**LAPORAN PRAKTIKUM**

**INTERNET OF THINGS**



**Disusun oleh :**

Atika Fitria Arifiana (233140700111064)

**PRODI D-III TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI**

**UNIVERSITAS BRAWIJAYA**

**2025**

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktik Simulasi Relay, Button & LED**

*Atika Fitria Arifiana*

*Fakultas Vokasi, Universitas Brawijaya*

*Email :* [*atikafit.arifiana@gmail.com*](mailto:atikafit.arifiana@gmail.com)

**ABSTRACT**

Bab ini membahas implementasi simulasi sederhana menggunakan Arduino dengan tiga komponen utama: relay, tombol (button), dan LED. Tujuan dari eksperimen ini adalah memahami cara kerja input dan output digital pada mikrokontroler, khususnya bagaimana sebuah tombol dapat mengontrol LED dan relay. Dengan menggunakan konsep pull-up resistor internal, tombol dapat mendeteksi perubahan kondisi dari HIGH ke LOW dan memicu aksi tertentu. Implementasi kode dijelaskan secara rinci untuk memastikan bahwa sistem dapat bekerja dengan stabil dan responsif.

*Keywords: Arduino, relay, button, LED, pull-up resistor, input digital, output digital,* simulasi, *mikrokontroler*

**PENDAHULUAN**

* 1. **Latar belakang**

Dalam dunia otomasi dan sistem kendali, relay digunakan sebagai saklar elektronik untuk mengontrol perangkat dengan arus tinggi menggunakan sinyal berdaya rendah. Button dan LED juga merupakan komponen dasar yang sering digunakan dalam sistem berbasis mikrokontroler untuk memberikan input dan indikator visual. Praktik simulasi ini bertujuan untuk memahami cara kerja relay, button, dan LED dalam sebuah sistem yang dikendalikan oleh Arduino.

* 1. **Tujuan eksperimen**

1. Mempelajari cara kerja relay sebagai saklar elektronik dalam rangkaian berbasis Arduino.
2. Mengimplementasikan button sebagai input untuk mengontrol relay dan LED.
3. Mengembangkan pemahaman tentang penggunaan pull-up resistor untuk membaca input digital.
4. Menyimulasikan dan mengamati respons sistem terhadap input button.

**METODOLOGI**

**2.1 Alat dan Bahan**

1. Arduino Uno
2. Relay 5V
3. LED (Light Emitting Diode)
4. Resistor 220Ω dan 10kΩ
5. Button (Push Button)
6. Breadboard
7. Kabel jumper
8. Software Arduino IDE

**2.2 Langkah Implementasi**

Proses implementasi eksperimen ini dilakukan dalam beberapa tahapan sebagai berikut:

1. **Persiapan Rangkaian**

* Sambungkan LED dengan resistor 220Ω ke pin digital Arduino.
* Hubungkan relay ke Arduino dan pastikan input daya sesuai dengan spesifikasi.
* Sambungkan button dengan pull-up resistor 10kΩ agar dapat membaca input dengan stabil.

1. **Pemrograman Arduino**

* Tulis kode untuk membaca input dari button.
* Implementasikan logika untuk menyalakan dan mematikan relay berdasarkan input button.
* Tambahkan LED sebagai indikator status relay.

1. **Pengujian dan Analisis**

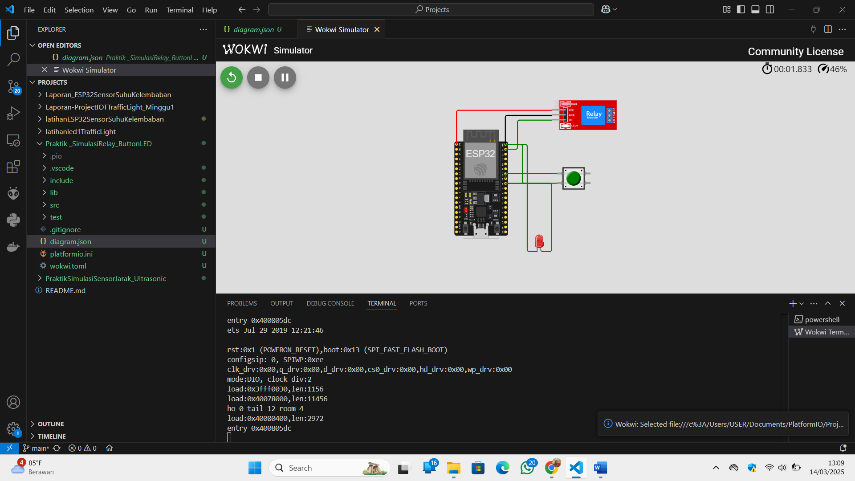
* Upload program ke Arduino dan jalankan simulasi.
* Amati perubahan status relay dan LED saat button ditekan.
* Evaluasi respons sistem dan pastikan fungsinya sesuai dengan ekspektasi.

**HASIL DAN PEMBAHASAN**

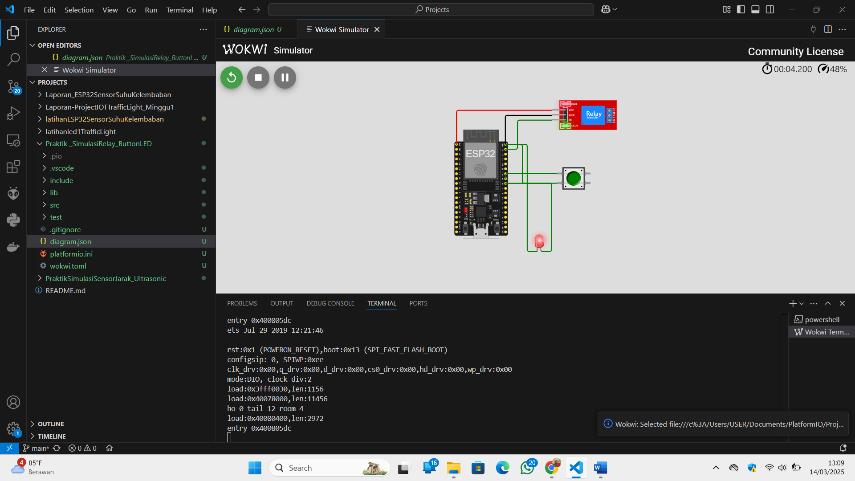
**3.1 Hasil Eksperimen**

Setelah menjalankan simulasi, hasil yang diperoleh menunjukkan bahwa:

* Relay dapat dikendalikan dengan input dari button menggunakan Arduino.
* LED menyala dan mati sesuai dengan kondisi relay, menandakan sistem berjalan dengan baik.
* Pull-up resistor membantu memastikan pembacaan input button yang stabil tanpa noise.
* Simulasi ini memberikan pemahaman dasar tentang integrasi relay, button, dan LED dalam sistem kendali berbasis mikrokontroler.



**Gambar 3.1 Hasil Simulasi di Wokwi**



**Gambar 3.2 Hasil Simulasi di Wokwi**

**LAMPIRAN**

**4.1 Kode Program ESP32 untuk Simulasi Relay, Button & LED**

#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

  }

}

**4.2 Diagram Skematik dalam Wokwi**

{

    "version": 1,

    "author": "Atika Fitria Arifiana",

    "editor": "wokwi",

    "parts": [

      { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },

      { "type": "wokwi-led", "id": "led1", "top": 198, "left": 147.8, "attrs": { "color": "red" } },

      {

        "type": "wokwi-pushbutton",

        "id": "btn1",

        "top": 73.4,

        "left": 201.6,

        "attrs": { "color": "green", "xray": "1" }

      },

      { "type": "wokwi-relay-module", "id": "relay1", "top": -57.4, "left": 192, "attrs": {} }

    ],

    "connections": [

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

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

      [ "relay1:VCC", "esp:3V3", "red", [ "h0" ] ],

      [ "relay1:GND", "esp:GND.2", "black", [ "h0" ] ],

      [ "btn1:1.l", "esp:19", "green", [ "h0" ] ],

      [ "btn1:2.l", "esp:GND.2", "green", [ "h-67.2", "v-86.2" ] ],

      [ "relay1:IN", "esp:23", "green", [ "h-67.2", "v19" ] ],

      [ "led1:C", "esp:GND.2", "green", [ "v0", "h-18.8", "v-163.2" ] ],

      [ "led1:A", "esp:18", "green", [ "v0", "h19.2", "v-38.4" ] ]

    ],

    "dependencies": {}

  }

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktik Simulasi Sensor Jarak (Ultrasonic)**

*Atika Fitria Arifiana*

*Fakultas Vokasi, Universitas Brawijaya*

*Email :* [*atikafit.arifiana@gmail.com*](mailto:atikafit.arifiana@gmail.com)

**ABSTRACT**

Pada bab ini, dilakukan praktik simulasi sensor jarak menggunakan sensor ultrasonik HC-SR04 pada Arduino. Sensor ini bekerja dengan memancarkan gelombang ultrasonik dan mengukur waktu pantulannya untuk menentukan jarak suatu objek. Eksperimen ini bertujuan untuk memahami prinsip kerja sensor ultrasonik, mengimplementasikan kode pemrograman untuk membaca jarak, serta menganalisis akurasi hasil pengukuran. Hasil eksperimen menunjukkan bahwa sensor ultrasonik dapat mendeteksi objek dengan cukup akurat dalam berbagai kondisi. Data hasil pengukuran ditampilkan dalam satuan sentimeter pada Serial Monitor Arduino, yang memungkinkan pemantauan jarak secara real-time.

*Keywords:* Sensor ultrasonik*, HC-SR04, Arduino,* pengukuran jarak, pulseIn(), kecepatan suara.

**PENDAHULUAN**

* 1. **Latar belakang**

Sensor ultrasonik merupakan salah satu komponen penting dalam sistem keamanan dan otomatisasi rumah pintar. Sensor ini bekerja dengan memancarkan gelombang ultrasonik dan mengukur waktu pantulannya untuk menghitung jarak suatu objek. Praktik ini bertujuan untuk memahami prinsip kerja sensor ultrasonik dan mengimplementasikannya dalam simulasi menggunakan Arduino.

* 1. **Tujuan eksperimen**

1. Memahami cara kerja sensor ultrasonik dalam mengukur jarak.
2. Mengimplementasikan sensor ultrasonik pada Arduino untuk mendeteksi objek di sekitarnya.
3. Menampilkan hasil pengukuran jarak dalam satuan cm pada Serial Monitor.
4. Menguji akurasi pengukuran sensor dalam berbagai kondisi.

**METODOLOGI**

**2.1 Alat dan Bahan**

1. Arduino Uno
2. Sensor Ultrasonic HC-SR04
3. Resistor (opsional)
4. Breadboard
5. Kabel jumper
6. Software Arduino IDE

**2.2 Langkah Implementasi**

Proses implementasi eksperimen ini dilakukan dalam beberapa tahapan sebagai berikut:

1. **Persiapan Rangkaian**

* Hubungkan **trigPin** sensor ke pin digital **5** pada Arduino.
* Hubungkan **echoPin** sensor ke pin digital **18** pada Arduino.
* Pastikan koneksi **VCC** dan **GND** sensor tersambung dengan benar.

1. **Pemrograman Arduino**

* Konfigurasi **trigPin** sebagai OUTPUT dan **echoPin** sebagai INPUT.
* Kirim sinyal **trigger** selama 10 mikrodetik untuk memancarkan gelombang ultrasonik.
* Hitung waktu pantulan gelombang dengan fungsi **pulseIn()** untuk mendapatkan jarak objek.
* Konversi hasil pengukuran ke satuan cm dan tampilkan di **Serial Monitor**.

1. **Pengujian dan Analisis**

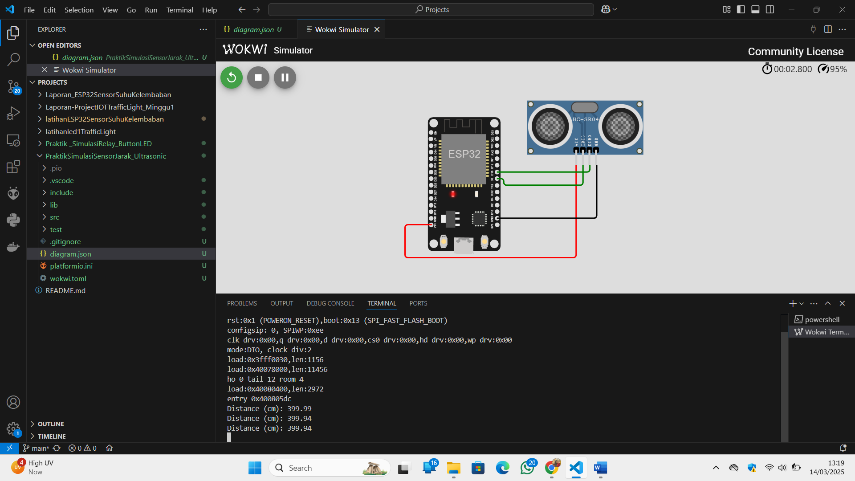
* Upload program ke Arduino dan jalankan simulasi.
* Letakkan objek di depan sensor dan amati perubahan jarak yang ditampilkan.
* Bandingkan hasil pengukuran dengan pengukuran manual untuk melihat akurasi sensor.

**HASIL DAN PEMBAHASAN**

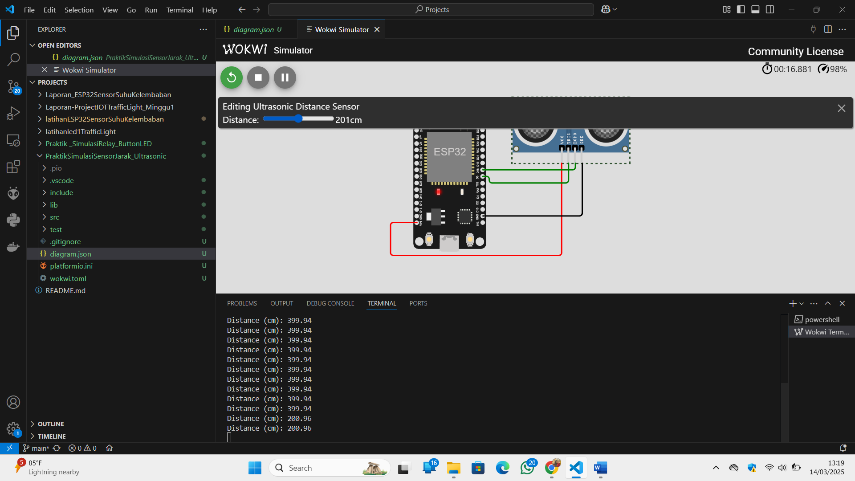
**3.1 Hasil Eksperimen**

Setelah menjalankan simulasi, hasil yang diperoleh menunjukkan bahwa:

* Sensor ultrasonik berhasil mengukur jarak objek dengan cukup akurat.
* Nilai jarak dalam **cm** berhasil ditampilkan di **Serial Monitor**.
* Perubahan jarak objek dapat langsung terdeteksi secara real-time.
* Sensor bekerja optimal dalam kondisi lingkungan dengan sedikit gangguan suara atau hambatan lainnya.



**Gambar 3.1 Hasil Simulasi di Wokwi**



**Gambar 3.2 Hasil Simulasi di Wokwi**

**LAMPIRAN**

**4.1 Kode Program ESP32 untuk Simulasi Sensor Jarak (Ultrasonic)**

#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);

}

**4.2 Diagram Skematik dalam Wokwi**

{

    "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**

*Atika Fitria Arifiana*

*Fakultas Vokasi, Universitas Brawijaya*

*Email :* [*atikafit.arifiana@gmail.com*](mailto:atikafit.arifiana@gmail.com)

**ABSTRACT**

Pada bab ini, dilakukan praktik simulasi sensor jarak menggunakan sensor ultrasonik HC-SR04 pada Arduino. Sensor ini bekerja dengan memancarkan gelombang ultrasonik dan mengukur waktu pantulannya untuk menentukan jarak suatu objek. Eksperimen ini bertujuan untuk memahami prinsip kerja sensor ultrasonik, mengimplementasikan kode pemrograman untuk membaca jarak, serta menganalisis akurasi hasil pengukuran. Hasil eksperimen menunjukkan bahwa sensor ultrasonik dapat mendeteksi objek dengan cukup akurat dalam berbagai kondisi. Data hasil pengukuran ditampilkan dalam satuan sentimeter pada Serial Monitor Arduino, yang memungkinkan pemantauan jarak secara real-time.

*Keywords:* Sensor ultrasonik*, HC-SR04, Arduino,* pengukuran jarak, pulseIn(), kecepatan suara.

**PENDAHULUAN**

* 1. **Latar belakang**

Dalam pengembangan sistem berbasis Internet of Things (IoT), komunikasi antara perangkat keras dan perangkat lunak menjadi aspek yang krusial. Salah satu metode yang umum digunakan untuk menghubungkan perangkat IoT dengan sistem backend adalah melalui Application Programming Interface (API). API memungkinkan pertukaran data antara sensor dan server secara efisien.

Laravel, sebagai salah satu framework PHP yang populer, menyediakan berbagai fitur untuk membangun API yang cepat, aman, dan mudah diimplementasikan. Ditambah dengan Ngrok, API yang dikembangkan dapat diakses secara publik tanpa perlu konfigurasi server yang kompleks. Oleh karena itu, pada bab ini akan dibahas praktik pembuatan API menggunakan Laravel 11 dan penerapannya untuk komunikasi dengan perangkat IoT.

* 1. **Tujuan eksperimen**
  2. Membangun API menggunakan Laravel 11 untuk mengelola data transaksi sensor.
  3. Menghubungkan API dengan database MySQL untuk penyimpanan dan pengambilan data sensor.
  4. Menguji API menggunakan Postman sebelum diintegrasikan dengan perangkat IoT.
  5. Mengonline-kan API menggunakan Ngrok agar dapat diakses dari jaringan publik.

**METODOLOGI**

**2.1 Alat dan Bahan**

1. Laptop/PC dengan sistem operasi Windows/Linux/MacOS
2. Composer untuk instalasi Laravel
3. PHP versi 8.x atau lebih tinggi
4. Laravel 11 sebagai framework pengembangan API
5. Postman untuk pengujian API
6. XAMPP atau MySQL Server sebagai database backend
7. Ngrok untuk membuka akses API secara publik

**2.2 Langkah Implementasi**

Proses implementasi eksperimen ini dilakukan dalam beberapa tahapan sebagai berikut:

* 1. **Instalasi Laravel 11**
* Menggunakan perintah composer create-project --prefer-dist laravel/laravel:^11.0 laravel-11
* Mengatur database MySQL dengan nama iot\_25 dan mengonfigurasi file .env
  1. **Pembuatan Model dan Migrasi Database**
* Membuat model TransaksiSensor dengan php artisan make:model TransaksiSensor -m
* Menyesuaikan skema tabel di file migrasi
  1. **Pembuatan API Controller dan Resource**
* Membuat API controller php artisan make:controller Api/TransaksiSensorController
* Mengatur fungsi CRUD untuk mengelola data sensor
  1. **Pengujian API dengan Postman**
* Menjalankan server Laravel dengan php artisan serve
* Menggunakan **Postman** untuk menguji endpoint **GET, POST, PUT, dan DELETE**
  1. **Mengonline-kan API dengan Ngrok**
* Menjalankan **Ngrok** menggunakan perintah ngrok http http://localhost:8000
* Menggunakan URL publik yang diberikan oleh Ngrok untuk mengakses API

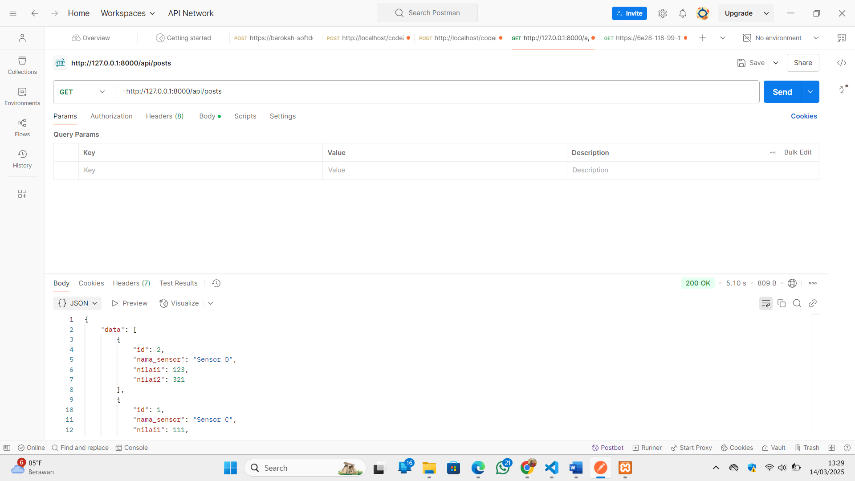
**HASIL DAN PEMBAHASAN**

**3.1 Hasil Eksperimen**

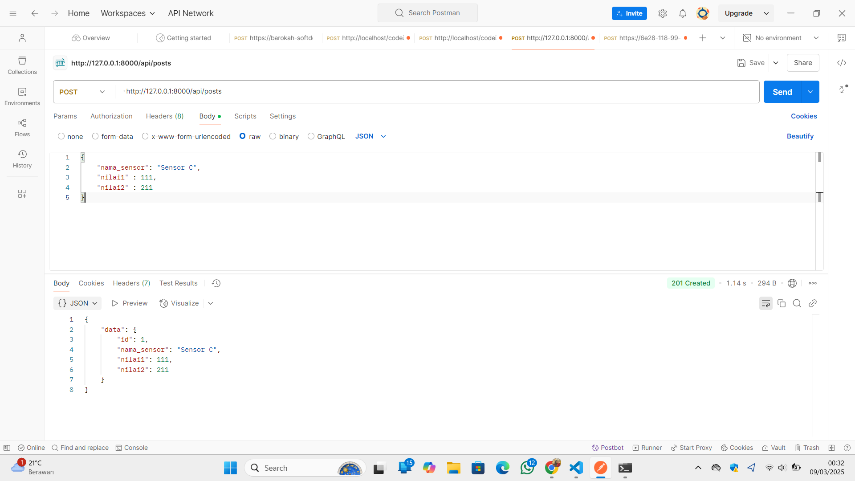
Hasil dari eksperimen ini menunjukkan bahwa API yang dikembangkan dapat berjalan dengan baik dan berfungsi untuk:

* Mengambil data dari database melalui endpoint GET /api/posts
* Menyimpan data baru ke dalam database menggunakan metode POST
* Memperbarui dan menghapus data sensor dengan PUT dan DELETE
* API dapat diakses melalui URL publik yang dihasilkan oleh **Ngrok**, memungkinkan perangkat IoT untuk berkomunikasi dengan server

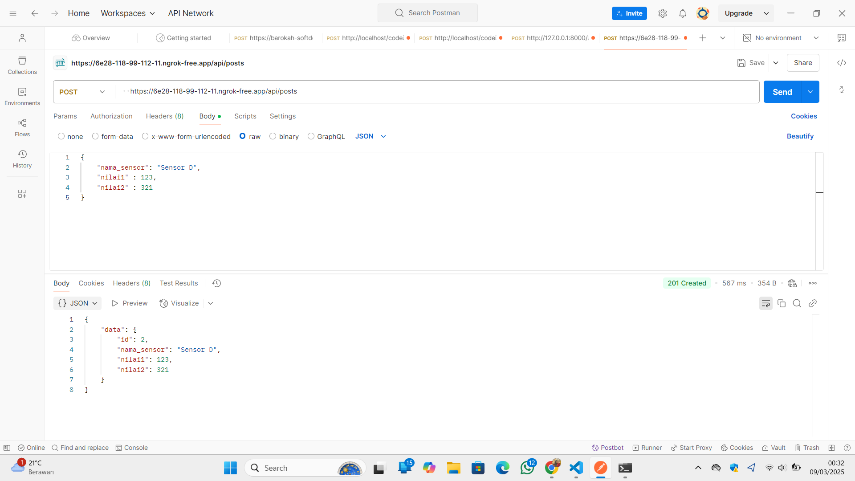
Dengan hasil ini, API yang telah dibuat dapat digunakan sebagai backend untuk sistem **IoT**, memungkinkan perangkat keras seperti **ESP32** untuk mengirim dan menerima data secara real-time melalui internet.



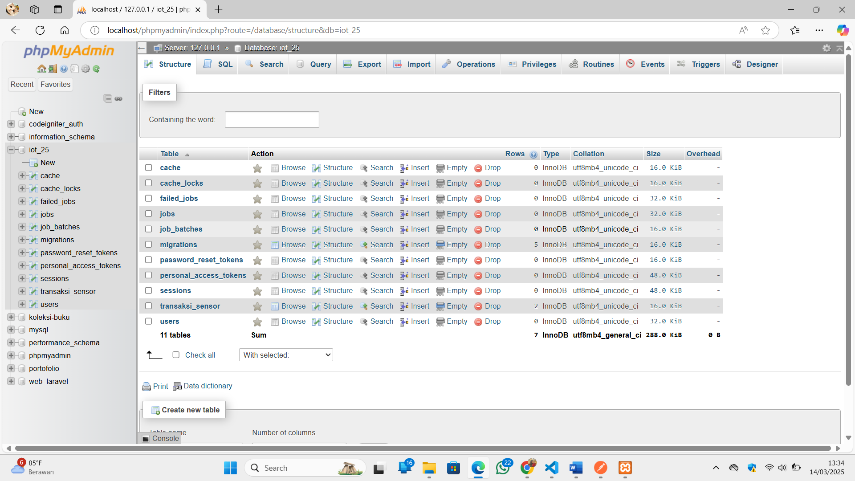
**Gambar 3.1 Postman GET http://127.0.0.1:8000/api/posts**



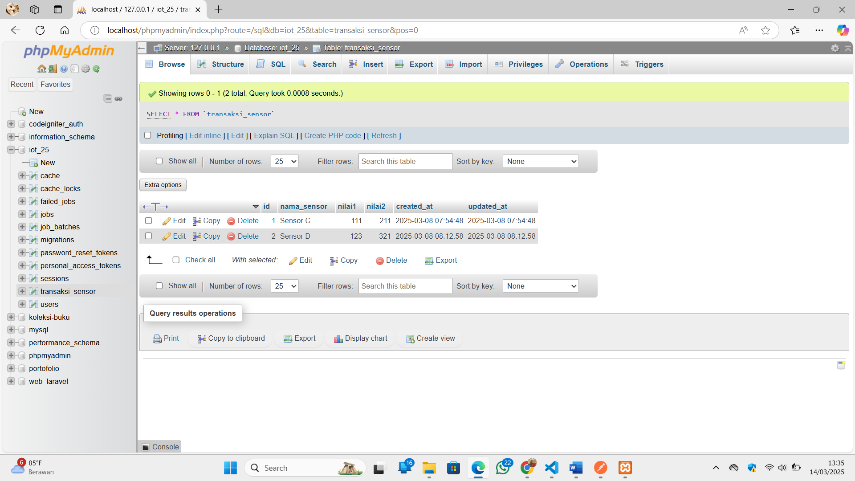
**Gambar 3.2 Postman POST http://127.0.0.1:8000/api/posts**



**Gambar 3.3 Postman POST https://6e28-118-99-112-11.ngrok-free.app/api/posts**



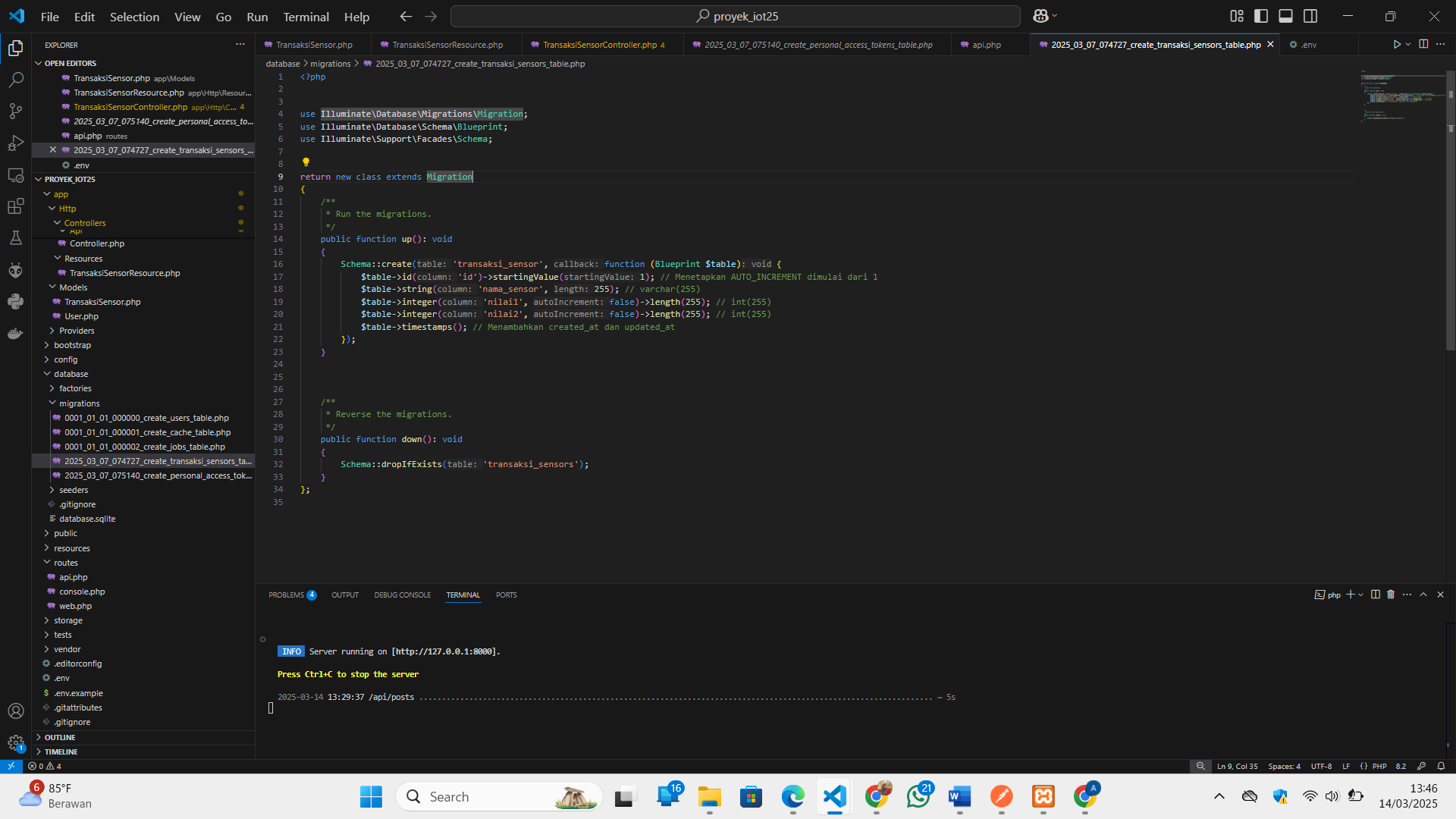
**Gambar 3.4 Database iot\_25**



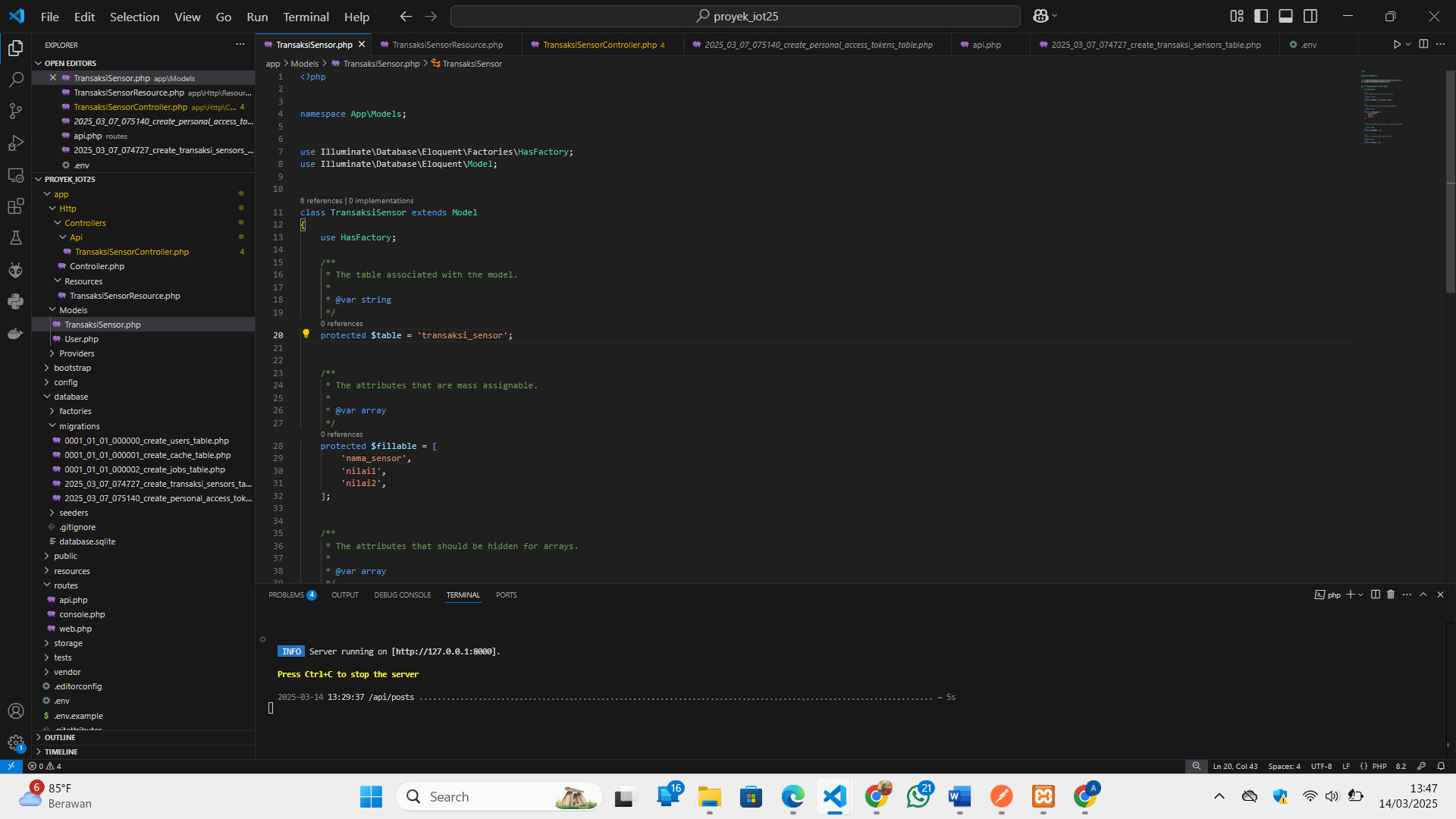
**Gambar 3.5 Transaksi Sensor di Database iot\_25**

**LAMPIRAN**

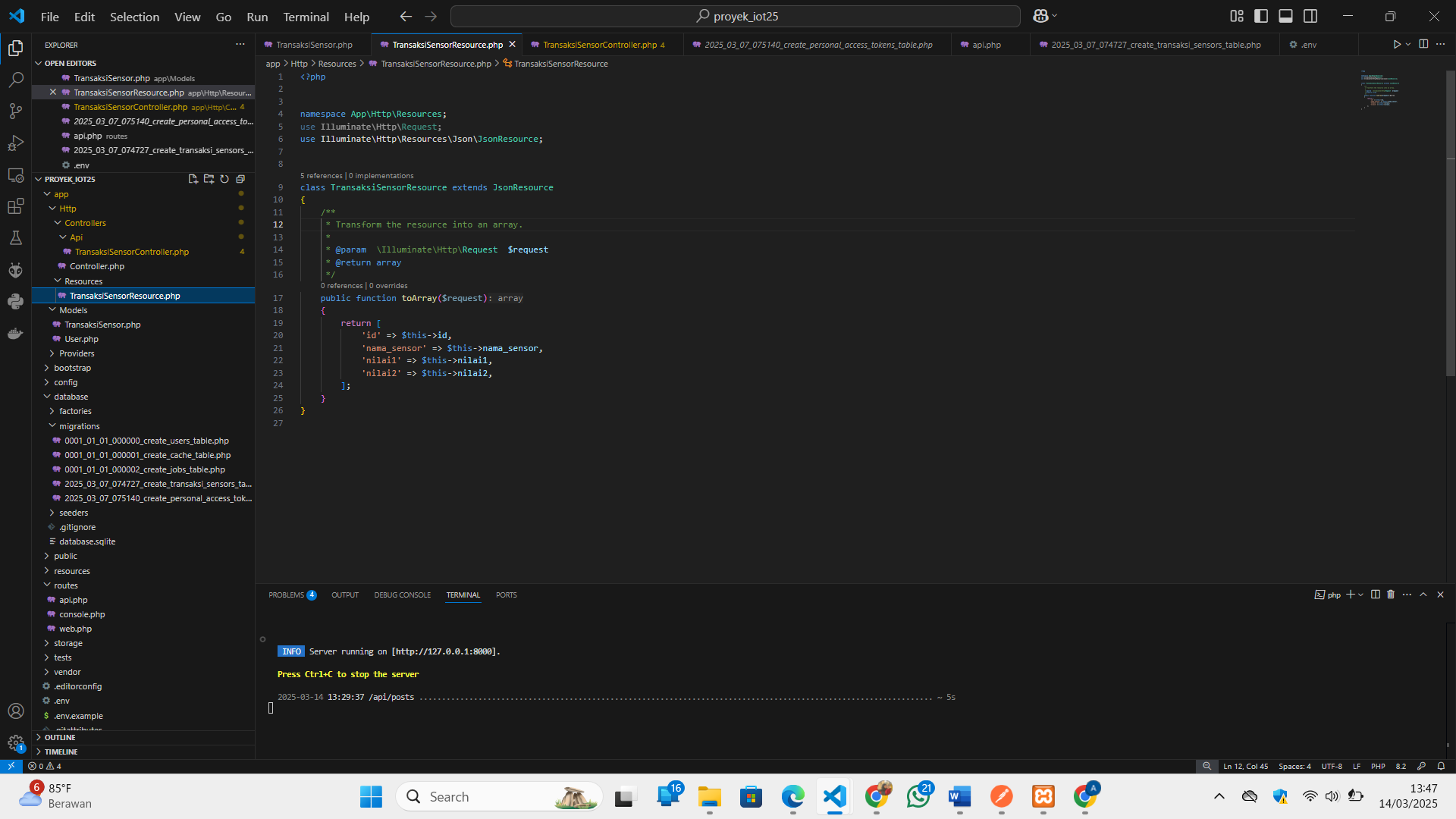
**4.1 Kode pemrograman 2025\_03\_07\_153007\_create\_transaksi\_sensors\_table**



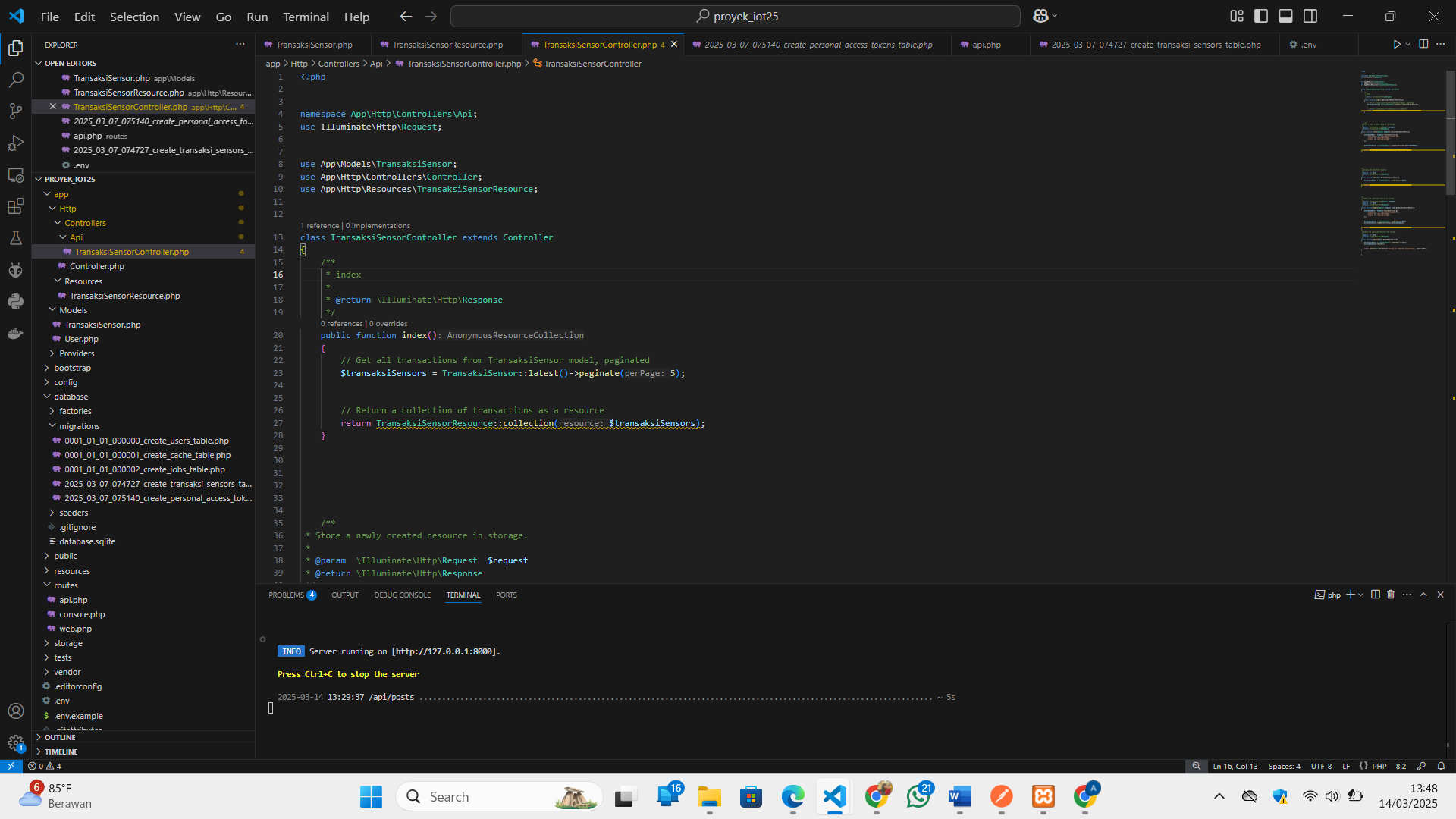
**4.2 Kode pemrograman app/Models/TransaksiSensor.php**



**4.3 Kode pemrograman TransaksiSensorResource.php**



**4.4 Kode pemrograman app/Http/Controllers/Api/TransaksiSensorController.php**



**4.5 Kode pemrograman routes/api.php**

