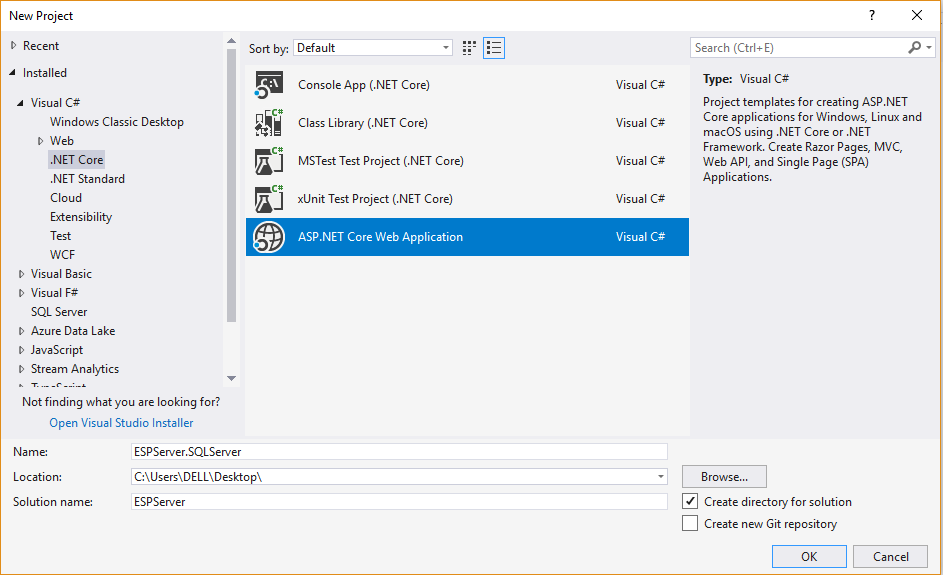
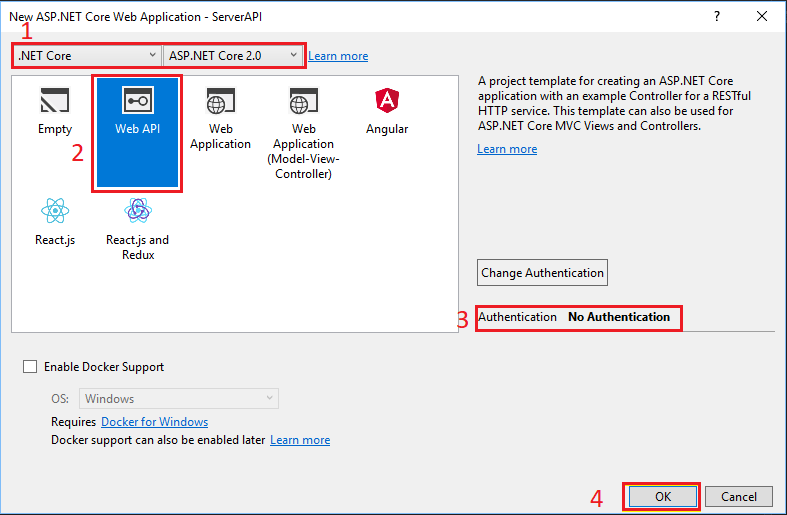
# CHƯƠNG XX: XÂY DỰNG WEB API BẰNG ASP.NET CORE TRÊN VISUAL STUDIO 2017

## Các bước thực hiện

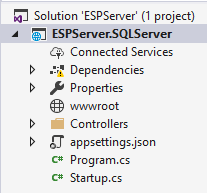
### Bước 1 : Tạo một project ASP.NET Core API



Hình 1.1: Tạo một project (1)



Hình 1.2: Tạo một project (2)



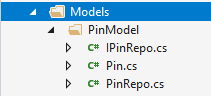
Hình 1.3: Kết quả sau khi tạo một project

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

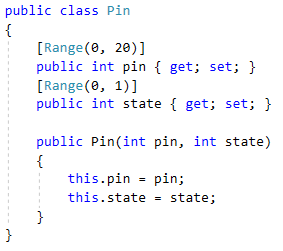
#### PinModel

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

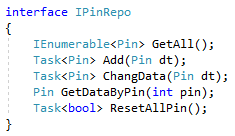
* Class Pin.cs : dùng để định nghĩa các thuộc tính dữ liệu của một GPIO
* Interface IPinRepo.cs : dùng để định nghĩa các phương thức
* Class PinRepo.cs : dùng để implement các phương thức từ IDataRepo.cs



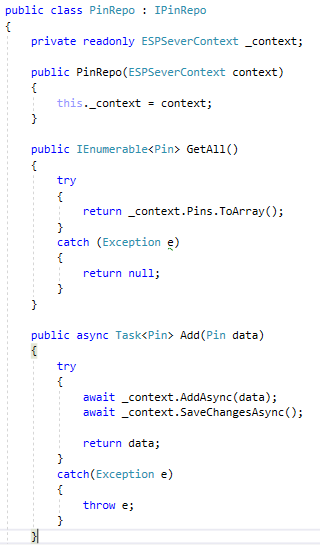
Hình 1.4: Tạo PinModel



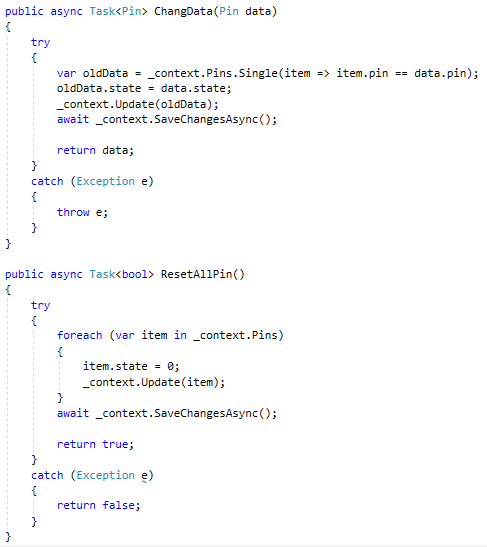
Hình 1.5: Nội dung file Pin..cs



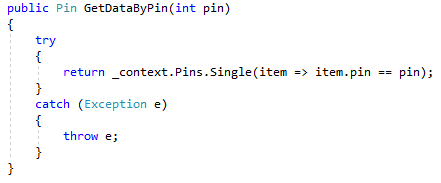
Hình 1.6: Nội dung file IPinRepo.cs



Hình 1.7: Nội dung file PinRepo.cs (1)



Hình 1.8: Nội dung file PinRepo.cs (2)

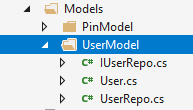


Hình 1.9: Nội dung file PinRepo.cs (3)

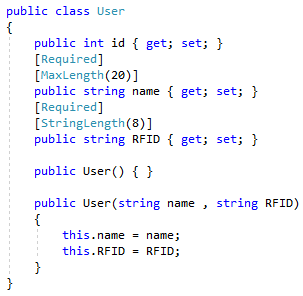
#### UserModel

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

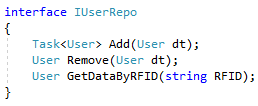
* Class User.cs : dùng để định nghĩa các thuộc tính của một user
* Interface IUserRepo.cs : dùng để định nghĩa các phương thức
* Class UserRepo.cs : dùng để implement các phương thức từ IUserRepo.cs



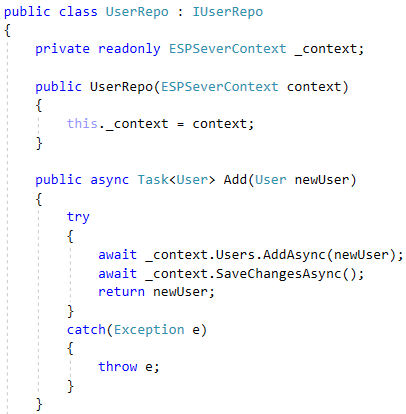
Hình 1.10: Tạo UserModel



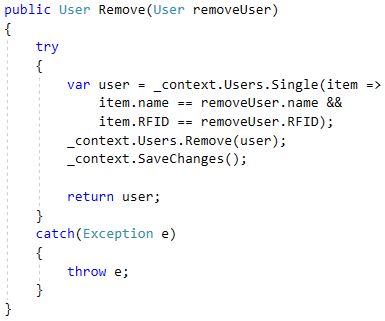
Hình 1.11: Nội dung file User.cs



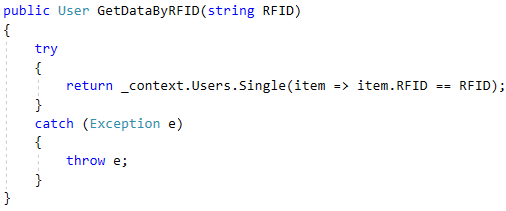
Hình 1.12: Nội dung file IUserRepo.cs



Hình 1.13: Nội dung file UserRepo.cs (1)



Hình 1.14: Nội dung file UserRepo.cs (2)

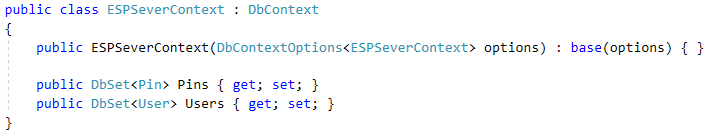


Hình 1.15: Nội dung file UserRepo.cs (3)

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



Hình 1.16: Tạo file ESPServerContext.cs



Hình 1.17: Nội dung file ESPServerContext.cs

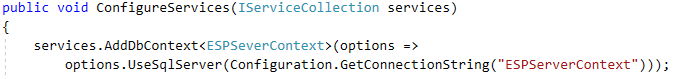
### Bước 4 : Thêm ConnectionStrings vào đầu file appsettings.json

"ConnectionStrings": {

"ServerAPIContext": "Server=(localdb)\\mssqllocaldb;Database=ServerAPI;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

### Bước 5 : Đăng kí ConnectionString trong file Startup.cs



Hình 1.18: Đăng kí ConnectionString

### Bước 6 : Install packet EntityFrameworkCore

1. Chọn Tools => Nuget Packet Manager => Packet Manager Console
2. 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***

1. 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***

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

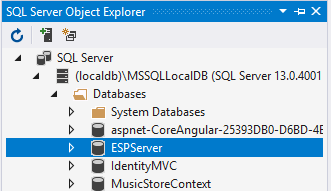


Hình 1.19: Thư mục Migrations

1. Các bạn có thể đọc nội dung để biết Migration config database như thế nào và sửa đổi trước khi tạo database.
2. Ta đã có file config database , tiếp theo chúng ta sẽ build database dựa trên file config này bằng lệnh

***Update-Database***

Kiểm tra database vừa tạo , vào View => SQL Server Object Explorer , ta sẽ có kết quả như hình.



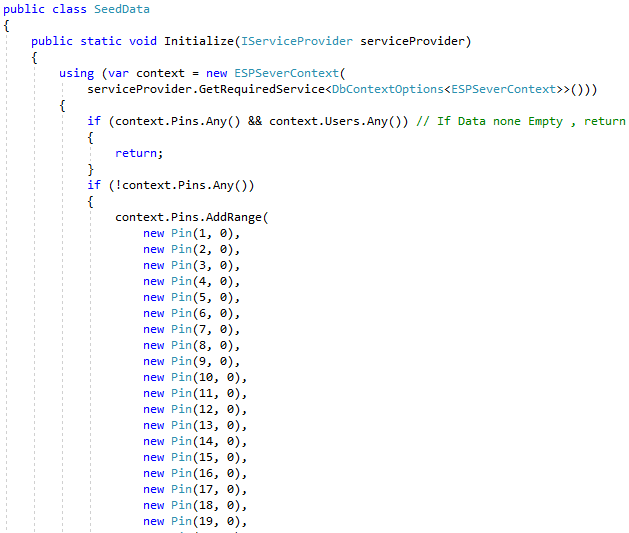
Hình 1.20: Kêt quả tạo Database bằng CodeFirst

### Bước 7 : Tạo dữ liệu đầu vào

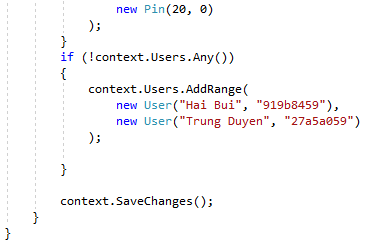
1. Các bạn có thể add dữ liệu bằng tay hoặc tạo file seed để đưa dữ liệu vào trong lần đầu tiên chạy.
2. Tạo một file SeedData.cs để cấu hình dữ liệu đầu vào



Hình 1.21: Tạo file SeedData.cs

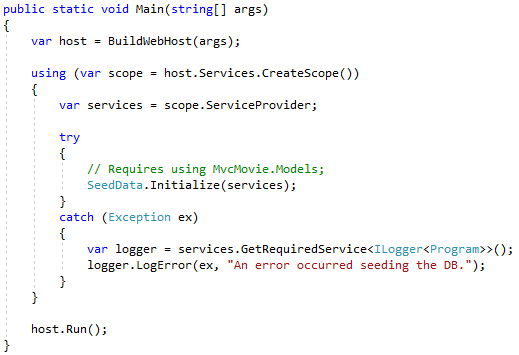


Hình 1.22: Nội dung file SeedData.cs (1)



Hình 1.23: Nội dung file SeedData.cs (2)

### Bước 8 : Đăng kí SeedData trong file Program.cs



Hình 1.24: Đăng kí SeedData trong Program.cs

### Bước 9 : Xóa ValueController.cs và tạo một Controller mới

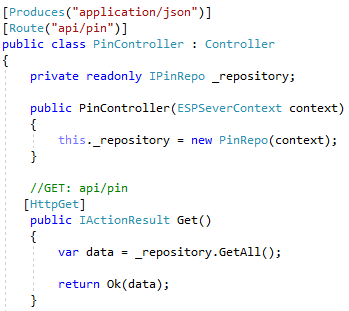


Hình 1.25: Tạo Controller

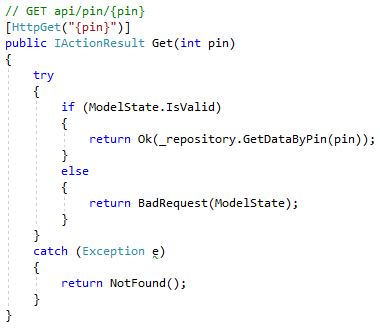
#### PinController

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

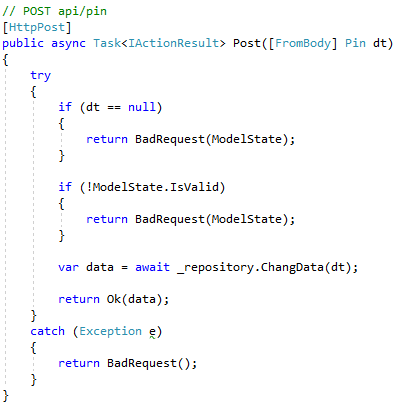
* Action Get() : Mặc định khi chạy server sẽ chạy hàm này đầu tiên , kết quả trả về là một mảng Data .
* Action Get(int pin) : Hàm này sẽ trả về đối tượng Data có giá trị pin bằng giá trị pin được yêu cầu .
* Action Post([FromBody] Data dt) : Hàm này sẽ nhận một đối tượng Data từ Request và chỉnh sửa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



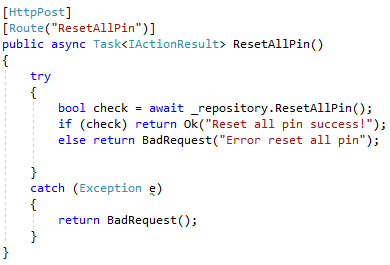
Hình 1.26: Nội dung file PinController.cs (1)



Hình 1.27: Nội dung file PinController.cs (2)



Hình 1.28: Nội dung file PinController.cs (3)

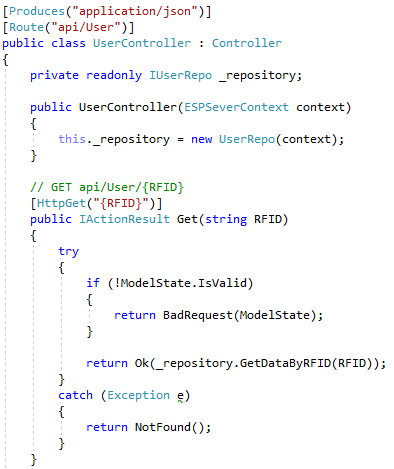


Hình 1.29: Nội dung file PinController.cs (4)

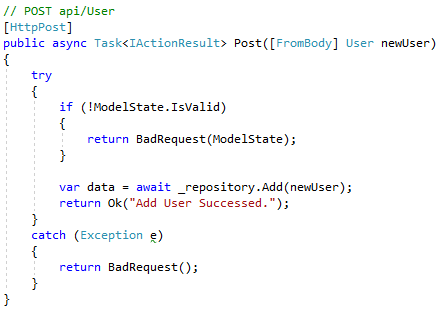
#### UserController

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

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



Hình 1.30: Nội dung file UserController.cs (1)



Hình 1.31: Nội dung file UserController.cs (2)



Hình 1.32: Nội dung file UserController.cs (3)

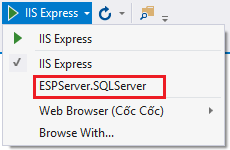
### Bước 10 : Thay đổi nội dung file Properties => lauchSettings.json

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



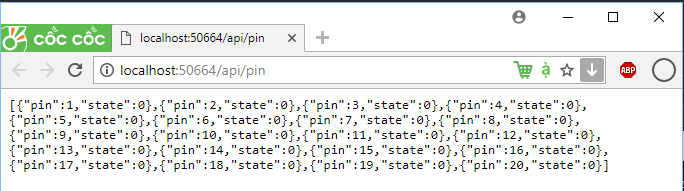
Hình 1.33: Nội dung file lauchSetting.json

### Bước 11 : Chạy Server



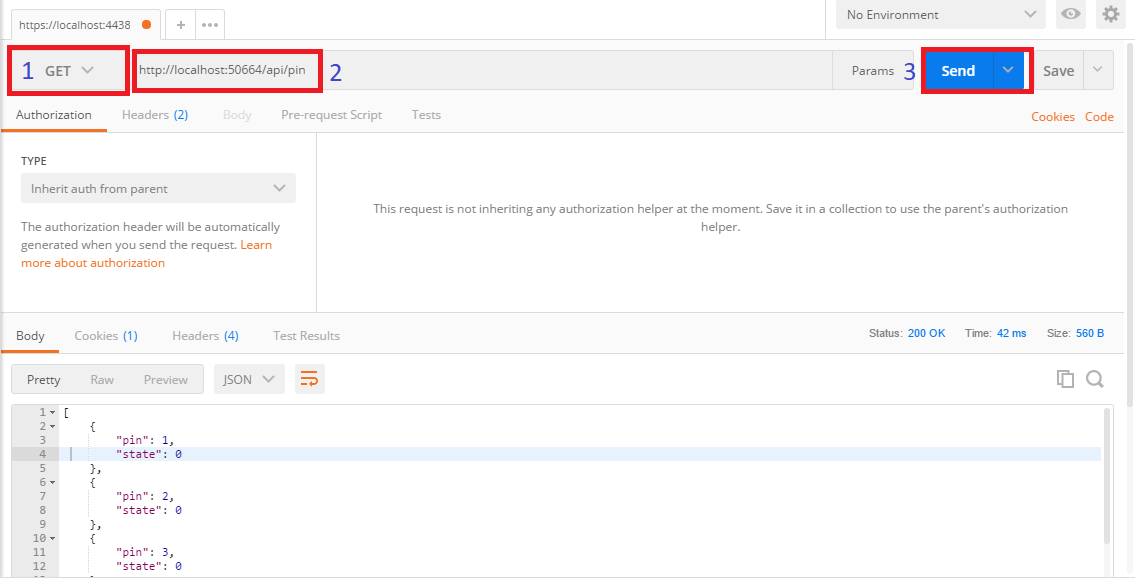
Hình 1.34: Run project bằng Command line

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

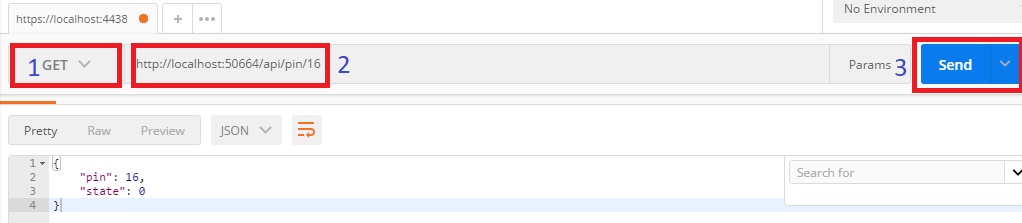


Hình 1.35: Kết quả sai khi chạy Server

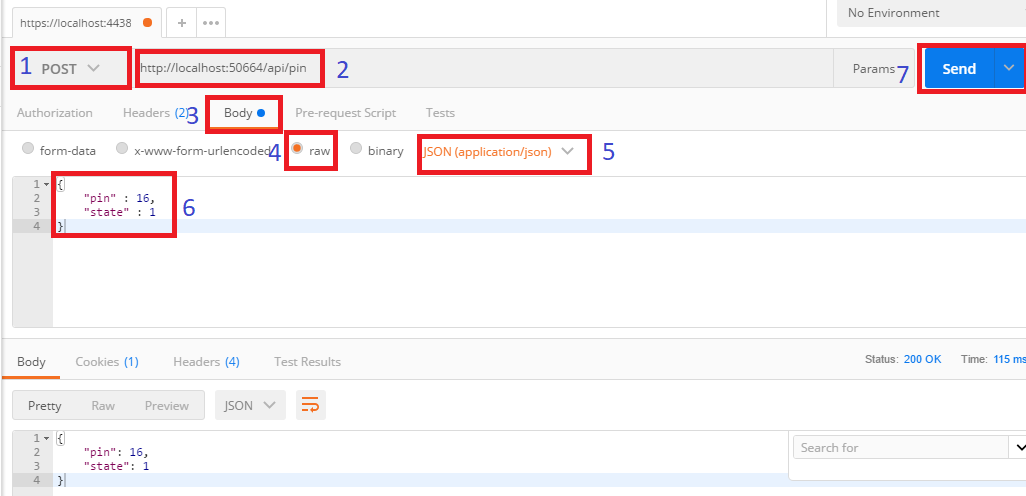
### Bước 13 : Test các hàm trong file PinController.cs bằng Postman



Hình 1.36: Action Get()



Hình 1.37: Action Get(int pin)



Hình 1.38: Action Post(Data dt)

1. Tương tự đối với UserController

## Ngrok

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

1. Mặc định thì Visual Studio sẽ chặn tất cả các kết nối , request từ bên ngoài ngoại trừ ứng dụng trên chính máy chạy Server . Vì thế khi bạn sử dụng điện thoại hoặc Arduino ESP8266 truy cập đến localhost của Server , mặc dù chung mạng LAN với máy chạy Server nhưng vẫn không kết nối, lấy dữ liệu từ Server được.
2. Để có thể kết nối dễ dàng thì ta sẽ sử dụng đến phần mềm tạo tunnel - Ngrok

### Tổng quan

1. 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
2. Ví dụ: mydomain.ngrok.io => localhost:80

### Đặc điểm của Ngrok

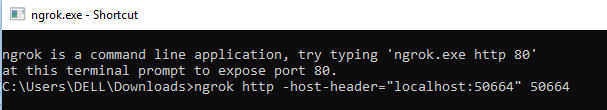
* Giúp bạn chạy demo dự án cho khách hàng xem từ chính máy của bạn mà không cần deploy lên server
* Bạn có thể test responsive trên mobile một cách dễ dàng thông qua URL mà ngrok cung cấp
* Xây dựng webhook tới localhost của bạn một cách dễ dàng
* Hỗ trợ http, https, tcp
* Hỗ trợ IP whitelist

### Cài đặt Ngrok

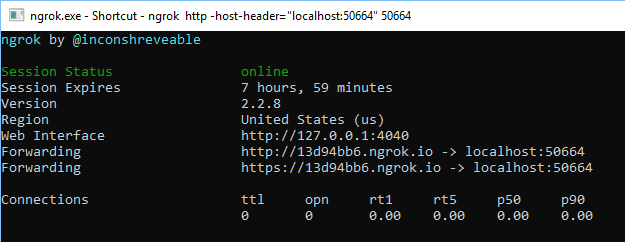
1. Link download : <https://ngrok.com/download>
2. Ngrok không cần cài đặt , chỉ cần download về và chạy file .exe

### Chạy Ngrok

1. Chạy file .exe và patse lệnh sau vào command line
2. “ngrok http -host-header="localhost:50664" 50664”
3. Lưu ý : Thay đổi port



Hình 1.39: Chạy Ngrok



Hình 1.40: Kết quả sau khi chạy Ngrok

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

### Limited connection

1. 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 1.41: Kết quả kiểm tra kết nối Ngrok trên điện thoại

### Kết luận

1. 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) .
2. Kết nối Server thông qua các thiết bị bên ngoài : điện thoại .
3. Phần tiếp theo ta sẽ kết nối đến Server thông qua Arduino ESP8266