

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



PHÁT TRIỂN ỨNG DỤNG INTERNET OF THINGS (CO3038)

MINI PROJECT: MINI SMART-HOME APPLICATION

SV thực hiện:

Nguyễn Trọng Tín 2012215



Contents

1 Giới thiệu	3
1.1 ADAFRUIT	3
1.2 MQTT	3
1.3 PYTHON	4
1.4 GOOGLE TEACHABLE MACHINE	5
1.5 Android Studio	7
1.6 JAVA và KOTLIN	8
1.7 ESP32	10
2 Kiến trúc hệ thống	11
3 Hiện thực	12
3.1 Thiết bị	12
3.2 Gateway	13
3.3 Dashboard	15
3.4 Android Mobile App	16
3.4.1 Overview	16
3.4.2 The Sign-in View	16
3.5 The Monitor View	17
3.5.1 MQTT Handler	19
4 Experiments and Discussion	20

List of Figures

1 Adafruit's Logo	3
2 MQTT Logo	4
3 Python Logo	5
4 Google Teachable Machine	6
5 ESP32 developed kit	10
6 Kiến trúc Hệ thống Smart Home	11
7 Thiết bị	12
8 Sensor data in house	13
9 Face Recognition Results	14



10	Feeds on Adafruit IO	15
11	Dashboard on Adafruit Io	15
12	Overview của Android Mobile App	16
13	Different parts in CardView	18
14	MQTTHelper's Implementation and Abstract	19
15	SingleTon Pattern	19
16	Lỗi đọc sai dữ liệu ở cảm biến	20

1 Giới thiệu

1.1 ADAFRUIT

Adafruit là một công ty nổi tiếng trong lĩnh vực sản xuất và cung cấp các sản phẩm và giải pháp công nghệ sáng tạo, chuyên đặc biệt trong lĩnh vực Internet of Things (IoT) và điện tử sáng tạo.

Adafruit cung cấp một loạt các sản phẩm bao gồm các mô-đun và linh kiện điện tử, các kit phát triển, và các bảng mạch đa dạng để hỗ trợ những người sáng tạo, lập trình viên, và những người muốn khám phá thế giới của công nghệ. Công ty này còn nổi tiếng với việc cung cấp tài liệu hướng dẫn rõ ràng và chất lượng cao, giúp cộng đồng tận dụng tối đa các sản phẩm của họ.

Adafruit không chỉ tập trung vào việc cung cấp sản phẩm mà còn thúc đẩy cộng đồng thông qua việc hỗ trợ các dự án mã nguồn mở và chia sẻ kiến thức thông qua các bài hướng dẫn và video giáo dục. Điều này đã làm cho Adafruit trở thành một nguồn tài nguyên quan trọng và trung tâm cộng đồng cho những người đam mê điện tử và IoT.



Figure 1: Adafruit's Logo

Hơn thế nữa, Adafruit cung cấp một nền tảng IoT mạnh mẽ, **Adafruit IO**, mang lại giải pháp đơn giản để kết nối, giám sát và kiểm soát các thiết bị từ xa. Với Adafruit IO, bạn có thể dễ dàng tạo ra ứng dụng IoT đa dạng thông qua giao diện đồ họa thân thiện và hỗ trợ nhiều loại thiết bị và giao thức. Nền tảng này cung cấp các công cụ quản lý dữ liệu và tương tác linh hoạt, làm cho việc phát triển ứng dụng IoT trở nên linh hoạt và hiệu quả.

Mỗi người dùng sẽ được cấp một tài khoản, được xác minh bằng **USERNAME** và **KEY**. Dữ liệu sẽ được lưu trữ tại các **FEEDS**. Thông qua **DASHBOARD**, chúng ta có thể trực quan hóa dữ liệu, giám sát, cũng như điều khiển từ xa các thiết bị. Ngoài ra, Adafruit IO còn hỗ trợ các **ACTION** có các chức năng như **REACT** lại các tín hiệu gửi lên, ví dụ như gửi email, thông báo qua Telegram, **SCHEDULED** và **TIMER** các hành động được thiết lập trước.

1.2 MQTT

MQTT, Message Queuing Telemetry Transport, là một giao thức truyền thông nhẹ được phát triển để hỗ trợ việc truyền tải dữ liệu giữa các thiết bị trong môi trường Internet of Things (IoT). Giao thức



này tập trung vào hiệu suất, tiết kiệm băng thông, và độ tin cậy.

Publish (Xuất bản): Là quá trình mà một thiết bị gửi thông điệp (message) đến một địa chỉ trên broker (một máy chủ MQTT). Thông điệp này có thể là dữ liệu cảm biến, trạng thái thiết bị, hoặc bất kỳ thông tin nào khác. Cụ thể khi sử dụng platform Adafruit IO, chúng ta sẽ publish lên các Feed.

Subscribe (Đăng ký): Là quá trình mà một thiết bị yêu cầu nhận thông điệp từ một địa chỉ cụ thể trên broker. Khi thông điệp được xuất bản đến địa chỉ đó, thiết bị đăng ký sẽ nhận và xử lý thông điệp đó. Chúng ta có thể Subscribe vào các Feed trên Adafruit IO.

MQTT tạo ra một mô hình giao tiếp linh hoạt giữa các thiết bị, cho phép chúng xuất bản và đăng ký thông tin một cách dễ dàng, tối ưu hóa quá trình truyền thông trong mạng IoT.



Figure 2: MQTT Logo

1.3 PYTHON

Python là một ngôn ngữ lập trình bậc cao, đa năng, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Python được sử dụng rộng rãi trong các lĩnh vực như phát triển web, phân tích dữ liệu, học máy và trí tuệ nhân tạo, thậm chí là phát triển Hệ thống Nhúng, IoT...

Python ra mắt lần đầu tiên vào năm 1991 và đã có nhiều phiên bản cải tiến kể từ đó. Hiện nay, Python có hai phiên bản chính là Python 2 và Python 3, trong đó Python 3 là phiên bản mới nhất và được khuyến khích sử dụng với các ưu điểm như là:

- Dễ Đọc và Viết: Python có cú pháp rõ ràng và gần với ngôn ngữ tự nhiên, giúp người lập trình viết mã một cách nhanh chóng và dễ đọc.
- Da Nhiệm và Linh Hoạt: Python hỗ trợ đa nhiệm và có thư viện đa dạng, điều này làm cho nó linh hoạt và phù hợp với nhiều ứng dụng khác nhau.
- Cộng Đồng Lớn: Python có một cộng đồng phát triển đông đảo, với nhiều thư viện và framework hỗ trợ, giúp giảm độ phức tạp của việc phát triển phần mềm.
- Hỗ Trợ Đa Nền Tảng: Python có thể chạy trên nhiều hệ điều hành và kiến trúc, từ máy tính cá nhân đến thiết bị nhúng.

- Thư Viện Đa Dạng: Python có nhiều thư viện và framework phổ biến như Flask, Django, và Twisted, giúp xây dựng các ứng dụng nhúng một cách dễ dàng.
- Hỗ Trợ MQTT và IoT: Python có các thư viện như Paho MQTT cho việc triển khai các giao thức IoT như MQTT, làm cho nó trở thành một lựa chọn tốt cho các dự án gateway nhúng kết nối thiết bị IoT.

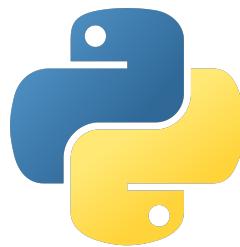


Figure 3: Python Logo

Python chủ yếu được sử dụng để phát triển các chương trình nhúng trong các gateway ví dụ như:

- Raspberry Pi: Raspberry Pi là một máy tính nhúng rất phổ biến và thường sử dụng hệ điều hành Raspbian, một phiên bản tùy chỉnh của Linux có sẵn hỗ trợ Python.
- Arduino (với MicroPython): Arduino thường sử dụng C/C++, nhưng có một phiên bản MicroPython được phát triển dành cho môi trường nhúng, cho phép bạn sử dụng Python trực tiếp.
- Jetson Nano: Jetson Nano của NVIDIA là một mô-đun máy tính nhúng mạnh mẽ, có thể chạy Python và hỗ trợ nhiều công nghệ AI.
- Orange Pi: Một loạt các bo mạch nhúng của Orange Pi chạy trên nền tảng Linux và hỗ trợ Python.

Đa số, các gateway, máy tính nhúng sẽ sử dụng hệ điều hành Linux, một nền tảng hệ điều hành mở, được phát triển rộng rãi sẽ được giới thiệu sau đây.

1.4 GOOGLE TEACHABLE MACHINE

Google Teachable Machine là một nền tảng trực tuyến do Google phát triển, cho phép người dùng dễ dàng đào tạo các mô hình học máy mà không cần kiến thức mã hóa sâu rộng. Nó cung cấp giao diện thân thiện với người dùng để tạo các mô hình học máy tùy chỉnh để nhận dạng hình ảnh, âm thanh hoặc tư thế.

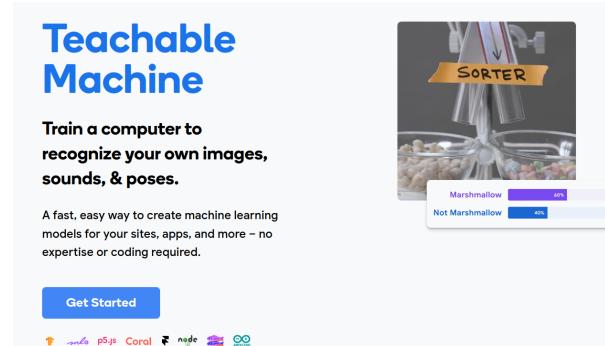


Figure 4: Google Teachable Machine

Các tính năng chính của GOOGLE TEACHABLE MACHINE bao gồm:

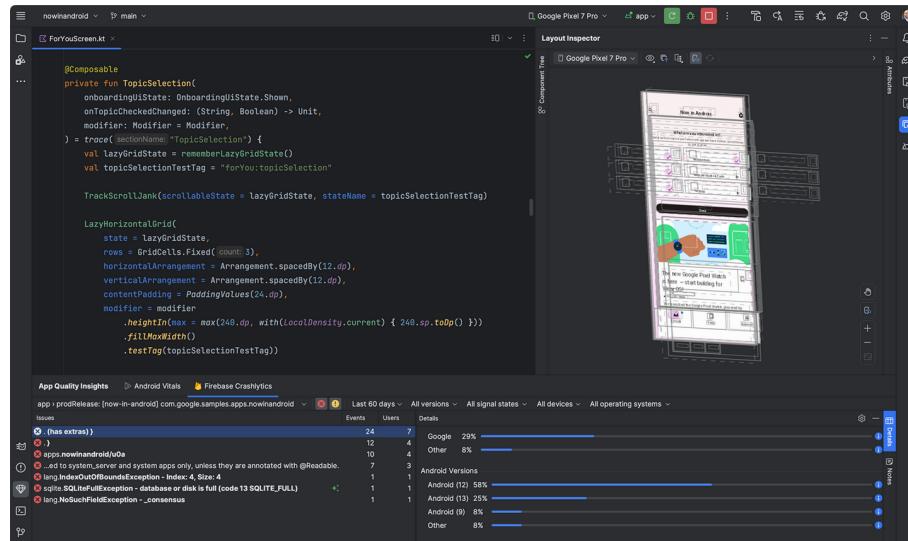
- Giao diện thân thiện với người dùng: Nền tảng được thiết kế để người dùng có ít hoặc không có kinh nghiệm viết mã có thể truy cập được. Nó đơn giản hóa quá trình đào tạo các mô hình học máy thông qua giao diện đồ họa trực quan.
- Nhận dạng hình ảnh, âm thanh và tư thế: Teachable Machine hỗ trợ nhiều loại mô hình học máy khác nhau, bao gồm phân loại hình ảnh, phân loại âm thanh và ước tính tư thế. Người dùng có thể huấn luyện người mẫu cách nhận biết các mẫu trong hình ảnh, âm thanh hoặc tư thế dựa trên các trường hợp sử dụng cụ thể của họ.
- Đào tạo theo thời gian thực: Nền tảng cho phép người dùng xem kết quả đào tạo trong thời gian thực, cho phép họ liên tục cải thiện mô hình của mình bằng cách chỉnh sửa dữ liệu đào tạo hoặc điều chỉnh các tham số.
- Xuất mô hình: Sau khi đào tạo một mô hình, người dùng có thể xuất mô hình đó để sử dụng trong các ứng dụng, trang web hoặc dự án của riêng họ. Các mô hình đã xuất có thể được tích hợp vào nhiều môi trường khác nhau hỗ trợ TensorFlow hoặc các khung máy học khác.
- Công cụ giáo dục: Máy có thể dạy của Google đóng vai trò là công cụ giáo dục nhằm giới thiệu cho các cá nhân, đặc biệt là sinh viên và người mới bắt đầu, về các khái niệm về học máy theo cách thực hành và tương tác.

Bằng cách dân chủ hóa quy trình tạo mô hình học máy, Google Teachable Machine hướng tới mục tiêu làm cho trí tuệ nhân tạo trở nên dễ tiếp cận và toàn diện hơn cho nhiều đối tượng hơn. Nó cung cấp một cách thực tế và hấp dẫn để người dùng khám phá các khả năng của máy học mà không cần đi sâu vào phát triển thuật toán và mã hóa phức tạp.



1.5 Android Studio

Android Studio là môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android, do Google cung cấp. Nó cung cấp một bộ công cụ và tính năng toàn diện để hợp lý hóa quy trình phát triển ứng dụng, từ thiết kế đến triển khai. Android Studio được xây dựng trên nền tảng IntelliJ IDEA và được thiết kế riêng cho việc phát triển Android.



Các tính năng và khía cạnh chính của Android Studio bao gồm:

- Trình chỉnh sửa mã thông minh:** Android Studio cung cấp trình chỉnh sửa mã thông minh với các tính năng như hoàn thành mã, đánh dấu cú pháp và tái cấu trúc mã để nâng cao năng suất. Nó hỗ trợ nhiều ngôn ngữ lập trình để phát triển Android, bao gồm **Java, Kotlin và C++**.
- Trình chỉnh sửa bố cục:** Trình chỉnh sửa bố cục cho phép nhà phát triển thiết kế trực quan giao diện người dùng cho ứng dụng của họ, với chức năng kéo và thả và xem trước theo thời gian thực. Các nhà phát triển có thể tạo bố cục phức tạp và tùy chỉnh các thành phần giao diện người dùng một cách dễ dàng.
- Trình mô phỏng tích hợp:** Android Studio bao gồm trình mô phỏng tích hợp sẵn để thử nghiệm ứng dụng trên các cấu hình thiết bị Android khác nhau. Nhà phát triển có thể mô phỏng nhiều kích thước màn hình, độ phân giải và cấu hình phần cứng khác nhau để đảm bảo ứng dụng của họ hoạt động liền mạch trên các thiết bị khác nhau.
- Công cụ lập hồ sơ hiệu suất:** Android Studio cung cấp các công cụ lập hồ sơ hiệu suất để giúp nhà phát triển phân tích và tối ưu hóa hiệu suất ứng dụng của họ. Các nhà phát triển có thể giám sát việc sử dụng CPU, bộ nhớ và mạng, xác định các tắc nghẽn về hiệu suất và tối ưu hóa mã của họ để có hiệu quả tốt hơn.



- **Tích hợp kiểm soát phiên bản:** Android Studio tích hợp với các hệ thống kiểm soát phiên bản như Git, cho phép các nhà phát triển quản lý và cộng tác trên cơ sở mã của họ một cách hiệu quả. Nó cung cấp các tính năng để tạo phiên bản mã, phân nhánh, hợp nhất và giải quyết xung đột.
- **Hệ thống xây dựng dựa trên Gradle:** Android Studio sử dụng hệ thống xây dựng Gradle để tự động hóa quá trình xây dựng, thử nghiệm và triển khai ứng dụng Android. Gradle cung cấp các tùy chọn tùy chỉnh và linh hoạt để quản lý các phần phụ thuộc, xây dựng các biến thể ứng dụng và tạo APK.
- **Google Play Services Integration:** Android Studio tích hợp liền mạch với các dịch vụ của Google Play, cho phép nhà phát triển dễ dàng tích hợp các tính năng như bản đồ, dịch vụ vị trí, xác thực và nhắn tin trên đám mây vào ứng dụng của họ. Nhà phát triển có thể tận dụng API dịch vụ Google Play để thêm chức năng mạnh mẽ vào ứng dụng của họ mà không tốn nhiều công sức.
- **Extensible:** Android Studio có khả năng mở rộng cao, hỗ trợ các plugin và tiện ích mở rộng giúp nâng cao chức năng và phục vụ các trường hợp sử dụng hoặc quy trình phát triển cụ thể. Nhà phát triển có thể cài đặt plugin để thêm tính năng, công cụ hoặc tiện ích tích hợp mới, tùy chỉnh Android Studio cho phù hợp với nhu cầu của họ.

1.6 JAVA và KOTLIN

Khi trải nghiệm Android Studio, chúng tôi phải tham gia và phát triển ứng dụng của mình bằng cách sử dụng hai ngôn ngữ mạnh mẽ và mạnh mẽ: Java và Kotlin.

Java là ngôn ngữ lập trình hướng đối tượng được sử dụng rộng rãi được phát triển bởi Sun Microsystems (hiện thuộc sở hữu của Tập đoàn Oracle) vào giữa những năm 1990. Nó được thiết kế độc lập với nền tảng và có thể chạy trên mọi thiết bị có Máy ảo Java (JVM), khiến nó có tính di động cao. Java được biết đến nhờ tính đơn giản, dễ đọc và hệ sinh thái thư viện và framework rộng lớn.



- **Độ độc lập nền tảng:** Các chương trình Java có thể chạy trên mọi thiết bị có JVM, bất kể hệ điều hành cơ bản là gì. Điều này làm cho Java có tính di động cao và phù hợp để phát triển các ứng dụng đa nền tảng.
- **Hướng đối tượng:** Java là ngôn ngữ lập trình hướng đối tượng thuận túy, nhấn mạnh khái niệm đối tượng và lớp. Nó hỗ trợ đóng gói, kế thừa và đa hình, cho phép các nhà phát triển viết mã mô-đun và có thể tái sử dụng.

- **Mạnh mẽ và an toàn:** Java cung cấp các tính năng tích hợp sẵn để xử lý lỗi, quản lý bộ nhớ và bảo mật. Nó bao gồm thu thập rác tự động, xử lý ngoại lệ và kiểm tra loại mạnh để đảm bảo mã mạnh mẽ và đáng tin cậy.
- **Hệ sinh thái lớn:** Java có một hệ sinh thái rộng lớn gồm các thư viện, khung và công cụ hỗ trợ phát triển phần mềm. Các framework phổ biến như Spring, Hibernate và Apache Struts được sử dụng rộng rãi để xây dựng các ứng dụng cấp doanh nghiệp.
- **Hỗ trợ cộng đồng:** Java có một cộng đồng các nhà phát triển rộng lớn và tích cực, góp phần vào sự phát triển và phát triển của nó. Quy trình cộng đồng Java (JCP) giám sát sự phát triển của ngôn ngữ và nền tảng Java, đảm bảo các bản cập nhật và cải tiến liên tục.



Kotlin là ngôn ngữ lập trình kiểu tĩnh, hiện đại được phát triển bởi JetBrains, công ty đứng sau IntelliJ IDEA, vào năm 2011. Kotlin được thiết kế để có thể tương tác hoàn toàn với Java, cho phép các nhà phát triển tích hợp liền mạch Kotlin với các dự án Java hiện có. Kotlin nhằm mục đích giải quyết một số hạn chế của Java trong khi cung cấp cú pháp ngắn gọn và biểu cảm hơn.

- Cú pháp ngắn gọn: Kotlin cung cấp cú pháp ngắn gọn và biểu cảm, giảm bớt mã soạn sẵn và cải thiện khả năng đọc. Các tính năng như suy luận kiểu, lớp dữ liệu và chức năng mở rộng cho phép các nhà phát triển viết mã nhỏ gọn và biểu cảm hơn.
- Null Safety: Kotlin cung cấp các tính năng an toàn null tích hợp để ngăn chặn các ngoại lệ con trỏ null, một nguồn lỗi phổ biến trong Java. Các loại có thể rỗng và không thể rỗng, cùng với toán tử cuộc gọi an toàn và toán tử Elvis, giúp các nhà phát triển viết mã an toàn hơn và mạnh mẽ hơn.
- Coroutines: Kotlin giới thiệu coroutines, một framework đồng thời gọn nhẹ dành cho lập trình không đồng bộ. Coroutine đơn giản hóa việc xử lý các tác vụ không đồng bộ, chẳng hạn như yêu cầu mạng hoặc tính toán chạy dài, bằng cách cung cấp cách tiếp cận đồng thời theo tuần tự và có cấu trúc.
- Khả năng tương tác với Java: Kotlin hoàn toàn có thể tương tác với Java, cho phép các nhà phát triển sử dụng mã Kotlin và Java cùng nhau trong cùng một dự án. Mã Kotlin có thể gọi mã Java và ngược lại, giúp dễ dàng di chuyển các dự án Java hiện có sang Kotlin hoặc dần dần áp dụng Kotlin.
- Được Google hỗ trợ chính thức: Năm 2017, Google đã công bố Kotlin là ngôn ngữ lập trình chính thức để phát triển ứng dụng Android, cùng với Java. Các tính năng hiện đại và khả năng tích hợp

liền mạch của Kotlin với các dự án Android hiện có đã khiến Kotlin ngày càng trở nên phổ biến đối với các nhà phát triển Android.

1.7 ESP32

ESP32 là một vi điều khiển đa chức năng và mạnh mẽ được phát triển bởi Espressif Systems. Nó kết hợp cả khả năng kết nối Wi-Fi và Bluetooth trong một chip đơn, cùng với nhiều tính năng khác như cổng GPIO, cổng SPI, cổng I2C và các chức năng xử lý tín hiệu nhúng.



Figure 5: *ESP32 developed kit*

- **Hiệu suất cao:** ESP32 được trang bị vi xử lý mạnh mẽ và nhanh nhạy, cho phép xử lý các tác vụ phức tạp và thời gian phản hồi nhanh chóng.
- **Tiêu thụ điện năng thấp:** Chip ESP32 được thiết kế để tiết kiệm năng lượng, giúp tăng tuổi thọ pin và phù hợp với các ứng dụng yêu cầu năng lượng thấp.
- **Kết nối đa dạng:** ESP32 hỗ trợ kết nối Wi-Fi và Bluetooth cùng một lúc, cung cấp sự linh hoạt trong việc truyền dữ liệu và kết nối mạng.
- **Cổng GPIO đa chức năng:** ESP32 có nhiều cổng GPIO cho phép kết nối với các thiết bị ngoại vi và cảm biến khác nhau, mở rộng khả năng ứng dụng của nó.
- **Hỗ trợ cho các giao thức phổ biến:** ESP32 hỗ trợ các giao thức không dây phổ biến như Wi-Fi, Bluetooth, BLE (Bluetooth Low Energy), và các giao thức mạng khác.
- **Cộng đồng lớn và tích cực:** Sự phát triển và hỗ trợ của ESP32 được hưởng lợi từ một cộng đồng lớn và năng động, cung cấp tài liệu, ví dụ và hỗ trợ kỹ thuật cho các nhà phát triển.
- **Dễ dàng tích hợp:** ESP32 có các công cụ và tài liệu hỗ trợ phong phú, giúp việc phát triển ứng dụng và tích hợp vào các dự án trở nên dễ dàng và nhanh chóng.

- **Chi phí hợp lý:** ESP32 được cung cấp với một mức giá phải chăng so với các vi điều khiển và module có tính năng tương đương, làm cho nó trở thành lựa chọn phổ biến cho nhiều dự án IoT và các ứng dụng nhúng.

ESP32 được sử dụng rộng rãi trong nhiều ứng dụng IoT (Internet of Things) như thiết bị nhà thông minh, cảm biến môi trường, hệ thống giám sát và điều khiển từ xa, và các dự án DIY (làm đồ tự làm). Nó cũng được sử dụng trong các ứng dụng như điều khiển robot, thiết bị đeo thông minh, và các dự án IoT khác đòi hỏi kết nối không dây và xử lý dữ liệu nhúng.

2 Kiến trúc hệ thống

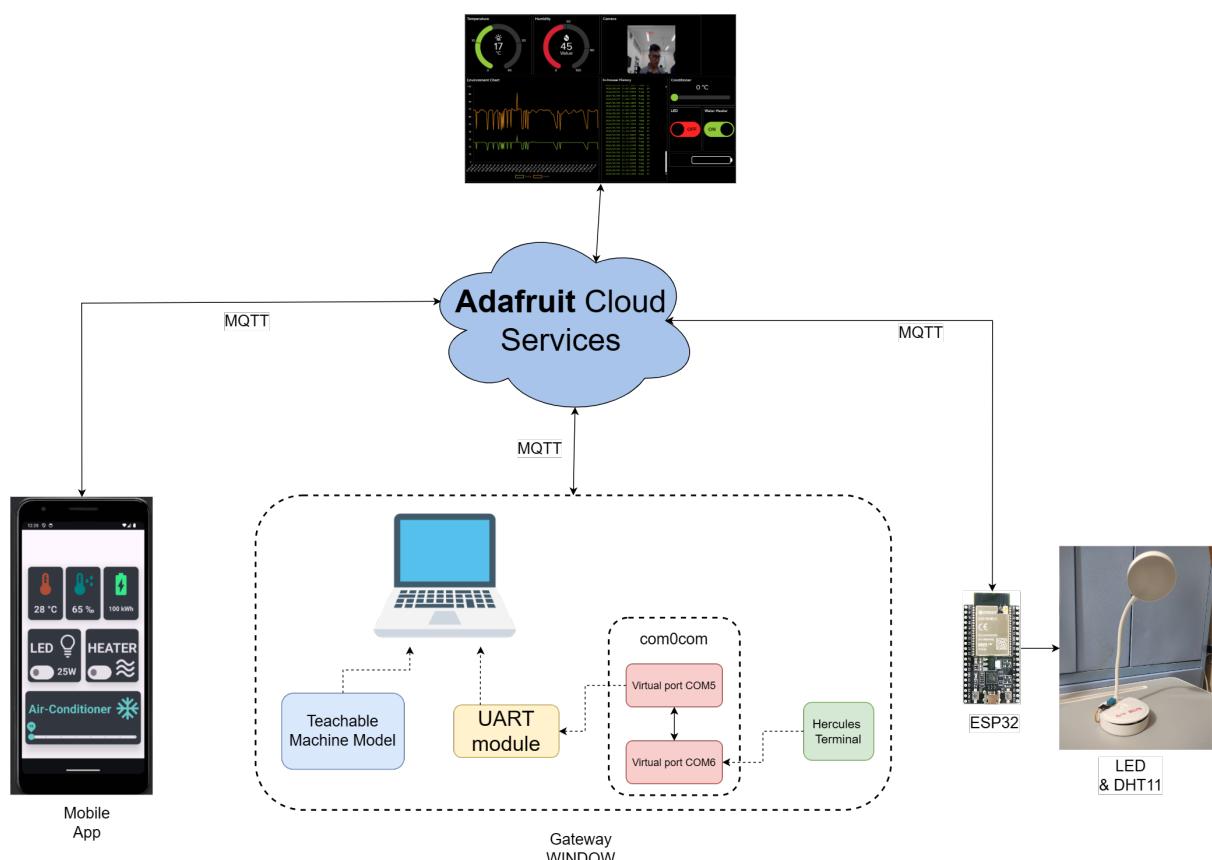


Figure 6: Kiến trúc Hệ thống Smart Home

Kiến trúc hệ thống tổng quát như hình trên, trong đó PC, Laptop, hoặc một máy tính nhúng sẽ hoạt động như 1 gateway chạy một chương trình Python. Gateway được xây dựng một mô hình **nhận dạng khuôn mặt** nhờ vào công cụ Google Teachable Machine giúp điểm danh thành viên có trong nhà, đồng thời dựa vào số lượng thành viên để điều chỉnh **Điều hòa, Quạt, Đèn** trong phòng.

Phía Platform Adafruit, bao gồm 7 FEEDS. Trong đó, 4 FEEDS dùng để **lưu trữ dữ liệu quan trắc**: Nhiệt độ, độ ẩm, dung lượng Pin mặt trời, và Hình ảnh từ Camera. 3 FEEDS còn lại dùng để **điều**

khiển các thiết bị: Đèn và quạt, Máy điều hòa, Máy nước nóng. Platform có nhiệm vụ thống kê các dữ liệu được đẩy lên, ví dụ **Nhiệt độ và Độ ẩm sẽ được visualize bằng line chart**, **Camera cũng được hiển thị**. Hơn thế nữa, các nút nhấn bật tắt đèn, bật tắt máy nước nóng, thanh trượt điều chỉnh điều hòa cũng được hiện thực.

App Mobile sẽ được cung cấp cho các thành viên trong gia đình dùng để theo dõi điều kiện trong nhà, đồng thời tự điều chỉnh Đèn và quạt, Máy điều hòa, Máy nước nóng.

Các thiết bị ngoại vi như Đèn, Cảm biến được kết nối thông qua vi điều khiển trung tâm ESP32, kết nối với WiFi và giao tiếp với Platform bằng giao thức MQTT.

3 Hiện thực

3.1 Thiết bị

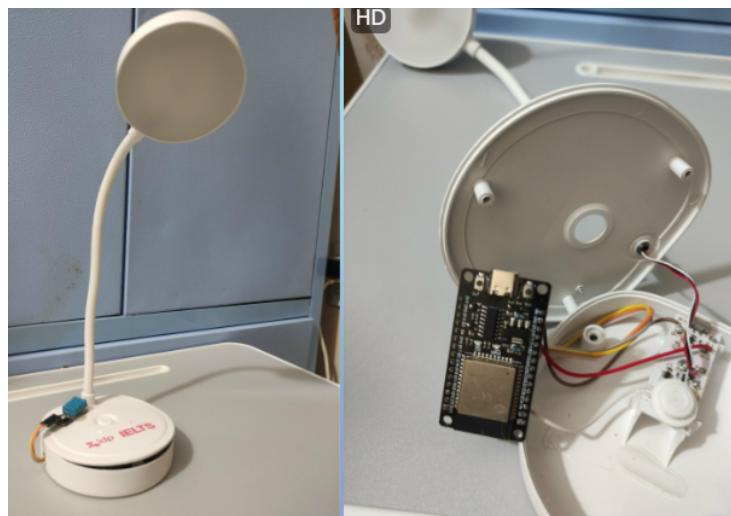


Figure 7: Thiết bị

Phần thiết bị được cải tiến từ đèn bàn cũ, bao gồm 3 phần: Đèn, Cảm biến DHT11 và Vi điều khiển ESP32. Thiết bị hoạt động như một MQTT Client kết nối với WiFi và giao tiếp với Server thông qua giao thức MQTT. **Mỗi 30 giây**, thiết bị sẽ đọc dữ liệu cảm biến **Nhiệt độ và Độ ẩm** trong nhà gửi lên Server. Sau đó, Platform sẽ visualize dữ liệu, mọi người sẽ theo dõi dữ liệu qua Mobile app, và quan trọng là Gateway sẽ nhận được dữ liệu từ đó đưa ra quyết định điều chỉnh Điều hòa trong phòng cho phù hợp.



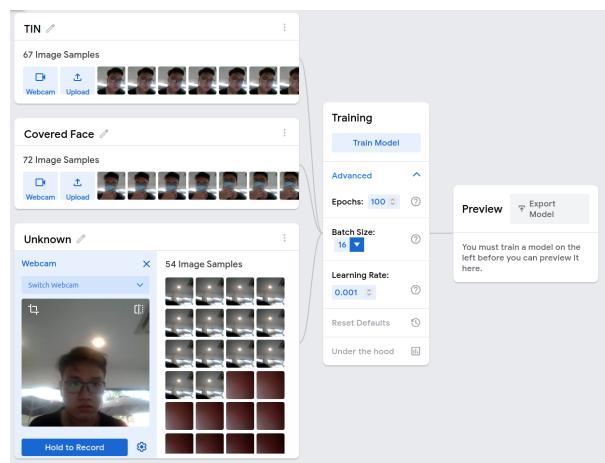
In-house History			
2024/05/09	11:18:23PM	Temp	44
2024/05/09	11:18:24PM	Humi	45
2024/05/09	11:22:18PM	Temp	17
2024/05/09	11:22:18PM	Humi	44
2024/05/09	11:22:57PM	Temp	26
2024/05/09	11:22:57PM	Humi	67
2024/05/09	11:23:36PM	Temp	26
2024/05/09	11:23:36PM	Humi	68
2024/05/09	11:24:15PM	Temp	26
2024/05/09	11:24:15PM	Humi	68
2024/05/09	11:24:54PM	Temp	26
2024/05/09	11:24:54PM	Humi	68
2024/05/09	11:25:33PM	Temp	26
2024/05/09	11:25:33PM	Humi	67
2024/05/09	11:26:12PM	Temp	26
2024/05/09	11:26:12PM	Humi	66
2024/05/09	11:26:51PM	Temp	26
2024/05/09	11:26:51PM	Humi	65
2024/05/09	11:30:06PM	Temp	17
2024/05/09	11:30:07PM	Humi	42
2024/05/09	11:30:45PM	Temp	26
2024/05/09	11:30:46PM	Humi	65
2024/05/09	11:31:24PM	Temp	26
2024/05/09	11:31:25PM	Humi	66
2024/05/09	11:32:03PM	Temp	26
2024/05/09	11:32:04PM	Humi	67

Figure 8: Sensor data in house

3.2 Gateway

Trung tâm điều khiển trong nhà là GATEWAY có các chức năng nhận dạng khuôn mặt và điều khiển thiết bị dựa vào dữ liệu cảm biến.

Dầu tiên về phần hiện thực mô hình nhận dạng khuôn mặt, giải pháp được sử dụng ở đây là Google Teachable Machine. Chủ nhà hay admin sẽ tạo nhiều classname của thành viên trong nhà để tạo mô hình phân loại, trong trường hợp này là 3 classname bao gồm Tin, Covered face và Unknown.



Sau đó là phần hiệu chỉnh thông số cho training model **Epoch**, **Batch size**, **Learning Rate**.

- **Epoch:** Rằng mọi mẫu đào tạo tập dữ liệu phải được cung cấp thông qua mô hình ít nhất một mẫu. Trong trường hợp này, Epoch được tăng từ 50 (theo mặc định) lên 100, điều đó có nghĩa là tất cả mẫu của tôi sẽ đi qua mô hình ít nhất 100 lần, nó sẽ tăng cường khả năng dự đoán nhưng mất nhiều thời gian hơn để training.
- **Batch size:** Điều đó cho biết kích thước của mẫu sẽ được đưa vào mô hình. Giả sử nó là 16 và tập dữ liệu bao gồm 80 hình ảnh, vì vậy có 5 batches (80/16). Sau khi 5 batches được cung cấp thông qua mô hình, một Epoch sẽ được tính. Không cần định cấu hình tham số này, nó được sử dụng để tách tập dữ liệu thành các phần nhóm nhỏ để tránh quá tải bộ nhớ và tính toán.
- **Learning rate:** Độ lưỡng kích thước của các bước đã thực hiện, đó là mức độ aggressive mà AI cần được train. Việc tăng nó sẽ làm cho mô hình đạt được mục tiêu nhanh hơn, nhưng nó có thể vượt quá mức. Ngược lại, việc tập luyện sẽ mất nhiều thời gian hơn để đạt được giá trị trọng lượng tối ưu. Google khuyến khích nên sửa đổi tham số này chỉ khi là chuyên gia trong lĩnh vực này.

Dầu tiên, chúng ta cần cài đặt một số packages và quan trọng nhất là TensorFlow. Dưới đây là các yêu cầu tiên quyết về hệ thống, để tăng tốc phần cứng bằng cách sử dụng GPU (NVIDIA RTX 2050 trong trường hợp của tôi). Thông qua thực nghiệm, chúng ta nên sử dụng LINUX để tối ưu hóa các hỗ trợ từ mô hình TensorFlow và AI. Hiệu suất của ứng dụng AI trên Window quá kém, phải mất nhiều hơn **89ms mỗi bước** để xử lý, nhưng **chỉ 15ms** trong trường hợp xây dựng trên nhân Linux. Bởi vì ứng dụng chạy trên Window sử dụng CPU để xử lý thay vì sử dụng GPU như Linux.

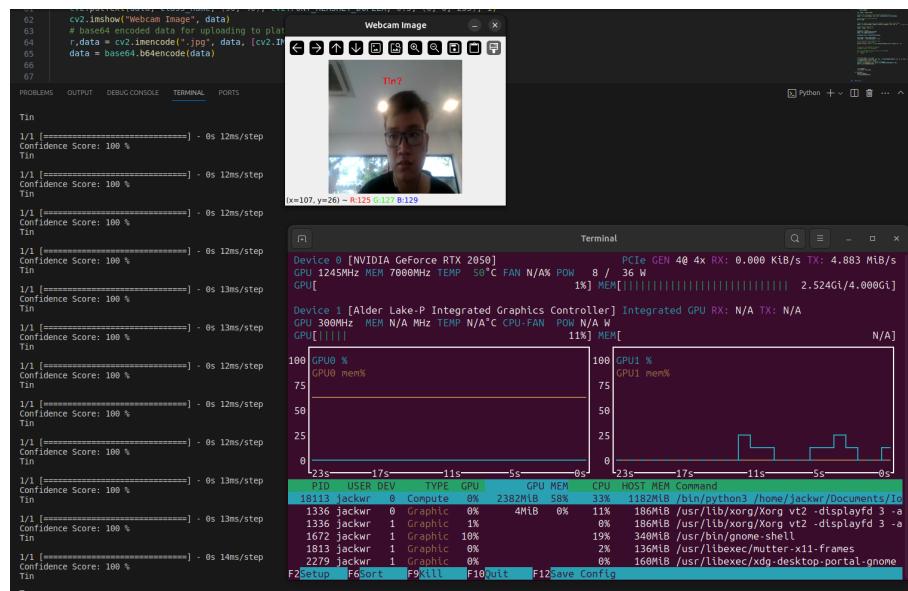


Figure 9: Face Recognition Results



3.3 Dashboard

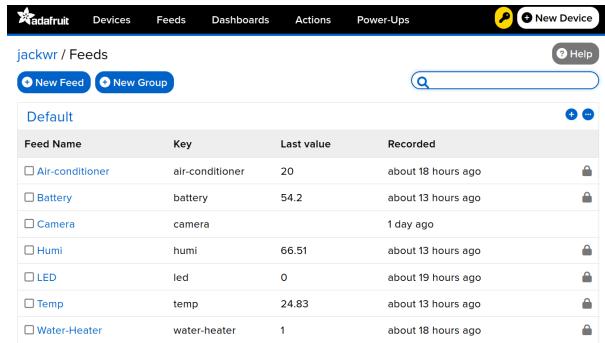


Figure 10: Feeds on Adafruit IO

Dashboard bao gồm 7 FEEDS được tạo trên Adafruit IO, trong đó Temperature (Nhiệt độ), Humidity (Độ ẩm) và Dung lượng Pin (Battery) sẽ được cập nhật theo thứ tự mỗi 5 giây. Chỉ số Nhiệt độ sẽ dao động quanh Nhiệt độ được điều chỉnh bởi Air-conditioner (Máy điều hòa). Dung lượng Pin sẽ giảm nhanh hoặc chậm dựa vào việc LED và Máy nước nóng có mở hay không.

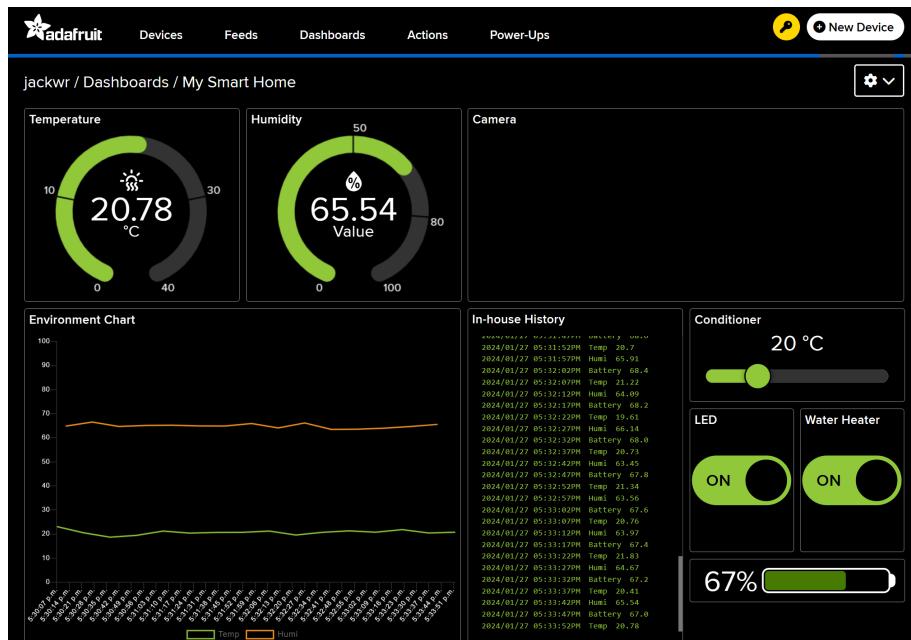


Figure 11: Dashboard on Adafruit Io

Dashboard trực quan trên Adafruit IO, trong đó Nhiệt độ (định rõ trong khoảng 10 - 30 °C) và Độ ẩm (định rõ trong khoảng 50 - 80 %) được theo dõi trên Gauge chart. Line chart dùng để theo dõi trực quan lịch sử dao động của Nhiệt độ và độ ẩm trong 1 giờ gần nhất. Ngoài ra, còn có bảng Lịch sử giá trị tất cả 7 Feeds trong 1 giờ gần nhất, và Dung lượng Pin.

Có thể điều khiển máy điều hòa trong khoảng 16 đến 30 °C bằng slider, công tắc bật tắt Đèn và Máy nước nóng

3.4 Android Mobile App

3.4.1 Overview

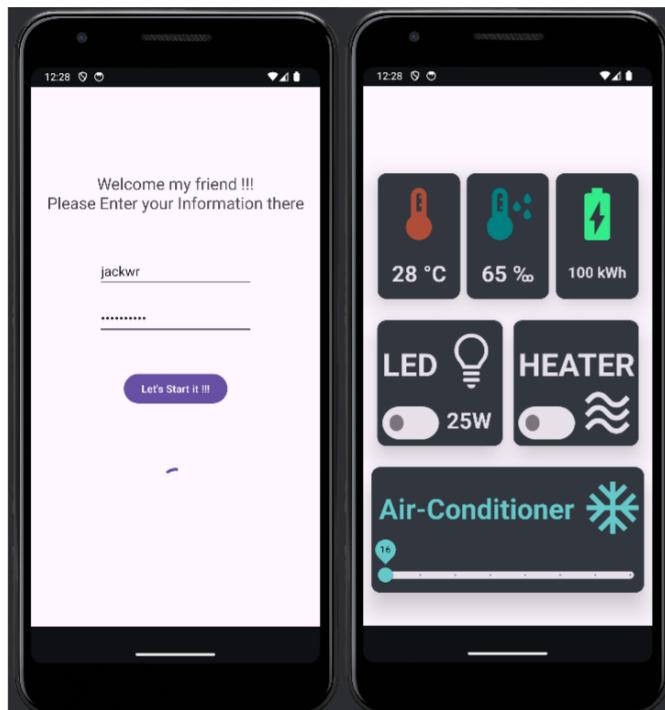


Figure 12: Overview của Android Mobile App

Ứng dụng mobile này sẽ được cung cấp cho các thành viên trong gia đình. Mỗi thành viên có tài khoản riêng để đăng nhập. Các thành viên có quyền xem các thông tin Nhiệt độ, Độ ẩm, Dung lượng pin Năng lượng mặt trời. Ngoài ra, các thành viên còn có thể điều khiển Đèn, Máy nước nóng và Điều hòa.

Ứng dụng Android hiện tại bao gồm hai View được gọi là **Activity** trong Android Studio.

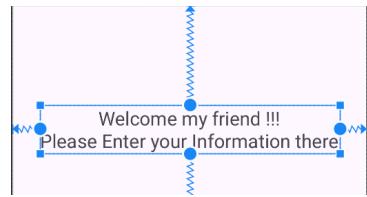
* Chế độ xem đầu tiên là **Chế độ đăng nhập**, chế độ xem này lấy **Tên người dùng** và **Mật khẩu** của bạn để có quyền truy cập vào **chế độ xem Điều khiển (Monitor)**.

* **Chế độ Điều khiển**, tại thời điểm này, có 3 thẻ điều khiển: LED, Heater và Air Conditioning và kèm theo 3 thẻ trạng thái. Chế độ xem này cho phép các thành viên giám sát các thiết bị theo các chức năng cụ thể như bật tắt bằng **Switch** và **Slider** để điều chỉnh nhiệt độ. Hơn nữa, có thể quan sát trực quan các số liệu **Nhiệt độ**, **Độ ẩm** và **Pin năng lượng mặt trời**

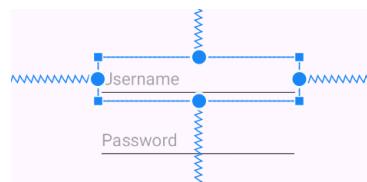
3.4.2 The Sign-in View

Nhìn chung có 3 loại thành phần: TextView, EditText và Button

- **TextView**: Phần tử giao diện người dùng (UI) được sử dụng để hiển thị văn bản trên màn hình. Chỉ có thể chỉnh sửa nó khi lập trình; do đó, người dùng không thể chỉnh sửa nó trong thời gian chạy (một số hàm phụ trợ trong thời gian chạy có thể thay đổi)



- **EditText:** dùng để nhập thông tin đầu vào của người dùng ở dạng văn bản. Nó cho phép người dùng nhập và chỉnh sửa dữ liệu văn bản, chẳng hạn như tên người dùng, mật khẩu, địa chỉ email hoặc bất kỳ loại thông tin văn bản nào khác.



- **Button:** phần tử được sử dụng để kích hoạt một hành động hoặc thực hiện một thao tác khi người dùng nhấn hoặc nhấp vào. Các nút thường được sử dụng để bắt đầu các hành động như gửi biểu mẫu, điều hướng đến màn hình khác hoặc thực thi một chức năng trong ứng dụng.

Trong trường hợp này, Button được sử dụng để xác minh thông tin đăng nhập và chuyển tiếp tới MonitorView tiếp theo.



3.5 The Monitor View

Chế độ xem màn hình này hơi nâng cao một chút, bao gồm một **CardView**, chứa 3 loại thành phần khác nhau như **TextView**, **ImageView** và **Switch**. Tất cả các thành phần này được thiết kế bởi **Material Component Android (MDC)**. Có thể truy cập, tham khảo và xem hướng dẫn từ MDC qua link sau: <https://github.com/material-components/material-components-android>

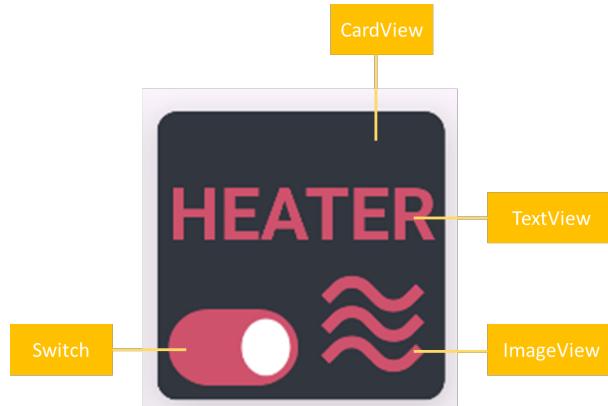
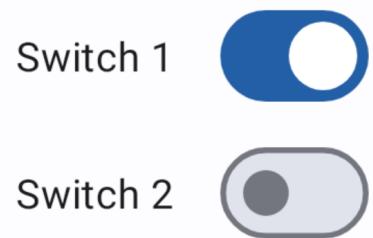


Figure 13: Different parts in CardView

- **CardView:** CardView là một thành phần giao diện người dùng trong Android cho phép nhà phát triển hiển thị nội dung theo bố cục giống như thẻ linh hoạt với các góc và độ cao được bo tròn, tạo ra hình thức thẻ hấp dẫn về mặt thị giác.

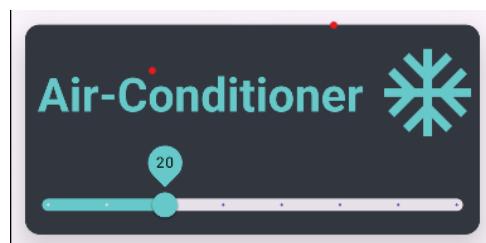
Sử dụng bằng cách <**com.google.android.material.card.MaterialCardView**> in the **Activity.xml** file.

- **Switch:** đại diện cho một nút có hai trạng thái bật và tắt. Công tắc thường được sử dụng nhiều nhất trên thiết bị di động để bật và tắt các tùy chọn trong menu tùy chọn.

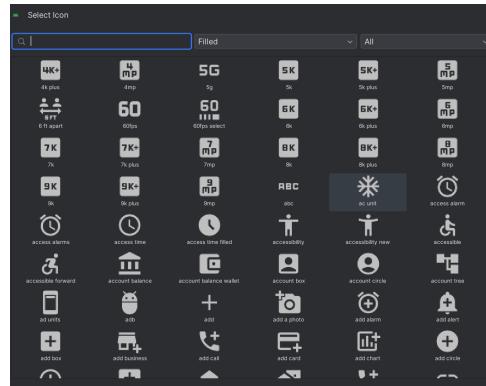


- **SliderView:** cho phép người dùng chọn một giá trị từ một phạm vi liên tục bằng cách trượt ngón tay cái dọc theo một rãnh. Nó cung cấp một cách trực quan để người dùng điều chỉnh cài đặt hoặc nhập các giá trị số trong một phạm vi được chỉ định.

Trong trường hợp này, Slider có phạm vi từ 16 đến 30 với mỗi đơn vị kéo là 2.



- **ImageView:** được sử dụng để hiển thị hình ảnh hoặc đối tượng có thể vẽ trong giao diện người dùng của ứng dụng. Cung cấp một cách để thể hiện hình ảnh một cách trực quan ở nhiều định dạng khác nhau, chẳng hạn như hình ảnh bitmap, hình vẽ vector hoặc đồ họa hoạt hình. Trong ứng dụng này, App sử dụng một số biểu tượng tích hợp phổ biến từ Android Studio, rất thuận tiện cho người mới.



3.5.1 MQTT Handler

Để liên lạc với Máy chủ MQTT của Adafruit, ứng dụng Android cần hoạt động như một Máy khách MQTT. Trước hết, tạo một lớp có tên MQTTHelper có một số Thuộc tính và Phương thức quan trọng.

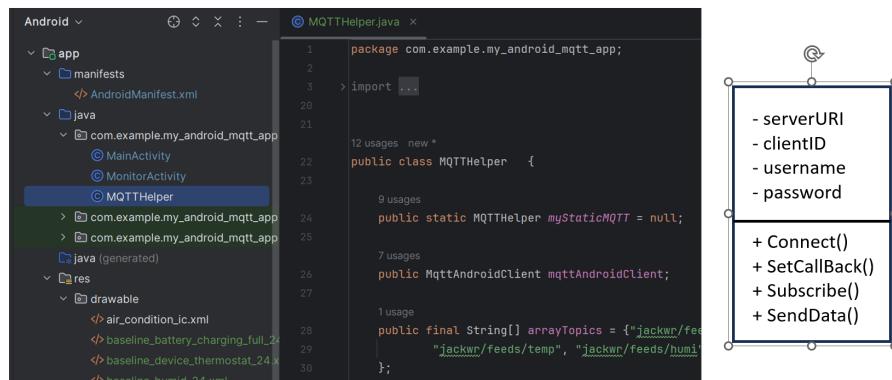


Figure 14: *MQTTHelper's Implementation and Abstract*

Tuy nhiên, chúng ta cần ứng dụng khách MQTT tồn tại trong suốt phiên và tồn tại trong **Activity View** khác nhau. Đó là lý do **Singleton Pattern** được sử dụng để lưu trữ ứng dụng khách MQTT, là một Static Object và sẽ tồn tại trong suốt vòng đời của Lớp MQTTHelper. Như vậy ứng dụng có thể gọi, ghi đè và triển khai MQTT Client trong mọi Activity View. Lưu ý rằng Construction Method của Singleton Object hơi khác một chút, phải sử dụng phương thức **MQTTHelper.GetMQTTClient()** thay vì sử dụng **new MQTTHelper()**. Kết quả là dù Construction Method được tạo mới ở bất kỳ đâu hay khai báo nhiều tên biến nhiều lần, nó luôn trả về Static MQTTClient duy nhất.

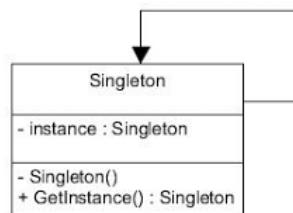


Figure 15: *Singleton Pattern*

Cuối cùng, chúng ta có thể sử dụng nó và ghi đè phương thức **SetCallBack** để yêu cầu nó xử lý các tin nhắn đã nhận. Hơn nữa, chúng ta có thể gọi phương thức "SendDataMQTT" để gửi tin nhắn đến máy chủ.

4 Experiments and Discussion

Sau khi hiện thực, hệ thống hoạt động khá ổn định, App mobile và Dashboard đồng bộ gần như là realtime với nhau. Ngoài ra, thiết bị cập nhật dữ liệu liên tục rất tốt cũng như là có thể điều khiển độ sáng của đèn realtime. Về phần Gateway, mô hình dữ liệu khuôn hoạt động tốt và không xảy ra lỗi, mỗi khi nhiệt độ tăng lên ngưỡng nhất định, Gateway tự động điều chỉnh điều hòa đúng với thiết kế.

Tuy nhiên, còn một vài sai số trong việc đọc cảm biến. Hướng giải quyết trong tương lai là có thể mua cảm biến tốt hơn, hoặc sử dụng các thuật toán lọc dữ liệu nhiều lần trước khi publish lên server.

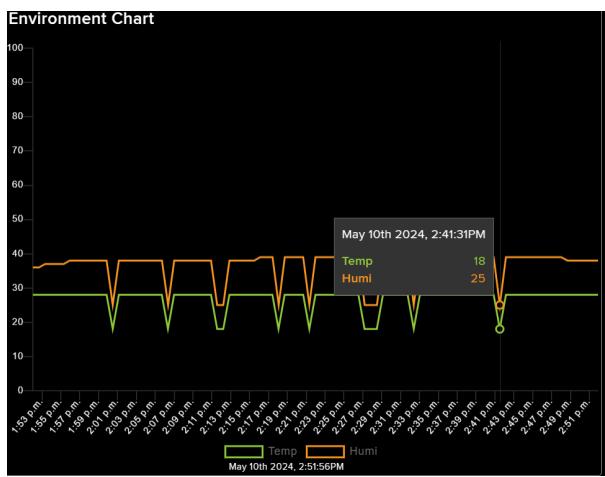


Figure 16: Lỗi đọc sai dữ liệu ở cảm biến

Video Demo upload here: <https://youtu.be/vw6Z-bhbDwc>

Please check my github for more information about source code and building problems
https://github.com/JackWrion/IoT_LAB_2024/tree/main/Mini_project