**ESP32 Real Hardware Practice**

*Agatha Herma Putra*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: agathaputra24@student.ub.ac.id*

**Abstract**

This practical project aims to implement the ESP32 microcontroller in a simple traffic light simulation system. Three LEDs—red, yellow, and green—are used to represent conventional traffic lights. The ESP32 acts as the central controller, managing the sequential switching of the lights based on a predefined timing logic. Programming was carried out using the Arduino IDE and the C++ programming language. The results show that the ESP32 can control the light sequence automatically and consistently. This project provides a fundamental understanding of ESP32 applications in Internet of Things (IoT) projects and demonstrates its use in basic real-world control systems.

*Keywords : ESP32, traffic light simulation, microcontroller, IoT, LED, Arduino IDE*

**1 Introduction**

**1.1 Background**

The rapid development of microcontroller technology has enabled the creation of smart systems that can be implemented in various real-life applications. One of the widely used microcontrollers is the ESP32, known for its built-in Wi-Fi and Bluetooth capabilities, making it ideal for Internet of Things (IoT) projects. Traffic light systems are essential components in transportation infrastructure to regulate vehicle flow and ensure road safety. Simulating a traffic light system using ESP32 and basic electronic components can serve as an effective introduction to microcontroller-based automation.

**1.2 Purpose of the experiment**

The purpose of this project is to design and implement a simple traffic light control system using the ESP32 microcontroller. Through this project, students can gain practical experience in programming microcontrollers, wiring electronic components, and understanding the basic logic of automation in traffic management systems.

**2.1 Methodology**

This project uses an ESP32 microcontroller as the main controller, along with three LEDs (red, yellow, and green) to simulate a traffic light system. The circuit is assembled on a breadboard with appropriate resistors connected to each LED. The ESP32 is programmed using the Arduino IDE with C++ code to control the timing and sequence of the LED lights. Each light turns on and off in a loop based on typical traffic light intervals. The setup is powered through a USB connection, and the code is uploaded via the Arduino IDE. Testing is done to ensure that the sequence operates as intended and mimics a real traffic light system.

**2.2 Tools & Materials**

To successfully complete the ESP32 Real Hardware Practice. These include:

* **Hardware Requirements**

1. An ESP32 Development Board
2. Three LEDs (Red, Yellow, Green)
3. Jumper Wires (Male to Male)
4. Breadboard (optional, for easier wiring)
5. A USB Cable (to connect ESP32 to the computer)
6. A computer or laptop with USB port and internet connection

* **Software Rewuirements**

1. Arduino IDE (for coding and uploading to ESP32)
2. Wokwi (optional, for online simulation of ESP32 projects)
3. CP210x USB to UART Bridge Driver (if needed for ESP32 board connection)
4. PlatformIO (alternative development environment integrated with Visual Studio Code)

**2.3 Implemention Steps**

Implementation for experiment making real ESP32 with LED:

**1. Install Driver for ESP 32**

To enable communication between the ESP32 board and the computer, it is necessary to install the **Silicon Labs CP210x USB to UART Bridge Driver**. This driver allows the computer to recognize the ESP32 when connected via USB.

* Visit the official Silicon Labs website: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
* Download the driver compatible with your operating system (Windows, macOS, or Linux).
* Run the installer and follow the on-screen instructions to complete the installation.
* After installation, reconnect the ESP32 board to ensure it is properly detected by the system.
* Open the Arduino IDE and select the correct **COM port** to begin uploading programs.

**2. Create Project**

* Click the **PlatformIO Home** icon on the bottom-left sidebar.
* Select **"New Project"**.
* Fill in the project details:
  + **Project Name**: esp32\_real\_led
  + **Board**: Select *ESP32 Dev Module*
  + **Framework**: Choose *Arduino*
  + Click **Finish**

**4. Add Your Code**

* Navigate to the src folder and open main.cpp.
* Replace the content with the following code:

cpp

Copy code

#include <Arduino.h> // Required for PlatformIO + ESP32

// LED pin declarations

int lampu = 33;

int lampu2 = 25;

void setup() {

Serial.begin(115200); // Initialize Serial communication

Serial.println("ESP32 Blinking LED");

// Set pins as OUTPUT

pinMode(lampu, OUTPUT);

pinMode(lampu2, OUTPUT);

}

void loop() {

// Turn on both LEDs

digitalWrite(lampu, HIGH);

digitalWrite(lampu2, HIGH);

Serial.println("LED ON");

delay(1000); // Wait 1 second

// Turn off both LEDs

digitalWrite(lampu, LOW);

digitalWrite(lampu2, LOW);

Serial.println("LED OFF");

delay(1000); // Wait 1 second before repeating

}

**5. Connect the ESP32 to Your Computer**

* Use a USB cable to connect the ESP32 board.
* Ensure the CP210x USB to UART driver is installed.

**6. Select the Correct Port**

* Click the plug icon at the bottom bar of PlatformIO or manually choose the correct COM port from the Devices list.

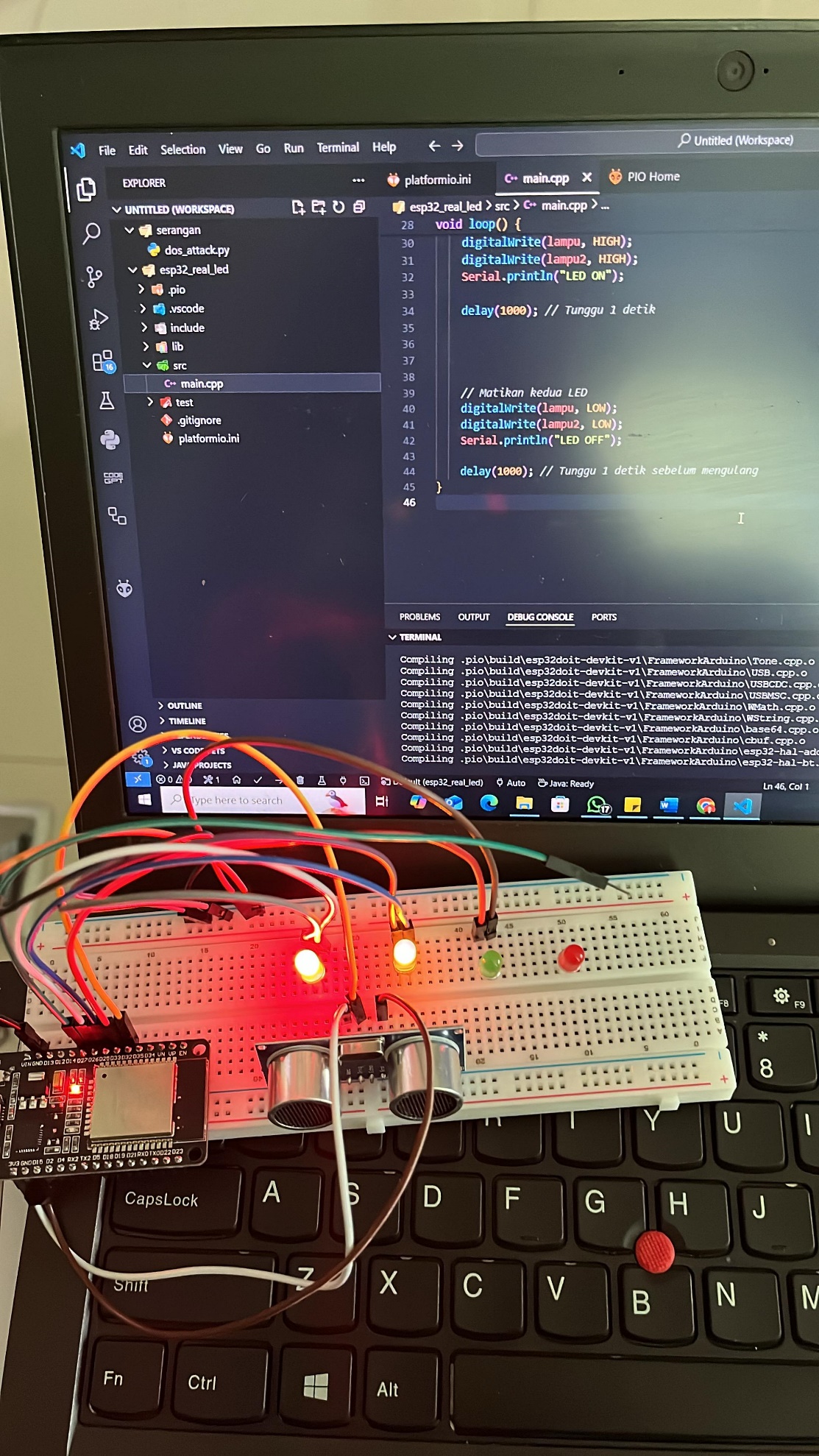
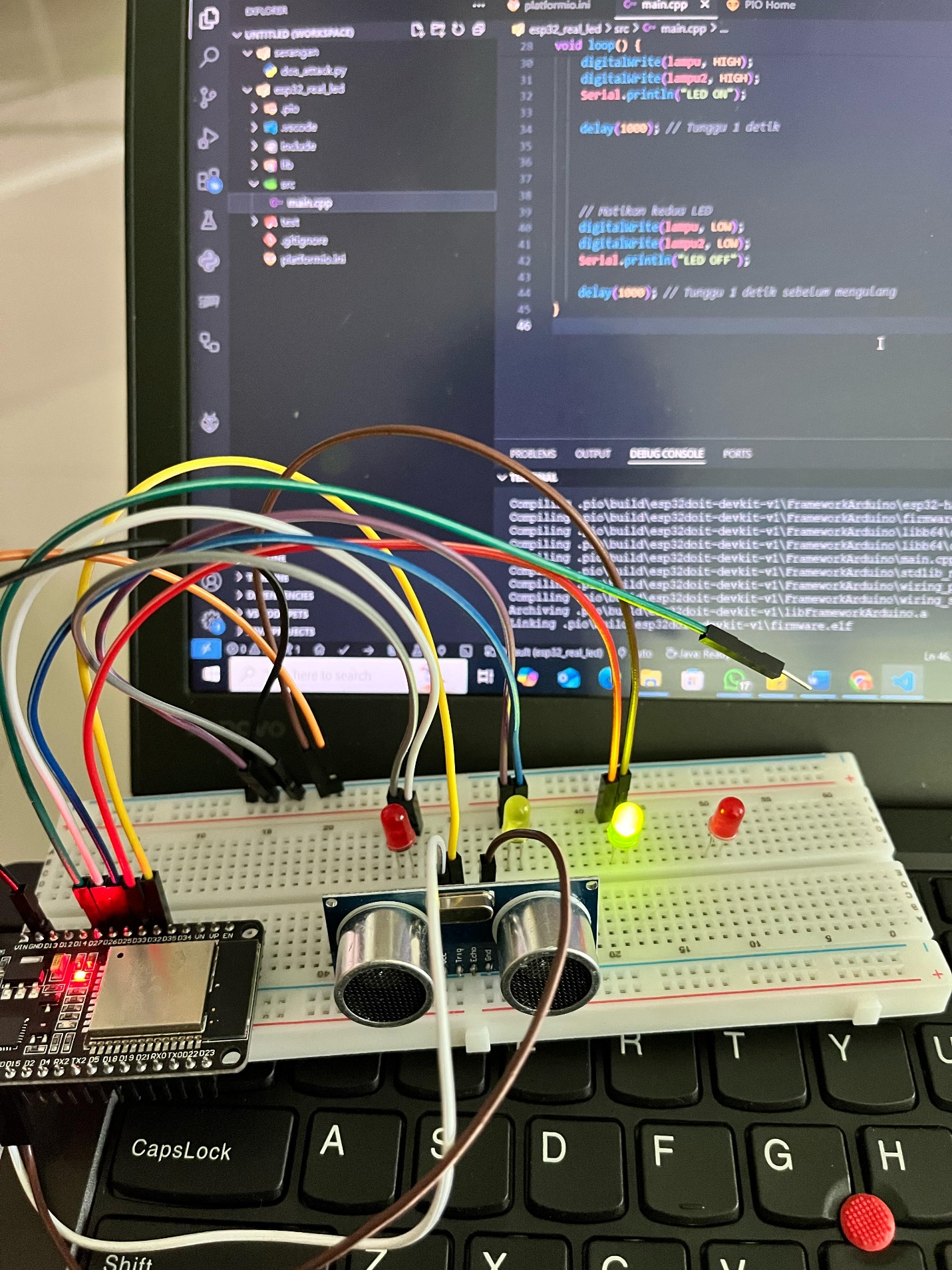
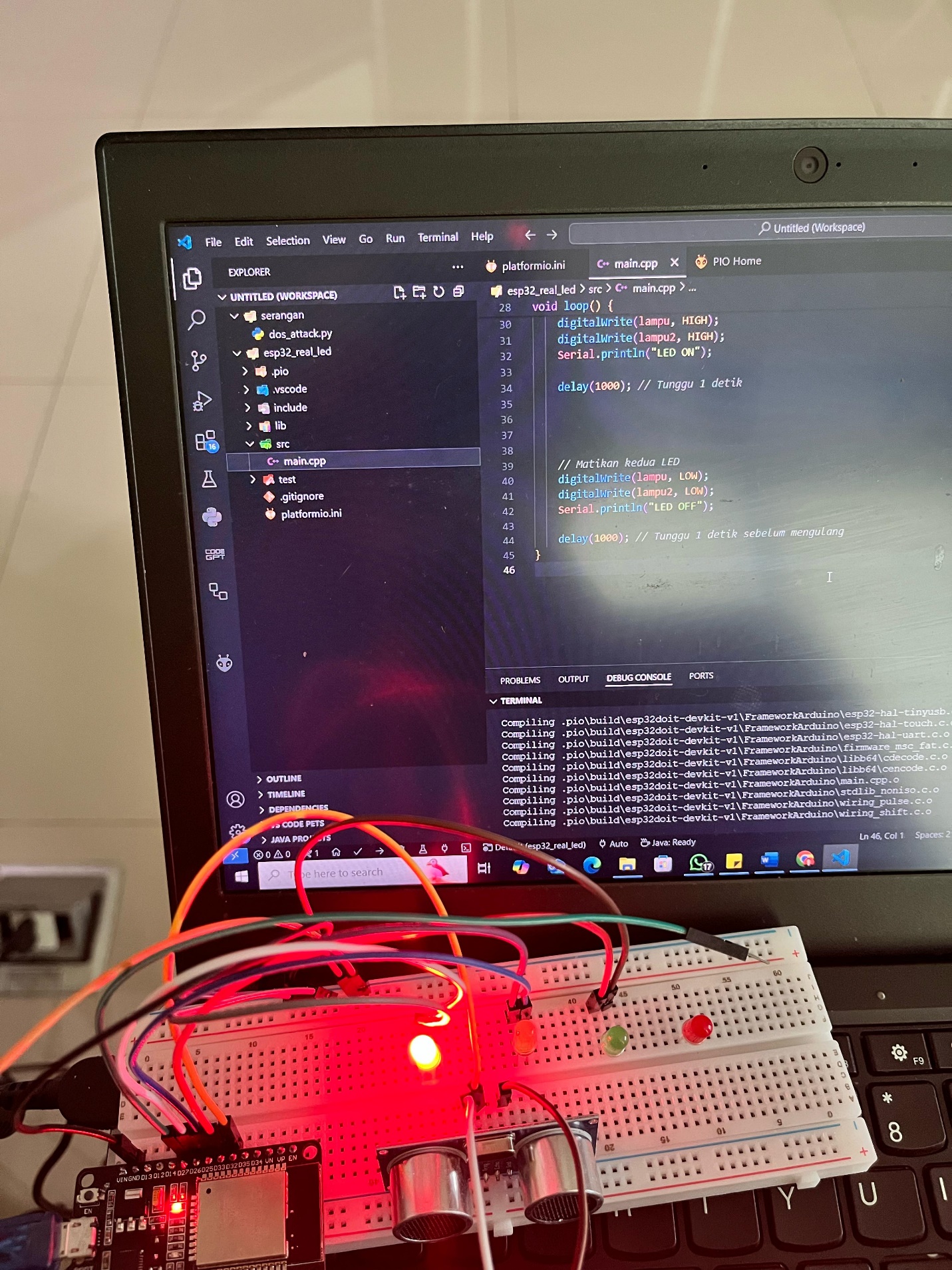
**7. Upload the Code to the ESP32**

* Click the **"Upload"** button (right-pointing arrow) at the bottom of the PlatformIO window.
* Wait for the code to compile and upload. Once done, your LEDs should start blinking as programmed.

**3. Results and Discussion**

After successfully uploading the program to the ESP32 development board via PlatformIO, the LEDs connected to GPIO pin 33 and GPIO pin 25 functioned as expected. Both LEDs turned ON simultaneously for one second, then turned OFF for another second, and continued this cycle repeatedly.

3.1 Experimental Results and Documentation



**Appendix**

**A. Links to Official Websites**

The following links were used during the experiment for account registration and platform access:

* **Wokwi**: <https://wokwi.com>
* **GitHub**: <https://github.com>
* **Platform.**io
* **Ngrok**
* **Laravel**
* **Xampp**
* **mysql**

**B. Required System Specifications**

To ensure smooth execution of the experiment, the following system requirements were met:

* **Device**: Laptop with at least 4GB RAM
* **Operating System**: Windows 10 / macOS / Linux
* **Browser**: Google Chrome (Version 100+)
* **Internet Connection**: Stable with a minimum speed of 5 Mbps