

# Si. ETHERNET

Se necesita...



Módulo Serie Ethernet conectado al sistema de desarrollo EasyAVR5A

Automatización doméstica, control doméstico, casa inteligente o digital, son sólo diferentes nombres para confort, conveniencia, seguridad y ahorro de energía. Los sistemas de ahorro de energía están adquiriendo, hoy día, una mayor importancia. Incluso si pensamos que dichos sistemas son muy caros, podemos asegurar que también son bastante baratos si los fabricamos por nosotros mismos. Hay varias formas de controlar un sistema doméstico inteligente. Uno de ellos es a través de Ethernet.

Por Srdjan Tomic  
MikroElektronika – Departamento de Software

Todo lo que necesitamos es un microcontrolador Atmega 32 y un circuito integrado Ethernet serie ENC28J60. Este circuito integrado es una gran solución para otras familias de microcontroladores, tales como PIC, dsPIC etc. El conector RJ-45 CJCBA8HF1Y0 de CviLux se usa para la conexión a la red Ethernet. Un diodo LED conectado al PORTA.0 del microcontrolador, simulan una aplicación doméstica que quiere el control.

El compilador *mikroPASCAL for AVR* contiene la librería `SPI_Ethernet` que simplificará considerablemente el proceso de escritura de un programa para el microcontrolador. Usando unas pocas rutinas de esta librería, es posible crear el programa que nos permitirá controlar aplicaciones eléctricas en nuestra casa a través de un explorador web.

Para ello, es necesario realizar las siguientes operaciones dentro del programa:

**Paso 1.** Crear una página html a través de la cual arrancar el microcontrolador. Importar el código como un bloque de texto ("string").

**Paso 2.** Configurar las direcciones IP, DNS, Gateway y máscaras de Subred proporcionadas por nuestro proveedor de Internet.

Por ejemplo, nuestros parámetros locales de red son los siguientes:

**IP:** 192.168.20.60 (dirección del Sistema de Control)

**DNS:** 192.168.20.1 (dirección del Domain Name System o Sistema de Nombres de Dominio)

**GATEWAY:** 192.168.20.6 (dirección de la pasarela o Gateway)

**SUBNET:** 255.255.255.0 (máscara de Subred)

**Paso 3.** Deshabilitar las entradas analógicas de PORTA. El terminal del microcontrolador debe ser borrado y configurado como una salida.

**Paso 4.** Inicializar el módulo SPI del microcontrolador Atmega 32.

**Paso 5.** Inicializar el módulo Serie Ethernet del circuito integrado ENC28J60.

**Paso 6.** Escribir el código dentro de la función `SPI_Ethernet_userTCP` que, después de recibir el comando a través del explorador web, encenderá/apagará el diodo LED conectado al PORTA.0

**Paso 7.** Leer los datos recibidos en un bucle sin fin.

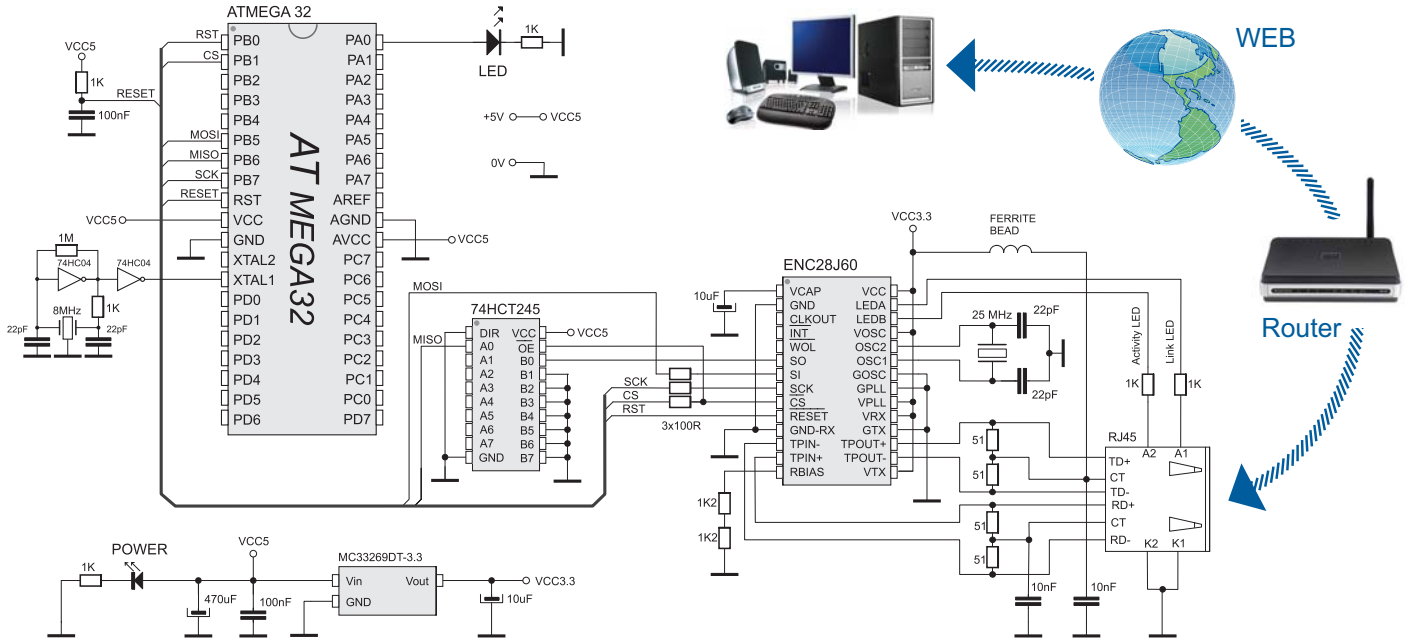
La parte más importante del programa es la función `SPI_Ethernet_userTCP`, que procesa todos los comandos recibidos.

Después de recibir la petición "GET" del navegador web, enviada desde nuestro ordenador a la dirección IP del sistema de control, el microcontrolador responderá con una página web almacenada en su memoria. Esta página será mostrada automáticamente en la pantalla del ordenador por el navegador web. Cuando se recibe el comando ON, el diodo LED conectado a PORTA.0 se encenderá.

Del mismo modo, cuando se recibe el comando OFF, el diodo LED se apaga. Si en lugar de un diodo LED tenemos un relé, es posible controlar cualquier aplicación como una lámpara, un sistema de seguridad, un sistema de calefacción, etc.

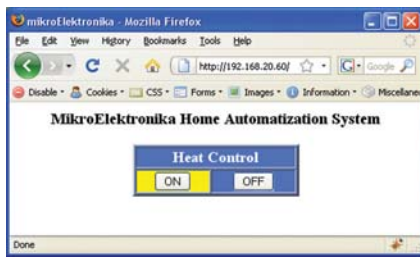


Figura 1. MikroElektronika Módulo Serie Ethernet con ENC28J60



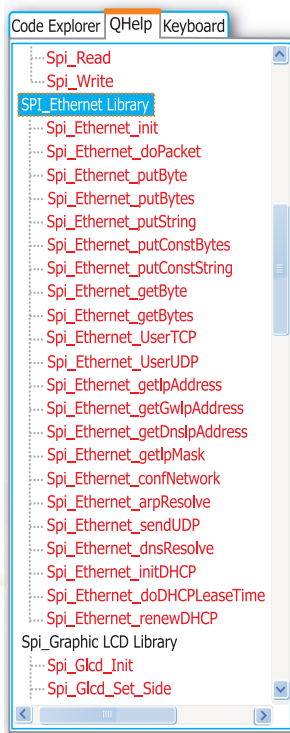
Esquema eléctrico 1. Conexión del módulo Serie Ethernet al Atmega 32

El control de cualquier aplicación doméstica consiste en la introducción de la dirección IP del sistema de control en el navegador web y especificar los comandos deseados. Por supuesto, es posible controlar más de un terminal del microcontrolador, lo cual nos permite gobernar un gran número de aplicaciones o un sistema de automatización doméstico completo.



La captura de pantalla ilustra la página web mostrada por el navegador después de introducir la dirección IP del sistema de control. En nuestro ejemplo, al pulsar sobre los botones ON y OFF provocaremos que el diodo LED se encienda y se apague, simulando el control de un sistema de calefacción.

Mas abajo está la lista de las funciones, ya creadas, contenidas en la librería SPI Ethernet Library. Esta librería esta integrada en el compilador mikroPASCAL for AVR.



<b>Spi_Ethernet_Init()</b>	Inicia el controlador ENC28J60
<b>Spi_Ethernet_Enable()</b>	Habilita de tráfico de la red
<b>Spi_Ethernet_Disable()</b>	Deshabilitar el tráfico de la red
<b>Spi_Ethernet_doPacket()</b>	Paquetes de procesos recibido
<b>Spi_Ethernet_putByte()</b>	Almacena un byte
<b>Spi_Ethernet_putBytes()</b>	Almacena bytes
<b>Spi_Ethernet_putConstBytes()</b>	Almacena bytes como constantes
<b>Spi_Ethernet_putString()</b>	Almacena string
<b>Spi_Ethernet_putConstString()</b>	Almacena string como constante
<b>Spi_Ethernet_getByte()</b>	Obtener un byte
<b>Spi_Ethernet_getBytes()</b>	Obtener bytes
<b>Spi_Ethernet_UserTCP()</b>	Código de manejo TCP
<b>Spi_Ethernet_UserUDP()</b>	Código de manejo UDP
<b>Spi_Ethernet_getIpAddress()</b>	Obtiene dirección IP
<b>Spi_Ethernet_getGwIpAddress()</b>	Obtiene dirección Gateway o pasarela
<b>Spi_Ethernet_getDnsIpAddress()</b>	Obtiene dirección DNS
<b>Spi_Ethernet_getIpMask()</b>	Obtiene máscara de dirección IP
<b>Spi_Ethernet_confNetwork()</b>	Establece los parámetros de red
<b>Spi_Ethernet_arpResolve()</b>	Envía una petición ARP
<b>Spi_Ethernet_sendUDP()</b>	Envía un paquete UDP
<b>Spi_Ethernet_dnsResolve()</b>	Envía una petición DNS
<b>Spi_Ethernet_initDHCP()</b>	Envía una petición DHCP
<b>Spi_Ethernet_doDHCPLeaseTime()</b>	Tiempo de proceso
<b>Spi_Ethernet_renewDHCP()</b>	Peticion de renovar DHCP

\*Funciones de la librería SPI Ethernet Library usadas en el programa

Otras funciones mikroPASCAL for AVR usadas en el programa:  
**Spi\_Init()** Inicializar el módulo SPI del microcontrolador  
**memcpy()** Copia posiciones de la memoria RAM del microcontrolador  
**memset()** Copia posiciones de la memoria RAM del microcontrolador

Ejemplo 1: Programa que demuestra el control a través de Ethernet

```

program enc_ethernet;
var myMacAddr : array[6] of byte; // my MAC address
    gwIpAddr : array[4] of byte; // gateway (router) IP address
    ipMask : array[4] of byte; // network mask (for example: 255.255.255.0)
    dnsIpAddr : array[4] of byte; // DNS server IP address
    indexPage : string[523]; // default html page
    getRequest : array[20] of byte; // HTTP request buffer

// mEthernet NIC pinout
Spi_Ethernet_Rst : sbit at PORTB.0;
Spi_Ethernet_CS : sbit at PORTB.1;
Spi_Ethernet_Rst_Direction : sbit at DDRB.0;
Spi_Ethernet_CS_Direction : sbit at DDRB.1;
// end ethernet NIC definitions

const httpHeader : string[30] = "HTTP/1.1 200 OK"+#10+#10; // HTTP header
const httpMimeTypeHTML : string[13] = "text/html"+#10+#10; // HTML MIME type
const httpMimeTypeScript : string[14] = "text/plain"+#10+#10; // TEXT MIME type

function putConstString(const s : byte) : word;
begin
    result := 0;
    while(s^ <> 0) do
        Spi_Ethernet_putByte(s^); s := s + 1; result := result + 1;
    end;
end;

function putString(s : byte) : word;
begin
    result := 0;
    while(s^ <> 0) do
        Spi_Ethernet_putByte(s^); s := s + 1; result := result + 1;
    end;
end;

function SPI_Ethernet_UserTCP(var remoteHost : array[4] of byte;
    remotePort, localPort, reqLength : word) : word;
var len : word; // my reply length
begin
    len := string(10);
    if(localPort < 80) then // I listen only to web request on port 80
        begin
            result := 0; exit;
        end;
    // get 10 first bytes only of the request, the rest does not matter here
    for len := 0 to 14 do getReqLen := SPI_Ethernet_getByte();
    getReqLen := 0;

    tmp := 'GET /';
    if(memcmp(getRequest, @tmp, 5) < 0) then // only GET method
        begin
            result := 0; exit;
        end;
    tmp := 'ON';
    if(memcmp(getRequest+11, @tmp, 2) = 0) then // do we have ON command
        PORTA.B0 := 1 // set PORTA bit 0
    else
        begin
            tmp := 'OFF';
            if(memcmp(getRequest+11, @tmp, 3) = 0) then // do we have OFF command
                PORTA.B0 := 0; // clear PORTA bit 0
            end;
        end;
    if(PORTA.B0) then
        begin
            tmp := '#FFFF00'; memcpy(@indexPage+340, @tmp, 6); // highlight (yellow) ON
            tmp := '#4974E2'; memcpy(@indexPage+431, @tmp, 6); // clear OFF
        end;
    else
        begin
            tmp := '#4974E2'; memcpy(@indexPage+340, @tmp, 6); // clear ON
            tmp := '#FFFF00'; memcpy(@indexPage+431, @tmp, 6); // highlight (yellow) OFF
        end;
    len := putConstString(httpHeader); // HTTP header
    len := len + putConstString(httpMimeTypeHTML); // with HTML MIME type
    len := len + putString(indexPage); // HTML page first part
    result := len; // return to the library with the number of bytes to transmit
end;

function SPI_Ethernet_UserUDP(var remoteHost : array[4] of byte;
    remotePort, destPort, reqLength : word) : word;
begin
    result := 0; // back to the library with the length of the UDP reply
end;

// set PORTA as output
PORTA.DIR := 0; // clear PORTA.B0
DDRA.B0 := 1; // set PORTA.B0 as output (rel control pin)

indexPage =
<html><head><title>mikroElektronika</title></head><body>+
<h3 align=center>MikroElektronika Home Automatization System</h3>+
<form name='input' action='/' method='get'+
<table align=center width=200 bgcolor=#4974E2 border=2><tr>+
<td align=center colspan=2><font size=4 color=white><b>Heat Control</b></td></tr>+
<tr>+
<td align=center colspan=2 align=center bgcolor=#4974E2><input name='tst1' width=60 +
+type='submit' value='ON'></td><td align=center bgcolor=#FFFF00>+
+<input name='tst2' type='submit' value='OFF'></td></tr></table>+
</body></html>;

myMacAddr[0] := 0x00; myMacAddr[1] := 0x14; myMacAddr[2] := 0xA5;
myMacAddr[3] := 0x76; myMacAddr[4] := 0x19; myMacAddr[5] := 0x3F;
ipMask[0] := 255; ipMask[1] := 255; ipMask[2] := 255; ipMask[3] := 0;
gwIpAddr[0] := 192; gwIpAddr[1] := 168; gwIpAddr[2] := 20; gwIpAddr[3] := 60;
dnsIpAddr[0] := 192; dnsIpAddr[1] := 168; dnsIpAddr[2] := 20; dnsIpAddr[3] := 1;

// starts ENC28J60 with: reset bit on PORTB.F0, CS bit on PORTB.F1,
Spi_Init_Advanced(SPI_MASTER, SPI_FCY_DIV4, SPI_CLK_LO_LEADING);
Spi_Rd_Ptr := @Spi_Read;
Spi_Ethernet_UserTCP_Ptr := @Spi_Ethernet_UserTCP;
Spi_Ethernet_UserUDP_Ptr := @Spi_Ethernet_UserUDP;
Spi_Ethernet_Init(myMacAddr, myIpAddr, Spi_Ethernet_FULLDUPLEX);

// dhcp will not be used here, so use preconfigured addresses
Spi_Ethernet_confNetwork(ipMask, gwIpAddr, dnsIpAddr);

while true do // do forever
    Spi_Ethernet_doPacket(); // process incoming Ethernet packets
end.

```

**NOTA:** El código para este ejemplo ha sido escrito para microcontroladores PIC® en lenguaje C, Basic y Pascal, del mismo modo que los programas han sido escritos para microcontroladores dsPIC® y AVR®. Todo ello lo pueden encontrar en nuestra página web: [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/)

