**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)**

**Fakultas Vokasi, Universitas Brawijaya**

**Praktik Simulasi Relay, Button, dan LED**

*Avrilla Agnesya Meifilistiara*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*avrila.agnesya@gmail.com*](mailto:avrila.agnesya@gmail.com)

**Abstract**

Praktikum ini bertujuan untuk mengimplementasikan sistem kontrol sederhana berbasis Internet of Things (IoT) menggunakan mikrokontroler ESP32. Simulasi dilakukan pada platform Wokwi untuk memahami interaksi antara komponen input berupa push button dan komponen output berupa relay module dan LED. Eksperimen diawali dengan perancangan rangkaian di Wokwi, kemudian dilanjutkan dengan integrasi kode menggunakan Visual Studio Code. Proses implementasi meliputi penambahan komponen, konfigurasi koneksi, dan pengujian. Hasil eksperimen menunjukkan bahwa sistem berhasil mengimplementasikan interaksi yang diharapkan, di mana saat button ditekan, LED merah dan indikator LED pada modul relay menyala secara bersamaan, dan mati saat button dilepas. Praktikum ini menyimpulkan bahwa simulasi berhasil mendemonstrasikan prinsip dasar kontrol input-output dalam sistem IoT sederhana, memberikan pemahaman praktis tentang integrasi komponen elektronik untuk pengembangan sistem IoT yang lebih kompleks.

*Keywords: Internet of Things, ESP32, Relay Module, Push Button, LED, Wokwi, Simulasi IoT*

**1. Introduction**

* 1. **Latar belakang**

Perkembangan teknologi Internet of Things (IoT) sangat berpengaruh dalam kehidupan sehari-hari. Salah satunya sebagai sistem pengendalian perangkat elektronik secara otomatis melalui jarak jauh. Praktik simulasi relay, button, dan LED dilakukan sebagai materi praktik dalam sistem IoT sederhana. Simulasi menggunakan platform wokwi yang bertujuan untuk pembelajaran bagaimana button berfungsi, menganalisis cara kerja relay serta LED dalam simulasi dan rancangan yang telah dibuat.

**1.2 Tujuan eksperimen**

1. Mempelajari cara mengimplementasikan rangkaian relay, button, dan LED menggunakan platform simulasi Wokwi
2. Menganalisis interaksi antara komponen input (button) dan output (relay dan LED) dalam sistem kontrol sederhana
3. Memahami cara mengintregasikan komponen ESP32 dan Relay

**2. Methodology (Metodologi)**

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

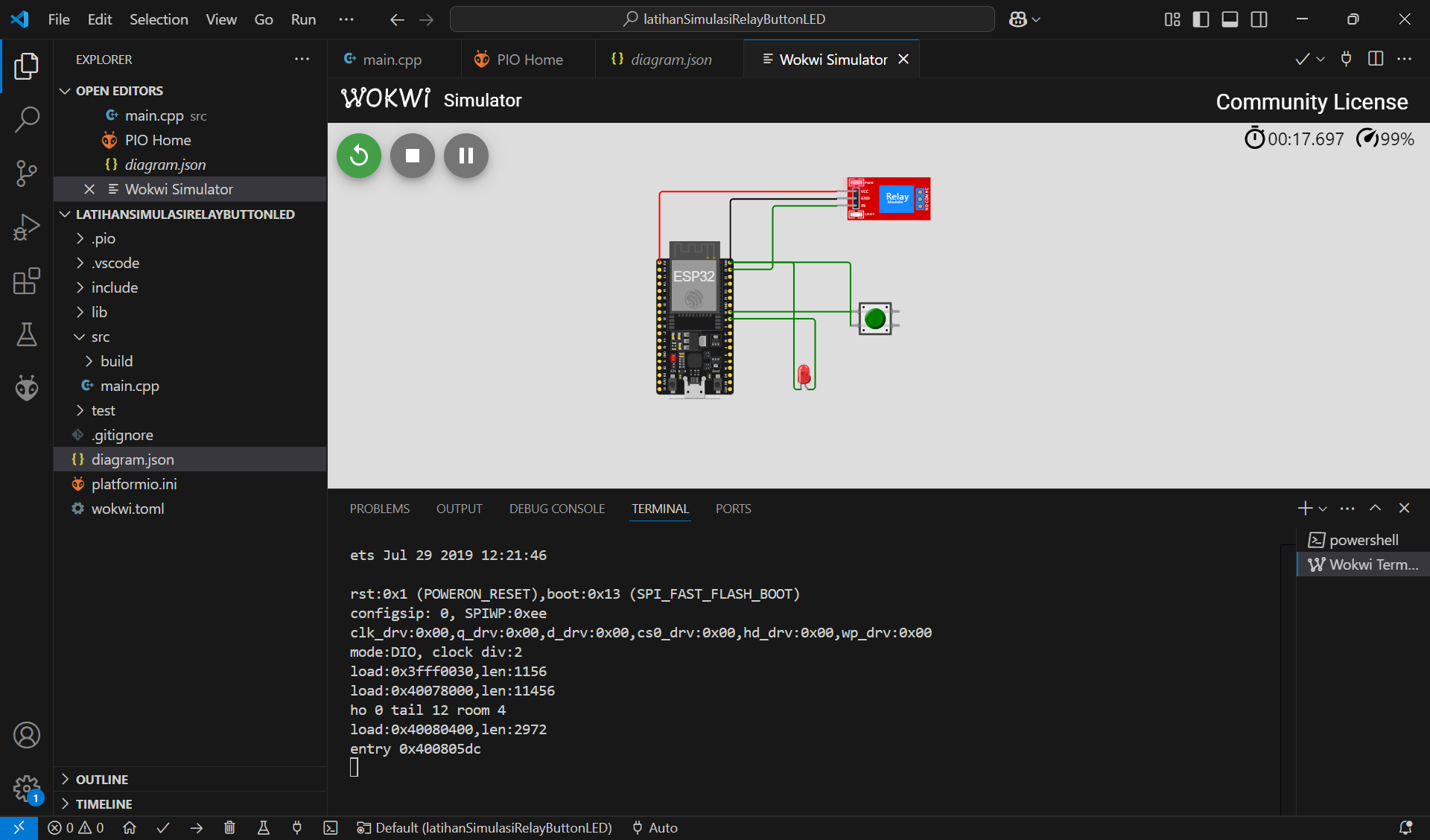
1. Mikrokontroler : ESP32
2. Komponen : Push button, Relay Module, LED, Resistor, dan Jumper wires (kabel penghubung warna merah, hitam, dan hijau)
3. Software : VS Code, Wokwi  
   1. **Implementation Steps (Langkah Implementasi)**
4. Buka platform simulasi Wokwi dan membuat project baru dengan memilih mikrokontroler ESP32
5. Tambahkan komponen ESP32 dan modul relay
6. Tambahkan push button dan LED yang kemudian sambungkan dengan resistornya
7. Menyambungkan code yang sudah ada di wokwi ke vscode, agar dapat dijalankan, setelah sudah diintregasikan maka file main.cpp dapat di build
8. Setelah melakukan build, maka didapati file firmware dan elf, anda dapat melakukan “copy relative path” dan keduanya disalin pada kutip dalam wokwi.toml
9. Jika dalam terminal menunjukkan success, maka diagram json dapat dilakukan simulasi atau mulai >wokwi: request a new license
10. Setelah menyetujui semua, maka simulator dapat dijalankan dengan >wokwi: start simulator
11. Saat button ditekan, maka led yang berwarna merah, akan otomatis menyala bersamaan dengan relay module-nya
12. Dengan langkah-langkah tersebut, simulator sudah benar berjalan.

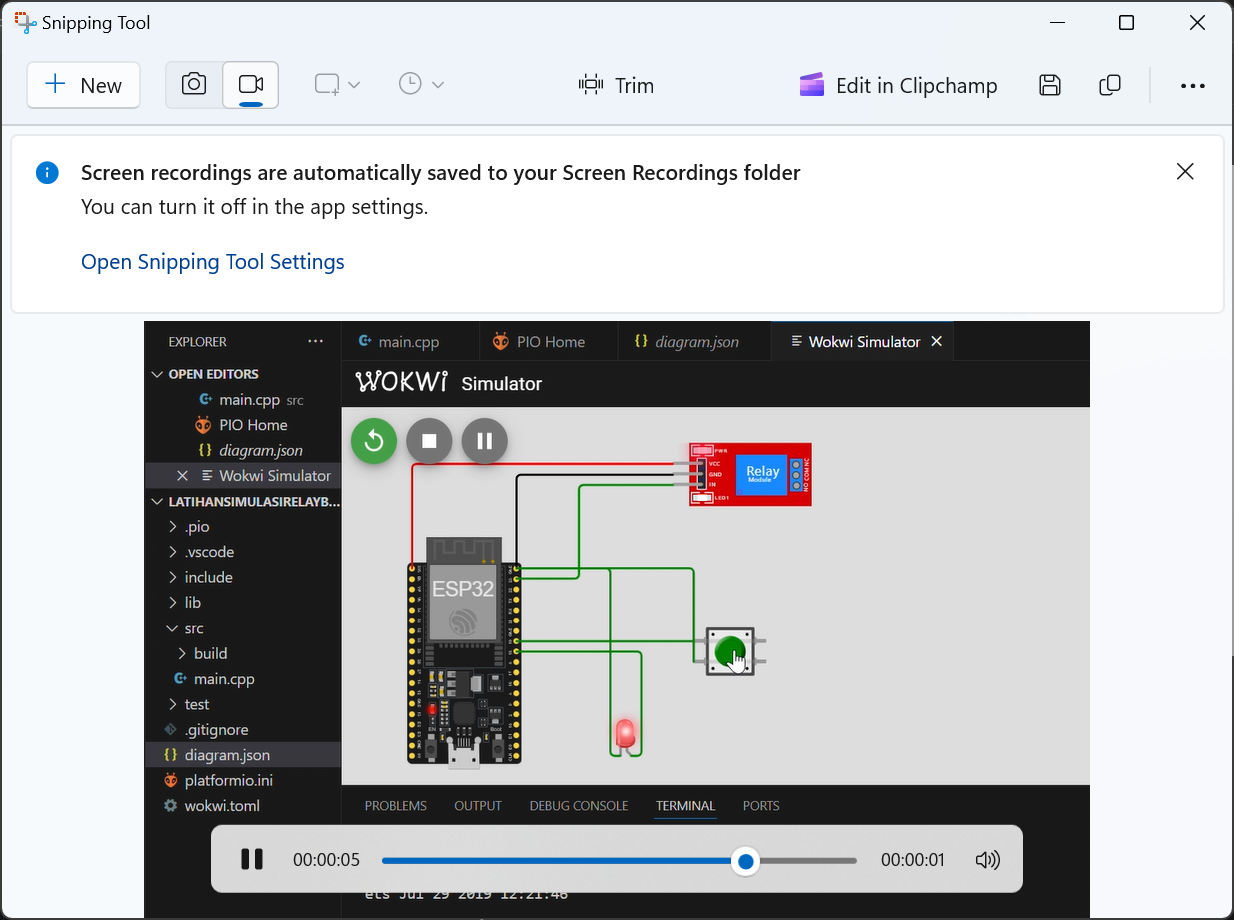
**3. Results and Discussion (Hasil dan Pembahasan)**

**3.1 Experimental Results (Hasil Eksperimen)**

Dari hasil eksperimen yang sudah saya lakukan, bahwa setelah seluruh komponen disambungkan antara mikrokontroler, button, serta relay module memiliki kesinambungan dalam ketiga komponen. Dengan output, saat button tersebut ditekan, makan bagian LED (berwarna merah) akan otomatis menyala serta LED 1 dalam komponen relay module. Jika button tidak ditekan, LED akan mati otomatis.

Berikut hasil outputnya:

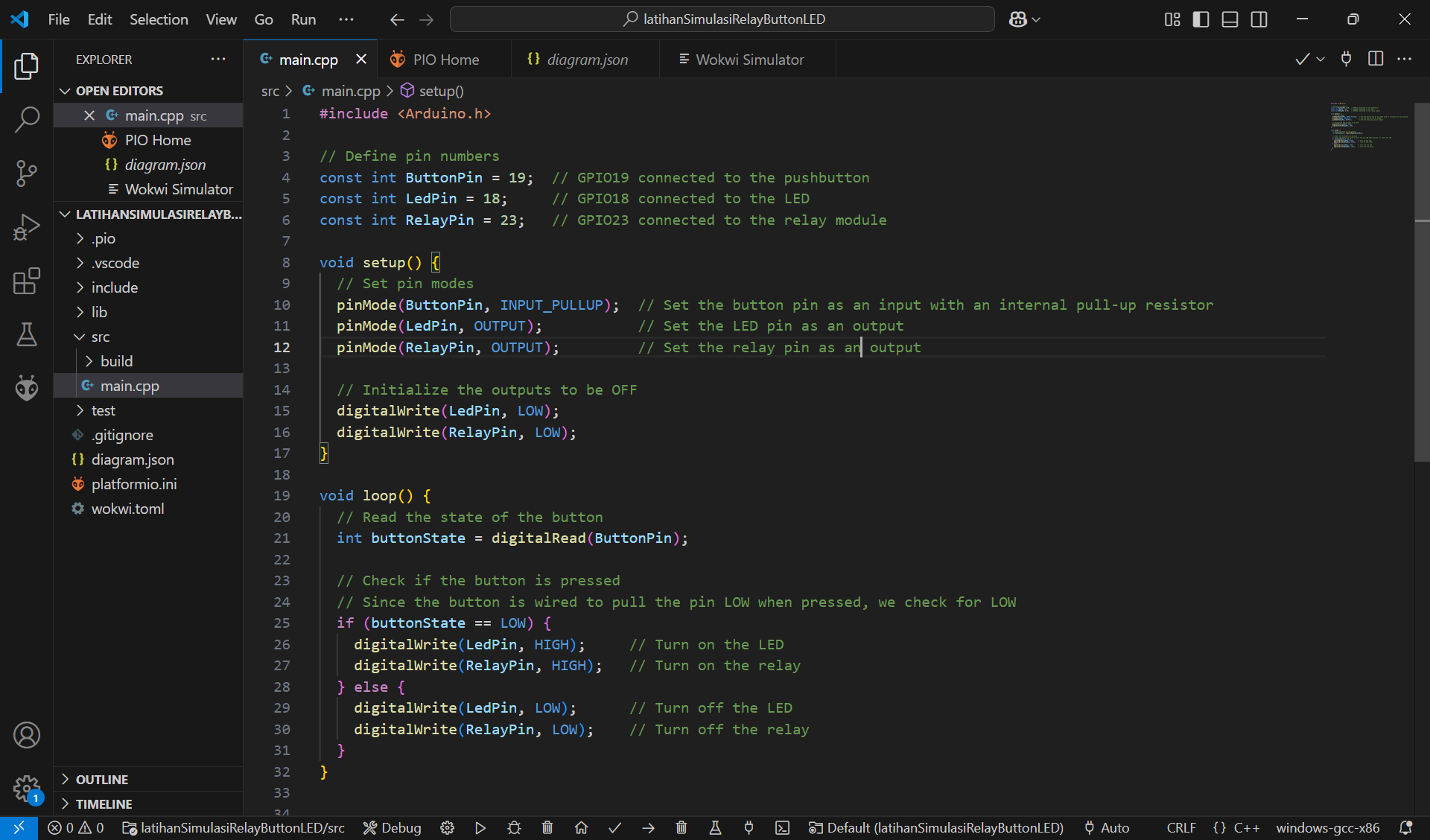




**4. Appendix (Lampiran, jika diperlukan)**

Berikut kode program untuk menjalankan simulator tersebut:

main.cpp:



Kode diagram.json :



**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)**

**Fakultas Vokasi, Universitas Brawijaya**

**Praktik Simulasi Sensor Jarak (Ultrasonic)**

*Avrilla Agnesya Meifilistiara*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*avrila.agnesya@gmail.com*](mailto:avrila.agnesya@gmail.com)

**Abstract**

Eksperimen ini mendemonstrasikan implementasi dan simulasi sensor jarak ultrasonik HC-SR04 menggunakan mikrokontroler ESP32 pada platform Wokwi. Rangkaian berhasil diimplementasikan dengan menghubungkan sensor HC-SR04 ke ESP32 melalui pin-pin yang sesuai. Simulasi menghasilkan pembacaan jarak yang konsisten berkisar antara 399.92-399.98 cm dengan mayoritas pembacaan stabil di nilai 399.94 cm. Hasil ini mengindikasikan bahwa sensor bekerja pada batas jangkauan maksimalnya yaitu sekitar 4 meter, yang sesuai dengan spesifikasi teknis sensor HC-SR04. Eksperimen berhasil mendemonstrasikan prinsip kerja sensor ultrasonik dalam mengukur jarak dan memvalidasi fungsi dasar sistem pendeteksi jarak berbasis ESP32. Melalui simulasi ini, diperoleh pemahaman tentang karakteristik sensor ultrasonik, batasan pengukuran, dan potensi aplikasinya dalam berbagai sistem IoT seperti sistem parkir otomatis.

*Keywords: Sensor Ultrasonik HC-SR04, ESP32, Simulasi Wokwi, Pengukuran Jarak, Internet of Things (IoT), Mikrokontroler.*

**1. Introduction**

* 1. **Latar belakang**

Simulasi sensor jarak ultrasonik dalam praktikum ini merupakan kelanjutan dari praktikum sebelumnya tentang relay, button, dan LED, yang telah dipelajari sebelumnya mengenai konsep dasar input-output dalam sistem IoT. Salah satu aspek penting dalam sistem IoT adalah kemampuan untuk mengukur jarak secara akurat melalui sensor. Praktikum simulasi sensor jarak ultrasonik HC-SR04 Ultrasonic Distance Sensor dengan ESP32 ini dilakukan untuk memahami konsep dasar pengukuran jarak. Dengan melakukan simulasi ini, diharapkan mendapatkan pemahaman praktis tentang bagaimana data dari sensor ultrasonik dapat diambil, diproses, dan dimanfaatkan oleh mikrokontroler ESP32 untuk berbagai aplikasi IoT.

* 1. **Tujuan eksperimen**

1. Mempelajari cara mengimplementasikan rangkaian sensor ultrasonik dengan mikrokontroler ESP32 menggunakan platform simulasi Wokwi
2. Mengimplementasikan kode program untuk mengukur jarak menggunakan sensor ultrasonic
3. Menganalisis hasil pengukuran jarak yang diperoleh dari simulasi sensor

**2. Methodology (Metodologi)**

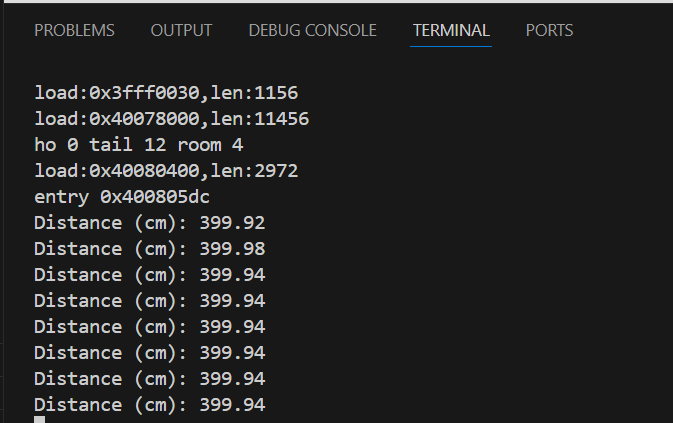
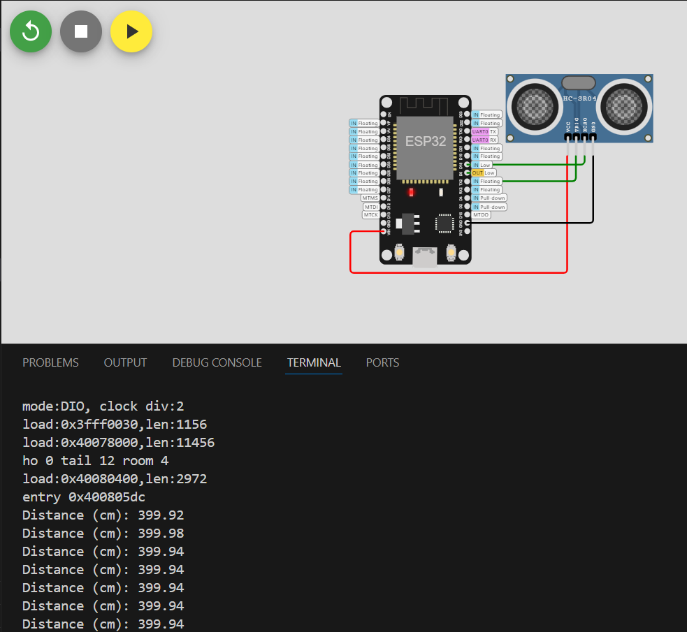
**2.1 Tools & Materials (Alat dan Bahan)**

1. Platform simulasi Wokwi
2. Mikrokontroler ESP32
3. Sensor jarak ultrasonik HC-SR04
4. Kabel jumper (warna merah, hijau, dan hitam)
5. Software: VS Code
   1. **Implementation Steps (Langkah Implementasi)**
6. Membuka platform simulasi Wokwi dan membuat project baru dengan mikrokontroler ESP32
7. Menambahkan sensor ultrasonik HC-SR04 Ultrasonic Distance Sensor disamping ESP32
8. Menghubungkan sensor ultrasonik dengan ESP32: Pin VCC sensor ke pin 5V ESP32 (kabel merah), Pin GND sensor ke pin GND ESP32 (kabel hitam), dst.
9. Membuat file kode program pada main.cpp dalam VS Code dan wokwi untuk mengontrol sensor ultrasonik juga menyalin kode diagram json yang ada di wokwi ke dalam VSCode
10. Mengintegrasikan kode VS Code dengan simulator Wokwi dengan melakukan build kode untuk menghasilkan file firmware dan elf, kemudian menyalin path relatif file firmware dan elf ke file wokwi.toml
11. Menjalankan simulator dengan “request new license” kemudian lanjut perintah "wokwi: start simulator" pada VS Code
12. Menganalisis hasil eksperimen dengan mengambil screenshot rangkaian dan output dalam simulasi sensor

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

* 1. **Experimental Results (Hasil Eksperimen)**

Dari hasil simulasi yang sudah dijalankan, dengan mikrokontroler ESP32 melalui wokwi rangkaian tersebut sudah berhasil dilakukan karena terlihaat pada output terminal yang menunjukkan proses inisialisasi entry data. Sensor ultrasonik tersebut menunjukkan output jarak yang dideteksi menggunakan satuan sentimeter (cm). berikut hasilnya :



Pembahasan :

Dari screenshoot hasil pengukuran diatas menunjukkan bahwa nilai dimulai antara 399.92 – 399.98 cm dan selanjutnya nilai 399.94 cm merupakan nilai mayoritas yang dijangkau oleh sensor.

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

Kode program main.cpp yang dgunakan :

#include <Arduino.h>

const int trigPin = 5;

const int echoPin = 18;

//define sound speed in cm/uS

#define SOUND\_SPEED 0.034

#define CM\_TO\_INCH 0.393701

long duration;

float distanceCm;

float distanceInch;

void setup() {

 Serial.begin(115200); // Starts the serial communication

 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

 pinMode(echoPin, INPUT); // Sets the echoPin as an Input

}

void loop() {

 // Clears the trigPin

 digitalWrite(trigPin, LOW);

 delayMicroseconds(2);

 // Sets the trigPin on HIGH state for 10 micro seconds

 digitalWrite(trigPin, HIGH);

 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds

 duration = pulseIn(echoPin, HIGH);

  // Calculate the distance

 distanceCm = duration \* SOUND\_SPEED/2;

  // Convert to inches

 distanceInch = distanceCm \* CM\_TO\_INCH;

  // Prints the distance in the Serial Monitor

 Serial.print("Distance (cm): ");

 Serial.println(distanceCm);

 // Serial.print("Distance (inch): ");

 // Serial.println(distanceInch);

  delay(1000);

}

Kode diagram.json :

{

    "version": 1,

    "author": "Uri Shaked",

    "editor": "wokwi",

    "parts": [

      { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 14.3, "left": -33.8, "attrs": {} },

      { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -8.1, "left": 111.1, "attrs": {} }

    ],

    "connections": [

      [ "esp:TX0", "$serialMonitor:RX", "", [] ],

      [ "esp:RX0", "$serialMonitor:TX", "", [] ],

      [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v134.4", "h-249.6", "v-67.2" ] ],

      [ "ultrasonic1:TRIG", "esp:D5", "green", [ "v28.8", "h-115.6", "v-28.8" ] ],

      [ "ultrasonic1:ECHO", "esp:D18", "green", [ "v0" ] ],

      [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]

    ],

    "dependencies": {}

  }

**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)**

**Fakultas Vokasi, Universitas Brawijaya**

**Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok**

*Avrilla Agnesya Meifilistiara*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*avrila.agnesya@gmail.com*](mailto:avrila.agnesya@gmail.com)

**Abstract**

Eksperimen ini mengimplementasikan pengembangan API untuk sistem IoT menggunakan framework Laravel 11 dan Ngrok sebagai tunneling service. Dengan memanfaatkan PHP 8.2, XAMPP, dan database MySQL, sebuah backend API dirancang untuk menerima dan menyimpan data dari perangkat IoT. Metodologi penelitian meliputi pembuatan project Laravel, konfigurasi database, implementasi model TransaksiSensor, migrasi tabel, dan pengembangan endpoint API. Hasil utama menunjukkan keberhasilan implementasi API dengan kemampuan operasi GET untuk mengambil data dan POST untuk menambahkan data baru, baik melalui localhost maupun URL publik Ngrok. Pengujian dengan Postman memverifikasi bahwa API berfungsi dengan baik melalui kedua metode akses tersebut, dengan respons JSON status 200 dan perubahan database yang sesuai. Eksperimen ini membuktikan bahwa kombinasi Laravel dan Ngrok menyediakan solusi efektif untuk pengembangan backend IoT dengan kemampuan komunikasi data melalui internet publik, memungkinkan perangkat IoT untuk mengirim dan mengakses data dari server backend secara remote.

*Keywords: Laravel 11, Ngrok, Internet of Things (IoT), Database MySQL, Postman*

**1. Introduction**

* 1. **Latar belakang**

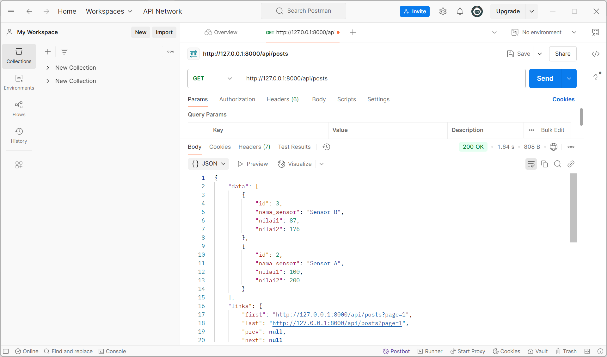
Perangkat IoT seperti sensor, mikrokontroler, dan aktuator membutuhkan mekanisme komunikasi yang efisien dan aman untuk mengirimkan data yang dikumpulkan ke server pusat, serta menerima instruksi untuk menjalankan fungsinya. Beberapa perangkat yang menghubungka IoT dengan praktikpengiriman data menggunakan Laravel 11 dan Ngrok untuk memeberikan kemudahan bagaimana implementasi sebuah perangkat melalui API.

* 1. **Tujuan eksperimen**

1. Mengimplementasikan framework Laravel 11 untuk membangun backend API
2. Melakukan pengujian API menggunakan Postman dan Ngrok
3. Membuat project Laravel 11 dan mengonfigurasi database phpmyadmin

**2. Methodology (Metodologi)**

* 1. **Tools & Materials (Alat dan Bahan)**

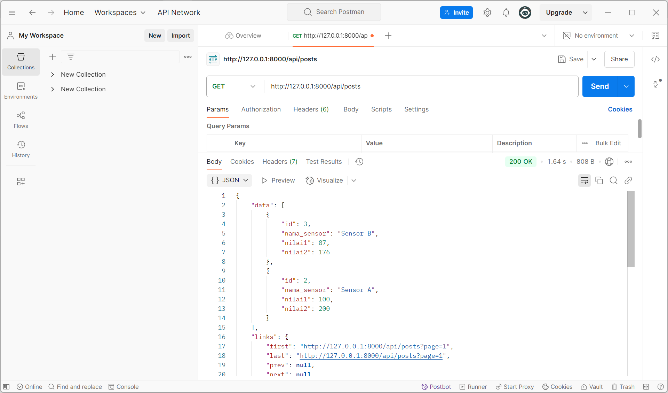
1. PHP 8.2 dan Composer (package manager PHP)
2. Laravel 11 Framework
3. Visual Studio Code
4. Postman untuk testing API dan Ngrok
5. XAMPP untuk server lokal
6. Git untuk version control
   1. **Implementation Steps (Langkah Implementasi)**
7. Download package yang dibutuhkan dalam implementasi praktik, termasuk Laravel 11
8. Membuat project Laravel melalui vscode dengan create-project
9. Membuat database iot\_25 di phpMyAdmin
10. Masuk ke folder project VScode iot\_25 pada bagian env. untuk mengganti database\_username sesuai dengan phpMyAdmin
11. Buat file model melalui terminal dengan menjalankan perintah **php artisan make:model TransaksiSensor -m**
12. Ganti kode program sesuai dengan perintah, setelah semuanya fix, maka dapat menjalankan migration table : **php artisan migrate**
13. Setelah semua file di ubah dan di migrate, Buat API controller dengan menjalankan perintah : **php artisan make:controller Api/TransaksiSensorController** di terminal
14. Setelah mengganti beberapa kode program file controller dan routes API, lanjut melakukan testing menggunakan tools postman
15. Jika belum mendownload, disarankan untuk mengunduh dan menginstall terlebih dahulu
16. Setelah instalasi, dapat melakukan percobaan akses API, dengan mengaktifkan XAMPP
17. Setelah itu, jalankan Laravel dengan menggunakan perintah **php artisan serve**
18. Karena sebelumnya database pada project sudah terhubung dengan phpMyAdmin melalui env, maka otomatis data akan masuk. Sebagai contohnya ada 2 data yg bisa dimasukkan secara manual pada phpMyAdmin
19. Untuk mengecek/ mengambil data tersebut, dapat melalui aplikasi postman
20. Masuk pada aplikasi dan tambahkan alamat yang akan dituju, melalui server berikut:  **http://127.0.0.1:8000/api/posts**
21. Metode “GET” disini diperuntukkan mengambil data yang sudah dimasukkan manual sebelumnya pada phpMyAdmin
22. Setelah itu, klik “send” untuk mendapatkan tampilan sebagai berikut:  
    jika hasil output seperti diatas, maka artinya data tersebut berhasil diambil.
23. Selanjutnya untuk mencoba metode “POST” tidak berbeda dengan metode “GET”, hanya saja mengganti metode dan bagian header menjadi “content-type” dan Value: “application/json”
24. Bagian body, diisi dengan data yang ingin ditambahkan.
25. Kemudian klik send. Pastikan data berhasil di-insert dengan mengecek di phpMyAdmin, yang sebelumnya hanya ada 2 data, akan menjadi 3 data jika metode post tersebut berhasil ditambahkan
26. Langkah selanjutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot
27. Lakukan download dan login akun Ngrok terlebih dahulu
28. Setelah itu, ekstrasi folder tersebut, dan buka melalui terminal
29. Jalankan ngrok melalui terminal menggunakan link yang sudah ada pada halaman ngrok
30. Setelah disalin, maka terminal akan memberikan URL untuk mengakses melalui internet
31. URL tersebut dapat di salin sebagai Alamat yg akan dikirim melalui postman
32. Dengan mengirim metode “GET” dan “POST” sama saja dengan cara sebelumnya, hanya saja yang membedakan di bagian URL antara Laravel dan Ngrok.

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

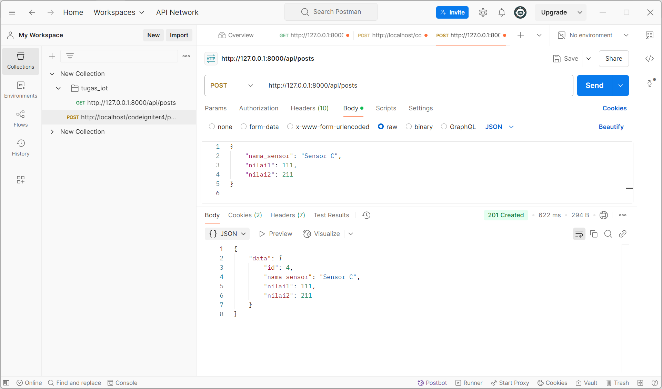
**3.1 Experimental Results (Hasil Eksperimen)**

Hasil dari eksperimen tersebut berhasil sesuai dengan cara dan urutannya. Mulai dari pembuatan database menggunakan Laravel, kemudian cara migrate seluruh table yang ada, dan cara menjalankan pengiriman dan pengambilan data pada phpMyAdmin melalui postman. Penggunaan Ngrok sebagai tunneling service berhasil mengekspos API Laravel yang berjalan di localhost ke internet publik. Pengujian API dengan metode POST ke endpoint yang sama berhasil menambahkan data baru ke database. Request body dalam format JSON berhasil diterima dan data disimpan ke database. Setelah pengujian, jumlah data di database bertambah dari 2 menjadi 3 record, dan seterusnya. Sehingga didapatkan hasil sebagai berikut:

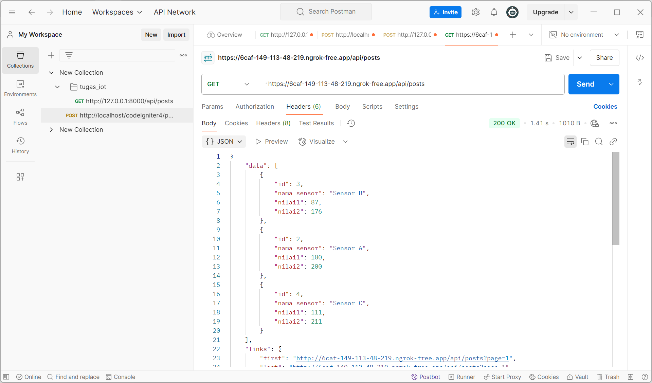
Menggunakan url laravel dengan metode GET:

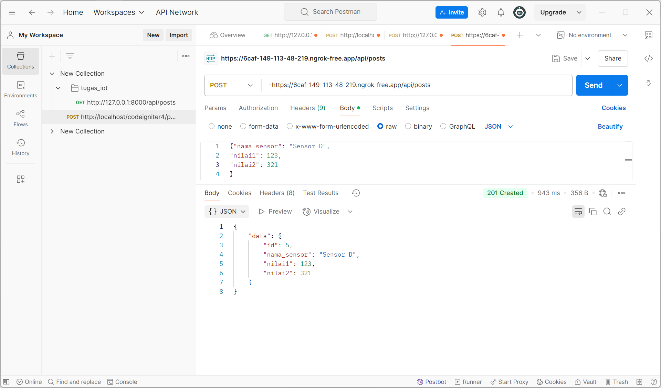


Menggunakan metode POST:

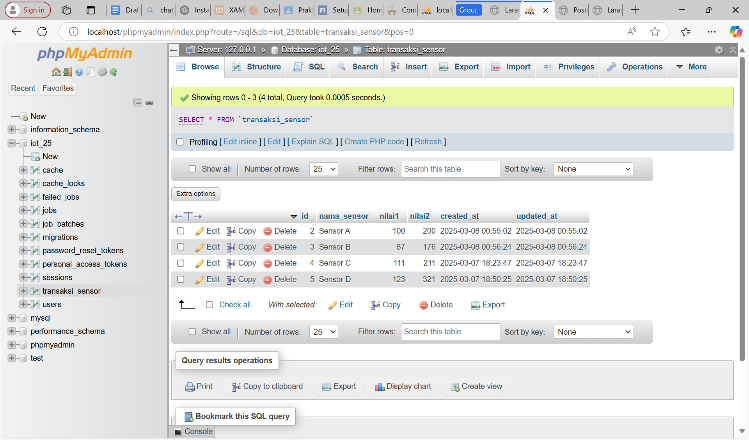


Menggunakan url Ngrok dengan metode GET:

  
Menggunakan url Ngrok dengan metode POST:



Hasil implementasi metode GET dan POST melalui postman yang bisa dicek melalui database:

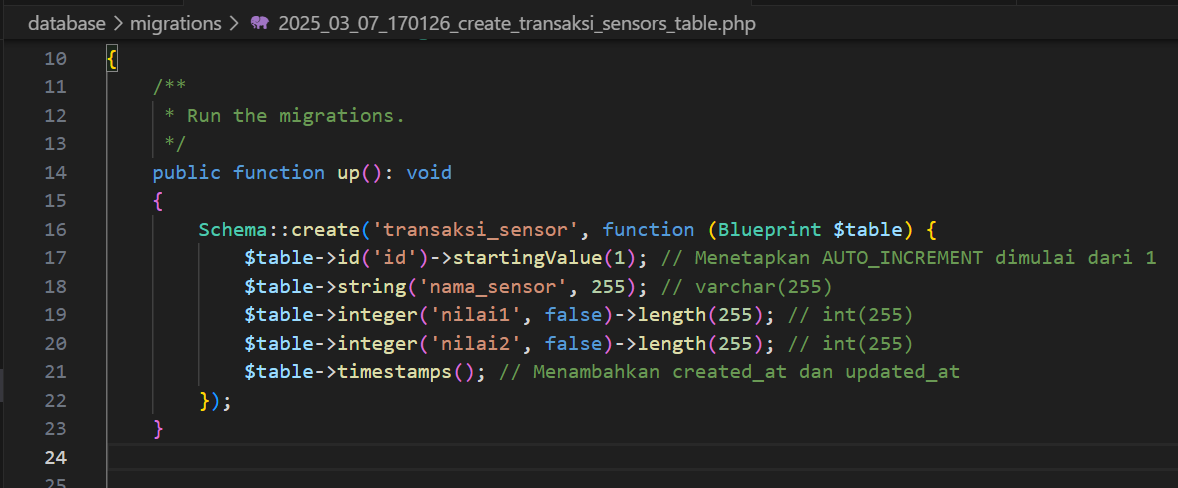


Pembahasan:

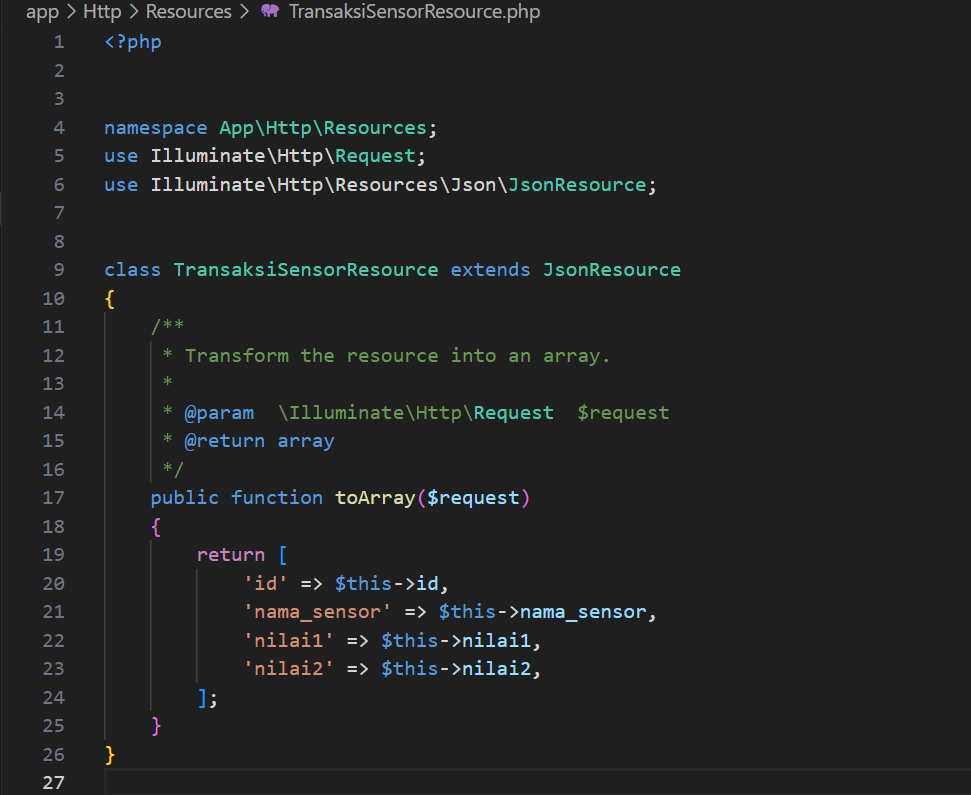
API berhasil menampilkan data sensor yang sebelumnya telah dimasukkan ke database dalam format JSON dengan respons status 200. Hal ini membuktikan bahwa konfigurasi routes dan controller untuk mengambil data berfungsi dengan benar. Hal yang sama terjadi saat menggunakan URL Ngrok untuk metode GET, dimana API hanya mengambil data tanpa memodifikasi database. Sedangkan pengujian metode POST ke endpoint yang sama dengan mengirimkan body JSON yang berisi data baru menghasilkan perubahan pada database. Setelah mengirim request POST, jumlah data di database akan berubah atau bertambah.

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

Kode migration **create\_transaksi\_sensors\_table.php**



Kode program **TransaksiSensorResource.php**



Kode program **controller Api/TransaksiSensorController.php**

<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;

use App\Models\TransaksiSensor;

use App\Http\Controllers\Controller;

use App\Http\Resources\TransaksiSensorResource;

class TransaksiSensorController extends Controller

{

    /\*\*

     \* index

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        // Get all transactions from TransaksiSensor model, paginated

        $transaksiSensors = TransaksiSensor::latest()->paginate(5);

        // Return a collection of transactions as a resource

        return TransaksiSensorResource::collection($transaksiSensors);

    }

    /\*\*

 \* Store a newly created resource in storage.

 \*

 \* @param  \Illuminate\Http\Request  $request

 \* @return \Illuminate\Http\Response

 \*/

public function store(Request $request)

{

    $validatedData = $request->validate([

        'nama\_sensor' => 'required|string|max:255',

        'nilai1' => 'required|integer',

        'nilai2' => 'required|integer',

    ]);

    $transaksiSensor = TransaksiSensor::create($validatedData);

    return new TransaksiSensorResource($transaksiSensor);

}

/\*\*

 \* Display the specified resource.

 \*

 \* @param  int  $id

 \* @return \Illuminate\Http\Response

 \*/

public function show($id)

{

    $transaksiSensor = TransaksiSensor::findOrFail($id);

    return new TransaksiSensorResource($transaksiSensor);

}

/\*\*

 \* Update the specified resource in storage.

 \*

 \* @param  \Illuminate\Http\Request  $request

 \* @param  int  $id

 \* @return \Illuminate\Http\Response

 \*/

public function update(Request $request, $id)

{

    $validatedData = $request->validate([

        'nama\_sensor' => 'required|string|max:255',

        'nilai1' => 'required|integer',

        'nilai2' => 'required|integer',

    ]);

    $transaksiSensor = TransaksiSensor::findOrFail($id);

    $transaksiSensor->update($validatedData);

    return new TransaksiSensorResource($transaksiSensor);

}

/\*\*

 \* Remove the specified resource from storage.

 \*

 \* @param  int  $id

 \* @return \Illuminate\Http\Response

 \*/

public function destroy($id)

{

    $transaksiSensor = TransaksiSensor::findOrFail($id);

    $transaksiSensor->delete();

    return response()->json(['message' => 'Deleted successfully'], 204);

}

}

Kode Program **routes/api.php**

