# **ESP32 Traffic Light Simulation Using 3 LEDs**

*Candra Wahyu Perdana*

Vocational Faculty, Brawijaya University

Email: candraega28@gmail.com

**Abstract**

This practical work aims to simulate a simple traffic light system using an ESP32 microcontroller. The system utilizes three LEDs (red, yellow, green) to represent standard traffic light behavior. The code was developed using Arduino framework in PlatformIO, and the hardware consists of ESP32, jumper wires, and breadboard. The simulation follows a timed loop of green (go), yellow (slow), and red (stop) signals. The setup serves as a foundational exercise for learning digital output control in embedded systems.

*Keywords—ESP32, Traffic Light, Arduino, PlatformIO, IoT, LEDs, Simulation*

**1. Introduction**

**1.1 Background**

Internet of Things (IoT) has become a core part of modern embedded systems and smart city solutions. One basic use case is traffic light simulation which helps beginners understand control flow, timing, and GPIO usage in microcontrollers. ESP32, with its built-in WiFi and powerful features, is widely used for both simple and complex IoT projects. This simulation uses digital output pins to control LEDs in a pattern resembling a real-world traffic light.

**1.2 Objectives**

* To simulate a traffic light using an ESP32 board.
* To understand GPIO pin control and delay functions in Arduino.
* To observe sequential logic and timing through LEDs.

**2. Methodology**

**2.1 Tools & Materials**

* **Software:** PlatformIO (VSCode), Arduino Framework
* **Hardware (simulated):**
  + ESP32 board
  + Breadboard
  + 3 LEDs (Red, Yellow, Green)
  + Jumper Wires
  + Micro USB Cable

#### **2.2 Circuit Implementation**

#### **1. Database Setup**

### **LED merah → GPIO 26**

### **LED kuning → GPIO 25**

### **LED hijau → GPIO 27**

### **Setiap LED terhubung ke resistor dan ground melalui breadboard.**

### **ESP32 terhubung ke laptop melalui kabel micro USB.**

**2.3 Source Code**

**Modify the migration file:**

| #include <Arduino.h>  // Deklarasi pin LED  const int lampuMerah = 26; // LED Merah  const int lampuKuning = 25; // LED Kuning  const int lampuHijau = 27; // LED Hijau  void setup() {  Serial.begin(115200);  Serial.println("ESP32 Traffic Light Simulation");  // Atur pin sebagai output  pinMode(lampuMerah, OUTPUT);  pinMode(lampuKuning, OUTPUT);  pinMode(lampuHijau, OUTPUT);  }  void loop() {  // Pastikan semua LED mati  digitalWrite(lampuMerah, LOW);  digitalWrite(lampuKuning, LOW);  digitalWrite(lampuHijau, LOW);  delay(10);  // Hijau menyala  digitalWrite(lampuHijau, HIGH);  Serial.println("Lampu Hijau - GO!");  delay(3000);  digitalWrite(lampuHijau, LOW);  delay(10);  // Kuning menyala  digitalWrite(lampuKuning, HIGH);  Serial.println("Lampu Kuning - SLOW!");  delay(1000);  digitalWrite(lampuKuning, LOW);  delay(10);  // Merah menyala  digitalWrite(lampuMerah, HIGH);  Serial.println("Lampu Merah - STOP!");  delay(3000);  digitalWrite(lampuMerah, LOW);  delay(10);  } |
| --- |

## 

## **3. Results & Discussion**

**3.1 Observations**

After uploading the code to the ESP32 board using PlatformIO, the system worked as expected. The green LED turns on for three seconds, followed by the yellow LED for one second, and finally the red LED for three seconds. This cycle repeats continuously, mimicking the behavior of a real-world traffic light.

Each time an LED turns on, a corresponding message is printed on the serial monitor, which helps in debugging and understanding the system's real-time behavior. The timing intervals can be adjusted by changing the delay() values in the code.

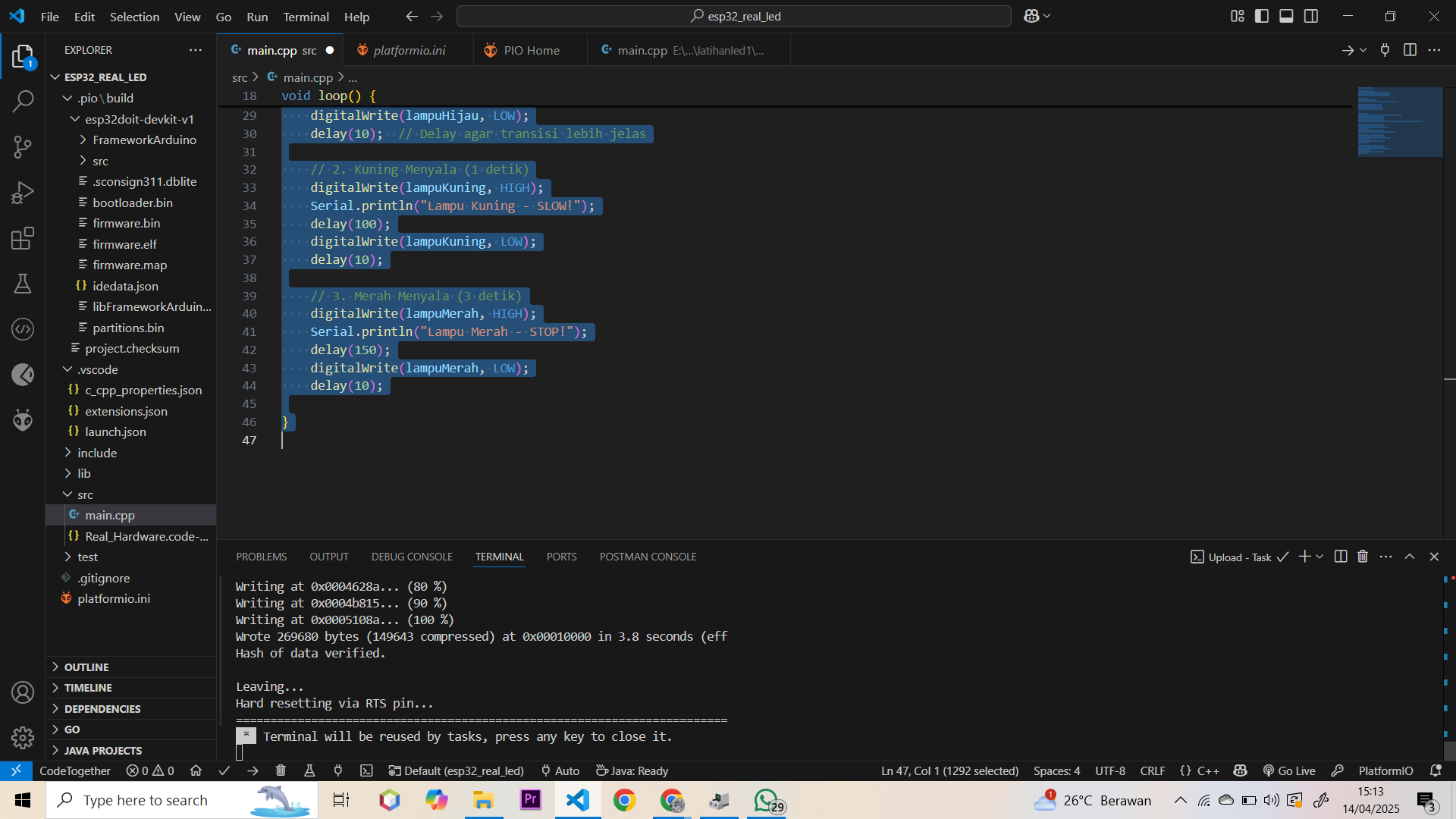
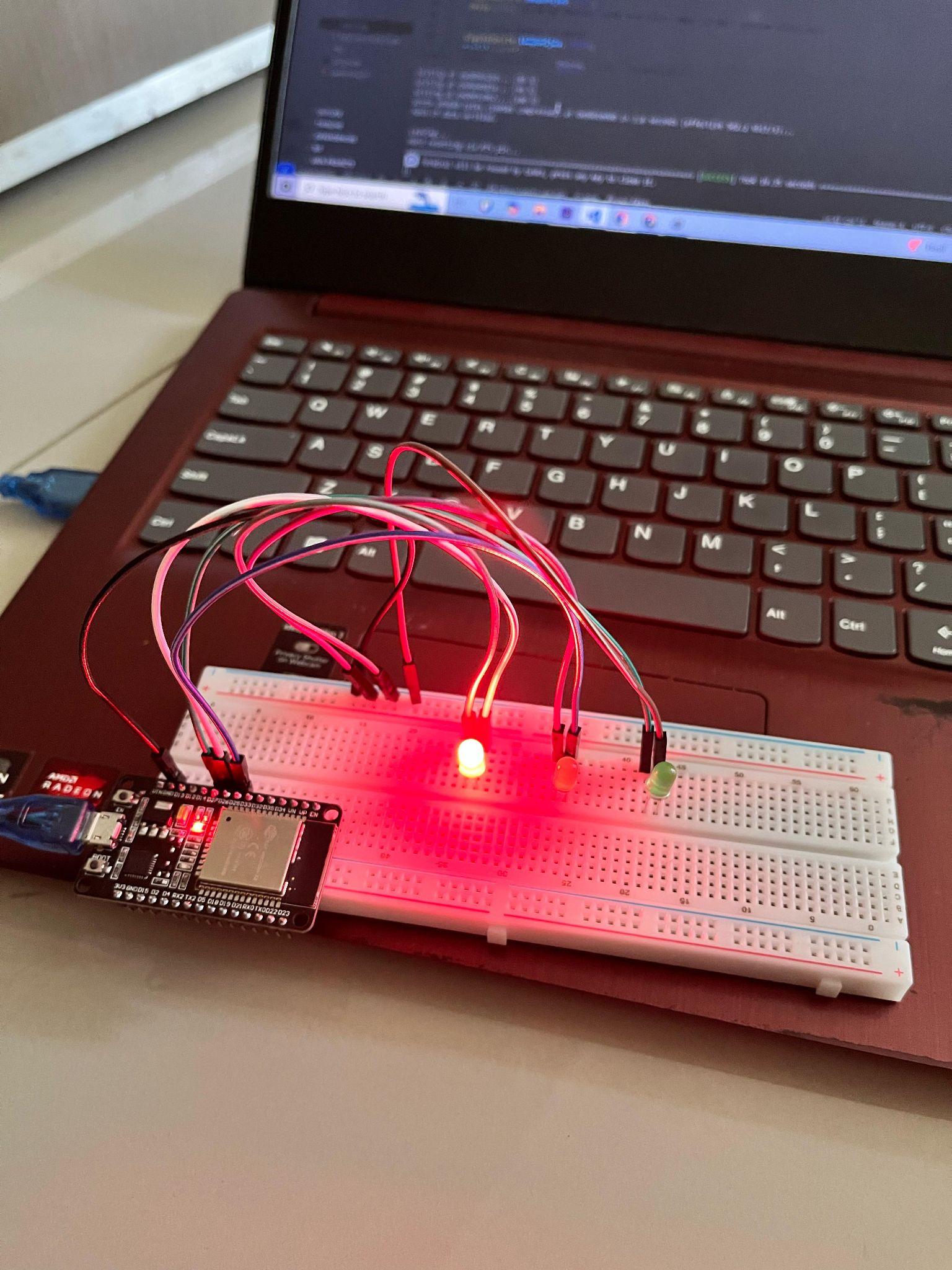
**3.2 Analysis**

This project demonstrates the effective use of GPIO pins for output, which is a core concept in embedded systems. Controlling LEDs is the foundation for controlling other actuators in IoT applications, such as buzzers, motors, or relays. The project also highlights how simple logical sequences can be implemented in real-world applications using only basic components.

Moreover, the experiment reinforces the importance of proper circuit design, including the use of resistors to protect the LEDs and understanding the physical layout using a breadboard.

### **3.3 Photo Evidence**

**Figure 1.** ESP32 Traffic Light Simulation with Red LED Active



## **4. Conclusion**

The ESP32-based traffic light simulation was successfully implemented using three LEDs. The project demonstrated how microcontrollers can control output devices using digital signals and timing functions. It served as a practical introduction to the ESP32 platform and embedded programming. Future improvements may include adding a push-button to simulate pedestrian crossing or using sensors to implement adaptive traffic light systems.

This experiment is valuable for students who are new to IoT and embedded systems, as it builds a strong foundation for more complex projects involving inputs, network communication, and real-time processing.