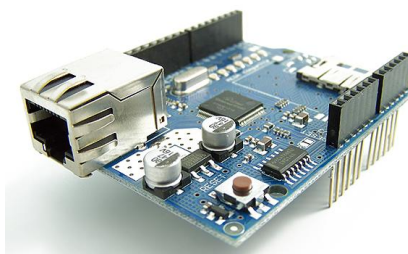
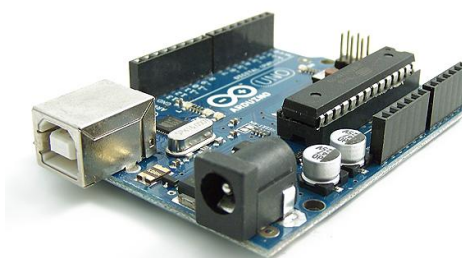


# Arduino + Ethernet Shield

**Implantación de Arduino en las redes Ethernet:  
“Arduino y el Internet de las Cosas”**



+



Ver. 1.0

Serie: Arduino Comunicación

**José Manuel Ruiz Gutiérrez**



## Índice

1. Objetivo de este trabajo
2. Presentación de Arduino Ethernet Shield
3. Conectando el Shield
4. Conectar con una página Web sabiendo su IP
5. Probando la aplicación Ethernet Shield con el Hyperterminal de Windows
6. Arduino como servidor en una red Ethernet: Lectura de 6 canales analógicos de la tarjeta Arduino UNO.
7. Gobierno de un Relé con el Shield Arduino Ethernet.
8. Leer una señal digital y una analógica.
9. ANEXO I: Librería Ethernet



Este trabajo está bajo licencia [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by-nc-nd/3.0/)

12 Enero de 2013 Versión de Documento: Versión. 1

Blog de referencia: <http://josemanuelruizgutierrez.blogspot.com>  
[j.m.r.gutierrez@gmail.com](mailto:j.m.r.gutierrez@gmail.com)

# 1. Objetivo de este trabajo.

Con este trabajo se pretende facilitar un “primer contacto” con el shield Ethernet para Arduino y sus aplicaciones orientadas al gobierno y la monitorización de señales en modo remoto, haciendo uso de la comunicación de protocolo TCP/IP característica de la red Internet

Con la aplicación de este shield se amplían notablemente las potencialidades de la plataforma Open Hardware Arduino y se crean nuevas expectativas de cara a la integración de Arduino en Internet bajo el nuevo paradigma denominado “El internet de las cosas”

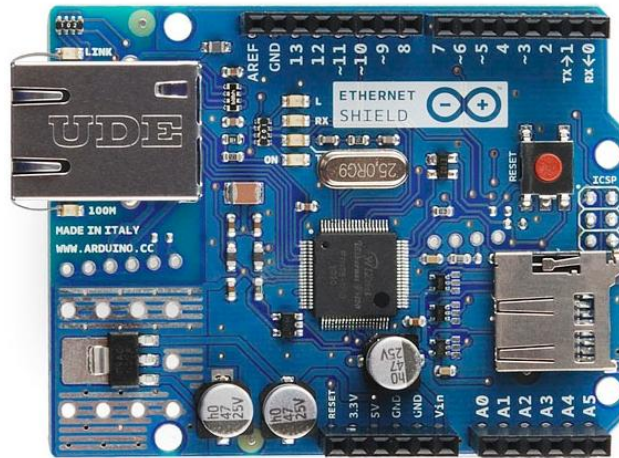
Existen diversas tarjetas que cumplen con las prestaciones de elemento Ethernet de este shield. En nuestro caso vamos a trabajar con la genuina, desarrollada por el grupo de diseño y aplicaciones de Arduino.

Pretendo con este trabajo facilitar una guía de iniciación a quienes deseen experimentar con esta tecnología.

El manual que sigue aborda una explicación básica del sistema, los métodos y herramientas para la programación de la unidad y unos cuantos ejemplos de aplicaciones.

Como con el resto de mis trabajos deseo que este documento sea de ayuda a quienes lo utilicen. Agradezco a las personas e instituciones que me precedieron en la elaboración de documentación sus aportaciones y animo a quien realice algún trabajo en este campo a que lo comparta con todos los que formamos esta enorme comunidad de usuarios de Arduino.

## 2. Presentación de Arduino Ethernet Shield



*Arduino Ethernet Shield*

Recomiendo la lectura de los documentos que figuran en el ANEXO de Bibliografía con el fin de poder acceder a la información ampliada.

El Shield Arduino Ethernet se conecta a Internet en cuestión de minutos. Sólo tiene que conectar este módulo en la placa Arduino, conectarlo a la red con un cable RJ45 y seguir algunas instrucciones sencillas para empezar a controlar el mundo a través de Internet. Como siempre con Arduino, todos los elementos de la plataforma - hardware, software y documentación - son de libre acceso y de código abierto.

Estas son algunas de las características del shield:

- Tensión de alimentación 5V (se alimenta directamente desde la tarjeta Arduino)
- Controlador Ethernet: W5100 con una memoria interna de 16K
- Velocidad de conexión: 10/100Mb
- Conexión con Arduino a través del Puerto SPI

### **Descripción**

Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Se basa en el chip Wiznet W5100 ethernet (hoja de datos). El W5100 Wiznet proporciona una red (IP) de pila capaz de TCP y UDP. Soporta hasta cuatro conexiones de socket simultáneas. Utilice la biblioteca de Ethernet a escribir sketches que se conectan a Internet a través de la pantalla. El shield de Ethernet se conecta a una placa Arduino con largas Wire Wrap-headers que se extienden a través del shield. Esto mantiene la disposición de las clavijas intacta y permite que otro shield para ser apilados en la parte superior.

The Shield Ethernet tiene un estándar de conexión RJ-45, con un transformador de línea integrada y Power over Ethernet habilitado.

Hay una ranura para insertar una tarjeta micro-SD, que puede ser usado para almacenar archivos para servir a través de la red. Es compatible con el Arduino Uno y Mega (usando la biblioteca de Ethernet). El CODEC de lector de tarjetas microSD se puede acceder a través de la Biblioteca SD. Cuando se trabaja con esta biblioteca, SS es el pin 4.

La pantalla también incluye un controlador de reajuste, para asegurarse de que el módulo Ethernet W5100 se reinicia correctamente en el encendido. Las revisiones anteriores del shield no eran compatibles con la Mega y la necesidad de restablecer manualmente después del encendido.

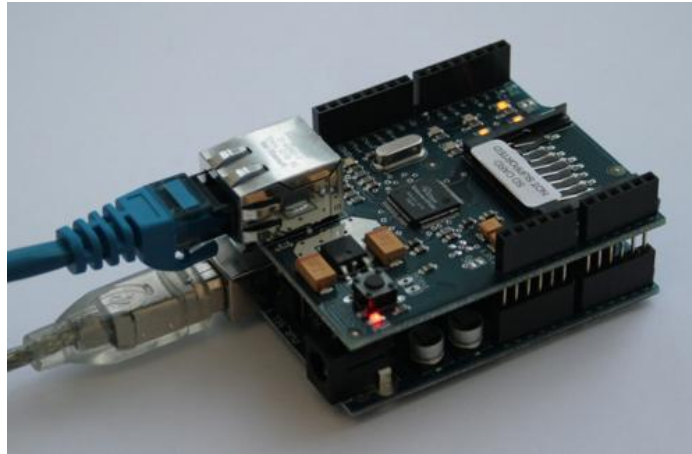
El shield actual tiene una alimentación a través de Ethernet (PoE), módulo diseñado para extraer energía de una convencional de par trenzado Categoría 5 cable Ethernet:IEEE802.3af compliant. El Shield dispone de un conector RJ45 para conectarse a una red Ethernet. El botón de reinicio sirve para reiniciar el Shield y la propia tarjeta Arduino. El shield contiene un número de LEDs informativos:

- **PWR**: indica que Arduino y el shield están alimentados
- **LINK**: indica la conexión a una red y parpadea cuando el shield transmite o recibe datos
- **FULLD**: indica que la conexión de red es full duplex
- **100M**: indica la presencia de una conexión de red 100 Mb / s (en lugar de 10 Mb / s)
- **RX**: parpadea cuando el shield recibe datos
- **TX**: parpadea cuando el shield envía datos
- **COLL**: parpadea cuando se detectan colisiones de red

El puente de soldadura de la marca "INT" puede conectarse para permitir que la placa Arduino reciba interrupciones y sean notificadas desde W5100, pero esta opción no está en la librería estándar de Ethernet. El puente conecta el pin INT del W5100 para pin digital 2 de la Arduino.

### 3. Conectando el Shield

El Shield Arduino Ethernet permite la inserción sobre una tarjeta convencional Arduino utilizando para su gobierno la [librería Ethernet](#) de la que hemos colocado un ANEXO en este documento..



Arduino Ethernet Shield sobre una placa Arduino

**Para la puesta en marcha hay dos pasos fundamentales:**

**PRIMERO:** Para usar la Ethernet Shield solo hay que montarla sobre la placa Arduino (p. e. una Diecimila). Para cargar los sketches a la placa Arduino se conecta esta al ordenador mediante el cable USB como se hace normalmente. Una vez que el sketch ha sido cargado se puede desconectar la placa del ordenador y alimentarla desde una fuente externa.

**SEGUNDO:** Conectar la Ethernet Shield a un ordenador, a un switch o a un enrutador utilizando un cable ethernet standard (CAT5 o CAT6 con conectores RJ45). La conexión al ordenador puede requerir el uso de un cable cruzado (aunque muchos ordenadores actuales, incluyendo [los últimos modelos Mac](#) pueden hacer el cruce de forma interna).

#### Configuración de la Red Network

Al shield se debe asignar una dirección MAC y una IP fija utilizando la función [Ethernet.begin\(\)](#). Una dirección MAC es un identificador global único para cada dispositivo en particular; asignar una al azar suele funcionar, pero no utilice la misma para mas de una placa.

Una dirección IP válida depende de la configuración de su red. Es posible usar DHCP para asignar una IP dinámica a la placa, pero esto aun no está implementado como función. Opcionalmente se pueden especificar la dirección de difusión y la máscara de subred.

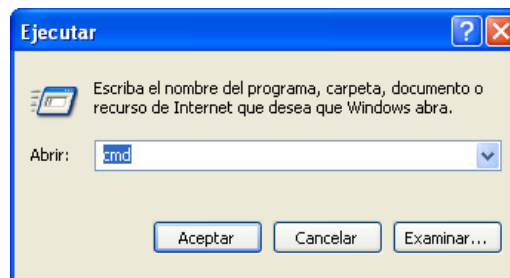
## 4. Conectar con una página Web sabiendo su IP

Con este ejemplo vamos a conectarnos a una página Web de la que previamente conoceremos su IP y recibiremos en nuestro nodo Ethernet Arduino la información que nos envíe esta página. Lo haremos haciendo uso del “monitor” del puerto serie que tiene el IDE Arduino.

Antes de nada diremos como averiguar la IP de una pagina Web.

### ¿Cómo AVERIGURA LA IP DE UNA PAGINA WEB?

Invocamos la consola de Windows “ejecutar” y escribimos **cmd**



Después escribimos “**ping** (la dirección web)”

Para averiguar, por ejemplo, la dirección de Facebook lo hacemos como sigue.

Escribimos “*ping www.facebook.com*”

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Usuario>ping www.facebook.com

Haciendo ping a www.facebook.com [69.171.247.37] con 32 bytes de datos:

Respuesta desde 69.171.247.37: bytes=32 tiempo=245ms TTL=239
Respuesta desde 69.171.247.37: bytes=32 tiempo=247ms TTL=239
Respuesta desde 69.171.247.37: bytes=32 tiempo=247ms TTL=239
Respuesta desde 69.171.247.37: bytes=32 tiempo=249ms TTL=239

Estadísticas de ping para 69.171.247.37:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 245ms, Máximo = 249ms, Media = 247ms

C:\Documents and Settings\Usuario>
    
```

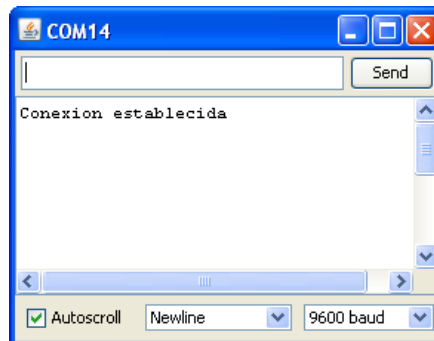
Y vemos que la IP es [69.171.247.37]

Colocaremos esa IP en el programa que se lista a continuación

***byte server[] = {69,171,242,74}; // Direccion IP de [www.facebook.com](http://www.facebook.com)***

Descargamos el programa en la tarjeta Arduino mediante el IDE de Arduino como siempre hacemos.

Abrimos la ventana de Visualización del puerto de comunicación de Arduino y vemos



A partir de ese momento comenzará a visualizarse la información en formato texto que nos llega a través de nuestro nodo Ethernet.

### Este es el programa

/\*

Web client

En este sketch Arduino se conecta a una página Web (<http://www.google.com>) usando un Arduino Wiznet Ethernet shield.

Circuit:

\* Ethernet shield ocupa los pines 10, 11, 12, 13

created 18 Dec 2009

modified 9 Apr 2012

by David A. Mellis

traducido J.M. Ruiz Nov. 2012

\*/

**#include <SPI.h>**

**#include <Ethernet.h>**

// Pone la dirección MAC en el controlador.

// Los Shields Ethernet más actuales Ethernet tienen una dirección MAC impresa en una etiqueta pegada en ellos

**byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };**

**IPAddress server(74,125,230,223); // Google**



```
// Inicializar la biblioteca de cliente Ethernet
// Con la dirección IP y el puerto del servidor
// Que desea conectarse (por defecto es el puerto 80 para HTTP):
EthernetClient client;

void setup() {
  // Abre el puerto serie para la comunicación:
  Serial.begin(9600);
  while (!Serial) {
    ; // esperar para para conectar. Necesario para Leonardo sólo
  }

  // inicia la conexión con Ethernet:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Falla la conexión configurada usando DHCP");
    // No tiene sentido seguir adelante, así que no hace nada:
    for(;;)
      ;
  }
  // intenta una segunda inicialización:
  delay(1000);
  Serial.println("conectando...");

  // si tienes una conexión, informar a través del puerto serie:
  if (client.connect(server, 80)) {
    Serial.println("conectado");
    // Make a HTTP request:
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  }
  else {
    // kf no recibió una conexión con el servidor:
    Serial.println("conexion fallida");
  }
}

void loop()
{
  // Si hay bytes disponibles entrantes
  // Del servidor los lee e imprime:
  if (client.available()) {
    char c = client.read();

```

```

    Serial.print(c);
}

// Si el servidor está desconectado, se detiene el cliente:
if (!client.connected()) {
    Serial.println();
    Serial.println("disconnectar.");
    client.stop();

    // no hace nada:
    for(;;)
        ;
}
}

```

## 5. Probando la aplicación Ethernet Shield con el Hyperterminal de Windows.

En esta aplicación vamos a utilizar el shield Ethernet como receptor de datos que le mandaremos desde un PC haciendo uso de Hyperterminal.

```
/*
```

```
Telnet client
```

Este sketch conecta a un servidor telnet (<http://www.google.com>)

Usando un shield Arduino Ethernet. Necesitaremos un servidor telnet para probarlo

El ejemplo de Processing ChatServer example (incluido en la librería de red) trabaja muy bien en el puerto 10002. Este puede encontrarse en la parte de ejemplos en la aplicación Processing disponible en <http://processing.org/>

Circuito:

\* Ethernet shield conectado a los pines 10, 11, 12, 13

creado 14 Sep 2010

modificado Apr 2012

por Tom Igoe

traducido por J.M. Ruiz diciembre 2012

```
*/
```

```
#include <SPI.h>
```

```
#include <Ethernet.h>
```

```
// Colocar la dirección MAC y la IP para nuestro ejemplo.
```

```
// La dirección IP dependerá de nuestra red local:
```

```
byte mac[] = {
```

```
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```
IPAddress ip(192,168,1,177);
```

```
// Colocar la dirección IP del servidor conectado a la red:
```

```
IPAddress server(1,1,1,1);
```

```
// Inicializar la librería Ethernet client
```

```
// con la IP y el Puerto del servidor
```

```
// que nos conectamos (Puerto 23 por defecto para telnet);
```

```
// si usamos el ChatServer de Processing's usaremos el puerto 10002):
```

```
EthernetClient client;
```

```
void setup() {
```

```
  // Inicia la conexión Ethernet:
```

```
  Ethernet.begin(mac, ip);
```

```
  // Abre la comunicación serie y espera a que se abra el puerto serie de comunicaciones:
```

```
Serial.begin(9600);
while (!Serial) {
  ; // wait for serial port to connect. Needed for Leonardo only
}

// se da un tiempo de un segundo para inicializar el shield Ethernet:
delay(1000);
Serial.println("connecting...");

// si se efectua la conexión se realice el reporte de esta via serial:
if (client.connect(server, 10002)) {
  Serial.println("connected");
}
else {
  // si no se efectúa la conexión del servidor:
  Serial.println("connection failed");
}
}

void loop()
{
  // si llegan bytes al servidor
  // estos se leen y se imprimen:
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  // as long as there are bytes in the serial queue,
  // read them and send them out the socket if it's open:
  while (Serial.available() > 0) {
    char inChar = Serial.read();
    if (client.connected()) {
      client.print(inChar);
    }
  }

  // si el servidor se desconecta se detiene el cliente:
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
    // do nothing:
    while(true);
  }
}
```

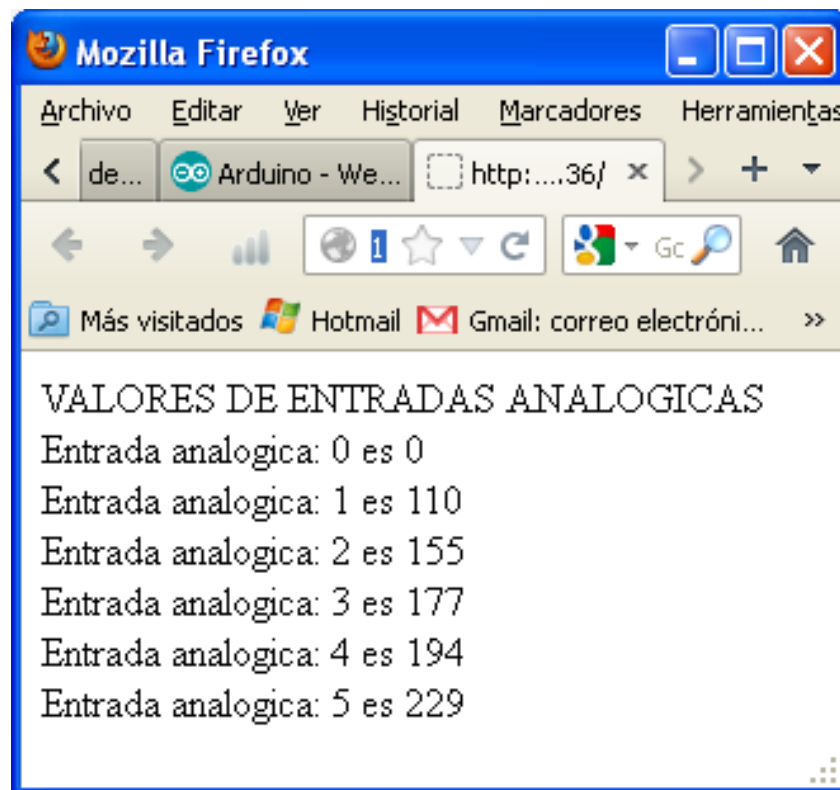
## 6. Arduino como servidor en una red Ethernet: Lectura de 6 canales analógicos de la tarjeta Arduino UNO.

Se trata de que en la misma IP a la que está conectado Arduino Ethernet se escriban los valores de los canales analógicos leídos en Arduino

*byte ip[] = {192,168,1,37};*

aquí hay que colocar la IP de nuestro puerto de Arduino.

Ahora que ya comprendemos mejor de que se trata el protocolo HTTP veamos un código que transforma nuestro Arduino en un servidor web.



/\*

Servidor Web

Un servidor web simple que muestra el valor de los pines de entrada analógica. utilizando un Arduino Ethernet Shield Wiznet.

Circuitos:

- \* Ethernet shield conectado a los pines 10, 11, 12, 13
- \* Entradas analogicas en los pines A0 a A5

created 18 Dec 2009

by David A. Mellis

modified 9 Apr 2012

by Tom Igoe

Traducido J.M. Ruiz Nov. 2012

\*/

**#include <SPI.h>**

**#include <Ethernet.h>**

// Escriba una dirección MAC y la dirección IP para el controlador.

// La dirección IP será dependiente de la red local:

**byte mac[] = {  
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };**  
**IPAddress ip(192,168,1,36);**

// Inicializar la libreria de servidor Ethernet

// Con la dirección IP y el puerto que desee utilizar

// (Puerto 80 es el valor predeterminado para HTTP):

**EthernetServer server(80);**

**void setup() {**

// Abre el puerto serie de comunicacion y espera:

**Serial.begin(9600);**

**while (!Serial) {**

  ; // poner aquí un delay de espera para que se conecte el puerto. Solo necesario para

// Arduino Leonardo

**}**

// inicia la conexion y el servidor:

**Ethernet.begin(mac, ip);**

**server.begin();**

**Serial.print("server is at ");**

**Serial.println(Ethernet.localIP());**

**}**

**void loop() {**

// Detectar los clientes entrantes

**EthernetClient client = server.available();**

**if (client) {**

**Serial.println("Nuevo cliente");**

  // Una petición http termina con una línea en blanco

**boolean currentLineIsBlank = true;**

**while (client.connected()) {**

**if (client.available()) {**

**char c = client.read();**

**Serial.write(c);**

      // Si se ha llegado al final de la línea (recibirá una nueva línea

      // con un Caracter en blanco, la petición http ha terminado,

```

// Para que pueda enviar una respuesta
if (c == '\n' && currentLineIsBlank) {
  // send a standard http response header
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Conexion Cerrada: cerrada");
  client.println();
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.print("VALORES DE ENTRADAS ANALOGICAS ");
  client.println("<br />");
  // Añadir una etiqueta para conseguir que el navegador se actualice cada 5 segundos:
  client.println("<meta http-equiv=\"refresh\" content=\"5\">");
  // salida del valor de cada entrada analogica
  for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
    int sensorReading = analogRead(analogChannel);
    client.print("Entrada analogica: ");
    client.print(analogChannel);
    client.print(" es ");
    client.print(sensorReading);
    client.println("<br />");
  }
  client.println("</html>");
  break;
}
if (c == '\n') {
  // se inicia una nueva linea
  currentLineIsBlank = true;
}
else if (c != '\r') {
  // ha llegado a un caracter en la línea actual
  currentLineIsBlank = false;
}
}
}
// facilita al navegador web un tiempo para recibir los datos
delay(1);
// cierra la conexión:
client.stop();
Serial.println("Cliente desconectado");
}
}

```

### Prueba del programa.

Para probar el programa lo que hacemos es en primer lugar descargarlo sobre la tarjeta Arduino y a continuación conectamos el conector de red a la conexión JP45 de la tarjeta Shield Arduino Ethernet.

Bastará que desde cualquier lugar de la red escribamos la dirección de nuestro servidor en la ventana de direcciones del navegador para que aparezcan escritos los valores números de los canales analógicos. Para actualizar el valor basta que demos al botón de recargar pagina y se refrescaran los datos.

Las pruebas las realice dentro de una red LAN pero teniendo los correspondientes permisos de acceso podemos hacerlo desde cualquier lugar de Internet, siempre que escribamos correctamente la IP de nuestro nodo Ethernet remoto.



## 7. Gobierno de un Relé con el Shield Arduino Ethernet

(Basado en el trabajo publicado en: <http://www.diarioelectronicohoy.com/arduino/arduino-ethernet-shield-relay> )

En este ejemplo se trata de encender y apagar una bombilla a través de Internet y, para ello, vamos a utilizar el Shield Ethernet Arduino. Serán necesarios unos conocimientos básicos de HTML, simplemente para poder hacer la página Web a nuestro gusto. A través de esta página Web, podremos encender y apagar nuestra bombilla cuando queramos. Desde el ordenador, iPad, tablet, o cualquier dispositivo con conexión WI-FI.



Todos los dispositivos que estén conectados a la misma red a la que esté conectado Arduino tendrán la posibilidad de acceder a la página que implementemos en el nodo Ethernet de Arduino, es decir, quien esté en la red de casa, de la oficina, del edificio, etc.

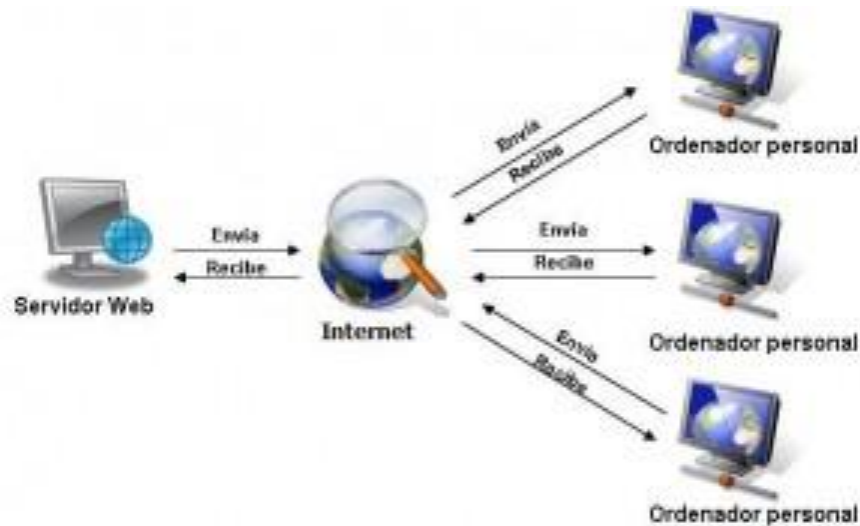
Se pone una bombilla como podríamos poner otra carga cualquiera e incluso varias.

Lo que vamos a crear con el Ethernet Shield, es un servidor Web, el cual nos proporcionará el código HTML para poder verlo en nuestro navegador y poder, así, interactuar con él.

Veamos brevemente que es un servidor Web.

En Internet, un servidor es un ordenador remoto que provee los datos solicitados por parte de los navegadores de otros ordenadores. En redes locales (LAN, Local Area Network), se entiende como el software que configura un PC como servidor para facilitar el acceso a la red y sus recursos. Los servidores almacenan información en forma de páginas Web y, a través del protocolo HTTP lo entregan a petición de los clientes (navegadores Web) en formato HTML.

Un servidor sirve información a los ordenadores que se conectan a él. Cuando los usuarios se conectan a un servidor, pueden acceder a programas, archivos y otra información del servidor. En la Web, un servidor es un ordenador que usa el protocolo HTTP para enviar páginas Web al equipo de un usuario cuando éste las solicita.



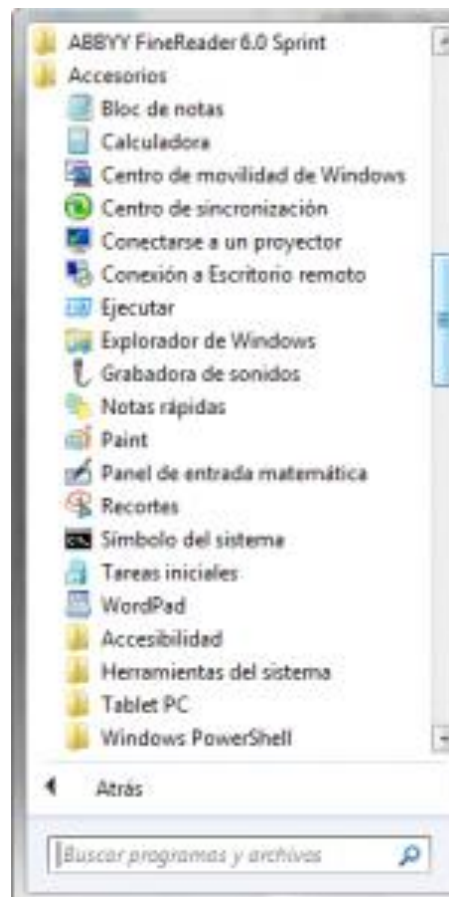
En este caso no vamos a utilizar la red Internet sino que lo haremos a nivel local en lo que se denomina una red LAN (Red de Area Local). Para acceder a la página Web que creemos, deberemos acceder con una dirección IP perteneciente a nuestra red, que será la que le hayamos asignado al Ethernet Shield. En el código que pondremos después veremos que la dirección IP que le he asignado es 192.168.0.100/24.

¿Qué quiere decir /24?

Es la máscara de red. Al ser /24, quiere decir que tendremos 24 bits a '1' y 8 a '0', es decir, que será 255.255.255.0.

Algunos ya sabréis qué dirección IP asignar al Ethernet Shield, pero, para los que no lo sepan, a continuación podréis ver cómo saber la dirección.

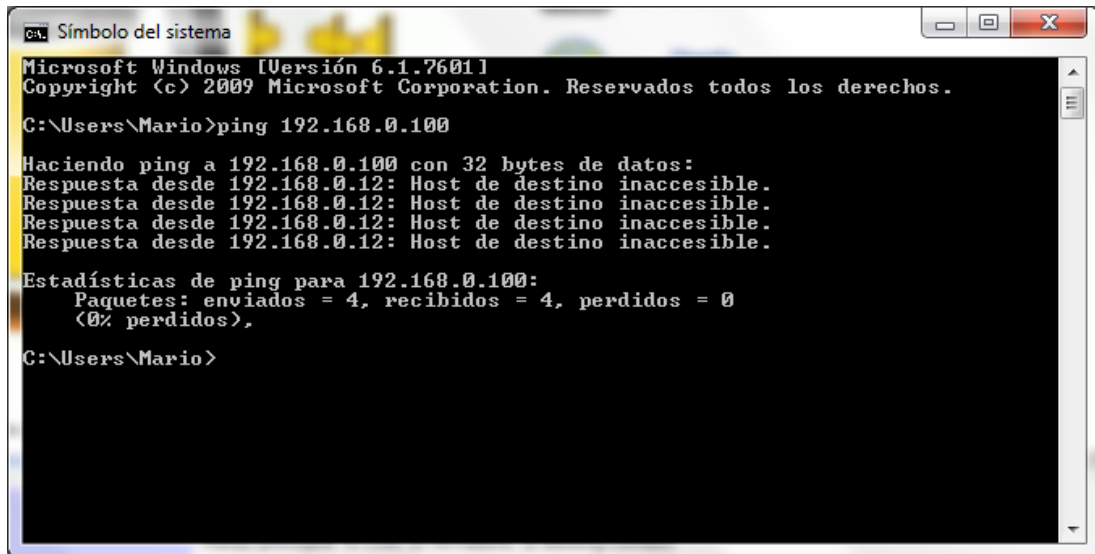
Para comenzar, iremos al programa **EJECUTAR**. Para ello, simplemente teclea “ejecutar” en la búsqueda de programas, o puedes ir a la carpeta accesorios para encontrarlo. Si quieres ser lo más rápido posible, puede pulsar la tecla *WINDOWS*+*R*. A continuación, escribiremos “cmd”.



A continuación, escribiremos “ipconfig” y, podremos ver en qué subred estamos. Mi puerta de enlace predeterminada es 192.168.0.1, así que, le asignaré al Ethernet Shield la dirección 192.168.0.100/24.

```
Dirección IPv4. . . . . : 192.168.0.12
Máscara de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . . : 192.168.0.1
```

Pero, antes de asignarle esta dirección, hay que comprobar que no hay ningún otro equipo con esa misma IP, por lo que realizaremos un PING a la dirección que queramos asignar. Si no obtenemos respuesta, perfecto.



```

C:\Users\Mario>ping 192.168.0.100

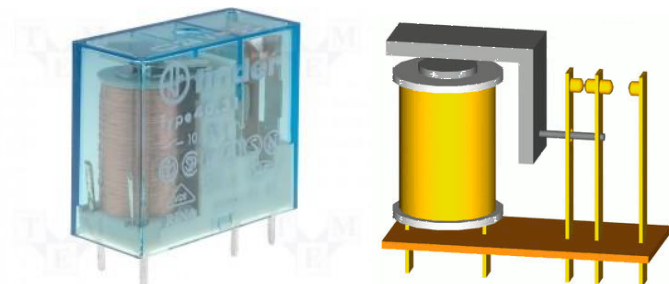
Haciendo ping a 192.168.0.100 con 32 bytes de datos:
Respuesta desde 192.168.0.12: Host de destino inaccesible.
Respuesta desde 192.168.0.12: Host de destino inaccesible.
Respuesta desde 192.168.0.12: Host de destino inaccesible.
Respuesta desde 192.168.0.12: Host de destino inaccesible.

Estadísticas de ping para 192.168.0.100:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
              (0% perdidos),

C:\Users\Mario>
    
```

Host de destino inaccesible, por lo que podremos asignar esa IP.

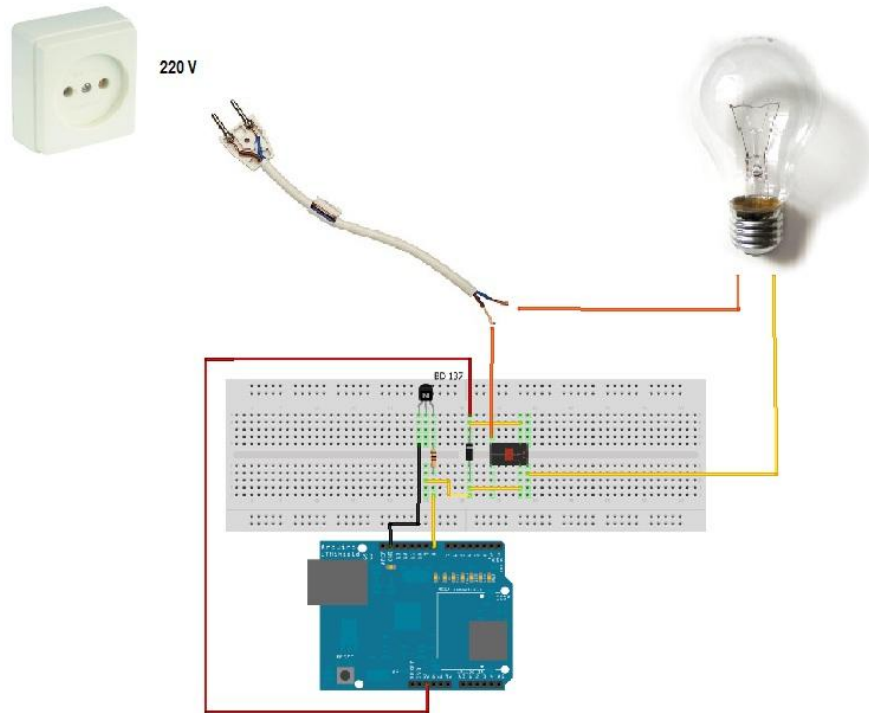
Por otra parte, el montaje del proyecto es muy sencillo, ya que sólo se necesita un diodo 1N4001, un transistor BD137, una resistencia de 1K Ohm y un relé. El relé que he utilizado es el [FINDER 5V DC, 10A, 250V~](#).



Un relé es un interruptor controlado por un electroimán. La conexión o desconexión entre sus terminales no será realizada por un usuario, sino que un electroimán será el encargado de mover las piezas necesarias para que el interruptor cambie de posición.

Como ya he mencionado antes, con este relay, seremos capaces, por ejemplo, de encender una lámpara, entre otras muchas cosas, como es el caso de este tutorial.

## MONTAJE



## Código

```

/*
Mario Pérez Esteso
http://www.diarioelectronicohoy.com/arduino
https://www.facebook.com/TutorialesArduino
http://www.twitter.com/_Mario_Perez
*/
#include <SPI.h>
#include <Ethernet.h>
//Declaración de la direcciones MAC e IP. También del puerto 80
byte mac[]={0xDE,0xAD,0xBE,0xEF,0xFE,0xED}; //MAC
IPAddress ip(192,168,1,36); //IP
EthernetServer servidor(80);
int PIN_LED=8;
String readString=String(30);
String state=String(3);

void setup()
{
  Ethernet.begin(mac, ip); //Inicializamos con las direcciones asignadas
  servidor.begin();
  pinMode(PIN_LED,OUTPUT);

```

```

digitalWrite(PIN_LED,HIGH);
state="OFF";
}
void loop()
{
  EthernetClient cliente= servidor.available();
  if(cliente)
  {
    boolean lineaenblanco=true;
    while(cliente.connected())      //Cliente conectado
    {
      if(cliente.available())
      {
        char c=cliente.read();
        if(readString.length()<30)    //Leemos petición HTTP caracter a caracter
        {
          readString.concat(c);      //Almacenar los caracteres en la variable readString
        }
        if(c=='\n' && lineaenblanco) //Si la petición HTTP ha finalizado
        {
          int LED = readString.indexOf("LED=");
          if(readString.substring(LED,LED+5)=="LED=T")
          {
            digitalWrite(PIN_LED,LOW);
            state="ON";
          } else if (readString.substring(LED,LED+5)=="LED=F")
          {
            digitalWrite(PIN_LED,HIGH);
            state="OFF";
          }
        }

        //Cabecera HTTP estándar
        cliente.println("HTTP/1.1 200 OK");
        cliente.println("Content-Type: text/html");
        cliente.println();
        //Página Web en HTML
        cliente.println("<html>");
        cliente.println("<head>");
        cliente.println("<title>LAMPARA ON/OFF</title>");
        cliente.println("</head>");
        cliente.println("<body width=100% height=100%>");
        cliente.println("<center>");
        cliente.println("<h1>LAMPARA ON/OFF</h1>");
      }
    }
  }
}

```

```
    cliente.print("<br><br>");
    cliente.print("Estado de la lampara: ");
    cliente.print(state);
    cliente.print("<br><br><br><br>");
    cliente.println("<input type=submit value=ON style=width:200px;height:75px
onClick=location.href='./?LED=T\'>");
    cliente.println("<input type=submit value=OFF style=width:200px;height:75px
onClick=location.href='./?LED=F\'>");
    cliente.println("</center>");
    cliente.println("</body>");
    cliente.println("</html>");
    cliente.stop();//Cierro conexión con el cliente
    readString="";
  }
}
}
}
```

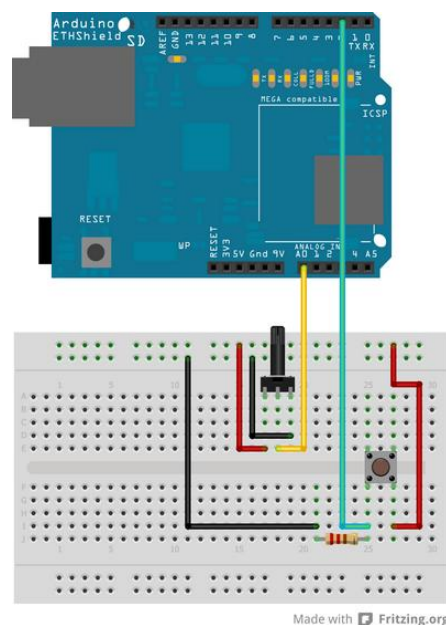
## 8. Leer una señal digital y una analógica.

El presente programa permite leer un dato digital y uno analógico en la tarjeta Arduino que está unida al Shield Ethernet.

*Entrada Digital*      *PIN2*

*Entrada Analógica*    *A0*

Para realizar las pruebas se colocara un pulsador en la entrada 2 y un potenciómetro en el canal analógico AN0 tal como se muestra en la siguiente figura.



Listado del código.

```

/*
-----
Servidor Web
-----
Muestra el valor de una entrada digital y una entrada análoga a través de una página web
Entrada digital -> Pulsador NO
Entrada análoga -> Sensor Ultrasonico LV-MaxSonar-EZ1
Cosas de Mecatrónica y Tienda de Robótica
*/
#include <Ethernet.h> // Incluye la librería Ethernet
#include <SPI.h>

// Identificador Ethernet único
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 1, 36 }; // Dirección IP asignada al Arduino Ethernet Shield
EthernetServer server(80); // Es el puerto HTML para conexión a Internet
    
```



```
//Función principal
```

```
void setup()
```

```
{
  Ethernet.begin(mac, ip); //Inicializa librería y configuraciones de red
  server.begin(); //Inicia comunicación a través del puerto
}
```

```
//Función cíclica
```

```
void loop()
```

```
{
  EthernetClient client = server.available();
  if (client) { //Una petición http termina con una línea en blanco
    boolean current_line_is_blank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        // Si hemos llegado al final de la línea (recibió una nueva línea
        // Carácter) y la línea está en blanco, la petición http ha terminado,
        // Para que podamos enviarle una respuesta

        if (c == '\n' && current_line_is_blank)
        {
          client.println("HTTP/1.1 200 OK"); // Envió encabezado estándar de respuesta HTTP
          client.println("Content-Type: text/html");
          client.println();
          client.print("Entrada digital "); //Imprimir valor entrada digital
          client.print("2");
          client.print(" es ");
          client.print(digitalRead(2)); // Lectura del pin 2 (digital)
          client.println("<br />");
          client.println("<br />");
          client.print("Entrada analoga"); //Imprimir valor entrada analoga
          client.print("0");
          client.print(" es ");
          client.print(analogRead(0)); // Lectura del pin 0 (analogo)
          client.println("<br />");
          client.println("<br />");
          break;
        }

        if (c == '\n')
        {
          current_line_is_blank = true; // Comenzaremos una nueva línea
        }
        else if (c != '\r')
        {
```

```
    current_line_is_blank = false; // Obtenemos un caracter en la línea actual
  }
}
// Damos un tiempo al servidor web para recibir los datos
delay(1); //Retardo de un 1 ms(milisegundo)
client.stop();
}
}
```

## ANEXO I

# Librería Ethernet

( Fuente: <http://arduino.cc/es/Reference/Ethernet> )

Junto con el shield Ethernet de Arduino Ethernet Shield, esta librería permite a la placa Arduino de conectarse a Internet. Puede funcionar tanto como servidor capaz de aceptar conexiones entrantes, como cliente permitiendo realizar conexiones de salida. La librería permite hasta cuatro conexiones simultáneas (entrantes, salientes, o una combinación de ambas).

### Clases de Ethernet

Las clases de Ethernet inicializan la librería Ethernet y la configuración de la red.

- [begin\(\)](#)

### Clases del modo servidor

Las clases del modo Servidor crean servidores que permiten enviar y recibir datos desde los clientes conectados(programas funcionando en otros ordenadores o dispositivos).

- [Server\(\)](#)
- [begin\(\)](#)
- [available\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [println\(\)](#)

### Clases del modo Cliente

Las clases del modo Cliente crean clientes que pueden conectarse con servidores y enviar datos a los mismos, o recibir datos de ellos.

- [Client\(\)](#)
- [connected\(\)](#)
- [connect\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [println\(\)](#)
- [available\(\)](#)
- [read\(\)](#)
- [flush\(\)](#)
- [stop\(\)](#)

## Ethernet.begin()

**Descripción:** Inicializa la librería Ethernet y la configuración de la red.

### Sintaxis

```
Ethernet.begin(mac, ip);
Ethernet.begin(mac, ip, gateway);
Ethernet.begin(mac, ip, gateway, subnet);
```

### Parámetros

mac: la dirección MAC del dispositivo (una secuencia -array- de 6 bytes)

ip: la dirección IP del dispositivo (una secuencia -array- de 4 bytes)

gateway: la dirección IP de la puerta de enlace de la red (una secuencia -array- de 4 bytes).

opcional: usar por defecto la dirección Ip del dispositivo con el último octeto establecido como 1

subnet: la máscara de subred de la red (una secuencia -array- de 4 bytes). opcional: usar por defecto 255.255.255.0

### Respuesta

Ninguna

### Ejemplo

```
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };

void setup()
{
  Ethernet.begin(mac, ip);
}

void loop () {}
```

## Server()

### Descripción

Crea un servidor que recibe conexiones entrantes en el puerto especificado.

### Sintaxis

```
Server(puerto);
```

## Parámetros

puerto: el puerto en el cual escuchar (int)

## Devuelve

Nada

## Ejemplo

```
#include <Ethernet.h>

// configuración de red, el gateway (puerta de enlace) y la subnet son opcionales.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte gateway[] = { 10, 0, 0, 1 };
byte subnet[] = { 255, 255, 0, 0 };

// telnet utiliza por defecto en el puerto 23
Server server = Server(23);

void setup()
{
  // inicializa el dispositivo ethernet
  Ethernet.begin(mac, ip, gateway, subnet);

  // comienza a recibir conexiones
  server.begin();
}

void loop()
{
  Client client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

## begin()

## Descripción

Indica al servidor que comience a recibir conexiones entrantes.

## Sintaxis

*server*.begin()

## Parámetros

Ninguno

## Devuelve

Nada

## Ejemplo

```
#include <Ethernet.h>

// configuración de red, el gateway (puerta de enlace) y la subnet son opcionales.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte gateway[] = { 10, 0, 0, 1 };
byte subnet[] = { 255, 255, 0, 0 };

// telnet utiliza por defecto en el puerto 23
Server server = Server(23);

void setup()
{
  // inicializa el dispositivo ethernet
  Ethernet.begin(mac, ip, gateway, subnet);

  // comienza a recibir conexiones
  server.begin();
}

void loop()
{
  Client client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

## available()

### Descripción

Obtiene un cliente que esté conectado al servidor y posea datos disponibles para lectura. Esta conexión se mantiene activa cuando el objeto-cliente obtenido salga del ámbito de la función; para cerrarla se debe realizar "client".stop().

### Sintaxis

*server*.available()

### Parámetros

Ninguno

## Devuelve

Nada

## Ejemplo

```
#include <Ethernet.h>

// configuración de red, el gateway (puerta de enlace) y la subnet son opcionales.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte gateway[] = { 10, 0, 0, 1 };
byte subnet[] = { 255, 255, 0, 0 };

// telnet utiliza por defecto en el puerto 23
Server server = Server(23);

void setup()
{
  // inicializa el dispositivo ethernet
  Ethernet.begin(mac, ip, gateway, subnet);

  // comienza a recibir conexiones
  server.begin();
}

void loop()
{
  Client client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

## write()

### Descripción

Escribe datos a todos los clientes conectados al servidor.

### Sintaxis

*servidor*.write(datos)

### Parámetros

datos: el valor a escribir (byte o char)

## Devuelve

Nada

## Ejemplo

```
#include <Ethernet.h>

// configuración de red, el gateway (puerta de enlace) y la subnet son opcionales.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte gateway[] = { 10, 0, 0, 1 };
byte subnet[] = { 255, 255, 0, 0 };

// telnet utiliza por defecto en el puerto 23
Server server = Server(23);

void setup()
{
  // inicializa el dispositivo ethernet
  Ethernet.begin(mac, ip, gateway, subnet);

  // comienza a recibir conexiones
  server.begin();
}

void loop()
{
  Client client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

## print()

### Descripción

Imprime datos a todos los clientes conectados al servidor. Imprime numeros como secuencia de digitos, cada uno como carácter ASCII (ej: el numero 123 es enviado como los tres caracteres '1', '2', '3').

Print data to all the clients connected to a server. Prints numbers as a sequence of digits, each an ASCII character (e.g. the number 123 is sent as the three characters '1', '2', '3').

### Sintaxis

```
servidor.print(datos)
servidor.print(datos, BASE)
```

### Parámetros

datos (opcional): los datos a imprimir (char, byte, int, long, o string)

BASE (opcional): la base en la que se imprimen los numeros: BIN para binarios (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).



## println()

### Descripción

Imprime datos, seguido por una línea nueva (newline), a todos los clientes conectados al servidor. Imprime números como secuencia de dígitos, cada uno como carácter ASCII (ej: el número 123 es enviado como los tres caracteres '1', '2', '3').

### Sintaxis

```
servidor.println()  
servidor.println(datos)  
servidor.println(datos, BASE)
```

### Parámetros

datos (opcional): los datos a imprimir (char, byte, int, long, o string)

BASE (opcional): la base en la que se imprimen los números: BIN para binarios (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

# Clases del modo Cliente

Las clases del modo Cliente crean clientes que pueden conectarse con servidores y enviar datos a los mismos, o recibir datos de ellos.

- [Client\(\)](#)
- [connected\(\)](#)
- [connect\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [println\(\)](#)
- [available\(\)](#)
- [read\(\)](#)
- [flush\(\)](#)
- [stop\(\)](#)

## Client()

### Descripción

Crea un cliente con el que poder conectar a una determinada IP y puerto.

### Sintaxis

Client(ip, port)

### Parámetros

ip: la IP a la que el cliente se conectará (cadena de 4 Bytes)

port: Número entero que indica el puerto al que el cliente se conectará.

### Ejemplo

```
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte server[] = { 64, 233, 187, 99 }; // Google

Client client(server, 80);

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);

  delay(1000);

  Serial.println("conectando...");
```

```

if (client.connect()) {
  Serial.println("conectado");
  client.println("GET /search?q=arduino HTTP/1.0");
  client.println();
} else {
  Serial.println("fallo de conexión");
}
}

void loop()
{
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  if (!client.connected()) {
    Serial.println();
    Serial.println("disconectando.");
    client.stop();
    for(;;)
      ;
  }
}

```

## connected()

### Descripción

Si el cliente está o no conectado. Recuerda que un cliente se considera como conectado, aunque la conexión se haya cerrado, si quedan datos sin leer.

### Sintaxis

*client.connected()*

### Parámetros

ninguno

### Retorna

Retorna cierto(true) si el cliente está conectado, falso(false) si no.

### Ejemplo

```

#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte server[] = { 64, 233, 187, 99 }; // Google

```

```
Client client(server, 80);

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);

  delay(1000);

  Serial.println("conectando...");

  if (client.connect()) {
    Serial.println("conectado");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  } else {
    Serial.println("fallo de conexión");
  }
}

void loop()
{
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  if (!client.connected()) {
    Serial.println();
    Serial.println("desconectando.");
    client.stop();
    for(;;)
      ;
  }
}
```

## connect()

### Descripción

Conecta a la IP y al puerto indicado en el constructor. El valor retornado, indica si la conexión falló o fue satisfactoria.

### Sintaxis

*client*.connect()

### Parámetros

none

## Retorna

Retorna cierto (true) si se estableció la conexión, falso (false) si no.

## Ejemplo

```
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte server[] = { 64, 233, 187, 99 }; // Google

Client client(server, 80);

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);

  delay(1000);

  Serial.println("conectando...");

  if (client.connect()) {
    Serial.println("conectado");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  } else {
    Serial.println("falló la conexión");
  }
}

void loop()
{
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  if (!client.connected()) {
    Serial.println();
    Serial.println("desconectando.");
    client.stop();
    for(;;)
      ;
  }
}
```

## write()

### Descripción

Escribe datos al servidor, por parte del cliente conectado.

## Sintaxis

*client.write(data)*

## Parámetros

data: El Byte o caracter a escribir.

## print()

### Descripción

Imprime datos al servidor, por parte del cliente conectado. Imprime números como una secuencia de dígitos ,cada dígito será un caracter ASCII (ej. el número 123 se envía como tres dígitos '1', '2', '3').

## Sintaxis

*client.print(data)*  
*client.print(data, BASE)*

## Parámetros

data: los datos a imprimir (char, byte, int, long, or string)

BASE (opcional): La base en la que se imprimirá el número: BIN para binario (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

## print()

### Descripción

Imprime datos al servidor, por parte del cliente conectado. Imprime números como una secuencia de dígitos ,cada dígito será un caracter ASCII (ej. el número 123 se envía como tres dígitos '1', '2', '3').

## Sintaxis

*client.print(data)*  
*client.print(data, BASE)*

## Parámetros

data: los datos a imprimir (char, byte, int, long, or string)

BASE (opcional): La base en la que se imprimirá el número: BIN para binario (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

## available()

### Descripción

Retorna el número de Bytes disponibles para su lectura (es decir , la cantidad de datos escritos en el cliente por el servidor al que está conectado).

### Sintaxis

*client.available()*

### Parámetros

ninguno

### Retorna

El número de Bytes disponibles.

```
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 10, 0, 0, 177 };
byte server[] = { 64, 233, 187, 99 }; // Google

Client client(server, 80);

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);

  delay(1000);

  Serial.println("conectando...");

  if (client.connect()) {
    Serial.println("conectado");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  } else {
    Serial.println("conexion fallida");
  }
}

void loop()
{
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
}
```

```

    }

    if (!client.connected()) {
        Serial.println();
        Serial.println("desconectando.");
        client.stop();
        for(;;)
            ;
    }
}

```

## read()

Lee el siguiente Byte recibido por el servidor al que el cliente está conectado (después de la última llamada a read()).

### Sintaxis

*client.read()*

### Parámetros

ninguno

### Retorna

El siguiente Byte (o caracter), o -1 si no hay nada que leer.

## read()

Lee el siguiente Byte recibido por el servidor al que el cliente está conectado (después de la última llamada a read()).

### Sintaxis

*client.read()*

### Parámetros

ninguno

### Retorna

El siguiente Byte (o caracter), o -1 si no hay nada que leer.

## stop()



## Descripción

Desconecta del servidor.

## Sintaxis

*client*.stop()

## Parámetros

ninguno

## Retorna

nada

El texto de la referencia de Arduino está publicado bajo la licencia [Creative Commons Reconocimiento-Compartir bajo la misma licencia 3.0](#). Los ejemplos de código de la referencia están liberados al dominio público.

## Documentación Software y Bibliografía

### Enlaces a páginas WEB de referencia y consulta

<http://arduino.cc/en/Main/arduinoBoardUno>

<http://arduino.cc/en/Tutorial/WebServer>

<http://portforward.com/>

<http://www.techspot.com/guides/287-default-router-ip-addresses/>

<http://tronixstuff.wordpress.com/2010/09/12/moving-forward-with-arduino-chapter-16-ethernet/>

### Enlace a documentos específicos:

[arduino-ethernet-shield-06-schematic.pdf](#),

[arduino-ethernet-shield-06-reference-design.zip](#)