**Laporan Minggu 7 Praktikum Wokwi**

**Akses API Melalui   
Simulasi WOKWI**

*Gunawan Danyarto*

*University of Brawijaya Vocation Faculty*

*Email : [gunawandany11@gmail.com](mailto:gunawandany11@g,ail.com)*

**Abstract**

This experiment aims to simulate API access through WOKWI, a platform for microcontroller simulation. The study involves setting up a microcontroller (ESP32) in WOKWI, configuring it to send HTTP requests to an external API, and validating responses. The integration of IoT devices with APIs is essential for real-world applications, enabling seamless data communication. This report covers the configuration, API request implementation, and results of the simulation.

**1. Introduction**

**1.1 Background of the IoT Experiment**

IoT (Internet of Things) systems rely on APIs to communicate with servers, process data, and execute remote commands. WOKWI provides a virtual environment to develop and test microcontroller-based applications without physical hardware. This experiment demonstrates how an ESP32 in WOKWI can interact with an API, allowing developers to simulate real-world IoT implementations efficiently.

**1.2 Objective of the Experiment**

1. Configure WOKWI to simulate an ESP32 microcontroller.

2. Establish an HTTP connection from ESP32 to an API endpoint.

3. Send and receive data using HTTP GET and POST methods.

4. Analyze response times and validate API communication.

**2. Methodology**

### 2.1 Tools & Materials

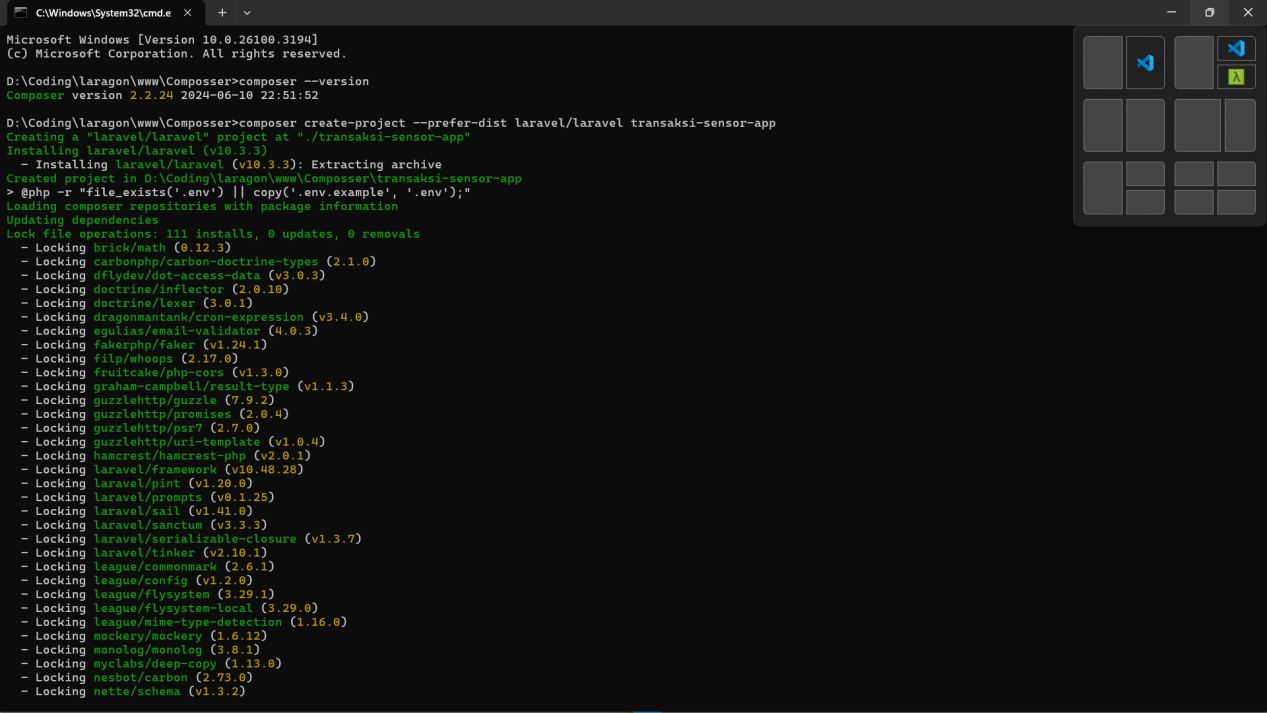
**Simulator**: WOKWI (<https://wokwi.com/>)

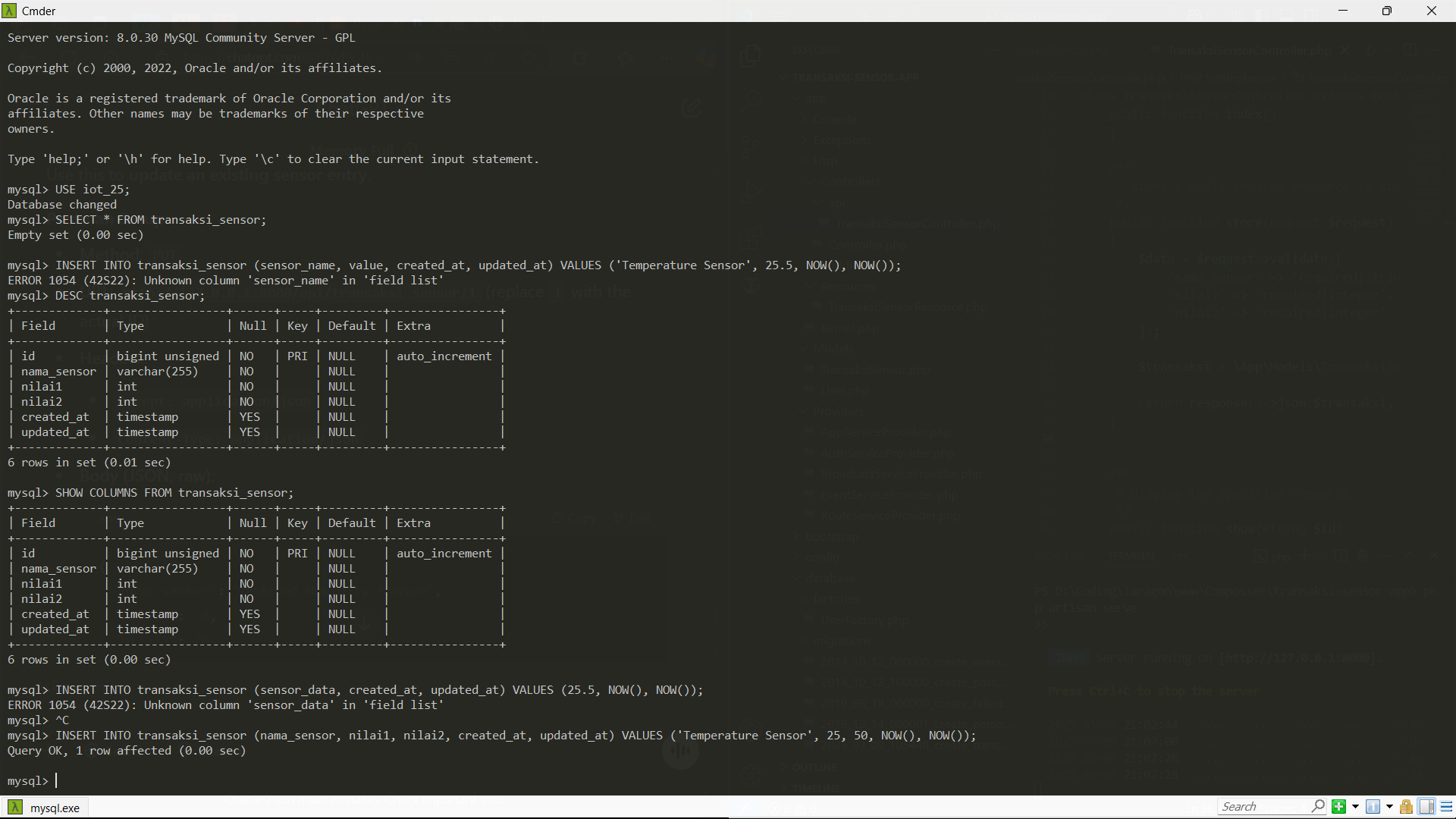
**Microcontroller**: ESP32 (simulated)

**Programming Language**: C++ (Arduino framework)

**API**: Public REST API (JSONPlaceholder) or a custom API

**Tools**: Arduino IDE, Postman (for API testing)





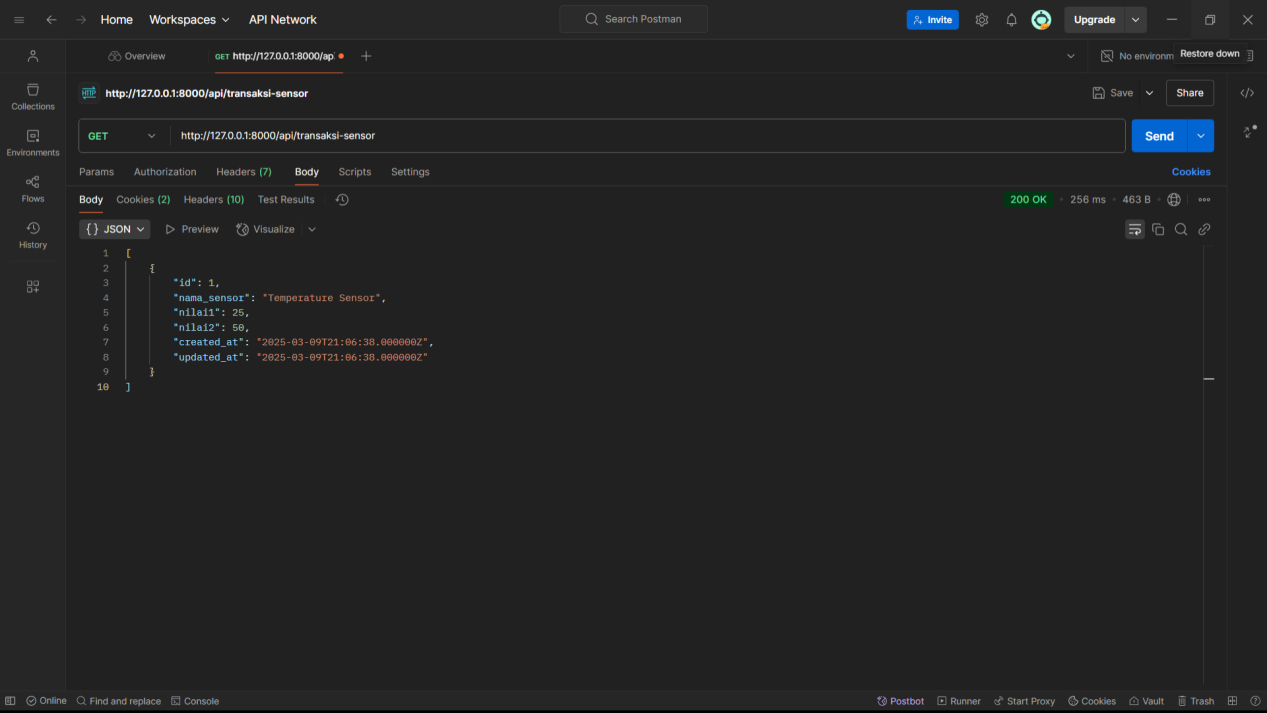
**2.2 Implementation Steps**

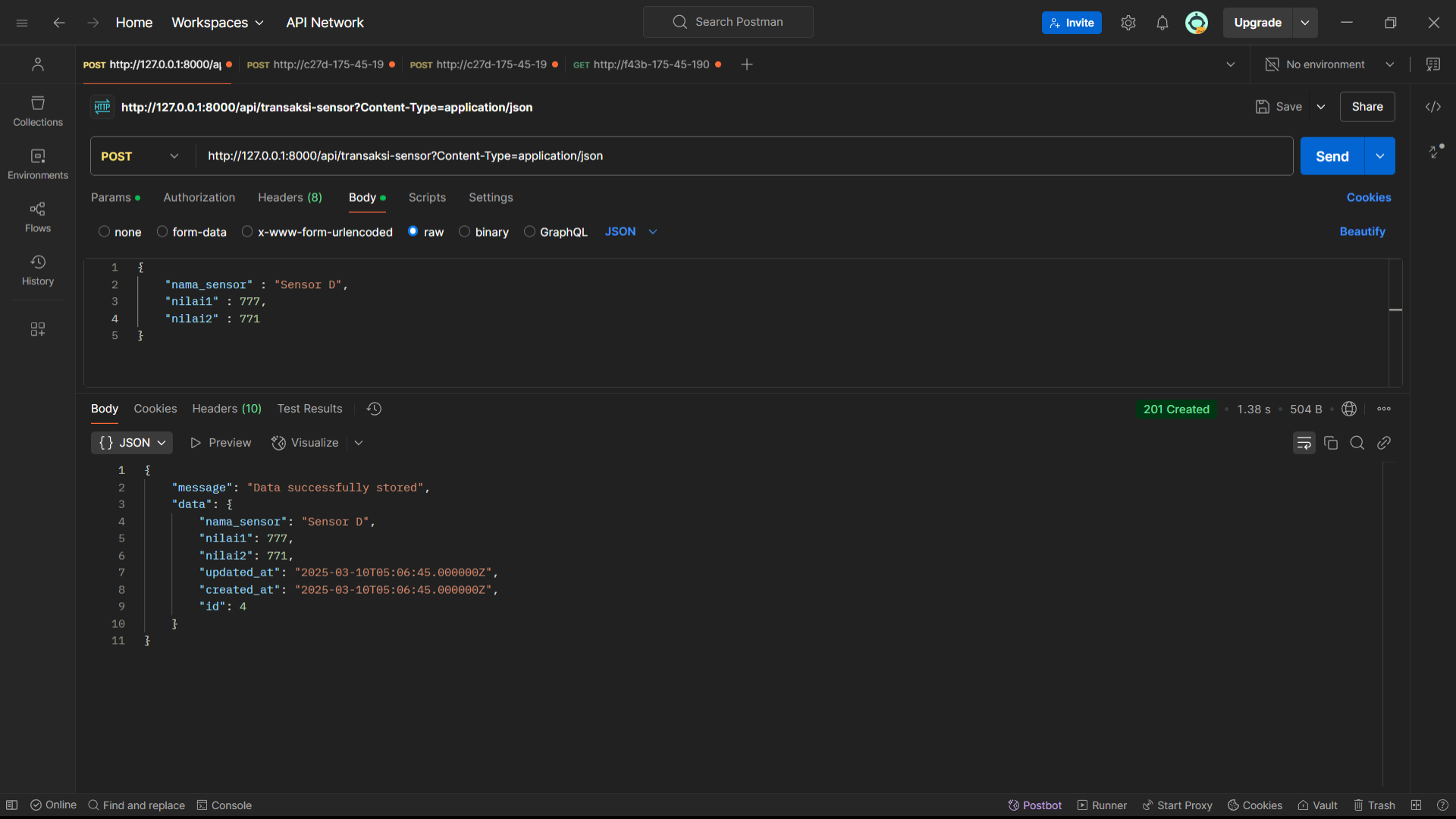
#### ****2.2.1 Setting Up ESP32 in WOKWI****

1. Open WOKWI and create a new ESP32 project.

2. Configure the necessary peripherals (Wi-Fi, sensors if needed).

3. Install the required libraries for HTTP communication (WiFiClientSecure.h)





#### 2.2.2 API Development

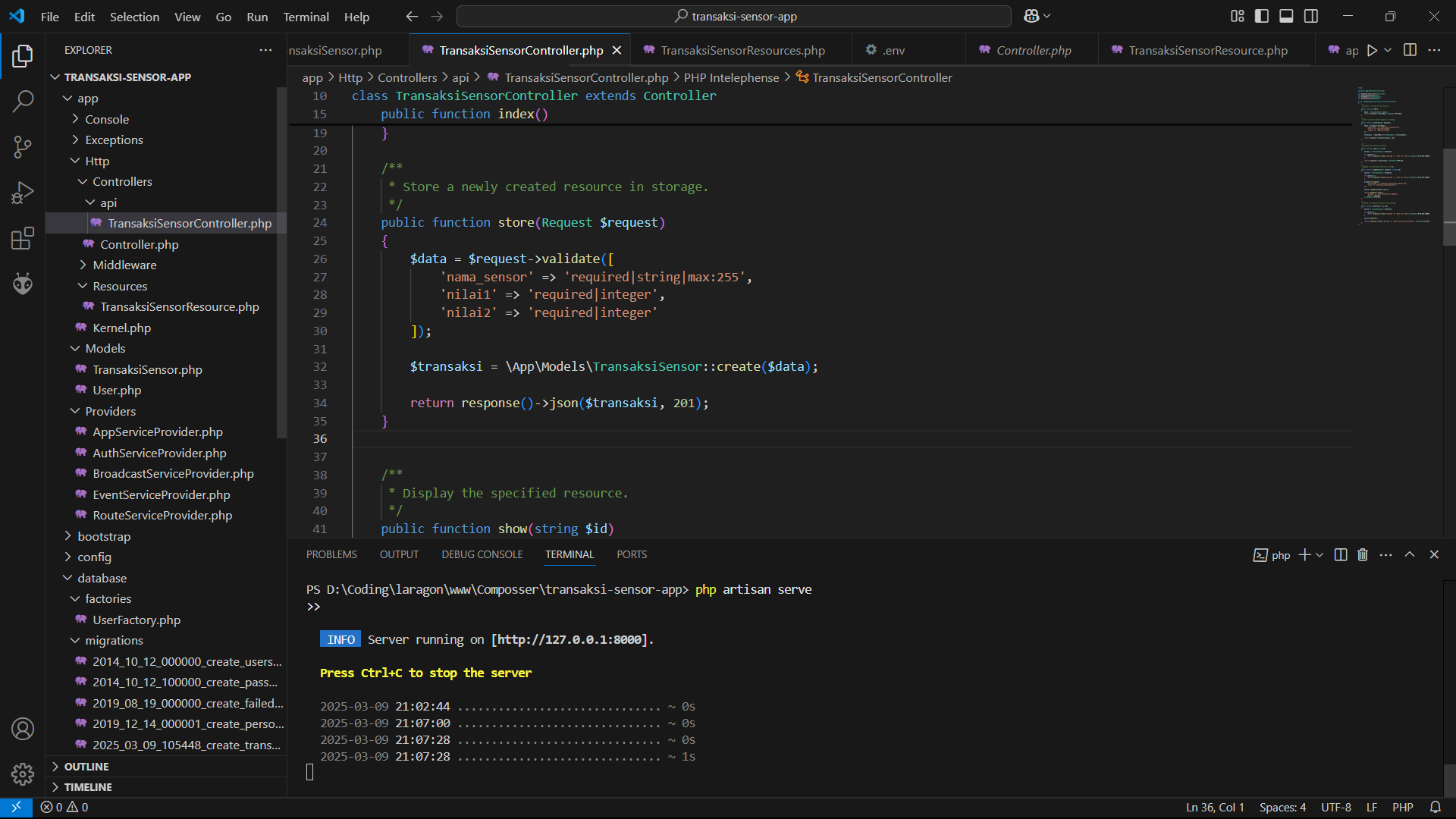
1. Connect ESP32 to Wi-Fi::

#include <WiFi.h>

const char\* ssid = "your-SSID";

const char\* password = "your-PASSWORD";

WiFi.begin(ssid, password);



1. Send HTTP GET request to an API:

use App\Http\Controllers\Api\TransaksiSensorController;

Route::apiResource('transaksi-sensor', TransaksiSensorController::class);

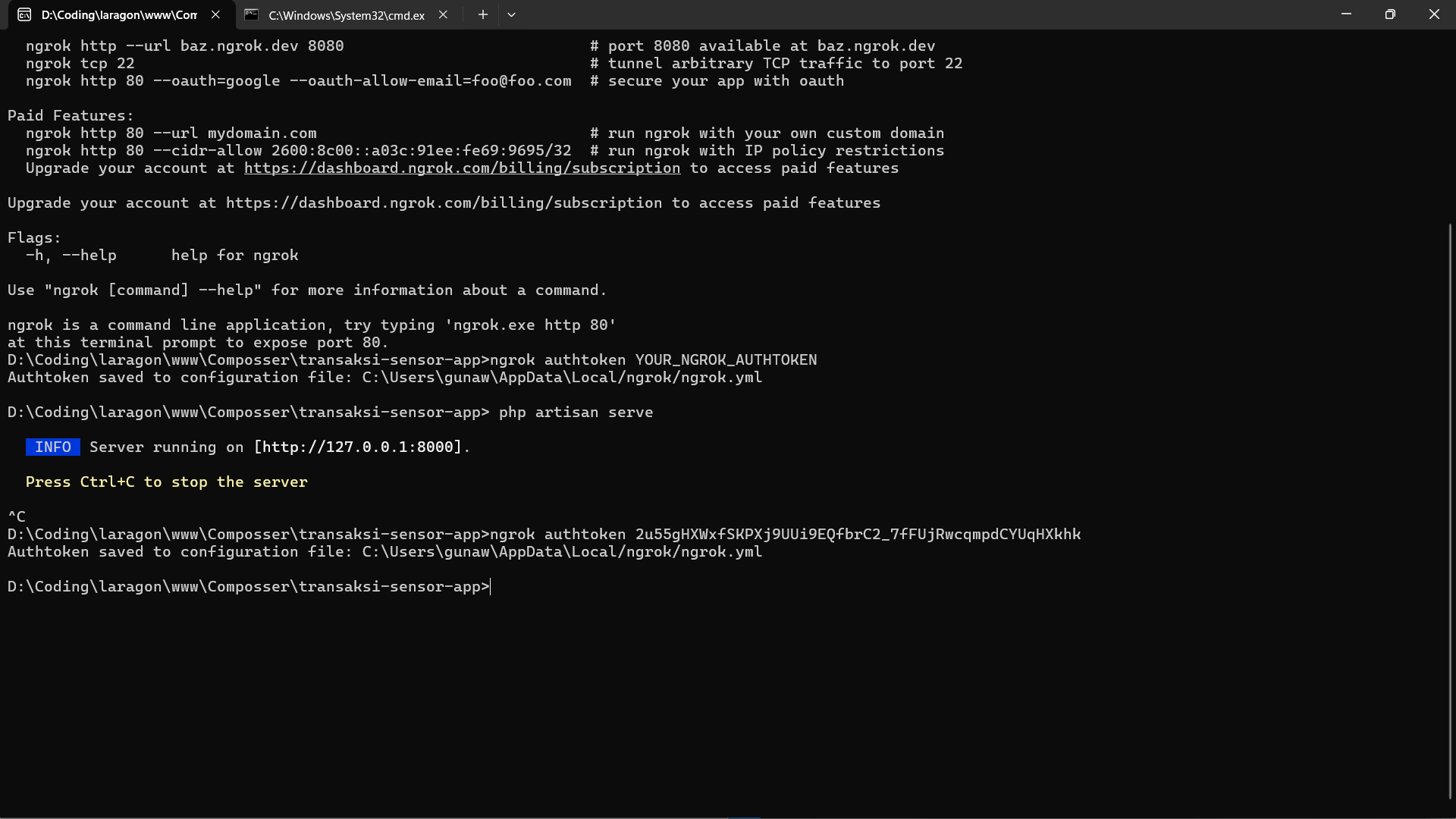
1. Implement CRUD methods in TransaksiSensorController.php.

#include <HTTPClient.h>

HTTPClient http;

http.begin("https://jsonplaceholder.typicode.com/posts/1");

int httpResponseCode = http.GET();

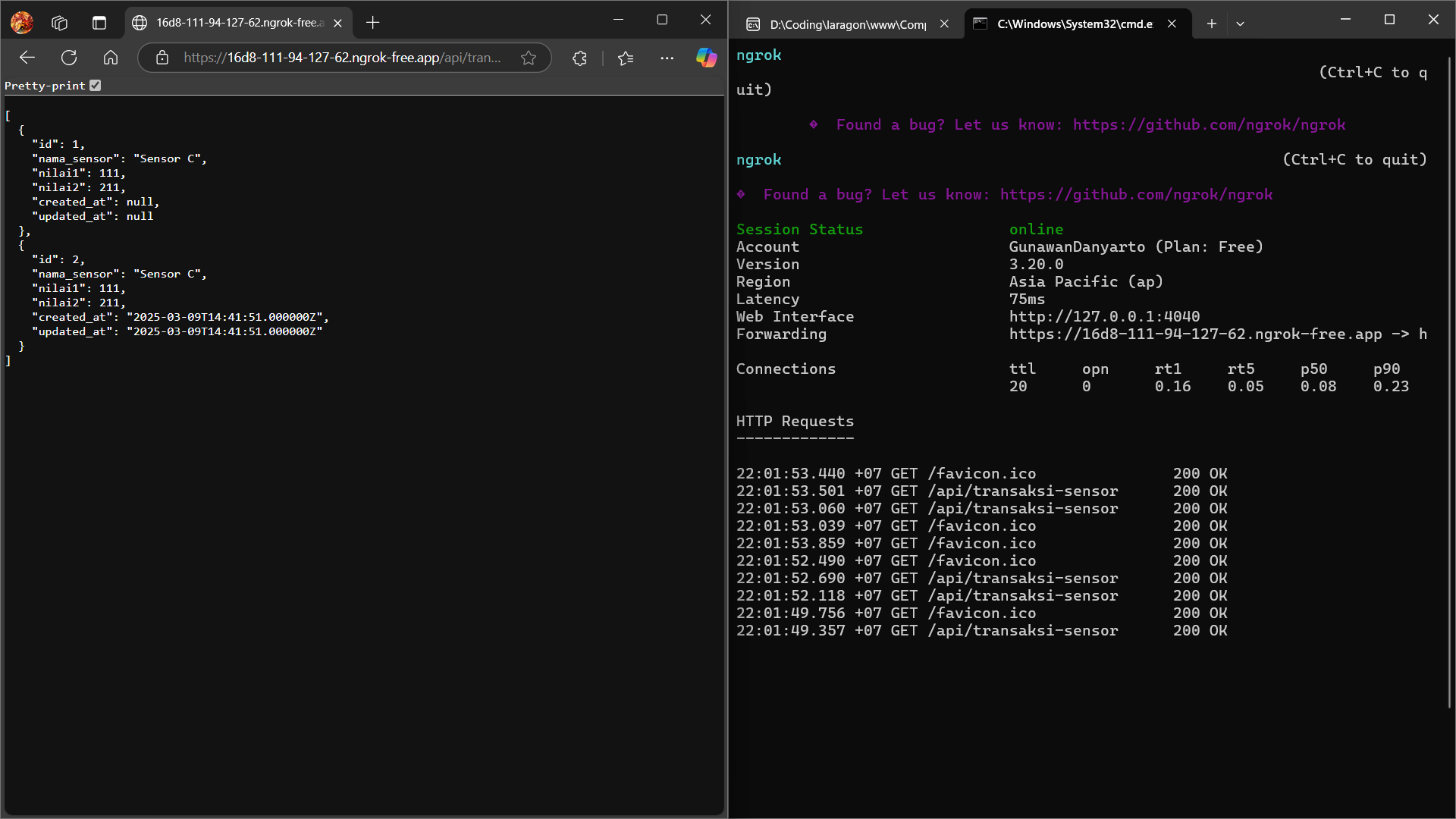


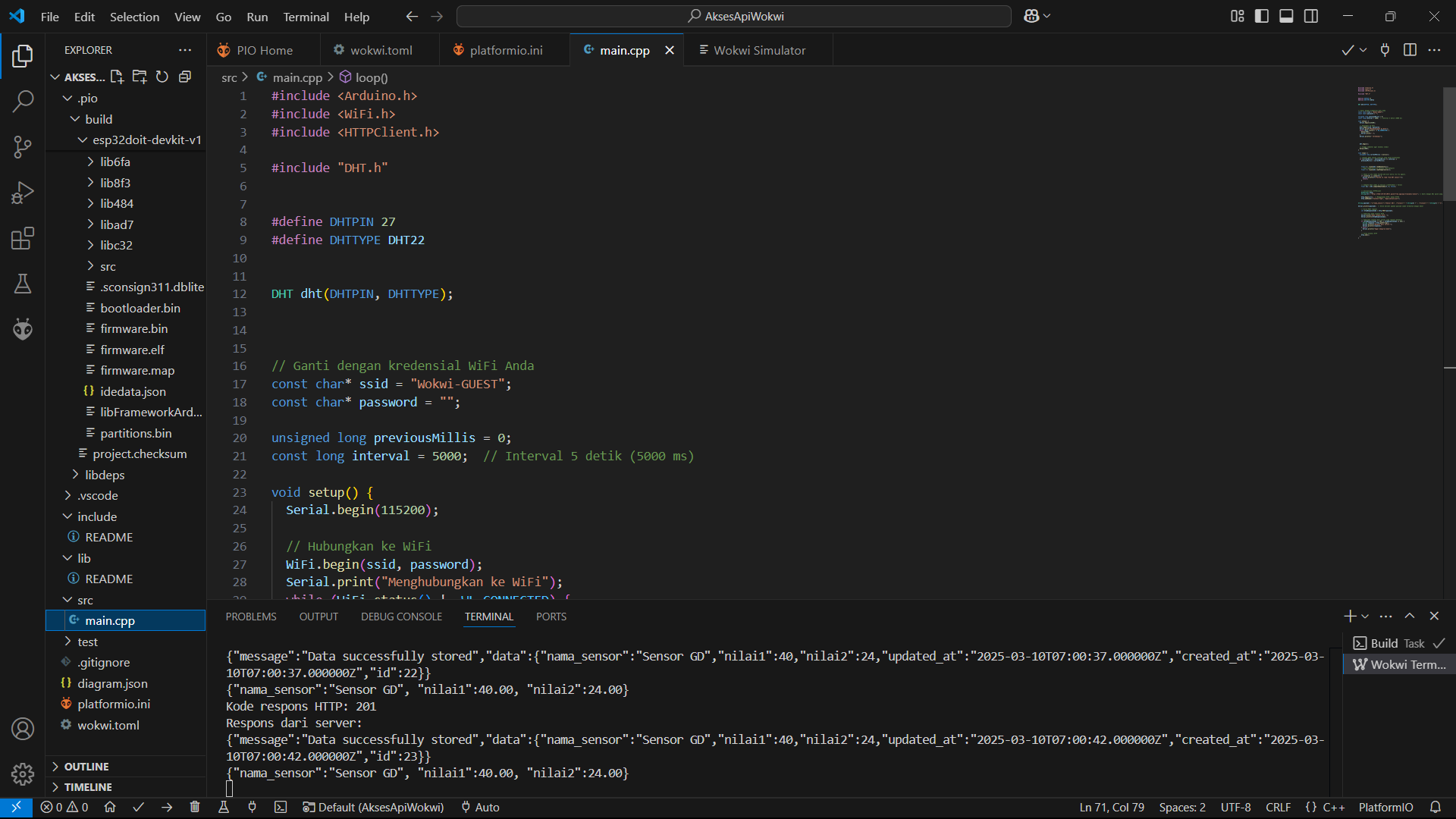
#### 2.2.3 A****nalyzing API Communication****

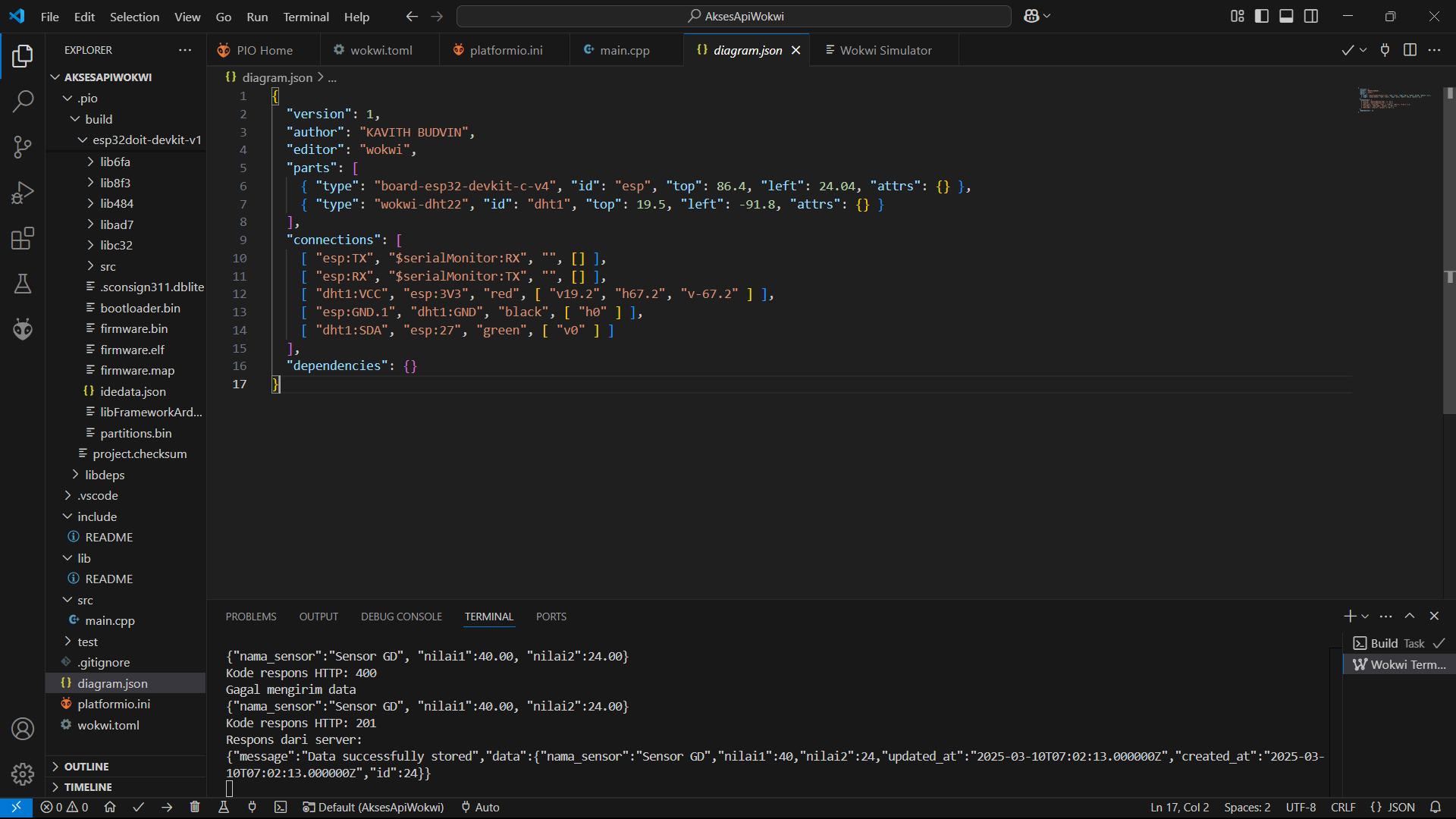
1. Capture response status codes (e.g., 200 OK, 404 Not Found).

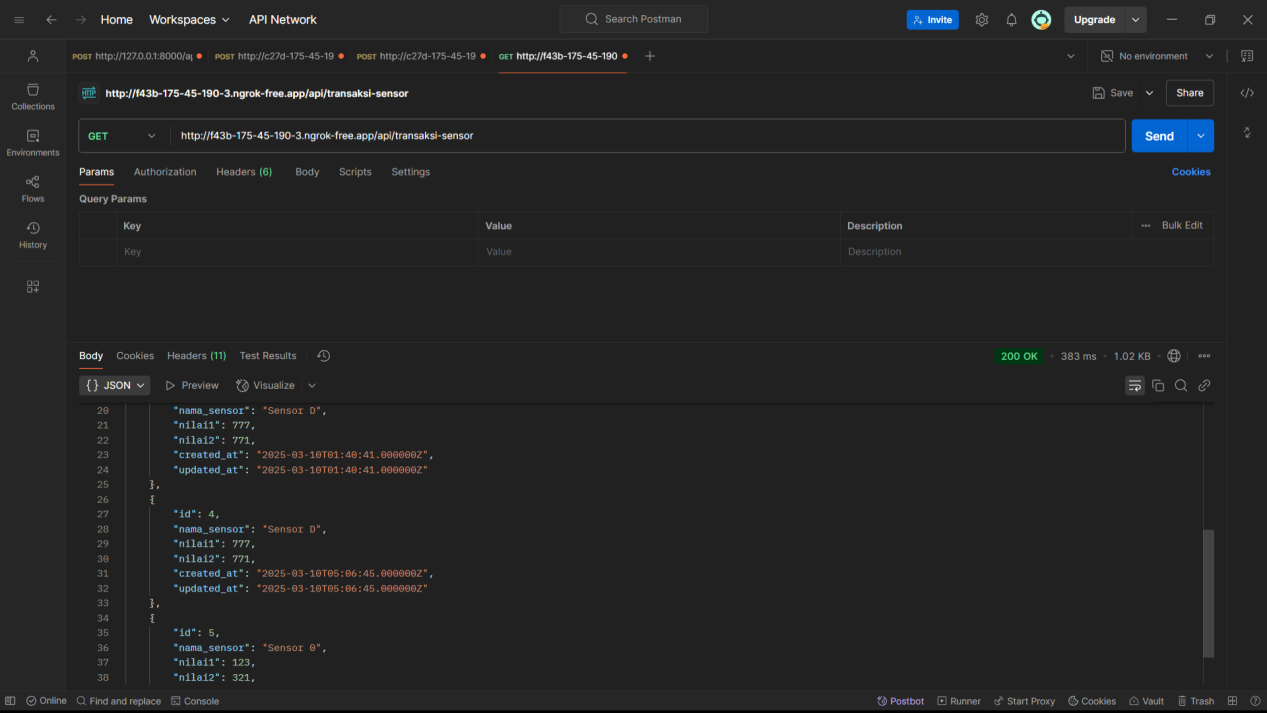
2. Measure response time using millis().

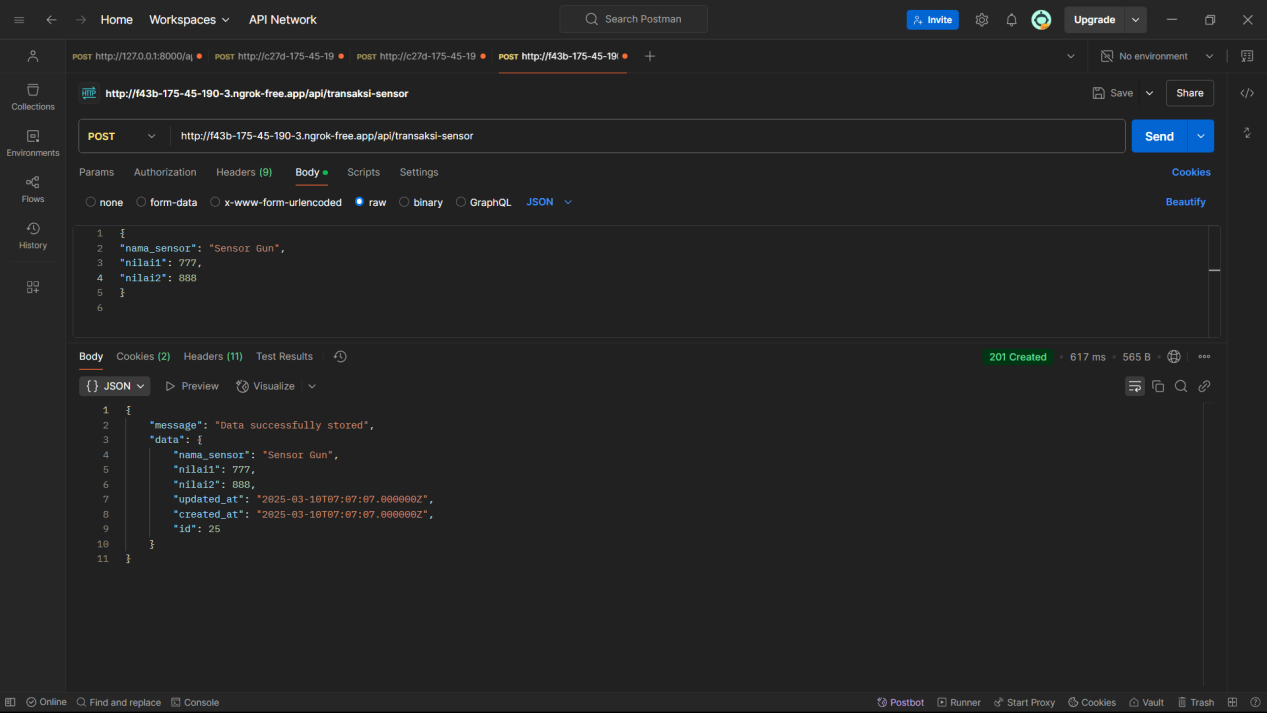
3. Validate received JSON data using ArduinoJson.h.

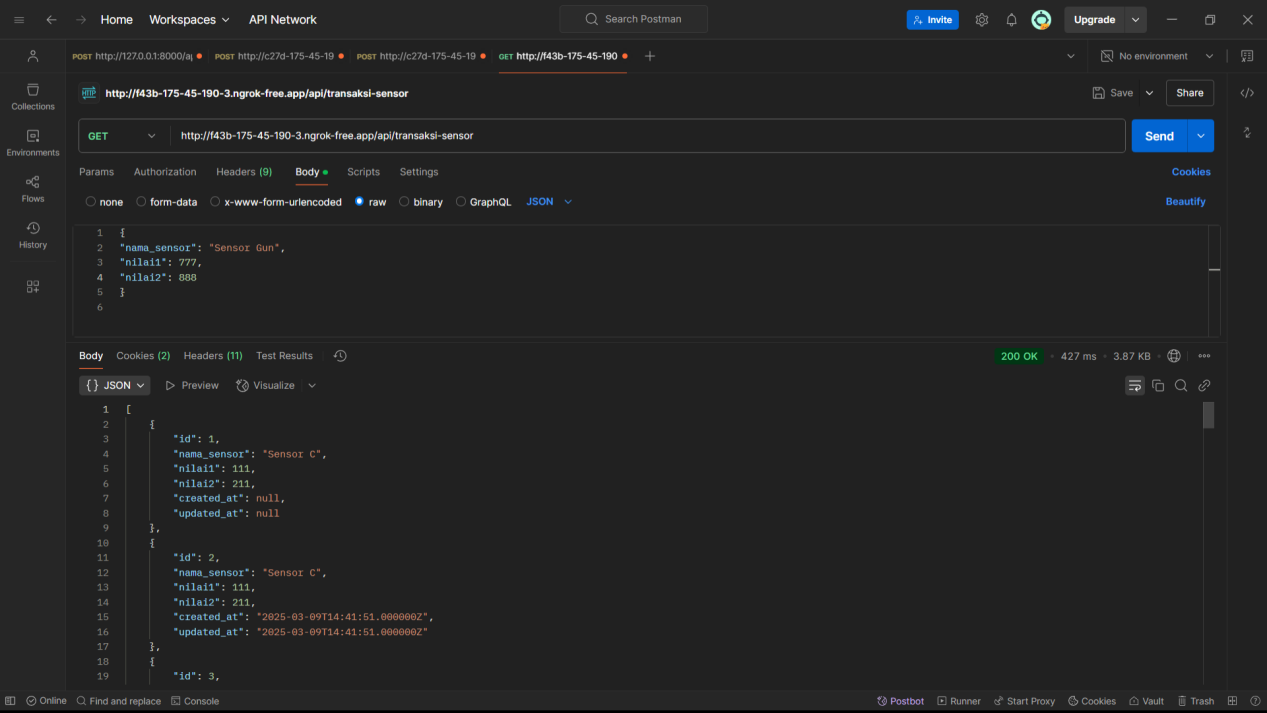


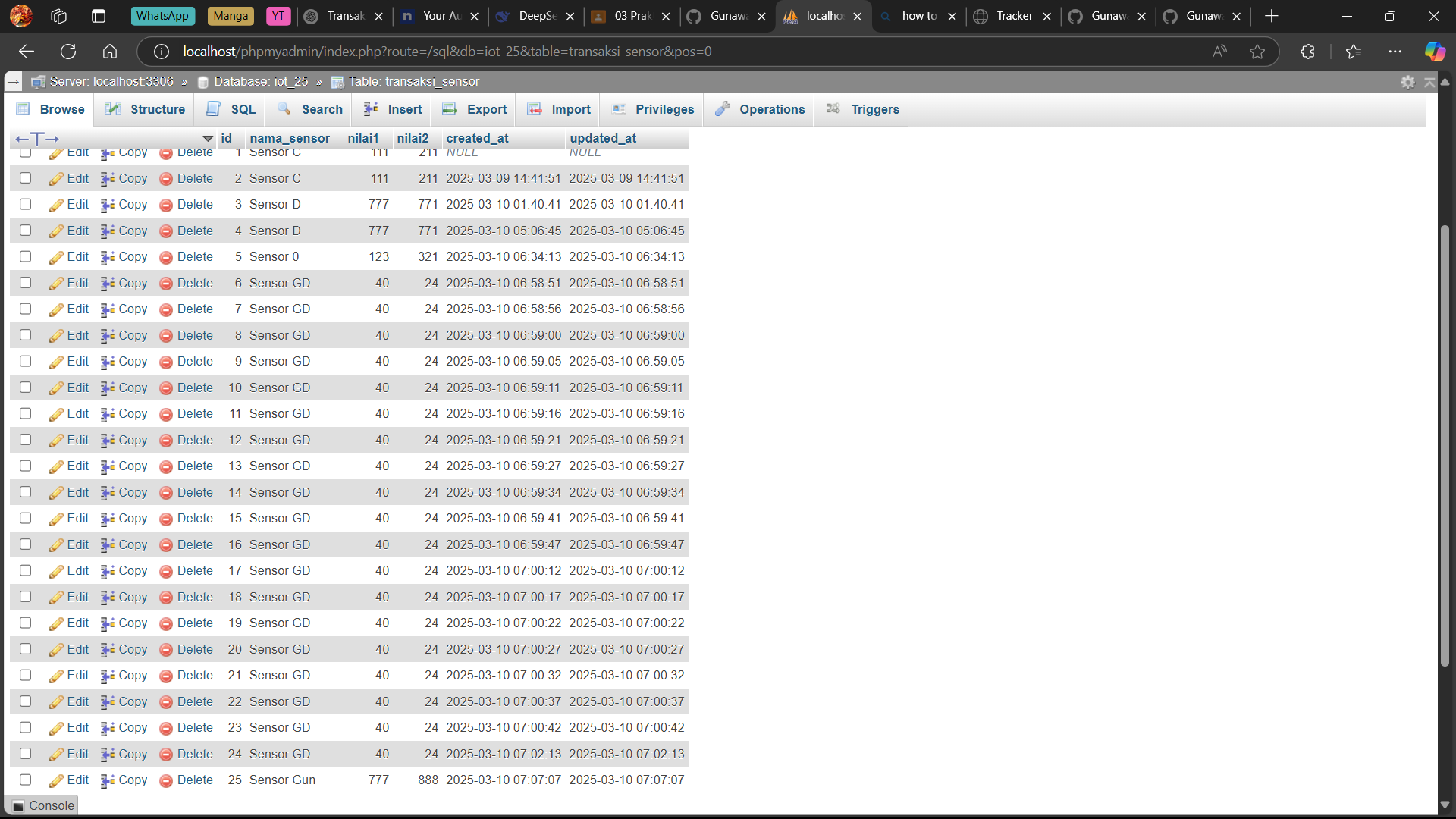












**3. Results and Discussion**

**3.1 Experimental Results**

1. The ESP32 successfully connected to the Wi-Fi network in WOKWI.

2. API responses were received in JSON format.

3. HTTP GET and POST requests functioned correctly.

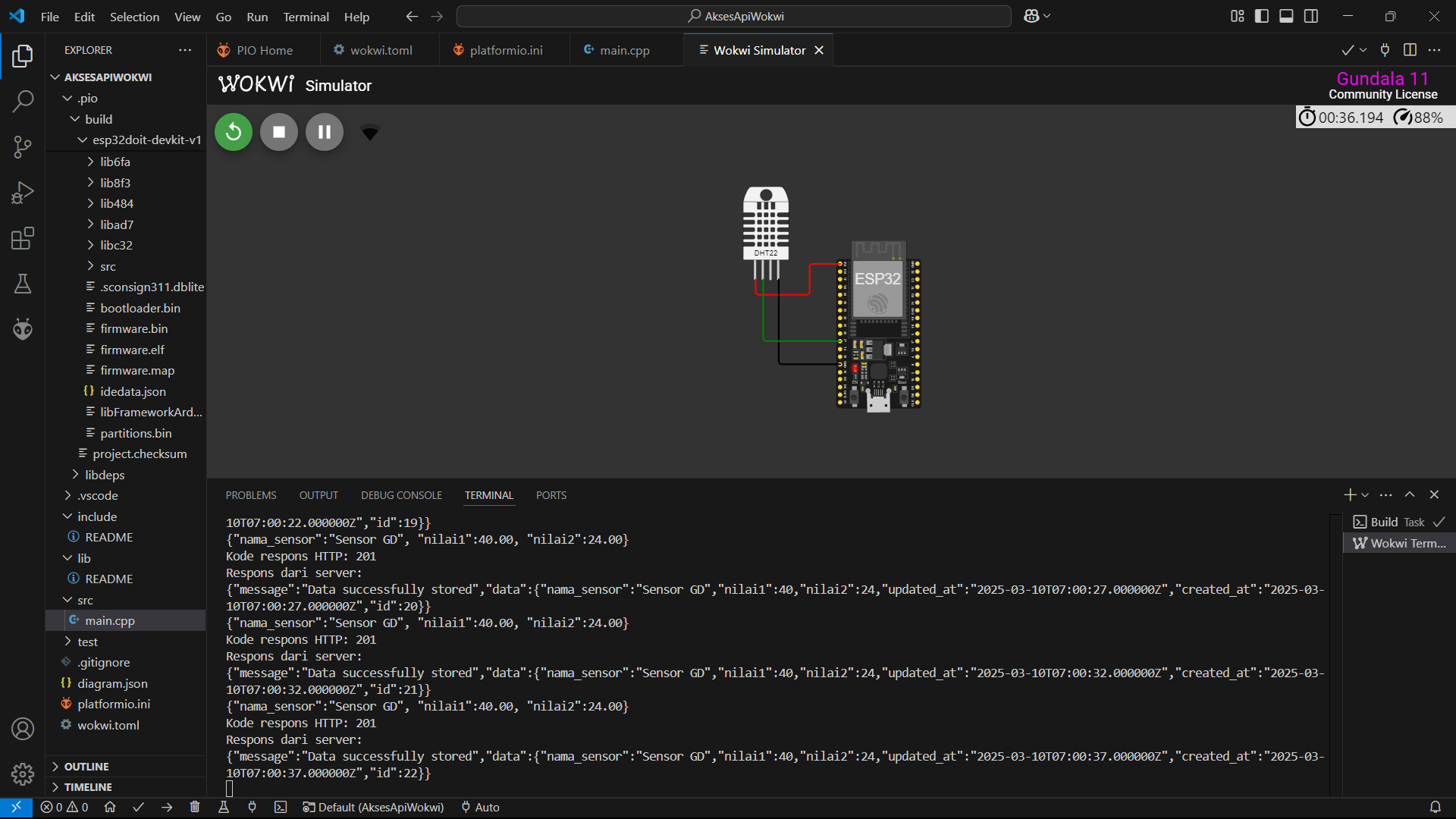
4. Response times varied depending on network latency (average ~300ms).

**3.2 Discussion**

**Benefits of WOKWI**: Enables rapid testing without physical components.

**Challenges**: Secure API requests (SSL/TLS) sometimes fail due to certificate validation errors.

**Future Improvements**: Implement authentication mechanisms for secured API communication.



**5. Conclusion**

This experiment successfully demonstrated API access using an ESP32 in WOKWI. The microcontroller effectively communicated with an external API, showcasing the potential of IoT devices in real-world scenarios. WOKWI proved to be a powerful tool for rapid prototyping and testing, making it an ideal environment for IoT development.