

MANUAL BOOK
“Monitoring dan Controlling Sistem Pendeteksi Api Menggunakan Platform MQTT
Panel”

Projek Akhir Praktikum Internet of Things



Disusun Oleh : Kelompok 4 / IOT A

NAMA	NIM
EGA SULFIKA	2009106011
MITHA AMALIA	2009106028
M. RIZKY NILZAMYAHYA	2009106029
ALAN NUZULAN	2009106032

Asisten :

Kandika Prima Putra	Delfan Rynaldo Laden	M. Rizky Amanullah	Muhammad Al Fahri
1915016015	1915016069	1915016073	1915026013

INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MULAWARMAN
2023

DAFTAR ISI

A.	LATAR BELAKANG SISTEM	3
B.	FUNGSI SISTEM	3
C.	KONSEP YANG DIGUNAKAN	3
D.	BOARD SCHEMATIC	4
E.	TAHAPAN PERANCANGAN SISTEM	4

A. LATAR BELAKANG SISTEM

Pendeteksian dini kebakaran sangat penting untuk mengidentifikasi ancaman secepat mungkin sehingga tindakan pencegahan dan tindakan darurat dapat diambil dengan cepat. Oleh karena itu, sistem pendeteksi api yang handal dan efektif sangat diperlukan untuk mengurangi resiko kerugian akibat kebakaran. Perangkat monitoring sering dipasang pada berbagai alat atau dipasang pada suatu ruangan untuk mengambil informasi yang dibutuhkan. Sistem ini dapat dimonitor melalui platform Internet of Things (IoT) MQTT Panel. Pada sistem ini pendeteksi api akan menyalakan buzzer serta LED untuk memberikan peringatan, dengan cara kerja sensor akan mendeteksi api sesuai dengan suhu yang bisa diinputkan melalui platform Internet of Things (IoT) MQTT Panel dan jika terdeteksi ada api diatas batas suhu yang ditentukan maka LED akan menyala dengan kelap-kelip dan Buzzer akan mengeluarkan suara sebagai tanda. Apabila terdeteksi api tapi tidak melewati batas suhu maka Buzzer menyala dan led menyala (tetapi tidak kelap-kelip).

B. FUNGSI SISTEM

1. Monitor suhu serta kelembapan yang terdeteksi melalui platform IoT MQTT Panel
2. Monitor api serta LED dan buzzer melalui platform IoT MQTT Panel
3. Controlling batas suhu melalui platform IoT MQTT Panel
4. Controlling LED dan buzzer melalui platform IoT MQTT Panel

C. KONSEP YANG DIGUNAKAN

1. MQTT

MQTT digunakan untuk komunikasi antar node. Kedua node terkoneksi pada server **broker.hivemq.com** port **1883** dengan topic **iot_unmul/iot_a_4**. Edge node mengirim data pada topic, sedangkan master node akan menerima data dari topic yang di-subscribe untuk mengolah datanya.

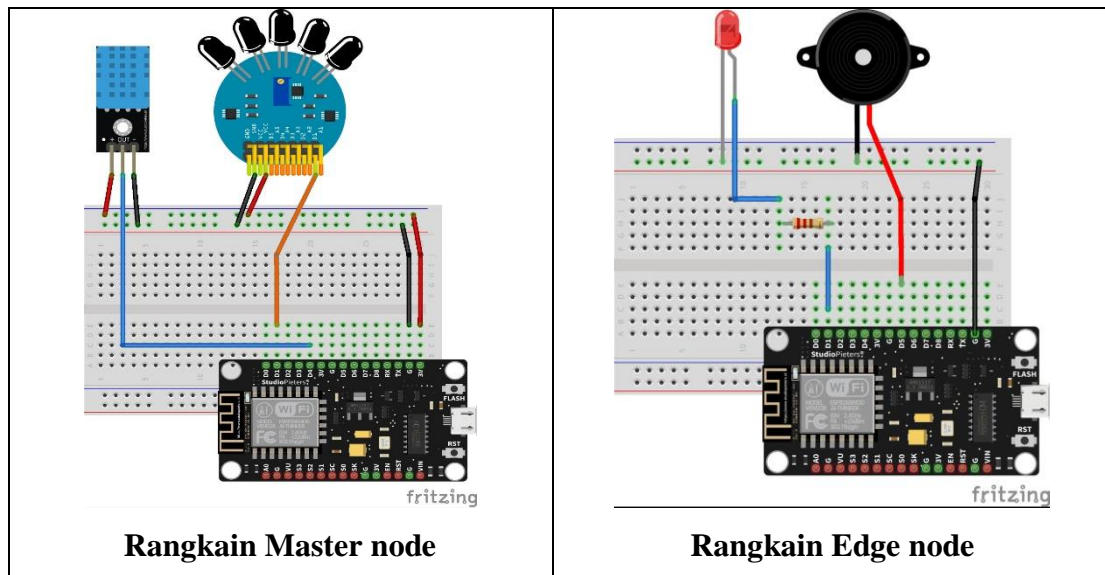
2. Platform IOT

Platform IoT MQTT Panel digunakan karena kemudahannya dalam mengaksesnya melalui mobile secara gratis, tetapi memiliki kekurangan yang tidak bisa diakses melalui website dan dikarenakan broker gratis sehingga saat pengiriman data terkena delay.

3. Sensor

Sensor yang digunakan pada sistem ini yaitu 5 Channel Sensor Api untuk mendeteksi adanya titik api dengan memanfaatkan 5 buah Infrared (IR) receiver yang terdapat pada modul sensor tersebut. Sensor ini memiliki 5 indikator LED (Light Emitting Diode) yang berguna sebagai indikator pendeteksian.

D. BOARD SCHEMATIC



Gambar 1 Board Schematic

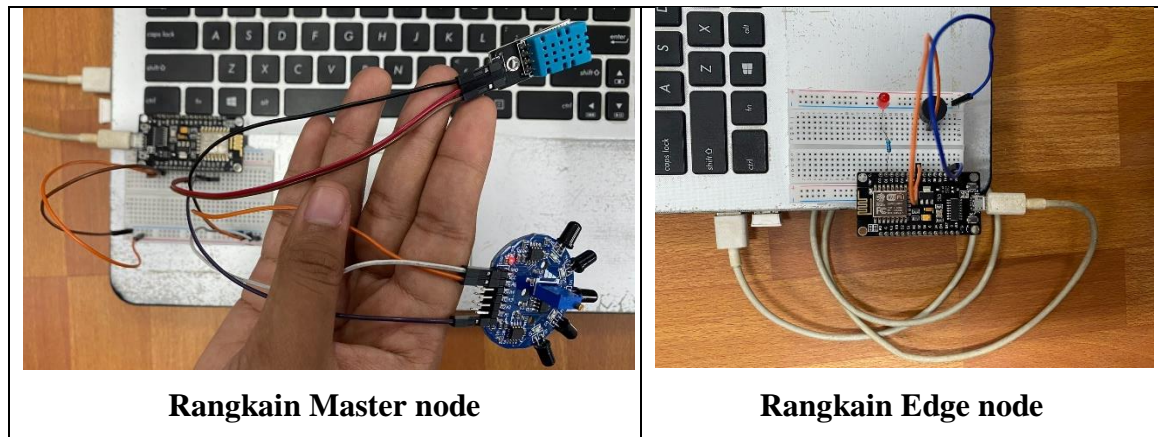
Komponen yang digunakan antara lain:

1. 5 Channel Sensor Api x 1
2. Buzzer x 1
3. LED x 1
4. Node MCU x 2
5. Resistor x 1
6. BreadBoard x 2
7. Kabel Jumper Female-Male x 6
8. Kabel Jumper Male-Male x 4

E. TAHAPAN PERANCANGAN SISTEM

Berikut adalah cara merancang sistem monitoring pendeteksi api berbasis IoT. Perancangan sistem terdiri dari tahap merangkai komponen elektronik, persiapan platform IoT, perancangan program Arduino, dan pengujian sistem.

1. Merangkai Komponen Elektronik

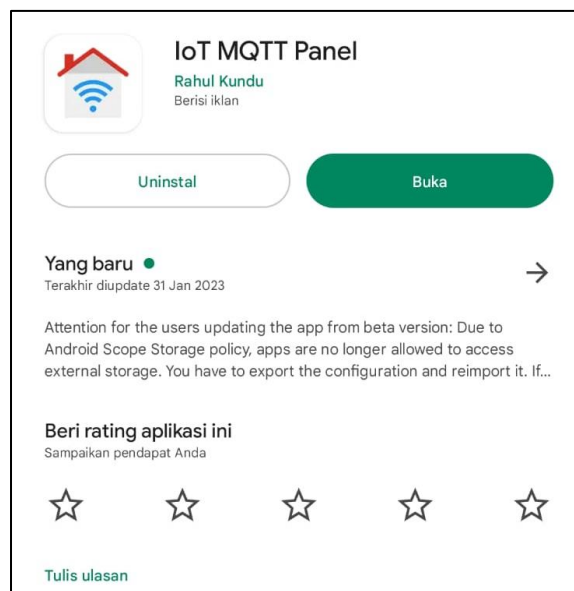


Gambar 2 Rangkaian Akhir Master node dan Edge node

Rangkaian komponen elektronik seperti pada Gambar 2. Setiap node akan disuplay daya 5V dari kabel USB.

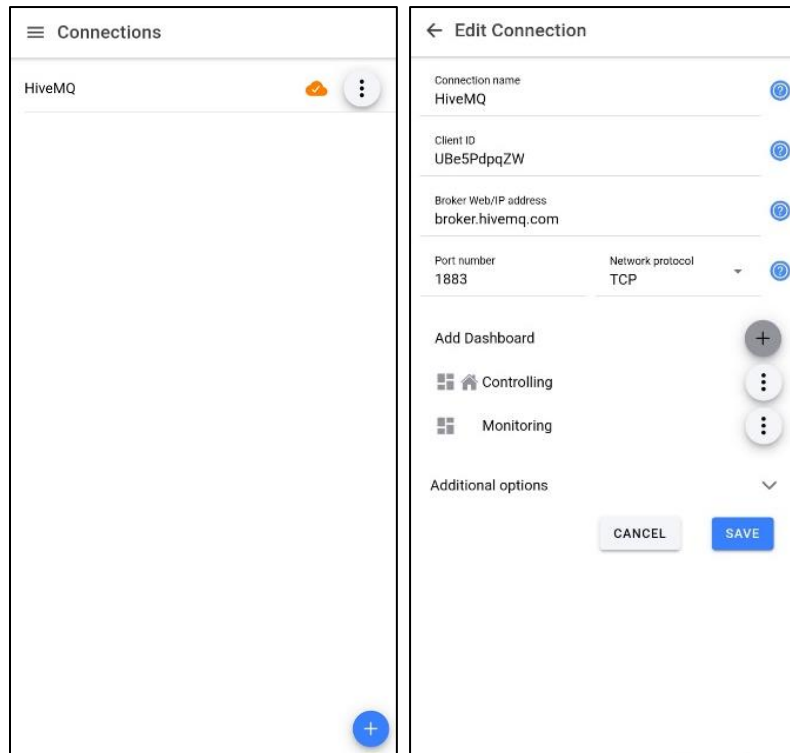
2. Persiapan Platform IoT

Download aplikasi IoT MQTT panel seperti pada Gambar 3.



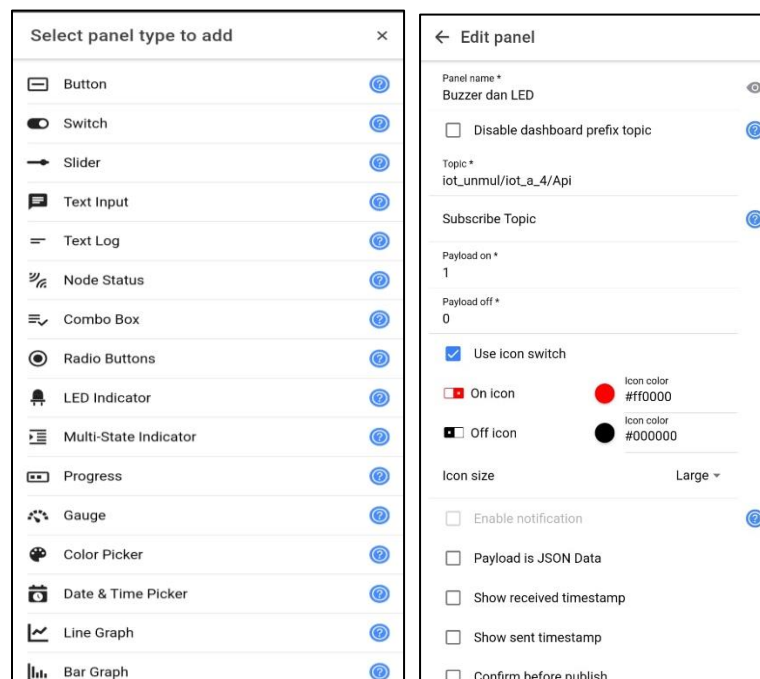
Gambar 3 Download aplikasi IoT MQTT panel

Jika ingin membuat dashboard baru klik floating button dan masukkan nama koneksi, client ID, broker web atau IP Address, Port number, Network protocol. Kemudian tekan create seperti pada Gambar 4.



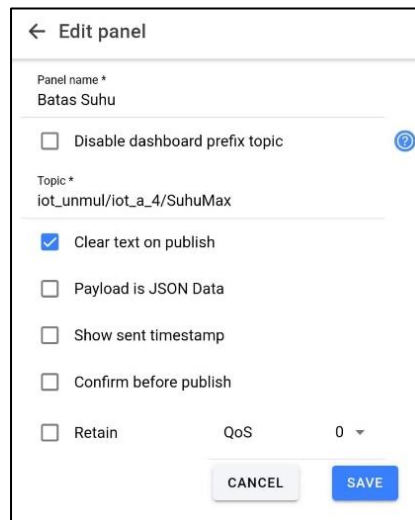
Gambar 4 Membuat Dashboard Awal

Untuk membuat sebuah controlling LED adalah langkah pertama klik floating button kemudian pilih switch dan isi data sesuai pada Gambar 5.



Gambar 5 Membuat Controlling LED

Untuk membuat sebuah inputan seperti pada inputan batas suhu yaitu klik floating button kemudian pilih Text Input dan masukan datanya seperti pada Gambar 6.



← Edit panel

Panel name *
Batas Suhu

☐ Disable dashboard prefix topic

Topic *
iot_unmul/iot_a_4/SuhuMax

☒ Clear text on publish

☐ Payload is JSON Data

☐ Show sent timestamp

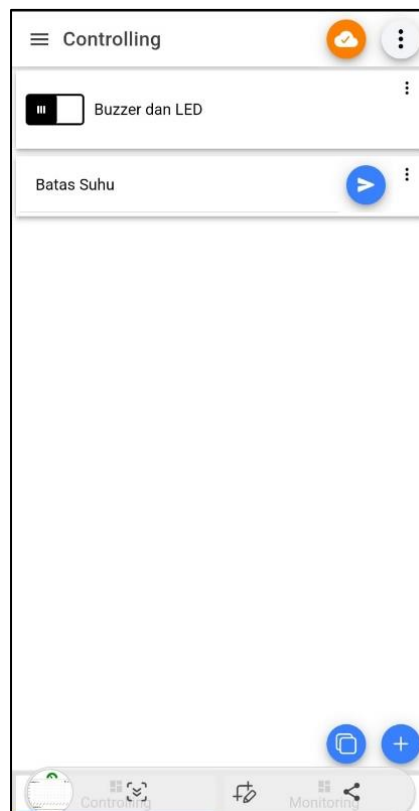
☐ Confirm before publish

☐ Retain QoS 0 ▾

CANCEL SAVE

Gambar 6 Membuat Inputan Batas Suhu

Pada Gambar 7 merupakan dashboard Controlling dimana pada dashboard ini bisa mengontrol LED dan buzzer serta menginput batas suhu yang diinginkan.



Gambar 7 Dashboard Controlling

Untuk membuat monitoring api, LED, dan buzzer menggunakan LED Indicator kemudian masukan informasi seperti pada Gambar 8.

The screenshot shows the 'Edit panel' configuration interface. The panel name is 'Api'. The topic is 'iot_unmul/iot_a_4/Api'. The payload on is '1' and the payload off is '0'. The 'On icon' is a red light bulb with a color of '#df0000'. The 'Off icon' is a green light bulb with a color of '#36ea5b'. The icon size is set to 'Large'. There are checkboxes for 'Disable dashboard prefix topic', 'Enable notification', 'Payload is JSON Data', and 'Show received timestamp'. The QoS is set to '0'. At the bottom, there are 'CANCEL' and 'SAVE' buttons.

Gambar 8 Membuat Monitoring Api

Untuk membuat monitoring suhu yaitu menggunakan Gauge dan masukan informasi seperti pada Gambar 9.

The screenshot shows the 'Edit panel' configuration interface for a temperature monitoring panel. The panel name is 'SUHU'. The topic is 'iot_unmul/iot_a_4/Suhu'. The payload min is '0' and the payload max is '100'. The unit is 'Unit' and the factor is '1'. The arc color is set to '33,33' (green) and '66,67' (yellow). There are checkboxes for 'Disable dashboard prefix topic', 'Enable notification', 'Payload is JSON Data', and 'Show received timestamp'. The QoS is set to '0'. At the bottom, there are 'CANCEL' and 'SAVE' buttons.

Gambar 9 Membuat Monitoring Suhu

Untuk membuat monitoring Kelembapan yaitu menggunakan Line Graph kemudian masukan informasi seperti pada Gambar 10.

The screenshot shows the 'Edit panel' configuration interface. At the top, there is a back arrow and the title 'Edit panel'. Below this, the 'Panel name' is set to 'KELEMBAPAN'. There is a checkbox for 'Disable dashboard prefix topic' which is unchecked. The 'Topic for graph 1' is 'iot_unmul/iot_a_4/Kelembapan', and the 'Label for graph 1' is also 'KELEMBAPAN'. Under 'Factor', the value is '1', and under 'Graph color', a red circle is shown next to the hex code '#ea1111'. There are several checkboxes: 'Show plot area' is checked, 'Show points and tooltip' is unchecked, 'Enable notification' is unchecked, and 'Payload is JSON Data' is unchecked. Below these is a section 'Add more graph' with a plus icon and a 'Smooth curve' checkbox which is unchecked. At the bottom, 'Max persistence' is set to '10' and 'QoS' is set to '0'. There are 'CANCEL' and 'SAVE' buttons at the very bottom.

Gambar 10 Membuat Monitoring Kelembapan

Pada Gambar 11 merupakan dashboard Monitoring dimana pada dashboard ini bisa memonitor Api, LED, Buzzer, Suhu dan Kelembapan



Gambar 11 Dashboar Monitoring

3. Perancangan Program pada Arduino IDE

Source code dapat diakses pada link dibawah.

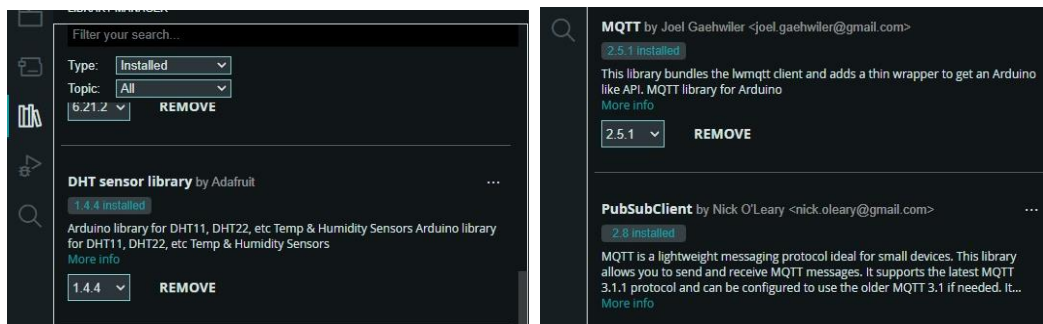
Master Node:

<https://github.com/land21/pa-praktikum-iot-unmul-a4/blob/main/PublisherPA.ino>

Edge Node:

<https://github.com/land21/pa-praktikum-iot-unmul-a4/blob/main/SubscriberPA.ino>

Library



Gambar 12 Library MQTT

Agar dapat menggunakan protokol MQTT untuk mengirim pesan, pastikan sudah menginstall library **DHT Sensor Library** dari **Adafruit**, **MQTT** dari **Joel Gaehwiler**, dan **PubSubClient** dari **Nick O'Leary**.

Kodingan Master node

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
```

Penjelasan :

Source Code di atas digunakan untuk menambahkan library yang akan di gunakan dalam master node. Library yang pertama yakni ESP8266WiFi.h yang digunakan untuk mengkonfigurasi sambungan wifi dari nodeMCU. Lalu ada PubSubClient.h yang digunakan untuk mengkonfigurasi client mqtt. Terakhir ada DHT.h yakni library yang berguna untuk mengkonfigurasi DHT.

```

// Mengatur SSID dan Password Jaringan yang di pakai
const char* ssid = "MITHA";
const char* password = "mithew13";

// server Mqtt broker
const char* mqtt_server = "broker.hivemq.com";

#define FS_PIN D1 // Sensor Api
#define DHTPIN D4 // Sensor Suhu
#define DHTTYPE DHT11 //Mengatur TYPE DHT (Karena ada 2 jenis
yaitu DHT11 & DHT22)
#define TIMEDHT 1000

unsigned long timerDHT = 0;

DHT dht(DHTPIN, DHTTYPE);

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int FS = 0;
int FSold = 0;
int value = 0;
float h;
float t;

```

Penjelasan :

Source code di atas berguna untuk mengatur SSID dan Password wifi yang akan digunakan, lalu link broker yang digunakan untuk MQTT. Selanjutnya adalah untuk mendefinisikan PIN sensor dan tipe dari DHT. Lalu ada variabel variabel yang di definisikan di awal.

```
// Fungsi koneksi wifi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.println("Mencoba Menghubungkan ke ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

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

    randomSeed(micros());

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

Penjelasan :

Fungsi setup wifi digunakan untuk melakukan koneksi dengan wifi yang digunakan.

```
// fungsi untuk menghubungkan ke broker
void reconnect() {
    // Loop sampai terkoneksi dengan broker
    while (!client.connected()) {
        Serial.println("Mencoba untuk menghubungkan dengan MQTT...");
        // membuat random client ID
        String clientId = "ESP8266Client- ";
        clientId += String(random(0xffff), HEX);
        // mencoba untuk connect
        if (client.connect(clientId.c_str())) {
            Serial.println("Terhubung");
            client.subscribe("iotunmulunik");
        } else {
            Serial.println("Gagal Terhubung");
        }
    }
}
```

```

        Serial.println("Mencoba dalam 5 detik");
        delay(5000);
    }
}
}

```

Penjelasan :

Fungsi reconnect digunakan untuk looping hingga terkoneksi dengan broker. Dibagian ini juga dibuatkan ID client dengan random.

```

void setup() {
    pinMode(FS_PIN,INPUT); // inisialisasi Sensor Api sebagai input
    Serial.begin(115200);
    dht.begin(); // memulai dht
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

```

Penjelasan :

Fungsi setup adalah untuk melakukan setup pada pinMode yang digunakan seperti FS untuk input, lalu lakukan pemulaian pada dht, dan melakukan set server client.

```

void loop() {
    if (!client.connected()) { // jika tidak terkoneksi dengan MQTT
        maka akan melakukan fungsi reconnect
        reconnect();
    }
    client.loop();

    if ((millis() - timerDHT) > TIMEDHT) { // jika detik di kurangi
        timer lebih dari TIMEDHT
        // Update pada timer menggunakan millis
        timerDHT = millis();
        // Membaca kelembapan
        h = dht.readHumidity();
        // Membaca Suhu
        t = dht.readTemperature();
    }
}

```

```

float f = dht.readTemperature(false);

// mengecek dht jika gagal dalam membaca suhu dan kelembapan
if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("DHT sensor gagal membaca!"));
    return;
}

}

// membuat nilai inputan DHT menjadi integer pada kelembapan dan
Suhu
int hum = (int)h;
int temp = (int)t;

Serial.print("Suhu : ");
Serial.println(temp);
Serial.print("Kelembaban : ");
Serial.println(hum);

delay(200);
snprintf (msg, MSG_BUFFER_SIZE, "%s", itoa(temp,msg,10));
client.publish("iot_unmul/iot_a_4/Suhu", msg); // publish data
Suhu
snprintf (msg, MSG_BUFFER_SIZE, "%s", itoa(hum,msg,10));
client.publish("iot_unmul/iot_a_4/Kelembapan", msg); // publish
data Kelembapan

int FS = digitalRead(FS_PIN); // membaca sensor Api
// jika nilai sensor api masih sama maka tidak akan ada publish
if (FS == FSold){
    FS = FSold;
} else {
    FSold = FS;
    delay(2000);
    snprintf (msg, MSG_BUFFER_SIZE, "%s", itoa(FS,msg,10));
    Serial.print("Api : ");

```

```

    Serial.println(FS);
    client.publish("iot_unmul/iot_a_4/Api", msg); } // publish data
Api
}

```

Penjelasan :

Pada fungsi loop melakukan beberapa aksi yakni, jika tidak terhubung dengan broker maka akan melakukan reconnet terus menerus. Lalu ada pengambilan nilai sensor DHT yang terdiri dari Humidity atau Kelembapan dan Temperatur atau Suhu, dimana Suhu dan temperatur di ambil tidak setiap saat tapi menggunakan rentang 1 detik. Lalu suhu dan temperatur akan di publish dengan topic “iot_unmul/iot_a_4/Suhu” dan “iot_unmul/iot_a_4/Kelembapan”. Selanjutnya pengambilan nilai sensor Api dengan Digital input jika ada api maka akan bernilai 1 dan jika tidak api akan bernilai 0, sensor akan melakukan publish dengan topic “iot_unmul/iot_a_4/Api” tapi jika nilai publish sama maka tidak akan melakukan publish.

Kodingan Edge node

```

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

```

Penjelasan :

Source Code di atas digunakan untuk menambahkan library yang akan di gunakan dalam edge node. Library yang pertama yakni ESP8266WiFi.h yang digunakan untuk mengkonfigurasi sambungan wifi dari nodeMCU. Dan ada PubSubClient.h yang digunakan untuk mengkonfigurasi client mqtt.

```

// Mengatur SSID dan Password Jaringan yang di pakai
const char* ssid = "Universitas Mulawarman";
const char* password = "";

// server Mqtt broker
const char* mqtt_server = "broker.hivemq.com";

// Buzzer & LED pin
#define BUZZER_PIN D5
#define LED_PIN D1

```

```

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
int Hum;
int Temp;
int Api;
int Max = 30;

```

Penjelasan :

Source code di atas berguna untuk mengatur SSID dan Password wifi yang akan digunakan, lalu link broker yang digunakan untuk MQTT. Selanjutnya adalah untuk mendefinisikan PIN sensor. Lalu ada variabel variabel yang di definisikan di awal.

```

// Fungsi koneksi wifi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Mencoba Menghubungkan ke ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

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

    randomSeed(micros());
    Serial.println("WiFi Terhubung");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

```

Penjelasan :

Fungsi setup wifi digunakan untuk melakukan koneksi dengan wifi yang digunakan.


```

// Fungsi untuk menerima data
void callback(char* topic, byte* payload, unsigned int length) {

    if(strcmp(topic, "iot_unmul/iot_a_4/Kelembapan") == 0) // jika
    topic yang di terima adalah kelembapan
    {
        String hum = "";
        for (int i = 0; i < length; i++) {
            hum += (char)payload[i];
        }
        Hum = hum.toInt(); // nilai kelembapan akan di tempatkan
pada variabel Hum
        Serial.print("Kelembapan [");
        Serial.print(Hum);
        Serial.println("] ");
    }

    else if(strcmp(topic, "iot_unmul/iot_a_4/Suhu") == 0) // jika
    topic yang di terima adalah Suhu
    {

        String temp = "";
        for (int i = 0; i < length; i++) {
            temp += (char)payload[i];
        }
        Temp = temp.toInt(); // nilai Suhu akan di tempatkan pada
variabel Temp
        Serial.print("Suhu [");
        Serial.print(Temp);
        Serial.println("] ");
    }

    else if(strcmp(topic, "iot_unmul/iot_a_4/Api") == 0) // jika
    topic yang di terima adalah Api
    {
        String fire = "";
        for (int i = 0; i < length; i++) {

```

```

        fire += (char)payload[i];
    }
    Api = fire.toInt(); // nilai keadaan Api akan di tempatkan
pada variabel Api
    Serial.print("Api [");
    Serial.print(Api);
    Serial.println("] ");
}

else if(strcmp(topic, "iot_unmul/iot_a_4/SuhuMax") == 0) // jika
topic yang di terima adalah menentukan Maksimal Suhu
{
    String maks = "";
    for (int i = 0; i < length; i++) {
        maks += (char)payload[i];
    }
    Max = maks.toInt(); // nilai maksimal Suhu akan di tempatkan
pada variabel Max
}
}

```

Penjelasan :

Fungsi callback digunakan untuk menerima data dari broker mqtt. Data yang diterima disesuaikan dengan topic, jika topic suhu maka edge node akan menerima suhu dan menyimpannya begitupun dengan topik Api, Kelembapan dan Maksimal Suhu.

```

// fungsi untuk mengubungkan ke broker
void reconnect() {
    // Loop sampai terkoneksi dengan broker
    while (!client.connected()) {
        Serial.println("Mencoba untuk menghubungkan dengan MQTT...");
        // membuat random client ID
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);
        // mencoba untuk connect
        if (client.connect(clientId.c_str())) {
            Serial.println("Terhubung");
        }
    }
}

```

```

        client.subscribe("iot_unmul/iot_a_4/#");
    } else {
        Serial.println("Gagal Terhubung ");
        Serial.println("Mencoba dalam 5 detik");
        delay(5000);
    }
}
}
}

```

Penjelasan :

Fungsi reconnect digunakan untuk looping hingga terkoneksi dengan broker. Dibagian ini juga dibuatkan ID client dengan random.

```

void setup() {
    pinMode(BUZZER_PIN, OUTPUT); // Inisialisasi pin BUZZER
    pinMode(LED_PIN, OUTPUT);    // Inisialisasi pin LED
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

```

Penjelasan :

Fungsi setup adalah untuk melakukan setup pada pinMode yang digunakan seperti Buzzer dan LED untuk output, lalu lakukan set pada callback, dan melakukan set server client.

```

void loop() {
    if (!client.connected()) { // jika tidak terkoneksi dengan MQTT
        maka akan melakukan fungsi reconnect
        reconnect();
    }
    client.loop();

    if (Temp > Max) // jika temperatur lebih dari nilai maksimal
    {
        // melakukan blink pada LED dengan delay 500
        digitalWrite(LED_PIN, HIGH);
    }
}

```

```

        delay(500);
        digitalWrite(LED_PIN, LOW);
        delay(500);
    }
    else if (Temp <= Max) // jika temperatur kurang dari sama dengan
maksimal
    {
        digitalWrite(LED_PIN, LOW); // LED akan mati
    }

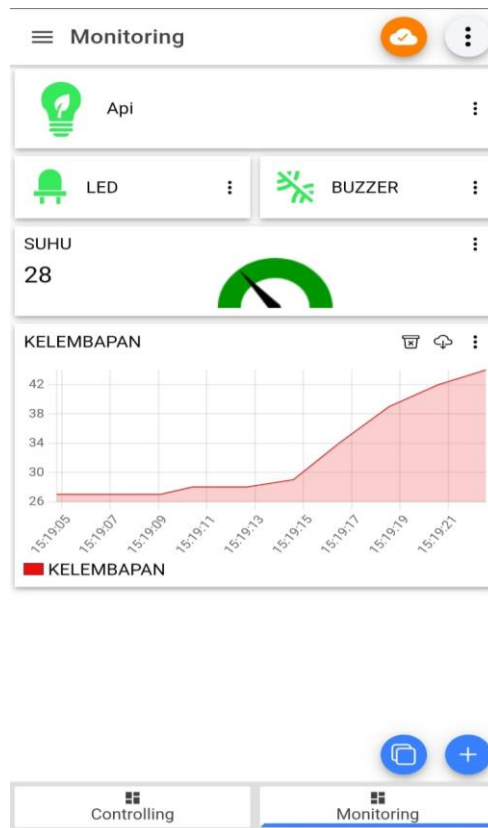
    if (Api == 1) // jika api terdeteksi
    {
        digitalWrite(LED_PIN, HIGH); // LED akan menyala
        tone(BUZZER_PIN,1000); // Buzzer akan berbunyi dengan frekuensi
1000
    }
    else if (Api == 0) // jika api tidak terdeteksi
    {
        digitalWrite(LED_PIN, LOW); // LED Mati
        noTone(BUZZER_PIN); // Buzzer mati
    }
}

```

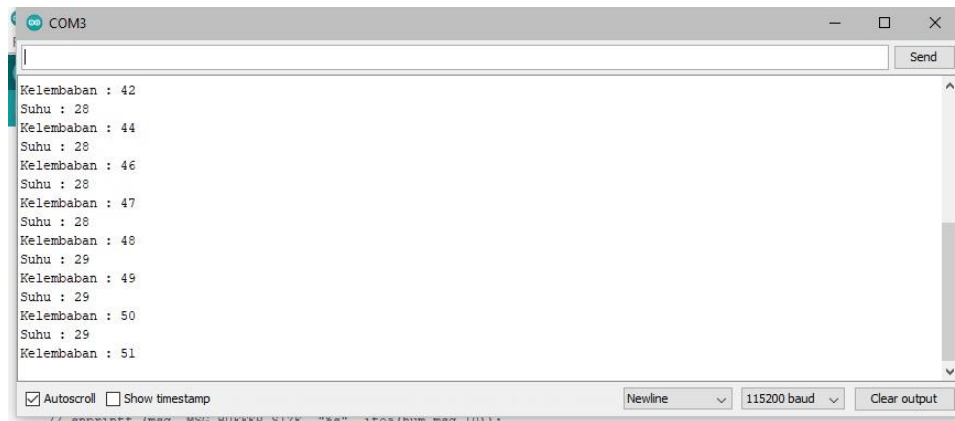
Penjelasan :

Pada fungsi loop melakukan beberapa aksi yakni, jika tidak terhubung dengan broker maka akan melakukan reconnet terus menerus. Lalu jika suhu melebihi maksimal ketentuan suhu maka LED akan melakukan Blink. Terakhir jika Api terdeteksi maka Buzzer akan berbunyi dengan frekuensi 1000 di dampingi oleh LED yang menyala.

4. Pengujian Sistem



Gambar 13 Hasil Monitoring pada Platform IoT



Gambar 14 Hasil Monitoring pada Serial Monitor

Setelah program di upload, hasil monitoring dapat dilihat pada platform seperti Gambar 6. Dimana terlihat pada platform.