

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Praktik Simulasi Relay, Button & LED



Femas Alfaridzi

Fakultas Vokasi , Universitas Brawijaya

Email : femasalfaridzi17@gmail.com

**Fakultas Vokasi
Universitas Brawijaya**

Abstrak

Praktikum ini bertujuan untuk mensimulasikan penggunaan relay, tombol (button), dan LED menggunakan Visual Studio Code (VSCode) serta Wokwi Simulator. Pada simulasi ini, relay digunakan sebagai saklar elektronik untuk mengontrol aliran listrik, sementara tombol berfungsi sebagai input untuk mengaktifkan atau menonaktifkan LED. Wokwi Simulator dipilih karena kemampuannya dalam memvisualisasikan rangkaian elektronik secara interaktif tanpa memerlukan perangkat keras fisik.

Metode yang digunakan meliputi perancangan rangkaian pada Wokwi Simulator, pemrograman menggunakan bahasa C untuk mikrokontroler berbasis Arduino, serta pengujian skenario interaksi antara relay, tombol, dan LED. Hasil praktikum menunjukkan bahwa kombinasi perangkat ini dapat bekerja secara efektif dalam sistem otomatisasi sederhana, di mana tombol dapat mengendalikan relay untuk mengaktifkan atau mematikan LED sesuai dengan logika yang telah diprogram.

Dengan adanya simulasi ini, mahasiswa dapat memahami prinsip kerja relay, tombol, dan LED dalam sistem elektronik serta meningkatkan keterampilan dalam pemrograman mikrokontroler.

Keywords: Relay, LED,

1. Pendahuluan

1.1 Latar Belakang

Praktikum ini dilakukan untuk memahami cara kerja relay, tombol (button), dan LED serta bagaimana ketiganya berinteraksi dalam sebuah rangkaian elektronik. Dengan menggunakan Wokwi Simulator di VSCode, praktikum ini memungkinkan simulasi tanpa perangkat fisik, sehingga lebih efisien dalam pembelajaran dasar otomasi sederhana.

1.2 Tujuan Eksperimen

- Memahami prinsip kerja relay, tombol, dan LED dalam sistem elektronik.
- Mensimulasikan interaksi antara relay, tombol, dan LED menggunakan Wokwi Simulator di VSCode

2. Metodologi

2.1 Alat & Bahan

Laptop, Visual Studio Code, dan koneksi internet

2.2 Langkah Implementasi

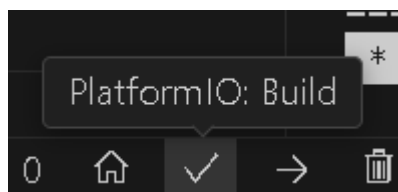
Ubah Kode di file **main.cpp** pada folder **src** menjadi seperti ini

```

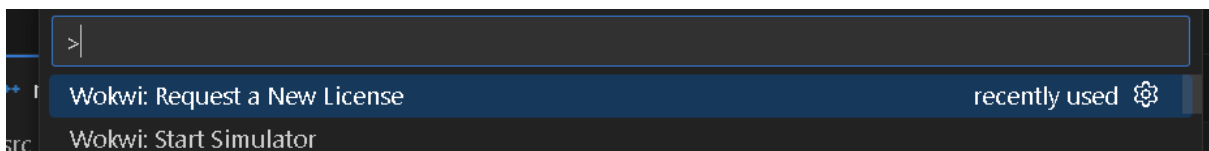
1 #include <Arduino.h>
2 // Define pin numbers
3 const int ButtonPin = 19; // GPIO19 connected to the pushbutton
4 const int LedPin = 18; // GPIO18 connected to the LED
5 const int RelayPin = 23; // GPIO23 connected to the relay module
6
7 void setup() {
8   // Set pin modes
9   pinMode(ButtonPin, INPUT_PULLUP); // Set the button pin as an input with an internal pull-up resistor
10
11   pinMode(LedPin, OUTPUT); // Set the LED pin as an output
12   pinMode(RelayPin, OUTPUT); // Set the relay pin as an output
13
14   // Initialize the outputs to be OFF
15   digitalWrite(LedPin, LOW);
16   digitalWrite(RelayPin, LOW);
17 }
18
19 void loop() {
20   // Read the state of the button
21   int buttonState = digitalRead(ButtonPin);
22
23   // Check if the button is pressed
24   // Since the button is wired to pull the pin LOW when pressed, we check for LOW
25   if (buttonState == LOW) {
26     digitalWrite(LedPin, HIGH); // Turn on the LED
27     digitalWrite(RelayPin, HIGH); // Turn on the relay
28   } else {
29     digitalWrite(LedPin, LOW); // Turn off the LED
30     digitalWrite(RelayPin, LOW); // Turn off the relay
31   }
32 }

```

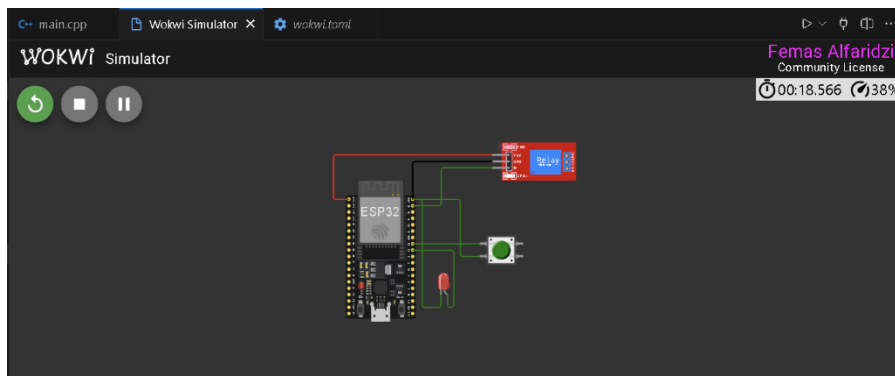
Kemudian **build**



Lalu **compile** dengan menekan **ctrl+shift + p**, pilih yang **wokwi : start simulator**

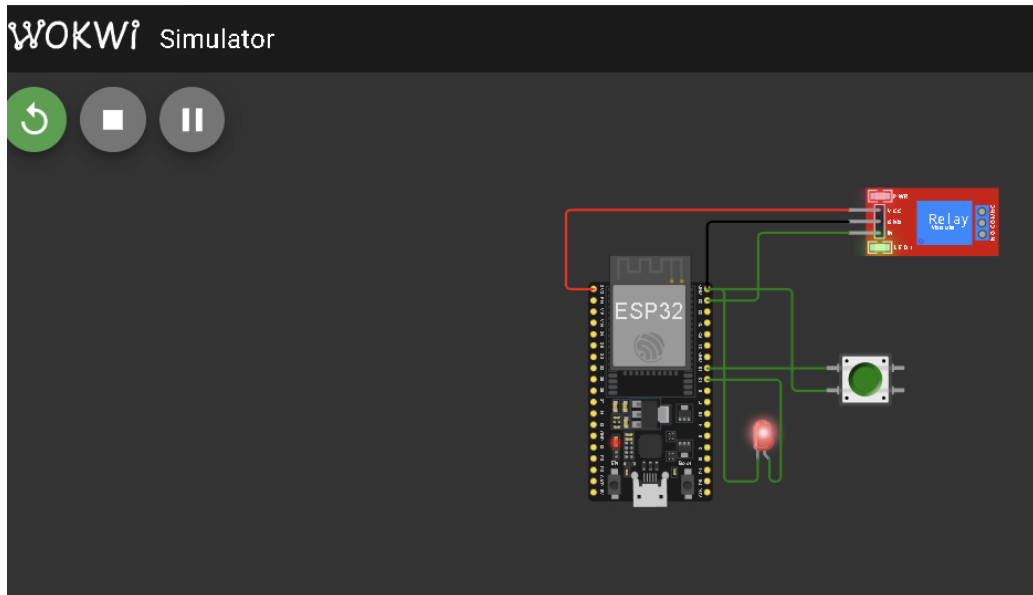


Tampilan jika berhasil



3. Hasil dan Pembahasan

3.1. Hasil Eksperimen



4. Lampiran

Pendefinisian Pin

- ButtonPin (GPIO19) → Terhubung ke push button sebagai input.
- LedPin (GPIO18) → Terhubung ke LED sebagai output.
- RelayPin (GPIO23) → Terhubung ke relay sebagai output.

setup() (Inisialisasi Awal)

- ButtonPin diatur sebagai INPUT_PULLUP (menggunakan resistor pull-up internal).
- LedPin & RelayPin diatur sebagai OUTPUT dan dinonaktifkan (LOW) saat awal.

loop() (Logika Kontrol)

- Membaca kondisi push button.
- Jika tombol ditekan (LOW), LED & relay menyala (HIGH).
- Jika tombol dilepas (HIGH), LED & relay mati (LOW).

Program ini memastikan LED dan relay hanya aktif selama tombol ditekan.

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Praktik Simulasi Sensor Jarak (Ultrasonic)



Femas Alfaridzi

Fakultas Vokasi , Universitas Brawijaya

Email : femasalfaridzi17@gmail.com

**Fakultas Vokasi
Universitas Brawijaya**

Abstrak

Praktik ini bertujuan untuk mensimulasikan sistem pengukuran jarak menggunakan **sensor ultrasonik HC-SR04** yang terhubung dengan **ESP32** dalam lingkungan **Wokwi Simulator** di **VSCode**. Sensor ini bekerja dengan mengirimkan gelombang ultrasonik melalui **pin Trigger** dan menerima pantulannya melalui **pin Echo**, lalu menghitung jarak berdasarkan waktu tempuh gelombang suara. Program dikembangkan menggunakan **Arduino C**, dengan konfigurasi untuk membaca data dari sensor dan menampilkannya dalam satuan **centimeter (cm)** pada **serial monitor**. Hasil simulasi menunjukkan bahwa sistem dapat mengukur jarak objek secara akurat dalam lingkungan virtual, memberikan wawasan mengenai penerapan sensor ultrasonik dalam berbagai proyek berbasis IoT dan otomatisasi.

Keywords: ESP32, HC-SR04, Sensor Ultrasonik

1. Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi sensor semakin pesat dan telah banyak diterapkan dalam berbagai bidang, termasuk sistem otomatisasi dan Internet of Things (IoT). Salah satu sensor yang sering digunakan adalah **sensor ultrasonik HC-SR04**, yang mampu mengukur jarak suatu objek dengan memanfaatkan gelombang ultrasonik. Sensor ini memiliki keunggulan dalam kecepatan dan akurasi pengukuran, sehingga sering digunakan dalam berbagai aplikasi seperti sistem parkir otomatis, robotika, dan pemantauan lingkungan.

Dalam praktik ini, dilakukan simulasi **sensor ultrasonik HC-SR04** menggunakan **ESP32** dalam **Wokwi Simulator** di **VSCode**. Penggunaan ESP32 sebagai mikrokontroler dipilih karena kemampuannya yang fleksibel serta dukungan konektivitas yang luas. Simulasi ini bertujuan untuk memahami cara kerja sensor ultrasonik dalam mengukur jarak objek, mengonversi data waktu tempuh gelombang menjadi satuan **centimeter (cm)**, serta menampilkan hasil pengukuran secara real-time di **serial monitor**.

Melalui praktik ini, diharapkan peserta dapat memahami prinsip kerja **sensor ultrasonik**, konfigurasi perangkat keras dalam lingkungan virtual, serta implementasi kode pemrograman berbasis **Arduino C** dalam ESP32. Selain itu, simulasi ini juga memberikan wawasan mengenai potensi penerapan sensor ultrasonik dalam berbagai bidang industri dan penelitian.

1.2 Tujuan Eksperimen

- Memahami prinsip kerja sensor ultrasonik HC-SR04 dalam mengukur jarak berdasarkan waktu tempuh gelombang ultrasonik. Mensimulasikan interaksi antara relay, tombol, dan LED menggunakan Wokwi Simulator di VSCode
- Menampilkan hasil pengukuran jarak secara real-time melalui serial monitor di Wokwi Simulator.

2. Metodologi

2.1. Alat & Bahan

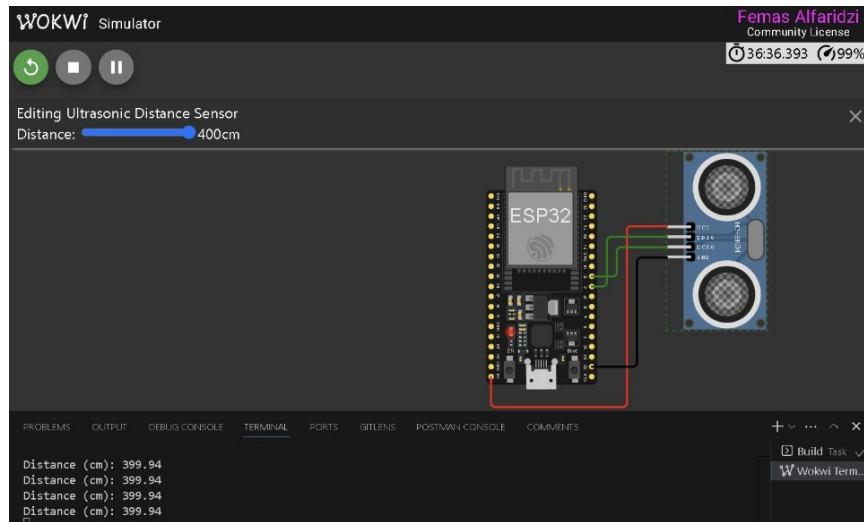
Laptop, Visual Studio Code, dan koneksi internet

2.2. Langkah Implementasi

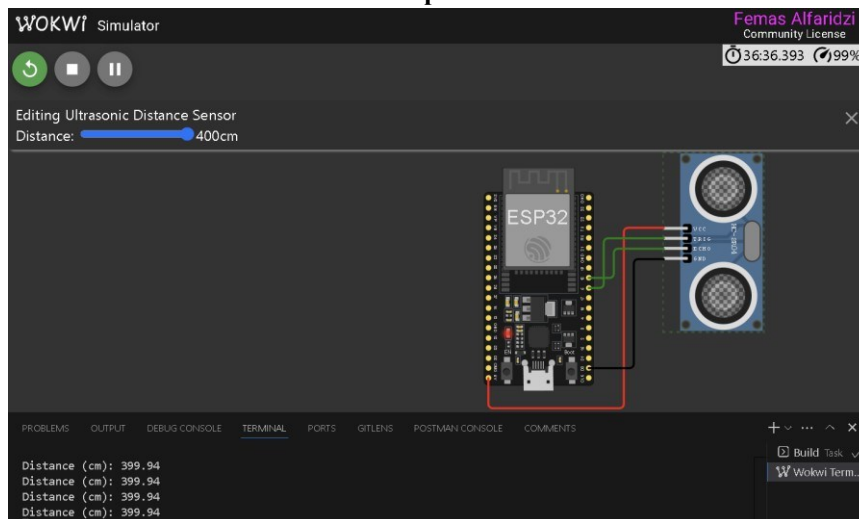
Ubah code pada file **main.cpp** yang ada pada **folder src** menjadi seperti ini

```
1  #include<Arduino.h>
2  const int trigPin = 5;
3  const int echoPin = 18;
4  //define sound speed in cm/uS
5  #define SOUND_SPEED 0.034
6  #define CM_TO_INCH 0.393701
7  long duration;
8  float distanceCm;
9  float distanceInch;
10 void setup() {
11   Serial.begin(115200); // Starts the serial communication
12   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
13   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
14 }
15 void loop() {
16   // Clears the trigPin
17   digitalWrite(trigPin, LOW);
18   delayMicroseconds(2);
19   // Sets the trigPin on HIGH state for 10 micro seconds
20   digitalWrite(trigPin, HIGH);
21   delayMicroseconds(10);
22   digitalWrite(trigPin, LOW);
23   // Reads the echoPin, returns the sound wave travel time in microseconds
24   duration = pulseIn(echoPin, HIGH);
25   // Calculate the distance
26   distanceCm = duration * SOUND_SPEED/2;
27   // Convert to inches
28   distanceInch = distanceCm * CM_TO_INCH;
29   // Prints the distance in the Serial Monitor
30   Serial.print("Distance (cm): ");
31   Serial.println(distanceCm);
32   // Serial.print("Distance (inch): ");
33   // Serial.println(distanceInch);
34   delay(1000);
35 }
```

Kemudian **Build**, lalu **compile** dengan cara **ctrl+shift+p** , pilih yang **start:wokwi simulator**, jika berhasil maka tampilannya akan seperti ini



3. Hasil dan Pembahasan 3.1. Hasil Eksperimen



4. Lampiran

1. Deklarasi Pin dan Konstanta
 - trigPin (GPIO 5) sebagai output untuk mengirim sinyal ultrasonik.
 - echoPin (GPIO 18) sebagai input untuk menerima pantulan sinyal.
 - SOUND_SPEED didefinisikan sebagai **0.034 cm/μs** untuk menghitung jarak.
 - CM_TO_INCH digunakan untuk mengonversi hasil pengukuran ke satuan inci.
2. Fungsi setup()
 - Inisialisasi komunikasi serial pada baud rate **115200** untuk menampilkan hasil.
 - Mengatur **trigPin** sebagai **OUTPUT** dan **echoPin** sebagai **INPUT**.

3. **Fungsi loop()** ○ Membersihkan **trigPin** dengan **LOW** selama **2 μs**. ○

Mengaktifkan **trigPin** selama **10 μs** untuk mengirimkan pulsa ultrasonik.

- Membaca durasi pantulan sinyal pada **echoPin** menggunakan `pulseIn()`.
- **Menghitung jarak** dengan rumus:

$$\text{distanceCm} = \frac{\text{duration} \times \text{SOUND_SPEED}}{2}$$

- Delay **1 detik** sebelum melakukan pengukuran berikutnya.

Kode ini memungkinkan pengguna untuk memonitor jarak objek secara real-time dengan sensor ultrasonik dalam lingkungan simulasi **Wokwi**.

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

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



Femas Alfaridzi

Fakultas Vokasi , Universitas Brawijaya

Email : femasalfaridzi17@gmail.com

**Fakultas Vokasi
Universitas Brawijaya**

Abstrak

Praktikum ini bertujuan untuk membuat API menggunakan Laravel 11 dan Ngrok pada bab 12. Pembuatan API menggunakan Ngrok dimaksudkan agar API dapat diakses secara *online* kapanpun dan dimanapun. Praktik dilakukan dengan menggunakan beberapa *tools* seperti XAMPP sebagai server database, Herd sebagai alat pengelola proyek laravel 11, Postman sebagai media uji coba API, dan Ngrok sebagai terowongan untuk menghubungkan server lokal dengan internet, serta teks editor Visual Studio Code. Hasilnya, pembuatan API telah berhasil terhubung dengan Ngrok dan dapat diakses melalui internet.

Keywords: Laravel 11, API, Ngrok, Xampp, Herd, Postman

1. Pendahuluan

1.1 Latar Belakang

API (Application Programming Interface) adalah serangkaian protokol yang memungkinkan satu aplikasi saling berkomunikasi dengan aplikasi lain misalnya klien dengan server. Pembuatan API dapat menggunakan berbagai macam framework, salah satunya yang paling populer adalah laravel 11.

Laravel 11 merupakan *framework* PHP yang menyediakan berbagai *tools* dan *library* yang memudahkan pengembang dalam pembuatan API. Laravel 11 memungkinkan kita mengelola berbagai *tools* seperti *routing*, autentikasi, *middleware*, dan pengelolaan database.

Ngrok adalah *proxy* server untuk membuat jaringan *private* melalui NAT atau *firewall* untuk menghubungkan server lokal ke internet dengan aman. Ngrok membuat URL publik yang dapat digunakan untuk mengakses API secara *online* kapanpun dan dimanapun. Ngrok sangat berguna untuk mengelola perangkat IoT dari jarak jauh tanpa perlu pengaturan IP publik atau NAT traversal.

Postman adalah alat yang digunakan untuk melakukan uji coba API dengan mengirimkan permintaan http ke API dan memeriksa respon yang diterima, sehingga pengembang dapat memastikan apakah API berjalan sesuai yang diharapkan atau tidak. Dengan ini kita akan membuat dan menghubungkan API dengan Ngrok, serta mengujinya menggunakan Postman.

1.2 Tujuan Eksperimen

Tujuan dari praktik ini adalah membangun API yang akan diintegrasikan dengan perangkat IoT menggunakan laravel 11, menghubungkannya dengan Ngrok supaya dapat diakses secara online, dan menguji API berjalan atau tidak menggunakan postman.

2. Metodologi

2.1 Alat & Bahan

Laptop, Visual Studio Code, XAMPP, phpMyAdmin, Herd, Laravel 11, Postman, Ngrok, dan koneksi internet

2.2 Langkah Implementasi

1. Buat Proyek *Framework* Laravel 11 menggunakan Herd

Create new table

Nama tabel


iot_25


Jumlah kolom

4

Buat

Create New Site


New Laravel project


Link existing project

Start a fresh Laravel project.

Previous

Next

Create New Site

Project Name: bab12

Testing Framework: Pest

Target Location: C:\Users\Jimmy Indrianto\Documents\Platfo

Your project will be created in this folder.

Previous

Next

2. Buat database bernama iot_25 di phpmyadmin.

3. Buka proyek laravel 11 pada vs code dan edit file .env pada bagian DB_CONNECTION seperti berikut:
4. Buat file TransaksiSensor.php menggunakan perintah :
php artisan make:model TransaksiSensor -m

<pre>DB_CONNECTION=sqlite # DB_HOST=127.0.0.1 # DB_PORT=3306 # DB_DATABASE=laravel # DB_USERNAME=root # DB_PASSWORD=</pre>	<pre>DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=iot_25 DB_USERNAME=root DB_PASSWORD=</pre>
--	--

```
PS C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12> php artisan make:model TransaksiSensor -m

INFO Model [C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12\app\Models\TransaksiSensor.php] created successfully.

INFO Migration [C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12\database\Migrations\2025_03_12_062514_create_transaksi_sensors_table.php] created successfully.
```

Lalu ubah file **database\Migrations\2025_03_08_003518_create_transaksi_sensors_table.php** seperti kode dibawah ini

```
1  <?php
2
3
4  use Illuminate\Database\Migrations\Migration;
5  use Illuminate\Database\Schema\Blueprint;
6  use Illuminate\Support\Facades\Schema;
7
8
9  return new class extends Migration
10 {
11     /**
12      * Run the migrations.
13      */
14     public function up(): void
15     {
16         Schema::create('transaksi_sensor', function (Blueprint $table) {
17             $table->id('id')->startingValue(1); // Menetapkan AUTO_INCREMENT dimulai dari 1
18             $table->string('nama_sensor', 255); // varchar(255)
19             $table->integer('nilai1', false)->length(255); // int(255)
20             $table->integer('nilai2', false)->length(255); // int(255)
21             $table->timestamps(); // Menambahkan created_at dan updated_at
22         });
23     }
24
25
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('transaksi_sensors');
33     }
34 };
35
```

5. Ubah isi file `app/Models/TransaksiSensor.php` dengan kode berikut

```
1  <?php
2
3
4  namespace App\Models;
5
6
7  use Illuminate\Database\Eloquent\Factories\HasFactory;
8  use Illuminate\Database\Eloquent\Model;
9
10
11  class TransaksiSensor extends Model
12  {
13      use HasFactory;
14
15      /**
16       * The table associated with the model.
17       *
18       * @var string
19       */
20      protected $table = 'transaksi_sensor';
21
22
23      /**
24       * The attributes that are mass assignable.
25       *
26       * @var array
27       */
28      protected $fillable = [
29          'nama_sensor',
30          'nilai1',
31          'nilai2',
32      ];
33
34
35      /**
36       * The attributes that should be hidden for arrays.
37       *
38       * @var array
39       */
40      protected $hidden = [];
41
42
43      /**
44       * The attributes that should be cast.
45       *
46       * @var array
47       */
48      protected $casts = [];
49  }
50
```

6. Kemudian jalankan perintah `php artisan migrate` untuk membuat tabel pada database.

```
PS C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12> php artisan migrate

INFO: Preparing database.

Creating migration table ..... 24.66ms DONE

INFO: Running migrations.

0001_01_01_000000_create_users_table ..... 109.40ms DONE
0001_01_01_000001_create_cache_table ..... 18.75ms DONE
0001_01_01_000002_create_jobs_table ..... 80.10ms DONE
2025_03_12_062514_create_transaksi_sensors_table ..... 16.34ms DONE
```

7. Buat Resource dengan menjalankan perintah `php artisan make:resource TransaksiSensorResource`

```
PS C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12> php artisan make:resource TransaksiSensorResource

INFO: Resource [C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12\app\Http\Resources\TransaksiSensorResource.php] created successfully.
```

Lalu ubah isi file `app\Http\Resources\TransaksiSensorResource.php` dengan kode dibawah

```
1 <?php
2
3
4 namespace App\Http\Resources;
5 use Illuminate\Http\Request;
6 use Illuminate\Http\Resources\Json\JsonResource;
7
8
9 class TransaksiSensorResource extends JsonResource
10 {
11     /**
12      * Transform the resource into an array.
13      *
14      * @param \Illuminate\Http\Request $request
15      * @return array
16      */
17     public function toArray($request)
18     {
19         return [
20             'id' => $this->id,
21             'nama_sensor' => $this->nama_sensor,
22             'nilai1' => $this->nilai1,
23             'nilai2' => $this->nilai2,
24         ];
25     }
26 }
27
```


8. Buat API *controller* dengan perintah
php artisan make:controller Api/TransaksiSensorController

```
>>
INFO Controller [C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12\app\Http\Controllers\Api\TransaksiSensorController.php] created successfully.
```

Lalu ubah isi file app/Http/Controllers/Api/TransaksiSensorController.php dengan kode dibawah

```
1 <?php
2
3
4 namespace App\Http\Controllers\Api;
5 use Illuminate\Http\Request;
6
7
8 use App\Models\TransaksiSensor;
9 use App\Http\Controllers\Controller;
10 use App\Http\Resources\TransaksiSensorResource;
11
12
13 class TransaksiSensorController extends Controller
14 {
15     /**
16      * index
17      *
18      * @return \Illuminate\Http\Response
19      */
20     public function index()
21     {
22         // get all transactions from transaksiSensor model, paginated
23         $transaksiSensors = TransaksiSensor::latest()->paginate(5);
24
25         // Return a collection of transactions as a resource
26         return TransaksiSensorResource::collection($transaksiSensors);
27     }
28
29
30
31
32
33
34
35     /**
36      * Store a newly created resource in storage.
37      *
38      * @param \Illuminate\Http\Request $request
39      * @return \Illuminate\Http\Response
40      */
41     public function store(Request $request)
42     {
43         $validatedData = $request->validate([
44             'nama_sensor' => 'required|string|max:255',
45             'nilai1' => 'required|integer',
46             'nilai2' => 'required|integer',
47         ]);
48
49         $transaksiSensor = TransaksiSensor::create($validatedData);
50
51         return new TransaksiSensorResource($transaksiSensor);
52     }
53
54
55
56
57
58
59
60
61
62
63
64
65     /**
66      * Display the specified resource.
67      *
68      * @param int $id
69      * @return \Illuminate\Http\Response
70      */
71     public function show($id)
72     {
73         $transaksiSensor = TransaksiSensor::findOrFail($id);
74
75         return new TransaksiSensorResource($transaksiSensor);
76     }
77
78
79
80
81
82
83
84
85     /**
86      * Update the specified resource in storage.
87      *
88      * @param \Illuminate\Http\Request $request
89      * @param int $id
90      * @return \Illuminate\Http\Response
91      */
92     public function update(Request $request, $id)
93     {
94         $validatedData = $request->validate([
95             'nama_sensor' => 'required|string|max:255',
96             'nilai1' => 'required|integer',
97             'nilai2' => 'required|integer',
98         ]);
99
100         $transaksiSensor = TransaksiSensor::findOrFail($id);
101         $transaksiSensor->update($validatedData);
102
103         return new TransaksiSensorResource($transaksiSensor);
104     }
105
106     /**
107      * Remove the specified resource from storage.
108      *
109      * @param int $id
110      * @return \Illuminate\Http\Response
111      */
112     public function destroy($id)
113     {
114         $transaksiSensor = TransaksiSensor::findOrFail($id);
115         $transaksiSensor->delete();
116
117         return response()->json(['message' => 'Deleted successfully'], 204);
118     }
119
120
121
122
123
124
125
126
```

9. Buat *route* khusus API dengan perintah **php artisan install:api**, jika ada pilihan yes/no pilih yes

```
> yes

INFO Running migrations.

2025_03_12_070954_create_personal_access_tokens_table .....

INFO API scaffolding installed. Please add the [Laravel\Sanctum\HasApiTokens] trait to your User model.
```

Ubah isi file `routes/api.php` dengan kode berikut

```
1 <?php
2
3
4 use Illuminate\Auth\Middleware\Authenticate;
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Route;
7
8
9 Route::get('/user', function (Request $request) {
10     return $request->user();
11 }->middleware(Authenticate::using('sanctum'));
12
13
14 //posts
15 Route::apiResource('/posts', App\Http\Controllers\Api\TransaksiSensorController::class);
16
```

10. Pastikan route sudah terbentuk dengan memasukkan perintah `php artisan route:list`, dan pastikan tampilan pada terminal seperti gambar dibawah ini.

```
PS C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12> php artisan route:list

GET|HEAD / .....
GET|HEAD api/posts ..... posts.index > Api\TransaksiSensorController@index
POST api/posts ..... posts.store > Api\TransaksiSensorController@store
GET|HEAD api/posts/{post} ..... posts.show > Api\TransaksiSensorController@show
PUT|PATCH api/posts/{post} ..... posts.update > Api\TransaksiSensorController@update
DELETE api/posts/{post} ..... posts.destroy > Api\TransaksiSensorController@destroy
GET|HEAD api/user .....
GET|HEAD sanctum/csrf-cookie ..... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show
GET|HEAD storage/{path} ..... storage.local
GET|HEAD up .....

Showing [10] routes
```

11. Testing dengan menggunakan *tools* postman. Jika belum punya, *download* postman dan lakukan instalasi. Untuk mencoba mengakses API, pastikan laravel berjalan dengan perintah **php artisan serve**

```
PS C:\Users\Jimmy Indrianto\Documents\PlatformIO\Projects\Minggu3\bab12> php artisan serve

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

Lalu pastikan ada data pada tabel `transaksi_sensor`, misalnya

Server: 127.0.0.1 » Database: iot_25 » Tabel: transaksi_sensor

Jelajahi Struktur SQL Cari Tambahkan Ekspor Impor Hak Akses Operasi

[Edit dikotak] [Ubah] [Buat kode PHP]

✓ Menampilkan baris 0 - 1 (total 2, Pencarian dilakukan dalam 0,0005 detik.)

`SELECT * FROM `transaksi_sensor``

☐ Profil [Edit dikotak] [Ubah] [Jelaskan SQL] [Buat kode PHP] [Segarkan]

☐ Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

Extra options

		id	nama_sensor	nilai1	nilai2	created_at	updated_at
<input type="checkbox"/>	Ubah Salin Hapus	1	Sensor A	100	200	NULL	NULL
<input type="checkbox"/>	Ubah Salin Hapus	2	Sensor B	87	176	NULL	NULL

12. Pada bagan url, masukkan alamat server laravel dan route API-nya : **http://localhost:8000/api/posts** , lalu pilih *method GET* dan klik **Send** seperti gambar dibawah,

http://localhost:8000/api/posts

GET http://localhost:8000/api/posts Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
-----	-------	-------------

Body Cookies Headers (7) Test Results 200 OK • 585 ms • 808 B

Pretty Raw Preview Visualize JSON

```

1 {
2   "data": [
3     {
4       "id": 1,
5       "nama_sensor": "Sensor A",
6       "nilai1": 100,
7       "nilai2": 200
8     },
9     {
10      "id": 2,
11      "nama_sensor": "Sensor B",
12      "nilai1": 87,
13      "nilai2": 176
14    }
15  ],
16  "links": {

```

```

17    "first": "http://localhost:8000/api/posts?page=1",
18    "last": "http://localhost:8000/api/posts?page=1",
19    "prev": null,
20    "next": null
21  },
22  "meta": {
23    "current_page": 1,
24    "from": 1,
25    "last_page": 1,
26    "links": [
27      {
28        "url": null,
29        "label": "&laquo; Previous",
30        "active": false
31      },
32    ]

```

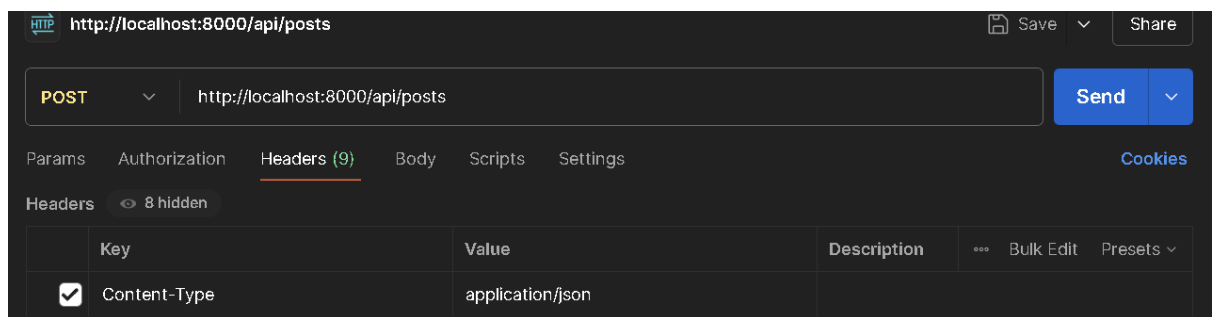
```

33         "url": "http://localhost:8000/api/posts?page=1",
34         "label": "1",
35         "active": true
36     },
37     {
38         "url": null,
39         "label": "Next &raquo;",
40         "active": false
41     }
42 ],
43     "path": "http://localhost:8000/api/posts",
44     "per_page": 5,
45     "to": 2,
46     "total": 2
47 }
48 }

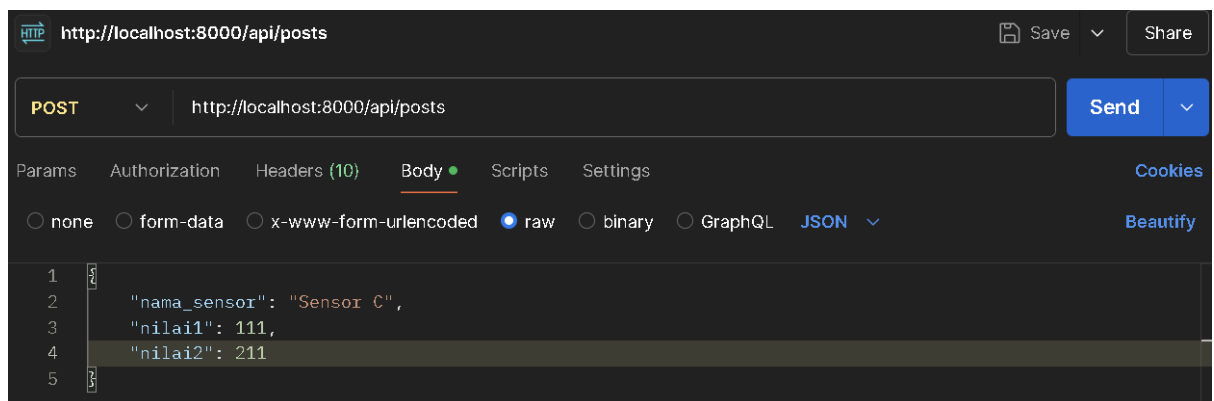
```

dan API sudah berhasil mengambil data dari database.

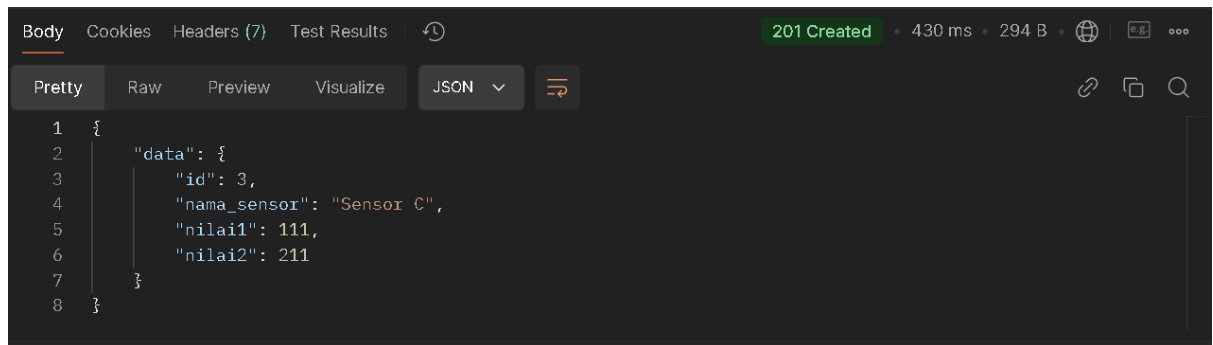
13. Lakukan percobaan *insert* data ke database dengan mengganti *method* menjadi POST lalu pastikan *Content-Type* pada *Headers* adalah *application/json*



kemudian pada *Body* gunakan *raw* dan format json, lalu isi seperti berikut



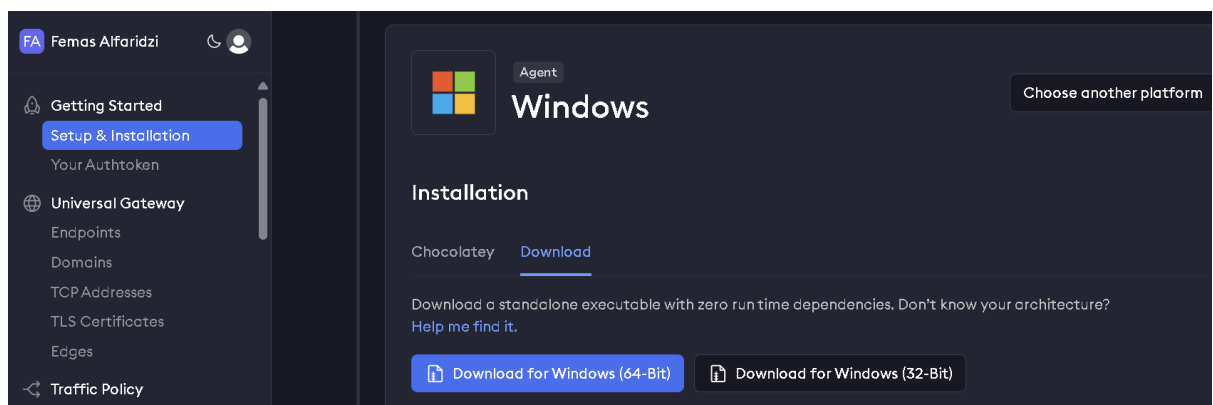
Kemudian klik send, Jika berhasil maka hasilnya akan seperti ini



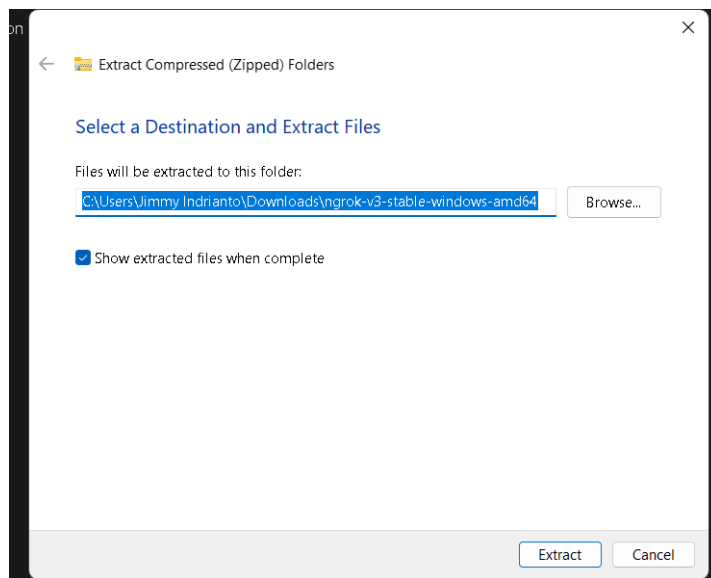
Jangan lupa cek di phpmyadmin untuk memastikan datanya benar-benar bertambah



14. Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot. Buka url <https://dashboard.ngrok.com/signup> dan lakukan register (saya menggunakan akun github), lalu download Ngrok sesuai sistem operasi laptop.



ekstrak file zip yang sudah di *download* dan klik 2x file ngrok.exe untuk membuka cmd.



```
ngrok [command] [flags]

COMMANDS:
  config      update or migrate ngrok's configuration file
  http        start an HTTP tunnel
  tcp         start a TCP tunnel
  tunnel      start a tunnel for use with a tunnel-group backend

EXAMPLES:
  ngrok http 80                                # secure public URL for port 80 we
  ngrok http --url baz.ngrok.dev 8080          # port 8080 available at baz.ngrok
  ngrok tcp 22                                  # tunnel arbitrary TCP traffic to
  ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com # secure your app with oauth

Paid Features:
  ngrok http 80 --url mydomain.com              # run ngrok with your own custom d
  ngrok http 80 --cidr-allow 2600:8c00::a03c:91ee:fe69:9695/32 # run ngrok with IP policy restric
  Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

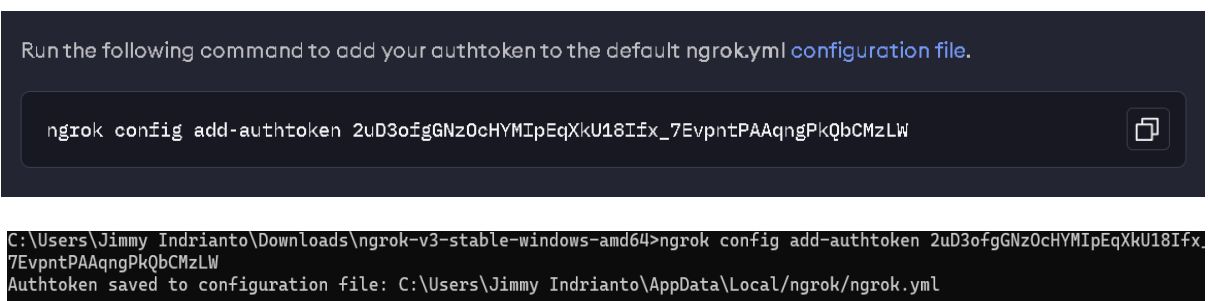
Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Flags:
  -h, --help      help for ngrok

Use "ngrok [command] --help" for more information about a command.

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\Jimmy Indrianto\Downloads\ngrok-v3-stable-windows-amd64>
```

15. Jalankan perintah konfigurasi yang ada pada akun Ngrok di cmd



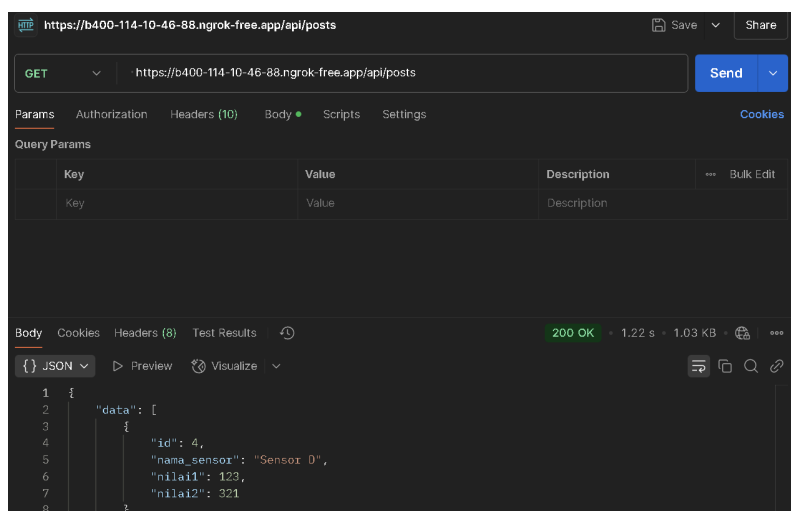
16. Jalankan perintah berikut untuk mengonline-kan laravel melalui port 8000, ngrok http <http://localhost:8000>

```
ngrok
❖ Protect endpoints w/ IP Intelligence: https://ngrok.com/r/ipintel
Session Status      online
Account             Femas Alfaridzi (Plan: Free)
Version             3.20.0
Region              Asia Pacific (ap)
Web Interface        http://127.0.0.1:4040
Forwarding            https://7575-114-10-46-88.ngrok-free.app -> http://localhost:8000

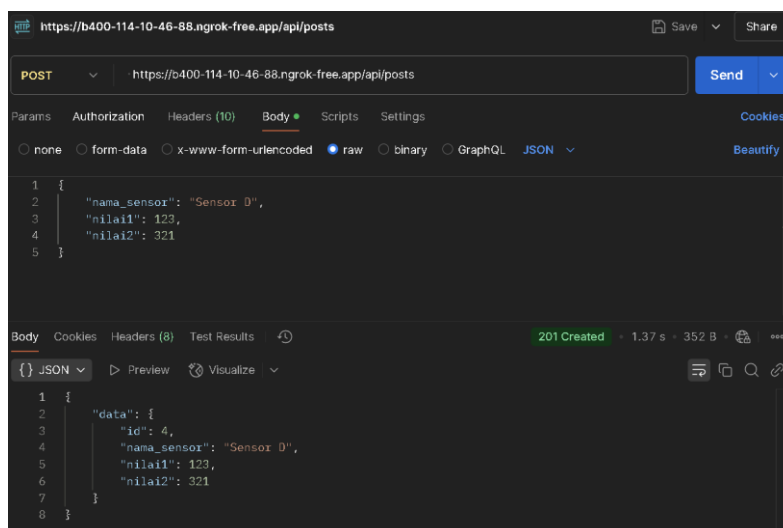
Connections          ttl      opn      rt1      rt5      p50      p90
0                    0        0.00     0.00     0.00     0.00
```

lalu copy link pada *Forwarding* untuk melakukan percobaan pada postman
<https://7575-114-10-46-88.ngrok-free.app>

17. Lakukan percobaan untuk mengambil data dari database menggunakan link yang telah disalin, gunakan *method* GET dan tambahkan *route* /api/posts.



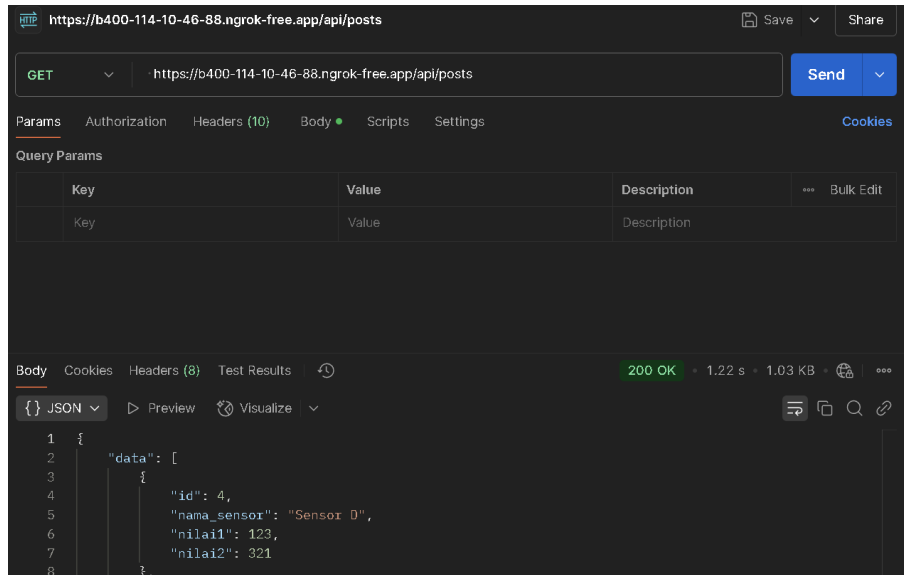
18. Berikutnya adalah melakukan percobaan untuk menambahkan data pada database, gunakan method POST, lalu pada *Body*, pilih *raw* dan format json



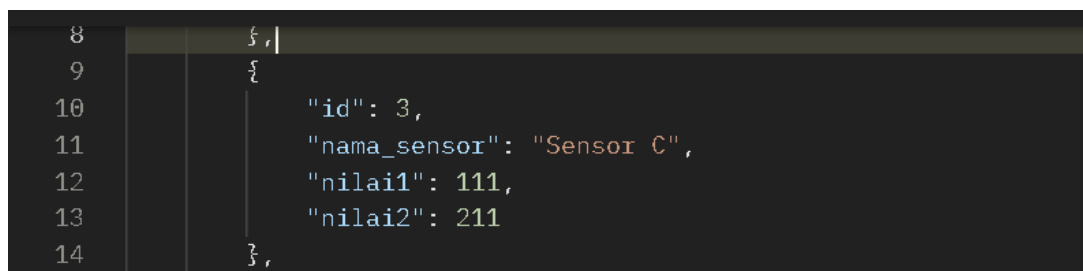
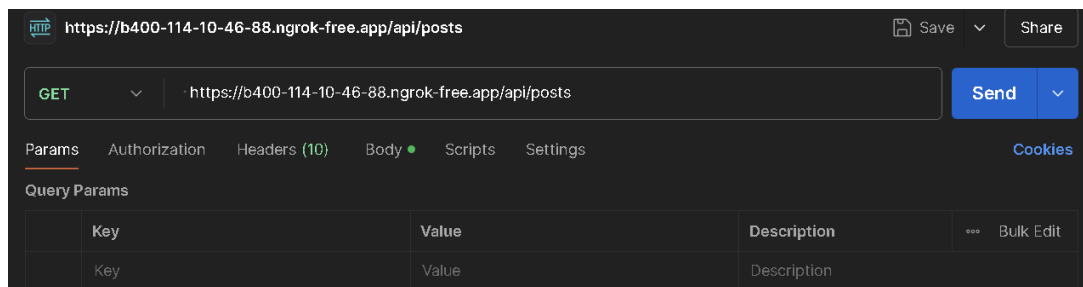
3. Hasil dan Pembahasan

3.1 Hasil Eksperimen

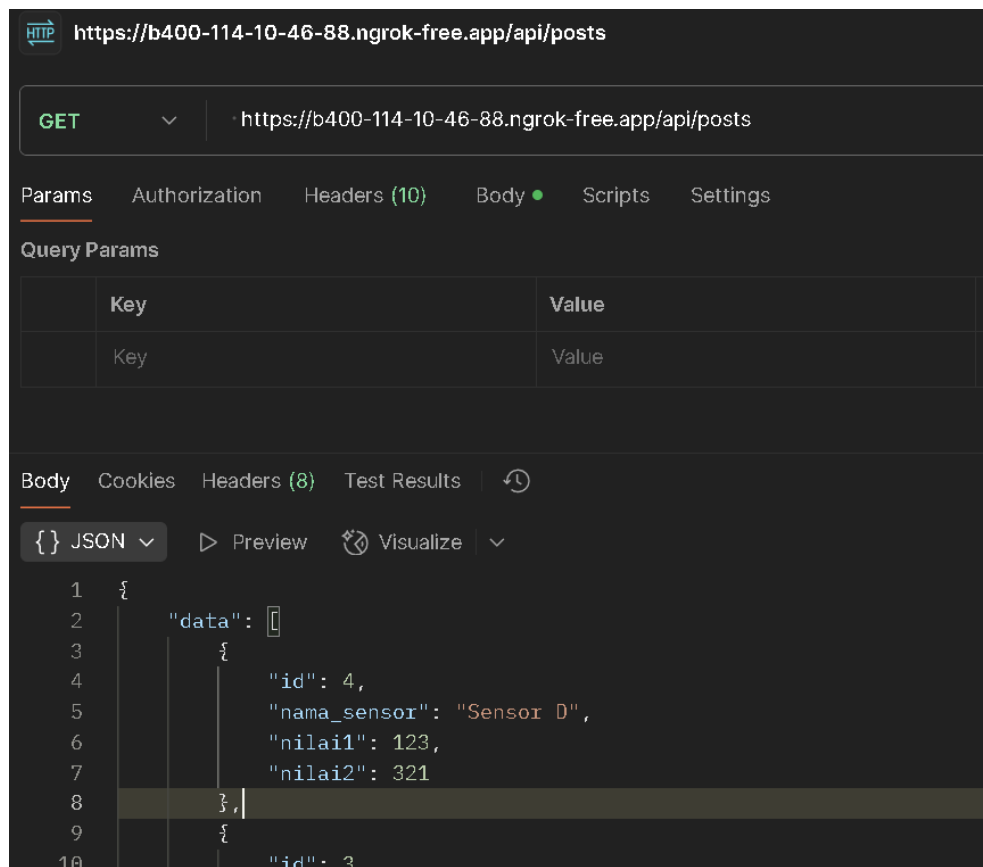
Tampilan ketika berhasil mengambil data dari database menggunakan API:



Tampilan ketika berhasil menambahkan data ke database menggunakan API:



Tampilan ketika berhasil menambahkan data ke database menggunakan API & Ngrok:



4. Lampiran

Tampilan terminal dan cmd ketika API & Ngrok berhasil mengambil dan menambahkan data pada database.

```
2025-03-13 21:42:32 /api/posts ..... ~ 23s
2025-03-13 21:45:17 /api/posts ..... ~ 1s
2025-03-13 21:47:28 /api/posts ..... ~ 1s
```

```
ngrok
♦ Route traffic by anything: https://ngrok.com/r/iep

Session Status      online
Account             Femas Alfaridzi (Plan: Free)
Version             3.20.0
Region              Asia Pacific (ap)
Latency             61ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://b400-114-10-46-88.ngrok-free.app -> http://localhost:8000

Connections         ttl    opn    rt1    rt5    p50    p90
                   2      0      0.00   0.00   1.09   1.15

HTTP Requests
-----
21:47:28.593 +07 GET  /api/posts          200 OK
21:45:17.307 +07 POST /api/posts          201 Created
```