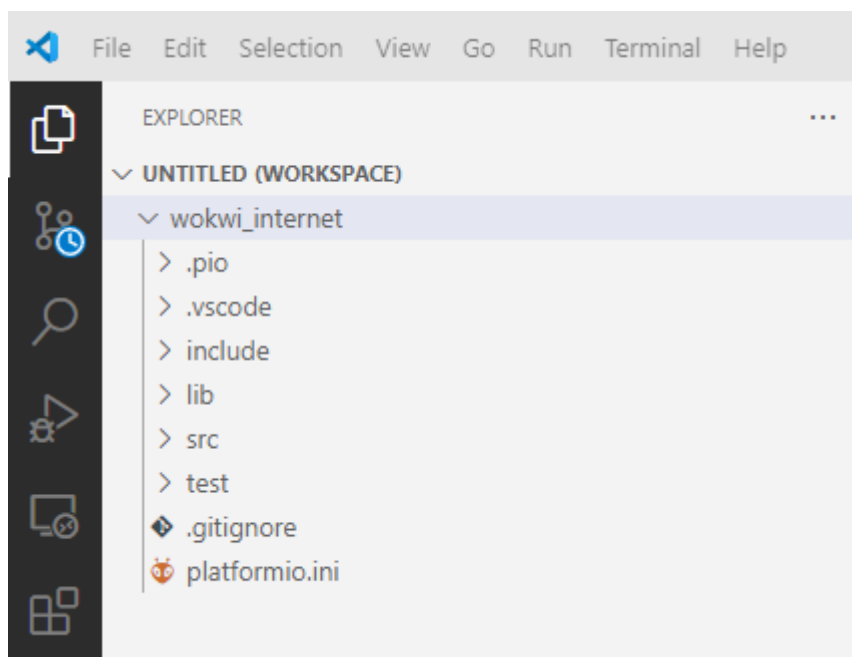
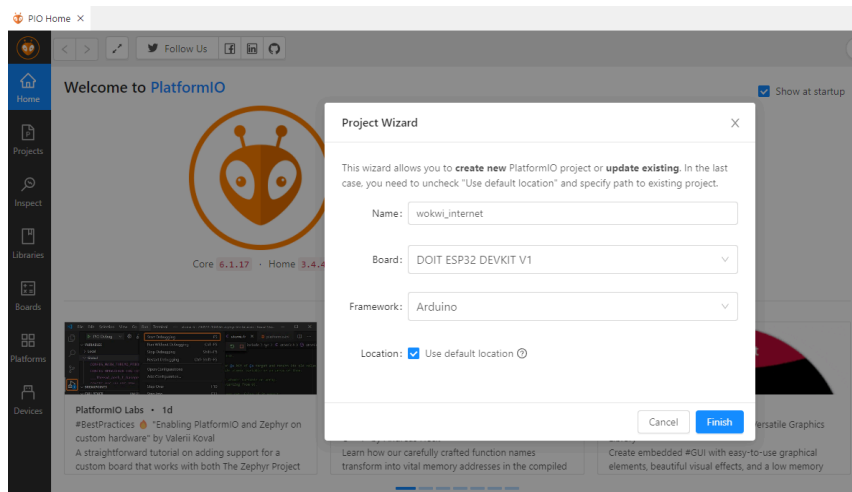
- Perangkat Lunak:
 - Laravel 11
 - PHP 8+
 - Composer
 - Ngrok
 - Postman atau aplikasi serupa untuk menguji API
- Perangkat Keras:
 - Laptop atau PC dengan sistem operasi Windows/Linux/macOS
 - Koneksi internet yang stabil

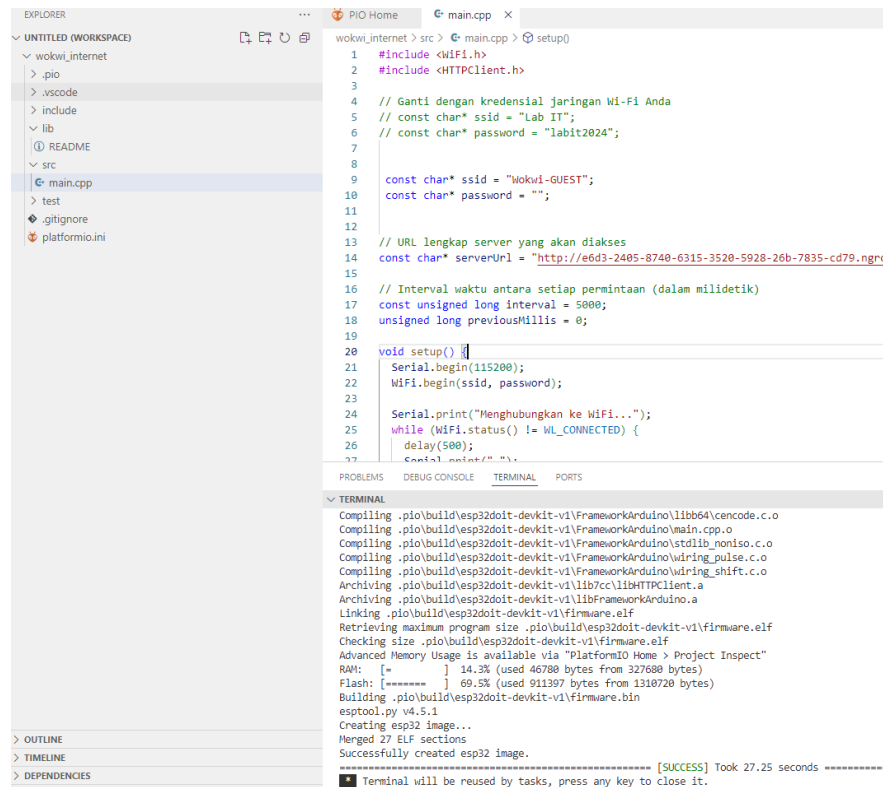
2.2. Langkah Implementasi

Jalankan API laravel dengan perintah **php artisan serve --host=0.0.0.0 --port=8080**

Perintah diatas memastikan API laravel dapat diakses dari IP Address manapun dan memastikan bekerja pada port 8080.

Buat file baru wokwi simulator di platform.io dengan nama file wokwi_internet





The screenshot shows the PlatformIO IDE interface. On the left, the Explorer pane displays the project structure for 'wokwi_internet', including folders like .pio, .vscode, include, lib, and src, and files like main.cpp, test, .gitignore, and platformio.ini. The main.cpp file is selected and its content is shown in the editor. The code defines Wi-Fi credentials, a server URL, and a loop that connects to the Wi-Fi and sends a GET request to the server. The terminal at the bottom shows the compilation and upload process, indicating a successful upload of the firmware to the ESP32 device.

```
1 #include <WiFi.h>
2 #include <HTTPClient.h>
3
4 // Ganti dengan kredensial jaringan Wi-Fi Anda
5 // const char* ssid = "Lab IT";
6 // const char* password = "labit2024";
7
8
9 const char* ssid = "Wokwi-GUEST";
10 const char* password = "";
11
12
13 // URL lengkap server yang akan diakses
14 const char* serverUrl = "http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok";
15
16 // Interval waktu antara setiap permintaan (dalam milidetik)
17 const unsigned long interval = 5000;
18 unsigned long previousMillis = 0;
19
20 void setup() {
21   Serial.begin(115200);
22   WiFi.begin(ssid, password);
23
24   Serial.print("Menghubungkan ke WiFi...");
25   while (WiFi.status() != WL_CONNECTED) {
26     delay(500);
27     Serial.print(".");
28   }
29 }
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
Compiling .pio\build\esp32doit-devkit-v1\framework-arduino\lib64\cencode.c.o
Compiling .pio\build\esp32doit-devkit-v1\framework-arduino\main.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\framework-arduino\stdlib_noniso.c.o
Compiling .pio\build\esp32doit-devkit-v1\framework-arduino\wiring_pulse.c.o
Compiling .pio\build\esp32doit-devkit-v1\framework-arduino\wiring_shift.c.o
Archiving .pio\build\esp32doit-devkit-v1\lib7cc\libHTTPClient.a
Archiving .pio\build\esp32doit-devkit-v1\lib7cc\libframework-arduino.a
Linking .pio\build\esp32doit-devkit-v1\firmware.elf
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 911397 bytes from 1310720 bytes)
Building .pio\build\esp32doit-devkit-v1\firmware.bin
esptool.py v4.5.1
Creating esp32 image...
Merged 27 ELF sections
Successfully created esp32 image.
===== [SUCCESS] Took 27.25 seconds =====
Terminal will be reused by tasks, press any key to close it.
```

Berikut adalah script main.cpp

```
#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://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.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.");
        }
    }
}

```

Perhatikan pada bagian

```

// URL lengkap server yang akan diakses
const char* serverUrl = "http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts";

```

URL diatas adalah URL hasil dari generate perintah NGROK di komputer Anda. Sesuaikan dengan alamat URL yang diberikan oleh NGROK. Cara menjalankan perintah NGROK berbeda dengan Bab sebelumnya, perintah berikut memastikan NGROK memberikan alamat URL dalam bentuk **http** bukan **https**. Pada

saat ini ESP32 yang digunakan hanya support http sehingga pastikan NGROK memberikan URL dalam bentuk http bukan https.

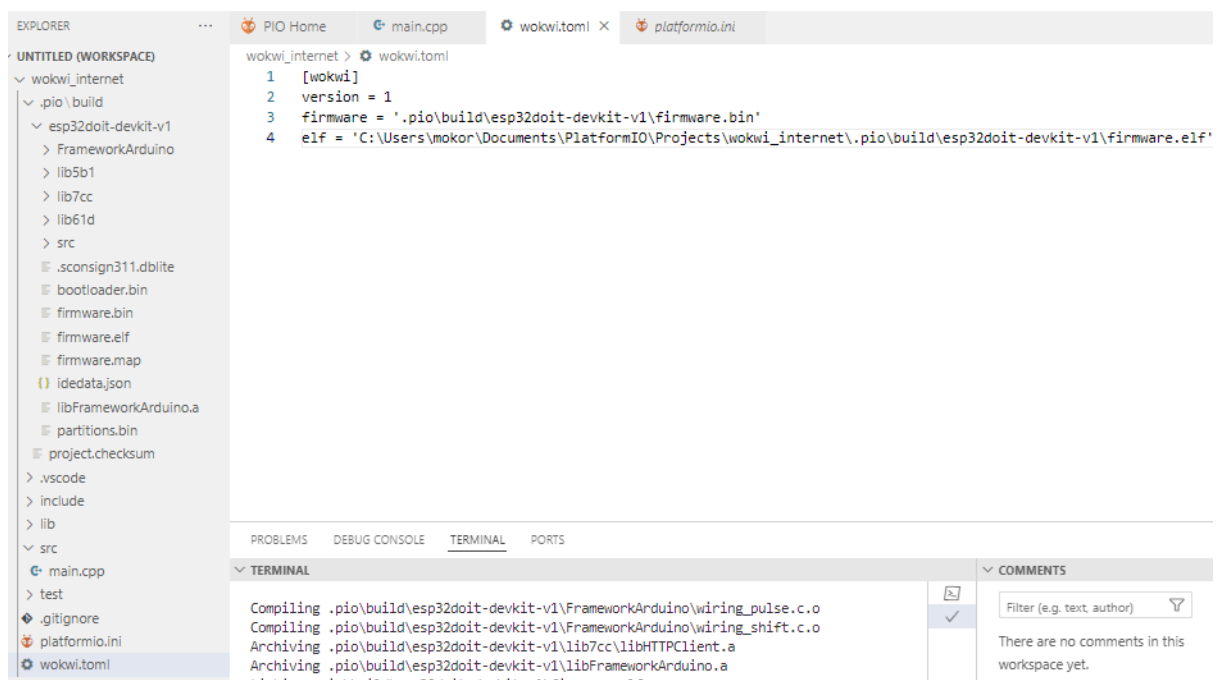
ngrok http --scheme=http 8080. Sesuaikan alamat port 8080 sesuai dengan port berjalannya aplikasi Laravel Anda.

```
ngrok (Ctrl+C to quit)
Route traffic by anything: https://ngrok.com/r/iep

Session Status      online
Account             mokoraden (Plan: Free)
Version             3.20.0
Region              Asia Pacific (ap)
Latency             25ms
Web Interface       http://127.0.0.1:4040
Forwarding           http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app -> http://localhost:8080

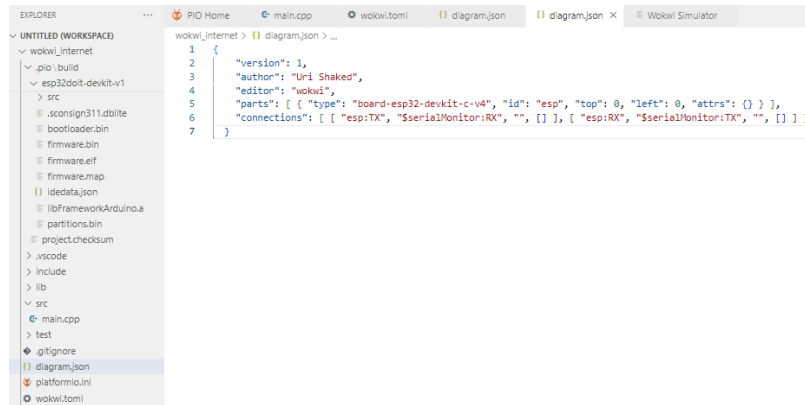
Connections          ttl    opn    rt1    rt5    p50    p90
                    280    0      0.20   0.22   0.25   0.27
```

Tambahkan file wokwi.toml



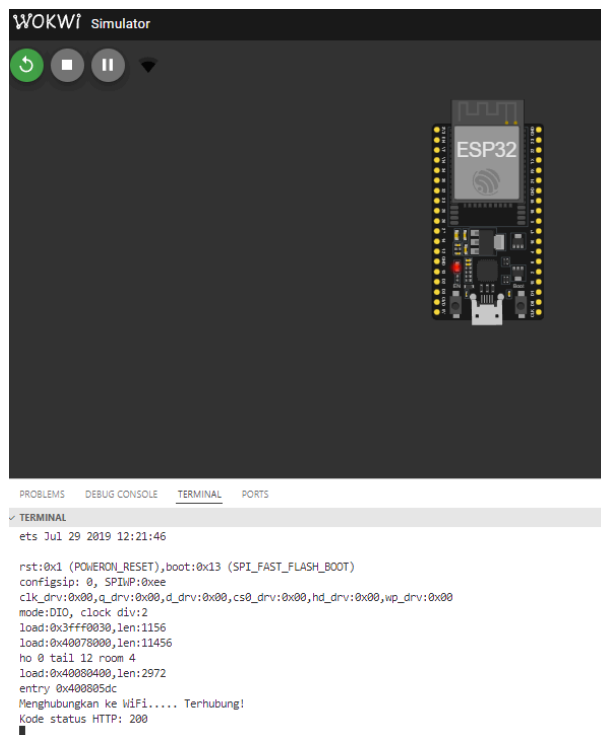
```
[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf='C:\Users\mokor\Documents\PlatformIO\Projects\wokwi_internet\.pio\build\esp32doit-devkit-v1\firmware.elf'
```

Tambahkan file **diagram.json**



```
{
  "version": 1,
  "author": "Uri Shaked",
  "editor": "wokwi",
  "parts": [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": { } } ],
  "connections": [ [ "esp:TX", "$SerialMonitor:RX", "", [ ] ], [ "esp:RX", "$SerialMonitor:TX", "", [ ] ] ]
}
```

Langkah berikutnya adalah melakukan simulasi. Build file main.cpp dan jalankan simulasi dengan perintah
> **Wokwi Start Simulator**

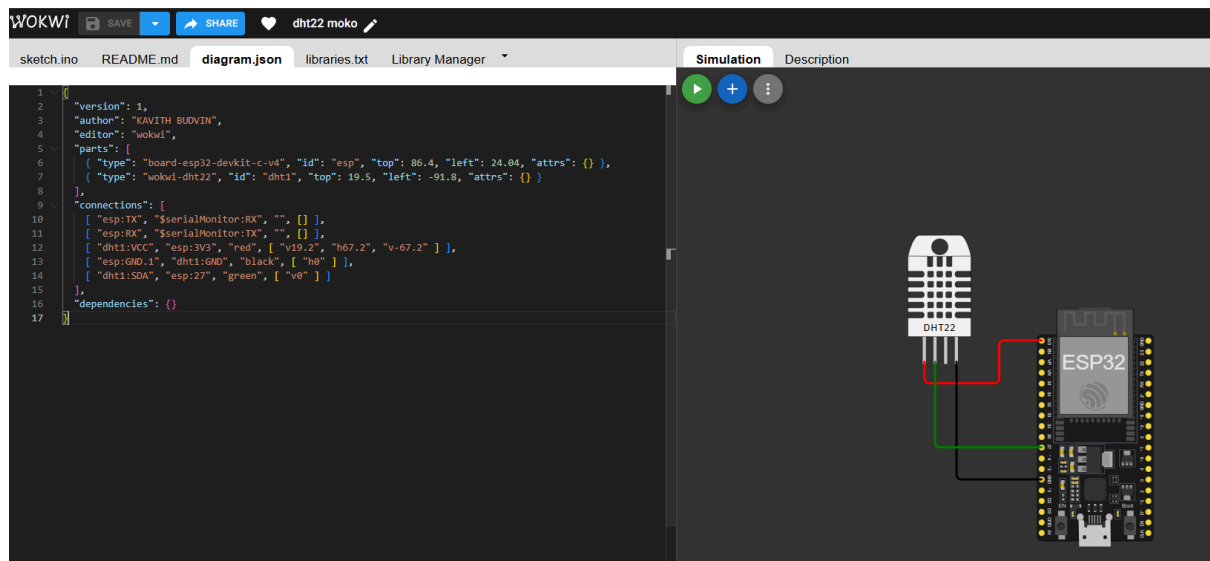


Simulasi diatas menunjukkan, ESP32 berhasil terhubung ke WIFI Wokwi-GUEST dan berhasil mengakses API laravel yang sudah dibuat pada bab sebelumnya.

Kode Status HTTP:200

HTTP status code 200 artinya adalah "OK". Ini berarti bahwa permintaan (request) yang dikirim oleh klien (misalnya browser web atau aplikasi IoT) telah berhasil diproses oleh server. Dengan kata lain, halaman web atau data yang diminta telah berhasil dikirim kembali oleh server dan ditampilkan dengan benar kepada pengguna.

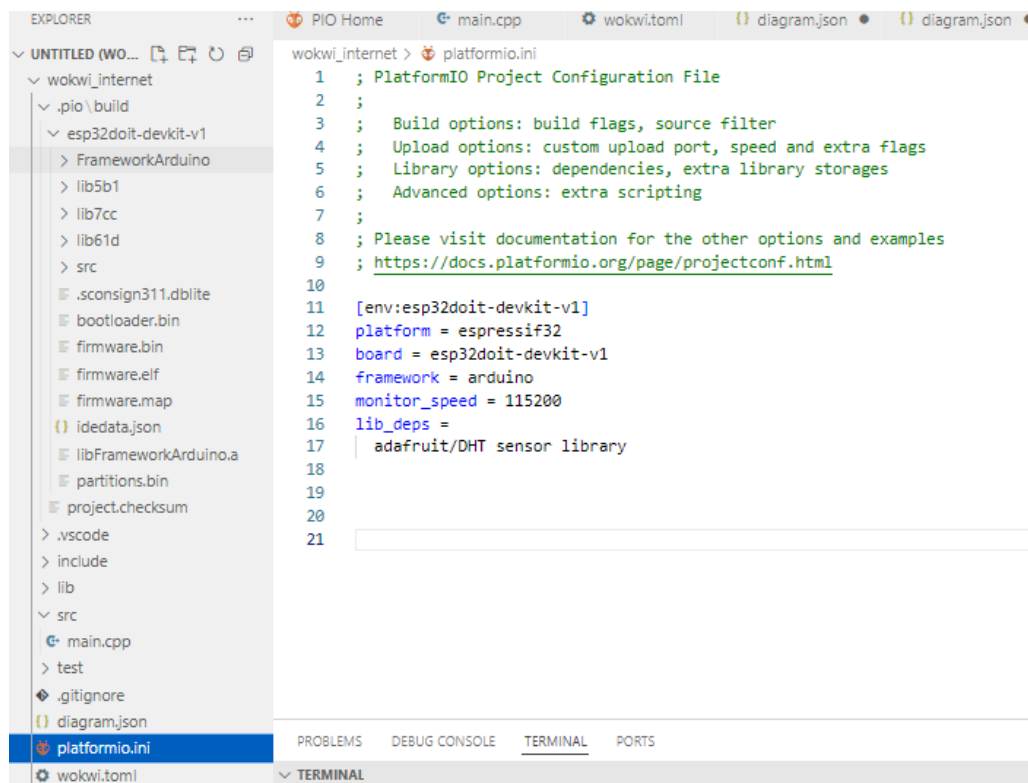
Berikutnya adalah melakukan modifikasi simulasi dengan menambahkan sensor suhu dan kelembaban. Skenarionya adalah, wokwi simulator akan mengirimkan data suhu dan kelembaban ke API dan menyimpannya ke database mysql seperti yang telah dibuat pada bab sebelumnya.



Rangkai sensor DHT22 dengan ESP32 seperti contoh diatas. Kemudian salin kode **diagram.json** ke file diagram.json yang ada di vscode.

```
{
  "version": 1,
  "author": "KAVITH BUDVIN",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 86.4, "left": 24.04, "attrs": {} },
    { "type": "wokwi-dht22", "id": "dht1", "top": 19.5, "left": -91.8, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "dht1:VCC", "esp:3V3", "red", [ "v19.2", "h67.2", "v-67.2" ] ],
    [ "esp:GND.1", "dht1:GND", "black", [ "h0" ] ],
    [ "dht1:SDA", "esp:27", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

Kemudian ubah setting file **platformio.ini** sebagai berikut :



Pada perubahan diatas, ada tambahan 2 setting yaitu monitor speed dan lib_deps

Monitor speed digunakan untuk memonitor status pengiriman data dari wokwi simulator ke server api laravel yang telah dibuat. Sedangkan lib_deps adalah library yang digunakan sensor DHT (sensor suhu dan kelembaban).

Modifikasi file **main.cpp**

```
#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://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.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();
}
}

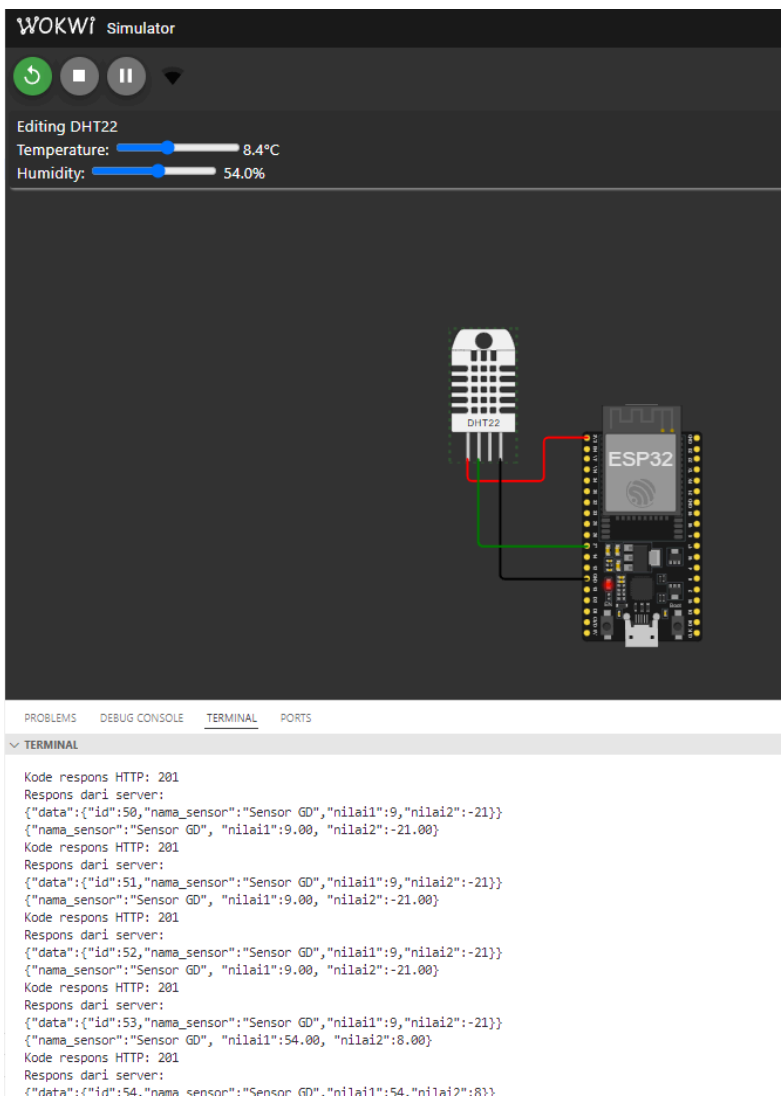
```

Pada bagian berikut sesuaikan dengan URL NGROK anda

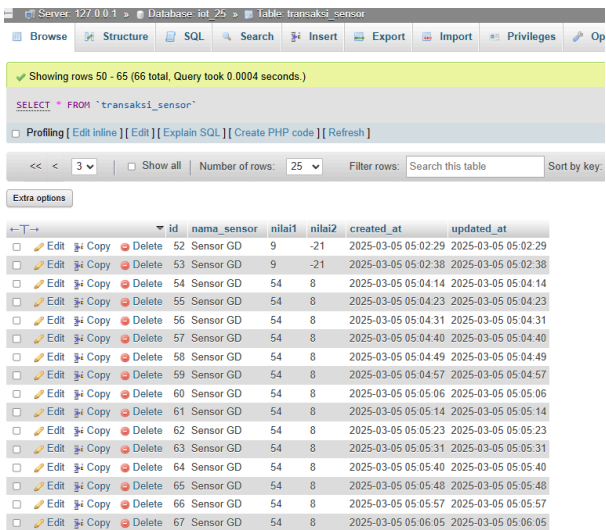
"**http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts**"; // Ganti dengan URL ngrok yang benar

Jalankan simulasi > **Wokwi Start Simulator**

Berikut adalah tampilan pada simulator dan serial monitor.



Pastikan di database, data telah muncul dan tersimpan



3. Hasil dan Pembahasan

3.1 Hasil Eksperimen

Proses pengembangan API menggunakan Laravel 11 berjalan dengan baik. API yang dibuat memiliki endpoint sederhana yang mengembalikan data dalam format JSON. Setelah Laravel berhasil diinstal dan dijalankan di server lokal, API dapat diakses melalui browser maupun aplikasi pengujian seperti Postman. Dalam pengujian awal, API berfungsi dengan baik di localhost tanpa kendala. Respons yang dihasilkan juga cukup cepat karena berjalan langsung di lingkungan pengembangan. Hal ini menunjukkan bahwa Laravel 11 dapat digunakan dengan baik untuk membangun API yang ringan dan responsif. Agar API dapat diakses dari jaringan eksternal, Ngrok digunakan sebagai layanan tunneling. Setelah menjalankan perintah `ngrok http 8000`, sistem menghasilkan URL publik yang memungkinkan API Laravel diakses dari luar jaringan lokal. Setelah mendapatkan URL dari Ngrok, API diuji kembali dengan mengaksesnya dari perangkat lain menggunakan koneksi internet yang berbeda. Hasilnya, API tetap berfungsi dengan baik dan memberikan respons yang sesuai. Penggunaan Ngrok memberikan beberapa keuntungan, di antaranya:

- Memudahkan akses API dari mana saja tanpa harus mengkonfigurasi firewall atau port forwarding.
- Cocok untuk pengujian dan debugging API pada tahap pengembangan.
- Dapat digunakan untuk menghubungkan sistem lokal dengan aplikasi berbasis cloud atau mobile.

Namun, ada beberapa keterbatasan yang ditemukan selama pengujian:

- URL yang dihasilkan oleh Ngrok bersifat sementara, sehingga berubah setiap kali Ngrok dijalankan ulang (kecuali menggunakan akun premium).
- Terdapat sedikit latensi dibandingkan dengan akses langsung ke localhost, meskipun perbedaannya masih dalam batas wajar.

Hasil pengujian menunjukkan bahwa API Laravel 11 yang dihubungkan dengan Ngrok dapat diakses dengan baik dari berbagai perangkat dan jaringan. Latensi tambahan yang muncul saat menggunakan Ngrok masih dalam batas wajar dan tidak menghambat fungsi API secara signifikan.

3.2 Kesimpulan

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa pengembangan API menggunakan Laravel 11 dan Ngrok berjalan dengan baik. API dapat dibuat dengan mudah dan diakses melalui server lokal tanpa kendala. Integrasi dengan Ngrok memungkinkan API untuk diakses dari luar jaringan lokal, sehingga mempermudah proses pengujian dan pengembangan. Ngrok terbukti sebagai solusi yang praktis untuk mengakses API dari luar jaringan lokal tanpa perlu konfigurasi tambahan. Namun, terdapat beberapa keterbatasan seperti URL yang berubah setiap kali Ngrok dijalankan ulang dan sedikit latensi dalam proses pengiriman data. Meskipun demikian, untuk keperluan pengujian dan pengembangan, kombinasi Laravel 11 dan Ngrok sangat direkomendasikan karena kemudahan penggunaannya. Jika API akan digunakan dalam lingkungan produksi, disarankan untuk menggunakan server atau layanan hosting yang lebih stabil untuk menghindari perubahan URL dan meningkatkan kecepatan akses API. Dengan eksperimen ini, dapat disimpulkan bahwa Laravel 11 dan Ngrok dapat digunakan secara efektif untuk membangun API yang dapat diakses secara global dengan cepat dan efisien.

4. Lampiran Jika diperlukan

Hasil pengujian pada praktik yang telah dibuat

