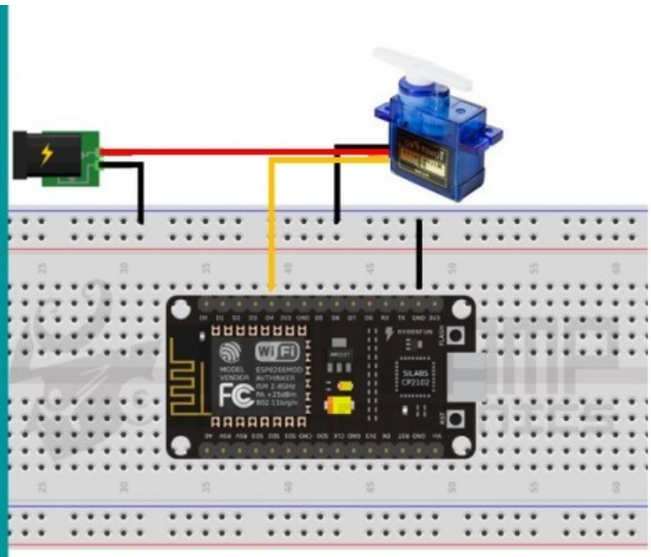
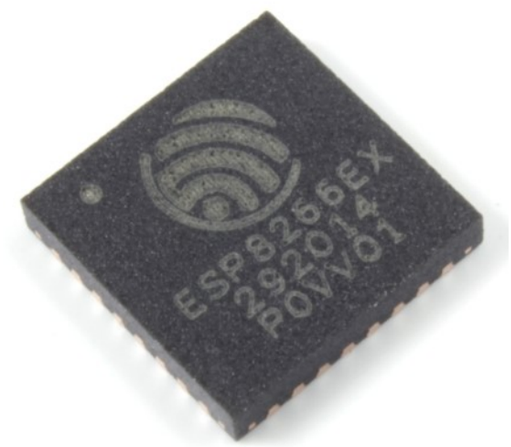
Usando ESP8266 con el IDE de Arduino

(Versión 25-5-19)



En este tutorial explicaremos como instalar el plugin en el IDE de Arduino para poder programar el ESP8266 (sin usar la placa Arduino), y desarrollaremos algunos ejemplos básicos con el fin de comparar las funcionalidades del ESP8266 y Arduino.

El ESP8266 es un SoC (System on Chip), con capacidades de 2.4 GHz Wi-Fi (802.11 b / g / n, soporte WPA / WPA2), 16 GPIO de propósito general (entrada / salida), I²C, convertidor analógico-digital (ADC de 10 bits), SPI, I²S, UART y modulación de ancho de pulso (PWM), emplea un CPU RISC de 32 bits basado en el Tensilica Xtensa LX106 funcionando a 80 MHz (o overclocked a 160 MHz). Tiene una memoria ROM de inicio de 64 KB, memoria RAM de instrucciones de 64 KB y 96 KB de RAM de datos. Memoria flash externa de 4MB pero este último varía entre diferentes versiones de modulo.



En pocas palabras podemos decir que nuestro ESP8266 es un microcontrolador con WIFI incorporado. Y al igual que muchos microcontrolador podemos reprogramar a nuestra necesidad, por defecto o de fábrica nuestro ESP8266 viene con un firmware de comandos AT, que nos permite manipular el ESP8266 desde otro microcontrolador, nosotros en este tutorial vamos a programar el ESP8266 pero usando el IDE Arduino.

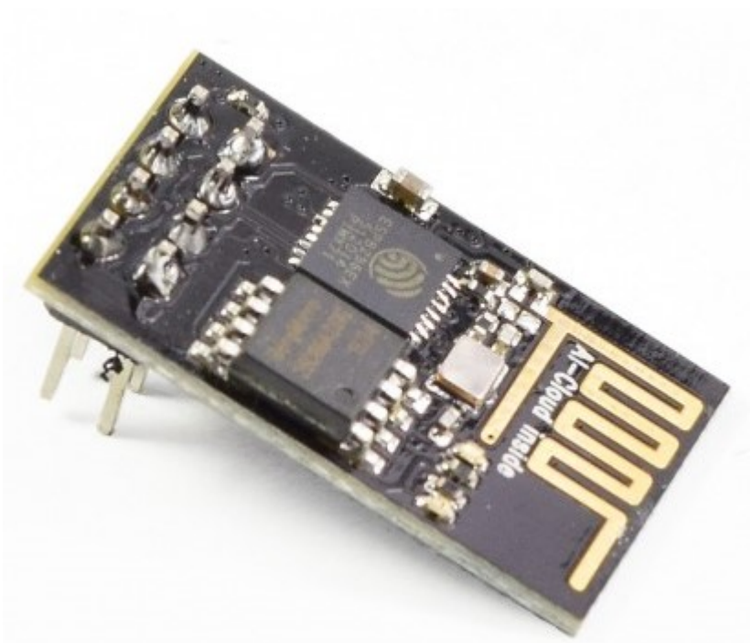
El chip ESP8266 comercialmente se lo encuentra en diferentes módulos cuya diferencia son la cantidad de pines disponibles y en algunos casos el tamaño de memora flash. En el siguiente enlace pueden revisar los diferentes tipos de módulos:

https://www.esp8266.com/wiki/doku.php?id=esp8266-module-family

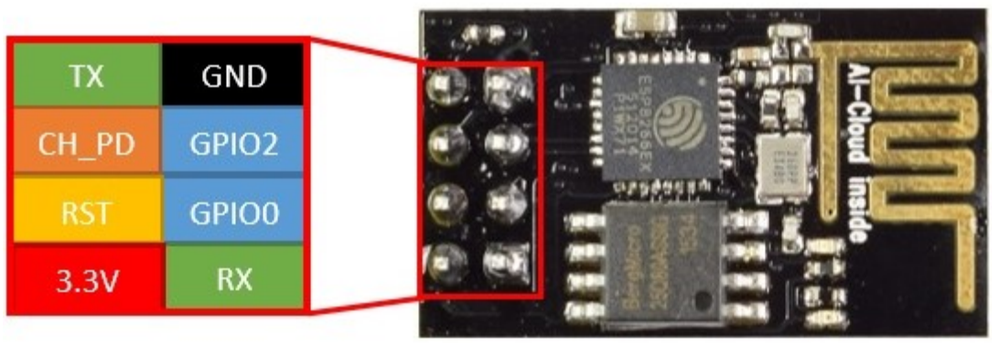
Nosotros detallaremos dos de los más comerciales

ESP-01

Este módulo es comúnmente llamado Modulo wifi y está diseñado para usar como un transiver Wifi, para ser controlado de forma serial con un microcontrolador usando comandos AT.

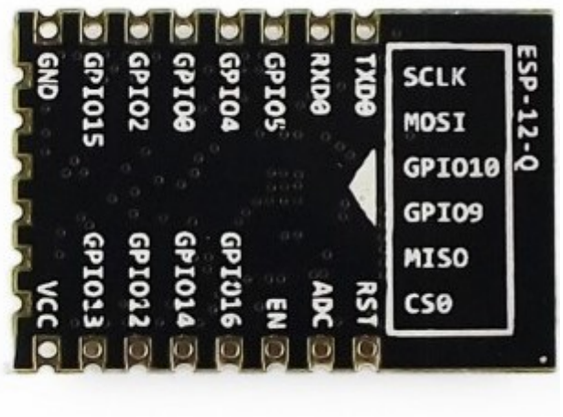


Solo posee los pines de comunicación, de alimentación y dos GPIO.

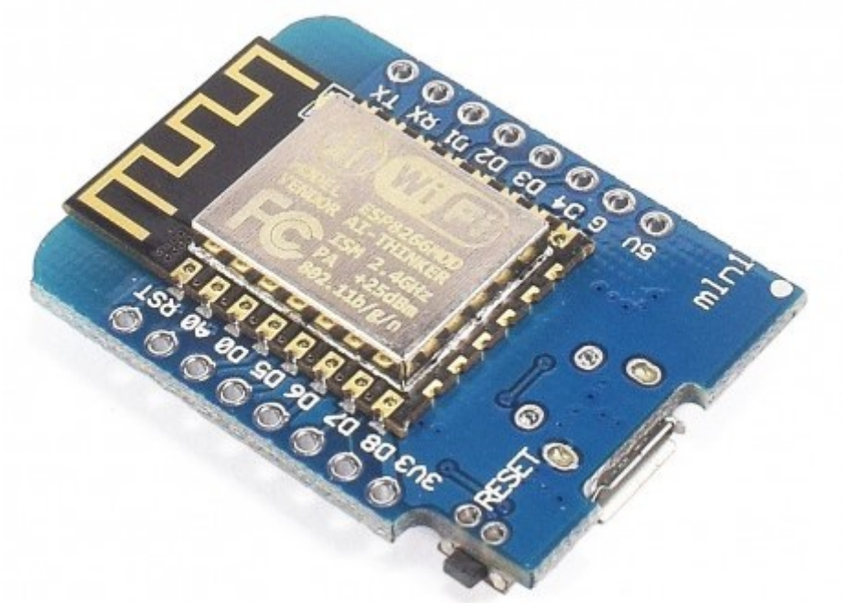


ESP-12E

Este módulo es uno de los más usados, tiene disponible todos los pines que el chip ESP8266 posee.



Este chip es la base de varias plataformas de Iot, siendo las más populares NodeMCU y Wemos



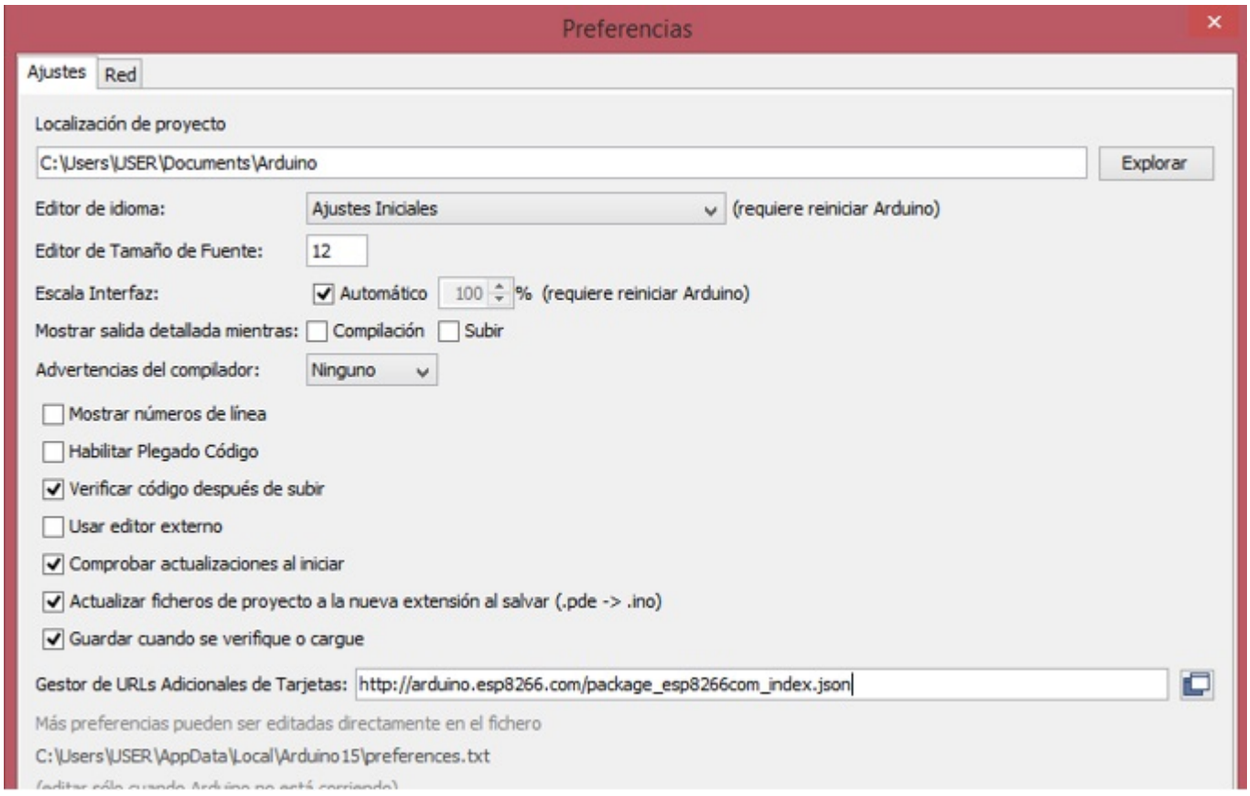
Instalando Plugin del ESP8266 para Arduino.

Este paso es necesario para que nuestro IDE de Arduino reconozca a nuestro ESP8266 como una tarjeta.

Debemos tener ya instalado nuestro Arduino IDE con versión 1.6.4 o superior.

Seguidamente vamos a archivo>Preferencias y en la casilla “Gestor de URLs Adicionales de Tarjetas” agregamos:

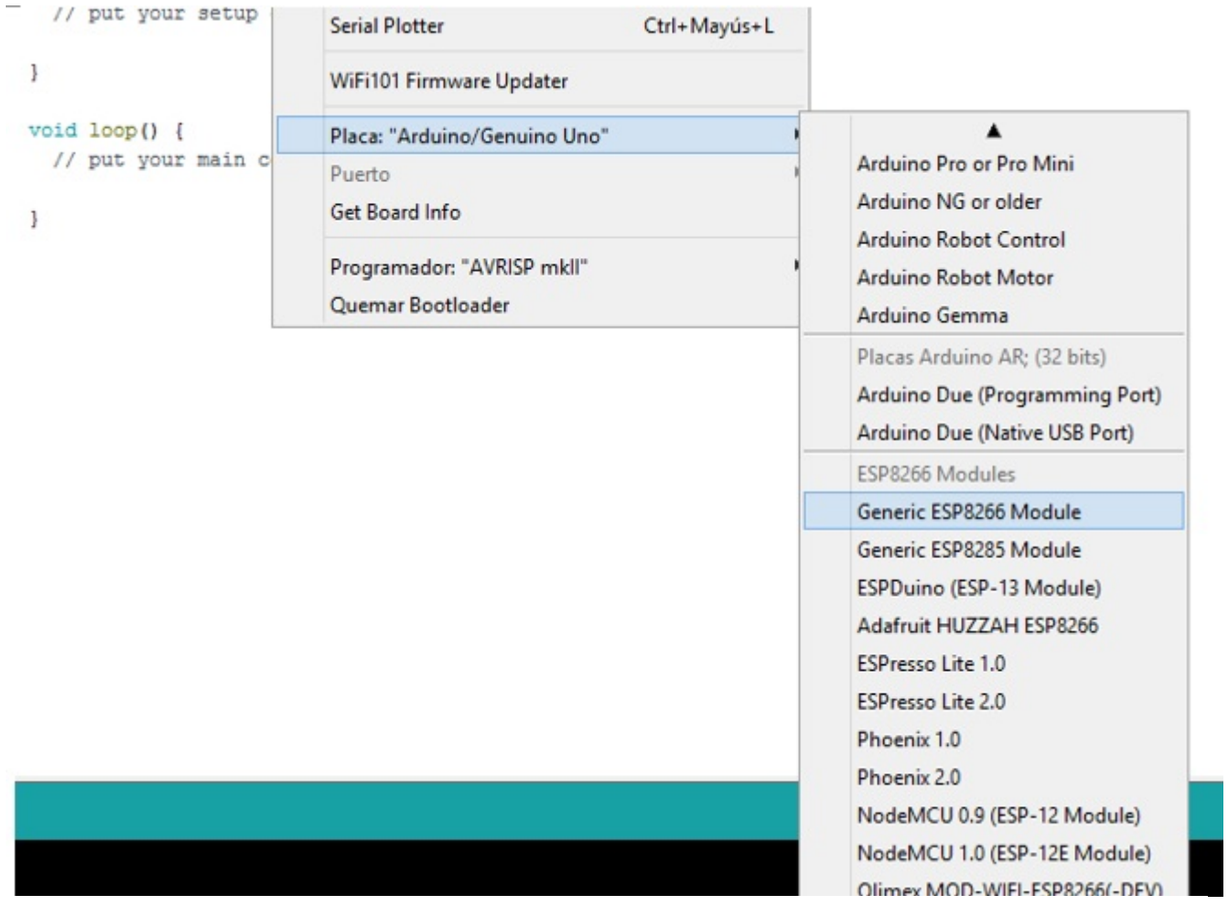
http://arduino.esp8266.com/stable/package_esp8266com_index.json



Seguidamente vamos a Herramientas>placa: … >Gestor de Tarjetas
Y buscamos en la lista “esp8266 by ESP8266 Community“, lo seleccionamos e instalamos

La instalación va a demorar un poco, al finalizar, el ítem del ESP8266 les debe marcar como instalado.

Ahora en herramientas>placas, deben de estar las nuevas placas instaladas.



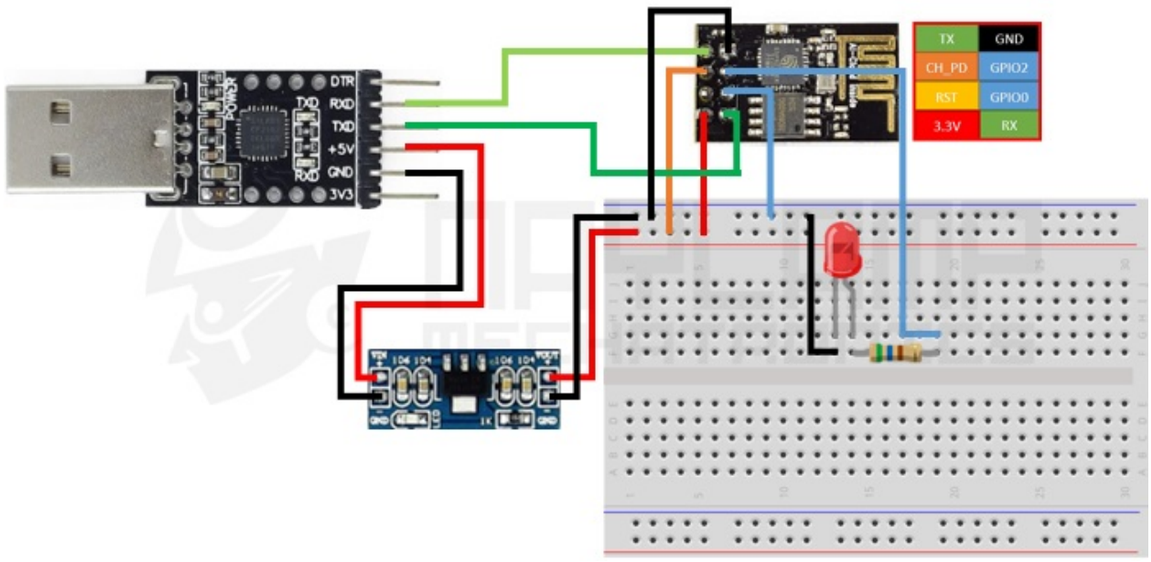
Conexión ESP8266

Esto dependerá del modelo ESP que tengan:

ESP-01:

Este modelo tal vez sea el más molestoso para conectar: para que el modulo funcione es necesario conectar los pines VCC (3.3V), GND y CH_PD(a 3.3V) con esto empieza a funcionar. Adicionalmente como necesitamos la conexión con la PC vamos conectar el RX y el TX a un conversor USB serial, con esto ya está la comunicación con la PC, pero solo para programar se necesita que el ESP8266 arranque con el pin GPIO0 en estado LOW, de esta forma el ESP entra a modo de programación.

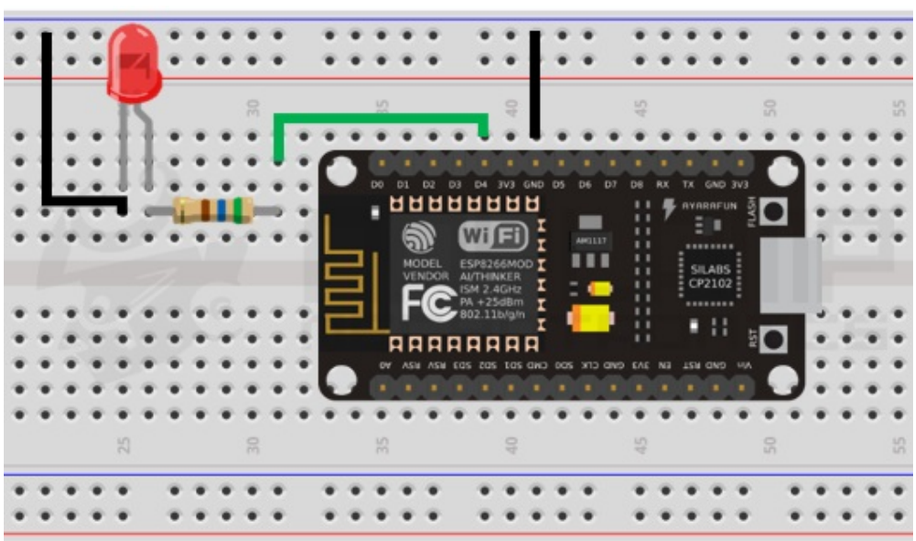
Después de programar puedes desconectar el GPIO0 para que no entre nuevamente en estado de programación cada vez que se enciende el módulo.



Notar que usamos un regulador de 3.3V, esto porque dependiendo del conversor USB serial que usen los 3.3V no podría tener la suficiente corriente para alimentar el ESP y esto podría causar que se cuelgue o reinicie el modulo.

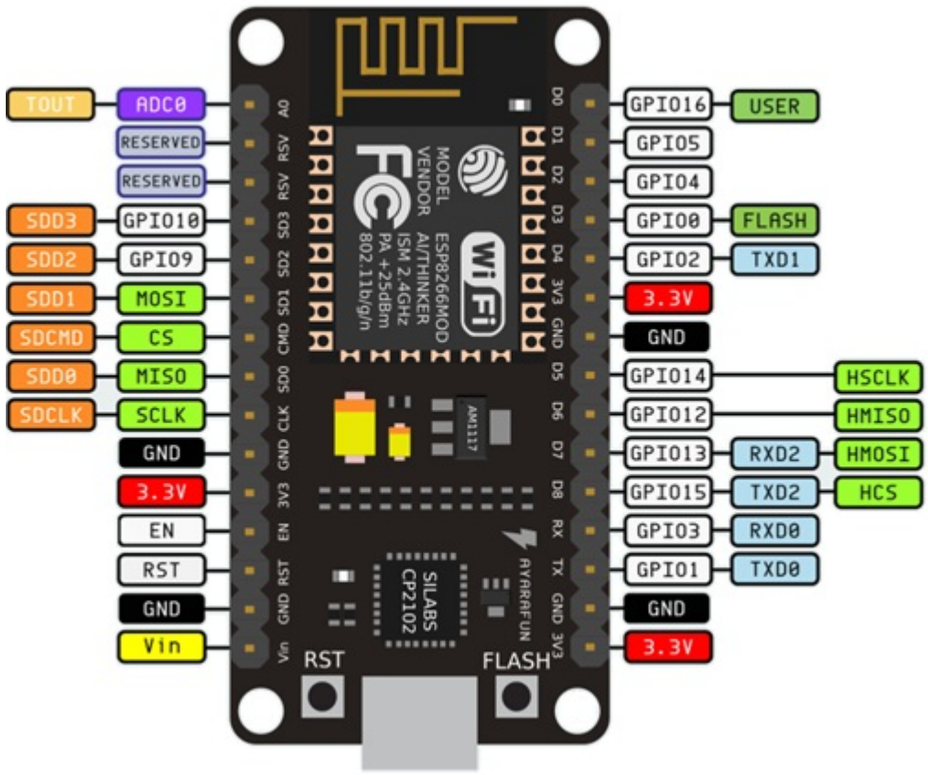
NodeMCU (ESP-12E)

En este caso solo necesitan conectar su cable USB y conectar un led en el pin D4, esto para el primer ejemplo que veremos más adelante.



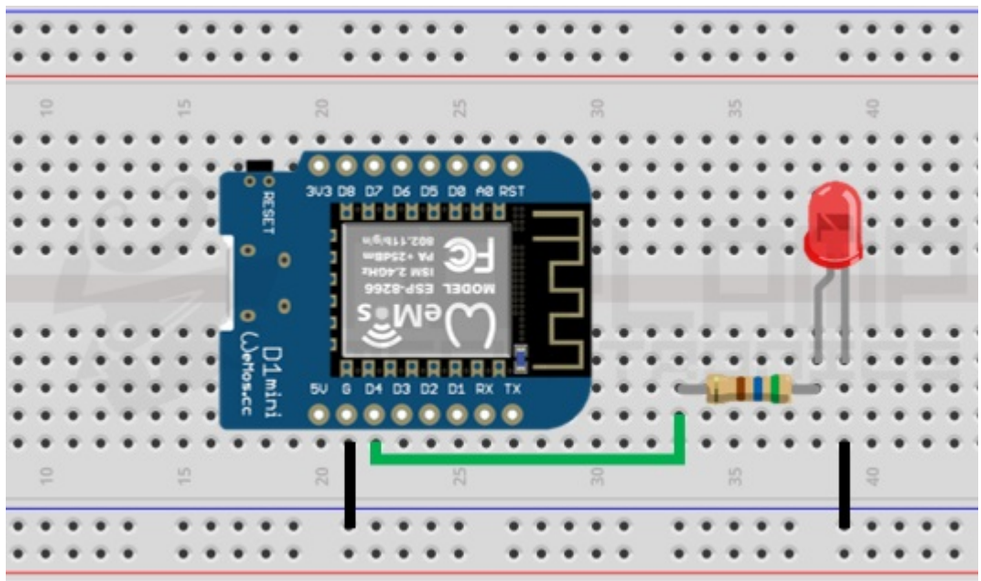
El pin D4 es el pin GPIO2 del ESP-12E, desde el IDE de arduino nos guiaremos por los GPIO en lugar de la numeración de pines de la palca:

A continuación pueden ver la distribución de pines GPIO en el NodeMCU:

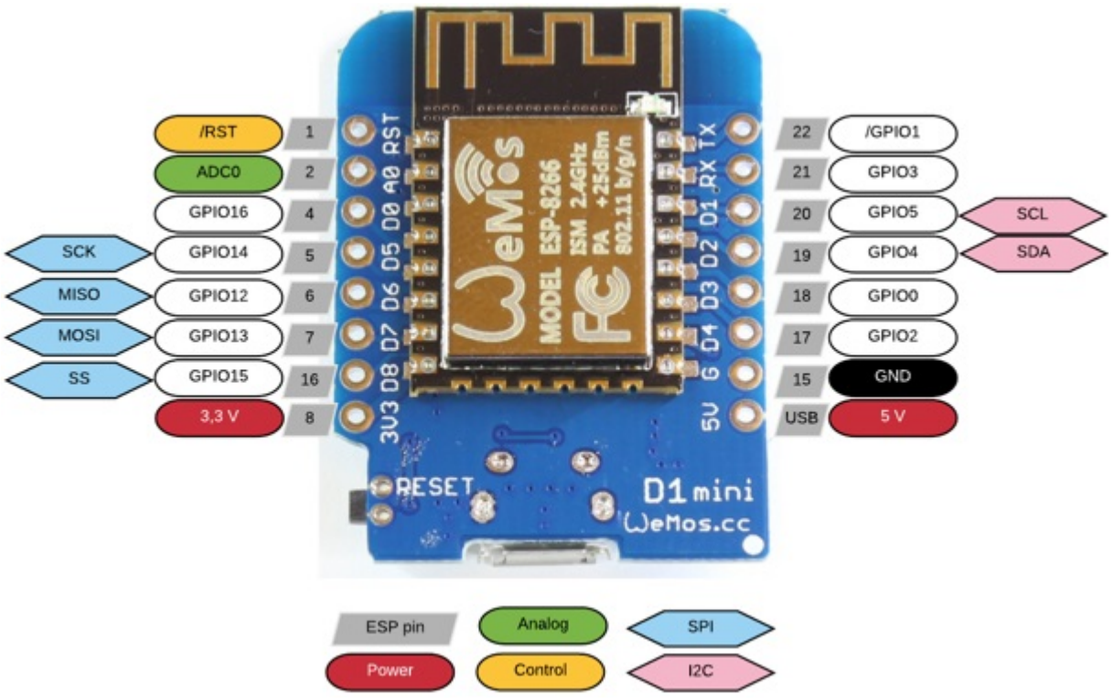


WeMos (ESP-12E)

El Wemos que usaremos será el Wimos D1 Mini y al igual con el MCU solo necesitan conectar el Led al pin D4 (GPIO2), esta conexión será para el primer ejercicio que desarrollaremos.



La distribución de pines GPIO en el WeMos Mini D1 es de la siguiente forma:



Ej.1 Blinking LED con ESP8266

En este ejemplo haremos el hola mundo de microcontroladores que es hacer parpadear un led.

EL código e el siguiente:

```
void setup() {
  pinMode(2, OUTPUT);
}

void loop() {
  digitalWrite(2, HIGH);    // Led ON
  delay(1000);                    // pausa 1 seg
  digitalWrite(2, LOW);     // Led OFF
  delay(1000);                    // pausa 1seg
}
```

Como se observa el código es igual como si se estuviera programando un arduino.

En el caso del ESP01 (módulo wifi) es necesario que al momento de cargar el ESP esté en modo de programación (debe haber iniciado con el GPIO2 en Low) y en el caso del NodeMCU o el WeMos tan solo hay que cargar desde el IDE como si fuera una arduino.

En cualquiera de los casos hay que verificar que en el IDE se haya seleccionado la placa y el puerto correspondiente (En el caso del ESP-01 debe estar seleccionado la placa: Generic ESP8266 Module)

A diferencia de un arduino durante el proceso de carga, en la parte inferior del IDE nos muestra el porcentaje de la carga que se está realizando:

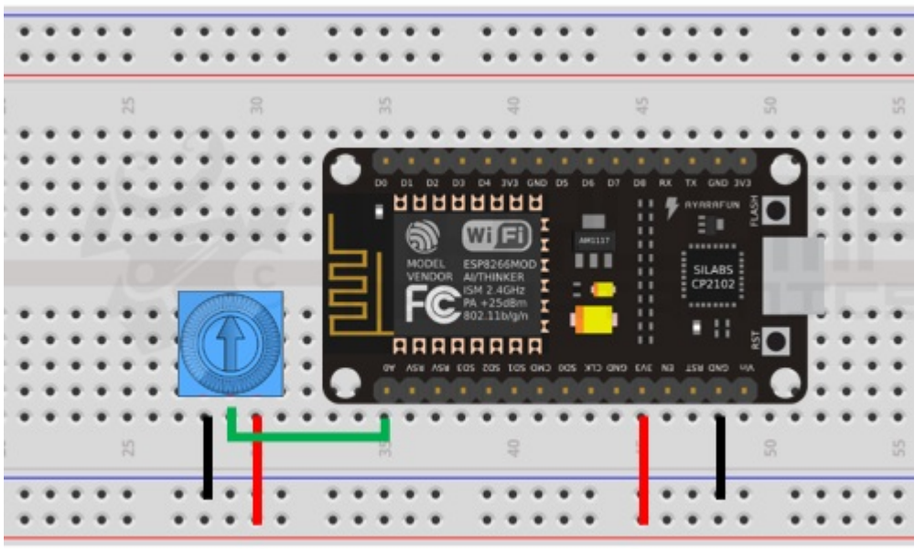
Si han realizado bien todos los pasos anteriores, debe parpadear el LED, en el caso del NodeMCU y WeMos notaran que también parpadea el Led del ESP-12E, esto porque el GPIO2 generalmente se usa como TX1 para hacer debugging y el Led muestra el estado del pin GPIO2 pero negado.

Si su led Parpadea entonces significa que tienen todo configurado correctamente y están listos para realizar los siguientes ejemplos:

Ej.2 Lectura analógica con ESP8226

EL ESP8266 tiene una sola entrada analógica y al igual que un arduino es de 10 bits pero trabaja con un voltaje de referencia interno de 1V. En el NodeMCU y WeMos en la entrada A0 tiene un divisor de voltaje, para adaptar el rango hasta los 3.3V.

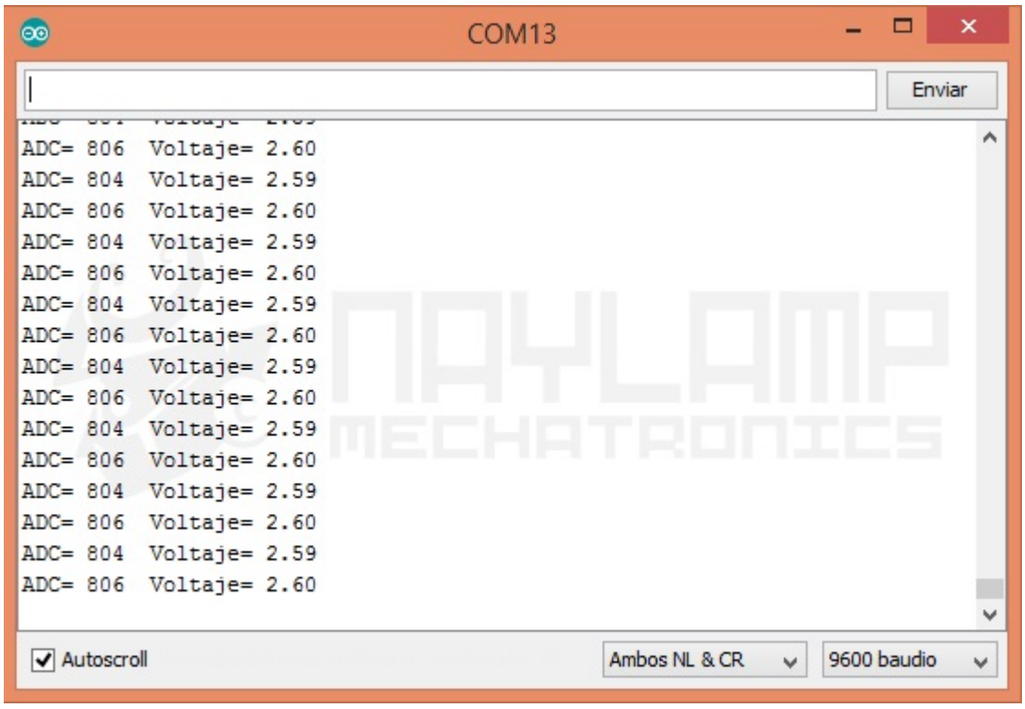
Este ejemplo no podrán implementar los que están trabajando con el ESP-01, y en el caso del NodeMCU y WeMos deben conectar un potenciómetro en el pin A0 pero usando los 3.3v



A nivel de código es igual como si se trabajase con la entrada A0 de un arduino, como se muestra en el siguiente sketch:

```
void setup() {
  //Iniciamos la Comunicacion Serial a 9600 baudios
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0); //Lectura del ADC
  float voltage = sensorValue * (3.3 / 1023.0); //escalamos a voltaje


 //Enviamos por el puerto serie
  Serial.print("ADC= ");
  Serial.print(sensorValue);
  Serial.print("  Voltaje= ");
  Serial.println(voltage);

  delay(100);
}
```

Notar también que estamos usando la comunicación serial y en este ejemplo lo utilizamos para enviar el valor de la lectura analogía a la PC



Otro punto que hay que tener en cuenta es que el rango no es exactamente hasta 3.3V esto porque el divisor de voltaje esta hecho con valores comerciales de resistencias e incluso una resistencia por más precisa que sea puede tener pequeñas variaciones, y si necesitamos un valor exacto del voltaje es necesario escalar el voltaje correctamente.

Para obtener la ecuación correcta para escalar, necesitamos conectar el pin A0 a 3.3V, seguidamente debemos de anotar el valor del ADC, en nuestro caso 966, dicho valor deberán poner en la ecuación en remplazó del número 1023.

```
float voltage = sensorValue * (3.3 / 966.0); //escalamos a voltaje
```

Ej.3 Usando el PWM del ESP8266.

En el ESP8266 podemos usar todos sus pines GPIO como salidas de PWM, la resolución del PWM es de 10 bits a diferencia de un arduino que es de 8bits.

Otra diferencia positiva para el ESP8266 es que podemos modificar la frecuencia del PWM, siendo por defecto de 1KHZ.

Para realizar el ejemplo se puede usar la conexión inicial (el Led en el GPIO2) pero les recomendamos usar esta función en otro pin puesto que le GPIO2 siempre va a parpadear al iniciar el modulo.

```
void setup() {
    pinMode(2, OUTPUT);
}

void loop() {


    for (int PWM_duty = 0; PWM_duty < 1023; PWM_duty++)
    {
      analogWrite(2, PWM_duty);
      delay(1);
    }
    delay(500);
    for (int PWM_duty = 1023; PWM_duty >= 0; PWM_duty--)
    {
      analogWrite(2, PWM_duty);
      delay(1);
```

```
    }
    delay(500);

}
```

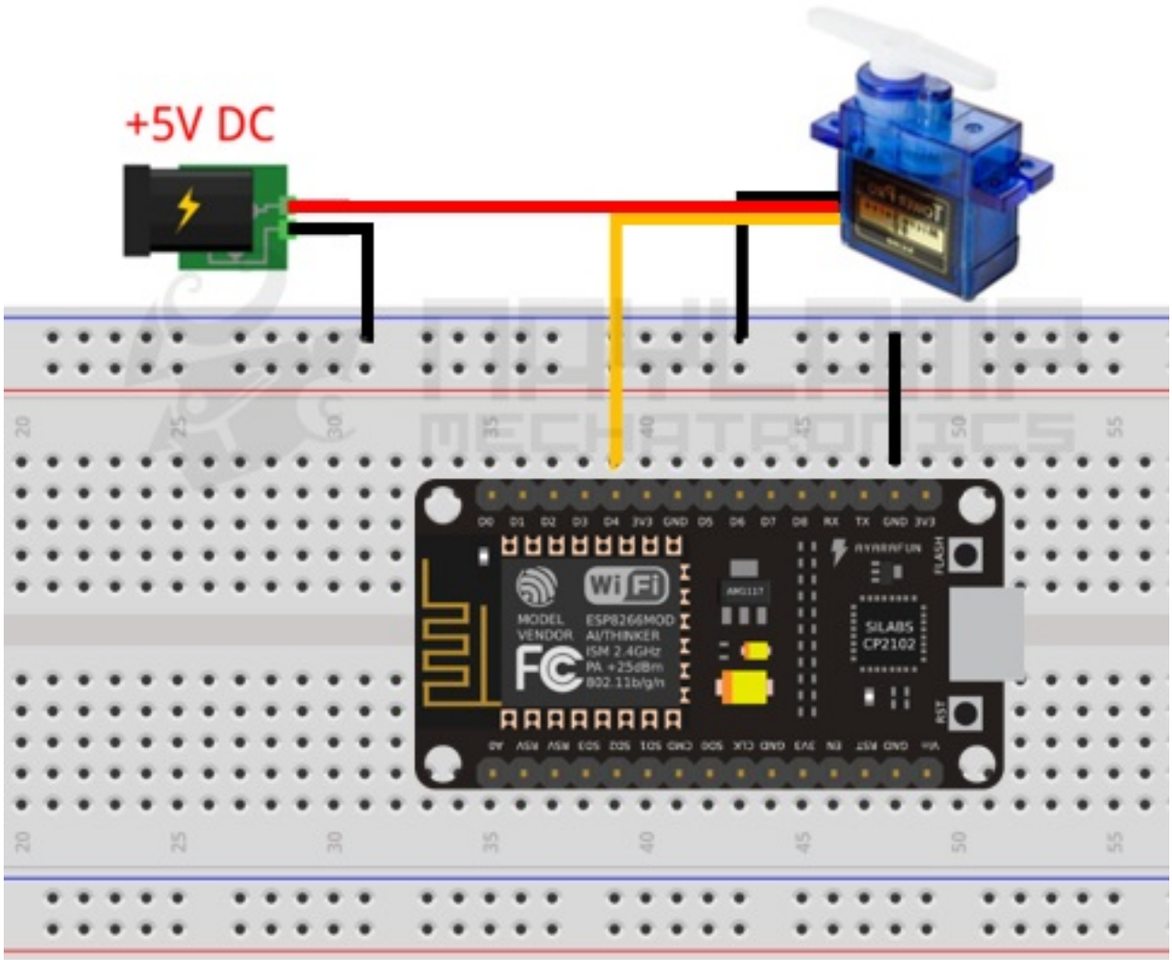
Después de cargar el código el led debe de aumentar y disminuir su intensidad.

En caso quieran modificar la frecuencia del PWM deben de poner la siguiente línea de código en el void setup () con la frecuencia deseada:

```
    analogWriteFreq(2000);  //2KHz
```

Ej.4 Controlando servos con el ESP8266

En este caso hay que tener cuidado con el Pin que usemos, por ejemplo el Pin GPIO2 se utiliza como salida de datos para depurar, por esta razón siempre cuando se inicie o recete el ESP el GPIO2 va a enviar datos por ese pin, eso podría causar movimientos involuntarios en el servo.
En el ejemplo trabajaremos con ese pin pues el ESP-01 solo tiene disponible el GPIO2 pero los que estén trabajando con el NodeMCU o WeMos pueden cambiar de pin.



El código es el siguiente

```
#include  <Servo.h>

Servo  myservo;

void  setup()  {
```

```
  myservo.attach(2);    // servo en el pin 2
}

void loop() {
  for (int angulo = 0; angulo <= 180; angulo += 1)
  {
    myservo.write(angulo);
    delay(10);
  }

  for (int angulo = 180; angulo >= 0; angulo -= 1)
  {
    myservo.write(angulo);
    delay(10);
  }
}
```

Ej.5 Servidor web con ESP8266

En este ejemplo veremos la comunicación wifi y usaremos el ESP8266 como un servidor web.

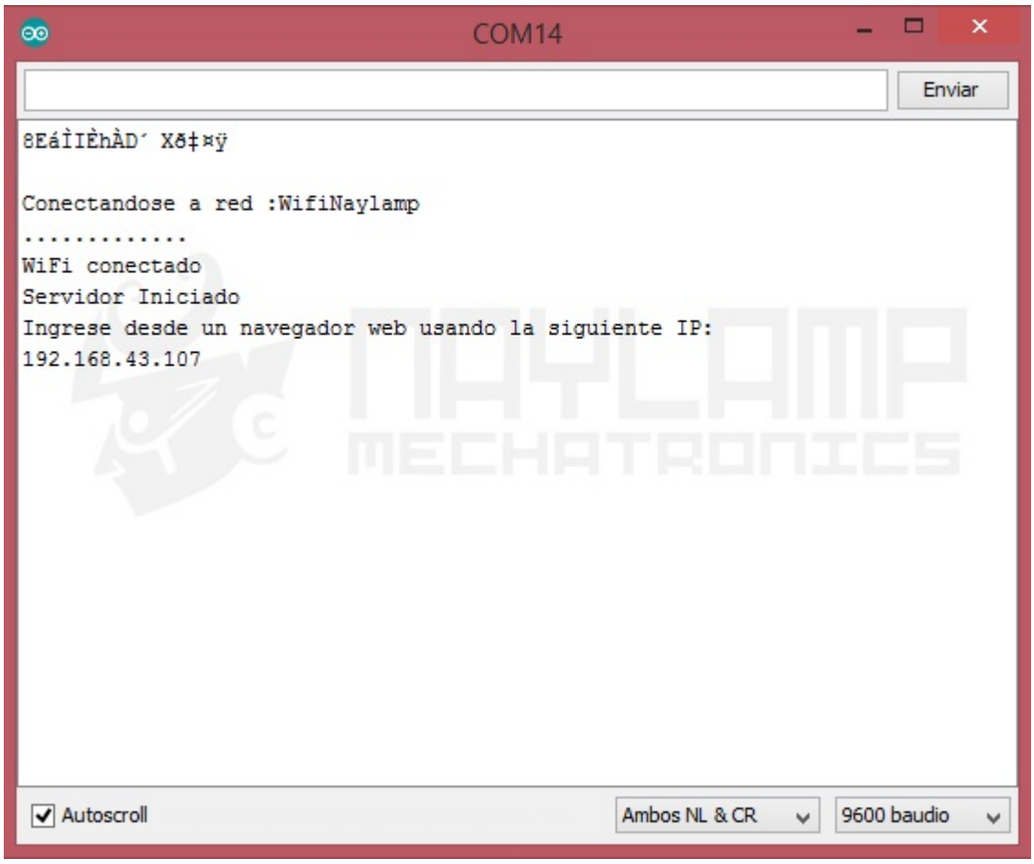
Manipularemos un led que está conectado al GPIO2 del ESP a través de un navegador web. La conexión es la misma del primer ejemplo y el código es el siguiente:

```
#include <ESP8266WiFi.h>

const char* ssid = "WifiNaylamp";
const char* password = "123456789";


WiFiServer server(80);

void setup() {
  Serial.begin(9600);
  delay(10);

  //Configuración del GPIO2
  pinMode(2, OUTPUT);
  digitalWrite(2,LOW);


  Serial.println();
  Serial.println();
  Serial.print("Conectandose a red : ");
  Serial.println(ssid);

  WiFi.begin(ssid, password); //Conexión a la red

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");


  server.begin(); //Iniciamos el servidor
  Serial.println("Servidor Iniciado");


  Serial.println("Ingrese desde un navegador web usando la siguiente IP:");
  Serial.println(WiFi.localIP()); //Obtenemos la IP
}

void loop() {

  WiFiClient client = server.available();
  if (client) //Si hay un cliente presente
  {
    Serial.println("Nuevo Cliente");

    //esperamos hasta que hayan datos disponibles
```

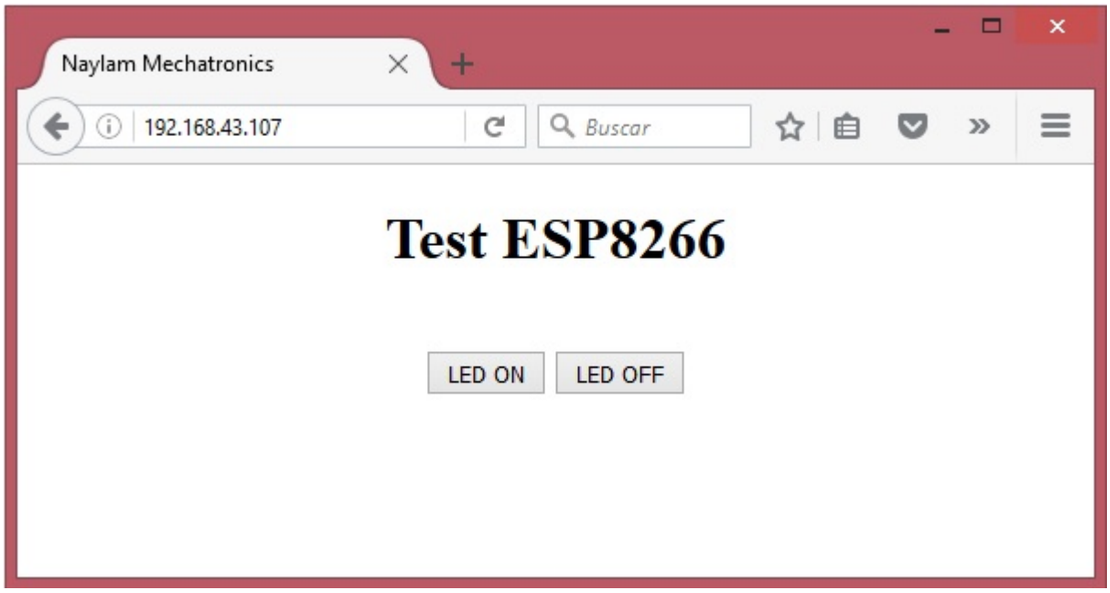
```
      while(!client.available()&&client.connected())
      {
      delay(1);
      }

      // Leemos la primera línea de la petición del cliente.
      String  linea1  =  client.readStringUntil('r');
      Serial.println(linea1);

      if  (linea1.indexOf("LED=ON")>0)  //Buscamos un LED=ON en la 1°Linea
      {
         digitalWrite(2,HIGH);
      }
      if  (linea1.indexOf("LED=OFF")>0)//Buscamos un LED=OFF en la 1°Linea
      {
         digitalWrite(2,LOW);
      }

      client.flush();

      Serial.println("Enviando respuesta...");
      //Encabesado http
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close");// La conexión se cierra después de finalizar de la respuesta
      client.println();
      //Pagina html  para en el navegador
      client.println("<!DOCTYPE HTML>");
      client.println("<html>");
      client.println("<head><title>Naylam Mechatronics</title>");
      client.println("<body>");
      client.println("<h1 align='center'>Test ESP8266</h1>");
      client.println("<div style='text-align:center;'>");
      client.println("<br />");
      client.println("<button onClick=location.href='./?LED=ON'>LED ON</button>");
      client.println("<button onClick=location.href='./?LED=OFF'>LED OFF</button>");
      client.println("<br />");
      client.println("</div>");
      client.println("</body>");
      client.println("</html>");

      delay(1);
      Serial.println("respuesta enviada");
      Serial.println();

   }
}
```

En el código deben modificar las variables correspondientes al nombre y clave de la red.

Después de cargar el programa es necesario abrir el monitor serial para ver si se ha conectado correctamente a la red, y a la vez ver la IP que se nos ha asignado.

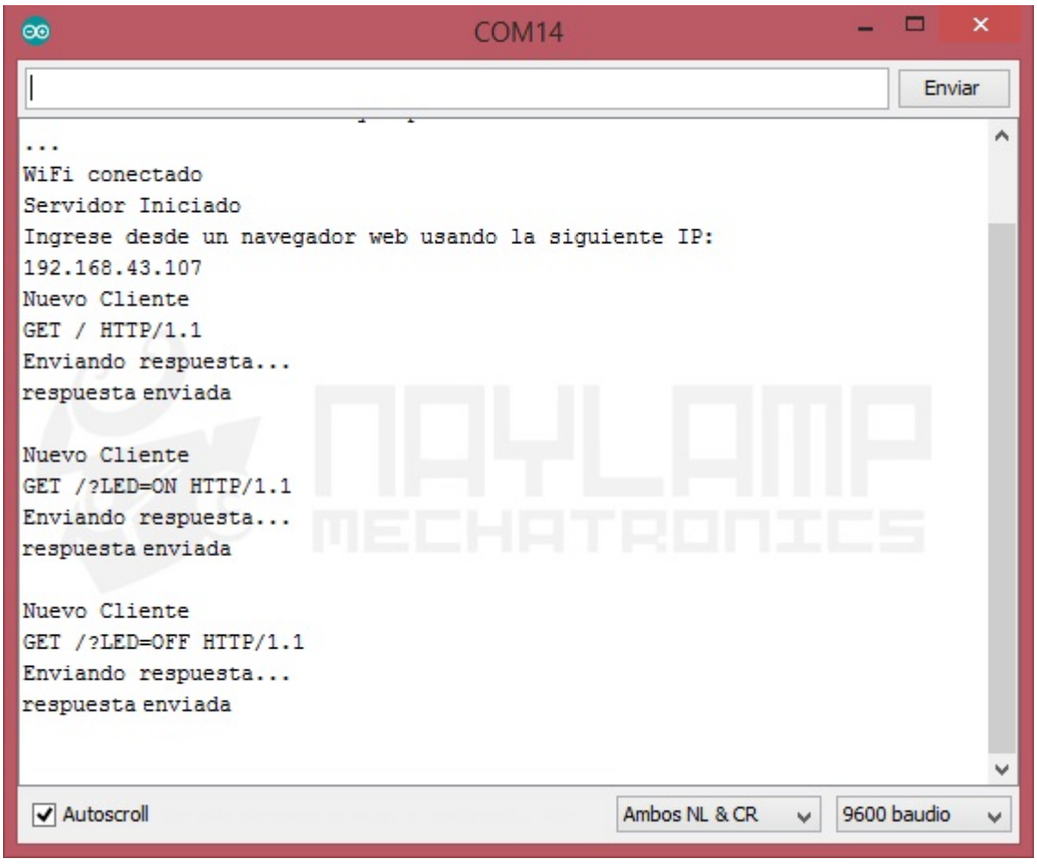


Posteriormente debemos abrir un navegador e ingresar la IP, es necesario que la pc o celular estén en la misma red al que está conectado el ESP8266. Si todo va bien les debe cargar la siguiente página:



Ahora al presionar el botón nuestro ESP realizará la acción correspondiente.

En el monitor serial pueden monitorear las peticiones que se hacen al ESP8266, solo mostramos la primera línea de cada petición http pues es allí donde están los datos que pasamos por URL.


COM14
Enviar
...
WiFi conectado
Servidor Iniciado
Ingrese desde un navegador web usando la siguiente IP:
192.168.43.107
Nuevo Cliente
GET / HTTP/1.1
Enviando respuesta...
respuesta enviada
Nuevo Cliente
GET /?LED=ON HTTP/1.1
Enviando respuesta...
respuesta enviada
Nuevo Cliente
GET /?LED=OFF HTTP/1.1
Enviando respuesta...
respuesta enviada
Autoscroll
Ambos NL & CR
9600 baudio