

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)
Praktik Pembuatan Akun Wokwi dan Github



Callysta Mahelina Ta'ek
Fakultas Vokasi, Universitas Brawijaya
Email: email@example.com

Abstract (Abstrak)

This experiment aims to implement a traffic light system using ESP32 in an IoT environment. The system is designed to control traffic lights through programmed timing sequences simulated on Wokwi, with code management via GitHub. The results indicate that the traffic lights operate correctly according to predefined rules, demonstrating IoT applications in smart transportation.

Keywords—Internet of Things, ESP32, Traffic Light, Wokwi, GitHub

Introduction

1.1 Background

IoT is widely applied in various fields, including transportation. A traffic light system integrated with IoT enables automated control for better traffic management. This practicum simulates a traffic light system using ESP32, Wokwi, and GitHub for development and version control.

1.2 Objectives

- Understand traffic light simulation using Wokwi.
- Learn ESP32 programming for traffic control.
- Use GitHub for code management.
- Test and evaluate system functionality.

Methodology

2.1 Tools & Materials

- Hardware: ESP32
- Software: Wokwi Simulator, Arduino IDE, GitHub
- Components: LEDs (Red, Yellow, Green), Resistors, Breadboard, Jumper Wires

2.2 Implementation Steps

1. Practicing Wokwi Platform Usage

- Watch the installation and simulation guide: YouTube Tutorial
- Access Wokwi at: <https://wokwi.com/>
- Create an account using GitHub
- Choose a starter template
- Add electronic components by clicking Add New Part

2. LED Control Practice

```
#include <Arduino.h>
```

```

int lamp1 = 26;
int lamp2 = 33;

void setup() {
    Serial.begin(115200);
    Serial.println("ESP32 Blinking LED");
    pinMode(lamp1, OUTPUT);
    pinMode(lamp2, OUTPUT);
}

void loop() {
    digitalWrite(lamp1, HIGH);
    digitalWrite(lamp2, HIGH);
    Serial.println("LED ON");
    delay(1000);

    digitalWrite(lamp1, LOW);
    digitalWrite(lamp2, LOW);
    Serial.println("LED OFF");
    delay(1000);
}

```

3. Running Simulation on VS Code (Alternative to Wokwi Web)

Due to Wokwi's free version limitations, VS Code can be used for compiling via Wokwi Simulator and PlatformIO.

1. Install Wokwi Simulator and PlatformIO extensions in VS Code.
2. Create a new PlatformIO project.
3. Configure project settings and copy the code from Wokwi to main.cpp.
4. Compile the code by clicking the checkmark button.
5. If the build button is inactive, ensure C Compiler is installed:
 - Install MinGW-w64 (Windows) or gcc/g++ (Linux/Mac).
 - Add the compiler path to Environment Variables.
 - Check PlatformIO installation via `pio --version`.
 - If issues persist, run `pio upgrade` and `pio update`.
6. Upon successful compilation, retrieve `firmware.bin` and `firmware.elf` files.
7. Update `wokwi.toml` with file paths.
8. Copy the `diagram.json` from Wokwi.
9. Request a Wokwi license: Wokwi: Request a New License.
10. Start the simulation: Wokwi: Start Simulator.

Results and Discussion

3.1 Experimental Results (Hasil Eksperimen)

The traffic light system successfully operated as programmed, following the sequence: Red → Green → Yellow. Below is the timing sequence:

1. Red : 5 Second
2. Green : 4 Second
3. Yellow : 2 Second

This experiment highlights IoT's role in traffic management and automation.

4. Appendix (Lampiran, jika diperlukan)

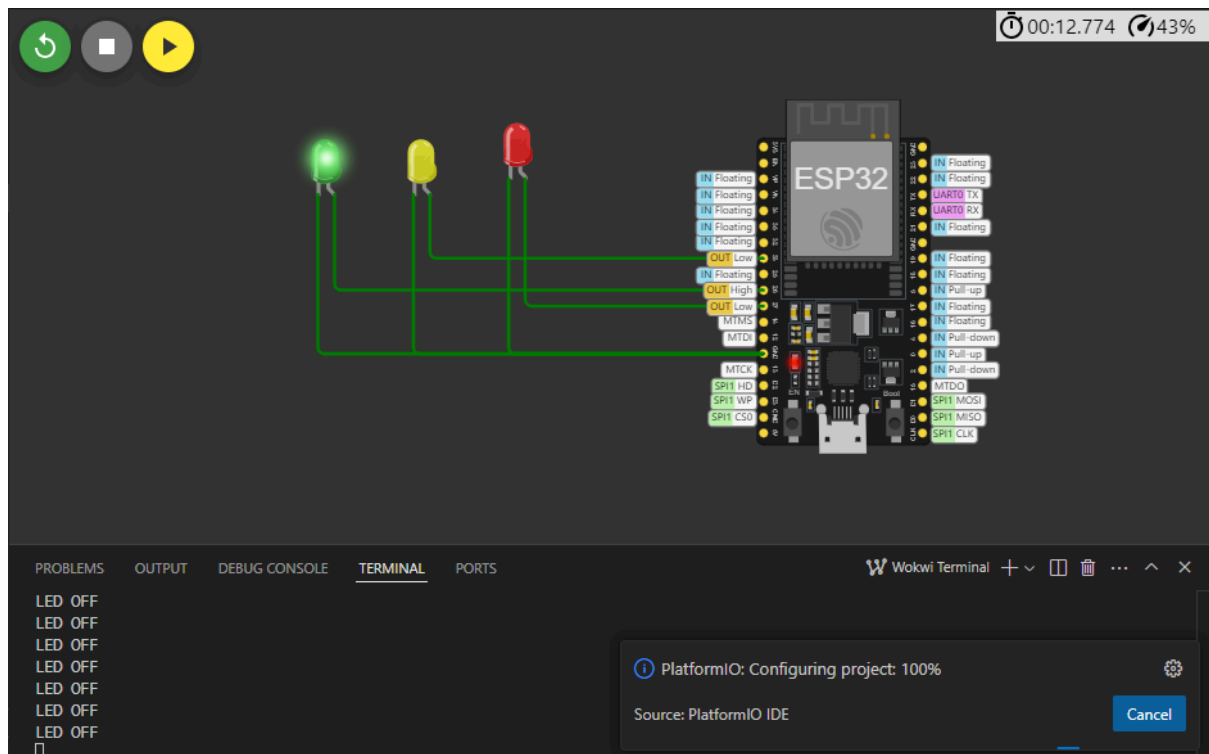
```
int red = 13;
int yellow = 12;
int green = 14;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}

void loop() {
  digitalWrite(red, HIGH);
  delay(5000);
  digitalWrite(red, LOW);

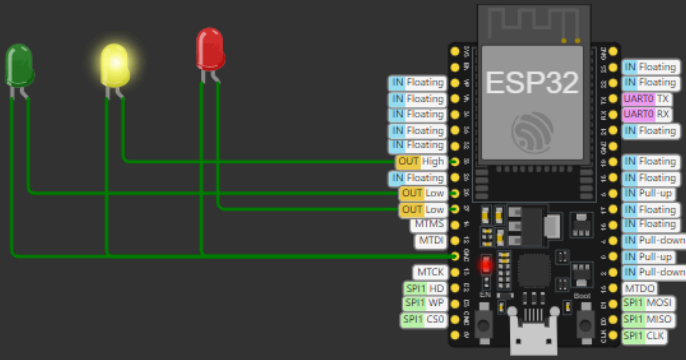
  digitalWrite(green, HIGH);
  delay(4000);
  digitalWrite(green, LOW);

  digitalWrite(yellow, HIGH);
  delay(2000);
  digitalWrite(yellow, LOW);
}
```



00:11.258 58%

Resume



The diagram shows an ESP32 microcontroller with three LEDs connected to its pins. The green LED is connected to pin 5 (IN Floating), the yellow LED to pin 6 (IN Floating), and the red LED to pin 7 (IN Floating). The pins are labeled with their functions: IN Floating, OUT High, OUT Low, MTMS, MTDI, MTCK, SPI1 HD, SPI1 WP, SPI1 CS0, IN Floating, IN Floating, UART0 TX, UART0 RX, IN Floating, IN Floating, IN Pull-up, IN Pull-down, IN Pull-up, IN Pull-down, MTDQ, SPI1 MOSI, SPI1 MISO, and SPI1 CLK.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

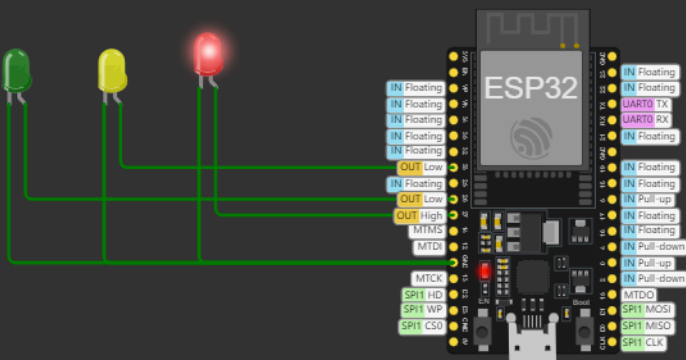
LED OFF
LED OFF
LED OFF
LED OFF
LED OFF
LED OFF
LED OFF

PlatformIO: Configuring project
Source: PlatformIO IDE

Cancel

00:11.591 37%

Resume



The diagram shows an ESP32 microcontroller with three LEDs connected to its pins. The green LED is connected to pin 5 (IN Floating), the yellow LED to pin 6 (IN Floating), and the red LED to pin 7 (IN Floating). The pins are labeled with their functions: IN Floating, OUT Low, OUT Low, OUT High, MTMS, MTDI, MTCK, SPI1 HD, SPI1 WP, SPI1 CS0, IN Floating, IN Floating, UART0 TX, UART0 RX, IN Floating, IN Floating, IN Pull-up, IN Pull-down, IN Pull-up, IN Pull-down, MTDQ, SPI1 MOSI, SPI1 MISO, and SPI1 CLK.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

LED OFF
LED OFF
LED OFF
LED OFF
LED OFF
LED OFF
LED OFF

PlatformIO: Configuring project: 80%
Source: PlatformIO IDE

Cancel