

**BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN**



**KHOÁ LUẬN TỐT NGHIỆP  
IOT VÀ ỨNG DỤNG  
(SMART HOME)**

Sinh viên thực hiện : 14127871 Phan Bảo Trinh  
14026751 Vương Quốc Việt  
Giáo viên hướng dẫn : ThS. Nguyễn Thành Thái  
Lớp : ĐHCNTT10B

***TP. HỒ CHÍ MINH, THÁNG 05 NĂM 2019***

## LỜI NÓI ĐẦU

Internet of Things – kết nối vạn vật đang dần trở nên phổ biến trong cuộc Cách Mạng Công Nghiệp 4.0. Khi mà các thiết bị được kết nối với nhau thông qua Internet. Chúng có khả năng giao tiếp với nhau, chia sẻ cho nhau dữ liệu, tự đưa ra quyết định và được con người giám sát, điều khiển thông qua các hệ thống quản lý hàng loạt. Việc này đang dần có tác động mạnh mẽ lên nhiều lĩnh vực của đời sống xã hội con người. Nhiều doanh nghiệp công nghệ thông tin của Việt Nam cũng như trên thế giới đã và đang nắm bắt cơ hội này như một bước phát triển vượt bậc về công nghệ. Tiêu biểu như các hệ thống nông nghiệp thông minh, xe tự hành, hệ thống phát hiện cảnh báo cháy, ....

Với sự phát triển hạ tầng của Internet như hiện nay, Internet of Things được xem như xu hướng trong tương lai. Nên nhóm đã tìm hiểu và dựa theo kiến thức đã tích lũy được, nhóm đã chọn đề tài phát triển mô hình nhà thông minh (Smart Home). Với mong muốn thiết kế một mô hình nhà mà các thiết bị sẽ kết nối với nhau thông qua Internet. Người dùng sẽ kiểm soát mọi thứ trong nhà mình và có thể điều khiển thiết bị trong nhà.

Mô hình bao gồm các thiết bị điện tử thông dụng và được phát triển bằng ngôn ngữ lập trình Python và Angular để xử lý dữ liệu và lập trình giao diện web. Quá trình xây dựng và phát triển mô hình đã chạy ổn định.

Trong quá trình thực hiện và phát triển đề tài nhóm chúng em đã gặp một số vướng mắc về kỹ thuật và sự thiếu hiểu biết. Nhưng nhờ sự trợ giúp và hỗ trợ nhiệt tình của thầy Nguyễn Thành Thái nhóm đã cố gắng để hoàn thiện đề tài một cách tốt nhất có thể. Do kiến thức và kinh nghiệm của nhóm em có hạn chế nên sẽ không tránh khỏi những thiếu sót mong các thầy cô bổ sung và đóng góp ý kiến để đề tài được phát triển một cách hoàn thiện hơn. Nhóm chúng em xin chân thành cảm ơn sự giúp đỡ tận tình của thầy Nguyễn Thành Thái.

## **DANH MỤC TỪ VIẾT TẮT**

API	Application Programming Interface
CSS	Cascading Style Sheets
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
JRE	Java Runtime Environment
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
SSH	Secure Shell

## DANH MỤC HÌNH ẢNH

Hình 1.1. Mô hình IoT cơ bản.....	9
Hình 2.1. Raspberry Pi 3 Model B+.....	14
Hình 2.2. Mô hình giao thức MQTT.....	18
Hình 3.1. Tải xuống JRE.....	21
Hình 3.2. Tải xuống Arduino.....	22
Hình 3.3. Giao diện Arduino IDE.....	22
Hình 3.4. Thư mục Arduino Drivers.....	23
Hình 3.5. Cấu hình Arduino IDE.....	24
Hình 3.6. Cài đặt thư viện cho ESP8266.....	24
Hình 3.7. Tải xuống NOOBS.....	26
Hình 3.8. Phần mềm SD Formatter.....	26
Hình 3.9. Kết nối Raspberry Pi qua SSH.....	28
Hình 3.10. Phần mềm Remote Desktop.....	29
Hình 3.11. Python3 version.....	30
Hình 3.12. Pip3 version.....	30
Hình 3.13. Phiên bản NPM và Angular.....	31
Hình 3.14. Kiểm tra Mosquitto.....	32
Hình 4.1. Mô hình kết nối thiết bị.....	33
Hình 4.2. Kết nối giữa ESP8266 và Raspberry Pi.....	35
Hình 4.3. Cấu trúc cơ sở dữ liệu.....	35
Hình 4.4. Mô hình Web Server trên Raspberry Pi.....	36
Hình 4.5. Mô hình tổng quát.....	37
Hình 4.6. Thông số cấu hình Wifi của ESP8266.....	38
Hình 4.7. ESP8266 kết nối Wifi.....	38
Hình 4.8. ESP8266 kết nối MQTT Server.....	38

Hình 4.9. Hàm bật/tắt đèn. ....	39
Hình 4.10. Create_database_script.sql. ....	40
Hình 4.11. Create_table_script.sql. ....	40
Hình 4.12. Create_data_static.sql. ....	41
Hình 4.13. Xây dựng giao diện Dashboard. ....	44
Hình 4.14. Xây dựng giao diện Monitor. ....	44
Hình 4.15. Xây dựng giao diện Analytic. ....	45
Hình 4.16. Thư mục mã nguồn. ....	45
Hình 4.17. Thông báo nạp mã nguồn thành công. ....	46
Hình 4.18. Dữ liệu nhận và gửi trên ESP8266. ....	47
Hình 4.19. Kiểm tra tạo cơ sở dữ liệu. ....	48
Hình 4.20. Kiểm tra dữ liệu bảng device_status. ....	48
Hình 4.21. Kiểm tra API. ....	49
Hình 4.22. Kiểm tra giao diện Dashboard. ....	49
Hình 4.23. Kiểm tra giao diện Monitor. ....	50
Hình 4.24. Kiểm tra giao diện Analytic. ....	50

## DANH MỤC BẢNG BIỂU

Bảng 2.1. Bảng kinh phí.....	18
Bảng 4.1. Các cổng kết nối thiết bị với ESP8266. ....	34
Bảng 4.2. Kênh truyền thiết bị. ....	34
Bảng 4.3. Định nghĩa giá trị trả về của thiết bị. ....	39
Bảng 4.4. Cấu hình kết nối cơ sở dữ liệu. ....	41
Bảng 4.5. Cấu hình kết nối MQTT Server. ....	42
Bảng 4.6. Nhận message từ MQTT Server. ....	42
Bảng 4.7. Xử lý dữ liệu nhiệt độ. ....	42
Bảng 4.8. Lưu trữ dữ liệu vào cơ sở dữ liệu.....	43
Bảng 4.9. Thông số API. ....	43

## MỤC LỤC

LỜI NÓI ĐẦU .....	1
DANH MỤC TỪ VIẾT TẮT.....	2
DANH MỤC HÌNH ẢNH .....	3
DANH MỤC BẢNG BIỂU .....	5
MỤC LỤC .....	6
Chương 1: TỔNG QUAN. ....	9
1.1. Internet of Things. ....	9
1.1.1. Giới thiệu. ....	9
1.1.2. Mô hình.....	10
1.1.3. Ứng dụng. ....	10
1.1.4. IoT và Cách Mạng Công Nghiệp 4.0.....	11
1.2. Smart Home.....	11
1.2.1. Giới thiệu. ....	11
1.2.2. Mô tả đề tài. ....	12
1.2.3. Công nghệ, thiết bị sử dụng.....	12
1.2.4. Mục tiêu, phạm vi đề tài. ....	12
Chương 2: THIẾT BỊ VÀ GIAO THỨC. ....	14
2.1. Thiết bị.....	14
2.1.1. Raspberry.....	14
2.1.2. Arduino. ....	15
2.1.3. Cảm biến.....	16
2.1.4. Kinh phí. ....	17
2.2. Giao thức Message Queue Telemetry Transport.....	18
2.2.1. Khái niệm. ....	18
2.2.2. Kiến trúc. ....	18

2.2.3. Ưu điểm của MQTT .....	19
2.2.4. Message và Topic. ....	19
2.2.5. Publish và Subscribe.....	19
2.2.6. Qualities of Service. ....	20
Chương 3: CÀI ĐẶT VÀ CẤU HÌNH THIẾT BỊ.....	21
3.1. Cài đặt và cấu hình Arduino.....	21
3.1.1. Cài đặt JRE. ....	21
3.1.2. Cài đặt Arduino IDE.....	22
3.1.3. Cài đặt Driver. ....	23
3.1.4. Cấu hình Arduino IDE lập trình ESP8266. ....	23
3.2. Cài đặt Raspberry. ....	25
3.2.1. Hệ điều hành Rasbian. ....	25
3.2.2. Cấu hình Raspbian.....	27
3.2.3. Cài đặt môi trường lập trình. ....	29
3.2.4. Cơ sở dữ liệu PostgreSQL.....	31
3.2.5. MQTT Server. ....	32
Chương 4: XÂY DỰNG ỨNG DỤNG.....	33
4.1. Thiết kế.....	33
4.1.1. Mô hình kết nối cảm biến với ESP8266.....	33
4.1.2. Mô hình kết nối giữa ESP8266 và Raspberry Pi.....	35
4.1.3. Cấu trúc cơ sở dữ liệu.....	35
4.1.4. Mô hình Web Server.....	36
4.1.5. Mô hình tổng quát.....	37
4.2. Xây dựng và triển khai. ....	37
4.2.1. Lập trình ESP8266.....	37
4.2.2. Tạo cơ sở dữ liệu và bảng.....	40



4.2.3. Kết nối cơ sở dữ liệu: .....	41
4.2.4. Kết nối MQTT Server.....	42
4.2.5. Thu thập và xử lý dữ liệu.....	42
4.2.6. Xây dựng giao diện lập trình ứng dụng.....	43
4.2.7. Xây dựng giao diện người dùng. ....	44
4.2.8. Chạy ứng dụng.....	45
4.3. Kiểm thử.....	46
4.3.1. Kết nối giữa các thiết bị.....	46
4.3.2. Lưu trữ dữ liệu.....	48
4.3.3. Ứng dụng web. ....	49
KẾT LUẬN.....	51
NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN .....	52
TÀI LIỆU THAM KHẢO.....	53

## Chương 1: TỔNG QUAN.

Giới thiệu tổng quan về Internet of Thing trong cuộc Cánh Mạng Công Nghệ 4.0 và đề tài Smart Home. Mô tả đề tài công nghệ sử dụng và mục tiêu, phạm vi thực hiện đề tài của nhóm.

### 1.1. Internet of Things.

#### 1.1.1. Giới thiệu.

Internet of Things (IoT) là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết cũng như sự tồn tại của chúng trong một kiến trúc mạng tính kết nối. Đây là một viễn cảnh trong đó mọi vật, mọi con vật hoặc con người được cung cấp các định danh và khả năng tự động truyền tải dữ liệu qua một mạng lưới mà không cần sự tương tác giữa con người với con người hoặc con người với máy tính. IoT tiến hoá từ sự hội tụ của các công nghệ không dây, hệ thống vi cơ điện tử (MEMS) và Internet. Là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó. Cụm từ IoT được đưa ra bởi Kevin Ashton vào năm 1999. Một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT.



Hình 1.1. Mô hình IoT cơ bản

IoT đã phát triển do sự hội tụ của nhiều công nghệ, phân tích thời gian thực, máy học, cảm biến hàng hoá và hệ thống nhúng.

### **1.1.2. Mô hình.**

Mô hình cơ bản của IoT gồm 3 phần:

1. Cảm biến và thiết bị truyền động: có nhiệm vụ đọc giá trị từ các cảm biến như âm thanh, ánh sáng, nhiệt độ, ... và chuyển thành tín hiệu điện để giúp cho các thiết bị hiểu và đưa ra những hành động hợp lý.
2. Kết nối: các tín hiệu đọc được sẽ được truyền tải lên mạng lưới thông qua các phương thức giao tiếp khác nhau như Wifi, Bluetooth, ZigBee, Lora.
3. Con người và quy trình: các đầu vào của mạng lưới IoT sẽ được tổng hợp thành một hệ thống bao gồm dữ liệu, con người và các quy trình với mục đích đưa ra quyết định tốt hơn.

Kết quả sau cùng sẽ được hiển thị trên trình duyệt web hoặc thiết bị di động của con người để người dùng có thể giám sát, điều khiển, đưa ra quyết định.

### **1.1.3. Ứng dụng.**

Trong thực tế, IoT có thể ứng dụng trong nhiều lĩnh vực khác nhau, đem lại hiệu quả cao cho các lĩnh vực ứng dụng. Một số ứng dụng phổ biến của IoT:

- Smart Home – Ngôi nhà thông minh: là ứng dụng phổ biến và là chủ đề IoT được tìm kiếm nhiều nhất trên Google. Là ứng dụng cho phép người dùng giám sát, điều khiển ngôi nhà của mình thông qua các thiết bị di động, ngoài ra còn tự động hóa một số công việc như đóng/mở cửa, bật/tắt đèn, ... Đây là một ứng dụng tiện ích, nâng cao chất lượng cuộc sống của mọi gia đình.
- Smart Car – Chiếc xe thông minh: với khả năng tự nhận biết vị trí, tốc độ và chương ngại vật để có thể tự lái và cảnh báo cho người lái xe.
- Smart City – Thành phố thông minh: tập hợp gồm nhiều thiết bị IoT, với khả năng thu thập và đáp ứng mọi nhu cầu của người dân.

#### **1.1.4. IoT và Cách Mạng Công Nghiệp 4.0.**

Công nghiệp 4.0 là thuật ngữ dùng để chỉ quá trình phát triển trong quản lý sản xuất và sản xuất dây chuyền, ngoài ra còn được dùng để nói đến cuộc Cách Mạng Công Nghiệp lần thứ tư. Mục tiêu của nền công nghiệp 4.0 là phát triển dây chuyền sản xuất tự động. Trong Cách Mạng Công Nghiệp 4.0, con người có thể giao tiếp và giám sát thiết bị thay vì vận hành chúng.

Với sự phát triển của cuộc Cách Mạng Công Nghiệp 4.0, IoT được xem là một nhánh phát triển với việc đảm nhận nhiệm vụ kết nối mọi thiết bị lại với nhau, giúp các thiết bị có thể truyền, nhận dữ liệu thu thập từ các cảm biến, đồng thời kết nối con người với thiết bị.

### **1.2. Smart Home.**

#### **1.2.1. Giới thiệu.**

Trong các ngôi nhà hiện đại ngày nay, số lượng trang thiết bị điện, điện tử đang không ngừng gia tăng. Tuy nhiên, do khác nhau về kiến trúc, việc điều khiển các thiết bị đôi khi bất cập. Thêm vào đó, việc điều khiển các thiết bị một cách thủ công với khoảng các địa lý lớn không dễ. Vì vậy, việc áp dụng các công nghệ điều khiển tự động nhằm giải quyết tương tác giữa môi trường và các thiết bị trong nhà một cách linh hoạt, dễ dàng là điều tất yếu, khái niệm nhà thông minh ra đời.

Nhiều công nghệ đã được áp dụng khi xây dựng nhà thông minh. Tuy nhiên, sự phức tạp nằm ở chỗ các hệ thống điều khiển phải cân bằng giữa sự phức tạp của hệ thống và tính tiện dụng cho người dùng, đặc biệt là có thể được điều khiển ở bất cứ đâu, từ trong chính ngôi nhà đó hay bất kỳ nơi nào trên thế giới thông qua điện thoại hoặc Internet.

Với IoT mỗi đồ vật, thiết bị, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. Bên cạnh đó, IoT có thể triển khai một mạng lưới các thực thể thông minh, có khả năng tự tổ chức và hoạt động tùy theo tình huống, môi trường, đồng thời chúng cũng có thể liên lạc với nhau để trao đổi thông tin, dữ liệu. Với khả năng định danh

cao, số lượng các thực thể trong hệ thống được định danh chính xác, duy nhất, đảm bảo tốt khả năng quản lý, điều khiển của hệ thống.

### **1.2.2. Mô tả đề tài.**

Với một ngôi nhà thông minh, con người sẽ quan tâm tới các thông tin về thiết bị trong nhà như nhiệt độ, độ ẩm, thiết bị chiếu sáng, khí gas, ... Và bên cạnh đó là điều khiển các thiết bị từ xa và tự động theo một quy tắc nào đó. Dựa vào những tiêu chí cơ bản đó, nhóm đã chọn sử dụng các thiết bị cảm biến đơn giản như nhiệt độ, độ ẩm, cảm biến khí gas, cảm biến ánh sáng để thu thập dữ liệu và thiết bị giả lập bóng đèn để diễn khiển. Từ những dữ liệu đã thu thập sẽ được lưu trữ vào cơ sở dữ liệu và hiển thị và gửi cảnh báo qua mail cho người dùng theo thời gian thực. Người dùng cũng có thể theo dõi, giám sát ngôi nhà của mình thông qua web và điều khiển bật tắt đèn.

### **1.2.3. Công nghệ, thiết bị sử dụng.**

Công nghệ:

- Ngôn ngữ: Arduino, Python, Angular, HTML, CSS, JQuery.
- Cơ sở dữ liệu: PostgreSQL.
- Web Server: Nginx.
- Giao thức: HTTP, MQTT, WebSocket.

Thiết bị: Raspberry Pi, Arduino, Cảm biến.

### **1.2.4. Mục tiêu, phạm vi đề tài.**

Mục tiêu:

- Thu thập dữ liệu từ các thiết bị cảm biến.
- Lưu trữ trong cơ sở dữ liệu.
- Hiển thị thông tin cho người dùng thông qua web.
- Gửi cảnh báo khi có dữ liệu bất thường qua mail.
- Cho phép người dùng điều khiển thiết bị.

Phạm vi đề tài: vấn đề về tài chính và thời gian thực hiện cũng như hiểu biết của nhóm nên đề tài thực hiện trong phạm vi các cảm biến đơn giản như cảm biến nhiệt độ, độ ẩm, cảm biến cường độ ánh sáng, cảm biến khí gas và thiết bị điều khiển là

bóng đèn LED. Độ chính xác của cảm biến là tương đối nên dữ liệu sẽ không đảm bảo tính tuyệt đối.

**Kết luận:** IoT đang là xu thế của tương lai và Smart Home được xem là một đề tài nổi bật nhất để nghiên cứu, phát triển và ứng dụng cho cuộc sống con người.

## Chương 2: THIẾT BỊ VÀ GIAO THỨC.

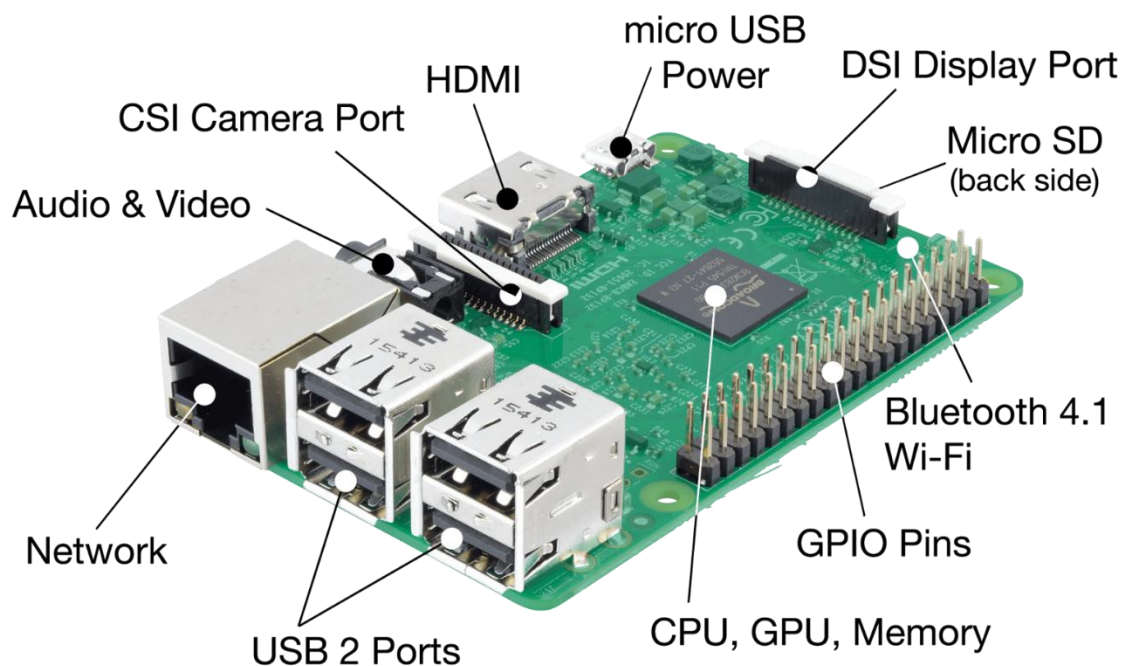
Giới thiệu cơ bản các thiết bị sử dụng: Raspberry, Arduino, cảm biến. Và giao thức kết nối giữa các thiết bị được nhóm chọn sử dụng.

### 2.1. Thiết bị.

#### 2.1.1. Raspberry.

Raspberry Pi là một máy tính rất nhỏ gọn, chỉ gồm có một board mạch (hay còn gọi là máy tính nhúng) kích thước chỉ bằng một thẻ tín dụng, được phát triển tại Anh với mục đích ban đầu là thúc đẩy việc giảng dạy về khoa học máy tính cơ bản trong các trường học và các nước đang phát triển.

Raspberry được sử dụng trong đề tài là Raspberry Pi 3 Model B+.



Hình 2.1. Raspberry Pi 3 Model B+

Thông số kỹ thuật:

- Vi xử lý: Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC @1.4GHz.

- RAM: 1GB LPDDR2 SDRAM.
- Kết nối: 2.4GHz and 5GHz IEEE 802.11 b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0 (Tối đa 300Mbps).
- Hỗ trợ: 40-pin GPIO, 4 cổng USB2.0.
- Video và âm thanh: 1 cổng full-sized HDMI, Cổng MIPI DSI Display, cổng MIPI CSI Camera, cổng stereo output và composite video 4 chân.
- Multimedia: H.264, MPEG-4 decode (1080p30), H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics.
- Lưu trữ: MicroSD.

Điện áp hoạt động: 5V/2.5A DC cổng microUSB, 5V DC trên chân GPIO, Power over Ethernet (PoE) (yêu cầu thêm PoE HAT).

### **2.1.2. Arduino.**

Arduino giống như một máy tính nhỏ mà bạn có thể lập trình để làm nhiều việc khác nhau, và nó tương tác với thế giới thông qua các cảm biến điện tử, đèn, và động cơ. Về cơ bản, nó giúp các dự án điện tử trở nên dễ dàng hơn với bất cứ ai - nhờ đó mà các nghệ sĩ và những tuýp người sáng tạo có thể tập trung biến những ý tưởng của họ thành hiện thực.

Arduino được sử dụng là loại Arduino ESP8266 – là một Arduino được tích hợp sẵn wifi.

Thông số kỹ thuật:

- CPU ESP8266.
- Hỗ trợ kết nối WiFi.
- Tương thích Arduino UNO.
- Có thể lập trình được bằng C/C++, Arduino IDE, Micropython, NodeMCU – Lua.



- Nguồn 9-24V hay 5V từ USB.
- 11 IO, 1 Analog in.
- 4 Mbytes Flash.
- Module ESP-12F chỉ 3.3VDC (tối đa 3.6 VDC).

### 2.1.3. Cảm biến.

#### a. Cảm biến ánh sáng quang trở.

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

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

Thông số kỹ thuật:

- Điện áp làm việc: 3.3 ~ 5VDC.
- Output: Digital.
- Có thể điều chỉnh cường độ ánh sáng phát hiện bằng biến trở gắn trên cảm biến.
- Kích thước: 3.2cm x 1.4cm.

#### b. Cảm biến khí gas MQ2.

Cảm biến khí gas MQ2 là một trong những loại cảm biến được sử dụng để nhận biết: LPG, i-butan, Propane, Methane, Alcohol, Hydrogen, Smoke và khí ga. Được thiết kế với độ nhạy cao, thời gian đáp ứng nhanh. Giá trị đọc được từ cảm biến sẽ được đọc về từ chân Analog của vi điều khiển.

Thông số kỹ thuật:

- Nguồn hoạt động: 5VDC.

- Dòng: 150mA.
- Tính hiệu tương tự (analog).
- Hoạt động trong thời gian dài, ổn định.

### c. Cảm biến số nhiệt độ, độ ẩm DHT11.

DHT11 Là cảm biến rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1-wire (giao tiếp digital 1-wire truyền dữ liệu duy nhất). Cảm biến được tích hợp bộ tiền xử lý tín hiệu giúp dữ liệu nhận về được chính xác mà không cần phải qua bất kỳ tính toán nào.

Thông số kỹ thuật:

- Điện áp hoạt động: 3V - 5V (DC).
- Dãy độ ẩm hoạt động: 20% - 90% RH, sai số  $\pm 5\% RH$ .
- Dãy nhiệt độ hoạt động:  $0^{\circ}C \sim 50^{\circ}C$ , sai số  $\pm 2^{\circ}C$ .
- Khoảng cách truyền tối đa: 20m.

#### 2.1.4. Kinh phí.

Bảng kinh phí bao gồm các thiết bị đã mua và giá tại thời điểm mua.

STT	Tên sản phẩm	Số lượng	Đơn giá (VNĐ)
1	Raspberry Pi 3 Model	1	1.050.000
2	ESP8266 – IoT Wifi Uno	1	250.000
3	Nguồn USB 5V 2.5A Cáp Micro	1	125.000
4	Vỏ Case cho Raspberry	1	110.000
5	Cảm biến khí ga MQ2	1	50.000
6	Breadboard 830 tie – points MB – 102	1	45.000
7	Cảm biến số nhiệt độ, độ ẩm DHT11	1	35.000
8	Cảm biến ánh sáng quang trở	1	18.000
9	Cable Micro USB – B HTC	1	15.000
10	Bộ 20 dây cắm	1	12.000

11	Bộ 20 dây cắm testboard Female – Male 20cm	1	12.000
12	Bộ 20 dây cắm testboard Male – Male 20cm	1	12.000
13	LED chân cắm 5mm	1	4.000
<b>TỔNG CỘNG</b>			1.738.000

Bảng 2.1. Bảng kinh phí.

## 2.2. Giao thức Message Queue Telemetry Transport.

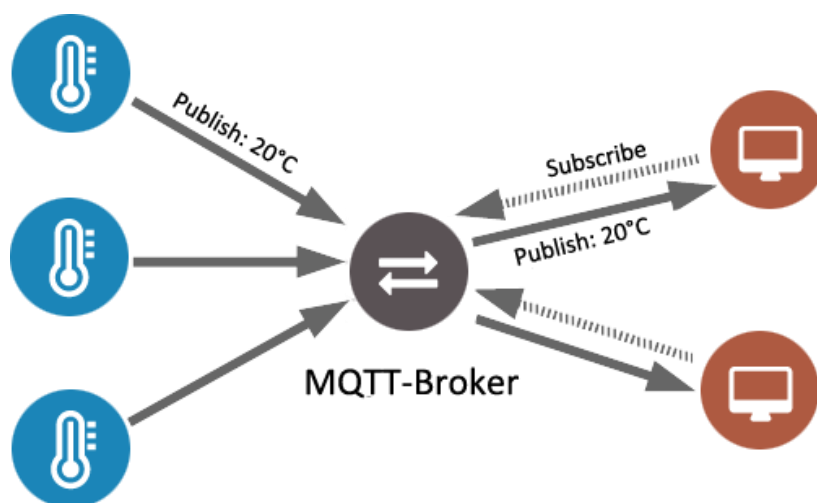
### 2.2.1. Khái niệm.

Message Queue Telemetry Transport (MQTT) là một giao thức thông điệp theo mô hình publish/subscribe (xuất bản – theo dõi), sử dụng băng thông thấp, độ tin cậy cao và có khả năng hoạt động trong điều kiện đường truyền không ổn định.

### 2.2.2. Kiến trúc.

Kiến trúc mức cao của MQTT gồm 2 phần chính là Broker và Clients.

Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận thông điệp từ publisher, xếp các thông điệp theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật thông điệp, lưu trữ, ...



Hình 2.2. Mô hình giao thức MQTT

Client thì được chia thành 2 nhóm là publisher và subscriber. Client là các software component hoạt động tại edge device nên chúng được thiết kế để có thể hoạt động một cách linh hoạt. Client chỉ làm ít nhất một trong 2 việc là publish các thông điệp lên một kênh cụ thể hoặc subscribe một kênh nào đó để nhận thông điệp.

### **2.2.3. Ưu điểm của MQTT**

- Chuyên thông tin hiệu quả hơn.
- Tăng khả năng mở rộng.
- Giảm đáng kể tiêu thụ băng thông mạng.
- Giảm tốc độ cập nhật xuống giây.
- Rất phù hợp cho điều khiển và đo lường.
- Tối đa hóa băng thông có sẵn.
- Chi phí cực rẻ.
- Rất an toàn với bảo mật dựa trên sự cho phép.
- Tiết kiệm thời gian đầu tư.

### **2.2.4. Message và Topic.**

Trong giao thức MQTT, message là thông điệp muốn gửi đi, có định dạng mặc định là plain-text nhưng có thể cấu hình thành định dạng khác.

Topic có thể coi như là một kênh logic giữa hai điểm là publisher và subscriber. Về cơ bản, khi message được publish vào một topic thì tất cả những subscriber của topic đó sẽ nhận được message này.

### **2.2.5. Publish và Subscribe.**

Trong một hệ thống sử dụng giao thức MQTT, nhiều MQTT Client kết nối tới một MQTT Server. Mỗi client sẽ đăng ký một vài topic. Quá trình đăng ký này gọi là subscribe, client sẽ lắng nghe thông tin từ những topic này.

Khi có một client gửi message lên topic nào đó thì được gọi là publish. Mỗi khi publish, message sẽ được lên MQTT Server và sau đó server sẽ gửi cho tất cả client subscribe topic đó.

### 2.2.6. Qualities of Service.

Qualities of Service (QoS) khi publish và subscribe:

- QoS0: Server/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi giao thức TCP/IP.
- QoS1: Server/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận được dữ liệu.
- QoS2: Server/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua bốn bước bắt tay.

**Kết luận:** Các thiết bị cảm biến và giao thức MQTT đều là những thiết bị và giao thức thông dụng, được sử dụng rộng rãi trong IoT, tuy có sai sót nhưng những thiết bị cơ bản đáp ứng được yêu cầu của đề tài.

### Chương 3: CÀI ĐẶT VÀ CẤU HÌNH THIẾT BỊ.

Cài đặt và cấu hình môi trường lập trình cho ESP8266 và Raspberry Pi bao gồm Arduino IDE, hệ điều hành Raspbian, PostgreSQL, MQTT Server, ...

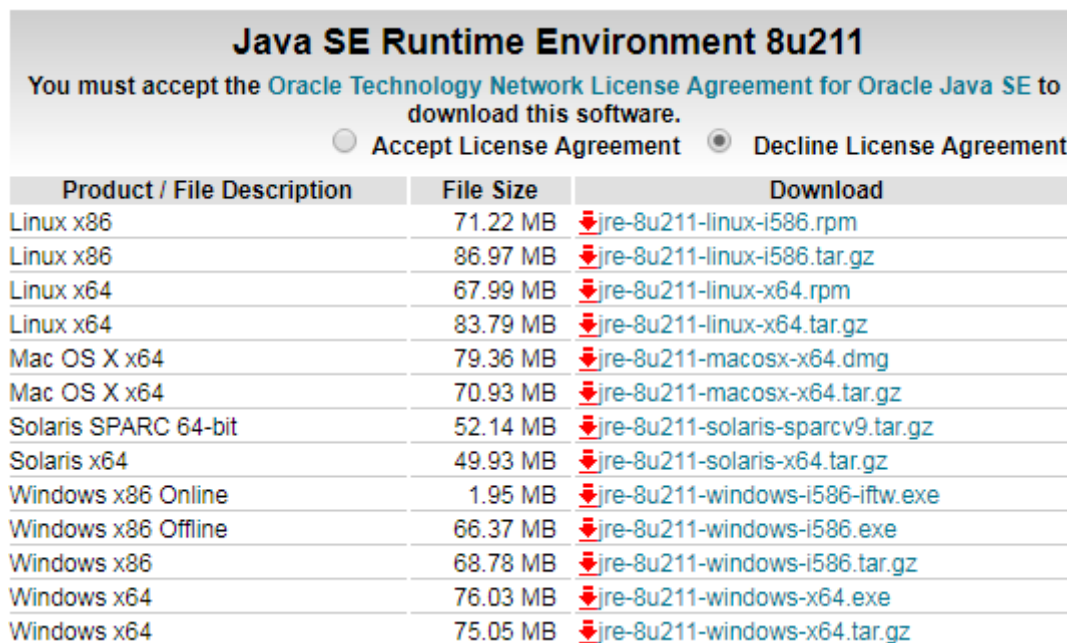
#### 3.1. Cài đặt và cấu hình Arduino.

Trước khi lập trình cho các dự án của Arduino, việc đầu tiên là cần cài đặt phần mềm lập trình và môi trường lập trình trên máy tính để phác thảo và tải lên bảng mạch của Arduino. Môi trường hệ điều hành được sử dụng là môi trường Windows và công cụ sử dụng cho lập trình là Arduino Intergrated Development Environment (IDE).

##### 3.1.1. Cài đặt JRE.

Arduino IDE được viết trên Java nên cần phải cài đặt Java Runtime Environment (JRE) trước khi cài đặt Arduino IDE.

Truy cập vào trang chủ của Oracle để tải về phiên bản JRE phù hợp: [www.oracle.com/technetwork/java/javase/downloads/index.html](http://www.oracle.com/technetwork/java/javase/downloads/index.html). Nhấn vào lựa chọn “Accept License Agreement” và chọn phiên bản phù hợp để tải xuống.



Java SE Runtime Environment 8u211		
You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux x86	71.22 MB	<a href="#">jre-8u211-linux-i586.rpm</a>
Linux x86	86.97 MB	<a href="#">jre-8u211-linux-i586.tar.gz</a>
Linux x64	67.99 MB	<a href="#">jre-8u211-linux-x64.rpm</a>
Linux x64	83.79 MB	<a href="#">jre-8u211-linux-x64.tar.gz</a>
Mac OS X x64	79.36 MB	<a href="#">jre-8u211-macosx-x64.dmg</a>
Mac OS X x64	70.93 MB	<a href="#">jre-8u211-macosx-x64.tar.gz</a>
Solaris SPARC 64-bit	52.14 MB	<a href="#">jre-8u211-solaris-sparcv9.tar.gz</a>
Solaris x64	49.93 MB	<a href="#">jre-8u211-solaris-x64.tar.gz</a>
Windows x86 Online	1.95 MB	<a href="#">jre-8u211-windows-i586-iftw.exe</a>
Windows x86 Offline	66.37 MB	<a href="#">jre-8u211-windows-i586.exe</a>
Windows x86	68.78 MB	<a href="#">jre-8u211-windows-i586.tar.gz</a>
Windows x64	76.03 MB	<a href="#">jre-8u211-windows-x64.exe</a>
Windows x64	75.05 MB	<a href="#">jre-8u211-windows-x64.tar.gz</a>

Hình 3.1. Tải xuống JRE.

Sau khi tải về file cài đặt tiến hành cài đặt trên máy theo các hướng dẫn trên màn hình.

### 3.1.2. Cài đặt Arduino IDE.

Truy cập vào trang chủ của Arduino để tải về phiên bản chương trình Arduino IDE phù hợp với hệ điều hành: [www.arduino.cc/en/Main/Software](http://www.arduino.cc/en/Main/Software).



Hình 3.2. Tải xuống Arduino.

Chọn “Windows Installer, for Windows XP and up” để tải xuống phiên bản cài đặt cho hệ điều hành Windows. Sau khi tải xuống, mở file cài đặt và lựa chọn các cài đặt sau đó nhấn next để cài đặt Arduino IDE.



Hình 3.3. Giao diện Arduino IDE.

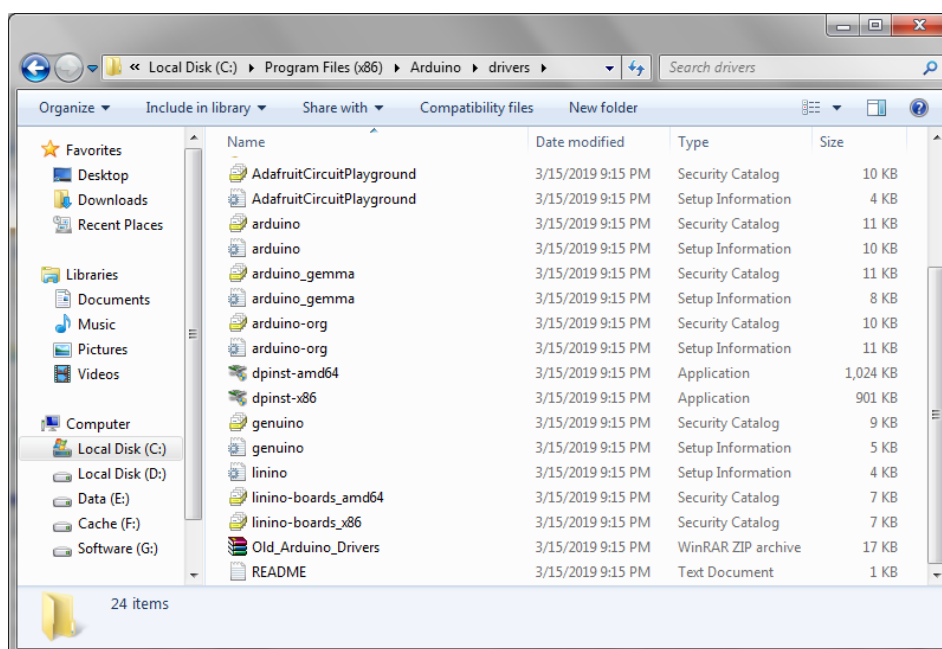
Giao diện của Arduino IDE bao gồm ba vùng chính:

- Vùng lệnh: gồm các nút lệnh (File, Edit, Sketch, Tools, Help) và các icon cho phép sử dụng nhanh các chức năng thường dùng.
- Vùng viết chương trình: nơi viết các đoạn mã cho bảng mạch.
- Vùng thông báo: chứa các thông báo.

### 3.1.3. Cài đặt Driver.

Sau khi đã cài đặt thành công Arduino IDE, tiếp theo cần cài đặt driver để thiết bị có thể được nhận dạng khi cắm vào máy tính.

Mở thư mục cài đặt Arduino trên máy tính và mở thư mục drivers.



Hình 3.4. Thư mục Arduino Drivers

Chạy file dpinst-amd64.exe cho Windows 64bit hoặc dpinst-x86 dành cho Windows 32bit.

### 3.1.4. Cấu hình Arduino IDE lập trình ESP8266.

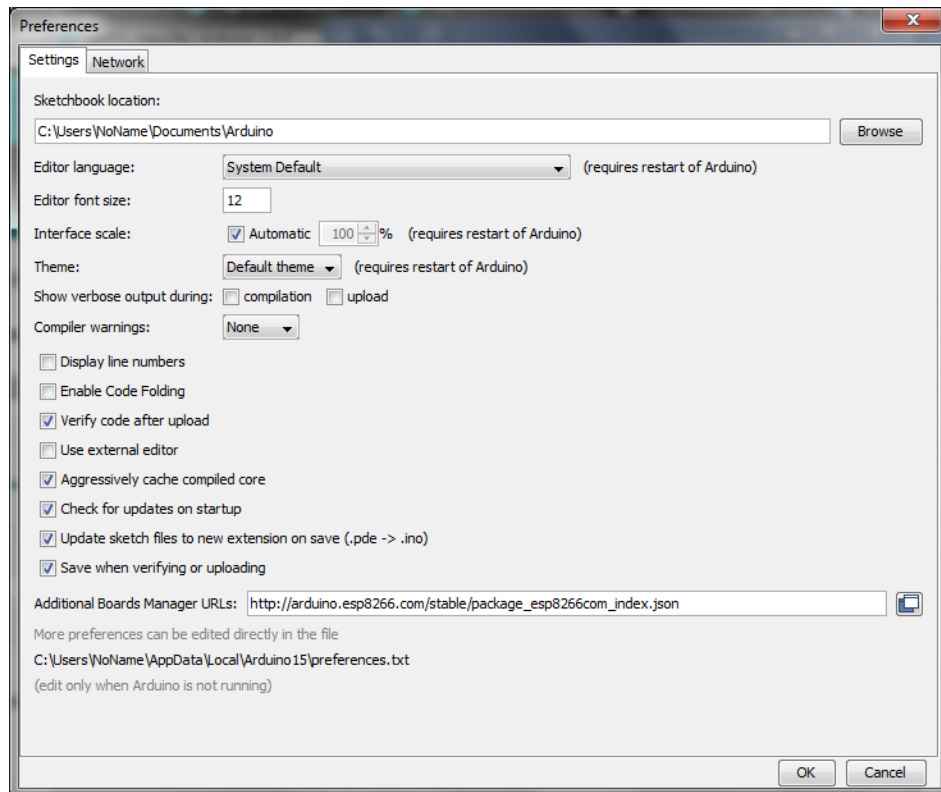
Để tiến hành cài đặt thư viện và chức năng nạp code cho IDE ta cần cấu hình thư viện để có thể tải về.

Bước 1: Vào **File -> Preferences**.



Bước 2: Thêm vào ô **Additional Board Manager URLs** đường dẫn bên dưới:

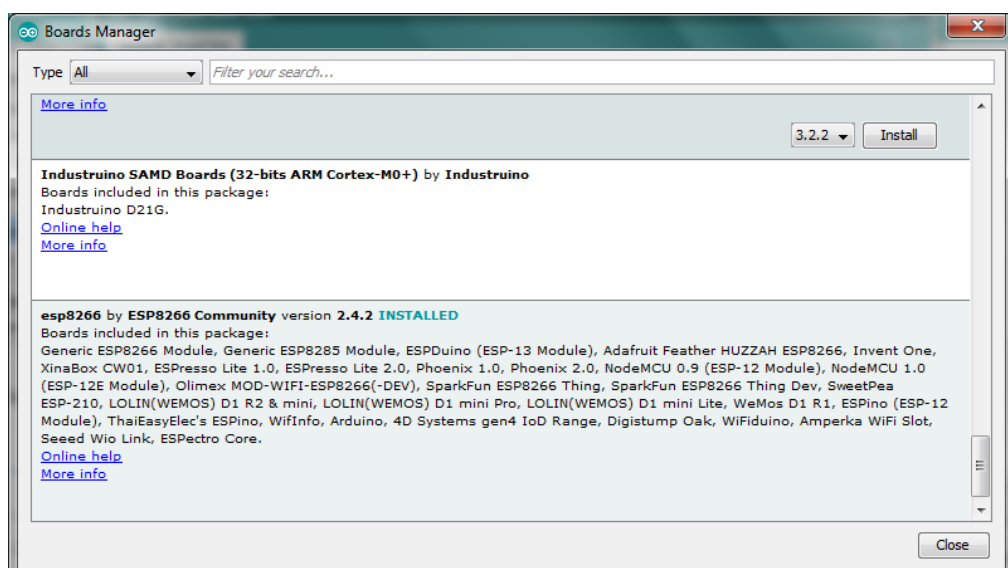
*[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)*



Hình 3.5. Cấu hình Arduino IDE.

Bước 3: Nhấn **OK**.

Bước 4: Vào **Tool -> Board -> Boards Manager**.



Hình 3.6. Cài đặt thư viện cho ESP8266.

Bước 5: Chọn thư viện **esp8266 by ESP8266 Community** và nhấn **Install**.

Bước 6: Chọn Board để lập trình cho ESP8266:

- Kết nối ESP8266 vào máy tính bằng cổng USB.
- Vào **Tool -> Board -> Generic ESP8266 Module**.
- Chọn cổng COM tương ứng với ESP8266 đã kết nối.

### 3.2. Cài đặt Raspberry.

Raspberry sử dụng thẻ nhớ để lưu trữ dữ liệu cũng như hệ điều hành của nó. Mặc định khi mua Raspberry Pi thì sẽ không bao gồm thẻ nhớ và hệ điều hành đi theo, nên chúng ta cần mua thêm thẻ nhớ ngoài và cài đặt hệ điều hành cho Raspberry Pi trên chiếc thẻ nhớ đó. Yêu cầu thẻ nhớ là loại classic 10 và dung lượng tối thiểu là 8GB. Thẻ nhớ được sử dụng có dung lượng 16GB để có thể đáp ứng nhu cầu lưu trữ dữ liệu.

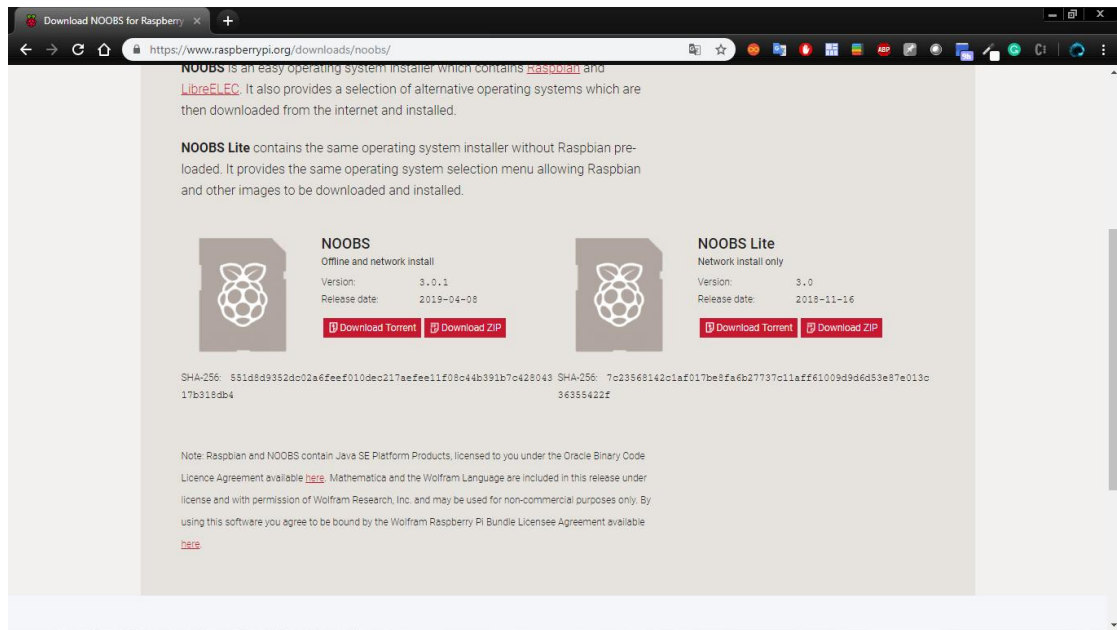
Hệ điều hành thông dụng cho Raspberry Pi:

- Raspbian: cơ bản, phổ biến nhất và do chính Raspberry Pi Foundation cung cấp. Được hàng khuyến cáo sử dụng cho người mới bắt đầu làm quen với Raspberry Pi.
- Ubuntu Mate: tương tự như Raspbian nhưng có giao diện bắt mắt hơn.
- Windows 10 IoT Core: chỉ có nhân Windows, được sử dụng cho mục đích phát triển các ứng dụng IoT.

Dựa theo mục đích sử dụng cũng như hiểu biết, nhóm sử dụng hệ điều hành Raspbian cho Raspberry Pi.

#### 3.2.1. Hệ điều hành Rasbian.

Người mới bắt đầu nên bắt đầu với New Out of the Box Software (NOOBS). Truy cập vào trang chủ của Raspberry Pi để tải xuống phiên bản cài đặt NOOBS mới nhất: [www.raspberrypi.org/downloads/noobs](http://www.raspberrypi.org/downloads/noobs).

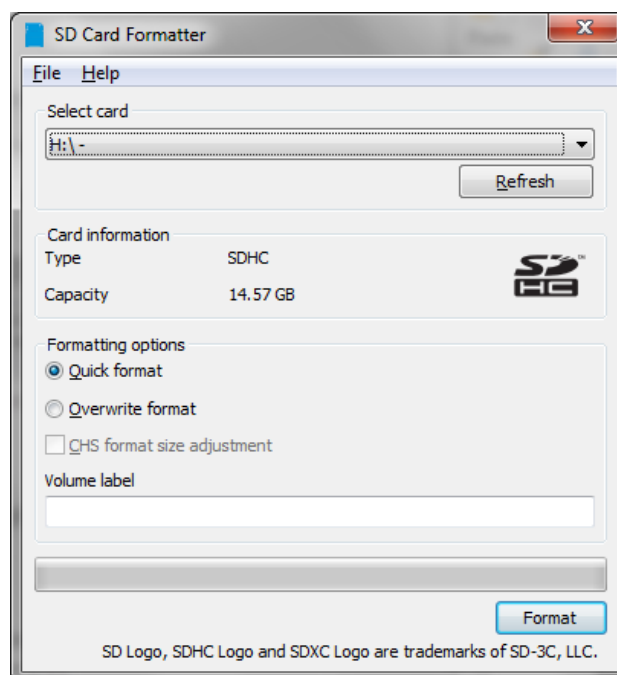


Hình 3.7. Tải xuống NOOBS.

Sau khi tải, giải nén ra một thư mục riêng.

Tiếp theo tải phần mềm SD Formatter 4.0 và làm theo từng bước để cài đặt phần mềm này: [www.sdcard.org/downloads/formatter](http://www.sdcard.org/downloads/formatter).

Sau khi đã cài đặt xong, tiến hành gắn thẻ nhớ vào máy tính. Mở phần mềm SD Formatter lên và lựa chọn định dạng lại thẻ nhớ.



Hình 3.8. Phần mềm SD Formatter.

Cuối cùng copy toàn bộ dữ liệu đã giải nén ở trên vào thẻ nhớ. Sau đó gắn thẻ nhớ và khởi động Raspberry Pi. Hệ điều hành sẽ tự động cài đặt khi khởi động. Yêu cầu Raspberry Pi phải được cắm dây mạng và có màn hình để thao tác. Khi khởi động sẽ có các lựa chọn và cài đặt để người dùng lựa chọn cài đặt cho Raspbian.

Mặc định người dùng khi của Raspbian:

- ❖ Tên người dùng: pi.
- ❖ Mật khẩu: raspberry.

### 3.2.2. Cấu hình Raspbian.

Raspberry Pi không có màn hình nên khi cài đặt hệ điều hành cần màn hình nhưng khi đã có hệ điều hành ta có thể sử dụng Secure Socket Shell (SSH) hoặc Remote Desktop để kết nối và lập trình trên Raspberry Pi.

Trước khi sử dụng cũng như cài đặt phần mềm hay thư viện trên Raspbian cần phải cập nhật cho hệ điều hành bằng lệnh sau:

```
$ sudo apt-get update
```

#### a. SSH.

SSH là một giao thức mạng dùng để thiết lập kết nối mạng một cách bảo mật, cho phép thao tác giữa máy chủ và máy khách, sử dụng cơ chế mã hóa đủ mạnh nhằm ngăn chặn các hiện tượng nghe trộm, đánh cắp thông tin trên đường truyền. Các công cụ SSH phổ biến như PuTTY, OpenSSH, Git Bash cung cấp giao diện thân thiện để thiết lập một kết nối được mã hóa.

Mặc định SSH đã được bật nhưng nếu chưa bật thì ta phải tự bật bằng cách mở Terminal và gõ lệnh:

```
$ sudo raspi-config
```

Xuất hiện hộp thoại Config trên Terminal, chọn ssh. Màn hình sẽ xuất hiện thông báo “Would you like the SSH server enabled or disabled?”. Chọn Enable để bật SSH.

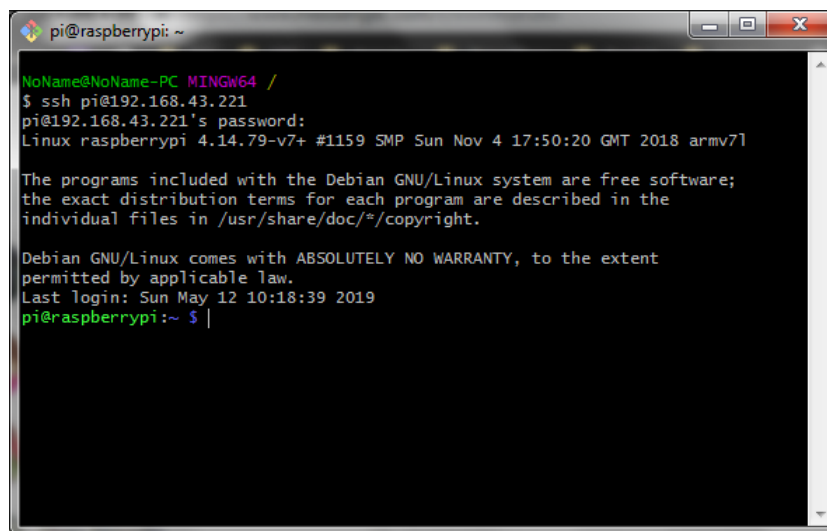
Để kết nối SSH tới Raspberry Pi thì máy tính cần phải cài đặt SSH hoặc cài đặt công cụ như PuTTY, Git Bash, ... Tiếp theo cần có địa chỉ IP của Raspberry Pi, mở Terminal và gõ lệnh:

*\$ ifconfig*

Kết nối SSH thông qua Git Bash sử dụng câu lệnh:

*\$ ssh <tên người dùng>@<địa chỉ IP của Raspberry Pi>*

Nhập mật khẩu và Enter.



Hình 3.9. Kết nối Raspberry Pi qua SSH.

### **b. Remote Desktop.**

Remote Desktop đề cập đến khả năng kết nối máy tính từ xa đến máy tính khác và kiểm soát màn hình. Remote Desktop thường được truy cập thông qua cổng 3389 và sử dụng phần mềm đi kèm với Windows hoặc một số chương trình của bên thứ ba như TeamViewer, PC Anywhere, ...

Trên các phiên bản hệ điều hành Windows đều được trang bị sẵn tính năng Remote Desktop. Để sử dụng tính năng này đầu tiên phải kích hoạt chúng trước. Tuy nhiên trên Raspbian thì cần phải cài đặt để có thể sử dụng.

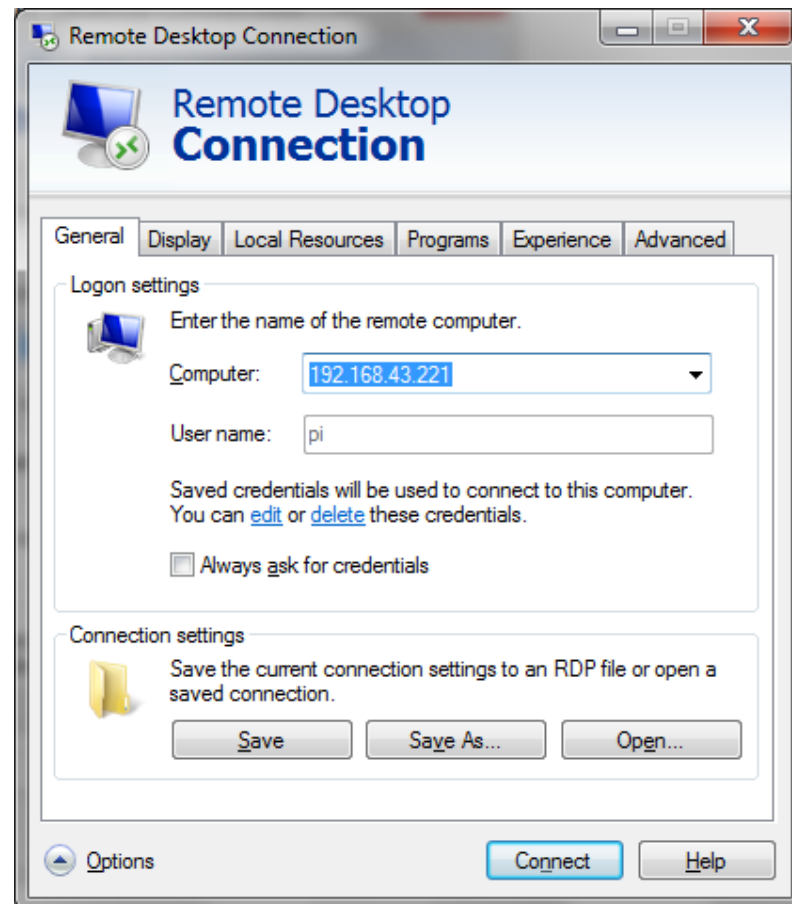
Các bước cài đặt Remote Desktop trên Raspbian:

Bước 1: mở Terminal và gõ lệnh:

*\$ sudo apt-get install xrdp*

Bước 2: Sau khi cài đặt xong, khởi động lại Raspberry Pi.

Bước 3: Đối với máy tính Windows, mở chương trình Remote Desktop lên và nhập địa chỉ IP cũng như tên và mật khẩu người dùng để truy cập tới Raspberry Pi.



Hình 3.10. Phần mềm Remote Desktop.

### 3.2.3. Cài đặt môi trường lập trình.

Raspberry Pi đã được cấu hình đầy đủ ở các bước trên, tiếp theo cần cài đặt và cấu hình môi trường để có thể lập trình trên Raspberry Pi.

#### a. Python.

Hệ thống sẽ được thiết kế phía Back-End sử dụng Flask framework nên cần cài đặt Python 3.x vì Python 2.x hiện không hỗ trợ một số thư viện của Flask. Và Python 3.x đang được nhà sản xuất khuyến cáo vì tương lai Python 2.x sẽ không còn được hỗ trợ.

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum. Rất dễ dàng tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình.

Mặc định Raspbian đã được cài đặt sẵn Python 2.x, việc gỡ bỏ hoàn toàn Python 2.x sẽ ảnh hưởng đến hệ thống nên Python 3.x sẽ được cài đặt song song với Python 2.x. Mở Terminal và gõ lệnh bên dưới để cài đặt:

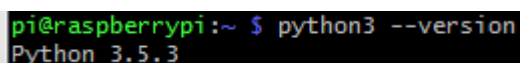
```
$ sudo apt-get install python3
```

Cài đặt pip để cài đặt thư viện cho Python:

```
$ sudo apt-get install python3-pip
```

Kiểm tra phiên bản của Python và pip:

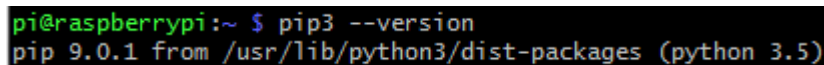
```
$ python3 --version
```



```
pi@raspberrypi:~ $ python3 --version  
Python 3.5.3
```

Hình 3.11. Python3 version.

```
$ pip3 --version
```



```
pi@raspberrypi:~ $ pip3 --version  
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.5)
```

Hình 3.12. Pip3 version.

## **b. Angular.**

Angular là một Frameworks JavaScript giúp chúng ta xây dựng ứng dụng đầy đủ tính năng từ phía Client. Được phát triển trên nền tảng JavaScript của Google, kế thừa các đặc điểm của Angular JS và phát triển một phương thức tiếp cận xây dựng ứng dụng hoàn toàn mới.

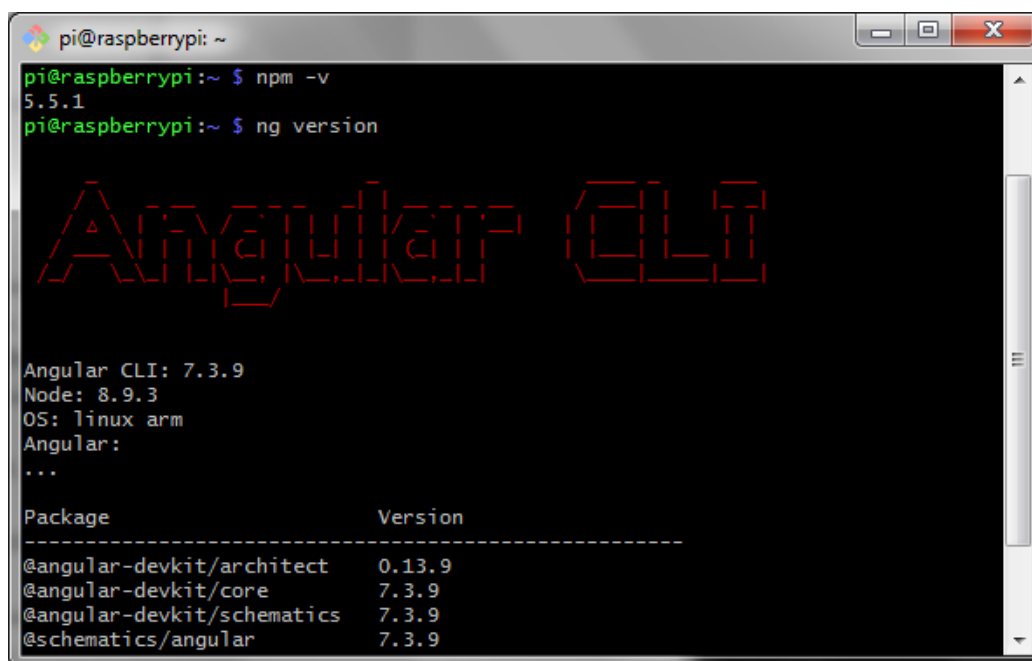
Để cài đặt được Angular trên Raspbian, đòi hỏi trên hệ điều hành phải cài Node.js và npm:

```
$ sudo apt-get install nodejs npm
```

Tiếp theo ta cài đặt Angular thông qua npm:

```
$ sudo npm install -g @angular/cli
```

Sau khi cài đặt xong kiểm tra phiên bản npm:



```
pi@raspberrypi: ~  
pi@raspberrypi:~$ npm -v  
5.5.1  
pi@raspberrypi:~$ ng version  
  
Angular CLI  
  
Angular CLI: 7.3.9  
Node: 8.9.3  
OS: linux arm  
Angular:  
...  
  
Package                                Version  
-----  
@angular-devkit/architect              0.13.9  
@angular-devkit/core                   7.3.9  
@angular-devkit/schematics             7.3.9  
@schematics/angular                    7.3.9
```

Hình 3.13. Phiên bản NPM và Angular.

### c. Git.

Git là một hệ thống quản lý phiên bản phân tán (distributed version control system). Là một công cụ quản lý code và làm việc nhóm hiệu quả. Chức năng chính của Git ở đây được xem như nơi lưu trữ mã nguồn và tải xuống mã nguồn cho Raspberry Pi.

Cài đặt Git trên Raspberry Pi: *\$ sudo apt-get install git.*

### 3.2.4. Cơ sở dữ liệu PostgreSQL.

PostgreSQL là một hệ thống quản trị cơ sở dữ liệu quan hệ đối tượng có mục đích chung, hệ thống cơ sở dữ liệu mã nguồn mở tiên tiến nhất hiện nay. Được thiết kế để chạy trên các nền tảng tương tự UNIX. Tuy nhiên, PostgreSQL cũng được điều chỉnh linh động để có thể chạy trên nhiều nền tảng khác nhau như Mac OS và Windows.

Lý do sử dụng PostgreSQL vì đây là một phần mềm mã nguồn mở miễn phí và không yêu cầu quá nhiều công tác bảo trì bởi có tính ổn định cao. Do đó, nếu phát triển các ứng dụng dựa trên PostgreSQL, chi phí sẽ thấp hơn so với các hệ thống quản trị dữ liệu khác.



Tải xuống PostgreSQL và các phụ thuộc đi kèm:

```
$ sudo apt-get install postgresql postgresql-contrib
```

Sau khi cài đặt xong, khởi động PostgreSQL:

```
$ sudo systemctl start postgresql
```

Đăng nhập với người dùng mặc định của PostgreSQL là postgres và đặt mật khẩu cho postgres:

```
$ sudo -u postgres psql
```

```
$ ALTER USER postgres PASSWORD <mật khẩu>;
```

### 3.2.5. MQTT Server.

Rasbian hỗ trợ thư viện Mosquitto đóng vai trò như một MQTT Server.

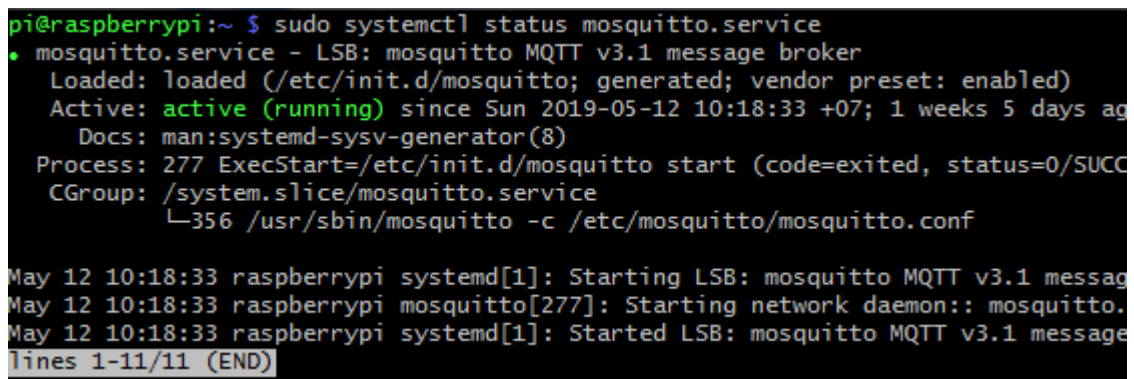
Tải xuống và cài đặt Mosquitto:

```
$ sudo apt-get install mosquitto mosquitto-clients
```

Sau khi đã cài đặt thành công, khởi động Mosquitto:

```
$ sudo systemctl start mosquitto.service
```

Kiểm tra Mosquitto có chạy hay không:



```
pi@raspberrypi:~$ sudo systemctl status mosquitto.service
• mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated; vendor preset: enabled)
   Active: active (running) since Sun 2019-05-12 10:18:33 +07; 1 weeks 5 days ago
     Docs: man:systemd-sysv-generator(8)
  Process: 277 ExecStart=/etc/init.d/mosquitto start (code=exited, status=0/SUCCESS)
    CGroup: /system.slice/mosquitto.service
            └─356 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

May 12 10:18:33 raspberrypi systemd[1]: Starting LSB: mosquitto MQTT v3.1 message broker: mosquitto.
May 12 10:18:33 raspberrypi mosquitto[277]: Starting network daemon:: mosquitto.
May 12 10:18:33 raspberrypi systemd[1]: Started LSB: mosquitto MQTT v3.1 message broker: mosquitto.
lines 1-11/11 (END)
```

Hình 3.14. Kiểm tra Mosquitto.

**Kết luận:** ESP8266 và Raspberry đã được cài đặt môi trường và cấu hình đầy đủ cho việc phát triển ứng dụng.

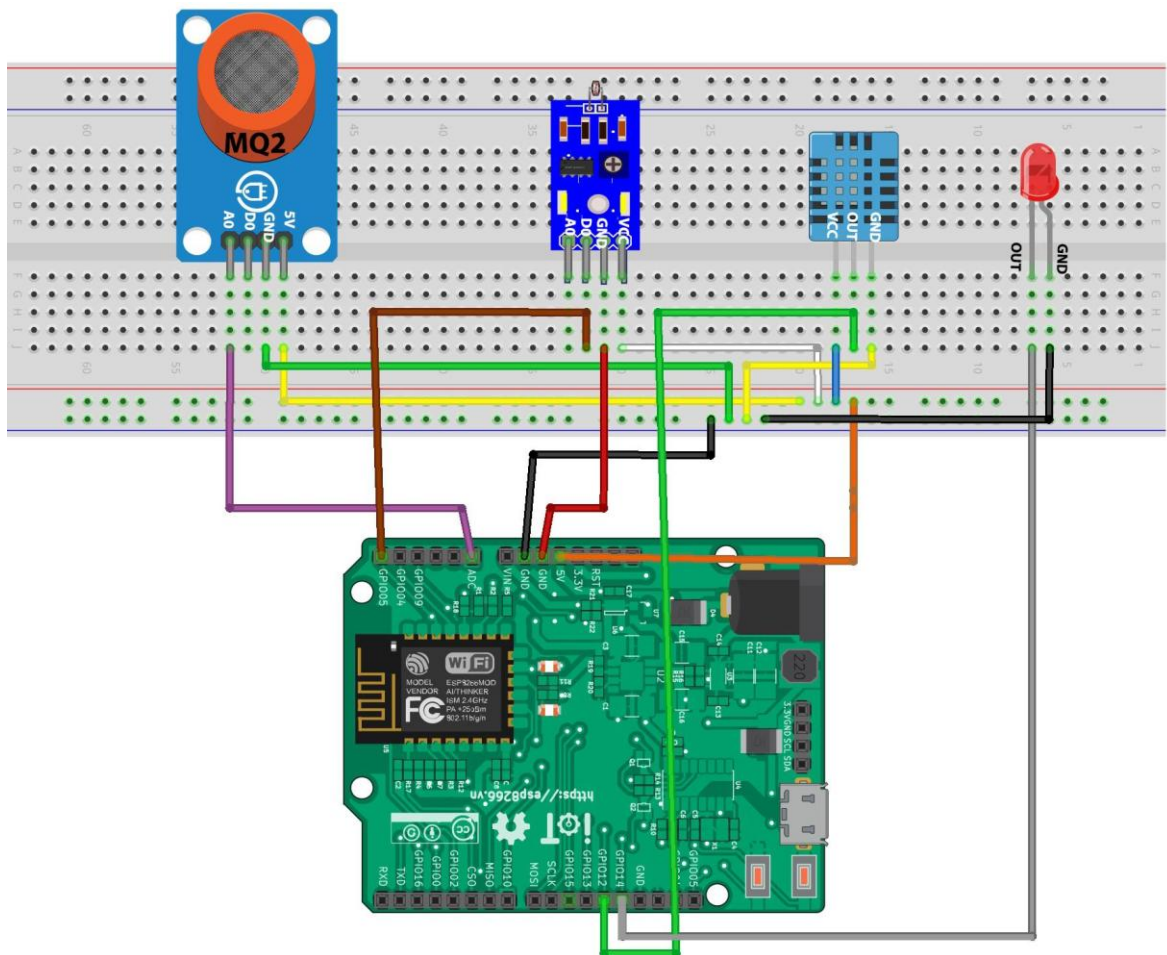
## Chương 4: XÂY DỰNG ỨNG DỤNG.

Các bước thiết kế, kết nối các thiết bị, xây dựng mô hình ứng dụng và kiểm thử ứng dụng.

### 4.1. Thiết kế.

#### 4.1.1. Mô hình kết nối cảm biến với ESP8266.

Các thiết bị cảm biến bao gồm cảm biến nhiệt độ, độ ẩm, cảm biến ánh sáng, cảm biến khí GAS và đèn LED sẽ được kết nối với ESP8266 thông qua dây cắm Breadboard và bảng mạch điện theo mô hình bên dưới:



Hình 4.1. Mô hình kết nối thiết bị.

Thông số các cổng kết nối:

Thiết bị	Cổng của thiết bị	Cổng của ESP8266
Cảm biến nhiệt độ, độ ẩm	VCC(+)	5V
	GND(-)	GND
	OUT	GPIO12
Cảm biến ánh sáng	VCC	5V
	GND	GND
	D0	GPIO05
Cảm biến khí GAS	VCC	5V
	GND	GND
	A0	ADC
Đèn LED	GND	GND
	OUT	GPIO14

Bảng 4.1. Các cổng kết nối thiết bị với ESP8266.

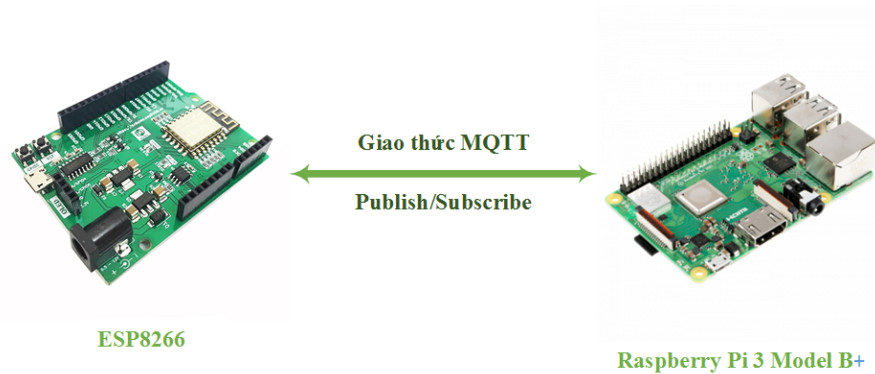
Định nghĩa các kênh truyền MQTT và WebSocket cho thiết bị:

Thiết bị	Giao thức MQTT	Giao thức WebSocket
Cảm biến nhiệt độ	LIVINGROOM/TEMPERATURE	temperature
Cảm biến độ ẩm	LIVINGROOM/HUMIDITY	humidity
Cảm biến khí GAS	LIVINGROOM/GAS	gas
Cảm biến ánh sáng	LIVINGROOM/LIGHT	light
Đèn LED	LIVINGROOM/FLASH_LIGHT	flashLight
	LIVINGROOM/FLASH_LIGHT/CONTROL	

Bảng 4.2. Kênh truyền thiết bị.

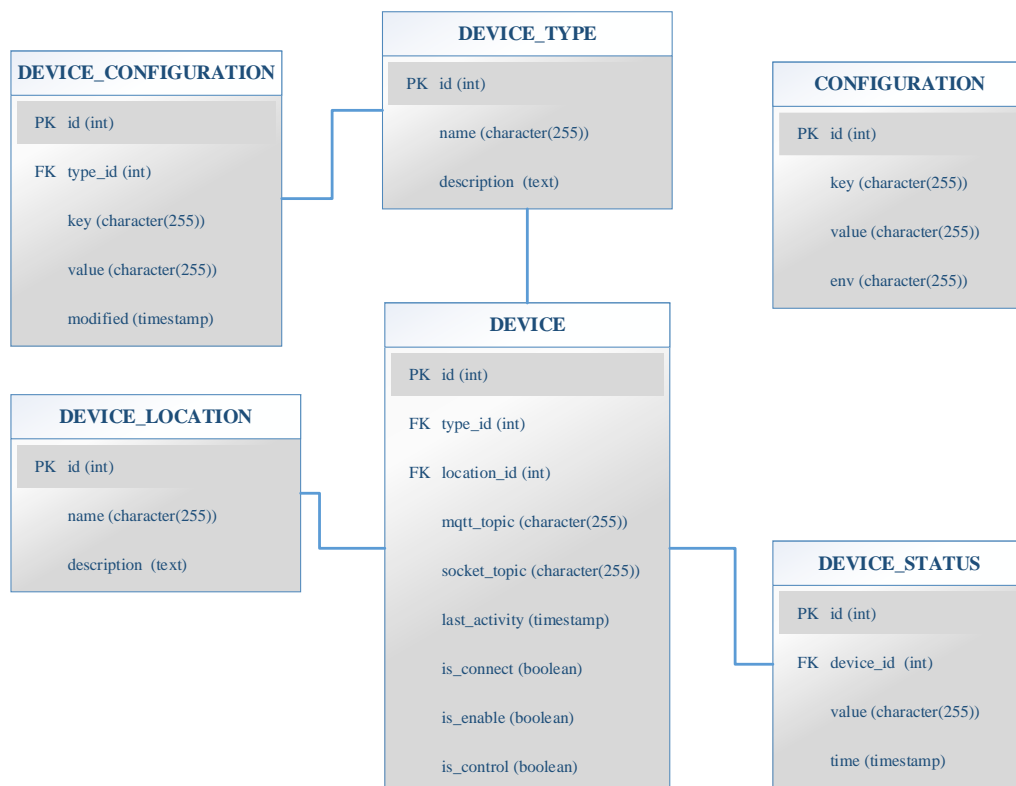
#### 4.1.2. Mô hình kết nối giữa ESP8266 và Raspberry Pi.

ESP8266 sẽ kết nối với Raspberry Pi thông qua giao thức MQTT để nhận và gửi dữ liệu bằng việc Publish và Subscribe. Để có thể nói giao tiếp với nhau thì cả hai thiết bị này phải kết nối chung một Access Point để nhận biết được nhau.



Hình 4.2. Kết nối giữa ESP8266 và Raspberry Pi.

#### 4.1.3. Cấu trúc cơ sở dữ liệu.

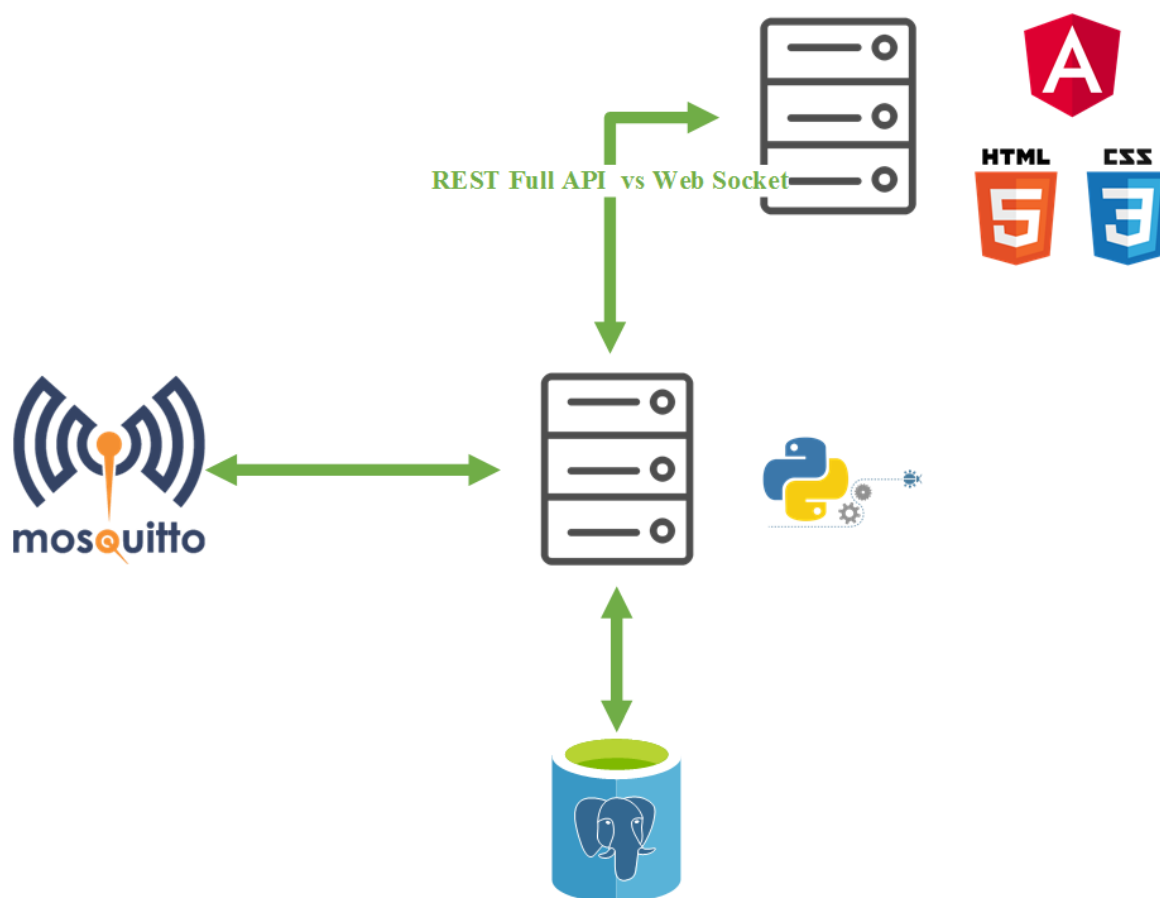


Hình 4.3. Cấu trúc cơ sở dữ liệu.

Nhằm đáp ứng khả năng mở rộng thiết bị và ứng dụng trong tương lai nên cơ sở dữ liệu được thiết kế để phù hợp cho việc thêm các thiết bị cảm biến khác và lưu trữ các thông tin cơ bản của thiết bị.

#### 4.1.4. Mô hình Web Server.

Web Server sẽ được triển khai trên Raspberry Pi theo hướng Microservice. Bao gồm bốn phần chính:



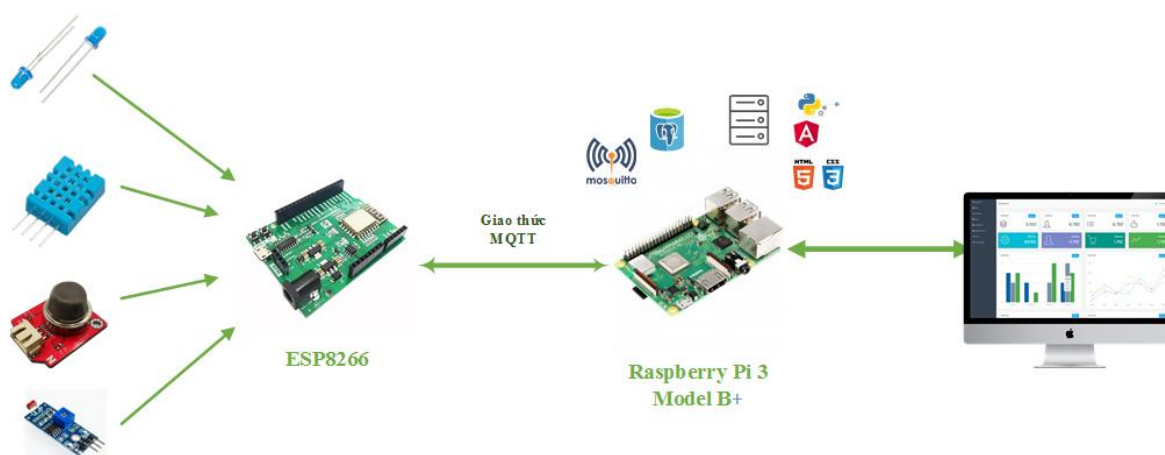
Hình 4.4. Mô hình Web Server trên Raspberry Pi.

- MQTT Server: là Mosquitto, có nhiệm vụ nhận và gửi dữ liệu giữa ESP8266 và Raspberry Pi.
- Database PostgreSQL: lưu trữ dữ liệu.
- Back end: sử dụng ngôn ngữ lập trình Python, nhiệm vụ là nhận dữ liệu được gửi lên từ ESP8266 thông qua MQTT Server và xử lý, lưu trữ xuống cơ sở dữ liệu và giao tiếp với front end.

- Front end: sử dụng ngôn ngữ lập trình Angular và HTML, CSS. Nhiệm vụ chính là hiển thị giao diện web site cho người dùng và thao tác trên đó.

Khi có dữ liệu được gửi lên từ ESP8266, MQTT Server sẽ nhận message đó và gửi cho tất cả những client đã subscribe topic đó. Back end subscribe topic và khi nhận được message sẽ xử lý dữ liệu đó bao gồm lưu trữ, gửi cho front end qua web socket. Khi có message gửi lên front end sẽ hiển thị message đó theo thời gian thực.

#### 4.1.5. Mô hình tổng quát.



Hình 4.5. Mô hình tổng quát.

## 4.2. Xây dựng và triển khai.

### 4.2.1. Lập trình ESP8266.

Để sử dụng được Wifi và MQTT trên ESP8266, ta cần import thư viện của ESP8266 vào code và khai báo đối tượng:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient espClient;
PubSubClient client(espClient);
```

Cấu hình để ESP8266 kết nối vào mạng wifi và kết nối tới MQTT Server:

```
// Update these with values suitable for your network.
const char* ssid = ""; // WiFi username
const char* password = ""; // WiFi password
const char* mqtt_server = ""; // MQTT address
const char* userName = ""; // MQTT username
const char* passWord = ""; // MQTT password
```

Hình 4.6. Thông số cấu hình Wifi của ESP8266.

Thông số:

- ssid: tên Wifi mà chúng ta kết nối tới, tối đa 32 ký tự.
- password: mật khẩu của Wifi, từ 8 đến 64 ký tự.
- mqtt\_server: địa chỉ IP của Raspberry Pi.
- userName: tên truy cập MQTT Server.
- password: mật khẩu truy cập MQTT Server.

Sau khi đã cấu hình cho ESP8266, tiếp theo cần kết nối tới Wifi:

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
```

Hình 4.7. ESP8266 kết nối Wifi.

Theo mặc định ESP8266 sẽ cố kết nối lại đến mạng Wifi sau khi bị disconnect. Nhưng để đảm bảo việc kết nối không gặp lỗi nên code sẽ kiểm tra nếu chưa kết nối được thì sẽ đợi 0,5s để thực hiện việc kết nối lại. Tiếp theo là kết nối đến MQTT Server trên Raspberry Pi.

```
client.setServer(mqtt_server, 1883); // connect to MQTT on gate 1883
client.setCallback(callback); //Callback executed when message received
```

Hình 4.8. ESP8266 kết nối MQTT Server.

ESP8266 sẽ kết nối tới MQTT Server thông qua địa chỉ IP của Raspberry Pi và cổng mặc định là 1883. Hàm callback sẽ được gọi khi Raspberry publish message lên topic mà ESP8266 sẽ subscribe. Callback có chức năng bật/tắt đèn:

```

void callback(char* topic, byte* payload, unsigned int length) {
  if ((char)payload[0] == '1') {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}

```

Hình 4.9. Hàm bật/tắt đèn.

Tiếp theo cần lập trình cho ESP8266 đọc dữ liệu digital từ các cảm biến và publish lên các kênh đã định nghĩa trước đó qua câu lệnh:

*client.publish(<topic>, <message>);*

Trong đó topic là kênh truyền đã được thiết kế trước đó và message là giá trị của cảm biến. Ngoại trừ cảm biến nhiệt độ, độ ẩm các cảm biến và thiết bị khác sẽ được định nghĩa là 0 và 1.

Thiết bị cảm biến	Giá trị trả về
Cảm biến nhiệt độ	Từ 0 đến 100
Cảm biến độ ẩm	Từ 0 đến 100
Cảm biến ánh sáng	Có ánh sáng: 0 Không có ánh sáng: 1
Cảm biến khí GAS	Không có khí GAS: 0 Có khí GAS: 1
Đèn LED	Tắt: 0 Bật: 1

Bảng 4.3. Định nghĩa giá trị trả về của thiết bị.

Nhằm đảm bảo giá trị được trả về liên tục và không xuất hiện lỗi khi kết nối nên chương trình sẽ có thời gian trễ là 5s:

*delay(5000);*

Quá trình lập trình hoàn thành, tiến hành kết nối ESP8266 vào máy tính và nạp code. Việc nạp code có thể mất từ 1 đến 3 phút. Sau khi nạp code xong ngắt kết nối và cấp điện để ESP8266 hoạt động với đoạn code vừa nạp.



#### 4.2.2. Tạo cơ sở dữ liệu và bảng.

Mở PostgreSQL bằng lệnh:

```
$ sudo -u postgres psql
```

Nhập mật khẩu đã tạo để đăng nhập và chạy lệnh sau để tạo database.

```
CREATE DATABASE <database name>
WITH
OWNER = postgres
ENCODING = 'UTF8'
CONNECTION LIMIT = -1;
```

Hình 4.10. Create\_database\_script.sql.

Tiếp theo, copy và chạy script để tạo các bảng cùng các ràng buộc đã thiết kế trong thư mục `~/smart-home/deploy/create_table_script.sql`.

```
-- Drop Table
DROP TABLE IF EXISTS public.device_configuration;
DROP TABLE IF EXISTS public.device_status;
DROP TABLE IF EXISTS public.device;
DROP TABLE IF EXISTS public.device_type;
DROP TABLE IF EXISTS public.device_location;
DROP TABLE IF EXISTS public.configuration;

-- Drop Sequence
DROP SEQUENCE IF EXISTS public.device_configuration_id_seq;
DROP SEQUENCE IF EXISTS public.device_status_id_seq;
DROP SEQUENCE IF EXISTS public.device_id_seq;
DROP SEQUENCE IF EXISTS public.device_type_id_seq;
DROP SEQUENCE IF EXISTS public.device_location_id_seq;
DROP SEQUENCE IF EXISTS public.configuration_id_seq;

-- Create Table
CREATE TABLE public.device_configuration (
    id INTEGER NOT NULL,
    type_id INTEGER NOT NULL,
    key CHARACTER VARYING(255) NOT NULL,
    value CHARACTER VARYING(255) NOT NULL,
    created TIMESTAMP WITHOUT TIME ZONE,
    modified TIMESTAMP WITHOUT TIME ZONE
);

CREATE TABLE public.device_status (
    id INTEGER NOT NULL,
    device_id INTEGER NOT NULL,
    value CHARACTER VARYING(255) NOT NULL,
    time TIMESTAMP WITHOUT TIME ZONE
```

Hình 4.11. Create\_table\_script.sql.

Chạy file *create\_data\_static.sql* để tạo dữ liệu ban đầu.

```
INSERT INTO public.device_type(id, name, description)
VALUES (1,'TEMPERATURE', 'Nhiệt độ');
INSERT INTO public.device_type(id, name, description)
VALUES (2,'HUMIDITY', 'Độ ẩm');
INSERT INTO public.device_type(id, name, description)
VALUES (3,'LIGHT', 'Độ sáng');
INSERT INTO public.device_type(id, name, description)
VALUES (4,'GAS', 'Khí gas');
INSERT INTO public.device_type(id, name, description)
VALUES (5,'FLASH_LIGHT', 'Trạng thái bóng đèn');

INSERT INTO public.device_location(id, name, description)
```

Hình 4.12. Create\_data\_static.sql.

Sau khi đã hoàn tất việc tạo cơ sở dữ liệu chạy lệnh `\d` để kiểm tra lại.

Các file script để tạo database và table cùng các dữ liệu ban đầu đều nằm trong thư mục `~/smart-home/deploy`.

#### 4.2.3. Kết nối cơ sở dữ liệu:

Sử dụng Python để kết nối với cơ sở dữ liệu PostgreSQL cần cài đặt thư viện Flask-SQLAlchemy.

```
$ sudo pip3 install flask-sqlalchemy
```

Sau khi đã cài đặt xong thư viện, ta tiến hành import thư viện vào code và cấu hình các thông tin cơ bản của cơ sở dữ liệu đã tạo trước đó trong file `~/smart-home/src/sh-service/setting.py`:

```
class DatabaseConfiguration:
    USER_NAME = 'postgres'
    PASSWORD = <mật khẩu>
    NAME = <tên cơ sở dữ liệu>
    HOST = 'localhost'
    PORT = 5432
    POSTGRES = {
        'user': USER_NAME,
        'pw': PASSWORD,
        'db': NAME,
        'host': HOST,
        'port': PORT,
    }
    URI = 'postgresql://%(user)s:%(pw)s@%(host)s:%(port)s/%(db)s' % POSTGRES
    TRACK_MODIFICATION = False
    ECHO = False
```

Bảng 4.4. Cấu hình kết nối cơ sở dữ liệu.

#### 4.2.4. Kết nối MQTT Server.

Cài đặt thư viện Flask-Mqtt để có thể kết nối và giao tiếp với MQTT server.

```
$ sudo pip3 install flask-mqtt
```

Cấu hình để có thể kết nối tới Mosquitto trong file `~/smart-home/src/sh-service/setting.py`:

```
class MQTTConfiguration:
    CLIENT_NAME = "Raspberry-Server-1"
    SERVER = 'localhost'
    PORT = 1883
    USER_NAME = 'pi'
    PASSWORD = <Mật khẩu của raspberry>
    KEEP_ALIVE = 60
    TLS_ENABLED = False
    TOPIC_TEMPERATURE = "LIVINGROOM/TEMPERATURE"
    TOPIC_HUMIDITY = "LIVINGROOM/HUMIDITY"
    TOPIC_LIGHT = "LIVINGROOM/LIGHT"
    TOPIC_GAS = "LIVINGROOM/GAS"
    TOPIC_FLASH_LIGHT = "LIVINGROOM/FLASH_LIGHT"
    TOPIC_FLASH_LIGHT_CONTROL = "LIVINGROOM/FLASH_LIGHT/CONTROL"
```

Bảng 4.5. Cấu hình kết nối MQTT Server.

#### 4.2.5. Thu thập và xử lý dữ liệu.

Hoàn tất cấu hình cho cơ sở dữ liệu và MQTT thì tiếp theo cần xử lý việc nhận dữ liệu. MQTT Server sẽ nhận dữ liệu của ESP8266 gửi lên Raspberry thông qua các topic đã được định nghĩa nên ta cần subscribe các topic đó để nhận dữ liệu.

```
@mqtt.on_message()
def handle_mqtt_message(client, userdata, message):
    topic = message.topic
    payload = message.payload.decode()
```

Bảng 4.6. Nhận message từ MQTT Server.

Phân loại topic, lưu vào cơ sở dữ liệu và gửi lên UI thông qua WebSocket.

```
if topic == MQTTConfiguration.TOPIC_TEMPERATURE:
    save_device(payload, 1)
    SocketIoService.send_message("temperature", payload)
```

Bảng 4.7. Xử lý dữ liệu nhiệt độ.

```

def save_device(value, device_id):
    try:
        device_status = DeviceStatus(device_id, value, datetime.now())
        with app.app_context():
            db.session.add(device_status)
            db.session.commit()
    except Exception as ex:
        print(ex)

```

Bảng 4.8. Lưu trữ dữ liệu vào cơ sở dữ liệu.

#### 4.2.6. Xây dựng giao diện lập trình ứng dụng.

Giao diện lập trình ứng dụng (API) là giao diện cho phép trao đổi dữ liệu giữa các ứng dụng khác nhau. Angular sẽ lấy dữ liệu của cơ sở dữ liệu thông qua Python sử dụng giao thức kết nối qua API.

Thông số API:

Phương thức	URL	Chức năng
GET	/device-status/<device_id>	Lấy dữ liệu một thiết bị gần nhất.
	/device-status/chart/<device_id>	Lấy dữ liệu 10 thiết bị gần nhất.
	/device-status/analytic/<time>	Lấy dữ liệu phân tích nhiệt độ, độ ẩm.
	/device/flash-light/control/on	Bật đèn.
	/device/flash-light/control/off	Tắt đèn

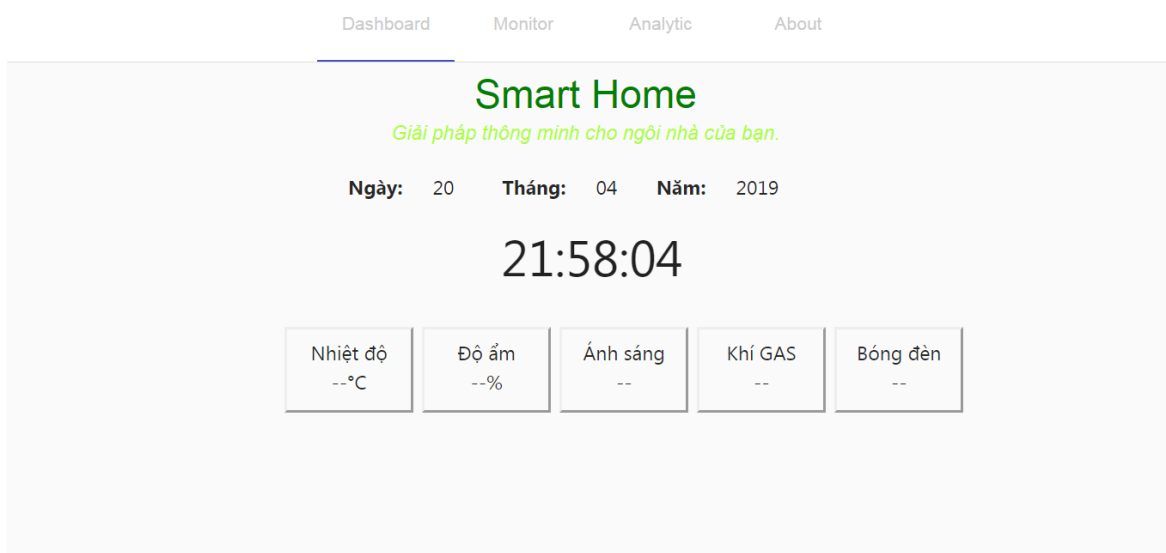
Bảng 4.9. Thông số API.

Kiểu dữ liệu trả về của tất cả API là application/json. Một kiểu dữ liệu phổ biến và thông dụng được sử dụng trong lập trình ứng dụng. Về vấn đề bảo mật, hiện tại API chưa có xây dựng chức năng xác thực người dùng.

#### 4.2.7. Xây dựng giao diện người dùng.

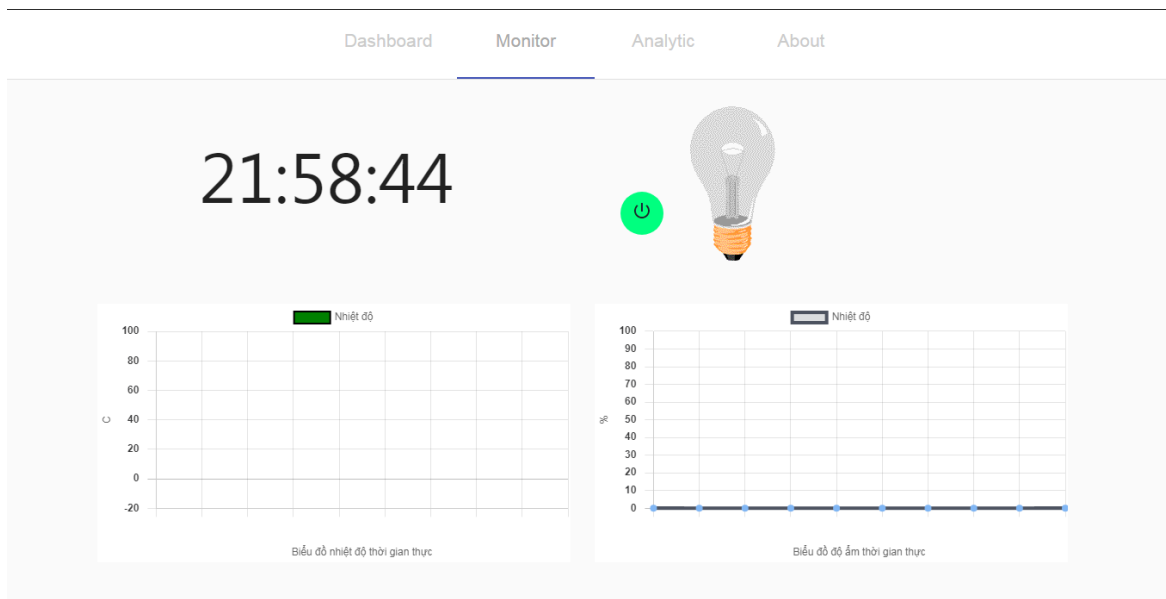
Giao diện Website sẽ gồm 3 phần chính:

- **Dashboard:** hiển thị các thông số của cảm biến theo thời gian thực. Các thông số ngày giờ sẽ được lấy từ hệ thống (Raspberry Pi). Các thông số cảm biến sẽ được hiển thị, thay đổi theo thời gian thực.



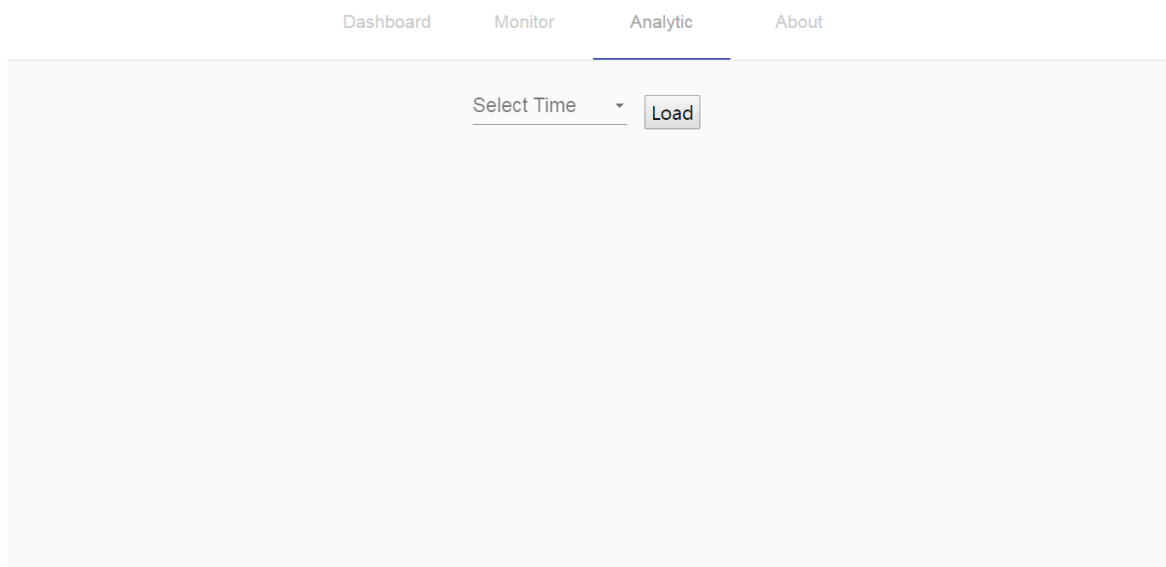
Hình 4.13. Xây dựng giao diện Dashboard.

- **Monitor:** hiển thị các thông số theo dạng biểu đồ nhiệt độ, độ ẩm và điều khiển đèn LED, cảnh báo khí GAS khi có dấu hiệu của khí GAS.



Hình 4.14. Xây dựng giao diện Monitor.

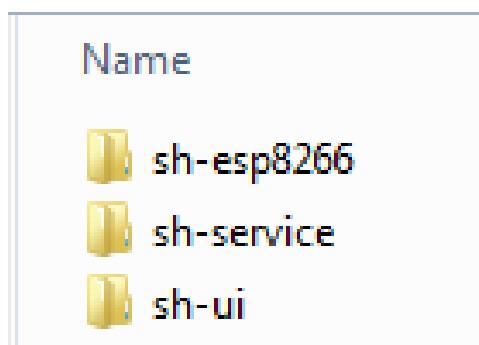
- **Analytic:** thống kê dữ liệu dựa trên lựa chọn thời gian trước đó: một giờ, sáu giờ, một ngày, một tuần.



Hình 4.15. Xây dựng giao diện Analytic.

#### 4.2.8. Chạy ứng dụng.

Sau khi nạp mã nguồn cho ESP8266. Tiến hành cấp điện cho ESP8266 và Raspberry Pi. Đảm bảo cả hai đều kết nối vào cùng mạng Wifi. Tải toàn bộ mã nguồn đã lập trình ở trên vào Raspberry Pi. Chuyển đến thư mục `~/smart-home/src`.



Hình 4.16. Thư mục mã nguồn.

Di chuyển vào thư mục `sh-service` và chạy back end:

```
$ cd sh-service
```

```
$ sudo python3 application.py
```

Nếu có báo lỗi không tìm thấy thư viện thì chạy lệnh để tự động cài thư viện trong file *requirement.txt*:

```
$ sudo pip3 install -r requirement.txt
```

Quay lại thư mục mã nguồn ban đầu và tiếp tục di chuyển vào thư mục *sh-ui*, chạy front end:

```
$ cd sh-ui
```

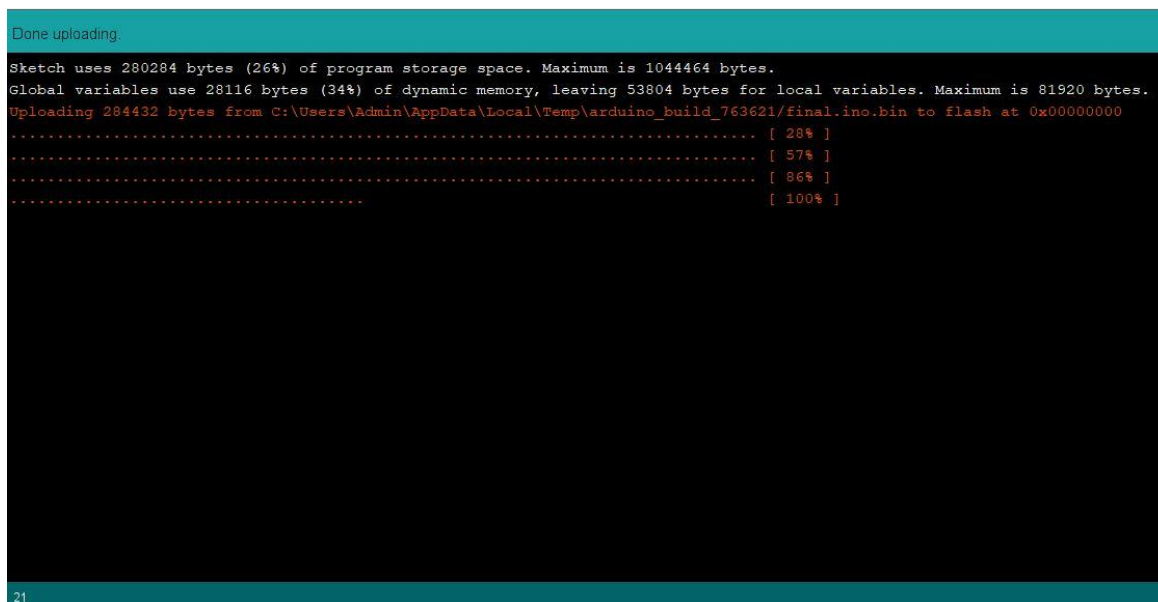
```
$ sudo ng serve --host <IP> --port <port>
```

Trong đó, IP và port lần lượt là IP của Raspberry Pi và cổng muốn chạy ứng dụng web.

### 4.3. Kiểm thử.

#### 4.3.1. Kết nối giữa các thiết bị.

Mã nguồn trong quá trình nạp vào ESP8266 sẽ hiển thị bên dưới vùng thông báo của Arduino IDE. Lỗi sẽ được thông báo nếu mã nguồn có lỗi về cú pháp, biên dịch và kết nối. Bên dưới là kết quả của nạp mã nguồn thành công vào ESP8266.



Hình 4.17. Thông báo nạp mã nguồn thành công.

Kết nối ESP8266 với máy tính để xem quá trình gửi dữ liệu từ cảm biến lên ESP8266 và lên Raspberry Pi.



```

COM5
21:27:36.467 ->
21:27:36.467 -> Connecting to Wifi.
21:27:36.973 -> WiFi connected
21:27:36.973 -> IP address:
21:27:37.007 -> 192.168.1.213
21:27:37.007 -> Attempting MQTT connection...connected
21:27:37.747 -> LIVINGROOM/LIGHT
21:27:37.747 -> 1
21:27:37.781 -> -----
21:27:37.781 -> LIVINGROOM/GAS
21:27:37.814 -> 0
21:27:37.814 -> -----
21:27:37.847 -> LIVINGROOM/HUMIDITY
21:27:37.847 -> ????▲
21:27:37.847 -> -----
21:27:37.881 -> LIVINGROOM/TEMPERATURE
21:27:37.914 -> 32.60
21:27:37.914 -> -----
21:27:37.947 -> LIVINGROOM/FLASH_LIGHT
21:27:37.981 -> Light is off
21:27:37.981 -> -----
21:27:42.879 -> LIVINGROOM/LIGHT
21:27:42.879 -> 1
21:27:42.879 -> -----
21:27:42.913 -> LIVINGROOM/GAS
21:27:42.913 -> 0
21:27:42.913 -> -----
21:27:42.948 -> LIVINGROOM/HUMIDITY
21:27:42.981 -> 61
21:27:42.981 -> -----
21:27:43.014 -> LIVINGROOM/TEMPERATURE
21:27:43.014 -> 32.60
21:27:43.048 -> -----
21:27:43.048 -> LIVINGROOM/FLASH_LIGHT
21:27:43.081 -> Light is off
21:27:43.114 -> -----
21:27:48.012 -> Message arrived [
21:27:48.012 -> LIVINGROOM/FLASH_LIGHT/CONTROL] 1
21:27:48.046 -> Turn on light
21:27:48.079 -> LIVINGROOM/LIGHT
21:27:48.079 -> 1
21:27:48.079 -> -----
21:27:48.113 -> LIVINGROOM/GAS
21:27:48.146 -> 0

☐ Autoscroll ☒ Show timestamp

```

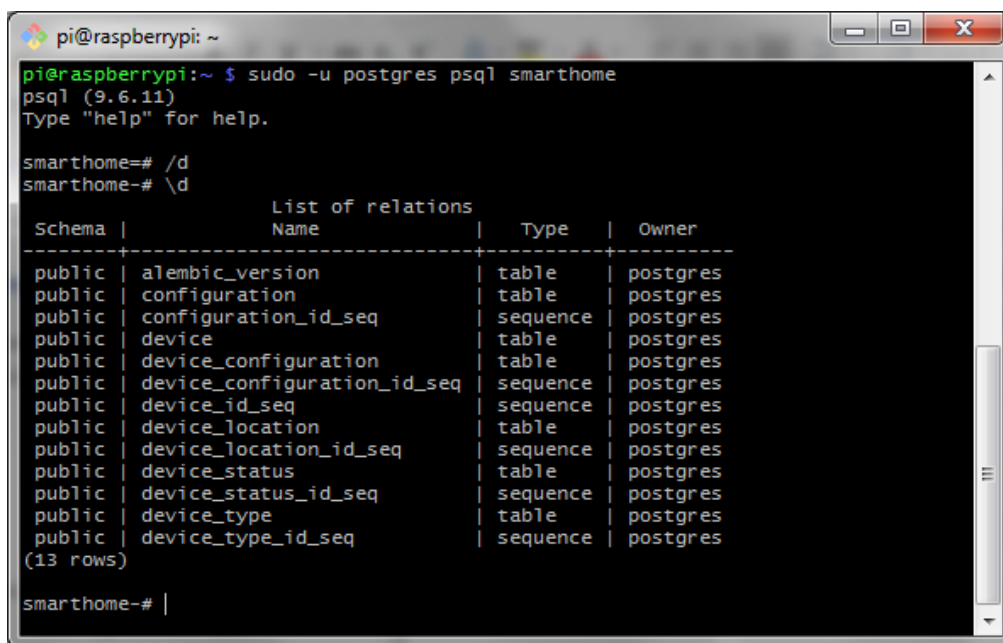
Hình 4.18. Dữ liệu nhận và gửi trên ESP8266.

Kết quả trên cho thấy việc đã kết nối giữa cảm biến và Raspberry với ESP8266. Đã kết nối được mạng Wifi và dữ liệu đã được ghi nhận và gửi lên Raspberry thông qua giao thức MQTT.



### 4.3.2. Lưu trữ dữ liệu.

Truy cập vào cơ sở dữ liệu PostgreSQL để kiểm tra cơ sở dữ liệu đã được tạo thành công.



```

pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo -u postgres psql smarthome
psql (9.6.11)
Type "help" for help.

smarthome=# \d
smarthome=# \d

```

Schema	Name	Type	Owner
public	alembic_version	table	postgres
public	configuration	table	postgres
public	configuration_id_seq	sequence	postgres
public	device	table	postgres
public	device_configuration	table	postgres
public	device_configuration_id_seq	sequence	postgres
public	device_id_seq	sequence	postgres
public	device_location	table	postgres
public	device_location_id_seq	sequence	postgres
public	device_status	table	postgres
public	device_status_id_seq	sequence	postgres
public	device_type	table	postgres
public	device_type_id_seq	sequence	postgres

```

(13 rows)

smarthome=# |

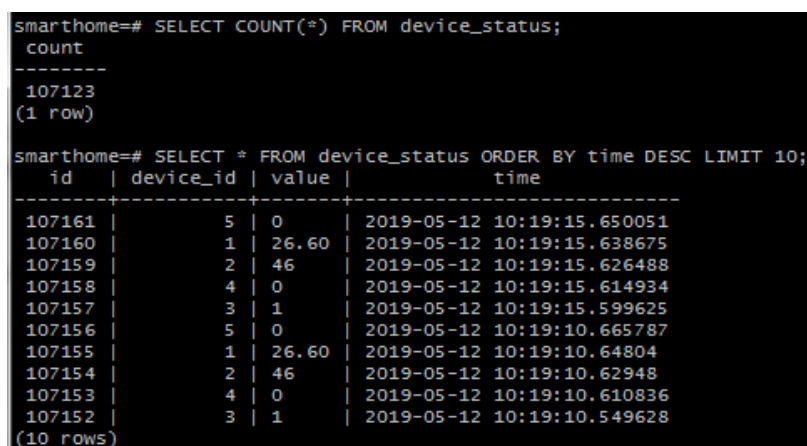
```

Hình 4.19. Kiểm tra tạo cơ sở dữ liệu.

Thiết bị đã được kết nối với nhau sau khi chạy chương trình, tất cả các dữ liệu được gửi lên từ ESP8266 sẽ được lưu trữ trong bảng device\_status. Sử dụng câu truy vấn để kiểm tra dữ liệu đã được lưu trữ:

```
$ SELECT COUNT(*) FROM device_status;
```

```
$ SELECT * FROM device_status ORDER BY time DESC LIMIT 10;
```



```

smarthome=# SELECT COUNT(*) FROM device_status;
count
-----
107123
(1 row)

smarthome=# SELECT * FROM device_status ORDER BY time DESC LIMIT 10;
 id | device_id | value |                time
-----+-----+-----+-----
107161 | 5 | 0 | 2019-05-12 10:19:15.650051
107160 | 1 | 26.60 | 2019-05-12 10:19:15.638675
107159 | 2 | 46 | 2019-05-12 10:19:15.626488
107158 | 4 | 0 | 2019-05-12 10:19:15.614934
107157 | 3 | 1 | 2019-05-12 10:19:15.599625
107156 | 5 | 0 | 2019-05-12 10:19:10.665787
107155 | 1 | 26.60 | 2019-05-12 10:19:10.64804
107154 | 2 | 46 | 2019-05-12 10:19:10.62948
107153 | 4 | 0 | 2019-05-12 10:19:10.610836
107152 | 3 | 1 | 2019-05-12 10:19:10.549628
(10 rows)

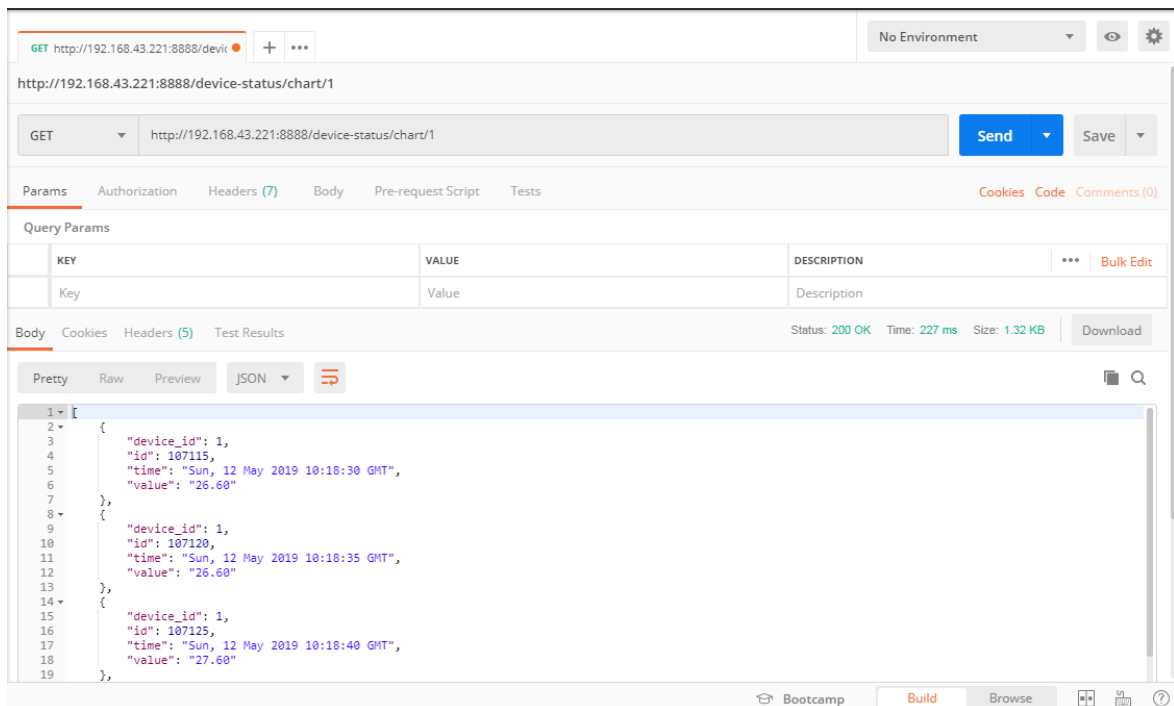
```

Hình 4.20. Kiểm tra dữ liệu bảng device\_status.

Từ kết quả trên, việc kết nối với cơ sở dữ liệu và thao tác lưu trữ cơ sở dữ liệu đã hoạt động.

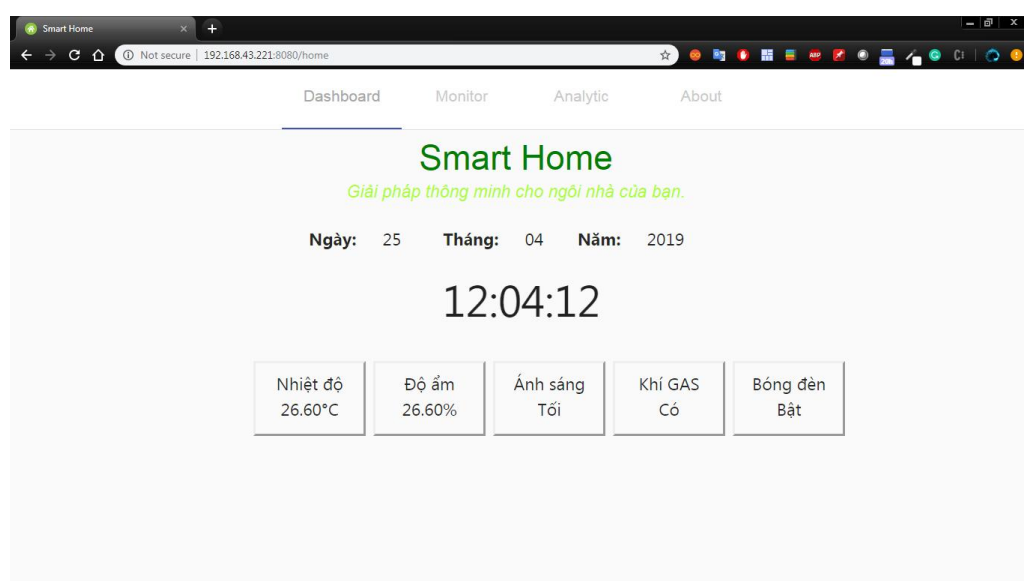
### 4.3.3. Ứng dụng web.

Kiểm tra hoạt động của API:

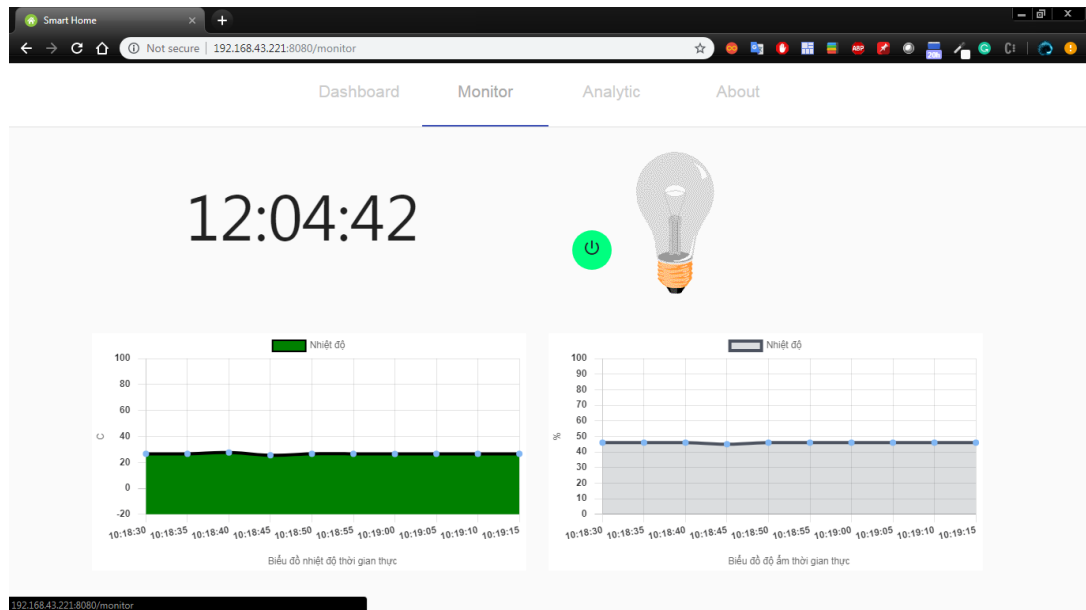


Hình 4.21. Kiểm tra API.

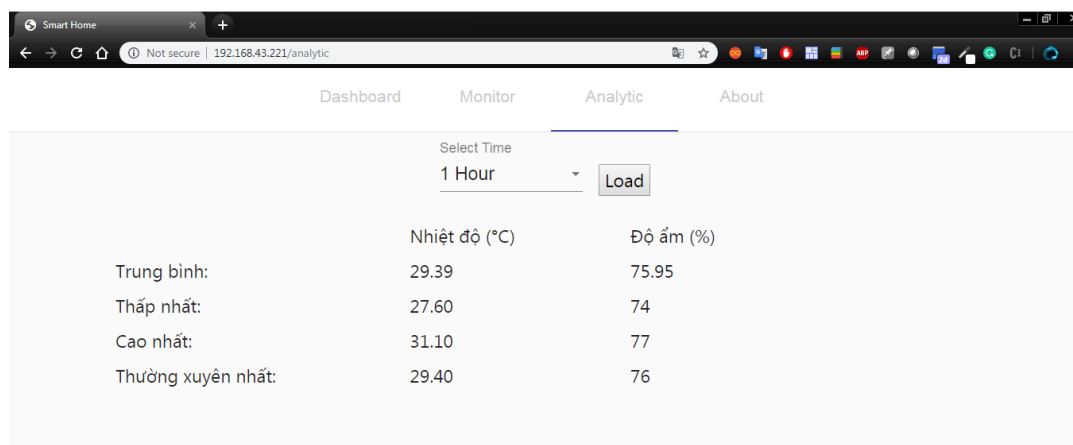
Truy cập vào ứng dụng theo địa chỉ IP và cổng đã cấu hình ở trên:



Hình 4.22. Kiểm tra giao diện Dashboard.



Hình 4.23. Kiểm tra giao diện Monitor.



Hình 4.24. Kiểm tra giao diện Analytic.

Giao diện và API đã hoạt động theo như thiết kế.

**Kết luận:** Thiết kế và xây dựng ứng dụng trên Raspberry đã hoàn thành và đã thu thập được dữ liệu lưu trữ dưới cơ sở dữ liệu.

## KẾT LUẬN

Quá trình hiểu về đề tài cũng như IoT, đến việc thiết kế và xây dựng ứng dụng, nhóm đã gặp không ít khó khăn về việc lập trình và cấu hình thiết bị. Các thiết bị điện tử dù không phải là chuyên môn của nhóm nhưng nhóm đã cố gắng tìm hiểu và học hỏi để phát triển đề tài một cách hoàn thiện. Qua đề tài này, nhóm em đã học hỏi thêm nhiều kinh nghiệm về làm việc nhóm và các công nghệ mới đang phát triển có sự ảnh hưởng tới công việc sau này của các thành viên trong nhóm.

Ứng dụng đã xây dựng hoàn thành và chạy ổn định trên Raspberry tuy nhiên vẫn còn gặp một số lỗi và chưa phải là ứng dụng hoàn hảo nhất. Dữ liệu khi thu thập được đã được thống kê nhưng mục đích của việc thu thập dữ liệu còn có thể áp dụng cho việc dự đoán và phân tích sâu hơn trong tương lai. Mô hình của đề tài đã mô phỏng lại cách hoạt động của các hệ thống IoT lớn.

Hướng phát triển tương lai của ứng dụng là phát triển cảnh báo sự bất thường qua mail, cảnh báo dự trên dự đoán từ dữ liệu trước đó, phát triển theo hướng nhiều thiết bị cảm biến. Nhóm đã phát triển tính năng cảnh báo qua mail nhưng hiện chưa chạy ổn định nên chưa đưa vào mô hình.

## NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày ... Tháng ... Năm 2019

Ký tên

## **TÀI LIỆU THAM KHẢO**

1. 30 Arduino™ Projects for the Evil Genius™ - Simon Monk .
2. Arduino Cookbook