TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

KHOA CÔNG NGHỆ ĐIỆN TỬ - THÔNG TIN

BÁO CÁO

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC SINH VIÊN

IOT VÀ MÔ HÌNH NÔNG NGHIỆP THÔNG MINH

Nhóm ngành: Điện tử - thông tin

Giảng viên hướng dẫn: ThS. Đặng Hoàng Anh

Sinh viên thực hiện : Nguyễn Thị Ngọc Lan

Nguyễn Văn Mạnh

Nguyễn Mạnh Quân

Khoa : Công nghệ Điện tử - Thông tin

Trường : Đại học Mở Hà Nội

Ngành học: Công nghệ kỹ thuật Điện Tử-Viễn thông

Hà Nội, 2019

DANH SÁCH THÀNH VIÊN THAM GIA NGHIÊN CỨU ĐỀ TÀI

VÀ ĐƠN VỊ PHỐI HỢP CHÍNH

Danh sách thành viên tham gia nghiên cứu đề tài

TT	**Họ và tên**	**Lớp/ Khóa học**
1.	Nguyễn Thị Ngọc Lan (trưởng nhóm)	K18A khoa CN Điện Tử - Thông Tin
2.	Nguyễn Văn Mạnh	K18A khoa CN Điện Tử - Thông Tin
3.	Nguyễn Mạnh Quân	K20A khoa CN Điện Tử - Thông Tin

TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

KHOA CN ĐIỆN TỬ - THÔNG TIN

THÔNG TIN KẾT QUẢ NGHIÊN CỨU

1. Thông tin chung

Tên đề tài: IoT và mô hình nông nghiệp thông minh

Nhóm sinh viên thực hiện: Nguyễn Thị Ngọc Lan (trưởng nhóm)

Nguyễn Văn Mạnh

Nguyễn Mạnh Quân

Khoa: CN Điện Tử - Thông Tin

Trường: Đại học Mở Hà Nội

Ngành học: Công nghệ kỹ thuật Điện Tử-Viễn thông

2. Mục tiêu

Thiết kế, xây dựng mô hình nông nghiệp thu nhỏ "Tiny garden" thu thập dữ liệu nhiệt độ độ ẩm không khí, độ ẩm đất, ánh sáng từ môi trường, điều khiển và tự động hóa thiết bị.

3. Tính cấp thiết, tính mới

Tính cấp thiết

Hiện nay cùng với sự phát triển của xã hội, cuộc sống ngày càng được nâng cao thì việc áp dụng công nghệ khoa học kỹ thuật vào đời sống công việc ngày càng cần thiết. Cùng với sự phát triển của các ngành khoa học kỹ thuật, công nghệ điện tử mà trong đó đặc biệt là kỹ thuật điều khiển tự động đóng vai trò quan trọng trong mọi lĩnh vực khoa học kỹ thuật, quản lý, công nghiệp, nông nghiệp, đời sống, quản lý thông tin,...

Ngành nông nghiệp trong một thập niên vừa qua có tốc độ tăng trưởng tương đối cao nhưng không đáp ứng được yêu cầu an ninh lương thực toàn cầu. Với dân số đã cán mốc 9 tỷ người, tình hình lương thực để đáp ứng được tiêu thụ sẽ là một bài toán lớn không chỉ riêng cho một quốc gia nào, bên cạnh cả những vấn đề phòng trị dịch bệnh , an toàn thực phẩm gia tăng đột biến trong tương lai.

Vậy, giải pháp cho chăn nuôi làm thế nào thế nào để đảm bảo an ninh lương thực trong tình hình hảng giả, hàng kém chất lượng trên thị trường, các tình trạng xung đột và bất ổn, dịch bệnh, biến đổi khí hậu diễn biến càng ngày càng phức tạp.

Tình hình trên thế giới

Quá trình phát triển nông nghiệp trên thế giới đến nay cũng như quá trình phát triển các cuộc cách mạng công nghiệp. Với cuộc cách mạng nông nghiệp 4.0 phát triển diễn ra đồng thời với sự phát triển của thế giới về công nghiệp 4.0. Thuật ngữ nông nghiệp 4.0 được sử dụng lần đầu tiên tại Đức năm 2011.

Công nghệ 4.0 là sự phát triển mạnh mẽ các thiết bị cảm biến có kết nối Internet (IoT); các thiết bị bay không người lái; công nghệ đèn LED; Robot quản trị, nhà thông minh, thành phố thông minh…

Điểm nổi bật của mô hình nông nghiệp thông minh trên thế giới đó là cơ sở hạ tầng, thiết bị, phạm vi quy mô ứng dụng lớn, được sử dụng nhiều, đặc biệt như ở Nhật Bản, một nước nghèo về tài nguyên thiên nhiên.

Tình hình trong nước

Tại Việt Nam, IoT đã được ứng dụng từ lâu dưới các hình thức tự động hóa hệ thống hóa như hệ thống điều khiển đèn giao thông, hệ thống tưới tiêu tự động,…Tuy nhiên, chỉ đến năm 2015 thì khái niệm IoT ở Việt Nam mới được nhắc đến nhiều thông qua các hội thảo, hội nghị về xu hướng công nghệ của Cisco, Inter, Hội tin học TP.HCM, và một số công ty trong nước như Mobiphone, DTT, Sao Bắc Đẩu,…

Tuy nhiên, hiện tại chưa có ứng dụng IoT thực sự nào ảnh hưởng tới đời sống xã hội trong nước. Với giao thông đô thị thông minh, trong thời gian tới một số ứng dụng như thu phí không dừng, phạt nguội bằng camera dự báo sẽ phổ biến tại các thành phố lớn như TP HCM, Hà Nội. Các lĩnh vực tiềm năng như y tế điện tử, nông nghiệp thông minh, bất động sản thông minh sẽ cần thêm thời gian để có những ứng dụng IoT phù hợp với Việt Nam.

Trong doanh nghiệp nội địa, sản phẩm IoT của doanh nghiệp trong nước hiện chỉ đếm được trên đầu ngón tay như: sản phẩm chip vi mạch của Trung tâm Nghiên cứu và đào tạo thiết kế vi mạch, hệ thống cảm ứng độ ẩm, nhiệt độ trong nông nghiệp của công ty Mimosa tại hệ sinh thái khởi nghiệp Công nghệ- Khu công nghệ phần mềm Đại học Quốc gia TP HCM, chương trình TUHOC STEM và các dịch vụ trên nền OEP của công ty DTT (trụ sở chính tại Hà Nội).

Mặt khác, các hệ thống IoT ở Việt Nam hiện có đều là của các doanh nghiệp nước ngoài, các doanh nghiệp trong nước chỉ mới tập trung vào các ứng dụng trên nền tảng điện thoại di động, máy tính và còn chưa khai thác hết tính thông minh của hệ thống cảm biến hay khai thác dữ liệu Big Data. Đặc biệt, các thiết bị phần cứng thì hầu hết là nhập khẩu như camera, thiết bị RFID hay các cảm biến hóa học.

Thực tế giải pháp cho chăn nuôi sản xuất ở nước ta phụ thuộc vào vùng sinh thái; loại cây trồng, vật nuôi; quy mô sản xuất, do đó chủ trang trại không nhất thiết phải ứng dụng toàn bộ sang công nghệ; có thể kết hợp phát triển nông nghiệp song song để tìm những giải pháp tối ưu nhất qua lại trước khi áp dụng toàn bộ công nghệ vào phát triển nông nghiệp. Vẫn phải hướng đến mục tiêu hiệu quả kinh tế là chính, song việc ứng dụng IoT là công nghệ cốt lõi cần trong tương lai gần, đó là phát triển nông nghiệp bền vững trong điệu biến đổi khí hậu và đất nông nghiệp ngày càng bị thu hẹp.

Hiện nay nông nghiệp thông minh ở nước ta mới chỉ phổ biến ở mức tự động: hẹn giờ, xử lý tại chỗ… mà chưa có hệ thống giám sát, quản lý. Điều này thực sự cần thiết đối với các sản phẩm nông nghiệp tạo hiệu quả kinh tế, giảm thiệt hại do điều kiện môi trường, khí hậu.

Mô hình mà chúng em đưa ra kết hợp giữa tự động và giám sát hệ thông với nhiều chức năng có thể phát triển thêm. Tuy chưa được hoàn thiện một cách hoàn hảo nhưng phần nào đáp ứng được nhu cầu bức thiết hiện tại.

4. Tổng quan tình hình nghiên cứu

Với phạm vi nghiên cứu của sinh viên, chúng em tạo ra mô hình thu nhỏ của hệ sinh thái nông nghiệp gồm đất – nước khá hoàn chỉnh đáp ứng được những nhu cầu cơ bản của thực tế sản phẩm nếu đưa vào thực tế áp dụng.

5. Mục tiêu và phạm vi nghiên cứu

5.1. Mục tiêu nghiên cứu:

Thực tế đối với từng loại cây, hoa lại có từng yêu cầu về môi trường khác nhau. Quản lý thời gian cây ra hoa, quả đúng thời điểm đáp ứng nhu cầu của người tiêu dùng sẽ đem lại hiệu quả kinh tế rất lớn. Nhờ đó giảm thiểu được việc cầu vượt quá cung dẫn đến việc rất nhiều sản phẩm nông nghiệp của người nông dân không có nơi tiêu thụ mà bị bỏ hỏng.

Trước vấn đề đó chúng em đã xây dựng ý tưởng về khu vườn thu nhỏ "Tiny garden" làm mô hình cho hệ thống giám sát và điều khiển hệ sinh thái nông nghiệp kiểm soát độ ẩm, nhiệt độ không khí, độ ẩm đất, ánh sáng môi trường đồng thời điều khiển và tự động một số chức năng giúp tiết kiệm nhân công và giảm thời gian chăm sóc. Mô hình hệ thống này thân thiện với người dùng, dễ dàng tiện lợi khi ở bất kì đâu có kết nối mạng internet và truy cập được trình duyệt web đều có thể giám sát điều khiển được hệ thống.

Nhờ vào những tính năng đó mà hiệu quả kinh tế từ nông nghiệp sẽ tăng lên đem lại giá trị sản phẩm tốt hơn.

5.2. Phạm vi nghiên cứu:

Với mục đích của đề tài chúng em thực hiện nghiên cứu với mô hình nhỏ bao gồm bể nước và chậu cây được thiết kế trồng tầng lên nhau, tạo cảnh quan giống mô hình tiểu cảnh có thể đặt trong gia đình tạo cảnh quan xanh mát.

6. Phương pháp nghiên cứu

Với khả năng nghiên cứu cấp sinh viên trong lĩnh vực điện tử - thông tin, chúng em đưa ra đề tài với hướng giải quyết như sau:

- Tham khảo một số loại cây đặc thù
- Nghiên cứu chung về IoT
- Nghiên cứu về board mạch arduino là bộ xử lý trung tâm, trái tim của hệ thống
- Sử dụng mạch thu phát wifi Esp8266
- Sử dụng các loại cảm biến thông dụng để cập nhật những thông số từ môi trường
- Lập trình kết nối các thiết bị với nhau
- Tạo mô hình, giao diện ứng dụng web điều khiển giám sát hệ thống thân thiện với người dùng sử dụng ngôn ngữ JavaScript với Nodejs

7. Những đóng góp của đề tài

7.1 Về mặt khoa học

Với đề tài này chúng em tập chung nghiên cứu về IoT nói chung và ứng dụng của nó trong nông nghiệp của nước ta hiện nay với việc kết nối phần cứng của mạch, lập trình cho vi điều khiển, lập trình giao diện ứng dụng web thân thiện với người dùng, dễ dàng tiện lợi khi ở bất kì đâu có kết nối mạng internet và truy cập được trình duyệt web đều có thể giám sát điều khiển được hệ thống.

7.2 Về mặt thực tế

Về thực tế áp dụng với mô hình nhỏ nhóm em tại ra có thể sử dụng ngày trong gia đình tao cảnh quanh sinh thái hoặc để cung cấp nguồn thực phẩm sạch cho chính giá đình mà không mất quá nhiều công chăm sóc. Mở rộng phát triển hơn khi đưa vào mô hình nhà kính hay trang trại sẽ được lắp đặt tùy thuộc và từng nơi và đặc điểm của nó. Nếu phát triển nông nghiệp theo hướng hiện đại hay công nghiệp thì việc kiểm soát được hệ sinh thái nông nghiệp là rất quan trọng. Với sự đơn giản của nó thì

bất kì người nông dân nào cũng có thể sử dụng. nó càng có ý nghĩa hơn đối với các ký sư nông nghiệp khi các thông số môi trường đối với họ là rất quan trọng trong việc đưa ra giải pháp cho những sản phẩm nông nghiệp của mình.

8. Nội dung chính:

- Phần 1: Tìm hiểu về IoT
- Khái niệm
- Những ứng dụng
- Khó khăn triển vọng
- Kiến trúc tham chiếu của hệ thống
- Hạ tầng và công nghệ
- Phần 2: Xây dựng mô hình hệ thống "Tiny garden"
- Phân tích yêu cầu, giải pháp giải quyết đề tài
- Nguyên lý hoạt động của hệ thống
- Tìm hiểu về các linh hiện điện tử sử dụng trong mô hình: Arduino UNO R3, Wemos D1 R2, module cảm biến độ ẩm đất, ánh sáng với quang trở, dht 11
- Nắm bắt kiến thức về các ngôn ngữ lập trình sử dụng: C, JavaScript, html, framework AngularJS

MỞ ĐẦU

Hiện nay cùng với sự phát triển của xã hội, cuộc sống ngày càng được nâng cao thì việc áp dụng công nghệ khoa học kỹ thuật vào đời sống công việc ngày càng cần thiết. Cùng với sự phát triển của các ngành khoa học kỹ thuật, công nghệ điện tử mà trong đó đặc biệt là kỹ thuật điều khiển tự động đóng vai trò quan trọng trong mọi lĩnh vực khoa học kỹ thuật, quản lý, công nghiệp, nông nghiệp, đời sống, quản lý thông tin,...

Nước ta là một đất nước nông nghiệp, tuy nhiên trong nhiều năm quy mô cũng như chất lượng và sản lượng nông nghiệp của nước ta luôn thấp hơn so với các nước khác mà nguyên nhân chính là việc công nghệ sản xuất của nước ta quá lạc hậu, chủ yếu dựa vào tay chân. Do đó, IoTs đã và đang dẫn đầu trong việc cải thiện chất lượng cũng như năng suất nuôi trồng nông nghiệp nước ta hiện nay. Tất cả được điều chỉnh và điều khiển hoàn toàn tự động và áp dụng công nghệ khoa học kỹ thuật vào quy trình giám sát và sản xuất. Việc sử dụng nhà kính tự động giúp chúng ta có thể tiết kiệm nhân lực, tăng độ chính xác trong giám sát và điều khiển môi trường nhằm nâng cao chất lượng sản phẩm.

Trước vấn đề đó em đã xây dựng ý tưởng về khu vườn thu nhỏ “Tiny garden” làm mô hình cho hệ thống giám sát và điều khiển hệ sinh thái nông nghiệp gồm cây-cá, đất- nước. Với quy mô nhỏ có thể áp dụng trong hộ gia đình tạo cảnh quan sinh thái, phù hợp với người hay phải đi công tác nhiều ngày. Nếu phát triển đề tài có thể ứng dụng vào thực tế với quy mô nhà kính, trang trại.

MỤC LỤC

Trang

Phần I: IoT 1

Chương 1: Giới thiệu về IoT 1

1.1. IoT là gì? 1

1.2. Điểm mấu chốt 3

1.3. Ứng dụng 4

1.4. Khó khăn 8

1.5. Triển vọng và nhận định 10

Chương 2: Kiến trúc tham chiếu của IoT 11

2.1. Tổng quan 11

2.2. Giao thức dùng để kết nối trong IoT 11

2.3. Các yêu cầu của kiến trúc tham chiếu cho IoT 15

2.4. Mô hình tham chiếu 18

2.5. Hạ tầng mạng và điện toán đám mây 21

2.6. Công nghệ truyền thông trong IoT 23

Phần II: Xây dựng mô hình IoT 30

Chương 1: Phân tích đề tài 30

1.1. Yêu cầu 30

1.2. Giải pháp thiết kế 30

1.2.1. Sơ đồ khối 31

1.2.2. Nguyên lý hoạt động 32

1.2.3. Chọn linh kiện 33

1.2.4. Phần mềm lập trình 44

1.2.5. Ngôn ngữ lập trình 48

Chương 2: Tính toán và thiết kế 51

2.1. Thiết kế, lắp ráp mạch 51

2.2. Lập trình hệ thống: 51

2.2.1. Arduino 51

2.2.2. ESP8266 (Wemos D1 R2) 54

2.2.3. Lập trình server 56

2.3. Thi công và kết quả 62

2.4. Nhận xét đánh giá 67

2.5. Hướng phát triển..... 67
DANH MỤC TÀI LIỆU THAM KHẢO 68

DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH

Hình 1.1: Sự kết nối của Internet of things *2*
Hình 1.2. Mô hình công nghệ thành phần của IoT *4*
Hình 1.3. Giao thức MQTT *12*
Hình 1.4. Ví dụ về mô hình sử dụng giao thức CoAP và HTTP *12*
Hình 1.5. Giao thức AMQP *13*
Hình 1.6. Giao thức DDS *14*
Hình 1.7. ví dụ về XMPP *15*
Hình 1.8. Mô hình tham chiếu cho IoT của WSO2 ..... *18*
Hình 1.9. Điện toán đám mây *23*
Hình 2.1. Mô hình Socket Server và Socket Client *31*
Hình 2.2. Sơ đồ khối của Arduino UNO R3 ..... *33*
Hình 2.3. Sơ đồ chân của ATmega328P và board Arduino Uno R3 ..... *34*
Hình 2.4. Sơ đồ chân ESP8266EX *37*
Hình 2.5. Wemos D1 R2 ..... *39*
Hình 2.6. Sơ đồ chân Wemos D1 R2 ..... *40*
Hình 2.7. Sơ đồ chân cảm biên DHT11 ..... *41*
Hình 2.8. Module cảm biến độ ẩm đất và mạch nguyên lý *41*
Hình 2.11. Cài đặt board ESP8266 ..... *46*
Hình 2.12. Khởi động Library Manager *46*
Hình 2.14. Mô phỏng mạch bằng frizing *51*
Hình 2.15. Deploy this app use heroku *64*
Hình 2.16. Giao diện webapp trên máy tính khi kết nối *65*
Hình 2.17. Giao diện webapp trên điện thoại khi kết nối *65*
Hình 2.18. Chạy mạch thử nghiệm *66*

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

IoT	Internet of Things, Mạng lưới vạn vật kết nối Internet
AI	Artifical Intelligence Trí tuệ nhân tạo
SoC	system-on-a-chip, Hệ thống trên một vi mạch
HTTP	HyperText Transfer Protocol - Giao thức truyền tải siêu văn bản
POP	Post Office Protocol. Cả hai giao thức đều là giao thức email.
FTP	File Transfer Protocol dịch ra là "Giao thức truyền tập tin"
UDP	User Datagram Protocol trong giao thức cốt lõi của giao thức TCP/IP
DDS	diaminodiphenyl sulfone
XACML	eXtensible Access Control Markup Language
API	Application Programming Interface - giao diện lập trình ứng dụng
M2M	Machine-to-Machine
IP	Internet Protocol - - giao thức Internet
BCM	BUSINESS CONTINUITY MANAGEMENT
SMTP	Arpanet là Mail Box Protocol - giao thức truyền tải thư tín đơn giản
IMAP	Internet Message Access Protocol - giao thức chuẩn Internet
NFC	Near-Field Communications - giao thức kết nối giữa hai thiết bị điện tử ở khoảng cách gần
TCP	Transmission Control Protocol - "Giao thức điều khiển truyền vận"
UI	User Interface có nghĩa là giao diện người dùng
UUID	Universally Unique IDentifier

Phần I: IoT

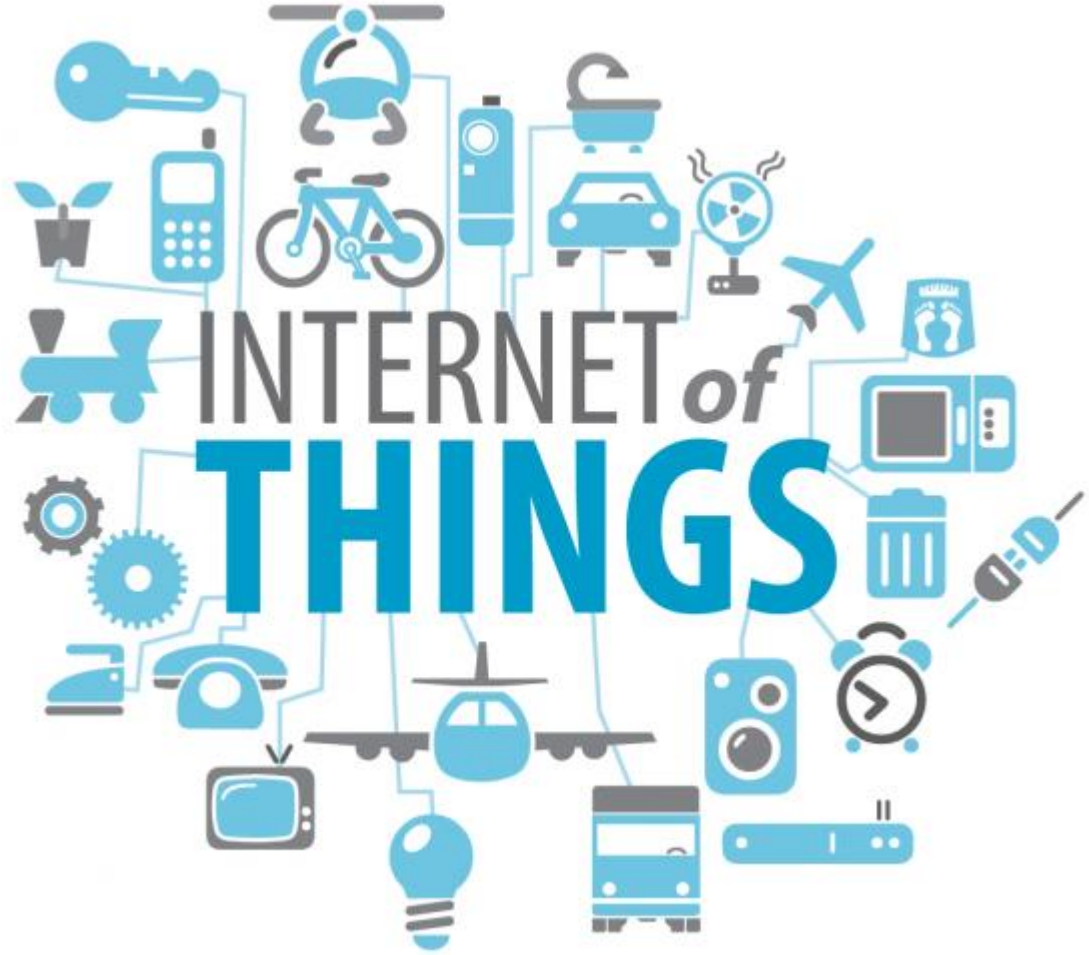
Chương 1: Giới thiệu về IoT

1.1. IoT là gì?

Chắc hẳn trong thời đại của nền Công nghiệp 4.0 chắc không mấy người chưa từng được nghe và nhắc đến IoT. Cách mạng công nghệ lần thứ 4 đó là sự phát triển của kỹ thuật số, công nghệ sinh học, vật lý bao gồm: trí tuệ nhân tạo, robot, 3d,big data và không thể thiếu, đó chính là *Internet of Things* viết tắt là IoT.

Internet of Things, hay cụ thể hơn là Mạng lưới vạn vật kết nối Internet hoặc là Mạng lưới thiết bị kết nối Internet là một liên mạng, trong đó các thiết bị, phương tiện vận tải (được gọi là "thiết bị kết nối" và "thiết bị thông minh"), phòng ốc và các trang thiết bị khác được nhúng với các bộ phận điện tử, phần mềm, cảm biến, cơ cấu chấp hành cùng với khả năng kết nối mạng máy tính giúp cho các thiết bị này có thể thu thập và truyền tải dữ liệu. Mạng lưới vạn vật kết nối Internet 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 là bộ cảm ứng được lắp ráp trong một chiếc tủ lạnh để ghi lại nhiệt độ, là một trái tim được cấy ghép trong cơ thể con người,… Hiểu đơn giản, IoT có thể khiến mọi vật giờ đây có thể giao tiếp với nhau dễ dàng hơn và ưu điểm lớn nhất của "Thông minh" là khả năng phòng ngừa và cảnh báo tại bất kì đâu.



Hình 1.1: Sự kết nối của Internet of things

Cụm từ Internet of things được đưa ra bởi Kevin Ashton vào năm 1999, tiếp sau đó nó cũng được dùng nhiều trong các ấn phẩm đến từ các hãng và nhà phân tích. Họ cho rằng IoT là một hệ thống phức tạp, bởi nó là một lượng lớn các đường liên kết giữa máy móc, thiết bị và dịch vụ với nhau. Ban đầu, IoT không mang ý nghĩa tự động và thông minh. Về sau, người ta đã nghĩ đến khả năng kết hợp giữa hai khái niệm IoT - Autonomous control lại với nhau. Nó có thể quan sát sự thay đổi và phản hồi với môi trường xung quanh, cũng có thể tự điều khiển bản thân mà không cần kết nối mạng. Việc tích hợp trí thông minh vào IoT còn có thể giúp các thiết bị, máy móc, phần mềm thu thập và phân tích các dữ liệu điện tử của con người khi chúng ta tương tác với chúng. Xu hướng tất yếu trong tương lai, con người có thể giao tiếp với máy móc chỉ qua mạng internet không dây mà không cần thêm bất cứ hình thức trung gian nào khác.

Câu hỏi đặt ra là, điều gì giúp IoT "thông minh" và "hiểu" con người? Ban đầu, người ta cho rằng Internet của vạn vật chủ yếu xoay quanh giao tiếp M2M (các thiết bị kết nối với nhau thông qua một thiết bị khác điều khiển). Nhưng khi hướng đến sự "thông minh hóa", đó không chỉ là giao tiếp giữa M2M nữa mà cần phải đề cập đến các cảm biến (sensor). Và cũng đừng lầm tưởng rằng Sensor là một cỗ máy hoạt động dưới sự vận hành của các thiết bị khác mà thực chất, nó tương tự như đôi mắt và đôi tai của loài người với sự ghi nhận liên tục những đo lường, định lượng, thu thập dữ liệu từ thế giới bên ngoài. Suy cho cùng, Internet of things đem đến sự kết nối giữa máy móc và cảm biến, và nhờ đến dữ liệu điện toán đám mây để mã hóa dữ liệu. Những ứng dụng điện toán đám mây là mắt xích quan trọng giúp cho Internet of things có thể hoạt động nhờ sự phân tích, xử lí và sử dụng dữ liệu mà các cảm biến thu thập được.

1.2. Điểm mấu chốt

Những vấn đề quan trọng nhất của hệ thống IoT bao gồm trí thông minh nhân tạo, kết nối, cảm biến và các thiết bị nhỏ nhưng mang tính cơ động cao, chúng được mô tả sơ lược như bên dưới:

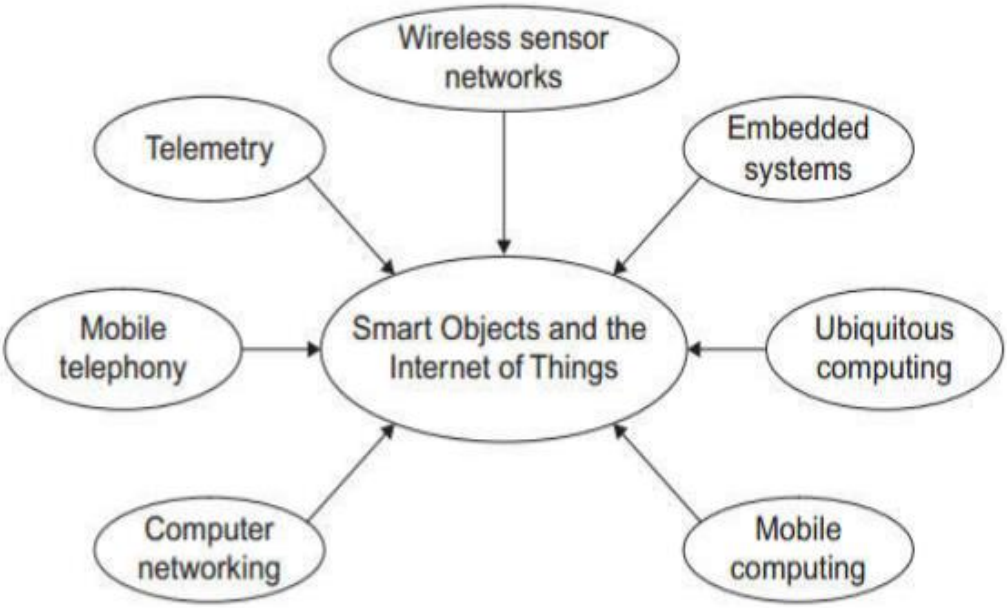
AI (Artifical Intelligence) - Hệ thống IoT về cơ bản được hiểu là làm cho mọi thiết bị trở nên thông minh, nghĩa là nó giúp nâng cao mọi khía cạnh của cuộc sống bằng những dữ liệu thu thập được, thông qua các thuật toán tính toán nhân tạo và kết nối mạng.

Connectivity - Là một đặc trưng cơ bản của IoT, hiện nay các mạng thiết bị đang trở nên phổ biến, nhiều mạng thiết bị ngày càng nhỏ hơn, rẻ hơn và được phát triển phù hợp với thực tế cũng như nhu cầu của người dùng .

Sensors - IoT sẽ mất đi sự quan trọng của mình nếu không có sensors. Các cảm biến hoạt động giống như một công cụ giúp IoT chuyển từ mạng lưới các thiết bị thụ động sang mạng lưới các thiết bị tích cực, đồng thời có thể tương tác với thế giới thực.

Active Engagement Ngày nay, phần lớn các tương tác của những công nghệ kết nối xảy ra 1 cách thụ động. IoT được cho là sẽ đem đến những hệ thống mang tích tích cực về nội dung, sản phẩm cũng như các dịch vụ gắn kết.

Small Devices - Như đã được dự đoán từ trước, các thiết bị ngày càng được tối ưu với mục đích nâng cao độ chính xác, khả năng mở rộng cũng như tính linh hoạt. Nó được thiết kế ngày càng nhỏ hơn, rẻ hơn và mạnh mẽ hơn theo thời gian.



Hình 1.2. Mô hình công nghệ thành phần của IoT

1.3. Ứng dụng

Theo Gartner, Inc. (một công ty nghiên cứu và tư vấn công nghệ), sẽ có gần 26 tỷ thiết bị trên IoT vào năm 2020. ABI Research ước tính rằng hơn 30 tỷ thiết bị sẽ được kết nối không dây với "Kết nối mọi thứ" (Internet of Everything) 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 / Cloud of Things, nhúng và tính toán đeo (và các hệ thống năng động, tương ứng)sẽ có tác động rộng rãi và mang lại lợi ích đến năm 2025. Như vậy, rõ ràng là IoT sẽ bao gồm một số lượng rất lớn các thiết bị được kết nối với Internet.

Tích hợp với mạng Internet có nghĩa rằng thiết bị này sẽ sử dụng một địa chỉ IP như là một định danh duy nhất. Tuy nhiên, do sự hạn chế không gian địa chỉ của IPv4 (cho phép 4,3 tỷ địa chỉ duy nhất), các đối tượng trong IOT sẽ phải sử dụng IPv6 để phù hợp với không gian địa chỉ cực kỳ lớn cần thiết. Các đối tượng trong IoT sẽ không chỉ có các thiết bị có khả năng cảm nhận xung quanh, mà còn cung cấp khả năng truyền động (ví dụ, củ hoặc khóa điều khiển thông qua Internet) Ở một mức độ lớn, tương lai của Internet of Things sẽ không thể không có sự hỗ trợ của IPv6; và do đó việc áp dụng toàn cầu của IPv6 trong những năm tới sẽ rất quan trọng cho sự phát triển thành công của IOT trong tương lai.

Khả năng kết nối vào mạng của thiết bị nhúng với CPU, bộ nhớ giới hạn và năng lượng bền bỉ. IoT được ứng dụng trong hầu hết các lĩnh vực. Hệ thống như vậy có thể có nhiệm vụ thu thập thông tin trong các thiết lập khác nhau, từ các hệ sinh thái tự nhiên cho các tòa nhà và các nhà máy, do đó việc tìm kiếm các ứng dụng trong lĩnh vực cảm biến môi trường và quy hoạch đô thị.

Mặt khác, hệ thống IoT cũng có thể thực hiện các hành động, không chỉ cảm nhận mọi thứ xung quanh. Hệ thống mua sắm thông minh, ví dụ, có thể theo dõi thói quen mua người dùng cần ở một cửa hàng bằng cách theo dõi điện thoại di động của họ. Người dùng sau đó có thể được cung cấp các cập nhật trên sản phẩm yêu thích của họ, hoặc thậm chí là vị trí của các mục mà họ cần, hay tủ lạnh của họ cần. Tất cả đã tự động chuyển vào điện thoại. Ví dụ bổ sung các cảm biến trong các ứng dụng phản ứng lại với nhiệt độ môi trường, điện và quản lý năng lượng, cũng như hỗ trợ hành trình của các hệ thống giao thông vận tải.

Tuy nhiên, các ứng dụng của IoT không chỉ giới hạn trong các lĩnh vực này. Trường hợp sử dụng chuyên ngành khác của IoT cũng có thể tồn tại. Một cái nhìn tổng quan về một số lĩnh vực ứng dụng nổi bật nhất được cung cấp ở đây. Dựa trên các miền ứng dụng, sản phẩm IoT có thể chia thành năm loại khác nhau: thiết bị đeo thông minh, nhà thông minh, thành phố thông minh, môi trường thông minh, và doanh nghiệp thông minh. Các sản phẩm và giải pháp IoT trong mỗi thị trường có đặc điểm khác nhau.

IoT có ứng dụng rộng vô cùng, có thể kể ra một số thứ như sau:

- ✓ Quản lý chất thải
- ✓ Quản lý và lập kế hoạch quản lý đô thị
- ✓ Quản lý môi trường
- ✓ Phản hồi trong các tinh huống khẩn cấp
- ✓ Mua sắm thông minh
- ✓ Quản lý các thiết bị cá nhân
- ✓ Đồng hồ đo thông minh
- ✓ Tự động hóa ngôi nhà

Một trong những vấn đề với IoT đó là khả năng tạo ra một ứng dụng IoT nhanh chóng. Để khắc phục, hiện nay nhiều hãng, công ty, tổ chức trên thế giới đang nghiên cứu các nền tảng giúp xây dựng nhanh ứng dụng dành cho IoT. Đại học British Columbia ở Canada hiện đang tập trung vào một bộ toolkit cho phép phát triển phần mềm IoT chỉ bằng các công nghệ/tiêu chuẩn Web cũng như giao thức phổ biến. Công ty như ioBridge thì cung cấp giải pháp kết nối và điều khiển hầu như bất kì thiết bị nào có khả năng kết nối Internet, kể cả đèn bàn, quạt máy...

Broadcom mới đây cũng đã giới thiệu hai con chip có mức tiêu thụ điện thấp và giá rẻ dành cho các thiết bị "Internet of things". SoC đầu tiên, BCM4390, được tích hợp một bộ thu phát sóng Wi-Fi 802.11 b/g/n hiệu suất cao để có thể dùng với các vi điều khiển 8 hoặc 16-bit. Broadcom nói rằng sản phẩm này có thể dùng trong các nồi nấu ăn thông minh, bóng đèn, hệ thống an ninh cũng như các thiết bị gia dụng có khả năng điều khiển và quản lý từ xa. SoC thứ hai, BCM20732, thì được tích hợp bộ thu phát tín hiệu Bluetooth và nhắm đến những máy móc như bộ đo nhịp tim, bộ đo bước chạy, thiết bị cảnh báo khi có vật gì đến gần hoặc ổ khóa cửa thông minh. Broadcom cũng đã đóng góp các tập lệnh phần mềm hỗ trợ cho cả công nghệ Bluetooth thường và Bluetooth Smart vào dự án Android Open Source (AOSP). Hiện bản mẫu của hai con chip này đang được giao đến đối tác phần cứng và dự kiến sẽ được sản xuất đại trà trong quý 4 năm nay.

Quản lý hạ tầng

Giám sát và kiểm soát các hoạt động của cơ sở hạ tầng đô thị và nông thôn như cầu, đường ray tàu hỏa, và trang trại là một ứng dụng quan trọng của IoT. Các cơ sở hạ tầng IoT có thể được sử dụng để theo dõi bất kỳ sự kiện hoặc những thay đổi trong điều kiện cơ cấu mà có thể thỏa hiệp an toàn và làm tăng nguy cơ. Nó cũng có thể được sử dụng để lập kế hoạch hoạt động sửa chữa và bảo trì một cách hiệu quả, bằng cách phối hợp các nhiệm vụ giữa các nhà cung cấp dịch vụ khác nhau và người sử dụng của các cơ sở này. Thiết bị IoT cũng có thể được sử dụng để kiểm soát cơ sở hạ tầng quan trọng như cầu để cung cấp truy cập vào tàu. Cách sử dụng của các thiết bị iốt để theo dõi và hạ tầng hoạt động có khả năng cải thiện quản lý sự cố và phối hợp ứng phó khẩn cấp, và chất lượng dịch vụ, tăng lần và giảm chi phí hoạt động trong tất cả các lĩnh vực cơ sở hạ tầng liên quan. Ngay cả các lĩnh vực như quản lý chất thải đứng được hưởng lợi từ tự động hóa và tối ưu hóa có thể được đưa vào bởi IoT.

Y tế

Thiết bị IoT có thể được sử dụng để cho phép theo dõi sức khỏe từ xa và hệ thống thông báo khẩn cấp. Các thiết bị theo dõi sức khỏe có thể dao động từ huyết áp và nhịp tim màn với các thiết bị tiên tiến có khả năng giám sát cấy ghép đặc biệt, chẳng hạn như máy điều hòa nhịp hoặc trợ thính tiên tiến. Cảm biến đặc biệt cũng có thể được trang bị trong không gian sống để theo dõi sức khỏe và thịnh vượng chung là người già, trong khi cũng bảo đảm xử lý thích hợp đang được quản trị và hỗ trợ người dân lấy lại mất tính di động thông qua điều trị là tốt. Thiết bị tiêu dùng khác để khuyến khích lối sống lành mạnh, chẳng hạn như, quy mô kết nối hoặc máy theo dõi tim mạch, cũng là một khả năng của IoT.

Xây dựng và tự động hóa nhà

Thiết bị IoT có thể được sử dụng để giám sát và kiểm soát các hệ thống cơ khí, điện và điện tử được sử dụng trong nhiều loại hình tòa nhà (ví dụ, công cộng và tư nhân, công nghiệp, các tổ chức, hoặc nhà ở). Hệ thống tự động hóa, như các tòa nhà tự động hóa hệ thống, thường được sử dụng để điều khiển chiếu sáng, sưởi ấm, thông gió, điều hòa không khí, thiết bị, hệ thống thông tin liên lạc, giải trí và các thiết bị an ninh gia đình để nâng cao sự tiện lợi, thoải mái, hiệu quả năng lượng và an ninh.

Giao thông

Các sản phẩm IoT có thể hỗ trợ trong việc tích hợp các thông tin liên lạc, kiểm soát và xử lý thông tin qua nhiều hệ thống giao thông vận tải. Ứng dụng của IoT mở rộng đến tất cả các khía cạnh của hệ thống giao thông, tức là xe, cơ sở hạ tầng, và người lái xe hoặc sử dụng. Năng động, tương tác giữa các thành phần của một hệ thống giao thông vận tải cho phép truyền thông giữa nội và xe cộ, điều khiển giao thông thông minh, bãi đậu xe thông minh, hệ thống thu phí điện tử, quản lý đội xe, điều khiển xe, an toàn và hỗ trợ đường bộ.

1.4. Khó khăn

Chưa có một ngôn ngữ chung

Ở mức cơ bản nhất, Internet là một mạng dùng để nối thiết bị này với thiết bị khác. Nếu chỉ riêng có kết nối không thôi thì không có gì đảm bảo rằng các thiết bị biết cách nói chuyện nói nhau. Cũng giống như là bạn có thể đi từ Việt Nam đến Mỹ, nhưng không đảm bảo rằng bạn có thể nói chuyện với người Mỹ.

Để các thiết bị có thể giao tiếp với nhau, chúng sẽ cần một hoặc nhiều giao thức (protocols), có thể xem là một thứ ngôn ngữ chuyên biệt để giải quyết một tác vụ nào đó. Chắc chắn bạn đã ít nhiều sử dụng một trong những giao thức phổ biến nhất thế giới, đó là HyperText Transfer Protocol (HTTP) để tải web. Ngoài ra chúng ta còn có SMTP, POP, IMAP dành cho email, FTP dùng để trao đổi file.

Những giao thức như thế này hoạt động ổn bởi các máy chủ web, mail và FTP thường không phải nói với nhau nhiều, khi cần, một phần mềm biên dịch đơn giản sẽ đứng ra làm trung gian để hai bên hiểu nhau. Còn với các thiết bị IoT, chúng phải đảm đương rất nhiều thứ, phải nói chuyện với nhiều loại máy móc thiết bị khác nhau. Đáng tiếc rằng hiện người ta chưa có nhiều sự đồng thuận về các giao thức để IoT trao đổi dữ liệu. Nói cách khác, tình huống này gọi là "giao tiếp thất bại", một bên nói nhưng bên kia không thể nghe.

Hàng rào subnetwork

Như đã nói ở trên, thay vì giao tiếp trực tiếp với nhau, các thiết bị IoT hiện nay chủ yếu kết nối đến một máy chủ trung tâm do hãng sản xuất một nhà phát triển nào đó quản lý. Cách này cũng vẫn ổn thôi, những thiết bị vẫn hoàn toàn nói chuyện được

với nhau thông qua chức năng phiên dịch của máy chủ rồi. Thế nhưng mọi chuyện không đơn giản như thế, cứ mỗi một mạng lưới như thế tạo thành một subnetwork riêng, và buồn thay các máy móc nằm trong subnetwork này không thể giao tiếp tốt với subnetwork khác.

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

Một số trong những vấn đề nói trên chỉ đơn giản là vấn đề về kiến trúc mạng, về kết nối mà các thiết bị sẽ liên lạc với nhau (Wifi, Bluetooth, NFC,...). Những thứ này thì tương đối dễ khắc phục với công nghệ không dây ngày nay. Còn với các vấn đề về giao thức thì phức tạp hơn rất nhiều, nó chính là vật cản lớn và trực tiếp trên còn đường phát triển của Internet of Things.

Có quá nhiều "ngôn ngữ địa phương"

Bây giờ giả sử như các nhà sản xuất xe ô tô nhận thấy rằng họ cần một giao thức chung để xe của nhiều hãng có thể trao đổi dữ liệu cho nhau và họ đã phát triển thành công giao thức đó. Thế nhưng vấn đề vẫn chưa được giải quyết. Nếu các trạm thu phí đường bộ, các trạm bơm xăng muốn giao tiếp với xe thì sao? Mỗi một loại thiết bị lại sử dụng một "ngôn ngữ địa phương" riêng thì mục đích của IoT vẫn chưa đạt được đến mức tối đa. Đồng ý rằng chúng ta vẫn có thể có một trạm kiểm soát trung tâm, thế nhưng các thiết bị vẫn chưa thật sự nói chuyện được với nhau.

Tiền và chi phí

Cách duy nhất để các thiết bị IoT có thể thật sự giao tiếp được với nhau đó là khi có một động lực kinh tế đủ mạnh khiến các nhà sản xuất đồng ý chia sẻ quyền

điều khiển cũng như dữ liệu mà các thiết bị của họ thu thập được. Hiện tại, các động lực này không nhiều. Có thể xét đến ví dụ sau: một công ty thu gom rác muốn kiểm tra xem các thùng rác có đầy hay chưa. Khi đó, họ phải gặp nhà sản xuất thùng rác, đảm bảo rằng họ có thể truy cập vào hệ thống quản lý của từng thùng một. Điều đó khiến chi phí bị đội lên, và công ty thu gom rác có thể đơn giản chọn giải pháp cho một người chạy xe kiểm tra từng thùng một.

1.5. Triển vọng và nhận định.

Nếu xu hướng hiện nay tiếp tục, dữ liệu được các thiết bị gửi và nhận sẽ nằm trong các "hầm chứa" mang tính chất tập trung (centralized silo). Các công ty, nhà sản xuất có thể kết nối đến các hầm này để thu thập dữ liệu, từ đó tạo ra các bộ giao thức của riêng mình. Tuy nhiên, nhược điểm của mô hình này đó là dữ liệu sẽ trở nên khó chia sẻ hơn bởi người ta cứ phải tạo ra các đường giao tiếp mới giữa các silo. Dữ liệu sẽ phải di chuyển xa hơn và làm chậm tốc độ kết nối. Chưa kể đến các nguy cơ bảo mật và nguy cơ về quyền riêng tư của người dùng nữa.

Trái ngược với hướng đi trên, nếu như các nhà sản xuất có thể thống nhất được các bộ giao tiếp chung thì sẽ tạo ra các "Internet của các ốc đảo" (Internet of Islands). Thiết bị trong một căn phòng có thể giao tiếp với nhau, giao tiếp với các máy móc khác trong nhà và thậm chí là cả... nhà hàng xóm. Dữ liệu sẽ được phân bố trong một khu vực hẹp hơn nên đảm bảo các vấn đề bảo mật, đồng thời tăng tốc độ hoạt động. Dữ liệu cũng nhờ đó mà linh hoạt hơn, các thiết bị có thể phản hồi nhanh hơn. Bên cạnh đó, một khi các thiết bị có thể nói chuyện tốt với nhau, một hệ thống tự động hóa có thể bắt đầu học hỏi những gì đang diễn ra ở thế giới xung quanh, từ đó đưa ra hành động đúng ý muốn của người dùng.

Chương 2: Kiến trúc tham chiếu của IoT

2.1. Tổng quan

Kiến trúc tham chiếu IoT (IoT RA) là kiến trúc ở mức hệ thống mang tính khái quát chung, tổng quát cho các hệ thống IoT có sử dụng chung các miền. IoT RA cung cấp một điểm khởi đầu nhất quán để phát triển và triển khai các giải pháp kiến trúc cho các hệ thống IoT sao cho tất cả các hệ thống được tạo ra đều có các điểm chung như sau:

- Nhất quán về sự tổ hợp thành phần cũng như các mảng thiết kế hệ thống;

- Giảm chi phí bằng cách tận dụng tối đa việc tái sử dụng các dịch vụ, sản phẩm, dữ liệu, các định nghĩa, vv;

- Giảm thời gian bằng cách bắt đầu với IoT RA hiện tại và có thể điều chỉnh cho kiến trúc của một hệ thống IoT mục tiêu;

- Giảm nguy cơ bằng cách: Kết hợp các khả năng toàn cầu cần thiết; Tận dụng các bài học kinh nghiệm và chuyên môn liên quan được nhúng trong IoT RA.

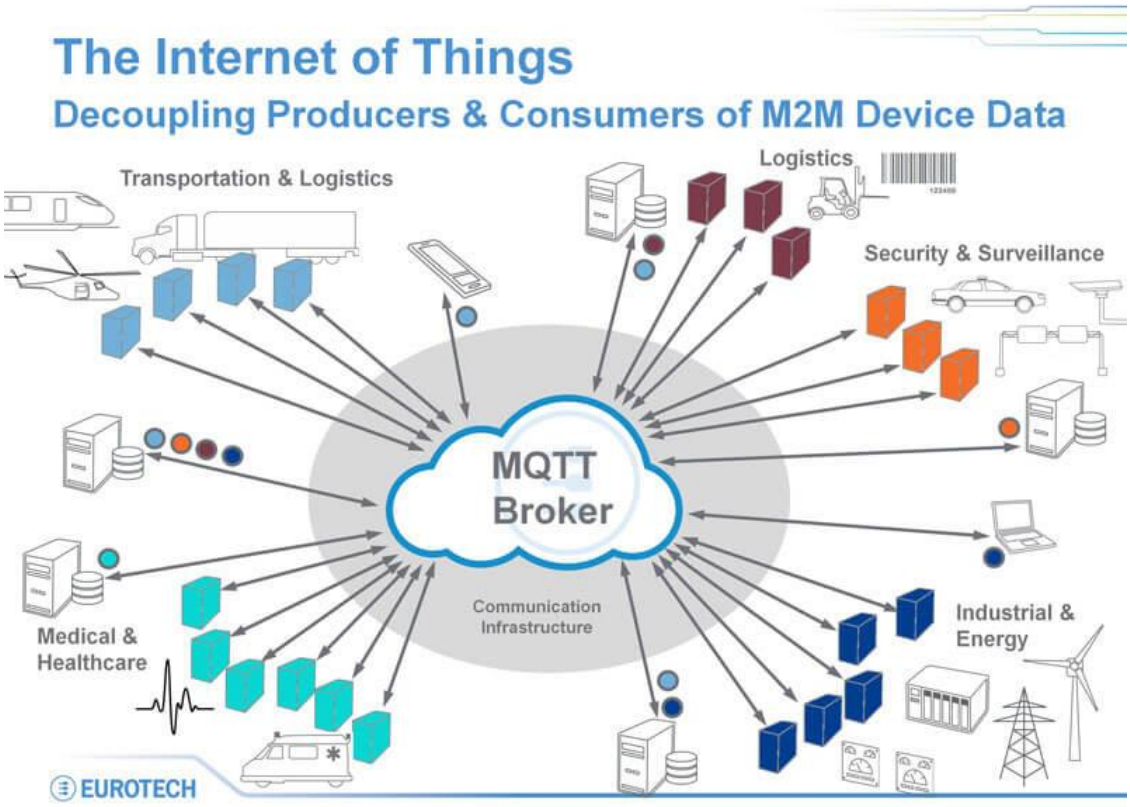
2.2. Giao thức dùng để kết nối trong IoT

MQTT (Message Queue Telemetry Transport)

MQTT là một giao thức mã nguồn mở để truyền các messages giữa nhiều Client (Publisher và Subscriber) thông qua một Broker trung gian, được thiết kế để đơn giản và dễ dàng triển khai. Kiến trúc MQTT dựa trên Broker trung gian và sử dụng kết nối TCP long-lived từ các Client đến Broker.

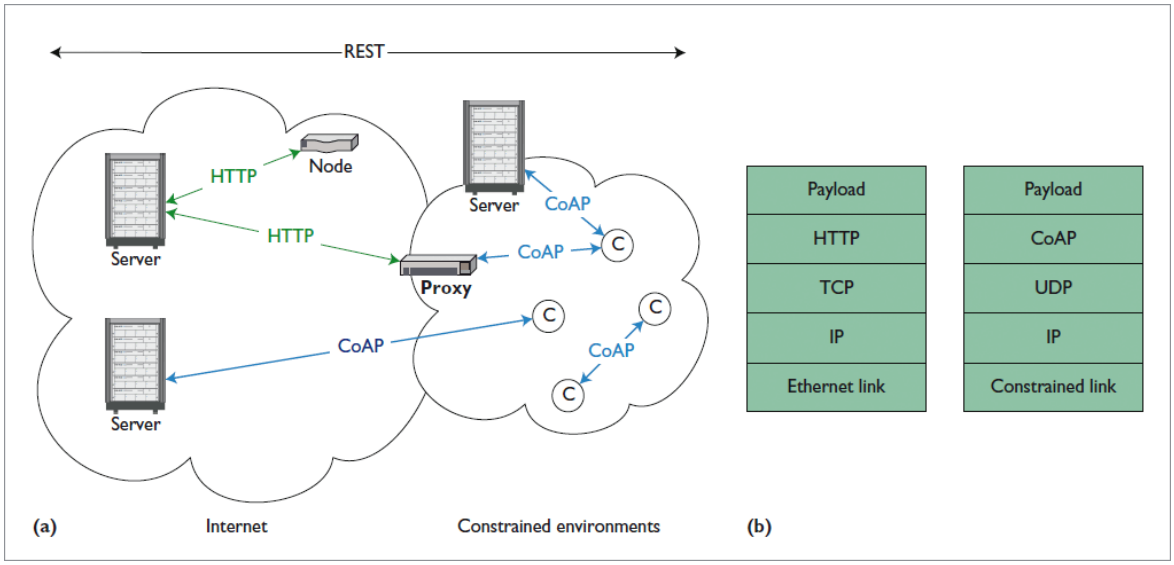
MQTT hỗ trợ tổ chức hệ thống theo các Topics có tính phân cấp, như một hệ thống tập tin (vd: /Home/kitchen/humidity), cung cấp nhiều lựa chọn điều khiển và QoS (Quality of Service).

MQTT là một giao thức khá nhẹ nên có thể được sử dụng cho truyền thông 2 chiều thông qua các mạng có độ trễ cao và độ tin cậy thấp, nó cũng tương thích với các thiết bị tiêu thụ điện năng thấp.



Hình 1.3. Giao thức MQTT

CoAP (Constrained Applications Protocol)



Hình 1.4. Ví dụ về mô hình sử dụng giao thức CoAP và HTTP

CoAP là một giao thức truyền tải tài liệu theo mô hình client/server dự trên internet tương tự như giao thức HTTP nhưng được thiết kế cho các thiết bị ràng buộc.

Giao thức này hỗ trợ một giao thức one-to-one để chuyển đổi trạng thái thông tin giữa client và server.

CoAP sử dụng UDP (User Datagram Protocol), không hỗ trợ TCP, ngoài ra còn hỗ trợ địa chỉ broadcast và multicast, truyền thông CoAP thông qua các datagram phi kết nối (connectionless) có thể được sử dụng trên các giao thức truyền thông dựa trên các gói.

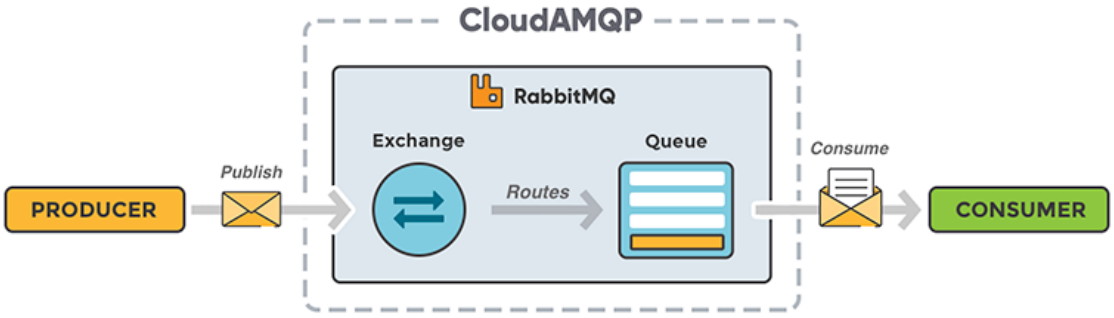
UDP có thể dễ dàng triển khai trên các vi điều khiển hơn TCP nhưng các công cụ bảo mật như SSL/TSL không có sẵn, tuy nhiên ta có thể sử dụng Datagram Transport Layer Security (DTLS) để thay thế.

AMQP (Advanced Message Queue Protocol)

AMQP là một giao thức làm trung gian cho các gói tin trên lớp ứng dụng với mục đích thay thế các hệ thống truyền tin độc quyền và không tương thích. Các tính năng chính của AMQP là định hướng message, hàng đợi, định tuyến (bao gồm point-to-point và publish-subscribe) có độ tin cậy và bảo mật cao. Các hoạt động sẽ được thực hiện thông qua broker, nó cung cấp khả năng điều khiển luồng (Flow Control).

Một trong các Message Broker phổ biến là RabbitMQ, được lập trình bằng ngôn ngữ Erlang, RabbitMQ cung cấp cho lập trình viên một phương tiện trung gian để giao tiếp giữa nhiều thành phần trong một hệ thống lớn.

Không giống như các giao thức khác, AMQP là một giao thức có dây (wire-protocol), có khả năng diễn tả các message phù hợp với định dạng dữ liệu, có thể triển khai với rất nhiều loại ngôn ngữ lập trình.

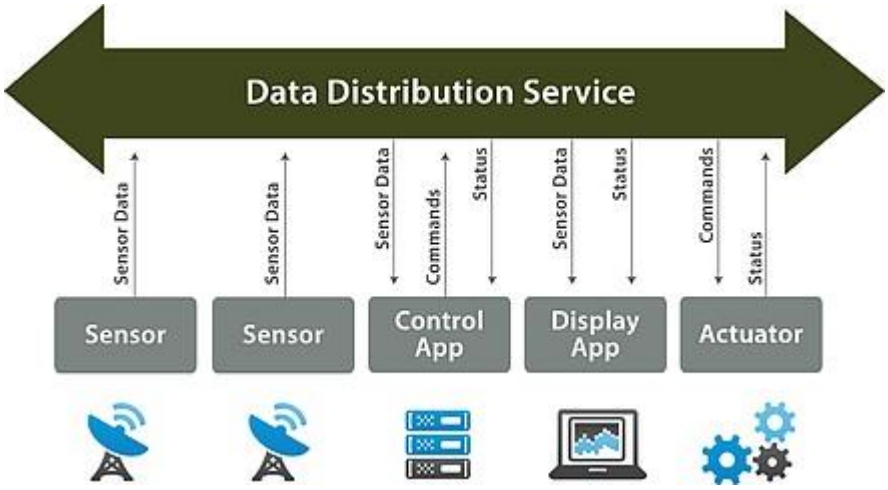


Hình 1.5. Giao thức AMQP

DDS (Data Distribution Service)

DDS là một ngôn ngữ trung gian dựa vào dữ liệu tập trung sử dụng để cho phép khả năng mở rộng, thời gian thực, độ tin cậy cao và trao đổi dữ liệu tương tác.

Đây là một giao thức phi tập trung (broker-less) với truyền thông ngang hàng trực tiếp theo kiểu peer-to-peer giữa các publishers và subscribers và được thiết kế để trở thành một ngôn ngữ và hệ điều hành độc lập. DDS gửi và nhận dữ liệu, sự kiện, và thông tin lệnh trên UDP nhưng cũng có thể chạy trên các giao thức truyền tải khác như IP Multicast, TCP / IP, bộ nhớ chia sẻ,… DDS hỗ trợ các kết nối được quản lý many-to-many theo thời gian thực và ngoài ra còn hỗ trợ dò tìm tự động (automatic discovery). Các ứng dụng sử dụng DDS cho truyền thông được tách riêng và không yêu cầu sự can thiệp từ các ứng dụng của người dùng, có thể đơn giản hóa việc lập trình mạng phức tạp. Các tham số QoS được sử dụng để xác định các cơ chế tự dò tìm của nó được thiết lập một lần.



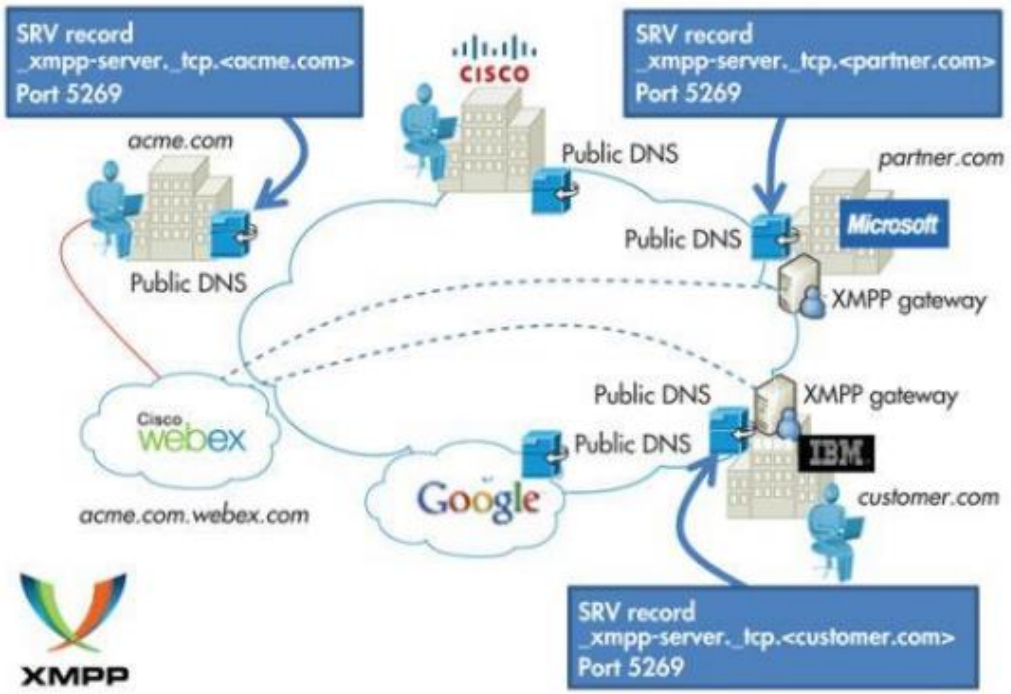
Hình 1.6. Giao thức DDS

XMPP (Extensible Messaging và Presence Protocol)

XMPP (trước đây gọi là "Jabber") là giao thức truyền thông dùng cho định hướng tin nhắn trung gian dựa trên ngôn ngữ XML.

XMPP là mô hình phân quyền client-server phi tập trung, được sử dụng cho các ứng dụng nhắn tin văn bản. Có thể nói XMPP gần như là thời gian thực và có thể mở rộng đến hàng trăm hàng nghìn nút. Dữ liệu nhị phân phải được mã hóa base64 trước

khi nó được truyền đi trong băng tần. XMPP tương tự như MQTT, có thể chạy trên nền tảng TCP.



Hình 1.7. ví dụ về XMPP

2.3. Các yêu cầu của kiến trúc tham chiếu cho IoT

Yêu cầu chung là kiến trúc tham chiếu cho IoT phải trung lập với nhà sản xuất, không phụ thuộc vào công nghệ cụ thể nào. Có những yêu cầu khá đặc thù cho thiết bị IoT và môi trường hỗ trợ nhưng cũng có những yêu cầu xuất phát từ quá trình sản xuất và sử dụng. Có thể tóm tắt các yêu cầu của kiến trúc tham chiếu cho IoT trong 5 nhóm sau:

Kết nối và giao tiếp

Các giao thức hiện tại như HTTP có vị trí rất quan trọng trong nhiều thiết bị. Thậm chí ngay cả bộ điều khiển 8 bit cũng có thể tạo ra lệnh GET, POST và giao thức HTTP tạo ra kết nối đồng nhất quan trọng. Tuy nhiên, phần đầu của HTTP và một số giao thức Internet truyền thống chưa phù hợp cho giao tiếp IoT vì hai lý do. Thứ nhất, kích cỡ bộ nhớ của chương trình quá lớn so với những thiết bị nhỏ. Vấn đề thứ hai còn lớn hơn: HTTP yêu cầu công suất tương đối cao so với thiết bị IoT nhỏ. Do vậy, cần có một giao thức nhị phân đơn giản, nhỏ gọn và có khả năng đi qua tường

lửa. Đối với thiết bị IoT kết nối Internet qua gateway cần 2 giao thức: một giao thức kết nối gateway còn giao thức khác kết nối từ gateway với Internet. Cuối cùng, kiến trúc phải hỗ trợ giao thức truyền thông (communications) và bắc cầu. Ví dụ, giao thức nhị phân dùng cho thiết bị IoT phải cho phép API (dựa trên HTTP) điều khiển được thiết bị khi kết nối với bên thứ ba.

Quản lý thiết bị

Hiện nay, nhiều thiết bị IoT chưa được quản lý ở chế độ tích cực (active) trong khi chế độ quản lý này đã phổ biến và ngày càng trở nên quan trọng trong PC, điện thoại di động và các thiết bị khác. Như vậy, chế độ tích cực là yêu cầu cần có trong quản lý thiết bị IoT, cụ thể bao gồm những tính năng sau:

- Khả năng ngắt kết nối với thiết bị hỏng hoặc bị đánh cắp.

- Khả năng cập nhật phần mềm trên thiết bị.

- Cập nhật các thông số bảo mật.

- Khả năng bật/tắt từ xa một số tính năng phần cứng.

- Xác định vị trí thiết bị mất cắp.

- Xóa dữ liệu bảo mật khỏi thiết bị mất cắp.

- Cấu hình lại từ xa các thông số của Wi-Fi, GPRS hoặc các mạng khác.

Thu thập, phân tích và khởi động dữ liệu

Một số thiết bị IoT có giao diện UI nhưng nói chung hầu hết các thiết bị IoT tập trung vào trang bị 1-2 bộ cảm biến có thể kèm theo (hoặc không) 1 bộ kích động (actuator) hoặc nhiều hơn. Yêu cầu của hệ thống là thu thập dữ liệu từ rất nhiều thiết bị, lưu trữ, phân tích và ra quyết định hành động dựa trên kết quả phân tích gần như theo thời gian thực. Kiến trúc tham chiếu được thiết kế để quản lý số lượng lớn thiết bị. Nếu những thiết bị đó tạo ra dòng dữ liệu ổn định, liên tục thì sẽ có lượng dữ liệu khá lớn. Yêu cầu là hệ thống lưu trữ cần có tính khả mở cao, xử lý được khối lượng dữ liệu lớn và đa dạng.

Tính khả mở (scalability)

Bất kỳ kiến trúc phía máy chủ (server-side) nào đều cần tính khả mở cao, có thể hỗ trợ hàng triệu thiết bị liên tục gửi, nhận và hoạt động dựa trên dữ liệu. Tuy nhiên, "kiến trúc khả mở cao" kéo theo giá thành cao cả về phần cứng, phần mềm và độ phức tạp. Yêu cầu quan trọng đối với kiến trúc này là phải hỗ trợ tính mở từ mức triển khai nhỏ cho đến số lượng thiết bị rất lớn. Tính năng có thể mở co giãn (elastic) và có thể triển khai dạng cloud như dịch vụ Amazon EC2 là rất cần thiết.

An toàn bảo mật

Đây là yêu cầu quan trọng nhất. Các thiết bị IoT thường thu thập dữ liệu có tính cá nhân cao và đưa lên Internet (và ngược lại). Điều đó dẫn đến hai loại nguy cơ mất an toàn:

- Những nguy cơ gắn với bản chất mất an toàn thông tin của mạng Internet mà nhà thiết kế/sản xuất thiết bị IoT không nhận thức được. Đó có thể chỉ đơn giản là việc mở các cổng trên thiết bị. Ví dụ, tủ lạnh kết nối Internet có server dùng giao thức SMTP không an toàn nên có thể gửi tin rác (spam).

- Những nguy cơ riêng từ chính bản thân thiết bị IoT, nhất là phần cứng. Ví dụ, nhiều thiết bị IoT quá nhỏ, không hỗ trợ cơ chế mã hóa phi đối xứng phù hợp nên không bảo đảm an toàn.

Có hai vấn đề quan trọng, đặc thù của bảo mật IoT là quản lý định danh và truy nhập. Quản lý định danh cho thiết bị IoT và giao tiếp máy-máy (M2M) bằng cách dùng cặp "tên/mật khẩu" là sai lầm thông dụng. Giải pháp lý tưởng là sử dụng token cấp theo chuẩn của OAuth/OAuth2. Đối với quản lý truy nhập, sai lầm thường thấy là sử dụng luật truy nhập đã lập trình cứng trong mã nguồn phía client và server. Cách tiếp cận tốt nhất hiện nay là sử dụng tiêu chuẩn XACML. Cách tiếp cận này không dùng mạch logic lập trình sẵn mà đưa vào chính sách ra quyết định cho phép truy nhập. Ưu điểm của cách này là:

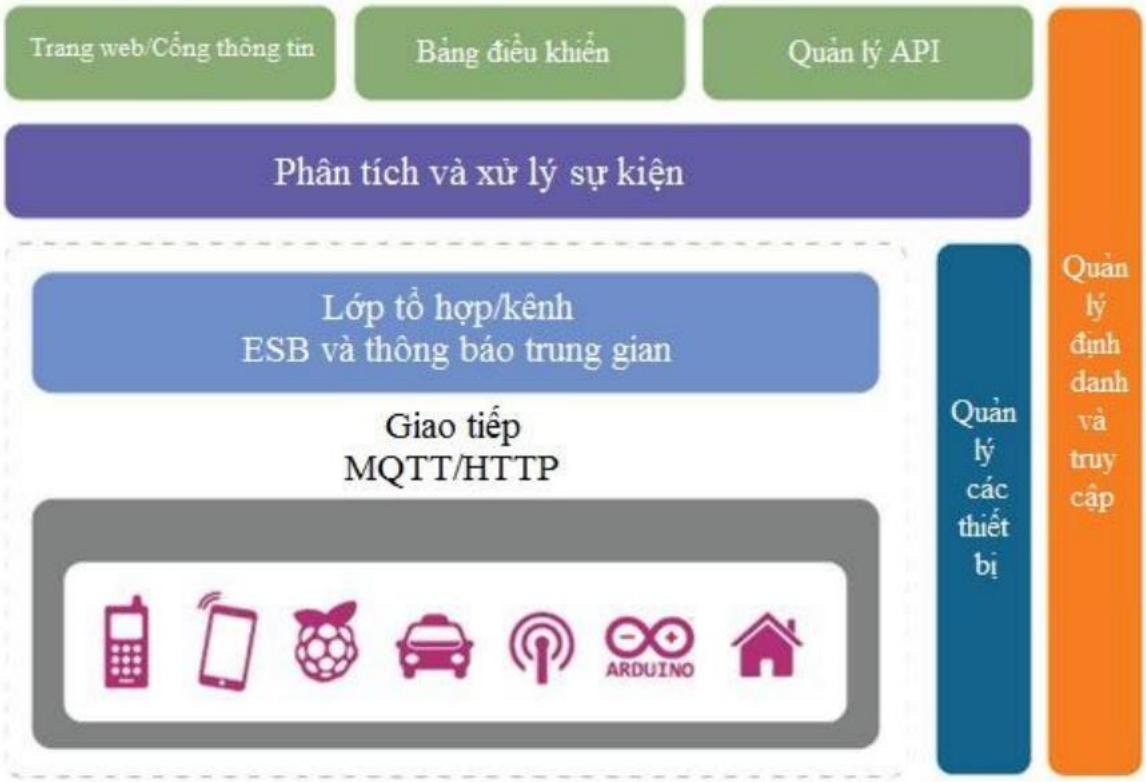
- Đưa ra quyết định phù hợp hơn.

- Dựa trên bối cảnh cụ thể như: vị trí, hoặc mạng nào đang được sử dụng, hoặc thời gian.

- Có thể phân tích và thẩm định yêu cầu truy nhập.

- Có thể cập nhật, thay đổi chính sách truy nhập mà không cần lập trình lại hay chỉnh sửa thiết bị.

2.4. Mô hình tham chiếu



Hình 1.8. Mô hình tham chiếu cho IoT của WSO2

Mô hình tham chiếu IoT bao gồm 5 lớp xếp chồng. Mỗi lớp có một chức năng riêng, có thể minh họa bằng những công nghệ cụ thể. Có 2 lớp theo chiều dọc là quản lý thiết bị và quản lý định danh & truy nhập.

- Lớp Thiết bị (Devices)

Các thiết bị IoT phải có giao thức truyền thông trực tiếp (Arduino, Raspberry Pi, Intel Galileo qua Ethernet hoặc Wi-Fi) hoặc gián tiếp kết nối được với Internet (ZigBee, Bluetooth hoặc Bluetooth công suất thấp qua điện thoại di động,...).

Mỗi thiết bị cần có định danh thuộc một trong các loại: định danh duy nhất (UUID) ghi sẵn trong phần cứng (thường là một phần của SoC hoặc chip thứ cấp), UUID gửi qua hệ thống vô tuyến phụ (ví dụ: định danh Bluetooth, địa chỉ Wi-Fi

MAC), token OAuth2 Refresh/Bearer (có thể là bổ sung cho các loại khác), định danh lưu trong bộ nhớ chỉ đọc như EEPROM.

Các chuyên gia khuyến nghị, mỗi thiết bị IoT nên có một UUID (tốt nhất lưu cố định trong phần cứng) và một token OAuth2 Refresh/Bearer lưu trong EEPROM. OAuth2 token có mục đích tạo ra một token định danh tách biệt với số định danh cố định ghi trong mỗi thiết bị. Bearer token được dùng ban đầu để gửi đến bất kỳ server hay dịch vụ nào cần định danh. Bearer token có thời gian sống ngắn hơn Refresh token. Nếu Bearer token hết hạn, Refresh token được gửi đến lớp Định danh để tạo ra bản cập nhật của Bearer token.

Lớp Truyền thông (Communications)

Lớp truyền thông hỗ trợ kết nối các thiết bị. Nhiều giao thức có thể sử dụng trong lớp này như HTTP/HTTPS, MQTT 3.1/3.1.1, CoAP (Constrained Application Protocol). Trong đó, HTTP là giao thức lâu đời và phổ biến nhất nên có nhiều thư viện hỗ trợ. Vì đó là giao thức dựa trên ký tự đơn giản nên nhiều thiết bị nhỏ như bộ điều khiển 8 bit đều có thể hỗ trợ HTTP. Các thiết bị 32 bit lớn hơn có thể sử dụng các thư viện HTTP client đầy đủ.

Có một số giao thức được tối ưu riêng cho IoT, trong đó nổi bật nhất là 2 giao thức MQTT và CoAP. MQTT được phát minh vào năm 1999 để giải quyết các vấn đề của hệ thống nhúng và SCADA. Giao thức MQTT có phần mào đầu nhỏ (chỉ 2 byte/message), chạy được trên nền TCP và có khả năng chịu được môi trường mạng thường bị gián đoạn và suy hao cao. Ủy ban kỹ thuật của tổ chức tiêu chuẩn OASIS đang xem xét chuẩn hóa phiên bản MQTT hiện nay (3.1.1). Giao thức CoAP do tổ chức tiêu chuẩn IETF xem xét phát triển trên cơ sở HTTP nhưng dựa trên mã nhị phân chứ không phải ký tự nên nhỏ gọn hơn, có thể chạy trên nền UDP.

Lớp Hợp nhất/Bus (Aggregation/ Bus)

Đây là lớp quan trọng để hợp nhất và chuyển đổi các thông điệp (message broker hay middleware) truyền thông với 3 chức năng sau:

- Hỗ trợ máy chủ HTTP và/hoặc chức năng chuyển đổi MQTT để giao tiếp

- Hợp nhất nội dung truyền từ các thiết bị khác nhau và định tuyến truyền thông tới một thiết bị cụ thể (có thể qua gateway).

- Bắc cầu và chuyển đổi giữa 2 giao thức khác nhau, ví dụ chuyển đổi API dựa trên HTTP ở lớp trên vào thông điệp MQTT đến thiết bị.

Lớp Bus cũng có thể cung cấp một số tính năng tương quan (correlation) và ánh xạ đơn giản từ các mô hình tương quan khác nhau (nghĩa là ánh xạ số định danh thiết bị sang số định danh của người sở hữu thiết bị và ngược lại).

Cuối cùng, lớp Hợp nhất/Bus cần thực hiện 2 nhiệm vụ an toàn bảo mật là Máy chủ tài nguyên OAuth2 (thẩm định Bearer token và các truy nhập tài nguyên liên quan) và Điểm tăng cường chính sách (PEP) đối với truy nhập dựa trên chính sách. Trong mô hình ở Hình 2, lớp Bus yêu cầu lớp Quản lý truy nhập và Định danh thẩm định các yêu cầu truy nhập. Lớp Quản lý truy nhập và Định danh đóng vai trò như Điểm quyết định chính sách (PDP) trong quá trình này. Sau đó, lớp Bus thực hiện theo kết quả do PDP mang đến, nghĩa là cho phép hoặc không cho phép truy nhập tài nguyên.

Lớp Xử lý Sự kiện và Phân tích (Event Processing and Analytics)

Lớp này xử lý các sự kiện từ lớp Bus chuyển lên. Yêu cầu chủ yếu ở đây là khả năng lưu trữ dữ liệu vào cơ sở dữ liệu. Mô hình truyền thống sẽ viết một ứng dụng phía máy chủ (ví dụ, JAX-RS). Tuy nhiên, có một số cách tiếp cận khác linh hoạt hơn. Thứ nhất là sử dụng các nền tảng phân tích dữ liệu lớn. Đó là nền tảng dựa trên cloud khả mở hỗ trợ các công nghệ như Apache Hadoop để cung cấp những phân tích map-reduce (quy trình xử lý dữ liệu siêu lớn) đối với tập hợp dữ liệu đến từ các thiết bị. Cách tiếp cận thứ hai là hỗ trợ phương thức Xử lý Sự kiện Phức tạp (Complex Processing Event) để thực hiện các hoạt động gần như theo thời gian thực và ra quyết định hành động dựa theo kết quả phân tích dữ liệu từ các thiết bị chuyển đến.

Lớp Truyền thông ngoài (External Communication)

Lớp nào tạo ra giao diện giúp quản lý các thiết bị IoT như: web/portal, dashboard (bảng hiển thị tổng hợp) hoặc hệ thống quản lý API. Với web/portal, kiến trúc cần hỗ trợ các công nghệ Web phía máy chủ như Java Servlets/JSP, PHP, Python,

Ruby ... Web server dựa trên Java phổ biến nhất là Apache Tomcat. Dashboard là hệ thống tái sử dụng tập trung vào việc trình bày đồ thị mô tả dữ liệu đến từ các thiết bị và lớp xử lý sự kiện. Lớp quản lý API có 3 chức năng. Thứ nhất là cung cấp portal tập trung vào hỗ trợ lập trình viên tác nghiệp (chứ không phải là người sử dụng như portal thông thường) và quản lý các phiên bản của API được xuất bản. Thứ hai là đóng vai trò gateway quản lý truy nhập vào các API, kiểm tra việc điều khiển truy nhập (đối với yêu cầu từ bên ngoài), điều tiết sử dụng dựa trên chính sách, định tuyến và cân bằng tải. Cuối cùng là chức năng gateway đẩy dữ liệu vào lớp phân tích để lưu trữ và xử lý, giúp hiểu được các API đã được sử dụng như thế nào.

Lớp Quản lý Thiết bị (Device Management)

Trong lớp Quản lý thiết bị, hệ thống phía máy chủ DM (Device Manager) giao tiếp với các thiết bị thông qua các giao thức khác nhau và điều khiển phần mềm của từng thiết bị hoặc một nhóm thiết bị (có thể khóa hoặc xóa dữ liệu trên thiết bị khi cần), quản lý định danh các thiết bị và ánh xạ vào chủ nhân các thiết bị đó. DM phải phối hợp với lớp Quản lý Định danh và Truy nhập để quản lý việc điều khiển truy nhập vào thiết bị (những người có quyền truy nhập vào thiết bị ngoài chủ nhân, quyền điều khiển của chủ nhân thiết bị so với người quản trị...).

Lớp Quản lý Định danh và Truy nhập (Identity and Access Management)

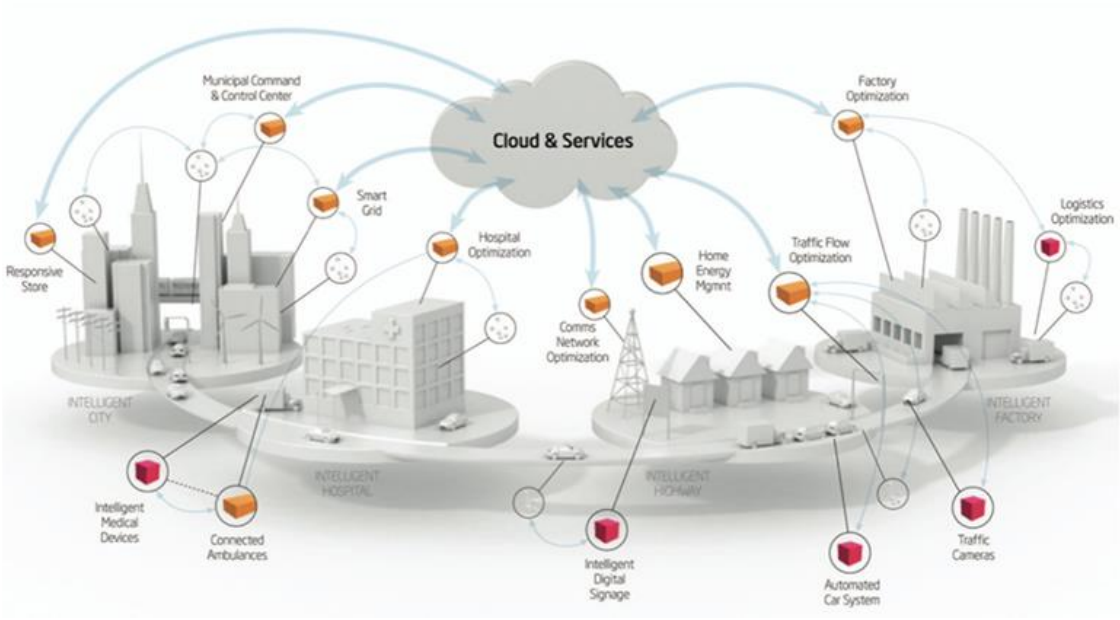
Lớp này cần cung cấp các dịch vụ: Phát hành và thẩm định Oauth2 token; Các dịch vụ định danh khác, gồm cả SAML2 SSO và OpenID Connect; XACML PDP; Danh bạ cho người dùng (ví dụ: LDAP); Quản lý chính sách điều khiển truy nhập (PCP).

2.5. Hạ tầng mạng và điện toán đám mây

- Cơ sở hạ tầng kết nối: Internet là một hệ thống toàn cầu của nhiều mạng IP được kết nối với nhau và liên kết với hệ thống máy tính. Cơ sở hạ tầng mạng này bao gồm thiết bị định tuyến, trạm kết nối, thiết bị tổng hợp, thiếp bị lặp và nhiều thiết bị khác có thể kiểm soát lưu lượng dữ liệu lưu thông và cũng được kết nối đến mạng lưới viễn thông và cáp – được triển khai bởi các nhà cung cấp dịch vụ.

- Trung tâm dữ liệu/ hạ tầng điện toán đám mây: Các trung tâm dữ liệu và hạ tầng điện toán đám mây bao gồm một hệ thống lớn các máy chủ, hệ thống lưu trữ và mạng ảo hóa được kết nối. Điện toán đám mây (Cloud computing) có thể hiểu một cách đơn giản là: các nguồn điện toán khổng lồ như phần mềm, dịch vụ… sẽ nằm tại các máy chủ ảo (đám mây) trên Internet thay vì trong máy tính gia đình và văn phòng (trên mặt đất) để mọi người kết nối và sử dụng mỗi khi họ cần. Với các dịch vụ sẵn có trên Internet, doanh nghiệp không phải mua và duy trì hàng trăm, thậm chí hàng nghìn máy tính cũng như phần mềm. Họ chỉ cần tập trung sản xuất bởi đã có người khác lo cơ sở hạ tầng và công nghệ thay họ. Bạn có thể truy cập đến bất kỳ tài nguyên nào tồn tại trong "đám mây (cloud)" tại bất kỳ thời điểm nào và từ bất kỳ đâu thông qua hệ thống Internet.

50 tỉ thiết bị được kết nối vào năm 2020; đây là một con số khổng lồ và sẽ tạo ra một lượng dữ liệu vô cùng lớn được thu thập từ các thiết bị trong mạng IoT. Điều này đòi hỏi phải có nền tảng đủ mạnh và linh hoạt để phân tích, lưu trữ; đồng thời có khả năng phát triển các ứng dụng tương hỗ với IoT. Những yêu cầu trên sẽ hoàn toàn được đáp ứng, nếu chúng ta triển khai IoT kết hợp với hạ tầng điện toán đám mây (Cloud Computing). Điện toán đám mây có thể cung cấp cơ sở hạ tầng ảo hóa cho các ứng dụng, thiết bị giám sát, thiết bị lưu trữ, công cụ phân tích, nền tảng hình ảnh... Mô hình dựa trên tiện ích cung cấp bởi điện toán đám mây sẽ cho phép các doanh nghiệp và người dùng truy cập các ứng dụng theo yêu cầu mọi lúc, mọi nơi và bất cứ đâu. Một số nền tảng đám mây IoT lớn và phổ biến nhất hiện nay bao gồm Amazon Web Services, GE Predix, Google Cloud IoT, Microsoft Azure IoT Suite, IBM Watson và Salesforce IoT Cloud.



Hình 1.9. Điện toán đám mây

2.6. Công nghệ truyền thông trong IoT

1. Bluetooth

Một công nghệ giao tiếp truyền thông trong khoảng cách ngắn vô cùng quan trọng, đó là Bluetooth. Hiện nay, bluetooth xuất hiện hầu hết ở các thiết bị như máy tính, điện thoại/ smartphone,....và nó được dự kiến là chìa khóa cho các sản phẩm IoT đặc biệt, cho phép giao tiếp thiết bị với các smartphone - một "thế lực hùng hậu" hiện nay.

Hiện nay, BLE - Bluetooth Low Energy - hoặc Bluethooth Smart là một giao thức được sử dụng đáng kể cho các ứng dụng IoT. Quan trọng hơn, cùng với một khoảng cách truyền tương tự như Bluetooth, BLE được thiết kế để tiêu thụ công suất ít hơn rất nhiều.

Tuy nhiên, BLE không thực sự được thiết kế cho các ứng dụng dùng để truyền file và sẽ phù hợp hơn cho khối dữ liệu nhỏ. Nó có một lợi thế vô cùng lớn trong bối cảnh hiện nay, smartphone đang là thiết bị không thể thiếu được của mỗi người. Theo Bluetooth SIG, hiện có hơn 90% điện thoại smartphone được nhúng Bluetooth, bao gồm các hệ điều hành IOS, Android và Window, và dự kiến đến năm 2018 sẽ là " Smart Ready".

2. Zigbee

Zigbee, giống như Bluetooth, là một loại truyền thông trong khoảng cách ngắn, hiện được sử dụng với số lượng lớn và thường được sử dụng trong công nghiệp. Điển hình, Zigbee Pro và Zigbee remote control (RF4CE) được thiết kế trên nền tảng giao thức IEEE802.15.4 - là một chuẩn giao thức truyền thông vật lý trong công nghiệp hoạt động ở 2.4Ghz thường được sử dụng trong các ứng dụng khoảng cách ngắn và dữ liệu truyền tin ít nhưng thường xuyên, được đánh giá phù hợp với các ứng dụng trong smarthome hoặc trong một khu vực đô thị/khu chung cư.

Zigbee / RF4CE có một lợi thế đáng kể trong các hệ thống phức tạp cần các điều kiện: tiêu thụ công suất thấp, tính bảo mật cao, khả năng mở rộng số lượng các node cao...ví dụ như yêu cầu của các ứng dụng M2M và IoT là điển hình. Phiên bản mới nhất của Zigbee là 3.0, trong đó điểm nổi bật là sự hợp nhất của các tiêu chuẩn Zigbee khác nhau thành một tiêu chuẩn duy nhất. Ví dụ, sản phẩm và kit phát triển của Zigbee của TI là CC2538SF53RTQT Zigbee System-On-Chip T và CC2538 Zigbee Development Kit.

3. Z-wave

Tương tự Zigbee, Z-Wave là chuẩn truyền thông không dây trong khoảng cách ngắn và tiêu thụ rất ít năng lượng. Dung lượng truyền tải với tốc độ 100kbit/s, quá đủ cho nhu cầu giao tiếp giữa các thiết bị trong các hệ thống IoT, M2M. Chuẩn kết nối Z-Wave và Zigbee cùng hoạt động với tần số 2.4GHz, và cùng được thiết kế với mức tiêu thụ năng lượng rất ít nên có thể sử dụng với các loại PIN di động.Zwave hoạt động ở tần số thấp hơn so với Zigbee/wifi, dao động trong các dải tần của 900Mhz, tùy theo quy định ở từng khu vực khác nhau.

Ưu điểm của Z-Wave là tiêu thụ năng lượng cực ít và độ mở (open platform) cực cao. Hiện nay, Z-Wave được ứng dụng chủ yếu trong ứng dụng smarthome. Đặc biệt, mỗi thiết bị Z-Wave trong hệ thống là một thiết bị có thể vừa thu và vừa phát sóng nên tính ổn định hệ thống được nâng cao.

Đặc biệt, Z-Wave đã được nhiều nhà sản xuất thiết bị tích hợp vào, đây là một công nghệ đang được chú ý và các nhà sản xuất đang tập trung nhiều hơn vào nó.

4. 6LoWPAN

6LoWPAN là tên viết tắt của IPv6 protocol over low-power wireless PANs (tức là: sử dụng giao thức IPv6 trong các mạng PAN không dây công suất thấp). 6LoWPAN được phát triển bởi hiệp hội đặc trách kỹ thuật Internet IETF (Internet Engineering Task Foce), cho phép truyền dữ liệu qua các giao thức IPv6 và IPv4 trong các mạng không dây công suất thấp với các cấu trúc mạng điểm - điểm (P2P: point to point) và dạng lưới (mesh). Tiêu chuẩn được đặt ra để quy định các đặc điểm của 6LoWPAN - cho phép sử dụng rộng rãi trong các ứng dụng IoT.

Điểm khác của 6LoWPAN so với Zigbee, Bluetooth là: Zigbee hay bluetooth là các giao thức ứng dụng, còn 6LoWPAN là giao thức mạng, cho phép quy định cơ chế đóng gói bản tin và nén header. Đặc biệt, IPv6 là sự kế thừa của IPv4 và cung cấp khoảng 5 x 1028 địa chỉ cho tất cả mọi đối tượng trên thế giới, cho phép mỗi đối tượng là một địa chỉ IP xác định để kết nối với Internet.

Được thiết kế để gửi các bản tin IPv6 qua mạng IEEE802.15.4 và các tiêu chuẩn IP mở rộng như: TCP, UDP, HTTP, COAP, MQTT và Websocket, là các tiêu chuẩn cung cấp nodes end-to-end, cho phép các router kết nối mạng tới các IP.

5. Thread

Thread là một giao thức IP mới, dựa trên nền tảng mạng IPv6 được thiết kế riêng cho mảng tự động hóa trong các tòa nhà và nhà. Nó không phải là một giao thức được yêu thích để ứng dụng trong các bài toán IoT như Zigbee hay Bluetooth.

Được ra mắt vào giữa năm 2014 bởi Theard Group, giao thức Thread dựa trên các tiêu chuẩn khác nhau, bao gồm IEEE802.15.4, IPv6 và 6LoWPAN, và cung cấp một giải pháp dựa trên nền tảng IP cho các ứng dụng IoT. Được thiết kế để làm việc với các sản phẩm chip của Freescale và Silicon Labs (vốn hỗ trợ chuẩn IEÊ802.15.4), đặc biệt có khả năng xử lý lên đến 250 nút với độ xác thực và tính mã hóa cao. Với một bản phần mềm upgrade đơn giản, cho phép người dùng có thể chạy Theard trên các thiết bị hỗ trợ IEEE802.15.4 hiện nay.

6. Wifi

Wifi (là viết tắt từ Wireless Fidelity hay mạng 802.11) là hệ thống mạng không dây sử dụng sóng vô tuyến, cũng giống như điện thoại di đông, truyền hình và radio. Kết nôi Wifi thường là sự lựa chọn hàng đầu của rất nhiều kỹ sư giải pháp bởi tính thông dụng và kinh tế của hệ thống wifi và mạng LAN với mô hình kết nối trong một phạm vi địa lý có giới hạn.

Các sóng vô tuyến sử dụng cho WiFi gần giống với các sóng vô tuyến sử dụng cho thiết bị cầm tay, điện thoại di động và các thiết bị khác. Nó có thể chuyển và nhận sóng vô tuyến, chuyển đổi các mã nhị phân 1 và 0 sang sóng vô tuyến và ngược lại. Tuy nhiên, sóng WiFi có một số khác biệt so với các sóng vô tuyến khác ở chỗ: Chúng truyền và phát tín hiệu ở tần số 2.4 GHz hoặc 5 GHz. Tần số này cao hơn so với các tần số sử dụng cho điện thoại di động, các thiết bị cầm tay và truyền hình. Tần số cao hơn cho phép tín hiệu mang theo nhiều dữ liệu hơn.

Hiện nay, đa số các thiết bị wifi đều tuân theo chuẩn 802.11n, được phát ở tần số 2.4Ghz và đạt tốc độ xử lý tối đa 300Megabit/giây

7. Cellular

Với các ứng dụng IoT/M2M yêu cầu khoảng cách truyền thông dài, hoặc không bị giới hạn bởi khoảng cách địa lý thì việc lựa chọn đường truyền dữ liệu thông qua mạng điện thoại di động GPRS/3G/LTE là một lựa chọn sáng suốt. Tất nhiên, đối với các kỹ sư thiết kế giải pháp,ai cũng hiểu rằng, truyền dữ liệu đi xa thì sẽ tốn năng lượng tương ứng. Và yếu tố tiêu hao năng lượng dễ được chấp nhận trong bài toán này.

Hiện nay, các thiết bị/các điểm đầu cuối trong công nghiệp đều được hỗ trợ tích hợp các cổng giao tiếp vật lý theo chuẩn như: RS232 , RS485, RS422 hay Ethernet. Các phương tiện truyền thông qua mạng di động đều hỗ trợ đầu vào là các cổng Serial hay Ethernet nên việc tích hợp giải pháp truyền thông không dây không còn khó khăn hay bị giới hạn bởi yếu tố khách quan nào khác.

8. NFC

NFC (Near-Field Communications) là công nghệ kết nối không dây trong phạm vi tầm ngắn trong khoảng cách 4 cm. Công nghệ này sử dụng cảm ứng từ trường để thực hiện kết nối giữa các thiết bị (smartphone, tablet, loa, tai nghe …) khi có sự tiếp xúc trực tiếp (chạm).

Khi hai thiết bị đều có kết nối NFC, bạn có thể chạm chúng vào nhau để kích hoạt tính năng này và nhanh chóng truyền tập tin gồm danh bạ, nhạc, hình ảnh, video, ứng dụng hoặc địa chỉ website... Ở các nước phát triển, NFC còn được xem là chiếc ví điện tử khi có thể thanh toán trực tuyến, tiện lợi và nhanh chóng.

Ngoài việc giúp truyền tải dữ liệu như trên thì NFC còn mở rộng với những công dụng ví dụ như bạn đến quán café có một thẻ NFC để trên bàn, trong thẻ này đã cài đặt sẵn wifi, thông tin của quán…lúc này bạn lấy chiếc điện thoại chạm vào NFC này thì máy sẽ bật tất cả tính năng được cài sẵn trong thẻ đó mà không cần phải nhờ gọi nhân viên. Hoặc tiên tiến hơn thì sau này có thể khi mua đồ trong siêu thị lớn thì quẹt NFC của điện thoại để thanh toán tiền luôn.

9. Sigfox

Sigfox là hệ thống giống như mạng di động, sử dụng công nghệ Ultra Band (UNB) để kết nối các thiết bị từ xa. Mục tiêu của công nghệ là sử dụng trong các ứng dụng truyền thông với tốc độ thấp, khoảng cách truyền xa và mức tiêu thụ năng lượng cực thấp. Ngoài ra, nó đòi hỏi yêu cầu về antenna thấp hơn so với mạng di động GSM/CDMA. Sigfox sử dụng các dải tần ISM được sử dụng miễn phí mà không cần phải được cấp phép để truyền dữ liệu.

Ý tưởng ra đời của Sigfox được hình thành từ nhu cầu: Đối với các ứng dụng M2M sử dụng nguồn bằng Pin và chỉ đòi hỏi tốc độ truyền dữ liệu thấp thì phạm vi truyền của Wifi lại quá ngắn, còn với mạng di động thì lại quá đắt đỏ và tốn năng lượng. Với công nghệ UNB, và được thiết kế để chỉ xử lý đường truyền dữ liệu từ 10 đến 1000 bit trên giây, giúp chỉ tiêu thụ mức năng lượng 50 microwatts so với 5000 microwatts của việc dùng mạng điện thoại di động. Hay đơn giản, với một cục pin

2,5Ah thì với công nghệ Sigfox cho phép bạn dùng tới 25 năm thay vì 0,2 năm nếu dùng truyền thông qua mạng điện thoại di động.

10. Neul

Tương tự Sigfox và hoạt động ở băng tần 1Ghz, với mục tiêu cung cấp một mạng không dây có chi phí thấp với các đặc trưng tiêu biểu: độ mở rộng cao, phủ sóng cao và tiêu thụ năng lượng cực thấp. Neul sử dụng chip Iceni, mà trong truyền thông sử dụng "the white space radio" để truy cập vào băng tần UHF chất lượng cao hiện đang có sẵn do sự chuyển đổi từ kỹ thuật ti vi tương tự sang kỹ thuật số. Công nghệ truyền thông được gọi là "Weightless", tức là một công nghệ mạng không dây phủ trên diện rộng, được thiết kế cho các ứng dụng Iot, cạnh tranh trực tiếp với các giải pháp đang có sẵn như GPRS, 3G, CDMA và LTE WAN. Tốc độ truyền dữ liệu có thể dao động từ vài bits trên giây tới 100kbps trên cùng một liên kết, và đặc biệt là với công nghệ này, thiết bị có thể tiêu thụ công suất rất nhỏ, từ 20 tới 30mA từ pin 2xÂ, tức là có thể sử dụng đươc từ 10 đến 15 năm với cục pin.

11. LIFI

LIFI là một công nghệ không dây sử dụng các bóng đèn LED để truyền dữ liệu với tốc độ nhanh hơn Wifi tới 100 lần. Như vậy, với bóng đèn LED với chức năng thắp sáng, giờ có thêm chức năng truyền dữ liệu tốc độ cao. Công ty Velmenni đã có vài dự án thí điểm, trong đó có tạo một không gian mạng ko dây trong văn phòng, sử dụng ánh sáng đèn LED thay vì dùng sóng radio để truyền dữ liệu như của Wi-Fi. CEO của Velmenni, Deepak Solanki, hồi giữa năm 2015 cho rằng công ty hy vọng sẽ mang sản phẩm này đến được với nhiều người sử dụng trong vòng 3-4 năm tới. Công nghệ đột phá này được công ty đặt cho cái tên là Li-Fi, lần đầu được một giáo sư đại học Edinburgh, giáo sư Harald Haas, giới thiệu cách nay 4 năm.

Li-Fi sử dụng dải tần ánh sáng mà mắt người nhìn thấy được để làm phương tiện truyền dữ liệu. Tuy vậy, người dùng không thể sử dụng bất kỳ nguồn ánh sáng đèn điện nào mà phải cần một nguồn sáng riêng để điều biến tín hiệu, tạo thành luồng dữ liệu. Hiện thời, tính năng này chỉ thực hiện được với các bóng đèn LED đạt chuẩn, có tích hợp một chip đặc biệt và có thêm một bộ nhận tín hiệu ánh sáng đặc biệt để có thể giải mã được tín hiệu ánh sáng truyền đi từ đèn LED.

Kỹ thuật điều biến ánh sáng không ảnh hưởng gì đến sức khỏe con người, nhất là về mắt. Giáo sư Haas giới thiệu công nghệ này tại diễn đàn TED Global hồi năm 2011, cho rằng chúng ta thậm chí có thể giảm độ sáng của đèn thật thấp đến mức gần như là tắt, nhưng tín hiệu truyền dữ liệu vẫn hoạt động như thường.

12. LoRa

LoRa là viết tắt của Long Range Radio được nghiên cứu và phát triển bởi Cycleo và sau này được mua lại bởi công ty Semtech năm 2012. Với công nghệ này, chúng ta có thể truyền dữ liệu với khoảng cách lên hàng km mà không cần các mạch khuếch đại công suất; từ đó giúp tiết kiệm năng lượng tiêu thụ khi truyền/nhận dữ liệu. Do đó, LoRa có thể được áp dụng rộng rãi trong các ứng dụng thu thập dữ liệu như sensor network trong đó các sensor node có thể gửi giá trị đo đạc về trung tâm cách xa hàng km và có thể hoạt động với battery trong thời gian dài trước khi cần thay pin.

LoRa sử dụng kỹ thuật điều chế gọi là Chirp Spread Spectrum. Có thể hiểu nôm na nguyên lý này là dữ liệu sẽ được băm bằng các xung cao tần để tạo ra tín hiệu có dãy tần số cao hơn tần số của dữ liệu gốc (cái này gọi là chipped); sau đó tín hiệu cao tần này tiếp tục được mã hoá theo các chuỗi chirp signal (là các tín hiệu hình sin có tần số thay đổi theo thời gian; có 2 loại chirp signal là up-chirp có tần số tăng theo thời gian và down-chirp có tần số giảm theo thời gian; và việc mã hoá theo nguyên tắc bit 1 sẽ sử dụng up-chirp, và bit 0 sẽ sử dụng down-chirp) trước khi truyền ra anten để gửi đi.

Phần II: Xây dựng mô hình IoT

Chương 1: Phân tích đề tài

1.1. Yêu cầu

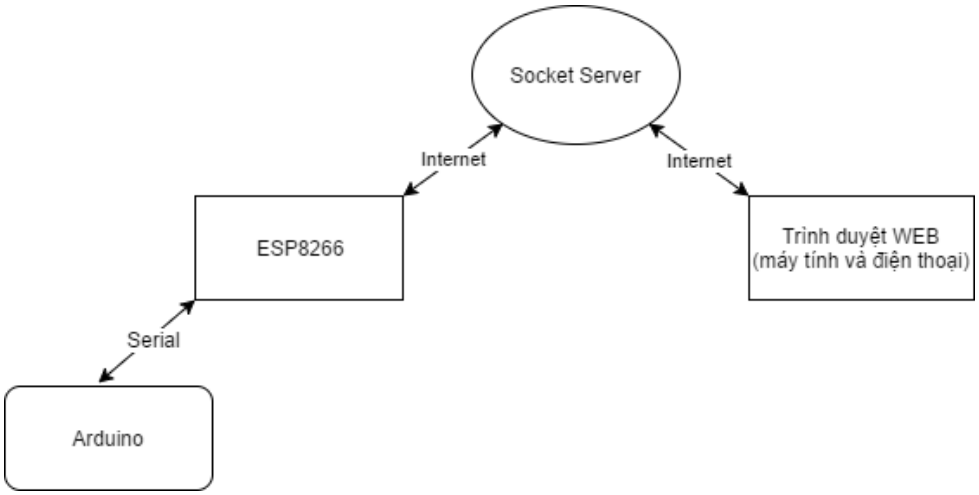
Vấn đề đặt ra ở đây là làm sao để kiểm soát môi trường và điều khiển những chức năng hữu ích tiết kiệm nhân lực. Với một số loài cây việc kiểm soát môi trường sinh thái của nó rất quan trọng trong quá trình cây phát triển. Điều khiển thời gian ra hoa thụ phấn, làm củ, phát triển lá rễ phụ thuộc rất nhiều vào nhiệt độ, độ ẩm, ánh sáng… mà từng loại cây lại có đặc tính riêng đòi hỏi người làm vườn phải nắm bắt để có phương án tính toán cho khu vườn của mình thu được hiệu quả và năng suất cao. Đồng thời với đó là tiết kiệm thời gian nhân lực.

1.2. Giải pháp thiết kế

Để giải quyết yêu cầu của bài toán, mô hình "Tiny Garden" của em có có thực hiện một số chức năng như sau:

- Đo nhiệt độ độ ẩm trong không khí gửi về cho người dùng
- Đo độ ẩm của đất gửi về cho người dùng
- Tưới nước tự động với mức độ ẩm mà người dùng có thể tùy chỉnh
- Điều khiển một số chức năng người dùng có thể tự lắp đặt phù hợp với từng loại và điều kiện khác nhau

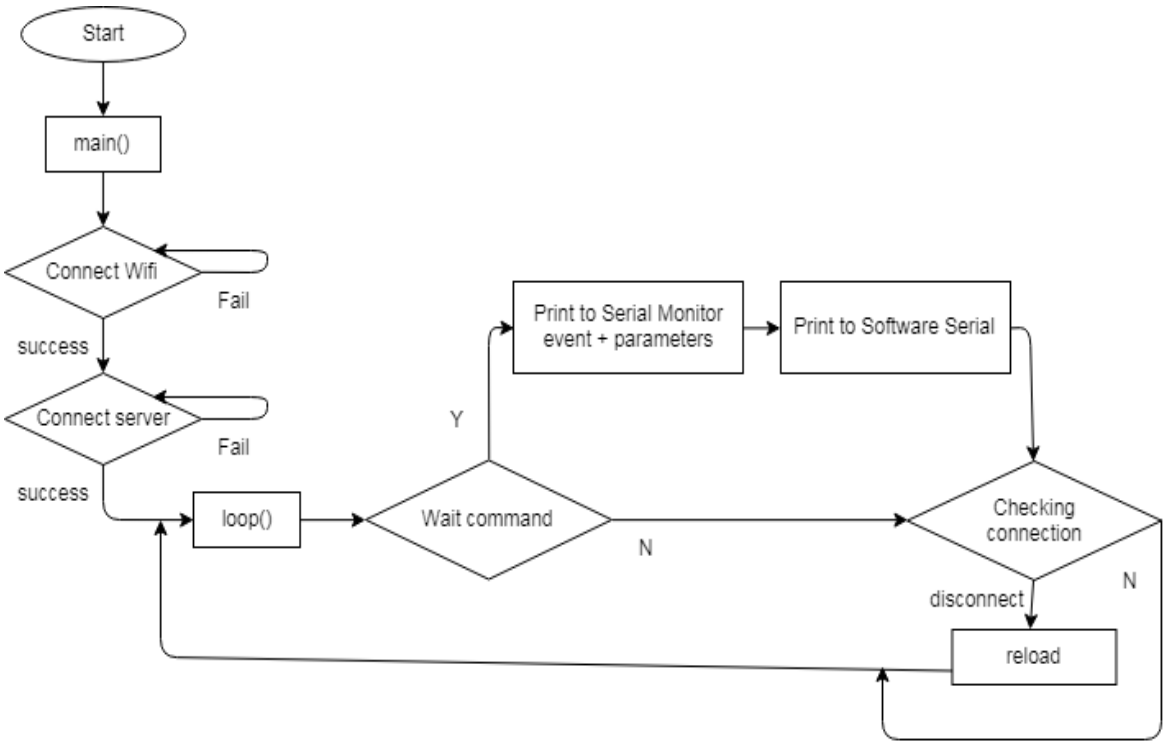
1.2.1. Sơ đồ khối



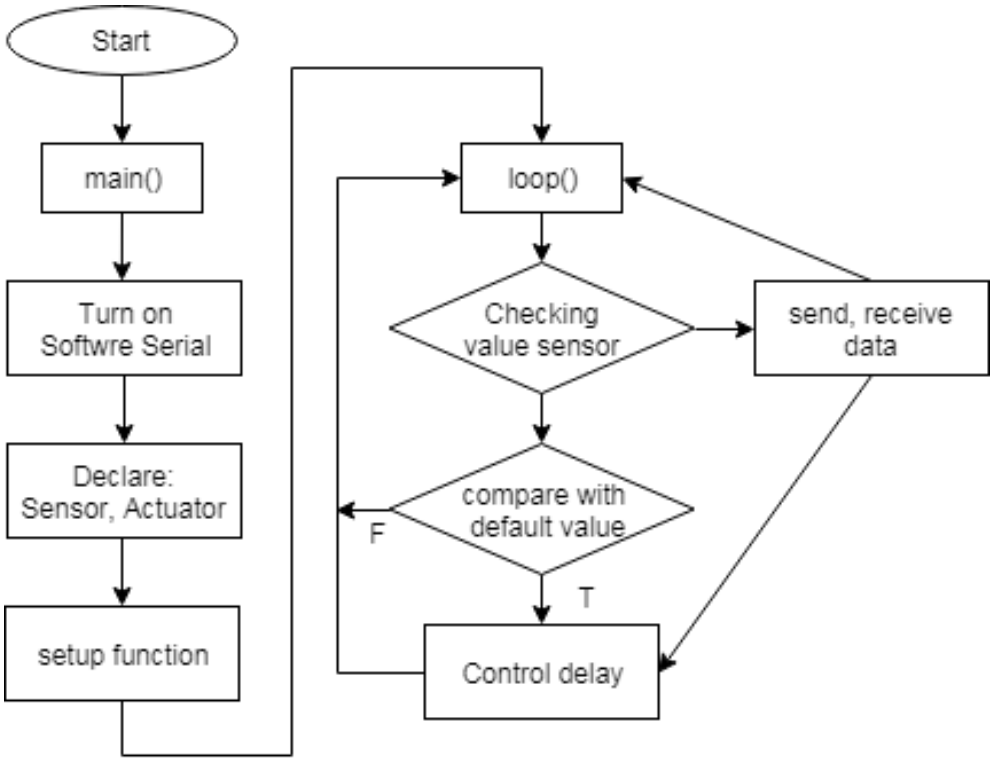
Hình 2.1. Mô hình Socket Server và Socket Client

Ở cấp thấp với arduino mô hình có thể được chia nhỏ hơn như sau:

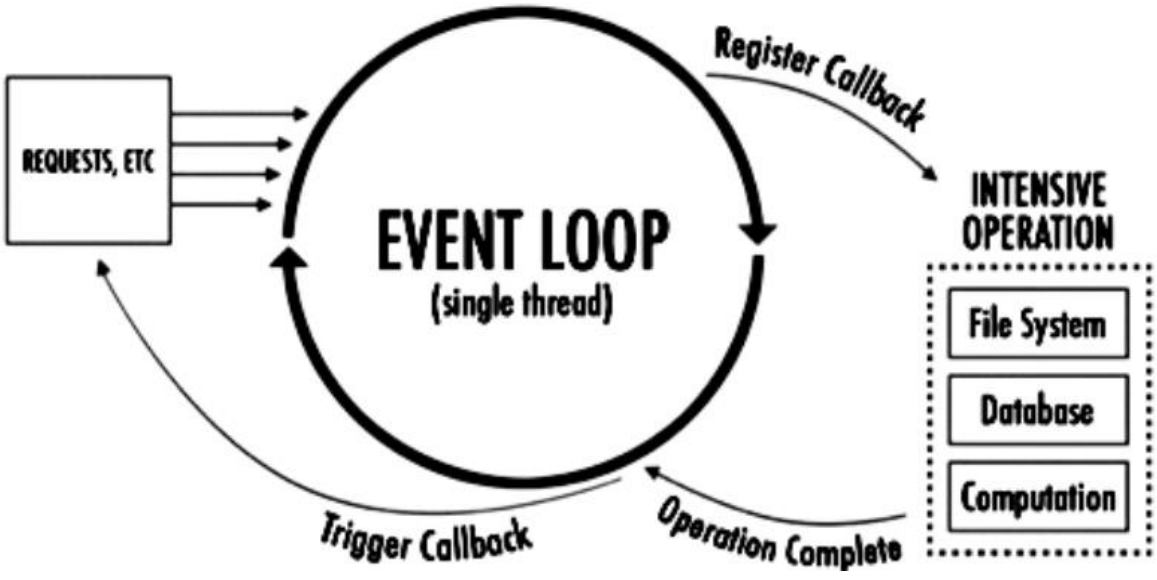
ESP8266



Arduino



Server



1.2.2. Nguyên lý hoạt động

Để đo và lấy được độ ẩm nhiệt độ trong không khí ở bài toán này ta sử dụng cảm biến dht11 với giá thành rẻ (ngoài ra có thể nhắc đến dht22 cùng 1 số loại cảm biến nhiệt độ, độ ẩm khác cũng có thể thay thế). Với độ ẩm đất ta sử dụng module

cảm biến độ ẩm đất có xuất tín hiệu analog và digital. Với ánh sáng ta sử dụng module cảm biến ánh sáng với quang trở.

Những dữ liệu sau khi được sensor cảm biến tiếp nhận từ môi trường sẽ gửi về cho vi điều khiển để xử lý. Ở đây em sử dụng kit Arduino Uno R3 khá phổ biến và phù hợp với những mô hình nhỏ của sinh viên. Nó sử dụng chip vi điều khiển ATmega328 8 bit với 6 chân Analog và 14 chân Digital trong đó có 6 chân PWM.

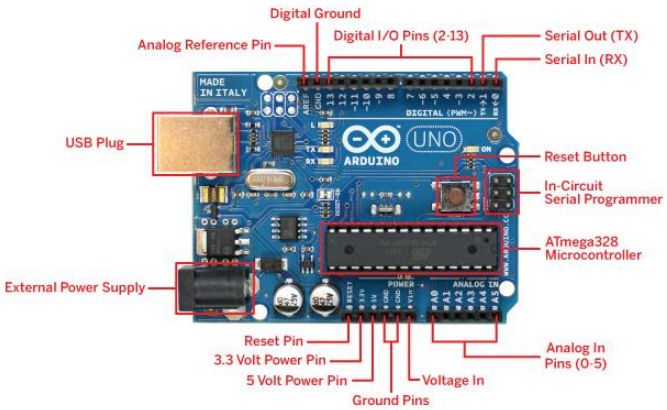
Những dữ liệu sau khi được xử lý bởi arduino sẽ được đưa đến esp8266 tích hợp trong board kit Wemos D1 R2 (có thể sử dung các board khác nhưng để tiện trong quá trình học tập em sửa dụng kit này để dễ dàng cho việc nạp test, tận dụng có sẵn). Chúng giao tiếp với nhau qua cổng serial mà khi cái đặt sẽ khai báo để truyền dữ liệu đã được nén vào chuối json.

Tiếp theo đó, ESP sẽ chuyển những dữ liệu này lên socket server qua internet. Sever sẽ nhận và giải nén những dữ liệu đã đóng gói qua giao diện web để người dùng nhận dữ liệu và điều khiển.

Phần điều khiển và tự động có lắp một số delay hoạt động do sự điều khiển và tự động theo sự thay đổi của môi trường. Ở đây em sửa dụng đèn và máy bơm.

1.2.3. Chọn linh kiện

Arduino Uno R3



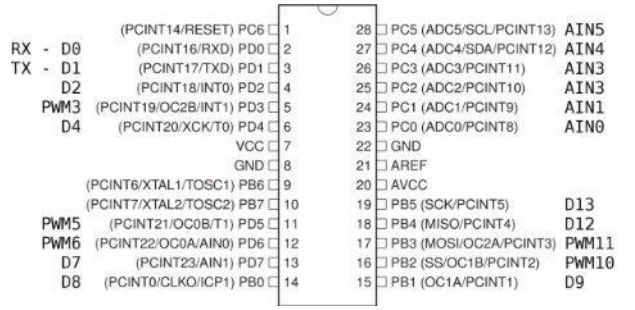
Hình 2.2. Sơ đồ khối của Arduino UNO R3

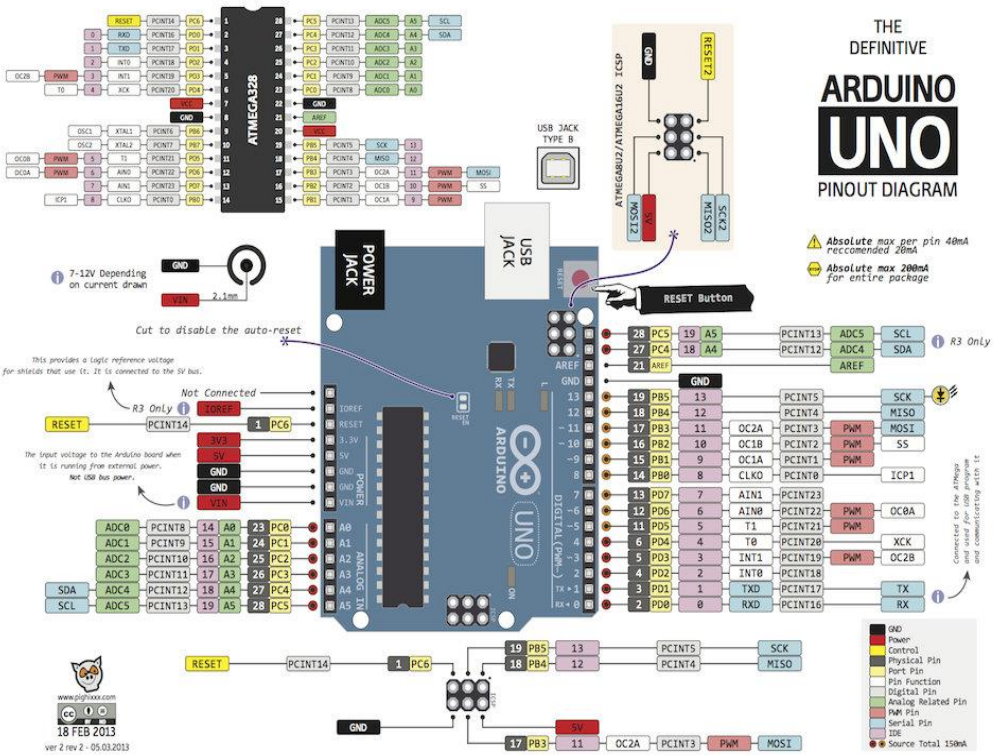
Arduino Uno là một bo mạch vi điều khiển dựa trên chip Atmega328P. Uno có 14 chân I/O digital (trong đó có 6 chân xuất xung PWM), 6 chân Input analog, 1 thạch anh 16MHz, 1 cổng USB, 1 jack nguồn DC, 1 nút reset.

Atmega328P: Là một chíp vi điều khiển được sản xuất bởi hãng Atmel thuộc họ MegaAVR có sức mạnh hơn hẳn Atmega8. Atmega 328P là một bộ vi điều

khiển 8 bít dựa trên kiến trúc RISC bộ nhớ chương trình 32KB ISP flash có thể ghi xóa hàng nghìn lần, 1KB EEPROM, một bộ nhớ RAM vô cùng lớn trong thế giới vi xử lý 8 bít (2KB SRAM)

Với 23 chân có thể sử dụng cho các kết nối vào hoặc ra I/O, 32 thanh ghi, 3 bộ timer/counter có thể lập trình, có các gắt nội và ngoại (2 lệnh trên một vector ngắt), giao thức truyền thông nối tiếp USART, SPI, I2C. Ngoài ra có thể sử dụng bộ biến đổi số tương tự 10 bít (ADC/DAC) mở rộng tới 8 kênh, khả năng lập trình được watchdog timer (bộ đếm thời gian hoạt động liên tục nhằm tự động thực hiện một nhiệm vụ nào đó), hoạt động với 5 chế độ nguồn, có thể sử dụng tới 6 kênh điều chế độ rộng xung (PWM), hỗ trợ bootloader.





Hình 2.3. Sơ đồ chân của ATmega328P và board Arduino Uno R3

Bộ nhớ:

+ *32KB bộ nhớ Flash*: những đoạn lệnh bạn lập trình sẽ được lưu trữ trong bộ nhớ Flash của vi điều khiển. Thường thì sẽ có khoảng vài KB trong số này sẽ được dùng cho bootloader.

+ *2KB cho SRAM* (Static Random Access Memory): giá trị các biến bạn khai báo khi lập trình sẽ lưu ở đây. Khai báo càng nhiều biến thì càng cần nhiều bộ nhớ RAM. Khi ngắt nguồn thì dữ liệu cũng sẽ mất.

+ *1KB cho EEPROM* (Electrically Eraseble Programmable Read Only Memory): đây giống như một chiếc ổ cứng mini – nơi có thể đọc và ghi dữ liệu vào đây.

Nguồn cho Arduino :

- **Arduino UNO R3** có thể được cấp nguồn 5VDC thông qua cổng USB hoặc cấp nguồn ngoài thông qua Adaptor với điện áp khuyên dùng là 7-9VDC.
- Nếu cấp nguồn vượt quá ngưỡng giới hạn như trên sẽ hỏng Arduino UNO R3.
- **GND (Ground):** cực âm của nguồn điện cấp cho Arduino UNO. Khi bạn dùng các thiết bị sử dụng những nguồn điện riêng biệt thì những chân này phải được nối với nhau.
- **5V:** cấp điện áp 5V đầu ra. Dòng tối đa cho phép ở chân này là 500mA.
- **3.3V:** cấp điện áp 3.3V đầu ra. Dòng tối đa cho phép ở chân này là 150mA.
- **Vin (Voltage Input):** Để cấp nguồn ngoài cho Arduino UNO, bạn nối cực dương của nguồn với chân này và cực âm của nguồn với chân GND.
- **RESET:** Khi nhấn nút Reset trên board để Reset vi điều khiển tương đương với việc chân Reset được nối với GND qua 1 điện trở 10KΩ.

Các chân vào ra của Arduino Uno R3:

- **Arduino UNO R3 có 14 chân digital dùng để đọc hoặc xuất tín hiệu**. Chúng chỉ có 2 mức điện áp là 0V và 5V với dòng vào/ra tối đa trên mỗi chân là 40mA. Ở mỗi chân đều có các điện trở pull-up từ được cài đặt ngay trong vi điều khiển ATmega328 (mặc định thì các điện trở này không được kết nối).
- **2 chân Serial 0 (RX) và 1 (TX):** dùng để gửi (Transmit – TX) và nhận (Receive – RX) dữ liệu TTL Serial. Arduino Uno có thể giao tiếp với thiết bị khác thông qua 2 chân này. Kết nối bluetooth có thể nói là kết nối Serial không dây. Nếu không cần giao tiếp Serial bạn không nên sử dụng 2 chân này nếu không cần thiết
- **Chân PWM (~): 3, 5, 6, 9, 10, và 11:** Cho phép bạn xuất ra xung PWM với độ phân giải 8bit (giá trị từ 0 → 28-1 tương ứng với 0V → 5V) bằng hàm analogWrite(). Nói một cách đơn giản, bạn có thể điều chỉnh được điện áp ra ở chân này từ mức 0V đến 5V thay vì chỉ cố định ở mức 0V và 5V như những chân khác.
- **Chân giao tiếp SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** Ngoài các chức năng thông thường, 4 chân này còn dùng để truyền phát dữ liệu bằng giao thức SPI với các thiết bị khác.

- **LED 13**: trên Arduino UNO có 1 đèn led màu cam (kí hiệu chữ L). Khi bấm nút Reset, bạn sẽ thấy đèn này nhấp nháy để báo hiệu. Nó được nối với chân số 13. Khi chân này được người dùng sử dụng, LED sẽ sáng.
- **Arduino UNO có 6 chân analog (A0 → A5)** cung cấp độ phân giải tín hiệu 10bit để đọc giá trị điện áp trong khoảng 0V → 5V. Với chân AREF trên board, bạn có thể để đưa vào điện áp tham chiếu khi sử dụng các chân analog. Tức là nếu bạn cấp điện áp 2.5V vào chân này thì bạn có thể dùng các chân analog để đo điện áp trong khoảng từ 0V → 2.5V với độ phân giải vẫn là 10bit.
- **Đặc biệt**, Arduino UNO có 2 chân A4 (SDA) và A5 (SCL) hỗ trợ giao tiếp I2C/TWI với các thiết bị khác.

Lưu ý:

-Arduino UNO không có bảo vệ cắm ngược nguồn vào. Do đó phải hết sức cẩn thận, kiểm tra các cực âm – dương của nguồn trước khi cấp cho Arduino UNO. Việc làm chập mạch nguồn vào của Arduino UNO sẽ biến nó thành một miếng nhựa chặn giấy. mình khuyên bạn nên dùng nguồn từ cổng USB nếu có thể.

-Các chân 3.3V và 5V trên Arduino là các chân dùng để cấp nguồn ra cho các thiết bị khác, không phải là các chân cấp nguồn vào. Việc cấp nguồn sai vị trí có thể làm hỏng board. Điều này không được nhà sản xuất khuyến khích.

-Cấp nguồn ngoài không qua cổng USB cho Arduino UNO với điện áp dưới 6V có thể làm hỏng board.

-Cấp điện áp trên 13V vào chân RESET trên board có thể làm hỏng vi điều khiển ATmega328.

-Cường độ dòng điện vào/ra ở tất cả các chân Digital và Analog của Arduino UNO nếu vượt quá 200mA sẽ làm hỏng vi điều khiển.

-Cấp điệp áp trên 5.5V vào các chân Digital hoặc Analog của Arduino UNO sẽ làm hỏng vi điều khiển.

-Cường độ dòng điện qua một chân Digital hoặc Analog bất kì của Arduino UNO vượt quá 40mA sẽ làm hỏng vi điều khiển. Do đó nếu không dùng để truyền nhận dữ liệu, bạn phải mắc một điện trở hạn dòng.

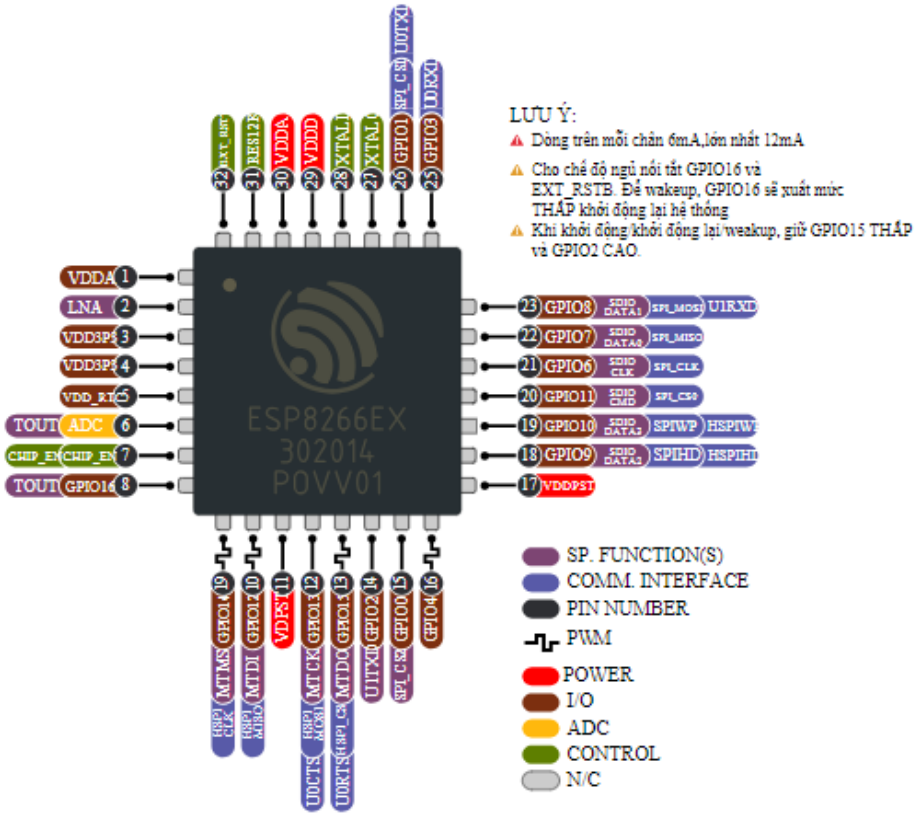
ESP8266

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



Hình 2.4. Sơ đồ chân ESP8266EX

Thông số phần cứng

- 32-bit RISC CPU : Tensilica Xtensa LX106 chạy ở xung nhịp 80 MHz
- Hỗ 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
- Module ESP12E

KIT ARDUINO WIFI ESP8266 WEMOS D1 R2

KIT ARDUINO WIFI ESP8266 WEMOS phiên bản D1 R2 là phiên bản mới nhất chính hãng Wemos được nâng cấp từ phiên bản D1 fix các lỗi về phần cứng và Firmware với khả năng hoạt động ổn định tối đa.

Kit Arduino Wifi ESP8266 NodeMCU Lua WeMos D1 R2 là phiên bản mới nhất từ WeMos được thiết kế với hình dáng tương tự Arduino Uno nhưng trung tâm lại là module wifi Soc ESP8266 được build lại firmware NodeMCU Lua để có thể chạy với chương trình Arduino.

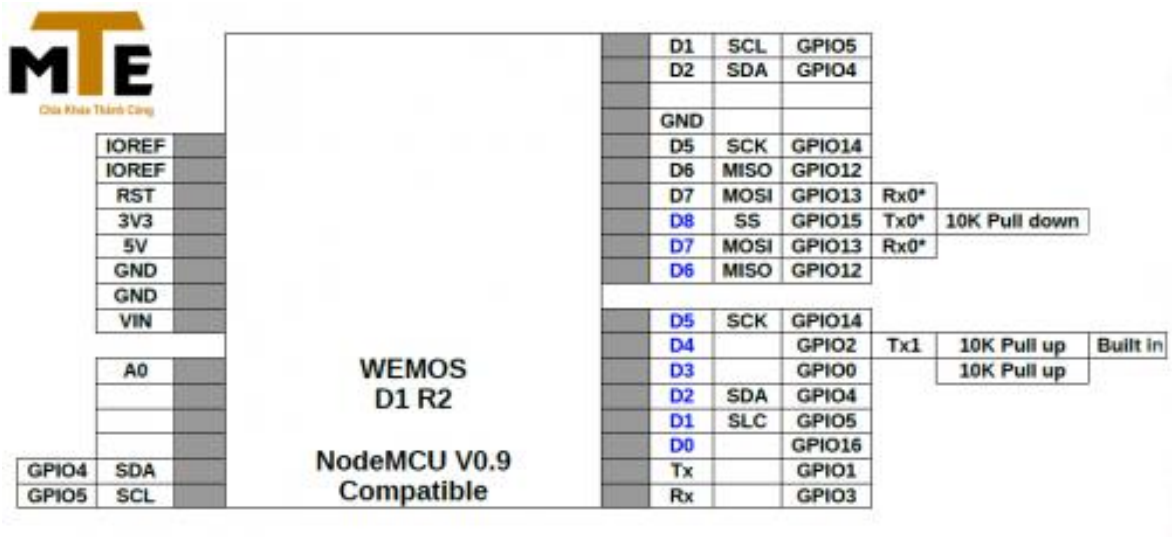
Kit Arduino Wifi ESP8266 NodeMCU, WeMos D1 R2 thích hợp và dễ dàng thực hiện các ứng dụng thu thập dữ liệu và điều khiển qua Wifi.

Hình 2.5. Wemos D1 R2

Thông số kỹ thuật

- Vi điều khiển: ESP8266EX
- Điện áp hoạt động: 3.3V
- Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)
- Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)
- Bộ nhớ Flash: 4MB
- Điện áp vào: 9-24V
- Điện áp ra: 5V – Dòng max: 1A
- Giao tiếp: Cable Micro USB
- Wifi: 2.4 GHz
- Hỗ trợ bảo mật: WPA/WPA2
- Tích hợp giao thức TCP/IP
- Kích thước: 68.6mm x 53.4mm (2.701″ x 2.102″)

- Lập trình trên các ngôn ngữ: C/C++, Micropython, NodeMCU – Lua



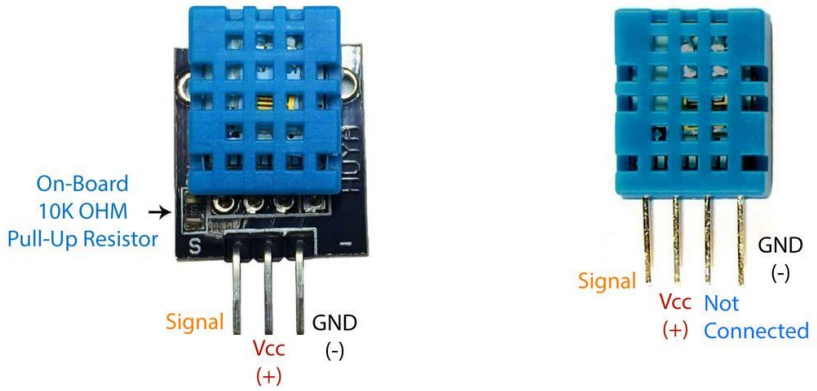
Hình 2.6. Sơ đồ chân Wemos D1 R2

Module cảm biến nhiệt độ độ ẩm DHT11

Cảm biến độ ẩm, nhiệt độ DHT11 ra chân được tích hợp sẵn điện trở 5,1k giúp người dùng dễ dàng kết nối và sử dụng hơn so với cảm biến DHT11 chưa ra chân, module lấy dữ liệu thông qua giao tiếp 1 wire (giao tiếp 1 dây). Bộ tiền xử lý tín hiệu tích hợp trong cảm biến giúp bạn có được dữ liệu chính xác mà không cần phải qua bất kỳ tính toán nào. Module được thiết kế hoạt động ở mức điện áp 5VDC.

Thông Tin Kỹ Thuật :

- Điện áp hoạt động : 5VDC
- Chuẩn giao tiếp: TTL, 1 wire.
- Khoảng đo độ ẩm: 20%-80%RH sai số ± 5%RH
- Khoảng đo nhiệt độ:0-50 °C sai số ± 2°C
- Tần số lấy mẫu tối đa 1Hz (1 giây / lần)
- Kích thước : 28mm x 12mm x10mm



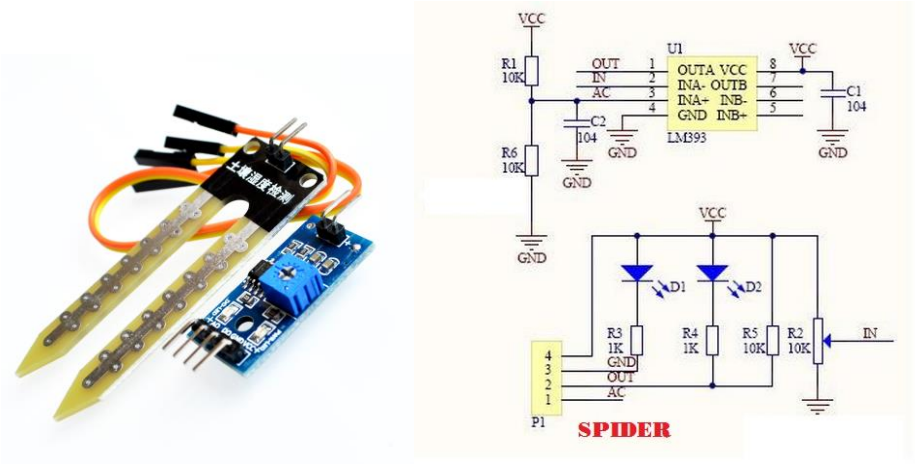
Hình 2.7. Sơ đồ chân cảm biến DHT11

Module cảm biến độ ẩm đất

Hai đầu đo của cảm biến được cắm vào đất để phát hiện độ ẩm. Dùng dây nối giữa cảm biến và module chuyển đổi. Thông tin về độ ẩm đất sẽ được đọc về và gởi tới module chuyển đổi.

Module chuyển đổi có cấu tạo chính gồm một IC so sánh LM393, một biến trở , 4 điện trở dán 100 ohm và 2 tụ dán. Biến trở có chức năng định ngưỡng so sánh với tín hiệu độ ẩm đất đọc về từ cảm biến. Ngưỡng so sánh và tín hiệu cảm biến sẽ là 2 đầu vào của IC so sánh LM393. Khi độ ẩm thấp hơn ngưỡng định trước, ngõ ra của IC là mức cao (1), ngược lại là mức thấp (0).

Cảm biến độ ẩm đất có thể được sử dụng cho các ứng dụng nông nghiệp, tưới nước tự động cho các vườn cây khi đất khô, hoặc dùng trong các ứng dụng của hệ thống nhà thông minh.



Hình 2.8. Module cảm biến độ ẩm đất và mạch nguyên lý

Đặc điểm

- Điện áp hoạt động: 3.3V-5V
- Kích thước PCB: 3cm * 1.6cm
- Led báo hiệu
  - Led đỏ báo nguồn
  - Led xanh báo mức độ ẩm ở pin DO
- Mô tả các pin trên module

Pin	Mô tả
VCC	3.3V-5V
GND	GND
DO	Đầu ra tín hiệu số (0 và 1)
AO	Đầu ra Analog (tín hiệu tương tự)

Module cảm biến ánh sáng

Cảm biến ánh sáng quang trở phát hiện cường độ ánh sáng, sử dụng bộ cảm biến photoresistor loại nhạy cảm, cho tín hiệu ổn định, rõ ràng và chính xác hơn so với quang trở.

Ngõ ra D0 trên cảm biến được dùng để xác định cường độ sáng của môi trường, khi ở ngoài sáng, ngõ ra D0 là giá trị 0, khi ở trong tối, ngõ ra D0 là 1. Trên cảm biến có 1 biến trở để điều chỉnh cường độ sáng phát hiện, khi văn cùng chiều kim đồng hồ thì sẽ làm giảm cường độ sáng nhận biết của cảm biến, tức là môi trường phải ít sáng hơn nữa thì cảm biến mới đọc gía trị digital là 1.

Ngõ ra D1 đọc giá trị điện áp, nếu sử dụng arduino uno r3 sẽ có 1024 giá trị chia đều để lấy phần trăm độ sáng trong môi trường.

Module RELAY 1 KÊNH 5V & 12V HIGH/LOW

Rơ-le là một công tắc. Nhưng khác với công tắc ở một chỗ cơ bản, rơ-le được kích hoạt bằng điện thay vì dùng tay người. Chính vì lẽ đó, rơ-le được dùng làm công tắc điện tử! Vì rơ-le là một công tắc nên nó có 2 trạng thái: đóng và mở.

Các mức hiệu điện thế tối đa và cường độ dòng điện tối đa của đồ dùng điện khi nối vào module relay:

- 0A - 250VAC: Cường độ dòng điện tối đa qua các tiếp điểm của rơ-le với hiệu điện thế <= 250V (AC) là 10A.
- 10A - 30VDC: Cường độ dòng điện tối đa qua các tiếp điểm của rơ-le với hiệu điện thế <= 30V (DC) là 10A.
- 10A - 125VAC: Cường độ dòng điện tối đa qua các tiếp điểm của rơ-le với hiệu điện thế <= 125V (AC) là 10A.
- 10A - 28VDC: Cường độ dòng điện tối đa qua các tiếp điểm của rơ-le với hiệu điện thế <= 28V (DC) là 10A.
- SRD-05VDC-SL-C: Hiện điện thế kích tối ưu là 5V.

Rơ-le bình thường gồm có 6 chân. Trong đó có 3 chân để kích, 3 chân còn lại nối với đồ dùng điện công suất cao.

3 chân dùng để kích

+: cấp hiệu điện thế kích tối ưu vào chân này.

- : nối với cực âm

S: chân tín hiệu, tùy vào loại module rơ-le mà nó sẽ làm nhiệm vụ kích rơ-le

Nếu bạn đang dùng module rơ-le kích ở mức cao và chân S bạn cấp điện thế dương vào thì module rơ-le của bạn sẽ được kích, ngược lại thì không.

Tương tự với module rơ-le kích ở mức thấp.

3 chân còn lại nối với đồ dùng điện công suất cao:

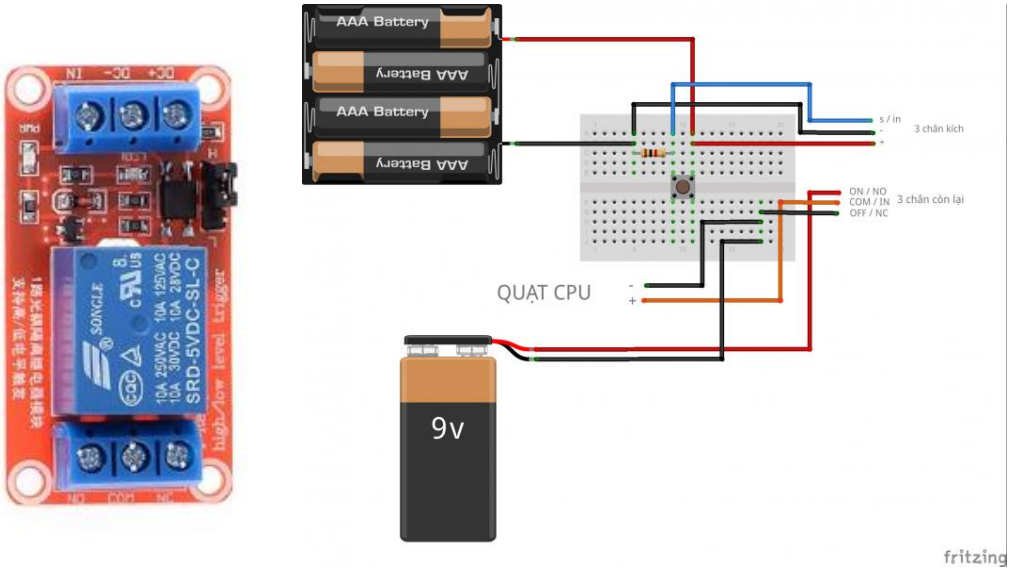
- COM: chân nối với 1 chân bất kỳ của đồ dùng điện, nhưng mình khuyên bạn nên mắc vào đây chân lửa (nóng) nếu dùng hiệu điện thế xoay chiều và cực dương nếu là hiệu điện một chiều.
- ON hoặc NO: chân này bạn sẽ nối với chân lửa (nóng) nếu dùng điện xoay chiều và cực dương của nguồn nếu dòng điện một chiều.

- OFF hoặc NC: chân này bạn sẽ nối chân lạnh (trung hòa) nếu dùng điện xoay chiều và cực âm của nguồn nếu dùng điện một chiều.

Relay 1 Kênh 12V được thiết kế chắc chắn, khả năng cách điện tốt, có sẵn 4 lỗ gắn ốc 3mm. Trên module đã có sẵn mạch kích relay sử dụng transistor và IC cách ly quang giúp cách ly hoàn toàn mạch điều khiển (vi điều khiển) với rơ le bảo đảm vi điều khiển hoạt động ổn định.

Có sẵn header rất tiện dụng khi kết nối với vi điều khiển.

Tín hiệu kích hoạt rơ le đóng: LOW , tín hiệu HIGH sẽ làm rơ le mở trở lại.



Hình 2.9. Relay và cách lắp đặt

1.2.4. Phần mềm lập trình

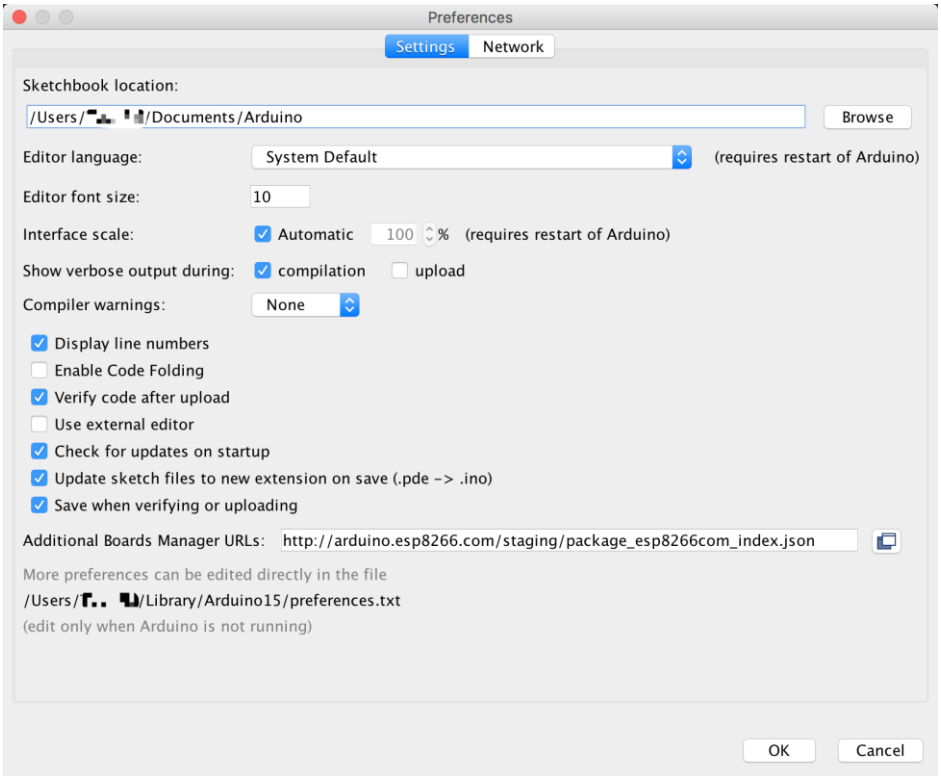
Arduino IDE

Cài đặt Arduino IDE.

Download và cài đặt arduino từ trang chủ của arduino. Link donwload: www.arduino.cc/en/Main/Software. Tùy hệ điều hành mà chọn gói cài đặt thích hợp.

Cài đặt bộ công cụ, trình biên dịch, SDK hỗ trợ chip ESP8266 trong IDE.

Với bộ công cụ này, chúng ta có thể dễ dàng lập trình, biên dịch và sử dụng các thư viện dành cho ESP8266 trực tiếp trên Arduino IDE. Mở Arduino IDE, trên thanh Menu chọn File → Preferecens, trong tab settings chọn các tùy chọn như hình dưới:



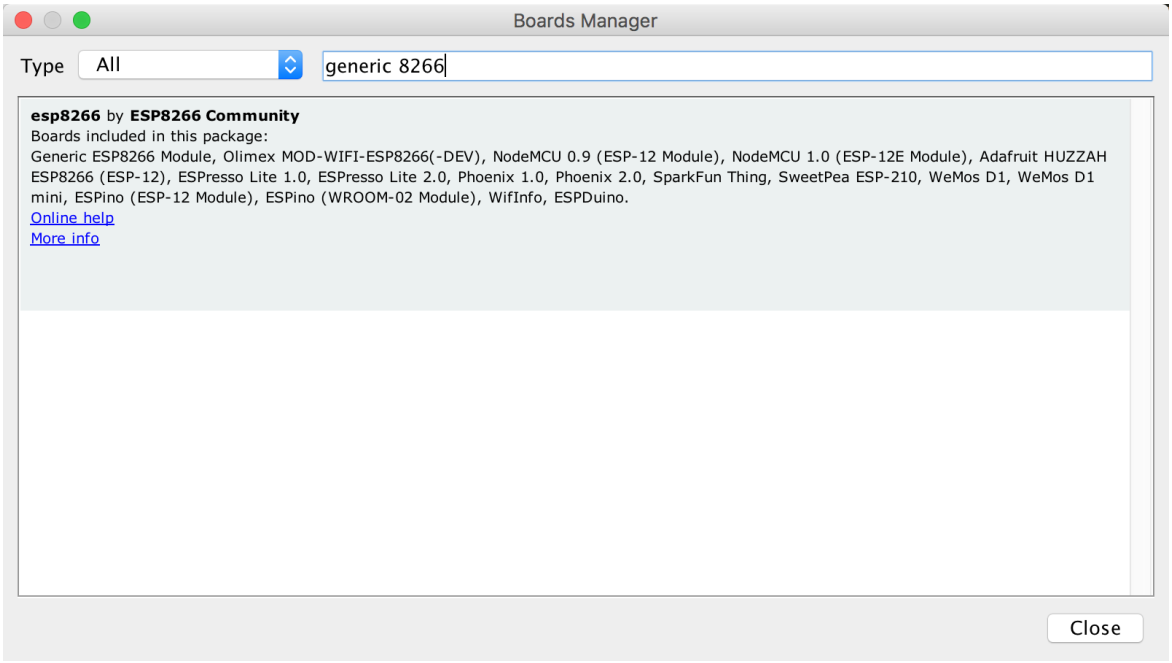
Hình 2.10. Thêm file thông tin board ESP8266

Sketchbook location là đường dẫn mà bạn muốn lưu Sketch (file chương trình), trên các hệ điều hành Unix liked đường dẫn mặc định là: /home/name_your_computer/Arduino. Đây cũng sẽ là vị trí lưu những thư viện mà chúng ta sẽ thêm vào sau này.

Mục Additional Board Manager URLs field nhập đường dẫn http://arduino.esp8266.com/stable/package_esp8266com_index.json.

Cài đặt board ESP8266.

Mở Boards Manager ở mục Tools trên thanh menu-bar → tìm board cần sử dụng với keyword Generic 8266 → chọn board cần cài đặt như hình và nhấn vào install.

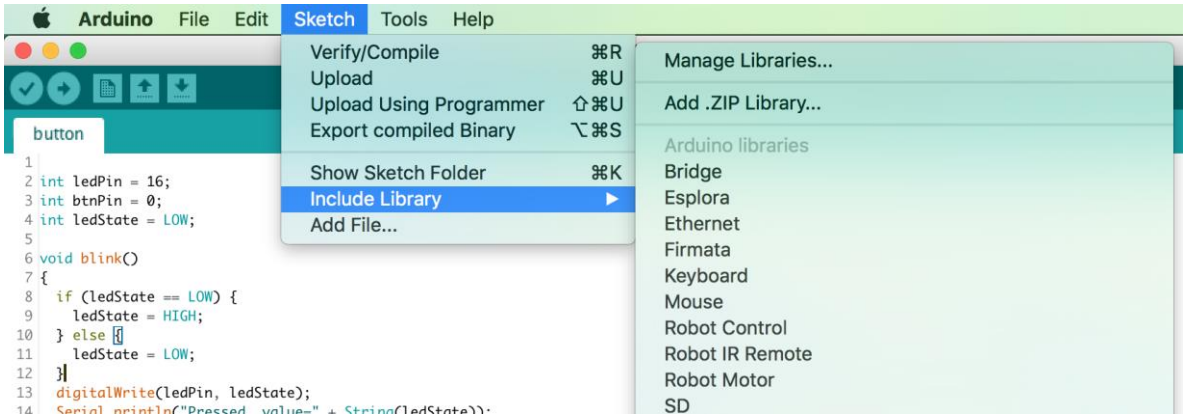


Hình 2.11. Cài đặt board ESP8266

Cài đặt thư viện Arduino

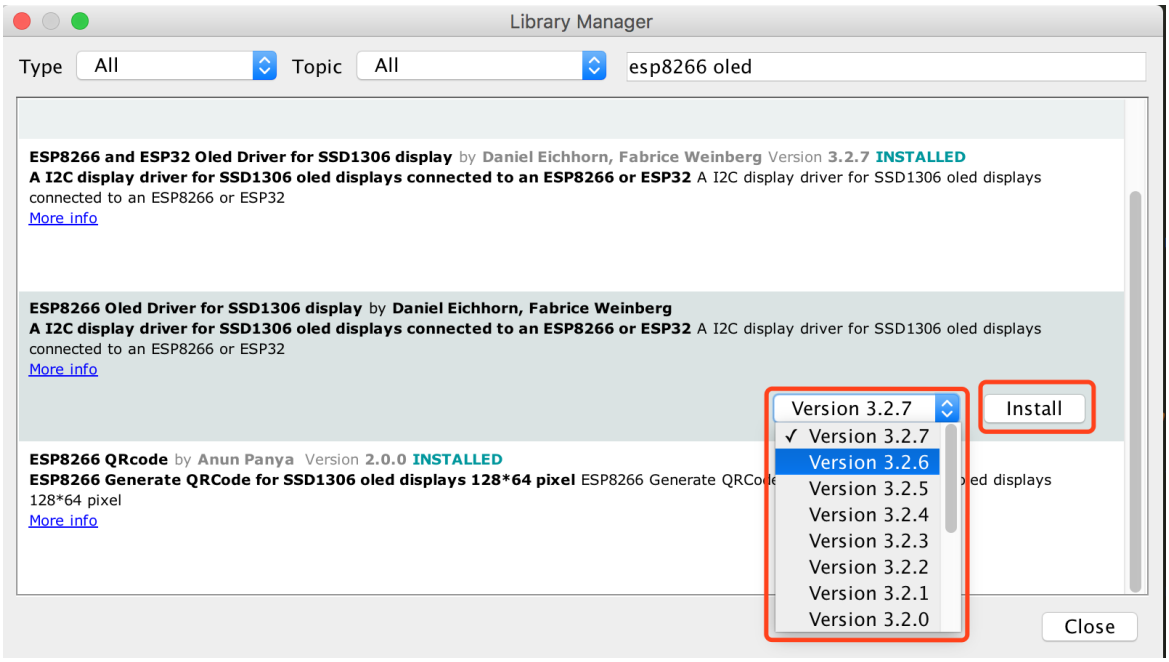
Một số thư viện do các nhà phát triển khác công bố và được tự do sử dụng có thể cài đặt trực tiếp bằng công cụ Library Manager của Arduino.

Khởi động arduino IDE và chọn mục Sketch ⇒ include library ⇒ Manage libraries:



Hình 2.12. Khởi động Library Manager

Trong mục libbrary manager nhập nội dung thư viện cần tìm tại hộp thoại text box, chọn phiên bản, rồi nhấn install, Những thư viện đã được cài đặt sẽ có text hiển thị INSTALLED ở đầu mỗi thư viện. Ví dụ tìm thư viện OLED liên quan đến ESP8266:



Hình 2.13. Cài đặt thư viện

Node.js

Node.js là một Javascript Run time Cross Platform (chạy đa hệ điều hành) được xây dựng dựa trên mã nguồn mở Google's V8 JavaScript engine cho Chrome (Browser). Node.js cho phép các lập trình viên có thể xây dựng ứng dụng Server Side, truy cập vào tài nguyên hệ thống và thực hiện được phần lớn các tác vụ hệ điều hành có thể thực hiện bằng ngôn ngữ Javascript, hoặc liên kết C++.

Hiện nay trên thế giới đã có nhiều công ty ứng dụng Node.js xây dựng các hệ thống production lớn, như Paypal, hoặc các microservice dựa trên Node.js cũng được triển khai ở đa số các hãng hàng đầu về công nghệ.

Nền tảng Cloud của gần như tất cả các nhà phát triển lớn hiện nay đều hỗ trợ thực thi Node.js, điển hình như Amazon Lambda, Google Script, IBM Blumix, Microsoft Azure …

Ngôn ngữ lập trình Javascript được cải tiến liên tục, hiện nay là Ecmascript 6 (ES5, ES2015) và đang được cải tiến rất nhanh, với nhiều ưu điểm như dễ học, xúc tích, OOP…

Một lý do Node.js được ưa chuộng nữa là đa phần các lập trình viên viết Web, Mobile đều biết, và giờ đây, nhờ Node.js mà họ có thể triển khai các ứng dụng Server

Side bằng Javascript, mà không cần dùng ngôn ngữ nào khác (như trước kia phải cần Java, PHP …)

Trình soạn thảo nodejs

Nếu ở phần Chip, lập trình cho ESP8266 ta đã có Arduino IDE, bao gồm cả trình soạn thảo. Nhưng với Node.js thì ta cần 1 trình soạn thảo khác. Có thể lựa chọn các trình soạn thảo phổ biến hiện nay như Atom, Visual Code, Sublime Text, Adobe dreamweaver… Đồng thời trong khi viết chương trình sẽ sử dụng thêm các frameword, các thư viện có sẵn để sử dụng trong project.

1.2.5. Ngôn ngữ lập trình

Arduino và ESP

Để lập trình phần nhúng cho Arduino và ESP ta sử dụng ngôn ngữ C đơn giản và dễ mã hóa thành file hex để nạp vào vi xử lý. Đó là lý do ta cần sử dụng đến trình biên dịch để vi xử lý hiểu được những lệnh mà ta cài đặt cho nó. Giữa ngôn ngữ người đến ngôn ngữ lập trình và ngôn ngữ máy. Vậy là ta có thể giao tiếp với máy và điều khiển nó.

Web App

Để viết một chương trình điều khiển Arduino từ điện thoại thông minh, thì theo cách thông thường, chúng ta phải học Java để lập trình Socket Client cho Android, học Object-C hoặc Swift để lập trình Socket Client trên iOS. Thay vì viết ứng dụng cho từng dòng máy ta có thể xây dựng trang web mà mọi thiết bị kết nối internet và đăng nhập vào trình duyệt web đều có thể sử dụng.

Trang web đơn lẻ được viết bằng html gọi là Web Page. Tập hợp nhiều trang web đơn lẻ, thành một trang web lớn, có chung tên miền, được gọi là Website. Ban đầu, các website chỉ bao gồm text, hình ảnh và video, liên kết với nhau thông qua các link. Tác dụng của website là lưu trữ và hiển thị thông tin. Người dùng chỉ có thể đọc, xem, click các link để di chuyển giữa các page. Về sau, với sự ra đời của các ngôn ngữ server: CGI, Perl, PHP, … các website đã trở nên "động" hơn, có thể tương tác với người dùng. Từ đây, người dùng có thể dùng web để "thực hiện một công việc nào đó bằng máy tính", do đó web app ra đời.

Nói dễ hiểu, web app là những ứng dụng chạy trên web. Thông qua web app, người dùng có thể thực hiện một số công việc: tính toán, chia sẻ hình ảnh, mua sắm … Tính tương tác của web app cao hơn website rất nhiều. Với một số người không rành về IT, tất cả những thứ online, vào được bằng trình duyệt đều là website cả. Do đó họ thường yêu cầu bạn là: website quản lý siêu thị, website bán hàng, … thực chất chúng đều là webapp hết.

Cách mà web application hoạt động:

• Người dùng kích hoạt request tới web server qua Internet, thông qua trình duyệt web hoặc giao diện người dùng của ứng dụng.

• Web server chuyển tiếp request này đến web application server thích hợp.

• Máy chủ ứng dụng Web (web application server) thực hiện nhiệm vụ được yêu cầu - chẳng hạn như truy vấn cơ sở dữ liệu hoặc xử lý dữ liệu - sau đó tạo ra các kết quả của dữ liệu được yêu cầu.

• Máy chủ ứng dụng web gửi kết quả đến máy chủ web với thông tin được yêu cầu hoặc dữ liệu đã được xử lý.

• Máy chủ web phản hồi response lại cho khách hàng các thông tin được yêu cầu sau đó xuất hiện trên màn hình của người dùng.

HTML

HTML là chữ viết tắt của cụm từ HyperText Markup Language((Xem thêm tại http://vi.wikipedia.org/wiki/HTML)) (dịch là Ngôn ngữ đánh dấu siêu văn bản) được sử dụng để tạo một trang web, trên một website có thể sẽ chứa nhiều trang và mỗi trang được quy ra là một tài liệu HTML (thi thoảng mình sẽ ghi là một tập tin HTML). Cha đẻ của HTML là Tim Berners-Lee, cũng là người khai sinh ra World Wide Web và chủ tịch của World Wide Web Consortium (W3C – tổ chức thiết lập ra các chuẩn trên môi trường Internet).

Một tài liệu HTML được hình thành bởi các phần tử HTML (HTML Elements) được quy định bằng các cặp thẻ (tag), các cặp thẻ này được bao bọc bởi một dấu ngoặc ngọn (ví dụ <html>) và thường là sẽ được khai báo thành một cặp, bao gồm thẻ mở và thẻ đóng (ví dụ và). Các văn bản muốn được đánh dấu

bằng HTML sẽ được khai báo bên trong cặp thẻ (ví dụ Đây là chữ in đậm). Nhưng một số thẻ đặc biệt lại không có thẻ đóng và dữ liệu được khai báo sẽ nằm trong các thuộc tính (ví dụ như thẻ <img>).

Một tập tin HTML sẽ bao gồm các phần tử HTML và được lưu lại dưới đuôi mở rộng là .html hoặc .htm.

CSS

CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (ví dụ như HTML). Bạn có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,…thì CSS sẽ giúp chúng ta có thể thêm một chút "phong cách" vào các phần tử HTML đó như đổi màu sắc trang, đổi màu chữ, thay đổi cấu trúc,…rất nhiều.

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

JavaScript

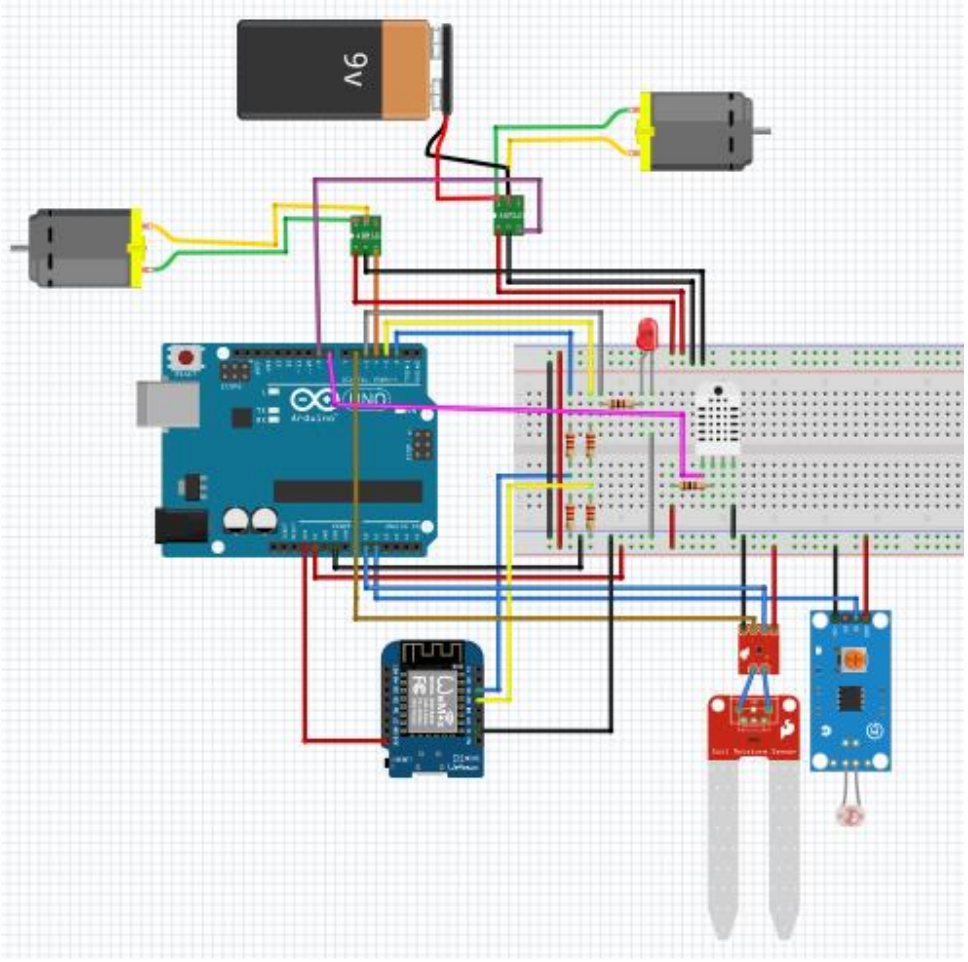
Javascript là một ngôn ngữ lập trình kịch bản dựa vào đối tượng phát triển có sẵn hoặc tự định nghĩa ra, javascript được sử dụng rộng rãi trong các ứng dụng Website. Javascript được hỗ trợ hầu như trên tất cả các trình duyệt như Firefox, Chrome, ... thậm chí các trình duyệt trên thiết bị di động cũng có hỗ trợ.

Những ứng dụng to lớn của Javascript khiến người ta không thể quên nó được. Hiện nay có rất nhiều libraries và framework được viết từ Javascript như:

- AngularJS: Một thư viện dùng để xây dựng ứng dụng Single Page
- NodeJS: được phát triển phía Server dùng để xây dựng ứng dụng realtime
- Sencha Touch: Một Framework dùng để xây dựng ứng dụng Mobile
- ExtJS: Một Framework dùng xây dựng ứng dụng quản lý (Web Applications)
- jQuery: Một thư viện rất mạnh về hiệu ứng
- ReactJS: Một thư viện viết ứng dụng mobie

Chương 2: Tính toán và thiết kế

2.1. Thiết kế, lắp ráp mạch



Hình 2.14. Mô phỏng mạch bằng frizing

2.2. Lập trình hệ thống:

Chúng ta sẽ tiến hành lập trình cho vi điều khiển và phía socket client, server

2.2.1. Arduino

Chương trình phía dưới sẽ được nạp vào arduino với 4 thư viện được khai báo. Thư viện được tải xuống từ nguồn mở đã được sửa đổi bổ sung.

```
1. #include<DHT.h>
2. #include <ArduinoJson.h>
3. #include <SoftwareSerial.h>
4. #include <SerialCommand.h>
5. const byte RX = 3;
```

```
6.  const byte TX = 2;
7.
8.  SoftwareSerial mySerial = SoftwareSerial(RX, TX);
9.  SerialCommand sCmd(mySerial);
10.
11. int mayBom = 4, den = 5;
12. int doAmSensor = 6;
13. int autoTuoi = 9;
14. int autoDen=10;
15. int doAmAnalog = A0;
16. int camBienAnhSang= A1;
17. const int DHTPIN = 8;
18.
19. const int DHTTYPE = DHT11;
20. DHT dht(DHTPIN, DHTTYPE);
21. const unsigned long CHU_KY_1_LA_BAO_NHIEU = 5000UL;
22.
23. void setup() {
24.   Serial.begin(57600);
25.   mySerial.begin(57600);
26.
27.   pinMode(mayBom,OUTPUT);
28.   pinMode(den,OUTPUT);
29.   pinMode(autoTuoi,OUTPUT);
30.   pinMode(autoDen,OUTPUT);
31.
32.   pinMode(doAmSensor,INPUT);
33.   pinMode(doAmAnalog,INPUT);
34.   dht.begin();
35.   sCmd.addCommand("DELAY",    delay);
36.   sCmd.addCommand("doAm",   doAm_detect);
37.   sCmd.addCommand("DHT11",   dht11);
38.   Serial.println("Da san sang nhan lenh");
39. }
40.
41. unsigned long chuky1 = 0;
42. void loop() {
43.   if (millis() - chuky1 > CHU_KY_1_LA_BAO_NHIEU) {
44.     chuky1 = millis();
45.     doAm_detect();
46.     dht11();
47.     if(digitalRead(doAmSensor)==1){
48.       digitalWrite(autoTuoi,1);
49.     } else{
50.       digitalWrite(autoTuoi,0);
51.     }
52.     int value= analogRead(camBienAnhSang);
53.     float A1=(5/(value*0.004887585533));
```

```
54.     float Rldr1 = 10/(A1 - 1);
55.     float anhSang = 500/Rldr1;
56.     if(anhSang<200){
57.       digitalWrite(autoDen,1);
58.     } else digitalWrite(autoDen,0);
59.   }
60.   sCmd.readSerial();
61. }
62. void delay() {
63.   Serial.println("DELAY");
64.   char *json = sCmd.next();
65.   Serial.println(json);
66.   StaticJsonBuffer<200> jsonBuffer;
67.   JsonObject& root = jsonBuffer.parseObject(json);
68.
69.   int mayBomStatus = root["delay"][0];
70.   int denStatus = root["delay"][1];
71.
72.   //kiểm tra giá trị
73.   Serial.print(F("mayBomStatus "));
74.   Serial.println(mayBomStatus);
75.   Serial.print(F("denStatus "));
76.   Serial.println(denStatus);
77.
78.   StaticJsonBuffer<200> jsonBuffer2;
79.   JsonObject& root2 = jsonBuffer2.createObject();
80.   root2["mayBomStatus"] = mayBomStatus;
81.   root2["denStatus"] = denStatus;
82.
83.   JsonArray& data = root2.createNestedArray("data");
84.   data.add(mayBomStatus);
85.   data.add(denStatus);
86.
87.   mySerial.print("DELAY_STATUS");
88.   mySerial.print('\r');
89.   root2.printTo(mySerial);
90.   mySerial.print('\r');
91.   root2.printTo(Serial);
92.
93.   digitalWrite(mayBom, mayBomStatus);
94.   digitalWrite(den, denStatus);
95. }
96.
97. void doAm_detect() {
98.   StaticJsonBuffer<200> jsonBuffer;
99.   JsonObject& root = jsonBuffer.createObject();
100.        root["digital"] = digitalRead(doAmSensor);
```

```
        root["message"] = digitalRead(doAmSensor) ? "Cần tưới nước!" : "Đủ nước" ;
        int value = analogRead(doAmAnalog);
        int a = map(value, 0, 1023, 0, 100);
        int percent = 100-a;
        root["analog"] = percent;
        mySerial.print("doAm");
        mySerial.print('\r');
        root.printTo(mySerial);
        mySerial.print('\r');
      }
      void dht11() {
        StaticJsonBuffer<200> jsonBuffer;
        JsonObject& root = jsonBuffer.createObject();
        float hum = dht.readHumidity();
        float tem = dht.readTemperature();
        root["hum"] = hum;
        root["tem"] = tem;
        int value= analogRead(camBienAnhSang);
        float A1=(5/(value*0.004887585533));
        float Rldr1 = 10/(A1 - 1);
        float anhSang = 500/Rldr1;
        root["camBienAnhSang"] = anhSang;

        mySerial.print("DHT11");
        mySerial.print('\r');
        root.printTo(mySerial);
        mySerial.print('\r');
      }
```

2.2.2. ESP8266 (Wemos D1 R2)

```
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <SocketIOClient.h>
#include <SerialCommand.h>

extern "C" {
  #include "user_interface.h"
}
const byte Rx = 0;
const byte Tx = 2;

SoftwareSerial mySerial(Rx, Tx, false, 256);
SerialCommand sCmd(mySerial);

SocketIOClient client;
char host[] = "lanzbc.herokuapp.com";
```

```
17. int port =80;
18. 
19. const char* ssid = "Lanzbc";
20. const char* password = "meomeomeo";
21. 
22. char namespace_esp8266[] = "esp8266";
23. extern String RID;
24. extern String Rfull;
25. 
26. void setup()
27. {
28. 
29.     Serial.begin(57600);
30.     mySerial.begin(57600);
31.     delay(10);
32. 
33.     Serial.print("Ket noi vao mang ");
34.     Serial.println(ssid);
35.     WiFi.begin(ssid, password);
36.     while (WiFi.status() != WL_CONNECTED) {
37.         delay(500);
38.         Serial.print('.');
39.     }
40. 
41.     Serial.println();
42.     Serial.println(F("Da ket noi WiFi"));
43.     Serial.println(F("Di chi IP cua ESP8266 (Socket Client ESP8266): "));
44.     Serial.println(WiFi.localIP());
45. 
46.     if (!client.connect(host, port, namespace_esp8266)) {
47.         Serial.println(F("Ket noi den socket server that bai!"));
48.         return;
49.     }
50.     sCmd.addDefaultHandler(defaultCommand);
51.     Serial.println("Da san sang nhan lenh");
52. }
53. 
54. void loop()
55. {
56.     if (client.monitor()) {
57.         mySerial.print(RID);
58.         mySerial.print('\r');
59.         mySerial.print(Rfull);
60.         mySerial.print('\r');
61. 
62.         uint32_t free = system_get_free_heap_size();
63.         Serial.println(free);
64.     }
```

```
65.     if (!client.connected()) {
66.       client.reconnect(host, port, namespace_esp8266);
67.     }
68.
69.     sCmd.readSerial();
70. }
71.
72. void defaultCommand(String command) {
73.   char *json = sCmd.next();
74.   client.send(command, (String) json);
75. }
```

2.2.3. Lập trình server

Cấu trúc thư mục của server như sau:

- node_modules/ #thư mục chứa các module cần dùng
- index.js #file thực thi tạo socket server
- webapp/ #Thư mục chứa code của webapp

- index.html #file html cài đặt CSS3, JAVASCRPT
- webapp.js #file thực thi tạo socket client và cấu trúc
- home.html #giao diện chương trình điều khiển

index.js

```
1. const PORT = 3484;
2.
3. var http = require('http');
4. var express = require('express');
5. var socketio = require('socket.io')
6.
7. var app = express();

8. var server = http.Server(app)
9.
10. var io = socketio(server);
11.
12. var webapp_nsp = io.of('/webapp')
13. var esp8266_nsp = io.of('/esp8266')
14.
15. var middleware = require('socketio-wildcard')();
16. esp8266_nsp.use(middleware);
```

```
17. webapp_nsp.use(middleware);

18.
19. server.listen(process.env.PORT || 3484);
20.
21. app.use(express.static("node_modules/mobile-angular-ui"))
22. app.use(express.static("node_modules/angular"))
23. app.use(express.static("node_modules/angular-route"))

24. app.use(express.static("node_modules/socket.io-client"))
25. app.use(express.static("node_modules/angular-socket-io"))
26. app.use(express.static("webapp"))

27.
28. function ParseJson(jsondata) {
29.     try {
30.         return JSON.parse(jsondata);
31.     } catch (error) {
32.         return null;
33.     }
34. }
35.
36. esp8266_nsp.on('connection', function(socket) {
37.       console.log('esp8266 connected')
38.
39.       socket.on('disconnect', function() {
40.             console.log("Disconnect socket esp8266")
41.       })
42.
43.       socket.on("*", function(packet) {
44.             console.log("esp8266 rev and send to webapp packet: ",
    packet.data)
45.             var eventName = packet.data[0]
46.             var eventJson = packet.data[1] || {}
47.             webapp_nsp.emit(eventName, eventJson)
48.       })
49. })
50.
51.
52. webapp_nsp.on('connection', function(socket) {
53.
54.       console.log('webapp connected')
55.
56.       socket.on('disconnect', function() {
57.             console.log("Disconnect socket webapp")
58.       })
59.
60.       socket.on('*', function(packet) {
```

```
61.             console.log("webapp rev and send to esp8266 packet: ",
    packet.data)
62.             var eventName = packet.data[0]
63.             var eventJson = packet.data[1] || {}
64.             esp8266_nsp.emit(eventName, eventJson)
65.         });
66. })
```

Home.html

```
1. <div class="scrollable">
2.         <div class="scrollable-content">
3.                 <div class="list-group text-center">
4.                         <div class="list-group-item list-group-item-home">
5.                                 <h1>
6.                                         <font face="Georgia"
    color="009900"><b>TINY GARDEN<b/></font>
7.                                 </h1>
8.                         </div>
9.
10.                        <div class="list-group-item list-group-item-home">
11.                                <div>
12.                                        <img src="/th.png" width="50"
    eight="5h0"> <br/><br/>
13.                                </div>
14.                                <div>
15.                                        <center>
16.                                        <table border="0px" width="300px" >
17.                                    <tr>
18.                                        <td class="home-heading">
19.                                            <center>
20.                                                Độ ẩm
21.                                            </center>
22.                                        </td>
23.                                        <td class="home-heading">
24.                                            <center>
25.                                                Nhiệt độ
26.                                            </center>
27.                                        </td>
28.                                    </tr>
29.                                    <tr>
30.                                        <td>
31.                                            <center>
32.                                            <h3> {{humidity}} </h3>
33.                                            </center>
34.                                        </td>
35.                                        <td>
36.                                        <center>
```

```
37.                                         <h3> {{temperature}} </h3>
38.                                     </center>
39.                                     </td>
40.                                 </tr>
41.                             </table>
42.                             </center>
43.                             </div>
44.                     </div>
45.
46.                     <div class="list-group-item list-group-item-home">
47.                             <div>
48.                                     <img src="/doam.png" width="50"
    height="50"> <br/><br/>
49.                             </div>
50.                             <div>
51.                                     <h4 class="home-heading">Độ ẩm đất</h4>
52.                                     <h3>{{doAmAnalog}}</h3>
53.                                     {{CamBienDoAm}}
54.                             </div>
55.
56.                     </div>
57.
58.                     <div class="list-group-item list-group-item-home">
59.                             <div>
60.                                     <img src="/brightness.png" width="50"
    height="50"> <br/><br/>
61.                             </div>
62.                             <div>
63.                                     <h4 class="home-heading">Độ sáng</h4>
64.                                     <h3>{{camBienAnhSang}}</h3>

65.                             </div>
66.
67.                     </div>
68.                     <div class="list-group-item list-group-item-home">
69.                             <div>
70.                                     <img src="/control.png" width="50"
    height="50"> <br/><br/>
71.                             </div>
72.                             <div>
73.                                     <h3 class="home-heading">Điều khiển</h3>
74.                                     <div ng-repeat="delay_value in
    delays_status  track by $index">
75.                                             {{$index ? "Đèn"  : "Máy bơm"}}
    đang {{delay_value ? "bật" : "tắt"}}
76.                                             <input type="checkbox" ng-true-
    value="1" ng-false-value="0"
```

```
77.                                         ng-change="changeDelay()" ng-
   model="delays_status[$index]" />
78.
79.                                     </div>
80.                                 </div>
81.                             </div>
82.                             <div class="list-group-item list-group-item-home">
83.
84.                                 <div>
85.                                     <button class="button" ng-
   click="updateSensor()">
86.                                         UPDATE
87.                                     </div>
88.                                 </div>
89.                             </div>
90.
91.
92.                 </div>
93.         </div>
94. </div>
```

Index.html

```
1.  <!DOCTYPE html>
2.  <html>
3.          <head>
4.                  <meta charset="utf-8">
5.                  <title>My Tiny garden</title>
6.                  <link rel="stylesheet" href="/dist/css/mobile-angular-ui-
   hover.min.css" />
7.                  <link rel="stylesheet" href="/dist/css/mobile-angular-ui-
   base.min.css" />
8.                  <link rel="stylesheet" href="/dist/css/mobile-angular-ui-
   desktop.min.css" />
9.                  <script src="/angular.min.js"></script>
10.                 <script src="/angular-route.min.js"></script>
11.                 <script src="/dist/js/mobile-angular-ui.min.js"></script>
12.                 <script src="/dist/socket.io.min.js"></script>
13.                 <script src="/socket.min.js"></script>
14.                 <script src="/webapp.js"></script>
15.                 <meta name="viewport" content="initial-scale=1, maximum-
   scale=1, user-scalable=no, width=device-width">
16.         </head>
17.         <body  ng-app="myApp">
18.                 </div>
19.                 <div class="sidebar sidebar-right">
20.                 </div>
21.                 <div class="app">
```

```
                    <div class="navbar navbar-app navbar-absolute-top">
                    </div>
                    <div class="navbar navbar-app navbar-absolute-bottom">
                    </div>
                    <div class='app-body'>
                          <div class='app-content'>
                                <ng-view>
                                </ng-view>
                          </div>
                    </div>
              </div>
              <div ui-yield-to="modals"></div>
        </body>
</html>
```

Webapp.js

```
var myApp= angular.module('myApp', [
    'ngRoute',
    'mobile-angular-ui',
      'btford.socket-io'
]);

myApp.config(function($routeProvider) {
    $routeProvider.when('/', {
        templateUrl: 'home.html',
        controller: 'Home'
    });
});

myApp.factory('mySocket', function (socketFactory) {
      var myIoSocket = io.connect('/webapp');

      mySocket = socketFactory({
            ioSocket: myIoSocket
      });
      return mySocket;

});

myApp.controller('Home', function($scope, mySocket) {
    $scope.humidity = "Chưa kết nối";
    $scope.temperature = "Chưa kết nối";
    $scope.CamBienDoAm = "Chưa kết nối";
    $scope.doAmAnalog = "Chưa kết nối";
    $scope.camBienAnhSang = "Chưa kết nối";
    $scope.delays_status = [0, 0]
      $scope.updateSensor  = function() {
```

```
32.             mySocket.emit("doAm")
33.             mySocket.emit("DHT11")
34.       }
35.
36.       $scope.changeDelay = function() {
37.             console.log("send status ", $scope.delays_status)
38.
39.             var json = {
40.                   "delay": $scope.delays_status
41.             }
42.             mySocket.emit("DELAY", json)
43.       }
44.
45.       mySocket.on('DHT11', function(json) {
46.             $scope.humidity =  json.hum + "%"
47.             $scope.temperature = json.tem + "oC"
48.             $scope.camBienAnhSang= json.camBienAnhSang+" lux"

49.       })
50.
51.       mySocket.on('doAm', function(json) {
52.             $scope.doAmAnalog =  json.analog +"%"
53.             $scope.CamBienDoAm = (json.digital == 1) ? "Cần tưới nước!" :
   "Đủ nước"
54.       })
55.       mySocket.on('DELAY_STATUS', function(json) {
56.             console.log("recv DELAY", json)
57.             $scope.delays_status = json.data
58.       })
59.
60.
61.       mySocket.on('connect', function() {
62.             console.log("connected")
63.             mySocket.emit("doAm")
64.             mySocket.emit("DHT11")
65.             mySocket.emit("DELAY")
66.       })
67.
68. });
```

2.3. Thi công và kết quả

Để thực hiện triển khai chương trình ta thực hiện các bước sau:

- Lắp ráp mạch theo đúng sơ đồ nguyên lý

- Vì một số thư viện ArduinoIDE không có sẵn nên ta cần phải tải xuống và cài đặt những thư viện đã khai báo sử dụng trong project trên mã nguồn mở như github và một số cộng đồng khác.

- Thực hiện khai báo chân của các thiết bị nhận và thực hiện của board arduino để có sự đồng bộ giữa mạch cứng và phần lập trình cho vi mạch điều khiển thì khi nạp code arduino sẽ chạy đúng như những gì mình đã lập trình cho nó.

- Để arduino có thể kết nối với mạng thì việc cần thiết đồng hành đó là kết nối được ESP8266 với mạng LAN để test được trước khi Deploy lên server kết nối với mạng WAN.

- Nạp code cho board mạch qua cổng UART kết nối với 2 board. Do 2 board sử dụng đều có cổng nạp mạch riêng biệt nên ta có thể nạp cùng lúc 2 thiết bị. Và ta có thể nhận được một số tín hiệu đã in ra cổng serial minitor trên phần mềm ArduinoIDE.

- Đồng thời với đó, ta thực hiện xây dựng giao diện cho chương trình để giao tiếp với người dùng. Đối với Nodejs ta viết chương trình trên bất cứ tool soạn thảo nào và lưu với đuôi .js và khởi chạy qua cmd là có thể nối với server localhost tại port mà ta khai báo.

```
1. var http = require("http");
2.
3. http.createServer(function(request, response) {
4.    response.writeHead(200, {"Content-Type": "text/plain"});
5.    response.write("Xin chào thế giới! ");
6.    response.end();
7. }).listen(3000);
```

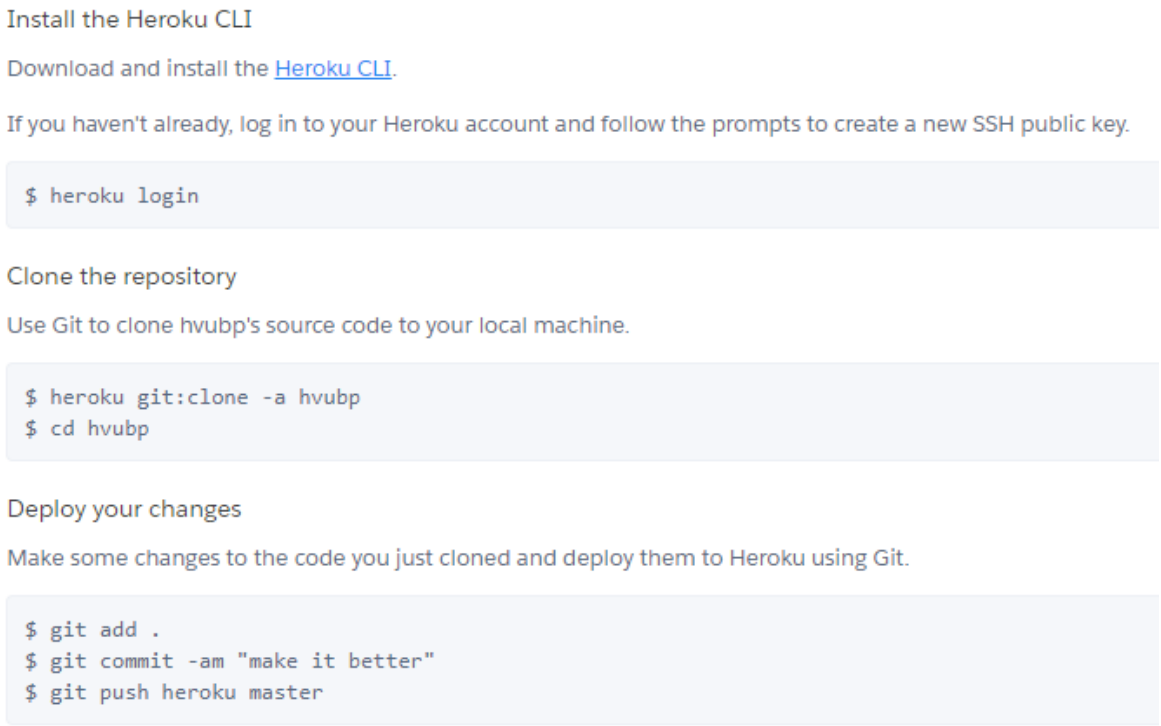
Với 6 dòng lệnh như trên ta có thể bắt đầu lập trình web rồi.

- Để tải thư viện cho chương trình ta chỉ cần sử dụng lệnh “npm install + tên thư viện cần tải”. Tiến hành chạy chương trinh trên localhost với “node index”.

- Sau khi projet đã chạy hoàn chỉnh và ổn định hoạt động tại cục bộ, WWW dành cho “World Wide Web” và chúng ta sẽ biến server cục bộ thành một server toàn thế giới. Chúng ta sẽ sử dụng Heroku cloud application platform cho việc này. Heroku là một nền tảng điện toán đám mây như là một dịch vụ. Nó cho phép triển khai web

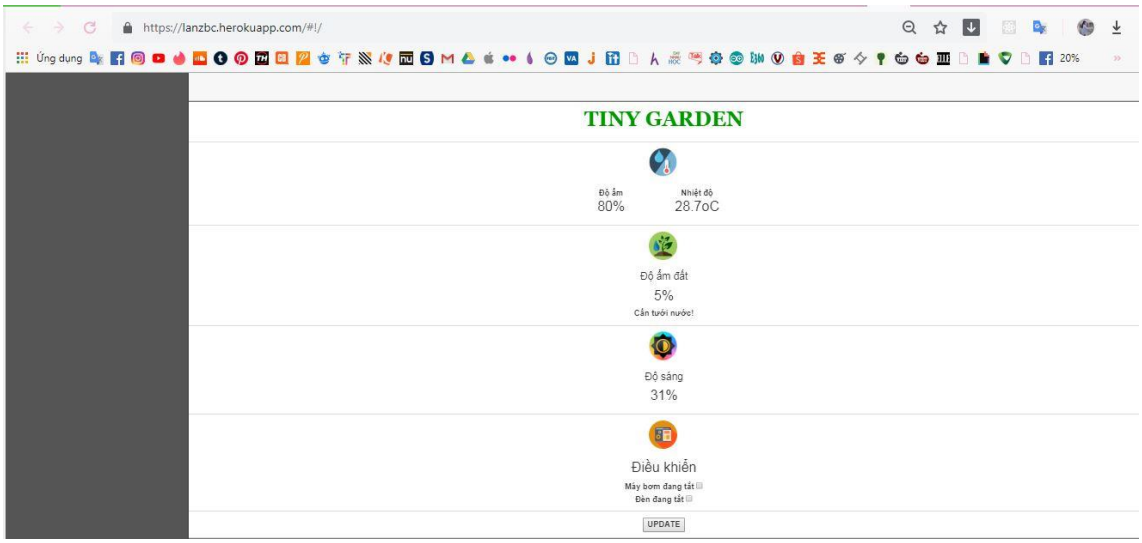
server, vì vậy mọi người có thể sửa dụng app và điều khiển hệ thống "Tiny garden" bằng điện thoại, máy tính có kết nối internet.

Sau khi đăng ký tài khoản trên https://dashboard.heroku.com/apps chọn create new app và làm theo hướng dẫn. Trước khi tải lên nhớ đặt lại địa chỉ để server lắng nghe đồng thời đổi link cho esp có thể truy cập vào trang mình khởi tạo

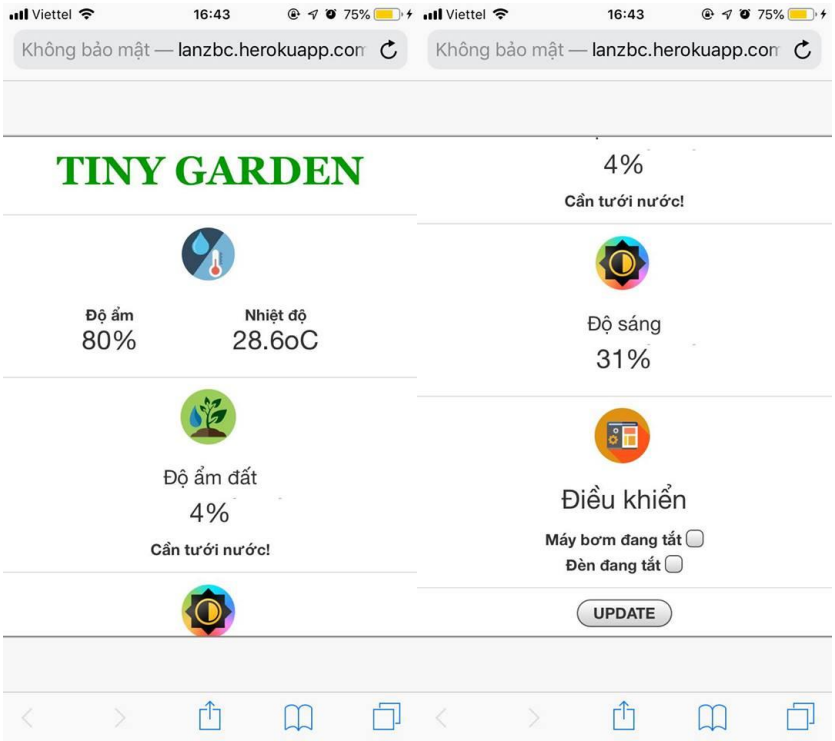


Hình 2.15. Deploy this app use heroku

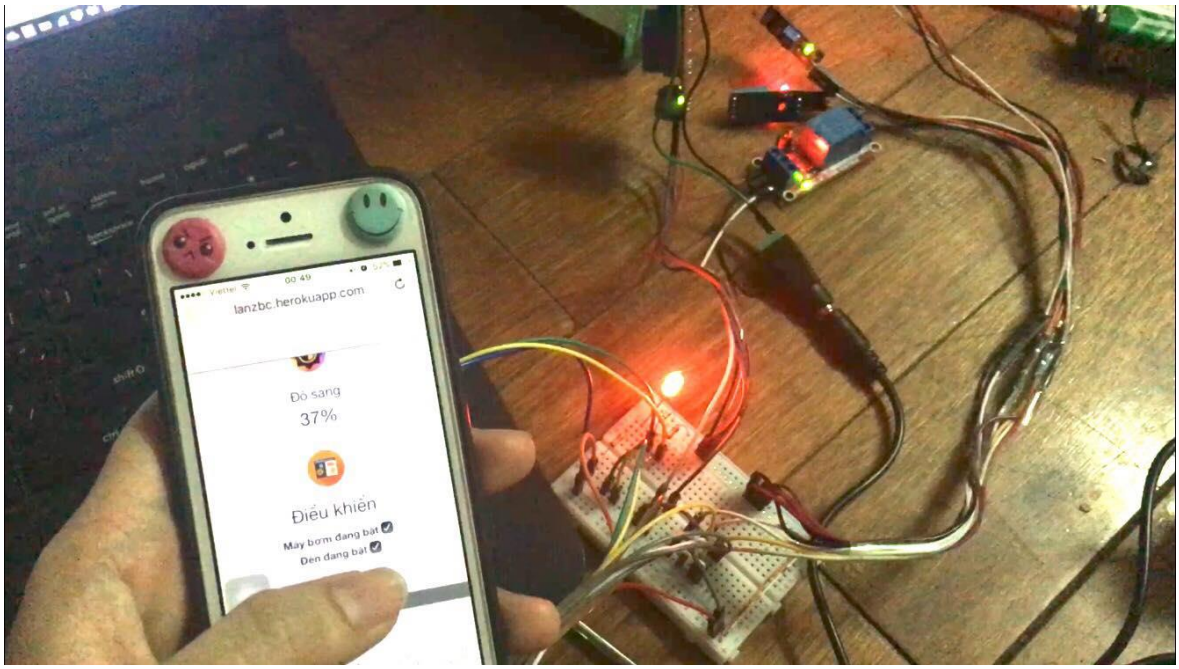
Và đây là kết quả ban đầu của mô hình nông nghiệp thông minh thu nhỏ "Tiny garden":

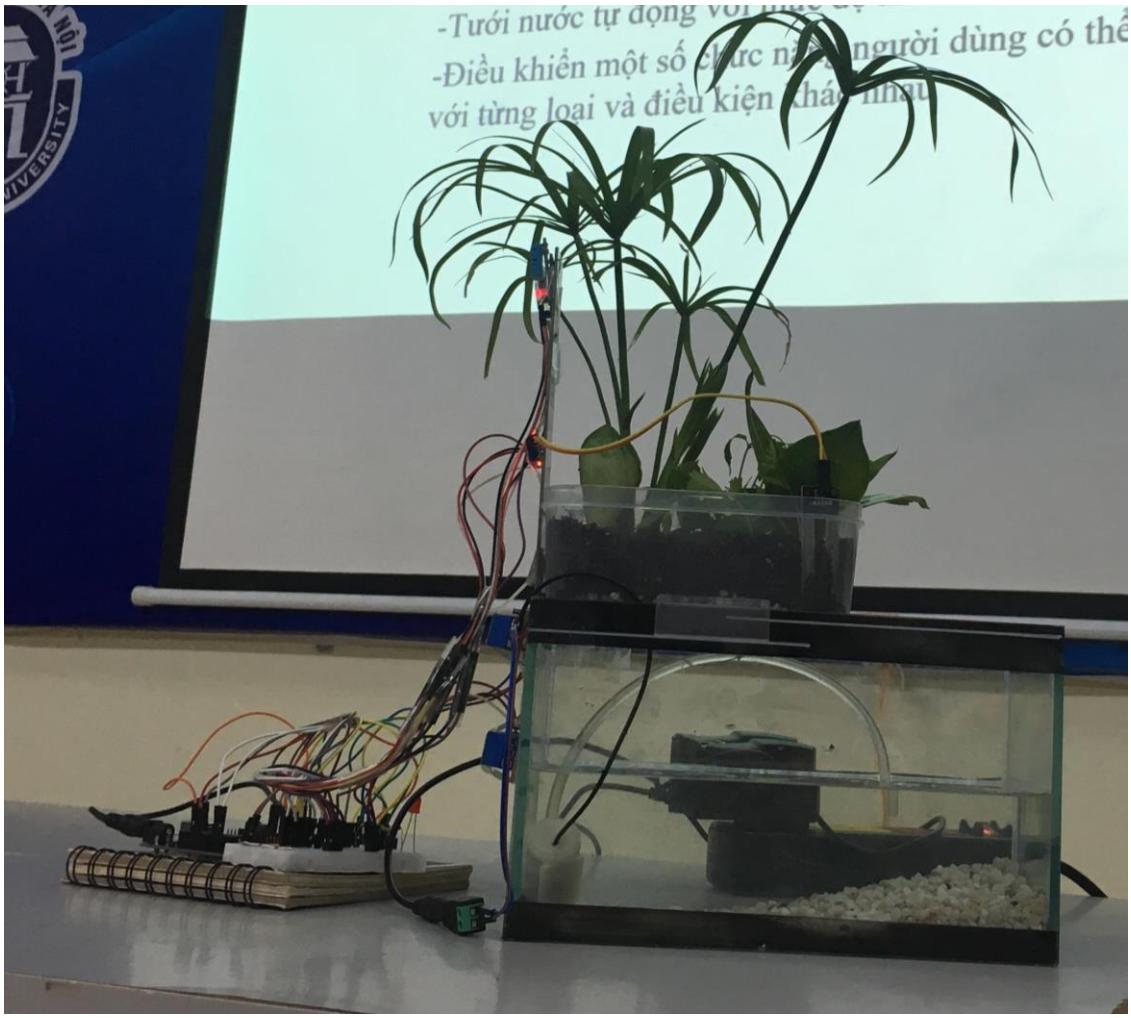


Hình 2.16. Giao diện webapp trên máy tính khi kết nối



Hình 2.17. Giao diện webapp trên điện thoại khi kết nối





Hình 2.18. Chạy mạch thử nghiệm

2.4. Nhận xét đánh giá

Nhìn chung với mô hình sử dụng để học tập và nghiên cứu thì "Tiny garden" đã phần nào thể hiện được những kiến thức cơ bản đã được học ở đại học về Điện tử, lập trình vi mạch, lập trình ứng dụng với một số ngôn ngữ lập trình cơ bản đồng thời với đó là một số kiến thức về mạng. Trong quá trình thực hiện giúp ta hiểu thêm về IoT

Trong thực tế IoT khá rộng và sâu về nhiều mảng khác nhau nên để thực hiện hoàn thiện một cách hoàn hảo là khá khó với kiến thưc và khả năng hiện tại của em. Phải tham khảo từ nhiều nguồn và học tập từ nhiều người, đặc biệt là những cơ sở nơi em đã thực tập. Trên mô hình của em còn một số thiếu xót như: giao diện người dùng chưa được đẹp, mạch chưa được nhỏ gọn và được bảo vệ cẩn thận, chưa có mô hình mô phỏng khu vườn hoàn chỉnh, chưa có bảo mật…

2.5. Hướng phát triển

Với mô hình nhỏ này bước dầu em sẽ hoàn thiện để bản thân có thể sử dụng trong gia đình chăm sóc cho những cây nhỏ của em và nuôi thêm bầy cá nhỏ tạo hệ sinh thái mang đến không gian thiên nhiên. Và điều đặc biệt là khi đi xa vài ngày thì em có thể chăm sóc được cho khu vườn nhỏ này. Với mô hình đó em có thể triển khai thêm một số chức năng nhỏ như cho cá ăn hẹn giờ, báo mực nước trong bể cá…

Chúng ta có thể thêm một số tính năng như đo lượng nước hay mức PH của nước để phát triển điều khiển kiểm soát hệ sinh thái dưới nước. Bên cạnh đó có thể nghiên cứu một số vấn đề sinh học để mô hình có tính khả thi riêng biệt cho từng đối tượng áp dụng. Từ đó tạo ra một vòng lặp, một hệ sinh thái an toàn, thông minh, bảo vệ môi trường.

Đối với mô hình lớn hơn quy mô nhà vườn trang trại, chúng em sẽ xây dựng các node nhỏ gửi giữ liệu về trạm trung tâm để xử lýĐể mử rộng quy mô cần phải nghiên cứu về điều kiện thực tế nhiều hơn để áp dụng một cách kinh tế nhất, đem lại hiệu quả thực sự cho người làm vườn và người làm nông nghiệp thủy hải sản.

DANH MỤC TÀI LIỆU THAM KHẢO

1. wikipedia
   https://www.wikipedia.org/
2. Internet Of Things (IoT) : cho người mới bắt đầu
   https://iotmakervn.github.io/iot-starter-book/
3. Ksp - Cộng đồng Arduino Việt Nam
   http://arduino.vn/users/ksp
4. Hướng dẫn ESP8266
   https://www.youtube.com/playlist?list=PL14WBbXTfR-46Io9ZkdpImV9Ozrct6YPk
5. Học Lập Trình Website từ A đến Z
   https://www.youtube.com/playlist?list=PLLAJJPGNwNkghoNSB9xq22EJ_Z1rX7Ygs
6. [Node JS Level 01 - Hoàn thành] Khóa học Node JS cơ bản | Node JS Tutorials For Beginners
   https://www.youtube.com/playlist?list=PLb9GOiDj30nSJLKeai7yWUtl7SD2cTTGn
7. AngularJS căn bản cho người mới bắt đầu
   https://www.youtube.com/playlist?list=PLRhlTlpDUWsw70vZAkJgALJ1yhgYsqDGx
8. Techtalk – Xu hướng công nghệ
   https://techtalk.vn/
9. vietjack
   https://vietjack.com/index.jsp