LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktik Akses API melalui Simulasi Wokwi (Minggu 6)**

*Fayola Carani Malya  
Fakultas Vokasi, Universitas Brawijaya  
Email: fayolacarani@gmail.com*

**Abstract**   
Application Programming Interface (API) merupakan sekumpulan protokol dan alat yang memungkinkan perangkat lunak untuk saling berkomunikasi. API menjadi elemen penting dalam pengembangan perangkat lunak modern, terutama dalam integrasi antara perangkat IoT dan layanan berbasis web. Dalam praktikum ini, API digunakan untuk memungkinkan board ESP32, yang disimulasikan menggunakan platform WOKWI, mengakses dan berinteraksi dengan server berbasis Laravel melalui koneksi Wi-Fi. Proses ini melibatkan konfigurasi API, pengiriman permintaan HTTP, dan penerimaan respons dari server. Eksperimen ini bertujuan untuk memahami cara kerja komunikasi antara perangkat IoT dan server, serta mengevaluasi keberhasilan ESP32 dalam mengakses API. Hasil eksperimen menunjukkan bahwa ESP32 berhasil mengirim dan menerima data dengan respons HTTP 200 (OK), yang menandakan permintaan berhasil diproses

Keywords — API, ESP32, IOT, Laravel, Koneksi Wifi.

**1. Introduction** (Pendahuluan)

**1.1 Latar belakang**

Perkembangan teknologi Internet of Things (IoT) telah membawa perubahan signifikan dalam berbagai aspek kehidupan, termasuk dalam pengelolaan data dan otomatisasi perangkat. Salah satu elemen penting dalam ekosistem IoT adalah Application Programming Interface (API), yang memungkinkan perangkat IoT untuk berkomunikasi dengan server atau layanan berbasis web.

Board ESP32 merupakan salah satu perangkat IoT yang populer karena kemampuannya dalam menangani koneksi Wi-Fi dan Bluetooth, serta mendukung berbagai aplikasi yang membutuhkan komunikasi data secara real-time. Untuk memahami lebih jauh mengenai integrasi IoT dengan layanan berbasis web, eksperimen ini menggunakan ESP32 yang disimulasikan melalui platform WOKWI. Dalam eksperimen ini, API Laravel diimplementasikan sebagai server yang akan diakses oleh ESP32 melalui koneksi Wi-Fi.

Simulasi ini dirancang untuk memberikan pemahaman mendalam tentang cara kerja API, proses pengiriman permintaan HTTP, serta penerimaan respons dari server. Dengan adanya eksperimen ini, diharapkan pengguna dapat memahami bagaimana perangkat IoT seperti ESP32 dapat diintegrasikan dengan layanan berbasis web untuk mendukung berbagai kebutuhan teknologi modern.

**1.2 Tujuan eksperimen**

Penelitian ini bertujuan :   
1. Konsep dasar Application Programming Interface (API) dan fungsinya dalam komunikasi antar perangkat

2. Mempelajari cara menghubungkan board ESP32 dengan server berbasis Laravel melalui koneksi Wifi

3. Mensimulasikan proses pengiriman permintaan HTTP dari ESP32 menggunakan platform WOKWi

4. Mengevaluasi respons server terhadap permintaan HTTP dari ESP32

5. Mengaplikasikan teknologi IOT untuk mendukung integrasi perangkat keras dan layanan berbasis web  
6. Mengimplementasikan logika control sederhana untuk mengatur hubungan antara relay, button, dan LED

**2. Methodology (Metodologi)**

**2.1 Tools & Materials (Alat dan Bahan)**

Alat dan Bahan yang digunakan :   
1. Mikrokontroler : ESP 32 Devkit VI  
2. Software : Wokwi, Visual Studio Code, API Laravel, Postman, Wifi (Koneksi Internet yang stabil)

**2.2 Implementation Steps (Langkah Implementasi)**

1. Melakukan simulasi ESP32 di Wokwi untuk praktik Akses API sebelum ke Visual Studio Code (code editor). Pastikan juga computer terhubung dengan internet yang stabil

2. Menjalankan API Laravel dengan kode “php artisan serve –host=0.0.0.0 –port=8000”

3. Membuat file baru di wokwi simulator di platform.io

4. Menuliskan kode program di 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://4c14-175-45-191-14.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();

  }

}

NB: Pastikan url nya dalam bentuk HTTP dan sesuaikan pada alamat port 8000 ngroknya pada masing masing laptop.

5. Tambahkan file wokwi.toml

6. Tambahkan file diagram.json dengan menuliskan code

{

  "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": {}

}

6. Lakukan simulasi dengan klik build main.cpp dan jalankan simulasi dengan perintah “> Wokwi Start Simulator”

7. Cek Kode status, apabila tertulis kode status : 200 berarti berhasil

8. Melakukan modifikasi simulasi dengan menambahkan sensor suhu dan kelembaban. Rangkai seperti pada modul

9. Perbarui kode diagram.json yang ada di vscode. Ubah juga setting platformio.ini dengan menambahkan monitor speed dan lib\_deps

10. Pada main.cpp ada perubahan dan sesuaikan URL ngrok dengan yang telah dijalankan

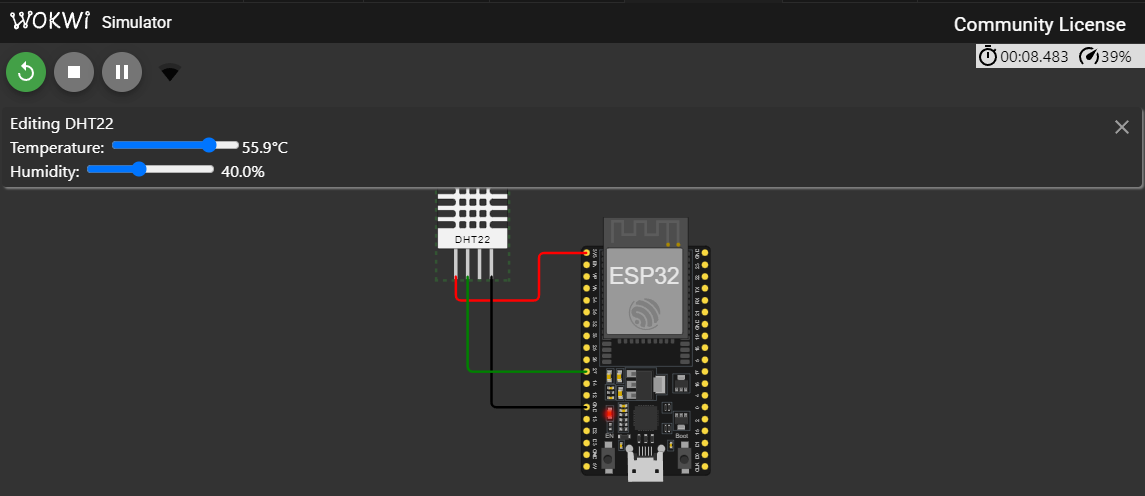
11. Jalankan simulasi wokwi nya dan pastikan pada database data telah tersimpan.

**3. Results and Discussion (Hasil dan Pembahasan)**

**3.1 Experimental Results (Hasil Eksperimen)**

Setelah kode program di unggah di Visual Studio Code, akan memunculkan hasil :   
- Data akan tertampil pada terminal dan disimpan pada Database

**Screenshoot hasil simulasi pada Visual Studio Code :**



**Pembahasan :   
-** Simulasi berjalan sesuai dengan yang diharapkan

**4. Appendix (Lampiran, jika diperlukan)**

{

  "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": {}

}