Vacas Martinez, José Antonio

# Tema 10 - Comunicaciones

En este tema vamos a ver muchas formas de comunicación que podemos utilizar con Arduino. Algunas nos dan cortos alcances como el bluetooth y los infrarrojos y otros nos permitirán conectarlos a dispositivos remotos vía wifi o ethernet

## Comunicaciones serie

El protocolo de comunicaciones más sencillo es que ya hemos usado: el protocolo serie.

La forma natural de comunicarse con Arduino es utilizando el puerto serie, siendo este el método de depuración más usado.

En el [siguiente vídeo](https://youtu.be/A6BH4cfqS4k) veremos detalles sobre este protocolo, cómo usarlo en Arduino y cómo leer datos desde Arduino.

Actualización: en el vídeo no aparecen los métodos readString y readStringUntil, muy útiles de la librería Serial. Nos facilitan enormemente el recuperar cadena desde el puerto serie.

Os dejo una presentación donde se trata en más detalle "Tema 10 - Comunicaciones serie.pdf"

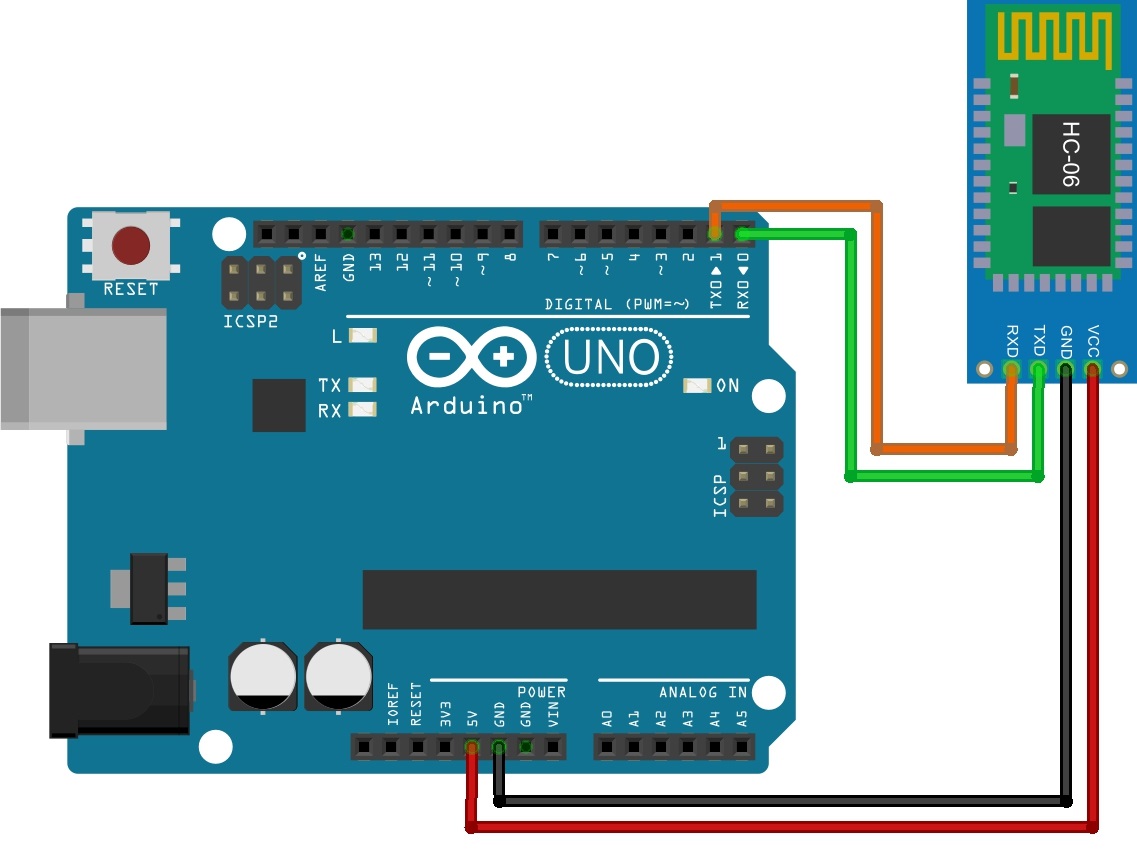
## Comunicaciones bluetooth

Una vez podemos comunicarnos vía puerto serie el pasar a comunicaciones inalámbricas es  sencillo con bluetooth.

Existen unos dispositivos capaces de enviar al dispositivo con el que están emparejados los datos que ellos reciben por sus pines.

En esta imagen podemos ver uno de ellos. Bajo este aspecto existen diferentes tipos de dispositivos.  Para hacer lo que pretendemos basta la versión más básica, que será un bluetooth de tipo esclavo (los master pueden costar hasta 4 veces más)

La conexión con Arduino es muy sencilla (sólo hay que tener en cuenta que hay que conectar con los cables cruzados: TX-RX y RX-TX)



Conexion

El código para controlar un rele via bluetooth es muy sencillo

const int rele=9;  
  
void setup() {  
 Serial.begin(9600);  
 pinMode(rele,OUTPUT);  
}  
  
void loop() {  
 while(Serial.available()>0)  
 {  
 int iChar=Serial.read();  
 Serial.println((char)iChar);  
 if(iChar=='1') // Un caracer '1' lo enciende y cualquier otro lo apaga  
 digitalWrite(rele,HIGH);  
 else  
 digitalWrite(rele,LOW);  
 }  
}

Para controlar desde el móvil, debemos emparejar el Bluetooth con nuestro móvil y utilizando alguna aplicación que nos permita enviar datos como por ejemplo Blueterm. Existen mucha aplicaciones que nos permiten controlar muchas funciones, basta con buscar "arduino bluetooth" en google play

En [este vídeo](https://youtu.be/N04DzxnTXqA) vemos cómo utilizarlos y configurarlos

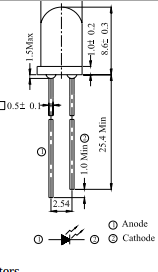
En el [siguiente vídeo](https://youtu.be/j5j-pfIGdoM) vamos a usar un dispositivo bluetooth para activar un rele remotamente desde un móvil

## Comunicaciones infrarrojas

Una de las formas más habituales de control remoto de dispositivos sobre todo en el entorno doméstico son los controles infrarrojos. Casi todos los mandos a distancia domésticos los utilizan. Existen varios protocolos (casi tanto como fabricantes) de comunicaciones infrarrojas https://en.wikipedia.org/wiki/Infrared\_remote\_control.

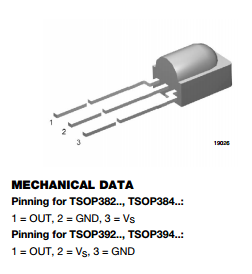
Utilizan luz infrarroja de 940 nm de longitud de onda, modulándolo con una onda de 38kHz. El hardware necesario para utilizar estas comunicaciones es extremadamente barato. El único inconveniente que tiene es que es muy susceptible a interferencia por luces o por el sol.

Para realizar la comunicación necesitaremos un emisor



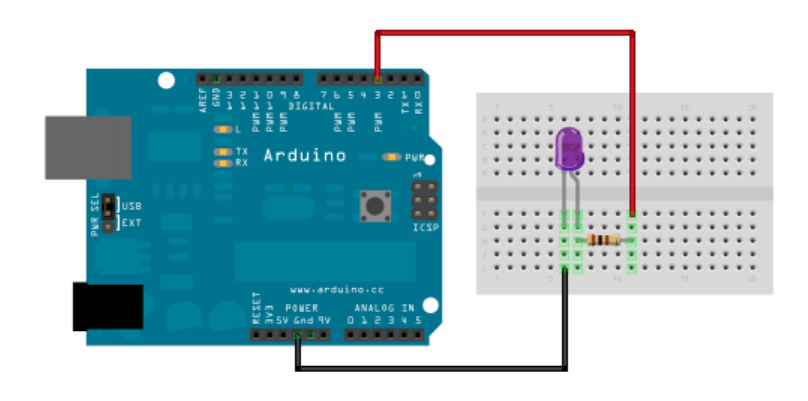
Emisor IR

Y un receptor



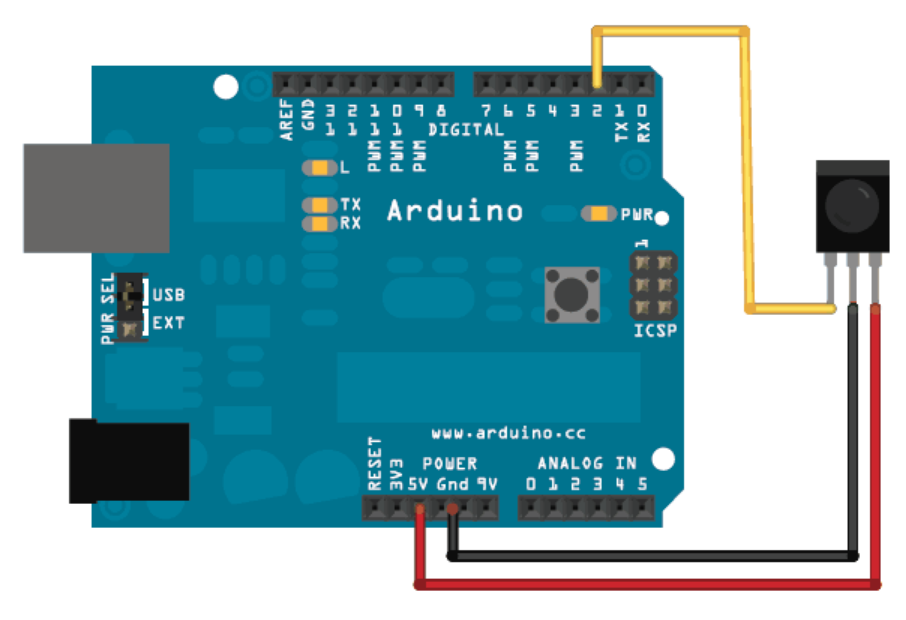
Receptor\_Ir.png

Los montajes eléctricos son muy sencillos. Para el emisor:



Montaje\_EmisorIR.png

Y el montaje para el receptor (hay que ver el patillaje concreto de nuestro modelo)



Montaje\_Receptor\_IR.png

Para controlarlo con Arduino la forma más sencilla es usar la librería IRremote.

Ejemplo de código para enviar códigos para encender una TV de marca Sony (emisor en el pin 12)

#include <IRremote.h>  
  
 IRsend irsend;  
  
 // Sony TV power code  
 irsend.sendSony(0xa90, 12);

Código para recibir códigos (receptor en pin 11) y mostrarlos por el monitor Serie

#include <IRremote.h>  
  
int RECV\_PIN = 11;  
  
IRrecv irrecv(RECV\_PIN);  
  
decode\_results results;  
  
void setup()  
{ Serial.begin(9600);  
 irrecv.enableIRIn(); // Start the receiver}  
  
void loop() {  
 if (irrecv.decode(&results)) {  
 Serial.println(results.value, HEX);  
 irrecv.resume(); // Receive the next value }  
}

En el [siguiente vídeo](https://www.youtube.com/embed/S-wWNDfEPqw) vemos todo esto en detalle

## Procotocolos de comunicaciones

En el [siguiente vídeo](https://www.youtube.com/embed/DePCak9WNPM) hablaremos sobre algunos de los protocolos de comunicaciones más usados, como son I2C y SPI.

La presentación está disponible en el fichero "Tema 10 - Protocolos de comunicaciones.pdf"

### Pines SPI

¿Dónde están los pines SPI en las distintas placas?



Pines SPI

Uno de los protocolos de comunicaciones más usados en Arduino es el SPI. Se utiliza para comunicarse con multitud de dispositivos. Cómo norma general, la manera más sencilla de encontrar los pines SPI es en el conector ICSP. Algunas placas los duplican entre los pines digitales, pero otras no.

|  |  |  |  |
| --- | --- | --- | --- |
| Placa | SCK | MOSI | MISO |
| Arduino UNO | D13 | D11 | D12 |
| Arduino Leonardo | D3 | D4 | D1 |
| Arduino Mega | D52 | D51 | D50 |

## Ethernet

Si de verdad queremos comunicar nuestro proyecto con internet necesitamos un shield ethernet.

Existen básicamente dos tipos de shields, los basados en el chip Wiznet W5100

[Ethernet\_5100.png](./images/Ethernet_5100.png)

y en la librería  Ethernet de  Arduino’s IDE  (cuya documentación podemos encontrar http://arduino.cc/en/Reference/Ethernet) y los basados en el chip enc28J60 http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889



c28j60.png

y que utilizan la librería ethercard https://github.com/jcw/ethercard (de JeeLabs).

Las dos admiten esencialmente la misma funcionalidad y se conectan a Arduino por SPI. Sin embargo la diferencia de coste es sorprendente. ¿A qué se debe?:

A que el Wiznet W5100 incluye el stack TCP-IP completamente en hardware mientras que en el enc28J60 es necesario que nuestro microcontrolador realice todas las operaciones por software. Es decir el W5100 libera a nuestro micro de la mayor parte del trabajo duro del acceso a internet, mientras que con el enc28J60, tiene que sufrirlo entero.

Por tanto si tenemos que realizar un proyecto donde se usará mucho acceso a internet claramente necesitaremos la solución hardware.

En [este vídeo](https://youtu.be/s8D1DZWbubg) vamos a ver los detalles de como usar cada uno de ellos.

En el [siguiente vídeo](https://youtu.be/6W_2I--XwrY) comentamos el hardware necesario para comunicarnos por internet

## Wifi

Existen diversas formas de dar funcionalidad Wifi a Arduino

* Podemos usar placas Arduino que lo incorporan, como son la Arduino Uno Wifi, la Arduino YUN o MKR
* Usar un shield Wifi
* Conectar un dispositivo Wifi que nos proporcione esa funcionalidad con el ESP8266

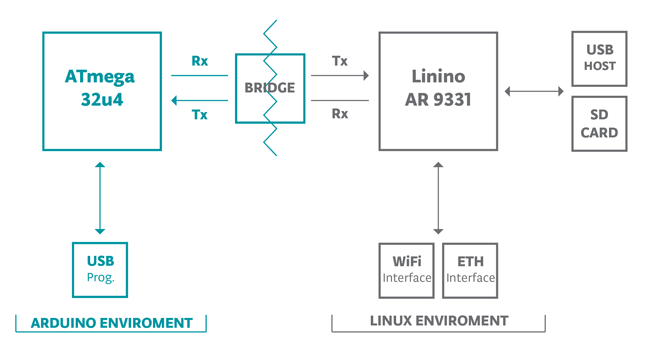
## Arduino YUN

Yún significa nube y precisamente esa es la vocación de esta placa, conectar con servicios en la nube de una forma sencilla para el usuario.

Este dispositivo fue el primero de varios  dispositivos Arduino que incluían en una misma placa un Arduino tradicional (normalmente Leonardo) con los habituales conectores y un equipo más potente capaz de ejecutar un sistema operativo potente como Linux. El sistema Línux incluye una distribución llamada Linino http://arduino.cc/en/Main/ArduinoBoardDue que basada en Open-WRT

De esta forma podemos usar todo lo desarrollado previamente, tanto en software como en hardware y añadir nuevas funcionalidades avanzadas, sobre todo en el aspecto de conectividad.

Estos dos dispositivos están conectados por un interface llamado Bridge, tanto a nivel hardware como software (la librería bridge)

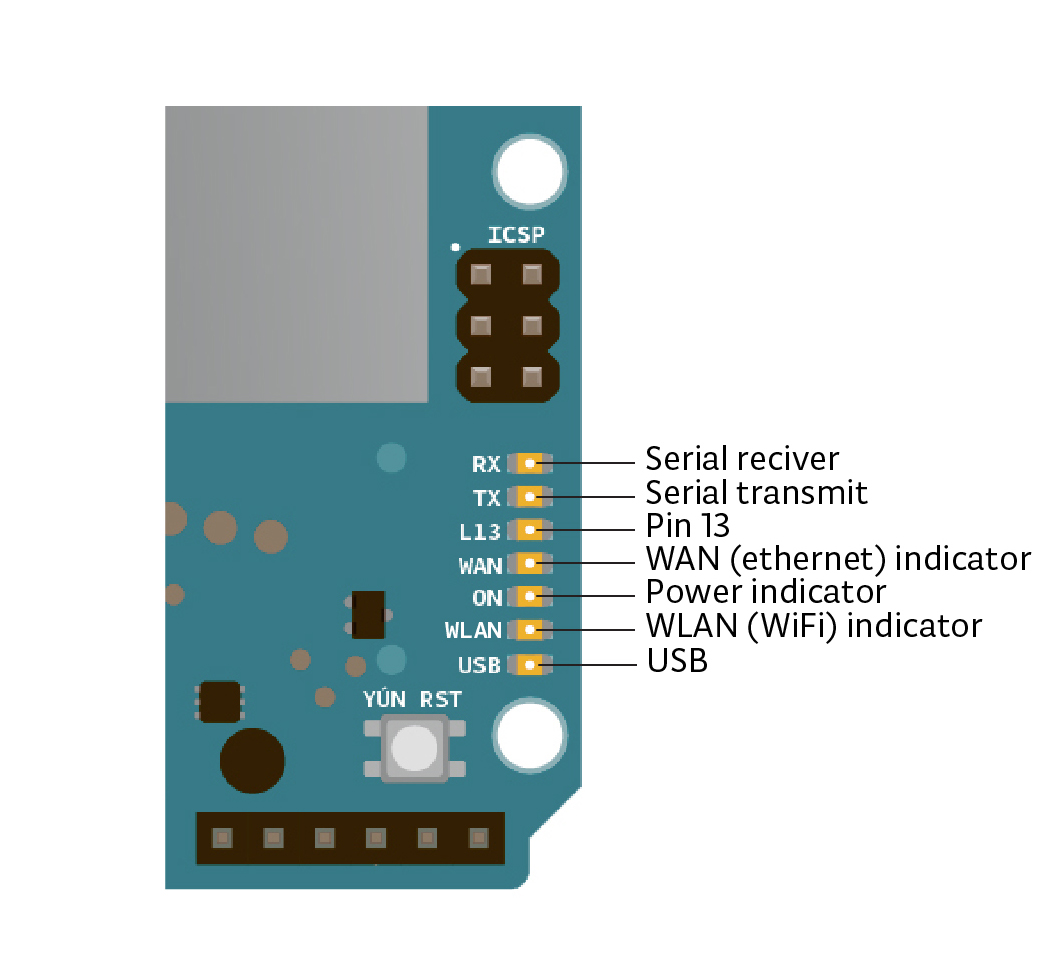


ArquitecturaYUN.png

Veamos las características del Arduino ArquitecturaYUN

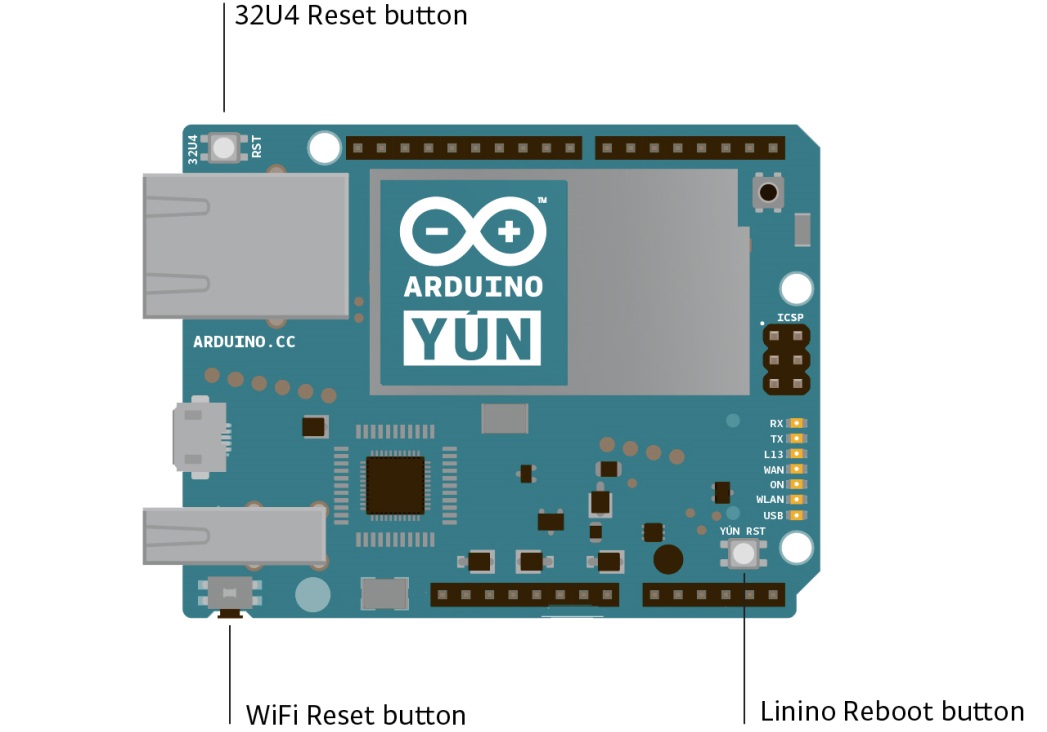
|Arduino || |---|---| |Microcontroller|ATmega32u4 |Operating Voltage|5V |Input Voltage|5V |Digital I/O Pins|20 |PWM Channels|7 |Analog Input Channels|12 |DC Current per I/O Pin|40 mA |DC Current for 3.3V Pin|50 mA |Flash Memory|32 KB (of which 4 KB used by bootloader) |SRAM|2.5 KB |EEPROM|1 KB |Clock Speed|16 MHz |Linux || |Processor|Atheros AR9331 |Architecture|MIPS @400MHz |Operating Voltage|3.3V |Ethernet|IEEE 802.3 10/100Mbit/s |WiFi|IEEE 802.11b/g/n |USB Type-A|2.0 Host/Device |Card Reader|Micro-SD only |RAM|64 MB DDR2 |Flash Memory|16 MB |PoE| compatible 802.3af card support   Yún dispone de un interface Ethernet y Wifi y USB Host, lo que le proporciona una conectividad extraordinaria. En breve seguro que se publican aplicaciones donde se conecta una cámara al USB pudiendo acceder a ella por Wifi.

Dada la complejidad del equipo dispone de muchos LEDs indicadores



ArduinoYUNLeds.png

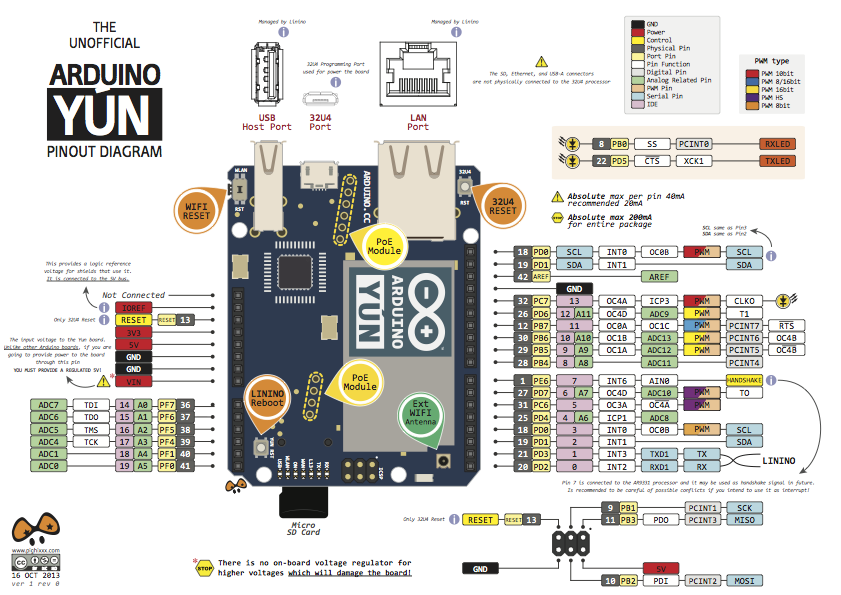
Y de 3 pulsadores Reset: • Uno para el equipo Arduino • Otro para el equipo Linux • Otro para el wifi. Si realizamos una pulsación suficientemente larga sobre ellos se devuelve al sistema hasta su configuración y estado por defecto (borrando los cambios que hayamos hecho)



ArduinoYUNButtons.png

Una de las novedades de Arduino Yún es que permite ser programado inalámbricamente, via Wifi, lo que se conoce como OTA (igual que los móviles). Al conectarse a nuestro wifi aparecerá un puerto más que representa esta conexión wifi.

Vamos a incluir el pinout completo de la placa



Pinout\_yun.png

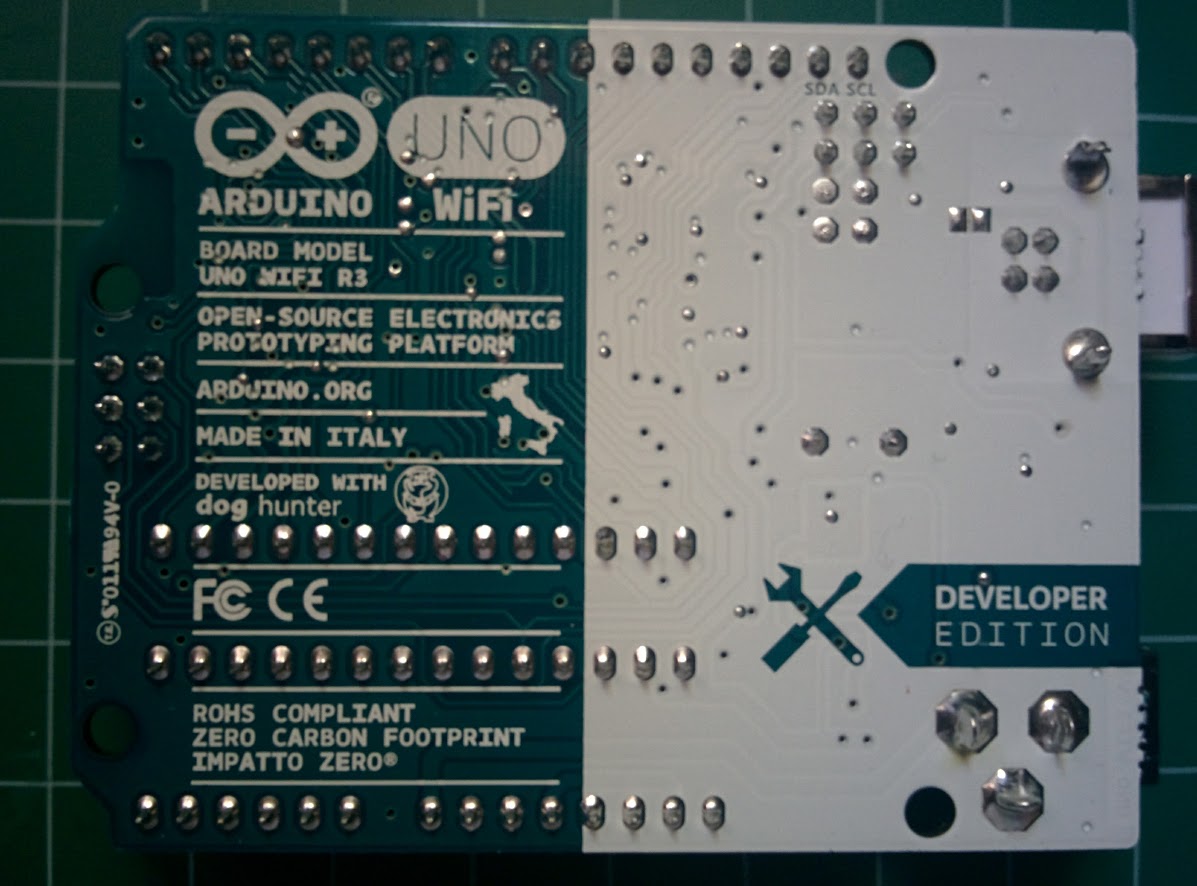
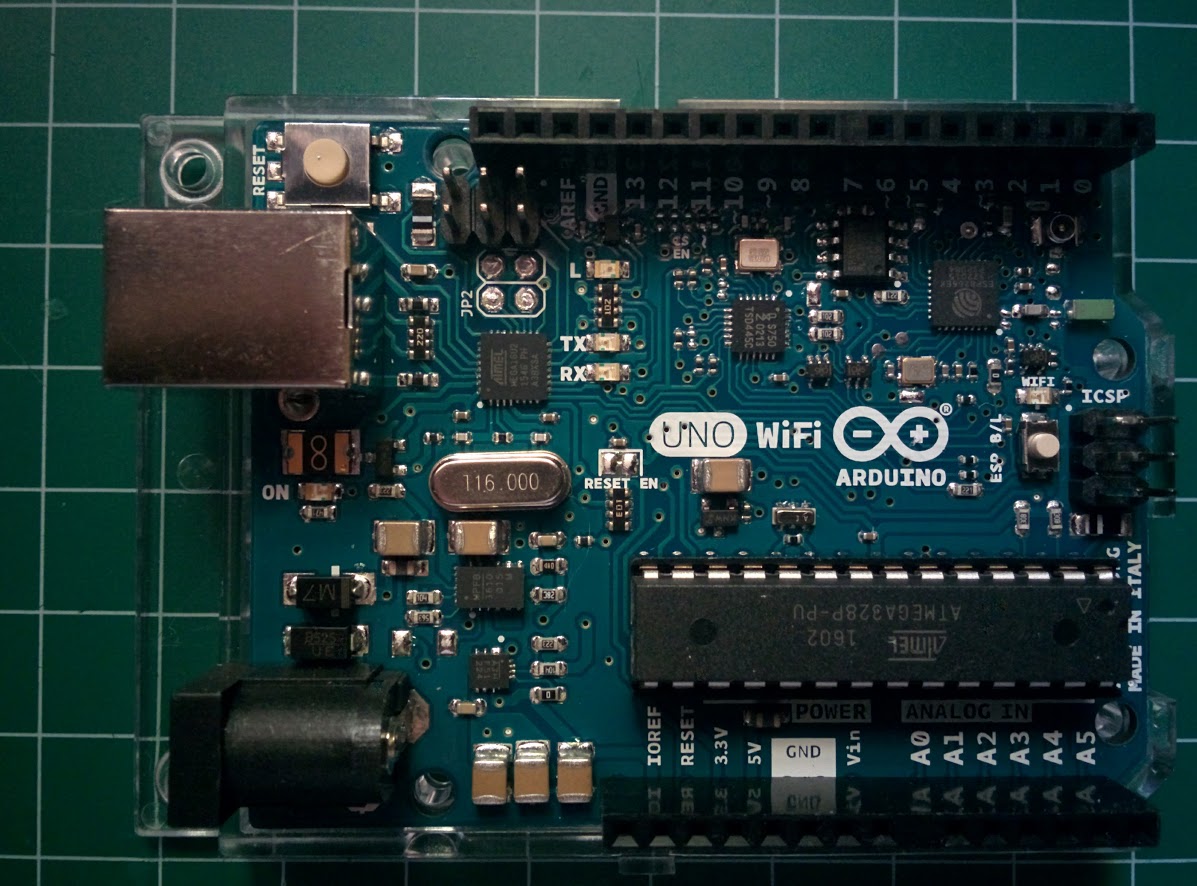
A continuación un par de vídeos sobre Yún

A continuación un par de vídeos sobre Yún [Primeras impresiones Arduino Yún I](https://youtu.be/i5Pzz_ENyag) [Primeras impresiones Arduino Yún II](https://youtu.be/UeSwu1LSrac)

## Arduino Uno Wifi

Vamos a ver ahora otra placa que incorpora wifi la [Arduino UNO Wifi](http://www.arduino.org/products/boards/arduino-uno-wifi)

La placa [Arduino UNO Wifi](http://www.arduino.org/products/boards/arduino-uno-wifi) no es otra cosa que una placa Arduino UNO con un [ESP8266](http://www.esp8266.com/) integrado que le proporciona la conectividad Wifi y que están conectados entre si por medio I2C con lo que tenemos disponible todos los pines.

### ¿Wifi?

Como hemos dicho la placa incluye un ESP8266 que nos permite tanto crear una red Wifi como conectarnos a una dada. [¿Cómo configurar el wifi?](http://www.arduino.org/learning/getting-started/getting-started-with-arduino-uno-wifi)

### Características

Microcontrolador: ATMega328 @ 16 MHz  
Módulo Wifi ESP8266EX @ 80 MHz 802.11 b/g/n 2.4 GHz  
Voltaje de funcionamiento (pines I/O): 5V  
Alimentación: 7 a 12 V  
Memoria Flash: 32 Kb  
SRAM: 2Kb  
Pines I/O: 20  
Salidas PWM: 6  
Consumo: 93 mA  
Pines analógicos: 6  
EEPROM: 1Kb

### Uso

Para usarla tenemos que tener una versión moderna del ide (Posterior a la 1.6.x)

A partir de aquí podemos programar la placa de la forma standard.

En [este vídeo veremos como usarlo](https://youtu.be/Mp81ekoXV4Y)

## Código

Como siempre tenemos disponibles ejemplos al seleccionar la placa

### Webserver

Se trata de poder ver el valor de las entradas analógica desde un navegador web conectándolos a la ip del Arduino UNO Wifi.

#include <Wire.h>  
#include <ArduinoWiFi.h>  
  
// Nos conectamos a la IP de la placa arduino (192.168.240.1 por defecto)   
// Para ello nos conectamos a una red wifi que crea con nombre Arduinoxxxxx  
  
void setup() {  
 Wifi.begin();  
 Wifi.println("WebServer Server arrancado");  
}  
  
void loop() {  
  
 while(Wifi.available()){  
 process(Wifi);  
 }  
 delay(50);  
}  
  
void process(WifiData client) {  
 // Leemos el comando (petición)  
 String command = client.readStringUntil('/');  
  
 if (command == "webserver") {  
 WebServer(client);  
 }  
}  
  
void WebServer(WifiData client) {  
 client.println("HTTP/1.1 200 OK");  
 client.println("Content-Type: text/html");  
 client.println("Connection: close");   
 client.println("Refresh: 20"); // refresh the page automatically every sec  
 client.println();   
 client.println("<html>");  
 client.println("<head> <title>UNO WIFI Example</title> </head>");  
 client.print("<body>");  
  
 for (int analogChannel = 0; analogChannel < 4; analogChannel++) {  
 int sensorReading = analogRead(analogChannel);  
 client.print("analog input ");  
 client.print(analogChannel);  
 client.print(" is ");  
 client.print(sensorReading);  
 client.print("<br/>");  
 }  
  
 client.print("</body>");  
 client.println("</html>");  
 client.print(DELIMITER); // very important to end the communication !!!   
}

### Webserver con capacidad de controlar pines

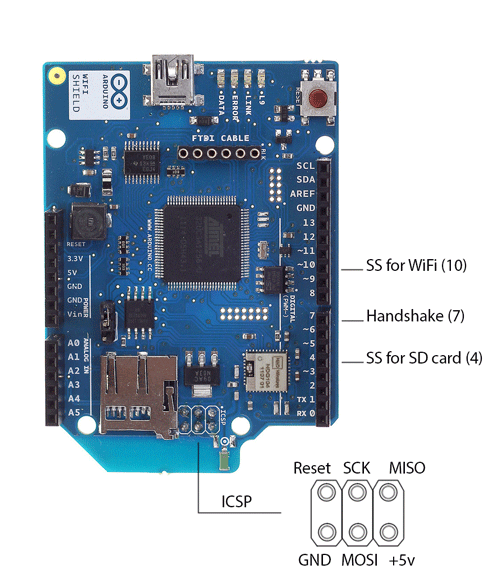
En este ejemplo podemos encender/apagar el pin 13 del arduino Wifi remotamente

#include <Wire.h>  
#include <ArduinoWiFi.h>  
  
// Nos conectamos a http://<IP>/arduino/webserver/  
  
void setup() {  
 pinMode(13,OUTPUT);  
 Wifi.begin();  
 Wifi.println("WebServer Server is up");  
}  
void loop() {  
  
 while(Wifi.available()){  
 process(Wifi);  
 }  
 delay(50);  
}  
  
void process(WifiData client) {  
 // read the command  
 String command = client.readStringUntil('/');  
  
 // is "digital" command?  
 if (command == "webserver") {  
 WebServer(client);  
 }  
  
 if (command == "digital") {  
 digitalCommand(client);  
 }  
}  
  
void WebServer(WifiData client) {  
  
 client.println("HTTP/1.1 200 OK");  
 client.println("Content-Type: text/html");  
 client.println();  
 client.println("<html>");  
  
 client.println("<head> </head>");  
 client.print("<body>");  
  
 client.print("Click<input type=button onClick=\"var w=window.open('/arduino/digital/13/1','\_parent');w.close();\"value='ON'>pin13 ON<br>");  
 client.print("Click<input type=button onClick=\"var w=window.open('/arduino/digital/13/0','\_parent');w.close();\"value='OFF'>pin13 OFF<br>");  
  
 client.print("</body>");  
 client.println("</html>");  
 client.print(DELIMITER); // very important to end the communication !!!  
  
}  
  
void digitalCommand(WifiData client) {  
 int pin, value;  
  
 // Read pin number  
 pin = client.parseInt();  
  
 // If the next character is a '/' it means we have an URL  
 // with a value like: "/digital/13/1"  
 if (client.read() == '/') {  
 value = client.parseInt();  
 digitalWrite(pin, value);  
 }  
  
 // Send feedback to client  
 client.print(F("Pin D"));  
 client.print(pin);  
 client.print(F(" set to "));  
 client.print(value);  
 client.print(EOL);  
  
}

## Shield Wifi

Hasta hace poco las placas wifi para arduino eran caras (entre los 60€ y 70€), consumían mucho y no funcionaban demasiado bien. Tengo en casa varias y ninguna de ellas es realmente operativa.

Una de las mas usadas ha sido el [Arduino Wifi Shield](https://www.arduino.cc/en/Guide/ArduinoWiFiShield)



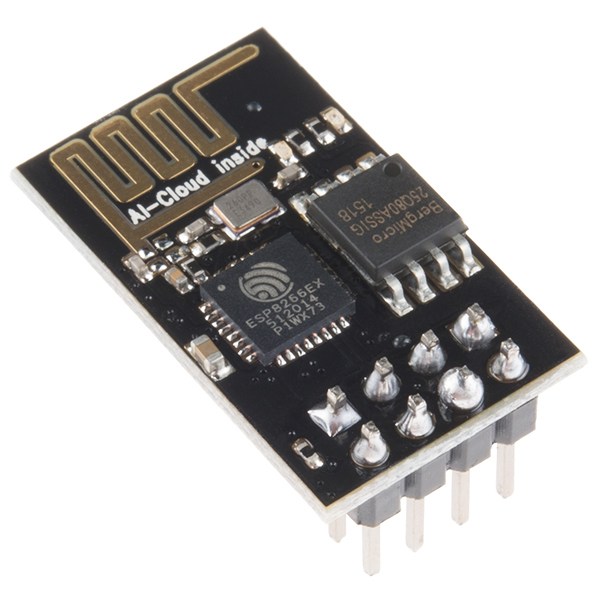
https://www.arduino.cc/en/uploads/Guide/wifi\_shield3.png

Que tiene la ventana de incluir un lector de tarjetas SD

La arduino UNO Wifi me ha parecido una placa muy interesante, con buenas prestaciones, un precio muy ajustado y que facilita totalmente acceso wifi, no obstante vamos a ver otra forma de proporcionar conectividad wifi más sencilla y barata

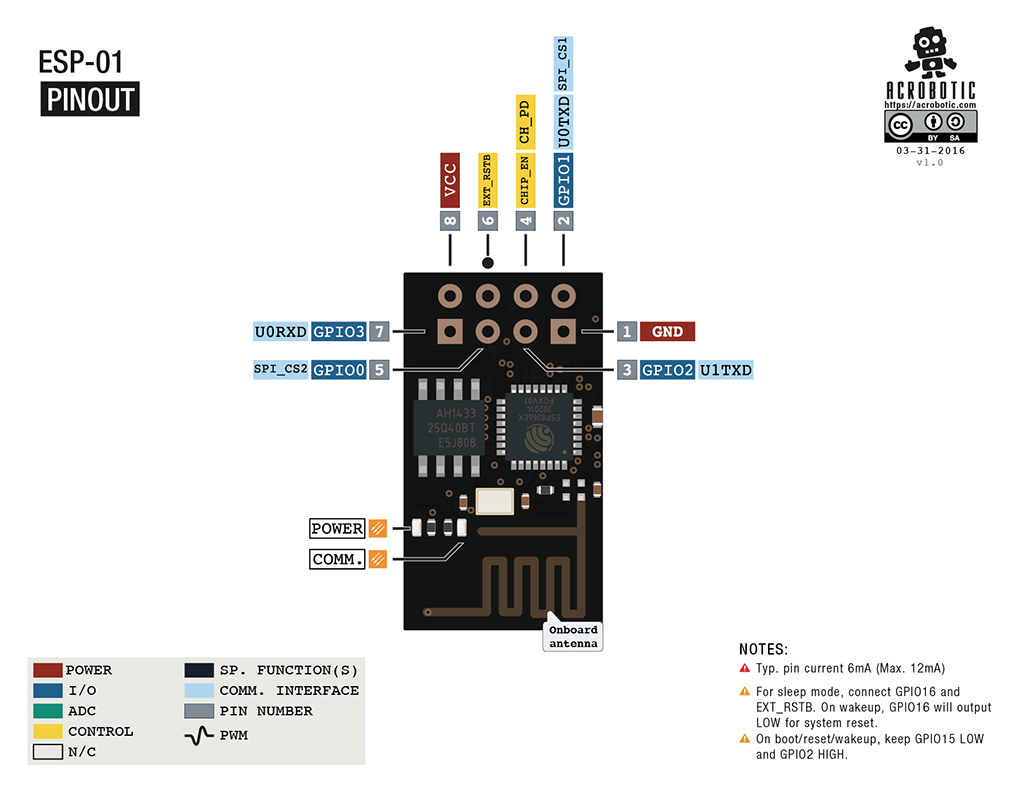
## Comunicaciones Wifi con ESP8266

El ESP8266 es una pequeña placa diseñada por [Espressif](https://www.espressif.com) para proporcionar conectividad Wifi a microcontroladores. Se controla vía serie usando comandos AT (como los Modem, y lo teléfonos GPRS y GSM)



ESP8266-ESP-01.jpg

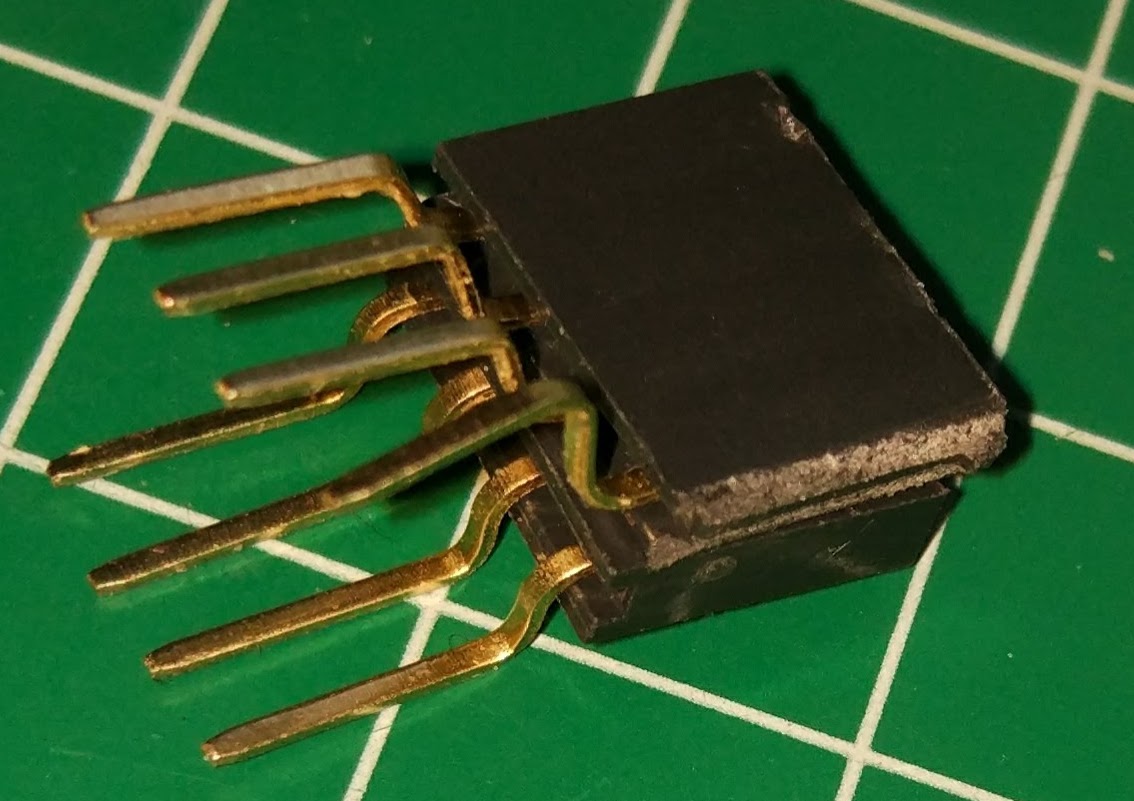
La placa dispone sólo de 8 patillas: \* GND \* Vcc que debe ser de 3.3V \* TX para comunicaciones \* RX para comunicaciones \* EXT\_RSTB reset \* CH\_EN Enabled \* GPIO01 Disponible para programación \* GPIO02 Disponible para programación



esp8266\_esp01\_horizontal-01.png

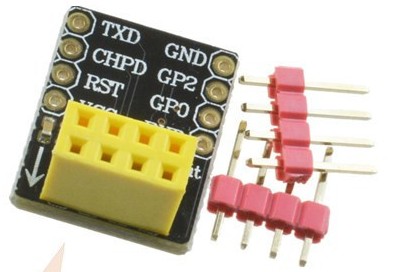
Como vemos la placa funciona a 3.3V pero parece ser tolerante a 5V (al menos no se daña directamente con conectar a las patillas de Arduino)

Como vemos las patillas no están pensadas para usar en protoboard, para conectarlo podemos usar un adaptador como estos que existen o construirlo nosotros mismos con 2 gurpos de 4 pines hembra de patilla larga



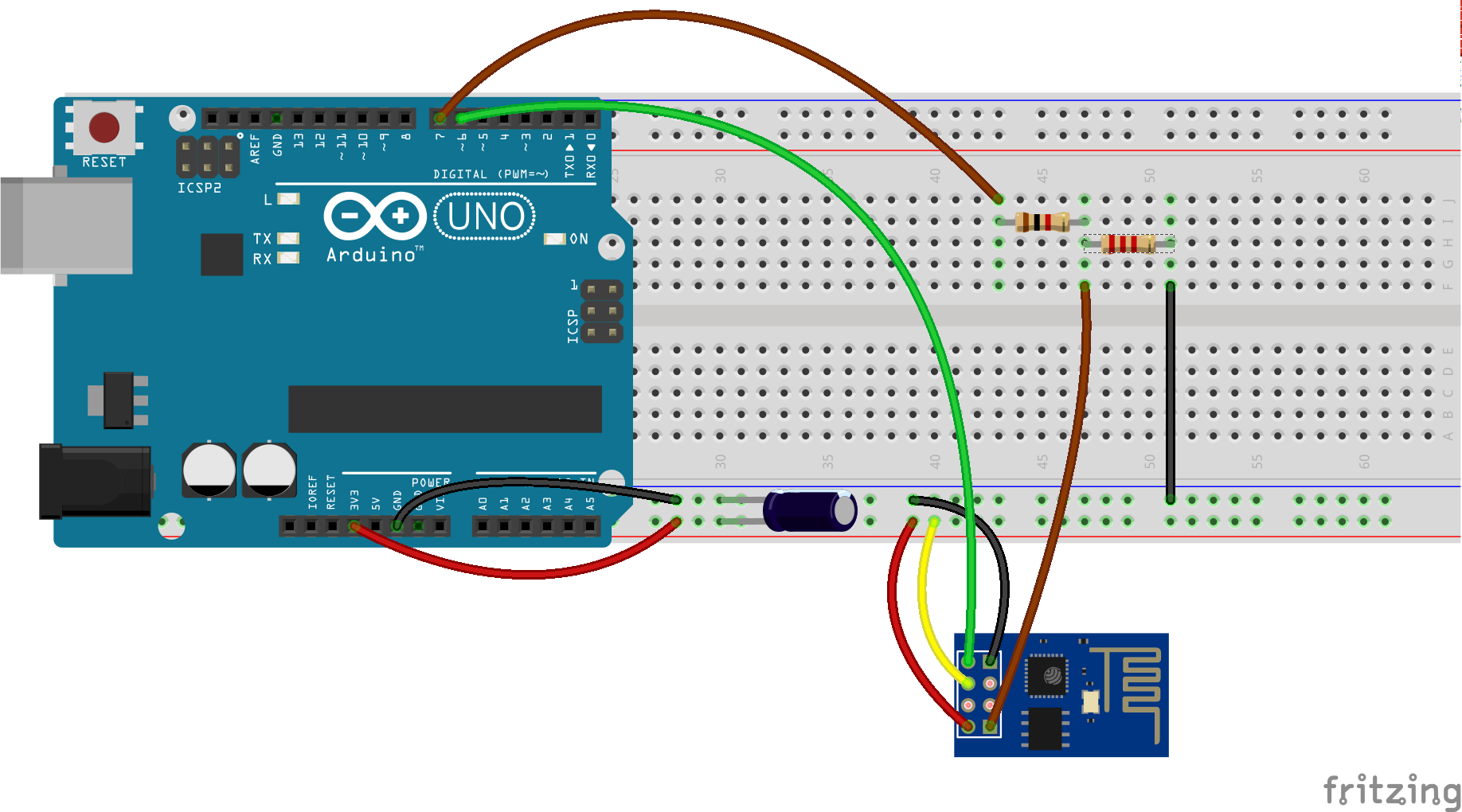
AdaptadorESP-protoboard.jpg

O comprando un adaptador como este otro



HTB1W9adOVXXXXb7XFXXq6xXFXXX8.jpg

Hemos hecho este montaje



Wifi\_ESP8266\_bb.png

Utilizando un divisor de tensión para el pin RX del ESP8266 utilizando una resistencia de 1KΩ y otra de 2.2KΩ de manera que el voltaje que reciva el ESP sea aproximadamente de 3.3V (Si no tenéis resistencias de 2.2KΩ podéis usar 2 de 1KΩ en serie)

Lo hemos alimentado del pin 3.3V de Arduino y funciona bien, pero podríamos tener problemas si usamos el wifi de manera intensiva pues esta salida da poca intensidad. En caso de necesitarlo habría que usar una alimentación externa de 3.3V conectando su GND con el de Arduino.

Para evitar estos problemas de alimentación hemos incluido un condensador de 100microFaradios que contribuye a estabilizar la alimentación.

### Programación

Como hemos dicho tendríamos que usar comandos AT para comunicarnos con el ESP8266 que enviamos por su puerto serie para configurarlo, aquí tenéis el [listado de todos ellos](https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf)

Una forma más sencilla de hacerlos es usando una librería que haga el trabajo dificil por nosotros.

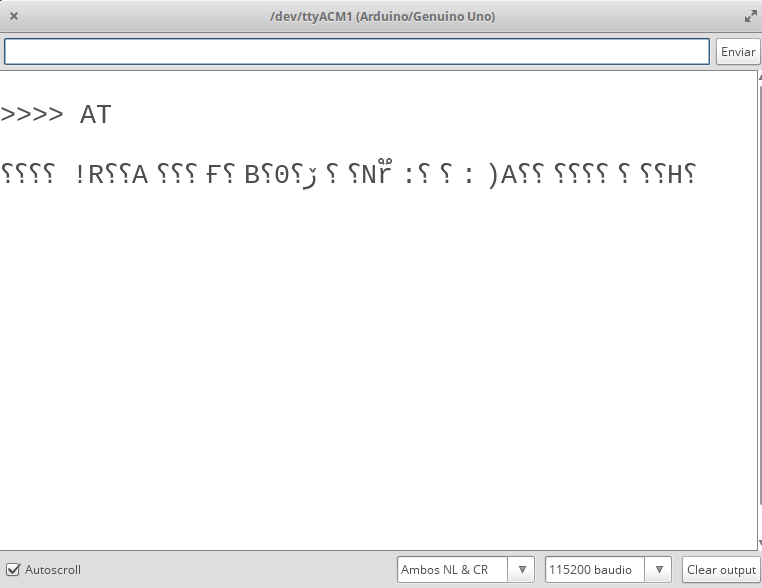
Instalaremos la librería WifiESP desde su repositorio https://github.com/JiriBilek/WiFiEsp descargamos el fichero Zip y lo incorporamos en el entorno desde la opción "Añadir librería desde ZIP" en el menú Programa -> Incluir librería

Una vez importada tendremos sus ejemplos y los usaremos para poner a punto nuestro ESP y la comunicación con Arduino

Abrimos primero "test->ESPDebug" que nos permitirá enviar directamente comandos AT para ver que todo va OK. Lo subimos a nuestro Arduino y abrimos el monitor serie. Escribimos AT

Si recibimos OK de respuesta, ESP8266 está configurado a 9600baudios y podemos pasar a usar los siguientes ejemplos.

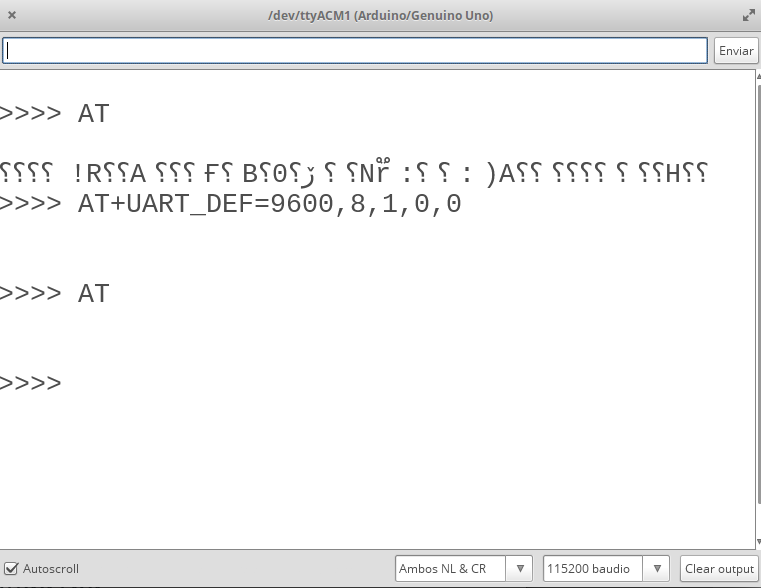
Si recibimos unos caracteres extraños



AT EROR

Para configurarlo a la velocidad correcta enviamos lo siguiente

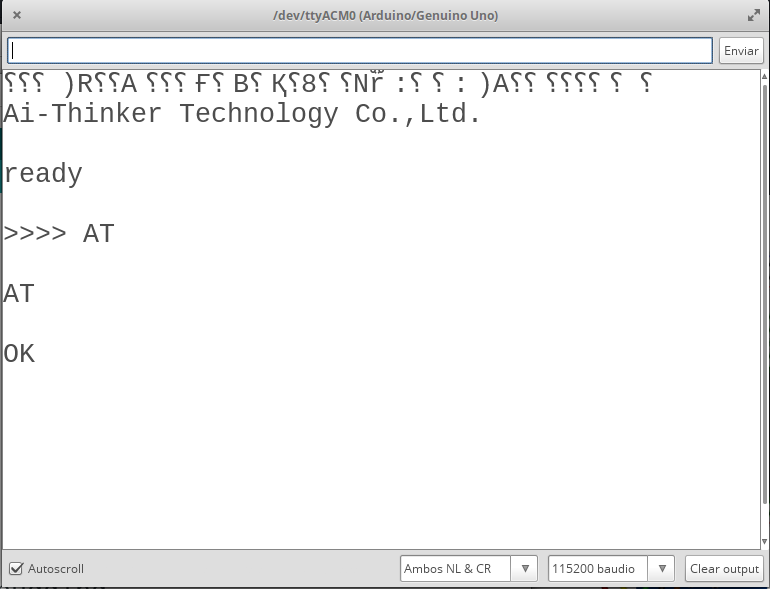
AT+UART\_DEF=9600,8,1,0,0



AT EROR

y reseteamos todo, apagando y encendiendo

Ahora al enviar AT recibimos OK



AT EROR

Ya lo tenemos listo para usarlo.

Empezaremos con el ejemplo WebServer, al que cambiaremos el SSID y el password por los de nuestro wifi. En el monitor serie veremos la ip que tiene y podremos conectarnos con un navegador



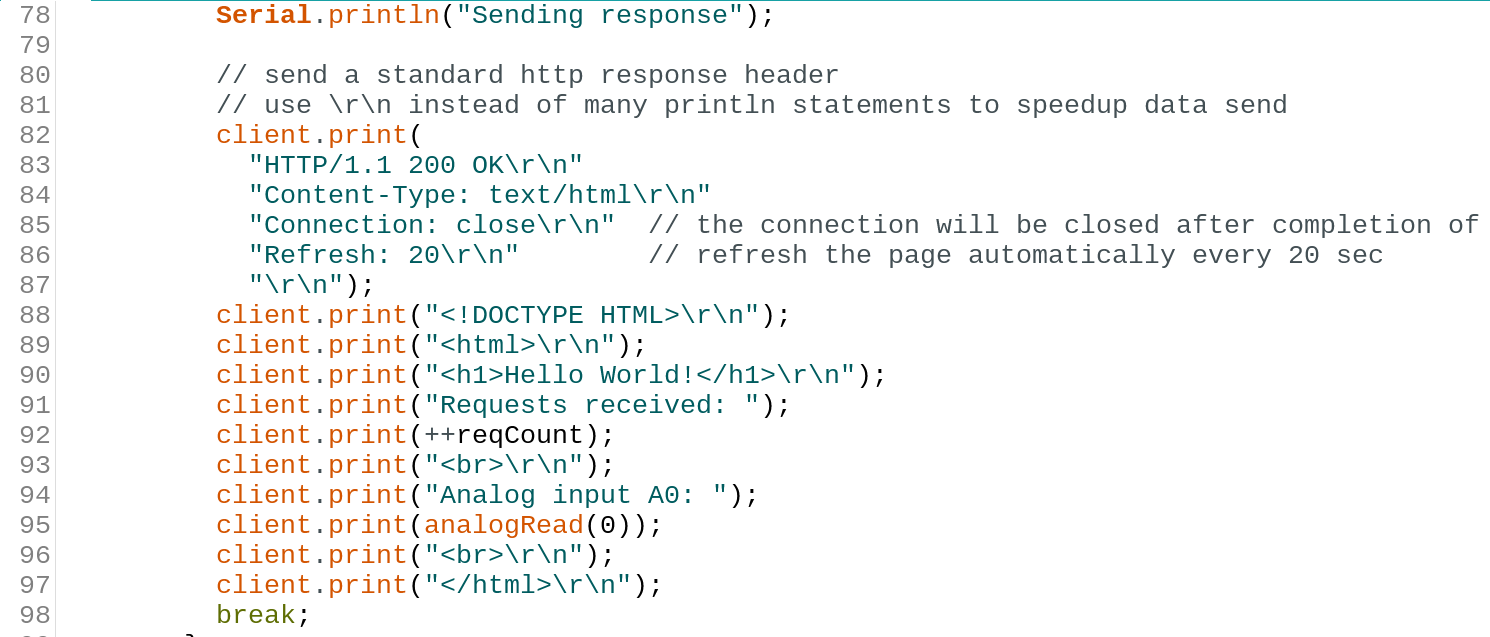
ESP8266-Webserver

Nos conectamos y vemos la web



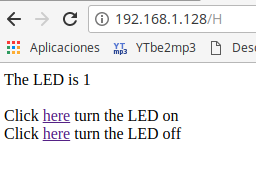
ESP-Webserver

Si analizamos el código vemos que lo que hace es enviar el HTML de la página mediente llamadas client.print("...") y en medio incluye el valor de la lectura analógica



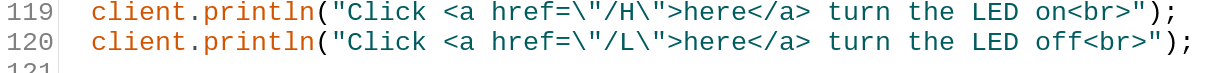
ESPcodigoRespuesta

Vamos a usar ahora el ejemplo WebServerLed que nos permite controlar el estado de un led vía web



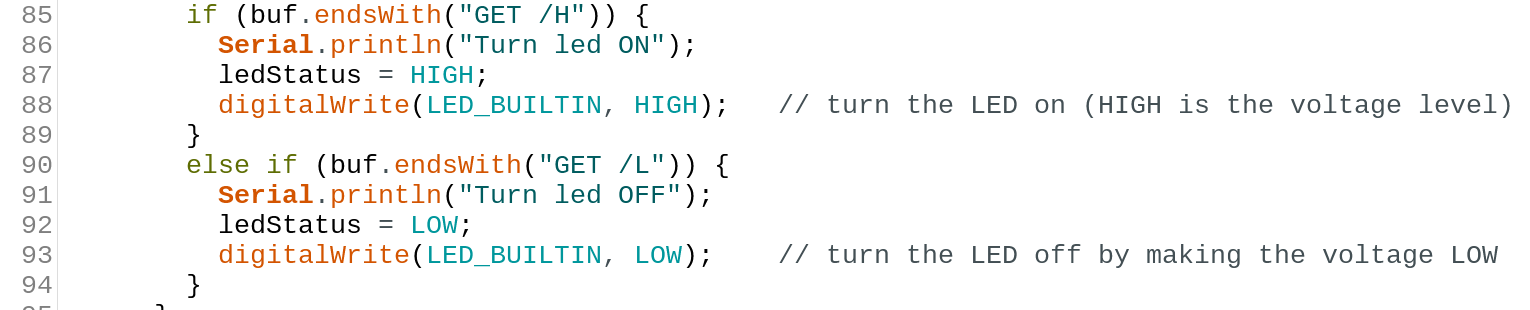
ESP-WEBServerLEd

Donde vemos que según pulsemos un enlace u otro se activa o desacticva el led del pin 13. El "truco" ahora es generar unos enlaces para apagar o encender



ESP-LED-link

Y al recibir una petición se comprueba si viene de H o L



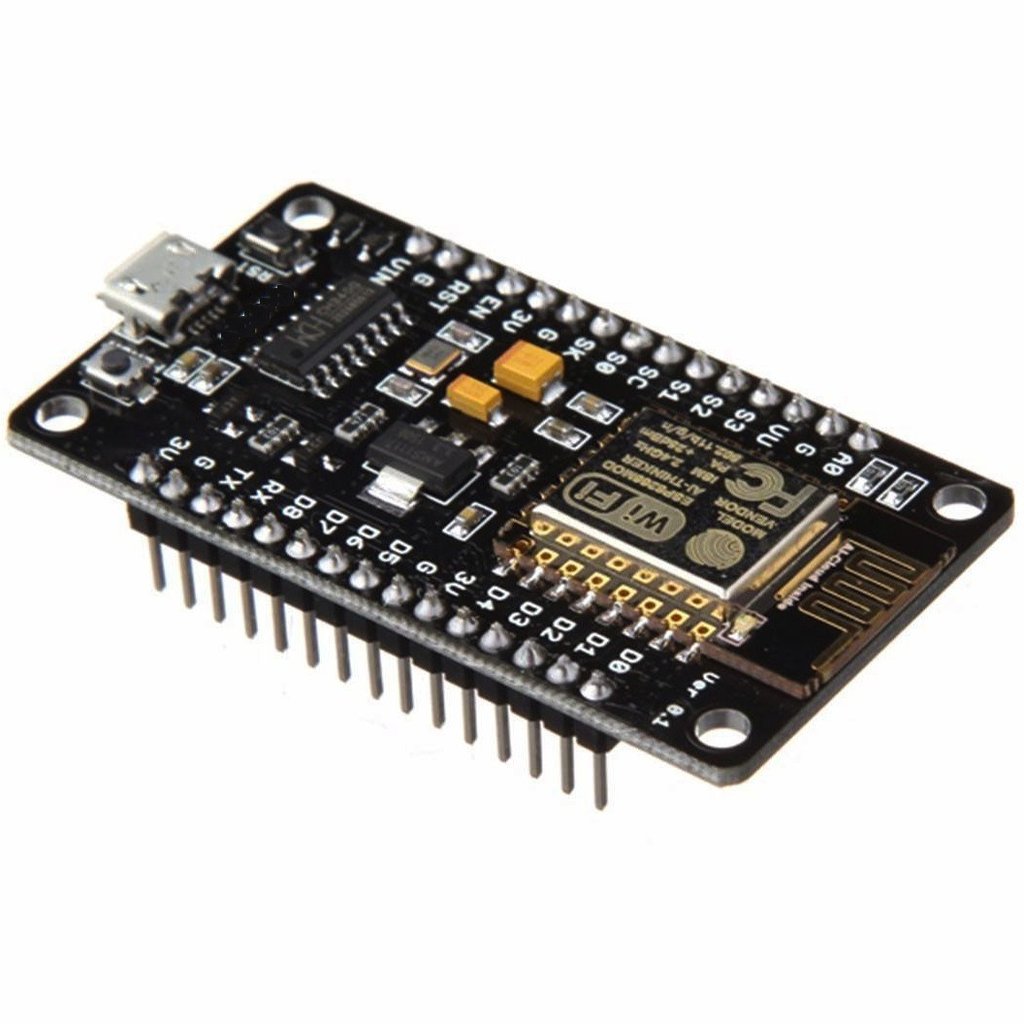
ESP-LED\_Check.png

### ¿Qué más podemos hacer con un ESP8266?

Internamente este pequeño ESP8266 tiene un micropocesador de 32 bits (necesario para implementar WPA) funcionando a 80MHz (y que puede hacerlo a 160MHz) y con 32 Kb de memoria de instrucciones y 80KB de datos de usuario ([Especificiaciones](https://en.wikipedia.org/wiki/ESP8266))

Con estas características es una pena usarlo sólo para comunicaciones....

Si en lugar de usar esta pequeña placa donde la mayoría de los pines no están conectados al exterior lo usamos en un formato mayor como es el ESP12 en una placa NodeMCU tiene sentido eliminar Arduino y hacer todo el trabajo en esta placa



https://cdn.shopify.com/s/files/1/0672/9409/products/NodeMCU\_ESP8266\_development\_board\_1024x1024.jpg?v=1464135546

Pero eso será tema para otro curso...

### Referencias

Os recomiendo esta [estupenda entrada sobre la conexión con ESP8266 de Luis Llamas](https://www.luisllamas.es/arduino-wifi-esp8266-esp01/)

## Comunicaciones por Radio

Algo que cada vez se requiere en más proyectos es la comunicación inalámbrica entre diferentes componentes del mismo. El realizar proyectos distribuidos y el añadir capacidad de procesamiento lleva a utilizar cada vez más sistemas semiíntenligentes

Existen diferentes tecnologías para conseguir esta comunicación por radio. Unas usan tecnología digital, otras analógica, los primeros consiguen mayor calidad, frente al menor precio de los segundos. Algunos dispositivos digitales son capaces de garantizar la entrega de los paquetes y permiten que el microcontrolador no tenga que realizar tareas de este tipo.

Cuando vayamos a utilizarlos en algún proyecto debemos estudiar la capacidad de comunicación que necesita nuestro proyecto, tanto en ancho de banda como la calidad del mismo.

Normalmente tendremos que desarrollar un protocolo propio para enviar nuestra información, protocolo que se basará en el propio protocolo de transporte del dispositivo usado. En [este vídeo](https://www.youtube.com/embed/_uRbiYx_nPQ) vamos a ver algunas de las distintas formas de utilizar comunicaciones radio con Arduino

Veamos ahora cómo usarlos, con librerías y conexiones en [este vídeo](https://www.youtube.com/embed/QKExgg_kUtM)

Existen otras librerías con mayor funcionalidad, capaces de controlar los dispositivos Nordic. Las vemos en [este vídeo](https://www.youtube.com/embed/3lMUoepgeQg)

Si tuviéramos que elegir un sistema de comunicaciones inalámbricas sería sin duda el NRF: es un todo-terreno que podría sustituir a cualquier 485. Además nos evita el cableado que es un infierno en obra ya terminada.

Efectivamente existen 2 librerías muy usadas Mirf y la RF24. La primera creo que es más usada por motivos históricos, pero la RF24 es más avanzada y además se puede usar en la serie ATTiny lo que te permite montar un nodo de la red en un espacio completamente enano.

Sobre el tema de la interferencias, cobertura,  y alcance no hay que preocuparse, porque si bien los modelos más baratos tienen un alcance algo limitado en interiores existen otros equipos con conector SMA a los que les puedes poner antenas de alta ganancia alcanzando hasta Km sin problema. Otro ventaja en este tema es que puedes regular la potencia de transmisión con lo que puedes ahorrar energía.

Sobre las interferencias es algo que te evitas al usar un protocolo con ACK  (confirmación de llegada) y reintento automático. Si no necesitas enviar muchos datos por segundo sólo tendrías que ampliar el timeout para garantizar que siempre se entrega. Un sencillo contador de paquetes perdidos puede permitirte llevar una estadísticas y en caso de detectar pérdidas, ampliar la potencia de envío.

## Uso del GPS

Un GPS no es más que un dispositivo que recibe señales de los satélites GPS, las procesa para obtener información sobre la posición y las envía (tramas NMEA) por un puerto de comunicaciones, normalmente vía puerto serie.

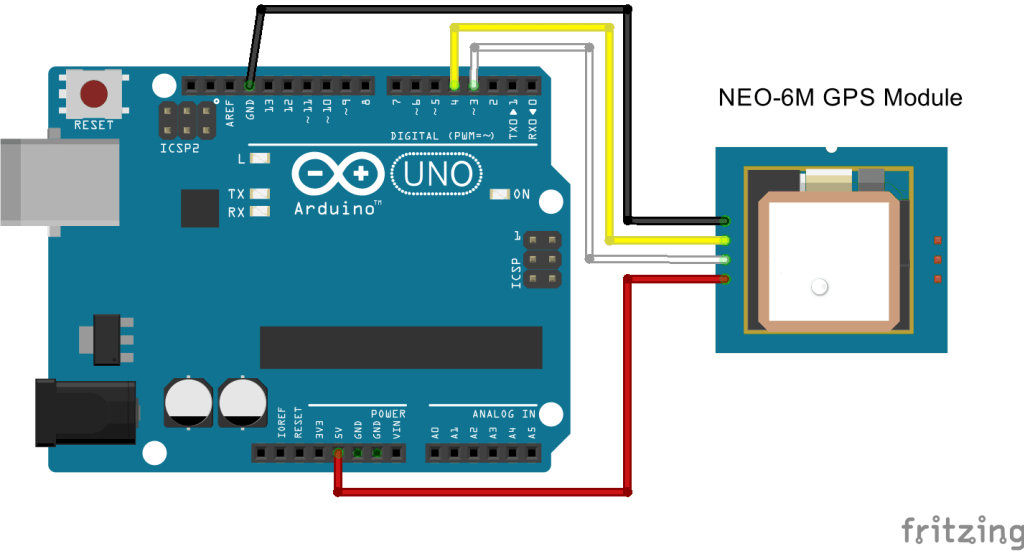
En mi caso voy a usar un módulo [ATGM336H GPS BDS GNSS](http://www.icstation.com/atgm336h-gnss-dual-mode-positioning-module-eeprom-25ma-vehicle-location-p-12420.html) de ICStation



http://www.icstation.com/images/big/products/12420\_0152.JPG

Su consumo es muy bajo (25mA) lo que lo hace ideal para sistemas desatendidos.

Su conexión es muy sencilla pues basta con alimentarlo y conectar sus pines Tx y Rx a Arduino y hacer un programa que lea las tramas.



https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2018/01/NEO\_6M\_Arduino.png?resize=1024%2C551&ssl=1

En el programa usaremos la librería SoftwareSerie para poder usar otros pines de Arduino, en concreto vamos a usar los pines 3 y 4

/\*  
 \* Basado en el codigo de  
 \* Rui Santos  
 \* Complete Project Details http://randomnerdtutorials.com  
 \*  
 \* modificado por JAVacasM  
 \*/  
  
#include <SoftwareSerial.h>  
  
// leeremos la senial PPS (Pulso Por Segundo) con el pin 12  
// Encenderemos el led 13 con esa frecuencia  
  
#define PIN\_PPS 12  
#define PIN\_PULSO 13  
  
// The serial connection to the GPS module  
SoftwareSerial ss(4, 3);  
  
void setup(){  
 pinMode(PIN\_PPS,INPUT);  
 pinMode(PIN\_PULSO,OUTPUT);  
  
 Serial.begin(9600);  
 ss.begin(9600);  
}  
  
void loop(){  
 while (ss.available() > 0){  
 // get the byte data from the GPS  
 byte gpsData = ss.read();  
 Serial.write(gpsData);  
  
 if(digitalRead(PIN\_PPS) == LOW) // ha cambiado el pulso  
 digitalWrite(PIN\_PULSO,!digitalRead(PIN\_PULSO)); // Invertimos el pin 13  
 }  
  
}

Hay que recordar que los receptores GPS (y sobre todo los baratos) tardan un tiempo en conseguir suficientes señales de los satélites y necesitan de un espacio abierto para que estas lleguen.

Cuando empiecen a recibirse tramas serán algo así:

$GPGGA,140817.00,4105.2344,N,00831.54761,W,1,05,2.68,129.0,M,50.1,M,,\*42  
 $GPGSA,A,3,06,09,30,07,23,,,,,,,,4.43,2.68,3.53\*02  
 $GPGSV,3,1,11,02,48,298,24,03,05,101,24,05,17,292,20,06,71,227,30\*7C

Que nos informa de la posición, altura, etc... Por ejemplo la primera línea nos dice

* Hora de la señal 14:08:17
* Latitud 4105.2344,N
* Longitud 831.54761,W

Podemos procesar estos datos usando librerías por ejemplo [TinyGPSPlus](https://github.com/mikalhart/TinyGPSPlus) que después de descargarla e instalarla podemos usar en el siguiente código

/\*  
 \* Basado en el codigo de  
 \* Rui Santos  
 \* Complete Project Details http://randomnerdtutorials.com  
 \*  
 \* modificado por JAVacasM  
 \*/  
  
#include <TinyGPS++.h>  
#include <SoftwareSerial.h>  
  
  
// The TinyGPS++ object  
TinyGPSPlus gps;  
  
// The serial connection to the GPS device  
SoftwareSerial ss(4, 3);  
  
void setup(){  
 Serial.begin(9600);  
 ss.begin(9600);  
}  
  
void loop(){  
  
 while (ss.available() > 0){ // si hay datos serie los procesamos  
 gps.encode(ss.read());   
 if (gps.location.isUpdated()){ // Esta actualizado  
 Serial.print("Latitude= ");  
 Serial.print(gps.location.lat(), 6);  
 Serial.print(" Longitude= ");  
 Serial.println(gps.location.lng(), 6);  
 } else { // Mostramos los satelites y la fecha  
 Serial.print("Number os satellites in use = ");  
 Serial.println(gps.satellites.value());  
  
 Serial.print("Raw date DDMMYY = ");  
 Serial.println(gps.date.value());  
  
 }  
 }  
}

### Referencias

[Página del producto ATGM336H GPS BDS GNSS](http://www.icstation.com/atgm336h-gnss-dual-mode-positioning-module-eeprom-25ma-vehicle-location-p-12420.html)

Para aprender más sobre GPS os recomiendo [el artículo de Rui Santos](https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/) del que saqué la información

Más detalles sobre el módulo GPS en el su [datasheet](http://www.icofchina.com/d/file/xiazai/2016-12-05/b5c57074f4b1fcc62ba8c7868548d18a.pdf)

## Uso de RFIDs

El sistema RFID está pensado para permitir la lectura de etiquetas a distancia sin establecer contato físico entre el lector y la etiqueta.

Las hay de corta distancia como las NFC, de media distancia como las de los sistemas antirrobo de los comercios (los aparatos que hay en la salidas son enormes antenas) y de larga distancia, como las que se usan en los parkings de los aeropuertos para reconocer los vehículos autorizados o en los autobuses para detectar que están en una parada.

El sistema está formador por un lector/emisor de radiofrecuencia que incluye una antena (que puede ser de mayor o menor tamaño)



Lector RFIDs

y unas etiquetas (Tags) que son de diferentes formatos, según la frecuencia a la que funcionan y de la cantidad de información que permiten almacenar. Vemos que todas las etiquetas tienen un circuito que actua como antena receptora, con una frecuencia determinada para la que resuena



RFID Tag

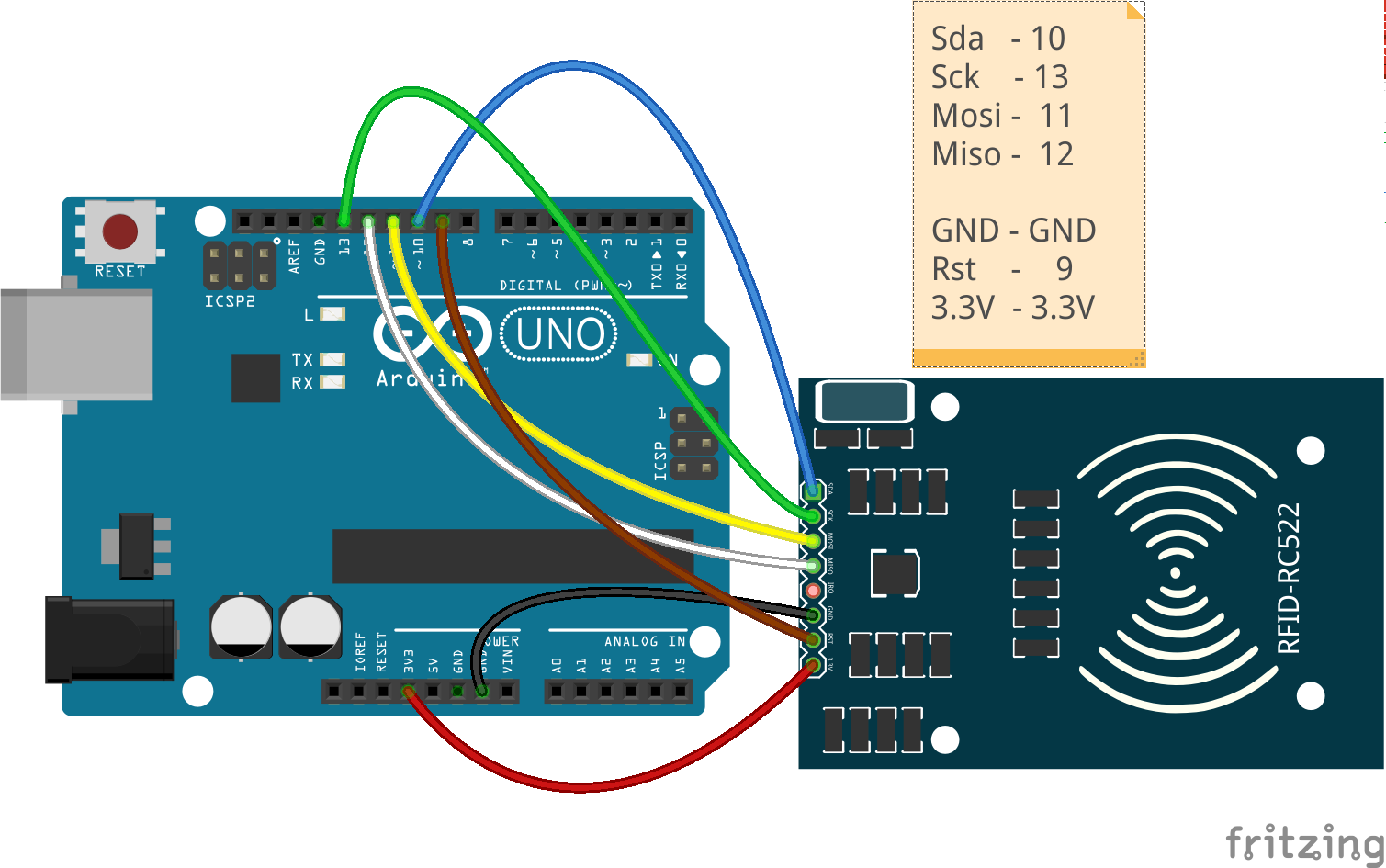
El emisor genera un señal de radiofrecuencia que la etiqueta recibe generando un eco que es leído por el lector.

Algunas tarjetas permiten la escritura, pudiendo guardar información en ellas (alrededor de 1KB)

### Montaje

El montaje es sencillo y sólo necesita el lector de RFIDs, un arduino y 7 cables macho-hembra. Para conectarlos seguimos el típico esquema de conexión de un dispositivo SPI:

|  |  |
| --- | --- |
| Lector RFID | arduino |
| Sda | 10 |
| Sck | 13 |
| Mosi | 11 |
| Miso | 12 |
|  |  |
| GND | GND |
| Rst | 9 |
| 3.3V | 3.3V |

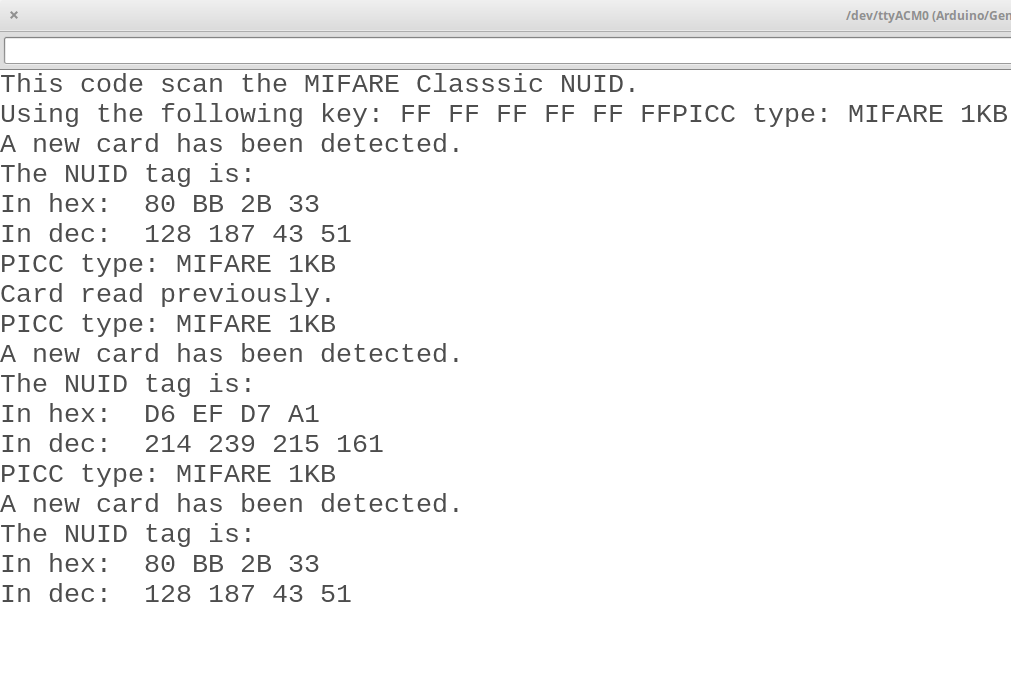


### Programación

Para este tipo de lectores usaremos la librería MFRC522 que podemos instalar desde el gestor de librerías de Arduino.

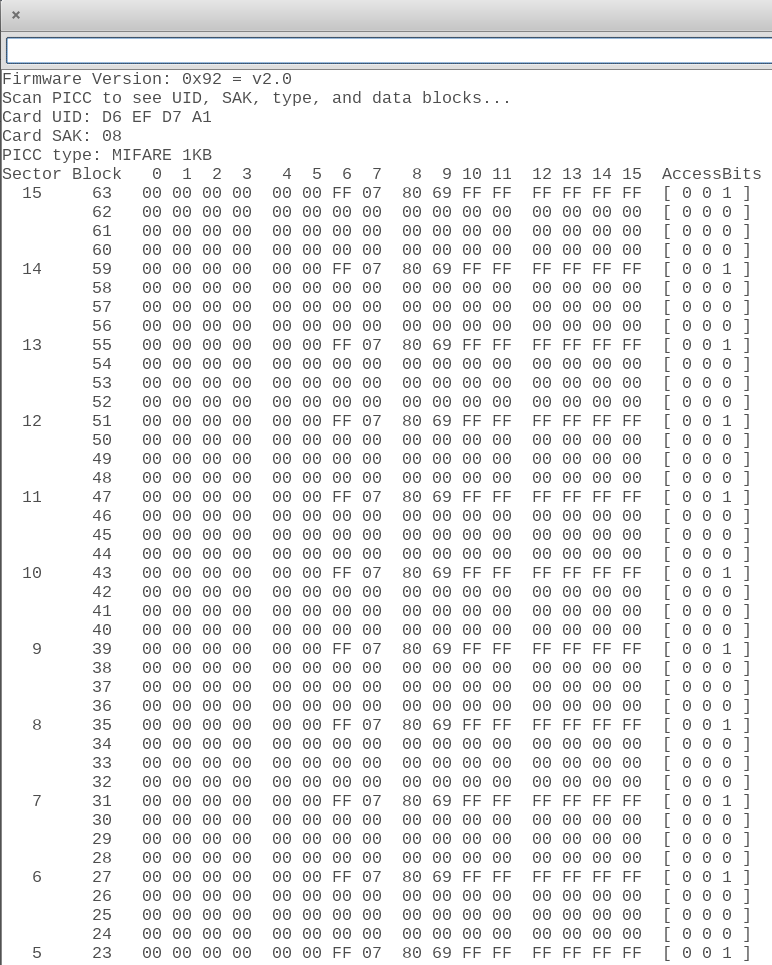
Una vez instalada tendremos ejemplos para hacer todas las funciónes disponibles:

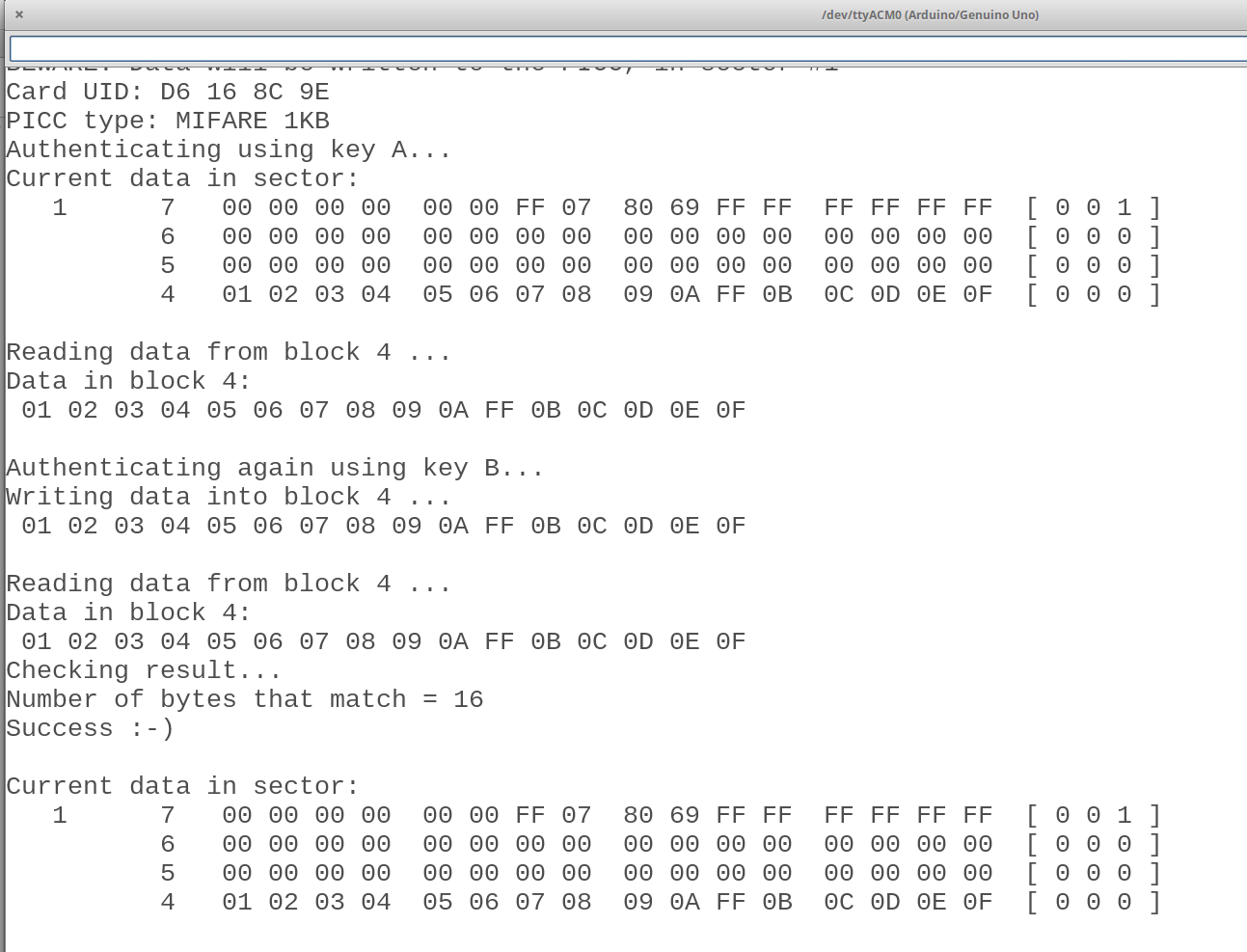
* Ejemplo ReadNUID: lee el ID de la tarjeta y lo muestra



RFID\_Lectura.png

* Ejemplo DumpInfo: lee toda la información de la tarjeta y la envía al monitor serie

 \* Ejemplo ReadAndWrite: lee la información y cambia el bloque 4 escribiendo 0, 1,2, ... 15 en él



RFID\_Escritura.png

En [este vídeo](https://youtu.be/suunQECfMM8) hablamos de RFIDs.

### Referencias

[Tutorial de Luis Llamas](https://www.luisllamas.es/arduino-rfid-mifare-rc522/)

El ejemplo de "Valida Tarjeta" no es correcto, falta copiar lo leído y comparar con ese valor leído