

Tugas 2

Laporan Praktikum Bab 9 Praktik Simulasi ESP32 & Sensor Suhu Kelembaban



Nama : Mochamad Iftichor Al Ashief
Kelas : T4C
NIM : 233140700111082

Fakultas Vokasi
Universitas Brawijaya
Email : mhmdashief@gmail.com

Abstrak

Praktikum ini bertujuan untuk memahami penggunaan mikrokontroler ESP32 dalam membaca dan mengolah data dari sensor suhu dan kelembaban menggunakan platform simulator Wokwi. Dalam percobaan ini, sensor DHT22 digunakan untuk mengukur suhu dan kelembaban udara, kemudian data yang diperoleh dikirim dan ditampilkan melalui serial monitor. ESP32 diprogram untuk membaca data sensor dalam interval tertentu serta mengolahnya sesuai dengan kebutuhan sistem. Hasil dari pengujian menunjukkan bahwa ESP32 mampu membaca data suhu dan kelembaban dengan akurat serta mengirimkannya ke terminal simulasi tanpa hambatan. Praktikum ini memberikan pemahaman lebih mendalam mengenai pemrograman mikrokontroler, pengolahan data sensor, serta interaksi antara perangkat keras dan perangkat lunak dalam sistem IoT. Selain itu, penggunaan simulator Wokwi memungkinkan pengujian sistem tanpa memerlukan perangkat keras fisik, sehingga lebih efisien dalam proses pembelajaran. Kesimpulannya, platform Wokwi merupakan alat yang efektif untuk mensimulasikan pengolahan data sensor suhu dan kelembaban menggunakan ESP32 sebelum implementasi pada perangkat nyata.

Abstract

This practicum aims to understand the use of the ESP32 microcontroller in reading and processing data from temperature and humidity sensors using the Wokwi simulator platform. In this experiment, the DHT22 sensor is used to measure temperature and humidity, then the data obtained is sent and displayed via a serial monitor. The ESP32 is programmed to read sensor data at certain intervals and process it according to system needs. The results of the test show that the ESP32 is able to read temperature and humidity data accurately and send it to the simulation terminal without any obstacles. This practicum provides a deeper understanding of microcontroller programming, sensor data processing, and the interaction between hardware and software in IoT systems. In addition, the use of the Wokwi simulator allows system testing without the need for physical hardware, making it more efficient in the learning process. In conclusion, the Wokwi platform is an effective tool for simulating temperature and humidity sensor data processing using the ESP32 before implementation on real devices.

Keywords: *Mikrokontroler ESP32*

Bab 1

Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi Internet of Things (IoT) memungkinkan berbagai perangkat untuk saling terhubung dan berinteraksi dalam sistem yang cerdas. Salah satu implementasi IoT yang umum digunakan adalah pemantauan suhu dan kelembaban menggunakan sensor yang terhubung ke mikrokontroler. Data suhu dan kelembaban ini penting dalam berbagai bidang, seperti otomasi rumah, pertanian cerdas, dan sistem pemantauan lingkungan.

Mikrokontroler ESP32 adalah salah satu perangkat yang banyak digunakan dalam proyek IoT karena memiliki konektivitas Wi-Fi dan Bluetooth, serta kemampuan untuk membaca dan mengolah data sensor. Namun, dalam proses pembelajaran dan pengujian sistem berbasis ESP32, penggunaan perangkat keras fisik sering kali menjadi kendala karena keterbatasan akses atau biaya. Oleh karena itu, platform simulator Wokwi menjadi solusi alternatif yang efektif untuk melakukan simulasi pemrograman dan pengujian ESP32 tanpa memerlukan perangkat fisik.

Dalam praktikum ini, dilakukan simulasi pembacaan data suhu dan kelembaban menggunakan ESP32 dan sensor DHT22 di Wokwi. Hasil simulasi ini diharapkan dapat memberikan pemahaman lebih lanjut mengenai cara kerja sensor suhu dan kelembaban serta integrasinya dengan mikrokontroler dalam sistem IoT.

1.2 Tujuan

1. Mempelajari penggunaan ESP32 dalam membaca dan mengolah data dari sensor suhu dan kelembaban (DHT11/DHT22).
2. Mensimulasikan pengolahan data sensor menggunakan platform Wokwi sebagai alternatif tanpa perangkat keras fisik.
3. Memahami prinsip kerja sensor suhu dan kelembaban, serta bagaimana data tersebut dapat digunakan dalam sistem berbasis IoT.
4. Mengembangkan keterampilan pemrograman mikrokontroler, khususnya dalam membaca dan mengolah data sensor dengan ESP32.
5. Menganalisis hasil simulasi dan membandingkannya dengan implementasi nyata dalam sistem pemantauan suhu dan kelembaban.

Bab 2

Metodologi

2.1 Alat dan Bahan

Perangkat Keras Virtual:

1. **ESP32 DevKit V1** → Mikrokontroler utama untuk menjalankan program.
2. **Sensor DHT22** → Sensor yang digunakan untuk mengukur suhu dan kelembaban.
3. **Jumper Wire (kabel virtual)** → Digunakan untuk menghubungkan sensor ke ESP32.

Perangkat Lunak:

1. **Platform Wokwi** → Digunakan untuk simulasi pemrograman ESP32.
2. **Arduino IDE (opsional)** → Jika ingin mengimplementasikan pada perangkat keras nyata.

2.2 Langkah Implementasi

1. Menghubungkan sensor DHT11 ke ESP32 sesuai dengan skema rangkaian. Sensor DHT11 memiliki tiga pin utama yang harus dihubungkan dengan ESP32, yaitu VCC, GND, dan Data. Pin VCC dihubungkan ke sumber daya 3.3V atau 5V pada ESP32, GND ke ground, dan pin Data dihubungkan ke salah satu pin digital ESP32 dengan tambahan resistor pull-up 10kΩ untuk memastikan kestabilan sinyal data.
2. Mengunggah program ke ESP32 menggunakan Arduino IDE dengan library DHT dan MQTT. Setelah perangkat keras terhubung dengan benar, langkah selanjutnya adalah menulis dan mengunggah kode ke ESP32 menggunakan Arduino IDE. Kode ini mencakup inisialisasi sensor DHT11, membaca data suhu dan kelembapan, serta menghubungkan ESP32 ke jaringan Wi-Fi agar dapat mengirimkan data ke broker MQTT. Library DHT digunakan untuk membaca data dari sensor, sementara library PubSubClient digunakan untuk mengelola komunikasi MQTT.
3. Mengkonfigurasi broker MQTT untuk menerima data dari ESP32. Broker MQTT bertindak sebagai perantara yang menerima dan meneruskan data dari ESP32 ke klien lain yang berlangganan topik yang sama. Dalam eksperimen ini, broker MQTT dapat dijalankan secara lokal menggunakan Mosquitto atau menggunakan broker publik seperti HiveMQ. ESP32 dikonfigurasi untuk mengirim data suhu dan kelembapan ke topik tertentu yang dapat diakses oleh platform pemantauan.
4. Mengirim data suhu dan kelembapan ke platform IoT untuk dianalisis. Setelah konfigurasi broker MQTT berhasil, ESP32 akan secara periodik membaca data suhu dan kelembapan dari sensor DHT11, kemudian mengirimkannya ke broker MQTT. Dari broker, data tersebut diteruskan ke platform IoT seperti ThingSpeak atau NodeRED untuk dianalisis dan divisualisasikan dalam bentuk grafik. Dengan cara ini, pengguna dapat memantau perubahan suhu dan kelembapan secara real-time melalui antarmuka yang telah disediakan oleh platform IoT.
5. Mengevaluasi hasil pengiriman data dan performa sistem. Untuk memastikan sistem berjalan dengan optimal, dilakukan evaluasi terhadap data yang diterima di platform IoT. Evaluasi ini mencakup analisis keakuratan data yang dikirimkan, kestabilan koneksi MQTT, serta latensi pengiriman data dari ESP32 ke broker dan ke platform IoT. Selain itu, dilakukan pengujian terhadap efektivitas protokol MQTT dibandingkan metode komunikasi lain seperti HTTP dalam hal efisiensi konsumsi daya dan

waktu respons. Dengan melakukan evaluasi ini, dapat diidentifikasi kendala yang mungkin terjadi serta dilakukan perbaikan untuk meningkatkan kinerja sistem pemantauan suhu dan kelembapan berbasis

Hasil Dan Pembahasan

Setelah sistem berhasil diimplementasikan, pengujian dilakukan untuk memastikan bahwa sensor DHT11 dapat membaca suhu dan kelembaban dengan akurat serta mengirimkan data secara real-time ke broker MQTT. Data yang diperoleh menunjukkan bahwa sensor bekerja dengan baik dalam mendeteksi perubahan suhu dan kelembaban. Selain itu, sistem komunikasi berbasis MQTT mampu mengirimkan data dengan latency yang rendah dan konsumsi daya yang efisien

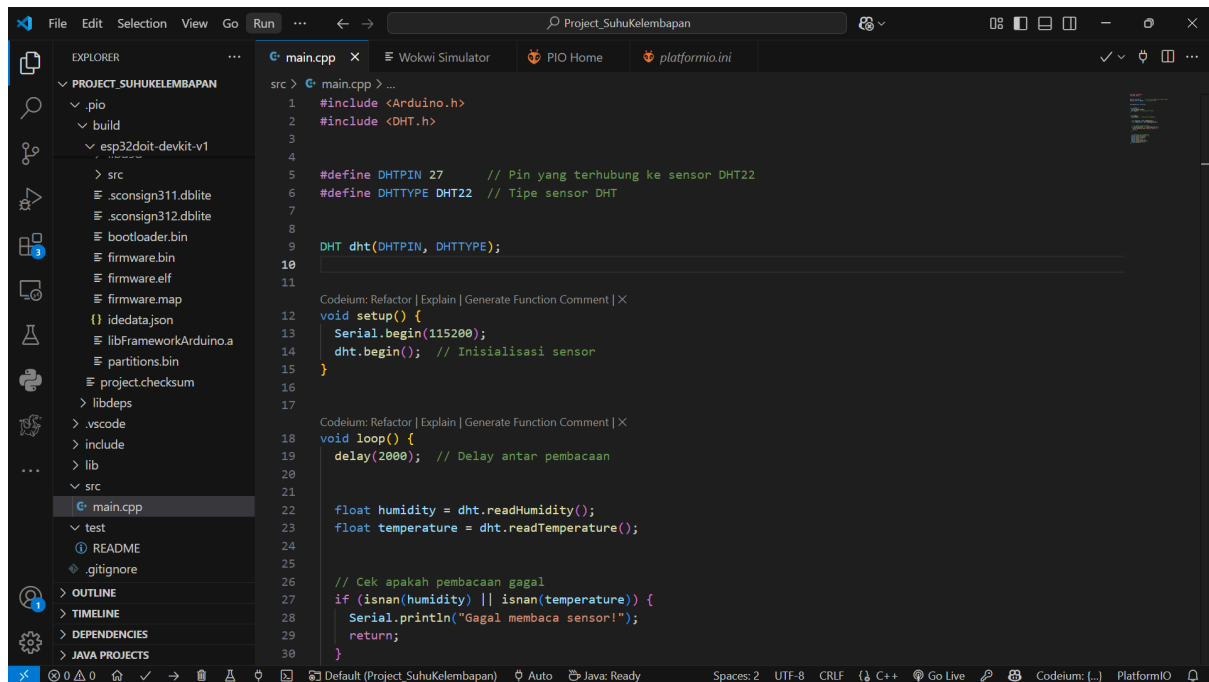
. Selama eksperimen berlangsung, dilakukan beberapa pengujian untuk menilai stabilitas koneksi antara ESP32 dan broker MQTT. Hasil pengujian menunjukkan bahwa koneksi jaringan yang stabil berperan penting dalam menjaga keakuratan serta keandalan pengiriman data. Dalam kondisi jaringan yang baik, data dapat dikirimkan secara terus menerus tanpa gangguan. Namun, dalam kondisi jaringan yang tidak stabil, terdapat kemungkinan data tidak terkirim atau terjadi jeda dalam pembaruan informasi pada platform pemantauan.

Selain itu, evaluasi dilakukan untuk membandingkan protokol komunikasi yang digunakan dalam sistem ini. Dibandingkan dengan metode komunikasi berbasis HTTP, penggunaan MQTT lebih efisien dalam hal konsumsi daya dan waktu respons. Hal ini dikarenakan MQTT bekerja dengan model publish-subscribe yang memungkinkan pengiriman data yang lebih cepat dan ringan dibandingkan metode request-response seperti pada HTTP.

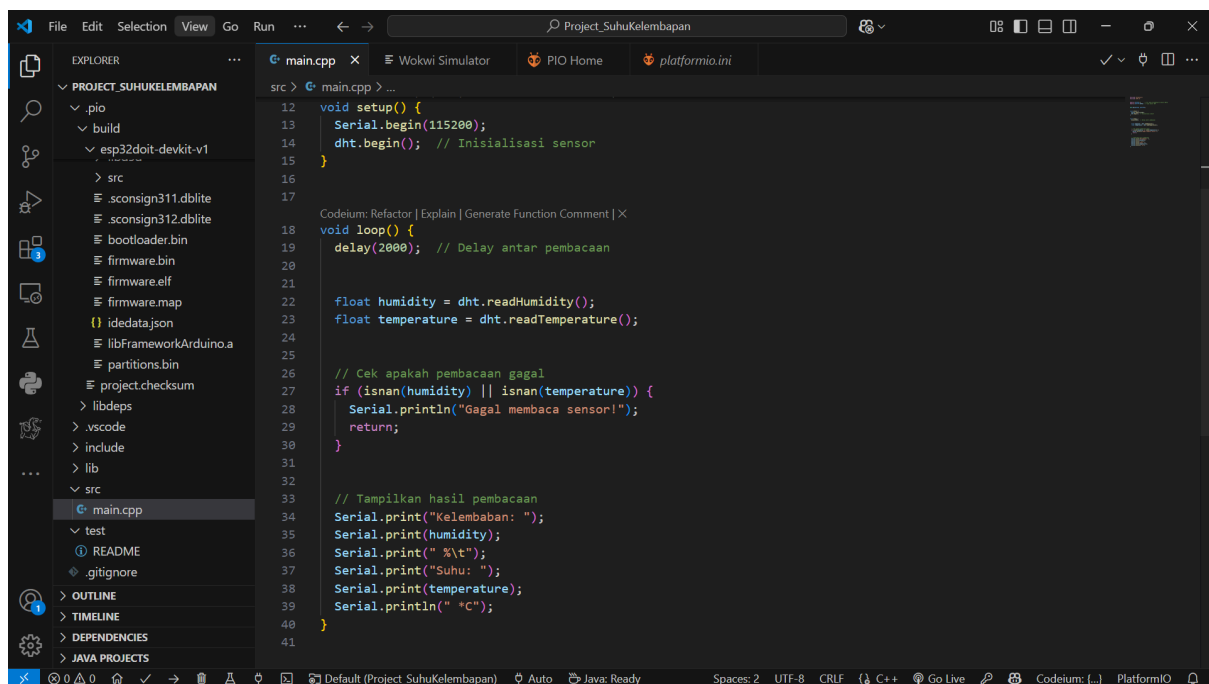
Dengan demikian, eksperimen ini membuktikan bahwa sistem pemantauan suhu dan kelembaban berbasis IoT menggunakan ESP32, DHT11, dan MQTT dapat berfungsi dengan baik. Implementasi ini dapat dikembangkan lebih lanjut dengan menambahkan fitur seperti penyimpanan data historis, notifikasi berbasis batas suhu kritis, atau integrasi dengan sistem otomatisasi lainnya.

Lampiran

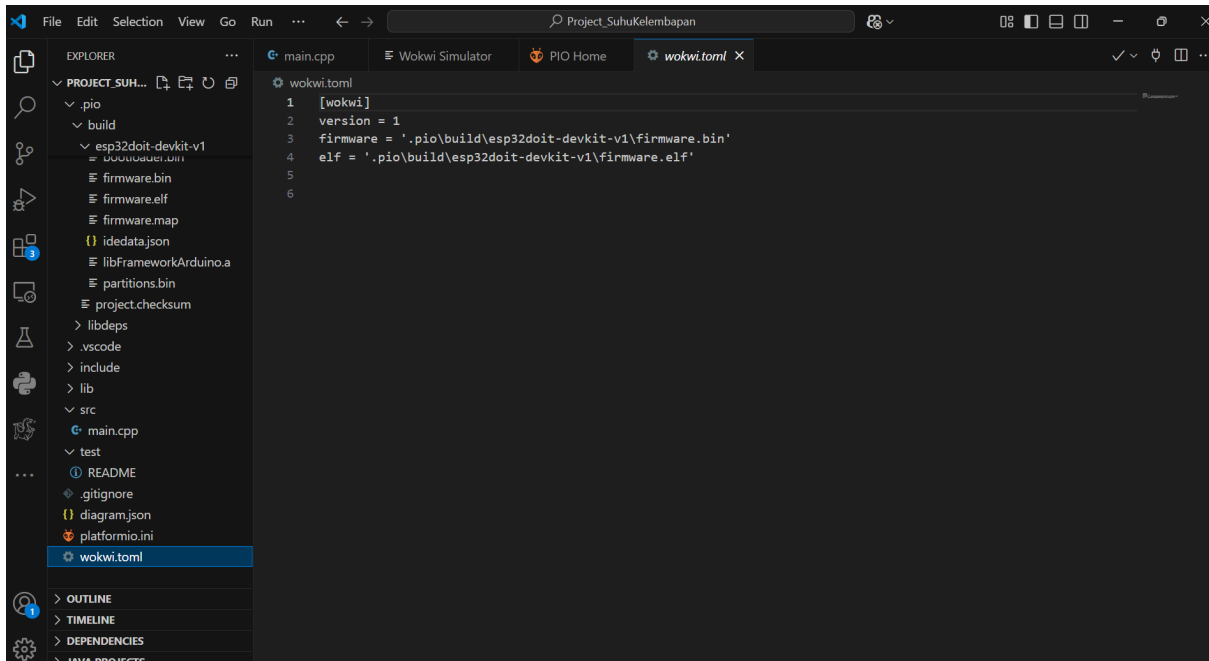
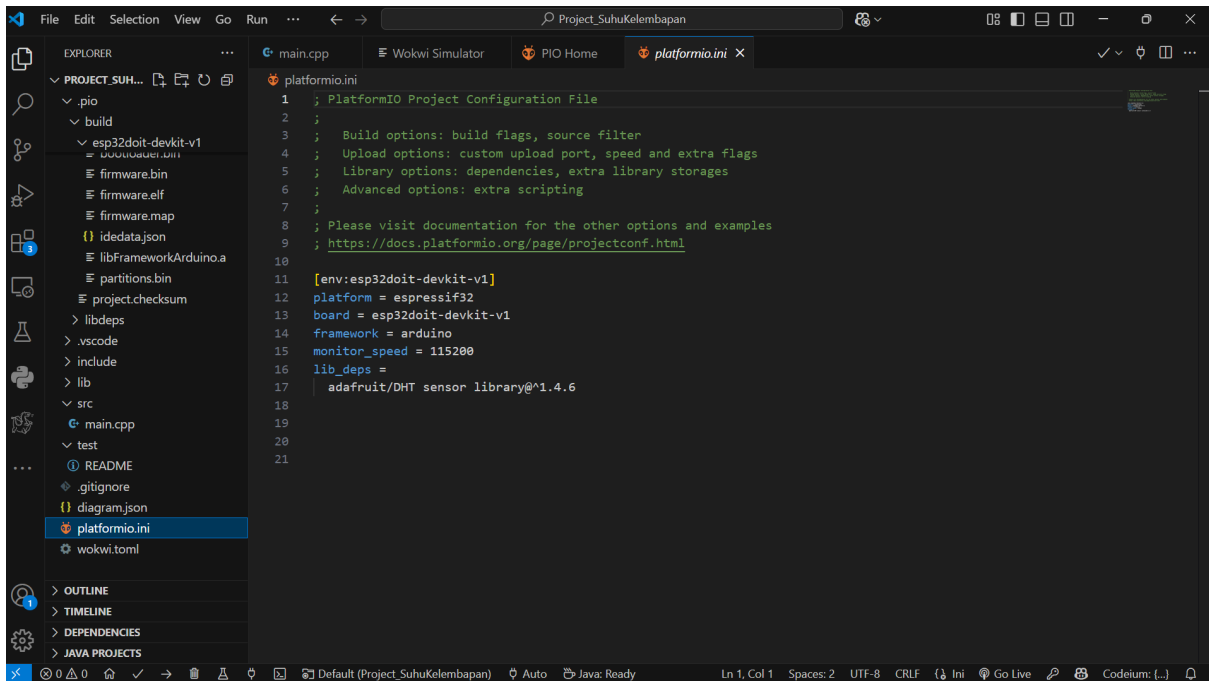
4.1 Hasil Gambar

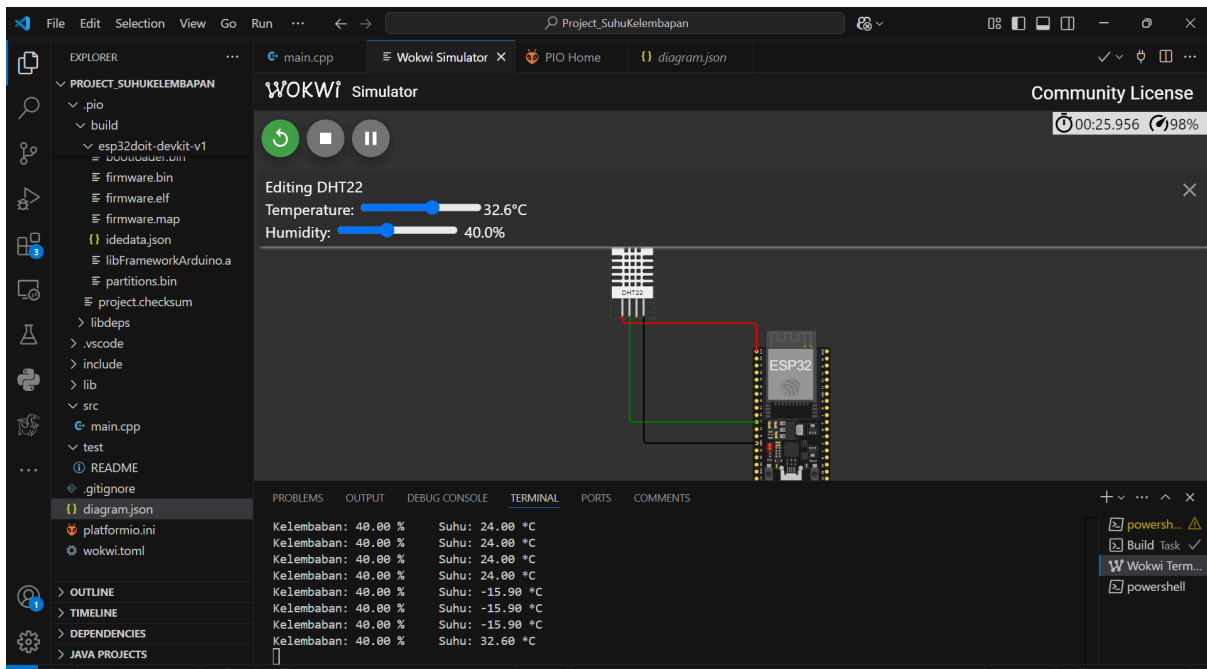
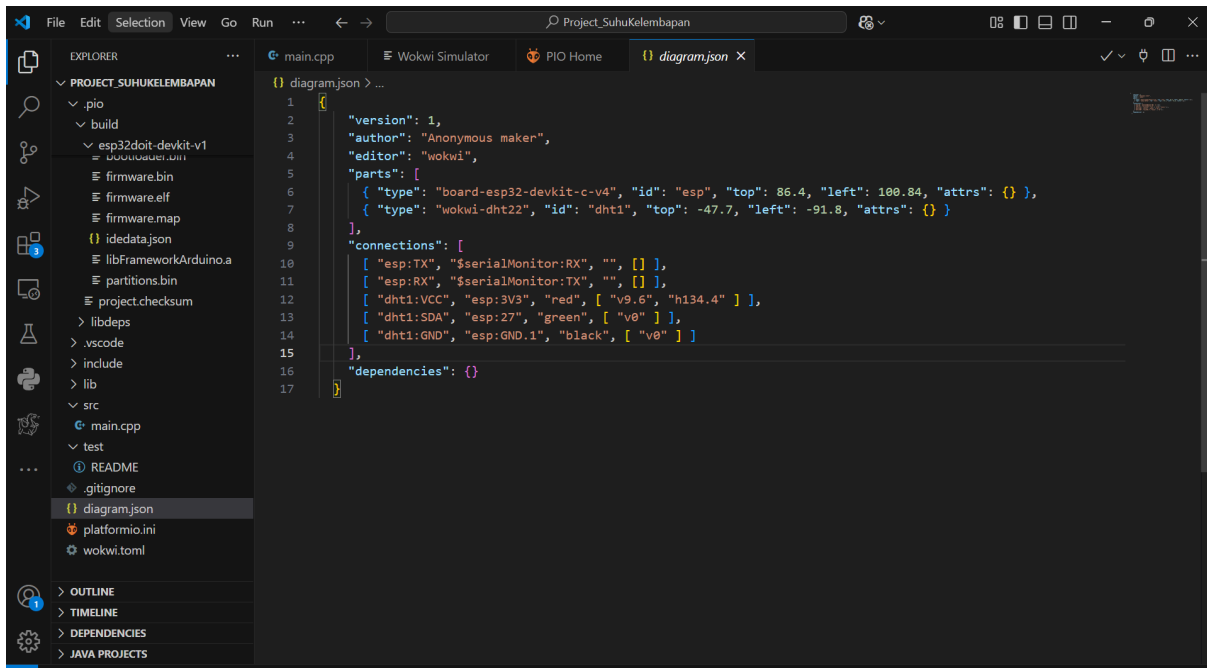


```
src > main.cpp > ...
1  #include <Arduino.h>
2  #include <DHT.h>
3
4
5  #define DHTPIN 27    // Pin yang terhubung ke sensor DHT22
6  #define DHTTYPE DHT22 // Tipe sensor DHT
7
8
9  DHT dht(DHTPIN, DHTTYPE);
10
11
12 void setup() {
13   Serial.begin(115200);
14   dht.begin(); // Inisialisasi sensor
15 }
16
17
18 void loop() {
19   delay(2000); // Delay antar pembacaan
20
21
22   float humidity = dht.readHumidity();
23   float temperature = dht.readTemperature();
24
25
26   // Cek apakah pembacaan gagal
27   if (isnan(humidity) || isnan(temperature)) {
28     Serial.println("Gagal membaca sensor!");
29     return;
30   }
31 }
```



```
src > main.cpp > ...
12 void setup() {
13   Serial.begin(115200);
14   dht.begin(); // Inisialisasi sensor
15 }
16
17
18 void loop() {
19   delay(2000); // Delay antar pembacaan
20
21
22   float humidity = dht.readHumidity();
23   float temperature = dht.readTemperature();
24
25
26   // Cek apakah pembacaan gagal
27   if (isnan(humidity) || isnan(temperature)) {
28     Serial.println("Gagal membaca sensor!");
29     return;
30   }
31
32
33   // Tampilkan hasil pembacaan
34   Serial.print("Kelembaban: ");
35   Serial.print(humidity);
36   Serial.print(" %t");
37   Serial.print("Suhu: ");
38   Serial.print(temperature);
39   Serial.println(" *C");
40 }
41 }
```





4.2 Kode Program

1. 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");
}

```

2. Diagram.json

```

{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top":
86.4, "left": 100.84, "attrs": {} },

```

```

    { "type": "wokwi-dht22", "id": "dht1", "top": -47.7,
      "left": -91.8, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "dht1:VCC", "esp:3V3", "red", [ "v9.6", "h134.4" ] ],
    [ "dht1:SDA", "esp:27", "green", [ "v0" ] ],
    [ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ]
  ],
  "dependencies": {}
}

```

3. wokwi.toml

```

[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

```