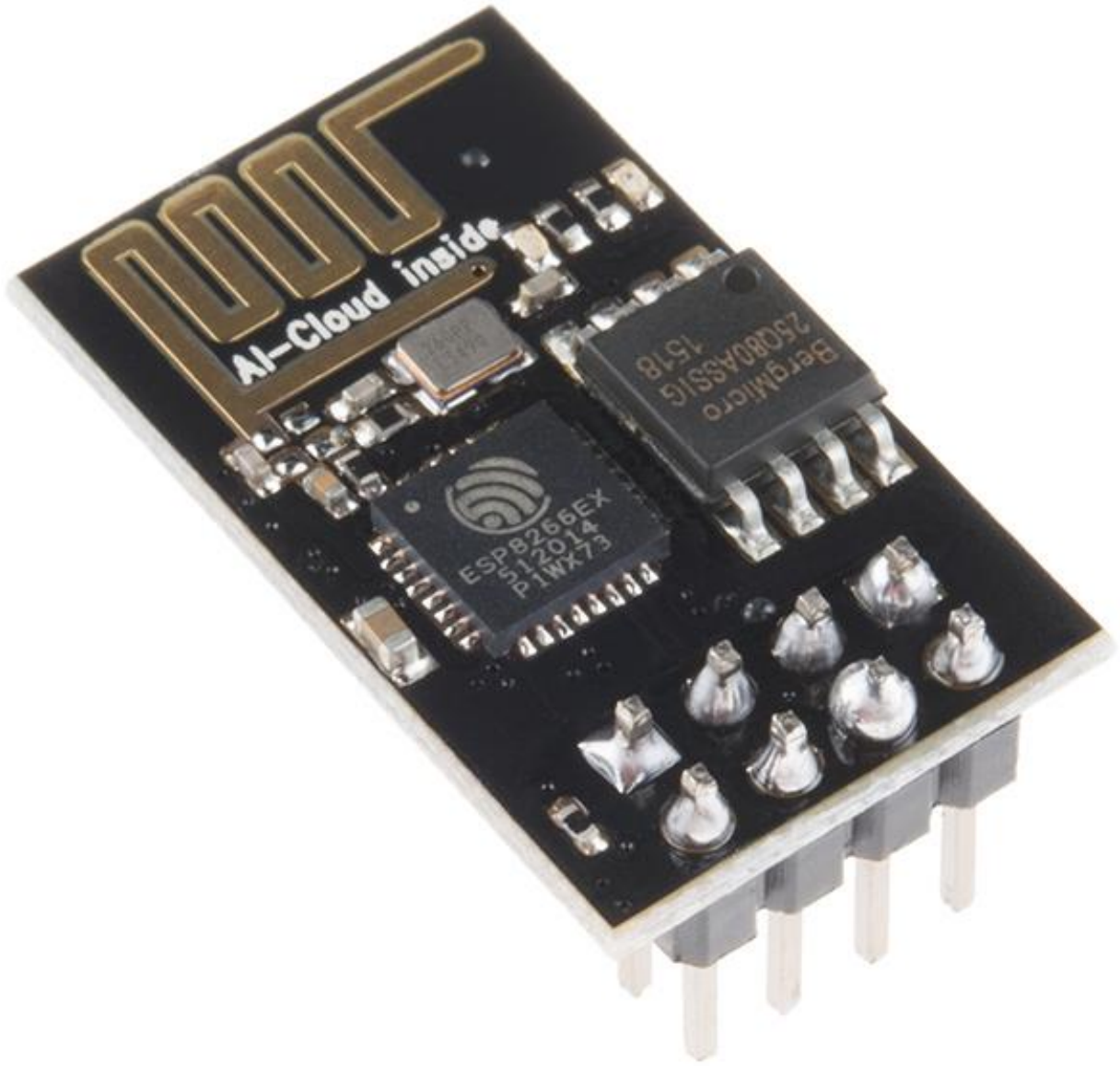


26/06/2017

Chip ESP8266

Microcontrolador i chip WiFi

Miquel Àngel Román Colom

Taula de continguts

Esp8266 – Que és?

L'ESP8266 és una família de chips Wi-Fi de baix cost que inclouen tota la pila de protocols TCP/IP i que, a més, posseeix una MCU (*Micro Controller Unit)*.

La família de chips ESP8266 ha agafat moltíssima força dins al mercat degut a la seva potència, preu i mida dins els "mons" de *Internet of Things*, *makers* i *Do It Yourself* entre d'altres.

Així mateix, la família ESP8266 en aquest moment consta d'uns 14 membres, sense incloure *subversions*. Aquests s'identifiquen per el seu nom, ESP8266, i la seva variant del mòdul, -01 a -14.

***Figura1**: En aquesta figura es veuen, en aquest ordre, els ESP8266: -02,-12 i -01*

Com que en aquest document es treballa amb l'ESP8266-01, només entrarem en detall en aquesta versió encara que les altres versions funcionaran de manera molt similar.

Característiques del ESP8266-01	
CPU	80MHz CPU. 32-bit RISC*.
RAM	64KB per codi. 96KB per dades (variables).
Pins	8 pins. 2 GPIO*. RX i TX.
WiFi	IEEE 802.11 Wi-Fi
Voltatge emprat	3.3V

***Taula 1**: Característiques del ESP8266-01.*

**RISC: Reduced Instruction Set Computer.*

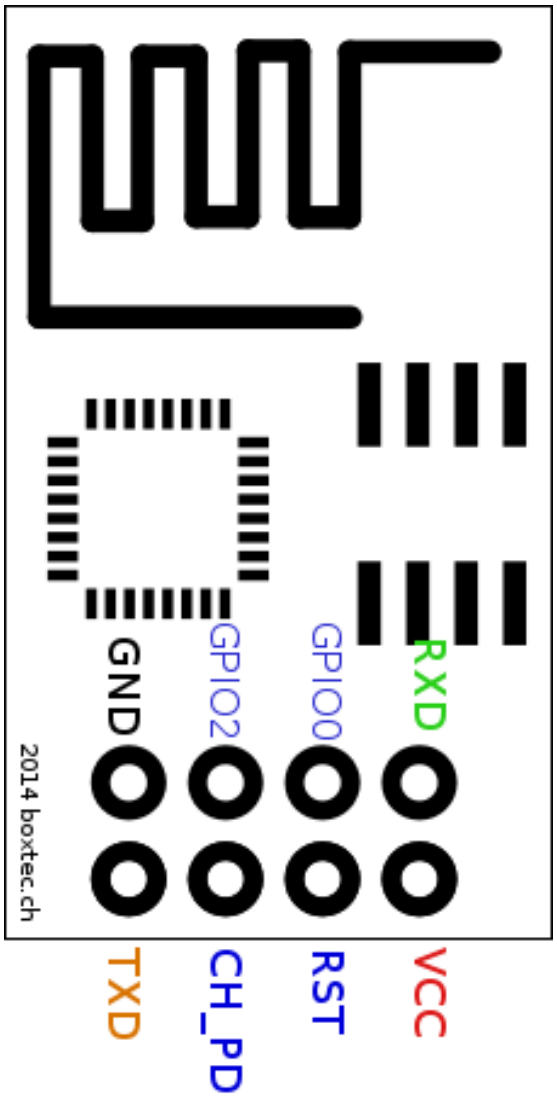
**GPIO: General Purpose Input Output (pins).*

ESP8266-01 - Pinout

En aquesta secció es descriurà el *pinout* del chip i de quina manera es pot dur a terme una connexió bàsica i més que suficient per poder començar a treballar amb el chip juntament amb altres controladors com Arduino.

Cal tenir en compte que la manera amb que els fabricants presenten el pinout del chip es mirant-lo des de la part de dalt, és a dir, mirant des de la cara on es pot apreciar l'antena PCB encara que els pins surtin per la cara oposada.

Com ja s'ha especificat anteriorment el chip és realment senzill i només disposa de 8 pins.



PIN	DESCRIPCIO
VCC	Voltatge d'entrada. 3.3V
GND	*Ground*, terra o massa.
GPIO0, GPIO2	*General Purpose Input Output Pins.* Funcionen de la mateixa manera que els pins de l'Arduino.
RXD	Pin de recepció de dades tipus Serial.
TXD	Pin d'enviament de dades tipus Serial.
CH_PD	Pin de "*Chip select*". Es conecta juntament amb VCC per tal d'encendre el dispositiu.
RST	Pin que quan hi passa corrent, reinicia el dispositiu.

***Taula 2**: Descripció dels pins corresponents a la figura 2.*

***Figura 2**: Pinout del ESP8266-01 vist desde la part superior del chip, on les línies son l'antena PCB.*

Tot i que el chip es totalment programable, només serà mostrat com a mòdul complementari a Arduino ja que és realment l'objecte de treball.

Premisses del circuit

En aquest cas; com que treballarem amb una *Arduino Mega*. No per res en especial, però aquesta placa disposa de tres ports Serial ja integrats que funcionen molt millor que els que es puguin crear amb la llibreria *SoftwareSerial*.

El gran problema de la llibreria mencionada més a dalt és que a *baud ratios* molt alts, perd precisió en alguns dels bits i acaba donant informació corrupta.

En qualsevol dels casos els sketch emprats funcionen en totes les plaques de manera anàloga sempre que es canviïn els pins de manera adequada.

Les especificacions són les següents:

OVERVIEW | TECH SPECS | DOCUMENTATION

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13

***Figura 3**: Especificacions tècniques de la Arduino Mega.*

De manera similar que en el cas del mòdul HC-06 podem tenir les mateixes consideracions:

Pel que podem observar:

Operating Voltage	5 V

***Figura 4**: La placa opera amb voltatges de 5V, i els pins RX i TX del mòdul només treballen amb 3,3V.*

Anem a esbrinar quina seria la resistència necessària a afegir per no fer mal bé el mòdul.

Per la llei d'Ohm:

$$I = \frac{V}{R}$$

La intensitat que tenen els I/O pins de la placa és la següent:

DC Current per I/O Pin	20 mA

***Figura 5**: Amperatge de la placa en els pins I/O*

Per tant ens queda la següent equació:

$$5V - X = 3.3V$$

$$X = 1.7V$$

S'ha de provocar una caiguda de tensió de 1,7V. Anem a calcular la resistència necessària.

I = 20mA = 20 · 10^{-3} A; V = 1.7V i R = R. (Valor desconegut)

$$20 \cdot 10^{-3} \times R = 1.7$$

$$\boldsymbol{R = 85\Omega}$$

Com que aquest valor no és més gran que de 100 Ω, el podem menysprear i directament no afegir cap tipus de resistència.

Per tant no fa falta afegir divisors de voltatge per els exemples següents.

Justificació de la elecció de la placa

Es coneix que es creen varis errors si emprem la llibreria *SoftwareSerial* i treballem amb *baud ratios* bastant elevats. (En aquest cas 115200).

```
#define BAUDS 115200

int second = 1000;

void setup() {
  Serial.begin(BAUDS);
  Serial1.begin(BAUDS);
  delay(second);
  Serial.println("SETUP READY");
}

void loop() {
  byte aux;

  while(Serial1.available()>0){
    aux = Serial1.read();
    Serial.write(aux);
  }

  while(Serial.available()>0){
    aux = Serial.read();
    Serial1.write(aux);
  }

}
```

```
#include <SoftwareSerial.h>

#define bauds 115200
SoftwareSerial esp(10, 11); //RX, TX.

int second = 1000;
void setup() {
  Serial.begin(bauds);
  esp.begin(bauds);
  delay(second);
  Serial.println("SETUP READY");
}

void loop() {
   byte aux;

  while(esp.available()>0){
    aux = esp.read();
    Serial.write(aux);
  }

  while(Serial.available()>0){
    aux = Serial.read();
    esp.write(aux);
  }
}
```

***Figura 7**: A la dreta podem veure l'sketch que fa us del HardwareSerial mentre que el de la dreta empra la llibreria SoftwareSerial. Els sketchs són exactament el mateix, exceptuant el fet de l'ús de la llibreria mencionada.*

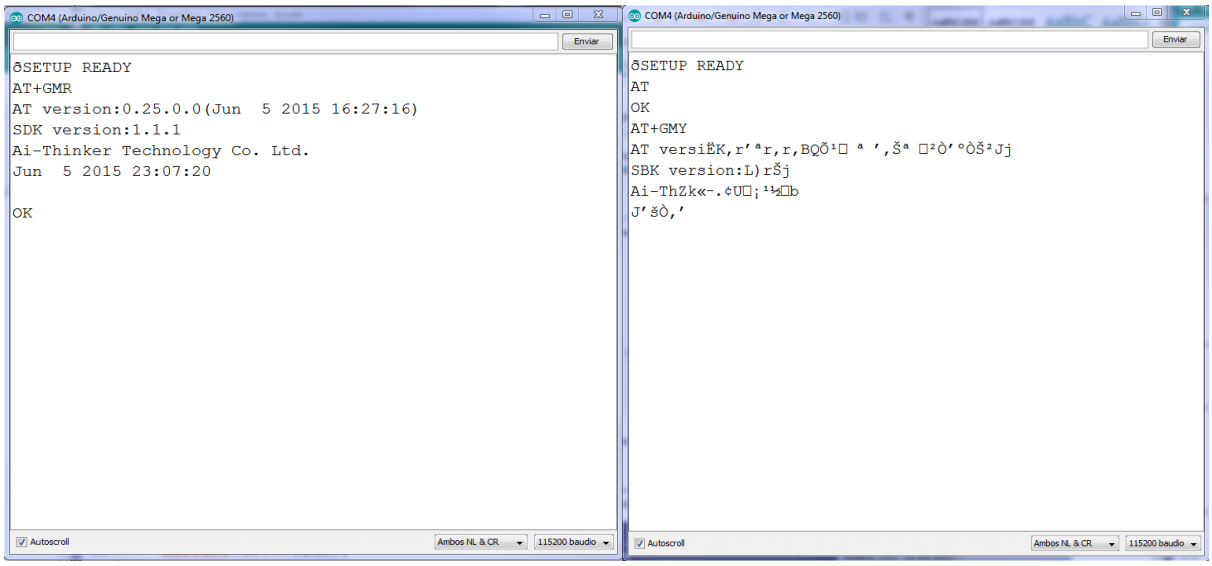


***Figura 8**: A la dreta podem veure el terminal que executa l'sketch que fa us del HardwareSerial mentre que el de la dreta empra la llibreria SoftwareSerial. És veu una clara corrupció del stream de dades.*

Existeix la manera de configurar el ESP8266-01 de tal manera que treballi amb *baud ratios* més petits, la qual s'explicarà al final d'aquest document ja que està relacionada amb les comanes AT del chip, però és més fàcil emprar un Hardware Serial directament.

S'ha triat la placa Mega ja que és la única placa amb més d'un Hardware Serial, que jo tingui a la meva disposició, per tal de poder aïllar millor els problemes i resoldre'ls de manera més clara i estructurada.

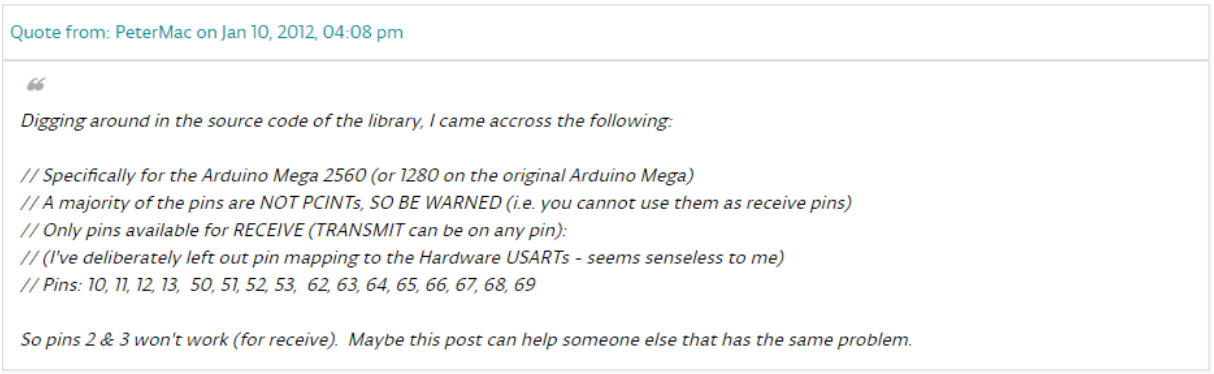
Cal afegir, que qualsevol dels Serial que duen ja inclosos la gran majoria de les Arduino, pot esser emprat com a HardwareSerial per funcionar als 115200 baudis, però d'aquesta manera separem un poquet més les coses i no embullem res del built-in serial que alhora està connectat al USB i és el que podrem visualitzar per el *Serial Terminal* del IDE.

Els 4 HardwareSerial de tal placa són els següents:

Identificador a l'IDE	PIN RX	PIN TX
Serial	0	1
Serial1	19	18
Serial2	17	16
Serial3	15	14

Tot i haver explicat tot el que s'ha exposat fins ara, cal afegir que no tots els pins de les plaques són adecuats i/o suporten l'ús de la llibreria *SoftwareSerial*. Per a més informació s'hauria de fer un poc més de recerca a la pàgina web oficial sobre la placa en la qual s'està treballant.

En aquest cas:


Quote from: PeterMac on Jan 10, 2012, 04:08 pm

Digging around in the source code of the library, I came accross the following:

```
// Specifically for the Arduino Mega 2560 (or 1280 on the original Arduino Mega)
// A majority of the pins are NOT PCINTs, SO BE WARNED (i.e. you cannot use them as receive pins)
// Only pins available for RECEIVE (TRANSMIT can be on any pin):
// (I've deliberately left out pin mapping to the Hardware USARTs - seems senseless to me)
// Pins: 10, 11, 12, 13,  50, 51, 52, 53,  62, 63, 64, 65, 66, 67, 68, 69
```

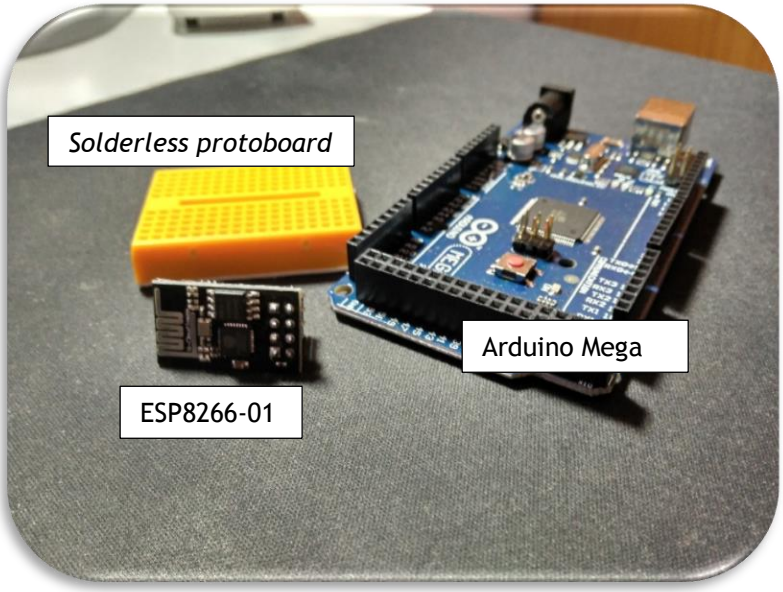
So pins 2 & 3 won't work (for receive). Maybe this post can help someone else that has the same problem.

***Figura 9**: Extret del fòrum de arduino.cc. No existeixen problemes per als pins de transmissió, però sí en els pins de recepció de dades. En el cas de la placa Mega, només es poden emprar els pins 10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69 com a pins RX.*

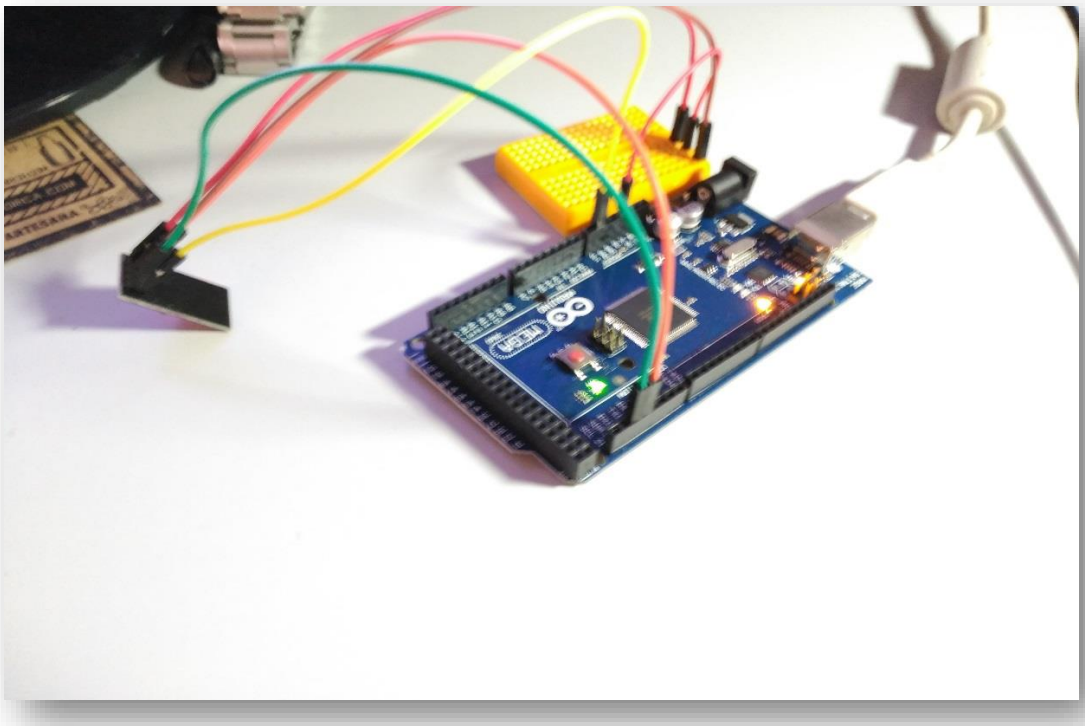
Muntatge

La següent imatge resumeix el material necessari per tal de poder començar a treballar.

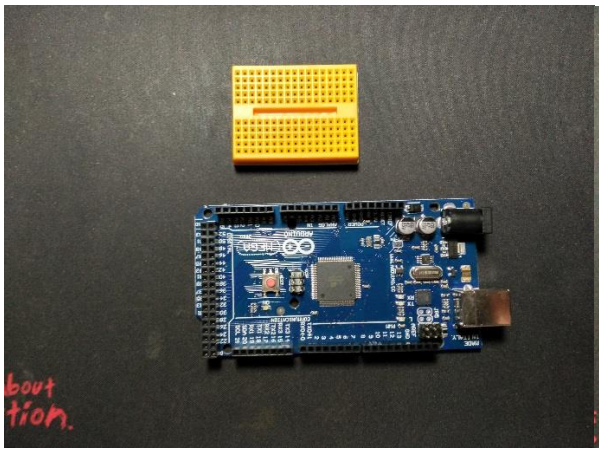
- Un ESP8266-01
- Una *protoboard*
- Cablejat suficient
- Una Arduino. Qualsevol.

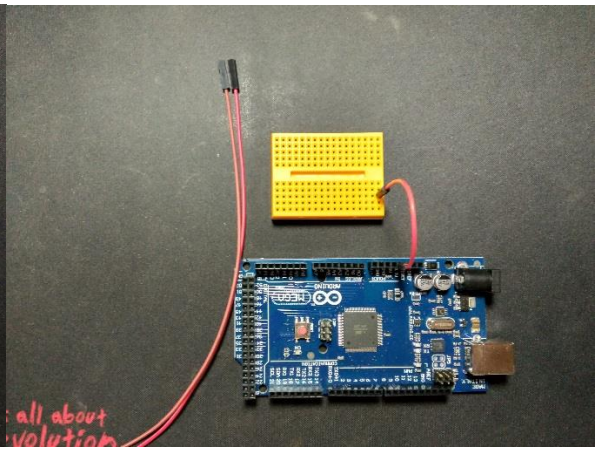


En aquest apartat s'explicarà de manera detallada i passa per passa, la manera amb la qual es construirà el següent circuit:

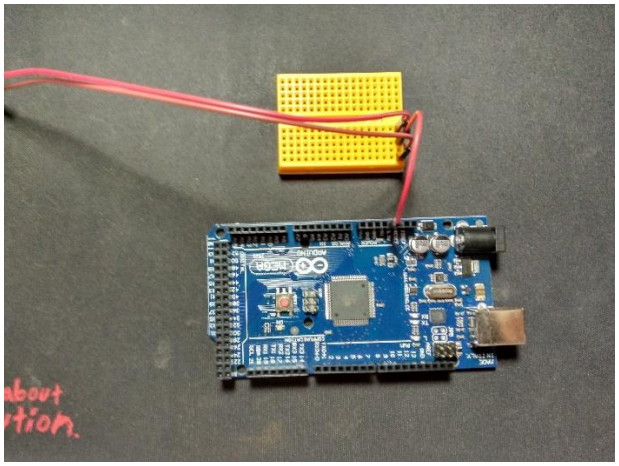


Construcció del circuit

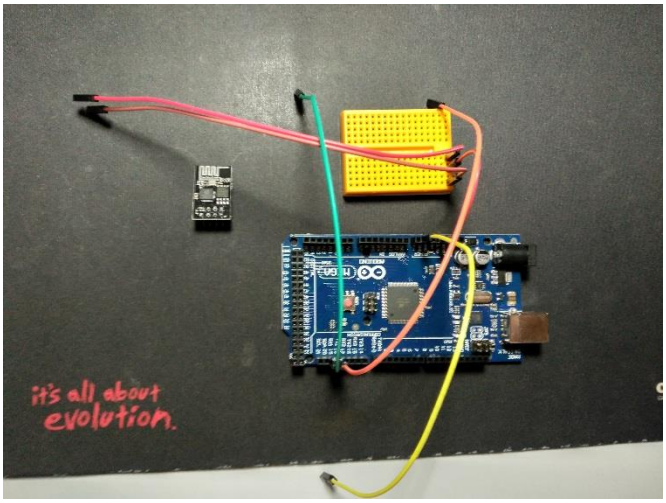




Comencem per treure la *solderless protoboard* i la placa microcontroladora amb 3 cables. En connectem un al pin de 3.3V i aquest el connectam a la protoboard.



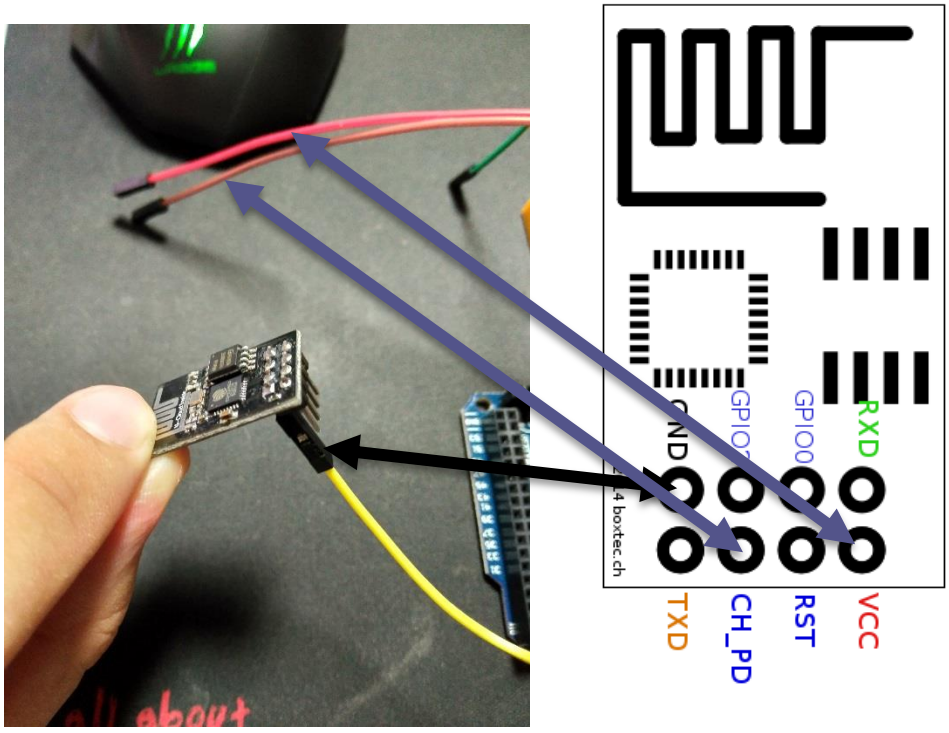
La idea és emprar la protoboard per poder aconseguir dos cables amb una tensió de 3.3V per tal de poder alimentar els pins VCC i CH_PD del ESP8266-01.

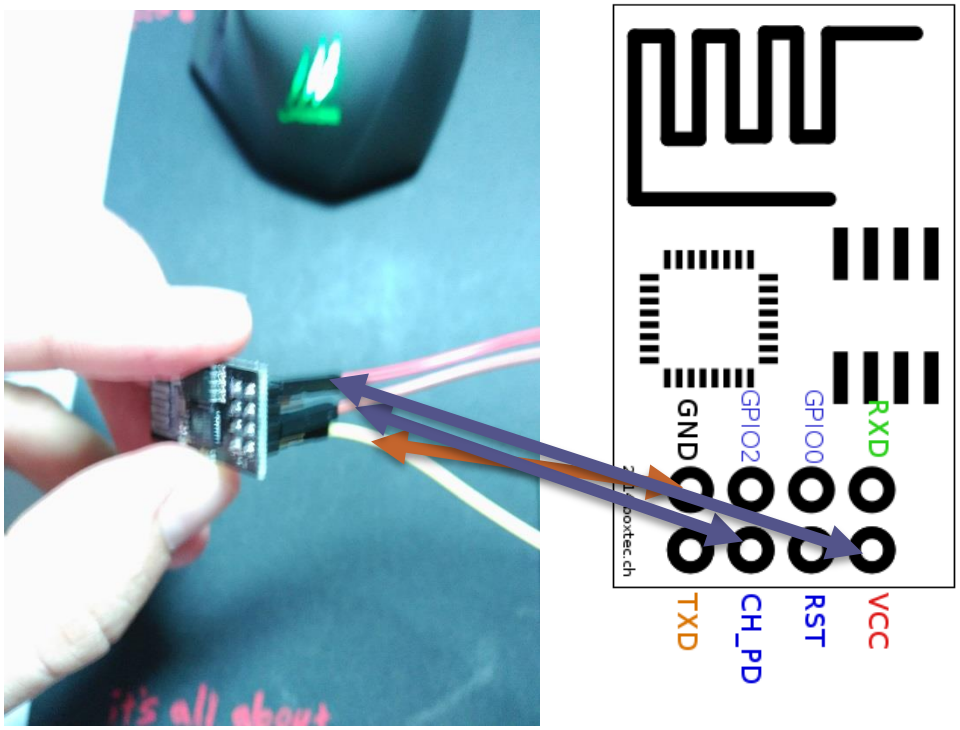


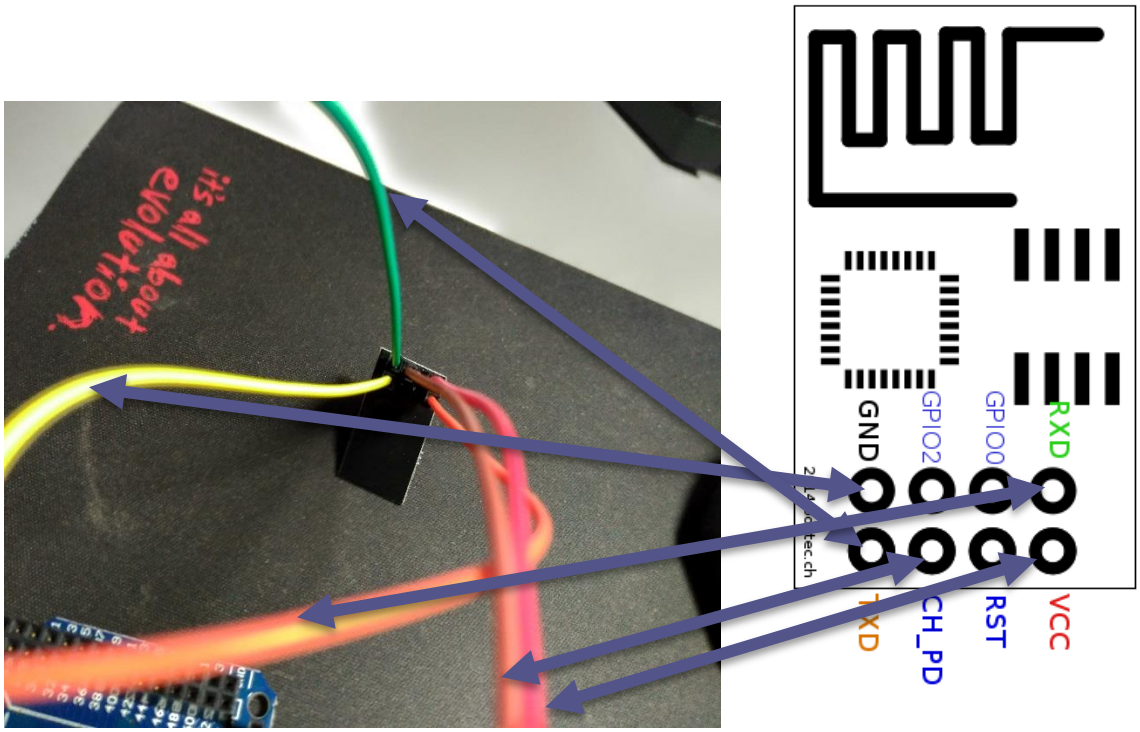
Posteriorment, connectem el cable groc al pin GND de l'Arduino, que posteriorment serà connectat el GND del chip.

Llavors, connectem el cable verd al pin RX del Arduino que serà connectat al TX del chip i el pin TX del Arduino al RX del chip.

Una bona metàfora per entendre-ho seria veure-ho de la següent manera: S'ha de connectar la "*boca*" del Arduino amb "*l'orella*" del chip ja que sinó no s'entenen.

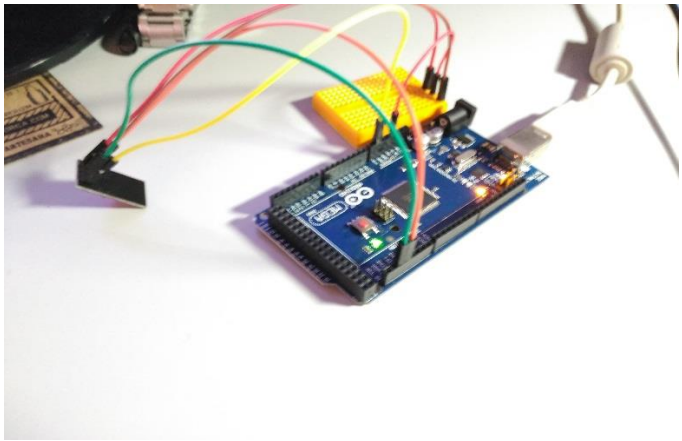

RXD
GPIO0
GPIO2
GND
VCC
RST
CH_PD
TXD


RXD
GPIO0
GPIO2
GND
VCC
RST
CH_PD
TXD



RESUM DE LES CONNEXIONS			
CABLE	PIN ARDUINO	PIN ESP8266-01	PROTOBOARD
GROC	GND	GND	NO
VERD	RX (19 MEGA)	TXD	NO
TARONJA	TX (18 MEGA)	RXD	NO
VERMELL	CONECTAT A PBOARD	VCC	SÍ
MARRÓ	CONECTAT A PBOARD	CH_PD	SÍ

*** Els cables que estan connectats a la protoboard estan directament connectats al cable vermell que uneix la protoboard amb el pin 3.3V de la Arduino.*



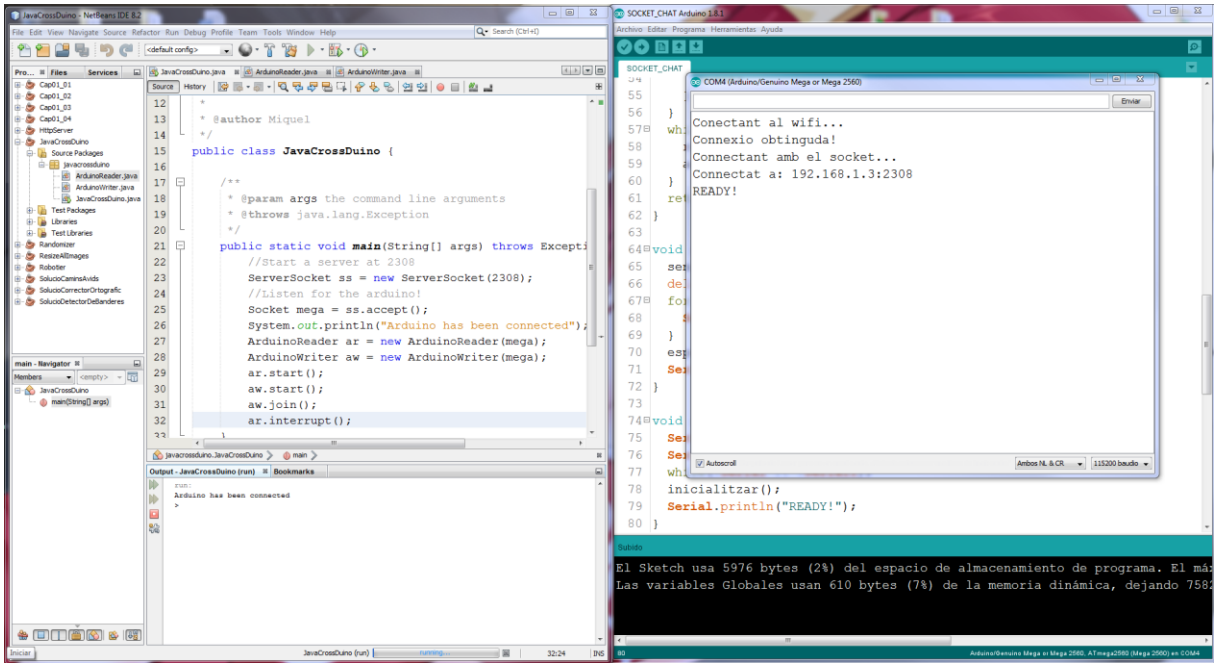
Part lògica. Software

Els sketch.

SOCKET_CHAT

Un simple sketch que ens permet xatejar des de l'Arduino amb un altre dispositiu.

Aquest sketch funciona independentment de la màquina, el llenguatge i el sistema. S'ha provat amb els llenguatges Java i Go en màquines tant Windows 7 com Ubuntu 16.04LTS i ARMBIAN Xenial (Orange Pi).



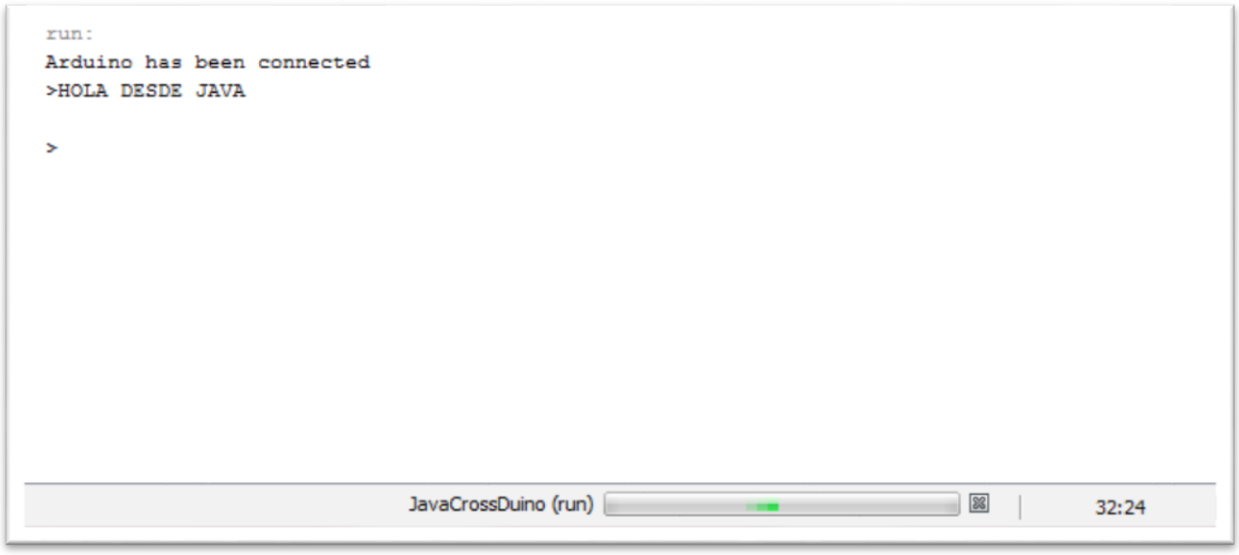


***Figura 10**: Output i command prompt del programa escrit en Java.*

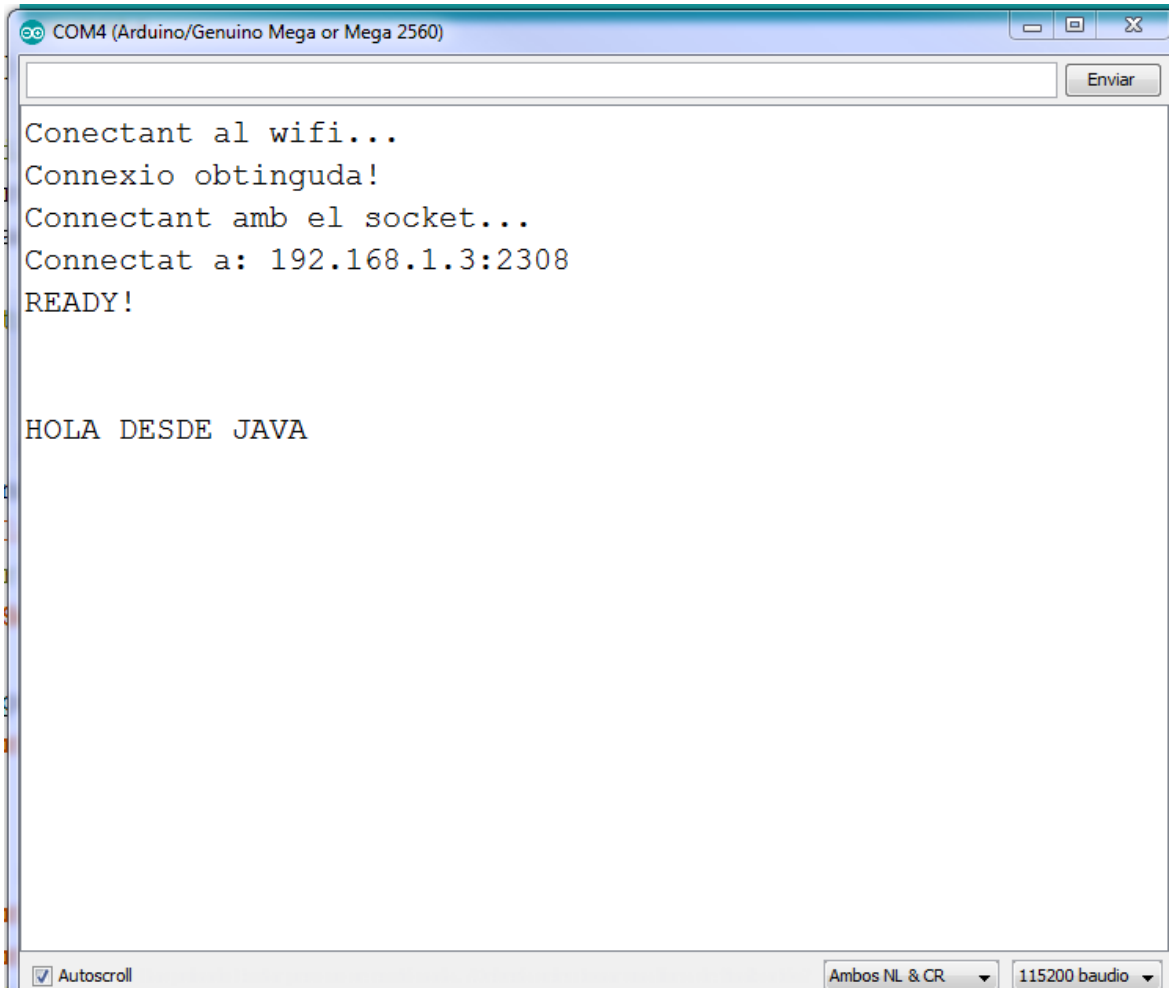


***Figura 11**: Output rebut a l'Arduino*

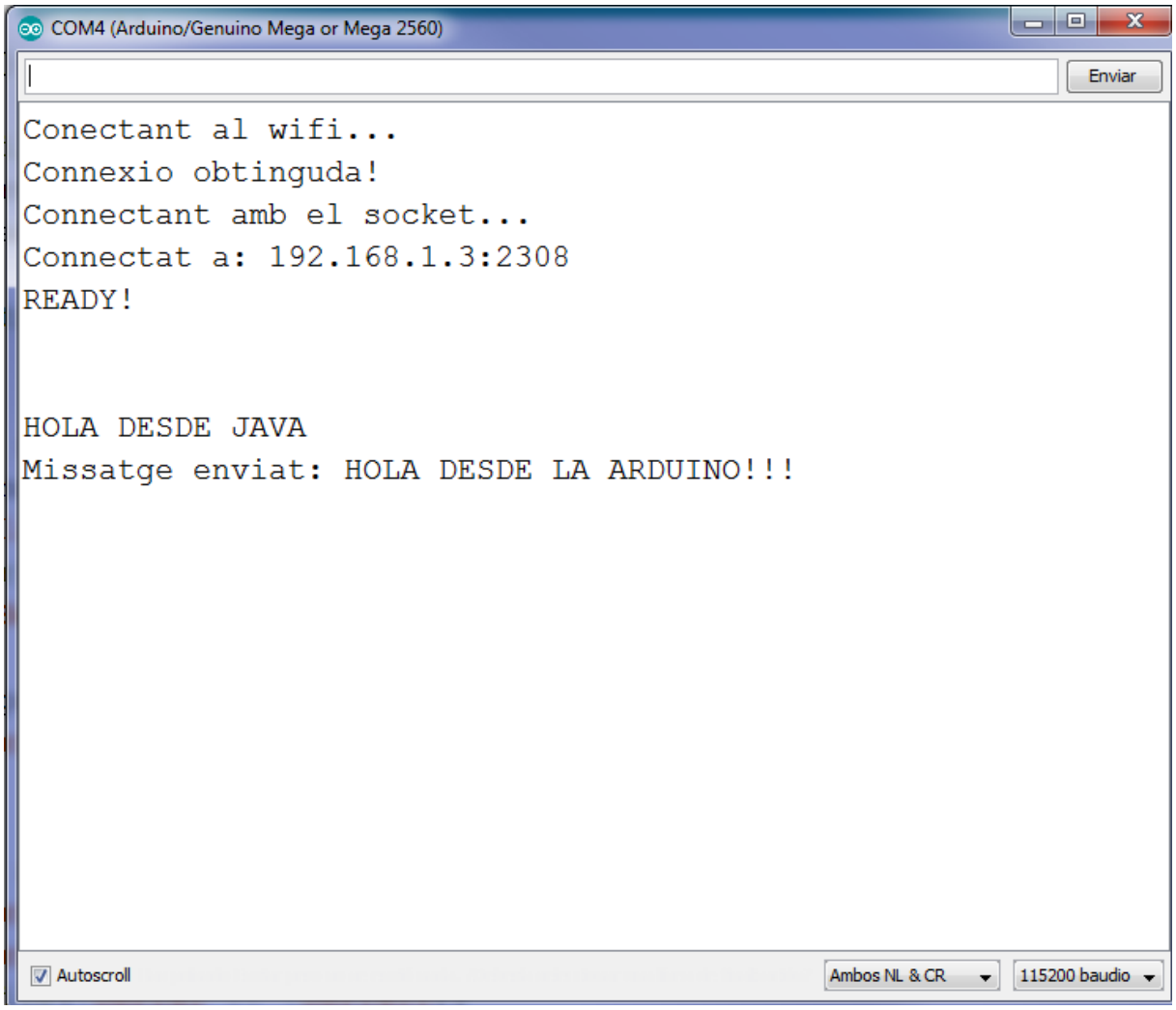


***Figura 12**: Escrivint un missatge des de el serial monitor cap a Java.*

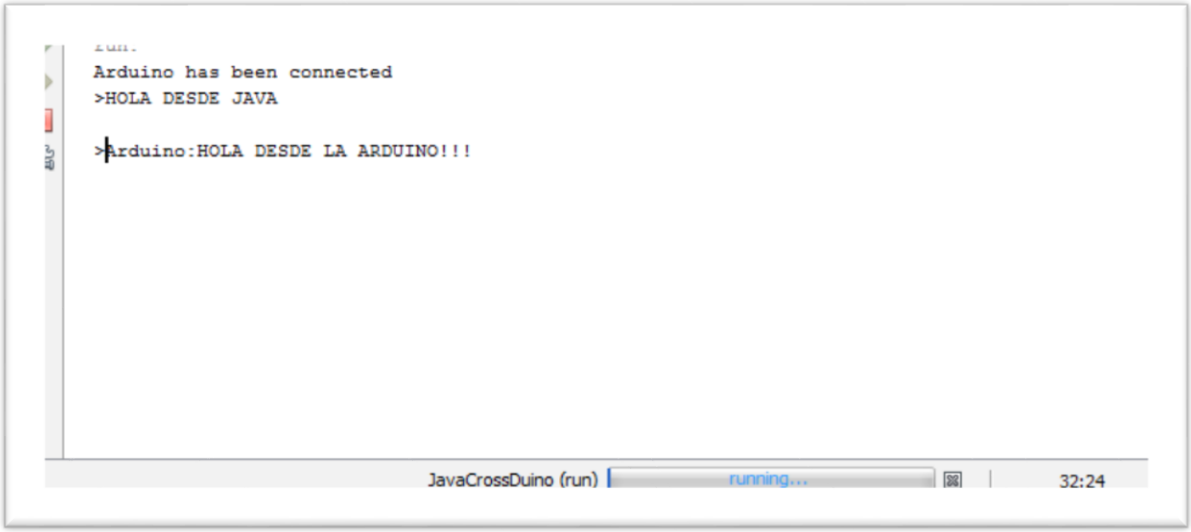


***Figura 13**: Rebent el missatge del Arduino al programa escrit en Java.*

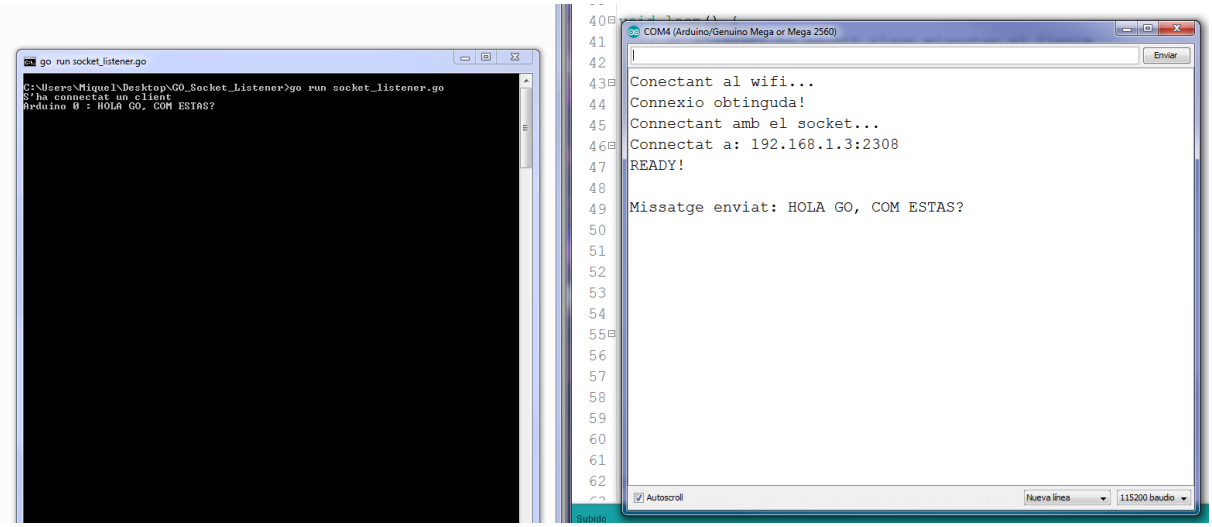


***Figura 14**: Socket server escrit amb Go amb el mateix sketch del Arduino.*

Cal dir que sense modificar massa més el codi és pot automatitzar l'enviament de dades de tal manera que l'Arduino enviï dades sense fer ús del monitor sèrie del IDE. Amb un programa d'aquest tipus el podríem lligar directament al un websocketd i treballar amb això per tal de poder crear una aplicació de navegador web.

El codi del sketch no figura aquí ja que es trobarà adjunt a la documentació i està més que suficientment documentat.

HTTP_REQUEST

El següent sketch demostra de manera simple com podem realitzar una petició a un *webserver*. En aquest exemple s'il·lustra la manera en la qual podem realitzar una petició de tipus GET. Les peticions POST es fan de manera anàloga.

De primera mà comptem amb un *webserver* ja creat, tot i que podríem realitzar la petició a qualsevol altre servidor web com per exemple google.es.

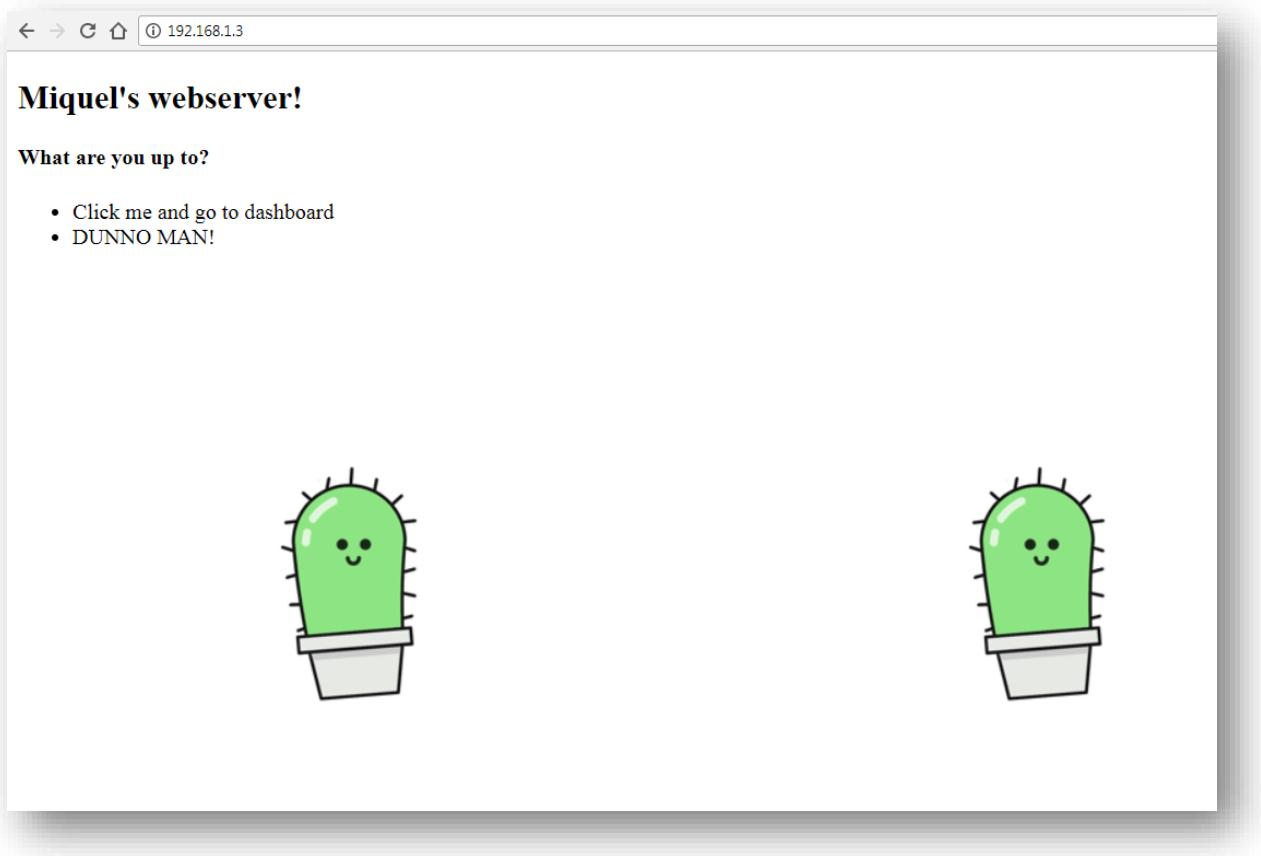


***Figura 15**: Simple webserver creat amb XAMPP.*

Llavors, el que ens resta és mirar de realitzar una petició al webserver perquè aquest ens entregui el codi HTML de la pàgina web en qüestió.

Això ho podem fer de manera realment senzilla amb el següent sketch:

```
#define bauds 115200

void setup() {
  // Inicialitzam les connexions serial.
  Serial.begin(bauds);
  Serial1.begin(bauds);
  // Esperam a que aquestes estiguin llestes.
  while(!Serial || !Serial1);
  Serial.println("READY");
}

void loop() {

  Serial1.write("AT+CIPSTART=\"TCP\",\"192.168.1.3\",80\r\n");
  delay(1000);
  byte b;
  // Mostram la resposta del modul quan s'ha obert la conexio
  // al port i adreça especificats.
  while (Serial1.available() > 0) {
    b = Serial1.read();
    Serial.write(b);
  }

  // Peticio GET de la arrel del webserver.
  String request = "GET / HTTP/1.0\r\n\r\n";
  // Comanda a executar.
  String cipsend = "AT+CIPSEND=";

  // Completam la comanda afegint-hi
  //    1. Els bytes a enviar
  //    2. La cua de les comanes AT
  cipsend += request.length();
  cipsend += "\r\n";
  // Executam la comanda
  Serial1.print(cipsend);
  // Esperam a que es processi.
  delay(1000);
```

```
// Mostram la resposta del mòdul
while (Serial1.available() > 0) {
  b = Serial1.read();
  Serial.write(b);
}

// Realitzam la peticio HTTP
Serial1.print(request);
// Bloquejam la placa mostrant tota la INFO
// que ens envii el webserver.
while (true) {
  if (Serial1.available() > 0) {
    b = Serial1.read();
    Serial.write(b);
  }
}
```

Aquest sketch assumeix que el mòdul ESP ja té disponible una connexió a una red Wi-Fi i directament intenta obrir una connexió a l'adreça especificada i al port 80 (port web).

El que es mostra per terminal és el següent:

```
READY
AT+CIPSTART="TCP","192.168.1.3",80
CONNECT

OK
AT+CIPSEND=18

OK
>
Recv 18 bytes

SEND OK

+IPD,1146:HTTP/1.1 200 OK
Date: Thu, 29 Jun 2017 21:41:49 GMT
Server: Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.21
Last-Modified: Thu, 25 May 2017 23:14:36 GMT
ETag: "369-55061626bfeaf"
Accept-Ranges: bytes
Content-Length: 873
Connection: close
Content-Type: text/html


<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title id="title">MIQUEL'S WEBSERVER</title>
        <link rel="icon" href="media/images/silvestre.jpg">
        <script src="js/basic_operations.js"></script>
    </head>
    <body>
        <script type="text/javascript">

        </script>
        <h2>Miquel's webserver!</h2>

        <h4>What are you up to?</h4>
        <ul>
            <li id="dash">Click me and go to dashboard</li>
            <li> DUNNO MAN!</li>
        </ul>

        <img src="https://media.giphy.com/media/l0HlHgiDgsW2c32p2/giphy.gif"
        <img src="https://media.giphy.com/media/l0HlHgiDgsW2c32p2/giphy.gif"
    </body>
    <script type="text/javascript">
        //document.getElementById('dash').addEventListener("click", redirectD
    </script>
</html>
CLOSED
```

Que efectivament és el codi HTML de la pagina web. D'aquesta mateixa manera es poden realitzar peticions (inserts i querys) a la base de dades del webserver, només que hauríem de fer les peticions amb els "formularis emplenats" i fer la petició del .php, .jsp, .asp... corresponent.

Altres dades d'interès

LES COMANDES AT DUEN SEQÜENCIA D'ACABAMENT (\r\n)

Llistat de comanes AT per al mòdul ESP8266-01

Abans de dir res més, cal dir que aquest llistat de comanes AT és directament depenent de la versió del *firmware* de la placa i del SDK que dugui carregat. També, aquí s'inclouran les comanes AT més interessants i útils ja que n'hi ha infinitat.

Adjunt a aquest document es troba el llistat complet de comanes AT, en format pdf, descarregat de la empresa fabricant del chip; per tal de que si aquí hi manqués alguna cosa sempre seria bona idea pegar-hi una ullada.

Comandes AT del dispositiu bàsiques.

- AT
  Comprova que es pot enviar i rebre dades de manera satisfactòria.

- AT+UART_DEF
  Guarda la configuració aplicada a la memòria flash del dispositiu.
  AT+UART_DEF="baud","data_bits","stopbits","parity","flowcontrol"
  AT+UART_DEF=9600,8,1,0,0

- AT+RST
  Reinicia el mòdul.

- AT+GMR
  Comprova la versió del firmware, SDK i hora de compilació d'aquest.

- AT+RESTORE
  Recupera la configuració de fàbrica del dispositiu.

Comandes AT relacionades amb el mòdul Wi-Fi del dispositiu.

- AT+CWMODE
  Configura el mode del dispositiu:
  1. AT+CWMODE=1 //Configura el mòdul com a estació.
  2. AT+CWMODE=2 //Configura el mòdul com a punt d'accés.
  3. AT+CWMODE=3 //Configura el mòdul com a estació i punt d'accés.

- AT+CWLAP
  Llista tots els AP que estan disponibles a l'abast del chip.

- AT+CWJAP
  Es connecta a un *access point* disponible.
  AT+CWJAP="SSID","CONTRASEÑA"

- AT+CWQAP
  Es desconnecta del *access point* al qual estava connectat en un principi.

- AT+CWSAP
  Configura l'access point que crea el dispositiu en cas d'estar en mode AP.
  AT+CWSAP="ESP8266-01","contrassenya123",1,3 // per exemple.

- AT+CWLIF
  Llista totes les direccions IP i MAC que estan connectades actualment al seu AP.

- AT+CWAUTOCONN
  Indica si el dispositiu s'ha de connectar a un AP conforme s'encén.
  1. AT+CWAUTOCONN=0 // No es connecta automàticament
  2. AT+CWAUTOCONN=1 // Sí que es connecta automàticament

- AT+CWHOSTNAME
  Configura el nom de estació que té el mòdul.

Comandes AT relacionades amb el els protocols TCP/IP.

- AT+CIPSTART
  Crea una connexió del tipus indicat, amb la direcció indicada al port indicat.
  AT+CIPSTART=”TCP”,”192.168.1.3”,8909

- AT+CIFSR
  Informa sobre les direccions IP (local) i MAC del dispositiu.

- AT+CIPSEND
  Envia un *stream* de N bytes indicat.
  AT+CIPSEND=4
  >HOLA

- AT+CIPSENDEX
  Similar al anterior, tot i que si es vol enviar menys dades de les indicades podem indicar l’acabament del buffer amb el caràcter \0

- AT+CIPCLOSE
  Tanca la connexió oberta o la indicada en cas de que el mòdul estigui en mode MUX=1.
  AT+CIPCLOSE || AT+CIPCLOSE=0

- AT+CIPMUX
  Permet habilitar o des-habilitar el fet de tenir vàries connexions obertes al mateix temps.
  AT+CIPMUX=0

- AT+CIPSERVER
  Crea o destrueix un servidor de tipus TCP.
  AT+CIPSERVER=0,80 //Destrueix el servidor TCP creat al port 80
  AT+CIPSERVER=1,2309 // Crea un servidor al port 2309

- AT+CIPSTO
  Defineix la quantitat de segons que han de passar perquè es tanqui una connexió inactiva amb el servidor.
  AT+CIPSTO=80 //s.

Emprant la placa com a interfície per configurar el mòdul ESP8266-01

Per poder configurar el mòdul ho podem fer de dues maneres:

Carregant el següent sketch

```
#define BAUDS 115200

int second = 1000;

void setup() {
  Serial.begin(BAUDS);
  Serial1.begin(BAUDS);
  delay(second);
  Serial.println("SETUP READY");
}

void loop() {
  byte aux;

  while(Serial1.available()>0){
    aux = Serial1.read();
    Serial.write(aux);
  }

  while(Serial.available()>0){
    aux = Serial.read();
    Serial1.write(aux);
  }

}
```

Essencialment, aquest sketch s'encarrega de llegir tot el que entra per el monitor sèrie del IDE i passar-ho directament al hardware serial al qual està connectat el mòdul.

D'aquesta manera només resta escriure les comanes AT de configuració necessàries, deixar la placa llesta i muntar els següents circuits amb els seus sketch adients!

Carregant un sketch buit a la placa i connectant RX amb RX i TX amb TX

D'aquesta manera, tot el que s'enviï des de el serial monitor del IDE es transmetrà directament al mòdul tal i com si l'USB estigués directament connectat al ESP8266-01.

Llavors executem totes les comanes AT que trobem necessàries i llestos, ja el tenim configurat.

Una solució al problema del SoftwareSerial

Una de les maneres més senzilles de solucionar aquest problema plantejat és, que abans de emprar la placa, tenir-la ja configurada que vagi al *baud ratio* desitjat i que sigui *plug'n'play*.

També una altra opció podria ser executar el següent codi:

```
#include <SoftwareSerial.h>

#define RX 10
#define TX 11

SoftwareSerial esp(RX,TX);

void setup() {
  esp.begin(115200);

  //Esperam a que s'inicii el serial.
  while(!esp);

  esp.write("AT+CIOBAUD=9600\r\n");
  delay(1000);

  esp.begin(9600);
  Serial.println("READY!");
}
```

Però seria jugar un poc a la loteria ja que no tenim cap tipus de garantia de que la comanda enviada no es corrompria.