**Simulasi Pemantauan Suhu dan Kelembaban Berbasis Internet of Things (IoT) Menggunakan ESP32 dan Protokol MQTT**

**Abstrak**

Penelitian ini bertujuan untuk merancang serta mengimplementasikan sistem pemantauan suhu dan kelembaban berbasis Internet of Things (IoT) dengan memanfaatkan mikrokontroler ESP32, sensor DHT22, dan protokol komunikasi MQTT. Sistem yang dikembangkan mampu membaca data suhu dan kelembaban dari sensor DHT22 secara periodik, kemudian mengirimkan data tersebut ke broker MQTT publik (test.mosquitto.org) melalui koneksi WiFi. Selain itu, sistem dapat menerima instruksi dari broker untuk mengendalikan LED sebagai indikator, berdasarkan data yang dikirim ke topik tertentu. Komunikasi dua arah ini memungkinkan pemantauan dan pengendalian secara real-time dari jarak jauh. Implementasi dilakukan melalui simulasi pada platform Wokwi, sehingga pengujian dapat dilakukan tanpa perangkat keras fisik. Hasil pengujian menunjukkan sistem berhasil mengirimkan data sensor setiap dua detik serta merespons perintah pengendalian LED sesuai input yang diterima melalui topik MQTT.

**Pendahuluan**

Perkembangan teknologi Internet of Things (IoT) telah memberikan kemajuan signifikan di berbagai sektor, salah satunya pada sistem monitoring dan kendali jarak jauh. IoT memungkinkan perangkat fisik saling terhubung serta bertukar data melalui jaringan internet. Salah satu implementasinya adalah sistem pemantauan suhu dan kelembaban berbasis mikrokontroler. Dengan sensor lingkungan, perangkat IoT dapat membaca dan mengirimkan informasi secara real-time ke server atau pengguna. Pada praktikum ini, ESP32 digunakan sebagai pusat pengendali, DHT22 sebagai sensor suhu dan kelembaban, serta MQTT sebagai protokol komunikasi data yang efisien.

**Latar Belakang**

Pemantauan suhu dan kelembaban sangat penting di berbagai bidang, seperti pertanian dan industri. Pemantauan manual dinilai kurang efisien dan tidak dapat memberikan data secara real-time, terutama untuk pemantauan jarak jauh. Melalui teknologi IoT, proses pemantauan dapat diotomatisasi dengan memanfaatkan sensor dan mikrokontroler yang terhubung melalui internet. Praktikum ini menggunakan ESP32 sebagai pusat kendali, DHT22 untuk membaca data suhu dan kelembaban, serta MQTT sebagai protokol komunikasi yang ringan dan mendukung sistem publish-subscribe. Seluruh sistem dirancang agar mampu memonitor lingkungan sekaligus mengendalikan perangkat seperti LED secara jarak jauh melalui internet. Praktikum ini dilakukan secara virtual tanpa perangkat fisik, sehingga mahasiswa dapat memahami konsep dan alur kerja sistem IoT secara menyeluruh, mulai dari pemrograman mikrokontroler, pengambilan data sensor, hingga komunikasi dengan broker MQTT

**Tujuan Eksperimen**

Eksperimen ini bertujuan untuk:

* Mengirimkan data sensor secara periodik ke broker MQTT menggunakan koneksi WiFi.
* Memahami alur kerja sistem IoT, mulai dari pengambilan data sensor, pengiriman data ke broker, hingga penerimaan perintah untuk aktuator melalui jaringan internet1.

**Metodologi**

**Alat dan Bahan**

* Platform simulasi Wokwi untuk simulasi sistem digital.
* Visual Studio Code dengan ekstensi PlatformIO.
* Kode program berbasis C/C++ untuk mengontrol LED virtual.
* Mikrokontroler ESP32 (virtual).
* Sensor DHT22 untuk membaca suhu dan kelembaban.
* LED (merah) sebagai indikator output.
* MQTT Broker (test.mosquitto.org).
* Library: PubSubClient.h, WiFi.h, DHTesp.h1.

**Langkah Implementasi**

* Membuat project baru di Visual Studio Code dengan PlatformIO.
* Membuat project di simulator Wokwi.
* Menambahkan perangkat, menghubungkan, dan menambahkan library yang dibutuhkan.
* Menambahkan kode program dan melakukan pengujian.
* Membuka MQTT Explorer, mengatur koneksi dan topik, lalu mengamati hasil simulasi.
* Menjalankan program di Wokwi dan mengamati komunikasi data antara VSC dan MQTT Explorer.
* LED menyala jika data 1 diterima, dan mati jika data 0 diterima1.

**Hasil dan Pembahasan**

Hasil pengujian menunjukkan bahwa sistem berhasil membaca dan mengirimkan data suhu serta kelembaban secara periodik setiap dua detik. Sistem juga mampu merespons perintah pengendalian LED sesuai input yang diterima melalui topik MQTT. Hal ini membuktikan bahwa protokol MQTT efektif dalam komunikasi antar perangkat IoT, bahkan dalam simulasi tanpa perangkat fisik1.

**Lampiran**

**Kode Program (Main.cpp)**

#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 = "test.mosquitto.org";

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);

}

}

WiFi.begin(ssid, password); //koneksi ke jaringan wifi

while (WiFi.status() != WL\_CONNECTED) { //perintah tunggu esp32 sampi terkoneksi ke wifi 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) { //perintah untuk menampilkan data ketika esp32 di setting sebagai subscriber

Serial.print("Message arrived ["); Serial.print(topic);

Serial.print("] ");

for (int i = 0; i < length; i++) { //mengecek jumlah data yang ada di topik mqtt Serial.print((char)payload[i]);

}

Serial.println();

// Switch on the LED if an 1 was received as first character if ((char)payload[0] == '1') {

digitalWrite(LED\_RED, HIGH); // Turn the LED on

} else {

digitalWrite(LED\_RED, LOW); // Turn the LED off

}

}

void reconnect() { //perintah koneksi esp32 ke mqtt broker baik itu sebagai publusher atau subscriber

// Loop until we're reconnected while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// perintah membuat client id agar mqtt broker mengenali board yang kita gunakan String clientId = "ESP32Client-";

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

// Attempt to connect

if (client.connect(clientId.c\_str())) { Serial.println("Connected");

// Once connected, publish an announcement...

client.publish("IOT/Test1/mqtt", "Test IOT"); //perintah publish data ke alamat topik yang di setting

// ... and resubscribe

client.subscribe("IOT/Test1/mqtt"); //perintah subscribe data ke mqtt broker

} else { Serial.print("failed, rc="); Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying delay(5000);

}

}

}

void setup() {

pinMode(LED\_RED, OUTPUT); // inisialisasi pin 2 / ledbuiltin sebagai output Serial.begin(115200);

setup\_wifi(); //memanggil void setup\_wifi untuk dieksekusi client.setServer(mqtt\_server, 1883); //perintah connecting / koneksi awal ke broker

client.setCallback(callback); //perintah menghubungkan ke mqtt broker untuk subscribe data dht.setup(DHT\_PIN, DHTesp::DHT22);//inisialiasi komunikasi dengan sensor dht22

}

void loop() {

if (!client.connected()) { reconnect();

}

client.loop();

unsigned long now = millis();

if (now - lastMsg > 2000) { //perintah publish data lastMsg = now;

TempAndHumidity data = dht.getTempAndHumidity();

String temp = String(data.temperature, 2); //membuat variabel temp untuk di publish ke broker mqtt

client.publish("IOT/Test1/temp", temp.c\_str()); //publish data dari varibel temp ke broker mqtt

String hum = String(data.humidity, 1); //membuat variabel hum untuk di publish ke broker mqtt client.publish("IOT/Test1/hum", hum.c\_str()); //publish data dari varibel hum ke broker mqtt

Serial.print("Temperature: "); Serial.println(temp); Serial.print("Humidity: "); Serial.println(hum);

}

}

# Kode Program (diagram.json)

{

"version": 1,

"author": "Cagita Dian A’yunin", "editor": "wokwi",

"parts": [

{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": {} },

{ "type": "wokwi-dht22", "id": "dht1", "top": -9.3, "left": -111, "attrs": {} },

{ "type": "wokwi-led", "id": "led1", "top": 102, "left": 186.2, "attrs": { "color": "red" } }

],

"connections": [

[ "esp:TX0", "$serialMonitor:RX", "", [] ],

[ "esp:RX0", "$serialMonitor:TX", "", [] ],

[ "dht1:GND", "esp:GND.2", "black", [ "v0" ] ],

[ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],

[ "dht1:SDA", "esp:D15", "green", [ "v0" ] ],

[ "led1:C", "esp:GND.1", "green", [ "v0" ] ],

[ "esp:D2", "led1:A", "green", [ "h61.9", "v-53.6", "h86.4", "v57.6" ] ]

],

"dependencies": {}

}