

Smart Garden IoT Dashboard - Panduan Implementasi

Berikut adalah panduan untuk mengimplementasikan sistem Smart Garden Dashboard yang telah dibuat:

Prasyarat

- Node.js v18.20.5 (sesuai dengan yang telah diinstall di laptop Anda)
- SQLite3
- MQTT Broker (Mosquitto, HiveMQ, atau broker MQTT lainnya)

Langkah-Langkah Instalasi

1. Clone atau Salin Kode Proyek

Download atau salin semua file yang telah dibuat ke dalam folder proyek Anda.

2. Instalasi Dependensi

Buka terminal atau command prompt, navigasi ke folder proyek dan jalankan:

```
bash
```

```
npm install
```

Perintah ini akan menginstal semua dependensi yang diperlukan berdasarkan file `package.json`.

3. Konfigurasi Environment Variables

Sesuaikan file `.env` dengan konfigurasi yang diperlukan:

```
# Server Configuration
PORT=3000 # Port untuk web server
NODE_ENV=development # Ubah ke 'production' pada lingkungan produksi

# Session Secret
SESSION_SECRET=smart-garden-dashboard-secret-key # Ganti dengan string acak untuk keamanan

# MQTT Configuration
MQTT_BROKER=mqtt://localhost:1883 # Alamat MQTT broker Anda
MQTT_CLIENT_ID=smart-garden-dashboard
MQTT_USERNAME= # Isi jika MQTT broker memerlukan autentikasi
MQTT_PASSWORD= # Isi jika MQTT broker memerlukan autentikasi

# Database Configuration
DB_PATH=./data/smart_garden.db
```

4. Inisialisasi Database

Jalankan perintah berikut untuk membuat dan menginisialisasi database:

bash

```
npm run init-db
```

Perintah ini akan membuat database SQLite dan membuat user admin default dengan:

- Username: `admin`
- Password: `admin123`

5. Menjalankan Aplikasi

Untuk menjalankan aplikasi dalam mode development dengan auto-reload:

bash

```
npm run dev
```

Untuk menjalankan aplikasi dalam mode produksi:

bash

```
npm start
```

Aplikasi akan berjalan di `http://localhost:3000` (atau port yang dikonfigurasi di `.env`).

Konfigurasi MQTT dengan SSL/TLS

Untuk menghubungkan ke broker MQTT yang menggunakan SSL/TLS (seperti HiveMQ Cloud), ikuti langkah-langkah berikut:

1. Persiapkan Sertifikat CA

Saya telah menyediakan script `utils/prepare-certs.js` untuk mempersiapkan sertifikat CA:

bash

```
node utils/prepare-certs.js
```

Script ini akan:

- Membuat direktori `data/certs/` jika belum ada
- Mengunduh sertifikat ISRG Root X1 (Let's Encrypt) dari letsencrypt.org dan menyimpannya di lokasi yang ditentukan

2. Atau Upload Sertifikat CA Manual

Jika Anda sudah memiliki file sertifikat CA (seperti `isrgrootx1.pem`), Anda dapat menyalinnya ke lokasi yang dikonfigurasi di `.env`:

```
MQTT_CA_FILE=data/certs/isrgrootx1.pem
```

3. Konfigurasi MQTT dengan SSL/TLS

Edit file `.env` untuk menyesuaikan konfigurasi MQTT dengan broker Anda:

```
MQTT_HOST=3895fc47c772499f9963cb3b380dfb13.s1.eu.hivemq.cloud
MQTT_PORT=8883
MQTT_USERNAME=smart_garden
MQTT_PASSWORD=G44rdeninggg555
MQTT_CLIENT_ID=smart-garden-dashboard
MQTT_USE_SSL=true
MQTT_CA_FILE=data/certs/isrgrootx1.pem
```

Pengaturan yang penting untuk SSL/TLS:

- `MQTT_HOST` - Hostname broker MQTT (tanpa protokol)

- `MQTT_PORT` - Biasanya 8883 untuk MQTT dengan SSL
- `MQTT_USE_SSL` - Set ke `true` untuk mengaktifkan SSL/TLS
- `MQTT_CA_FILE` - Path ke file sertifikat CA

Format Data MQTT

Format data yang diharapkan untuk setiap topik adalah:

Temperature

28.5

(angka desimal dalam satuan °C)

Humidity

65.3

(angka desimal dalam satuan %)

Soil Moisture

45.2

(angka desimal dalam satuan %)

Valve Status dan Control

on

atau

off

Keamanan

- Sistem menggunakan autentikasi username dan password
- Sesi pengguna disimpan di database SQLite
- Password di-hash menggunakan bcrypt

- Pengguna dapat mengubah password setelah login

Penyesuaian Tampilan

Tampilan dapat disesuaikan dengan mengedit file CSS di `public/css/style.css` dan template EJS di folder `views`.

Implementasi pada Server Windows

Untuk menjalankan aplikasi sebagai layanan Windows:

1. Install modul `node-windows` secara global:

bash

```
npm install -g node-windows
```

2. Buat script untuk mendaftarkan aplikasi sebagai layanan Windows (`install-service.js`):

javascript

```
const Service = require('node-windows').Service;
const path = require('path');

const svc = new Service({
  name: 'Smart Garden Dashboard',
  description: 'Dashboard IoT untuk Smart Garden',
  script: path.join(__dirname, 'app.js')
});

svc.on('install', function() {
  svc.start();
  console.log('Layanan terpasang dan berjalan.');
});

svc.install();
```

3. Jalankan script untuk memasang layanan:

bash

```
node install-service.js
```

Menghubungkan dengan Mikrokontroler

Arduino atau ESP8266/ESP32 dapat dikonfigurasi untuk mengirim data ke MQTT broker dengan library seperti PubSubClient (Arduino) atau ESPMQTTClient (ESP8266/ESP32).

Contoh kode untuk ESP8266 dengan DHT11 dan sensor soil moisture:

cpp

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

#define DHTPIN 2
#define DHTTYPE DHT11
#define SOIL_MOISTURE_PIN A0
#define VALVE_PIN 4

const char* ssid = "WiFi_SSID";
const char* password = "WiFi_PASSWORD";
const char* mqtt_server = "MQTT_BROKER_IP";

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    . pinMode(VALVE_PIN, OUTPUT);
    . digitalWrite(VALVE_PIN, LOW);
    .
    dht.begin();
    .
    WiFi.begin(ssid, password);
    .
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    .
    // Baca dan publish data sensor setiap 5 detik
    static unsigned long lastPublish = 0;
    if (millis() - lastPublish > 5000) {
        lastPublish = millis();
        .
        // Baca suhu dan kelembaban
        float temp = dht.readTemperature();
        float humidity = dht.readHumidity();
```

```

// Baca kelembaban tanah
int soilRaw = analogRead(SOIL_MOISTURE_PIN);
float soilMoisture = map(soilRaw, 1023, 0, 0, 100);

// Publish data
if (!isnan(temp)) {
    client.publish("smart-garden/sensors/temperature", String(temp).c_str());
}

if (!isnan(humidity)) {
    client.publish("smart-garden/sensors/humidity", String(humidity).c_str());
}

client.publish("smart-garden/sensors/soil-moisture", String(soilMoisture).c_str());

// Publish status valve
client.publish("smart-garden/status/valve", digitalRead(VALVE_PIN) == HIGH ? "on" : "off");
}

}

// Callback untuk perintah kontrol
void callback(char* topic, byte* payload, unsigned int length) {
    String payloadStr = "";
    for (int i = 0; i < length; i++) {
        payloadStr += (char)payload[i];
    }

    if (String(topic) == "smart-garden/control/valve") {
        if (payloadStr == "on") {
            digitalWrite(VALVE_PIN, HIGH);
            client.publish("smart-garden/status/valve", "on");
        } else if (payloadStr == "off") {
            digitalWrite(VALVE_PIN, LOW);
            client.publish("smart-garden/status/valve", "off");
        }
    }
}

void reconnect() {
    while (!client.connected()) {
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);

```

```
... if (client.connect(clientId.c_str())) {
    client.subscribe("smart-garden/control/valve");
} else {
    delay(5000);
}
}
```

Informasi Tambahan

- Username dan password default: admin / admin123
- Ganti password default setelah login pertama kali
- Database akan disimpan di folder `data/` dalam proyek
- Log aktivitas perangkat tersimpan di database

Catatan Penting

- Pastikan Node.js dan NPM telah terinstal dengan benar
- Pastikan MQTT broker berjalan dan dapat diakses
- Cadangkan database secara berkala untuk menghindari kehilangan data