|  |  |
| --- | --- |
|  |  |
| HÀ QUANG DƯƠNG | **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **---------------------------------------** |
|  |
|  |
|  |
|  |
| **Họ và tên tác giả luận văn** |
|  |
|  |
| **HÀ QUANG DƯƠNG** |
|  |
|  |
| CÔNG NGHỆ THÔNG TIN |  |
|  |
| **TÊN ĐỀ TÀI LUẬN VĂN** |
|  |
|  |
| Một framework cung cấp khả năng tương tác trong môi trường IoT |
|  |
|  |
|  |
|  |  |
|  | LUẬN VĂN THẠC SĨ KĨ THUẬT  CÔNG NGHỆ THÔNG TIN |
|  |  |
|  |  |
| KHOÁ 2016B |  |
|  |
|  |
|  |
| Hà Nội – Năm 2019 |

|  |
| --- |
|  |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **---------------------------------------** |
|  |
| **Họ và tên tác giả luận văn** |
|  |
| **HÀ QUANG DƯƠNG** |
|  |
|  |
|  |
| **TÊN ĐỀ TÀI LUẬN VĂN** |
|  |
| Một framework cung cấp khả năng tương tác trong môi trường IoT |
|  |
|  |
| Chuyên ngành: Công nghệ thông tin |
|  |
|  |
|  |
|  |
|  |
| LUẬN VĂN THẠC SĨ KĨ THUẬT  Công nghệ thông tin |
|  |
|  |
|  |
|  |
| NGƯỜI HƯỚNG DẪN KHOA HỌC : |
| 1. Tiến sĩ Vũ Tuyết Trinh |
|  |
|  |
|  |
| Hà Nội – Năm 2019 |

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

**Độc lập – Tự do – Hạnh phúc**

**BẢN XÁC NHẬN CHỈNH SỬA LUẬN VĂN THẠC SĨ**

**Họ và tên tác giả luận văn :** …………………………………........……………..

**Đề tài luận văn:** ………………………………………….....……………...............….

**Chuyên ngành:**……………………………...…………………........................…..........

**Mã số SV**:………………………………….. …………………....................................…...

Tác giả, Người hướng dẫn khoa học và Hội đồng chấm luận văn xác nhận tác giả đã sửa chữa, bổ sung luận văn theo biên bản họp Hội đồng ngày….........................………… với các nội dung sau:

……………………………………………………………………………………………………..……………………………………………………………………………………………………..……………………………………………………………………………………………………..……………………………………………………………………………………………………..……………………………………………………………………………………………………..……………………………………………………………………………………………………..……………………………………………………………………………………..

Ngày tháng năm

**Giáo viên hướng dẫn Tác giả luận văn**

**CHỦ TỊCH HỘI ĐỒNG**

# MỤC LỤC

# Trang

[MỤC LỤC 4](#_Toc4701889)

[Trang 4](#_Toc4701890)

[DANH MỤC CÁC HÌNH VẼ 6](#_Toc4701891)

[DANH MỤC CÁC BẢNG 8](#_Toc4701892)

[DANH MỤC CÁC TỪ VIẾT TẮT 9](#_Toc4701893)

[MỞ ĐẦU 10](#_Toc4701894)

[MỤC TIÊU CỦA LUẬN VĂN 14](#_Toc4701895)

[ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 15](#_Toc4701896)

[CÁCH TIẾP CẬN VÀ PHƯƠNG PHÁP NGHIÊN CỨU 16](#_Toc4701897)

[CHƯƠNG 1: GIỚI THIỆU 18](#_Toc4701898)

[1. Tổng quan tình hình nghiên cứu và phát triển công nghệ vạn vật kết nối Internet (IoT) 18](#_Toc4701899)

[1.1. Phân tích, đánh giá tình hình nghiên cứu và phát triển vạn vật kết nối Internet trên thế giới 18](#_Toc4701900)

[1.2. Phân tích, đánh giá tình hình nghiên cứu và phát triển vạn vật kết nối Internet trong nước 22](#_Toc4701901)

[2. Tính cấp thiết của đề tài 27](#_Toc4701902)

[CHƯƠNG 2: NỀN TẢNG CÔNG NGHỆ 32](#_Toc4701903)

[1. Khái niệm về các nền tảng phần mềm IoT 32](#_Toc4701904)

[2. Đánh giá một số nền tảng IoT 33](#_Toc4701905)

[3. Công nghệ MQTT hàng đợi thông điệp cho IoT 37](#_Toc4701906)

[CHƯƠNG 3: GIẢI PHÁP KẾT NỐI CÁC NỀN TẢNG IOT 39](#_Toc4701907)

[1. Triển khai thử nghiệm mô hình IoT 39](#_Toc4701908)

[1.1. Kiến trúc tổng quan 39](#_Toc4701909)

[1.2. Các thiết bị triển khai ở trung tâm Tính toán Hiệu năng cao 41](#_Toc4701910)

[1.3. Hệ thống PAS 42](#_Toc4701911)

[2. Thiết kế và phát triển phần mềm tích hợp các nền tảng IoT 46](#_Toc4701912)

[2.1. Mô hình dữ liệu tập trung 46](#_Toc4701913)

[2.2. Các thành phần dịch vụ 48](#_Toc4701914)

[2.3. Lưu trữ 49](#_Toc4701915)

[2.4. Các giao thức phục vụ giao tiếp 50](#_Toc4701916)

[2.5. Engine xử lý luật 53](#_Toc4701917)

[CHƯƠNG 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ 56](#_Toc4701918)

[1. Kịch bản kiểm thử 56](#_Toc4701919)

[2. Thử nghiệm hạ tầng với các nền tảng IoT 57](#_Toc4701920)

[3. Thử nghiệm đánh giá thời gian khi thực hiện một số thao tác với hệ thống tích hợp các nền tảng IoT 60](#_Toc4701921)

[4. Xây dựng ứng dụng cảnh báo với dữ liệu thu thập từ ba nền tảng IoT 66](#_Toc4701922)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 70](#_Toc4701923)

[TÀI LIỆU THAM KHẢO 72](#_Toc4701924)

# DANH MỤC CÁC HÌNH VẼ

[Hình 1: Kiến trúc 3 tầng của IoT 28](#_Toc4701978)

[Hình 2: Kiến trúc điển hình của nền tảng IoT 33](#_Toc4701979)

[Hình 3: Kiến trúc của MQTT [2] 37](#_Toc4701980)

[Hình 4: Luồng dữ liệu giữa các thành phần phần cứng 40](#_Toc4701981)

[Hình 5: Sơ đồ triển khai tại HPC 41](#_Toc4701982)

[Hình 6: Mô hình triển khai 42](#_Toc4701983)

[Hình 7: Mô hình lắp đặt các thiết bị 43](#_Toc4701984)

[Hình 8: Kiến trúc tổng quan của hệ thống PAS 44](#_Toc4701985)

[Hình 9: Mô hình luồng dữ liệu trong hệ thống PAS 45](#_Toc4701986)

[Hình 10: Mô hình dữ liệu 47](#_Toc4701987)

[Hình 11: Kiến trúc tổng quan của hệ thống phần mềm tích hợp các nền tảng IoT 48](#_Toc4701988)

[Hình 12: Engine xử lý các luật điều khiển đa nền tảng IoT 54](#_Toc4701989)

[Hình 13: Giao diện của HomeAssistant 58](#_Toc4701990)

[Hình 14: Giao diện của OpenHab 58](#_Toc4701991)

[Hình 15: Giao diện của ThingsBoard 59](#_Toc4701992)

[Hình 16: Mô hình triển khai trong quá trình kiểm thử 61](#_Toc4701993)

[Hình 17: Biểu đồ trình tự quá trình thu thập dữ liệu từ nền tảng IoT 61](#_Toc4701994)

[Hình 18: Biểu đồ trình tự quá trình cập nhật dữ liệu vào hệ thống 61](#_Toc4701995)

[Hình 19: Biểu đồ trình từ quá trình lấy trạng thái của một thiết bị trong hệ thống 62](#_Toc4701996)

[Hình 20: Biểu đồ trình tự quá trình thay đổi trạng thái của thiết bị 62](#_Toc4701997)

[Hình 21: Biểu đồ trung bình kết quả khi thay đổi số lượng thiết bị 63](#_Toc4701998)

[Hình 22: Biểu đồ trung bình kết quả khi thay đổi số lượng thiết bị 64](#_Toc4701999)

[Hình 23: Biểu đồ trung bình kết quả khi thay đổi chu kỳ của các thiết bị 65](#_Toc4702000)

[Hình 24: Mô hình xây dựng ứng dụng giám sát 66](#_Toc4702001)

[Hình 25: Biểu đồ trình tự của ứng dụng giám sát 67](#_Toc4702002)

[Hình 26: Màn hình ứng dụng giám sát 67](#_Toc4702003)

[Hình 27: Màn hình khi ứng dụng đưa ra cảnh báo 68](#_Toc4702004)

# DANH MỤC CÁC BẢNG

[Bảng 1: So sánh một số các nền tảng IoT hiện nay 36](#_Toc4702005)

[Bảng 2: So sánh các giao thức giao tiếp của IoT 38](#_Toc4702006)

[Bảng 3: Cấu trúc lệnh điều chỉnh luồng dữ liệu 51](#_Toc4702007)

[Bảng 4: Cấu trúc lệnh điều chỉnh luồng dữ liệu 53](#_Toc4702008)

[Bảng 5*:* Các thuộc tính của một cấu hình 53](#_Toc4702009)

[Bảng 6: Kịch bản kiểm thử hệ thống 56](#_Toc4702010)

[Bảng 7: Kết quả khi thực hiện một số thao tác với các nền tảng IoT 59](#_Toc4702011)

[Bảng 8: Kết quả khi thực hiện một số trường hợp kiểm thử với ứng dụng giám sát 68](#_Toc4702012)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **VIẾT TẮT** | **ĐẦY ĐỦ** | **CHÚ THÍCH** |
| API | Application Programming Interface | Giao diện lập trình |
| AWS | Amazon Web Services | Một dịch vụ đám mây |
| CoAP | Constrained Application Protocol | Một giao thức truyền dữ liệu của IoT |
| DDoS | Denial-of-service Attack | Tấn công từ chối dịch vụ |
| HTTP | Hypertext Transfer Protocol | Giao thức truyền dữ liệu |
| IaaS | Infrastructure as a Service | Hạ tầng như một dịch vụ |
| IoT | Internet of Things | Vạn vật kết nối |
| JVM | Java Virtual Machine | Máy ảo Java |
| M2M | Machine to Machine |  |
| MQTT | Message Queuing Telemetry Transport | Một giao thức truyền dữ liệu của IoT |
| PAS | Person Authentication System | Hệ thống quản lý người ra vào sử dụng thẻ RFID |
| SSO | Single Sign-on | Đăng nhập 1 lần |
| VM | Virtual Machine | Máy ảo |
| WSN | Wireless Sensor Network | Mạng cảm biến không dây |
| YAML | YAML Ain't Markup Language | Một ngôn ngữ đặc tả dữ liệu |

# MỞ ĐẦU

Vạn vật kết nối Internet (Internet of Things – IoT) đã và đang trở thành xu hướng thu hút rất nhiều các nhà khoa học, các doanh nghiệp và tổ chức nghiên cứu và phát triển ứng dụng những năm trở lại đây. Khái niệm và mô hình hoạt động của IoT liên quan mật thiết tới sự xuất hiện khắp nơi của các vật (thing) và đối tượng (object) trong môi trường thực tế. Các vật và đối tượng này sẽ kết nối với nhau thông qua mạng không dây hoặc có dây. Mỗi một vật hoặc đối tượng đều được đánh địa chỉ hoặc được đặt tên trong mạng lưới đó. Nhờ vậy, chúng có khả năng liên kết và cùng hoạt động dưới dạng các quy trình hoặc giao diện của ứng dụng, và dịch vụ “thông minh” nhằm hỗ trợ cho cuộc sống của con người trong mọi lĩnh vực. Theo hướng này, mục tiêu của IoT là cho phép các vật và các đối tượng có thể kết nối với nhau vào bất kì lúc nào (Anytime), bất kì nơi đâu (Anyplace) với bất kì vật nào khác (Anything) hoặc ai khác (Anyone) sử dụng các hệ thống mạng khác nhau (4A). Có thể nói, IoT chính là cuộc cách mạng mới của Internet, các đối tượng kết nối trong đó trở nên thông minh khi tự đưa ra các quyết định, chiến lược hoạt động nhờ có dữ liệu và thông tin mà chúng trao đổi với nhau.

IoT cũng là sự kết hợp của nhiều ngành công nghệ khác nhau. Ví dụ: trong ngữ cảnh gồm các cột đèn đường thông minh và có khả năng kết nối Internet, mọi thiết bị có khả năng cảm thụ môi trường khi đi qua khu vực đó có thể liên kết với các cột đèn này để truyền và nhận dữ liệu. Trong ví dụ này, các công nghệ liên quan tới phần cứng cảm biến, lập trình phần mềm điều khiển nhúng, thậm chí nhận dạng ảnh trong thời gian thực (với các cảm biến về chuyển động, giám sát đám đông), các công nghệ kết nối, v.v… đều hiện diện và đóng vai trò quan trọng trong toàn bộ chuỗi công nghệ để tạo ra ứng dụng, dịch vụ IoT [4]. Các công nghệ cho IoT được tổng kết trong nghiên cứu [26] có thể chia thành ba nhóm chính sau:

* Nhóm các công nghệ cho phép các vật nhận biết được thông tin ngữ cảnh/môi trường.
* Nhóm các công nghệ cho phép các vật xử lý thông tin ngữ cảnh/môi trường.
* Nhóm các công nghệ đảm bảo an toàn và tính riêng tư của dữ liệu.

Hai nhóm công nghệ đầu tiên có thể cùng hiểu là các công cụ, phương pháp, chức năng cần thiết để xây dựng tính “thông minh” cho các vật. Đây là sự khác biệt giữa IoT và mạng Internet thông thường. Nhóm công nghệ thứ ba không phải là công cụ, phương pháp hay chức năng nhưng là các yêu cầu cần có cho IoT. Thiếu các yêu cầu này, sự mở rộng của IoT có thể bị hạn chế. Sự phát triển của IoT ẩn dụ rằng: các môi trường, thành phố, tòa nhà, máy móc, quần áo, các thiết bị di động và các đối tượng khác sẽ ngày càng tạo ra nhiều thông tin, dữ liệu và chúng được liên kết với nhau để cảm thụ, giao tiếp và tạo ra các thông tin và tri thức mới. Bên cạnh đó, mạng kết nối là vấn đề cần đầu tư nghiên cứu và là yếu tố công nghệ rất quan trọng cho IoT. Các cách thức giao tiếp này cần băng thông cao, đảm bảo cho một lượng lớn thiết bị hoặc vật kết nối, có độ trễ truyền dữ liệu nhỏ giữa nguồn và đích đến, tiêu thụ năng lượng thấp, chi phí đầu tư lắp đặt nhỏ. Công nghệ 5G ứng dụng trong tương lai gần sẽ là kim chỉ nam cho sự phát triển của IoT được tổng kết trong [37].

Theo như báo cáo phân tích của IDC [39] số lượng các vật hoặc thiết bị đang được nối mạng trên toàn thế giới hiện nay ước tính khoảng 30 tỉ, tuy nhiên con số này được dự đoán sẽ tăng lên đạt 212 tỉ chỉ trong vài năm tới cho tới trước năm 2020. IDC đã dự đoán sự tăng trưởng rất nhanh của IoT trong tương lai gần cùng đó là sự ra đời của rất nhiều các hệ thống thông minh, truy cập và phân tích dữ liệu. Các đặc điểm chính của IoT được tổng kết trong nghiên cứu [35] bao gồm:

1. *Tính kết nối:* trong môi trường IoT, bất kì vật nào cũng có thể được nối mạng để trao đổi dữ liệu với các vật khác.
2. *Tính liên kết dịch vụ:* IoT cho phép tạo ra mối quan hệ từ vật kết nối mạng tới dịch vụ đầu cuối trong một ngữ cảnh thông minh nào đó. Theo hướng này, các công nghệ hỗ trợ cho IoT và cách thức xử lý thông tin sẽ thay đổi để phù hợp với nhu cầu truyền dữ liệu từ vật tới dịch vụ (cần phù hợp, nhanh chóng và hiệu quả hơn) và ngược lại.
3. *Tĩnh hỗn tạp:* các thiết bị trong IoT là hỗn tạp bao gồm các nền tảng phần cứng, phần mềm, và mạng.
4. *Tính hay thay đổi:* trạng thái của các thiết bị, vật sẽ thường xuyên thay đổi, ví dụ như các thiết bị sẽ có thời gian “ngủ” và “thức”, hoặc “kết nối” hay “không kết nối”. Thậm chí, trong môi trường IoT, trạng thái của ngữ cảnh sẽ bị phụ thuộc vào số lượng các thiết bị được kết nối với nhau theo thời gian.
5. *Tính mở rộng*: số lượng các thiết bị IoT cần được điều khiển và quản lý sẽ lớn hơn rất nhiều các thiết bị kết nối Internet ngày nay. Bên cạnh đó, số lượng dữ liệu và thông tin cần xử lý sẽ rất lớn, việc phân tích đưa ra các cư xử thông minh dựa vào dữ liệu đó cũng là một thách thức cho các nhà lập trình ứng dụng IoT.

Trong luận văn này, mô hình IoT được xem xét bao gồm có 3 tầng chính gồm có: tầng các thiết bị cảm biến, tầng truyền dẫn (hay còn gọi là điện toán sương mù – fog computing hoặc điện toán biên – edge computing), và tầng dịch vụ điện toán đám mây. Ở tầng cảm biến, thông thường số lượng các cảm biến trong IoT sẽ là rất lớn. Nhiệm vụ chủ yếu của các cảm biến là cảm thụ, giám sát môi trường sau đó truyền dữ liệu thu thập được về phía các tầng thấp hơn như điện toán sương mù (điện toán biên) và điện toán đám mây để xử lý.

Trong tầng điện toán sương mù, việc tồn tại rất nhiều các nền tảng IoT, sử dụng các công nghệ giao tiếp khác nhau và việc khai thác các nền tảng này vào bài toán cụ thể không tuân theo chuẩn (standard) sẽ ảnh hưởng tới khả năng mở rộng phạm vi kết nối các vật hay thiết bị IoT trong thực tế. Ví dụ, một tòa nhà thông minh có thể dùng OpenM2M như một công cụ quản lý thiết bị thông minh, tuy nhiên, tòa nhà thứ hai bên cạnh đó lại dùng nền tảng khác như Kura hay ThingSpeak để thực hiện các tác vụ tương tự. Vấn đề đặt ra ở đây khi kết hợp hai tòa nhà với các thiết bị thông minh này thành một giải pháp IoT tổng thể cho phép quản lý, kết nối và điều khiển chúng trong một ngữ cảnh chung, công việc đầu tiên cần phải làm là liên kết được các nền tảng này đã được triển khai sẵn này lại với nhau. Việc mở rộng phạm vi lớn hơn như trình bày ở trên sẽ dẫn tới nhu cầu phải có một công cụ cho phép nhiều nền tảng IoT với các công nghệ khác nhau nói chuyện được với nhau. Từ đó mô hình IoT mới đạt được khả năng kết nối các thiệt bị hay vật một cách rộng khắp như định nghĩa của nó.

Trong nghiên cứu này, tôi tập trung giải quyết vấn đề của sự tồn tại đa nền tảng IoT trong quá trình triển khai các dịch vụ hoặc ứng dụng thông minh trong phạm vi trung bình và rộng (tòa nhà thông mình, hoặc thành phố thông minh). Theo đó, tôi sẽ xây dựng một khung lập trình cho phép các nhà lập trình ứng dụng IoT có khả năng tương tác, sử dụng các nền tảng IoT khác nhau. Lợi ích của khung lập trình này là giúp các nhà phát triển ứng dụng IoT giảm bớt khó khăn về mặt kĩ thuật (do sự khác biệt), và vấn đề thời gian (cần phải làm quen) khi tạo ra các ứng dụng của họ trong môi trường IoT trong phạm vi lớn.

# MỤC TIÊU CỦA LUẬN VĂN

Xây dựng khung phần mềm hỗ trợ dễ dàng liên kết các nền tảng, dịch vụ, ứng dụng, thiết bị trong môi trường IoT, tập trung ở lớp điện toán sương mù được triển khai trên các gateway.

# ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

**Đối tượng nghiên cứu**

Vấn đề hiện nay của IoT là cùng lúc có rất nhiều các nhà cung cấp phần mềm nền tảng cho tầng điện toán sương mù cũng như điện toán đám mây, sự xuất hiện của rất nhiều các dịch vụ này chính là rào cản cho sự mở rộng của IoT khi người dùng cần kết nối mạng lưới các vật (thing) trong một phạm vi rộng. Chính vì vậy, đề tài tập trung nghiên cứu đối tượng sau: Phần mềm cho phép quản lý, điều khiển các nền tảng IoT khác nhau triển khai trên các gateway trong điện toán sương mù.

**Phạm vi nghiên cứu**

Phạm vi đề tài này được đặt dưới góc nhìn của công nghệ thông tin, hệ thống thông tin và phần mềm nền tảng cho phép quản lý. Trong khi đó, các yếu tố công nghệ liên quan tới phần cứng và giao tiếp, mạng chưa được xét tới trong đề tài này. Cụ thể đề tài giới hạn trong phạm vi sau:

1. Nghiên cứu nhằm tích hợp các nền tảng phần mềm IoT phổ biến trong điện toán sương mù. Giải pháp phần mềm tích hợp này sẽ cho phép khả năng mở rộng (tích hợp được) các nền tảng khác một cách dễ dàng.
2. Mô hình quản trị dữ liệu tập trung (data-centric) của điện toán sương mù cho phép truy vấn thông tin và thực hiện các bài toán tối ưu.

# CÁCH TIẾP CẬN VÀ PHƯƠNG PHÁP NGHIÊN CỨU

**Cách tiếp cận**

Trên cơ sở nghiên cứu tài liệu trong và ngoài nước có liên quan đến đề tài nhằm đưa ra và đánh giá các vấn đề còn tồn tại của điện toán sương mù và điện toán đám mây hiện nay trong bối cảnh IoT đang phát triển nhanh chóng, tôi nghiên cứu các kỹ thuật liên quan đến các phần mềm nền tảng triển khai trong điện toán sương mù, từ đó xây dựng giải pháp với giao diện lập trình (API) thống nhất quản lý các thiết bị thuộc các nền tảng IoT có sẵn được triển khai trong thực tế. Đề tài sẽ áp dụng các cách tiếp cận nghiên cứu như sau:

* *Cách tiếp cận tích hợp hệ thống*: Có hai hệ thống, mô hình cần nghiên cứu xây dựng khi áp dụng vào luận văn. (1) hệ thống tích hợp các phần mềm nền tảng IoT trong tầng điện toán sương mù (2) Mô hình dữ liệu cho phép truy vấn thông tin hiệu quả trong môi trường sương mù.
* *Cách tiếp cận xây dựng lớp trừu tượng lập trình*: Với mục tiêu tạo ra một giải pháp không phụ thuộc vào các nhà cung cấp dịch vụ (không yêu cầu các nhà cung cấp dịch vụ phải tích hợp vào hệ thống của họ), cách tiếp cận giải quyết vấn đề này đó là tạo ra các lớp trừu tượng lập trình (abstraction programming) cho phép định nghĩa lại các API, và chức năng được cung cấp bởi các sản phẩm, dịch vụ có sẵn dưới một lớp giao diện (interface) thống nhất. Về cơ bản, việc tạo ra các lớp trừu tượng sẽ cho phép đóng gói (encapsulate) thông tin, ẩn các chi tiết phức tạp (hide implementation detail) giúp người dùng và các nhà lập trình ứng dụng khác có khả năng dễ dàng quản trị, điều khiển các hệ thống khác nhau thông qua các đối tượng của phần mềm được định nghĩa (software-defined) đại diện cho các thiết bị vật lý.
* *Cách tiếp cận mô hình hóa dữ liệu*: Dữ liệu và thông tin của hệ thống tích hợp được lưu trữ tập trung cho phép quản trị các truy vấn, và các mối quan hệ giữa các tài nguyên, thiết bị IoT mà không cần chi tiết hóa các mệnh lệnh sinh ra từ phía người sử dụng. Việc mô hình hóa dữ liệu và quản trị dữ liệu dưới giao diện thống nhất cho phép tối ưu hóa nhiều công đoạn hoạt động trong một hệ thống tích hợp lớn. Đây là cũng chính là yếu tố quan trọng ảnh hưởng cách phát triển, mở rộng của IoT trong bối cảnh đa dịch vụ như đã phân tích ở các phần trên.

**Phương pháp nghiên cứu**

Các phương pháp nghiên cứu khác nhau sẽ được phối hợp để giải quyết các mục tiêu đặt ra của đề tài. Cụ thể như sau:

* Nghiên cứu triển khai giải pháp nền tảng trên các IoT gateway và các giải pháp đa đám mây nhằm đánh giá hai vấn đề tích hợp hệ thống của điện toán sương mù và đám mây cho IoT.
* Triển khai, thử nghiệm thực tế nguyên mẫu mô hình IoT với các thiết bị mua sẵn (cảm biến, gateway – Raspberry Pi), các dịch vụ đám mây nhằm đánh giá khả năng triển khai trong thực tế của các nền tảng IoT, và đa đám mây, thu thập dữ liệu trong thời gian thực. Đây là yếu tố cần thiết để hiểu và nắm bắt các công nghệ IoT liên quan tới phần mềm.
* Triển khai, tích hợp các giải pháp được phát triển cho phép quản lý, điều khiển nền tảng IoT trên gateway trong điện toán sương mù và đám mây với môi trường thử nghiệm IoT.
* Thử nghiệm và đánh giá hệ thống cho phép quản lý, điều khiển nền tảng IoT trên các gateway trong điện toán sương mù.

# CHƯƠNG 1: GIỚI THIỆU

## Tổng quan tình hình nghiên cứu và phát triển công nghệ vạn vật kết nối Internet (IoT)

### Phân tích, đánh giá tình hình nghiên cứu và phát triển vạn vật kết nối Internet trên thế giới

Mạng vạn vật kết nối (*Internet of Things* - IoT) theo cách nhìn nhận rộng rãi là một khái niệm về một thế gồm các đối tượng vật lý (các thiết bị, cảm biến) được nối kết qua Internet và là nền tảng công nghệ để thu thập, trao đổi các thông tin nhằm xây dựng các dịch vụ giá trị gia tăng trên đó. IoT đang dần thay đổi cuộc sống của chúng ta bằng việc cung cấp một môi trường giao tiếp thuận tiện và hiệu quả hơn. Các lĩnh vực ứng dụng tiềm năng của IoT phải kể đến bao gồm chăm sóc sức khoẻ, tự động hóa công nghiệp, lưới điện thông minh, nhà thông minh, thành phố thông minh, khu công nghiệp thông minh, giám sát năng lượng, giám sát môi trường, v.v… Cùng với sự phát triển của IoT cũng như các công nghệ như điện toán đám mây (cloud computing), dữ liệu lớn (big data), các công nghệ ngữ nghĩa (semantic technologies), v.v.. xu hướng tích hợp các công nghệ này trở nên khó tránh khỏi và sự tích hợp đó đem lại những cơ hội phát triển cả về công nghệ cũng như các ứng dụng.

*Một trong những đặc điểm của IoT là sự không đồng nhất của các thiết bị, nền tảng, công nghệ truyền thông (không dây, có dây), giao thức truyền thông, và các ứng dụng. Sự đa dạng của thiết bị, nền tảng, ứng dụng và dịch vụ IoT đặt ra một thách thức lớn và khó khăn cần phải được giải quyết để hỗ trợ hoạt động liên tục và có thể triển khai rộng rãi và hiệu quả IoT trong tương lai gần.*

Hiện nay, một số công trình trên thế giới đã giới thiệu các mô hình khác nhau để quản lý tài nguyên IoT, đặc biệt hướng vào giải quyết bài toán không đồng nhất của IoT như trình bày ở trên. Trong [53] mỗi quan hệ giữa các vật vật lý (physical things), cảm biến, điều khiển (actuators), v.v… được trình bày và phân tích chi tiết nhằm đề xuất một cơ chế có khả năng quản lý một cách đồng bộ trong môi trường IoT. Trong [54], Oteafy và cộng sự chỉ ra một mô hình hoạt động cho các đối tượng IoT và làm thế nào các tài nguyên chung đó có thể được cùng được sử dụng (khai thác) bởi ứng dụng. Zhang và cộng sự [74] cũng mô hình hóa tài nguyên như là các đối tượng lập trình bao gồm cả việc ánh xạ từ tài nguyên tới các dịch vụ/ứng dụng để cung cấp các chức năng quản lý cần thiết cho các tài nguyên IoT đó. Benazzouz và cộng sự [71] đề xuất trong dự án ClouT một mô hình thông tin quản lý tập trung cho các tài nguyên, dịch vụ và các thiết bị của IoT. Zhang và Meng [22] đưa ra mô hình tài nguyên đa chiều bao gồm chức năng, không gian, thời gian, và mức độ ưu tiên sử dụng. *Mặc dù vậy, tất cả các tiếp cận trên chủ yếu cung cấp mô hình chi tiết cho thiết bị IoT nhưng không một mô hình nào đề cập tới đặc tính và cách tiếp cận phân tán trong việc quản lý các thiết bị này*.

Ranjan và cộng sự [49] đề xuất kiến trúc tài nguyên phía trên đám mây (cloud) và hạ tầng IoT (các gateway) nhằm quản lý tài nguyên ảo hóa và thiết bị IoT hiệu quả hơn. Lopez và cộng sự [60] lại quan tâm tới độ phức tạp của mạng kết nối trong IoT khi quản lý các thiết bị và đề xuất kiến trúc phân cụm quản lý bằng cách chia nhỏ mạng ra thành nhiều thành phần đơn lẻ. Trong khi đó, OpenIoT [29] là dự án nghiên cứu thiết kế khung (framework) quản lý tài nguyên IoT. Trong dự án này, một phần mềm trung gian (middleware) được phát triển để truy xuất thông tin từ các cảm biến mà không cần biết chính xác cảm biến nào được sử dụng. Một vài giải pháp như phát hiện tài nguyên, quản lý việc cấp phát tài nguyên trong [48] và [52] đang được phát triển. *Tuy nhiên, các cách tiếp cận này chưa đưa ra mô hình áp dụng ở phạm vi rộng, đặc biệt là với ngữ cảnh sử dụng phần mềm định nghĩa toàn bộ các tài nguyên IoT.*

Việc liên kết và quản lý tài nguyên IoT trên các hệ thống đám mây cũng nhận được sự quan tâm rất lớn của cộng đồng nghiên cứu gần đây. Trong [73], [29] và [24], các tác giả chủ yếu đề cập tới hạ tầng ảo hóa IoT và việc quản lý chúng trên các nền tảng đám mây. Một số các tiếp cận khác như [59] và [3] sử dụng khía cạnh ngữ nghĩa (semantic) để tìm kiếm, liên kết và phối hợp khai thác các thiết bị IoT. Trong [7] và [34] các tác giả đề xuất khai thác các dịch vụ đám mây để tăng hiệu quả xử lý dữ liệu thu thập được từ các cảm biến. Nghiên cứu [13] và [33] trình bày kiến trúc SOA đề xuất cho phép kết nối các thiết bị IoT và dịch vụ ứng dụng triển khai trong đám mây. Trong [29] các tác giả lại tập trung vào phát triển hạ tầng ảo hóa cho phép xây dựng ứng dụng cảm biến và truyền dẫn như là một dịch vụ trên mây. Họ đề xuất một gói phần mềm bao gồm các chức năng: hỗ trợ quản lý thiết bị, kết hợp các thiết bị với nhau và cung cấp như dịch vụ trực tuyến. Mặc dù vậy, cách tiếp cận này lại tập trung vào một ngữ cảnh duy nhất. Nghiên cứu được trình bày trong [73] đưa ra hạ tầng cảm biến – đám mây theo cách ảo hóa các tài nguyên (ở đây là các cảm biến) và đặt trên đám mây dịch vụ, đồng thời cung cấp cơ chế quản lý và giám sát cho các cảm biến ảo (định nghĩa bằng phần mềm) này. Tuy nhiên các cảm biến dịch vụ này phù thuộc khá lớn vào các mẫu (template) được định nghĩa sẵn trong một vài ngữ cảnh rời rạc. SenaaS [3] là dự án phát triển phần mềm cảm biến dịch vụ tập trung vào ngữ nghĩa đám mây phía trên hạ tầng vật lý. Kết quả đề xuất của dự án này định nghĩa một lớp trung gian tương tác với các thiết bị và định dạng dữ liệu khá hỗn tạp (nhiều loại) của IoT cho phép trích xuất ra các luồng sự kiện để quản lý ở tầng đám mây.

Trong môi trường điện toán sương mù nơi bao gồm các IoT gateway, các thiết bị mạng, các đám mây dịch vụ nhỏ (cloudlet), một số các cách tiếp cận khác cũng đang nổi lên gần đây. Lấy ví dụ, trong bài báo [26] các tác giả trình bày về khái niệm điện toán sương mù với các đặc tính cơ bản là vị trí các thiết bị thường phân tán, mục tiêu của sương mù là giảm độ trễ khi truyền nhận dữ liệu, và tăng QoS. Nghiên cứu này cũng đề xuất định nghĩa một nền tảng ảo quản lý các thiết bị trong tầng sương mù, cho phép phát triển và triển khai các ứng dụng logic (phần mềm định nghĩa) trên tài nguyên được ảo hóa. Tương tự như vậy Cloudlets [55] và các đám mây nhỏ [56] được đề xuất với vai trò tính toán trên các nút hạ tầng ở điện đoán sương mù. Mô hình kiến trúc này có thể ứng dụng để giảm độ trễ của mạng, giảm thời gian xử lý và chi phí tính toán. *Tuy nhiên, đáng tiếc rằng, việc tối ưu truy vấn bằng mô hình hóa dữ liệu lại chưa được đề cập tới trong các nghiên cứu trên.*

Một số các tiếp cận khác khai thác sự mở rộng của khái niệm phần mềm định nghĩa nhằm quản lý tài nguyên đám mây IoT, ví dụ như phần mềm định nghĩa lưu trữ [63], phần mềm định nghĩa trung tâm dữ liệu [12], đặc biệt gần đây nổi lên là phần mềm định nghĩa mạng – SDN [31] cho phép quản trị và lập trình được các tài nguyên phần cứng và phần mềm. Kì vọng vào hiệu quả của công nghệ 5G [50] thu hút các nghiên cứu tập trung vào việc tối ưu hóa sự sử dụng mạng cho các ứng dụng IoT là một mục tiêu nghiên cứu của dự án NOVI (tài trợ bởi EU), tác giả van der Ham và cộng sự [30] đưa ra tập hợp các mô hình thông tin cho việc định nghĩa bằng phần mềm tài nguyên mạng ảo và các dịch vụ liên quan. Một số mô hình quản lý mạng khác được biết đến như NetJSON [44] và YANG [72], một số các công cụ SDN nổi lên là OpenDayLight [46] và Weave [70]. *Tuy nhiên các giải phải này tập trung vào một lĩnh vực hạ tầng phần cứng/tài nguyên ảo hóa riêng biệt: tính toán, lưu trữ, mạng. Vẫn chưa có giải pháp nào tích hợp các tài nguyên này dưới một phần mềm duy nhất.*

Lĩnh vực nghiên cứu lập trình vĩ mô (macro-programming) cho mạng cảm biến [38], [74], [42] và [6] cũng liên quan một phần tới đề tài nghiên cứu này. Ví dụ, các tác giả trong [38] đưa ra một giao diện lập trình giống SQL, trong đó toàn bộ một mạng kết nối được trừu tượng hóa (abstract) như là một bảng liên kết của cơ sở dữ liệu*. Cách tiếp cận này có một số hạn chế là khả năng mô hình hóa một mạng kết nối trong một cấu trúc bảng thường khó khăn phụ thuộc nhiều vào bản chất hoạt động và chức năng cho phép cấu hình của các thiết bị phần cứng trong môi trường thực tế.*

Trong [42], các tác giả đề cập tới việc cho phép thay đổi phạm vi của các mạng cảm biến (WSN). Nghiên cứu này chủ yếu tập trung vào vấn đề phân phối các công việc và trao đổi dữ liệu giữa các nút WSN nhằm tối ưu sự hỗn tạp trong phân bố của các nút cảm biến cũng như các kết nối giữa chúng. Trong [9] các tác giả đề xuất mô hình phân phối các nút lân cận tối ưu. Cách tiếp cận của họ là dựa vào các nút logic (là các mẫu), từ đó nhóm các nút có cùng tham số này lại với nhau. Để thuận tiện trong việc giao tiếp giữa các nút láng giềng này, nghiên cứu trên cũng cung cấp một cơ chế hiệu quả tìm đường (routing). Trong [6], các tác giả lại giới thiệu một khung lập trình bằng cách thống nhất giao diện API nhờ kĩ thuật trừu tượng hóa WSN nhằm cho phép các lập trình viên dễ dàng quản lý, điều khiển các nút trong WSN. Mặc dù các nghiên cứu liên quan tới WSN này đều hướng tới việc kết nối giữa chức năng cung cấp tài nguyên và tạo ra các nghiệp vụ logic (business logic), mục tiêu chính của chúng là tạo trực tiếp ra các ứng dụng hơn là việc tổng quát hóa mọi thứ như là mã lập trình (everything-as-code), từ đó cho phép quản trị thông tin tập trung và đưa ra các bài toán tối ưu hóa về truy vấn.

### Phân tích, đánh giá tình hình nghiên cứu và phát triển vạn vật kết nối Internet trong nước

Cũng như trên thế giới, IoT đang trở thành trào lưu công nghệ và khởi nghiệp tại Việt Nam. IoT như là một xu hướng tất yếu không khi các công nghệ phục vụ cho nó như điện toán đám mây, mạng, phần cứng, v.v… đã phát triển tới độ chín và có khả năng hội tụ ở một hệ thống phân tán rộng khắp. Hiện nay, một số lượng lớn các nhóm, các công ty tại Việt Nam đã nhìn thấy tiềm năng to lớn của mảnh đất công nghệ màu mỡ này. Họ đã đầu tư xây dựng các sản phẩm liên quan tới IoT như nhà thông minh, giải pháp ứng dụng công nghệ thông tin trong nông nghiệp (trồng trọt và chăn nuôi). Mặc dù vậy, phải nhìn nhận rằng sự phát triển của của IoT tại Việt Nam còn mang tính tự phát nhỏ lẻ của các đơn vị, tổ chức nhỏ, hiện nay chưa có định hướng rõ ràng và cụ thể từ phía các cơ quan nhà nước nhằm tạo lộ trình từng bước hướng tới xã hội thông minh. Các giải pháp, dự án liên quan tới IoT ở Việt nam hiện nay có thể chia thành một số nhóm tùy theo đầu ra của ứng dụng là gì, cụ thể: các giải pháp cho nông nghiệp, giao thông thông minh, nhà và môi trường sống thông minh giám sát biến đổi khí hậu.

Trong lĩnh vực nông nghiệp thông minh, tham vọng của công ty khởi nghiệp MimosaTek [40] xây dựng nền tảng nông nghiệp IoT giúp người nông dân tăng tối đa năng xuất và giảm tối thiểu rủi ro trong quá trình sản xuất nông nghiệp. Giải pháp chính của Mimosa là hệ thống quản lý tưới chính xác với các chức năng: giám sát gió, mưa, nhiệt độ, độ ẩm không khí, ánh sáng, độ ẩm đất; cảnh báo điều kiện môi trường bất lợi; cảnh báo hạn hán, lũ lụt; kiểm soát chính xác tưới bằng phần mềm bất cứ nơi nào; xây dựng hình ảnh nhật ký để theo dõi tình trạng thực vật hàng ngày; báo cáo thống kê điều kiện môi trường và các hoạt động tưới hàng ngày. Công ty Green Leap [21] cung cấp gói giải pháp cho người làm vườn bao gồm: tưới nước thông minh, theo dõi nhiệt độ và độ ẩm của đất, duy trì độ ẩm thích hợp trong tất cả các loại thời tiết; lập lịch để tự động tưới theo mong muốn của nông dân. Vào tháng 10 năm 2014, Fujisu hợp tác cùng tập đoàn FPT xây dựng nhà kính thí nghiệm trồng rau sạch áp dụng các giải pháp công nghệ thông tin tại Châu Quỳ Hà Nội với tên Fujisu – FPT Akisai Farm [16] sử dụng gói sản phẩm IoT của Fujisu. Vai trò của FPT là phối hợp với công ty Nhật trong việc đưa nông nghiệp thông minh vào thực tế sản xuất và học hỏi chuyển giao công nghệ này vào Việt nam. Có hai hoạt động của dự án, bao gồm: trồng trong nhà kính và xây dựng nhà máy rau sạch. Ở hoạt động thứ nhất, bằng cách sử dụng nhà kính Akisai và dịch vụ phần mềm lưu trú trên mây, người sản xuất sẽ được cung cấp thông tin về môi trường (nhiệt độ, độ ẩm, khí CO2, ánh sáng mặt trời, mưa, hướng gió và tốc độ) trong thời gian thực thông qua cac cảm biến được cài đặt trong nhà kính. Các dữ liệu này sau khi được xử lý sẽ giúp điều khiển các thiết bị như rèm, quạt gió, v.v… Bằng cách kết hợp với công nghệ trồng không cần đất, chỉ dùng nước dinh dưỡng (công nghệ này đang được phát triển mạnh mẽ ở Nhật), sản phẩm cà chua của nhà kính sẽ vừa có giá trị dinh dưỡng và hàm lượng đường cao. Với việc xây dựng nhà máy rau sạch, dự án sẽ cũng cấp quy trình hoàn toàn khép kín được phát triển bởi Fujisu. Nhà máy rau sạch sẽ dử dụng dữ liệu môi trường (nhiệt độ, độ ẩm, khí CO2, nhiệt độ chất lỏng dinh dưỡn thu thập từ các cảm biến, và đưa ra các lệnh điều khiển cho phép quá trình trồng trọt tối ưu nhất. Tháng 11năm 2014, ba đối tác là trung tâm Nghiên cứu và thiết kế vi mạch (ICDREC), khu Nông nghiệp công nghệ cao (AHTP) và công ty Mimosa Tek đã liên minh để triển khai mô hình ứng dụng công nghệ cao trên nền tảng IoT cho sản xuất nuôi tôm ở thôn Hoà Hiệp (Long Hoà, Cần Giờ, TP.HCM) [28]. Tháng 6 năm 2015, mô hình trên đã hoàn thiện phần cứng và phần mềm cũng như đội ngũ chuyên gia để vận hành mô hình IoT. Mô hình này sử dụng ba đầu cảm biến (sensor) chức năng: nhiệt độ nước, đo độ pH và nồng độ oxy trong nước. Những thay đổi của ba giá trị trên sẽ được các sensor ghi nhận, dữ liệu sẽ được truyền về các trạm thông tin (do ICDREC thiết kế), sau đó bằng kết nối không dây, dữ liệu chuyển về các server để các chuyên gia của AHTP tư vấn. Thông qua các thiết bị di động hoặc máy tính được cài đặt phần mềm (do Mimosa Tek thiết kế), nông dân có thể điều chỉnh hệ thống sục khí hoạt động hay ngưng, hoặc trực tiếp đến các vuông tôm để theo dõi. Tháng 7 năm 2015, tại khu Công nghệ cao TP.HCM (SHTP), ba đối tác: SHTP, hội Kỹ thuật chính xác Nhật Bản và viện Khoa học công nghệ Nhật Bản, cùng với ICDREC lên kế hoạch thành lập liên minh nghiên cứu thiết kế và chế tạo cảm biến sensor và ứng dụng cảm biến không dây và IoT trong nông nghiệp [69]. Trong liên minh này, SHTP sẽ nghiên cứu về sensor. ICDREC nghiên cứu về chip. Phía Nhật Bản sẽ đóng góp về công nghệ không dây (wireless) và các thiết bị điều khiển đầu cuối trên các ruộng.

Trong lĩnh vực giao thông thông minh, công ty Trí Nam [65] đang phát triển hệ thống tích hợp nhằm quản lý giao thông với nhiều thành phần khác nhau như hệ thống giám sát và quản lý cho các trạm thu phí, hệ thống quản lý giao thông với các camera giám sát. Mặc dù vậy, Trí Nam mới nhắm tới việc chỉ cung cấp phần mềm quản lý trong khí đó, việc thu thập, dữ liệu thông qua phần cứng và các cảm biến vẫn chưa được đề cập tới ở dự án này. Tại trường ĐH BKHN, trung tâm NAVIS đã thực hiện dự án hợp tác quốc tế “Nghiên cứu và phát triển hệ thống định vị thời gian thực với độ chính xác tới centimet” [51]. Mục tiêu của dự án là phát triển một hệ thống hoàn chỉnh chỉ ra chính xác vị trí tới centimet. Hệ thống bao gồm các thành phần: thu, hệ thống NTRIP cho truyền dữ liệu hiệu chỉnh. Kết quả của nghiên cứu này là cần thiết cho các ứng dụng đòi hỏi định vị với độ chính xác cao trong thời gian thực. Các kết quả nghiên cứu được công bố ở đây [61] và [66]. Đây là dự án nghiên cứu sẽ ứng dụng tốt trong môi trường giao thông thông tin với các thiết bị cảm biến, các thành phần ra quyết định, trợ giúp lái xe, v.v… Magiwan [17] là ví dụ sản phẩm thông minh khác cho bài toán giao thông tại Việt nam. Giải pháp chính của Magiwan là giám sát xe và hàn hóa sử dụng sóng di động (GSM hoặc 3G), hệ thống định vị (GPS), RFID, các cảm biến cho hàng hóa (ví dụ như cảm biến khí gas, dầu và bê tông), và giám sát bằng camera IP. Điểm sáng của Magiwan đó là mỗi giải phải của công ty được bán hoàn chỉnh trong một hộp thiết bị đóng kín, và điều duy nhất mà khách hàng cần phải làm đó là cắm hộp thiết bị giám sát vào xe của mình. Thêm vào đó, giải pháp Magiwan cũng cung cấp cho người sử dụng ứng dụng hiển thị thông tin trên thiết bị di động và máy tính, cho phép họ dễ dàng giám sát, theo dõi hàng hóa và quản lý các khâu vận tải. Hiện tại, trong quảng cáo của công ty, điện toán đám mây chưa dược sử dụng vào quá trình xử lý dữ liệu hoặc chứa các ứng dụng với vai trò máy chủ backend. Ngoài Magiwan, một số các giải pháp khác cùng lĩnh vực giao thông thông minh đã được phát triển thành sản phẩm như V-tracking (phát triển bởi Viettel) [18], Binh Anh (phát triển bởi công ty điện tử Bình Anh) [19], Vcomsat (phát triển bởi công ty Vcomsat) [20]. Ứng dụng Smart BK Traffic [57] được phát triển bởi một nhóm nghiên cứu đứng đầu là tiến sĩ Phạm Trần Vũ, trường đại học Bách Khoa TpHCM. Nhóm nghiên cứu đã sử dụng thiết bị GPS được cài đặt trên các xe buýt công cộng để tính toán các khả năng tắc nghẽn giao thông, hiển thị lên Google Maps, và từ đó giúp quản lý giao thông hiệu quả.

Trong lĩnh vực nhà thông minh, một loạt các dự án đã được sản phẩm hóa từ rất sớm, tiêu biểu nhất là giải pháp Smart Home của BKAV [58]. Giải pháp này kết nối tất cả các thiêt bị trong nhà với nhau thông qua mạng cho phép điều khiển, sử dụng các vật dụng trong nhà trong ngữ cảnh thông minh. Các thiết bị bao gồm hệ thống chiếu sáng, rèm cửa, máy lạnh, truyền hình, hệ thống âm thanh, máy nước nóng, thông gió, máy hút ẩm, các cảm biến môi trường, chuông cửa có video, camera an ninh, hàng rào điện tử, báo động khí rò rỉ, báo khói, hệ thống tưới vườn, v.v… Thông thường với một hệ thống nhà thông minh, người dùng sẽ cần hàng chục hoặc thậm chí lên đến hàng trăm thiết bị chuyển mạnh, BKAV cung cấp giải pháp tích hợp cho phép điều khiển các thiết bị này bằng một vài màn hình cảm ứng đơn giản hoặc thậm chí trên các thiết bị di động như máy tính bảng hoặc điện thoại thông minh với giao diện trực quan 3D (các thiết bị được mô phỏng như thực tế). Với các công nghệ truyền thông tiên tiến như Zigbee (công nghệ truyền dữ liệu không dây), PLC (Power Line Communication), hệ thống SmartHome có thể được triển khai trong nhà ở hiện có hoặc mới. Do nhu cầu và nhu cầu thay đổi số lượng thiết bị trong quá trình sử dụng SmartHome phát sinh là không thể tránh khỏi. Vì vậy, thiết kế không dây rất thuận lợi cho việc mở rộng và biến đổi của nhu cầu người dùng. Tương tự như BKAV, giải pháp Lumi [36] và Acis [1] cũng cung cấp các gói sản phẩm nhà thông minh. Các thiết bị được kết nối trong nhà bao gồm: công tắc cảm ứng, cảm biến ánh sáng, rèm cửa điều khiển, cảm biến chuyển động, điều khiển hồng ngoại, cảm biến mở cửa, cảm biến môi trường, điều khiển tập trung, v.v… Tất cả các thiết bị này được tích hợp công nghệ Zigbee.

Bên cạnh hướng đi tập trung vào sản phẩm ứng dụng IoT, một số công ty phát triển hệ sinh thái IoT của riêng mình bằng cách tạo ra nền tảng mở IoT cung cấp cho cộng đồng. Ubisen [68] là một trong những giải pháp tiên phong trong việc xây dựng nền tảng IoT. Với Ubisen người dùng có thể xây dựng gói dịch vụ IoT cho bất kỳ hệ thống thông minh nào. Giải pháp IoT này sẽ cho phép tiết kiệm thời gian trong việc phát triển hệ thống tự động phục vụ các kế hoạch kinh doanh khác nhau như theo yêu cầu sản xuất của doanh nghiệp và cá nhân. Cốt lõi của nền tảng Ubisen là một tập hợp các thành phần gồm phần mềm, dịch vụ đám mây, dịch vụ lưu trữ dữ liệu cho phép kết nối các thiết bị điện tử xung quanh chúng ta để tạo ra một môi trường IoT theo trên nhu cầu của người sử dụng. Konexy [32] là một công ty khởi nghiệp với sản phẩm nền tảng IoT như Ubisen. Gói sản phẩm của Konexy được cung cấp theo các dạng khác nhau. Việc quản lý các thiết bị và dữ liệu được cấu trúc dạng cây như hầu hết các nền tảng IoT khác trên thế giới. Điều này giúp người dùng, nhà lập trình dễ dàng truy cập và điều khiển các thiết bị. Mặc dù công ty công bố giải pháp dựa trên nền tảng đám mây, tuy nhiên trong thực tế dữ liệu lưu trữ trong hệ thống máy chủ vật lý. Đầu năm 2016, công ty DTT cũng công bố kế hoạch xây dựng nền tảng IoT mở - Open IoT platform [45]. Mục tiêu của dự án này là cung cấp một nền tảng IoT cho hệ sinh thái tảng sáng tạo mã nguồn mở. Dự án hứa hẹn cung cấp một môi trường cho các nhà lập trình phát triển các ứng dụng IoT của mình.

Tại trường đại học Tân Tạo, Long An, TS Cao Tiến Dũng và nhóm của mình hiện cũng đang tập trung nghiên cứu về IoT. Một số các kết quả đã được công bố liên quan tới dữ liệu cảm biến về con người người theo thời gian thực [62] và [71]. Nhóm nghiên cứu của TS. Ngô Quỳnh Thu, TS. Trương Diệu Linh ĐHBK HN tập trung nghiên cứu về truyền dẫn dữ liệu thông qua mạng không dây LPWan, ZigBee. TS. Trần Hải Anh đã phát triển được hệ thống đánh giá tình trạng phổi thông qua khí thở cài đặt trên các thiết bị di động [4]. Đây là ứng dụng phục vụ chăm sóc sức khỏe cộng đồng và có khả năng tích hợp vào hệ sinh thái IoT trong tương lai gần. Nhóm nghiên cứu từ viện Điện tử viễn thông (TS. Trương Thu Hương, PGS. Nguyễn Hữu Thanh, …) tập trung nghiên cứu vào các bài toán truyền dẫn không dây, các vấn đề về tiết kiệm năng lượng trong truyền dẫn không dây ứng dụng cho giao thông thông minh và cho kết nối các trung tâm dữ liệu. Hiện nay chính phủ và các Bộ, ngành cũng có một số các chính sách hoặc thành lập các chương trình nghiên cứu hướng tới sử dụng IoT. Ví dụ như chương trình Biến đổi khí hậu của Bộ tài nguyên và môi trường [8], khuyến khích và cấp kinh phí cho các nghiên cứu giám sát biến đổi khí hâu, hiện tượng xâm mặn ở đồng bằng song Cửu Long, sự thay đổi của diện tích rừng, mở rộng của thành phố, v.v…

Một số các kĩ sư công nghệ quan tâm tới IoT tại Hà Nội và TPHCM đã kết hợp lại với nhau thành các nhóm cộng đồng nghiên cứu về IoT. Tiêu biểu như nhóm Hackanoi [23] tập trung vào nghiên cứu phát triển máy in 3D, cài đặt và thử nghiệm các thiết bị kết nối và điều khiển các cảm biến, trong đó chủ yếu là Raspberry Pi, AdruiKhông, Mega, v.v… Nhóm IoT Alliance [27] chuyên tư vấn các giải phải IoT sử dụng công nghệ mã nguồn mở (ví dụ nền tảng IoT Kaa) hoặc các phần cứng có sẵn (Arduino, Rasdberry Pi) trong các bài toán cụ thể như nông nghiệp, nhà thông minh.

## Tính cấp thiết của đề tài

IoT là một lĩnh vực rất rộng kết hợp nhiều công nghệ khác nhau cho phép kết nối một số lượng lớn các vật (things) để truyền và nhận dữ liệu nhằm mục đích phối hợp cùng hoạt động trong một ngữ cảnh nào đó. Mô hình IoT được mô tả trong các bài báo như [2], [39], [35] và [4] bao gồm có 3 tầng chính gồm có: tầng các thiết bị cảm biến, tầng truyền dẫn (hay còn gọi là điện toán sương mù hoặc điện toán biên), và dịch vụ điện toán đám mây. Kiến trúc của 3 tầng được trình bày ở Hình 1.



Hình 1: Kiến trúc 3 tầng của IoT

Tầng cao nhất là tầng cảm biến, thông thường số lượng các cảm biến trong IoT là rất lớn. Nhiệm vụ chủ yếu của các cảm biến này là cảm thụ, giám sát môi trường sau đó truyền dữ liệu thu thập được về phía các tầng thấp hơn như điện toán sương mù (điện toán biên) và điện toán đám mây. Tầng cảm biến đòi hỏi tích hợp các công nghệ về phần cứng, thiết kế mạch, thiết kế cảm biến, lập trình nhúng. Các hướng nghiên cứu ở tầng này chủ yếu đảm bảo khả năng truyền dẫn tối ưu, tiết kiệm năng lượng, chất lượng cảm thụ, giám sát sự vật, sự kiện, chất lượng phần cứng (độ bền) trong môi trường thực tế.

Tầng thứ hai, thường được gọi là điện toán sương mù, tầng này bao gồm các thiết bị truyền dẫn dữ liệu như router, switch, các hệ thống có thể lập trình nhúng, các hệ thống máy tính cá nhân và thiết bị di động. IBM gọi mô hình tính toán này là điện toán biên, ám chỉ nơi xa nhất có thể có kết nối Internet và là nơi thế giới thực của hàng tỉ vật bắt đầu. Các trung tâm dữ liệu tập trung (data center) khi đó chỉ nằm ở “trung tâm” của mạng lưới, trong khi các máy tính cá nhân, điện thoại, camera giám sát, … sẽ nằm ở ngoài rìa. Điện toán biên hay sương mù sẽ thực hiện các chức năng chính là:

* Truyền dẫn dữ liệu từ tầng cảm biến về tầng các dịch vụ đám mây;
* Điều hướng tối ưu trong việc truyền tải dữ liệu;
* Tiền xử lý dữ liệu trong trường hợp các phân tích, ra quyết định nhanh không cần tới khả năng tính toán mạnh mẽ của công nghệ đám mây.

*Tuy nhiên, có thể dễ dàng thấy rằng, việc phát triển rất nhiều các nền tảng IoT, sử dụng các công nghệ giao tiếp khác nhau và việc ứng dụng các nền tảng, công cụ này vào các bài toán cụ thể riêng biệt sẽ ảnh hưởng tới khả năng mở rộng phạm vi kết nối các vật hay thiết bị IoT trong thực tế.* Ví dụ cụ thể, một tòa nhà thông minh có thể dùng OpenM2M như một công cụ để quản lý, điều khiển và kết nối các thiết bị trong tòa nhà đó, tuy nhiên, tòa nhà thứ hai bên cạnh đó lại dùng nền tảng khác như Kura hay ThingSpeak để thực hiện tác vụ tương tự. Vấn đề đặt ra ở đây là khi kết hợp hai tòa nhà với các thiết bị thông minh này thành một giải pháp IoT tổng thể cho phép quản lý, kết nối và điều khiển chúng trong một ngữ cảnh chung (như khu chung cư thông minh), công việc đầu tiên cần nhất là phải liên kết được các nền tảng này đã được triển khai này lại với nhau. Việc mở rộng phạm vi lớn hơn như trình bày ở trên sẽ dẫn tới nhu cầu phải có một công cụ cho phép nhiều nền tảng IoT với các công nghệ khác nhau nói chuyện được với nhau. Từ đó mô hình IoT mới đạt được khả năng kết nối các thiệt bị hay vật một cách rộng khắp như định nghĩa của nó.

Tầng thấp nhất là điện toán mây. Tầng này bao gồm các hệ thống máy chủ được triển khai công nghệ ảo hóa nhằm giúp tối ưu hiệu quả sử dụng tài nguyên và phục vụ các tác vụ xử lý dữ liệu tập trung và cần hiệu năng cao. Thông thường dữ liệu được tạo ra bởi 1 số lượng khổng lồ các cảm biến sẽ được truyền về thông qua điện toán sương mù (bao gồm các thiết bị mạng, truyền dẫn) để tới đám mây nhằm lưu trữ và xử lý. Hiện nay số lượng các nhà cung cấp dịch vụ điện toán đám mây công cộng là khá lớn, tuy nhiên trong số đó nổi lên một vài tên tuổi lớn như Amazon, ElasticHosts, Rackspace, v.v… Bên cạnh đó, một số không nhỏ các đám mây riêng được các tổ chức, doanh nghiệp tự triển khai thông qua sử dụng các đám mây mã nguồn mở (OpenStack, CloudStack) bên trong phạm vi sử dụng của họ nhằm tận dụng tốt ưu điểm của điện toán đám mây nhưng vẫn đảm bảo tính bảo mật.

Trong môi trường IoT, rõ ràng rằng, với sự mở rộng quy mô và phạm vi các vật kết nối mạng theo chiều ngang (horizontal) là rất lớn, cách thức sử dụng các đám mây cũng sẽ rất phức tạp bao gồm có thể cùng một lúc sử dụng nhiều loại đám mây khác nhau (riêng, công cộng) và thuộc nhiều nhà cung cấp.

Nếu không có các giải pháp đa đám mây, hoặc nói một cách khác cho phép khả năng tương tác giữa các đám mây, việc mở rộng kết nối các thiết bị hoặc vật trong khái niệm IoT sẽ phụ thuộc vào một hoặc một vài đám mây nào đó. Nhưng quan trọng hơn nữa giữa các đám mây này, dữ liệu, ứng dụng sẽ không có khả năng trao đổi, di chuyển, và kết hợp xử lý với nhau để ra những quyết định phù hợp với ngữ cảnh đòi hỏi trí thông minh nhân tạo của con người. Chính vì vậy, yêu cầu đặt ra là phải có một giải pháp hoặc công cụ cho phép sử dụng (bao gồm các chức năng: quản lý, điều khiển tài nguyên, phát triển và triển khai ứng dụng) đa đám mây. Đây là một nhu cầu tất yếu, đặc biệt trong thời đại IoT khi đứng ở góc nhìn ở tầng dịch vụ đám mây trong kiến trúc chung của IoT. Theo hướng này, dữ liệu được truyền từ các tầng điện toán sương mù (biên) hoặc tầng cảm biến sẽ được cung cấp một giao diện hoặc cách thức để lưu trữ, xử lý trên các đám mây khác nhau tùy thuộc vào nhu cầu của người lập trình và sử dụng. Các chuyên gia lập trình cũng sẽ có công cụ để phát triển và triển khai ứng dụng IoT trên nhiều đám mây một cách dễ dàng (phát triển một lần, triển khai và sử dụng trên bất cứ đám mây nào).

Qua các phân tích ở trên, có thể thấy nổi lên vấn đề ở tầng điện toán sương mù cần giải quyết để đảm bảo sự phát triển và mở rộng cho IoT: Cần một giải pháp liên kết các nền tảng IoT có sẵn hiện, từ đó cho phép chúng có thể giao tiếp với nhau, và nhà lập trình, quản trị có thể điều khiển các thiết bị ở phạm vi rộng không bó hẹp trong một không gian quản lý của một nền tảng IoT đã được triển khai. Công cụ này cũng sẽ hỗ trợ việc truyền dữ liệu từ tầng điện toán sương mù về tầng điện toán đám mây một cách dễ dàng thông qua một giao diện chung thống nhất.

Trong đề tài này, tôi tập trung giải quyết vấn đề ở trên bằng cách xây dựng khung (framework) lập trình theo hướng tiếp cận trừu tượng hóa (abstraction) nhằm cung cấp các giải pháp liên kết các nền tảng IoT khác. Bên cạnh đó, chúng tôi cũng xây dựng mô hình dữ liệu cho phép tích hợp dữ liệu trong mô trường IoT phục vụ cho việc quản lý và điều khiển các thiết bị một cách hiệu quả.

# CHƯƠNG 2: NỀN TẢNG CÔNG NGHỆ

### Khái niệm về các nền tảng phần mềm IoT

Các nền tảng IoT tập trung chủ yếu vào việc cung cấp một kiến trúc và giao diện chung, thống nhất cho phép quản lý các thiết bị (devices) các vật (things) trong môi trường IoT, từ đó người dùng có thể lập trình các ứng dụng điều khiển thiết bị, vật một cách dễ dàng. Tuy nhiên hiện nay, số lượng nền tảng IoT có sẵn trên thế giới rất nhiều [41] và [2] và mỗi một nhóm người dùng lại có xu hướng dùng một nền tảng IoT khác nhau. Bên cạnh đó, công nghệ được sử dụng trong các nền tảng này cũng khá đa dạng, đòi hỏi phải có nghiên cứu, phân tích chi tiết nhằm đưa ra được một số điểm chung hoặc công nghệ được coi là chuẩn không chính thức của IoT hiện nay. Nhìn chung, một nền tảng phần mềm (platform) IoT thường bao gồm các thành phần (chức năng) sau:

* **Định danh** để ánh xạ tên, địa chỉ và các đối tượng trong môi trường cụ thể nào đó. Các đối tượng có thể là các cảm biến, các trình điều khiển (actuators), các gateway, các thiết bị di động, các máy tính, máy chủ, dịch đám mây … cho phép thu thập, trao đổi, chia sẻ các dữ liệu và xử lý thông tin. Định danh của các đối tượng tượng thường được sử dụng để tìm kiếm trong khi địa chỉ của đối tượng sử dụng để truy cập đến đối tượng trên mạng.
* **Cảm biến** (sensors) là các thiết bị để thu thập dữ liệu và gửi về các máy chủ, các đơn vị lưu trữ dữ liệu (có thể trên các dịch vụ đám mây) .
* **Hạ tầng kết nối** đảm bảo kết nối các thiết bị với nhau và các thiết bị với các thành phần khác của nền tảng IoT. Một đặc điểm quan trọng trong IoT là kết nối vạn vật (không đồng nhất) với nhau dựa trên các giao thức truyền thông đa dạng. Có thể kể đến các phương thức kết nối hay được sử dụng trong các nền tảng IoT như *http/https*, *MQTT*, *COAP*, …
* **Tài nguyên tính toán** gồm các bộ xử lý, các thiết bị lưu trữ và các phần mềm quản trị các tài nguyên tính toán, lưu trữ, mạng, … cho phép biểu diễn và lưu trữ dữ liệu tạm thời và xử lý dữ liệu, phân bố và cân bằng tải, các kết nối, … Các thiết bị hay được sử dụng trong các ứng dụng IoT có thể kể đến như Arduino, Intel Galileo, Raspberry Pi,… Nền tảng tính toán đảm mây (Cloud platform) cũng là một phần quan trọng trong IoT. Nền tảng tính toán đám mây cho phép gửi/nhận dữ liệu thu thập từ các đối tượng, cung cấp các xử lý tính toán theo thời gian thực và từ đó có thể trích rút tri thức, biểu diễn ngữ nghĩa, phân tích dữ liệu lớn.
* **Các dịch vụ** cung cấp các cơ chế quản lý các thiết bị (định danh và định vị), mô hình hóa và biểu diễn dữ liệu, quản trị các sự kiến, tính toán, phân tích dữ liệu, các dịch vụ quản trị và xử lý dữ liệu. Các dịch vụ có thể chia làm 4 loại:
  + Dịch vụ định danh (identity-related dịch vụs).
  + Dịch vụ tính toán tổng hợp thông tin (information aggregation dịch vụs).
  + Dịch vụ thực hiện các nhiệm vụ cộng tác (collaboratives-aware dịch vụs).
  + Dịch vụ tính toán khắp nơi (ubiquitous dịch vụs).
* **Ngữ nghĩa (semantic)** trong nền tảng IoT cho phép trích rút tri thức từ khối lượng lớn dữ liệu thu thập phục vụ các xử lý cần thiết. Trích rút tri thức bao gôm phát hiện tri thức từ các nguồn, biểu diễn tri thức và tích hợp các thông tin, tri thức thu thập được.



Hình 2: Kiến trúc điển hình của nền tảng IoT

Hình 2 [1] miêu tả các thành phần chính của một nền tảng IoT nói chung.

### Đánh giá một số nền tảng IoT

Để triển khai kịch bản thử nghiệm, nhóm nghiên cứu lựa chọn ba nền tảng IoT bao gồm: OpenHab, Home Assistant và ThingBroads để quản lý các thiết bị. Lý do lựa chọn các nền tảng này bao gồm:

* Đơn giản trong quá trình triển khai thử nghiệm.
* Có khả năng triển khai, tích hợp với các thiết bị.
* Mã nguồn mở.
* Hỗ trợ hầu hết các công nghệ, giao thức truyền thông tin hiện nay của IoT như MQTT, HTTP, CoAP.
* Được sử dụng khá phổ biến (theo nghiên cứu, đánh giá).

**OpenHAB**

OpenHAB [47] viết tắt của Open Home Automation Bus. Đây là dự án nhằm cung cấp nền tảng tích hợp cho mọi vật xung quanh nhà tự động được viết bằng Java dựa trên framework OSGi. OpenHAB cho phép nhà lập trình có thể thu thập dữ liệu từ các loại cảm biến hoặc thiết bị khác nhau trong nhà và lập trình tạo ra các quyết định dựa vào các trạng thái của từng thiết bị hoặc cả hệ thống. Về mặt kiến trúc, OpenHAB sử dụng cách giao tiếp publish/subcribe trên một bus sự kiện chung. Điều này giúp giảm năng lượng cũng như cài đặt các bộ thu nhận dữ liệu phức tạp.

Các thiết bị trong hệ thống được OpenHAB coi là các “item”, thực chất là các đối tượng lập trình (object) để điều khiển. Các thiết bị này được cấu hình và quản lý theo cấp bậc (hierarchical). Một nhóm các đối tượng thiết bị được gọi là “Group”. Group có thể nằm trong một Group khác. Các Group có thể có nhiều items hoặc không có items nào. Dữ liệu thu thập và trạng thái của các thiết bị được được cấu hình trong tệp “config” để lưu trữ.

Nền tảng OpenHAB có thể được triển khai trên các nền tảng khác nhau hỗ trợ máy ảo Java (JVM) như Linux, Mac, Windows). Do vậy, OpenHAB khá độc lập so với phần cứng. Nền tảng này hỗ trợ RESTful API và một giao diện web cho phép điều khiển các thiết bị thông minh, kết nối với các hệ thống khác dễ dàng.

**Home Assistant**

Home Assistant [25] là một nền tảng mã nguồn mở có thể tải xuống và sử dụng miễn phí. Nền tảng này công bố phiên bản đầu tiên vào năm 2012. Các thiết bị được Home Assistant hỗ trợ kết nối chủ yếu thông qua giao thức MQTT. YAML là định dạng tệp cấu hình của nền tảng này. Về cơ bản công việc cấu hình thông qua YAML khá đơn giản và nhà lập trình ứng dụng có khả năng nhanh chóng nắm bắt được. Cũng giống như các nền tảng IoT khác, Home Assistant cũng cung cấp các cơ chế xác thực, tạo các sự kiện hành động (trigger), định nghĩa các luật hoạt động cho một tập các thiết bị và lưu các mẫu kịch bản này cho lần thực thi tiếp theo. Giao diện của Home Assistant trong việc quản lý thiết bị khá đơn giản. Nền tảng này có thể được triển khai dễ dàng trên hầu hết các phiên bản hệ điều hành Linux. Chính vì vậy, nó có khả năng triển khai trên các gateway một cách dễ dàng. Các dịch vụ có thể cài đặt bổ sung bao gồm: thống kê, các bộ kết nối với các cơ sở dữ liệu khác nhau, triển khai trên các hệ điều hành của thiết bị di động (iOS và Adroid).

**ThingBroads**

ThingsBoard [64] là một nền tảng IoT nguồn mở cung cấp các chức năng thu thập, xử lý, trực quan hóa dữ liệu và quản lý thiết bị. Nền tảng này cho phép kết nối thiết bị thông qua các giao thức IoT tiêu chuẩn công nghiệp như MQTT, CoAP và HTTP. Nền tảng IoT này cũng hỗ trợ việc triển khai một cách dễ dàng lên trên các hạ tầng dịch vụ đám mây. Các tính năng chính của ThingsBroad gồm:

* Thu thập dữ liệu từ xa: Thingsboard hỗ trợ nhà lập trình ứng dụng thu thập và lưu trữ dữ liệu từ xa theo cách đáng tin cậy. Các nhà lập trình có thể truy cập dữ liệu đã thu thập bằng cách sử dụng giao diện web tùy chỉnh hoặc API.
* Hiển thị trực quan dữ liệu đã thu thập: Thingsboard cung cấp hơn 30 tiện ích có sẵn phục vụ hiển thị trực quan các dữ liệu thu thập được.
* Công cụ tạo chuỗi quy tắc kéo thả thân thiện: cho phép xử lý dữ liệu bằng chuỗi quy tắc linh hoạt dựa trên thuộc tính của thực thể hoặc nội dung thông điệp.
* Quản lý thiết bị:Thingsboard cung cấp khả năng đăng ký và quản lý thiết bị thông qua chức năng theo dõi các trạng thái của thiết bị và cung cấp API để điều khiển ác thiết bị đó.
* Quản lý các báo động: Thingsboard cung cấp khả năng tạo và quản lý các cảnh báo liên quan đến các thực thể.

Ngoài các nền tảng IoT kể trên, hiện nay còn rất nhiều các nền tảng IoT khác. Mỗi nền tảng lại có một đặc tính kĩ thuật riêng. Chúng tôi cũng xem xét liệt kê, so sánh và đánh giá một số nền tảng IoT tiêu biểu hiện nay thông qua Bảng 1.

Bảng 1: So sánh một số các nền tảng IoT hiện nay

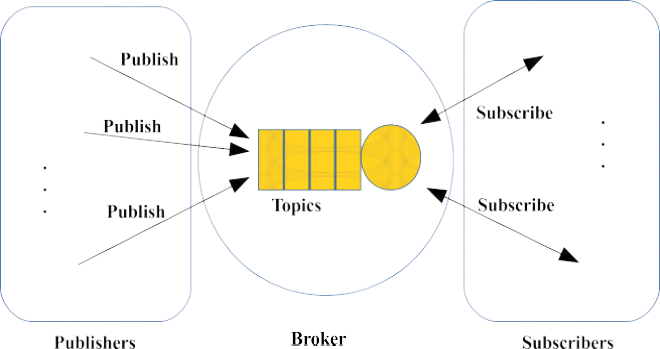
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TT** | **Nền tảng** | **Hỗ trợ thiết bị** | **Kiến trúc** | **Mã nguồn mở** | **REST** | **Mô hình truy cập dữ liệu** | **Dịch vụ tìm kiếm** |
| [1](#_bookmark29) | TM  AirVantage | Cần gateway | Triển khai được trên mây | Libraries only (Apache v2, MIT and Eclipse v1.0 | Có | OAuth2 | Không |
| [2](#_bookmark33) | Arkessa | Có | Triển khai được trên mây | Không | Chưa rõ. | Giống Facebook | Không |
| [3](#_bookmark34) | ARM mbed | Thiết bị nhúng | Tập trung/ Triển khai được trên mây | Không | CoAP | Ngưởi sử dụng chọn | Không |
| [4](#_bookmark27) | Carriots | Có | Triển khai được trên mây | Không | Có | Truy cập an toàn ( | Không |
| [5](#_bookmark30) | DeviceCloud | Có | Triển khai được trên mây | Không | Có | Chưa rõ. | Không |
| [7](#_bookmark31) | EveryAware | Có | Tập trung | Không | Có | 4 mức | Không |
| [8](#_bookmark32) | Everyware | Cần gateway | Triển khai được trên mây | Không | Có | Chưa rõ. | Không |
| [9](#_bookmark36) | EvryThng | Có | Tập trung | Không | Có | Có | Không |
| [10](#_bookmark37) | Exosite | Có | Triển khai được trên mây | BSD license | Có | Chưa rõ. | Không |
| [11](#_bookmark38) | Fosstrack | RFID | Tập trung | Không | Không | Lưu cục bộ | Không |
| [12](#_bookmark39) | GroveStreams | Không | Triển khai được trên mây | Không | Có | Tùy vai trò | Không |
| [13](#_bookmark42) | H.A.T. | Thiết bị trong nhà | Không tập trung | Có | Có | Lưu cục bộ | Có |
| [14](#_bookmark43) | IoT-framework | Có | Tập trung | Apache license 2.0 | Có | Lưu cục bộ | Có |
| [15](#_bookmark44) | IFTTT | Có | Tập trung | Không | Không | Không storage | Limited |
| [16](#_bookmark40) | Kahvihub | Có | Tập trung | Apache license 2.0 | Có | Lưu cục bộ | Có |
| [17](#_bookmark41) | LinkSmart | Thiết bị nhúng | Không tập trung | LGPLv3 | Không | Lưu cục bộ | Có |
| [18](#_bookmark45) | MyRobots | Robots | Triển khai được trên mây | Không | Có | 2 mức | Không |
| [19](#_bookmark46) | Niagara | Có | Distributed | Không | Chưa rõ. | Chưa rõ. | Chưa rõ. |
| [20](#_bookmark47) | Nimbits | Có | Tập trung/ Triển khai được trên mây | Apache license 2.0 | Có | 3 mức | Không |
| [21](#_bookmark48) | NinjaPlatform | Cần gateway | Triển khai được trên mây | Có | Có | OAuth2 | Không |
| [22](#_bookmark50) | Khôngde-RED | Có | Tập trung | Apache license 2.0 | Không | Tùy quyền của người dùng | Không |
| [23](#_bookmark51) | OpenIoT | Có | Không tập trung | LGPLv3 | Không | Tùy quyền của người dùng | Có |
| [24](#_bookmark52) | OpenMTC | Có | Tập trung/ Triển khai được trên mây | Không | Có | Truy cập an toàn | Không |
| [25](#_bookmark53) | OpenRemote | Thiết bị trong nhà | Tập trung | Affero GNU Public License | Có | Lưu cục bộ | Không |
| [26](#_bookmark54) | Open.Sen.se | Kết nối LAN | Triển khai được trên mây | Không | Có | 2 mức | Giới hạn |
| [27](#_bookmark55) | realTime.io | Cần gateway | Triển khai được trên mây | Không | Có | Truy cập an toàn | Không |
| [28](#_bookmark49) | TM  SensorCloud | Không | Triển khai được trên mây | Không | Có | Chưa rõ. | Không |
| [29](#_bookmark57) | SkySpark | Không | Tập trung/ Triển khai được trên mây | Không | Có | Chưa rõ. | Không |
| [30](#_bookmark58) | Swarm | Có | Triển khai được trên mây | Client là mã nguồn mở | Có | Chưa rõ. | Chưa rõ. |
| [31](#_bookmark59) | TempoDB | Không | Triển khai được trên mây | Không | Có | Truy cập an toàn | Không |
| [32](#_bookmark60) | TerraSwarm | Có | Không tập trung | Chưa rõ. | Chưa rõ. | Chưa rõ. | Có |
| [33](#_bookmark61) | The thing sys- tem | Thiết bị trong nhà | Tập trung | M.I.T. | Có | Người dùng chọn | Không |
| [34](#_bookmark62) | Thing Broker | Có | Tập trung | Có | Có | Lưu cục bộ | Không |
| [35](#_bookmark63) | ThingSpeak | Có | Tập trung/ Triển khai được trên mây | GNU GPLv3 | Có | 2 mức | Giới hạn |
| [36](#_bookmark64) | ThingSquare | Thiết bị nhúng | Triển khai được trên mây | Gateway firmware là mã nguồn mở | Có | Không | Không |
| [37](#_bookmark65) | ThingWorx | Có | Triển khai được trên mây | Không | Có | Tùy quyền của người dùng | Có |
| [38](#_bookmark66) | WoTkit | Có | Triển khai được trên mây | Không | Có | Truy cập an toàn | Có |
| [39](#_bookmark68) | Xively | Có | Triển khai được trên mây | Libraries là mã nguồn mở (BSD 3- clause), platform không | Có | Truy cập an toàn | Có |

### Công nghệ MQTT hàng đợi thông điệp cho IoT

MQTT viết tắt của Message Queue Telemetry Transport được phát triển bới IBM và EuroTech năm 1999 và trở thành chuẩn của OASIS năm 2013 [11]. Mục tiêu của MQTT là kết nối các thiết bị nhúng và mạng với ứng dụng. Hoạt động kết nối sử dụng cơ chế tìm đường (một-với-một, một-tới-nhiều, và nhiều-tới-nhiều) cho phép MQTT được sử dụng rộng rãi trong môi trường IoT và M2M.

MQTT dựa vào cơ chế publish/subcribe để cung cấp cách thức giao tiếp mềm dẻo, đơn giản. Giao thức này được viết dựa tren TCP nên đảm bảo được chất lượng truyền, gửi thông điệp tốt. Có hai phiên bản MQTT: MQTT 3.1 và MQTT-SN [67] V1.2. Phiên bản thứ hai định nghĩa các thông số phù hợp cho mạng cảm biến với ba thành phần: giao tiếp ngữ nghĩa (connection semantics), tìm đường (routing) và endpoint. Ngoài thành phần publish và subcribe, MQTT còn có thành phần broker. Thực chất đây là các hàng đợi trong quá trình nhận gửi dữ liệu của publish và subcribe. Bên cạnh đó, broker còn có vai trò đảm bảo toàn dữ liệu bằng cách kiểm tra thông tin xác thực giữa hai thành phần còn lại. Một số lượng lớn các ứng dụng hiện này sử dụng MQTT như ứng dụng chăm sóc sức khỏe, giám sát tài nguyên, đồng hồ đo điện, cảnh báo (notification) của Facebook.

Thông điệp của MQTT sử dụng hai bytes đầu tiên làm phần header. Tiếp sau đó, các giá trị có thể nhận trong phần MESSAGE TYPE là: CONNECT (1 byte), CONNACK (1 byte), PUBLISH (3 bytes) và SUBCRIBE (8 bytes). Ngoài ra, còn có phần DUP chỉ ra dữ liệu có thể trùng lặp với lần gửi trước đó. Một số các phần lựa chọn khác trong định dạng của thông điệp MQTT có thể là QoS level (theo ba mức QoS) và Retain (phần thân thông điệp).



Hình 3: Kiến trúc của MQTT [2]

Như phân tích ở trên, có thể thấy rằng, các thiết bị IoT hiện nay sử dụng một số các giao thức để giao tiếp khác nhau. Nôi lên trong số đó là CoAP, MQTT, XMPP, AMQP, v.v… Các giao thức này quyết định cách thức hoạt động của các nền tảng IoT. Do đó, trong khuôn khổ của đề tài, nghiên cứu các giao thức giao tiếp này là yếu tố quan trọng trong việc xây dựng một công cụ cho phép quản lý, điều khiển các thiết bị IoT thông qua các nền tảng khác nhau. Bảng 2 [2] sau so sánh sơ lược các giao thức giao tiếp ứng dụng của IoT.

Bảng 2: So sánh các giao thức giao tiếp của IoT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Giao thức** | **RESTful** | **Transport** | **Publish/subcribe** | **Request/Response** | **Security** | **QoS** | **Kích thước Header (Bytes)** |
| **CoAP** | Có | UDP | Có | Có | DTLS | Có | 4 |
| **MQTT** | Không | TCP | Có | Không | SSL | Có | 2 |
| **MQTT-SN** | Không | TCP | Có | Không | SSL | Có | 2 |
| **XMPP** | Không | TCP | Có | Có | SSL | Không | - |
| **AMQP** | Không | TCP | Có | Không | SSL | Có | 8 |
| **DDS** | Không | TCP | Có | Không | SSL | Có | - |
| UDP | Có | DTLS |
| **HTTT** | Có | TCP | Không | Có | SSL | Không | - |

Có thể thấy rằng số lượng nền tảng IoT là rất lớn. Tuy nhiên chưa có một nhà cung cấp nào nổi lên đưa nền tảng của mình trở thành một chuẩn của IoT, và hiện trạng như đã phân tích trong các phần trên, mỗi bối cảnh IoT lại có thể sử dụng một nền tảng IoT khác nhau. Bên cạnh đó, các công nghệ được ứng dụng trong các nền tảng này cũng khá khác biệt, dẫn tới khó khăn cho người dùng khi muốn sử dụng, khai thác hoặc kết nối các nền tảng IoT cho một kịch bản mở rộng của môi trường thông minh.

# CHƯƠNG 3: GIẢI PHÁP KẾT NỐI CÁC NỀN TẢNG IOT

## Triển khai thử nghiệm mô hình IoT

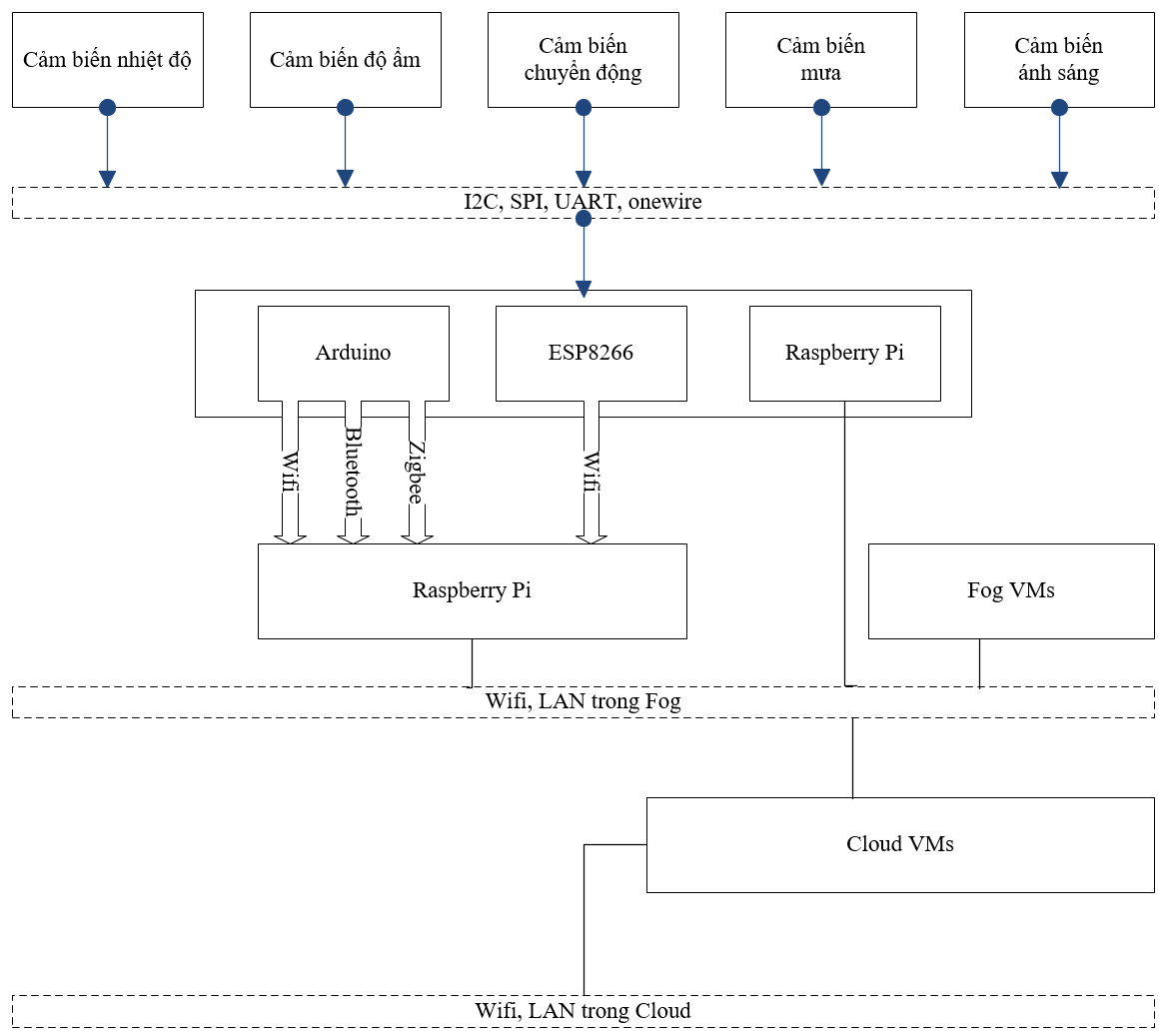
Hệ thống IoT thực tế có những đặc diểm khác biệt với hệ thống thông tin truyền thống do có sự tham gia của các thiết bị điện điện tử, đồng thời cấu hình phần cứng của các nút trong mạng rất khác nhau, kết nối mạng đa dạng. Như đã trình bày ở phần phạm vi đề tài, chúng tôi tạm thời đặt các yếu tố về công nghệ mạng, công nghệ phần cứng của thiết bị ra ngoài phạm vi nghiên cứu. Do đó, chúng tôi sử dụng một hệ thống thử nghiệm với một số bộ gateway, cảm biến với các kết nối có dây và không dây để thử nghiệm và phát triển các tính năng của hệ thống. Hệ thống triển khai thử nghiệm IoT gồm một số thành phần chính như sau:

* Đám mây: nền tảng tính toán với sức mạnh lớn, đặt trên hạ tầng đám mây sử dụng OpenStack, chịu trách nhiệm xử lý các tính toán đòi hòi năng lực tính toán, lưu trữ lớn. Như đã trình bày ở chương trước, chúng tôi triển khai thêm dịch vụ API của Amazon EC2 cho OpenStack. Như vậy, với hai API khác nhau, chúng tôi giả định có hai đám mây dịch vụ khác nhau để có môi trường đa đám mây.
* Sương mù: chứa các nút với cấu hình đa dạng, nhiều dạng kết nối mạng, có thể được hiện thực hóa sử dụng máy tính Raspberry Pi, Arduino, Intel Galileo, máy PC, máy ảo.
* Các cảm biến, các bộ thu thập dữ liệu của cảm biến, truyền dẫn: các cảm biến nhiệt độ, độ ẩm, phát hiện chuyển động, khoảng cách ….. các bộ thu thập dữ liệu cảm biến, có thể sử dụng Arduino, nền tảng ESP8266, các thiết bị truyền dẫn không dây qua Wifi, và truyền dẫn có dây qua mạng LAN.

Các thiết bị chúng tôi thử nghiệm có sẵn trên thị trường, được triển khai và chưa xét đến yếu tố chất lượng của thiết bị (công nghệ). Gói tích hợp đa nền tảng IoT đã được đưa lên Github tại địa chỉ <https://github.com/HPCC-Cloud-Computing/IoTPlane>.

### Kiến trúc tổng quan

Luồng kết nối các thành phần trong hệ thống được mô tả bởi Hình 4.



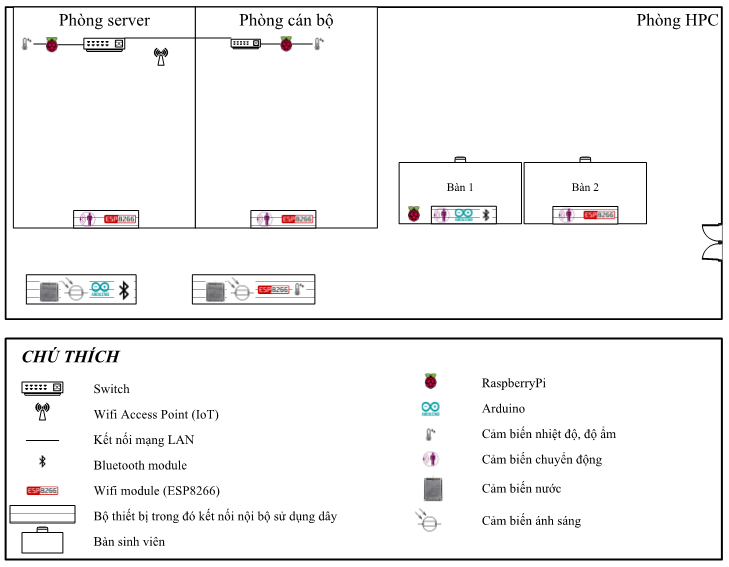
Hình 4: Luồng dữ liệu giữa các thành phần phần cứng

Trong đề tài này, chúng tôi sử dụng các cảm biến đơn giản, được kết nối qua các giao tiếp như I2C, SPI, UART, 1-Wire,… Do các cảm biến này chỉ cung cấp các giao tiếp điện tử, tương đương tầng vật lí trong mô hình OSI, do đó chúng tôi không thể kết nối trực tiếp các cảm biến này vào dịch vụ đám mây mà phải qua một số thiết bị trung gian để thu thập dữ liệu như Arduino, ESP8266, Raspberry Pi, Intel Galileo.

Với các thiết bị trung gian như Arduino và ESP8266 có năng lực xử lý hạn chế, không thích hợp để triển khai các dịch vụ trong tầng sương mù, cũng như mở các kết nối phức tạp đến tầng đám mây nên dữ liệu thu thập được từ các thiết bị này sẽ được truyền đến Raspberry Pi để tổng hợp và tiếp tục thực hiện một số bước xử lý cơ bản trước khi chuyển lên đám mây. Trong khi đó, ESP8266 là một Wifi module, do đó kết nối được sử dụng để kết nối Raspberry Pi là kết nối Wifi thông qua mạng wifi chung hoặc kết nối trực tiếp vào Wifi do Raspberry Pi phát (ở chế độ AP). Với Arduino, có thể sử dụng các module không dây (ví dụ ESP8266 hoặc ESP32) để kết nối thông qua Wifi.

### Các thiết bị triển khai ở trung tâm Tính toán Hiệu năng cao

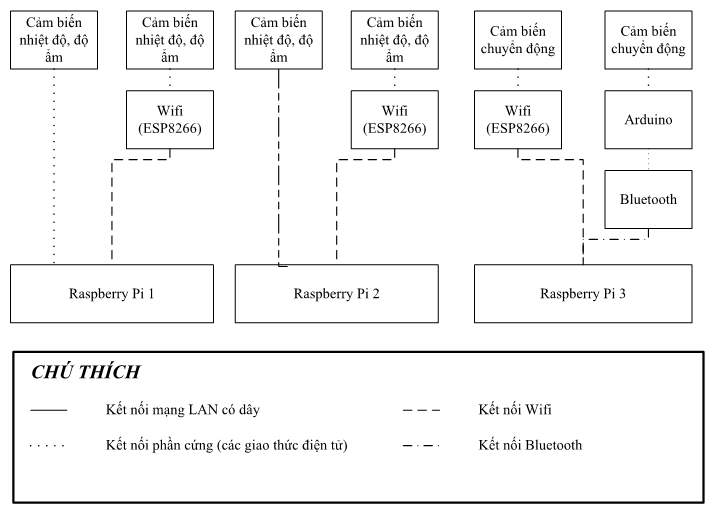
Hệ thống được triển khai gồm 3 bộ cố định được thể hiện trong Hình 5.



Hình 5: Sơ đồ triển khai tại HPC

Thiết bị trong phòng cán bộ và phòng máy chủ bao gồm 1 bộ Raspberry Pi gắn trực tiếp cảm biến nhiệt độ-độ ẩm, kết nối có dây đến switch để kết nối vào mạng sương mù và đám mây, thực hiện truyền tải dữ liệu lên mây và xử lý trong tầng sương mù. Ngoài ra hai phòng này được lắp đặt một bộ cảm biến chuyển động phía trên cửa ra vào để xác định trong phòng có người hay không, cảm biến này được kết nối vào module ESP8266, module ESP8266 này kết nối vào mạng Wifi của môi trường thử nghiệm, bộ phát Wifi (AP) được đặt trong phòng server, các thiết bị tham gia mạng Wifi này có thể kết nối đến các nút trong tầng Sương mù. Các thiết bị trong 2 phòng này sử dụng trực tiếp điện từ adapter (không dùng pin).

Bên cạnh các thiết bị được đặt trong phòng server và phòng giáo viên, có 1 bộ thiết bị phát hiện chuyển được được đặt trên bàn sinh viên, gần cửa ra vào nhằm thử nghiệm sự ảnh hưởng gây ra bởi khác biệt giữa các kết nối không dây.



Hình 6: Mô hình triển khai

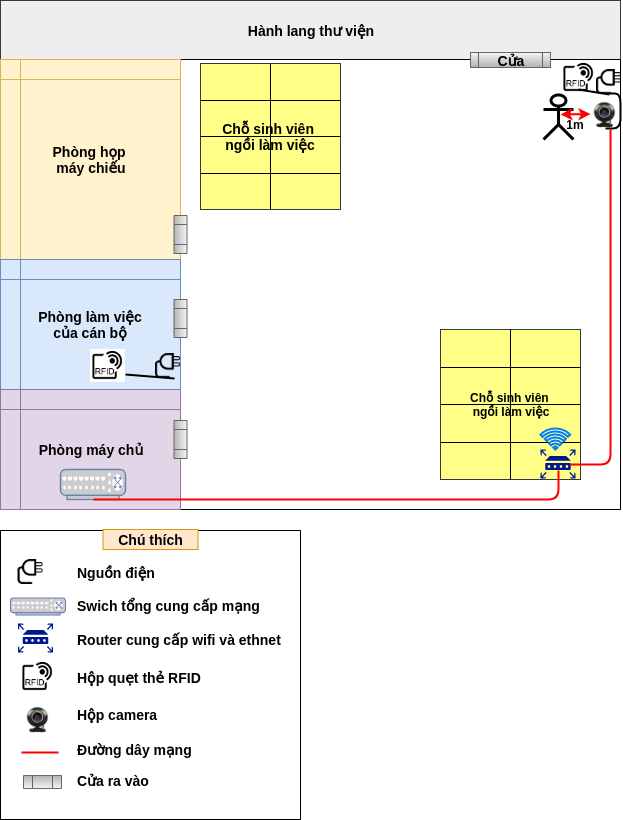
### Hệ thống PAS

Để mở rộng hệ thống hơn, chúng tôi đã triển khai hệ thống xác thực người ra vào Person Authentication System (PAS). Hệ thống này sử dụng kết hợp hai cơ chế là thẻ RFID và nhận diện mặt người để xây dựng cơ chế định danh và xác thực cho các thành viên trong trung tâm HPC. Các hệ thống con được triển khai bao gồm:

* Hệ thống thiết bị quét mã thẻ RFID
* Hệ thống thiết bị quay camera để trích xuất được khuôn mặt phục vụ cho việc nhận diện khuôn mặt để xác thực thành viên.

Các thiết bị quét mã thẻ và quay camera này được lắp đặt ở gần cửa ra vào để thuận tiện cho các thành viên trong quá trình quẹt thẻ và nhận diện khuôn mặt, đồng thời giúp dễ dàng liên kết thiết bị quét mã thẻ với khóa thông minh trên cửa ra vào để xây dựng cơ chế hạn chế sự ra vào trong quá trình phát triển hệ thống PAS sau này.

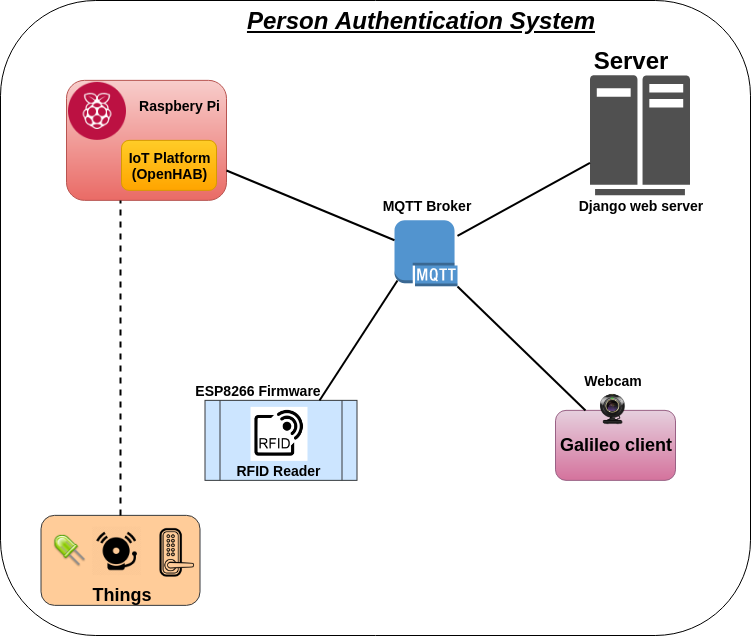
Ngoài ra, để người quản trị có thể cấp thẻ cho các thành viên trong trung tâm HPC thì cần có thêm một thiết bị quét mã thẻ RFID đặt ở vị trí bàn làm việc của admin để có thể dễ dàng quét mã thẻ để lưu vào trong cơ sở dữ liệu của hệ thống, sau đó cấp thẻ đó cho thành viên. Từ các phân tích trên và kết hợp với sơ đồ bố trí các phòng của trung tâm HPCC, hệ thống PAS có thiết kế vật lý như sau:



Hình 7: Mô hình lắp đặt các thiết bị

Như trong mô hình lắp đặt trên thì hai hộp thiết bị quẹt thẻ và hộp camera sẽ được lắp ở gần cửa ra vào, nơi đã có ổ điện để cấp nguồn. Khi các thành viên ra vào trung tâm sẽ cần phải quẹt thẻ tại trước cửa, đồng thời đứng quay mặt vào camera để camera có thể trích xuất được ảnh khuôn mặt để nhận diện. Khi đó, khoảng cách từ người quẹt thẻ tới camera khoảng một mét. Với việc sử dụng webcam Logitech C310 có độ phân giải 5MP, với chất lượng quay HD 720p thì vẫn thu được video chứa hình ảnh khuôn mặt một cách rõ ràng từ khoảng cách một mét để có thể trích xuất ảnh khuôn mặt thông qua phương pháp phát hiện mặt người của Viola-John được cung cấp trong thư viện OpenCV hoặc hệ thống nhận diện khuôn mặt ứng dụng mạng neuron CNN.

Hộp thiết bị camera bao gồm hai thiết bị chính là webcam Logitech C310 và máy tính nhúng Intel Galileo gen 2. Intel Galileo có vai trò điều khiển webcam quay video và chạy thuật toán Viola-John để trích xuất ảnh khuôn mặt. Vì vậy cần đường dây mạng kết nối Intel Galileo với server để tiến hành nhận điện.

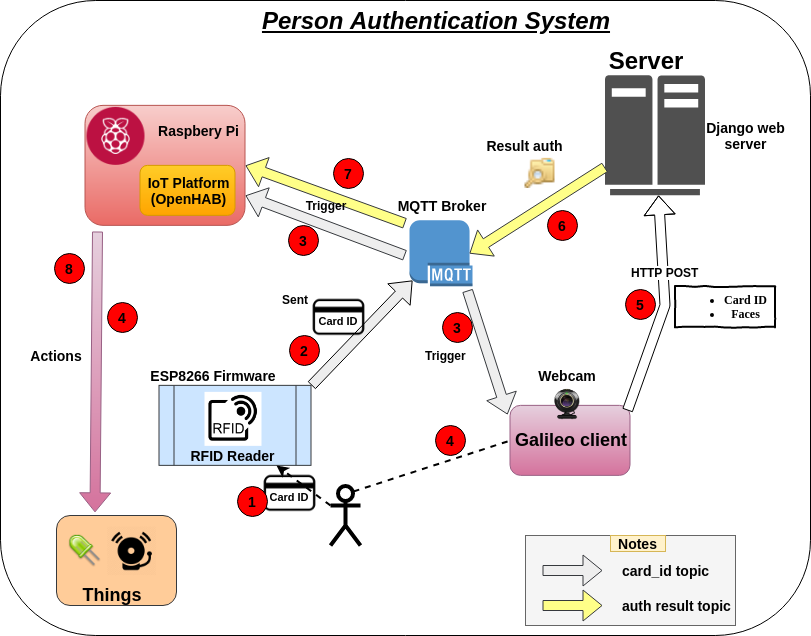


Hình 8: Kiến trúc tổng quan của hệ thống PAS

Diễn giải các thành phần cơ bản:

* ESP 8266 board và RFID Reader: được sử dụng cùng nhau, làm nhiệm vụ đọc thông tin từ thẻ RFID và gửi thông tin về thẻ tới MQTT Broker.
* MQTT Broker: là server thực hiện chạy các topic, đồng thời nhận đăng kí (subscribe) từ các clients yêu cầu các topics, nhận các messages từ clients và chuyển tiếp chúng.
* Galileo và webcam: thực hiện việc quay camera và trích xuất ảnh khuôn mặt để gửi lên server
* Server: là nơi triển khai hệ thống web server và thực hiện quá trình xác thực mã thẻ và nhận diện khuôn mặt.
* Raspbery Pi: thiết bị để triển khai IoT Platform OpenHaB, giúp truyền các hành động tới các thiết bị IoT khi nhận được tín hiệu xác thực từ server, như hành động bật đèn led và còi buzzer để báo hiệu xác thực thành công hay thất bại.
* Things: Các thiết bị phần cứng IoT, được điều khiển bởi OpenHaB.

Luồng dữ liệu chung của hệ thống PAS là quá trình từ lúc một thành viên quẹt thẻ cho tới lúc nhận được thông báo về kết quả nhận diện khuôn mặt. Chi tiết luồng dữ liệu được thể hiện trong sơ đồ dưới đây:



Hình 9: Mô hình luồng dữ liệu trong hệ thống PAS

Diễn giải các bước trong mô hình luồng dữ liệu trên:

**Bước 1:** Người dùng quẹt thẻ tại thiết bị quét thẻ RFID Reader RC522

**Bước 2:** Mã thẻ sau khi quét được gửi lên card\_id topic trên MQTT Broker

**Bước 3**: Sau khi nhận được mã thẻ trên card\_id topic, MQTT broker sẽ gửi message chứa mã thẻ tới tất cả các client đã subcribe card\_id topic là Galileo client và OpenHAB.

**Bước 4:** Quá trình xử lý khi nhận được message chứa mã thẻ trên card\_id topic diễn ra song song tại:

* OpenHab: Thực hiện hành động bật đèn led màu vàng thông báo yêu cầu người dùng nhìn vào webcam.
* Galileo client: điều khiển webcam quay camera và thực hiện quá trình trích xuất ảnh khuôn mặt

**Bước 5:** Sau khi đã trích xuất đủ 4 ảnh gương mặt, Galileo client gửi dữ liệu bao gồm 4 ảnh khuôn mặt và mã thẻ RFID lên Django server thông qua HTTP POST.

**Bước 6:** Server tiến hành quá trình xác thực mã thẻ và nhận diện khuôn mặt, sau đó gửi message chứa kết quả xác thực lên auth\_result topic trên MQTT broker.

**Bước 7:** MQTT broker gửi message chứa kết quả xác thực tới OpenHAB

**Bước 8:** Dựa trên kết quả xác thực, OpenHAB sẽ thực hiện một trong hai hành động sau:

* Bật đèn led đỏ báo hiệu thất bại nếu kết quả xác thực thất bại.
* Bật còi báo hiệu thành công nếu kết nhận được kết quả xác thực thành công.

Ngoài ra, chúng tôi triển khai một số bộ thiết bị sử dụng kết nối Wifi, sử dụng thành phần xử lý là Raspberry Pi hoặc, Arduino kết hợp module truyền dẫn không dây (Wifi) hoặc sử dụng module ESP8266 với các trường hợp đơn giản, trên các bộ thiết bị này được gắn một hoặc một số cảm biến như cảm biến nhiệt độ, độ ẩm, cảm biến nước, cảm biến ánh sáng. Các thiết bị có thành phần xử lý là Raspberry Pi đóng vai trò một nút trong tầng sương mù.

## Thiết kế và phát triển phần mềm tích hợp các nền tảng IoT

### Mô hình dữ liệu tập trung

Trong phần này chúng tôi thiết kế một mô hình thông nhất, phục vụ việc trong đổi thông tin giữa các nền tảng khác nhau, và các thành phần giao tiếp trong hệ thống. Chúng tôi thiết kế mô hình tổng quát và khả mở. Hình 10 thể hiện mô hình thông tin này, bao gồm các thành phần chính sau:



Hình 10: Mô hình dữ liệu

* IoTSource: cung cấp khả năng cho các nguồn dữ liệu (ví dụ: các nền tảng IoT, gateway, các dịch vụ mạng, dịch vụ xử lý thông tin trong tầng sương mù và đám mây, và các dịch vụ khác)
* Timestamp: thể hiện các điểm thời gian khi dữ liệu được thu thập, trao đổi từ các nguồn.
* Metric: được sử dụng để thể hiện các độ đo khác nhau của các dữ liệu thu thập được
* DataPoint: các Metric lưu các điểm dữ liệu để hoàn thiện các thông tin thu thập.
* Log: thể hiện các dữ liệu nhật kí (ví dụ, hoạt động, lưu vết) được khi trong các đối tượng IoT
* Capability: Lưu thông tin về các giao diện của các IoTSource, cho phép các thành phần khác nhau có khả năng trao đổi thông tin, điều khiển linh hoạt và có khả năng mở rộng.

Các thiết kế mô hình dữ liệu tập trung, triển khai hạ tầng thử nghiệm IoT của chúng tôi được trình bày trong bài báo [43].

### Các thành phần dịch vụ

Hệ thống gồm các thành phần chính như: Driver, Filter, Forwarder, DBreader DBwriter, Collector, Registry



Hình 11: Kiến trúc tổng quan của hệ thống phần mềm tích hợp các nền tảng IoT

* Module Driver có nhiệm vụ kết nối với các nền tảng IoT bên dưới, đại diện cho các nền tảng này trong giao tiếp với hạ tầng hệ thống, các giao tiếp này bao gồm thu thập dữ liệu, nhận lệnh điều khiển nền tảng. Trong tài liệu này, nền tảng IoT và Driver có thể được dùng thay thế cho nhau.
* Các Filter trong tầng sương mù cung cấp các chức năng tiền xử lý dữ liệu, ra quyết định, và một số chức năng khác có thể thực thi trong tầng sương mù. Các filter nhận dữ liệu từ một nguồn nhất định, xử lý và chuyển kết quả đến một điểm khác cho trước. Forwarder cũng có thể coi là 1 filter tuy nhiên không có tính năng xử lý dữ liệu mà chỉ đóng vai trò chuyển tiếp dữ liệu giữa các tầng.
* Các module DBreader, DBwriter có nhiệm vụ cung cấp giao diện với cơ sở dữ liệu lưu trữ các dữ liệu cảm biến trong quá trình vận hành.
* Module Collector xử lý đơn giản, lưu trữ (thông qua DBwriter) dữ liệu từ cái driver đưa đến.
* Registry quản lý, lưu trữ các thông tin về trạng thái hệ thống bao gồm trạng thái các driver, sensor, các module filter,…

### Lưu trữ

Dữ liệu của các thiết bị khác nhau sẽ được lưu trên hệ quản trị CDSL InfluxDB. Đây là hệ quản trị cơ sở dữ liệu dòng thời gian phù hợp với các dữ liệu sinh ra liên tục của các cảm biến. Nhờ có mô hình dữ liệu thống nhất về định dạng, các dữ liệu của thiết bị sẽ được lưu một cách hiệu quả cho việc truy vấn ở tầng ứng dụng. Mô hình dữ liệu được trình bày ở phần 3.2. Ví dụ, dữ liệu trả về nhiệt độ từ cảm biến quản lý bởi nền tảng OpenHAB có định dạng như sau:

Temperature 30.2

Location Zone1

Property current

Unit C

Trong khi đó, dữ liệu cảm biến nhiệt độ được thu thập thông qua nền tảng HomeAssissteant có dạng như sau:

{

"name": "Outside",

"device": "weather-ha",

"type": "air",

"data": {"temp": "24C"}

}

Bởi vì tính đa dạng của định dạng dữ liệu giữa các nền tảng khác nhau như ví dụ trên, chúng tôi phải chuyển đổi định dạng sử dụng mô hình dữ liệu đề xuất nhằm hỗ trợ việc truy vấn dữ liệu ở tầng ứng dụng xuống hiệu quả và nhanh chóng. Theo đó, dữ liệu sau khi được chuyển đổi sẽ có định dạng như sau:

<Metric>

<MetricName>Temperature</MetricName>

<Units>

<Data>Celsius</Data>

</Units>

<DataPoint>

<DataType>Float</DataType>

<Value>24</Value>

</DataPoint>

</Metric>

<IoTSource>

<SourceId>outside-sensor-temp</SourceId>

<SourceType>Sensor</SourceType>

<State>active</State>

<Description>{"device": "weather-ha", "type": "air"}</Description>

<PlatformType>HomeAssistant</PlatformType>

<Version>v1</Version>

</IoTSource>

### Các giao thức phục vụ giao tiếp

Phần này trình bày các giao thức sử dụng trong hệ thống, được chia thành các nhóm:

1. Các giao thức trao đổi thông tin

Cung cấp khả năng trao đổi thông tin giữa các nền tảng IoT với nhau và các thành phần khác trong hệ thống là một trong các yêu cầu cốt lõi đối với hệ thống. Các thông tin được trao đổi bao gồm các dữ liệu thu thập được từ môi trường, trong hệ thống và được suy diễn ra trong quá trình hoạt động của hệ thống; đồng thời các thông tin điều khiển cũng cần được trao đổi, truyền thông trong suốt trong toàn bộ hệ thống. Trong luận văn này đề xuất các giao thức trao đổi thông tin có khả thực hiện các thao tác sau:

* Gửi thông tin chủ động từ các IoTSources lên các bộ phận xử lý tiếp theo.
* Yêu cầu gửi thông tin và gửi thông tin bị động khi được yêu cầu.
* Gửi thông tin tin cậy/không tin cậy.
* Chuyển đổi cách biểu diễn thông tin giữa các nền tảng (thực hiện trên các driver)

1. Các giao thức điều khiển hệ thống (quản lý thành phần hệ thống)

Các giao thức điều khiển là các giao thức liên quan việc điều khiển hệ thống, bao gồm quản lý các thành phần hệ thống (IoT driver, dịch vụ). Khi một dịch vụ muốn tham gia vào hệ thống, dịch vụ đó cần gửi thông tin yêu cầu đăng nhập với Registry, nếu đây là dịch vụ mới, registry sẽ cấp cho dịch vụ này một ServiceId, dịch vụ có nhiệm vụ lưu trữ Id này, sử dụng khi đăng nhập. Tiếp theo, registry gửi yêu cầu cung cấp các thông tin, khả năng của nút, được thể hiện qua các trường của IoTSource, registry sẽ lưu trữ các thông tin này trong cơ sở dữ liệu, kèm theo ngày cập nhật và gửi một thông điệp Ack cho dịch vụ để thông báo dịch vụ có thể hoạt động trong hệ thống. Khi một dịch vụ đã có ServiceId, dịch vụ này có thể gửi thông tin yêu cầu đăng nhập hệ thống, kèm theo ServiceId, khi Registry đã xác nhận, các bước sau thực hiện giống yêu cầu tham gia hệ thống, tuy nhiên yêu cầu UpdateInfoRequest chỉ yêu cầu gửi các thông tin thay đổi so với lần cập nhật trước (nếu dịch vụ chưa cài đặt tính năng này, có thể gửi toàn bộ thông tin như khi đăng ký).

1. Các giao thức điều khiển luồng dữ liệu

Các giao thức này cho phép điều khiển luồng dữ liệu (gửi dưới dạng batch hay gửi từng data item, phương pháp đóng gói giảm thiểu lượng dữ liệu truyền tải). Giao thức điều khiển luồng dữ liệu được sử dụng bởi các module filter trong tầng sương mù, cung cấp các chức năng sau:

* Gom các data item nhỏ thành một gói truyền tải, từ đó giảm overhead so với truyền tải từng item riêng lẻ.
* Điều chỉnh tần suất gửi, cập nhật dữ liệu.
* Cung cấp khả năng chặn/dừng luồng dữ liệu.

Bảng 3: Cấu trúc lệnh điều chỉnh luồng dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên tham số** | **Ý nghĩa** | **Bắt buộc/Tùy chọn** | **Giá trị** |
| ServiceId | Id của service cần điều chỉnh | Tùy chọn | Nếu bằng NULL: tất cả các service nhận được lệnh này đều điều chỉnh |
| MetricName | Tên metric cần điều chỉnh | Tùy chọn | Nếu bằng NULL: tất cả các Metric trên service này đều được điều chỉnh |
| Mode | Chế độ cần chuyển:   * Gửi liên tục (Continouse) * Gửi theo lô (Batch) * Dừng (Pause) | Bắt buộc | Continuous/Batch/Pause |
| BatchMode | Chế độ đóng gói dữ liệu:   * Theo thời gian * Theo số lượng | Bắt buộc với Mode = Batch | Time/Quantity |
| Frequency | Với chế độ gửi liên tục: thay đổi tần suất lấy mẫu (nếu có thể)  Với chế độ gửi lô:   * Theo thời gian: khoảng thời gian giữa 2 lần gửi * Theo số lượng: số lượng item nhận được trước khi gửi | Bắt buộc với Mode khác Pause | * Tần suất: item/s * Thời gian: s * Số lượng: số item |
| FlushBuffer | Sử dụng khi rời khỏi trạng thái Pause, chỉ định service có gửi các dữ liệu chưa được gửi không, (tùy thuộc service hỗ trợ) | Tùy chọn | True/False |

1. Các giao thức điều khiển kết nối các thành phần

Giao thức điều khiển kết nối các thành phần trong hệ thống cho phép các nút, dịch vụ đăng ký, đăng nhập vào hệ thống, kiểm soát phiên và phân quyền. Các nút và dịch vụ được quản lý bao gồm cả ở tầng sương mù và đám mây.

Các giao thức kết nối các thành cho phép kết nối động giữa các thành phần, từ đó tăng tính mềm dẻo của hệ thống. Các giao thức này phần lớn dựa trên hệ thống truyền thông sử dụng message queue. Do mục tiêu thiết kế hệ thống là khả mở, linh hoạt trong việc thêm bớt các dịch vụ, điều chỉnh linh hoạt luồng dữ liệu và việc xử lý dữ liệu, do đó cần thiết có một cơ chế cho phép điều chỉnh việc kết nối giữa các service với nhau một cách linh hoạt, dựa trên mô hình thông tin đã trình bày ở 2.1 trong chương này. Chúng tôi đề xuất sử dụng các khái niệm, chức năng của các message queue: các service được kết nối với nhau thông qua các topic gửi/nhận dữ liệu. Như vậy, việc điều chỉnh kết nối các thành phần có thể thực hiện thông qua việc cấu hình các topic gửi/nhận dữ liệu của các service.

Bảng 4: Cấu trúc lệnh điều chỉnh luồng dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên tham số** | **Ý nghĩa** | **Bắt buộc/Tùy chọn** | **Giá trị** |
| ServiceId | Service nhận lệnh này | Bắt buộc |  |
| InputTopic | Topic nhận dữ liệu | Bắt buộc |  |
| OutputTopic | Topic gửi dữ liệu | Bắt buộc |  |

Bên cạnh đó, việc quản lý cấu hình là một phần không thể thiếu trong mọi hệ thống, với hệ thống phân tán, đa dạng trong môi trường IoT, các cấu hình cần được quản lý tập trung, cho phép cập nhật dễ dàng. Các cấu hình được lưu trữ theo từng service, từng node, được lưu dưới dạng JSON hoặc được serialize, được đánh phiên bản với nguyên tắc không sửa cấu hình cũ mà chỉ tạo ra phiên bản mới.

Bảng 5*:* Các thuộc tính của một cấu hình

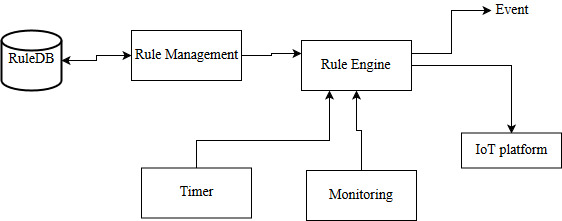
|  |  |
| --- | --- |
| **Tên trường** | **Ý nghĩa** |
| ID | ID của cấu hình |
| ServiceID | ID của service tương ứng |
| ParentID | Cấu hình được kế thừa |
| Configuration | Cấu hình tương ứng |
| Version | Phiên bản |

### Engine xử lý luật

1. Kiến trúc tổng quan

Với các thông tin đã thu thập được, khả năng xử lý, ra quyết định là không thể thiếu đối với một hệ thống IoT, các quyết định có thể là các hành động tác động đến các hệ thống vật lý, ví dụ: đóng/mở cửa, tắt/bật đèn,…., các hành động tác động đến hệ thống: block node, thay đổi các tham số thành phần trong hệ thống, thay đổi cấu trúc mạng, kết nối của hệ thống. Kết quả của việc xử lý cũng có thể tạo ra các sự kiện mới trong hệ thống. Phục vụ cho engine xử lý luật có các module:

* Engine xử lý
* Module quản lý luật
* Module quản lý sự kiện, thời gian.



Hình 12: Engine xử lý các luật điều khiển đa nền tảng IoT

1. Cấu trúc luật

Một số yêu cầu với engine xử lý luật:

* Đầu vào:
* Các độ đo trong hệ thống
* Các hàm áp dụng lên các độ đo (ví dụ các hàm xử lý chuỗi, số ….)
* Sự kiện trong hệ thống (với một số điều kiện về tham số đi kèm)
* Sử dụng các toán tử so sánh, logic, gom nhóm.
* Kết quả:
* Thiết lập trạng thái của hệ thống (trạng thái nội bộ hoặc trạng thái thiết bị kết nối với hệ thống như đèn,…)
* Thiết lập timer.
* Phát sinh một sự kiện mới trong hệ thống.

Cấu trúc luật:

<rule> ::= IF <condition> THEN <result>

<condition> ::= <expr> |

<condition> “and” <condition> |

<condition> “or” <condition> |

“not” <condition> | “(“ <condition> “)”

<expr> ::= <expr> <op> <value> |

func(<expr>)

<op> ::= “<” | “>” | “<=” | “>=” | “==” | “!=”

<result> ::= “set\_state(“ <thing> “)” |

“event”

Trong đó “value” là một giá trị số hoặc chuỗi; func là các hàm xử lý dữ liệu, trả ra giá trị logic.

Ví dụ:

* + Cảnh báo khi nhiệt độ > 50 độ.
  + Nếu không có người trong phòng thì tắt đèn.
  + Nếu không có người trong phòng thì kích hoạt timer “no\_person”
  + Timer “no\_person” timeout thì tắt đèn.
  + Nếu có người đi vào phòng thì hủy timer “no\_person”.

# CHƯƠNG 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ

## Kịch bản kiểm thử

Kịch bản kiểm thử của hệ thống được phát triển trong đề tài dựa trên thuyết minh đã được phê duyệt. Cụ thể kịch bản tự kiểm thử được trình bày ở bảng 6.

Bảng 6: Kịch bản kiểm thử hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kịch bản thử nghiêm** | **Yêu cầu nghiệm thu** |
| 1. | IoT Workload Orchestration | * Kết nối ít nhất 02 nền tảng IoT phổ biến được triển khai trên các gateway trong môi trường sương mù. * Cung cấp giao diện lập trình thống nhất cho phép quản lý, gửi nhận dữ liệu từ các cảm biến. | * Điều khiển được các cảm biển được quản lý bởi ít nhất 02 nền tảng IoT khác nhau với các chức năng cơ bản. * Thu thập được dữ liệu sinh từ các cảm biến thuộc quản lý bởi ít nhất 02 nền tảng IoT khác nhau. |

Để thực hiện thử nghiệm dựa trên kịch bản kiểm thử trên, chúng tôi tự thực hiện các thử nghiệm như sau:

1. **Thử nghiệm 1:** triển khai ba nền tảng IoT khác nhau trên các gateway trong môi trường điện toán sương mù (Rasberry Pi 3). Thử nghiệm việc điều khiển các thiết bị thành công trên giao diện web của từng nền tảng.

**Mục đích:** thử nghiệm này được thực hiện nhằm mục đích tạo môi trường thử nghiệm cho tất cả các kịch bản kiểm thử và chứng minh chúng tôi đã xây dựng được mô hình thử nghiệm IoT. Kết hợp với thử nghiệm 2, thử nghiệm này thỏa mãn kịch bản kiểm thử 1.

1. **Thử nghiệm 2:** sử dụng hệ thống quản lý đa nền tảng IoT (IoT Workload Orchestration) của chúng tôi để nhận các luồng dữ liệu lên (từ tầng thiết bị, qua tầng sương mù tới tầng đám mây) và gửi luồng dữ liệu xuống (từ tầng đám mây, qua tầng sương mù xuống tầng thiết bị) cho cả ba nền tảng IoT thông qua giao diện thống nhất.

**Mục đích:** thử nghiệm này được thực hiện để chứng minh hệ thống của chúng tôi kết nối được ít nhât 03 nền tảng IoT (triển khai trong các gateway) và cung cấp được giao diện lập trình thống nhất để điều khiển được các cảm biến thuộc 03 nền tảng IoT đó (luồng dữ liệu xuống) và thu thập được dữ liệu từ các nền tảng IoT này (luồng dữ liệu lên). Kết hợp với thử nghiệm 1, thử nghiệm này thỏa mãn kịch bản kiểm thử 1.

1. **Thử nghiệm 3:** xây dựng giao diện web điều khiển đa nền tảng IoT và ứng dụng cảnh báo dựa vào dữ liệu thu thập được từ các thiết bị trong môi trường IoT. Các ứng dụng này được triển khai trên máy ảo đám mây dùng CAL. Thử nghiệm giao diện web và ứng dụng cho phép nhận dữ liệu từ các thiết bị và điểu khiển các thiết bị (luồng dữ liệu từ thiết bị đi lên và luồng điều khiển từ thiết bị đi xuống).

**Mục đích:** xây dựng đượcứng dụng thử nghiệm trong môi trường đa đám mây điều khiển, nhận, gửi dữ liệu của các cảm biến thành công thông qua giải pháp phần mềm đề xuất của đề tài. Chứng minh hệ thống tích hợp đa nền tảng IoT và đa đám mây đã hoàn thành. Thử nghiệm này thỏa mãn kịch bản thử nghiệm 3.

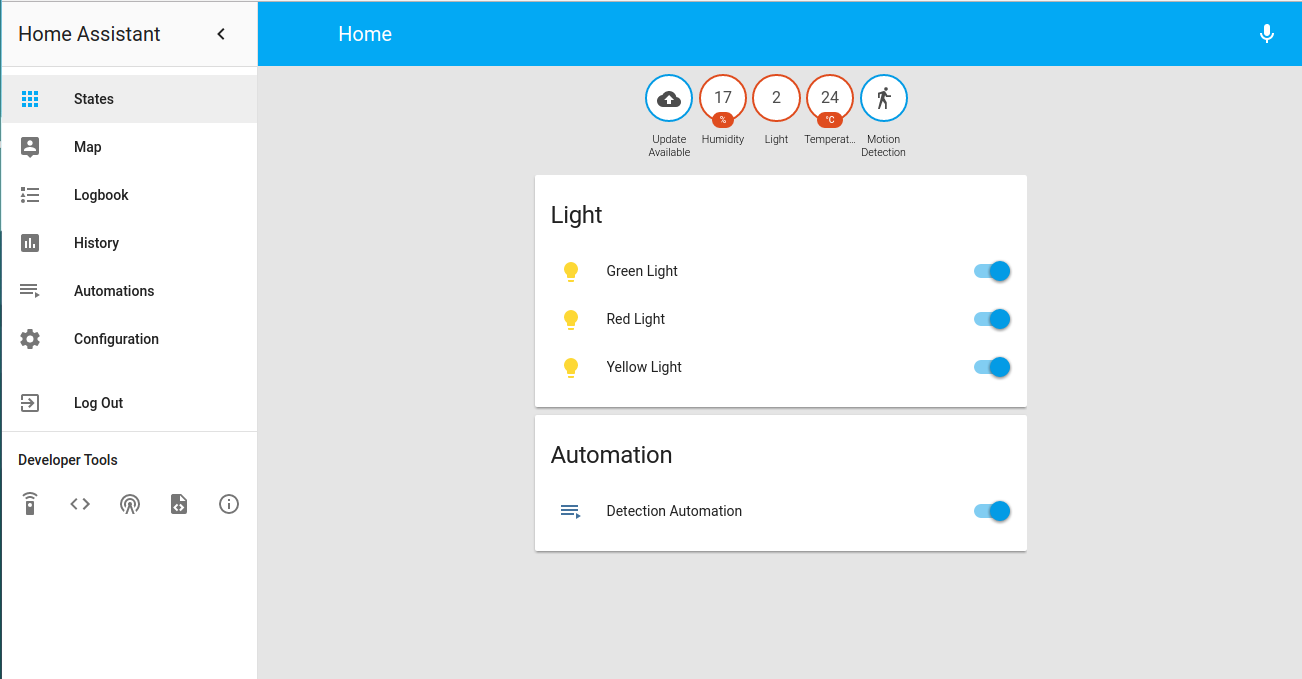
Các thử nghiệm được trình bày lần lượt ở các mục 2, 3, 4 và 5 bên dưới tương ứng với thứ tự các thử nghiệm ở trên.

## Thử nghiệm hạ tầng với các nền tảng IoT

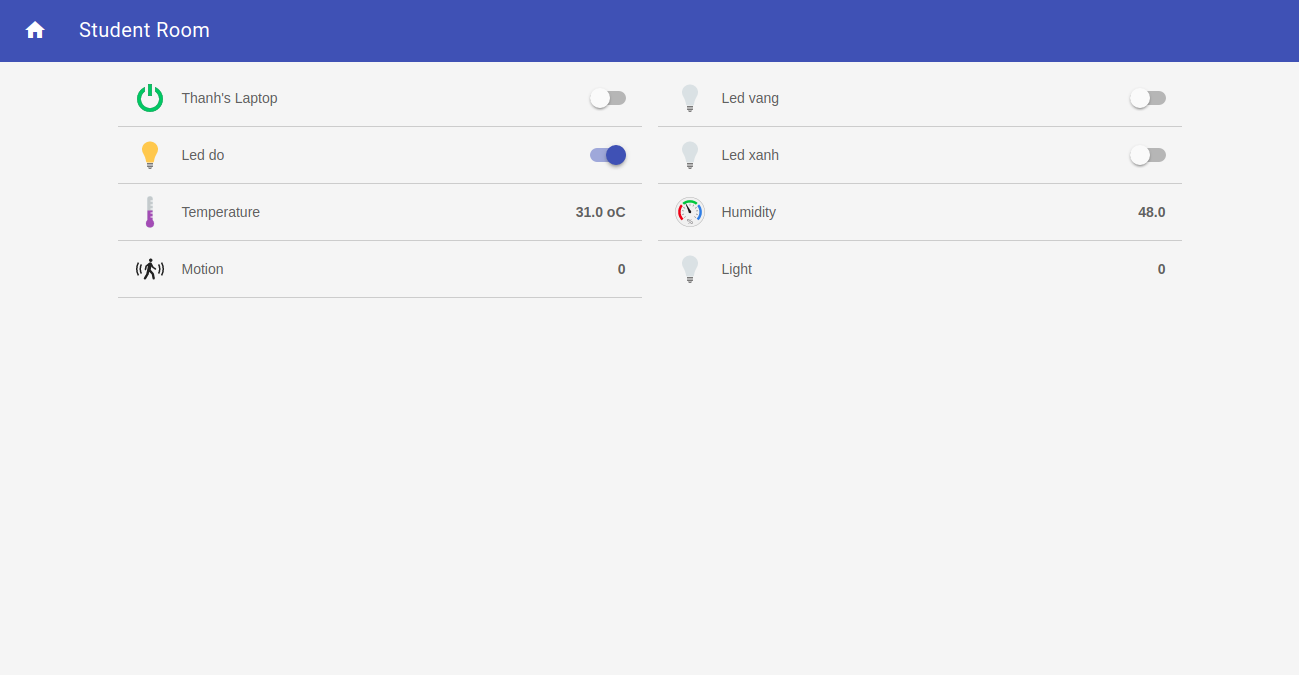
Như đã trình bày ở các chương trước, chúng tôi sẽ triển khai ba nền tảng IoT là Home-Assisant, OpenHab và ThingsBoard. Mỗi một nền tảng IoT sẽ có các thiết bị như sau:

* 1 cảm biến nhiệt độ
* 1 cảm biến độ ẩm
* 1 cảm biến ánh sáng
* 1 cảm biến chuyển động
* 3 đèn LED

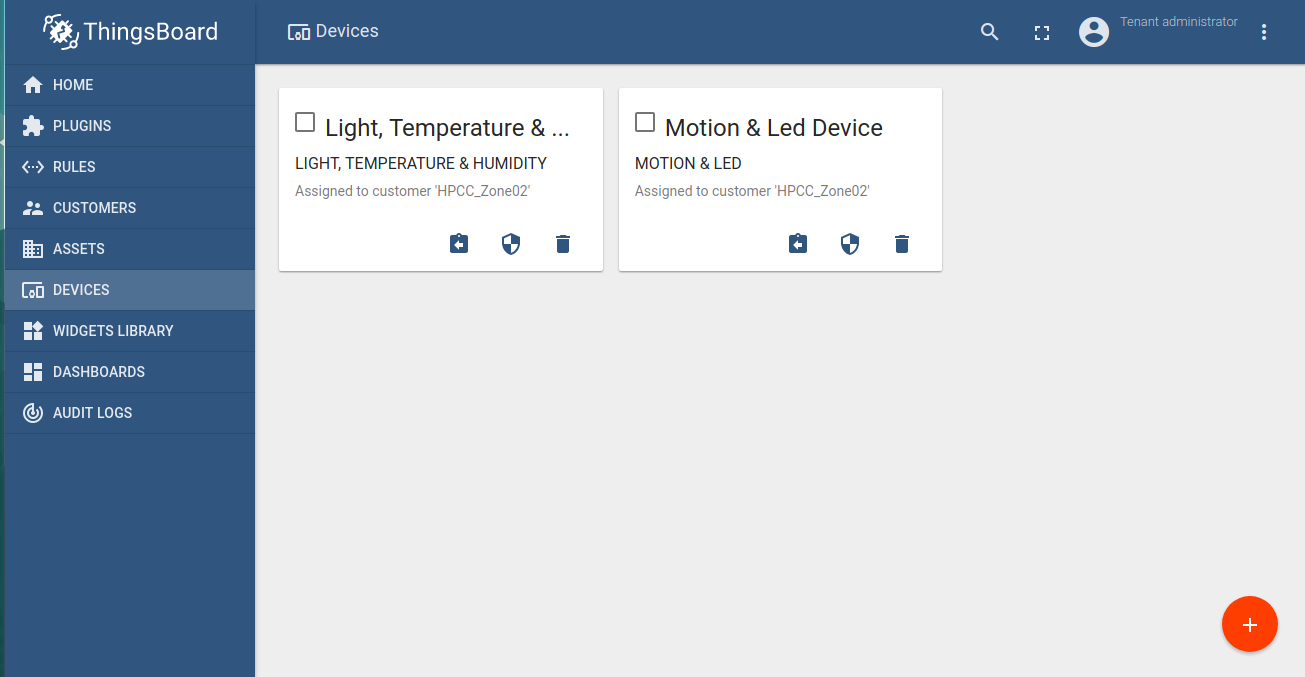
Để thu thập được dữ liệu, chúng tôi đã kết nối các cảm biến và đèn LED với thiết bị Arduino. Arduino chịu trách nhiệm lấy dữ liệu và đóng gói dữ liệu chuyển đến ESP 8266. ESP 8266 sẽ được kết nối tới mạng LAN, nhờ thế mà khi nhận dữ liệu từ Arduino chuyển đến nó có thể chuyển tiếp dữ liệu đến nền tảng IoT được triển khai trên Raspberry Pi 3 trong cùng mạng LAN. Sau khi hoàn thành quá trình cài đặt và triển khai, chúng ta có 3 nền tảng IoT với giao diện như sau:



Hình 13: Giao diện của HomeAssistant



Hình 14: Giao diện của OpenHab



Hình 15: Giao diện của ThingsBoard

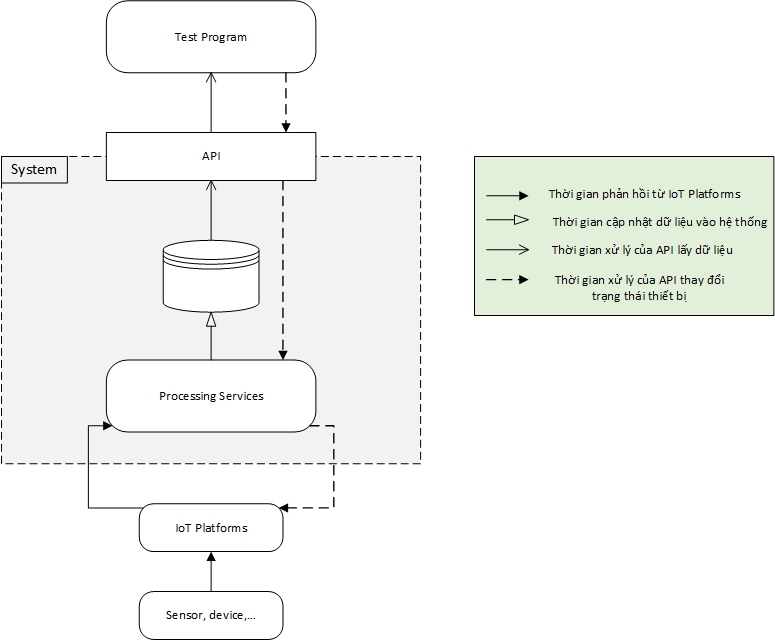
Kết quả thử nghiệm các thao tác trên các nền tảng IoT được trình bày ở Bảng 7. Thông qua kết quả thử nghiệm, có thể kết luận rằng, chúng tôi đã triển khai được các nền tảng IoT và các thiết bị phần cứng liên quan. Các thiết bị đã hoạt động đúng theo yêu cầu phục vụ cho việc thử nghiệm và đánh giá phần mềm liên kết trong các thí nghiệm sau.

Bảng 7: Kết quả khi thực hiện một số thao tác với các nền tảng IoT

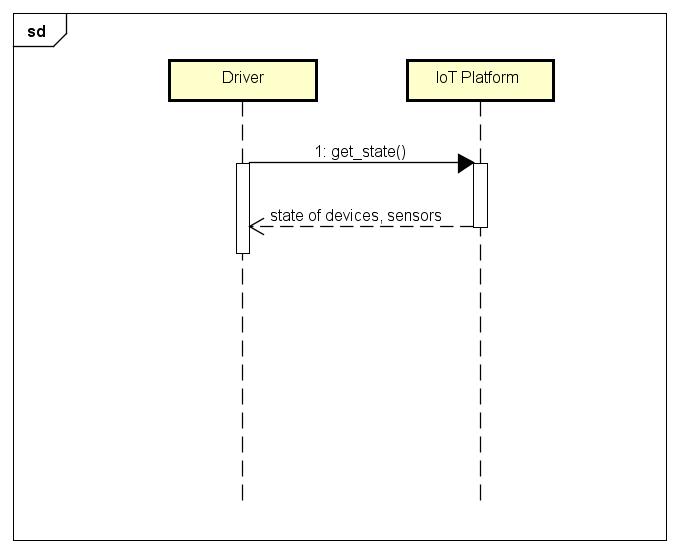
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Kịch bản** | **Kết quả mong muốn** | **Kết quả thực tế** | | |
| Home-Assistant | OpenHab | ThingsBoard |
| 1 | Bật 1 đèn | Đèn bật và trên giao diện hiển thị trạng thái bật | Đạt | Đạt | Đạt |
| 2 | Tắt 1 đèn | Đèn tắt và trên giao diện hiển thị trạng thái tắt | Đạt | Đạt | Đạt |
| 3 | Tắt cả 3 đèn | 3 đèn đều tắt và giao diện hiển thị trạng thái 3 đèn đều tắt | Đạt | Đạt | Đạt |
| 4 | Bật cả 3 đèn | 3 đèn đều bật và giao diện hiển thị trạng thái 3 đèn đều bật | Đạt | Đạt | Đạt |
| 5 | Có người di chuyển trong khu vực | Trên giao diện hiển thị có người | Đạt | Đạt | Đạt |

## Thử nghiệm đánh giá thời gian khi thực hiện một số thao tác với hệ thống tích hợp các nền tảng IoT

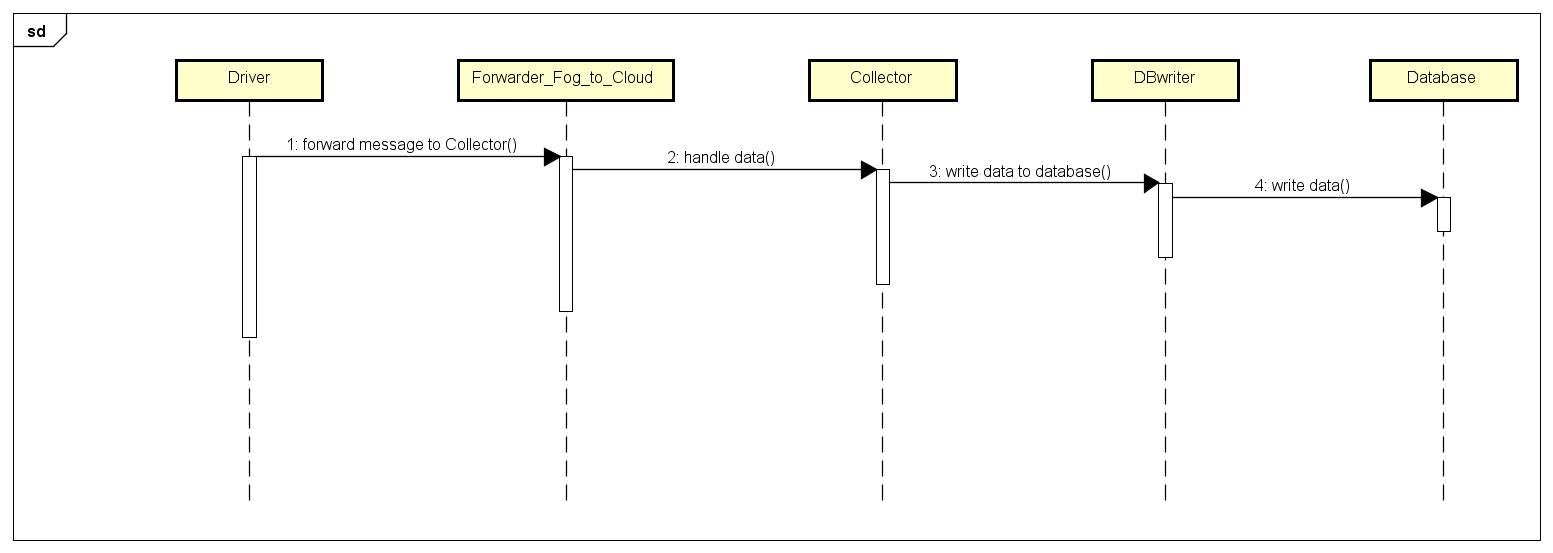
Ở phần thử nghiệm này, chúng tôi sẽ lập trình chương trình nhỏ để có thể lấy dữ liệu liên tục và đồng thời thay đổi trạng thái của một số thiết bị. Mô hình triển khai như sau:



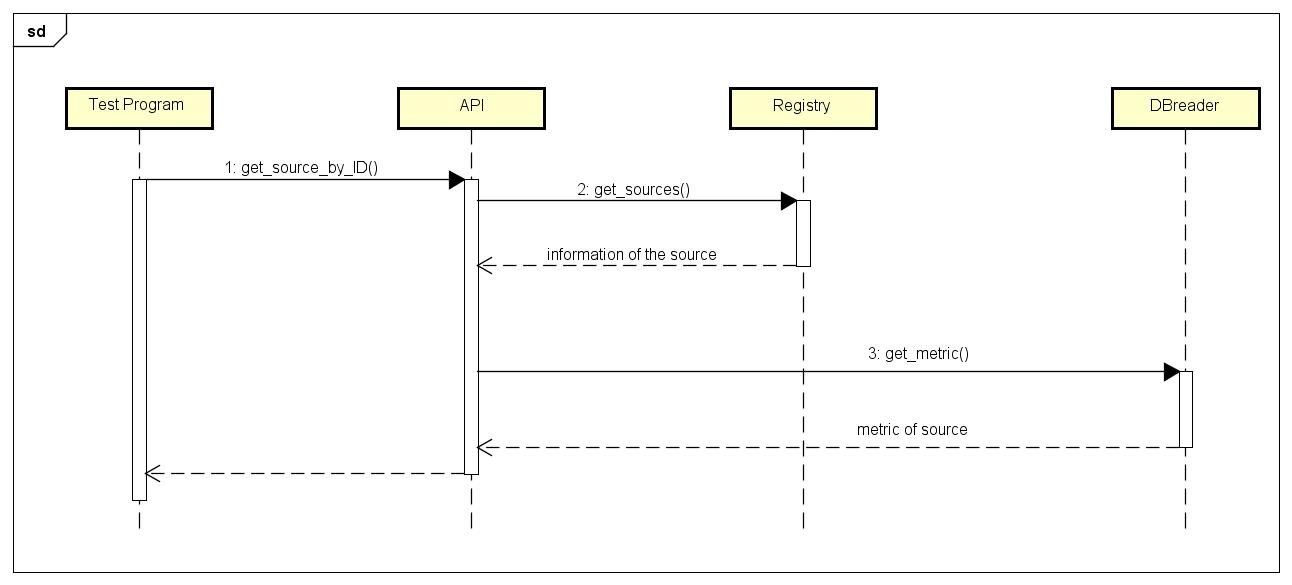
Hình 16: Mô hình triển khai trong quá trình kiểm thử



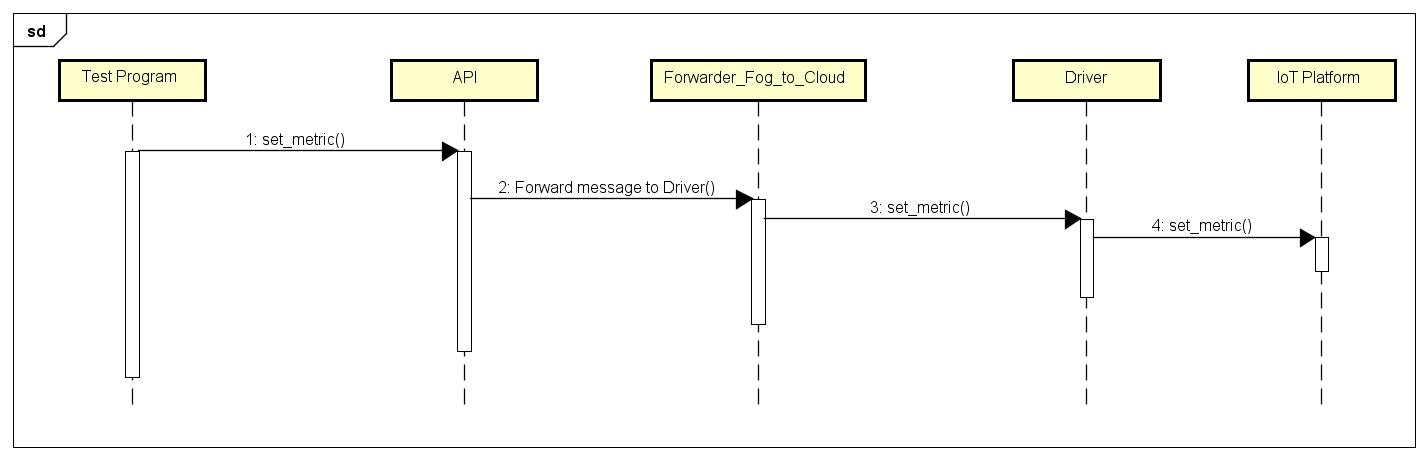
Hình 17: Biểu đồ trình tự quá trình thu thập dữ liệu từ nền tảng IoT



Hình 18: Biểu đồ trình tự quá trình cập nhật dữ liệu vào hệ thống



Hình 19: Biểu đồ trình từ quá trình lấy trạng thái của một thiết bị trong hệ thống



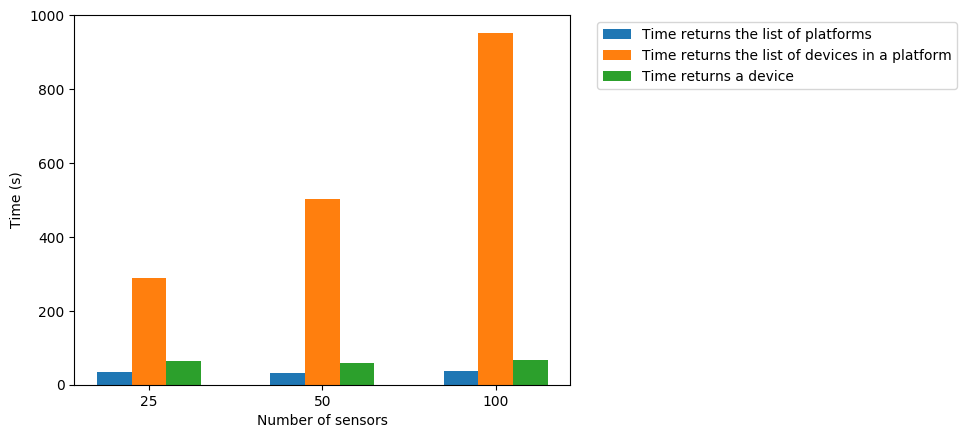
Hình 20: Biểu đồ trình tự quá trình thay đổi trạng thái của thiết bị

Với mô hình trên, nhóm đề tài đã thay đổi một số thông số cài đặt của hệ thống, lên kế hoạch nhiều kịch bản và đo đạc nhiều lần để đưa ra các chỉ số trong nhiều trường hợp khác nhau đồng thời mang tính khách quan cao nhất. Các thông số được đặt như sau:

* Thay đổi số lượng cảm biến *giả lập* trong hệ thống (25 cảm biến, 50 cảm biến, 100 cảm biến).
* Thay đổi chu kỳ gửi dữ liệu của các cảm biến ( 1 giây, 1.5 giây, 2 giây).

Chúng tôi đã thay đổi số lượng thiết bị, cảm biến trong hệ thống để đo đạc 3 chức năng thường dùng nhất đó là:

* Chức năng (API) lấy danh sách các nền tảng IoT trong hệ thống
* Chức năng (API) lấy danh sách tất cả các thiết bị trong một nền tảng IoT bất kỳ
* Chức năng (API) lấy một thiết bị bất kỳ trong hệ thống



Hình 21: Biểu đồ trung bình kết quả khi thay đổi số lượng thiết bị

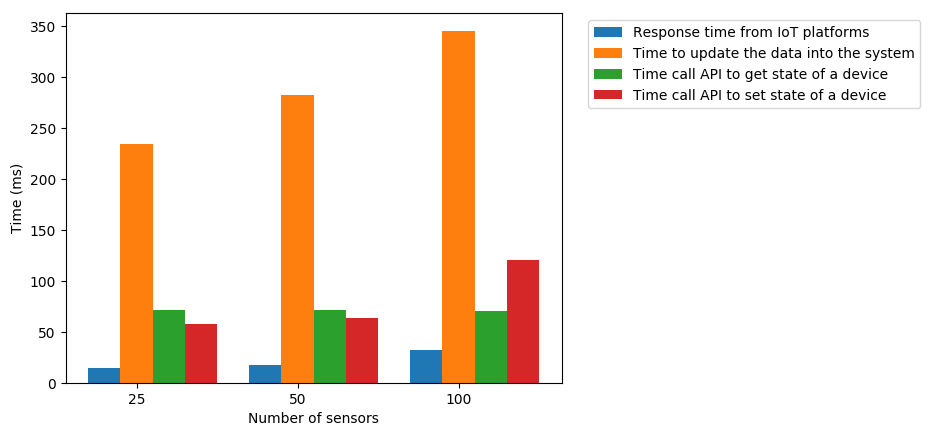
Biểu đồ trung bình kết quả theo thời gian khi thay đổi số lượng thiết bị được trình bày ở Hình 21. Ta có thể đưa ra một số nhận xét như sau:

* Thời gian lấy ra danh sách các nền tảng IoT không thay đổi dù cho số lượng thiết bị hay cảm biến trong hệ thống có thay đổi tăng hay giảm
* Thời gian lấy ra danh sách tất cả các thiết bị sẽ tăng nếu số lượng các thiết bị tăng và ngược lại. Điều này cũng rất dễ hiểu do một số nguyên nhân sau:
* Xử lý với cơ sở dữ liệu: số lượng thiết bị lớn thì thời gian phản hồi và truy vấn trong cơ sở dữ liệu sẽ tăng là điều không tránh khỏi
* Thời gian truyền tải: lượng dữ liệu trả về cho lớn thì thời gian truyền tải tăng cao
* Dù thời gian lấy ra danh sách tất cả các thiết bị có tăng khi số lượng thiết bị tăng nhưng ta có thể thấy với 100 thiết bị nhưng thời gian xử lý cũng chưa đến 1 giây. Đây cũng không phải là một con số quá đáng lo ngại khi xử lý với một số lượng thiết bị lớn như vậy nên có thể chấp nhận được.
* Thời gian lấy ra một thiết bị bất kỳ trong hệ thống không hề thay đổi dù số lượng các thiết bị có thay đổi như thế nào.

Nhóm thực hiện đề tài tiếp tục thay đổi số lượng thiết bị và chu kỳ thay đổi trạng thái của các thiết bị. Từ đó, tính toán thời gian xử lý của một số tác vụ sau:

* Thời gian phản hồi của các nền tảng IoT khi hệ thống yêu cầu gửi dữ liệu.
* Thời gian cập nhật dữ liệu mới nhất vào hệ thống tích hợp.
* Thời gian xử lý của API của hệ thống tích hợp lấy dữ liệu mới nhất của thiết bị
* Thời gian xử lý của API của hệ thống tích hợp để thay đổi trạng thái thiết bị

Hình 22 biểu diễn kết quả đo đạc được (thời gian cần) nhằm thay đổi số lượng thiết bị.

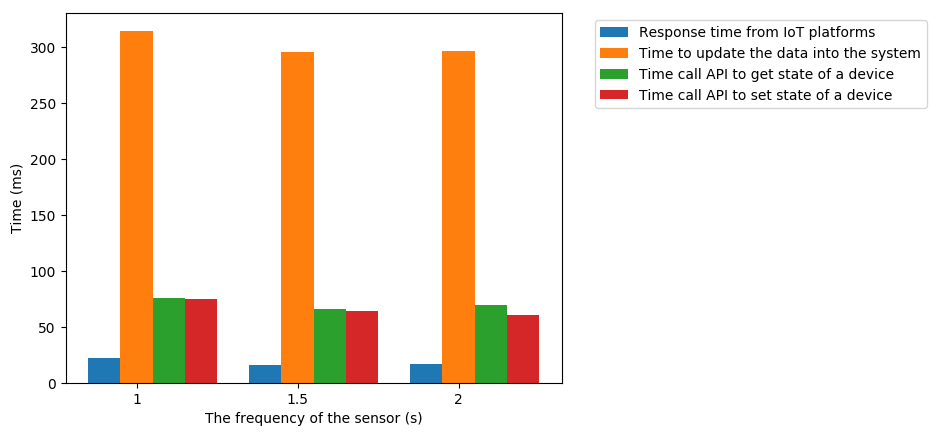


Hình 22: Biểu đồ trung bình kết quả khi thay đổi số lượng thiết bị

Từ hình trên ta có một số nhận xét sau:

* Thời gian phản hồi của các nền tảng IoT thay đổi khi tăng số lượng thiết bị. Điều này cũng dễ hiểu vì khi quản lý một số lượng thiết bị lớn đòi hỏi mất nhiều thời gian xử lý để trả trả về dữ liệu đúng yêu cầu.
* Thời gian cập nhật dữ liệu mới nhất vào hệ thống tích hợp thì tăng tuyến tính theo chiều tăng của số lượng thiết bị. Điều này chủ yếu là do dữ liệu lớn mất nhiều thời gian trên đường truyền và quá trình đọc ghi cơ sở dữ liệu là điều không tránh khỏi.
* Thời gian xử lý của API khi lấy thông tin về một thiết bị thì rất ổn định mặc dù thay đổi số lượng thiết bị. Điều này chứng tỏ hệ thống hoạt động rất ổn định và không ảnh hưởng quá nhiều đến người dùng khi số lượng thiết bị tăng lên đáng kể.
* Thời gian xử lý của API khi thay đổi trạng thái của một thiết bị cũng tăng khi số lượng thiết bị tăng. Nguyên nhân ở đây không phải do hệ thống xử lý kém hiệu quả khi số lượng thiết bị lớn mà chính là lý do đã nêu trên: các nền tảng IoT phản hồi, xử lý khá chậm khi số lượng thiết bị tăng.

Hình 23 biểu thị kết quả trung bình khi thay đổi chu kì gửi thông điệp dữ liệu của các thiết bị.



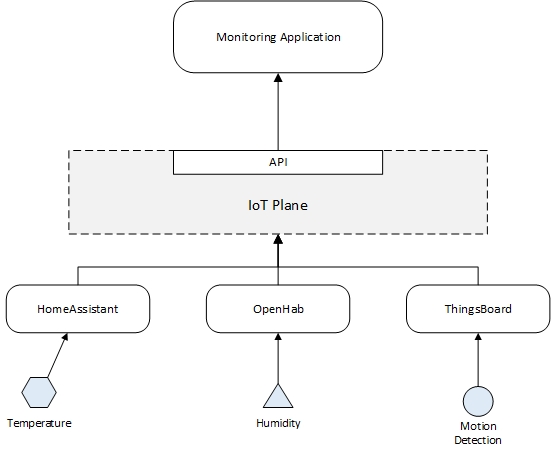
Hình 23: Biểu đồ trung bình kết quả khi thay đổi chu kỳ của các thiết bị

Từ hình trên ta có một số nhận xét sau:

* Thời gian phản hồi của các nền tảng IoT không thay đổi quá nhiều khi giữ nguyên số lượng thiết bị
* Thời gian cập nhật dữ liệu mới nhất vào hệ thống không thay đổi quá nhiều khi giữ nguyên số lượng thiết bị
* Thời gian xử lý của API khi lấy một thiết bị thì rất ổn định mặc dù thay đổi số lượng thiết bị. Điều này chứng tỏ hệ thống hoạt động rất ổn định và không ảnh hưởng quá nhiều đến người dùng khi thay đổi chu kỳ của các thiết bị.
* Thời gian xử lý của API khi thay đổi trạng thái của một thiết sẽ tăng nếu chu kỳ giảm. Nguyên nhân ở đây không phải do hệ thống xử lý kém hiệu quả khi số lượng thiết bị lớn mà chính là do các nền tảng IoT phải cập nhật dữ liệu rất nhiều khi chu kỳ thay đổi trạng thái của thiết bị nhỏ. Vì thế mà ta yêu cầu thay đổi trạng thái của thiết bị thì nền tảng IoT sẽ xử lý rất lâu.

## Xây dựng ứng dụng cảnh báo với dữ liệu thu thập từ ba nền tảng IoT

Ở phần này, chúng tôi sẽ xây dựng một ứng dụng đơn giản chạy trên phần mềm tích hợp nền tảng IoT để thử nghiêm. Mô hình ứng dụng như sau:

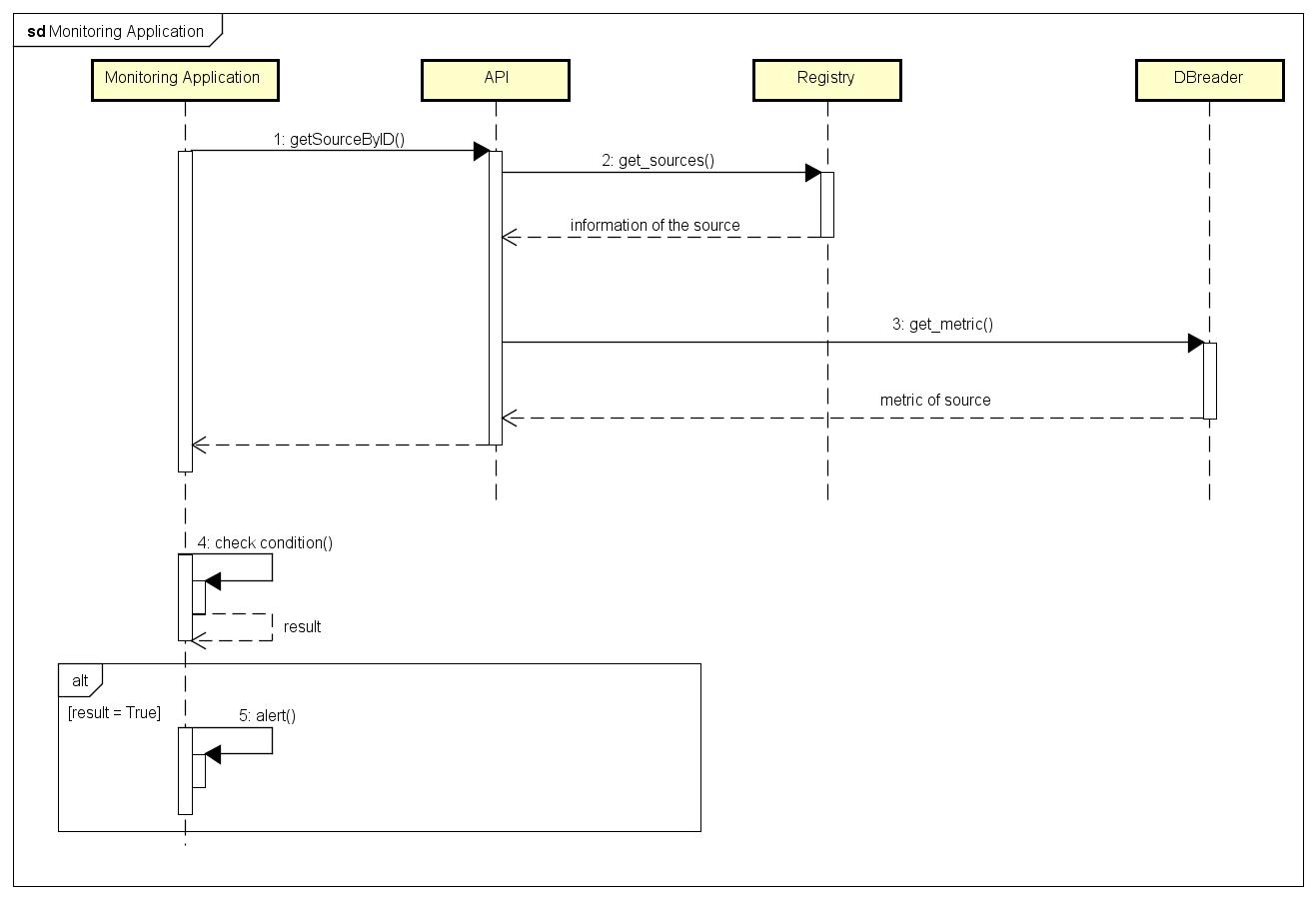


Hình 24: Mô hình xây dựng ứng dụng giám sát

Ứng dụng triển khai theo dõi dữ liệu nhiệt độ trên 3 nền tảng riêng biệt là Home-Assistant, OpenHab và ThingsBoard:

* Home-Assistant: theo dõi dữ liệu của cảm biến nhiệt độ
* OpenHab: theo dõi dữ liệu của cảm biến độ ẩm
* ThingsBoard: theo dõi dữ liệu của cảm biến chuyển động

Ngoài ra cơ sử dữ liệu được lưu trên đám mây Openstack sử dụng 2 API khác nhau để điều khiển (Nova và EC2 – giả lập môi trường đa đám mây).



Hình 25: Biểu đồ trình tự của ứng dụng giám sát

Ứng dụng sẽ phát ra cảnh báo âm thanh và thông báo trên màn hình giao diện nếu các cảm biến đang theo dõi trả về dữ liệu thoả mãn tất cả các điều kiện sau:

* Nhiệt độ > 35 độ C
* Độ ẩm < 50%
* Có chuyển động trong khu vực



Hình 26: Màn hình ứng dụng giám sát



Hình 27: Màn hình khi ứng dụng đưa ra cảnh báo

Như vậy với ứng dụng đã xây dựng đã chứng minh được rằng:

* Hệ thống tích hợp nền tảng của chúng tôi xây dựng hoàn toàn có thể thu thập, theo dõi dữ liệu từ nhiều nền tảng IoT khác nhau. Đây chỉ là một ứng dụng giám sát đơn giản nhưng với quy mô rộng hơn hệ thống hoàn toàn có thể thực hiện được các tác vụ phức tạp khi người dùng muốn thao tác cùng lúc trên nhiều nền tảng IoT cùng một lúc - điều tưởng chừng như không thể nếu như các nền tảng IoT hoạt động độc lập.
* API mà hệ thống cung cấp có thể giúp người dùng triển khai được các ứng dụng liên quan tùy thuộc vào nhu cầu. Đó chính là khả năng mở rộng của hệ thống khi các lập trình viên có thể nhanh chóng hiểu và sử dụng API một các dễ dàng.

Bảng 8: Kết quả khi thực hiện một số trường hợp kiểm thử với ứng dụng giám sát

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Dữ liệu hiện tại** | **Kết quả mong muốn** | **Kết quả thực tế** |
| 1 | Nhiệt độ 25 độ C (<35 độ C)  Độ ẩm 40% (<50%)  Có chuyển động | Không đưa ra cảnh báo | Đạt |
| 2 | Nhiệt độ 36 độ C (> 35 độ C)  Độ ẩm 52% (>50%)  Có chuyển động | Không đưa ra cảnh báo | Đạt |
| 3 | Nhiệt độ 36 độ C (> 35 độ C)  Độ ẩm 40% (< 50%)  Không có chuyển động | Không đưa ra cảnh báo | Đạt |
| 4 | Nhiệt độ 36 độ C (> 35 độ C)  Độ ẩm 44% (< 50%)  Có chuyển động | Đưa ra cảnh báo | Đạt |

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

IoT không phải là một cuộc cách mạng mới, mà là sự kết hợp và tiếp nối sau quá trình phát triển của hàng loạt các công nghệ như điện toán đám mây, công nghệ kết nối tốc độ cao, các công nghệ lập trình, các công nghệ phần cứng, v.v… Sự liên kết thông qua mạng và chia sẻ dữ liệu của các thiết bị có khả năng cảm thụ môi trường mang lại cho con người nhiều cơ hội cũng như thách thức trong vấn đề phát triển ứng dụng, tạo lập không gian thông minh, vấn đề mở rộng, kết hợp tài nguyên, quản lý, an toàn và bảo mật thông tin, bảo vệ quyền riêng tư, v.v… Ứng dụng của IoT trong thực tế là rất rõ ràng bao phủ mọi lĩnh vực của cuộc sống từ nhà thông minh, chăm sóc sức khỏe, giải trí, giao thông thông minh, … Mặc dù vậy, một trong những thách thức của IoT là tạo ra khả năng mở rộng các giải pháp có sẵn nhằm hình thành được các không gian thông minh rộng hơn, hoạt động hiệu quả hơn trong thực tế.

Trong đề tài này, tôi tập trung giải quyết một trong các vấn đề trong bài toán mở rộng các không gian thông minh: cung cấp giải pháp và thử nghiệm việc kết hợp các nền tảng IoT khác. Việc này giúp các nhà lập trình phát triển ứng dụng có khả năng tạo ra các sản phẩm hoạt động trong môi trường hỗn tạp các nguồn tài nguyên IoT. Đây chính là một trong những yếu tố quan trọng hỗ trợ bài toán mở rộng phạm vi ứng dụng IoT trong thực tế.

Để thực hiện mục tiêu này, chúng tôi đề xuất một số các giải pháp, cơ chế bao gồm: (1) xây dựng một mô hình dữ liệu thống nhất cho phép các nền tảng IoT có thể trao đổi các dữ liệu thông tin với nhau một cách dễ dàng. Mô hình dữ liệu này có khả năng tích hợp nhiều các nền tảng IoT khác nhau, với các cách định nghĩa cấu trúc, hình thái dữ liệu khác nhau. (2) Xây dựng các cơ chế tích hợp thông qua cách tiếp cận tạo các driver lập trình cho mỗi một nền tảng IoT và tạo lớp trừu tượng lập trình (API) cung cấp giao diện đồng bộ thống nhất cho tầng quản lý chung các nền tảng IoT. (4) Xây dựng thử nghiệm ứng dụng điều khiển đa nền tảng IoT (được triển khai trên các gateway trong môi trường điện toán sương mù). Cần nhắc lại rằng, phạm vi của đề tài liên quan tới việc liên kết các phần mềm và hệ thống và chưa cân nhắc tới các yếu tố kĩ thuật liên quan tới phần cứng và giao tiếp trong môi trường IoT.

Kết quả đạt được cụ thể của luận văn như sau:

* Hệ thống các phần mềm cho IoT: đã hoàn thành và thử nghiệm thành công hệ thông: Liên kết các nền tảng IoT triển khai trên các gateway trong môi trường điện toán sương mù phổ biến dưới một giao diện lập trình: có khả năng mở rộng hỗ trợ các platform khác tùy theo nhu cầu thực tế của bài toán ứng dụng. Đã đưa lên Github (<https://github.com/HPCC-Cloud-Computing/IoTPlane>) và được kiểm thử thành công dựa trên kịch bản và yêu cầu nghiệm thu đăng kí trong thuyết minh.
* Mô hình thiết bị thử nghiệm IoT: đã triển khai thử nghiệm mô hình IoT với các nền tảng IoT và thiết bị khác nhau tại phòng 609 thư viện Tạ Quang Bửu và thử nghiệm thành công.

Phần thực hiện của đề tài vẫn còn một số các nhược điểm bao gồm: chưa thử nghiệm nhiều các nền tảng IoT khác. Nguyên nhân cơ bản là do thời gian thực hiện còn hạn chế nên tôi thực hiện đã giới hạn phạm vi thử nghiệm nhằm hoàn thành các mục tiêu chính. Dựa trên kết quả thực hiện của luận văn này, tôi đề xuất một số hướng phát triển: cung cấp tính năng truy vấn dựa trên ngữ nghĩa, các module luật điều khiển việc gửi nhận dữ liệu nhằm tối ưu năng lượng, băng thông, giải thuật quản lý năng lượng thông minh.

# TÀI LIỆU THAM KHẢO

1. Acis Smart Home Solution, online at http://acis.com.vn/
2. Al-Fuqaha, Ala, et al. "Internet of things: A survey on enabling technologies, protocols, and applications." Communications Surveys & Tutorials, IEEE 17.4 (2015): 2347-2376.
3. Alam S, Chowdhury M, Noll J (2010) Senaas: An event-driven sensor virtualization approach for internet of things cloud. In: NESEA. pp 1–6. doi:10.1109/NESEA.2010.5678060
4. Tran, Hai Anh, Quynh Thu Ngo, and Huy Hoang Pham. "An application for diagnosing lung diseases on Android phone." Proceedings of the Sixth International Symposium on Information and Communication Technology. ACM, 2015.
5. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the Internet of Things. In: MCC Workshop on Mobile Cloud Computing. ACM, New York, NY, USA. pp 13–16. http://doi.acm.org/10.1145/2342509. 2342513
6. Casati F, Daniel F, Dantchev G, Eriksson J, Finne N, Karnouskos S, et al. (2012) Towards business processes orchestrating the physical enterprise with wireless sensor networks. In: ICSE’12. IEEE, Zurich, Switzerland. pp 1357–1360. doi:10.1109/ICSE.2012.6227080
7. Chun BG, Ihm S, Maniatis P, Naik M, Patti A (2011) Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the Sixth Conference on Computer Systems. ACM, New York, NY, USA. pp 301–314. http://doi.acm.org/10.1145/1966445.1966473
8. Chương trình Biến đổi khí hậu của Bộ tài nguyên và môi trường online at http://www.khcn-bdkh.vn/news/68-Khung-Chuong-trinh-khoa-hoc-va-cong-nghe-cap-quoc-gia-Khoa-hoc-va-cong-nghe-ung-pho-voi-bien-doi-khi-hau-quan-ly-tai-nguyen-va-moi-truong-giai-doan-2016-2020.html
9. Ciciriello P, Mottola L, Picco GP (2006) Building virtual sensors and actuators over logical neighborhoods. In: Proceedings of the International Workshop on Middleware for Sensor Networks. ACM, New York, NY, USA. pp 19–24. http://doi.acm.org/10.1145/1176866.1176870
10. Cao, Tien-Dung, and Hong-Linh Truong. "Analyzing and Conceptualizing Monitoring and Analytics as a Service for Grain Warehouses."
11. D. Locke, “MQ telemetry transport (MQTT) v3. 1 protocol specification,” IBM developerWorks, Markham, ON, Canada, Tech. Lib., 2010. [Online]. Available: Http://Www.Ibm.Com/Developerworks/ Webservices/Library/Ws-Mqtt/Index.Html
12. Davidson, Emily A (Softchoice Advisor): The Software-Defined-DataCenter (SDDC): Concept Or Reality? http://tinyurl.com/omhmbfv. [Online; accessed Jan-’15]
13. De Souza LMS, Spiess P, Guinard D, Köhler M, Karnouskos S, Savio D (2008) Socrades: A web service based shop floor integration infrastructure. In: The Internet of Things. pp 50–67. http://link.springer.com/chapter/10. 1007%2F978-3-540-78731-0\_4
14. Distefano S, Merlino G, Puliafito A (2012) Sensing and actuation as a service: a new development for clouds. In: NCA. pp 272–275. doi:10.1109/NCA.2012.38
15. E. Savitz, “Gartner: 10 Critical Tech Trends For The Next Five Year” online at ”http://www.forbes.com/sites/ericsavitz/2012/10/22/gartner-10-critical-tech-trends-for-the-next-five-years/#78666ced4c6f
16. Fujitsu and FPT Implement Smart Agriculture in Vietnam, online at https://www.fpt.com.vn/en/newsroom/detail/fpt-and-fujitsu-officially-open-fujitsu-fpt-akisai-farm-and-vegetable-factory
17. Giải pháp Công nghệ Giám sát Magiwan, online at http://www.magiwan.com/
18. Giải pháp V-tracking Viettel , online at http://quanlyxe.viettel.vn/Account/Login.aspx?basepath=/Supervision.aspx
19. Giải pháp giám sát Bình Anh, online at http://binhanh.vn/
20. Giải pháp giám sát hành trình Vcomsat, online at http://giamsathanhtrinh.vn/
21. Green Leap solution for gardeners, online at http://greenleap.vn/
22. 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. IEEE, 2014, pp. 124–127.
23. Hackanoi IoT research group, online at http://hackanoi.com/
24. Hassan MM, Song B, Huh EN (2009) A framework of sensor-cloudintegration opportunities and challenges. In: ICUIMC. ACM, New York, NY, USA. pp 618–626. doi:10.1145/1516241.1516350
25. Home Assistant - https://www.home-assistant.io/
26. ITI-T, Internet of Things Global Standards Initiative, online at http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx
27. IoT Vietnam Alliance, online at http://iotvietnam.vn/
28. IoT platform for Shrimp farming at Hoa Hiep hamlet (Long Hoa commune, Can Gio district, HCM city), online at http://aquanetviet.com/post/125502179525/%E1%BB%A9ng-d%E1%BB%A5ng-internet-of-things-iot-cho-n%C3%B4ng-nghi%E1%BB%87p
29. J. Kim and J.-W. Lee, “Openiot: An open service framework for the internet of things,” in Internet of Things (WF-IoT), 2014 IEEE World Forum on, March 2014, pp. 89–93.
30. J. van der Ham, J. St´eger, S. Laki, Y. Kryftis, V. Maglaris, and C. de Laat, “The novi information models,” Future Generation Computer Systems, vol. 42, pp. 64–73, 2015.
31. Koldehofe B, Dürr F, Tariq MA, Rothermel K (2012) The power of software-defined networking: line-rate content-based routing using OpenFlow. In: Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing. ACM, New York, NY, USA. pp 3:1–3:6. http://doi.acm.org/10.1145/2405178.2405181
32. Konexy platform, online at http://konexy.com/
33. Kovatsch M, Lanter M, Duquennoy S (2012) Actinium: A restful runtime container for scriptable internet of things applications. In: Internet of Things. pp 135–142. doi:10.1109/IOT.2012.6402315
34. Kumar K, Lu YH (2010) Cloud computing for mobile users: Can offloading computation save energy? Computer 43(4):51–6
35. Larios, V. M., et al. "IEEE-GDL CCD Smart Buildings Introduction." IEEE Guadalajara Physical Infrastructure Working Group for Smart Cities (2014).
36. Lumi Smart Home Solution, online at http://lumi.vn/nha-thong-minh?utm\_source=Google&utm\_medium=cpc&utm\_campaign=INSO\_GA\_Lumi\_160223&gclid=CMjrhs6h8MsCFYmTvQod2MME8w
37. METIS, Mobile and wireless communication Enablers for the Twenty-twenty (2020) Information Society, online at https://www.metis2020.com/
38. Madden SR, Franklin MJ, Hellerstein JM, Hong W (2005) TinyDB: an acquisitional query processing system for sensor networks. ACM Trans Database Syst (TODS) 30(1):122–73
39. Middleton, Peter, Peter Kjeldsen, and Jim Tully. "Forecast: The internet of things, worldwide, 2013." Gartner Research (2013).
40. MimosaTek smart agriculture solutions, online at https://mimosatek.com/
41. Mineraud, Julien, et al. "A gap analysis of Internet-of-Things platforms." arXiv preprint arXiv:1502.01181 (2015).
42. Mottola L, Pathak A, Bakshi A, Prasanna VK, Picco GP (2007) Enabling scope-based interactions in sensor network macroprogramming. In: MASS 2007. IEEE Computer Society, Pisa, Italy. pp 1–9. http://dx.doi.org/ 10.1109/MOBHOC.2007.4428655
43. Nguyen, Binh Minh, Huan Phan, Duong Quang Ha, and Giang Nguyen. "An Information-centric Approach for Slice Monitoring from Edge Devices to Clouds." Procedia computer science 130 (2018): 326-335.
44. NetJSON: data interchange format for networks, online at http://netjson.org/rfc.html
45. Open IoT platfrom – IOP (an innovative ecosystem platform for startup), online at http://dtt.vn/?p=6928&lang=vi
46. OpenDaylight: Open Source SDN Platform, online at https://www.opendaylight.org/
47. OpenHab - https://www.openhab.org/
48. P. Gomes, E. Cavalcante, T. Rodrigues, T. Batista, F. C. Delicato, and P. F. Pires, “A federated discovery service for the internet of things,” in Proceedings of the 2Nd Workshop on Middleware for Context-Aware Applications in the IoT, ser. M4IoT 2015. New York, NY, USA: ACM, 2015, pp. 25–30.
49. P. P. Jayaraman, D. Georgakopoulos, M. Zhang, and R. Ranjan, “Discovery-Driven Service Oriented IoT Architecture,” in IEEE International Conference on Collaboration and Internet Computing (CIC 2015), October 2015.
50. R. Inam, A. Karapantelakis, K. Vandikas, L. Mokrushin, A. V. Feljan, and E. Fersman, “Towards automated service-oriented lifecycle management for 5g networks,” in Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on. IEEE, 2015, pp. 1–8.
51. Research and Development of Realtime Positioning System for "cm-level of accuracy" Demanding Applications, NAVIS-HUST, online at http://navis.hust.edu.vn/index.php/projects
52. S. Datta, C. Bonnet, and N. Nikaein, “An iot gateway centric architecture to provide novel m2m services,” in Internet of Things (WF-IoT), 2014 IEEE World Forum on, March 2014, pp. 514–519.
53. 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.
54. 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.
55. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for vm-based cloudlets in mobile computing. Pervasive Comput 8(4):14–23. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5280678
56. Sixsq NuvlaBox. http://sixsq.com/products/nuvlabox.html. [Online; accessed Jan-’15]
57. Smart BK Traffic, online at http://traffic.hcmut.edu.vn/index.html
58. Smart Home BKAV, online at http://www.smarthome.com.vn/
59. Soldatos J, Serrano M, Hauswirth M (2012) Convergence of utility computing with the internet-of-things. In: IMIS. pp 874–9. doi:10.1109/IMIS.2012.135
60. T. S. L´opez, A. Brintrup, M.-A. Isenberg, and J. Mansfeld, Architecting the Internet of Things. Berlin, Heidelberg: Springer Berlin Heidelberg,2011, ch. Resource Management in the Internet of Things: Clustering, Synchronisation and Software Agents, pp. 159–193.
61. T.H. Ta, M. Pini, and L. L. Presti, "Combined GPS L1C/A and L2C Signal Acquisition Architectures Leveraging Differential Combination", IEEE Transactions on Aerospace and Electronic Systems, vol. 50, Issue 4, pp. 2914 - 2924, October 2014
62. TIEN-DUNG, C. A. O., et al. "MARSA: A Marketplace for Realtime Human-Sensing Data."
63. Thereska E, Ballani H, O’Shea G, Karagiannis T, Rowstron A, Talpey T, et al. (2013) IoTFlow: A software-defined storage architecture. In: SOSP. ACM, Farmington, PA, USA. pp 182–96. http://doi.acm.org/10.1145/2517349. 2522723
64. ThingsBoard - Open-source IoT Platform - https://thingsboard.io/
65. Trí Nam TMS – Transportation Management System, online at http://hca.org.vn/Upload/Images/Original/2015/VIO%202015/THAM%20LUAN/TRI%20NAM.pdf
66. Tung Hai Ta, N. Shivaramaiah, A. Dempster, L. L. Presti, “Significance of Cell Correlations in GNSS Matched Filter Acquisition Engines”, IEEE Transactions on Aerospace and Electronic Systems, vol. 48, Issue 2, pp. 1264 - 1286, April 2012.
67. U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S—A publish/subscribe protocol for wireless sensor networks,” in Proc. 3rd Int. Conf. COMSWARE, 2008, pp. 791–798.
68. Ubisen platform, online at https://www.ubisen.com/
69. Vietnam-Japanese research cooperation in smart agriculture, online at http://aquanetviet.com/post/125502179525/%E1%BB%A9ng-d%E1%BB%A5ng-internet-of-things-iot-cho-n%C3%B4ng-nghi%E1%BB%87p
70. Weave Net - Weaving Containers into Applications, online at https://github.com/weaveworks/weave
71. Y. Benazzouz, C. Munilla, O. Gunalp, M. Gallissot, and L. Gurgen, “Sharing user iot devices in the cloud,” in Internet of Things (WF-IoT), 2014 IEEE World Forum on, March 2014, pp. 373–374.
72. YANG Data Modeling Language, online at http://netconfcentral.org/yang\_docs
73. Yuriyama M, Kushida T (2010) Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing. In: NBiS’10. pp 1–8. doi:10.1109/NBiS.2010.32
74. Zhang, Huijuan, and Chao Meng. "A multi-dimensional ontology-based IoT resource model." Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on. IEEE, 2014.