

ĐẠI HỌC QUỐC GIA THÀNH PHỐ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**  
*Ngành Kỹ Thuật Máy Tính*

**Nghiên cứu và hiện thực hệ thống  
quan trắc khí thải của các phương  
tiện giao thông**

Giảng viên hướng dẫn:

**TS. Phạm Hoàng Anh**

Sinh viên thực hiện:

**51200436 - Nguyễn Mạnh Cường**

**51202655 - Huỳnh Phạm So Ny**

**51204417 - Võ Tân Tùng**

Tháng 12 Năm 2016

## **Lời cam đoan**

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Tháng 12 Năm 2016

## Lời cảm ơn

Em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của trường Đại học Bách Khoa thành phố Hồ Chí Minh, đặc biệt là các thầy cô khoa Khoa Học và Kỹ Thuật Máy Tính của trường đã tạo điều kiện cho em thực tập ở khoa để có nhiều thời gian cho luận văn tốt nghiệp. Đặc biệt, xin gửi lời cảm ơn chân thành Tiến sĩ Phạm Hoàng Anh đã nhiệt tình hướng dẫn hướng dẫn hoàn thành tốt đợt thực tập. Cám ơn các bạn Nguyễn Thị Trang 91103730, Đỗ Thành Nhân G1302688 đã có những đóng góp quý báu cùng với nhóm.

Trong quá trình thực tập, cũng như là trong quá trình làm bài báo cáo, khó tránh khỏi sai sót, rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp Thầy, Cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài luận văn sắp tới. Em xin chân thành cảm ơn!

Nhóm thực hiện đề tài.

## **Tóm tắt**

Do nhu cầu áp dụng khoa học và kỹ thuật vào phát triển hệ thống giám sát môi trường và giao thông, sinh viên nhóm chúng tôi đã tìm hiểu và thực hiện dự án hệ thống thu thập tình trạng ô nhiễm do khí thải của các phương tiện giao thông. Nhờ sự phát triển mạnh mẽ của Internet of Things, đề tài được định hướng phát triển theo mô hình IoT kết hợp với các kiến thức về môi trường nhằm để xây dựng hệ thống thiết bị đo đặc và lấy dữ liệu để phân tích. Nhóm bắt đầu tìm hiểu về các loại khí thải của phương tiện giao thông, tìm hiểu các cảm biến, các loại vi xử lý hỗ trợ tốt cho hệ thống IoT, kết hợp với hệ thống Cloud Database và ứng dụng Web theo dõi tình trạng ô nhiễm thực tế trên một số đoạn đường đã được đo đặc.

# Mục lục

Danh sách hình vẽ	viii
Danh sách bảng	xii
<b>1 GIỚI THIỆU</b>	<b>1</b>
1.1 Ý tưởng và tính cấp thiết của đề tài . . . . .	1
1.2 Mục tiêu và nhiệm vụ của đề tài . . . . .	2
1.2.1 Mục tiêu đề tài . . . . .	2
1.2.2 Nhiệm vụ đề tài . . . . .	3
1.3 Cấu trúc báo cáo . . . . .	3
<b>2 CƠ SỞ LÝ THUYẾT</b>	<b>4</b>
2.1 Giới thiệu về Internet of Things và các ứng dụng giám sát . . . . .	4
2.1.1 Internet of Things (IoT) . . . . .	4
2.1.2 Các hệ thống giám sát được phát triển dựa trên IoT . . . . .	12
2.2 Giới thiệu về các công cụ hỗ trợ phát triển ứng dụng IoT . . . . .	17
2.3 Những yếu tố ảnh hưởng môi trường từ khí thải phương tiện giao thông	18
2.4 Các hệ thống quan trắc hiện hữu . . . . .	20
2.4.1 VoV giao thông . . . . .	20
2.4.2 BKTraffic . . . . .	21
2.4.3 Hệ thống quan trắc môi trường ở Hà Nội . . . . .	22
2.5 Các thiết bị phần cứng . . . . .	23
2.5.1 Mạch vi xử lý Arduino . . . . .	23
2.5.2 Module SIM800L . . . . .	27
2.5.3 Pin năng lượng mặt trời . . . . .	34
2.6 Kiến thức căn bản về ứng dụng web - di động và xây dựng Server . . .	36
2.6.1 Giao thức TCP . . . . .	36

2.6.2	Web API . . . . .	38
2.6.3	Căn bản về RESTful Web services . . . . .	39
2.6.4	NodeJs . . . . .	43
2.6.5	Javascripts . . . . .	44
2.6.6	Websocket - Socket.io . . . . .	45
2.6.7	Socket.IO . . . . .	46
2.6.8	RethinkDB . . . . .	47
2.6.9	Raspberry . . . . .	49
2.6.10	Framework Ionic . . . . .	50
2.6.11	Framework AngularJS . . . . .	52
2.6.12	Framework Cordova . . . . .	54
<b>3</b>	<b>THIẾT KẾ VÀ HIỆN THỰC</b>	<b>55</b>
3.1	Thiết kế hệ thống . . . . .	55
3.1.1	Kiến trúc mô hình hệ thống . . . . .	55
3.1.2	Các node cảm biến . . . . .	57
3.1.3	Chuẩn giao tiếp giữa các node tới máy chủ . . . . .	68
3.1.4	Thiết kế API . . . . .	69
3.1.5	Cách thức tương tác cập nhật dữ liệu . . . . .	70
3.1.6	Mô hình ứng dụng trình bày dữ liệu . . . . .	72
3.1.7	Các ràng buộc của hệ thống . . . . .	75
3.2	Hiện thực node cảm biến . . . . .	78
3.2.1	Hiện thực prototype node cảm biến . . . . .	78
3.2.2	Hiện thực mô hình tổng thể node cảm biến . . . . .	83
3.2.3	Mức độ tiêu hao năng lượng . . . . .	90
3.3	Hệ thống Server lưu trữ dữ liệu và cung cấp API . . . . .	92
3.3.1	Cấu trúc tổ chức tập tin . . . . .	92
3.3.2	Xây dựng Database . . . . .	95
3.3.3	Hiện thực API . . . . .	97
3.3.4	Xây dựng Web Server . . . . .	104
3.3.5	Hiện thực giao diện . . . . .	104
3.4	Ứng dụng thiết bị di động . . . . .	107
3.4.1	Hiện thực giao diện và chức năng . . . . .	107

<b>4 Kết quả thực nghiệm và Đánh giá</b>	<b>110</b>
4.1 Tính ổn định của node cảm biến . . . . .	110
4.2 Tính ổn định của Server . . . . .	110
4.3 Đánh giá lập trình ứng dụng Mobile sử dụng APIs . . . . .	111
<b>5 Tổng kết và Hướng phát triển cho tương lai</b>	<b>112</b>
5.1 Tổng kết . . . . .	112
5.1.1 Những gì đạt được . . . . .	112
5.1.2 Những khó khăn và hạn chế . . . . .	113
5.2 Hướng phát triển tương lai . . . . .	113
<b>Tài liệu tham khảo</b>	<b>115</b>

# Danh sách hình vẽ

1.1	Mô hình IoT . . . . .	2
2.1	Hình ảnh mô tả Internet of Things . . . . .	5
2.2	Mô hình IoT không có Gateway . . . . .	8
2.3	Mô hình IoT có Gateway . . . . .	9
2.4	Kiến trúc mô hình IoT tham khảo . . . . .	10
2.5	Ứng dụng IoT trong xây dựng . . . . .	13
2.6	Ứng dụng IoT trong năng lượng . . . . .	14
2.7	Ứng dụng IoT trong dân dụng . . . . .	15
2.8	Ứng dụng IoT trong y tế . . . . .	16
2.9	Ứng dụng phát triển IoT Thingspeak . . . . .	17
2.10	Trạm giám sát VOV Giao thông . . . . .	21
2.11	Ứng dụng Web SmartBKTraffic . . . . .	22
2.12	Trạm quan trắc môi trường tại Hà Nội . . . . .	23
2.13	Mạch vi xử lý Arduino Nano . . . . .	24
2.14	Môi trường phát triển của Arduino . . . . .	26
2.15	Sơ đồ module sim800l . . . . .	28
2.16	Tấm pin năng lượng mặt trời . . . . .	34
2.17	Tấm pin năng lượng mặt trời . . . . .	35
2.18	Cấu trúc của một đối tượng JSON . . . . .	40
2.19	Cấu trúc mảng của JSON . . . . .	41
2.20	Cấu trúc của một đối tượng JSON . . . . .	41
2.21	Cấu trúc mô tả XML-JSON . . . . .	42
2.22	Nền tảng Nodejs . . . . .	43
2.23	Mô hình hoạt động Nodejs . . . . .	44
2.24	Mô hình hoạt động Websocket . . . . .	45
2.25	Mô hình cơ bản của Socket.IO . . . . .	46

2.26 Mô hình hoạt động giao tiếp của Socket.IO . . . . .	47
2.27 Biểu tượng của RethinkDB . . . . .	48
2.28 Đoạn code mẫu RethinkDB . . . . .	48
2.29 Thiết bị Raspberry Pi . . . . .	50
2.30 Một số giao diện của Ionic . . . . .	51
2.31 Angular JS kết hợp với ionic . . . . .	53
2.32 Apache Cordova . . . . .	54
 3.1 Kiến trúc mô hình hệ thống . . . . .	56
3.2 Mô hình căn nhà 3D . . . . .	57
3.3 Cảm biến bụi GP2 . . . . .	58
3.4 Sơ đồ GP2 kết nối với Arduino . . . . .	59
3.5 Cảm biến MQ135 . . . . .	60
3.6 Giản đồ chỉ sự biến đổi của Rs/Ro và giá trị ppm của MQ135 . . . . .	61
3.7 Giản đồ chỉ sự biến đổi của Rs/Ro đối với nhiệt độ, độ ẩm . . . . .	61
3.8 Cảm biến MQ07 . . . . .	63
3.9 Giản đồ chỉ sự biến đổi của Rs/Ro và giá trị ppm của MQ07 . . . . .	64
3.10 Giản đồ chỉ sự biến đổi của Rs/Ro đối với nhiệt độ, độ ẩm . . . . .	64
3.11 Cảm biến nhiệt độ DS18B20 . . . . .	66
3.12 Sơ đồ kết nối DS18B20 . . . . .	67
3.13 Sơ đồ trạng thái GPRS cho đơn kết nối . . . . .	68
3.14 Mô hình tương tác polling . . . . .	70
3.15 Mô hình tương tác realtime dựa trên Socket . . . . .	71
3.16 Mô hình tương tác cập nhật nhiều Server . . . . .	71
3.17 Biểu đồ High level Usecase . . . . .	73
3.18 Giản đồ High level Usecase của ứng dụng di động . . . . .	74
3.19 Toàn vẹn dữ liệu . . . . .	76
3.20 Tính bảo mật . . . . .	77
3.21 Mô hình prototype đầu tiên . . . . .	78
3.22 Mô hình khối nguồn . . . . .	79
3.23 Mạch tăng áp mini HT106 . . . . .	80
3.24 Mạch sạc pin Lithium TP4056v2 . . . . .	80
3.25 Pin năng lượng mặt trời Poly 5V /220 mA . . . . .	81
3.26 Kết quả prototype đầu tiên . . . . .	82
3.27 Kết quả prototype đầu tiên 2 . . . . .	83

3.28	Bố trí board mạch vi xử lý . . . . .	84
3.29	Lưu đồ xử lý hàm main . . . . .	86
3.30	Bố trí các cảm biến . . . . .	88
3.31	Mô hình đặt cảm biến GP2 . . . . .	88
3.32	Mô hình bố trí các linh kiện khác . . . . .	89
3.33	Mô hình cấu trúc dữ liệu của Sensor Node . . . . .	96
3.34	Mô hình cấu trúc dữ liệu của User . . . . .	97
3.35	Hiển thị các file log . . . . .	103
3.36	Giao diện chính của Web Server . . . . .	104
3.37	Giao diện đồ thị dữ liệu . . . . .	105
3.38	Giao diện nhận feedback người dùng . . . . .	106
3.39	Giao diện quản lý các node cảm biến . . . . .	107
3.40	Giao diện chính ứng dụng di động . . . . .	108
3.41	Giao diện xem dữ liệu trên ứng dụng di động . . . . .	109

# Danh sách bảng

2.1	My caption . . . . .	12
2.2	Thông số của Vi xử lý Arduino . . . . .	24
2.3	Tập lệnh AT+CBC: kết quả dung lượng pin hiện tại . . . . .	30
2.4	Tập lệnh AT+CCLK: trả về giá trị thời gian của module sim . . . . .	30
2.5	Tập lệnh AT+CSTT: cài đặt các thông số APN, USER NAME, PASS-WORD cho Sim . . . . .	31
2.6	Tập lệnh AT+CIFSR: lấy thông tin địa chỉ IP của moduleSIM . . . . .	31
2.7	Tập lệnh AT+CIPSTART: thiết lập kết nối TCP hoặc UDP . . . . .	32
2.8	Tập lệnh AT+CDNSGIP: phân giải DNS tên miền . . . . .	32
2.9	Tập lệnh AT+CIPSEND: gửi dữ liệu qua giao thức TCP/UCP . . . . .	33
3.1	Bảng API tương tác với dữ liệu về các node . . . . .	69
3.2	Bảng API tương tác với dữ liệu về các node . . . . .	70
3.3	Bảng mô tả giản đồ Usecase của Web Server . . . . .	73
3.4	Bảng mô tả giản đồ Usecase của ứng dụng di động . . . . .	75
3.5	Bảng tiêu thụ năng lượng buổi sáng . . . . .	91
3.6	Bảng tiêu thụ năng lượng buổi tối . . . . .	92

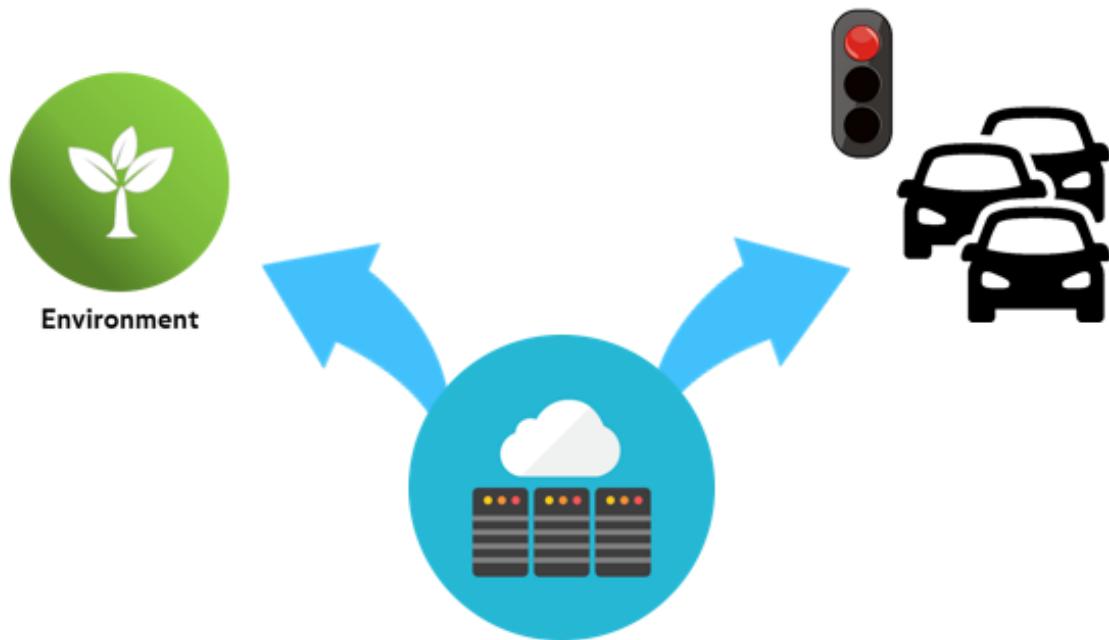
# Chương 1

## GIỚI THIỆU

### 1.1 Ý tưởng và tính cấp thiết của đề tài

Ngày nay, bài toán giao thông vẫn đang là bài toán hóc búa vẫn chưa được giải quyết được ở Việt Nam và nhiều nước đang phát triển. Tình trạng kẹt xe, ùn tắc kéo dài gây ra sự chậm trễ trong công việc, hơn nữa còn gây gia tăng ô nhiễm môi trường, giảm chất lượng môi trường sống. Một trong những khó khăn làm bài toán giao thông khó giải quyết đó chính là không có đầy đủ dữ liệu cần thiết. Chúng ta không thể giải bài toán nếu không có đủ dữ kiện, cũng như giải quyết kẹt xe ta cần phải có dữ liệu về lưu lượng giao thông.

Vì yếu tố trên, dự án Smart Traffic được hình thành và phát triển theo hướng IoT với mục đích thu thập dữ liệu về các yếu tố môi trường (nồng độ CO, nhiệt độ, độ ẩm, nồng độ bụi, độ ồn...) và xác định mối tương quan giữa lưu lượng giao thông với môi trường xung quanh khu vực đó. Từ đó ta có thể có được 1 phần dữ liệu cần thiết về tình trạng các con đường theo thời gian thực, cũng như có được lịch sử và dùng dữ liệu ấy để phát triển dự đoán tình trạng giao thông tiếp theo.



Hình 1.1 Mo ta tam hinh Mo hinh IoT

Những dữ liệu đó có thể được phân tích và xử lý, đưa ra những kết luận về tình trạng lưu thông trên đoạn đường đó, về sự thay đổi môi trường tại 1 khu vực ở những khoảng thời gian khác nhau. Và những dữ liệu này có thể ứng dụng cho bất động sản, môi trường và nghiên cứu. Do đó đề tài mang tính thiết thực và ứng dụng cao và được lựa chọn để làm Thực tập tốt nghiệp và có thể lên Luận văn tốt nghiệp.

## 1.2 Mục tiêu và nhiệm vụ của đề tài

### 1.2.1 Mục tiêu đề tài

- Tìm hiểu về khí thải xe máy và ô tô.
- Hiện thực mạch cảm biến để thu thập khí thải.

- Khảo sát và đo khí thải thực tế tại một số điểm thường xuyên xảy ra kẹt xe trên địa bàn thành phố.
- Tổng hợp các số liệu thu thập và các yếu tố môi trường khác (nhiệt độ, độ ẩm, ...) để đề xuất mô hình để hướng đến phát triển hệ thống cảnh báo kẹt xe dựa vào tình trạng ô nhiễm không khí.

### **1.2.2 Nhiệm vụ đề tài**

Phân chia công việc tại đây!!!! Sony, Tùng, Cường làm những gì, vẽ giản đồ gantt.

## **1.3 Cấu trúc báo cáo**

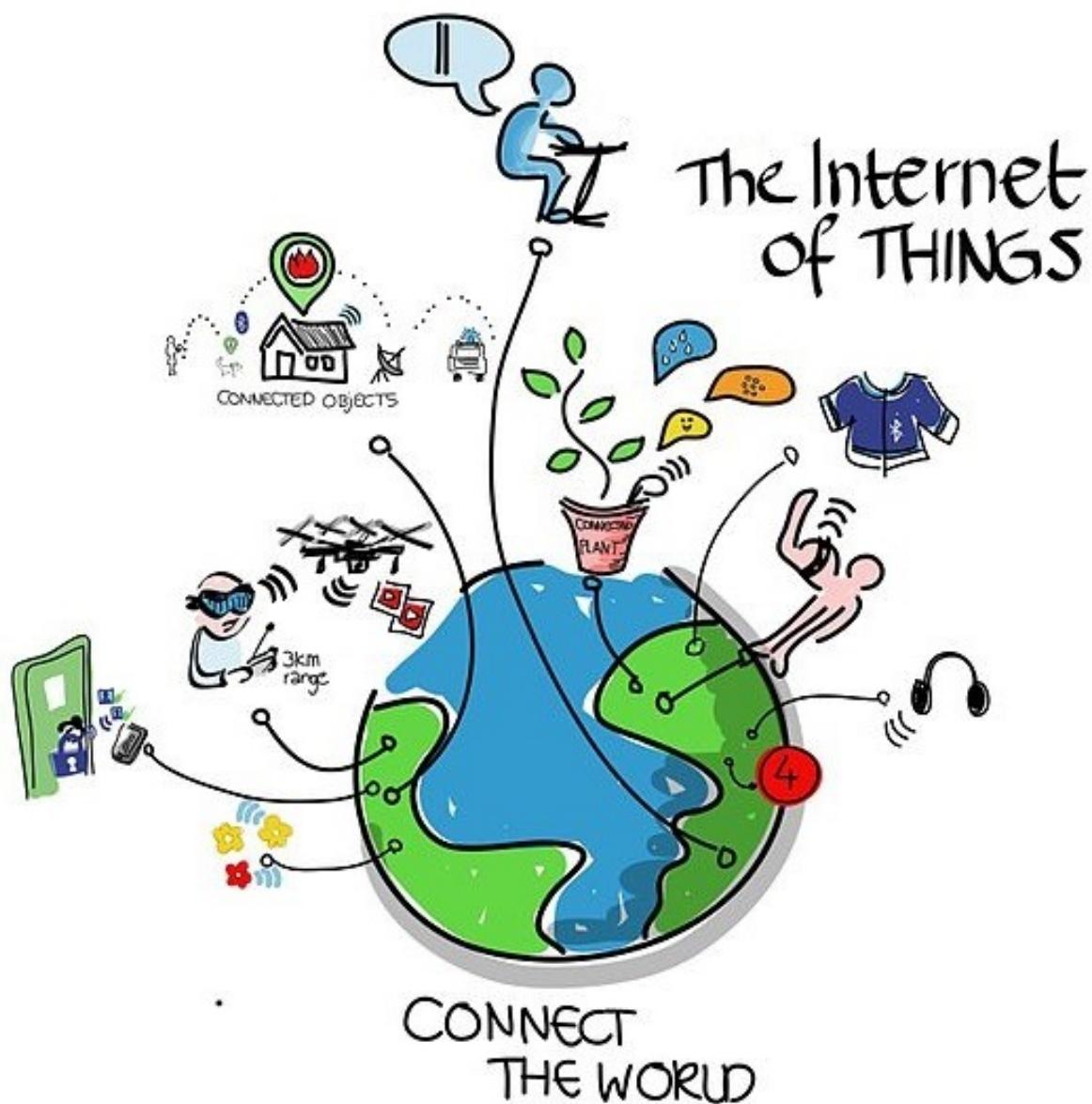
## Chương 2

### CƠ SỞ LÝ THUYẾT

#### 2.1 Giới thiệu về Internet of Things và các ứng dụng giám sát

##### 2.1.1 Internet of Things (IoT)

Với sự phát triển nhanh chóng của công nghệ máy tính hiện nay, những công nghệ cao đã đang và ngày càng được ứng dụng trong nhiều lĩnh vực của cuộc sống con người. IoT là mạng lưới của các đối tượng vật lý, các thiết bị, các phương tiện và các đối tượng này được nhúng với các thiết bị điện tử, cảm biến, phần mềm điều khiển, và nó có khả năng trao đổi dữ liệu và thao tác với nhau. IoT cho phép các đối tượng được lắng nghe và điều khiển từ xa trên cơ sở hạ tầng mạng hiện có, các hệ thống máy tính có thể thu thập dữ liệu và điều khiển các thiết bị đối tượng một cách hiệu quả, chính xác và lợi ích kinh tế nhất có thể.



Hình 2.1 Hình ảnh mô tả Internet of Things

### **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. Nếu mọi đối 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 lý được nó thông qua máy tính. Việc đánh dấu 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.

## Xu hướng và tính chất của The Internet of Things

**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.

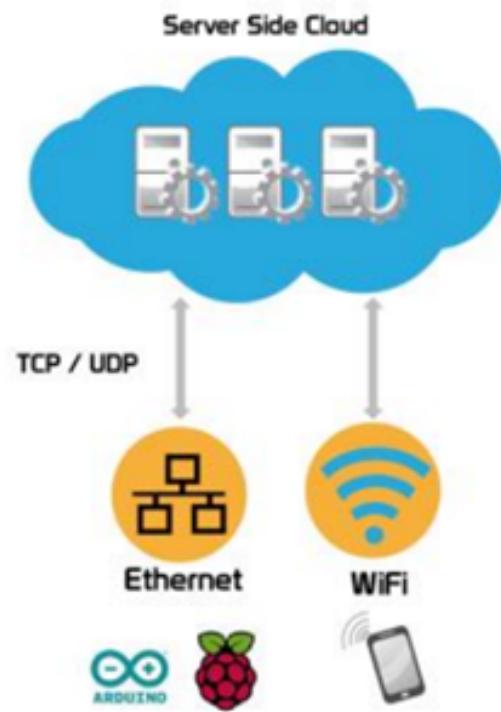
**Là một hệ thống phức tạp:** Trong một thế giới mở, IoT sẽ mang tính chất phức tạp bởi nó bao gồm một lượng lớn các đường liên kết giữa những thiết bị, máy móc, dịch vụ với nhau, ngoài ra còn bởi khả năng thêm vào các nhân tố mới.

**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 gian, 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 lý 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 lý 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 lý dữ liệu đó được xem như không hiệu quả. Ngoài ra, việc xử lý một khối lượng lớn dữ liệu trong thời gian ngắn đủ để đáp ứng cho hoạt động của các đối tượng cũng là một thách thức hiện nay.

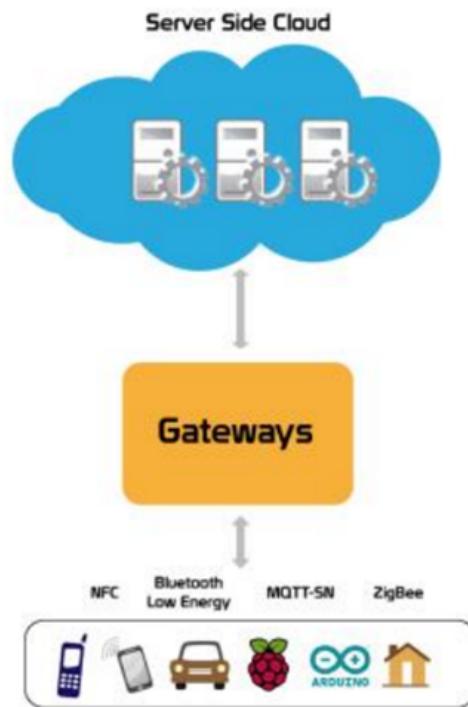
## Mô hình theo xu hướng IoT thực tế

Mô hình (Hình 2.2): Các thiết bị sẽ có thể kết nối với nhau bằng TCP/UDP và có thể kết nối trực tiếp đến máy chủ mà không cần thông qua thiết bị khác.



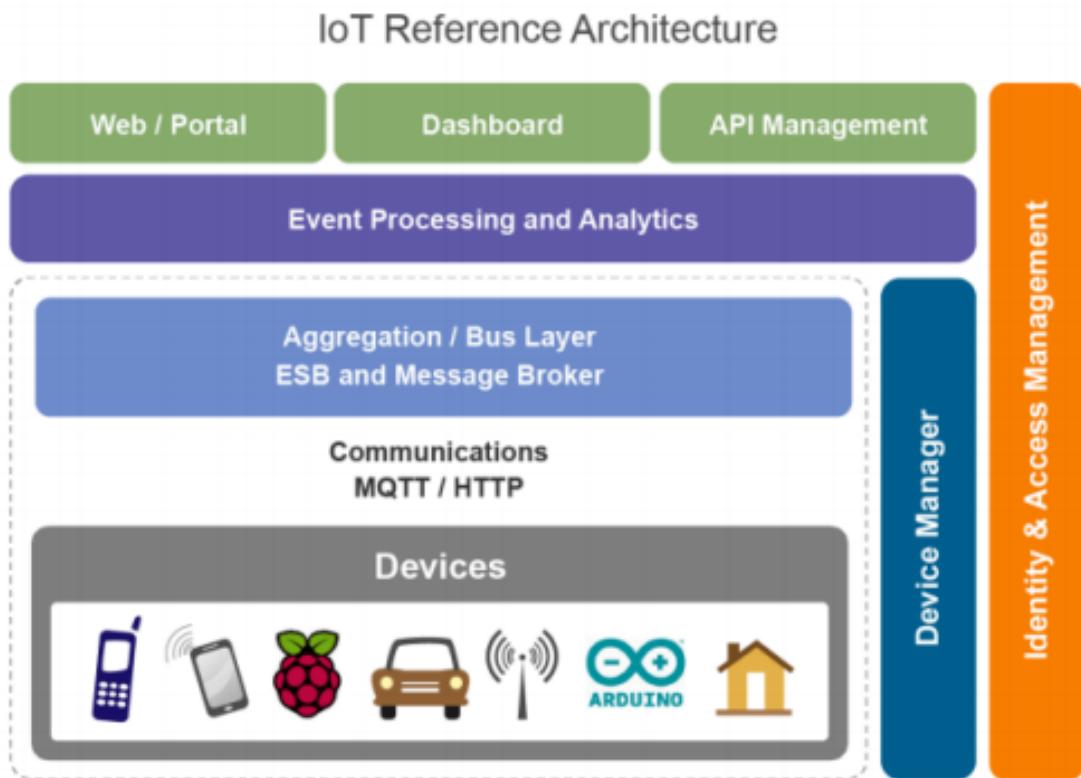
Hình 2.2 Mô hình IoT không có Gateway

Mô hình (Hình 2.3): Các thiết bị sẽ được hỗ trợ nhiều chuẩn giao tiếp khác nhau (BLE, ZWave,...) nên cần cầu nối là Gateway để kết nối đến máy chủ.



Hình 2.3 Mô hình IoT có Gateway

Sơ đồ lớp Chung của cả hai mô hình trên:



Hình 2.4 Kiến trúc mô hình IoT tham khảo

- Lớp Device

Lớp dưới cùng của kiến trúc là lớp thiết bị. Thiết bị có thể được các loại khác nhau, nhưng để có thể được xem là thiết bị IOT, nó phải có một số thông tin liên lạc hoặc gián tiếp hoặc trực tiếp với Internet.

Mỗi thiết bị thường cần một ID. ID có thể là: Bluetooth identifier, Wi-Fi MAC Address...

- Lớp Communications

Các lớp truyền thông hỗ trợ các kết nối của các thiết bị tới máy chủ. Có nhiều giao thức để giao tiếp giữa các thiết bị và máy chủ. Nổi bật nhất là HTTP hoặc MQTT.

HTTP rất thông dụng. Bởi vì nó là một giao thức dựa trên văn bản đơn giản,

nhiều thiết bị nhỏ như bộ điều khiển 8-bit có thể hỗ trợ một phần các giao thức - ví dụ đủ dữ liệu để POST hoặc GET một nguồn tài nguyên. Các thiết bị lớn hơn 32-bit có thể sử dụng đầy đủ các thư viện HTTP client đúng cách để hiện thực toàn bộ giao thức.

MQTT được phát minh vào năm 1999 để giải quyết các vấn đề trong hệ thống nhúng và SCADA. Nó đã được thông qua một số lần lặp lại và phiên bản hiện tại (3.1.1) đang trải qua những tiêu chuẩn hóa trong OASIS MQTT kỹ thuật Committee8. MQTT là một hệ thống tin nhắn publish-subscribe dựa trên một mô hình broker. Các giao thức có một chi phí rất nhỏ (ít nhất là 2 byte cho mỗi tin nhắn). MQTT được thiết kế để vận hành qua TCP.

- Lớp Aggregation/Bus

Một lớp quan trọng của kiến trúc là lớp mà tập hợp và là cầu nối, có khả năng tổng hợp và kết hợp các thông tin liên lạc từ các thiết bị khác nhau và đưa thông tin liên lạc đến một thiết bị cụ thể (có thể thông qua một gateway).

- Lớp Event Processing and Analytic:

Lớp này có các sự kiện từ lớp Bus và cung cấp khả năng xử lý và hành động theo những sự kiện này. Yêu cầu phải lưu trữ các dữ liệu vào một cơ sở dữ liệu nên buộc phải có một ứng dụng ở máy chủ.

- Lớp External Communications (Top)

Lớp này cung cấp một cách cho các thiết bị để chúng giao tiếp ra bên ngoài hệ thống một cách có định hướng và thân thiện với người dùng (Web, Dashboard, API Management)

Bảng 2.1 My caption

MÔ HÌNH	ƯU ĐIỂM	NHƯỢC ĐIỂM
Mô hình cũ	Đơn giản hiện thực	Gateway cồng kềnh vì phải có bộ phận thu/phát hồng ngoại và các mạch để giao tiếp RF.
Mô hình 1	Không cần thông qua gateway, mọi thông tin thiết bị đều được lưu trên máy chủ.	Hạn chế về phương thức giao tiếp vì chỉ sử dụng TCP/UDP để giao tiếp. Không thể điều khiển thiết bị khi không truy cập được máy chủ.
Mô hình 2	Quản lý thiết bị thông qua gateway làm trung gian, nên các thiết bị có thể tương tác với nhau dễ dàng mà không cần đến máy chủ.	Chi phí cao hơn, hệ thống phức tạp hơn. Độ bảo mật cũng cần được quan tâm hơn.

### 2.1.2 Các hệ thống giám sát được phát triển dựa trên IoT

Khi kết nối một chuỗi khổng lồ các cảm biến (sensor) thu thập dữ liệu, thiết bị và máy móc với nhau, điều quan trọng cần nhận ra là thông tin sẽ được chuyển đổi thành hành động với một tốc độ mà chúng ta chưa từng thấy trước kia. Chúng ta đang tiến đến gần, nếu không phải là đã chạm được vào một thế giới của những khoảng thời gian phản ứng cực nhỏ, phản hồi tức thì với mọi điều kiện biến đổi, và mức độ điều khiển chưa từng có trong việc quản lý tài nguyên và tài sản. Điểm mấu chốt ở đây là đừng nghĩ hẹp. Internet of Things (IoT) không đơn thuần là mang đến sự tiết kiệm trong các mô hình công nghiệp hiện tại. Nó đảo lộn hoàn toàn những mô hình cũ, tạo ra những sản phẩm và dịch vụ mới. Không có một lĩnh vực nào mà trong đó IoT tạo ra ảnh hưởng đặc biệt lớn nhất; bởi IoT sẽ thay đổi hoàn toàn mọi lĩnh vực một cách không thể tưởng tượng được, bao gồm nông nghiệp, năng lượng, an ninh, quản lý thảm họa, y tế, và đó chỉ là một vài lĩnh vực được nhắc đến.

### Ứng dụng trong xây dựng



Hình 2.5 Ứng dụng IoT trong xây dựng

Ví dụ: Các công ty xây dựng đã bắt đầu trang bị các silo (hầm chứa đồ) và xe tải có các cảm biến theo dõi mức hàng tồn kho, như là lượng bê tông, và biến đổi nó thông qua platform trên nền điện toán đám mây để gia tăng tốc độ phân phối và đảm bảo một dòng lưu thông vật liệu ổn định. Các ông lớn trong ngành công nghiệp dầu mỏ đã bắt đầu thực thi các công nghệ mobile, cảm biến tới máy móc để dự phòng từ trước cho các tai nạn thông qua các phân tích nhanh chóng và hành động tức thời. Khi các cảm biến phát hiện ra một sự cố như rò rỉ hoặc thất thoát đường ống, công nghệ IoT cho phép các công nhân lập tức xác định vị trí của chúng.

## Ứng dụng trong năng lượng



Hình 2.6 Ứng dụng IoT trong năng lượng

Một ví dụ khác của công nghệ IoT mới ứng dụng trong công nghiệp dầu mỏ, đó là giếng thông minh. Đây là một dạng giếng cài đặt các thiết bị điều khiển dòng chảy và cảm biến lỗ khoan, để có thể giám sát và điều khiển từ trên bề mặt mà không đe dọa an toàn của công nhân. Giếng thông minh có trang bị công nghệ địa chấn 4D, cho phép theo dõi sự rò rỉ khí ga, dòng chảy nước, thay đổi áp lực, và bất cứ thay đổi nào khác gây ra bởi những biến động địa chấn, giúp cho việc dự đoán và điều khiển các tác động địa chấn có thể gây ra những hỏng hóc nghiêm trọng. Nhưng như thế, chúng ta vẫn nghĩ quá hẹp. Hãy vượt ra khỏi lĩnh vực xây dựng hay năng lượng. Chúng ta có các cảm biến có thể đo lực, tải, moment, và áp lực; các cảm biến có thể ngửi thấy mùi khí ga hay hóa chất; những cảm biến có thể nghe thấy rung động và phân biệt giữa các âm hưởng khác nhau; những cảm biến có thể đo nhiệt độ, phát hiện chuyển động, vận tốc và chuyển vị; xác định vị trí, sự có mặt, và khoảng cách. Nói cách khác, chúng ta có khả năng thu thập những hiểu biết gần như không giới hạn, trong thời gian thực.

## Ứng dụng trong dân dụng



Hình 2.7 Ứng dụng IoT trong dân dụng

Làm thế nào chúng ta có thể tận dụng thông tin thời gian thực từ rất nhiều sensor? Hãy nhìn vào ngôi nhà của chúng ta. Những phần nào trong đó có thể thông minh hóa? Ví dụ đơn giản. Tôi từng quan sát 1 hệ thống video conference cho phép người chủ nói chuyện với chú cún của mình, gọi nó đến, cho nó ăn từ xa thông qua một thiết bị thông minh. Hãy nghĩ lớn hơn nữa. Một ngôi nhà biết khi nào bạn về nhà bởi nó kết nối với một cảm biến trên xe hay smartphone của bạn. Một ngôi nhà kết nối các cảm biến báo khói, hệ thống an ninh, và thiết bị giải trí tới điện thoại của bạn. Một ngôi nhà với các cảm biến được gắn vào đường ống để có thể phát hiện ra rò rỉ trước cả khi điều đó thực sự diễn ra.

## Ứng dụng trong y tế



Hình 2.8 Ứng dụng IoT trong y tế

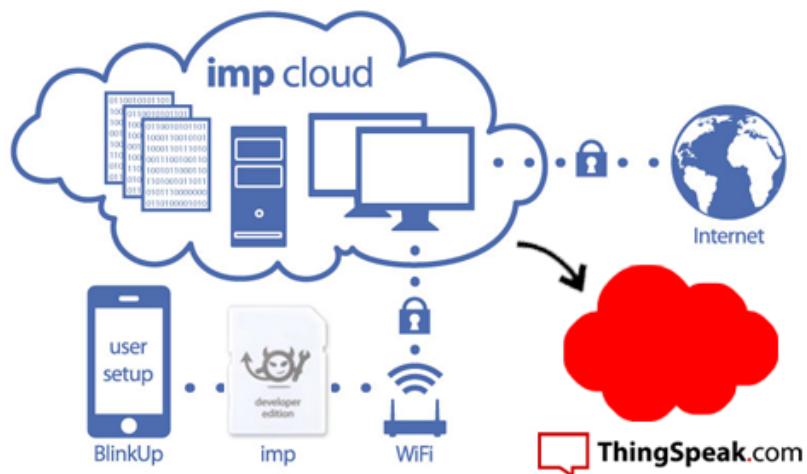
Công nghệ thiết bị đeo cũng sẽ biến đổi hoàn toàn lĩnh vực chăm sóc sức khỏe theo những cách phi thường. Chúng ta đều biết rằng đồng hồ Apple Watch sẽ tích hợp một cảm biến theo dõi nhịp tim và cung cấp cho chủ của nó những ứng dụng tạo điều kiện và khuyến khích một cách sống lành mạnh. Chúng ta đã có các cảm biến gắn trong giày để theo dõi việc chạy xa đến đâu và bao nhiêu calo đã được đốt. Còn gì tiếp theo? Sẽ có một quy trình tối ưu chăm sóc sức khỏe, theo đó có những cảm biến có thể phát hiện vi khuẩn trong thiết bị, và thiết bị diệt khuẩn phát hiện virus có thể di chuyển từ bệnh nhân.

## 2.2 Giới thiệu về các công cụ hỗ trợ phát triển ứng dụng IoT

### Thingspeak

ThingSpeak là một mã nguồn mở cho các ứng dụng của Internet of Things. Mã nguồn này hỗ trợ các API lưu trữ, lấy dữ liệu từ các thiết bị, sản phẩm sử dụng HTTP qua Internet hoặc thông qua một Local Area Network. Như một HUB đợi các thông tin cảm biến từ thiết bị và có nhiệm vụ lưu trữ và xử lý dữ liệu, với ThingSpeak, bạn có thể tạo ra các ứng dụng phân tích dữ liệu, lưu trữ dữ liệu, quản lý dữ liệu một cách đơn giản.

ThinkSpeak được phát triển bởi ioBridge và được opensource trên GITHUB <https://github.com/iobridge/thingspeak>



Hình 2.9 Ứng dụng phát triển IoT Thingspeak

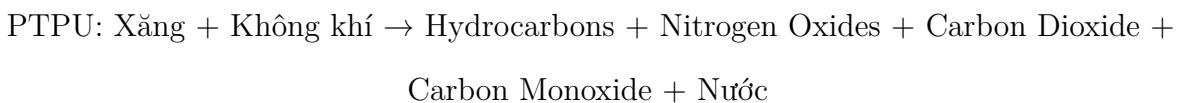
## 2.3 Những yếu tố ảnh hưởng môi trường từ khí thải phương tiện giao thông

Để hình dung được những yếu tố ảnh hưởng môi trường từ khí thải xe máy và ô tô thì chúng ta cần biết được quá trình hoạt động của xe máy và ô tô, từ đó chúng ta sẽ biết được những loại khí thải nào mà xe máy và ô tô sẽ thải ra môi trường. Qua quá trình tìm hiểu và đọc thông tin tài liệu trên mạng khí thải của xe máy và ô tô tuỳ thuộc chủ yếu vào chất lượng đốt cháy hỗn hợp xăng và không khí bên trong buồng đốt (combustion chamber) của động cơ, cũng như nồng độ các chất ô nhiễm trong khí xả phụ thuộc vào đặc điểm động cơ cũng như các thông số điều chỉnh, vận hành.

Động cơ mới và được điều chỉnh đúng cho phản ứng cháy hoàn chỉnh (complete combustion) hay phản ứng cháy thừa oxy:



Các phó sản (by-product) chủ yếu của phản ứng này là nước ( $\text{H}_2\text{O}$ ) và carbon dioxide ( $\text{CO}_2$ ), do đó ống thoát khí cháy (tail pipe) của một động cơ tốt thường có nước nhão ra, dễ nhận thấy khi động cơ đang trong quá trình làm nóng máy (warm up). Động cơ cũ hoặc không được điều chỉnh đúng cho phản ứng cháy không hoàn chỉnh (incomplete combustion) hay thiếu oxy:



Phản ứng này tạo thêm những phó sản như carbon monoxide (CO) và nitrogen oxides ( $\text{NO}_x$ ). Vậy chúng ta có các loại sản phẩm như sau:

- CO được sinh ra khi lượng oxy đưa vào buồng đốt không đủ.
- HC được sinh ra trong quá trình đốt cháy không hoàn toàn, cũng như CO.

- NOx được sinh ra do nitơ và ôxy trong hỗn hợp không khí-nhiên liệu, khi nhiệt độ của buồng đốt tăng cao trên 1800oC.

Nhiệt độ của buồng đốt càng cao, lượng NOx sản ra càng nhiều. Theo lý thuyết, khi đốt cháy xăng thì chỉ sinh ra CO<sub>2</sub> (cácbon điôxit) và H<sub>2</sub>O (hơi nước). Tuy nhiên, không phải toàn bộ xăng đều tham gia phản ứng như lí thuyết, do ảnh hưởng của các yếu tố như tỷ lệ hỗn hợp không khí-nhiên liệu, nitơ trong không khí, nhiệt độ cháy, thời gian cháy... Đó là nguyên nhân sinh ra các khí độc hại như CO, HC hoặc NOx. Trên cơ sở thực tế, động cơ đốt trong nó chỉ sử dụng được khoảng 30-45% nhiệt lượng để sinh công, phần còn lại bị hao hụt đi mất, do đó nó mang theo nhiệt lượng thải ra ngoài. Lượng nhiệt này thực tế đã tác động đến môi trường, làm cho nhiệt độ xung quanh những khu vực có xe máy và ô tô tăng lên. Việc đi lại trên những đoạn đường đông xe máy và ô tô thường gây cho chúng ta cảm giác nóng bức, và khó thở. Đó chính là những yếu tố khí thải đã trình bày như trên của xe máy và ô tô đã tác động đến môi trường. Một phần nhân tố cũng đáng chú ý trong quá trình hoạt động xe máy và ô tô hoạt động đó là bụi, vì đầu vào của động cơ là xăng và không khí, mà trong xăng hiện nay thường có cặn và lượng không khí đầu vào cũng có bụi mặc dù có bộ lọc khí nhưng không thể tránh được trong quá trình sử dụng lâu dài. Một số loại phương tiện giao thông sử dụng thời gian dài, không đi bảo trì sẽ thải ra một nồng độ ô nhiễm rất lớn. Từ những cơ sơ lý thuyết trên cho chúng ta biết được những yếu tố ảnh hưởng môi trường từ khí thải xe máy và ô tô bao gồm:

- Khí CO<sub>2</sub>, CO, HC, NOx.
- Khói bụi.
- Yếu tố về nhiệt.

## 2.4 Các hệ thống quan trắc hiện hữu

### 2.4.1 VoV giao thông

Sử dụng hệ thống camera an ninh kết hợp với lượng phóng viên thường trực khắp thành phố để giám sát trực tiếp và cảnh báo tình trạng lưu thông trên các đoạn đường. Hệ thống có tính hiệu quả rất cao vì sử dụng CCTV giám sát tình trạng giao thông theo thời gian thực nên có góc nhìn thực tế nhất và không phục thuộc, có thể hoạt động độc lập. Tuy nhiên hệ thống vẫn có nhiều hạn chế như:

- Vẫn chủ yếu hoạt động một cách thủ công, điều tiết và đưa ra kết quả bởi con người, chưa có áp dụng thuật toán xử lý máy tính vào công việc nhiều.
- Tốn nhiều công sức và nhân lực, cần có nhiều phóng viên trực thuộc các đoạn đường cũng như người quản lý theo dõi hệ thống camera giao thông.
- Chi phí duy trì và lắp đặt còn khá cao, và cần tốn chi phí bảo trì đắt đỏ và trình độ chuyên môn nhân viên điều hành cao, chi phí một hệ thống có thể lên tới hàng trăm tỉ. Cụ thể như hệ thống trị giá 270 tỉ VND bao gồm 19 trạm giám sát camera được triển khai ở Bà Rịa - Vũng Tàu.



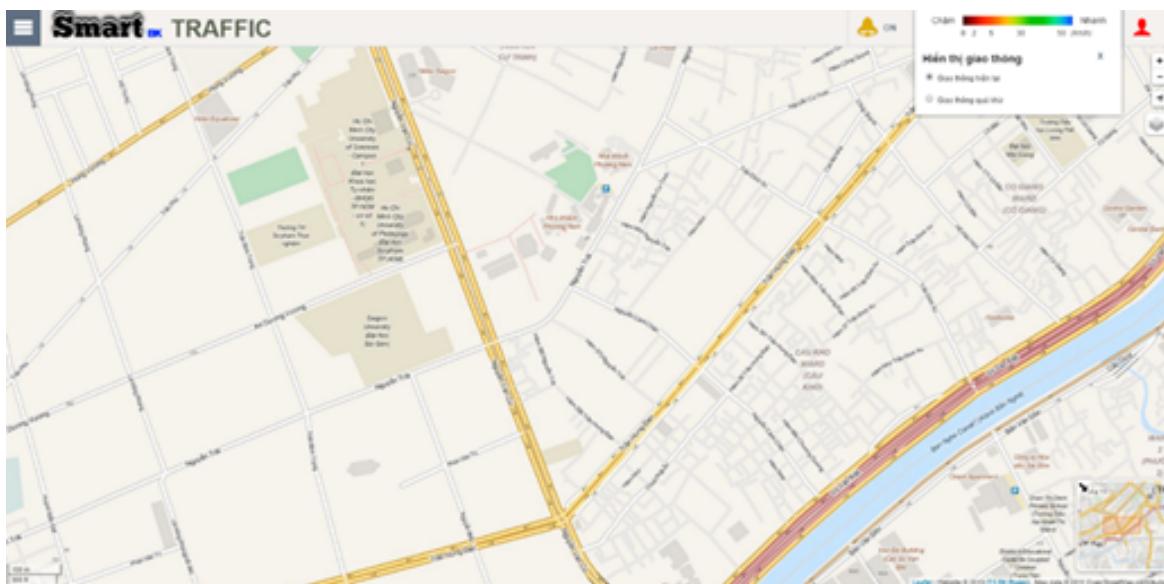
Hình 2.10 Trạm giám sát VOV Giao thông

### 2.4.2 BKTraffic

Bktraffic <http://traffic.hcmut.edu.vn> kết hợp hệ thống xe bus cùng với định vị GPS và gửi dữ liệu qua hạ tầng mạng 3G để theo dõi vị trí xe và tính toán tốc độ lưu thông trên đoạn đường xe đang lưu thông. Dự án này có hiệu quả cao vì hệ thống xe bus dày đặc và có thời gian hoạt động rộng. Tuy nhiên vẫn có những hạn chế như:

- Vì chỉ giám sát trên xe bus nhưng nhiều trường hợp không hiệu quả khi áp dụng cho giao thông tại Việt Nam. Bởi vì đa số phương tiện giao thông ở Việt Nam là xe 2 bánh, do đó hệ thống bus di chuyển không phản ánh được tình trạng lưu thông chung trên tuyến đường đó.

- Nhiều tuyến đường trên địa bàn thành phố mà xe bus vẫn chưa hoạt động, do vậy vẫn chưa có đủ dữ liệu cần thiết để.
- Mang tính phụ thuộc vào xe bus, vậy nên có thể có những trường hợp ảnh hưởng bởi hệ thống xe bus mà hệ thống sẽ không hoạt động hiệu quả được.



Hình 2.11 Ứng dụng Web SmartBKTraffic

### 2.4.3 Hệ thống quan trắc môi trường ở Hà Nội

Thủ đô Hà Nội vừa lắp đặt hệ thống 80 trạm quan trắc. Hệ thống này có tính tương đồng với đề tài đang thực hiện. Tuy nhiên mục đích chính là để theo dõi môi trường sau sự cố bụi Thủy ngân và chưa thấy hướng ứng dụng vào hệ thống giao thông. Hơn nữa, các trạm này chiếm diện tích lắp đặt nên hạn chế khả năng triển khai số lượng lớn trên diện rộng.



Hình 2.12 Trạm quan trắc môi trường tại Hà Nội

## 2.5 Các thiết bị phần cứng

### 2.5.1 Mạch vi xử lý Arduino

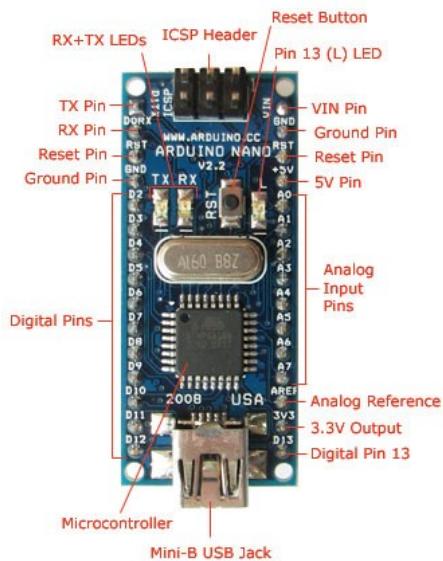
#### Giới thiệu về Arduino Nano

Arduino là một board mạch vi xử lý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM Atmel 32-bit. Những Model hiện tại được trang bị gồm 1 cổng giao tiếp USB, 6 chân đầu vào analog, 14 chân I/O kỹ thuật số tương thích với nhiều board mở rộng khác nhau. Đi cùng với nó là một môi trường phát triển tích hợp (IDE) chạy trên các máy tính cá nhân thông

Bảng 2.2 Thông số của Vi xử lý Arduino

Vi điều khiển	ATmega328 (hỗn 8bit)
Điện áp hoạt động	5VDC
Tần số hoạt động	16 MHz
Dòng tiêu thụ	30mA
Điện áp vào khuyên dùng	7-12 VDC
Điện áp vào giới hạn	6-20 VDC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	8 (độ phân giải 10bit)
Dòng tối đa trên mỗi chân I/O	40 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (ATmega328) với 2KB dùng bởi bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Kích thước	1.85cm x 4.3cm

thường và cho phép người dùng viết các chương trình cho Aduino bằng ngôn ngữ C hoặc C++.

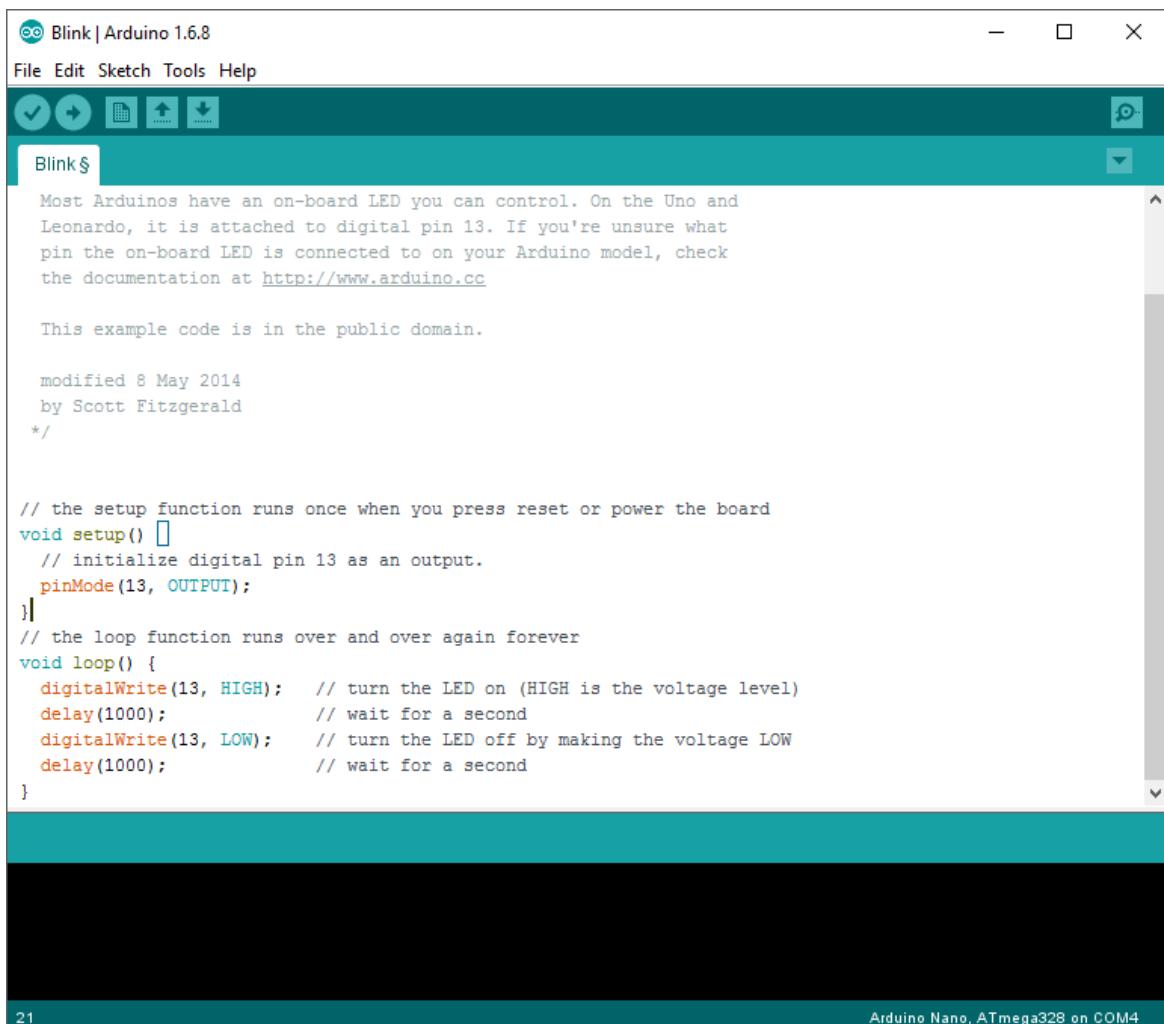


Hình 2.13 Mạch vi xử lý Arduino Nano

## Lập trình cho Arduino Nano

Arduino Nano sử dụng chương trình Arduino IDE để lập trình, và ngôn ngữ lập trình cho Arduino cũng tên là Arduino (được xây dựng trên ngôn ngữ C).

Arduino IDE là môi trường phát triển tích hợp (IDE) của Arduino là một ứng dụng cross-platform (nền tảng) được viết bằng Java, và từ IDE này sẽ được sử dụng cho ngôn ngữ lập trình xử lý (Processing programming language) và project Wiring. Nó được thiết kế để dành cho những người mới tập thành làm quen với lĩnh vực phát triển phần mềm. Nó bao gồm một chương trình code editor với các chức năng như đánh dấu cú pháp, tự động brace matching, và tự động canh lề, cũng như compile và upload chương trình lên board chỉ với 1 cú click chuột. Một chương trình hoặc code viết cho Arduino được gọi là một sketch.



Hình 2.14 Môi trường phát triển của Arduino

Các chương trình Arduino được viết bằng C hoặc C++. Arduino IDE đi kèm với một thư viện phần mềm được gọi là "Wiring", từ project Wiring gốc, có thể giúp các thao tác input/output được dễ dàng hơn. Người dùng chỉ cần định nghĩa 2 hàm để tạo ra một chương trình vòng thực thi (cyclic executive) có thể chạy được:

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

```
1 // the setup function runs once
2 void setup() {
3     // initialize digital pin 13 as an output.
4     pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs forever
8 void loop() {
9     // turn the LED on (HIGH is the voltage level)
10    digitalWrite(13, HIGH);
11    // wait for a second
12    delay(1000);
13    // turn the LED off by making the voltage LOW
14    digitalWrite(13, LOW);
15    // wait for a second
16    delay(1000);
17 }
```

Listing 2.1 Đoạn code nháy đèn cǎn bǎn

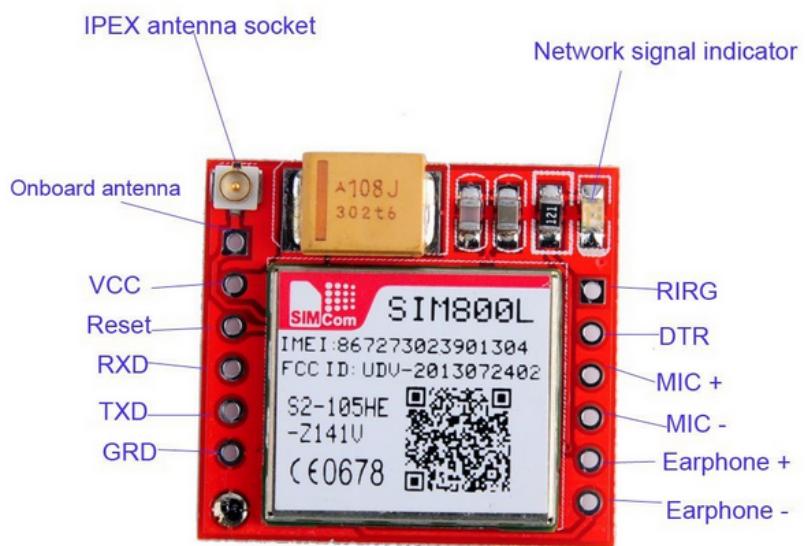
## 2.5.2 Module SIM800L

### Giới thiệu về Sim800L

Thừa kế các chức năng từ các thế hệ module sim trước như sim900, Module GSM sim 800L có khả năng nhắn tin SMS, nghe gọi, GPRS, ... như một điện thoại nhưng có kích thước nhỏ nhất trong các loại module SIM (25 mm x 22 mm). Điều khiển module sử dụng các bộ tập lệnh AT dễ dàng, chân kết nối dùng rào đặc thông dụng chuẩn 100mil. Thông số kỹ thuật:

- Nguồn cấp: 3.7 – 4.2 VDC, có thể sử dụng với nguồn dòng thấp từ 500mAh trở lên, nhưng để sim hoạt động đảm bảo thì dòng đủ 1A là được khuyến khích dùng hơn.
- Khe cắm SIM: MICROSIM
- Dòng khi ở chế độ chờ: 10mA
- Dòng khi hoạt động: 100mA đến 1A
- Hỗ trợ 4 băng tần phổ biến: GSM 850, EGSM 900, DCS 1800, PCS 1900.
- Nhiệt độ hoạt động: -40 C 85 C
- Tốc độ truyền downlink/uplink: cao nhất 85.6 kbps
- Tốc độ UART hỗ trợ: 1200bps đến 125200bps

Chức năng các chân của module Sim800L



Hình 2.15 Sơ đồ module sim800l

- TXD: chân truyền UART TX

- RXD: chân nhận UART RX
- DTR: chân UART DTR, thường ít xài.
- SPKP, SPKN: ngõ ra âm thanh, nối với loa để phát âm thanh.
- MICP, MICN: ngõ vào âm thanh, phải gắn thêm micro để thu âm thanh.
- Reset: chân khởi động lại SIM800L
- RING: báo cuộc gọi đến
- GND: chân Mass, cấp 0v.

### Một số tập lệnh AT của module Sim800L dùng trong đề tài luận văn

AT+CBC: trả về kết quả dung lượng pin hiện tại.

Lệnh AT	AT+CBC
Kết quả	<p>+ CBC: &lt;bcs&gt;, &lt;bcl&gt;, &lt;voltage&gt;</p> <p>OK</p> <p>Với các thông số</p> <p>&lt;bcs&gt;:</p> <ul style="list-style-type: none"> <li>• 0: Mạch không phải trạng thái đang sạc.</li> <li>• 1: Mạch đang trong trạng thái sạc.</li> <li>• 2: Mạch đã sạc xong.</li> </ul> <p>&lt;bcl&gt;: giá trị % dung lượng pin: 1 ... 100 %.</p> <p>&lt;voltage&gt;: Giá trị nguồn đầu vào của module sim (mV)</p>

Bảng 2.3 Tập lệnh AT+CBC

AT+ CCLK trả về giá trị thời gian của module sim

Lệnh AT	AT+CCLK
Kết quả	<p>+ CBC: &lt;time&gt;</p> <p>OK</p> <p>Với các thông số</p> <p>&lt;time&gt;: là một dạng chuỗi String, chuỗi String này được format dưới dạng <i>yy/MM/dd, hh : mm : ss + -zz</i></p>

Bảng 2.4 Tập lệnh AT+CCLK

AT+CSTT cài đặt các thông số APN, USER NAME, PASSWORD cho Sim

Lệnh AT	<p>AT+CSTT= &lt;apn&gt;,&lt;user&gt;,&lt;password&gt;</p> <p>&lt;apn&gt; Địa điểm truy cập GPRS</p> <p>&lt;user&gt; Tên đăng nhập</p> <p>&lt;pass&gt; Mật khẩu đăng nhập</p> <p>3 thông số trên được cung cấp theo từng nhà mạng của mỗi sim khi được gắn vào module sim, mục đích dùng để xác nhận việc truy cập GPRS vào các nhà mạng.</p> <p>Ví dụ thông số của nhà mạng vinaphone: &lt;m3-world&gt;,&lt;mms&gt;,&lt;mms&gt;</p>
Kết quả	+ OK

Bảng 2.5 Tập lệnh AT+CSTT

AT+CIFSR: Lấy thông tin địa chỉ IP của moduleSIM

Lệnh AT	AT+CIFSR
Kết quả	<p>&lt;IP address&gt;</p> <p>&lt;IP address&gt;: địa chỉ IP được gắn sau khi tham gia vào GPRS.</p>

Bảng 2.6 Tập lệnh AT+CIFSR

AT+CIPSTART: thiết lập kết nối TCP hoặc UDP

Lệnh AT	AT+CIPSTART="TCP","192.168.0.1","80" Kết nối tới địa chỉ 192.168.0.1 tại cổng 80 với giao thức TCP.  AT+CIPSTART="UDP","192.168.0.1","8888" Kết nối tới địa chỉ 192.168.0.1 tại cổng 8888 với giao thức UDP.
Kết quả	+ OK  CONNECT OK

Bảng 2.7 Tập lệnh AT+CIPSTART

AT+CDNSGIP: được dùng để phân giải DNS của một tên miền, nó được sử dụng cho việc thiếp lập kết nối TCP/UCP thông qua tên miền bằng câu lệnh “AT+CIPSTART=<mode>,<domain name>, <port>”, và sau đó người dùng có thể gửi dữ liệu đến máy chủ thông qua câu lệnh AT+CIPSEND.

Lệnh AT	AT+CDNSGIP=<domain name>  <domain name>: Tên miền muốn phân giải DNS, ví dụ như AT+CDNSGIP= “www.codingyourfuture.com “
Kết quả	OK + CDNSGIP: 1,”www.codingyourfuture.com”,”113.173.153.194”,”119.75.217.56”

Bảng 2.8 Tập lệnh AT+CDNSGIP

AT+CIPSEND: gửi dữ liệu thông qua giao thức kết nối TCP/UCP

Lệnh AT	AT+CIPSEND=<length>  <length>: độ dài dữ liệu muốn gửi.
Kết quả	>
Lệnh AT	Dữ liệu muốn gửi đi
Kết quả	OK

Bảng 2.9 Tập lệnh AT+CIPSEND

### 2.5.3 Pin năng lượng mặt trời

Pin năng lượng mặt trời (pin mặt trời/pin quang điện) là thiết bị giúp chuyển hóa trực tiếp năng lượng ánh sáng mặt trời (quang năng) thành năng lượng điện (điện năng) dựa trên hiệu ứng quang điện. Hiệu ứng quang điện là khả năng phát ra điện tử (electron) khi được ánh sáng chiếu vào của vật chất.

Tấm pin mặt trời, những tấm có bề mặt lớn thu thập ánh nắng mặt trời và biến nó thành điện năng, được làm bằng nhiều tế bào quang điện có nhiệm vụ thực hiện quá trình tạo ra điện từ ánh sáng mặt trời.



Hình 2.16 Tấm pin năng lượng mặt trời

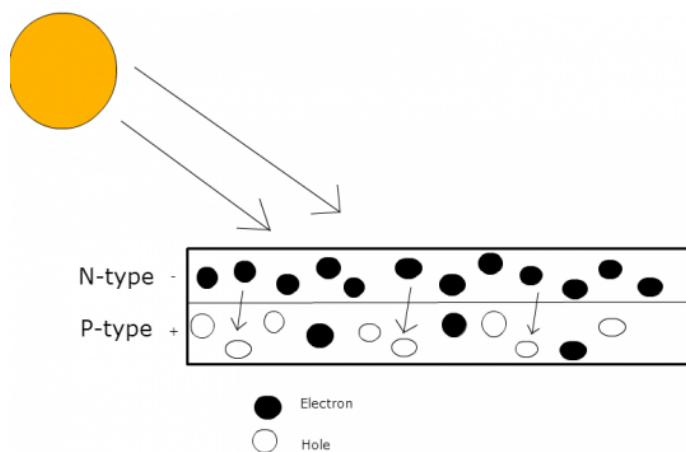
#### Chất bán dẫn Silicon

Silicon được biết đến là một chất bán dẫn. "Chất bán dẫn là vật liệu trung gian giữa chất dẫn điện và chất cách điện. Chất bán dẫn hoạt động như một chất cách điện ở

nhiệt độ thấp và có tính dẫn điện ở nhiệt độ phòng". Với tính chất như vậy, silicon là một thành phần quan trọng trong cấu tạo của pin năng lượng mặt trời.

Silicon tuy có mức dẫn điện hạn chế nhưng nó có cấu trúc tinh thể rất phù hợp cho việc tạo ra chất bán dẫn. Nguyên tử silicon cần 4 electron để trung hòa điện tích nhưng lớp vỏ bên ngoài một nguyên tử silicon chỉ có một nửa số electron cần thiết nên nó sẽ bám chặt với các nguyên tử khác để tìm cách trung hòa điện tích.

Để tăng độ dẫn điện của silicon, các nhà khoa học đã "tạp chất hóa" nó bằng cách kết hợp nó với các vật liệu khác. Quá trình này được gọi là "doping" và silicon pha tạp với các tạp chất tạo ra nhiều electron tự do và lỗ trống. Một chất bán dẫn silicon có hai phần, mỗi phần được pha tạp với một loại vật liệu khác. Phần đầu tiên được pha với photpho, photpho cần 5 electron để trung hòa điện tích và có đủ 5 electron trong vỏ của nó. Khi kết hợp với silicon, một electron sẽ bị dư ra. Electron đặc trưng cho điện tích âm nên phần này sẽ được gọi là silicon loại N (điện cực N). Để tạo ra silicon loại P (điện cực P), các nhà khoa học kết hợp silicon với boron. Boron chỉ cần 3 electron để trung hòa điện tích và khi kết hợp với silicon sẽ tạo ra những lỗ trống cần được lấp đầy bởi electron.



Hình 2.17 Tấm pin năng lượng mặt trời

Khi chất bán dẫn silicon tiếp xúc với năng lượng, các electron tự do ở điện cực N sẽ di chuyển sang để lấp đầy các lỗ trống bên điện cực P. Sau đó, các electron từ điện cực N và điện cực P sẽ cùng nhau tạo ra điện trường. Các tế bào năng lượng mặt trời sẽ trở thành một diode, cho phép electron di chuyển từ điện cực P đến điện cực N, không cho phép di chuyển ngược lại.

Tất nhiên, để kích hoạt quá trình cần có năng lượng tiếp xúc với các tế bào silicon. Ánh sáng mặt trời bao gồm các hạt rất nhỏ gọi là photon được tỏa ra từ mặt trời, các hạt nhỏ năng lượng có thể tiếp xúc với các tế bào năng lượng mặt trời và nối lồng liên kết của các electron ở điện cực N. Sự di chuyển của các electron tự do từ điện cực N tới điện cực P tạo ra dòng điện.

### **Hiệu suất của pin mặt trời**

Các công nghệ biến ánh sáng mặt trời thành điện hiện tại vẫn kém hiệu quả. Các tấm pin mặt trời chưa thể hấp thụ toàn bộ năng lượng của ánh sáng mặt trời. Nói chung, những tế bào năng lượng mặt trời tốt nhất hiện tại chỉ có thể chuyển 25% năng lượng mà nó nhận được thành điện. Tại sao vậy? Thực tế là ánh sáng mặt trời, như tất cả các loại ánh sáng khác, bao gồm một quang phổ với các bước sóng khác nhau, mỗi bước sóng có một cường độ khác nhau. Có những bước sóng quá yếu không thể giải phóng các electron còn một số bước sóng lại quá mạnh với silicon.

## **2.6 Kiến thức căn bản về ứng dụng web - di động và xây dựng Server**

### **2.6.1 Giao thức TCP**

TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các

máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.[9]

TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có WWW, thư điện tử và Secure Shell.

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin không đáng tin cậy. TCP làm nhiệm vụ của tầng giao vận trong mô hình OSI đơn giản của các mạng máy tính.

Các ứng dụng gửi các dòng gồm các byte 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các đoạn (segment) có kích thước thích hợp (thường được quyết định dựa theo kích thước của đơn vị truyền dẫn tối đa (MTU) của tầng liên kết dữ liệu của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một liên mạng tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (sequence number). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin báo nhận" (acknowledgement) cho các gói tin đã nhận được thành công; một "đồng hồ" (timer) tại nơi gửi sẽ báo time-out nếu không nhận được tin báo nhận trong khoảng thời gian bằng một round-trip time (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (giá trị kiểm tra) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

## 2.6.2 Web API

Web API là giao diện lập trình ứng dụng (API – Application Programming Interface) cho cả web server và web browser.[7]

Web API giúp người phát triển xây dựng lên các Service cung cấp dịch vụ cho các ứng dụng web, window... Trước web API, để có các service API người dùng phải cấu hình, xây dựng các ứng dụng wcf, web service khá phức tạp. Các ứng dụng client như website, ứng dụng winform, wpf có thể kết nối vào Web API để lấy các dữ liệu về xử lý, cũng như cập nhật thông tin lại Web API.

Web API dùng phương thức trao đổi dữ liệu là HTTP, kiểu dữ liệu trao đổi có thể là JSON, XML, hoặc một chuẩn dữ liệu bất kỳ. Đây là những chuẩn dữ liệu hướng đối tượng được dùng khá nhiều trong việc lưu chuyển thông tin trên Internet, do đó các trang web sử dụng web API tương tác dữ liệu có tốc độ khá cao. Ngoài ra do Web API dùng giao thức HTTP nên hầu như tất cả các ứng dụng trên các công nghệ đều có thể kết nối tới để lấy cũng như tương tác với web API cụ thể như chúng ta có thể dùng các công nghệ web như: Asp.NET (MVC, Web Page, WebForm), PHP, jsp hay các ứng dụng desktop như: winform, wpf đều có thể dễ dàng kết nối tới web API. Với Web API chúng ta có thể xây dựng và phân tách các ứng dụng web lớn, cấu hình từng thành phần riêng biệt của website: đâu là tầng data, đâu là tầng xử lý, đâu là tầng dịch vụ, ... Nền tảng của các ứng dụng lớn luôn là các service để các website thành viên có thể kết nối tương tác dữ liệu. Do đó với Web API chúng ta có thể ứng dụng vào các dự án web (cũng như window) lớn để phát triển trên nhiều tầng xử lý khác nhau. Dùng Web API chúng ta dễ dàng xây dựng các ứng dụng window kiểu toán (dữ liệu ở server) còn client chỉ cài giao diện, hay có thể xây dựng các website Single Page Application (SPA – tất cả web chỉ gói gọn trong một trang). Ứng dụng này tương tác khá cao với người dùng, tốc độ nhanh (do dùng ajax, sẽ được nói rõ hơn ở phần

sau) thường được dùng làm các website tương tác với các thiết bị di động (các thiết bị di động thường có kết nối Internet chậm).

### 2.6.3 Căn bản về RESTful Web services

#### Tổng quan về REST – Representational State Transfer

REST là một kiểu kiến trúc dựa trên các tiêu chuẩn web và giao thức HTTP. REST được mô tả đầu tiên bởi Roy Fielding trong luận văn tiến sĩ của ông vào năm 2000.

Trong một kiến trúc REST thì tất cả mọi thứ là một nguồn tài nguyên. Một nguồn tài nguyên được truy cập thông qua một giao diện phổ biến dựa trên các phương pháp tiêu chuẩn HTTP.

Một kiến trúc REST thường có một REST server cung cấp truy cập đến các nguồn tài nguyên và một Rest client truy cập và trình sửa các tài nguyên đó.

REST cho phép các nguồn tài nguyên biểu diễn dưới nhiều kiểu khác nhau như Text, XML, JSON ... Rest client có thể yêu cầu một kiểu biểu diễn cụ thể thông qua giao thức HTTP.

#### Các phương thức HTTP

Các phương thức PUT, GET, POST và DELETE thường được sử dụng trong kiến trúc REST

- **GET** định nghĩa một truy cập đọc nguồn tài nguyên mà không có tác dụng phụ. Nguồn tài nguyên không bao giờ thay đổi thông qua một yêu cầu GET.
- **PUT** dùng để tạo một tài nguyên mới.
- **DELETE** loại bỏ các nguồn tài nguyên.
- **POST** cập nhật một nguồn tài nguyên hiện có hoặc tạo ra một nguồn tài nguyên mới.

#### RESTful webservice

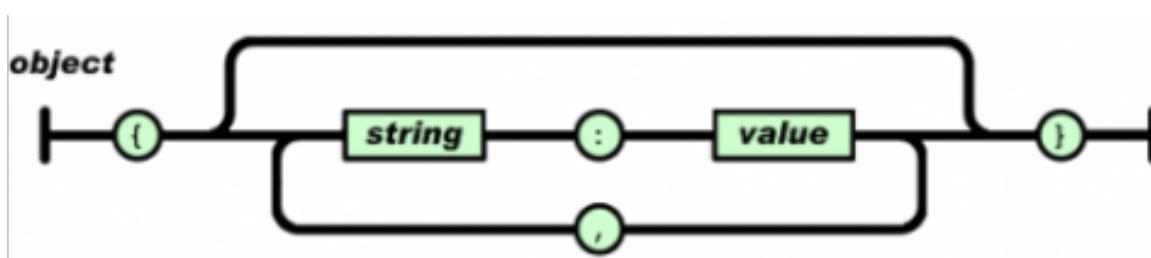
Một RESTful webservice được dựa trên các phương thức HTTP và các khái niệm về REST. Một RESTful webservice thường xác định URI cơ sở cho các dịch vụ, hỗ trợ các kiểu MIME (XML, Text, JSON,...) và tập các hoạt động (POST, GET, PUT, DELETE) đều được hỗ trợ.

### JSON (JavaScript Object Notation)

Là một định dạng hoán vị dữ liệu nhanh. Chúng dễ dàng cho chúng ta đọc và viết. Dễ dàng cho thiết bị phân tích và phát sinh. Chúng là cơ sở dựa trên tập hợp của Ngôn Ngữ Lập Trình JavaScript, tiêu chuẩn ECMA-262 phiên bản 3 – tháng 12 năm 1999. JSON là một định dạng kiểu text mà hoàn toàn độc lập với các ngôn ngữ hoàn chỉnh, thuộc họ hàng với các ngôn ngữ họ hàng C, gồm có C, C++, C, Java, JavaScript, Perl, Python, và nhiều ngôn ngữ khác. Những đặc tính đó đã tạo nên JSON một ngôn ngữ hoán vị dữ liệu lý tưởng.

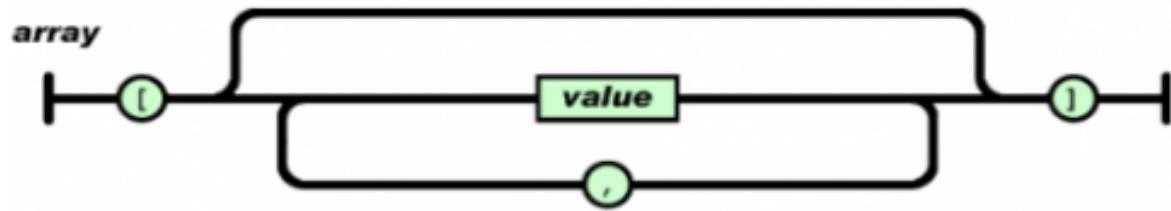
JSON được xây dựng trên 2 cấu trúc:

- Là tập hợp của các cặp tên và giá trị name – value. Trong những ngôn ngữ khác nhau, đây được nhận thấy như là một đối tượng (object), sự ghi (record), cấu trúc (struct), từ điển (dictionary), bảng băm (hash table), danh sách khóa (keyed list), hay mảng liên hợp.
- Là một tập hợp các giá trị đã được sắp xếp. Trong hầu hết các ngôn ngữ, nó được nhận thấy như là một mảng, vector, list hay là một dãy sequence.



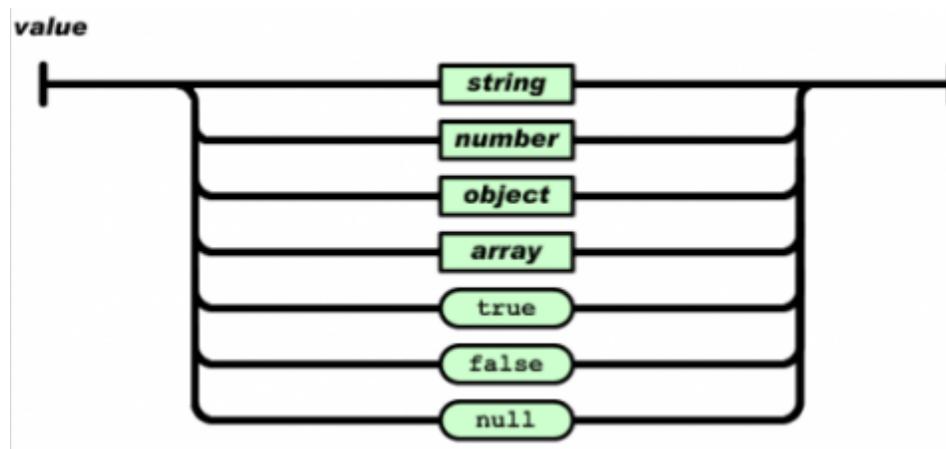
Hình 2.18 Cấu trúc của một đối tượng JSON

Một đối tượng là một tập hợp của các cặp tên/giá trị (name/value). Một đối tượng bắt đầu bởi dấu ““ và kết thúc với dấu “”. Từng tên được sau bởi dấu “:” và các cặp tên/giá trị được tách ra bởi dấu “,”.



Hình 2.19 Cấu trúc mảng của JSON

Một mảng là một tập hợp các giá trị đã được xắp xếp, một mảng bắt đầu bởi dấu “[” và kết thúc với dấu ”]”. Các giá trị được cách nhau bởi dấu “,”.



Hình 2.20 Cấu trúc của một đối tượng JSON

Một giá trị có thể là chuỗi, số, đúng sai, hoặc là đối tượng hoặc mảng. Những cấu trúc này có thể được lồng vào nhau.

### JSON và Webservice

Việc trao đổi thông tin, truyền tải dữ liệu giữa Webservice và ứng dụng người ta thường dùng XML hay JSON vì sự tiện nghi và cấu trúc dữ liệu nhỏ gọn của nó. JSON là một ngôn ngữ mới với tuổi đời con trẻ nhưng vẫn được ứng dụng nhiều bởi những điểm lợi của nó:

- “Con người” có thể đọc được.
- Cú pháp gần với JavaScript nên rất dễ để sử dụng.
- Dữ liệu truyền tải ngắn gọn so với những định dạng dữ liệu như: XML, HTML...
- Việc phân tích (parse) dữ liệu từ dạng chuỗi (nhận từ server) sang dữ liệu có thể sử dụng được (thành Object, Number, Array) dễ dàng.

```


personal.xml



```

4 <personnel>
5   <person id="Big.Boss">
6     <name>
7       <family>Boss</family>
8       <given>Big</given>
9     </name>
10    <email>chief@oxygencode.com</email>
11    <link subordinates="one.worker"/>
12  </person>
13  <person id="one.worker">
14    <name>
15      <family>Worker</family>
16      <given>One</given>
17    </name>
18    <email>one@oxygencode.com</email>
19    <link manager="Big.Boss"/>
20  </person>
21  <person id="two.worker">
22    <name>
23      <family>Worker</family>
24      <given>Two</given>
25    </name>
26    <email>two@oxygencode.com</email>
27    <link manager="Big.Boss"/>
28  </person>
29  <person id="three.worker">
30    <name>

```



personal.json



```

1  {"personnel": {"person": [
2    {
3      "id": "Big.Boss",
4      "name": {
5        "family": "Boss",
6        "given": "Big"
7      },
8      "email": "chief@oxygencode.com",
9      "link": {"subordinates": "one.worker"}
10    },
11    {
12      "id": "one.worker",
13      "name": {
14        "family": "Worker",
15        "given": "One"
16      },
17      "email": "one@oxygencode.com",
18      "link": {"manager": "Big.Boss"}
19    },
20    {
21      "id": "two.worker",
22      "name": {
23        "family": "Worker",
24        "given": "Two"
25      },
26      "email": "two@oxygencode.com",
27      "link": {"manager": "Big.Boss"}
28    }
29  ]}}

```


```

Hình 2.21 Cấu trúc mô tả XML-JSON

Nhằm mục đích giúp thời gian tương tác giữa người sử dụng với ứng dụng tiến gần đến thời gian thực, thì tốc độ được đặt lên hàng đầu, việc tăng tốc bằng cách “giảm tải” cũng là một giải pháp, chính vì vậy JSON được ưu ái hơn vì cú pháp ngắn gọn, đơn giản, tiết kiệm hơn XML.

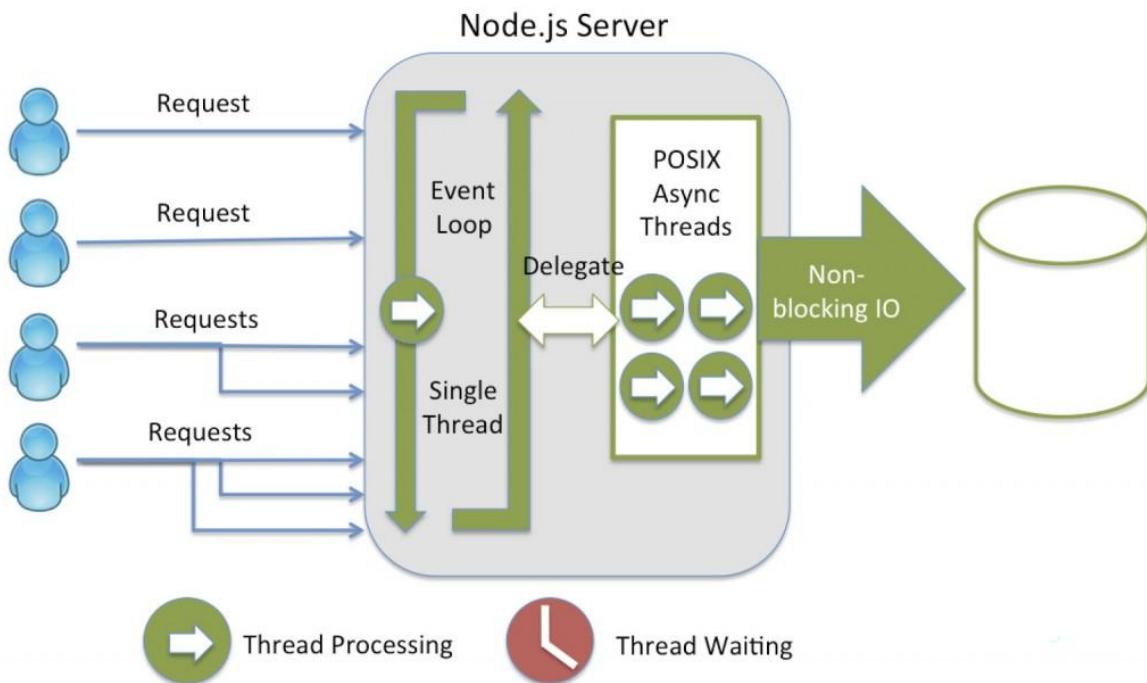
### 2.6.4 NodeJS

NodeJS là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009. Định nghĩa Nodejs bởi tài liệu chính thức : Node.js là một nền tảng dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. [6]



Hình 2.22 Nền tảng Nodejs

Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.



Hình 2.23 Mô hình hoạt động Nodejs

### 2.6.5 Javascripts

JavaScript, theo phiên bản hiện hành, là một ngôn ngữ lập trình kịch bản dựa trên đối tượng được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web, nhưng cũng được dùng để tạo khả năng viết script sử dụng các đối tượng nằm sẵn trong các ứng dụng. Nó vốn được phát triển bởi Brendan Eich tại Hãng truyền thông Netscape với cái tên đầu tiên Mocha, rồi sau đó đổi tên thành LiveScript, và cuối cùng thành JavaScript. Giống Java, JavaScript có cú pháp tương tự C, nhưng nó gần với Self hơn Java. .js là phần mở rộng thường được dùng cho tập tin mã nguồn JavaScript.[8]

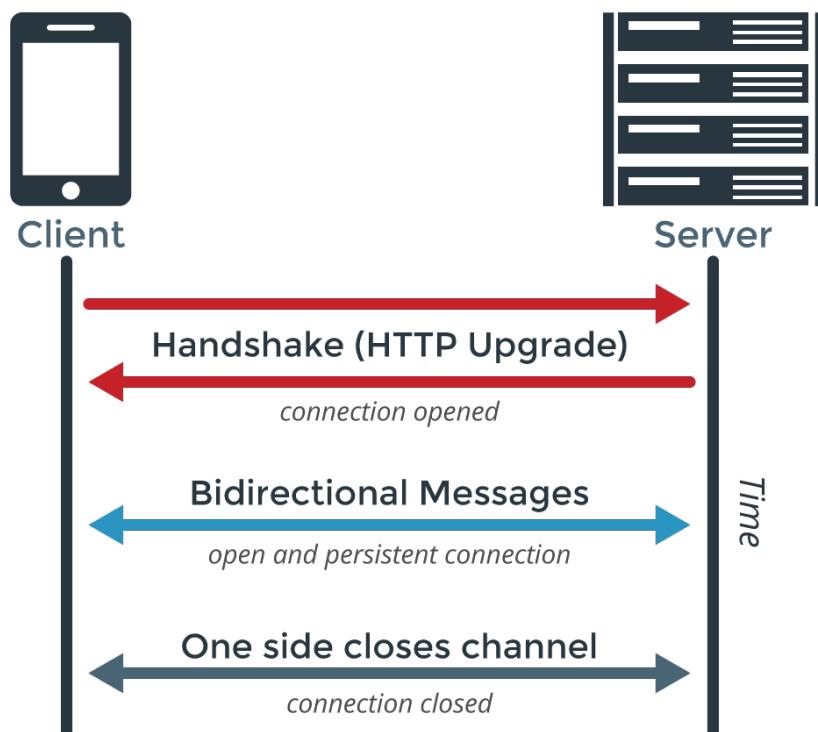
Những ứng dụng to lớn của Javascript khiến người ta không thể quên nó được. Hiện nay có rất nhiều libraries, Framework được viết như: [2]

- AngularJS: Một thư viện dùng để xây dựng ứng dụng Single Page

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

### 2.6.6 Websocket - Socket.io

WebSoket là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.[1]



Hình 2.24 Mô hình hoạt động Websocket[3]

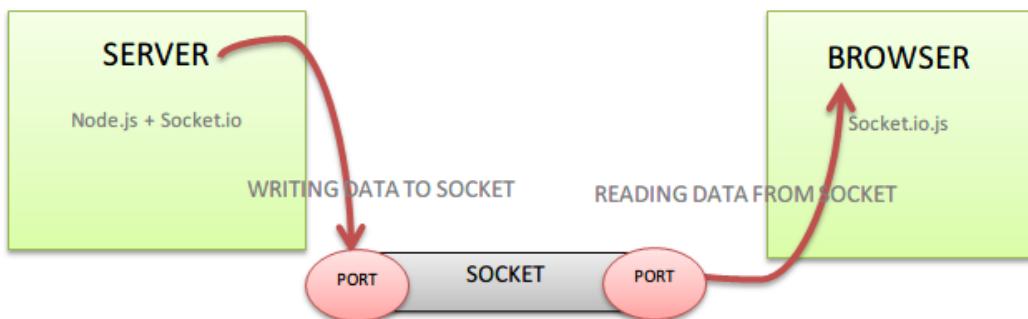
Kết nối được duy trì có thể viết và nhận dữ liệu bằng JavaScript như khi bạn đang sử dụng một TCP socket đơn thuần.

So sánh dung lượng giữa hai hình thức sử dụng polling HTTP và Websocket trong việc truyền 10000 gói tin:

- HTTP:  $871 \times 10,000 = 8,710,000$  bytes
- WebSocket:  $2 \times 10,000 = 20,000$  bytes

### 2.6.7 Socket.IO

Socket.IO là một thư viện javascript có mục đích tạo ra các ứng dụng realtime trên trình duyệt cũng như thiết bị di động. Việc sử dụng thư viện này cũng rất đơn giản và giống nhau ở cả server lẫn client.

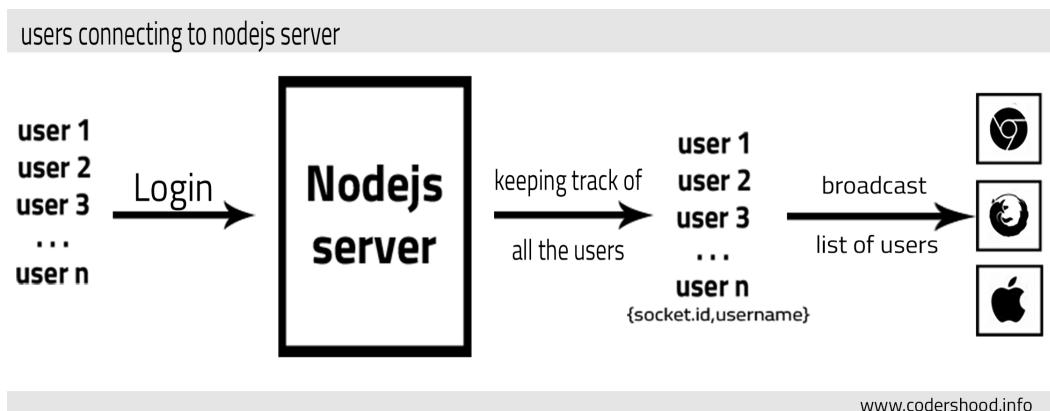


Hình 2.25 Mô hình cơ bản của Socket.IO

#### Tại sao Socket.IO ra đời?

- Javascript là ngôn ngữ lập trình hướng sự kiện, mà trong lập trình thời gian thực, cách tiếp cận bằng lập trình sự kiện là cách tiếp cận khôn ngoan nhất.
- Node.js chạy non-blocking việc hệ thống không phải tạm ngừng để xử lý xong một request sẽ giúp cho server trả lời client gần như ngay tức thì.
- Lập trình socket yêu cầu bạn phải xây dựng được mô hình lắng nghe – trả lời từ cả 2 bên. Nói khác đi, vai trò của client và server phải tương đương nhau, mà client thì chạy bằng javascript, nên nếu server cũng chạy bằng javascript nữa, thì việc lập trình sẽ dễ dàng và thân thiện hơn.

Bởi vì thế, socket.io được ra đời để phát triển việc giao tiếp theo thời gian thực giữa client và server dựa trên sức mạnh có sẵn của Javascript và Nodejs.



Hình 2.26 Mô hình hoạt động giao tiếp của Socket.IO

Socket.IO hoạt động dựa trên các events tương tự như Websocket:

- **connect()**: kết nối với server socket
- **on(event\_name, listener)**: đăng ký lắng nghe sự kiện từ server trả về
- **emit(event\_name, data)**: gửi một sự kiện lên server
- **off(event\_name)**: ngừng lắng nghe một sự kiện nào đó

## 2.6.8 RethinkDB

### RethinkDB là gì?

RethinkDB là một mã nguồn mở, dựa theo cấu trúc cơ sở dữ liệu NoSQL [4], tổ chức dữ liệu theo kiểu document-oriented database - một thiết kế riêng biệt cho việc lưu trữ document. Các cài đặt có thể là giả lập tương tác trên relational database, object database hay key-value store. Cơ sở dữ liệu được lưu trữ dưới dạng JSON với các giản đồ động, và được thiết kế giúp hoạt động dễ dàng trong việc cập nhật và truy xuất dữ liệu theo thời gian thực với các ứng dụng. [5]



# RethinkDB

Hình 2.27 Biểu tượng của RethinkDB

RethinkDB sử dụng ngôn ngữ truy vấn ReQL được phát triển theo hướng đối tượng. Và ngôn ngữ này hỗ trợ cho các thao tác nhập bảng, quản lý nhóm, tích hợp và các hàm chức năng truy vấn cũng như event theo thời gian thực.

```
r.table('game').orderBy('score').limit(3).changes()
```

● STREAMING RETHINKDB RESULTS...

```
{'player': 'matt',
 'score': 129}
{'player': 'tim', 'score': 115}
{'player': 'tim', 'score': 146}
{'player': 'yoshiko',
 'score': 154}
{'player': 'tim', 'score': 140}
{'player': 'pierre',
 'score': 67}
{'player': 'anthony',
 'score': 112}
{'player': 'connor',
 'score': 53}
{'player': 'marshall',
 'score': 126}
```

TOP PLAYER SCORES

1. yoshiko: 154 points
2. tim: 146 points
3. tim: 140 points

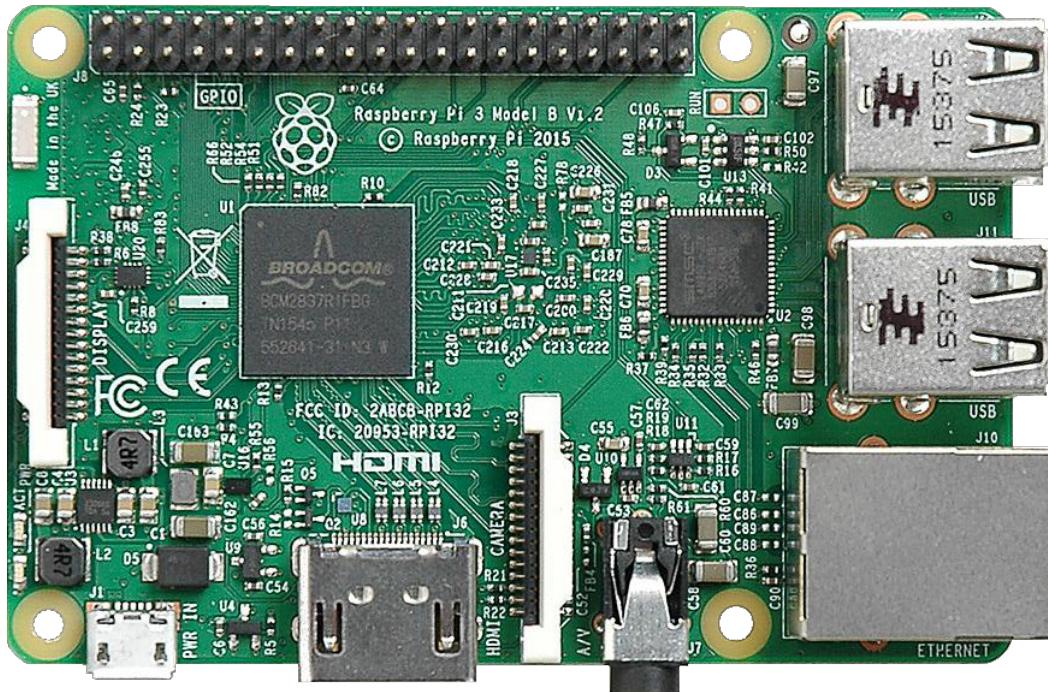
Hình 2.28 Đoạn code mẫu RethinkDB

Như hình 2.28 là ví dụ trực quan được demo trực tiếp tại trang chủ <https://www.rethinkdb.com>, có thể nhận thấy được cách thức khai báo và sử dụng RethinkDB:

- **r** : là đối tượng RethinkDB chúng ta khai báo trong ứng dụng.
- **table()** : là bảng giá trị đang truy vấn.
- **orderBy()** : hàm sắp xếp dữ liệu theo trường giá trị được khai báo.
- **limit()** : chọn ra những giá trị trong phạm vi được khai báo trong hàm.
- **changes()** : đây chính là đặc điểm khiến RethinkDB khác biệt so với các hệ quản lý dữ liệu khác. RethinkDB sẽ gọi hàm thực thi khi vùng giá trị truy vấn có thay đổi giá trị theo thời gian thật. không hoạt động theo cơ chế polling thường thấy ở các kiểu truy vấn dữ liệu khác.

### 2.6.9 Raspberry

Raspbian là hệ điều hành mở và miễn phí được phát triển tối ưu hóa dựa trên nền Debian dành riêng cho thiết bị phần cứng Raspberry Pi. Hệ điều hành này bao gồm nhóm các chức năng và ứng dụng giúp Raspberry hoạt động. Hơn nữa, Raspbian cung cấp nhiều hơn là hệ điều hành thuần: hỗ trợ tới hơn 35,000 gói mở rộng và dễ dàng cài đặt trên Raspberry Pi. Raspbian ổn định và hiệu suất cao được hoàn thiện vào tháng 6 năm 2012. Tuy nhiên Raspbian vẫn đang được phát triển tới ngày nay và ngày càng cải thiện hiệu suất cũng như hỗ trợ các gói chức năng nhiều nhất có thể. [10]



Hình 2.29 Thiết bị Raspberry Pi

Vì sao lựa chọn Raspberry:

- Là một máy tính độc lập chạy trên nền tảng Linux, điều này làm cho việc chạy các ứng dụng Nodejs hay RethinkDB hoạt động được trên Raspberry Pi.
- Hiệu suất đáp ứng nhu cầu làm server của đề tài yêu cầu với mức độ tiêu thụ năng lượng cực kì ít.

### 2.6.10 Framework Ionic

#### Giới thiệu về Ionic framework

Ionic là một framework được sử dụng để phát triển các ứng dụng di động dựa trên nền tảng công nghệ web HTML5 để tạo ra các ứng dụng lai hóa [ hybrid apps ] chạy trên các thiết bị di động khác nhau. Ứng dụng lai hóa ở đây được hiểu cơ bản là các ứng dụng chạy trên nền các trình duyệt được nhúng vào trong một ứng dụng được cài đặt

và có thể giao tiếp, truy cập, điều khiển sử dụng các thiết bị phần cứng và hệ điều hành mobile.

Hãy tưởng tượng Ionic như là một khung phát triển giao diện người dùng để xử lý tất cả các tương tác giao diện người dùng làm cho ứng dụng trở nên thuyết phục và dễ dàng sử dụng. Kiểu như “Bootstrap for Native”, tức là như các ứng dụng Native nhưng có sự hỗ trợ bởi một loạt các thành phần giao diện, động và thiết kế đẹp.



Hình 2.30 Một số giao diện của Ionic

Bản chất Ionic là một khung phát triển HTML5, nó cần một vỏ bọc native như Cordova hay PhoneGap để có thể khởi động như một ứng dụng native. Ionic được hỗ trợ mạnh mẽ bởi Cordova. Việc cài đặt và khởi tạo một dự án Ionic rất dễ dàng thông qua giao diện dòng lệnh dựa trên Node.js

### Ưu điểm và nhược điểm của Ionic

Ưu điểm:

- Dễ học, thời gian phát triển nhanh, có thể sử dụng các kỹ năng từ lập trình web.
- Da nền tảng.

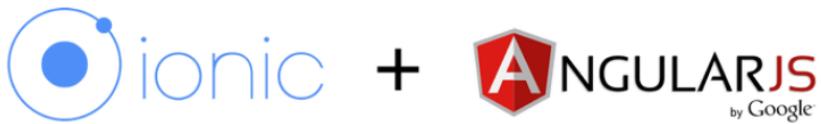
- Khả năng truy cập đến các tính năng của thiết bị và hệ điều hành như bluetooth, camera,..
- Dễ dàng thiết kế giao diện cho các thiết bị có kích cỡ khác nhau
- Việc sử dụng AngularJS làm core giúp phần xử lý UI linh động hơn so với javascript hay thư viện Jquery.
- Việc sử dụng AngularJS làm core cũng mang lại lợi thế lớn so với các framework cho ứng dụng hybrid khác.
- Ionic cung cấp đầy đủ các thành phần trong giao diện người dùng như Pull-to-Refresh, Infinite-loader, tabs, ...

Nhược điểm:

- Vẫn còn trong giai đoạn phát triển.
- Hiệu năng vẫn chưa cao và ổn định.
- Cộng đồng phát triển ứng dụng vẫn còn chưa đông.

### 2.6.11 Framework AngularJS

Angular là một bộ Javascript Framework rất mạnh và thường được sử dụng để xây dựng project Single Page Application (SPA). Nó hoạt động dựa trên các thuộc tính mở rộng HTML (các attributes theo quy tắc của Angular). Đây là một Framework mã nguồn mở hoàn toàn miễn phí và được hàng ngàn các lập trình viên trên thế giới ưa chuộng và sử dụng.



Hình 2.31 Angular JS kết hợp với ionic

Ionic sử dụng AngularJS để cung cấp các cấu trúc ứng dụng, trong khi bản thân Ionic tập trung chính vào giao diện người dùng. Nói cách khác, chúng ta thấy được sự phối hợp ăn ý giữa sức mạnh của AngularJS và vẻ đẹp của Ionic UI.

Ionic cung cấp một tập các Angular directives (nghĩa là các phần tử HTML tùy biến) để làm các thành phần của nó, tạo ra sự dễ dàng để sử dụng các tiện ích gọn để viết mã HTML. Ngoài các directives, Ionic còn sử dụng và thêm vào các thành phần khác như: Angular touch recognizers, view animation logic, HTML sanitation, và asynchronous communication.

Việc xây dựng ứng dụng dựa trên AngularJS đòi hỏi mã nguồn phải có khả năng mở rộng cao để bổ sung các tính năng mới. Tuy nhiên với Ionic, người ta có thể tái sử dụng các chức năng trong ứng dụng trên các nền tảng khác nhau đồng thời vẫn có thể tùy chỉnh giao diện người dùng cho mỗi nền tảng riêng biệt. Các thành phần trong Ionic như danh sách, slide,... chính là các directive(các thuộc tính của thẻ HTML dùng trong Angular) của AngularJS. Đó là lí do khiến cho Ionic và AngularJS kết hợp rất tốt với nhau.

Mặc dù, Ionic có thành phần giao diện sử dụng Angular, nhưng developers vẫn có thể sử dụng và hỗ trợ các framework khác như Knockout hay EmberJS.

### 2.6.12 Framework Cordova



Hình 2.32 Apache Cordova

Apache Cordova là một bộ khung để xây dựng ứng dụng di động sử dụng HTML, CSS và Javascript. Apache Cordova bao gồm một tập hợp các API thiết bị cho phép người lập trình di động truy cập, sử dụng các chức năng native của thiết bị như là camera hay cảm biến gia tốc bằng Javascript. Kết hợp với một bộ khung phát triển giao diện như jQuery Mobile or Dojo Mobile hoặc Ionic, cho phép ứng dụng di động có thể được phát triển chỉ dựa trên HTML, CSS và Javascript.

Khi sử dụng Cordova API, một ứng dụng có thể được xây dựng mà không phải sử dụng bất kỳ một đoạn mã native code nào. Thay vào đó, công nghệ web sẽ được sử dụng, và chúng sẽ được tổ chức trên chính ứng dụng đầy chữ không cần thông qua một server nào.

Cordova cung cấp một tập hợp các thư viện Javascript đã được chuẩn hóa để có thể sử dụng. Cordova hiện có thể sử dụng cho các nền tảng như iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada và Symbian.

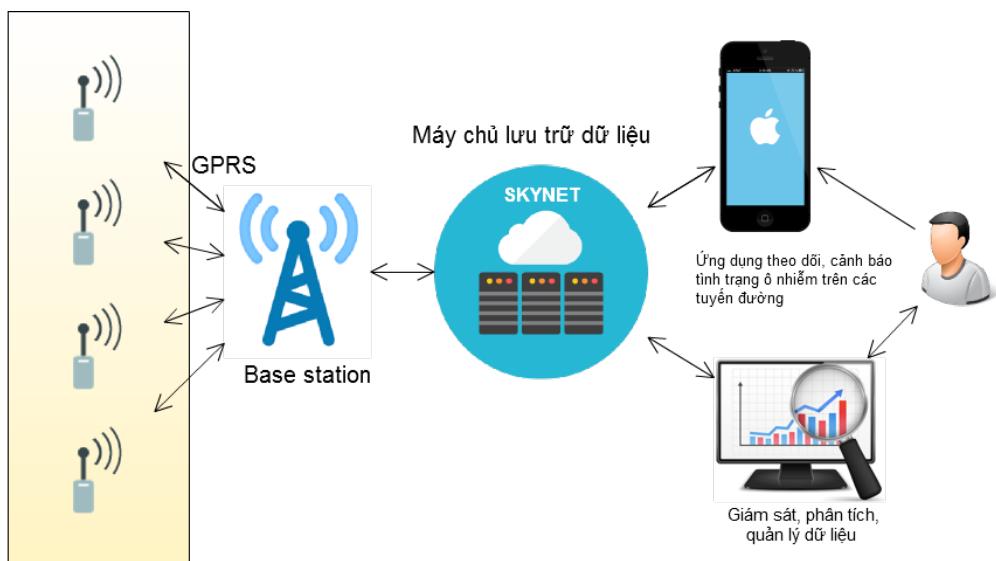
# **Chương 3**

## **THIẾT KẾ VÀ HIỆN THỰC**

### **3.1 Thiết kế hệ thống**

#### **3.1.1 Kiến trúc mô hình hệ thống**

Như đã giới thiệu một số mô hình IoT thực tế ở Mục 2.1.1, mô hình IoT phù hợp mà nhóm sẽ áp dụng cho việc pháp triển hệ thống giám sát được trình bày như Hình 2.2. Hệ thống bao gồm các thành phần chính như sau: các node cảm biến, trung tâm lưu trữ dữ liệu và ứng dụng truy xuất và hiển thị thông tin phân tích dữ liệu.



Các Node cảm biến đẩy dữ liệu

Hình 3.1 Kiến trúc mô hình hệ thống

### 3.1.2 Các node cảm biến

Các node cảm biến: là một thiết bị được lắp đặt trên những đoạn đường hoặc các địa điểm thích hợp cần để lấy dữ liệu tại khu vực đó, nhiệm vụ chủ yếu là lấy dữ liệu từ các cảm biến đo được sau đó gửi dữ liệu đó lên máy chủ bằng module Sim GPRS. Để các node cảm biến này được hoạt động dưới tác nhân của môi trường như gió, mưa, bụi, và hơn hết có thể tận dụng nguồn năng lượng mặt trời để nạp pin cho mạch hoạt động độc lập với điện lưới, chúng tôi đã đề xuất mô hình thiết kế node cảm biến để có thể đáp ứng các yêu cầu như sau:



Hình 3.2 Mô hình căn nhà 3D

Lấy ý tưởng từ căn nhà gồm 4 bức tường và 2 mái che như Hình 3.2, tấm năng lượng mặt trời được thiết kế và gắn như mái che của căn nhà, còn các mạch điện và cảm biến sẽ được đặt bên trong ngôi nhà xung quanh các bức tường được làm từ nhựa mica. Việc lắp đặt node cảm biến này theo hướng các tấm năng lượng mặt trời 1 phía hướng về đông và 1 phía hướng về tây nhằm giúp tấm năng lượng có thể lấy được năng lượng tối đa.

Từ kết quả tìm hiểu các loại khí thải trong mục 2.3 đã được nêu kết hợp với tình trạng các loại cảm biến hiện có trên thị trường hiện nay thì nhóm chúng tôi đã quyết định đưa ra các loại cảm biến cần thiết cho một node cảm biến như sau:

#### Cảm biến bụi GP2

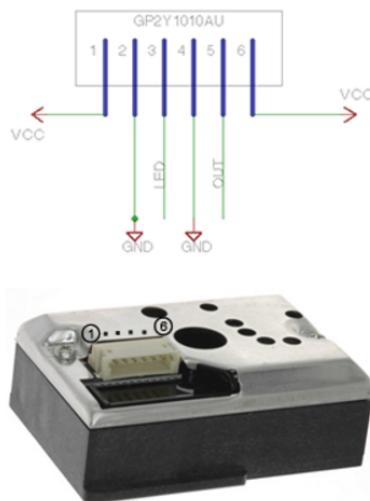
Cảm biến bụi GP2 là một bộ cảm biến chất lượng không khí quang học, được thiết kế đo những hạt bụi. Cảm biến được thiết kế một diốt phát hồng ngoại và một phototransistor được sắp xếp thành đường chéo thiết bị này, cho phép nó phát hiện ánh sáng phản xạ của bụi trong không khí. Nó đặc biệt hiệu quả trong việc phát hiện

các hạt rất mịn như khói thuốc lá, và thường được sử dụng trong các hệ thống lọc không khí.

Cảm biến bụi GP2 tiêu tốn dòng rất ít (20 mA cao nhất, 11mA chế độ chạy thường), có thể hỗ trợ nguồn cung cấp lên tới 7VDC. Tín hiệu đầu ra của cảm biến là dạng tín hiệu tương tự dùng để đo mật độ bụi, với độ nhạy  $0.5V/0.1mg/m^3$ .

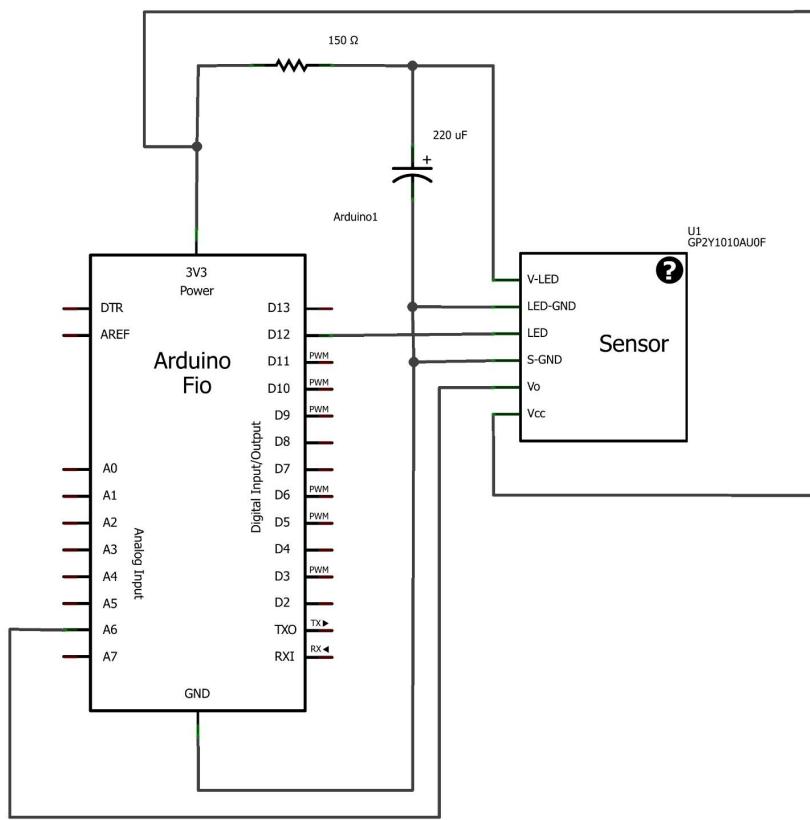
Thông số kỹ thuật:

- Nguồn hoạt động: 5VDC
- Dòng tiêu thụ: 10mA
- Ngõ ra analog
- Nhiệt độ hoạt động:  $-40^\circ - 85^\circ$  C



Hình 3.3 Cảm biến bụi GP2

Sơ đồ kết nối cảm biến bụi GP2 với Arduino Theo hướng dẫn của tài liệu kỹ thuật, nó có tất cả gồm 6 chân đầu ra để kết nối tới thiết bị. Để có thể cảm biến bụi GP2 hoạt động với vi xử lý Arduino cần kết nối như sơ đồ luận lý sau:



Made with Fritzing.org

Hình 3.4 Sơ đồ GP2 kết nối với Arduino

### Cảm biến chất lượng không khí MQ135

Cảm biến chất lượng không khí MQ135 thường được dùng trong các thiết bị kiểm tra chất lượng không khí bên trong cao ốc, văn phòng, thích hợp để phát hiện khí NH<sub>3</sub>, NOx, Ancol, Benzen, khói và khí CO<sub>2</sub>. Cảm biến này với độ nhạy cao và thời gian đáp ứng nhanh. Tín hiệu ngõ ra dạng analog và digital. Cảm biến này có thể hoạt động ở nhiệt độ từ khoảng: -10°C đến 50°C và tiêu thụ dòng khoảng 300mA tại 5V.

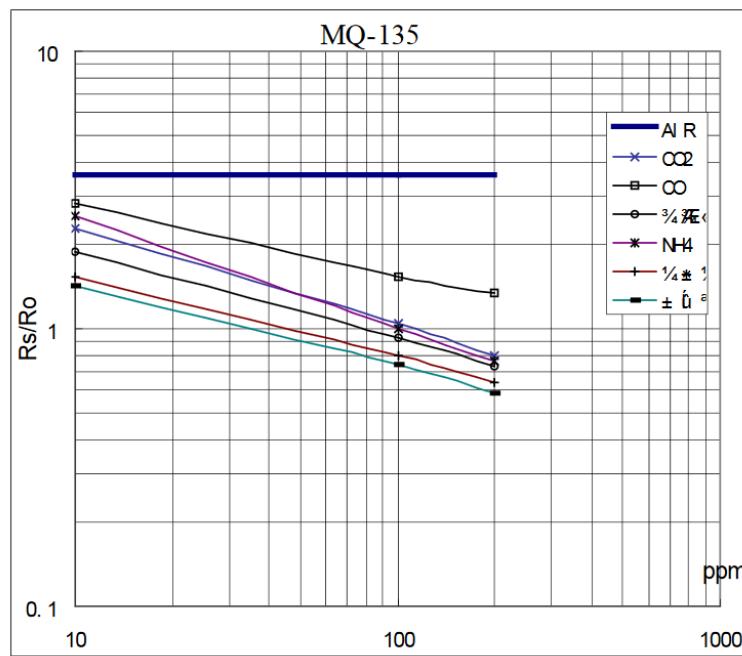


Hình 3.5 Cảm biến MQ135

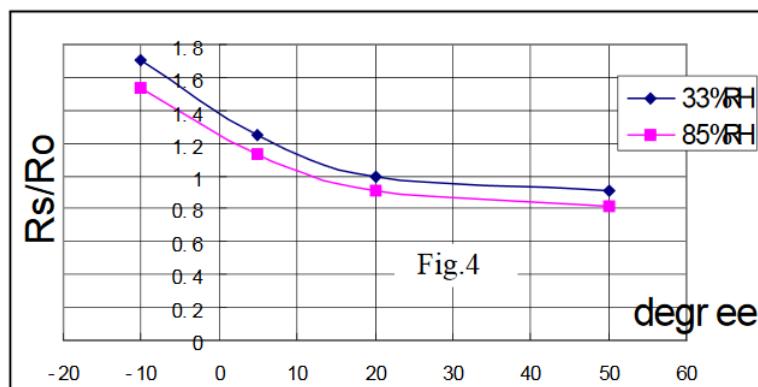
Thông số kỹ thuật:

- Điện áp cung cấp: <=24 VDC
- Điện áp heater: 5V AC/DC
- Sử dụng chip so sánh LM393c.
- Hai tín hiệu đầu ra (digital và analog)

- Tín hiệu analog từ 0 – 5V.
- Dải phát hiện từ 10 đến 1000ppm



Hình 3.6 Giản đồ chỉ sự biến đổi của  $Rs/Ro$  và giá trị ppm của MQ135



Hình 3.7 Giản đồ chỉ sự biến đổi của  $Rs/Ro$  đối với nhiệt độ, độ ẩm

Giá trị chất lượng không khí(ppm) được xác định theo biểu đồ Hình 3.6 giá trị này thay đổi theo sự biến đổi của giá trị  $Rs/Ro$ , theo Hình 3.7 thì giá trị  $Rs/Ro$  bị ảnh hưởng bởi nhiệt độ và độ ẩm.

Để xác định được giá trị Rs/Ro từ ngõ ra analog của mạch cảm biến MQ-135 ta có công thức như sau:

$$\frac{R_S}{R_L} = \frac{V_C - V_{RL}}{V_{RL}}$$

Giá trị Rs/Ro này được vi xử lý tính toán dựa trên đầu ra analog của cảm biến MQ-135 và được đưa vào thư viện mq135.h để có thể lấy được giá trị chất lượng không khí ppm theo sự thay đổi của nhiệt độ và độ ẩm của môi trường thực tế.

Thư viện mq135.h cung cấp cho chúng ta các hàm toán học đã được nhà sản xuất tính toán sẵn để có thể lấy được giá trị chất lượng không khí ppm dựa vào thay đổi của môi trường nhiệt độ và độ ẩm.

### Cảm biến chất lượng không khí MQ07

Cảm biến khí CO MQ-7 có thể pháp hiện khí CO tập trung ở những nơi khác nhau từ 10 đến 1000ppm. Cảm biến này với độ nhạy cao và thời gian đáp ứng nhanh. Tín hiệu ngõ ra dạng analog và digital. Cảm biến này có thể hoạt động ở nhiệt độ từ khoảng -10C đến 50C và tiêu thụ dòng khoảng 250mA tại 5V.

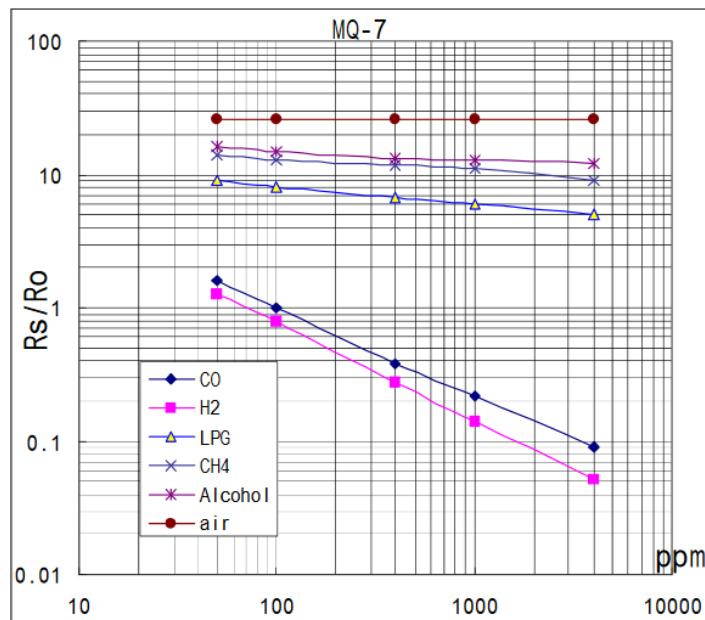


Hình 3.8 Cảm biến MQ07

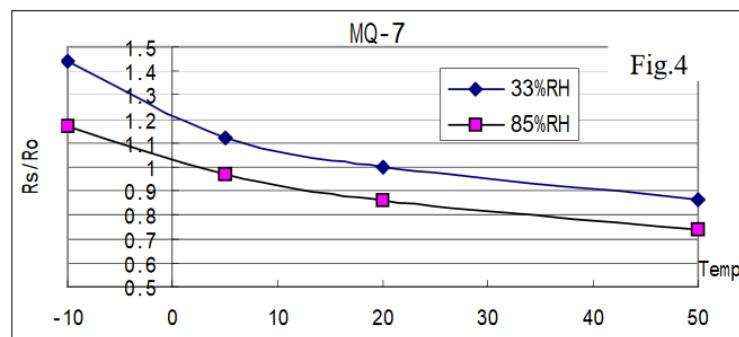
Thông số kỹ thuật:

- Điện áp cung cấp: 3 – 5VDC.
- Sử dụng chip so sánh LM393 và MQ-7.
- Hai tín hiệu đầu ra (digital và analog)
- Tín hiệu analog từ 0 – 5V.

- Dải phát hiện từ 10 đến 1000ppm
- Kích thước: 33 x 20 x 16mm.



Hình 3.9 Giản đồ chỉ sự biến đổi của  $Rs/Ro$  và giá trị ppm của MQ07



Hình 3.10 Giản đồ chỉ sự biến đổi của  $Rs/Ro$  đối với nhiệt độ, độ ẩm

Giá trị chất lượng không khí(ppm) được xác định theo biểu đồ Hình 3.9 giá trị này thay đổi theo sự biến đổi của giá trị  $Rs/Ro$ , theo Hình 3.10 thì giá trị  $Rs/Ro$  bị ảnh hưởng bởi nhiệt độ và độ ẩm.

Để xác định được giá trị Rs/Ro từ ngõ ra analog của mạch cảm biến MQ-7 ta có công thức như sau:

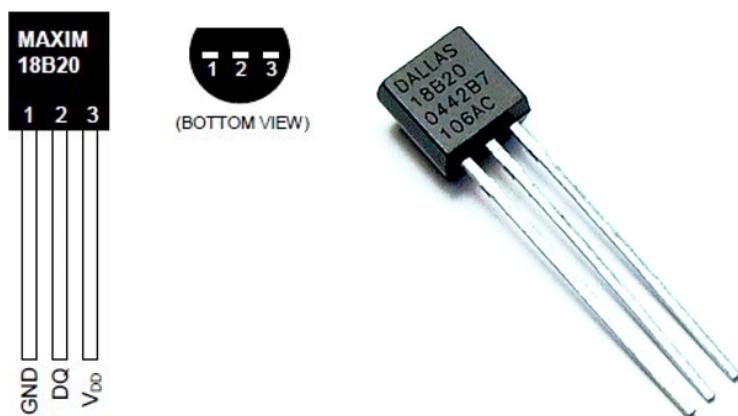
$$\frac{R_S}{R_L} = \frac{V_C - V_{RL}}{V_{RL}}$$

Giá trị Rs/Ro này được vi xử lý tính toán dựa trên đầu ra analog của cảm biến MQ-7 và được đưa vào thư viện mq07.h để có thể lấy được giá trị chất lượng không khí ppm theo sự thay đổi của nhiệt độ và độ ẩm của môi trường thực tế.

Thư viện mq07.h cung cấp cho chúng ta các hàm toán học đã được nhà sản xuất tính toán sẵn để có thể lấy được giá trị chất lượng không khí ppm dựa vào thay đổi của môi trường nhiệt độ và độ ẩm.

### Cảm biến nhiệt độ DS18B20 IC

Cảm biến nhiệt độ DS18B20 là cảm biến loại tín hiệu đầu ra digital đo nhiệt độ mới của hãng MAXIM với độ phân giải cao (12bit). IC được sử dụng giao tiếp 1 dây rất gọn gàng, dễ lập trình và giao tiếp nhiều DS18B20 trên cùng 1 dây. IC còn có chức năng cảnh báo nhiệt độ khi vượt ngưỡng và đặc biệt hơn là có thể cấp nguồn từ chân data.

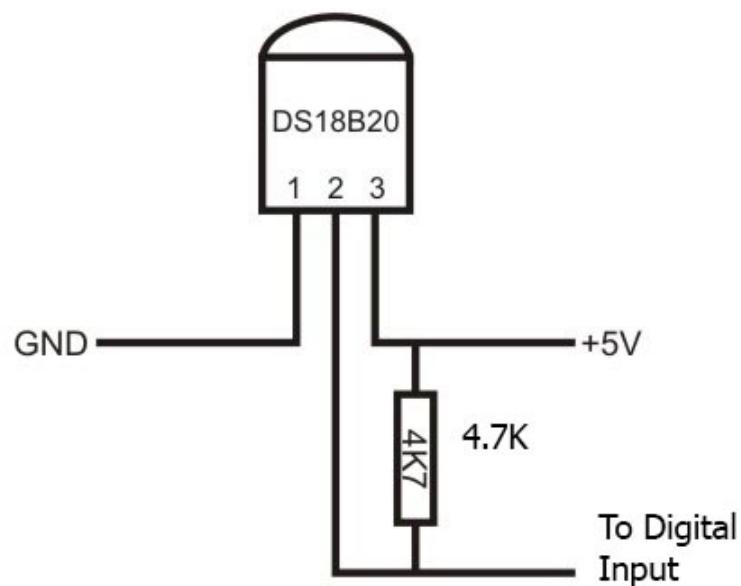


Hình 3.11 Cảm biến nhiệt độ DS18B20

Thông số kỹ thuật:

- Nguồn: 3-5.5V
- Dải đo nhiệt độ: -55 -125 \*C
- Sai số: +-0.5 \*C
- Độ phân giải: 9-12bits
- Thời gian chuyển đổi nhiệt độ: 750ms

Sơ đồ kết nối IC DS18B20 để lấy được tín hiệu data

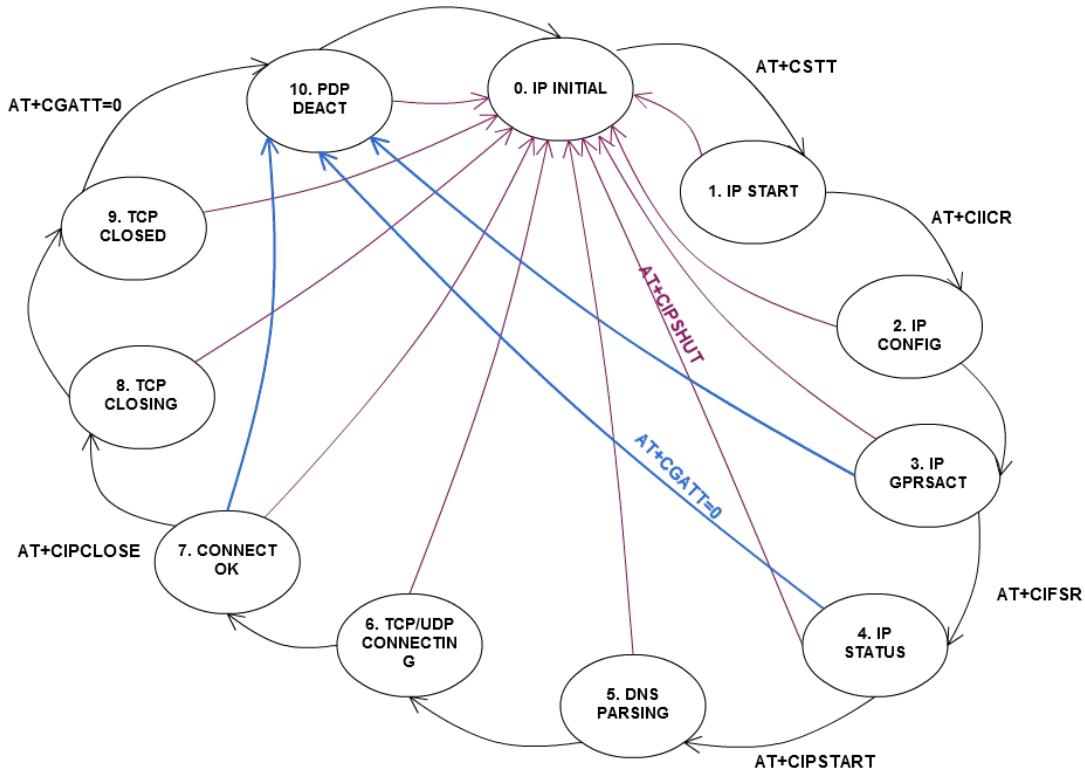


Hình 3.12 Sơ đồ kết nối DS18B20

Chân Digital input sẽ được đưa tới chân Digital input của board arduino nano, hiện tại nhà sản xuất cũng cung cấp cho lập trình viên về bộ thư viện để có thể sử dụng trong việc phát triển.

### 3.1.3 Chuẩn giao tiếp giữa các node tới máy chủ

Để các node cảm biến gửi dữ liệu lên máy chủ thì chúng tôi đã sử dụng module SIM800L được giới thiệu trong mục 2.5.2. Quá trình kết nối của các node cảm biến tới máy chủ thông qua giao thức TCP được thực hiện dựa trên mô hình máy trạng thái như sau:



Hình 3.13 Sơ đồ trạng thái GPRS cho đơn kết nối

Để gửi được dữ liệu đến máy chủ cần phải thiết lập được kết nối TCP từ module SIM800 đến máy chủ thông qua 11 trạng thái như Hình 3.13, các trạng thái được chuyển đổi thông qua các tập lệnh AT đã được đề cập tại mục 2.5.2. Đến trạng thái 7 (CONNECT OK) chúng ta đã giữ được kết nối TCP tới máy chủ và tại trạng thái này dùng lệnh AT+CIPSEND để thực hiện việc gửi gói dữ liệu lên cho máy chủ.

Gói dữ liệu được gửi theo phương thức GET, bên máy chủ phải hỗ trợ được phương thức GET này và định nghĩa rõ ràng các kiểu và số lượng dữ liệu gửi lên.

Dữ liệu sensor node gửi lên cho máy chủ gồm các dữ liệu sau:

NODE ID	Giá trị CO (ppm)	Giá trị nhiệt độ (Độ C)	Giá trị độ bụi (ppm)	Giá trị chất lượng không khí (ppm)	Giá trị dung lượng pin (%)
---------	------------------	-------------------------	----------------------	------------------------------------	----------------------------

### 3.1.4 Thiết kế API

Mục tiêu đề tài không chỉ thu thập và thông kê mà còn chia sẻ dữ liệu. Để làm được điều đó thì cần phải có API cung cấp cho những nhà phát triển thứ 3, có thể dùng API được cung cấp để sử dụng dữ liệu để phát triển. Bởi vì vậy hệ thống có xây dựng hệ thống API mở.

Ở bảng 3.1, hệ thống cung cấp các API về quản lý và truy vấn dữ liệu của các sensor node. Sử dụng hai method POST và GET để gửi request tới Server.

Bảng 3.1 Bảng API tương tác với dữ liệu về các node

Method	URL	Miêu tả	Yêu cầu xác thực người dùng
POST	/node/initnew	Khởi tạo node mới	Có
POST	/node/updatenode	Cập nhật thông tin node	Có
POST	/node/replacenode	Thay thế node	Có
GET	/node/pushdata	Push dữ liệu	Không
GET	/node/getdata	Get dữ liệu thu thập của node	Không
GET	/node/getinfo	Get thông tin của node	Không

Giao thức được lựa chọn sử dụng trong việc push data là giao thức HTTP mà không phải các giao thức thuần để phát triển IoT như MQTT hay CoAp vì tính phổ biến và dễ thiết lập sử dụng. Chỉ cần thiết lập chuỗi URL có đính kèm thông tin là có thể hòa nhập với hệ thống mà không cần thiết lập gì thêm, có thể ví như là "plug and play" vì tính đơn giản và hoạt động với mọi thiết bị có hỗ trợ kết nối Internet.

Bù lại, so với các giao thức tối ưu dung lượng gói tin cho phát triển IoT như MQTT, gói tin gửi qua HTTP có dung lượng nhiều hơn đáng kể nhưng với tần số hoạt động

truyền dữ liệu của đề tài thì vẫn có thể chấp nhận được và không tạo ra sự khác biệt lớn khi hoạt động.

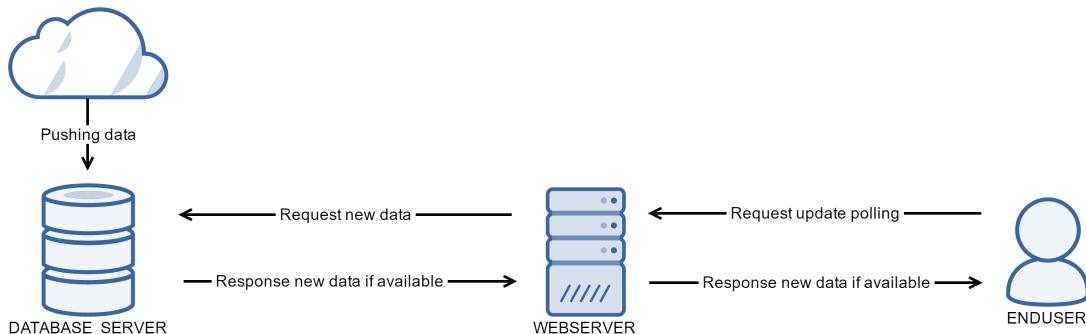
Về phần API trợ giúp đăng ký và xác thực người dùng được quy ước như bảng 3.2.

Bảng 3.2 Bảng API tương tác với dữ liệu về các node

Method	URL	Miêu tả
POST	/auth/login	Đăng nhập
GET	/auth/logout	Đăng xuất
POST	/user/register	Đăng ký người dùng mới

### 3.1.5 Cách thức tương tác cập nhật dữ liệu

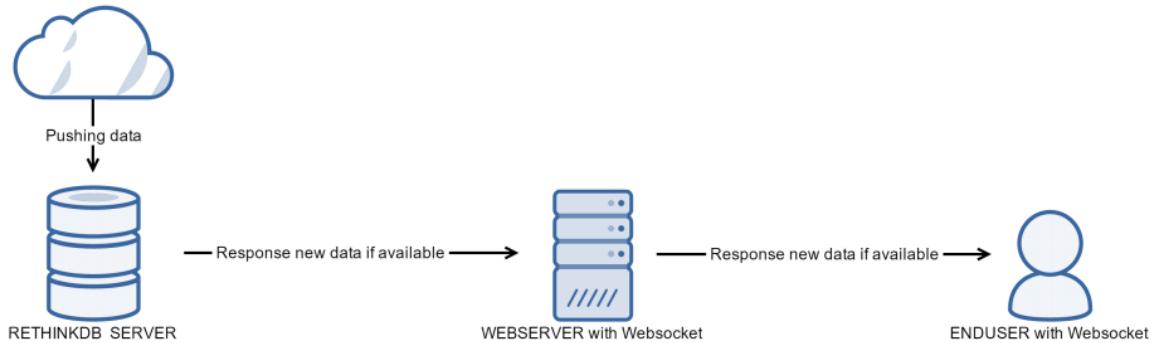
Trong việc tương tác cập nhật dữ liệu, hiện nay mô hình truy thường được sử dụng là **polling**. Client và server phải liên tục gửi các request liên tục cách nhau trong một thời gian cố định để kiểm tra liệu có sự thay đổi cập nhật dữ liệu.



Hình 3.14 Mô hình tương tác polling

Phương pháp này dễ hiện thực và áp dụng, tuy nhiên sẽ mắc nhược điểm là tồn nhiều gói tin request vô dụng làm giảm hiệu năng hệ thống, và không đáp ứng tính realtime nếu như chu kì gửi request dài.

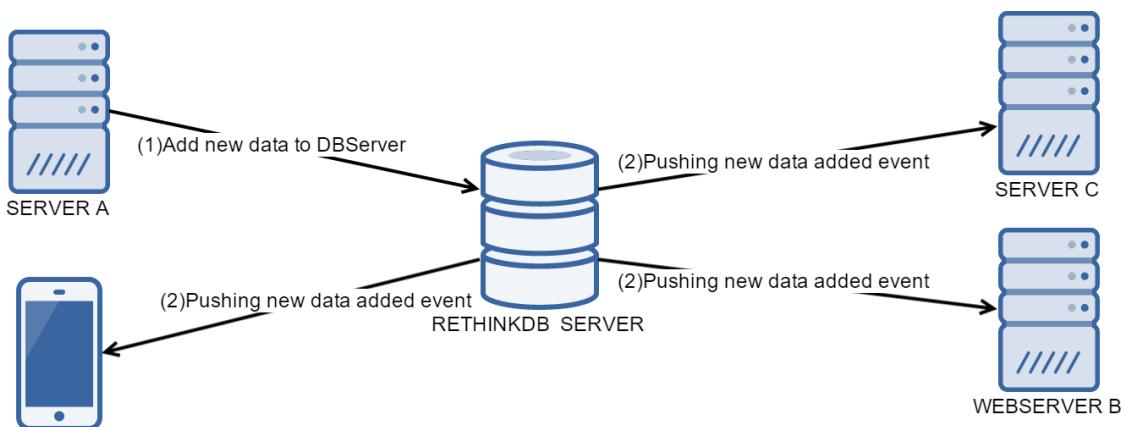
Vì hiện tại nhóm chúng tôi đang hướng tới hướng phát triển IoT, đòi hỏi yêu cầu về tính realtime cũng như tối ưu hóa hệ thống. Và để đáp ứng yêu cầu đó, nhóm đã tìm tới giải pháp giữ socket kết nối từ **Database** (RethinkDB) <-> **Server** (Chạy trên nền tảng Nodejs) <-> **Client** (web browser)



Hình 3.15 Mô hình tương tác realtime dựa trên Socket

Như hình 3.15, mô hình hoạt động đơn giản hơn rất nhiều, giảm bớt số lượng request không đáng có và đảm bảo tính realtime cho hệ thống.

Một điểm mạnh ở mô hình này nữa là có thể phát triển nhiều server hoặc ứng dụng di động sử dụng chung một hệ quản lý cơ sở dữ liệu nhưng vẫn đảm bảo sự đồng bộ dữ liệu cập nhật tới enduser theo thời gian thực.



Hình 3.16 Mô hình tương tác cập nhật nhiều Server và Client

### 3.1.6 Mô hình ứng dụng trình bày dữ liệu

#### Chức năng Web Server

Đối với người dùng:

- Cung cấp cho người dùng thông tin về nơi chứa node cảm biến.
- Người dùng có thể xem biểu đồ dữ liệu thông số môi trường tại vị trí tương ứng với node cảm biến đã được thiết lập.
- Xem được tình trạng và biểu đồ thống kê của các node cảm biến đang hoạt động.
- Gửi phản thông tin phản hồi về cho người quản lý thông qua email.
- Cung cấp thông tin lịch sử các hoạt động sử dụng API, giúp cho bên thứ 3 có thể dễ dàng phát triển.

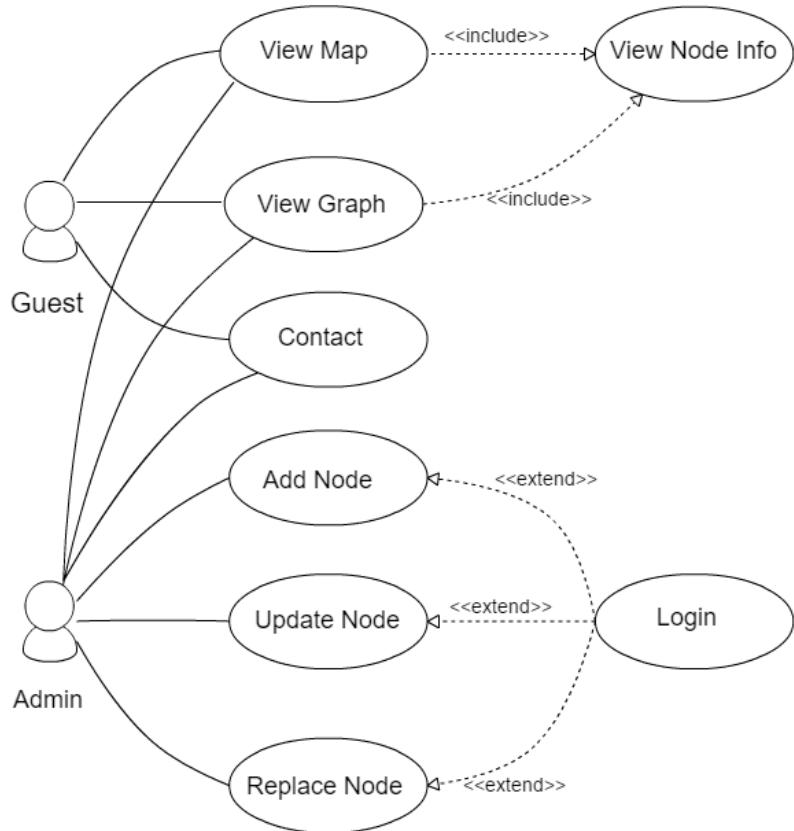
Đối với người quản lý:

- Bao gồm tất cả chức năng của người dùng bình thường.
- Quản lý chỉnh sửa thông tin các node cảm biến: thêm, xóa, cập nhật, thay đổi.

Bảng 3.3 Bảng mô tả giản đồ Usecase của Web Server

STT	Tên	Miêu tả
1	View map	Người dùng thấy được các node đang chạy trên google maps
2	View Node Info	Thông tin tương ứng của Node( Lat, Lng, Phone)
3	View Graph	Đồ thị hoạt động của Node(các số liệu đo được theo ngày)
4	Contact	Phản hồi ý kiến người dùng qua Gmail
5	Add Node	Thêm mới Node vào hoạt động
6	Update Node	Thay đổi thông tin của Node( Lat, Lng, Phone)
7	Replace Node	Thay đổi 1 Node xảy ra sự cố bằng 1 Node khác

Biểu đồ High level Usecase của Web Server



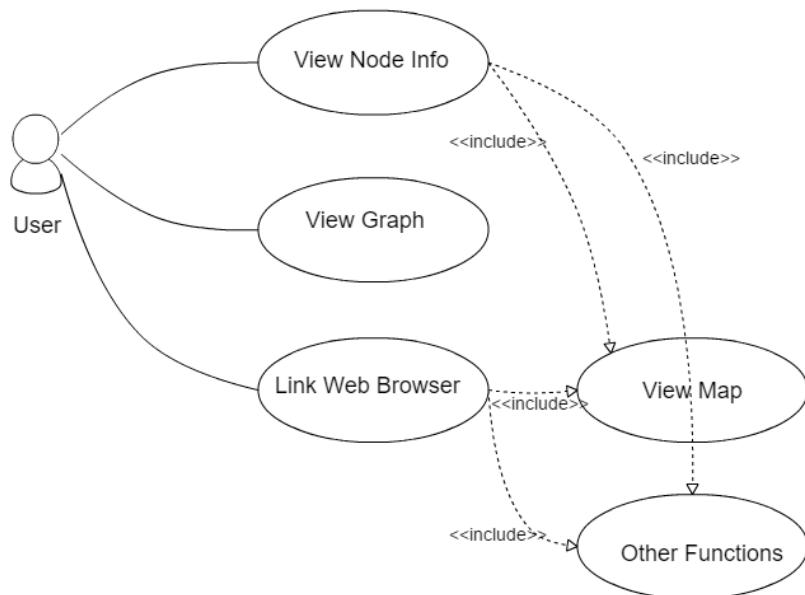
Hình 3.17 Biểu đồ High level Usecase

Mô tả giản đồ Usecase

### Chức năng Ứng dụng di động

Hiện thị thông tin các node cảm biến đang hoạt động qua màn hình chính, và biểu đồ dữ liệu của từng node cảm biến theo từng ngày, thêm sự kết nối với web browser tạo sự thuận tiện cho việc theo dõi, hiện thị đồ thị (2 kiểu đồ thị là line chart và bar chart).

### Biểu đồ High level Usecase của ứng dụng di động



Hình 3.18 Giản đồ High level Usecase của ứng dụng di động

Mô tả Usecase

Bảng 3.4 Bảng mô tả giản đồ Usecase của ứng dụng di động

STT	Tên Use-case	Mô tả
1	View Node Info	Thông tin của Node( Lat, lng, Phone, ID)
2	View Graph	2 dạng đồ thị theo ngày (line chart, bar chart của các thông số)
3	Link web browser	Liên kết tới web browser
4	View Map	Hiện thị Google Map và vị trí các Node
5	Other Functions	Các chức năng của web browser

### 3.1.7 Các ràng buộc của hệ thống

Cũng như các dự án IoT quy mô rộng khác, để tài cũng yêu cầu nhiều đặc tính ràng buộc như:

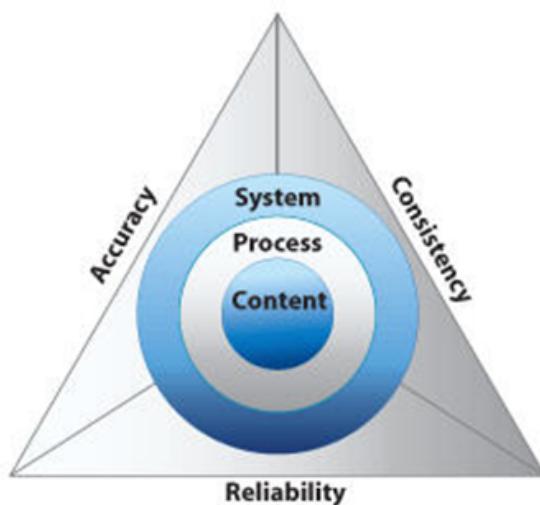
#### Tính đáp ứng thời gian thực

Để giải quyết tình trạng giao thông hiện thời và trong thời gian gần, ta cần phải có dữ liệu thời gian thực. Ta không thể sử dụng dữ liệu của nửa tiếng hoặc 1 giờ trước để vẽ nên bản đồ lưu thông hiện tại cũng như cách điều hướng giải quyết. Do đó, hệ thống cần phải có thời gian đáp ứng nhanh khi có yêu cầu dữ liệu khi được gọi. Để đạt được kết quả đó, ta cần phải có những phần cứng với thiết lập kết nối có khả năng đáp ứng nhanh, cùng với khả năng hoạt động ổn định liên tục trong thời gian dài để cung cấp dữ liệu liên tục và liền mạch.

#### Toàn vẹn dữ liệu (Data Integrity)

Mục đích chính của dự án là thu thập dữ liệu và từ đó phát triển ứng dụng. Vì thế tính toàn vẹn dữ liệu đóng vai trò quan trọng. Dữ liệu thu thập cần phải đáp ứng đủ các yếu tố: tin cậy, chính xác và đầy đủ. Ta cần phải có đủ dữ liệu thì mới có thể đủ dữ kiện để giải quyết bài toán. Ta không thể giải quyết khi chỉ có được dữ liệu một

đoạn đường, 1 thời gian ngắn mà đòi hỏi phải có dữ liệu của một khu vực đủ lớn và tương quan với nhau. Bên cạnh đó cần phải có tính chính xác dữ liệu và đòi hỏi sự ổn định của lượng dữ liệu đó qua yếu tố độ tin cậy.



Hình 3.19 Toàn vẹn dữ liệu

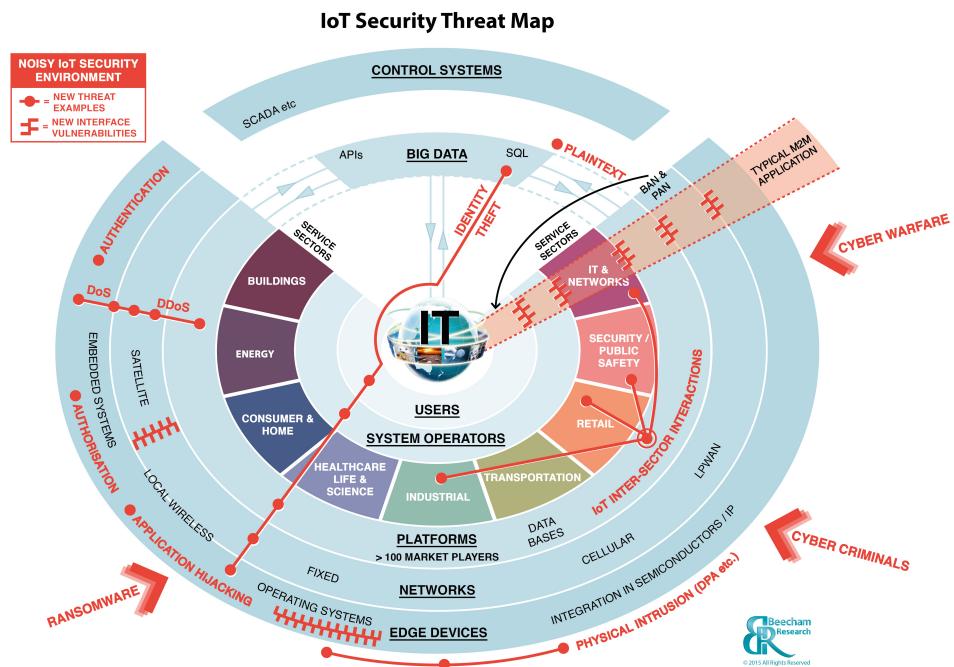
## Đồng bộ hóa

Hệ thống giám sát môi trường có nhiều thiết bị khác nhau, nhiều phương pháp và cách thức truyền dữ liệu khác nhau. Do đó ta cần phải có sự chuẩn hóa các giao thức giao tiếp, đồng bộ các gói dữ liệu. Tuy nhiên điều này sẽ dẫn đến vấn nạn được đề cập ở mục sau.

## Tính bảo mật

Vấn đề bảo mật và an toàn dữ liệu hiện tại vẫn là một trong những khó khăn mà hệ thống IoT đang mắc phải. Vì hệ thống IoT đòi hỏi cần phải có giao thức kết nối giữa các thiết bị phải được chuẩn hóa và đồng bộ, cũng như tối giản kích thước gói tin để tối ưu trong việc truyền dữ liệu giữa các thiết bị, do đó gói tin truyền đi có thiết lập

cấu trúc đơn giản và hệ thống dễ bị thâm nhập. Điều này sẽ dẫn đến hệ thống có thể bị đánh cắp dữ liệu, và thậm chí mất quyền kiểm soát toàn bộ hệ thống bởi vì tất cả đều được kết nối với nhau. Vậy nên cần phải cân nhắc trade-off giữa hiệu năng hệ thống và tính bảo mật.



Hình 3.20 Tính bảo mật

Các mối nguy hại chính ảnh hưởng trực tiếp tới hệ thống IoT như: DDoS, Ransomware, Cyber Criminals và Cyber Warfare.

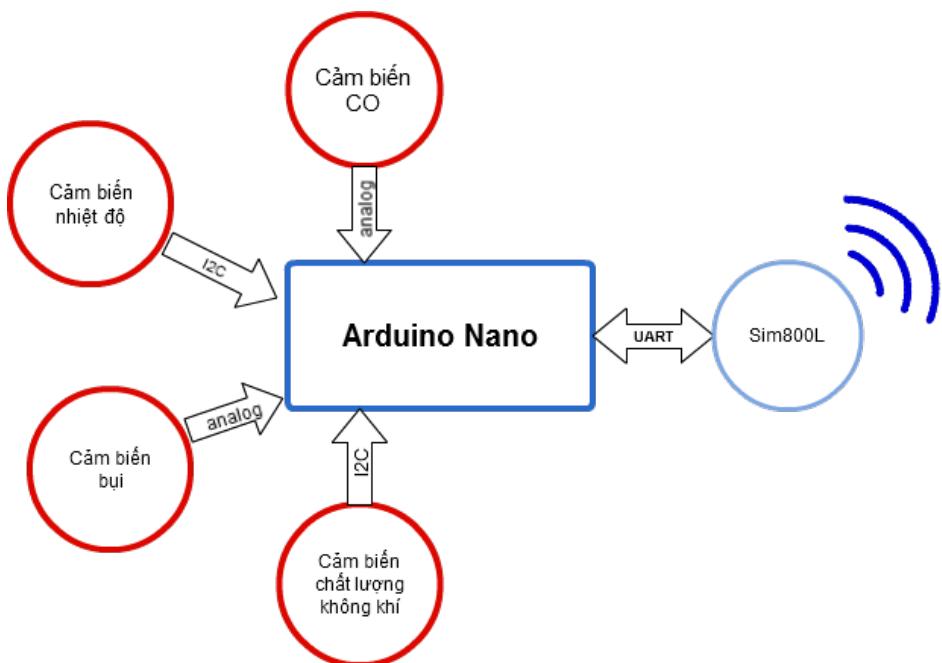
## 3.2 Hiện thực node cảm biến

### 3.2.1 Hiện thực prototype node cảm biến

Chức năng chính của node cảm biến là dùng để lấy các dữ liệu môi trường như giá trị nồng độ CO, chất lượng không khí, nhiệt độ và lượng bụi, trong quá trình tìm các loại cảm biến trên thị trường hiện có thì nhóm chúng tôi đã đề xuất sử dụng các cảm biến được đề cập tại mục 3.1.2 như sau:

- Cảm biến GP2: dùng để đo lượng bụi có trong không khí.
- Cảm biến MQ135: dùng để đo chất lượng của môi trường không khí.
- Cảm biến MQ07: dùng để đo nồng độ khí CO trong không khí.
- Cảm biến DS18B20: dùng để đo nhiệt độ môi trường xung quanh.

Như vậy, các thành phần chính của prototype biến đầu tiên chúng tôi đã xây dựng như sau:



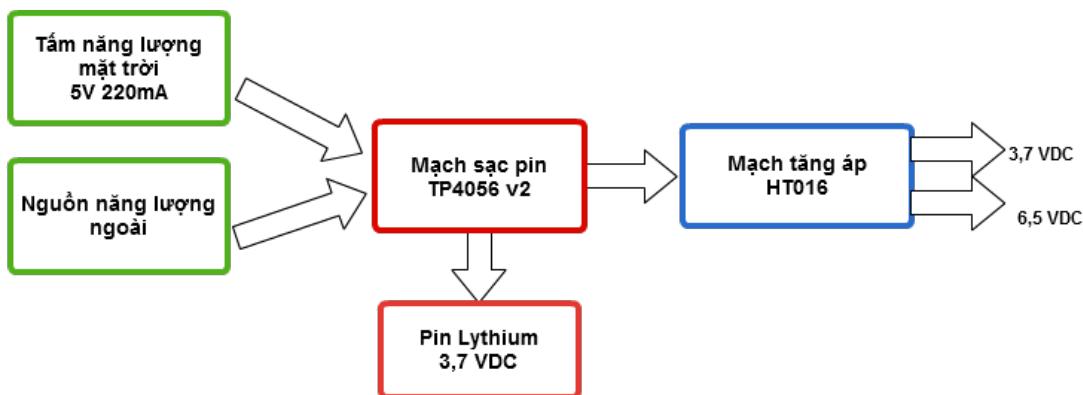
Hình 3.21 Mô hình prototype đầu tiên

Ngoài những cảm biến đã đề cập thì sơ đồ hoạt động của các khối chức năng trong Hình 3.21 còn có thêm những khối chính sau:

- Khối Arduino Nano: mạch xử lý trung tâm có nhiệm vụ định thời và điều khiển các khối cảm biến khác để thu thập dữ liệu cảm biến sau đó sẽ xử lý các số liệu thô sang số liệu chính xác, cuối cùng gửi dữ liệu đã được xử lý lên cho máy chủ thông qua khối Sim800L.
- Khối Sim800L: chức năng mở kết nối TCP tới máy chủ thông qua dịch vụ GPRS của nhà mạng cung cấp và đưa các dữ liệu từ mạch vi xử lý gửi thông qua giao thức UART.

### Mô hình khối nguồn cho node cảm biến

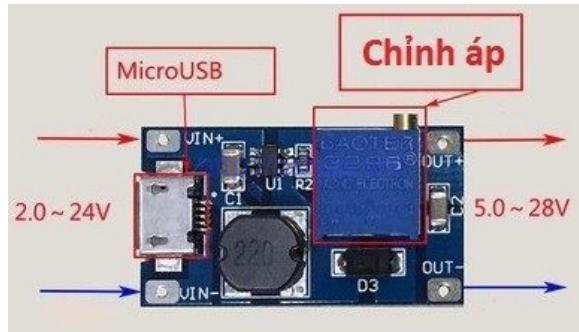
Tiếp đến phần thiết kế thì để có thể tận dụng nguồn năng lượng mặt trời để nạp pin cho mạch hoạt động độc lập với điện lưới thì chúng tôi đã đề xuất mô hình khối nguồn như sau:



Hình 3.22 Mô hình khối nguồn

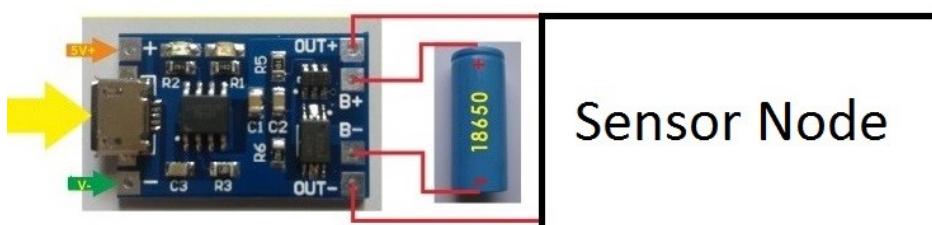
- Mạch tăng áp mini HT106 là module tăng áp cực kì nhỏ gọn, ngõ vào là cổng microUSB. Mạch có khả năng tăng đến 28v tối đa. Dưới đây là một số thông số kỹ thuật:
  - Điện áp vào: 2-24V

- Điện áp ra: 5-28V
- Dòng tối đa: 2A đỉnh, dòng liên tục khoảng 1A.
- Công suất: 6W
- Hiệu suất: 93



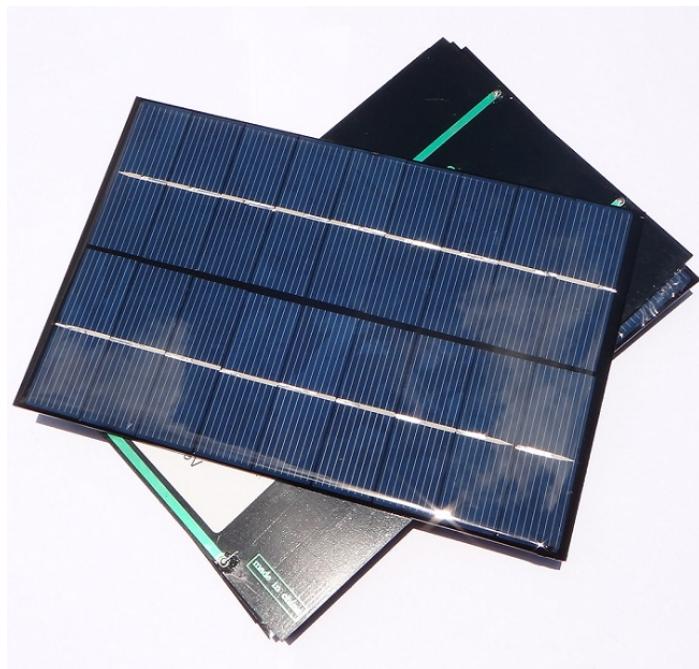
Hình 3.23 Mạch tăng áp mini HT106

- Mạch sạc pin Lithium TP4056v2 sử dụng IC quản lý sạc TP4056 nhưng được nâng cấp để có thể ngắt khi pin yếu dưới 2.4V nhằm bảo vệ pin. Một số thông số kỹ thuật của mạch sạc pin Lithium TP4056v2
  - Điện áp đầu vào: 5VDC mini USB
  - Điện áp ngưỡng ra tự động ngắt: 4.2V +- 1- Dòng sạc tối đa: 1000mA
  - Điện áp ngưỡng cần sạc là 2.5V
  - Tích hợp tự động ngắt bảo vệ pin khi pin yếu dưới 2.4 V qua 2 chân OUT+ và OUT-



Hình 3.24 Mạch sạc pin Lithium TP4056v2

- Pin năng lượng mặt trời Poly 5.5V /220 mA
  - VỚI KÍCH THƯỚC: 100x80x2mm
  - Nguồn đầu ra: 5.5V
  - Dòng tối đa cung cấp được: 220mA



Hình 3.25 Pin năng lượng mặt trời Poly 5V /220 mA

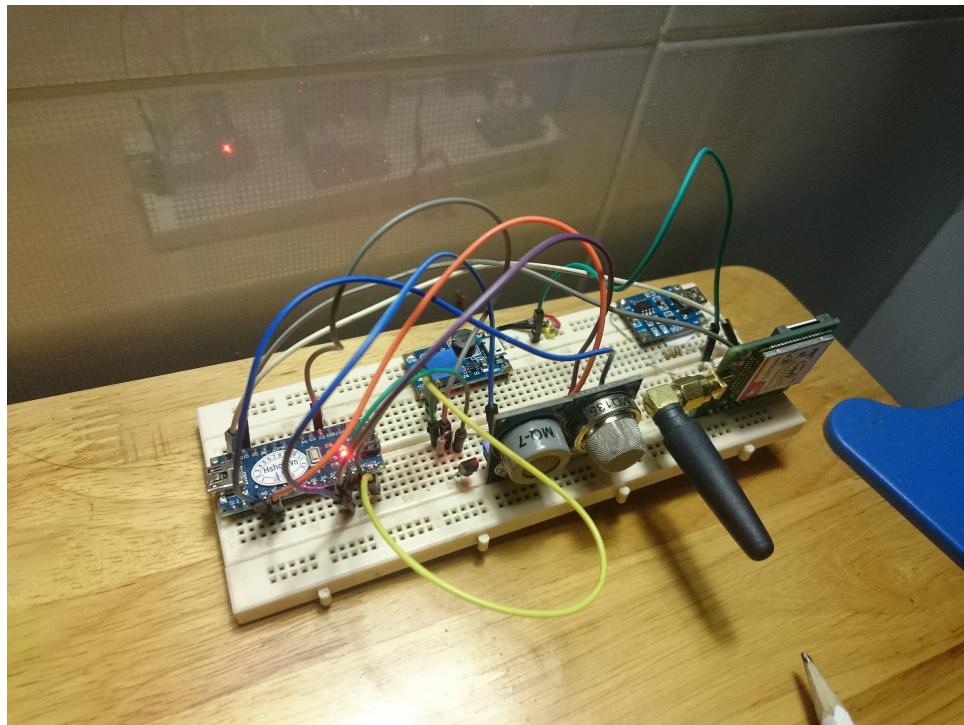
- Nguồn năng lượng ngoài: sử dụng trong một số trường hợp thời tiết mưa nhiều hoặc gió nhiều mà thiếu nguồn năng lượng mặt trời, chúng ta có thể tận dụng nguồn mưa và gió nhờ các tuabin phát năng lượng nhờ vào sức gió và mưa.
- Nguồn đầu ra 3.7VDC dùng để cấp cho module Sim800L còn đối với nguồn 6.5VDC dùng để cung cấp cho mạch vi xử lý.

### Các thư viện đã sử dụng

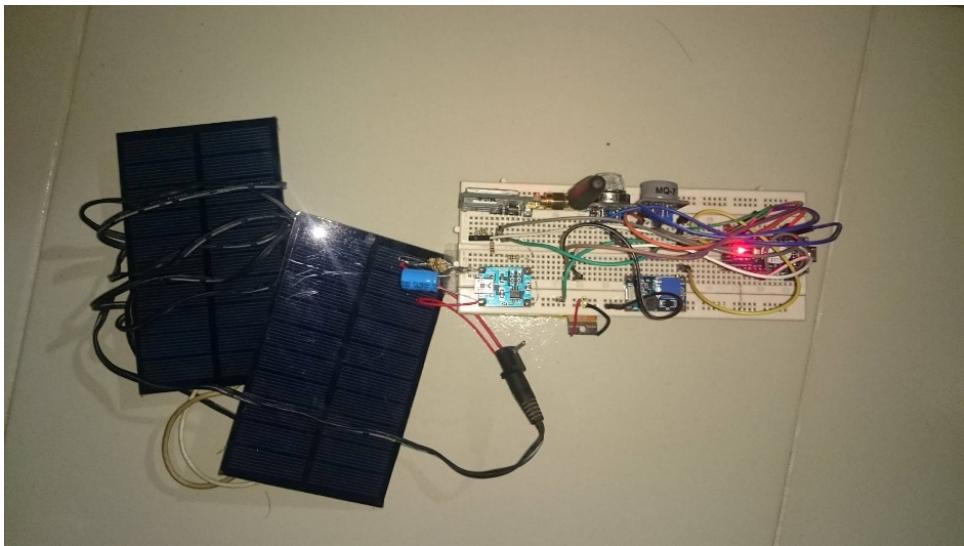
```
1 #include "MQ135.h" //Thu vien cac ham toan hoc de xu ly so lieu MQ135
```

```
2 #include "MQ07.h" // Thu vien cac ham toan hoc de xu ly so lieu MQ07  
3 #include "OneWire.h" // Thu vien cau hinh giao thuc I2C  
4 #include "DallasTemperature.h" // Thu vien cho cam bien nhiet do  
5 #include "SoftwareSerial.h" // Thu vien mo rong cac chan UART  
6 #include "TimerOne.h" // Thu vien cau hinh timer
```

### Kết quả hiện thực



Hình 3.26 Kết quả prototype đầu tiên



Hình 3.27 Kết quả prototype đầu tiên 2

### Nhận xét

- Hoàn thành các chức năng có thể đọc được các dữ liệu từ cảm biến MQ135, MQ07, DS18B20, và GP2.
- Xử lý được dữ liệu thông qua các thư viện do nhà sản xuất cảm biến cung cấp.
- Gửi được dữ liệu lên máy chủ thông qua dịch vụ GPRS của Sim800L, hệ thống máy chủ của chúng tôi ban đầu chưa được hoàn thành nên nhờ đến công cụ phát triển IoT Thingspeak thì chúng tôi đã kiểm tra được, xem thêm tại: <https://thingspeak.com/channels/157120>

### 3.2.2 Hiện thực mô hình tổng thể node cảm biến

#### Hoàn thiện chức năng

Sau khi hoàn thành prototype đầu tiên node cảm biến thì chúng tôi đã bắt đầu lên kế hoạch nâng cấp chúng thêm những chức năng để có thể tối ưu tốt nhất có thể. Chức năng của node cảm biến bao gồm:

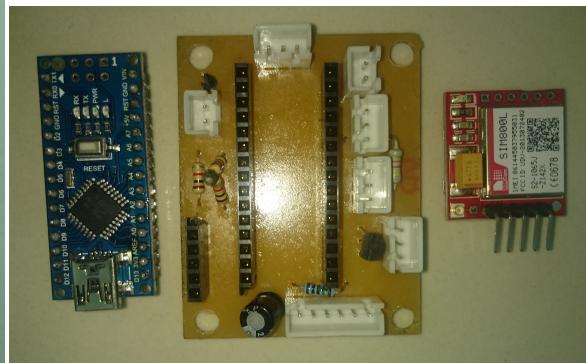
- 
- 
- 

### Hiện thực board mạch vi xử lý

Board mạch vi xử lý được thiết kế và hiện thực gọn nhất có thể và hỗ trợ các header nối các dây bus đến các cảm biến và module Sim800L. Chúng tôi đã thực hiện thiết kế board mạch vi xử lý như Hình 3.28



(a) Mô hình 1



(b) Mô hình 2

Hình 3.28 Bố trí board mạch vi xử lý

Một số khai báo được sử dụng như sau:

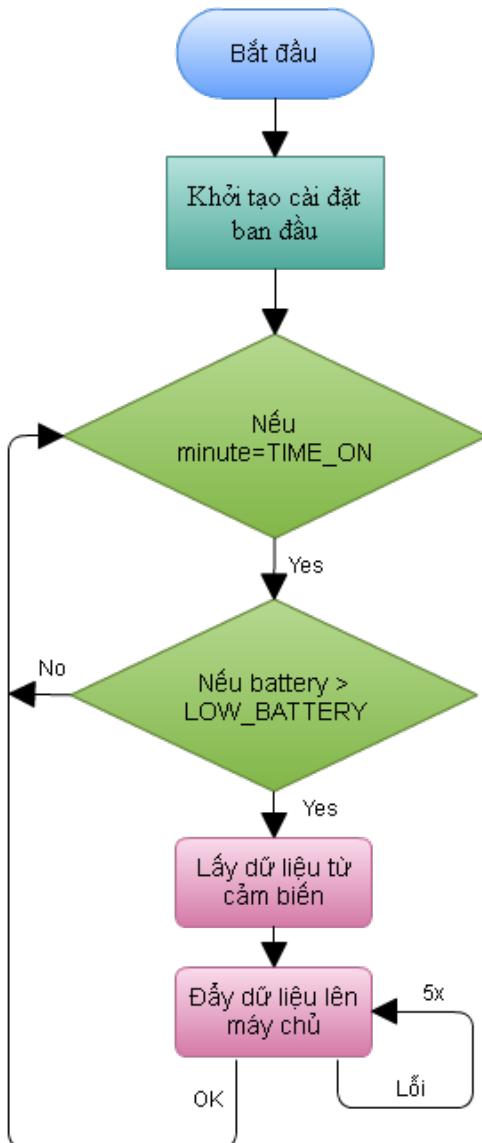
```

1 #define ONE_WIRE_BUS 4
2 #define MQ135_PIN A0
3 #define MQ7_PIN A1
4 #define RX_SIM_OUT 11
5 #define TX_SIM_OUT 10

```

```
6 #define DUST_OUT A2
7 #define DUST_PIN 2
8
9 // Gia tri % Pin thap se khong cho phep mach hoat dong
10#define LOW_BATTERY 30
11
12// So lan gui lai du lieu len may chu neu khong thanh cong
13#define TIMEFAIL_SENDSMS 5
14
15// Mo rong UART tren chan so 11 va 10
16SoftwareSerial sim900(RX_SIM_OUT, TX_SIM_OUT);
17
18int TIMEON = 3; //Thoi gian dot nong cam bien MQ135, MQ07
19
20//Thoi gian ngat nguon cam bien MQ135, MQ07 ban ngay
21int TIMEOFF_DAY = 13;
22//Thoi gian ngat nguon cam bien MQ135, MQ07 ban dem
23int TIMEOFF_NIGHT = 28;
24
25// Moc thoi gian xac dinh buoi sang va buoi toi
26int time_s1 = 6;
27int time_s2 = 19;
28
29String apiKey = "7"; //Node ID
```

Lập trình cho vi xử lý Arduino Nano được mô hình hóa theo lưu đồ xử lý dưới Hình 3.29:



Hình 3.29 Lưu đồ xử lý hàm main

- Khởi tạo cài đặt ban đầu gồm các công việc:
  - Thiết lập chế độ hoạt động của các chân trên vi xử lý.
  - Thiết lập tốc độ truyền UART.
  - Thiết lập timer định thời.
  - Khởi động Module Sim800L.
  - Đồng bộ thời gian module Sim800L với máy chủ The Network Time Proto-

col(NTP).

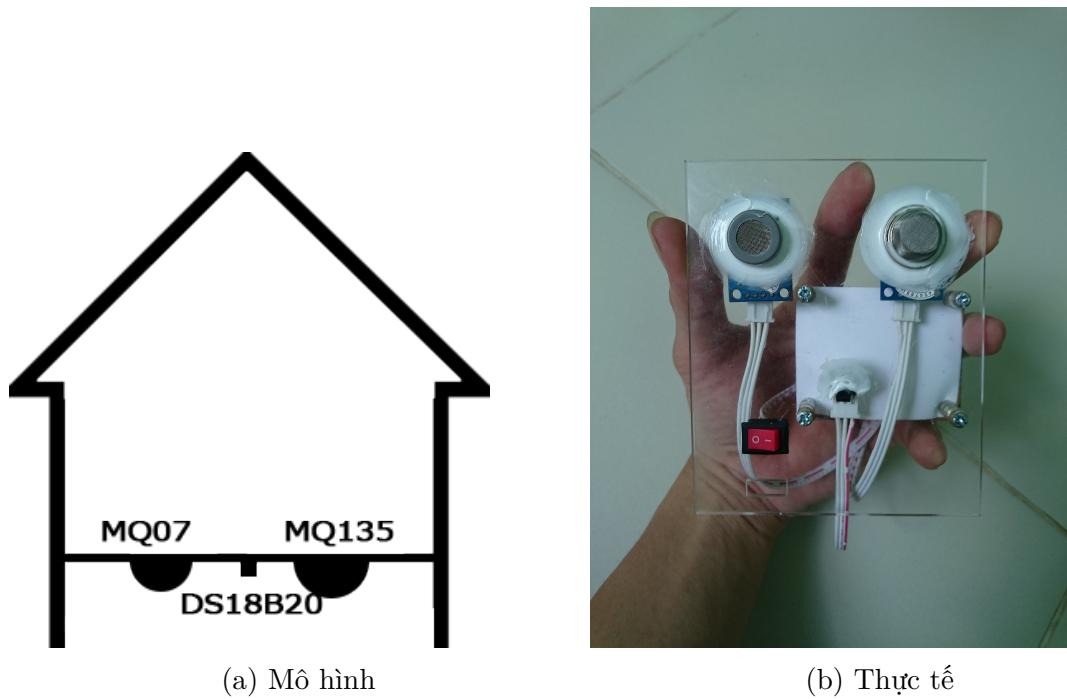
Vìệc Đồng bộ thời gian module Sim800L với máy chủ NTP được Sim800L hỗ trợ qua các tập lệnh sau:

```
1 void syncTime()
2 {
3     // Network time sync
4     sim900.println("AT+SAPBR=1,1");
5     delay(2000);
6     sim900.println("AT+CNTPCID=1");
7     delay(2000);
8     sim900.println("AT+CNTP=\\"128.4.24.98\\",28");
9     delay(2000);
10    sim900.println("AT+CNTP");
11    delay(4000);
12    sim900.println("AT+SAPBR=0,1");
13    delay(2000);
14    Serial.println("Sync time - Done");
15    updateTime();
16 }
```

### Bố trí các linh kiện trong hộp cảm biến

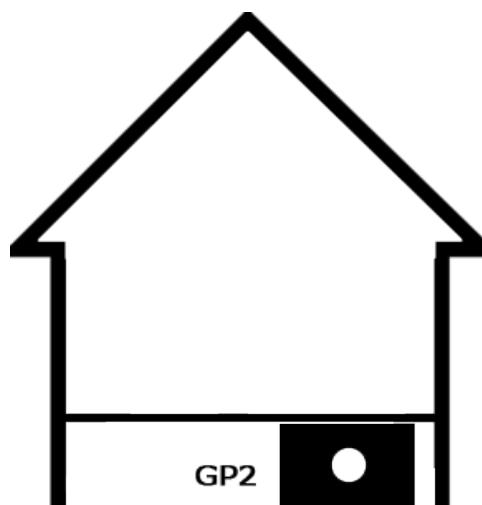
Vìệc bố trí các linh kiện trong hộp cảm biến nhằm giúp cho việc thu thập dữ liệu của các cảm biến được hiệu quả hơn và không bị ảnh hưởng bởi mưa. Nó cũng giúp cho việc phát triển trong tương lai nếu các cảm biến hoặc linh kiện bị hư thì dễ dàng thay thế hơn.

Đối với các cảm biến MQ135, MQ07 và cảm biến nhiệt DS18B20 thì được bố trí đặt ở lớp dưới của chiếc hộp như Hình 3.30



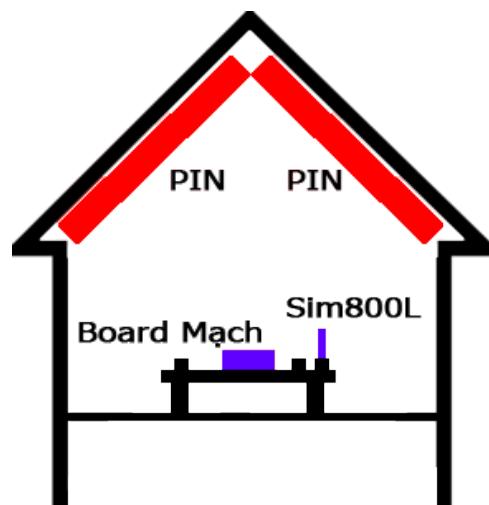
Hình 3.30 Bố trí các cảm biến

Đối với các cảm biến bụi GP2 chúng tôi đề xuất vị trí được đặt như Hình ?? sau:



Hình 3.31 Mô hình đặt cảm biến GP2

Còn các board mạch vi xử lý, module Sim800L và pin được đặt vị trí bên trong chiếc hộp như Hình 3.32 sau:



Hình 3.32 Mô hình bố trí các linh kiện khác

### 3.2.3 Mức độ tiêu hao năng lượng

Để biết được mức tiêu hao năng lượng của mạch hoạt động như thế nào thì chúng tôi đã bắt đầu tính toán từ những thông số tiêu thụ chuẩn của các linh kiện được sử dụng trong node cảm biến. Nhằm nắm bắt được và đưa ra dung lượng pin cần thiết cho các node cảm biến chạy mà không phụ thuộc vào mạng lưới điện.

Chú ý: các thông số tính toán dưới đây từ thông số chuẩn của nhà sản xuất kèm với một số thông số chuẩn của chúng tôi như sau:

- Thời gian cho việc đốt nóng cảm biến MQ135 và MQ07 trước khi lấy dữ liệu: 3 phút
- Ban ngày từ 7AM-8PM: cứ mỗi 15 phút gửi dữ liệu lên máy chủ.
- Ban đêm từ 8PM-7AM: cứ mỗi 30 phút gửi dữ liệu lên máy chủ.

Mức tiêu hao năng lượng vào 1 buổi sáng 7AM-8PM:

Bảng 3.5 Bảng tiêu thụ năng lượng buổi sáng

STT	Module	Dòng tiêu thụ (mA)	Giờ hoạt động (h)	Tổng dòng tiêu thụ(mA)	Công suất (mW)
1	Cảm biến bụi	20	13	260	858
2	Cảm biến MQ07	250	2.6	650	3250
3	Cảm biến MQ135	300	2.6	780	3900
4	Cảm biến nhiệt độ 18BS20	1.5	13	19.5	97.5
5	Module Sim800L (idea mode)	10	12.35	123.5	456.95
6	Module Sim800L (Hoạt động)	500	0.65	325	1202.5
7	Arduino nano	10	13	130	780
<b>Tổng</b>		<b>1091.5</b>		<b>2288</b>	<b>10544.95</b>

Mức tiêu hao năng lượng vào 1 buổi tối 8PM-7AM:

Bảng 3.6 Bảng tiêu thụ năng lượng buổi tối

STT	Module	Dòng tiêu thụ (mA)	Giờ hoạt động (h)	Tổng dòng tiêu thụ(mA)	Công suất (mW)
1	Cảm biến bụi	20	11	220	726
2	Cảm biến MQ07	250	2.2	550	2750
3	Cảm biến MQ135	300	2.2	660	3300
4	Cảm biến nhiệt độ 18BS20	1.5	11	16.5	82.5
5	Module Sim800L (idea mode)	10	10.45	104.5	386.65
6	Module Sim800L (Hoạt động)	500	0.55	275	1017.5
7	Arduino nano	10	11	110	660
<b>Tổng</b>		<b>1091.5</b>		<b>1936</b>	<b>8922.65</b>

### 3.3 Hệ thống Server lưu trữ dữ liệu và cung cấp API

#### 3.3.1 Cấu trúc tổ chức tập tin

##### Các thư mục và chức năng

Chúng ta có những thư mục được làm việc trực tiếp:

- **libs**: chứa các hàm function hỗ trợ về xác thực và log.
- **models**: thư mục này bao gồm các mô hình quản lý cơ sở dữ liệu.
- **routes**: điều khiển và điều hướng theo các url.
- **log**: chứa các file ghi lại log trong quá trình hoạt động.

- **views:** là bộ mặt của web server, giúp người dùng có thể tương tác được với server qua web browser, bao gồm các file giao diện như html, ejs...

### Các file khởi tạo và cấu hình Server

#### Các file khởi tạo này bao gồm:

- **emap.js:** file có chức năng đọc file cấu hình và khởi tạo kết nối server.

```

1 ...
2 server.on('listening', onListening); // open for listening
3 function onListening() {
4   var addr = server.address();
5   var bind = typeof addr === 'string' ?
6     'pipe' + addr :
7     'port' + addr.port;
8   debug('Listening on ' + bind);
9   console.log('Listening on ' + bind);
10 }
11 ...

```

Listing 3.1 emap.js

- **app.js:** là file điều khiển và thiết lập các chức năng, khai báo các modules chính được sử dụng. Bên cạnh đó, còn có chức năng gán điều hướng tới các file trong thư mục route

```

1 ...
2 // initial server
3 app.use(passport.initialize());
4 app.use(passport.session());
5 app.set('views', path.join(__dirname, 'views'));
6 app.engine('ejs', engine);
7 ...
8 // control route

```

```
9 app.use(logger('dev'));
10 app.use('/log', express.static(path.join(__dirname, 'log')));
11 app.use(express.static(path.join(__dirname, 'public')));
12 app.use('/log', serveIndex('./log'));
13 app.use('/', routes);
14 app.use('/user', user);
15 app.use('/node', node);
16 app.use('/auth', auth);
17 ...
```

Listing 3.2 app.js

Các file cấu hình này bao gồm:

- **package.json**: khai báo các thông tin cơ bản về project và các modules được sử dụng.

```
{
  "name": "emap-server",
  "version": "1.0.0",
  "description": "This is serverside part of IoT project",
  "main": "emap.js",
  "author": "Cuong, Tung and Ny",
  "license": "ISC",
  "homepage": "www.codingyourfuture.com",
  "dependencies": { // dependancies modules
    "angular-chart.js": "^1.0.3",
    "asyncawait": "^1.0.6",
    "babel": "^6.5.2",
    "ejs": "^2.5.2",
    ...
  }
}
```

```

    "rethinkdb": "^2.3.3",
    "serve-index": "^1.8.0",
    "socket.io": "^1.5.0",
}
}

```

- **config.json**: khai báo các thông số cấu hình được sử dụng. Tại đây ta cấu hình thông số kết nối tới RethinkDB và port ứng dụng server nodejs.

```

{
  "app": {
    "port": "8888"
  },
  "rethinkdb": {
    "host": "localhost",
    "port": "28015",
    "db": "emap",
    "address": "localhost:28015",
    "tableList": ["nodeData", "nodeList", "user"]
  }
}

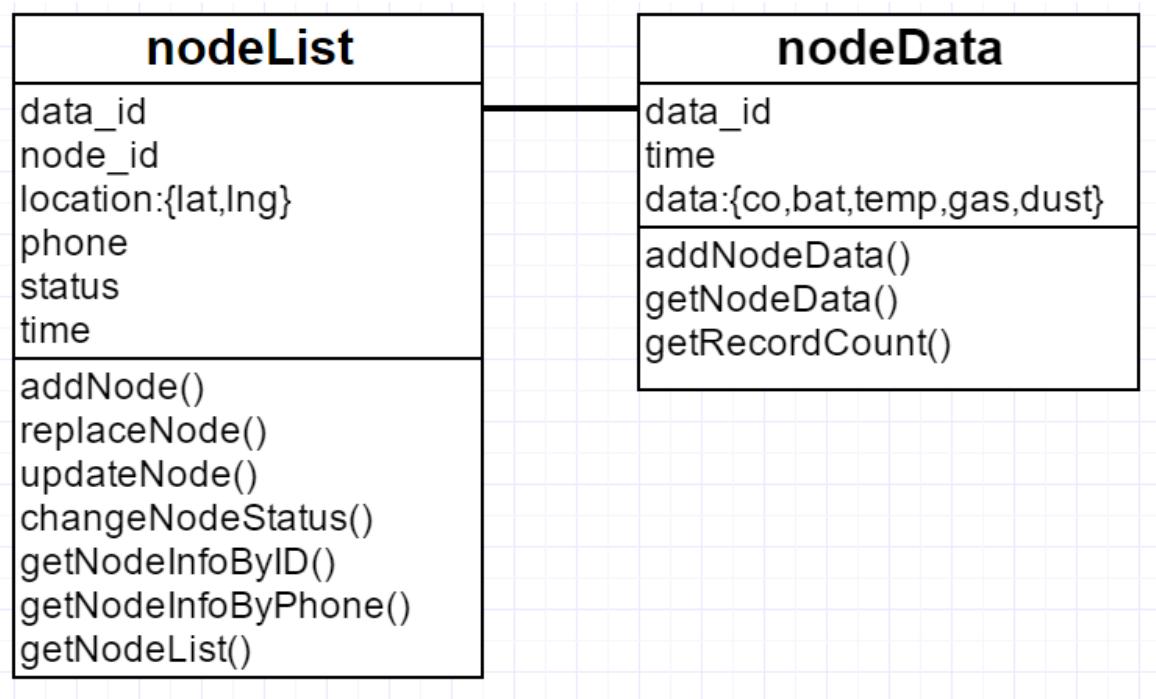
```

### 3.3.2 Xây dựng Database

Hệ quản lý cơ sở dữ liệu được sử dụng tại hệ thống là RethinkDB như mô hình thiết kế hệ thống tại mục 3.1.1 . Cơ sở dữ liệu được chia làm 3 bảng: nodeList, nodeData và user được miêu tả tại các hình 3.33 và 3.34

Các dữ liệu về node được chia làm hai bảng:

- nodeList: chứa các thông tin về các sensor node, trong đó data\_id đóng vai trò tương tự như foreign key của bảng nodeData để có thể truy xuất dữ liệu.
- nodeData: tập trung những dữ liệu thu thập được từ các sensor node theo thời gian thực.



Hình 3.33 Mô hình cấu trúc dữ liệu của Sensor Node

Về phía user, chỉ có một bảng quản lý người dùng bao gồm các thông tin cơ bản để xác thực và quyền thay đổi chỉnh sửa dữ liệu sensor node.

<b>user</b>	
username	
password	
fullname	
status	
role	
created_time	
updateUser()	
registerUser()	
getUserList()	
getUserByID()	
getUserByMail()	

Hình 3.34 Mô hình cấu trúc dữ liệu của User

Vì mục hệ thống phát triển dừng ở mức xây dựng và thiết kế hệ thống thu thập và truy vấn dữ liệu từ các sensor node nên chưa có bảng phân quyền sở hữu giữa người dùng và sensor node.

### 3.3.3 Hiện thực API

Dựa trên mô hình thiết kế API tại 3.1.4, API được hiện thực và phát triển.

#### API đăng ký và xác thực người dùng

Các API này được xây dựng để quản lý và phân quyền người dùng, cho phép sử dụng các API về quản lý thông tin của các sensor node.

##### **Đăng nhập:**

Hệ thống sử dụng module Passport chạy trên nền Nodejs, có chức năng dễ dàng thiết lập xác thực người dùng. Module này có thể liên kết với các tài khoản phổ biến như Facebook, Google, Twitter... để hỗ trợ phương thức đăng nhập tiện lợi nhất cho

người dùng. Tuy nhiên hệ thống chỉ sử dụng phương thức đăng nhập và quản lý người dùng ở mức local vì là phương thức phù hợp với đề tài.

Chi tiết API đăng nhập:

POST: /auth/login

data: {username, password}

=====

RESPOND:

case Success:

{

    code: 1,

    username,

    message: 'Login successful'

}

case Failure

{

    code: 0,

    message: "Invalid username or password"

}

=====

### Lưu thông tin người dùng bằng session:

Sau khi đăng nhập, một session, còn gọi là phiên làm việc, sẽ được hình thành và lưu trữ trong thời gian nhất định. Session cho phép người dùng có thể vào các trang quản lý với quyền hạn cho phép mà không yêu cầu đăng nhập lại, cũng như được phép sử dụng các API quản lý sensor node.

Session sẽ tự động xóa sau một thời gian quy định, cụ thể ở hệ thống này là 10 ngày kể từ lần đăng nhập gần nhất. Người dùng có thể xóa thủ công bằng cách đăng xuất qua phương thức:

GET: /auth/logout

=====

RESPOND:

case Success:

{

    code: 1,

    message: "User has logged out"

}

case Failure:

{

    code: 0,

    message: "User has not logged in yet"

}

Đăng ký:

POST: /user/register

data:{username, password, name, mail}

=====

RESPOND:

{

    code: -1,

    message: "Username existed"

}

{

```

        code: -2,
        message: "Mail is used",
    }
{
    code: 1,
    message: "Account created successful"
}
=====
```

### API cung cấp và quản lý sensor node

Các API được miêu tả tại bảng 3.1, các request và respond được miêu tả chi tiết:

- API thiết lập và cấu hình các sensor node: initnew, updatenode và replace node yêu cầu phải có đăng nhập từ API xác thực người dùng để giữ session và sau đó mới có thể sử dụng. Các API này có cấu trúc tương tự nhau và được mô tả:

```

POST: /node/initnew
data: {node_id,lat{lng},phone,status}
=====
RESPOND:
case Success:
{
    code: 1,
    message: 'Add node successful'
}
case Duplicated:
{
    code: 0,
```

```

        message: 'Node duplicated'
    }

case Failure:
{
    code: 0,
    message: 'Add node failure'
}

case Not authenticated:
{
    code: -1,
    message: 'You are not authenticated'
}
=====
```

- API get thông tin và dữ liệu các sensor node: sử dụng phương thức GET để truy xuất dữ liệu về thông tin sensor node và dữ liệu các cảm biến. Hệ thống được thiết kế mang tính mở và chia sẻ nên không yêu cầu đăng nhập để truy xuất dữ liệu. Các thông tin đặc tả chi tiết API:

GET: /node/getinfo?\*query  
Ex: /node/getinfo?id=1&status=0  
/nodes/getinfo?list=1&status=1

Query:

- + id=\*node ID\* để get node bằng ID
- + phone: \*node phonenumbers\* để get node bằng phone number
- + list=1 để get list các node
- + status=1 để get node đang ở trạng thái active, = 0 để node ở tắt cả trạng thái, cũng như lịch sử thay đổi node

---

=====

RESPOND:

data:

- single node:{node\_id,location:{lat,lng},phone,,time,data\_id,status}
  - list: array of {node\_id,location:{lat,lng},phone,time,data\_id,status}
- 
- =====

- API push dữ liệu từ sensor node: cũng tương tự như API get dữ liệu, API push dữ liệu sử dụng method GET thay cho POST để hỗ trợ dễ dàng trong việc thiết lập các thiết bị IoT đóng góp dữ liệu. Lượng dữ liệu của các sensor node không quá lớn, chỉ bao gồm những trường sensor ID và node ID nên có thể tích hợp vào thanh header của request.

GET: /node/pushdata?\*query

Ex: /node/pushdata?node\_id=1&s1=22&s2=33&s3=44&s4=55&s5=66

Query:

- + node\_id: node ID
  - + s1,s2,s3,s4,s5: value of each sensor
- 
- =====

RESPOND:

case Success:

```
{
    code: 1,
    message: 'Add node data successful'
}
```

case Failure:

```
{
    code: 0,
```

```

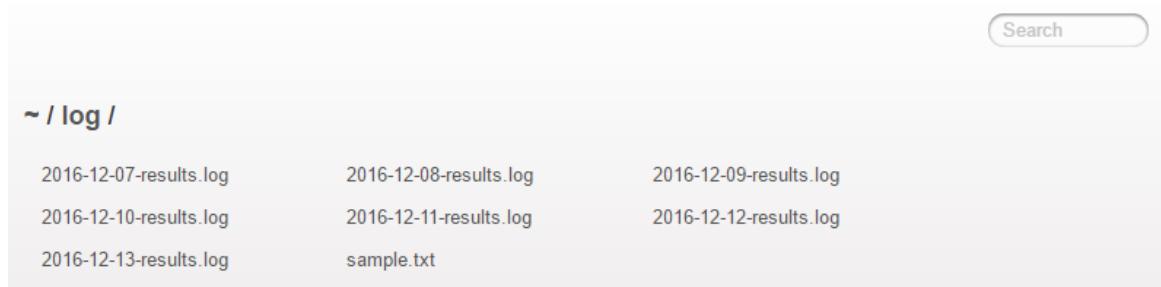
        message: 'Add node data failure'
    }

=====

```

### Cung cấp API log

Để hỗ trợ bên nhà phát triển thử ba ứng dụng API tốt hơn, hệ thống có cung cấp các thông tin hoạt động của server, các request và respond cũng như các dữ liệu được truy vấn. Khi truy cập vào đường dẫn /log/, chúng ta sẽ có những file log tương ứng với các ngày như hình 3.35 :



Hình 3.35 Hiển thị các file log

Ví dụ cụ thể của một đoạn log:

```

1 {"level": "info", "message": "IP:123.151.42.61 GET /route", "timestamp": "10:48:56"}
2 {"level": "info", "message": "IP:14.161.14.94 GET /statis route", "timestamp": "10:49:22"}
3 {"level": "info", "message": "IP:14.161.14.94 GET /graph route", "timestamp": "10:49:27"}
4 {"level": "info", "message": "IP:14.161.14.94 GET /node/getinfo: get list node info - number of nodes 12", "timestamp": "10:49:27"}
5 {"level": "info", "message": "IP:14.161.14.94 GET /node/getinfo: get node info by id:9", "timestamp": "10:49:29", {"data_id": "106c59fe-753d-4a76-8418-88fd0df68d0c", "id": "87b7f859-8c1e-4608-9645-5105"}

```

```

    ebb38e9b", "location": {"lat": 10.91085286140159, "lng":  

    : 106.99790954589844}, "node_id": "9", "phone": "0123456899", "status":  

    : 1, "time": "2016-12-06T02:08:57.943Z"}}  

6 {"level": "info", "message": "IP:14.161.14.94 GET /node/getdata: Node  

    data ID 9 number of records: 120", "timestamp": "10:49:29"}

```

Các thông tin được thể hiện trong file log đủ thông tin để cho phép nhà phát triển theo dõi và áp dụng trong việc hiện thực sản phẩm.

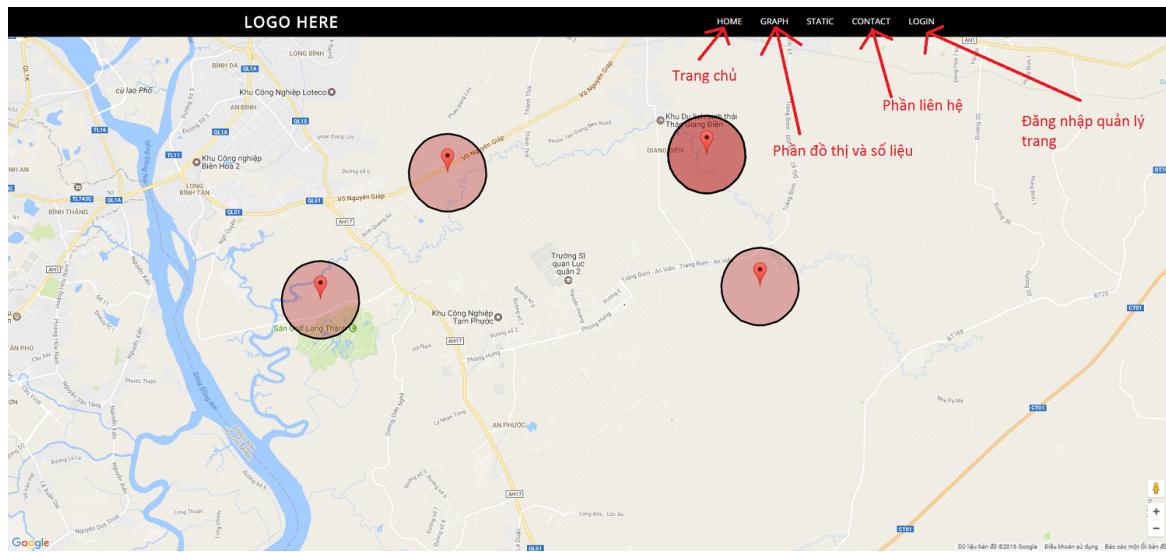
### 3.3.4 Xây dựng Web Server

#### 3.3.5 Hiện thực giao diện

Sử dụng framework express của Nodejs và HTML để xây dựng frontend

##### Giao diện chính của Web Server

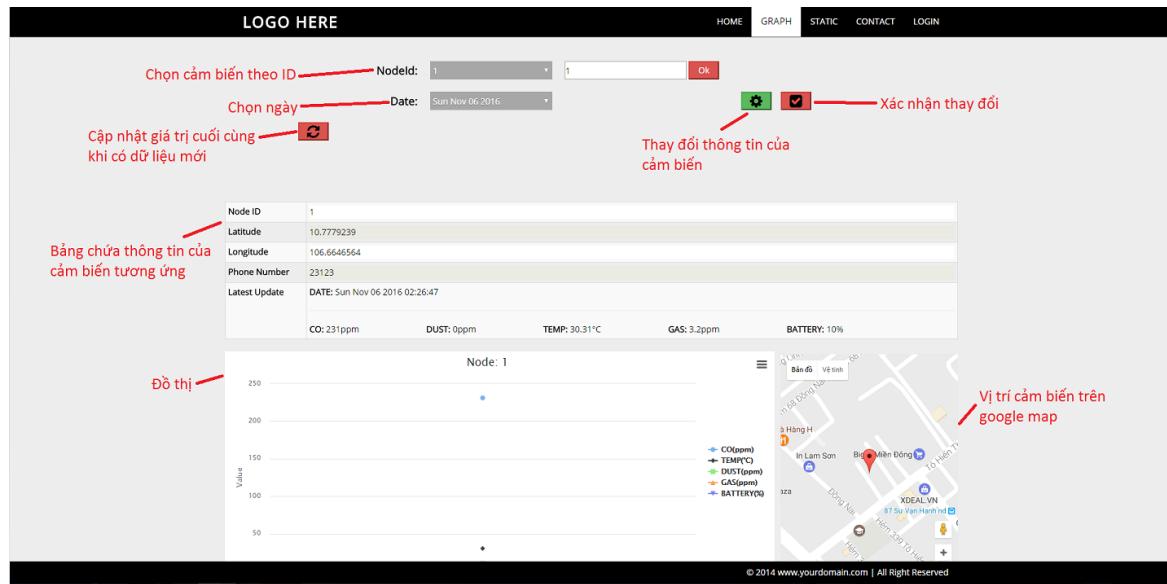
Phần này chứa các nút menu chức năng cùng với google map để theo dõi vị trí hoạt động của các cảm biến.



Hình 3.36 Giao diện chính của Web Server

## Giao diện đồ thị dữ liệu

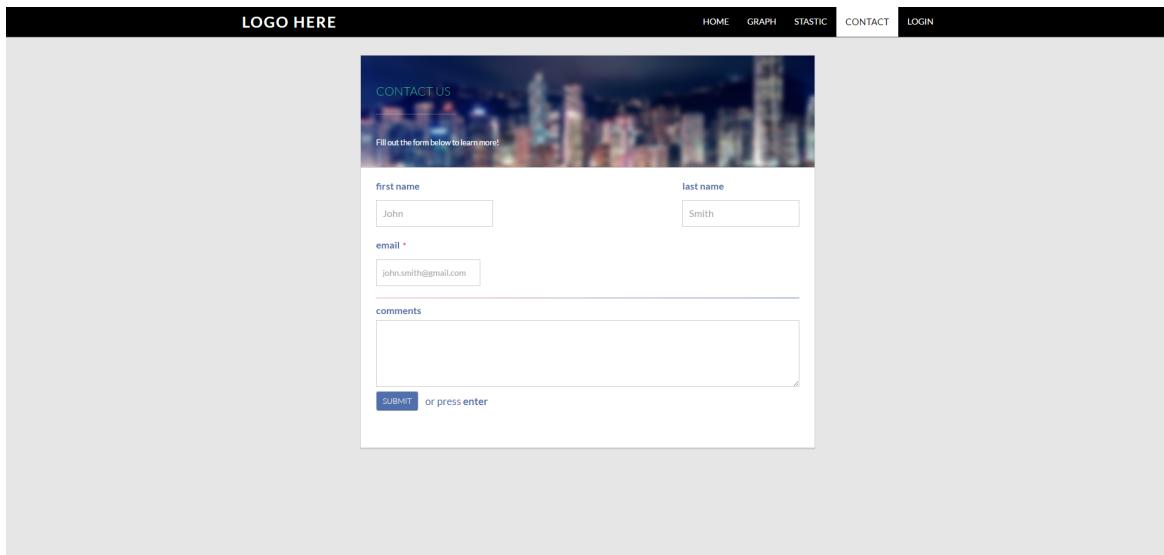
Là nơi hiện thị biểu đồ của các thông số cũng như thông tin tương ứng của các cảm biến.



Hình 3.37 Giao diện đồ thị dữ liệu

## Giao diện nhận feedback từ người dùng qua email

Tương tác giữa người dùng với người quản lý server. Nội dung được gửi thông qua gmail.



Hình 3.38 Giao diện nhận feedback người dùng

### Giao diện dành cho người quản lý

Khu vực này dành cho người quản lý truy cập nhằm mục đích chỉnh sửa, thay thế các node cảm biến.

NODE INFORMATIONS						
NodeID	Lat	Lng	Phone	Status		
9	10.91085286140159	106.99790954589844	123456	1	Update Node: 9	Replace Node: 9
8	10.91085286140159	106.99790954589844	123456	1	Update Node: 8	Replace Node: 8
5	10.795789678778172	106.70162200927734	123456	1	Update Node: 5	Replace Node: 5
7	10.745620111593173	106.66497230529785	123456	1	Update Node: 7	Replace Node: 7
1	10.7779239	106.6646564	23123	1	Update Node: 1	Replace Node: 1

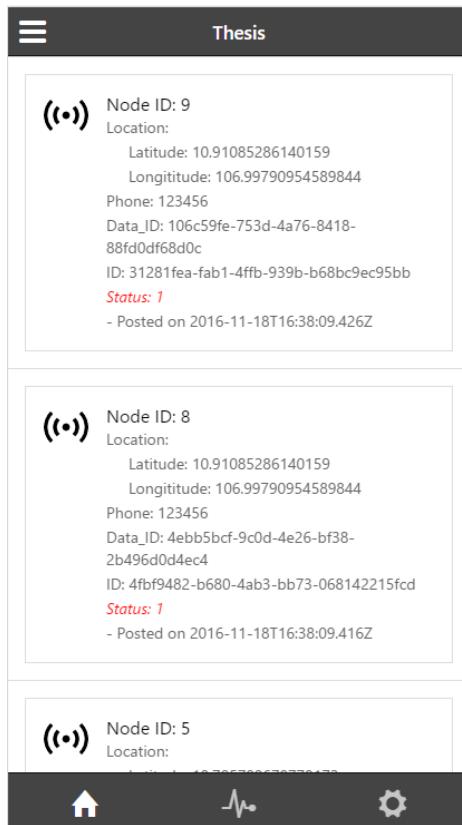
Hình 3.39 Giao diện quản lý các node cảm biến

## 3.4 Ứng dụng thiết bị di động

### 3.4.1 Hiện thực giao diện và chức năng

#### Giao diện chính ứng dụng di động

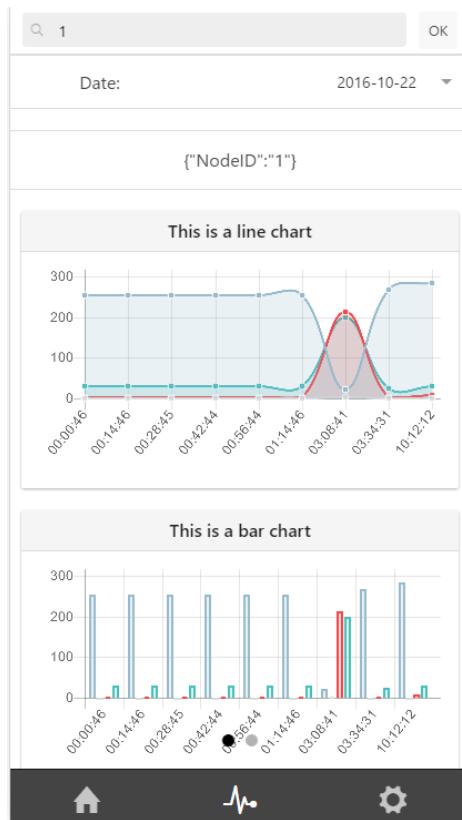
Tại giao diện chính của ứng dụng di động người dùng có thể xem được danh sách thông tin chi tiết của các node cảm biến có trong hệ thống.



Hình 3.40 Giao diện chính ứng dụng di động

### Giao diện xem dữ liệu trên ứng dụng di động

Giao diện này cho phép người dùng theo dõi được dữ liệu của từng node cảm biến theo 2 loại biểu đồ như Hình 3.41 thể hiện.



Hình 3.41 Giao diện xem dữ liệu trên ứng dụng di động

# Chương 4

## Kết quả thực nghiệm và Đánh giá

### 4.1 Tính ổn định của node cảm biến

này nói về máy tháng đo, gió mưa đồ

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

### 4.2 Tính ổn định của Server

Nói về thời gian đáp ứng truyền nhận dữ liệu

## 4.3 Đánh giá lập trình ứng dụng Mobile sử dụng APIs

- Ứng dụng được viết có thể được xài trên nhiều platforms.
- Đã hiện thực được file APK cho mobile.
- Xây dựng một số các chức năng cơ bản như: hiển thị thông tin các node cảm biến cho màn hình chính, vẽ đồ thị, liên kết với web browser ...
- Vẫn còn một vài chức năng cần hoàn thiện: đăng nhập quản lý người dùng, các chức năng phụ liên quan đến user ...
- Khó khăn trong việc debugging vì một số plugin hay chức năng không chạy trên nền web app.
- Hiệu suất thấp do nó không hoàn toàn tự nhiên, có thể gặp sự cố khi làm ứng dụng game hay các ứng dụng dùng nhiều tài nguyên.

-----

# Chương 5

## Tổng kết và Hướng phát triển cho tương lai

### 5.1 Tổng kết

#### 5.1.1 Những gì đạt được

##### Server

- Cung cấp API và API log cho nhà phát triển.
- Áp dụng mô hình kiến trúc hệ thống thuận tiện cho phát triển ứng dụng realtime.
- Xây dựng giao diện trực quan và thân thiện với người dùng, liên kết cơ sở dữ liệu thời gian thực cung cấp thông tin kịp thời để tiện lợi cho quá trình theo dõi và đánh giá.
- Đồ thị cho việc theo dõi các thông số.
- Giao tiếp thông qua gmail giữa người dùng và người quản lý.

### Ứng dụng thiết bị di động

- Sử dụng Framework Ionic xây dựng được ứng dụng di động trên nhiều nền tảng khác nhau.
- Liên kết ứng dụng với API được cung cấp, thực hiện một số chức năng cơ bản: hiển thị thông tin node, đồ thị...

#### 5.1.2 Những khó khăn và hạn chế

- Dành nhiều thời gian để tiếp cận với hướng lập trình phần mềm và công nghệ mới.
- Dữ liệu xử lý lớn, server cần phải đáp ứng được vấn đề thời gian thực và tránh xảy ra lỗi.
- Chỉ dùng ở mức thu thập và thống kê, chưa phát triển phân tích các thông số môi trường.
- Vẫn còn một số chức năng chưa hoàn thiện: quản lý người dùng cho Mobile app.

## 5.2 Hướng phát triển tương lai

- Tiếp tục hoàn thiện phần Server hoạt động hiệu quả, đáp ứng hoạt động cho một số lượng lớn các node cảm biến.
- Xây dựng hệ thống quản lý node theo người dùng, đáp ứng nhu cầu quản lý và thống kê theo từng người.
- Xây dựng thuật toán phân tích dữ liệu môi trường và áp dụng Machine learning trong việc phân tích dữ liệu để đưa ra kết quả dự đoán tính trạng ô nhiễm môi trường và mật độ giao thông.

- Hoàn thành các chức năng cần thiết cho Mobile app.

# Tài liệu tham khảo

- [1] Blog, R. (2014). Giới thiệu về websocket và node.js. <http://blog.rikkeisoft.com/seminar-gioi-thieu-ve-websocket-va-node-js/>. [Online; Posted 11-Sept-2014].
- [2] Freetuts.net (2016). Javascript là gì? <http://freetuts.net/javascript-la-gi-viet-ung-dung-javascript-dau-tien-263.html>. [Online; Posted 06-Jan-2015].
- [3] Jaitla, J. (2015). Websockets vs rest: Understanding the difference. <https://www.pubnub.com/blog/2015-01-05-websockets-vs-rest-api-understanding-the-difference/>. [Online; Posted 05-Jan-2015].
- [4] RethinkDB (2015a). What is rethinkdb? <http://nosql-database.org/>. [NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable].
- [5] RethinkDB (2015b). What is rethinkdb? <https://rethinkdb.com/faq/>. [Online; RethinkDB FAQ].
- [6] Vietjack (2015). Nodejs là gì? [http://vietjack.com/nodejs/nodejs\\_la\\_gi.jsp](http://vietjack.com/nodejs/nodejs_la_gi.jsp).
- [7] Wikipedia (2016a). Api - application programming interface. [https://en.wikipedia.org/wiki/Application\\_programming\\_interface](https://en.wikipedia.org/wiki/Application_programming_interface). [Online; lastest update 06-Dec-2016].
- [8] Wikipedia (2016b). Javascripts. <https://vi.wikipedia.org/wiki/JavaScript>. [Online; lastest update 1-Nov-2016].
- [9] Wikipedia (2016c). Tcp - transmission control protocol. <https://vi.wikipedia.org/wiki/TCP>. [Online; lastest update 17-May-2016].
- [10] www.raspbian.org (2012). What is raspbian? [www.raspbian.org](http://www.raspbian.org).