PHẦN I. TỔNG QUAN

1. MÔ TẢ BÀI TOÁN

Với sự bùng nổ của Internet, công nghệ thông tin cùng với những tiến bộ của khoa học kĩ thuật, ngày càng có nhiều sản phẩm có những tính năng đa dạng và phong phú mà không thể không nhắc tới các ứng dụng Internet of Things (IoT) hay còn gọi là Internet vạn vật, một xu hướng đang phát triển mạnh mẽ và hứa hẹn sẽ còn phát triển hơn nữa trong tương lai. Lấy ý tưởng cốt lõi là kết nối mọi vật thông qua mạng lưới Internet, IoT giúp cho con người có thể kiểm soát mọi đồ vật một cách dễ dàng và tiện lợi thông qua các thiết bị công nghệ hiện đại như laptop, smartphone, máy tính bảng,… Bên cạnh đó, nền tảng web là một trong những nền tảng phổ biến nhất hiện này vì độ tương thích của nó, chúng ta có thể truy cập web bằng nhiều thiết bị khác nhau vào bất cứ khi nào và tại bất cứ đâu chỉ cần có kết nối Internet.

Nhận thấy những thuận lợi mà IoT và web mang lại, nhóm chúng em quyết định thực hiện đề tài "Hệ thống Arduino giám sát chỉ số môi trường trên nền tảng Web" nhằm mục đích sẽ ứng dụng được các công nghệ sẵn có để tạo ra một sản phẩm IoT có chức năng thu thập thông số cơ bản về môi trường trong không khí có tích hợp cảnh báo cháy có thể chạy trên nền tảng web. Hệ thống này tuy đơn giản nhưng có phạm vi ứng dụng rộng lớn. Một số ví dụ có thể kể đến là hỗ trợ giám sát điều kiện trồng nông sản trong nông nghiệp, bảo quản thực phẩm, sản phẩm trong công nhiệp và còn nhiều ứng dụng trong những lĩnh vực khác.

2. MỤC TIÊU ĐỀ TÀI

Giúp sinh viên có thêm kiến thức về công dụng và cách sử dụng của một số linh kiện, vi mạch điều khiển Arduino cũng như các công nghệ web hiện có. Đồng thời, tạo một ứng dụng web giúp thu thập các dữ liệu nhiệt độ, độ ẩm, độ cồn trong không khí và còn có thể phát hiện, cảnh báo cháy.

3. ĐỐI TƯỢNG VÀ PHẠM VI

Trong đề tài này, nhóm sử dụng mạch ESP8266 NodeMCU tích hợp thu phát WiFi điều khiển một số cảm biến, trong đó có: cảm biến thu thập nhiệt độ và độ ẩm DHT11, cảm biến phát hiện lửa và cảm biến nồng độ cồn MQ-3 trong môi trường Arduino. Ngoài ra, nhóm cũng dùng các ngôn ngữ lập trình web gồm HTML, CSS và JavaScript cùng một số công nghệ web hiện nay như ứng dụng đối tượng XMLHttpRequest (XHR) trong AJAX và lập trình bất đồng bộ.

4. NỘI DUNG NGHIÊN CỨU

Cài đặt Arduino IDE, board cùng các thư viện cần thiết. Nghiên cứu và thử nghiệm mạch xử lý tích hợp thu phát wifi ESP8266 cùng các module cảm biến DHT11, MQ-3 và cảm biến lửa. Thiết kế và lập trình trang web để hiển thị dữ liệu ra màng hình browser. Nghiên cứu giao thức HTTP và xử lý bất đồng bộ trong lập trình web. Lập trình server bất đồng bộ trên ESP8266.

PHẦN II. CƠ SỞ LÝ THUYẾT

1. MODULE LINH KIỆN

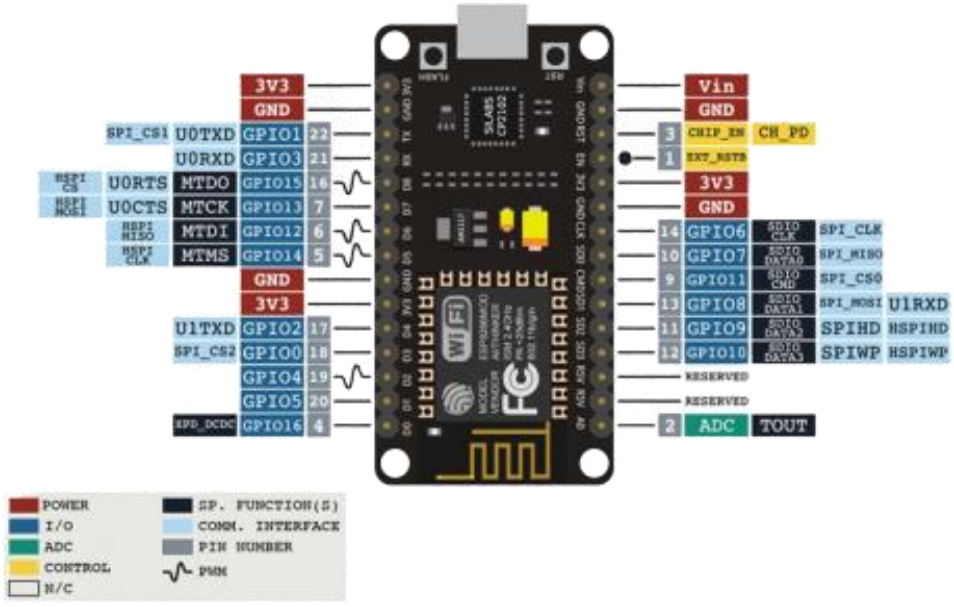
1.1. Chip ESP8266 NodeMCU xử lý tích hợp thu phát WiFi

ESP8266 NodeMCU là một trong những mạch phổ biến trong việc phát triển các dự án IoT, được phát triển dựa trên nền chip Wifi SoC ESP8266. Ưu điểm của ESP8266 NodeMCU sử dụng một vi điều khiển mạnh mẽ hơn so với Arduino nguyên thủy. Ngoài ra, thiết kế của ESP8266 NodeMCU rất nhỏ gọn, giá rẻ, đơn giản để sử dụng và có thể dùng trực tiếp trình biên dịch của Arduino (Arduino IDE) để lập trình và nạp code thông qua cổng micro USB. Điều này giúp việc sử dụng và lập trình các ứng dụng trên ESP8266 trở nên dễ dàng và tiện lợi hơn rất nhiều. Một điểm mạnh nữa chính là module wifi đã được tích hợp sẵn, giúp giảm chi phí lắp đặt trong việc phát triển các hệ thống điều khiển không dây.

Hình 1. Chip ESP8266 NodeMCU

Thông số kỹ thuật:

- Ic chính: ESP8266 Wifi SoC, phiên bản firmware: NodeMCU Lua
- Chip nạp và giao tiếp UART: CP2102
- Nguồn cấp: 5V DC Micro USB hoặc Vin
- GPIO giao tiếp mức logic: 3.3V
- Tích hợp Leb báo trạng thái, nút Reset, nút Flash

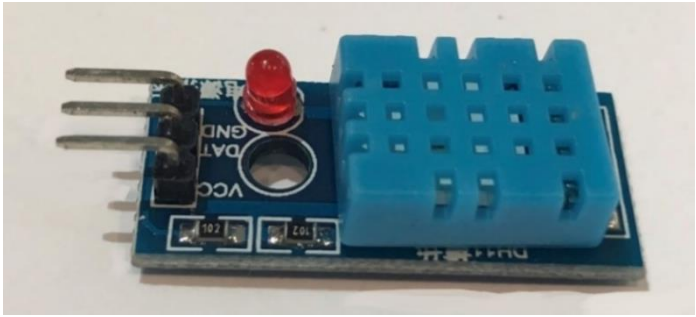


Hình 2. Sơ đồ chân ESP8266 NodeMCU

Chức năng chính trong đề tài: Là mạch xử lý chính, điều khiển các cảm biến thu thập các thông số từ môi trường, là khối giao tiếp không dây đảm nhận kết nối WiFi và đóng vai trò là một server để xử lý các yêu cầu từ các thiết bị client.

1.2. Module cảm biến nhiệt độ, độ ẩm DHT11

DHT11 là cảm biến có tích hợp chức năng đo nhiệt độ và độ ẩm không khí rất thông dụng hiện nay vì giá thành rẻ và dễ sử dụng, có thể 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.



Hình 3. Module cảm biến nhiệt độ và độ ẩm DHT11

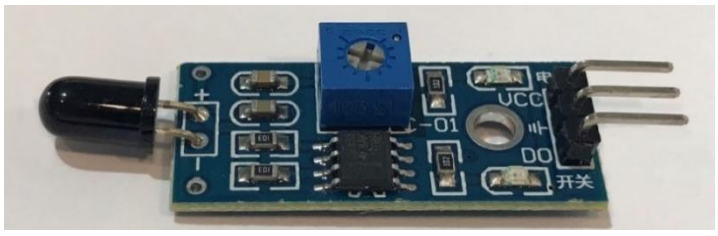
Thông số kỹ thuật:

- Điện áp hoạt động: 3 – 5V DC
- Dãi độ ẩm hoạt động: 20 - 90% RH, sai số ±5%RH
- Dãi nhiệt độ hoạt động: 0 – 50°C, sai số ±2°C
- Khoảng cách truyền tối đa: 20 m
- Chuẩn giao tiếp TTL, 1 wire
- Pins: VCC – nguồn, DAT – digital output, GND – nối đất

Chức năng chính trong đề tài: Thu thập các chỉ số về nhiệt độ và độ ẩm của môi trường.

1.3. Module cảm biến lửa 3 chân

Cảm biến chuyên dùng để phát hiện lửa sử dụng cảm biến hồng ngoại YG1006 với tốc độ đáp ứng nhanh và độ nhạy cao giúp dễ dàng phát hiện lửa hoặc nguồn sáng có bức xạ tương tự nên module này thường được ứng dụng trong các thiết bị báo cháy. Ngoài ra, module còn tích hợp IC LM393 để so sánh tạo mức tính hiệu và có thể chỉnh được độ nhạy bằng biến trở.



Hình 4. Module cảm biến lửa 3 chân

Thông số kỹ thuật:

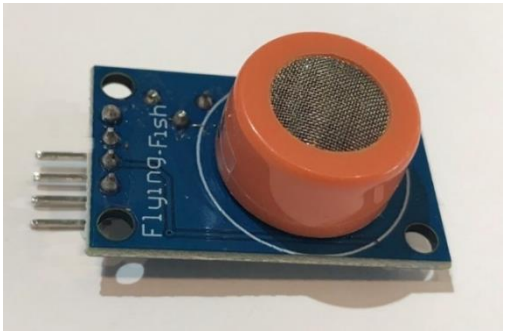
- Điện áp hoạt động: 3.3 – 5V DC

- Dòng tiêu thụ: 15 mA
- Bước sóng phát hiện lửa: 760 – 1100 nm
- Góc quét: 0 – 60°C
- Khoảng cách phát hiện: dưới 1 m (80cm)
- Nhiệt độ hoạt động: -25 – 85°C
- Kích thước: 3.2 x 1.4 cm
- Pins: VCC – nguồn, GND – nối đất, DO – digital output

Chức năng chính trong đề tài: Phát hiện ngọn lửa trong phạm vi cho phép.

1.4. Module cảm biến cồn MQ-3

Cảm biến MQ-3 được làm từ vật liệu SnO_2, dùng để đo nồng độ cồn. Vật liệu này có tính dẫn điện kém trong môi trường không khí sạch nhưng lại rất nhạy cảm với hơi cồn và có thể hoạt động ổn định trong thời gian dài. Module cảm biến cồn MQ-3 hoạt động dựa trên nguyên tắc điện trở thay đổi do C_2H_5OH bay hơi tác động lên lớp SnO_2 phủ trong cảm biến, khi nồng độ cồn càng cao thì giá trị cảm biến càng nhỏ. Theo thử nghiệm cho thấy cảm biến phát hiện nồng độ cồn còn chịu ảnh hưởng bởi điều kiện nhiệt độ [1].



Hình 5. Module cảm biến cồn MQ-3

Thông số kĩ thuật:

- Kích thước: 32 x 22 x 27 mm
- Chip chính: LM393, MQ-3 cảm biến khí
- Có 2 dạng tín hiệu đầu ra là dạng Analog và TTL
- Chất phản ứng: Ethanol (C_2H_5OH)
- Dải đo: 0.05 – 10 mg/L tương ứng với điện trở 1 – 8 M **Ω**
- Điện áp làm việc: dưới 24V
- Pins: VCC – nguồn, GND – nối đất, DOUT – digital output, AOUT – analog output

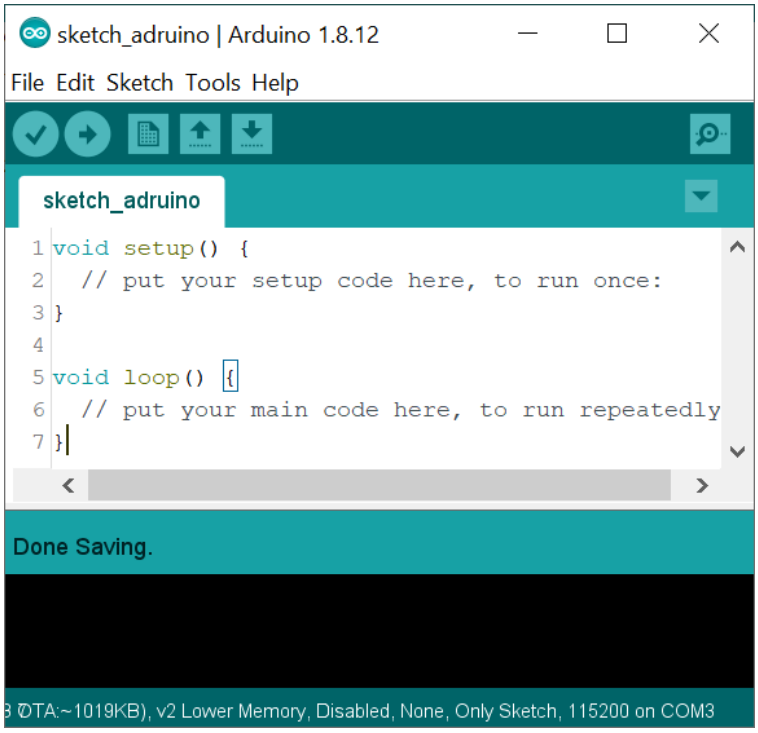
Chức năng chính trong đề tài: Thu thập thông số nồng độ cồn trong môi trường không khí, Trong đề tài này, đơn vị hiển thị nồng độ cồn trong không khí là ppm (parts per million).

2. PHẦN MỀM IDE VÀ THINGSPEAK

2.1. Arduino IDE

Arduino IDE là một môi trường phát triển Arduino mã nguồn mở và đa nền tảng được phát triển từ C và C++, cho phép người dùng nạp code và tải lên bo mạch để có thể sử dụng các cảm biến, linh kiện tùy chỉnh để phù hợp với từng mục đích nhu cầu khác nhau. Arduino IDE cung cấp đầy đủ các thư viện, các mô hình mẫu giúp người dùng có thể dễ dàng thao

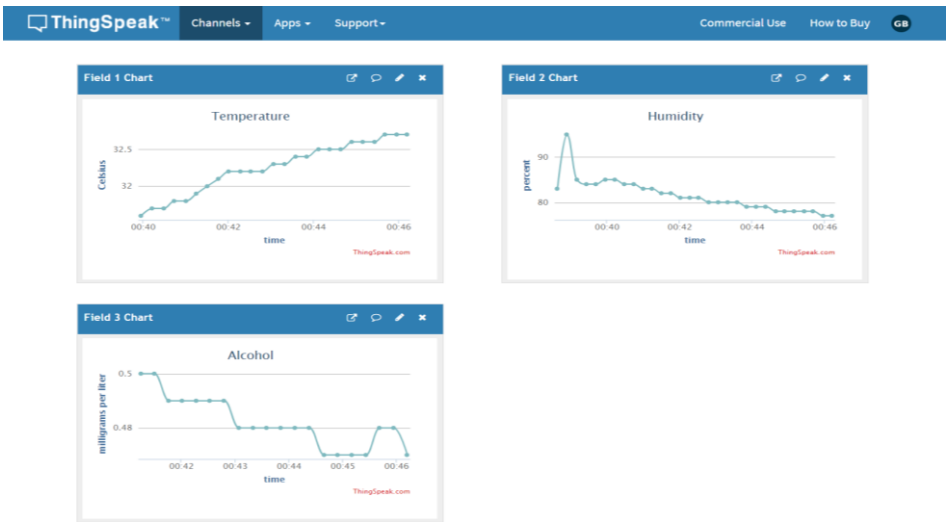
tác tạo và thử nghiệm các sản phẩm. Một mã bản phác thảo Arduino gồm có hai hàm chính: một hàm bắt đầu bản phác thảo chỉ thực thi một lần mỗi khi cấp nguồn hoặc reset board và một hàm lặp chính.



Hình 6. Giao diện Arduino IDE

2.2. ThingSpeak

ThingSpeak là một nền tảng Internet of Things, một cloud service khá phổ biến cho phép người dùng thu thập dữ liệu và lưu dữ liệu cảm biến trên cloud và phát triển các ứng dụng IoT. ThingSpeak cung cấp các ứng dụng phân tích và trực quan hóa dữ liệu của người dùng trong MATLAB, người dùng dễ dàng gửi dữ liệu và cung cấp các giao thức đồ họa hiển thị dữ liệu thông qua giao thức HTTP. Thiết bị hoặc ứng dụng của người dùng có thể giao tiếp với ThingSpeak bằng API RESTful và người dùng có thể giữ dữ liệu của mình ở chế độ riêng tư hoặc chế độ công khai.



Hình 7. ThingSpeak

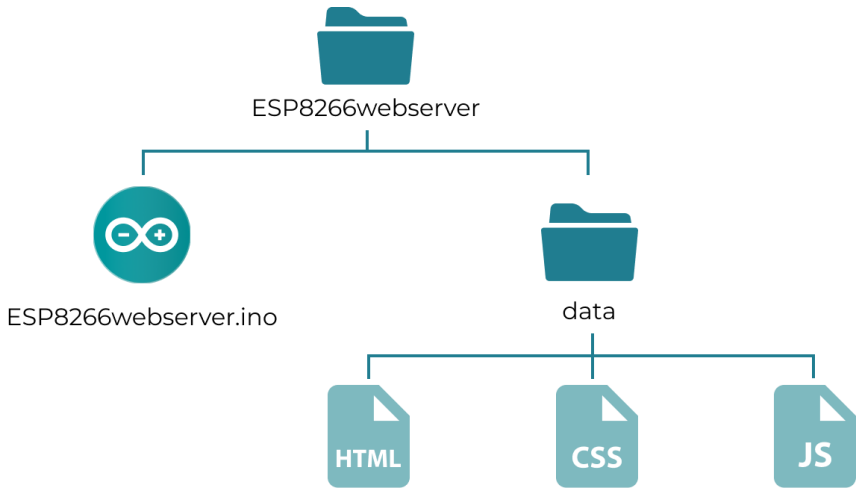
3. CÁC CÔNG NGHỆ ỨNG DỤNG

3.1. Hệ thống tệp flash giao diện ngoại vi nối tiếp SPIFFS

SPIFFS là viết tắt của Serial Peripheral Interface Flash File System, hay còn được gọi là hệ thống tập tin flash giao diện ngoại vi tối tiếp. Đây là một hệ thống tập tin nhẹ dành cho các bộ vi điều khiển chip flash được kết nối bằng bus SPI. SPIFFS cho phép truy cập và thao tác vào bộ nhớ flash một cách dễ dàng như một hệ thống tập tin bình thường trên máy tính nhưng đơn giản và hạn chế hơn. Hệ thống này hỗ trợ nhiều phiên bản lưu trữ khác nhau (1MB, 2MB hoặc 3MB), có thể được sử dụng để lưu trữ các tài liệu không thường xuyên thay đổi như: trang web, cấu hình, dữ liệu hiệu chuẩn cảm biến, v.v.

Bằng cách sử dụng SPIFFS chúng ta có thể tách riêng phần mã nguồn html, css, javascript, image, v.v ra khỏi mã chương trình phác thảo Arduino (Arduino sketch). Điều này có thể phần nào làm giảm tốc độ thực thi so với việc web được nhúng trực tiếp vào flash. Tuy nhiên, việc tách mã nguồn trang web sẽ giúp chương trình trở nên rõ ràng và dễ nhìn, đồng thời, việc thiết kế giao diện, chức năng của website cũng sẽ tiện lợi hơn.

Cấu trúc thư mục lưu bản phác thảo Arduino:

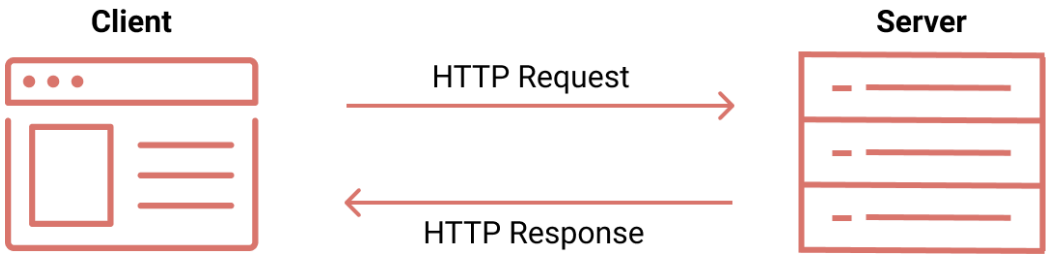


Hình 8. Cấu trúc thư mục lưu Arduino sketch

3.2. Giao thức HTTP

HTTP là viết tắt của từ HyperText Transfer Protocol, nghĩa là giao thức truyền tải siêu văn bản, là một giao thức ứng dụng của bộ giao thức TCP/IP dành cho nền tảng Internet.

HTTP hoạt động dựa trên mô hình server – client, dùng trong liên lạc thông tin giữa máy chủ cung cấp dịch vụ và máy khách sử dụng dịch vụ. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gởi yêu cầu (request) đến máy chủ (server) và chờ đợi trả lời (response) từ những máy chủ này.



Hình 9. Nguyên lý hoạt động HTTP

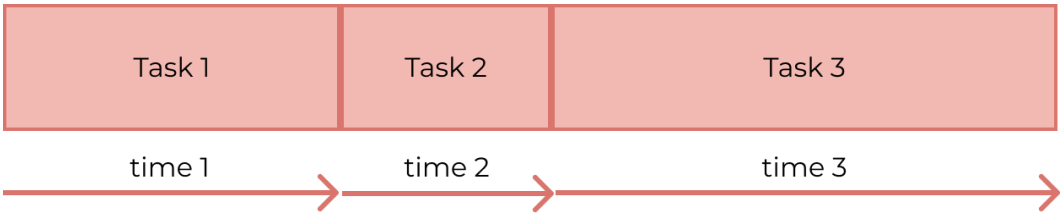
Cấu trúc của một HTTP Request và HTTP Response

Bảng 1. Cấu trúc HTTP Request và HTTP Response

HTTP Request	**HTTP Response**
Requset line: <method> <URL> <version> • Method: phương thức đang được sử dụng, thường là GET hoặc POST, ngoài ra còn có HEAD, PUT, DELETE, … • URL: đại chỉ định danh của tài nguyên • Version: phiên bản HTTP đang được sử dụng, thường là 1.1	Status line: <version> <status code> <status text> • Version: phiên HTTP cao nhất mà server hỗ trợ. • Status code: mã gồm 3 chữ số thể hiện trạng thái của kết nối. • Stauts text: mô tả status code.
Header fields: Gồm nhiều trường trường có cấu trúc như sau: <header field> : <value> cho phép gửi thêm các thông tin bổ sung về thông điệp yêu cầu hoặc phản hồi cũng như thông tin của máy gửi đi.	
Message body: thường trống	

3.3. Xử lý bất đồng bộ

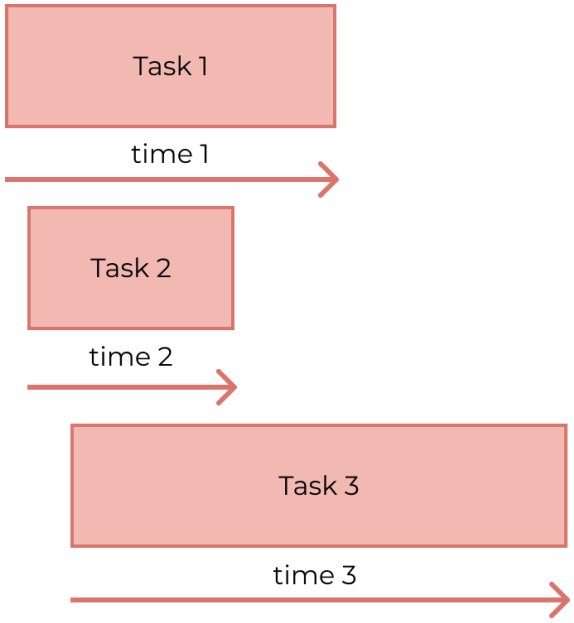
Xử lý đồng bộ (Synchronous) là một mô hình rất quen thuộc trong lập trình. Với xử lý đồng bộ, các công việc được sắp xếp theo một thứ tự được định sẵn. Trong một chương trình đồng bộ, các câu lệnh sẽ được thực hiện theo thứ tự từ trên xuống dưới, câu lệnh sau chỉ được thực hiện khi câu lệnh trước hoàn thành, và chỉ cần một câu lệnh sai thì cả chương trình sẽ lập tức bị dừng lại đồng thời sẽ hiện thị thông báo lỗi.



Hình 10. Minh họa mô hình xử lý đồng bộ

Khác với xử lý đồng bộ là xử lý bất đồng bộ (Asynchronous). Trong mô hình này, các công việc có thể được thực hiện cùng một lúc. Do vậy, công việc sau không phải chờ đợi công việc trước nữa. Chính vì thế mà đối khi sẽ có những trường hợp công việc sau kết thúc trước và cho ra kết quả trong khi công việc trước đó còn đang thực thi. Vậy nên, kết quả của chương trình có thể sẽ không theo đúng thứ tự trực quan của nó. Tuy nhiên, do hạn chế

tối đa việc "chờ đời" nên tổng thời gian thực hiện cả chương trình sẽ được rút ngắn một cách đáng kể.



Hình 11. Minh họa xử lý bất đồng bộ

Đối với lập trình server, việc xử lý bất đồng bộ không chỉ giảm thời gian đáp ứng mà còn khai thác được khả năng xử lý song song, giúp server có thể đáp ứng nhiều hơn một kết nối trong cùng một lúc. Từ các ưu điểm trên, nhóm quyết định thiết lập một máy chủ HTTP bất đồng bộ trên ESP8266 để chạy nền tảng web sử dụng thư viện ESPAsyncWebServer [2], thư viện hỗ trợ lập trình máy chủ HTTP không đồng bộ và máy chủ WebSocket.

3.4. Các ngôn ngữ, công nghệ web khác

Các ngôn ngữ và công nghệ web được sử dụng trong đề tài này gồm có HTML, CSS, JavaScript và AJAX.

HTML là chữ viết tắt cho cụm từ HyperText Markup Language (có nghĩa là ngôn ngữ đánh dấu siêu văn bản) là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web hiện đang được sử dụng để tạo ra tất cả các website trên thế giới. HTML mô tả cấu trúc của văn bản hay nội dung trang web, bênh cạnh đó còn có thể giúp người lập trình quy định màu sắc, thiết kế của các thành phần có trong trang web.

CSS hay Cascading Style Sheet language, ngôn ngữ tạo phong cách cho trang web, được dùng để định dạng các kiểu cho những yếu tố được viết dưới dạng ngôn ngữ đánh dấu, như là HTML. CSS có chức năng điều khiển định dạng của nhiều trang web cùng lúc giúp tiết kiệm công sức cho lập trình viên, ngoài ra còn giúp tách biệt phần quy định cách hiển thị của trang web với nội dung chính của trang. Phương thức hoạt động của CSS là sẽ tìm kiếm các vùng được chọn, có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác và sau đó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

JavaScript là một ngôn ngữ lập trình hoặc ngôn ngữ kịch bản cho phép triển khai các chức năng phức tạp trên trang ví dụ như trình chiếu ảnh, đồ họa hoạt hình hay hình thức tương tác, v.v. Mô hình đối tượng tài liệu (DOM) cho phép tạo và kiểm soát nội dung của trang

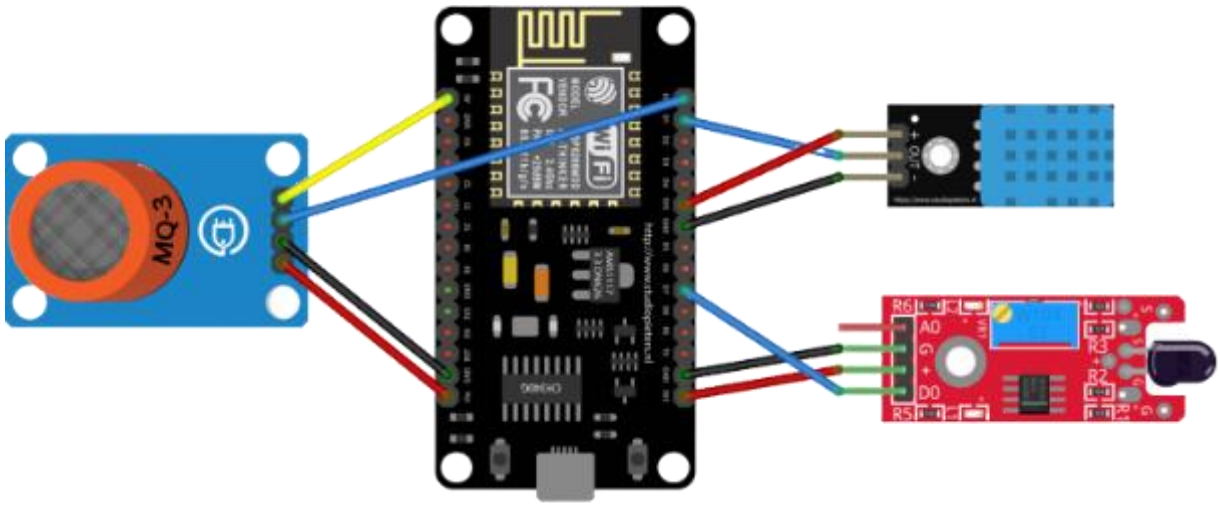
web động với JS, có thể thay đổi nội dung trang HTML, có thể hiển thị / ẩn các phần tử trang HTML, có thể thay đổi các thuộc tính HTML của phần tử HTML, v.v. mà không yêu cầu người dùng tải lại trang web theo cách thủ công.

AJAX (Asynchronous JavaScript and XML) là một bộ các kỹ thuật thiết kế web giúp cho các ứng dụng web hoạt động bất đồng bộ. Và trong đó, yếu tố quyết định của AJAX là đối tượng XMLHttpRequest. Tất cả các trình duyết hiện tại đều hỗ trợ đối tượng XMLHttpRequest. Đối tượng XMLHttpRequest cho phép các trang web được cập nhật không đồng bộ bằng cách trao đổi dữ liệu với máy chủ web từ phía sau và điều này có nghĩa là có thể cập nhật các phần của trang web mà không cần tải lại toàn bộ trang web.

PHẦN III. THIẾT KẾ VÀ CÀI ĐẶT

1. Khối điều khiển và cảm biến

Khối điều khiển và cảm biến gồm có 4 linh kiện chính: ESP8266, DHT11, MQ-3 và cảm biến lửa. Trong đó ESP8266 là mạch xử lý chính, giữ nhiệm vụ cấp nguồn, điều khiển và nhận dữ liệu từ các cảm biến kể trên. Nguồn cấp cho mạch xử lý ESP8266 có điện áp 5V sẽ được cắm trực tiếp từ laptop, và các cảm biến khác sẽ được cấp nguồn thông qua ESP8266. Dữ liệu sẽ được đọc từ các chân digital và analog có kết nối với cảm biến. Sơ đồ nối chân của các linh kiện trong khối điều khiển được minh họa trong hình vẽ bên dưới:



Hình 12. Sơ đồ nối chân của khối điều khiển và cảm biến

Bảng 2. Bảng thông số nối chân của khối điều khiển và cảm biến

STT	Sensor	Sensor pin	ESP8266 Pin
1	DHT11	VCC	3.3V
		GND	GND
		DAT	D1
2	Flame Sensor	VCC	3.3V
		GND	GND
		DO	D7
3	MQ-3	VCC	Vin (5V)
		GND	GND
		AOUT	A0
		DOUT	D0

2. LẬP TRÌNH VÀ CÀI ĐẶT

2.1. Tập tin JavaScript dùng để tự động cập nhật các chỉ số môi trường

Dưới đây là một phần của mã lệnh trong tập tin JavaScript dùng để cập nhật các chỉ số nhiệt độ, độ ẩm và nồng độ cồn một cách tự động mà không cần phải chạy lại toàn bộ trang web.

Khai báo biến `time` lưu trữ thời gian tự động cập nhật và biến `flame` dùng để lưu trữ trạng thái tín hiệu báo cháy.

```
var time = 1000; // 1000 millis seconds = 1 second
var flame = 1; // flame: 1-Fire, 0-No flame
```

Đoạn mã chương trình sau đây sẽ mô tả việc tự động cập nhật chỉ số nhiệt độ, làm tương tự với chỉ số độ ẩm và nồng độ cồn. Sử dụng phương thức `setInterval(F(), T)` dùng để thiết lập độ trễ cho hàm `F` nằm ở tham số đầu tiên mộ khoản thời gian bằng `T` milliseconds. Như vậy, hàm `function()` được định nghĩa ở bên dưới sẽ được thực thi liên tục trong mỗi giây.

```
setInterval(function() {
```

Khởi tạo một đối tượng `XMLHttpRequest()` để chuẩn bị gửi yêu đến server.

```
  var xhttp = new XMLHttpRequest();
```

Sau khi nhận được phản hồi từ server, tự động cập nhật giá trị của phần tử có id là “temperature” trong tập tin HTML lại cho phù hợp.

```
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
      document.getElementById("temperature").innerHTML =
this.responseText;
  };
```

Gửi một yêu cầu GET với URL “/temperature” để đọc giá trị nhiệt độ mới nhất.

```
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, time);
```

Tương tự như trên, đoạn mã dưới đây dùng để tự động hiển thị cảnh báo cháy ra màng hình browser mỗi khi có tín hiệu phát hiện được lửa.

```
setInterval(function() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
    {
      var newF = this.responseText;
      if (newF == 0 && newF != flame)
        document.getElementById('flame-modal').style.display =
"flex";
      flame = newF;
    }
```

```
  };
  xhttp.open("GET", "/flame", true);
  xhttp.send();
}, time);
```

2.2. Bản phát thảo Adruino chính của hệ thống

Sau đây là mã lệnh của bản phát thảo chính của hệ thống, ESP8266webserver.ino, và các giải thích cụ thể. Đầu tiên, thêm các thư viện cần thiết để có thể kết nối WiFi, sử dụng cảm biến DHT11, SPIFFS và lập trình webserver bất đồng bộ.

```
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <FS.h>
#include <DHT.h>
#include <Wire.h>
```

Nhập tên và mật khẩu của mạng WiFi để ESP8266 có thể kết nối được với Internet.

```
const char* ssid = ""; // WiFi SSID
const char* password = ""; // WiFi password
```

Tạo một đối tượng `AsynWebServer` đặt tên là `server` để lắng nghe yêu cầu trên cổng mặc định của HTTP là 80.

```
AsyncWebServer server(80);
```

Định nghĩa các biến cần thiết gồm có chân pin nối với các chân digital output của các cảm biến và các biến để lưu giá trị của các chỉ số môi trường.

```
#define dhtPin D1
#define dhtType DHT11
#define flamePin D7
DHT dht(dhtPin, dhtType);
float t = 0.0; // temperature
float h = 0.0; // humidity
float a = 0.0; // alcohol
int f = HIGH; // flame: HIGH: no flame; LOW: fire!
```

Định nghĩa các thông số liên quan đến kênh ThingSpeak được sử dụng.

```
const int channelID = 1063062;
String writeAPIKey = "RAOKCHK6EQOSPSTR"; // write API key for
your ThingSpeak Channel
const char* thingspeak_server = "api.thingspeak.com";
WiFiClient client;
```

Hàm `processor()` có chức năng xử lý việc gán giá trị thu được từ cảm biến vào các placeholder trong tệp HTML. Placeholder được định nghĩa trong hai dấu phần trăm, ví dụ: `%TEMPERATURE%`

```
String processor(const String& var) {
  if (var == "TEMPERATURE") return String(t);
  else
    if (var == "HUMIDITY") return String(h);
```

```
    else
      if (var == "ALCOHOL") return String(a);
      else
        if (var == "FLAME") return String(f);
  return String();
}
```

Bắt đầu hàm `setup()` bằng việc khởi tạo Serial Monitor và các cảm biến, GPIO làm đầu đọc dữ liệu.

```
void setup() {
  Serial.begin(115200);
  // Initialize sensor
  dht.begin();
  pinMode(flamePin, INPUT);
```

Khởi tạo SPIFFS.

```
  // Initialize SPIFFS
  if (!SPIFFS.begin()){
    Serial.println("Error SPIFFS");
    return;
  }
```

Kết nối WiFi cho ESP8266 và hiển thị IP ra màn hình Serial.

```
  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.println("Connecting");
    delay(5000);
  }
  Serial.println("Connected");
  Serial.println(WiFi.localIP());
```

Sử dụng thư viện ESPAsyncWebServer để cấu hình các tuyến đường (route) để máy chủ lắng nghe và xử lý các HTTP Request. Sử dụng phương thức `on` cho đối tượng `server` như sau:

```
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
      request->send(SPIFFS, "/index.html", String(), false,
processor);
    });
```

Khi máy chủ nhận được yêu cầu trên đường dẫn gốc "/", máy sẽ gửi tập tin `index.html` đã được lưu trong hệ thống tập tin flash của ESP8266 đến máy client. Tham số cuối của hàm `send()` là hàm `processor` giúp ta có thể thay thế các placeholder tương ứng trong các tệp HTML. Làm tương tự khi máy chủ nhận được yêu cầu trên các đường dẫn khác:

```
  server.on("/index.html", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/index.html", String(), false,
processor);
```

```
    });
  server.on("/content.html", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/content.html", String(), false,
processor);
    });
  server.on("/chart.html", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/chart.html", String(), false,
processor);
    });
```

Do các trang html có sử dụng tham chiếu đến tập tin CSS và JavaScript, nên máy khách cũng sẽ đưa ra yêu cầu những tập tin đó. Do vậy ta cần lập trình cho máy chủ sẽ gửi tập tin tương ứng cho máy khách:

```
  server.on("/css/styles.css", HTTP_GET,
[](AsyncWebServerRequest *request) {
      request->send(SPIFFS, "/css/styles.css", "text/css");
    });
  server.on("/css/index.css", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/css/index.css", "text/css");
    });
  server.on("/css/content.css", HTTP_GET,
[](AsyncWebServerRequest *request) {
      request->send(SPIFFS, "/css/content.css", "text/css");
    });
  server.on("/css/chart.css", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/css/chart.css", "text/css");
    });
  server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send(SPIFFS, "/script.js", "text/javascript");
    });
```

Trong tập tin JavaScript có sử dụng một đoạn mã yêu cầu các thông số nhiệt độ, độ ẩm, độ cồn và tín hiệu có lửa trên các tuyến /temperature, /humidity, /alcohol và /flame mỗi giây. Khi nhận được các yêu cầu như vậy, server sẽ gửi các giá trị cảm biến đã được lưu trong các biến `t`, `h`, `a` và `f` đã được định nghĩa và cập nhật giá trị thường xuyên dưới dạng plain text thông qua phương thức `c_str()`.

```
  // Response to XMLHttpRequest
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest
*request){
      request->send_P(200, "text/plain", String(t).c_str());
    });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest
*request){
      request->send_P(200, "text/plain", String(h).c_str());
    });
```

```
  server.on("/alcohol", HTTP_GET, [](AsyncWebServerRequest
*request){
      request->send_P(200, "text/plain", String(a).c_str());
    });
  server.on("/flame", HTTP_GET, [](AsyncWebServerRequest
*request) {
      request->send_P(200, "text/plain", String(f).c_str());
    });
```

Sử dụng phương thức `begin()` trên đối tượng `server` để máy chủ bắt đầu lắng nghe yêu cầu từ máy khách.

```
  // Start server
  server.begin();
}
```

Hàm `analogToPPM()` dùng để chuyển đổi giá trị analog từ MQ-3 sang đơn vị PPM [3].

```
// Caculate analog value to PPM
float analogToPPM(float sensor_value) {
  float sensor_volt = (float) sensor_value / 1024 * 5.0;
  float RS = ((5 - sensor_volt)/sensor_volt);
  float R0 = 0.02;
  float ratio = RS/R0;
  float lgPPM = (log10(ratio)* (-2.6)) + 2.7;
  return (pow(10, lgPPM));
}
```

Hàm `loop()` dùng để cập nhật các chỉ số từ cảm biến và gửi dữ liệu đến ThingSpeak.

```
void loop() {
  delay(1000); //Update every 1 second
```

Đọc các thông số từ cảm biến.

```
  float newT = dht.readTemperature();
  float newH = dht.readHumidity();
  float newA = analogRead(A0);
  newA = analogToPPM(newA);
  int newF = digitalRead(flamePin);
```

Hiển thị cảnh báo phát hiện có lửa ra màn hình Serial.

```
  f = newF;
  if (f == LOW) Serial.println("FIRE");
```

Kiểm tra tính hợp lệ của dữ liệu đọc được từ cảm biến.

```
  if (isnan(newT) || isnan(newH))
    Serial.println("Fail to read from DHT sensor");
  else
    if (isnan(newA))
      Serial.println("Fail to read alcohol value");
    else {
```

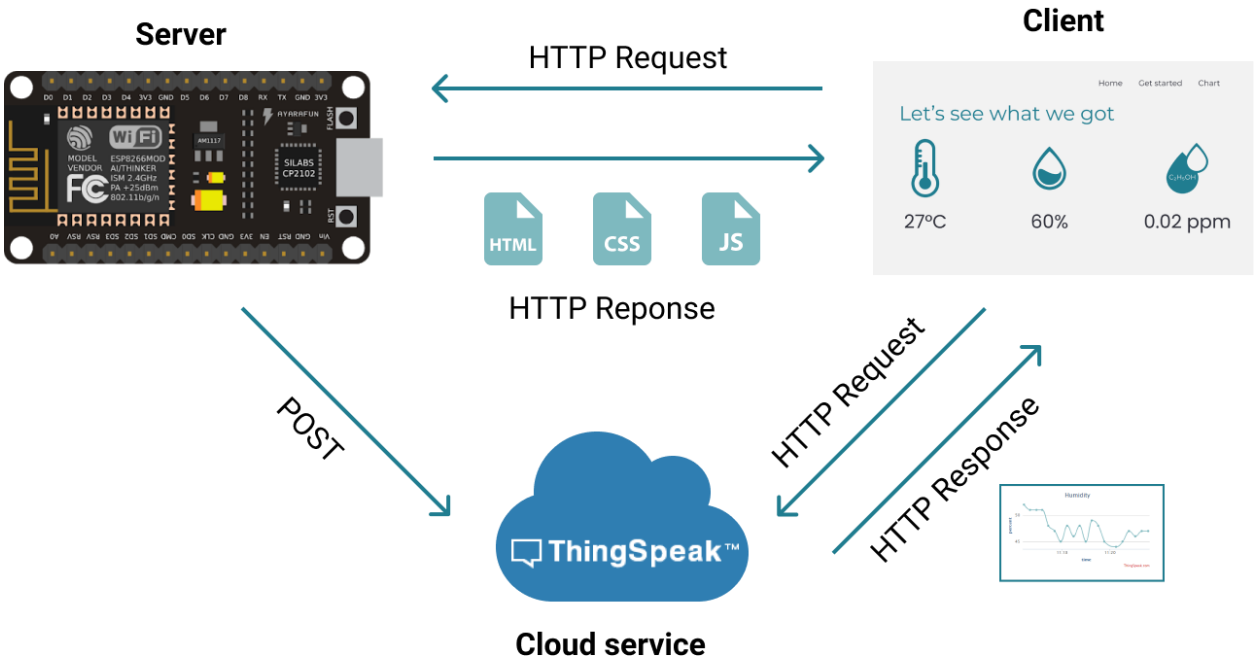
Khi dữ liệu đọc vào là hợp lệ, cập nhật lại các chỉ số của môi trường vào biến toàn cục tương ứng đồng thời hiển thị ra màn hình Serial.

```
      t = newT;
      h = newH;
      a = newA;
      Serial.print(t); Serial.print(" "); Serial.print(h);
Serial.print(" "); Serial.println(a);
```

Tạo kết nối với ThingSpeak và tiến hành gửi dữ liệu.

```
      if (client.connect(thingspeak_server, 80)){
        String str = "field1=" + String(t, 2) + "&field2=" +
String(h, 2) + "&field3=" + String(a, 2);
        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + writeAPIKey +
"\n");
        client.print("Content-Type: application/x-www-form-
urlencoded\n");
        client.print("Content-Length: ");
        client.print(str.length());
        client.print("\n\n");
        client.print(str);
        client.print("\n\n");
      }
      client.stop();
    }
```

3. MÔ HÌNH HỆ THỐNG VÀ NGUYÊN LÝ HOẠT ĐỘNG

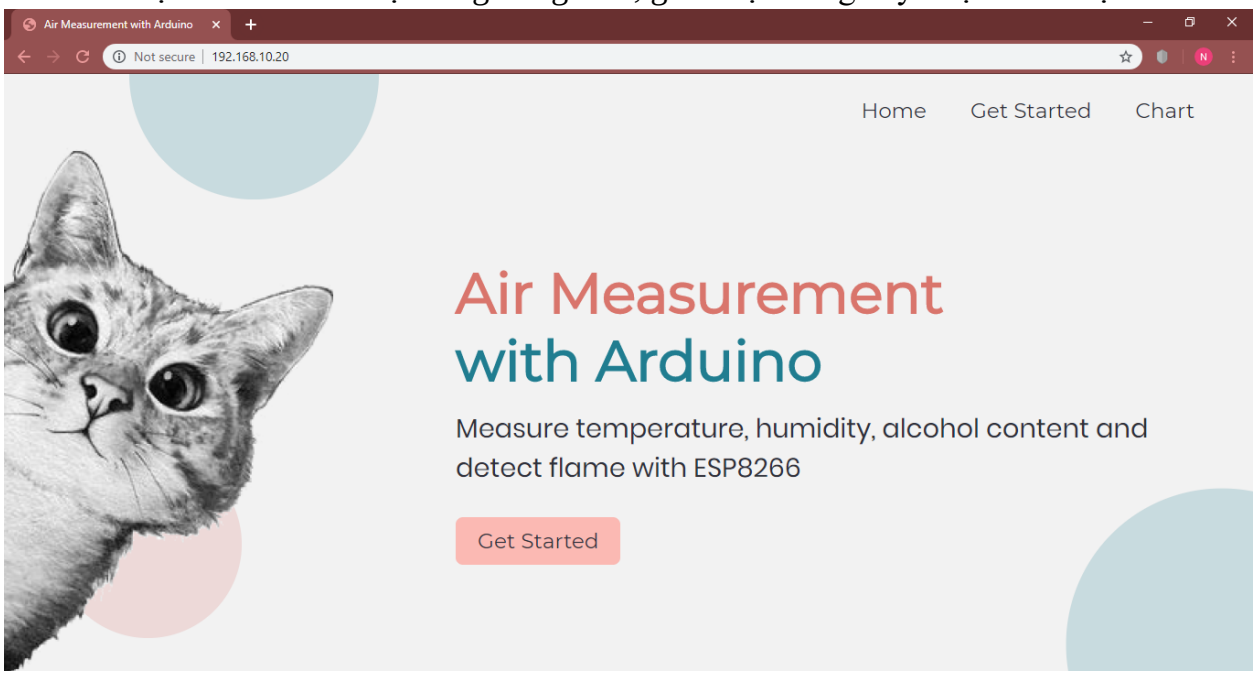


Hình 13. Minh họa nguyên lý hoạt động của hệ thống

Ngoài vai trò là chip xử lý chính, giữ nhiệm vụ điều khiển và thu thập dữ liệu từ các cảm biến, ESP8266 còn là một server HTTP bất đồng bộ. Khi có yêu cầu từ máy khách, server sẽ gửi các tài liệu liên quan đến trang web . Ngoài ra hệ thống còn sử dụng thêm một bên thứ ba là ThingSpeak để lưu trữ và vẽ biểu đồ nhằm trực quan hóa các dữ liệu thu thập được. Quá trình đó diễn ra như sau, khi thu thập các số liệu, ESP8266 cũng sẽ tạo một kết nối và gửi yêu cầu dạng POST để gửi dữ liệu lên ThingSpeak. Sau đó, ThingSpeak và máy client cũng sẽ dùng giao thức HTTP để truyền thông điệp cho nhau nhằm mục đích hiển thị được biểu đồ ra màn hình browser.

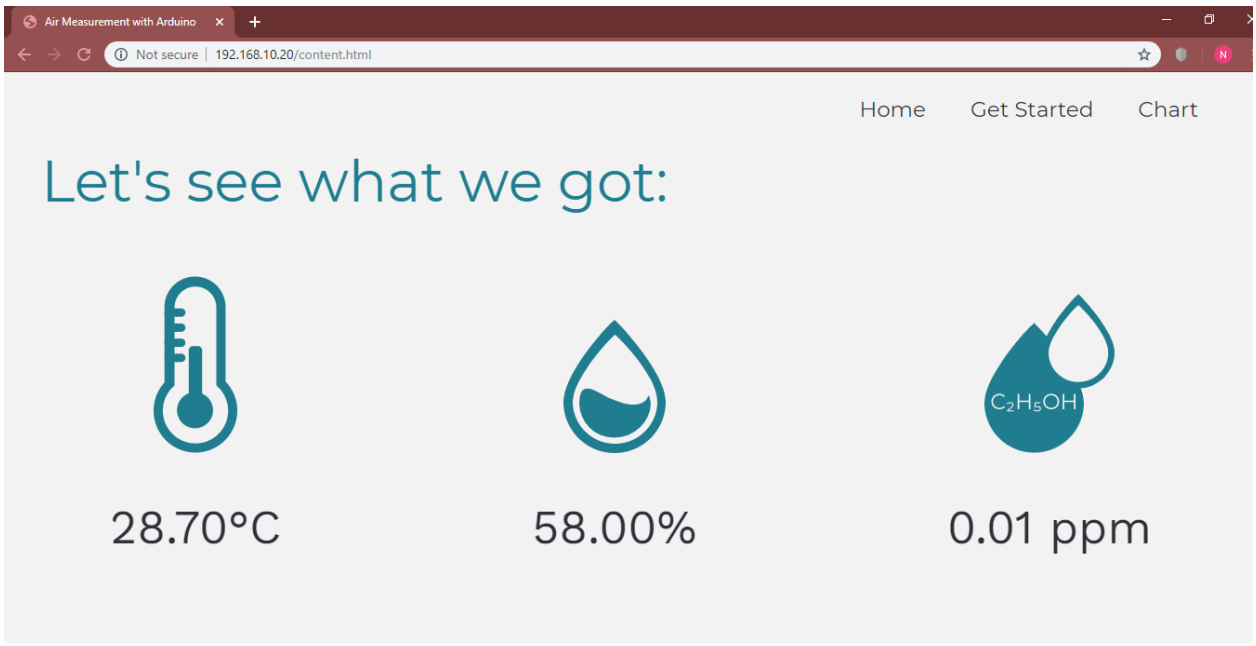
4. KẾT QUẢ ĐẠT ĐƯỢC

Giao diện website hiển thị các chỉ số về môi trường gồm có 3 trang chính. Đầu tiên là trang chủ hiển thị tên và tóm tắt nội dung trang web, giao diện trang này được minh họa như sau:



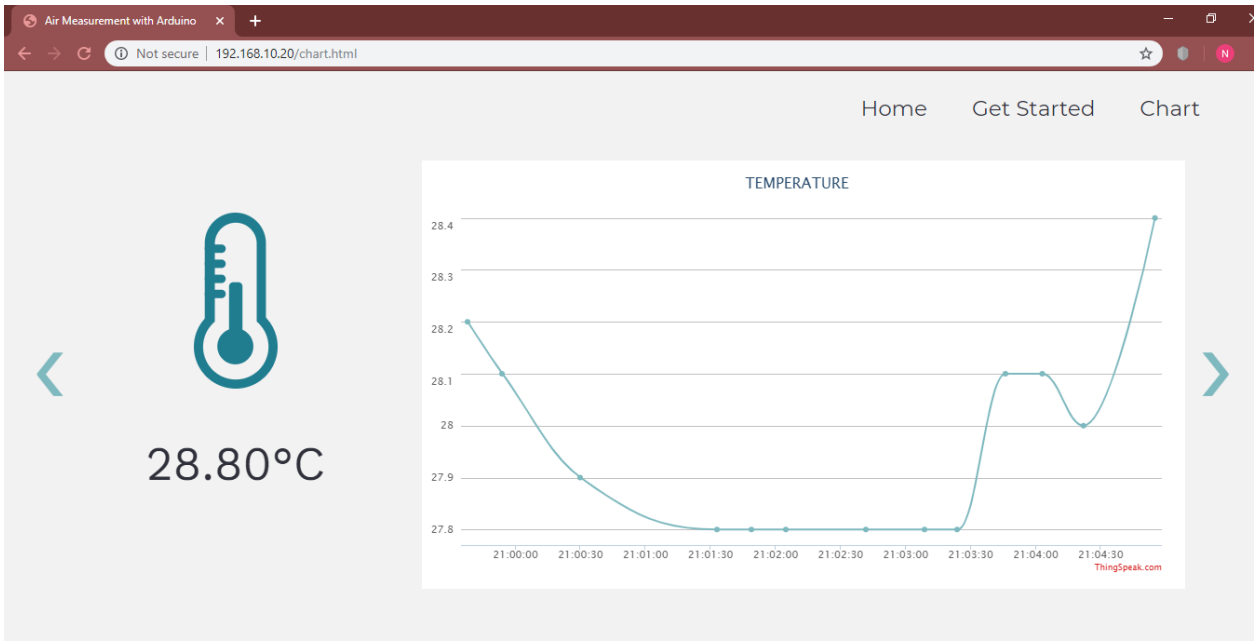
Hình 14. Giao diện trang chủ website

Trang thứ hai có chức năng hiển thị ba dữ liệu nhiệt độ, độ ẩm và độ cồn trong môi trường không khí. Cả ba thông số này sẽ được cập nhật liên tục trong thời gian thực.



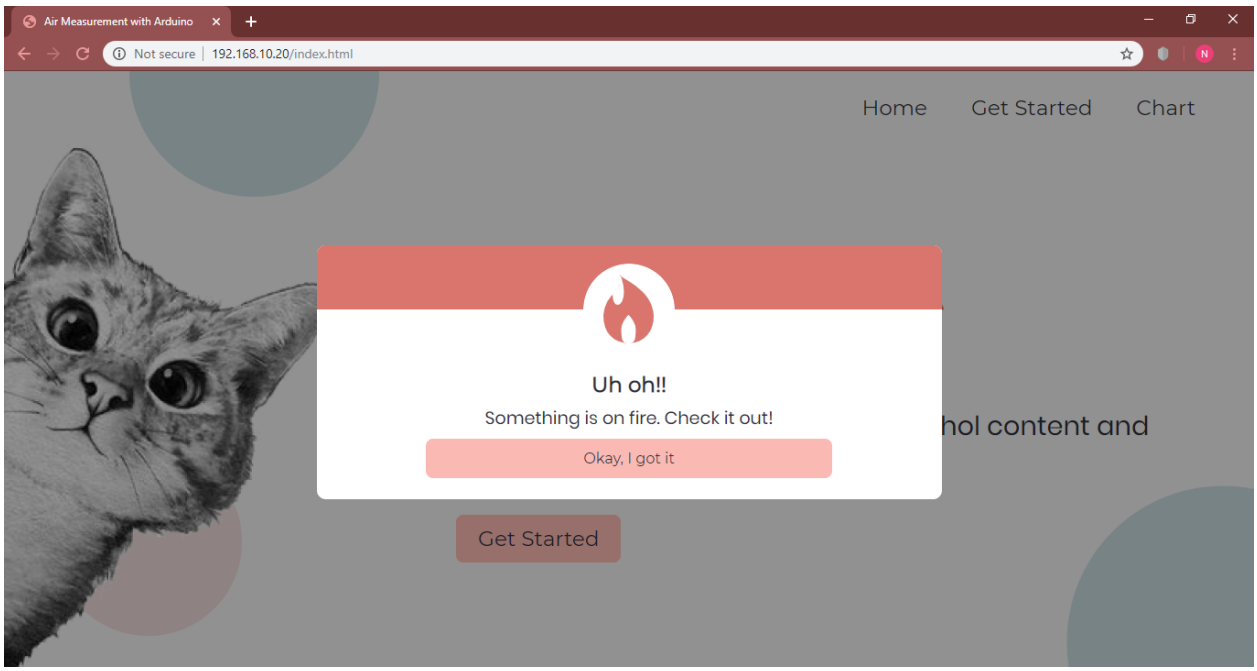
Hình 15. Giao diện trang hiển thị chỉ số môi trường

Trang thứ ba là trang hiển thị bảng trình chiếu với từng chỉ số môi trường và trực quan hóa các dữ liệu đã được thu thập thông qua biểu đồ tương ứng được nhúng từ Thingspeak.



Hình 16. Giao diện trang hiển thị biểu đồ trực quan tương ứng với từng thông số

Khi hệ thống nhận được tín hiệu báo cháy sẽ tự động hiển thị một hộp thoại cảnh báo cho người dùng. Chức năng này đều hoạt động tốt ở cả ba trang kể trên.



Hình 17. Hộp thoại chức năng phát hiện và cảnh báo cháy

PHẦN IV. KẾT LUẬN

Sau thời gian tìm hiểu và thử nghiệm nhóm đã xây dựng được sản phẩm là một hệ thống tương đối hoàn chỉnh có thể thu thập được những chỉ số cơ bản từ môi trường và hiển thị qua website. Đây là một sản phẩm có tính ứng dụng cao và có thể mở rộng ra trong nhiều đề tài cũng như các lĩnh vực khác. Qua đề tài này chúng em đã được tiếp cận và hiểu biết hơn Arduino và một số công nghệ web đang được sử dụng phổ biến hiện nay.

Bên cạnh đó nhóm chúng em cũng đề xuất một số hướng phát triển cho đề tài thêm phần hoàn thiện hơn. Đầu tiên, nhóm đang sử dụng Webserver bất đồng bộ được lập trình trực tiếp trên mạch xử lý ESP8266 vì có thể khai thác được ưu điểm ổn định và dễ dàng cập nhật, chỉnh sửa trong quá trình thử nghiệm sản phẩm, phù hợp với mục tiêu làm quen và tìm hiểu về lập trình Arduino. Tuy nhiên, nếu đặt mục tiêu ứng dụng đề tài vào thực tiễn, hệ thống nên có một Webserver nằm ngoài ESP8266 và một khối nguồn riêng cho ESP8266 để có thể hoạt động độc lập trong thời gian dài. Bên cạnh đó, tùy vào mục đích sử dụng mà có thể linh hoạt thay đổi các cảm biến cho phù hợp và có thể tích hợp thêm một số linh kiện ngoại vi, chẳng hạn như màn hình LCD, đèn LED hay loa.