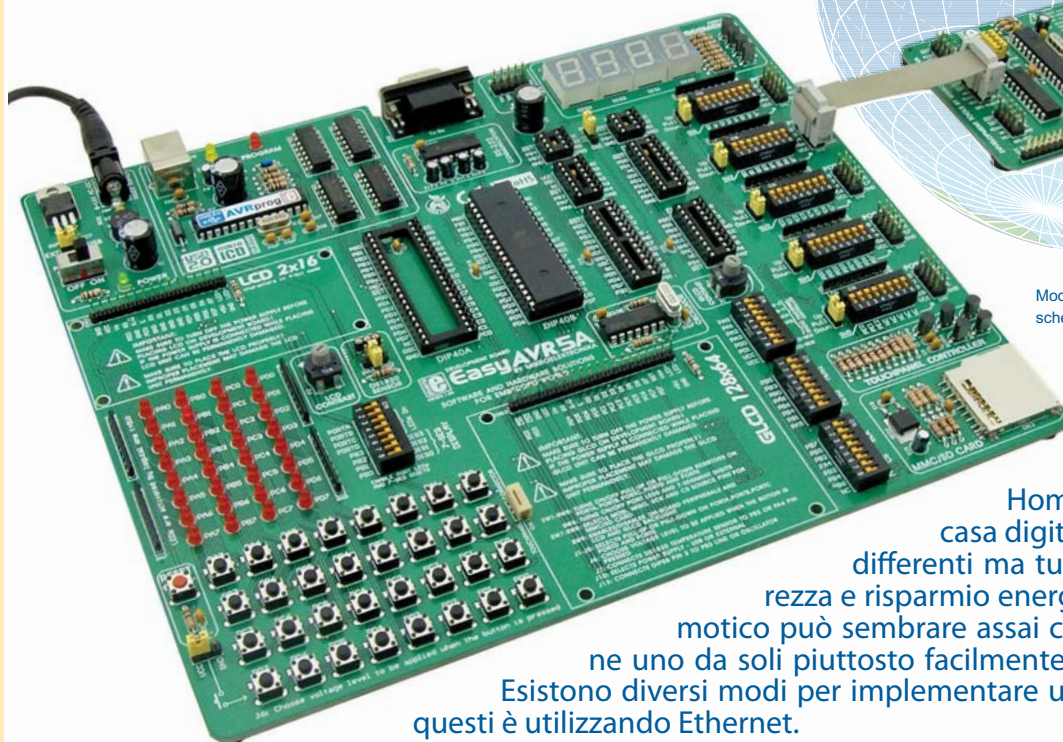


OK. Ora hai bisogno di... ETHERNET



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 *mikroBASIC 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.

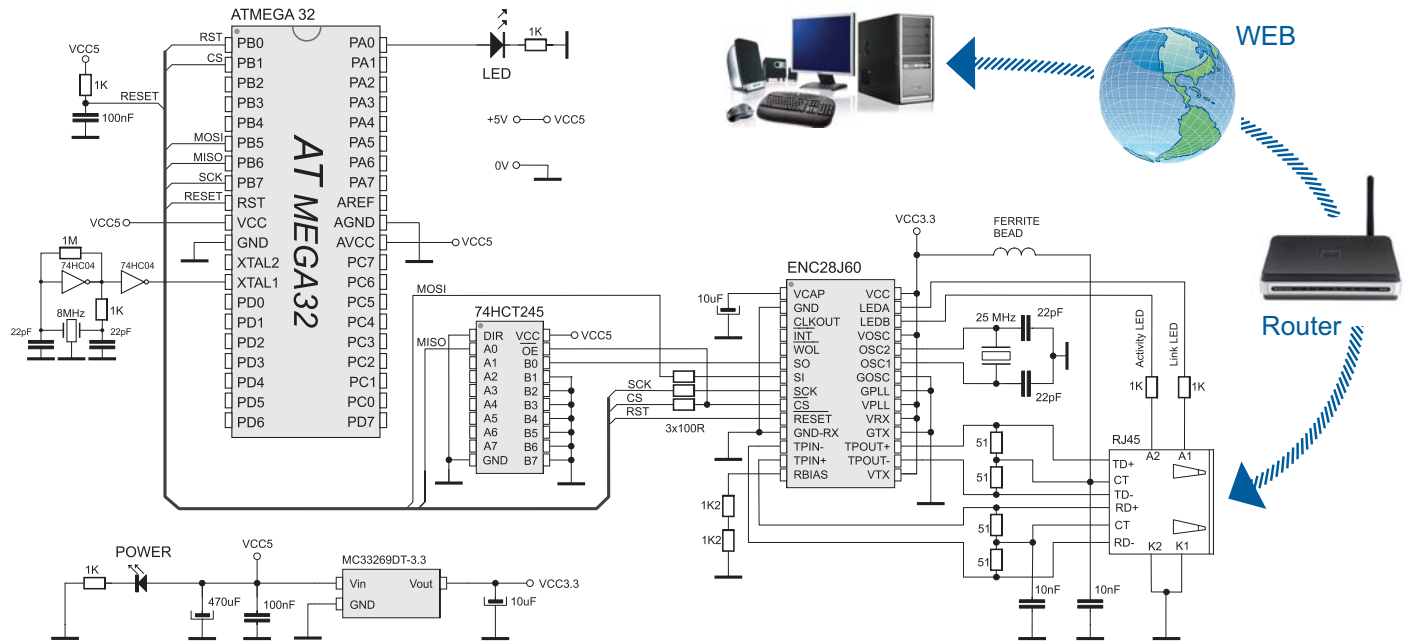
La parte più importante del programma è la funzione `Spi_Ethernet_userTCP`. Quando una richiesta di tipo "GET" viene inviata all'indirizzo IP del nostro sistema, il microcontrollore risponde con una pagina web che verrà visualizzata nel browser.

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.

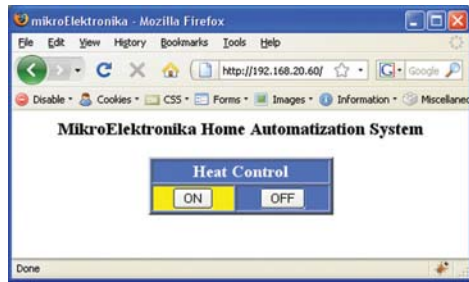
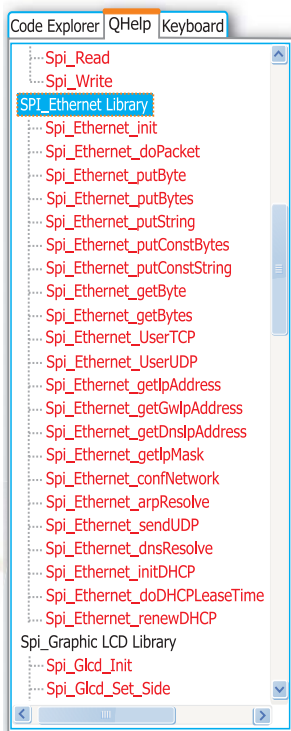


Figura 2. La pagina web di gestione del sistema

La libreria SPI Ethernet del compilatore *mikroBASIC 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:

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

```

program enc_ethernet

dim myMacAddr as byte[6] ' my MAC address
myIpAddr as byte[4] ' my IP address
myGwAddr as byte[4] ' gateway (router) IP address
ipMask as byte[4] ' network mask (for example: 255.255.255.0)
dnsIpAddr as byte[4] ' DNS server IP address
indexPage as string[33] ' default html page
getReq as byte[20] ' HTTP request buffer
httpHeader as string[30] ' HTTP header
httpMimeTypeHTML as string[13] ' HTML MIME type
httpMimeTypeScript as string[14] ' TEXT MIME type

' mE ethernet NIC pinout
SPI_Ethernet_Rst as sbit at PORTB.B0
SPI_Ethernet_CS as sbit at PORTB.B1
SPI_Ethernet_Rst_Direction as sbit at DDRB.B0
SPI_Ethernet_CS_Direction as sbit at DDRB.B1
' end ethernet NIC definitions

sub function putString(dim s as ^byte) as word
result = 0
while(s <> 0)
Spi_Ethernet_putByte(s)
Inc(s)
Inc(result)
wend
end sub

sub function SPI_Ethernet_UserTCP(dim byref remoteHost as byte[4],
dim remotePort, localPort, reqLength as word) as word
my reply length

if(localPort <> 80) then ' I listen only to web request on port 80
result = 0 exit
end if

' get 15 first bytes only of the request, the rest does not matter here
for cnt = 0 to 14 getReq[cnt] = SPI_Ethernet_GetByte() next cnt
getReq[cnt] = 0

tmp = "GET/"
if(memcmp(getReq, @tmp, 5) <> 0) then ' only GET method
result = 0 exit
end if

tmp = "ON"
if(memcmp(getReq+11, @tmp, 2) = 0) then ' do we have ON command
PORTA.B0 = 1
set PORTA bit 0
else
tmp = "OFF"
if(memcmp(getReq+11, @tmp, 3) = 0) then ' do we have OFF command
PORTA.B0 = 0
clear PORTA bit 0
end if

if (PORTA.B0) then
tmp = "#FFFF00" memcpy(indexPage+340, @tmp, 6) ' highlight (yellow) ON
tmp = "#4974E2" memcpy(indexPage+431, @tmp, 6) ' clear OFF
else
tmp = "#4974E2" memcpy(indexPage+340, @tmp, 6) ' clear ON
tmp = "#FFFF00" memcpy(indexPage+431, @tmp, 6) ' highlight (yellow) OFF
end if

cnt = putString(@httpHeader) ' HTTP header
cnt = cnt + putString(@httpMimeTypeHTML) ' with HTML MIME type
cnt = cnt + putString(indexPage) ' HTML page first part
result = cnt ' return to the library with the number of bytes to transmit
end sub

sub function SPI_Ethernet_UserUDP(dim byref remoteHost as byte[4],
dim remotePort, destPort, reqLength as word) as word
result = 0 ' back to the library with the length of the UDP reply
end sub

main:
' set PORTA as output
PORTA bit = 0 ' clear PORTA.B0
DDRA.B0 = 1 ' set PORTA.B0 as output (relay control pin)

httpHeader = "HTTP/1.1 200 OK" + chr(10) + "Content-type:"
httpMimeTypeHTML = "text/html" + chr(10) + chr(10)
httpMimeTypeScript = "text/plain" + chr(10) + chr(10)
indexPage =
"<html><head><title>mikroElektronika</title></head><body>"+
"<h3 align=center>MikroElektronika Home Automation System</h3>"+
"<form name="+chr(34)+"input"+chr(34)+" action="+chr(34)+"?"+chr(34)+" method="+
chr(34)+" get "+chr(34)+"><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>"+
"</td></tr><tr><td align=center bgcolor=#4974E2><input name="+chr(34)+"tst1"+
chr(34)+" width=60 type="+chr(34)+" submit="+chr(34)+" value="+chr(34)+" ON"+
chr(34)+"></td><td align=center bgcolor=#FFFF00><input name="+chr(34)+"tst2"+
chr(34)+" type="+chr(34)+" submit="+chr(34)+" value="+chr(34)+" OFF"+chr(34)+"></td></tr></table></form></body></html>"

myMacAddr[0] = 0x00 myMacAddr[1] = 0x14 myMacAddr[2] = 0x2A
myMacAddr[3] = 0x76 myMacAddr[4] = 0x19 myMacAddr[5] = 0x3F
ipMask[0] = 255 ipMask[1] = 255 ipMask[2] = 255 ipMask[3] = 0
myIpAddr[0] = 192 myIpAddr[1] = 168 myIpAddr[2] = 20 myIpAddr[3] = 60
myGwAddr[0] = 192 myGwAddr[1] = 168 myGwAddr[2] = 20 myGwAddr[3] = 6
dnsIpAddr[0] = 192 dnsIpAddr[1] = 168 dnsIpAddr[2] = 20 dnsIpAddr[3] = 3

' starts ENC28J60 with: reset bit on PORTB.F1,
my MAC & IP address, full duplex
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, myGwAddr, dnsIpAddr)

while true ' do forever
SPI_Ethernet_doPacket() ' process incoming Ethernet packets
wend
end

```

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/

