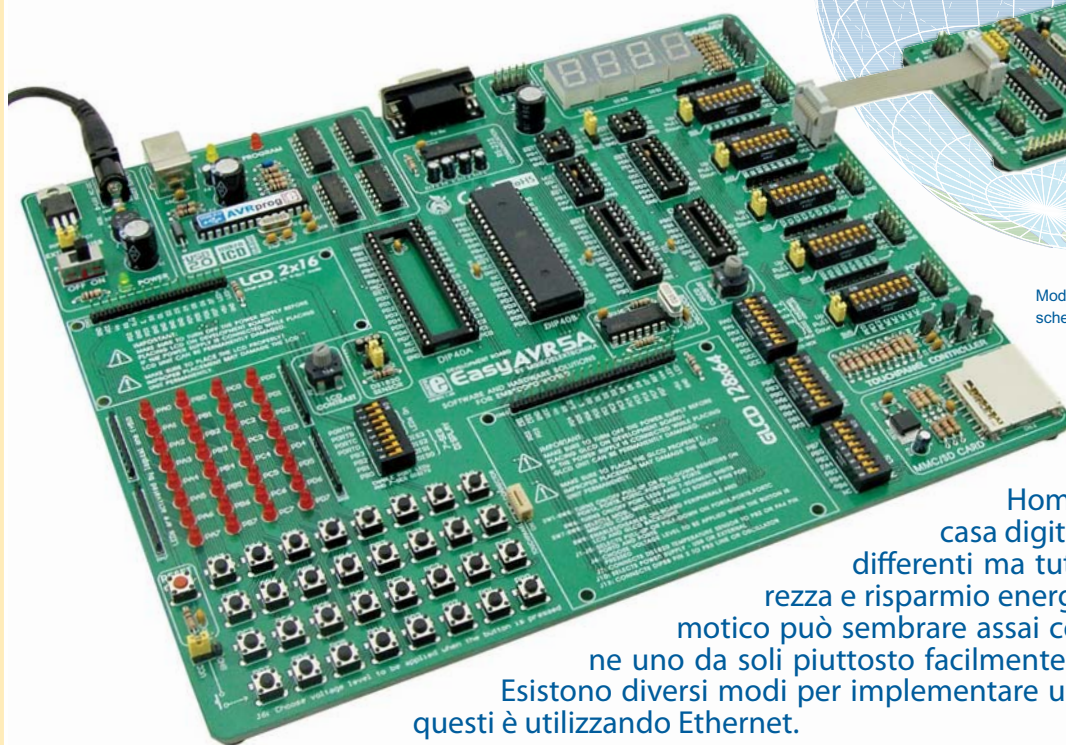


OK. Ora hai bisogno di... ETHERNET



Modulo seriale-ethernet connesso alla scheda di sviluppo easyAVR5A

Home automation, home control, casa digitale o smart home sono nomi differenti ma tutti sinonimi di comfort, sicurezza e risparmio energetico. Creare un sistema domotico può sembrare assai costoso, ma è possibile crearne uno da soli piuttosto facilmente e senza grossi investimenti. Esistono diversi modi per implementare un sistema domotico e uno di questi è utilizzando Ethernet.

Srdjan Tomic
MikroElektronika - Software Department

Tutto il necessario è limitato ad un microcontrollore Atmega 32 ed un chip ENC28J60 per la gestione dell'Ethernet. L'ENC28J60 è dotato di interfaccia seriale SPI e costituisce una soluzione ottimale per molte famiglie di microcontrollori quali AVR, dsPIC, ecc... Per la connessione alla rete Ethernet viene utilizzato un connettore RJ-45 CviLux CJCBA8HF1Y0. L'apparecchiatura domestica da controllare viene simulata con un LED collegato alla porta RB0 del microcontrollore. Il compilatore *mikroC PRO for AVR* contiene la libreria Ethernet e ci permette di semplificare fortemente la fase di scrittura del programma. Usando alcune routine di questa libreria è possibile scrivere il programma che consentirà di accendere e spegnere le apparecchiature attraverso un browser web.

Il programma dovrà svolgere le seguenti operazioni:

- Step 1.** Creare una pagina html attraverso la quale si gestisce il micro. Importare il codice come stringa.
- Step 2.** Impostare gli indirizzi IP, DNS, Gateway e Subnet mask utilizzando i dati forniti dal vostro provider per la connessione ad internet.

I parametri della nostra rete saranno i seguenti:

IP : 192.168.20.60 (Control System address)
DNS : 192.168.20.6 (Domain Name System address)
GATEWAY : 192.168.20.1 (Gateway address)
SUBNET : 255.255.255.0 (Subnet mask)

- Step 3.** Disabilitare gli ingress analogici della PORTA. Tutti i pin della porta A del microcontrollore andranno azzerati e configurati come uscite.
- Step 4.** Inizializzare il modulo SPI del Atmega 32.
- Step 5.** Inizializzare il chip ENC28J60.
- Step 6.** Scrivere il codice usando la funzione `Spi_Ethernet_userTCP` la quale, dopo aver ricevuto un comando dal browser web, accenderà o spegnerà il LED connesso alla porta PA.0.
- Step 7.** Leggere i vari comandi mediante un ciclo infinito.

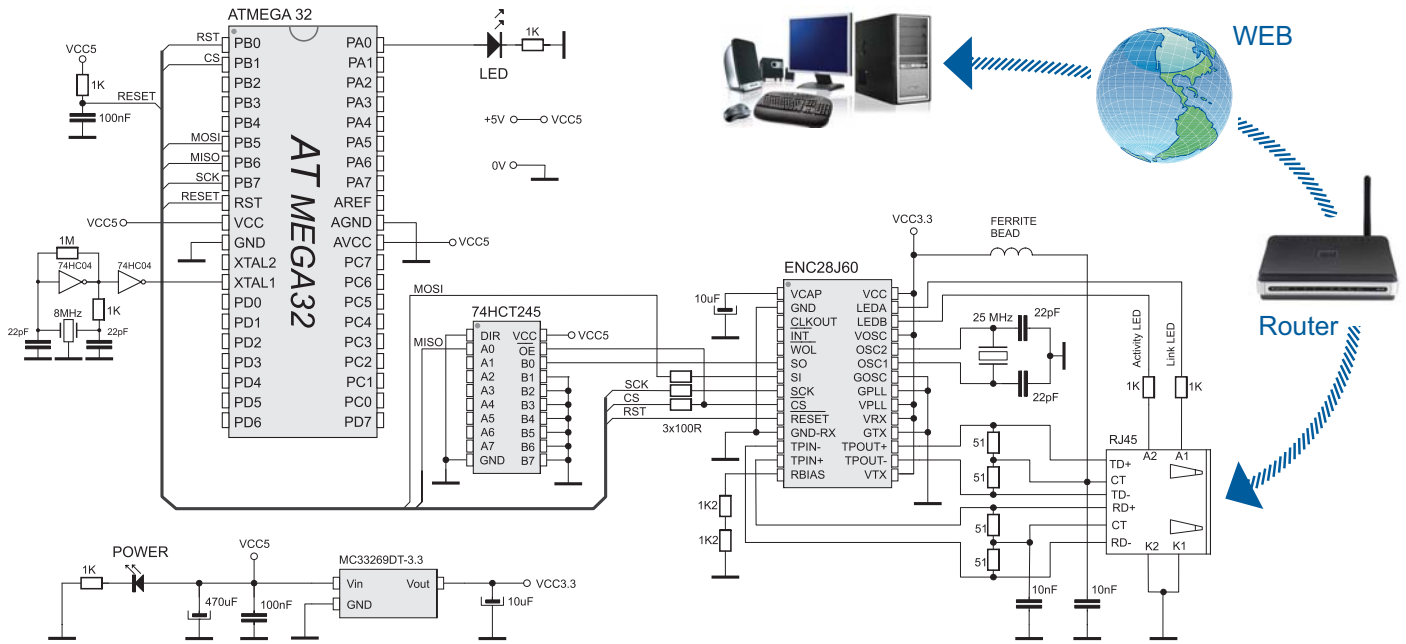
La parte più importante del programma è la funzione `Spi_Ethernet_userTCP`. Quando una richiesta di tipo "GET" viene inviata all'indirizzo IP del nostro sistema, il microcontrollore risponde con una pagina web che verrà visualizzata nel browser.

Quando viene ricevuto un comando ON, il LED connesso alla porta PA.0 verrà acceso automaticamente. Analogamente all'arrivo di un comando OFF lo stesso LED verrà spento. Inserendo un relè al posto del LED sarà così possibile accendere e spegnere attraverso la pagina web apparecchiature come lampade, sistemi di allarme, riscaldamento, ecc...

Il controllo dell'intero sistema viene fatto inserendo l'indirizzo IP del sistema nella barra di indirizzi del browser e selezionando i vari comandi direttamente dalla pagina web. Ovviamente è possibile modificare il programma per pilotare contemporaneamente più uscite del microcontrollore per creare un vero e proprio sistema di controllo dell'appartamento veramente completo.



Figure 1. Mikroelektronika Serial Ethernet modulo



schema 1. Connessione del modulo Serial Ethernet al Atmega 32

La Figura 2 mostra la pagina web visualizzata nel browser dopo aver inserito l'indirizzo IP del sistema di controllo. Cliccando sui pulsanti ON e OFF si provoca rispettivamente l'accensione e lo spegnimento del LED. Nel nostro esempio la pagina web controlla l'accensione e lo spegnimento dell'impianto di riscaldamento della casa.

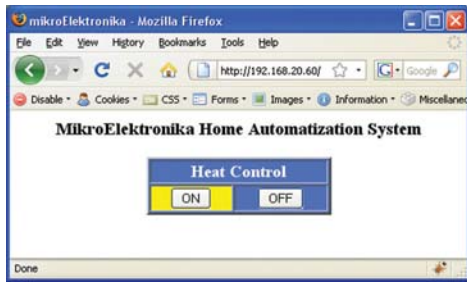
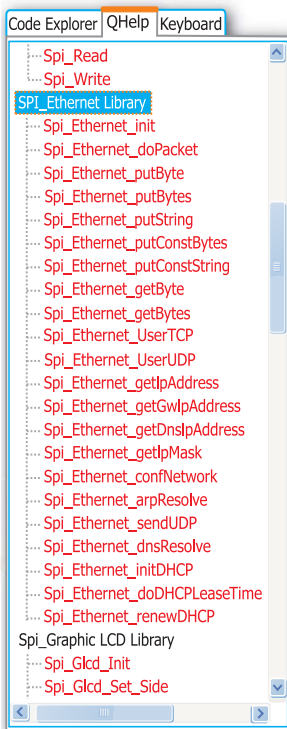


Figura 2. La pagina web di gestione del sistema

La libreria SPI Ethernet del compilatore *mikroC PRO for AVR* con le funzioni pronte all'uso.



Spi_Ethernet_Init()*	Inizializzazione del controller ENC28J60
Spi_Ethernet_Enable()	Abilita traffico di rete
Spi_Ethernet_Disable()	Disabilita traffico di rete
Spi_Ethernet_doPacket()*	Processa il pacchetto ricevuto
Spi_Ethernet_putByte()	Memorizza un byte
Spi_Ethernet_putBytes()	Memorizza i bytes
Spi_Ethernet_putConstBytes()	Memorizza i bytes delle costanti
Spi_Ethernet_putString()*	Memorizza la stringa
Spi_Ethernet_putConstString()*	Memorizza la stringa di costanti
Spi_Ethernet_getByte()	Preleva un byte
Spi_Ethernet_getBytes()	Preleva più bytes
Spi_Ethernet_UserTCP()*	Codice di gestione del
Spi_Ethernet_UserUDP()	Codice di gestione del UDP
Spi_Ethernet_getIpAddress()	Ottiene l'indirizzo IP
Spi_Ethernet_getGwIpAddress()	Ottiene l'indirizzo del gateway
Spi_Ethernet_getDnsIpAddress()	Ottiene l'indirizzo del DNS
Spi_Ethernet_getIpMask()	Ottiene la subnet mask
Spi_Ethernet_confNetwork()*	Imposta i parametri
Spi_Ethernet_arpResolve()	Invia una richiesta ARP
Spi_Ethernet_sendUDP()	Invia un pacchetto UDP
Spi_Ethernet_dnsResolve()	Invia una richiesta DNS
Spi_Ethernet_initDHCP()	Invia una richiesta DHCP
Spi_Ethernet_doDHCPLeaseTime()	Tempo di processo
Spi_Ethernet_renewDHCP()	Richiesta di rinnovo DHCP

Funzioni usate nel programma:

- Spi_Init()** Inizializzazione del modulo SPI
- memcpy()** copia in memoria
- memcmp()** confronta in memoria

Esempio 1: Programma dimostrativo per le operazioni Serial 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>\r\n"
"<h3 align=center>MikroElektronika Home Automatization System</h3>\r\n"
"<form name='input' action='/' method='get'>\r\n"
"<table align=center width=200 bgcolor=#4974E2 border=2><tr>\r\n"
"<td align=center colspan=2><font size=4 color=white><b>Heat Control</b></font>\r\n"
"</td></tr><tr><td align=center bgcolor=#4974E2><input name='tst1' width=60\r\n"
" type='submit' value='ON'></td><td align=center bgcolor=#FFFFFF>\r\n"
"<input name='tst2' type='submit' value='OFF'></td></tr></table>\r\n"
"</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, 1}; // 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();
  getrequest[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, "#FFFFFF00", 6); // highlight (yellow) ON
    memcpy(indexPage+431, "#4974E2", 6); // clear OFF
  }
  else
  {
    memcpy(indexPage+340, "#4974E2", 6); // clear ON
    memcpy(indexPage+431, "#FFFFFF00", 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 (rele 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
}

```

NOTE: Il codice per AVR® microcontrollers disponibile in C, Basic e Pascal ed i programmi scritti per dsPIC® e PIC® possono essere scaricati dal sito web: www.mikroe.com/en/article/

