SIMULASI PEMANTAUAN SUHU DAN KELEMBABAN BERBASIS IOT MENGGUNAKAN ESP32 dan NODERED

Feby Kurnia Putri

Teknologi Informasi, Fakultas Vokasi, Universitas Brawijaya Jl. Veteran No.10-11, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145

febystudentub@gmail.com

Abstrack -- This practicum is a design and simulation practicum of an Internet of Things (IoT)-based temperature and humidity monitoring system using an ESP32 microcontroller and a DHT22 sensor, developed virtually through the Wokwi platform and Visual Studio Code. The system uses broker broker.emqx.io. The data generated by the sensor is sent through topics with two topics namely humidity and temperature and then visualized using Node-RED. Node-RED plays a role in managing data flow, starting from receiving sensor data, processing, storing using InfluxDB as a time-series database, and presenting data visually in the form of graphs and indicators (gauges) from both topics. In addition, the system is also able to receive incoming data from brokers, for example to turn on or off the LED as a form of two-way interaction (publish-subscribe). This practicum is done without physical devices so that it can make it easier to do experiments before going down to real hardware. The simulation results show that the system runs stably, responsively, and can handle sending and receiving data in real-time.

Keyword : ***Internet of Things, Wokwi, NodeRed, DHT22***

Abstrak **--** Praktikum ini merupakan praktikum perancangan dan simulasi sistem monitoring suhu dan kelembapan berbasis Internet of Things (IoT) menggunakan mikrokontroler ESP32 dan sensor DHT22, yang dikembangkan secara virtual melalui platform Wokwi dan Visual Studio Code. Sistem menggunakan broker broker.emqx.io. Data yang dihasilkan oleh sensor dikirim melalui topik dengan terdapat dua topik yakni humidity dan temperature lalu divisualisasikan menggunakan Node-RED. Node-RED berperan dalam mengelola alur data, mulai dari penerimaan data sensor, pemrosesan, penyimpanan menggunakan InfluxDB sebagai database time-series, dan penyajian data secara visual dalam bentuk grafik dan indikator (gauge) baik dari kedua topik. Selain itu, sistem juga mampu menerima data masuk dari broker, misalnya untuk menyalakan atau mematikan LED sebagai bentuk interaksi dua arah (publish–subscribe). Pada praktikum ini dilakukan tanpa perangkat fisik sehingga dapatlebih memudahkan dalam melakukan percobaan sebelum turun ke perangkat keras asli. Hasil simulasi menunjukkan bahwa sistem berjalan dengan stabil, responsif, dan dapat menangani pengiriman maupun penerimaan data secara real-time.

Kata Kunci : **Internet of Things, Wokwi, NodeRed, DHT22**

I. PENDAHULUAN

Kemajuan teknologi Internet of Things (IoT) telah memungkinkan terciptanya sistem pemantauan lingkungan yang dapat bekerja secara otomatis dan real-time. Sistem ini mampu mendukung proses pengambilan keputusan dengan memberikan data kondisi lingkungan secara cepat, akurat, dan terus-menerus (Ray, 2016). Salah satu contoh implementasinya adalah pemantauan suhu dan kelembapan udara yang penting untuk berbagai aplikasi seperti rumah pintar, pertanian, dan industri. Dalam praktikum ini, dibangun sebuah sistem monitoring berbasis IoT yang menggunakan mikrokontroler ESP32 untuk membaca data dari sensor DHT22. Data dikirimkan melalui protokol komunikasi ringan MQTT, lalu disimpan di InfluxDB untuk keperluan historis, serta divisualisasikan menggunakan Node-RED dalam bentuk dashboard. Dengan arsitektur ini, data dapat dimonitor secara real-time dan memungkinkan pengguna untuk memantau kondisi lingkungan dari jarak jauh (Kaur & Kaur, 2021). Tujuan praktikum ini adalah memberikan pemahaman tentang proses perancangan sistem monitoring berbasis IoT, mulai dari pembacaan data sensor, pengiriman data melalui jaringan, penyimpanan, hingga visualisasi data dalam bentuk grafis untuk analisis dan pengambilan keputusan.

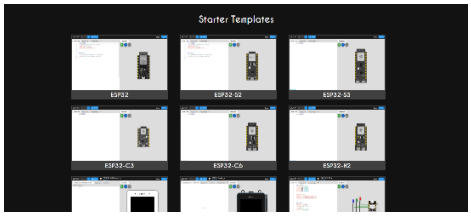
II. METODOLOGI

A. ALAT & BAHAN

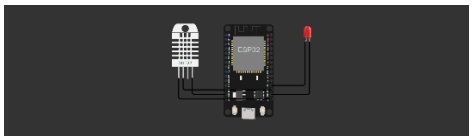
Visual Studio Code (C++*), PlatformIO, Sensor DHT22, LED, Library (ESPx PubSubClient, Nodejs, Influxdb, NodeRed.*

B. LANGKAH IMPLEMENTASI

1. Buka https://wokwi.com/ pilih ESP32, kemudian pilih *Starter Templates.*



2. Atur rangkain diagram pada wokwi seperti gambar berikut, sesuaikan dengan angka PIN.



3. Buat *file* baru bernama *diagram.json* pada *Visual Studio Code.* Kemudian pergi ke *wokwi* untuk meng *copy code diagram.json.*

```
{
  "version": 1,
  "author": "Cagita Dian A'yunin",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 19.2, "left": -4.76, "attrs": {} },
    {
     "type": "wokwi-led",
     "id": "led1",
     "top": -10.8,
     "left": 285,
     "rotate": 90,
     "attrs": { "color": "red" }
    },
    {
     "type": "wokwi-led",
     "id": "led2",
     "top": 94.8,
     "left": 265.8,
     "rotate": 90,
     "attrs": { "color": "yellow" }
    },
    {
     "type": "wokwi-led",
     "id": "led3",
     "top": 46.8,
     "left": 275.4,
     "rotate": 90,
     "attrs": { "color": "limegreen" }
    },
    {
     "type": "wokwi-resistor",
     "id": "r1",
     "top": 32.75,
     "left": 163.2,
     "attrs": { "value": "1000" }
    },
    {
     "type": "wokwi-resistor",
     "id": "r2",
     "top": 61.55,
     "left": 163.2,
     "attrs": { "value": "1000" }
    },
    {
     "type": "wokwi-resistor",
     "id": "r3",
     "top": 90.35,
     "left": 163.2,
     "attrs": { "value": "1000" }
    }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "esp:23", "r1:1", "green", [ "h0" ] ],
    [ "esp:22", "r2:1", "green", [ "h0" ] ],
    [ "esp:21", "r3:1", "green", [ "h0" ] ],
    [ "r1:2", "led1:A", "green", [ "v0" ] ],
    [ "r2:2", "led3:A", "green", [ "v0" ] ],
    [ "r3:2", "led2:A", "green", [ "v0" ] ],
    [ "led1:C", "led3:C", "green", [ "h0" ] ],
    [ "led3:C", "led2:C", "green", [ "h0" ] ],
    [ "led2:C", "esp:GND.3", "green", [ "h0" ] ]
  ],
  "dependencies": {}
}
```

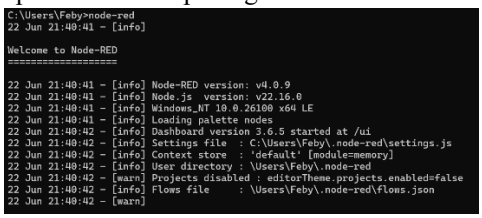
4. Buat file baru untuk code C++ pada main

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int LED_RED = 2;
```

```
const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "broker.emqx.io";

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  if ((char)payload[0] == '1') {
    digitalWrite(LED_RED, HIGH);
  } else {
    digitalWrite(LED_RED, LOW);
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");

      client.publish("IOT/Test1/mqtt", "Test IOT");

      client.subscribe("IOT/Test1/mqtt");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}

void setup() {
  pinMode(LED_RED, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  dht.setup(DHT_PIN, DHTesp::DHT22);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    TempAndHumidity data = dht.getTempAndHumidity();

    String temp = String(data.temperature, 2);
    client.publish("IOT/Test1/temp", temp.c_str());

    String hum = String(data.humidity, 1);
    client.publish("IOT/Test1/hum", hum.c_str());

    Serial.print("Temperature: ");
    Serial.println(temp);
```

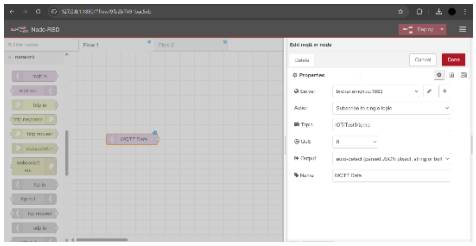
```
    Serial.print("Humidity: ");
    Serial.println(hum);
  }
}
```

5. Buka CMD lalu install npm install -g --unsafe-perm node-red, kemudian cek apakah sudah terpasang.

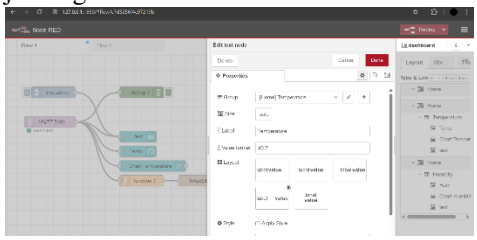


6. Buka server running yang telah didapat di CMD http://127.0.0.1:1880/ , lalu install juga "npm install node-red-dashboard"

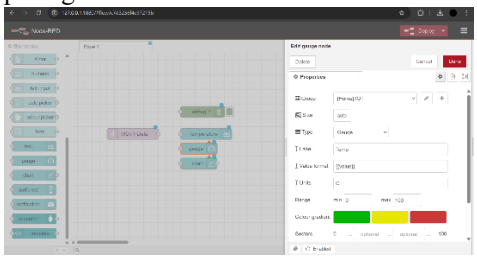
7. Buka NodeRed, pada filter nodes pilih Network >> mqtt in, klik dua kali lalu lengkapi setiap bagian yang kosong. Server dan Topic disamakan dengan code pada wokwi.



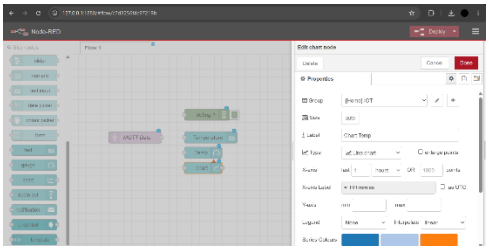
8. Pada nodes pilih commone > debug, pillih dashboard > text (untuk menambahkan group), bisa disesuiaikan dengan keinginan, jika ingin ada menu bar.



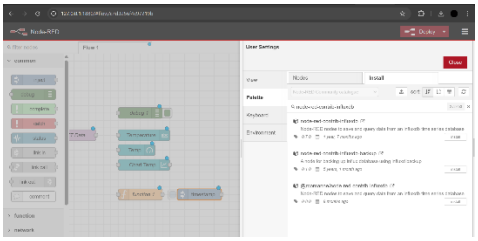
9. Tambahkan Gauge untuk konfigurasi temperature, pilih dashboard pada nodes, isi dan sesuaikan dengan mengikuti langkah pada gambar berikut.



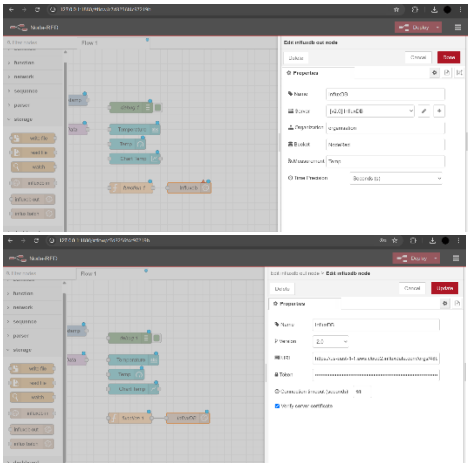
10. Tambahkan juga Chart, pilih dashboard pada nodes, isi dan sesukan seperti gambar berikut.



11. Selanjutnya tambahkan function, lalu pada common pilih inject, setelah itu pada menu kanan atas pada dashboard NodeRed, pilih "Manage palette" > Install > node-red-contrib-influxdb > Klik Install > kemudian akan muncul filter nodes "Storage" lalu pilih influxdb out



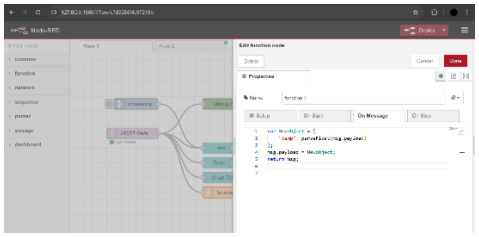
12. Buka situs resmi influxdb, lakukan Sign In dan Log In, kemudian ikuti proses pendaftaran. Jika sudah pilih Manage Database & Security > Access Token Manager > Go To Tokens > Generate API Token > All Access API Token > Copy Token



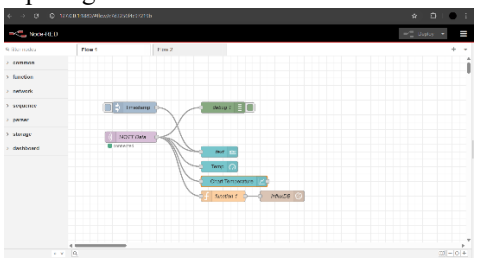
13. Kemudian klik function, isi code berikut

```
var Newobject = {
    "temp":
parseFloat(msg.payload)
};
msg.payload = Newobject;
```

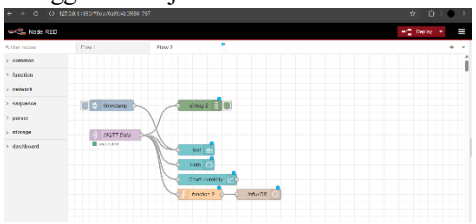
```
return msg;

isi juga untuk Humidity
var Newobject = {
    "hum":
parseFloat(msg.payload)
};
msg.payload = Newobject;
return msg;
```



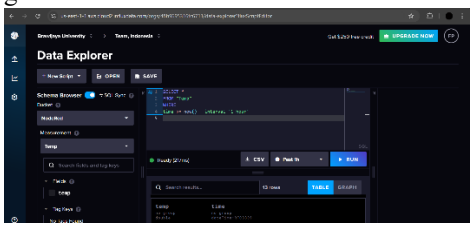
14. Hubungkan semua hinggga muncul notifikasi “connected pada MQTT Data” seperti gambar berikut.



15. Buat flow 2 lalu lakukan hal sama untuk Humidity dan penting untuk mengubah di MQTT data pada bagian Topic untuk mengganti menjadi IOT/Test1/hum

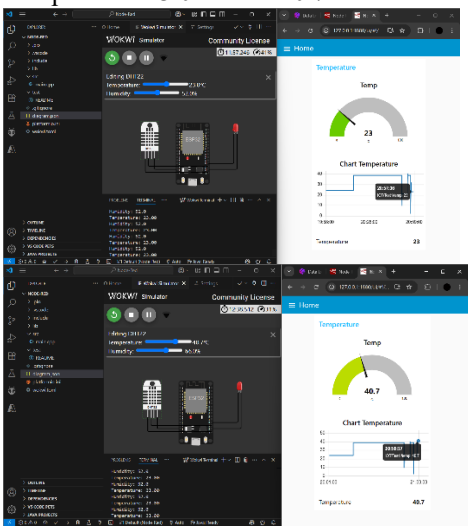


16. Deploy kedua flow yang sudah dibuat, masuk ke http://127.0.0.1:1880/ui dan runninf DHT MQTT pada Wokwi/Visual Studio Code. Kemudian masuk ke website influxdb, pilih Data Explorer ikuti seperti gambar berikut.

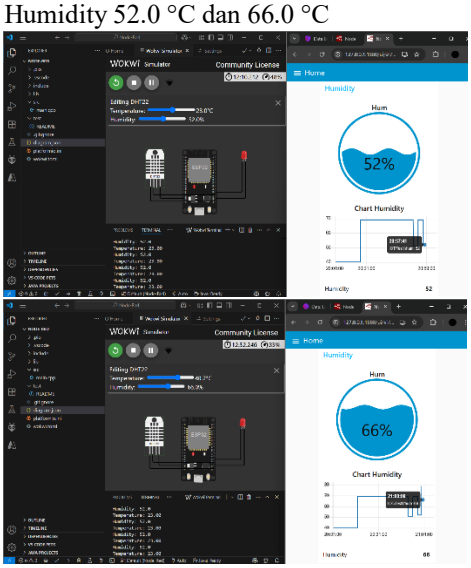


III. HASIL DAN PEMBAHASAN

1. Temperature 23.0 °C dan 40.7 °C



2. Humidity 52.0 °C dan 66.0 °C



IV. KESIMPULAN

Hasil praktikum monitoring suhu dan kelembaban menunjukkan bahwa dashboard dapat mengirimkan data secara realtime dari folder project baik dari wokwi maupun Visual Studio Code ke dashboard tampilan yang sudah diatur diagram alurnya dan konfigurasi setiap bagian. Dahboard terbagi menjadi dua halaman yakni halamn untuk IOT/Temperature dan halaman Humidity, keaduanya memiliki alur pengejerjaan yang sama tetapi yang membedakan adalah topic yang dipilih berbeda dan untuk elemen lainnya disesuaikan sesuai kebutuhan masing-masing.

V. DAFTAR PUSTAKA

Kaur, J., & Kaur, R. (2021). Real time monitoring of temperature and humidity using Internet of Things. Journal of Emerging Technologies and Innovative Research (JETIR), 8(6), 182-186. http://www.jetir.org/papers/JETIR2106029.pdf

Ray, P. P. (2016). A survey on Internet of Things architectures. Journal of King Saud University - Computer and Information Sciences, 30(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003