Nhóm HHH – ĐỘI TUYỂN PHẦN MỀM NGUỒN MỞ OLYMPIC TIN HỌC

UIT-OSS | UIT-VNUHCM

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

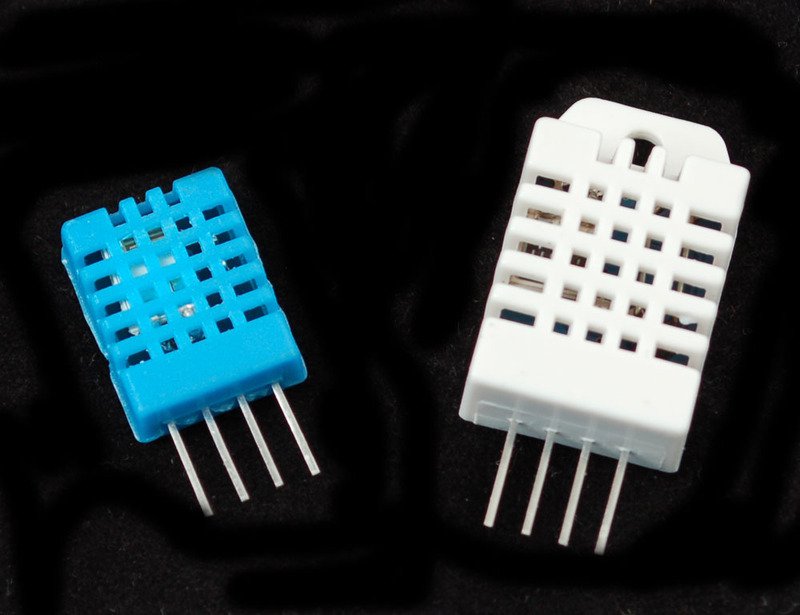
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

1. CẢM BIẾN DHT11

# Tổng quan về cảm biến DHT

Nguồn tham khảo: <https://learn.adafruit.com/dht>

Cảm biến DHT bao gồm 2 thành phần: thành phần cảm biến độ ẩm và thành phần cảm biến nhiệt độ (nhiệt kế). Có một con chip bên trong để chuyển đổi tín hiệu tuần tự sang tín hiệu số và phân tách ra thành giá trị nhiệt độ và giá trị độ ẩm.

Có 2 phiên bản DHT là DHT11 và DHT22.

*Cảm biến DHT11 (bên trái) và DHT22 (bên phải)*

Một số đặc tả kỹ thuật về 2 loại cảm biến trên[[1]](#footnote-1):

|  |  |
| --- | --- |
| **DHT11** | **DHT22** |
| Ultra low cost | Low cost |
| 3 to 5V power and I/O | 3 to 5V power and I/O |
| 2.5mA max current use during conversion (while requesting data) | 2.5mA max current use during conversion (while requesting data) |
| Good for **20-80% humidity** readings with 5% accuracy | Good for **0-100% humidity** readings with 2-5% accuracy |
| Good for **0-50°C temperature** readings *±2°C accuracy* | Good for **-40 to 125°C temperature** readings *±0.5°C accuracy* |
| No more than 1 Hz sampling rate (once every second) | No more than 0.5 Hz sampling rate (once every 2 seconds) |
| Body size 15.5mm x 12mm x 5.5mm | Body size 15.1mm x 25mm x 7.7mm |
| 4 pins with 0.1" spacing | 4 pins with 0.1" spacing |

# Thư viện hỗ trợ lập trình DHT trên Arduino Uno Wifi

Trên M.E.O project: <https://github.com/makerhanoi/meo-guide/tree/master/support-tools>

Giải nén có thư mục: DHT\_sensor\_library

Các thành phần quan trọng:

**DHT.h:** định nghĩa lớp (class) DHT theo kiểu lập trình hướng đối tượng và các thuooccj tính, nguyên mẫu hàm các phương thức trên lớp đó. Một số hàm quan trọng:

* DHT(uint8\_t pin, uint8\_t type, uint8\_t count=6) : hàm khởi tạo
* void begin(void): setup pin
* float readTemperature(bool S=false, bool force=false): đọc giá trị nhiệt độ trả về biến kiểu số thực, trong đó S là Scale : True là Fahrenheit, False là Celcius.
* float convertCtoF(float): chuyển đổi từ độ C sang độ F
* float convertFtoC(float): chuyển đổi từ độ F sang độ C
* float computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit=true); tính chỉ số HeatIndex. Xem thêm chỉ số heatindex tại: <http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml>
* float readHumidity(bool force=false); Đọc giá trị độ ẩm.

**DHT.cpp:** hiện thực các phương thức/hàm có trong DHT.hz**DHT\_U.h và DHT\_U.cpp**: thực hiện việc chuẩn hóa dữ liệu cảm biến theo *Adafruit Unified Sensor[[2]](#footnote-2)*

# Thử nghiệm các ví dụ trong thư mục

**Ví dụ 1:** DHTTest

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

// Read temperature **as Celsius** (the default)

float t = dht.readTemperature();

// Read temperature **as Fahrenheit** (isFahrenheit = true)

float f = dht.readTemperature(**true**);

// Check if any reads failed and exit early (to try again).

**if (isnan(h) || isnan(t) || isnan(f))** {

Serial.println("Failed to read from DHT sensor!");

return;

}

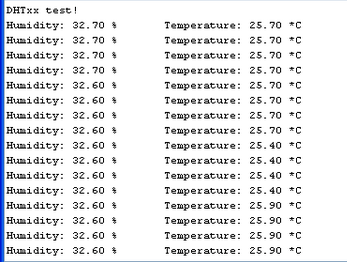
// Compute **heat index** in Fahrenheit (the default)

float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

float hic = dht.computeHeatIndex(t, h, false);

Kết quả hiển thị trên serial montor như hình sau:

**Ví dụ 2:** DHT\_U Example, tương tự chạy DHT\_U Example và so sánh kết quả.

1. THƯ VIỆN WIFIMANAGER.h

# Tổng quát thư viện WifiManager

Thư viện WiFiManager hỗ trợ ESP8266 mở 1 Webserver + 1 Access Point(AP), để user có thể kết nối vào và cấu hình wifi password.

Khi khởi động, ESP sẽ ở chế độ Station (điểm thu sóng) và thử kết nối với các Access Point (điểm phát sóng) đã lưu trước đó.

Nếu không thể kết nối, ESP sẽ chuyển qua chế độ AP và tạo một Webserver. Có thể kết nối vào AP vừa được tạo và cấu hình WIFI cho ESP.

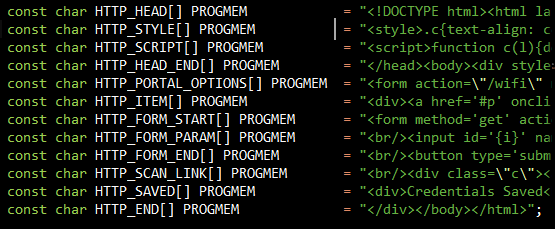
# Phân tích mã nguồn

Mã nguồn từ GitHub: <https://github.com/tzapu/WiFiManager>

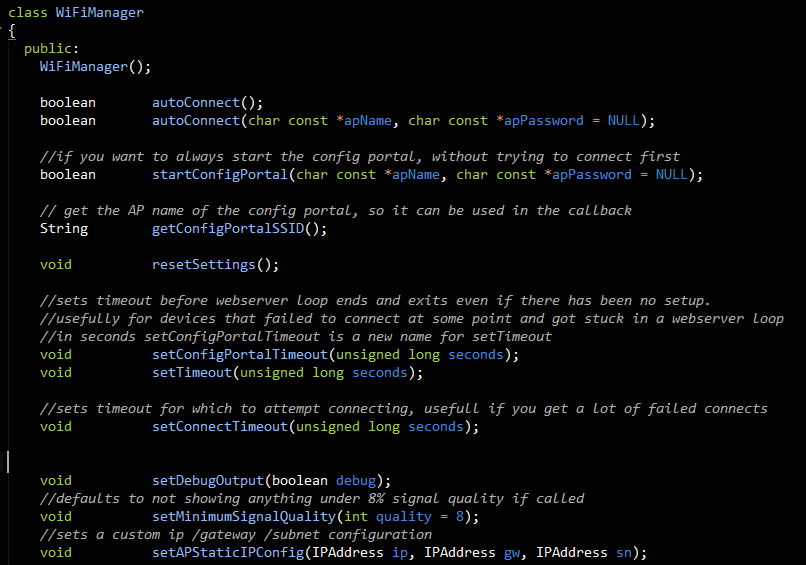
Cài đặt thư viện vào Arduino IDE từ Libraries Manges: Sketch -> Include Library -> Manager Libraries. Tìm kiếm thư viện WiFiManager và bấm Install.

## WiFiManager.h

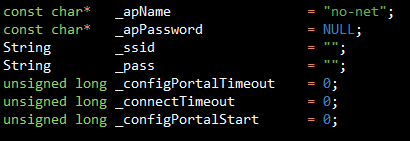
File header chứa các định nghĩa lớp, phương thức, thuộc tính của WiFiManager.

Phần khai báo cho WebServer dùng làm Portal để cấu hình WiFi:

Một đoạn định nghĩa các phương thức Public của lớp WiFiManager:

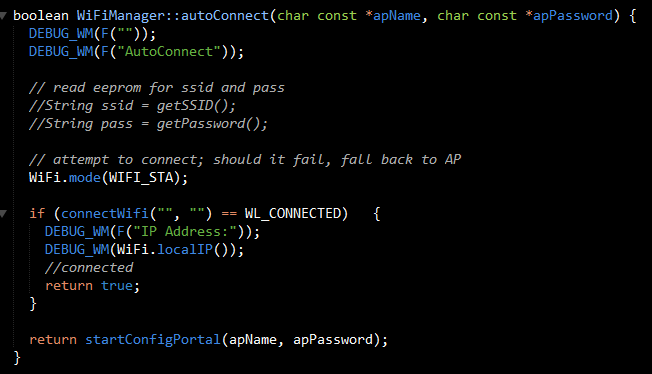


Trong đó 2 hàm quan trọng và thường dùng để cấu hình Wifi là autoConnect (có kiểm tra lại kết nối trước đó trước khi chuyển sang portal để cấu hình) còn hàm startConfigPortal là mở Portal Web lên để cấu hình mà không kiểm tra kết nối trước đó.

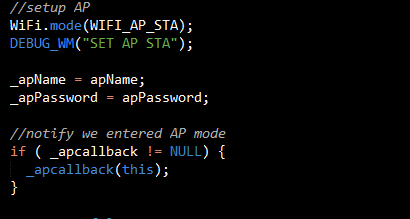
Các thuộc tính Private của lớp bao gồm: AP Name, Password, SSID,, Portal TimeOut, Static IP, Gateway,…

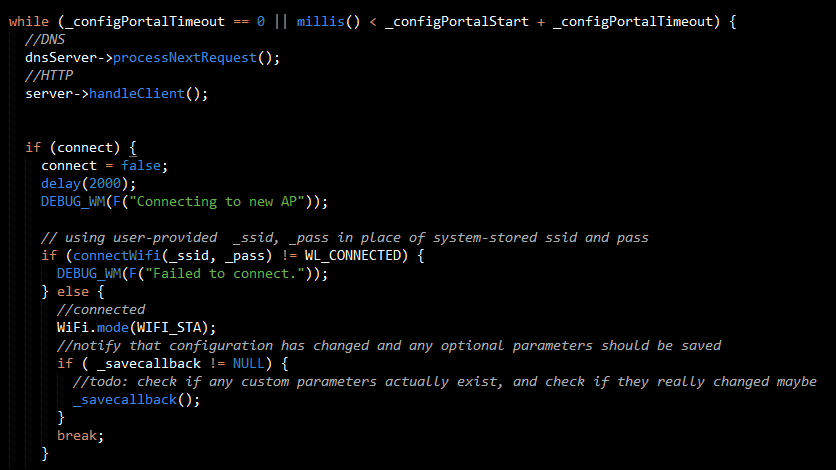
## WiFiManager.cpp

Đây là file hiện thực các hàm, phương thức có trong file header. Một số hàm quan trọng như:

*autoConnect*: Kiểm tra lại EEPROM xem có kết nối trước đó không, nếu có thì kết nối Wifi thành công, ngược lại bật chế độ AP và chuyển sang cấu hình kết nối bằng Portal

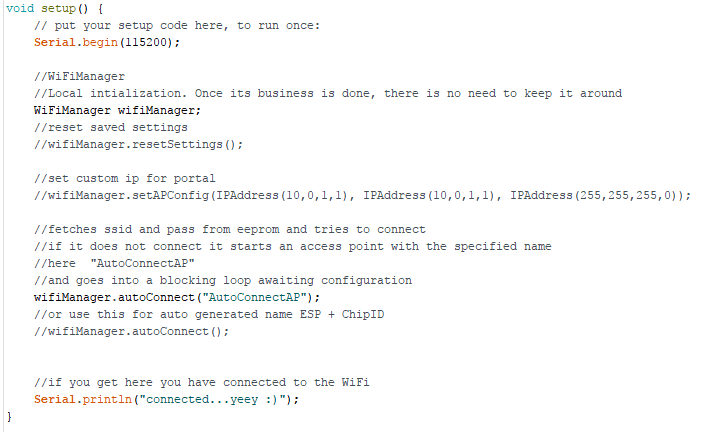
*startConfigPortal*

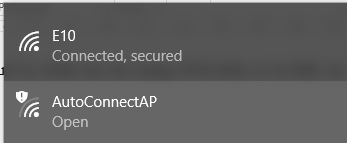
Đầu tiên chuyển sang chế độ AP, khởi tạo Web Server quản lý Portal để cấu hình cài đặt Wifi qua giao diện Web.

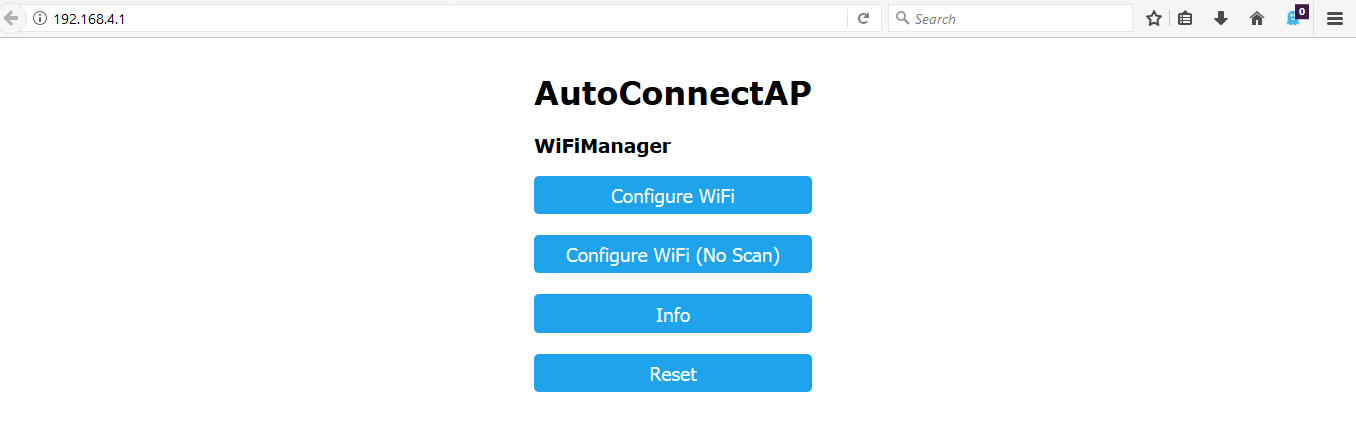


# Một số ví dụ có sẵn

## AutoConnect

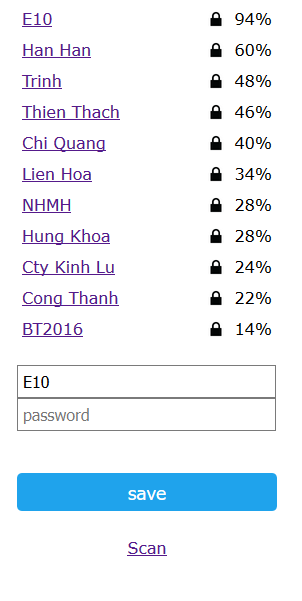
Sử dụng ví dụ AutoConnect trong thư viện để biết cách sử dụng hàm có sẵn và kết quả khi chạy thực tế.

Sau khi biên dịch và nạp lên board ta, kiểm tra các mạng Wifi hiện có ta thấy xuất hiện một AP mở mới với tên “AutoConnectAP”

Kết nối vào mạng mở này trình duyệt mặc định sẽ chuyển hướng sang trang portal có giao diện như sau:

Sử dụng các ô trên trình duyệt để cấu hình wifi cho ESP. Chọn vào ô Configure Wifi ta được chuyển tới màn hình như bên dưới.

Lúc này chọn mạng Wifi hiện có và truy cập vào để kết nối Wifi cho ESP.

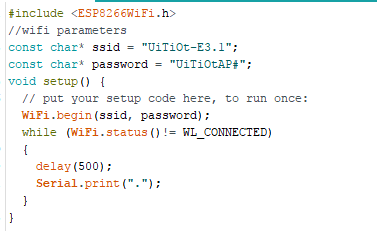


Sau khi Save, ESP đã được kết nối WiFi thành công!

Mở rộng: tiếp tục thử nghiệm các ví dụ khác trong thự viện để hiểu rõ hoạt động cũng như chức năng của các hàm có trong thư viện WiFiManager.h

# Các vấn đề liên quan

Như vậy, thư viện WiFiManager.h chỉ cung cấp các cách thức để quản lý việc kết nối WiFi cho thiết bị ESP8266 chứ không bao hàm tất cả các phương thức liên quan tới Wifi như tạo kết nối TCP/IP, giao tiếp với bên ngoài bằng Wifi…

Việc thực hiện kết nối WiFi cho ESP khi phát triển ứng dụng thường được thực hiện đơn giản bằng thư viện ESP8266WiFi.h như sau:

Tìm hiểu thêm các thư viện khác liên quan tới Wifi để biết cách sử dụng khi cần thiết:

ESP8266WiFi.h, WiFiClient.h, WiFiServer.h…. *(sách KolBan’s Book ESP8266)*

1. GIAO THỨC MQTT VÀ PUB\_SUB CLIENT

# Tổng quan về giao thức MQTT

## MQTT là gì?

MQTT (Message Queuing Telemetry Transport) là một giao thức gởi dạng ublish/subscribe sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định.

## Các định nghĩa quan trọng:

***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.

Tham khảo thêm tại[**QoS**](https://code.google.com/p/mqtt4erl/wiki/QualityOfServiceUseCases)

## https://i1.wp.com/hocarm.org/wp-content/uploads/2017/02/MQTT_mobile.jpg?resize=520%2C426&ssl=1MQTT Server (Broker)

Như hình vẽ thì MQTT Borker là một trạm trung chuyển, có nhiệm vụ nhận thông tin từ nhiều thiết bị khác nhau và chuyển tới từng thiết bị riêng biệt.

Có 2 cách để sử dụng MQTT Broker:

* Cài đặt, cấu hình MQTT Broker trên máy tính (ubuntu, rasp pi, server….) 🡺 phù hợp để phát triển, tùy biến.
* Sử dụng các dịch vụ MQTT Broker có sẵn 🡺 dễ sử dụng cho người mới tìm hiểu.

Một số công cụ hỗ trợ MQTT Broker: <https://github.com/mqtt/mqtt.github.io/wiki/tools>

🡺 Broker phổ biến được dùng nhiều và được trình bày trong tài liệu này là Mosquito: https://mosquitto.org/

## MQTT Client: các thiết bị kết nối tới broker để cập nhật dữ liệu. Ở đây sử dụng ESP8266 với thư viện hỗ trợ là Pubsubclient sẽ được trình bày trong phần sau.

# Thư viện PubSubClient

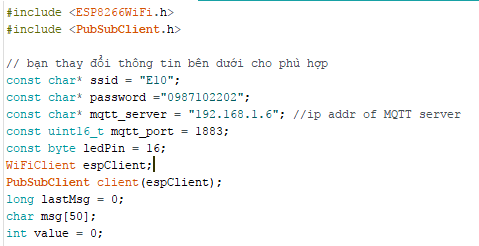
Thư viện PubSubClient định nghĩa lớp PubSub hoạt động như một MQTT Client trên ESP8266 bao gồm các hàm như sau:

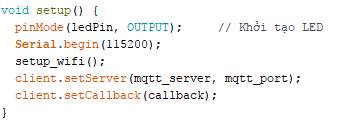


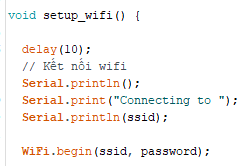
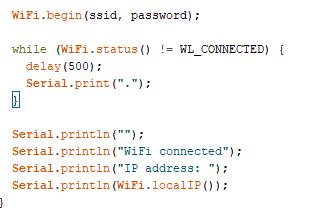
Tham khảo và tìm hiểu chi tiết tại : <https://pubsubclient.knolleary.net/api.html>

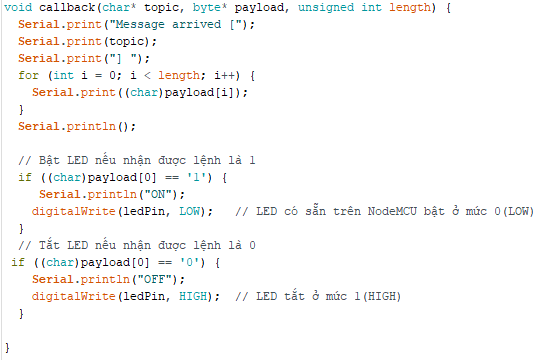
# Sử dụng thư viện PubSubClient

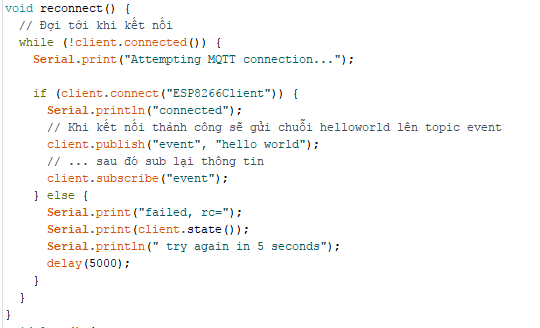
Các bước lập trình ứng dụng sử dụng thư viện Pubsub Client (minh họa là ứng dụng điều khiển đèn led từ xa)

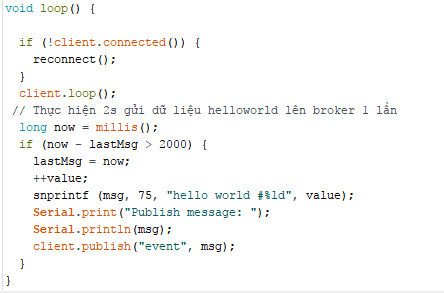
**Bước 1:** Khai báo các thành phần cần thiết: SSID Wifi, Password Wifi, địa chỉ MQTT Server, MQTT Port, Pin điều kiển LED, khai báo các đối tượng cần thiết (Wificlient, Pubsubclient)

**Bước 2:** Hoàn thiện hàm setup bằng:

**Bước 3:** Viết hàm quản lý việc kết nối Wifi cho thiết bị

**Bước 4:** Viết hàm callback – hàm này sẽ định nghĩa những gì xảy ra khi nhận được các thông điệp điều khiển, dữ liệu nhận về sẽ nằm trong payload, bóc tách lấy các giá trị này về để điều khiển thông ý muốn.

**Bước 5:** Viết hàm reconnect để duy trì kết nối giữa client và server

**Bước 6:** Hoàn chỉnh hàm loop với các công việc sẽ cần phải lặp lại vô hạn

Link demo kết quả: <https://goo.gl/HQPzXB>

**---- HẾT -----**

1. https://learn.adafruit.com/dht [↑](#footnote-ref-1)
2. <https://github.com/adafruit/Adafruit_Sensor> [↑](#footnote-ref-2)