Arduino core for ESP8266 WiFi chip

This project brings support for ESP8266 chip to the Arduino environment. It lets you write sketches using familiar Arduino functions and libraries, and run them directly on ESP8266, no external microcontroller required.

ESP8266 Arduino core comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, work with SD cards, servos, SPI and I2C peripherals.

Contents

Installing options:

Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

Install Arduino 1.6.5 from the Arduino website.

Start Arduino and open Preferences window.

Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into *Additional Board Manager URLs* field. You can add multiple URLs, separating them with commas.

Open Boards Manager from Tools > Board menu and install *esp8266* platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

The best place to ask questions related to this core is ESP8266 community forum: http://www.esp8266.com/arduino. If you find this forum or the ESP8266 Boards Manager package useful, please consider supporting it with a donation.

Available versions

Stable version

Boards manager link: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Documentation: http://esp8266.github.io/Arduino/versions/2.1.0/

Staging version

Boards manager link: http://arduino.esp8266.com/staging/package_esp8266com_index.json

Documentation: http://esp8266.github.io/Arduino/versions/2.1.0-rc2/

Using git version

Install Arduino 1.6.7

Go to Arduino directory

Clone this repository into hardware/esp8266com/esp8266 directory (or clone it elsewhere and create a symlink)

cd hardware

mkdir esp8266com

cd esp8266com

git clone https://github.com/esp8266/Arduino.git esp8266

Download binary tools (you need Python 2.7)

cd esp8266/tools

python get.py

Restart Arduino

Using PlatformIO

PlatformIO is an open source ecosystem for IoT development with cross platform build system, library manager and full support for Espressif (ESP8266) development. It works on the popular host OS: Mac OS X, Windows, Linux 32/64, Linux ARM (like Raspberry Pi, BeagleBone, CubieBoard).

What is PlatformIO?

PlatformIO IDE

Quick Start with PlatformIO IDE or PlatformIO CLI

Advanced using - custom settings, uploading to SPIFFS, Over-the-Air (OTA) or using stage version

Integration with other IDE - Atom, CLion, Eclipse, Emacs, NetBeans, Qt Creator, Sublime Text, VIM and Visual Studio

Project Examples

Building with make

makeEspArduino is a generic makefile for any ESP8266 Arduino project. Using make instead of the Arduino IDE makes it easier to do automated and production builds.

Documentation

Documentation for latest development version:

Reference

Libraries

File system

OTA update

Supported boards

Change log

Issues and support

If you encounter an issue, you are welcome to submit it here on Github: https://github.com/esp8266/Arduino/issues. Please provide as much context as possible: version which you are using (you can check it in Boards Manager), your sketch code, serial output, board model, IDE settings (board selection, flash size, etc).

If you can not find the answers above, you can also try ESP8266 Community Forum

Contributing

For minor fixes of code and documentation, go ahead and submit a pull request.

Check out the list of issues which are easy to fix — easy issues for 2.2.0. Working on them is a great way to move the project forward.

Larger changes (rewriting parts of existing code from scratch, adding new functions to the core, adding new libraries) should generally be discussed in the chat first.

Feature branches with lots of small commits (especially titled "oops", "fix typo", "forgot to add file", etc.) should be squashed before opening a pull request. At the same time, please refrain from putting multiple unrelated changes into a single pull request.

License and credits

Arduino IDE is developed and maintained by the Arduino team. The IDE is licensed under GPL.

ESP8266 core includes an xtensa gcc toolchain, which is also under GPL.

Esptool written by Christian Klippel is licensed under GPLv2, currently maintained by Ivan Grokhotkov:https://github.com/igrr/esptool-ck.

Espressif SDK included in this build is under Espressif MIT License.

ESP8266 core files are licensed under LGPL.

SPI Flash File System (SPIFFS) written by Peter Andersson is used in this project. It is distributed under MIT license.

umm_malloc memory management library written by Ralph Hempel is used in this project. It is distributed under MIT license.

ARDUINO Y WIFI ESP8266

Conectado Arduino a las redfes WIFI

Home Arduino Y WIFI ESP8266

OBJETIVOS

- Presentar el **módulo WIFI ESP8266.**
- Describir sus posibilidades.
- Presentar un circuito de prueba para conectarlo a nuestros Arduinos.
- Empezar con los comandos AT que acepta.
- Montarun pequeño servidor Web a través de WIFI..

MATERIAL REQUERIDO.

Arduino UNO o equivalente.
Una Protoboard .
Algunos cables de protoboard, preferiblemente Dupont macho/hembra.
Un módulo WIFI **ESP8266**

ARDUINO Y LAS CONEXIONES WIFI

A medida que te acostumbras a la idea de que puedes conectar tu Duino al mundo exterior vía una conexión inalámbrica, seguir usando cables se hace un poco cuesta arriba, da como pereza.

Ya hemos visto cómo usar una primera opción de conexión sin hilos mediante Bluetooth en las sesiones previas, pero antes o después llegaremos a querer usar **WIFI** para esto.

Las razones son fáciles de entender. Las WIFIs mezclan la comodidad de uso de las conexiones inalámbricas Bluetooth, con un mayor alcance y por si fuera poco, muy probablemente tengas un acceso WIFI en casa o en el trabajo y listo para conectarte a el.

Así, pues, parece que ha llegado el momento de comprar un **Shield WIFI** similar al shield Ethernet. Acostumbrados, como estamos, a que las cosas en Arduino sean baratitas, nos acercamos a nuestra página de compra preferida y nos llevamos un buen susto.

No hay modo de comprar un shield WIFI por menos de 50€ y los hay hasta de 80€. ¿Qué está pasando aquí?

Bueno, la verdad es que no está muy claro, pero mientras que puedes comprar un adaptador USB WIFI para tu PC por menos de lo que vale una entrada de cine, cuando hablamos de Arduino, los precios se disparan.

Puede ser que desarrollar la tarjeta sea cara, especialmente el software o simplemente que han visto la ocasión de hacer un negóciate con nosotros, pero sea como sea es lo que hay.

Por eso yo decidí que por ahora de **Shields WIFI** ni hablar a ese precio y me puse a ver que había por ahí que me pudiera servir y mira por donde hay cosas de precio mucho más atractivo si estás dispuesto a aceptar ciertas limitaciones.

Y en cuanto empieces a buscar algo de WIFI barato te encontrarás con el módulo **WIFI ESP8266**, que es algo muy parecido a los módulos Bluetooth de las sesiones previas de comunicaciones, y que al igual que ellos incluye toda la electrónica necesaria para la comunicación Radio Frecuencia en la banda WFI, así como la pila TCPIP y que se comunica con nosotros a través de un puerto serie.

De hecho, exactamente igual que los modos HC-06 y HC-05 se gobierna mediante comandos AT, algo que ya no tiene secretos para los seguidores de estas sesiones, y todo por un precio similar al de los Bluetooth.

Así, que vamos a ver como usamos estos interesantes módulos.

CONECTANDO EL MÓDULO ESP8266

Lo primero es decir que este es modulo muy sencillo y diseñado desde el principio con la Internet of Things en mente (IOT), y por eso incluye todo lo necesario para conectarse a un punto de acceso WIFI mediante **comandos de texto AT**, vía una puerta serie, que puede ser configurada a diferentes velocidades.

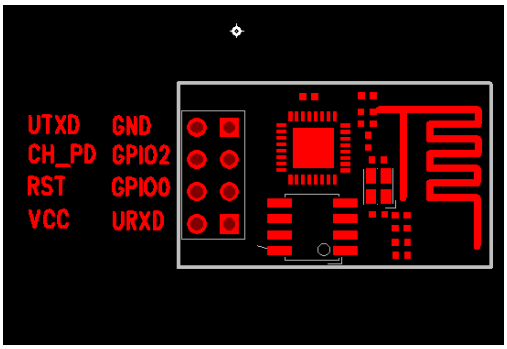
Una vez que le instruimos para que se conecte a nuestra WIFI, el modulo es capaz de enviar información que le remitimos vía la puerta serie a una dirección IP y puerto que deseemos.

Cuando se trata de recibir, limpia todo el empaquetado TCPIP y nos reenvía por la puerta serie la información de datos limpia de polvo y paja, con lo que tiene la enorme virtud de permitirnos olvidarnos de la gestión del TCPIP y de las demandas de procesador y memoria que suponen.

A cambio no es exactamente una conexión WIFI, porque no tenemos acceso al stack o al socket IP pero para el Arduino esto es casi una ventaja.

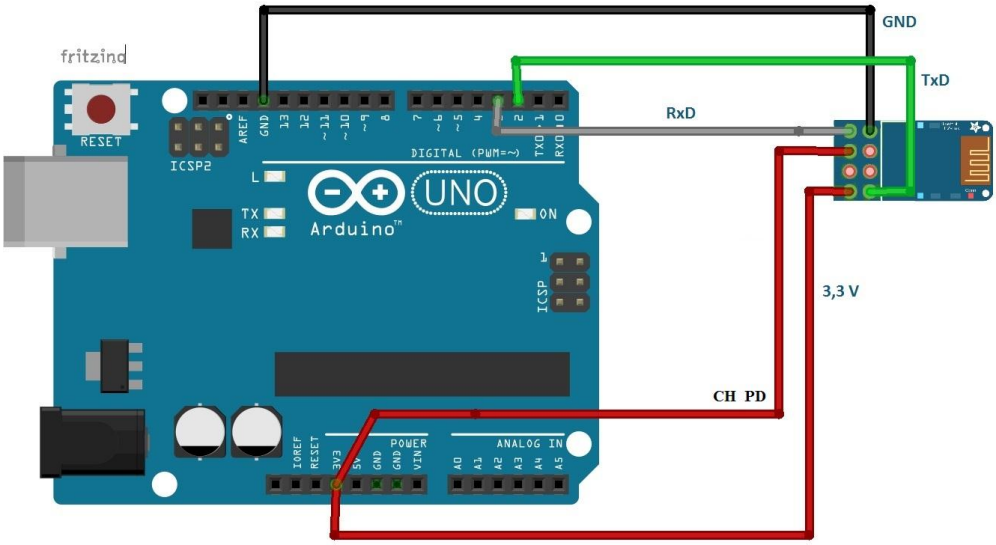
- *De hecho el **módulo ESP8266** incluye un pequeño procesador interno que podríamos programar para funcionar de modo autónomo y que incluso dispone de un par de puerto GPIO (General Purpose Input Output) para su uso como activador de algo, pero esto e una historia para otro día.*

Veamos como conectarlo a nuestro Arduino, el patillaje del módulo visto desde la parte superior donde se puede observar la antena integrada, es así:



La fuente interna de 3.3V del Arduino da un máximo de 50 mA, cuando el consumo del módulo suele ser en el arranque bastante superior a esto, lo que le llevara a unos arranques poco fiables, y aunque se acaba consiguiendo, deben repetirse una y otra vez (aunque naturalmente el modulo sufrirá).

Si disponéis de una fuente externa de alimentación de 3.3V no dudéis en usar la para alimentar este módulo ESP8266. Se inicia como una seda, mientras que si no disponéis de ella, el montaje que indico a continuación os permitirá hacer pruebas, lo he comprobado, pero os costará arrancar.



- *Las hojas de normas del* ***módulo WIFI ESP8266*** *especifica que debe ser alimentado a 3,3 Voltios y no recomienda conectarle 5V directamente so pena de quemarlo.*

- *Menos claro está el hecho de que RXD y TXD deban ser a 3,3 V. He visto varias notas por Internet recomendando montar un divisor de tensión en cada pin para evitar problemas, pero yo los he conectado directamente a Arduino sin problemas (Me compré dos, por si acaso, son muy baratos).*
- *Según la versión de firmware que incluya el modulo, el pin RST debe o no ser conectado a tensión (3,3V) para poder activar el uso del módulo. El que yo he recibido, desde luego no se activa sin él, pero he leído que hay versiones más antiguas que no lo necesitan*.

En lo que he mirado por Internet, estamos de nuevo con la historia de conectar este módulo a los pines digital 0 y 1 del Arduino para aprovechar la conexión serie hardware, porque dicen que la conexión mediante la librería SotwareSerial es problemática a 115.200 baudios.Problema que yo al menos nunca he detectado

Yo soy demasiado vago para andar conectando y desconectando el modulo, cada vez que quiero reprogramar mi Duino, así que he hecho pruebas y a mi este montaje me funciona sin problemas, así que es el que os lo recomiendo.

- *Como norma general, y mientras no se demuestre lo contrario, en estas páginas nunca conectaremos las líneas de comunicación serie a los pines 0 y 1 del Arduino, siempre intentaremos usar otros pines y dejar estos libres para la comunicación vía USB.*

PRIMEROS COMANDOS AT CON ESP8266

Aunque al principio los modulos ESP8266 venian programados a una velocidad de comunicacion de 9.600 ultimamente estan viniendo a 115.200 y por eso convendria que supongais esa velocidad de entrada e ir bajando si es necesario.

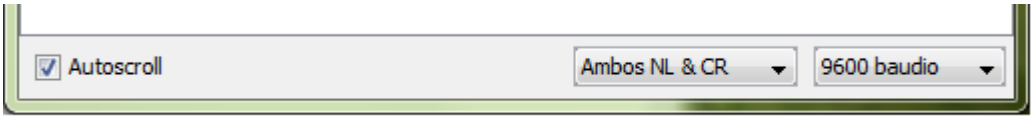
Ojo: Los Arduinos UNO no van muy finos a 115200 con la libreria serie, usad un MEGA o DUE si podeis, porque hay muchos errores de transmision en caso contrario

Vamos a volcar a nuestro Arduino el siguiente programita, que heredamos de la conexión Bluetooth, que simplemente vuelca la consola al puerto serie y viceversa:

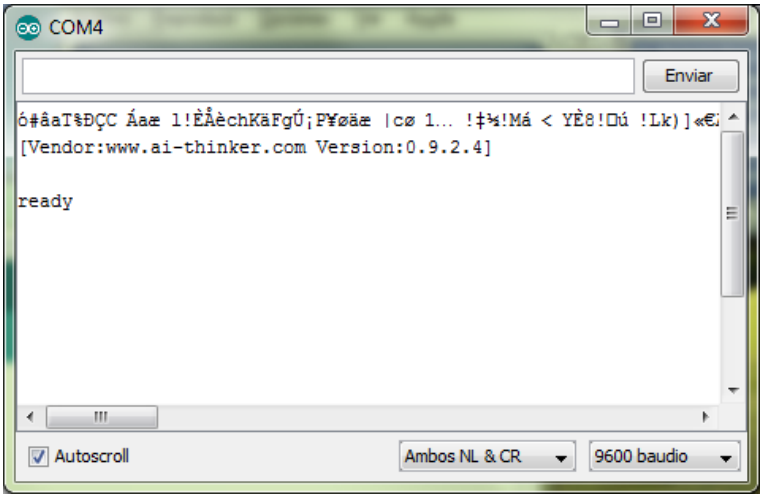
```
#include <SoftwareSerial.h>

SoftwareSerial BT1(3, 2); // RX | TX


void setup()
 {  Serial.begin(115200);
    BT1.begin(115200);
 }
```

```
void loop()
 {  String B= "." ;
    if (BT1.available())
       { char c = BT1.read() ;
         Serial.print(c);
       }
    if (Serial.available())
       {  char c = Serial.read();
          BT1.print(c);
       }
 }
```

Abrid ahora la consola y aseguraros de que enviamos seleccionamos ambos en la terminación de línea. Además en mi caso la comunicación venia definida a 115200 baudios de fábrica, por lo que en principio conviene que lo probéis así.



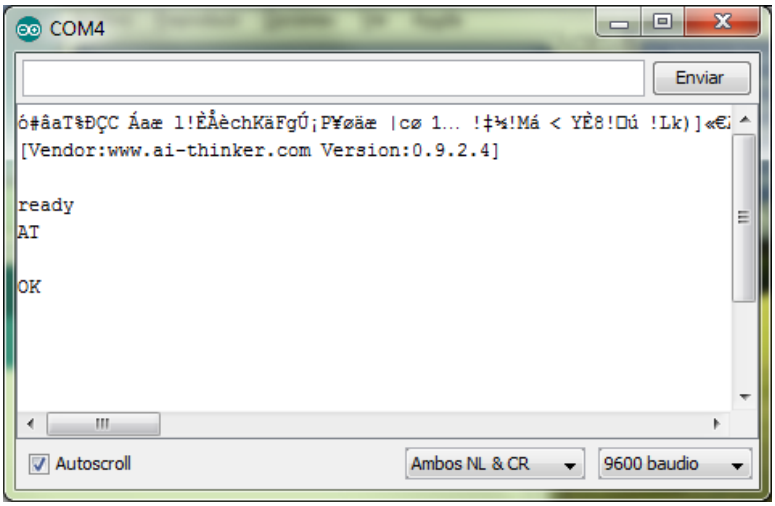
Si con la conexión que hemos descrito en el apartado anterior hay veces que el ESP8266 parece que no arranca. Soltad la alimentación del módulo, y volved a conectarla al cabo de uso segundos. Deberíais ver algo así:



Si no tenéis el mensaje de ready, repetid el proceso. A mí me ha costado varias veces en algún momento.

- *No os preocupéis por toda esa basurilla que sale antes de la versión del Soft. Parece que es normal*.

Vamos con nuestro primer **comando AT,** simplemente pedir atención: AT + [Intro]:

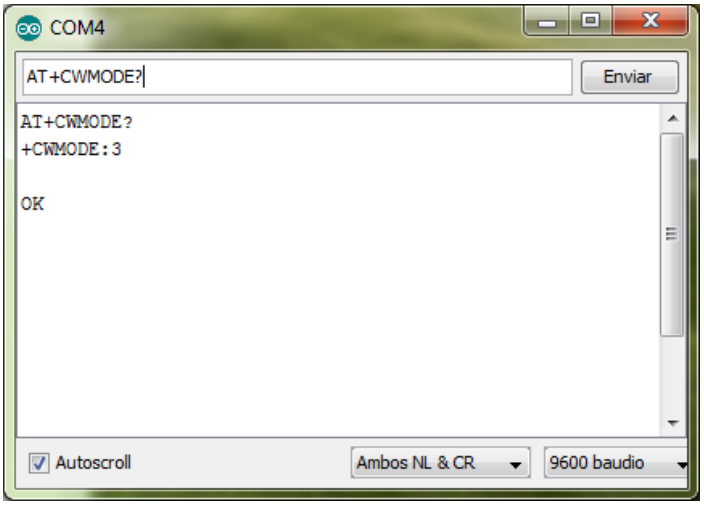


El modulo responde con un sencillo OK, para indicar que tenemos line abierta. En caso negativo probad a cambiar la velocidad de transmisión hasta que recibáis un mensaje legible.

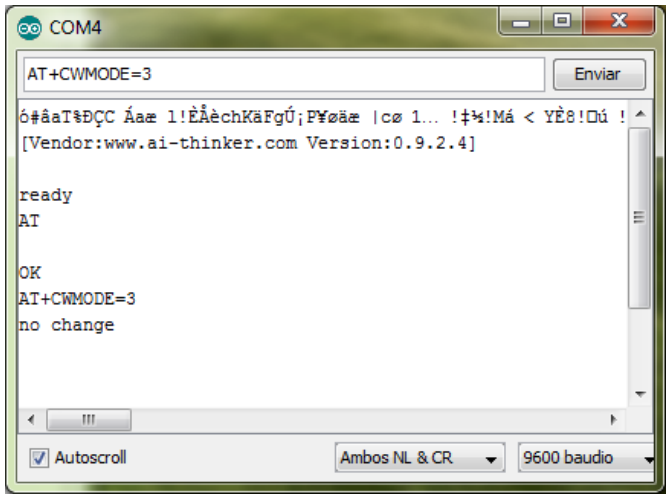
Para resetear el modulo probad AT+RST

No he encontrado una descripción muy clara de los modos de funcionamiento pero se puede cambiar la instrucción AT+CWMODE=n, donde n es 1,2 o 3. Para saber en qué modo estáis:

AT+CWMODE?

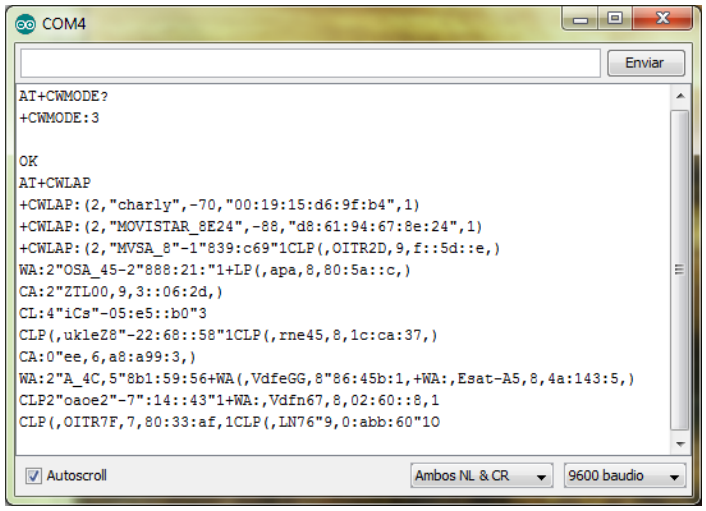


Aparentemente la buena es la 3: Probad



En mi caso responde que “No change” porque ya le había dado esta instrucción antes.

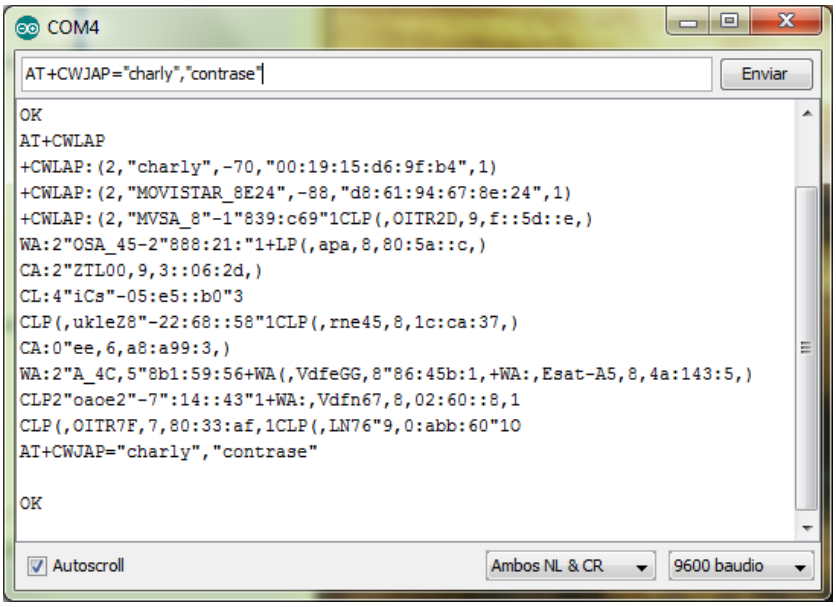
Vamos ahora a ver qué puntos de acceso WIFI tenemos en las inmediación es:



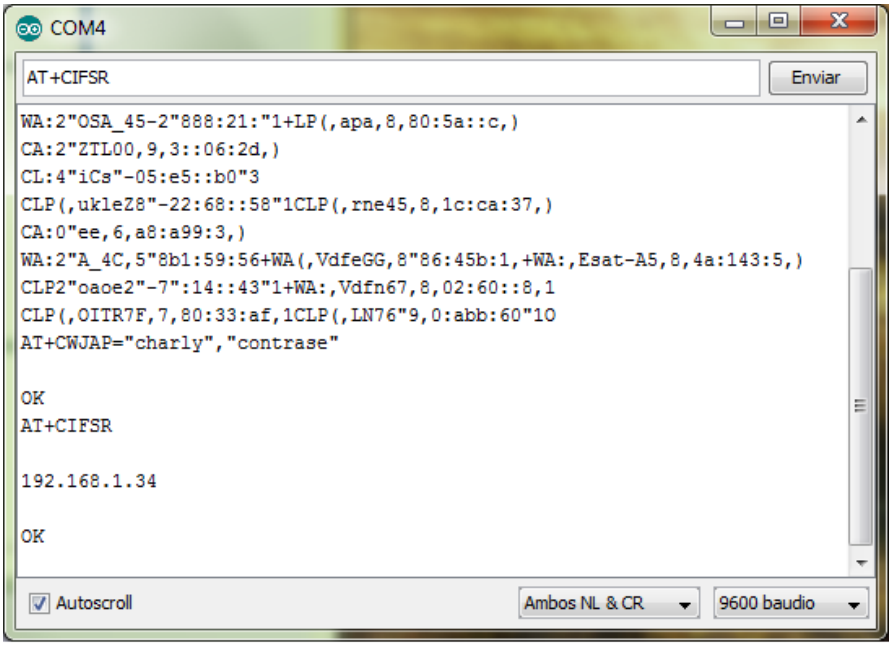
Al principio del mensaje indica 3 redes disponibles, Charly (la mía) mas MOVISTAR_8E24 y MVSA_8, a la que podríamos conectarnos.

Para conectarme a mi Router, necesito como siempre el nombre SSID que publica el punto de acceso (En mi caso Charly) y la contraseña de uso. La instrucción a usar es:

AT+CWJAP=”charly”,”contrase”



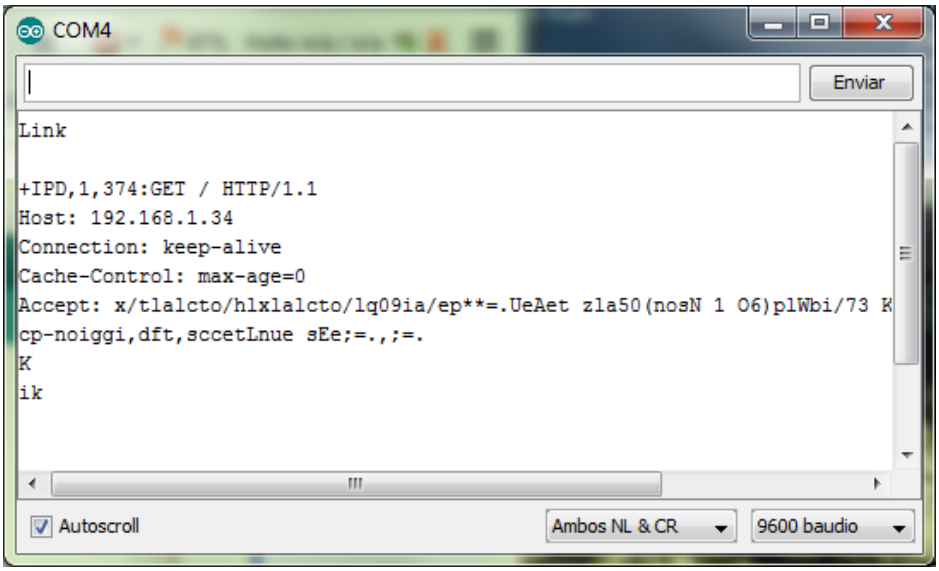
Al cabo de un momento si no hay problemas, responde con un OK. Y para ver que IP nos ha asignado hacemos: AT+CIFSR



Que en mi caso es la 192.168.1.34. Vamos a empezar a jugar con algo un poco más interesante. Prueba con:

AT+CIPMUX=1
AT+CIPSERVER=1,80

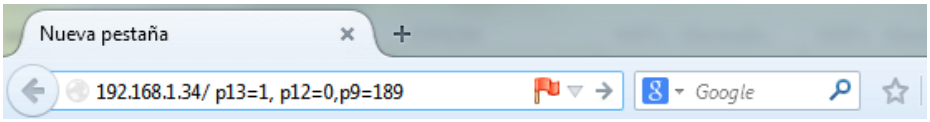
El primero habilita múltiples conexiones simultaneas, y el segundo arranca un servicio web (con el número de servicio = 1) en el puerto 80. Si ahora vas a tu navegador y escribes la dirección IP de tu módulo ESP8266, recibirás en la consola los mensajes correspondientes a la conexión:



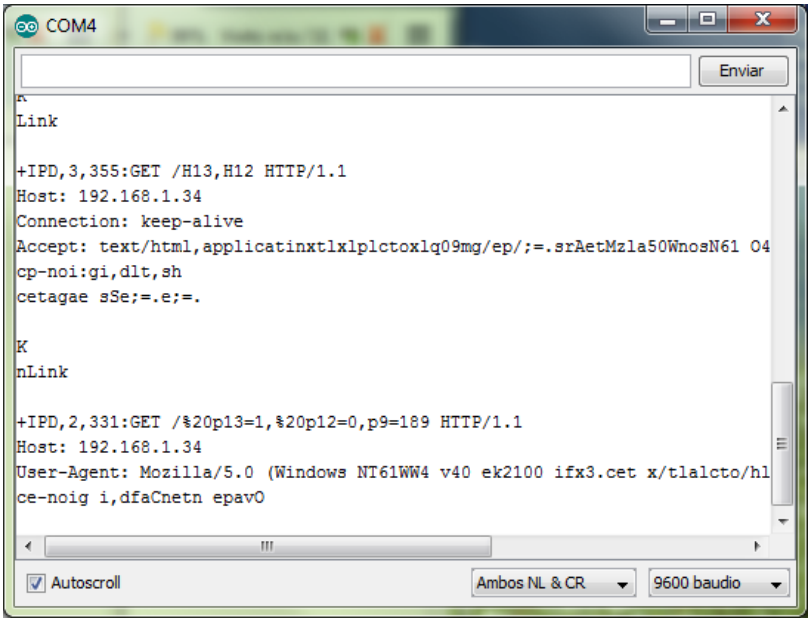
Acabamos de montar un pequeño servidor web con unas pocas instrucciones AT, no está mal.

¿Podríamos usarlo para enviar órdenes a nuestro Arduino como hacíamos con el BlueTooth, o con la tarjeta Ethernet? Me imagino que ya sabéis la respuesta.

Hay una forma muy fácil de pasar parámetros a nuestro Arduino, directamente desde el navegador, sin más que pasarle los parámetros después de la dirección IP, como por ejemplo:



Y lo que recibe la consola es:



Fíjate que en la 4ª línea empezando por abajo, pone:

+IPD,2,331:GET /%20p13=1,%20p12=0,p9=189 HTTP/1.1

Que es poco más que una copia de lo que escribimos arriba, sin más que sustituir los espacios por %20 (Una manía que viene de largo).

Podemos ya usar esto con un parser para activar comandos en nuestro Arduino.

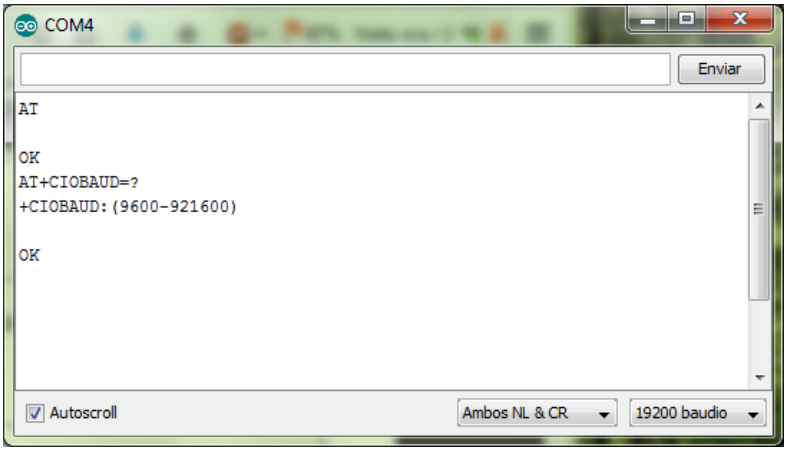
- *Un parser es un programa que en la jerga informática, analiza textos buscando instrucciones a realizar*.

Basta con escribir un programa que revisa las entradas de texto en la consola y gobierne los pines de Arduino, en función de lo que se encuentre. Ya hicimos cosas así en las sesiones previas relativas al shield Ethernet y siguen siendo plenamente aplicables aquí.

Cambiando la velocidad de comunicación

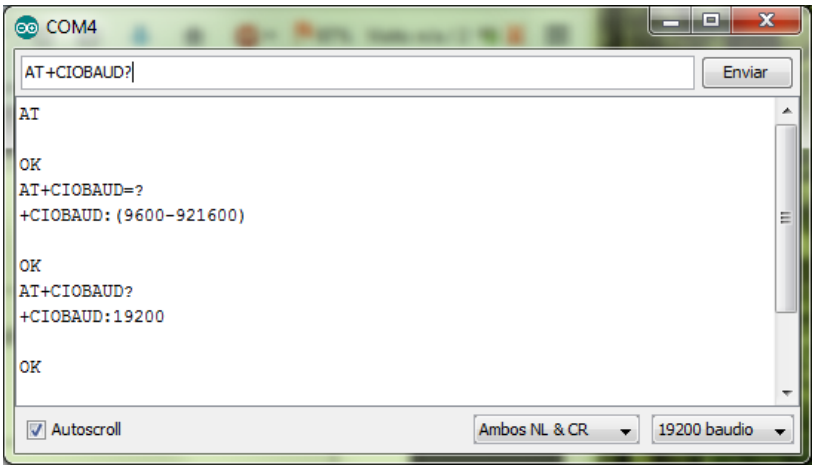
Para saber la velocidad de comunicación a la que tu modulo puede funcionar, tenemos el comando:

AT+CIOBAUD=?



Que nos informa, de que acepta velocidades de comunicación entre 9600 y 921600, casi nada.

Para conocer la velocidad actual, tenemos otro comando:



Y para modificar la velocidad de comunicación por la puerta serie:

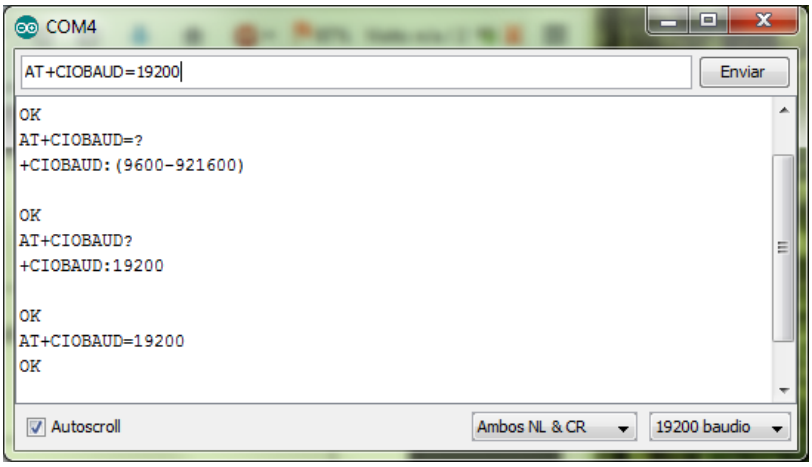
AT+CIOBAUD=xxxx

Pero según el manual del ESP8266, solo podemos elegir entre, 9600, 19200, 38400, 74880, 115200, 230400, 460800 y 921600 . El problema es que solo coincidimos con la consola Arduino en las velocidades de 9600 (El valor por defecto), 19200 y 115200.

Ya hemos dicho que la opinión en Internet es que a 115200, la comunicación no es fiable y por tanto elegiremos 19200 como velocidad estándar, lo que no es mucho si vamos a usar la WIFI para algo más que mandar mensajes de texto.

Asi que por ahora elegiremos 19200 como velocidad estándar, con el comando

AT+CIOBAUD=19200



Recordad ahora reprogramar vuestro Arduino con la nueva velocidad modificando el primer programa que usamos en esta sesión.

En las próxima sesión veremos cómo presentar valores de lecturas por ejemplo, en el servidor Web que acabamos de montar.

RESUMEN DE LA SESIÓN

- Hemos visto un primer contacto con el módulo WIFI ESP8266.
- Presentamos un pequeño circuito de prueba para conectarlo a nuestros Arduinos, sin necesidad de programas externos de comunicación.
- Vimos las primeras instrucciones AT que acepta este módulo.
- Montamos un mínimo servidor Web con nuestro modula, apto ya para recibir instrucciones a través de un navegador Web, incluyendo el de teléfonos y tabletas.