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

**Fakultas Vokasi, Universitas Brawijaya**

**PRAKTIK PEMBUATAN API   
Menggunakan Laravel 11 dan Ngrok**

*Author Bayu Maha Resi*

*Email: bayumaharesii@gmail,com*

**Abstract**

Pemantauan dan pengukuran jarak secara real-time menjadi aspek penting dalam berbagai aplikasi, seperti sistem keamanan, robotika, dan otomasi industri. Eksperimen ini bertujuan untuk mensimulasikan penggunaan ESP32 dalam membaca data dari sensor jarak ultrasonik sebagai bagian dari sistem deteksi dan pengukuran jarak. ESP32 berfungsi sebagai mikrokontroler utama yang mengolah data dari sensor ultrasonik dan menampilkan hasilnya dalam bentuk digital melalui komunikasi serial atau antarmuka berbasis web.Dalam eksperimen ini, dilakukan konfigurasi perangkat keras dan pemrograman ESP32 untuk membaca serta menginterpretasikan data dari sensor jarak ultrasonik. Hasil simulasi menunjukkan bahwa sistem dapat secara akurat mendeteksi dan mengukur jarak secara real-time. Eksperimen ini memberikan wawasan mengenai implementasi sensor jarak dalam sistem berbasis IoT serta bagaimana ESP32 dapat digunakan untuk membangun sistem deteksi yang efisien dan fleksibel.

**Kata kunci**: API, Laravel 11, Ngrok, pengembangan web, integrasi sistem

**1. Introduction**

Dalam pengembangan perangkat lunak modern, Application Programming Interface (API) memainkan peran penting dalam memungkinkan komunikasi antara berbagai sistem dan layanan. API digunakan dalam berbagai aplikasi, mulai dari sistem berbasis web hingga integrasi dengan perangkat IoT. Dengan API, pengembang dapat membangun sistem yang lebih modular, fleksibel, dan mudah diintegrasikan dengan layanan lain.Pada praktik ini, dilakukan simulasi pembuatan API menggunakan Laravel 11 sebagai framework backend yang menyediakan berbagai fitur untuk pengelolaan data dan autentikasi. Selain itu, digunakan Ngrok untuk mengekspose API secara publik tanpa perlu konfigurasi jaringan yang kompleks. Laravel 11 dipilih karena keunggulannya dalam pengelolaan routing, middleware, serta dukungan terhadap standar keamanan modern. Sementara itu, Ngrok memungkinkan pengujian API secara langsung melalui internet, sehingga mempermudah proses debugging dan integrasi dengan aplikasi lain.

Tujuan dari praktik ini adalah untuk memahami cara kerja API, bagaimana membangun dan mengelola endpoint dengan Laravel 11, serta bagaimana menghubungkan API ke layanan eksternal menggunakan Ngrok. Selain itu, praktik ini juga memberikan wawasan tentang pengujian API, implementasi autentikasi, dan praktik terbaik dalam membangun sistem backend yang efisien serta scalable.

* 1. **Latar belakang**

Dalam era digital saat ini, **Application Programming Interface (API)** menjadi komponen fundamental dalam pengembangan perangkat lunak, memungkinkan sistem yang berbeda untuk saling berkomunikasi secara efisien. API digunakan dalam berbagai aplikasi, mulai dari sistem informasi, e-commerce, hingga integrasi layanan berbasis cloud. Dengan API, pengembang dapat membangun arsitektur perangkat lunak yang modular, scalable, dan lebih mudah diintegrasikan dengan layanan pihak ketiga.Laravel 11, sebagai salah satu framework PHP modern, menyediakan berbagai fitur untuk membangun API yang aman dan efisien, seperti **routing yang fleksibel, middleware untuk autentikasi, serta manajemen database yang terstruktur**. Sementara itu, **Ngrok** digunakan untuk mengekspose API secara publik tanpa memerlukan konfigurasi jaringan yang rumit, sehingga mempermudah proses pengujian dan integrasi API dengan aplikasi eksternal.

Dengan kombinasi Laravel 11 dan Ngrok, pengembangan API menjadi lebih cepat dan praktis, memungkinkan pengujian secara real-time dalam lingkungan berbasis cloud. Oleh karena itu, praktik pembuatan API ini bertujuan untuk memberikan pemahaman mengenai bagaimana membangun, mengelola, serta menguji API dengan standar yang baik menggunakan Laravel 11 dan Ngrok sebagai alat bantu utama.

**1.2 Tujuan eksperimen**

Eksperimen ini bertujuan untuk membangun dan menguji API menggunakan Laravel 11 serta mengekspose-nya secara publik dengan Ngrok. Laravel 11 digunakan untuk membuat endpoint API, mengelola routing, dan menerapkan autentikasi, sementara Ngrok mempermudah akses tanpa konfigurasi server yang kompleks.Pengujian dilakukan menggunakan Postman atau cURL untuk memastikan API berfungsi dengan baik. Eksperimen ini memberikan wawasan tentang pengembangan API yang efisien, aman, dan scalable, serta integrasinya dalam sistem berbasis cloud.

**2. Methodology**

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

* Website Wokwi
* Laptop
* Github
* VScode

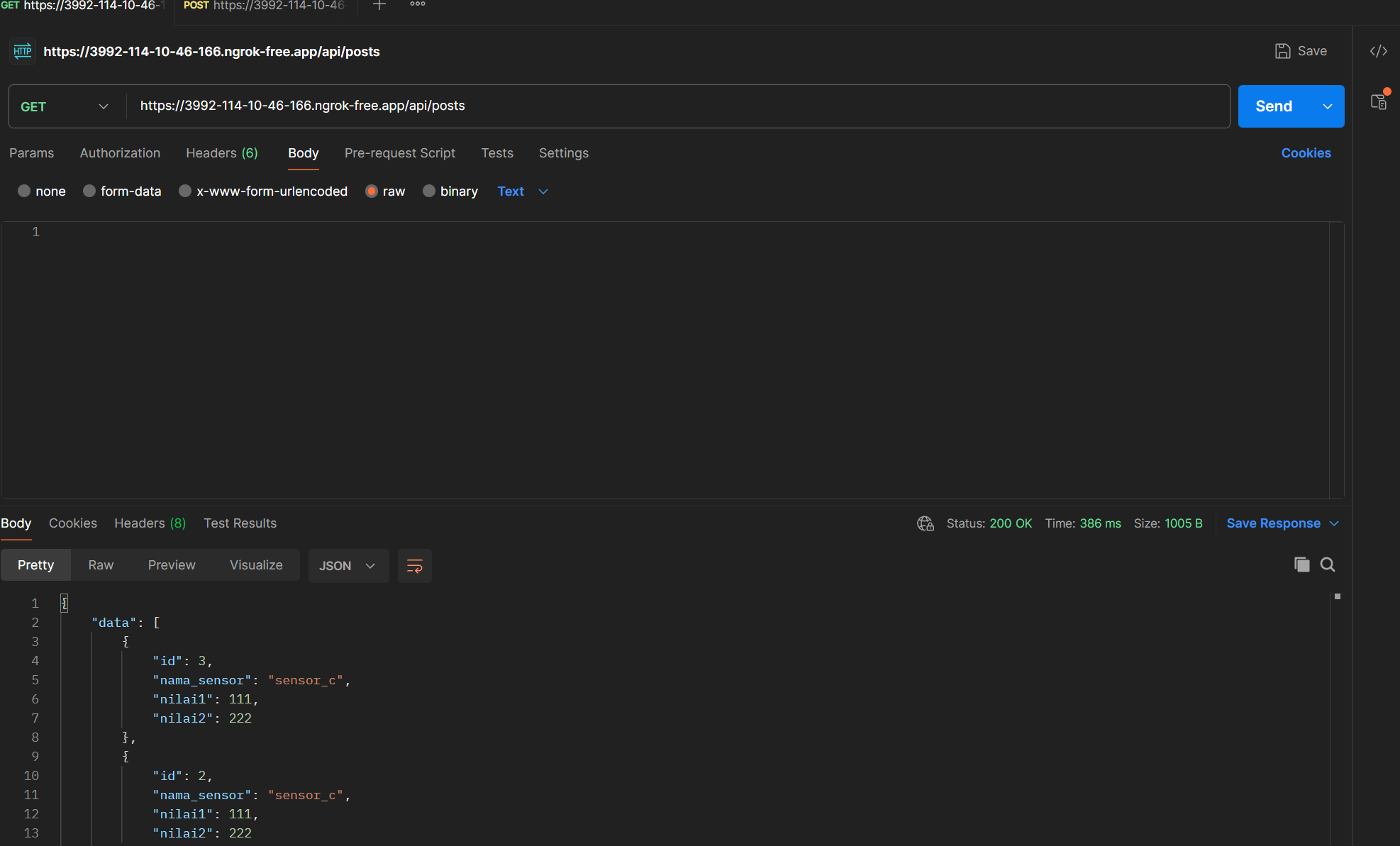
**2.2 Implementation Steps (Langkah Implementasi)**

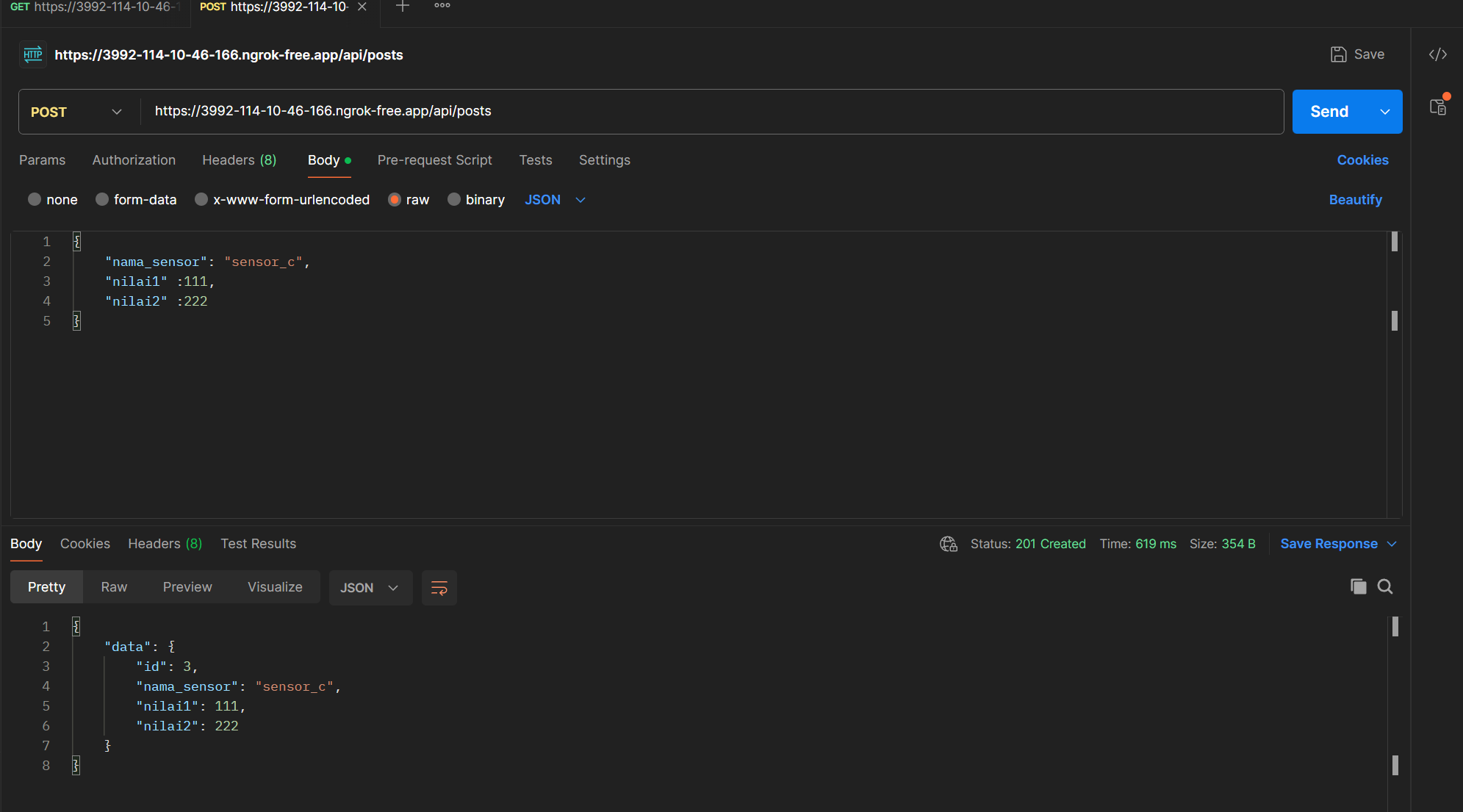
Buat akun wokwi dengan GitHub

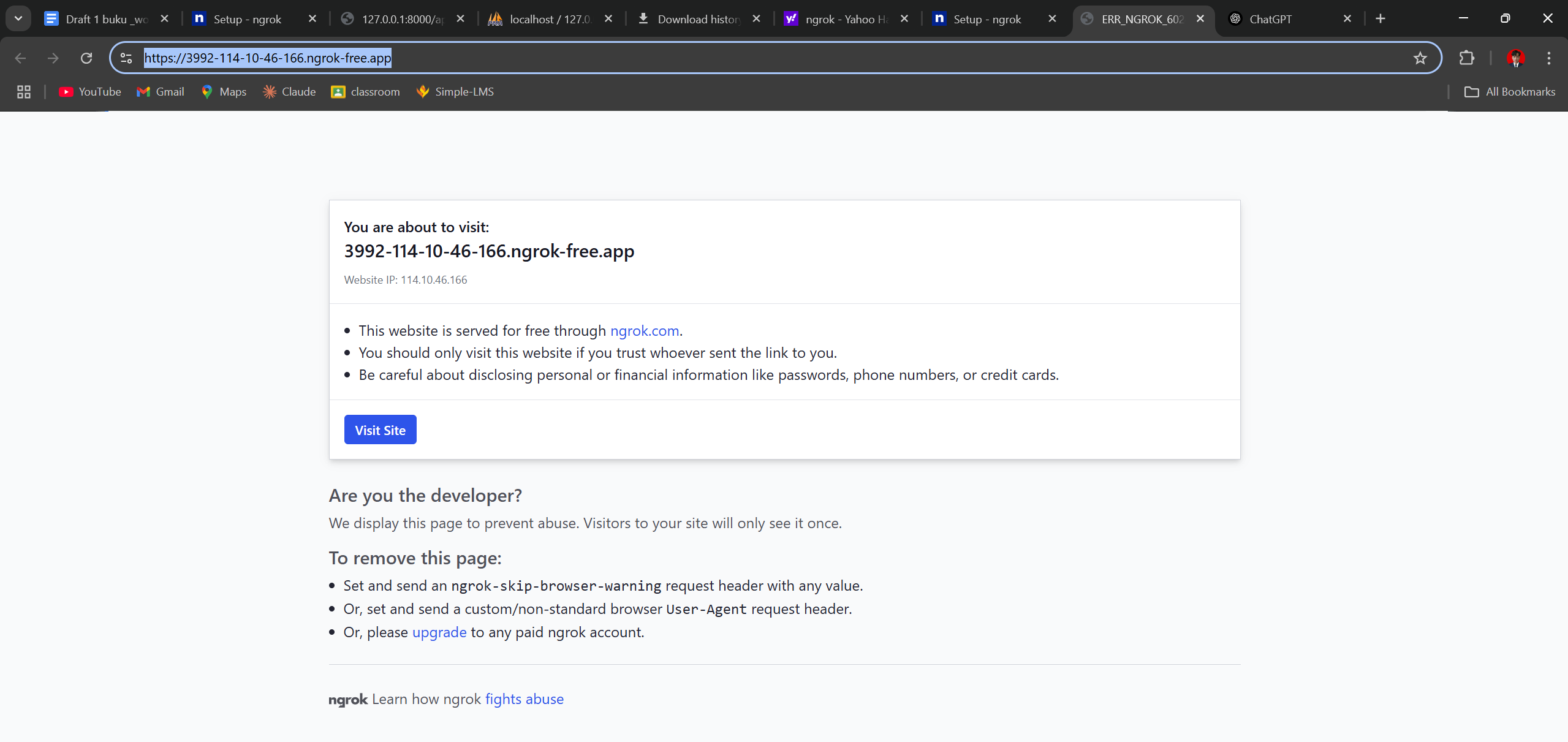
* Membuat topology
* Memindah diagram.json di Wokwi kedalam Vscode
* Memindahkan code dari wokwi ke Vs Code
* Lakukan configurasi ulang
* Tambahkan code pada file platform io ini

**3. Results and Discussion**

**3.1 Experimental Results**







**4. Appendix**

**Code Program:**

**Transakasi Controler sensor.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);

}

}

**Transaksi sensor Resource.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,

        ];

    }

}

**Transaksisensor.php**<?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 = [];

}

**2025\_03\_09\_090727\_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); *// Menetapkan AUTO\_INCREMENT dimulai dari 1*

            $table->string('nama\_sensor', 255); *// varchar(255)*

            $table->integer('nilai1', false)->length(255); *// int(255)*

            $table->integer('nilai2', false)->length(255); *// int(255)*

            $table->timestamps(); *// Menambahkan created\_at dan updated\_at*

        });

    }

*/\*\**

*\* Reverse the migrations.*

*\*/*

    public function down(): void

    {

        Schema::dropIfExists('transaksi\_sensors');

    }

};

**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);