TIGA SERANGKAI

since 1959

Dokumentasi Project

Praktik Industri

PT. Tiga Serangkai Pustaka Mandiri

Dibuat oleh:

1. **Anggoro Bimo Prihadi (K3521007)**
2. **Danang Kurniawan (K3521016)**
3. **Febrian Sabila Firdaus (K3521026)**

Instalasi Thingsboard Server - Community Edition

Pada Ubuntu Server

```
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

Versi Ubuntu yang digunakan adalah versi 22.04

Step 1. Instalasi OpenJDK11

1. Install Java 11 (Open JDK)

```
sudo apt update
sudo apt install openjdk-11-jdk
```

2. Atur OS agar menggunakan OpenJDK11

```
sudo update-alternatives --config java
```

3. Periksa versi java

```
java -version
```

```
febri@thingsboard:~$ java -version
openjdk version "11.0.21" 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

Step 2. Instalasi Thingsboard Service Versi 3.6.2

1. Download paket instalasi thingsboard

```
wget
https://github.com/thingsboard/thingsboard/releases/download/v3.6.2/thingsbo
ard-3.6.2.deb
```

```
febri@thingsboard:~$ wget https://github.com/thingsboard/thingsboard/releases/download/v3.6.2/thingsboard-3.6.2.deb
```

2. Install paket Thingsboard

```
sudo dpkg -i thingsboard-3.6.2.deb
```

Step 3. Konfigurasi Database

1. Install PostgreSQL

```
sudo apt install postgresql-14
```

2. Jalankan service PostgreSQL

```
sudo service postgresql start
```

3. Ubah password untuk user postgresql

```
sudo su - postgres
psql
\password
\q
```

Ctrl + D untuk kembali.

4. Buat database thingsboard

```
psql -U postgres -d postgres -h 127.0.0.1 -W
CREATE DATABASE thingsboard;
\q
```

5. Edit file konfigurasi thingsboard

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

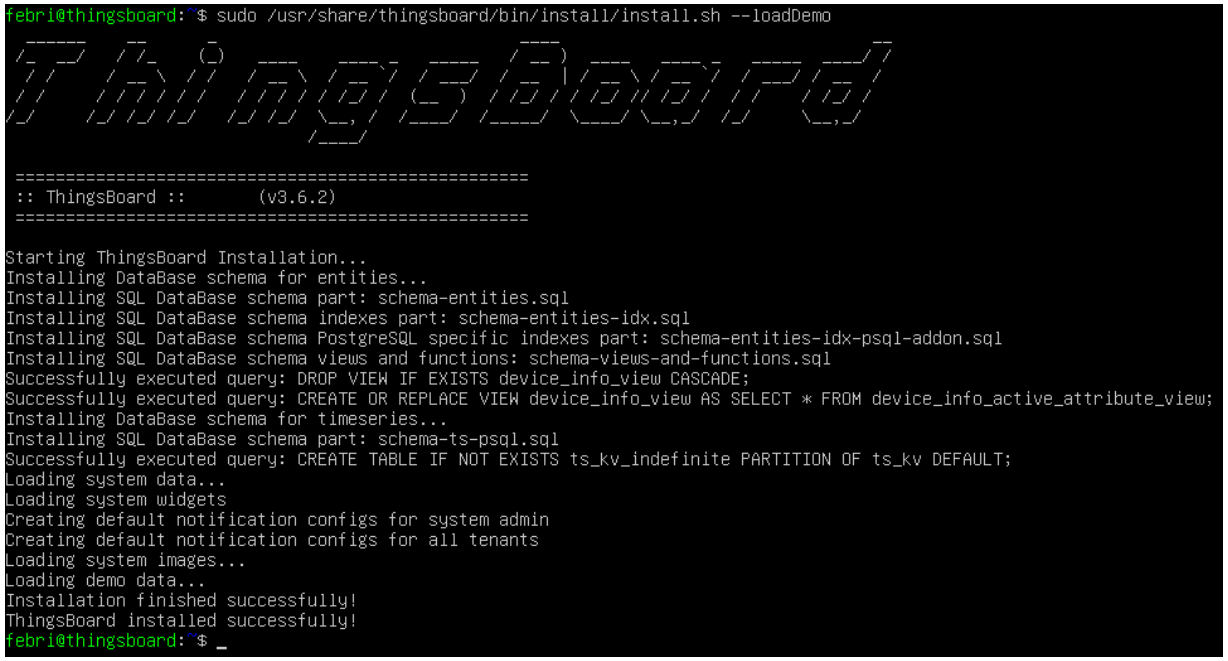
Tambahkan baris berikut

```
# DB Configuration
export DATABASE_TS_TYPE=sql
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=PASSWORD_UNTUK_POSTGRES
export SQL_POSTGRES_TS_KV_PARTITIONING=MONTHS
```

Step 4. Jalankan script instalasi

1. Setelah thingsboard terinstall dan database telah terkonfigurasi jalanakan

```
sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo
```



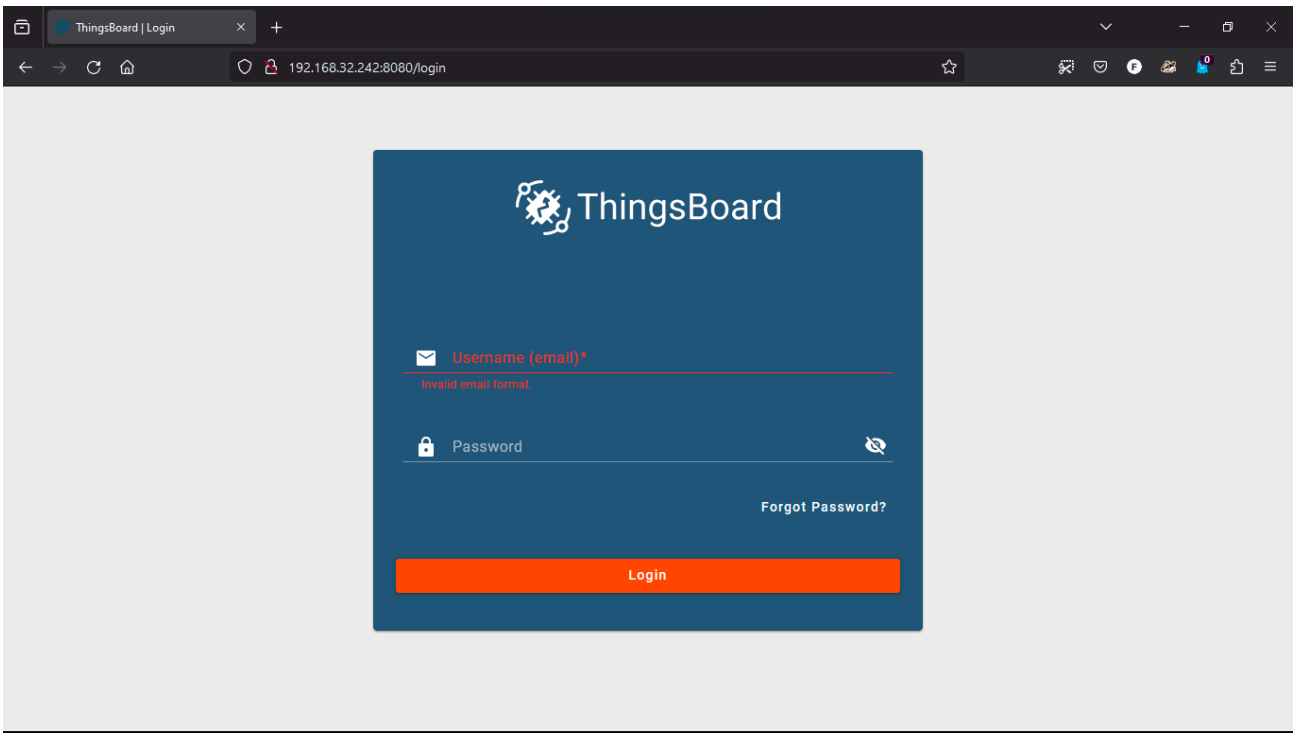
```
febri@thingsboard:~$ sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo

 ==================================================
 :: ThingsBoard ::       (v3.6.2)
 ==================================================

Starting ThingsBoard Installation...
Installing DataBase schema for entities...
Installing SQL DataBase schema part: schema-entities.sql
Installing SQL DataBase schema indexes part: schema-entities-idx.sql
Installing SQL DataBase schema PostgreSQL specific indexes part: schema-entities-idx-psql-addon.sql
Installing SQL DataBase schema views and functions: schema-views-and-functions.sql
Successfully executed query: DROP VIEW IF EXISTS device_info_view CASCADE;
Successfully executed query: CREATE OR REPLACE VIEW device_info_view AS SELECT * FROM device_info_active_attribute_view;
Installing DataBase schema for timeseries...
Installing SQL DataBase schema part: schema-ts-psql.sql
Successfully executed query: CREATE TABLE IF NOT EXISTS ts_kv_indefinite PARTITION OF ts_kv DEFAULT;
Loading system data...
Loading system widgets
Creating default notification configs for system admin
Creating default notification configs for all tenants
Loading system images...
Loading demo data...
Installation finished successfully!
ThingsBoard installed successfully!
febri@thingsboard:~$ _
```

Step 5. Jalankan service thingsboard

1. Dengan menggunakan perintah

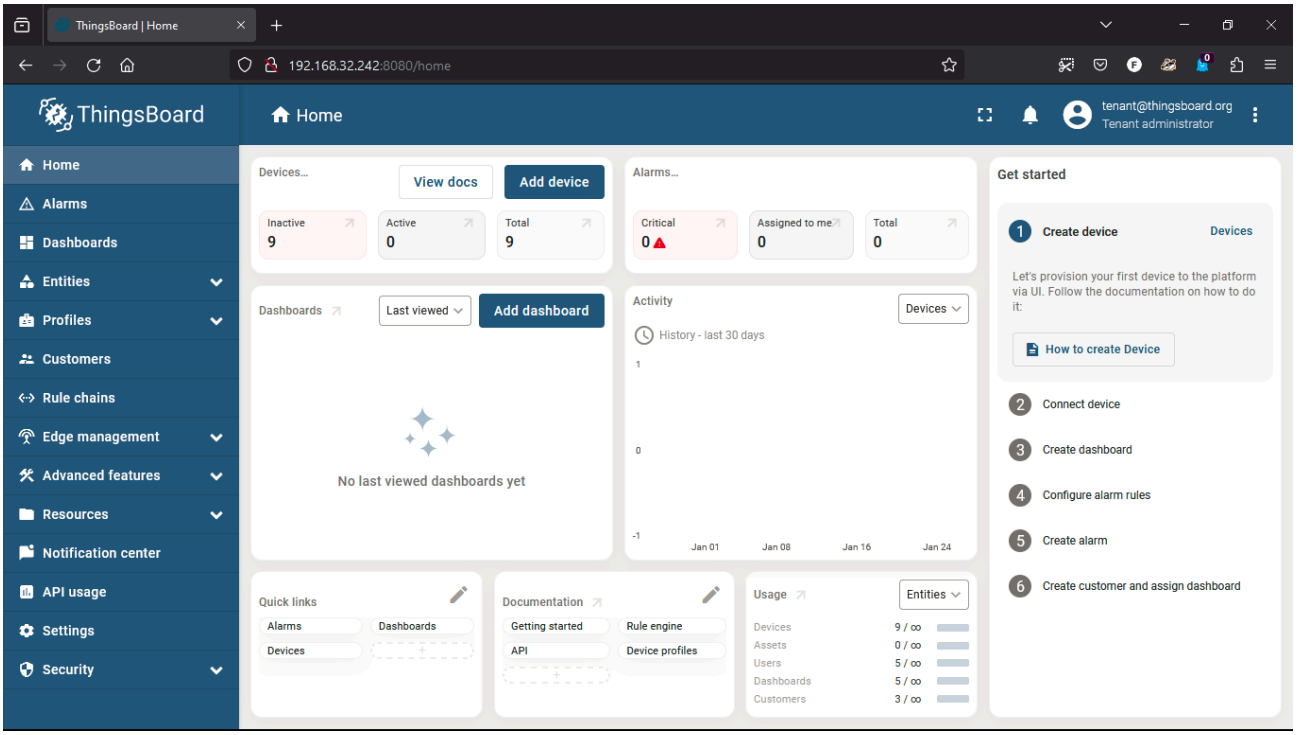
```
sudo service thingsboard start
```

2. Buka browser dan akses alamat ip ubuntu server dengan port 8080, terkadang butuh beberapa saat hingga dashboard thingsboard dapat diakses



Akun default yang dapat digunakan ketika menginstall thingsboard dengan parameter `--loadDemo`

- **System Administrator**: sysadmin@thingsboard.org / sysadmin
- **Tenant Administrator**: tenant@thingsboard.org / tenant
- **Customer User**: customer@thingsboard.org / customer



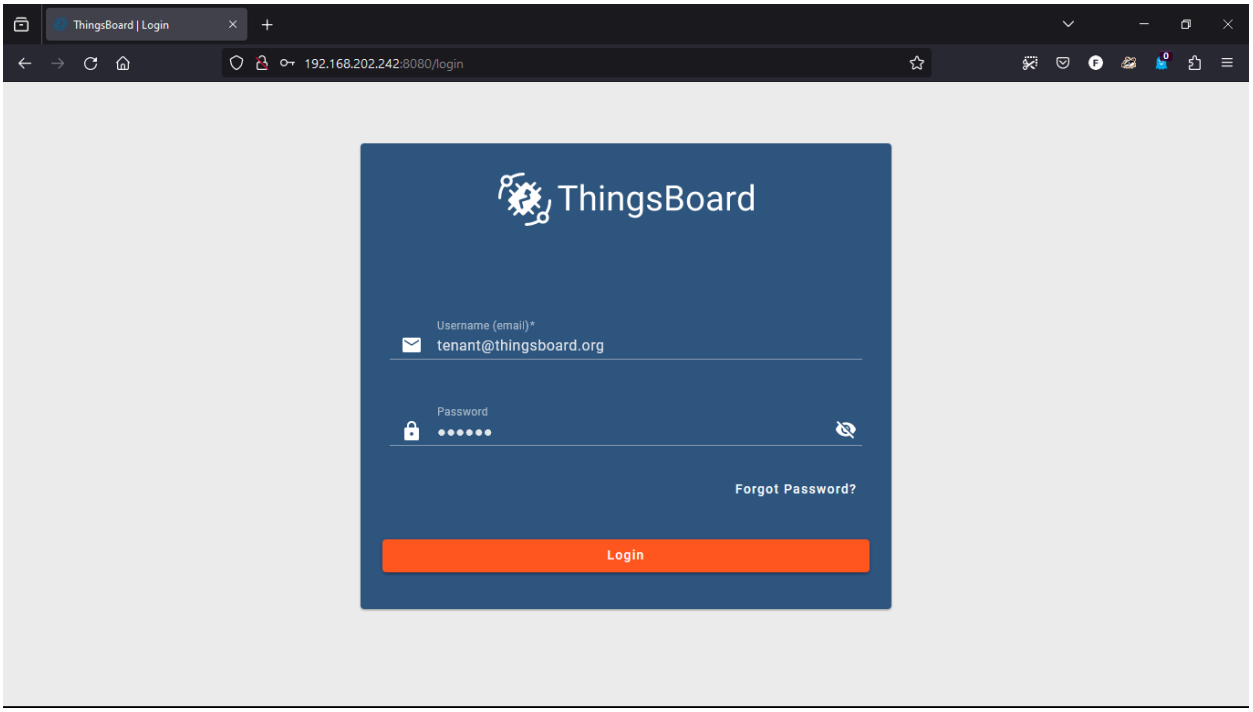
Referensi :

https://thingsboard.io/docs/user-guide/install/installation-options/
https://thingsboard.io/docs/user-guide/install/ubuntu/

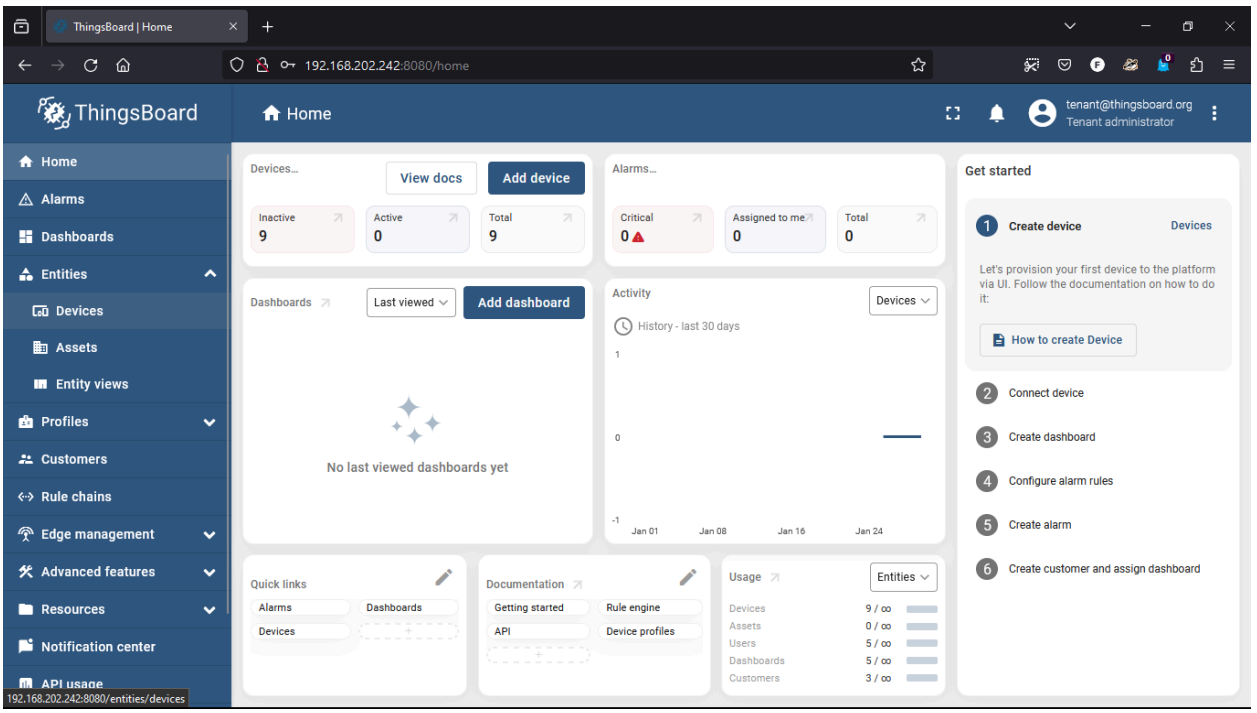
Menampilkan data sensor pada Dashboard Thingsboard

Step 1. Menambahkan Device

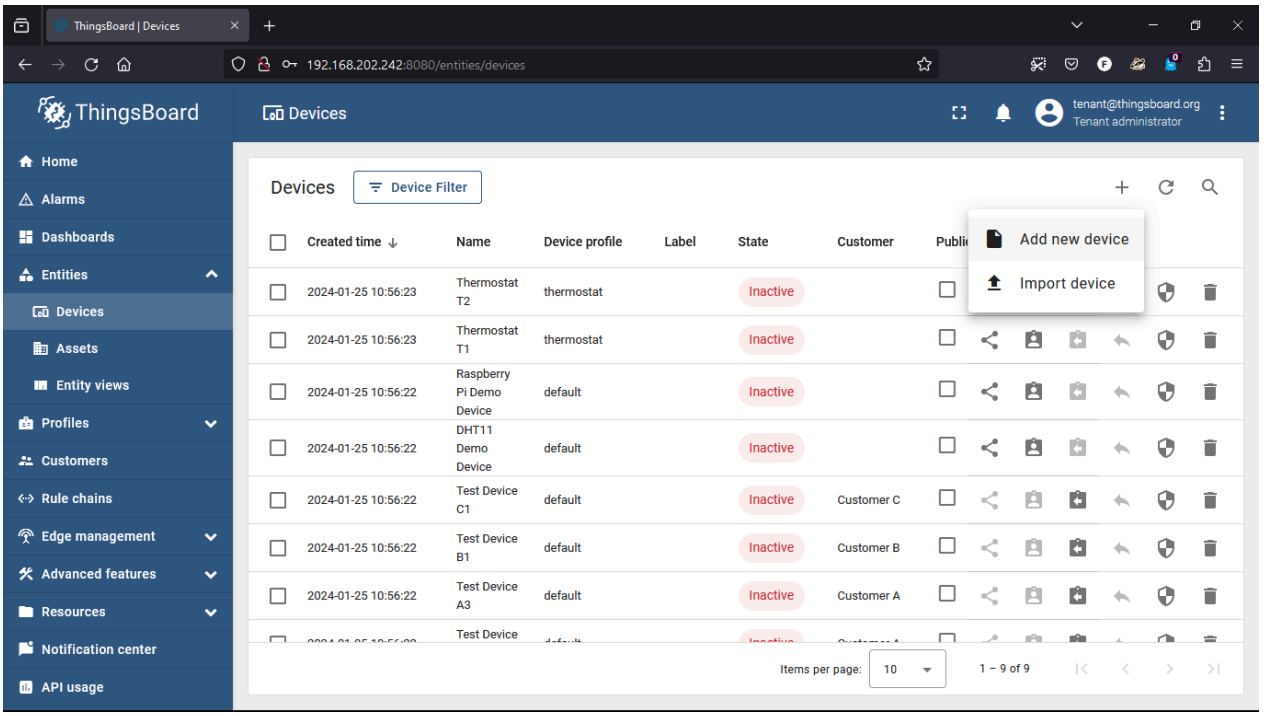
1. Buka alamat server Thingsboard dan login menggunakan akun yang ada.



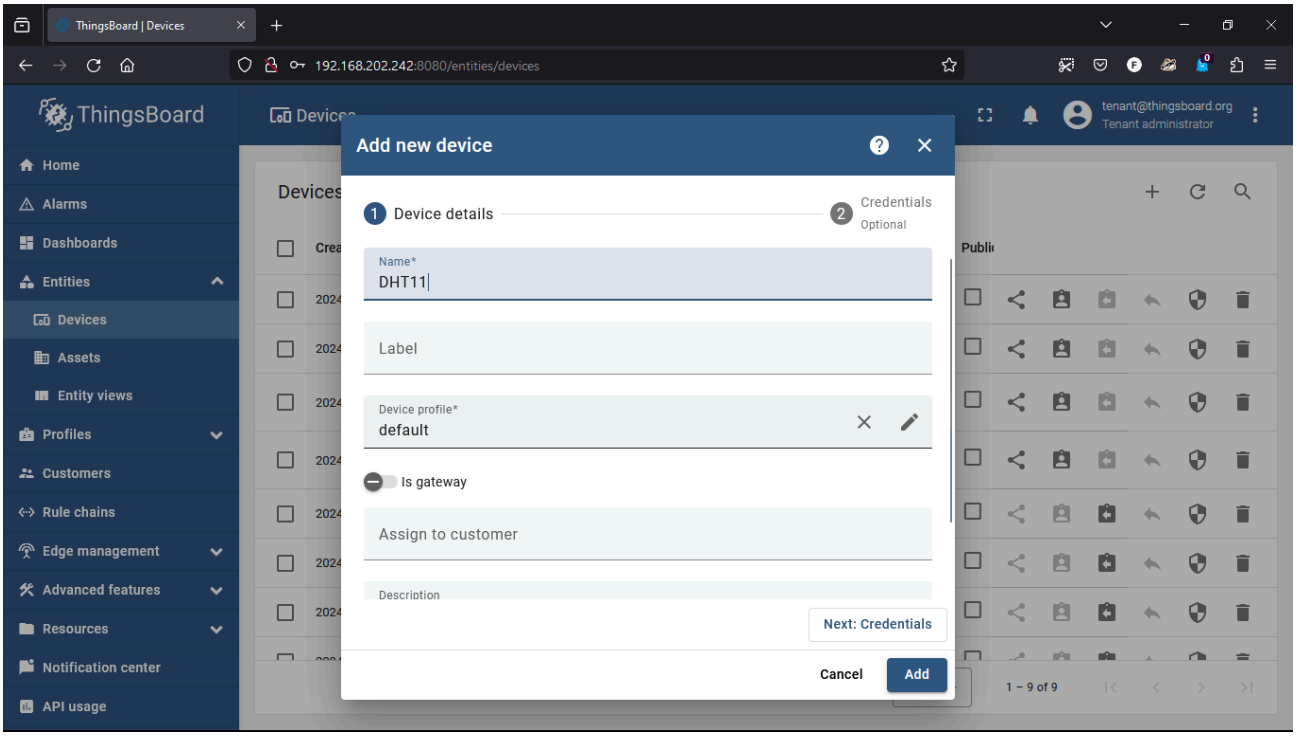
2. Di halaman home Thingsboard, pada panel sebelah kiri klik pada menu Entities > Devices



3. Pada halaman Devices tambahkan device baru

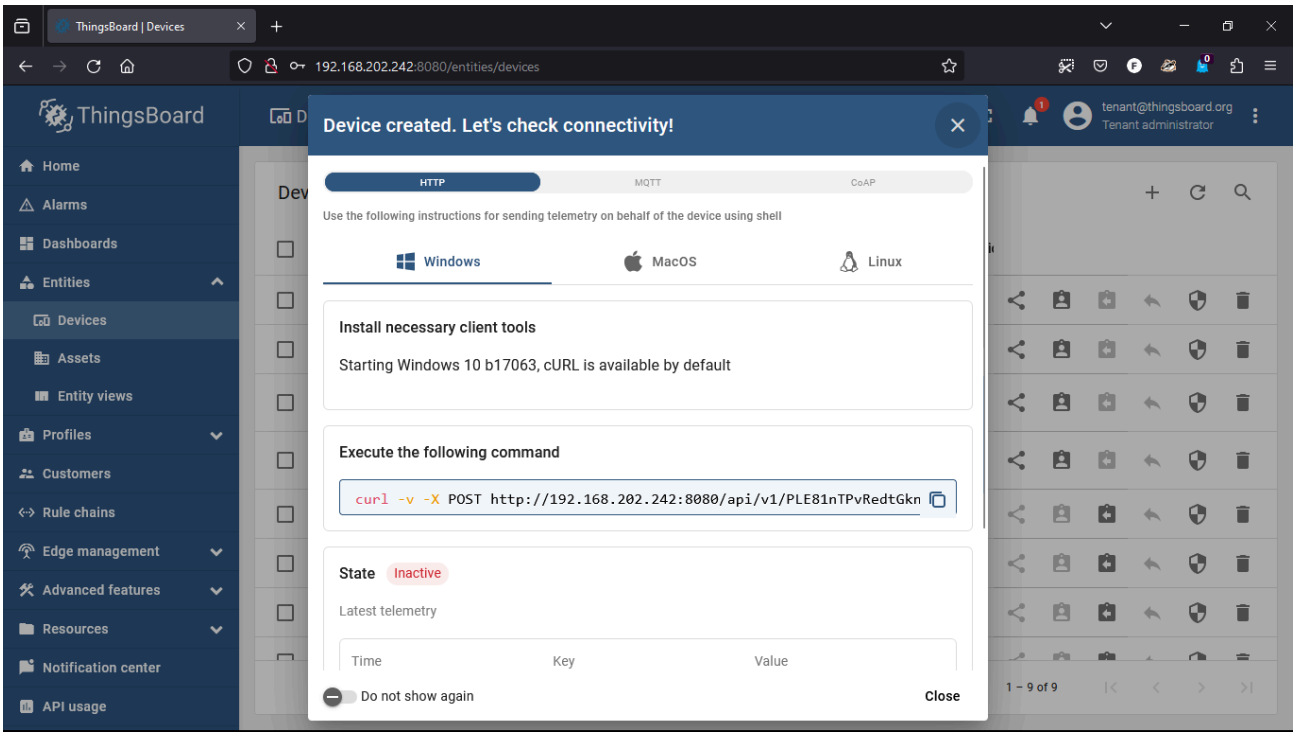


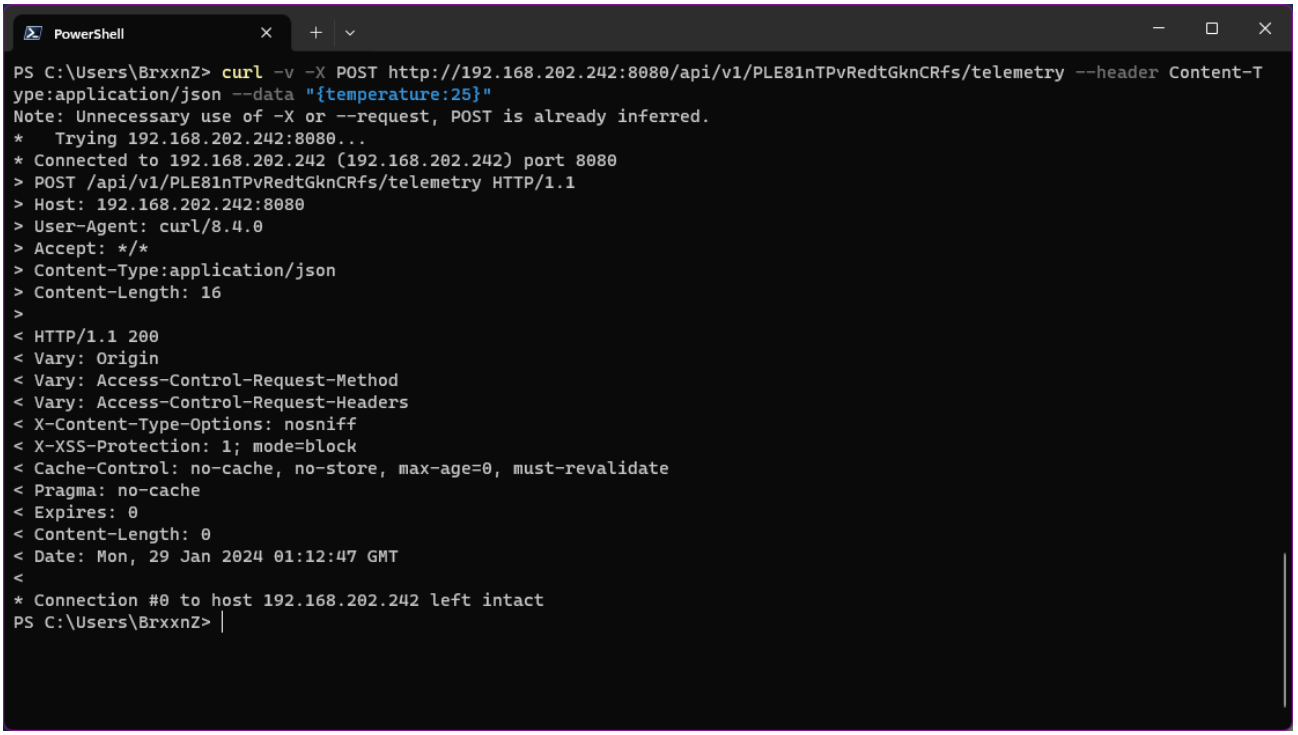
4. Masukkan informasi untuk device yang akan ditambahkan setelah itu klik Add

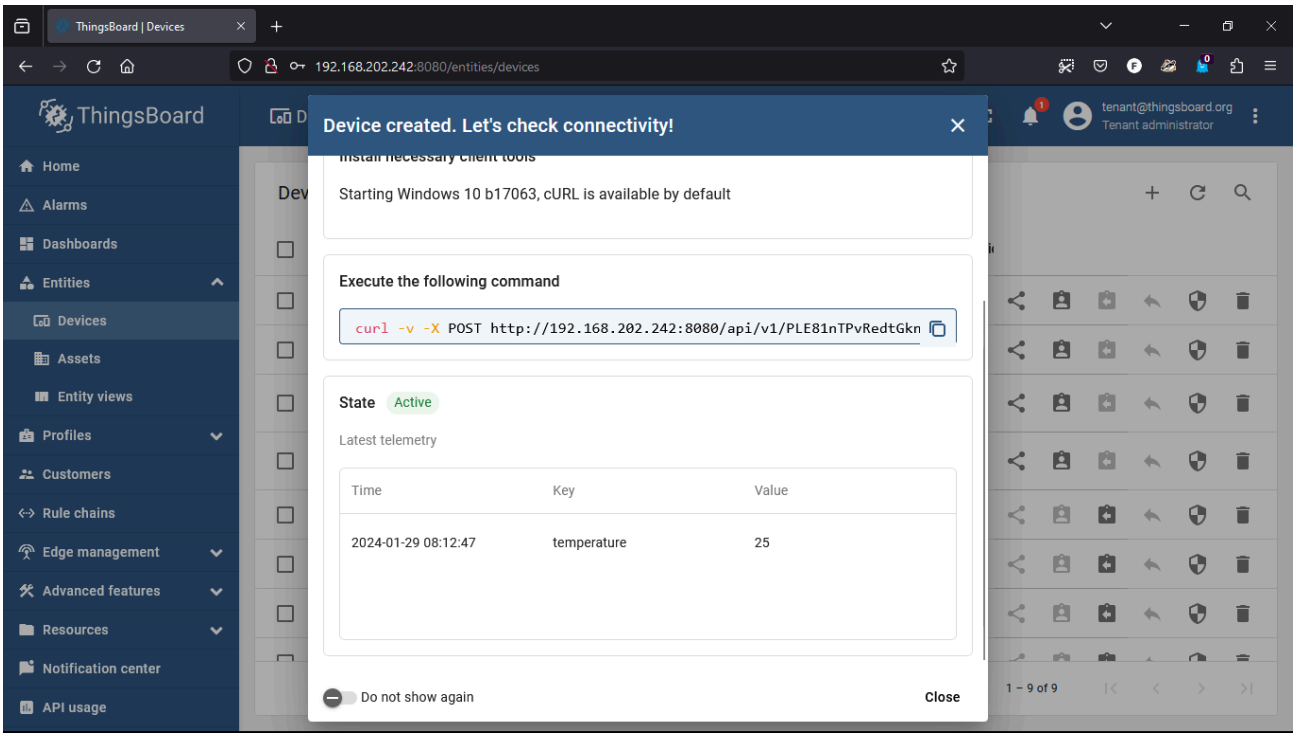


Step 2. Melakukan Pengujian

1. Halaman selanjutnya akan menampilkan instruksi untuk menguji konektivitas ke thingsboard device



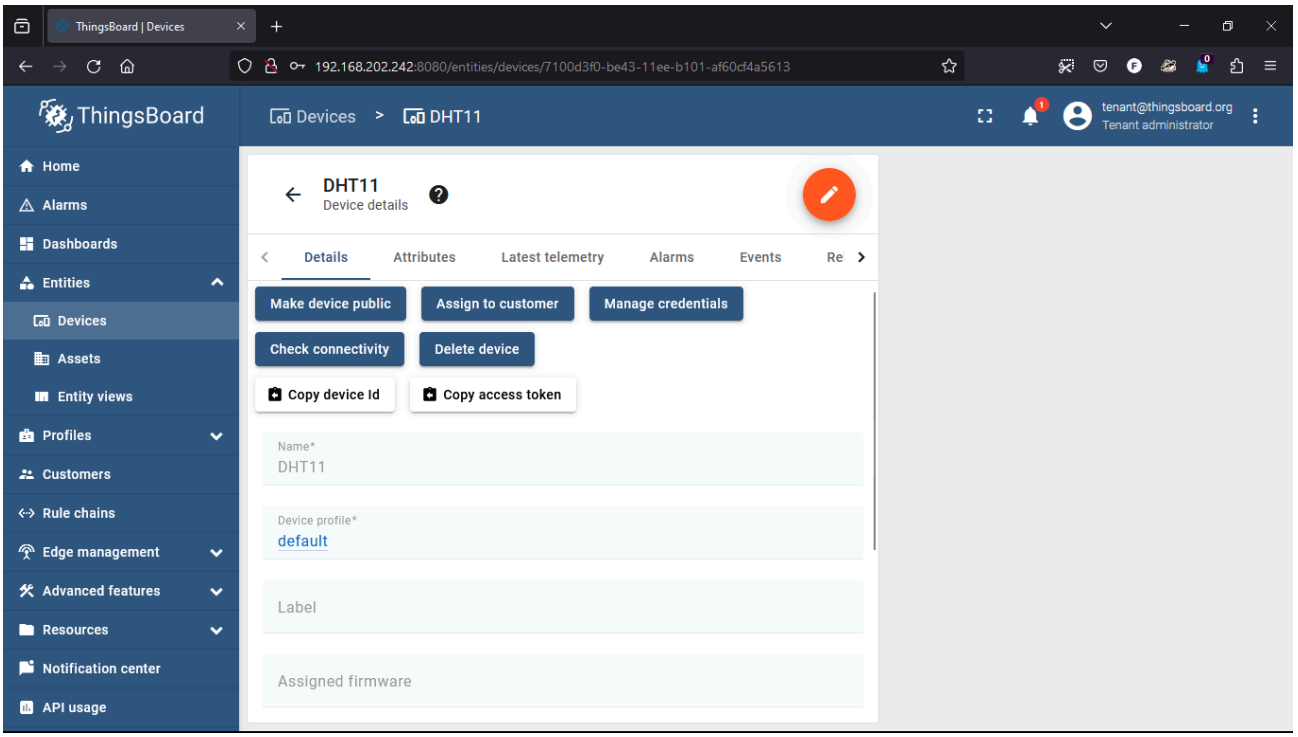


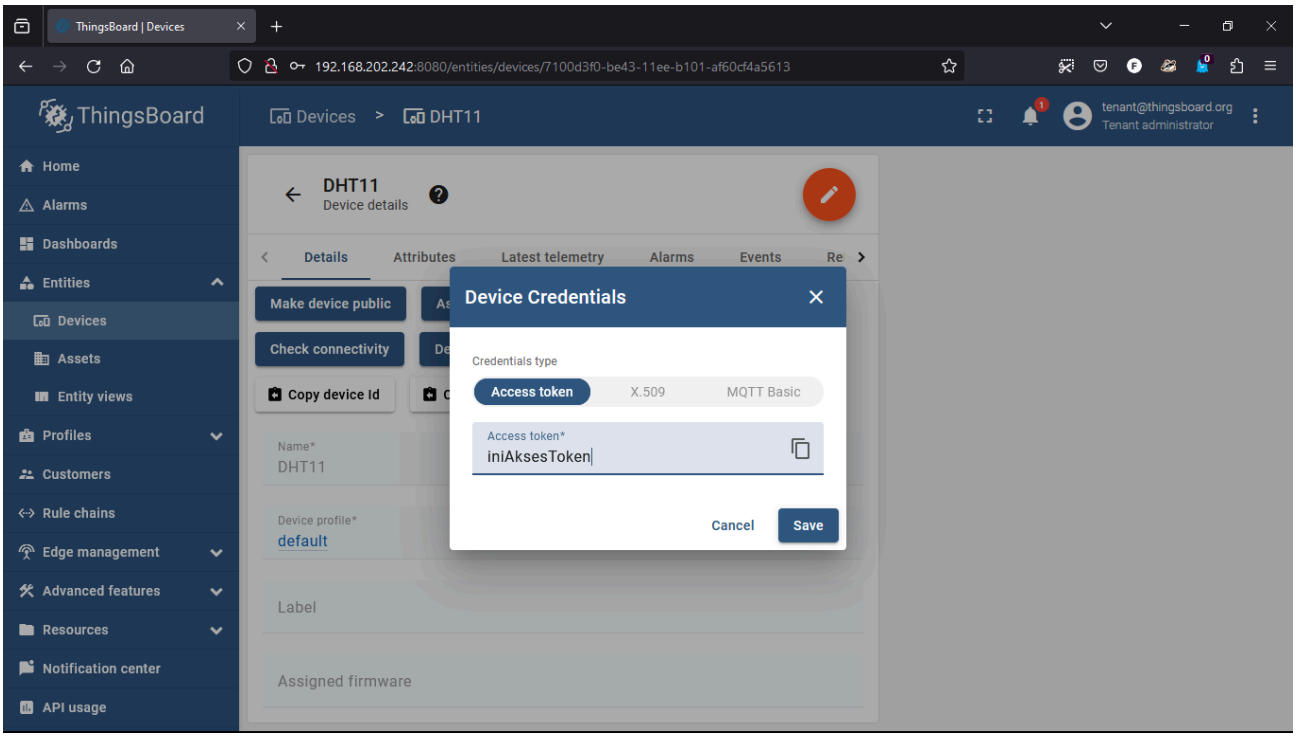


Pastikan bahwa hasil pengujian konektivitas berhasil dan pada Device muncul data telemetry.

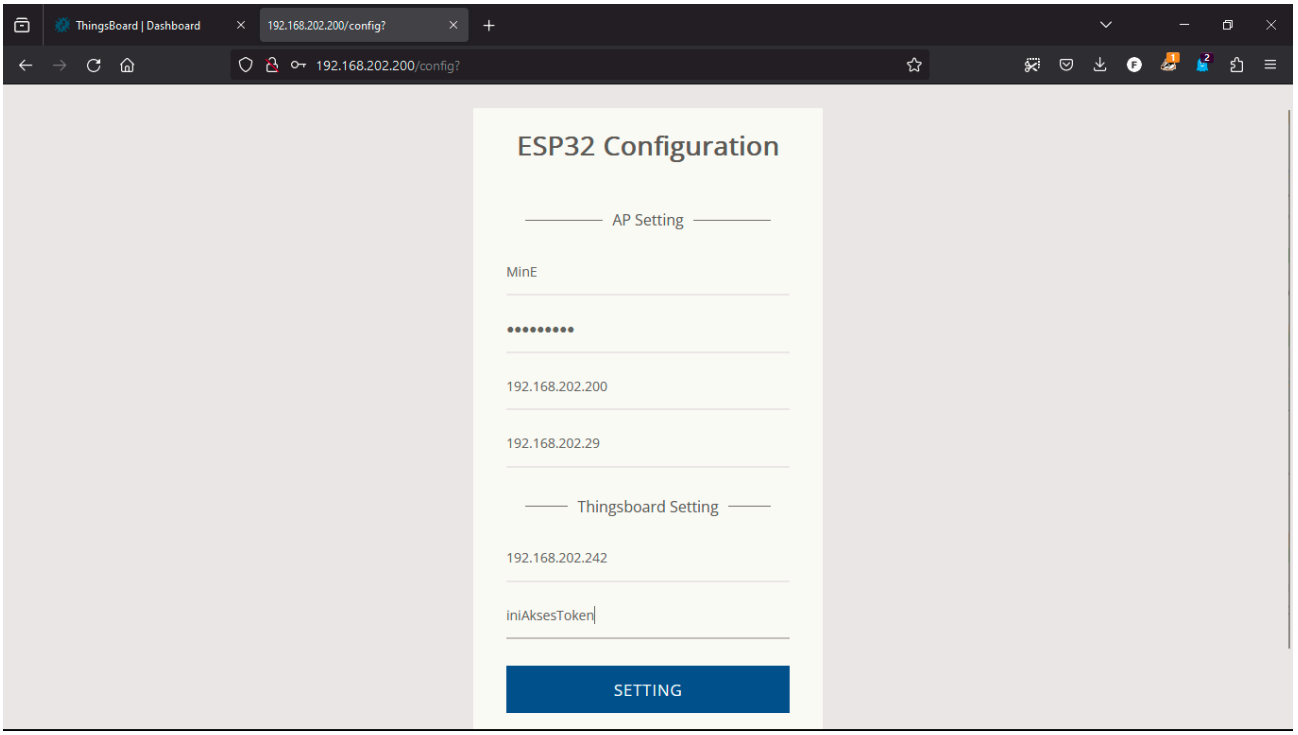
Step 3. Mengubah Token Akses (Opsional)

1. Untuk mengubah token akses, pada halaman Device Details, klik pada tombol Manage Credential

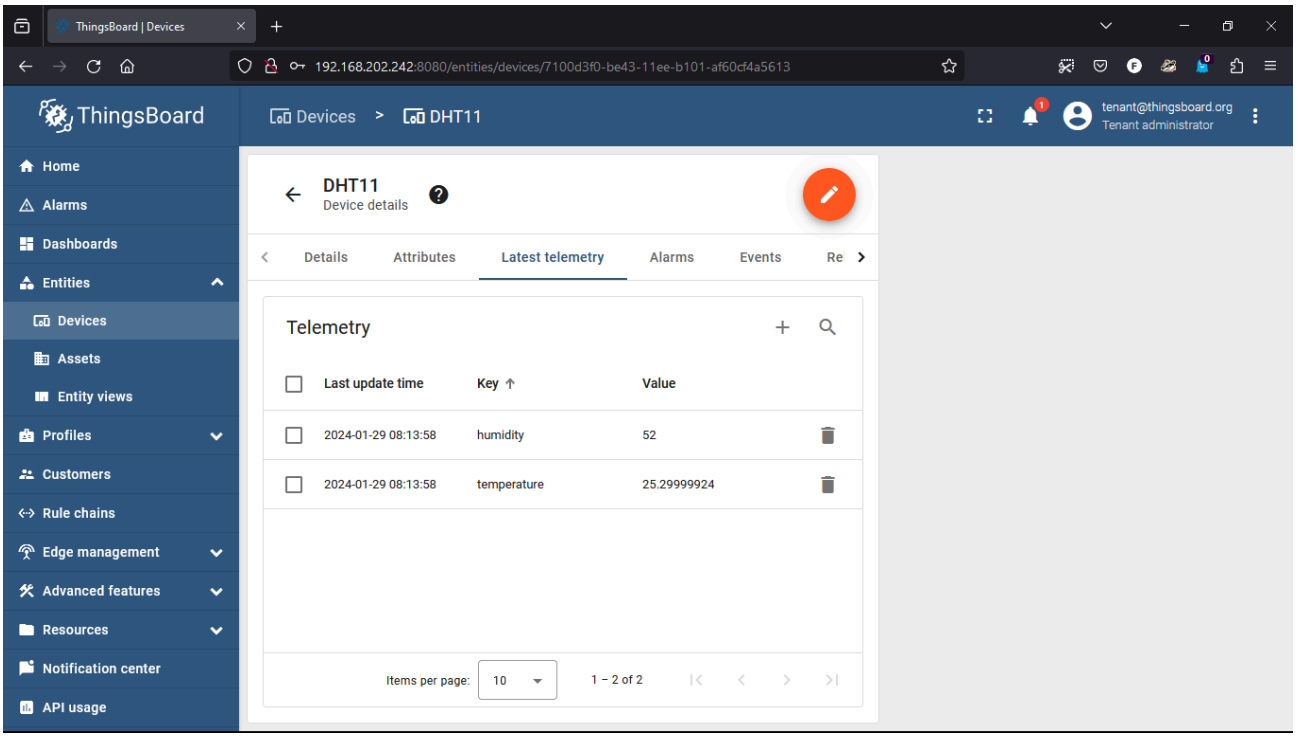




2. Konfigurasi pada perangkat ESP32

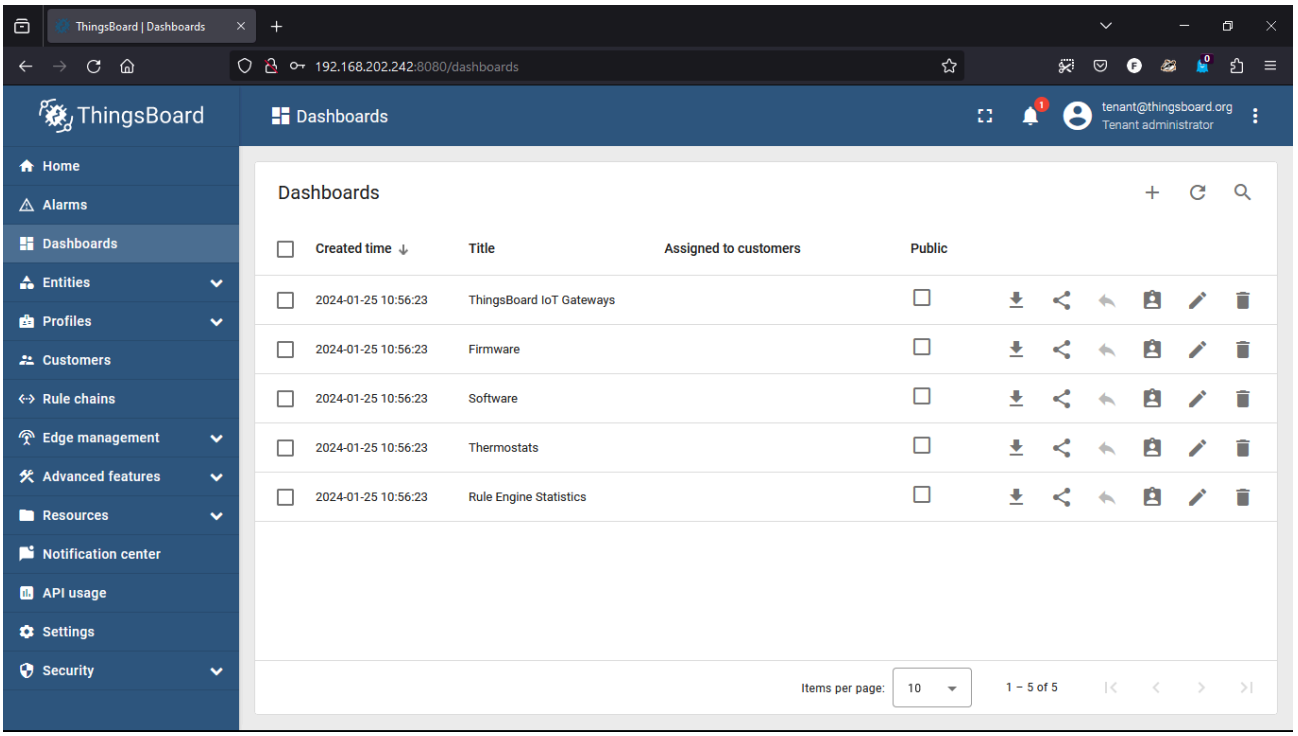


3. Jika konfigurasi telah sesuai maka akan muncul data telemetry baru.

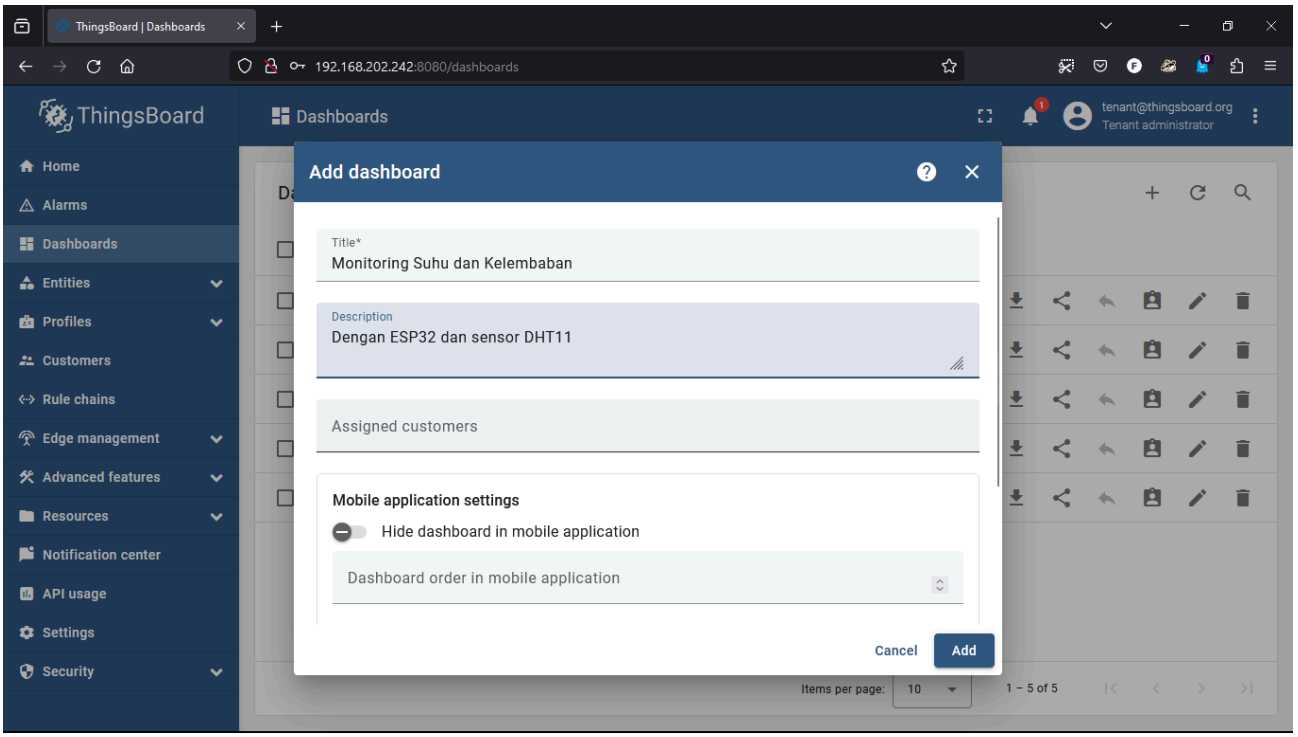


Step 4. Menampilkan data ke Dashboard

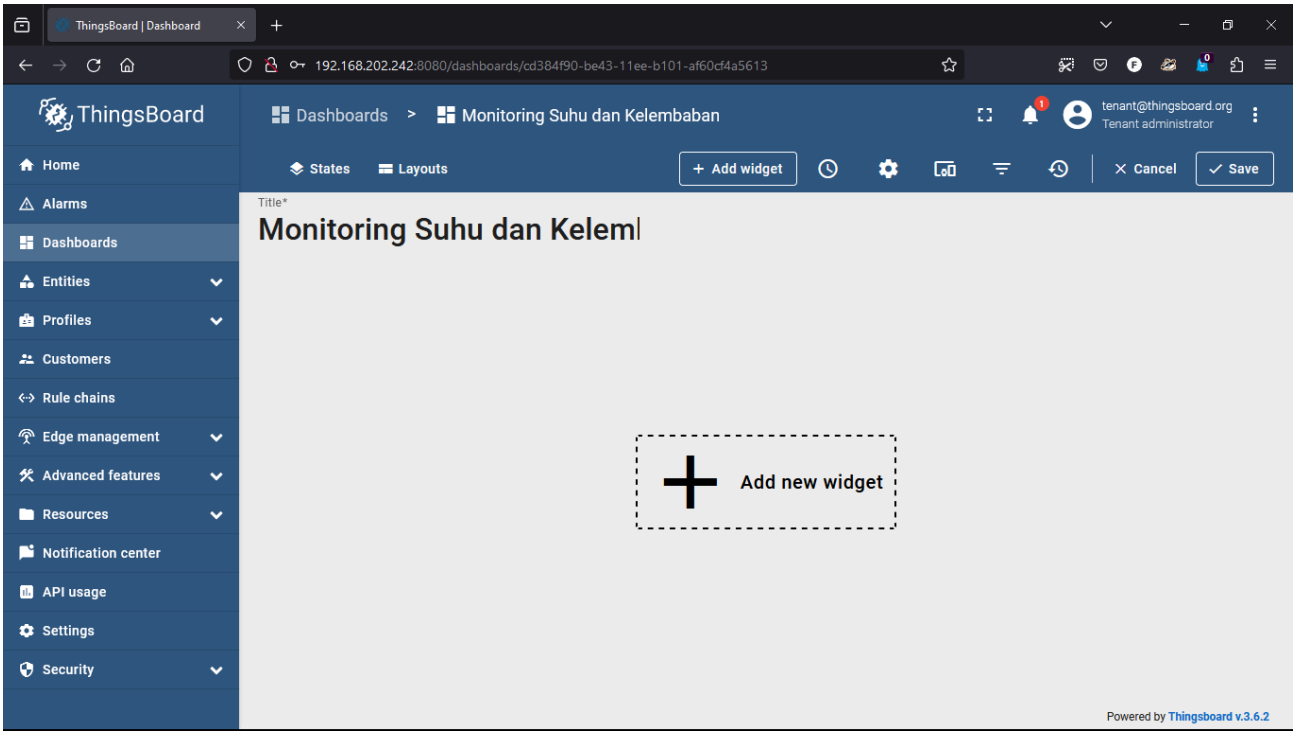
1. Klik pada tombol Dashboard di panel menu kiri untuk berpindah ke halaman Dashboard

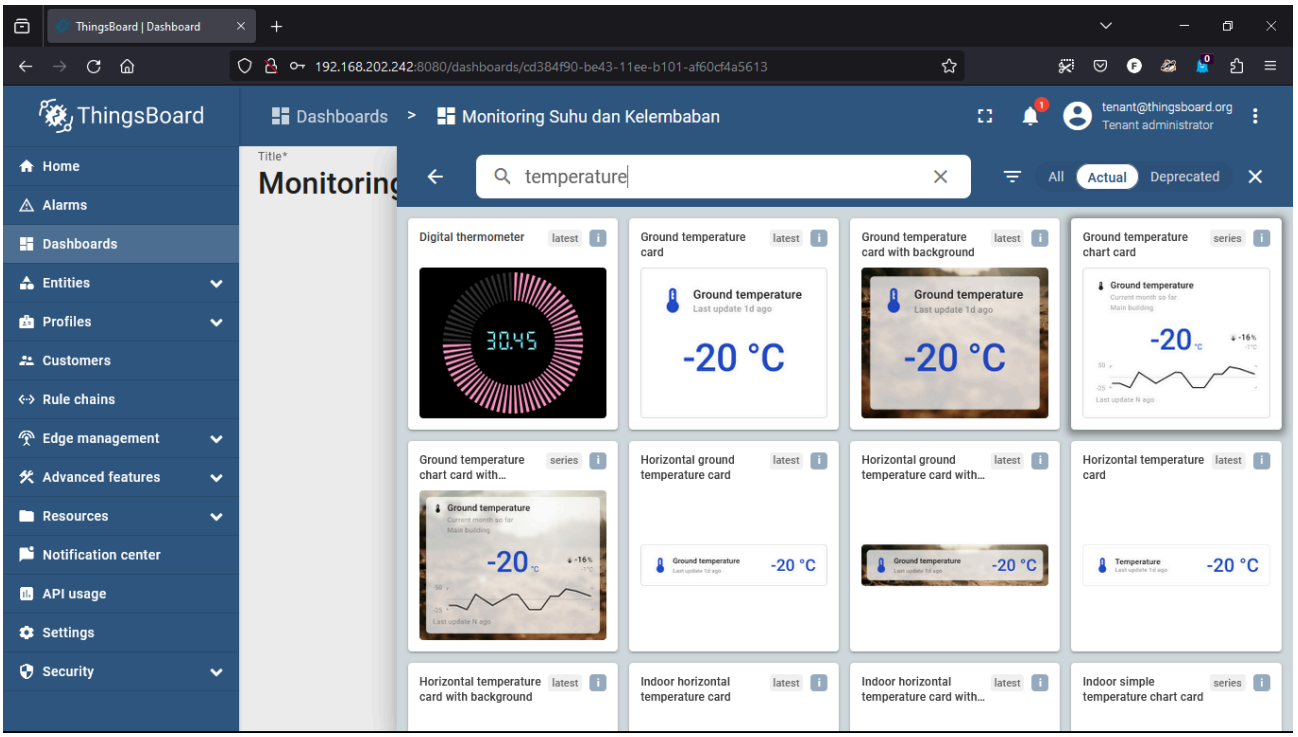


2. Tambahkan dashboard baru dan masukkan informasi untuk dashboard yang akan ditambahkan.

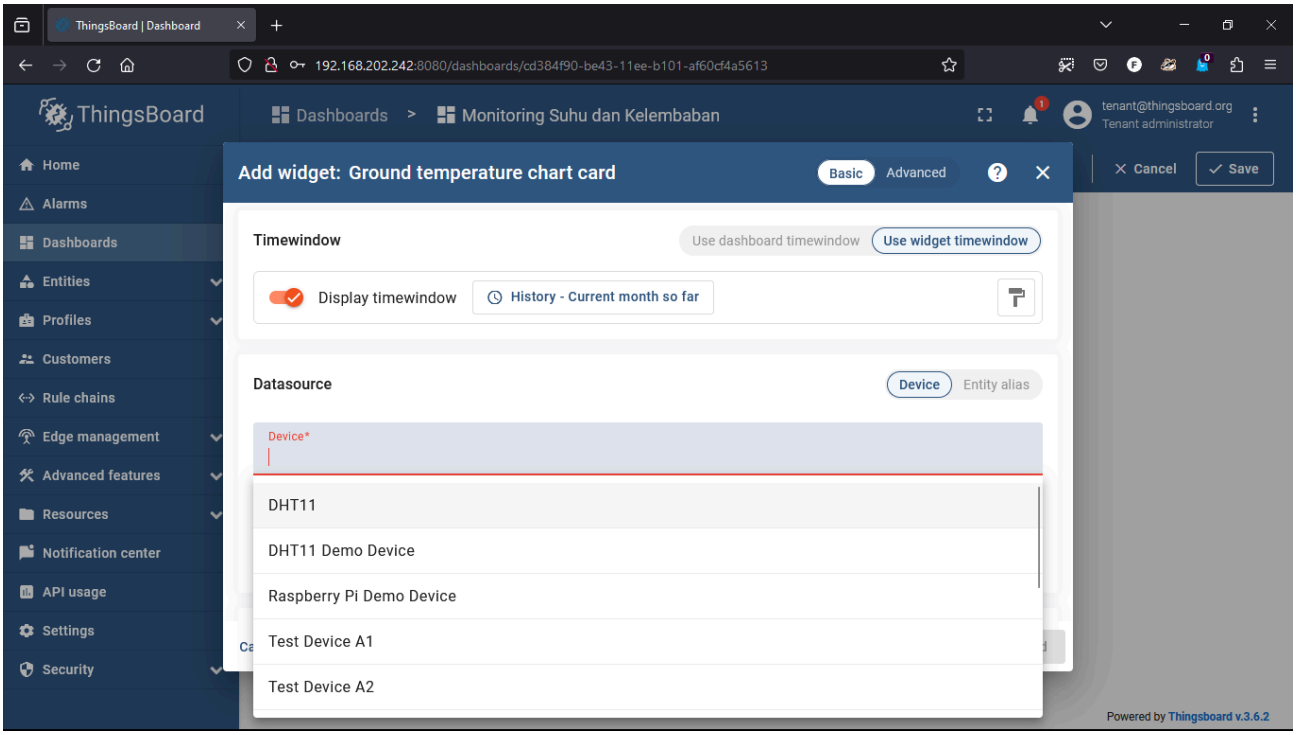


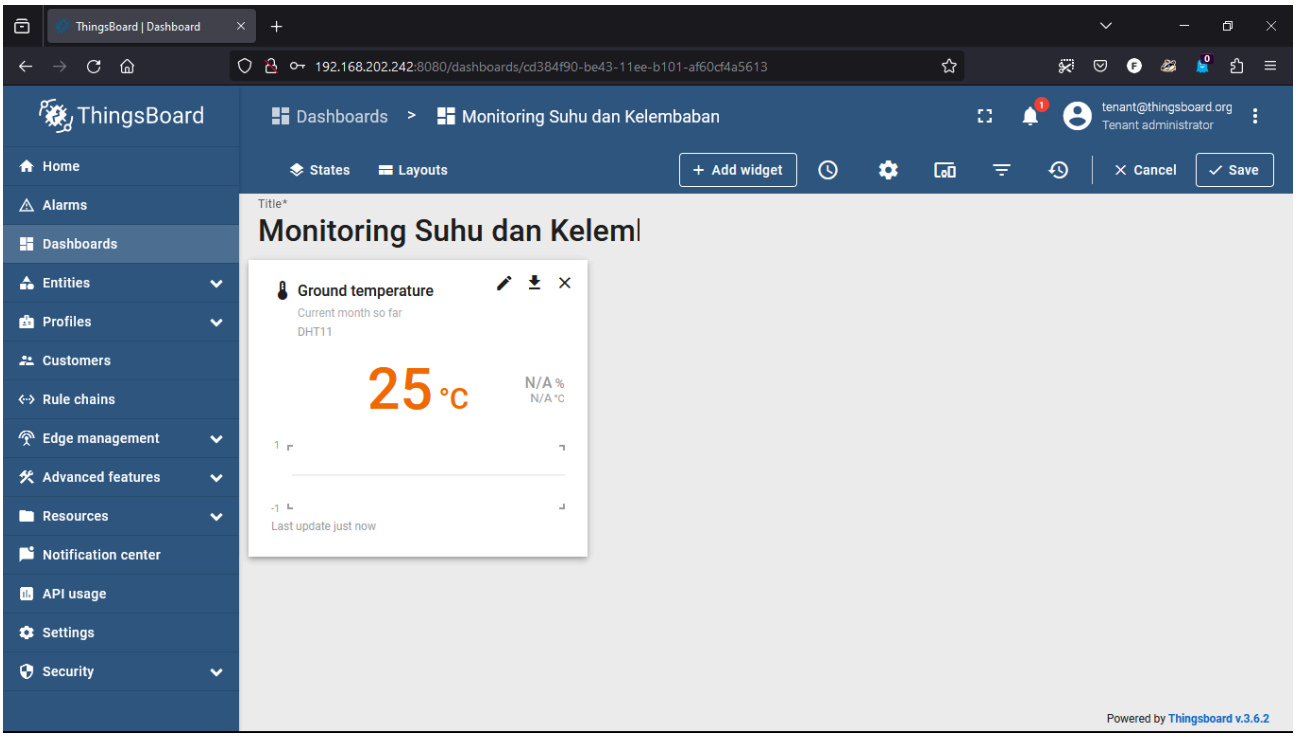
3. Pada halaman dashboard yang baru, tambahkan widget yang sesuai.



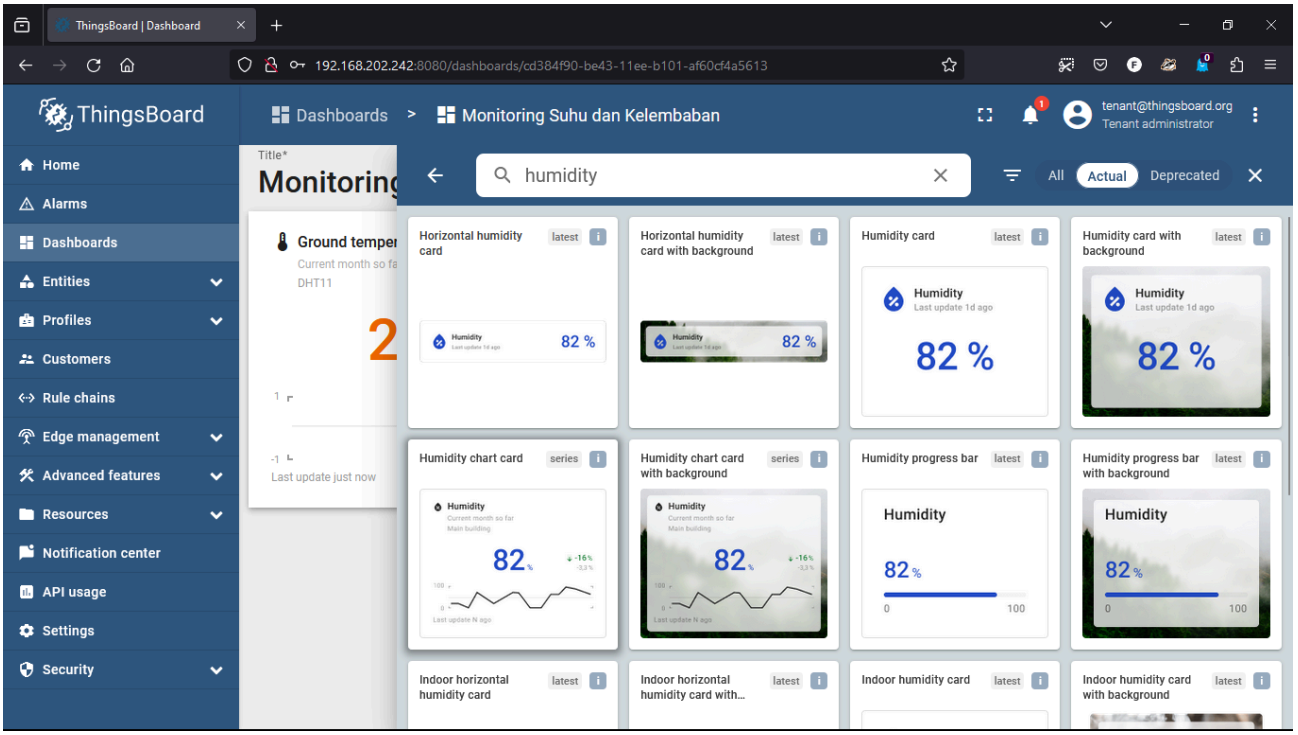


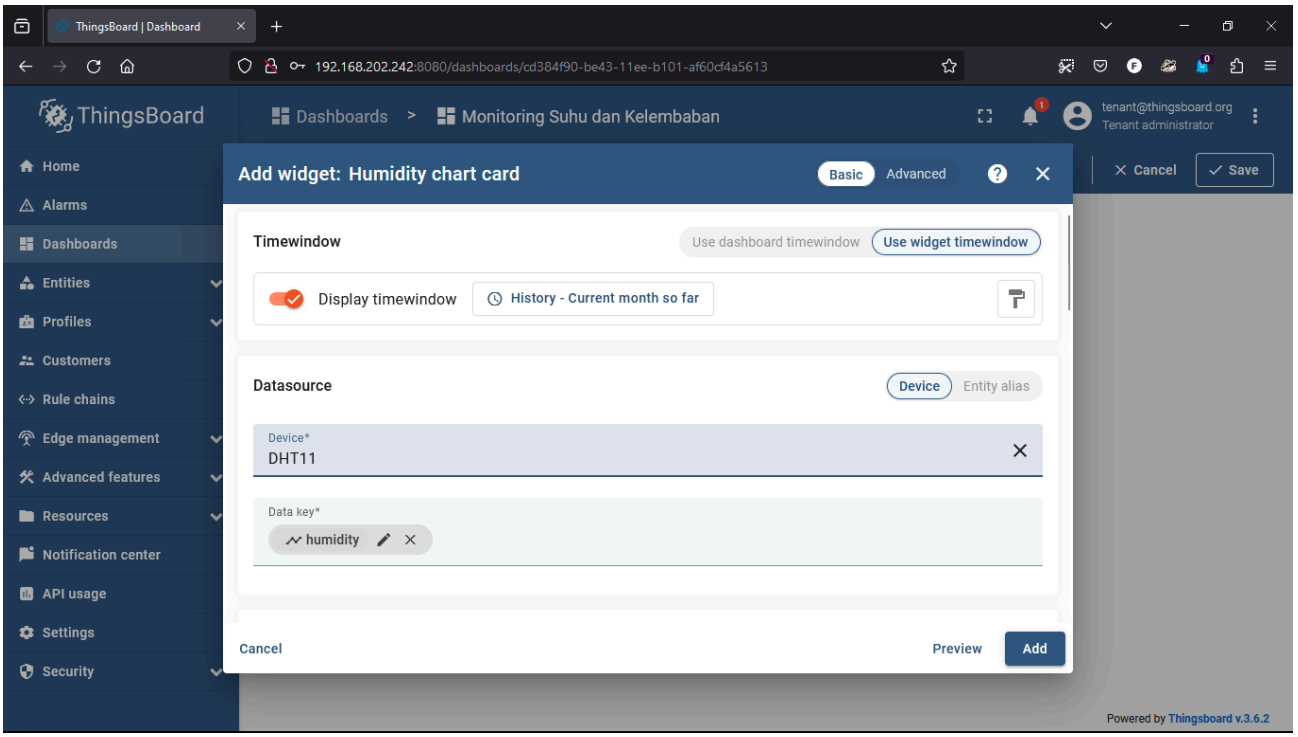
4. Atur bagian datasource dan data key agar menggunakan device yang telah dibuat sebelumnya.



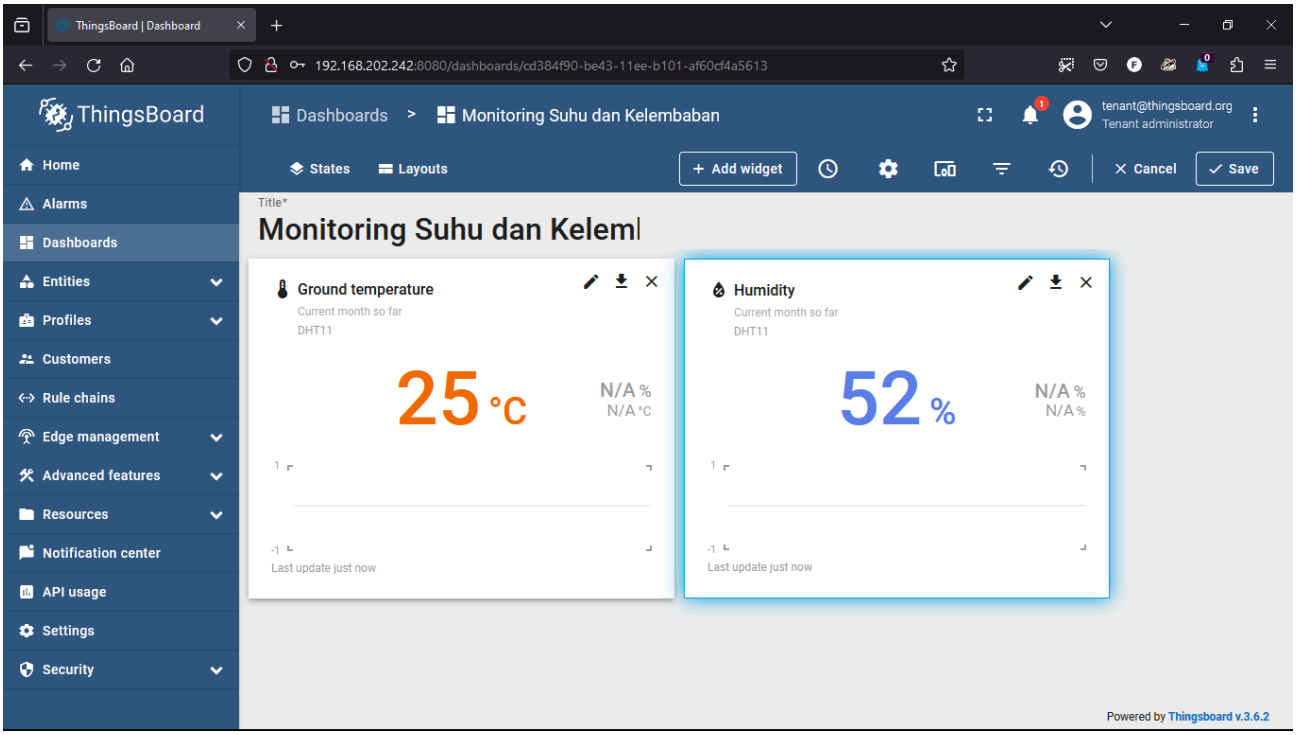


5. Tambahkan widget baru untuk menampilkan data yang lain

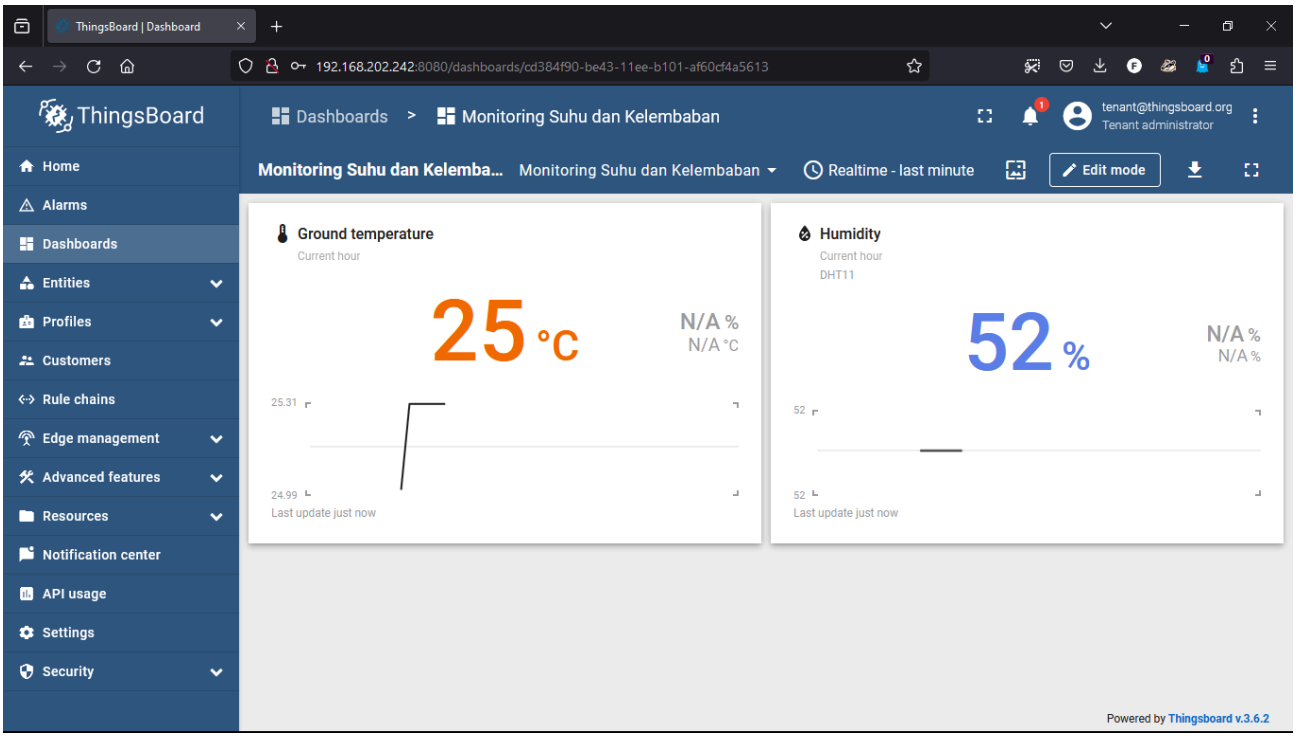




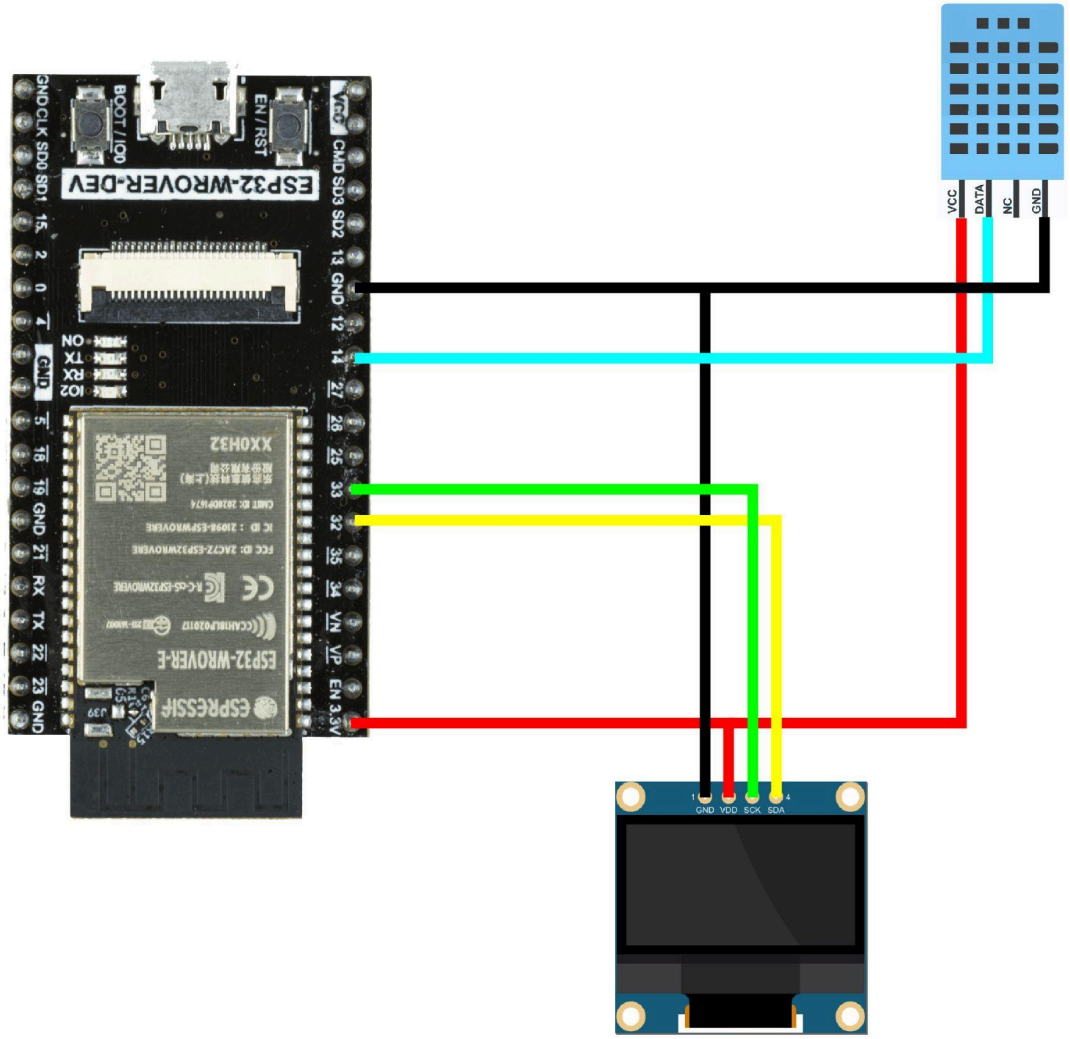
6. Dan hasil akhirnya sudah ada 2 widget untuk menampilkan data hasil pembacaan sensor DHT11. Jangan lupa klik Save untuk menyimpan dashboard.



7. Tampilan akhirnya akan menjadi seperti berikut.


ThingsBoard | Dashboard
192.168.202.242:8080/dashboards/cd384f90-be43-11ee-b101-af60cf4a5613
ThingsBoard
Dashboards > Monitoring Suhu dan Kelembaban
tenant@thingsboard.org
Tenant administrator
Home
Alarms
Dashboards
Entities
Profiles
Customers
Rule chains
Edge management
Advanced features
Resources
Notification center
API usage
Settings
Security
Monitoring Suhu dan Kelemba...
Monitoring Suhu dan Kelembaban
Realtime - last minute
Edit mode
Ground temperature
Current hour
25 °C
N/A %
N/A °C
25.31
24.99
Last update just now
Humidity
Current hour
DHT11
52 %
N/A %
N/A %
52
52
Last update just now
Powered by Thingsboard v.3.6.2

Skematik ESP32 DHT11 + QR



Ket:

Perangkat yang digunakan :

1. ESP32 Wrover Dev + Cam
2. Modul 1306 OLED Display
3. Modul Sensor DHT11

Pengkabelan :

1. SDA pada OLED disambungkan ke pin 32 pada ESP32 Wrover Dev
2. SCL pada OLED disambungkan ke pin 33 pada ESP32 Wrover Dev
3. GND (Ground) pada OLED disambungkan ke GND (Ground) pada pada ESP32 Wrover Dev
4. VCC pada OLED disambungkan ke 3.3V pada pada ESP32 Wrover Dev
5. Data pada DHT11 disambungkan ke pin 14 pada ESP32 Wrover Dev
6. GND (Ground) pada DHT11 disambungkan ke GND (Ground) pada pada ESP32 Wrover Dev
7. VCC pada DHT11 disambungkan ke 3.3V pada pada ESP32 Wrover Dev

Penjelasan Kode

Import Library yang diperlukan

```
// Library yang dibutuhkan
#include <ThingsBoard.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include <SPIFFS.h>
#include <DHT.h>
#include <Arduino_MQTT_Client.h>
#include <ArduinoJson.h>
#include <ESP32QRCodeReader.h>
#include <Wire.h>
#include "SSD1306Wire.h"
#include <qrcodeoled.h>
#include <HTTPClient.h>
```

Melakukan pendefinisian parameter yang berkaitan dengan tipe display

```
// Tipe Layar yang digunakan adalah adalah OLED
#define OLEDDISPLAY

// Ukuran OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Pin SDA OLED ke pin 32 ESP32
#define OLED_SDA 32
// Pin SCL OLED ke pin 33 ESP32
#define OLED_SCL 33
```

Konfigurasi untuk sensor DHT11

```
// Pengaturan pin dan tipe DHT yang digunakan
#define DHTPIN 14 // Pin data DHT -> Pin 14 ESP32
#define DHTTYPE DHT11
DHT dhtSensor(DHTPIN, DHTTYPE);
```

Instansiasi Object

```
// Instansiasi Class untuk display, camera, dan qrcode
SSD1306Wire display(0x3c, OLED_SDA, OLED_SCL);
ESP32QRCodeReader reader(CAMERA_MODEL_WROVER_KIT);
QRcodeOled qrcode (&display); // Class qrcode untuk generate QR
```

Instansiasi Object untuk WiFi dan Thingsboard

```
WiFiClient wifiClient;
// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

Arduino_MQTT_Client mqttClient(wifiClient);
ThingsBoard tb(mqttClient);
```

Variabel untuk pengaturan timing dan interval

```
// Variabel untuk timing
unsigned long previousMillis;
unsigned long sendDataMillis;

int send_delay = 1000; // Interval untuk mengirim data ke server
unsigned long millis_counter;
const long intervalWifi = 10000; // interval to wait for Wi-Fi connection
(milliseconds)

unsigned long prevMillis; // Variabel untuk menyimpan data waktu layar OLED
ditampilkan
unsigned long prevMillisQR;
const long intervalQr = 3000; // Interval untuk reset display setelah
menampilkan hasil scan QR
```

readDHTTemperature()

Fungsi ini digunakan untuk membaca data Suhu dari sensor DHT dan mengembalikannya dalam bentuk String. Jika data tidak berhasil dibaca dari sensor maka yang akan dikembalikan adalah karakter “--”.

```
// Fungsi untuk membaca nilai temperatur dari sensor DHT
String readDHTTemperature() {
  float t = dhtSensor.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  } else {
    // Serial.println(t);
    return String(t);
  }
}
```

readDHTHumidity()

Fungsi ini digunakan untuk membaca data Kelembaban dari sensor DHT dan mengembalikannya dalam bentuk String. Jika data tidak berhasil dibaca dari sensor maka yang akan dikembalikan adalah karakter “--”.

```
// Fungsi untuk membaca nilai kelembaban dari sensor DHT
```

```
String readDHTHumidity() {
  float h = dhtSensor.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  } else {
    // Serial.println(h);
    return String(h);
  }
}
```

readFile()

Fungsi untuk membaca konten dari file di SPIFFS

```
// Read File from SPIFFS
String readFile(fs::FS &fs, const char *path) {
  Serial.printf("Reading file: %s\r\n", path);

  File file = fs.open(path);
  if (!file || file.isDirectory()) {
    Serial.println("- failed to open file for reading");
    return String();
  }

  String fileContent;
  while (file.available()) {
    fileContent = file.readStringUntil('\n');
    break;
  }
  return fileContent;
}
```

writeFile()

Fungsi untuk menulis konten ke file di SPIFFS

```
// Write file to SPIFFS
void writeFile(fs::FS &fs, const char *path, const char *message) {
  Serial.printf("Writing file: %s\r\n", path);

  File file = fs.open(path, FILE_WRITE);
  if (!file) {
    Serial.println("- failed to open file for writing");
    return;
  }
  if (file.print(message)) {
    Serial.println("- file written");
  } else {
    Serial.println("- write failed");
```

```
  }
}
```

initWiFi()

Fungsi ini digunakan untuk mengkonfigurasi ESP32 dalam mode Station jika terdapat data SSID dan Password dalam file konfigurasi, apabila dalam interval 10 detik ESP32 tidak dapat terhubung ke WiFi maka fungsi ini akan mengembalikan nilai False.

```
// Initialize WiFi
bool initWiFi() {
  if (ssid == "" || ip == "") {
    Serial.println("Undefined SSID or IP address.");
    return false;
  }

  WiFi.mode(WIFI_STA);
  localIP.fromString(ip.c_str());
  localGateway.fromString(gateway.c_str());

  if (!WiFi.config(localIP, localGateway, subnet)) {
    Serial.println("STA Failed to configure");
    return false;
  }
  WiFi.begin(ssid.c_str(), pass.c_str());
  Serial.println("Connecting to WiFi...");

  while (WiFi.status() != WL_CONNECTED) {
    if (millis() - previousMillis >= intervalWifi) {
      Serial.println("Failed to connect.");
      previousMillis = millis();
      return false;
    }
  }

  Serial.println(WiFi.localIP());
  return true;
}
```

reconnect()

Fungsi untuk membuat ESP32 menghubungkan kembali ke WiFi apabila terputus saat program berjalan.

```
// Fungsi untuk menghubungkan kembali ke WiFi ketika terputus
void reconnect() {
  // Loop until we're reconnected
  WiFi.begin(ssid.c_str(), pass.c_str());
  int attempts = 0;
```

```
  while (WiFi.status() != WL_CONNECTED && attempts < 100) {
    Serial.println(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
  } else {
    Serial.println("\nFailed to connect to WiFi. Please check your
credentials");
  }
}
```

drawText()

Fungsi untuk memudahkan dalam menampilkan informasi ke layar OLED

```
// Fungsi untuk menampilkan String ke layar OLED
void drawText(int x, int y, OLEDDISPLAY_TEXT_ALIGNMENT textAlign, String
text, bool clearOled = false)
{
  if (clearOled) {
    display.clear();
  }
  //  display.clear();
  display.setTextAlignment(textAlign);
  display.drawString(x, y, text);
  display.display();
}
```

setup()

Pada fungsi setup(), lakukan inisialisasi serial monitor untuk proses debugging

```
  // Serial port for debugging purposes
  Serial.begin(115200);
```

Inisialisasi sensor, display, qrcode, dan SPIFFS. Juga menampilkan teks “QRScanner” di layar OLED dengan durasi 2 detik.

```
  dhtSensor.begin();
  display.init();
  display.flipScreenVertically();
  display.setFont(ArialMT_Plain_10);

  drawText(64, 22, TEXT_ALIGN_CENTER, "QRScanner", true);
  delay(2000);
  display.clear();
  qrcode.init();
```

```
initSPIFFS();
reader.setup();
reader.begin();
```

Membaca file konfigurasi yang dalam bentuk JSON, memecahnya dan disimpan pada masing masing variabel. Kemudian ditampilkan pada serial monitor.

```
config = readFile(SPIFFS, configPath);

JsonDocument configVar;

deserializeJson(configVar, config);

ssid = configVar["ssid"].as<String>();
pass = configVar["pass"].as<String>();
ip = configVar["ip"].as<String>();
gateway = configVar["gateway"].as<String>();
tbserver = configVar["tbserver"].as<String>();
tbtoken = configVar["tbtoken"].as<String>();
apiaddress = configVar["apiaddress"].as<String>();

Serial.println(ssid);
Serial.println(pass);
Serial.println(ip);
Serial.println(gateway);
Serial.println(apiaddress);
```

Pengaturan routing untuk web server ketika mode WiFi perangkat sebagai station

```
if (initWiFi())
{
  // Routes for web service
  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/index.html", "text/html", false, processor);
  });

  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send_P(200, "text/plain", readDHTTemperature().c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send_P(200, "text/plain", readDHTHumidity().c_str());
  });

  server.on("/config", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/config.html", "text/html");
  });

  server.on("/config", HTTP_POST, [](AsyncWebServerRequest * request)
  {
    int params = request->params();

    // JSON Buffer for temporary data
    JsonDocument tempData;

    for (int i = 0; i < params; i++) {
```

```
server.on("/config", HTTP_POST, [](AsyncWebServerRequest * request)
{
  int params = request->params();

  // JSON Buffer for temporary data
  JsonDocument tempData;

  for (int i = 0; i < params; i++) {
    AsyncWebParameter *p = request->getParam(i);
    if (p->isPost()) {
      // HTTP POST ssid value
      if (p->name() == AP_SSID) {
        ssid = p->value().c_str();
        Serial.print("SSID set to: ");
        Serial.println(ssid);
        // Add data to json buffer
        tempData["ssid"] = ssid;
      }
      // HTTP POST pass value
      if (p->name() == AP_PASS) {
        pass = p->value().c_str();
        Serial.print("Password set to: ");
        Serial.println(pass);
        // Add data to json buffer
        tempData["pass"] = pass;
      }
      // HTTP POST ip value
      if (p->name() == AP_IP) {
        ip = p->value().c_str();
        Serial.print("IP Address set to: ");
        Serial.println(ip);
        // Add data to json buffer
        tempData["ip"] = ip;
      }
```

```
      // HTTP POST ip value
      if (p->name() == AP_IP) {
        ip = p->value().c_str();
        Serial.print("IP Address set to: ");
        Serial.println(ip);
        // Add data to json buffer
        tempData["ip"] = ip;
      }
      // HTTP POST gateway value
      if (p->name() == AP_GATEWAY) {
        gateway = p->value().c_str();
        Serial.print("Gateway set to: ");
        Serial.println(gateway);
        // Add data to json buffer
        tempData["gateway"] = gateway;
      }
      // HTTP POST tbserver value
      if (p->name() == TB_SERVER) {
        tbserver = p->value().c_str();
        Serial.print("Thingsboard Server set to: ");
        Serial.println(tbserver);
        // Add data to json buffer
        tempData["tbserver"] = tbserver;
      }
      // HTTP POST tbtoken value
      if (p->name() == TB_TOKEN) {
        tbtoken = p->value().c_str();
        Serial.print("Token Device set to: ");
        Serial.println(tbtoken);
        // Add data to json buffer
        tempData["tbtoken"] = tbtoken;
      }
      // HTTP POST api address value
      if (p->name() == API_ADDRESS) {
```

```
      }
      // HTTP POST api address value
      if (p->name() == API_ADDRESS) {
        apiaddress = p->value().c_str();
        Serial.print("API Address set to: ");
        Serial.println(apiaddress);
        // Add data to json buffer
        tempData["apiaddress"] = apiaddress;
      }
    }
    char tempConfig[200];
    serializeJson(tempData, tempConfig);
    writeFile(SPIFFS, configPath, tempConfig);
  }

  request->send(200, "text/plain", "DONE!!!. ESP akan restart, koneksikan ke router " + ssid + " dan jalankan pada browser IP Address: "
  delay(3000);
  ESP.restart();
});
```

Disini ketika user mengakses halaman konfigurasi dan melakukan setting, setelah user menyimpan konfigurasi, data nya akan ditulis ke dalam file config menggunakan SPIFFS. Setelah itu perangkat akan memulai ulang dan konfigurasinya akan diterapkan.

Pengaturan routing untuk web server ketika user ingin membuka halaman Generate QR.

```
  server.on("/qr", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/qrgenerator.html", "text/html");
  });

  server.on("/qr", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/qrgenerator.html", "text/html");
  });

  server.on("/qr", HTTP_POST, [](AsyncWebServerRequest * request)
  {
    int params = request->params();
    for (int i = 0; i < params; i++) {
      AsyncWebParameter *p = request->getParam(i);
      if (p->isPost()) {
        // HTTP POST text value for qr
        if (p->name() == QR_TEXT) {
          qrText = p->value().c_str();
        }
      }
    }
    prevMillisQR = millis();
    qrGenerate = true;
    request->send(SPIFFS, "/qrgenerator.html", "text/html");
  });

  server.begin();
}
```

Ketika perangkat menerima request POST dari user untuk melakukan generate QR, data text tersebut disimpan dalam variabel qrText, dan nilai dari variabel qrGenerate berubah menjadi true yang kemudian akan men trigger perangkat untuk menampilkan gambar QR nya ke layar OLED.

Pengaturan routing untuk web server ketika mode WiFi perangkat sebagai AP hampir sama ketika perangkat dalam mode Station

```
else
{
  // Mode AP
  Serial.println("Setting AP (Access Point)");

  WiFi.softAP("WIFI ESP32", NULL); // → tanpa password

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/index.html", "text/html", false, processor);
  });

  server.on("/config", HTTP_GET, [](AsyncWebServerRequest * request)
  {
    request->send(SPIFFS, "/config.html", "text/html");
  });

  server.on("/config", HTTP_POST, [](AsyncWebServerRequest * request)
  {
    // JSON Buffer for temporary data
    JsonDocument tempData;

    int params = request->params();
    for (int i = 0; i < params; i++) {
```

```
server.on("/config", HTTP_POST, [](AsyncWebServerRequest * request)
{
  int params = request->params();

  // JSON Buffer for temporary data
  JsonDocument tempData;

  for (int i = 0; i < params; i++) {
    AsyncWebParameter *p = request->getParam(i);
    if (p->isPost()) {
      // HTTP POST ssid value
      if (p->name() == AP_SSID) {
        ssid = p->value().c_str();
        Serial.print("SSID set to: ");
        Serial.println(ssid);
        // Add data to json buffer
        tempData["ssid"] = ssid;
      }
      // HTTP POST pass value
      if (p->name() == AP_PASS) {
        pass = p->value().c_str();
        Serial.print("Password set to: ");
        Serial.println(pass);
        // Add data to json buffer
        tempData["pass"] = pass;
      }
      // HTTP POST ip value
      if (p->name() == AP_IP) {
        ip = p->value().c_str();
        Serial.print("IP Address set to: ");
        Serial.println(ip);
        // Add data to json buffer
        tempData["ip"] = ip;
      }
      // HTTP POST gateway value
      if (p->name() == AP_GATEWAY) {
```

```
      // HTTP POST gateway value
      if (p->name() == AP_GATEWAY) {
        gateway = p->value().c_str();
        Serial.print("Gateway set to: ");
        Serial.println(gateway);
        // Add data to json buffer
        tempData["gateway"] = gateway;
      }
      // HTTP POST tbserver value
      if (p->name() == TB_SERVER) {
        tbserver = p->value().c_str();
        Serial.print("Thingsboard Server set to: ");
        Serial.println(tbserver);
        // Add data to json buffer
        tempData["tbserver"] = tbserver;
      }
      // HTTP POST tbtoken value
      if (p->name() == TB_TOKEN) {
        tbtoken = p->value().c_str();
        Serial.print("Token Device set to: ");
        Serial.println(tbtoken);
        // Add data to json buffer
        tempData["tbtoken"] = tbtoken;
      }
      // HTTP POST api address value
      if (p->name() == API_ADDRESS) {
        apiaddress = p->value().c_str();
        Serial.print("API Address set to: ");
        Serial.println(apiaddress);
        // Add data to json buffer
        tempData["apiaddress"] = apiaddress;
      }
    }
    char tempConfig[200];
    serializeJson(tempData, tempConfig);
    writeFile(SPIFFS, configPath, tempConfig);
  }
```

```
    }
    char tempConfig[200];
    serializeJson(tempData, tempConfig);
    writeFile(SPIFFS, configPath, tempConfig);
  }
  request->send(200, "text/plain", "DONE!!!. ESP akan restart, koneksikan ke router " + ssid + " dan jalankan pada browser IP Address: "
  delay(3000);
  ESP.restart();
});

server.on("/qr", HTTP_GET, [](AsyncWebServerRequest * request)
{
  request->send(SPIFFS, "/qrgenerator.html", "text/html");
});

server.on("/qr", HTTP_POST, [](AsyncWebServerRequest * request)
{
  int params = request->params();
  for (int i = 0; i < params; i++) {
    AsyncWebParameter *p = request->getParam(i);
    if (p->isPost()) {
      // HTTP POST text value for qr
      if (p->name() == QR_TEXT) {
        qrText = p->value().c_str();
      }
    }
  }
  prevMillisQR = millis();
  qrGenerate = true;
  request->send(SPIFFS, "/qrgenerator.html", "text/html");
});

server.begin();
}
}
```

Baik dalam mode AP ataupun Station, setelah konfigurasi disimpan, perangkat akan melakukan mulai ulang dan konfigurasi akan diterapkan.

Ketika perangkat menerima request POST dari user untuk melakukan generate QR, data text tersebut disimpan dalam variabel qrText, dan nilai dari variabel qrGenerate berubah menjadi true yang kemudian akan men trigger perangkat untuk menampilkan gambar QR nya ke layar OLED.

loop()

Pada fungsi loop(), pertama adalah memeriksa status koneksi WiFi, apabila tidak terkoneksi maka fungsi reconnect() akan dipanggil kembali dengan durasi waktu sebesar intervalWifi (10 detik).

```
if (WiFi.status() != WL_CONNECTED && ssid)
 {
   wifiStatus = "Disconnected";
   if (millis() - previousMillis >= intervalWifi)
   {
     reconnect();
     previousMillis = millis();
   }
 }
```

Kemudian membaca data dari sensor DHT dan disimpan ke dalam variabel

```
 // Read temperature as Celsius (the default)
 String t = readDHTTemperature();
 String h = readDHTHumidity();
```

Data tersebut kemudian digabungkan ke dalam satu variabel dengan format JSON dan dikirimkan ke server thingsboard dengan fungsi `sendTelemetryJson()` tiap interval send_delay (1 detik) hanya jika perangkat terkoneksi ke server thingsboard.

```
// Check if it is a time to send DHT11 temperature and humidity
 if (millis() - millis_counter > send_delay)
 {
   if (!tb.connected())
   {
     // Connect to the ThingsBoard
     Serial.print("Connecting to: ");
     Serial.print(tbserver);
     Serial.print(" with token ");
     Serial.println(tbtoken);

     if (!tb.connect(tbserver.c_str(), tbtoken.c_str(), TB_PORT))
     {
       Serial.println("Failed to connect");
     }
   }
   else
   { // Kirim telemetry jika terkoneksi ke server
     wifiStatus = "Connected";
     Serial.println("Sending data...");

     // JSON Buffer for Temperature
     JsonDocument tempData;
     tempData["temperature"] = t.toFloat();
```

```
      tempData["humidity"] = h.toFloat();

      char jsonData[200];
      serializeJson(tempData, jsonData);

      Serial.print("Temperature:");
      Serial.print(t);
      Serial.print(" Humidity ");
      Serial.println(h);

      // Sending telemetry to Thingsboard server
      tb.sendTelemetryJson(jsonData);

      HTTPClient http;
      http.begin(wifiClient, apiaddress);

      http.addHeader("Content-Type", "application/json");
      int httpResponseCode = http.POST(jsonData);

      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
    }
    millis_counter = millis(); // reset millis counter
  }
```

Secara spesifik bagian ini digunakan untuk melakukan POST ke API. Parameter yang dikirimkan berbentuk JSON. Kemudian kode respon dari server akan ditampilkan ke Serial monitor untuk debugging.

```
      HTTPClient http;
      http.begin(wifiClient, apiaddress);

      http.addHeader("Content-Type", "application/json");
      int httpResponseCode = http.POST(jsonData);

      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
```

Bagian ini berfungsi untuk memeriksa apakah perangkat sedang melakukan scanning QR. Jika proses scan nya berhasil dan datanya valid, hasilnya akan ditampilkan ke layar OLED dan Serial monitor. Apabila hasil scannya tidak valid akan muncul informasi di Serial monitor tetapi tidak menampilkan apapun di layar OLED.

```
struct QRCodeData qrCodeData;
  if (reader.receiveQrCode(&qrCodeData, 100))
  {
```

```
    Serial.println("Found QRCode");
    if (qrCodeData.valid)
    {
      qrCodePayload = (const char *)qrCodeData.payload;
      Serial.print("Payload: ");
      Serial.println((const char *)qrCodeData.payload);

      drawText(64, 22, TEXT_ALIGN_CENTER, "Scan berhasil", true);
      display.setTextAlignment(TEXT_ALIGN_CENTER);
      display.drawStringMaxWidth(64, 30, 128, qrCodePayload);
      display.display();
      prevMillis = millis();
    }
    else
    {
      Serial.print("Invalid: ");
      Serial.println((const char *)qrCodeData.payload);
    }
  }
```

Bagian kode ini memeriksa apakah ada request untuk melakukan generate kode QR dari web server. Jika ada hasil generate QR tersebut akan ditampilkan di layar OLED selama 10 detik.

```
  if (qrGenerate)
  {
    qrcode.create(qrText);
    while (true)
    {
      // Interval untuk menampilkan kode QR adalah 10 detik (10.000 ms)
      if (millis() - prevMillisQR > 10000)
      {
        prevMillisQR = millis();
        qrGenerate = false;
        break;
      }
    }
  }
```

Bagian ini digunakan untuk menampilkan informasi pada layar OLED sekaligus melakukan refresh layar tiap interval 3 detik

```
  // Reset layar OLED sesuai dengan nilai interval 3 detik
  if (millis() - prevMillis > intervalQr)
  {
    prevMillis = millis();
    display.clear();
```

```
  drawText(5, 0, TEXT_ALIGN_LEFT, "T : " + t);
  drawText(115, 0, TEXT_ALIGN_RIGHT, "H : " + h);
  drawText(64, 22, TEXT_ALIGN_CENTER, "Arahkan QR ke kamera!!!");
  drawText(5, 52, TEXT_ALIGN_LEFT, "Status : " + wifiStatus);
}
```

Full Code

https://github.com/febriansf/ESP32-QR-Scanner-DHT11/blob/main/QR-DHT.ino

Cara menggunakan device:

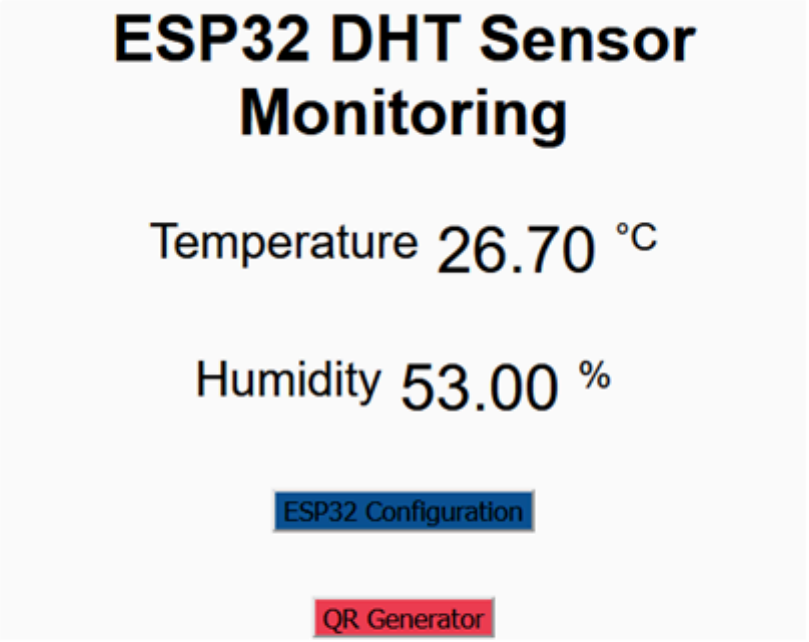
Mengkonfigurasi esp32 untuk koneksi ke router:

1. Pertama, sambungkan device dengan PC atau laptop melalui kabel USB

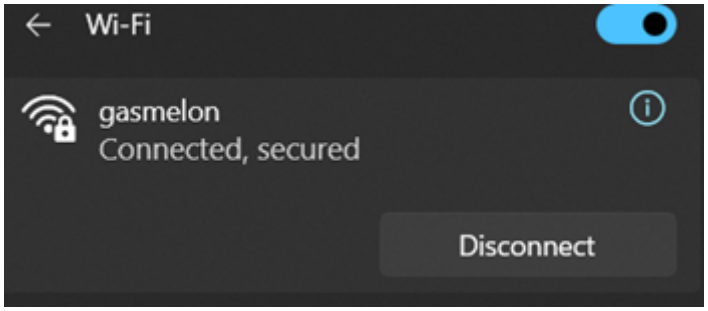
2. Setelah tersambung,tekan tombol reset (EN/RST) pada mikrokontroler ESP32 Wrover + Cam untuk menyalakan sistem dan tampilan pada OLED SSD1306

3. Kemudian sambungkan wifi pada laptop ke SSID “WIFI ESP32”

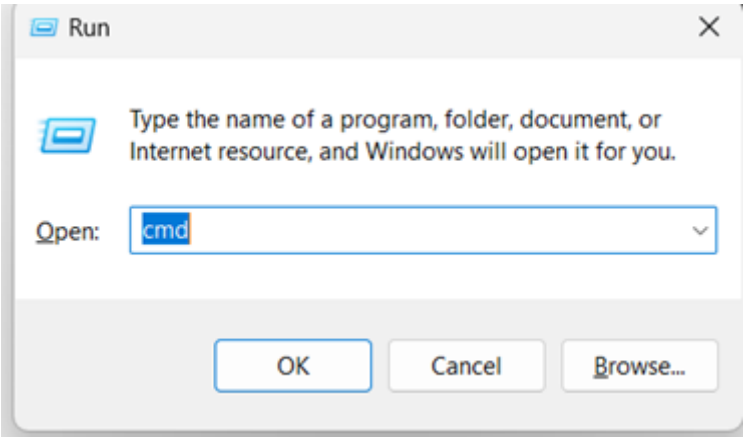
4. Buka browser apapun (Pada tutorial ini menggunakan Mozilla Firefox), lalu ketikkan 192.168.4.1 pada *search bar* hingga muncul tampilan seperti di bawah:



5. Selanjutnya, untuk melakukan koneksi dari ESP32 ke router kita harus mengecek IP router dan gatewaynya terlebih dahulu dengan menyambung ke SSID router target (dalam tutorial ini akan disambungkan ke SSID “gasmelon” dengan password “padatcair”)



6. Kemudian masuk ke command prompt dengan menekan WINDOWS + R, lalu ketikkan cmd lalu tekan ENTER

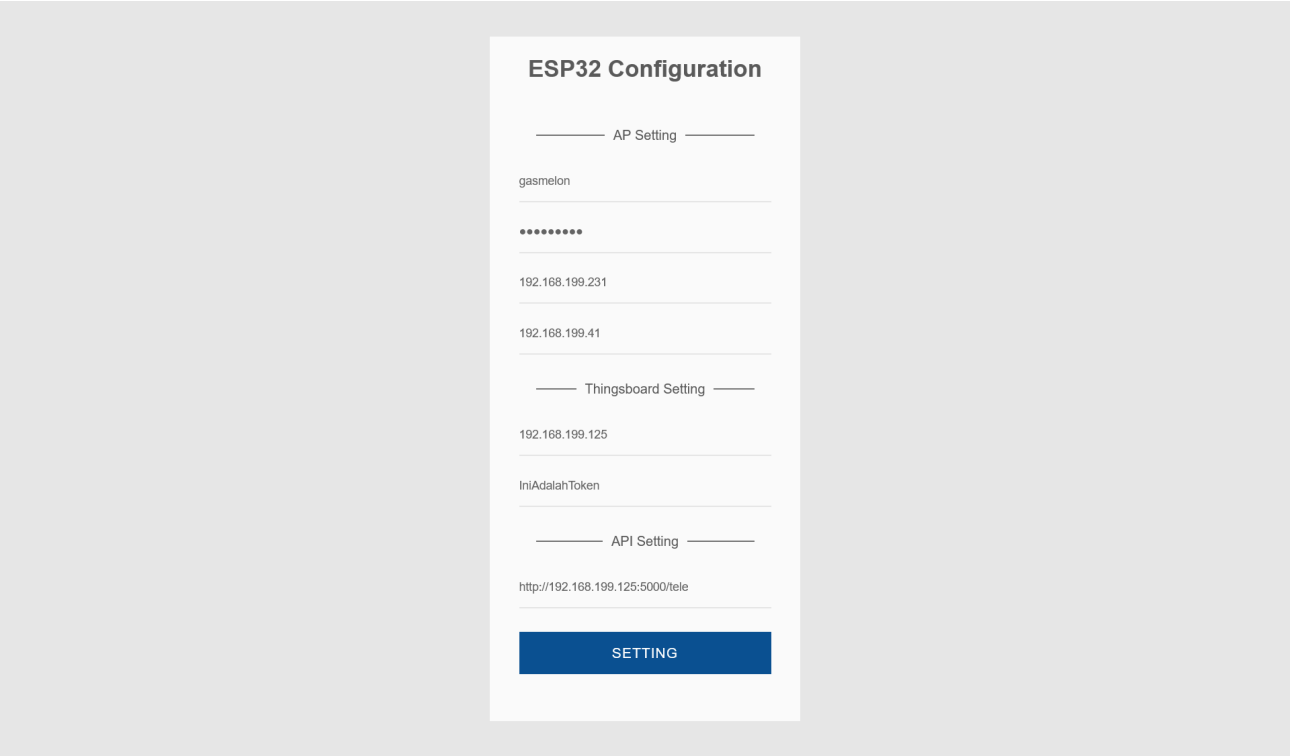


7. Pada command prompt ketikkan ipconfig /all untuk melihat IP dan gateway router tujuan

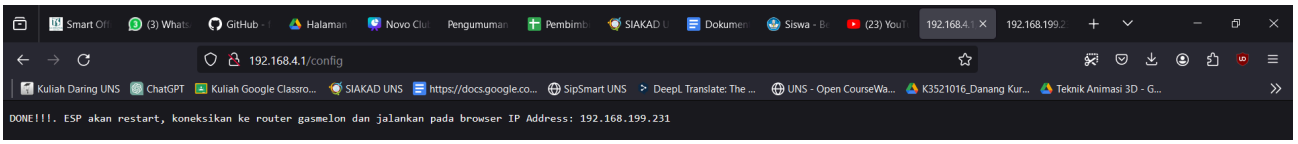
```
C:\WINDOWS\system32\cmd.

Wireless LAN adapter Wi-Fi 3:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
   Physical Address. . . . . . . . . : D8-80-83-76-87-0F
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::9423:7647:31d2:8081%8(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.199.231(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Thursday, 01 February 2024 10:14:43
   Lease Expires . . . . . . . . . . : Thursday, 01 February 2024 11:14:42
   Default Gateway . . . . . . . . . : 192.168.199.41
   DHCP Server . . . . . . . . . . . : 192.168.199.41
   DHCPv6 IAID . . . . . . . . . . . : 752386179
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2C-D7-BF-6C-E4-A8-DF-CA-7F-29
   DNS Servers . . . . . . . . . . . : 192.168.199.41
   NetBIOS over Tcpip. . . . . . . . : Enabled

Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Realtek PCIe GbE Family Controller
   Physical Address. . . . . . . . . : E4-A8-DF-CA-7F-29
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

C:\Users\Lenovo>
```

8. Setelah mendapatkan ip dan gateway router sambungkan lagi wifi pada laptop ke SSID “WIFI ESP32”

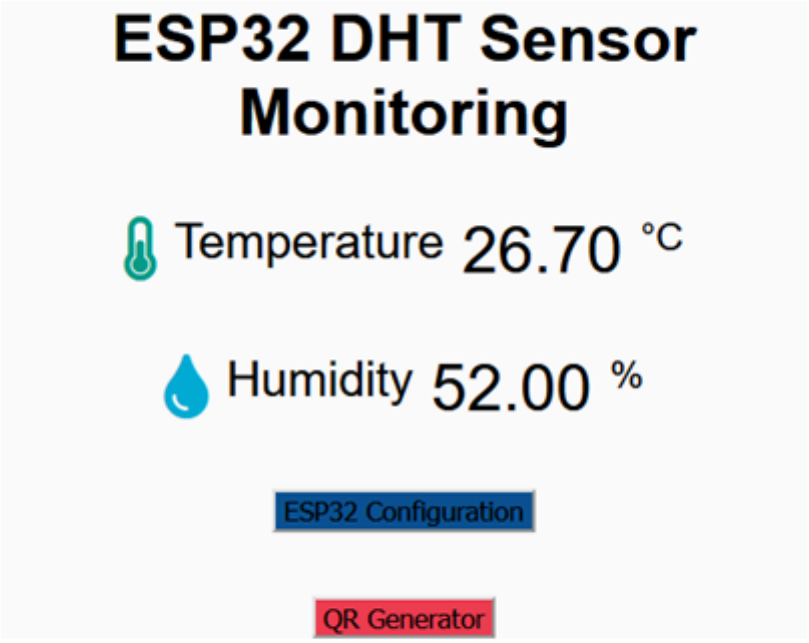
9. Kembali ke browser lalu tekan tombol esp configuration, lalu mending isi SSID, password, alamat IP untuk ESP32, alamat IP gateway, alamat IP server thingsboard, dan token thingsboard seperti gambar di bawah, lalu klik tombol SETTING:



10. Setelah tombol SETTING di klik, akan muncul tulisan pada layer browser seperti gambar di bawah:



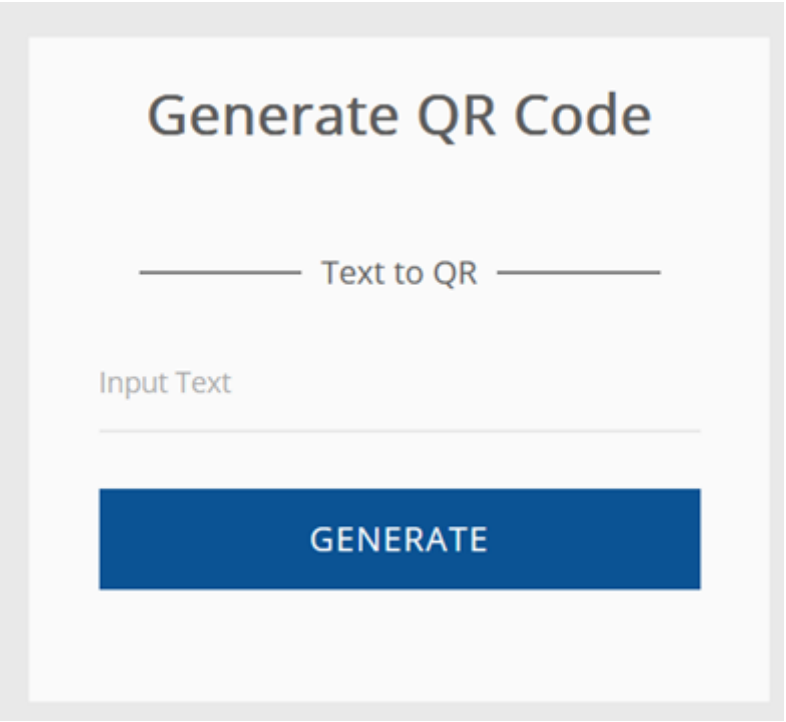
11. Lalu sambungkan wifi ke SSID “gasmelon”, lalu ketikkan ip pada browser sesuai dengan langkah ke-10 sehingga muncul seperti gambar di bawah:



Menampilkan QRCode pada layar OLED:

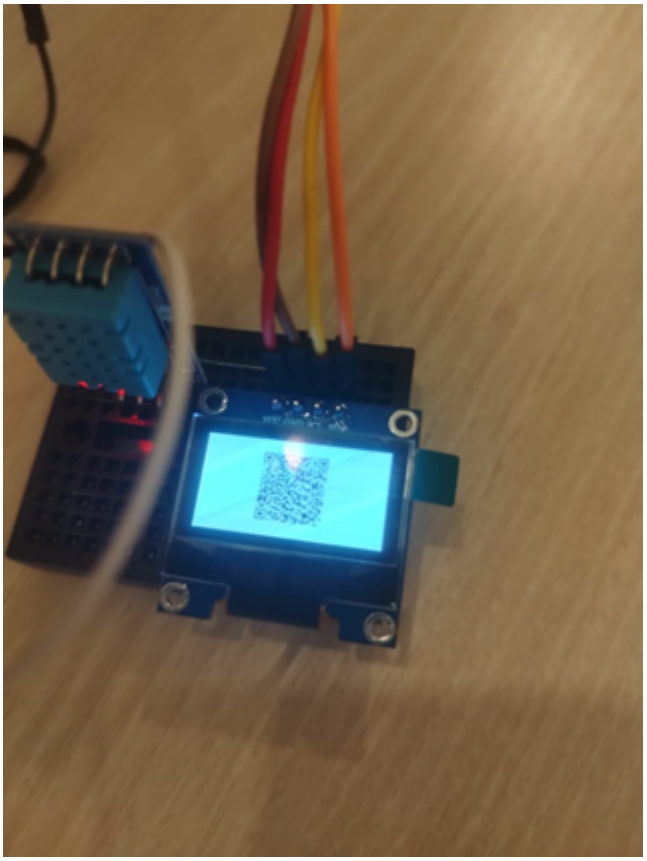
1. Sekarang konfigurasi esp menjadi AP sudah selesai jika ingin mengubah koneksi ke router yang berbeda ulangi langkah ke-1 hingga langkah ke-11

2. Kemudian untuk menampilkan qrcode pada layer OLED SSD1306, tekan tombol QR Generator sehingga muncul tampilan pada browser seperti di bawah:



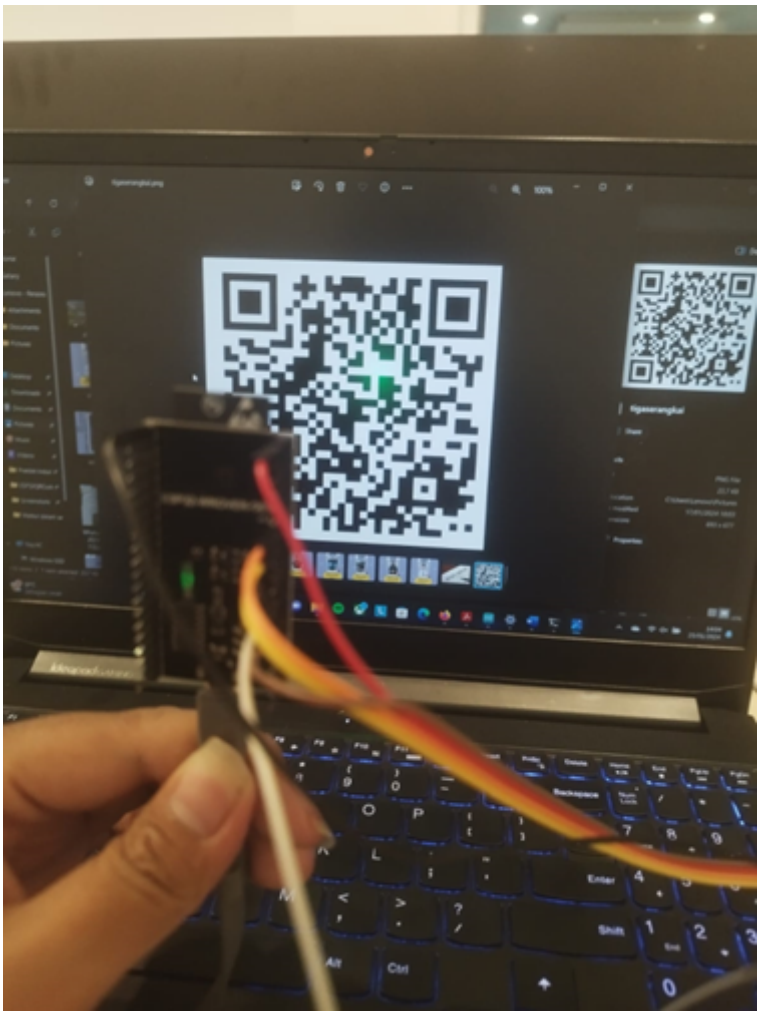
3. Ketikkan apa saja yang ingin dijadikan QR Code (pada tutorial ini diketikkan "https://tigaserangkai.com"), lalu klik tombol GENERATE

4. Berikut tampilan QRCODE pada OLED SSD1306

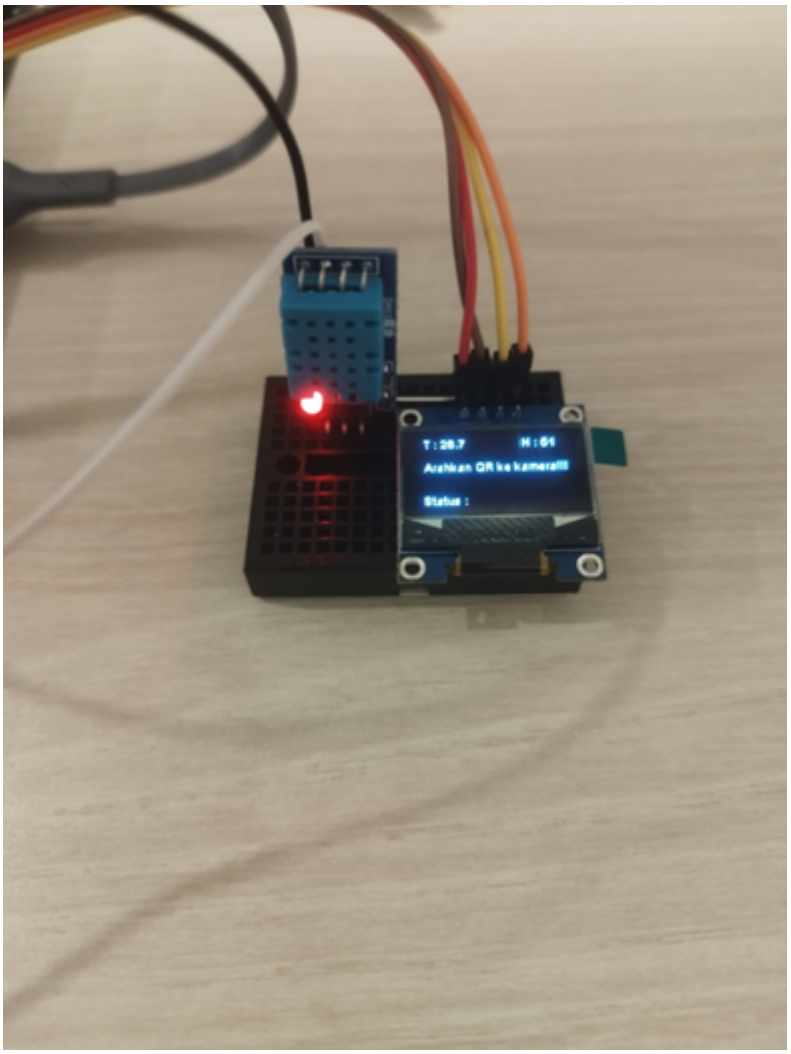


Melakukan Scan QR Code menggunakan ESP32 WROVER + CAM:

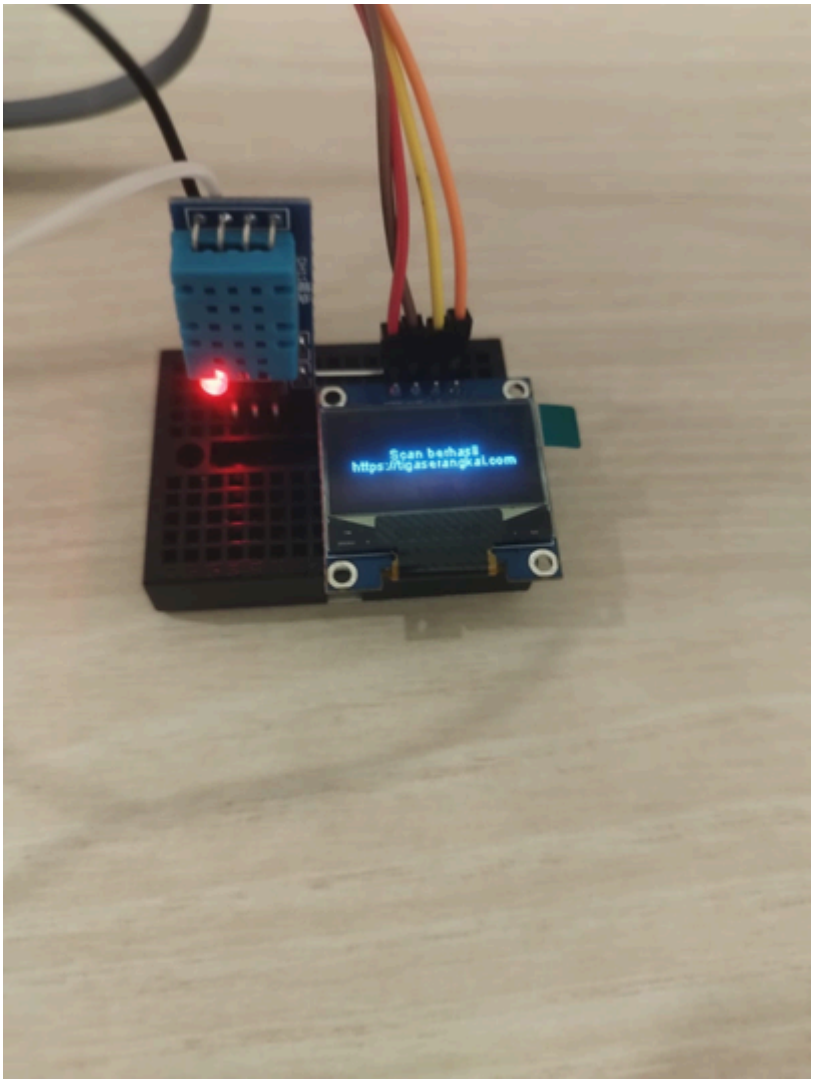
1. Pertama siapkan QRCode yang ingin di scan usahakan ukurannya cukup besar agar mudah di scan, lalu arahkan ESP32 WROVER + CAM ke QRCode tujuan seperti gambar di bawah:



2. Kemudian akan muncul tampilan pada OLED SSD1306 seperti gambar di bawah:
    - Sebelum scan qrcode:



- Setelah scan qrcode:



Konfigurasi API

Installing FeathersJS

1. Install terlebih dahulu NodeJS, jika sudah terinstall, cek di terminal dengan command :

```
node --version
npm --version
```

2. Install TypeScript, NodeJS wrapper, dan berbagai macam package

```
npm init --yes
npm i typescript ts-node @types/node --save-dev
npx tsc --init --target es2020
```

3. Install Feathers

```
npm install @feathersjs/feathers --save
```

4. Install Socket.io memiliki kegunaan seperti WebSockeet dan KoaJS agar service kita menjadi REST API secara otomatis

```
npm install @feathersjs/socketio @feathersjs/koa --save
```

5. Module yang digunakan

```
npm install @feathersjs/adapter-commons --save
npm install @feathersjs/authentication --save
npm install @feathersjs/authentication-client --save
npm install @feathersjs/configuration --save
npm install @feathersjs/errors --save
npm install @feathersjs/knex --save
npm install @feathersjs/koa --save
npm install @feathersjs/schema --save
npm install @feathersjs/transport-commons --save
npm install @feathersjs/typebox --save
npm i knex
npm i koa-router
npm install --save-dev @types/node-telegram-bot-api
npm install pg

npm i cli
npm install @feathersjs/rest-client --save
npm install axios
npm i cross-env
npm i mocha
npm i nodemon
npm i prettier
```

Membuat Project

1. Di terminal, lakukan command

```
npm create feathers@latest <nama>
```

2. Sesuaikan pilihan dengan apa yang dibutuhkan atau pilih saja yang direkomendasi, contoh pilihan :

```
? Do you want to use JavaScript or TypeScript? JavaScript
? Write a short description
? Which HTTP framework do you want to use? KoaJS (recommended)
? What APIs do you want to offer? HTTP (REST), Real-time
? Which package manager are you using? npm
? Generate client? Can be used with React, Angular, Vue, React Native, Node.js etc. Yes
? What is your preferred schema (model) definition format? Schemas allow to type, validate, secure and populate your data and configuration TypeBox (recommended)
? Which database are you connecting to? Databases can be added at any time PostgreSQL
? Enter your database connection string postgres://postgres:@localhost:5432/newserv
   Wrote file test\app.test.js
   Wrote file src\app.js
   Wrote file src\channels.js
   Wrote file test\client.test.js
   Wrote file src\client.js
   Wrote file config\default.json
   Wrote file config\test.json
   Wrote file config\custom-environment-variables.json
   Wrote file src\configuration.js
   Wrote file public\index.html
   Wrote file src\index.js
   Wrote file src\logger.js
   Wrote file src\hooks\log-error.js
   Wrote file package.json
   Wrote file .prettierrc
   Wrote file readme.md
   Wrote file src\services\index.js
   Wrote file src\validators.js
   Wrote file src\postgresql.js
   Wrote file knexfile.js
   Updated src\app.js
   Running npm install @feathersjs/knex@^5.0.21 knex@^3.1.0 pg@^8.11.3 @feathersjs/feathers@^5.0.21 @feathersjs/errors@^5.0.21 @feathersjs/schema@^5.0.21 @feathersj
```

```
   Running npm install @feathersjs/knex@^5.0.21 knex@^3.1.0 pg@^8.11.3 @feathersjs/feathers@^5.0.21 @feathersjs/errors@^5.0.21 @feathersjs/schema@^5.0.21 @feathersj
s/configuration@^5.0.21 @feathersjs/transport-commons@^5.0.21 @feathersjs/adapter-commons@^5.0.21 @feathersjs/authentication@^5.0.21 @feathersjs/authentication-clien
t@^5.0.21 winston@latest @feathersjs/socketio@^5.0.21 @feathersjs/koa@^5.0.21 @feathersjs/typebox@^5.0.21

added 232 packages, and audited 233 packages in 1m

33 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
   Running npm install nodemon@latest axios@^1.6.6 mocha@^10.2.0 cross-env@latest prettier@latest @feathersjs/cli@5.0.21 @feathersjs/rest-client@^5.0.21 --save-dev

added 145 packages, and audited 378 packages in 55s

72 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


Hooray! Your Feathers app is ready to go! 🚀
Go to the newserv folder to get started.
```

Membuat Service baru

1. Untuk membuat service dengan database, gunakan command

```
npx feathers generate service
```

2. Berikan nama service baru, untuk pathnya tekan saja Enter agar diberikan secara otomatis, untuk authentication, database, dan schema, pilih sesuai keperluan

```
? What is the name of your service? botTele
? Which path should the service be registered on? bot-tele
? Does this service require authentication? No
? What database is the service using? SQL
? Which schema definition format do you want to use? Schemas allow to type, validate, secure and populate data TypeBox  (recommended)
   Updated src\client.js
```

Penjelasan Kode

Menyimpan schema dan mendefinisikan struktur objek dalam schema

```
export const teleSchema = Type.Object(
  {
    id: Type.Number(),
    temperature: Type.Number(),
    humidity: Type.Number()
  },
  { $id: 'Tele', additionalProperties: false }
)
```

Validator, untuk memvalidasi data agar sesuai dengan ketentuan teleSchema

```
export const teleValidator = getValidator(teleSchema, dataValidator)
```

Resolver data + external resolver

```
export const teleResolver = resolve({})
export const teleExternalResolver = resolve({})
```

Menyambungkan bot Telegram dengan menggunakan Token yang diberikan
Polling agar bot selalu mendapatkan update
Membuat bot telegram baru
Mendeklarasikan variabel global yang digunakan untuk menyimpan pesan

```
import TelegramBot from 'node-telegram-bot-api';
const token = 'token bot telegram';
const options = {
  polling: true
};
const bot = new TelegramBot(token, options);
let globalMessage;
```

Event Handler agar bot dapat mengirimkan pesan berisi temperature dan humidity ketika diberikan perintah /suhuruangan di telegram

```
bot.onText(/\/suhuruangan/, function onGetData(msg) {
  console.log('Received /suhuruangan command.');

  if (msg.chat && msg.chat.id) {
    if (globalMessage && globalMessage.trim() !== '') {
      bot.sendMessage(msg.chat.id, globalMessage)
        .then(sentMessage => {
          console.log('Message sent successfully:', sentMessage);
        })
        .catch(error => {
          console.error('Error sending message:', error);
        });
    } else {
      console.warn('globalMessage is empty. Ensure temperature and humidity have values.');
    }
  } else {
    console.warn('Invalid chat ID. Make sure msg.chat and msg.chat.id are defined.');
  }
});
```

Middleware untuk membuat entity baru menangani POST request ke /api/send-data

```
// Middleware
app.use(async (ctx, next) => {
  if (ctx.method === 'POST' && ctx.path === '/api/send-data') {
    try {
      const receivedData = ctx.request.body;
      const result = await global.teleService.create(receivedData);
      ctx.body = { success: true, result };
    } catch (error) {
      ctx.status = 500;
      ctx.body = { success: false, error: error.message };
    }
  } else {
    await next();
  }
});
```

Membuat instance dari router koa dan menentukan route untuk menangani permintaan POST request ke /api/send-data

```
// Define a route to handle POST requests
router.post('/api/send-data', async (ctx) => {
  try {
    const receivedData = ctx.request.body; // Assuming data is sent in the request body
    // Handle the received data, e.g., pass it to the TeleService create method
    const result = await teleService.create(receivedData);
    ctx.body = { success: true, result };
  } catch (error) {
    ctx.status = 500;
    ctx.body = { success: false, error: error.message };
  }
});
```