

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



PHÁT TRIỂN ỨNG DỤNG IoT (CO3038)

Project: SMART HOME SYSTEM

GVHD: Huỳnh Hoàng Kha
Lê Trọng Nhân

Sinh viên: Nguyễn Long Nhật - 1813367
Huỳnh Ngọc Tấn - 1813952
Bùi Tiến Bình - 1912713



Content

1	Nhiệm vụ thực hiện	2
2	Giới thiệu đề tài	3
2.1	Tìm hiểu lợi ích của Smart Home	3
2.2	Sự phát triển của Smart Home	4
2.3	Cấu trúc tổng quan và danh sách thiết bị	6
2.4	Xây dựng sơ đồ kiến trúc hệ thống	10
2.4.1	HiveMQ Server	11
2.4.2	Firebase	12
3	Hiện thực kết nối với Sensors	13
3.1	Xử lý dữ liệu từ sensors	13
3.2	Mở rộng chức năng của hàm <i>processData</i>	14
4	Hiện thực Gateway	16
4.1	UART received data	16
4.2	Error Control	16
4.3	Data uploading process	20
4.4	Data received process	20
4.5	Model AI Detect Person	21
5	Định cấu hình cho Server MQTT	22
6	Xây dựng ứng dụng Mobile	25
6.1	Chức năng bổ sung 1: Sử dụng Firebase cho ứng dụng	32
6.2	Chức năng bổ sung 2: Đăng nhập với tài khoản Google	32
6.3	Chức năng bổ sung 3: Sử dụng thêm Server MQTT khác: HiveMQ	32
7	Kết luận	33
	Reference	34

1 Nhiệm vụ thực hiện

Student ID	Name	Task
1813367	Nguyễn Long Nhật	<ul style="list-style-type: none">– Lên ý tưởng cho đề tài, tìm hiểu thực trạng đề tài.– Xây dựng sơ đồ kiến trúc hệ thống.– Thiết kế giao diện cho ứng dụng Mobile.– Hiện thực các chức năng cho ứng dụng Mobile.– Xây dựng chức năng đăng nhập, đăng xuất và đăng nhập bằng tài khoản Google.– Tìm hiểu HiveMQ, sử dụng server này cho Gateway và ứng dụng Mobile.– Hiện thực tính năng kiểm soát lỗi cho Gateway.– Tích hợp Model AI phát hiện có người cho Gateway.– Viết final report cho project.
1813952	Huỳnh Ngọc Tấn	<ul style="list-style-type: none">– Lên ý tưởng cho đề tài, tìm hiểu thực trạng đề tài.– Xây dựng sơ đồ kiến trúc hệ thống.– Xây dựng các Sequence Diagram cho các chức năng của ứng dụng Mobile.– Thiết kế Mockup Diagram cho ứng dụng Mobile.– Chỉnh sửa giao diện cho ứng dụng Mobile.– Unit Testing cho toàn bộ tính năng của hệ thống.– Hiện thực Gateway cho hệ thống.
1912713	Bùi Tiến Bình	<ul style="list-style-type: none">– Lên ý tưởng cho đề tài, tìm hiểu thực trạng đề tài.– Xây dựng sơ đồ kiến trúc hệ thống.– Xây dựng các Sequence Diagram cho các chức năng của ứng dụng Mobile.– Chỉnh sửa giao diện cho ứng dụng Mobile.– Unit Testing cho toàn bộ tính năng của hệ thống.– Thiết kế Mockup Diagram cho ứng dụng Mobile.– Hiện thực Gateway cho hệ thống

2 Giới thiệu đề tài

2.1 Tìm hiểu lợi ích của Smart Home

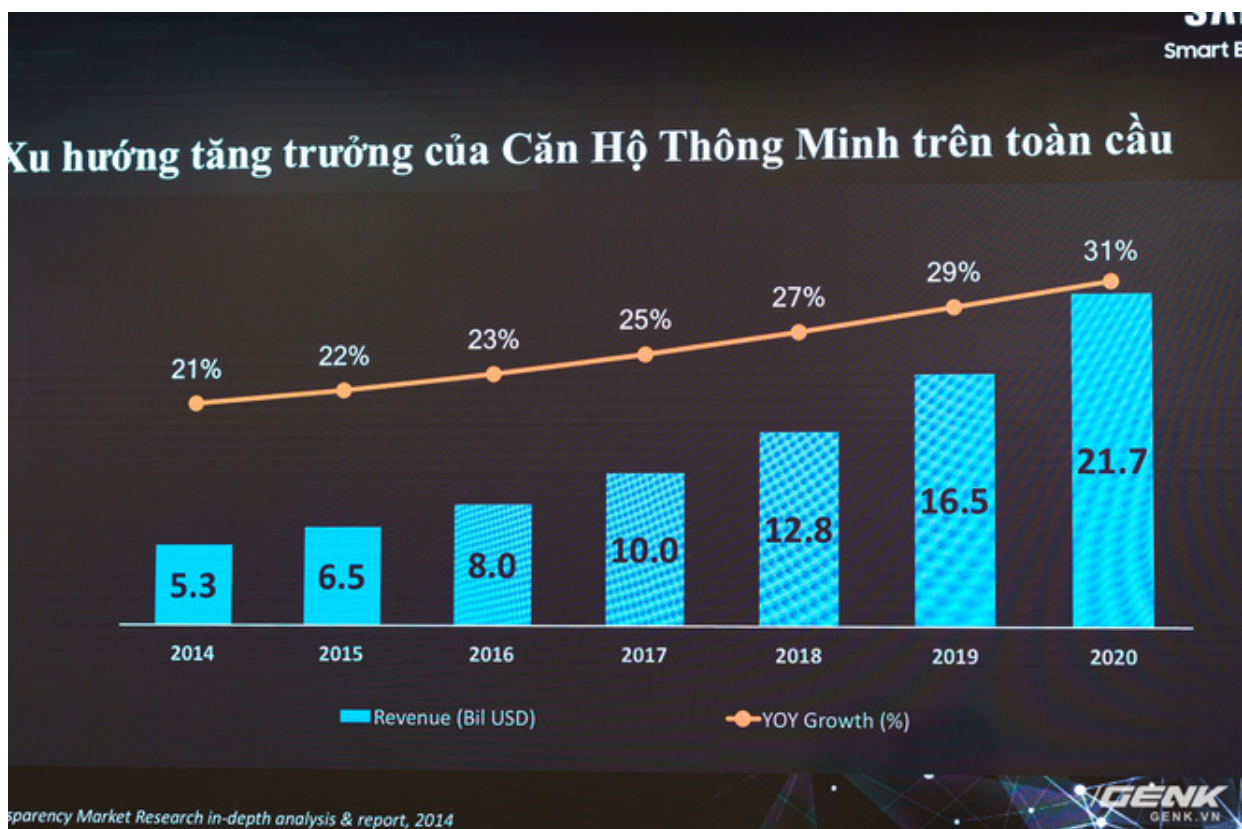
Smart Home giúp cuộc sống của con người trở nên dễ dàng hơn, thoải mái hơn, những lợi ích to lớn mà Smart Home mang lại cho con người là:

- **Tiết kiệm năng lượng và tiền bạc:** đối với nhiều người, động lực để họ chọn một ngôi nhà thông minh là tiết kiệm năng lượng và tiền bạc với một hệ thống đèn hoặc điều hòa không khí tự động (chẳng hạn như bộ điều hòa nhiệt độ do công ty con Nest của Alphabet sản xuất có thể tự động hóa nhanh chóng việc làm mát hoặc sưởi ấm ngôi nhà), từ đó giúp hóa đơn tiền điện của chủ sở hữu giảm đáng kể. Những sản phẩm tự động kiểu như thế này luôn là lựa chọn hàng đầu trong checklist của bất kì ai muốn biến ngôi nhà của mình thành Smart Home.
- **Làm cuộc sống trở nên đơn giản hơn:** con người ta luôn muốn cuộc sống của mình trở nên đơn giản hơn, và Smart Home giúp chúng ta làm điều đó. Chúng ta hoàn toàn có thể điều khiển những thiết bị trong nhà như bật/tắt đèn, đóng/mở cửa chỉ thông qua một chiếc điện thoại di động.
- **An ninh và an toàn:** Smart Home hoàn toàn có thể tạo cho chủ sở hữu cảm giác an toàn với những thiết bị cảm biến cảnh báo khi có người đột nhập trái phép, camera và hệ thống báo động cũng đóng góp một phần đáng kể vào vấn đề an ninh này của Smart Home.
- **Xử lý các công việc hàng ngày:** robot hút bụi Roomba của iRobot là một ví dụ về việc những thiết bị điện tử có thể giúp chúng ta làm những việc tẻ nhạt hàng ngày. Smart Home cũng có khả năng làm được điều đó, ví dụ với những chiếc máy giặt đặt lịch tự động giặt mỗi ngày, hoặc tủ lạnh tự động đặt hàng khi trong tủ hết đồ ăn.
- **Làm ngôi nhà trở nên thú vị hơn:** các thiết bị thông minh có thể giúp chúng ta chiếu những bộ phim, hay phát những bản nhạc phụ thuộc vào sở thích của chúng ta, hoặc nó có thể phát hiện khi chúng ta buồn để làm một điều gì đó giúp chúng ta vui hơn, như giả tiếng chim hót,...

2.2 Sự phát triển của Smart Home

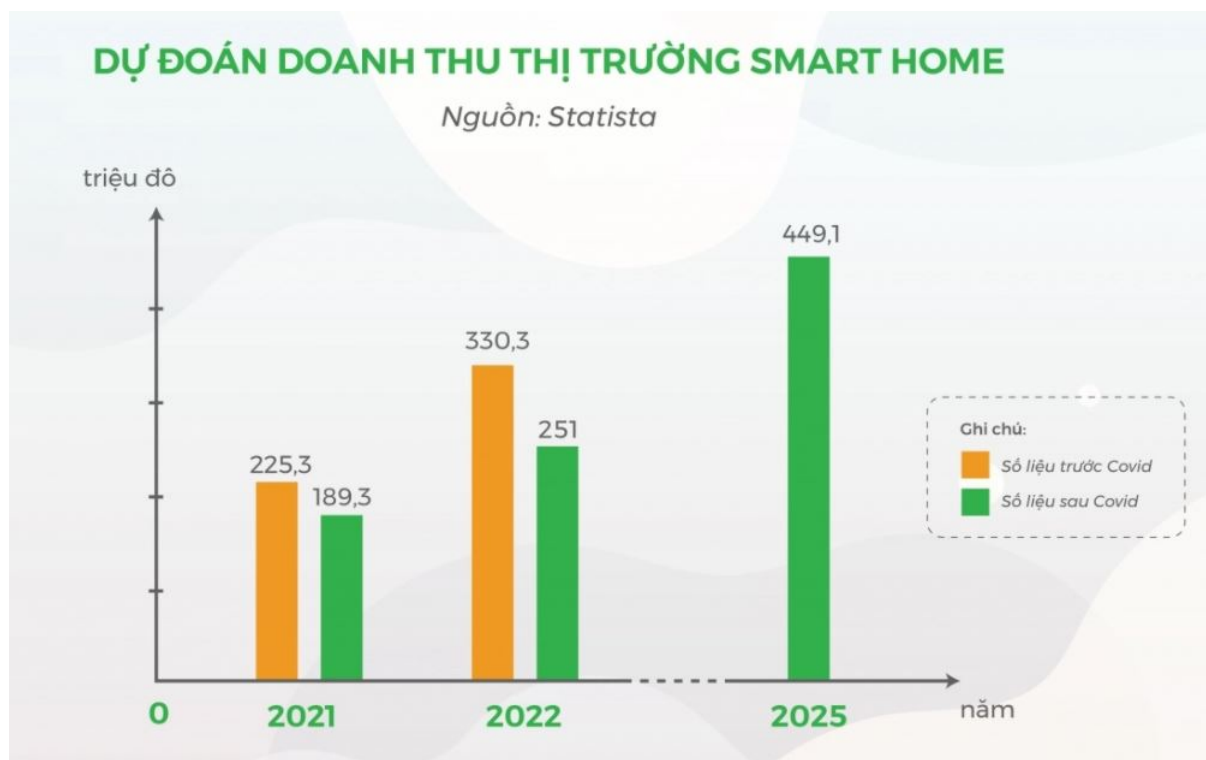
Trong khi Bắc Mỹ và Châu Âu là những thị trường tiên phong, khu vực Châu Á-Thái Bình Dương đang chứng kiến sự tăng trưởng đáng kể số lượng các dự án nhà ở mới có triển khai Smart Home, với nhu cầu khổng lồ về các thiết bị điều khiển tự động trong chiếu sáng, an ninh và giải trí.

Smarthome là một thị trường cực kỳ tiềm năng với giá trị không ngừng gia tăng mạnh qua từng năm. Ước tính năm 2019, thị trường Smart Home toàn cầu tăng trưởng 29%, đạt mức 16,5 tỷ đô và sẽ tăng lên thành 21,7 tỷ đô trong năm sau.



Hình 1: Sự phát triển của Smart Home trên thế giới.

Theo báo cáo công bố trong giai đoạn 2019 – 2020, Statista dự đoán thị trường Nhà thông minh tại Việt Nam sẽ đạt doanh thu 225,3 triệu đô vào năm 2021 và 330,4 triệu đô vào năm 2022. Tuy nhiên, trong báo cáo mới nhất được công bố cuối 2020, những số liệu này đã có sự thay đổi vì những ảnh hưởng trực tiếp của đại dịch COVID-19. Cụ thể, doanh thu Nhà thông minh tại thị trường Việt Nam được dự đoán sẽ lần lượt đạt mức 183,9 triệu đô vào 2021, 251 triệu đô vào 2022 và con số này sẽ tiếp tục tăng lên 449,1 triệu đô vào 2025.

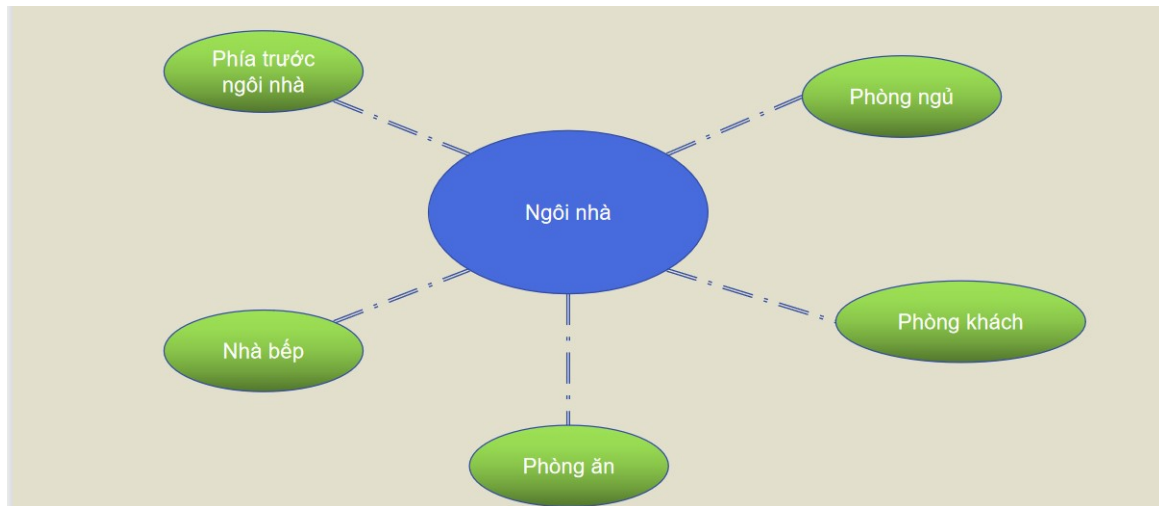


Hình 2: Sự phát triển của Smart Home ở Việt Nam.

Những số liệu của Statista chỉ ra rằng, quy mô thị trường Smart home Việt sẽ vẫn ngày càng mở rộng, cộng hưởng cùng với xu thế phát triển chung trên thế giới. Tuy nhiên, những ảnh hưởng của đại dịch COVID-19 sẽ kìm hãm sự giãn nở của thị trường Smart Home nói chung. Thực tế này xuất phát từ tâm lý e ngại và dè dặt hơn của người tiêu dùng trong quyết định chi trả cho những sản phẩm phần nào còn mới mẻ và cần giáo dục thị trường như Smart home.

2.3 Cấu trúc tổng quan và danh sách thiết bị

Trong khuôn khổ của project này, ta giả sử ngôi nhà của người dùng bao gồm các loại căn phòng dưới đây, trong mỗi căn phòng sẽ có các thiết bị thông minh với nhiệm vụ và chức năng khác nhau:



Hình 3: Cấu trúc tổng quan của ngôi nhà.

- Danh sách thiết bị và chức năng trong phòng ngủ:

Tên thiết bị	Mô tả chức năng dự kiến
Light	Bóng đèn trong phòng ngủ, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Fan	Quạt trong phòng ngủ, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Air Conditioner	Điều hòa trong phòng ngủ, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Table Light	Đèn ngủ trong phòng ngủ, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Tivi	Tivi trong phòng ngủ, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Cảm biến nhiệt độ	Đo nhiệt độ trong phòng ngủ.
Cảm biến khói	Phát hiện có khói xuất hiện trong phòng, cảnh báo qua loa cho người dùng.

Hình 4: Danh sách thiết bị và chức năng trong phòng ngủ.

- Danh sách thiết bị và chức năng trong phòng khách:

Tên thiết bị	Mô tả chức năng dự kiến
Light	Bóng đèn trong phòng khách, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Fan	Quạt trong phòng khách, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Air Conditioner	Điều hòa trong phòng khách, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Speaker	Bật/tắt và phát bản nhạc theo lựa chọn của người dùng.
Tivi	Tivi trong phòng khách, thông qua ứng dụng người dùng có thể xem trạng thái hoạt động của nó, điều chỉnh bật/tắt thích hợp.
Cảm biến nhiệt độ	Đo nhiệt độ trong phòng khách.
Cảm biến khói	Phát hiện có khói xuất hiện trong phòng, cảnh báo qua loa cho người dùng.

Hình 5: Danh sách thiết bị và chức năng trong phòng khách.

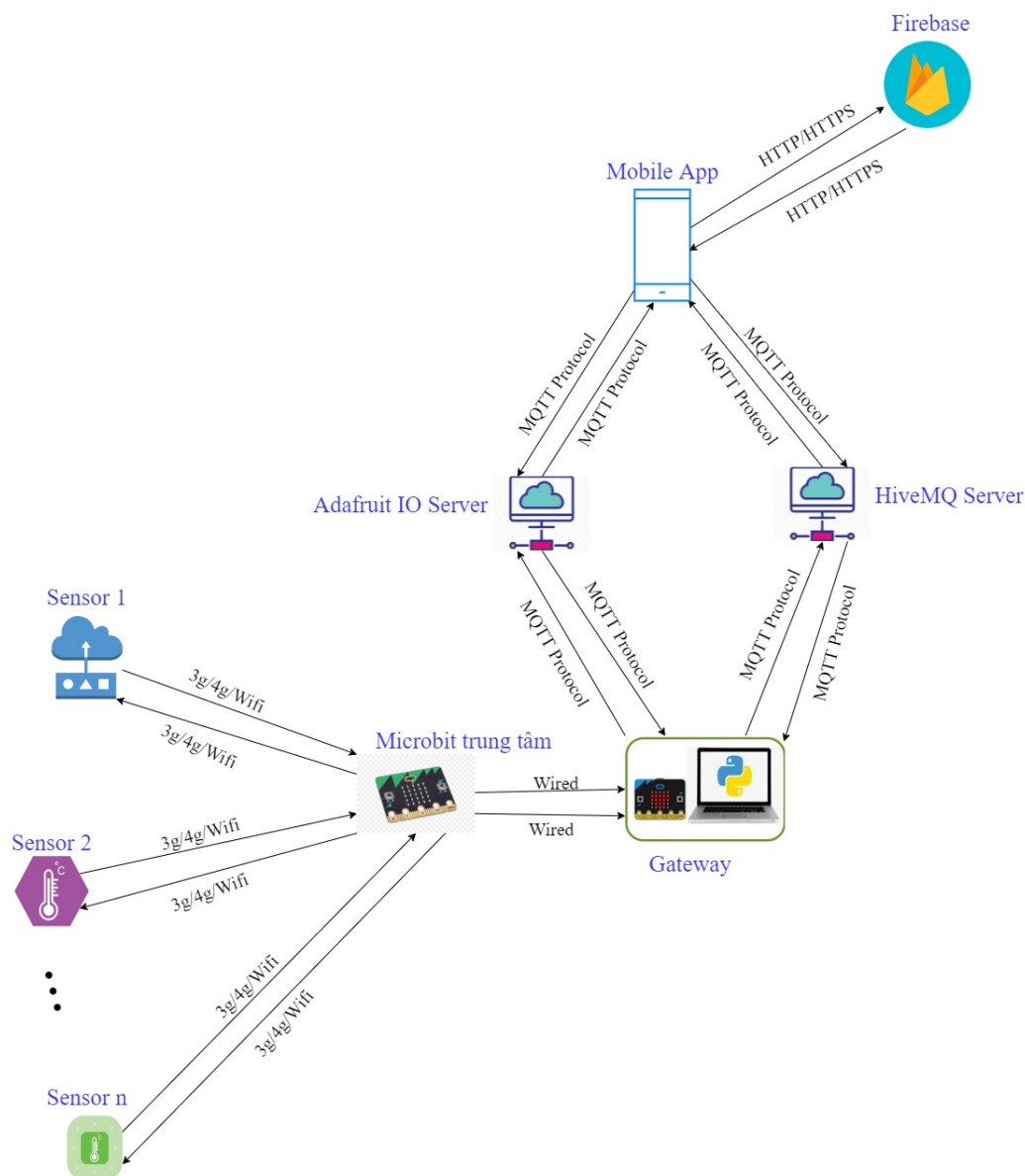
- Danh sách thiết bị và chức năng trong phòng ăn và nhà bếp nhìn chung giống như phòng ngủ và phòng khách.

- Danh sách thiết bị và chức năng phía trước ngôi nhà:

Tên thiết bị	Mô tả chức năng dự kiến
Cảm biến ánh sáng	Đo thông tin về cường độ ánh sáng phía trước ngôi nhà.
Camera	Chụp hình ảnh phía trước ngôi nhà sau một khoảng thời gian nhất định và gửi về chương trình AI được tích hợp trong gateway để xử lý, mục đích của chương trình này là phân tích có người xuất hiện trước ngôi nhà hay không.
Light	Có 2 chế độ: chế độ auto và chế độ bình thường. Trong chế độ bình thường, người dùng có thể bật/tắt bóng đèn tùy ý thông qua mobile app. Trong chế độ auto, tùy thuộc vào cường độ ánh sáng và có người trước ngôi nhà hay không mà bóng đèn sẽ tự động bật/tắt.

Hình 6: Danh sách thiết bị và chức năng phía trước ngôi nhà

2.4 Xây dựng sơ đồ kiến trúc hệ thống



Hình 7: Sơ đồ kiến trúc hệ thống

Ngoài những thành phần đã được hướng dẫn trong khóa học, ta sẽ sử dụng thêm các thành phần khác để bổ sung những tính năng mới cho hệ thống.





2.4.1 HiveMQ Server

HiveMQ là một đội ngũ phát triển các product, cloud, solutions cho các mảng Big Data, IoT,...

HiveMQ Cloud là một trong những sản phẩm của đội ngũ này.

HiveMQ Cloud có nhiều phiên bản khác nhau từ miễn phí đến có phí.

Phiên bản miễn phí nhìn chung có các tính năng giống với Adafruit IO Server.

	FREE	PAY AS YOU GO	BUSINESS	DEDICATED
Number of devices	Up to 100	Up to 1,000	Above 1,000	Unlimited
Data limits	Up to 10 GB	Up to 100 GB	Above 100 GB	Contact Us
HiveMQ Control Center	✖	✖		
Infrastructure	Shared	Shared	Shared	Dedicated
Private Network (VPC Peering)	✖	✖	✖	
SLA	No SLA	99.5%	99.9%	99.99%
Support	Community Forum	Basic	Basic	24/7
Price	Free	\$ 0.10 / device \$ 0.15 / GB  Price Calculator	For pricing please	For pricing please

Hình 8: Các phiên bản hiện tại của HiveMQ

2.4.2 Firebase

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu. Nói một cách khác, nó là một nền tảng để phát triển ứng dụng di động và trang web, bao gồm các API đơn giản và mạnh mẽ mà không cần backend hay server.



Hình 9: Logo của Firebase

Firebase có nhiều dịch vụ khác nhau như: Authentication, Firestore Database, Real-time Database, Storage,... Trong project này, ta sẽ sử dụng 2 dịch vụ của Firebase là Authentication và Firestore Database:

- **Authentication:** Hoạt động nổi bật của Firebase là xây dựng các bước xác thực người dùng bằng Email, Facebook, Twitter, GitHub, Google. Đồng thời cũng xác thực nặc danh cho các ứng dụng. Hoạt động xác thực có thể giúp thông tin cá nhân của người sử dụng được an toàn và đảm bảo không bị đánh cắp tài khoản. Ta sẽ sử dụng Authentication để quản lý tài khoản người dùng, đăng nhập bằng tài khoản Google.
- **Firestore Database:** là cơ sở dữ liệu mới của Firebase phát triển dành cho ứng dụng di động, web. Giống như Firebase Realtime Database, nó giúp dữ liệu đồng bộ hóa trên các ứng dụng Client thông qua việc đăng ký Realtime và cung cấp hỗ trợ ngoại tuyến cho thiết bị di động và web. Ta sẽ sử dụng Firestore để lưu tên đầy đủ của người dùng khi người dùng đăng ký tài khoản, lưu trữ trạng thái hoạt động của các thiết bị.

3 Hiện thực kết nối với Sensors

3.1 Xử lý dữ liệu từ sensors

Dữ liệu được gửi từ các sensors đến gateway sẽ có định dạng dữ liệu như sau:

!	ID	Sensor name	Value	#
---	----	-------------	-------	---

Hình 10: Định dạng dữ liệu được gửi về gateway.

Trong đó:

- **!**, **#**: hai kí tự để đánh dấu lần lượt các vị trí bắt đầu và kết thúc của dữ liệu.
- **Sensor name**: tên của sensor, mô tả sensor đó có chức năng gì, được sử dụng ở phòng nào,...
- **Value**: giá trị sensor gửi về gateway.

Bên phía gateway sẽ liên tục đọc dữ liệu được gửi về từ các cảm biến, cụ thể như sau:

Listing 1: Đọc dữ liệu từ mạch Microbit trung tâm gửi về

```
1 mess = ""
2 def readSerial():
3     bytesToRead = ser.inWaiting()
4     if (bytesToRead > 0):
5         global mess
6         mess = mess + ser.read(bytesToRead).decode("UTF-8")
7         while ("#" in mess) and ("!" in mess):
8             start = mess.find("!")
9             end = mess.find("#")
10            processData(mess[start:end + 1])
11            if (end == len(mess)):
12                mess = ""
13            else:
14                mess = mess[end+1:]
```

Khi nhận được đủ một frame data ở dòng 7 (có 2 kí tự bắt đầu và kết thúc frame), gateway sẽ rút trích đoạn frame đó và gửi cho hàm *processData* xử lý, phần còn lại của hàm *readSerial* sẽ tiếp tục nhận và xử lý những đoạn frame tiếp theo mà các sensor gửi về. Chức năng của hàm *processData* sẽ được giới thiệu ở phần tiếp theo.

3.2 Mở rộng chức năng của hàm *processData*

Listing 2: Hàm processData

```
1 def processData(data):
2     data = data.replace("!", "")
3     data = data.replace("#", "")
4     splitData = data.split(":")
5     print(splitData)
6     if splitData[1] == "BEDROOMTEMP":
7         g.buffer.append({"feed_id": "bedroom-temp", "payload": ←
splitData[2], "numOfAttempt": 0})
8     elif splitData[1] == "BEDROOMAIR":
9         g.buffer.append({"feed_id": "bedroom-air", "payload": ←
splitData[2], "numOfAttempt": 0})
10    elif splitData[1] == "BEDROOMFAN":
11        g.buffer.append({"feed_id": "bedroom-fan", "payload": ←
splitData[2], "numOfAttempt": 0})
12    elif splitData[1] == "BEDROOMLIGHT":
13        g.buffer.append({"feed_id": "bedroom-light", "payload": ←
splitData[2], "numOfAttempt": 0})
14    elif splitData[1] == "BEDROOMSMOKE":
15        g.buffer.append({"feed_id": "bedroom-smoke", "payload": ←
splitData[2], "numOfAttempt": 0})
16    elif splitData[1] == "BEDROOMSPEAKER":
17        g.buffer.append({"feed_id": "bedroom-speaker", "payload": ←
: splitData[2], "numOfAttempt": 0})
18    elif splitData[1] == "BEDROOMTIVI":
19        g.buffer.append({"feed_id": "bedroom-tivi", "payload": ←
splitData[2], "numOfAttempt": 0})
20    elif splitData[1] == "BEDROOMTLIGHT":
21        g.buffer.append({"feed_id": "bedroom-tlight", "payload": ←
splitData[2], "numOfAttempt": 0})
22    elif splitData[1] == "HALLINFRARED":
23        g.buffer.append({"feed_id": "hall-infrared", "payload": ←
splitData[2], "numOfAttempt": 0})
24    elif splitData[1] == "HALLLIGHT":
25        g.buffer.append({"feed_id": "hall-light", "payload": ←
splitData[2], "numOfAttempt": 0})
26    else:
27        g.buffer.append({"feed_id": "no-exist", "payload": ←
splitData[2], "numOfAttempt": 0})
```


Phía gateway sẽ có một message queue để lưu trữ các frame data đã qua xử lý. Những message nằm trong message queue này sẽ được lần lượt gửi từ gateway lên server, nó làm tăng khả năng sẵn sàng, phản hồi và kiểm soát lỗi của gateway, sẽ được giới thiệu kĩ hơn ở phần tiếp theo.

Trong hàm *processData*, tùy vào sensor name mà message sẽ có *feed_id* khác nhau, *payload* sẽ bằng với *value* của frame data, đồng thời còn có thêm trường *numOfAttempt* được sử dụng cho tính năng kiểm soát lỗi của gateway.

4 Hiện thực Gateway

Gateway có nhiệm vụ luân chuyển dữ liệu giữa sensor và server, là một trong những thành phần quan trọng nhất của một hệ thống IoT. Trong project này, những chức năng của gateway cụ thể như sau:

4.1 UART received data

Listing 3: Nhận và đọc dữ liệu thông qua serial

```
1 def readSerial():
2     bytesToRead = ser.inWaiting()
3     if (bytesToRead > 0):
4         global mess
5         mess = mess + ser.read(bytesToRead).decode("UTF-8")
```

Khi nhận được một byte từ serial, gateway sẽ decode byte đó về định dạng *UTF-8* và cộng dồn vào biến toàn cục mess.

4.2 Error Control

Kiểm soát lỗi là một trong những nhiệm vụ quan trọng nhất của gateway, nó giúp làm tăng độ tin cậy của hệ thống IoT và giúp các thành phần trong hệ thống tương tác với nhau một cách có hiệu quả hơn. Để thực hiện nhiệm vụ này, gateway duy trì một số thông số như sau:

Listing 4: Những thông số của gateway

```
1 TIMEOUT_MS = 2000
2 LONG_SLEEP_MS = 60000
3
4 TIME_TO_CALL_AI = 30000
5
6 MAX_NUM_OF_TOTAL_ATTEMPTS = 5
7 MAX_NUM_OF_PARTIAL_ATTEMPTS = 3
8 MAX_NUM_OF_REMOVE_ATTEMPTS = 3
9 lastPayload = ''
10 lastFeedId = ''
11 lastSentOK = True
12 numOfAttempts = 0
13 numOfFailed = 0
14 waitForSubscribe = True
15
```

16 `buffer = []`

Ý nghĩa của các thông số:

- **TIMEOUT_MS**: khoảng thời gian để chờ phản hồi từ phía server, nếu nhận được phản hồi từ phía server trong khoảng thời gian này tức message đã được publish thành công, ngược lại message được đánh dấu là gửi thất bại tạm thời và sẽ được gửi lại hoặc xóa bỏ.
- **LONG_SLEEP_MS**: gateway sẽ tạm dừng hoạt động trong khoảng thời gian này nếu phát hiện có quá nhiều message không thể publish thành công.
- **TIME_TO_CALL_AI**: sẽ có một chương trình AI được tích hợp trong gateway. Chương trình AI này nhận đầu vào là những hình ảnh từ camera (sử dụng webcam làm minh họa) và cho ra kết quả là phát hiện có người hay không. Khoảng thời gian này là khoảng thời gian mà chương trình AI sẽ đọc dữ liệu từ camera và xử lý.
- **MAX_NUM_OF_TOTAL_ATTEMPTS**: số lượng message gửi thất bại tối đa cho phép, vượt quá con số này thì gateway sẽ tạm thời ngừng hoạt động trong khoảng thời gian là **LONG_SLEEP_MS**
- **MAX_NUM_OF_PARTIAL_ATTEMPTS**: khi một message được đánh dấu là gửi thất bại tạm thời, nó sẽ được gửi lại với số lần tối đa quy định bởi thông số này, nếu sau số lần gửi lại đó message vẫn gửi không thành công, nó sẽ bị di chuyển đến cuối message queue và đợi đến lượt gửi tiếp theo.
- **MAX_NUM_OF_REMOVE_ATTEMPTS**: sau số lần bị di chuyển xuống cuối hàng đợi và chờ đến lượt tiếp theo để gửi lại mà vẫn gửi không thành công, message sẽ bị xóa khỏi queue và được đánh dấu là gửi thất bại hoàn toàn, lúc này gateway sẽ gửi cảnh báo ngược trở lại sensor của chính message, bên phía sensor sẽ có những xử lý phù hợp khi nhận được cảnh báo này của gateway.
- **lastPayload, lastFeedId, lastSentOK**: là những biến để lưu trữ payload, feed id cuối cùng được gửi và payload đó có được gửi thành công hay không.

Như đã giới thiệu ở phần trước, gateway sẽ giữ cho mình một message queue, có chức năng lưu trữ các frame data đã qua xử lý, gateway sẽ truy xuất những message nằm trong queue này để publish lên phía server. Khi gateway được khởi động, message queue sẽ rỗng và luôn ở trong trạng thái sẵn sàng nhận và xử lý khi có bất kỳ frame data nào được chuyển đến. Khi frame data được chuyển đến và xử lý thành message, message này sẽ được đưa vào cuối queue, queue này hoạt động theo cơ chế *First Come First Serve* kết hợp với *Round Robin*. Khi gateway nhận thấy có message nằm trong queue, nó sẽ lập tức lấy message đầu tiên ra để publish lên server, đồng thời khởi tạo một số thông số để thực hiện nhiệm vụ kiểm soát lỗi.

Listing 5: Gateway sẽ lấy những message nằm trong queue để publish lên Server

```
1 if g.lastSentOK and g.buffer.__len__() > 0:
2     g.lastPayload = g.buffer[0].get("payload")
3     g.lastFeedId = g.buffer[0].get("feed_id")
4     g.lastSentOK = False
5     ackTimer = g.TIMEOUT_MS
6     send(client, g.lastFeedId, g.lastPayload)
```

Nếu trong vòng **TIMEOUT_MS** giây mà nhận được ACK phản hồi từ server, message sẽ được xóa khỏi queue và nhường chỗ cho những message khác; ngược lại, message sẽ được gửi lại **MAX_NUM_OF_PARTIAL_ATTEMPTS** lần, nếu vẫn gửi không thành công, message sẽ được di chuyển xuống cuối hàng đợi và chờ đợt gửi lại tiếp theo (ý tưởng của giải thuật *Round Robin*).

Listing 6: Gửi lại message nếu sau 2s chưa nhận được ACK từ Server

```
1 if not g.lastSentOK and ackTimer == 0 and not goToSleep and g.↵
    buffer.__len__() > 0:
2     ackTimer = g.TIMEOUT_MS
3     send(client, g.lastFeedId, g.lastPayload, True)
4     g.numOfAttempts += 1
5     print("Attempt resent " + g.lastFeedId + ": " + str(g.↵
        numOfAttempts))
```

Nếu sau **MAX_NUM_OF_PARTIAL_ATTEMPTS** gửi lại không thành công, message sẽ bị di chuyển xuống cuối queue và nhường chỗ cho những message khác. Nếu sau số lần quy định bị di chuyển xuống cuối queue mà vẫn không thành công, message sẽ bị xóa khỏi queue:

Listing 7: Gateway sẽ quyết định nên di chuyển message xuống cuối queue hay xóa nó

```
1 if g.numOfAttempts >= g.MAX_NUM_OF_PARTIAL_ATTEMPTS and ackTimer↵
```

```
    == 0 and g.buffer.__len__() > 0:
2   if g.buffer[0]["numOfAttempt"] >= g.↵
    MAX_NUM_OF_REMOVE_ATTEMPTS:
3       print("*****")
4       ser.write((g.lastFeedId + ":" + str(g.lastPayload) + ":"↵
    + "ERROR" + "#").encode())
5       print("Attempt resent with topic " + g.lastFeedId + " ↵
    failed, remove payload !!")
6       g.buffer.pop(0)
7       g.lastSentOK = True
8       g.numOfFailed += 1
9   else:
10      g.buffer[0]["numOfAttempt"] += 1
11      print("*****")
12      print("Move payload with topic " + g.lastFeedId + " to ↵
    end of buffer !!")
13      g.buffer.append(g.buffer[0])
14      g.buffer.pop(0)
15      g.numOfAttempts = 0
16      g.lastSentOK = True
```

Nếu số lượng message bị xóa khỏi hàng đợi do gửi không thành công vượt quá số lượng cho phép, gateway sẽ đi vào trạng thái ngủ trong khoảng thời gian **LONG_SLEEP_MS** và sau đó tiếp tục vòng đời của mình.

Listing 8: Gateway sẽ đi vào trạng thái ngủ nếu có nhiều message gửi không thành công

```
1  if g.numOfFailed >= g.MAX_NUM_OF_TOTAL_ATTEMPTS and ackTimer == ↵
    0 and g.buffer.__len__() > 0:
2      print(str(g.numOfAttempts) + " attempts failed. Stop system ↵
    for 60s then try again.")
3      longSleepTimer = g.LONG_SLEEP_MS
4      g.numOfFailed = 0
5      goToSleep = True
```

4.3 Data uploading process

Message sẽ được gateway publish lên Server với hàm *send* được hiện thực như sau:

Listing 9: Hàm gửi dữ liệu lên Server MQTT

```
1 def send(client, topic, payload, resend = False):
2     print("-----")
3     client.publish(topic, payload)
4     if resend:
5         print("Re-sent payload: " + str(payload) + " of topic: " + ↵
+ topic)
6     else:
7         print("Sent payload: " + str(payload) + " of topic: " + ↵
topic)
```

4.4 Data received process

Gateway cũng subscribe vào một số topic để giúp người dùng có thể điều khiển những thiết bị từ xa, đồng thời cũng giúp gateway hiện thực tính năng kiểm soát lỗi, điều này được thực hiện bởi hàm *message*

Listing 10: Hàm gửi dữ liệu lên Server MQTT

```
1 def message(client, feed_id, payload):
2     print("ACK: " + payload)
3     if g.lastFeedId == feed_id and g.lastPayload == payload:
4         print("success")
5         g.lastSentOK = True
6         g.numOfFailed = 0
7         g.numOfAttempts = 0
8         g.buffer.pop(0)
9     if g.lastFeedId != feed_id:
10        ser.write((feed_id + ":" + str(payload) + "#").encode())
```

4.5 Model AI Detect Person

Gateway có tích hợp thêm một Model AI, có nhiệm vụ đọc dữ liệu từ camera trước ngôi nhà và phát hiện có người hay không. Chi tiết hiện thực Model này như sau:

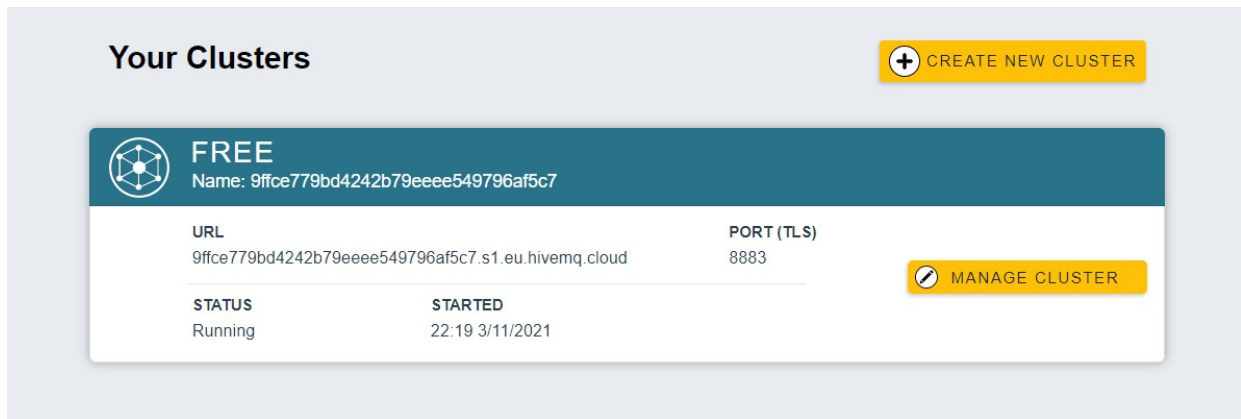
Listing 11: Model AI Detect Person

```
1 from keras.models import load_model
2 from PIL import Image, ImageOps
3 import numpy as np
4 import cv2
5 import globals as g
6
7 # Load the model
8 model = load_model('keras_model.h5')
9 cam = cv2.VideoCapture(0)
10
11 def detectPerson():
12     ret, frame = cam.read()
13     cv2.imwrite("img_detect.png", frame)
14     data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
15     image = Image.open('img_detect.png')
16     size = (224, 224)
17     image = ImageOps.fit(image, size, Image.ANTIALIAS)
18     image_array = np.asarray(image)
19     normalized_image_array = (image_array.astype(np.float32) / ←
127.0) - 1
20     data[0] = normalized_image_array
21     # run the inference
22     prediction = model.predict(data)
23     if prediction[0][0] >= prediction[0][1]:
24         print("AI: Nobody !!")
25         g.buffer.append({"feed_id": "hall-infrared", "payload": ←
"0", "numOfAttempt": 0})
26     else:
27         print("AI: Detect person !!")
28         g.buffer.append({"feed_id": "hall-infrared", "payload": ←
"1", "numOfAttempt": 0})
```

5 Định cấu hình cho Server MQTT

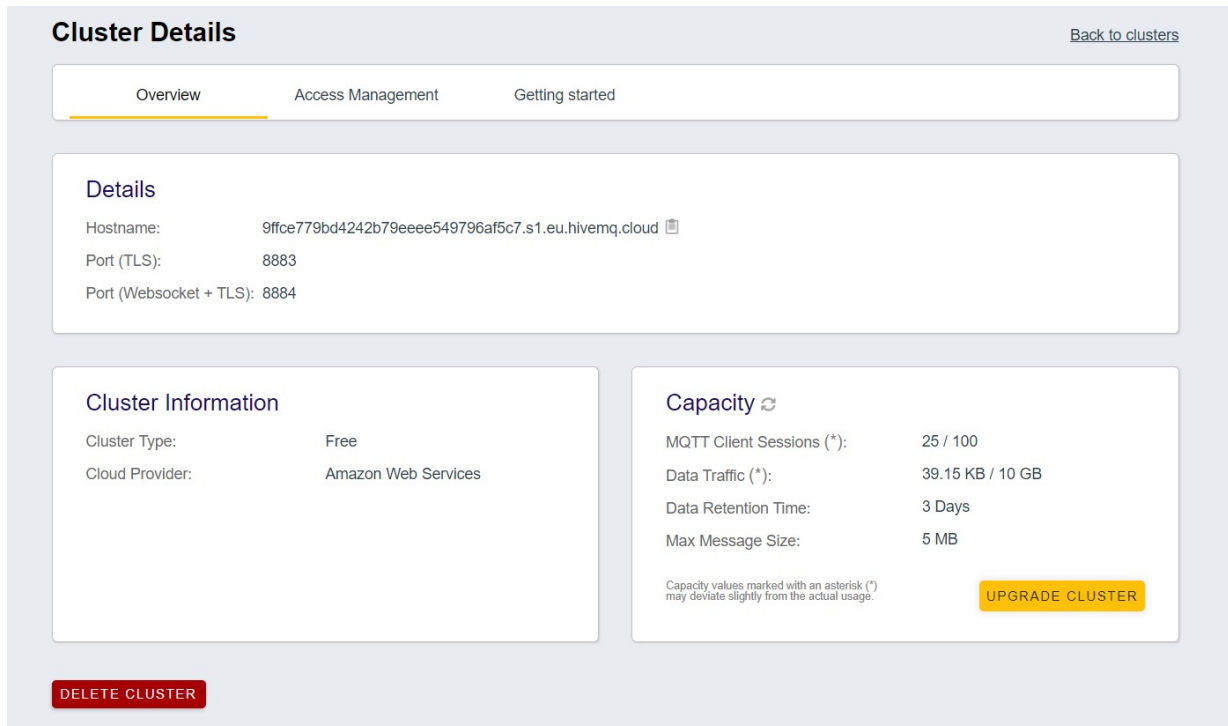
Project sử dụng 2 Server MQTT là Adafruit IO và HiveMQ Server. Trong đó việc cấu hình cho một server mới như HiveMQ ban đầu gặp nhiều khó khăn do chưa hiểu rõ kiến trúc của nó. Việc cấu hình cho server này được trình bày trong những hình dưới đây:

- Ta tạo một host mới trên server, thông tin về Hostname và Port được hiển thị như hình bên dưới



Hình 11: Tạo một host mới trên server

- Sau khi đã tạo host thành công, ta truy cập vào host này. Ở màn hình Overview sẽ hiển thị thông tin chi tiết về host như: hostname, port (TLS) để các ứng dụng client có thể publish hoặc subscribe dữ liệu, port (Websocket + TLS) để người dùng thông qua Web UI xem thông tin chi tiết về dữ liệu của các topic có trên host.



Cluster Details [Back to clusters](#)

Overview Access Management Getting started

Details

Hostname: 9ffce779bd4242b79e549796af5c7.s1.eu.hivemq.cloud ⓘ

Port (TLS): 8883

Port (Websocket + TLS): 8884

Cluster Information

Cluster Type: Free

Cloud Provider: Amazon Web Services

Capacity ⓘ

MQTT Client Sessions (*): 25 / 100

Data Traffic (*): 39.15 KB / 10 GB

Data Retention Time: 3 Days

Max Message Size: 5 MB

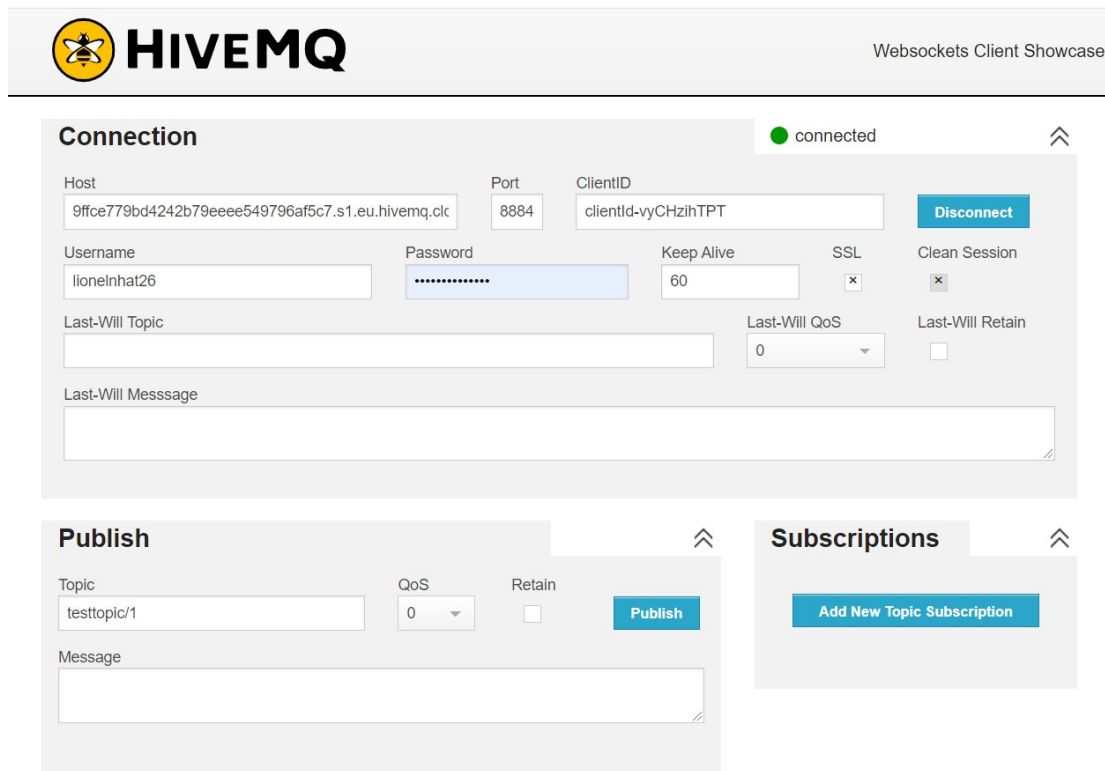
Capacity values marked with an asterisk (*) may deviate slightly from the actual usage.

[UPGRADE CLUSTER](#)

[DELETE CLUSTER](#)

Hình 12: Thông tin chi tiết về host đã tạo

- Để xem thông tin chi tiết về dữ liệu của các topic có trên host, ta sử dụng Web UI để truy cập vào host này. Cần có các thông tin như username, password, port,...



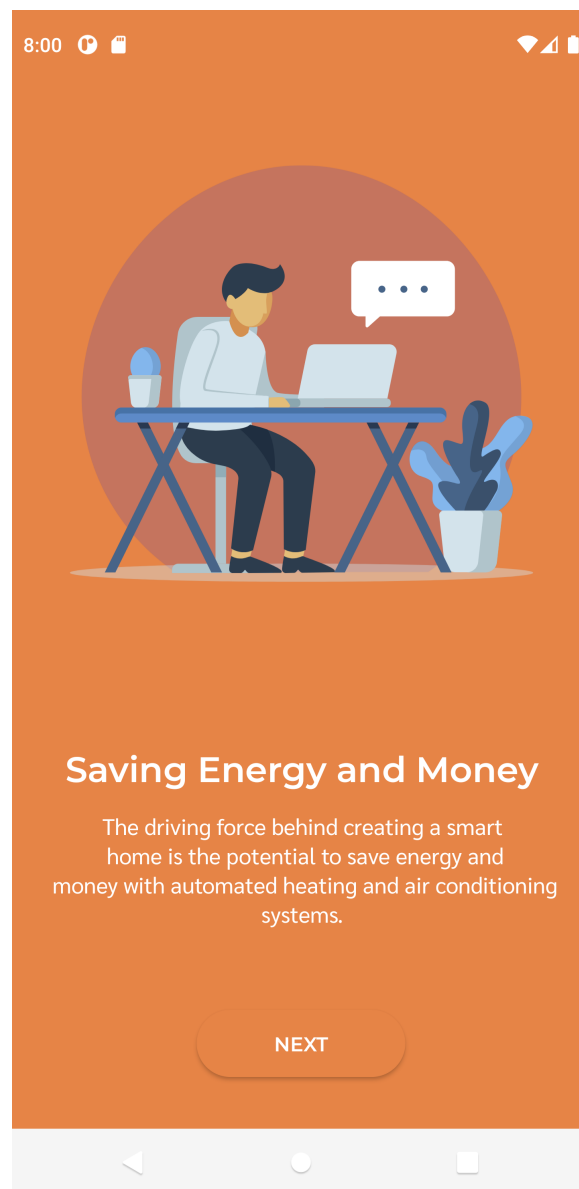
The image shows the Hivemq Websockets Client Showcase interface. At the top, there's a header with the Hivemq logo and the text "Websockets Client Showcase". Below this, there's a "Connection" section with a status indicator "connected" and a "Disconnect" button. The connection details include Host (9ffce779bd4242b79eeee549796af5c7.s1.eu.hivemq.clc), Port (8884), ClientID (clientId-vyCHzihTPT), Username (lioneinhathat26), Password (masked), Keep Alive (60), SSL (checked), Clean Session (checked), Last-Will Topic, Last-Will QoS (0), Last-Will Retain (checked), and Last-Will Message. Below the connection section, there's a "Publish" section with fields for Topic (testtopic/1), QoS (0), Retain (unchecked), and a "Publish" button. There's also a "Subscriptions" section with an "Add New Topic Subscription" button.

Hình 13: Truy cập để xem dữ liệu có trên host

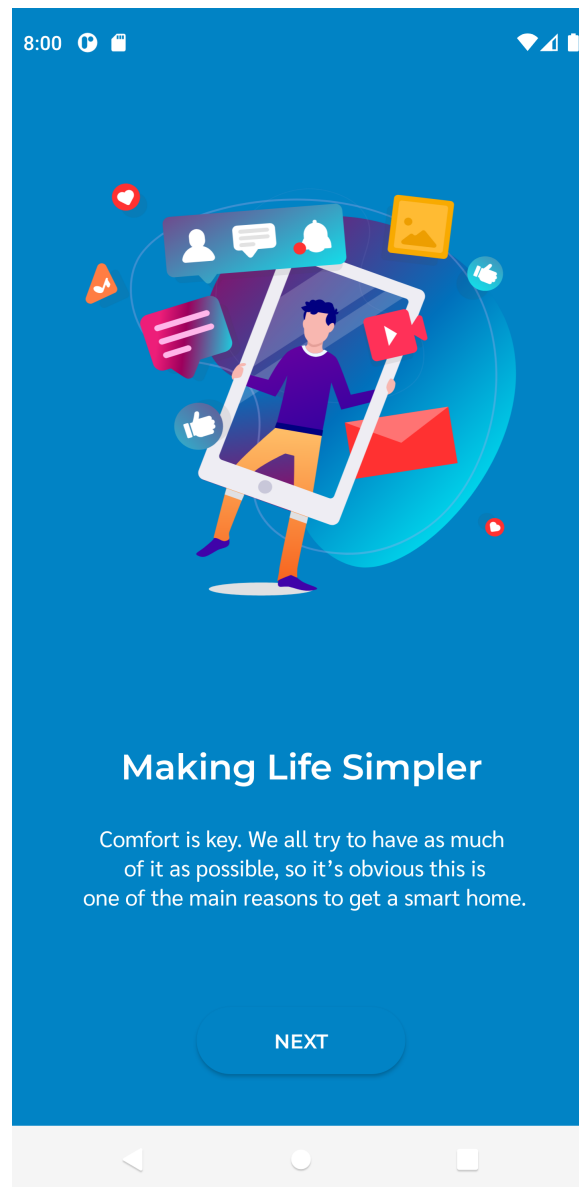
6 Xây dựng ứng dụng Mobile

Giao diện Mobile App bao gồm các màn hình chính sau đây:

- **Onboarding Screen:** đây là những màn hình giới thiệu các lợi ích của hệ thống IoT khi người dùng lần đầu tiên sử dụng ứng dụng, sau này khi người dùng đã tạo tài khoản, những màn hình này sẽ không xuất hiện nữa.

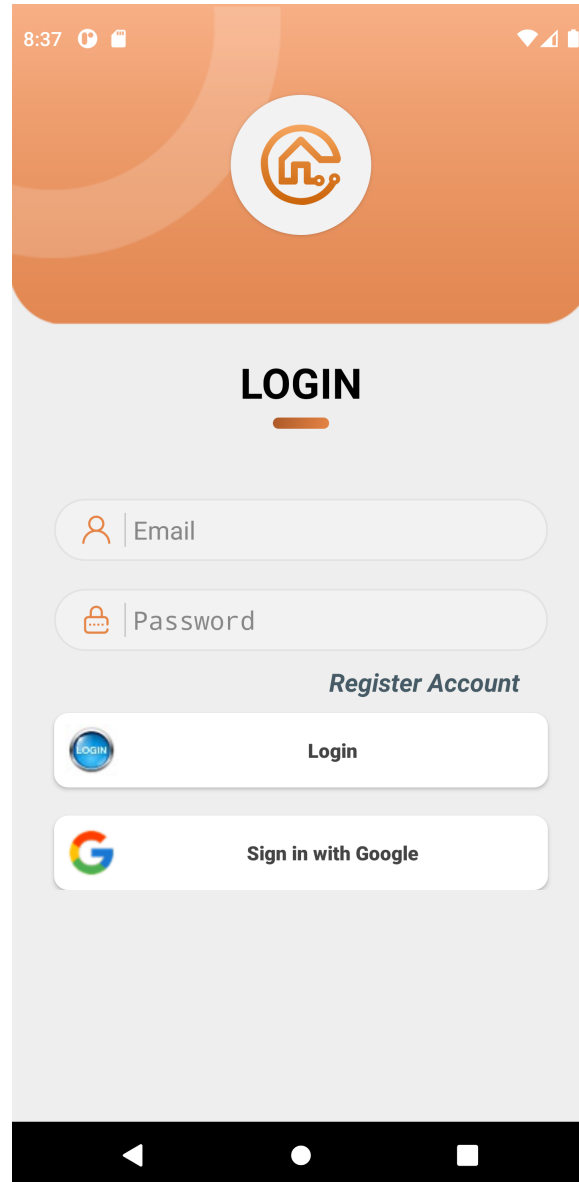


Hình 14: Màn hình Onboarding 1.

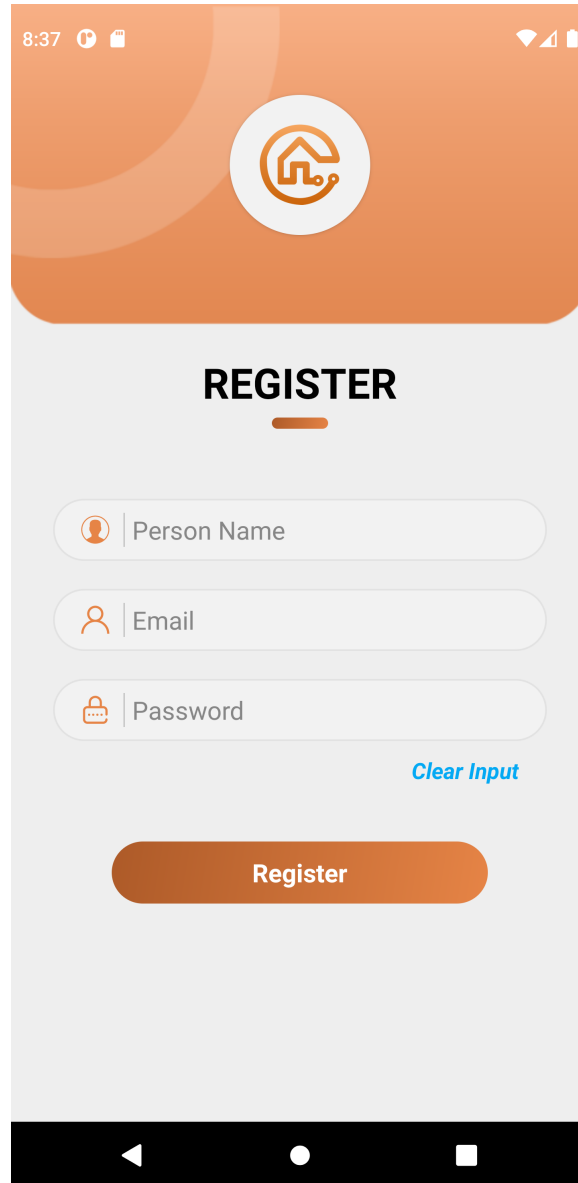


Hình 15: Màn hình Onboarding 2.

- **Signin/Signup Screen:** đây là màn hình quản lý việc đăng nhập và đăng ký tài khoản của người dùng.



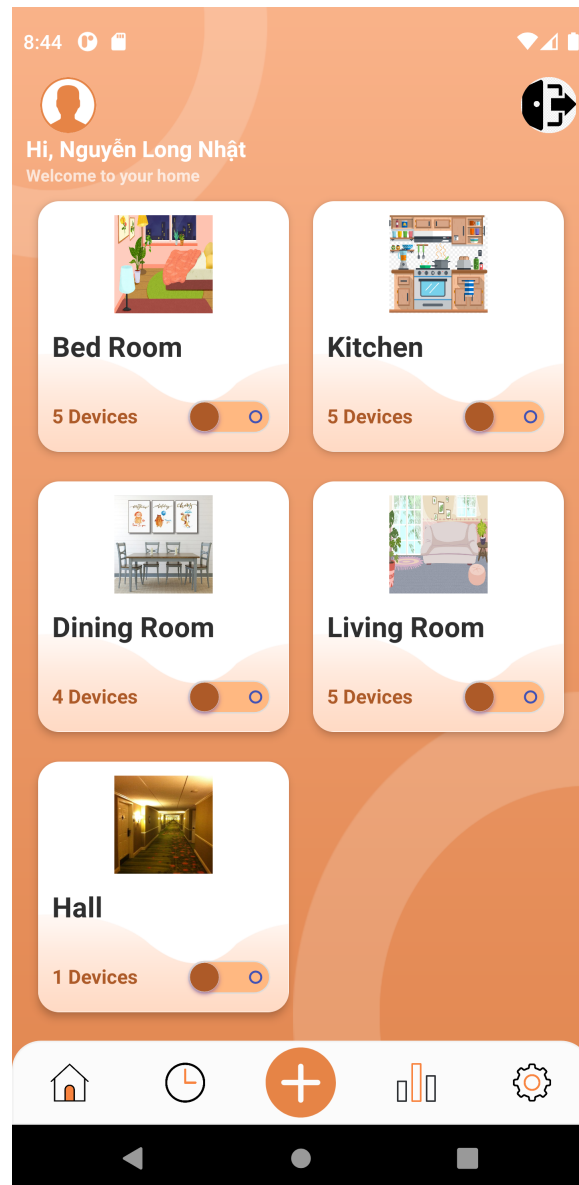
Hình 16: Màn hình đăng nhập.



The image shows a mobile application interface for registration. At the top, there is a status bar with the time 8:37 and icons for signal, Wi-Fi, and battery. Below the status bar is a header with an orange background and a white circular icon containing a house with a key. The main content area has a light gray background and is titled "REGISTER" in bold black text. Below the title are three input fields: "Person Name" with a person icon, "Email" with an email icon, and "Password" with a lock icon. To the right of the "Password" field is a blue link labeled "Clear Input". At the bottom of the form is a large orange button labeled "Register". The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a circle, and a square.

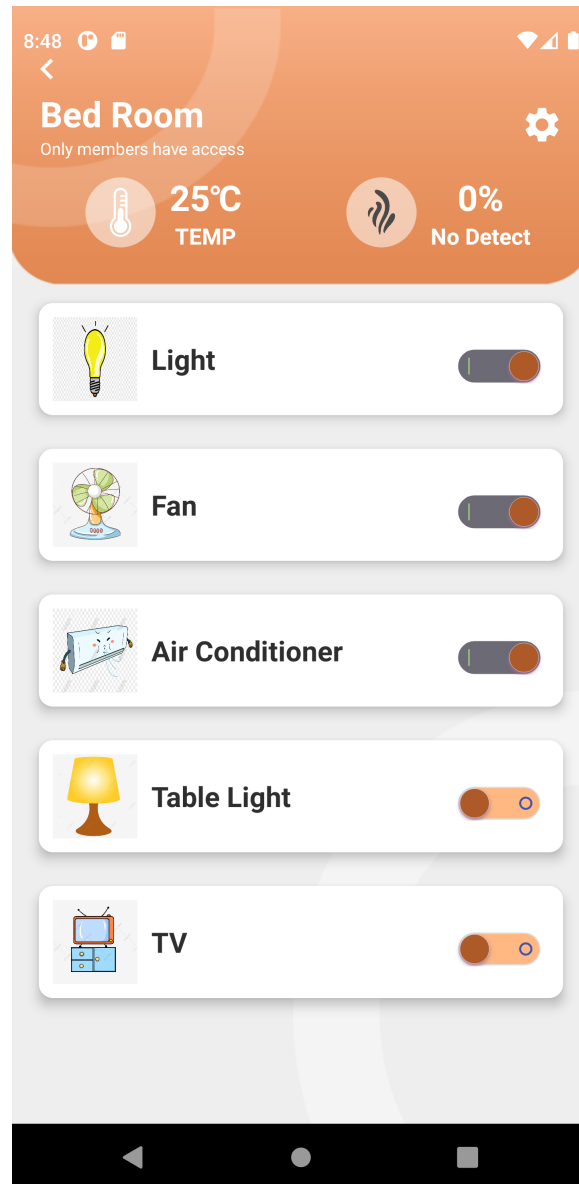
Hình 17: Màn hình đăng kí.

- **Main Screen:** đây là màn hình giao diện chính dành cho người dùng. Ở giao diện này, người dùng sẽ nhìn thấy những căn phòng trong ngôi nhà của mình và số lượng thiết bị trong mỗi căn phòng đó. Để xem thông tin chi tiết và quản lý từng thiết bị trong mỗi phòng khác nhau, người dùng sẽ click vào các phòng này. Ngoài ra trên màn hình cũng hiển thị tên người dùng và button Logout.

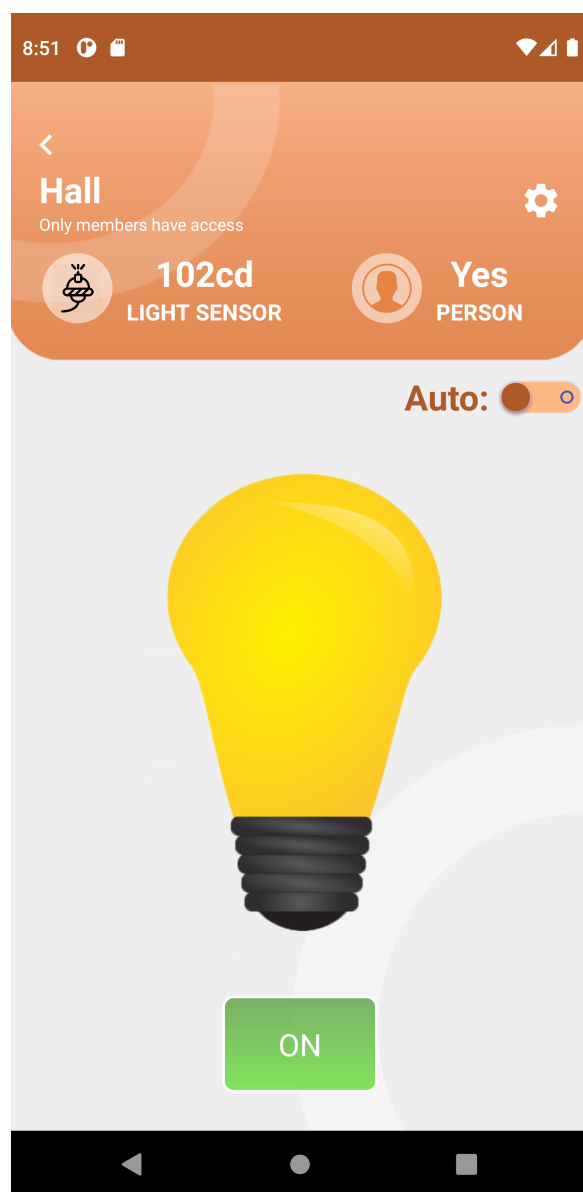


Hình 18: Màn hình giao diện chính của ứng dụng.

- **Detail Screen:** đây là màn hình giao diện các thành phần chi tiết của mỗi căn phòng. Ở màn hình này, người dùng sẽ nhìn thấy thông tin từ các cảm biến gửi về như nhiệt độ, khí gas,... Người dùng cũng có thể điều khiển các thiết bị từ xa, xem trạng thái hoạt động của các thiết bị, điều chỉnh mode,...



Hình 19: Màn hình chi tiết chức năng trong phòng ngủ.



Hình 20: Màn hình chi tiết chức năng phía trước ngôi nhà.

6.1 Chức năng bổ sung 1: Sử dụng Firebase cho ứng dụng

Để lưu thông tin của người dùng như email, password, username và các trạng thái hoạt động của thiết bị, ứng dụng sử dụng thêm Firestore Database của Firebase bằng cách sử dụng thêm các *dependencies* sau:

Listing 12: Dependencies để sử dụng Firebase

```
1 implementation platform('com.google.firebase:firebase-bom:28.4.1')  
2 implementation 'com.google.firebase:firebase-firestore'
```

6.2 Chức năng bổ sung 2: Đăng nhập với tài khoản Google

Ngoài lựa chọn tạo cho mình một tài khoản riêng biệt, người dùng có thể đăng nhập nhanh hơn bằng tài khoản Google. Để hiện thực chức năng này, ứng dụng sử dụng thêm các *dependencies* sau:

Listing 13: Dependencies để đăng nhập với tài khoản Google

```
1 implementation 'com.firebaseui:firebase-ui-auth:7.2.0'  
2 implementation 'com.google.firebase:firebase-auth'  
3 implementation 'com.google.android.gms:play-services-auth:19.2.0'
```

6.3 Chức năng bổ sung 3: Sử dụng thêm Server MQTT khác: HiveMQ

Ngoài Adafruit IO là server MQTT chính mà ứng dụng nói riêng và hệ thống nói chung sử dụng, ứng dụng còn sử dụng thêm một server MQTT khác là HiveMQ, để tương tác với server này, chúng ta thêm *dependencies* sau:

Listing 14: Dependencies để sử dụng HiveMQ Server

```
1 implementation 'com.hivemq:hivemq-mqtt-client:1.2.1'
```

7 Kết luận

- Thông qua hiện thực project đã giúp nhóm hiểu hơn về các thành phần cơ bản của một hệ thống IoT nói chung và mô hình Smart Home nói riêng, hiểu được cách các thành phần cơ bản đó tương tác qua lại lẫn nhau và là nền tảng để xây dựng các hệ thống IoT lớn hơn sau này.
- Không chỉ dừng lại ở đó, project còn giúp nhóm hiểu và vận dụng được tính năng kiểm soát lỗi cho Gateway, dựa trên ý tưởng của cơ chế *First Come First Serve* và giải thuật *Round Robin*, từ đó làm cho Gateway tăng khả năng sẵn sàng, linh hoạt và phản hồi.
- Ngoài Adafruit IO Server là server chính của hệ thống, nhóm cũng đã mở rộng tìm hiểu và sử dụng thêm một server MQTT khác là HiveMQ, có nhiều tính năng tương tự như Adafruit IO Server ở phiên bản miễn phí. Trong tương lai với những hệ thống rộng và thực tế hơn, hoàn toàn có thể sử dụng HiveMQ Server phiên bản trả phí, những IoT Platform khác như Thingsboard,...
- Dashboard cũng là một thành phần không kém phần quan trọng trong các hệ thống IoT. Thông qua project, nhóm đã xây dựng được một Mobile App có đầy đủ tính năng của một Dashboard IoT. Thông qua App này, người dùng có thể xem thông tin chi tiết mà các sensor gửi về, đồng thời tương tác từ xa với những thiết bị thông minh một cách rất thuận tiện. Ứng dụng cũng tích hợp khả năng quản lý người dùng bằng cách tạo tài khoản, đăng nhập với tài khoản Google,... Trong tương lai với những dự án thực tế rộng hơn, hoàn toàn có thể phát triển thêm App hiện tại để phù hợp, hoặc xây dựng Web App để trực quan dữ liệu dựa trên nền tảng Web.



Tài liệu

- [Web] Adafruit IO Server, <https://io.adafruit.com/>
- [Web] HiveMQ Server, <https://www.hivemq.com/>
- [Web] Firebase, <https://firebase.google.com/>