**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**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

[1 CHƯƠNG 1: GIỚI THIỆU 11](#_Toc511852818)

[2 CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/API 12](#_Toc511852819)

[2.1 ASP.NET 12](#_Toc511852820)

[2.1.1 Tổng quan 12](#_Toc511852821)

[2.1.2 ASP.NET Framework 12](#_Toc511852822)

[2.1.2.1 Tổng quan 12](#_Toc511852823)

[2.1.2.2 Lịch sử phát triển 13](#_Toc511852824)

[2.1.3 ASP.NET Core 14](#_Toc511852825)

[2.1.3.1 Tổng quan 14](#_Toc511852826)

[2.1.3.2 Lịch sử ra đời 14](#_Toc511852827)

[2.1.3.3 Những cải tiến trong ASP.NET Core 14](#_Toc511852828)

[2.1.4 So sánh ASP.NET Framework và ASP.NET Core 15](#_Toc511852829)

[2.2 NodeJS 15](#_Toc511852830)

[2.2.1 Tổng quan 15](#_Toc511852831)

[2.2.2 Lịch sử phát triển 16](#_Toc511852832)

[2.2.3 Một số đặc điểm của NodeJS 16](#_Toc511852833)

[2.2.3.1 Không đồng bộ và Phát sinh sự kiện (Event Driven): 16](#_Toc511852834)

[2.2.3.2 Chạy rất nhanh 16](#_Toc511852835)

[2.2.3.3 Không đệm 16](#_Toc511852836)

[2.2.3.4 Có giấy phép 16](#_Toc511852837)

[2.2.3.5 Các tiến trình đơn giản nhưng hiệu năng cao 16](#_Toc511852838)

[2.3 So sánh ASP.NET Core và NodeJS 17](#_Toc511852839)

[3 CHƯƠNG 3: MỘT SỐ MODULE ĐƯỢC SỬ DỤNG TRONG MÔ HÌNH 18](#_Toc511852840)

[3.1 Arduino ESP8266 Wifi Uno 18](#_Toc511852841)

[3.1.1 Tổng quan 18](#_Toc511852842)

[3.1.2 Arduino IDE 18](#_Toc511852843)

[3.1.3 Một số module và sensor hỗ trợ Arduino ESP8266 19](#_Toc511852844)

[3.1.3.1 OLED 128x64 0.96 inch – 12C 19](#_Toc511852845)

[3.1.3.2 Cảm biến nhiệt độ, độ ẩm DHT11 19](#_Toc511852846)

[3.1.3.3 Cảm biến phát hiện lửa 19](#_Toc511852847)

[3.1.3.4 Cảm biến chuyển động HC-SR501 20](#_Toc511852848)

[3.1.3.5 Cảm biến khoảng cách HC-SR04 20](#_Toc511852849)

[3.1.3.6 Động cơ Servo 21](#_Toc511852850)

[3.1.3.7 RFID RC522 21](#_Toc511852851)

[3.2 Raspberry Pi 2(3) 22](#_Toc511852852)

[3.2.1 Tổng quan 22](#_Toc511852853)

[3.2.2 Phần cứng 23](#_Toc511852854)

[3.2.3 Hệ điều hành: 23](#_Toc511852855)

[3.2.4 Module sử dụng: Raspberry Pi Camera Module: 23](#_Toc511852856)

[4 CHƯƠNG 4: MỘT SỐ GIAO THỨC MẠNG 25](#_Toc511852857)

[4.1 Wifi 25](#_Toc511852858)

[4.1.1 Tổng quan 25](#_Toc511852859)

[4.1.2 Các chuẩn Wifi: 25](#_Toc511852860)

[4.2 MQTT 25](#_Toc511852861)

[4.2.1 Tổng quan 25](#_Toc511852862)

[4.2.2 Một số đặc trưng 25](#_Toc511852863)

[4.2.3 Một số khái niệm 26](#_Toc511852864)

[4.2.3.1 Broker , Subscribe , Publish 26](#_Toc511852865)

[4.2.3.2 QoS: 26](#_Toc511852866)

[4.2.3.3 Retain: 26](#_Toc511852867)

[4.2.3.4 LWT: 27](#_Toc511852868)

[4.2.4 Mô hình MQTT: 27](#_Toc511852869)

[4.2.5 Ưu điểm của MQTT: 28](#_Toc511852870)

[4.2.6 Nhược điểm của MQTT: 29](#_Toc511852871)

[4.2.7 Bảo mật: 29](#_Toc511852872)

[4.3 COAP 29](#_Toc511852873)

[4.3.1 Tổng quan 29](#_Toc511852874)

[4.3.2 Kiến trúc: 29](#_Toc511852875)

[4.3.3 Đặc điểm: 30](#_Toc511852876)

[4.3.3.1 Tầng ứng dụng QoS: 30](#_Toc511852877)

[4.3.3.2 Tính linh động: 30](#_Toc511852878)

[4.3.3.3 Khả năng quan sát: 30](#_Toc511852879)

[4.3.3.4 Các vấn đề về NAT: 31](#_Toc511852880)

[4.3.3.5 Mô hình hoạt động: 31](#_Toc511852881)

[4.3.3.6 Ưu điểm: 31](#_Toc511852882)

[4.3.3.7 Nhược điểm: 31](#_Toc511852883)

[4.3.3.8 Bảo mật: 31](#_Toc511852884)

[4.4 So sánh các giao thức mạng 32](#_Toc511852885)

[5 CHƯƠNG 5: KỊCH BẢN THỰC HIỆN 33](#_Toc511852886)

[6 CHƯƠNG 6: XÂY DỰNG WEB API BẰNG ASP.NET CORE TRÊN VISUAL STUDIO 2017 34](#_Toc511852887)

[6.1 Các bước thực hiện 34](#_Toc511852888)

[6.1.1 Bước 1 : Tạo một project ASP.NET Core API 34](#_Toc511852889)

[6.1.2 Bước 2 : Tạo model để quản lí dữ liệu 35](#_Toc511852890)

[6.1.2.1 DataModel 35](#_Toc511852891)

[6.1.2.2 UserModel 38](#_Toc511852892)

[6.1.3 Bước 3 : Xóa ValueController.cs và tạo một Controller mới 40](#_Toc511852893)

[6.1.3.1 PinController 40](#_Toc511852894)

[6.1.3.2 UserController 42](#_Toc511852895)

[6.1.4 Bước 4 : Thay đổi nội dung file Properties => lauchSettings.json 45](#_Toc511852896)

[6.1.5 Bước 5 : Chạy Server 45](#_Toc511852897)

[6.1.6 Bước 6 : Kiểm tra kết quả trên trình duyệt 45](#_Toc511852898)

[6.1.7 Bước 7 : Test các hàm trong file PinController.cs bằng Postman 46](#_Toc511852899)

[6.2 Ngrok 47](#_Toc511852900)

[6.2.1 Tại sao lại cần sử dụng Ngrok trong đồ án này ? 47](#_Toc511852901)

[6.2.2 Tổng quan 47](#_Toc511852902)

[6.2.3 Đặc điểm của Ngrok 47](#_Toc511852903)

[6.2.4 Cài đặt Ngrok 48](#_Toc511852904)

[6.2.5 Chạy Ngrok 48](#_Toc511852905)

[6.2.6 Limited connection 49](#_Toc511852906)

[6.2.7 Kiểm tra kết nối đến Ngrok 49](#_Toc511852907)

[6.2.8 Kết luận 49](#_Toc511852908)

[7 CHƯƠNG 7: CODE ARDUINO 50](#_Toc511852909)

[7.1 Sơ lượt 50](#_Toc511852910)

[7.2 Kết nối Wifi 50](#_Toc511852911)

[7.2.1 Import thư viện 50](#_Toc511852912)

[7.2.2 Khai báo Wifi 50](#_Toc511852913)

[7.2.3 Hàm setupWifi() 50](#_Toc511852914)

[7.2.4 Vị trí đặt hàm setupWifi() 51](#_Toc511852915)

[7.3 Kết nối Local Server 51](#_Toc511852916)

[7.3.1 Import thư viện 52](#_Toc511852917)

[7.3.2 Khai báo kết nối 52](#_Toc511852918)

[7.4 Get dữ liệu từ Server 52](#_Toc511852919)

[7.5 Post dữ liệu lên Server 52](#_Toc511852920)

[7.6 Xử lý dữ liệu nhận được thành Json 53](#_Toc511852921)

[7.6.1 Hàm displayStringAsJson() 53](#_Toc511852922)

[7.6.2 Hàm displayStringAsJsonArray() 53](#_Toc511852923)

[7.6.3 Hàm convertStringToJson() 54](#_Toc511852924)

[7.6.4 Hàm getIndexAtJsonArray() 54](#_Toc511852925)

[7.7 Xử lý kết nối RFID 55](#_Toc511852926)

[7.7.1 Import thư viện 55](#_Toc511852927)

[7.7.2 Hướng dẫn nối chân dành cho ESP8266 Wifi UNO 55](#_Toc511852928)

[7.7.3 Khai báo kết nối 55](#_Toc511852929)

[7.7.4 Hàm setup\_rfid() 55](#_Toc511852930)

[7.7.5 Hàm dump\_byte\_array() 56](#_Toc511852931)

[7.7.6 Hàm startRFID() 56](#_Toc511852932)

[7.8 Xử lý kết nối OLED 56](#_Toc511852933)

[7.8.1 Import thư viện 56](#_Toc511852934)

[7.8.2 Khai báo kết nối 57](#_Toc511852935)

[7.8.3 Hàm setupOled() 57](#_Toc511852936)

[7.8.4 Code thay đổi screen 57](#_Toc511852937)

[7.9 FIX các lỗi đã gặp 57](#_Toc511852938)

[7.9.1 Lỗi ESP8266 không nhận Wifi 57](#_Toc511852939)

[7.9.1.1 Nguyên nhân 57](#_Toc511852940)

[7.9.1.2 Giải pháp 57](#_Toc511852941)

[7.9.2 Lỗi ESP8266 không kết nối được đến localhost 58](#_Toc511852942)

[7.9.2.1 Nguyên nhân 58](#_Toc511852943)

[7.9.2.2 Giải pháp 58](#_Toc511852944)

[7.9.3 Lỗi “Firmware Unknown” khi ESP8266 kết nối RFID 58](#_Toc511852945)

[7.9.3.1 Nguyên nhân 58](#_Toc511852946)

[7.9.3.2 Giải pháp 58](#_Toc511852947)

[8 CHƯƠNG 8: CODE RASPBERRY PI 59](#_Toc511852948)

[8.1 Cài đặt hệ điều hành Raspbian trên Linux: 59](#_Toc511852949)

[8.1.1 Cài đặt 59](#_Toc511852950)

[8.1.2 Kết nối wifi và SSH: 59](#_Toc511852951)

[8.1.2.1 Kết nối wifi: 59](#_Toc511852952)

[8.1.3 Kết nối SSH: 59](#_Toc511852953)

[8.2 OpenCV 60](#_Toc511852954)

[8.2.1 Giới thiệu 60](#_Toc511852955)

[8.2.2 Mục đích 60](#_Toc511852956)

[8.2.3 Cài đặt OpenCV 60](#_Toc511852957)

[8.2.4 Ứng dụng 60](#_Toc511852958)

[8.3 FIX các lỗi đã gặp 60](#_Toc511852959)

[9 Chương 9: xxxx 61](#_Toc511852960)

[10 CHƯƠNG xxx: OPENSTACK MITAKA (Xây dựng xong local mới triển khai trên openstack) 62](#_Toc511852961)

[10.1 Tổng quan 62](#_Toc511852962)

[10.2 Một số OS có thể triển khai Web Server 62](#_Toc511852963)

[10.2.1 Window Server 62](#_Toc511852964)

[10.2.2 Azure 62](#_Toc511852965)

[10.2.3 Linux 62](#_Toc511852966)

[10.2.3.1 Ubuntu 62](#_Toc511852967)

[10.2.3.2 CentOS 62](#_Toc511852968)

[10.3 Triển khai IIS trên Window Server 62](#_Toc511852969)

[10.3.1 Cài đặt và sử dụng OpenVPN 62](#_Toc511852970)

[10.3.2 Tạo một Instance chạy Window Server trên OpenStack 62](#_Toc511852971)

[10.3.3 Truy cập đến Instance chạy Window Server bằng Remote Destop Connection 62](#_Toc511852972)

[10.3.4 Cấu hình cho Instance 62](#_Toc511852973)

[10.4 Public file lên Instance bằng Visual Studio 62](#_Toc511852974)

[Bibliography 63](#_Toc511852975)

**DANH SÁCH HÌNH VẼ**

[Hình 3.1 : Board Arduino ESP8266 Wifi UNO 19](#_Toc511852976)

[Hình 3.2 : OLED 20](#_Toc511852977)

[Hình 3.3 : DHT11 20](#_Toc511852978)

[Hình 3.4 : Cảm biến lửa 21](#_Toc511852979)

[Hình 3.5 : Cảm biến chuyển động 21](#_Toc511852980)

[Hình 3.6 : Cảm biến khoảng cách 22](#_Toc511852981)

[Hình 3.7 : Động cơ servo 22](#_Toc511852982)

[Hình 3.8 : RFID 23](#_Toc511852983)

[Hình 3.9 : Raspberry Pi 23](#_Toc511852984)

[Hình 3.10 : Module Raspberry Pi Camera 25](#_Toc511852985)

[Hình 4.1: Mô hình cơ bản của giao thức MQTT 28](#_Toc511852986)

[Hình 4.2: Cơ chế hoạt động của giao thức MQTT 29](#_Toc511852987)

[Hình 6.1 : Tạo một project (1) 35](#_Toc511852988)

[Hình 6.2 : Tạo một project (2) 36](#_Toc511852989)

[Hình 6.3 : Kết quả sau khi tạo một project 36](#_Toc511852990)

[Hình 6.4 : Tạo DataModel 37](#_Toc511852991)

[Hình 6.5 : Nội dung file Data.cs 37](#_Toc511852992)

[Hình 6.6 : Nội dung file IDataRepo.cs 37](#_Toc511852993)

[Hình 6.7 : Nội dung file DataRepo.cs (1) 38](#_Toc511852994)

[Hình 6.8 : Nội dung file DataRepo.cs (2) 39](#_Toc511852995)

[Hình 6.9 : Nội dung file DataRepo.cs (3) 39](#_Toc511852996)

[Hình 6.10 : Tạo UserModel 40](#_Toc511852997)

[Hình 6.11 : Nội dung file User.cs 40](#_Toc511852998)

[Hình 6.12 : Nội dung file IUserRepo.cs 40](#_Toc511852999)

[Hình 6.13 : Nội dung file UserRepo.cs (1) 40](#_Toc511853000)

[Hình 6.14 : Nội dung file UserRepo.cs (2) 41](#_Toc511853001)

[Hình 6.15 : Tạo Controller 41](#_Toc511853002)

[Hình 6.16 : Nội dung file PinController.cs (1) 42](#_Toc511853003)

[Hình 6.17 : Nội dung file PinController.cs (2) 42](#_Toc511853004)

[Hình 6.18 : Nội dung file PinController.cs (3) 43](#_Toc511853005)

[Hình 6.19 : Nội dung file PinController.cs (4) 43](#_Toc511853006)

[Hình 6.20 : Nội dung file UserController.cs (1) 44](#_Toc511853007)

[Hình 6.21 : Nội dung file UserController.cs (2) 45](#_Toc511853008)

[Hình 6.22 : Nội dung file UserController.cs (3) 45](#_Toc511853009)

[Hình 6.23 : Nội dung file lauchSetting.json 46](#_Toc511853010)

[Hình 6.24 : Run project bằng ServerAPI 46](#_Toc511853011)

[Hình 6.25 : Kết quả sai khi chạy Server 47](#_Toc511853012)

[Hình 6.26 : Action Get() 47](#_Toc511853013)

[Hình 6.27 : Action Get(int pin) 47](#_Toc511853014)

[Hình 6.28 : Action Post(Data dt) 48](#_Toc511853015)

[Hình 6.29 : Chạy Ngrok 49](#_Toc511853016)

[Hình 6.30 : Kết quả sau khi chạy Ngrok 49](#_Toc511853017)

[Hình 6.31 : Kết quả kiểm tra kết nối Ngrok trên điện thoại 50](#_Toc511853018)

[Hình 7.1 : Hàm kết nối Wifi 52](#_Toc511853019)

[Hình 7.2 : Vị trí đặt hàm setupWifi 52](#_Toc511853020)

[Hình 7.3 : Hàm getHttpResponse() 53](#_Toc511853021)

[Hình 7.4 : Hàm Post() 54](#_Toc511853022)

[Hình 7.5 : Hàm displayStringAsJson() 54](#_Toc511853023)

[Hình 7.6 : Hàm displayStringAsJsonArray() 55](#_Toc511853024)

[Hình 7.7 : Hàm convertStringToJson() 55](#_Toc511853025)

[Hình 7.8 : Hàm getIndexAtJsonArray() 56](#_Toc511853026)

[Hình 7.9 : Hàm setup\_rfid() 57](#_Toc511853027)

[Hình 7.10 : Hàm dump\_byte\_array() 57](#_Toc511853028)

[Hình 7.11 : Hàm startRFID() 57](#_Toc511853029)

[Hình 7.12 : Hàm setupOled() 58](#_Toc511853030)

**DANH SÁCH BẢNG**

[Bảng 2.1 : Lịch sử ra đời của ASP.NET Framework 16](#_Toc511853031)

[Bảng 2.2 : Lịch sử ra đời của ASP.NET Core 16](#_Toc511853032)

[Bảng 2.3 : So sánh ASP.NET Framework và ASP.NET Core 17](#_Toc511853033)

[Bảng 2.4 : So sánh ASP.NET Core và NodeJS 19](#_Toc511853034)

1. CHƯƠNG 1: GIỚI THIỆU
2. Internet of Things là một hệ thống các thiết bị đồ dùng được kết nối với nhau qua mạng Internet. Chúng có khả năng trao đổi và truyền tải thông tin, dữ liệu một cách hiệu quả, tiện lợi thông qua mạng Internet mà không cần sự tương tác trực tiếp giữa người với thiết bị hay giữa người với người. Điều đó có nghĩa là khi mọi thiết bị đã được “Internet hóa”, chỉ với một thiết bị thông minh, chẳng hạn như Smart tivi, Smartphone hay thậm chí chỉ bằng một chiếc smartwatch nhỏ bé trên tay đã được hỗ trợ IoT, người dùng có thể điều khiển chúng mọi lúc mọi nơi mà không bị giới hạn về mặt thời gian và không gian.
3. Mọi thứ con người cần chỉ là một thiết bị Smart phone để có thể theo dõi và điều khiển mọi thứ và tất nhiên là phải cần có kết nối Internet. IoT – Internet of Things đã giúp chúng ta làm được điều đó, IoT có khả năng thay đổi thế giới giống như cách mà Internet đã thay đổi cuộc sống của chúng ta.
4. Vì lí do đó mà nhóm đã chọn đề tài “Phát triển giải pháp xuất nhập cho nhà thông minh sử dụng điện toán đám mây”. Mục tiêu nhóm hướng đến là có thể tự động hóa trong việc bảo mật, cửa tự động cho smart house.
5. CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/API
   1. ASP.NET
      1. Tổng quan
6. 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.
7. 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
8. 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

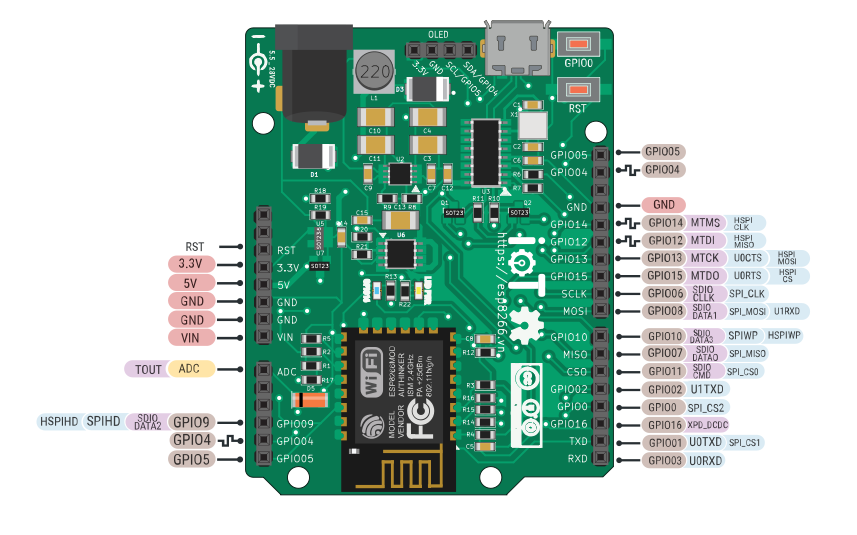
* 1. NodeJS
     1. Tổng quan

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 .

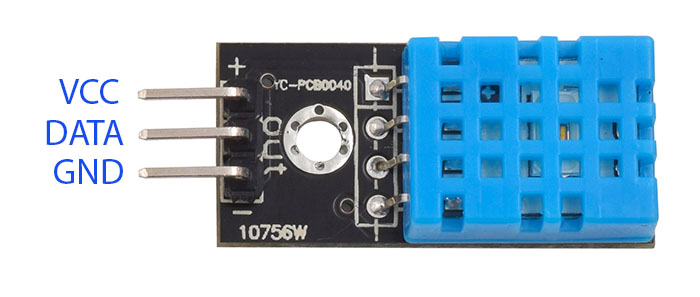
Được hỗ trợ đa nền tảng bao gồm Windows , Mac OS , Linux. Phiên bản hiện tại là Arduino IDE v1.8.5

* + 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

1. Raspberry Pi là cái máy tính giá 35USD có kích thước nhỏ và chạy hệ điều hành linux với mục tiêu chính là giảng dạy máy tính cho trẻ em. Raspberry Pi được phát triển bởi Raspberry Pi Foundation - tổ chức phi lợi nhuận với tiêu chí xây dựng hệ thống và nhiều người có thể sử dụng được trong những công việc tùy biến khác nhau.



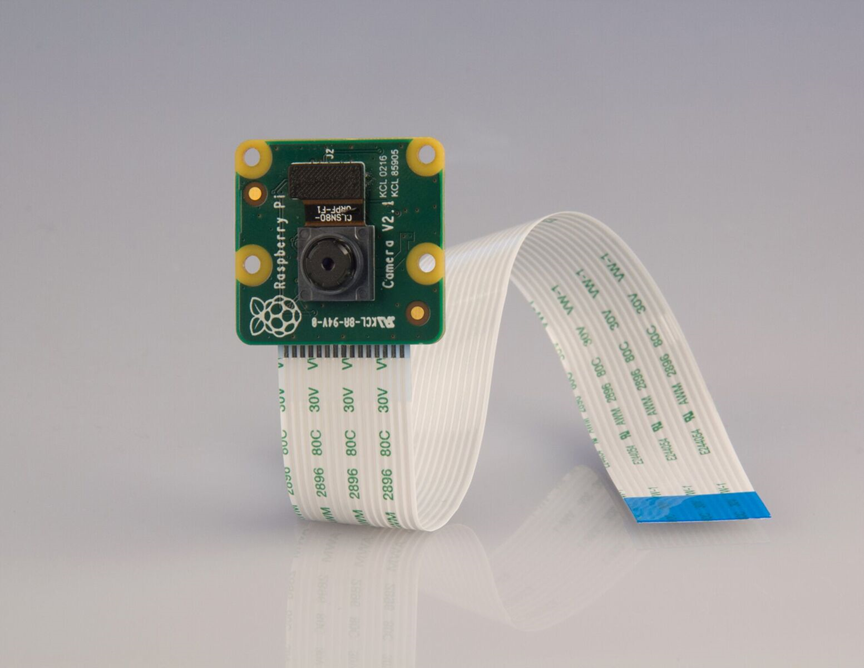
Hình 3.9 : Raspberry Pi

* + 1. Phần cứng

1. Hiện nay có nhiều dòng Raspberry Pi, mỗi dòng đều có một cấu trúc riêng, tuy nhiên về cơ bản, chúng có cấu trúc gần giống nhau. Cấu trúc cơ bản của một Raspberry Pi như sau:

* CPU.
* Khe cắm thẻ micro SD.
* Cổn USB.
* Cổng Ethernet.
* Cổng HDMI.
* Ngõ ra Audio-Video.
* Cổng cấp nguồn Micro USB.
* Cổng DSI (Display Serial Interface).
* Cổng CSI (Camera Serial Interface).
* GPIO (General Purpose Input and Output).
  + 1. Hệ điều hành:

1. Raspberry Pi có thể chạy được trên nhiều OS linux khác nhau. Một số Distributions Linux chạy trên Raspberry Pi như Raspbian, Pidora, openSUSE, OpenWRT, …
2. Trong phạm vi sử dụng cho đề tài này, sử dụng hệ điều hành Raspbian. Đây là bản build Linux dựa trên Debian với giao diện LXDE (thay vì GNOME). Hệ điều hành này có đầy đủ web browser, media player, tools, … thích hợp cho những người muốn dùng Raspberry Pi như một PC.
   * 1. Module sử dụng: Raspberry Pi Camera Module:



Hình 3.10 : Module Raspberry Pi Camera

**Một số thông số kỹ thuật:**

* Ống kính tiêu cự cố định.
* Cảm biến độ phân giải 5megapixel có khả năng chụp ảnh kích thước 3280 x 2464.
* Hỗ trợ video 1080p30, 720p60 và 640x480p90.
* Kích thước 25mm x 23mm x 9mm.
* Kết nối với Raspberry Pi thông qua cáp ribbon đi kèm dài 15cm.

1. CHƯƠNG 4: MỘT SỐ GIAO THỨC MẠNG
   1. Wifi
      1. Tổng quan
2. Wifi là từ viết tắt của Wireless Fidelity, nó là một hệ thống hoạt động dựa trên sóng vô tuyến không dây, hay còn được gọi là mạng IEEE 802.11. Wifi cho phép truy cập mạng internet ở một khoảng cách xác định mà không cần kết nối vật lý. Kết nối Wifi thường là sự lựa chọn hàng đầu của rất nhiều kỹ sư bởi tính thông dụng và kinh tế và mạng LAN với mô hình kết nối trong một phạm vi địa lý có giới hạn.
   * 1. Các chuẩn Wifi:
3. Sóng Wifi ngày nay được sử dụng rộng rãi phục vụ đời sống con người, đặc điểm của sóng wifi là thu phát ở tầng số từ 2.4GHz đến 5GHz cao hơn so với sóng vô tuyến truyền hình, sóng điện thoại và radio nên khá an toàn trong vấn đề bảo toàn thông tin khi truyền và nhận dữ liệu. Các chuẩn wifi hiện nay:

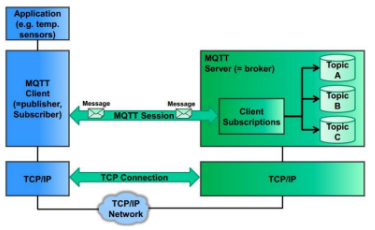
* Chuẩn 802.11b: thu phát ở tần số 2.4GHz và có tốc độ truyền nhận dữ liệu là 11 Mb/s.
* Chuẩn 802.11a: hoạt động ở tần số 5GHz và có tốc độ truyền 54Mb/s.
* Chuẩn 802.11g: thu phát ở tần số 2.4GHz và tốc độ truyền là 54Mb/s.
* Chuẩn 802.11n: hoạt động trên băng tần 2.4GHz nhưng tốc độ tăng lên đáng kể. Lên đến 300Mb/s và sau một thời gian ngắn đã tăng lên 450 Mb/s.
* Chuẩn 802.11ac: hoạt động ở băng tần 5GHz, chuẩn ac là một trong những chuẩn wifi có số thiết bị được tích hợp rộng lớn nhất hiện nay.
  1. MQTT
     1. Tổng quan

1. 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

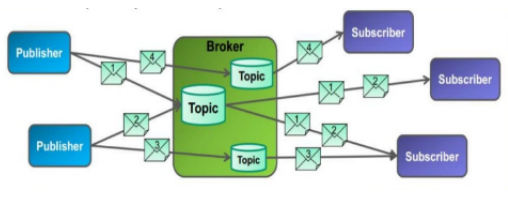
1. Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client - gọi tắt là client) kết nối tới một MQTT server (gọi là Broker). Mỗi client sẽ đăng ký một vài kênh (topic),ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là "Subscribe",giống như chúng ta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

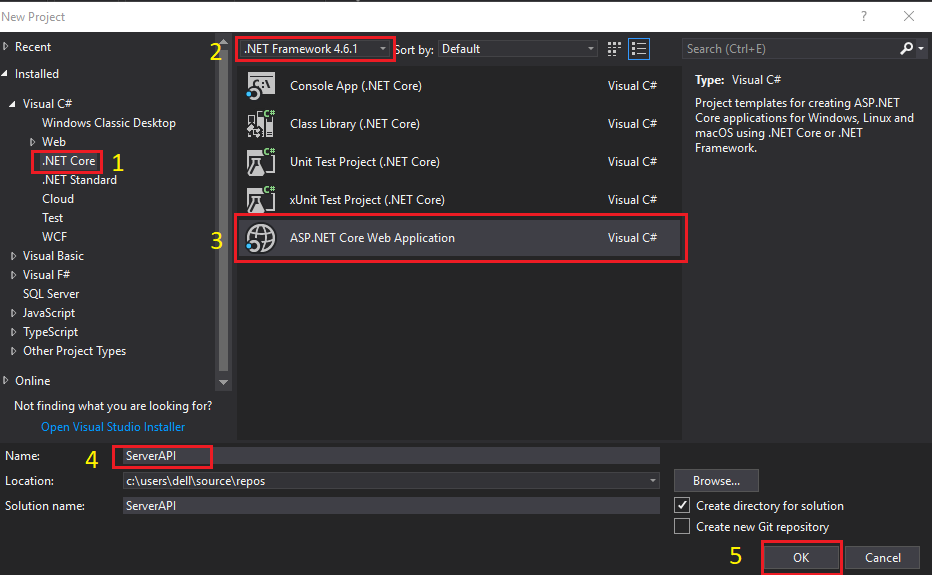
1. Mỗi client MQTT phải hỗ trợ TCP và thường sẽ giữ một kết nối mở đến broker ở mọi thời điểm. Đối với một số môi trường mà xác suất mất bản tin cao hay không có sẵn máy tính thì đây là cả một vấn đề.
2. Tên kênh MQTT thường là các chuỗi dài làm chúng không thỏa đáng với tiêu chuẩn 802.15.4.
   * 1. 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:

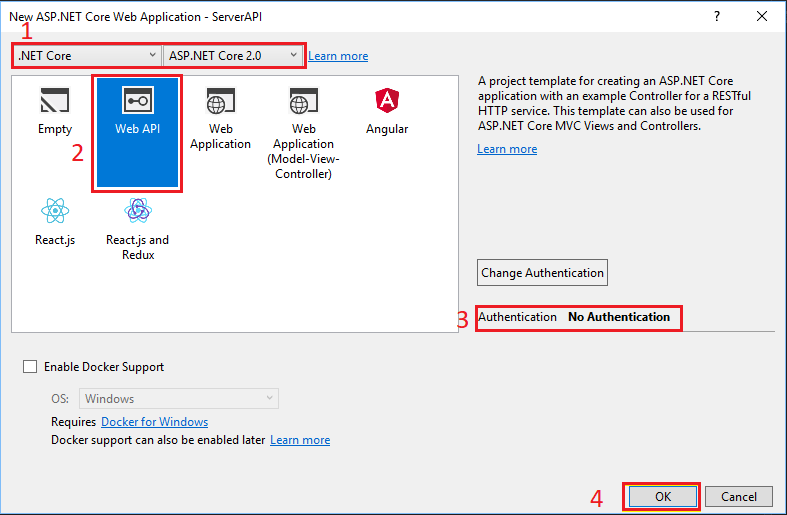
1. Bởi vì CoAP được xây dựng trên UDP chứ không phải TCP, SSL/TLS không có sẵn để cung cấp khả năng bảo mật. DTLS (Datagram Transport Layer Security) cung cấp sự đảm bảo giống như TLS nhưng để truyền dữ liệu qua UDP. Thông thường, các thiết bị DTLS sẽ hỗ trợ RSA và AES hoặc ECC và AES.
   1. So sánh các giao thức mạng
2. CHƯƠNG 5: KỊCH BẢN THỰC HIỆN
3. Xây dựng và phát triển giải pháp xuất nhập cho nhà thông minh sử dụng điện toán đám mây. Kịch bản được xây dựng như sau:
4. Mỗi thành viên trong nhà sẽ có một thẻ RFID có chức năng như là một chìa khóa cửa. Khi người sử dụng quét thẻ, hệ thống sẽ kiểm tra thông tin thẻ, nếu hợp lý thì sẽ gửi tín hiện để cửa tự động mở. Sau khi người sử dụng vào nhà cửa sẽ tự động đóng lại.
5. Đối với mỗi thành viên khác nhau sẽ dùng một thiết bị tương ứng để tượng trưng sau khi quét thẻ . Ví dụ :

* Thành viên thứ nhất quét thẻ : cửa mở và đèn sáng
* Thành viên thứ hai quét thẻ : cửa mở và loa kêu

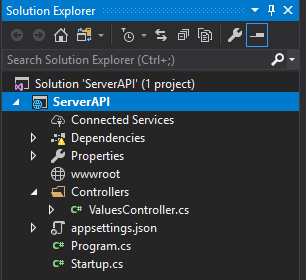
1. Khi người sử dụng bật chế độ security, hệ thống camera sẽ bắt đầu hoạt động. Hệ thống camera sẽ quay video trực tiếp và gửi về server. Ở server sẽ sử dụng kỹ thuật nhận dạng để nhận dạng có người xuất hiện hay không. Nếu có thì sẽ gửi thông báo đến cho chủ nhà. Tất cả quá trình này đều thực hiện trên cloud, vì vậy dù chủ nhà ở bất cứ đâu vẫn có thể nhận được thông tin.
2. 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)

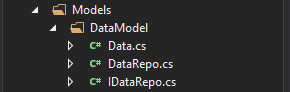


Hình 6.3 : Kết quả sau khi tạo một project

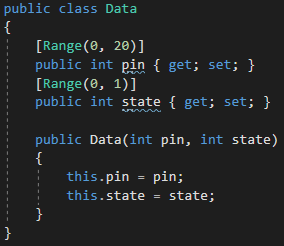
* + 1. Bước 2 : Tạo model để quản lí dữ liệu
       1. DataModel

Dùng để quản lí các chân GPIO của Arduino. Bao gồm :

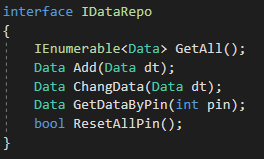
* Class Data.cs : dùng để định nghĩa các thuộc tính dữ liệu của một GPIO
* Interface IDataRepo.cs : dùng để định nghĩa các phương thức
* Class DataRepo.cs : dùng để implement các phương thức từ IDataRepo.cs



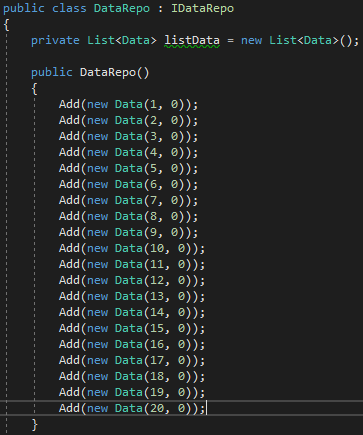
Hình 6.4 : Tạo DataModel



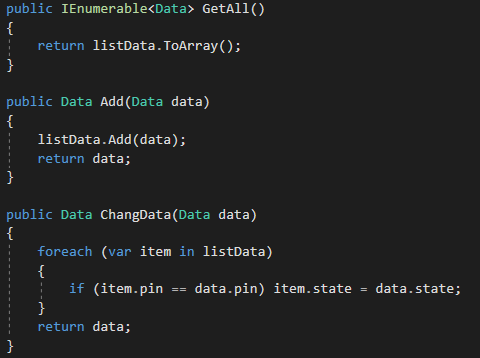
Hình 6.5 : Nội dung file Data.cs



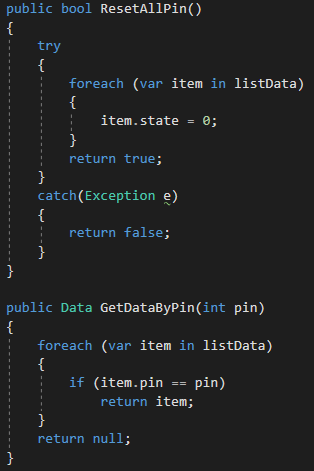
Hình 6.6 : Nội dung file IDataRepo.cs



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



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

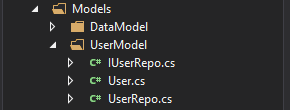


Hình 6.9 : Nội dung file DataRepo.cs (3)

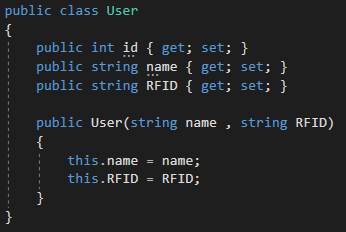
* + - 1. UserModel

Dùng để quản lí User (chưa sử dụng database) . Bao gồm :

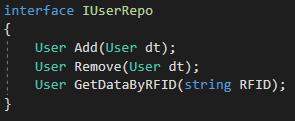
* Class User.cs : dùng để định nghĩa các thuộc tính của một user
* Interface IUserRepo.cs : dùng để định nghĩa các phương thức
* Class UserRepo.cs : dùng để implement các phương thức từ IUserRepo.cs



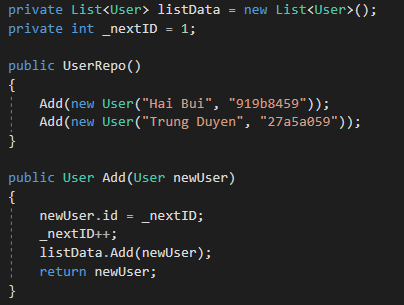
Hình 6.10 : Tạo UserModel



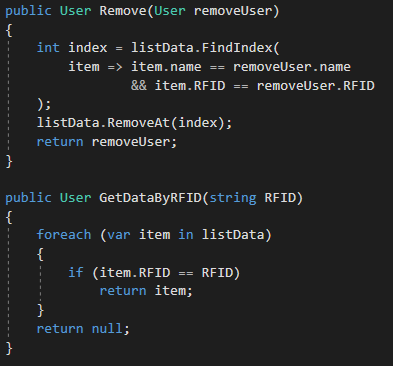
Hình 6.11 : Nội dung file User.cs



Hình 6.12 : Nội dung file IUserRepo.cs



Hình 6.13 : Nội dung file UserRepo.cs (1)



Hình 6.14 : Nội dung file UserRepo.cs (2)

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

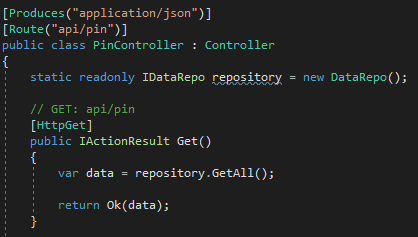


Hình 6.15 : Tạo Controller

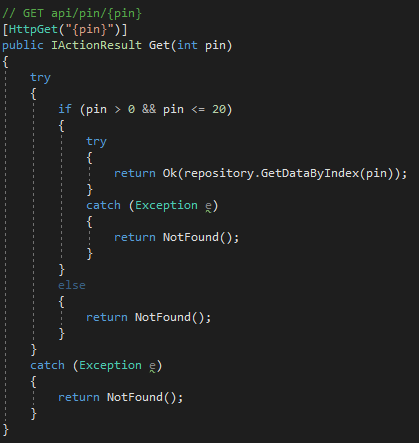
* + - 1. PinController

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

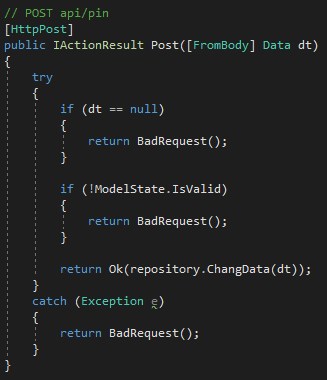
* Action Get() : Mặc định khi chạy server sẽ chạy hàm này đầu tiên , kết quả trả về là một mảng Data .
* Action Get(int pin) : Hàm này sẽ trả về đối tượng Data có giá trị pin bằng giá trị pin được yêu cầu .
* Action Post([FromBody] Data dt) : Hàm này sẽ nhận một đối tượng Data từ Request và chỉnh sửa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



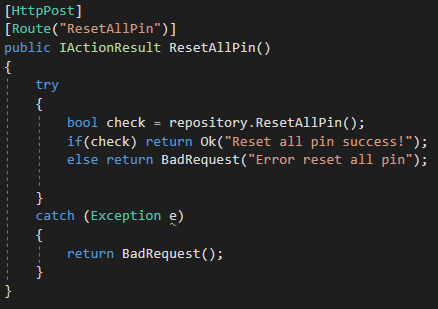
Hình 6.16 : Nội dung file PinController.cs (1)



Hình 6.17 : Nội dung file PinController.cs (2)



Hình 6.18 : Nội dung file PinController.cs (3)

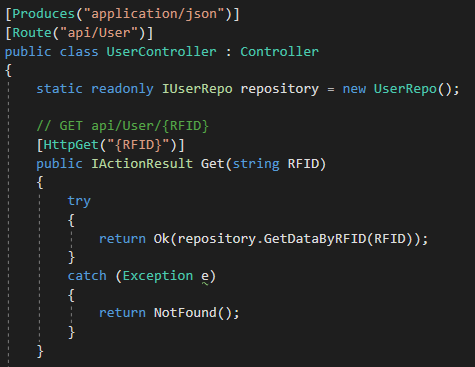


Hình 6.19 : Nội dung file PinController.cs (4)

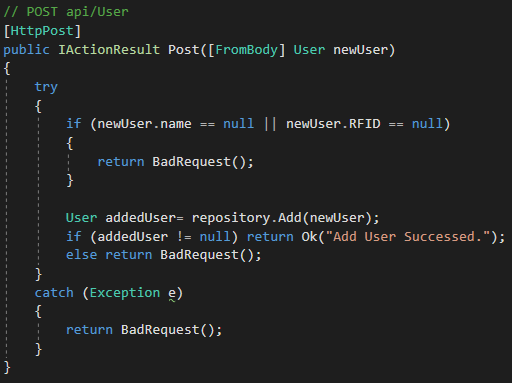
* + - 1. UserController

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

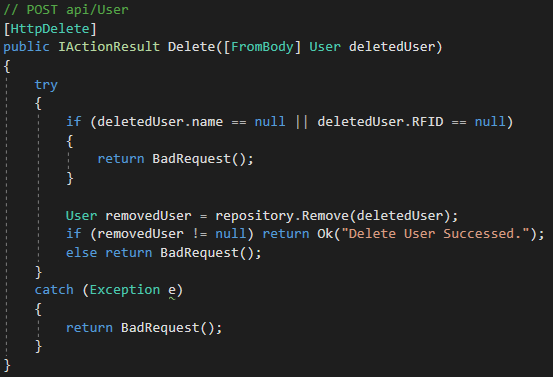
* Action Get(string RFID): Hàm này trả về một đối tượng có giá trị RFID tương ứng
* Action Post([FromBody] User newUser): Hàm này tạo ra một User mới
* Action Delete([FromBody] User deletedUser): Hàm này dung để xóa User



Hình 6.20 : Nội dung file UserController.cs (1)



Hình 6.21 : Nội dung file UserController.cs (2)



Hình 6.22 : Nội dung file UserController.cs (3)

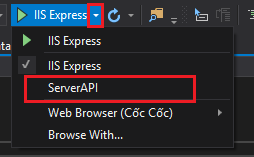
* + 1. Bước 4 : Thay đổi nội dung file Properties => lauchSettings.json

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



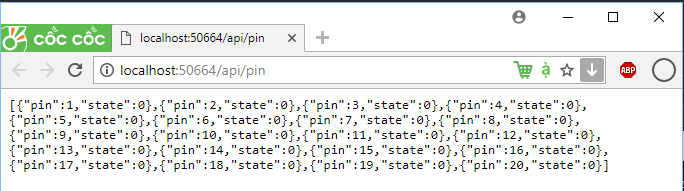
Hình 6.23 : Nội dung file lauchSetting.json

* + 1. Bước 5 : Chạy Server



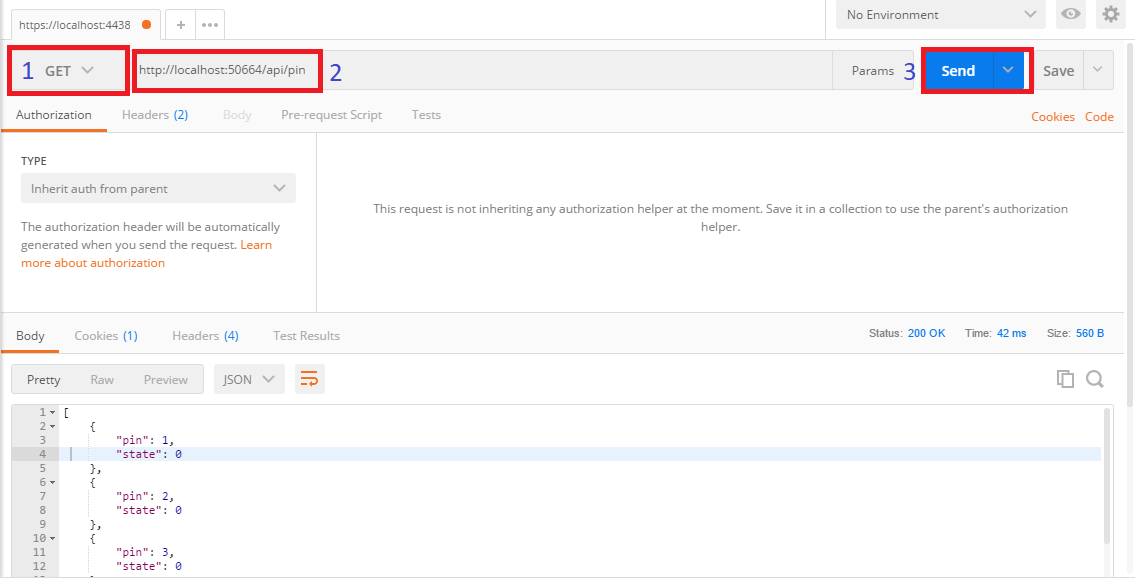
Hình 6.24 : Run project bằng ServerAPI

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

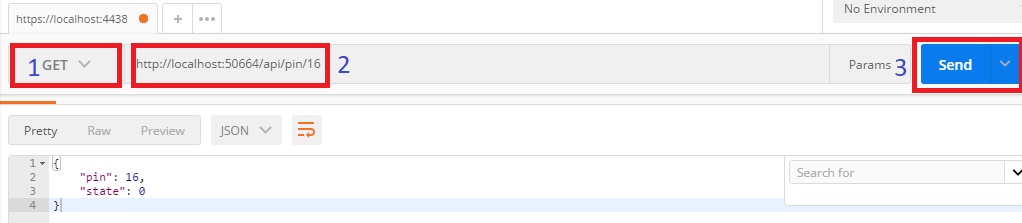


Hình 6.25 : Kết quả sai khi chạy Server

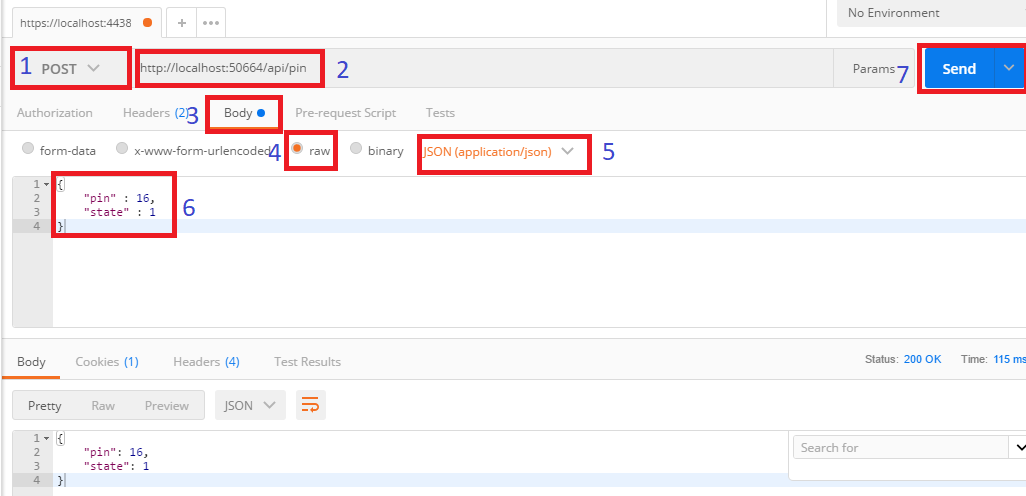
* + 1. Bước 7 : Test các hàm trong file PinController.cs bằng Postman



Hình 6.26 : Action Get()



Hình 6.27 : Action Get(int pin)

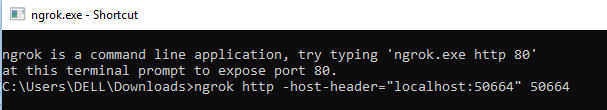


Hình 6.28 : Action Post(Data dt)

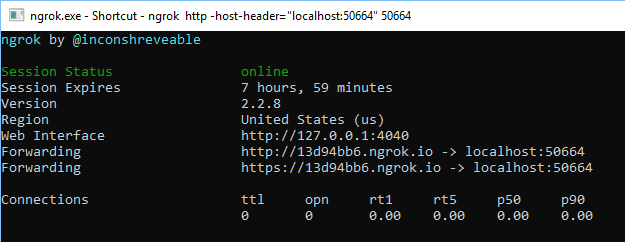
1. Tương tự đối với UserController
   1. Ngrok
      1. Tại sao lại cần sử dụng Ngrok trong đồ án này ?
2. Mặc định thì Visual Studio sẽ chặn tất cả các kết nối , request từ bên ngoài ngoại trừ ứng dụng trên chính máy chạy Server . Vì thế khi bạn sử dụng điện thoại hoặc Arduino ESP8266 truy cập đến localhost của Server , mặc dù chung mạng LAN với máy chạy Server nhưng vẫn không kết nối, lấy dữ liệu từ Server được.
3. Để có thể kết nối dễ dàng thì ta sẽ sử dụng đến phần mềm tạo tunnel - Ngrok
   * 1. Tổng quan
4. 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
5. 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 6.29 : Chạy Ngrok



Hình 6.30 : 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. Limited connection
2. Đối với Ngrok phiên bản free thì chúng ta chỉ được giới hạn 20 connections trong 1 phút , để tăng số lượng connection thì ta có thể làm như sau :

* Cách 1 : Tạo tài khoản trên trang chủ ngrok và chạy lệnh

***ngrok authtoken <your token>***

để thiết lập account cho Ngrok . Khi đó chúng ta được tối đa 40 connections trong 1 phút

* Cách 2 : Trả phí chúng ta sẽ được 60 connections trong 1 phút
  + 1. Kiểm tra kết nối đến Ngrok

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



Hình 6.31 : 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 .

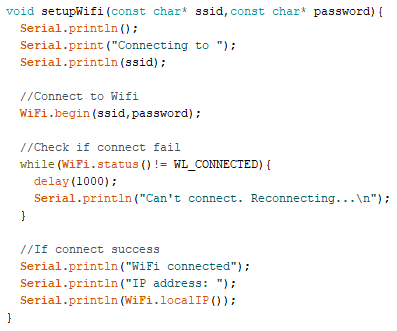
Phần tiếp theo ta sẽ kết nối đến Server thông qua Arduino ESP8266

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**

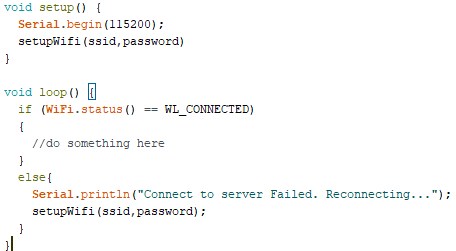
* 1. Kết nối Wifi
     1. Import thư viện

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

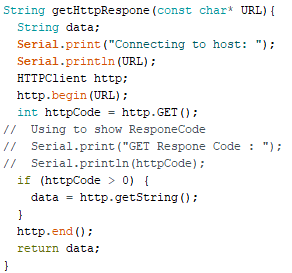
* + 1. Vị trí đặt hàm setupWifi()



Hình 7.2 : Vị trí đặt hàm setupWifi

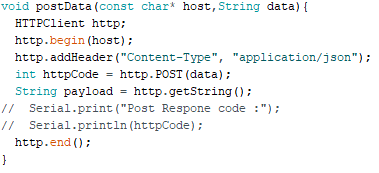
* 1. Kết nối Local Server
     1. Import thư viện

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()

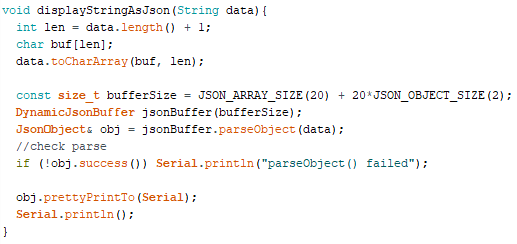
* 1. Post dữ liệu lên Server



Hình 7.4 : Hàm Post()

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

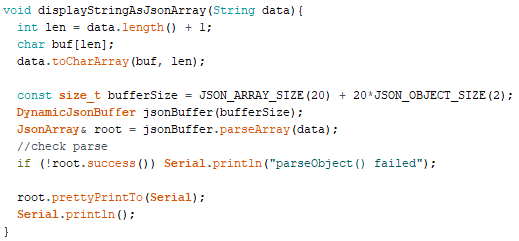
1. Dùng để hiển thị một đối tượng được trả về từ Server dưới dạng Json
2. “bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/



Hình 7.5 : Hàm displayStringAsJson()

* + 1. Hàm displayStringAsJsonArray()

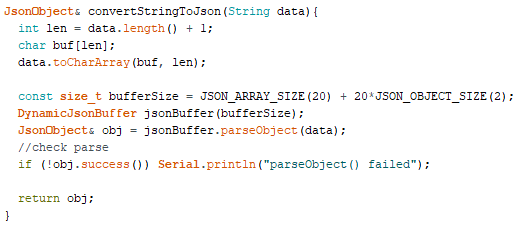
1. Dùng để hiển thị một danh sách các đối tượng được trả về từ Server dưới dạng Json
2. “bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/



Hình 7.6 : Hàm displayStringAsJsonArray()

* + 1. Hàm convertStringToJson()

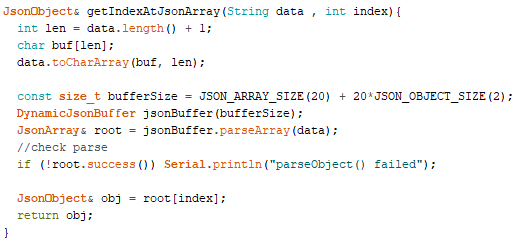
1. Convert chuỗi lấy được từ Server thành Json để dễ dàng dữ lý



Hình 7.7 : Hàm convertStringToJson()

* + 1. Hàm getIndexAtJsonArray()

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



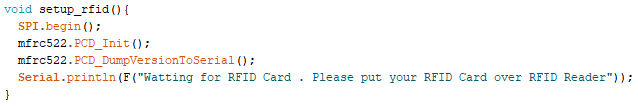
Hình 7.8 : Hàm getIndexAtJsonArray()

* 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

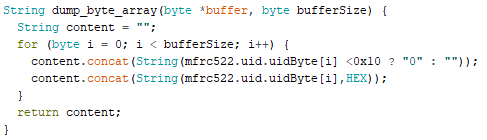
* SDA(SS) 15
* SCK 14
* MOSI 13
* MISO 12
* RST 2
  + 1. Khai báo kết nối

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



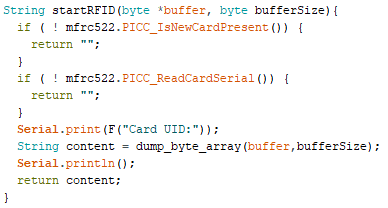
Hình 7.9 : Hàm setup\_rfid()

* + 1. Hàm dump\_byte\_array()



Hình 7.10 : Hàm dump\_byte\_array()

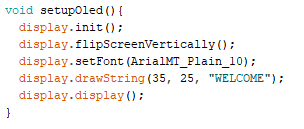
* + 1. Hàm startRFID()



Hình 7.11 : Hàm startRFID()

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

1. #include <Adafruit\_NeoPixel.h>
2. #include "SSD1306.h"
3. #include "Wire.h"
   * 1. Khai báo kết nối
4. // SCL pin 4
5. // SDA pin 5
6. SSD1306 display(0x3c, 4, 5);
   * 1. Hàm setupOled()
7. Đặt hàm trong hàm setup()



Hình 7.12 : Hàm setupOled()

* + 1. Code thay đổi screen

1. display.clear();
2. display.drawString(35, 15, "WELCOME");
3. display.drawString(35, 35, String(username));
4. display.display();

//Đặt đoạn code vào những chỗ phù hợp

* 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. Cài đặt hệ điều hành Raspbian trên Linux:
      1. Cài đặt

* Bước 1: Tải bản Raspbian stretch with desktop ở địa chỉ

https://www.raspberrypi.org/Downloads/raspbian/

* Bước 2: Giải nén file mới tải về.
* Bước 3: Cắm thẻ nhớ vào (thẻ nhớ từ 4GB trở lên), mở terminal và thực hiện lệnh

sudo fdish -l để xác định tên disk, thông thường là /dev/mmcblk0

* Bước 4: Vào thư mục có lưu file giải nén, thực hiện lệnh:

sudo dd bs=4M if=2018-03-13-raspbian-stretch.img of=/dev/mmcblk0

* Bước 5: Sau khi thực hiện xong, cắm thẻ nhớ vào Raspberry Pi và sử dụng.
  + 1. Kết nối wifi và SSH:
       1. Kết nối wifi:

1. Rút thẻ nhớ vửa tạo ra và cắm lại. Lúc này trên máy Linux sẽ mount 2 phân vùng bao gồm boot và phân vùng root.
2. Truy cập tới phân vùng root, vào thư mục /etc/wpa-supplicant/ và sử file wpa-supplicant.conf như sau:

ctrk\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev

update\_config=1

network={

ssid=”your ssid”

psk=”your pass”

}

* + 1. Kết nối SSH:

1. Từ tháng 11/2016 Raspberry Pi đã khóa SSH khi khởi động. Để mở khóa SSH cần tạo 1 file ssh (file này không có phần mở rộng và là file rỗng) trong phân vùng boot của thẻ SD.
2. Cắm thẻ nhớ vào Raspberry Pi.
3. Mở chương trình scan địa chỉ IP của Raspberry Pi, sau đó mở terminal lên và ssh vào: ssh pi@a.a.a.a. Trong đó a.a.a.a là địa chỉ của Raspberry Pi đã scan được.
4. Sau đó thiết lập cấu hình ban đầu cho Raspberry Pi.
   1. OpenCV
      1. Giới thiệu
5. OpenCV là một thư viện mã nguồn mở phục vụ cho việc nghiên cứu hat phát triển về thị giác máy tính, tối ưu hóa và xử lý các ứng dụng trong thời gian thực. Giúp cho việc xây dựng các ứng dụng xử lý ảnh, thị giác máy tính,… một cách nhanh hơn. OpenCV có hơn 500 hàm khác nhau, được chia làm nhiều phần phục vụ cho các công việc như: xử lý hình ảnh y tế, an ninh, camera quan sát, nhận diện,...
   * 1. Mục đích
6. Xử lý ảnh, quan sát, an ninh,… tất cả những gì liên quan đến thị giác máy tính thì có thể sử dụng OpenCV để làm.
   * 1. Cài đặt OpenCV
     2. Ứng dụng

* Nhận dạng ảnh.
* Xử lý hình ảnh.
* Phục hồi hình ảnh/video.
* Thực tế ảo.
* Các ứng dụng khác.
  1. FIX các lỗi đã gặp

1. Chương 9: xxxx
2. 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