**LỜI CẢM ƠN**

Trong suốt quá trình hoàn thành khóa luận tốt nghiệp, tôi đã nhận được nhiều sự giúp đỡ từ gia đình, thầy cô, anh chị, bạn bè, và mọi người thân thương luôn sát cánh bên tôi trong thời gian vừa qua.

Đầu tiên, tôi xin gửi lời cảm ơn của mình đến nhà trường, khoa Vật Lý, bộ môn Vật Lý Tin học, vì đã tạo cho tôi một môi trường học tập thật tốt. Xin chân thành cảm ơn các Thầy Cô bộ môn Vật Lý Tin Học đã tận tình chỉ bảo, dạy dỗ truyền đạt những kinh nghiệm, kiến thức quý báu trong suốt quá trình học tập và phấn đấu tại trường, điều đó đã tạo cho tôi có những hành trang để bước vào đời.

ThS Nguyễn Anh Huy, Cô Nguyễn Vương Thùy Ngân – bộ môn Vật Lý Tin Học, Khoa Vật Lý, Trường Đại Học Khoa Học Tự Nhiên, là giảng viên trực tiếp hướng dẫn và tận tình chỉ dạy những kiến thức rất bổ ích để tôi hoàn thành tốt khóa luận tốt nghiệp này.

Và hơn hết, xin gửi lời cảm ơn đến gia đình, bạn bè đã giúp đỡ cho tôi những khi khó khăn, động viên tôi những khi suy sụp, chia sẻ cùng tôi những niềm vui và nỗi buồn.

Xin chân thành cảm ơn đến các Thầy Cô trong hội đồng chấm khóa luận đã cho tôi những đóng góp quý báu để hoàn chỉnh khóa luận này.

Xin cảm ơn các bạn trong lớp Vật Lý Tin Học K12 đã giúp đỡ trong suốt quá trình học tập và làm khóa luận.

Xin chân thành cảm ơn!

TP.Hồ Chí Minh, tháng 6 năm 2016

Nguyễn Toàn Thắng

# **MỤC LỤC**

[**MỤC LỤC** ii](#_Toc454635683)

[**DANH MỤC CÁC TỪ VIẾT TẮT** iv](#_Toc454635684)

[**DANH SÁCH CÁC HÌNH** v](#_Toc454635685)

[**DANH SÁCH CÁC BẢNG** vii](#_Toc454635686)

[**Lời mở đầu** 1](#_Toc454635687)

[**Chương 1 GIỚI THIỆU ĐỀ TÀI** 3](#_Toc454635688)

[**1.1 Đặt vấn đề** 3](#_Toc454635689)

[**1.2 Giới thiệu đề tài** 4](#_Toc454635690)

[**Chương 2 TỔNG QUAN VỀ INTERNET OF THINGS** 5](#_Toc454635691)

[**1.1 Giới thiệu Internet of things (IoT)** 5](#_Toc454635692)

[***1.1.1 Internet of thing là gì*** 5](#_Toc454635693)

[***1.1.2 Xu hướng và tính chất của Internet of thing*** 6](#_Toc454635694)

[***1.1.3 Ứng dụng của Internet of thing*** 8](#_Toc454635695)

[***1.1.4 Những tác nhân ngăn cản sự phát triển của Internet of thing*** 9](#_Toc454635696)

[**1.2 Yêu cầu của một hệ thống Internet of thing** 11](#_Toc454635697)

[**1.3 Cấu trúc của một hệ thống Internet of thing** 12](#_Toc454635698)

[**CHƯƠNG 3 GIAO THỨC MQTT** 16](#_Toc454635699)

[**3.1 Giới thiệu giao thức MQTT** 16](#_Toc454635700)

[**3.2 Mô hình hoạt động của MQTT** 16](#_Toc454635701)

[**3.3 QoS (Quality of service)** 18](#_Toc454635702)

[**3.4 Các gói dữ liệu cơ bản của MQTT** 21](#_Toc454635703)

[***3.4.1 Gói Connection*** 21](#_Toc454635704)

[***3.4.2 Gói Connack*** 22](#_Toc454635705)

[***3.4.3 Gói Publish*** 23](#_Toc454635706)

[***3.4.4 Gói Subscribe*** 24](#_Toc454635707)

[***3.4.5 Gói Suback*** 25](#_Toc454635708)

[***3.4.6 Gói Unsubscribe*** 26](#_Toc454635709)

[***3.4.7 Gói Unsuback*** 27](#_Toc454635710)

[**Chương 4 CÁC MODULE TRONG HỆ THỐNG** 28](#_Toc454635711)

[**4.1 Giới thiệu module ESP8266** 28](#_Toc454635712)

[***4.1.1 Phần mềm lập trình cho ESP8266*** 29](#_Toc454635713)

[***4.1.2 Cấu hình phần mềm để lập trình cho ESP8266*** 30](#_Toc454635714)

[***4.1.3 Các thư viện cần thiết*** 30](#_Toc454635715)

[***4.1.4 Nạp firmware cho module ESP8266*** 31](#_Toc454635716)

[**4.2 Module điều khiển thông qua hồng ngoại** 33](#_Toc454635717)

[***4.2.1 Sơ đồ nguyên lý*** 34](#_Toc454635718)

[***4.2.2 Hoạt động*** 35](#_Toc454635719)

[***4.2.3 Thư viện dùng cho module*** 35](#_Toc454635720)

[**4.3 Module cảm biến độ ẩm nhiệt độ** 36](#_Toc454635721)

[***4.3.1 Sơ đồ nguyên lí*** 36](#_Toc454635722)

[***4.3.2 Hoạt động*** 37](#_Toc454635723)

[***4.3.3 Các thư viện cần thiết*** 38](#_Toc454635724)

[**Chương 5 PHẦN MỀM TRÊN DI ĐỘNG** 40](#_Toc454635725)

[**5.1 Giới thiệu hệ điều hành di động IOS** 40](#_Toc454635726)

[**5.2 Công cụ lập trình và debug** 41](#_Toc454635727)

[***5.2.1 Công cụ lập trình Xcode*** 41](#_Toc454635728)

[***5.2.2 Công cụ debug Instrument*** 42](#_Toc454635729)

[**5.3 Ngôn ngữ lập trình Swift** 42](#_Toc454635730)

[**5.4 Hoạt động của ứng dụng** 45](#_Toc454635731)

[**5.5 Giao diện** 46](#_Toc454635732)

[**KẾT LUẬN** 48](#_Toc454635733)

[**TÀI LIỆU THAM KHẢO** 49](#_Toc454635734)

# **DANH MỤC CÁC TỪ VIẾT TẮT**

IOT Internet of Thing

ID Identifier

MQTT Message Queue Telemetry Transport

QoS Quality of Service

GPIO Genaral Purpose Input Output

UART universal asynchronous receiver/transmitter

# **DANH SÁCH CÁC HÌNH**

Hình 2.1 Xe đạp thông minh, một ứng dụng của IOT

Hình 2.2 Cấu trúc của hệ thống IOT

Hình 3.1 Mô hình Publish/Subscribe

Hình 3.2 Publish với QoS 0

Hình 3.3 Publish với QoS 1

Hình 3.4 Publish với QoS 2

Hình 3.5 Gói dữ liệu connect

Hình 3.6 Gói dữ liệu connack trả về khi client connect

Hình 3.7 Gói dữ liệu publish của client

Hình 3.8 Gói dữ liệu subscribe của client

Hình 3.9 Gói dữ liệu suback trả về khi client subscribe

Hình 3.10 Gói dữ liệu unsubscribe của client

Hình 3.11 Gói dữ liệu unsuback trả về khi client unsubscribe

Hình 4.1 Chip ESP8266

Hình 4.2 Giao diện phần mềm Arduino

Hình 4.3 Giao diện board manager của arduino

Hình 4.4 Giao diện cài đặt thư viện arduino

Hình 4.5 Module USB - UART

Hình 4.6 Cách thiết lập cài đặt board

Hình 4.7 Module điều khiển hồng ngoại

Hình 4.8 Sơ đồ nguyên lý module hồng ngoại

Hình 4.9 Module cảm biến nhiệt độ độ ẩm

Hình 4.10 Sơ đồ nguyên lý module cảm biến

Hình 4.11 Cảm biến nhiệt độ, độ ẩm DHT22

Hình 4.12 Sơ đồ chuẩn giao tiếp One Wire

Hình 5.1 Các phiên bản hệ điều hành IOS

**DANH SÁCH CÁC HÌNH**

Hình 5.2 Giao diện XCode

Hình 5.3 Giao diện Instrument

Hình 5.4 Sơ đồ use case của ứng dụng

Hình 5.5 Giao diện dash board của ứng dụng

Hình 5.6 Giao diện quản lý thiết bị

Hình 5.7 Giao diện Setting

Hình 5.8 Giao diện chức năng Add Scene

# **DANH SÁCH CÁC BẢNG**

Bảng 3.1 Bảng kết quả trả về khi client kết nối tới broker

Bảng 3.2: Bảng kết quả trả về khi client subscribe

Bảng 4.1 Cách nối dây giữa module UART và ESP8266

# 

# **Lời mở đầu**

“Ngôi nhà thông minh” hay còn gọi là smarthome là một cụm từ không còn xa lạ đối với nền công nghệ phát triển hiện nay. Lúc đầu, ý tưởng được thực hiện nhờ vào tia hồng ngoại để điều khiển từ xa, nhưng khoảng cách là hạn chế. Về sau, nhiều nghiên cứu nhằm cải thiện khoảng cách điều khiển mang lại nhiều thành công và có ý nghĩa thực tiễn như điều khiển thông qua đường dây điện thoại.

Khi công nghệ wireless phát triển, người ta lại nghĩ đến điều khiển qua mạng không dây, điều khiển từ xa dùng máy tính ra đời. Không dừng lại ở đó, khi mà các mạng điện thoại đang cạnh tranh gay gắt, chiếc điện thoại trở nên vật dùng không thể thiếu với mỗi cá nhân, người ta lại nghĩ về một chiếc điện thoại tích hợp khả năng điều khiển từ xa.

Đặc biệt, với sự phát triển chóng mặt của SmartPhone và công nghệ 3G hiện nay, việc tích hợp các chức năng này vào SmartPhone đang trở thành một giải pháp tối ưu và mang lại nhiều ưu điểm.

Đi cùng xu hướng đó, bài viết này giới thiệu một giải phát điều khiển và giám sát ngôi nhà một cách thông minh thông qua internet và có thể sử dụng điện thoại để giám sát các thiết bị, báo cháy, báo trộm. Giải pháp này được đưa ra rất khả khi với cơ chết hoạt động chính xác và mang tính ổn định để tạo bước phát triển một thiết bị nhỏ gọn tham gia một mảng của nhà thông minh giá rẻ. Vì vậy đề tài “Xây dựng bộ điều khiển nhà thông minh” được chọn nhằm mục đích tạo ra một hệ thống điều khiển nhà thông minh giá rẻ bao gồm các module nhỏ gọn và quản lý bằng một thiết bị di động. Hệ thống có các thành phần sau:

* Các module điều khiển các thiết bị bằng hồng ngoại.
* Một module cảm biến nhiệt độ, độ ẩm.
* Một web server giao tiếp và lưu dữ liệu người dùng.
* Một phần mềm chạy trên nền tảng di động.

Hệ thống có các chức năng chính sau:

* Module điều khiển hồng ngoại có thể điều khiển các thiết bị có hỗ trợ giao tiếp hồng ngoại như TV, quạt điện, máy nghe nhạc, đầu DVD…
* Module cảm biến ghi lại nhiệt độ và độ ẩm lưu lại trên webserver để người dùng theo dõi.
* Tất cả các module sẽ được quản lý bởi một phần mềm trên di động.

Nội dung của khóa luận được phân thành các phần sau:

CHƯƠNG 1 GIỚI THIỆU ĐỀ TÀI

CHƯƠNG 2 TỔNG QUAN VỀ INTERNET OF THING

CHƯƠNG 3 GIỚI THIỆU GIAO THỨC MQTT

CHƯƠNG 4 GIỚI THIỆU CÁC MODULE SỬ DỤNG TRONG HỆ THỐNG

CHƯƠNG 5 PHẦN MỀM TRÊN DI ĐỘNG

Và phần KẾT LUẬN

# 

# **Chương 1 GIỚI THIỆU ĐỀ TÀI**

## **1.1 Đặt vấn đề**

Ngày nay cùng với sự phát triển mạnh mẽ của các ngành khoa học kỹ thuật, công nghệ kỹ thuật điện tử mà trong đó là kỹ thuật tự động điều khiển đóng vai trò quan trọng trong mọi lĩnh vực khoa học kỹ thuật, quản lí, công nghiệp, cung cấp thông tin ... Do đó, chúng ta phải biết nắm bắt và vận dụng nó một cách có hiệu quả nhằm góp phần vào sự phát triển nền khoa học kỹ thuật thế giới nói chung và trong sự phát triển kỹ thuật điện tử nói riêng.

Bên cạnh đó còn là sự thúc đẩy sự phát triển của nền kinh tế nước nhà. Như chúng ta cũng đã biết, gần như các thiết bị trong đời sống của các gia đình ngày nay đều hoạt động độc lập với nhau, mỗi thiết bị có một quy trình sử dụng khác nhau tuỳ thuộc vào sự thiết lập, cài đặt của người sử dụng. Chúng chưa có một sự liên kết nào với nhau về mặt dữ liệu. Nhưng đối với hệ thống điều khiển thiết bị từ xa thông qua mạng Internet thì lại khác. Ở đây, các thiết bị điều khiển tự động được kết nối với nhau thành một hệ thống hoàn chỉnh qua một một thiết bị trung tâm và có thể giao tiếp với nhau về mặt dữ liệu. Điển hình của một hệ thống điều khiển thiết bị trong nhà từ xa thông qua mạng Ethernet gồm có các thiết bị đơn giản như bóng đèn, quạt máy, lò sưởi đến các thiết bị tinh vi, phức tạp như tivi, máy giặt, hệ thống báo động … Nó hoạt động như một ngôi nhà thông minh. Nghĩa là tất cả các thiết bị này có thể giao tiếp với nhau về mặt dữ liệu thông qua một đầu não trung tâm. Đầu não trung tâm ở đây có thể là một máy vi tính hoàn chỉnh hoặc có thể là một bộ xử lí đã được lập trình sẵn tất cả các chương trình điều khiển. Bình thường, các thiết bị trong ngồi nhà này có thể được điều khiển từ xa thông qua mạng Internet của chủ nhà.

Chẳng hạn như việc tắt quạt, đèn điện … khi người chủ nhà quên chưa tắt trước khi ra khỏi nhà. Hay chỉ với một thao tác kích, người chủ nhà có thể bật máy điều hòa để làm mát phòng trước khi về nhà trong một khoảng thời gian nhất định. Bên cạnh đó nó cũng gửi thông báo cho người điều khiển biết nhiệt độ trong phòng hiện tại là bao nhiêu, đồng thời phát tín hiệu cảnh báo khi nhiệt độ phòng vượt quá giới hạn cho phép.

Từ những yêu cầu thực tế, những đòi hỏi ngày càng cao của cuộc sống, cộng với sự hợp tác, phát triển mạnh mẽ của mảng di động nên đề tài này đã giải quyết được các phần trên một cách cơ bản nhất.

## **1.2 Giới thiệu đề tài**

Trong phần này chúng tôi sẽ trình bày sơ lược hệ thống điều khiển này, các phần sau sẽ đi vào chi tiết.

Hệ thống bao gồm 3 phần:

* Phần điều khiển, cảm biến, báo động, hay còn được gọi là module. Các module này sẽ được phân làm 2 loại là module điều khiển và module cảm biến. Trong khuôn khổ đề tài này mỗi loại chúng tôi sẽ trình bày một module là module điều khiển bằng hồng ngoại và module cảm biến nhiệt độ, độ ẩm.
* Phần thứ 2 là một web server dùng để lưu dữ liệu người dùng và giao tiếp với các thiết bị thông qua giao thức MQTT. Trong khuôn khổ đề tài này chúng tôi sử dụng một số web server có sẵn miễn phí trên internet.
* Phần cuối cùng là một app trên di đông, trên nền tảng IOS của công ty Apple (cụ thể trong đề tài này chúng tôi viết trên một thiết bị duy nhất là iphone), và được viết bằng ngôn ngữ Swift, một ngôn ngữ mới do công ty Apple sáng tạo. Và app này cũng sẽ giao tiếp vơi web server bằng giao thức MQTT.

Cả 3 phần này giao tiếp với nhau tạo nên hệ thống điều khiển nhà thông minh quản lý trên một thiết bị di động.

# **Chương 2 TỔNG QUAN VỀ INTERNET OF THINGS**

## **1.1 Giới thiệu Internet of things (IoT)**

### ***1.1.1 Internet of thing là gì***

Internet of thing (IoT) là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết (identifiable) cũng như chỉ sự tồn tại của chúng trong một kiến trúc mang tính kết nối. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT, nơi thiết lập các quy chuẩn toàn cầu cho RFID (một phương thức giao tiếp không dây dùng sóng radio) cũng như một số loại cảm biến khác. IoT sau đó cũng được dùng nhiều trong các ấn phẩm đến từ các hãng và nhà phân tích.

Vào tháng 6 năm 2009, Ashton từng cho biết rằng "hiện nay máy tính và Internet gần như phụ thuộc hoàn toàn vào con người để chuyển tải dữ liệu. Gần như tất cả trong số 50 petabyte dữ liệu đang có trên Internet (vào thời điểm đó) đều được ghi lại hoặc tạo ra bởi con người chúng ta, thông qua các các thức như gõ chữ, nhấn nút, chụp ảnh, quét mã vách...". Con người chính là nhân tố quyết định trong thế giới Internet hiện nay. Thế nhưng con người lại có nhiều nhược điểm: chúng ta chỉ có thời gian hạn chế, khả năng tập trung và độ chính xác cũng ở mức thấp so với máy móc. Điều đó có nghĩa là chúng ta không giỏi trong việc thu thập thông tin về thế giới xung quanh, và đây là một vấn đề lớn.

Ví dụ đơn giản như sau: chiếc tủ lạnh thông thường của bạn không được kết nối với thiết bị nào khác. Nếu chúng ta muốn ghi lại nhiệt độ ở từng thời điểm của tủ, chúng ta chỉ có cách ghi lại thủ công rồi nhập vào một máy tính hay thiết bị lưu trữ nào đó. Hay như bóng đèn neon ở nhà chẳng hạn, chúng ta muốn thu thập, điều chỉnh độ sáng của nó thì phải đo thủ công rồi ghi lại.

Còn nếu như máy tính có khả năng giúp con người thu thập tất cả những dữ liệu về mọi thứ xung quanh, chúng ta có thể "theo dõi và đếm mọi thứ, giúp giảm hao phí, chi phí và lỗ. Chúng ta sẽ biết chính xác khi nào các vật dụng cần phải sửa chữa, thay thế, khi nào chúng còn mới và khi nào thì chúng hết hạn sử dụng. Chưa kể đến việc chúng ta có thể kiểm soát chúng mọi lúc mọi nơi. IoT có tiềm năng thay đổi thế giới, giống như cách mà Internet đã thay đổi cuộc sống của chúng ta. Ngôi nhà thông minh với các bóng đèn thông minh, máy giặt thông minh, tủ lạnh thông minh,... có thể xem là bước đầu của IoT bởi chúng đều được liên kết với nhau và/hoặc liên kết vào Internet.

Một chi nhánh của Auto-ID tại Châu Âu từng nói về IoT như sau: "Chúng tôi có một tầm nhìn rất rõ ràng - tạo ra một thế giới nơi mà mọi thứ - từ những chiếc máy bay phản lực khổng lồ cho đến từng cây kim khâu - đều được kết nối vào Internet. Mục tiêu này chỉ có thể đạt được khi và chỉ khi tất cả mọi người áp dụng nó ở tất cả mọi nơi".

Việc trang bị những công nghệ theo dõi, nhận biết vào những vật thông dụng trong đời sống sẽ làm thay đổi rất nhiều cách chúng ta tương tác với đồ vật cũng như cách tương tác giữa người với người.

### ***1.1.2 Xu hướng và tính chất của Internet of thing***

**Khả năng định danh độc nhất:**

Điểm quan trọng của IoT đó là các đối tượng phải có thể được nhận biết và định dạng (identifiable). Nếu mọi đội tượng, kể cả con người, được "đánh dấu" để phân biệt bản thân đối tượng đó với những thứ xung quanh thì chúng ta có thể hoàn toàn quản lí được nó thông qua máy tính. Việc đánh dấu (tagging) có thể được thực hiện thông qua nhiều công nghệ, chẳng hạn như RFID, NFC, mã vạch, mã QR, watermark kĩ thuật số...

Việc kết nối thì có thể thực hiện qua Wi-Fi, mạng viễn thông băng rộng (3G, 4G), Bluetooth, ZigBee, hồng ngoại... Ngoài những kĩ thuật nói trên, nếu nhìn từ thế giới web, chúng ta có thể sử dụng các địa chỉ độc nhất để xác định từng vật, chẳng hạn như địa chỉ IP. Mỗi thiết bị sẽ có một IP riêng biệt không nhầm lẫn. Sự xuất hiện của IPv6 với không gian địa chỉ cực kì rộng lớn sẽ giúp mọi thứ có thể dễ dàng kết nối vào Internet cũng như kết nối với nhau.

**Thông minh:**

Sự thông minh và tự động trong điều khiển thực chất không phải là một phần trong ý tưởng về IoT. Các máy móc có thể dễ dàng nhận biết và phản hồi lại môi trường xung quanh (ambient intelligence), chúng cũng có thể tự điều khiển bản thân (autonomous control) mà không cần đến kết nối mạng. Tuy nhiên, trong thời gian gần đây người ta bắt đầu nghiên cứu kết hợp hai khái niệm IoT và autonomous control lại với nhau. Tương lai của IoT có thể là một mạng lưới các thực thể thông minh có khả năng tự tổ chức và hoạt động riêng lẻ tùy theo tình huống, môi trường, đồng thời chúng cũng có thể liên lạc với nhau để trao đổi thông tin, dữ liệu.

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 dấu vết điện tử của con người khi chúng ta tương tác với những thứ thông minh, từ đó phát hiện ra các tri thức mới liên quan tới cuộc sống, môi trường, các mối tương tác xã hội cũng như hành vi con người.

**Kiến trúc dựa trên sự kiện:**

Các thực thể, máy móc trong IoT sẽ phản hồi dựa theo các sự kiện diễn ra trong lúc chúng hoạt động theo thời gian thực. Một số nhà nghiên cứu từng nói rằng một mạng lưới các sensor chính là một thành phần đơn giản của IoT.

**Là một hệ thống phức tạp:**

Trong một thế giới mở, IoT sẽ mang tính chất phức tạp bởi nó bao gồm một lượng lớn các đường liên kết giữa những thiết bị, máy móc, dịch vụ với nhau, ngoài ra còn bởi khả năng thêm vào các nhân tốc mới.

**Kích thước:**

Một mạng lưới IoT có thể chứa đến 50 đến 100 nghìn tỉ đối tượng được kết nối và mạng lưới này có thể theo dõi sự di chuyển của từng đối tượng. Một con người sống trong thành thị có thể bị bao bọc xung quanh bởi 1000 đến 5000 đối tượng có khả năng theo dõi.

**Vấn đề không, thời gian:**

Trong IoT, vị trí địa lý chính xác của một vật nào đó là rất quan trọng. Hiện nay, Internet chủ yếu được sử dụng để quản lí thông tin được xử lý bởi con người. Do đó những thông tin như địa điểm, thời gian, không gian của đối tượng không mấy quan trọng bởi người xử lí thông tin có thể quyết định các thông tin này có cần thiết hay không, và nếu cần thì họ có thể bổ sung thêm. Trong khi đó, IoT về lý thuyết sẽ thu thập rất nhiều dữ liệu, trong đó có thể có dữ liệu thừa về địa điểm, và việc xử lí dữ liệu đó được xem như không hiệu quả. Ngoài ra, việc xử lí một khối lượng lớn dữ liệu trong thời gian ngắn đủ để đáp ứng cho hoạt động của các đối tượng cũng là một thác thức hiện nay.

**Các hệ thống phụ thuộc Internet of thing:**

Không phải tất cả mọi thứ nằm trong IoT đều nhất thiết phải kết nối vào một mạng lưới toàn cầu, chúng ta có thể hoạt động trong từng hệ thống đơn lẻ (subsystem). Hãy tưởng tượng đến một căn nhà thông minh, trong đó các đồ điện gia dụng có thể tự chúng tương tác với nhau và hoạt động mà không cần phải vào Internet, trừ khi chúng ta cần điều khiển nó từ xa. Ngôi nhà này có thể được xem là một subsystem. Cũng giống như hiện nay chúng ta có các mạng LAN, WAN, mạng ngang hàng nội bộ chứ không kết nối trực tiếp vào Internet.

### ***1.1.3 Ứng dụng của Internet of thing***

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

* Quản lý chất thải.
* Quản lý và lập kế hoạch quản lý đô thị.
* Quản lí môi trường.
* Phản hồi trong các tinh huống khẩn cấp.
* Mua sắm thông minh.
* Quản lí 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 lí 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.

Hình 2.1 Xe đạp thông minh, một ứng dụng của IOT

### ***1.1.4 Những tác nhân ngăn cản sự phát triển của Internet of thing***

**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. Ví dụ, bạn có thể đi từ Việt Nam đến Mỹ, nhưng không đảm bảo rằng bạn có thể nói chuyện tới 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, vâng vâng và vâng vâng.

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 phiên 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 thèm (và 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 lí. Cách này cũng vẫn ổn thôi, những thiết bị vẫn hoàn toàn nói 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 vản 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 được với nhau.

**Tiền và chi phí:**

Cách duy nhất để các thiết bị IOT có thể thật sự giao tiếp đó là khi có một động lực kinh tế để 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 lí 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.

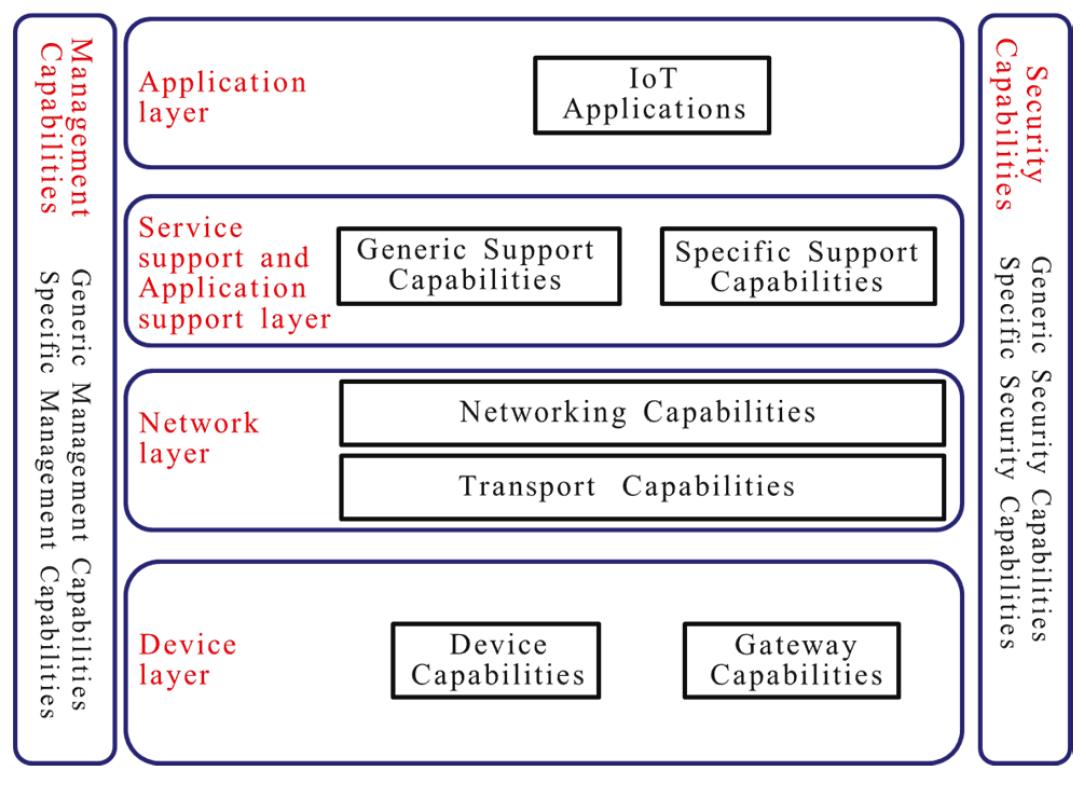
## **1.2 Yêu cầu của một hệ thống Internet of thing**

Một hệ thống Internet of thing phải thỏa mãn những yêu cầu sau:

* Kết nối dựa trên sự nhận diện: Nghĩa là các thiết bị phải có ID riêng biệt. Hệ thống IOT cần hỗ trợ các kết nối giữa các thiết bị, và kết nối được thiết lập dựa trên định danh ID của thiết bị.
* Khả năng cộng tác: hệ thống IOT khả năng tương tác qua lại giữa network và thiết bị.
* Khả năng tự quản của network: Bao gồm tự quản lý, tự cấu hình, tự chữa bệnh, tự tối ưu hóa và tự có cơ chế bảo vệ. Điều này cần thiết để network có thể thích
* ứng với các domains ứng dụng khác nhau, môi trường truyền thông khác nhau, và nhiều loại thiết bị khác nhau.
* Dịch vụ thoả thuận: dịch vụ này để có thể được cung cấp bằng cách thu thập, giao tiếp và xử lý tự động các dữ liệu giữa các thiết bị dựa trên các quy tắc(rules) được thiết lập bởi người vận hành hoặc tùy chỉnh bởi các người dùng.
* Các Khả năng dựa vào vị trí(location-based capabilities): Thông tin liên lạc và các dịch vụ liên quan đến một cái gì đó sẽ phụ thuộc vào thông tin vị trí của Things và người sử dụng. Hệ thống IOT có thể biết và theo dõi vị trí một cách tự động.
* Bảo mật: Trong IOT, nhiều thiết bị được kết nối với nhau. Chình điều này làm tăng mối nguy trong bảo mật, chẳng hạn như bí mật thông tin bị tiết lộ, xác thực sai, hay dữ liệu bị thay đổi hay làm giả.
* Bảo vệ tính riêng tư: tất cả các thiết bị đều có chủ sở hữu và người sử dụng của nó. Dữ liệu thu thập được từ các thiết bị có thể chứa thông tin cá nhân liên quan chủ sở hữu hoặc người sử dụng nó. Các hệ thống IOT cần bảo vệ sự riêng tư trong quá trình truyền dữ liệu, tập hợp, lưu trữ, khai thác và xử lý. Bảo vệ sự riêng tư không nên thiết lập một rào cản đối với xác thực nguồn dữ liệu.
* Plug and play: các Things phải được plug-and-play một cách dễ dàng và tiện dụng.
* Khả năng quản lý: hệ thống IOT cần phải hỗ trợ tính năng quản lý các thiết bị để đảm bảo network hoạt động bình thường. Ứng dụng IOT thường làm việc tự động mà không cần sự tham gia người, nhưng toàn bộ quá trình hoạt động của họ nên được quản lý bởi các bên liên quan.

## **1.3 Cấu trúc của một hệ thống Internet of thing**

Bất kì hệ thống Internet of thing nào cũng được xây dựng từ 4 hệ thống layer sau:

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

Hình 2.2 Cấu trúc của hệ thống IOT

**Lớp ứng dụng:**

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

**Lớp hỗ trợ dịch vụ và hỗ trợ ứng dụng:**

* Nhóm dịch vụ chung: Các dịch vụ hỗ trợ chung, phổ biến mà hầu hết các ứng dụng IOT đều cần, ví dụ như xử lý dữ liệu hoặc lưu trữ dữ liệu.
* Nhóm dịch vụ cụ thể, riêng biệt: Những ứng dụng IOT khác nhau sẽ có nhóm dịch phụ hỗ trợ khác nhau và đặc thù. Trong thực tế, ví dụ trong SmartFarming, nhóm dịch vụ cụ thể riêng biệt là tính toán độ tăng trưởng của cây mà đưa ra quyết định tưới nước hoặc bón phân.

**Lớp mạng:**

* Chức năng Networking: cung cấp chức năng điều khiển các kết nối kết nối mạng, chẳng hạn như tiếp cận được nguồn tài nguyên thông tin và chuyển tài nguyên đó đến nơi cần thiết, hay chứng thực, uỷ quyền…
* Chức năng Transporting: tập trung vào việc cung cấp kết nối cho việc truyền thông tin của dịch vụ/ứng dụng IOT.

**Lớp thiết bị:**

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

* Thiết bị thông thường: Device này sẽ tương tác trực tiếp với network: Các thiết bị có khả năng thu thập và tải lên thông tin trực tiếp (nghĩa là không phải sử dụng gateway) và có thể trực tiếp nhận thông tin từ các network. Device này cũng có thể tương tác gián tiếp với network: Các thiết bị có thể thu thập và tải network gián tiếp thông qua khả năng gateway. Ngược lại, các thiết bị có thể gián tiếp nhận thông tin từ network. Trong thực tế, các Thiết bị thông thường bao gồm các cảm biến, các phần cứng điều khiển motor, đèn, …
* Thiết bị Gateway: Gateway là cổng liên lạc giữa device và network. Một Gateway hỗ trợ 2 chức năng sau:

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

Chức năng chuyển đổi giao thức: Chức năng này cần thiết trong hai tình huống là khi truyền thông ở lớp Device, nhiều device khác nhau sử dụng giao thức khác nhau, ví dụ, ZigBee với Bluetooth, và là khi truyền thông giữa các Device và Network, device dùng giao thức khác, network dùng giao thức khác, ví dụ, device dùng ZigBee còn tầng network thì lại dùng công nghệ 3G. Trong thực tế, Gateway có thể được build từ các board như Raspberry Pi hay Arduino, hoặc Gateway được sản xuất công nghiệp bởi các tập đoàn lớn như Intel hay Texas Instrument.

# 

# **CHƯƠNG 3 GIAO THỨC MQTT**

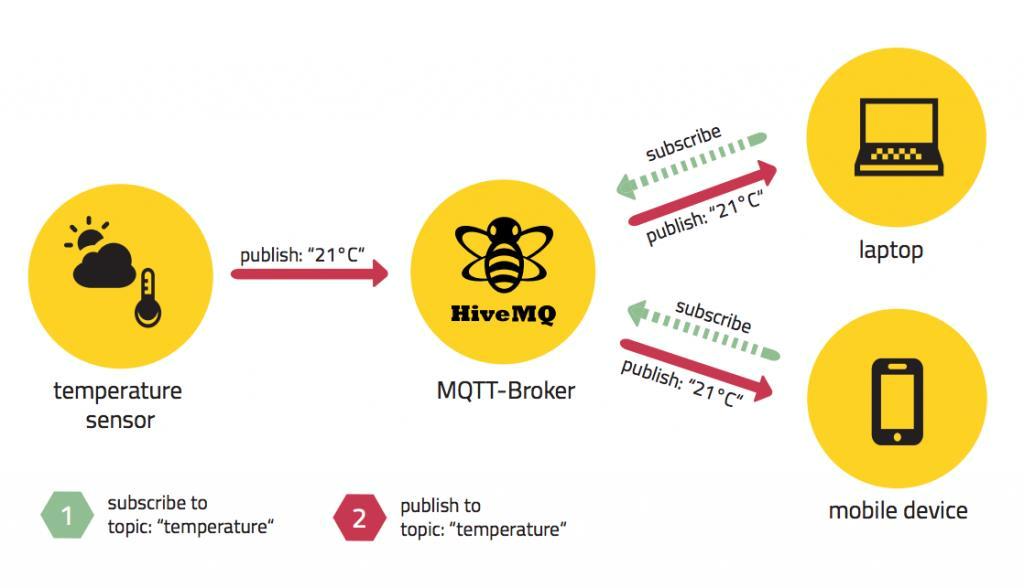
## **3.1 Giới thiệu giao thức MQTT**

MQTT (Message Queue Telemetry Transport) là một giao thức gửi dạng publish/subscribe sử dụng cho các thiết bị internet of thing với bang thông thấp và độ tin cậy cao và có khả năng sử dụng được trong mạng lưới không ổn định.

Bởi vì giao tiếp này sử dụng bang thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M. Điểm nổi bật của giao thức MQTT là rất dễ thực hiện trên các máy client, vì thế cho phép chúng ta thiết lập một client MQTT trên bất kì nền tảng nào rất nhanh chóng.

Mô hình MQTT gồm có các node trạm (gọi tắt là client) kết nối tới một MQTT server (gọi tắt là broker). Các client sẽ kết nối với broker để nhận và gửi dữ liệu. Một client có thể là một thiết bị bất kì như điện thoại, máy tính, các mạch điện tử nhúng…

## **3.2 Mô hình hoạt động của MQTT**



Hình 3.1 Mô hình Publish/Subscribe

Publish/Subscribe là một cách để thay thế cho mô hình client-server truyền thống, cách mà các client giao tiếp trực tiếp với server. Mỗi client khi kết nối đến broker sẽ đăng kí một vài kênh (topic), ví dụ: client1/chanel1, client1/chanel2.... Quá trình đăng kí này gọi là subscribe. Mỗi client sẽ nhận được dữ liệu khi bất kì client nào gửi dữ liệu tới kênh đó (hoặc cũng có thể là chính client đó). Quá trình gửi dữ liệu này gọi là publish

Publish/Subscribe tách riêng cho một client. Một client đang gửi dữ liệu (gọi là publisher) tới một hoặc nhiều client khác (gọi là subscriber) có thể không hoạt động.

Điều đó có nghĩa là sẽ không có subscriber, một publisher có thể gửi dữ liệu vào một kênh nào đó mà không có subscribe nào subscribe nó, có nghĩa là publisher và subscribe không biết đến sự tồn tại của nhau. Có một thành phần thứ ba có để đảm nhận nhiệm vụ truyền nhận dữ liệu đã đề cập ở trên, là broker, nơi mà lọc ra các dữ liệu và truyền chúng đi tới đúng subscriber.

**Các đặc điểm của publisher và subscriber:**

* Tách nhau về mặt không gian: các publisher và subscriber không cần thiết phải biết đến sự tồn tại của nhau (như địa chỉ IP và PORT).
* Tách nhau về mặt thời gian: các publisher và subscribe không cần thiết phải được chạy ở cùng một thời điểm, điều này có nghĩa là khi publisher đang kết nối đến server thì có thể subscriber không kết nối và ngược lại.
* Sự đồng bộ: các hoạt động ở cả publisher và subscriber không thể dừng trong quá trình publish và subscribe.

Mô hình publish/subscribe cung cấp khả năng mở rộng tuyệt vời hơn mô hình client-server truyền thống. Điều này do các hoạt động trên broker có thể được xử lý song song. Bộ nhớ dữ liệu đệm và bộ định tuyến thông minh có tính quyết định trong việc nâng cao khả năng mở rộng. Nhưng sẽ là một thách thức đối với quy mô lớn nơi có hang triệu kết nối từ client tới broker. Điều này có thể được giải quyết bằng cách sử dụng nhiều broker và phân phối dữ liệu trên nhiều broker đó để giảm bớt sức nặng.

Vậy làm thế nào để broker có thể gửi dữ liệu tới đúng client đã subscribe kênh đó?

**Bằng cách lọc dữ liệu tới theo những cách thức sau:**

* Lọc theo kênh: đây là cách lọc dựa vào kênh, cái được định sẵn trong dữ liệu gửi tới. Client đăng kí một kênh mà nó quan tâm với broker và từ kênh đó nó sẽ nhận toàn bộ dữ liệu trên kênh đó. Kênh là một chuỗi kí tự đơn giản, cho phép broker xử lý và lọc được. Lọc theo nội dung: broker lọc dữ liệu theo nội dung của dữ liệu đó. Những client đăng kí với broker một nội dung mà nó quan tâm. Nội dung của những dữ liệu đó sẽ được broker đọc trước khi sử dụng nhưng sẽ không bị thay đổi.
* Lọc theo loại tin nhắn: khi sử dụng ngôn ngữ hướng đối tượng, publisher phải publish dữ liệu đó với một loại xác định. Client sẽ nhận được toàn bộ các dữ liệu gửi tới và sẽ sử dụng loại của dữ liệu đó để xác định có phải dữ liệu mà nó quan tâm hay không. Trong trường hợp sử dụng lọc theo kênh thì cả publisher và subscriber đều phải biết chính xác kênh mà chúng đang thao tác. Và trong suốt chương trình chúng ta sẽ sử dụng cách này.

## **3.3 QoS (Quality of service)**

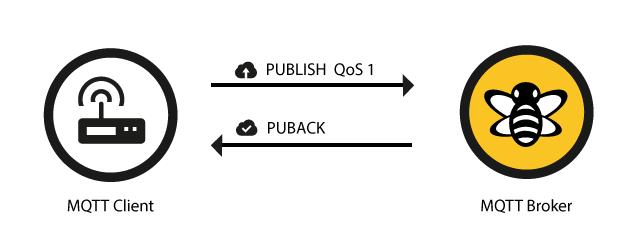
QoS là một thỏa thuận giữa publisher và subscriber cho việc truyền và nhận dữ liệu. Có 3 cấp độ là: Qos0, Qos1 và QoS2. Mức bảo mật tăng dần theo từng cấp độ. Khi nói về QoS luôn có hai thành phần khác nhau. Client publish với một QoS xác định và subscribe cũng với một QoS xác định. Và QoS cũng có thể hạ cấp nếu client publish và subscribe với hai mức QoS khác nhau, nhưng không có điều ngược lại (tức là một mức QoS thấp hơn không thể được nâng lên mức QoS cao hơn).

QoS là tính năng chính của giao thức MQTT. Nó giúp cho các giao tiếp dễ dàng hơn nhiều và đảm bảo việc cung cấp dữ liệu chính xác. Ngoài ra nó còn trao quyền cho một client để lựa chọn mức độ QoS phù hợp với mạng lưới truyền thông.

**Ba mức độ của QoS:**

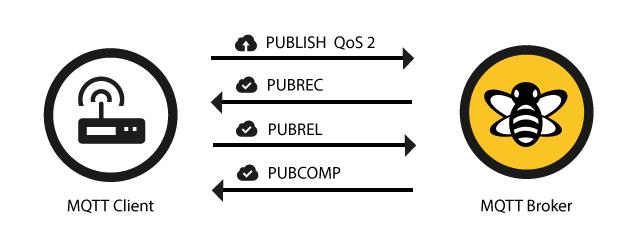
* QoS 0: đây là mức độ thấp nhất của QoS. Dữ liệu sẽ không được xác nhận bởi subscriber và không được lưu lại ở publisher. Nó còn được gọi là “fire and forget” tức là dữ liệu sẽ được gửi bởi publisher đúng một lần và được broker nhận đúng một lần. Nếu broker không nhận đươc vì lý do nào đó thì sẽ không thể nhận được dữ liệu đó vì nó không được lưu lại. Và publisher cũng không quan tâm dữ liệu mình gởi đi có được nhận hay không.

Hình 3.2 Publish với QoS 0

* QoS 1: Khi publisher gởi một gói dữ liệu với QoS 1. Nó đảm bảo rằng broker sẽ nhận được dữ liệu đó ít nhất một lần, và tất nhiên là có thể nhiều hơn một lần. Publisher sẽ lưu lại dữ liệu cho tới khi nào nhận được gói dữ liệu Puback từ broker. Sau đó việc publish sẽ hoàn thành khi publisher so sánh Package

Hình 3.3 Publish với QoS 1

Identifiers từ 2 gói dữ liệu publish và puback. Nếu Publisher không nhận được puback trong một khoảng thời gian thì nó sẽ gửi lại dữ liệu đó. Khi broker nhận được dữ liệu với QoS 1 thì gói dữ liệu đó sẽ được xử lý ngay lập tức, ví du như là gởi tới tất cả các subscriber và gởi gói dữ liệu puback tới publisher.

* QoS 2: là mưc độ cao nhất của QoS, nó đảm bảo broker chỉ nhận dữ liệu đúng một lần. Là mức QoS an toàn nhất nhưng tốc độ truyền dữ liệu cũng chậm nhất. Nó có 2 luồng dữ liệu và 2 gói dữ liệu trả về từ broker.

Hình 3.4 Publish với QoS 2

Nếu broker nhận được gói dữ liệu với QoS 2 thì nó sẽ xử lý và trả về gói dữ liệu Pubrec. Broker sẽ lưu lại tham chiếu của gói dữ liệu đó cho tới khi nó gửi đi gói pubcomp. Điều này đảm bảo broker chỉ nhận đúng gói dữ liệu đó 1 lần. Khi publisher nhận được gói pubrec, nó sẽ xóa dữ liệu gửi trước đó vì nó biết rằng broker đã nhận được và nó sẽ lưu gói pubrec và gửi lại broker gói pubrel. Khi broker nhận được gói pubrel, nó sẽ xóa toàn bộ thông tin về dữ liệu đó và trả lại gói pubcomp. Khi publisher nhận được gói pubcomp thì điều tương tự cũng xảy ra.

Khi các luồng trao đổi dữ liệu hoàn thành thì broker đã nhận được dữ liệu đúng một lần và publisher cũng biết điều đó

Package Identifiers là một chuỗi ký tự duy nhất giữ client và broker nhưng không phải duy nhất cho tất cả các client. Khi kết thúc một luồng gửi dữ liệu thì nó có thể sử dụng lại Package Identifiers của lần trước. Điều này cho phép giao thức không cần sử dụng quá nhiều Package ID.

**Và các QoS được sử dụng khi nào:**

* QoS 0 được sử dụng khi chúng ta có một đường truyền internet ổn định và chúng ta cũng không quan tâm là dữ liệu có bị mất trên đường truyền hay không.
* QoS 1 được sử dụng khi chúng ta cần chắc chắn rằng phải nhận được dữ liệu mà không cần biết là nhận bao nhiều lần, bởi vì mỗi dữ liệu được nhận đã được xử lý.
* QoS 2 được sử dụng khi chúng ta cần mức bảo mật cao nhất tuy nhiên tốc đọ truyền dữ liệu sẽ bị giảm đi nhiều. Vì vậy mức QoS 1 thường được sử dụng hơn.

## **3.4 Các gói dữ liệu cơ bản của MQTT**

Khi client và broker giao tiếp với nhau, chúng làm việc với các gói dữ liệu publish, subscribe và connection. Các gói dữ liệu này đã được định nghĩa trong giao thức, và client muốn giao tiếp được với broker, chúng phải gửi và nhận các gói dữ liệu giống như những gì giao thức đã định nghĩa. Các gói dữ liệu cơ bản như sau:

### ***3.4.1 Gói Connection***

Hình 3.5 Gói dữ liệu connect

Đây là gói dữ liệu mà client phải gửi tới broker để thiết lập giao tiếp. Nếu gói dữ liệu này không đúng hoặc nó mất quá lâu để gửi đi thì broker sẽ đóng kết nối, việc này đảm bảo broker không mất quá nhiều thời gian để xử lý cho một client. Gói dữ liệu này bao gồm:

ClientID (client Identifier): là một định danh cho mỗi client kết nối tới broker. Như nghĩa của từ Identifier, nó là duy nhất cho mỗi broker. Broker dùng nó để định danh client và lưu lại trạng thái hiện tại của client đó. Nếu chúng ta không muốn lưu lại trạng thái hiện tại của client, chúng ta có thể gửi một clientID rỗng.

Clean Session: là một cờ để chỉ định broker có lưu lại trạng thái của client hay không. Nếu clean session là “true” thì broker sẽ không lưu lại bất cứ thứ gì của client. Có nghĩa là nếu chúng ta gửi một clientID rỗng thì chúng ta cũng phải gửi một clean session rỗng.

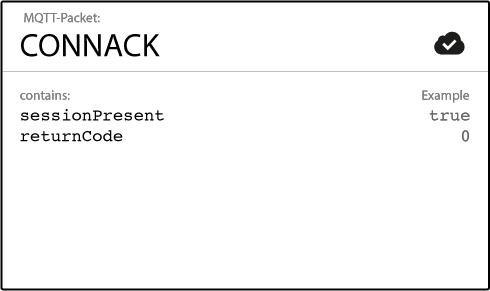
Username và password: dùng để bảo mật cho client, và điều này là không bắt buộc, có nghĩa là chúng ta có thể không gửi username và password.

Last Will: đây là một phần chức năng của giao thức MQTT. Cho phép các client khác nhận được thông báo khi client mất kết nối với broker. Bao gồm last will message, last will topic và last will QoS. Khi một client kết nối với broker mà có đăng kí Last Will thì khi client đó mất kết nối với broker, broker sẽ gửi last will message vào last will topic với QoS mà client đó đã đăng kí. Các client các chỉ cần subscribe topic đó thì sẽ nhận được thông báo. Và điều này cũng không bắt buộc, client có thể không đăng kí last will.

Keep Alive: đây là một khoảng thời gian mà client phải gửi một gói dữ liệu ping tới broker và broker cũng gửi trả lại client gói dữ liệu ping. Điều này cho phép broker biết được client có còn đang kết nối hay không.

### ***3.4.2 Gói Connack***

Đây là gói dữ liệu broker trả về khi nhận được gói connection của client. Xác nhận cho client rằng kết nối thành công hay thất bại.

Session Present: đây là tín hiệu xác nhận client có thông tin được lưu lại trên broker hay không. Nếu client kết nối với clean session là “true” thì Session Present sẽ là “false”. Nếu client kết nối với clean session là “false” thì sẽ có hai trường hợp. Nếu đây là lần đầu tiên client kết nối thì Session Present là “false”, còn nếu client đã có thông tin trên broker thì Session Present là “true”.

Hình 3.6 Gói dữ liệu connack trả về khi client connect

Return Code: là mã trả về trạng thái kết nối của client. Được cho trong bảng 3.1.

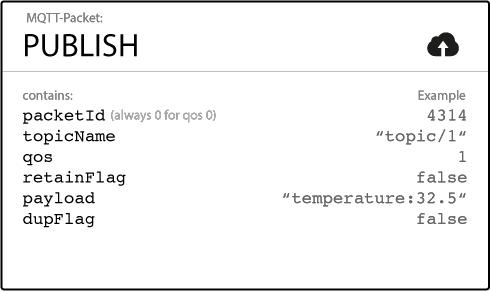
Bảng 3.1 Bảng kết quả trả về khi client kết nối tới broker

|  |  |
| --- | --- |
| Return code | Kết quả |
| 0 | Kết nối được chấp nhận |
| 1 | Kết nối thất bại, phiên bản giao thức lỗi |
| 2 | Kết nối thất bại, ClientID bị từ chối |
| 3 | Kết nối thất bại, server không tồn tại |
| 4 | Kết nối thất bại, username và password không chấp nhận |
| 5 | Kết nối thất bại, không được ủy quyền |

### ***3.4.3 Gói Publish***

Đây là gói dữ liệu client publish lên broker. Mỗi gói dữ liệu publish bắt buộc phải có kênh (topic) để broker gửi dữ liệu đó đến các client subscribe kênh đó và một chuỗi tin nhắn (payload). Gói dữ liệu publish cơ bản như sau:

Topic: là một chuỗi đơn giản, được phân cấp bằng dấu “/”. Ví dụ như: client1/chanel1, client1/chanel2… Broker dùng chuỗi này để gửi dữ liệu đến các client subscribe cùng chuỗi này.

QoS: là một thỏa thuận cho việc truyền nhận dữ liệu như đã đề cập ở trên, có ba cấp độ 0,1 và 2 đảm bảo việc truyền nhận dữ liệu.

Hình 3.7 Gói dữ liệu publish của client

Retain Flag: là một cờ cho biết dữ liệu này có được broker lưu lại hay không. Nếu retain flag là “false” thì dữ liệu sẽ không được lưu lại, nếu retain flag là “true” thì dữ liệu này sẽ được broker lưu lại. Và broker sẽ gửi gói dữ liệu này cho tất cả các client subscribe kênh đó sau này. Và mỗi kênh chỉ có một gói dữ liệu có retain flag là “true”, nếu ta publish một gói khác với retain flag là “true” thì broker sẽ xóa gói dữ liệu với retain flag trước đó.

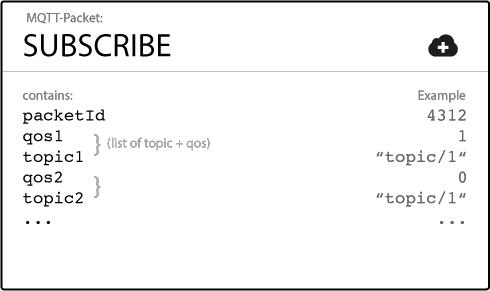
Payload: là chuỗi tin nhắn mà ta muốn gửi đi, broker có thể xử lý các ký tự trong chuỗi này với mọi định dạng.

Package ID: là một định danh cho gói dữ liệu trong luồng truyền dữ liệu giữa client và broker. Nó luôn có giá trị nếu QoS là 1 hoặc 2, và luôn bằng không nếu QoS là 0.

DUP flag (Duplicate flag): là một cờ cho biết gói dữ liệu này có được nhân đôi hay không. Với QoS là 1 hoặc 2, khi client gửi gói dữ liệu mà không nhận được gì thì nó sẽ gửi lại gói dữ liệu này và lúc đó DUP flag là “true”.

### ***3.4.4 Gói Subscribe***

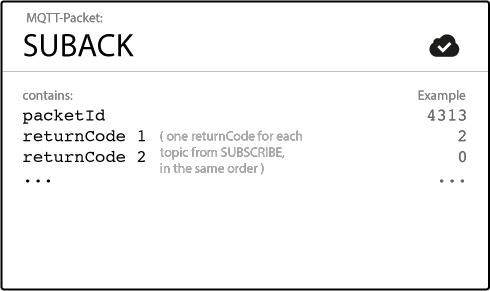
Đây là gói dữ liệu client dùng để subscribe một kênh trên broker. Cấu trúc gói tin đó như sau:



Hình 3.8 Gói dữ liệu subscribe của client

List of subscription: đây là một danh sách chứa các subscription của client. Mỗi subscription chứa QoS và kênh mà client muốn subscribe. Nếu có nhiều subscription cùng một kênh thì subscription có QoS cao nhất sẽ được sử dụng.

### ***3.4.5 Gói Suback***

Đây là gói dữ liệu mà broker trả về khi nhận được gói subscribe của client. Chứa thông tin về quá trình subscribe của client. Gói này chứa PackageID và một danh sách các return Code mô tả client có subscribe thành công hay không

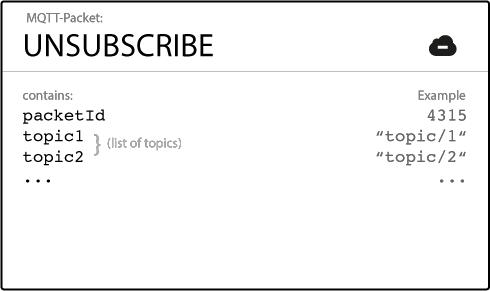
Hình 3.9 Gói dữ liệu suback trả về khi client subscribe

Return code được cho trong bảng 3.2.

Bảng 3.2: Bảng kết quả trả về khi client subscribe

|  |  |
| --- | --- |
| Return code | Kết quả |
| 0 | Thành công – QoS 0 |
| 1 | Thành công – QoS 1 |
| 2 | Thành công – QoS 2 |
| 128 | Thất bại |

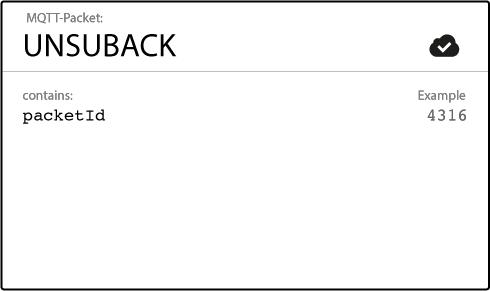
### ***3.4.6 Gói Unsubscribe***

Đây là gói dữ liệu client dùng để hủy subscribe các topic mà client đã subscribe. Cũng giống như gói subscribe, unsubscribe cũng có PackageID và một danh sách các kênh.

Hình 3.10 Gói dữ liệu unsubscribe của client

Topic: chứa các kênh mà client muốn unsubscribe. Gói này sẽ không chứa các subscription, bởi vì các subscription đã được lưu trên broker. Nên chúng ta chỉ cần chỉ định các kênh là đủ.

### ***3.4.7 Gói Unsuback***

Cũng giống như subscribe, khi broker nhận được gói unsubscribe, nó cũng sẽ trả về gói unsuback để xác nhận thông tin unsubscribe của client. Gói này chỉ gồm một PackageID, bởi vì client chỉ cần xác nhận được là các kênh mà nó đã subscribe đã được hủy.

Hình 3.11 Gói dữ liệu unsuback trả về khi client unsubscribe

Tất cả trên đây là các gói dữ liệu cơ bản của giao thức MQTT. Mọi client hay broker đều phải sử dụng nó để giao tiếp. Một điểm lợi của các gói dữ liệu này là chúng ta có thể dễ dàng dùng các ngôn ngữ lập trình để mô tả chúng.

# 

# **Chương 4 CÁC MODULE TRONG HỆ THỐNG**

## **4.1 Giới thiệu module ESP8266**

Mỗi một module trong hệ thống đều được điều khiển bởi một module ESP8266, đây là một loại chip SOC (System on Chip). Chip SOC là một chip tích hợp nhiều chức năng trên một con chip.

ESP8266 là module wifi giá rẻ và được đánh giá rất cao cho các ứng dụng liên quan đến Internet và Wifi cũng như các ứng dụng truyền nhận sử dụng thay thế cho các module RF khác.

ESP8266 là một chip tích hợp cao, được thiết kế cho nhu cầu của một thế giới kết nối mới, thế giới Internet of thing. Nó cung cấp một giải pháp kết nối mạng WiFi đầy đủ và khép kín, cho phép nó có thể lưu trữ các ứng dụng hoặc để giảm tải tất cả các chức năng kết nối mạng WiFi từ một bộ xử lý ứng dụng.

ESP8266 có xử lý và khả năng lưu trữ mạnh mẽ cho phép nó được tích hợp với các bộ cảm biến, vi điều khiển và các thiết bị ứng dụng cụ thể khác thông qua GPIO với một chi phí tối thiểu và một PCB (Printed Circurt Board) tối thiểu.

Hình 4.1 Chip ESP8266

ESP8266 có 16 chân GPIO phục vụ mục đích điều khiển và kết nối với các thiết bị ngoại vi, hỗ trợ đầy đủ các giao thức I2C, SPI, One Wire phục vụ mục đích giao tiếp. Có thể nói với giá thành của ESP8266 và các chức năng mà nó hỗ trợ, đây được đánh giá là một module rất mạnh trong các ứng dụng có liên quan đến IOT.

**Các thông số kĩ thuật:**

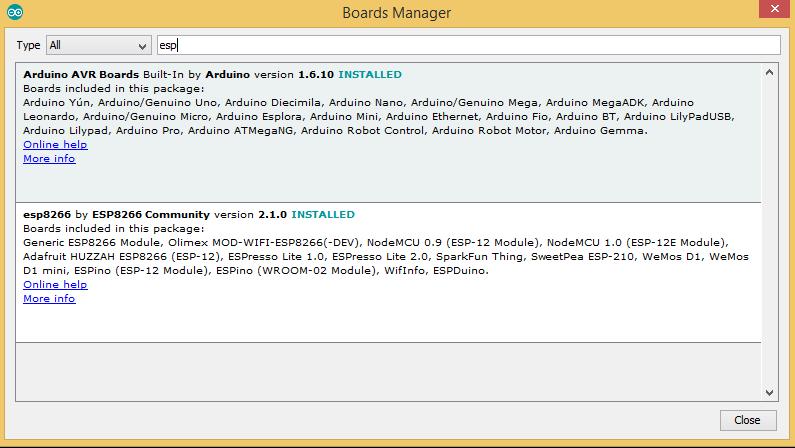
* Hỗ trợ chuẩn wifi 802.11 b/g/n
* Wifi 2.4 GHz, hỗ trợ WPA/WPA2
* Chuẩn điện áp hoạt động 3.3V
* Giao tiếp UART với tốc độ baund lên đến 115200
* Có 3 chế độ hoạt động: client, access point và cả client và access point
* Hỗ trợ các chuẩn bảo mật như: OPEN, WEP, WPA\_PSK, WPA2\_PSK, WPA\_WPA2\_PSK
* Hỗ trợ cả 2 giao tiếp TCP và UDP
* Làm việc như các máy chủ có thể kết nối tới 5 máy con

### ***4.1.1 Phần mềm lập trình cho ESP8266***

Hình 4.2 Giao diện phần mềm Arduino

Có rất nhiều phần mềm để lập trình và nạp firmware cho ESP8266, nhưng tôi chọn phần mềm Arduino. Một phần mềm mã nguồn mở được sử dụng rất rộng rãi trên thế giới bởi vì tính gọn nhẹ và dễ sử dụng của nó, đắc biệt nó hỗ trợ rất tốt cho ESP8266

### ***4.1.2 Cấu hình phần mềm để lập trình cho ESP8266***

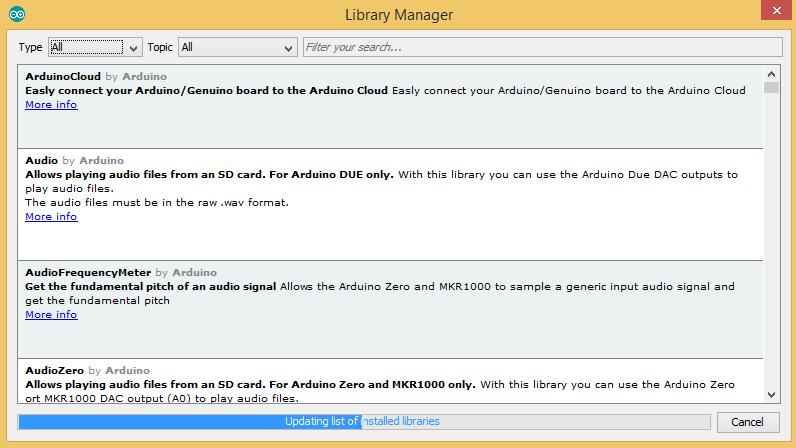
Trước khi bắt đầu lâp trình, chúng ta phải cấu hình cho Arduino IDE (Integrated Development Enviroment) lập trình được ESP8266, vì IDE này dùng để lập trình cho dòng board Arduino nên mặc định nó sẽ không có sẵn module khác. Nên chúng ta sẽ thêm module ESP8266 vào phần mềm Arduino bằng cách sử dụng chức năng board manager.

Hình 4.3 Giao diện board manager của arduino

### ***4.1.3 Các thư viện cần thiết***

Arduino IDE là một phần mềm được rất nhiều người sử dụng bởi vì kho thư viện phong phú của nó. Và cách để tìm một thư viện cũng rất đơn giản. Sử dụng chức năng library manager chúng ta có thể dễ dàng tìm và cài đặt một thư viện.

**Các thư viện cần thiết trong chương trình là:**

* Bộ SDK của ESP8266 bao gồm các thư viện liên quan về wifi
* Thư viên wifi manager
* Thư viện MQTT client
* Thư viện IR remote
* Thư viện DTH

Hình 4.4 Giao diện cài đặt thư viện arduino

### ***4.1.4 Nạp firmware cho module ESP8266***

Việc nạp firmware cho module ESP8266 cũng rất đơn giản. Chúng ta cần có một module nữa để chuyển giao tiếp USB sang UART vì phần mềm Arduino nạp firmware thông qua giao tiếp UART.

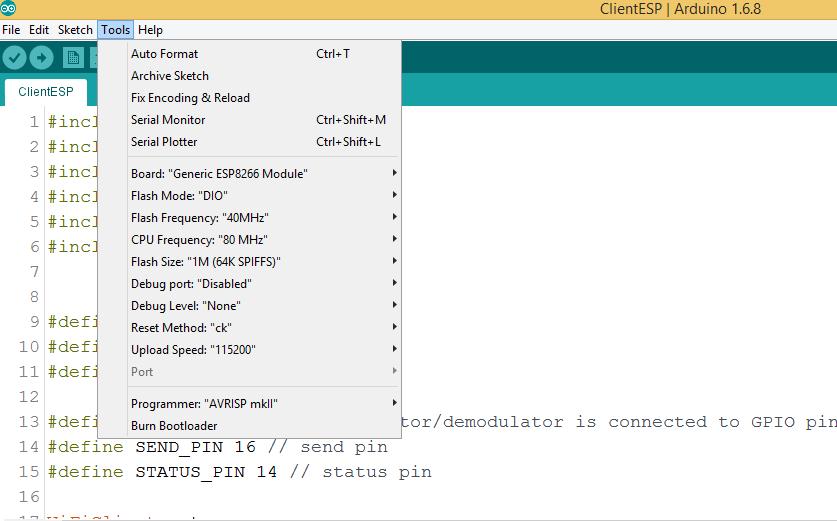


Hình 4.5 Module USB - UART

|  |  |  |
| --- | --- | --- |
| ESP8266 | Register | USB - UART |
| Vcc |  | 3.3V |
| GND |  | GND |
| TX or GPIO2 |  | RX |
| RX |  | TX |
| GPIO0 |  | GND |
| Reset |  | RST |
| GPIO15 | Kéo lên |  |
| CH\_PD | Kéo xuống |  |

Cách nối dây được cho trong bảng 4.1

Bảng 4.1 Cách nối dây giữa module UART và ESP8266

Chúng ta cài đặt board để nạp như sau:

Hình 4.6 Cách thiết lập cài đặt board

* Board: Generic ESP8266 Module.
* Flash Mode: DIO.
* Flash Frequency: 40MHz.
* Flash Size: 1M (64K SPIFFS).
* Debug port: Disabled.
* Debug level: None.
* Reset Method: ck.
* Upload Speed: 115200.
* Port: chọn cổng com kết nối với USB-UART (tùy từng máy).
* Sau khi cài đặt xong ta có thể nạp firmware cho ESP8266 bằng phần mềm arduino. Khi nạp xong ta kéo chân GPIO0 lên cao để ESP8266 bắt đầu chạy.

## **4.2 Module điều khiển thông qua hồng ngoại**

Hình 4.7 Module điều khiển hồng ngoại

### ***4.2.1 Sơ đồ nguyên lý***

Hình 4.8 Sơ đồ nguyên lý module hồng ngoại

Chip ESP8266 cần nguồn điện 3.3V nên ta cần một bộ chuyển đổi AC/DC từ 220V AC về 5V DC và chip SC189ZSKTRT để chuyển nguồn điện từ 5V về 3.3V. Mạch gồm có 8 led hồng ngoại được điều khiển bởi chân GPIO16 của chip ESP8266. Mỗi led phát được nối qua một transistor nhằm tăng dòng qua led vì led hồng ngoại sử dụng dòng điện lớn. Một led thu hồng ngoại được nối vào chân GPIO4.

### ***4.2.2 Hoạt động***

Module có chip điều khiển chính là ESP8266, kết nối với internet và giao tiếp với điện thoại thông qua giao thức MQTT.

Module này có chức năng điều khiển các thiết bị có hỗ trợ hồng ngoại bằng cách học các lệnh hồng ngoại từ remote điều khiển của thiết bị. Sau khi học, module sẽ gửi mã hồng ngoại này về điện thoại thông qua giao tiếp MQTT. Điện thoại sẽ lưu lại và khi cần điều khiển ta chỉ cần gữi mã đó xuống module để điều khiển trực tiếp.

Module còn có chức năng hẹn giờ phát lệnh hồng ngoại, chức năng này sẽ được cài đặt trên điện thoại. Và có thể phát nhiều mã lệnh liên tiếp để thực hiện một chức năng.

Ví dụ: Ta muốn 5h TiVi sẽ bật lên và chuyển đến kênh 12. Ta cài đặt trên điện thoại module sẽ phát đi mã hồng ngoại của phím ON, phím 1, và phím 2 liên tiếp.

Module không có mạch thời gian thực nên thời gian sẽ được lấy trực tiếp từ internet. Thời gian của module đồng bộ với internet nên rất chính xác, sẽ không bị trễ.

Module có 8 led phát hồng ngoại và 1 led thu hồng ngoại. Led thu dùng để nhận tín hiệu hồng ngoại từ bên ngoài, xử lý và gửi dữ liệu về điện thoại. 8 led phát được xếp thành vòng tròn 6 led và 2 led hướng lên đảm bảo tín hiệu phát ra đi tới được thiết bị cho dù có đặt module ở vị trí nào.

### ***4.2.3 Thư viện dùng cho module***

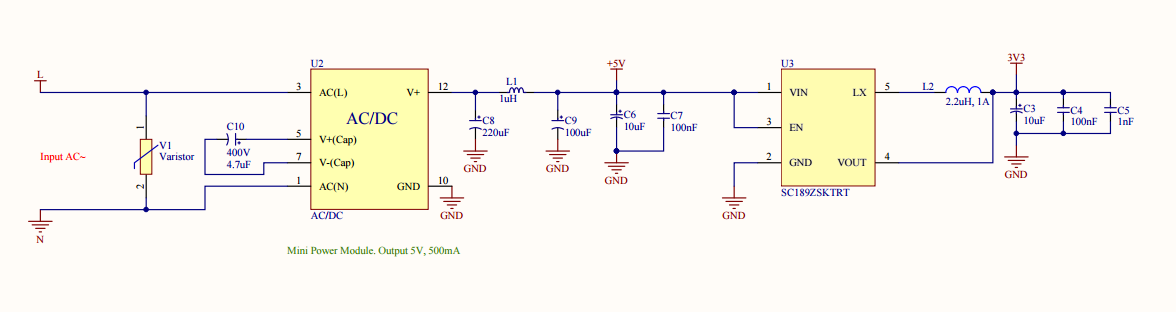
* Bộ SDK của ESP8266: chứa các hàm liên quan đến ESP8266 cũng như việc thu phát wifi của thiết bị.
* Thư viện WiFi Manager: là một thư viện quản lý về wifi, có chức năng cho phép người dùng thiết lập wifi cho ESP8266 bằng giao diện đồ họa mà việc đó trước đây phải sửa code.
* Thư viện IRremote: là thư viện thu phát hồng ngoại dùng để điều khiển

Bộ SDK sẽ được tự động cài đặt khi ta thêm chip ESP8266 vào phần mềm Arduino, các thư viện còn lại có thể dễ dàng cài đặt qua Library Manager như đã đề cập ở trên.

## **4.3 Module cảm biến độ ẩm nhiệt độ**

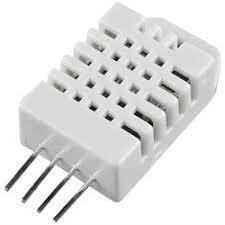
Hình 4.9 Module cảm biến nhiệt độ độ ẩm

### ***4.3.1 Sơ đồ nguyên lí***



Hình 4.10 Sơ đồ nguyên lý module cảm biến

### ***4.3.2 Hoạt động***

Module này sử dụng cảm biến nhiệt độ độ ẩm DHT22 là một cảm biến giá rẻ, dễ sử dụng. Sử dụng giao tiếp One Wire cho tốc độ giao tiếp cao.

Hình 4.11 Cảm biến nhiệt độ, độ ẩm DHT22

**Một vài thông số kĩ thuật của cảm biến:**

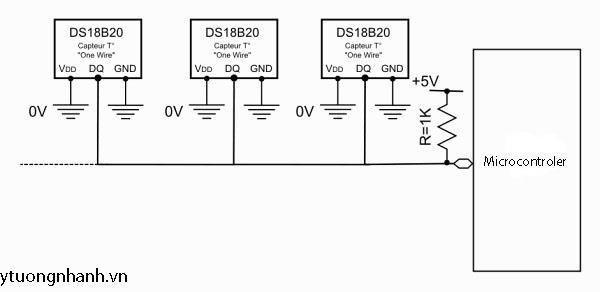
* Điện áp: 3V-5V
* Dòng điện: 2.5mA khi truyền dữ liệu
* Đo tốt ở độ ẩm 0100%RH với sai số 2-5%
* Đo tốt ở nhiệt độ -40 tới 80 \*C với sai số 0.5 \*C
* Tần số lấy mẫu tối đa là 0.5 Hz (2 giây lấy mẫu 1 lần)

Module này có chức năng cảm biến nhiệt độ, độ ẩm và gửi dữ liệu về điện thoại. Giúp chúng ta theo dõi được nhiệt độ và độ ẩm trong nhà mọi lúc mọi nơi.

**Chuẩn giao tiếp One Wire:**

Chuẩn giao tiếp 1 dây (1 wire) do hãng Dallas giới thiệu. Trong chuẩn giao tiếp này chỉ cần 1 dây để truyền tín hiệu và làm nguồn nuôi (Nếu không tín dây mass). Là chuẩn giao tiếp không đồng bộ và bán song công (half-duplex). Trong giao tiếp này tuân theo mối liên hệ chủ tớ một cách chặt chẽ. Trên một bus có thể gắn 1 hoặc nhiều thiết bị slave. Nhưng chỉ có một master có thể kết nối đến bus này.

Hình 4.12 Sơ đồ chuẩn giao tiếp One Wire

Bus dữ liệu khi ở trạng thái rãnh (khi không có dữ liệu trên đường truyền) phải ở mức cao do vậy bus dữ liệu phải được kéo lên nguồn thông qua một điện trở. Giá trị điện trở này có thể tham khảo trong datasheet của thiết bị.

Các thiết bị tớ (slave) kết nối với cùng một bus được phân biệt với nhau nhờ 64 bit địa chỉ duy nhất (64-bit serial number). 8 byte (64 bit) này và được chia làm ba phần chính:

Bắt đầu với LSB, là byte đầu tiên là mã họ thiết bị có độ lớn 8 bit (8-bit family codes) xác định kiểu thiết bị. 6 byte tiếp theo lưu trữ địa chỉ riêng của thiết bị. Byte cuối cùng (MSB) là byte kiểm tra tính toàn vẹn dữ liệu cyclic redundancy check có giá trị tương ứng với giá trị của 7 byte đầu tiên. Nhờ byte CRC giúp cho master xác định có địa chỉ được đọc có bị lỗi hay không. Với 224 địa chỉ khác nhau tạo ra một số lượng rất lớn các địa chỉ.Do vậy vấn đề về địa chỉ không phải là vấn đề chính trong chuẩn giao tiếp này.

### ***4.3.3 Các thư viện cần thiết***

* Bộ SDK của ESP8266: chứa các hàm liên quan đến ESP8266 cũng như việc thu phát wifi của thiết bị.
* Thư viện WiFi Manager: là một thư viện quản lý về wifi, có chức năng cho phép người dùng thiết lập wifi cho ESP8266 bằng giao diện đồ họa mà việc đó trước đây phải sửa code.
* Thư viện DHT: là thư viện dùng để giao tiếp với cảm biến DHT22 bằng giao thức One Wire.

Bộ SDK sẽ được tự động cài đặt khi ta thêm chip ESP8266 vào phần mềm Arduino, các thư viện còn lại có thể dễ dàng cài đặt qua Library Manager như đã đề cập ở trên.

# 

# **Chương 5 PHẦN MỀM TRÊN DI ĐỘNG**

## **5.1 Giới thiệu hệ điều hành di động IOS**

IOS là  [hệ điều hành trên các thiết bị di động](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh_tr%C3%AAn_c%C3%A1c_thi%E1%BA%BFt_b%E1%BB%8B_di_%C4%91%E1%BB%99ng&action=edit&redlink=1) của [Apple.](https://vi.wikipedia.org/wiki/Apple_Inc.) Ban đầu hệ điều hành này chỉ được phát triển để chạy trên [iPhone](https://vi.wikipedia.org/wiki/IPhone) (gọi là iPhone OS), nhưng sau đó nó đã được mở rộng để chạy trên các thiết bị của Apple như  [iPod touch,](https://vi.wikipedia.org/w/index.php?title=IPod_touch&action=edit&redlink=1)  [iPad](https://vi.wikipedia.org/wiki/IPad) và  [Apple TV.](https://vi.wikipedia.org/w/index.php?title=Apple_TV&action=edit&redlink=1) Ngày 31 tháng 5, 2011,  [App Store](https://vi.wikipedia.org/w/index.php?title=App_Store&action=edit&redlink=1) của Apple chứa khoảng 500 000 ứng dụng iOS, và được tải về tổng cộng khoảng 15 tỷ lần. Thị phần của hệ điều hành IOS hiện nay là 15,1% trên tổng số smartphone toàn cầu, đứng sau số smartphone chạy android.

Giao diện người dùng của IOS dựa trên cơ sở thao tác bằng tay. Người dùng có thể tương tác với hệ điều hành này thông qua rất nhiều động tác bằng tay trên màn hình cảm ứng của các thiết bị của  [Apple](https://vi.wikipedia.org/wiki/Apple_Inc.).

IOS đã trải qua 9 phiên bản từ IOS 1 ra mắt năm 2007 tới nay là IOS 9.3 ra mắt vào tháng 3/2016 với nhiều tính năng cải tiến vượt bậc các phiên bản trước. Hệ điều hành IOS chỉ chạy được trên những thiết bị di động của apple như iphone, ipad và ipod.

Hình 5.1 Các phiên bản hệ điều hành IOS

**Ưu điểm:**

* Nền tảng ổn định, ứng dụng có khả năng tương thích cao. Ứng dụng trên IOS có thể hoạt động mượt mà vì chỉ được tối ưu hóa cho các thiết bị của Apple.
* Độ tin cậy và bảo mật cao.
* Chu kỳ ra mắt là một năm vì vậy bạn sẽ yên tâm không phải chạy đua quá nhiều.
* Ứng dụng phong phú, chất lượng và cập nhật nhanh hơn khi có phiên bản mới.

**Nhược điểm:**

* Không thể chạy nhiều ứng dụng cùng lúc trên màn hình.
* Trao đổi dữ liệu đều phải qua iTunes gây bất tiện và tốn thời gian vì đây là con đường duy nhất.
* Khả năng tuỳ chỉnh hạn chế.

Với tất cả những ưu điểm đó nên tôi chọn hệ điều hành này để thiết kế app trên đó, phiên bản trên android sẽ được làm sau khi phiên bản trên IOS ra mắt.

## **5.2 Công cụ lập trình và debug**

### ***5.2.1 Công cụ lập trình Xcode***

Do tính đặc thù của các thiết bị của apple nên muốn lập trình được cho IOS cần phải có một máy tính Mac của apple. Apple chỉ cung cấp một công cụ duy nhất dùng để lập trình cho IOS là Xcode và phần mềm này chỉ chạy được trên các máy Mac.

Hình 5.2 Giao diện XCode

### ***5.2.2 Công cụ debug Instrument***

Hình 5.3 Giao diện Instrument

Apple cung cấp một công cụ debug dành cho các phần mềm chạy trên IOS và OS X là Instrument. Đây là một công cụ cực kì mạnh mẽ dùng để thu thập số liệu của ứng dụng như thời gian khởi động, sử dụng CPU, GPU, bộ nhớ RAM, quản lý tiến trình, core animation…

Dựa vào các số liệu đó chúng ta có thể cải thiện lại bộ nhớ, performance cho ứng dụng. Làm cho hiệu năng của ứng dụng tốt hơn, sử sụng năng lượng hợp lý hơn, quản lý tài nguyên tốt hơn.

## **5.3 Ngôn ngữ lập trình Swift**

Apple cung cấp hai ngôn ngữ dùng để lập trình cho IOS là objective C và Swift. Objective C là ngôn ngữ đã ra đời từ khá lâu, và được apple sử dụng để lập trình các ứng dụng IOS của mình. Objective C là ngôn ngữ kế thừa từ C và là ngôn ngữ hướng đối tượng, cấu trúc ngôn ngữ này khá phức tạp cho những người mới học nếu chưa được học ngôn ngữ C.

Swift là một ngôn ngữ lập trình hướng đối tượng của Apple dành cho việc phát triển iOS và OS X, được giới thiệu bởi Apple tại hội nghị WWDC 2014. Swift được mong đợi sẽ tồn tại song song cùng Objective-C, ngôn ngữ lập trình hiện tại dành cho các hệ điều hành của Apple. Swift được thiết kế là một ngôn ngữ có khả năng phòng chống lỗi cao. Nó được biên dịch với trình biên dịch LLVM. Swift cũng là một ngôn ngữ cho giao diện lập trình Cocoa và Cocoa Touch cho hệ điều hành OS X cho máy tính và IOS cho di động.

Swift là một ngôn ngữ lập trình hoàn toàn mới có thể giúp quá trình phát triển ứng dụng trở nên dễ dàng hơn, nhanh hơn và ổn định hơn, tất cả nhằm tạo ra những sản phẩm ứng dụng cho người dùng cuối tốt hết mức có thể.

**Ưu điểm:**

* Các ngôn ngữ dạng script như Python rất dễ viết và kiểm tra, song chúng không mạnh mẽ và cũng không đủ nhanh để thực hiện các tác vụ phức tạp của ứng dụng. Nói cách khác, Python không thực sự phù hợp để viết các game nặng, vốn có yêu cầu phải tận dụng tối đa sức mạnh của thiết bị. Các ngôn ngữ lập trình truyền thống (ví dụ như Objective-C) mang tới hiệu năng khi chạy trên iPhone/iPad, cho phép tạo ra các ứng dụng phức tạp như iMovie hoặc Call of Duty. Song, việc sử dụng Objective-C thường đi kèm với một lượng lớn thời gian compile và test ứng dụng. Việc học cách viết Objective-C cũng là một quá trình dài, có thể gây khó khăn với các lập trình viên ít kinh nghiệm.
* Nếu thực hiện được tất cả các lời hứa của Apple, Swift sẽ mang tất cả các điểm mạnh của cả 2 loại ngôn ngữ script và compile truyền thống. Trong một số thử nghiệm benchmark, Swift cho hiệu năng nhanh hơn Python và thậm chí là nhanh hơn cả Objective-C. Trên bộ IDE (phần mềm lập trình) Xcode của Apple, mã nguồn Swift sẽ được đồ thị hóa dựa trên tính năng playground theo thời gian thực. Điều này có nghĩa rằng lập trình viên có thể chạy và kiểm tra mã nguồn Swift một cách dễ dàng không kém gì Python cả.
* Khả năng phát triển ứng dụng di động và kiểm tra thành quả của mình theo thời gian thực sẽ giúp cho quá trình code của các lập trình viên trở nên nhanh và dễ dàng hơn rất nhiều. Ngoài ra, việc không phải chạy chương trình compiler và các chương trình test quá thường xuyên cũng sẽ tiết kiệm được sức mạnh phần cứng cho nhà phát triển. Thay vì tập trung kiểm tra, thử nghiệm các tính năng nhỏ lẻ, lập trình viên Swift có thể dành nhiều thời gian để thực hiện các bài kiểm tra ở mức độ tích hợp cao hơn một cách kỹ càng hơn.
* Lợi ích thứ 2 của Swift là các dòng code rất ngắn và dễ đọc. Trong bài trình bày của Apple, 3 dòng code Objective C có thể gói gọn vào 1 dòng code Swift. Điều này sẽ đẩy nhanh quá trình phát triển ứng dụng và tăng hiệu quả cho quá trình bảo trì, vá lỗi trong tương lai.
* Việc lập trình với Swift cũng trực quan hơn nhờ phương thức sắp xếp hợp lý ở giao diện sử dụng và kiểu hiển thị kết quả theo thời gian thực trong X Code Runtime. Tức trong lúc viết mã, kết quả sẽ ngay lập tức hiện ra để tiện việc chỉnh sửa và hình ảnh cũng có thể xem ngay từ bên trong IDE. Ngoài ra, XCode cũng sẽ dựng các kịch bản thực hiện trong các ứng dụng, tính năng này giúp lập trình viên có thể quản lý ứng dụng khỏi bị sự cố tràn bộ nhớ.
* Cuối cùng và có lẽ là quan trọng nhất, nếu Swift tạo ra các ứng dụng có sức ép phần cứng thấp hơn Objective-C, chúng ta sẽ được tận hưởng trải nghiệm game ấn tượng hơn trên các thiết bị IOS. Khi kết hợp Swift cùng nền tảng Metal mới của Apple, ngay cả vi xử lý A7 cũ kỹ cũng có thể mang tới những trải nghiệm đồ họa ấn tượng nhất.

**Nhược điểm:**

* Điểm yếu lớn nhất của Swift sẽ là tuổi đời quá trẻ của ngôn ngữ lập trình này.
* Các nhà phát triển sẽ phải học lại một ngôn ngữ lập trình hoàn toàn mới.
* Các nhà phát triển ứng dụng IOS sẽ mất một khoảng thời gian đáng kể để làm chủ ngôn ngữ mới của Apple.
* Giảm tốc độ phát triển của hệ sinh thái ứng dụng Apple.

## **5.4 Hoạt động của ứng dụng**

Quản lý thiết bị

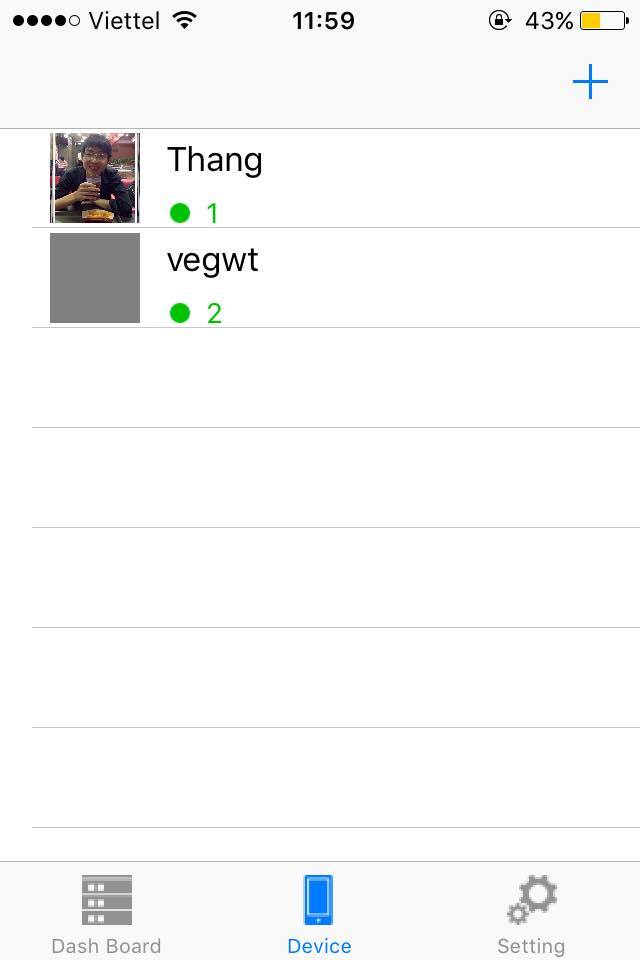
Tạo kịch bản để điều khiển

Điều khiển, nhận thông tin

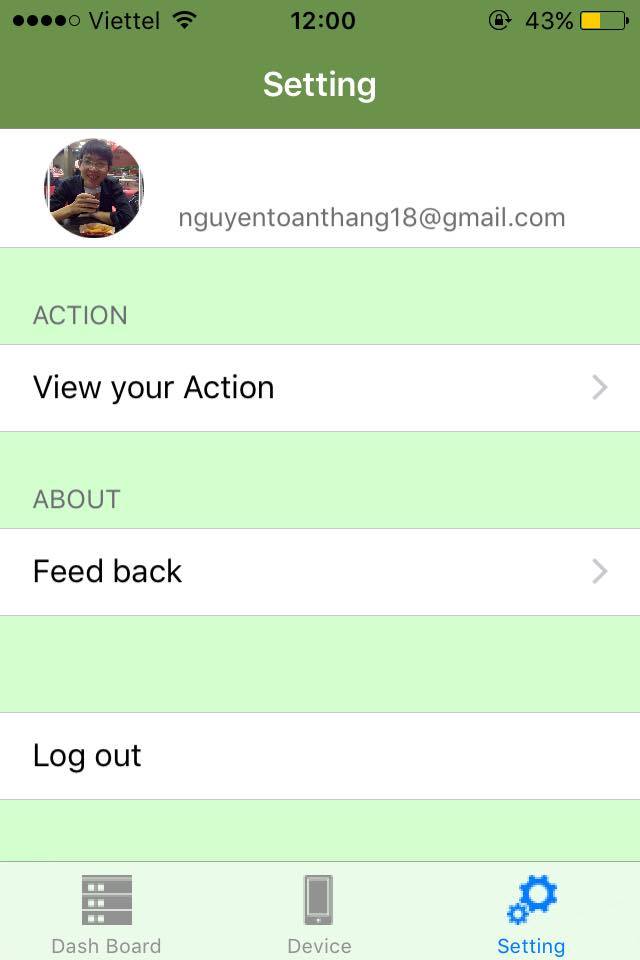
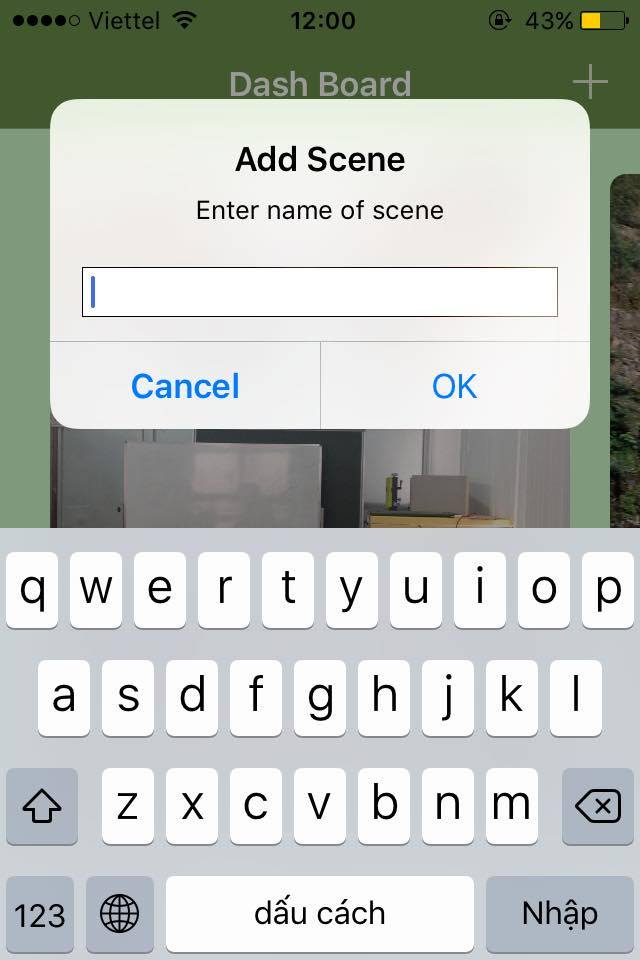
Hình 5.4 Sơ đồ use case của ứng dụng

* Quản lý thiết bị: Người sử dụng có thể quản lý được thiết bị của mình như đổi tên, kiểm tra thiết bị có đang được kết nối mạng hay không, thêm và xóa thiết bị.
* Điều khiển, nhận thông tin: Người sử dụng có thể điều khiển các thiết bị đó và nhận thông tin về từ các thiết bị.
* Tạo kịch bản điều khiển: Người sử dụng có thể tạo trước một kịch bản để các thiết bị có thể tự động chạy theo.

## **5.5 Giao diện**

Hình 5.5 Giao diện dash board của ứng dụng

Hình 5.6 Giao diện quản lý thiết bị

Hình 5.7 Giao diện Setting

Hình 5.8 Giao diện chức năng Add Scene

# 

# **KẾT LUẬN**

**Kết quả đạt được:**

* Thiết kế được 2 module dùng để điều khiển và cảm biến.
* Xây dựng app trên IOS.
* Điều khiển thành công và nhận dữ liệu thành công.
* Xây dựng được hệ thống IOT đơn giản.

**Kiến thức, kinh nghiệm đạt được:**

* Có thêm kiến thức về vi điều khiển ESP8266.
* Học được ngôn ngữ lập trình Swift.
* Có kiến thức lập trình di động IOS.

**Hạn chế:**

* Hệ thống còn đơn giản, chưa thiết kế được nhiều module khác.
* Chưa điều khiển được máy lạnh.
* Giao diện app còn đơn giản và hệ thống chưa tối ưu.

**Hướng phát triển:**

* Xây dựng nhiều module hơn nữa.
* Thiết kế lại giao diện app, quản lý mạnh mẽ hơn, tối ưu hệ thống hơn.
* Thêm nhiều chức năng cho hệ thống như có thể nạp firmware qua internet.
* Xậy dựng app trên hệ điều hành android.

# **TÀI LIỆU THAM KHẢO**

**Tiếng Anh**

[1] Apple Inc (2014), “The Swift Programming Language (Swift 2.2)”, Apple Inc.

[2] Adam Freeman (2015), “Pro Design Patterns in Swift”, Apress.

[3] Magesh Jayakumar (2015), “Nodemcu dev kit using Arduino IDE: Get started with ESP8266”, Magesh Jayakumar.

[4] Mohsen Hallaj Asghar (2016), “Internet of Things Architecture and Research App with MQTT Protocol”, LAP LAMBERT Academic Publishing.

[5] Peter Waher (2015), “Learning Internet of Things”, Packt Publishing.

[6] Matt Neuburg (2015), “iOS 9 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics”, O'Reilly Media.

**Các website**

[7] <https://www.raywenderlich.com/category/swift>.

[8] <https://www.raywenderlich.com/category/ios>.

[9] Swift.org.

[10] https://developer.apple.com/develop/.