**Pratice Making A Framework Scanning Wifi and Traffic Light**

*Damar Lintang Priyatama*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*damarpriya138@gmail.com*](mailto:damarpriya138@gmail.com)

**Abstract**

This project demonstrates utilizing an ESP32 microcontroller to perform Wi-Fi network scanning and control LED states based on the scanning results. The code is divided into two sections: the first part scans for available Wi-Fi networks, displaying the SSID and signal strength (RSSI) of each network detected. The second part controls two LEDs, turning them on and off with a 1-second delay, providing visual feedback. The integration of Wi-Fi scanning and LED control in this project showcases the versatility of the ESP32 in real-time wireless communication and hardware control, making it ideal for IoT applications where network monitoring and device interaction are required.

*Keywords—* *ESP32, Wi-Fi scanning, LED control, IoT, Arduino*

**1. Introduction**

**1.1 Background**

Wi-Fi scanning is a fundamental capability in many wireless IoT applications, as it allows devices to discover available networks in their vicinity, assess signal strength, and make intelligent decisions based on the network conditions. The ability to measure signal strength (RSSI – Received Signal Strength Indicator) is particularly important for tasks like optimizing device placement, ensuring stable communication, and managing power consumption in wireless communication systems.

LEDs, on the other hand, are commonly used for providing visual feedback in embedded systems. In many IoT applications, LEDs are used to indicate the status of a device, such as whether it is successfully connected to a network, or whether a certain action has been triggered. Combining Wi-Fi scanning and LED control on the ESP32 provides a simple yet effective demonstration of real-time wireless network monitoring and device interaction.

This project aims to leverage the Wi-Fi scanning functionality of the ESP32 to detect nearby wireless networks and display their details in the serial monitor, while simultaneously using LEDs to provide visual feedback based on the scanning process. The use of LED indicators enhances the interactivity and user experience, making the system more intuitive for monitoring and debugging purposes. By integrating these functionalities, the project illustrates how the ESP32 can be used in network-based IoT applications while also controlling hardware elements like LEDs for real-time feedback.

**1.2 Purpose Experience**  
By running this project, users will gain a hands-on experience of interacting with both the software and hardware aspects of the ESP32. The visual feedback through the LED indicators serves as an intuitive way for users to understand the process flow of the network scan, with the LEDs turning on and off in a predictable, time-based pattern. In addition, the serial monitor provides detailed insights into the networks detected by the ESP32, which helps users understand the context of the scan, such as network availability, signal strength, and security encryption.

**2. Methodology**

**2.1 Tools & Materials**

-Laptop Asus

-Vscode

-Arduino IDE

-ESP32 Board Support

**2.2 Implementation Steps**

 **Steps from Scanning Wifi**

-first in wokwi choose ESP32

-copy the code that has been prepared by the module

-paste the code into main.c

-then run and wait for the scan results to appear netwroks found

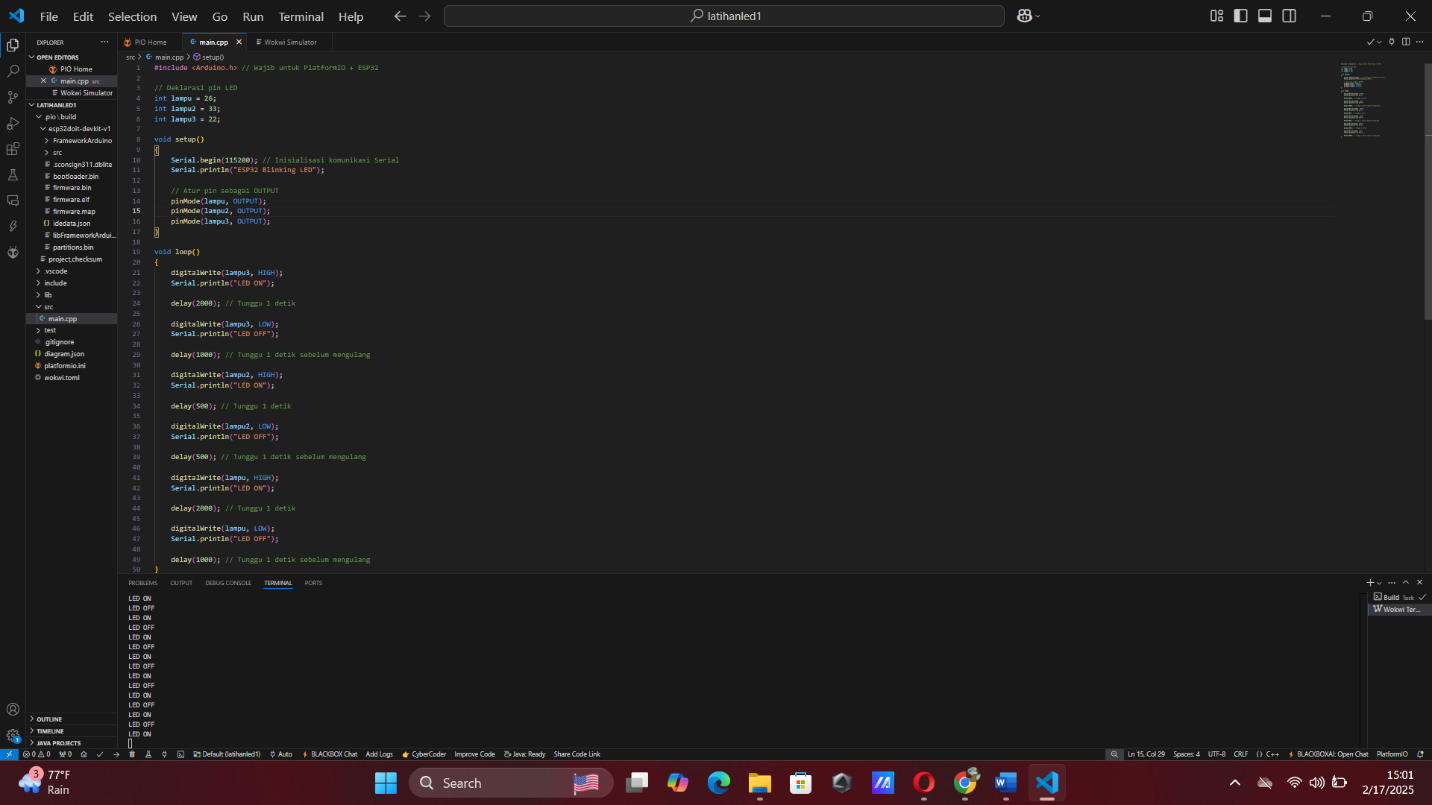
-If it appears then the WiFi scan is declared successful

 **Steps from LED control**

**-**Open Arduino IDE and choose ESP32 for traffic

-Arrange the ESP32 framework by adding red, yellow, green lights

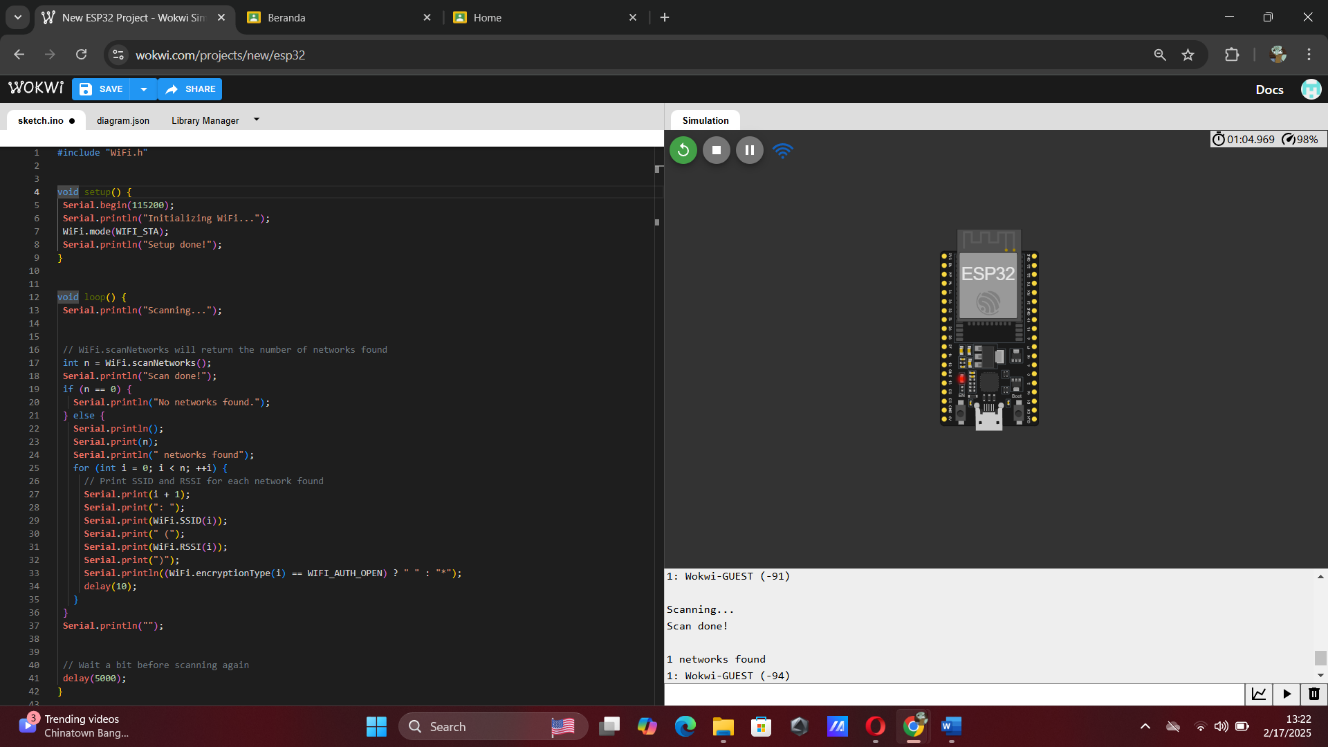
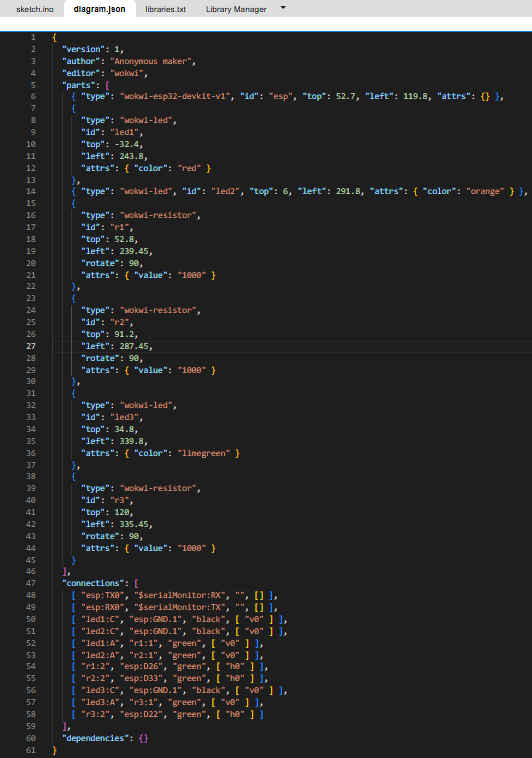
-Then adjust the light to the LED pin declaration according to its color

-Type the code like this in main.c

-If the results of the lights are flashing in a loop according to the task, then you have finished completing the traffic light

**3. Results and Discussion**

**3.1 Experimental Results**

* Code and result from Scanning Wifi
* Diagram Json from traffic light
* Result from Json from traffic light