

### **Tugas 3**

## **Laporan Praktik Simulasi Relay, Button & LED, Simulasi Jarak dan Pembuatan API Menggunakan Laravel 11 dan Ngrok**



Nama : Bintang Putra Nala Saki  
Kelas : T4C  
NIM 233140700111077

**Fakultas Vokasi**  
**Universitas Brawijaya**  
**Email :bintangskrafti867@gmail.com**

## **ABSTRAK**

Laporan ini membahas praktik simulasi dalam penerapan Internet of Things (IoT), yang mencakup tiga aspek utama: simulasi relay, button, dan LED; simulasi jarak; serta pembuatan API menggunakan Laravel 11 dan Ngrok. Dalam era digital saat ini, teknologi IoT semakin berkembang dan memberikan manfaat yang signifikan di berbagai sektor, seperti industri, kesehatan, dan otomasi rumah. Salah satu faktor utama dalam penerapan IoT adalah kemampuan untuk mengontrol perangkat elektronik secara otomatis dan menghubungkan sistem dengan jaringan berbasis cloud. Oleh karena itu, laporan ini bertujuan untuk memahami prinsip kerja relay dalam mengontrol perangkat elektronik melalui button dan LED, mengukur jarak menggunakan sensor ultrasonik, serta membangun API berbasis Laravel yang dapat diakses secara global melalui Ngrok.

Metode yang digunakan dalam penelitian ini adalah eksperimen langsung dengan melakukan simulasi terhadap masing-masing komponen dan menguji fungsionalitasnya. Hasil dari eksperimen menunjukkan bahwa penggunaan relay, button, dan LED dapat meningkatkan efisiensi dalam pengendalian perangkat elektronik, sensor ultrasonik memiliki akurasi yang cukup tinggi dalam pengukuran jarak, serta API yang dikembangkan dengan Laravel dan Ngrok memungkinkan sistem IoT untuk berkomunikasi secara lebih fleksibel dan luas. Dengan adanya laporan ini, diharapkan dapat memberikan wawasan yang lebih mendalam mengenai konsep dan implementasi IoT bagi pengembang serta mahasiswa yang tertarik dalam bidang ini.

## **BAB 1: PENDAHULUAN**

### **1.1 Latar Belakang**

Perkembangan teknologi di era digital telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia, salah satunya adalah teknologi Internet of Things (IoT). IoT merupakan konsep yang memungkinkan perangkat elektronik saling terhubung dan berkomunikasi melalui jaringan internet, sehingga dapat dikendalikan dan dimonitor secara real-time. Penerapan IoT saat ini telah banyak digunakan dalam bidang otomasi rumah, industri, kesehatan, dan transportasi untuk meningkatkan efisiensi serta mempermudah pekerjaan manusia.

Dalam praktik implementasi IoT, terdapat beberapa aspek penting yang perlu diperhatikan, seperti pengendalian perangkat keras menggunakan relay, pemanfaatan sensor untuk mendapatkan data lingkungan, serta komunikasi antara perangkat dengan server berbasis cloud melalui API. Oleh karena itu, laporan ini membahas tiga praktik utama dalam implementasi IoT, yaitu simulasi relay, button, dan LED untuk mengontrol perangkat elektronik; simulasi jarak menggunakan sensor ultrasonik untuk pengukuran data lingkungan; serta pembuatan API menggunakan Laravel 11 dan Ngrok untuk mendukung komunikasi antar perangkat.

Pemahaman terhadap prinsip kerja relay dan button sangat penting karena kedua komponen ini banyak digunakan dalam sistem otomasi, seperti pengendalian lampu, kipas, dan perangkat lainnya. Selain itu, pengukuran jarak dengan sensor ultrasonik dapat diterapkan dalam berbagai aplikasi, seperti kendaraan pintar, sistem parkir otomatis, dan robotika. Sedangkan API berperan sebagai jembatan komunikasi antara perangkat IoT dengan aplikasi berbasis web, sehingga memungkinkan pengolahan dan pengelolaan data secara lebih efisien.

Dengan adanya penelitian ini, diharapkan mahasiswa dan pengembang teknologi dapat memahami konsep dasar serta penerapan IoT dalam kehidupan sehari-hari. Melalui

eksperimen yang dilakukan, laporan ini juga memberikan wawasan praktis mengenai cara kerja dan integrasi perangkat IoT untuk mendukung inovasi teknologi di masa depan.

## **BAB 2: TUJUAN**

### **2.1 Tujuan Simulasi Relay, Button & LED**

- Memahami prinsip kerja relay dalam mengontrol perangkat listrik.
- Mengetahui cara kerja button sebagai pemicu aksi.
- Menggunakan LED sebagai indikator status relay.

### **2.2 Tujuan Simulasi Jarak**

- Memahami cara kerja sensor jarak ultrasonik.
- Menampilkan hasil pengukuran jarak dalam satuan cm atau m.
- Mengintegrasikan sensor dengan microcontroller.

### **2.3 Tujuan Pembuatan API Menggunakan Laravel 11 dan Ngrok**

- Mempelajari dasar pembuatan API menggunakan Laravel 11.
- Menggunakan Ngrok untuk membuat API dapat diakses secara publik.
- Menghubungkan API dengan perangkat IoT.

## **BAB 3: HASIL DAN PEMBAHASAN**

### **3.1 Hasil dan Pembahasan Simulasi Relay, Button & LED**

Simulasi ini menggunakan microcontroller untuk mengontrol relay yang terhubung dengan LED dan button. Saat button ditekan, relay akan aktif dan menyalakan LED. Implementasi ini menunjukkan bagaimana perangkat elektronik dapat dikontrol melalui sistem digital secara efisien.

### **3.2 Hasil dan Pembahasan Simulasi Jarak**

Sensor ultrasonik digunakan untuk mengukur jarak dengan prinsip pemantulan gelombang suara. Hasil pengukuran ditampilkan dalam layar serial monitor. Dari hasil uji coba, ditemukan bahwa sensor memiliki akurasi yang cukup tinggi dalam jarak tertentu, meskipun terdapat deviasi kecil akibat faktor lingkungan.

### **3.3 Hasil dan Pembahasan Pembuatan API Menggunakan Laravel 11 dan Ngrok**

Dalam praktik ini, Laravel 11 digunakan untuk membuat API yang dapat menerima dan mengirim data ke perangkat IoT. Ngrok digunakan untuk menjadikan API dapat diakses dari luar jaringan lokal. Implementasi ini memungkinkan komunikasi dua arah antara perangkat IoT dan server berbasis cloud, membuka peluang integrasi lebih lanjut dalam sistem IoT berbasis web.

## **KESIMPULAN**

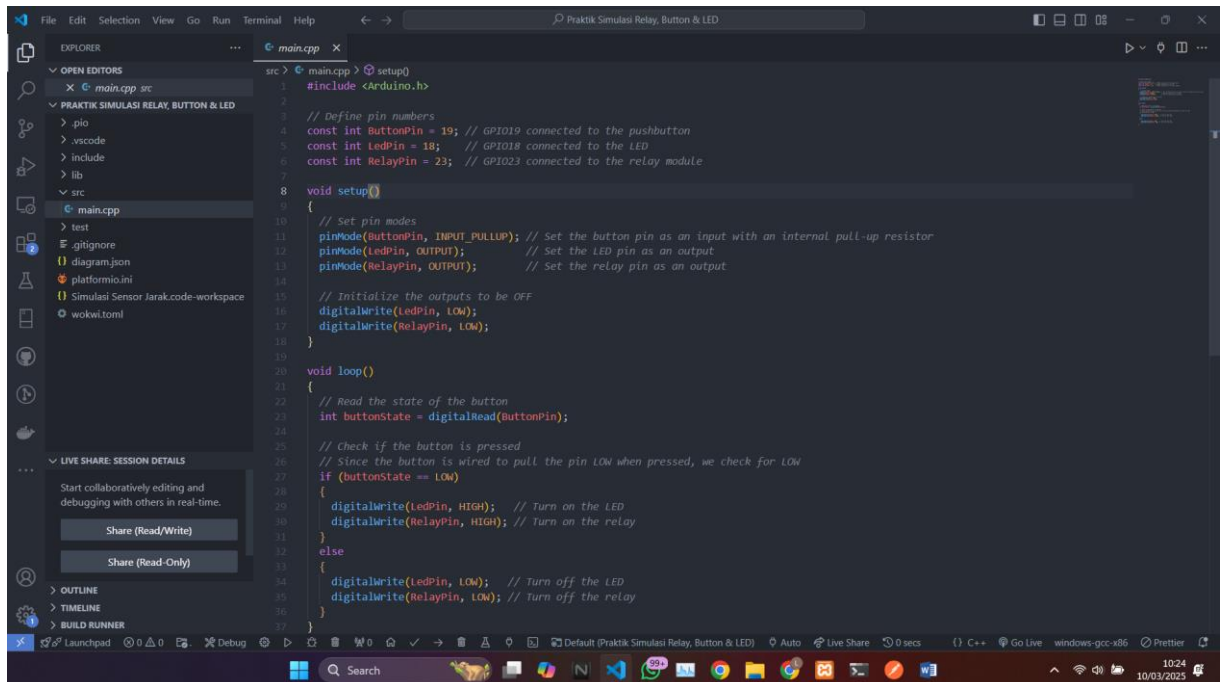
Laporan ini membahas praktik IoT dari berbagai aspek, yaitu simulasi relay dan button, pengukuran jarak menggunakan sensor ultrasonik, serta pembuatan API dengan Laravel 11 dan Ngrok. Hasil praktik menunjukkan bahwa setiap komponen berperan penting

dalam membangun ekosistem IoT yang lebih efisien dan dapat diakses dari mana saja. Dengan memahami dasar-dasar ini, pengembangan IoT dapat lebih dioptimalkan untuk berbagai kebutuhan teknologi masa depan.

## BAB 4 LAMPIRAN & DOKUMENTASI

### 1. Proyek Simulasi Relay, Relay & Button

#### a. Main.cpp



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

```
// Define pin numbers
```

```
const int ButtonPin = 19; // GPIO19 connected to the pushbutton
```

```
const int LedPin = 18; // GPIO18 connected to the LED
```

```
const int RelayPin = 23; // GPIO23 connected to the relay module
```

```
void setup()
```

```
{
```

```
// Set pin modes
```

```
pinMode(ButtonPin, INPUT_PULLUP); // Set the button pin as an input with an  
internal pull-up resistor
```

```
pinMode(LedPin, OUTPUT); // Set the LED pin as an output
```

```
pinMode(RelayPin, OUTPUT); // Set the relay pin as an output
```

```
// Initialize the outputs to be OFF
```

```
digitalWrite(LedPin, LOW);
```

```
digitalWrite(RelayPin, LOW);
```

```
}
```

```
void loop()
```

```
{
```

```
// Read the state of the button
```

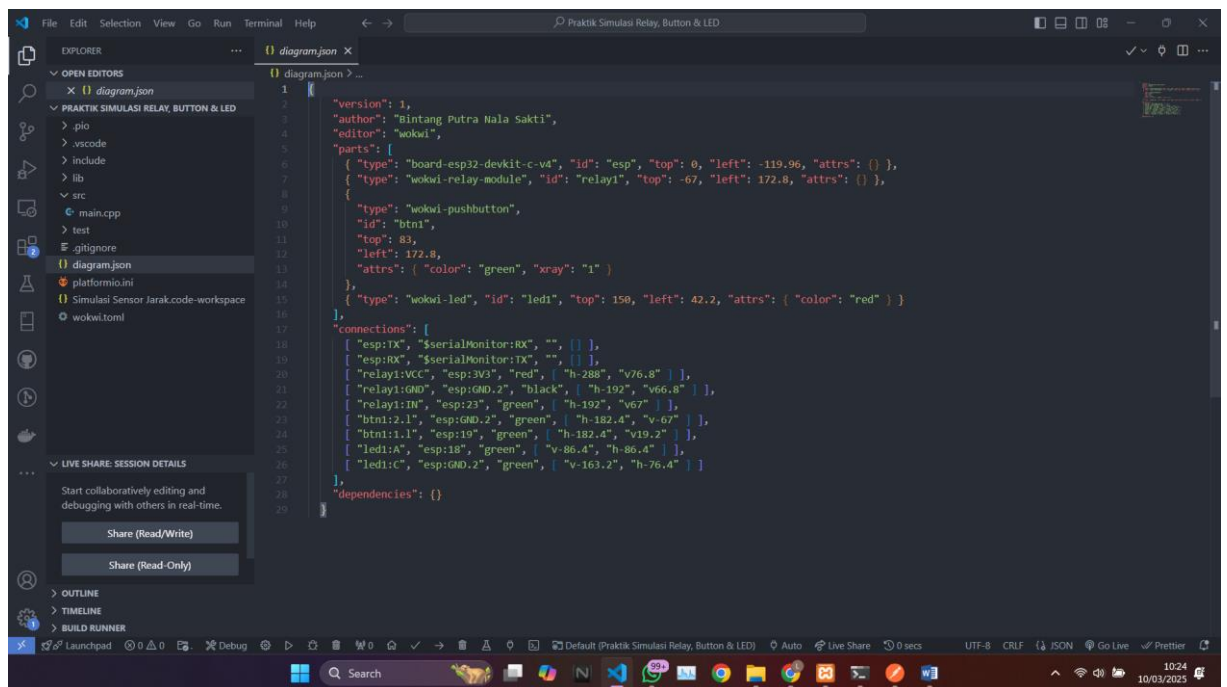
```

int buttonState = digitalRead(ButtonPin);

// Check if the button is pressed
// Since the button is wired to pull the pin LOW when pressed, we check for LOW
if (buttonState == LOW)
{
    digitalWrite(LedPin, HIGH); // Turn on the LED
    digitalWrite(RelayPin, HIGH); // Turn on the relay
}
else
{
    digitalWrite(LedPin, LOW); // Turn off the LED
    digitalWrite(RelayPin, LOW); // Turn off the relay
}
}

```

## b. Diagram json



```

{
  "version": 1,
  "author": "Bintang Putra Nala Sakti",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": -119.96, "attrs":
    { } },
    { "type": "wokwi-relay-module", "id": "relay1", "top": -67, "left": 172.8, "attrs":
    { } },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 83,
      "left": 172.8,
      "attrs": { "color": "green", "xray": "1" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 150, "left": 42.2, "attrs": { "color": "red" }
    }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [ ] ],
    [ "esp:RX", "$serialMonitor:TX", "", [ ] ],
    [ "relay1:VCC", "esp:3V3", "red", [ "h-288", "v76.8" ] ],
    [ "relay1:GND", "esp:GND-2", "black", [ "h-192", "v66.8" ] ],
    [ "relay1:IN", "esp:23", "green", [ "h-192", "v67" ] ],
    [ "btn1:2.1", "esp:GND-2", "green", [ "h-182.4", "v-67" ] ],
    [ "btn1:1.1", "esp:19", "green", [ "h-182.4", "v19.2" ] ],
    [ "led1:A", "esp:18", "green", [ "v-86.4", "h-86.4" ] ],
    [ "led1:C", "esp:GND-2", "green", [ "v-163.2", "h-76.4" ] ]
  ],
  "dependencies": { }
}

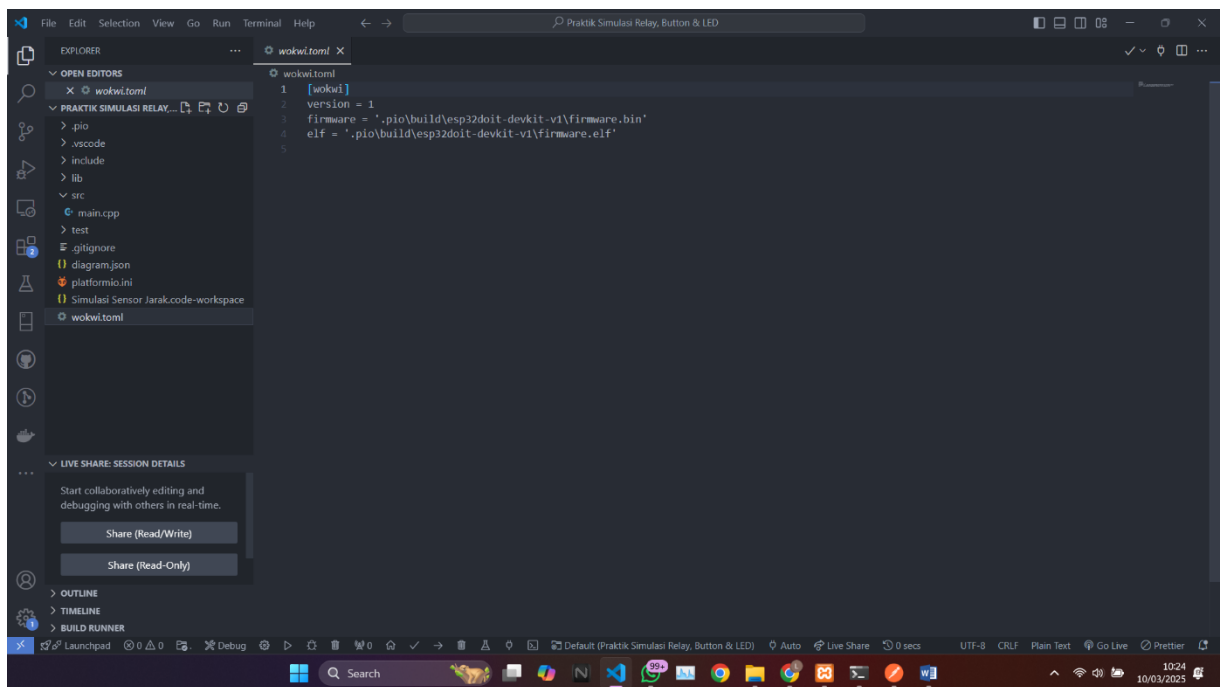
```

```

    { "type": "wokwi-led", "id": "led1", "top": 150, "left": 42.2, "attrs": { "color":
"red" } }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "relay1:VCC", "esp:3V3", "red", [ "h-288", "v76.8" ] ],
    [ "relay1:GND", "esp:GND.2", "black", [ "h-192", "v66.8" ] ],
    [ "relay1:IN", "esp:23", "green", [ "h-192", "v67" ] ],
    [ "btn1:2.1", "esp:GND.2", "green", [ "h-182.4", "v-67" ] ],
    [ "btn1:1.1", "esp:19", "green", [ "h-182.4", "v19.2" ] ],
    [ "led1:A", "esp:18", "green", [ "v-86.4", "h-86.4" ] ],
    [ "led1:C", "esp:GND.2", "green", [ "v-163.2", "h-76.4" ] ]
  ],
  "dependencies": {}
}

```

### c. Wokwi.toml

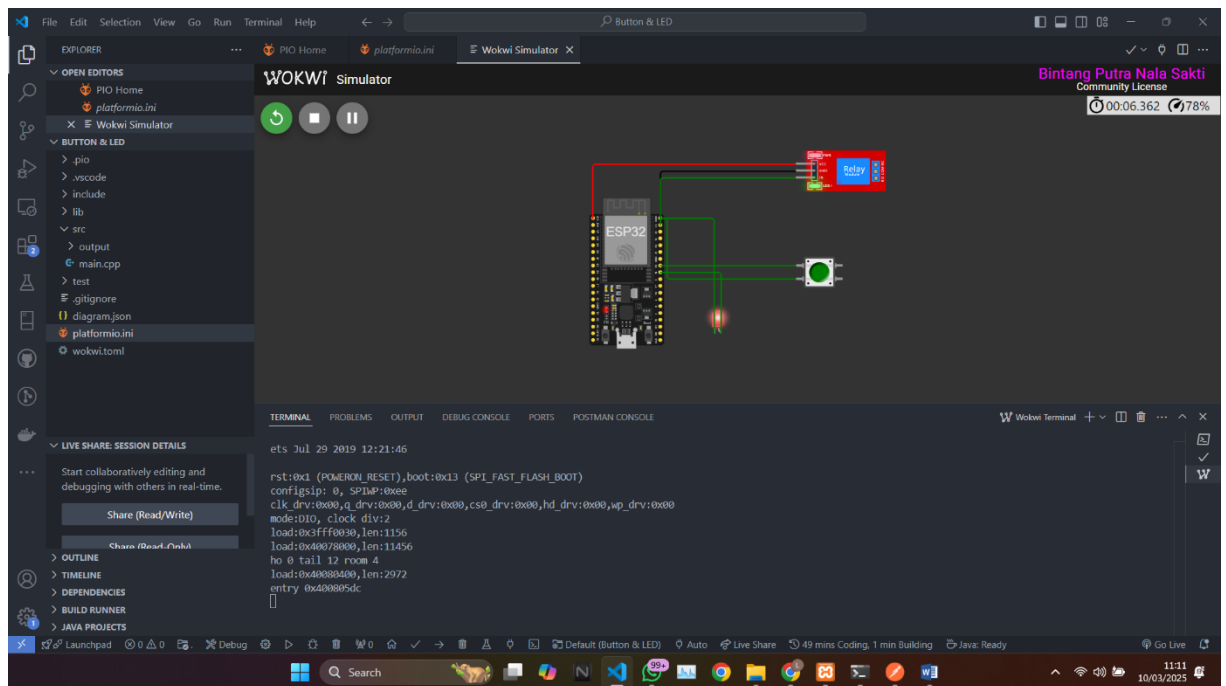


```

[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

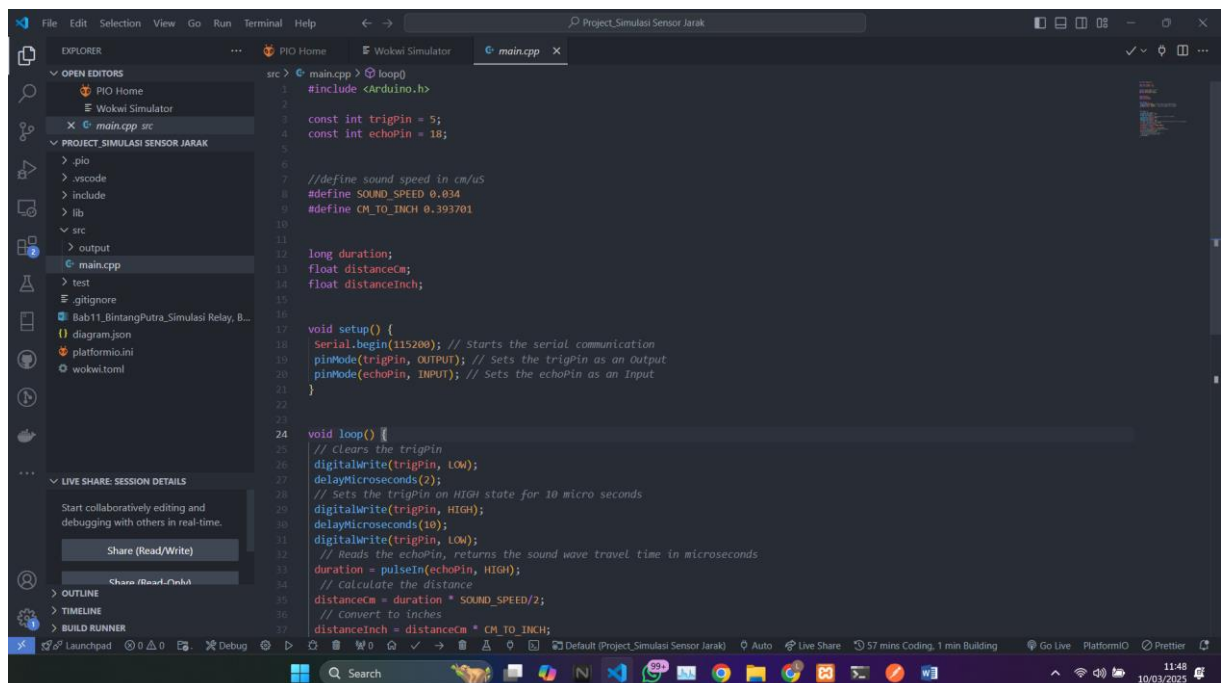
```

## D. Hasil



## 2. Proyek Simulasi Jarak

### a. Main.cpp



```
#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
}

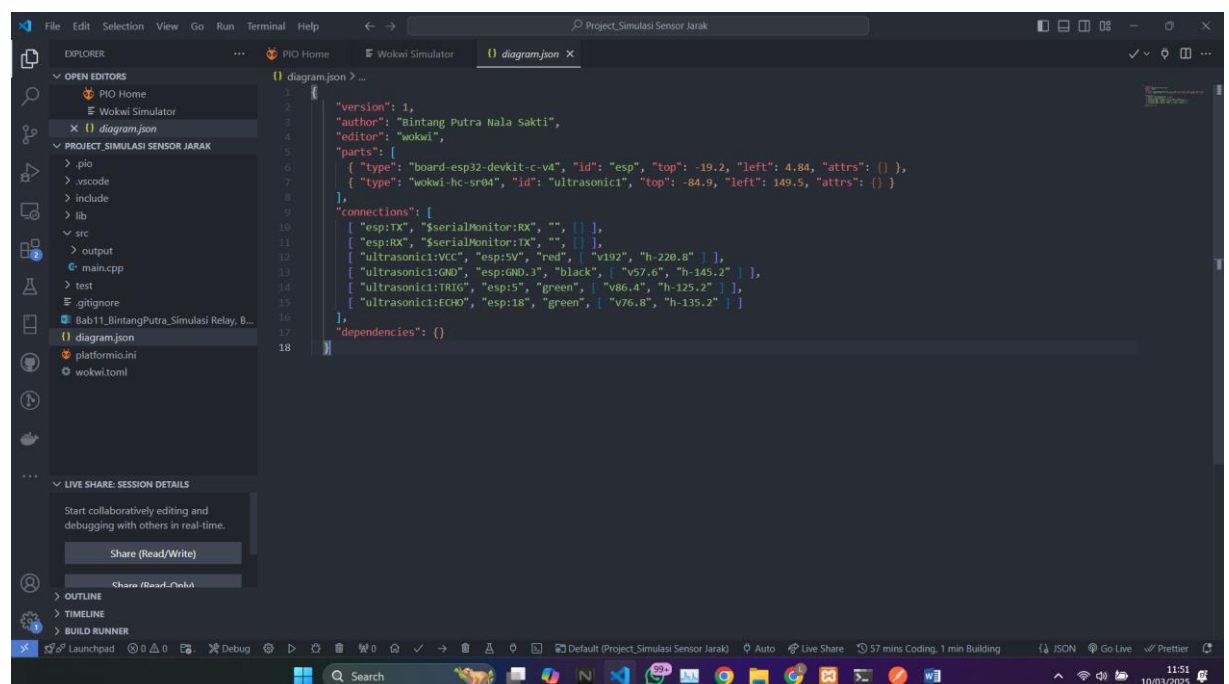
```

```

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);
}

```

## b. Digaram json



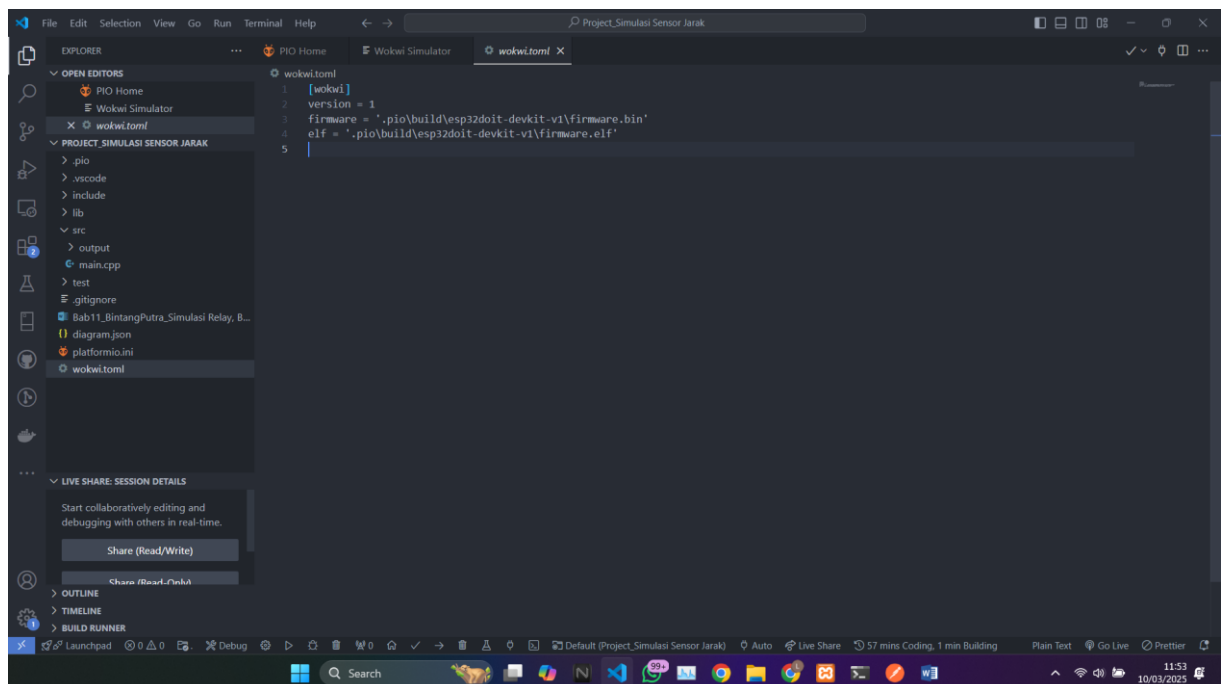


```

{
  "version": 1,
  "author": "Bintang Putra Nala Sakti",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": -19.2, "left": 4.84, "attrs":
  {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -84.9, "left": 149.5, "attrs":
  {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [ ] ],
    [ "esp:RX", "$serialMonitor:TX", "", [ ] ],
    [ "ultrasonic1:VCC", "esp:5V", "red", [ "v192", "h-220.8" ] ],
    [ "ultrasonic1:GND", "esp:GND.3", "black", [ "v57.6", "h-145.2" ] ],
    [ "ultrasonic1:TRIG", "esp:5", "green", [ "v86.4", "h-125.2" ] ],
    [ "ultrasonic1:ECHO", "esp:18", "green", [ "v76.8", "h-135.2" ] ]
  ],
  "dependencies": {}
}

```

### C. Wokwi.toml

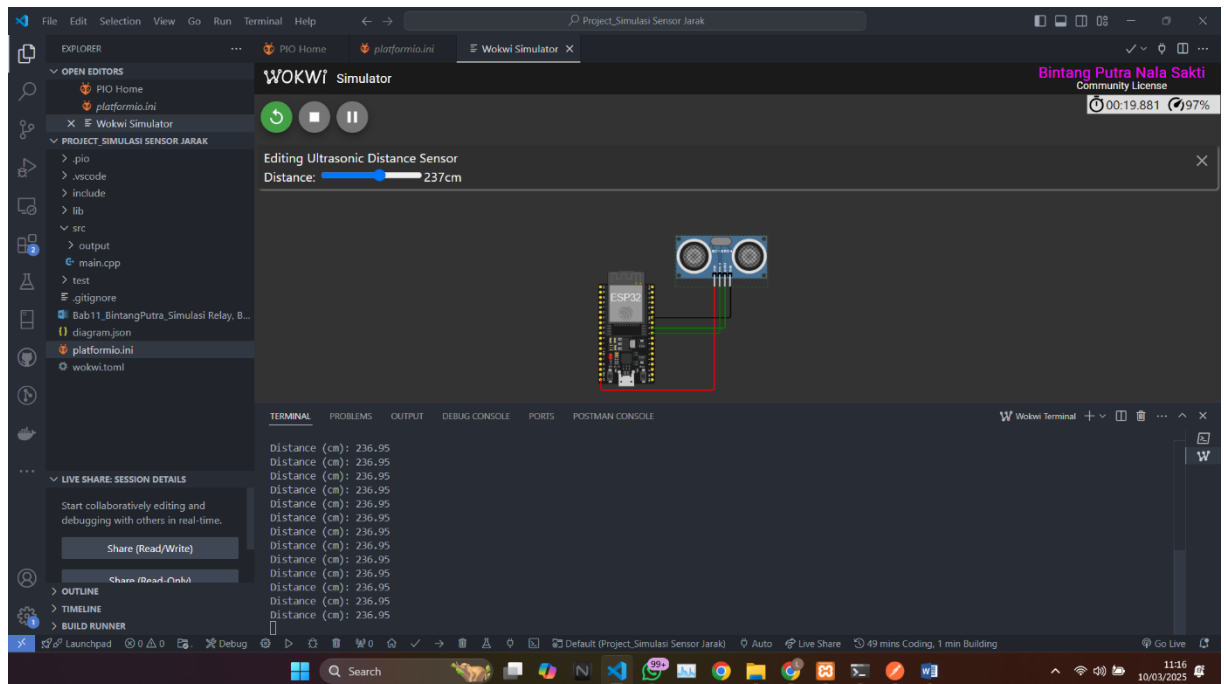


```

[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

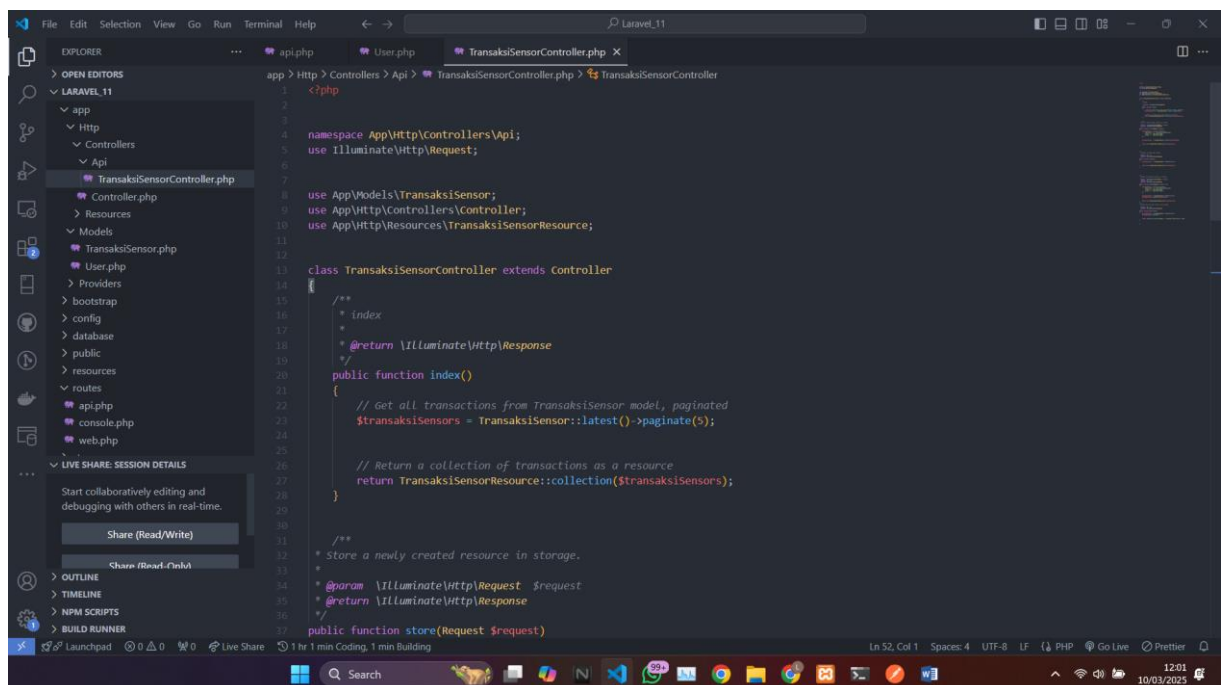
```

## D. Hasil



## 3. Proyek Pembuatan API Menggunakan Laravel 11 dan Ngrok

### a. 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',
    ]);

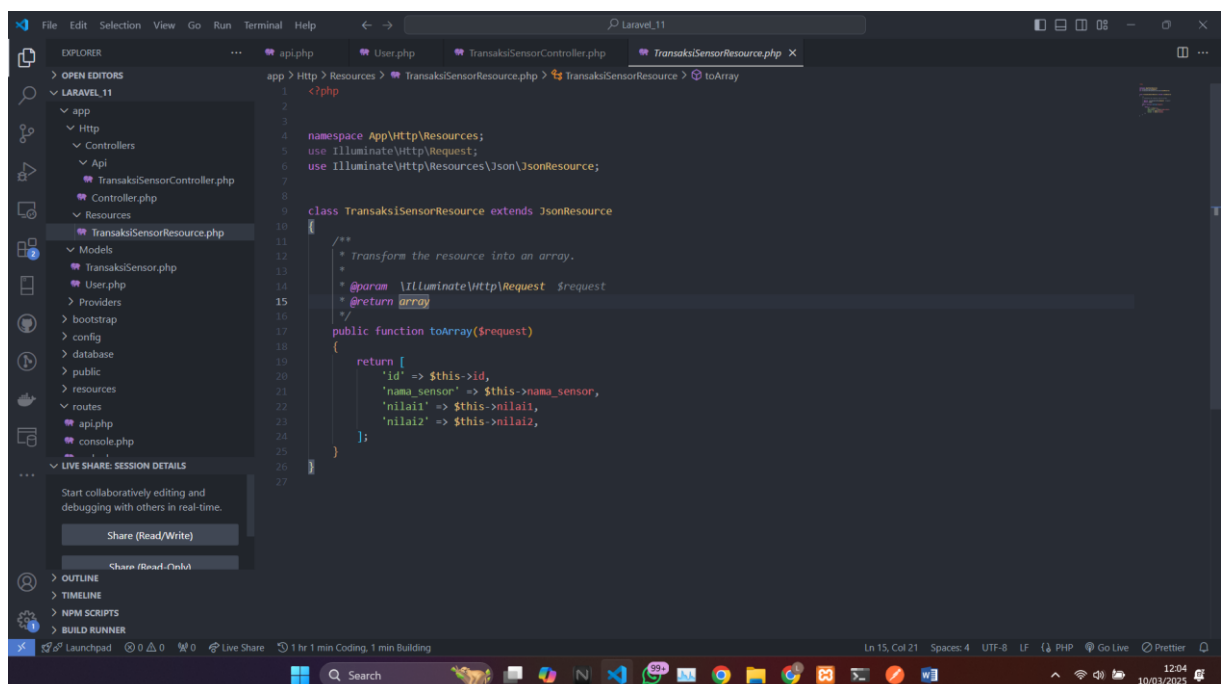
    $transaksiSensor = TransaksiSensor::findOrFail($id);
    $transaksiSensor->update($validatedData);

    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);
}}

```

## b. TransaksiSensorResource.php

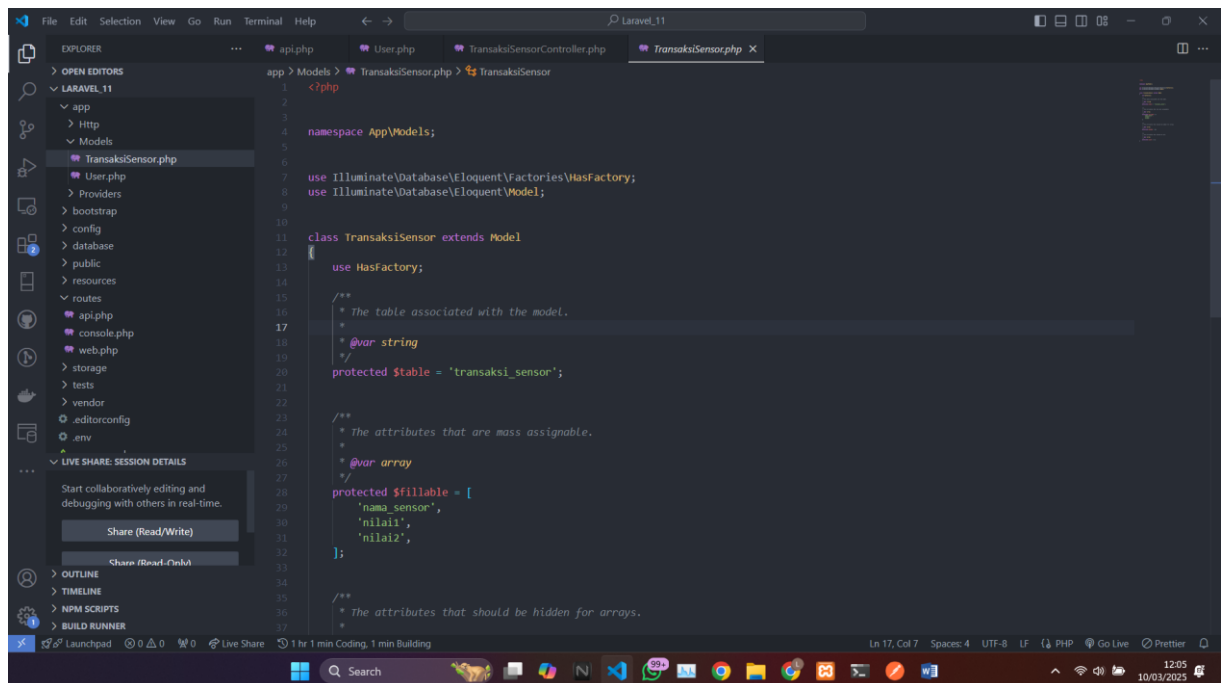


```
<?php
```

```
namespace App\Http\Resources;
use Illuminate\Http\Request;
use Illuminate\Http\Resources\Json\JsonResource;
```

```
class TransaksiSensorResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array
     */
    public function toArray($request)
    {
        return [
            'id' => $this->id,
            'nama_sensor' => $this->nama_sensor,
            'nilai1' => $this->nilai1,
            'nilai2' => $this->nilai2,
        ];
    }
}
```

### c. TransaksiSensor



```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class TransaksiSensor extends Model
{
```

```

use HasFactory;

/**
 * The table associated with the model.
 *
 * @var string
 */
protected $table = 'transaksi_sensor';

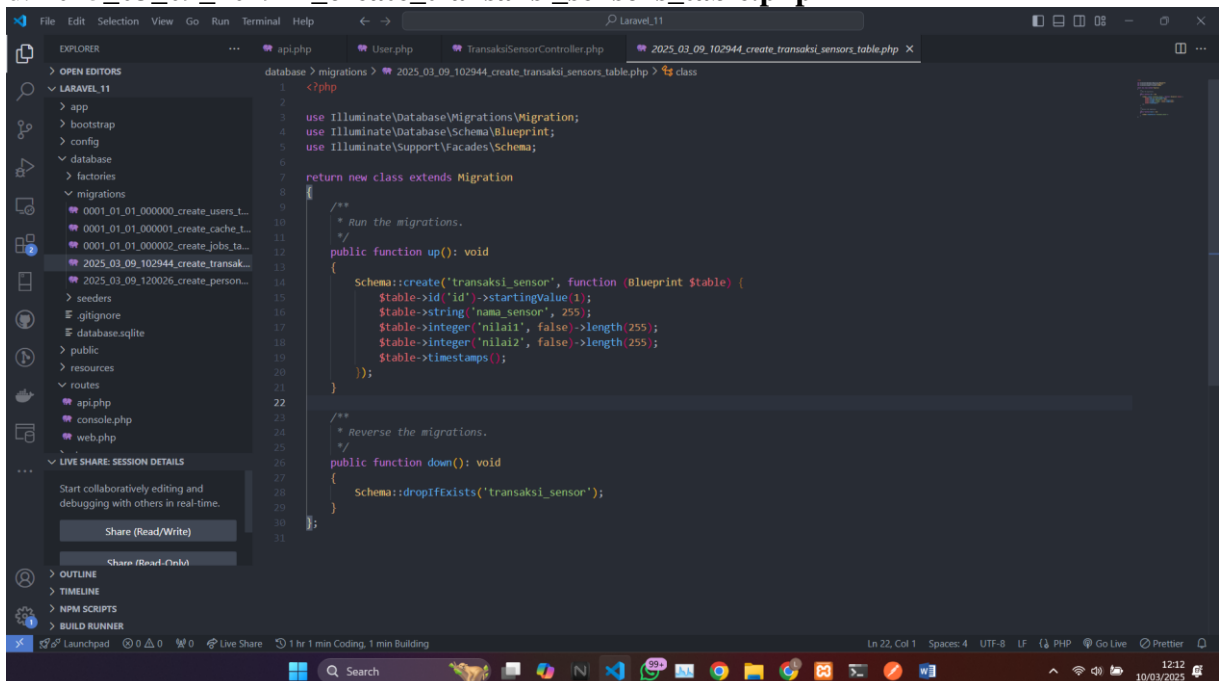
/**
 * The attributes that are mass assignable.
 *
 * @var array
 */
protected $fillable = [
    'nama_sensor',
    'nilai1',
    'nilai2',
];

/**
 * The attributes that should be hidden for arrays.
 *
 * @var array
 */
protected $hidden = [];

/**
 * The attributes that should be cast.
 *
 * @var array
 */
protected $casts = [];
}

```

#### d. 2025\_03\_09\_102944\_create\_transaksi\_sensors\_table.php



```
<?php
```

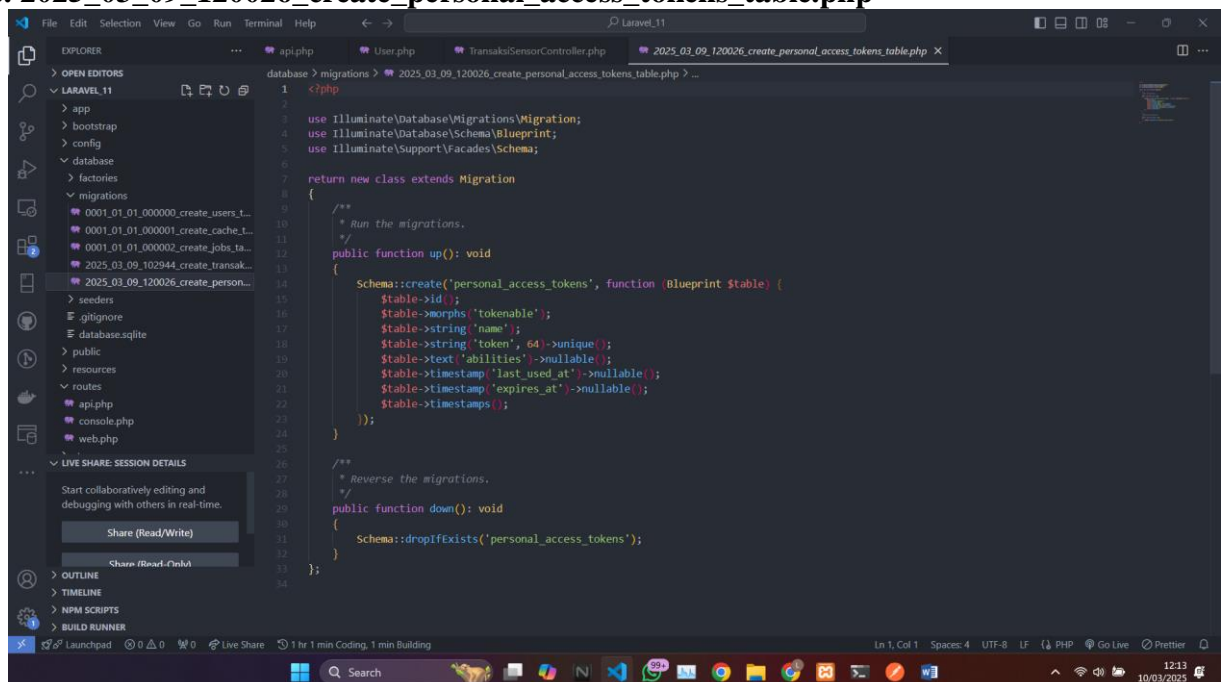
```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('transaksi_sensor', function (Blueprint $table) {
            $table->id('id')->startingValue(1);
            $table->string('nama_sensor', 255);
            $table->integer('nilai1', false)->length(255);
            $table->integer('nilai2', false)->length(255);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('transaksi_sensor');
    }
};
```

#### e. 2025\_03\_09\_120026\_create\_personal\_access\_tokens\_table.php



```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```

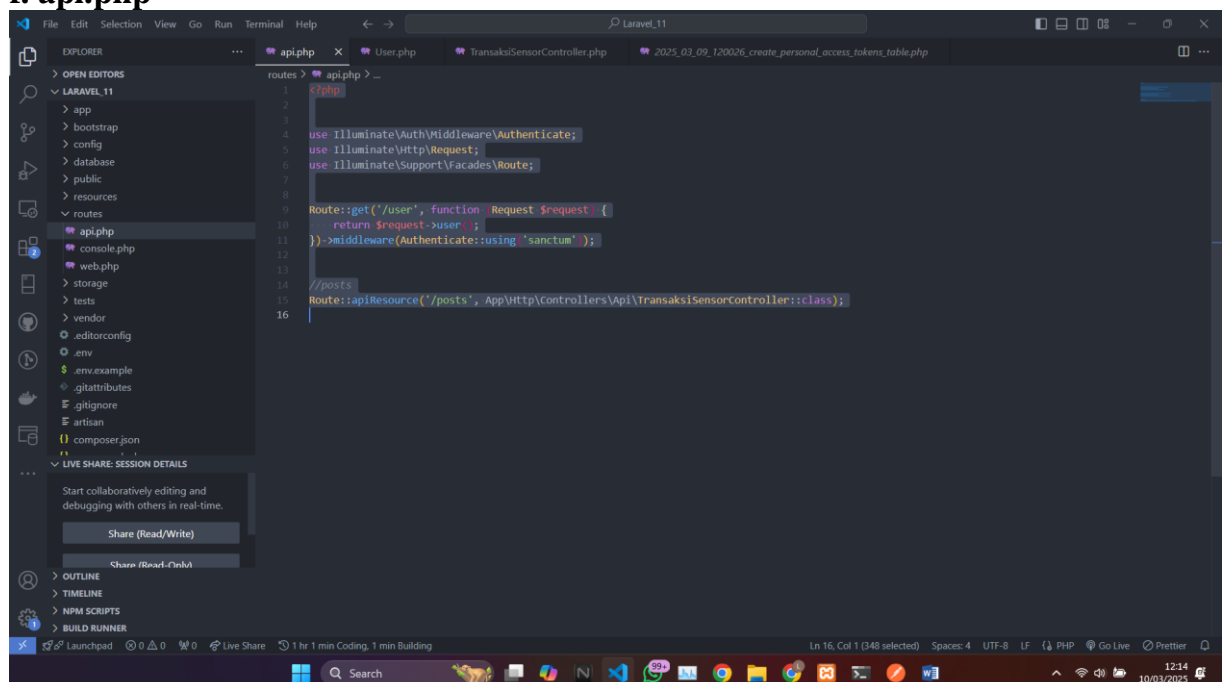
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};

```

## f. api.php





```
<?php
```

```
use Illuminate\Auth\Middleware\Authenticate;
```

```
use Illuminate\Http\Request;
```

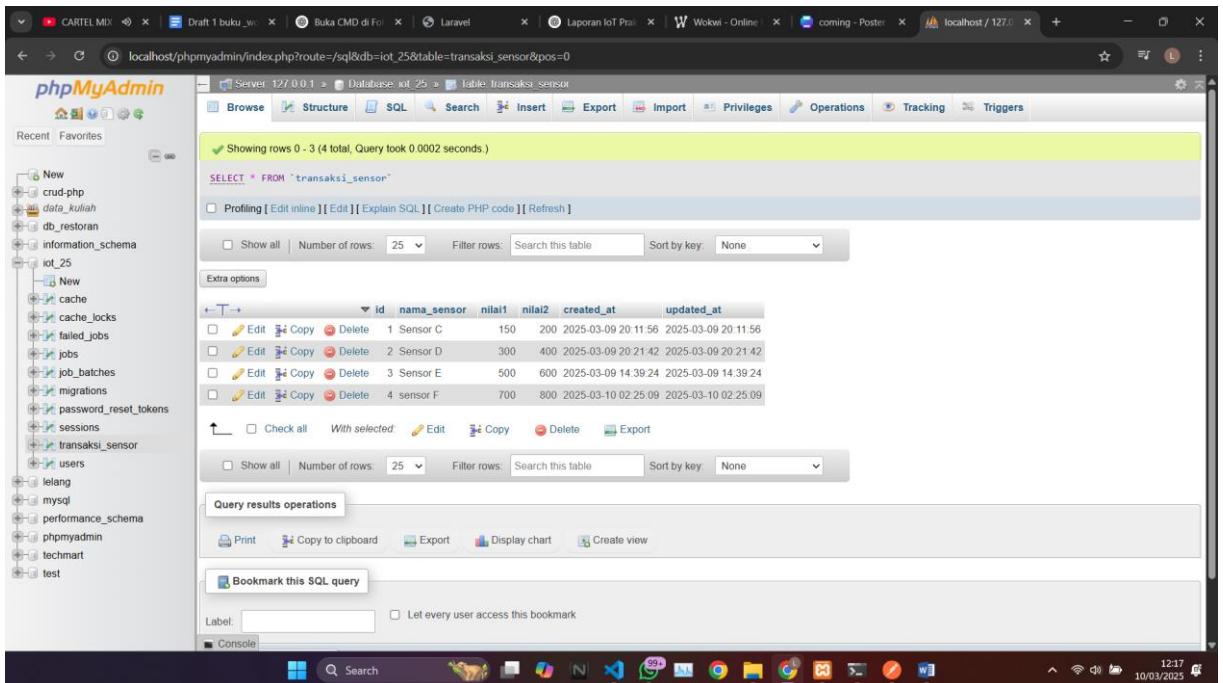
```
use Illuminate\Support\Facades\Route;
```

```
Route::get('/user', function (Request $request) {  
    return $request->user();  
})->middleware(Authenticate::using('sanctum'));
```

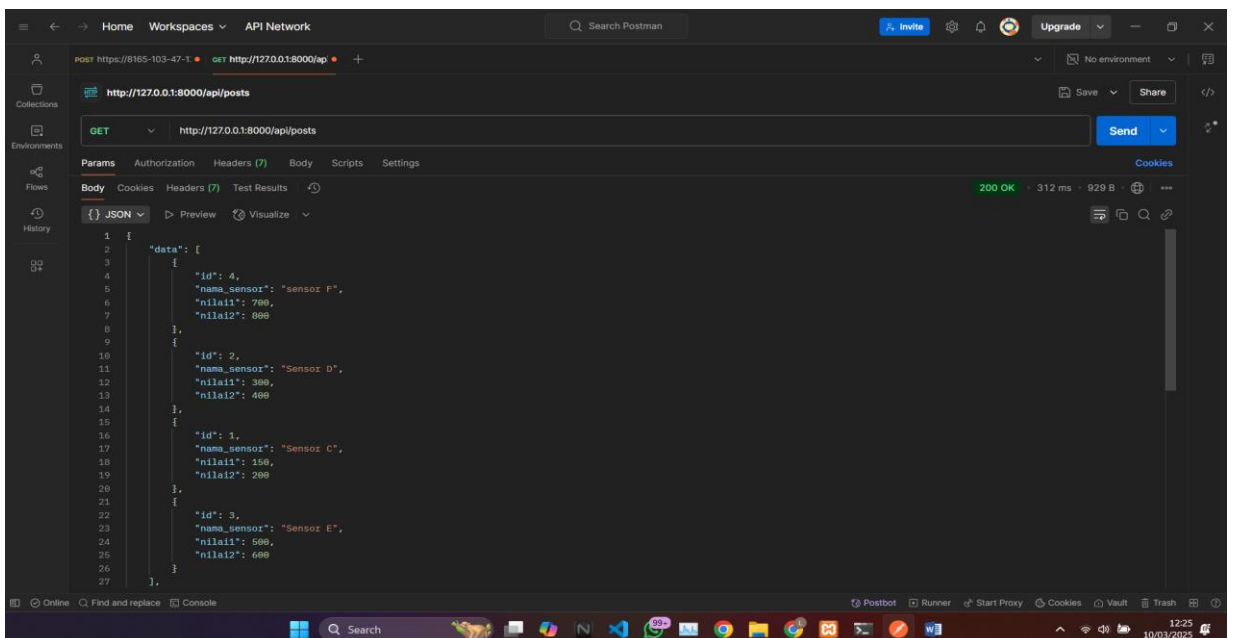
```
//posts
```

```
Route::apiResource('/posts',  
App\Http\Controllers\Api\TransaksiSensorController::class);
```

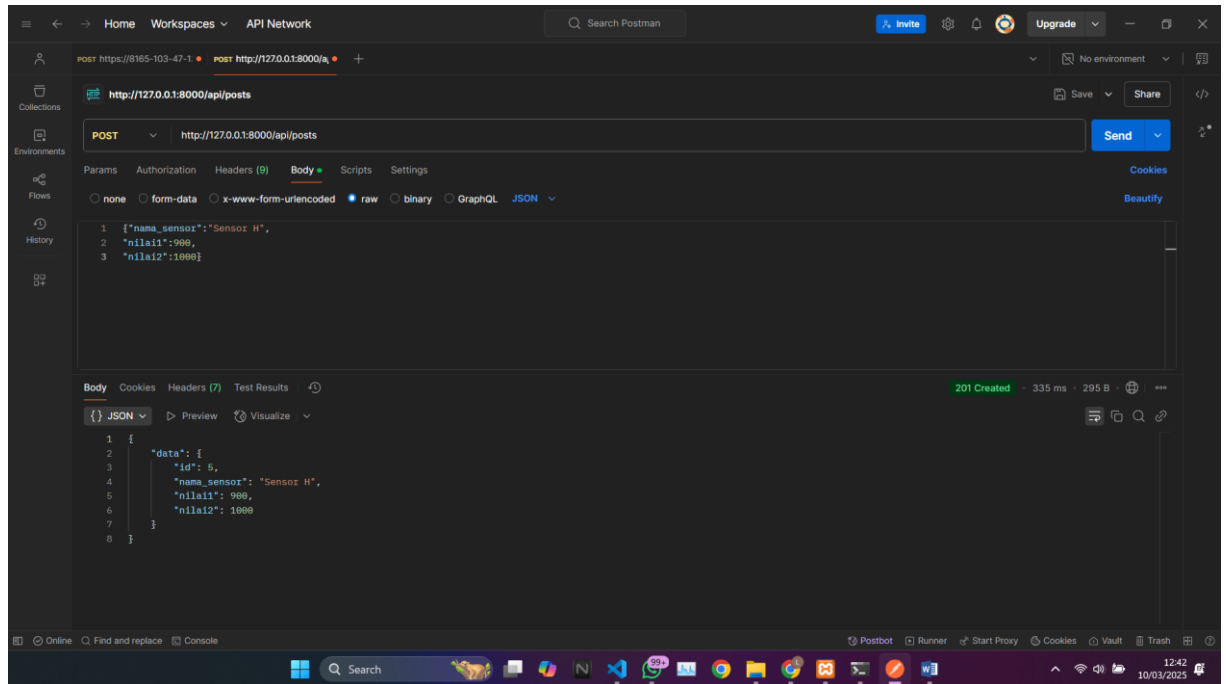
## g. Database



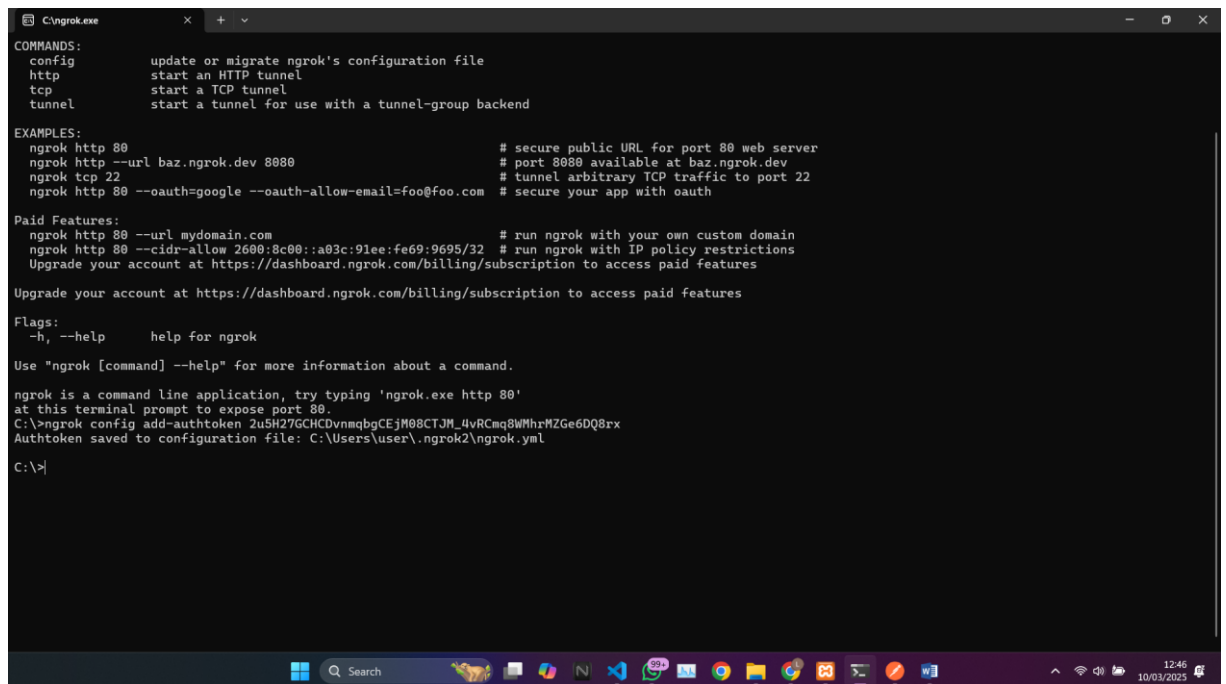
## H, Laravel API (Get Postman)



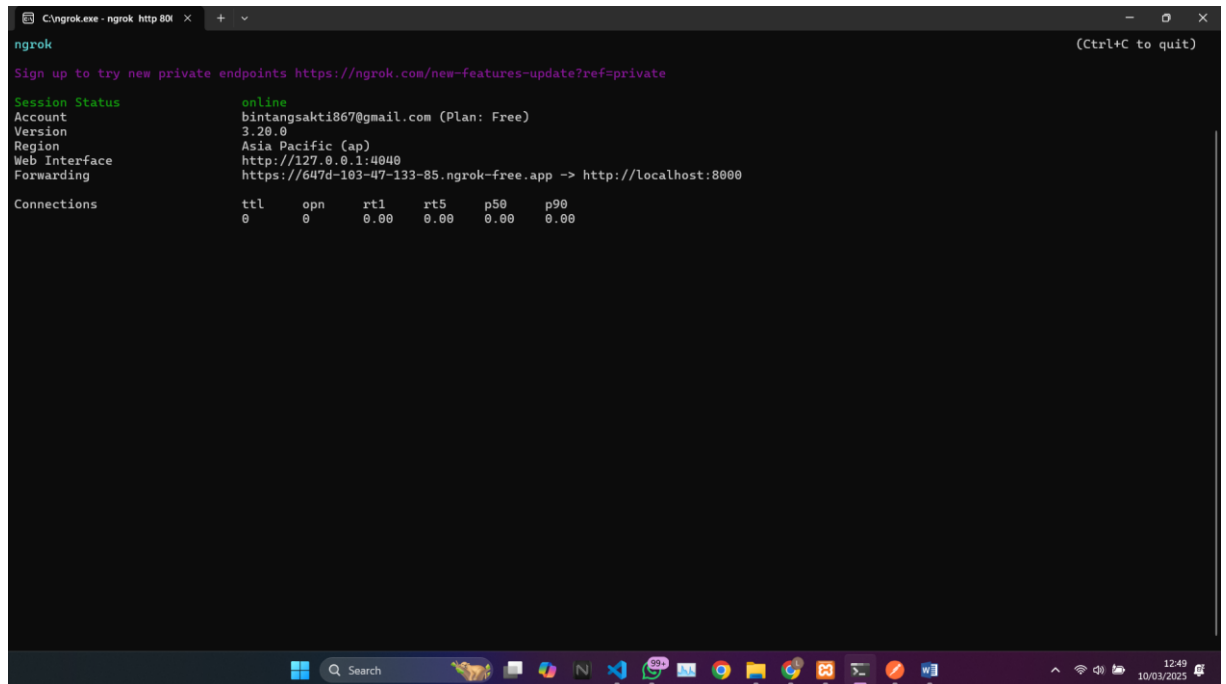
## I. Laravel API (Post Postman)



## J. ngrok.exe



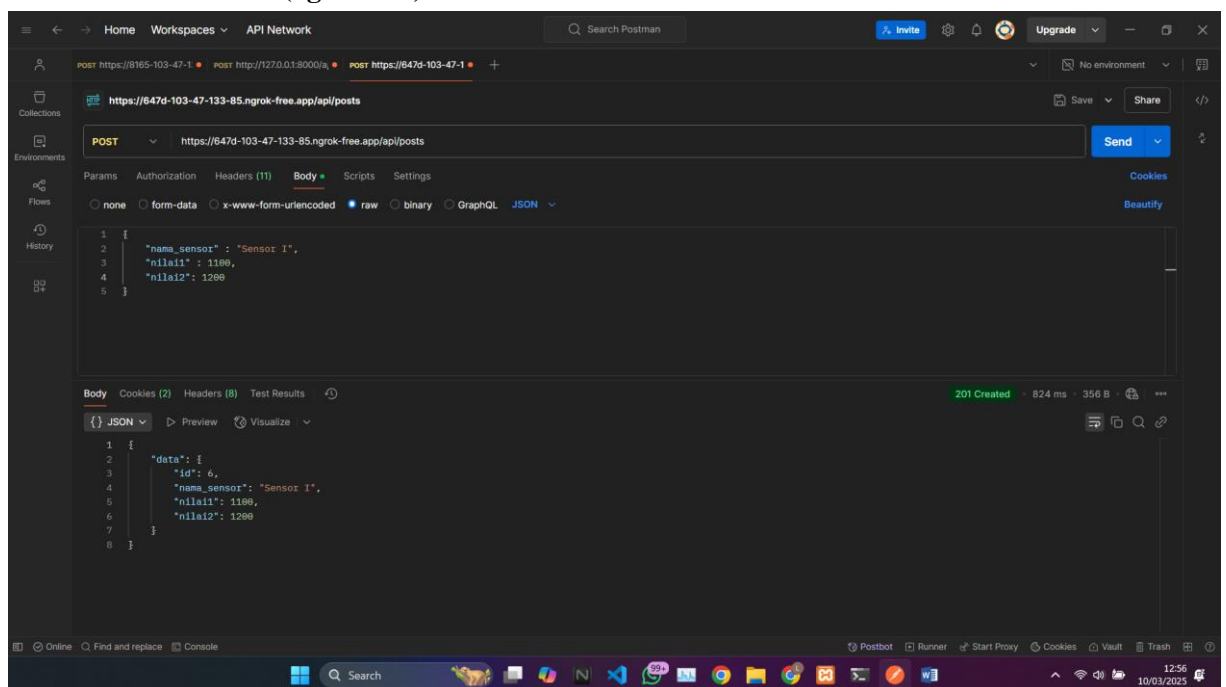
## K.Mengonline kan ngrok.exe



The screenshot shows the ngrok application window. At the top, it says "ngrok" and "(Ctrl+C to quit)". Below that, there is a link to sign up for private endpoints. The "Session Status" section shows the account is "online" for "bintangsaiki867@gmail.com (Plan: Free)". It lists the version as "3.20.0", the region as "Asia Pacific (ap)", and the web interface URL as "http://127.0.0.1:4840". The forwarding URL is "https://647d-103-47-133-85.ngrok-free.app -> http://localhost:8000". A "Connections" table is displayed at the bottom.

	t1	opn	rt1	rt5	p50	p90
Connections	0	0	0.00	0.00	0.00	0.00

## L.Menambahkan DB (ngrok.exe)



The screenshot shows the Postman application interface. A POST request is configured to "https://647d-103-47-133-85.ngrok-free.app/api/posts". The request body is in raw JSON format. The response is a 201 Created status with a JSON body containing a list of data.

Request Body (raw):

```
1 {
2   "nama_sensor": "Sensor I",
3   "nilai1": 1100,
4   "nilai2": 1200
5 }
```

Response Body (JSON):

```
1 {
2   "data": {
3     "id": 6,
4     "nama_sensor": "Sensor I",
5     "nilai1": 1100,
6     "nilai2": 1200
7   }
8 }
```