**ESP32 Simulation Practice and** **Temperature Humidity Sensor**

*Agatha Herma Putra*

*Fakultas Vokasi, Universitas Brawijaya*

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

**Abstract**

This report presents the implementation of ESP32 simulation practice with a temperature and humidity sensor using Wokwi, Visual Studio Code (VS Code), and PlatformIO. The study aims to explore the integration of ESP32 with environmental sensors in a simulated environment to facilitate IoT-based applications. The simulation was conducted using Wokwi, a web-based simulation tool that enables the testing of embedded systems without physical hardware. The programming and debugging processes were carried out in VS Code with the PlatformIO extension, which provides an efficient development environment for ESP32. The temperature and humidity sensor used in this study was successfully integrated with the ESP32, and real-time data readings were observed through serial communication. The results indicate that Wokwi is a viable alternative for prototyping ESP32-based projects before deploying them on physical hardware. This approach enhances the development process by reducing hardware dependency and facilitating debugging.

*Keywords : ESP32, Simulation, Temperature Sensor, Humidity Sensor, Wokwi, VS Code, PlatformIO, IoT Development.*

**1 Introduction**

**1.1 Background**

The Internet of Things (IoT) has become an essential field in modern technology, allowing devices to communicate and exchange data. The ESP32 microcontroller is widely used in IoT applications due to its low power consumption and built-in Wi-Fi and Bluetooth capabilities. However, testing and debugging ESP32 projects can be challenging without proper tools. This study explores how Wokwi, an online simulation platform, can be used to test ESP32-based projects before implementation on actual hardware. By using VS Code and PlatformIO, developers can efficiently write, debug, and simulate their ESP32 projects.

**1.2 Purpose of the experiment**

The primary purpose of this experiment is to simulate the functionality of an ESP32-based temperature and humidity monitoring system using Wokwi. This approach allows for testing sensor integration, data communication, and debugging in a virtual environment before deploying the project to real hardware. Additionally, this study aims to demonstrate the effectiveness of using VS Code and PlatformIO as development tools for ESP32 programming.

**2.1 Methodology**

The simulation was conducted using Wokwi, an online platform that provides a virtual environment for microcontroller projects. VS Code was used as the primary development environment with the PlatformIO extension for ESP32 programming. A temperature and humidity sensor was integrated into the Wokwi simulation, and a simple program written in C++ was used to read and process the sensor data. Serial communication was utilized to observe real-time sensor readings, ensuring proper data transmission. Debugging was performed using PlatformIO tools within VS Code to identify and fix potential issues. Finally, the simulation results were analyzed to verify the correct functionality of the sensor readings and ESP32 integration before implementing the project on actual hardware

**2.2 Tools & Materials**

To successfully complete the account creation process on Wokwi and GitHub, several tools and materials are required. These include:

1. Hardware Requirements
   * A computer or laptop with an internet connection
   * A web browser (Google Chrome, Mozilla Firefox, Microsoft Edge, or any other compatible browser)
2. Software & Online Platforms
   * Wokwi (accessible at <https://wokwi.com>)
   * Microsoft Visual Studio Code
   * Platform.io
3. Additional Tools (Optional)
   * A text editor (such as Visual Studio Code or Notepad++) for working with GitHub repositories
   * A Git client (such as Git Bash or GitHub Desktop) for version control testing

These tools and materials ensure a smooth and efficient process for creating accounts and exploring the basic functionalities of both platforms.

**2.3 Implemention Steps**

Implementation for experiment making ESP32 Simulation Practice and Temperature Humidity Sensor:

**1. Accessing the Platforms**

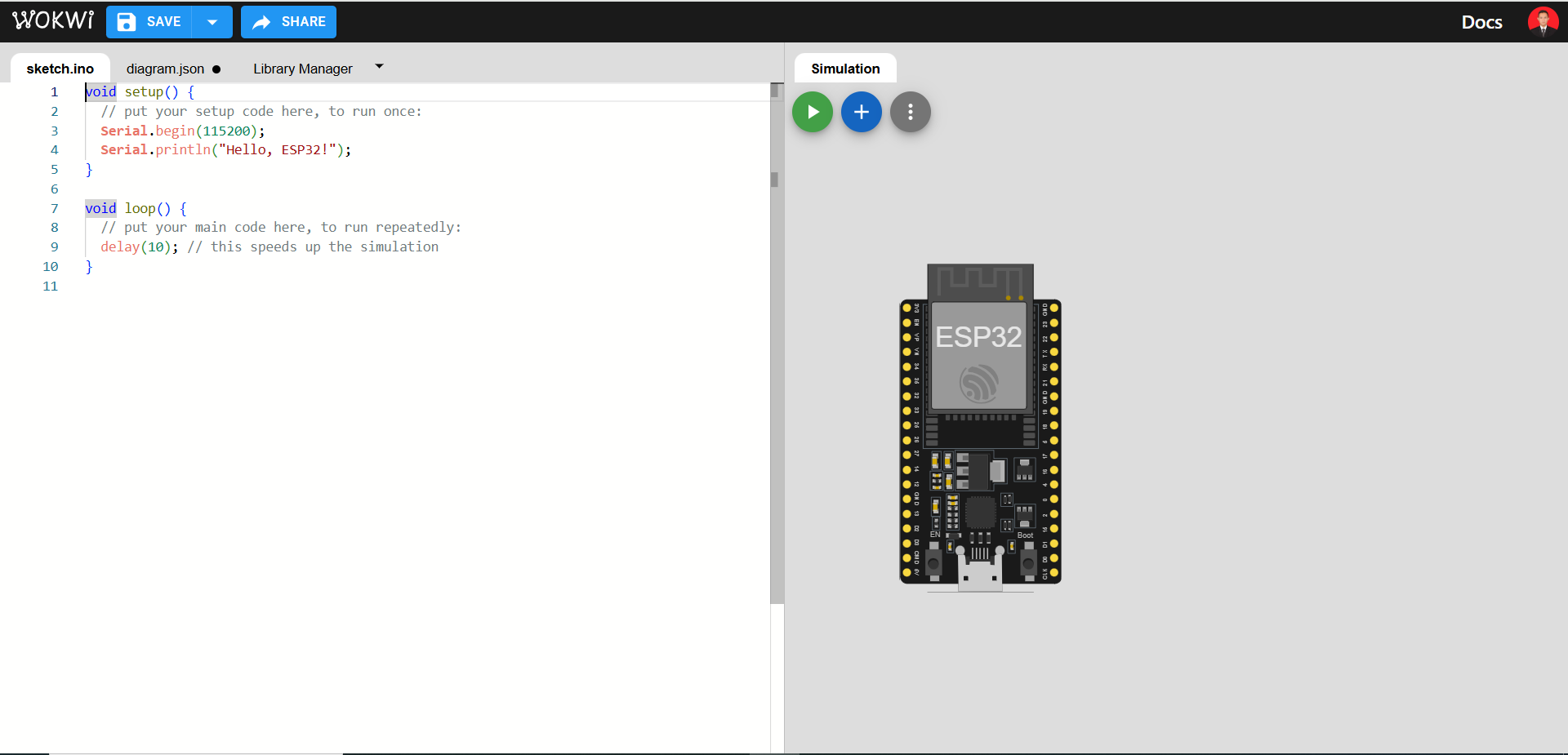
* Open a web browser on a computer or laptop.
* Navigate to the official websites:
  + **Wokwi**: <https://wokwi.com>

**2. Sign Up Wokwi Account**

* Click on the **"Sign Up"** button.
* Enter the required details, including:
  + A valid email address
  + A username
* Click **"Sign Up"** to proceed.
* Log in to the Wokwi account and explore the available features, such as microcontroller simulations.

**3. Create Starter Template using esp32**

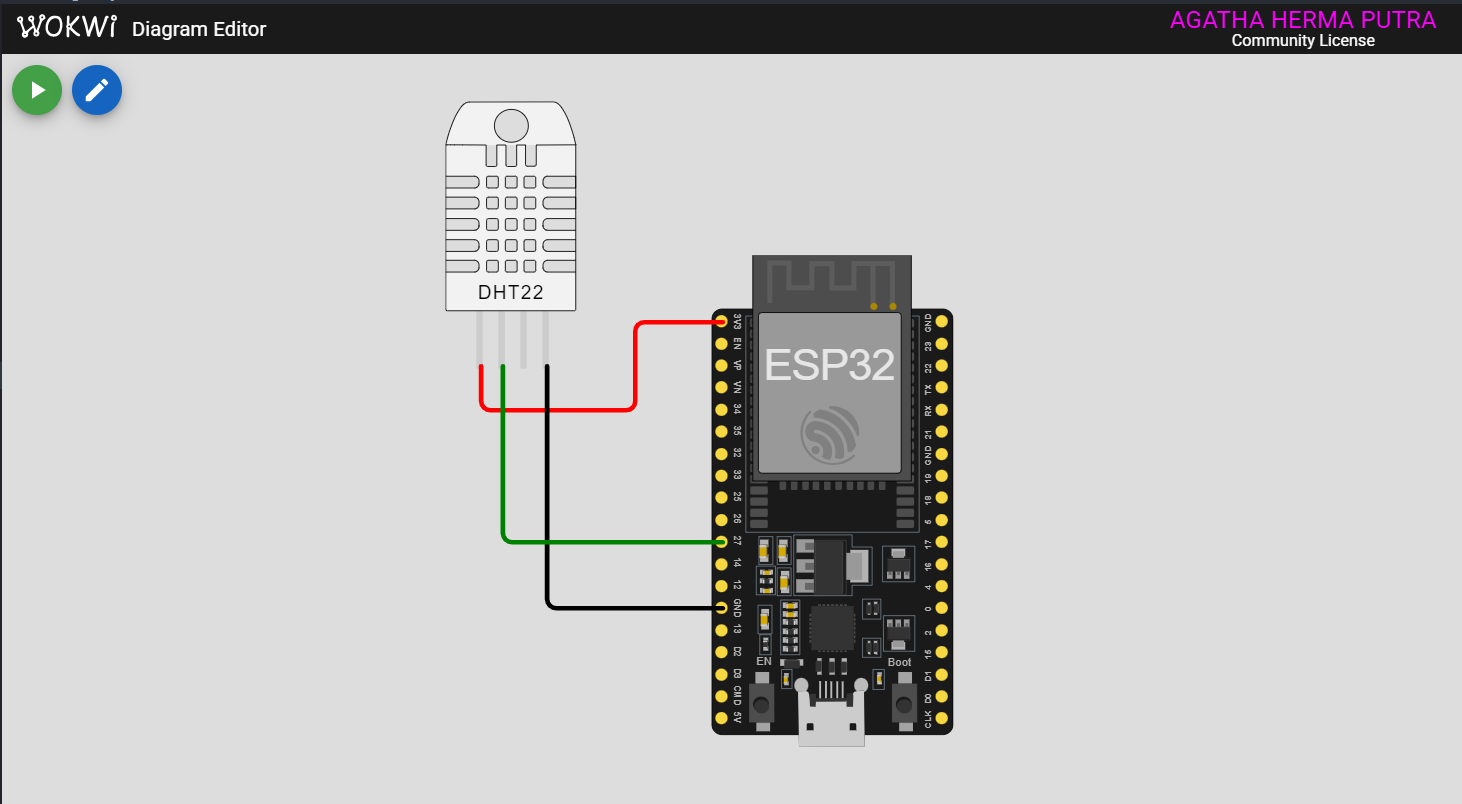
* Open Wokwi on the initial display or homepage.
* On the main page select ESP32 controller
* Scroll down until you find the starter template.
* Select on ESP 32 template.



ESP 32 Starter template main view

**4. Added electronic components to the ESP 32**

* In the worksheet click on the plus button.
* In the plus sign section, add an additional tool to the Wokwi, namely a humidity sensor with the tool code DHT 22
* And connect each DHT 22 to the ESP32 according to the following picture



DHT 22 Display and Cable Placement

**5. Enter the coding to run the light command**

* Copy the following code and paste it into wokwi simulator or VS Code to run the Temperature Humidity Sensor, in file main.cpp
* #include <Arduino.h>
* #include <DHT.h>
* #define DHTPIN 27 *// Pin yang terhubung ke sensor DHT22*
* #define DHTTYPE DHT22 *// Tipe sensor DHT*
* DHT dht(DHTPIN, DHTTYPE);
* void setup()
* {
* Serial.begin(115200);
* dht.begin(); *// Inisialisasi sensor*
* }
* void loop()
* {
* delay(2000); *// Delay antar pembacaan*
* float humidity = dht.readHumidity();
* float temperature = dht.readTemperature();
* *// Cek apakah pembacaan gagal*
* if (isnan(humidity) || isnan(temperature))
* {
* Serial.println("Gagal membaca sensor!");
* return;
* }
* *// Tampilkan hasil pembacaan*
* Serial.print("Kelembaban: ");
* Serial.print(humidity);
* Serial.print(" %\t");
* Serial.print("Suhu: ");
* Serial.print(temperature);
* Serial.println(" \*C");
* }

Coding to execute commands

* In the next step, create a file called diagram.json, then paste the code from the wokwi website into the digram.json file, which will automatically change the appearance of the diagram.json file.
* "version": 1,
* "author": "vokasi",
* "editor": "wokwi",
* "parts": [
* {
* "type": "board-esp32-devkit-c-v4",
* "id": "esp",
* "top": 86.4,
* "left": 24.04,
* "attrs": {}
* },
* {
* "type": "wokwi-dht22",
* "id": "dht1",
* "top": 19.5,
* "left": -91.8,
* "attrs": {}
* }
* ],
* "connections": [
* [
* "esp:TX",
* "$serialMonitor:RX",
* "",
* []
* ],
* [
* "esp:RX",
* "$serialMonitor:TX",
* "",
* []
* ],
* [
* "dht1:VCC",
* "esp:3V3",
* "red",
* [
* "v19.2",
* "h67.2",
* "v-67.2"
* ]
* ],
* [
* "esp:GND.1",
* "dht1:GND",
* "black",
* [
* "h0"
* ]
* ],
* [
* "dht1:SDA",
* "esp:27",
* "green",
* [
* "v0"
* ]
* ]
* ],
* "dependencies": {}
* }

Diagram.json

* After that, create a file called wokwi.toml which is useful for detecting files with the contents of the file

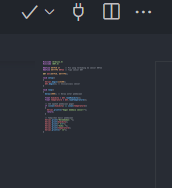
[wokwi]

version = 1

firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'

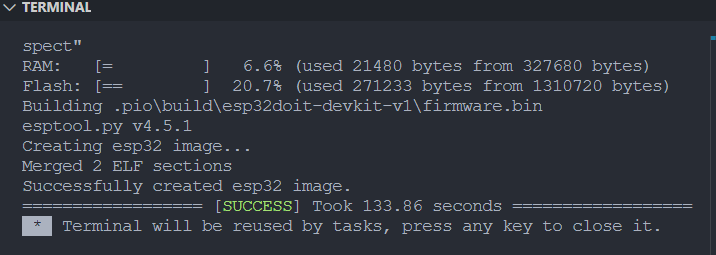
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

* After everything has been set, make sure that main.cpp is compiled first by pressing the check button in the top right corner of the file.



Check Button

* If the terminal display says SUCCESS then the program is ready to run.



Program Ready to Run

**3. Results and Discussion**

**3.1 Experimental Results**

This experiment involved creating a Traffic Light Scheme in Wokwi and VSCode and testing the basic functionality of the account. The following results were obtained:

* **Schema Development**

The traffic light schema was successfully created in Wokwi and VSCode.

* **Functionality Testing**

Basic functionality of the account was tested and verified.

* **Code Implementation**

The traffic light program was executed as designed.

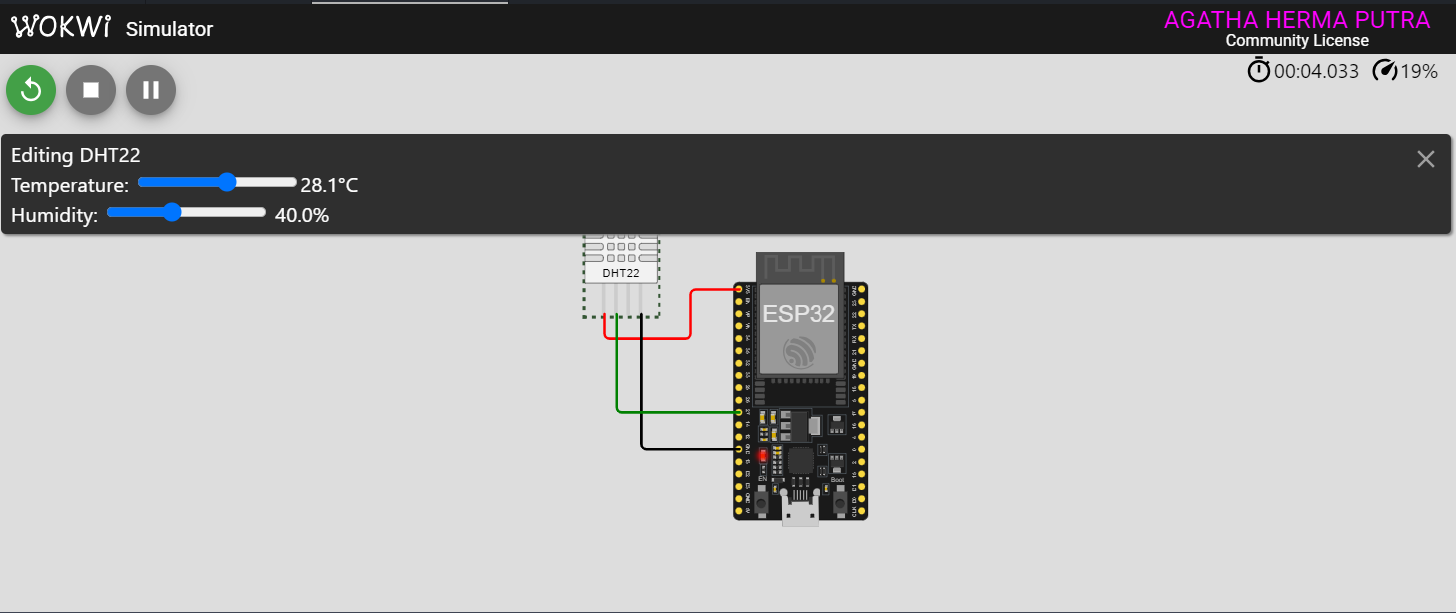
* **Observation Results**

The system functioned as expected, following traffic light rules.

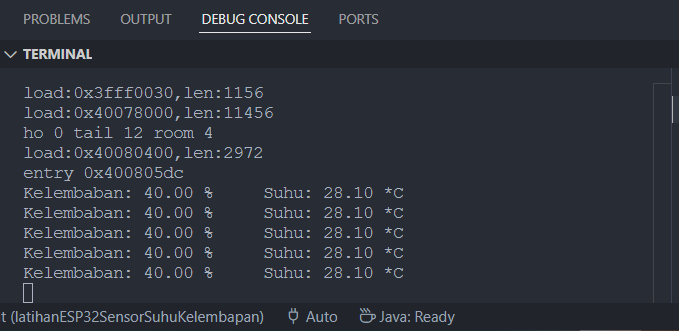
* **Evaluation**

No major issues were found, but further optimization is needed for code efficiency.

**3.2 Performance and User Experience Documentation**

****

Enable to editing the temperature and Humidity

****

Result in Terminal

**4. 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

**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