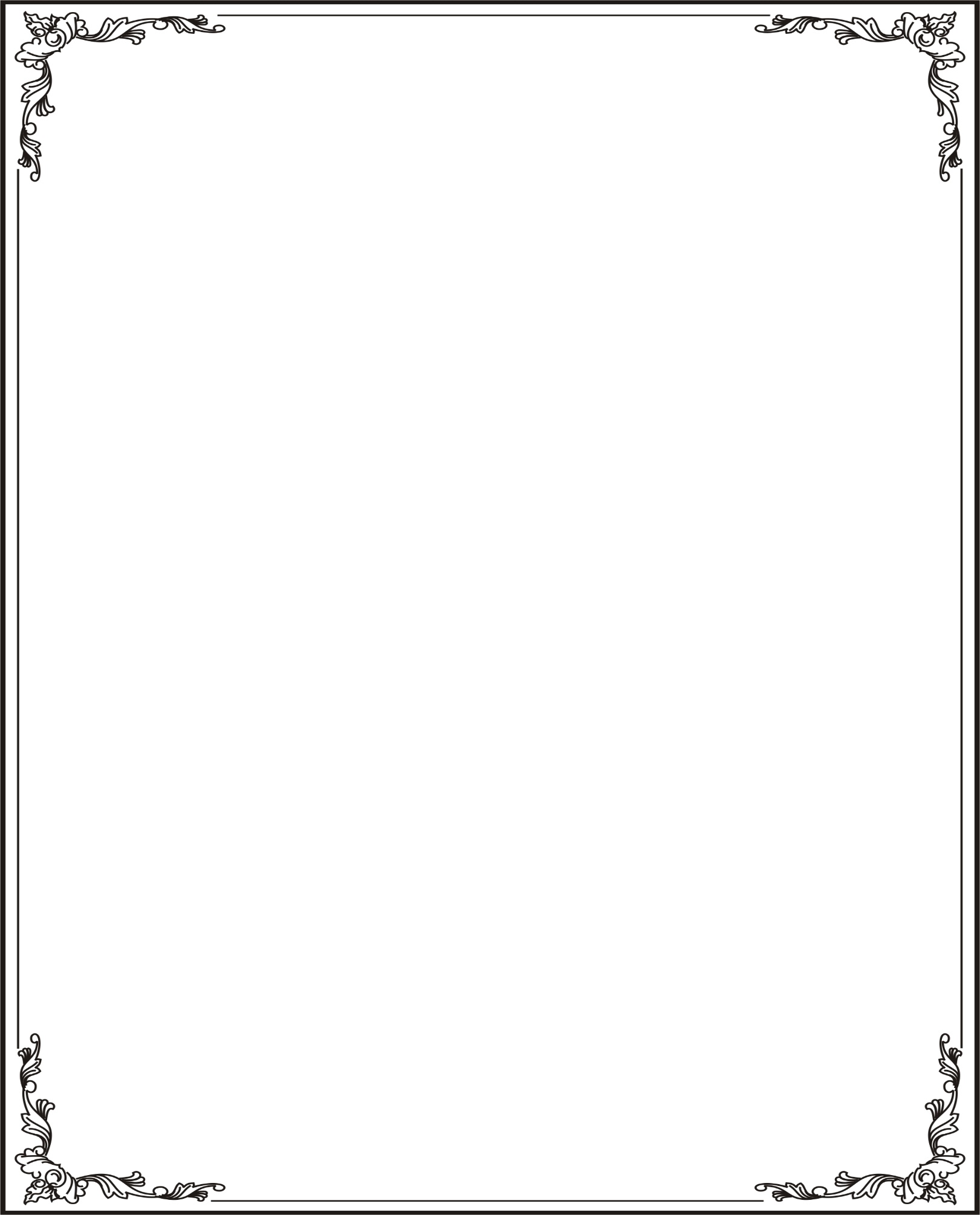
**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH: KỸ THUẬT PHẦN MỀM**

**ĐỀ TÀI: XÂY DỰNG GIẢI PHÁP NHÀ THÔNG MINH SỬ DỤNG KIT WEMOS D1 R2**

Sinh viên thực hiện: NGUYỄN MẠNH TÙNG

Lớp KTMP3 – K10

Giảng viên hướng dẫn: TS Phạm Văn Hà

**Hà Nội, 4/2019**

[**PHẦN I. MỞ ĐẦU** 8](#_Toc7119988)

[**PHẦN II. NỘI DUNG** 12](#_Toc7119989)

[**CHƯƠNG I. TỔNG QUAN VỀ INTERNET OF THINGS VÀ HỆ THỐNG NHÀ THÔNG MINH** 12](#_Toc7119990)

[**1.1** **Tổng quan về Internet of Things** 12](#_Toc7119991)

[**1.1.1** **Khái niệm Internet of Things** 12](#_Toc7119992)

[**1.1.2** **IOT từ góc nhìn kỹ thuật** 13](#_Toc7119993)

[**1.1.3** **Đặc trưng của các hệ thống IoT** 17](#_Toc7119994)

[**1.1.3.1** **Đặc tính cơ bản** 17](#_Toc7119995)

[**1.1.3.2** **Yêu cầu ở mức cao đối với một hệ thống IoT** 18](#_Toc7119996)

[**1.1.4** **Mô hình của một hệ thống IoT** 19](#_Toc7119997)

[**1.1.4.2** **Service support and application support layer(Tầng hỗ trợ dịch vụ và hỗ trợ ứng dụng)** 20](#_Toc7119998)

[**1.1.4.3** **Network Layer(Tầng mạng)** 20](#_Toc7119999)

[**1.1.4.4** **Device Layer(Tầng thiết bị)** 21](#_Toc7120000)

[**1.2** **Tìm hiểu hệ thống nhà thông minh** 22](#_Toc7120001)

[**1.2.1** **Giới thiệu chung về nhà thông minh** 22](#_Toc7120002)

[**CHƯƠNG II. XÂY DỰNG MÔ HÌNH MÔ PHỎNG HỆ THỐNG NHÀ THÔNG MINH ĐƠN GIẢN** 27](#_Toc7120004)

[**2.1** **Tìm hiểu về Arduino** 27](#_Toc7120005)

[**2.1.1**  **Lịch sử** 27](#_Toc7120006)

[**2.1.2** **Giới thiệu** 27](#_Toc7120007)

[**2.1.3** **Phần cứng** 28](#_Toc7120008)

[**2.1.4** **Các Broad chính thức** 30](#_Toc7120009)

[**2.1.5** **Shield(Bo mạch mở rộng)** 31](#_Toc7120010)

[**2.1.6** **Môi trường phát triển** 32](#_Toc7120011)

[**2.2**  **KIT ESP 8266 WeMos D1 R2** 35](#_Toc7120012)

[**2.2.1** **Làm quen với KIT ESP8266 Wemos D1 R2** 36](#_Toc7120013)

[**2.2.2** **Các ứng dụng của KIT WeMos D1 R2** 39](#_Toc7120014)

[**2.3** **Một số cảm biến sử dụng** 40](#_Toc7120015)

[**2.3.1** **Cảm biến nhiệt độ-độ ẩm DHT11** 40](#_Toc7120016)

[**2.4** **Tìm hiểu về kiến trúc Webservice-RESTful, Webservice với Spring framework.** 41](#_Toc7120017)

[**2.4.1 Tìm hiểu về Webserver** 41](#_Toc7120018)

[**2.4.2** **Webservice** 43](#_Toc7120019)

[**2.4.3** **RESTFul webservice** 45](#_Toc7120022)

[**2.4.4** **Website** 48](#_Toc7120028)

[**2.4.5** **Webservice với Spring framework.** 49](#_Toc7120030)

[**2.4.5.1 Giới thiệu.** 49](#_Toc7120031)

[**2.4.5.2 Spring Rest webservie.** 49](#_Toc7120032)

[**2.5 Xây dựng hệ thống nhà thông minh** 53](#_Toc7120036)

[**2.5.1 Mô tả hệ thống** 54](#_Toc7120037)

[**2.5.2** **Hoạt động của hệ thống** 55](#_Toc7120041)

[**2.5.3** **Vai trò của ESP8266** **Wemos D1 R2** 56](#_Toc7120042)

[**2.5.4**  **Hệ thống Webserver và Webservice** 56](#_Toc7120043)

[**2.5.5**  **Website(Hệ thống View) với Angular 7** 60](#_Toc7120044)

**DANH MỤC CÁC HÌNH ẢNH**

Hình 1.1 Các khía cạnh của IoT……………………………………………..14

Hình 1.2 Hệ thống IoT từ góc nhìn kỹ thuật…………………………………15

Hình 1.3: Các loại thiết bị khác nhau và mối quan hệ………………………..16

Hình 1.4 Mô hình IoT………………………………………………………...20

Hình 2.1 Một mạch Arduino Uno chính thức với các mô tả về các cổng I/O 27

Hình 2.2 Một board Arduino đời đầu gồm một cổng giao tiếp RS-232 (góc phía trên-bên trái) và một chip vi xử lý Atmel ATmega8 (màu đen, nằm góc phải-phía dưới); 14 chân I/O số nằm ở phía trên và 6 chân analog đầu vào ở phía đáy…………………………………………………………………………...28

Hình 2.3 Các shield Arduino điển hình………………………………………32

Hình 2.4 Một chương trình Arduino đơn giản……………………………......34

Hình 2.5 Một module tích hợp phổ biến (Module ESP12E)…………………36

Hình 2.6 Mặt trên của KIT ESP8266 Wemos D1 R2…………………………37

Hình 2.7 Sơ đồ Pin trên KIT………………………………………………….38

Hình 2.8 Sự khác biệt khi bố trí Pin trên UNO và Wemos-D1 R2…………..39

Hình 2.9 Cảm biến nhiệt độ độ ẩm DHT11…………………………………..40

Hình 2.10 Mô hình hoạt động của webserver………………………………...42

Hình 2.12 Kiến trúc cơ bản của webservice…………………………………..44

Hình 2.13 Khác nhau giữa Webservice và RESTFul…………………………45

Hình 2.14 Giao diện website cơ bản………………………………………….48

Hình 2.15 Hoạt động của hệ thống…………………………………………...55

Hình 2.16 Entity model………………………………………………………59

Hình 2.17 Một class ở tầng Persistence………………………………………59

Hình 2.18 Một class ở tầng Controller………………………………………..60

Hình 2.19 Một Class ở tầng Service………………………………………….60

Hình 2.20 Giao diện trang chủ với những thông số cụ thể trong nhà………….61

Hình 2.21 Xử lý dữ liệu với Angular………………………………………....63

**LỜI NÓI ĐẦU**

Ngày nay khoa học kỹ thuật trên thế giới nói chung, ở Việt Nam nói riêng đang trên đà phát triển mạnh mẽ và không ngừng nâng cao phát triển về mọi mặt. Đặc biệt là công nghệ thông tin với cuộc cách mạng công nghiệp 4.0. Cách mạng Công nghiệp 4.0 (Industry 4.0) tạo ra môi trường mà máy tính, tự động hoá và con người sẽ làm việc cùng nhau theo những cách thức hoàn toàn mới. Tại đây, robot và các loại máy móc sẽ được kết nối vào những hệ thống máy tính, những hệ thống này sẽ sử dụng các thuật toán để điều khiển mà không cần sự can thiệp của con người.

**Nhà thông minh** (tiếng Anh là “Smart Home”) hoặc hệ thống nhà thông minh là một ngôi nhà/căn hộ được trang bị hệ thống tự động tiên tiến dành cho điều khiển đèn chiếu sáng, nhiệt độ, truyền thông đa phương tiện, an ninh, rèm cửa, cửa và nhiều tính năng khác nhằm mục đích làm cho cuộc sống ngày càng tiện nghi, an toàn và góp phần sử dụng hợp lý các nguồn tài nguyên. Nhà thông minh sử dụng các thiết bị kết nối Internet để có thể quản lý và giám sát các thiết bị và hệ thống từ xa như ánh sáng và nhiệt độ. Công nghệ nhà thông minh Smart Home, còn được biết đến như **Home Automation** (tự động hóa ngôi nhà), cung cấp cho người dùng sự an toàn, thoải mái, tiện lợi và tiết kiệm năng lượng bằng cách cho phép họ kiểm soát các thiết bị thông minh bởi một ứng dụng smarthome trên điện thoại smartphone hoặc các thiết bị kết nối mạng khác. Một phần của mạng lưới vạn vật kết nối (Internet of Things – IoT), các hệ thống và thiết bị nhà thông minh thường hoạt động cùng nhau, chia sẻ dữ liệu người dùng và tự động hóa các hành động dựa trên quyền ưu tiên của người dùng.

Dựa vào mô hình cụ thể cùng kiến thức đã có, em đã tìm hiểu và đi sâu vào nghiên cứu, thực hiện đề tài “***Xây dựng giải pháp nhà thông minh sử dụng Kit Wemos D1 R2***”.

# **PHẦN I. MỞ ĐẦU**

1. **Lý do chọn đề tài.**

Cách mạng Công nghiệp 4.0 đang diễn ra tại nhiều nước phát triển. Nó mang đến cho nhân loại cơ hội để thay đổi bộ mặt các nền kinh tế. Là xu hướng hiện thời trong việc [tự động hóa](https://vi.wikipedia.org/wiki/T%E1%BB%B1_%C4%91%E1%BB%99ng_h%C3%B3a) và trao đổi dữ liệu trong công nghệ sản xuất. Nó bao gồm các hệ thống không gian mạng thực-ảo (*cyber-physical system*),  và [điện toán đám mây](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_to%C3%A1n_%C4%91%C3%A1m_m%C3%A2y)

Đi cùng với sự phát triển của xã hội loài người, những cuộc cách mạng về khoa học kỹ thuật đóng một vai trò rất quan trọng, chúng đang từng ngày từng giờ làm thay đổi cuộc sống của con người, theo hướng hiện đại hơn. Tuy nhiên chính những thay đổi đó đã tác động đến tự nhiên, đến môi trường sống và hiện tại chúng ta đang phải hứng chịu sự tác động trở lại của môi trường như ô niễm môi trường, thay đổi khí hậu,…Dân số tăng, nhu cầu của con người cũng tăng theo. Vì vậy việc áp dụng khoa học kỹ thuật như các ngành công nghệ thông tin, điện tử viễn thông, điện – điện tử, tự động hóa,… vào sản xuất trở nên quan trọng. Thuật ngữ "Internet of Things" (IoT) lần đầu tiên được sử dụng vào năm 1999 bởi nhà tiên phong công nghệ người Anh Kevin Ashton để mô tả một hệ thống trong đó các vật thể trong thế giới vật lý có thể kết nối với Internet. Ngày nay, Internet of Things đã trở thành một thuật ngữ phổ biến và việc áp dụng công nghệ Internet of Things vào sản xuất nông nghiệp trở thành một nhu cầu không thể thiếu. Ứng dụng IoT là một xu hướng hiện đại phù hợp với yêu cầu của cuộc cách mạng công nghiệp 4.0 mà Đảng và Nhà nước đang rất trú trọng và quan tâm. Bên cạnh đó, IoT đang ở giai đoạn chín muồi và chúng ta đang được thừa hưởng những thành tựu của công nghệ này. Nó giúp ích cho việc triển khai hệ thống như giảm chi phí, tiết kiệm thời gian và công sức. Ngoài ra, bản thân xuất phát từ một gia đình làm nông nghiệp em thấu hiểu được những vất vả của người nông dân vì vậy ý tưởng áp dụng công nghệ thông tin vào sản xuất đã ấp ủ trong suy nghĩ em từ rất lâu với hy vọng có thể giúp ích cho gia đình và xã hội. Ở trên các nước phát triển đã không còn xa lạ với các hệ sinh thái thông minh, các nhà thông minh nơi mà ở đó con người có thể tận hưởng cuộc sống với sự giúp đỡ tận tình của các hệ thống máy tính và cảm biến, hầu hết tất cả các hoạt động trong ngôi nhà sẽ được thực hiện một cách tự động bởi máy tính nhưng vẫn sẽ ưu tiên quyền của con người. Trong căn nhà thông minh, đồ dùng trong nhà từ phòng ngủ, phòng khách đến toilet đều gắn các bộ điều khiển điện tử có thể kết nối với [Internet](https://vi.wikipedia.org/wiki/Internet) và [điện thoại di động](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_tho%E1%BA%A1i_di_%C4%91%E1%BB%99ng), cho phép chủ nhân điều khiển vật dụng từ xa hoặc lập trình cho thiết bị ở nhà hoạt động theo lịch. Thêm vào đó, các đồ gia dụng có thể hiểu được ngôn ngữ của nhau và có khả năng tương tác với nhau.

Chính vì lý do đó mà em quyết định chọn đề tài: *“****Xây dựng giải pháp nhà thông minh sử dụng Kit Wemos D1 R2****”.* Đây là một đề tài hấp dẫn, cập nhật công nghệ, có tính ứng dụng cao,phù hợp và thiết thực sau quá trình đi thực tập tốt nghiệp.

1. **Tính cấp thiết của đề tài.**

Hiện nay nhà thông minh áp dụng ở Việt Nam mới bắt đầu và chưa có nhiều, các công ty đều là đại diện của nước ngoài, việc hiểu biết về nhà thông minh cũng như áp dụng như thế nào ở Việt Nam cho hiệu quả, hướng nghiên cứu để phát triển và nội địa hóa, tận dụng lợi thế của Việt Nam là rất cần thiết.Cuộc sống của con người ngày càng có nhu cầu cao, mặt khác các nguồn năng lượng và vật liệu đang cạn kiệt dần do đó nhu cầu ở trong một ngôi nhà thông minh là rất cần thiết nhằm giảm thiểu tiêu hao năng lượng, tận dụng nguồn nguyên vật liệu lợi thế của địa phương.

Hiện nay công nghệ nhà thông minh trên thế giới ứng dụng một thời gian và Việt Nam cũng đã áp dụng ở một số công trình lớn cũng như nhà ở gia đình. Việc áp dụng nhà ở thông minh là cần thiết và thực tế để đạt được. Không nhất thiết cứ phải áp dụng toàn bộ công nghệ mới là nhà thông minh. Nhà thông minh ở đây có thể hiểu là áp dụng kiến trúc truyền thống phù hợp khí hậu Việt Nam như nào cho hiệu quả hoặc áp dụng một phần của công nghệ, hoặc dùng các giải pháp tiết kiệm và phát triển bền vững năng lượng tận dụng những lợi thế của Việt Nam (nước ta có ánh sáng và gió nhiều do ở vùng biển nhiệt đới.) -Những mong đợi từ kết quả nghiên cứu của đề tài: Đề tài sẽ góp một phần trong việc tiếp cận nhà ở thông minh của các ngành khoa học liên quan và là một trong những tài liệu hướng dẫn áp dụng nhà thông minh một cách hiệu quả ở Việt Nam, đóng góp một phần nhỏ vào quá trình công nghiệp hóa hiện đại hóa của đất nước.

1. **Mục đích nghiên cứu của đề tài**

Tìm hiểu tổng quan về nhà thông minh nhằm nâng cao hiểu biết về các công nghệ nhà thông minh tiên tiến trên thế giới.

Điều khiển đèn,quạt,rèm... bằng vi điều khiển theo 2 chế độ tự động hoặc bán tự động. Hiện thị nhiệt độ,độ ẩm,… cho người dùng biết

Nghiên cứu giải pháp lưu trữ thông tin trên cloud và điều khiển thiết bị đa nền tảng

Ứng dụng xây dựng mô phỏng hệ thống nhà thông minh đơn giản.

1. **Phương pháp nghiên cứu**

Phương pháp nghiên cứu chính được sử dụng trong báo cáo là phương pháp thực nghiệm. Bên cạnh đó báo cáo còn sử dụng các phương pháp thống kê, phân tích số liệu thu thập được để đưa ra các giải pháp điều khiển phù hợp.

# **PHẦN II. NỘI DUNG**

## **CHƯƠNG I. TỔNG QUAN VỀ INTERNET OF THINGS VÀ HỆ THỐNG NHÀ THÔNG MINH**

* 1. **Tổng quan về Internet of Things**
     1. **Khái niệm Internet of Things**

Thuật ngữ ” Internet of things”( viết tắt là IoT) dạo gần đây xuất hiện khá nhiều và thu hút không ít sự quan tâm chú ý của thế giới công nghệ. Vì sự bùng nổ của IoT trong tương lai sẽ có tác động mãnh mẽ tới cuộc sống, công việc và xã hội loài người.Thực tế, Internet of things đã manh nha từ nhiều thập kỹ trước. Tuy nhiên mãi đến năm 1999 cụm từ IoT mới được đưa ra bởi Kevin Ashton , Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT, nơi thiết lập các quy chuẩn toàn cầu cho RFID (một phương thức giao tiếp không dây dùng sóng radio) cũng như một số loại cảm biến khác.

Mạng lưới vạn vật kết nối Internet hoặc là Mạng lưới thiết bị kết nối Internet viết tắt là IoT (tiếng Anh: Internet of Things) là một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và Internet. Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

IoT có thể được coi là một tầm nhìn sâu rộng của công nghệ và cuộc sống. Từ quan điểm của tiêu chuẩn kỹ thuật, IoT có thể được xem như là một cơ sở hạ tầng mang tính toàn cầu cho xã hội thông tin, tạo điều kiện cho các dịch vụ tiên tiến thông qua sự liên kết các “Things”. IoT dự kiến sẽ tích hợp rất nhiều công nghệ mới, chẳng hạn như các công nghệ thông tin machine-to-machine, mạng tự trị, khai thác dữ liệu và ra quyết định, bảo vệ sự an ninh và sự riêng tư, điện toán đám mây. Như hình dưới, một hệ thống thông tin trước đây đã mang đến 2 chiều – “Any TIME” và “Any PLACE” communication. Giờ IoT đã tạo thêm một chiều mới trong hệ thống thông tin đó là “Any THING” Communication (Kết nối mọi vật).

Hay hiểu một cách đơn giản IoT là tất cả các thiết bị có thể kết nối với nhau . Việc kết nối thì có thể thực hiện qua Wi-Fi, mạng viễn thông băng rộng (3G, 4G), Bluetooth, ZigBee, hồng ngoại… Các thiết bị có thể là điện thoại thông minh, máy pha cafe, máy giặt, tai nghe, bóng đèn, và nhiều thiết bị khác. Cisco, nhà cung cấp giải pháp và thiết bị mạng hàng đầu hiện nay dự báo: Đến năm 2020, sẽ có khoảng 50 tỷ đồ vật kết nối vào Internet, thậm chí con số này còn gia tăng nhiều hơn nữa. IoT sẽ là mạng khổng lồ kết nối tất cả mọi thứ, bao gồm cả con người và sẽ tồn tại các mối quan hệ giữa người và người, người và thiết bị, thiết bị và thiết bị. Một mạng lưới IoT có thể chứa đến 50 đến 100 nghìn tỉ đối tượng được kết nối và mạng lưới này có thể theo dõi sự di chuyển của từng đối tượng. Một con người sống trong thành thị có thể bị bao bọc xung quanh bởi 1000 đến 5000 đối tượng có khả năng theo dõi.

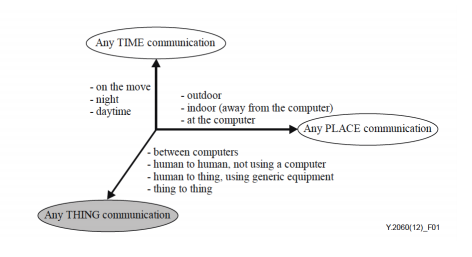
* + 1. **IOT từ góc nhìn kỹ thuật**

IoT được nhắc đến nhiều trong vài năm gần đây. Theo dự báo của hãng Ericsson trong báo cáo di động tháng 6/2016, trong vòng 2 năm tới, số lượng thiết bị kết nối vào mạng IoT sẽ vượt tổng số điện thoại di động đang hoạt động trên toàn cầu và trở thành phân khúc thiết bị kết nối lớn nhất. Tuy được nhắc đến nhiều, được dự đoán là tăng trưởng lớn nhưng khái niệm về IoT ít được nhắc đến. Tháng 6/2012, tổ chức viễn thông thế giới (ITU) đã đưa ra một số khái niệm về IoT, các đặc trưng cơ bản, các yêu cầu, mô hình tham chiếu của IoT trong khuyến nghị ITU-T Y2060.

Một số định nghĩa đã được tổ chức ITU đưa ra như sau:

* IoT là một kiến trúc toàn cầu của xã hội thông tin, có khả năng cung cấp dịch vụ đặc biệt bằng cách kết nối các đối tượng (đối tượng vật lý hoặc đối tượng đã được ảo hóa) dựa trên công nghệ truyền thông và thông tin đã có, đang phát triển.
* Device: là một phần của thiết bị, bắt buộc phải có khả năng truyền thông và có các khả năng tùy chọn như cảm biến, thực thi, thu thập dữ liệu, xử lý dữ liệu, lưu trữ dữ liệu.
* Thing: là đối tượng của thế giới thật hoặc đối tượng của thế giới ảo, có khả năng định danh, và tích hợp trong mạng truyền thông.

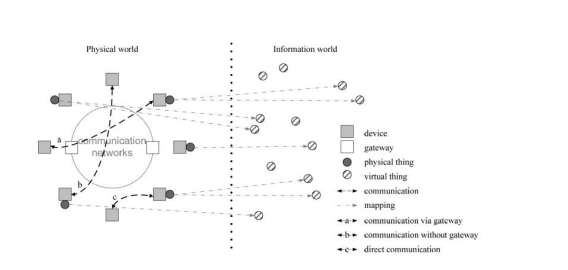
IoT có 3 khía cạnh: bất kỳ đối tượng, bất kỳ thời gian, bất kỳ địa điểm như hình



Hình 1.1 Các khía cạnh của IoT

Trong hệ thống IoT, “Things” là đối tượng của thế giới vật chất (Physical) hoặc các thông tin (Virtual). “Things” có khả năng nhận diện và có thể tích hợp vào mạng thông tin. “Things” có liên quan đến thông tin, có thể là tĩnh hay động. “Physical Things” t n tại trong thế giới vật lý và có khả năng được cảm nhận, được kích thích và kết nối. Ví dụ về “Physical Things” bao g m các môi trường xung quanh, robot công nghiệp, hàng hóa, hay thiết bị điện. “Virtual Things” t n tại trong thế giới thông tin và có khả năng được lưu trữ, xử lý, hay truy cập. Ví dụ về “Virtual Things” bao g m các nội dung đa phương tiện và các phần mềm ứng dụng.

Như đề cập ở 1.1, “Things” trong IoT có thể là đối tượng vật lý (Physical) hoặc là đối tượng thông tin (hay còn gọi là đối tượng ảo – Virtual). Hai loại đối tượng này có thể ánh xạ (mapping) qua lại lẫn nhau. Một đối tượng vật lý có thể được trình bày hay đại diện bởi một đối tượng thông tin, tuy nhiên một đối tượng thông tin có thể t n tại mà không nhất thiết phải được ánh xạ từ một đối tượng vật lý nào.

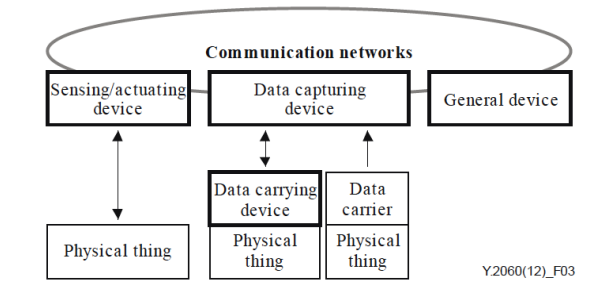


Hình 1.2 Hệ thống IoT từ góc nhìn kỹ thuật

Trong hình 1.2, một “device” là một phần của hệ thống IoT. Chức năng bắt buộc của một thiết bị là giao tiếp, và chức năng không bắt buộc là cảm biến, thực thi, thu thập dữ liệu, lưu trữ dữ liệu và xử lý dữ liệu. Các thiết bị thu thập các loại thông tin khác nhau và cung cấp các thông tin đó cho các network khác nơi mà thông tin được tiếp tục xử lý. Một số thiết bị cũng thực hiện các hoạt động dựa trên thông tin nhận được từ network. Truyền thông thiết bị - thiết bị: Có 3 cách các devices sẽ giao tiếp lẫn nhau. (a) Các thiết bị giao tiếp thông qua các mạng lưới thông tin liên lạc gọi là gateway, hoặc (b) các thiết bị tiếp qua mạng lưới thông tin liên lạc mà không có một gateway, hoặc (c) các thiết bị iên lạc trực tiếp với nhau qua mạng nội bộ.

Trong hình 1.2, mặc dù ta thấy chỉ có sự tương tác diễn ra ở Physical Things (các thiết bị giao tiếp với nhau). Thực ra vẫn còn hai sự tương tác khác đồng thời diễn ra. Đó là tương tác Virtual Things (trao đổi thông tin giữa các virtual things), và tương tác giữa Physical Things và Virtual Things.

Các ứng dụng IoT rất đa dạng, ví dụ, “hệ thống giao thông thông minh”, “Lưới điện thông minh”, “sức khỏe điện tử”, hoặc “nhà thông minh”. Các ứng dụng có thể được dựa trên một nền tảng riêng biệt,cũng có thể được xây dựng dựa trên dịch vụ chung, chẳng hạn như chứng thực, quản lý thiết bị, tính phí, thanh toán… Các “Communication networks” chuyển dữ liệu được thu thập từ devices đến các ứng dụng và device khác, và ngược lại, các network này cũng chuyển các mệnh lệnh thực thi từ ứng dụng đến các device. Vai trò của communication network là truyền tải dữ liệu một cách hiệu quả và tin cậy.



*Hình 1.3: Các loại thiết bị khác nhau và mối quan hệ*

Yêu cầu tối thiểu của các “thiết bị” trong IOT là khả năng giao tiếp . Thiết bị sẽ được phân loại vào các dạng như thiết bị mang thông tin, thiết bị thu thập dữ liệu, thiết bị cảm ứng (sensor), thiết bị thực thi:

Thiết bị mang dữ liệu (Data carrierring device): Một thiết bị mang thông tin được gắn vào một Physical Thing để gián tiếp kết nối các Physical Things với các mạng lưới thông tin liên lạc.

Thiết bị thu thập dữ liệu (Data capturing device): Một device thu thập dữ liệu có thể được đọc và ghi, đ ng thời có khả năng tương tác với Physical Things. Sự tương tác có thể xảy ra một cách gián tiếp thông qua device mang dữ liệu, hoặc trực tiếp thông dữ liệu gắn liền với Physical Things. Trong trường hợp đầu tiên, các device thu thập dữ liệu sẽ đọc thông tin từ một device mang tin và có ghi thông tin từ các network và các device mang dữ liệu.

Thiết bị cảm ứng và thiết bị thực thi (sensing device and actuation device): Một thiết bị cảm nhận và thiết bị thực thi có thể phát hiện hoặc đo lường thông tin liên quan đến môi trường xung quanh và chuyển đổi nó sang tín hiệu dạng số. Nó cũng có thể chuyển đổi các tín hiệu kỹ thuật số từ các mạng thành các hành động(như tắt mở đèn, hù còi báo động …). Nói chung, thiết bị và thiết bị thực thi kết hợp tạo thành một mạng cục bộ giao tiếp với nhau sử dụng công nghệ truyền thông không dây hoặc có dây và các gateway.

Thiết bị chung: Một thiết bị chung đã được tích hợp các network thông qua mạng dây hoặc không dây. Thiết bị chung bao gồm các thiết bị và được dùng cho các domain khác nhau của IOT, chẳng hạn như máy móc, thiết bị điện trong nhà, và Smart phone.

* + 1. **Đặc trưng của các hệ thống IoT**
       1. **Đặc tính cơ bản**

Tính kết nối liên thông (interconnectivity): với IoT, bất cứ điều gì cũng có thể kết nối với nhau thông qua mạng lưới thông tin và cơ sở hạ tầng liên lạc tổng thể.

Những dịch vụ liên quan đến “Things”: hệ thống IoT có khả năng cung cấp các dịch vụ liên quan đến “Things”, chẳng hạn như bảo vệ sự riêng tư và nhất quán giữa Physical Thing và Virtual Thing. Để cung cấp được dịch vụ này, cả công nghệ phần cứng và công nghệ thông tin(phần mềm) sẽ phải thay đổi.

Tính không đ ng nhất: Các thiết bị trong IoT là không đ ng nhất vì nó có phần cứng khác nhau, và network khác nhau. Các thiết bị giữa các network có thể tương tác với nhau nhờ vào sự liên kết của các network.

Thay đổi linh hoạt: Status của các thiết bị tự động thay đổi, ví dụ, ngủ và thức dậy, kết nối hoặc bị ngắt, vị trí thiết bị đã thay đổi,và tốc độ đã thay đổi… Hơn nữa, số lượng thiết bị có thể tự động thay đổi.

Quy mô lớn: Sẽ có một số lượng rất lớn các thiết bị được quản lý và giao tiếp với nhau. Số lượng này lớn hơn nhiều so với số lượng máy tính kết nối Internet hiện nay. Số lượng các thông tin được truyền bởi thiết bị sẽ lớn hơn nhiều so với được truyền bởi con người.

* + - 1. **Yêu cầu ở mức cao đối với một hệ thống IoT**

Một hệ thống IoT phải thoả mãn các yêu cầu sau:

Kết nối dựa trên sự nhận diện: Nghĩa là các “Things” phải có ID riêng biệt. Hệ thống IOT cần hỗ trợ các kết nối giữa các “Things”, và kết nối được thiết lập dựa trên định danh (ID) của Things.

Khả năng cộng tác: hệ thống IoT khả năng tương tác qua lại giữa các mạng và Things.

Khả năng tự quản của mạng: Bao g m tự quản lý, tự cấu hình, tự recovery, tự tối ưu hóa và tự có cơ chế bảo vệ. Điều này cần thiết để mạng có thể thích ứng với các lĩnh vực ứng dụng khác nhau, môi trường truyền thông khác nhau, và nhiều loại thiết bị khác nhau.

Dịch vụ thoả thuận: dịch vụ này để có thể được cung cấp bằng cách thu thập, giao tiếp và xử lý tự động các dữ liệu giữa các “Things” dựa trên các quy tắc (rules) được thiết lập bởi người vận hành hoặc tùy chỉnh bởi các người dùng.

Các khả năng dựa vào vị trí (location-based capabilities): Thông tin liên lạc và các dịch vụ liên quan đến một cái gì đó sẽ phụ thuộc vào thông tin vị trí của Things và người sử dụng. Hệ thống IoT có thể biết và theo dõi vị trí một cách tự động. Các dịch vụ dựa trên vị trí có thể bị hạn chế bởi luật pháp hay quy định, và phải tuân thủ các yêu cầu an ninh.

Bảo mật: Trong IoT, nhiều “Things” được kết nối với nhau. Chình điều này làm tăng mối nguy trong bảo mật, chẳng hạn như bí mật thông tin bị tiết lộ, xác thực sai, hay dữ liệu bị thay đổi hay làm giả.

Bảo vệ tính riêng tư: tất cả các “Things” đều có chủ sở hữu và người sử dụng của nó. Dữ liệu thu thập được từ các “Things” có thể chứa thông tin cá nhân liên quan chủ sở hữu hoặc người sử dụng nó. Các hệ thống IoT cần bảo vệ sự riêng tư trong quá trình truyền dữ liệu, tập hợp, lưu trữ, khai thác và xử lý. Bảo vệ sự riêng tư không nên thiết lập một rào cản đối với xác thực ngu n dữ liệu.

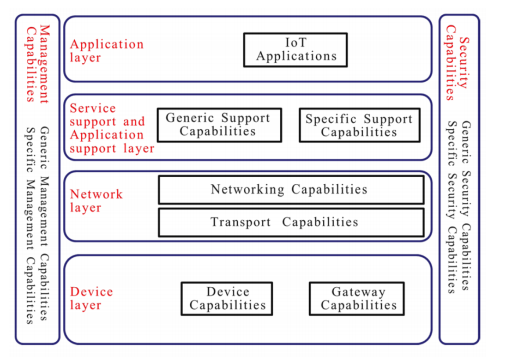
Plug and play: các Things phải được plug-and-play một cách dễ dàng và tiện dụng.

Khả năng quản lý: hệ thống IoT cần phải hỗ trợ tính năng quản lý các “Things” để đảm bảo mạng hoạt động bình thường. Ứng dụng IoT thường làm 14 việc tự động mà không cần sự tham gia của con người, nhưng toàn bộ quá trình hoạt động của họ nên được quản lý bởi các bên liên quan.

* + 1. **Mô hình của một hệ thống IoT**

Bất kỳ một hệ thống IOT nào cũng được xây dựng lên từ sự kết hợp của 4 tầng sau :

* Tầng ứng dụng (Application Layer)
* Tầng hỗ trợ dịch vụ và hỗ trợ ứng dụng (Service support and application support layer)
* Tầng mạng (Network Layer)
* Tầng thiết bị (Device Layer)



*Hình 1.4 Mô hình IoT*

* + - 1. **Application Layer(Tầng ứng dụng)**

Tầng ứng dụng cũng tương tự như trong mô hình OSI 7 lớp, lớp này tương tác trực tiếp với người dùng để cung cấp một chức năng hay một dịch vụ cụ thể của một hệ thống IOT.

* + - 1. **Service support and application support layer(Tầng hỗ trợ dịch vụ và hỗ trợ ứng dụng)**

Nhóm dịch vụ chung: Các dịch vụ hỗ trợ chung, phổ biến mà hầu hết các ứng dụng IOT đều cần, ví dụ như xử lý dữ liệu hoặc lưu trữ dữ liệu.

Nhóm dịch vụ cụ thể, riêng biệt: Những ứng dụng IOT khác nhau sẽ có nhóm dịch phụ hỗ trợ khác nhau và đặc thù. Trong thực tế, nhóm dịch vụ cụ thể riêng biệt là tính toán nhiệt độ độ ẩm ánh sáng để đưa ra quyết định điều chỉnh

* + - 1. **Network Layer(Tầng mạng)**

Lớp Network có 2 chức năng :

Chức năng Networking: cung cấp chức năng điều khiển các kết nối kết nối mạng, chẳng hạn như tiếp cận được nguồn tài nguyên thông tin và chuyển tài nguyên đó đến nơi cần thiết, hay chứng thực, uỷ quyền…

Chức năng Transporting: tập trung vào việc cung cấp kết nối cho việc truyền thông tin của dịch vụ/ứng dụng IOT.

* + - 1. **Device Layer(Tầng thiết bị)**

Lớp Device chính là các phần cứng vật lý trong hệ thống IOT. Device có thể phân thành hai loại như sau:

Thiết bị thông thường: Device này sẽ tương tác trực tiếp với network: Các thiết bị có khả năng thu thập và tải lên thông tin trực tiếp (nghĩa là không phải sử dụng gateway) và có thể trực tiếp nhận thông tin (ví dụ, lệnh) từ các network. Device này cũng có thể tương tác gián tiếp với network: Các thiết bị có thể thu thập và tải network gián tiếp thông qua khả năng gateway. Ngược lại, các thiết bị có thể gián tiếp nhận thông tin (ví dụ, lệnh) từ network. Trong thực tế, các Thiết bị thông thường bao g m các cảm biến, các phần cứng điều khiển motor, đèn,…

Thiết bị Gateway: Gateway là cổng liên lạc giữa device và network. Một Gateway hỗ trợ 2 chức năng sau:

Có nhiều chuẩn giao tiếp: Vì các Things khác nhau có kiểu kết nối khác nhau, nên Gateway phải hỗ trợ đa dạng từ có dây đến không dây, chẳng hạn CAN bus, ZigBee, Bluetooth hoặc Wi-Fi. Tại Network layer, gateway có thể giao tiếp thông qua các công nghệ khác nhau như PSTN, mạng 2G và 3G, LTE, Ethernet hay DSL.

Chức năng chuyển đổi giao thức: Chức năng này cần thiết trong hai tình huống là khi truyền thông ở lớp Device, nhiều device khác nhau sử dụng giao thức khác nhau, ví dụ, ZigBee với Bluetooth, và là khi truyền thông giữa các Device và Network, device dùng giao thức khác, network dùng giao thức khác, ví dụ, device dùng ZigBee còn tầng network thì lại dùng công nghệ 3G.

Trong thực tế, Gateway có thể được build từ các board như Raspberry Pi hay Arduino, hoặc Gateway được sản xuất công nghiệp bởi các tập đoàn lớn như Intel hay Texas Instrument.

* 1. **Tìm hiểu hệ thống nhà thông minh**
     1. **Giới thiệu chung về nhà thông minh**

Nhà thông minh ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *home automation, domotics, smarthome* hoặc *Intellihome*) là kiểu nhà được lắp đặt các thiết bị điện, điện tử có thể được điều khiển hoặc [tự động hoá](https://vi.wikipedia.org/w/index.php?title=T%E1%BB%B1_%C4%91%E1%BB%99ng_ho%C3%A1&action=edit&redlink=1) hoặc bán tự động, thay thế con người trong thực hiện một hoặc một số thao tác quản lý, điều khiển. Hệ thống điện tử này giao tiếp với người dùng thông qua bảng điện tử đặt trong nhà, [ứng dụng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_%E1%BB%A9ng_d%E1%BB%A5ng) trên [điện thoại di động](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_tho%E1%BA%A1i_di_%C4%91%E1%BB%99ng), [máy tính bảng](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh_b%E1%BA%A3ng) hoặc một giao diện web.  Khi chủ nhà về nhà cũng sẽ không còn phải đến từng vị trí công tắc để bật tắt đèn hay tìm điều khiển để mở rèm nữa, các thiết bị sẽ hoạt động hoàn toàn tự động theo mong muốn của gia chủ. Công nghệ thông minh có thể giúp chủ nhà đỡ tốn thời gian hơn vào những vấn đề vặt vãnh hàng ngày, cuộc sống tiện nghi là điều chắc chắn các [**thiết bị nhà thông minh**](http://lumi.vn/san-pham) sẽ mang lại.

Trong căn nhà thông minh, đồ dùng trong nhà từ phòng ngủ, phòng khách đến toilet đều gắn các bộ điều khiển điện tử có thể kết nối với [Internet](https://vi.wikipedia.org/wiki/Internet) và [điện thoại di động](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_tho%E1%BA%A1i_di_%C4%91%E1%BB%99ng), cho phép chủ nhân điều khiển vật dụng từ xa hoặc lập trình cho thiết bị ở nhà hoạt động theo lịch. Thêm vào đó, các đồ gia dụng có thể hiểu được ngôn ngữ của nhau và có khả năng tương tác với nhau.

Các lợi ích của nhà thông minh:

* Tiện nghi hơn
* Tiết kiệm hơn
* An toàn hơn
* Kiểm soát tốt hơn
* Đẳng cấp hơn

Ai trong chúng ta cũng có lúc vội vã rời khỏi nhà mà quên tắt thiết bị điện, máy lạnh hay quên đóng cửa sổ …... lúc đó chúng ta không tránh khỏi cảm giác lo lắng về an toàn cho ngôi nhà mình. Giờ đây, bạn có thể yên tâm vì đã có sự trợ giúp của nhà thông minh. Nhà thông minh sẽ biến ngôi nhà của mình trở nên tiện ích không chỉ phục vụ cho mọi sinh hoạt hàng ngày trong gia đình mà chủ nhà còn có thể dễ dàng kiểm soát ngôi nhà của mình từ xa.

Hệ thống nhà thông minh với các tiện ích vượt trội về an ninh, tiện ích, an toàn, tiết kiệm… giúp thực hiện tự động các kịch bản được lập trình sẵn theo ý thích riêng của mỗi gia đình, đồng thời chủ nhà còn có thể điều khiển và theo dõi từ xa mọi hoạt động trong  ngôi nhà của mình. Điều này đem đến cho gia chủ cảm giác thoái mái và hài lòng khi mọi việc được thực hiện trong tầm tay.

**Hệ thống an ninh tự động**:Ngôi nhà thông minh của bạn sẽ được đảm bảo an ninh chặt chẽ từ ngoài tường rào đến từng căn phòng với các thiết bị cảm biến hiện đại.

Cảm biến hàng rào: dùng để phát hiện và báo động khi có sự đột nhập trái phép tại tường rào, hay leo qua cổng chính vào nhà. Khi đó hệ thống sẽ phát tín hiệu về trung tâm xử lý, lập tức chuông báo động sẽ phát liên tục và hệ thống sẽ gửi cảnh báo qua điện thoại: tin nhắn SMS, Email giúp chủ nhà biết tình hình và kịp thời xử lý tình huống.

Cảm biến cửa: được lắp đặt vị trí cửa ra vào, cửa sổ. Cảm biến này sẽ phát hiện có sự xuất hiện của con người ở vị trí cửa ra vào/cửa sổ và cùng kết hợp với những cảm biến khác thực hiện các kịch bản đã lập trình theo yêu cầu của chủ nhà. Trong trường hợp chủ nhà đi vắng, cảm biến này có nhiệm vụ phát hiện sự đột nhập trái phép vào nhà và phát tín hiệu về bộ xử lý trung tâm, ngay lập tức hệ thống sẽ gửi thông báo cho chủ nhân.

Cảm biến chuyển động: Tự động bật đèn khi có người đi lại trong vùng cảm biến,và tắt đèn (thiết bị) khi có người rời khỏi khu vực giúp tiết kiệm chi phí.

Bất kỳ chuyển động nào khả nghi, bất thường khi chủ nhà đi vắng sẽ được cảm biến này phát hiện và báo động (qua chuông) đồng thời gửi cảnh báo đến điện thoại của chủ nhà qua tin nhắn SMS, Email hoặc Push.

Cảm biến báo khói: Hệ thống cảnh báo khói giúp phát hiện và báo động kịp thời khi có lượng khói lớn bất thường trong nhà, giúp cảnh báo cháy và giảm thiểu tổn thất về người và tài sản.

**Điều khiển nhiệt độ và điều hòa không khí:**Hệ thống kiểm soát nhiệt độ thông minh cho phép tự điều chỉnh nhiệt độ theo đối tượng, theo mùa, theo ngữ cảnh và hỗ trợ điều khiển thông qua điện thoại, máy tính bảng. Là giải pháp không chỉ chăm sóc sức khỏe cho gia đình bạn mà còn sử dụng tiết kiệm năng lượng.

Điều hòa sẽ tự động tắt khi không có người trong nhà, tự động điều chỉnh nhiệt độ về khuya và duy trì   nhiệt độ ở mức hợp lý .

Hệ thống hỗ trợ điều khiển bằng điện thoại, máy tính bảng: hiển thị trạng thái nhiệt độ tại các khu vực trong nhà, cho phép tăng giảm nhiệt độ, thay đổi chế độ cool/fan và nhiều tính năng đi kèm thiết bị bằng giao diện điều khiển trực quan. Giờ đây, dù ở bất cứ vị trí nào bạn cũng có thể chăm sóc cho những người than yêu trong gia đình.

Tự thiết lập chế độ thời tiết tùy thích: bạn có thể được cảm nhận cái se lạnh của mùa đông hay dịu mát của mùa xuân,... cùng với ánh sáng lung linh và giai điệu du dương của âm nhạc thổi bùng những xúc cảm ngay chính trong ngôi nhà của mình bằng tính năng thiết lập chế độ ngữ cảnh.

Tiết kiệm điện năng tiêu thụ: bằng khả năng tự kiểm soát nhiệt độ thông minh và tự động xác định tình trạng bật tắt thiết bị điều hòa đúng với nhu cầu sử dụng của từng thời điểm, từng đối tượng sẽ làm giảm một lượng lớn nguồn điện tiêu hao.

**Điều khiển và giám sát thiết bị :**Tự động bật sáng khi có nhu cầu: với một hệ thống ánh sáng và kiểm soát bóng mờ được triển khai, bạn cũng có thể biến ngôi của bạn trở thành một trải nghiệm thú vị. Giờ đây, bạn không cần phải dùng tay bật tắt từng thiết bị chiếu sáng khi di chuyển mà đơn giản chỉ một nút nhấn để thắp sáng cả lối đi hoặc hệ thống sẽ tự bật sáng khi bạn di chuyển hoặc tắt đi khi không có nhu cầu ánh sáng tại khu vực đó.

Lượng sáng thay đổi theo môi trường: hệ thống sẽ tự điều chỉnh độ sáng theo môi trường (ngày/đêm; theo mùa) theo ngữ cảnh đã được thiết lập. Thiết bị chiếu sáng sẽ bổ sung một lượng sáng phù hợp tại những không gian thiếu sáng hoặc thay đổi màu sắc/trang thái vào những thời điểm đã xác lập.

Thiết lập ngữ cảnh: hệ thống sẽ kiểm soát trang thái các thiết bị chiếu sáng, phối hợp các hiệu ứng màu sắc làm nổi bậc không gian kiến trúc theo chủ đề. Bạn có thể thiết lập chức năng chiếu sáng theo chủ đề: theo mùa, theo trạng thái cảm xúc, theo chức năng,…

Kích hoạt hoặc dừng hệ thống khi không có nhu cầu: khi rời khỏi nhà, bạn chỉ cần nhấn "Vắng nhà" hệ thống sẽ cung cấp cho bạn một thời lượng đủ để di chuyển ra khỏi nhà và sẽ tắt toàn bộ thiết bị chiếu sáng ngay sau đó. Hay là chức năng "Vào nhà" sẽ kích hoạt hệ thống tự vận hành theo thói quen hàng ngày của bạn.

Tiết kiệm năng lượng: tiết kiệm năng lượng bằng việc kiểm soát cân bằng và bổ sung lượng sáng cần thiết. Bên cạnh đó, Việc thay đổi màu sắc ánh sáng trong nhà cũng có tác dụng ngăn cản hoặc hấp thụ lượng nhiệt từ bên ngoài, giúp không gian bên trong trở nên mát mẻ và sinh động.

**Hệ thống âm thanh và hình ảnh đa vùng:**Hệ thống âm thanh đa vùng là hệ thống điều khiển và quản lý âm thanh chung cho cả tòa nhà

Ngôi nhà được phân thành nhiều vùng thưởng thức âm thanh độc lập với nhau

Nguồn âm thanh đầu vào như Server Âm Thanh, Ipod, DVD, CD Player, Tuner, Catssette ... được chia sẻ chung cho cả tòa nhà

Tại mỗi vùng nghe, người thưởng thức chỉ việc chọn lựa nguồn nhạc, bản nhạc ... từ màn hình điều khiển cảm ứng hoặc điều khiển từ xa.

**Giải pháp tiết kiệm của nhà thông minh**:Nhà thông giúp bạn tiết kiệm chi phí sinh hoạt trong gia đình vì mỗi thiết bị điện đều có mức tiêu thụ điện năng thấp và đồng thời chúng quản lý cực kỳ hiệu quả những thiết bị điện khác trong căn nhà. Chẳng hạn như các thiết bị điện trong nhà  sẽ tự động tắt nếu phát hiện không có người trong nhà, đèn tự động tắt nếu trong phòng đủ ánh sáng, máy lạnh sẽ tắt khi bạn mở cửa sổ hay tự điều chỉnh nhiệt độ về khuya….nhờ vậy tránh lãng phí điện năng đáng kể.

## **CHƯƠNG II. XÂY DỰNG MÔ HÌNH MÔ PHỎNG HỆ THỐNG NHÀ THÔNG MINH ĐƠN GIẢN**

### 2.1 Tìm hiểu về Arduino

#### **2.1.1 Lịch sử**

Arduino được khởi động vào năm 2005 như là một dự án dành cho sinh viên trại Interaction Design Institute Ivrea (Viện thiết kế tương tác Ivrea) tại Ivrea, Italy. Vào thời điểm đó các sinh viên sử dụng một "BASIC Stamp" (con tem Cơ Bản) có giá khoảng $100, xem như giá dành cho sinh viên. Massimo Banzi, một trong những người sáng lập, giảng dạy tại Ivrea. Cái tên "Arduino" đến từ một quán bar tại Ivrea, nơi một vài nhà sáng lập của dự án này thường xuyên gặp mặt. Bản thân quán bar này có được lấy tên là Arduino, Bá tước của Ivrea, và là vua của Italy từ năm 1002 đến 1014.

Lý thuyết phần cứng được đóng góp bởi một sinh viên người Colombia tên là Hernando Barragan. Sau khi nền tảng Wiring hoàn thành, các nhà nghiên cứu đã làm việc với nhau để giúp nó nhẹ hơn, rẻ hơn, và khả dụng đối với cộng đồng mã nguồn mở. Trường này cuối cùng bị đóng cửa, vì vậy các nhà nghiên cứu, một trong số đó là David Cuarlielles, đã phổ biến ý tưởng này.

Giá hiện tại của board mạch này dao động xung quanh $30 và được làm giả đến mức chỉ còn $9. Một số mạch bắt chước đơn giản Arduino Mini Pro được sản xuất từ Trung Quốc có giá rẻ hơn $4 bao gồm cả phí bưu điện.

* + 1. **Giới thiệu**

Arduino là một board mạch vi xử lý, 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.

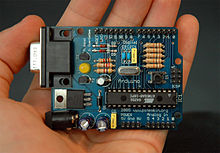
Được giới thiệu vào năm 2005, Những nhà thiết kế của Arduino cố gắng mang đến một phương thức dễ dàng, không tốn kém cho những người yêu thích, sinh viên và giới chuyên nghiệp để tạo ra những thiết bị có khả năng tương tác với môi trường thông qua các cảm biến và các cơ cấu chấp hành. Những ví dụ phổ 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 Aduino bằng ngôn ngữ C hoặc C++.

Giá của các board Arduino dao động xung quanh $20 - $27, nếu được "làm giả" thì giá có thể giảm xuống thấp hơn $9. Các board Arduino có thể được đặt hàng ở dạng được lắp sẵn hoặc dưới dạng các kit tự-làm-lấy. 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 ngàn mạch Arduino chính thức đã được sản xuất thương mại, và vào năm 2013 có khoảng 700 ngàn mạch chính thức đã được đưa tới tay người dùng.

* + 1. **Phần cứng**

[](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_top-1.jpg)

*Hình 2.1 Một mạch Arduino Uno chính thức với các mô tả về các cổng I/O*

[](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino316.jpg)

*Hình 2.2 Một board Arduino đời đầu gồm một cổng giao tiếp RS-232 (góc phía trên-bên trái) và một chip vi xử lý Atmel ATmega8 (màu đen, nằm góc phải-phía dưới); 14 chân I/O số nằm ở phía trên và 6 chân analog đầu vào ở phía đáy*.

Một mạch Arduino bao gồm một vi điều khiển AVR với nhiều linh kiện bổ sung giúp dễ dàng lập trình và có thể mở rộng với các mạch khác. Một khía cạnh quan trọng của Arduino là các kết nối tiêu chuẩn của nó, cho phép người dùng kết nối với CPU của board với các module thêm vào có thể dễ dàng chuyển đổi, được gọi là *shield*. Vài shield truyền thông với board Arduino trực tiếp thông qua các chân khách nhau, nhưng nhiều shield được định địa chỉ thông qua serial bus I²C-nhiều shield có thể được xếp chồng và sử dụng dưới dạng song song. Arduino chính thức thường sử dụng các dòng chip megaAVR, đặc biệt là ATmega8, ATmega168, ATmega328, ATmega1280, và ATmega2560. Một vài các bộ vi xử lý khác cũng được sử dụng bởi các mạch Aquino tương thích. Hầu hết các mạch gồm một bộ điều chỉnh tuyến tính 5V và một thạch anh dao động 16 MHz (hoặc bộ cộng hưởng ceramic trong một vài biến thể), mặc dù một vài thiết kế như LilyPad chạy tại 8 MHz và bỏ qua bộ điều chỉnh điện áp onboard do hạn chế về kích cỡ thiết bị. Một vi điều khiển Arduino cũng có thể được lập trình sẵn với một boot loader cho phép đơn giản là upload chương trình vào bộ nhớ flash on-chip, so với các thiết bị khác thường phải cần một bộ nạp bên ngoài. Điều này giúp cho việc sử dụng Arduino được trực tiếp hơn bằng cách cho phép sử dụng 1 máy tính gốc như là một bộ nạp chương trình.

Theo nguyên tắc, khi sử dụng ngăn xếp phần mềm Arduino, tất cả các board được lập trình thông qua một kết nối RS-232, nhưng cách thức thực hiện lại tùy thuộc vào đời phần cứng. Các board Serial Arduino có chứa một mạch chuyển đổi giữa RS232 sang TTL. Các board Arduino hiện tại được lập trình thông qua cổng USB, thực hiện thông qua chip chuyển đổi USB-to-serial như là FTDI FT232. Vài biến thể, như Arduino Mini và Boarduino không chính thức, sử dụng một board adapter hoặc cáp nối USB-to-serial có thể tháo rời được, Bluetooth hoặc các phương thức khác. (Khi sử dụng một công cụ lập trình vi điều khiển truyền thống thay vì ArduinoIDE, công cụ lập trình AVR ISP tiêu chuẩn sẽ được sử dụng.)

Board Arduino sẽ đưa ra hầu hết các chân I/O của vi điều khiển để sử dụng cho những mạch ngoài. Diecimila, Duemilanove, và bây giờ là Uno đưa ra 14 chân I/O kỹ thuật số, 6 trong số đó có thể tạo xung PWM (điều chế độ rộng xung) và 6 chân input analog, có thể được sử dụng như là 6 chân I/O số. Những chân này được thiết kế nằm phía trên mặt board, thông qua các header cái 0.10-inch (2.5 mm). Nhiều shield ứng dụng plug-in cũng được thương mại hóa. Các board Arduino Nano, và Arduino-compatible Bare Bones Board và Boarduino có thể cung cấp các chân header đực ở mặt trên của board dùng để cắm vào các breadboard.

Có nhiều biến thể như Arduino-compatible và Arduino-derived. Một vài trong số đó có chức năng tương đương với Arduino và có thể sử dụng để thay thế qua lại. Nhiều mở rộng cho Arduino được thực thiện bằng cách thêm vào các driver đầu ra, thường sử dụng trong các trường học để đơn giản hóa các cấu trúc của các 'con rệp' và các robot nhỏ. Những board khác thường tương đương về điện nhưng có thay đổi về hình dạng-đôi khi còn duy trì độ tương thích với các shield, đôi khi không. Vài biến thể sử dụng bộ vi xử lý hoàn toàn khác biệt, với các mức độ tương thích khác nhau.

* + 1. **Các Broad chính thức**

Phần cứng Arduino gốc được sản xuất bởi công ty Italy tên là Smart Projects. Một vài board dẫn xuất từ Arduino cũng được thiết kế bởi công ty của Mỹ tên là SparkFun Electronics. Sáu phiên bản phần cứng của Arduino cũng đã được sản xuất thương mại tính đến thời điểm hiện tại.

|  |  |  |  |
| --- | --- | --- | --- |
| Tên | Ảnh | Tên | Ảnh |
| Arduino Diecimila in Stoicheia | [https://upload.wikimedia.org/wikipedia/commons/thumb/9/96/Arduino_Diecimila_6.jpg/120px-Arduino_Diecimila_6.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Diecimila_6.jpg) | Arduino MEGA 2560 R3 (mặt trước) | [https://upload.wikimedia.org/wikipedia/commons/thumb/8/80/Arduino_MEGA_2560_R3%2C_front_side.jpg/120px-Arduino_MEGA_2560_R3%2C_front_side.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_MEGA_2560_R3,_front_side.jpg) |
| Arduino Duemilanove (rev 2009b) | [https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Arduino_Duemilanove_2009b.jpg/120px-Arduino_Duemilanove_2009b.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Duemilanove_2009b.jpg) | Arduino MEGA 2560 R3 (mặt sau) | [https://upload.wikimedia.org/wikipedia/commons/thumb/0/0d/Arduino_MEGA_2560_R3%2C_back_side.jpg/120px-Arduino_MEGA_2560_R3%2C_back_side.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_MEGA_2560_R3,_back_side.jpg) |
| Arduino UNO  (AVR 8-bit ATmega328P) | [https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Arduino_UNO_unpacked.jpg/120px-Arduino_UNO_unpacked.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_UNO_unpacked.jpg) | Arduino Nano | [https://upload.wikimedia.org/wikipedia/commons/thumb/8/8d/Arduino_Nano.jpg/55px-Arduino_Nano.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Nano.jpg) |
| Arduino Leonardo | [https://upload.wikimedia.org/wikipedia/commons/thumb/3/38/Arduino_Leonardo.jpg/120px-Arduino_Leonardo.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Leonardo.jpg) | Arduino Due (nền tảng ARM) | [https://upload.wikimedia.org/wikipedia/commons/thumb/1/16/Arduino_Due.jpg/90px-Arduino_Due.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Due.jpg) |
| Arduino Mega | [https://upload.wikimedia.org/wikipedia/commons/thumb/0/01/Arduino_Mega.jpg/120px-Arduino_Mega.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Mega.jpg) | LilyPad Arduino (rev 2007) | [https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/LilyPad_Arduino_Main_Board.JPG/120px-LilyPad_Arduino_Main_Board.JPG](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:LilyPad_Arduino_Main_Board.JPG) |

*Bảng 2.1 Các board Arduino mẫu*

* + 1. **Shield(Bo mạch mở rộng)**

Các board Arduino và Arduino-compatible sử dụng các *shield*— các board mạch in mở rộng được dùng bằng cách cắm vào các chân header của Arduino. Các shield có thể là module điều khiển cho động cơ, GPS, ethernet, LCD, hoặc cũng có thể là breadboard. Một số lượng lớn các shield cũng có thể được chế tạo bởi DIY (những người thích tự làm lấy các ứng dụng cho riêng họ).

|  |  |
| --- | --- |
| *[https://upload.wikimedia.org/wikipedia/commons/thumb/d/df/Arduino_Protoboard_Shields.jpg/120px-Arduino_Protoboard_Shields.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Arduino_Protoboard_Shields.jpg) Nhiều shield có thể được xếp chồng lên nhau. Trong ví dụ này shield ở trên cùng có chứa một breadboard chưa hàn* | *[https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Adafruit_Motor_Shield_-_ARSH-02-MS_01.jpg/120px-Adafruit_Motor_Shield_-_ARSH-02-MS_01.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Adafruit_Motor_Shield_-_ARSH-02-MS_01.jpg)Shield Adafruit Motor với các đầu domino dùng để kết nối với động cơ* |
| *[https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/Wingshield_on_Arduino_-_ARSH-05-WI.jpg/120px-Wingshield_on_Arduino_-_ARSH-05-WI.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Wingshield_on_Arduino_-_ARSH-05-WI.jpg) Shield này sử dụng các đầu domino bắt vít dùng để đấu các đầu dây vào* | *[https://upload.wikimedia.org/wikipedia/commons/thumb/e/ee/ARSH-09-DL_03.jpg/120px-ARSH-09-DL_03.jpg](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:ARSH-09-DL_03.jpg)Shield Adafruit Datalogging với một khe chứa thẻ nhớ SD và chip clock Real-Time* |

*Hình 2.3 Các shield Arduino điển hình*

* + 1. **Môi trường phát triển**

|  |  |
| --- | --- |
| Arduino Software IDE | |
| [Phát triển bởi](https://vi.wikipedia.org/wiki/Nh%C3%A0_ph%C3%A1t_tri%E1%BB%83n_ph%E1%BA%A7n_m%E1%BB%81m) | Arduino Software |
| [Repository](https://vi.wikipedia.org/w/index.php?title=Repository_(version_control)&action=edit&redlink=1) | [github.com/arduino/Arduino](https://github.com/arduino/Arduino) |
| Được viết bằng | [Java](https://vi.wikipedia.org/w/index.php?title=Java_(programming_language)&action=edit&redlink=1), [C](https://vi.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&redlink=1) và [C++](https://vi.wikipedia.org/wiki/C%2B%2B) |
| [Hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) | [Cross-platform](https://vi.wikipedia.org/wiki/Cross-platform) |
| [Thể loại](https://vi.wikipedia.org/w/index.php?title=Danh_s%C3%A1ch_th%E1%BB%83_lo%E1%BA%A1i_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) | [Integrated development environment](https://vi.wikipedia.org/w/index.php?title=Integrated_development_environment&action=edit&redlink=1) |
| [Giấy phép](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_ph%E1%BA%A7n_m%E1%BB%81m) | [LGPL](https://vi.wikipedia.org/wiki/GNU_Lesser_General_Public_License) or [GPL](https://vi.wikipedia.org/wiki/GNU_General_Public_License) license |
| Trang mạng | [arduino.cc](http://arduino.cc/en/Main/Software) |

*Bảng 2.2 Môi trường phát triển Arduino*

Môi trường phát triển tích hợp ([IDE](https://vi.wikipedia.org/wiki/IDE)) của Arduino là một ứng dụng [cross-platform](https://vi.wikipedia.org/wiki/Cross-platform) (nền tảng) được viết bằng [Java](https://vi.wikipedia.org/w/index.php?title=Java_(programming_language)&action=edit&redlink=1), và từ IDE này sẽ được sử dụng cho [Ngôn ngữ lập trình xử lý](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh_x%E1%BB%AD_l%C3%BD&action=edit&redlink=1) (Processing programming language) và project [Wiring](https://vi.wikipedia.org/w/index.php?title=Wiring_(development_platform)&action=edit&redlink=1). Nó được thiết kế để dành cho những người mới tập làm quen với lĩnh vực phát triển phần mềm. Nó bao gồm một chương trình code editor với các chức năng như đánh dấu cú pháp, tự động [brace matching](https://vi.wikipedia.org/w/index.php?title=Brace_matching&action=edit&redlink=1), và tự động canh lề, cũng như compile(biên dịch) và upload chương trình lên board chỉ với 1 cú nhấp chuột. Một chương trình hoặc code viết cho Arduino được gọi là một *sketch*.

Các chương trình Arduino được viết bằng [C](https://vi.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&redlink=1) hoặc [C++](https://vi.wikipedia.org/wiki/C%2B%2B). Arduino IDE đi kèm với một [thư viện phần mềm](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) được gọi là "[Wiring](https://vi.wikipedia.org/w/index.php?title=Wiring_(development_platform)&action=edit&redlink=1)", từ project Wiring gốc, có thể giúp các thao tác input/output được dễ dàng hơn. Người dùng chỉ cần định nghĩa 2 hàm để tạo ra một chương trình [vòng thực thi](https://vi.wikipedia.org/w/index.php?title=V%C3%B2ng_th%E1%BB%B1c_thi&action=edit&redlink=1) (cyclic executive) có thể chạy được:

* setup(): hàm này chạy mỗi khi khởi động một chương trình, dùng để thiết lập các cài đặt
* loop(): hàm này được gọi lặp lại cho đến khi tắt nguồn board mạch

Một chương trình điển hình cho một bộ vi điều khiển đơn giản chỉ là làm cho một bóng đèn Led sáng/tắt. Trong môi trường Arduino, ta sẽ phải viết một chương trình giống như sau

#define LED\_PIN 13

void setup () {

pinMode (LED\_PIN, OUTPUT); *// Đặt chân 13 làm đầu ra digital*

}

void loop () {

digitalWrite (LED\_PIN, HIGH); *// Bật LED on*

delay (1000); *// chờ trong 1 giây (1000 mili giây)*

digitalWrite (LED\_PIN, LOW); *// Tắt LED off*

delay (1000); *// chờ trong 1s*

}

*Hình 2.4 Một chương trình Arduino đơn giản*

Một đặc điểm của hầu hết các board Arduino là chúng có một đèn LED và điện trở nối giữa chân 13 với đất; một đặc điểm thuận tiện cho nhiều ứng dụng đơn giản. Đoạn code ở trên không thể đọc được bởi một [compiler](https://vi.wikipedia.org/wiki/Compiler) C++ chuẩn như là một chương trình đúng, vì vậy khi ta click vào nút "Upload to I/O board" trong IDE này, một bản copy của đoạn code này sẽ được ghi vào một file tạm với một extra include header ở phía trên cùng và một hàm main () đơn giản nằm ở phía đáy, để làm cho thàn một chương trình C++ khả dụng.

Arduino IDE này sử dụng [GNU toolchain](https://vi.wikipedia.org/w/index.php?title=GNU_toolchain&action=edit&redlink=1) và AVR Libc để biên dịch chương trình, và sử dụng avrdude để upload chương trình lên board.Vì nền tảng của Arduino là các vi điều khiển của Atmel, cho nên môi trường phát triển của Atmel, AVR Studio hoặc các phiên bản Atmel Studio mới hơn, cũng có thể được sử dụng để làm phần mềm phát triển cho Arduino.

**2.2 KIT ESP 8266 WeMos D1 R2**

ESP8266 là dòng chip tích hợp Wi-Fi 2.4Ghz có thể lập trình được, rẻ tiền được sản xuất bởi một công ty bán dẫn Trung Quốc: Espressif Systems.

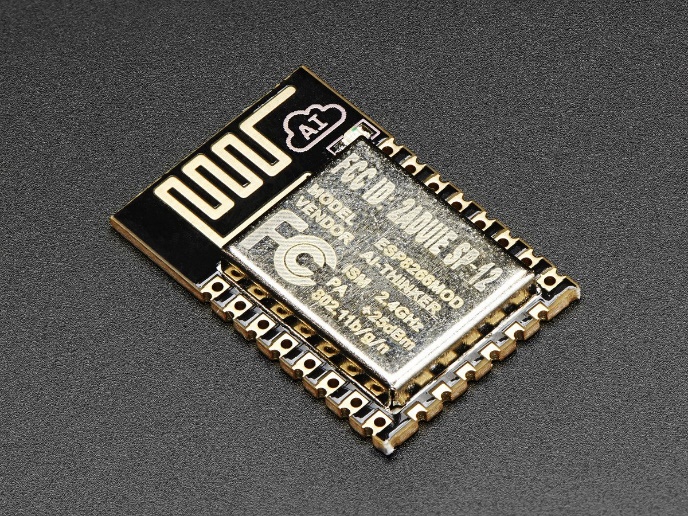
Được phát hành đầu tiên vào tháng 8 năm 2014, đóng gói đưa ra thị trường dạng Mô dun ESP-01, được sản xuất bởi bên thứ 3: AI-Thinker. Có khả năng kết nối Internet qua mạng Wi-Fi một cách nhanh chóng và sử dụng rất ít linh kiện đi kèm. Với giá cả có thể nói là rất rẻ so với tính năng và khả năng ESP8266 có thể làm được.

ESP8266 có một cộng đồng các nhà phát triển trên thế giới rất lớn, cung cấp nhiều Module lập trình mã nguồn mở giúp nhiều người có thể tiếp cận và xây dựng ứng dụng rất nhanh.

Hiện nay tất cả các dòng chip ESP8266 trên thị trường đều mang nhãn ESP8266EX, là phiên bản nâng cấp của ESP8266

Thông số phần cứng

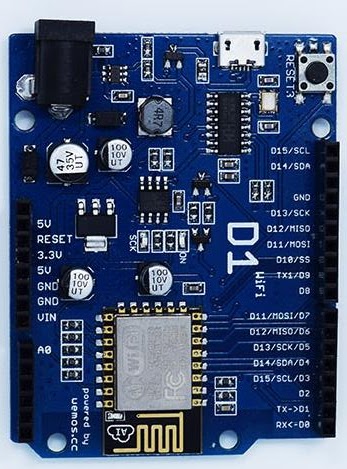
* 32-bit RISC CPU : Tensilica Xtensa LX106 chạy ở xung nhịp 80 MHz
* Hổ trợ Flash ngoài từ 512KiB đến 4MiB
* 64KBytes RAM thực thi lệnh
* 96KBytes RAM dữ liệu
* 64KBytes boot ROM
* Chuẩn wifi EEE 802.11 b/g/n, Wi-Fi 2.4 GHz
  + Tích hợp TR switch, balun, LNA, khuếch đại công suất và matching network
  + Hổ trợ WEP, WPA/WPA2, Open network
* Tích hợp giao thức TCP/IP
* Hổ trợ nhiều loại anten
* 16 chân GPIO
* Hổ trợ SDIO 2.0, UART, SPI, I²C, PWM,I²S với DMA
* 1 ADC 10-bit
* Dải nhiệt độ hoạt động rộng : -40C ~ 125C



*Hình 2.5 Một module tích hợp phổ biến (Module ESP12E)*

#### **2.2.1 Làm quen với KIT ESP8266 Wemos D1 R2**

WEMOS D1 R2 là kit phát triển phiên bản mới nhất từ WeMos, kit được thiết kế với hình dáng tương tự Arduino Uno nhưng trung tâm lại là module wifi Soc ESP8266EX được build lại firmware để có thể chạy với chương trình Arduino.  Kit thích hợp và dễ dàng thực hiện các ứng dụng thu thập dữ liệu và điều khiển qua Wifi.

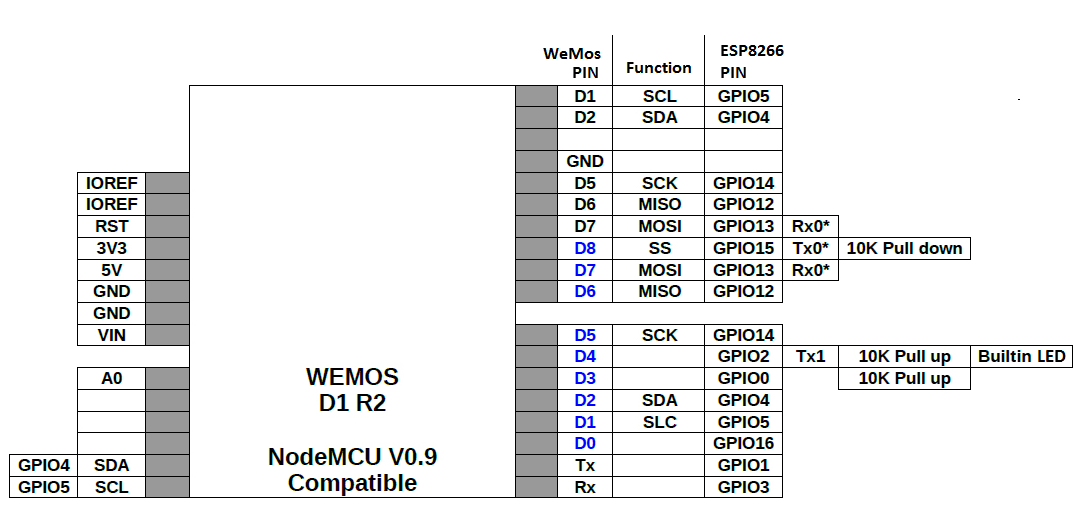
[](http://k3.arduino.vn/img/2017/09/28/0/3877_812450-1506569643-0-d1-2.jpg)

*Hình 2.6 Mặt trên của KIT ESP8266 Wemos D1 R2*

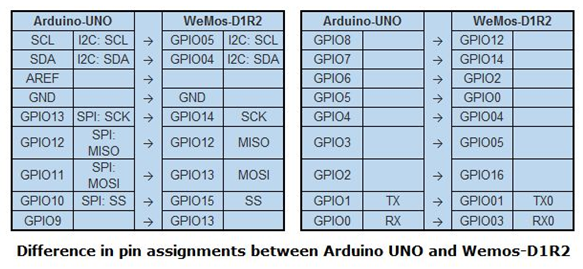
|  |  |
| --- | --- |
| **Vi điều khiển** | **ESP8266EX** |
| Điện áp hoạt động | 3V3 |
| I/O Digital Pin | 11 |
| Analog Pin | 1 (Max input=3V2) |
| Xung clock | 80MHz/160MHz |
| Flash | 4Mb |
| Khối lượng | 25g |
| Kích thước | 68.6mmX53.4mm |

*Bảng 2.2 Thông số kĩ thuật KIT ESP8266 Wemos D1 R2*

Sơ đồ pin Digital



*Hình 2.7 Sơ đồ Pin trên KIT*



*Hình 2.8 Sự khác biệt khi bố trí Pin trên UNO và Wemos-D1 R2*

|  |  |  |
| --- | --- | --- |
| **Wemos D1 R2 PINs** | **ESP8266 PINs** | **Grove Arduino Base Shield PINs** |
| D0 | GPIO16 | D2 |
| D1 | GPIO5 (SLC) | D3 |
| D2 | GPIO4 (SDA) | D4 |
| D3 | GPIO0 | D5 |
| D4 | GPIO2 (Builtin LED) | D6 |
| D5 | GPIO14 (SCK) | D7 |
| D6 | GPIO12 | D8 |
| D7 |  |  |
| D8 |  |  |

*Bảng 2.3 Mapping pin giữa Wemos D1 R2,ESP8266 và Shield*

Lí do cần có bảng này là để dễ dàng trong việc viết code. Bởi khi viết code nếu muốn xuất tín hiệu HIGH cho pin số 3 trên kit thì ta ko thể viết "digitalWrite(3,HIGH);" mà phải viết là "digitalWrite(0,HIGH);" (Theo trên bảng thì chân D3 trên kit là chân 0 của ESP). Hay nói cách khác là ta phải điều khiển chân trên ESP8266. Và khi ta code xuất chân D3 mức HIGH thì chân D15 cũng được xuất HIGH, lí do là vì: các chân từ D11=>D15 là các chân "giả" để kit trông giống arduino hơn (ESP chỉ có 11 pin digital), vì vậy các pin D11=>D15 được nối lần lượt với D7=>D3.

* + 1. **Các ứng dụng của KIT WeMos D1 R2**

Làm Robot. ESP8266 có khả năng đọc các thiết bị cảm biến, điều khiển động cơ,… nên nó thường được dùng để làm bộ xử lý trung tâm của rất nhiều loại robot.

Game tương tác: ESP8266 có thể được sử dụng để tương tác với Joystick, màn hình,… khi chơi các game như Tetrix, phá gach, Mario…

Máy bay không người lái. Thiết bị giám sát,

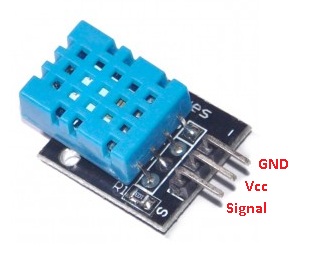
Điều khiển đèn tín hiệu giao thông, làm hiệu ứng đèn Led nhấp nháy trên các biển quảng cáo…

Điều khiển các thiết bị cảm biến ánh sáng, âm thanh.Làm máy in 3D,

WebServer,HTTP Client gọi API,các hệ thống thông minh tự động

* 1. **Một số cảm biến sử dụng**

#### **2.3.1 Cảm biến nhiệt độ-độ ẩm DHT11**

[](http://k3.arduino.vn/img/2014/05/27/0/481_81220-1401185335-0-dht-11-truoc-300x250.jpg)

*Hình 2.9 Cảm biến nhiệt độ độ ẩm DHT11*

Cảm biến DHT11 đã được tích hợp trong một mạch duy nhất, bạn chỉ việc nối dây nguồn (Vcc, GND) và dây tín hiệu (Signal) vào mạch Arduino là xong.

**Thông số kĩ thuật**

* Điện áp hoạt động: 3-5.5V DC
* Ngưỡng độ ẩm: 20 - 90%
* Sai số độ ẩm: ± 5%
* Ngưỡng nhiệt độ: 0 - 55oC
* Sai số nhiệt độ: ± 2oC

**Kết nối cảm biến DHT11 với mạch Arduino**

|  |  |
| --- | --- |
| **DHT11** | **ESP8266** |
| GND | GND |
| Vcc | 5V |
| Signal | D2 |

* 1. **Tìm hiểu về kiến trúc Webservice-RESTful, Webservice với Spring framework.**

#### **2.4.1 Tìm hiểu về Webserver**

Hệ thống sử dụng web server là cây cầu nối, thực hiện việc trao đổi giao tiếp giữa người dùng và các thiết bị điện tử trong nhà.

“Web server” có thể là phần cứng hoặc phần mềm, hoặc cả hai.

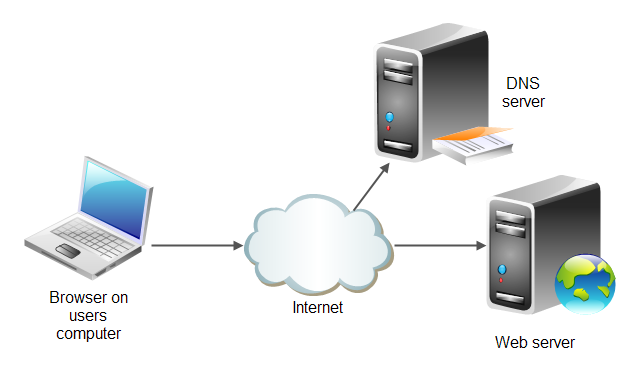
* **Ở khía cạnh phần cứng**

Một web server là một máy tính lưu trữ các file thành phần của một website (ví dụ: các tài liệu HTML, các file ảnh, CSS và các file JavaScript) và có thể phân phát chúng tới thiết bị của người dùng cuối (end-user). Nó kết nối tới mạng Internet và có thể truy cập tới thông qua một tên miền giống như luongit.pro.

* Ở khía cạnh phần mềm

Một web server bao gồm một số phần để điều khiển cách người sử dụng web truy cập tới các file được lưu trữ trên một HTTP server(máy chủ HTTP). Một HTTP server là một phần mềm hiểu được các URL (các địa chỉ web) và HTTP (giao thức trình duyệt của bạn sử dụng để xem các trang web).

Ở mức cơ bản nhất, bất cứ khi nào một trình duyệt cần một file được lưu trữ trên một web server, trình duyệt request (yêu cầu) file đó thông qua HTTP. Khi một request tới đúng web server (phần cứng), HTTP server (phần mềm) gửi tài liệu được yêu cầu trở lại, cũng thông qua HTTP.



*Hình 2.10 Mô hình hoạt động của webserver*

Để có một website, ta cần một static hoặc dynamic web server.

Một **static web server**, hoặc stack, bao gồm một máy tính (hardware) với một HTTP server (phần mềm). Chúng ta gọi nó là "static" bởi vì server (máy chủ) gửi các file nó lưu trữ "nguyên vẹn" (as-is) tới trình duyệt của bạn.

Một **dynamic web server** bao gồm một static web server cộng với các phần mềm mở rộng, phổ biến nhất là một application server (máy chủ ứng dụng) và một database. Chúng ta gọi nó là "dynamic" bởi vì application server cập nhật các file được lưu trữ trước khi gửi chúng tới tình duyệt của bạn thông qua HTTP server.

Ví dụ, để tạo ra các trang web mà chúng ta nhìn thấy trong trình duyệt, application server cung cấp một HTML template với những nội dung lấy từ một database. Các website giống như MDN hay Wikipedia có hàng nghìn trang web, nhưng chúng không phải là các tài liệu HTML thực sự, mà chỉ là các HTML template và một database khổng lồ. Thiết lập này làm cho nó dễ dàng và nhanh hơn để bảo dưỡng và phân phối nội dụng.

Tìm hiểu sâu hơn để truy cập một trang web, trình duyệt của chúng ta gửi một request tới web server, nó sẽ tìm kiếm file được yêu cầu, lưu trữ trên ổ đĩa của nó. Khi tìm thấy file, server đọc nó, xử lý nếu cần, và gửi nó tới trình duyệt. Hãy xem xét các bước này chi tiết hơn. Đầu tiên, một web server phải lưu trữ các file của website, đó là các tài liệu HTML và các tài nguyên liên quan đến nó, bao gồm các ảnh, file CSS, file JavaScript, fonts và videos.

Về mặt kỹ thuật, bạn có thể lưu trữ tất cả các file trên máy tính của mình, nhưng có nhiều lợi ích hơn khi lưu trữ chúng trên một máy chủ riêng biệt như:

* Luôn luôn sẵn sàng (up and running)
* Luôn luôn kết nối tới mạng Internet
* Có một địa chỉ IP cố định
* Được bảo dưỡng bởi nhà cung cấp (third-party provider)

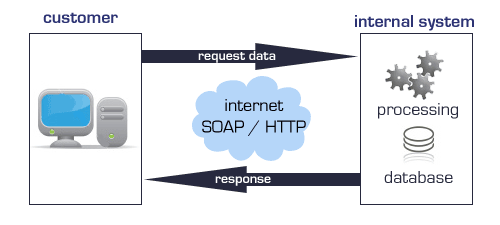
Vì tất cả những lý do này, tìm một nhà cung cấp máy chủ (hosting provider) tốt là một phần quan trọng trong việc xây dựng website của bạn. Tìm hiễu kỹ dịch vụ mà các công ty cung cấp và chọn một cái phù hợp với nhu cầu và túi tiền của người dùng.

* + 1. **Webservice**

### Web service Là những thành phần ứng dụng dùng để chuyển đổi một ứng dụng thông thường sang một ứng dụng web. Đồng thời nó cũng xuất bản các chức năng của mình để mọi người dùng internet trên thế giới đều có thể sử dụng thông qua nền tảng web. Web Serviece truyền thông bằng cách sử dụng các giao thức mở, tài nguyên phần mềm có thể xác định bằng địa chỉ URL, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu, các ứng dụng độ lập và tự mô tả chính nó. Nó bao gồm các modun độc lập cho hoạt động của khác hàng và doanh nghiệp và bản thân nó được thực thi trên server. Nền tảng cơ bản của webservice là XML+ HTTP. Bất cứ một ứng dụng nào cũng đều có thể có một thành phần webservice. Webservice có thể được tạo ra bằng bất kỳ một ngôn ngữ lập trình nào.

### Đặc điểm:

* Cho phép client và server tương tác ngay cả trong môi trường khác nhau. (Ví dụ server chạy linux, client chạy windows).
* Phần lớn được xây dựng dựa trên mã nguồn mở và phát triển các chuẩn đã được công nhận. (Ví dụ XML).
* Nó có thể triển khai bởi 1 phần mềm ứng dụng phía server (Ví dụ : PHP, Oracle Application server, Microsoft .NET)



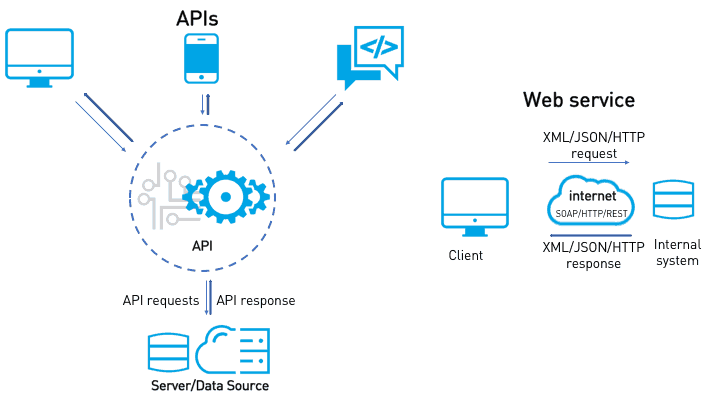
*Hình 2.11 Kiến trúc cơ bản của webservice*

Cách thức hoạt động của webservice: Nền tảng cơ bản của WS là XML + HTTP.

XML cung cấp một ngôn ngữ mà có thể được sử dụng giữa ngôn ngữ lập trình và các nền tảng khác. Đồng thời, nó còn có thể được dùng để mô tả những thông điệp và chức năng phức tạp. Do web service là sự kết hợp của nhiều thành phần khác nhau, do đó web serices sử dụng các tính năng và đặc trưng ủa các thành phần này để giao tiếp với nhau. Vì vậy XML là một công cụ chính yếu để giải quyết vấn đề này. Web service tận dụng khả năng giải quyết vấn đề của các ứng dụng lớn trên các hệ điều hành khác nhau cho chúng giao tiếp với nhau. Yêu cầu này được đáp ứng với lập trình Java, một ngôn ngữ việt một lần sử dụng mọi nơi là một lựa chọn thích hợp cho phát triển web service.

Giao thức HTTP là giao thức được sử dụng nhiều nhất trong các giao thúc trên internet.Nền tảng của WS bao gồm các thành phần:SOAP (Sinple Object Access Protocol): Giao thức truy cập đối tượng đơn giản.UDDI (Universal Description, Discovered and Integrated).WSDL (Web Services Desription Language): Ngôn ngữ mô tả WS.

* + 1. **RESTFul webservice**



*Hình 2.13 Khác nhau giữa Webservice và RESTFul*

RESTful web service là các web service được viết dựa trên kiến trúc REST. RESTful web service nhẹ, có khả năng dễ mở rộng và bảo trì.

REST là viết tắt của REpresentational State Transfer (dịch nôn na là chuyển trạng thái đại diện) là một kiểu kiến trúc lập trình, nó định nghĩa các quy tắc để thiết kết các web service chú trọng vào tài nguyên hệ thống. Trong kiến trúc REST mọi thứ đều được coi là tài nguyên, chúng có thể là: tệp văn bản, ảnh, trang html, video, hoặc dữ liệu động… REST server cung cấp quyền truy cập vào các tài nguyên, REST client truy cập và thay đổi các tài nguyên đó. Ở đây các tài nguyên được định danh dựa vào URI, REST sử dụng một vài đại diện để biểu diễn các tài nguyên như văn bản, JSON, XML.

### Sử dụng các phương thức HTTP một cách rõ ràng

Điểm chú ý đầu tiên khi cài đặt RESTful web service là sử dụng một cách rõ ràng các phương thức HTTP theo cách mà chúng được định nghĩa bởi RFC 2616. Ví dụ phương thức GET được định nghĩa với mục đích để client sử dụng vào việc lấy dữ liệu từ server hoặc thực hiện một truy vấn để server tìm và trả về một tập các tài nguyên phù hợp. REST yêu cầu các nhà phát triển sử dụng một các rõ ràng các phương thức HTTP và nhất quán với cách mà chúng được định nghĩa. Quy tắc này của REST giúp thiết lập ánh xạ một một giữa các hành động tạo, đọc, cập nhật và xóa với các phương thức HTTP. Theo đó sẽ có:Để tạo một tài nguyên trên server ta dùng phương thức POST.Để lấy(đọc) tài nguyên trên server ta dùng phương thức GET.Để update tài nguyên trên server ta dùng phương thức PUT.Để xóa tài nguyển trên server ta dùng phương thức DELETE.

Việc sử dụng các phương thức HTTP một cách không rõ ràng ví dụ như sử dụng GET để tạo hoặc cập nhật tài nguyên, dữ liệu hệ thống sẽ gây ra vấn đề về mặt ngữ nghĩa. Web server được thiết kế để phản hồi lại các yêu cầu GET bằng việc lấy và trả về chúng bằng một dạng nào đó, chứ không phải để tạo hoặc cập nhật bản ghi trong cơ sở dữ liệu. Tương tự với các phương thức HTTP khác, theo REST chúng nên được sử dụng đúng với các chức năng nêu trên. Ngoài mặt ngữ nghĩa, còn có một vấn đề nữa có thể dẫn đến việc thay đổi dữ liệu phía server một cách không chủ tâm. Chú ý rằng các nguyên tắc ở trên là không bắt buộc, thực tế chúng ta có thể sử dụng phương thức **GET** để yêu cầu lấy dữ liệu, chèn, sửa hoặc xóa dữ liệu trên Server. Tuy nhiên **REST** đưa ra các nguyên tắc ở trên mục đích đưa mọi thứ trở lên rõ ràng và dễ hiểu.

### Phi trạng thái

### Theo REST trạng thái hoặc được giữ trên client hoặc được chuyển thành trạng thái của tài nguyên. Nói một cách khác một server sẽ không bao giờ giữ trạng thái thông tin trao đổi với bất kỳ client nào nó giao tiếp, mỗi request lên server thì client phải đóng gói thông tin đầy đủ để thằng server hiểu được. Điều này giúp hệ thống của bạn dễ phát triển,bảo trì, mở rộng vì không cần tốn công CRUD trạng thái của client. Ngoài ra còn một nguyên nhân quan trọng hơn đó là nó tách biệt client khỏi sự thay đổi của server.

### Cấu trúc thư mục như URI

URI trong RESTful web service phải tự mô tả, hoặc tham chiếu được cái mà nó trỏ tới và các tài nguyên liên quan. Ngoài ra URI cũng phải đơn giản, có thể đoán biết được, và dễ hiểu. Để tạo ra URI với yêu cầu trên thì ta nên định nghĩa URI có câu trúc giống thư mục. Loại URI này có phân cấp, có gốc là một đường dẫn đơn, các nhánh từ gốc là các đường dẫn phụ dẫn đến các các vùng service chính.

Ví dụ về URI trong RESTful web service:

[*http://samples.openweathermap.org/data/2.5/weather?lat=35&lon=139&appid=b6907d289e10d714a6e88b30761fae22*](http://samples.openweathermap.org/data/2.5/weather?lat=35&lon=139&appid=b6907d289e10d714a6e88b30761fae22)

Với URI có cấu trúc như thư mục cho phép nhà phát triển dễ dàng trong việc cài đặt service của mình hướng vào một loại tài nguyên cụ thể nào đó.

### Chuyển đổi XML, JSON hoặc cả hai

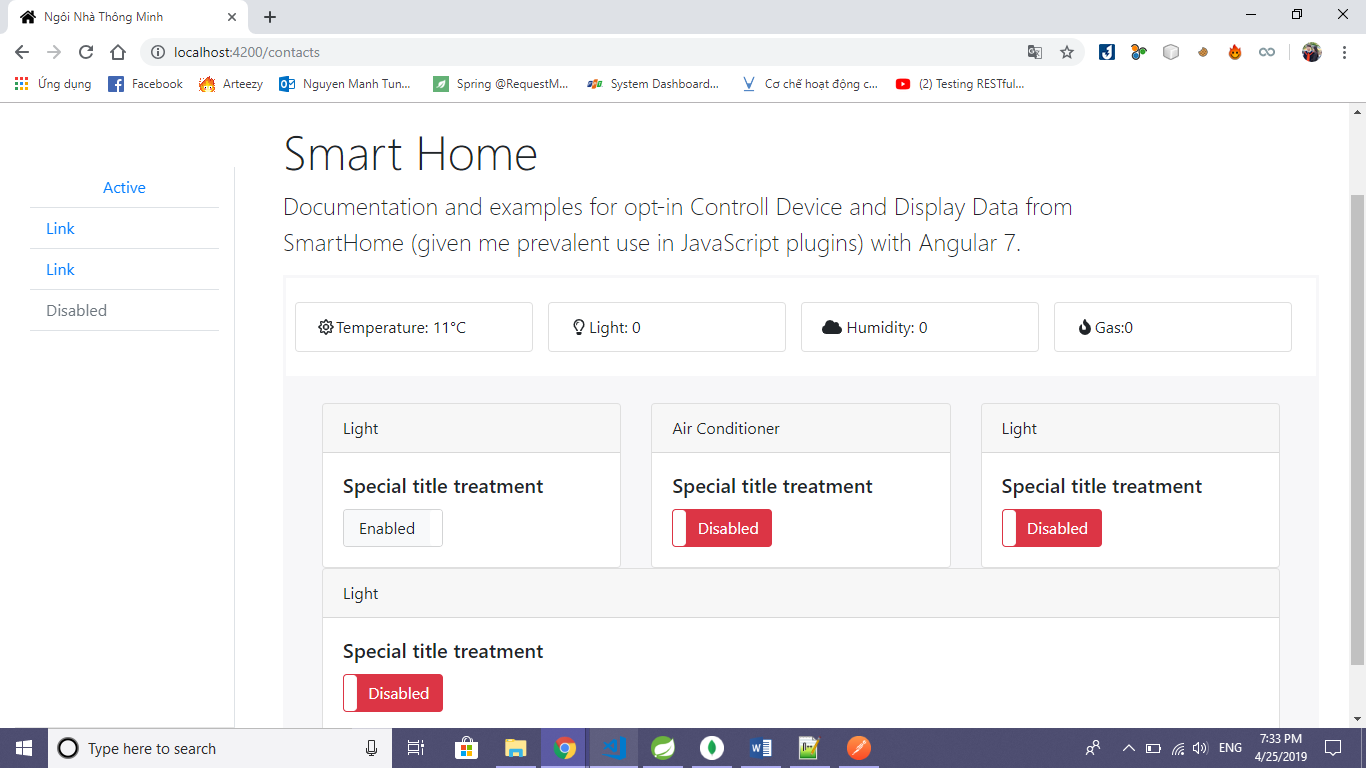
Điều cuối cùng trong tập các ràng buộc khi thiết kế RESTful web service phải làm là định dạng dữ liệu mà ứng dụng và service trao đổi trong phụ tải request/response hoặc trong HTTP body. Cung cấp nhiều đại diện biểu diễn cho tài nguyên cho các request khác nhau. Cụ thể ở đây ta có thể sử dụng các một vài kiểu MIME thông dụng sau:JSON,XML.

Điều này cho phép các service sử dụng bởi các client viết bởi các ngôn ngữ khác nhau, chạy trên nhiều nền tảng và thiết bị khác nhau. Sử dụng các kiểu MIME cho phép client chọn dạng dữ liệu phù hợp với nó.

* + 1. **Website**

Website (tiếng anh: website), còn gọi là trang web (có thể nhầm lẫn với "web page") hoặc trang mạng, là một tập hợp trang web, thường chỉ nằm trong một tên miền hoặc tên miền phụ trên World Wide Web của Internet. Một trang web là tập tin HTML hoặc XHTML có thể truy nhập dùng giao thức HTTP. Trang mạng có thể được xây dựng từ các tệp tin HTML (trang mạng tĩnh) hoặc vận hành bằng các CMS chạy trên máy chủ (trang mạng động).

Trang mạng có thể được xây dựng bằng nhiều ngôn ngữ lập trình khác nhau (PHP, ASP.NET, Java, Ruby,…).



### *Hình 2.14 Giao diện website cơ bản.*

Website được giao tiếp và hiển thị cho người dùng truy cập bằng các phần mềm được gọi là trình duyệt website. Một sô trình duyệt website nổi tiếng có thể kể đến như Internet Explorer được cài đặt mặc định vào mỗi máy tính cài hệ điều hành windows và được thay thế bởi Edge trên Windows 10 hay Chrome được phát triển bởi Google và Firefox được phát triển bởi Mozilla.

Website tĩnh là website mà người quản trị (những người không phải là lập trình viên) không thể tùy ý thay đổi nội dung và hình ảnh mà phải cần kiến thức về HTML cơ bản. Website tĩnh được viết hoàn toàn dựa trên nền tảng HTML CSS và thêm các hiệu ứng từ Javascript nếu muốn

Website động là website được viết kèm theo một bộ công cụ quản trị để tùy biến nội dung dành cho webmaster (người quản trị) có thể dễ dàng thay đổi nội dung, hình ảnh. Website động được thiết kế bởi các lập trình viên để làm sao cho phép website có thể thay đổi được nội dung thường xuyên. Một số công nghệ, ngôn ngữ để xây dựng website động bao gồm PHP, ASP.NET, Java,..

* + 1. **Webservice với Spring framework.**

### Giới thiệu.

Spring Web Services (Spring-WS) là một sản phẩm của cộng đồng Spring tập trung vào việc tạo các dịch vụ Web dựa trên tài liệu. Spring Web Services nhằm mục đích tạo điều kiện phát triển dịch vụ SOAP đầu tiên theo hợp đồng, cho phép tạo ra các dịch vụ web linh hoạt bằng một trong nhiều cách để thao tác tải trọng XML. Sản phẩm này dựa trên chính Spring, có nghĩa là bạn có thể sử dụng các Spring concept như là inversion of control hay dependency injection như một phần không thể thiếu trong dịch vụ Web.

### Spring Rest webservie.

REST đã nhanh chóng trở thành tiêu chuẩn thực tế để xây dựng các dịch vụ web trên web vì chúng dễ xây dựng và dễ tiêu thụ.Có một cuộc thảo luận lớn hơn nhiều về cách REST phù hợp với thế giới microservice, nhưng - đối với hướng dẫn này - chúng ta hãy nhìn vào việc xây dựng các dịch vụ RESTful.

Tại sao lại là REST? REST tuân theo các quy chuẩn, bao gồm kiến ​​trúc, lợi ích và mọi thứ khác cung cấp một loạt các tính năng:

* Hành động phù hợp ( GET, POST, PUT, DELETE, ...)
* Bộ nhớ đệm
* Chuyển hướng và chuyển tiếp
* Bảo mật (mã hóa và xác thực)

Đây là tất cả các yếu tố quan trọng để xây dựng các dịch vụ kiên cường. Nhưng đó không phải là tất cả. Web được xây dựng từ rất nhiều thông số kỹ thuật nhỏ, do đó nó có thể phát triển dễ dàng mà không bị sa lầy vào "cuộc chiến tiêu chuẩn".

Các nhà phát triển có thể rút ra các bộ công cụ của bên thứ 3 thực hiện các thông số kỹ thuật đa dạng này và ngay lập tức có cả công nghệ máy khách và máy chủ trong tầm tay.Vì vậy, dựa trên HTTP, các API REST cung cấp các phương tiện để xây dựng các API linh hoạt có thể:

* Hỗ trợ tương thích ngược
* API có thể phát triển
* Dịch vụ mở rộng
* Dịch vụ bảo đảm
* Một loạt các dịch vụ phi trạng thái

Điều quan trọng để nhận ra là REST, tuy nhiên phổ biến, không phải là một tiêu chuẩn, cho mỗi gia nhập , nhưng một cách tiếp cận, một phong cách, một tập hợp của những hạn chế về kiến trúc của bạn có thể giúp bạn xây dựng hệ thống web quy mô.

Spring framework cung cấp một loạt các API từ gói spring-webmvc giúp việc xây dựng ứng dụng webservice trở nên nhanh chóng hơn, các nhà phát triển không cần quan tâm quá nhiều đến các configuration mà tập trung vào xử lý bussiness của hệ thống.

Dưới đây là ví dụ một controller ném ra các rest api.

package payroll;

import java.util.List;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

class EmployeeController {

private final EmployeeRepository repository;

EmployeeController(EmployeeRepository repository) {

this.repository = repository;

}

// Aggregate root

@GetMapping("/employees")

List<Employee> all() {

return repository.findAll();

}

@PostMapping("/employees")

Employee newEmployee(@RequestBody Employee newEmployee) {

return repository.save(newEmployee);

}

// Single item

@GetMapping("/employees/{id}")

Employee one(@PathVariable Long id) {

return repository.findById(id)

.orElseThrow(() -> new EmployeeNotFoundException(id));

}

@PutMapping("/employees/{id}")

Employee replaceEmployee(@RequestBody Employee newEmployee, @PathVariable Long id) {

return repository.findById(id)

.map(employee -> {

employee.setName(newEmployee.getName());

employee.setRole(newEmployee.getRole());

return repository.save(employee);

})

.orElseGet(() -> {

newEmployee.setId(id);

return repository.save(newEmployee);

});

}

@DeleteMapping("/employees/{id}")

void deleteEmployee(@PathVariable Long id) {

repository.deleteById(id);

}

}

### Annotation @RestController chỉ ra rằng dữ liệu trả về của mỗi phương thức sẽ được render thẳng ra HTML dưới định dạng Json hoặc XML.

### Đối tượng EmployeeRepository được tiêm thông qua hàm khởi tạo cùng với cơ chế dependency injection, giúp cho các nhà phát triển không cần quá lo lắng về việc khi nào khởi tạo các đối tượng đồng nghĩa với việc giảm áp lực cho heap.

### Spring cung cấp các annotation @GetMapping, @PostMaping, @RequestMapping tương ứng với các phương thức HTTP GET,POST,… Ngoài ra còn cung cấp các cơ chế xử lý exception rất mạnh mẽ.

### 2.5 Xây dựng hệ thống nhà thông minh

Hệ thống SmartHome gồm các bộ điều khiển không dây, thiết bị mạng, cảm biến và phần mềm điều khiển… với tổng cộng khoảng 10 đầu sản phẩm.

Trong hệ thống của mô hình nhà thông minh, đồ dùng trong nhà gồm đèn, tivi, rèm cửa sổ, điều hòa điều được gắn các bộ điều khiển điện tử có thể kết nối với Internet, cho phép chủ nhân điều khiển vật dụng từ xa hoặc lập trình cho thiết bị ở nhà hoạt động theo lịch. Thêm vào đó, các đồ gia dụng có thể hiểu được ngôn ngữ của nhau và có khả năng tương tác với nhau.

#### **2.5.1 Mô tả hệ thống**

Nhà thông minh với phân hệ điều khiển từ xa được thực hiện với những thành phần cụ thể sau:

Hệ thống nhà gồm 4 thành phần:

* Thiết bị điện, điện tử
* Bộ kit Arduino
* Hệ thống web server, service (API)
* Hệ thống website

Miêu tả cụ thể :

Các thiết bị điện tử trong nhà bao gồm: Rèm cửa, đèn, điều hòa, tivi làm việc một cách tự động hoặc bán tự động.

Bộ kit Arduino là một board mạch vi xử lý, 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. Dùng bộ kit để thực hiện việc điều khiển các thiết bị điện tử một cách tự động hóa hoặc bán tự đông (điều khiển bởi con người).

Hệ thống web service (API), xây dựng để thực hiện việc giao tiếp của con người với các thiết bị trong nhà qua nhiều nền tảng khác nhau (ứng dụng điện thoại, phần mềm máy tính, website).

Xây dựng hệ thống website để thực hiện việc giao tiếp dễ dàng, giữa người dùng và các thiết bị trong nhà. Cùng với việc dễ dàng kiểm soát những thiết bị và thông số trong gia đình.

### Mang lại giá trị nghiệp vụ

### Tăng khả năng xử lý: các thiết bị trong nhà được xử lý một cách tự động, có thể xử lý đồng thời và cho kết quả nhanh chóng, chính xác. Thông tin về  xung quanh ngôi nhà được cập nhật, thông báo cho người dùng một cách nhanh chóng.

Đáp ứng yêu cầu nghiệp vụ một cách tin cậy, chính xác, an toàn, bí mật

### Mang lại giá trị sử dụng

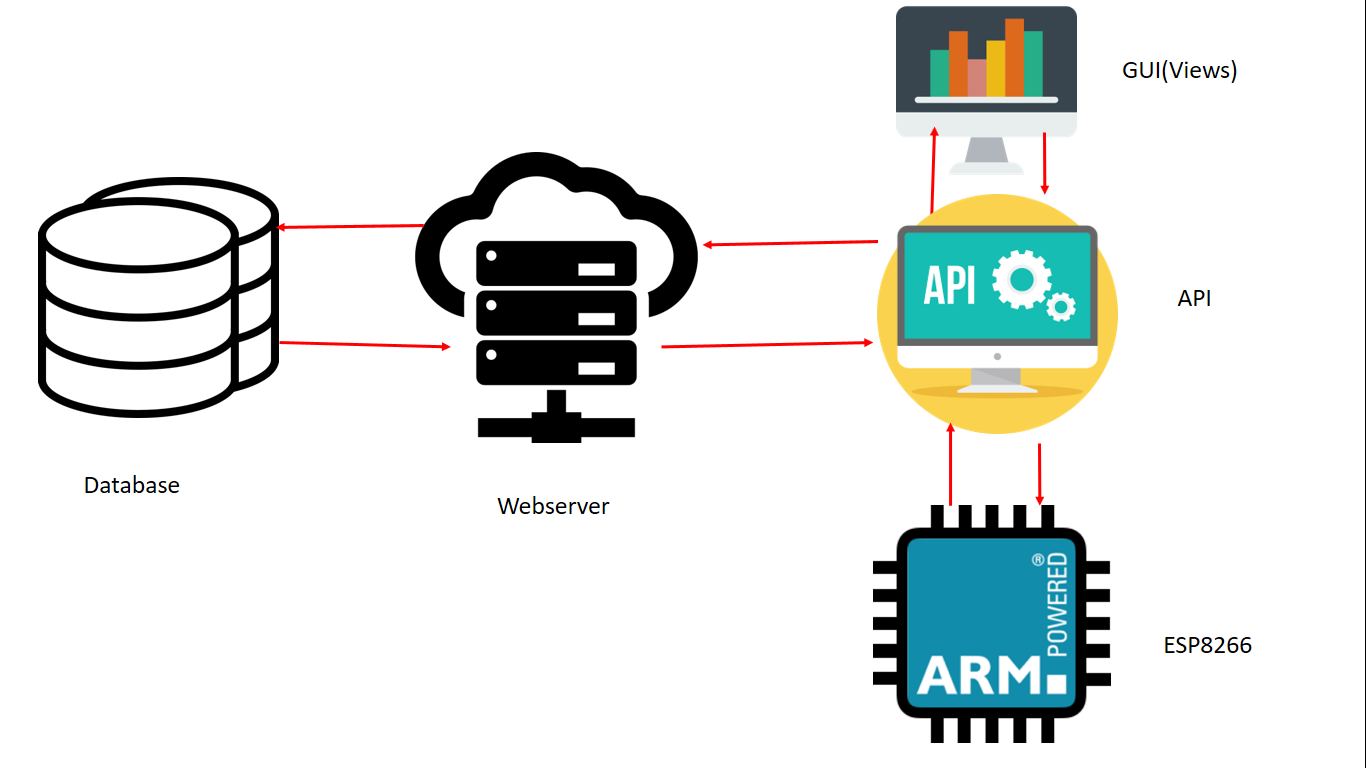
Mọi thành viên trong gia đình dễ dàng kiểm soát hoạt động của ngôi nhà, điều khiển các thiết bị khắp mọi nơi chỉ cần có Internet.

Tiện lợi phục vụ cho những gia đình bận rộn với công việc.Dễ dàng quản lý việc tự động hóa của các thiết bị trong nhà.

* Các yêu cầu của hệ thống

Trong quá trình sử dụng thì hệ thống cần đáp ứng các yêu cầu sau :Tốc độ truy xuất xử lý một cách nhanh chóng nhất.Giao diện dễ dùng, thân thiện, trực quan với người sử dụng Ngôn ngữ sử dụng phải phổ biến, phù hợp với người dùng.

#### **2.5.2 Hoạt động của hệ thống**



*Hình 2.15 Hoạt động của hệ thống*

Các thiết bị điện tử sẽ được điều khiển bởi bộ Kit, thực hiện các công việc tự động hoặc bán tự động theo lệnh của bộ kit.

Bộ kit là nơi xử lý dữ liệu và đưa ra quyết định cho các thiết bị điện tử được sử dụng. Thực hiện ra lệnh tắt, bật các thiết bị điện tử bằng việc đóng ngắt mạch điện của các thiết bị. Nơi thực hiện thu thập thông tin từ các thiết bị cảm biến (sensor). Nó cũng là nơi thực hiện việc cập nhật dữ liệu lên hệ thống webservice, cùng việc lấy thông tin các câu lệnh yêu cầu từ phía server gửi về.

Server là nơi lưu trữ dữ liệu của hệ thống, hình thành và tạo ra bộ RESTfull API để phục vụ việc sử dụng, xây dựng ứng dụng đa nền tảng để chủ nhà điều khiển hệ thống, điều khiển ngôi nhà của mình một cách dễ dàng nhất.

Views là các máy trạm (client) sự dụng bộ API từ server để người dùng dễ dàng tương tác, điều khiển, kiểm soát các thiết bị trong nhà một cách dễ dàng nhất.

#### **2.5.3 Vai trò của ESP8266** **Wemos D1 R2**

Bộ kit Arduino đóng vài trò như một máy trạm (client) của hệ thống nhà thông minh. Nó làm việc thông qua các dữ liệu được gửi và nhận từ dữ liệu với service. Bộ kit sẽ thực hiện xử lý các dữ liệu để đưa ra quyết định cuối cùng cho các thiết bị điện tử.

Các thư viện sử dụng trong khi lập trình cho Wemos D1 R2**:**

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <ArduinoJson.h>

#include "DHTesp.h"

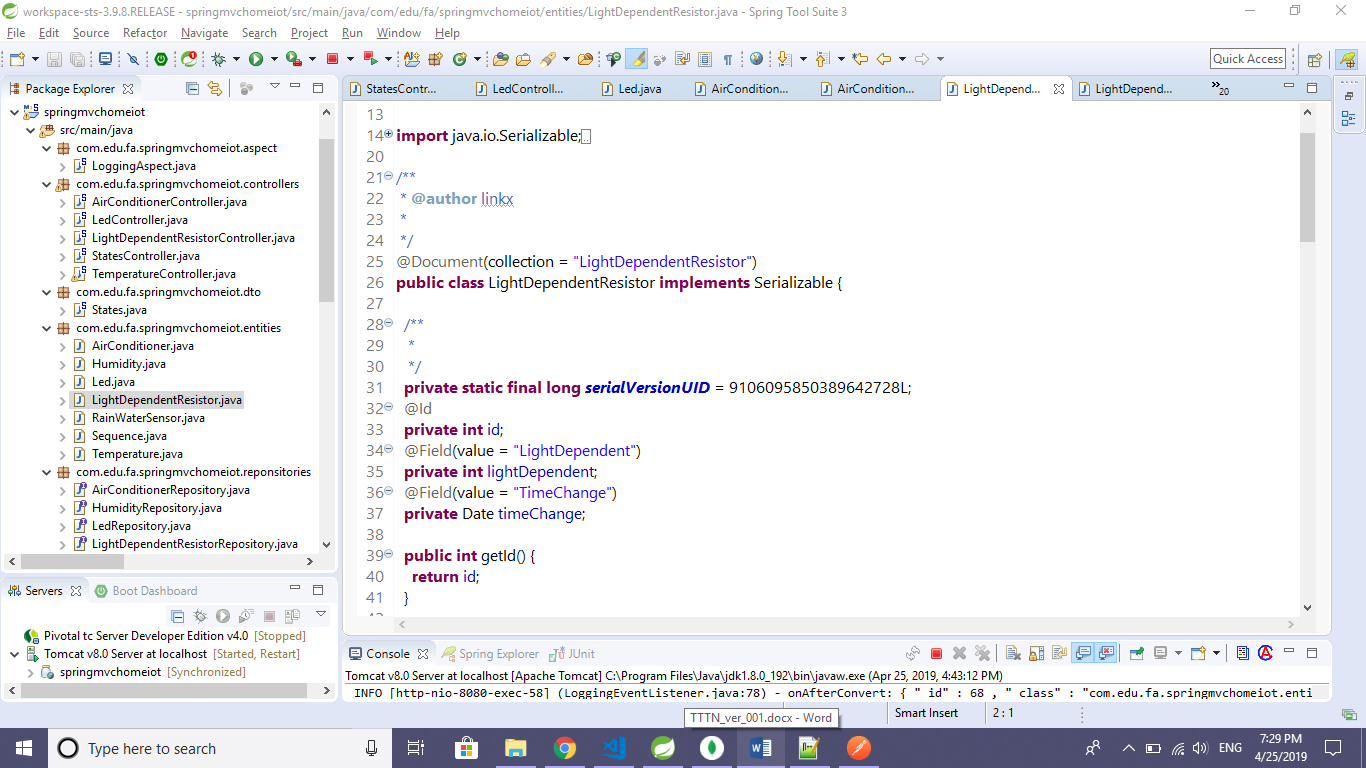
#### **2.5.4 Hệ thống Webserver và Webservice**

Web server và web service tomcat được xây dựng bằng ngôn ngữ Java với công nghệ Sping MVC. Công việc của webserver trong hệ thống sẽ là nơi lưu trữ, chuyển giao dữ liệu của các thông số trong nhà để cung cấp cho các máy trạm (client) để phục vụ nhu cầu. Hệ thống sẽ hoạt động 24h không cần sự tác động của con người để phục vụ một cách nhanh chóng nhất.

[Apache Tomcat®](http://tomcat.apache.org/) là một phần mềm mã nguồn mở thực hiện cài đặt công nghệ Java Servlet, JavaServer Pages, Java Expression Languate và Java WebSocket.Tomcat là một ứng dụng máy chủ gọn nhẹ, thường dùng để deploy các ứng dụng Java Web. Nó được phát triển bởi Apache và hoàn toàn miễn phí.

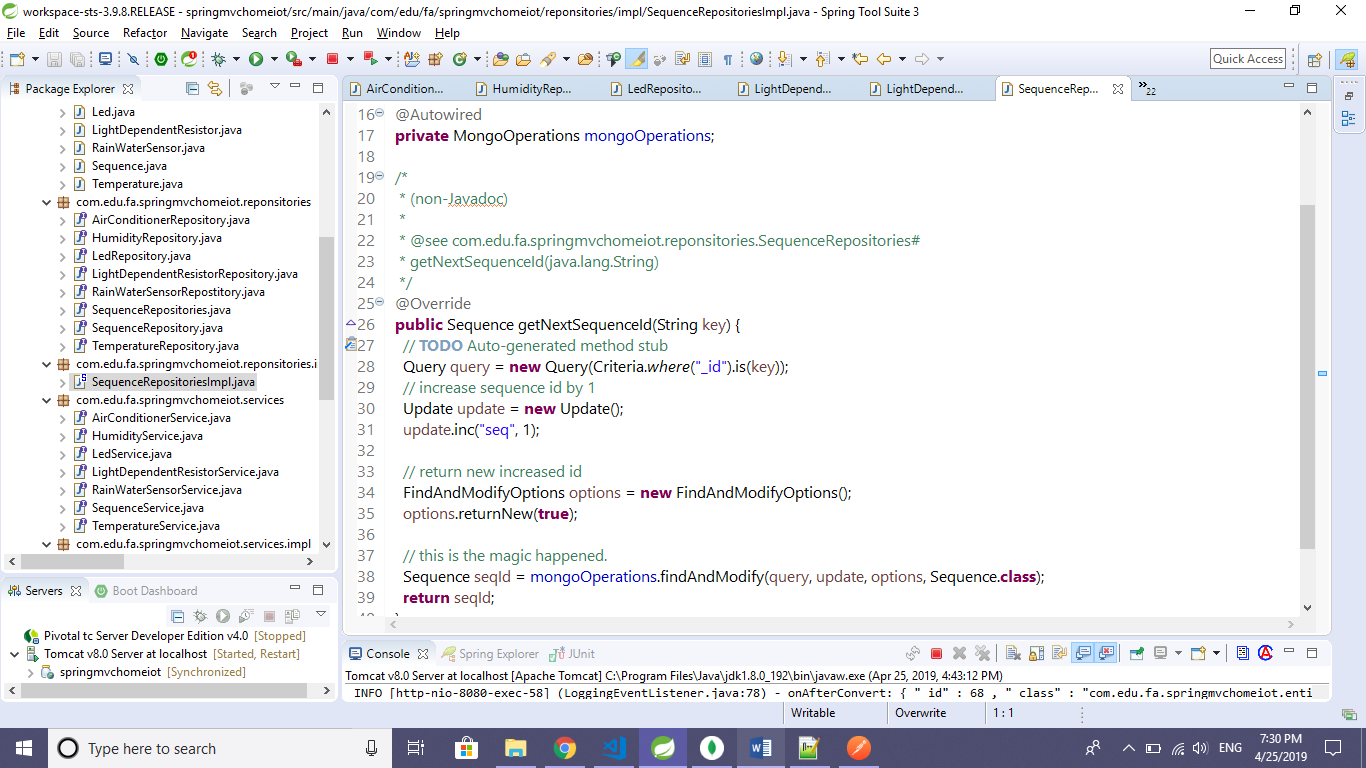
Web service sẽ tương tác trực tiếp với Database và thực hiện việc gửi và nhận dữ liệu từ phía máy trạm (client), thông qua giao thức http protocol.

* Ví dụ một entity model



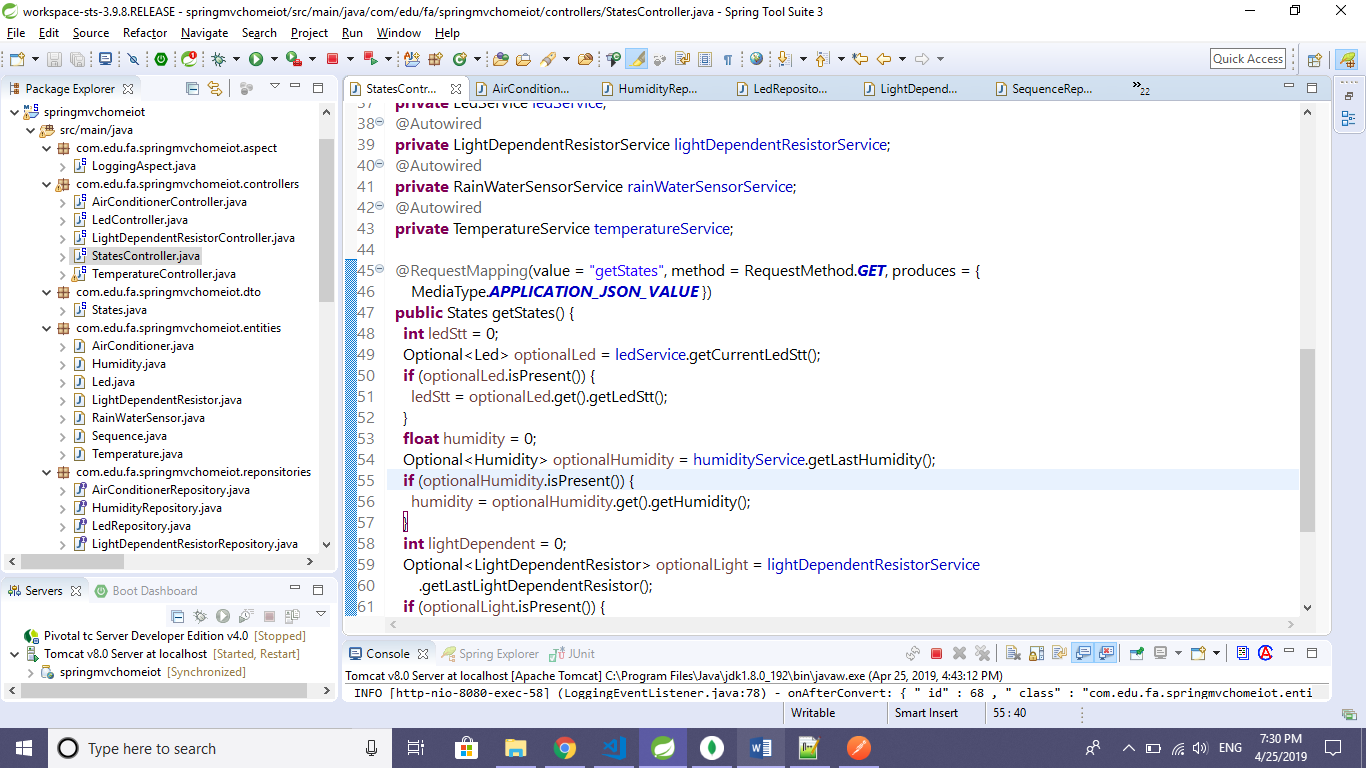
*Hình 2.16 Entity model*

* Một số class ở tầng Persistence



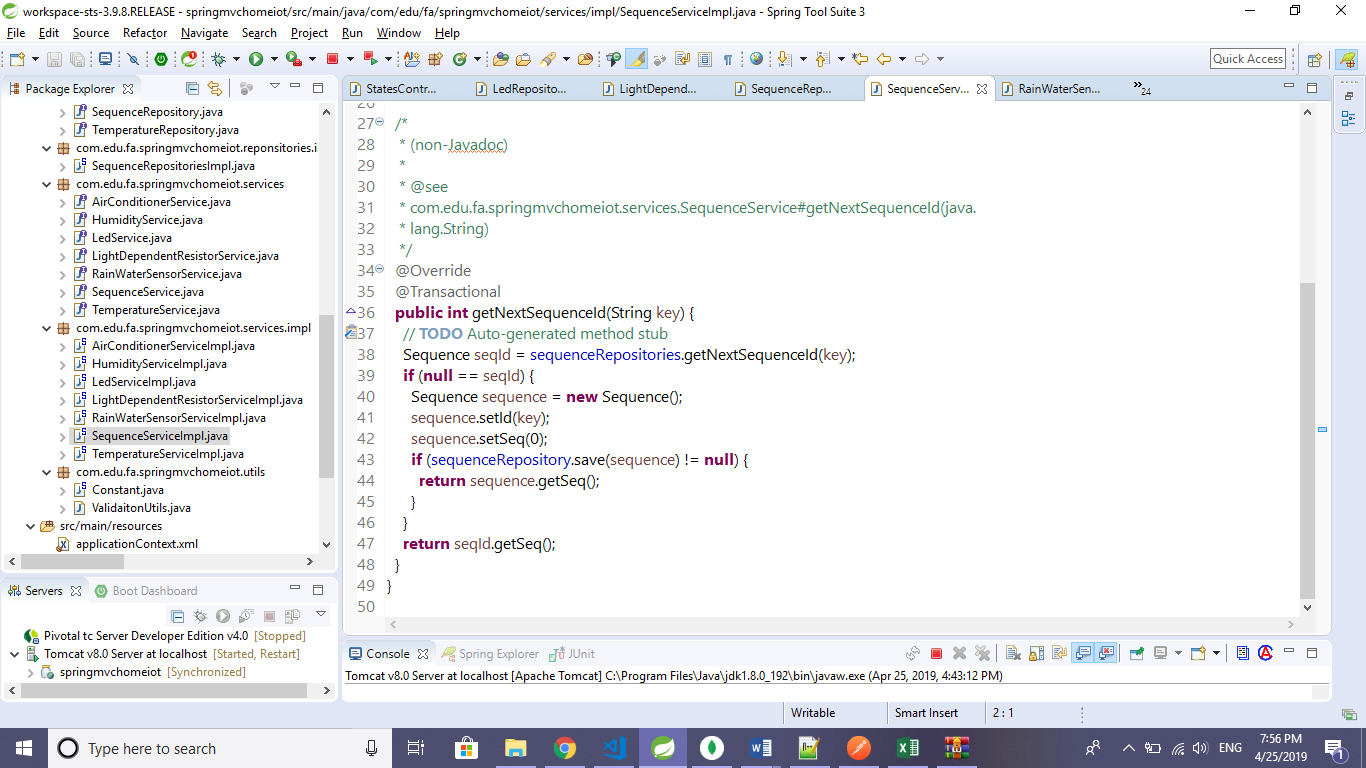
*Hình 2.17 Một class ở tầng Persistence*

* Ví dụ một class ở tầng Controller



*Hình 2.18 Một class ở tầng Controller*

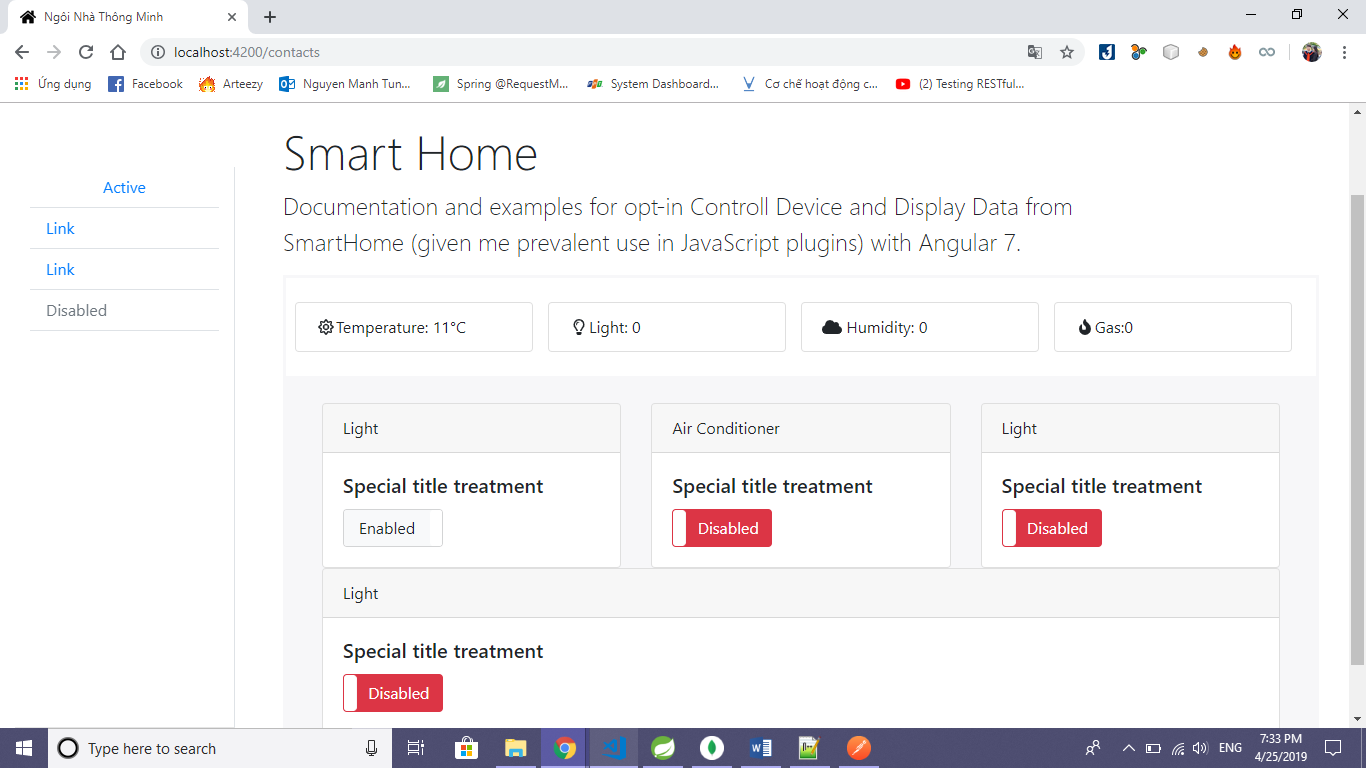
* Ví dụ một class ở tầng Service



*Hình 2.19 Một Class ở tầng Service*

#### **2.5.5 Website(Hệ thống View) với Angular 7**

Website được xây dựng tạo thành nơi để gia chủ có thể giao tiếp, kiểm soát với những gì đang diễn ra trong ngôi nhà của mình.



*Hình 2.20 Giao diện trang chủ với những thông số cụ thể trong nhà*

Mỗi khi nhận được lệnh từ phía người dùng, trang web sẽ gửi một yêu cầu (request) lên tới Web service thông qua API. Từ đó dữ liệu sẽ được cập nhật lại, và bộ kit nhận được sự thay đổi sẽ thực hiện ra lệnh cho các thiết bị điện tử trong gia đình.Hệ thống giao diện điều khiển ngôi nhà sử dụng Framework Angular để gửi request lên Webservice.

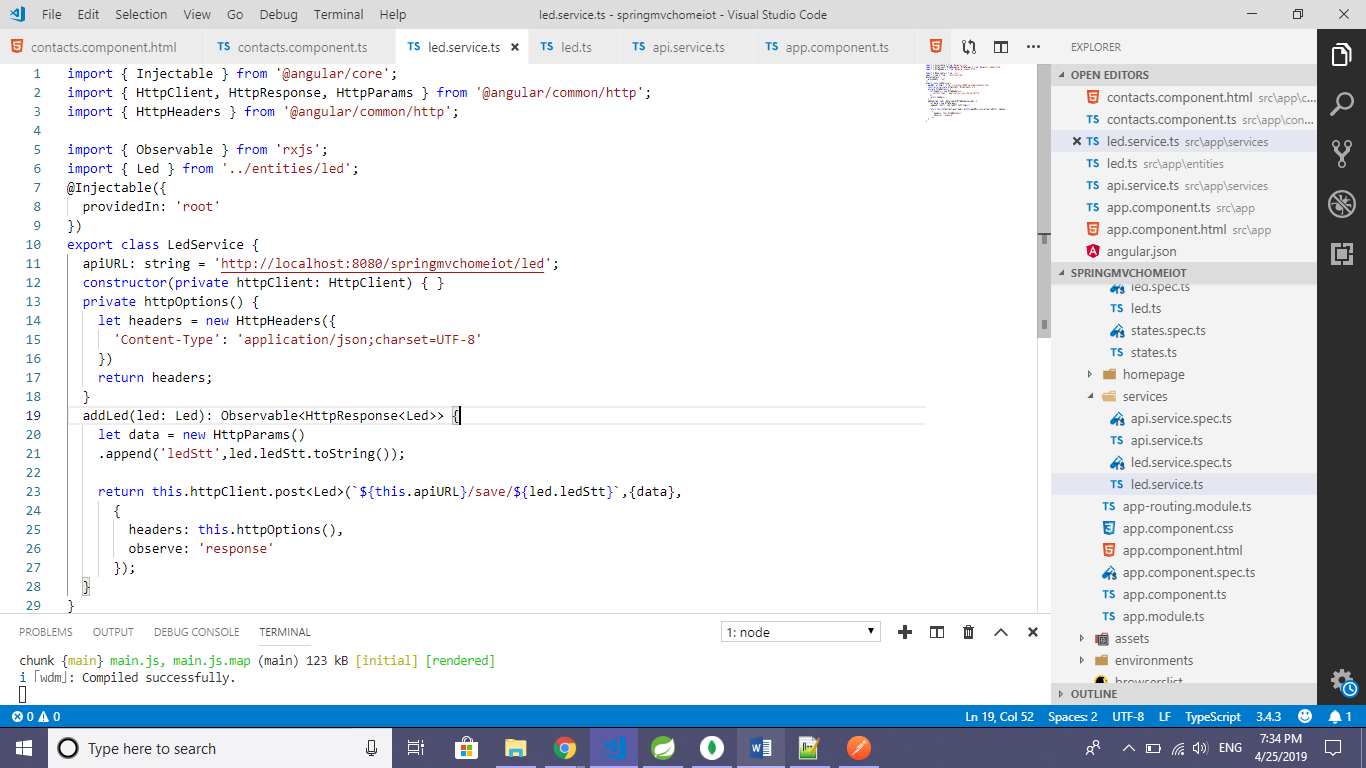
**2.5.5.1 Angular là gì?**

Angular 2 là 1 framework UI để xây dựng ứng dụng web trên desktop và mobile.Nó được xây dựng dựa trên Javascript. Chúng ta có thể dùng nó để xây dựng 1 ứng dụng client side thú vị dùng HTML, CSS và Javascript.Angular 2 có rất nhiều cải tiến so với angular 1 để dễ dàng học và phát triển các ứng dụng quy mô doanh nghiệp.Với angular 2 thì chúng ta dễ dàng xây dựng được một ứng dụng có thể dễ dàng mở rộng, bảo trì, kiểm nghiệm và chuẩn hóa ứng dụng của mình. Angular 2 hoàn toàn được viết bằng Typescript. Điều đó đồng nghĩa là nó hỗ trợ cho ES6 Modules, class frameworks, đồng thời cung cấp các component riêng rẽ. Mỗi một component đại diện cho 1 phần tử UI. Chúng ta có thể có nhiều component trong 1 single web page. Các component là độc lập với nhau và quản lý 1 vùng của trang. Component có thể có component con và component cha. Từ đó ứng dụng khi xây dựng sẽ giảm tính phụ thuộc và tăng tính kết dính.

**2.5.5.2 Các tính năng trong Angular 2**

Dưới đây là các tính năng nổi bật trong angular 2

* Two-way data binding đây là một trong những tính năng tuyệt với nhất trong angular 2. Dữ liệu được binding một cách tự động và nhanh chóng, những thay đổi trong view sẽ được tự động cập nhật vào trong các component class.
* Powerful routing support Angular 2 hỗ trợ mạnh mẽ các routing thông qua cách tải trang không đồng bộ trên cùng một trang cho phép chúng ta tạo ra một single page application.
* Expressive HTML Angular 2 cho phép chúng ta dùng các cấu trúc lập trình như câu lệnh if, vòng lặp for, .. để render data và kiểm soát các trang HTML.
* Modular by design Angular 2 được thiết kế theo hướng modul hóa để tổ chức và quản lý code một cách tốt hơn.
* Built in back end support Angular 2 được xây dựng để hỗ trợ việc giao tiếp với back-end servers và thực thi bất kỳ business logic hoặc lấy dữ liệu.
* Active community Angular 2 được hỗ trợ bởi google và có một cộng đồng đông đảo sẵn sàng hỗ trợ và giải đáp bất cứ câu hỏi nào của bạn.



*Hình 2.21 Xử lý dữ liệu với Angular*

1. **Chương trình DEMO**

**KẾT LUẬN**

Công nghệ IoT nói chung và webservice hẹn tạo ra những ứng dụng đầy tiềm năng, có thể áp dụng trong nhiều lĩnh vực khác nhau, mà đối với các công nghệ khác còn nhiều hạn chế. Nhờ quá trình mô phỏng người thiết kế hệ thống có thể đánh giá được chất lượng dịch vụ mạng cung cấp, để từ đó có thể thiết kế hệ thống theo cách tối ưu nhất. Ngoài ra hệ thống có thể dễ dàng triển khai trên một máy chủ web(tomcat,websphere,…) trên các máy tính nhúng cỡ nhỏ như là Orange Pi hoặc Raspberry Pi

Trên cơ sở nghiên cứu tổng quan về IoT, công nghệ webservice và các ứng dụng trong thực tiễn, em đã xây dựng thành công một chương trình thực nghiệm có tính khả thi cao (như đã trình bày trong phần đánh giá kết quả chạy thử nghiệm). Tuy nhiên, do thời gian và số lượng các cảm biến hạn chế, chương trình chưa chạy thử nghiệm với một số lượng lớn các thiết bị cảm biến, vì vậy chưa đánh giá hết được một số vấn đề như: việc truyền nhận dữ liệu từ thiết bị cảm biến đến các node mạng cảm biến, vấn đề xung đột dữ liệu… Đây cũng là một trong những hướng nghiên cứu, phát triển tiếp theo của đề tài.

**TÀI LIỆU THAM KHẢO**

1. Luigi Atzori, Antonio Iera, Giacomo Morabito, *Internet of Things: A survey*, Computer Networks 54 (2010) 2787–2805.

2. Dr. Ovidiu Vermesan, Dr. Peter Friess, Patrick Guillemin, *Internet of Things Strategic Research Roadmap*, 2009 Strategic Research Agenda, The IoT European Research Cluster - European Research Cluster on the Internet of Things (IERC).

3. Everton Cavalcante, Marcelo Pitanga Alves, *An Analysis of Reference Architectures for the Internet of Things*, Corba 2015.

4. Anna Ha’c, *Wireless Sensor Network Designs*, University of Hawaii at Manoa, Honolulu, USA, John Wiley & Sons Ltd, Copyright 2003.

# 5. Gaurav Kumar Arora, *SOLID Principles Succinctly,* Syncfusion, Technology Resource Portal, October 31, 2016.

6. *Pablo’s SOLID Software Development*, LosTechies.com.

7. <https://spring.io/guides/gs/rest-service/>

8. [https://docs.spring.io/spring-data/mongodb/docs/](https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/)

9. <https://docs.spring.io/spring/docs/current/spring-framework-reference/>

10. <https://www.arduino.cc/en/Main/Education>

11. <https://angular.io/docs>