

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
_____ * _____

TỐT NGHIỆP ĐẠI HỌC

NGÀNH CÔNG NGHỆ THÔNG TIN

HỆ THỐNG TÍCH HỢP QUẢN LÝ CÁC IOT PLATFORM

Sinh viên thực hiện : **Bùi Ngọc Luân**
Lớp : **CNTT-TT 2.03 – K57**
Giáo viên hướng dẫn : **TS. Nguyễn Bình Minh**

HÀ NỘI, 12-2016

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên: **BÙI NGỌC LUÂN**

Điện thoại liên lạc: **0165.985.6822**

Email: ngocluan.bka@gmail.com

Lớp: **CNTT-TT 2.03 – K57**

Hệ đào tạo: **Đại học chính quy**

Đồ án tốt nghiệp thực hiện tại: **Bộ môn Hệ thống thông tin, Viện Công nghệ thông tin và truyền thông, Đại học Bách Khoa Hà Nội.**

Thời gian làm ĐATN: Từ ngày đến 4/9/2016 đến 22/12/2016

2. Mục đích nội dung của ĐATN

Xây dựng một hệ thống để có thể tích hợp nhiều IoT Platform lại với nhau.

3. Các nhiệm vụ cụ thể của đồ án

- Xây dựng một giao diện duy nhất để điều khiển các thiết bị do nhiều IoT Platform quản lý.
- Cập nhật dữ liệu trạng của các thiết bị (ví dụ như cảm biến...) theo thời gian thực.
- Lưu trữ dữ liệu thu thập được từ các thiết bị.
- Cung cấp các công cụ để có thể viết ra các luật để điều khiển hệ thống tự động.
- Xây dựng một hệ thống có tính khả mở.

4. Lời cam đoan của sinh viên

Tôi – *Bùi Ngọc Luân* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. Nguyễn Bình Minh.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm
Tác giả ĐATN

Họ và tên sinh viên

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm
Giáo viên hướng dẫn

TS. Nguyễn Bình Minh

TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Đồ án tiến hành nghiên cứu và xây dựng một nền tảng có thể tích hợp nhiều IoT Platform khác nhau.

Đồ án giải quyết các vấn đề:

- Xây dựng một giao diện duy nhất để điều khiển các thiết bị do nhiều IoT Platform quản lý.
- Cập nhật dữ liệu trạng của các thiết bị (ví dụ như cảm biến...) theo thời gian thực.
- Lưu trữ dữ liệu thu thập được từ các thiết bị.
- Cung cấp các công cụ để có thể viết ra các luật để điều khiển hệ thống tự động.
- Xây dựng một hệ thống có tính khả mở.

Nội dung của đồ án tốt nghiệp gồm 3 chương.

Chương I, Chương mở đầu: Trình bày các vấn đề gặp phải, cách giải quyết vấn đề và mục đích của đề tài

Chương II, Các kết quả đạt được:

1. Trình bày về thiết kế và kiến trúc của hệ thống
2. Cung cấp mô tả đầy đủ về tất cả các chức năng chính cũng như giới hạn của hệ thống
3. Trình bày kết quả kiểm thử của hệ thống

Chương III, Kết luận và hướng phát triển: Tổng kết các kết quả đạt được, trình bày hướng phát triển cho tương lai.

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, tôi đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc nhất, tôi xin gửi lời cảm ơn đến quý Thầy cô ở Viện công nghệ thông tin và truyền thông – Trường Đại học Bách Khoa Hà Nội đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho tôi để tôi có thể kết thúc chặng đường 5 năm học tập tại trường.

Tôi xin cảm ơn TS. Nguyễn Bình Minh đã hướng dẫn tận tình, theo sát quá trình thực hiện đồ án tốt nghiệp. Những kinh nghiệm, kỹ năng, tác phong nghiên cứu khoa học của thầy sẽ giúp ích cho tôi rất nhiều trong công việc cũng như trong cuộc sống.

Tôi xin cảm ơn các anh chị cán bộ viện ICSE đã tạo điều kiện thuận lợi để tôi nghiên cứu và hoàn thành đồ án.

Tôi xin cảm ơn bạn Phan Công Huân cùng các bạn khóa K57 đang thực tập tại viện ICSE đã giúp đỡ, trao đổi kiến thức để tôi có thể hoàn thành đồ án.

Và cuối cùng, con xin gửi lời cảm ơn đến bố mẹ, anh chị những người luôn đi bên cạnh động viên, truyền cho con động lực để con có thể hoàn thành đồ án tốt nghiệp này.

Hà Nội, ngày 23 tháng 12 năm 2016
Bùi Ngọc Luân
Lớp CNTT-TT 2.03 K57
Viện CNTT-TT, Đại học BKHN

Mục Lục

TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP	2
LỜI CẢM ƠN	3
Danh mục ảnh	6
Chương I, Chương mở đầu	7
1. Giới thiệu về Internet of Things.....	7
1.1 IoT là gì ?	7
1.2 Tiềm năng của Internet of Things	8
1.3 Sự bùng nổ của IoT	8
1.4 Các lớp công nghệ IoT	9
2. IoT Platform là gì ?	10
2.1 Định nghĩa.....	10
2.2 Tính chất	10
3, Khảo sát một số nền tảng IoT.	10
3.1. OpenHAB	11
3.2. Home-Assistant	14
3.3. Eclipse Kura.....	15
3.4 Một số IoT Platform khác	17
4, Đặt vấn đề	19
4.1 Đưa ra bài toán.....	19
4.2 Tiêu chuẩn cho IoT Platform	20
4.3 Giải pháp cho việc quản lí tập trung các thiết bị IoT.....	21
4.4 Mục đích của đồ án.....	22
5. Phạm vi của đồ án và các công cụ được sử dụng	22
5.1 Phạm vi của đồ án.....	22
5.2 Python2.7/Flask	22
5.3 Angular2	24
5.4 InfluxDB	24
5.5 Git và GitHub.....	24
5.6 Docker.....	25
6. Tóm tắt chương	25
Chương II, Các kết quả đạt được	27
1, Kiến trúc hệ thống.....	27
1. 1 IoT Platform.....	28
1.2 State Service	28
1.3 Rules Service	28

1.4 RESTful API.....	29
1.5 Web Interface.....	29
2, Phân tích chức năng.....	30
2.1 Các chức năng của hệ thống.	30
2.2 Danh sách các REST API do hệ thống cung cấp.....	32
2.3 Mô tả các ca sử dụng	32
3. Thiết kế lớp.....	41
3.1 Chi tiết các lớp.....	42
3.2 Tính đa hình trong thiết kế lớp	44
4, Phân tích hành vi người dùng	44
4.1 Thu thập dữ liệu	45
4.2 Truy vấn tất cả thông tin của thiết bị	45
4.3 Truy vấn thông tin của một thiết bị.....	46
4.4 Truy vấn lịch sử trạng thái của thiết bị	46
4.5 Thiết lập trạng thái cho thiết bị.....	47
4.6 Thiết lập trạng thái cho thiết bị theo kiểu thiết bị.....	48
5, Thiết kế cơ sở dữ liệu	48
6, Thiết kế giao diện đồ họa.....	49
7, Tính khả mở của hệ thống	50
Chương III, Kiểm thử hệ thống	51
1. Cài đặt môi trường kiểm thử.....	51
2. Tiến hành kiểm thử	52
2.1 Xem dữ liệu	52
2.2 Bật/Tắt thiết bị	53
2.3 Xem lịch sử trạng thái thiết bị.....	53
2.4 Chạy kịch bản tự động 1	53
2.5 Chạy kịch bản tự động 2	54
Chương VI, Kết luận và hướng phát triển	55
1 Kết luận.....	55
2. Hướng phát triển	55
2.1 Tích hợp thêm nều IoT Platform	55
2.2 Phát triển hệ thống AI.....	55
Tài liệu Tham khảo.....	56

Danh mục ảnh

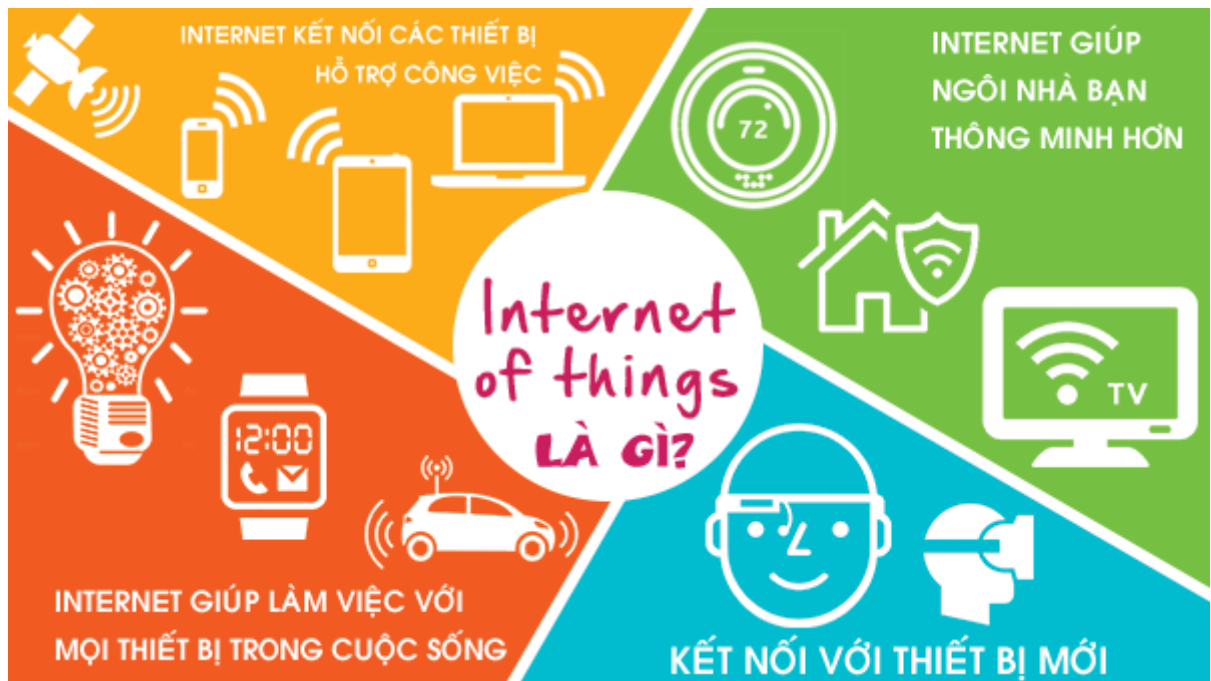
Hình 1. Internet of things là gì ?.....	7
Hình 2. Ba lớp công nghệ IoT.....	9
Hình 3. Giao diện điều khiển của OpenHAB.....	11
Hình 4. Trạng thái các item của OpenHAB.....	13
Hình 5. Trạng thái của 1 item chỉ định.....	13
Hình 6. Giao diện điều khiển của Home-Assitant.....	14
Hình 7. Giao diện quản lí MQTT Service Kura.....	16
Hình 8. Giao diện quản lí Cloud Service của Kura.....	16
Hình 9. Nhà thông minh quản lí bởi OpenHAB.....	19
Hình 10. Hệ thống trong phòng ngủ do Home-Assitant quản lý.....	20
Hình 11. 14 trong số hơn 300 IoT Platforms.....	21
Hình 12. So sánh docker với virtual machine.....	25
Hình 13. Kiến trúc hệ thống.....	27
Hình 14. Usecase tổng quan.....	30
Hình 15. Usecase xem trạng thái thiết bị.....	32
Hình 16. Usecase bật thiết bị trên giao diện.....	33
Hình 17. Uscase tắt thiết bị trên giao diện.....	34
Hình 18. Usecase xem biểu đồ lịch sử của thiết bị.....	34
Hình 19. Usecase thêm kịch bản.....	35
Hình 20. Usecase xóa kịch bản.....	36
Hình 21. Usecase truy vấn thông tin các thiết bị qua API.....	37
Hình 22. Usecase truy vấn thông tin một thiết bị.....	38
Hình 23. Usecase truy vấn lịch sử của thiết bị.....	39
Hình 24. Usecase thay đổi trạng thái qua API.....	40
Hình 25. Biểu đồ lớp của hệ thống.....	41
Hình 26. Sequence Diagram thu thập dữ liệu.....	45
Hình 27. Sequence Diagram truy vấn thông tin thiết bị.....	45
Hình 28. Sequence Diagram Truy vấn thông tin một thiết bị.....	46
Hình 29. Sequence Diagram Truy vấn lịch sử trạng thái của thiết bị.....	46
Hình 30. Sequence Diagram Thiết lập trạng thái cho thiết bị.....	47
Hình 31. Sequence Diagram thiết lập trạng thái cho thiết bị theo kiểu thiết bị.....	48

Chương I, Chương mở đầu

1. Giới thiệu về Internet of Things

1.1 IoT là gì ?

Mạng lưới vạn vật kết nối Internet hoặc là mạng lưới thiết bị kết nối với Internet viết tắt là IoT (tiếng Anh Internet of Things) là một kịch bản về một thế giới kết nối khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của các thiết bị vật lý, các thiết bị kết nối, cơ sở hạ tầng, công nghệ vi cơ điện tử, phần mềm, cảm biến, kết nối mạng và nhiều yếu tố khác cho phép các thiết bị có thể thu thập và trao đổi dữ liệu. Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài [1]



Hình 1. Internet of things là gì ?^(nguồn 1)

Công nghệ IoT là một hệ thống mạng vật lý, mà dựa vào các thiết bị truyền động và cảm biến để thu thập dữ liệu từ thế giới thực. Các hệ thống lớn bao gồm nhiều thành phần trong IoT như là thành phố thông minh, nhà thông minh, giao thông thông minh và ngôi nhà thông minh. Các thiết bị IoT (ví dụ như cảm biến, ...) có thể được tích hợp qua các hệ thống máy tính nhúng và kết nối tới cơ sở hạ tầng Internet hiện có. Theo Cisco [2], sẽ có khoảng 50 tỷ thiết bị IoT vào năm 2020.

¹ Iotvietnam.com.vn

1.2 Tiềm năng của Internet of Things

Kevin Ashton, đồng sáng lập và là giám đốc điều hành của Trung tâm Auto-ID tại Viện công nghệ Massachusetts, đã đề cập đến Internet of Things lần đầu tiên tại buổi thuyết trình ở công ty Procter & Gamble [3]. Và đây là các Ashton đã giải thích về tiềm năng của Internet of Things:

"Ngày nay máy tính, và Internet, hầu như hoàn toàn phụ thuộc vào con người mới có thông tin. Gần như tất cả trong số khoảng 50 petabyte (1 petabyte là 1.024 terabyte) dữ liệu trên Internet lần đầu tiên được con người nắm và tạo ra bằng cách đánh máy, nhấn nút ghi âm, chụp ảnh hoặc quét mã vạch.

Vấn đề là, con người rất hạn chế về thời gian, sự chú ý và chính xác – nghĩa là con người không được tốt lắm trong việc lưu giữ dữ liệu về mọi thứ trong thế giới. Nếu chúng ta có những chiếc máy tính biết mọi thứ - sử dụng được dữ liệu chúng thu thập mà không cần sự giúp đỡ của con người – thì chúng ta sẽ có thể theo dõi và đếm mọi thứ, điều này sẽ giúp giảm rất lớn sự lãng phí, thất bại và chi phí. Chúng ta sẽ biết khi nào mọi thứ cần thay thế, sửa chữa hoặc phục hồi và liệu chúng còn có thể còn tiếp tục hoạt động hay hoạt động tốt nhất nữa không".

1.3 Sự bùng nổ của IoT

Khái niệm IoT được ra đời từ năm 1999, nhưng trong vòng vài năm trở lại IoT đang dẫn đầu 1 xu hướng, xu hướng kết hợp giữa các hệ thống ảo và thực thể, vạn vật kết nối Internet (IoT) và các hệ thống kết nối Internet (Internet of System - IoS). Xu hướng này có tác động không nhỏ tới nền kinh tế thế giới, cũng như cách mà con người đang làm việc. Một số người gọi đây là **cuộc cách mạng công nghiệp lần thứ 4**.

Song hành cùng sự phát triển của IoT luôn có sự góp mặt của các ông lớn công nghệ trong tất cả các lĩnh vực liên quan

- Platform:

- + IBM với IBM Watson IoT
- + Amazon với AWS internet of things
- + Intel với Intel IoT Platform
- + Microsoft với Azure IoT Suite
- + Samsung ARTIK Smart IoT Platform
- + Rất nhiều các sản phẩm IoT Platform mã nguồn mở được hỗ trợ mạnh mẽ bởi cộng đồng như OpenHab, Home-Assistant, OM2M, Kura...

- Hardware (sensor, Kit):

- + Apple với Apple Home Kit

- + Samsung với SmartThings Starter Kit
- + Google với Google Brillo
- + Intel với Intel IoTivity
- + Qualcomm AllJoyn cho đến UPnP Forum, ARM mbed và nhiều tay đua mới nổi khác.
- + Cùng với đó là sự ra đời của rất vô số loại sensor được tích hợp vào trong các sản phẩm thân thuộc đời thường như TV, Tủ Lạnh, Quạt điện, Ổ cắm.....

Qua những ví dụ kể trên, chúng ta có thể thấy được sự chú ý, quan tâm của thế giới trong ngành công nghiệp IoT và có cái nhìn cơ bản về sự phát triển của IoT trong vài năm trở lại đây.

1.4 Các lớp công nghệ IoT

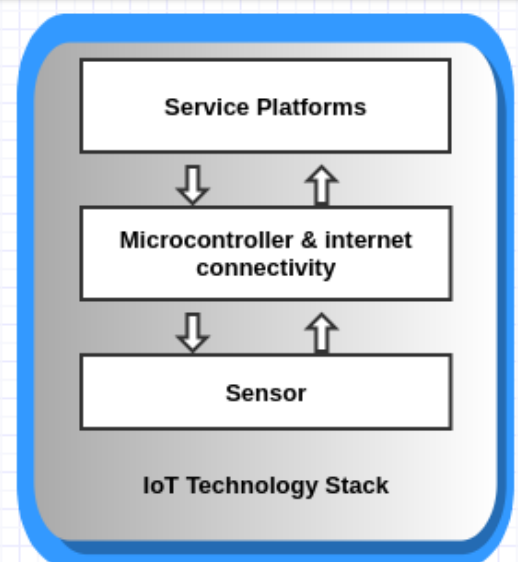
Các lớp công nghệ IoT là một mô hình gồm các lớp của một hệ thống IoT. Trong đó định nghĩa ra, chức năng hoạt động của mỗi lớp và cách lớp giao tiếp với nhau. Các lớp công nghệ IoT gồm có 3 lớp cơ bản: sensor, microcontrollers and internet connectivity and service platforms.

Lớp 1: Các sensor được nhúng vào các đối tượng vật lý để thu thập dữ liệu, điều khiển các đối tượng theo ý muốn.

Lớp 2: Cho phép lưu trữ, xử lý dữ liệu, và kết nối tới internet.

IoT cần kết nối internet để có thể truyền dữ liệu từ sensor tới các đám mây hoặc các trung tâm xử lý dữ liệu. Bởi vì có rất nhiều cảm biến tạo ra rất nhiều dữ liệu (10000 điểm dữ liệu/s) [4]. Nên lớp này có ý nghĩa là nơi tiền xử lý dữ liệu trước khi gửi đi để giảm thiểu khối lượng dữ liệu không cần thiết.

Lớp 3: Service Platform là lớp cung cấp cho người dùng những dịch vụ để người dùng có thể tiếp cận được tới những giá trị đã được khai thác từ việc thu thập và phân tích dữ liệu.



Hình 2. Ba lớp công nghệ IoT

2. IoT Platform là gì ?

2.1 Định nghĩa

Để có thể triển khai các mô hình IoT, chúng ta cần một hạ tầng công nghệ mà trung tâm của nó là các nền tảng IoT. Nền tảng IoT được định nghĩa là nền tảng mà ở đó nó thực hiện các chức năng chính:

- Triển khai các ứng dụng, trình giám sát, quản lý, và kiểm soát các thiết bị đã kết nối.
- Thu thập dữ liệu từ xa từ các thiết bị kết nối.
- Kết nối độc lập và an toàn giữa các thiết bị.
- Quản lý các cảm biến/thiết bị

2.2 Tính chất

Từ định nghĩa trên, ta có thể lọc ra được 8 tính chất mà IoT Platform cần phải có.

- Kết nối và chuẩn hóa: Đem các giao thức và định dạng dữ liệu khác nhau vào trong một “phần mềm” trong khi vẫn đảm bảo chính xác luồng dữ liệu và tương tác tốt với tất cả thiết bị.
- Quản lý thiết bị: Liên tục chạy các bản vá lỗi và cập nhật phần mềm sao cho đảm bảo các thiết bị đã kết nối đều hoạt động tốt.
- Cơ sở dữ liệu: Thu thập và lưu trữ dữ liệu từ các thiết bị đã kết nối.
- Quản lý các hành động: cho phép thiết lập các quy tắc để có thể thực hiện các hành động thông minh dựa trên dữ liệu đã thu thập được.
- Phân tích: Thực hiện một loạt các phân tích phức tạp từ các dữ liệu đã thu thập được để có thể đưa ra những thông tin giá trị.
- Hình ảnh: Cho phép người dùng có thể xem trạng thái thiết bị, các biểu đồ trực quan thông qua các biểu đồ.
- Các công cụ khác: Cho phép các lập trình viên có thể phát triển các plugin, để tạo nên hệ sinh thái phong phú và đa dạng hơn.
- Tính khả mở: Tích hợp được với các hệ thống bên thứ 3, thông qua API hoặc các SDK, gateways.

3, Khảo sát một số nền tảng IoT.

Sau khi xác định được mục đích của đề án, đề án tiến hành khảo sát một số nền tảng IoT tiêu biểu .

3.1. OpenHAB ⁵

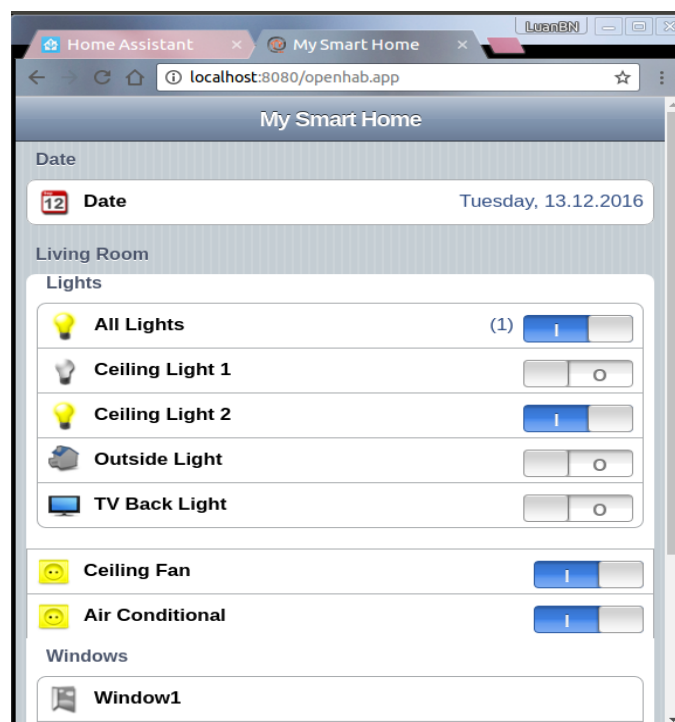
a, Giới thiệu về OpenHab

OpenHAB là một IoT Platforms mã nguồn mở dùng cho hệ thống nhà thông minh cho phép người dùng tích hợp các công nghệ khác nhau thành một hệ thống duy nhất. Openhab cho phép thiết lập các quy tắc tự động hóa trong hệ thống một cách mềm dẻo và cung cấp cho người dùng một giao diện điều khiển thống nhất.

OpenHab định nghĩa ra một khái niệm gọi là **Item**. Một Item đơn vị nhỏ nhất của hệ thống. OpenHAB không quan tâm tới giá trị của Item đến từ các thiết bị vật lí, hay dữ liệu ảo hóa được gửi từ web hoặc các dữ liệu mô phỏng. Tất cả các thông tin liên quan tới item đều nằm ở tầng trừu tượng mà openHAB định nghĩa ra. Nghĩa là chúng ta không thể truy vấn trực tiếp tới các thông tin liên quan của thiết bị kết nối trực tiếp với item như địa chỉ IP, ID...đây có thể là một điểm trừ nhưng cũng là điểm cộng khi làm cho người sử dụng có thể dễ dàng thay đổi công nghệ ở phía dưới mà không cần thay đổi các quy tắc và giao diện.

b, Các tính năng của OpenHab

- Có thể chạy trên bất kì thiết bị nào có môi trường JVM (Linux, Mac, Window).
- Cho phép dễ dàng tích hợp các công nghệ (sensor, device) của các hãng khác nhau vào một sản phẩm hoàn thiện
- Có giao diện điều khiển nền web, và phiên bản mobile dành cho iOS và Android.
- Hoàn toàn là mã nguồn mở.
- Cung cấp REST API để có thể tích hợp với bên thứ 3.



Hình 3. Giao diện điều khiển của OpenHAB

c, Cơ chế binding của OpenHAB

Binding là các gói tùy chọn có thể giúp mở rộng chức năng của OpenHAB. Binding cung cấp cơ chế tích hợp công nghệ nhà thông minh và các thiết bị với nhiều bundle khác nhau hỗ trợ tích hợp và giao tiếp với các mạng xã hội, nhắn tin, điện toán đám mây của các nền tảng IoT và rất nhiều các thành phần khác. Với sự hỗ trợ của cơ chế bindings, người dùng openHAB có thể dễ dàng điều khiển Samsung Smart TV hoặc điều hòa Daikin và còn nhiều nhiều các sản phẩm khác được OpenHAB hỗ trợ.

Mỗi công nghệ hoặc thiết bị, mạng xã hội hay nền tảng điện toán đám mây tích hợp vào OpenHAB đều được hỗ trợ bởi các gói cụ thể. Các gói này là tùy chọn và có thể dễ dàng thêm vào OpenHAB khi cần.

3 Bước cài đặt:

- Download gói dữ liệu để binding vào trong thư mục cố định.
- Thiết lập cấu hình cho binding trong file openhab.cfg.
- Thiết lập các hành động và các luật cho item vừa mới binding.

d, REST API của OpenHab

REST API của openhab phục vụ cho nhiều mục đích khác nhau. Nó có thể được sử dụng để kết nối OpenHAB với các hệ thống khác vì nó cho phép đọc trạng thái của các items, cũng như cho phép cập nhật trạng thái của items bằng cách gửi các lệnh tới items.

REST API của openHAB rất nhanh, vì vậy nó có thể sử dụng cho các thành phần tương tác thời gian thực với openHAB, đặc biệt là từ giao diện của bên thứ 3.

Trong REST API của openHAB cung cấp 3 giao diện chính để tương tác:
Send Command: gửi một lệnh để thực hiện một số hành động (bật/tắt đèn).
Send Status: chỉ ra trạng thái của một số item đã thay đổi.
Get Status: lấy về trạng thái hiện tại của một item.

Ví dụ:

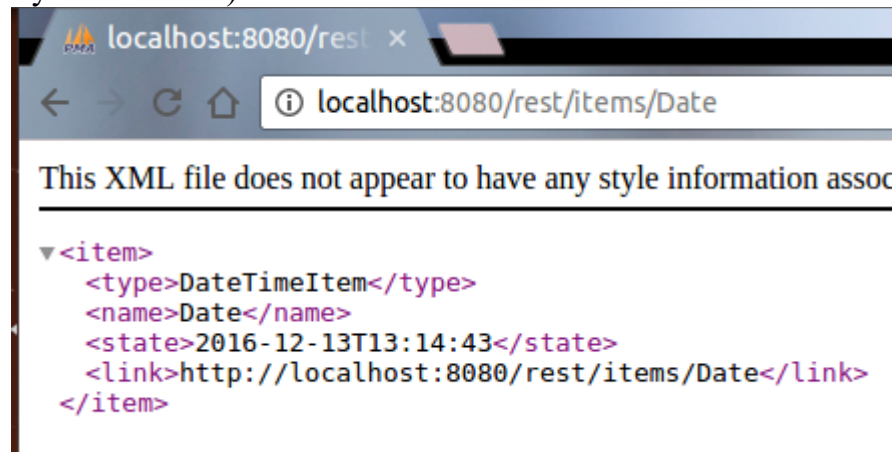
- ❖ Truy vấn trạng thái của tất cả sensor được thiết lập trong OpenHAB
+ Vào đường link sau bằng trình duyệt:
 - <http://{host}:{port}/rest/states>

Dữ liệu sẽ được trả về ở dạng xml như hình dưới:

```
1  [
2  {
3    "attributes": {
4      "assumed_state": true,
5      "auto": true,
6      "entity_id": [
7        "light.garage_outside_light"
8      ],
9      "friendly_name": "all lights",
10     "hidden": true,
11     "order": 1
12   },
13   "entity_id": "group.all_lights",
14   "last_changed": "2016-12-13T13:18:34.692337+00:00",
15   "last_updated": "2016-12-13T13:18:34.692337+00:00",
16   "state": "on"
17 },
18 {
19   "attributes": {
20     "assumed_state": true,
21     "friendly_name": "Ventilators",
22     "icon": "mdi:fan"
23   },
24   "entity_id": "switch.ventilators",
25   "last_changed": "2016-12-13T13:18:33.639397+00:00",
26   "last_updated": "2016-12-13T13:18:33.639397+00:00",
27   "state": "on"
28 },
29 {
30   "attributes": {
31     "assumed_state": true,
32     "friendly_name": "Garage outside light",
33     "icon": "mdi:spotlight",
34     "supported_features": 0
35   },
36   "entity_id": "light.garage_outside_light",
37   "last_changed": "2016-12-13T13:18:34.689111+00:00",
38   "last_updated": "2016-12-13T13:18:34.689111+00:00",
39   "state": "on"
40 },
41 ]
```

Hình 4. Trạng thái các item của OpenHAB

- ❖ Lấy trạng thái của 1 item chỉ định:
(ở đây là item Date)



```
<item>
  <type>DateTimeItem</type>
  <name>Date</name>
  <state>2016-12-13T13:14:43</state>
  <link>http://localhost:8080/rest/items/Date</link>
</item>
```

Hình 5. Trạng thái của 1 item chỉ định

- ❖ Ngoài ra, việc cài đặt và cập nhật trạng thái cho sensor được thực hiện thông qua phương thức POST và PUT
 - o `curl --header "Content-Type: text/plain" --request PUT --data "OFF" http://192.168.100.21:8080/rest/items/MyLight/state`
 - o `curl --header "Content-Type: text/plain" --request POST --data "ON" http://192.168.100.21:8080/rest/items/MyLight`

3.2. Home-Assistant ⁶

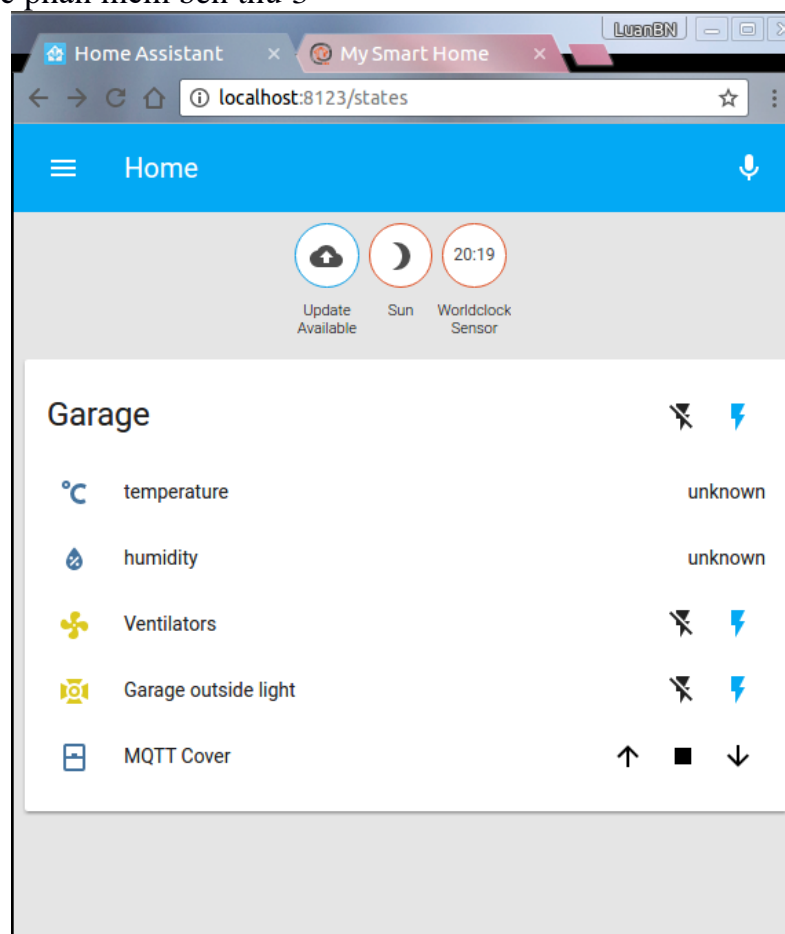
a, Giới thiệu về Home-Assitant

Home-Assistant là một nền tảng nhà thông minh chạy trên nền Python 3. Mục tiêu của Home-Assistant là có thể theo dõi và kiểm soát tất cả các thiết bị ở trong nhà và cung cấp cho người dùng một nền tảng điều khiển tự động.

Home-Assistant quan niệm rằng hệ thống nhà thông minh thì nên chạy tại nhà, chứ không phải chạy trong đám mây. Chính vì lí do đó, Home-Assitant hỗ trợ rất nhiều các thiết bị IoT (484 các loại thiết bị và nền tảng) đảm bảo việc triển khai hệ thống nhà thông minh diễn ra một cách dễ dàng nhất. Mỗi thiết bị được kết nối tới Home-Assistant gọi là một component. Mỗi component sẽ được config theo một cách chuyên biệt trong file config.

b, Các tính năng của Home-Assistant

- Cho phép chạy trên các môi trường có cài đặt Python 3.5 trở lên (Window, Mac, Linux).
- Hỗ trợ tích hợp gần 500 các loại thiết bị và giao thức.
- Giao diện thân thiện với người dùng (có bản web và mobile).
- Hỗ trợ tính năng push notification.
- Cho phép đặt ra các luật tự động trong nhà thông minh.
- Hỗ trợ REST API và có thể giao tiếp qua MQTT để có thể dễ dàng giao tiếp với các phần mềm bên thứ 3



Hình 6. Giao diện điều khiển của Home-Assistant

c, Config trong Home-Assistant

Trong Home-Assistant, hầu hết các cài đặt đều được để ở trong file `configuration.yaml`. File này chứa toàn bộ các component sẽ được load lên và config riêng của các component này.

Ví dụ, để cấu hình một Philips Hue Light vào trong Openhab thì chỉ cần thêm config

```
# Example configuration.yaml entry
light:
  platform: hue
  host: DEVICE_IP_ADDRESS
```

Nhưng để config một bóng đèn đi kèm trong bộ kit Z-wave thì cần phải thực hiện qua tương đối nhiều bước như config Z-wave platform trước (cần cài đặt mạng, cài đặt thêm driver..) sau đó mới config đến Z-wave light.

d, REST API của Home Assistant

Home-Assistant hỗ trợ REST API trả dữ liệu về dưới dạng JSON để người dùng có thể:

- Kiểm tra xem API có chạy không.
- Lấy các về config của các component.
- Truy vấn các hành động của người dùng.
- Lấy về trạng thái của các đối tượng đang hoạt động.
- Lấy về trạng thái của một sensor đang hoạt động
- Truy cập lịch sử trạng thái của 1 component chỉ định.
- Cập nhật hoặc cập nhật trạng thái của một component.

Ví dụ về việc truy vấn dữ liệu trong Home-Assistant

Lấy về toàn bộ trạng thái của các component đang hoạt động.

Lấy về trạng thái của một sensor chỉ định.

3.3. Eclipse Kura⁷

a, Giới thiệu về Kura

Eclipse Kura là một dự án Eclipse IoT cung cấp một nền tảng cho việc xây dựng các IoT Gateway. Nó là một tập hợp các ứng dụng thông minh cho phép quản lý các gateway từ xa và cung cấp một loạt các API để có thể hỗ trợ người dùng triển khai ứng dụng IoT của riêng họ.

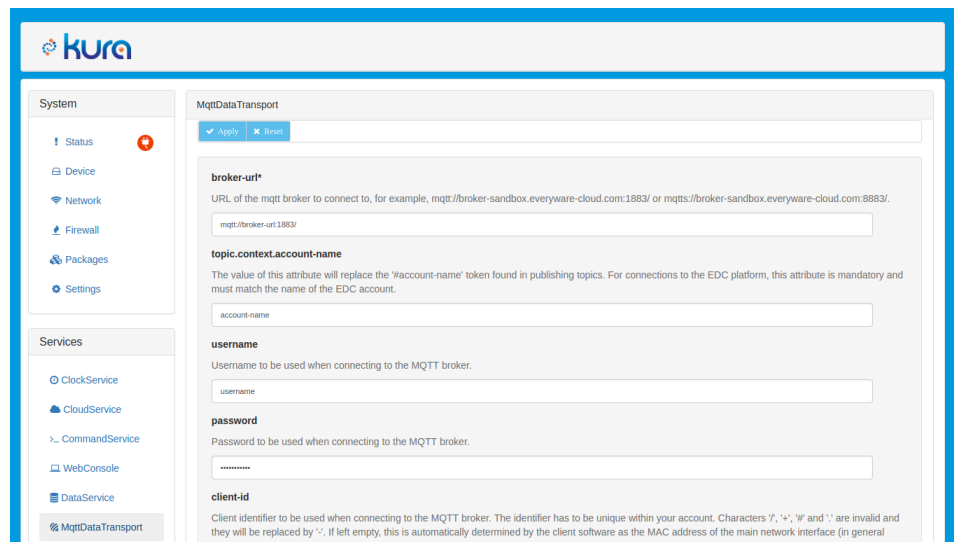
(IoT gateway có vai trò tổng hợp dữ liệu của cảm biến, thông dịch giữa các giao thức của các cảm biến khác nhau, xử lý các dữ liệu này trước khi gửi đi).

b, Tính năng của Kura

Kura chạy trên môi trường Java Virtual Machine (JVM) và cung cấp một số chức năng chính sau:

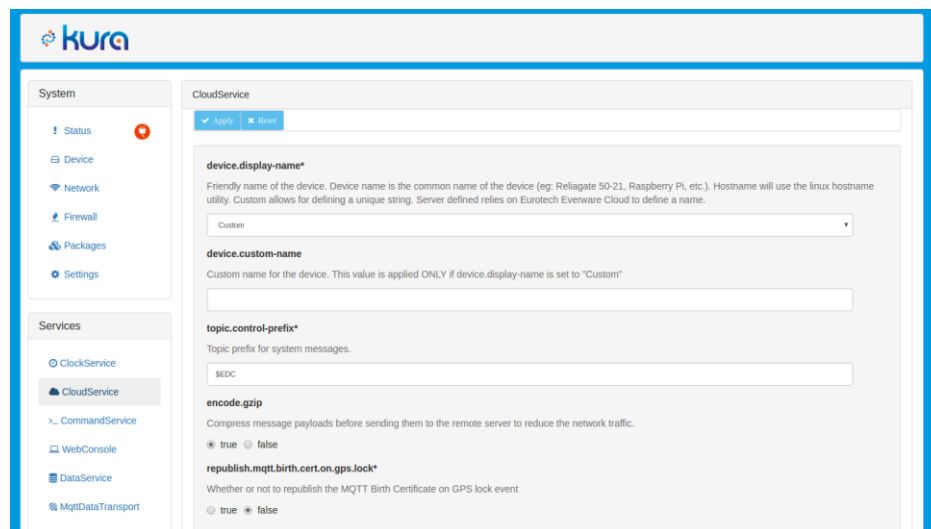
- Quản lí từ xa:
 - Cho phép quản lý từ xa các ứng dụng IoT được cài đặt trong Kura, bao gồm các dự án đã triển khai, nâng cấp, quản lý cấu hình và các dịch vụ đám mây.
- Networking:
 - Cung cấp các API để cấu hình cho các network có sẵn ở gateway như Ethernet, Wifi hay modem di động.
- Giao diện quản trị web
 - Cung cấp một trang web để quản lý các gateway.

Giao diện quản lí, kết nối tới MQTT Service của Kura



Hình 7. Giao diện quản lí MQTT Service Kura

Giao diện quản lí, kết nối tới Cloud Service



Hình 8. Giao diện quản lí Cloud Service của Kura

c, REST API của Kura

Kura quản lý các gateway bằng các bundle giống như openHAB. Tuy nhiên, Kura phép người đọc tự build các bundle của mình dựa trên các package mà Kura đã cho sẵn và có thể tích hợp các bundle đó vào Kura dưới hình thức là một OSGi plugin. Chính vì thế, bạn có thể tự xây dựng một REST API cho mình dựa trên các thư viện có sẵn của Kura.

3.4 Một số IoT Platform khác

Ngoài những IoT Platform được tìm hiểu chi tiết ở trên, em đã tiến hành khảo sát và thống kê tính chất, đặc điểm của một số IoT Platform khác.

Tất cả được trình bày ở trong bảng sau [8]:

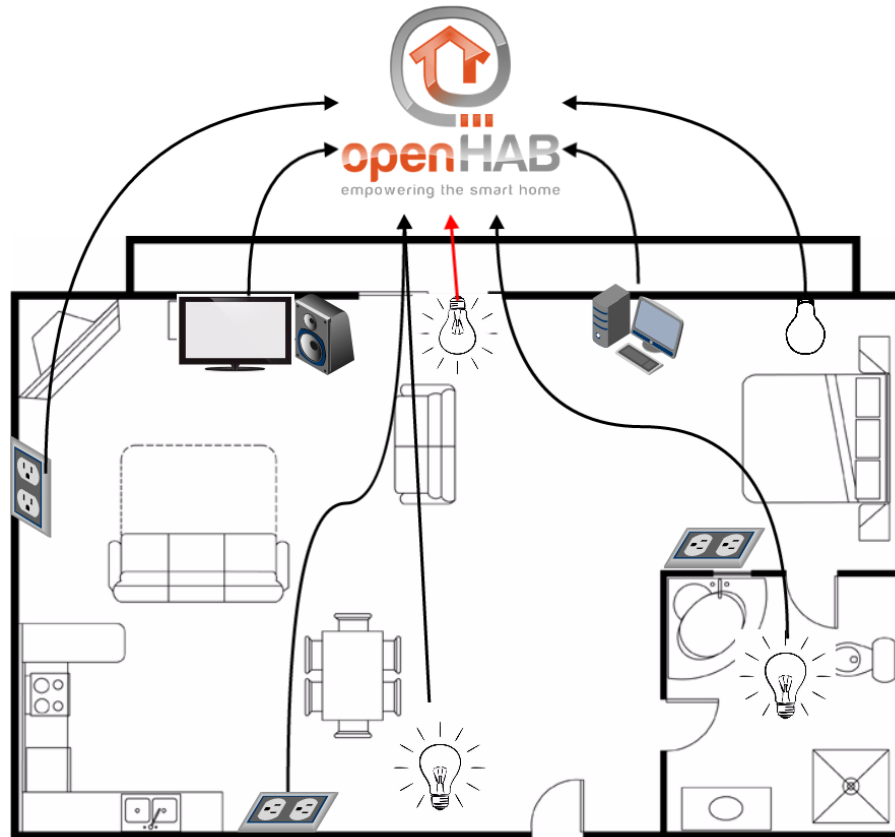
Tên IoT Platform	Có chức năng quản lý thiết bị ?	Kết nối	Các giao thức thu thập dữ liệu	Các kiểu phân tích	Hỗ trợ hiển thị ?
2lemetry - IoT Analytics Platform (acquired by AWS IoT)	Có	Salesforce, Heroku, ThingWorx APIs	MQTT, CoAP, STOMP, M3DA	Real-time analytics (Apache Storm)	Không có
Appcelerator	Không có	REST API	MQTT, HTTP	Real-time analytics (Titanium [1])	Có (Titanium UI Dashboard)
AWS IoT platform	Có	REST API	MQTT, HTTP1.1	Real-time analytics (Rules Engine, Amazon Kinesis, AWS Lambda)	Có (AWS IoT Dashboard)
Bosch IoT Suite - MDM IoT Platform	Có	REST API	MQTT, CoAP, AMQP, STOMP	Không rõ	Có (User Interface Integrator)
Ericsson Device Connection Platform (DCP) - MDM IoT Platform	Có	REST API	CoAP	Không rõ	Không có
EVERYTHING - IoT Smart Products Platform	Không có	REST API	MQTT, CoAP, WebSockets	Real-time analytics (Rules Engine)	Có (EVERYTHING IoT Dashboard)

IBM IoT Foundation Device Cloud	Có	REST and Real-time APIs	MQTT, HTTPS	Real-time analytics (IBM IoT Real-Time Insights)	Có (Web portal)
ParStream - IoT Analytics Platform (acquired by Cisco)	Không có	R, UDX API	MQTT	Real-time analytics, Batch analytics (ParStream DB)	Có (ParStream Management Console)
PLAT.ONE - end-to-end IoT and M2M application platform	Có	REST API	Link Encryption (SSL), Identity Management (LDAP)	MQTT, SNMP	Không rõ
ThingWorx - MDM IoT Platform	Có	REST API	MQTT, AMQP, XMPP, CoAP, DDS, WebSockets	Predictive analytics(ThingWorx Machine Learning), Real-time analytics (ParStream DB)	Có (ThingWorx SQUEAL)
Xively- PaaS enterprise IoT platform	Không có	REST API	HTTP, HTTPS, Sockets/ Websocket, MQTT	Không rõ	Có (Management console)

4, Đặt vấn đề

4.1 Đưa ra bài toán

Đồ án hình thành từ giả thiết một ngôi nhà gồm nhiều phòng, trong đó phòng khách được lắp đặt hệ thống nhà thông minh như hình dưới.



Hình 9. Nhà thông minh quản lí bởi OpenHAB

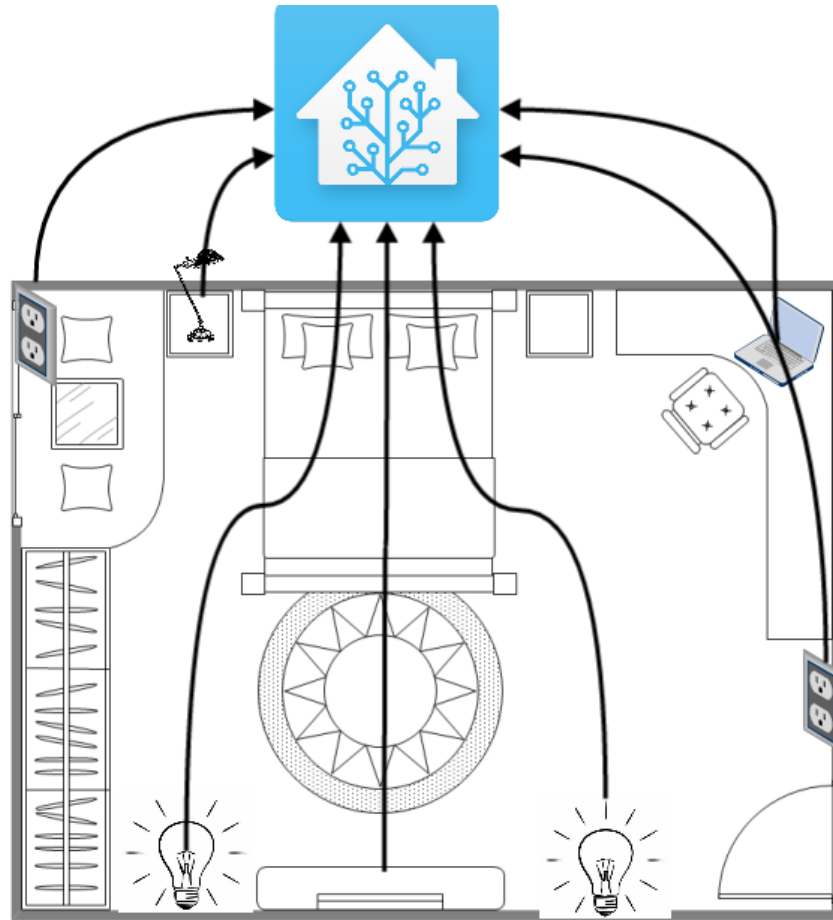
Ở đây, phía lắp đặt đã sử dụng **OpenHAB** để quản lí các thiết bị thông minh như:

- Đèn chiếu sáng
- Nguồn điện, công tắc điện
- Tivi, loa
- Dàn máy tính

Sau hai năm, chủ nhà có mong muốn lắp đặt hệ thống thông minh trong căn phòng ngủ và tích hợp nó vào hệ thống có sẵn. Tuy nhiên lúc này OpenHAB đã không được hỗ trợ nữa, và phía lắp đặt chuyển sang sử dụng IoT Platform khác là Home-Assistant. Khi này có 2 vấn đề chính mà người triển khai phải đối mặt:

- Home-Assistant không hỗ trợ các sensor cũ trong hệ thống.
- OpenHAB không hỗ trợ các sensor mới.

Nên chủ nhà đã đồng ý lắp đặt một hệ thống hoàn toàn mới trong phòng ngủ sử dụng Home-Assitant như hình dưới:



Hình 10. Hệ thống trong phòng ngủ do Home-Assitant quản lý

Vấn đề nảy sinh ở đây là trong một căn nhà mà bạn phải điều khiển thông qua 2 phần mềm khác nhau (2 điều khiển khác nhau). Dẫn đến sự khó khăn trong điều khiển và giảm chất lượng trải nghiệm của bạn khi sử dụng nhà thông minh. Và nghĩ xa hơn là trong một đến hai năm nữa bạn lại coi nói nhà của mình, thì biết được lúc ấy IoT platform nào sẽ được sử dụng để lắp đặt trong những căn phòng mới đó.

Và từ ví dụ ở trên, ta đặt ra câu hỏi: liệu có cách nào tích hợp nhiều IoT Platform lại với nhau để người dùng có thể dễ dàng điều khiển được căn nhà của mình thông qua một nền tảng duy nhất.

4.2 Tiêu chuẩn cho IoT Platform

Tính đến ngày hôm nay, đã có hơn 300 Iot Platform[9] và con số ấy đang tiếp tục tăng lên. Tuy nhiên, không phải mọi nền tảng đều như nhau, mỗi nền tảng đang được hình thành từ các chiến lược thị trường của các công ty khác nhau để cố gắng tận dụng hết tiềm năng của IoT. Các Startups sáng tạo, các nhà sản xuất phần cứng, thiết bị mạng định đám, đến các công ty quản lý di động đều cạnh tranh để nền tảng

IoT của họ trở thành nền tảng IoT tốt nhất trên thị trường.



Hình 11. 14 trong số hơn 300 IoT Platforms

IoT đang trên đà phá triển, tuy nhiên tại triển lãm khoa học và công nghệ CES 2016 được tổ chức tại Las Vegas (Mỹ), khi các thiết bị Internet of Things ra mắt thì người ta đặt câu hỏi làm sao có thể kết nối và đem tất cả các thiết bị thông minh gộp chung lại và sử dụng cùng nhau.

Mỗi một sản phẩm được phát triển một công ty khác nhau, trong khi vẫn chưa có một chuẩn chung nào cho tất cả các thiết bị làm cho việc tích hợp các thiết bị này với nhau rất khó khăn. Một sensor có thể hoạt động tốt với IoT platform này nhưng có thể không thể tích hợp với các IoT Platform khác. Trong khi các sản phẩm mới ra liên tục, dẫn đến sự tích hợp, thay đổi cập nhật và phát triển các sản phẩm IoT vướng phải rất nhiều khó khăn.

Lấy ví dụ như xe ô tô chẳng hạn. Một chiếc Ford Focus có thể giao tiếp cực kì tốt đến các dịch vụ và trung tâm dữ liệu của Ford khi gửi dữ liệu lên mạng. Nếu một bộ phận nào đó cần thay thế, hệ thống trên xe sẽ thông báo về Ford, từ đó hãng tiếp tục thông báo đến người dùng. Nhưng trong trường hợp chúng ta muốn tạo ra một hệ thống cảnh báo kẹt xe thì mọi chuyện rắc rối hơn nhiều bởi xe Ford được thiết lập chỉ để nói chuyện với server của Ford, không phải với server của Honda, Audi, Mercedes hay BMW. Lý do cho việc giao tiếp thất bại? Chúng ta thiếu đi một ngôn ngữ chung. Và để thiết lập cho các hệ thống này nói chuyện được với nhau thì rất tốn kém, đắt tiền.

4.3 Giải pháp cho việc quản lý tập trung các thiết bị IoT

Quản lý tài nguyên hiệu quả trong hệ thống IoT là bắt buộc nếu muốn khai thác nguồn tài nguyên từ các thiết bị này. Chính vì lí do đó, đã có nhiều công trình nghiên cứu đi theo hướng “IoT resource modeling”, tức là đưa ra các mô hình quản lý tài nguyên IoT.

[10] Trong [11] mối quan hệ giữa các sự vật tự nhiên, cảm biến và các thiết bị

truyền thông đã được mô hình hóa, Oteafy et al. [12] xác định mô hình cơ bản cho các đối tượng IoT và cách mà các nguồn tài nguyên có thể chia sẻ bởi nhiều ứng dụng. Zhang et al. [13] mô hình các đối tượng IoT thành các nguồn lực có thể được kết nối tới các dịch vụ cung cấp các chức năng quản lý cần thiết, Bezazzouz et al [14] đề xuất trong dự án ClouT về mô hình thông tin cho IoT với các nhóm tài nguyên, dịch vụ và thiết bị. Tất cả các phương pháp trên đều cung cấp những mô hình khá chi tiết cho thiết bị IoT, nhưng không có phương pháp nào xem xét cách tiếp cận bằng cách xây dựng một hệ thống các lớp trừu tượng chứa các phương thức chung để phủ lên tất cả các mô hình IoT như trong đề án này.

Công việc nghiên cứu và phát triển đang diễn ra từng ngày, mỗi mô hình đều có điểm mạnh, điểm yếu của nó. Tuy nhiên mỗi công trình nghiên cứu được đưa ra đều đóng góp một khối lượng tri thức không nhỏ để có thể xây dựng lên một hệ thống quản lý IoT hoàn thiện.

4.4 Mục đích của đề án

Chính từ tư duy ở trên, đề án được thực hiện với mục đích tạo ra một nền tảng cho phép các IoT Platforms có thể kết hợp lại với nhau. Cung cấp cho người dùng một giao diện trực quan để dễ dàng sử dụng hệ thống mà không cần quan tâm quá nhiều đến các IoT Platform ở phía dưới.

Chức năng chính của hệ thống:

- Tích hợp việc điều khiển các thiết bị do nhiều IoT Platform vào một giao diện duy nhất.
- Cập nhật dữ liệu của các thiết bị (sensor, device ...) theo thời gian thực.
- Lưu trữ dữ liệu thu thập được từ các IoT Platform.
- Cung cấp các API để người dùng có thể từ đó thiết lập các rules cho mục đích điều khiển tự động.
- Có tính khả mở (có thể tích hợp thêm IoT Platform một cách dễ dàng).

5. Phạm vi của đề án và các công cụ được sử dụng

5.1 Phạm vi của đề án

Đề án được thực hiện với mong muốn có thể tạo ra một nền tảng hỗ trợ điều khiển và tích hợp tất cả các IoT Platform. Tuy nhiên, trong phạm vi giới hạn về thời gian của đề án, hai IoT Platform gồm OpenHab và Home-Assistant được sử dụng như là ví dụ để chứng minh kết quả của đề án..

5.2 Python2.7/Flask

Đề án được chia làm 2 phần Back-end và Front-end. Trong đó phần Back-end được

viết bằng ngôn ngữ Python2.7 và framework Flask.

Python là một ngôn ngữ lập trình đa mục đích được tạo ra vào cuối những năm 1980s, và được đặt tên theo nhóm kịch Monty Python, nó được sử dụng bởi hàng ngàn người để làm những việc từ kiểm thử vi mạch tại hãng Intel, sử dụng trong ứng dụng Instagram, cho tới xây dựng các video game với thư viện PyGame. Nó nhỏ và chặt chẽ như ngôn ngữ tiếng Anh, và có hàng trăm các thư viện của bên thứ ba (third-party).

Lựa chọn Python vì:

- Cú pháp Python rất dễ đọc

Python có điểm chặt chẽ rất giống với ngôn ngữ tiếng Anh, sử dụng những từ như 'not' và 'in' nên khi bạn đọc một chương trình, script, hoặc khi đọc to cho người khác nghe mà không cảm thấy giống như bạn đang nói một thứ ngôn ngữ bí mật nào đó. Điều này cũng được hỗ trợ bởi các quy tắc chấm phẩy câu rất nghiêm ngặt của Python, có nghĩa là lập trình viên không có những dấu ngoặc nhọn ({}) trong code của mình.

- Các thư viện phong phú

Python đã tồn tại khoảng hơn 20 năm, vì vậy có rất nhiều code viết bằng Python được xây dựng qua nhiều thập kỷ, và là một ngôn ngữ mã nguồn mở, rất nhiều trong số này được phát hành cho người khác sử dụng. Hầu như tất cả chúng được tập hợp lại trên trang web <https://pypi.python.org>, bạn phát âm nó là "pie-pee-eye", hoặc còn được gọi bằng một cái tên phổ biến hơn là "the CheeseShop". Lập trình viên có thể cài đặt phần mềm này lên hệ thống của mình để sử dụng bởi các dự án của riêng. Ví dụ, nếu muốn sử dụng Python để xây dựng những script với các đối số dòng lệnh, lập trình viên nên cài đặt thư viện "click" và sau đó import nó vào trong các script của mình rồi sử dụng nó. Có những thư viện sử dụng được cho khá nhiều trường hợp từ thao tác với hình ảnh, cho tới tính toán khoa học, và tự động hóa máy chủ.

- Python có một cộng đồng sử dụng lớn

Python có nhiều nhóm người sử dụng ở khắp mọi nơi, thường được gọi là các PUG, và họ tiến hành những cuộc hội thảo lớn trên tất cả mọi châu lục ngoại trừ Nam Cực. PyCon NA, hội nghị về Python lớn nhất ở Bắc Mỹ, đã bán ra 2.500 vé trong năm nay. Hội nghị này phản ánh cam kết đa dạng hóa của Python, vì có trên 30% diễn giả là phụ nữ. PyCon NA 2013 cũng bắt đầu một xu hướng của việc đưa ra workshop gọi là "Young Coder", nơi mà những người tham dự đã dạy Python cho trẻ em từ 9 đến 16 tuổi trong vòng một ngày, để cho chúng làm quen với ngôn ngữ này và cuối cùng giúp chúng hack và mod một số trò game trên con Raspberry Pi mà chúng được nhận. Việc trở thành một phần của một cộng đồng tích cực như vậy sẽ luôn tạo ra rất nhiều động lực cho những người lập trình.

Khi sử dụng Python, có rất nhiều Python Framework như Django, web2py, CubicWeb...tuy nhiên Flask được lựa chọn để viết back-end vì thấy nó phù hợp với mục đích bài toán mà mình đang làm: đơn giản, dễ học trong thời gian ngắn. Đơn giản, nhưng Flask lại rất hữu dụng vì:

- Bên trong Flask có hỗ trợ ORM, routing đầy đủ.

- Flask hỗ trợ mở rộng các tính năng qua extension.
- Các Extension của Flask rất nhiều, hỗ trợ từ xác nhận, xử lý tải file,....
- Flask và các Extension được cập nhật liên tục.

5.3 Angular2

Angular 2 là một **javascript framework** (nôm na là 1 thư viện javascript được đóng gói và xây dựng nhằm dễ dàng tái sử dụng và xây dựng các ứng dụng có quy mô lớn cần yếu tố tổ chức và quy chuẩn). AngularJS 1.x được khai sinh từ năm 2009 với sự đỡ đầu và phát triển của Google, vì thế mà nó ngày một khẳng định được xu thế một cách nhanh chóng hơn các js framework cùng thời. Với bản chất là mã nguồn mở đúng nghĩa, Angular Js được đông đảo các lập trình viên đón nhận. Và sau một thời gian phát triển bấy lâu, đội ngũ Angular Team đã cho ra mắt Angular 2 kết hợp với TypeScript từ Microsoft để trở nên hoàn thiện hơn về cơ cấu tổ chức ứng dụng, cũng như tốc độ xử lý và hiệu năng khi sử dụng. Chính vì thế, nếu ai đã có kiến thức cơ bản về những mô hình trên, sẽ cảm thấy Angular 2 thật dễ dàng tiếp cận.

Angular 2 thích hợp xây dựng ứng dụng theo mô hình SPA (Single Page Application). Mô hình ứng dụng một trang duy nhất, các phân bố dữ liệu đều được truyền nhận âm thầm với kỹ thuật ajax kết hợp API tương tác với Web API. Chính vì tính tiện dụng này mà Angular 2 thường được ưu tiên lựa chọn cho các mô hình web application chuyên về front-end.

5.4 InfluxDB

InfluxDB là một cơ sở dữ liệu mã nguồn mở lưu trữ theo thời gian. (open source time series database). InfluxDB có một số các tính năng chính sau:

- Được xây tích hợp sẵn HTTP API.
- Có thể gắn thẻ tag cho dữ liệu, cho phép truy cập linh hoạt.
- Ngôn ngữ truy vấn SQL
- Dễ dàng cài đặt và quản lý đồng thời cũng dễ dàng truy cập.
- Được thiết kế để phản hồi các truy vấn thời gian thực, điều đó có nghĩa là mỗi điểm dữ liệu có thể được thêm vào và truy vấn ngược trở lại trong thời gian <100ms.

5.5 Git và GitHub

Git là tên gọi của một **Hệ thống quản lý phiên bản phân tán** (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.

Có rất nhiều lợi thế để bạn nên sử dụng Git:

- Git dễ sử dụng, an toàn và nhanh chóng.
- Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc

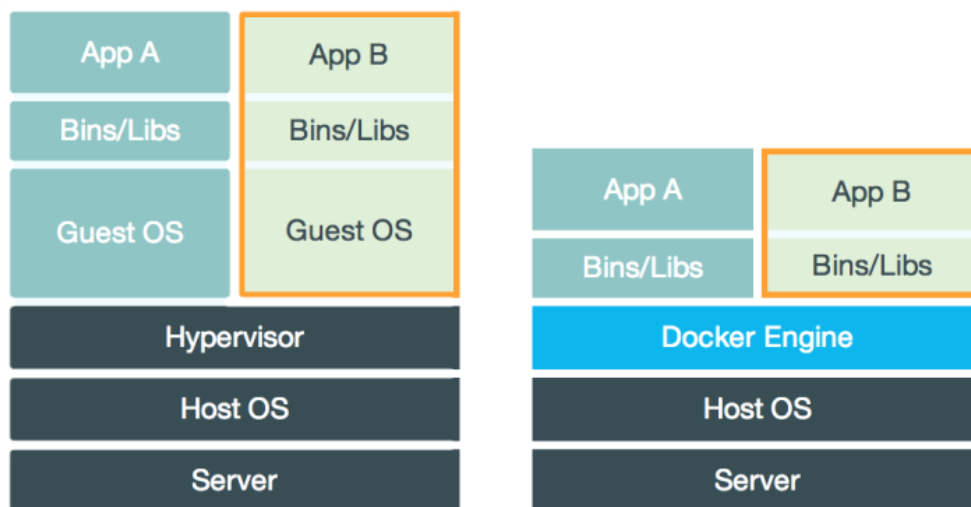
kết hợp các phân nhánh (branch).

- Bạn có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa.
- Dễ dàng trong việc deployment sản phẩm.
- Và nhiều hơn thế nữa.

5.6 Docker

Trong phạm vi đề án, đề án sử dụng docker làm môi trường để vận hành các IoT Platform, giúp các IoT Platform có thể chạy độc lập trên môi trường 1 máy tính, đồng thời hỗ trợ việc testing của hệ thống.

Docker là một Open Platform để xây dựng, vận chuyển và chạy các ứng dụng phân tán (Build-Ship-Run). Ban đầu viết bằng Python, hiện tại đã chuyển sang Go-lang. Docker đưa ra một giải pháp mới cho vấn đề ảo hóa, thay vì tạo ra các máy ảo con chạy độc lập kiểu hypervisors (tạo phần cứng ảo và cài đặt hệ điều hành lên đó), các ứng dụng sẽ được đóng gói lại thành các Container riêng lẻ. Các Container này chạy chung trên nhân hệ điều hành qua LXC (Linux Containers), chia sẻ chung tài nguyên của máy mẹ, do đó, hoạt động nhẹ và nhanh hơn các máy ảo dạng hypervisors.



Hình 12. So sánh docker với virtual machine

Điểm khác biệt chính là các containers sử dụng chung kernel với Host OS nên các thao tác bật, tắt rất nhẹ nhàng, nhanh chóng.

- **Ưu điểm:** nhanh, nhẹ, có thể chia sẻ dễ dàng qua DockerHub.
- **Nhược điểm:** còn mới, cập nhật thay đổi thường xuyên.

6. Tóm tắt chương

IoT đang là tâm điểm chú ý trong nhiều năm trở lại đây, nhiều người đã nhận ra bản chất thực sự của IoT và ý nghĩa của nó tới người tiêu dùng, doanh nghiệp cũng như sự phát triển của nền công nghệ thế giới. Năm bắt xu thế mới, nhiều doanh

ngành đã đầu tư đáng kể vào thị trường IoT, các con chip ngày càng rẻ và được tích hợp nhiều tính năng hơn, các dịch vụ triển khai hệ thống IoT như nhà thông minh, công ty thông minh, thành phố thông minh... ngày càng nhiều, cùng với đó là nhu cầu quản lý các thiết bị IoT cũng tăng theo nên số lượng IoT Platform cũng tăng lên đáng kể.

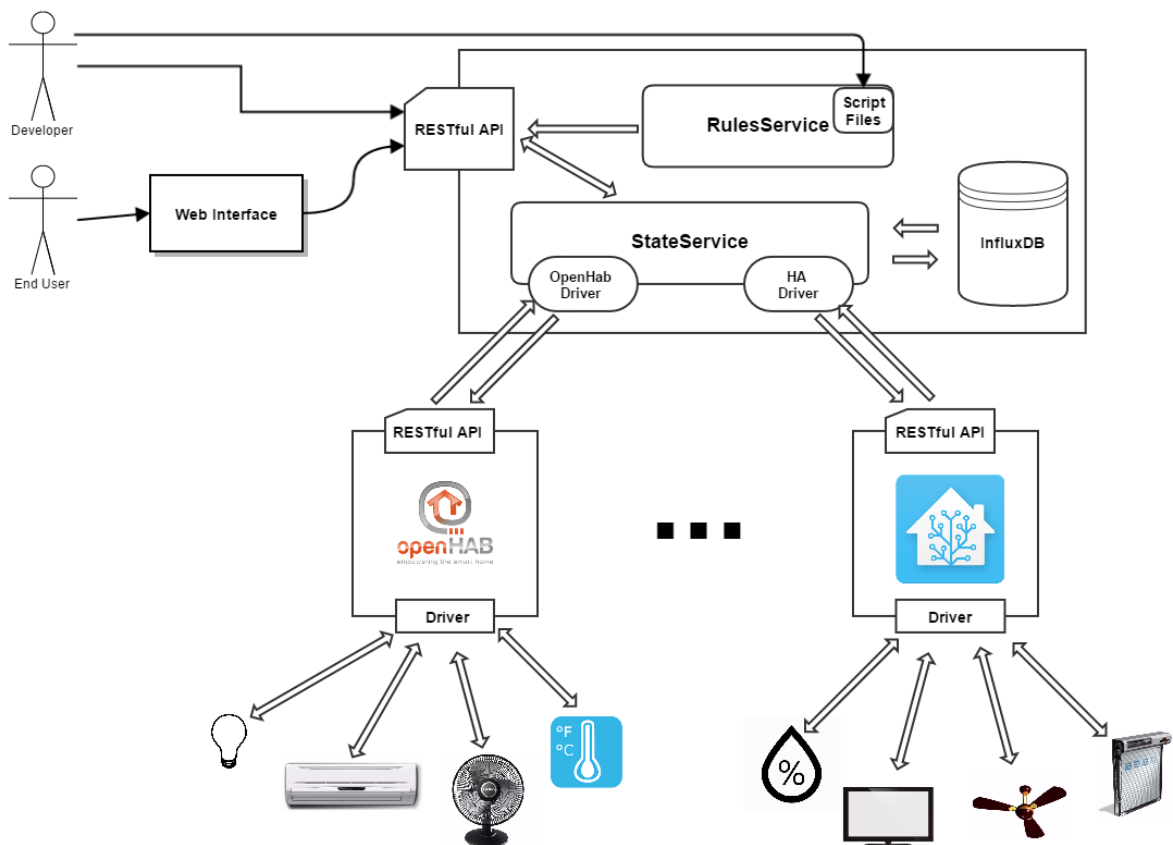
Số lượng IoT Platform tăng lên nhiều nhưng lại tất cả chúng lại không tuân theo một quy chuẩn nhất định. Điều này dẫn đến việc tích hợp các hệ thống IoT Platform riêng biệt lại với nhau gặp rất nhiều khó khăn. Đi theo một hướng tiếp cận để giải quyết vấn đề này, đồ án của tôi mong muốn tạo ra một nền tảng nằm ở phía trên các IoT Platform để có thể điều khiển các thiết bị do các IoT Platform khác nhau quản lý trong một giao diện duy nhất.

Chương II, Các kết quả đạt được

Dựa theo mục tiêu đặt ra ban đầu của đề án là tạo ra một nền tảng có khả năng tích hợp tất cả các IoT Platform lại với nhau. Trong phạm vi đề án, em đã phát triển một nền tảng theo đúng mục đích ban đầu, và lựa chọn 2 IoT Platform là OpenHab và Home-Assistant như 2 nền tảng thử nghiệm để chứng minh nền tảng của em có thể tích hợp các IoT Platform lại với nhau.

1, Kiến trúc hệ thống

Hình dưới đây mô tả mô tả kiến trúc tổng quan của hệ thống:



Hình 13. Kiến trúc hệ thống

Hệ thống được chia thành 4 thành phần chính:

Phần 1: IoT Platforms – Là thành phần quản lý và tương tác trực tiếp với các sensor.

Phần 2: State Service – Tương tác với các API của IoT Platform để thu thập dữ liệu, lưu trữ trong InfluxDB.

Phần 3: Rules Service – Theo dõi các luật (các script files), tương tác với API để chạy các luật này.

Phần 4: RESTful API – Cung cấp cho người dùng một giao diện để có thể tương tác với các sensor.

Phần 5: Web Interface – Giao diện web, cho phép dễ dàng điều khiển các device được kết nối tới các IoT Platform.

Để có thể hiểu rõ hơn về các hoạt động của hệ thống, dưới đây là trình bày các luồng hoạt động cơ bản của hệ thống.

Luồng thu thập dữ liệu trạng thái:

1. Các IoT Platform thu thập dữ liệu từ các thiết bị IoT và cung cấp API cho bên thứ 3.
2. State Service thông qua các driver sẽ thu thập dữ liệu từ các IoT Platform đã được cài đặt và lưu trữ vào trong InfluxDB

Luồng truy vấn dữ liệu trạng thái:

1. Người dùng thông qua API hoặc Web Interface gọi đến các hàm riêng trong StateService. Tùy vào chức năng của mỗi hàm này sẽ gọi đến
2. StateService sẽ gọi đến InfluxDB để truy cập dữ liệu đã thu thập được.

Luồng điều khiển thiết bị:

1. Người dùng thông qua API hoặc Web Interface để gọi đến các hàm riêng trong State Service.
2. State Service sẽ thông qua các IoT Platform Driver để gửi các lệnh điều khiển đến các IoT Platform ở phía dưới.

Luồng chạy hệ thống tự động:

1. Rules Service sẽ theo dõi các file trong thư mục kịch bản
2. Các kịch bản này gọi đến các REST API để điều khiển hệ thống.

1.1 IoT Platform

Tương tác với thiết bị IoT để điều khiển, cập nhật dữ liệu trạng thái. Đồng thời cung cấp các API để bên thứ 3 có thể gián tiếp điều khiển và thu thập dữ liệu.

1.2 State Service

State Service đóng vai trò chủ đạo trong phần backend. State Service được chia làm hai phần chính:

- Các driver của các IoT Platform.
- Các core function để tương tác với IoT Platform và Influxdb.

Các IoT Platform driver trong State Service được định nghĩa theo chuẩn chung, giúp cho State Service có thể lấy được dữ liệu và điều khiển các IoT Platform một cách dễ dàng nhất. Các dữ liệu thu thập được thông qua API đều được lưu trữ lại trong InfluxDB. Các thông tin mà State Service thu thập gồm:

- Tên của thiết bị (name)
- Id của thiết bị (id)
- Trạng thái của thiết bị (state)
- Loại thiết bị (type)

Các thông tin này sẽ được thu thập 1s một lần và được cập nhật liên tục vào trong InfluxDB.

1.3 Rules Service

Rules Service là thành phần có nhiệm vụ theo dõi các file kịch bản được đặt trong

Script folder và tự động chạy các kịch bản này. Các kịch bản này các đoạn mã được định nghĩa bởi người dùng và do đó có thể được chia sẻ và tái sử dụng ở các nơi khác nhau.

Người dùng chỉ cần copy các file kịch bản của mình vào trong thư mục `${ipd.home}/IPD/script`. Trong thư mục này mặc định đã có một vài kịch bản có sẵn do tác giả viết, vì vậy người dùng có thể lấy đó làm cấu trúc tham khảo cho các kịch bản khác.

Ngôn ngữ của kịch bản chính là ngôn ngữ của hệ thống Python. Python là một ngôn ngữ dễ học, nhiều thư viện và có cộng đồng lớn mạnh. Chính vì vậy em đã lựa chọn luôn Python là ngôn ngữ để viết các đoạn script này.

1.4 RESTful API

RESTful API là một thành phần không thể thiếu được trong hệ thống. Nó cung cấp một giao diện giúp cho người dùng có thể tương tác được với các thiết bị được quản lý bởi các IoT Platform khác nhau.

Thông qua RESTful API, chúng ta có thể thực hiện các truy vấn và điều khiển sau:

- Lấy toàn bộ các thông tin của các thiết bị trong hệ thống.
- Lấy được trạng thái của một thiết bị trong hệ thống nếu biết tên/id.
- Lấy lịch sử trạng thái của một thiết bị nếu biết tên/id.
- Thay đổi trạng thái của thiết bị.
- Thay đổi trạng thái của một loại thiết bị.

1.5 Web Interface

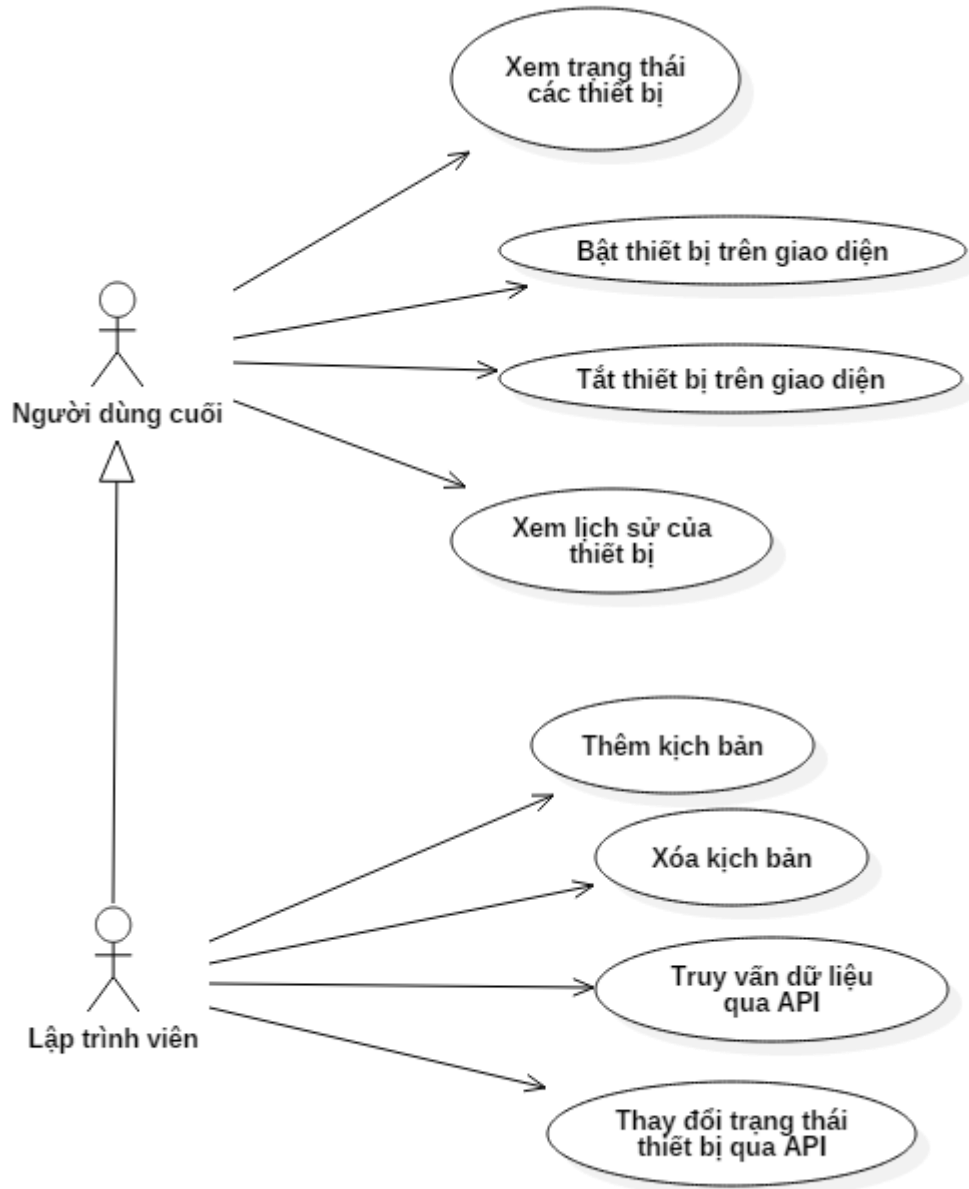
Phiên bản giao diện web được viết trên nền Angular 2, tương thích với nhiều loại màn hình khác nhau. Giao diện cung cấp cho người dùng khả năng khả năng theo dõi trạng thái của các thiết bị một cách real-time, đồng thời cho phép thực hiện điều khiển bật tắt các thiết bị.

Ngoài ra, giao diện còn cho phép người dùng xem lịch sử trạng thái của các thiết bị thông qua các biểu đồ.

2, Phân tích chức năng

2.1 Các chức năng của hệ thống.

a, Biểu đồ UseCase tổng quan



Hình 14. Usecase tổng quan

b, Các tác nhân tham gia hệ thống

STT	Tên tác nhân	Kế thừa	Mô tả
1	Người dùng cuối		Là người dùng hoặc sử dụng phiên bản web để điều khiển các thiết bị do các IoT Platform quản lí.
2	Lập trình viên	Người dùng cuối	Là người phát triển các rules mới cho hệ thống hoặc bên thứ 3 sử dụng API để tích hợp vào các hệ thống khác.

c, Danh sách các usecase

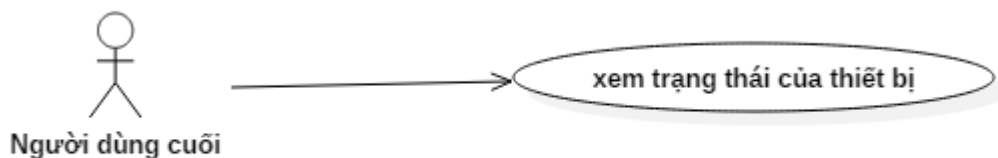
UsecaseID	Tên UseCase	Mô tả	Tác nhân tham gia
UC01	Xem trạng thái của thiết bị	Người dùng vào giao diện web để xem trạng thái của các thiết bị	Người dùng cuối
UC02	Bật thiết bị trên giao diện	Người dùng bật thiết bị trên giao diện	Người dùng cuối
UC03	Tắt thiết bị trên giao diện	Người dùng tắt thiết bị trên giao diện	Người dùng cuối
UC04	Xem biểu đồ lịch sử trạng thái của thiết bị	Người dùng xem lịch sử trạng thái của thiết bị được hiển thị dưới dạng biểu đồ	Người dùng cuối
UC05	Thêm kịch bản	Người dùng add thêm file script vào trong folder mặc định để thêm kịch bản vào	Lập trình viên
UC06	Xóa kịch bản	Người dùng xóa một file script trong folder script để xóa kịch bản đi	Lập trình viên
UC07	Truy vấn dữ liệu qua API	Người dùng thực hiện truy vấn thông tin của các thiết bị qua REST API	Lập trình viên
UC08	Truy vấn toàn bộ thông tin các thiết bị	Người dùng gửi HTTP REQUEST tới REST và nhận về toàn bộ thông tin thiết bị dưới dạng JSON	Lập trình viên
UC09	Truy vấn thông tin một thiết bị	Người dùng gửi HTTP REQUEST tới REST API để nhận được thông tin thiết bị trả về dưới dạng JSON	Lập trình viên
UC10	Truy vấn lịch sử trạng thái của thiết bị	Người dùng gửi HTTP REQUEST tới REST API để nhận được lịch sử của thiết bị trả về dưới dạng JSON	Lập trình viên
UC11	Thay đổi trạng thái qua API	Người dùng gửi HTTP REQUEST tới REST API để thay đổi trạng thái của thiết bị.	Lập trình viên

2.2 Danh sách các REST API do hệ thống cung cấp

STT	Phương thức	API	Mô tả
1	GET	/api/states	Cho phép người sử dụng truy vấn toàn bộ các thông tin trạng thái của các thiết bị do các IoT Platform quản lý
2	GET	/api/states/<name>	Cho phép người sử dụng truy vấn thông tin trạng thái của thiết bị có tên là <<name>>
3	GET	/api/states/history/<name>	Cho phép người sử dụng truy vấn lịch sử trạng thái của thiết bị có tên là <<name>>
4	POST	/api/states/<name> JSON: {state:ON/OFF}	Cho phép người sử dụng thay đổi trạng thái của thiết bị có tên là name
5	POST	/api/states/type/<type> JSON: {state:ON/OFF}	Cho phép người sử dụng thay đổi trạng thái của tất cả các thiết bị thuộc kiểu <<type>>

2.3 Mô tả các ca sử dụng

a, UC01



Hình 15. Usecase xem trạng thái thiết bị

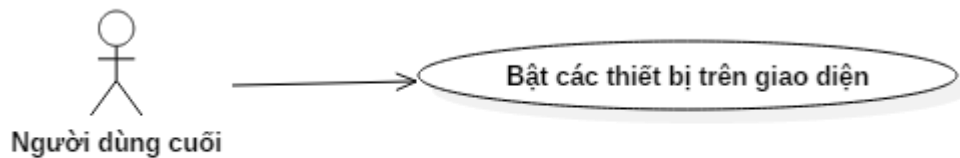
Tác nhân: Người dùng cuối

Mô tả chung: Người dùng truy cập vào trang chủ của hệ thống

Mã Use Case:	UC01		
Tên Use Case:	Xem trạng thái của thiết bị		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Người dùng cuối		
Tác nhân:	Người dùng sử dụng hệ thống trên giao diện web		
Trigger:	1. Người dùng vào trang homepage của hệ thống theo đường link		
Điều kiện tiên quyết:			

Điều kiện sau:	1. Nếu thành công hệ thống sẽ trả về giao diện gồm danh sách các thiết bị được quản lý và trạng thái của các thiết bị này.	
Thao tác chính:	Người Dùng	Hệ Thống
	1. Người dùng vào trang homepage	1. Kiểm tra đường link và trả về giao diện trang chủ
Thao tác khác:	Người Dùng	Hệ Thống
	1. Nhập sai link	1. Trả về trang 404, đường link vừa nhập không tồn tại
Ngoại lệ:		
Yêu cầu Usecase:		
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định	

b, UC02



Hình 16. Usecase bật thiết bị trên giao diện

Mô tả tổng quan: Người dùng bấm vào công tắc trên giao diện điều khiển để bật thiết bị

Mã Use Case:	UC02		
Tên Use Case:	Bật các thiết bị trên giao diện		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Người dùng cuối		
Tác nhân:	Người dùng bật thiết bị từ giao diện		
Trigger:	1. Người dùng bấm vào công tắc của thiết bị đang tắt để bật thiết bị		
Điều kiện tiên quyết:	1, Người dùng đang ở trang chủ		
Điều kiện sau:	1. Nếu thành công thiết bị sẽ được bật lên, nếu không thành công thì trạng thái của thiết bị sẽ trở về OFF		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Bấm vào công tắc ở thiết bị đang tắt	1. Thực hiện bật thiết bị, nếu thành công thì thiết bị sẽ bật lên, không thành công thì trạng thái của thiết bị vẫn giữ nguyên.	
Thao tác khác:	Người Dùng	Hệ Thống	
Ngoại lệ :			
Yêu cầu Usecase:	Usecase : Xem trạng thái các thiết bị		

Yêu cầu đặc biệt:	<ul style="list-style-type: none"> - Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định
-------------------	--

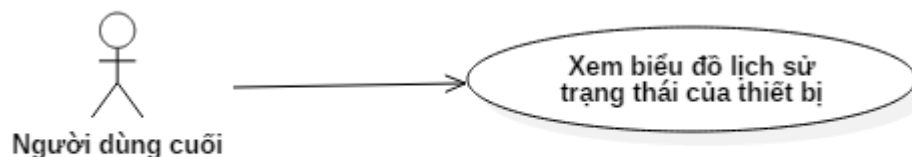
c, UC03



Hình 17. Uscase tắt thiết bị trên giao diện

Mã Use Case:	UC03		
Tên Use Case:			
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Người dùng cuối		
Tác nhân:	Tất thiết bị trên giao diện		
Trigger:	1. Người dùng bấm vào công tắc của thiết bị đang bật để tắt thiết bị		
Điều kiện tiên quyết:	1, Người dùng đang ở trang chủ		
Điều kiện sau:	1. Nếu thành công thì thiết bị sẽ được tắt đi, nếu không thành công thì thiết bị vẫn ở trạng thái ON		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Bấm vào công tắc ở thiết bị đang bật	1. Thực hiện bật thiết bị và trả về kết quả của hành động.	
Thao tác khác:	Người Dùng	Hệ Thống	
Ngoại lệ:			
Yêu cầu Uscase:	Uscase : Xem trạng thái các thiết bị		
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		

d, UC04



Hình 18. Uscase xem biểu đồ lịch sử của thiết bị

Mã Use Case:	UC04
Tên Use Case:	Xem biểu đồ lịch sử trạng thái của thiết bị

Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Người dùng cuối		
Tác nhân:	Người dùng xem biểu đồ thể hiện lịch sử trạng thái của thiết bị theo thời gian		
Trigger:	1. Người dùng click vào tên của thiết bị để vào trang xem biểu đồ trạng thái của thiết bị đó		
Điều kiện tiên quyết:	1, Đang ở trang chủ		
Điều kiện sau:	1.		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Người dùng click vào tên của thiết bị	1. Hệ thống sẽ chuyển đến trang hiển thị biểu đồ trạng thái của thiết bị.	
Thao tác khác:	Người Dùng	Hệ Thống	
Ngoại lệ:			
Yêu cầu Usecase:	Usecase : Xem trạng thái các thiết bị		
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		

e, UC05



Hình 19. Usecase thêm kịch bản

Mã Use Case:	UC05		
Tên Use Case:	Thêm kịch bản		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Người dùng add thêm file script vào trong folder mặc định để thêm kịch bản vào hệ thống		
Trigger:	1. Copy file kịch bản vào trong folder script		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1. Hệ thống không báo lỗi		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Copy file kịch bản vào thư mục định sẵn	1. Thực hiện cập nhật và chạy kịch bản	
Thao tác khác:	Người Dùng	Hệ Thống	
Ngoại lệ:			

Yêu cầu Usecase:	Usecase : Xem trạng thái các thiết bị
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định

f, UC06



Hình 20. Usecase xóa kịch bản

Mã Use Case:	UC06		
Tên Use Case:	Xóa kịch bản		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Người dùng xóa một file script trong folder script để xóa kịch bản đi		
Trigger:	1. Xóa file kịch bản vào trong thư mục kịch bản		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1. Hệ thống không báo lỗi		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Xóa file kịch bản trong thư mục định sẵn	1. Thực hiện cập nhật và chạy kịch bản còn lại	
Thao tác khác:	Người Dùng	Hệ Thống	
Ngoại lệ:			
Yêu cầu Usecase:	Usecase : Xem trạng thái các thiết bị		
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		



Hình 21. Usecase truy vấn thông tin các thiết bị qua API

Mã Use Case:	UC08		
Tên Use Case:	Truy vấn toàn bộ thông tin các thiết bị		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Truy vấn toàn bộ thông tin các thiết bị qua REST API		
Trigger:	1. Gửi HTTP REQUEST đến REST API để truy vấn toàn bộ thông tin thiết bị		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1.		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Sử dụng các công cụ có thể gửi HTTP Request gửi 1 gói tin tới URL: http://{ \$host }:{ \$port }/api/states METHOD: GET HEADER: Appication/JSON	1. Kiểm tra HEADER và gửi trả về kết quả về thông tin của các sensor dưới dạng JSON	
Thao tác khác:	Người Dùng	Hệ Thống	
	1.Người dùng gửi sai địa chỉ URL hoặc sai HEADER	1. Hệ thống không chấp nhận request	
Ngoại lệ:			
Yêu cầu Usecase:			
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		



Hình 22. Usecase truy vấn thông tin một thiết bị

Mã Use Case:	UC09		
Tên Use Case:	Truy vấn thông tin một thiết bị		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Truy vấn thông tin một thiết bị qua REST API		
Trigger:	1. Gửi HTTP REQUEST đến REST API để truy vấn thông tin một thiết bị		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1.		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Sử dụng các công cụ có thể gửi HTTP Request gửi 1 gói tin tới URL: <a href="http://{ \$host }:{ \$port }/api/states/<name>">http://{ \$host }:{ \$port }/api/states/<name> METHOD: GET HEADER: Appication/JSON	1.Gửi trả về thông tin của thiết bị dưới dạng JSON	
Thao tác khác:	Người Dùng	Hệ Thống	
	1.Người dùng gửi sai địa chỉ URL hoặc sai HEADER 2.Người dùng gửi sai tên của thiết bị	1. Hệ thống không chấp nhận request 2. Hệ thống trả về thông báo không tồn tại thiết bị trong hệ thống	
Ngoại lệ:			
Yêu cầu Usecase:			
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		

i, UC10



Hình 23. Usecase truy vấn lịch sử của thiết bị

Mã Use Case:	UC10		
Tên Use Case:	Truy vấn lịch sử trạng thái của thiết bị		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Truy vấn lịch sử trạng thái của thiết bị qua REST API		
Trigger:	1. Gửi HTTP REQUEST đến REST API để truy vấn lịch sử trạng thái của thiết bị		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1.		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Sử dụng các công cụ có thể gửi HTTP Request gửi 1 gói tin tới URL: <a href="http://{Shost}:{Sport}/api/states/history/<name>">http://{Shost}:{Sport}/api/states/history/<name> METHOD: GET HEADER: Appication/JSON	1.Gửi trả về lịch sử của thiết bị dưới dạng JSON	
Thao tác khác:	Người Dùng	Hệ Thống	
	1.Người dùng gửi sai địa chỉ URL hoặc sai HEADER 2.Người dùng gửi sai tên của thiết bị	1. Hệ thống không chấp nhận request 2. Hệ thống trả về thông báo không tồn tại thiết bị trong hệ thống	
Ngoại lệ:			
Yêu cầu Usecase:			
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		

j, UC11

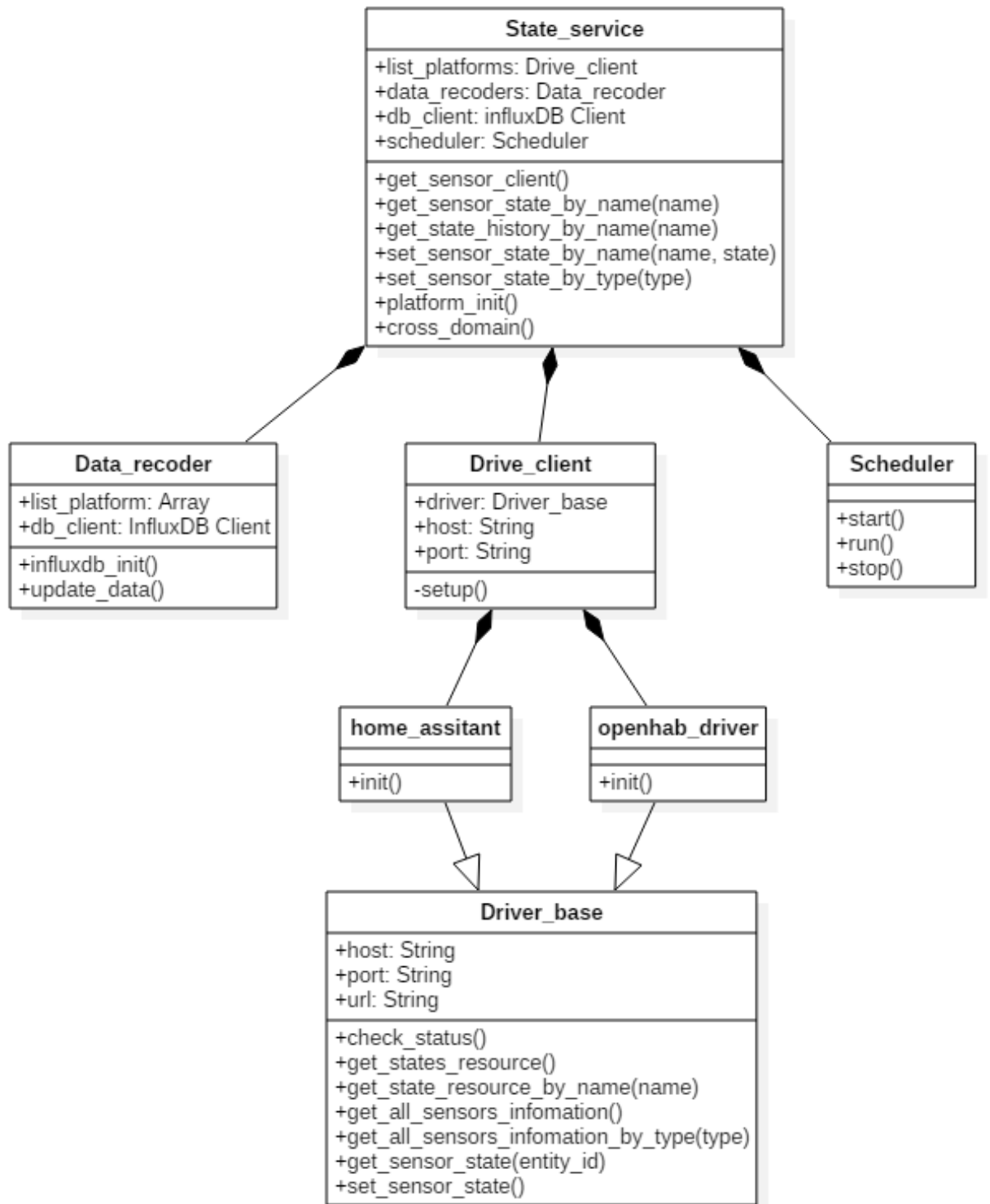


Hình 24. Usecase thay đổi trạng thái qua API

Mã Use Case:	UC11		
Tên Use Case:	Thay đổi trạng thái qua API		
Người tạo:	Bùi Ngọc Luân	Last Updated By:	Bùi Ngọc Luân
Date Created:	10/12/2016	Date Last Updated:	10/12/2016
Tác nhân:	Lập trình viên		
Tác nhân:	Thay đổi trạng thái của thiết bị thông qua API		
Trigger:	1.		
Điều kiện tiên quyết:	1,		
Điều kiện sau:	1.		
Thao tác chính:	Người Dùng	Hệ Thống	
	1. Sử dụng các công cụ có thể gửi HTTP Request gửi 1 gói tin tới URL: <a href="http://{ \$host }:{ \$port }/api/states/history/<name>">http://{ \$host }:{ \$port }/api/states/history/<name> METHOD: POST HEADER: Appication/JSON	1. Thay đổi trạng thái của thiết bị.	
Thao tác khác:	Người Dùng	Hệ Thống	
	1.Người dùng gửi sai địa chỉ URL hoặc sai HEADER 2.Người dùng gửi sai tên của thiết bị	1. Hệ thống không chấp nhận request 2. Hệ thống trả về thông báo không tồn tại thiết bị trong hệ thống	
Ngoại lệ:			
Yêu cầu Usecase:	Usecase : Xem trạng thái các thiết bị		
Yêu cầu đặc biệt:	- Hệ thống máy chủ đang chạy ổn định - Các IoT Platform và các service chạy ổn định		

3. Thiết kế lớp

Biểu đồ lớp tổng quát của phần back-end:



Hình 25. Biểu đồ lớp của hệ thống

3.1 Chi tiết các lớp

a. Lớp *Driver_base*

Đây là lớp tiêu chuẩn cho các IoT Platform Driver, định nghĩa ra các phương thức chung cho tất cả các Driver. Dưới đây là đặc tả chi tiết các **thuộc tính và phương thức**

Thuộc tính	Giải thích
host	Địa chỉ IP của Platform
port	Cổng của Platform
url	Địa chỉ gồm host và port

Phương thức	Giải thích
check_status()	Kiểm tra xem hệ thống có đang chạy bình thường hay không
get_states_resource()	Gọi đến API của IoT Platform để lấy thông tin của các sensor dưới dạng XML, JSON....
get_states_resource_by_name(name)	Gọi đến API của IoT Platform để lấy thông tin của 1 sensor chỉ định, dữ liệu trả về dạng XML, JSON.....
get_all_sensors_infomation()	Xử lý dữ liệu đã lấy được ở get_states_resource() và trả về dưới dạng mảng 2 chiều [[id, state, type].....]
get_all_sensors_infomation_by_type(type)	Xử lý dữ liệu đã lấy được ở get_states_resource(), lọc theo kiểu sensor và trả về dưới dạng mảng 2 chiều [[id, state, type].....]
get_sensor_state(entity_id)	Trả về trạng thái của thiết bị có id trùng khớp
Set_sensor_state(entity_id, state)	Thiết lập trạng thái cho thiết bị có entity_id trùng khớp về “state”

b, Hai lớp *home-assistant* và *openhab*

Hai lớp này kế thừa toàn bộ các thuộc tính và phương thức của lớp *Driver_base*, và @override lại toàn bộ các phương thức được kế thừa để thích ứng với từng IoT Platform

c, *Scheduler*

Lớp này cung cấp chức năng hẹn giờ cho một hàm, các đối tượng thuộc lớp này sẽ nhận đầu vào là một hàm và thời gian quay vòng để thực hiện hàm đó. Mục đích chính của Scheduler là để gọi đến phương thức thu thập dữ liệu và cập nhật vào trong csdl

Đặc tả thuộc tính:

Thuộc tính	Đặc tả
sleep_time	Biến thời gian được truyền vào, sau một khoảng sleep_time nhất định sẽ thực hiện gọi đến phương thức funtion()

Đặc tả các phương thức:

Phương thức	Đặc tả
start()	Khởi chạy lần đầu
run()	Sau khi bị gọi thì sẽ chạy theo cài đặt sẵn
stop()	Dừng vòng lặp

d, Drive_client

Một đối tượng drive_client sẽ đại diện cho một IoT Platform được cài đặt ở phía dưới. Trong mỗi Drive_client sẽ có chứa driver của IoT Platform tương ứng (Vì một driver có thể được dùng cho nhiều client, nên cần tách Drive_client riêng ra).

Đặc tả về thuộc tính:

Thuộc tính	Đặc tả
driver	Đây là driver tương ứng được truyền vào từ constructor()
host	Địa chỉ IP của IoT Platform
port	Địa chỉ cổng của IoT Platform

e, Data_recoder

Đối tượng Data_recoder có 2 chức năng chính:

- Gọi đến các hàm ở danh sách các Drive_client để truy vấn dữ liệu
- Cài đặt và kết nối tới Influxdb, thêm dữ liệu truy vấn được ở phía trên vào InfluxDB

Đặc tả Thuộc tính

Thuộc tính	Đặc tả
list_platform	Danh sách các Drive_client dưới dạng mảng
db_client	InfluxDB Client, là đối tượng tạo từ driver cho InfluxDB cung cấp

Đặc tả phương thức:

Phương thức	Đặc tả
influxdb_init()	Cấu hình và cài đặt cho InfluxDB Client
update_data()	Cập nhật dữ liệu vào InfluxDB Client

f, State_service

Đây được coi như hàm main của hệ thống, cung cấp các hàm có thể gọi thông qua API, đồng thời cũng là nơi khởi chạy server

Đặc tả thuộc tính

Thuộc tính	Đặc tả
list_platforms	Danh sách các Drive_client, dùng để gọi đến các platform.
data_recoder	Khởi tạo data_recoder để có thể gọi đến hàm update

db_client	InfluxDB sau khi đã config.
scheduler	Đối tượng đại diện cho Scheduler

Đặc tả phương thức

Phương thức	Đặc tả
get_sensor_states()	Lấy toàn bộ dữ liệu sensor ở tất cả các IoT Platform và trả về dưới dạng JSON.
get_sensor_state_by_name(name)	Lấy dữ liệu sensor theo tên và trả về dưới dạng JSON
get_state_history_by_name(name)	Lấy dữ liệu lịch sử của sensor theo tên và trả về dưới dạng JSON
set_sensor_state_by_name(name, state)	Gửi lệnh đến các IoT Platform để thay đổi trạng thái của sensor đã định danh.,
set_sensor_state_by_type(type, state)	Gửi lệnh đến các IoT Platform để thay đổi trạng thái của một kiểu sensor (tắt bóng đèn chẳng hạn).
platform_init()	Khởi tạo danh sách các IoT Platform mà service quản lý
cross_domain()	Dùng để chấp nhận các header có quyền truy cập.

3.2 Tính đa hình trong thiết kế lớp

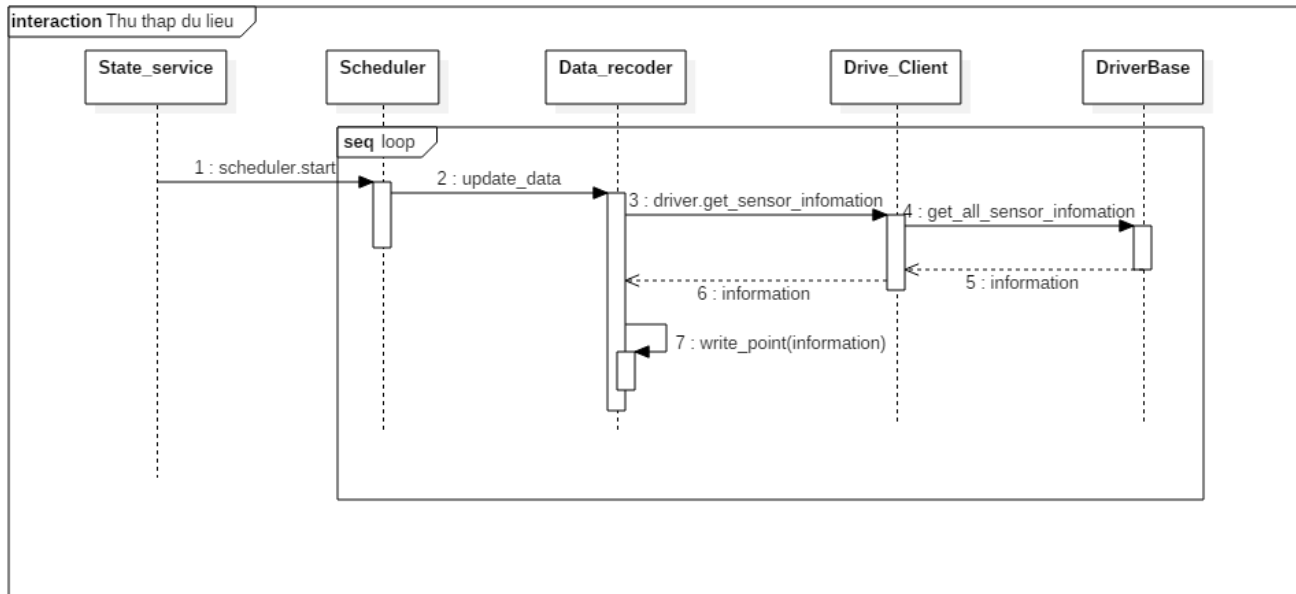
Tính đa hình cho phép các chức năng (method) khác nhau được thực thi khác nhau trên các đối tượng khác nhau. Nói nôm na đơn giản hơn là bạn có 3 lớp A,B,C kế thừa nhau và trong 3 lớp này có ba method cùng tên là show(); . Khi ta tạo mới đối tượng cho các lớp trên thì những đối tượng này gọi tới method show() nằm ở lớp nào thì thực thi chức năng phương thức ở lớp đó. Bởi vì chúng kế thừa nên sử dụng rõ tính đa hình này.

Ứng dụng điều đó trong thiết kế lớp của hệ thống, lớp Driver_base định nghĩa ra những phương thức chung nhất trong hệ thống. Tất cả các driver của các IoT Platform đều phải kế thừa từ lớp Driver_base này. Khi này các driver sẽ có các phương thức có tên giống nhau, tuy nhiên các phương thức này sẽ xử lý hoàn toàn khác nhau để phù hợp với từng Platform khác nhau.

4, Phân tích hành vi người dùng

Phần này sẽ đi sâu vào việc cách tương tác giữa các đối tượng trong hệ thống với các kịch bản cụ thể dựa trên biểu đồ tuần tự.

4.1 Thu thập dữ liệu

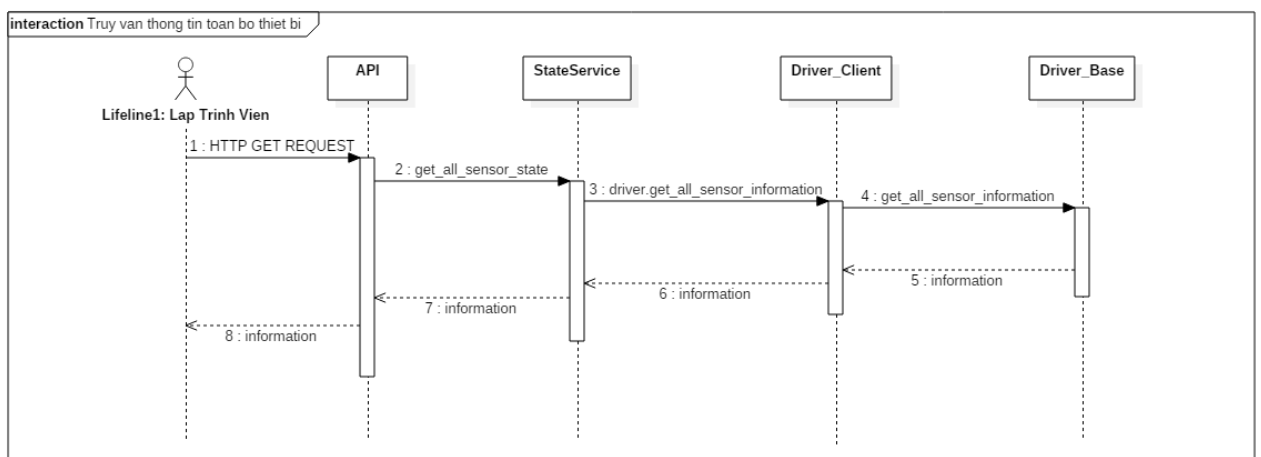


Hình 26. Sequence Diagram thu thập dữ liệu

Mô tả:

Khi hệ thống bắt đầu khởi động thì vòng lặp để bắt đầu hàm thu thập dữ liệu cũng bắt đầu theo. Khởi đầu một vòng lặp, **Scheduler** sẽ gọi đến hàm **update_data** trong của đối tượng **Data_recoder**. Đối tượng này sẽ duyệt qua vòng lặp của các **Driver_Client** và thông qua các **Driver_client** này gọi đến các hàm thu thập dữ liệu nằm trong **Driver** của từng **IoT Platform**. Dữ liệu thu thập sẽ được trả về **Data_recoder**, khi này **Data_recoder** sẽ làm nhiệm vụ phân loại và ghi dữ liệu (dữ liệu lấy từ **IoT Platform** nào sẽ được phân loại theo cách đây) vào trong **InfluxDB**.

4.2 Truy vấn tất cả thông tin của thiết bị

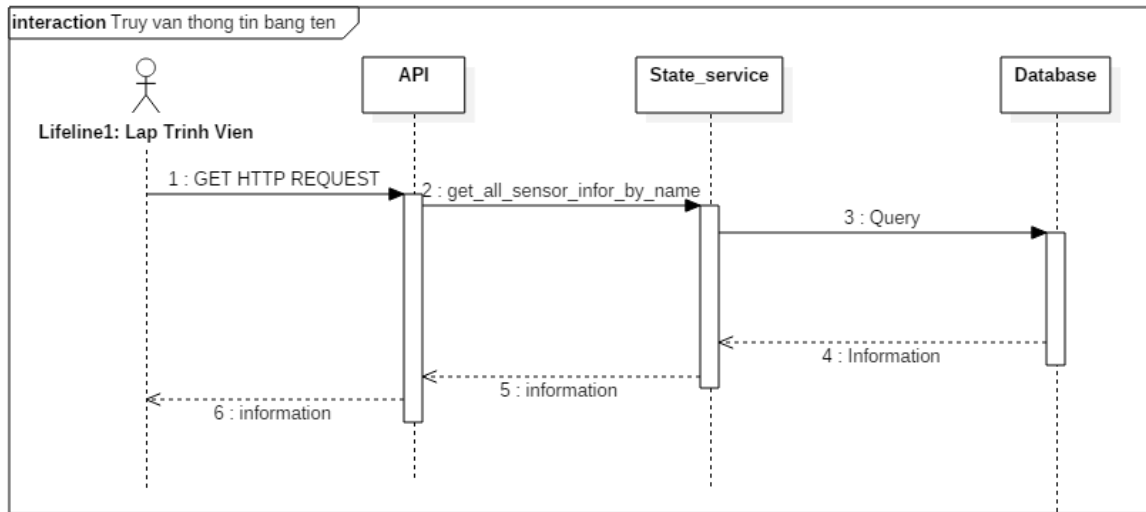


Hình 27. Sequence Diagram truy vấn thông tin thiết bị

Đầu tiên lập trình viên sẽ gửi **Request** đến **API**, sau đó framework sẽ tự động điều hướng gọi tới phương thức **get_all_sensor_method** ở **StateService**. Lúc này , **StateService** sẽ dùng vòng lặp để sử dụng từng **Driver_Client** mình đang quản lí

gọi đến các **Driver** để lấy dữ liệu. Dữ liệu chuyển tổng hợp về **StateService** sẽ được đóng lại dưới dạng **JSON** và trả về cho người dùng.

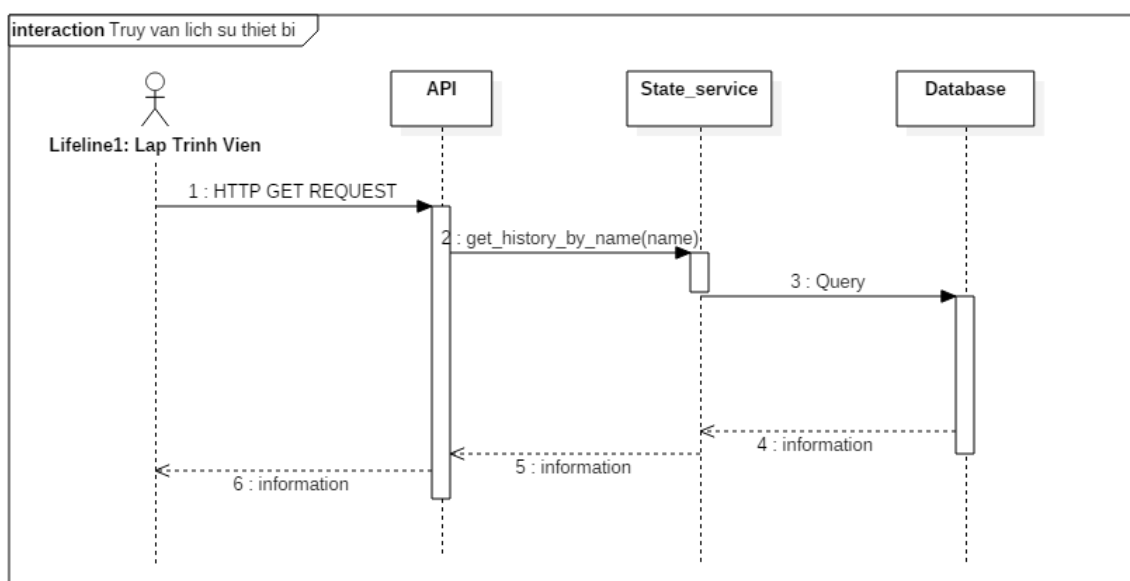
4.3 Truy vấn thông tin của một thiết bị



Hình 28. Sequence Diagram Truy vấn thông tin một thiết bị

Đầu tiên người dùng sẽ gửi **REQUEST** đến **API** kèm theo biến **name** (biến để lấy sensor), sau đó framework sẽ tự động điều hướng gọi tới phương thức **get_all_sensor_infor_by_name()**, trong phương thức này **State_service** sẽ sử dụng ngôn câu lệnh SQL để truy vấn dữ liệu từ **InfluxDB**. Dữ liệu sẽ được trả về cho **State_service**, sau đó được định dạng lại dưới dạng JSON và gửi kết quả này cho người truy vấn.

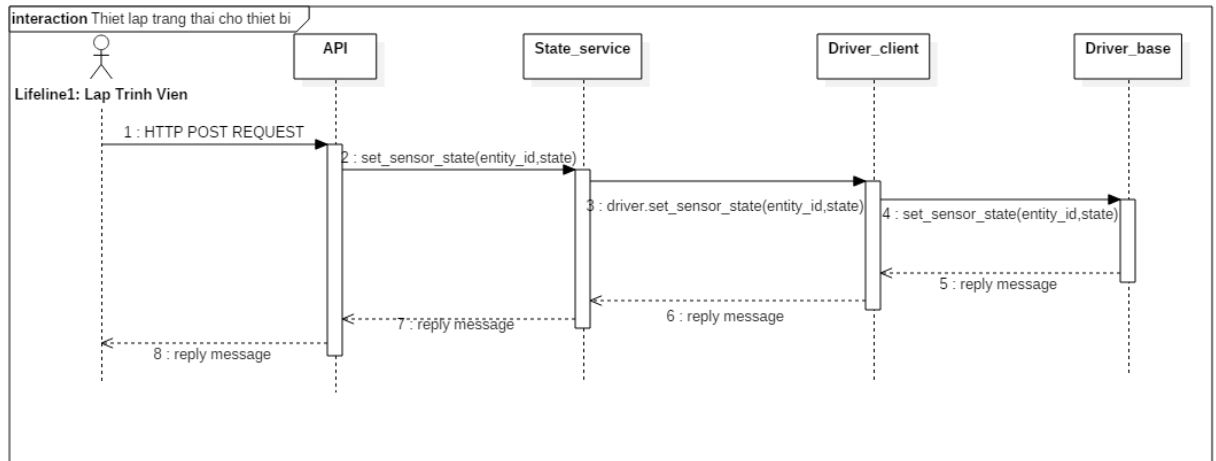
4.4 Truy vấn lịch sử trạng thái của thiết bị



Hình 29. Sequence Diagram Truy vấn lịch sử trạng thái của thiết bị

Lập trình viên gửi **HTTP REQUEST** đến **API** kèm theo biến name đại diện cho định danh của thiết bị. **Framework** sẽ dùng định hướng từ **API** đến hàm **get_history_by_name(name)** của **State_service**, hàm này sử dụng câu lệnh SQL để truy vấn lịch sử dữ liệu trong **InfluxDB**, dữ liệu trả về dữ liệu lịch sử dưới dạng mảng, khi **State_service** nhận được dữ liệu trả về sẽ đóng gói **JSON** và trả về cho người dùng.

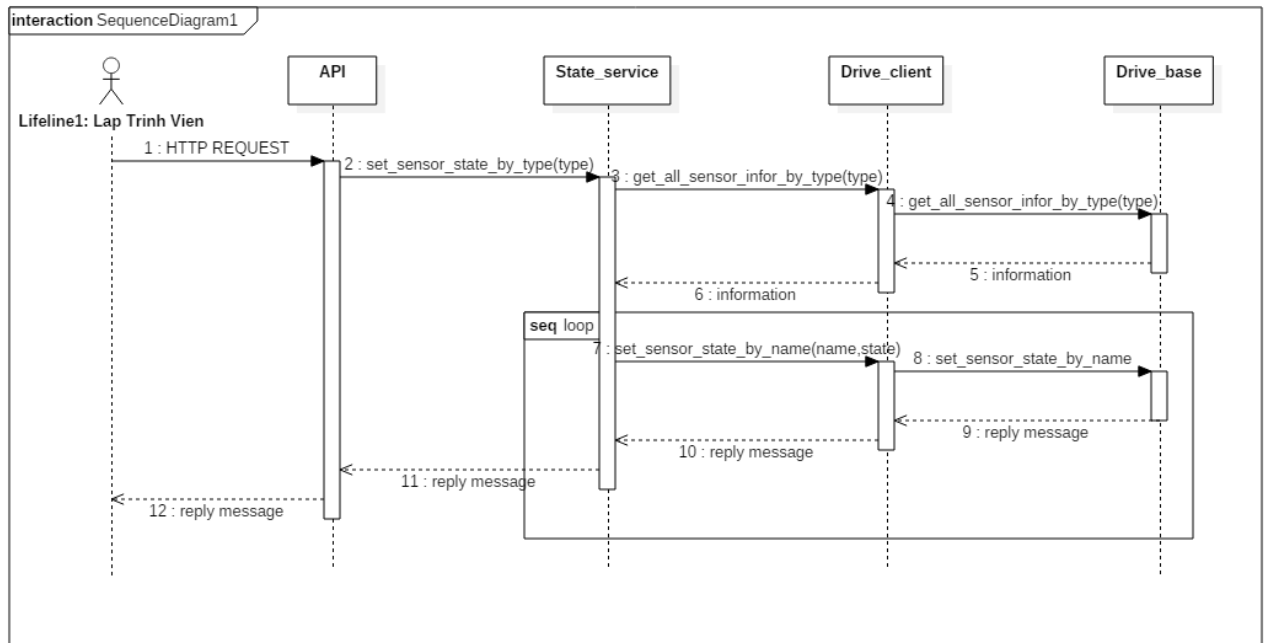
4.5 Thiết lập trạng thái cho thiết bị



Hình 30. Sequence Diagram Thiết lập trạng thái cho thiết bị

Người dùng gửi **HTTP REQUEST** tới **API**, **framework** sẽ tự động điều hướng tới hàm **set_sensor_state(entity_id, state)** ở **State_service** trong đó **entity_id** là định danh của của thiết bị (giống với tên) và **state** là trạng thái muốn cài đặt cho thiết bị. Khi này **State_service** sẽ truy vấn đến cơ sở dữ liệu để xác định xem thiết bị này do **IoT Platform** nào quản lý. Sau khi xác định được thiết bị này thuộc **Platform** nào quản lý, **State_service** thông qua các **Driver_Client** sẽ gọi đến hàm **set_sensor_state(entity_id,state)** ở **Driver_base** để cài đặt trạng thái cho thiết bị. Sau khi có kết quả của việc cài đặt (thành công hoặc thất bại), kết quả sẽ được gửi từ Driver đến **Driver_Client**, **State_service**, rồi nén lại dưới định dạng **JSON** và gửi lại cho người dùng.

4.6 Thiết lập trạng thái cho thiết bị theo kiểu thiết bị



Hình 31. Sequence Diagram thiết lập trạng thái cho thiết bị theo kiểu thiết bị

Lập trình viên sẽ gửi **HTTP REQUEST** đến **API**, **framework** sẽ điều hướng gọi tới hàm **set_sensor_state_by_type(type)** trong đó **type** là kiểu thiết bị cần thay đổi trạng thái. Trước tiên để **State_service** sẽ phải thông qua **Driver_client** để gọi tới **Driver_base** để truy vấn danh sách các thiết bị thuộc kiểu type. Sau khi lấy được danh sách các tên các thiết bị thuộc kiểu type rồi, thì **State_service** sẽ chạy vòng lặp để thiết lập trạng thái cho từng thiết bị này. Công việc trong mỗi vòng lặp tương tự như hành động ở **sequence diagram** đã nói ở trên.

5, Thiết kế cơ sở dữ liệu

Với mong muốn thiết kế một cơ sở dữ liệu có thể cập nhật dữ liệu theo thời gian thực, đồng thời có thể dễ dàng truy vấn và thêm dữ liệu, Hệ thống đã lựa chọn InfluxDB là cơ sở dữ liệu của mình.

InfluxDB là cơ sở dữ liệu có định dạng tương đối khác với cơ sở dữ liệu truyền thống, chính vì vậy đơn vị lưu trữ của influxDB như sau:

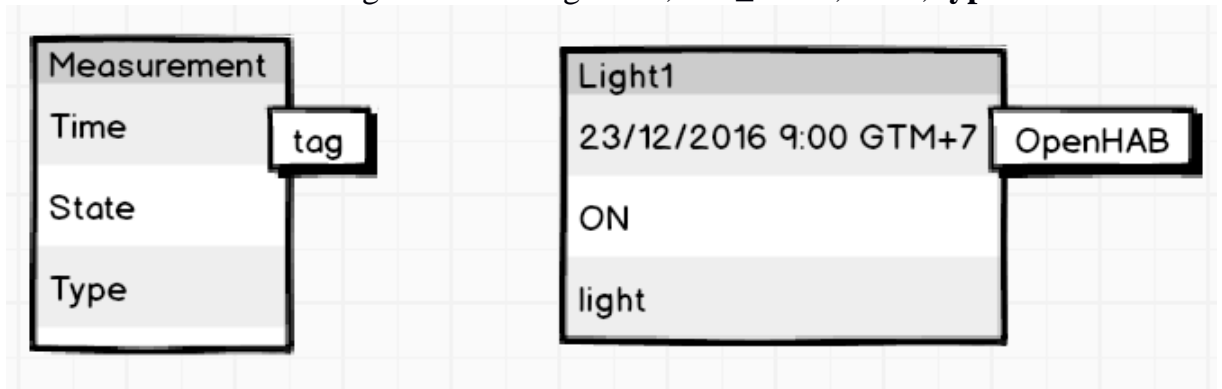
- Có nhiều cơ sở dữ liệu, trong mỗi cơ sở dữ liệu có thể chứa nhiều measurement
- Measurement: Lưu trữ một tập các dữ liệu, các dữ liệu sẽ được thêm vào measurement theo các cột đã được định danh trước.
- Point: mỗi point là một điểm dữ liệu trong measurement
- Tag: dùng để đánh tag, nhằm phân biệt các dữ liệu với nhau.

Khi thiết kế cơ sở dữ liệu cho hệ thống, việc quan trọng nhất là lọc ra được những sự tương đồng trong dữ liệu trả về của các IoT Platform khác nhau. Trong thực nghiệm tích hợp 2 IoT Platform là OpenHAB và Home-Assistant, cơ sở dữ liệu được thiết kế theo nguyên tắc sau

- Dữ liệu của mỗi thiết bị sẽ được lưu trữ tại một **measurement**.
- Các **measurement** này sẽ được đánh **tag** tương ứng với IoT Platform trực

tiếp quản lí thiết bị này.

- Mỗi measurement gồm các trường: **time**, **soft_name**, **state**, **type**



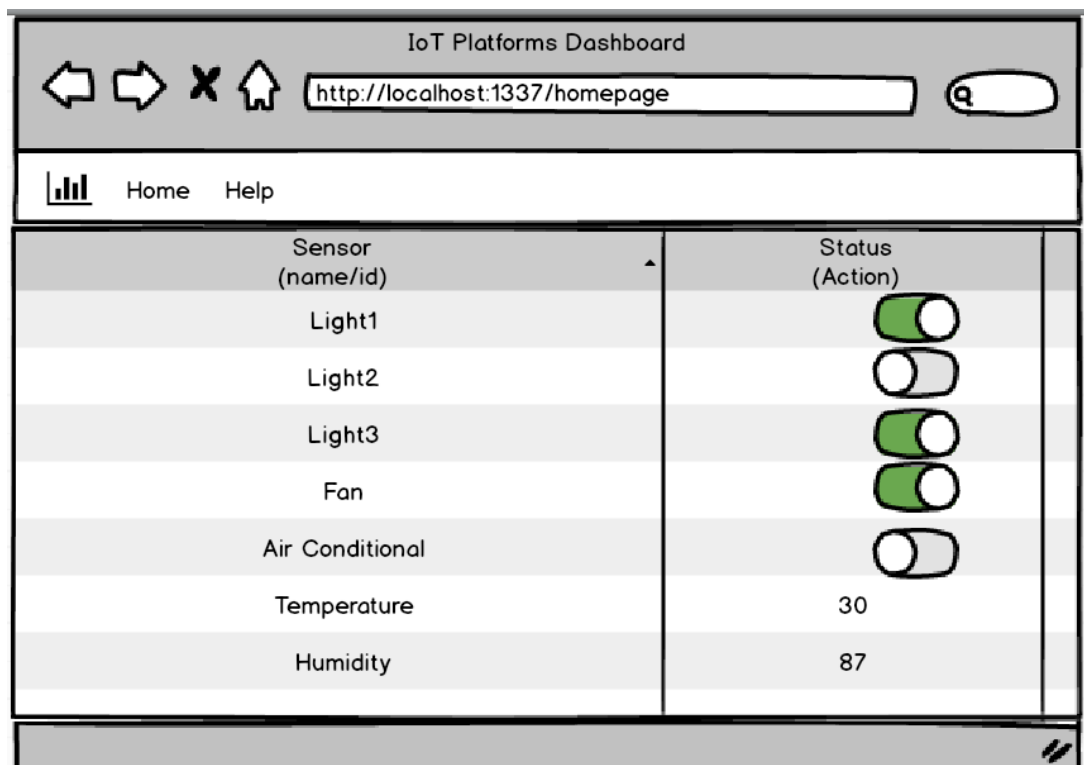
Hình 31. Thiết kế cơ sở dữ liệu

Ở trên là cấu trúc lưu trữ của **InfluxDB** và bên phải là ví dụ về 1 bản ghi trong **Measurement Light1**

6, Thiết kế giao diện đồ họa

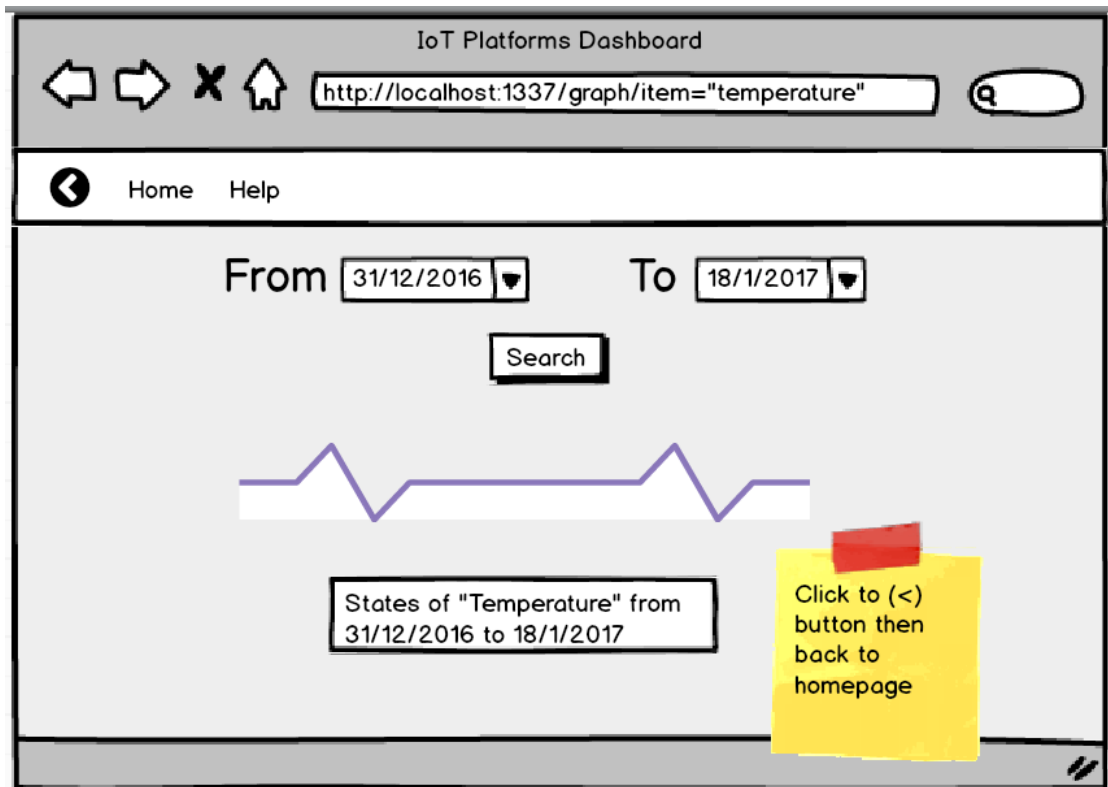
Giao diện của hệ thống sẽ gồm 2 phần chính là phần điều khiển và phần xem lịch sử trạng thái của thiết bị.

Giao diện điều khiển của thiết bị:



Hình 32. Thiết kế giao diện điều khiển

Giao diện xem lịch sử của thiết bị



Hình 33. Thiết kế giao diện xem lịch sử của thiết bị

7, Tính khả mở của hệ thống

Với mục tiêu ban đầu là thiết kế một hệ thống có tính khả mở cao, đồ án đã được thiết kế để có thể dễ dàng tích hợp thêm các IoT Platform khác và mẫu chốt chính là Driver_base.

Tính khả mở theo chiều dọc:

Giả thiết đặt ra rằng nếu OpenHab và Home-Assistant cùng phát triển thêm một tính năng mới. Thì tính năng mới đó sẽ được cập nhật vào Driver_base trước để định nghĩa ra chuẩn chung giữa các IoT Platform. Sau đó và việc cập nhật thêm các driver mà kế thừa từ Driver_Base này.

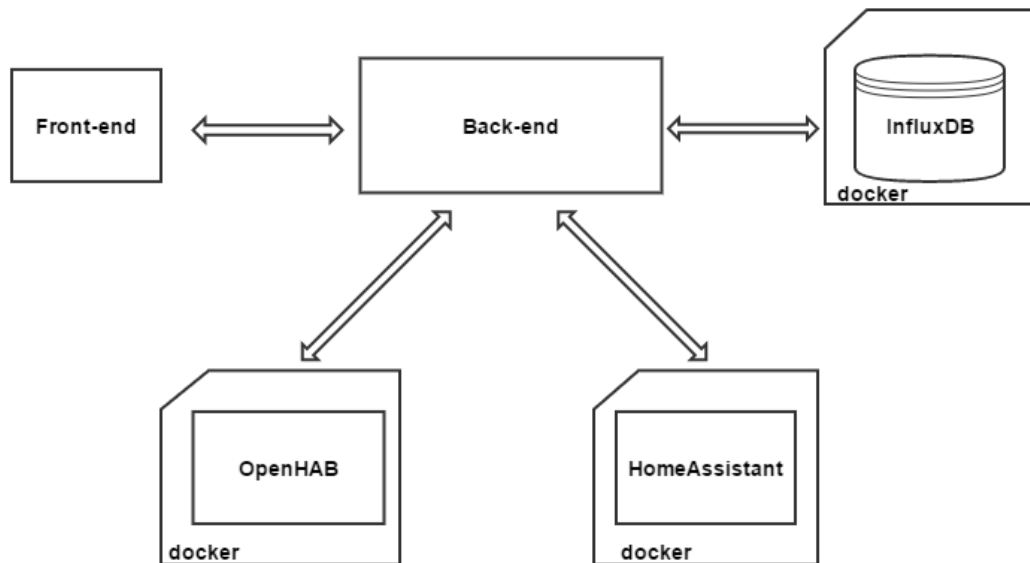
Tính khả mở theo chiều ngang:

Khi muốn tích hợp thêm các IoT Platform khác vào hệ thống, thì trước tiên cần viết driver cho Platform đó. Việc viết driver sẽ kế thừa từ các hàm và thuộc tính của Driver_base. Sau khi viết xong driver thì chỉ cần khai báo driver này vào trong Drive_client là có thể sử dụng. Điều này rất tiện lợi vì toàn bộ lớp mô phỏng và cung cấp API ở phía trên đều được giữ nguyên.

Chương III, Kiểm thử hệ thống

1. Cài đặt môi trường kiểm thử

Môi trường kiểm thử của hệ thống được thiết lập như hình sau:



Hình 34. Mô hình kiểm thử

Hệ thống kiểm thử hiện đang được triển khai trên một node duy nhất chạy Ubuntu 16.04 với cấu hình:

- 4GB RAM
- CPU 4 lõi
- Ổ cứng SSD 120GB.

Giải thích các thành phần trong mô hình kiểm thử ở trên:

- **Front-end** là web interface, chạy ở một server riêng (do npm quản lý)
- **Back-end** gồm các thành phần StateService, RulesService và cung cấp API cho người dùng và được chạy bằng server Python 2.7
- **OpenHAB** và **Home-Assitant** là hai IoT Platform đang được tích hợp trong hệ thống. Vì hệ thống chạy trên một máy, để đảm bảo tính độc lập và phân tán của hai IoT Platform này. Hệ thống đã dùng môi trường **Docker Container** để 2 IoT Platform có thể hoạt động độc lập nhau.
- **InfluxDB** cũng được đưa lên môi trường **Docker Container** để có thể khởi động nhanh hơn.

Trong mô hình trên OpenHAB được cài đặt để quản lý các thiết bị:

- Ceiling Light 1
- Ceiling Light 2
- OutSide Light
- TV Back Light
- Ceiling Fan
- Air Conditional
- Window1
- Window2
- Temperature Sensor

- Humidity Sensor
- Và Home-Assitant quản lí các thiết bị:
- Temperature
 - Humidity
 - Ventilator
 - Garage Light

2. Tiến hành kiểm thử

Danh sách các chức năng được kiểm thử

STT	Tên	Mô tả	Mục đích
1	Xem dữ liệu	Vào trang điều khiển, thay đổi trạng thái của các thiết bị xem dữ liệu có được cập nhật liên tục không	Kiểm tra chức năng có hoạt động ổn định hay không
2	Bật/Tắt thiết bị	Vào trang điều khiển, bật tắt thiết bị thiết bị có thay đổi trạng thái không	Kiểm tra chức năng có hoạt động ổn định hay không
3	Xem lịch sử thiết bị	Từ trang điều khiển, click vào thiết bị, xem hệ thống có chuyển sang trang hiển thị biểu đồ lịch sử thiết bị không. Đồng thời xem biểu đồ có hoạt động bình thường không.	Kiểm tra chức năng có hoạt động ổn định hay không
4	Chạy thử kịch bản tự động hóa 1	Kịch bản: Khi nhiệt độ trong phòng khách lớn hơn 30 thì bật quạt lên.	Kiểm tra chức năng có hoạt động ổn định hay không
5	Chạy thử kịch bản tự động hóa 2	Kịch bản: Khi trời quá 6h tối thì bật tắt cả đèn lên	Kiểm tra chức năng có hoạt động ổn định hay không

2.1 Xem dữ liệu

- Điều kiện tiên quyết: Tất cả các thành phần trong hệ thống đã được khởi chạy.
- Kết quả mong đợi: Hệ thống cập nhật dữ liệu của sensor một cách liên tục.
- Các bước thực hiện
 1. Vào giao diện quản lí để xem trạng thái hiện tại
 2. Thay đổi trạng thái từ phía IoT Platform và xem giao diện có cập nhật trạng thái không
- Kết quả kiểm thử:

STT	Thay đổi trạng thái thiết bị	Kết quả cập nhật
1	Thay đổi trạng thái Ceiling Light 1	Thành công
2	Thay đổi trạng thái Fan	Thành công
3	Thay đổi trạng thái Temperature	Thành công
4	Thay đổi trạng thái Ventilator	Thành công
5	Thay đổi trạng thái Garage Light	Thành công

2.2 Bật/Tắt thiết bị

- Điều kiện tiên quyết: Tất cả các thành phần trong hệ thống đều được khởi chạy.
- Kết quả mong đợi: Khi thay đổi trạng thái ở giao diện chính thì trạng thái của sensor cũng thay đổi theo (theo dõi ở các IoT Platform)
- Các bước thực hiện:
 1. Vào giao diện điều khiển, bấm nút bật tắt thiết bị.
 2. Theo dõi xem thiết bị có được bật tắt đúng với câu lệnh hay không.
- Kết quả kiểm thử:

STT	Hành động	Kết quả cập nhật
1.	Bật tắt Ceiling Light1	Thành công
2	Bật tắt Air Conditional	Thành công
3	Bật tắt TV Back Light	Thành công
4	Bật tắt Ceiling Light2	Thành công
5	Bật tắt Vanlidator	Thành công

2.3 Xem lịch sử trạng thái thiết bị

- Điều kiện tiên quyết: Tất cả thành phần trong hệ thống đều được khởi chạy, đang ở giao diện quản lý.
- Kết quả mong đợi: Khi click vào mỗi thiết bị ở giao diện quản lý thì sẽ chuyển trang và hiển thị biểu đồ trạng thái của thiết bị. Biểu đồ này được cập nhật theo thời gian.
- Các bước thực hiện:
 1. Vào giao diện trạng thái, bấm vào tên một thiết bị bất kì
 2. Theo dõi xem chuyển sang trang mới có mượt không, biểu đồ cập nhật có liên tục không.
- Kết quả kiểm thử:

STT	Hành động kiểm thử	Kết quả
1	Vào trang xem lịch sử của Ceiling Light2	Thành công, hệ thống chạy ổn định
2	Vào trang xem lịch sử của Humidity (OpenHAB)	Lỗi, cập nhật realtime bị gián đoạn
3	Vào lại trang xem lịch sử của Humidity (OpenHAB)	Thành công, hệ thống chạy ổn định
4	Vào trang xem lịch sử của temperature	Thành công, hệ thống chạy ổn định
5	Vào trang xem lịch sử của humidity	Thành công, hệ thống chạy ổn định

2.4 Chạy kịch bản tự động 1

```
If getValue(Temperature) > 30 then{  
  setState(Fan,ON)  
}
```

Điều kiện tiên quyết: Hệ thống vận hành bình thường

Kết quả mong đợi: Quạt sẽ tự động bật khi nhiệt độ phòng trên 30 độ

Các bước thực hiện:

- Tắt quạt đi
- Copy file kịch bản vào trong thư mục kịch bản
- Truyền nhiệt độ phòng vào giá trị lớn hơn 30 °C hoặc nhỏ hơn 30 °C
- Nếu nhiệt độ phòng nhỏ hơn 30 °C, quạt sẽ tự bật, nếu nhỏ hơn thì sẽ không có gì xảy ra cả.

Kết quả kiểm thử:

STT	Nhiệt độ truyền vào	Kết quả
1	31 °C	Quạt bật
2	40 °C	Quạt bật
3	29 °C	Quạt không bật
4	10 °C	Quạt không bật
5	35 °C	Quạt bật

2.5 Chạy kịch bản tự động 2

```
If time > 18h00 then
{
  setStateByType(Light, ON)
}
```

Điều kiện tiên quyết: Hệ thống vận hành bình thường

Kết quả mong đợi: Sau 6h tối thì sẽ tự động bật tất cả đèn lên.

Các bước thực hiện:

- Sao chép file kịch bản vào thư mục kịch bản
- Cài đặt đồng hồ để chạy gần đến 6h tối
- Theo dõi trạng thái của đèn trước và sau 6h tối

Kết quả kiểm thử:

STT	Kết quả
1	Đèn đều bật lên
2	Đèn đều bật lên
3	Đèn đều bật lên
4	Đèn đều bật lên
5	Đèn đều bật lên

Chương VI, Kết luận và hướng phát triển

1 Kết luận

Đồ án đã thu được kết quả sau:

- Định nghĩa được một lớp trừu tượng giữa các IoT Platform khác nhau để có thể dễ dàng truy vấn và điều khiển dữ liệu đồng thời cung cấp các API và giao diện đồ họa thân thiện, đơn giản, hiệu quả.
- Cung cấp chế độ tự động chạy kịch bản
- Đồ án có tính khả mở cao.
- Lưu trữ dữ liệu thu thập được từ các IoT Platform
- Cập nhật dữ liệu các thiết bị theo thời gian thực

Trong đó, đóng góp chính của đồ án là việc đặt ra một lớp trừu tượng, lấy đó làm tiêu chuẩn để kết nối các IoT Platform khác nhau vào một khối. Đồng thời có thể dễ dàng mở rộng, tích hợp các IoT Platform khác.

2. Hướng phát triển

2.1 Tích hợp thêm nhiều IoT Platform

Mô tả:

Trong khuôn khổ của đồ án và giới hạn số lượng thời gian cho phép, hệ thống hiện tại mới chỉ tích hợp được 2 IoT Platform vào với nhau. Định hướng phát triển là tích hợp thêm nhiều các IoT Platform khác để hệ thống đa năng và hoàn thiện hơn.

Hướng tiếp cận:

Xác định các IoT Platform đang được ưa chuộng mà cung cấp giao tiếp qua REST API. Viết driver cho các IoT platform này và tích hợp vào hệ thống.

2.2 Phát triển hệ thống AI

Mô tả:

Việc phát triển và tích hợp thêm AI để hệ thống có thể học được hành vi của người dùng, từ đó có thể tối ưu trải nghiệm người dùng bằng những điều khiển tự động và phù hợp nhất.

Hướng tiếp cận:

Thu thập thêm dữ liệu người dùng và phân tích hành vi bằng phương pháp học máy, từ đó đưa ra được những quyết định hợp lý nhất.

Tài liệu Tham khảo

- [1] <http://wikipedia.com.vn> truy cập lần cuối 5/12/2016
- [2] Dave Evans, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, *on Cisco*
- [3] <http://www.rfidjournal.com/articles/view?4986> truy cập lần cuối 5/12/2016
- [4] <http://www.cognitiveclouds.com/insights/the-internet-of-things-iot-tech-stack-explained/> truy cập lần cuối 8/12/2016 truy cập lần cuối 6/12/2016
- [5] OpenHab HomePage, www.openhab.org truy cập cuối cùng ngày 7/12/2016
- [6] HomeAssistant Homepage, <https://home-assistant.io/> truy cập cuối cùng ngày 8/12/2016
- [7] Kura Introduction, <http://eclipse.github.io/kura/doc/intro.html> truy cập cuối cùng ngày 8/12/2016
- [8] IoT software platform comparison, <https://dzone.com/articles/iot-software-platform-comparison> truy cập cuối cùng ngày 9/12/2016
- [9] <https://iot-analytics.com/5-things-know-about-iot-platform/> truy cập lần cuối 11/12/2016
- [10] Duc-Hung Le, Nanjangud Narendra, Hong-Linh Truong “HINC – Harmonizing Diverse Resource Information Across IoT, Network Functions and Clouds”
- [11] S. Haller, A. Serbanati, M. Bauer, and F. Carrez, “A domain model for the internet of things,” in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCoM), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 411–417.
- [12] S. Oteafy and H. Hassanein, “Towards a global iot: Resource reutilization in wsns,” in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, Jan 2012, pp. 617–622.
- [13] H. Zhang and C. Meng, “A multi-dimensional ontology-based iot resource model,” in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, June 2014, pp. 124–127.
- [14]