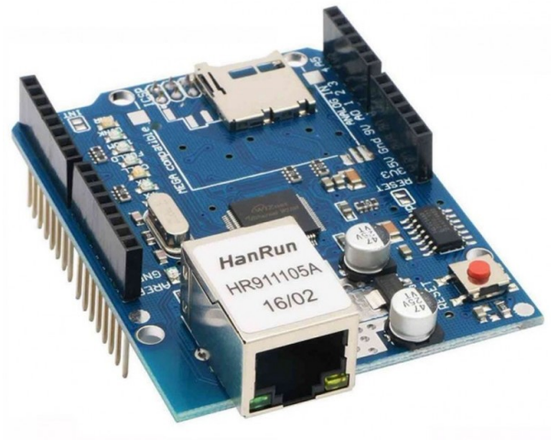
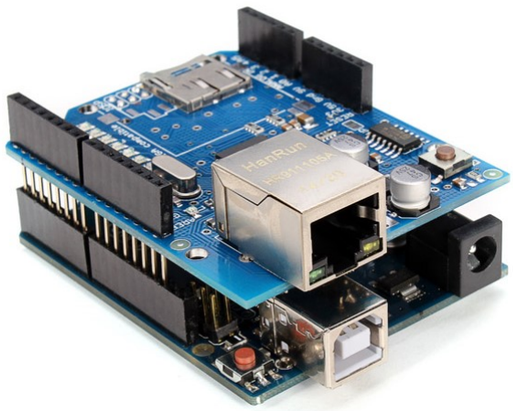
IOT - Internet de las cosas

(Versión 3-6-19)

Presentamos nuevamente la Ethernet Shield



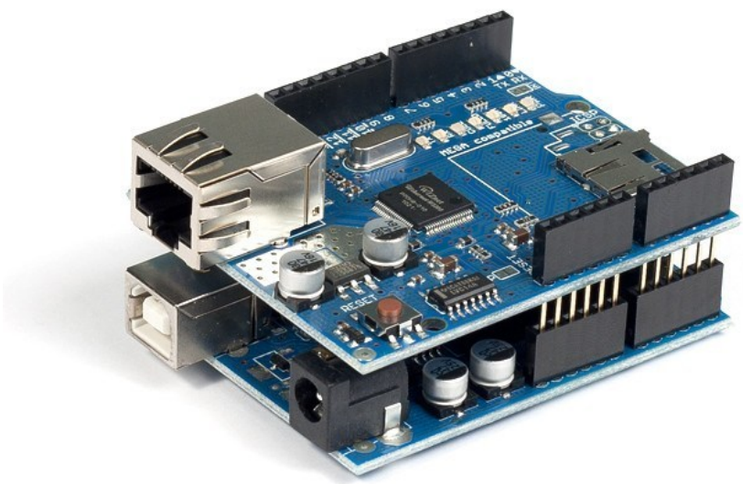
Colocada sobre Arduino Uno



A continuación se transcribe el contenido de un tutorial encontrado en Internet. El profesor Bolaños desarrollara ejemplos propios ensayados en el APENDICE de este apunte.

Cómo usar el Ethernet Shield de Arduino

Posted on abril 22, 2016 by David Cervantes Caballero



Arduino Ethernet Shield

Entre las diferentes formas de comunicarnos con un Arduino, una de ellas es utilizar un ***Shield Ethernet*** que nos permitirá que nuestro *Arduino* se comporte como un **servidor o un cliente**. A continuación os explicaremos ponerlo en funcionamiento de manera básica.

Establecer MAC y la IP

En primer lugar debemos establecer la ***MAC*** en el código de nuestro *Arduino*, dependiendo de la versión del *Shield Ethernet* puede que venga una etiqueta con la *MAC*, en caso contrario podemos poner la que queramos, con cuidado ya que no puede existir la misma *MAC* varias veces en la misma red. Por otro lado es aconsejable que configuréis vuestro *router* para que asigne una *IP* fija, en vez de asignar una *automáticamente mediante DHCP,* asociada a la *MAC* que hemos elegido previamente.

Para los ejemplos que vamos a hacer la *MAC* será ***AA:AA:AA:AA:AA:AA*** y la IP ***192.168.0.122***.

```
byte mac[] = {
 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA };
IPAddress ip(192,168,0,122);
```

Ejemplo básico de servidor con Arduino

Este es un ejemplo básico para que nuestro *Arduino* se comporte como un servidor, cuando reciba una petición devolverá una web en *HTML* con un ***Hola Mundo!***. Para probarlo simplemente tendremos que entrar en ***http://192.168.0.122/*** y veremos el texto *«Hola Mundo!»* en nuestro navegador.

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

byte mac[] = {
 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA };
IPAddress ip(192,168,0,122);

EthernetServer server(80);

void setup() {
 // Inicia la conexion Ethernet y el servidor:
 Ethernet.begin(mac, ip);
 server.begin();
}


void loop() {
 EthernetClient client = server.available();
 if (client) {
 boolean currentLineIsBlank = true;
 while (client.connected()) {
 if (client.available()) {
 char c = client.read();
 if (c == '\n' && currentLineIsBlank) {
 // Cabecera HTTP
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/html");
 client.println("Connection: close");
 client.println("Refresh: 5");
 client.println();
 client.println("<!DOCTYPE HTML>");
 client.println("<html>");
 client.println("Hola Mundo!");
 client.println("</html>");
 break;
 }
 if (c == '\n') {
 currentLineIsBlank = true;
 }
 else if (c != '\r') {
 currentLineIsBlank = false;
 }
```

```
 }
 }
 delay(1);
 client.stop();
 }
}
```

Ejemplo básico de cliente con Arduino

Si queremos que nuestro *Arduino* se conecte a una web, ya sea para extraer información o para enviar datos tendremos que usarlo como cliente. En el siguiente ejemplo veremos como Arduino se conecta a ***scidle.com*** imprimiendo en el puerto serie la respuesta del servidor, que será el *HTML* de la portada.

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


byte mac[] = { 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA };
IPAddress ip(192,168,0,122);

char server[] = "scidle.com";

EthernetClient client;

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

 // Inicia la conexion Ethernet:
 if (Ethernet.begin(mac) == 0) {
 Serial.println("Error al configurar Ethernet usando DHCP");
 Ethernet.begin(mac, ip);
 }

 delay(1000);
 Serial.println("conectando...");

 if (client.connect(server, 80)) {
 Serial.println("conectado");
 // Realiza la peticion HTTP:
 client.println("GET /search?q=arduino HTTP/1.1");
 client.println("Host: scidle.com");
 client.println("Connection: close");
 client.println();
 }
 else {
 Serial.println("Fallo en la conexion");
 }
}

void loop()
{
 // Si se ha conectado correctamente imprime la respuesta del servidor, que sera
el codigo HTML de la web
 if (client.available()) {
 char c = client.read();
 Serial.print(c);
 }

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

 while(true);
 }
}
```

Nota de Prof: Bolaños.

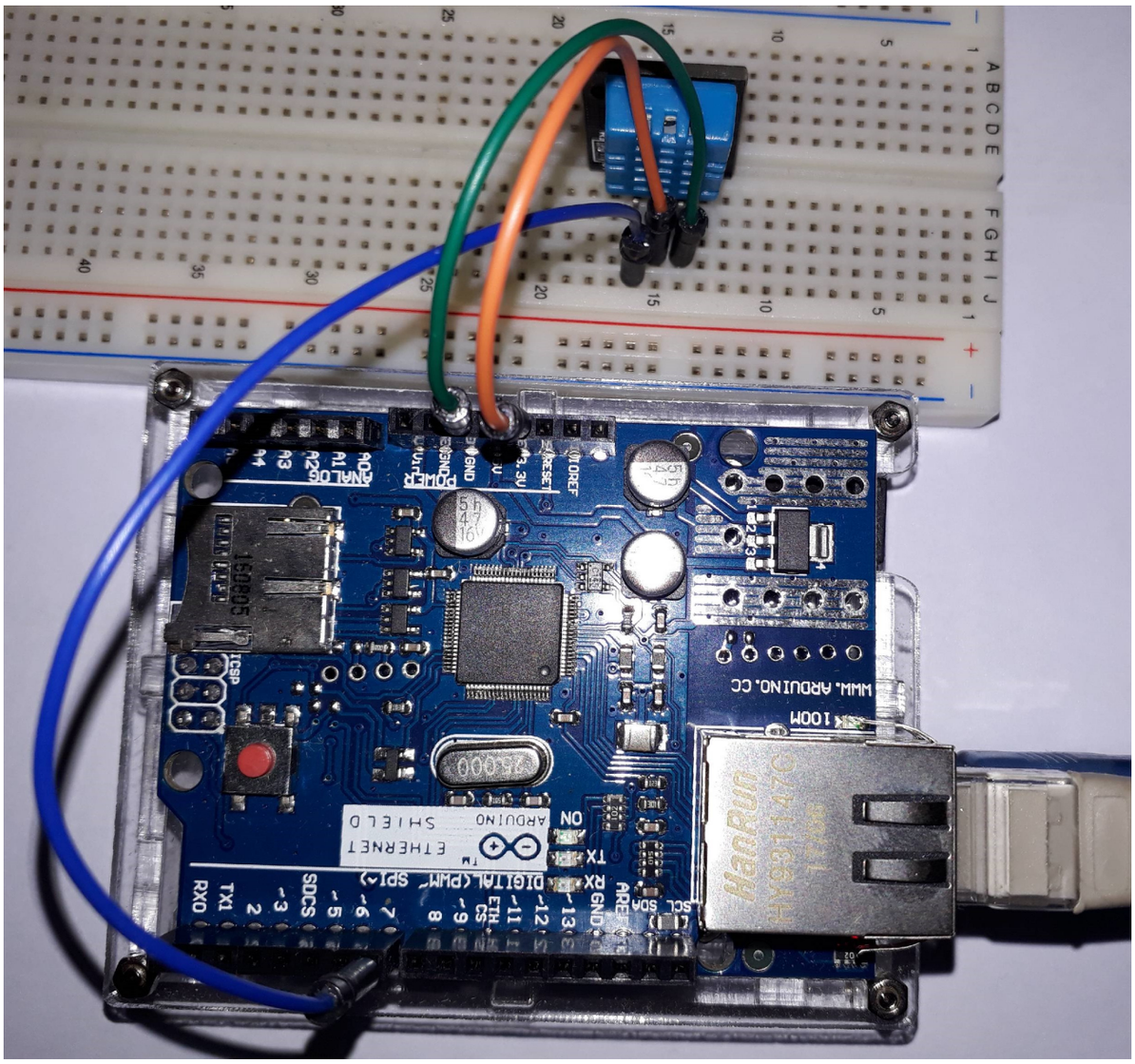
Entrando a los menues de MI ROUTER

Ethernet Shield

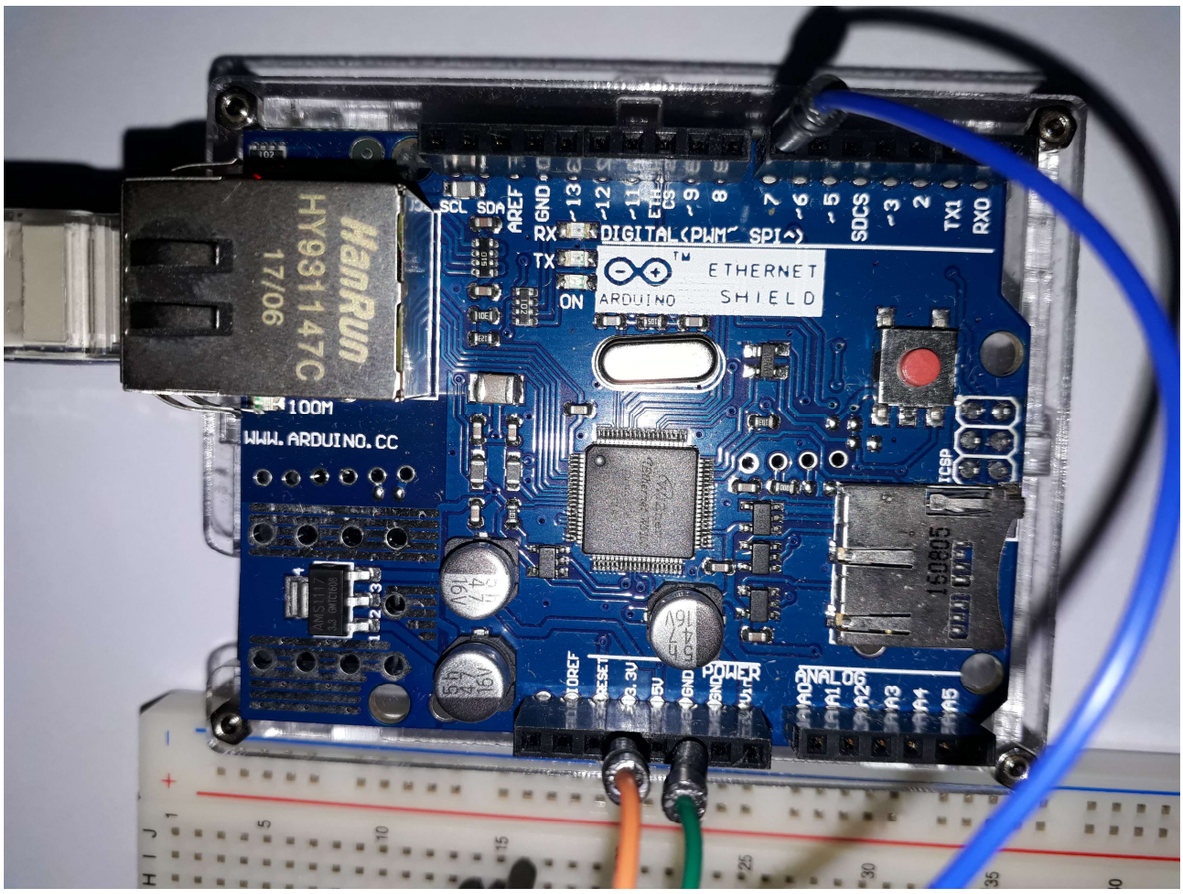
192.168.1.106	DE:AD:BE:EF:FE:ED

APENDICE - Prof: Bolaños DJ

Se armara el siguiente circuito:



Tomamos la información proveniente de un sensor DHT11 cuyos datos ingresan por el PIN 7 del Arduino Uno.



Los datos van a ser enviados a un canal de Thingspeak.

Se supone que el lector cuenta con un canal gratuito en Thingspeak

Con este objetivo el Shield Ethernet trabajara como cliente.

El programa a emplear es:

```
//Shield Ethernet como cliente prueba A1
//Este funciono satisfactoriamente 3-6-19
//Envia a Thingspeak datos de DHT11 utilizando Shield Ethernet
//ATENCION- DATOS DE MI CANAL TempHumDJB borrados

#include "DHT.h"        // including the library of DHT11 temperature and humidity sensor

#define DHTTYPE DHT11   // DHT 11


#define dht_dpin 7
DHT dht(dht_dpin, DHTTYPE);

#include <SPI.h>
#include <Ethernet.h>


byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192,168,1,106);//En nuestra RED

char server[] = "184.106.153.149";//Servidor Thingspeak
```

```
String apiKey ="xxxxxxxxxxxxxxxx"; //AQUI API KEY DE CANAL


EthernetClient client;

void setup()
{
dht.begin(); //Inicializamos el sensor
Serial.begin(9600);
delay(1000);
}

void loop()
{


delay(10000);//cada 10 segundos


 //Definimos variables tipo float para lectura
 //de temperatura y humedad y realizamos la lectura

    float hum = dht.readHumidity();
    float temp = dht.readTemperature();
//-----------------------------------------------

Serial.println("enviado dato");


enviarTempTS(temp,hum);


}

//-------------------------------------------------



//---ZONA DE FUNCIONES-----------------------

//----Funcion para Enviar Datos a Thingspeak----

void enviarTempTS(float temp,float hum)
{

  Ethernet.begin(mac, ip);

  //delay(1000);
  Serial.println("conectando...");


   if (client.connect(server, 80))

   {
   Serial.println(" Client connected ");

   String postStr = apiKey;
   postStr += "&field1=";
   postStr += String(temp);//Envia dato temperatura

   postStr += "&field2=";
   postStr += String(hum);//Envia dato huemedad
```

```
  postStr += "\r\n\r\n";

  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n");
  client.print(postStr);
  delay(1000);

  }
client.stop();

 Serial.println("uso funcion");
}
```

Obteniéndose como resultado: