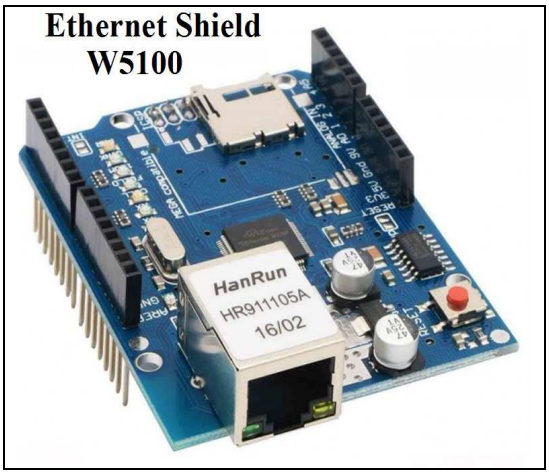
ETHERNET W5100 (Parte I)

(Ethernet Shield W5100)

En esta guía, realizaremos ejemplos sencillos, para entender el funcionamiento del Shield Ethernet, a través del cual accederemos desde nuestro Arduino a la LAN de tu casa o a Internet y desde la LAN de tu casa o desde Internet a nuestro Arduino. Pero en todos los caso, no solo dependerá de cuanto aprendamos de este dispositivo, sino de cuanto sepamos de programación, HTML y comunicación TCP. Para esto, se adicionaron algunos apéndices a esta guía, que intentan cubrir el conocimiento mínimo requerido para comenzar con este fascinante tema, y se complementan con los apéndices de la Guía en donde trabajamos con Lectora/Grabadora de Tarjetas SD (Grabación y Lectura de Archivos).
Resumiendo, la Ethernet Shield brinda la capacidad de conectar un Arduino a una red Ethernet. Es la parte física que implementa la pila de protocolos TCP/IP. Este componente abre innumerables opciones para comunicar nuestro Arduino a través de Internet o de la LAN de nuestra casa. Permitiendo integrar las redes a nuestro proyectos de Domótica, automatización, Internet (IoT), control, monitoreo remoto, etc.

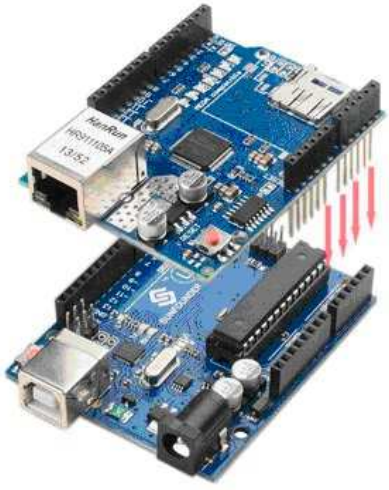


ETHERNET W5100 es compatible con los modelos Arduino "**Uno**", "**Mega**", "**Leonardo**" y según he podido leer, aunque no probado, pueden conectarse con otros modelos como "**Nano**" y "**Mini Pro**".

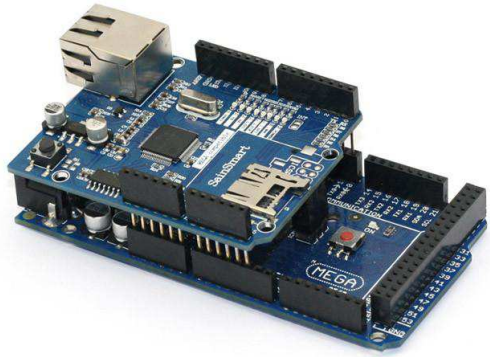
Por otro lado, las librerías usadas "**SPI**" para soporte y manejo de los pines asociados, "**Ethernet**" para la comunicación, "**SD**" para el manejo de las memorias Micro SD, ya están incluidas en el IDE de Arduino, por lo que no hay necesidad de descargarlas.

El slot para tarjetas Micro-SD, puede ser empleado para almacenar archivos que podrás poner disponibles a través de la red. O simplemente para almacenar información recolectada. Este dispositivo también incluye un controlador de reset, esto es para asegurarse que el módulo W5100 Ethernet inicie correctamente al energizarlo, y al reiniciar nuestro W5100, también se reinicia nuestra placa Arduino. Aunque rara vez es necesario usar este controlador de reset.

Una gran ventaja de este shield es que en los modelos "**Uno**", "**Mega**" es apilable; esto significa que la conexión es directa y además tendrás a disposición casi todos sus pines para usar otros dispositivos. Aunque esto nos obliga a prestar especial atención a algunos pines según el modelo de Arduino que usemos, y en los modelos como "**Leonardo**", "**Nano**" y "**Mini Pro**" deberás investigar un poco, armarte de paciencia y algunos cables, pero se puede.



W5100 Sobre un Arduino modelo UNO



W5100 Sobre un Arduino modelo Mega

El W5100 está diseñado para facilitar la implementación de conectividad a Internet sin necesidad de un SO, lo que lo hace interesante para MCU y aplicaciones de IoT.

Incluye una pila de TCP/IP por hardware y buffer interno de 16Kbytes para Tx/Rx. Esto permite liberar de estas tareas al procesador, siendo una de sus principales ventajas frente a otros controladores de Ethernet como el ENC28J60.

El W5100 puede conectarse mediante SPI, por lo que es muy sencilla la conexión con la mayoría de procesadores. Algunos modelos, como el que acá usamos, incorporan un lector de tarjeta SD, donde podemos guardar archivos (html, txt, jpg, png) con los que trabajar cuando actuemos como servidor.

Admite velocidades de 10/100 Mbits/s, soportando modos Full-Duplex y Half-Duplex con detección y corrección automática de la polaridad. Cumple con las especificaciones IEEE 802.3 10BASE-T y 802.3u 100BASE-TX. La pila TCP/IP soporta conexiones TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE, y hasta **4 conexiones simultáneas**.

Antes de comenzar con las conexiones y uso les dejo otras especificaciones técnicas a tener en cuenta:

- Voltaje de Operación: 5V DC.
- Chip Ethernet: Wiznet W5100.
- Velocidad Ethernet: 10/100 Mbps.
- Internase: SPI.
- Buffer interno de 16K. Por lo tanto No Consume Memoria de Arduino.
- Compatible con Arduino y alimentación directa con "**Uno**", "**Mega**" y "**Leonardo**".
- Lector Tarjetas Micro SD.
- Conector RJ45 estándar para Ethernet.
- Posee un botón de reset que inicializa tanto el Shield como el Arduino.

Como Conectar Ethernet Shield W5100 con Arduino

La conexión dependerá del modelo que estemos empleando. En el caso de que uses una placa Arduino "**Uno**" o "**Mega**", la conexión es inmediata, ya que solo se acoplan, para otros modelos compatibles, habrá que realizar el cableado, pero es igualmente sencillo ya que la conexión se realiza a través del SPI.

Para Conectar (encajar) el **shield** en nuestro Arduino Uno o Mega:

- *La Placa Arduino debe estar apagada - Desconectada..*
- *Los pines tienen que encajar con suavidad primero, y después presionar cuidadosamente para que se una a Arduino. Sin forzarlo.*
- *Asegúrate de que todos y cada uno de los pines encaja en los conectores de abajo.*
- Una vez que haya encajado podéis alimentar el Arduino y luego conectar el cable de red,

En esta parte de la guía nos centraremos en el uso del Ethernet Shield W5100 montado sobre los modelos "Uno" y "Mega". Y Recuerda que Todo lo visto en Guía sobre "Lectora/Grabadora Tarjetas SD (Grabación y Lectura de Archivos)" Funciona directamente en este dispositivo. (Pruébalo).

DETALLES MUY IMPORTANTES:

1) Conectar Lectora/Grabadora de SD: deberás disponer de un **Pin Digital Extra**, al que deberás Asociar este dispositivo, y este Pin debe ser uno LIBRE y disponible, de los que actualmente se encuentran compartidos por nuestro modulo Ethernet Shield W5100 y la placa Arduino (Opciones de Pines Disponibles 2, 3, 5, 6, 7, 8, 9).

2) Selección de dispositivo: Debido a que el **W5100** y la **tarjeta SD** comparten el mismo bus SPI (mientras estén acoplados con Arduino), solo uno podrá estar activo a la vez. Si debes usar ambos dispositivos simultáneamente en tu programa, debes "**desactivar**" el que NO USAS y "**Activar**" explícitamente al que SI DEBES USAR. Para desactivar la tarjeta SD, configura el pin 4 como salida y escríbele HIGH. Para desactivar el W5100, configura el pin 10 como una salida y escribe HIGH. (Puedes ver este procedimiento de Activación y desactivación en los ejemplos). **PERO....**, si solo usas uno de ellos en tu programa (**W5100** o **tarjeta SD**), úsalo directamente sin seleccionarlo, y si no usas las lectora/grabadora, no dejes la tarjeta insertada.
3) Pines Ocupados (No debes usarlos para conectar otra cosa): Mientras estén Acoplados el **Ethernet Shield W5100 y** Arduino, estarán utilizando los pines digitales 11, 12 y 13 (SPI) para comunicarse entre si. El modelo Mega reserva además los pines 50, 51, 52 y 53 para la conexión por cables. Adicionalmente en ambas tarjetas (Uno y Mega) el pin 10 es empleado para seleccionar el W5100 y el pin 4 para la tarjeta SD. **Resumiendo**, mientras emplees las funcionalidades de Ethernet **estos pines no estarán disponibles** (4, 10 11, 12, 13, 50, 51, 52 y 53).
4) Para alimentar nuestro Arduino ya conectado a **Ethernet Shield W5100,** puedes con el cable USB conectado a la PC, pero se recomienda usar una fuente externa, sobre todo si después se va a trabajar con más módulos.

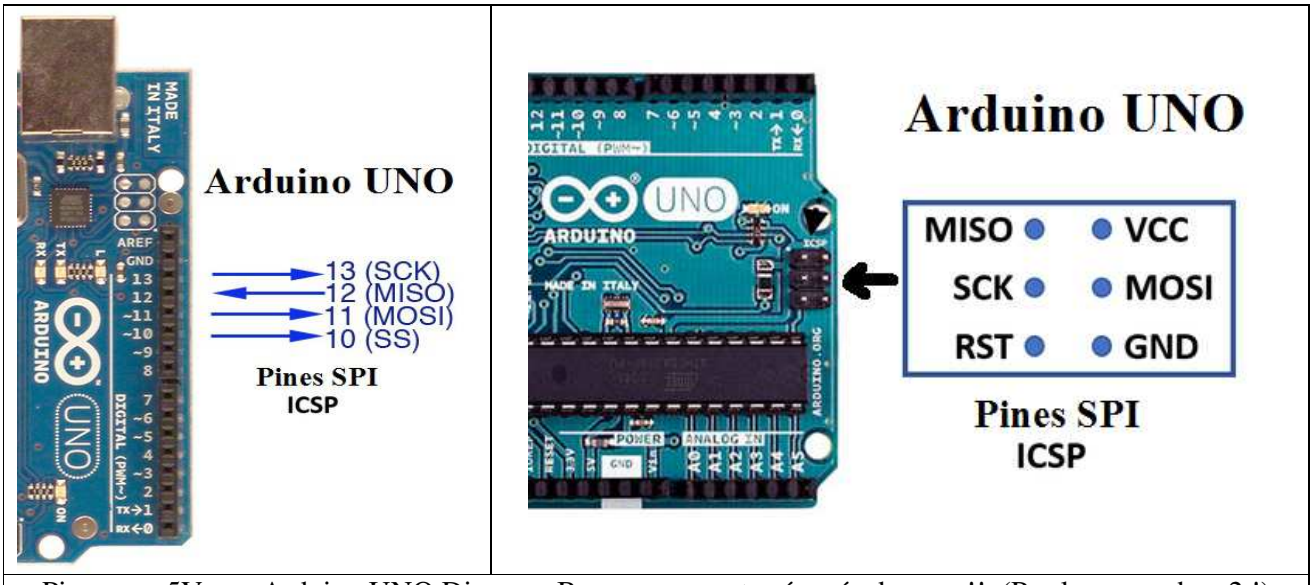
Si Usas un Modelo que no se acopla con tu Arduino o un Arduino que no permite el acople, deberás tener en cuenta el Cableado y los pines que deberán usar con Arduino.

Arduino dispone de soporte SPI por hardware vinculado físicamente a ciertos pines.
También es posible emplear cualquier otro grupo de pines como bus SPI a través de sofware, pero en ese caso la velocidad será mucho menor y deberás encontrar la librería que mejor se adapte a tu situación.

Si prefieres o debes usar cables, estos serian los pines				
MODELO	**SS**	**MOSI**	**MISO**	**SCK**
Uno	10	11	12	13
Nano	10	11	12	13
Mini Pro	10	11	12	13
Mega	53	51	50	52
Leonardo				

Los pines asociados a SPI varían de un modelo a otro. La siguiente tabla muestra la disposición en alguno de los principales modelos. Para otros modelos, consultar el esquema de Pines correspondiente (PinOut).

También deberás buscar el PinOut de tu Dispositivo para identificar que pata del modulo debes conectar al pin adecuado de Arduino. (https://www.arduino.cc/en/Reference/Ethernet)



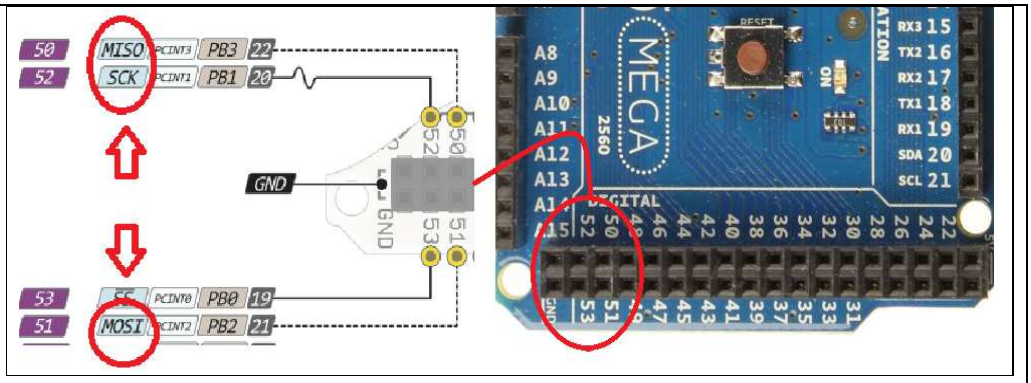
Pines con 5V que Arduino UNO Dispone. Busca, y encontrarás más de uno..!! (Por lo menos hay 2.!)

-o-

Sugiero que repases la guía correspondiente a "**Lector / Grabador de Tarjetas SD y Micro SD**" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan** acá y sin modificaciones.

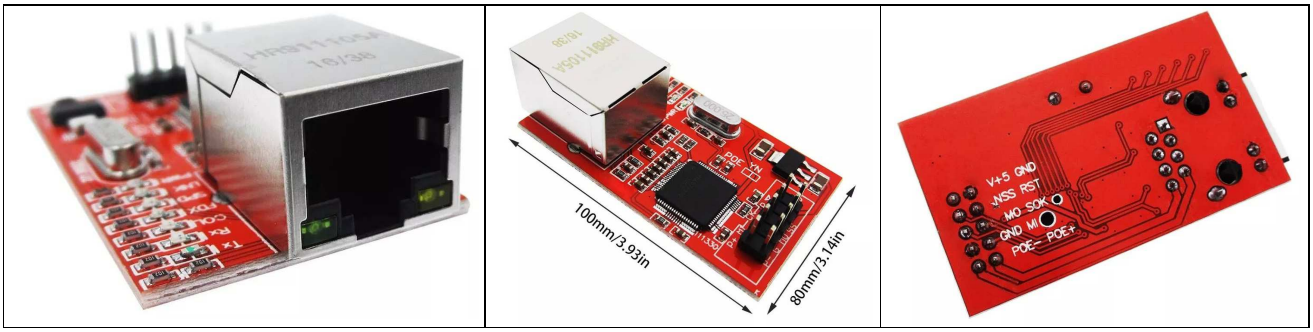
-o-

Pin 50 → MISO
Pin 51 → MOSI
Pin 52 → SCK
Pin 53 → SS
Pin GND → GND

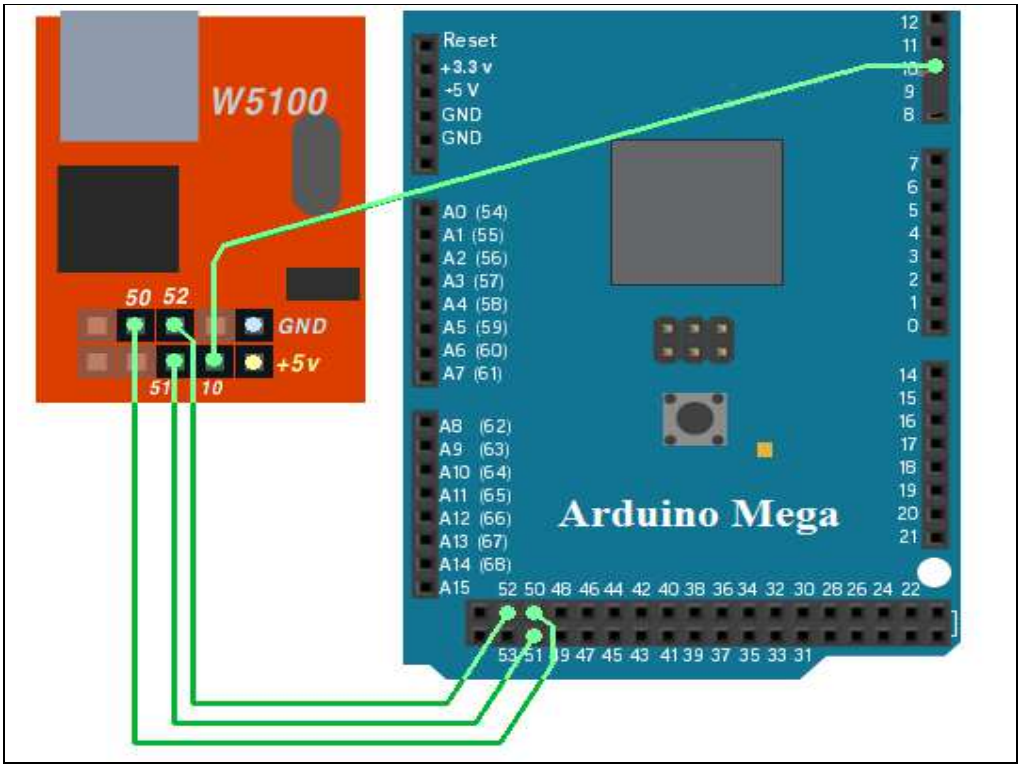


Tenemos mas de un GND y solo nos Falta el Pin de **5V**, pero podemos usar cualquiera de los Pines con 5V que Arduino MEGA Dispone. Busca, y encontrarás más de uno..!! (Por lo menos hay 4.!)

Por ejemplo, uno de los modelos más difundidos que no se acoplan directamente con Arduino es el **Ethernet w5100** con placa (tablero) rojo o azul. Este modelo, además de no poder acoplar directamente, no tiene una Lectora/Grabadora SD o Micro SD y los precios son muy similares. **Así serán las conexiones:**



-o-



QUE HAY QUE SABER ANTES DEL PRIMER PROGRAMA.

Antes de comenzar con la programación, hay algunas cosas que no vimos previamente, en ninguna de las anteriores guías, y estas son las que a continuación recomiendo interiorizarse.

a) Dirección Mac de nuestro dispositivo **Ethernet Shield W5100.** Puede consultar el **"Apéndice A - La Dirección MAC**" al Final de esta guía.

b) Dirección IP. Si desconoce el significado de IP, puede consultar "**Apéndice B - Dirección IP**", y si no sabe como obtener la dirección IP de un Sitio, puede consultar el "**Apéndice C - Ventana de Comandos o Símbolo del Sistema**".

c) Que es el Protocolo HTTP y que versiones existen. Puedes interiorizarse sobre las generalidades de este protocolo en el "**Apéndice C - Protocolo HTTP**".

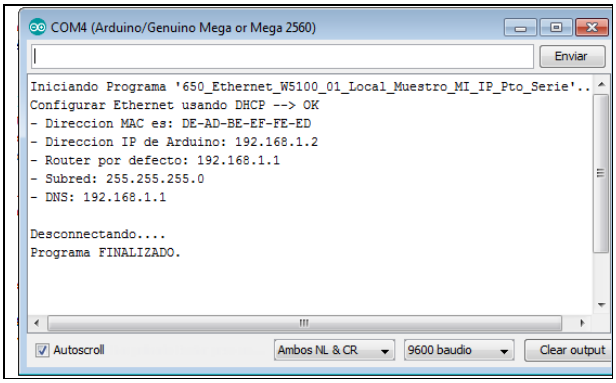
1) Mostrar en el Monitor Serie los datos correspondientes a nuestro Arduino: a) Dirección MAC. b) Dirección IP. c) Getway. d) Sub Mascara de Red. e) DNS asignado.

Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - "Apéndice C - Protocolo HTTP" - "Apéndice E - Funciones de la Librería Ethernet"**.

(Programa "650_Ethernet_W5100_01_Local_Muestro_MI_IP_Pto_Serie")

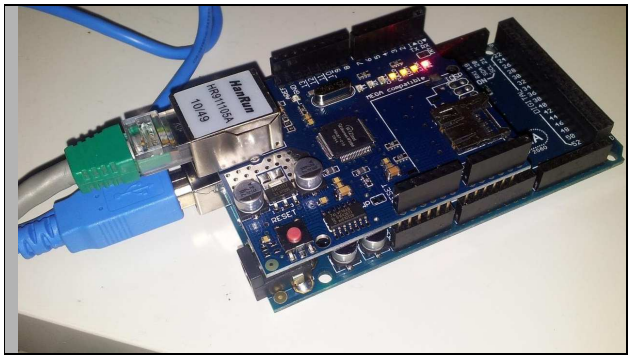
```
1   #include <SPI.h>        // Librería integrada en IDE Arduino
2   #include <Ethernet.h>  // Librería integrada en IDE Arduino
3
4   // Dirección MAC (siglas en inglés de media access control.
5   //              en español "control de acceso al medio")
6   byte mac[ ] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // ver "Apéndice A - La Dirección MAC"
7
8   EthernetClient client;
9   byte macBuffer[6],  //  Creamos un buffer para Guardar la Dirección MAC
10        B;                // Variable usada par los for( )
11
12  void setup( ){
13    Serial.begin(9600);
14    while (!Serial) {
15      ; // Esperamos que el puerto serie este abierto.
16    }
17    Serial.println("Iniciando Programa '650_Ethernet_W5100_01_Local_Muestro_MI_IP_Pto_Serie'...");
18    if (Ethernet.begin(mac) == 0){
19      Serial.println("Error al configurar Ethernet usando DHCP");
20    }else{
21      Serial.println("Configurar Ethernet usando DHCP --> OK");
22      Serial.print("- Direccion MAC es: ");
23      Ethernet.MACAddress(macBuffer);
24      for ( B = 0; B < 6; B++) {
25         Serial.print(macBuffer[B], HEX);
26         if (B < 5) Serial.print('-');
27      }
28      Serial.print("\n- Dirección IP de Arduino: ");
29      //Serial.println(Ethernet.localIP( ));  // Esta Línea Puede Reemplazar el for
30      for ( B = 0; B < 4; B++){
31        Serial.print( Ethernet.localIP( )[B], DEC);
32        if (B < 3) Serial.print(".");
33      }
34      Serial.print("\n- Router por defecto: ");
35      //Serial.println(Ethernet.gatewayIP( ));  // Esta Línea Puede Reemplazar el for
36      for( B = 0; B < 4; B++) {
37        Serial.print(Ethernet.gatewayIP( )[B], DEC);
38        if (B < 3) Serial.print(".");
39      }
40      Serial.print("\n- Subred: ");
```

```
41      //Serial.println(Ethernet.subnetMask( )); // Esta Línea Puede Reemplazar el for
42      for ( B = 0; B < 4; B++) {
43         Serial.print(Ethernet.subnetMask( )[B], DEC);
44         if (B < 3) Serial.print(".");
45      }
46      Serial.print("\n- DNS: ");
47      //Serial.println(Ethernet.dnsServerIP( )); // Esta Línea Puede Reemplazar el for
48      for ( B = 0; B < 4; B++) {
49         Serial.print(Ethernet.dnsServerIP( )[B], DEC);
50         if (B < 3) Serial.print(".");
51      }
52      Serial.print("\n");
53    }
54  }
55
56  void loop( ){
57    if (client.available( )){
58       client.stop( );
59       client.flush( );
60    }
61
62    if (!client.connected( )){
63      Serial.println("\nDesconnectando....");
64      client.stop( );
65
66      Serial.println("Programa FINALIZADO.");
67
68      while (true); // Para que no repita
69    }
70  }
```



Abrir el Monitor del Puerto Serie y ver resultados

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro modem, router o switch, 2 Metros, 1 Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

Una vez conectado y funcionando, el Shield contiene LEDs para información

ON: indica que la placa y el Shield están alimentadas.
LINK: indica la presencia de un enlace de red y parpadea cuando el Shield envía o recibe datos.
100M: indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/s).
RX: parpadea cuando el Shield recibe datos.
TX: parpadea cuando el Shield envía datos.

Verificar una Conexión Valida: Podemos comprobar que la conexión es válida. Normalmente nuestro Shield incluirá un LED bajo el conector que se encenderá con el cable conectado al Switch, y se apagará si lo sueltas. (Y lo mismo ocurrirá en el Switch, que encenderá un piloto en el frontal, cuando detecte la conexión Ethernet.

Sugiero que repases la guía correspondiente a "Lector / Grabador de Tarjetas SD y Micro SD" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan acá y sin modificaciones**.

2) Crear Una página HTM (Cliente) y esta se actualice automáticamente cada 30 segundo o manualmente cuando el usuario lo requiera.

Crear y mostrar en una página HTML local, un simple Mensaje. La página debe actualizarse automáticamente cada 30 segundos o cuando el Usuario lo requiriera.
Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - Apéndice C - Protocolo HTTP" - "Apéndice E - Funciones de la Librería Ethernet**".

Para analizar y entender este ejemplo, debemos considerar a la página como el cliente que pide información. Y a nuestro Arduino (el servidor que brinda la información). Para actualizar los datos manualmente, simplemente usamos un link que hace un llamado a la misma página (Se llama a si misma).

(Programa "650_Ethernet_W5100_03_Local_Actualizacion_Autom_y_Manual_HTML")

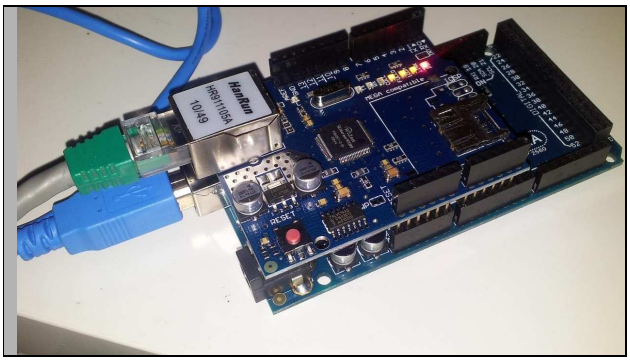
```
1  #include <SPI.h>          // Esta Librería esta incluida en el IDE de Arduino
2  #include <Ethernet.h>  // Esta Librería esta incluida en el IDE de Arduino
3
4  // Dirección MAC (siglas en inglés de media access control.
5  //              en español "control de acceso al medio")
6  // Dirección MAC - Ver Apéndice A de esta Guía.
7  byte mac[ ] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
8
9  // Ante cualquier duda puedes ver "Apéndice B - Dirección IP"
10 IPAddress ip(192,168,1,150); // Dirección IP que asignamos nuestro Arduino 192.168.1.150
11
12 // Ante cualquier duda puedes repasar "Apéndice F - PUERTOS DE COMUNICACIÓN"
13 EthernetServer server(80); // Crea servidor que escuche conexiones entrantes por el puerto
14                             //especificado. en este caso, el puerto 80
15 int Cuenta_Actual = 0;
16 void setup( ){
17   Serial.begin(9600);
18   while (!Serial); // Esperamos que el puerto serie este abierto.
19
20   Serial.println("Iniciando Programa '650_Ethernet_W5100_03_Local_Actualizacion_Autom_y_Manual_HTML...");
21
22   Ethernet.begin(mac, ip); // Conectamos a la Red local, con nuestro MAC y Nuestra IP
23   server.begin( );  // Le dice al servidor que comience a escuchar las conexiones entrantes.
24                     //(Nuestra Placa Arduino).
25
26   Serial.print("Cliente esta en Dirección IP: ");
27   Serial.println(Ethernet.localIP( )); // Retorna el IP que le hayamos Configurado
28 }
29
30 EthernetClient client;       // Variable para usar en Loop
31 bool ActualLineaVacia;    // Variable para usar en Loop
32
33 void loop( ){
34    client = server.available( ); //Devuelve el cliente que esté conectado al servidor y
35                                  //tiene datos disponibles a leer. Recordar que la conexión
36                                  // persiste cuando el objeto de cliente devuelto queda
37                                  //fuera de alcance
38    if(client){  // Pregunta si hay algún cliente conectado al Servidor
39       // Recordar que este programa, genera el Cliente (la página que lee nuestro Arduino)
40       // y Maneja el Servidor, Nuestra placa Arduino
41       Serial.println("cliente nuevo..."); // Cada vez que se inicia la página se considera
42       // una página nueva, es decir un nuevo cliente (pide datos a nuestro Arduino).
43
44       ActualLineaVacia = true;
45
46       while (client.connected( )){ // Mientras el Cliente este conectado
```

```
47          if (client.available( )){  // Pregunta Si el cliente tiene caracteres para Leer
48            char c = client.read( );   //Lee un caractér
49            Serial.write(c);           // Muestra el Caractér leído por el monitor del puerto serie
50
51            // Al recibir línea en blanco, Servir página a cliente
52            if (c == '\n' && ActualLineaVacia){
53  /****** Comienza envío de partes de página que se mostrara en la dirección de IP configurada ******/
54              client.println(F("HTTP/1.1 200 OK\nContent-Type: text/html"));
55              client.println( );
56              client.println(F("<html>\n<head>"));
57              client.println(F("<title>Castelli Horacio</title>"));  // Titulo de la Pagina
58  /************************* Actualización Automática cada 30 segundos **********************/
59              client.println(F("<meta http-equiv= refresh content=30>")); //Actualiza Página automática
60
61              client.println(F("</head>\n<body>"));
62              client.println(F("<div style='text-align:center;'>"));
63              client.println(F("<h1>Esta página es el Cliente</h1>"));
64              Cuenta_Actual++;
65              client.print(F("<h3>"));
66              client.print(Cuenta_Actual);
67              client.println(F("</h3>"));
68  /********** Acá se Arma el link para que la página se llame a si misma (para refrescar datos) *******/
69              client.print(F("<a href='http://"));
70              client.print(Ethernet.localIP( ));   //Entrega el IP que tenia la página
71              client.println(F("'><h2>Refrescar<h2></a>"));
72              client.println(F("</div>\n</body></html>"));
73              break;
74  /* Cuando en la página se hace clic sobre un link, se genera una petición de mostrar pagina, la petición se hace
75     al servidor, que casualmente es nuestra placa Arduino, y en consecuencia Muestra otra vez la página.*/
76  /*********************************************************************************************/
77            } //Final de la Pregunta si hay que construir la página (el cliente)
78            if( c == '\n'){ // '\n' es una nueva línea - Comienza nueva línea de Respuesta
79              ActualLineaVacia = true;
80            }else  if(c != '\r'){ //"\r" es un "retorno de carro" - Esta leyendo una Respuesta.
81              ActualLineaVacia = false;
82            }
83          }// Final de la pregunta si el Cliente tiene para leer
84       }// Final del While(Cliente conectado)
         delay(5); // Demoro Programa para dar tiempo al explorador
         client.stop( );
       }// Finaliza pregunta si hay un cliente conectado
     }
```

Una vez que hayas copiado el código en tu editor, hay que hacerlo funcionar, y esto es muy fácil: Abre el Monitor del puerto Serie, y te dará la IP dende se ha creado el Cliente (La página web).

Copia la Dirección IP, y pégalo en cualquier navegador, y en el acto veras la página. Y si mantienes abierto también el Monitor del Puerto Serie, podrás ver la comunicación entre página y servidor. Es interesante que comiences a entenderla.

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

ACTUALIZACIÓN DE VALORES EN LA PÁGINA

Una Actualización o Reinicio automático de Pagina, puede ser combinada con una Actualización manual. Asegurando que cada cierto tiempo los valores serán actualizados automáticamente, y la parte manual, servirá para no tener que espera que se actualice automáticamente, cuando el usuario así lo requiera.

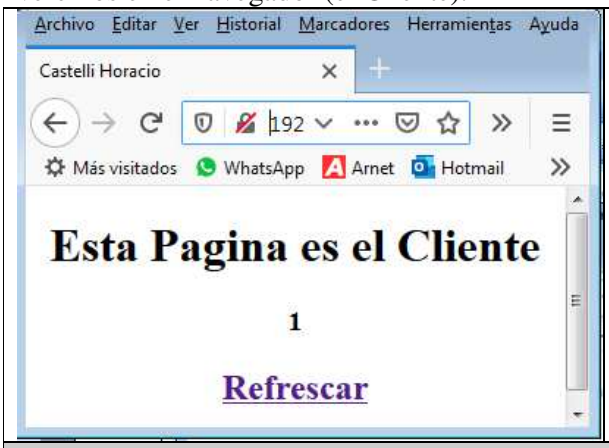
Reinicio Automático y Título de Pagina		
	client.println("<head>");	Asegúrese que siempre el reinicio, al igual que el titulo estén entre las sentencias <head> y </head>
	client.println("<title>Castelli Horacio</title>");	Título de la Página
	client.println("<meta http-equiv= refresh content= 10 >");	Reinicio Automático: Asegura la actualización según tiempo configurado (10 segundos). El tiempo puede cambiarlo.
	client.println("</head>");	Asegúrese que siempre el reinicio, al igual que el titulo estén entre las sentencias <head> y </head>
Estas líneas deben ir siempre juntas.		

-o-

Reinicio Manual de Página		
	client.print(F("<a href='http://"));	Estas tres líneas forman el Link o autollamado de la pagina
	client.print(Ethernet.localIP( ));	Entrega el IP que tenia la página
	client.println(F("'><h2>Refrescar<h2>"));	El titulo que veremos en el link
Estas líneas deben ir siempre juntas.		

UNA VEZ QUE FUNCIONA, ESTO ES LO QUE VEREMOS

Presta atención al el código enviado por nuestro programa (el Servidor) que construye la Página que veremos en el navegador (el Cliente).




```
<html>
<head>
<title>Castelli Horacio</title>
<meta http-equiv= refresh content=30>
</head>
<body>
<div style='text-align:center;'>
<h1>Esta Pagina es el Cliente</h1>
<h3>1</h3>
<a href='http://192.168.1.150'> <h2>Refrescar<h2></a>
</div>
</body>
</html>
```

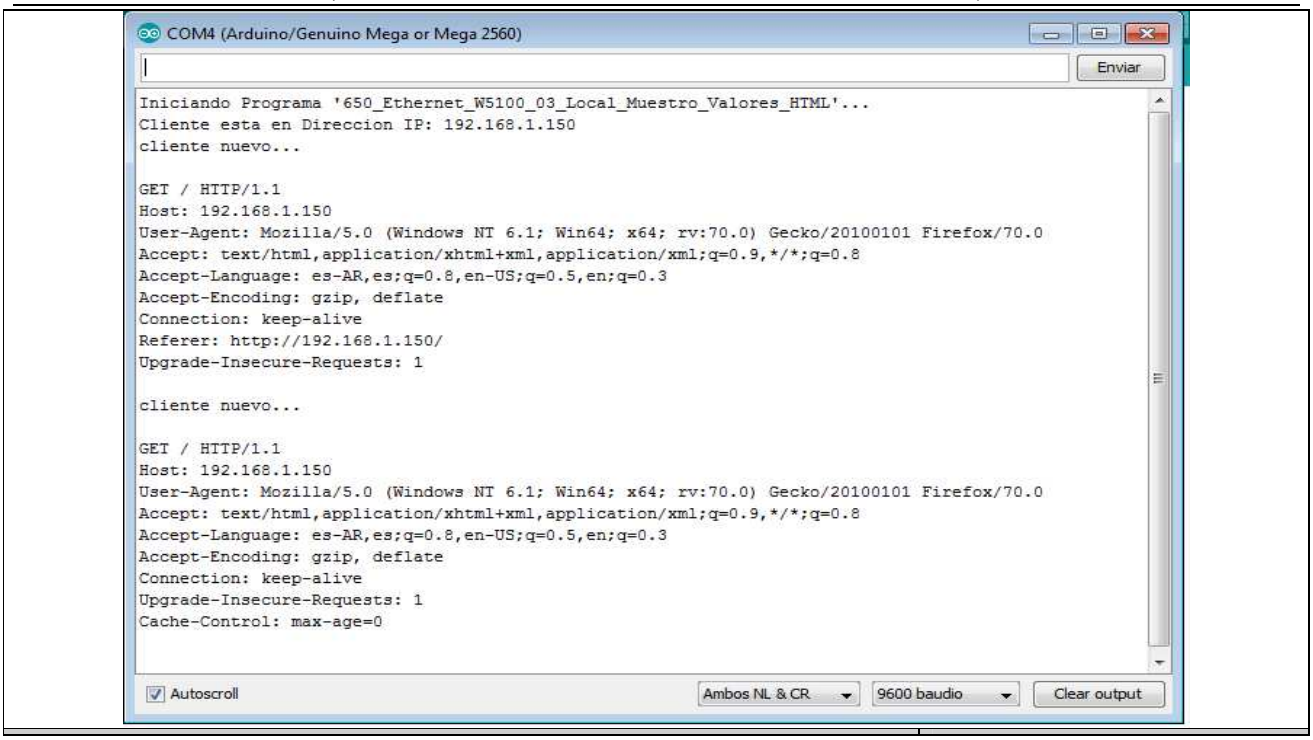
Esto es lo que se visualizará en el Monitor del Puerto Serie.

Todo el texto puede dividirse en tres partes:

Primera: Es el texto que se pone al iniciar el programa. Muestra el IP donde se publicara la Página. Que se debe copiar y pegar en cualquier navegador.

Segunda: Durante una Actualización manual. Es la comunicación entre Cliente y el Servidor.

Tercera: Durante una Actualización Automática. Es la comunicación entre Cliente y el Servidor.



Sugiero que repases la guía correspondiente a "Lector / Grabador de Tarjetas SD y Micro SD" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan acá y sin modificaciones**.

3) Grabar en un Archivo de texto, los valores que tienen cada uno de los Pines de nuestra Placa Arduino, en un momento determinado. Actualizar Automáticamente el Archivo cada 10 Segundos.

Leer los valores de cada uno de los pines de nuestra placa Arduino tiene en un memento determinado y grabarlos en un archivo de texto. Realizar los controles necesarios para que el archivo se actualice cada 10 segundos.

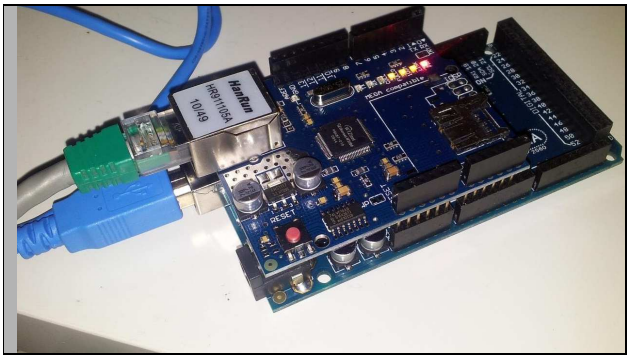
(Programa "660_Ethernet_W5100_01_ Grabo_Valores_de_Pines")

```
1   #include<SD.h>              // Esta Librería se encuentra en el IDE de Arduino
2   #define PinLectora_SD 7  // Este es el Pin denominado destinado al dispositivo Lector/Grabador
3
4   File ArchivoEjemplo; // Variable del tipo Archivo
5   int  i;                        // Valor usado para el for que recorre los Pines
6
7   /*************** VARIABLES  Y FUNCIÓN PARA CONTROLAR TIEMPO ***************/
8   unsigned long Inicia_Segundos, Queda_Tiempo;
9   unsigned long Intervalo(unsigned int H, unsigned long M, unsigned long S){
10     unsigned long IntervaloSegundos = H*3600 + M*60 + S;
11     unsigned long Pasaron =  millis( )/1000 - Inicia_Segundos;
12     unsigned long RegresivaSegundos = IntervaloSegundos - Pasaron;
13     if( Pasaron >= IntervaloSegundos){
14        Inicia_Segundos = millis( )/1000;
15        RegresivaSegundos = 0;        // Aseguro que retorne Cero
16     }
17     return (RegresivaSegundos);
18  }
19
20  /********* Programo las Acciones que se deben hacer por única vez en la Función Setup( ) ********/
21  void setup( ){
22    Serial.begin(9600); // Abro el Puerto Serie
23    while (!Serial); // Esperamos que el puerto serie este abierto.
24    Serial.println("Iniciando Programa '660_Ethernet_W5100_01_Grabo_Valores_de_Pines'");
```

```
25
26  /*************** Controlando que inicie bien la Lectora/Escritora de Tarjetas ****************/
27    Serial.print( F("Iniciando SD ...") );
28    if (!SD.begin( PinLectora_SD )){
29      Serial.println( F("Error al iniciar") );
30      return;
31    }
32    Serial.println(F("Iniciado correctamente\n"));
33  /*********************************************************************************************/
34    Inicia_Segundos = millis( ) / 1000;  // Toma el tiempo Inicial
35  }
36
37  void loop( ){
38    //                                H,M,  S
39    Queda_Tiempo = Intervalo(0,  0, 10);
40    Serial.print("Tiempo para Actualización: ");
41    Serial.println(Queda_Tiempo);
42
43    if(Queda_Tiempo == 0){
44       Serial.println("\nActualizando Archivo...\n");
45       ArchivoEjemplo = SD.open("Pines.txt", FILE_WRITE); // Abriendo para Escribir
46       if(ArchivoEjemplo != NULL) { // Si NO pudo abrirse, la Variable Tiene Valor NULL
47          ArchivoEjemplo.println("Pines Digitales. Valores Censados" );
48          for (int i = 0; i < 13; i++){ // Cada valor de i, representa un numero de Pin
49             ArchivoEjemplo.print("D");  // Grabo letra "D" indicando que es un Pin Digital
50             ArchivoEjemplo.print(i);      // Grabo el numero de Pin
51             ArchivoEjemplo.print(" = ");  // Grabo Signo Igual
52             ArchivoEjemplo.println(digitalRead(i)); // Leo el PIN y muestro el valor Digital
53          }
54          ArchivoEjemplo.println("\nPines Analógicos. Valores Censados \n" );
55          for (int i = 0; i < 7; i++){   // Cada valor de i, representa un numero de Pin
56             ArchivoEjemplo.print("A");  // Grabo letra "D" indicando que es un Pin Digital
57             ArchivoEjemplo.print(i);     // Grabo el numero de Pin
58             ArchivoEjemplo.print(" = ");  // Grabo Signo Igual
59             ArchivoEjemplo.println(analogRead(i)); // Leo el PIN y muestro el valor Digital
60          }
61          ArchivoEjemplo.println("\n\n==================================\n\n"); // Grabo Línea
62          ArchivoEjemplo.close( ); // Cerrando archivo
63       }else{
64          Serial.println("Error al abrir el archivo");
65       }
66    }
67    delay(1000);
68  } // Llave que Finaliza la función loop
```

La información que se graba es tomada directamente de los pines de Arduino, aun cuando no se estén usando, ya que siempre hay valores aleatorios. Pero si conectáramos algún sensor, esos valores serian representativos de la realidad.

CIRCUITO PARA NUESTRO PROYECTO

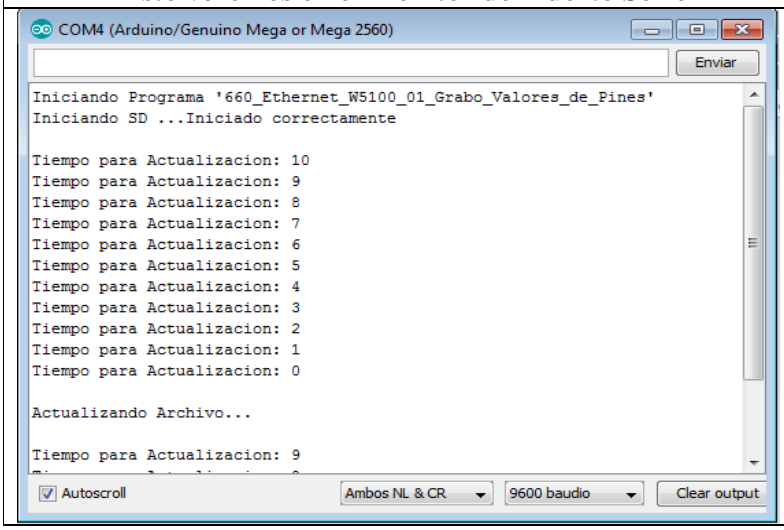


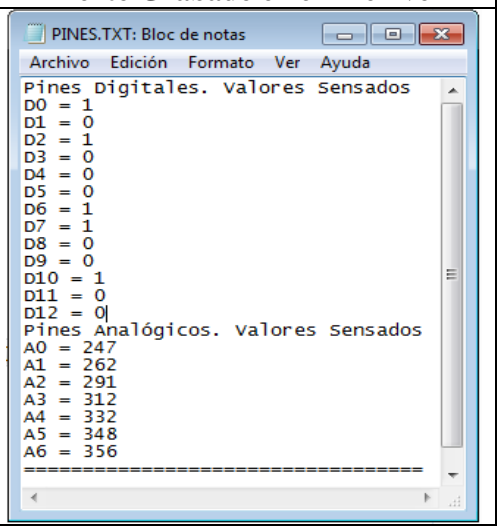
Lista de Materiales: 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Tarjeta Micro SD con adaptador (Máx. 16Gb), 1 Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

UNA VEZ QUE FUNCIONA, ESTO ES LO QUE VEREMOS

Esto veremos en el Monitor del Puerto Serie	Texto Grabado en el Archivo





4) Mostrar en página HTML los datos de los pines de nuestra Placa Arduino.

Se quiere mostrar en una página HTML local, los valores que se generan en cada uno de los pines de nuestra placa Arduino. La página debe actualizarse cada vez que sea requerido o cada 10 segundos en forma Automática. Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - Apéndice C - Protocolo HTTP** - **"Apéndice E - Funciones de la Librería Ethernet**".

Para analizar y entender este ejemplo, debemos considerar a la página como el cliente que pide información. A nuestro Arduino (el servidor que brinda la información). Y para actualizar los datos, simplemente usamos un link que hace un llamado a la misma página (Se llama a si misma).

(Programa "650_Ethernet_W5100_04_Local_Muestro_Valores_HTML")

```
1  #include <SPI.h>         // Esta Librería esta incluida en el IDE de Arduino
2  #include <Ethernet.h>  // Esta Librería esta incluida en el IDE de Arduino
3
4  // Dirección MAC (siglas en inglés de media access control.
5  //              en español "control de acceso al medio")
6  // Dirección MAC - Ver Apéndice A de esta Guía.
7  byte mac[ ] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
8
9  // Ante cualquier duda puedes ver "Apéndice B - Dirección IP"
10 IPAddress ip(192,168,1,150); // Dirección IP que asignamos nuestro Arduino 192.168.1.150
11
12 // Ante cualquier duda puedes repasar "Apéndice F - PUERTOS DE COMUNICACIÓN"
13 EthernetServer server(80); // Crea servidor que escuche conexiones entrantes por el puerto
14                             // especificado, en este caso, el puerto 80
15 void setup( ){
16   Serial.begin(9600);
17   while (!Serial) ; // Esperamos que el puerto serie este abierto.
18
19   Serial.println("Iniciando Programa '650_Ethernet_W5100_03_Local_Muestro_Valores_HTML'...");
20
21   Ethernet.begin(mac, ip); // Conectamos a la Red local, con nuestro MAC y Nuestra IP
22   server.begin( );  // Le pide al servidor que comience a escuchar las conexiones entrantes.
23                     // (Nuestra Placa Arduino).
24
25   Serial.print("Server esta en la Direccion IP: ");
26   Serial.println(Ethernet.localIP( )); // La función Ethernet.localIP( ) Retorna el IP que le hayamos Configurado
27 } // Finalización de la Función Setup.
28
```

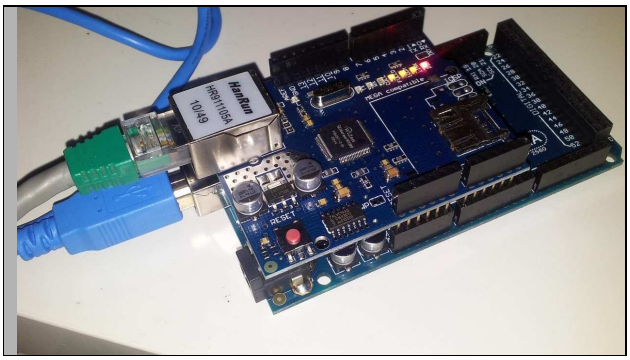
```
29  EthernetClient client;          // Variable declarada para usar dentro del Loop
30  bool LineaActualEstaVacia; // Variable declarada para usar dentro del Loop
31
32  void loop( ){
33    client = server.available( ); //Devuelve el cliente que esté conectado al servidor y
34                                  //tiene datos disponibles a leer. Recordar que la conexión
35                                  // persiste cuando el objeto de cliente devuelto queda
36                                  //fuera de alcance
37    if(client){  // Pregunta si hay algun cliente conectado al Servidor
38                 // Recordar que este programa, genera el Cliente (la página que lee nuestro Arduino)
39                 // y Maneja el Servidor, Nuestra placa Arduino
40      Serial.println("cliente nuevo..."); // Cada vez que se inicia la página se considera
41                                          // una página nueva, es decir un nuevo cliente
42                                          //(pide datos a nuestro Arduino).
43      LineaActualEstaVacia = true;
44
45      while (client.connected( )){ // Mientras el Cliente este conectado
46        if (client.available( )){  // Si el cliente tiene datos para leer
47          char c = client.read( );  // Leo un caractér
48          Serial.write(c);          // Muestro caractér leído por el Monitor del Puerto Serie
49
50          // Al detectar línea en blanco, servir página a cliente
51          if (c == '\n' && LineaActualEstaVacia){
52  /**** Comienza envío de partes de página que se mostrara en la dirección de IP configurada ****/
53            client.println(F("HTTP/1.1 200 OK\nContent-Type: text/html"));
54            client.println( );
55
56            client.println(F("<html>\n<head>"));
57            client.println(F("<title>Castelli Horacio</title>"));  // Titulo de la Pagina
58  /************************ Actualización Automática cada 30 segundos ***********************/
59            client.println(F("<meta http-equiv= refresh content= 10 >")); //Asegura cada 10 segundos un
60            client.println(F("</head>\n<body>"));                         // reinicio Automático de página
61            client.println(F("<div style='text-align:center;'>"));
62            client.println(F("<h1>Esta Página es el Cliente</h1>"));
63  /*******************************************************************************************/
64            client.println(F("<h2>Entradas digitales</h2>"));
65            for (int i = 0; i < 13; i++){ // Cada valor de i, representa un numero de Pin
66              client.print("D");    // Muestro letra "D" indicando que es un Pin Digital
67              client.print(i);      // Muestro el numero de Pin
68              client.print(" = ");  // Signo Igual
69              client.print(digitalRead(i)); // Leo el PIN y muestro el valor Digital
70              client.println(F("<br />"));  //Bajo un Renglón en página
71            }
72  /*******************************************************************************************/
73            client.println(F("<h2>Entradas analógicas</h2>"));
74            for (int i = 0; i < 7; i++){    // Cada valor de i, representa un numero de Pin
75              client.print("A");   // Muestro letra "A" indicando que es un Pin Analógico
76              client.print(i);     // Muestro el numero de Pin
77              client.print(" = "); // Signo Igual
78              client.print(analogRead(i)); // Leo el PIN y muestro el valor Analógico
79              client.println(F("<br />")); //Bajo un Renglón en página
80            }
81  /*******************************************************************************************/
82   /***** Acá se Arma el link para que la página se llame a si misma (Refrescar datos Manualmente) ****/
83            //client.println(F("<a href='http://192,168,1,150'><h2>Refrescar<h2></a>"));
84            client.print(F("<a href='http://"));
85            client.print(Ethernet.localIP( ));   //Entrega el IP que tenia la página
86            client.println(F("'><h2>Refrescar<h2></a>"));
87            client.println(F("</div>\n</body></html>"));
88            break;
89  /* Cuando en la página se hace clic sobre el link, se genera una petición de mostrar pagina, la petición se
90     hace al servidor, que casualmente es nuestra placa Arduino, y en consecuencia muestra otra vez la pagina.*/
91  /*******************************************************************************************/
92          } //Final de la Pregunta si hay que construir la página (el cliente)
93          if( c == '\n'){ // '\n' es una nueva línea - Comienza nueva línea de Respuesta
```

```
            LineaActualEstaVacia = true;
          }else if(c != '\r'){ //"\r" es un "retorno de carro" - Esta leyendo una Respuesta.
            LineaActualEstaVacia = false;
          }
        }// Final de la pregunta si el Cliente tiene para leer
      }// Final del While(Cliente conectado)
      delay(1); // Pequeña demora, que puede variar según sea el gusto del programador
      client.stop( );
    }// Finaliza pregunta si hay un cliente conectado al Servidor
}
```

Una vez que hayas copiado el código en tu editor, hay que hacerlo funcionar, y esto es muy fácil: Abre el Monitor del puerto Serie, y te dará la IP dende se ha creado el Cliente (La página web).

Copia este IP, y pégalo en cualquier navegador, y en el acto veras la página. Y si mantienes abierto también el Monitor del Puerto Serie, podrás ver la comunicación entre página y servidor. Es interesante que comiences a entenderla.

CIRCUITO PARA NUESTRO PROYECTO

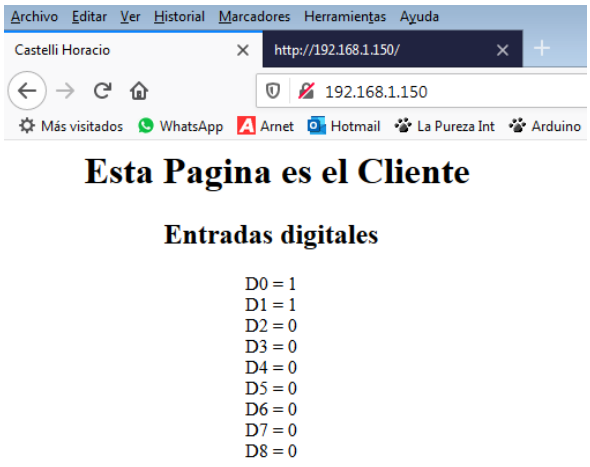


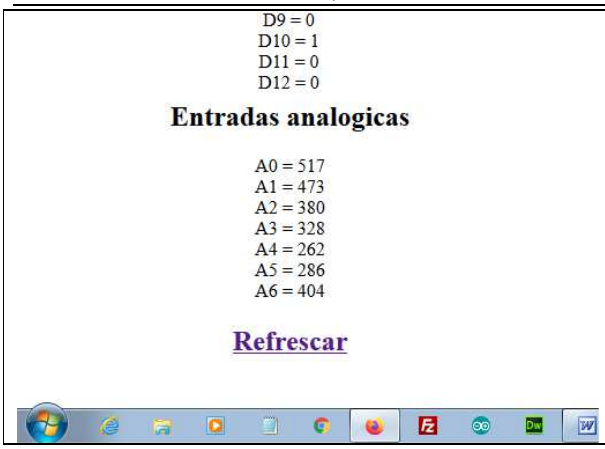
Lista de Materiales: 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

UNA VEZ QUE FUNCIONA, ESTO ES LO QUE VEREMOS

Presta atención al el código enviado por nuestro programa (el Servidor) que forma la página ya formada (el Cliente) y con la información tomada desde nuestra Placa Arduino.

Vista de la Página (Cliente) en el navegador	Código de la Página (cliente)
	<html> <head> <title>Castelli Horacio</title> **<meta http-equiv= refresh content=10>** </head> <body> Esta página es el Cliente Entradas digitales D0 = 1 D1 = 1 D2 = 0 D3 = 0 D4 = 0 D5 = 0 D6 = 0 D7 = 0 D8 = 0 D9 = 0 D10 = 1 D11 = 0 D12 = 0 Entradas analógicas A0 = 517 A1 = 473 A2 = 380




```
A3 = 328<br />
A4 = 262<br />
A5 = 286<br />
A6 = 404<br />
<a href='http://192.168.1.150'> <h2>Refrescar<h2></a>
</div>
</body></html>
```

Sugiero que repases la guía correspondiente a "Lector / Grabador de Tarjetas SD y Micro SD" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan acá y sin modificaciones**.

5) Combinando los dos Programas anteriores, ahora simultáneamente Grabar en un Archivo de texto y Mostrar en página HTML los datos leídos directamente de los pines de nuestra Placa Arduino.

Se quiere Grabar los datos leídos directamente de cada pin de nuestra placa Arduino en un Archivo de texto, a la vez que se los muestra en una página HTML

Los datos del archivo y pagina, deben actualizarse cada vez que sea requerido por el usuario o automáticamente cada 10 segundos. Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - Apéndice C - Protocolo HTTP" - "Apéndice E - Funciones de la Librería Ethernet**".

Para analizar y entender este ejemplo, debemos considerar a la página como el cliente que pide información. A nuestro Arduino (el servidor que brinda la información). Y para actualizar los datos, simplemente usamos un link que hace un llamado a la misma página (Se llama a si misma).

(Programa "660_Ethernet_W5100_02_Grabo_y_Local_Muestro_Valores_HTML")

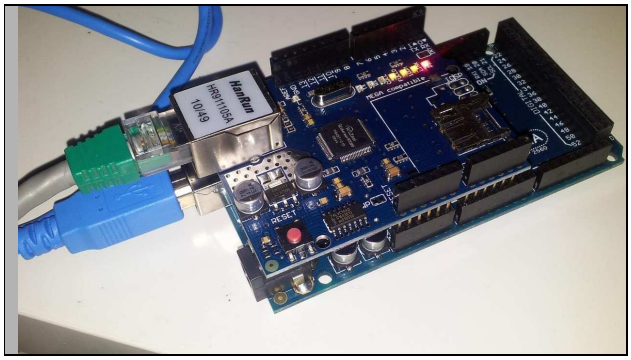
```
1  #include<SD.h>        // Esta Librería se encuentra en el IDE de Arduino
2  #include <SPI.h>      // Esta Librería esta incluida en el IDE de Arduino
3  #include <Ethernet.h> // Esta Librería esta incluida en el IDE de Arduino
4
5  const int Pin_Habilta_W5100 = 10,
6            Pin_Habilta_SD   = 4;
7
8  const int Pin_Lectora_SD = 7; // Valores Posibles 2,3,5,6,7,8,9
9  File Archivo_Ejemplo;  // Variable del tipo Archivo
10
11 // Dirección MAC (siglas en inglés de media access control.
12 //                          en español "control de acceso al medio")
13 // Dirección MAC - Ver Apéndice A de esta Guía.
14 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
15
16 // Ante cualquier duda puedes ver "Apéndice B - Dirección IP"
17 IPAddress ip(192,168,1,150); // Dirección IP que asignamos nuestro Arduino 192.168.1.150
18
19 // Ante cualquier duda puedes repasar "Apéndice F - PUERTOS DE COMUNICACIÓN"
20 EthernetServer server(80);  // Crea servidor que escuche conexiones entrantes por el puerto
21                              //especificado. en este caso, el puerto 80
22 int Activar_W5100(void){
23   digitalWrite(Pin_Habilta_SD, HIGH);       //Valor HIGH Deshabilita
24   digitalWrite(Pin_Habilta_W5100, LOW);  // Valor LOW Habilita
25   return(1);
26 }
27
```

```
28  int Activar_SD(void){
29     digitalWrite(Pin_Habilta_W5100, HIGH);    // Valor HIGH Deshabilita
30     digitalWrite(Pin_Habilta_SD, LOW);          // Valor LOW Habilita
31     return(1);
32  }
33
34  void setup( ){
35     Serial.begin(9600);
36     while (!Serial); // Esperamos que el puerto serie este abierto.
37
38     Serial.println("Iniciando Programa '660_Ethernet_W5100_02_Grabo_y_Local_Muestro_Valores_HTML'");
39
40     pinMode( Pin_Habilta_W5100, OUTPUT);
41     pinMode( Pin_Habilta_SD, OUTPUT);
42
43     Serial.println(F("Iniciando Modulo Lectora/grabador de Tarjetas SD ..."));
44     Activar_SD( );
45     if ( !SD.begin( Pin_Lectora_SD ) ){
46        Serial.print( F("Error al iniciar (Verifique Micro SD)") );
47        return(0);
48     }
49     Serial.print(F("Lectora/grabador SD, Iniciado Correctamente - Usando Pin "));
50     Serial.println(Pin_Lectora_SD);
51
52     Serial.println(F("Iniciando Modulo ETHERNET W5100 ..."));
53     Activar_W5100( );
54     Ethernet.begin(mac, ip);
55     server.begin( );  // Le dice al servidor que comience a escuchar las conexiones entrantes.
56     Serial.print("Server esta en la Dirección IP: ");
57     Serial.println(Ethernet.localIP( )); // Retorna el IP que le hayamos Configurado
58  }
59
60  EthernetClient client;             // Variable Usadas en Loop
61  bool LineaActualEstaVacia;   // Variable Usadas en Loop
62  String Linea;                       // Variable Usadas en Loop
63
64  void loop( ){
65    Activar_SD( );
66    Archivo_Ejemplo = SD.open("Pines.txt", FILE_WRITE); // Abriendo para Escribir
67    Activar_W5100( );
68    client = server.available( ); //Devuelve el cliente que esté conectado al servidor y
69                                  //tiene datos disponibles a leer. Recordar que la conexión
70                                  // persiste cuando el objeto de cliente devuelto queda
71                                  //fuera de alcance
72    if(client){  // Pregunta si hay algún cliente conectado al Servidor
73                 // Recordar que este programa, genera el Cliente (la Pagina que lee nuestro Arduino)
74                 // y Maneja el Servidor, Nuestra placa Arduino
75       Serial.println("cliente nuevo..."); // Cada vez que se inicia la Pagina se considera una
76                  //  página nueva, es decir un nuevo cliente (pide datos a nuestro Arduino).
77
78       LineaActualEstaVacia = true;
79
80       while (Activar_W5100( ) && client.connected( )){ // Mientras el Cliente este conectado
81         if (client.available( )){  // Si el cliente puede leer
82           char c = client.read( );
83           Serial.write(c);
84
85           // Al recibir línea en blanco, servir página a cliente
86           if (c == '\n' && LineaActualEstaVacia){
87  /**** Comienza envio de partes de pagina que se mostrara en la dirección de IP configurada ****/
88              client.println(F("HTTP/1.1 200 OK\nContent-Type: text/html"));
89              client.println( );    //Una línea vacía indica que toda la meta-información ha sido enviada.
90              client.println(F("<html>\n<head>"));
91              client.println(F("<title>Castelli Horacio</title>"));  // Titulo de la Pagina
92              client.println(F("<meta http-equiv= refresh content=15>")); //Actualiza cada 15 seg. Automáticamente
```

```
93              client.println(F("</head>\n<body>"));
94              client.println(F("<div style='text-align:center;'>"));
95              client.println(F("<h1>Esta Pagina es el Cliente</h1>"));
96    /*************************************************************************************/
97              Linea = "Entradas digitales";
98              client.print(F("<h2>"));
99              client.print(Linea);
100             client.println(F("</h2>"));
101
102             Activar_SD( );
103             Linea += "\n";
104             Archivo_Ejemplo.println( Linea );
105             for (int i = 0; i < 13; i++){ // Cada valor de i, representa un numero de Pin
106                Linea = "D";
107                Linea += i;
108                Linea += "=";
109                Linea += digitalRead(i);
110
111                Activar_W5100( );
112                client.print(Linea);
113                client.println(F("<br />"));  //Bajo un Renglón en página HTML
114
115                Activar_SD( );
116                Linea += "\n";
117                Archivo_Ejemplo.println( Linea );
118             }
119   /*************************************************************************************/
120             Linea = "Entradas analógicas";
121
122             Activar_W5100( );
123             client.print(F("<h2>"));
124             client.print(Linea);
125             client.println(F("</h2>"));
126
127             Activar_SD( );
128             Linea += "\n";
129             Archivo_Ejemplo.println( Linea );
130             for (int i = 0; i < 7; i++){    // Cada valor de i, representa un numero de Pin
131                Linea = "A";
132                Linea += i;
133                Linea += "=";
134                Linea += analogRead(i);
135
136                Activar_W5100( );
137                client.print(Linea);
138                client.println(F("<br />"));  //Bajo un Renglón en página HTML
139
140                Activar_SD( );
141                Linea += "\n";
142                Archivo_Ejemplo.println( Linea );
143             }
144             Archivo_Ejemplo.println( "==================================\n\n" );
145   /*************************************************************************************/
146             Activar_W5100( );
147    /***** Acá se Arma el link para que la pagina se llame a si misma (para refrescar datos) ****/
148             client.print(F("<a href='http://"));
149             client.print(Ethernet.localIP( ));   //Entrega el IP que tenia la pagina
150             client.println(F("'><h2>Refrescar<h2></a>"));
151             client.println(F("</div>\n</body></html>"));
152             break;
153   /* Cuando en la pagina se hace clic sobre el link, se genera una petición de mostrar pagina, la petición se
154      hace al servidor, que casualmente es nuestra placa Arduino, y en consecuencia muestra otra vez la pagina.*/
155   /*************************************************************************************/
156          } //Final de la Pregunta si hay que construir la pagina (el cliente)
157          if( c == '\n'){          // '\n' es una nueva línea - Comienza nueva línea de Respuesta
```

```
158             LineaActualEstaVacia = true;
159           }else if(c != '\r'){    //"\r" es un "retorno de carro" - Esta leyendo una Respuesta.
160             LineaActualEstaVacia = false;
161           }
162         }// Final de la pregunta si el Cliente tiene para leer
163       }// Final del While(Cliente conectado)
164       delay(1);
165       client.stop( );
166     }// Finaliza pregunta si hay un cliente conectado
167     Activar_SD( );
168     Archivo_Ejemplo.close( ); // Cerrando archivo
169   } // Final de la Función Loop
```

CIRCUITO PARA NUESTRO PROYECTO

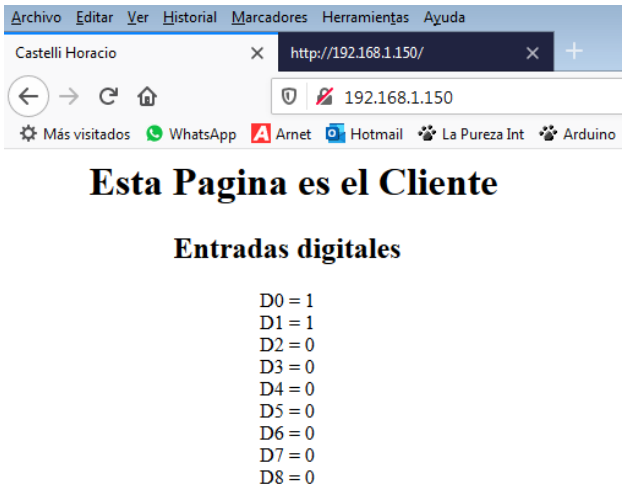


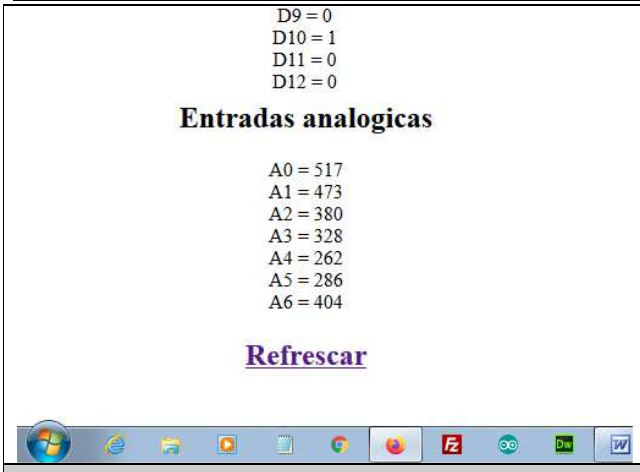
Lista de Materiales: 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Tarjeta Micro SD con adaptador (Máx. 16Gb), 1 Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

UNA VEZ QUE FUNCIONA, ESTO ES LO QUE VEREMOS

Presta atención al el código enviado por nuestro programa (el Servidor) que forma la página ya formada (el Cliente) y con la información tomada desde nuestra Placa Arduino.

Vista de la Página (Cliente) en el navegador	Código de la Página (cliente)
	<html> <head> <title>Castelli Horacio</title> **<meta http-equiv= refresh content=10>** </head> <body> Esta página es el Cliente Entradas digitales D0 = 1 D1 = 1 D2 = 0 D3 = 0 D4 = 0 D5 = 0 D6 = 0 D7 = 0 D8 = 0 D9 = 0 D10 = 1 D11 = 0 D12 = 0 Entradas analógicas A0 = 517 A1 = 473 A2 = 380 A3 = 328 A4 = 262 A5 = 286




```
A6 = 404<br />
<a href='http://192.168.1.150'> <h2>Refrescar<h2></a>
</div>
</body></html>
```

Esto veremos en el Monitor del Puerto Serie	Texto Grabado en el Archivo
Iniciando Programa '660_Ethernet_W5100_02_Grabo_y_Local_Muestro_Valores_HTML' Iniciando Modulo Leto/Grabador de Tarjetas SD ... Lectora/grabador SD, Iniciado Correctamente - Usando Pin 7 Iniciando Modulo ETHERNET W5100 ... Server esta en la Direccion IP: 192.168.1.150 cliente nuevo... GET / HTTP/1.1 Host: 192.168.1.150 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate Connection: keep-alive Upgrade-Insecure-Requests: 1 cliente nuevo... GET / HTTP/1.1 Host: 192.168.1.150 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate Connection: keep-alive Upgrade-Insecure-Requests: 1 Cache-Control: max-age=0	**Entradas digitales** D0=1 D1=0 D2=0 D3=0 D4=0 D5=0 D6=0 D7=1 D8=0 D9=0 D10=1 D11=0 D12=0 **Entradas analógicas** A0=248 A1=298 A2=351 A3=369 A4=376 A5=298 A6=260 ========== **Entradas digitales** D0=1 D1=1 D2=0 D3=0 D4=0 D5=0 D6=0 D7=1 D8=0 D9=0 D10=1 D11=0 D12=0 **Entradas analógicas** A0=517 A1=473 A2=380 A3=328 A4=262 A5=286 A6=404 ==========

En el Monitor del puerto serie, encontramos 3 grupos de información, la primera corresponde al inicio del dispositivo, la segunda al despliegue de los datos Automático, y el tercero al pedido del usuario de la actualización de datos.

A la derecha, visualizamos lo que se graba en el archivo ante cada uno de las actualizaciones. la manual y la automática.

6) Prender y Apagar dos Leds desde una página HTML Local.

Se quiere prender y apagar dos Leds (uno controlado por un Pin Digital, y otro controlado por un Pin Analógico) Ambos Leds, desde una página Web. La pagina HTML, se actualizara Automáticamente cada 15 segundos, o cada vez que el Usuario lo requiera. Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - Apéndice C - Protocolo HTTP" - "Apéndice E - Funciones de la Librería Ethernet**".

(Programa "670_Ethernet_W5100_02_Prendo_2_Led")

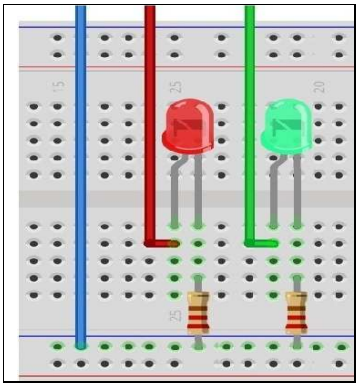
```
1   #include <SPI.h>
2   #include <Ethernet.h>
3
4   byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
5   IPAddress ip(192, 168, 1, 150); // 192.168.1.150
6   EthernetServer server(80);
7
8   const int pinLed1 =    3,
9             pinLed2 =  A0;
10
11  void setup( ){
12    Serial.begin(9600);
13
14    Ethernet.begin(mac, ip);
15    server.begin( );
16
17    Serial.print("Servidor esta en Dirección IP: ");
18    Serial.println(Ethernet.localIP( ));
19
20    pinMode(pinLed1, OUTPUT);
21    pinMode(pinLed2, OUTPUT);
22
23    digitalWrite(pinLed1, LOW);
24    digitalWrite(pinLed2, LOW);
25  }
26  /*************************************************************************************/
27  /***** Variables para usar en Función Loop ***************************************/
28  EthernetClient   client;
29  bool   ActualLineaVacia;
30  String   cadena,
31             Comando;
32  char    c;
33  int      posicion,
34        Cuenta_Actual = 0;
35  /*************************************************************************************/
36
37  void loop( ){
38    client = server.available( );
39    if (client){
40      Serial.println("Nuevo Cliente");
41      ActualLineaVacia = true;
42      cadena = "";
43      while (client.connected( )){
44        if (client.available( )){
45          c = client.read( );
46          Serial.write(c);
47  /*************************************************************************************/
48  /***** Identificación del comando (prender o apagar cada led) ********************/
49          if (cadena.length( )<50){
50              cadena.concat(c);
51              // Buscar campo "Led_"
52              posicion = cadena.indexOf("Led_");
53              Comando = cadena.substring(posicion);
54
```

```
55        // Cuando encuentro Campo "Led_" solo queda identificar numero entre 01 y 99
56        // y prender o apagar el led correspondiente
57              if (Comando == "Led_01=0"){
58                 digitalWrite(pinLed1, LOW);
59              }else if (Comando == "Led_01=1"){
60                 digitalWrite(pinLed1, HIGH);
61              }else if (Comando == "Led_02=0"){
62                 digitalWrite(pinLed2, LOW);
63              }else if (Comando == "Led_02=1"){
64                 digitalWrite(pinLed2, HIGH);
65              }
66           }
67  /****** Final de la identificación de comando ********************************************/
68  /*****************************************************************************************/
69            // Al recibir línea en blanco, servir página a cliente
70           if (c == '\n' && ActualLineaVacia){
71              client.println(F("HTTP/1.1 200 OK\nContent-Type: text/html"));
72              client.println( );    //Una línea vacía indica que toda la meta-información ha sido enviada
73              client.println(F("<html>\n<head>"));
74              client.println(F("<title>Castelli Horacio</title>"));
75              client.println(F("<meta http-equiv= refresh content=15>")); //Actualización Automática de Página
76              client.println(F("</head>\n<body>"));
77              client.println(F("<div style='text-align:center;'>"));
78  /*****************************************************************************************/
79  /***** Generación de botones y envío de comandos para cada led ***************************/
80              client.println(F("<h2>Control Salida Digital</h2>"));
81              client.print(F("Estado LED 1 = "));
82              client.println(digitalRead(pinLed1) == LOW ? "OFF" : "ON");
83              client.println(F("<br/>"));
84              client.println(F("<button onClick=location.href='./?Led_01=1'>ON</button>"));
85              client.println(F("<button onClick=location.href='./?Led_01=0'>OFF</button>"));
86              client.println(F("<br/><br/>"));
87
88              client.println(F("<h2>Control Salida Analogica</h2>"));
89              client.print(F("Estado LED 2 = "));
90              client.println(digitalRead(pinLed2) == LOW ? "OFF" : "ON");
91              client.println(F("<br/>"));
92              client.println(F("<button onClick=location.href='./?Led_02=1'>ON</button>"));
93              client.println(F("<button onClick=location.href='./?Led_02=0'>OFF</button>"));
94              client.println(F("<br/>"));
95  /*****************************************************************************************/
96  /****** Contamos las actualizaciones y se muestran en la pagina HTML **********************/
97              Cuenta_Actual++;
98              client.print(F("<h3> Actualizacion Nro: "));
99              client.print(Cuenta_Actual);
100             client.println(F("</h3>"));
101 /*****************************************************************************************/
102 /****** Actualización Manual de la Página HTML ********************************************/
103             client.print(F("<a href='http://"));
104             client.print(Ethernet.localIP( ));   //Entrega el IP que tenia la pagina
105             client.println(F("'><h2>Refrescar<h2></a>"));
106 /*****************************************************************************************/
107             client.println(F("</div>\n</body></html>"));
108             break;
109          }
110          if (c == '\n'){
111             ActualLineaVacia = true;
112          }else if (c != '\r'){
113             ActualLineaVacia = false;
114          }
115       } //Fin if cliente puede leer
116    } // Fin while(cliente conectado
117
118    delay(10);
119    client.stop( );
```

```
120   } // final if si hay cliente conectado
121 } //Final de la función Loop
122
```

Al pulsar alguno de los botones, para prender o apagar un Led, se dispara una nueva solicitud al servidor Arduino, con diferente URL cada vez, y esta URL contiene el comando. Arduino captura la nueva solicitud, y con la URL recibida, analiza el contenido, ya que es el comando, que usaremos para realizar las acciones.

CIRCUITO PARA NUESTRO PROYECTO

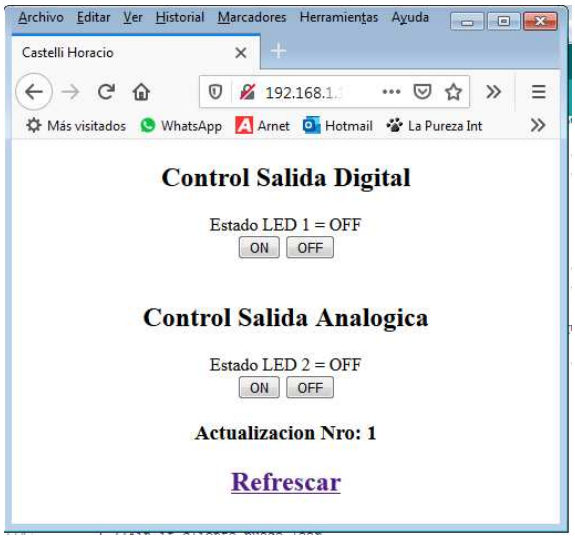


Lista de Materiales: 1 Placa Protoboard - 2 LED - 2 Resistencia de 470Ω - 3 Cables Macho-Macho - 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Placa Arduino y 1 Cable USB - Placa Arduino.

Los cables deberán ser conectados: Antes de comenzar a conectar los cables, recomiendo fijarse bien como están ahora los pines, ya que esta conectada la Placa Arduino y el modulo **Ethernet Shield W5100.**
Como podrá observar, los pines del modulo W5100, mantienen la numeración que tiene Arduino, así que simplemente conectaremos a cada pin, tal como lo hacíamos antes.

Comencemos entonces. Primero conectar el cable USB a la PC y el cable Ethernet a un router o modem. Luego, Cable Azul a GND. Cable Rojo al Pin 3. **Cable Verde** al Pin A0.

UNA VEZ QUE FUNCIONA, ESTO ES LO QUE VEREMOS




```
<html> <head>
<title>Castelli Horacio</title>
<meta http-equiv= refresh content=15>
</head>
<body><div style='text-align:center;'>
<h2>Control Salida Digital</h2>
Estado LED 1 = OFF
<br/>
<button onClick=location.href='./?Led_01=1'>ON</button>
<button onClick=location.href='./?Led_01=0'>OFF</button>
<br/><br/>
<h2>Control Salida Analogica</h2>
Estado LED 2 = OFF
<br/>
<button onClick=location.href='./?Led_02=1'>ON</button>
<button onClick=location.href='./?Led_02=0'>OFF</button>
<br/>
<h3> Actualizacion Nro: 1</h3>
<a href='http://192.168.1.150'><h2>Refrescar<h2></a>
</div>
</body> </html>
```

Sugiero que repases la guía correspondiente a "Lector / Grabador de Tarjetas SD y Micro SD" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan acá y sin modificaciones**.

7) Prender y Apagar dos Leds desde una página HTML Local. Grabar en un archivo el estado de los LEDs cada vez que alguno cambia de estado.

Se quiere prender y apagar dos Leds (uno controlado por un Pin Digital, y otro controlado por un Pin Analógico) Ambos Leds, desde una página Web. La pagina

HTML, se actualizara Automáticamente cada 15 segundos, o cada vez que el Usuario lo requiera. Ante cualquier duda, puede consultar al final de la guía los apéndices: **"Apéndice A - La Dirección MAC" - Apéndice C - Protocolo HTTP" - "Apéndice E - Funciones de la Librería Ethernet**".

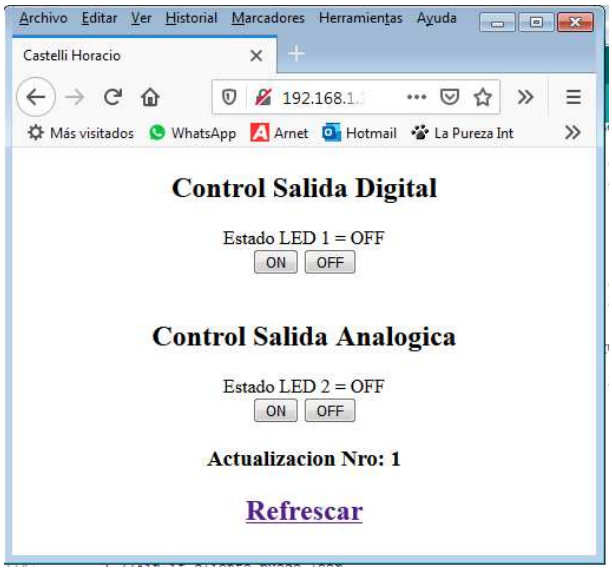
(Programa "670_Ethernet_W5100_03_Prendo_2_Led_Con_Log")

```
1   #include<SD.h>          // Esta Librería se encuentra en el IDE de Arduino
2   #include <SPI.h>         // Esta Librería se encuentra en el IDE de Arduino
3   #include <Ethernet.h>  // Esta Librería se encuentra en el IDE de Arduino
4
5   const int Pin_Habilta_W5100 = 10,
6           Pin_Habilta_SD           =  4;
7
8   const int Pin_Lectora_SD = 7; // Valores Posibles 2,3,5,6,7,8,9
9   File Archivo_Ejemplo;           // Variable del tipo Archivo
10
11  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
12  IPAddress ip(192, 168, 1, 150); // 192.168.1.150
13  EthernetServer server(80);
14
15  const int pinLed1 =    3,
16              pinLed2 = A0;
17
18  EthernetClient client;
19  bool ActualLineaVacia;
20  String Cadena,
21           Comando;
22  char c;
23  int Posicion,
24      Cuenta_Actual = 0;
25
26  /*****************************************************************************/
27  /***** Función Calcula Tiempo desde Inicio de Arduino ************************/
28  unsigned long totalSeconds, minutes, seconds;
29  unsigned int hours;
30  int dias;
31
32  String Calcula_Tiempo( ){
33     String TCadena;
34     totalSeconds =  millis( )/1000;            // Recordar que % calcula el resto de la división entera
35     hours        =  totalSeconds / 3600;       // Guarda Cantidad de Horas Transcurridas
36     minutes      = (totalSeconds % 3600) / 60;  // Descarta Horas y Guarda Minutos
37     seconds      =  totalSeconds % 60;       //Descarta todo menos los segundos que no completan el minuto
38     dias         =  (int) hours / 24;
39
40     TCadena = dias < 24 ? "0" : "";
41     TCadena += dias;
42     TCadena += ":";
43     TCadena += hours < 10 ? "0" : "";
44     TCadena += hours;
45     TCadena += ":";
46     TCadena += minutes < 10 ? "0" : "";
47     TCadena += minutes;
48     TCadena += ":";
49     TCadena += seconds < 10 ? "0" : "";
50     TCadena += seconds;
51     return(TCadena);
52  }
53  /*****************************************************************************/
54
55  void setup( ){
56    Serial.begin(9600);
57    while (!Serial);    // Esperamos que el puerto serie este abierto.
58    Serial.println("Iniciando Programa '670_Ethernet_W5100_03_Prendo_2_Led_Con_Log'");
59
60    pinMode( Pin_Habilta_W5100, OUTPUT);
61    pinMode( Pin_Habilta_SD, OUTPUT);
```

```
62
63    Serial.println(F("Iniciando Modulo Lectora/grabador de Tarjetas SD ..."));
64    Activar_SD( );
65    if( !SD.begin( Pin_Lectora_SD ) ){
66         Serial.print( F("Error al iniciar (Verifique Micro SD)") );
67         return(0);
68    }
69    Cadena = "Lectora/grabador SD, Iniciado Correctamente - Usando Pin ";
70    Cadena +=Pin_Lectora_SD;
71    Serial.println( Cadena );
72    Graba_Log( Cadena );
73
74    Activar_W5100( );
75    Ethernet.begin(mac, ip);
76    server.begin( );
77
78    Cadena = "Servidor esta en Direcion IP: ";
79    for ( Posicion = 0; Posicion < 4; Posicion++){
80       Cadena += Ethernet.localIP( )[Posicion];
81       if (Posicion < 3) Cadena += ".";
82    }
83    Serial.print( Cadena );
84    Graba_Log( Cadena );
85
86    pinMode(pinLed1, OUTPUT);
87    pinMode(pinLed2, OUTPUT);
88
89    digitalWrite(pinLed1, LOW);
90    digitalWrite(pinLed2, LOW);
91  }
92
93  void loop( ){
94    client = server.available( );
95    if (client){
96       Serial.println("Nuevo Cliente");
97       ActualLineaVacia = true;
98       Cadena = "";
99       while (client.connected( )){
100        if (client.available( )){
101           c = client.read( );
102           Serial.write(c);
103 /*****************************************************************************************/
104 /***************** Identificación del comando (prender o apagar cada led) ***************/
105           if ( Cadena.length( )<50){
106              Cadena.concat(c);
107
108              Posicion = Cadena.indexOf("Led_");      //Encuentra la Posición de la cadena "Led_"
109              Comando = Cadena.substring(Posicion); // Copia la cadena desde Posición encontrada
110              // Si no encontró la cadena Buscada, en Posición Guardara -1 y no copia nada
111
112 // Cuando encuentro Campo "Led_" solo queda identificar numero entre 01 y 99 y prender  o apagar
113 // el led correspondiente
114              if (Comando == "Led_01=0"){
115                 digitalWrite(pinLed1, LOW);
116              }else if (Comando == "Led_01=1"){
117                 digitalWrite(pinLed1, HIGH);
118              }else if (Comando == "Led_02=0"){
119                 digitalWrite(pinLed2, LOW);
120              }else if (Comando == "Led_02=1"){
121                 digitalWrite(pinLed2, HIGH);
122              }
            }
 /************** Final de la identificación de comando ************************************/
 /*****************************************************************************************/
            // Al recibir linea en blanco, servir página a cliente
```

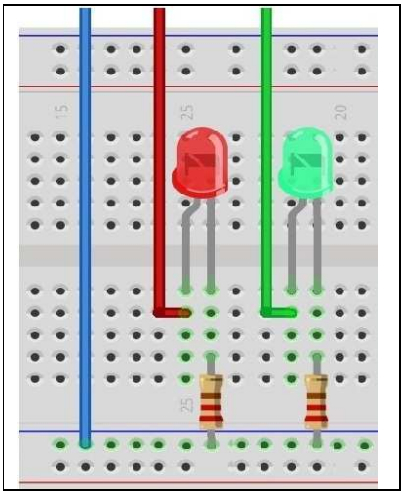
```
          if (c == '\n' && ActualLineaVacia){
             Cuenta_Actual++;
             client.println(F("HTTP/1.1 200 OK\nContent-Type: text/html"));
             client.println( );    //Una línea vacía indica que toda la meta-información ha sido enviada.

             client.println(F("<html>\n<head>"));
             client.println(F("<title>Castelli Horacio</title>"));
             client.println(F("<meta http-equiv= refresh content=15>")); //Actualización Automática de Página
             client.println(F("</head>\n<body>"));
             client.println(F("<div style='text-align:center;'>"));
/*********************************************************************************/
/************** Generación de botones y envío de comandos para cada led *********************/
             client.println(F("<h2>Control Salida Digital</h2>"));
             client.print(F("Estado LED 1 = "));
             client.println(digitalRead(pinLed1) == LOW ? "OFF" : "ON");
             Graba_Log(Cuenta_Actual, 1, digitalRead(pinLed1) == LOW ? "OFF" : "ON");
             client.println(F("<br/>"));
             client.println(F("<button onClick=location.href='./?Led_01=1'>ON</button>"));
             client.println(F("<button onClick=location.href='./?Led_01=0'>OFF</button>"));
             client.println(F("<br/><br/>"));

             client.println(F("<h2>Control Salida Analogica</h2>"));
             client.print(F("Estado LED 2 = "));
             client.println(digitalRead(pinLed2) == LOW ? "OFF" : "ON");
             Graba_Log(Cuenta_Actual, 2, digitalRead(pinLed2) == LOW ? "OFF" : "ON");
             client.println(F("<br/>"));
             client.println(F("<button onClick=location.href='./?Led_02=1'>ON</button>"));
             client.println(F("<button onClick=location.href='./?Led_02=0'>OFF</button>"));
             client.println(F("<br/>"));
/*********************************************************************************/
/***************** Contamos las actualizaciones y se muestran en la pagina HTML ***************/
             client.print(F("<h3> Actualizacion Nro: "));
             client.print(Cuenta_Actual);
             client.println(F("</h3>"));
/*********************************************************************************/
/************ Actualización Manual de la Página HTML ************************************/
             client.print(F("<a href='http://"));
             client.print(Ethernet.localIP( ));   //Entrega el IP que tenia la pagina
             client.println(F("'><h2>Refrescar<h2></a>"));
/*********************************************************************************/
             client.println(F("</div>\n</body></html>"));
             Graba_Log("\n_____________________________________________\n");
             break;
          }
          if (c == '\n'){
             ActualLineaVacia = true;
          }else if (c != '\r'){
             ActualLineaVacia = false;
          }
       } //Fin if cliente puede leer
    } // Fin while(cliente conectado
    delay(10);
    client.stop( );
  } // final if si hay cliente conectado
} //Final de la función Loop
/*********************************************************************************/
int Activar_W5100(void){
    digitalWrite(Pin_Habilta_SD, HIGH);       //Valor HIGH Deshabilita
    digitalWrite(Pin_Habilta_W5100, LOW);   // Valor LOW Habilita
    return(1);
}
/*********************************************************************************/
int Activar_SD(void){
    digitalWrite(Pin_Habilta_W5100, HIGH);    // Valor HIGH Deshabilita
    digitalWrite(Pin_Habilta_SD, LOW);          // Valor LOW Habilita
```

```
    return(1);
}
/*****************************************************************************************/
int Graba_Log(int NroActualizacion, int NroLed, char Estado[]){
  String Linea = "Actualizacion Nro: ";
  Linea += NroActualizacion;
  Linea += " - Led Numero: ";
  Linea += NroLed;
  Linea += " --> ";
  Linea += Estado;
  return( Graba_Log(Linea));
}
/*****************************************************************************************/
int Graba_Log(String Linea){
  String Renglon = Calcula_Tiempo( );
  Renglon += " !!_";
  Renglon += Linea;
  Activar_SD( );
  Archivo_Ejemplo = SD.open("Sistema.log", FILE_WRITE); // Abriendo Archivo para Escribir
  Archivo_Ejemplo.println( Renglon );
  Archivo_Ejemplo.close( ); // Cerrando archivo
  Activar_W5100( );
}
```




```
<html> <head>
<title>Castelli Horacio</title>
<meta http-equiv= refresh content=15>
</head>
<body><div style='text-align:center;'>
<h2>Control Salida Digital</h2>
Estado LED 1 = OFF
<br/>
<button onClick=location.href='./?Led_01=1'>ON</button>
<button onClick=location.href='./?Led_01=0'>OFF</button>
<br/><br/>
<h2>Control Salida Analogica</h2>
Estado LED 2 = OFF
<br/>
<button onClick=location.href='./?Led_02=1'>ON</button>
<button onClick=location.href='./?Led_02=0'>OFF</button>
<br/>
<h3> Actualizacion Nro: 1</h3>
<a href='http://192.168.1.150'><h2>Refrescar<h2></a>
</div>
</body> </html>
```

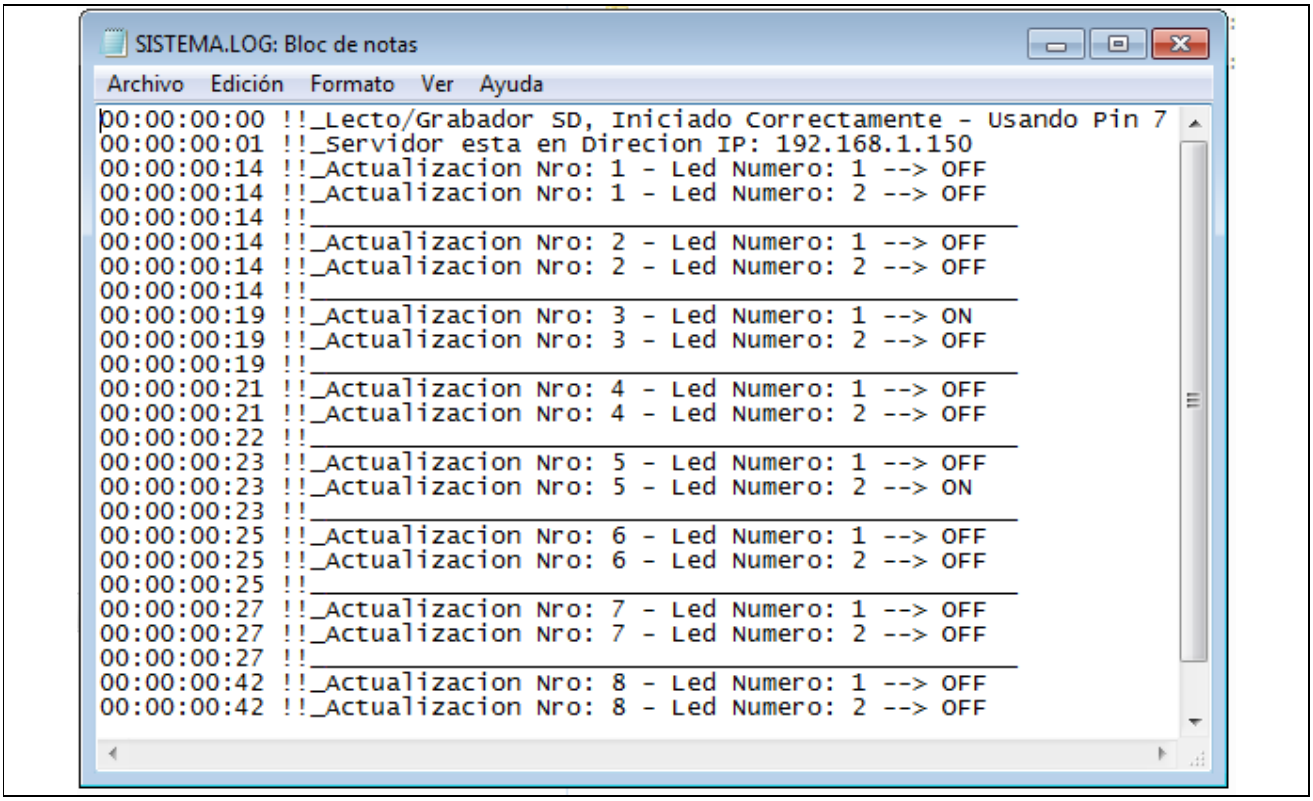
CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Placa Protoboard - 2 LED - 2 Resistencia de 470Ω - 3 Cables Macho-Macho - 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Tarjeta Micro SD con adaptador (Máx. 16Gb) 1 Placa Arduino y 1 Cable USB -

Los cables deberán ser conectados: Antes de comenzar a conectar los cables, recomiendo fijarse bien como están ahora los pines, ya que esta conectada la Placa Arduino y el modulo **Ethernet Shield W5100.**
Como podrá observar, los pines del modulo W5100, mantienen la numeración que tiene Arduino, así que simplemente conectaremos a cada pin, tal como lo hacíamos antes.
Comencemos entonces. Primero conectar el cable USB a la PC y el cable Ethernet a un router o modem. Luego, Cable Azul a GND. Cable

Rojo al Pin 3. **Cable Verde** al Pin A0.
Y en el Archivo "Sistema.log", donde grabamos todo lo sucedido durante la ejecución del programa, encontraremos lo siguiente:



Sugiero que repases la guía correspondiente a "Lector / Grabador de Tarjetas SD y Micro SD" Ya que todo la parte de programación Aprendida en esa guía, será usado en esta. También debo aclarar que **los programas funcionan acá y sin modificaciones**.

8) Este Programa Verifica La Existencia de un Archivo, luego intenta Crearlo y finalmente intenta Borrarlo. Emitiendo en todos los casos los mensajes que corresponden.

La finalidad de este ejemplo es mostrar la simplicidad con la que podemos trabajar archivos, que luego nos resultara muy útil a la hora de manejar dispositivos complejos.
(Programa "700_Lector_Grabador_Sd_05_Acciones_Con_Archivos_02")

```
1  #include <SPI.h>
2  #include <SD.h>
3
4  const int PinLectoGrabadora = 7;
5  String NombreArchivo = "Ejemplo.txt";
6  File MiArchivo;
7
8  /*********************************************************************/
9  const int Pin_Habilta_W5100  = 10,
10           Pin_Habilta_SD       = 4;
11 int Activar_SD(void){
12    digitalWrite(Pin_Habilta_W5100, HIGH);    // Valor HIGH Deshabilita
13    digitalWrite(Pin_Habilta_SD, LOW);        // Valor LOW Habilita
14 }
15 /*********************************************************************/
16 void setup( ) {
17   Serial.begin(9600);
18    while (!Serial) {
19     ; // Esperando se conecte el Puerto serie
```

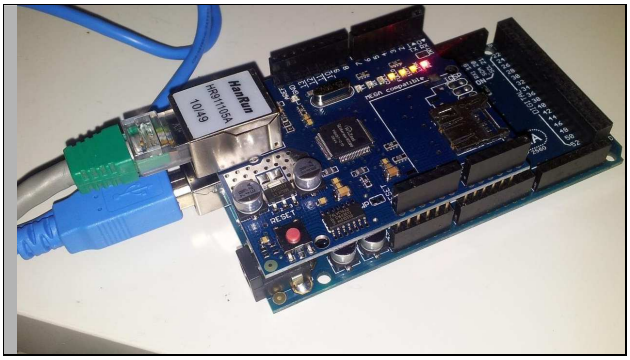
Puede consultar el "**Apéndice A - Librería "SD.h" Clases y Métodos Disponibles**". En la Guía correspondiente a "Lectora/Grabadora Tarjetas SD (Grabación y Lectura de Archivos)"

ME aseguro que el dispositivo W510 este realmente desactivado y la Lectora/Grabadora este Achicada ahora y durante el resto del programa.

Programo Acciones y comandos en la Función **Setup( )** para que se ejecuten solo una vez.

```
20      }
21    Serial.print("Inicializando Tarjeta SD...");
22    Activar_SD();
23    if (!SD.begin(PinLectoGrabadora)) {
24      Serial.println("Falla de Inicialización!");
25      while (1);
26     }
27     Serial.println("inicializacion OK!.");
28     Serial.println(" ");
29
30    if ( SD.exists( NombreArchivo ) ){
31      Serial.println("Archivo " + NombreArchivo + " SI EXISTE");
32     } else {
33      Serial.println("Archivo " + NombreArchivo + " NO EXISTE");
34     }
35     Serial.println(" ");
36
37     Serial.println("Creando Archivo " + NombreArchivo);
38     if ( !SD.exists( NombreArchivo ) ){
39       MiArchivo = SD.open( NombreArchivo, FILE_WRITE);
40       MiArchivo.close( );
41       Serial.println("Archivo " + NombreArchivo + " CREADO Exitosamente.!");
42     }else{
43       Serial.println("ERROR. El Archivo " + NombreArchivo + " Ya Existía.!");
44     }
45     Serial.println(" ");
46
47     Serial.println("Borrando Archivo " + NombreArchivo );
48     if ( SD.exists( NombreArchivo ) ) {
49       SD.remove( NombreArchivo );
50       Serial.println(NombreArchivo + " Borrado Exitosamente!.");
51     }else{
52       Serial.println("Error...No se Encontró el Archivo");
53     }
54     Serial.println(" ");
55   }
56  void loop( ) {
57    // Ninguna acción por ahora.
58   }
```

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 modulo **Ethernet Shield W5100** - cable Ethernet para conectarnos a nuestro Modem, router o switch, 2 Metros, 1 Tarjeta Micro SD con adaptador (Máx. 16Gb) 1 Placa Arduino y 1 Cable USB -

Los cables deberán ser conectados: En este proyecto, no hay cables que conectar, salvo el cable USB a la PC y conectar el cable Ethernet a un router o modem, y con eso estamos listos para programar.

Apéndice A - La Dirección MAC

En las redes, la dirección MAC (siglas en inglés de **media access control**; en español "**control de acceso al medio**") es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física, y es única para cada

dispositivo. Está determinada y configurada por el IEEE (los últimos 24 bits) y el fabricante (los primeros 24 bits) utilizando el organizationally unique identifier.

La mayoría de los protocolos que trabajan en la capa 2 del modelo OSI usan una de las tres numeraciones manejadas por el IEEE: MAC-48, EUI-48, y EUI-64, las cuales han sido diseñadas para ser identificadores globalmente únicos. No todos los protocolos de comunicación usan direcciones MAC, y no todos los protocolos requieren identificadores globalmente únicos.

Las direcciones MAC son únicas a nivel mundial, puesto que son escritas directamente, en forma binaria, en el hardware en su momento de fabricación.

En el caso del shield Ethernet, la MAC no está incluida en el integrado que implementa la pila de protocolos TCP/IP, sino que nosotros debemos configurar desde el programa.

La Dirección MAC asociada a nuestro **Ethernet Shield W5100, que debemos usar en nuetro programa es:** { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }.

Y podemos declararla así:
```
byte mac[ ] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

Lo normal es que no nos de problemas la habitual, pero si debes usar más de un Shield, necesitas cambiar al menos una. Tiene que ser diferentes para cada dispositivo. Acá dejo otra por si te hace Falta.

Y podemos declararla así:
```
byte mac2[ ] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
```

¿Qué diferencia hay entre dirección IP y dirección MAC?

La dirección IP es diferente a la dirección MAC, porque esta última es un identificador de 48 bits para identificar de forma única la tarjeta de red y no depende del protocolo de conexión utilizando la red.

Apéndice B - Dirección IP

Una dirección IP, en el caso específico de la informática, se trata de una expresión compuesta por letras y/o números que alude a una localización en la memoria de un equipo informático. IP, por su parte, es la sigla inglesa que alude a "**Internet Protocol**" (“Protocolo de Internet”).

Dicho de otra forma, una dirección IP, es un número que identifica (Dirección), de manera lógica y jerárquica, a una Interfaz en red (elemento de comunicación/conexión) de un dispositivo (computadora, tableta, portátil, teléfono inteligente, etc.) que utilice el protocolo o (Internet Protocol), que corresponde al nivel de red del modelo TCP/IP.

Puede decirse que la dirección IP es la identificación que posibilita a los dispositivos informáticos encontrarse y conectarse entre sí.
Aquellos que necesitan una conexión permanente requieren una dirección IP estática (fija) para que puedan ser localizados en la red: un servidor que aloja un sitio Web, un servidor de correo electrónico, etc.

Para que los usuarios puedan recordar la localización de los sitios en Internet, se utilizan nombres de dominio que están vinculados a las direcciones IP.

Un equipo al que se le asigna una dirección IP dinámica, en cambio, no cuenta con un número fijo. Por eso la identificación en cuestión cambia de manera periódica.

Puede entenderse a la dirección IP, en definitiva, como una etiqueta que identifica a la interfaz de un dispositivo en una red que se rige por el protocolo IP. Estas identificaciones, en la actualidad, se

componen de cuatro bloques numéricos, cada uno formado por números del 0 al 255. Por ejemplo: 205.45.128.30 podría ser una dirección IP.

Además de todo lo establecido, no podemos pasar por alto otra serie de singularidades y aspectos importantes sobre la mencionada dirección IP como son los siguientes:
-Nadie puede proceder a navegar por la Red sin una IP.
-De la misma manera, hay que establecer que ninguna página Web puede estar online correctamente si no está asociada a una IP.
-En ocasiones, suele confundirse la IP con la MAC. No obstante, esta última es la identificación que se le otorga de manera individual a las llamadas tarjetas de red.

Además de la dirección IP pública tenemos que establecer que también existen las llamadas IP privadas. Estas son las que se utilizan, por ejemplo, cuando en una casa se procede a conectar varios dispositivos a una misma red Wifi. En ese caso, cada uno de esos (móviles, tablet, ordenador…) cuenta con su propia IP.

No menos relevante es saber que en cuanto a esa numeración podemos destacar que hay miles de combinaciones. No obstante, no es menos cierto que hay tres rangos que se reservan de forma exclusiva para lo que son las IP privadas:

- Clase A, de 10.0.0.0. a 10.255.255.255, que está reservada para lo que son las redes de gran tamaño como pueden ser las grandes multinacionales.
- Clase B, de 172.16.0.0. a 172.31.255.255, que se usa para lo que son redes privadas de tamaño mediano como puede ser, por ejemplo, la red de una universidad.
- Clase C, de 192.168.0.0. a 192.168.255.255, que se emplea para lo que son redes privadas más pequeñas, como las de tipo doméstico.

¿Qué diferencia hay entre dirección IP y dirección MAC?

La dirección IP es diferente a la dirección MAC, porque esta última es un identificador de 48 bits para identificar de forma única la tarjeta de red y no depende del protocolo de conexión utilizando la red.

Si quieres saber tu IP (Privada) o el de Alguna URL en Particular, puedes leer "**Apéndice C - Ventana de Comandos o Símbolo del Sistema**" que se encuentra en la guía sobre **Lectora/Grabadora Tarjetas SD (Grabación y Lectura de Archivos)**.

Para conocer tu IP Publica, puedes visitar esta pagina "www.cual-es-mi-ip.net/"

Apéndice C - Protocolo HTTP

HTTP (Hypertext Transfer Protocol) es el protocolo que permite la transferencia de información a través de la web. Este protocolo fue lanzado en 1991 y desde ahí ha ido evolucionando hasta llegar a la versión es más ampliamente conocida la 1.1.

1991- HTTP 0.9 Al inicio de la historia el protocolo HTTP solo permitía realizar peticiones sin especificar el verbo, es decir solo se podían hacer peticiones GET.
1996- HTTP 1.0 Luego se mejoraron las cosas y se agregó el soporte a algunos verbos como GET, POST y HEAD, se implementó los códigos de estado HTTP entre otras muchas mejoras.
1999/2000- HTTP 1.1 Ya en la versión 1.1 teníamos los verbos GET, POST, PUT, DELETE, etc y la web se empezaba a orientar a recursos (REST), teníamos las cabeceras en las peticiones, etc. Sobre esta última especificación se empezó a desarrollar HTTP1.2 pero luego termino convirtiéndose en

una extensión de la versión 1.1, lo que tienen en común todas estas versiones es que tanto las respuestas como las peticiones se realizan a través de texto plano.

¿Qué es HTTP/2?
HTTP/2 es un protocolo binario que conserva la misma semántica que el protocolo HTTP1.X lo que significa que todos los verbos, cabeceras, etc. siguen funcionando sin cambios. De hecho, HTTP/2 busca resolver los defectos que tiene la comunicación a través TCP (la capa de transporte dentro del protocolo HTTP).

Muchos consideran a HTTP/2 el reemplazo del protocolo SPDY que desarrollo Google para mejorar el rendimiento de sus servicios en su navegador Chrome, de hecho el protocolo HTTP/2 está basado en algunas de las ideas del protocolo SPDY, el cual actualmente se considera obsoleto pues se ha apostado completamente por el protocolo HTTP/2.

En octubre de 2015 se anunció que IIS en Windows 10 añadió soporte para HTTP/2. Y dado que Windows 10 ya está disponible, el soporte para HTTP/2 está presente en Windows 10 y Windows Server 2016 Technical Preview.

IIS actualmente soporta HTTP/2 sólo a través de TLS. Al realizar una conexión HTTPS a un servidor web que se ejecuta IIS en Windows 10, HTTP/2 se utiliza si el cliente y el servidor lo soportan.

En base a lo anterior, lo único que necesitamos además de Windows 10 o Windows Server 2016 TP es que nuestro sitio se ejecute a través de TLS.

Comandos del protocolo HTTP

DELETE	Pedir al servidor que elimine un documento.
GET	Pedir al servidor un documento.
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Enviar documento al servidor (datos de cumplimentación de un formulario, pe).
PUT	Pedir al servidor que haga accesible el documento que se le envía en una URL determinada.
TRACE	Obtener del servidor copia de la petición que le llega.

-o-

Puedes Ampliar tu conocimiento en el tema, lo que te será muy útil en próximos proyectos. Ver Guia Anexo sobre Protocolo HTTP.

Apéndice D - PUERTOS DE COMUNICACION

El ordenador tiene muchas entradas, esas entradas se inventaron para cada cosa especifica tenga su lugar de estrada y no tengan que pasaran todas juntas y desordenadamente por una única puerta, además, Asignando puertas a cada canal de datos se puede controlar que entradas dejar abiertas y cuales no (estos controles son los llamados **Firewall**.... Te suena?).

Así es que a cada puerta (puerto) se le asigna un numero y por esa puerta solo van a pasar ciertos tipos de datos. Estos puedes ser del tipo SMTP, FTP, TELNET o HTTP, etc.). A continuación dejo algunos de los números de puertos mas usados:

El puerto 80 esta reservado para HTTP, con el que accedemos a las paginas webs no seguras (Deben ser controlados). Todas aquellas aplicaciones que funcionan en base a una dirección IP (por TCP o UDP) establecen comunicación con un servidor específico a través de un puerto, en el caso del HTTP.

El puerto 23 es el puerto por defecto para Telnet.
El puerto 8080 se hizo para que ciertos programas (con permisos especiales) puedan acceder a Internet sin tantos controles.
El puerto 5000 es el encargado de los dispositivos que se conectan por **USB**.

Apéndice E - Funciones de la Librería Ethernet

(Ver más)

La librería Ethernet se compone de 5 clases, cada una con sus métodos

FUNCIONES (Métodos) DE LA CLASE "Ethernet" Inicializa la librería Ethernet y las configuraciones de red.	
Ethernet.begin( )	Inicializa la librería Ethernet (Constructor). Es un método sobrecargo, y las opciones son: Ethernet.begin(mac); Ethernet.begin(mac, ip); Ethernet.begin(mac, ip, dns); Ethernet.begin(mac, ip, dns, gateway); Ethernet.begin(mac, ip, dns, gateway, subnet); **Los Parámetros son datos del dispositivo que estamos usando:** mac: MAC del dispositivo. Ver "Apéndice A - La Dirección MAC" ip: ip del dispositivo. Ver en esta guía "Apéndice B - Dirección IP" Solo la versión DHCP de la función, **Ethernet.begin(mac)**, devuelve un int: 1 cuando la conexión DHCP es exitosa y un 0 (cero) en caso de que la conexión falle. **Las otras versiones de esta función no devuelven nada.**
localIP( )	Obtiene la dirección IP. automáticamente cuando usamos DHCP
maintain( )	Solicita una renovación al servidor DHCP

-o-

FUNCIONES (Métodos) DE LA CLASE "IPAddress" Trabaja con IPs locales y remotas. Facilita el trabajo con direcciones IPs.	

IPAddress( )	Define una dirección IP. Se puede usar para declarar direcciones locales y remotas. Ver ejemplo.
	
	`#include <SPI.h>`
	`#include <Ethernet.h>`
	
	`byte mac[ ] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };`
	
	`IPAddress dnServer(192, 168, 0, 1);`
	`IPAddress gateway(192, 168, 0, 1);`
	`IPAddress subnet(255, 255, 255, 0);`
	`IPAddress ip(192, 168, 0, 2);`
	
	`void setup( ) {`
	`  Serial.begin(9600);`
	
	`  Ethernet.begin(mac, ip, dnServer, gateway, subnet);`
	
	`  Serial.print("IP = ");`

```
  Serial.println(Ethernet.localIP( ));
}

void loop( ) {
// Sin Acciones para repetir
}
```

-o-

FUNCIONES (Métodos) DE LA CLASE "Server"

La clase Servidor crea servidores que pueden enviar y recibir datos de clientes conectados (programas que se ejecutan en otras computadoras o dispositivos).

Server( )	Constructor de la clase server. **No se usa directamente**. El servidor es la clase base para todas las llamadas basadas en el servidor Ethernet. **No se llama directamente**, sino que se invoca cada vez que utiliza una función que se basa en ella.
EthernetServer( )	Crea un servidor que escucha por las conexiones entrantes del puerto definido. En el código se usaría así: `EthernetServer server(80);`

server.begin( )	Le dice al servidor que comience a escuchar las conexiones entrantes. Recordar que usamos el método del objeto creado.
	`EthernetServer server(80);`
	`server.begin( )`

server.available( )	Devuelve el cliente que esté conectado al servidor y tiene datos disponibles a leer. Recordar que la conexión persiste cuando el objeto de cliente devuelto queda fuera de alcance; puede cerrarlo llamando a client.stop( ).
	`EthernetServer server(80);`
	`EthernetClient client = server.available( );`

server.write( )	Escribir datos a todos los clientes conectados a un servidor. Estos datos se envían como un byte o serie de bytes. Hay dos formas de usar estas funciones, y la diferencia esta en los datos que se le entregan: 1) server.write(val) 2) server.write(buf, len) Y los parámetros o datos entregados son: **val:** valor que se envía como un simple byte (byte o char) **buf:** un array que se envía como serie de bytes (byte o char) **len:** Es la longitud del buffer
server.print( )	Imprime datos a todos los clientes conectados a un servidor. Imprime los números como una secuencia de dígitos, cada uno con un carácter ASCII (por ejemplo, el número 123 se envía como los tres caracteres '1', '2', '3'). Hay dos formas de usar estas funciones, y la diferencia esta en los datos que se le entregan: `server.print(dato)` | `server.print(dato, BASE)` Y los parámetros o datos entregados son: **dato:** simplemente el dato a imprimir, que debe ser char, byte, int, long, o string. **BASE:** (opcional): la base en la cual se debe imprimir el numero: BIN para binario (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

	Esta función, en cualquiera de sus formatos, retorna el número de bytes escritos, aunque leer ese número es opcional int r = server.print(dato)
println( )	Escribe datos a todos los clientes conectados al servidor seguido de una nueva línea. Imprime los números como una secuencia de dígitos, cada uno con un carácter ASCII (por ejemplo, el número 123 se envía como los tres caracteres '1', '2', '3'). Hay tres formas de usar estas funciones, y la diferencia esta en los datos que se le entregan: *server*.println( ) | *server*.println(dato) | *server*.println(dato, BASE) Y los parámetros o datos entregados son: **dato:** simplemente el dato a imprimir, que debe ser char, byte, int, long, o string. **BASE:** (opcional): la base en la cual se debe imprimir el numero: BIN para binario (base 2), DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16). Esta función, en cualquiera de sus formatos, retorna el número de bytes escritos, aunque leer ese número es opcional int r = server.println(dato)
NOTA: Cuando se crea un servidor con la clase Server, dejo un puerto escuchando peticiones por ese puerto. En el caso que entre una nueva petición, esta queda en el buffer. Cuando el buffer tiene datos, llamo a la función server.available( ) que devuelve un cliente (de la clase client) que está conectado al servidor y está disponible para leer.	

-o-

FUNCIONES (Métodos) DE LA CLASE "Client"

Crea un cliente que se conecta a un servidor y puede mandar y recibir datos. (Ver Mas)

Client	Constructor de la clase client. No se usa directamente
EthernetClient( )	Crea un cliente que puede conectarse a una dirección IP y puerto de Internet especificados (definidos en la función client.connect ( )
if (EthernetClient)	Indica si el cliente Ethernet está preparado. Y retorna verdadero si el cliente especificado está disponible.
connected( )	Devuelve verdadero si el cliente está Conectado. Pero, tenga en cuenta que un cliente se considera conectado, si la conexión se ha cerrado pero todavía hay datos no leídos.
connect( )	Se conecta a una dirección IP y puerto especificados. El valor de retorno indica éxito o fracaso. También admite búsquedas DNS cuando se usa un nombre de dominio. Esta función dispone de tres opciones, en las que solo difieren los parámetros que recibe. client.connect( ) client.connect(ip, port) client.connect(URL, port) **Parámetros** **ip:** la dirección IP a la que se conectará el cliente (matriz de 4 bytes) **URL:** el nombre de dominio al que se conectará el cliente (cadena, por ej .:"www.castelli.com.ar") **port:** el puerto al que se conectará el cliente (int).

<table>
<tr><td></td><td>Cualquiera de las las tres opciones, retornará un valor entero, y los posibles valores son: 1,-1,-2,-3,-4. Valor que indican el estado de la conexión:

<table>
<tr><td>1</td><td>SUCCESS</td><td>ÉXITO</td></tr>
<tr><td>-1</td><td>TIMED_OUT</td><td>CADUCADO</td></tr>
<tr><td>-2</td><td>INVALID_SERVER</td><td>SERVIDOR INVALIDO</td></tr>
<tr><td>-3</td><td>TRUNCATED</td><td>TRUNCADO</td></tr>
<tr><td>-4</td><td>INVALID_RESPONSE</td><td>RESPUESTA INVALIDA</td></tr>
</table></td></tr>
<tr><td>write( )</td><td>Escribe datos en el servidor al que está conectado el cliente. Estos datos se envían como un byte o serie de bytes.

Podemos variar los datos enviados, aunque siempre tendremos el mismo resultado.

<table>
<tr><td>client.write(val)</td></tr>
<tr><td>client.write(buf, len)</td></tr>
</table>
Parámetros
val: un valor para enviar como un solo byte (byte o char)
buf: una matriz para enviar como una serie de bytes (byte o char)
len: la longitud del búfer

En cualquiera de los dos casos, esta función devuelve el número (tipo bytes) que representa el número de bytes escritos. No es necesario leer este valor.</td></tr>
<tr><td>print( )</td><td>Escribe datos en el servidor al que está conectado el cliente. Imprime o escribe los números como una secuencia de dígitos, cada uno cono un carácter ASCII (por ejemplo, el número 123 se envía como los tres caracteres '1', '2', '3').

Podemos variar los datos enviados, aunque siempre tendremos el mismo resultado.

<table>
<tr><td>client.print(data)</td></tr>
<tr><td>client.print(data, BASE)</td></tr>
</table>
Parámetros
datos: los datos que puede enviar o escribir son del tipo (char, byte, int, long o string)
BASE (opcional): la base en que podemos imprimir números son: DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).

En todos los casos, retorna un numero (del tipo byte) que representa el número de bytes escritos, aunque leer ese número es opcional</td></tr>
<tr><td>println( )</td><td>Escribe datos seguido de un retorno de carro y una nueva línea, en el servidor al que está conectado un cliente. Imprime los números como una secuencia de dígitos, cada uno con un carácter ASCII (por ejemplo, el número 123 se envía como los tres caracteres '1', '2', '3').

Tenemos tres formas de usar esta función, y es variando los parámetros entregados, aunque siempre tendremos el mismo resultado.

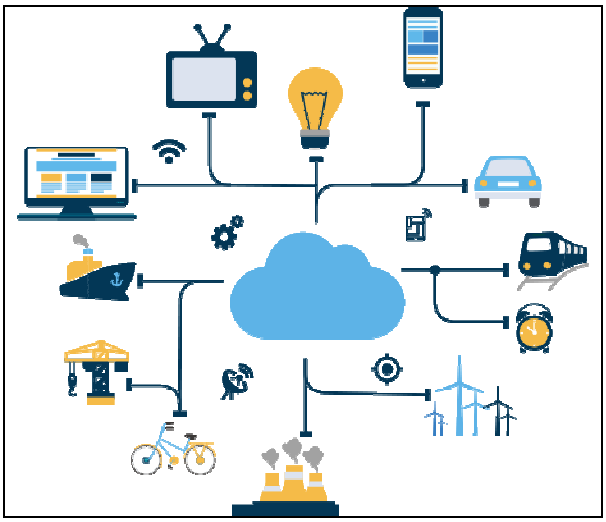
<table>
<tr><td>client.println( )</td></tr>
<tr><td>client.println(data)</td></tr>
<tr><td>client.print(data, BASE)</td></tr>
</table>
Parámetros
data(opcional): los datos a imprimir (char, byte, int, long o string)
BASE (opcional): la base para imprimir números: DEC para decimal (base 10), OCT para octal (base 8), HEX para hexadecimal (base 16).</td></tr>
</table>

	En todos los casos, retorna un numero (del tipo byte) que representa el número de bytes escritos, aunque leer ese número es opcional
available( )	Devuelve el número de bytes disponibles para leer (es decir, la cantidad de datos que el servidor al que está conectado ha escrito para el cliente).
read( )	Lee el siguiente byte recibido desde el servidor al que está conectado el cliente.
flush( )	Borrar todos los bytes que han sido escritos en el cliente pero no leídos
stop( )	Desconecta el cliente del servidor

-o-

FUNCIONES (Métodos) DE LA CLASE "EthernetUDP"	
Habilita el envío y recepción de mensajes UDP. (Ver Mas)	
begin( )	Inicializar la librería UDP
read( )	Lee datos UDP
write( )	Escribe datos UDP a la conexión remota.
beginPacket( )	Comienza una conexión para escribir paquetes UDP
endPacket( )	Finaliza una conexión UDP después de escribir
parsePacket( )	Comprueba la presencia de un paquete UDP
available( )	Devuelve el nº de bytes disponible para leer en el buffer
stop( )	Desconecta del servidor
remoteIP( )	Obtiene la IP de la conexión remota
remotePort( )	Obtiene el puerto de la conexión remota

Apéndice F - ¿Qué es IoT?



El Internet **of Things (IoT)** es un concepto que describe la red de objetos físicos (cosas) que llevan censores integrados, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet.

Estos dispositivos abarcan desde objetos domésticos cotidianos hasta sofisticadas herramientas industriales. Con más de 7.000 millones de dispositivos de IoT conectados en 2019, 10.000 millones en 2020 y los expertos prevén que este número aumentará hasta llegar a un mínimo de 22.000 millones en 2025.

En Definitiva, **el Internet de las cosas** se refiere a una interconexión digital de objetos cotidianos con Internet , pero la conexión de Internet más con objetos que con personas. También se suele conocer como Internet de todas las cosas o Internet en las cosas.

01010100 01101111 01100100 01101111 00100000 01101100 01101111 00100000 01110001
01110101 01100101 00100000 01110101 01101110 01100001 00100000 01110000 01100101
01110010 01110011 01101111 01101110 01100001 00100000 01110000 01110101 01100101

01100100 01100101 00100000 01101001 01101101 01100001 01100111 01101001 01101110
01100001 01110010 00101100 00100000 01000011 01101111 01101110 00100000 01100101
01110011 01110100 01110101 01100100 01101001 01101111 00101100 00100000 01101110
01101111 01110011 01101111 01110100 01110010 01101111 01110011 00100000 01010001
01010101 01001001 01011010 11000011 10000001 01010011 00100000 01010000 01001111
01000100 01000001 01001101 01001111 01010011 00100000 01101000 01100001 01100011
01100101 01110010 01101100 01101111 00100000 01110010 01100101 01100001 01101100
01101001 01100100 01100001 01100100 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.