by Luis del Valle
Revisión by Miguel

Desde que irrumpió el ESP8266 en el mercado, han surgido muchas variantes de este microcontrolador. En el podcast puedes escuchar el capítulo que dedicamos al ESP8266 donde hablamos de las diferentes alternativas. Quizás el más famoso entre los Makers sea el **ESP-01 debido a que fue el primero y a su bajo coste**. De éste precisamente voy a hablarte hoy en este artículo.

Si lo que estás buscando es información sobre NodeMCU puedes seguir el tutorial paso a paso o la guía para poder programar este kit de desarrollo desde el IDE de Arduino.

La primera vez que compré un Arduino UNO, quedé fascinado al encender un simple LED. Tras varios años haciendo proyectos, me entró la curiosidad y empecé a investigar las posibilidades que nos ofrece Arduino conectado a una red o Internet.

Rápidamente me compré un **Shield Ethernet** y la verdad, funciona muy bien. La gran pega es que tiene que tener una toma de red cerca para que se conecte. Una alternativa para evitar esto es comprar un **Shield WiFi**, pero el precio tira para atrás. Cuesta 4 veces más que la propia placa de Arduino, no tiene sentido.

De repente, descubrí que una empresa de China había sacado un módulo WiFi de muy bajo coste y que permitía conectarse a un Arduino UNO a través de él. Este módulo era el **ESP-01** y la empresa **Espressif**. Cuando lo compré estaba en una etapa muy temprana sin apenas documentación y muy complicado de programar.

Tras unas semanas probando y configurando lo metí en un cajón donde ha permanecido mucho tiempo. Su dificultad a la hora de utilizarlo con Arduino era máxima y no merecía la pena.

Con el tiempo ha ido evolucionando y en la actualidad se puede utilizar como un módulo autónomo sin necesidad de utilizarlo con Arduino es decir, como si fuera un Arduino. Hay que recordar que el ESP8266 es un microcontrolador y como tal, puede hacer las mismas funciones que un Arduino UNO. Eso sí, tiene sus limitaciones.

Firmware para el ESP-01

Empiezo explicando un tema que debemos tener muy claro ya que hay mucha confusión al respecto en Internet. El **firmware es el software de bajo nivel que nos va a permitir controlar los circuitos** eléctricos.

Por defecto, en el ESP-01 viene **instalado la versión con la que podemos comunicar con el ESP8266 mediante comandos AT** a través del puerto serie. No voy a entrar muy en detalle qué son estos comandos, lo único es decir que fue creado por la compañía Hayes y se convirtió en un estándar para la configuración y parametrización de modems.

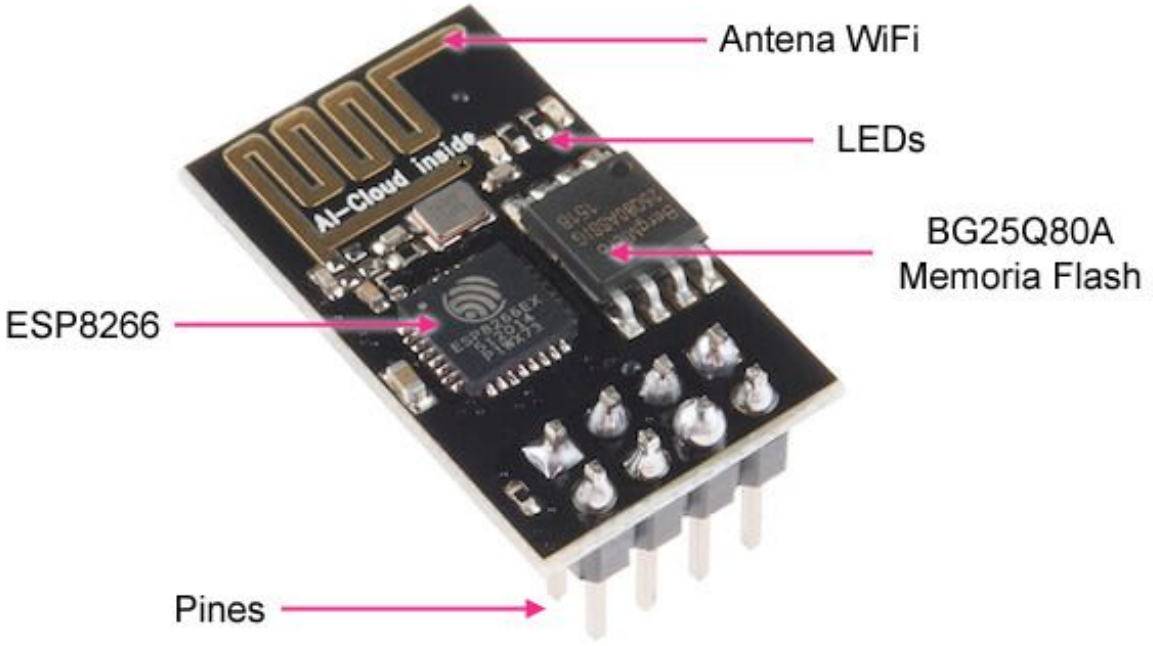
Este tipo de comunicación nos permitirá hacer un puente entre Arduino y el ESP8266. De esta manera conseguimos que un Arduino UNO se conecte a una red WiFi. Puedes obtenerlo en la web oficial de Espressif.

Pero si lo que queremos es cargar nuestro propio programa como si de un Arduino se tratara, solo tenemos que crear el sketch y cargarlo. Esto sobrescribirá el firmware de comandos AT que viene por defecto en el ESP-01. Todo esto lo veremos más adelante.

Existen también **otros firmwares** que funcionan como intérpretes. Hacen precisamente esa función, interpretar un lenguaje de alto nivel y lo traducen para poder controlar el microcontrolador ESP8266. Hay intérpretes para Python (microPython), Javascript (Espruino) o Basic (ESPBasic).
Una vez que tenemos claro **qué es un firmware**, ya podemos empezar a programar nuestro ESP-01. Primero debemos alimentarlo y saber cómo podemos programar nuestro ESP-01 basado en el ESP8266.

ESP-01 la placa microcontroladora

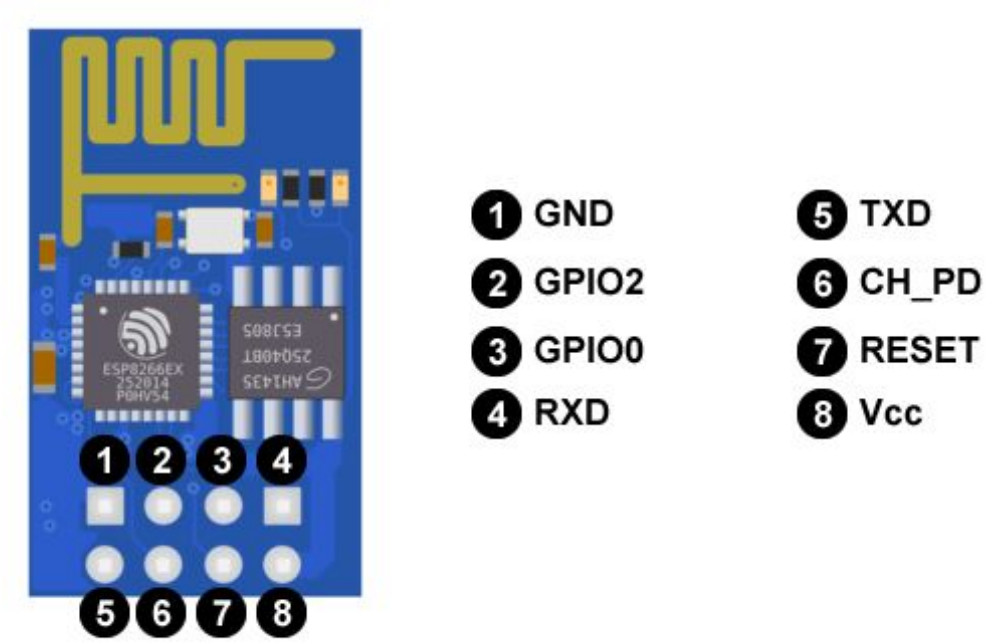
Antes de nada, vamos a conocer el ESP-01. En la siguiente imagen puedes ver las **partes más importantes**.



- ESP8266 es el microcontrolador del módulo ESP-01.
- Pines donde conectaremos la alimentación, sensores y transmisión de programa.
- BG25Q80A es la memoria *flash* donde residen los programas o *sketchs*. El ESP8266 no dispone de este tipo de memoria y por eso es un chip a parte.
- LEDs que nos informan de si está encendido o no y de la transmisión de datos (Tx y Rx).
- La antena WiFi para conectarse a una red/Internet.

Pines del ESP-01, una de sus limitaciones

El ESP-01 tiene 8 pines, cada uno de ellos está pensado para una tarea concreta.



1. GND es la toma de tierra.
2. GPIO2 es una entrada salida de propósito general. Es el pin digital número 2.
3. GPIO0 es una entrada salida de propósito general. Es el pin digital número 0.
4. RXD es el pin por donde se van a recibir los datos del puerto serie. Trabaja a 3,3 V. También se puede utilizar como pin digital GPIO: sería el número 3.
5. TXD es el pin por donde se van a transmitir los datos del puerto serie. Trabaja a 3,3 V. También se puede utilizar como pin digital GPIO: sería el número 1.
6. CH_PD pin para apagar y encender el ESP-01: si lo ponemos a 0 V (LOW) se apaga, y a 3,3 V (HIGH) se enciende.
7. RESET pin para resetear el ESP-01: si lo ponemos a 0 V (LOW) se resetea.
8. Vcc es por donde alimentamos el ESP-01. Funciona a 3,3 V y admite un máximo de 3,6 V. La corriente suministrada debe ser mayor que 200 mA.

Como puedes comprobar, no tenemos **ningún pin analógico y solo 4 pines digitales GPIO0, GPIO2, RXD (GPIO3) y TXD (GPIO1)**. Estos dos últimos funcionan como pines I/O siempre y cuando el programa que carguemos a la placa no los utilice, por ejemplo, para mostrar información a través del monitor serie.

El ESP-01 soporta la comunicación I2C: si nuestros sensores se comunican mediante este protocolo no tendremos problema y, **con tan solo dos pines de nuestro ESP-01**, podremos comunicarnos con **decenas de sensores**.

Por último, con respecto a los **pines digitales o GPIO**, es importante recalcar que **funcionan a 3,3 V,** es decir, el estado alto (HIGH) serán 3,3 V. Los pines RXD y TXD cuando funcionan para comunicar mediante puerto serie también utilizan este voltaje. Luego veremos cómo sortear esto si lo conectamos a un Arduino UNO para programarlo.

Opciones para alimentar el ESP-01

Antes de hacer nada, tenemos que alimentar el ESP-01: funciona a una **tensión de 3,3 V y admite un máximo de 3,6 V**. Se recomienda una **intensidad mayor de 200 mA** debido a que cuando está transmitiendo a través de WiFi puede alcanzar picos de consumo de más de 200 mA.

Existen diferentes posibilidades para alimentar el ESP-01. Vamos a ver las ventajas e inconvenientes que presentan cada una de estas posibilidades.

Opción 1: a través de un Arduino UNO

Arduino UNO tiene dos pines de alimentación dentro del bloque destinado a alimentar sensores y actuadores: puede suministrar 5 V y 3,3 V. Seguramente estés pensando en alimentar el ESP-01 con el **pin de 3,3 V (en el rango de operación del ESP8266)** pero aunque efectivamente estamos dentro del rango de voltaje, el inconveniente es que **este pin puede suministrar un máximo de 50 mA**. Esto supone que en ocasiones puede quedarse corto, sobre todo cuando estamos haciendo un uso intensivo de la parte WiFi.

Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Fuente Arduino.cc

Por lo tanto, **alimentar un ESP-01 basado en el ESP8266 con un Arduino UNO no es una buena opción**. Puedes hacerlo y probablemente te funcionará en muchos casos, pero llegará un punto en que deje de hacerlo.

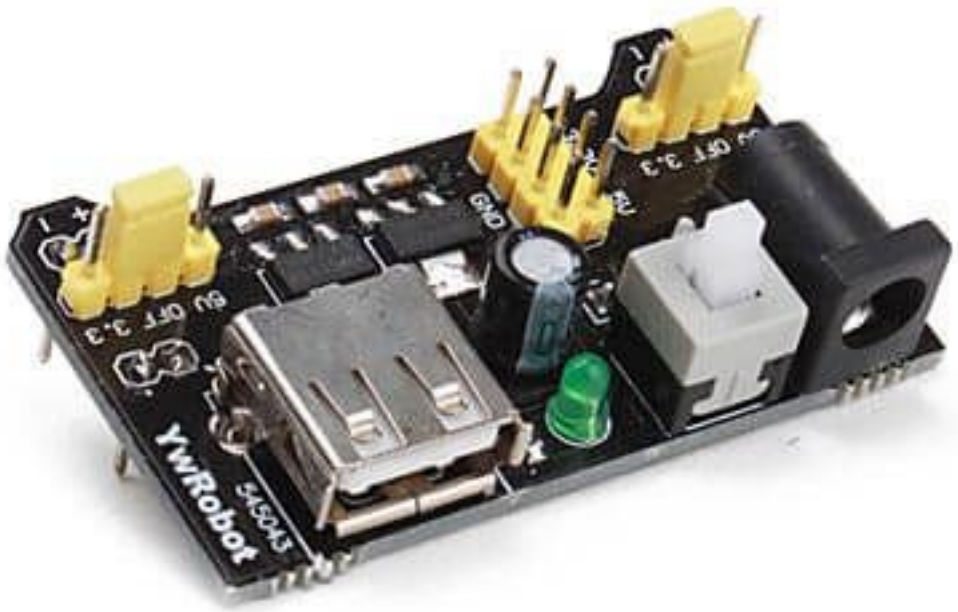
Opción 2: a través de un adaptador USB-Serie

Con un adaptador USB-Serie (también llamados FTDI o TTL), podemos programar un ESP8266 y también alimentarlo.

En el mercado existen multitud de adaptadores de este tipo. Lo importante es conocer la corriente que puede suministrar cada uno de ellos. Lo típico es que no lleguen al mínimo requerido por el ESP-01 así que, generalmente, tampoco es una buena opción: la intensidad que suelen suministrar estos componentes varía entre los 50 y los 120 mA.

Opción 3: fuente de alimentación de 3,3 V y 5 V

Quizás **la mejor opción que tengamos para alimentar un ESP8266** sea una fuente de alimentación de este estilo: suministra 3,3 V y/o 5 V lo que supone una ventaja ya que muchos de los sensores y componentes que vamos a utilizar necesitan 5 V, como por ejemplo el sensor de temperatura LM35 o el sensor de ultrasonidos.



Existen muchas marcas y modelos en el mercado. La que yo voy a utilizar es la **MB102** **de YwRobot**. Casi todas suministran entre 700 y 800 mA, más que suficiente para alimentar el ESP-01 y los sensores.

Conectando el ESP-01 a la fuente de alimentación

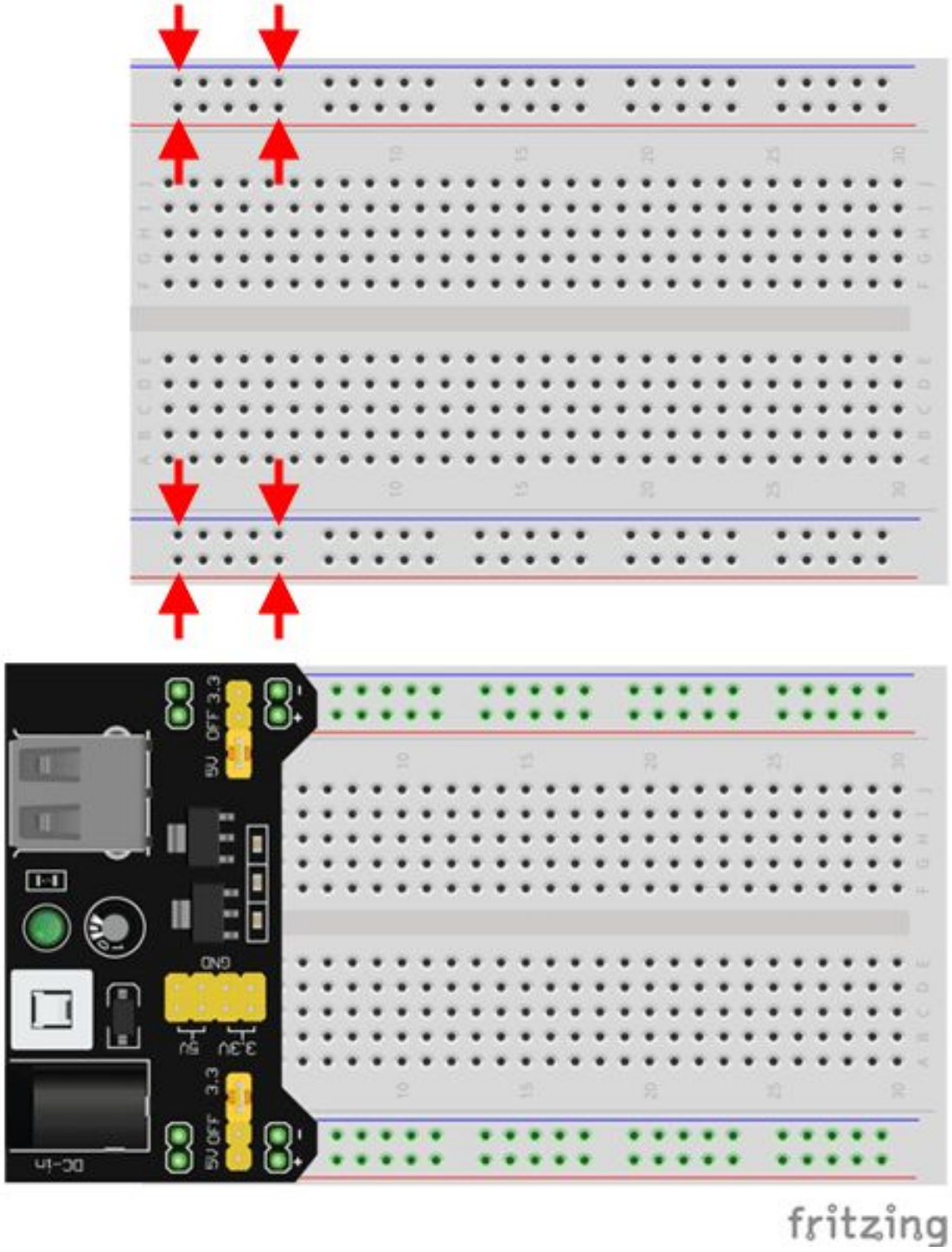
Lo primero es ver cómo configurar la fuente para que nos de los 3,3 V necesarios para alimentar el ESP-01.

Fuente de alimentación MB102

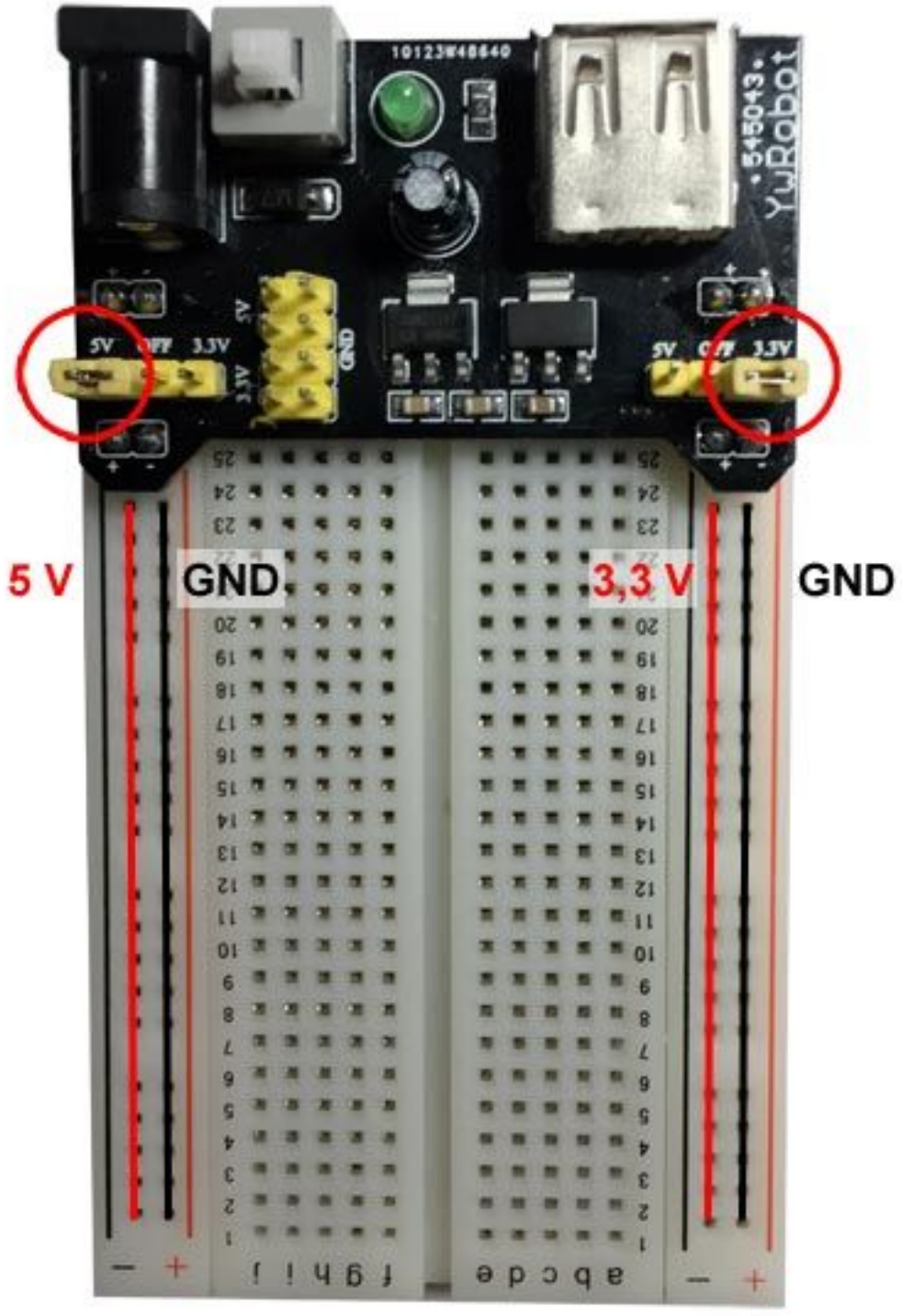
Este tipo de fuentes de alimentación para electrónica tienen la particularidad de que **suministran 3,3 V y/o 5 V**. Tienen dos salidas de potencia colocadas en los extremos y dependiendo de cómo coloquemos el *jumper* suministrará un voltaje u otro.

Lo primero es colocar la placa en la protoboard: debemos encajar 8 pines, 4 por cada línea de potencia.

Lo colocamos en la protoboard en las filas donde pone + y -.



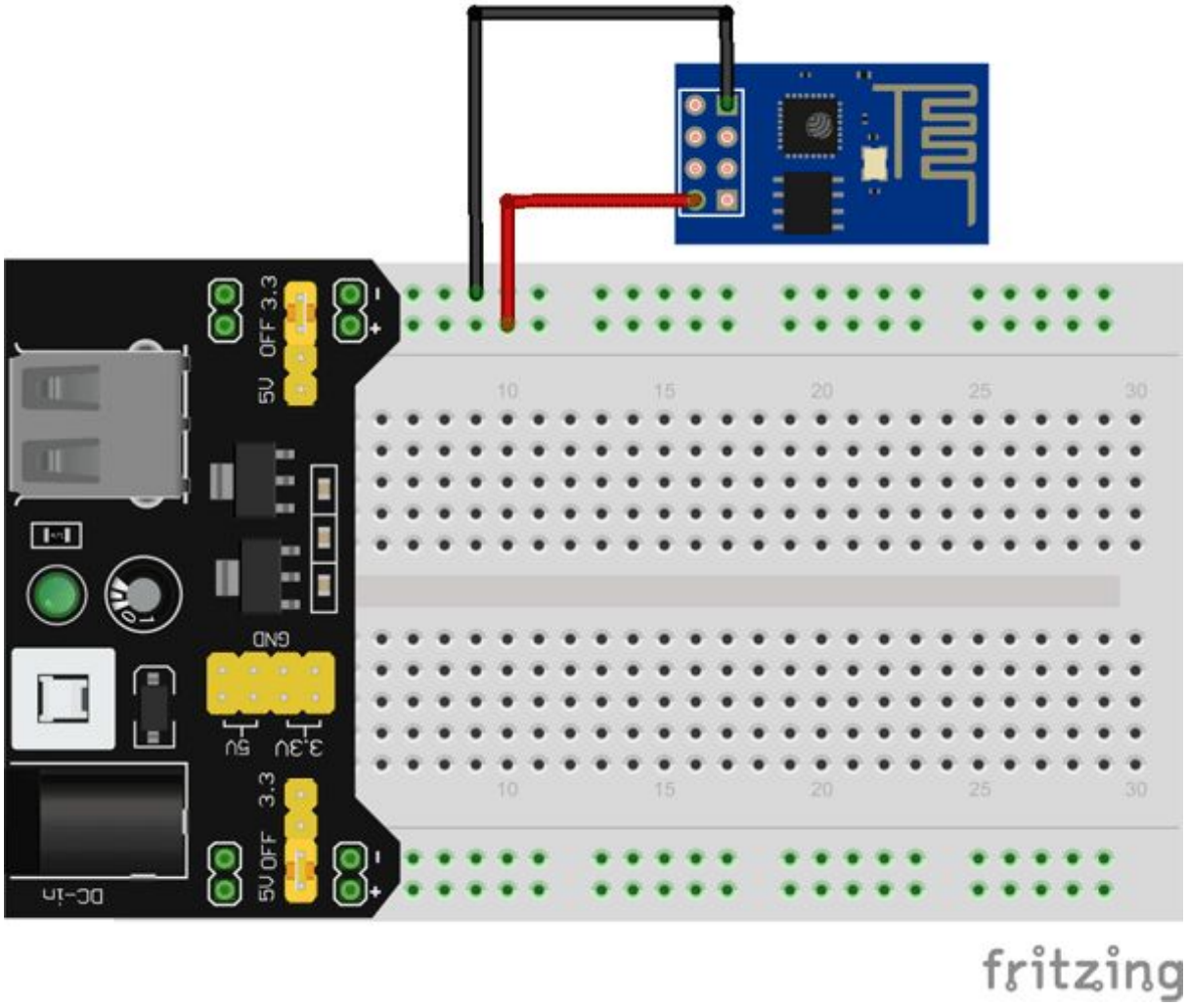
Para elegir el voltaje tenemos unos *jumpers* que según los coloquemos, la fuente suministrará 3,3 V y/o 5 V. Lo más práctico es tener una línea con 3,3 V y otra con 5 V.



Conectando el ESP-01 a la protoboard

Ya tenemos todo listo y podemos alimentar el ESP-01 con la fuente de alimentación. Utilizaremos la línea que está suministrando 3,3 V. Debes tener un cable macho-hembra o un cable macho-macho y otro hembra-hembra.

Los pines son el 1 para GND y el 8 para Vcc. El esquema sería el siguiente.



Cómo programar el ESP-01

Una vez que tenemos alimentado el ESP-01 vamos a empezar con la **programación**. En este apartado también hay diferentes opciones: las veremos y así podrás elegir la que más te guste :)

Para **cargar el programa** lo haremos a través del **puerto serie**, lo que supone que **utilizaremos los pines RX y TX** para transmitir los datos a la memoria Flash, donde se almacenará el *sketch*.

Normalmente, cuando trabajamos con **Arduino, no debemos decir si vamos a cargar un programa o si debe ejecutarlo: todo esto lo hace de forma interna y automática.**

Sin embargo, con el **ESP-01** esto no sucede así y **debemos ser nosotros, de forma manual, quienes activemos los diferentes modos**. Existen dos modos de operación y se configuran a través de los puertos GPIO0 y GPIO2: el **modo Flash** y el **modo UART**.

Modo UART: carga de programa en la memoria del ESP-01

Cuando queremos cargar un programa en el ESP-01 debemos encenderlo o resetearlo teniendo el pin GPIO0 a nivel bajo (LOW = 0 V = GND) y el GPIO2 a nivel alto (HIGH = 3,3 V = Vcc). Te recuerdo que el ESP8266 trabaja con niveles lógicos de 3,3 V. El pin GPIO2 está por defecto a HIGH, ya que tiene un *pull-up* interno, por lo que podemos dejarlo simplemente desconectado.

Modo Flash: ejecución de programa en el ESP-01

Una vez cargado el programa hay que ejecutarlo: esto lo conseguimos teniendo el GPIO0 y el GPIO2 a nivel alto (3,3 V). Tanto el GPIO0 como el GPIO2 están por defecto a HIGH, ya que ambos tienen un *pull-up* interno, por lo que podemos dejarlos simplemente desconectados.

En la siguiente tabla puedes ver un resumen de los dos modos y los valores de los pines.

	GPIO0	**GPIO2**
Modo UART (carga programa)	LOW	HIGH (desconectado)
Modo Flash (ejecuta programa)	HIGH (desconectado)	HIGH (desconectado)

En este punto te estarás preguntando, si utilizamos RX y TX para cargar el programa en la placa y GPIO0 y GPIO2 para indicar el modo de trabajo **¿cómo conectamos los sensores y actuadores al ESP-01?**
Hay que tener en cuenta **dos cosas**:

- RX y TX los utilizamos para cargar el programa. Una vez finalizada la carga los podemos utilizar como pines de entrada y salida digitales.
- Los modos de trabajo se indican a través de los pines 0 y 2 **cuando se resetea o reinicia la placa**. Una vez que tengamos el modo de ejecución funcionando (funcionamiento normal), ya podemos conectar cualquier componente a estos pines.

Preparando el entorno de desarrollo oficial de Arduino

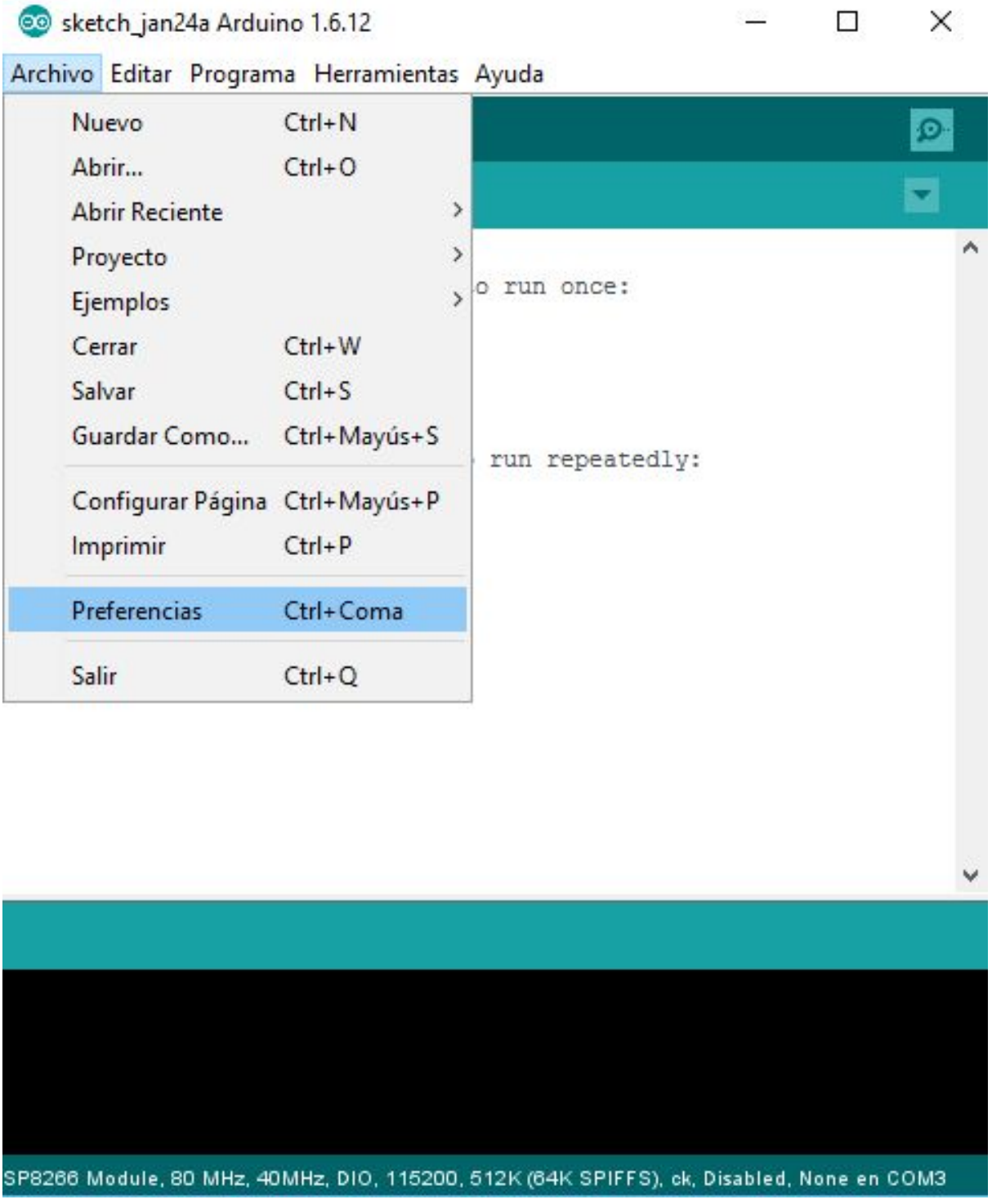
Puedes ver el siguiente vídeo donde se explica los pasos que vamos a ver en este artículo.
https://youtu.be/0g7sazWXfEI

Antes de meternos de lleno en preparar el circuito para programar el ESP-01, vamos a preparar el entorno de desarrollo: para ello utilizaremos el entorno de desarrollo de Arduino que lo puedes descargar desde aquí.

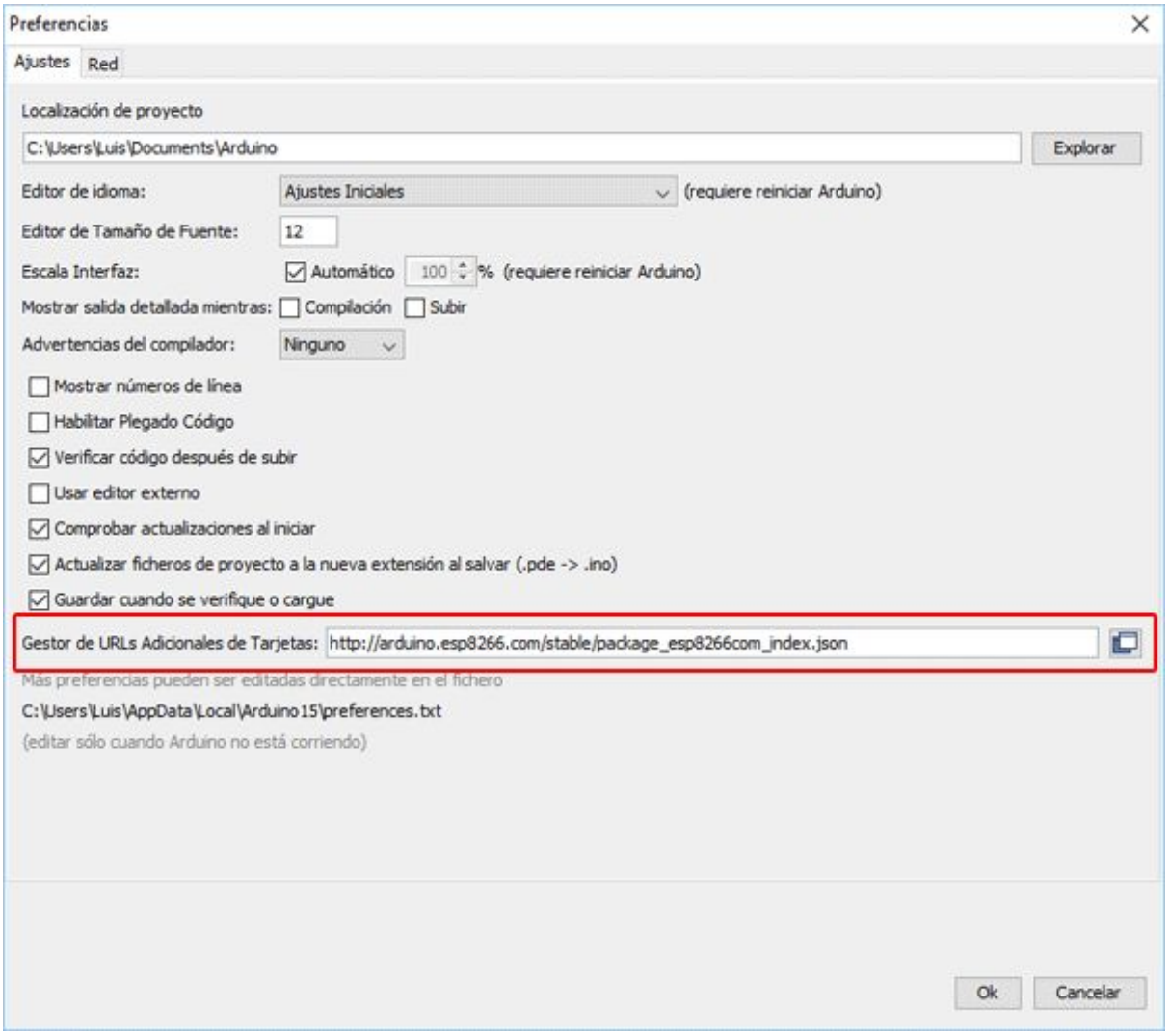
El lenguaje con el que vamos a programar es el nativo de Arduino. Seguramente encontremos diferencias en las librerías, pero son mínimas: vamos a poder **utilizar el 90% de ellas para el ESP8266**.

Paso 1: añadir la URL para placas adicionales

Abre las preferencias que se encuentran en *Archivo* > *Preferencias*

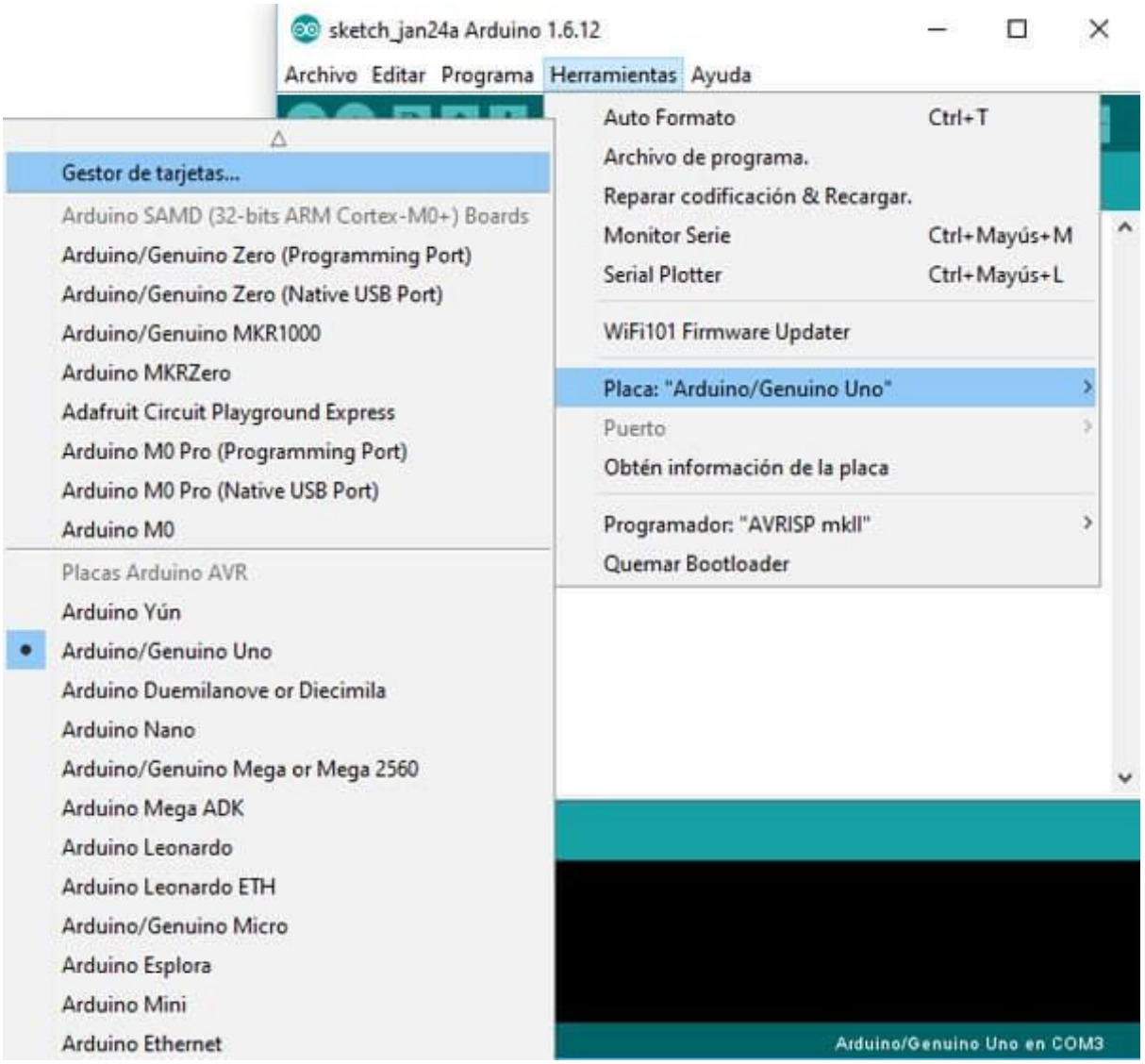


En donde pone *Gestor de URLs Adicionales de Tarjetas* copia el siguiente enlace:
http://arduino.esp8266.com/stable/package_esp8266com_index.json

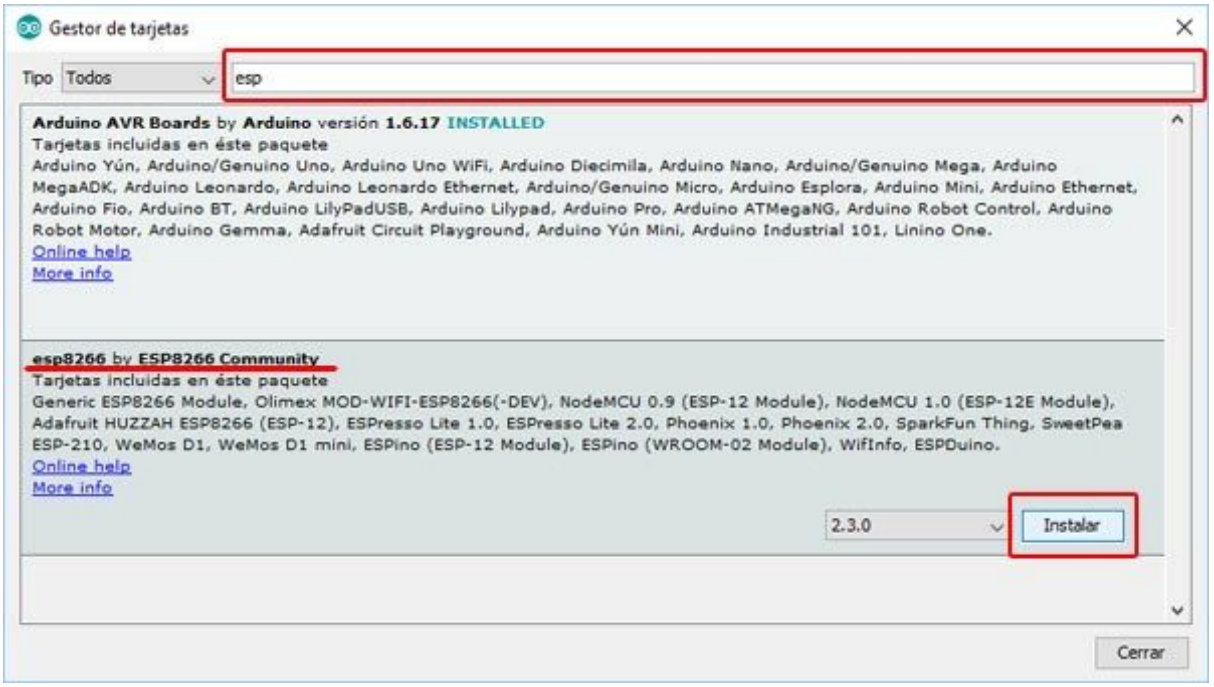


Y haces click en “Ok”.

Paso 2: añadir los drivers del ESP8266
Vete a *Herramientas > Placa: "Arduino UNO" > Gestor de Tarjetas...*

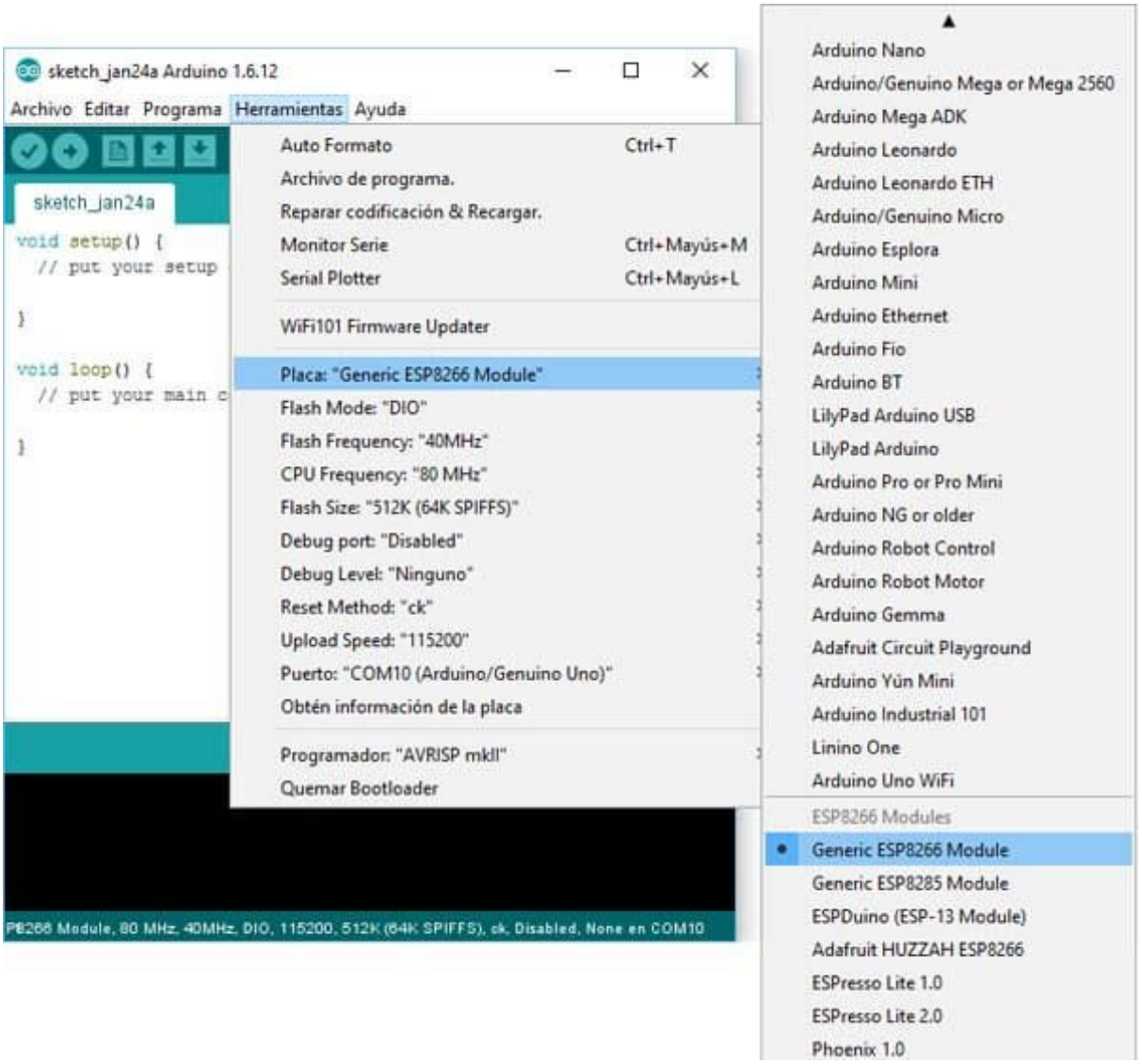


En el gestor de tarjetas buscas “esp” e instalas el ***ESP8266 by ESP8266 Community***



Paso 3: seleccionar la placa *Generic ESP8266 module*

Una vez hemos instalado la placa, ya podemos elegir entre los diferentes tipos de módulos ESP que se nos ofrecen: para utilizar el ESP-01 seleccionaremos ***Generic ESP8266 module***.



Y con estos pasos, ya tenemos el IDE de Arduino configurado para programar nuestra placa ESP-01.

Para este tipo de placas tenemos otros posibles parámetros de configuración que podemos dejar en sus valores por defecto:

- *Flash Mode* → "DIO"
- *Flash Frequency → "40MHz"*
- *CPU Frequency → "80MHz"*
- *Flash Size → "512K (64K SPIFFS)"*
- *Debug port → "Disabled"*
- *Debug Level → "Ninguno"*
- *Reset Method → "ck"*
- *Upload Speed → "115200"*

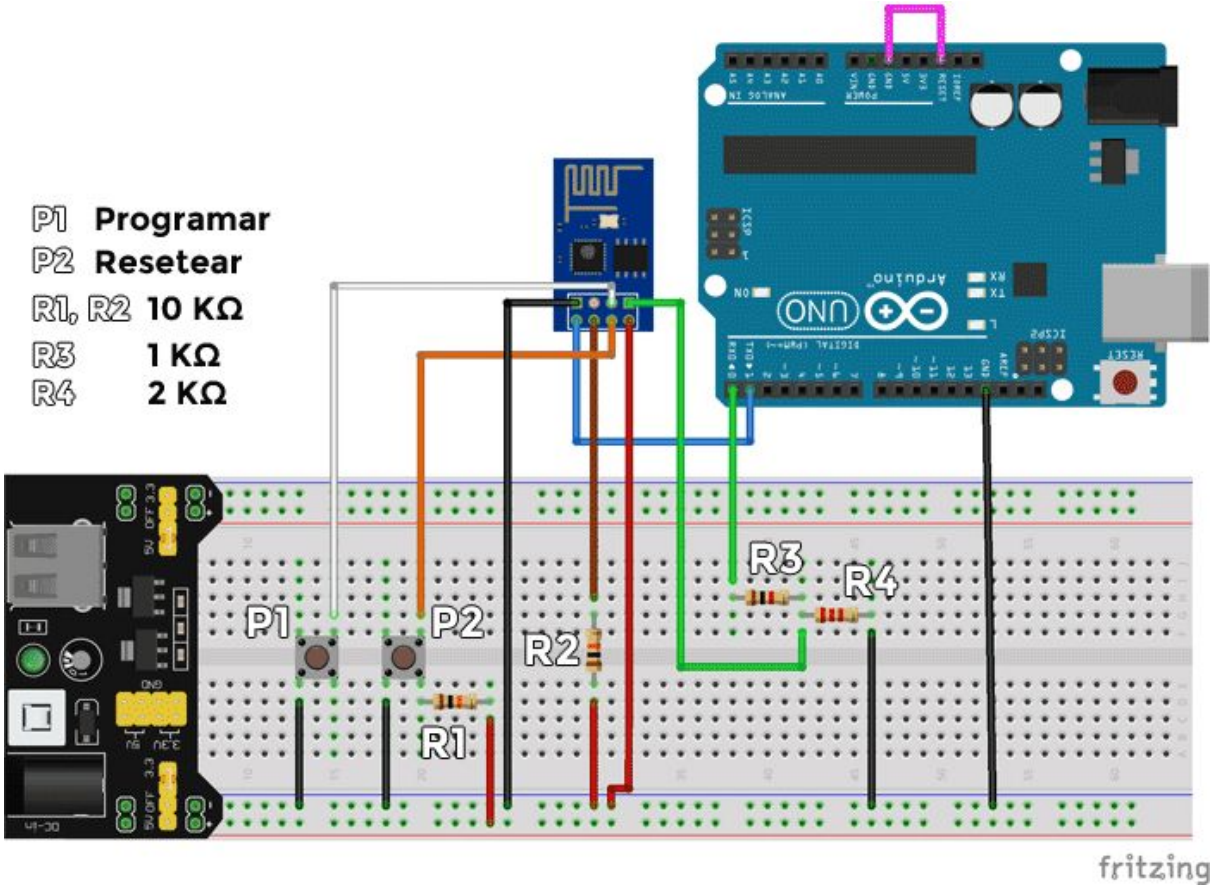
Programando el ESP-01 desde la placa de Arduino

Vamos a comenzar programando el ESP-01 desde la placa de Arduino UNO, usándola como mera pasarela (*passthrough*). Tenemos dos inconvenientes a la hora de utilizar este

método: por un lado la alimentación, ya que **no suministra los 200 mA necesarios**, y por otro lado el voltaje de operación, ya que **Arduino UNO funciona con 5 V** y esto también incluye a los pines RX y TX.

Con estas dos premisas necesitaremos una fuente de alimentación, como la que hemos visto antes la MB102, y hacer un **divisor de tensión para bajar el voltaje del pin RX** de Arduino y pasar de 5 V a 3,3 V y evitar así dañar el ESP-01.

Circuito eléctrico ESP-01 con Arduino



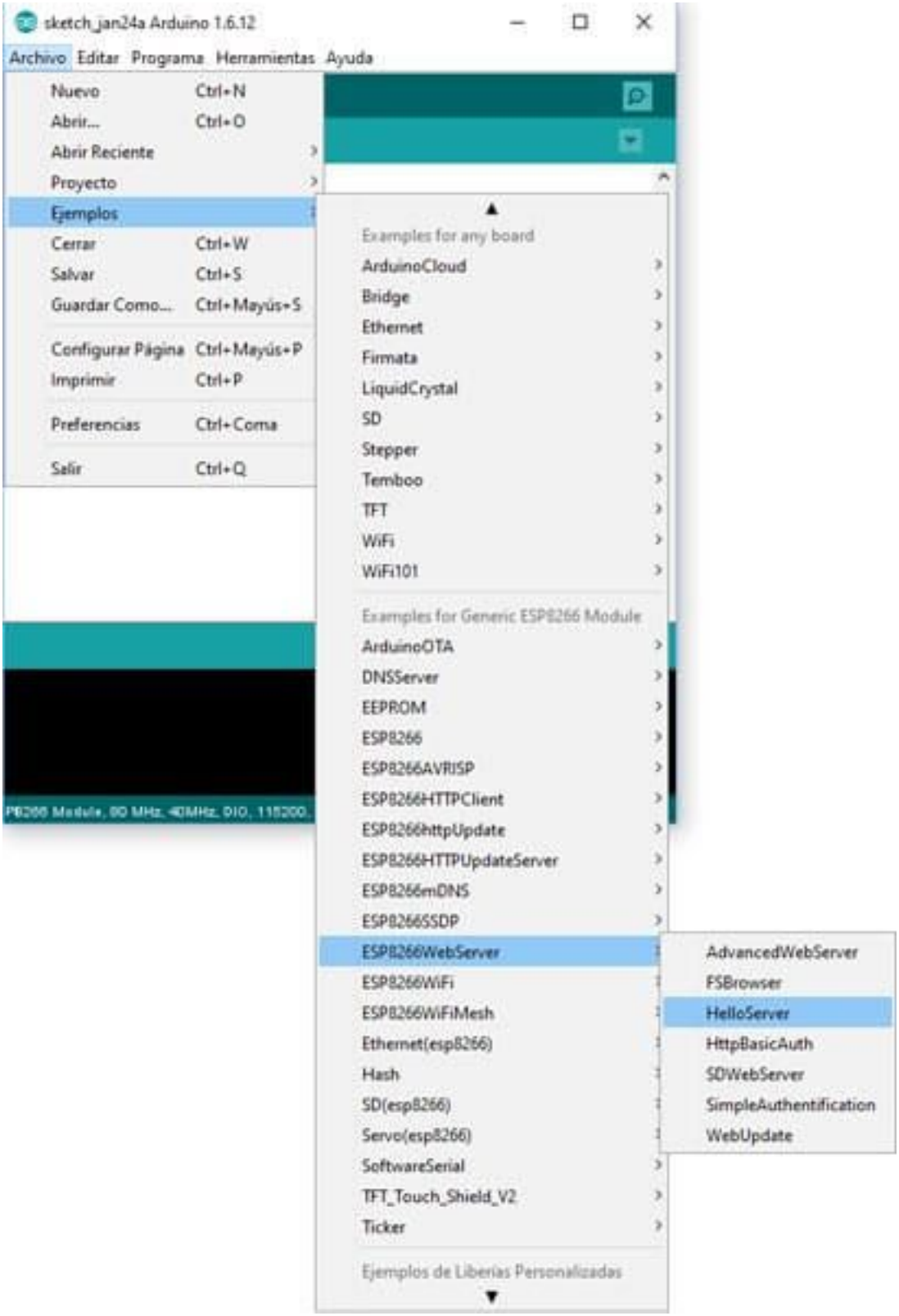
- El TX del Arduino conectado al TX del ESP-01
- El RX del Arduino conectado a la entrada del divisor de tensión
- La salida del divisor de tensión al RX del ESP-01
- Se hace un puente entre el pin GND y el pin RESET en el Arduino UNO (esto aísla al procesador de los pines GPIO, convirtiéndolo en una pasarela)
- Conectamos todas las GND entre sí
    - GND de Arduino
    - GND del ESP-01
    - La tierra del divisor de tensión
    - GND de la fuente de alimentación
- El pin CH_PD del ESP-01 a 3,3 V con una resistencia de 10 KΩ en serie
- El pin RESET del ESP-01 a 3,3 V con una resistencia de 10 KΩ en serie
- Para habilitar el modo de programación UART, conectamos el pin GPIO0

- Por último quedaría el pin GPIO2 que, como esta placa tiene una resistencia pull-up, lo podemos dejar al aire

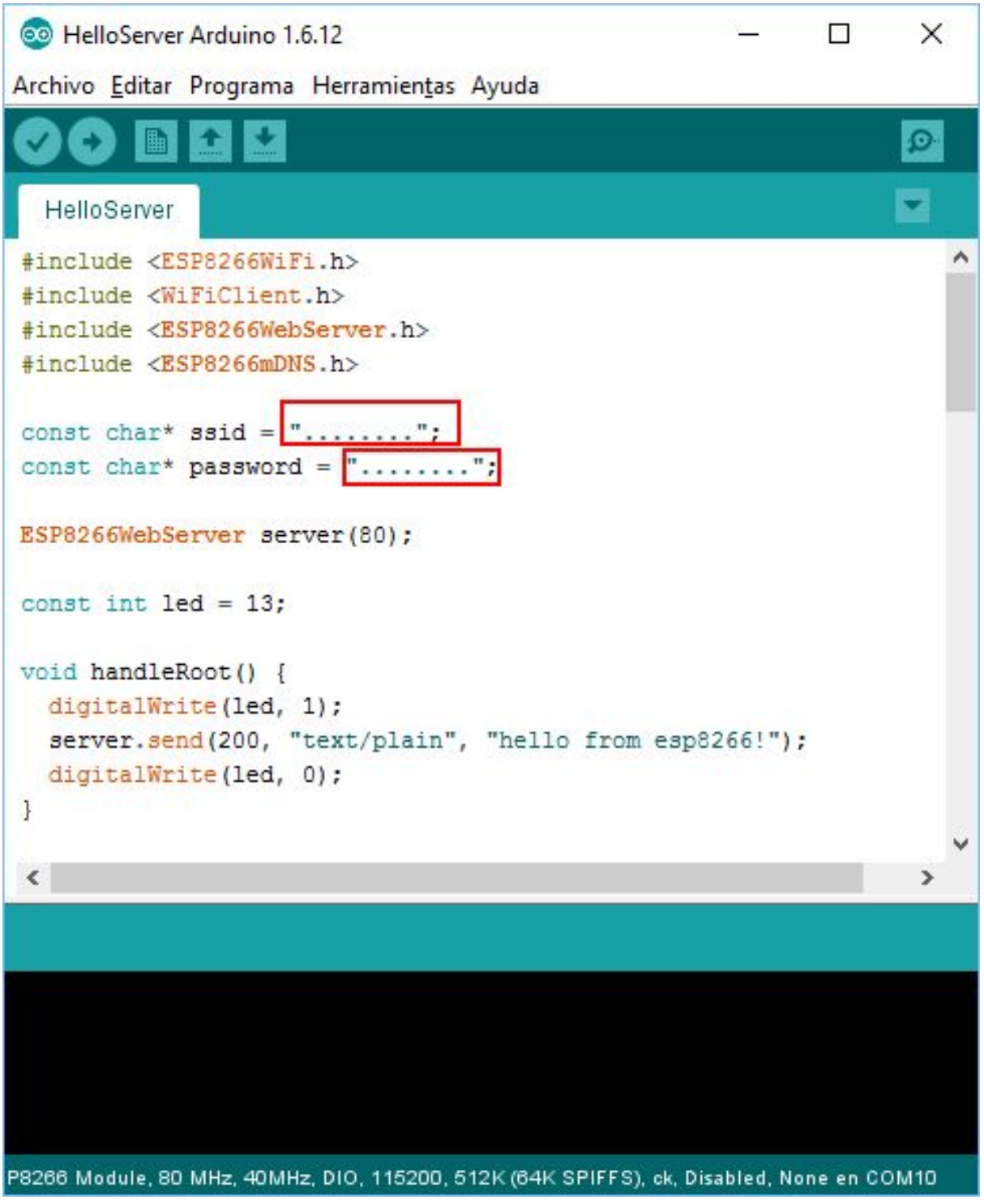
Probando el primer programa del ESP-01 con Arduino

Ahora ya lo tenemos todo preparado para hacer la primera prueba, pero antes tienes que **seleccionar en el IDE de Arduino el puerto donde tienes conectado el propio Arduino UNO**: *Herramientas* > *Puerto*.

Una vez lo tengas seleccionado vamos a cargar un ejemplo: *Archivo* > *Ejemplos* > *ESP8266WebServer* > *HelloServer*.



Sustituye los valores de *ssid* por el nombre de tu WiFi y *password* por la contraseña de tu WiFi.



Le das a subir programa y esperas a que termine de compilar y subir al ESP-01. Puede tardar un rato.

Abre el monitor serie que se encuentra en *Herramientas* > *Monitor Serie* y asegúrate de que tienes seleccionado *Sin ajuste de línea y 115200 baudio*.

Copia la IP que aparece y la pones en un navegador. El resultado que te tiene que dar es el siguiente.

Lo que hemos hecho es crear un *servidor web dentro del ESP-01*. Sí, es increíble que podamos **tener un servidor que nos cabe en un bolsillo y solo por 2€**.

Recomendaciones

Por último y para acabar con el método de ***programar un ESP-01 desde un Arduino***, te dejo unas indicaciones:

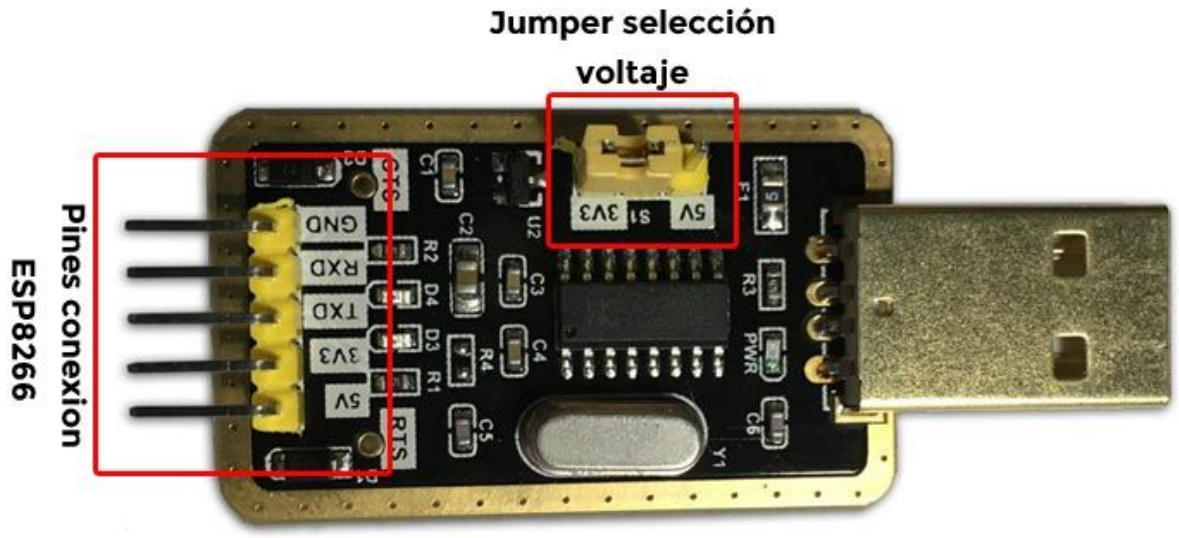
- Siempre que termina de cargarse un programa automáticamente se ejecuta, aunque estemos en modo UART (modo programación).
- Si reinicias el ESP-01 y vuelves a intentar acceder a la página web no podrás: está esperando a que vuelvas a cargar otro programa. Deberías cambiar el modo de funcionamiento en el ESP-01 a modo Flash (modo ejecución) desconectado el pin GPIO0.
- Una vez hayas terminado la lógica, cambia al modo Flash (desconectando el pin GPIO0) para que cuando se reinicie ejecute el programa automáticamente.

Programando el ESP-01 con un convertidor USB-serie

Ahora vamos con este segundo método que es bastante más sencillo ya que viene todo integrado dentro del propio circuito. Existen multitud de convertidores de este tipo llamados también **convertidor USB a TTL o FTDI**. Prácticamente todos funcionan de la misma forma salvo las características específicas de cada uno.

Es un convertidor entre protocolos. En este caso se utiliza para **convertir los datos estándar USB al puerto serie y viceversa**.

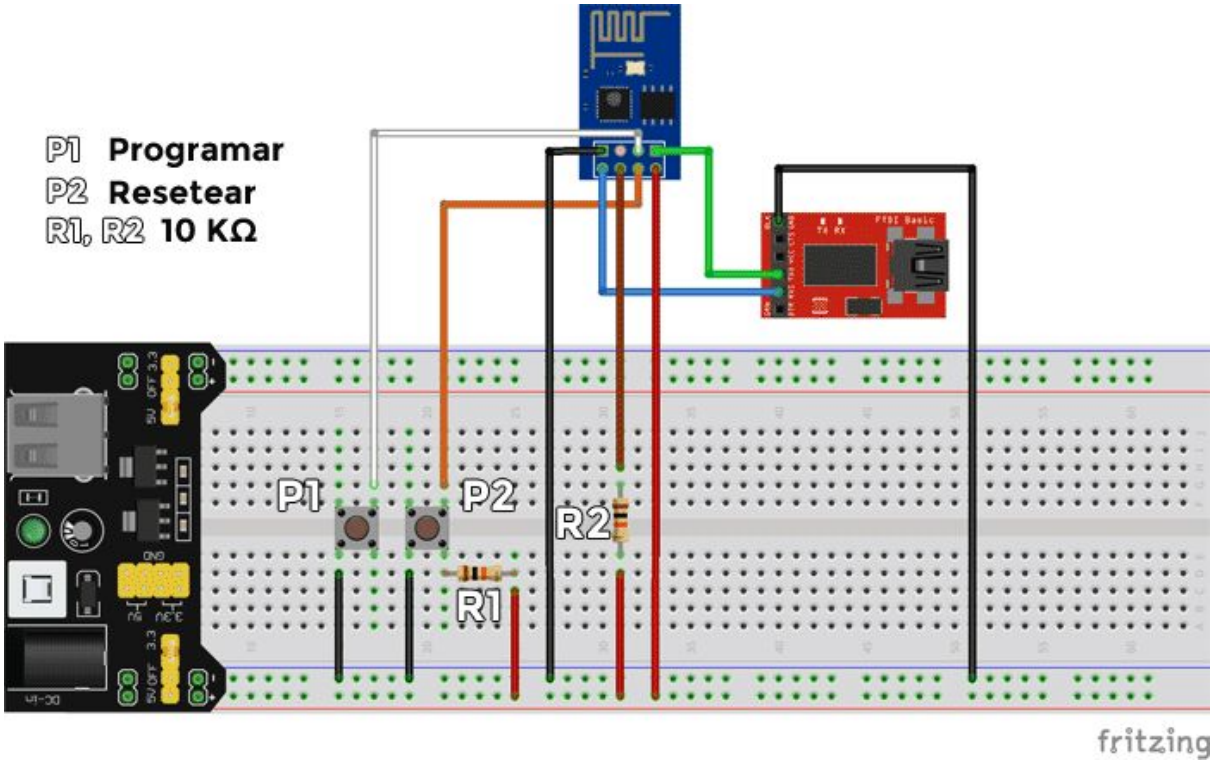
Hay algunos que trabajan a 3,3 V, otros a 5 V y otros pueden trabajar con los dos voltajes, como es mi caso. Yo tengo el modelo CH340G y tiene un *jumper* para seleccionar el voltaje con el que quieres trabajar.



Circuito eléctrico

Todos tienen **los mismos pines** de conexión, **Vcc, GND, RX y TX**. Por lo tanto la conexión se hace igual en todos los modelos. Como ya hemos visto antes, el ESP-01 tiene que ser alimentado con una fuente de alimentación dedicada y eso es lo que haremos en este método.

Además, he incorporado unos pulsadores que nos van a permitir cambiar de modo muy fácilmente. En el siguiente esquema te muestro un ejemplo.



Te explico ahora cómo conectar cada uno de los pines del ESP-01:

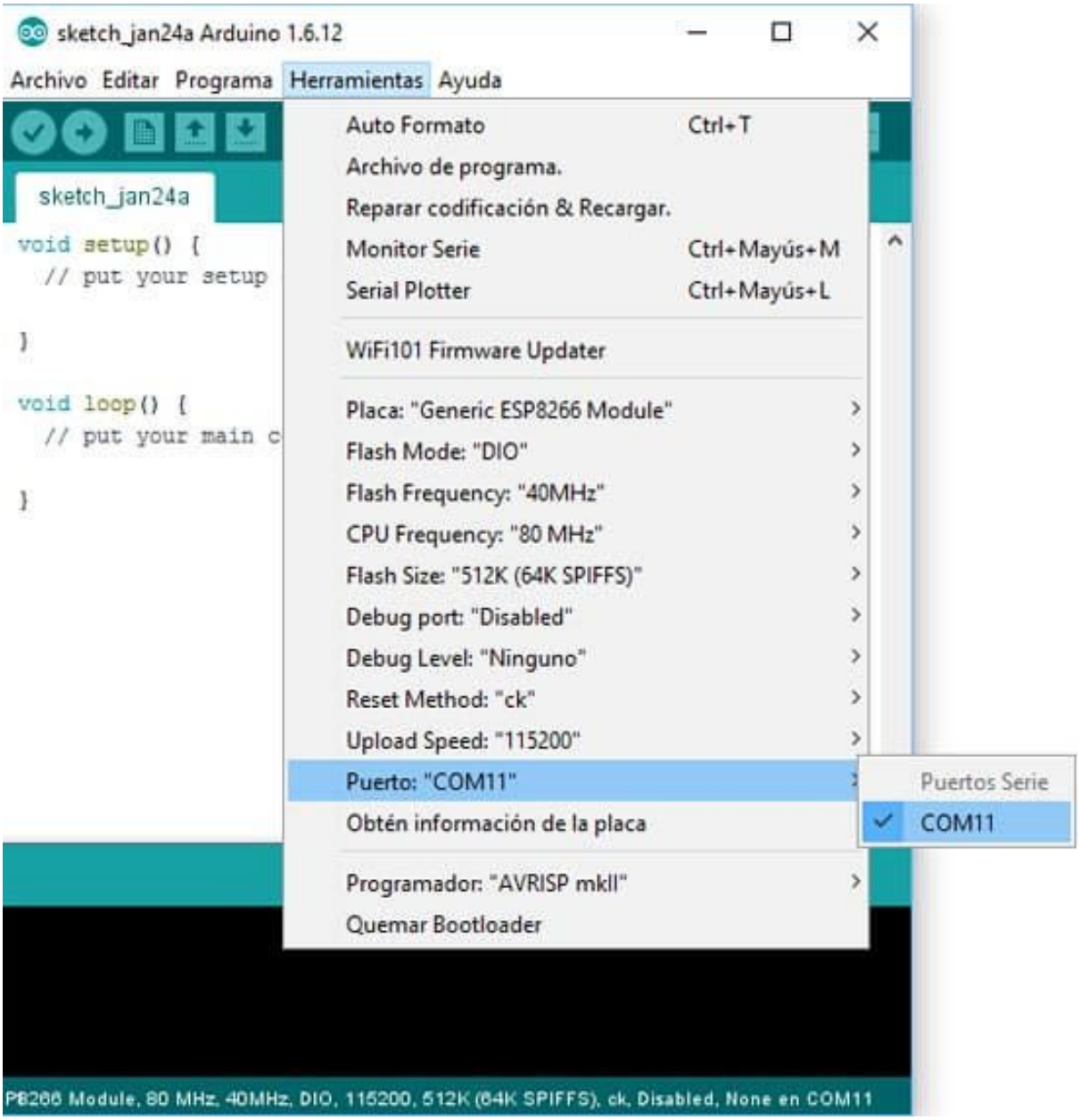
- Conectamos todas las masas a un mismo punto: la del ESP-01, fuente de alimentación y convertidor USB serie.
- El GPIO2 lo dejamos al aire: ya tiene una resistencia *pull-up* que mantendrá el estado a alto (*HIGH*).
- El GIPIO0 lo vamos a conectar a tierra a través de un pulsador. Igual que el pin anterior, tiene una resistencia pull-up que lo mantiene alto (*HIGH*). Para poder cargar un programa debemos tener este pin a bajo (*LOW*). Veamos los casos:
    - No se aprieta el pulsador: GPIO0 a *HIGH* y GPIO2 a *HIGH* → Modo Flash → ejecuta el programa cargado.
    - Se aprieta el pulsador: GPIO0 a *LOW* y GPIO2 a HIGH → Modo UART → a la espera de cargar un programa. (Pulsar antes de encender/resetear el ESP-01 y mantenerlo durante al menos 5 segundos).
- El RX del convertidor al TX del ESP-01.
- El TX del convertidor al RX del ESP-01.
- El pin CH_PD conectado a la fuente de alimentación (3,3 V) con una resistencia de 10 kΩ para evitar un exceso de consumo.
- El pin RESET lo conectamos a un pulsador con una resistencia *pull-up*.
    - Cuando no está pulsado, 3,3 V en la entrada RESET. No se resetea.
    - Cuando pulsamos, 0 V y reseteamos el ESP-01.
- El Vcc a los 3,3 V de la fuente de alimentación.

Este sería el circuito recomendado para poder trabajar de una manera fácil y cómoda. Con los pulsadores vamos a poder resetear y cambiar del modo el ESP-01.

Programación del ESP-01 con FTDI

La programación es igual que en el caso de que utilicemos un Arduino UNO: la configuración del IDE de Arduino es idéntica y la única precaución que tenemos que tener es a la hora de seleccionar el puerto.

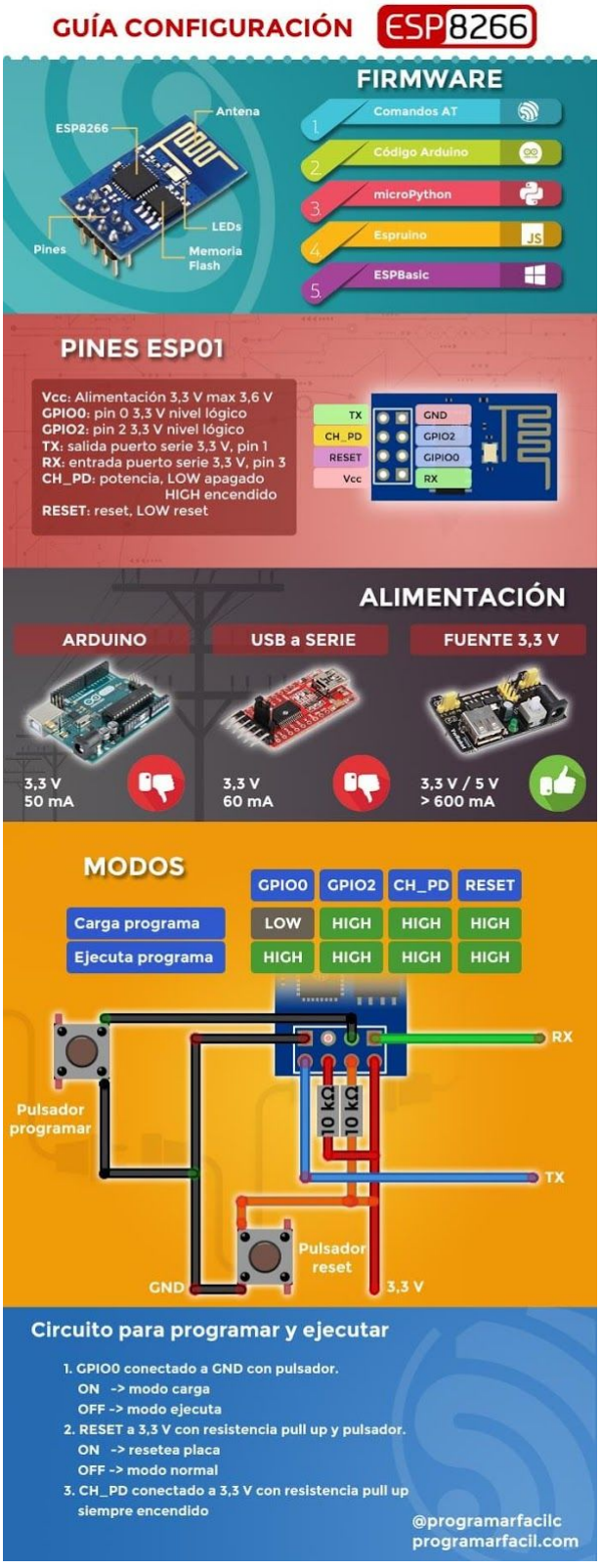
Como hemos conectado el FTDI a nuestro ordenador, ahora nos aparecerá un puerto nuevo dentro del IDE de Arduino sin nombre: éste es el que tenemos que seleccionar.



Una vez seleccionado ya puedes cargar el programa de ejemplo que hemos visto anteriormente.

Resumen de cómo configurar el ESP-01

Para finalizar y antes de darte los últimos consejos, te dejo esta infografía donde puedes ver un resumen de las partes más importantes de cómo configurar un ESP-01 basado en el microcontrolador ESP8266.



Recomendaciones finales

Para finalizar, unas recomendaciones a modo de resumen que deberías en cuenta en todo momento:

1. Por defecto solo se dispone de 512 kB de memoria Flash (aunque dependerá del modelo exacto de ESP-01 que tengas).
2. El ESP-01 dispone de 4 pines accesibles: aprovecha también los pines RX y TX una vez que hayas cargado el programa.
3. Todos los GPIO funcionan a 3,3 V.
4. No alimentes la placa con más de 3,6 V, la puedes dañar.
5. Intenta utilizar el protocolo I2C para comunicar con los sensores, así ahorrarás pines.
6. La corriente necesaria es de unos 200 mA cuando la WiFi está transmitiendo, no alimentes el ESP-01 con menos.
7. Si quieres que trabaje como interfaz WiFi de un Arduino, puedes utilizar el Firmware de comandos AT.
8. Existen otros Firmwares para otros lenguajes como Python, JavaScript o Basic.
9. Alimenta la placa con una fuente que suministre 3,3 V y más de 200 mA.
10. Si cargas los programas desde Arduino, asegúrate de utilizar un divisor de tensión para RX ya que puedes dañar la placa.
11. Si programas con un convertidor TTL recuerda que el RX del TTL va a TX del ESP-01 y viceversa.

Y con esto damos por concluida la guía sobre cómo trabajar con el ESP-01 de principio a fin. Ahora depende de ti crear nuevos proyectos con esta placa basada en el ESP8266.