**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**------------------------------**



**BÁO CÁO**

**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài:***

**“Xây dựng website tìm phòng trọ cho sinh viên”**

**Người hướng dẫn : THẠC SĨ HUỲNH TRỌNG THƯA**

**Sinh viên thực hiện: ĐẶNG QUANG NHẬT**

**Mã số sinh viên : N12DCCN028**

**Lớp : D12CQPM01**

**Khóa : 2012 – 2017**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 11 / 2016**

**LỜI CẢM ƠN**

Để hoàn thành chuyên đề báo cáo tốt nghiệp này trước hết em xin gửi đến quý thầy cô giáo trong khoa Công nghệ thông tin 2 lời cảm ơn chân thành.

Đặc biệt em xin gửi đến giảng viên hướng dẫn, Ths.Huỳnh Trọng Thưa, người đã tận tình hướng dẫn, giúp đỡ em hoàn thành chuyên đề báo cáo tốt nghiệp này lời cảm ơn sâu sắc nhất.

Vì kiến thức bản thân còn hạn chế, việc hoàn thiện đề tài tốt nghiệp này em không tránh khỏi những sai sót, hạn chế, kính mong nhận được những ý kiến đóng góp từ các thầy cô giáo.

Em xin chân thành cám ơn.

Sinh viên thực hiện

**Đặng Quang Nhật**

**MỤc lỤc**

[LỜI MỞ ĐẦU 1](#_Toc459926084)

[CHƯƠNG 1: ĐẶT VẤN ĐỀ 2](#_Toc459926085)

[1.1. Môi trường sống của cây trồng: 2](#_Toc459926086)

[1.2. Trồng cây trong nhà kinh: 3](#_Toc459926087)

[CHƯƠNG 2: MÔ HÌNH HỆ THỐNG 4](#_Toc459926088)

[2.1. Tổng quan: 4](#_Toc459926089)

[2.2. IBM Bluemix 5](#_Toc459926090)

[2.3. Giao thức truyền tin MQTT: 6](#_Toc459926091)

[2.3.1. Dạng truyền tin kiểu publish / subscribe: 6](#_Toc459926092)

[2.3.2. Khái quát về MQTT: 7](#_Toc459926093)

[2.4. Bộ cảm biến thu thập dữ liệu: 9](#_Toc459926094)

[2.5. Dịch vụ web xử lý dữ liệu: 11](#_Toc459926095)

[2.5.1. Cấu trúc REST: 11](#_Toc459926096)

[2.5.2. Dịch vụ web RESTful: 12](#_Toc459926097)

[2.5.3. Cách thức hoạt động của dịch vụ web: 12](#_Toc459926098)

[2.6. Cơ sở dữ liệu: 13](#_Toc459926099)

[CHƯƠNG 3: TRIỂN KHAI VÀ LẮP ĐẶT HỆ THỐNG 15](#_Toc459926100)

[3.1. Triển khai và lắp đặt bộ cảm biến: 15](#_Toc459926101)

[3.1.1. Lập trình module wifi esp8266: 16](#_Toc459926102)

[3.1.2. Lập trình vi điều khiển TM4C123GH6PM: 18](#_Toc459926103)

[3.2. Triển khai dịch vụ web: 20](#_Toc459926104)

[3.2.1. Dịch vụ web với cơ sở dữ liệu: 20](#_Toc459926105)

[3.2.2. Dịch vụ web với MQTT broker: 20](#_Toc459926106)

[CHƯƠNG 4: HỆ THỐNG TRONG THỰC TIỄN 23](#_Toc459926107)

[4.1. Nhà kính: 23](#_Toc459926108)

[4.2. Bộ cảm biến: 23](#_Toc459926109)

[4.3. Dịch vụ web: 24](#_Toc459926110)

[KẾT LUẬN 26](#_Toc459926111)

[TÀI LIỆU THAM KHẢO 27](#_Toc459926112)

# LỜI MỞ ĐẦU

Bước vào cuộc sống đại học, các tân sinh viên đã phải đối mặt với không ít khó khăn, từ việc sống xa nhà, xa gia đình, làm quen với chỗ ăn, chỗ ở mới,… Trong đó, việc tìm được một chỗ ở tốt, thuận tiện cho việc sinh hoạt hằng ngày và yên tâm cho việc học tập quả thật rất khó khăn.

Từ đó, cùng với sự giúp đỡ của thầy hướng dẫn, em đã tiến hành thực hiện đề tài: “Xây dựng website tìm phòng trọ cho sinh viên” tạo ra một cộng đồng tin cậy cung cấp thông tin về các phòng trọ dành riêng cho sinh viên, đáp ứng tốt nhất cho các tiêu chí của sinh viên. Sinh viên, đặc biệt là các bạn tân sinh viên, và gia đình sẽ không còn những nối lo về việc tìm kiếm chỗ ở, tập trung hoàn toàn vào việc học hành.

Nội dung báo cáo được chia thành 5 chương sau:

Chương 1: Đặt vấn đề

Chương 2: Mô hình hoạt động website

Chương 3: Công nghệ sử dụng

Chương 4: Triền khai website

Chương 5: Kết quả đạt được

# CHƯƠNG 1: ĐẶT VẤN ĐỀ

## Phòng trọ cho sinh viên:

Mỗi năm đầu tháng 9, có thêm hàng triệu tân sinh viên trúng cử vào các trường đại học, thế nhưng phòng trọ, nhà ở thì không “nở” ra thêm được, dẫn tới việc tìm kiếm một phòng trọ phù hợp, ưng ý khó khăn. Không những thế, lợi dụng sự khó khăn này, giới cò mồi và chủ phòng trọ còn đưa ra những chiêu trò nâng giá khá cao, đặc biệt tại các phòng trọ gần trường, thế nhưng cơ sở vật chất thì lại không đáp ứng đủ. Một lựa chọn khác nữa thường được các bạn tân sinh viên chọn là ở ký túc xá của trường. Việc ở ký túc xá trường sẽ bảo đảm cho bạn việc học tập tốt hơn, mọi hoạt động, sinh hoạt của trường bạn đều dễ dàng nắm bắt, thế nhưng vì là một lựa chọn “tốt”, nên rất nhiều sinh viên đăng ký, và mỗi phòng thì bạn phải ở chung với ít nhất là 4 5 người bạn nữa. Về lâu dài có thể bạn sẽ cảm thấy khó khăn vì chung đụng với nhiều người, không có không gian riêng và an ninh thì không được đảm bảo bởi rất khó kiểm soát với số lượng sinh viên quá đông trong từng phòng. Vì thế sinh viên lại có xu hướng tìm phòng trọ, nhưng tìm ở đâu, tìm như thế nào, giá cả ra sao, phải tìm được chỗ thuận tiện cho việc học tập, sinh hoạt, … rất nhiểu vấn đề mà các sinh viên băn khoăn khi tìm một phòng trọ.

## Các tiêu chí phòng trọ sinh viên:

Qua số liệu thu thập được, có một số tiêu chí chung mà các bạn sinh viên thường đề cập tới khi quyết định chọn một phòng trọ :

* Địa điểm: càng gần trường học càng tốt, gần chợ, nơi buôn bán
* Giá cả: hợp lý, tiền điện , nước phù hợp với quy định của nhà nước, sử dụng tự do, dùng bao nhiêu trả bấy nhiêu, có đồng hồ đo điện, nước riêng.
* An ninh trật tự: không có tệ nạn, gần các chốt dân phòng, tránh các hẻm quá tối tăm, vắng người mướn.
* Thuê nguyên phòng hay ở ghép: điều này cũng làm khá nhiều bạn sinh viên băn khoăn bởi giá thuê nguyên phòng thì khá cao, tuy nhiên có không gian riêng tư, còn ở ghép thì giá cả các thứ sẽ được chia ra, tuy nhiên phải sống hòa đồng với nhau. Có một trường hợp thuê nữa là ở chung với chủ, tuy khá ít nhưng hình thức này được nhiều bạn sinh viên quan tâm bởi lý do an ninh.
* Cơ sở vật chất: sạch sẽ, có phòng vệ sinh riêng, có thể thêm gác lửng, đủ không gian để sinh hoạt, học tập.
* Dịch vụ : có nơi để rác, người hốt rác, nơi gửi xe, có wifi, internet, tiền chi trả rõ ràng, minh bạch.
* Bạn bè: việc ở chung với bạn cùng trường, cùng lớp hay cùng niên khóa sẽ dễ dàng hơn cho việc hòa nhập với nơi ở mới, đồng thời giúp ích khá nhiều cho việc học cũng như giờ giấc sinh hoạt.

Qua một số tiêu chí kể trên, ta có thể thấy các điểm khác biệt trong việc tìm kiếm phòng trọ của sinh viên so với những đối tượng khác. Vì thế em muốn tạo ra một website dành riêng cho các bạn sinh viên, với những tiêu chí riêng của mình về phòng trọ, là nơi tin cậy, an toàn cho các bạn trong việc lựa chọn cho mình một nơi ở. Website được xây dựng dựa trên ba công nghệ chủ yếu :

* Angular 2: phiên bản Angular thứ hai với sự thay đổi hoàn toàn và hoàn thiện hơn so với phiên bản đầu tiên, hơn thế dự án Angular được “gã khổng lồ” trong ngành công nghệ Google tài trợ, cộng với một cộng đồng lớn mạnh đã có sẵn hứa hẹn sẽ phát triển nhanh và ưu việt hơn cho website.
* JAX-RS: là một JAVA API hỗ trợ trong việc xây dựng một web service hoàn chỉnh theo ngôn ngữ JAVA, cụ thể ở đây là một RESTful web service.
* MySQL: là hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến nhất hiện nay, giao tiếp dễ dàng với các ứng dụng và trong việc triển khai cơ sở dữ liệu lên server.

Chương tiếp theo đây, em xin trình bày mô hình hoạt động của website,

# CHƯƠNG 2: MÔ HÌNH HOẠT ĐỘNG WEBSITE

## 2.1. Tổng quan:

Hệ thống được chia làm ba phần chính:

* + Bộ cảm biến thu thập dữ liệu về môi trường sống của cây trồng trong nhà kính.
  + Dịch vụ web xử lý dữ liệu về môi trường sống của cây trồng trong nhà kính.
  + Cơ sở dữ liệu.

Bộ cảm biến thu thập dữ liệu: các cảm biến về môi trường sống bao gồm: cảm biến nhiệt độ, độ ẩm không khí, cảm biến độ ẩm đất và cảm biến ánh sáng sẽ được điều khiển và tổng hợp lại bởi vi điều khiển TM4C123GH6PM, sau đó vi điều khiển truyền dữ liệu qua module wifi esp8266 để gửi lên IBM Cloud thông qua giao thức MQTT.

Dịch vụ web xử lý dữ liệu: dịch vụ được xây dựng trên kiến trúc REST, nên dịch vụ là dịch vụ web Restful, dịch vụ này có chức năng chính là lắng nghe các gói dữ liệu được gửi lên từ bộ cảm biến, khi nhận được sẽ tiến hành xử lý, so sánh dữ liệu hiện tại với các yêu cầu môi trường sống của cây trồng, và đưa ra quyết định việc tắt, mở các thiết bị đèn chiếu sáng, máy bơm nước trong nhà kính nhằm đảm bảo điều kiện khí hậu, môi trường sống trong nhà kính phù hợp nhất với cây trồng.

Khi hệ thống chạy thì không cần đến sự tác động của con người nữa, công việc sẽ chỉ diễn ra giữa máy với máy, giữa bộ cảm biến thu thập dữ liệu với dịch vụ web xử lý dữ liệu đó, tạo ra tính tự động hóa trong nhà kính, từ đó giúp giảm chi phí lao công, tăng năng suất cây trồng.

Trong các phần tiếp theo của chương, em sẽ trình bày lần lượt cụ thể hơn về cách thức hoạt động, công năng cũng như cách xây dựng của hai phần trong hệ thống: bộ cảm biến và dịch vụ web .

Nhựng trước hết, trong phần tiếp theo ngay sau đây, em xin giới thiệu về mộ thành phần quan trọng không kém quan trọng so với hai thành phần chính ở trên của hệ thống, là thành phần trung gian giữa bộ cảm biến và dịch vụ web, là nơi lưu trữ dữ liệu, ứng dụng, là nơi cung cấp các dịch vụ cần thiết cho việc xây dựng hệ thống, đó chính là IBM Bluemix.

## 2.2. IBM Bluemix

Bluemix là một dịch vụ điện toán đám mây của IBM. Bluemix được thực hiện trên nền tảng Cloud Foundry, một nền tảng dịch vụ mã nguồn mở (open source Platform as a Service). Bluemix tạo ra các dịch vụ mà người dùng có thể dễ dàng tích hợp vào các ứng dụng của mình.

Bluemix hiện cung cấp khoảng 150 dịch vụ cho người dùng, thuộc mọi lĩnh vực như điện toán đám mây, máy chủ ảo, phân tích và thiết kế database, lưu trữ dữ liệu, DevOps, mobile, web, dịch vụ web, trí tuệ nhân tạo, IoT (Internet of Thing),… đồng thời Bluemix còn cho phép xây dựng hệ thống dựa trên các ứng dụng có sẵn, các database bên ngoài có thể kết nối với các ứng dụng trong Bluemix, tất cả chỉ cần tích hợp các dịch vụ mà Bluemix cung cấp sẵn vào ứng dụng là đã có thể tạo ra một hệ thống hoàn chỉnh được xây dựng hoàn toàn trên Cloud một cách nhanh chóng, dễ dàng nhất.

Trong phạm vi đề tài tốt nghiệp này, em có sử dụng ba dịch vụ của Bluemix:

* + API Connect: dịch vụ giúp tạo, hay tích hợp vào ứng dụng để trở thành một dịch vụ Web API dễ dàng. Dịch vụ web đề xử lý dữ liệu về môi trường sống của cây trồng trong nhà kính đang được tích hợp với dịch vụ này để tạo ra dịch vụ Web API trên Bluemix.
  + ClearDB MySQL Database: dịch vụ này được cung cấp bởi bên thứ ba, không nằm trong hệ thống của IBM, tuy nhiên dịch vụ này vẫn được hỗ trợ đầy đủ, tối đa như những dịch vụ khác. Dịch vụ này được tích hợp với ứng dụng phpmyadmin nhằm tạo ra môi trường trên web lưu trữ, sửa đổi hoặc tạo mới database, cụ thể hơn là cho MySQL.
  + Internet of Things Platform: dịch vụ này là tối quan trọng cho đề tài thực tập, nó đóng vai trò là một broker trong giao thức mqtt, sẽ được giới thiệu trong các phần sau, giúp cho bộ cảm biến có thể gửi gói dữ liệu , đồng thời nhận gói tín hiệu điều khiển ngược lại từ dịch vụ web, mặc dù cả dịch vụ web và bộ cảm biến đều kết nối, gởi hay nhận gói dữ liệu thông qua dịch vụ này chứ không hề biết sự tồn tại của nhau.

Phần trình bày ở trên đã cho thấy nhiệm vụ quan trọng của IBM Bluemix trong mô hình hệ thống của đề tài thực tập này, về cách thức giao tiếp của hai phần chính trong hệ thống với nhau. Thế nhưng để hai bộ phận chính của hệ thống giao tiếp được với IBM Bluemix, đề tài thực tập sử dụng giao thức MQTT để truyền tải thông tin giữa hệ thống với IBM Bluemix, và phần tiếp sau đây, em xin trình bày rõ hơn về giao thức truyền tin MQTT, lý do vì sao lại chọn MQTT mà không phải các giao thức khác.

## 2.3. Giao thức truyền tin MQTT:

### 2.3.1. Dạng truyền tin kiểu publish / subscribe:

Dạng truyền tin kiểu publish / subscribe tách client (bên gửi tin, hay nhận tin) thành hai đối tượng gọi là: publisher (bên gửi tin) và subscriber (bên nhận tin). Hai đối tượng này sẽ không biết sự hiện diện của nhau, nghĩa là giữa chúng sẽ có một thành phần thứ ba, gọi là broker, publisher hay subscriber sẽ kết nối, gởi hoặc nhận tin với broker này. Broker có nhiệm vụ nhận mẫu tin từ publisher, lọc theo yêu cầu và gởi đi các mẫu tin tương ứng với mỗi subscriber.

Đặc điểm chính của dạng truyền tin này là sự tách biệt giữa các bên gửi hay nhận tin, tạo ra hai đối tượng là publisher và subscriber:

* + Tách biệt về không gian: publisher và subscriber không cần biết thông tin về nhau, cụ thể như địa chỉ IP hay port của nhau.
  + Tách biệt về thời gian: publisher và subscriber không nhất thiết phải chạy cùng nhau.
  + Tách biệt về đồng bộ hóa: các hoạt động của publisher sẽ không bị gián đoạn bởi hoạt động của subscriber và ngược lại.

Vì có sự tách biệt giữa các bên gửi, nhận tin, nên nhiệm vụ của broker sẽ nặng nề hơn, bởi broker chính là thành phần kết nối với tất cả, vừa kết nối với subscriber, vừa phải kết nối với publisher, đồng thời theo yêu cầu bên subscriber, phải lọc ra các mẫu tin phù hợp, broker lọc các mẫu tin thường theo những yếu tố sau:

* + Lọc theo chủ đề: chủ đề thường được xây dựng theo kiểu string, có cấu trúc phân cấp, ví dụ như: nhakinh/nhietdo hay nhakinh/thiebi/trangthai,… Chủ đề nằm trong nội dung của các mẫu tin, chủ đề thường không quá dài, vì thế việc lọc tương đối nhanh và dễ dàng. Đây là bộ lọc chủ yếu của giao thức truyền tin MQTT.
  + Lọc theo nội dung mẫu tin: lọc theo nội dung có một khuyết điểm lớn là nội dung của mẫu tin broker phải được biết trước, không thể mã hóa hay thường xuyên thay đổi, vì rất khó cho broker trong việc lọc.
  + Lọc theo kiểu của mẫu tin: Khi thiết kế theo hướng đối tượng, các mẫu tin có thể có các kiểu như Exception chẳng hạn.

Sự tách biệt giữa việc nhận và gửi mẫu tin dù mang lại một số lợi ích nhưng cũng có các mặt hạn chế, chẳng hạn như: subscriber phải biết trước cấu trúc phần dữ liệu trong mẫu tin để có thể xử lý tốt được, nếu subscriber yêu cầu lọc theo chủ đề, thì cả hai bên publisher và subscriber đều phải biết chính xác tên chủ đề, thêm một hạn chế nữa là các mẫu tin được gửi lên thì không chắc sẽ được gửi cho đúng bên, đúng người, publisher không thể nào biết được mẫu tin có đến được tới bên cần nhận hay không.

MQTT được xây dựng dựa trên dạng truyền tin publish / subscribe này, thế nhưng với thêm một số đặc điểm khác nhằm hạn chế bớt các mặt yếu kém của nó.

### 2.3.2. Khái quát về MQTT:

MQTT (MQ Telemetry Transport) được phát triển bởi IBM, xây dựng trên lớp TCP/IP, dạng truyền tin kiểu publish / subscribe. MQTT là giao thức truyền tin dành riêng cho các thiết bị nhúng, thiết bị cảm biến, vi điều khiển, các nơi bị giới hạn về dung lượng băng thông, về năng lượng. MQTT là giao thức truyền tin cho các hệ thống Internet of Things bởi tính gọn nhẹ của mẫu tin, ít tiêu tốn năng lượng, có thể truyền cho nhiều bên cùng một lúc.

MQTT hiện nay đã lên tới phiên bản v3.1.1 theo chuẩn OASIS.

Các đặc điểm chính của MQTT:

* + MQTT xây dựng trên lớp TCP/IP.
  + MQTT xây dựng theo dạng truyền tin kiểu publish / subscribe, nhưng broker chỉ chủ yếu sử dụng bộ lọc chủ đề.
  + Cũng như dạng truyền tin kiểu publish / subscribe, MQTT cũng chia các bên gởi, nhận mẫu tin thành publisher và subscriber, được gọi chung là client, thành phần trung gian là broker.
  + Broker MQTT ngoài việc nhận, lọc và gởi các mẫu tin phù hợp với chủ đề mà subscriber mong muốn, nó còn giữ session của các subscriber, giữ lại các chủ đề muốn nhận cũng như các mẫu tin chưa nhận được khi đang ngoại tuyến, broker còn có thể nhận dạng và phân quyền người dùng.

MQTT được phát triển gọn nhẹ, ít tiêu hao năng lượng, được sử dụng cho những trường hợp đòi hỏi tiêu tốn ít tài nguyên, một phần là bởi các mẫu tin của MQTT rất gọn nhẹ, các loại mẫu tin của MQTT gồm: CONNECT, CONNACK, PUBLISH, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, … Với mỗi mẫu tin, công dụng và nội dung cũng tương đối khác nhau, tuy nhiên thường chứa các trường sau:

* + clientId: broker dùng để nhận dạng client, và xem xét trạng thái của client, client có còn session hay không.
  + packetId: broker nhận biết các mẫu tin khác nhau trong một lần kết nối với client, vì thế packetId có thể lặp lại trong các lần kết nối khác. Nếu QoS là 0 thì packetId luôn có giá trị 0.
  + topicName: tên chủ đề, cấu trúc phân cấp.
  + payload: nội dung của mẫu tin, có thể gửi bất kì như hình ảnh, dạng văn bản, dạng JSON,…
  + cleanSession: biến cờ dùng để thông báo client có muốn lưu trạng thái của mình hay không. Nếu biến này mang giá trị true: broker sẽ xóa sạch mọi thông tin về client từ các session trước. Ngược lại nếu biến mang

giá trị false: broker sẽ xem đã lưu session của client này hay chưa, nếu chưa thì tạo session mới, lưu mọi chủ đề mà client muốn lắng nghe và các mẫu tin client chưa nhận được khi đang ngoại tuyến; nếu đã có session thì tiến hành gởi các mẫu tin mà client chưa nhận được, nếu client muốn tạo session với broker, thì để broker thông báo đã tạo session thành công hay chưa, trong gói CONNACK sẽ chứa trường sessionPresent giá trị true là thành công và false là thất bại hoặc client không muốn tạo.

* + retainFlag: biến cờ dùng để đánh dấu mẫu tin này là mẫu tin đúng nhất, mới nhất của một chủ đề,và được broker lưu lại. Khi bất kỳ một subscriber nào đăng ký chủ đề này, thì lập tức broker sẽ gửi mẫu tin có retainFlag này tới cho subscriber mà không cần đợi publisher gởi một gói tin nào lên trước.
  + dupFlag: đánh dấu mẫu tin có phải được gởi lại hay không, được sử dụng khi QoS khác 0.
  + Username / Password: broker dùng để nhận dạng và phân quyền client.
  + Will messages: khi client gửi nội dung này lên, broker sẽ lưu thông tin này cho đến khi client bị mất kết nối không theo ý muốn, broker sẽ gửi cho toàn bộ các client khác thông tin này, chức năng này thường được sử dụng để biết tình trạng hiện tại của một thiết bị, ứng dụng nào đó. Như thế nào là một client ngắt kết nối không mong muốn:
    - Lỗi I/O hoặc lỗi mạng do broker phát hiện được.
    - Client không kết nối lại với broker trong khoảng thời gian sống (keep alive time).
    - Client đóng kết nối nhưng không gởi mẫu tin DISCONNECT trước khi đóng.
    - Server đóng kết nối bởi lỗi giao thức.
  + keepAlive: khoảng thời gian mà broker chờ nhận một mẫu tin từ client, nếu client muốn duy trì kết nối với broker nhưng chưa có dữ liệu gì để gởi lên thì có thể gởi gói PING lên cho broker nhằm reset lại khoảng thời gian chờ này.
  + returnCode: thường nằm trong các gói xác nhận của broker như CONNACK, SUBACK,… để thông báo công việc có thành công hay không.

Để hạn chế sự yếu kém của dạng truyền tin publish / subscribe, MQTT đã thêm một đặc tính nữa là QoS. QoS được phân biệt giữa publisher và subscriber, nghĩa là mặc dù publisher gởi gói tin lên broker theo mức 2, thì subscriber vẫn có thể nhận gói tin theo mức 0 hoặc 1, có ba mức QoS sau:

* + QoS 0 (At most once): chỉ gởi một lần duy nhất đến bên nhận, không quan tâm đến việc bên nhận có lấy được mẫu tin hay không.
  + QoS 1 (At least once): bên gởi sẽ gởi gói liên tục sau một khoảng thời gian nếu chưa nhận được gói xác nhận từ bên nhận. Ở đây packetId có

công dụng là xác nhận, nếu bên nhận đã nhận được mẫu tin thì trả về mẫu tin xác nhận chỉ cần chứa packetId của mẫu tin đó.

* + QoS 2 (Exactly once): đảm bảo gói tin được truyền đến bên nhận, cách thức truyền theo QoS 2 được chia thành hai giai đoạn:
    - Giai đoạn 1: bên gởi gởi mẫu tin cho bên nhận và đợi mẫu tin xác nhận lại, khi nhận được mẫu tin xác nhận, bên gởi xóa mẫu tin gởi đi, lưu mẫu tin xác nhận.
    - Giai đoạn 2: bên gởi lúc này gởi gói xác nhận qua cho bên nhận, bên nhận khi đã thấy gói xác nhận này thì đã có thể yên tâm xóa hết các thông tin cho việc truyền thông tin này, cuối cùng gởi thêm một gói xác nhận nữa cho bên gởi. Lúc này bên gởi nhận gói xác nhận cuối cũng sẽ xóa hết các thông tin còn lại của việc truyền thông tin này, kết thúc việc truyền tin.

MQTT vừa thừa kế được những đặc điểm nổi bật của dạng truyền tin publish / subscribe như tính gọn nhẹ, truyền được cho nhiều đối tượng, có sự tách biệt, độc lập giữa các client, đồng thời MQTT bổ sung thêm một số tính năng nhằm khắc phục các hạn chế của dạng truyền tin này như QoS, lưu session, lưu mẫu tin có retainFlag. Vì thế MQTT rất được ưa dùng trong các công việc có môi trường khó khăn về dung lượng băng thông, năng lượng.

Bộ khung của hệ thống thông tin cây trồng trong nhà kính được xây dựng dựa trên IBM Bluemix và giao thức MQTT đã được nói ở trên, phần tiếp theo của chương sẽ đề cập cụ thể đến hai bộ phận chính của hệ thống là: bộ cảm biến thu thập dữ liệu và dịch vụ web xử lý dữ liệu về mội trường sống của cây trồng trong nhà kính.

## 2.4. Bộ cảm biến thu thập dữ liệu:

Bộ cảm biến thu thập dữ liệu về môi trường sống của cây trồng trong nhà kính gồm các thành phần sau:

* + Vi điều khiển TM4C123GH6PM: điều khiển mọi hoạt động thu thập dữ liệu về môi trường sống của cây trồng trong nhà kính. Đồng thời là bộ phận chuyển các dữ liệu đó qua module wifi esp8266 để gởi lên IBM Bluemix.
  + Cảm biến độ ẩm đất (YL-100): cảm biến có hai ngõ ra là D0 hoặc A0. Với ngõ ra D0 thì luôn ở mức 0, khi đất thiếu nước ngõ ra D0 cho ra tín hiệu mức cao, độ nhạy của cảm biến có thể được điều chỉnh bởi chiết áp gắn trên mạch. Với ngõ ra A0, tín hiệu cho ra là analog, sẽ cho ra giá trị độ ẩm đất thời điểm hiện tại, vi điều khiển có thể lấy giá trị ở ngõ ra A0 này để so sánh với yêu cầu về độ ẩm trong đất của cây trồng.
  + Cảm biến nhiệt độ và độ ẩm không khí (DHT11): dữ liệu ngõ ra của cảm biến là dạng số, có thể dùng vi điều khiển để lấy dữ liệu một cách dễ dàng. Dữ liệu độ ẩm mà cảm biến đo được từ 20% - 90%, sai số 5%.

Nhiệt độ đo được trong khoảng 0 – 50oC, sai số 2oC. Thời gian trả dữ liệu thấp hơn 50ms.

* + Cảm biến ánh sáng (BH1750): cảm biến đo cường độ ánh sáng với dữ liệu trả về theo đơn vị Lux. Cảm biến có độ phân giải cao: 1 – 65535lx. Độ ảnh hưởng của ánh sáng hồng ngoại lên cảm biến rất nhỏ. Khả năng chống nhiễu sáng ở tần số 50 / 60Hz.
  + Module wifi esp8266: module wifi kết nối với các thiết bị khác thông qua cổng giao tiếp UART. Module wifi esp8266 được nạp sẵn firmware chứa các tập lệnh AT, tuy nhiên để sử dụng cho đề tài thực tập này, module wifi esp8266 được nạp firmware NodeMCU, firmware được lập trình và phát triển theo ngôn ngữ Lua, giúp cho người sử dụng dễ dàng hơn trong việc sử dụng, thiết lập chức năng cho esp8266, thay vì sử dụng tập lệnh AT được nạn sẵn trước đó.
  + Module relay 2 kênh opto cách ly: module gồm 2 relay hoạt động ở điện áp 5VDC, chịu được hiệu điện thế lên đến 250VAC 10A. Trên module được lắp sẵn mạch kích relay sử dụng transistor và IC cách ly quang giúp cách ly hoàn toàn mạch điều khiển với rơ le, bảo đảm vi điều khiển hoạt động ổn định. Relay có thể được kích ở mức cao (5V) hoặc mức thấp (0V), tùy theo yêu cầu sử dụng mà điều chỉnh jumper để định mức kích.

Với các cảm biến trên được nối với vi điều khiển TM4C123GH6PM, vi điều khiển sau khi lấy đủ dữ liệu từ các cảm biến thì chuyển dữ liệu đó sang module wifi esp8266 bằng cổng UART. Với việc nạp firmware NodeMCU cho module wifi esp8266, việc lập trình cho module kết nối, nhận cũng như gởi dữ liệu qua cổng giao tiếp UART trở nên dễ dàng hơn, đồng thời firmware cũng cung cấp một số thư viện, trong đó có thư viện hỗ trợ giao thức MQTT, nhờ đó việc gởi dữ liệu lên IBM Bluemix thông qua giao thức MQTT trở nên tiện lợi, nhanh chóng, chính xác và hiệu quả.

Khi module wifi esp8266 đã gởi dữ liệu thành công, thì lập tức chuyển sang trạng thái lắng nghe các chủ đề đăng ký được lập trình trước, khi có bất kỳ dữ liệu nào được gửi tới, module sẽ chuyển toàn bộ dữ liệu sang vi điều khiển xử lý và điều khiển các thiết bị tương ứng nhằm đảm bảo điều kiện sống tốt nhất cho cây trồng.

Việc đưa dữ liệu về môi trường sống của cây trồng trong nhà kính đã được bộ cảm biến thực hiện, thế nhưng để hệ thống có thể tự động hóa, các hoạt động diễn ra đều từ máy tới máy, hạn chế sự can thiệp của con người thì việc xử lý dữ liệu được đưa lên này cũng không kém phần quan trọng. Phần tiếp nối sau đây, em xin trình bày về bộ phận xử lý dữ liệu được thu thập bởi bộ cảm biến, dịch vụ web api.

## 2.5. Dịch vụ web xử lý dữ liệu:

Bộ cảm biến thu thập dữ liệu và đưa lên IBM Bluemix nhằm cập nhật tình trạng hiện thời của môi trường sống trong nhà kính, thế nhưng nó vẫn chỉ là dữ liệu thô, và phải nhờ tới bộ phận thứ hai trong hệ thống là dịch vụ web, dữ liệu mới được xử lý, lưu vào cơ sở dữ liệu và trả về tín hiệu điều khiển thiết bị cho vi điều khiển bên dưới.

Dịch vụ web được viết theo ngôn ngữ Java, xây dựng dựa trên cấu trúc REST, vì thế còn được gọi là dịch vụ web RESTful. Để hiểu được cấu trúc, cách vận hành của dịch vụ web này, trước hết ta đi tìm hiểu thế nào được gọi là một hệ thống REST.

### 2.5.1. Cấu trúc REST:

Cấu trúc REST (Representational State Transfer): sử dụng phương thức HTTP để giao tiếp, truyền thông tin. Cấu trúc này coi mọi thứ đều là tài nguyên, và các tài nguyên sẽ được truy xuất thông qua các giao thức chuẩn của HTTP (GET, POST, PUT, DELETE,…).

Khi áp dụng đầy đủ các ràng buộc và quy ước sau vào hệ thống, ta sẽ có một hệ thống REST, hay gọi là hệ thống RESTful:

* + Hoạt động theo mô hình client – server: client và server được phát triển độc lập với nhau, giao tiếp với nhau thông qua các interface định sẵn, mỗi tài nguyên sẽ được chỉ định bởi một URI do server cung cấp.
  + Phi trạng thái: server sẽ không lưu bất kỳ thông tin, trạng thái nào của client, vì thế mỗi khi client có yêu cầu sử dũng một dịch vụ của server thì phải đóng gói đầy đủ các thông tin cần thiết của mình về cho server.
  + Có khả năng cache: hệ thống có thiết bị trung gian đóng vai trò lưu trữ tạm các dữ liệu đã được yêu cầu trước đó, vì thế client có thể lấy dữ liệu một cách nhanh chóng, đồng thời giảm tần suất gọi dịch vụ trên server của client.
  + Tính phân lớp: Hệ thống được chia ra thành từng lớp, trong đó các lớp kề nhau thì giao tiếp với nhau, không giao tiếp với các lớp khác, các lớp dưới thường cung cấp dịch vụ phục vụ cho các lớp trên.
  + Có các giao tiếp thống nhất: đây là ràng buộc quan trọng và cơ bản của một hệ thống REST, sự thống nhất về các giao tiếp giúp client và server có thể được phát triển độc lập với nhau, dù cho client là mobile, web, hay một thiết bị nhúng bất kỳ đều có thể gọi dịch vụ trên server thông qua các giao tiếp thống nhất này.

Một hệ thống RESTful phải có các ràng buộc ở trên, tuy nhiên đối với một dịch vụ web RESTful ngoài những ràng buộc ở trên, dịch vụ web còn phải đáp ứng một số nguyên tắc thiết kế khác sẽ được trình bày trong phần tiếp theo đây.

### 2.5.2. Dịch vụ web RESTful:

Một dịch vụ web RESTful cần tuân thủ theo bốn nguyên tắc thiết kế sau đây:

* + Sử dụng các phương thức chuẩn của HTTP một cách rõ ràng: các phương thức chuẩn của HTTP như GET, POST, PUT, DELETE,… phải được sử dụng theo đúng định nghĩa bởi RFC 2616. Nguyên tắc thiết kế này thiết lập một ánh xạ 1-1 giữa các phương thức của HTTP với các thao tác tạo, đọc, sửa và xóa dữ liệu (CRUD):
    - GET: truy xuất một tài nguyên.
    - POST: tạo mới một tài nguyên.
    - PUT: cập nhật, thay đổi trạng thái của một tài nguyên.
    - DELETE: xóa một tài nguyên.
  + Hệ thống phi trạng thái: nhằm đáp ứng ràng buộc của một hệ thống REST.
  + Hiển thị cấu trúc thư mục giống với các URI: Cấu trúc các thư mục, nơi đặt các tài nguyên, phải được sắp xếp theo đúng đường dẫn URI mà server chỉ định cho tài nguyên đó. Một số lưu ý về cấu trúc địa chỉ URI:
    - Chữ thường.
    - Thay thế khoảng trống bằng gạch nối (-) hay gạch chân ( \_ ).
  + Sự chuyển đổi sang JSON (Javascript Object Notation) và XML: các dịch vụ sẽ trả về kết quả cho client kiểu dữ liệu thuộc dạng JSON hoặc XML, mặc dù dữ liệu khi được lấy lên từ database không thuộc cả JSON và XML. Điều này giúp cho các ứng dụng, thiết bị bên client, được lập trình với mọi ngôn ngữ, đều có thể sử dụng dịch vụ và lấy được dữ liệu.

### 2.5.3. Cách thức hoạt động của dịch vụ web:

Dịch vụ web xử lý dữ liệu được chia làm hai phần:

* + Xử lý với cơ sở dữ liệu: dịch vụ web sẽ gọi các store procedure trong cơ sở dữ liệu và chỉ cần lấy kết quả về, chuyển đổi sang kiểu dữ liệu JSON và trả về cho client.
  + Xử lý với MQTT: dịch vụ web sẽ đăng ký và lắng nghe các sự kiện theo chủ đề người dùng mong muốn, nếu có bất kỳ gói tin nào được gởi lên, thường là các gói tin về dữ liệu môi trường trong nhà kính thời điểm hiện tại, dịch vụ web sẽ cập nhật vào cơ sở dữ liệu tình trạng hiện tại của nhà kính, trả về cho client các giá trị nhận được theo kiểu dữ liệu JSON, đồng thời thực hiện các tính toán nhằm điều khiển các thiết bị hoạt động tạo môi trường sống tốt nhất cho cây trồng: so sánh các giá trị này với các tiêu chuẩn được đặt sẵn dành riêng cho từng cây trồng, nếu giá trị nằm trong tiêu chuẩn, sẽ không thay đổi trạng thái của thiết bị trong nhà kính, ngược lại nếu các giá trị vượt ngưỡng hoặc thấp hơn mức quy định, dịch vụ web sẽ điều chỉnh hợp lý lại các thiết bị theo độ

ưu tiên kiểm soát độ ẩm đất, nhiệt độ, ánh sáng và cuối cùng là độ ẩm không khí.

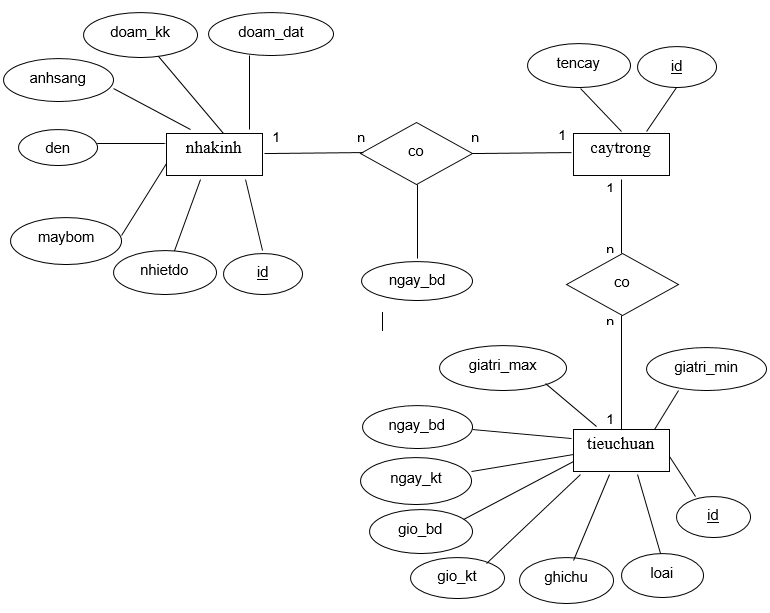
Với mô hình hệ thống thông tin môi trường sống của cây trồng trong nhà kính đã được giới thiệu ở trên, việc triển khai và lắp đặt hệ thống này vào thức tế như thế nào, gặp những bất cập gì, chương tiếp theo đây em xin đề cập đến vấn đề đó.

## 2.6. Cơ sở dữ liệu:

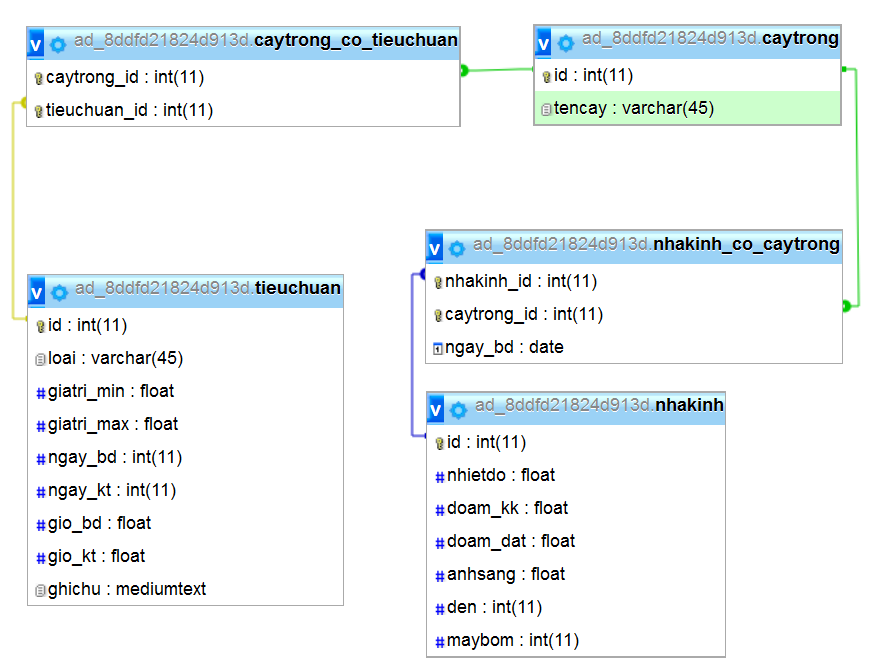
Cơ sở dữ liệu được xây dựng với hệ quản trị cơ sở dữ liệu: MySQL

Cơ sở dữ liệu gồm ba thực thể chính: nhà kính, cây trồng và tiêu chuẩn. Từ các thực thể đó, ta có năm bảng sau:

* + Nhakinh: id, nhietdo, doam\_kk, doam\_dat, anhsang, den, maybom.
  + Caytrong: id và tencay.
  + Tieuchuan: id, loai, giatri\_min, giatri\_max, ngay\_bd, ngay\_kt, gio\_bd, gio\_kt, ghichu.
  + Caytrong\_co\_tieuchuan: caytrong\_id, tieuchuan\_id.
  + Nhakinh\_co\_caytrong: nhakinh\_id, caytrong\_id, ngay\_bdMô hình thực thể - quan hệ của cơ sở dữ liệu như sau:

Bảng 2.1: Mô hình thực thể - quan hệ

Cơ sở dữ liệu được sinh ra trong MySQL như sau:



Bảng 2.2: Cơ sở dữ liệu

# CHƯƠNG 3: CÔNG NGHỆ SỬ DỤNG

## 3.1. AngularJS:

### 3.1.1. AngularJS là gì?

AngularJS là một framework được bảo trợ bởi Google cho phép xây dựng các ứng dụng phía client bằng HTML và Javascript. Trong khi HTML được thiết kế nhằm xây dựng các chương trình tĩnh, website tĩnh, thì AngularJS chính là công cụ cần thiết mà HTML hướng tới để xây dựng các chương trình , website động.

Vậy thế nào là một website tĩnh, và thế nào là một website động ?

Một website tĩnh là một trang web mà nội dung truyền tải của nó tới người dùng sẽ không thay đổi một khi đã được tạo.

Một website động là một trang web mà nội dung của nó sẽ được thay đổi mỗi khi người dùng truy cập tới.

Để tạo một website động từ website tĩnh được xây dựng từ HTML, Javascript được thiết kế để làm điều đó, cùng với đó là việc sử dụng kết hợp nhiều thư viện hay framework khác nhau như : Jquery, Ember. Thế nhưng với AngularJS, nó thay đổi và thêm mới trực tiếp vào cấu trúc của HTML, AngularJS tạo ra các “directive” dùng để chỉ thị việc hiển thị nội dung động một cách dễ dàng, AngularJS tạo ra các “service” thực hiện các logic của chương trình, đồng thời giao tiếp với server, AngularJS tạo ra các “component” để nhóm lại các thành phần liên quan với nhau giúp cho việc lập trình rõ ràng, dễ sử dụng, dễ kiểm tra, dễ triển khai, dễ bảo trì và dễ dàng cho việc sử dụng lại.

Hiện nay AngularJS được phát triển lên phiên bản 2, tuy nhiên phiên bản 2 này lại không hoàn toàn là bản nâng cấp từ phiên bản 1, mà gần như thay đổi hoàn toàn cấu trúc cũng như nâng cao đáng kể về hiệu suất. AngularJS 2 được phát triển với mục tiêu chạy trên mọi nền tảng, đặc biệt hướng tới mobile trong khi AngularJS 1 thì không, AngularJS 2 được xây dựng với nhiều ngôn ngữ hơn không chỉ Javascript thuần nữa mà mở rộng hơn với TypeScript, Dart , hoặc Javascript viết theo chuẩn ES5, ES6, nhờ đó AngularJS 2 được hỗ trợ tốt hơn từ các IDE và plugin được xây dựng cho các ngôn ngữ này. Trong đề tài này, em xin trình bày phiên bản 2 của AngularJS mà bỏ qua phiên bản 1 do sự khác biệt rất lớn này.

### 3.1.2. Kiến trúc AngularJS 2:

Kiến trúc AngularJS 2 được xây dựng dựa trên 8 thành phần chính:

- Modules

- Components

- Templates

- Metadata

- Data binding

- Directives

- Services

- Dependency injection

Modules: Angular Module là một class thông thường được gắn với decorator @NgModule.

Decorator @NgModule được sử dụng nhằm khai báo cho AngularJS biết rằng class này được xử lý như thế nào so với các phần tử khác cũng đều là một class thông thường trong Javascript như : directive, pipe, component,… Decorator là một phương thức dùng metadata để thực thi. Khi một class được gán decorator @NgModule, thì phải khai báo thêm các thuộc tính cần thiết trong metadata mà @NgModule dùng để phân biệt các Module với nhau, các thuộc tính gồm:

- declarations: gồm các component, directive và pipe thuộc module.

- imports: gồm các module khác bên ngoài cần sử dụng thêm.

- exports: tập con của declarations, dùng để khai báo các thành phần được public để các module khác có thể sử dụng được.

- providers: gồm các service được các component sử dụng trong modules.

- bootstrap: xác định component nào sẽ là component bắt đầu.

Module được dùng để chứa tất cả thành phần của một ứng dụng gọi là root module, và tất cả các ứng dụng đều có ít nhất một module này, trong đó có thể bao gồm nhiều module nhỏ hơn bao gồm các component, directive , service có liên quan đến nhau, đến một logic, đến một trang của website,… nhằm giúp cho việc tái sử dụng dễ dàng, các đoạn code rõ ràng hơn, dễ thay đổi, dễ kiểm tra, dễ đọc.

Components: quản lý một “view” của webiste, vẫn là một class thông thường trong Javascript và được gán với decorator @Component, mỗi decorator @Component khai báo cần một số thuộc tính sau:

- Selector: tag tùy chỉnh AngularJS, là nơi đặt template của component vào đó.

- Template: được viết dựa trên HTML, AngularJS dựa trên template để hiển thị view.

- Style: chứa các CSS để trang trí cho view

Template và các biến, phương thức được khai báo trong Class Component sẽ tương tác với nhau thông qua Data Binding.

Metadata: một thành phần quan trọng, nó cho AngularJS biết cách thức xử lý với các class thông thường với vai trò khác nhau, ví dụ như Module, Component, Service, …. Tất cả đều là class Javascript thông thường cho đến khi được gán thêm với decorator lên chúng, và decorator sẽ dùng các metadata để định nghĩa về class đó. Ví dụ về metadata:

@Component({

selector: ‘app-test’,

templateUrl: ‘app-test.component.html’,

styleUrls: [‘app-test.component.css’]

})

Trong đó:

- selector: ghi theo selector của CSS báo cho AngularJS vị trí sẽ đặt Component, cụ thể là template vào đó, nghĩa là với mỗi nơi có cặp tag <app-test></app-test> thì Component sẽ được đặt vào trong nó.

- templateUrl: địa chỉ tương đối dẫn đến file HTML template của Component.

- styleUrls: các địa chỉ tương đối dẫn tới các file CSS của template.

Data binding: như đã nói ở trên, Template sẽ giao tiếp với các biến, các phương thức trong Component thông qua dịch vụ Data Binding. Data binding gồm hai dạng chính là một chiều và hai chiều, trong đó một chiều có các dạng sau:

Template -> Component: truyền dữ liệu từ Template vào một biến của component hoặc gọi một phương thức trong Component, ví dụ : (click)=”handle()”, được đặt như các thuộc tính của một tag HTML,

Component -> Template: truyền dữ liệu từ Component tới Template, có hai cách để Component có thể truyền dữ liệu cho Template:

- Sử dụng hai dấu ngoặc nhọn: {{var}}, trong đó var là một biến được khai báo trong Component.

- Property Binding: với những thuộc tính của một tag HTML thông thường như value, id, src,… Ta chỉ cần thêm dấu dấu ngoặc vuông [value]=”var”, với biến var được khai báo trong Component thì thuộc tính value của tag HTML sẽ có giá trị của biến var.

Data binding hai chiều có dạng như sau:

Template <--> Component: đặt [(ngModel)]=”var” với biến var được khai báo trong Component trong một số tag HTMl như input, textarea để có thể lấy được giá trị của tag đó trong Component theo thời gian thực mỗi khi người dùng thay đổi giá trị của phần tử đó.

Directives: là một class được gán với decorator @Directive, Component thật ra chính là một Directive có thêm template. Ngoài Component, Directive còn có hai dạng nữa là : structural và attribute directives.

- Structural directive: thay đổi layout bằng cách thêm, xóa hoặc sửa các phần tử trong cây DOM của HTML.

Ví dụ: <li \*ngIf=”selectedLi”></li>

\*ngIf là một structural directive được xây dựng sẵn của AngularJS, mục đích là để thêm hoặc xóa phần tử trong DOM.

- Attribute directive:

Ví dụ: <img [src]=”srcImage”>

Attribute directive trông giống attribute của các tag HTML với thêm hai dấu ngoặc vuông, directive này thường dùng để lấy giá trị từ các biến trong Component gán vào giá trị các thuộc tính của tag HTML, nhờ đó các giá trị của thuộc tính có thể được thay đổi thường xuyên.

Services: bất kỳ một phương thức, tính năng nào mà chương trình cần đều có thể là service. Service là một class thông thường, được tạo ra nhằm phục vụ một mục đích nhất định và duy nhất. Ví dụ: logging service, data service.… Service thường được gán với decorator @Injector với mục đích sử dụng cho Dependency Injection.

Dependency Injection (DI): DI giúp tạo instance của một class mà không cần phải khai báo new. Đa số được sử dụng với các Service. Bằng cách gán decorator @Injectable lên class Service, mỗi khi Component cần sử dụng Service này, ta chỉ cần khai báo ở tham biến của hàm constructor của Component là đã có thể sử dụng được, ví dụ: constructor(private service: Service) - khai báo biến service với kiểu là class Service, khi đó ở bất kỳ nơi nào trong Component, ta đều có thể sử dụng ngay service này mà không cần phải tạo instance cho nó nữa (nghĩa là service = new Service()).

Ngoài 8 thành phần chính, trong đề tài này, em còn sử dụng thêm một số tính năng khác như: pipe, router, http service.

Pipe: dùng để định dạng lại giá trị hiển thị trên template HTML, Ví dụ: {{varDate | date:”dd/MM/yy”}} - | được gọi là pipe operator, và date là một pipe được định nghĩa sẵn từ AngularJS, với biến varDate là một biến trong Component, giả sử biến varDate có giá trị là April 10, 1998, khi sử dụng pipe với định dạng hiển thị là dd/MM/yy, thì giá trị của biến varDate sẽ được hiển thị : 10/04/98.

Router: dùng trong việc điều hướng giữa các view trong website với nhau, router định nghĩa từng đường dẫn tới mỗi view, tương ứng với một Component xác định, ví dụ :

const appRoutes: Routes = [{

path: ‘test’, component: testComponent

}];

Http Service: dùng để lấy dữ liệu từ server, cụ thể ở đây là sử dụng các phương thức GET, POST, PUT, DELETE có trong API của Web Service để lấy dữ liệu hiển thị lên website.

Tổng hợp lại, với kiến trúc đã được trình bày như trên, phía client của website được tạo như sau:

- Để lấy dữ liệu, giao tiếp với server: Http Service và một số service tự tạo thực hiện logic trả về dữ liệu.

- Để hiển thị dữ liệu lên website:

+ Template: được viết bằng HTML và thêm các cú pháp mới được AngularJS biên soạn thêm để giao tiếp trực tiếp với Component, nhằm lấy giá trị cũng như cập nhật giá trị các biến trong Component, nhờ vào cơ chế Data binding.

+ Style: là các CSS được viết để trang trí cho từng template.

- Để đóng gói từng view, từng trang của website, ta sử dụng Component. Mỗi Component chứa một template, nhiều style cho template đó, và một selector nơi đặt template.

- Các Component sẽ sử dụng các Service, và nhờ cơ chế DI giúp cho việc sử dụng dễ dàng hơn, không còn phải khai báo các instance của Service mỗi khi dùng, chỉ việc khai báo một lần trong dãy tham biến của hàm constructor.

- Để điều hướng qua lại giữa các Component, các view, ta sử dụng Angular Router.

- Modules được dùng để đóng gói tất cả các Component của một chương trình, được dùng để khai báo các Service được sử dụng chung cho toàn bộ Component, được dùng để khai báo Component cha được khởi tạo đầu tiên.

- Và cuối cùng là Metadata đi theo với các Decorator nhằm xác định các class được tạo ra , class nào là Module, class nào là Component, class nào là Directive,…

## 3.2. Công nghệ JAX-RS:

Vi điều khiển sẽ kết nối và lấy toàn bộ dữ liệu từ các cảm biến, sau đó truyền dữ liệu sang cho module wifi esp8266 gởi lên IBM Bluemix. Bắt đầu chương trình, vi điều khiển sẽ truyền tín hiệu chạy file script1.lua sang cho module wifi esp8266, khi module wifi thông báo đã chạy file script1.lua, vi điều khiển truyền tiếp tục hai thông số là tên wifi (SSID) muốn kết nối và password của wifi đó, delay 2 giây và bắt đầu chạy chương trình chính.

Chương trình chính của vi điều khiển sẽ bắt đầu từ việc lấy toàn bộ các giá trị của cảm biến, chuyển nó sang module wifi và chờ tín hiệu gởi thành công lên IBM Bluemix hay không, trong một khoảng thời gian nếu không có tín hiệu trả lời của module wifi, vi điều khiển sẽ quay lại tiếp tục lấy giá trị từ các cảm biến và truyền dữ liệu sang cho esp8266, nếu vi điều khiển nhận được tín hiệu đã gửi thành công lên IBM Bluemix, lúc này chuyển sang vòng lặp chờ hai, để nhận gói điều khiển các thiết bị từ dịch vụ web, trong khoảng 20 giây, nếu không nhận được bất kỳ tín hiệu gì, vi điều khiển lập tức quay lại vòng lặp lấy giá trị từ các cảm biến và gởi cho module wifi, nếu dịch vụ web xử lý lâu, để vi điều khiển chờ, không tiếp tục gửi gói dữ liệu lên tiếp thì dịch vụ web sẽ gửi gói thông tin: “doing” xuống cho vi điều khiển, vi điều khiển sẽ bắt đầu lại vòng lặp và chờ tiếp trong 20 giây. Khi vi điều khiển nhận được dữ liệu về điều khiển thiết bị, nó sẽ ngưng chờ và ghi dữ liệu ra hai chân điều khiển module relay, điều khiển các thiết bị máy bơm nước và bóng đèn, tất cả đều thành công thì vi điều khiển sẽ delay khoảng 30 giây trước khi gởi các giá trị của cảm biến lên lần nữa.

Sau đây là lưu đồ giải thuật của chương trình vi điều khiển:



Hình 3.2: Lưu đồ giải thuật vi điều khiển

## 3.3. Hệ quản trị cơ sở dữ liệu MySQL:

### 3.3.1. Thế nào là một hệ quản trị cơ sở dữ liệu:

Hệ quản trị cơ sở dữ liệu là một phần mềm được dùng để thiết kế và quản trị cơ sở dữ liệu, hỗ trợ cho người dùng trong việc tạo, lấy, thay đổi và quản lý dữ liệu.

### 3.3.2. Đặc điểm của hệ quản trị cơ sở dữ liệu MySQL:

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở, được phát triển và hỗ trợ bởi Oracle.

MySQL sử dụng cơ sở dữ liệu quan hệ, sử dụng ngôn ngữ SQL làm ngôn ngữ truy xuất dữ liệu.

MySQL được viết bằng ngôn ngữ C và C++.

MySQL hỗ trợ nhiều hệ điều hành khác nhau: Linux, Windows, Mac OS,…

MySQL hỗ trợ nhiều ngôn ngữ lập trình : Java, C, C++, PHP, Perl, Python và Tcl.

MySQL hỗ trợ chuẩn giao tiếp ODBC dành cho các ngôn ngữ như Delphi, Visual Basic, ngoài ra tạo riêng chuẩn giao tiếp cho .NET là Connector/NET

MySQL hỗ trợ Trigger, View, Cursors.

MySQL hỗ trợ SubSelect.

MySQL hỗ trợ Full-text search.

MySQL hỗ trợ Replication.

MySQL hỗ trợ Query caching.

MySQL hỗ trợ Stored Procedure với cách viết tuân thủ theo SQL/PSM (tiêu chuẩn ISO về Stored Procedure)

MySQL hỗ trợ quá trình roll back , commit và crash recovery khi sử dụng transactions.

### 3.3.3. Stored Procedure trong MySQL:

Stored Procedure

Cú pháp tạo một Stored Procedure trong MySQL

CREATE PROCEDURE `mysp\_test`(IN \_varInput int, OUT \_varOutput varchar(45), INOUT \_varIO int)

BEGIN

END ;;

DELIMITER ;

Trong đó :

- mysp\_test là tên Stored Procedure.

- Các tham số \_varInput, \_varOutput và \_varIO kiểu dữ liệu lần lượt là int, varchar(45) và int, với tham số được quy định là IN, hay INOUT Stored Procedure có thể lấy giá trị các tham số này sử dụng vào các câu truy vấn SQL, tuy nhiên nếu muốn truyền giá trị ra ngoài thì chỉ có thể select hoặc set giá trị cho các biến OUT hoặc INOUT.

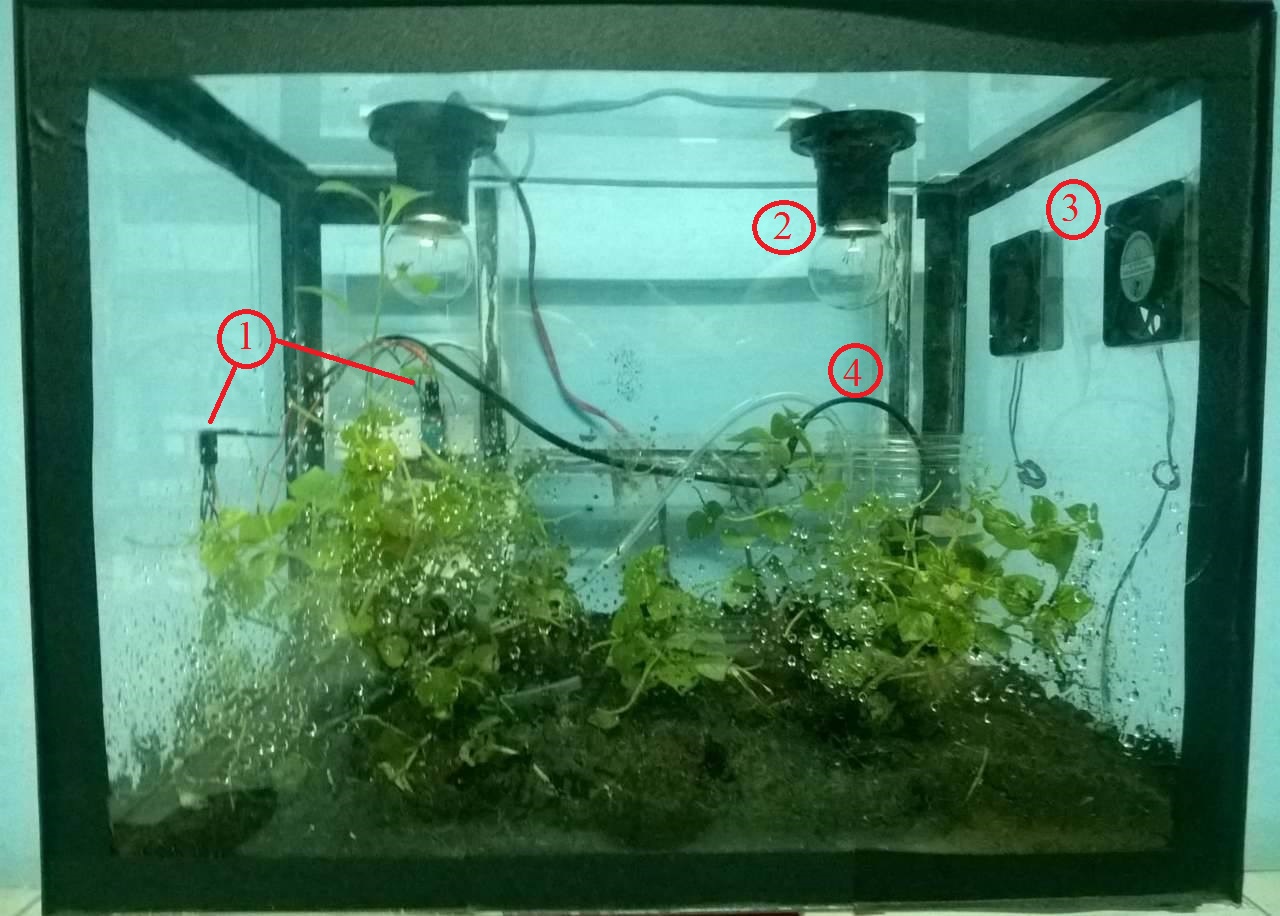
- Các câu lệnh truy vấn sẽ được viết toàn bộ trong khối BEGIN và END.

Để thực thi các Stored Procedure trong MySQL, ta dùng lệnh call mysp\_test(@input, @output, @io), khi thực hiện xong, ta có thể SELECT @output và @io để lấy dữ liệu thực hiện tiếp vì @output là OUT tham số và @io là INOUT tham số.

Hình 3.3: Lưu đồ giải thuật của api subscribe

# CHƯƠNG 4: HỆ THỐNG TRONG THỰC TIỄN

## 4.1. Nhà kính:



Hình 4.1: Mô hình nhà kính

Mô hình nhà kính gồm các bộ phận sau:

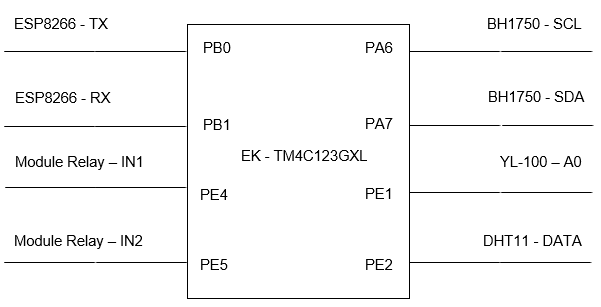
* + (1): bộ cảm biến.
  + (2): bộ hai bóng đèn tròn.
  + (3): bộ hai quạt thông gió ngược chiều nhau.
  + (4): máy bơm nước.

## 4.2. Bộ cảm biến:

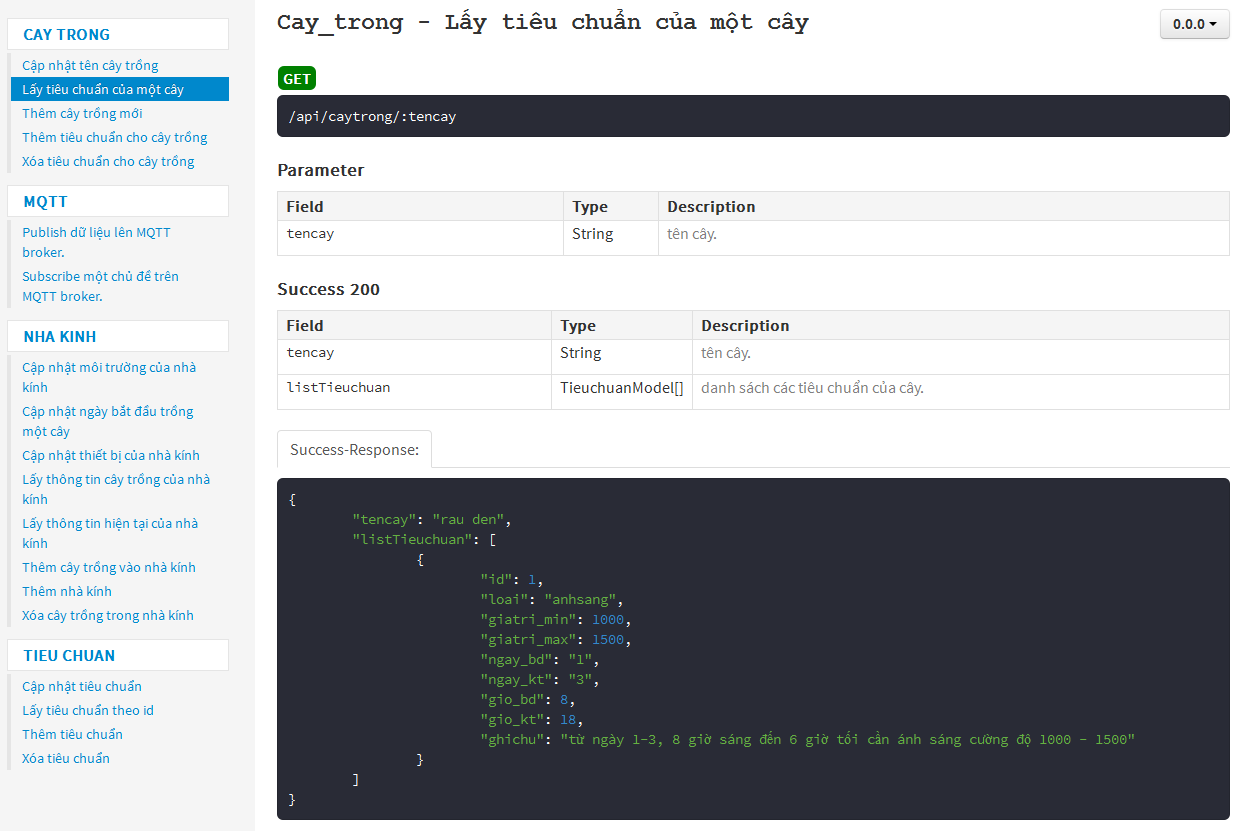
Bộ cảm biến gồm các thành phần:

* + Cảm biến độ ẩm đất (YL-100).
  + Cảm biến nhiệt độ và độ ẩm không khí (DHT11).
  + Cảm biến ánh sáng (BH1750).
  + Module hai relay opto cách ly.
  + Module wifi esp8266.
  + Vi điều khiển TM4C123GH6PM (kit TM4C123GXL)

Bộ cảm biến được mắc nối như hình sau:

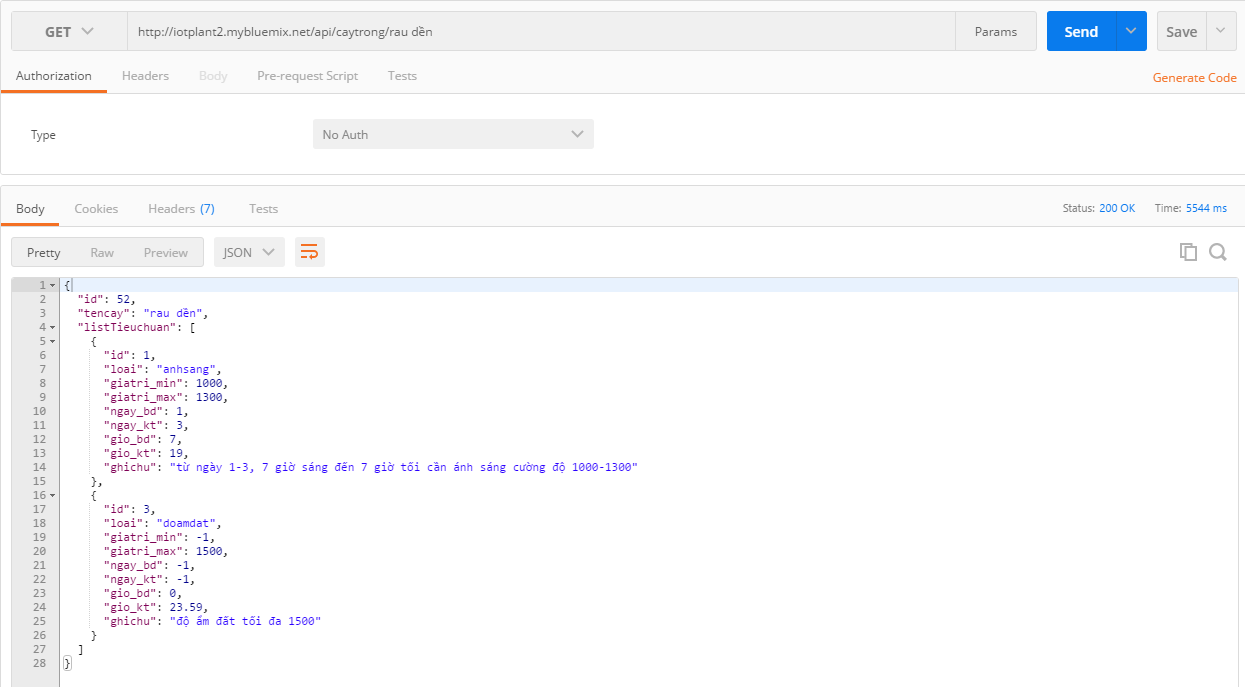
Hình 4.2: Bộ cảm biến

## 4.3. Dịch vụ web:

Hình 4.3: Trang chủ dịch vụ web

Các API được chia thành các nhóm nhỏ: cây trồng, nhà kính và tiêu chuẩn, MQTT. Trong đó chỉ có các API thuộc MQTT xử lý với MQTT broker trên IBM Bluemix, ba nhóm còn lại đều là các api xử lý với cơ sở dữ liệu.

Ta có thễ kiểm tra dịch vụ web bằng ứng dụng Postman, mộ ứng dụng chạy được cài đặt trên trình duyệt web Chrome.

Hình 4.4: Postman

# KẾT LUẬN

Có thể thấy việc trồng cây trong nhà kính mang lại rất nhiều lợi ích như: sự tiện lợi, tự động hóa cao, bảo vệ cây trồng mùa màng khỏi sâu bệnh, tai ương, thiên tai tốt, có thể trồng các loại rau củ quả trái mùa mà năng suất vẫn cao, giảm chi phí lao công. Việc trồng cây trong nhà kính ở nước ta hiện nay đang dần phổ biến và được ưa chuộng nhiều hơn. Vì thế việc đầu tư và phát triển hệ thống thông tin môi trường của cây trồng trong nhà kính được quan tâm và chú trọng khá nhiều, bởi nó ảnh hưởng tới sự tự động hóa, một trong những đặc điểm được đánh giá cao của việc trồng cây trong nhà kính với trồng cây ngoài vuờn.

Với một hệ thống thông tin môi trường của cây trồng trong nhà kính, ta có khá nhiều hướng mở rộng, điển hình như xây dựng một ứng dụng trên mobile theo dõi và ghi nhận các họa động của nhà kính, môi trường hiện tại bên trong, thông tin chi tiết về cây trồng, ngày trồng, cây phát triển như thế nào, các thống kê so sánh với các vụ mùa trước đây, người trồng có thể điều khiển các thiết bị trong nhà kính theo ý muốn, ở mọi nơi, mọi lúc miễn là có mạng, ngoài ra ứng dụng còn có thể giúp cho người mua biết các thông tin về nơi sản xuất như được trồng ngày nào, trồng ra sao, xài các loại phân bón gì, thu hoạch ngày nào,… Không những xây dựng một ứng dụng trên mobile, ta có thể xây dựng thêm trang web giúp người trồng theo dõi tốt hơn khi đang ở nhà hoặc sử dụng mobile nhưng không cài ứng dụng.

# TÀI LIỆU THAM KHẢO

1. IBM Bluemix Docs: <https://console.ng.bluemix.net/docs/>
2. IBM IOT Foundation documentation: <https://docs.internetofthings.ibmcloud.com/>
3. NodeMCU documentation: <https://nodemcu.readthedocs.io/en/master/>
4. IBM developerWorks: http://www.ibm.com/developerworks/
5. Series MQTT Essentials: http://www.hivemq.com/mqtt-essentials/