**1. Data Flow (Luồng dữ liệu)**

**Tổng quan hệ thống**

**Hệ thống bao gồm:**

* **Gateways: Thu thập dữ liệu từ các Beacon qua Bluetooth, gửi tới NanoMQ qua MQTT.**
* **Listener: Nhận dữ liệu từ NanoMQ, lưu vào Redis.**
* **Processor: Xử lý dữ liệu từ Redis, xác định trạng thái Beacon, cập nhật vào Redis.**
* **Publisher: Gửi dữ liệu trạng thái từ Redis lên AWS IoT Core.**
* **UI: Hiển thị dữ liệu từ Redis qua Flask UI, đồng thời gửi lệnh điều khiển Gateway qua NanoMQ.**

**Luồng dữ liệu chi tiết**

1. **Gateway → NanoMQ:** 
   * **Gateway phát hiện Beacon, gửi dữ liệu thô (raw data) tới NanoMQ.**
   * **Dữ liệu bao gồm: beacon\_id, rssi, timestamp.**
   * **Topic: bluetooth/<gateway\_id>/data.**
2. **NanoMQ → Listener:** 
   * **listener.py subscribe topic bluetooth/+/data (dùng wildcard + để nhận từ tất cả Gateway).**
   * **Nhận dữ liệu, thêm gateway\_id từ topic, lưu vào Redis:** 
     + **Queue beacon\_data: Dữ liệu thô từ Gateway.**
     + **Queue logs: Log hoạt động của Listener.**
3. **Listener → Processor:** 
   * **processor.py lấy dữ liệu từ queue beacon\_data trong Redis.**
   * **Xử lý dữ liệu:** 
     + **Tính trung bình RSSI và tần suất (frequency) trong khoảng WINDOW\_SIZE.**
     + **Xác định Gateway gần nhất dựa trên score (RSSI + frequency).**
   * **Cập nhật trạng thái vào Redis:** 
     + **Hash beacon\_state: Lưu trạng thái hiện tại của Beacon (gateway\_id, detected, timestamp).**
     + **Queue aws\_queue: Dữ liệu cần gửi lên AWS (khi trạng thái thay đổi).**
4. **Processor → Publisher:** 
   * **publisher.py lấy dữ liệu từ queue aws\_queue trong Redis.**
   * **Gửi dữ liệu trạng thái lên AWS IoT Core qua topic beacon/status.**
5. **UI → NanoMQ:** 
   * **Người dùng điều khiển Gateway (bật/tắt) qua giao diện Flask (/api/gateways/control/<gateway\_id>).**
   * **ui.py publish lệnh tới NanoMQ qua topic control/<gateway\_id>.**
6. **NanoMQ → Gateway:** 
   * **Gateway subscribe topic control/<gateway\_id> để nhận lệnh (on/off) từ UI.**
7. **Redis → UI:** 
   * **ui.py đọc dữ liệu từ Redis để hiển thị:** 
     + **beacon\_data: Dữ liệu thô của Beacon.**
     + **beacon\_state: Trạng thái hiện tại của Beacon.**
     + **logs: Log hệ thống.**
     + **aws\_queue: Dữ liệu chờ gửi lên AWS.**
   * **Cập nhật realtime qua WebSocket.**

**Sơ đồ luồng dữ liệu**

**text**

**CollapseWrapCopy**

**[Gateways] → [NanoMQ: bluetooth/<gateway\_id>/data] → [Listener] → [Redis: beacon\_data, logs]**

**↓**

**[UI] ← [Redis: beacon\_data, beacon\_state, logs, aws\_queue] ← [Processor] → [Redis: beacon\_state, aws\_queue]**

**↓**

**[UI] → [NanoMQ: control/<gateway\_id>] → [Gateways] [Publisher] → [AWS IoT Core: beacon/status]**

**2. Cách đặt tên các topic MQTT**

**Nguyên tắc đặt tên**

* **Phân cấp rõ ràng: Dùng / để phân tách các cấp (namespace, entity, action/data type).**
* **Ngắn gọn, dễ hiểu: Tránh tên quá dài nhưng vẫn mô tả được ý nghĩa.**
* **Wildcard hỗ trợ: Dùng cấu trúc cho phép subscribe nhiều topic cùng lúc (nếu cần).**
* **Phân biệt nguồn: Bao gồm thông tin về Gateway hoặc thiết bị gửi dữ liệu.**

**Danh sách topic**

1. **Topic dữ liệu từ Gateway → Listener:** 
   * **Tên: bluetooth/<gateway\_id>/data**
   * **Mô tả: Dữ liệu thô từ Gateway gửi tới NanoMQ.**
   * **Ví dụ:** 
     + **bluetooth/GW1/data**
     + **bluetooth/GW2/data**
   * **Payload:**

**json**

**CollapseWrapCopy**

**{**

**"beacon\_id": "Beacon1",**

**"rssi": -70,**

**"timestamp": "2025-03-11T10:00:00"**

**}**

* + **Subscribe trong listener.py: bluetooth/+/data.**

1. **Topic điều khiển từ UI → Gateway:** 
   * **Tên: control/<gateway\_id>**
   * **Mô tả: Lệnh bật/tắt Gateway từ UI.**
   * **Ví dụ:** 
     + **control/GW1**
     + **control/GW2**
   * **Payload:**

**json**

**CollapseWrapCopy**

**{**

**"action": "on" *// hoặc "off"***

**}**

* + **Publish trong ui.py: control/<gateway\_id>.**

1. **Topic dữ liệu từ Publisher → AWS IoT Core:** 
   * **Tên: beacon/status**
   * **Mô tả: Trạng thái Beacon gửi lên AWS.**
   * **Ví dụ: beacon/status**
   * **Payload:**

**json**

**CollapseWrapCopy**

**{**

**"beacon\_id": "Beacon1",**

**"gateway\_id": "GW1",**

**"detected": 1,**

**"timestamp": "2025-03-11T10:00:00"**

**}**

* + **Được cấu hình trong .env (AWS\_TOPIC).**

**Lý do chọn tên**

* **bluetooth/<gateway\_id>/data:** 
  + **bluetooth: Namespace chỉ loại dữ liệu (Bluetooth Beacon).**
  + **<gateway\_id>: Định danh Gateway gửi dữ liệu, giúp phân biệt nguồn.**
  + **data: Loại thông tin (dữ liệu thô).**
* **control/<gateway\_id>:** 
  + **control: Namespace chỉ lệnh điều khiển.**
  + **<gateway\_id>: Định danh Gateway nhận lệnh.**
* **beacon/status:** 
  + **beacon: Namespace chỉ đối tượng (Beacon).**
  + **status: Loại thông tin (trạng thái).**

**Chạy hệ thống**

**Bước 5.1: Chạy NanoMQ**

bash

CollapseWrapCopy

docker run -d -p 1883:1883 nanomq/nanomq:0.6.6-slim

* Kiểm tra:

bash

CollapseWrapCopy

docker ps

**Bước 5.2: Chạy Redis**

bash

CollapseWrapCopy

docker run -d -p 6379:6379 redis

* Kiểm tra:

bash

CollapseWrapCopy

docker ps

**Bước 5.3: Chạy các service**

Mở từng terminal riêng trong thư mục dự án (sau khi đã kích hoạt venv):

* Terminal 1:

bash

CollapseWrapCopy

python listener.py

* Terminal 2:

bash

CollapseWrapCopy

python processor.py

* Terminal 3:

bash

CollapseWrapCopy

python publisher.py

* Terminal 4:

bash

CollapseWrapCopy

python ui.py

**6. Kiểm tra hệ thống**

**Bước 6.1: Test NanoMQ**

* Chạy đoạn code bạn đã test:

python

CollapseWrapCopy

import paho.mqtt.client as mqtt

client = mqtt.Client()

client.connect("localhost", 1883, 60)

print("Connected to NanoMQ")

* Nếu thấy "Connected to NanoMQ", NanoMQ hoạt động tốt.

**Bước 6.2: Test giao diện**

* Truy cập http://localhost:5000.
* Đăng nhập bằng admin / admin123.
* Kiểm tra:
  + Dashboard: Dữ liệu thời gian thực (Messages/Second, AWS Updates/Second).
  + Gateways: Biểu đồ số Beacon.
  + Beacons: Biểu đồ RSSI.
  + Logs: Log và biểu đồ latency.

**Bước 6.3: Test MQTT**

* Dùng MQTTX hoặc script để publish dữ liệu tới topic bluetooth/GW1:

python

CollapseWrapCopy

import paho.mqtt.client as mqtt

import json

client = mqtt.Client()

client.connect("localhost", 1883, 60)

client.publish("bluetooth/GW1", json.dumps({

"beacon\_id": "Beacon1",

"rssi": -70,

"timestamp": "2025-03-11T10:00:00"

}))

* Kiểm tra terminal chạy listener.py xem có in "Pushed to queue" không.

**7. Xử lý lỗi (nếu có)**

* **NanoMQ không chạy:** Nếu docker ps không thấy container, kiểm tra log:

bash

CollapseWrapCopy

docker logs <container\_id>

* **Kết nối thất bại:** Đảm bảo cổng 1883 không bị chặn (tường lửa Windows).
* **UI không load:** Kiểm tra terminal chạy ui.py xem lỗi cụ thể.