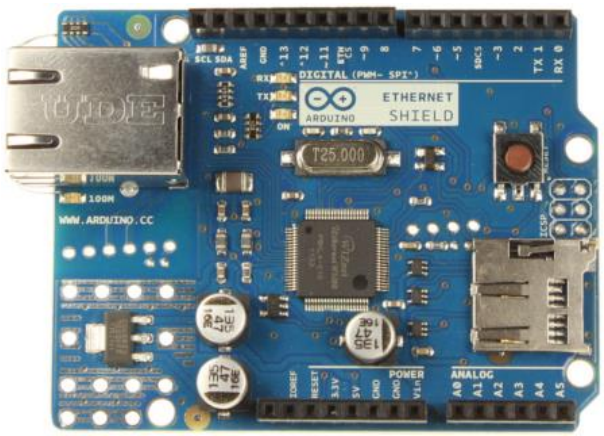
Internet de las Cosas



Práctica 10 – Teoría (v1.6.1 octubre 2022)

Software para robots

Cristian González García

gonzalezcristian@uniovi.es

Basado en el material original de Jordán Pascual Espada

Índice

1. Objetos conectados

En esta práctica se presenta como utilizar varios componentes electrónicos y como conectar un Arduino a internet para crear un servidor Web con este microcontrolador, o bien utilizar el Arduino ara realizarle peticiones para obtener datos.

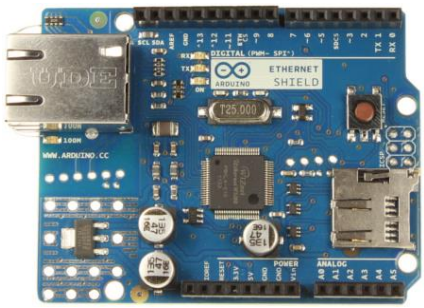
Códigos de estado

Estos son los códigos de error que puede lanzar. Pueden ser consultados en el archivo «wl_definitions.h» de la librería: https://github.com/arduino-libraries/WiFiLink-Library/blob/master/src/utility/wl_definitions.h.

- WL_NO_SHIELD = 255
- WL_IDLE_STATUS = 0
- WL_NO_SSID_AVAIL = 1
- WL_SCAN_COMPLETED = 2
- WL_CONNECTED = 3
- WL_CONNECT_FAILED = 4
- WL_CONNECTION_LOST = 5
- WL_DISCONNECTED = 6

2. Servidor Web (Con Arduino LAN)

En este ejemplo veremos cómo crear un servidor web utilizando un microcontrolador Arduino como servidor. Con ello, permitiremos encender y apagar un LED que está conectado al Arduino dependiendo de las peticiones que reciba.



Arduino Ethernet Shield: esta placa de extensión permite conectar un Arduino a Internet mediante un cable de Ethernet.

Soporta varios sockets de conexión simultáneos. Para su uso se recomienda utilizar la Ethernet Library. https://www.arduino.cc/en/Reference/Ethernet

Incluye lector de tarjetas microSD.

Construcción del circuito:

Encajar la placa de extensión cuidadosamente (el Shield) en la placa Arduino. Si la placa está dañada o mal encajada puede provocar un mal funcionamiento parcial de algunos pines, o incluso total.

¡Cuidado! Al encajar algunos modelos de placa **el pin digital 13 queda debilitado**, lo que implica que si se conecta un LED este **alumbre menos o no alumbre**. Otras veces, **funciona de forma intermitente** ignorando nuestros comandos.

Conectamos el cable de red a una toma de un ordenador. Estos ordenadores no tienen IP publica, por lo que se generará un servidor local accesible por ordenadores que estén en la misma red. Así que, previamente, mirar los datos de ese ordenador con el comando «ipconfig» desde el CMD.

Conectamos la patilla R del LED RGB a la patilla 8, y la patilla - del led a una GND

Programación de Arduino:

Requiere incluir dos librerías oficiales de Arduino:

- Program -> Include Library -> (SPI & Ethernet).
  - SPI: https://www.arduino.cc/en/Reference/SPI
    - Permite comunicar el Arduino con dispositivos utilizando la interfaz SPI (Serial Peripheral Interface), como puede ser con otro microcontrolador.
  - Ethernet: https://www.arduino.cc/en/Reference/Ethernet
    - Permite al Arduino conectarse a Internet y convertirlo en servidor web, recibir peticiones y enviarlas.
    - Requiere importar la librería SPI.

Importante

- Hay que cambiar la IP del ejemplo. Mirad que IP tiene el ordenador de al lado, copiad sus datos y coged esa toma de Ethernet. Si al mismo tiempo varios Arduino tienen la misma IP habrá problemas de conexión. **¡Cuidado!**
- La MAC también tiene que ser única, cambiad el último dígito por el de vuestro grupo. Muchas redes piden que sea única por motivos de seguridad y si dos colisionan, pueden bloquear esa MAC; o incluso la boca de Ethernet. **¡CUIDADO!**
- Si la IP y/o la MAC están repetidas en la misma red local, se encontrarán duplicados en la red y puede que envíe los mensajes de uno a otro, que funcione todo mal, que baneen esas MAC el servicio de informática por encontrar funcionamientos extraños, o incluso la boca de red, etc.
- Comprobad los datos del Gateway y de la subnet usando el comando ipconfig en el CMD de Windows. Si lo hacéis en casa o en otro laboratorio, estos pueden ser totalmente diferentes.

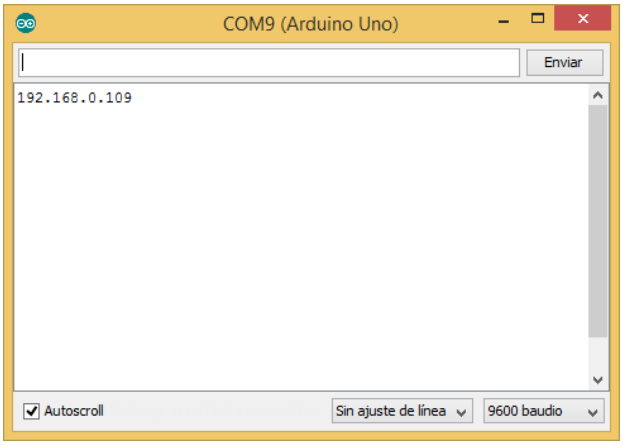
```
#include <SPI.h> //Importamos librería comunicación SPI
#include <Ethernet.h> //Importamos librería Ethernet
```

```
    byte mac[] = {0x54, 0x55, 0x58, 0x10, 0x00, completarla}; // Tiene
que ser única en la red local, cuidado, cambiad el último dígito por el
de vuestro grupo
    EthernetServer servidor(80); // Puerto en el 80
    // Estos datos cogedlos del ordenador que uséis o del de al lado
    IPAddress dnsServer(8, 8, 8, 8);
    IPAddress gateway(192, 168, 61, 13);
    IPAddress subnet(255, 255, 255, 0);

    // Que cada uno ponga la IP de su grupo (20X, dónde X es el número
del grupo) 201, 202, 203, que es el que tiene asignado. Tiene que ser
única en la red local, cuidado
    IPAddress ip(192, 168, 61, completar);

    int led = 8;
    void setup(){
     Serial.begin(9600);
       Ethernet.begin(mac, ip, dnsServer, gateway, subnet);
       servidor.begin();
       Serial.println("Setup");
       // Imprimir la IP
       Serial.println(Ethernet.localIP());

       // Inicializar el led
       pinMode(led,OUTPUT);
       digitalWrite(led,LOW);
    }

    void loop(){
      EthernetClient cliente = servidor.available();

      if (cliente) {
        Serial.println("Nueva peticion");
        String peticion="";
        while (cliente.connected()) {
          if (cliente.available()) {

            char  c  =  cliente.read();  //Leer   petición  carácter  a
carácter
            peticion.concat(c); // concatenar en un string

            // Ha acabado la peticion http
            // Si contiene index responder con una web

            // la petición ha acabado '\n' y contiene la cadena "index"
al principio: index, indexabc, indexación, etc. Usar equals para que sea
igual
            if (c == '\n' && peticion.indexOf("index") != -1){
                Serial.println("Responder");
                // Serial.println(peticion);

                // contiene la cadena "encender"
                if(peticion.indexOf("encender") != -1){
                  Serial.println("Encender Led");
                  digitalWrite(led,HIGH);
                }else if(peticion.indexOf("apagar") != -1){
                  // contiene la cadena "apagar"
                  Serial.println("Apagar led");
                  digitalWrite(led,LOW);
                }
```

```
                // Enviamos al cliente una respuesta HTTP
                cliente.println("HTTP/1.1 200 OK");
                cliente.println("Content-Type: text/html");
                cliente.println("Access-Control-Allow-Origin: *");
                cliente.println();
                break;
            }
          }
        }

        // Pequeña pausa para asegurar el envio de datos
        delay(1000);
        cliente.stop();// Cierra la conexión
      }
    }
```

Prueba

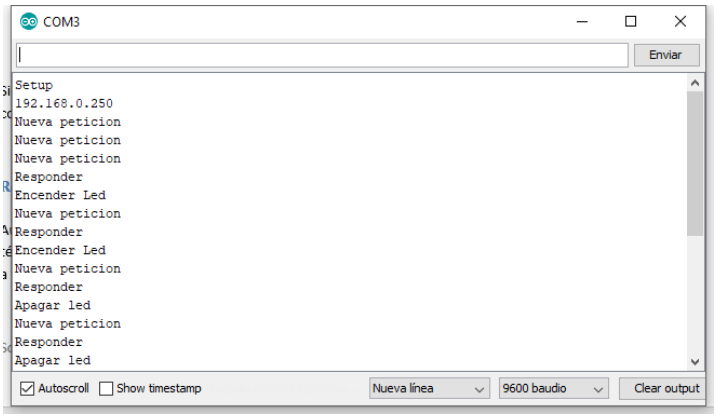
Ejecutamos la aplicación y abrimos el monitor. Tenemos que ver la IP que se ha asignado al servidor.



Desde un **navegador web**, que se encuentre en un ordenador **conectado a la misma red**, ejecutamos las siguientes peticiones:

- http://**<mi_ip>**/index?encender
- http://**<mi_ip>**/index?apagar

Si recibe una petición y esta todo configurado correctamente, debería de mostrarlo en la consola.



Si no muestra que le llegan peticiones, significa que está mal configurado. Revisar los datos del ordenador del cuál se los cogisteis y desenchufasteis el cable de Ethernet y comprobad que estáis accediendo desde otro ordenador en la misma red (por Ethernet).

Retornar HTML

Aunque no es muy recomendable que un Arduino Uno funcione como servidor web, es técnicamente posible, aunque solo será apropiado si sirve páginas web de muy poco peso que vayan a tener muy pocos usuarios (o solo uno).

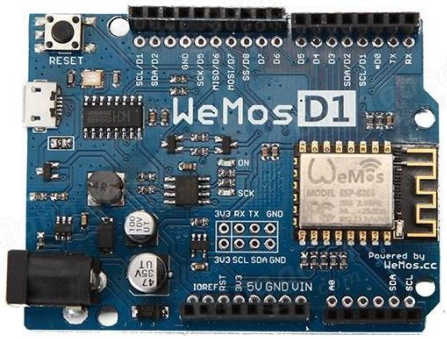
Si la página web es grande, debemos utilizar la tarjeta SD para almacenarla.

Incluir en la respuesta el siguiente código, que contiene la web que servirá el Arduino:

```
          // Enviamos al cliente una respuesta HTTP
          cliente.println("HTTP/1.1 200 OK");
          cliente.println("Content-Type: text/html");
          cliente.println("Access-Control-Allow-Origin: *");
          cliente.println();

          cliente.println("<html>");
          cliente.println("<body>");
          cliente.println("<h1>Control de luz por internet</h1>");
          cliente.println("<h2><a
href='index.html?p=encender'>Encender</a></h2>");
          cliente.println("<h2><a href='index.html?p=apagar'>Apagar</a></h2>");
          cliente.println("</body>");
          cliente.println("</html>");

          break;
```

3. Servidor Web (Con WeMos D1) (Obsoleta)

En este ejemplo veremos cómo crear un servidor web con la placa Arduino Wemos D1. No obstante, esta versión es la antigua de este microcontrolador y está obsoleta. Se deja como ejemplo de otro hardware diferente.



WeMos D1 es una placa compatible con Arduino. Esta placa cuenta con un módulo Wi-Fi ESP8266 Integrado:

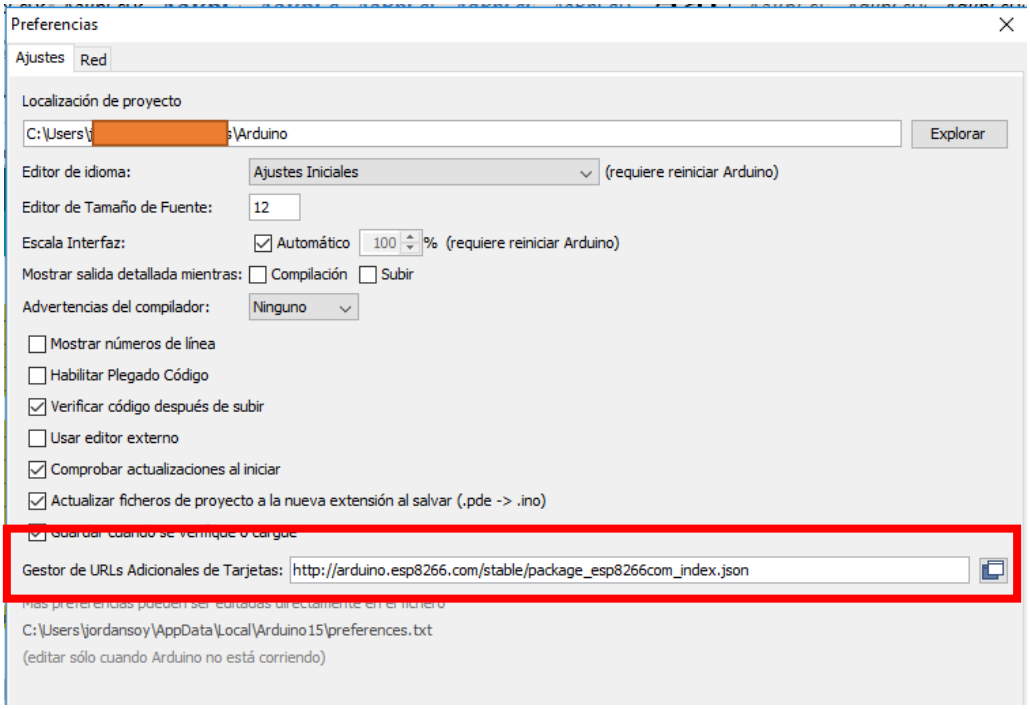
- Esta placa no está incluida en el IDE por lo que debemos instalar sus drivers.
- **Todos los pines digitales de esta placa admiten PWM**.
- **El LED integrado está en el pin D5 en lugar del 13.**
- **Tiene un único pin de entrada analógica**
- Hay otras versiones diferentes también, pues esta es de las primeras versiones y ya hay versiones mucho más nuevas y pequeñas, aunque requieren que se suelden sus pines: https://www.wemos.cc/en/latest/.

Se accede a ella utilizando Internet, vía Wi-Fi. Para simplificarlo, se puede utilizar una red Wi-Fi que no requiere usuario ni contraseña. Para esto, en vez de utilizar la de UniOvi es mejor y más sencillo crearse un punto de acceso que no los requiera desde el móvil o el portátil.

Instalación de la placa:

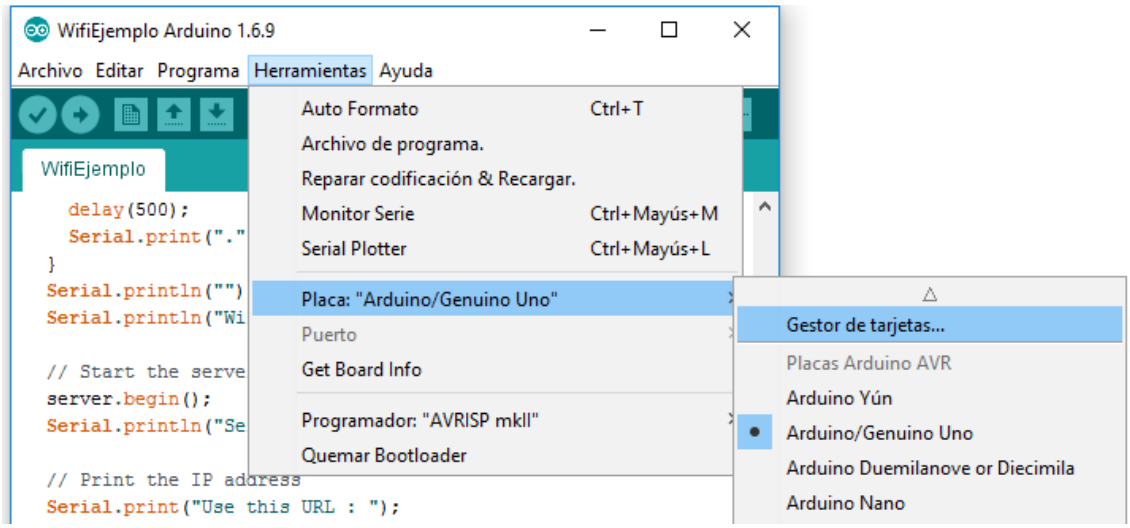
Debemos instalar el plugin ESP8266 (es el módulo Wi-Fi). Para ello, abrir el IDE de Arduino y entrar en **Archivo -> Preferencias.**

Localizar el campo «Gestor de URLs Adicionales de Tarjetas» e incluir la URL: https://arduino.esp8266.com/stable/package_esp8266com_index.json

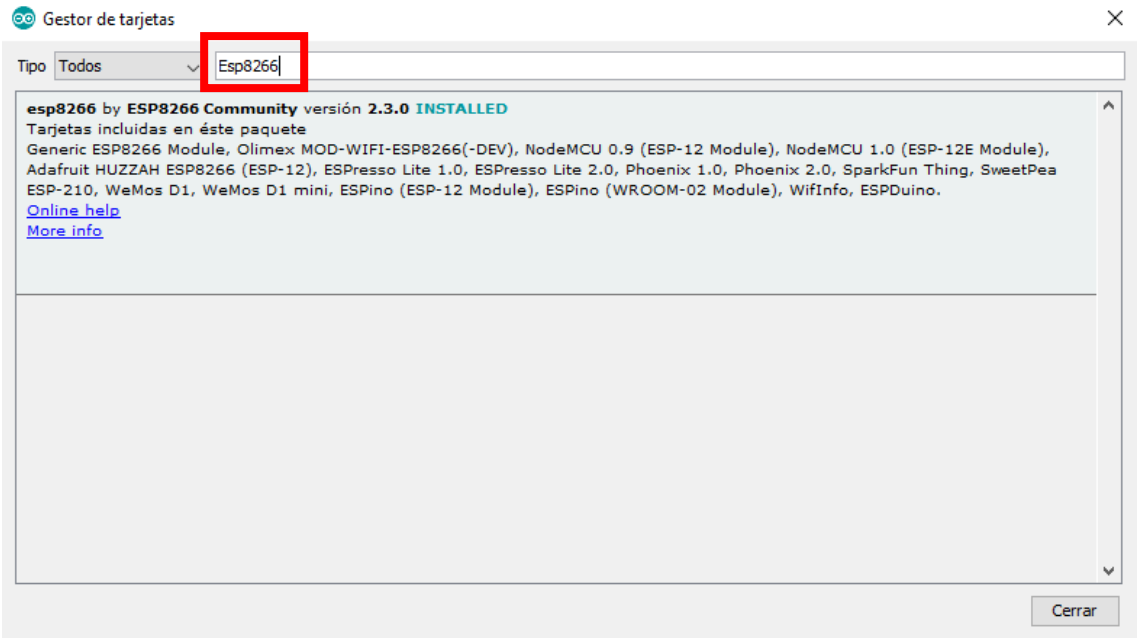


Pulsar **Ok** y volver a la pantalla principal del IDE de Arduino.

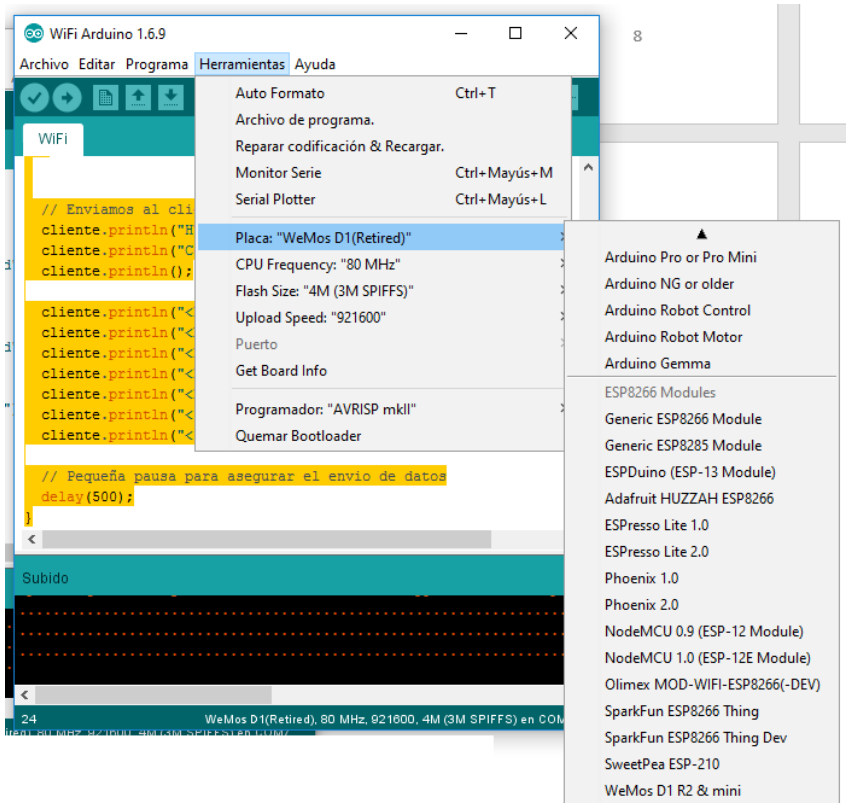
Seleccionar **Herramientas -> Placa -> Gestor de tarjetas…**



En el cuadro de búsqueda hay que introducir **Esp8266** y después pulsar en **Instalar** la coincidencia «esp8266 by ESP8266 Community». **Puede tardar un rato en reconocerla o salir**, pues tiene que bajarla después de que hayáis introducido la ruta en el paso anterior. Esta placa necesita un conectar micro USB tipo B micro en vez del USB tipo B del Arduino.



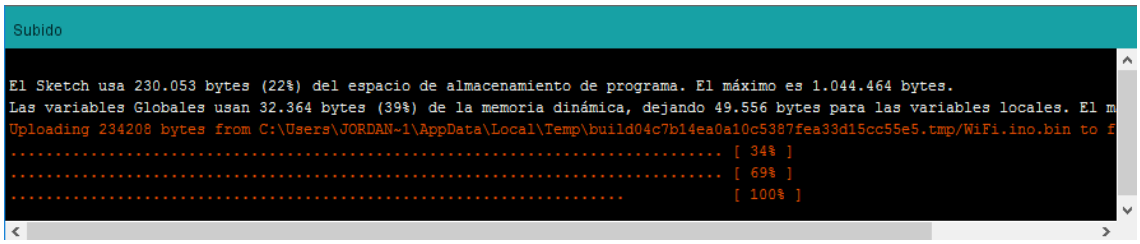
Luego nos aparecerá una lista de placas más amplia, bajábamos el scroll hasta encontrar «**Wemos D1 (Retired)**».



No debemos olvidarnos de seleccionar el **Puerto**, al igual que hacíamos con el resto de placas.

También se le puede cambiar la **CPU Frequency** a 160Mhz, pero lo que vamos a hacer no es demasiado crítico.

Probamos a ejecutar el programa vacío y debería aparecer el mensaje de subido.



Particularidades de la placa:

- En el código los pines digitales se nombran con D2, D3, D4, etc.
- En el pin D5 tiene un led integrado (Está en el medio de la placa).

Conectamos la patilla R del LED RGB a la patilla D8, y la patilla - del led a una GND.

Los pins digitales de está placa se mueven en un rango de valores 0 – 3.3V en lugar de 0 – 5V como ocurre en otras placas arduino.

Programación de Arduino:

Nota: si da problemas con una red que requiera usuario y contraseña, además del nombre de la red (SSID) y la contraseña, podéis probarlo en clase creando una red Wi-Fi portátil con vuestro móvil.

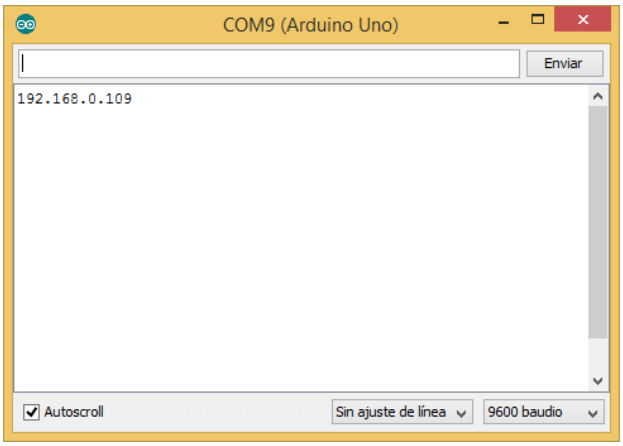
```
#include <ESP8266WiFi.h>

const char* ssid = "ICTCampus";
const char* password = "";

IPAddress dnServer(156, 35, 14 , 2);
IPAddress gateway(192, 168, 61, 13);
IPAddress subnet(255, 255, 255, 0);

// Que cada uno ponga la IP de su grupo 201, 202, 203, que es el de su
caja
IPAddress ip(192, 168, 61, 211);

int ledPin = D8;
WiFiServer server(80);
```

```
    void setup() {
      Serial.begin(9600);
      Serial.print("Start");
      delay(10);

      pinMode(ledPin, OUTPUT);
      digitalWrite(ledPin, LOW);

      // Conectar a la red Wifi
      Serial.print("Conectando a "+String(ssid));

      WiFi.config(ip, gateway, subnet);
      WiFi.begin(ssid, password);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }

      Serial.println("Conectado al WiFi");

      // Iniciar un servidor
      server.begin();
      Serial.print("Servidor Web en: http://");
      Serial.print(WiFi.localIP());
    }

    void loop() {
      // Comprueba clientes conectados
      WiFiClient cliente = server.available();
      if (!cliente) {
        // Mientras no haya cliente repito el bucle
        return;
      }

      // Hay cliente!
      Serial.println("Nuevo cliente, esperar mientras no está listo");
      while(!cliente.available()){
        delay(1);
      }

      // Leo la petición, hasta el salto de linea \n
      String peticion = cliente.readStringUntil('\n');
      Serial.println(peticion);
      cliente.flush();

      // Miro si contiene alguna de las dos cadenas
      if (peticion.indexOf("encender") != -1) {
        digitalWrite(ledPin, HIGH);
      } else if (peticion.indexOf("apagar") != -1){
        digitalWrite(ledPin, LOW);
      }


      // Enviamos al cliente una respuesta HTTP
      cliente.println("HTTP/1.1 200 OK");
      cliente.println("Content-Type: text/html");
      cliente.println();

      cliente.println("<html>");
      cliente.println("<body>");
      cliente.println("<h1>Control de luz por internet</h1>");
      cliente.println("<h2><a
href='index.html?p=encender'>Encender</a></h2>");
      cliente.println("<h2><a href='index.html?p=apagar'>Apagar</a></h2>");
      cliente.println("</body>");
      cliente.println("</html>");
```

```
        // Pequeña pausa para asegurar el envio de datos
        delay(500);
    }
```

Prueba

Ejecutamos la aplicación y abrimos el monitor. De esta forma veremos la IP que se ha asignado al servidor.



4. Sensores

Sensores de temperatura y humedad

DHT11

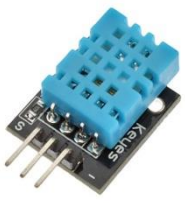
Es un sensor analógico que envía los datos por un pin digital. Esto es debido a que internamente hace ese cambio de analógico a digital y por medio de una librería podremos interpretar estos datos digitales. **Internamente**, se compone de un **sensor de humedad y un termistor**, además de un **conversor de analógico a digital**.

Utiliza su propio protocolo para enviar los valores y, o bien hay que cumplirlo, o bien utilizar una librería que lo haga por nosotros:

- 40 bits durante 4ms
    - 2 bytes para temperatura
    - 2 bytes para humedad
    - 1 byte para comprobación de errores

Gracias al uso de su propio protocolo, se puede utilizar en un cualquiera pin digital.

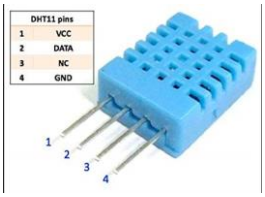
Cuidado: si se usa el *shield* de Ethernet, puede que solo funcione en los pines que coinciden con los pines PWM del Arduino. En el Arduino funciona en cualquier pin digital.



Sensor de temperatura y humedad DHT11. Recoge de forma digital las mediciones de temperatura y humedad. La frecuencia de actualización es de un segundo:

- Alimentación: 3-5V DC
- Rango de temperatura: 0 a 50ºC. Precisión de 2 grados.

- Rango de humedad: 20-95% de humedad relativa. Precisión de 5%.
- Rango de muestreo de 1MHz (1 vez por segundo). Si se pone más, lanzará error -1.
- Versión con placa: resistor pull-up y LED.



Si se usa la versión que viene sin placa, la tercera patilla por la izquierda, no se conecta. Además, cambia el orden de pines: 1 VCC, 2 pin digital, 4 GND. El pin 3 no hace nada.

También se necesitaría incluirle una resistencia de 5-10k en una la entrada de la patilla 1 y 2 para que funcionara correctamente.

DHT22

El sensor DHT22 es igual que el DHT11, pero tiene mayor precisión. Se conecta exactamente igual y se utiliza la misma librería.

DHT11 vs DHT22

A continuación, se muestra la tabla comparativa entre ambo sensores:

PARÁMETRO\SENSOR	DHT11	DHT22
ALIMENTACIÓN	3-5V DC	3.3-6V DC
SALIDA	Digital	Digital
RANGO TEMPERATURA	Entre 0 y 50 °C	De -40°C a 80 °C
PRECISIÓN TEMPERATURA	±2 °C	<±0,5 °C
RESOLUCIÓN TEMPERATURA	0,1°C	0,1°C
RANGO DE HUMEDAD	De 20% a 90% RH	De 0 a 100% RH
PRECISIÓN HUMEDAD	±4/5% RH	±2% RH
RESOLUCIÓN HUMEDAD	1%RH	0,1%RH
TIEMPO DE RESPUESTA / RESOLUCIÓN	1s / 8-bits	2s

TAMAÑO	12 x 15.5 x 5,5mm	14 x 18 x 5,5mm
PINES	4 (solo se usan el 1, 2 y 4)	4 (solo se usan el 1, 2 y 4)

Tabla 1 Comparativa entreel DHT11 y el DHT22

Ejemplo de uso del DHT

En este ejemplo veremos cómo utilizar un sensor de temperatura y humedad analógico DHT11 que envía los datos utilizando un **pin digital**.

- Conexiones
  - **S** a un pin digital, **Pin central** a 5v, y **–** a GND

Programa

Utilizaremos la **librería DHT Sensor Library de Adafruit**. Hay que buscarla en el gestor de librerías e instalar la última versión.

```
#include <DHT.h> // Librería externa

// Descomentar el sensor a utilizar:
#define DHTTYPE DHT11  // DHT 11
//#define DHTTYPE DHT22  // DHT 22 (AM2302)
//#define DHTTYPE DHT21  // DHT 21 (AM2301)

int pin_sensor = 5;

DHT dht(pin_sensor, DHTTYPE);

float temperature, humidity;

void setup(){
  Serial.begin(9600);
  dht.begin();

  temperature = 0;
  humidity = 0;
}

void loop(){
   temperature =  dht.readTemperature();
   humidity = dht.readHumidity();

   if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Fallo en la lectura del "+DHTTYPE);
    return;
   }

    Serial.println("Temperatura: "+String(temperature));
```

```
        Serial.println("Humedad: "+String(humidity));
        Serial.println("--------------------------------");

      delay(1000); // Actualizar cada segundo
    }
```

Envío y tratamiento de los datos

Para enviar los datos hay varias opciones, de más fácil a más difícil:

- Crear los datos en Arduino usando la opción el método «println()» de EthernetClient.
- Elegir como se enviarán los datos: texto plano, XML, JSON, etc.
- Incrustar estos datos en los «println» de Arduino.
- En JavaScript recibir esto e incorporarlo al mensaje o incrustarlo en la web. Dependiendo del método, igual hay que usar librería para parsear el XML (DOMParser().parseFromString(xmlStr, "application/xml")) o el JSON (JSON.parse), trabajar con estos datos y mostrarlos como se quieran.

Consideraciones:

- Colocar el Content-Type en el Arduino y el MimeType en JavaScript correspondiente:

Tipo	Ejemplo	Content-Type	MimeType
Texto	…" "+Tem: " + String(temperature) + " Hum: " + String(humidity)+" "…	'text/html'	'text/html'
Texto	String(temperature) String(humidity)	'text/html'	'text/html'
XML	<data> <tem>15</tem> <hum>15</hum> </data>	'application/xml'	'application/xml'
JSON	{"tem":"15", "hum":"65"}	'application/json'	'application/json'

5. Posibles problemas

Conexión a Internet

En la Universidad de Oviedo, e incluso en otros sitios, la Wi-Fi y Ethernet van por una red diferente. Luego, si conectamos el Arduino a Ethernet y nosotros accedemos por Wi-Fi no lo veremos y no podremos enviarle peticiones. Fijaros que en el ejemplo usamos la IP 192.168.X.X, que es local y no es accesible desde afuera. Esto mismo ocurre si tratamos de acceder utilizando los datos del móvil, pues es diferente red. En cambio, si se usa un ordenador del laboratorio, estos van por Ethernet.

En cambio, en casa, si solo tenemos un router, que suele ser lo normal, Ethernet y la Wi-Fi irán por la misma red.

Access-Control-Allow-Origin

Cuidado con el error «Access-Control-Allow-Origin» de «Cross-Origin Resource Sharing» y de activarlo en el Arduino: https://www.html5rocks.com/en/tutorials/cors/. Para evitar que esto suceda, hay que dar permisos en la página web creada/generada en el Arduino para que pueda recibir peticiones desde fuera con la cabecera: «Access-Control-Allow-Origin: *».

Si se pone con un *, cualquiera puede acceder y es la forma fácil de probar que todo funciona. No obstante, **cuidado** con poner un * en vuestros scripts, por **seguridad**, ya que esto permitiría acceder a cualquiera.

También se puede **restringir a IPs**, dominios, métodos, tipo de contenido, etc.: https://www.html5rocks.com/en/tutorials/cors/

Google Maps

- Tener habilitado el pago en https://console.cloud.google.com/project/_/billing/enable
  - Dan 200€ gratis, cuidado con las peticiones.
- ERR_CONNECTION_TIMED_OUT: demasiadas peticiones en poco tiempo, error o mala conexión en algún sensor (DHT11).
- Guías:
  - https://developers.google.com/maps/documentation/javascript/examples/map-coordinates
  - https://developers.google.com/maps/documentation/javascript/examples/event-poi
  - Marcadores: https://developers.google.com/maps/documentation/javascript/examples/marker-simple