**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



1. ĐỒ ÁN CHUYÊN NGÀNH

**Đề tài : Phát triển giải pháp xuất nhập cho**

**nhà thông minh tích hợp điện toán đám mây**

***Giáo viên hướng dẫn:* PGS.TS Lê Trung Quân**

***Sinh viên thực hiện:* Bùi Thanh Hải 14520236**

**Nguyễn Trung Duyên 14520229**

**Lớp: MMTT2014**

**Khoa: Mạng máy tính và truyền thông**

TP. Hồ Chí Minh, ngày tháng năm

**LỜI CÁM ƠN**

Nội dung

**DANH SÁCH HÌNH VẼ**

[Hình 3.1 : Board Arduino ESP8266 Wifi UNO 12](#_Toc510640274)

[Hình 3.2 : OLED 13](#_Toc510640275)

[Hình 3.3 : DHT11 13](#_Toc510640276)

[Hình 3.4 : Cảm biến lửa 14](#_Toc510640277)

[Hình 3.5 : Cảm biến chuyển động 14](#_Toc510640278)

[Hình 3.6 : Cảm biến khoảng cách 15](#_Toc510640279)

[Hình 3.7 : Động cơ servo 15](#_Toc510640280)

[Hình 3.8 : RFID 16](#_Toc510640281)

[Hình 4.1: Mô hình cơ bản của giao thức MQTT 19](#_Toc510640282)

[Hình 4.2: Cơ chế hoạt động của giao thức MQTT 20](#_Toc510640283)

[Hình 6.1 : Tạo một project (1) 25](#_Toc510640284)

[Hình 6.2 : Tạo một project (2) 26](#_Toc510640285)

[Hình 6.3 : Kết quả sau khi tạo một project 26](#_Toc510640286)

[Hình 6.4 : Tạo Model 27](#_Toc510640287)

[Hình 6.5 : Nội dung file Data.cs 27](#_Toc510640288)

[Hình 6.6 : Nội dung file IDataRepo.cs 28](#_Toc510640289)

[Hình 6.7 : Nội dung file DataRepo.cs (1) 28](#_Toc510640290)

[Hình 6.8 : Nội dung file DataRepo.cs (2) 29](#_Toc510640291)

[Hình 6.9 : Tạo Controller 29](#_Toc510640292)

[Hình 6.10 : Nội dung file PinController.cs (1) 30](#_Toc510640293)

[Hình 6.11 : Nội dung file PinController.cs (2) 30](#_Toc510640294)

[Hình 6.12 : Nội dung file PinController.cs (3) 31](#_Toc510640295)

[Hình 6.13 : Nội dung file lauchSetting.json 32](#_Toc510640296)

[Hình 6.14 : Run project bằng ServerAPI 32](#_Toc510640297)

[Hình 6.15 : Kết quả sai khi chạy Server 33](#_Toc510640298)

[Hình 6.16 : Action Get() 33](#_Toc510640299)

[Hình 6.17 : Action Get(int pin) 33](#_Toc510640300)

[Hình 6.18 : Action Post(Data dt) 34](#_Toc510640301)

[Hình 6.19 : Chạy Ngrok 35](#_Toc510640302)

[Hình 6.20 : Kết quả sau khi chạy Ngrok 35](#_Toc510640303)

[Hình 6.21 : Kết quả kiểm tra kết nối Ngrok trên điện thoại 36](#_Toc510640304)

[Hình 7.1 : Hàm kết nối Wifi 38](#_Toc510640305)

[Hình 7.2 : Vị trí đặt hàm setupWifi 38](#_Toc510640306)

[Hình 7.3 : Hàm getHttpResponse() 39](#_Toc510640307)

[Hình 7.4 : Hàm Post() 40](#_Toc510640308)

[Hình 7.5 : Hàm displayStringAsJson() 40](#_Toc510640309)

[Hình 7.6 : Hàm displayStringAsJsonArray() 41](#_Toc510640310)

[Hình 7.7 : Hàm convertStringToJson() 41](#_Toc510640311)

[Hình 7.8 : Hàm getIndexAtJsonArray() 42](#_Toc510640312)

[Hình 7.9 : Hàm setup\_rfid() 43](#_Toc510640313)

[Hình 7.10 : Hàm dump\_byte\_array() 43](#_Toc510640314)

[Hình 7.11 : Hàm startRFID() 43](#_Toc510640315)

**DANH SÁCH BẢNG**

[Bảng 2.1 : Lịch sử ra đời của ASP.NET Framework 9](#_Toc510640328)

[Bảng 2.2 : Lịch sử ra đời của ASP.NET Core 9](#_Toc510640329)

[Bảng 2.3 : So sánh ASP.NET Framework và ASP.NET Core 10](#_Toc510640330)

[Bảng 2.4 : So sánh ASP.NET Core và NodeJS 12](#_Toc510640331)

1. CHƯƠNG 1: GIỚI THIỆU
2. CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/API
   1. ASP.NET
      1. Tổng quan
3. ASP.NET là nền tảng phát triển web (web application frameword), cung cấp một mô hình lập trình, cơ sở hạ tầng phần mềm toàn diện và các dịch vụ cần thiết để xây dụng các ứng dụng web động mạnh mẽ cho máy tính cũng như trên các thiết bị di động.
4. ASP.NET là một phần của nền tảng Microsoft.NET. Ứng dụng ASP.NET được biên dịch mã, được viết bằng cách sử dụng mở rộng và tái sử dụng các thành phần hoặc đối tượng trong nền tảng NET. Các mã này được sử dụng cho toàn bộ hệ thống phân cấp của các class trong .NET
5. Các ứng dụng ASP.NET có thể được viết bằng bất kỳ ngôn ngữ nào sao đây:

* C#
* Visual Basic.Net
* Jscript
* J#

1. ASP.NET được phát hành đầu tiên vào 12/2/2012 với tên gọi ASP.NET Framework v1.0 cùng với Visual Studio.NET 2002
   * 1. ASP.NET Framework
        1. Tổng quan
2. .NET Framework là một nền tảng lập trình và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows được phát triển bởi Microsoft. Các chương trình được viết trên nền.NET Framework sẽ được triển khai trong môi trường phần mềm (ngược lại với môi trường phần cứng) được biết đến với tên Common Language Runtime (CLR). Môi trường phần mềm này là một máy ảo trong đó cung cấp các dịch vụ như an ninh phần mềm (security), quản lý bộ nhớ (memory management), và các xử lý lỗi ngoại lệ (exception handling).
3. .NET framework bao gồm tập các thư viện lập trình lớn, và những thư viện này hỗ trợ việc xây dựng các chương trình phần mềm như lập trình giao diện; truy cập, kết nối cơ sở dữ liệu; ứng dụng web; các giải thuật, cấu trúc dữ liệu; giao tiếp mạng... CLR cùng với bộ thư viện này là 2 thành phần chính của.NET framework.
4. .NET framework đơn giản hóa việc viết ứng dụng bằng cách cung cấp nhiều thành phần được thiết kế sẵn, người lập trình chỉ cần học cách sử dụng và tùy theo sự sáng tạo mà gắn kết các thành phần đó lại với nhau. Nhiều công cụ được tạo ra để hỗ trợ xây dựng ứng dụng.NET, và IDE (Integrated Developement Environment) được phát triển và hỗ trợ bởi chính Microsoft là Visual Studio.
   * + 1. Lịch sử phát triển

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phiên bản** | **Số hiệu phiên bản** | **Ngày phát hành** | **Visual Studio** | **Được phát hành kèm theo** |
| [1.0](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_1.0) | 1.0.3705.0 | 13 tháng 2 năm 2002 | [Visual Studio](https://vi.wikipedia.org/wiki/Visual_Studio).NET | Windows XP Tablet and Media Center Editions[[1]](https://vi.wikipedia.org/wiki/.NET_Framework#cite_note-1) |
| [1.1](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_1.1) | 1.1.4322.573 | 24 tháng 4 năm 2003 | Visual Studio.NET 2003 | Windows Server 2003 |
| [2.0](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_2.0) | 2.0.50727.42 | 7 tháng 11 năm 2005 | Visual Studio 2005 | Windows Server 2003 R2 |
| [3.0](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_3.0) | 3.0.4506.30 | 6 tháng 11 năm 2006 |  | Windows Vista, Windows Server 2008 |
| [3.5](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_3.5) | 3.5.21022.8 | 19 tháng 11 năm 2007 | Visual Studio 2008 | Windows 7, Windows Server 2008 R2 |
| [4.0](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_4) | 4.0.30319.1 | 12 tháng 4 năm 2010 | Visual Studio 2010 |  |
| [4.5](https://vi.wikipedia.org/wiki/.NET_Framework#.NET_Framework_version_history#.NET_Framework_4.5) | 4.5.50709 | 15 tháng 8 năm 2012 | Visual Studio 2012 | Windows 8, Windows Server 2012 |
| 4.6 |  | 20 tháng 7 năm 2015 | Visual Studio 2015 | Windows 10 |

Bảng 2.1 : Lịch sử ra đời của ASP.NET Framework

* + 1. ASP.NET Core
       1. Tổng quan

1. ASP.NET Core là một open-source mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối đám mây, giống như web apps, IoT và backend cho mobile. Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework.
2. Nó được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise. Nó bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy bạn giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bạn.
3. Điểm đặc biệt của ASP.NET Core là ta có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux . Điều mà ASP.NET Framework không làm được.
   * + 1. Lịch sử ra đời

|  |  |
| --- | --- |
| **Version Number** | **Release Date** |
| 1.0 | 2016-05-17 |
| 1.1 | 2016-11-18 |
| 2.0 | 2017-08-14 |

Bảng 2.2 : Lịch sử ra đời của ASP.NET Core

* + - 1. Những cải tiến trong ASP.NET Core
* Web UI và Web API được hợp nhất
* Tích hợp các frameworks hiện đại cho khách hàng và nhà phát triển
* A cloud-ready environment-based configuration system
* Built-in dependency injection
* New light-weight and modular HTTP request pipeline
* Khả năng lưu trữ trên IIS hoặc tự chủ trong tiến trình riêng của bạn
* Built on .NET Core, which supports true side-by-side app versioning
* Ships entirely as NuGet packages
* Công cụ vừa mới và đơn giản để phát triển các web hiện đại
* Xây dựng và chạy nền tảng ứng dụng ASP.NET không chỉ trên Windows mà còn trên MAC và Linux nữa
  + 1. So sánh ASP.NET Framework và ASP.NET Core

|  |  |
| --- | --- |
| **ASP.NET Framework** | **ASP.NET Core** |
| Chạy trên Window | Đa nền tảng : Windows , Mac , Linux |
| Mã nguồn đóng | Mã nguồn mở |
| Chạy trên phiên bản đầy đủ của ASP.Net Framework | Có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. |
| Nặng nhưng đầy đủ | Gọn nhẹ tích hợp các Framwork khác nhau lại trong một bản duy nhất là ASP.NET Core |
| Build được thực hiện trên giao diện Visual Studio | Build có thể thực hiện bằng Command line hoặc giao diện |
| Hỗ trợ đầy đủ | Không hỗ trợ Web Form |

Bảng 2.3 : So sánh ASP.NET Framework và ASP.NET Core

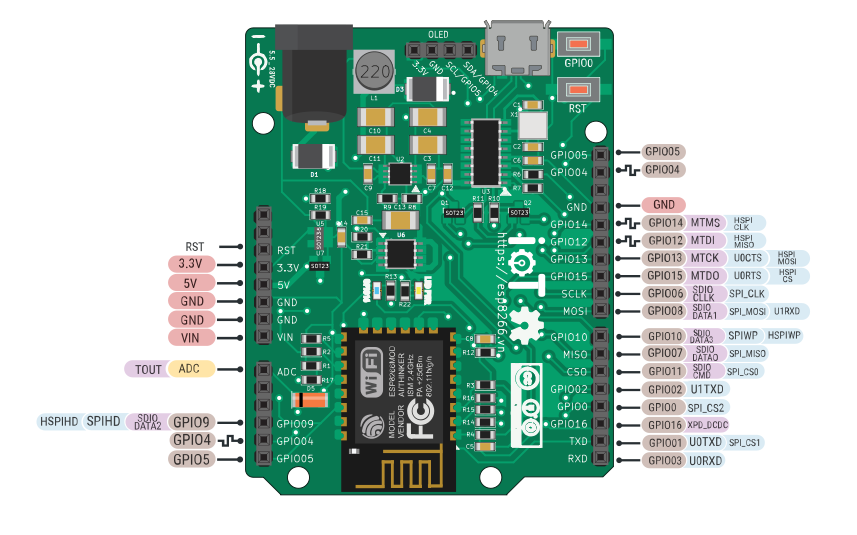
* 1. NodeJS
     1. Tổng quan

1. Node.js là một nền tảng dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.
2. Node.js là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành Window, Linux...
3. Node.js cũng cung cấp cho chúng ta các module Javascript đa dạng, có thể đơn giản hóa sự phát triển của các ứng dụng web sử dụng Node.js với các phần mở rộng.
   * 1. Lịch sử phát triển
4. Node.js là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009 và phiên bản hiện tại là node-v8.11.1
   * 1. Một số đặc điểm của NodeJS
        1. Không đồng bộ và Phát sinh sự kiện (Event Driven):
5. Tất các các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dự liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận đựa phản hồi từ các API gọi trước đó.
   * + 1. Chạy rất nhanh
6. Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.
   * + 1. Không đệm
7. Ứng dụng Node.js không lưu trữ các dữ liệu buffer.
   * + 1. Có giấy phép
8. Node.js được phát hành dựa vào MIT License.
   * + 1. Các tiến trình đơn giản nhưng hiệu năng cao
9. Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.
   1. So sánh ASP.NET Core và NodeJS

|  |  |
| --- | --- |
| **ASP.NET Core** | **NodeJS** |
| Đa nền tảng | Đa nền tảng |
| Chủ yếu phát triển Server Side | Phát triển cả Server Side và Client Side |
| Ngôn ngữ chính là C# | Ngôn ngữ chính là Javascript |
| Hosting hỗ trợ hiếm | Hosting hỗ trợ phong phú |
| Chạy đồng bộ hoặc bất đồng bộ | Chạy bất đồng bộ |

Bảng 2.4 : So sánh ASP.NET Core và NodeJS

1. CHƯƠNG 3: MỘT SỐ MODULE ĐƯỢC SỬ DỤNG TRONG MÔ HÌNH
   1. Arduino ESP8266 Wifi Uno
      1. Tổng quan
2. ESP8266 Arduino core đi kèm với thư viện kết nối WiFi hỗ trợ TCP, UDP và các ứng dụng HTTP, mDNS, SSDP, DNS Servers. Ngoài ra còn có thể thực hiện cập nhật OTA, sử dụng Filesystem dùng bộ nhớ Flash hay thẻ SD, điều khiển servos, ngoại vi SPI, I2C.



Hình 3.1 : Board Arduino ESP8266 Wifi UNO

* + 1. Arduino IDE

Arduino IDE là một trình soạn thảo vào upload code cho các thiết bị Arduino . Bao gồm một giao diện code , quản lý kết nối , thư viện , các loại board khác nhau . Đồng thời hỗ trợ compile và upload code một cách nhanh chóng .

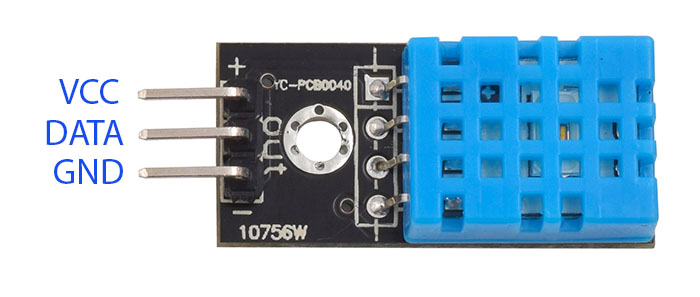
Được hỗ trợ đa nền tảng bao gồm Windows , Mac OS , Linux. Phiên bản hiện tại là Arduino IDE v1.8.5

* + 1. Một số module và sensor hỗ trợ Arduino ESP8266
       1. OLED 128x64 0.96 inch – 12C



Hình 3.2 : OLED

* + - 1. Cảm biến nhiệt độ, độ ẩm DHT11



Hình 3.3 : DHT11

* + - 1. Cảm biến phát hiện lửa



Hình 3.4 : Cảm biến lửa

* + - 1. Cảm biến chuyển động HC-SR501



Hình 3.5 : Cảm biến chuyển động

* + - 1. Cảm biến khoảng cách HC-SR04



Hình 3.6 : Cảm biến khoảng cách

* + - 1. Động cơ Servo



Hình 3.7 : Động cơ servo

* + 1. RFID RC522



Hình 3.8 : RFID

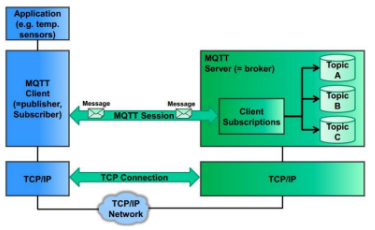
* 1. Raspberry Pi 2(3)
     1. Tổng quan
     2. abc
     3. abc
     4. abc

1. CHƯƠNG 4: MỘT SỐ GIAO THỨC MẠNG
   1. Wifi
      1. Tổng quan
      2. abc
      3. abc
      4. abc
   2. MQTT
      1. Tổng quan
2. MQTT (Message Queuing Telemetry Transport) là giao thức truyền message theo mô hình publish/subcribe. Nó phù hợp cho các ứng dụng M2M (Mobile to mobile), WSN (Wireless Sensor Networks) hay IoT (Internet of Things).
   * 1. Một số đặc trưng

* Dạng truyền message publish/subcribe cung cấp việc tuyền tin phân tán 1-n.
* Dựa trên nền TCP/IP để cung cấp đường truyền.
* Dữ liệu bao bọc đường truyền (overhead) nhỏ (độ dài cố định luôn là 2 byte).
* Cung cấp một cơ chế để thông báo đến các thuê bao khi đường truyền bị đứt bất thường, sử dụng Last Will và Testament feature.
  + 1. Một số khái niệm
       1. Broker , Subscribe , Publish

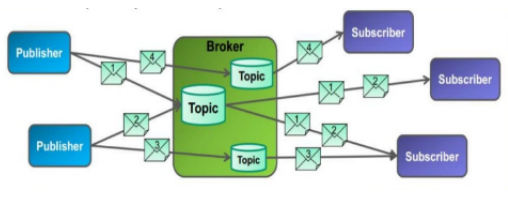
1. Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client - gọi tắt là client) kết nối tới một MQTT server (gọi là Broker). Mỗi client sẽ đăng ký một vài kênh (topic),ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là "Subscribe",giống như chúng ta 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".
   * + 1. QoS:
2. 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.
  + - 1. Retain:

1. 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.
2. 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.
3. 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 việc subscribe.
   * + 1. LWT:
4. 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.
   * 1. Mô hình MQTT:
5. Các thành phần chính của MQTT là clients, servers (=brokers), sessions, subscriptions và topics.
6. 

Hình 4.1: Mô hình cơ bản của giao thức MQTT

* MQTT client (publisher, subscriber): Client thực hiện subscribe đến topics để publish và receive các gói tin.
* MQTT server (broker): Servers thực hiện run các topic, đồng thời nhận subscriptions từ clients yêu cầu các topics, nhận các messages từ clients và forward chúng.
* Topic: Về mặt kỹ thuật, topics là các hàng đợi chứa message. Về logic, topics cho phép clients trao đổi thông tin và dữ liệu.
* Session: Một session được định nghĩa là kết nối từ client đến server. Tất cả các giao tiếp giữa client và server đều là 1 phần của session.
* Subscription: Không giống như sessions, subscription về mặt logic là kết nối từ client đến topic. Khi thực hiện subscribed đến topic, client có thể trao đổi messages với topic.
* Subscriptions có thể ở trạng thái ‘transient’ hoặc ‘durable’, phụ thuộc vào cờ clean session trong gói Connect.
* Message: Messages là các đơn vị dữ liệu được trao đổi giữa các topic clients.



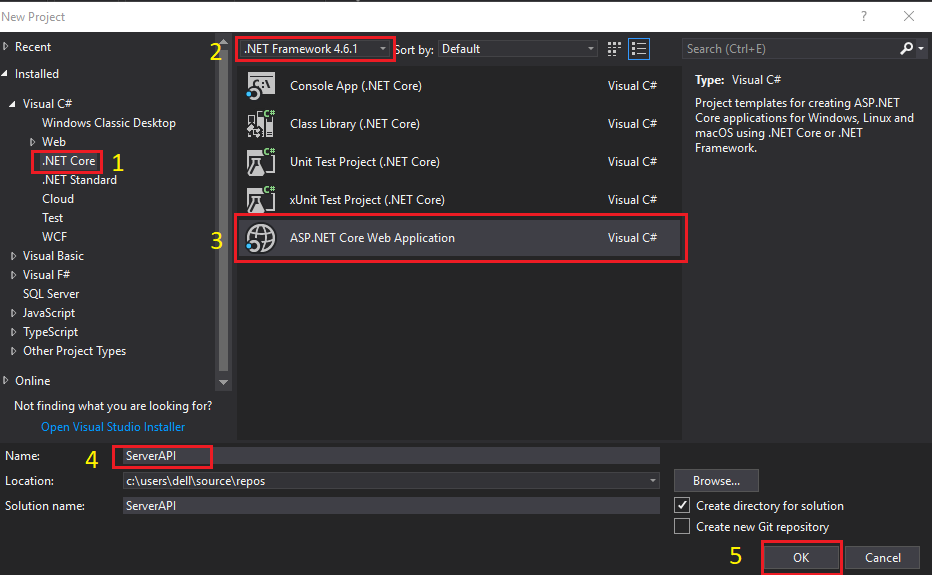
Hình 4.2: Cơ chế hoạt động của giao thức MQTT

* MQTT hoạt động theo cơ chế client/server, nơi mà mỗi cảm biến là một client và kết nối đến một máy chủ, có thể hiểu như là một điểm trung gian (broker), thông qua giao thức
* TCP ( Transmission Control Protocol). Broker chịu trách nhiệm điều phối tất cả các thông điệp từ phía gửi đến đúng phía nhận.
* MQTT là giao thức định hướng bản tin. Mỗi bản tin là một đoạn rời rạc của tín hiệu và broker không thể nhìn thấy. Mỗi bản tin được publish một địa chỉ, có thể hiểu như một kênh.
* Client đăng kí vào một vài kênh để nhận/gửi dữ liệu, gọi là subscribe. Client có thể subscribe vào nhiều kênh. Mỗi client sẽ nhận được dữ liệu khi bất kì trạm nào khác gửi dữ liệu vào kênh đã đăng ký. Khi một client gửi một bản tin đến một kênh nào đó gọi là publish.
  + 1. Ưu điểm của MQTT:
* MQTT là giao thức mở và chuẩn hóa bởi OASIS Technical Committee . Điều này làm cho giao thức này dễ dàng chấp nhận cho nhiều loại thiết bị, nền tảng và hệ điều hành IoT. Nhiều ứng dụng của MQTT có thể được phát triển chỉ bằng cách thực hiện CONNECT, PUBLISH, SUBSCRIBE, và DISCONNECT các gói tin kiểm soát.
* Lý tưởng cho các mạng bị ràng buộc ( băng thông thấp, độ trễ cao, giới hạn dữ liệu và kết nối mong manh).
* Tốc độ giao dịch nhanh chóng.
* Độ trễ thấp
* Dễ dàng mở rộng quy mô về số lượng các thiết bị kết nối.
* Bảo mật.
  + 1. Nhược điểm của MQTT:

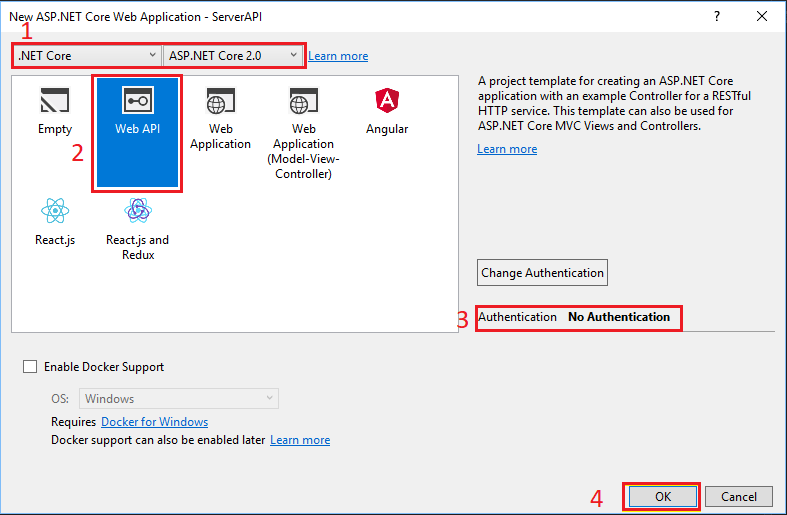
1. Mỗi client MQTT phải hỗ trợ TCP và thường sẽ giữ một kết nối mở đến broker ở mọi thời điểm. Đối với một số môi trường mà xác suất mất bản tin cao hay không có sẵn máy tính thì đây là cả một vấn đề.
2. Tên kênh MQTT thường là các chuỗi dài làm chúng không thỏa đáng với tiêu chuẩn 802.15.4.
   * 1. Bảo mật:
3. MQTT broker có thể yêu cầu tên người dùng và mật khẩu xác thực từ client để kết nối. Để đảm bảo tính bảo mật, kết nối TCP có thể được mã hóa với SSL/TLS (Transport Layer Security (TLS) protocol, Secure Sockets Layer (SSL) protocol).
   1. COAP
      1. Tổng quan
4. CoAP (Constrained Application Protocol) là giao thức ràng buộc ứng dụng Internet cho các thiết bị bị ràng buộc. Nó cho phép những thiết bị bị ràng buộc có thể giao tiếp với Internet bằng cách sử dụng các giao thức tương tự.
5. CoAP là một giao thức ứng dụng được sử dụng cho các thiết bị internet bị hạn chế tài nguyên,chẳng hạn như các nút mạng cảm biến không dây.
   * 1. Kiến trúc:
6. Giống như HTTP. CoAP là một giao thức truyền tải tài liệu. Không giống như HTTP, CoAP được thiết kế cho nhu cầu các thiết bị ràng buộc.
7. Các gói CoAP nhỏ hơn nhiều so với dòng HTTP TCP. Các trường bit và ánh từ chuỗi các số nguyên được sử dụng rộng rãi để tiết kiệm không gian. Các gói rất đơn giản có thể được tạo ra và phân tích tại chỗ mà không tốn thêm RAM trong các thiết bị bị hạn chế.
8. CoAP chạy trên UDP, không chyaj trên TCP. Client và máy chủ giao tiếp thông qua các gói kết nối. Việc loại bỏ TCP cho phép kết nối mạng IP đầy đủ trên các vi điều khiển nhỏ. CoAP cho phép phương thức gửi UDP broadcast và multicast.
9. CoAP theo mô hình client/server. Client gửi các yêu cầu đến máy chủ, máy chủ gưi lại phản hồi. Client có thể GET, PUT, POST và DELETE các tài nguyên.
10. CoAP được thiết kế để tương thích với HTTP và các web REDful rộng lớn thông qua các proxy đơn giản.
11. Bởi vì CoAP dựa trên cơ sở gói dữ liệu datagram, nó có thể được dùng ở đầu bản tin và các giao thức truyền dữ liệu gói khác.
    * 1. Đặc điểm:
         1. Tầng ứng dụng QoS:
12. Các yêu cầu và tin nhắn phản hồi có thể được đánh dấu là "confirmable" hoặc "nonconfirmable".Thông điệp “confirmable” phải được xác nhận bởi bên nhận bằng một gói ack.
13. Thông điệp “nonconfirmable” có nghĩa là gửi và quên.
    * + 1. Tính linh động:
14. Giống như HTTP, CoAP hỗ trợ đàm phán nội dung. Client sử dụng lựa chọn Accept để thể hiện một đối tượng được quan tâm trong nguồn tài nguyên, máy chủ hồi đáp với lựa chọn
15. Content-Type để nói với client nó đang có những gì. Như với HTTP, điều này cho phép client và máy chủ có thể cải tiến một cách độc lập, thêm đối tượng mới mà không ảnh hưởng đến nhau.
    * + 1. Khả năng quan sát:
16. CoAP yêu cầu các mô hình với khả năng quan sát tài nguyên. Khi cờ quan sát được thiết lập trên một yêu cầu GET, máy chủ có thể tiếp tục hồi đáp sau khi các tài liệu đã được truyền đi.
17. Điều này cho phép các máy chủ gửi những sự thay đổi trạng thái đến cho client khi điều đó xảy ra. Cả hai bên đều có thể hủy bỏ các yêu cầu quan sát.
    * + 1. Các vấn đề về NAT:
18. Trong CoAP, một nút cảm biến thường là một máy chủ, không phải một client mặc dù nó có thể là cả 2. Các cảm biến hoặc đầu dò cũng cấp tài nguyên mà có thể được truy cập bởi client để đọc hoặc thay đổi trạng thái của cảm biến.
19. Khi các cảm biến là máy chủ, chúng phải có khả năng nhận các gói tin gửi đến. Để hoạt động đúng đăng sau NAT, thiết bị đầu tiên phải gửi yêu cầu đến máy chủ, cho phép các router liên kết chúng lại. Mặc dù CoAP không yêu cầu IPv6, nó là phương thức đơn giản nhất được sử dụng trong môi trường IP nơi có các thiết bị định tuyến trực tiếp.
    * + 1. Mô hình hoạt động:
20. CoAP theo mô hình client/server. Client gửi các yêu cầu đến máy chủ, máy chủ gưi lại phản hồi. Client có thể GET, PUT, POST và DELETE các tài nguyên.
21. CoAP có hỗ trợ yêu cầu một nhóm multicast IP. Các điểm cuối trong CoAP cung cấp dịch vụ để các điểm cuối khác có thể tìm thấy.
    * + 1. Ưu điểm:

* Dễ dàng dịch sang HTTP để tích hợp với web.
* Chi phí thấp và đơn giản.
* Linh động.
* Bảo mật.
  + - 1. Nhược điểm:
* Chưa ổn định và phát triển.
* Thách thức về khả năng tương tác.
  + - 1. Bảo mật:

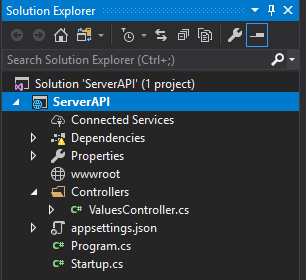
1. Bởi vì CoAP được xây dựng trên UDP chứ không phải TCP, SSL/TLS không có sẵn để cung cấp khả năng bảo mật. DTLS (Datagram Transport Layer Security) cung cấp sự đảm bảo giống như TLS nhưng để truyền dữ liệu qua UDP. Thông thường, các thiết bị DTLS sẽ hỗ trợ RSA và AES hoặc ECC và AES.
   1. So sánh các giao thức mạng
2. CHƯƠNG 5: KỊCH BẢN THỰC HIỆN
3. CHƯƠNG 6: XÂY DỰNG WEB API BẰNG ASP.NET CORE TRÊN VISUAL STUDIO 2017
   1. Các bước thực hiện
      1. Bước 1 : Tạo một project ASP.NET Core API



Hình 6.1 : Tạo một project (1)



Hình 6.2 : Tạo một project (2)

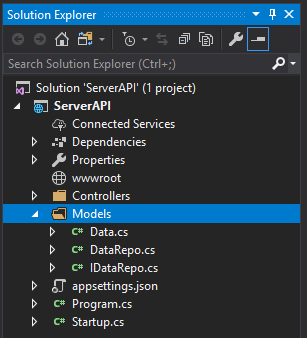


Hình 6.3 : Kết quả sau khi tạo một project

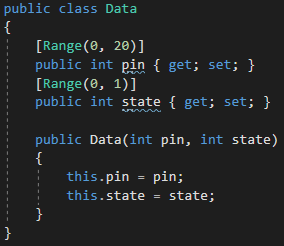
* + 1. Bước 2 : Tạo model để quản lí dữ liệu

1. Bao gồm :

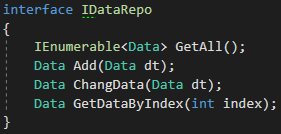
* Class Data.cs : dùng để định nghĩa các thuộc tính của dữ liệu
* Interface IDataRepo.cs : dùng để định nghĩa các phương thức của dữ liệu
* Class DataRepo.cs : dùng để implement các phương thức từ IDataRepo.cs



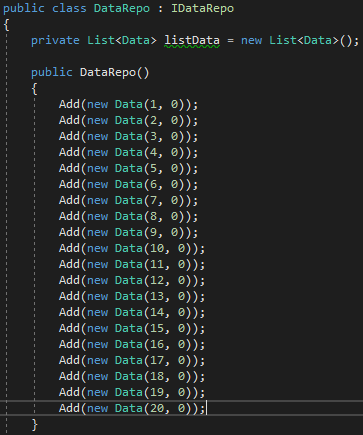
Hình 6.4 : Tạo Model



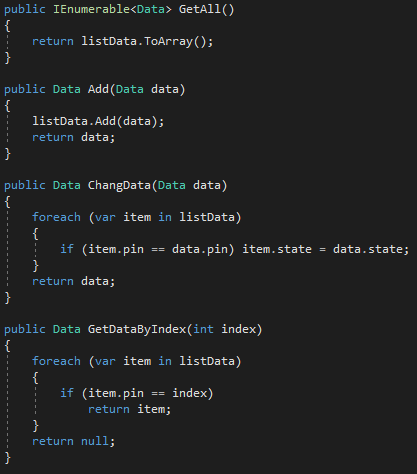
Hình 6.5 : Nội dung file Data.cs



Hình 6.6 : Nội dung file IDataRepo.cs



Hình 6.7 : Nội dung file DataRepo.cs (1)



Hình 6.8 : Nội dung file DataRepo.cs (2)

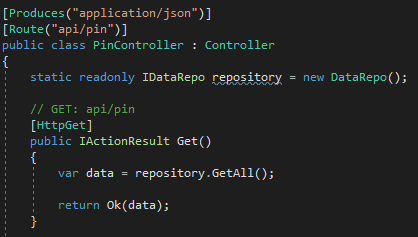
* + 1. Bước 3 : Xóa ValueController.cs và tạo một Controller mới



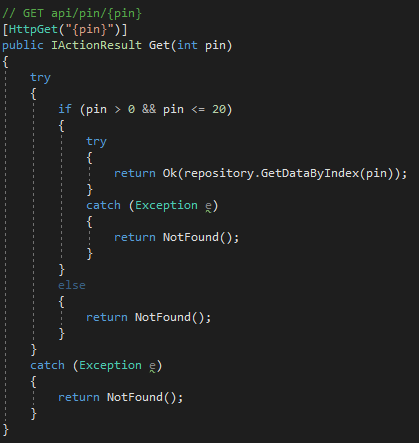
Hình 6.9 : Tạo Controller

1. Nội dung file PinController.cs bao gồm các Action :

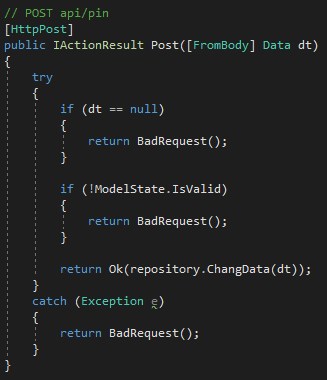
* Action Get() : Mặc định khi chạy server sẽ chạy hàm này đầu tiên , kết quả trả về là một mảng Data .
* Action Get(int pin) : Hàm này sẽ trả về đối tượng Data có giá trị pin bằng giá trị pin được yêu cầu .
* Action Post([FromBody] Data dt) : Hàm này sẽ nhận một đối tượng Data từ Request và chỉnh sửa giá trị đối tượng có pin tương ứng trong Model



Hình 6.10 : Nội dung file PinController.cs (1)



Hình 6.11 : Nội dung file PinController.cs (2)



Hình 6.12 : Nội dung file PinController.cs (3)

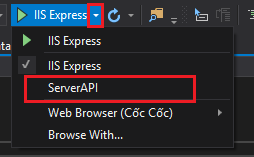
* + 1. Bước 4 : Thay đổi nội dung file Properties => lauchSettings.json

1. Lưu ý : port sẽ Random nên không cần sửa giá trị port



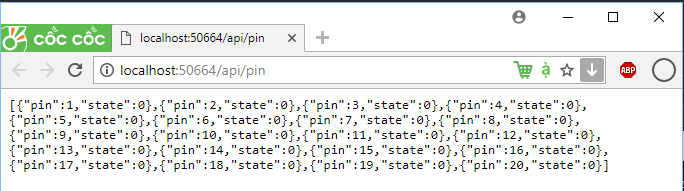
Hình 6.13 : Nội dung file lauchSetting.json

* + 1. Bước 5 : Chạy Server



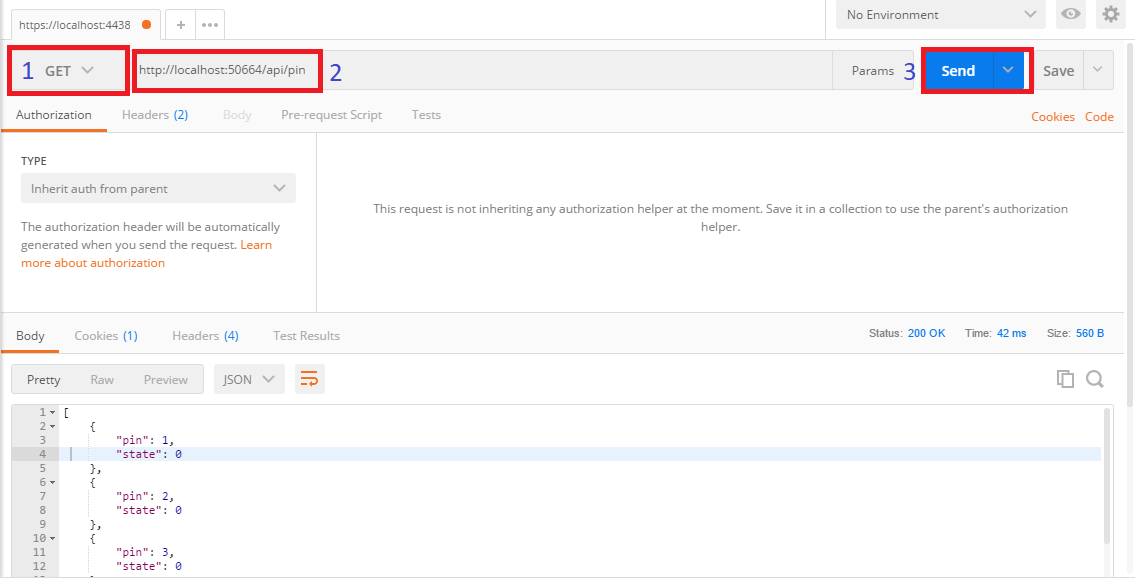
Hình 6.14 : Run project bằng ServerAPI

* + 1. Bước 6 : Kiểm tra kết quả trên trình duyệt

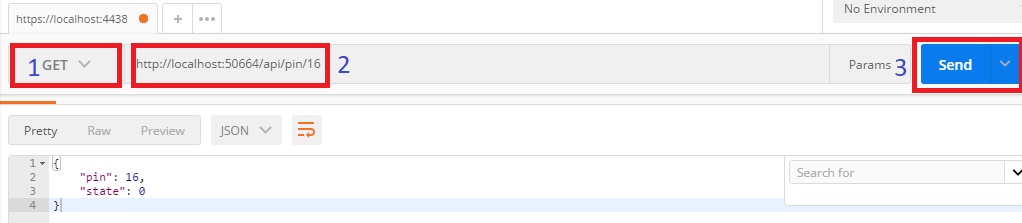


Hình 6.15 : Kết quả sai khi chạy Server

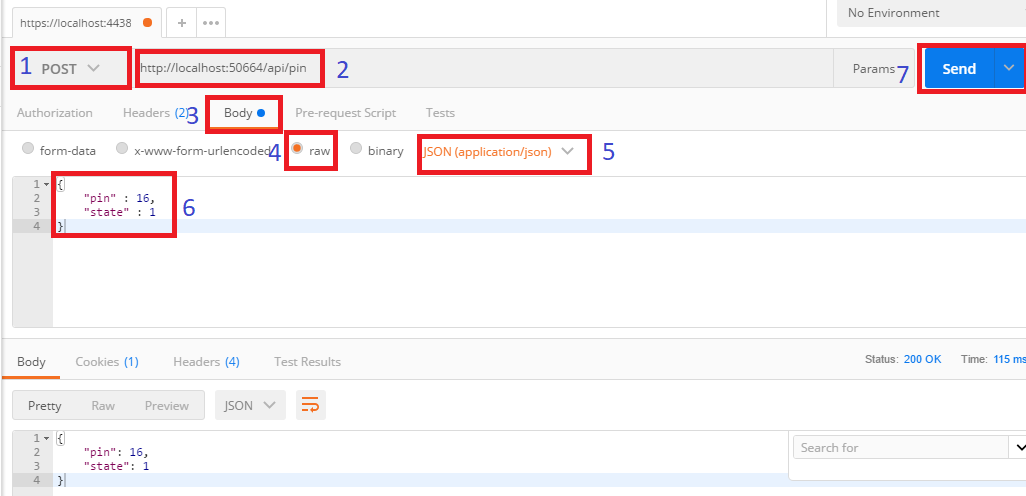
* + 1. Bước 7 : Test các hàm trong file PinController.cs bằng Postman



Hình 6.16 : Action Get()



Hình 6.17 : Action Get(int pin)



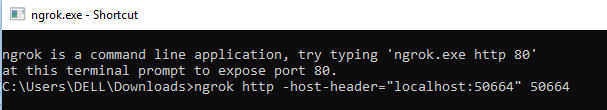
Hình 6.18 : Action Post(Data dt)

* 1. Ngrok
     1. Tại sao lại cần sử dụng Ngrok trong đồ án này ?

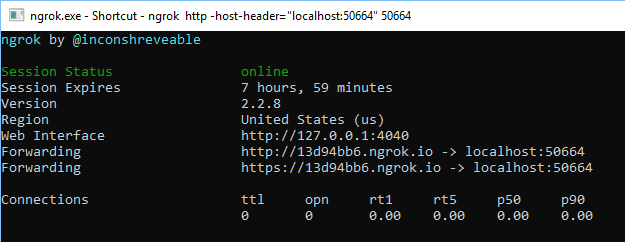
1. Mặc định thì Visual Studio sẽ chặn tất cả các kết nối , request từ bên ngoài ngoại trừ ứng dụng trên chính máy chạy Server . Vì thế khi bạn sử dụng điện thoại hoặc Arduino ESP8266 truy cập đến localhost của Server , mặc dù chung mạng LAN với máy chạy Server nhưng vẫn không kết nối, lấy dữ liệu từ Server được.
2. Để có thể kết nối dễ dàng thì ta sẽ sử dụng đến phần mềm tạo tunnel - Ngrok
   * 1. Tổng quan
3. Ngrok là công cụ tạo đường hầm (tunnel) giữa localhost của bạn và internet. Giúp người khác mạng có thể truy cập được localhost của bạn thông qua custom domain của ngrok
4. Ví dụ: mydomain.ngrok.io => localhost:80
   * 1. Đặc điểm của Ngrok

* Giúp bạn chạy demo dự án cho khách hàng xem từ chính máy của bạn mà không cần deploy lên server
* Bạn có thể test responsive trên mobile một cách dễ dàng thông qua URL mà ngrok cung cấp
* Xây dựng webhook tới localhost của bạn một cách dễ dàng
* Hỗ trợ http, https, tcp
* Hỗ trợ IP whitelist
  + 1. Cài đặt Ngrok

1. Link download : <https://ngrok.com/download>
2. Ngrok không cần cài đặt , chỉ cần download về và chạy file .exe
   * 1. Chạy Ngrok
3. Chạy file .exe và patse lệnh sau vào command line
4. “ngrok http -host-header="localhost:50664" 50664”
5. Lưu ý : Thay đổi port



Hình 6.19 : Chạy Ngrok



Hình 6.20 : Kết quả sau khi chạy Ngrok

1. Như vậy ta thấy địa chỉ localhost:50664 đã được thay bằng địa chỉ mà Ngrok random ra http://13d94bb6.ngrok.io
   * 1. Kiểm tra kết nối đến Ngrok

Sử dụng điện thoại truy cập đến đường dẫn : <http://13d94bb6.ngrok.io/api/pin>



Hình 6.21 : Kết quả kiểm tra kết nối Ngrok trên điện thoại

* + 1. Kết luận

Vậy ta đã tạo một Server API cơ bản có các Action có thể lấy dữ liệu , thay đổi dữ liệu nhằm mục đích điều khiển tín hiệu đơn giản ( ON /OFF) .

Kết nối Server thông qua các thiết bị bên ngoài : điện thoại .

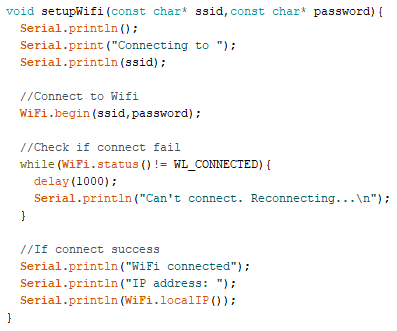
Phần tiếp theo ta sẽ kết nối đến Server thông qua Arduino ESP8266

1. CHƯƠNG 7: CODE ARDUINO
   1. Sơ lượt
2. Dưới đây là một số đoạn code ngắn dành cho từng phần trong code ESP8266 được cắt tỉa ra. Các bạn có thể chọn các phần thích hợp để ráp code lại và chạy hoặc ráp tất cả các code bên dưới thành một tùy theo mục đích của bạn.
3. Các bạn có thể tham khảo code được sử dụng từ những đoạn code ngắn này tại link này :

**https://github.com/haibui2207/ServerAPI**

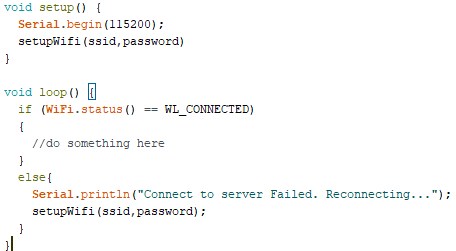
* 1. Kết nối Wifi
     1. Import thư viện

1. #include <ESP8266WiFi.h>
   * 1. Khai báo Wifi
2. const char\* ssid = “SSID”;
3. const char\* password = “Password”;
   * 1. Hàm setupWifi()



Hình 7.1 : Hàm kết nối Wifi

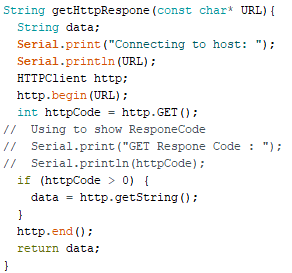
* + 1. Vị trí đặt hàm setupWifi()



Hình 7.2 : Vị trí đặt hàm setupWifi

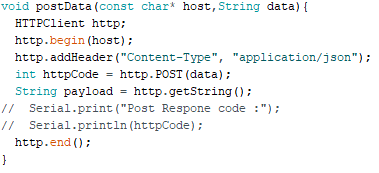
* 1. Kết nối Local Server
     1. Import thư viện

1. #include <ESP8266HTTPClient.h>
2. #include <ArduinoJson.h>
   * 1. Khai báo kết nối
3. const char\* host = “Đường\_dẫn\_Ngrok\_server”;
   1. Get dữ liệu từ Server



Hình 7.3 : Hàm getHttpResponse()

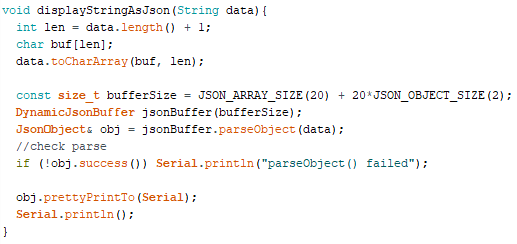
* 1. Post dữ liệu lên Server



Hình 7.4 : Hàm Post()

* 1. Xử lý dữ liệu nhận được thành Json
     1. Hàm displayStringAsJson()

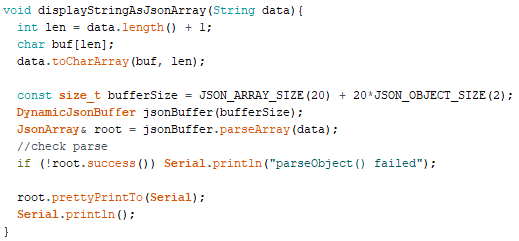
1. Dùng để hiển thị một đối tượng được trả về từ Server dưới dạng Json
2. “bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/



Hình 7.5 : Hàm displayStringAsJson()

* + 1. Hàm displayStringAsJsonArray()

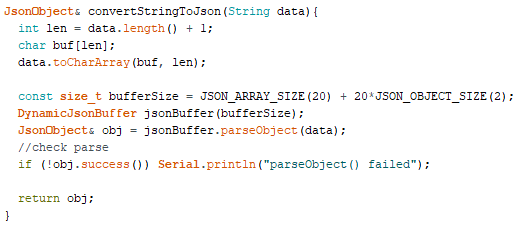
1. Dùng để hiển thị một danh sách các đối tượng được trả về từ Server dưới dạng Json
2. “bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/



Hình 7.6 : Hàm displayStringAsJsonArray()

* + 1. Hàm convertStringToJson()

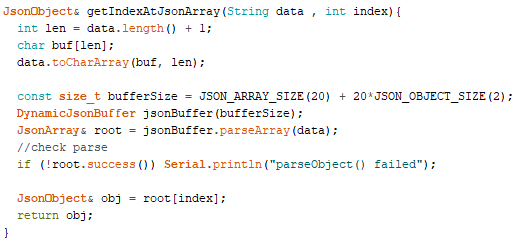
1. Convert chuỗi lấy được từ Server thành Json để dễ dàng dữ lý



Hình 7.7 : Hàm convertStringToJson()

* + 1. Hàm getIndexAtJsonArray()

1. Dùng để lấy ra một đối tượng từ mảng Json lấy được từ Server



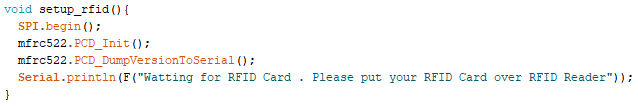
Hình 7.8 : Hàm getIndexAtJsonArray()

* 1. Xử lý kết nối RFID
     1. Import thư viện

1. #include <SPI.h>
2. #include "MFRC522.h"
   * 1. Hướng dẫn nối chân dành cho ESP8266 Wifi UNO

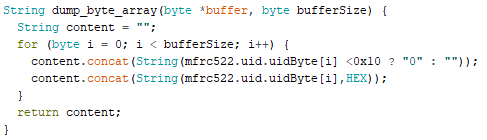
* SDA(SS) 15
* SCK 14
* MOSI 13
* MISO 12
* RST 2
  + 1. Khai báo kết nối

1. #define SS\_PIN 15
2. #define RST\_PIN 2
3. MFRC522 mfrc522(SS\_PIN, RST\_PIN);
   * 1. Hàm setup\_rfid()



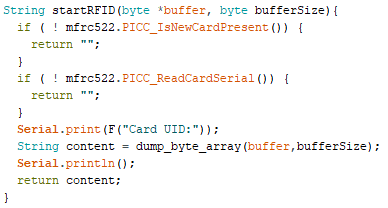
Hình 7.9 : Hàm setup\_rfid()

* + 1. Hàm dump\_byte\_array()



Hình 7.10 : Hàm dump\_byte\_array()

* + 1. Hàm startRFID()



Hình 7.11 : Hàm startRFID()

* 1. FIX các lỗi đã gặp
     1. Lỗi ESP8266 không nhận Wifi
        1. Nguyên nhân

1. Do phiên bản sử dụng không thích hợp với board đang sử dụng
   * + 1. Giải pháp
2. Vào Tool => Board => Board Manager => Search “esp8266” => Lựa chọn phiên bản phù hợp => Install
   * 1. Lỗi ESP8266 không kết nối được đến localhost
        1. Nguyên nhân
3. Mặc định IIS chặn tất cả các kết nối truy cập đến Server ngoại trừ máy chạy Server
   * + 1. Giải pháp
4. Thay vì chạy Server bằng IIS thì chạy chọn nút xuống và chọn chạy bằng ServerAPI ( ServerAPI được thay bằng tên solution tương ứng).
5. Tắt tường lửa
6. Sử dụng Ngrok như đã hướng dẫn ở trên
   * 1. Lỗi “Firmware Unknown” khi ESP8266 kết nối RFID
        1. Nguyên nhân
7. Mỗi Board có cách kết nối dây khác nhau
   * + 1. Giải pháp
8. Tìm cách nối dây tương ứng với board đang sử dụng . Link tham khảo https://github.com/miguelbalboa/rfid
9. CHƯƠNG 8: CODE RASPBERRY PI
   1. ABC
      1. abc
      2. abc
      3. abc
   2. ABC
      1. abc
      2. abc
      3. abc
   3. FIX các lỗi đã gặp
10. Chương 9: xxxx
11. CHƯƠNG xxx: OPENSTACK MITAKA (Xây dựng xong local mới triển khai trên openstack)
    1. Tổng quan
    2. Một số OS có thể triển khai Web Server
       1. Window Server
       2. Azure
       3. Linux
          1. Ubuntu
          2. CentOS
    3. Triển khai IIS trên Window Server
       1. Cài đặt và sử dụng OpenVPN
       2. Tạo một Instance chạy Window Server trên OpenStack
       3. Truy cập đến Instance chạy Window Server bằng Remote Destop Connection
       4. Cấu hình cho Instance
    4. Public file lên Instance bằng Visual Studio

# Bibliography