**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC BÌNH DƯƠNG**

**KHOA CÔNG NGHỆ THÔNG TIN, ROBOT**

**VÀ TRÍ TUỆ NHÂN TẠO**

**Ảnh có chứa biểu tượng, đồng hồ

Nội dung do AI tạo ra có thể không chính xác.**

**TIỂU LUẬN**

**MÔN: Công nghệ IoT**

**Đề tài: Đo và theo dõi tiếng ồn (openremote)**

Giảng viên hướng dẫn: Lê Duy Hùng

Sinh viên thực hiện:

***23050030 – Trần Nguyễn Khắc Duy 26TH01***

***23050051 – Nguyễn Trường Phú 26TH01***

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

GIẢNG VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

**Mục lục**

[**Chương 1.** GIỚI THIỆU PLATFORM: OPEN REMOTE. 6](#_Toc197373217)

[**1.1.** Mục đích. 6](#_Toc197373218)

[**1.2.** Các tính năng chính. 6](#_Toc197373219)

[**1.3.** Ưu điểm. 6](#_Toc197373220)

[**Chương 2.** PHÂN TÍCH KIẾN TRÚC CỦA HỆ THỐNG TRIỂN KHAI. 8](#_Toc197373221)

[**2.1.** Phần cứng. 8](#_Toc197373222)

[**2.1.1.** Board điều khiển: ESP32. 8](#_Toc197373223)

[**2.1.2.** Động cơ servo: SG92R 8](#_Toc197373224)

[**2.1.3.** Màn hình hiển thị: LCD 1602 8](#_Toc197373225)

[**2.1.4.** Cảm biến âm thanh: KY-037 9](#_Toc197373226)

[**2.2.** Phần mềm 9](#_Toc197373227)

[**2.2.1.** Cài đặt và cấu hình Open Remote 9](#_Toc197373228)

[**2.2.2.** Giao thức kết nối: MQTT 10](#_Toc197373229)

[**2.3.** Kiến trúc hệ thống 10](#_Toc197373230)

[**2.3.1.** Sơ đồ kiến trúc 10](#_Toc197373231)

[**Chương 3.** QUY TRÌNH HOẠT ĐỘNG 11](#_Toc197373232)

[**3.1.** Quy trình Kết nối 11](#_Toc197373233)

[**3.2.** SƠ ĐỒ KẾT NỐI 13](#_Toc197373234)

[**Chương 4.** CÁC BƯỚC KẾT NỐI THIẾT BỊ VÀO HỆ THỐNG 14](#_Toc197373235)

[**4.1.** 4.1. Cài đặt Phần mềm 14](#_Toc197373236)

[**4.1.1.** Cài đặt Docker 14](#_Toc197373237)

[**4.1.2.** Cài đặt Open Remote trên Docker 14](#_Toc197373238)

[**4.2.** Kết nối Phần cứng 19](#_Toc197373239)

[**4.2.1.** 4.2.1. Kết nối Cảm biến với Board Điều khiển 19](#_Toc197373240)

[**4.3.** Tích hợp vào Hệ thống 28](#_Toc197373241)

[**Chương 5.** SO SÁNH VỚI MỘT PLATFORM KHÁC 29](#_Toc197373242)

[**Chương 6.** TÀI LIỆU THAM KHẢO 30](#_Toc197373243)

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy người đã tận tâm hướng dẫn và hỗ trợ em trong suốt quá trình nghiên cứu và hoàn thiện đề tài này. Những bài giảng của thầy không chỉ mang đến kiến thức quý báu mà còn là nguồn cảm hứng lớn lao giúp em vượt qua những khó khăn trong học tập.

Em rất biết ơn thầy đã dành thời gian quý báu để góp ý, chỉ bảo cho em, giúp em có thể hoàn thành tiểu luận này một cách tốt nhất. Trong quá trình thực hiện, em đã nỗ lực hết mình và dành trọn tâm huyết, nhưng vẫn không thể tránh khỏi những thiếu sót và hạn chế. Vì vậy, em rất mong nhận được những ý kiến đóng góp và đánh giá công tâm từ thầy, để em có thể cải thiện và hoàn thiện đề tài của mình hơn nữa.

Một lần nữa, em xin chân thành cảm ơn thầy!

**LỜI MỞ ĐẦU**

Trong thời đại công nghệ thông tin phát triển nhanh chóng, việc quản lý và hiển thị dữ liệu một cách hiệu quả đã trở thành yếu tố then chốt cho sự thành công của các doanh nghiệp và tổ chức. Trong bối cảnh này, Docker nổi lên như một công cụ quan trọng, nhờ khả năng tạo ra các môi trường ảo hóa nhẹ và linh hoạt, giúp tối ưu hóa quy trình triển khai và quản lý ứng dụng.

Khi kết hợp với nền tảng Open Remote — một hệ thống quản lý dữ liệu mở và mạnh mẽ — việc sử dụng Docker để hiển thị dữ liệu không chỉ mang lại hiệu quả cao mà còn mở ra nhiều cơ hội mới cho các tổ chức.

Tiểu luận này sẽ khám phá quy trình triển khai Docker nhằm hiển thị dữ liệu trên nền tảng Open Remote. Chúng ta sẽ đi sâu vào các bước cài đặt, cấu hình và tối ưu hóa hệ thống, đồng thời phân tích những lợi ích và thách thức mà giải pháp này mang lại. Qua đó, tiểu luận sẽ cung cấp cái nhìn tổng quan và chi tiết về ứng dụng của Docker và Open Remote trong việc quản lý và hiển thị dữ liệu một cách hiệu quả.

# GIỚI THIỆU PLATFORM: OPEN REMOTE.

## Mục đích.

Open Remote là một nền tảng nhằm kết nối các chuyên gia và nhà tuyển dụng từ khắp nơi trên thế giới, tạo ra cơ hội làm việc từ xa trong các lĩnh vực khác nhau. Mục tiêu chính của nền tảng này là tối ưu hóa quá trình tuyển dụng, giúp người lao động dễ dàng tìm kiếm việc làm phù hợp và các doanh nghiệp tìm được nhân lực chất lượng.

## Các tính năng chính.

* Tìm kiếm việc làm: Cho phép người dùng tìm kiếm các cơ hội việc làm từ xa theo ngành nghề, địa điểm và mức lương.
* Đăng tin tuyển dụng: Doanh nghiệp có thể đăng tải thông tin tuyển dụng một cách dễ dàng và nhanh chóng.
* Hồ sơ ứng viên: Người tìm việc có thể tạo hồ sơ cá nhân, làm nổi bật kỹ năng và kinh nghiệm làm việc.
* Giao tiếp trực tiếp: Tính năng chat cho phép nhà tuyển dụng và ứng viên trao đổi thông tin một cách nhanh chóng.
* Đánh giá và phản hồi: Người dùng có thể đánh giá lẫn nhau, giúp xây dựng uy tín và độ tin cậy trên nền tảng.

## Ưu điểm.

* Tiếp cận toàn cầu: Người dùng có thể kết nối với nhà tuyển dụng và ứng viên từ khắp nơi trên thế giới.
* Tiết kiệm thời gian: Quá trình tìm kiếm và ứng tuyển được tối ưu hóa, giảm thiểu thời gian chờ đợi.
* Đa dạng ngành nghề: Nền tảng hỗ trợ nhiều lĩnh vực khác nhau, từ công nghệ thông tin đến marketing, thiết kế và hơn thế nữa.
* Dễ sử dụng: Giao diện thân thiện và dễ dàng điều hướng, phù hợp với mọi đối tượng người dùng.

# PHÂN TÍCH KIẾN TRÚC CỦA HỆ THỐNG TRIỂN KHAI.

## Phần cứng.

### Board điều khiển: ESP32.

**ESP32:** Đây là các board điều khiển vi xử lý hiện đại với khả năng kết nối Wi-Fi và Bluetooth, phù hợp cho các ứng dụng IoT nâng cao. ESP32 được ưa chuộng nhờ hiệu năng mạnh mẽ, nhiều chân GPIO, hỗ trợ ADC, DAC, và khả năng xử lý đa nhiệm. Nó cũng hỗ trợ các giao thức truyền thông như HTTP, MQTT, và BLE.

**ESP32:** Là một vi điều khiển đa năng với kết nối Wi-Fi và Bluetooth, rất phổ biến trong cộng đồng IoT nhờ hiệu suất cao, tính linh hoạt và dễ dàng trong lập trình, mở rộng các dự án thông minh.

### Động cơ servo: SG92R

**SG92R:** Đây là một loại động cơ servo mini phổ biến, thường được sử dụng trong các dự án Arduino, robot, và mô hình điều khiển. SG92R có kích thước nhỏ gọn, trọng lượng nhẹ, phản hồi nhanh và dễ điều khiển thông qua tín hiệu PWM.

**SG92R:** Là một động cơ servo sử dụng phổ biến trong các ứng dụng điều khiển vị trí nhờ vào khả năng phản hồi chính xác, độ tin cậy cao và dễ dàng tích hợp vào các mạch điều khiển vi xử lý.

### Màn hình hiển thị: LCD 1602

**LCD 1602:** Là một module hiển thị ký tự gồm 2 dòng, mỗi dòng 16 ký tự, sử dụng công nghệ LCD STN với đèn nền LED. Module này thường được tích hợp với vi điều khiển như Arduino hoặc ESP32 để hiển thị thông tin như dữ liệu cảm biến, trạng thái hệ thống, hoặc giao diện người dùng đơn giản.

**LCD 1602:** Hỗ trợ giao tiếp song song 4-bit hoặc 8-bit, và có thể dễ dàng điều khiển thông qua các thư viện phổ biến. Phiên bản sử dụng giao thức I2C giúp giảm số lượng chân kết nối, thuận tiện cho các dự án với không gian hạn chế.

### Cảm biến âm thanh: KY-037

**KY-037:** Là một mô-đun cảm biến âm thanh có độ nhạy cao, sử dụng microphone tụ điện (electret condenser microphone) để phát hiện âm thanh trong môi trường. Mô-đun này cung cấp cả hai đầu ra:

* + **Đầu ra analog (AO):** Cung cấp tín hiệu điện áp tương ứng với cường độ âm thanh thu được.
  + **Đầu ra digital (DO):** Phát tín hiệu mức cao khi âm thanh vượt ngưỡng đã thiết lập.

Ngưỡng âm thanh có thể được điều chỉnh thông qua biến trở tích hợp trên mô-đun. KY-037 thường được sử dụng trong các ứng dụng như điều khiển bằng âm thanh, hệ thống cảnh báo, hoặc các dự án tương tác với môi trường âm thanh.

## Phần mềm

### Cài đặt và cấu hình Open Remote

* **Cài đặt Open Remote:** Open Remote có thể được cài đặt trên nhiều nền tảng, bao gồm máy tính cá nhân, máy chủ ảo, hoặc dưới dạng Docker container. Việc sử dụng Docker giúp đơn giản hóa quá trình cài đặt và triển khai hệ thống, đồng thời đảm bảo tính linh hoạt và khả năng mở rộng.
* **Cấu hình Open Remote:** Sau khi cài đặt, cần cấu hình Open Remote để kết nối với các thiết bị IoT và cảm biến. Cấu hình này bao gồm việc thiết lập các endpoint cho các cảm biến, thiết lập các giao thức truyền thông (MQTT), và cấu hình giao diện dashboard để hiển thị dữ liệu.

### Giao thức kết nối: MQTT

**MQTT:** MQTT hoạt động dựa trên mô hình publish-subscribe, giúp giảm thiểu lưu lượng dữ liệu và cho phép cập nhật dữ liệu gần như tức thời từ các cảm biến đến hệ thống Open Remote. MQTT cũng có khả năng duy trì kết nối liên tục và đáng tin cậy trong môi trường mạng không ổn định.

## Kiến trúc hệ thống

### Sơ đồ kiến trúc

**Sơ đồ kiến trúc của hệ thống gồm các thành phần chính:**

1. **Thiết bị cảm biến (KY-037):** Đo độ lớn của âm thanh qua dạng datalog và Digital sau đó gửi lên board điều khiển.
2. **Màn hình LCD 1602:** Hiển thị thông tin trạng thái của servo, thông tin độ lơn âm thanh.
3. **Động cơ servo SG92R:** khi ở trạng thái bình thường sẽ set trục ở vị trí độ 0 khi có tinis hiệu từ board điều khiển sẽ quay 1 góc 90 độ để mở cửa.
4. **Board điều khiển (ESP32):** Nhận dữ liệu từ cảm biến KY-037 tính toán tính hiệu mà KY-037 gửi lên thay vào thông tin sang dạng dB(decibel) nếu độ lớn âm thanh vượt giới hạn cho phép, điều khiểu servo quay 90 độ, gửi điều khiển hiện thị vào LCD 1602 và xử lý trước khi truyền dữ liệu qua giao thức HTTP hoặc MQTT đến hệ thống Open Remote.
5. **Open Remote Server:** Hệ thống quản lý trung tâm nhận dữ liệu từ các board điều khiển thông qua HTTP hoặc MQTT, lưu trữ và phân tích dữ liệu. Hệ thống này có thể triển khai trên Docker để dễ dàng quản lý và mở rộng.

**Dashboard:** Giao diện người dùng của Open Remote, nơi dữ liệu nhiệt độ và độ ẩm được hiển thị dưới dạng biểu đồ, bảng số liệu, hoặc các widget tùy chỉnh. Người dùng có thể truy cập dashboard này từ bất kỳ thiết bị nào có kết nối internet để theo dõi và giám sát các thông số môi trường

# QUY TRÌNH HOẠT ĐỘNG

## Quy trình Kết nối

**Bước 1: Đo lường dữ liệu từ cảm biến KY-037**

* **Mô tả:** Là một mô-đun cảm biến âm thanh có độ nhạy cao, sử dụng microphone tụ điện (electret condenser microphone) để phát hiện âm thanh trong môi trường
* **Hoạt động:** khi có sự thay đổi về âm thanh thì sensor liên tục cập nhật tín hiệu là gửi tín hiệu cho board điều khiển.

**Bước 2: Truyền dữ liệu đến board điều khiển**

* **Mô tả:** Board điều khiển là thiết bị chịu trách nhiệm thu thập dữ liệu từ cảm biến KY-037. Nó được lập trình để đọc dữ liệu từ cảm biến và xử lý sơ bộ trước khi truyền đi.
  + **Hoạt động:** Board điều khiển được kết nối với cảm biến KY-037, servo, màn led thông qua các chân GPIO.
  + Một đoạn mã (thường viết bằng ngôn ngữ lập trình như C++ hoặc MicroPython) sẽ liên tục đọc dữ liệu từ cảm biến theo một chu kỳ nhất định.
  + Sau khi dữ liệu được đọc, board điều khiển có thể thực hiện một số xử lý cơ bản, như lọc dữ liệu, gửi dữ liệu lên led và servo.

**Bước 3: Truyền dữ liệu đến Open Remote thông qua HTTP/MQTT**

* **Mô tả:** Sau khi xử lý dữ liệu, board điều khiển sẽ gửi dữ liệu này đến hệ thống Open Remote bằng cách sử dụng giao thức truyền thông như HTTP hoặc MQTT.
* **Hoạt động:**
  + **Giao thức HTTP:** Board điều khiển gửi yêu cầu POST chứa dữ liệu (JSON/XML) về nhiệt độ và độ ẩm đến máy chủ Open Remote.
  + **Giao thức MQTT:** Board điều khiển xuất bản dữ liệu đến một topic cụ thể trên broker MQTT, nơi Open Remote đã đăng ký lắng nghe và nhận dữ liệu.
  + Cả hai giao thức đều đảm bảo dữ liệu được truyền một cách an toàn và chính xác từ thiết bị đến hệ thống trung tâm.

**Bước 4: Nhận và lưu trữ dữ liệu tại Open Remote**

* **Mô tả:** Khi Open Remote nhận được dữ liệu từ board điều khiển, hệ thống sẽ tiến hành lưu trữ và xử lý dữ liệu để chuẩn bị cho việc hiển thị trên dashboard.
* **Hoạt động:**
  + Dữ liệu được lưu trữ trong cơ sở dữ liệu của Open Remote, có thể là SQL, NoSQL hoặc các dịch vụ lưu trữ đám mây tùy thuộc vào cấu hình hệ thống.
  + Hệ thống có thể thực hiện các thao tác như làm sạch dữ liệu, tính toán các chỉ số bổ sung (ví dụ: chỉ số độ ẩm tương đối), và chuẩn bị dữ liệu cho hiển thị trực quan.

**Bước 5: Hiển thị dữ liệu trên Dashboard**

* **Mô tả:** Dữ liệu nhiệt độ và độ ẩm được hiển thị theo thời gian thực trên dashboard của Open Remote, nơi người dùng có thể dễ dàng quan sát và phân tích.
* **Hoạt động:**
* Dashboard của Open Remote cung cấp các biểu đồ, bảng dữ liệu, và các widget tương tác để hiển thị thông tin thu thập từ cảm biến.
* Người dùng có thể tùy chỉnh giao diện để hiển thị dữ liệu theo ý muốn, chẳng hạn như biểu đồ đường, biểu đồ cột.

## SƠ ĐỒ KẾT NỐI

Ảnh có chứa ảnh chụp màn hình, biểu đồ, mạch điện

Nội dung do AI tạo ra có thể không chính xác.

# CÁC BƯỚC KẾT NỐI THIẾT BỊ VÀO HỆ THỐNG

## 4.1. Cài đặt Phần mềm

### Cài đặt Docker

* **Mô tả:** Docker là một nền tảng giúp triển khai ứng dụng trong các container, đảm bảo tính nhất quán và dễ dàng quản lý. Để cài đặt Docker, bạn cần tải và cài đặt Docker Desktop từ trang web chính thức của Docker.
* **Hướng dẫn cài đặt:**
  1. **Tải Docker Desktop:** Truy cập trang Docker Desktop và tải phiên bản phù hợp với hệ điều hành của bạn.
  2. **Cài đặt Docker Desktop:** Mở tệp cài đặt và làm theo hướng dẫn để hoàn tất quá trình cài đặt.
  3. **Khởi động Docker Desktop:** Sau khi cài đặt, mở Docker Desktop và đảm bảo rằng Docker Engine đang chạy.

### Cài đặt Open Remote trên Docker

* **Mô tả:** Open Remote có thể được triển khai trên Docker để dễ dàng quản lý và bảo trì. Bạn sẽ sử dụng các tệp Docker Compose để cấu hình và triển khai Open Remote.
* **Hướng dẫn cài đặt:**
  1. **Tạo thư mục dự án:** Tạo một thư mục để chứa tệp cấu hình Docker Compose. A black screen with a black border

     Description automatically generated
  2. **Tạo tệp Docker Compose:** Tạo một tệp có tên docker-compose.yml với nội dung cấu hình cho Open Remote.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.  
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

File này có thể lấy trực tiếp từ trang chủ của OpenRemote platform..

* 1. **Khởi động Open Remote:** Chạy lệnh Docker Compose để bắt đầu triển khai Open Remote.



Việc này openremote sẽ chạy mặc định trên cổng 8883.

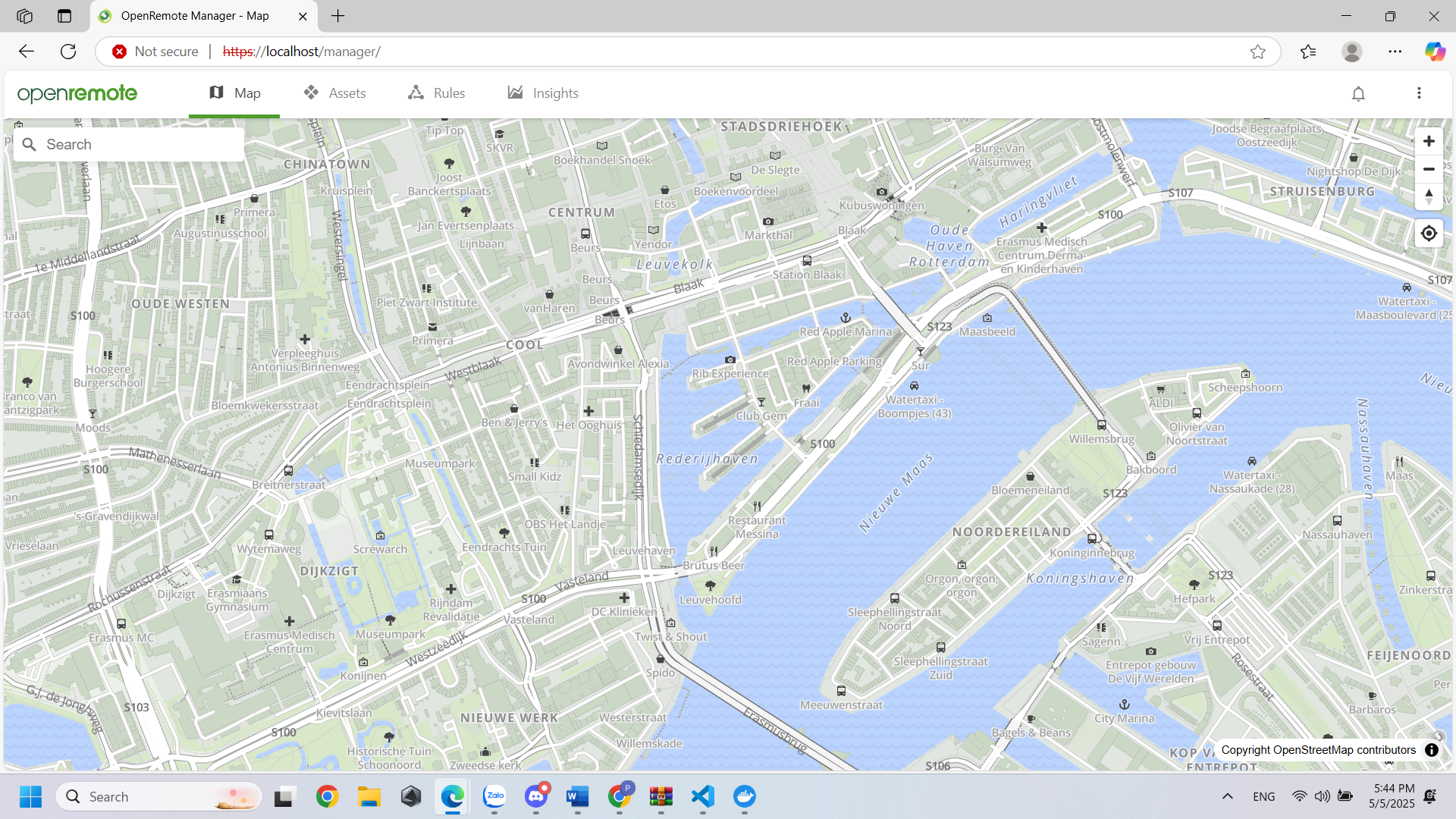
* 1. **Kiểm tra trạng thái:** Truy cập http://localhost:8080 để đảm bảo rằng Open Remote đang chạy.

Sử dụng tài khoản mặc định được cấp của platform để truy cập vào hệ thống

**Username: admin**

**Password: secret**

Trang web hiển thị với 4 Tab làm việc chính (giao diện admin)

  
Truy cập “Assets” → Click “+” (Add) để thêm thiết bị hoặc đại lý quản lý thiết bị -> chọn things assets và custom đơn giản để có thể Publish và Subscribe trong đây chỉ cần gửi dữ liệu nên ta đơn giản là thêm attribute, Đặt tên và click “Add”.để Publish như hình dưới

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Nội dung do AI tạo ra có thể không chính xác.

Lưu ý là phải lấy ID (ở trong url của things asset đó) của things assets có attribute cần gửi lên để điền thông tin cho mã topic cho lập trình ở board điều kiển để gửi và nhận dữ liệu theo quy định đã tạo trước đó

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Nội dung do AI tạo ra có thể không chính xác.

Lưu ý: để có thể gửi tín hiệu lên platform ta phải chú ý đến các id trong lập trình ở esp mqtt explorer (nếu dùng) và nó phải duy nhất không thay đổi hoặc trùng.

## Kết nối Phần cứng

### 4.2.1. Kết nối Cảm biến với Board Điều khiển

**Mô tả :** cảm biếnKY-037 kết nối với board ESP32 để truyền dữ liệu về âm thanh cho màn hình và servo.

**Cách kết nối :**

**ESP32**:

* + - **Chân 14** → Servo (tín hiệu)
    - **Chân 34** → Cảm biến âm thanh (analog)
    - **Chân 27** → Cảm biến âm thanh (digital)
    - **Chân SDA** → LCD (I2C)
    - **Chân SCL** → LCD (I2C)
    - **+5V** → Servo, LCD
    - **GND** → Servo, LCD, cảm biến âm thanh

**Cấp nguồn :** ESP32 được cấp nguồn qua USB

**Cấu hình Board điều khiển để truyền dữ liệu qua MQTT**

**Mô tả** **:** ESP32 cần lập trình để đọc dữ liệu của cảm biến âm thanh LM393 và truyền đến Open Remote qua MQTT.

**Hướng dẫn cấu hình :**

* + **Cài đặt Arduino IDE:** Nếu chưa có, tải và cài đặt Arduino IDE từ Arduino.
  + **Cài đặt thư viện:** Cài đặt các thư viện cần thiết cho ESP32 và KY-037 trong Arduino IDE.
  + **Viết mã nguồn :**

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include <PubSubClient.h>

#include <math.h>

#include <ESP32Servo.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// WiFi

const char\* ssid = "Fablab 2.4G";

const char\* password = "Fira@2024";

// MQTT

const char\* mqtt\_server = "192.168.69.197";

const int mqtt\_port = 8883;

const char\* mqtt\_user = "master:espclient";

const char\* mqtt\_password = "AW16vkkxjXzAp6yviZVOebhOXBdNiOdQ";

const char\* client\_id = "client1234";

// Asset & Attributes

const char\* assetId = "38r75BrlYpigQRYk5XUAUy";

const char\* attributeControlMode = "controlMode";

const char\* attributeServoState = "servoState";

const char\* attributeName = "writeAttribute"; // dB

const char\* servoStatusAttribute = "WDst"; // OPEN / CLOSED

// MQTT

WiFiClientSecure secureClient;

PubSubClient client(secureClient);

// Phần cứng

Servo myServo;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

bool Window = false;

// Trạng thái điều khiển

bool isAutoMode = true;

bool manualServoState = false;

// Cảm biến âm thanh

#define SOUND\_ANALOG\_PIN 34

#define SOUND\_DIGITAL\_PIN 27

#define SERVO\_PIN 14

#define SOUND\_DB\_THRESHOLD 41.0

float readSoundDb(int analogPin, int sampleWindow = 50) {

unsigned long startMillis = millis();

int signalMax = 0;

int signalMin = 1023;

while (millis() - startMillis < sampleWindow) {

int sample = analogRead(analogPin);

signalMax = max(signalMax, sample);

signalMin = min(signalMin, sample);

}

int peakToPeak = signalMax - signalMin;

peakToPeak = max(peakToPeak, 1);

return 20.0 \* log10((float)peakToPeak);

}

void setup\_wifi() {

Serial.begin(115200);

Serial.print("Đang kết nối WiFi...");

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWiFi đã kết nối");

Serial.print("IP: ");

Serial.println(WiFi.localIP());

}

void reconnect() {

while (!client.connected()) {

Serial.print("Kết nối MQTT...");

if (client.connect(client\_id, mqtt\_user, mqtt\_password)) {

Serial.println(" -> OK");

String topic1 = "master/" + String(client\_id) + "/attribute/" + attributeControlMode + "/" + assetId;

String topic2 = "master/" + String(client\_id) + "/attribute/" + attributeServoState + "/" + assetId;

client.subscribe(topic1.c\_str());

client.subscribe(topic2.c\_str());

Serial.println("Đã subscribe: " + topic1);

Serial.println("Đã subscribe: " + topic2);

} else {

Serial.print("Lỗi: ");

Serial.println(client.state());

delay(5000);

}

}

}

void callback(char\* topic, byte\* payload, unsigned int length) {

String msg;

for (int i = 0; i < length; i++) {

msg += (char)payload[i];

}

bool value = (msg == "true");

String topicStr = String(topic);

if (topicStr.indexOf(attributeControlMode) != -1) {

isAutoMode = value;

Serial.println("→ Nhận controlMode: " + String(isAutoMode ? "AUTO" : "MANUAL"));

}

if (topicStr.indexOf(attributeServoState) != -1) {

manualServoState = value;

Serial.println("→ Nhận servoState (manual): " + String(manualServoState ? "OPEN" : "CLOSED"));

}

}

void setup() {

setup\_wifi();

secureClient.setInsecure();

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

myServo.attach(SERVO\_PIN);

myServo.write(0);

Wire.begin();

lcd.init();

lcd.backlight();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("ESP Ready");

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

float soundDb = readSoundDb(SOUND\_ANALOG\_PIN);

if (isAutoMode) {

if (soundDb > SOUND\_DB\_THRESHOLD && !Window) {

myServo.write(90);

Window = true;

} else if (soundDb <= SOUND\_DB\_THRESHOLD && Window) {

myServo.write(0);

Window = false;

}

} else {

if (manualServoState) {

myServo.write(90);

Window = true;

} else {

myServo.write(0);

Window = false;

}

}

// Hiển thị

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("dB: ");

lcd.print(soundDb, 1);

lcd.setCursor(0, 1);

lcd.print(isAutoMode ? "AUTO " : "MANUAL ");

lcd.print(Window ? "OPEN " : "CLOSED");

// Publish dữ liệu lên MQTT

String pubNoise = "master/" + String(client\_id) + "/writeattributevalue/" + attributeName + "/" + String(assetId);

String pubServoStatus = "master/" + String(client\_id) + "/writeattributevalue/" + servoStatusAttribute + "/" + String(assetId);

String pubControlMode = "master/" + String(client\_id) + "/writeattributevalue/" + attributeControlMode + "/" + String(assetId);

String pubServoState = "master/" + String(client\_id) + "/writeattributevalue/" + attributeServoState + "/" + String(assetId);

client.publish(pubNoise.c\_str(), String(soundDb, 2).c\_str());

client.publish(pubServoStatus.c\_str(), Window ? "\"OPEN\"" : "\"CLOSED\"");

client.publish(pubControlMode.c\_str(), isAutoMode ? "true" : "false");

client.publish(pubServoState.c\_str(), manualServoState ? "true" : "false");

delay(5000);

}

## Tích hợp vào Hệ thống

**1. Thiết lập Open Remote để Nhận và Xử lý Dữ liệu từ Board Điều khiển**

* **Mô tả:** Open Remote cần được cấu hình để nhận dữ liệu từ ESP32 qua giao thức MQTT và hiển thị trên dashboard.
* **Hướng dẫn thiết lập:**
  1. **Truy cập Open Remote:** Mở trình duyệt và truy cập vào địa chỉ của Open Remote (http://localhost:8080).
  2. **Tạo kết nối MQTT:** Trong giao diện quản lý của Open Remote, cấu hình kết nối đến broker MQTT.
     + Cung cấp thông tin về broker MQTT, bao gồm địa chỉ IP và port (thường là 1883).
  3. **Thiết lập topic:** Đảm bảo rằng Open Remote đăng ký lắng nghe topic mà ESP32 xuất bản dữ liệu (ví dụ: "sensor/data").
  4. **Cấu hình dữ liệu:** Tạo các cấu hình dữ liệu và các widget trên dashboard để hiển thị dữ liệu nhiệt độ và độ ẩm nhận được từ cảm biến.
  5. **Kiểm tra và giám sát:** Đảm bảo rằng dữ liệu từ ESP32 được hiển thị chính xác trên dashboard và thực hiện các điều chỉnh nếu cần thiết.

# SO SÁNH VỚI MỘT PLATFORM KHÁC

**Nền Tảng Dựa Trên Docker**

* **Kiểm Soát & Linh Hoạt**: Các nhà phát triển có quyền kiểm soát chi tiết về môi trường chạy, cấu hình và phụ thuộc.
* **Tính Đồng Nhất**: Ứng dụng hoạt động nhất quán trên các môi trường khác nhau, từ phát triển cục bộ đến sản xuất, nhờ vào việc đóng gói trong container.
* **Khả Năng Di Chuyển**: Containers có thể được triển khai trên nhiều nhà cung cấp đám mây, tại chỗ, hoặc trong các thiết lập hybrid mà không cần sửa đổi.
* **Quản Lý Tài Nguyên**: Người dùng quản lý việc mở rộng theo cách thủ công hoặc sử dụng các công cụ điều phối như Kubernetes để xử lý các triển khai phức tạp.
* **Thời Gian Khởi Động**: Containers khởi động nhanh hơn so với máy ảo truyền thống, nhưng chậm hơn so với các chức năng serverless.

**Nền Tảng Serverless**

* **Khả Năng Mở Rộng**: Tự động mở rộng lên hoặc xuống dựa trên nhu cầu, loại bỏ sự can thiệp thủ công.
* **Chi Phí Hiệu Quả**: Chi phí dựa trên thời gian thực thi và mức tiêu thụ tài nguyên, giúp tiết kiệm cho các khối lượng công việc biến động.
* **Triển Khai Nhanh**: Các nhà phát triển có thể triển khai mã mà không cần quản lý cơ sở hạ tầng hoặc phụ thuộc vào hệ điều hành, cho phép cập nhật nhanh chóng.
* **Dựa Trên Sự Kiện**: Phù hợp cho các ứng dụng phản hồi dựa trên sự kiện, chẳng hạn như yêu cầu API hoặc tải tệp.
* **Thời Gian Khởi Động**: Các chức năng có thể thực thi gần như ngay lập tức, nhưng có thể gặp độ trễ trong các lần khởi động chậm.

# TÀI LIỆU THAM KHẢO

**Trang Chủ Open Remote:** https://www.openremote.io

**Tài Liệu Docker:** Docker Documentation: https://docs.docker.com

Docker Compose: https://docs.docker.com/compose/

**Tài Liệu MQTT:** MQTT Documentation: [MQTT.org](http://mqtt.org/)

**Arduino IDE:** Arduino Software: https://www.arduino.cc/en/software

**Thư Viện ESP32:** <https://github.com/arduino-libraries/WiFi>

**Thư viện WIFIClientSecure:** <https://github.com/pamelacece/WiFiClientSecure>

**Thư viện Pubsubclient:** <https://github.com/knolleary/pubsubclient>

Thư viện ESP32Servo <https://www.arduinolibraries.info/libraries/esp32-servo>

LiquidCrystal I2C : <https://www.arduinolibraries.info/libraries/liquid-crystal-i2-c>

Link GitHub :