

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

\*\*\*\*\*

ĐỒ ÁN  
**TỐT NGHIỆP ĐẠI HỌC**  
NGÀNH CÔNG NGHỆ THÔNG TIN

**Xây dựng ngôn ngữ truy vấn ngữ nghĩa  
trong lĩnh vực IoT**

Sinh viên thực hiện: **Đỗ Chí Thành**

Lớp CNTT2.4-K59

Giáo viên hướng dẫn: **TS. Vũ Văn Thiệu**

**Hà Nội, 05/2019**

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin sinh viên

Họ và tên sinh viên: Đỗ Chí Thành

Điện thoại liên lạc: 0326118018 Email: dothanhwork2017@gmail.com

Lớp : CNTT 2.4 K59 Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại: Đại học Bách Khoa Hà Nội

Thời gian làm đồ án: Từ 11/02/2019 đến 24/05/2019

2. Mục đích nội dung của ĐATN

Xây dựng ngôn ngữ truy vấn ngữ nghĩa cho hệ thống IoT.

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu tổng quan về ngôn ngữ truy vấn ngữ nghĩa.
- Xây dựng cấu trúc dữ liệu.
- Xây dựng cú pháp của ngôn ngữ
- Xây dựng cơ chế truy vấn ngữ nghĩa

4. Lời cam đoan của sinh viên

Tôi - Đỗ Chí Thành - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của TS. Vũ Văn Thiệu. Các kết quả nêu trong ĐATN là trung thực, không phải sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày 15 tháng 05 năm 2019

Tác giả ĐATN

Đỗ Chí Thành

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

Hà Nội, ngày 15 tháng 05 năm 2019

Giáo viên hướng dẫn

TS. Vũ Văn Thiệu

# Mở đầu

Trong xã hội hiện đại, Internet phát triển và sắp tới với sự phổ biến của mạng viễn thông thế hệ 5 (5G) sẽ khiến lĩnh vực IoT phát triển mạnh mẽ hơn. Trong thực tế, hiện tại, IoT đã rất phát triển, theo Gartner, thế giới sẽ có khoảng hơn 20 tỷ thiết bị IoT vào năm 2020 [1], và năm 2017 đã có khoảng 450 IoT Platform theo công ty nghiên cứu thị trường IoT-Analytics của Đức [2]. Một số IoT Platform đáng được dùng phổ biến như Google Cloud Platform, Salesforce IoT Cloud, IBM Watson IoT, AWS IoT. Ngoài ra, các cộng đồng mã nguồn mở cũng có một số IoT Platform phổ biến như OpenHAB, HomeAssistant, ThingsBoard, ... Sự đa dạng của các IoT platform dẫn tới sự không thống nhất của các dữ liệu được tạo ra từ các platform, do mỗi IoT platform có một cách biểu diễn dữ liệu khác nhau. Do đó, tôi và các bạn Đinh Hữu Hải Quân, Vũ Ngọc Hoàn với sự hướng dẫn của anh Hà Quang Dương (học viên cao học) và của thầy Nguyễn Bình Minh đã xây dựng hệ thống cross-platform IoT, đóng vai trò là một driver, giúp chuẩn hóa dữ liệu của các IoT platform khác nhau về một dạng thống nhất. Hệ thống cũng có tính khả mở, cho phép các IoT platform khác có thể tham gia vào hệ thống một cách dễ dàng.

Sau khi đã có được dữ liệu theo một cấu trúc thống nhất, để sử dụng dữ liệu thu thập được, đặt ra một bài toán là tìm kiếm dữ liệu. Phương pháp truyền thống là tìm kiếm theo từ khóa xuất hiện, ví dụ câu tìm kiếm "Tìm các đèn trong phòng 609 thư viện Tạ Quang Bửu". Tuy nhiên, phương pháp tìm kiếm theo từ khóa có một nhược điểm là không hiểu được câu tìm kiếm, mà chỉ liệt kê các tài liệu chứa các từ khóa trong câu tìm kiếm. Với ví dụ trên, kết quả mong muốn là tìm kiếm các đèn trong phòng 609 thư viện Tạ Quang Bửu, nhưng kết quả trả về sẽ bao gồm cả các tài liệu khác về phòng 609 thư viện Tạ Quang Bửu như điều hòa, tivi, các thiết bị trong phòng 609, không phải là kết quả mong muốn. Do đó, tôi thực hiện đề tài "Xây dựng ngôn ngữ truy vấn ngữ nghĩa cho hệ thống cross-platform IoT". Mặc dù ngôn ngữ truy vấn ngữ nghĩa này được thiết kế cho hệ thống cross-platform IoT, tuy nhiên, ta có thể áp dụng với các hệ thống IoT khác một cách tương tự.

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau:

**Chương 1** sẽ giới thiệu cơ sở lý thuyết của IoT và truy vấn ngữ nghĩa, đồng thời nêu lên hướng giải quyết của đề tài. **Chương 2** sẽ trình bày việc xây dựng hệ thống IoT và thiết kế ngôn ngữ truy vấn ngữ nghĩa. Cuối cùng, **chương 3** sẽ nêu lên những kết luận và hướng phát triển đề tài.

## **Lời cảm ơn**

Trước hết, tôi xin chân thành cảm ơn TS. Vũ Văn Thiệu, bộ môn Khoa học máy tính, viện Công nghệ thông tin và truyền thông, trường đại học Bách Khoa Hà nội là người đã tận tình giúp đỡ tôi trong suốt thời gian làm đề án này.

Đồng thời, tôi xin chân thành cảm ơn TS. Nguyễn Bình Minh, bộ môn Hệ thống thông tin và truyền thông, đại học Bách Khoa Hà Nội cùng bạn Đinh Hữu Hải Quân và bạn Vũ Ngọc Hoàn đã giúp đỡ tôi xây dựng các driver cho các platform.

Tôi cũng xin cảm ơn các thầy, cô giáo ở viện Công nghệ thông tin và truyền thông và các thầy cô giáo trong trường ĐHBK Hà Nội đã giảng dạy, giúp đỡ tôi trong quá trình học tập tại trường, cho tôi nền tảng kiến thức chuyên môn vững chắc để hoàn thành tốt đề án này.

Tuy đã cố gắng rất nhiều để hoàn thành tốt đề án, nhưng do năng lực bản thân còn hạn chế nên không thể không có những thiếu sót trong đề án này. Kính mong nhận được sự góp ý của thầy cô và các bạn.

# Mục lục

|  |           |
|--|-----------|
| <b>Mở đầu</b>  | <b>2</b>  |
| <b>Lời cảm ơn</b>  | <b>4</b>  |
| <b>1 Các cơ sở lý thuyết và hướng giải quyết đề tài</b>                | <b>9</b>  |
| 1.1 Giới thiệu về IoT . . . . .  | 9         |
| 1.1.1 Lịch sử của internet of things . . . . .                         | 10        |
| 1.1.2 Lợi ích của internet of things . . . . .                         | 11        |
| 1.1.3 Các công nghệ liên quan tới Internet of things . . . . .         | 12        |
| 1.1.4 Bảo mật và an toàn trong IoT . . . . .                           | 16        |
| 1.1.5 Các dự đoán về IoT . . . . .                                     | 18        |
| 1.2 Giới thiệu về truy vấn ngữ nghĩa . . . . .                         | 18        |
| 1.2.1 Ngôn ngữ truy vấn ngữ nghĩa . . . . .                            | 18        |
| 1.2.2 Ontology . . . . .   | 18        |
| 1.3 Nhiệm vụ của đề tài . . . . .                                      | 28        |
| <b>2 Xây dựng hệ thống IoT và thiết kế ngôn ngữ truy vấn ngữ nghĩa</b> | <b>29</b> |
| 2.1 Xây dựng hệ thống IoT . . . . .                                    | 29        |

|          |   |           |
|----------|---|-----------|
| 2.1.1    | Phần cứng . . . . .   | 29        |
| 2.1.2    | Các giao thức sử dụng . . . . .                             | 30        |
| 2.1.3    | Các IoT Platform sử dụng . . . . .                          | 33        |
| 2.2      | Thiết kế ngôn ngữ truy vấn cho IoT . . . . .                | 36        |
| 2.2.1    | Xây dựng ontology . . . . .                                 | 36        |
| 2.2.2    | Cơ chế ánh xạ các dữ liệu trong IoT theo ontology . . . . . | 47        |
| 2.2.3    | Cơ chế truy vấn ngữ nghĩa . . . . .                         | 48        |
| 2.2.4    | Tập các API . . . . .                                       | 51        |
| <b>3</b> | <b>Kết luận, hướng phát triển</b>                           | <b>54</b> |
| <b>4</b> | <b>Tài liệu tham khảo</b>                                   | <b>55</b> |

# Danh sách hình vẽ

|      |  |    |
|------|--|----|
| 1.1  | Kiến trúc hệ thống IoT . . . . .                                 | 14 |
| 1.2  | IoT Platform . . . . .   | 16 |
| 1.3  | Một ví dụ về ontology . . . . .                                  | 21 |
| 2.1  | Hai hộp đại diện cho hai "vật" . . . . .                         | 30 |
| 2.2  | Giao thức truyền tin MQTT . . . . .                              | 31 |
| 2.3  | Mô hình triển khai phần cứng . . . . .                           | 32 |
| 2.4  | Triển khai phần cứng trên phòng 609 Thư viện TQB . . . . .       | 33 |
| 2.5  | Giao diện quản lý thiết bị của platform OpenHAB . . . . .        | 34 |
| 2.6  | Giao diện quản lý thiết bị của platform Home Assistant . . . . . | 35 |
| 2.7  | Giao diện quản lý thiết bị của platform Thingsboard . . . . .    | 35 |
| 2.8  | SSN ontology . . . . .   | 37 |
| 2.9  | IoT-A ontology . . . . .   | 38 |
| 2.10 | IoT-Lite ontology . . . . .                                      | 39 |
| 2.11 | Thực thể smartcontext . . . . .                                  | 45 |
| 2.12 | Thực thể platform . . . . .                                      | 45 |
| 2.13 | Thực thể Source . . . . .  | 45 |

|   |    |
|---|----|
| 2.14 Thực thể Metric . . . . .                                  | 46 |
| 2.15 Thực thể Data Point . . . . .                              | 46 |
| 2.16 ontology cho lĩnh vực nhà thông minh . . . . .             | 47 |
| 2.17 Mô hình ánh xạ dữ liệu về một chuẩn của ontology . . . . . | 48 |
| 2.18 Mô hình ngôn ngữ - statement . . . . .                     | 49 |
| 2.19 Mô hình ngôn ngữ - condition . . . . .                     | 49 |
| 2.20 Mô hình ngôn ngữ - action . . . . .                        | 49 |
| 2.21 Mô hình ngôn ngữ - expression . . . . .                    | 50 |
| 2.22 Mô hình ngôn ngữ - term . . . . .                          | 50 |
| 2.23 Mô hình ngôn ngữ -const . . . . .                          | 50 |



# Chương 1

## Các cơ sở lý thuyết và hướng giải quyết đề tài

### 1.1 Giới thiệu về IoT

Internet vạn vật (Internet of Things - IoT) hay mạng lưới vật kết nối, trong đó, các thiết bị, phương tiện, nhà cửa, ... được nhúng các thiết bị điện tử, phần mềm, cảm biến cùng khả năng kết nối tới mạng máy tính giúp cho các thiết bị này có thể thu thập và gửi dữ liệu. Hệ thống IoT cho phép vật được cảm nhận hoặc được điều khiển từ xa thông qua hạ tầng mạng hiện hữu, tạo cơ hội cho thế giới thực được tích hợp trực tiếp hơn vào hệ thống điện toán, hệ quả là hiệu năng, độ tin cậy và lợi ích kinh tế được tăng cường bên cạnh việc giảm thiểu sự can dự của con người. Mục đích của Internet vạn vật là tạo ra các liên kết Máy-Máy (Machine-to-Machine), được xem như là thông minh. Các thiết bị thu thập dữ liệu hữu ích rồi sau đó tự động truyền dữ liệu sang các thiết bị khác. Các ví dụ về IoT có thể kể đến như tự động bật tắt đèn, hệ thống thông gió, hệ thống điều hòa nhiệt độ, ...

### 1.1.1 Lịch sử của internet of things

Năm 1999, Kevin Ashton, đồng sáng lập của trung tâm Auto-ID tại MIT, lần đầu tiên đề cập tới IoT trong bài trình bày của mình tại Procter & Gamble (P&G). Với mong muốn thu hút sự chú ý của các nhà quản lý của P&G với công nghệ RFID (radio frequency ID), Ashton gọi bài trình bày của ông là "Internet of Things" để phù hợp với xu hướng đang nổi lúc bấy giờ là Internet.

Đến nay, IoT khẳng định được bước tiến của mình nhờ sự hội tụ của nhiều công nghệ, bao gồm truyền tải vô tuyến hiện diện đầy đặc, phân tích dữ liệu thời gian thực, học máy, cảm biến hàng hóa, và hệ thống nhúng. Điều này có nghĩa là tất cả các dạng thức của hệ thống nhúng cổ điển, như mạng cảm biến không dây, hệ thống điều khiển, tự động hóa (bao gồm nhà thông minh và tự động hóa công trình),... đều đóng góp vào việc vận hành Internet Vạn Vật (IoT).

Mặc dù Ashton là người đầu tiên đề cập tới cụm từ "Internet of Things", ý tưởng về kết nối các thiết bị đã xuất hiện từ khoảng những năm 1970, dưới cái tên Internet nhúng (embedded internet) và tính toán rộng khắp (pervasive computing). Thiết bị được kết nối internet đầu tiên trên thế giới là một máy bán Coca-Cola tại đại học Carnegie Mellon từ đầu những năm 1980, có khả năng báo cáo kiểm kho và báo cáo độ lạnh của những chai nước mới bỏ vào máy. Bản mô tả sơ khai năm 1991 về điện toán phổ quát (ubiquitous computing) của Mark Weiser, "Máy tính thế kỷ XXI", cũng như những báo cáo về tầm nhìn đương đại của IoT từ các viện khoa học UbiComp và PerCom. Năm 1994, Reza Raji mô tả khái niệm này trên tờ IEEE Spectrum là "chuyển các gói dữ liệu nhỏ tới tập hợp các nút mạng lớn, để tích hợp và tự động hóa mọi thứ từ các thiết bị gia dụng với cả một nhà máy sản xuất". Giữa năm 1993 và 1996 một số công ty đề xuất giải pháp như At Work của Microsoft hay NEST của Novell. Tuy nhiên, chỉ đến năm 1999, lĩnh vực IoT mới bắt đầu có sự chuyển biến. Bill Joy mừng tượng tới phương thức truyền tải thiết bị-tới-thiết bị (D2D) ở một phần trong bộ khung "Six Webs" của ông, được ông diễn thuyết tại Diễn đàn Kinh tế Thế giới ở Davos năm 1999. Khái niệm Internet Vạn Vật trở nên phổ biến trong năm 1999 nhờ trung tâm Auto-ID thuộc viện công nghệ Massachusetts và các xuất bản phẩm

phân tích thị trường có liên quan. Radio-frequency identification (RFID) được Kevin Ashton (một trong những sáng lập gia của trung tâm Auto-ID) cho rằng là điều kiện tiên quyết cho IoT. Ashton thích dùng cụm từ "Internet cho các vật" hơn. Nếu tất cả các đồ vật và con người trong cuộc sống thường ngày đều được mang định danh, máy tính có thể điều khiển và can thiệp vào. Bên cạnh việc sử dụng RFID, các nhãn của các vật cũng có thể thu được thông qua các công nghệ như giao tiếp tầm gần (near field communication), barcode, QR code và digital watermarking.

IoT được phát triển từ việc các máy tính giao tiếp với nhau - M2M (Machine-to-machine communication) thông qua một mạng máy tính mà không cần có sự can thiệp của con người. Trong M2M, các thiết bị được kết nối tới các đám mây tính toán (cloud computing), các đám mây sẽ thực hiện thu thập và quản lý dữ liệu. Đưa M2M đi xa hơn nữa, IoT là một mạng cảm biến với hàng tỷ thiết bị thông minh kết nối con người, các hệ thống và các ứng dụng để thu thập và chia sẻ dữ liệu. M2M chính là nền tảng cho sự kết nối để đạt được IoT.

IoT cũng là sự mở rộng tự nhiên của SCADA (supervisory control and data acquisition - giám sát điều khiển và thu thập dữ liệu), một loại chương trình ứng dụng để thực hiện điều khiển, thu thập dữ liệu thời gian thực từ xa để điều khiển thiết bị. Các hệ thống SCADA bao gồm phần cứng và phần mềm. Thành phần phần cứng thu thập và chuyển dữ liệu đến máy tính có cài phần mềm SCADA, nơi dữ liệu sẽ được xử lý và hiển thị theo thời gian. Các thế hệ cuối của các hệ thống SCADA chính là thế hệ đầu tiên của hệ thống IoT. Khái niệm về hệ sinh thái IoT, lại không thực sự xuất hiện cho tới tận giữa năm 2010, khi chính phủ Trung Quốc nói rằng muốn IoT là mục tiêu chiến lược của mình trong kế hoạch năm năm tiếp theo.

### 1.1.2 Lợi ích của internet of things

Khi IoT được diễn ra phổ biến, sẽ dẫn tới quá trình tự động hóa đại trà trên nhiều lĩnh vực. Theo Garner, Inc. (một công ty nghiên cứu và tư vấn về công nghệ), sẽ có khoảng 26 tỷ thiết bị IoT vào năm 2020. Theo một cuộc khảo sát và nghiên cứu gần đây được thực hiện bởi dự án Internet Pew Research, một phần lớn các chuyên gia công nghệ đã

hưởng ứng tham gia sử dụng Internet of Things với 83% đồng ý quan điểm cho rằng Internet of Things sẽ có tác động rộng rãi và mang lại lợi ích đến năm 2025. Một số lợi ích của IoT với các doanh nghiệp có thể kể đến như sử dụng tài nguyên hiệu quả hơn, giảm công sức của con người đồng thời giảm chi phí và mang lại hoạt động hiệu quả hơn, nâng cao trải nghiệm người dùng, tiết kiệm thời gian và tiền bạc, ... Ngoài ra, đối với khách hàng, người sử dụng, những ngôi nhà thông minh có thể điều khiển từ xa, có các hệ thống điều khiển tự động giúp thuận tiện hơn cho người dùng; các thiết bị IoT có thể đeo được trên cơ thể con người giúp thu thập dữ liệu về con người, phân tích các dữ liệu đó và đưa ra các hỗ trợ cho con người ví dụ như trong các trường hợp y tế khẩn cấp,...

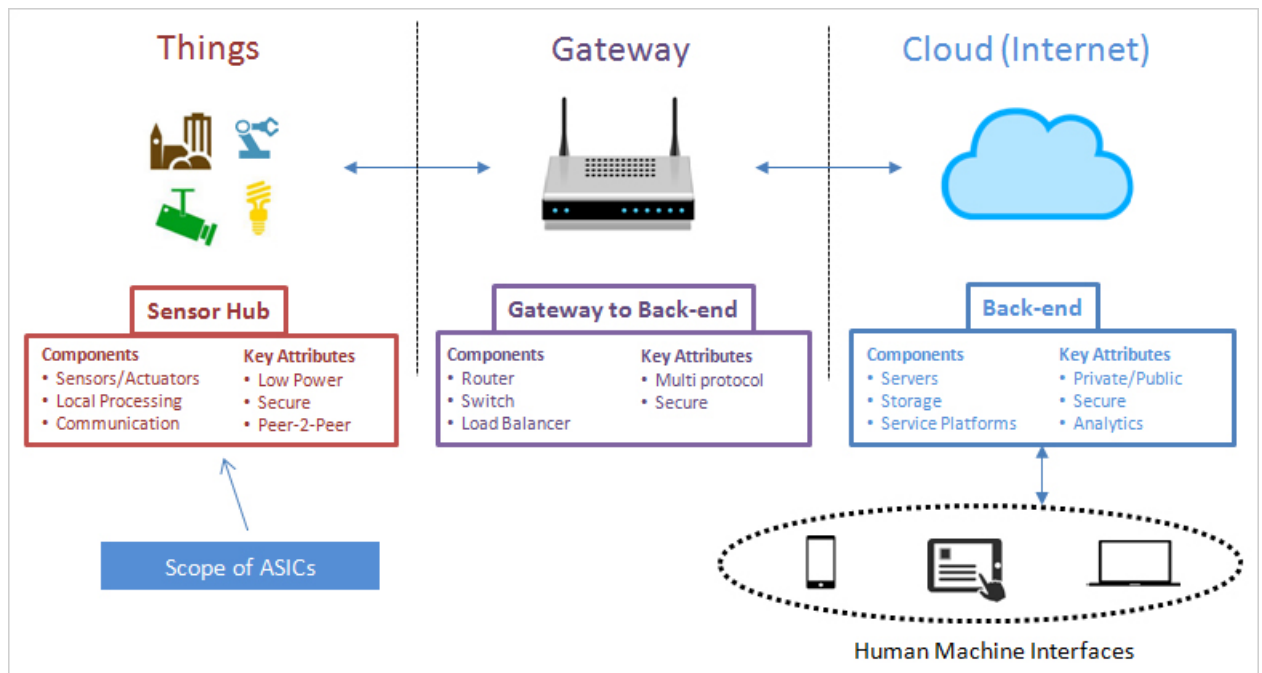
### 1.1.3 Các công nghệ liên quan tới Internet of things

#### 1.1.3.1 Phần cứng

Thành phần phần cứng cơ bản nhất của một hệ thống IoT là các cảm biến. Một cách khái quát, một cảm biến là một thiết bị có khả năng phát hiện những thay đổi trong môi trường. Nếu hoạt động độc lập, một cảm biến sẽ là vô dụng, tuy nhiên, khi chúng ta dùng chúng trong một hệ thống điện tử, cảm biến đóng vai trò chính yếu. Một cảm biến có thể đo đạc một hiện tượng vật lý (ví dụ như nhiệt độ, độ ẩm, áp suất, ...) và chuyển chúng thành các tín hiệu điện. Một cảm biến tốt cần phải có ba tính chất: Nhạy bén với các hiện tượng vật lý mà chúng đo đạc, không nhạy với các hiện tượng vật lý khác, và không làm thay đổi các giá trị đo trong quá trình đo đạc. Một số cảm biến phổ biến ngày nay nên có thể kể đến như cảm biến nhiệt độ, độ ẩm, ánh sáng, gia tốc, ...

Để đọc các giá trị được tạo ra bởi các cảm biến, các cảm biến thường được kết nối tới một bảng mạch vi xử lý. Một số mạch vi xử lý như Arduino, Raspberry Pi, Intel Galileo, ... trong đó Arduino được sử dụng phổ biến nhất bởi cả các chuyên gia lẫn những người mới bắt đầu. Arduino là một board mạch vi xử lý được sinh ra tại thị trấn Ivrea ở Ý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được

thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM Atmel 32-bit. Những Model hiện tại được trang bị gồm 1 cổng giao tiếp USB, 6 chân đầu vào analog, 14 chân I/O kỹ thuật số tương thích với nhiều board mở rộng khác nhau. Được giới thiệu vào năm 2005, Những nhà thiết kế của Arduino cố gắng mang đến một phương thức dễ dàng, không tốn kém cho những người yêu thích, sinh viên và giới chuyên nghiệp để tạo ra những thiết bị có khả năng tương tác với môi trường thông qua các cảm biến và cơ chế hành động. Những ví dụ phổ biến cho những người yêu thích mới bắt đầu bao gồm các robot đơn giản, điều khiển nhiệt độ và phát hiện chuyển động. Đi cùng với nó là một môi trường phát triển tích hợp (IDE) chạy trên các máy tính cá nhân thông thường và cho phép người dùng viết các chương trình cho Arduino bằng ngôn ngữ C hoặc C++. Thông tin thiết kế phần cứng được cung cấp công khai để những ai muốn tự làm một mạch Arduino bằng tay có thể tự mình thực hiện được (mã nguồn mở). Người ta ước tính khoảng giữa năm 2011 có trên 300 nghìn mạch Arduino chính thức đã được sản xuất thương mại, và vào năm 2013 có khoảng 700 nghìn mạch chính thức đã được đưa tới tay người dùng. Sau khi đã đọc được dữ liệu từ các cảm biến, các mạch vi xử lý gửi dữ liệu tới các máy tính để lưu trữ, xử lý dữ liệu. Tuy nhiên, một số mạch vi xử lý không hỗ trợ các giao thức mạng như wifi, bluetooth nên cần các mạch bổ trợ như esp8266, module HC06, ...



Hình 1.1: Kiến trúc hệ thống IoT

nguồn:<http://www.open-silicon.com/wp-content/uploads/2015/05/internet-of-things1.jpg>

### 1.1.3.2 Các giao thức trong IoT

Điều quan trọng trong Internet of Things là các thiết bị phải được đưa lên internet, tức là phải có một định danh duy nhất. Tuy nhiên, với giao thức IPv4, số lượng định danh có thể có là hơn 4 tỷ định danh. Mặc dù có thể dùng các kỹ thuật để tăng số lượng IPv4 (ví dụ NAT, ...) tuy nhiên không thể giải quyết triệt để vấn đề này trong khi số lượng thiết bị IoT ngày càng tăng với tốc độ nhanh chóng và sẽ lên tới hàng chục tỷ thiết bị. Do đó giao thức IPv6 (có thể có hơn 2 triệu tỷ định danh) đang được tích cực triển khai để giải quyết vấn đề này.

Một số giao thức truyền thông được sử dụng trong IoT là các giao thức truyền thông phổ biến như wifi, bluetooth, NFC, ... Tuy nhiên, cũng có các giao thức được sử dụng rộng rãi trong IoT nhưng ít được biết đến hơn như Zigbee, MQTT, AMQP, RFID, ... Zigbee là một giao thức mạng không dây được thiết kế chủ yếu cho môi trường công nghiệp với tần suất gửi dữ liệu nhỏ, phạm vi từ 10 tới 100m, ít tiêu thụ năng lượng, thường hoạt động ở tần số 2.4 GHz. MQTT (Message Queue Telemetry Transport) là một giao thức dựa trên cơ chế message - queue, sử dụng giao thức TCP để cung cấp cơ

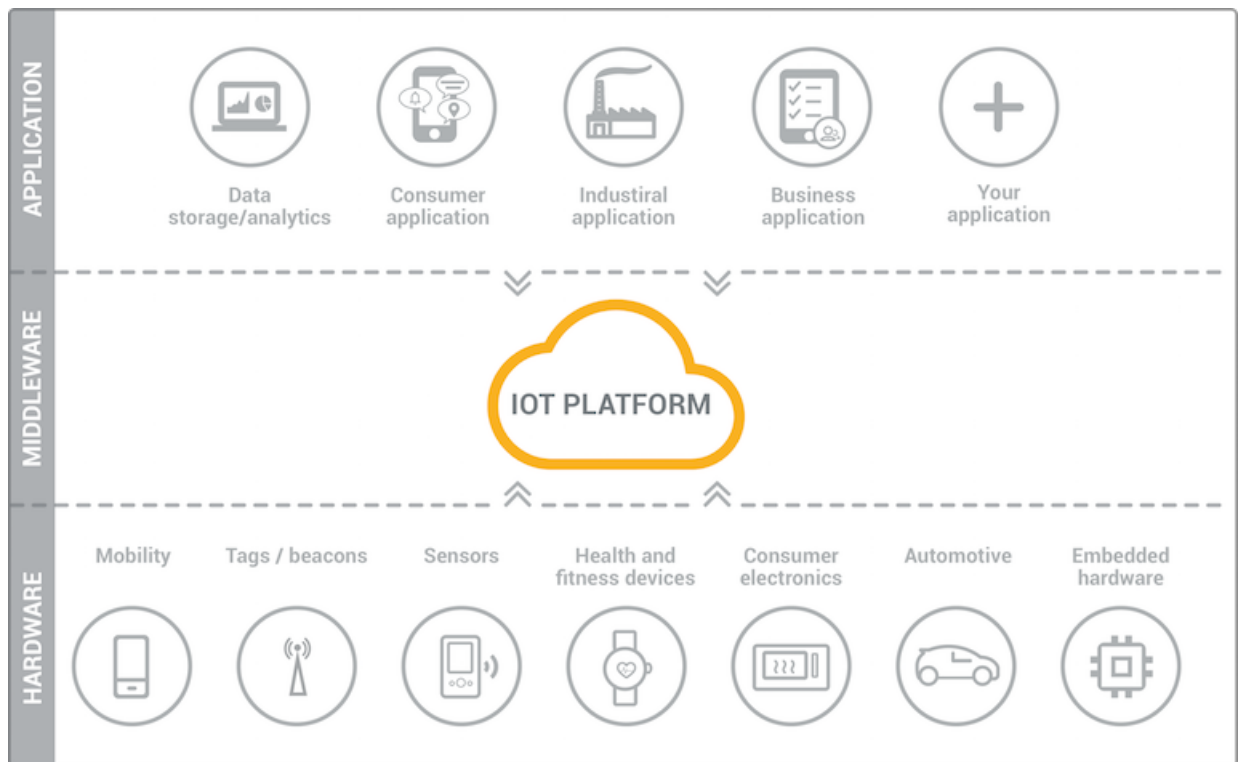
chế truyền tin đơn giản, tin cậy. Giao thức MQTT có ba thành phần chính: Subscriber, Publisher và Broker. Công việc của publisher là tạo ra dữ liệu và truyền dữ liệu cho subscriber với sự trợ giúp của broker; broker đảm bảo tính bảo mật, tin cậy dữ liệu bằng cách kiểm tra quyền truy cập của subscriber và publisher. Giao thức MQTT là giao thức được ưa thích cho các thiết bị IoT do có khả năng cung cấp đủ chức năng về thông tin cho các thiết bị nhỏ, ít bộ nhớ và ít tiêu thụ dữ liệu, làm việc trên các mạng bị giới hạn về băng thông. Tương tự như MQTT, AMQP (Advanced Message Queuing Protocol) cũng là một giao thức dựa trên cơ chế message-queue nhưng phức tạp hơn MQTT và cung cấp các tính năng nâng cao hơn MQTT.

Mạng không dây thế hệ thứ năm 5G hứa hẹn có tốc độ lớn hơn nhiều so với hiện tại và có khả năng kết nối đồng thời nhiều hơn các thiết bị thông minh. Mạng nhanh hơn đồng nghĩa với việc dữ liệu từ các thiết bị thông minh sẽ được thu thập, phân tích và xử lý với cấp độ cao hơn. Đây sẽ là nguồn năng lượng sáng tạo cho các công ty tạo ra các thiết bị IoT và thúc đẩy nhu cầu của khách hàng cho các sản phẩm mới.

### 1.1.3.3 Các IoT platform

IoT platform là một phần mềm giúp hỗ trợ quản lý các thiết bị, giúp đơn giản hóa việc giao tiếp, luồng dữ liệu, quản lý thiết bị và các chức năng của các ứng dụng. Nó đóng vai trò là lớp trung gian giữa tầng thiết bị và tầng ứng dụng. Công việc chủ yếu của nó bao gồm thu thập dữ liệu từ thiết bị thông qua các giao thức, mô hình mạng khác nhau, điều khiển và cấu hình các thiết bị.

Theo thống kê của công ty nghiên cứu thị trường IoT-Analytics của Đức năm 2017 đã có khoảng 450 IoT Platform theo [2]. Một số IoT Platform đáng được dùng phổ biến như Google Cloud Platform, Salesforce IoT Cloud, IBM Watson IoT, AWS IoT, ... Ngoài ra, các cộng đồng mã nguồn mở cũng có một số IoT Platform phổ biến như OpenHAB, HomeAssistant, ThingsBoard, ...



Hình 1.2: IoT Platform

nguồn:

[https://www.kaaproject.org/uploads/2016/12/WhatIsIoTPlatform\\_05-768x474.png](https://www.kaaproject.org/uploads/2016/12/WhatIsIoTPlatform_05-768x474.png)

#### 1.1.4 Bảo mật và an toàn trong IoT

Vấn đề bảo mật trong IoT đã trở thành một chủ đề thu hút được nhiều sự chú ý sau một loạt các vụ tấn công nghiêm trọng có liên quan tới các thiết bị IoT được sử dụng để xâm nhập và tấn công các mạng máy tính lớn. Việc thực hiện các biện pháp bảo mật đóng vai trò hết sức quan trọng trong việc đảm bảo mạng các thiết bị IoT được kết nối tới nhau an toàn.

Có một số thách thức ngăn cản sự bảo mật của các thiết bị IoT và đảm bảo bảo mật cho toàn bộ môi trường IoT. Ý tưởng về việc kết nối các thiết bị và các vật thành một mạng vẫn còn tương đối mới, do đó vấn đề bảo mật không phải luôn được xem như là ưu tiên trong suốt quá trình thiết kế sản phẩm. Thêm vào đó, vì IoT là một thị trường mới, nhiều nhà thiết kế sản phẩm và các công ty quan tâm tới việc đưa sản phẩm của họ ra thị trường hơn việc xây dựng cơ chế bảo mật ngay từ đầu. Vấn đề chính được nêu ra với việc bảo mật trong IoT là việc sử dụng các mật khẩu mặc định, có thể dẫn tới sự xâm nhập an ninh trái phép và dù mật khẩu được đổi, chúng lại thường không



đủ mạnh để ngăn chặn truy nhập trái phép. Một vấn đề phổ biến khác là các thiết bị IoT thường bị giới hạn về tài nguyên và không đủ năng lực tính toán cần thiết để cài đặt hệ thống bảo mật mạnh. Do đó, rất nhiều thiết bị không hoặc không thể cung cấp các tính năng bảo mật nâng cao. Ví dụ, một cảm biến nhiệt độ, độ ẩm không thể cài đặt cơ chế mã hóa hay các cơ chế bảo mật khác. Thêm vào đó, nhiều thiết bị IoT được đặt trong một cái máy và ở đó cho tới khi không sử dụng được nữa, chúng khó lòng được cập nhật các phiên bản bảo mật mới. Từ góc nhìn nhà sản xuất, xây dựng cơ chế bảo mật từ đầu vì có thể sẽ tốn kém, làm chậm quá trình phát triển.

Một số sự kiện về lỗ hổng, thâm nhập trái phép các thiết bị trong IoT đã được ghi nhận. Năm 2010, các nhà nghiên cứu đã phát hiện vi rút Stuxnet được sử dụng để gây thiệt hại tới các máy li tâm của Iran, các cuộc tấn công bắt đầu từ năm 2006 nhưng chủ yếu xảy ra vào năm 2009. Thường được xem là ví dụ đầu tiên của việc tấn công mạng IoT, Stuxnet nhắm vào hệ thống giám sát và thu thập dữ liệu (SCADA) với hệ thống điều khiển công nghiệp (ICS), sử dụng mã độc để đầu độc các lệnh gửi từ bộ điều khiển logic (PLC). Các vụ tấn công vào hệ thống công nghiệp vẫn tiếp tục diễn ra, với các mã độc như CrasshOverride, Triton, VPNFilter, ... Tháng 12 năm 2013, một nhà nghiên cứu tại công ty bảo mật Proofpoint Inc. đã mô tả mạng IoT botnet đầu tiên. Theo đó, hơn 25% của botnet được tạo ra từ các thiết bị IoT như smart TV, hệ thống giám sát trẻ và các đồ dùng trong nhà. Năm 2015, nhà nghiên cứu bảo mật Charlie Miller và Chris Valasek đã thực hiện hack vào một xe Jeep bằng mạng không dây, điều chỉnh máy phát radio trên xe, máy lạnh và dừng cảm biến gia tốc. Họ còn nói rằng, họ có thể tắt máy, phanh xe và tắt phanh. Miller và Valasek còn có thể thâm nhập vào mạng của chiếc xe thông qua hệ thống trong chiếc xe. Tháng 1 năm 2017, FDA cảnh báo hệ thống nhúng trong các thiết bị tim như máy điều hòa nhịp tim, máy khử rung tim và các thiết bị bất đồng bộ có thể bị thông nhập thông qua các lỗ hổng bảo mật.

### 1.1.5 Các dự đoán về IoT

Một số dự đoán về tương lai của IoT đã được đưa ra. Theo IoT Analytics, năm 2016 có khoảng 4.7 tỷ thiết bị, và tới năm 2021 số lượng thiết bị sẽ đạt tới 21 tỷ thiết bị. Nhờ sự phát triển của IoT, ngày càng nhiều thiết bị, nhà cửa, thành phố sẽ trở nên thông minh hơn, trong đó các thành phố sẽ có khả năng tự động một số công việc, quản lý từ xa và thu thập dữ liệu thông qua các ki-ốt, camera giám sát, các trạm cho thuê xe, ... Khi dữ liệu trở nên nhiều hơn, trí thông minh nhân tạo AI sẽ đóng vai trò quan trọng hơn. Học máy, một lĩnh vực trong AI sẽ giúp máy tính có thể "học" để đưa ra các gợi ý mang tính cá nhân hóa cao hơn. Trong lĩnh vực bảo mật, tội phạm mạng vẫn sẽ tiếp tục sử dụng các thiết bị IoT để thực hiện các cuộc tấn công DDos. Bên cạnh đó, mối quan tâm về bảo mật và quyền riêng tư sẽ dành nhiều hơn cho tính pháp lý và các quy định về dữ liệu, đảm bảo quyền bảo mật thông tin cá nhân của người dân tốt hơn.

## 1.2 Giới thiệu về truy vấn ngữ nghĩa

### 1.2.1 Ngôn ngữ truy vấn ngữ nghĩa

Ngôn ngữ truy vấn ngữ nghĩa là ngôn ngữ truy vấn cho phép truy vấn và phân tích ngữ cảnh tự nhiên của câu truy vấn. Truy vấn ngữ nghĩa cho phép lấy ra các thông tin rõ ràng cũng như các thông tin ẩn về dữ liệu. Để xây dựng cơ chế truy vấn ngữ nghĩa, ta cần xây dựng mô hình mối quan hệ của các dữ liệu, một trong những cách để thực hiện là xây dựng ontology.

### 1.2.2 Ontology

#### 1.2.2.1 Khái niệm ontology

Lịch sử của trí tuệ nhân tạo cho thấy rằng tri thức đóng vai trò hết sức quan trọng trong các hệ thống thông minh. Trong rất nhiều trường hợp, tri thức tốt có thể quan

trọng hơn các giải thuật trong việc giải quyết một bài toán. Để có được một hệ thống thực sự thông minh, trí thức cần được thu thập, xử lý, tái sử dụng và được trao đổi. Ontology hỗ trợ tất cả các công việc này. Khái niệm ontology có thể được định nghĩa như sự cụ thể hóa một cách rõ ràng các khái niệm trong một miền lĩnh vực. Ontology thể hiện cấu trúc của miền lĩnh vực, bao gồm cả các mối quan hệ ràng buộc của các thành phần trong lĩnh vực đó. Các khái niệm trong một lĩnh vực thường không thay đổi hoặc rất ít thay đổi, và ontology là một bản mô tả các khái niệm trong lĩnh vực đó dựa trên ngôn ngữ mô hình hóa và các từ vựng cụ thể. Việc mô tả một cách hình thức ontology là yêu cầu bắt buộc để có thể xử lý và thực hiện các thao tác một cách tự động trên ontology. Để xây dựng được ontology, ta cần xác định một cách hình thức các thành phần trong ontology như các thực thể của một khái niệm, các lớp (khái niệm), các thuộc tính của các khái niệm và mối quan hệ cũng như các ràng buộc, các luật, các tiên đề giữa các khái niệm.

Các thực thể là một thành phần cơ bản của ontology. Các thực thể trong một ontology có thể là các đối tượng cụ thể ví dụ như một người cụ thể, một con vật cụ thể, một hành tinh cụ thể, ...

Lớp hay khái niệm có thể được định nghĩa là một tập trùu tượng của các đối tượng. Các khái niệm có thể bao gồm các thực thể của các đối tượng hoặc cũng có bao gồm loại các khái niệm khác. Ví dụ, "động vật" là một khái niệm chỉ chung các cá thể động vật như chim, cá, hổ, báo, sư tử, con người, ... "Con người" lại có thể là một khái niệm chỉ tất cả những người sống trên Trái Đất.

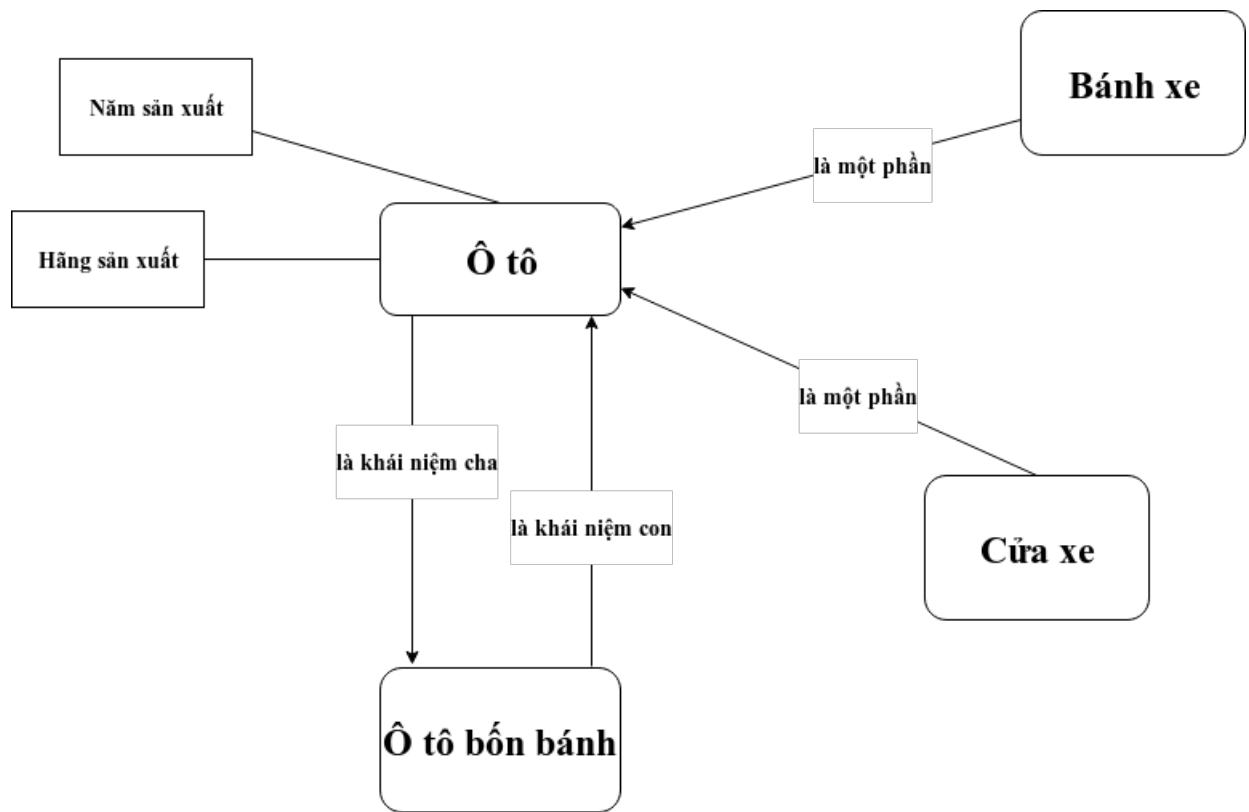
Các đối tượng trong một ontology có thể được mô tả bởi liên kết chúng tới các thành phần, các khía cạnh, ... của nó. Những thành phần, khía cạnh này được gọi là các thuộc tính tính. Mỗi thuộc tính cũng có thể là một lớp hay một thực thể. Ví dụ, "con người" có thể có thuộc tính "giới tính", "tên", "tuổi", ...

Mối quan hệ giữa các khái niệm trong một ontology xác định xem khái niệm này có liên quan tới khái niệm khác như thế nào. Ví dụ, khái niệm "con người" có mối quan hệ "là một loài" với khái niệm "động vật". Biểu diễn trở thành: "con người" là một loài "động vật" Sức mạnh của ontology đến từ khả năng miêu tả các mối quan hệ giữa

các khái niệm. Tập các mối quan hệ sẽ mô tả tính ngữ nghĩa của lĩnh vực. Một loại quan hệ rất quan trọng trong ontology là mối quan hệ: một khái niệm là khái niệm cha của một khái niệm khác. Ví dụ, "ô tô" là khái niệm cha của "ô tô bốn bánh". Ngược lại với mối quan hệ trên là mối quan hệ: một khái niệm là khái niệm con của một khái niệm khác. Ví dụ, "ô tô bốn bánh" là khái niệm con của "ô tô". Một mối quan hệ thường dùng khác là mối quan hệ: một khái niệm là một phần của một khái niệm khác. Ví dụ, "bánh xe", "cửa sổ" là một phần của "ô tô".

Tập luật là các mệnh đề dưới dạng điều kiện - kết quả (nếu-thì), dùng để mô tả phép suy dẫn logic từ một tập các mối quan hệ. Ví dụ, "nếu" "cá voi" là một loài "động vật" và "động vật" là một loài "sinh vật" thì "cá voi" là một loài sinh vật.

Các tiên đề là các khẳng định mang tính logic. Ví dụ "memo" là một "con cá" và "mọi con cá" đều biết bơi nên "memo" biết bơi.



Hình 1.3: Một ví dụ về ontology

Ontology không chỉ là bản mô tả có thể chia sẻ và tái sử dụng của một lĩnh vực mà còn có thể thêm các tri thức mới về lĩnh vực đó. Có một số cách khác có thể được sử dụng để mô tả một cách hình thức tri thức về một miền lĩnh vực như cơ sở dữ liệu quan hệ, sử dụng tập từ vựng của lĩnh vực, ... tuy nhiên, ontology nhấn mạnh mối quan hệ giữa các khái niệm và cho phép người sử dụng liên kết các khái niệm này với các khái niệm khác theo nhiều cách khác nhau.

### 1.2.2.2 Cách xây dựng một ontology

Có nhiều phương pháp để xây dựng một ontology, tuy nhiên, nhìn chung, các phương pháp đều thực hiện hai bước cơ bản: xây dựng cấu trúc lớp phân cấp và định nghĩa các thuộc tính cho các lớp. Trong thực tế, để xây dựng một ontology mô tả một lĩnh vực là một công việc khó khăn, phụ thuộc rất nhiều vào công cụ sử dụng, tính chất, quy mô, sự thường xuyên biến đổi của lĩnh vực đó, cũng như các mối quan hệ phức tạp trong đó. Những khó khăn này đòi hỏi quá trình xây dựng ontology phải là một quá trình lặp đi lặp lại, mỗi lần lặp cải thiện, tinh chế và phát triển dần sản phẩm

chứ không phải là một quy trình với các công đoạn tách rời nhau. Công việc xây dựng ontology cũng cần phải tính đến khả năng mở rộng lĩnh vực quan tâm trong tương lai, khả năng kế thừa các ontology đã có, cũng như có tính linh động để ontology có khả năng mô tả tốt nhất các mối quan hệ phức tạp trong thế giới thực. Một số nguyên tắc cơ bản trong xây dựng ontology:

- Không có một cách duy nhất nào để mô hình hóa một lĩnh vực. Luôn có rất nhiều cách có thể thay thế cho nhau. Việc xác định cách tốt nhất thường phụ thuộc vào ứng dụng của mô hình và các mở rộng trong tương lai mà ta đoán trước sẽ có thể xảy ra.
- Quá trình phát triển ontology là một quá trình lặp đi lặp lại
- Các khái niệm trong ontology nên sát với các đối tượng (vật lý hoặc logic) trong lĩnh vực mà nó mô tả. Các khái niệm thường là những danh từ, các mối quan hệ thường là những động từ được dùng trong lĩnh vực.

Do đó, việc quyết định xem ontology dùng để làm gì và mức độ chi tiết hay tổng quát của ontology sẽ ảnh hưởng tới các quyết định trong khi xây dựng ontology. Trong rất nhiều các lựa chọn, chúng ta phải xác định lựa chọn nào sẽ phù hợp nhất với mục đích của ứng dụng, lựa chọn nào sẽ giúp ontology có khả năng mở rộng và có tính ổn định cao. Chúng ta cũng cần nhớ rằng một ontology là một bản mô tả một phần của thế giới và các khái niệm trong ontology phải phản ánh được phần thế giới đó. Sau khi ta đã tạo ra được một phiên bản đầu tiên của một ontology, ta có thể đánh giá, sửa đổi nó dựa trên ứng dụng sử dụng nó, dựa trên ý kiến của các chuyên gia trong lĩnh vực,... Quá trình sửa đổi này sẽ tiếp tục cho tới hết vòng đời sử dụng của ontology. Về cơ bản, việc xây dựng ontology cần thực hiện các bước sau:

#### Bước 1: Xác định lĩnh vực và phạm vi của ontology

Thông thường, các yêu cầu đối với một hệ thống Ontology là mô tả lĩnh vực quan tâm nhằm phục vụ cơ sở tri thức trong việc giải quyết những mục đích chuyên biệt. Công việc đặc tả để xác định, phân tích, nhận diện chính xác yêu cầu được thực hiện bằng cách trả lời những câu hỏi sau:

- Ontology cần mô tả lĩnh vực nào?
- Mục đích sử dụng ontology là gì?
- Cơ sở tri thức trong Ontology sẽ giải quyết những câu hỏi gì?
- Ai là người sẽ xây dựng, quản trị Ontology?

Nhìn chung, câu trả lời cho các câu hỏi dạng này có thể sẽ thường xuyên thay đổi trong suốt quá trình xây dựng một Ontology. Nhất là khi có sự thay đổi về mục đích hoặc cần bổ sung tính năng trong việc sử dụng cơ sở tri thức. Tuy nhiên, việc trả lời chính xác các câu hỏi trên tại mỗi bước lập sẽ giúp giới hạn phạm vi của mô hình cần mô tả và dự trù các kỹ thuật sẽ sử dụng trong quá trình phát triển. Lấy ví dụ, nếu dự trù khả năng xảy ra sự khác biệt về ngôn ngữ giữa người phát triển và người sử dụng thì Ontology phải được bổ sung cơ chế ánh xạ (mapping) qua lại các thuật ngữ giữa các ngôn ngữ khác nhau. Hoặc giả sử Ontology cần xây dựng có chức năng xử lý ngôn ngữ tự nhiên, ứng dụng dịch tài liệu tự động thì cũng cần thiết phải có kỹ thuật xác định từ đồng nghĩa. Sau khi đã phác thảo phạm vi ontology dựa trên việc trả lời những câu hỏi trên, người thiết kế sẽ trả lời các câu hỏi mang tính đánh giá, qua đó tiếp tục tinh chỉnh lại phạm vi của hệ thống cần xây dựng. Các câu hỏi dạng này thường dựa trên cơ sở tri thức của Ontology và được gọi là câu hỏi kiểm chứng khả năng (competency question):

- Ontology đã có đủ thông tin để trả lời cho các câu hỏi được quan tâm trên cơ sở tri thức hay không?
- Câu trả lời của cơ sở tri thức đã đáp ứng được mức độ, yêu cầu nào của người sử dụng?
- Các ràng buộc và quan hệ phức tạp trong miền quan tâm đã được biểu diễn hợp lý chưa?

Bước 2: Xem xét việc kế thừa các ontology có sẵn:

Đây là một công đoạn thường hay sử dụng để giảm thiểu công sức xây dựng một

Ontology. Bằng cách kế thừa các ontology tương tự có sẵn, người xây dựng có thể thêm hoặc bớt các lớp, quan hệ giữa các lớp, thực thể để tinh chỉnh tùy theo mục đích của mình. Việc sử dụng lại các Ontology có sẵn cũng rất quan trọng khi cần sự tương tác giữa các ứng dụng khác nhau, các ứng dụng sẽ cần phải hiểu các lớp, thực thể, quan hệ của nhau để thuận tiện trong việc trao đổi hoặc thống nhất thông tin. Vấn đề xây dựng một ontology mới bằng cách kế thừa các hệ thống có sẵn liên quan đến một bài toán rất phức tạp là trộn (merging) các Ontology. Ví dụ trong trường hợp tên các khái niệm được định nghĩa trong các Ontology này có thể giống nhau trong khi chúng được dùng để mô tả các loại vật hoàn toàn khác nhau, và cũng có thể xảy ra trường hợp ngược lại, khi tên các khái niệm khác nhau nhưng cùng mô tả một sự vật, và một vấn đề nữa là làm thế nào để bổ sung các quan hệ, thuộc tính có sẵn vào một hệ thống mới. Tuy nhiên, hầu hết các Ontology sử dụng trong ngành khoa học máy tính nói chung và Web ngữ nghĩa nói riêng đều được xây dựng trên các hệ thống xây dựng và quản trị Ontology. Tên một số công cụ như: Sesame, Protégé, Ontolingua, Chimaera, OntoEdit, OidEd. Hiện nay, đa số các phần mềm này đều hỗ trợ chức năng tự động trộn các Ontology cùng hoặc thậm chí khác định dạng với nhau. Mặc dù vậy, ở mức nào đó, người xây dựng cũng cần phải kiểm tra lại một cách thủ công, nhưng đây không phải là một công việc phức tạp. Hiện có rất nhiều Ontology được chia sẻ trên Web nổi tiếng như: UNSPSC ([www.unspsc.org](http://www.unspsc.org)) do chương trình phát triển của Liên Hiệp Quốc hợp tác với tổ chức Dun & Bradstreet nhằm cung cấp các thuật ngữ của các sản phẩm và dịch vụ thương mại. Các Ontology trong lĩnh vực thương mại khác như: RosettaNet ([www.rosettanet.org](http://www.rosettanet.org)), DMOZ ([www.dmoz.org](http://www.dmoz.org)), eClassOwl Open Biological, BioPax trong lĩnh vực sinh vật học, UMLS trong lĩnh vực mạng ngữ nghĩa, GO (Gene Ontology), WordNet (đại học Princeton).

Bước 3: Liệt kê các thuật ngữ quan trọng trong Ontology:

Đây là bước rất hữu ích, làm tiền đề cho hai bước tiếp theo là xây dựng cấu trúc lớp phân cấp và định nghĩa các thuộc tính cho lớp. Công đoạn này bắt đầu bằng việc liệt kê tất cả các thuật ngữ xuất hiện trong miền quan tâm (có thể đồng nghĩa hoặc chồng nhau) như tên khái niệm, quan hệ, thuộc tính. Thông thường, các thuật ngữ là danh từ sẽ trở thành các lớp, tính từ sẽ trở thành thuộc tính, còn động từ sẽ là quan hệ giữa



các lớp.

Bước 4: Xây dựng các lớp/khái niệm và cấu trúc lớp/khái niệm phân cấp:

Đây là một trong hai bước quan trọng nhất của công việc xây dựng một Ontology. Nhiệm vụ của bước này là định nghĩa các lớp từ một số thuật ngữ đã liệt kê trong bước trên, sau đó xây dựng cấu trúc lớp phân cấp theo quan hệ lớp cha-lớp con. Lớp ở vị trí càng cao sẽ có mức độ tổng quát càng cao. Vị trí đầu tiên thuộc về lớp gốc, tiếp theo là các lớp trung gian, và cuối cùng là lớp lá. Lớp lá là lớp không thể triển khai được nữa và chỉ được biểu hiện bằng các thực thể. Quan hệ giữa thực thể của lớp con với lớp cha là quan hệ "là một", nghĩa là một thực thể của lớp con cũng "là-một" thực thể của lớp cha. Có nhiều hướng tiếp cận khác nhau cho vấn đề xây dựng cấu trúc lớp phân cấp. Có thể kể ra ba hướng như sau:

- Hướng xây dựng từ trên xuống (top-down): bắt đầu bằng các lớp có mức độ tổng quát cao nhất, sau đó triển khai dần đến lớp lá.
- Hướng xây dựng từ dưới lên (bottom-up): ngược với hướng xây dựng cấu trúc lớp phân cấp từ trên xuống, hướng này bắt đầu bằng việc xác định các lớp được cho là cụ thể nhất, sau đó tổng quát hóa đến khi được lớp gốc.
- Cách kết hợp (combination): cách này kết hợp cả hai hướng xây dựng trên. Đầu tiên chọn các lớp nổi bật nhất trong lĩnh vực quan tâm, sau đó tổng quát hóa và cụ thể hóa cho đến khi được cấu trúc mong muốn.

Không có phương pháp nào trong ba phương pháp này là tốt nhất trong mọi trường hợp. Việc lựa chọn phương pháp nào phụ thuộc vào góc nhìn cá nhân của người xây dựng vào lĩnh vực. Nếu người xây dựng có một cái nhìn hệ thống từ trên xuống thì phương pháp xây dựng từ trên xuống sẽ phù hợp. Ngược lại, một người có cái nhìn từ dưới lên sẽ phù hợp với cách xây dựng từ dưới lên. Phương pháp kết hợp thường là cách dễ nhất cho những người xây dựng.

Bước 5: Định nghĩa các thuộc tính và quan hệ cho lớp:

Bản thân các lớp nhận được ở bước trên chỉ mới là những thuật ngữ phân biệt với nhau bằng tên gọi. Về cơ bản, chúng chưa đủ để phục vụ cho việc biểu diễn tri thức.

Muốn như vậy, các thuộc tính của lớp cần được định nghĩa. Thuộc tính của lớp là các thông tin bên trong của lớp, mô tả một khía cạnh nào đó của lớp và được dùng để phân biệt với các lớp khác. Thuộc tính được chia làm nhiều loại khác nhau: Về mặt ý nghĩa, các thuộc tính có thể được chia làm hai loại: thuộc tính bên trong (intrinsic property) và thuộc tính bên ngoài (extrinsic property). Thuộc tính bên trong mô tả các tính chất nội tại bên trong sự vật, ví dụ: chất, lượng, cấu tạo. Trong khi đó, thuộc tính bên ngoài mô tả phần biểu hiện của sự vật, ví dụ: màu sắc, hình dạng. Về mặt giá trị, các thuộc tính cũng được chia làm hai loại: thuộc tính đơn (simple property) và thuộc tính phức (complex property). Thuộc tính đơn là các giá trị đơn, ví dụ: chuỗi, số,.. còn thuộc tính phức có thể chứa hoặc tham khảo đến một đối tượng khác. Một chú ý quan trọng nữa trong bước này là việc một lớp sẽ kế thừa toàn bộ các thuộc tính của tất cả các cha nó. Do đó cần phải xem xét một thuộc tính đã được định nghĩa ở các lớp thuộc mức cao hơn hay chưa. Thuộc tính chỉ nên được định nghĩa khi nó là tính chất riêng của lớp đang xét mà không được biểu hiện ở các lớp cao hơn.

#### Bước 6: Định nghĩa các ràng buộc của các thuộc tính

Các thuộc tính có thể có các ràng buộc như kiểu giá trị, miền giá trị, số lượng giá trị và các ràng buộc khác. Ví dụ, một tên phải là một chuỗi các chữ cái, do đó, thuộc tính "tên" phải có ràng buộc là một chuỗi (string). Một số ràng buộc của các thuộc tính thường gặp:

- Số lượng giá trị: Xác định một thuộc tính có thể có bao nhiêu giá trị. Một vài thuộc tính chỉ có thể có một giá trị, ví dụ thuộc tính "mã số sinh viên" của khái niệm "sinh viên" chỉ có thể nhận một giá trị. Một số thuộc tính lại có thể nhận nhiều giá trị, ví dụ thuộc tính "số điện thoại" của khái niệm "sinh viên" có thể nhận nhiều giá trị.
- Kiểu giá trị: Xác định loại giá trị nào mà thuộc tính có thể nhận. Một số kiểu giá trị thường gặp như:
  - Chuỗi (string): là kiểu giá trị đơn giản nhất, được dùng cho các thuộc tính như "tên", "quê quán", ...

- Số (Number): đôi khi có thể được phân nhỏ thành số nguyên và số thực, xác định thuộc tính dạng số, ví dụ như "tuổi", "số lượng", "giá", ...
- Đúng-sai (boolean): đơn giản chỉ nhận hai giá trị đúng hoặc sai. Kiểu giá trị này thường được dùng cho các thuộc tính như "giới tính", ...
- Liệt kê (enumerated): liệt kê các giá trị mà thuộc tính có thể nhận. Ví dụ, "kích cỡ" có thể nhận các giá trị to, trung bình, nhỏ.
- Miền giá trị: Xác định miền giá trị mà thuộc tính có thể có. Ví dụ "nhiệt độ phòng" có thể có miền giá trị từ 0 đến 100 oC.

#### Bước 7: Tạo ra các thực thể

Đây là bước cuối cùng khép lại một vòng lặp xây dựng Ontology. Công việc chính lúc này là tạo thực thể cho mỗi lớp và gán giá trị cho các thuộc tính. Nhìn chung, các thực thể sẽ tạo nên nội dung của một cơ sở tri thức.

#### **1.2.2.3 Ngôn ngữ ontology**

Ngôn ngữ Ontology (Ontology languages) là ngôn ngữ hình thức được sử dụng để xây dựng ontology. Nó cho phép việc mã hóa tri thức trong một lĩnh vực cụ thể và thường bao gồm các quy tắc suy luận cung cấp cho việc xử lý các yêu cầu dựa trên tri thức đó. Ngôn ngữ ontology thường là ngôn ngữ khai báo, và hầu hết là những sự tổng hợp của ngôn ngữ cấu trúc. Có rất nhiều ngôn ngữ Ontology đã được thiết kế và đưa ra tuân theo sự tiêu chuẩn hóa, ta sẽ tìm hiểu một ngôn ngữ Ontology được sử dụng thông dụng trong ngữ cảnh của Web ngữ nghĩa và biểu diễn tri thức hiện nay là RDF.

RDF (Resource Description Framework) là nền tảng cho việc biểu diễn dữ liệu trong lĩnh vực Web ngữ nghĩa. Thông tin biểu diễn theo mô hình RDF là một phát biểu (statement) ở dạng cấu trúc bộ ba gồm ba thành phần cơ bản là: đối tượng, thuộc tính hay quan hệ, giá trị. Trong đó:

- Đối tượng: chỉ đối tượng đang được mô tả đóng vai trò là chủ thể.

- Thuộc tính hay đối tượng là thuộc tính hay quan hệ của đối tượng.
- Giá trị: giá trị thuộc tính hay đối tượng của chủ thể đã nêu. Giá trị có thể là một giá trị nguyên thủy như số nguyên, chuỗi, ...

Ví dụ: Sinh viên có tuổi là 20 được hiểu là một phát biểu đối tượng sinh viên, có thuộc tính là có tuổi là và giá trị là 20. Có thể liệt kê một số ưu điểm của việc lưu trữ dữ liệu RDF so với dữ liệu truyền thống là: Tổ chức dữ liệu đơn giản, đồng nhất nên thông tin dễ dàng thêm bớt chỉnh sửa. Cấu trúc bộ ba giúp cho thông tin dễ truy xuất bởi các hệ thống suy luận, tìm kiếm ngữ nghĩa. Cũng nhờ vậy mà những bộ xử lý RDF có thể suy luận ra những thông tin mới không có trong hệ dữ liệu. Chia sẻ dữ liệu trên mạng dễ dàng nhờ sự đồng nhất. Ngoài RDF, còn có một số ngôn ngữ ontology được dùng khá phổ biến khác như RDFS, OWL,... để mô tả các tài nguyên trên môi trường web.

## 1.3 Nhiệm vụ của đề tài

Để xây dựng cơ chế truy vấn ngữ nghĩa cho lĩnh vực IoT, trước hết, ta phải xây dựng một ontology biểu diễn tri thức của lĩnh vực IoT. Sau khi đã có ontology, phải sử dụng ngôn ngữ hình thức mô tả dữ liệu IoT theo các thành phần của ontology. Do dữ liệu trong IoT là đa dạng, không đồng nhất về mặt cấu trúc, nên ta phải ánh xạ các dữ liệu đa dạng đó về một chuẩn chung của ngôn ngữ hình thức được sử dụng. Cuối cùng, ta phải xây dựng cơ chế để tạo ra các truy vấn dựa trên dữ liệu thống nhất đó để đạt được mục đích ngữ nghĩa.

## Chương 2

# Xây dựng hệ thống IoT và thiết kế ngôn ngữ truy vấn ngữ nghĩa

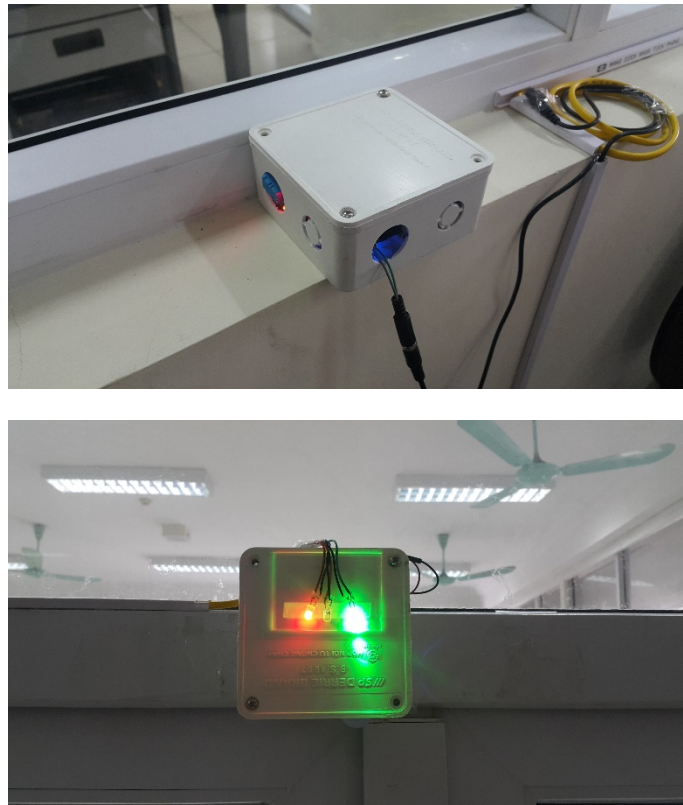
### 2.1 Xây dựng hệ thống IoT

#### 2.1.1 Phần cứng

Các thiết bị IoT được cài đặt trên 3 khu vực trong phòng 609 thư viện Tạ Quang Bửu. Trong mỗi phòng, có một máy tính nhỏ Raspberry Pi 3, cài đặt IoT platform để quản lý các thiết bị trong phòng. Trong mỗi phòng, gồm :

- Một cảm biến chuyển động
- Một cảm biến ánh sáng
- Một cảm biến nhiệt độ, độ ẩm
- Ba đèn LED tượng trưng cho ba thiết bị IoT có khả năng thiết lập các trạng thái khác nhau (bật/tắt).

Các thiết bị trong mỗi phòng được gom thành hai "vật" tương ứng với hai hộp. "Vật" 1 chứa cảm biến chuyển động và ba đèn LED; "vật" 2 chứa cảm biến ánh sáng, cảm biến nhiệt độ-độ ẩm.



Hình 2.1: Hai hộp đại diện cho hai "vật"

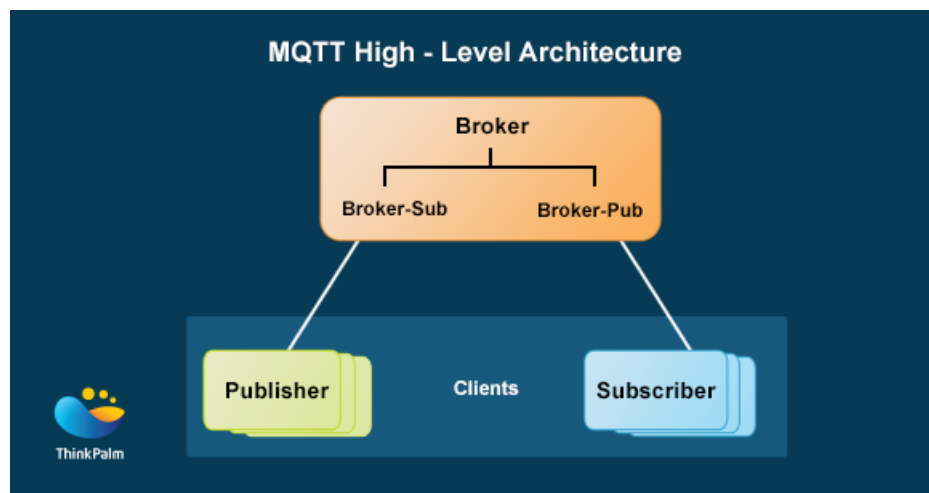
Để lấy được dữ liệu từ các thiết bị trên, ta sử dụng board vi mạch xử lý Arduino Uno. Tuy nhiên, vi mạch này không có chức năng gửi dữ liệu thông qua mạng wifi, do đó, cần phải lắp thêm module ESP8266 để gửi dữ liệu từ Arduino, qua ESP8266 rồi gửi lên platform.

### 2.1.2 Các giao thức sử dụng

Giao thức được dùng để truyền tải dữ liệu từ ESP8266 lên IoT platform là MQTT thông qua mạng wifi. MQTT (Message Queuing Telemetry Transport) là một giao thức gửi dạng publish/subscribe sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận message (gói tin) từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs,... Client thì được chia thành 2 nhóm là publisher và

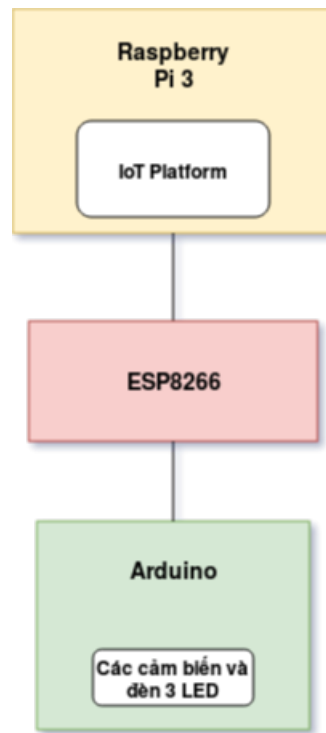
subscriber. Client là các chương trình được thiết kế để có thể hoạt động một cách linh hoạt (lightweight). Client chỉ làm ít nhất một trong 2 việc là publish các message lên một topic cụ thể hoặc subscribe một topic nào đó để nhận message từ topic này.

Các khái niệm đáng chú ý trong giao thức MQTT Message Trong giao thức MQTT, message còn được gọi là "message payload", có định dạng mặc định là plain-text (chữ viết người đọc được), tuy nhiên người sử dụng có thể cấu hình thành các định dạng khác. Topic Topic có thể coi như một "đường truyền" logic giữa 2 điểm là publisher và subscriber. Về cơ bản, khi message được publish vào một topic thì tất cả những subscriber của topic đó sẽ nhận được message này. Mô hình triển khai:



Hình 2.2: Giao thức truyền tin MQTT

Nguồn: <https://techmaster.vn/posts/34394/iot-giao-thuc-mqtt-va-ung-dung-trong-iot>

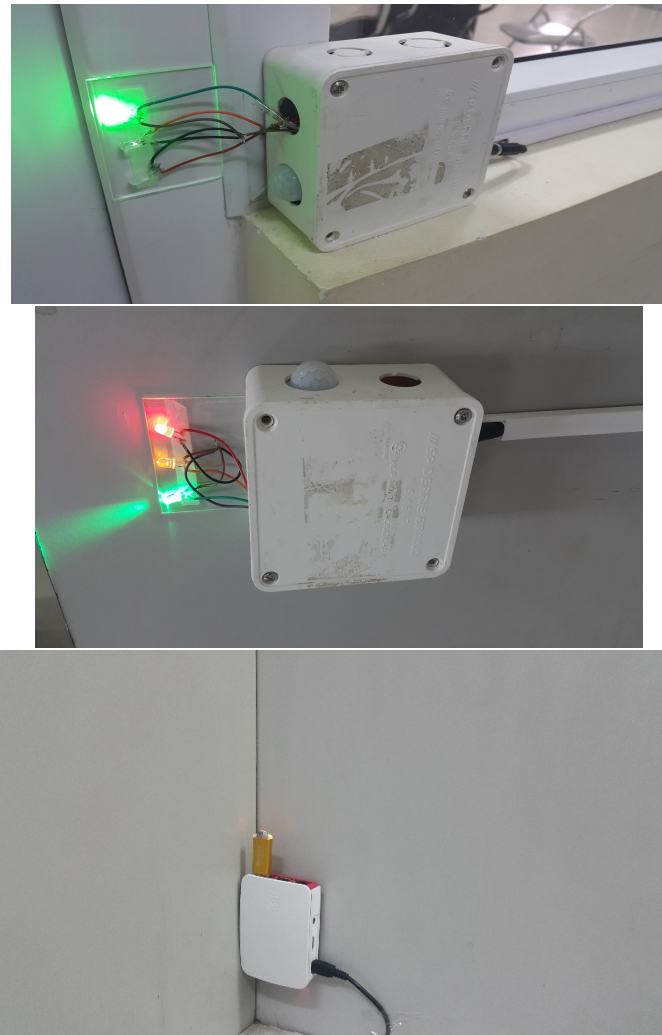


Hình 2.3: Mô hình triển khai phần cứng

Hệ thống triển khai thực tế :







Hình 2.4: Triển khai phần cứng trên phòng 609 Thư viện TQB

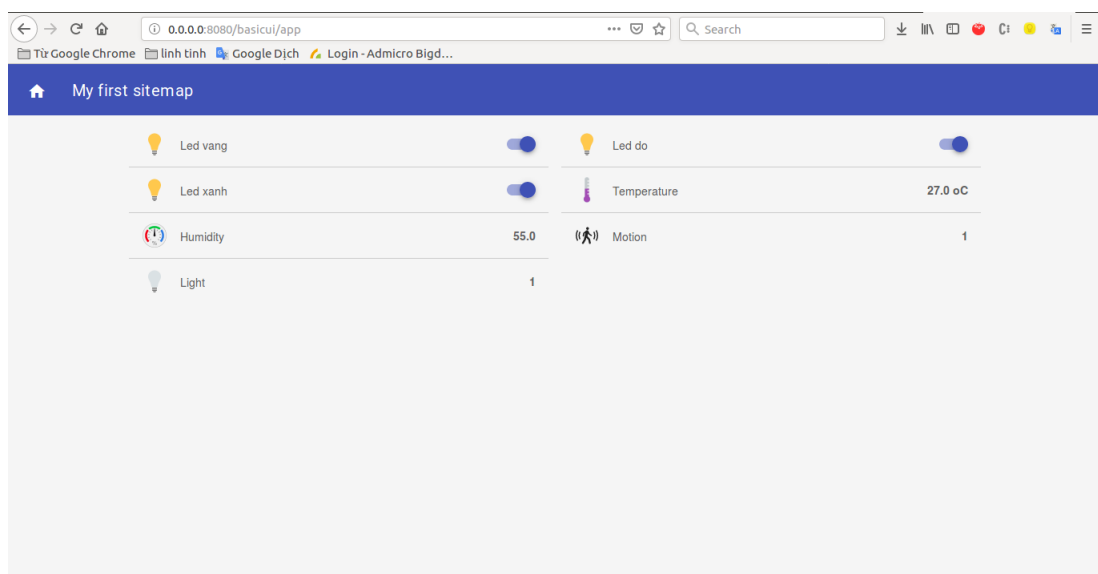
### 2.1.3 Các IoT Platform sử dụng

Hiện nay, trong thực tế có rất nhiều các IoT platform được sử dụng, cả các platform nguồn đóng và nguồn mở. Các IoT platform mã nguồn đóng thường được các công ty hoạt động vì lợi nhuận xây dựng, ví dụ như IBM, Amazon, ... còn các IoT platform mã nguồn mở thường được cộng đồng mã nguồn mở hoặc các công ty hoạt động phi lợi nhuận xây dựng. Do đề tài chỉ mang mục đích nghiên cứu, nên tôi sử dụng ba IoT platform mã nguồn mở để mô tả tính không đồng nhất về cấu trúc dữ liệu của các platform và cách ánh xạ các cấu trúc không đồng nhất này về dạng chuẩn chung của ontology. Ba platform được sử dụng là OpenHAB, HomeAssistant và ThingsBoard.

OpenHAB là một IoT platform được dùng chủ yếu để quản lý các thiết bị trong ngôi nhà thông minh. Các ưu điểm của OpenHAB là:

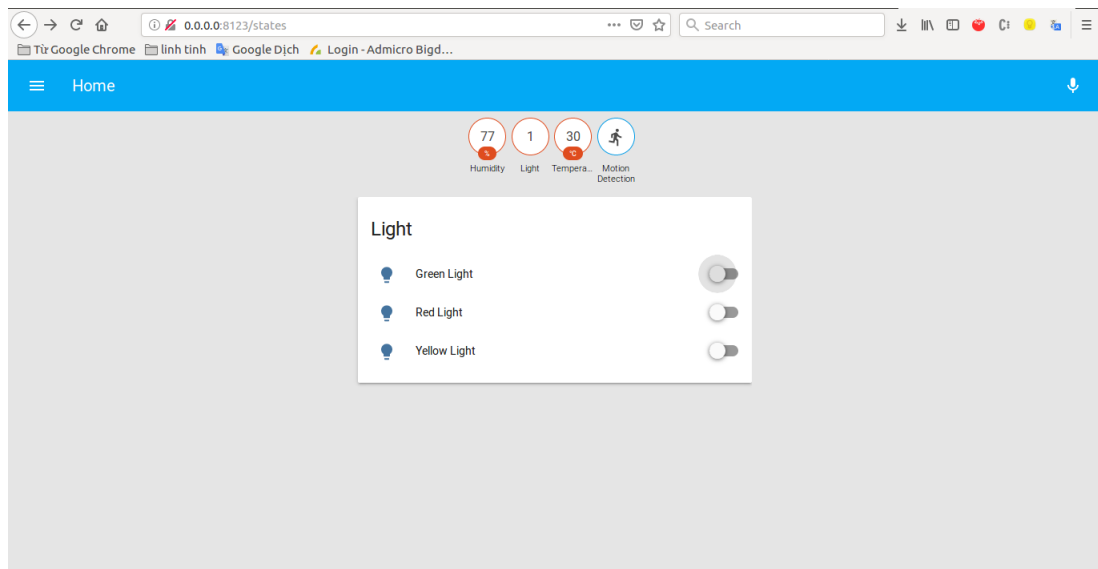
- Có khả năng tích hợp các thiết bị, hệ thống khác.
- Cung cấp giao diện thống nhất cho người dùng và người dùng có thể tạo ra các luật dựa trên thông tin của các thiết bị trong hệ thống.
- Cung cấp một công cụ linh động để tạo ra một ngôi nhà tự động. OpenHAB hỗ trợ tự thêm rất nhiều loại thiết bị, giao thức phổ biến trong IoT.

OpenHAB cung cấp tập các Restful API để các chương trình khác có thể sử dụng để lấy thông tin về các thiết bị hoặc ra lệnh cho các thiết bị.



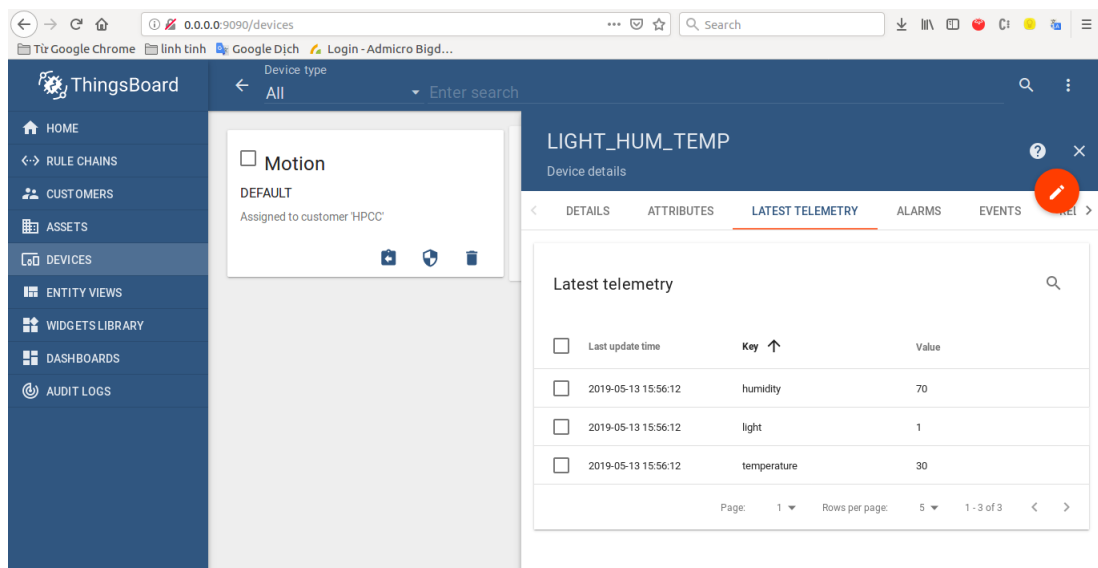
Hình 2.5: Giao diện quản lý thiết bị của platform OpenHAB

Home Assistant là một phần mềm mã nguồn mở ưu tiên quản lý các thiết bị trong ngôi nhà thông minh và có quan tâm tới tính riêng tư. Được cộng đồng lớn mạnh trên thế giới phát triển. Home Assistant phù hợp chạy trên Raspberry Pi hoặc một máy chủ tại chỗ. Home Assistant cũng lượng lớn hỗ trợ các thiết bị thông minh trên thị trường cũng như các giao thức phổ biến được sử dụng trong IoT. Giống với OpenHAB, Home Assistant cũng cung cấp các Restful API để lấy dữ liệu của thiết bị, đồng thời điều khiển các thiết bị qua các API này.



Hình 2.6: Giao diện quản lý thiết bị của platform Home Assistant

Thingsboard là một IoT platform mã nguồn mở giúp thu thập dữ liệu, xử lý, trực quan hóa dữ liệu và quản lý các thiết bị. Nó cho phép kết nối các thiết bị thông qua các giao thức chuẩn của IoT như MQTT, CoAp, HTTP. Thingsboard đảm bảo tính mở rộng, tính chịu lỗi và hiệu năng nên người dùng không sợ bị mất dữ liệu. Không giống như OpenHAB chưa có tính bảo mật dữ liệu, hay cơ chế bảo mật yếu Home Assistant, Thingsboard có cơ chế bảo mật dữ liệu của các thiết bị thông qua Token. Để truy cập được vào các thiết bị, bắt buộc phải biết token của các thiết bị này.



Hình 2.7: Giao diện quản lý thiết bị của platform Thingsboard

## 2.2 Thiết kế ngôn ngữ truy vấn cho IoT

### 2.2.1 Xây dựng ontology

Tuân theo các nguyên tắc và các bước xây dựng ontology, tôi thực hiện việc xây dựng ontology theo các bước sau:

Bước 1: Xác định lĩnh vực và phạm vi của ontology

Lĩnh vực cần xây dựng ontology là IoT. Tuy nhiên, lĩnh vực IoT rất rộng lớn, gồm đa dạng các thiết bị, cảm biến và gồm nhiều khái niệm. Do đó, việc giới hạn phạm vi xây dựng ontology trong đề tài này là công việc quan trọng. Từ đó, tôi giới hạn phạm vi nghiên cứu của đề tài là lĩnh vực IoT trong nhà thông minh gồm có các thành phần: các cảm biến, các thiết bị, các IoT platform, các ứng dụng/phạm vi sử dụng các IoT platform. Mục đích sử dụng của ontology là để tạo ra các câu truy vấn ngữ nghĩa đơn giản. Một số câu hỏi ontology cần có thể trả lời được như:

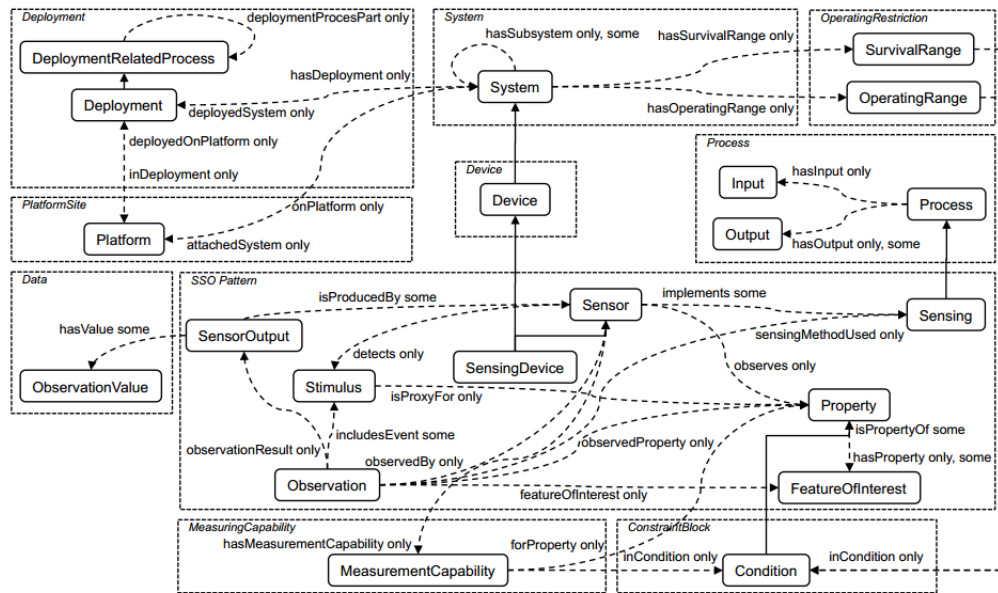
- Một khái niệm có những thuộc tính tương ứng nào?
- Một thực thể có các mối quan hệ với các thực thể nào khác?
- ...

Ngoài ra, ontology cũng cần có khả năng mở rộng so với các thành phần trong ngôi nhà thông minh.

Bước 2: Xem xét việc kế thừa các Ontology có sẵn:

Trong các ontology về IoT, SSN ontology là ontology được nhắc đến rất phổ biến. Ontology này mô tả các cảm biến, các quan sát và các khái niệm liên quan. Nó không mô tả các thành phần như thời gian, địa điểm, ... Những thành phần này, có thể được thêm vào từ các ontology khác thông qua ngôn ngữ OWL (Web Ontology Language). SSN ontology được phát triển bởi W3C Semantic Sensor Network Incubator Group (SSN-XG). Việc thiếu các khái niệm để mô tả cụ thể các loại cảm biến, đơn vị của các

giá trị thu được, thời gian, địa điểm, ... khiến cho SSN ontology khó tích hợp được vào các lĩnh vực cụ thể mà đôi khi không theo chuẩn chung.

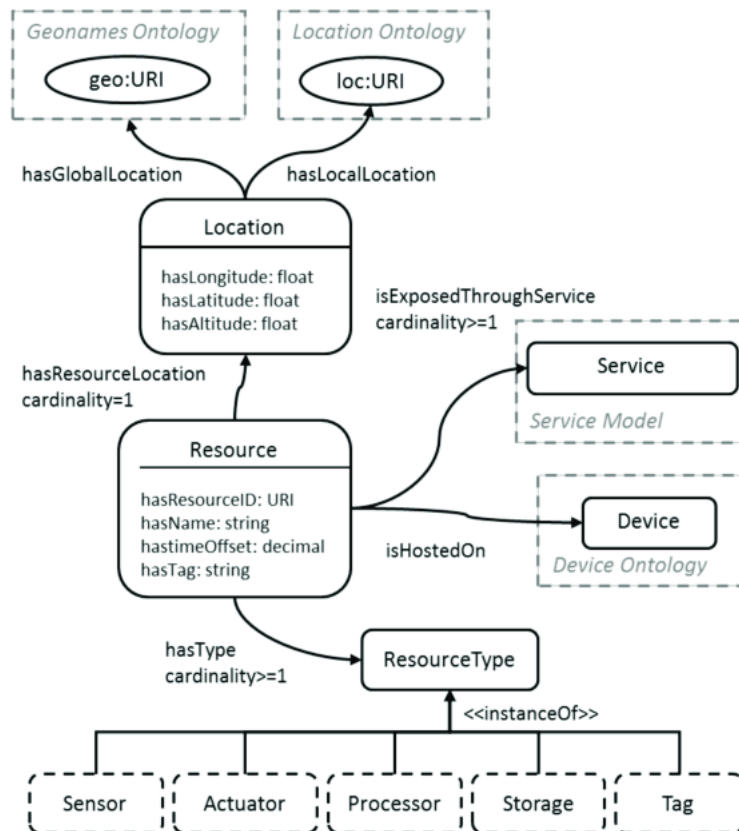


Hình 2.8: SSN ontology

nguồn:

[https://corescholar.libraries.wright.edu/knoesis/610/?utm\\_source=corescholar.libraries.wright.edu%2Fknoesis%2F610&utm\\_medium=PDF&utm\\_campaign=PDFCoverPages](https://corescholar.libraries.wright.edu/knoesis/610/?utm_source=corescholar.libraries.wright.edu%2Fknoesis%2F610&utm_medium=PDF&utm_campaign=PDFCoverPages)

Một ontology khác là ontology IoT-A, cung cấp các khái niệm chính như Service, và các khái niệm chủ yếu theo liên quan tới các Service. Nó chỉ tái sử dụng khái niệm *ssn:condition*. Hơn nữa, nó rất phức tạp, không sử dụng các ontology chuẩn, và gặp nhiều vấn đề về tính dư thừa.

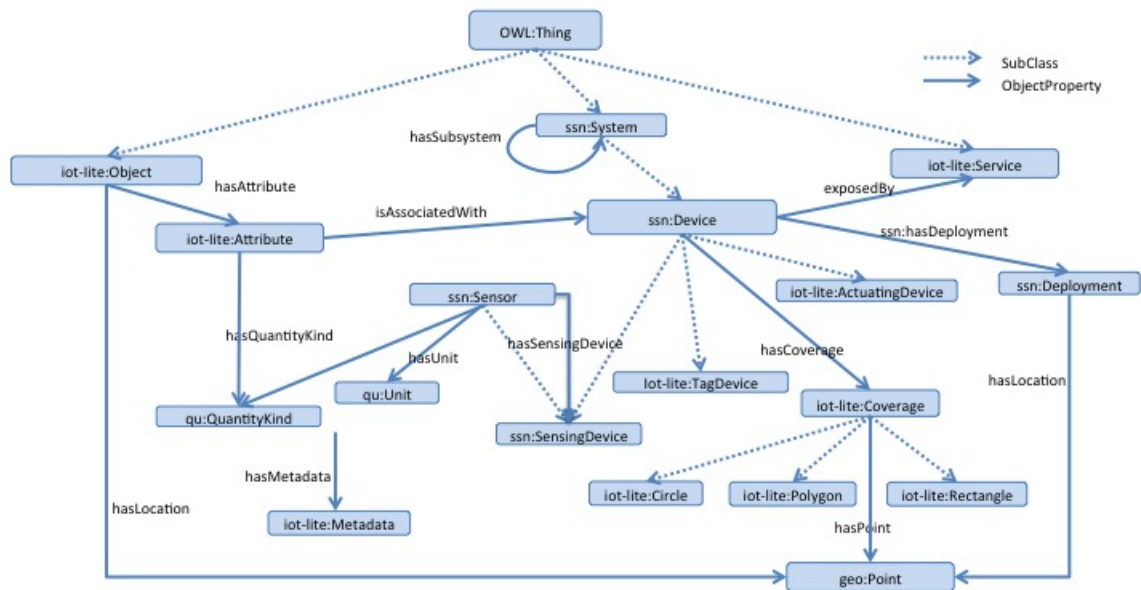


Hình 2.9: IoT-A ontology

nguồn:

[https://www.researchgate.net/profile/Suparna\\_De/publication/330729011/figure/fig1/AS:720756187090946@1548853074205/Internet-of-Things-Architecture-IoT-A-Resource-Model.png](https://www.researchgate.net/profile/Suparna_De/publication/330729011/figure/fig1/AS:720756187090946@1548853074205/Internet-of-Things-Architecture-IoT-A-Resource-Model.png)

IoT-Lite ontology là một ontology rút gọn của SSN ontology để biểu diễn các tài nguyên, thực thể và các dịch vụ trong IoT. Sự rút này cho phép IoT-Lite không quá lớn và không cần thời gian dài để xử lý khi thực hiện một truy vấn trên ontology này. Tuy nhiên, nó cũng cho phép khả năng mở rộng để biểu diễn các khái niệm trong IoT một cách chi tiết hơn, trong các lĩnh vực khác. Nó cũng có thể được kết hợp với các ontology biểu diễn dữ liệu IoT dạng stream như SAO ontology.



Hình 2.10: IoT-Lite ontology

nguồn:

<https://www.w3.org/Submission/2015/SUBM-iot-lite-20151126/figures/Ontology.jpg>

Mặc dù đã tồn tại nhiều ontology, tuy nhiên, các ontology này chủ yếu được viết theo ngôn ngữ RDF, OWL dựa trên nền tảng web. Hơn nữa, các ontology này khá phức tạp, không phù hợp với phạm vi của đề tài. Do đó, tôi chỉ kế thừa các khái niệm, thuộc tính và các mối quan hệ trong các ontology này để xây dựng ontology của mình.

### Bước 3: Liệt kê các thuật ngữ quan trọng trong Ontology

Một số khái niệm quan trọng trong lĩnh vực nhà thông minh:

- **Smart Context:** là một phạm vi triển khai hoặc một ứng dụng trong IoT. Đây sẽ là thành phần trên nhất trong cây phân cấp các khái niệm. Một smart context có thể chứa các smart context khác; một smart context cũng có thể nằm trong một smart context khác.
- **Platform:** là một phần mềm gồm nhiều thành phần, cho phép kết nối, quản lý và tự động hóa việc kết nối các thiết bị trong môi trường IoT. Nó kết nối các thiết bị phần cứng với nhau, cũng như với cloud bằng việc sử dụng linh hoạt các giao thức kết nối, cung cấp các cơ chế bảo mật và cung cấp năng lực xử lý dữ liệu.
- **Source:** Là một thiết bị vật lý hoặc thiết bị ảo. Source sẽ sinh ra dữ liệu về

môi trường hay bất kỳ thứ gì mà nó theo dõi. Một Source có thể là một Thing, Gateway hay một Process Unit.

- Thing: Là một thiết bị, bao gồm một hoặc một tập hợp các cảm biến
- Gateway: Là một thiết bị vật lý hay một trình phần mềm phục vụ như là một điểm kết nối giữa các thiết bị thông minh, các cảm biến với cloud.
- Container: Là một thiết bị ảo để theo dõi các tài nguyên trong một hệ thống IoT như CPU, RAM, ổ đĩa, ...
- Log: là dữ liệu được sinh ra từ Source. Các tệp tin log có thể được dùng để lấy ra các thông tin lịch sử của dữ liệu, trạng thái theo dõi các thiết bị, ...
- Metric được sử dụng để nhấn mạnh các loại độ đo khác nhau của dữ liệu trong IoT. Một số loại Metric có thể như Enumeration, Gauge, Counter, Histogram, Summary. Mỗi Metric có các đơn vị tương ứng như Percent, Time, Degree. Mỗi Source đều có một hoặc nhiều Metric của nó.
- Data Point: Được tạo ra khi một Source hoạt động. Dựa trên thông tin về Metric của một Source, dữ liệu sẽ có DataType, DataValue phù hợp.

Bước 4: Xây dựng các lớp/khái niệm và cấu trúc lớp/khái niệm phân cấp:

Xây dựng theo kiểu trên xuống, ta nhận thấy Smart Context là khái niệm có mức độ tổng quát cao nhất, là gốc của đồ thị. Các khái niệm Log, Platform, Source, Metric, Data Point là các khái niệm trung gian. Các khái niệm Thing, Gateway, Container, Percent, Time, Degree, ... là các khái niệm lá.

Bước 5: Định nghĩa các thuộc tính và quan hệ cho lớp:

Các thuộc tính của các khái niệm trong ontology: Smart context có các thuộc tính:

- SmartContextId: định danh duy nhất của một smart context
- SmartContextName: tên của một smartcontext

Platform có các thuộc tính:



- PlatformId: định danh duy nhất của một platform
- PlatformName: tên của platform
- PlatformType: loại platform
- PlatformHost: địa chỉ của máy cài đặt platform này
- PlatformPort: Cổng cài đặt platform
- PlatformStatus: trạng thái hoạt động của platform

Source có các thuộc tính:

- SourceId: Định danh toàn cục duy nhất của một Source
- EndPoint: Địa chỉ của Source
- SourceStatus: Trạng thái hoạt động của Source
- Description: Mô tả về Source
- SourceType: loại Source
- Label: nhãn của Source
- LocalId: Địa chỉ cục bộ của Source

Metric có các thuộc tính:

- MetricId: định danh toàn cục duy nhất của một Metric
- MetricName: tên của một Metric
- MetricLocalId: định danh cục bộ của một Metric
- MetricType: loại Metric
- Unit: đơn vị của Metric

- ValueDomain: trường giá trị để ánh xạ các khác nhau trong việc biểu diễn các giá trị dữ liệu của cùng một thiết bị nhưng trong các platform khác nhau. Ví dụ, cùng một cảm biến nhiệt độ, nhưng OpenHAB biểu diễn giá trị của nó là 0/1, HomeAssistant biểu diễn giá trị là on/off.

Data Point có các thuộc tính:

- DataPointId: định danh duy nhất của một Data Point
- DataType: Kiểu dữ liệu của một Data Point
- time: thời điểm Data Point được thu nhận
- value: giá trị của Data Point

Log có các thuộc tính:

- time: thời điểm ghi log
- message: nội dung của log

Mối quan hệ của các khái niệm: Smart Context

- isSubSmartContext: smart context nằm trong smart context khác
- hasSmartContextId: smart context chứa smart context khác.
- hasPlatform: smart context chứa platform nào.

Platform:

- hasSource: Platform chứa những Source nào

Source:

- hasMetric: Source chứa những Metric nào

Metric:

- hasDataPoint: Metric chứa những Data Point nào

Bước 6: Định nghĩa các ràng buộc của các thuộc tính Các ràng buộc của các thuộc tính: Smart context:

- SmartContextId: phải là một kiểu String
- SmartContextName: phải là một kiểu String

Platform:

- PlatformId: phải là một kiểu String
- PlatformName: phải là một kiểu String
- PlatformType: phải là một kiểu String
- PlatformHost: phải là một kiểu String
- PlatformPort: phải là một kiểu String
- PlatformStatus: phải là một kiểu String

Source:

- SourceId: phải là một kiểu String
- EndPoint: phải là một kiểu String
- SourceStatus: phải là một kiểu String
- Description: phải là một kiểu String
- SourceType: phải là một kiểu String
- Label: phải là một kiểu String

- LocalId: phải là một kiểu String

Metric:

- MetricId: phải là một kiểu String
- MetricName: phải là một kiểu String
- MetricLocalId: phải là một kiểu String
- MetricType: phải là một kiểu String
- Unit: có thể là kiểu của Percent, Time hoặc Degree
- ValueDomain: phải là một kiểu String

Data Point:

- DataPointId: phải là một kiểu String
- DataType: phải là một kiểu String
- time: phải là một kiểu datetime
- value: phải là một kiểu Number

Log có các thuộc tính:

- time: phải là một kiểu datetime
- message: phải là một kiểu String

Bước 7: Tạo ra các thực thể Để tạo ra thực thể cho các khái niệm, tôi dùng cú pháp json thay thế cho cú pháp bộ ba: đối tượng - thuộc tính - giá trị với ý nghĩa tương đương.

Một thực thể Smart context:

```

{
  "SmartContextId" : "HPCC_id",
  "SmartContextName" : "HPCC",
  "ParentSmartContextId" : [],
  "SubSmartContextId" : ["phong_sinh_vien_id", "phong_can_bo_id", "phong_may_chu_id"],
  "HasPlatform" : []
}

```

Hình 2.11: Thực thể smartcontext

Một thực thể Platform:

```

{
  "PlatformId" : "0bafb5bf-5c16-4122-9782-d240914705d3",
  "PlatformName" : "thingsboard",
  "PlatformType" : "",
  "PlatformHost" : "http://192.168.0.198",
  "PlatformPort" : "8080",
  "PlatformStatus" : "active",
  "HasSource" : ["temperature-humidity-light_thingsboard", "motion_thingsboard"]
}

```

Hình 2.12: Thực thể platform

Một thực thể Source:

```

{
  "SourceId" : "motion_homeassistant",
  "EndPoint" : "http://192.168.0.199:8123/source",
  "SourceStatus" : "active",
  "Description" : "",
  "SourceType" : "thing",
  "Label" : "sensor",
  "LocalId" : "motion_id",
  "HasMetric" : ["binary_sensor.motion_detection", "light.green_light", "light.red_light", "light.yellow_light"]
}

```

Hình 2.13: Thực thể Source

Một thực thể Metric:

```
{
  "MetricType": "gauge",
  "MetricLocalId": "b37a79a0-755c-11e9-abce-5bf295b292b2-humidity",
  "MetricId": "b37a79a0-755c-11e9-abce-5bf295b292b2-humidity",
  "CanSetState": "false",
  "HasDatapoint": [
    "b37a79a0-755c-11e9-abce-5bf295b292b2-humidity_datapoint"
  ],
  "MetricStatus": "active",
  "Unit": "",
  "MetricName": "humidity",
  "MetricDomain": "sensor"
}
```

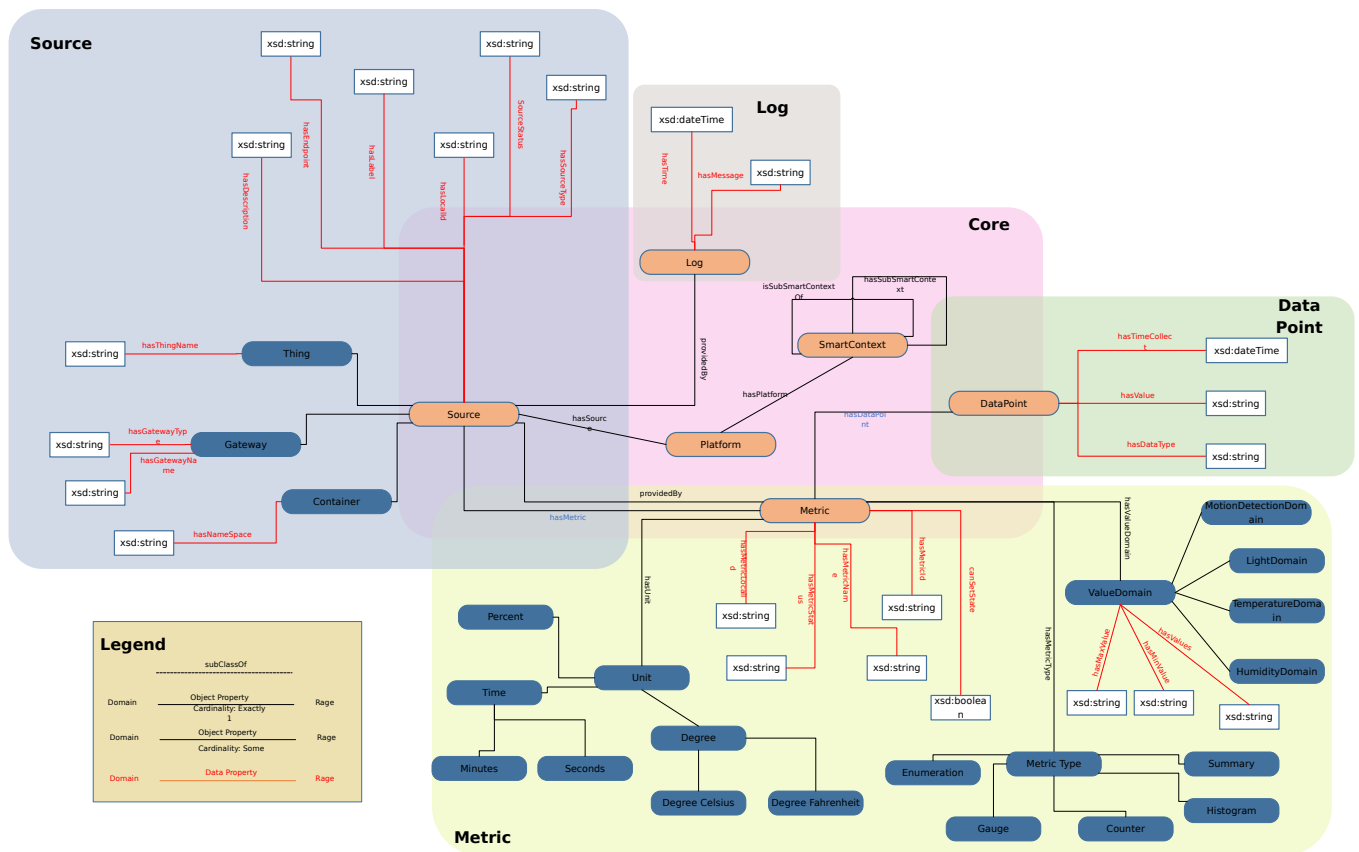
Hình 2.14: Thực thể Metric

Một thực thể Data Point:

```
{
  "DataType": "int",
  "time": "2019-05-13 18:21:26.213131",
  "DatapointId": "b37a79a0-755c-11e9-abce-5bf295b292b2-humidity_datapoint",
  "value": 70
}
```

Hình 2.15: Thực thể Data Point

Sau các bước trên, ta lập được một ontology về lĩnh vực IoT nhà thông minh:

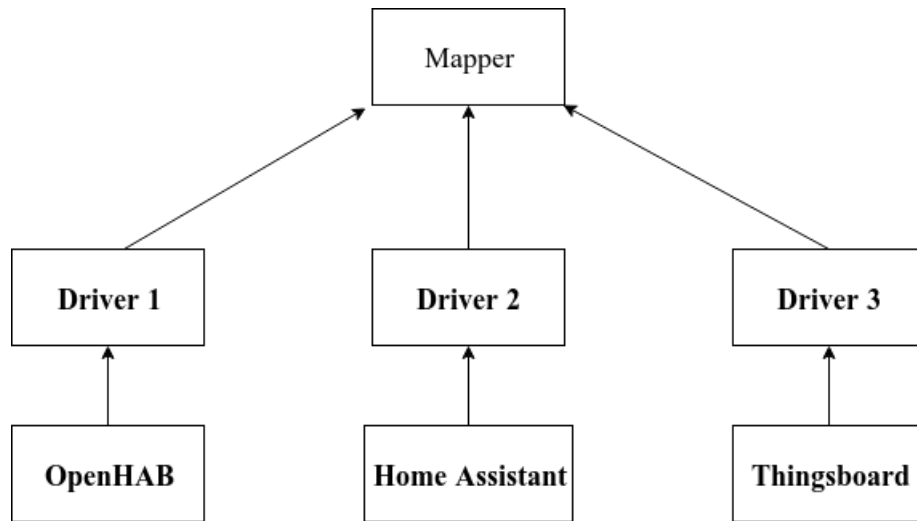


Hình 2.16: ontology cho lĩnh vực nhà thông minh

### 2.2.2 Cơ chế ánh xạ các dữ liệu trong IoT theo ontology

Các dữ liệu từ các cảm biến, thiết bị IoT rất khác nhau về cấu trúc. Do trong đề tài này, tất cả các cảm biến, thiết bị đều được gắn với một platform; mỗi platform biểu diễn dữ liệu thu được từ các cảm biến, thiết bị theo một định dạng chuẩn riêng ta phải ánh xạ các định dạng dữ liệu chuẩn của các platform về một dạng chung. Để làm được điều này, ta phải viết các driver cho mỗi platform, nhiệm vụ của driver là ánh xạ định dạng dữ liệu của mỗi platform về định dạng dữ liệu chung của ontology. Sau khi đã có một định dạng dữ liệu thống nhất, ta phải dùng ngôn ngữ hình thức để biểu diễn các

khái niệm trong ontology. Ở đề tài này, tôi dùng định dạng json để biểu diễn các khái niệm, các thuộc tính của các khái niệm cũng như mối quan hệ giữa các khái niệm.



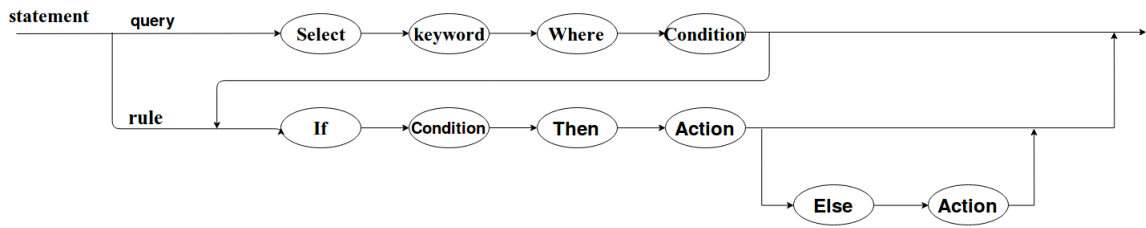
Hình 2.17: Mô hình ánh xạ dữ liệu về một chuẩn của ontology

### 2.2.3 Cơ chế truy vấn ngữ nghĩa

Từ các thực thể được biểu diễn theo ontology, ta phải xây dựng cơ chế để lấy được thông tin của các thực thể, mối quan hệ giữa các thực thể. Để làm được điều này, tôi xây dựng tập các API dựa vào mối quan hệ giữa các khái niệm để đạt được tính ngữ nghĩa. Đồng thời, tôi định nghĩa một ngôn ngữ truy vấn đơn giản, sử dụng tập API trên để tạo ra các câu truy vấn ngữ nghĩa.

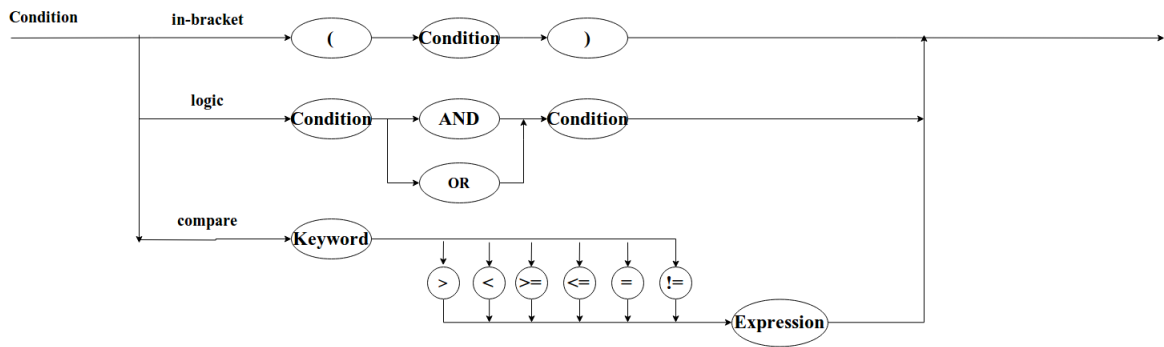
Mục đích của ngôn ngữ muốn tạo ra là tạo được các câu truy vấn, tương ứng với nhanh query trong sơ đồ. Để thực hiện câu truy vấn, ta cần chỉ rõ muốn lấy thuộc tính hay khái niệm nào, tập các thuộc tính và khái niệm được gọi là keyword. Ngoài việc chỉ rõ các thuộc tính, khái niệm cần truy vấn, ta cũng cần chỉ rõ điều kiện thực hiện câu truy vấn. Điều kiện này được chứa trong thành phần "Condition". Một chức năng khác mà ta mong muốn ngôn ngữ có thể thực hiện là kiểm tra các điều kiện, nếu điều kiện thỏa mãn thì thực hiện một hành động nào đó, hành động này tương ứng với việc điều khiển các thiết bị IoT dựa trên các API mà các platform cung cấp. Chức năng này được gọi là rule. Từ các mục đích trên, ta xây dựng được cấu trúc của ngôn ngữ truy vấn:





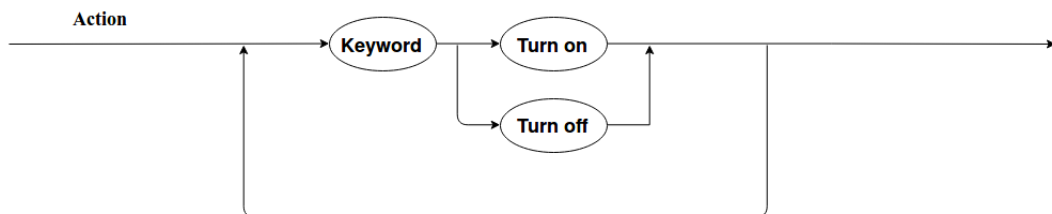
Hình 2.18: Mô hình ngôn ngữ - statement

Một condition là một phép so sánh giữa giá trị tương ứng một keyword với một biểu thức. Ví dụ so sánh PlatformId = "OpenHAB\_id". Các phép so sánh có thể có là <, >, >=, <=, =, !=. Tùy thuộc vào keyword mà có phép so sánh tương ứng. Condition cũng có thể là kết quả của một phép toán logic của một tập hợp các condition khác. Hai phép toán logic hỗ trợ là AND và OR. Ngoài ra, để thể hiện thứ tự ưu tiên của các phép toán logic, ta đưa thêm cặp dấu ngoặc đơn vào ngôn ngữ.



Hình 2.19: Mô hình ngôn ngữ - condition

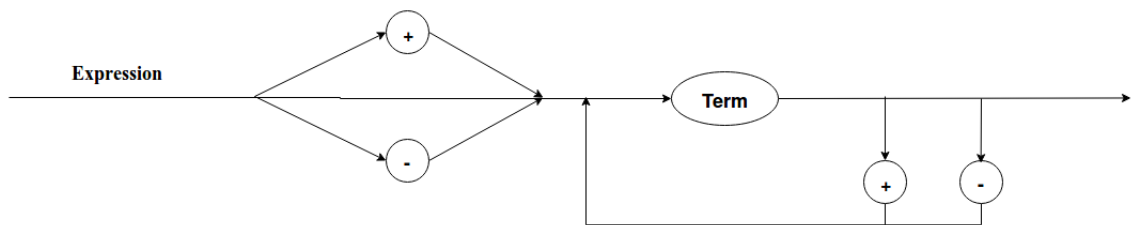
Một Action (hành động) là một lời gọi tới các API để điều khiển các thiết bị từ IoT platform. Do các thiết bị trong hệ thống chỉ cung cấp hai thao tác điều khiển là bật và tắt nên ngôn ngữ chỉ đưa vào hai hành động là bật và tắt.



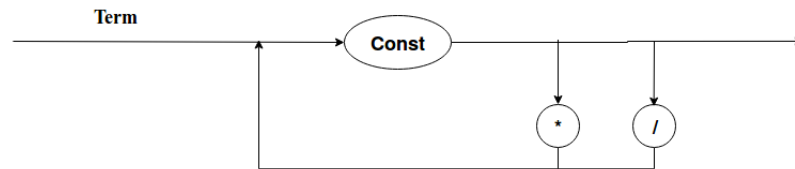
Hình 2.20: Mô hình ngôn ngữ - action

Một biểu thức là một toán tử hoặc các phép toán cộng, trừ các toán tử.

Một toán tử là một hằng số hoặc các phép toán của các hằng số.

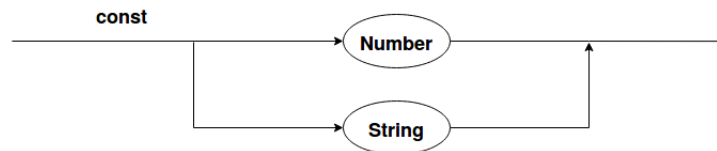


Hình 2.21: Mô hình ngôn ngữ - expression



Hình 2.22: Mô hình ngôn ngữ - term

Một hằng số là một chuỗi hoặc một số (nguyên hoặc thực).



Hình 2.23: Mô hình ngôn ngữ -const

### 2.2.4 Tập các API

Để cài đặt cơ chế ngữ nghĩa, tôi đã xây dựng tập các API, các API này dựa vào mối quan hệ giữa các khái niệm trong ontology để thực hiện các câu truy vấn ngữ nghĩa.

Các API cơ bản:

- Lấy tất cả các Data Point
- Lấy tất cả các Metric
- Lấy tất cả các Source
- Lấy tất cả các Platform
- Lấy tất cả các Smart Context
- Lấy tất cả các Smart Context nằm trong Smart Context nào đó. (lấy smart context con)
- Lấy tất cả các Smart Context chứa Smart Context nào đó. (lấy smart context cha)

Một số API phụ trợ:

- Lấy DataPointId/DataPoint của một DataPoint
- Lấy MetricId/Metric của một Metric
- Lấy SourceId/Source của một Source
- Lấy PlatformId/Platform của một Platform
- Lấy SmartContextId/SmartContext của một SmartContext

Các API thể hiện tính ngữ nghĩa dựa trên mối quan hệ giữa các khái niệm:

- Lấy DataPoint từ một Metric

- Lấy DataPoint từ một Source
- Lấy DataPoint từ một Platform
- Lấy DataPoint từ một SmartContext
- Lấy Metric từ một DataPoint
- Lấy Metric từ một Source
- Lấy Metric từ một Platform
- Lấy Metric từ một SmartContext
- Lấy Source từ một DataPoint
- Lấy Source từ một Metric
- Lấy Source từ một Platform
- Lấy Source từ một SmartContext
- Lấy Platform từ một DataPoint
- Lấy Platform từ một Metric
- Lấy Platform từ một Source
- Lấy Platform từ một SmartContext
- Lấy SmartContext từ một DataPoint
- Lấy SmartContext từ một Metric
- Lấy SmartContext từ một Source
- Lấy SmartContext từ một Platform

Ví dụ về cơ chế ngữ nghĩa trong API: Lấy DataPoint từ một SmartContext. Để lấy được DataPoint thuộc vào một SmartContext, trước hết ta phải lấy ra các Platform mà Smart Context đó chứa dựa vào mối quan hệ hasPlatform của SmartContext. Khi

đã xác định được các Platform, với mỗi Platform, xác định xem nó chứa các Source nào dựa vào mối quan hệ hasSource của Platform. Với mỗi Source, xác định xem nó chứa các Metric nào dựa vào mối quan hệ hasMetric của Source. Với mỗi Metric, xác định DataPoint mà nó chứa dựa vào thuộc tính hasDataPoint của Metric. Vậy sau khi duyệt qua các quan hệ trên ontology, ta sẽ lấy ra được các DataPoint nằm trong một SmartContext. Cơ chế truy vấn ngữ nghĩa này hoàn toàn tương tự đối với các API khác được xây dựng.

## Chương 3

# Kết luận, hướng phát triển

Sau khi đã thực hiện các công việc như đã trình bày trong chương 2, tôi đã xây dựng được một ngôn ngữ truy vấn có khả năng dựa trên mối quan hệ giữa các khái niệm trong ontology để thực hiện các câu truy vấn thể hiện được tính ngữ nghĩa, chính xác hơn với ngữ cảnh mà người truy vấn mong muốn. Tuy nhiên, do giới hạn về thời gian, hệ thống xây dựng còn khá đơn giản, chưa xử lý được các câu truy vấn ngữ nghĩa phức tạp. Do đó, trong thời gian tới, tôi sẽ tiếp tục phải cải tiến cấu trúc ngôn ngữ truy vấn xây dựng, đồng thời chỉnh sửa ontology nếu cần thiết để phù hợp hơn với những phát sinh trong tương lai. Phần mã nguồn của tôi tuy đã cố gắng viết sạch, dễ nhìn và cũng đã thực hiện unit test cho các API, tuy nhiên vẫn chưa kiểm tra hết được các khả năng có thể xảy ra lỗi. Do đó, trong tương lai, tôi cũng sẽ bổ sung các unit test cho toàn bộ phần mã nguồn quan trọng để đảm bảo hệ thống có ít lỗi nhất trong khả năng của mình.

# Chương 4

## Tài liệu tham khảo

1. Natalya F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
2. Nguyễn Hữu Nhật, Tổng quan về ontology, Đồ án tốt nghiệp, Đại học quốc gia thành phố Hồ Chí Minh, thành phố Hồ Chí Minh
3. Binh Minh Nguyen, Huan Phan, Duong Quang Ha, Giang Nguyen. An Information-centric Approach for Slice Monitoring from Edge Devices to Clouds, The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops, volum 130, Pages 326-335, Elsevier Science Publishers, 2018
4. internet of things (IoT), <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>, last visited May 2019
5. 11 Internet of Things (IoT) Protocols You Need to Know About, <https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about>, last visited May 2019

6. IoT security (internet of things security), <https://internetofthingsagenda.techtarget.com/definition/IoT-security-Internet-of-Things-security>, last visited May 2019