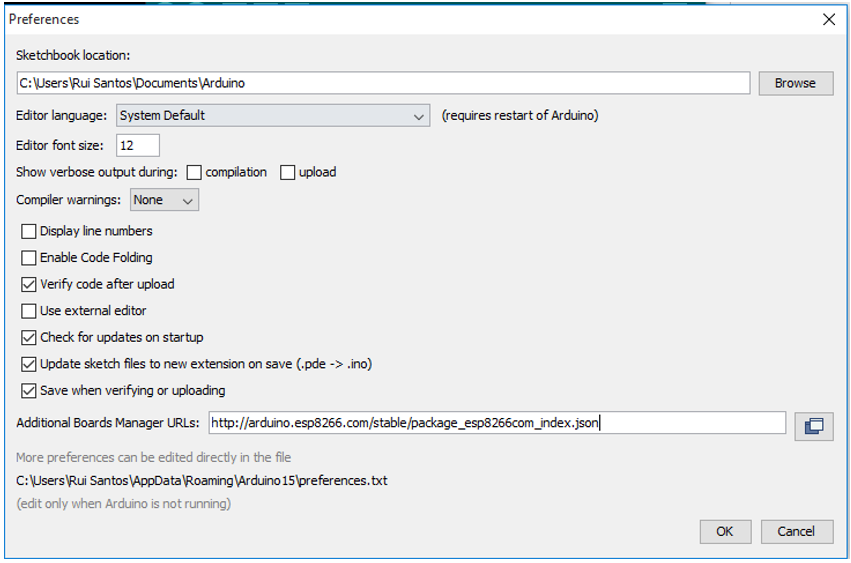
**Môi trường**

**Làm thế nào để lập trình cho Arduino ESP8266 bằng Arduino IDE?**

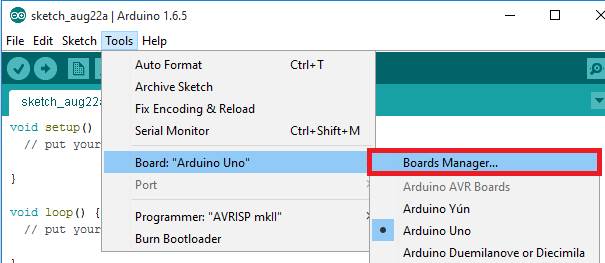
Vào **File→ Preferences**, vào textbox **Additional Board Manager URLs** thêm đường link sau vào

http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

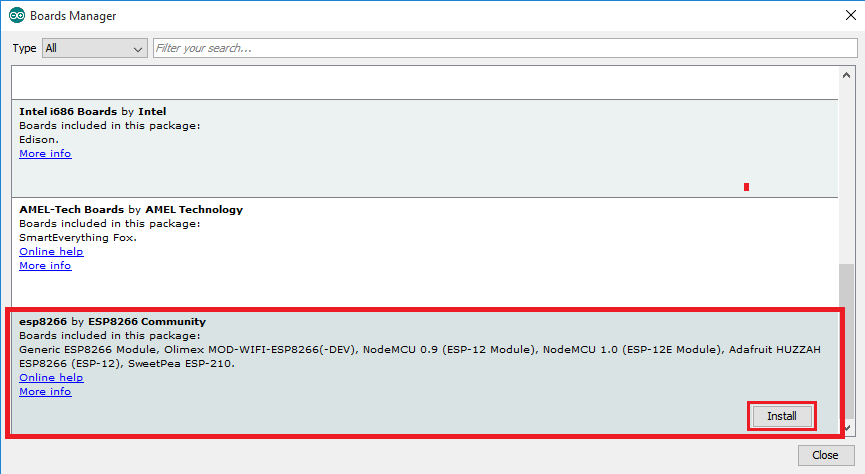
Click **OK** để chấp nhận.

[](http://k1.arduino.vn/img/2016/08/15/0/3077_123450-1471265967-0-arduino-ide-preferences.png)

Tiếp theo vào **Tool**→**Board**→**Boards Manager**

**[](http://k3.arduino.vn/img/2016/08/15/0/3036_812450-1471265965-0-arduino-ide-boards-manager.png)**

đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào **ESP8266 by ESP8266 Community**, click vào **Install**. Chờ phần mềm tự động download và cài đặt.

[](http://k2.arduino.vn/img/2016/08/15/0/3041_882450-1471265963-0-installing-esp-board.png)

Chọn Board để lập trình cho ESP8266:

Vào **Tool**→**Board**→Generic **ESP8266**  chọn cổng COM tương ứng với module **ESP8266**   tương ứng. Đối với các mạch khác như WeMos mini thì ta chọn WeMos D1 R2 & mini.

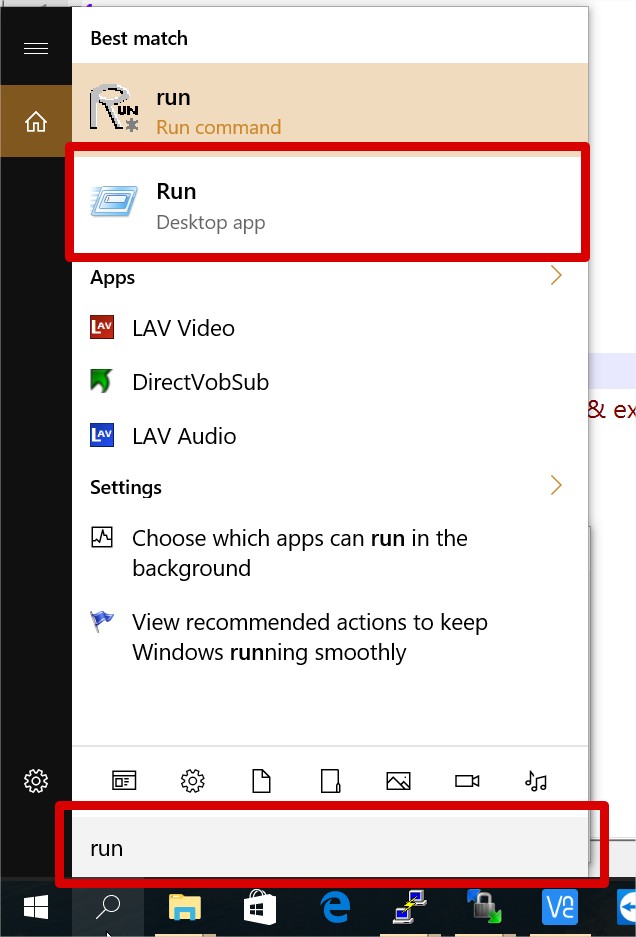
Ok, đến bước này là chúng ta đã có thể lập trình ESP8266 của bất cứ hãng nào.

**. Cài đặt NodeJS**

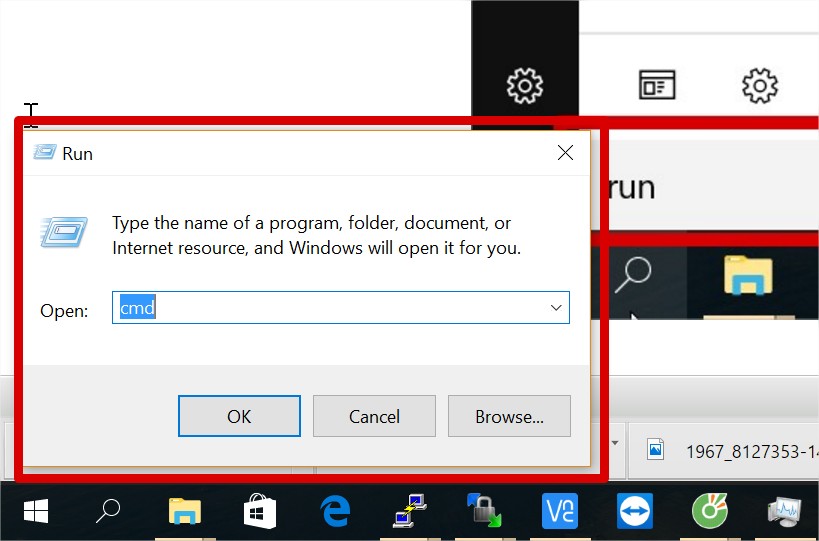
Để cài đặt NodeJS, bạn hãy đọc bài [Cài đặt NodeJS trên Windows - mô hình môi trường IOT chuyên nghiệp](http://arduino.vn/bai-viet/806-cai-dat-nodejs-tren-windows-mo-hinh-moi-truong-iot-chuyen-nghiep) nếu bạn muốn sử dụng máy tính Windows của bạn làm Socket Server. Nếu bạn cài đặt một mạch Raspberry Pi hay Orange Pi làm server thì hãy theo bài viết này [Lập trình nhúng trên Raspberry Pi bằng ngôn ngữ Javascript trong môi trường NodeJS - chìa khóa thành công](http://arduino.vn/bai-viet/888-lap-trinh-nhung-tren-raspberry-pi-bang-ngon-ngu-javascript-trong-moi-truong-nodejs-chia) của Raspi.

**SOCKET SERVER**

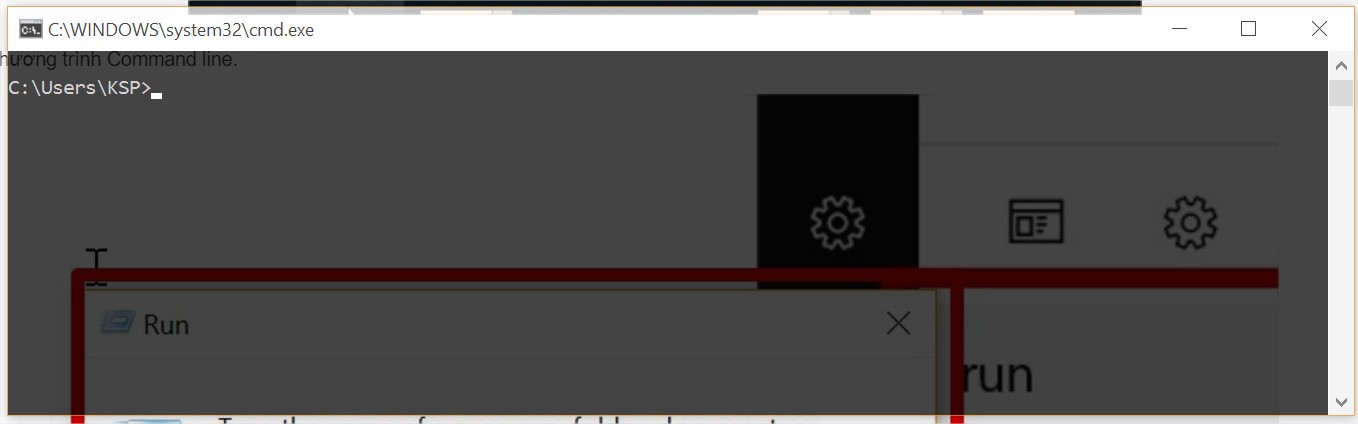
Bạn mở hộp thoại tìm kiếm của Windows và tìm chương trình **Run**như hình. Mục tiêu của bước này để vào Run để gõ các lệnh ở Command Line.

[](http://k2.arduino.vn/img/2017/02/26/0/3586_882450-1488100491-0-search.jpg)

Sau đó gõ lệnh **cmd**để bật chương trình Command line.

[](http://k3.arduino.vn/img/2017/02/26/0/3576_812450-1488100600-0-run.jpg)

Sau đó chương trình Command Line sẽ hiện ra, các bước trên chỉ với một mục đích duy nhất là giúp bạn mở được command line. Vì khi làm việc với NodeJS chúng ta không có IDE như với Arduino. Vì vậy, chúng ta dùng Command Line để khởi tạo các chương trình viết bằng Javascript trong môi trường NodeJS.

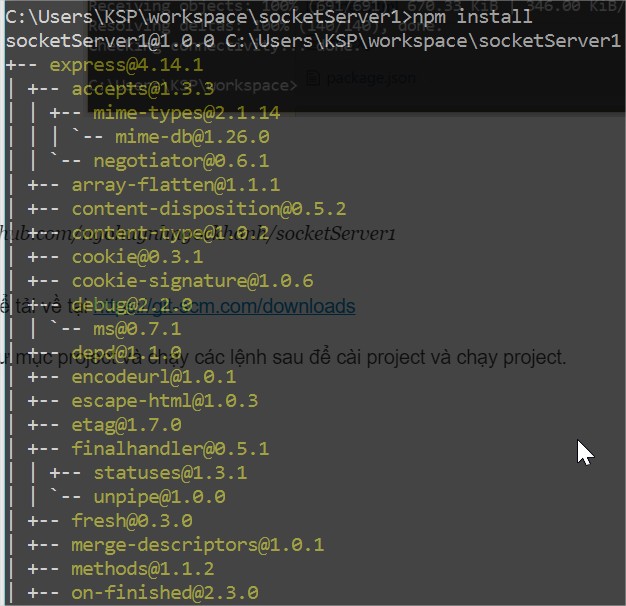
[](http://k1.arduino.vn/img/2017/02/26/0/3618_123450-1488100788-0-cwindowssystem32cmd.exe-.jpg)

Bạn sẽ tạo một thư mục tên là **workspace**và lưu các code liên quan đến NodeJS ở trong này. Các phiên bản tiếp theo mình cứ theo như trong này và làm việc.

Bạn copy folder socket server vào.

Nếu bạn chưa cài git, bạn có thể tải về tại [https://git-scm.com/downloads...](https://git-scm.com/downloads)

Sau đó, bạn hãy chuyển vào thư mục project và chạy các lệnh sau để cài project và chạy project.

[](http://k3.arduino.vn/img/2017/02/26/0/3577_812450-1488103375-0-cwindowssystem32cmd.exe-.jpg)

*npm install*

[http://k1.arduino.vn/img/2017/02/26/0/3619_123450-1488104305-0-select-cwindowssystem32cmd.exe---node-index.js-.jpg](http://k1.arduino.vn/img/2017/02/26/0/3619_123450-1488104305-0-select-cwindowssystem32cmd.exe---node-index.js-.jpg)

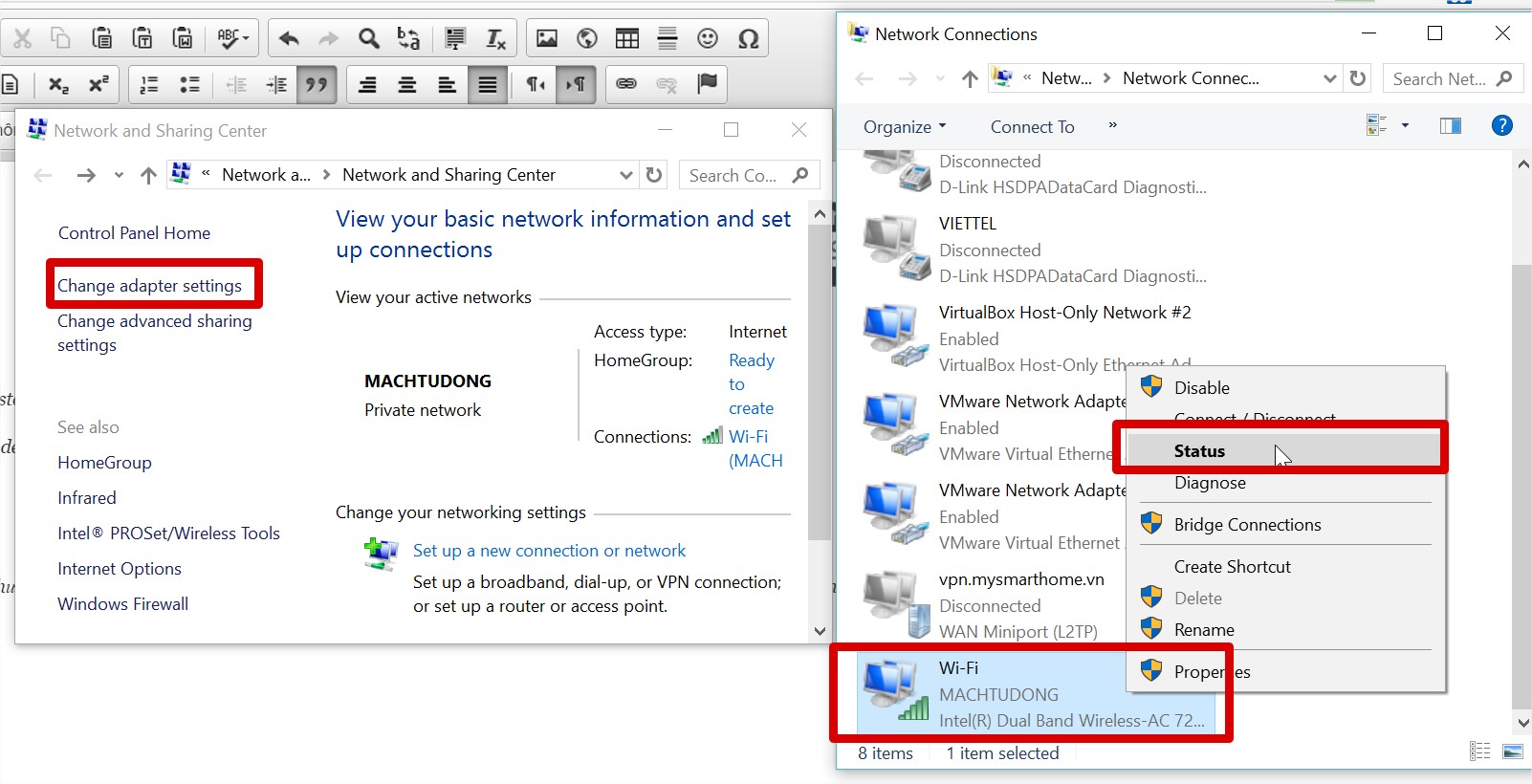
*node index.js*

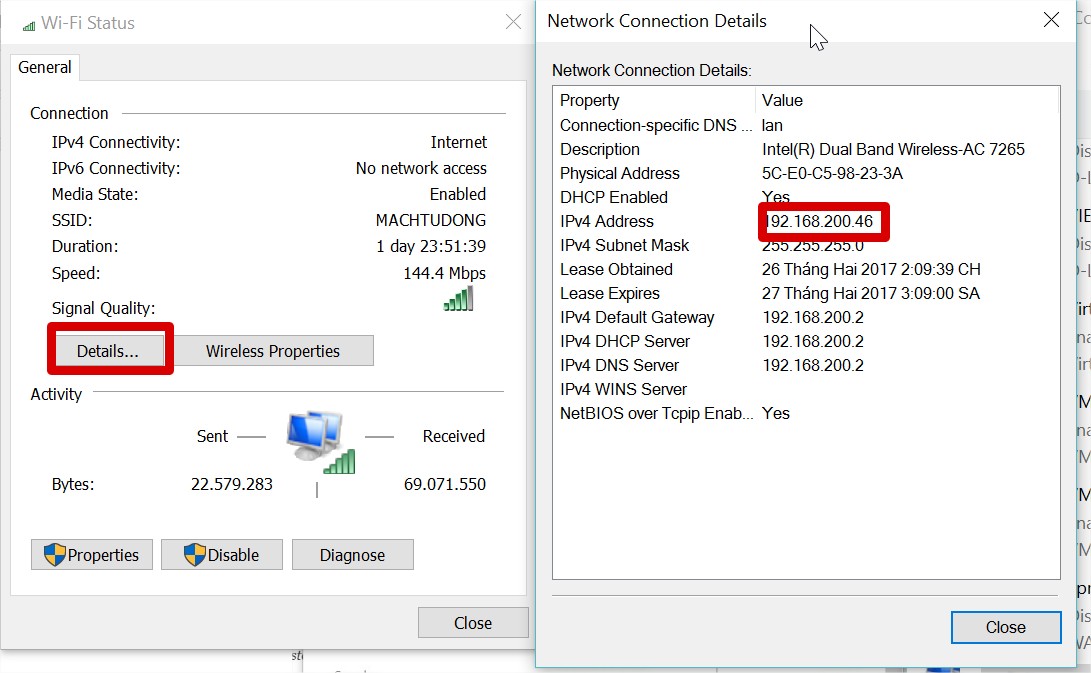
npm install #cài đặt socketServer

node index.js #Chạy chương trình index.js

**Chú ý**

Nếu chương trình của bạn chạy ra địa chỉ IP 169.254.x.x thì trên máy tính của bạn đang có những card mạng khác và bạn không thể sử dụng địa chỉ IP này cho các ví dụ phía sau. Nói chung, bạn hãy theo các bước sau để xem địa chỉ IP của card wifi trên máy tính để cho kết quả chính xác.

[](http://k2.arduino.vn/img/2017/02/26/0/3588_882450-1488104617-0-network-connections.jpg)

[](http://k3.arduino.vn/img/2017/02/26/0/3578_812450-1488104690-0-network-connection-details.jpg)

*Như vậy, kết quả địa chỉ IP trong mạng LAN (Wifi) của chúng ta đó là 192.168.200.46*

Nói tóm lại, chúng ta đã tạo một dịch vụ mạng được mở ở port 3434 ở địa chỉ ip 192.168.200.46 (địa chỉ ip này sẽ khác trên máy tính của bạn, hãy thay đổi nó phù hợp trong trường hợp của bạn).

Như vậy, bạn đã chạy thành công socket Server đầu tiên trên máy tính của mình.

Chúng ta sẽ dùng them khái niệm namespace của thư viện socket.io.

**ESP8266**

 Nhiệm vụ của chúng ta bây giờ là xây dựng một Socket Client trên con ESP8266 để nó có thể liên lạc với Socket Server. Và thật may mắn, chúng ta đã có thư viện Socket.io-v1.x-Library của [washo4evr](https://github.com/washo4evr). Tuy nhiên, mình không ưng ý lắm về các biến toàn cục xử lý của anh này. Vì vậy, mình đã chỉnh sửa lại một xíu, bao gồm:

* Thêm #define DEBUG (mặc định là false), để các lệnh DEBUG (Serial.println) trong thư viện không xuất hiện ra ngoài màn hình Serial Monitor.
* Thêm biến toàn cục Rfull chứa toàn bộ nội dung chuỗi JSON để xử lý sau, thay vì cắt ra luôn thành 3 biến toàn cục như các ví dụ trước đây của anh ta.

Và bây giờ mình sẽ hướng dẫn các bạn xây dựng một Socket Client trên con ESP8266 nhỏ nhắn này. Nhiệm vụ của nó rất đơn giản:

* Kết nối tới Socket Server.
* Nếu nó bị ngắt kết nối thì tự động kết nối lại.
* Cứ sau mỗi 2 giây sẽ gửi lệnh atime lên Socket Server để phá.
* Nếu nhận bất cứ lệnh gì từ socket server thì in ra console để xem chơi cho vui.

Và đây là đoạn code chương trình và lưu đồ thuật toán. Để chương trình có thể chạy được, bạn hãy tải về thư viện Socket.io-v1.x-Library của mình [tại đây](https://github.com/ngohuynhngockhanh/Socket.io-v1.x-Library) ([mirror](https://github.com/ngohuynhngockhanh/Socket.io-v1.x-Library/archive/master.zip)).

Còn SoftwareSerial giúp chúng ta tạo một cổng Serial thứ hai. Vì sao lại cần cổng Serial thứ hai này? Vì trên mỗi tầng (Arduino và ESP8266) thông thường chỉ có một cổng Serial phần cứng (Arduino có một cổng, ESP8266 có 1 cổng đầy đủ và một cổng gửi (chỉ TX)). Những cổng Serial này được dùng để nạp code. Vì vậy, nếu trong quá trình nạp code mà các cổng này nối với thiết bị ngoại vi, ví dụ như: nối Arduino với ESP8266 hay nối với con LED, thì sẽ không nạp code được mà phải ngắt kết nối với các thiết bị ngoại vi, sau đó mới kết nối được. Vì vậy, một cổng Serial là không đủ cho dân chơi chúng ta! Và thật vui, khi Cộng đồng nguồn mở đã tạo ra một thư viện mang tên SoftwareSerial để chúng ta có thể tạo ra nhiều ứng dụng thú vị như dự án này chẳng hạn. Và ở mỗi bên, chúng ta sẽ tạo ra một cổng Serial ảo, từ đó dùng cổng này giao tiếp với Module còn lại.

Nhiệm vụ của thư viện SerialCommand đó là cho phép chúng ta định nghĩa các chuỗi gửi đi. Tuy nhiên, ở thư viện SerialCommand gốc, nó có nhiều hạn chế, như:

* Chiều dài mỗi lệnh chỉ có 16 mà thôi. Vượt quá cái con số 16 này thì bỏ qua lệnh đó luôn devil. Trong khi đó ta lại gửi JSON mà, mà JSON thì độ dài phải thông thoán tầm vài chục đến vài trăm ký tự mới vui! Chứ 16 ký tự thì làm JSON ngán lắm ahihi. Nhiều bạn sẽ nghĩ đến việc vô file header và tăng buffer lên! Nhưng hãy bình tĩnh nào, nếu bạn tăng lên buffer lên 100 thì chúng ta sẽ vứt đi (100-16) \* 10 = 840 byte vào vùng nhớ stack đó. Thử tăng lên 255 thì bạn sẽ thấy độ bựa của việc tăng này. Bản chất của việc này, đó là tác giả xem lệnh bằng: **lệnh + danh sách các biến (nhiều biến cách nhau bằng giấu cách)**. Còn mình thì xem lệnh **là tên lệnh** mà thôi, còn danh sách biến nằm riêng, ngoài ra tất cả tham số được nén vô chuỗi JSON, và chuỗi này lưu trữ danh sách biến rồi. Dẫn đến thư viện cũ không còn phù hợp nữa.
* Tối đa 10 lệnh. Cái này có thể dễ dàng chỉnh sửa trong file header.

Vì vậy, mình đã chỉnh sửa lại thư viện SerialCommand cho phù hợp với nhu cầu của mình trong dự án này. Và mình chia sẻ nó với các bạn, các bạn nào đang dùng thư viện SerialCommand cũ thì xóa đi, rồi cài thư viện mới của mình vô. Sau đây là cải tiến của thư viện:

* Lệnh và danh sách các biến được chia ra làm hai phần khác nhau. Lệnh là một chuỗi có độ dài tối đa 10 ký tự (tính luôn ký tự \0 thì còn 9 ký tự).
* Danh sách các biến chỉ là một chuỗi JSON đã được mã hóa.
* Cú pháp được thay đổi từ: <Lệnh>(dấu cách)<danh sách các biến, cách nhau bởi dấu cách>\r
  + ​thành: <Lệnh>\r<Chuỗi JSON mã hóa các biến>\r
  + Xem thêm ở ví dụ đầu tiên, bạn sẽ hiểu ngay vấn đề đó mà.
* Mình chỉnh thư viện mặc định có thể chơi đến 16 lệnh. Các bạn có thẻ tăng hoặc giảm con số này khi bị báo lỗi không thể thêm lệnh được nữa. Mà mình thấy vầy là ok rồi ahihi.

Một số vấn đề mà mình thấy có thể xảy ra nhưng nếu bạn nối dây ngon lành thì nó sẽ không xảy ra. Nối dây ngon lành ở đây có nghĩa là bạn đã nối RX của mạch nhận vào TX của mạch gửi. Vậy là ok :D.

* Nếu bị lỏng dây mà chuỗi json chưa đến mà lệnh sau đã đến => Lỗi. Nói chung ít khi xảy ra, mắc dây nối ổn là ok :).
* Nếu bị tràn bộ đệm do chuỗi JSON quá dài thì coi như lệnh đó bị bỏ qua và không xử lý. Để khăc phục, bạn hãy tăng ARGUMENTBUFFER trong file header là ổn.

Để tiếp tục, bạn hãy tải về 2 thư viện nhé.

* [SerialCommand](https://github.com/ngohuynhngockhanh/ArduinoSerialCommand) ([mirror](https://github.com/ngohuynhngockhanh/ArduinoSerialCommand/archive/master.zip)) (bản mới mình đã chỉnh sửa, nếu bạn đã có rồi thì hãy xóa và cập nhập lại)
* SoftwareSerial (thông thường các bản Arduino 1.6.x trở lên đều có thư viện này)
* [ArduinoJson](https://github.com/bblanchon/ArduinoJson) ([mirror](https://github.com/bblanchon/ArduinoJson/archive/master.zip))

Nhớ nhập ip port của socket server

Tới đây là có thể nạp code cho esp8266 ,cho chắc thì căm gpio 0 ,15 xuống gnd để nạp.

**UNO**

Nạp code cho uno.

Note::

Khi code được nạp vào các mạch cắm như sơ đồ fritzing. Nếu esp8266 kết nối đc vào sẽ bắn ra log. Lâu quá thì reset.

Đảm bảo tất cả các thiết bị trong 1 giải ip

Reference:

<http://arduino.vn/bai-viet/1496-esp8266-ket-noi-internet-phan-1-cai-dat-esp8266-lam-mot-socket-client-ket-noi-toi>

<http://arduino.vn/bai-viet/1497-esp8266-ket-noi-internet-phan-2-arduino-gap-esp8266-hai-dua-noi-chuyen-bang-json>

<http://arduino.vn/bai-viet/1511-esp8266-ket-noi-internet-phan-3-arduino-gap-smartphone-hai-dua-noi-chuyen-bang-json>

ESP :

<http://hshop.vn/products/mach-thu-phat-wifi-esp8266-v1>

<https://xivila.com/profile/sangseu@gmail.com/blog/197/esp8266-phan-2-lap-trinh-esp8266-bang-arduino-ide/>