Mục lục

[Mục lục 1](#_Toc29116997)

[Danh mục hình vẽ 2](#_Toc29116998)

[Lời nói đầu 3](#_Toc29116999)

[Nội dung đề tài 4](#_Toc29117000)

[I. Công nghệ, thiết bị sử dụng 5](#_Toc29117001)

[1. Node MCU ESP8266 5](#_Toc29117002)

[2. Module cảm biến nhiệt độ, độ ẩm DHT11 6](#_Toc29117003)

[3. Google Assistant 7](#_Toc29117004)

[4. IFTTT 8](#_Toc29117005)

[5. Adafruit IO 8](#_Toc29117006)

[6. MQTT 9](#_Toc29117007)

[II. Hoạt động 11](#_Toc29117008)

[1. Mô hình dự án 11](#_Toc29117009)

[2. Hoạt động IFTTT 12](#_Toc29117010)

[3. Dữ liệu Adafruit IO 12](#_Toc29117011)

[4. Hoạt động module DHT11 13](#_Toc29117012)

[5. Hoạt động Node MCU 16](#_Toc29117013)

[6. Thư viện Node MCU ESP8266 sử dụng 18](#_Toc29117014)

[6.1. Kết nối Wifi sử dụng thư viện <ESP8266WiFi.h>. 18](#_Toc29117015)

[6.2. Kết nối module DHT11 sử dụng thư viện <DHT.h> và <DHT\_U.h> 19](#_Toc29117016)

[6.3. Kết nối server bằng giao thứ MQTT sử dụng thư viện <PubSubClient.h> 21](#_Toc29117017)

[Kết quả 23](#_Toc29117018)

[Danh mục tài liệu tham khảo 24](#_Toc29117019)

Danh mục hình vẽ

[Hình 1: hình ảnh Node MCU 5](#_Toc29119173)

[Hình 2: hình ảnh Module DHT11 6](#_Toc29119174)

[Hình 3: minh hoạ sử dụng google assistant 7](#_Toc29119175)

[Hình 4: mô hình dịch vụ if this then that 8](#_Toc29119176)

[Hình 5: hiểu thị dữ liệu trên adafruit io trong thời gian thực 9](#_Toc29119177)

[Hình 6: ví dụ về dữ liệu chương trình 13](#_Toc29119178)

[Hình 7: gửi tín hiểu Start cho DHT11 14](#_Toc29119179)

[Hình 8: bit 0 15](#_Toc29119180)

[Hình 9: bit 1 15](#_Toc29119181)

[Hình 10: sơ đồ khối hoạt động bước setup 16](#_Toc29119182)

[Hình 11: sơ đồ khối hoạt động bước loop 17](#_Toc29119183)

[Hình 12: IFTTT gửi thông báo nhiệt độ, độ ẩm qua Messenger 23](#_Toc29119184)

[Hình 13: điều khiển qua google assitant 23](#_Toc29119185)

[Hình 14: Lưu trữ thông tin Adafruit IO 23](#_Toc29119186)

[Hình 15: kết quả thực tế 23](#_Toc29119187)

Lời nói đầu

Ngay từ tên gọi của nó, Internet Of Things viết tắt là IOT chính là internet trong mọi thứ. IOT chính là mạng lưới vạn vật kết nối Internet hoặc mạng lưới kết nối thiết bị Internet . Là một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh riêng của nó 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.

Khi mà vạn vật đều có chung một mạng kết nối thì việc liên lạc và làm việc trở nên rất dễ dàng. Con người có thể hiện thực hóa mục đích của mình trong tương lai. Chúng ta hoàn toàn có thể kiểm soát mọi thứ. Giả sử 1 chiếc ví mà các bạn đang sử dụng có tích hợp công nghệ IOT. Chúng có nhiệm vụ kiểm tra số lượng tiền trong ví, kiểm tra ngày hết hạn của các giấy tờ mà các bạn để trong đó như: bảo hiểm y tế, hạn nộp học phí,.. và thông báo tình trạng của nó đến cho chúng ta biết thông qua các ứng dụng tin nhắn SMS, Facebook, skype, zalo,…

Hiện này công nghệ điều khiển thiết bị sử dụng giọng nói đang được ứng dụng rộng dãi. Đặc biết đến từ sự phát triển nhanh chóng của trí tuệ nhân tạo giúp việc nhận dạng và điều khiển thông qua tiếng nói trở nên dễ dàng và chính xác hơn. Trong đó, Google Assistant “trợ lý ảo google” kết hợp cùng với Google Translate giúp nhận diện các ngôn ngữ khác nhau với độ chính xác rất tốt. Bài viết này sẽ sử dụng Google Assistant để nhận diện điều khiển bật tắt đèn cũng như truyền số liệu nhiệt độ độ ẩm cho thiết bị.

Nội dung đề tài

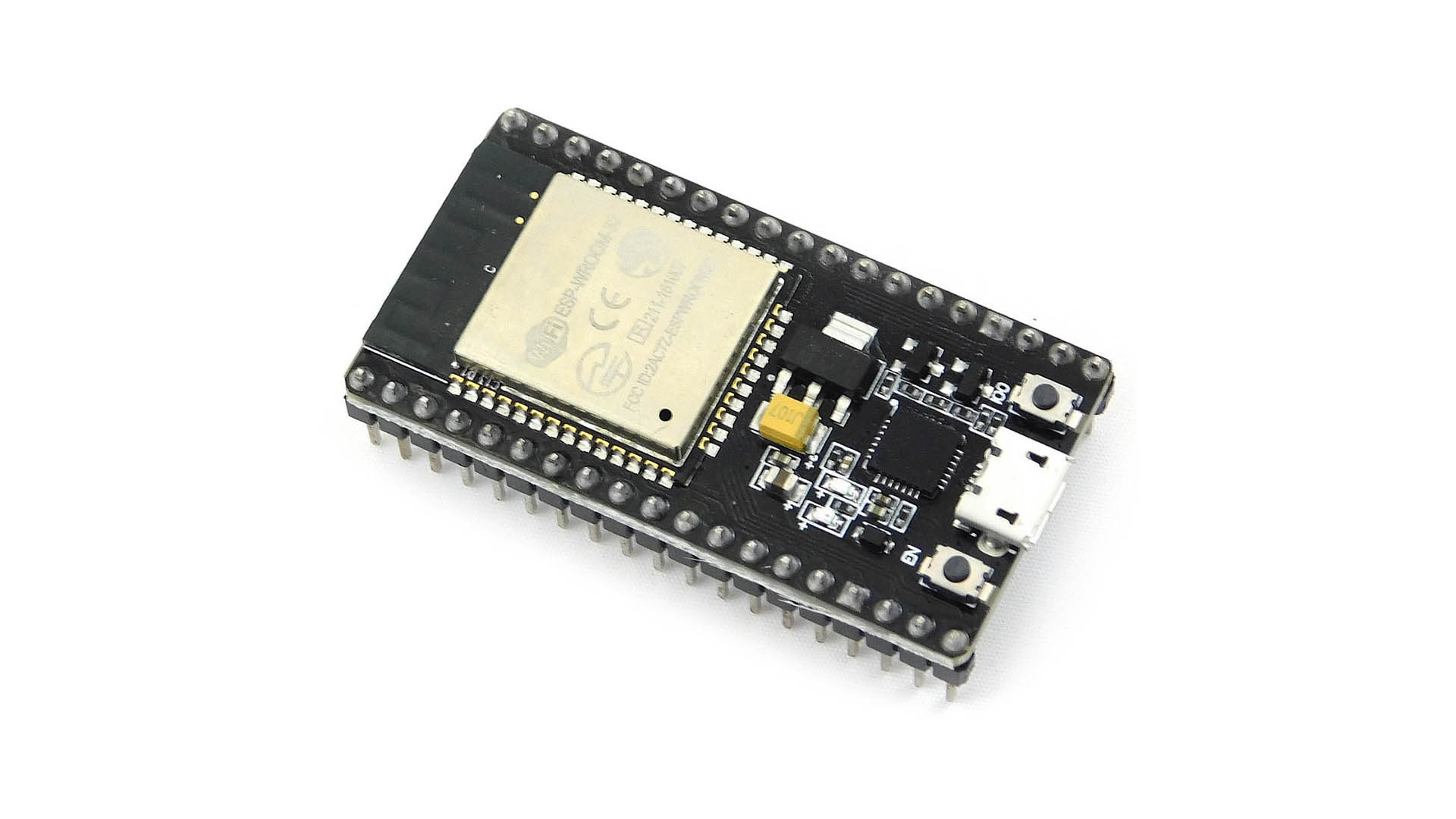
Đề tài: Điều khiển giọng nói thông qua internet bật tắt đèn.

Giải pháp:

* Sử dụng module Node MCU ESP8266 kết nối tới wifi, đọc dữ liệu từ server, điều khiển đèn (minh hoạ bằng hai đèn led). Yêu cầu nâng cao: gửi dữ liệu lên server (minh hoạ bằng việc đọc nhiệt độ độ ẩm từ module DHT11 và gửi lên server cứ hơn 30 giây một lần).
* Server lưu trữ dữ liệu, các request các bên gửi tới, hiển thị dữ liệu Adafruit IO.
* Server bắt các trigger từ các bên liên quan (khi người dùng yêu cầu giọng nói, có tin nhắn từ module esp gửi lên, …) và gửi các request tới các bên nhận yêu cầu IFTTT.
* Sử dụng google assistant để nhận lệnh từ người dùng.

I. Công nghệ, thiết bị sử dụng

1. Node MCU ESP8266



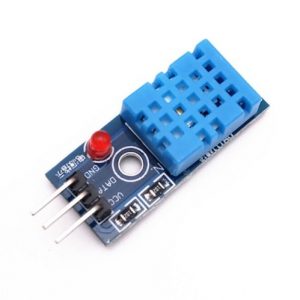
Hình : hình ảnh Node MCU

Đây là module NodeMCU dựa trên ESP8266 có kết nối Wifi, CP1202 trên board mạch và các phím. Nhờ tài nguyên mã nguồn mở phong phú, module này hỗ trợ phát triển theo nhiều cách khác nhau như AT/MicroPython/Arduino/IOT/C/C++.

Thông số kỹ thuật:

|  |  |  |
| --- | --- | --- |
| Loại | Thông tin | Ghi chú |
| Wi-Fi | Chứng nhận | Wi-Fi Alliance |
| Protocols | 802.11 b/g/n (HT20) |
| Dải tần số | 2.4G~2.5G |
| Tx Power | 802.11 b: +20 dBm  802.11 g: +17 dBm  802.11 n: +14 dBm |
| Rx Sensitivity | 802.11 b: -91 dbm (11 Mbps)  802.11 g: -75 dbm (54 Mbps)  802.11 n: -72 dbm (MCS7) |
| Ăng ten | PCB Trace, External, IPEX Connector, Ceramic Chip |
| Handware | CPU | Tensilica L106 32-bit processor |
| Giao thức kết nối ngoại vi | UART/SDIO/SPI/I2C/I2S/IR Remote Control  GPIO/ADC/PWM/LED Light & Button |
| Điện áp hoạt động | 2.5~3.6V |
| Cường độ điện áp | Giá trị trung bình: 80mA |
| Dải nhiệt độ hoạt động | -40°C~125°C |
| Software | Wi-Fi mode | Station/SoftAP/SoftAP+Station |
| Bảo mật | WPA/WPA2 |
| Mã hoá | WEP/TKIP/AES |
| Firmware update | UART Download / OTA (via network) |
| Software Development | Supports Cloud Server Development / Firmware and SDK for fast on-chip programming |
| Giao thức mạng | IPv4, TCP/UDP/HTTP |

1. Module cảm biến nhiệt độ, độ ẩm DHT11



Hình : hình ảnh Module 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)
* Dải độ ẩm hoạt động: 20%~90% RH, sai số ±5% RH
* Dải nhiệt độ hoạt động: 0°C~50°C, sai số ±2°C
* Khoảng cách truyền tối đa: 20M

1. Google Assistant

Google Assistant là một trợ lý cá nhân ảo được phát triển bởi Google cho thiết bị di động và nhà thông minh, được giới thiệu lần đầu tại hội nghị nhà phát triển của hãng vào tháng 5 năm 2016. Không giống như Google Now, Google Assistant có thể tham gia các cuộc trò chuyện hai chiều.



Hình : minh hoạ sử dụng google assistant

Assistant ban đầu được đưa vào ứng dụng nhắn tin Google Allo, và loa thông minh Google Home. Sau một thời gian chỉ có mặt trên hai chiếc điện thoại thông minh Pixel và Pixel XL của hãng, Google bắt đầu triển khai Assistant trên các thiết bị Android khác vào tháng 2 năm 2017, bao gồm cả các điện thoại thông minh bên thứ ba và các thiết bị Android Wear, và được phát hành dưới dạng ứng dụng riêng biệt trên iOS vào tháng 5. Cùng với sự ra mắt một bộ phát triển phần mềm vào tháng 4 năm 2017, Assistant đã và đang được tiếp tục mở rộng hỗ trợ cho một lượng lớn thiết bị, bao gồm cả xe hơi và các thiết bị nhà thông minh. Các chức năng của Assistant cũng có thể được bổ sung bởi các nhà phát triển bên thứ ba.

Người dùng chủ yếu có thể tương tác với Google Assistant qua giọng nói tự nhiên, hoặc có thể nhập qua bàn phím. Các chức năng cơ bản của nó cũng tương tự như Google Now, như tìm kiếm trên Internet, đặt sự kiện trên lịch và báo thức, điều chỉnh cài đặt phần cứng trên thiết bị người dùng và hiển thị thông tin từ tài khoản Google của người dùng. Google cũng bổ sung các tính năng khác cho Assistant bao gồm khả năng nhận diện vật thể và thu thập thông tin về vật thể thông qua máy ảnh của thiết bị, cùng với việc hỗ trợ mua sản phẩm và chuyển tiền.

1. IFTTT

If This Then That , còn được gọi là IFTTT là một dịch vụ dựa trên web miễn phí để tạo chuỗi các câu lệnh điều kiện đơn giản, được gọi là applet. Một applet được kích hoạt bởi những thay đổi xảy ra trong các dịch vụ web khác như Gmail, Facebook, Telegram, Instagram hoặc Pinterest, các dịch vụ như tin nhắn, google assistant, alexa,…



Hình : mô hình dịch vụ if this then that

Về cơ bản IFTTT là các Trigger, giúp bạn xử lý công việc được lập trình sẵn, khi xảy ra trường hợp này thì sẽ dẫn đến công việc tiếp theo. Khi có sự thay đổi trên ứng dụng này thì IFTTT sẽ thay đổi trên ứng dụng kia.

Ví dụ Twitter và Facebook chẳng có liên quan gì đến nhau nhưng nếu bạn sử dụng 1 hashtag trên Twitter như #SEO thì bạn có thể lập trình để IFTTT post bài viết đó lên Facebook một cách tự động mà không cần bạn phải thao tác thêm gì trên Facebook nữa.

1. Adafruit IO

Adafruit.io là một dịch vụ đám mây. Có thể kết nối với nó qua Internet. Nó chủ yếu là để lưu trữ và sau đó lấy dữ liệu. Những việc Adafruit có thể làm:

* Hiển thị dữ liệu trong thời gian thực.
* Kết nối project tới internet.
* Kết nối project tới dịch vụ web như: Twitter, RSS feeds, weather services.
* Kết nối project tới các thiết bị hỗ trợ internet khác.



Hình : hiểu thị dữ liệu trên adafruit io trong thời gian thực

1. MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức gởi dạng publish/subscribe sử dụng cho các thiết bị [Internet of Things](/tags/IoT) với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M.

**Publish, subscribe**: trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client - gọi tắt là client) kết nối tới một MQTT server (gọi là broker). Mỗi client sẽ đăng ký một vài kênh (topic), ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là **"subscribe"**, giống như chúng ta đăng ký nhận tin trên một kênh Youtube vậy. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gởi dữ liệu và kênh đã đăng ký. Khi một client gởi dữ liệu tới kênh đó, gọi là **"publish"**.

**QoS: ở** đây có 3 tuỳ chọn **QoS (Qualities of service)** khi "publish" và "subscribe":

* **QoS0** Broker/client sẽ gởi dữ liệu đúng 1 lần, quá trình gởi được xác nhận bởi chỉ giao thức TCP/IP, giống kiểu đem con bỏ chợ.
* **QoS1** Broker/client sẽ gởi dữ liệu với ít nhất 1 lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
* **QoS2** Broker/client đảm bảm khi gởi dữ liệu thì phía nhận chỉ nhận được đúng 1 lần, quá trình này phải trải qua 4 bước bắt tay.

Một gói tin có thể được gởi ở bất kỳ QoS nào, và các client cũng có thể subscribe với bất kỳ yêu cầu QoS nào. Có nghĩa là client sẽ lựa chọn QoS tối đa mà nó có để nhận tin. Ví dụ, nếu 1 gói dữ liệu được publish với QoS2, và client subscribe với QoS0, thì gói dữ liệu được nhận về client này sẽ được broker gởi với QoS0, và 1 client khác đăng ký cùng kênh này với QoS 2, thì nó sẽ được Broker gởi dữ liệu với QoS2.

Một ví dụ khác, nếu 1 client subscribe với QoS2 và gói dữ liệu gởi vào kênh đó publish với QoS0 thì client đó sẽ được Broker gởi dữ liệu với QoS0. QoS càng cao thì càng đáng tin cậy, đồng thời độ trễ và băng thông đòi hỏi cũng cao hơn.

**Retain**: nếu RETAIN được set bằng 1, khi gói tin được publish từ Client, Broker **PHẢI** lưu trữ lại gói tin với QoS, và nó sẽ được gởi đến bất kỳ Client nào subscribe cùng kênh trong tương lai. Khi một Client kết nối tới Broker và subscribe, nó sẽ nhận được gói tin cuối cùng có RETAIN = 1 với bất kỳ topic nào mà nó đăng ký trùng. Tuy nhiên, nếu Broker nhận được gói tin mà có QoS = 0 và RETAIN = 1, nó sẽ huỷ tất cả các gói tin có RETAIN = 1 trước đó. Và phải lưu gói tin này lại, nhưng hoàn toàn có thể huỷ bất kỳ lúc nào.

Khi publish một gói dữ liệu đến Client, Broker phải se RETAIN = 1 nếu gói được gởi như là kết quả của việc subscribe mới của Client (giống như tin nhắn ACK báo subscribe thành công). RETAIN phải bằng 0 nếu không quan tâm tới kết quả của viẹc subscribe.

**LWT**: Gói tin LWT (last will and testament) không thực sự biết được Client có trực tuyến hay không, cái này do gói tin KeepAlive đảm nhận. Tuy nhiên gói tin LWT như là thông tin điều gì sẽ xảy đến sau khi thiết bị ngoại tuyến.

II. Hoạt động

1. Mô hình dự án



Hoạt động chương trình bao gồm:

* Request(1): Người dùng bật ứng dụng google assistant của tài khoản mình lên (có thẻ là điện thoại, google home, tivi,…).
* Reply(2): Nếu yêu cầu người dùng khớp với những câu lệnh được cài đặt trước IFTTT thì google assistant tự động trả lời theo câu lệnh cài đặt sẵn. Google assistant tự động phiên dịch ngôn ngữ người dùng nói vào, tuy nhiên câu trả lời thì chỉ nói câu có sẵn.
* Trigger(3): Khi người dùng nó câu lệnh đặt trước, IFTTT đưa vào câu lệnh If tương ứng. Tại đó ta sẽ đẩy request feed tương ứng vào adafruit. Từ đó Node MCU có thể đọc và xử lý.
* Write data(4): đẩy request feed tương ứng từ IFTTT vào adafruit ở bước 3.
* New data(5): nếu có bất kì dữ liệu mới nào từ trường message feed của adafruit thì send đoạn dữ liệu đó đến Message của Facebook (gửi dữ liệu nhiệt độ độ ẩm).
* Read(6): Node MCU đọc dữ liệu request feed ở adafruit và xử lý chúng (nếu request tắt đèn phòng ngủ => tắt đèn phòng ngủ và gửi kết quả lên adafruit, bật đèn phòng khách => bật đèn phòng khách và gửi kết quả lên adafruit, yêu cầu nhiệt độ phòng => gửi message đến adafruit).
* Write(7): Node MCU gửi dữ liệu lên adafruit như kết quả bật tắt đèn, định kì gửi nhiệt độ, độ ẩm, gửi thông báo nhiệt độ độ ẩm để IFTTT gửi Message của Facebook.
* Send message(8): gửi thông báo (nhiệt độ, độ ẩm) đến Message của Facebook.

1. Hoạt động IFTTT

If This Then That để tạo chuỗi các câu lệnh điều kiện đơn giản, được gọi là applet. Một applet được kích hoạt bởi những thay đổi xảy ra trong các dịch vụ web khác như Gmail, Facebook, Telegram, Instagram hoặc Pinterest, các dịch vụ như tin nhắn, google assistant, alexa,… Các câu lệnh chính IFTTT chương trình sử dụng:

Bảng : câu lệnh IFTTT chương trình sử dụng

|  |  |  |
| --- | --- | --- |
| IF | THEN | Hoạt động |
| Bạn nói “turn on the living room lights” | Send data request feed “on\_light\_livingroom” | Yêu cầu Node MCU bật đèn phòng khách |
| Bạn nói “turn off the living room lights” | Send data request feed “off\_light\_livingroom” | Yêu cầu Node MCU tắt đèn phòng khách |
| Bạn nói “turn on the bed room lights” | Send data request feed “on\_light\_bedroom” | Yêu cầu Node MCU bật đèn phòng ngủ |
| Bạn nói “turn on the bed room lights” | Send data request feed “on\_light\_bedroom” | Yêu cầu Node MCU bật đèn phòng ngủ |
| Bạn nói “What is the humidity of the house?” | Send data request feed “request\_humidity” | Yêu cầu Node MCU gửi thông tin độ ẩm |
| Bạn nói “What is the temperature of the house?” | Send data request feed “request\_temperature” | Yêu cầu Node MCU gửi thông tin nhiệt độ |
| Bạn nói “What is the temperature and humidity of the house?” | Send data request feed “request\_temperature \_humidity” | Yêu cầu Node MCU gửi thông tin nhiệt độ, độ ẩm |
| Có dữ liệu mới ở message feed | Gửi dữ liệu nhận được đến Message của Facebook | Gửi thông tin nhiệt độ độ ẩm từ Node MCU đến Message của Facebook |

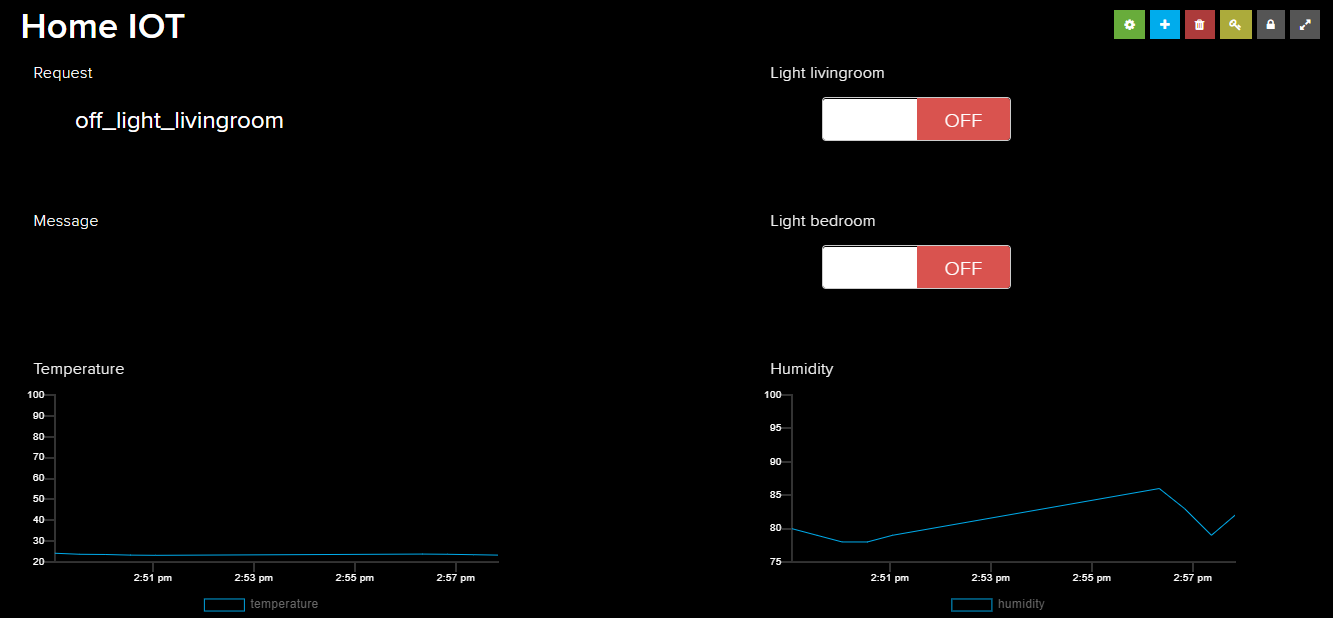
1. Dữ liệu Adafruit IO

Adafruit IO là server lưu trữ, hiển thị dữ liệu. Nhưng dữ liệu được cài đặt trong chương trình là:

|  |  |  |
| --- | --- | --- |
| Dữ liệu | Loại dữ liệu | Nội dung |
| request | text | Chứa nội dung IFTTT request tới. Node MCU đọc các request và xử lý. |
| messenger | text | Chứa nội dung Node MCU gửi lên. IFTTT bắt trigger đó và gửi cho messenger. |
| light\_livingroom | ON/OFF | Node MCU gửi trạng thái đèn phòng khách. |
| light\_bedroom | ON/OFF | Node MCU gửi trạng thái đèn phòng ngủ. |
| temperature | number | Node MCU gửi nhiệt độ căn phòng. |
| humidity | number | Node MCU gửi độ ẩm căn phòng. |

*Lưu ý: dữ liệu lưu ở dạng stream. Loại dữ liệu quy định giúp cho Adafruit IO có thể hiển thị theo đúng định dạng yêu cầu.*

Để truy cập vào dữ liệu ta cần phải có tên, AIO key của tài khoản. Dữ liệu truy cập nằm ở “<user\_name>/feeds/<data\_name>”. Dữ liệu hiển thị chương trình có dạng:



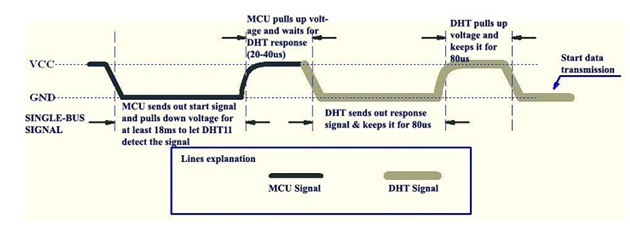
Hình : ví dụ về dữ liệu chương trình

1. Hoạt động module DHT11

DHT11 truyền dữ liệu qua một chân data. Nó sử dụng khoảng thời gian chênh lệch giữa mức điện áp cao và mức điện áp thấp.

Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

* Gửi tin hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.
* Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.

Bước 1: gửi tín hiệu Start

Hình : gửi tín hiểu Start cho DHT11

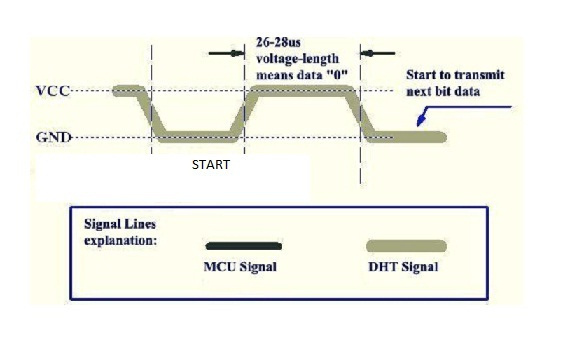
* MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Trong Code mình để 25ms. Khi đó DHT11 sẽ hiểu MCU muốn đo giá trị nhiệt độ và độ ẩm.
* MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.
* Sau khoảng 20-40us, DHT11 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA ko được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.
* Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp được với DHT11 ko. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DHT.

Bước 2: đọc giá trị trên DHT11

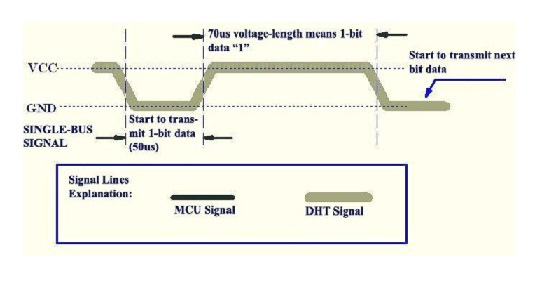
* DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte. Trong đó:
* Byte 1: giá trị phần nguyên của độ ẩm (RH%)
* Byte 2: giá trị phần thập phân của độ ẩm (RH%)
* Byte 3: giá trị phần nguyên của nhiệt độ (TC)
* Byte 4 : giá trị phần thập phân của nhiệt độ (TC)
* Byte 5 : kiểm tra tổng.
* Nếu Byte 5 = (8 bit) (Byte1 +Byte2 +Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.

Đọc dữ liệu:

* Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về MCU, tương ứng chia thành 5 byte kết quả của Nhiệt độ và độ ẩm.



Hình : bit 0



Hình : bit 1

Sau khi tín hiệu được đưa về 0, ta đợi chân DATA của MCU được DHT11 kéo lên 1. Nếu chân DATA là 1 trong khoảng 26-28 us thì là 0, còn nếu tồn tại 70us là 1. Do đó trong lập trình ta bắt sườn lên của chân DATA, sau đó delay 50us. Nếu giá trị đo được là 0 thì ta đọc được bit 0, nếu giá trị đo được là 1 thì giá trị đo được là 1. Cứ như thế ta đọc các bit tiếp theo.

1. Hoạt động Node MCU

Hoạt động Node MCU gồm hai bước chính:

* Bước setup: khởi tạo các giá trị như tần số thiết bị, chân vào ra I/O, khởi tạo kết nối wifi, server, sensor,…
* Bước loop: là vòng lặp sau bước cài đăt. Là hoạt động của thiết bị, quá trình này lặp đi lặp lại liên tục.



Hình 10: sơ đồ khối hoạt động bước setup

Tại bước setup bao gồm 5 bước chính:

* Thiết đặt tần số thiết bị, chân I/O (1): đối với chương trình thì đặt tần số 115200 Hz và đặt hai chân led output lần lượt là chân D0 và D2/
* Khởi tạo, kết nối wifi (2): ở bước này Node MCU sẽ quét và hiển thị những wifi gần đó và kết nối đến wifi đặt trước. Nếu không thể kết nối wifi thì sẽ lặp lại đến khi nào kết nối được.
* Khởi tạo, kết nối server (3): kết nối đến server io.adafruit.com tới cổng 1883 và set hàm callback cho nó.
* Khởi tạo, kết nối sensor DHT (4): thiết đặt chân data DHT (D7) và loại DHT là DHT11.
* Lấy giá trị thời gian (5): lấy giá trị thời gian kể từ lúc khởi động Node MCU để đếm thời gian định kì gửi dữ liệu nhiệt độ độ ẩm.



Hình 11: sơ đồ khối hoạt động bước loop

Tại bước loop gồm 9 bước chính:

* Nếu không kết nối server (1): là bước kiểm tra kết nối. Nếu sai thì sẽ sang bước gửi dữ liệu nhiệt độ, độ ẩm và cộng bộ đếm thời gian. Nếu đúng thì sang bước kết nối lại server
* Nếu không kết nối server (2): là vòng lặp while nếu sai thì đến bước gửi dữ liệu nhiệt độ, độ ẩm và cộng bộ đếm thời gian. Nếu đúng thì kết nối server.
* Kết nối server (3): kết nối với id thiết bị và user-pass người dùng trên server. Nếu kết nối được thì subsctibe feed. Nếu không kết nối được thì in ra thông báo và chờ 5s.
* In thông báo lỗi kết nối và chờ 5s (4).
* Subsctibe feed (5)
* Bộ đếm thời gian bằng 0 (6): nếu lần đầu tiên chạy hoặc qua một khoảng thời gian (bộ đếm đặt lại về 0). Nếu đúng thì đến bước gửi dữ liệu nhiệt độ độ ẩm. Nếu sai thì cộng bộ đếm thời gian.
* Gửi dữ liệu nhiệt độ độ ẩm (7): Gửi dữ liệu định kì nhiệt độ độ ẩm đến server.
* Cộng bộ đếm thời gian (8): Cộng thời gian mỗi lần gọi đến nếu lớn hơn 30s thì thiết đặt lại từ đầu.
* Xử lý dữ liệu nhạt được (9): được gọi callback khi có dữ liệu mới trên subscribe feed.

1. Thư viện Node MCU ESP8266 sử dụng
   1. Kết nối Wifi sử dụng thư viện <ESP8266WiFi.h>.

Thư viện này giúp ESP8266 kết nối wifi, kiểm tra kết nối wifi, reconnect wifi.

Hàm WiFi.begin(SSID, pass) chuyển đổi chế độ Station. Thiết đặt các tham số SSID và pass của wifi để module có thể kết nối đến một Access Point (AP) cụ thể. Theo mặc định ESP sẽ cố kết nối lại wifi khi bị disconnect. Do đó chúng ta không cần phải xử lý việc này trong code. Trả về trạng thái kết nối wifi:

* WL\_CONNECTED - Sau khi kết nối thành công được thiết lập.
* WL\_NO\_SSID\_AVAIL - Trong trường hợp cấu hình SSID không thể đạt được.
* WL\_CONNECT\_FAILED - Nếu mật khẩu không chính xác.
* WL\_IDLE\_STATUS - Khi Wi-Fi đang trong quá trình thay đổi giữa các trạng thái.
* WL\_DISCONNECTED - Nếu module không được cấu hình trong chế độ trạm.

|  |
| --- |
| wl\_status\_t WiFi.begin(char \*SSID,char \*pass); |

Hàm Wifi.status() chức năng trả về một trong những trạng thái kết nối wifi (như hàm Wifi.begin();

|  |
| --- |
| wl\_status\_t Wifi.status(); |

Hàm WiFi.scanNetworks() quét các mạng wifi hiện có và trả lại số lượng mạng hiện có.

|  |
| --- |
| int WiFi.scanNetworks(); |

Hàm WiFI.SSID(i) trả về SSID wifi số thứ tự i được quét (từ 0 đến n-1).

|  |
| --- |
| String WiFi.SSID(uint8\_t i); |

Hàm WiFI.RSSI(i) trả về giá trị cường độ tín hiệu wifi số thứ tự i được quét (từ 0 đến n-1).

|  |
| --- |
| int32\_t WiFi.RSSI(uint8\_t i); |

* 1. Kết nối module DHT11 sử dụng thư viện <DHT.h> và <DHT\_U.h>

Thư viện này giúp đọc dữ liệu nhiệt độ độ ẩm của họ DHT family (DHT11, DHT22, DHT21, AM2101).

Class DHT\_Unified giúp quản lý sensor (cắm vào chân nào, loại sử dụng) và chứa các hàm xử lý, giao tiếp với sensor. Ta phải khai báo chân cắm và loại sử dụng bằng hàm khởi tạo của class DHT\_Unified(pin, type).

|  |
| --- |
| /\* Define types of sensors. \*/  #define DHT11 11 /\*\*< DHT TYPE 11 \*/  #define DHT12 12 /\*\*< DHY TYPE 12 \*/  #define DHT22 22 /\*\*< DHT TYPE 22 \*/  #define DHT21 21 /\*\*< DHT TYPE 21 \*/  #define AM2301 21 /\*\*< AM2301 \*/  DHT\_Unified:: DHT\_Unified(uint8\_t pin,uint8\_t type) |

Hàm DHT\_Unified::begin() được gọi để setup sensor được gọi đầu tiên để cấu hình.

|  |
| --- |
| void DHT\_Unified::begin() |

Cấu trúc sensor\_t dùng để lưu trữ các giá trị của sensor (như tên, version, id, loại, giá trị lớn nhất, giá trị nhỏ nhất,…).

|  |
| --- |
| typedef struct  {  char name[12]; /\*\*< sensor name \*/  int32\_t version; /\*\*< version of the hardware + driver \*/  int32\_t sensor\_id; /\*\*< unique sensor identifier \*/  int32\_t type; /\*\*< this sensor's type (ex. SENSOR\_TYPE\_LIGHT) \*/  float max\_value; /\*\*< maximum value of this sensor's value in SI units \*/  float min\_value; /\*\*< minimum value of this sensor's value in SI units \*/  float resolution; /\*\*< smallest difference between two values reported by this sensor \*/  int32\_t min\_delay; /\*\*< min delay in microseconds between events. zero = not a constant rate \*/  } sensor\_t; |

Hàm getSensor(&sensor) ở class Temperature và Humidity thuộc class DHT\_Unified trả về dữ liệu sensor. Đi kèm với hàm temperature() và humidity() để lấy dữ liệu tương ứng với nhiệt độ và độ ẩm.

|  |
| --- |
| sensor\_t DHT\_Unified:: temperature().getSensor(sensor\_t &sensor);  sensor\_t DHT\_Unified:: humidity().getSensor(sensor\_t &sensor); |

Cấu trúc sensors\_event\_t cung cấp sự kiện senser. Cung cấp thông tin sensor và giá trị đọc được theo từng loại sensor. Ta có thể lấy giá trị nhiệt độ độ ẩm sau khi lấy event từ trường data trong cấu trúc (tuỳ thuộc vào loại dũ liệu mong muốn có thể là nhiệt độ event.temperature hay độ ẩm event.relative\_humidity.

|  |
| --- |
| /\* Sensor event (36 bytes) \*/  /\*\* struct sensor\_event\_s is used to provide a single sensor event in a common format. \*/  typedef struct  {  int32\_t version; /\*\*< must be sizeof(struct sensors\_event\_t) \*/  int32\_t sensor\_id; /\*\*< unique sensor identifier \*/  int32\_t type; /\*\*< sensor type \*/  int32\_t reserved0; /\*\*< reserved \*/  int32\_t timestamp; /\*\*< time is in milliseconds \*/  union  {  float data[4];  sensors\_vec\_t acceleration; /\*\*< acceleration values are in meter per second per second (m/s^2) \*/  sensors\_vec\_t magnetic; /\*\*< magnetic vector values are in micro-Tesla (uT) \*/  sensors\_vec\_t orientation; /\*\*< orientation values are in degrees \*/  sensors\_vec\_t gyro; /\*\*< gyroscope values are in rad/s \*/  float temperature; /\*\*< temperature is in degrees centigrade (Celsius) \*/  float distance; /\*\*< distance in centimeters \*/  float light; /\*\*< light in SI lux units \*/  float pressure; /\*\*< pressure in hectopascal (hPa) \*/  float relative\_humidity; /\*\*< relative humidity in percent \*/  float current; /\*\*< current in milliamps (mA) \*/  float voltage; /\*\*< voltage in volts (V) \*/  sensors\_color\_t color; /\*\*< color in RGB component values \*/  };  } sensors\_event\_t; |

Hàm getEvent(&event) ở class Temperature và Humidity thuộc class DHT\_Unified lấy event từ sensor. Đi kèm với hàm temperature() và humidity() để lấy event tương ứng với nhiệt độ và độ ẩm. Luôn trả về giá trị “true”.

|  |
| --- |
| void DHT\_Unified:: temperature().getEvent(sensor\_event\_t \*event)  void DHT\_Unified:: humidity().getEvent (sensor\_event\_t \*event); |

Hàm isnan(number) (is not a number) kiểm tra xem kết quả trả về có phải một số hay không. Sử dụng hàm này để kiểm tra giá trị lấy từ event (event.temperature và event.relative\_humidity) có chính xác hay không hay bị lỗi trong quá trình lấy dữ liệu.

|  |
| --- |
| bool isnan(float/double/long \_\_x);  //Kiểm tra nhiệt độ  isnan(event.temperature)  //Kiểm tra độ ẩm  isnan(event.relative\_humidity) |

* 1. Kết nối server bằng giao thứ MQTT sử dụng thư viện <PubSubClient.h>

Chỉ định client có thể kết nối tới một địa chỉ xác định ở đây là io.adafruit.com.

|  |
| --- |
| WiFiClient espClient;  PubSubClient client(espClient); |

Hàm PubSubClient::setServer(domain, port) kết nối tới domain với port. Trong trường hợp này thì domain là io.adafruit.com và port là 1883.

|  |
| --- |
| PubSubClient& PubSubClient::setServer(const char \* domain, uint16\_t port) |

Hàm PubSubClient::setCallback(callback) set hàm callback. Khi có dữ liệu mới từ subscribe feed từ trước, server sẽ gửi dữ liệu đến và được gọi tới hàm callback.

|  |
| --- |
| PubSubClient& PubSubClient::setCallback(MQTT\_CALLBACK\_SIGNATURE) |

Hàm callback có dạng:

|  |
| --- |
| void(char\* topic, uint8\_t\* payload, unsigned int length) |

Trong đó topic là feed dữ liệu mới (feed này phải được subscribe từ trước), payload là dữ liệu gửi đến và có chiều dài là length.

*Lưu ý: nếu có nhiều subscribe feed thì phân biết ở trường topic.*

Hàm PubSubClient::connected() kiểm tra xem đã kết nối với địa chỉ đó chưa. Trả về “true” nếu đã kết nối hoặc “false” nếu có lỗi xảy ra.

|  |
| --- |
| boolean PubSubClient::connected() |

Hàm PubSubClient::connect(id, user, pass) thiết lập kết nối tới địa chỉ. Trong đó id là tên thiết bị kết nối đến (do ta đặt tên), user-pass là user và pass của người dùng trên server đó.

*Lưu ý: pass là AIO key của tài khoản Adafruit IO.*

Hàm PubSubClient::subscribe(topic) thiết lập subscribe feed. Feed này nếu có dữ liệu mới thì sẽ gửi đến thiết bị.

|  |
| --- |
| boolean PubSubClient::subscribe(const char\* topic) |

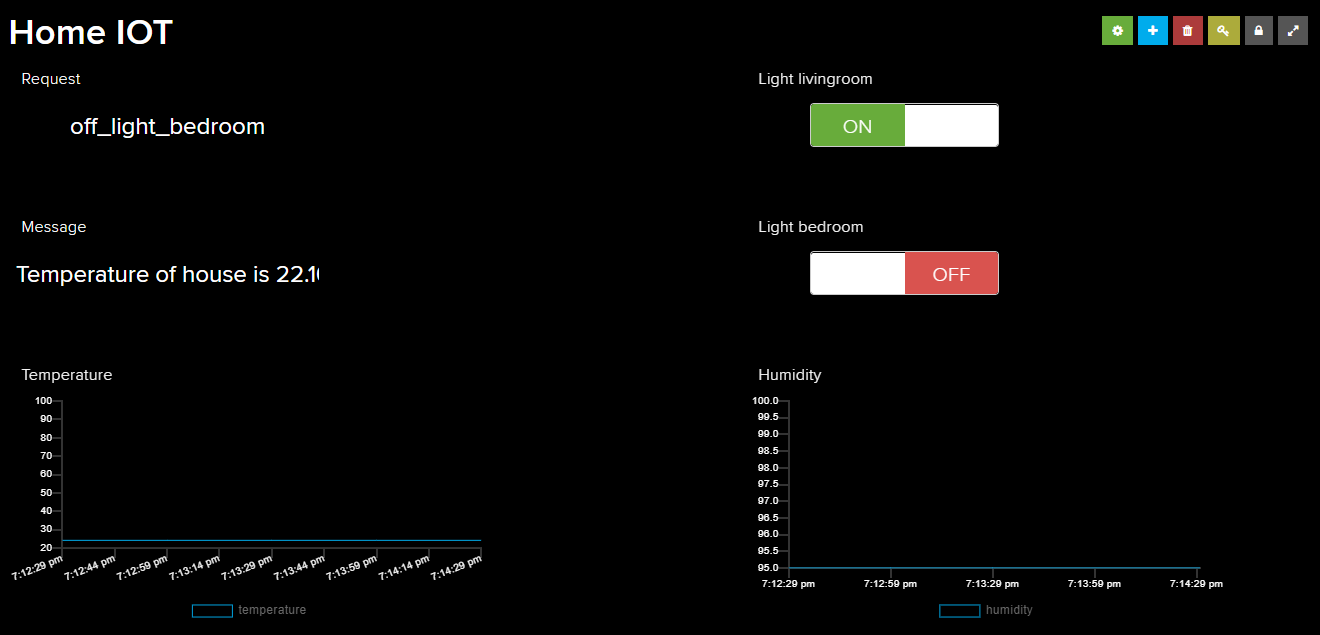
Hàm PubSubClient::publish(topic, payload) gửi dữ liệu-payload lên server đến topic có sẵn.

|  |
| --- |
| boolean PubSubClient::publish(const char\* topic, const char\* payload) |

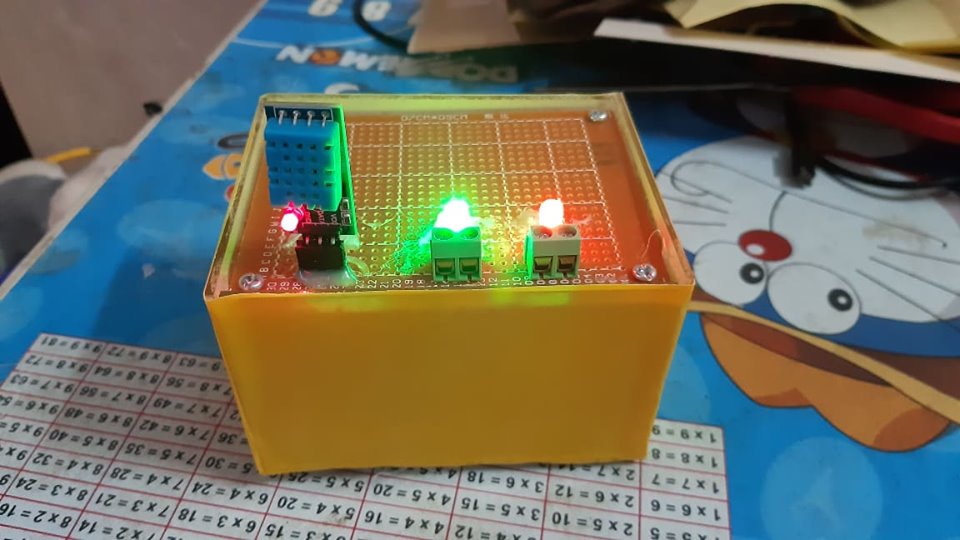
*Lưu ý: topic ở Adafruit có dạng “<user>/feed/<topic>” trong đó <topic> là trường dữ liệu đã tạo ra từ trước.*

Kết quả

|  |  |
| --- | --- |
| Hình : IFTTT gửi thông báo nhiệt độ, độ ẩm qua Messenger | Hình : điều khiển qua google assitant |



Hình : Lưu trữ thông tin Adafruit IO



Hình : kết quả thực tế

Danh mục tài liệu tham khảo

1. Module DHT11: <https://vidieukhien.xyz/vi/2018/04/06/bai-12-tim-hieu-va-giao-tiep-stm32f4-voi-dht11/>
2. <https://ifttt.com/>
3. [https://io.adafruit.com](https://io.adafruit.com/)