**API Access Practice Through WOKWI Simulation**

*Andreika Luna Alghivari*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: andreikaluna@student.ub.ac.id*

**Abstract**

This project focuses on the practice of accessing APIs through the Wokwi simulator using Laravel 11, executed in Visual Studio Code. Laravel facilitates the creation of RESTful APIs, while Wokwi provides a virtual IoT environment for testing and validating API functionality. The integration aims to simulate real-world scenarios where APIs interact with IoT devices, enabling efficient data exchange and control. This practice highlights the importance of API accessibility via public URLs, ensuring seamless communication between systems in IoT-based applications.

*Keywords: Laravel, DHT22 Sensor, MySQL, Wokwi, Visual Studio Code, PlatformIO.*

**1. Introduction**

* 1. **Background**

Laravel is a powerful PHP framework designed for web development, including RESTful API creation. RESTful APIs are essential for modern applications as they enable communication between distributed systems using standard HTTP methods like GET, POST, PUT, and DELETE. Wokwi is an IoT simulator that allows developers to design and test virtual circuits involving microcontrollers like ESP32 and sensors. By combining Laravel and Wokwi, developers can simulate API interactions with IoT devices without requiring physical hardware. Visual Studio Code provides an efficient development environment with debugging tools, making it ideal for managing both Laravel projects and Wokwi simulations.

* 1. **Objective**

1. To create and test RESTful APIs using Laravel 11 for IoT-based applications.
2. To integrate the Wokwi simulator with Laravel to simulate API interactions with virtual IoT devices.
3. To expose local APIs via public URLs using tools like Ngrok for remote access and testing.
4. To validate the functionality of APIs in controlling and retrieving data from simulated IoT device

**2. Methodology**

**2.1 Tools & Materials**

Laravel 11, Postman, Wokwi, DHT22 Sensor, XAMPP, Ngrok, Arduino IDE, and Visual Studio Code.

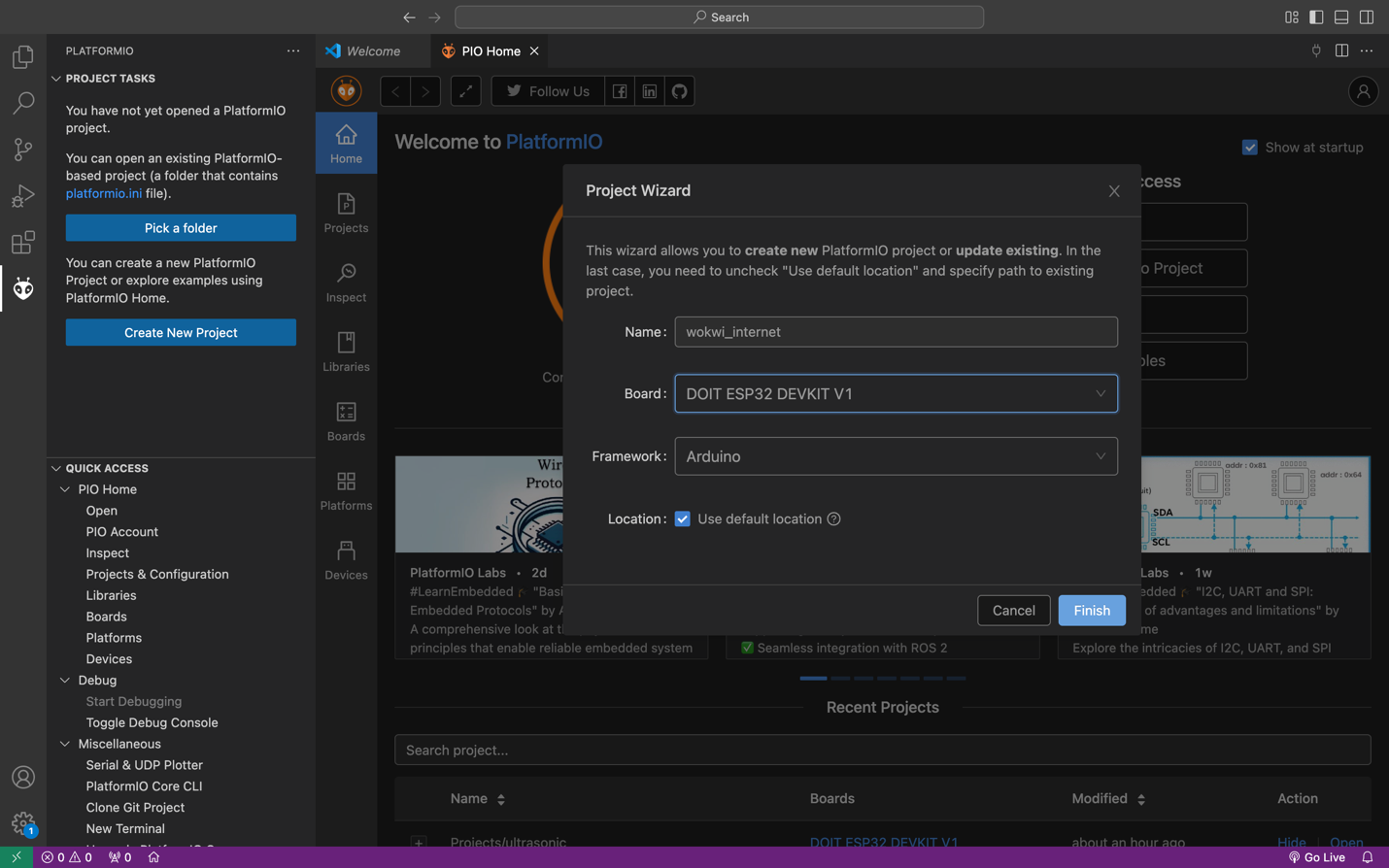
**2.2 Implementation Steps**

1. Running the Laravel PHP API artisan serve -host=0.0.0.0 -port=8080
2. Create a wokwi simulator file in platform.io
3. Modify the main.cpp file
4. Take the URL from Ngrok's generate command and add it to the wokwi.toml file and complete the main.cpp section (URL in http form)
5. Add the diagram.json file
6. Then build the main.cpp file
7. Assemble the DHT22 sensor with ESP32 and copy the diagram.json code to the diagram.json file in vscode
8. Change the platform.io file settings by adding 2 settings, namely monitor speed and lib\_deps
9. Run the simulator and make sure in the database that the data has been displayed and stored.

**3. Results and Discussion**

**3.1 Experimental Results**

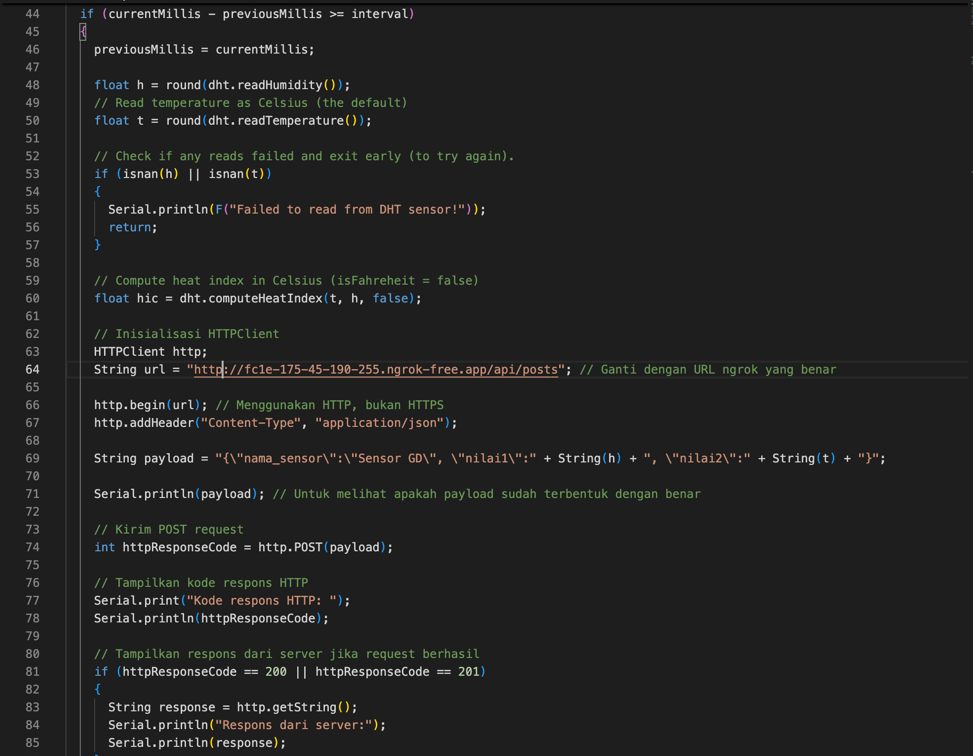
1. Create a wokwi simulator file in platform.io



1. Modify the main.cpp file

A screen shot of a computer screen

Description automatically generated



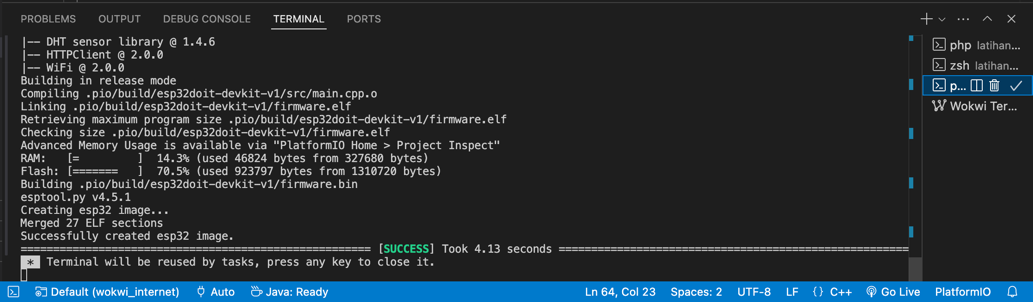
A black background with a white object

Description automatically generated with medium confidence

1. Take the URL from Ngrok's generate command and add it to the wokwi.toml file and complete the main.cpp section (URL in http form), then build

A computer screen shot of a program

Description automatically generated



1. Check ngrok url link on postman, the result is 201 created

A screenshot of a computer

Description automatically generated

1. Assemble the DHT22 sensor with ESP32 and copy the diagram.json code to the diagram.json file in vscode

A computer screen shot of a computer program

Description automatically generated

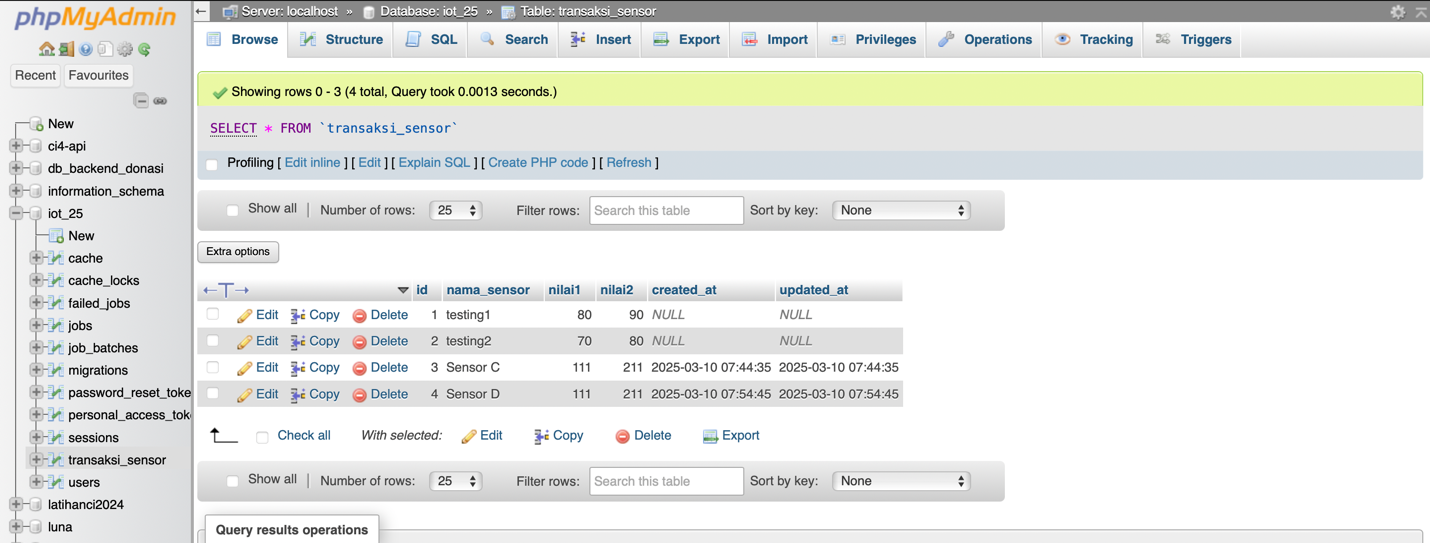
1. Run the simulator and make sure in the database that the data has been displayed and stored.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated



The practice of accessing the API encountered a significant issue where data could not be sent to the database, resulting in an error code of -1. Although Postman indicated a successful response with a status of 201 Created, the underlying problem persisted. Repeated attempts to generate a new Ngrok URL were unsuccessful, likely due to server congestion on the free Ngrok platform, which can affect performance and reliability. This situation highlights the challenges of using free services for API testing and development, emphasizing the need for stable connections when working with remote databases and APIs. Further investigation into server load and potential alternatives may be necessary to resolve these connectivity issues effectively.