**Kiến trúc hệ thống**

Hệ thống sẽ bao gồm các thành phần chính sau:

1. **Nanomq Broker**: Là một MQTT broker nhẹ, chịu trách nhiệm quản lý các topic và luồng dữ liệu giữa các client (publisher/subscriber).
2. **Backend Python**: Một ứng dụng Python sử dụng giao thức MQTT để subscribe vào topic từ Nanomq broker, nhận dữ liệu và xử lý chúng.
3. **WebSocket**: Kết nối thời gian thực giữa backend Python và frontend để truyền dữ liệu từ MQTT đến UI.
4. **Frontend (UI)**: Một giao diện web đơn giản (dùng HTML/CSS/JavaScript) hiển thị thông tin nhận được từ backend qua WebSocket.

**Luồng hoạt động**

* Một thiết bị hoặc ứng dụng (MQTT publisher) gửi dữ liệu đến một topic trên Nanomq broker.
* Backend Python subscribe vào topic đó, nhận dữ liệu từ broker.
* Backend gửi dữ liệu qua WebSocket đến frontend.
* Frontend hiển thị dữ liệu trên giao diện người dùng theo thời gian thực.

**Cách thực hiện**

**1. Chuẩn bị môi trường**

* **Cài đặt Nanomq**: Tải và cài đặt Nanomq broker từ trang chính thức (<https://nanomq.io/>). Chạy broker trên máy local hoặc server với cấu hình mặc định (port 1883).
* **Cài đặt Python**: Đảm bảo Python 3.x được cài đặt.
* **Thư viện cần thiết**:
  + paho-mqtt: Để kết nối MQTT trong Python.
  + websockets: Để tạo WebSocket server.
  + aiohttp: (Tùy chọn) Nếu muốn tích hợp web server đơn giản cho UI. Cài đặt qua lệnh:

bash

CollapseWrapCopy

pip install paho-mqtt websockets aiohttp

**2. Backend Python**

Tạo file backend.py với nội dung như sau:

python

CollapseWrapCopy

import paho.mqtt.client as mqtt

import asyncio

import websockets

import json

*# Danh sách để lưu trữ các client WebSocket*

connected\_clients = set()

*# Callback khi kết nối tới MQTT broker*

def on\_connect(client, userdata, flags, rc):

print("Connected to MQTT broker with code:", rc)

*# Subscribe vào topic khi kết nối thành công*

client.subscribe("test/topic")

*# Callback khi nhận được message từ topic*

def on\_message(client, userdata, msg):

print(f"Received message: {msg.payload.decode()} on topic {msg.topic}")

*# Gửi message tới tất cả client WebSocket*

if connected\_clients: *# Kiểm tra xem có client nào kết nối không*

asyncio.run(send\_to\_clients(msg.payload.decode()))

*# Hàm gửi dữ liệu qua WebSocket*

async def send\_to\_clients(message):

if connected\_clients:

await asyncio.gather(

\*[client.send(message) for client in connected\_clients]

)

*# WebSocket handler*

async def websocket\_handler(websocket, path):

*# Thêm client mới vào danh sách*

connected\_clients.add(websocket)

try:

*# Giữ kết nối mở*

await websocket.wait\_closed()

finally:

*# Xóa client khi ngắt kết nối*

connected\_clients.remove(websocket)

*# Khởi tạo MQTT client*

mqtt\_client = mqtt.Client()

mqtt\_client.on\_connect = on\_connect

mqtt\_client.on\_message = on\_message

*# Kết nối tới Nanomq broker*

mqtt\_client.connect("localhost", 1883, 60)

*# Chạy MQTT client trong luồng nền*

mqtt\_client.loop\_start()

*# Chạy WebSocket server*

async def main():

websocket\_server = await websockets.serve(websocket\_handler, "localhost", 8765)

print("WebSocket server started on ws://localhost:8765")

await websocket\_server.wait\_closed()

if \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

**Giải thích**:

* Backend sử dụng paho-mqtt để subscribe vào topic test/topic trên Nanomq broker.
* Khi nhận được dữ liệu, backend gửi nó qua WebSocket (port 8765) tới tất cả client kết nối.
* WebSocket server chạy bất đồng bộ bằng websockets.

**3. Frontend (UI)**

Tạo file index.html với nội dung:

html

PreviewCollapseWrapCopy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>MQTT Dashboard</title>

<style>

#messages { border: 1px solid #ccc; padding: 10px; height: 300px; overflow-y: scroll; }

</style>

</head>

<body>

<h1>MQTT Dashboard</h1>

<div id="messages"></div>

<script>

const ws = new WebSocket("ws://localhost:8765");

const messagesDiv = document.getElementById("messages");

ws.onmessage = function(event) {

const message = document.createElement("p");

message.textContent = event.data;

messagesDiv.appendChild(message);

messagesDiv.scrollTop = messagesDiv.scrollHeight; *// Cuộn xuống dòng mới nhất*

};

ws.onopen = function() {

console.log("Connected to WebSocket server");

};

ws.onerror = function(error) {

console.error("WebSocket error:", error);

};

</script>

</body>

</html>

**Giải thích**:

* Frontend kết nối tới WebSocket server tại ws://localhost:8765.
* Khi nhận dữ liệu từ backend, nó hiển thị lên giao diện trong thẻ <div>.

**4. Chạy hệ thống**

1. **Khởi động Nanomq broker**:

bash

CollapseWrapCopy

nanomq start

Đảm bảo broker chạy trên localhost:1883.

1. **Chạy backend**:

bash

CollapseWrapCopy

python backend.py

1. **Mở UI**:
   * Mở file index.html trong trình duyệt (có thể dùng live server hoặc một web server đơn giản như python -m http.server).
2. **Kiểm tra**:
   * Dùng một MQTT client (như mosquitto\_pub) để publish dữ liệu vào topic test/topic:

bash

CollapseWrapCopy

mosquitto\_pub -h localhost -t "test/topic" -m "Hello from MQTT!"

* + Dữ liệu sẽ xuất hiện trên UI trong trình duyệt.