**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**

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy Lê Trung Quân vì đã hướng dẫn trực tiếp cho nhóm trong suốt quá trình thực hiện đề tài này. Nhờ có thầy hướng dẫn, chỉ bảo và tạo điều kiện mà nhóm học hỏi được nhiều kiến thức mới, công nghệ mới mà trước đây chưa được biết, chưa được sử dụng, qua đó bổ sung kiến thức để hoàn thành đề tài này. Sự hỗ trợ và tư vấn của thầy đã giúp nhóm hoàn thành đề tài này.

Nhóm cũng xin gửi lời cám ơn đến quý thầy cô của trường Đại học Công nghệ thông tin nói chung và thầy cô khoa Mạng máy tính và Truyền thông nói riêng đã cung cấp thông tin bổ ích và chỉ bảo để nhóm có thể thực hiện đề tài này.

Cuối cùng nhóm xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn ở bên và động viên nhóm. Gia đình chính là chỗ dựa tinh thần vững chắc không chỉ trong quá trình học tập, nghiên cứu và còn trong mọi hoạt động hằng ngày của nhóm.

Mục lục

[1 CHƯƠNG 1: GIỚI THIỆU 18](#_Toc515567167)

[2 CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/API 19](#_Toc515567168)

[2.1 ASP.NET 19](#_Toc515567169)

[2.1.1 Tổng quan 19](#_Toc515567170)

[2.1.2 ASP.NET Framework 19](#_Toc515567171)

[2.1.3 ASP.NET Core 21](#_Toc515567172)

[2.1.4 So sánh ASP.NET Framework và ASP.NET Core 22](#_Toc515567173)

[2.2 Node.JS 22](#_Toc515567174)

[2.2.1 Tổng quan 22](#_Toc515567175)

[2.2.2 Lịch sử phát triển 23](#_Toc515567176)

[2.2.3 Một số đặc điểm của NodeJS 23](#_Toc515567177)

[2.3 So sánh ASP.NET Core và NodeJS 24](#_Toc515567178)

[3 CHƯƠNG 3: MỘT SỐ MODULE ĐƯỢC SỬ DỤNG TRONG MÔ HÌNH 25](#_Toc515567179)

[3.1 Arduino ESP8266 Wifi Uno 25](#_Toc515567180)

[3.1.1 Tổng quan 25](#_Toc515567181)

[3.1.2 Arduino IDE 25](#_Toc515567182)

[3.2 Một số module và sensor hỗ trợ Arduino ESP8266 26](#_Toc515567183)

[3.2.1 OLED 128x64 0.96 inch – 12C 26](#_Toc515567184)

[3.2.2 Cảm biến nhiệt độ, độ ẩm DHT11 26](#_Toc515567185)

[3.2.3 Cảm biến phát hiện lửa 27](#_Toc515567186)

[3.2.4 Cảm biến chuyển động HC-SR501 27](#_Toc515567187)

[3.2.5 Cảm biến khoảng cách HC-SR04 28](#_Toc515567188)

[3.2.6 Động cơ Servo 28](#_Toc515567189)

[3.2.7 RFID RC522 29](#_Toc515567190)

[3.3 Raspberry Pi 3 29](#_Toc515567191)

[3.3.1 Tổng quan 29](#_Toc515567192)

[3.3.2 Phần cứng 30](#_Toc515567193)

[3.3.3 Hệ điều hành: 30](#_Toc515567194)

[3.3.4 Module sử dụng: Raspberry Pi Camera Module 31](#_Toc515567195)

[4 CHƯƠNG 4: MỘT SỐ CÁCH THỨC TRUYỀN DỮ LIỆU 32](#_Toc515567196)

[4.1 Wifi 32](#_Toc515567197)

[4.1.1 Tổng quan 32](#_Toc515567198)

[4.1.2 Các chuẩn Wifi: 32](#_Toc515567199)

[4.1.3 Nguyên lý hoạt động: 32](#_Toc515567200)

[4.1.4 Các phương pháp bảo mật Wifi 33](#_Toc515567201)

[4.1.5 Ưu điểm của mạng Wifi 34](#_Toc515567202)

[4.1.6 Nhược điểm của mạng Wifi 34](#_Toc515567203)

[4.2 Bluetooth 34](#_Toc515567204)

[4.2.1 Tổng quan 34](#_Toc515567205)

[4.2.2 Các đặc điểm của Bluetooth 34](#_Toc515567206)

[4.2.3 Các tầng giao thức Bluetooth 35](#_Toc515567207)

[4.2.4 Ưu điểm của Bluetooth 36](#_Toc515567208)

[4.2.5 Nhược điểm của Bluetooth 36](#_Toc515567209)

[4.3 Z-Wave 36](#_Toc515567210)

[4.3.1 Tổng quan 36](#_Toc515567211)

[4.3.2 Đặc điểm của Z-Wave 37](#_Toc515567212)

[4.3.3 Các tầng giao thức Z-Wave 37](#_Toc515567213)

[5 CHƯƠNG 5: KỊCH BẢN THỰC HIỆN 38](#_Toc515567214)

[6 CHƯƠNG 6 : ENTITY FRAMEWORK 41](#_Toc515567215)

[6.1 Tổng quan 41](#_Toc515567216)

[6.2 Kiến trúc của Entity Framework 41](#_Toc515567217)

[6.3 Đặc điểm của Entity Framework 42](#_Toc515567218)

[6.4 Lợi ích khi sử dụng Entity Framework 42](#_Toc515567219)

[6.5 Thiết kế Entity Data Model: 42](#_Toc515567220)

[6.5.1 Database First: 43](#_Toc515567221)

[6.5.2 Model First: 43](#_Toc515567222)

[6.5.3 Code First: 43](#_Toc515567223)

[7 CHƯƠNG 7 : DATABASE 44](#_Toc515567224)

[7.1 SQLServer 44](#_Toc515567225)

[7.1.1 Khái niệm SQL Server 44](#_Toc515567226)

[7.1.2 Khái niệm về cấu trúc vật lý 44](#_Toc515567227)

[7.1.3 Lịch sử ra đời SQL Server 45](#_Toc515567228)

[7.1.4 Các thành phần cơ bản trong SQL Server 45](#_Toc515567229)

[7.1.5 Vì sao phát triển ứng dụng server-client? 46](#_Toc515567230)

[7.2 SQlite 47](#_Toc515567231)

[7.2.1 Khái niệm SQLite 47](#_Toc515567232)

[7.2.2 Đặc điểm của SQLite 47](#_Toc515567233)

[7.2.3 Lịch sử của SQLite 47](#_Toc515567234)

[7.2.4 Vì sao nên sử dụng SQLite? 47](#_Toc515567235)

[8 CHƯƠNG 8: XÂY DỰNG WEB API BẰNG ASP.NET CORE SỬ DỤNG SQLITE 49](#_Toc515567236)

[8.1 Các bước thực hiện 49](#_Toc515567237)

[8.1.1 Bước 1 : Tạo một project ASP.NET Core API 49](#_Toc515567238)

[8.1.2 Bước 2 : Tạo model để quản lí dữ liệu 50](#_Toc515567239)

[8.1.3 Bước 3 : Xây dựng ESPServerContext để giao tiếp với Database bằng CodeFirst 56](#_Toc515567240)

[8.1.4 Bước 4 : Tạo dữ liệu đầu vào 56](#_Toc515567241)

[8.1.5 Bước 5 : Đăng kí SeedData trong file Program.cs 58](#_Toc515567242)

[8.1.6 Bước 6 : Xóa ValueController.cs và tạo một Controller mới 58](#_Toc515567243)

[8.1.7 Bước 7 : Thay đổi nội dung file Properties => lauchSettings.json 63](#_Toc515567244)

[8.1.8 Bước 8 : Đăng kí ConnectionString trong file Startup.cs 63](#_Toc515567245)

[8.1.9 Bước 9: Thay đổi nội dung hàm Configure trong file Startup.cs 64](#_Toc515567246)

[8.1.10 Bước 10 : Install packet EntityFrameworkCore 64](#_Toc515567247)

[8.1.11 Bước 11 : Chạy Server 65](#_Toc515567248)

[8.1.12 Bước 12 : Kiểm tra kết quả trên trình duyệt 66](#_Toc515567249)

[8.1.13 Bước 13 : Test các hàm trong file PinController.cs bằng Postman 66](#_Toc515567250)

[8.2 Ngrok 67](#_Toc515567251)

[8.2.1 Tại sao lại cần sử dụng Ngrok trong đồ án này ? 67](#_Toc515567252)

[8.2.2 Tổng quan 67](#_Toc515567253)

[8.2.3 Đặc điểm của Ngrok 67](#_Toc515567254)

[8.2.4 Cài đặt Ngrok 68](#_Toc515567255)

[8.2.5 Chạy Ngrok 68](#_Toc515567256)

[8.2.6 Limited connection 69](#_Toc515567257)

[8.2.7 Kiểm tra kết nối đến Ngrok 69](#_Toc515567258)

[8.2.8 Kết luận 69](#_Toc515567259)

[9 CHƯƠNG 9: CODE ARDUINO 70](#_Toc515567260)

[9.1 Sơ lượt 70](#_Toc515567261)

[9.2 Kết nối Wifi 70](#_Toc515567262)

[9.2.1 Import thư viện 70](#_Toc515567263)

[9.2.2 Khai báo Wifi 70](#_Toc515567264)

[9.2.3 Hàm setupWifi() 71](#_Toc515567265)

[9.2.4 Vị trí đặt hàm setupWifi() 71](#_Toc515567266)

[9.3 Kết nối Local Server 72](#_Toc515567267)

[9.3.1 Import thư viện 72](#_Toc515567268)

[9.3.2 Khai báo kết nối 72](#_Toc515567269)

[9.3.3 Get dữ liệu từ Server 72](#_Toc515567270)

[9.3.4 Post dữ liệu lên Server 73](#_Toc515567271)

[9.4 Xử lý dữ liệu nhận được thành Json 73](#_Toc515567272)

[9.4.1 Hàm displayStringAsJson() 73](#_Toc515567273)

[9.4.2 Hàm displayStringAsJsonArray() 73](#_Toc515567274)

[9.4.3 Hàm convertStringToJson() 74](#_Toc515567275)

[9.4.4 Hàm getIndexAtJsonArray() 74](#_Toc515567276)

[9.5 Xử lý kết nối RFID 75](#_Toc515567277)

[9.5.1 Import thư viện 75](#_Toc515567278)

[9.5.2 Hướng dẫn nối chân dành cho ESP8266 Wifi UNO 75](#_Toc515567279)

[9.5.3 Khai báo kết nối 75](#_Toc515567280)

[9.5.4 Hàm setup\_rfid() 76](#_Toc515567281)

[9.5.5 Hàm dump\_byte\_array() 76](#_Toc515567282)

[9.5.6 Hàm startRFID() 76](#_Toc515567283)

[9.6 Xử lý kết nối OLED 76](#_Toc515567284)

[9.6.1 Import thư viện 76](#_Toc515567285)

[9.6.2 Khai báo kết nối 77](#_Toc515567286)

[9.6.3 Hàm setupOled() 77](#_Toc515567287)

[9.6.4 Code thay đổi screen 77](#_Toc515567288)

[9.7 FIX các lỗi đã gặp 77](#_Toc515567289)

[9.7.1 Lỗi ESP8266 không nhận Wifi 77](#_Toc515567290)

[9.7.2 Lỗi ESP8266 không kết nối được đến localhost 78](#_Toc515567291)

[9.7.3 Lỗi “Firmware Unknown” khi ESP8266 kết nối RFID 78](#_Toc515567292)

[10 CHƯƠNG 10: CODE RASPBERRY PI 79](#_Toc515567293)

[10.1 OpenCV 79](#_Toc515567294)

[10.1.1 Giới thiệu 79](#_Toc515567295)

[10.1.2 Mục đích 79](#_Toc515567296)

[10.1.3 Ứng dụng 79](#_Toc515567297)

[10.2 Kiến thức cơ bản về xử lý ảnh: 79](#_Toc515567298)

[10.2.1 Ảnh kỹ thuật số (Digital image): 79](#_Toc515567299)

[10.2.2 Độ phân giải ảnh (Resolution): 79](#_Toc515567300)

[10.2.3 Điểm ảnh (Pixel): 80](#_Toc515567301)

[10.2.4 Mức xám của ảnh (Grayscale): 80](#_Toc515567302)

[10.2.5 Ảnh màu: 80](#_Toc515567303)

[10.3 Code face-detection sử dụng Raspberry Pi, OpenCV 80](#_Toc515567304)

[10.3.1 Sơ lượt 80](#_Toc515567305)

[10.3.2 File camera.py 80](#_Toc515567306)

[10.3.3 File mail.py 82](#_Toc515567307)

[10.3.4 File main.py 82](#_Toc515567308)

[10.3.5 File requirements.txt 84](#_Toc515567309)

[10.3.6 Thư mục models 84](#_Toc515567310)

[10.3.7 Thư mục templates 84](#_Toc515567311)

[10.4 FIX các lỗi đã gặp 84](#_Toc515567312)

[10.4.1 Lỗi không ssh được Raspberry Pi 84](#_Toc515567313)

[10.4.2 Lỗi khi cài đặt Cryptography 85](#_Toc515567314)

[11 CHƯƠNG 11: CLOUD 85](#_Toc515567315)

[11.1 Tổng quan 85](#_Toc515567316)

[11.1.1 Khái niệm 85](#_Toc515567317)

[11.1.2 Thành phần 86](#_Toc515567318)

[11.1.3 Đặc tính 87](#_Toc515567319)

[11.1.4 Các mô hình triển khai 87](#_Toc515567320)

[11.2 Một số công cụ triển khai Web Server 87](#_Toc515567321)

[11.2.1 Internet Information Services 88](#_Toc515567322)

[11.2.2 Nginx 88](#_Toc515567323)

[11.3 Triển khai trên cloud Azure 89](#_Toc515567324)

[11.3.1 Đăng nhập vào Azure với tài khoản trường (tài khoản ms.uit.edu.vn) 89](#_Toc515567325)

[11.3.2 Tạo Web app 90](#_Toc515567326)

[11.3.3 Tạo storage lưu trữ hình ảnh 95](#_Toc515567327)

[12 Kết luận và hướng phát triển 97](#_Toc515567328)

[12.1 Kết luận 97](#_Toc515567329)

[12.2 Hạn chế 97](#_Toc515567330)

[12.3 Hướng phát triển 97](#_Toc515567331)

[13 Tài liệu tham khảo 98](#_Toc515567332)

[Mục lụcPAGEREF \_Toc515367925 \h3](#_Toc515367925)

[1 CHƯƠNG 1: GIỚI THIỆUPAGEREF \_Toc515367926 \h16](#_Toc515367926)

[2 CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/APIPAGEREF \_Toc515367927 \h17](#_Toc515367927)

[2.1 ASP.NETPAGEREF \_Toc515367928 \h17](#_Toc515367928)

[2.1.1 Tổng quanPAGEREF \_Toc515367929 \h17](#_Toc515367929)

[2.1.2 ASP.NET FrameworkPAGEREF \_Toc515367930 \h17](#_Toc515367930)

[2.1.3 ASP.NET CorePAGEREF \_Toc515367931 \h18](#_Toc515367931)

[2.1.4 So sánh ASP.NET Framework và ASP.NET CorePAGEREF \_Toc515367932 \h19](#_Toc515367932)

[2.2 Node.JSPAGEREF \_Toc515367933 \h20](#_Toc515367933)

[2.2.1 Tổng quanPAGEREF \_Toc515367934 \h20](#_Toc515367934)

[2.2.2 Lịch sử phát triểnPAGEREF \_Toc515367935 \h20](#_Toc515367935)

[2.2.3 Một số đặc điểm của NodeJSPAGEREF \_Toc515367936 \h20](#_Toc515367936)

[2.3 So sánh ASP.NET Core và NodeJSPAGEREF \_Toc515367937 \h21](#_Toc515367937)

[3 CHƯƠNG 3: MỘT SỐ MODULE ĐƯỢC SỬ DỤNG TRONG MÔ HÌNHPAGEREF \_Toc515367938 \h22](#_Toc515367938)

[3.1 Arduino ESP8266 Wifi UnoPAGEREF \_Toc515367939 \h22](#_Toc515367939)

[3.1.1 Tổng quanPAGEREF \_Toc515367940 \h22](#_Toc515367940)

[3.1.2 Arduino IDEPAGEREF \_Toc515367941 \h22](#_Toc515367941)

[3.2 Một số module và sensor hỗ trợ Arduino ESP8266PAGEREF \_Toc515367942 \h23](#_Toc515367942)

[3.2.1 OLED 128x64 0.96 inch – 12CPAGEREF \_Toc515367943 \h23](#_Toc515367943)

[3.2.2 Cảm biến nhiệt độ, độ ẩm DHT11PAGEREF \_Toc515367944 \h23](#_Toc515367944)

[3.2.3 Cảm biến phát hiện lửaPAGEREF \_Toc515367945 \h24](#_Toc515367945)

[3.2.4 Cảm biến chuyển động HC-SR501PAGEREF \_Toc515367946 \h24](#_Toc515367946)

[3.2.5 Cảm biến khoảng cách HC-SR04PAGEREF \_Toc515367947 \h25](#_Toc515367947)

[3.2.6 Động cơ ServoPAGEREF \_Toc515367948 \h25](#_Toc515367948)

[3.2.7 RFID RC522PAGEREF \_Toc515367949 \h26](#_Toc515367949)

[3.3 Raspberry Pi 3PAGEREF \_Toc515367950 \h26](#_Toc515367950)

[3.3.1 Tổng quanPAGEREF \_Toc515367951 \h26](#_Toc515367951)

[3.3.2 Phần cứngPAGEREF \_Toc515367952 \h27](#_Toc515367952)

[3.3.3 Hệ điều hành:PAGEREF \_Toc515367953 \h27](#_Toc515367953)

[3.3.4 Module sử dụng: Raspberry Pi Camera ModulePAGEREF \_Toc515367954 \h28](#_Toc515367954)

[4 CHƯƠNG 4: MỘT SỐ CÁCH THỨC TRUYỀN DỮ LIỆUPAGEREF \_Toc515367955 \h29](#_Toc515367955)

[4.1 WifiPAGEREF \_Toc515367956 \h29](#_Toc515367956)

[4.1.1 Tổng quanPAGEREF \_Toc515367957 \h29](#_Toc515367957)

[4.1.2 Các chuẩn Wifi:PAGEREF \_Toc515367958 \h29](#_Toc515367958)

[4.1.3 Nguyên lý hoạt động:PAGEREF \_Toc515367959 \h29](#_Toc515367959)

[4.1.4 Các phương pháp bảo mật WifiPAGEREF \_Toc515367960 \h30](#_Toc515367960)

[4.1.5 Ưu điểm của mạng WifiPAGEREF \_Toc515367961 \h30](#_Toc515367961)

[4.1.6 Nhược điểm của mạng WifiPAGEREF \_Toc515367962 \h31](#_Toc515367962)

[4.2 BluetoothPAGEREF \_Toc515367963 \h31](#_Toc515367963)

[4.2.1 Tổng quanPAGEREF \_Toc515367964 \h31](#_Toc515367964)

[4.2.2 Các đặc điểm của BluetoothPAGEREF \_Toc515367965 \h31](#_Toc515367965)

[4.2.3 Các tầng giao thức BluetoothPAGEREF \_Toc515367966 \h31](#_Toc515367966)

[4.2.4 Ưu điểm của BluetoothPAGEREF \_Toc515367967 \h32](#_Toc515367967)

[4.2.5 Nhược điểm của BluetoothPAGEREF \_Toc515367968 \h32](#_Toc515367968)

[4.3 Z-WavePAGEREF \_Toc515367969 \h33](#_Toc515367969)

[4.3.1 Tổng quanPAGEREF \_Toc515367970 \h33](#_Toc515367970)

[4.3.2 Đặc điểm của Z-WavePAGEREF \_Toc515367971 \h33](#_Toc515367971)

[4.3.3 Các tầng giao thức Z-WavePAGEREF \_Toc515367972 \h33](#_Toc515367972)

[5 CHƯƠNG 5: KỊCH BẢN THỰC HIỆNPAGEREF \_Toc515367973 \h33](#_Toc515367973)

[6 CHƯƠNG 6 : ENTITY FRAMEWORKPAGEREF \_Toc515367974 \h36](#_Toc515367974)

[6.1 Tổng quanPAGEREF \_Toc515367975 \h36](#_Toc515367975)

[6.2 Kiến trúc của Entity FrameworkPAGEREF \_Toc515367976 \h36](#_Toc515367976)

[6.3 Đặc điểm của Entity FrameworkPAGEREF \_Toc515367977 \h37](#_Toc515367977)

[6.4 Lợi ích khi sử dụng Entity FrameworkPAGEREF \_Toc515367978 \h37](#_Toc515367978)

[6.5 Thiết kế Entity Data Model:PAGEREF \_Toc515367979 \h37](#_Toc515367979)

[6.5.1 Database First:PAGEREF \_Toc515367980 \h38](#_Toc515367980)

[6.5.2 Model First:PAGEREF \_Toc515367981 \h38](#_Toc515367981)

[6.5.3 Code First:PAGEREF \_Toc515367982 \h38](#_Toc515367982)

[7 CHƯƠNG 7 : DATABASEPAGEREF \_Toc515367983 \h38](#_Toc515367983)

[7.1 SQLServerPAGEREF \_Toc515367984 \h38](#_Toc515367984)

[7.1.1 Khái niệm SQL ServerPAGEREF \_Toc515367985 \h38](#_Toc515367985)

[7.1.2 Khái niệm về cấu trúc vật lýPAGEREF \_Toc515367986 \h38](#_Toc515367986)

[7.1.3 Lịch sử ra đời SQL ServerPAGEREF \_Toc515367987 \h39](#_Toc515367987)

[7.1.4 Các thành phần cơ bản trong SQL ServerPAGEREF \_Toc515367988 \h40](#_Toc515367988)

[7.1.5 Vì sao phát triển ứng dụng server-client?PAGEREF \_Toc515367989 \h41](#_Toc515367989)

[7.2 SQlitePAGEREF \_Toc515367990 \h41](#_Toc515367990)

[7.2.1 Khái niệm SQLitePAGEREF \_Toc515367991 \h41](#_Toc515367991)

[7.2.2 Đặc điểm của SQLitePAGEREF \_Toc515367992 \h41](#_Toc515367992)

[7.2.3 Lịch sử của SQLitePAGEREF \_Toc515367993 \h42](#_Toc515367993)

[7.2.4 Vì sao nên sử dụng SQLite?PAGEREF \_Toc515367994 \h42](#_Toc515367994)

[8 CHƯƠNG 8: XÂY DỰNG WEB API BẰNG ASP.NET CORE SỬ DỤNG SQLITEPAGEREF \_Toc515367995 \h43](#_Toc515367995)

[8.1 Các bước thực hiệnPAGEREF \_Toc515367996 \h43](#_Toc515367996)

[8.1.1 Bước 1 : Tạo một project ASP.NET Core APIPAGEREF \_Toc515367997 \h43](#_Toc515367997)

[8.1.2 Bước 2 : Tạo model để quản lí dữ liệuPAGEREF \_Toc515367998 \h44](#_Toc515367998)

[8.1.3 Bước 3 : Xây dựng ESPServerContext để giao tiếp với Database bằng CodeFirstPAGEREF \_Toc515367999 \h50](#_Toc515367999)

[8.1.4 Bước 4 : Tạo dữ liệu đầu vàoPAGEREF \_Toc515368000 \h50](#_Toc515368000)

[8.1.5 Bước 5 : Đăng kí SeedData trong file Program.csPAGEREF \_Toc515368001 \h52](#_Toc515368001)

[8.1.6 Bước 6 : Xóa ValueController.cs và tạo một Controller mớiPAGEREF \_Toc515368002 \h52](#_Toc515368002)

[8.1.7 Bước 7 : Thay đổi nội dung file Properties => lauchSettings.jsonPAGEREF \_Toc515368003 \h56](#_Toc515368003)

[8.1.8 Bước 8 : Đăng kí ConnectionString trong file Startup.csPAGEREF \_Toc515368004 \h57](#_Toc515368004)

[8.1.9 Bước 9: Thay đổi nội dung hàm Configure trong file Startup.csPAGEREF \_Toc515368005 \h58](#_Toc515368005)

[8.1.10 Bước 10 : Install packet EntityFrameworkCorePAGEREF \_Toc515368006 \h58](#_Toc515368006)

[8.1.11 Bước 11 : Chạy ServerPAGEREF \_Toc515368007 \h59](#_Toc515368007)

[8.1.12 Bước 12 : Kiểm tra kết quả trên trình duyệtPAGEREF \_Toc515368008 \h60](#_Toc515368008)

[8.1.13 Bước 13 : Test các hàm trong file PinController.cs bằng PostmanPAGEREF \_Toc515368009 \h60](#_Toc515368009)

[8.2 NgrokPAGEREF \_Toc515368010 \h61](#_Toc515368010)

[8.2.1 Tại sao lại cần sử dụng Ngrok trong đồ án này ?PAGEREF \_Toc515368011 \h61](#_Toc515368011)

[8.2.2 Tổng quanPAGEREF \_Toc515368012 \h61](#_Toc515368012)

[8.2.3 Đặc điểm của NgrokPAGEREF \_Toc515368013 \h61](#_Toc515368013)

[8.2.4 Cài đặt NgrokPAGEREF \_Toc515368014 \h62](#_Toc515368014)

[8.2.5 Chạy NgrokPAGEREF \_Toc515368015 \h62](#_Toc515368015)

[8.2.6 Limited connectionPAGEREF \_Toc515368016 \h62](#_Toc515368016)

[8.2.7 Kiểm tra kết nối đến NgrokPAGEREF \_Toc515368017 \h63](#_Toc515368017)

[8.2.8 Kết luậnPAGEREF \_Toc515368018 \h63](#_Toc515368018)

[9 CHƯƠNG 9: CODE ARDUINOPAGEREF \_Toc515368019 \h64](#_Toc515368019)

[9.1 Sơ lượtPAGEREF \_Toc515368020 \h64](#_Toc515368020)

[9.2 Kết nối WifiPAGEREF \_Toc515368021 \h64](#_Toc515368021)

[9.2.1 Import thư việnPAGEREF \_Toc515368022 \h64](#_Toc515368022)

[9.2.2 Khai báo WifiPAGEREF \_Toc515368023 \h64](#_Toc515368023)

[9.2.3 Hàm setupWifi()PAGEREF \_Toc515368024 \h64](#_Toc515368024)

[9.2.4 Vị trí đặt hàm setupWifi()PAGEREF \_Toc515368025 \h65](#_Toc515368025)

[9.3 Kết nối Local ServerPAGEREF \_Toc515368026 \h65](#_Toc515368026)

[9.3.1 Import thư việnPAGEREF \_Toc515368027 \h65](#_Toc515368027)

[9.3.2 Khai báo kết nốiPAGEREF \_Toc515368028 \h65](#_Toc515368028)

[9.3.3 Get dữ liệu từ ServerPAGEREF \_Toc515368029 \h66](#_Toc515368029)

[9.3.4 Post dữ liệu lên ServerPAGEREF \_Toc515368030 \h66](#_Toc515368030)

[9.4 Xử lý dữ liệu nhận được thành JsonPAGEREF \_Toc515368031 \h66](#_Toc515368031)

[9.4.1 Hàm displayStringAsJson()PAGEREF \_Toc515368032 \h66](#_Toc515368032)

[9.4.2 Hàm displayStringAsJsonArray()PAGEREF \_Toc515368033 \h67](#_Toc515368033)

[9.4.3 Hàm convertStringToJson()PAGEREF \_Toc515368034 \h67](#_Toc515368034)

[9.4.4 Hàm getIndexAtJsonArray()PAGEREF \_Toc515368035 \h68](#_Toc515368035)

[9.5 Xử lý kết nối RFIDPAGEREF \_Toc515368036 \h68](#_Toc515368036)

[9.5.1 Import thư việnPAGEREF \_Toc515368037 \h68](#_Toc515368037)

[9.5.2 Hướng dẫn nối chân dành cho ESP8266 Wifi UNOPAGEREF \_Toc515368038 \h69](#_Toc515368038)

[9.5.3 Khai báo kết nốiPAGEREF \_Toc515368039 \h69](#_Toc515368039)

[9.5.4 Hàm setup\_rfid()PAGEREF \_Toc515368040 \h69](#_Toc515368040)

[9.5.5 Hàm dump\_byte\_array()PAGEREF \_Toc515368041 \h69](#_Toc515368041)

[9.5.6 Hàm startRFID()PAGEREF \_Toc515368042 \h70](#_Toc515368042)

[9.6 Xử lý kết nối OLEDPAGEREF \_Toc515368043 \h70](#_Toc515368043)

[9.6.1 Import thư việnPAGEREF \_Toc515368044 \h70](#_Toc515368044)

[9.6.2 Khai báo kết nốiPAGEREF \_Toc515368045 \h70](#_Toc515368045)

[9.6.3 Hàm setupOled()PAGEREF \_Toc515368046 \h70](#_Toc515368046)

[9.6.4 Code thay đổi screenPAGEREF \_Toc515368047 \h71](#_Toc515368047)

[9.7 FIX các lỗi đã gặpPAGEREF \_Toc515368048 \h71](#_Toc515368048)

[9.7.1 Lỗi ESP8266 không nhận WifiPAGEREF \_Toc515368049 \h71](#_Toc515368049)

[9.7.2 Lỗi ESP8266 không kết nối được đến localhostPAGEREF \_Toc515368050 \h71](#_Toc515368050)

[9.7.3 Lỗi “Firmware Unknown” khi ESP8266 kết nối RFIDPAGEREF \_Toc515368051 \h71](#_Toc515368051)

[10 CHƯƠNG 10: CODE RASPBERRY PIPAGEREF \_Toc515368052 \h72](#_Toc515368052)

[10.1 Cài đặt hệ điều hành Raspbian trên Linux:PAGEREF \_Toc515368053 \hError: Reference source not found](#_Toc515368053)

[10.1.1 Cài đặtPAGEREF \_Toc515368054 \hError: Reference source not found](#_Toc515368054)

[10.1.2 Kết nối wifi và SSH:PAGEREF \_Toc515368055 \hError: Reference source not found](#_Toc515368055)

[10.2 OpenCVPAGEREF \_Toc515368056 \h72](#_Toc515368056)

[10.2.1 Giới thiệuPAGEREF \_Toc515368057 \h72](#_Toc515368057)

[10.2.2 Mục đíchPAGEREF \_Toc515368058 \h72](#_Toc515368058)

[10.2.3 Cài đặt OpenCV:PAGEREF \_Toc515368059 \hError: Reference source not found](#_Toc515368059)

[10.2.4 Ứng dụngPAGEREF \_Toc515368060 \h72](#_Toc515368060)

[10.3 Kiến thức cơ bản về xử lý ảnh:PAGEREF \_Toc515368061 \h72](#_Toc515368061)

[10.3.1 Ảnh kỹ thuật số (Digital image):PAGEREF \_Toc515368062 \h72](#_Toc515368062)

[10.3.2 Độ phân giải ảnh (Resolution):PAGEREF \_Toc515368063 \h72](#_Toc515368063)

[10.3.3 Điểm ảnh (Pixel):PAGEREF \_Toc515368064 \h73](#_Toc515368064)

[10.3.4 Mức xám của ảnh (Grayscale):PAGEREF \_Toc515368065 \h73](#_Toc515368065)

[10.3.5 Ảnh màu:PAGEREF \_Toc515368066 \h73](#_Toc515368066)

[10.4 Code face-detection sử dụng Raspberry Pi, OpenCV:PAGEREF \_Toc515368067 \hError: Reference source not found](#_Toc515368067)

[10.5 FIX các lỗi đã gặpPAGEREF \_Toc515368068 \h73](#_Toc515368068)

[11 CHƯƠNG 11: CLOUDPAGEREF \_Toc515368069 \h78](#_Toc515368069)

[11.1 Tổng quanPAGEREF \_Toc515368070 \hError: Reference source not found](#_Toc515368070)

[11.2 Một số hệ điều hành có thể triển khai Web ServerPAGEREF \_Toc515368071 \h80](#_Toc515368071)

[11.2.1 Windows ServerPAGEREF \_Toc515368072 \hError: Reference source not found](#_Toc515368072)

[11.2.2 LinuxPAGEREF \_Toc515368073 \hError: Reference source not found](#_Toc515368073)

[12 CHƯƠNG 12 : TRIỂN KHAI TRÊN CLOUD AZUREPAGEREF \_Toc515368074 \hError: Reference source not found](#_Toc515368074)

[12.1 Đăng nhập vào Azure với tài khoản trường (tài khoản ms.uit.edu.vn)PAGEREF \_Toc515368075 \h82](#_Toc515368075)

**DANH SÁCH HÌNH VẼ**

[Hình 3.1: DHT11 25](#_Toc515567333)

[Hình 3.2: Cảm biến lửa 26](#_Toc515567334)

[Hình 3.3: Cảm biến chuyển động 26](#_Toc515567335)

[Hình 3.4: Cảm biến khoảng cách 27](#_Toc515567336)

[Hình 3.5: Động cơ servo 27](#_Toc515567337)

[Hình 3.6: RFID 28](#_Toc515567338)

[Hình 3.7: Raspberry Pi 29](#_Toc515567339)

[Hình 3.8: Module Raspberry Pi Camera 30](#_Toc515567340)

[Hình 4.9: Nguyên lý hoạt động Wifi 32](#_Toc515567341)

[Hình 6.10: Kiến trúc của Entity Framework 40](#_Toc515567342)

[Hình 6.11: Mối quan hệ và thức tự tạo ra giữa các thành phần dữ liệu trong EF 42](#_Toc515567343)

[Hình 7.12: Cấu trúc vật lý của SQL Server 43](#_Toc515567344)

[Hình 7.13: Các thành phần cửa SQL Server 44](#_Toc515567345)

[Hình 8.14: Tạo một project (1) 48](#_Toc515567346)

[Hình 8.15: Tạo một project (2) 49](#_Toc515567347)

[Hình 8.16: Kết quả sau khi tạo một project 49](#_Toc515567348)

[Hình 8.17: Tạo PinModel 50](#_Toc515567349)

[Hình 8.18: Nội dung file Pin..cs 50](#_Toc515567350)

[Hình 8.19: Nội dung file IPinRepo.cs 50](#_Toc515567351)

[Hình 8.20: Nội dung file PinRepo.cs (1) 51](#_Toc515567352)

[Hình 8.21: Nội dung file PinRepo.cs (2) 52](#_Toc515567353)

[Hình 8.22: Nội dung file PinRepo.cs (3) 52](#_Toc515567354)

[Hình 8.23: Tạo UserModel 53](#_Toc515567355)

[Hình 8.24: Nội dung file User.cs 53](#_Toc515567356)

[Hình 8.25: Nội dung file IUserRepo.cs 53](#_Toc515567357)

[Hình 8.26: Nội dung file UserRepo.cs (1) 54](#_Toc515567358)

[Hình 8.27: Nội dung file UserRepo.cs (2) 54](#_Toc515567359)

[Hình 8.28: Nội dung file UserRepo.cs (3) 55](#_Toc515567360)

[Hình 8.29: Tạo file ESPServerContext.cs 55](#_Toc515567361)

[Hình 8.30: Nội dung file ESPServerContext.cs 55](#_Toc515567362)

[Hình 8.31: Tạo file SeedData.cs 55](#_Toc515567363)

[Hình 8.32: Nội dung file SeedData.cs (1) 56](#_Toc515567364)

[Hình 8.33: Nội dung file SeedData.cs (2) 56](#_Toc515567365)

[Hình 8.34: Đăng kí SeedData trong Program.cs 57](#_Toc515567366)

[Hình 8.35: Tạo Controller 57](#_Toc515567367)

[Hình 8.36: Nội dung file PinController.cs (1) 58](#_Toc515567368)

[Hình 8.37: Nội dung file PinController.cs (2) 58](#_Toc515567369)

[Hình 8.38: Nội dung file PinController.cs (3) 59](#_Toc515567370)

[Hình 8.39: Nội dung file PinController.cs (4) 59](#_Toc515567371)

[Hình 8.40: Nội dung file UserController.cs (1) 60](#_Toc515567372)

[Hình 8.41: Nội dung file UserController.cs (2) 61](#_Toc515567373)

[Hình 8.42: Nội dung file UserController.cs (3) 61](#_Toc515567374)

[Hình 8.43: Nội dung file lauchSetting.json 62](#_Toc515567375)

[Hình 8.44:Đăng kí đường dẫn đến thư mục sqlite 62](#_Toc515567376)

[Hình 8.45: Nội dung hàm Configure trong file Startup.cs 63](#_Toc515567377)

[Hình 8.46: Install Microsoft.EntityFrameworkCore.Sqlite 63](#_Toc515567378)

[Hình 8.47: Thư mục Migrations 64](#_Toc515567379)

[Hình 8.48: Kêt quả tạo Database Sqlite 64](#_Toc515567380)

[Hình 8.49: Run project bằng Command line 64](#_Toc515567381)

[Hình 8.50: Kết quả sai khi chạy Server 65](#_Toc515567382)

[Hình 8.51: Action Get() 65](#_Toc515567383)

[Hình 8.52: Action Get(int pin) 65](#_Toc515567384)

[Hình 8.53: Action Post(Data dt) 66](#_Toc515567385)

[Hình 8.54: Chạy Ngrok 67](#_Toc515567386)

[Hình 8.55: Kết quả sau khi chạy Ngrok 67](#_Toc515567387)

[Hình 8.56: Kết quả kiểm tra kết nối Ngrok trên điện thoại 68](#_Toc515567388)

[Hình 9.57: Hàm kết nối Wifi 70](#_Toc515567389)

[Hình 9.58: Vị trí đặt hàm setupWifi 70](#_Toc515567390)

[Hình 9.59: Hàm getHttpResponse() 71](#_Toc515567391)

[Hình 9.60: Hàm Post() 72](#_Toc515567392)

[Hình 9.61: Hàm displayStringAsJson() 72](#_Toc515567393)

[Hình 9.62: Hàm displayStringAsJsonArray() 73](#_Toc515567394)

[Hình 9.63: Hàm convertStringToJson() 73](#_Toc515567395)

[Hình 9.64: Hàm getIndexAtJsonArray() 74](#_Toc515567396)

[Hình 9.65: Hàm setup\_rfid() 75](#_Toc515567397)

[Hình 9.66: Hàm dump\_byte\_array() 75](#_Toc515567398)

[Hình 9.67: Hàm startRFID() 75](#_Toc515567399)

[Hình 9.68: Hàm setupOled() 76](#_Toc515567400)

[Hình 12.69: Trang chủ Azure for Student 88](#_Toc515567401)

[Hình 12.70: Đăng nhập bằng tài khoản ms 89](#_Toc515567402)

[Hình 12.71: Trang chủ Azure Portal 89](#_Toc515567403)

[Hình 12.72: Thêm resource mới 90](#_Toc515567404)

[Hình 12.73: Chọn WebApp 90](#_Toc515567405)

[Hình 12.74: Tạo WebApp 91](#_Toc515567406)

[Hình 12.75: Kết quả sau khi tạo WebApp 91](#_Toc515567407)

[Hình 12.76: Giao diện WebApp 92](#_Toc515567408)

[Hình 12.77: Deploy project lên Azure 92](#_Toc515567409)

[Hình 12.78: Chọn WebApp có sẵn để publish 93](#_Toc515567410)

[Hình 12.79: Chọn WebApp đã tạo ở trên 93](#_Toc515567411)

[Hình 12.80: Kết quả sau khi deploy lên Azure 94](#_Toc515567412)

**DANH SÁCH BẢNG**

[Bảng 2.1: Lịch sử ra đời của ASP.NET Framework 13](#_Toc512353541)

[Bảng 2.2: Lịch sử ra đời của ASP.NET Core 13](#_Toc512353542)

[Bảng 2.3: So sánh ASP.NET Framework và ASP.NET Core 14](#_Toc512353543)

[Bảng 2.4: So sánh ASP.NET Core và NodeJS 16](#_Toc512353544)

# CHƯƠNG 1: GIỚI THIỆU

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.

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.

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.

# CHƯƠNG 2: MỘT SỐ PLATFORM PHÁT TRIỂN ỨNG DỤNG WEB/API

## ASP.NET

### Tổng quan

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.

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

Các ứng dụng ASP.NET có thể được viết bằng bất kỳ ngôn ngữ nào sao đây:

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

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

### ASP.NET Framework

#### Tổng quan

.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).

.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.

.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.

#### 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%23.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

### ASP.NET Core

#### Tổng quan

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.

ASP.NET Core được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (cloud) hoặc chạy on-promise. 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.

#### 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

#### 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

### 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

## Node.JS

### Tổng quan

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.

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...

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.

### Lịch sử phát triển

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

### Một số đặc điểm của NodeJS

#### Không đồng bộ và Phát sinh sự kiện (Event Driven):

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.

#### Chạy rất nhanh

Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.

#### Không đệm

Ứng dụng Node.js không lưu trữ các dữ liệu buffer.

#### Có giấy phép

Node.js được phát hành dựa vào MIT License.

#### Các tiến trình đơn giản nhưng hiệu năng cao

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.

## 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 |

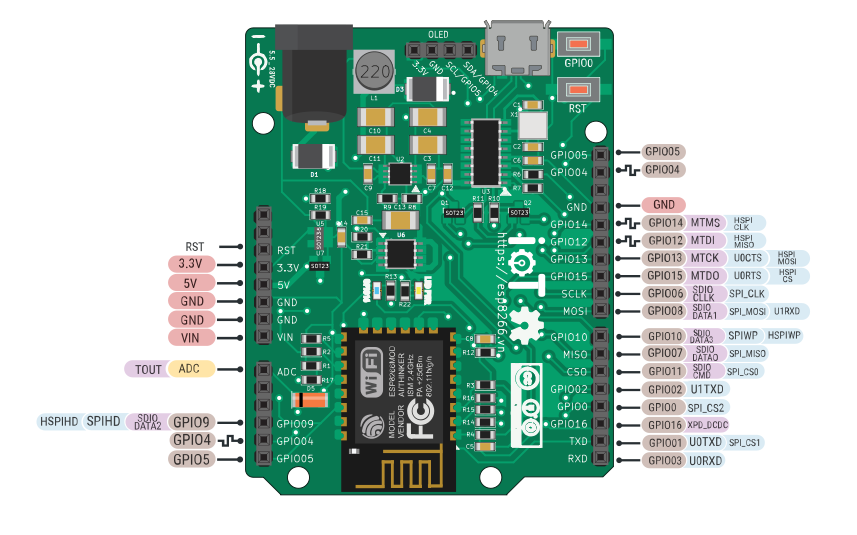
Bảng 2.4: So sánh ASP.NET Core và NodeJS

# CHƯƠNG 3: MỘT SỐ MODULE ĐƯỢC SỬ DỤNG TRONG MÔ HÌNH

## Arduino ESP8266 Wifi Uno

### Tổng quan

ESP8266 Arduino Uno 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

### 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 .

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

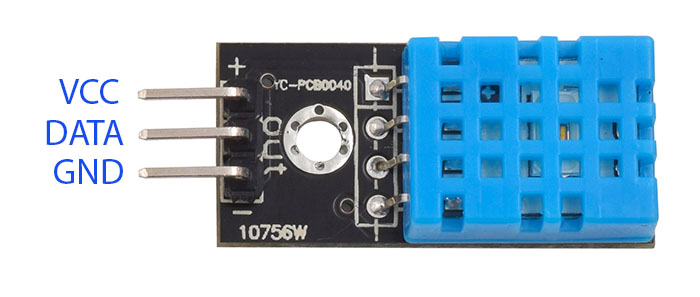
## Một số module và sensor hỗ trợ Arduino ESP8266

### OLED 128x64 0.96 inch – 12C



Hình 3.2: OLED

### Cảm biến nhiệt độ, độ ẩm DHT11



Hình 3.1: DHT11

### Cảm biến phát hiện lửa



Hình 3.2: Cảm biến lửa

### Cảm biến chuyển động HC-SR501



Hình 3.3: Cảm biến chuyển động

### Cảm biến khoảng cách HC-SR04



Hình 3.4: Cảm biến khoảng cách

### Động cơ Servo



Hình 3.5: Động cơ servo

### RFID RC522



Hình 3.6: RFID

## Raspberry Pi 3

### Tổng quan

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 3.7: Raspberry Pi

### Phần cứng

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:

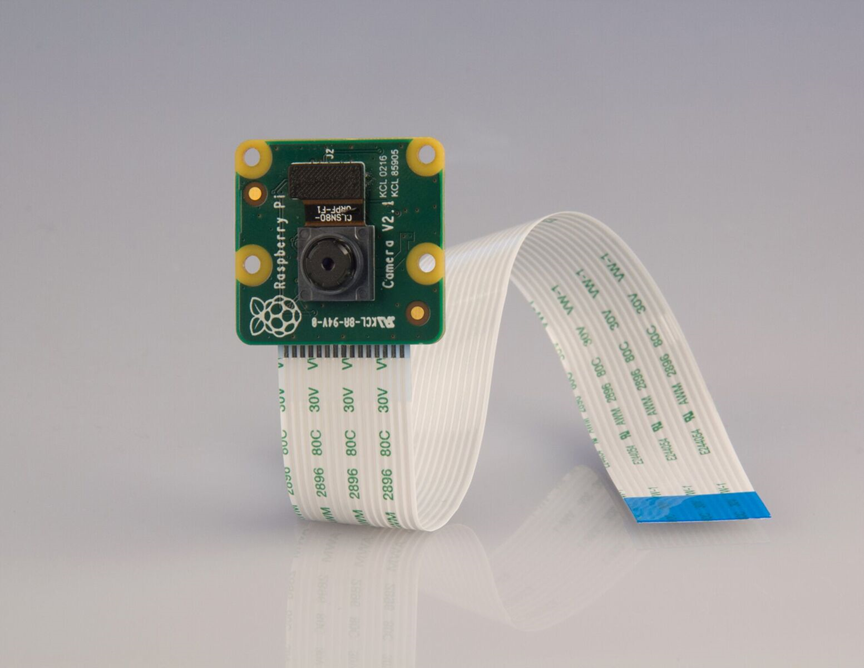
* CPU.
* Khe cắm thẻ micro SD.
* Cổn USB.
* Cổng Ethernet.
* Cổng HDMI.
* Ngõ ra Audio-Video.
* Cổng cấp nguồn Micro USB.
* Cổng DSI (Display Serial Interface).
* Cổng CSI (Camera Serial Interface).
* GPIO (General Purpose Input and Output).

### Hệ điều hành:

Raspberry Pi có thể chạy được trên nhiều OS linux khác nhau. Một số platform chạy trên Raspberry Pi như Raspbian, Pidora, openSUSE, OpenWRT, …

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.

### Module sử dụng: Raspberry Pi Camera Module



Hình 3.8: 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.

# CHƯƠNG 4: MỘT SỐ CÁCH THỨC TRUYỀN DỮ LIỆU

## Wifi

### Tổng quan

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.

### Các chuẩn Wifi:

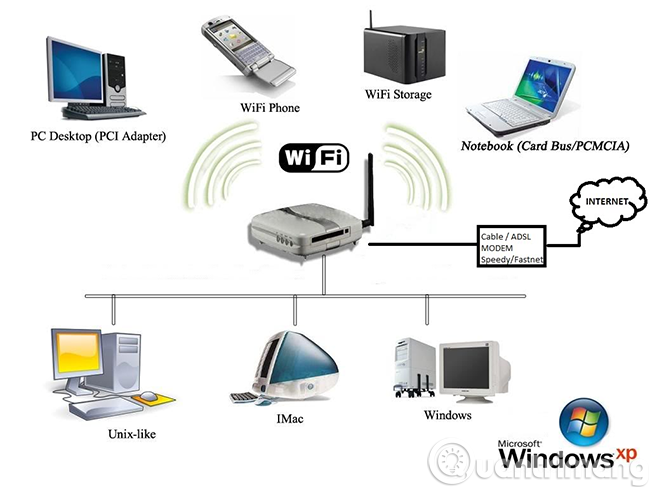
Sóng wifi ngày nay được sử dụng rộng rãi phục vụ đời sống con người, đặc điểm của sóng wifi là thu phát ở tầng số từ 2.4GHz đến 5GHz cao hơn so với sóng vô tuyến truyền hình, sóng điện thoại và radio nên khá an toàn trong vấn đề bảo toàn thông tin khi truyền và nhận dữ liệu. Các chuẩn wifi hiện nay:

* 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.

### Nguyên lý hoạt động:

Mạng Internet sẽ được các ISP ( nhà cung cấp dịch vụ Internet – Internet Services Provider) truyền đến bộ giải mã tín hiệu số (modem), thông qua định tuyến (router) hay là bộ phát wifi chuyển tín hiệu hữu tuyến thành kết nối vô tuyến và đưa đến các thiết bị di động không dây thông qua chuẩn kết nối Wifi.

Các thiết bị không dây tiếp nhận sóng wifi thông qua một thiết bị chuyển đổi tín hiệu gọi là Adapter (card Wifi) được cài đặt trực tiếp trên thiết bị. Tín hiệu vô tuyến sẽ được giải mã ngay trên thiết bị, từ đây người dùng có thể trực tiếp truy cập Internet như bình thường.



Hình 4.9: Nguyên lý hoạt động Wifi

### Các phương pháp bảo mật Wifi

Ngoài tính năng bảo mật phổ biến là AES (Advanced Encryption Standard) là chuẩn mã hóa tín hiệu nâng cao được quy chuẩn, Wifi còn được bảo mật bởi các mật khẩu WPA hoặc WPA2 có nhiệm vụ bảo đảm an ninh truy cập cho các hệ thống Wifi. WPA2 được phát triển để thay thế cho hệ thống WEP bị lỗi thời và hay bị giải mã. Với tính năng này, tuy không phải thực sự an toàn đối với mọi hình thức tấn công nhưng người dùng cũng đã có thể yên tâm sử dụng do việc giải mã các chuẩn bảo mật này cũng vô cùng phức tạp.

### Ưu điểm của mạng Wifi

Kết nối Wifi tiện dụng, đơn giản gọi nhẹ so với kết nối trực tiếp bằng cable truyền thống qua cổng RJ45. Người dùng có thể truy cập ở bất cứ vị trí nào trong vùng bán kính phủ sóng mà tại đó Router Wifi làm trung tâm.

Mạng sử dụng Wifi dễ sửa đổi và nâng cấp. Người dùng có thể tăng băng thông truy cập, tăng số lượng người sử dụng mà không cần nâng cấp thêm router hay dây cắm như các kết nối vật lý.

Tính thuận tiện: người dùng có thể duy trì kết nối kể cả khi đang di chuyển.

Tính bảo mật của mạng Wifi tương đối cao.

### Nhược điểm của mạng Wifi

Phạm vi kết nối của mạng Wifi tới thiết bị có giới hạn, đi càng xa router kết nối càng yếu dần đi.

Vấn đề băng thông cũng là một nhược điểm của Wifi, càng nhiều người kết nối vào mạng thì tốc độ truy cập giảm rõ rệt.

## Bluetooth

### Tổng quan

Bluetooth được phát triển đầu tiên bởi Ericsson (hiện nay là Sony Ericsson và Ericsson Mobile Platforms), và sau đó được chuẩn hóa bởi Bluetooth Special Interest Group (SIG). Chuẩn này được phát hành vào ngày 20 tháng 5 năm 1999. Bluetooth có chuẩn IEEE 802.15.1.

Bluetooth là công nghệ không dây cho phép các thiết bị điện, điện tử giao tiếp với nhau trong khoảng cách ngắn bằng sóng vô tuyến qua băng tần chung ISM (Industrial, Scientific, Medical) với băng tần 2.4 GHz. Tốc độ truyền dữ liệu có thể đạt tới mức tối đa 1 Mbps. Bluetooth hỗ trợ tốc độ truyền tải dữ liệu lên tới 720 Kbps trong phạm vi 10-100m. Kết nối Bluetooth là vô hướng.

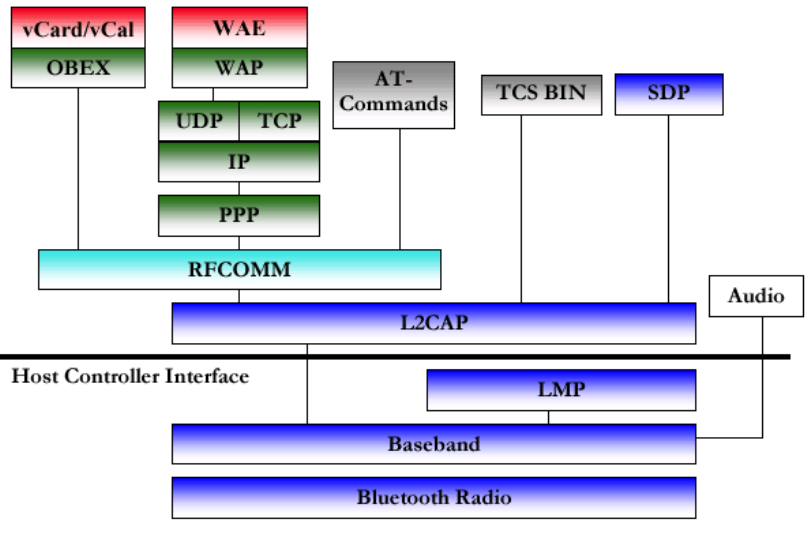
### Các đặc điểm của Bluetooth

Tiêu thụ điện năng thấp, cho phép ứng dụng được trong nhiều loại thiết bị, bao gồm cả các thiết bị cầm tay và thiết bị di động.

Dễ dàng trong việc triển khai ứng dụng: Blluetooth kết nối ứng dụng này với một ứng dụng khác thông qua các chuẩn “Bluetooth profiles”, do đó có thể độc lập về phần cứng cũng như hệ điều hành sử dụng.

Tính tương thích cao, được nhiều nhà sản xuất phần cứng cũng như phần mềm hỗ trợ.

### Các tầng giao thức Bluetooth

Bluetooth Radio: Định nghĩa các yêu cầu của các thiết bị truyền bluetooth hoạt đọng như dải tần số, thông số nhảy tần, …

Hình 4.3: Mô hình cơ bản của giao thức Bluetooth

Baseband: Định dạng địa chỉ, khung của gói tin, thời gian và các thuật toán điều khiển điện cần để thiết lập kết nối giữa các thiết bị bluetooth.

Link Manager protocol (LMP): Có trách nhiệm thiết lập và duy trì các thiết bị bluetooth. Bao gồm thông số xác thực và mã hóa.

Host Controller Interface (HCI): Sử dụng để giao tiếp giữa các module (phần cứng) với hệ thống máy chủ (phần mềm) thông qua các cơ chế nhập xuất tiêu chuẩn.

Logical link control and adaptation protocol (L2CAP): Có trách nhiệm phân chia các gói tin và tái lắp ráp các gói tin phân mảnh. Ngoài ra nó còn có trách nhiệm ghép kênh giữa các gói tin Bluetooth từ các ứng dụng khác nhau.

Service discovery protocol (SDP): Thông tin thiết bị, dịch vụ và các đặc tính của dịch vụ có thể truye cập để thiết lập kết nối giữa các thiết bị Bluetooth.

RFCOMM: Giao thức “thay thế cáp” mô phỏng các tín hiệu điều khiển EIA-232 và tín hiệu dữ liệu qua giao thức cơ sở Bluetooth.

TCS BIN (Telephony control specification binary): Là một giao thức định hướng bit xác định tín hiệu điều khiển cuộc gọi để thiết lập các cuộc nói chuyện.

### Ưu điểm của Bluetooth

* Giá thành rẻ.
* Tầm giao tiếp rộng.
* Tiêu thụ ít điện năng.
* Bảo mật tốt.
* Thời gian kết nối nhanh.

### Nhược điểm của Bluetooth

* Phạm vi kết nối ngắn.
* Dễ bị nhiễu.
* Bắt sóng kém khi có vật cản.

## Z-Wave

### Tổng quan

Z-Wave là một công nghệ truyền thông RF công suất thấp được thiết kế chủ yếu cho tự động hóa gia đình như bộ điều khiển đèn và cảm biến trong một số thiết bị khác.

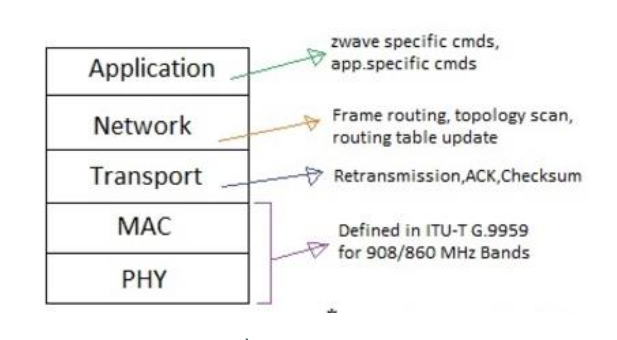
### Đặc điểm của Z-Wave

Z-Wave được tối ưu hóa cho truyền thông đáng tin cậy và độ trễ thấp của các gói dữ liệu nhỏ với tốc độ dữ liệu lên đến 100 kbit/s.

Z-Wave hoạt động ở băng tần 1GHz và không bị nhiễu bởi Wifi và các công nghệ không dây khác trong khoảng 2.4 GHz.

Z-Wave hỗ trợ mạng mesh mà không cần nút điều phối và có thể mở rộng, cho phép điều khiển tới 232 thiết bị.

### Các tầng giao thức Z-Wave

Physical layer (PHY): Đây là tầng có nhiệm vụ điều chế và mã hóa cũng như chèn những mẫu đã biết (preamble) được sử dụng để đồng bộ hóa tại máy thu. Ngoài ra nó còn thực hiện việc phân kênh RF.

Hình 4.4: Các tầng giao thức Z-Wave

MAC Layer (MAC): Kiểm soát truy cập trên các node slave. Nó theo dõi hoạt động của mạng dựa trên HomeID, NodeID và các tham số khác nằm trong frame.

Transport layer: Có nhiệm vụ truyền lại, nhận diện gói tin, đánh thức các node và xác thực nguồn gốc gói tin.

Network layer: Kiểm soát việc định tuyến đường đi các frame từ node này tới node kia.

Application layer: có trách nhiệm giải mã và thực thi câu lệnh.

# CHƯƠNG 5: KỊCH BẢN THỰC HIỆN

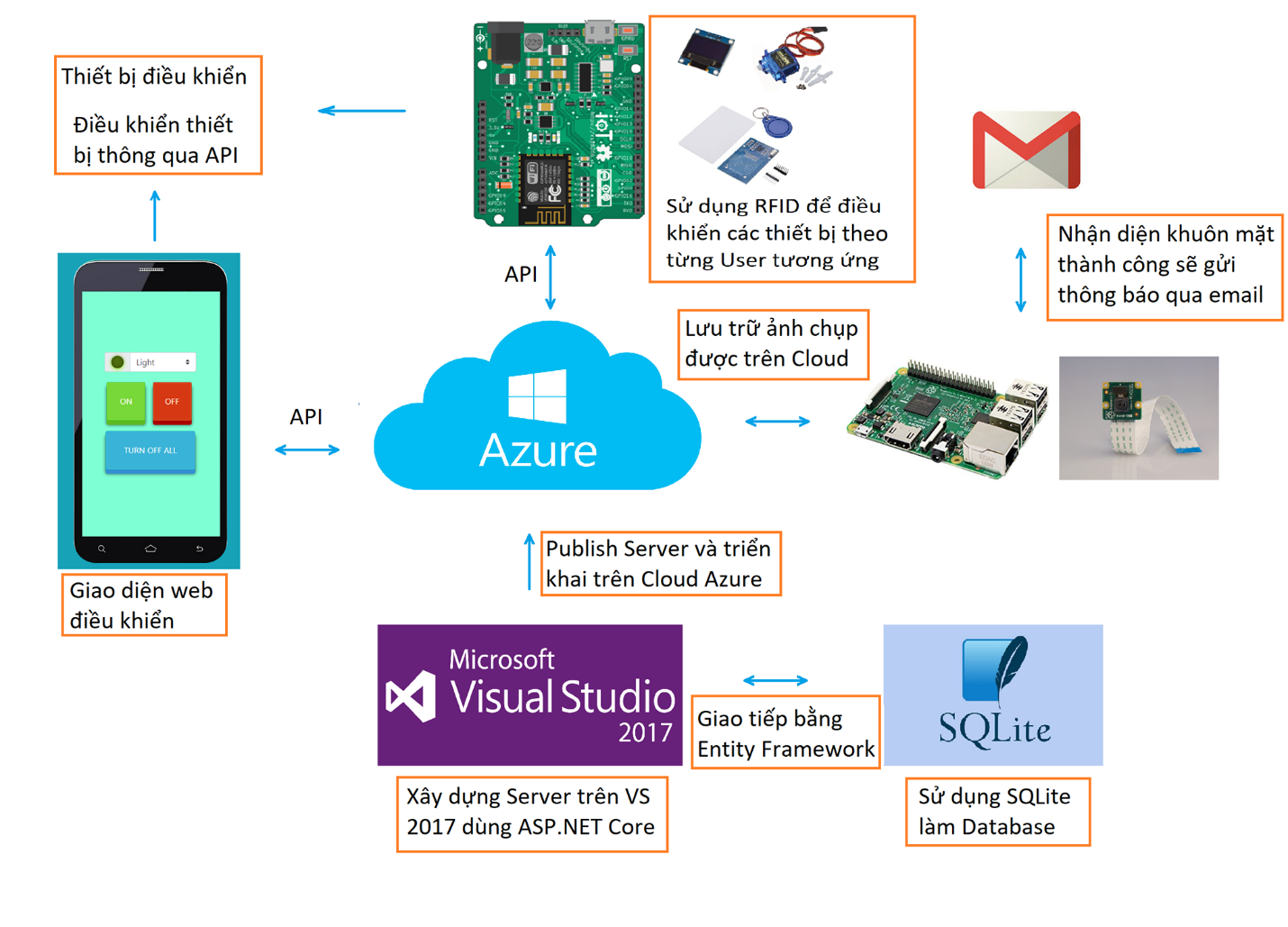
Xây dựng và phát triển giải pháp xuất nhập cho nhà thông minh sử dụng điện toán đám mây. Kịch bản được xây dựng như sau:

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.

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

Khi người sử dụng bật chế độ security, hệ thống camera sẽ bắt đầu hoạt động. Hệ thống camera sẽ quay video liên tục. Khi camera nhận diện được có người xuất hiện, sẽ có một email gửi đến cho chủ nhà hình ảnh có người xuất hiện, đồng thời hình ảnh này được gửi lên cloud.



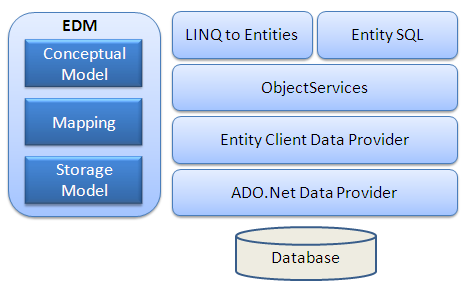
Hình 5.5: Sơ đồ mô hình thực hiện

# CHƯƠNG 6 : ENTITY FRAMEWORK

## Tổng quan

Entity Framework (EF) là một framework ánh xạ quan hệ đối tượng (ORM) dành cho ADO.NET, là một phần của .NET Framework. EF cho phép các nhà phát triển Web tương tác vói dữ liệu quan hệ theo phương pháp hướng đối tượng đặc trưng. EF được Microsoft hỗ trợ phát triển lâu dài và bền vững, vì vậy EF là một framework mạnh nhất hiện nay để phát triển ứng dụng Web với sự hỗ trợ đông đảo của các nhà phát triển Web.

## Kiến trúc của Entity Framework



Hình 6.10: Kiến trúc của Entity Framework

* EDM (Entity Data Model): bao gồm ba phần chính Conceptual model, Mapping và Storage model.
* Conceptual Model: là các lớp và mối quan hệ tương ứng với cơ sở dữ liệu.
* Storage Model: mô hình thiết kế cơ sở dữ liệu bao gồm Table, View, Store procedure, Rekationship, Key,…
* Mapping: gồm thông tin các khái niệm mô hình do develpoper ánh xạ tới mô hình lưu trữ hay cơ sở dữ liệu.
* LINQ to Entities: là ngôn ngữ truy vấn với mô hình đối tượng Object model.
* Entity SQL: là ngôn ngữ truy vấn giống như LINQ to Entity nhưng phức tạp hơn.
* Object Service: phục vụ cho việc truy cập. Object Service cung cấp đầy đủ dịch vụ để quá trình chuyển đổi dữ liệu từ thực thể đến cấu trúc đối tượng dễ dàng hơn.
* Entity Client Data Provider: Giao tiếp với ADO.NET data provider hoặc lấy dữ liệu từ cơ sở dữ liệu.
* ADO.Net Data Provider: lớp này giao tiếp với cơ sở dữ liệu theo chuẩn ADO.NET.

## Đặc điểm của Entity Framework

Entity Framework là một framework truy xuất dữ liệu.

Hỗ trợ các ứng dụng database.

Khả năng độc lập với CSDL

Được xây dựng dựa trên nền tảng ADO.NET

## Lợi ích khi sử dụng Entity Framework

Rút ngắn thời gian phát triển ứng dụng do EF đã hỗ trợ sẵn các tính năng cơ bản để truy xuất dữ liệu.

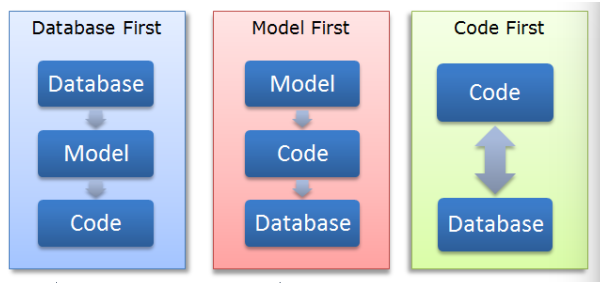
Lập trình viên được làm việc với ứng dụng hướng đối tượng.

Không phụ thuộc quá nhiều vào mô hình lưu trữ.

Công việc truy xuất dữ liệu trở nên dễ dàng hơn.

## Thiết kế Entity Data Model:

Là thành phần trung gian giữa ứng dụng và database, Entity Data Model (EDM) giúp tạo liên kết và mô tả giữa nguồn dữ liệu vật lý và các đối tượng bussiness trong ứng dụng. Dựa vào sự lựa chọn thành phần nào sẽ được tạo ra trước, một ứng dụng EF có thể sử dụng một trong ba cách tiếp cận: Database First, Model First và Code First.



Hình 6.11: Mối quan hệ và thức tự tạo ra giữa các thành phần dữ liệu trong EF

### Database First:

Đây là cách thông thường và được hỗ trợ từ phiên bản EF 1 trong visual Studio 2008.

EF sẽ tự động tạo ra data model và các class từ database có sẵn thông qua công cụ Entity Data Model Wizard. Có thể thay đổi data model và cập nhật lại vào database. Đây là cách tiếp cận phổ biến vì cách thực hiện đơn giản, nhanh chóng.

### Model First:

Hướng tiếp cận này được bổ sung từ phiên bản EF 4 trong Visual Studio 2010.

Có thể tạo một data model rỗng bằng công cụ Entity Data Model Designer, khi đó object layer cũng tự động được sinh ra. Sau khi hoàn thành việc thiết kế, có thể sử dụng chức năng Generate Database from Model để tạo ra các mã DDL ( data definition language) dựa trên mã SSDL (Store Schema Definition Language).

### Code First:

Hướng tiếp cận này bỏ qua phần Model, có thể thiết kế các class tương ứng với các table của database cần làm việc hoặc muốn tạo ra. Với hướng tiếp cận này, sử dụng DbContext thay vì ObjectContext để thao tác với database. Việc sử dụng Code First đòi hỏi thực hiện các công việc một cách thủ công nhưng bù lại nó linh hoạt và được sử dụng rất phổ biến.

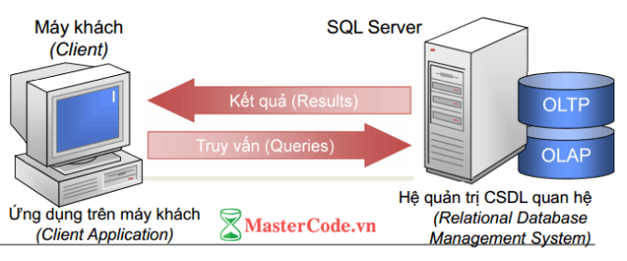
# CHƯƠNG 7 : DATABASE

## SQLServer

### Khái niệm SQL Server

SQL Server là một hệ quản trị dữ liệu quan hệ sử dụng câu lệnh SQL để trao đổi dữ liệu giữa máy cài SQL Server và máy Client. Một Relational Database Management System - RDBMS gồm có: databases, datase engine và các chương trình ứng dụng dùng để quản lý các bộ phận trong RDBMS và những dự liệu khác.

### Khái niệm về cấu trúc vật lý



Hình 7.12: Cấu trúc vật lý của SQL Server

Máy chủ (Server)

Máy trạm (Client): là các máy tính được phép truy xuất các tài nguyên đã được chia sẻ trên mạng.

Dây cáp mạng (Cable hoặc Media) là một hệ thống dây cáp kết nối vật lý các máy tính, máy in lại với nhau.

Dữ liệu chung (Shared data): là các tập tin, thư mục mà người sử dụng trong hệ thống mạng có thể truy xuất vào máy chủ từ các máy trạm.

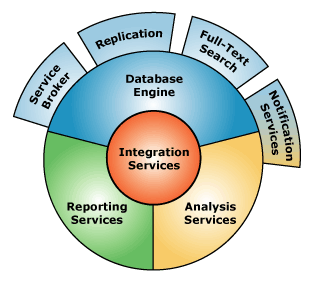
### Lịch sử ra đời SQL Server

Năm 1989, phiên bản đầu tiên của SQL Server 1.0 ra đời được dùng cho các hệ điều hành 16 bit và được phát triển cho tới ngày nay.

Cho tới khi SQL Server ra phiên bản 6.5 thì được thì trường chấp nhận rộng rãi. Một đột phá cải tiến cho SQL Server 7.0 khi được Microsoft viết lại một engine hoàn toàn mới. Đến khi SQL Server từ phiên bản 7.0 cải tiến lên 8.0 chủ yếu phát triển tính năng thiết kế web.

Cho đến ngày nay thì phiên bản mới nhất là SQL Server 2016 hỗ trợ bộ vi xử lý 64 bit ra đời vào ngày 1 tháng 6 năm 2016.

### Các thành phần cơ bản trong SQL Server



Hình 7.13: Các thành phần cửa SQL Server

Các thành phần cơ bản trong SQL Server gồm có: Reporting Services, Database Engine, Integration Services, Notification Services, Full Text Search Service,…

#### Database Engine

Đây là một engine có khả năng chứa dữ liệu ở các quy mô dưới dạng support và table. Ngoài ra nó còn có khả năng tự điều chỉnh, ví dụ trả lại tài nguyên cho hệ điều hành khi một user log off và sử dụng thêm tài nguyên của máy khi cần.

#### Integration Services

Là tập hợp các đối tượng lập trình và các công cụ đồ họa cho việc sao chép, di chuyển và chuyển đổi dữ liệu.

#### Analysis Services

Đây là một dịch vụ phân tích dữ liệu của Microsoft. Dữ liệu khi được lưu trữ vào database mà không lấy được thông tin thì không có ý nghĩa gì. Công cụ này ra đợi giúp việc phân tích dữ liệu một cách hiệu quả và dễ dàng bằng cách dùng kỹ thuật khai thác dữ liệu - datamining và khái niệm hình khối nhiều chiều - multi dimendion cubes.

#### Notification Services

Dịch vụ này là nền tảng cho sự phát triển và triển khai các ứng dụng soạn và gửi thông báo. Ngoài ra, dịch vụ này còn có chức năng gửi thông báo theo dịch thời đến hàng ngàn người đăng ký sử dụng trên nhiều thiết bị khác nhau.

#### Reporting Services

Là công cụ tạo, quản lý và triển khai báo cáo bao gồm: server và client. Ngoài ra nó còn là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.

#### Full Text Search Services

Là thành phần đặc biệt trong việc truy vấn và đánh chỉ mục dữ liệu văn bản không cấu trúc được lưu trữ trong các cơ sở dữ liệu SQL Server.

#### Services Broke

Là một môi trường lập trình cho việc tạo ra các ứng dụng trong việc nhảy qua các Instance.

### Vì sao phát triển ứng dụng server-client?

#### Giảm chi phí

Giảm chi phí bản quyền.

Giảm chi phí nâng cấp, bảo trì, quản lý.

#### Tốc độ nhanh

Các xử lý phức tạp có thể thực hiện tại server.

#### Tính tương thích cao

Nhiều công cụ lập trình được hỗ trợ bởi phần mềm làm việc trên máy chủ.

## SQlite

### Khái niệm SQLite

SQLite là một thư viện phần mềm mà triển khai một SQL Database Engine truyền thống, không cần server, không cần cấu hình và nhỏ gọn.

SQLite là một trong những Database Engine phát triển nhanh nhất, nhưng sự phổ biến của nó là về mặt phổ biến, không phải về mặt kích cỡ. SQLite là mã nguồn mở.

### Đặc điểm của SQLite

Đặc điểm của SQLite là gọn, nhẹ, đơn giản. Chương trình gồm một file duy nhất vỏn vẹn chưa đến 400kB, không cần cài đặt, không cần cấu hình hay khởi động mà có thể sử dụng ngay. Dữ liệu Database cũng được lưu ở một file duy nhất. Không có khái niệm user, password hay quyền hạn trong SQLite Database.

SQLite không thích hợp với những hệ thống lớn nhưng ở quy mô vừa tầm thì SQLite phát huy uy lực và không hề yếu kém về mặt chức năng hay tốc độ. Với các đặc điểm trên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm … và là sự lưa chọn phù hợp cho những người bắt đầu học Database.

SQLite Engine không là một Standalone Process giống như các cơ sở dữ liệu khác, bạn có thể liên kết nó một cách tĩnh hoặc một cách động tùy theo yêu cầu với ứng dụng của bạn. SQLite truy cập các file lưu giữ của nó một cách trực tiếp.

### Lịch sử của SQLite

Năm 2000: D.Richard Hipp đã thiết kế SQLite với mục đích không cần quản trị để điều hành một chương trình.

Vào tháng 8, SQLite 1.0 được công bố với GNU Database Manager.

Năm 2011: Hipp bổ sung UNQI Interface cho SQLite DB và để phát triển UNQLite (là một Document Oriented Database).

### Vì sao nên sử dụng SQLite?

SQLite không yêu cầu một tiến trình Server riêng rẽ để hoạt động.

SQLite không cần cấu hình, nghĩa là không cần thiết phải cài đặt.

Một SQLite Database đầy đủ được lưu giữ trong một disk file đơn.

SQLite rất nhỏ gọn, nhỏ hơn 400kB đã đươc cấu hình đầy đủ hoặc nhỏ hơn 250kB khi đã bỏ qua các tính năng tùy ý.

SQLite tự chứa, nghĩa là không có sự phụ thuộc vào ngoại vi.

Các Transaction trong SQLite là tuân theo đầy đủ chuẩn ACID, đảm bảo truy cập an toàn từ nhiều tiến trình hoặc thread.

SQLite hỗ trợ hầu hết các tính năng của một ngôn ngữ truy vấn trong chuẩn SQL92.

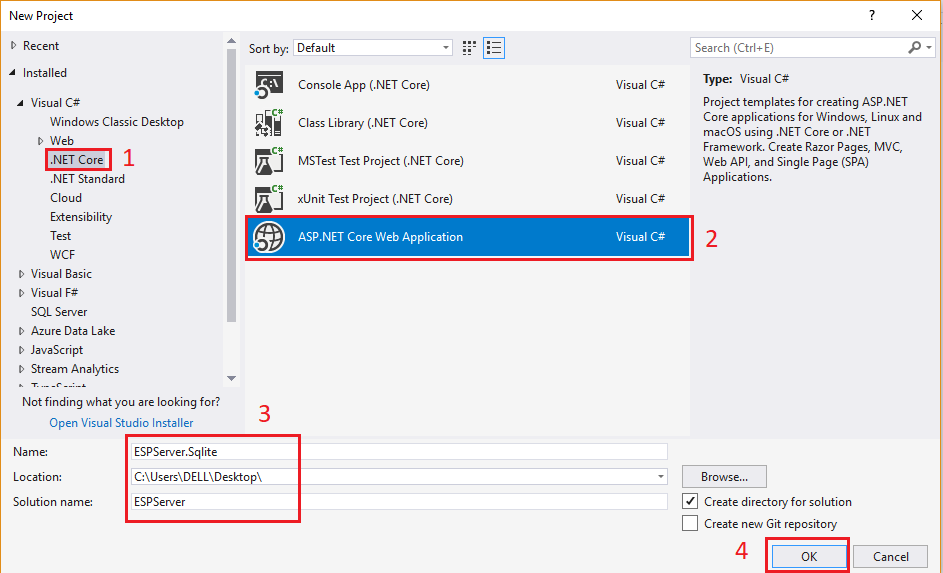
SQLite được viết bằng ANSI-C và cung cấp API đơn giản và dễ dàng sử dụng.

SQLite là có sẵn trên UNIX (Linux, Mac OS-X, Android, iOS) và Windows (Win32, WinCE, WinRT).

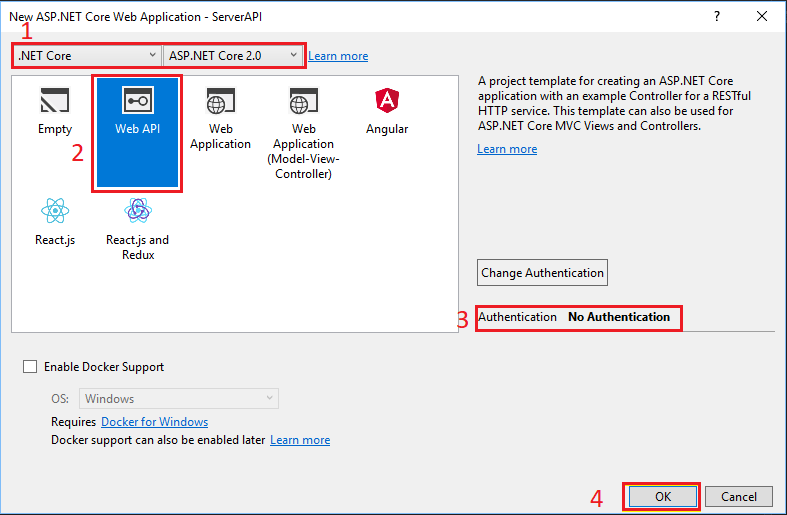
# CHƯƠNG 8: XÂY DỰNG WEB API BẰNG ASP.NET CORE SỬ DỤNG SQLITE

## Các bước thực hiện

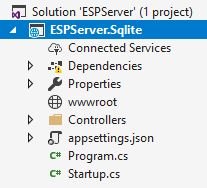
### Bước 1 : Tạo một project ASP.NET Core API



Hình 8.14: Tạo một project (1)



Hình 8.15: Tạo một project (2)



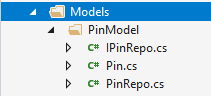
Hình 8.16: Kết quả sau khi tạo một project

### Bước 2 : Tạo model để quản lí dữ liệu

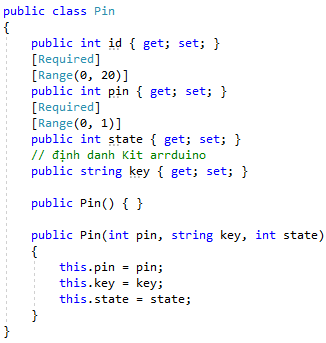
#### PinModel

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

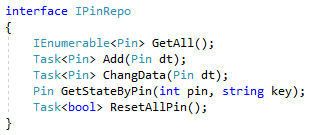
* Class Pin.cs : dùng để định nghĩa các thuộc tính dữ liệu của một GPIO
* Interface IPinRepo.cs : dùng để định nghĩa các phương thức
* Class PinRepo.cs : dùng để implement các phương thức từ IPinRepo.cs



Hình 8.17: Tạo PinModel



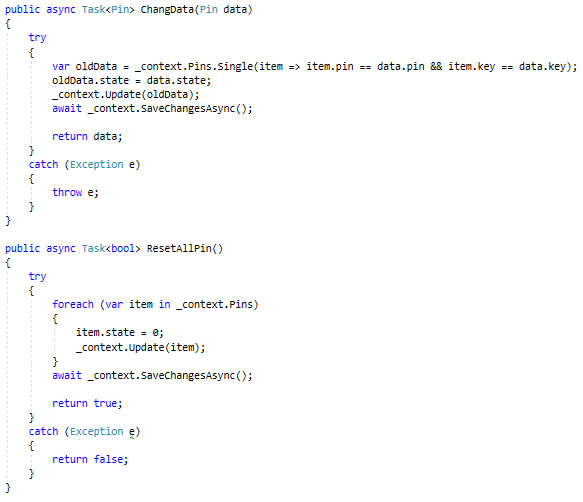
Hình 8.18: Nội dung file Pin..cs



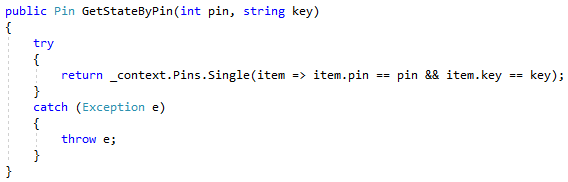
Hình 8.19: Nội dung file IPinRepo.cs



Hình 8.20: Nội dung file PinRepo.cs (1)



Hình 8.21: Nội dung file PinRepo.cs (2)

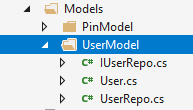


Hình 8.22: Nội dung file PinRepo.cs (3)

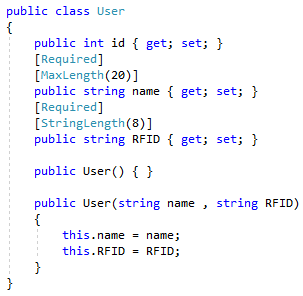
#### UserModel

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

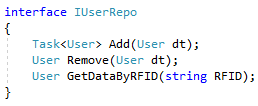
* Class User.cs : dùng để định nghĩa các thuộc tính của một user
* Interface IUserRepo.cs : dùng để định nghĩa các phương thức
* Class UserRepo.cs : dùng để implement các phương thức từ IUserRepo.cs



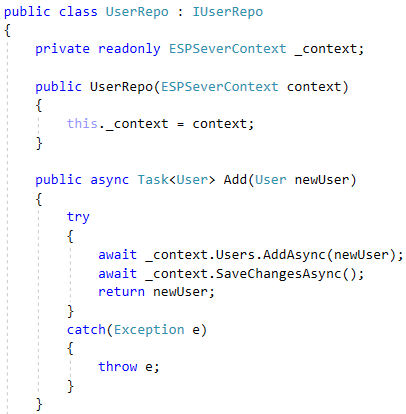
Hình 8.23: Tạo UserModel



Hình 8.24: Nội dung file User.cs



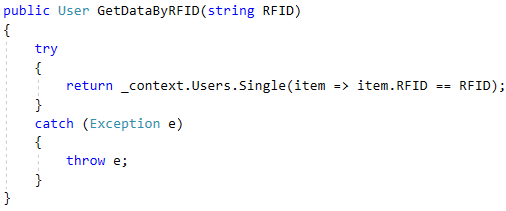
Hình 8.25: Nội dung file IUserRepo.cs



Hình 8.26: Nội dung file UserRepo.cs (1)



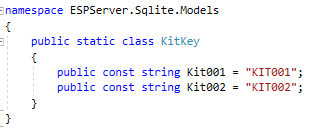
Hình 8.27: Nội dung file UserRepo.cs (2)



Hình 8.28: Nội dung file UserRepo.cs (3)

#### KitKey

Dùng để định danh cho từng kit Arduino

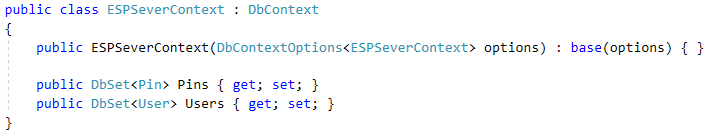


Hình 1.1: Nội dung file KitKey.cs

### Bước 3 : Xây dựng ESPServerContext để giao tiếp với Database bằng CodeFirst



Hình 8.29: Tạo file ESPServerContext.cs



Hình 8.30: Nội dung file ESPServerContext.cs

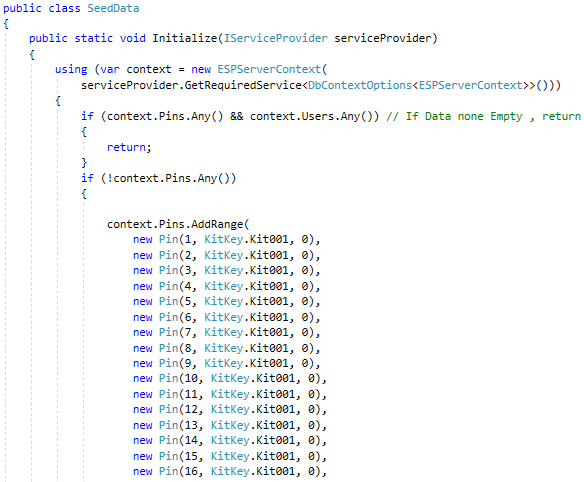
### Bước 4 : Tạo dữ liệu đầu vào

Các bạn có thể add dữ liệu bằng tay hoặc tạo file seed để đưa dữ liệu vào trong lần đầu tiên chạy.

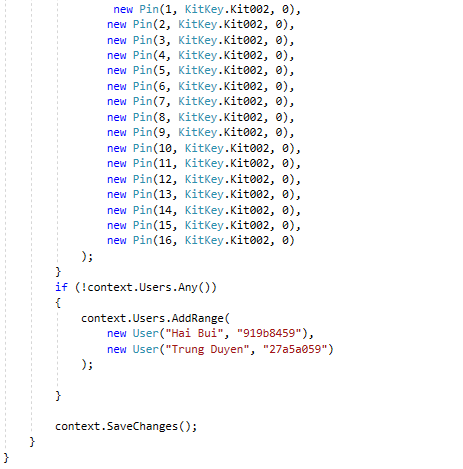
Tạo một file SeedData.cs để cấu hình dữ liệu đầu vào



Hình 8.31: Tạo file SeedData.cs

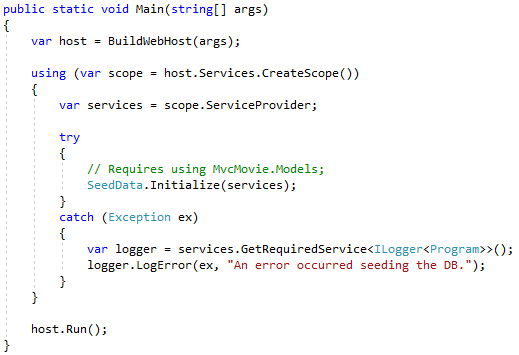


Hình 8.32: Nội dung file SeedData.cs (1)



Hình 8.33: Nội dung file SeedData.cs (2)

### Bước 5 : Đăng kí SeedData trong file Program.cs



Hình 8.34: Đăng kí SeedData trong Program.cs

### Bước 6 : Xóa ValueController.cs và tạo một Controller mới

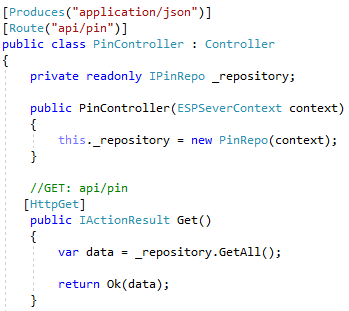


Hình 8.35: Tạo Controller

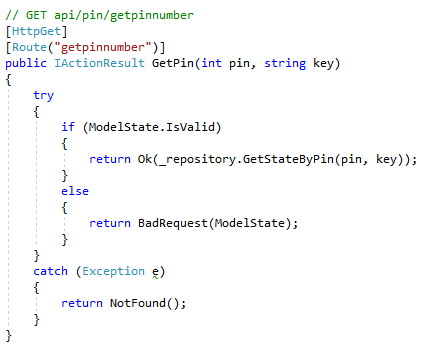
#### PinController

Nội dung file PinController.cs bao gồm các Action :

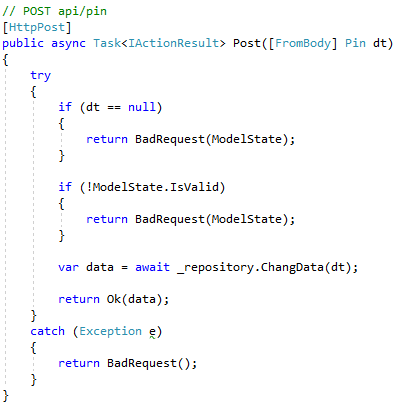
* Action Get() : Mặc định khi chạy server sẽ chạy hàm này đầu tiên , kết quả trả về là một mảng Data .
* Action Get(int pin) : Hàm này sẽ trả về đối tượng Data có giá trị pin bằng giá trị pin được yêu cầu .
* Action Post([FromBody] Data dt) : Hàm này sẽ nhận một đối tượng Data từ Request và chỉnh sửa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



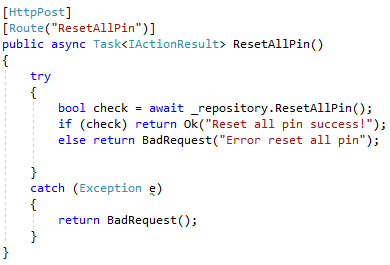
Hình 8.36: Nội dung file PinController.cs (1)



Hình 8.37: Nội dung file PinController.cs (2)



Hình 8.38: Nội dung file PinController.cs (3)

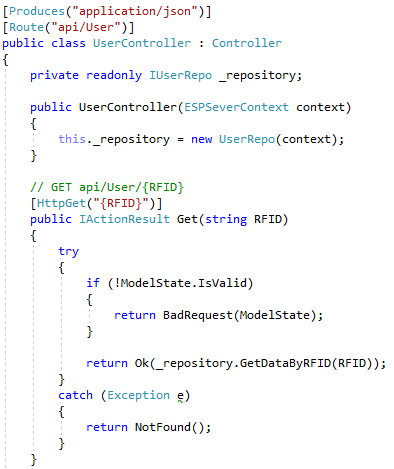


Hình 8.39: Nội dung file PinController.cs (4)

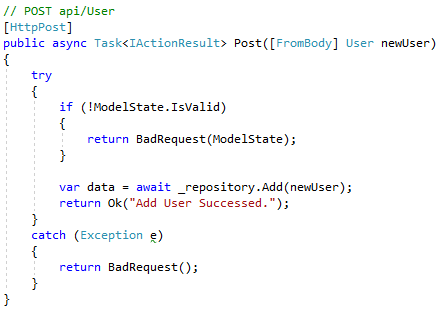
#### UserController

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 8.40: Nội dung file UserController.cs (1)



Hình 8.41: Nội dung file UserController.cs (2)



Hình 8.42: Nội dung file UserController.cs (3)

### Bước 7 : Thay đổi nội dung file Properties => lauchSettings.json

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



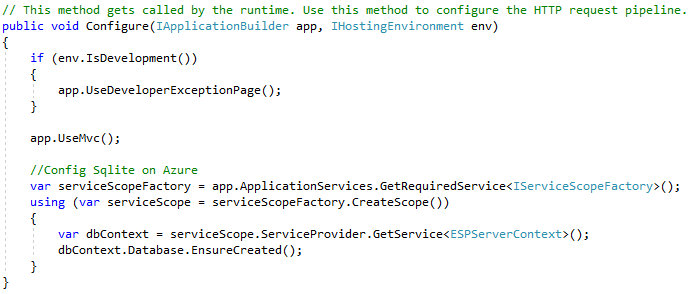
Hình 8.43: Nội dung file lauchSetting.json

### Bước 8 : Đăng kí ConnectionString trong file Startup.cs



Hình 8.44:Đăng kí đường dẫn đến thư mục sqlite

### Bước 9: Thay đổi nội dung hàm Configure trong file Startup.cs



Hình 8.45: Nội dung hàm Configure trong file Startup.cs

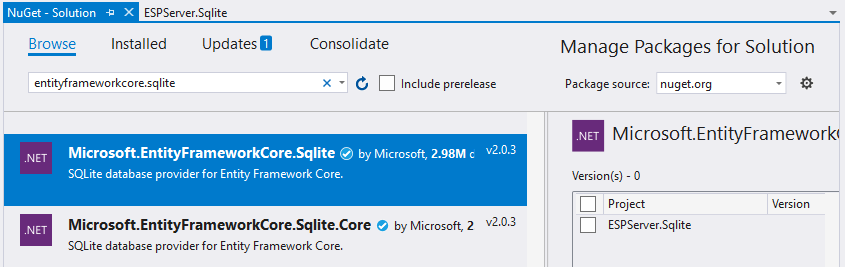
### Bước 10 : Install packet EntityFrameworkCore

Chọn Tools => Nuget Packet Manager => Packet Manager Console

Install các packet sau , lưu ý version có thể thay đổi theo thời gian

***Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 2.0.2***

***Install-Package Microsoft.EntityFrameworkCore.Tools***



Hình 8.46: Install Microsoft.EntityFrameworkCore.Sqlite

Sau khi Install packet ta tiến hành tạo file config để config database , ta không tạo bằng tay mà sử dụng Migration , chạy lệnh: ***Add-Migration InitialCreate***

Khi chạy xong ta có một thư mục tên là Migrations như hình

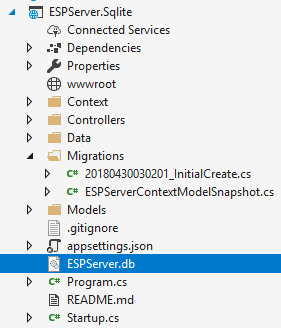


Hình 8.47: Thư mục Migrations

Các bạn có thể đọc nội dung để biết Migration config database như thế nào và sửa đổi trước khi tạo database.

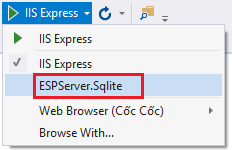
Ta đã có file config database , tiếp theo chúng ta sẽ build database dựa trên file config này bằng lệnh: ***Update-Database***

Sau khi chạy lệnh trên thì sẽ xuất hiện một thư mục ngay trong source file server như sau



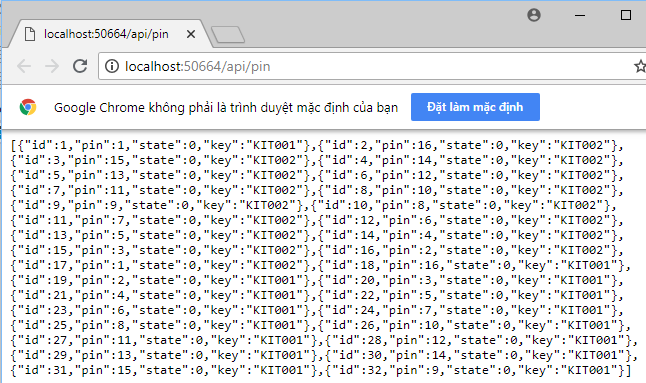
Hình 8.48: Kêt quả tạo Database Sqlite

### Bước 11 : Chạy Server



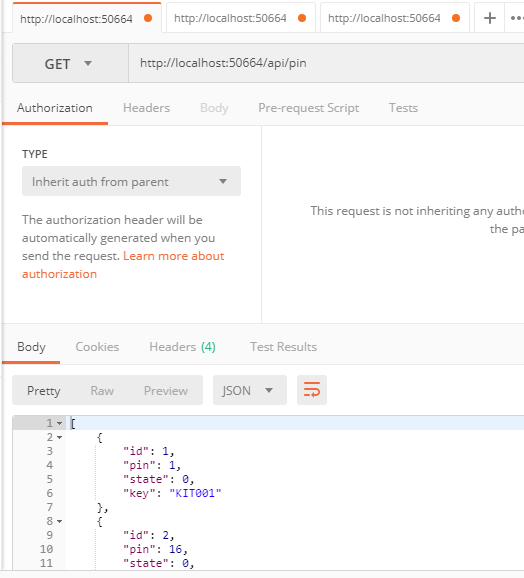
Hình 8.49: Run project bằng Command line

### Bước 12 : Kiểm tra kết quả trên trình duyệt

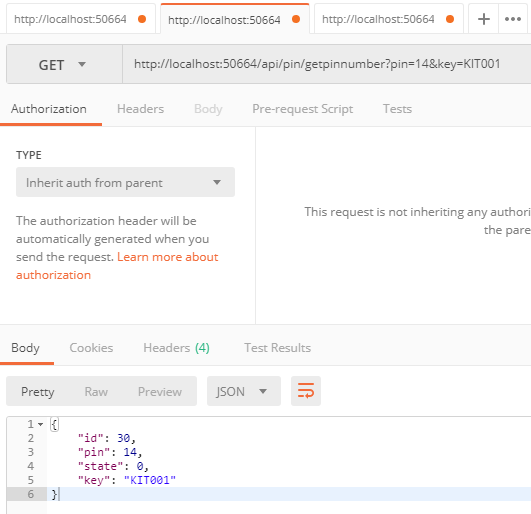


Hình 8.50: Kết quả sai khi chạy Server

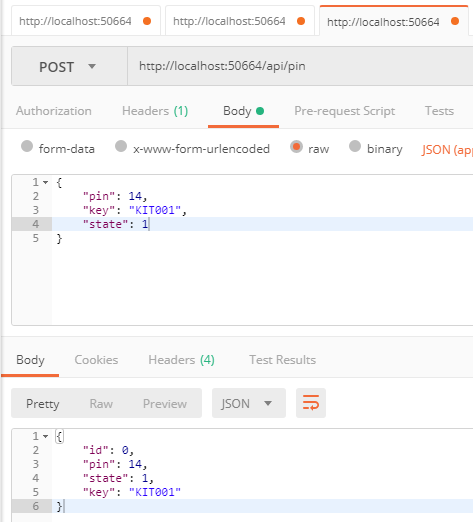
### Bước 13 : Test các hàm trong file PinController.cs bằng Postman



Hình 8.51: Action Get()



Hình 8.52: Action Get(int pin)



Hình 8.53: Action Post(Data dt)

Tương tự đối với UserController.

## Ngrok

### Tại sao lại cần sử dụng Ngrok trong đồ án này ?

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.

Để có thể kết nối dễ dàng thì ta sẽ sử dụng đến phần mềm tạo tunnel – Ngrok.

### Tổng quan

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.

Ví dụ: mydomain.ngrok.io => localhost:80

### 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 .

### Cài đặt Ngrok

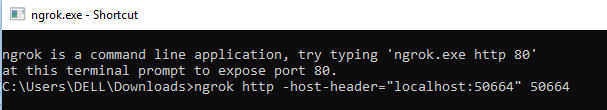
Ngrok không cần cài đặt , chỉ cần download về và chạy file .exe

### Chạy Ngrok

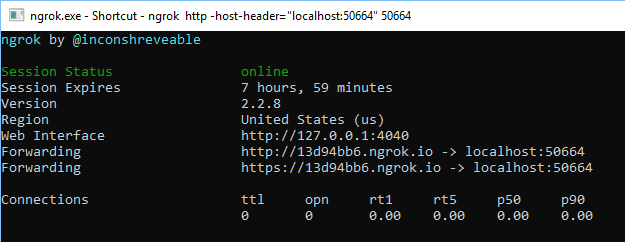
Chạy file .exe và patse lệnh sau vào command line

“ngrok http -host-header="localhost:50664" 50664”

Lưu ý : Thay đổi port



Hình 8.54: Chạy Ngrok



Hình 8.55: Kết quả sau khi chạy Ngrok

Như vậy ta thấy địa chỉ localhost:50664 đã được thay bằng địa chỉ mà Ngrok random ra http://13d94bb6.ngrok.io

### Limited connection

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

### Kiểm tra kết nối đến Ngrok

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



Hình 8.56: Kết quả kiểm tra kết nối Ngrok trên điện thoại

### 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

# CHƯƠNG 9: CODE ARDUINO

## Sơ lượt

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.

Các bạn có thể tham khảo code được sử dụng từ những đoạn code ngắn này tại link này :

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

## Kết nối Wifi

### Import thư viện

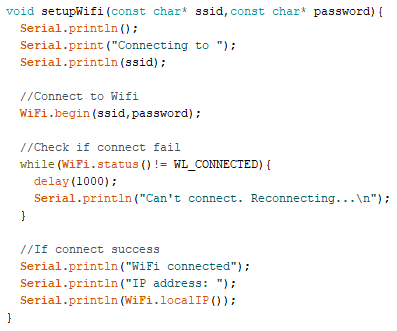
#include <ESP8266WiFi.h>

### Khai báo Wifi

const char\* ssid = “SSID”;

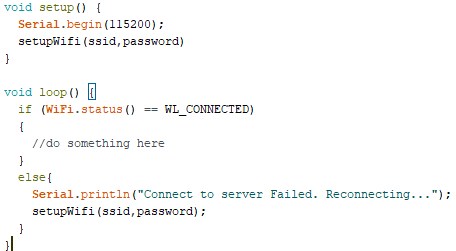
const char\* password = “Password”;

### Hàm setupWifi()



Hình 9.57: Hàm kết nối Wifi

### Vị trí đặt hàm setupWifi()



Hình 9.58: Vị trí đặt hàm setupWifi

## Kết nối Local Server

### Import thư viện

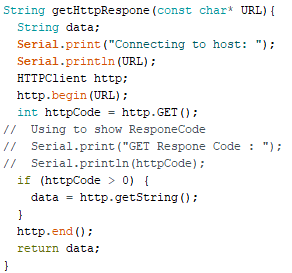
#include <ESP8266HTTPClient.h>

#include <ArduinoJson.h>

### Khai báo kết nối

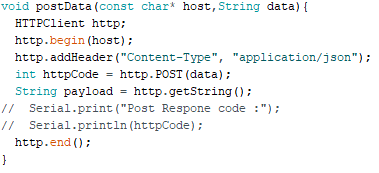
const char\* host = “Đường\_dẫn\_Ngrok\_server”;

### Get dữ liệu từ Server



Hình 9.59: Hàm getHttpResponse()

### Post dữ liệu lên Server



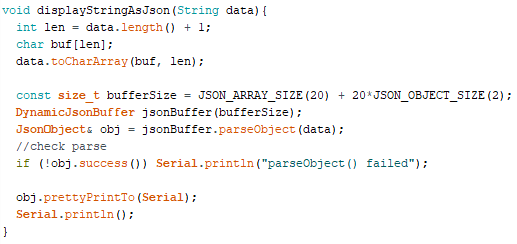
Hình 9.60: Hàm Post()

## Xử lý dữ liệu nhận được thành Json

### Hàm displayStringAsJson()

Dùng để hiển thị một đối tượng được trả về từ Server dưới dạng Json

“bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/

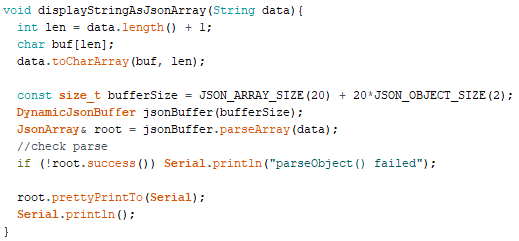


Hình 9.61: Hàm displayStringAsJson()

### Hàm displayStringAsJsonArray()

Dùng để hiển thị một danh sách các đối tượng được trả về từ Server dưới dạng Json

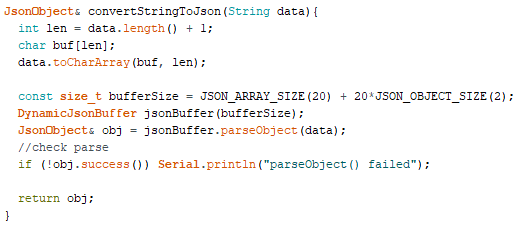
“bufferSize” có được dựa vào dạng Json mà Server gửi về . Tham khảo tại https://arduinojson.org/assistant/



Hình 9.62: Hàm displayStringAsJsonArray()

### Hàm convertStringToJson()

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



Hình 9.63: Hàm convertStringToJson()

### Hàm getIndexAtJsonArray()

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



Hình 9.64: Hàm getIndexAtJsonArray()

## Xử lý kết nối RFID

### Import thư viện

#include <SPI.h>

#include "MFRC522.h"

### Hướng dẫn nối chân dành cho ESP8266 Wifi UNO

SDA(SS) 15

SCK 14

MOSI 13

MISO 12

RST 2

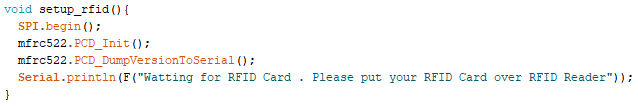
### Khai báo kết nối

#define SS\_PIN 15

#define RST\_PIN 2

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

### Hàm setup\_rfid()



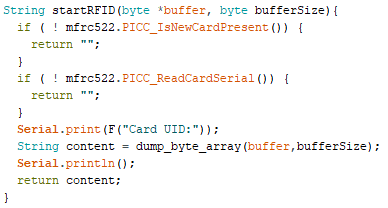
Hình 9.65: Hàm setup\_rfid()

### Hàm dump\_byte\_array()



Hình 9.66: Hàm dump\_byte\_array()

### Hàm startRFID()



Hình 9.67: Hàm startRFID()

## Xử lý kết nối OLED

### Import thư viện

#include <Adafruit\_NeoPixel.h>

#include "SSD1306.h"

#include "Wire.h"

### Khai báo kết nối

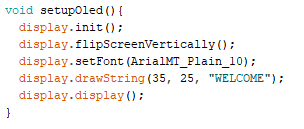
// SCL pin 4

// SDA pin 5

SSD1306 display(0x3c, 4, 5);

### Hàm setupOled()

Đặt hàm trong hàm setup()



Hình 9.68: Hàm setupOled()

### Code thay đổi screen

display.clear();

display.drawString(35, 15, "WELCOME");

display.drawString(35, 35, String(username));

display.display();

//Đặt đoạn code vào những chỗ phù hợp

## FIX các lỗi đã gặp

### Lỗi ESP8266 không nhận Wifi

#### Nguyên nhân

Do phiên bản sử dụng không thích hợp với board đang sử dụng

#### Giải pháp

Vào Tool => Board => Board Manager => Search “esp8266” => Lựa chọn phiên bản phù hợp => Install.

### Lỗi ESP8266 không kết nối được đến localhost

#### Nguyên nhân

Mặc định IIS chặn tất cả các kết nối truy cập đến Server ngoại trừ máy chạy Server.

#### Giải pháp

Thay vì chạy Server bằng IIS thì chạy chọn nút xuống và chọn chạy bằng ServerAPI ( ServerAPI được thay bằng tên solution tương ứng).

Tắt tường lửa.

Sử dụng Ngrok như đã hướng dẫn ở trên.

### Lỗi “Firmware Unknown” khi ESP8266 kết nối RFID

#### Nguyên nhân

Mỗi Board có cách kết nối dây khác nhau.

#### Giải pháp

Tìm cách nối dây tương ứng với board đang sử dụng . Link tham khảo https://github.com/miguelbalboa/rfid

# CHƯƠNG 10: CODE RASPBERRY PI

## OpenCV

### Giới thiệu

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,...

### Mục đích

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.

### Ứng dụng

Nhận dạng ảnh.

Xử lý hình ảnh.

Phục hồi hình ảnh/video.

Thực tế ảo.

Các ứng dụng khác.

## Kiến thức cơ bản về xử lý ảnh:

### Ảnh kỹ thuật số (Digital image):

Ảnh kỹ thuật số (Digital image) là một dạng biểu diễn của ảnh ở dạng ma trận số 2 chiều. Tùy vào độ phân giải của ảnh có cố định hay không, ảnh kỹ thuật số được chia ra làm 2 loại là ảnh vector (độ phân giải không cố định) và ảnh raster (hay còn gọi là bitmapped, độ phân giải cố định). Thuật ngữ ảnh kỹ thuật số thường được dùng để nói đến ảnh raster.

### Độ phân giải ảnh (Resolution):

Độ phân giải ảnh là mức độ chi tiết mà ảnh có thể thể hiện. Độ phân giải càng cao, ảnh càng nhiều chi tiết.

### Điểm ảnh (Pixel):

Trong ảnh kỹ thuật số, một điểm ảnh (pixel) là phần tử nhỏ nhất của ảnh raster (raster image). Mỗi một điểm ảnh là một mẫu (sample) của ảnh. Càng nhiều điểm ảnh, ảnh kỹ thuật số càng biểu diễn chính xác hơn về nội dung của ảnh gốc. Đặc trưng của một điểm ảnh gồm 2 thành phần: tọa độ (x,y) và cường độ sang (intensity).

### Mức xám của ảnh (Grayscale):

Mức xám của ảnh (Grayscale) là một trong những giá trị số của điểm ảnh biểu diễn mức độ ánh sáng (light intensity) tại điểm ảnh đấy. Thông thường, trong xử lý ảnh hiện tại, mức xám hay sử dụng nhất là 256 ( có giá trị mức xám từ 0 ->255).

### Ảnh màu:

Để biểu diễn ảnh màu, theo cách biểu diễn RGB, ba ma trận mức xám 256, ứng với ba màu đỏ (R), lục (G), lam (B) được sử dụng. Màu sắc của một điểm ảnh được quyết định bởi giá trị cường độ (intensity) tại ba ma trận màu cùng tọa độ.

## Code face-detection sử dụng Raspberry Pi, OpenCV

### Sơ lượt

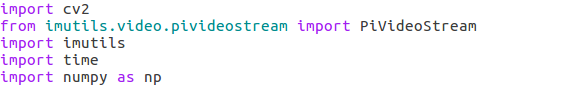
Mục đích của việc nhận dạng khuôn mặt trong đề tài này là khi bật chế độ security thì camera bắt đầu hoạt động. Camera có nhiệm vụ ghi hình, raspberry pi nhận hình ảnh từ camera và bắt đầu nhận diện khuôn mặt. Khi phát hiện có khuôn mặt sẽ gửi một email chứa hình ảnh nhận diện đến cho chủ nhà, đồng thời hình ảnh đó sẽ được gửi lên cloud. Sau khi gửi mail xong, mặc dù vẫn có nhận diện khuôn mặt nhưng không thể gửi mail liên tục được vì có nhiều bất tiện, do vậy chúng ta sẽ xét một khoảng thời gian nhất định để gửi mail.

Code nhận diện khuôn mặt sử dung Raspberry Pi, OpenCV bao gồm các file camera.py, mail.py, main.py, requirements.txt, thư mục models và thư mục templates với các chức năng được nói đến trong các mục sau.

### File camera.py

File camera.py có nhiệm vụ nhận dữ liệu từ camera, đóng khung hình ảnh khuôn mặt, lưu hình ảnh ngay lúc nhận diện được vào thư mục cùng cấp, trả về giá trị mã hoá hình ảnh thành bộ nhớ đệm dưới dạng chuỗi và trả về giá trị boolean cần thiết trong file main.py.

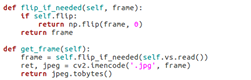
Import các thư viện cần thiết:



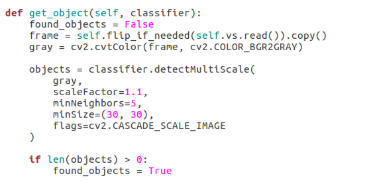
**Class VideoCamera**: gồm các hàm sau



Hình 10.6: Nhận tín hiệu bật tắt camera

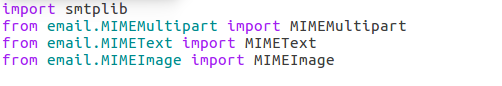


Hình 10.7: Cắt frame cần thiết và trả về giá trị mã hoá hình ảnh trong bộ nhớ đệm dưới dạng chuỗi



Hình 10.8: Nhận frame, vẽ khung hình vuông quanh mặt

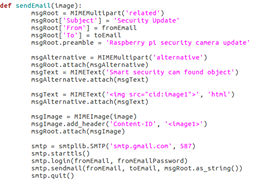
### File mail.py



Hình 10.9: Thư viện cần thiết để sử dụng mail

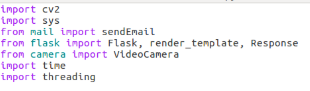


Hình 10.10: Mail nhận và mail gửi

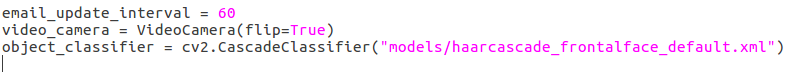


Hình 10.11: Hàm gửi email

### File main.py



Hình 10.12: Import thư viện



Hình 10.13: Khởi tạo các biến

email\_update\_interval: chỉ gửi email một lần trong khoảng thời gian này.

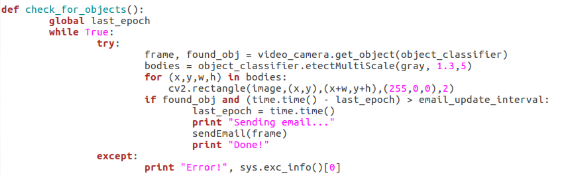
video\_camera: khởi tạo đối tượng camera theo chiều dọc.

object\_classifier: classifier Opencv dùng để nhận diện khuôn mặt.



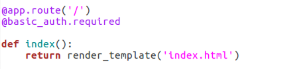
Hình 10.14: App Flobals của Flask

Sử dụng app globals này để yêu cầu đăng nhập khi truy cập.

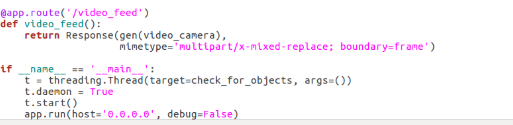


Hình 10.15: Hàm nhận diện khuôn mặt, gửi email

Sau khi nhận diện diện khuôn mặt, xét thời gian và gửi hình ảnh có đóng khung khuôn mặt cho email chủ nhà, đồng thời gửi hình ảnh đó lên cloud.

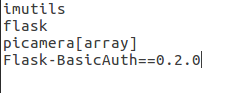


Hình 10.16: Mở trang index



Hình 10.17: Chạy chương trình ở địa chỉ local

### File requirements.txt



Hình 10.18: Các thành phần cần thiết

### Thư mục models



Hình 10.19: Các file nhận diện

Thư mục này chứa các file xml được dùng để nhận diện . Có thể là nhận diện khuôn mặt, nhận diện body,… Tuỳ vào mục đích sử dụng mà sử dụng file cần thiết.

### Thư mục templates

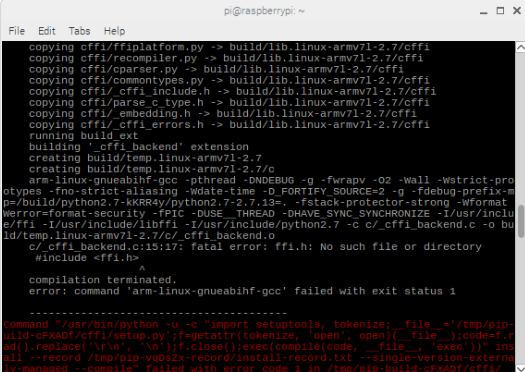
Thư mục này chứa file index.html là file hiển thị giao diện hiển thị.

## FIX các lỗi đã gặp

### Lỗi không ssh được Raspberry Pi

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.

### Lỗi khi cài đặt Cryptography



Hình 10.20: Lỗi cài đặt Cryptography

Giải pháp: Chạy lần lượt các câu lệnh sau:

sudo apt-get install build-essential libssl-dev libffi-dev python-dev

pip install cryptography

sudo pip install --pre azure

# CHƯƠNG 11: CLOUD

## Tổng quan

### Khái niệm

Theo Wikipedia thì Cloud Computing là môi trường tính toán dựa trên internet mà ở đó tất cả phần mềm, dữ liệu, tài nguyên được cung cấp cho máy tính và các thiết bị khác theo nhu cầu.

Theo Ian Foster: Cloud Computing là một mô hình điện toán phân tán có tính co giãn lớn mà hướng theo co giãn về mặt kinh tế, là nơi chứa các sức mạnh tính toán, kho lưu trữ, các nền tảng và các dịch vụ được trực quan, ảo hoá và co giãn linh động, sẽ được phân phối theo nhu cầu cho khách hàng bên ngoài thông qua internet.

Theo Rajkuma Buyya: Cloud là một hệ thống phân bố và xử lý song song gồm các máy tính ảo kết nối với nhau và được cung cấp động cho người dùng như một hoặc nhiều tài nguyên đồng nhất dựa trên sự thoả thuận dịch vụ giữa nhà cung cấp và người sử dụng.

Theo Viện tiêu chuẩn và Công nghệ quốc gia Bộ thương mại Mỹ (NIST): điện toán đám mây là một mô hình cho phép truy cập mạng thuận tiện, theo nhu cầu đến một kho tài nguyên điện toán dùng chung, có thể định cấu hình (ví dụ như mạng, lưu trữ, máy chủ, ứng dụng) có thể được cung cấp và thu hồi một cách nhanh chóng với nhu cầu tối thiểu về quản lý hoặc can thiệp của nhà cung cấp dịch vụ.

### Thành phần

Các thành phần của Cloud Computing bao gồm:

Hạ tầng: Cơ sở hạ tầng (Infracstructure) của Cloud Computing là phần cứng được cung cấp như là các dịch vụ, nghĩa là được chia sẻ và có thể sử dụng lại dễ dàng. Các tài nguyên phần cứng được cung cấp theo thời gian cụ thể theo yêu cầu.

Lưu trữ (Storage): Lưu trữ đám mây là khái niệm tách dữ liệu khỏi quá trình xử lý và chúng được lưu trữ ở những vị trí từ xa

Cloud Runtime: Là dịch vụ phát triển phần mềm ứng dụng và quản lý các yêu cầu phần cứng, nhu cầu phần mềm.

Dịch vụ: Dịch vụ đám mây là một phần độc lập có thể kết hợp với các dịch vụ khác để thực hiện tương tác, kết hợp giữa các máy tính với nhau để thực thi chương trình ứng dụng theo yêu cầu trên mạng.

Ứng dụng: Cloud Application là một đề xuất về kiến trúc phần mềm sẵn sàng phục vụ, nhằm loại bỏ sự cần thiết phải mua phần mềm, cài đặt,vận hành và duy trì ứng dụng tại thiết bị của người sử dụng.

Hạ tầng khách hàng (Client Infracstructure): là những yêu cầu phần mềm hoặc phần cứng để sử dụng các dịch vụ Cloud Computing trên mạng.

### Đặc tính

Sử dụng các tài nguyên tính toán động (Dynamic Computing Resources): Các tài nguyên được cấp phát cho doanh nghiệp đúng như những gì doanh nghiệp muốn một cách tức thời. Thay vì doanh nghiệp phải tính toán xem có nên mở rộng hay không, phải đầu tư bao nhiêu máy chủ thì nay doanh nghiệp chỉ cần yêu cầu và đám mây sẽ tự tìm kiếm tài nguyên rỗi để cung cấp.

Giảm chi phí: Doanh nghiệp sẽ có khả năng cắt giảm chi phí để mua bán, cài đặt và bảo trì tài nguyên. Rõ ràng thay vì việc phải cử chuyên gia đi mua máy chủ, cài đặt và bảo trì máy chủ thì nay chỉ cần xác định chính xác tài nguyên mình cần và yêu cầu.

Tăng khả năng sử dụng tài nguyên tính toán: Khi sử dụng tài nguyên đám mây thì không cần phải quan tâm đến việc đầu tư tài nguyên bao lâu thì nó sẽ hết khấu hao, đầu tư có lãi hay không, có bị outdate về công nghệ hay không.

### Các mô hình triển khai

Cho dù sử dụng loại mô hình dịch vụ nào đi nữa thì cũng có 3 mô hình triển khai chính là Public Cloud, Private Cloud và Hybrid Cloud.

Public Cloud: Các dịch vụ cloud được nhà cung cấp dịch vụ cung cấp cho mọi người sử dụng rộng rãi. Các dịch vụ được cung cấp và quản lý bởi một nhà cung cấp dịch vụ và các ứng dụng của người dùng đều nằm trên hệ thống cloud.

Private Cloud: cơ sở hạ tầng và các dịch vụ được xây dựng để phục vụ cho một tổ chức, doanh nghiệp duy nhất. Điều này giúp cho doanh nghiệp có thể kiểm soát tối đa đối với dữ liệu, bảo mật và chất lượng dịch vụ.

Hydrid Cloud: là sự kết hợp giữa Public Cloud và Private Cloud. Trong đó doanh nghiệp sẽ “out-source” các chức năng nghiệp vụ và dữ liệu không quan trọng, sử dụng các dịch vụ Public Cloud để giải quyết và xử lý các dữ liệu này. Đồng thời goanh nghiệp sẽ giữ lại các chức năng nghiệp vụ và dữ liệu tối quan trọng trong tầm kiểm soát (Private Cloud)

## Một số công cụ triển khai Web Server

Hiện tại ASP.NET Core được Microsoft hỗ trợ trên Windows , Mac OS và Linux nên việc triển khai ASP.NET Core trên Cloud trở nên dễ dàng hơn. Đối với mỗi hệ điều hành sẽ có những công cụ hỗ trợ chạy Server tương ứng.

### Internet Information Services

Đối với hệ điều hành Windows công cụ hỗ trợ phổ biến hiện nay là IIS (Internet Information Services). Đây là các dịch vụ dành cho máy chủ chạy trên hệ điều hành Window nhằm cung cấp và phân tán thông tin lên mạng. Nó bao gồm nhiều dịch vụ khác như Web Server, FTP Server,…

### Nginx

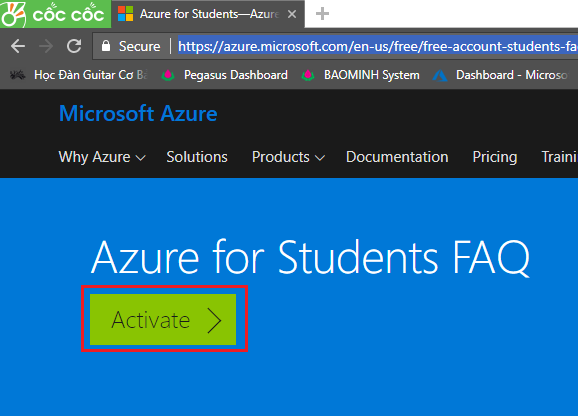
Nginx là một máy chủ reverse proxy mã nguồn mở cho các giao thức HTTP, HTTPS, SMTP, POP3, IMAP, cũng như là một máy chủ cân bằng tải, HTTP cache và web.

Nginx không dựa trên thread để xử lý yêu cầu. Nó sử dụng một kiến trúc bất đồng bộ hướng sự kiện linh hoạt. Kiến trúc này sử dụng ít nhưng quan trọng, là lượng bộ nhớ có thể dự đoán khi hoạt động.

## Triển khai trên cloud Azure

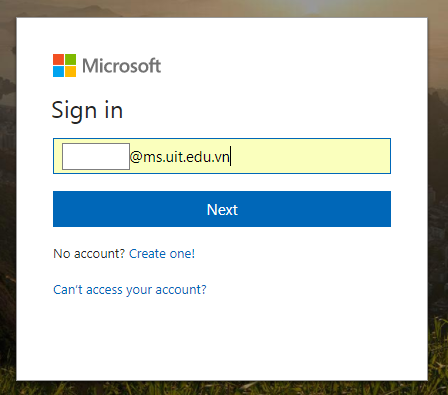
### Đăng nhập vào Azure với tài khoản trường (tài khoản ms.uit.edu.vn)

Vào link sau : <https://azure.microsoft.com/en-us/free/free-account-students-faq/>



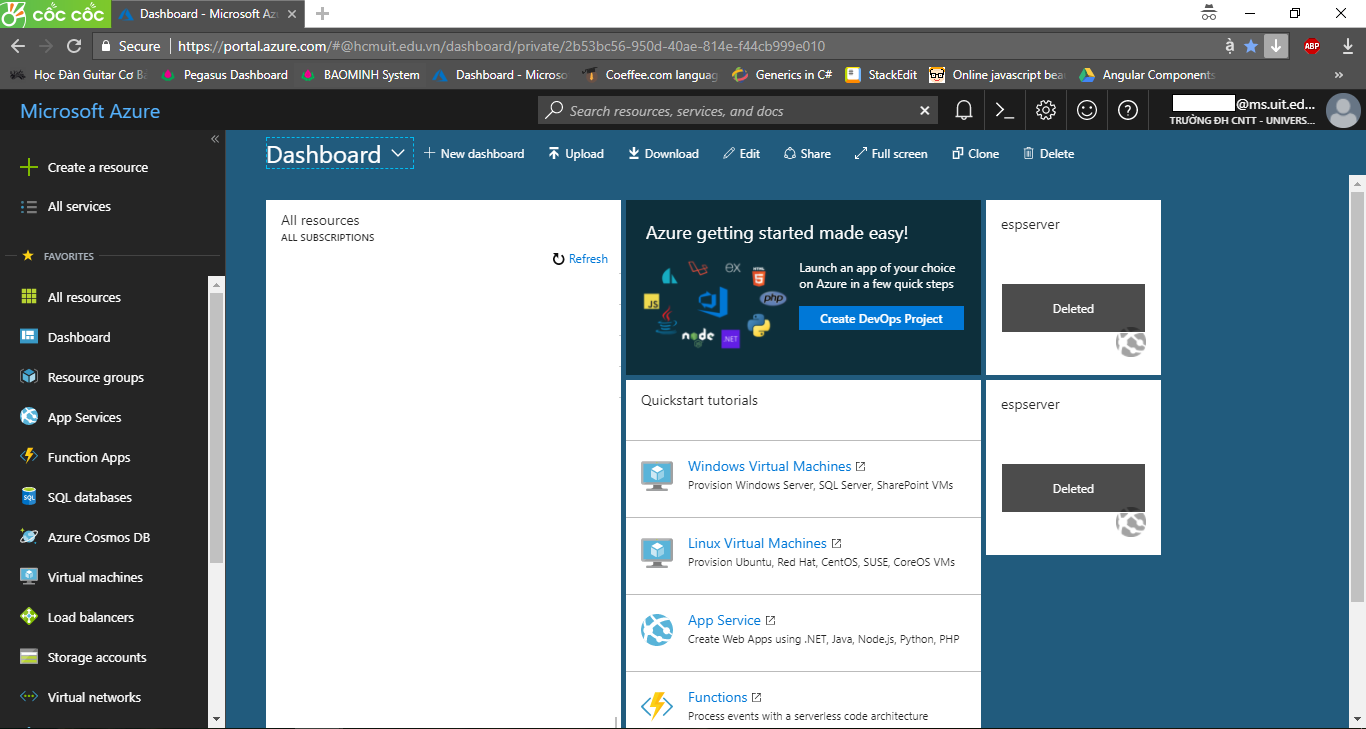
Hình 12.69: Trang chủ Azure for Student

Tiếp theo bấm Active ta sẽ được chuyển đến trang đăng nhập , đăng nhập bằng tài khoản ms của trường.



Hình 12.70: Đăng nhập bằng tài khoản ms

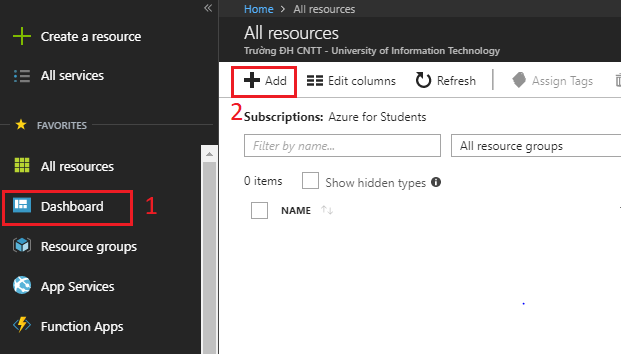
Sau khi đăng nhập thành công ta được chuyển đến trang chủ Azure Portal



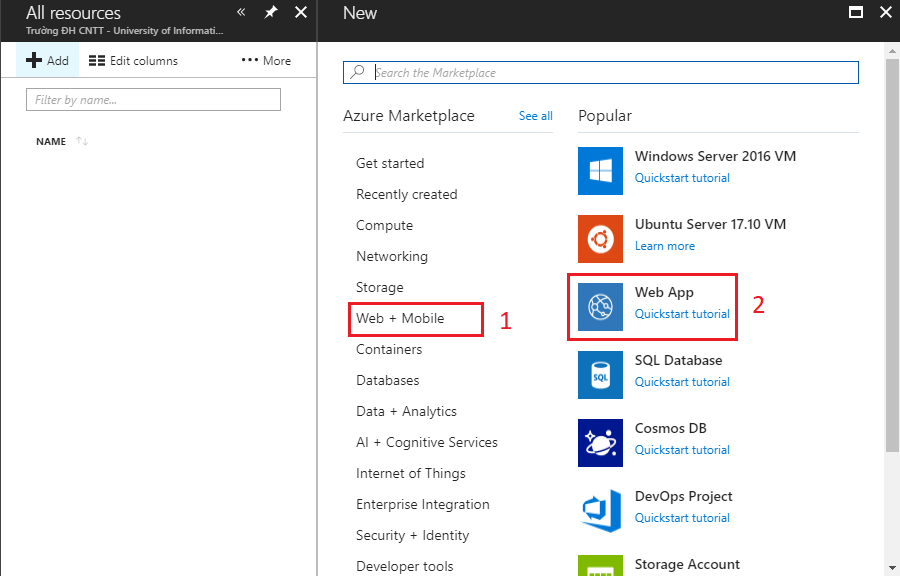
Hình 12.71: Trang chủ Azure Portal

### Tạo Web app

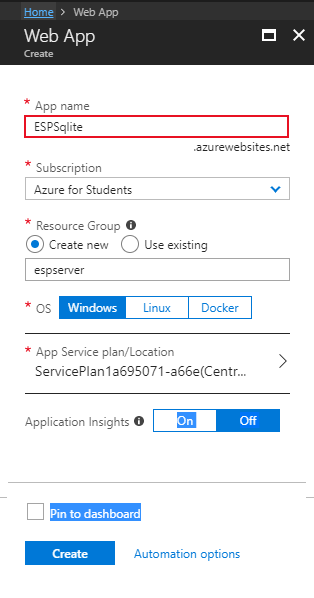
Ta tạo web app bằng cách thêm resource như sau



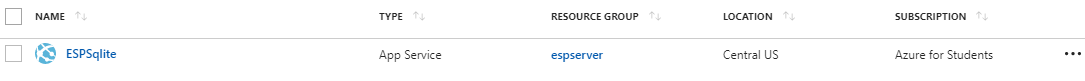
Hình 12.72: Thêm resource mới



Hình 12.73: Chọn WebApp

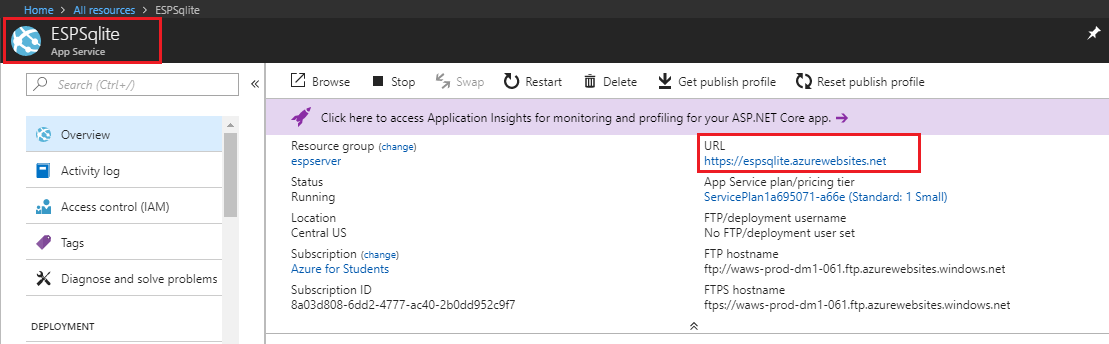


Hình 12.74: Tạo WebApp



Hình 12.75: Kết quả sau khi tạo WebApp

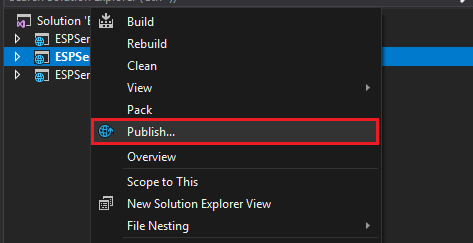
Nhấn vào WebApp vừa tạo ta có kết quả như hình sau



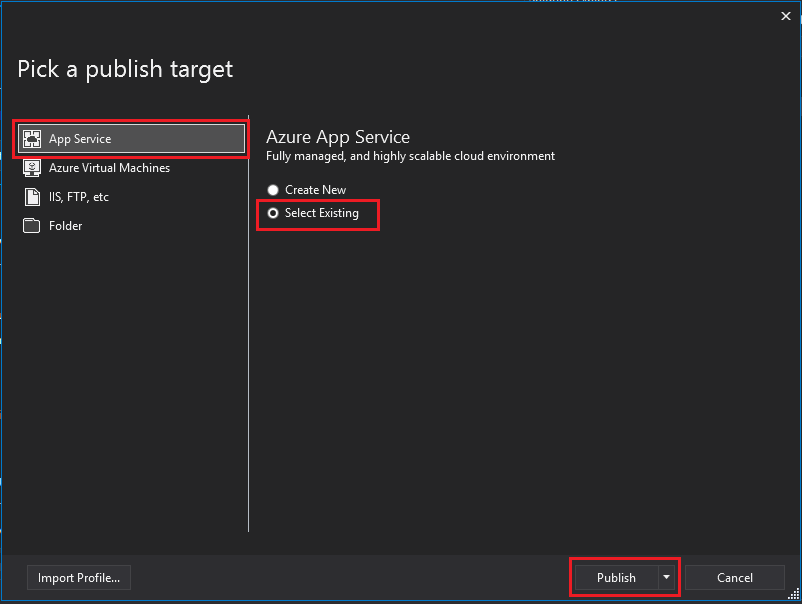
Hình 12.76: Giao diện WebApp

Click vào URL ta sẽ được chuyển đến trang API mà mình đã tạo. Nhưng trang này sẽ trả về 404 vì sai đường dẫn dẫn đến lỗi không tìm thấy .

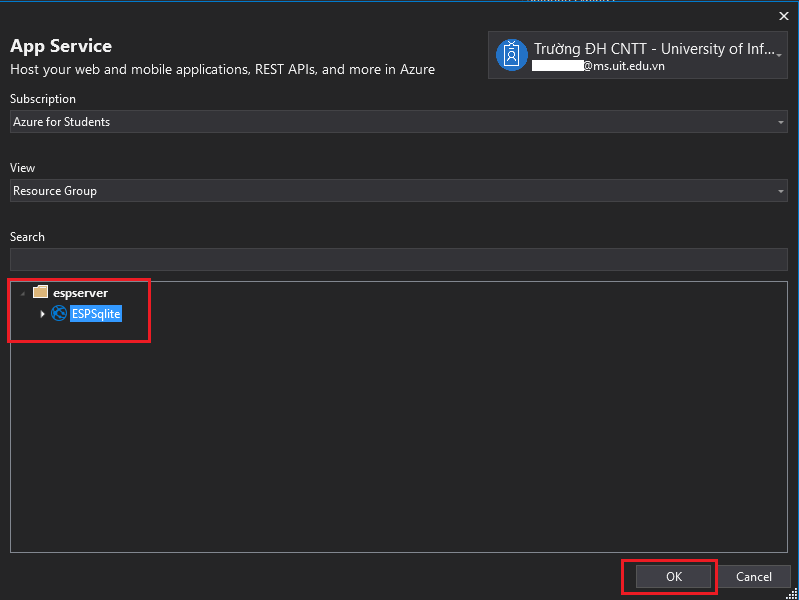
Tiếp theo ta publish Server từ máy chúng ta lên Azure thông qua Visual Studio. Chuột phải vào tên Project chọn Publish.



Hình 12.77: Deploy project lên Azure

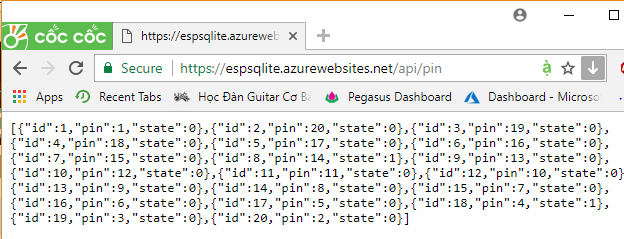


Hình 12.78: Chọn WebApp có sẵn để publish



Hình 12.79: Chọn WebApp đã tạo ở trên

Quá trình deploy sẽ diễn ra sau 1-2 phút . Kiểm tra lại đường dẫn WebApp ta được kết quả như sau

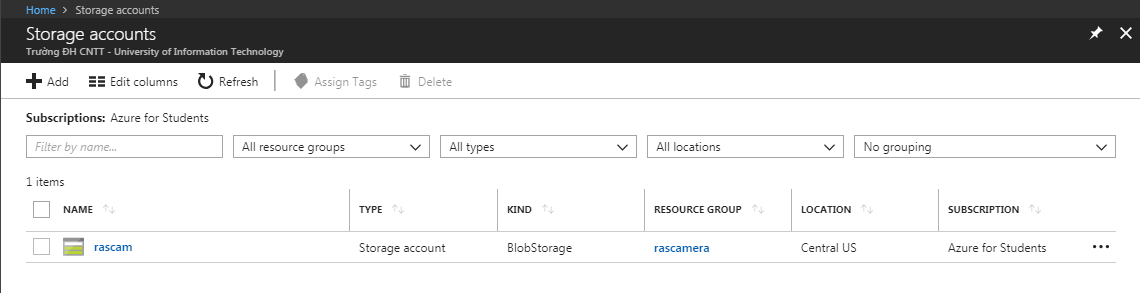


Hình 12.80: Kết quả sau khi deploy lên Azure

Đối với Azure hỗ trợ mặc định HTTP và HTTPS , các bạn có thể thử đường dẫn trên với HTTP , kết quả tương tự như trên.

### Tạo storage lưu trữ hình ảnh

Tạo storage account “rascame”, container “raspi”



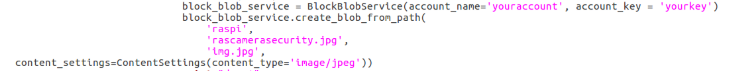
Hình 11.21: Azure storage account

Trong file main.py, ta thêm các thành phần để kết nối và gửi hình ảnh lên cloud như sau:



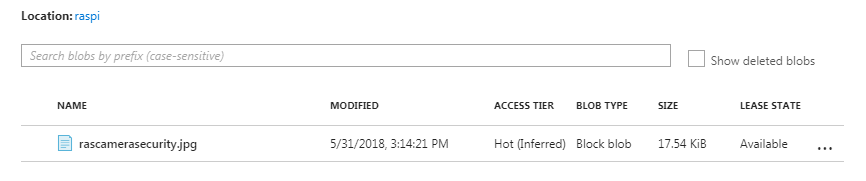
Hình 11.22: Import thư viện azure

Trong hàm check\_for\_object để gửi hình ảnh lên cloud ta thêm vào các dòng sau:

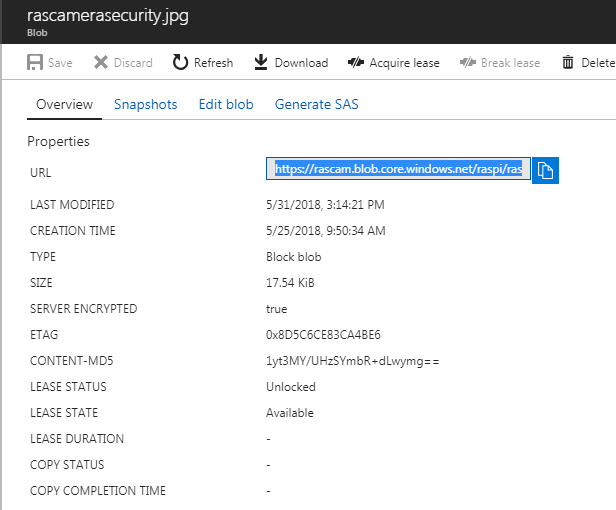


Hình 11.23: Gửi hình ảnh lên cloud

Sau khi nhận diện được, hình ảnh nhận diện sẽ được gửi lên cloud và kết quả như sau:

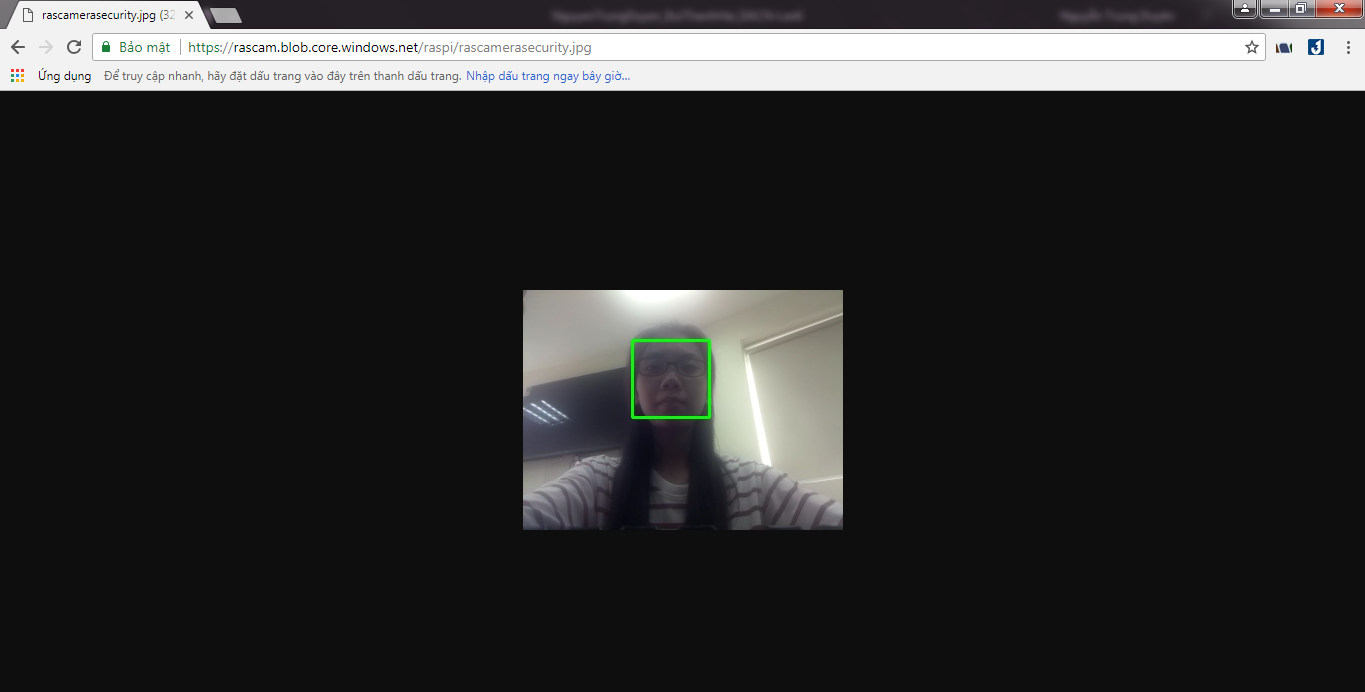


Hình 11.24: Upload hình ảnh lên azure



Hình 11.25: Thông tin hình ảnh

Truy cập vào đường link ta có kết quả:



Hình 11.26: Kết quả

# Kết luận và hướng phát triển

## Kết luận

Nhóm đã đạt được cơ bản mục tiêu đề ra như sử dụng RFID như một chìa khóa cửa. Mỗi thành viên trong gia đình sẽ có một RFID và có một kịch bản riêng. Khi một thẻ RFID được quét, hệ thống sẽ kiểm tra thông tin thẻ, nếu hợp lý thì gửi tín hiệu để cửa tự động mở, đồng thời tự động bật các thiết bị tương ứng với thẻ đó. Bên cạnh đó, nhóm đã bước đầu sử dụng công nghệ xử lý ảnh để nhận diện khuôn mặt và gửi email đến cho chủ nhà. Hi vọng đề tài này sẽ giúp được những bạn yêu thích về công nghệ IOT, cụ thể hơn là công nghệ IOT trong ngôi nhà thông minh có một khởi đầu thuận lợi hơn.

## Hạn chế

Do thời gian thực hiện đề tài còn hạn chế, nhóm vừa tiếp cận công nghệ mới trong thời gian ngắn nên còn nhiều hạn chế trong việc xử lý hệ thống, phân tích và lưu trữ dữ liệu. Một số chức năng vừa hoàn thiện về mặt cơ bản nhưng chưa đào sâu vào nghiên cứu và phát triển.

## Hướng phát triển

Qua việc thực hiện việc tự xây dựng mô hình IOT từ thiết lập phần mềm , phần cứng đến việc xây dựng Server để truyền dữ liệu . Nhóm đã hiểu rõ hơn quy trình để phát triển một ứng dụng IOT . Những công việc làm được trong đồ án này là bước tiến đầu tiên để xây dựng một mô hình IOT hoàn chỉnh. Trong tương lai nhóm sẽ tiếp tục xây dựng mô hình chi tiết hơn , bao gồm tính xác thực , bảo mật , mở rộng để hỗ trợ các mục tiêu tiếp theo mà nhóm hướng đến. Bên cạnh đó nhóm sẽ cố gắng tìm hiểu và sử dụng công nghệ AI để kết hợp vào sản phẩn IOT để phát triển đề tài hơn,

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | IOT Maker Việt Nam, "Lập trình IOT với ESP8266 Arduino," 2017. [Online]. Available: https://adruino.esp8266.vn/. |
| [2] | R. Anderson, "Working with SQLite in an ASP.NET Core MVC project," 7 4 2017. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app-xplat/working-with-sql?view=aspnetcore-2.1. |
| [3] | antepher, "ESP8266: HTTP GET Requests," 17 7 2016. [Online]. Available: https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/. |
| [4] | antepher, "ESP8266: HTTP POST Requests," 21 7 2016. [Online]. Available: https://techtutorialsx.com/2016/07/21/esp8266-post-requests/. |
| [5] | Dykstra, "Getting Started with Entity Framework 6 Code First using MVC 5," 22 10 2015. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application. |
| [6] | JorgenVikingGod, "WIFI RFID READER," 24 8 2015. [Online]. Available: http://www.instructables.com/id/WiFi-RFID-Reader/. |
| [7] | KhanhWAP, "Giới thiệu ngrok - Mang demo dự án web lên internet không cần deploy," 23 8 2016. [Online]. Available: https://viblo.asia/p/tooling-gioi-thieu-ngrok-mang-demo-du-an-web-len-internet-khong-can-deploy-naQZR7eqlvx. |
| [8] | J. Likness, "Create an ASP.NET Core web app in Azure," 14 6 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-dotnet. |
| [9] | Microsoft, "Entity Framework Code First to a New Database," 23 10 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx. |
| [10] | Đ. H. Toàn, "Cộng đồng Arduino Việt Nam," 12 9 2016. [Online]. Available: http://arduino.vn/bai-viet/1236-tim-hieu-giao-thuc-mqtt. |
| [11] | V. Anh, "Tìm hiểu và hướng dẫn setup web server Nginx," 26 6 2016. [Online]. Available: https://viblo.asia/p/tim-hieu-va-huong-dan-setup-web-server-nginx-OREGwBwlvlN. |
| [12] | Admin, "Các dịch vụ mạng," 10 3 2010. [Online]. Available: http://cntt01q5.forums-free.com/dich-vu-web-server-iis-t107.html. |
| [13] | A. Rosebrock, "Install OpenCV and Python on your Raspberry Pi 2 and B+," 23 2 2015. [Online]. Available: https://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/. |
| [14] | blogspot, "Nhận dạng khuôn mặt sử dụng camerapi trên raspberry pi. Face detect use Picamera+Python+Opencv," 30 3 20116. [Online]. Available: https://internetofthings23.blogspot.com/2016/03/nhan-dang-khuon-mat-su-dung-camerapi.html. |
| [15] | H. Shack, "Smart Security Camera," 19 9 2017. [Online]. Available: https://www.hackster.io/hackershack/smart-security-camera-90d7bd. |
| [16] | E. Wang, "Uploading PiCamera Images to Azure," 3 31 2017. [Online]. Available: https://www.hackster.io/ediwang/uploading-picamera-images-to-azure-44b994. |
| [17] | dinhnn, "Giới thiệu tổng quan về điện toán đám mây," 28 7 2014. [Online]. Available: https://dinhnn.com/2014/07/28/gioi-thieu-tong-quan-ve-dien-ton-dm-my/. |
| [18] | securitydaily, "Tìm hiểu về Internet Information Services (IIS)," 27 8 2014. [Online]. Available: https://securitydaily.net/tim-hieu-ve-internet-information-services-iis/. |