

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 *mikroC PRO 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.

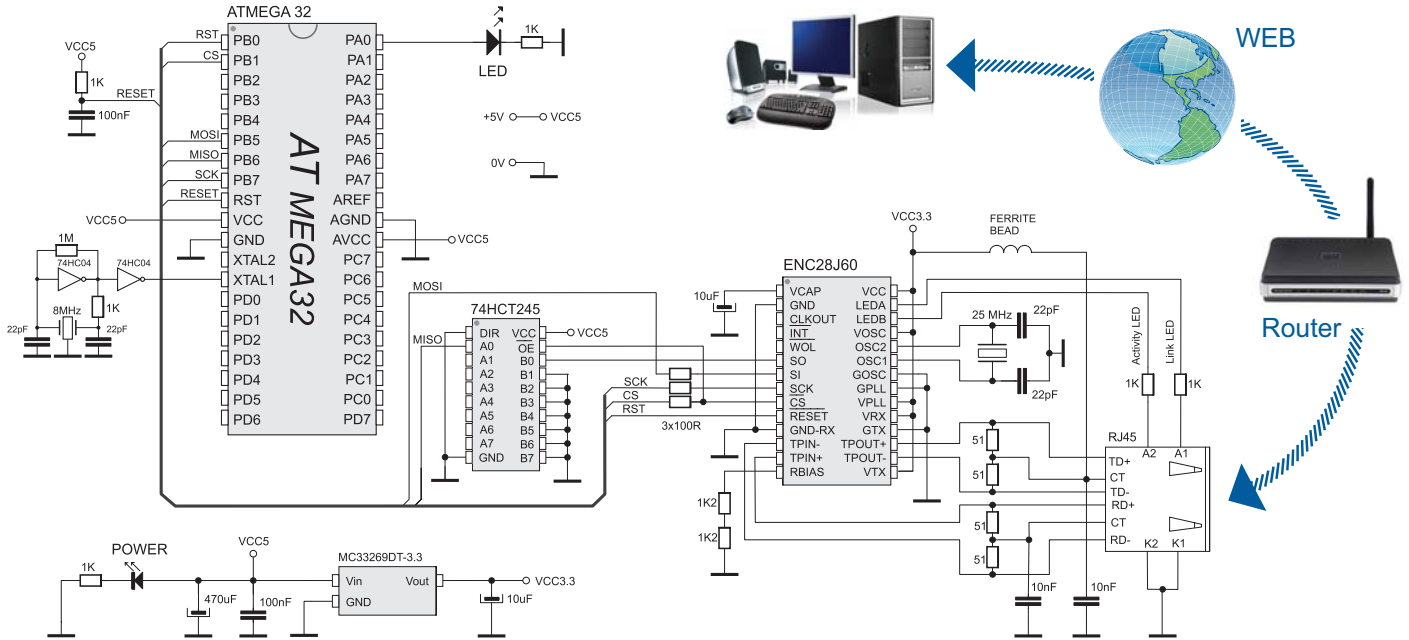
La parte más importante del programa es la función `SPI_Ethernet_userTCP`, que procesa todos los comandos recibidos.

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.

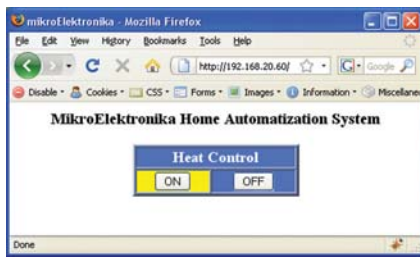


Figura 1. MikroElektronika Módulo Serie Ethernet con ENC28J60



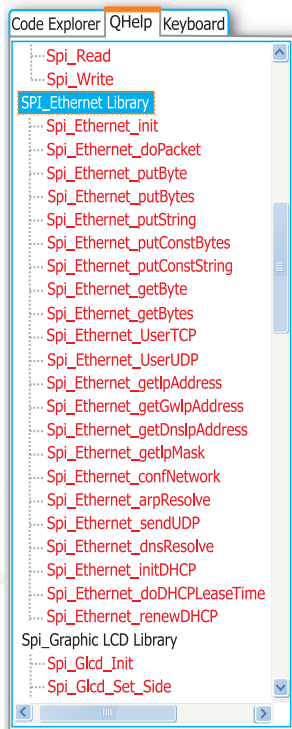
Esquema eléctrico 1. Conexión del módulo Serie Ethernet al Atmega 32

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 mikroC para PIC.



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

Otras funciones mikroC para PIC 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 memoria RAM del microcontrolador

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

```

// duplex config flags
#define Spi_Ethernet_HALFDUPLEX 0x00 // half duplex
#define Spi_Ethernet_FULLDUPLEX 0x01 // full duplex

// mE ethernet NIC pinout
sfr sbit SPI_Ethernet_Rst at PORTB.B0; // reset pin
sfr sbit SPI_Ethernet_CS at PORTB.B1; // chip select pin
sfr sbit SPI_Ethernet_Rst_Direction at DDRB.B0; // reset pin direction
sfr sbit SPI_Ethernet_CS_Direction at DDRB.B1; // chip select pin direction
// end ethernet NIC definitions

const char httpHeader[] = "HTTP/1.1 200 OK\r\nContent-type: "; // HTTP header
const char httpMimeTypeHTML[] = "text/html\r\n"; // HTML MIME type
const char httpMimeTypeScript[] = "text/plain\r\n"; // TEXT MIME type

// default html page
char 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></font>\
</td></tr><tr><td align=center bgcolor=#4974E2><input name='tst1' width=60\
type='submit' value='ON'></td><td align=center bgcolor=#FFFFFF>\
<input name='tst2' type='submit' value='OFF'></td></tr></table>\
</form></body></html>";

// network parameters
char myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f}; // my MAC address
char myIpAddr[4] = {192, 168, 20, 60}; // my IP address
char gwIpAddr[4] = {192, 168, 20, 6}; // gateway IP address
char dnsIpAddr[4] = {192, 168, 20, 6}; // dns IP address
char ipMask[4] = {255, 255, 255, 0}; // subnet mask
// end network parameters

unsigned char getRequest[20]; // HTTP request buffer

unsigned int SPI_Ethernet_UserTCP( char *remoteHost, unsigned int remotePort,
unsigned int localPort, unsigned int reqLength)
{
  unsigned int len; // my reply length
  if(localPort != 80) return(0); // I listen only to web request on port 80

  // get 10 first bytes only of the request, the rest does not matter here
  for(len = 0; len < 15; len++) getRequest[len] = SPI_Ethernet_getByte();
  getReqSize[len] = 0;

  if(memcmp(getRequest, "GET /", 5)) return(0); // only GET method

  if(memcmp(getRequest+11, "ON", 2)) // do we have ON command
  PORTA.F0 = 1; // set PORTA bit0
  else
  if(memcmp(getRequest+11, "OFF", 3)) // do we have OFF command
  PORTA.F0 = 0; // clear PORTA bit0

  if(PORTA.F0)
  {
    memcpy(indexPage+340, "#FFFF00", 6); // highlight (yellow) ON
    memcpy(indexPage+431, "#4974E2", 6); // clear OFF
  }
  else
  {
    memcpy(indexPage+340, "#4974E2", 6); // clear ON
    memcpy(indexPage+431, "#FFFF00", 6); // highlight (yellow) OFF
  }

  len = SPI_Ethernet_putConstString(httpHeader); // HTTP header
  len += SPI_Ethernet_putConstString(httpMimeTypeHTML); // with HTML MIME type
  len += SPI_Ethernet_putString(indexPage); // HTML page first part
  return len; // return to the library with the number of bytes to transmit
}

unsigned int SPI_Ethernet_UserUDP( char *remoteHost, unsigned int remotePort,
unsigned int destPort, unsigned int reqLength)
{
  return 0; // back to the library with the length of the UDP reply
}

void main()
{
  // set PORTA as output
  PORTA0_bit = 0; // clear PORTA.B0
  DDRA.F0 = 1; // set PORTA.B0 as output (relé control pin)

  // starts ENC28J60 with: reset bit on PORTB.F0, CS bit on PORTB.F1,
  // my MAC & IP address, full duplex
  SPI1_Init_Advanced(SPI_MASTER, SPI_FCY_DIV4, SPI_CLK_LO_LEADING);
  Spi_Rd_Ptr = SPI1_Read; // pass SPI Read function of used SPI module
  // full duplex, CRC + MAC Unicast + MAC Broadcast filtering
  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(1) { // do forever
    SPI_Ethernet_doPacket(); // process incoming Ethernet packets
  }
}

```

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/

