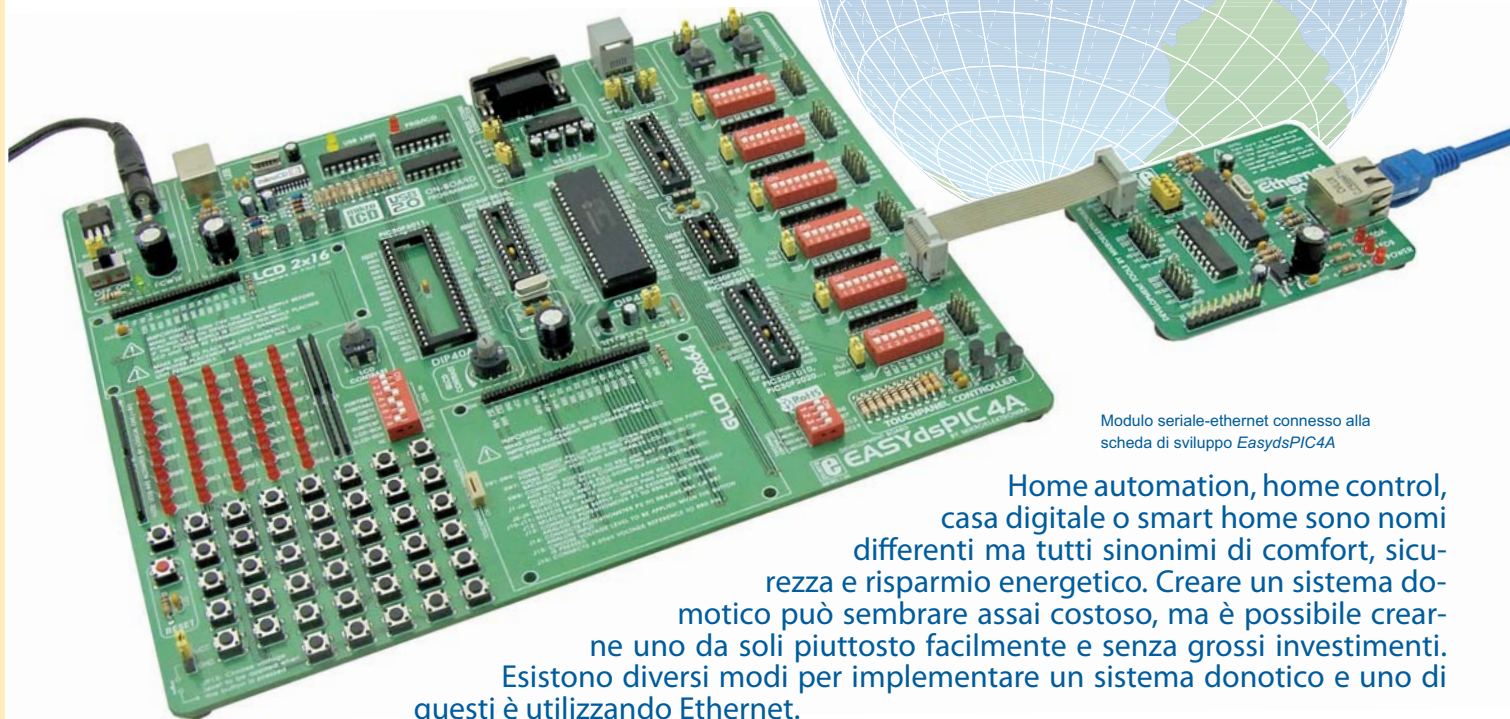


OK. Ora hai bisogno di... ETHERNET



Modulo seriale-ethernet connesso alla scheda di sviluppo EasydsPIC4A

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 dsPIC30F4013 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, PIC, 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 RD0 del microcontrollore. Il compilatore *mikroPASCAL for dsPIC* 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 PORTD. Tutti i pin della porta D del microcontrollore andranno azzerati e configurati come uscite.
- Step 4.** Inizializzare il modulo SPI del dsPIC30F4013.
- 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 RD0.
- Step 7.** Leggere i vari comandi mediante un ciclo infinito.

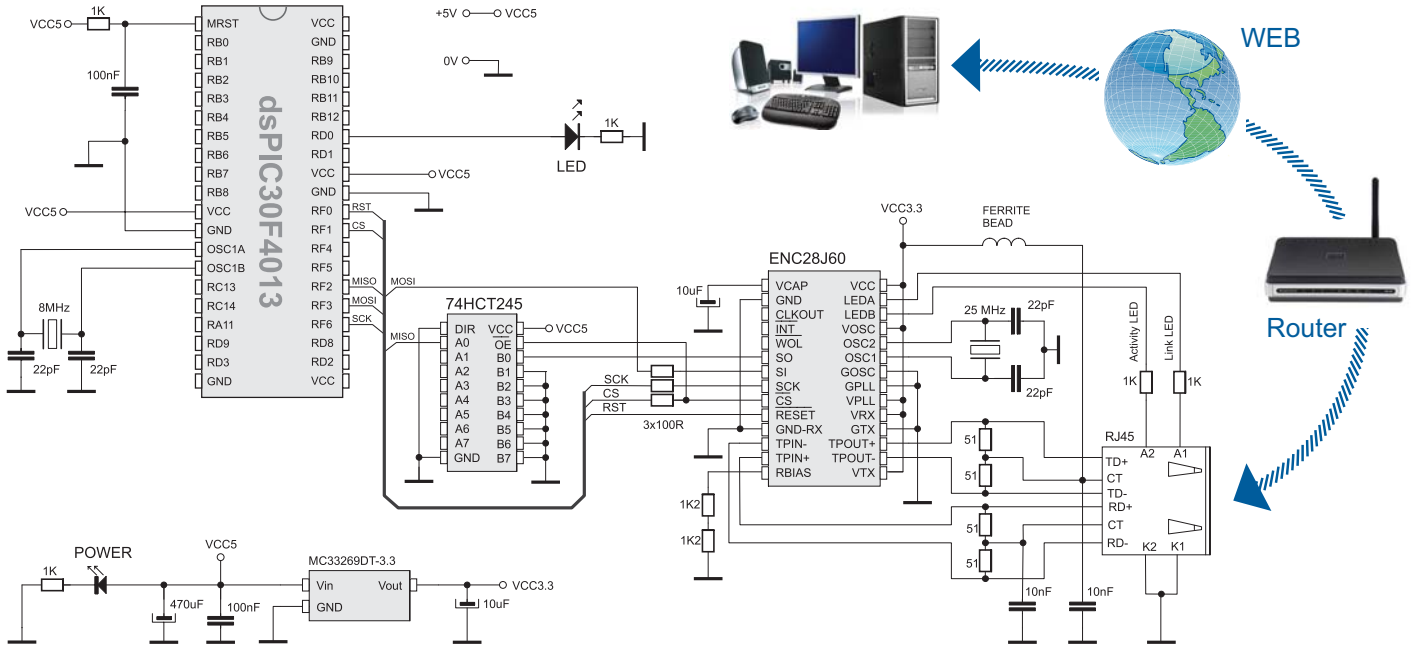
La parte più importante del programma è la funzione `Spi_Ethernet_userTCP`. Quando una richiesta di tipo "GET" viene inviata all'indirizzo IP del nostro sistema, il microcontrollore risponde con una pagina web che verrà visualizzata nel browser.

Quando viene ricevuto un comando ON, il LED connesso alla porta RD0 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 dsPIC30F4013

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.

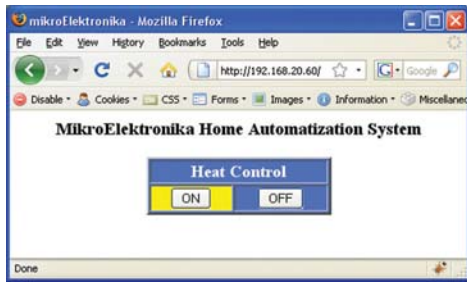
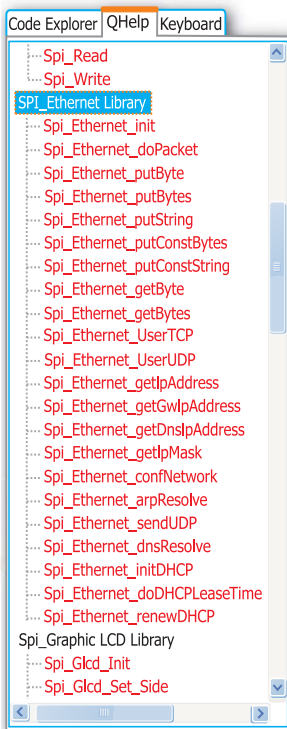


Figura 2. La pagina web di gestione del sistema

La libreria SPI Ethernet del compilatore *mikroPASCAL* for dsPIC 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 TCP
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 (two files)

```

program enc_ethernet;
uses home_auto_utils;

*****
RAM variables
var myMacAddr : array[6] of byte; // my MAC address
myIpAddr : array[4] of byte; // my IP address

begin
ADDFCFG := 0xFFFF; // no analog inputs
PORTD.0 := 0;
TRISD.0 := 0; // set PORTD.B0 as output (rele control pin)

indexPage :=
<html><head><title>mikroElektronika</title></head><body>+
<h3 align=center>MikroElektronika Home Automation System</h3>+
<table align=center width=200 bgcolor=#974E2 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=#FFF000 +
'<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;
myIpAddr[0]:=192; myIpAddr[1]:=168; myIpAddr[2]:=20; myIpAddr[3]:=60;

// starts ENC28J60 with: reset bit on PORTF.F0, CS bit on PORTF.F1
// my MAC & IP address, full duplex
Spi_Init();
// full duplex CRC + MAC Unicast + MAC Broadcast filtering
Spi_Ethernet_Init(PORTF.0, PORTF.1,
myMacAddr, myIpAddr, Spi_Ethernet_FULLDUPLEX);

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

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

// default html page
var indexPage : string[523];
var getRequest : array[20] of byte; // HTTP request buffer

implementation

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

function putString(var s : array[100] of byte) : word;
begin
result := 0;
while(s[result] <> 0) do
begin
Spi_Ethernet_putByte(s[result]); result := result + 1;
end;
end;

function SPI_Ethernet_UserTCP(var remoteHost : array[4] of byte;
remotePort, localPort, reqLength : word) : word;
var len : word;
tmp : string[10];
begin
if(localPort < 80) then // I listen only to web request on port 80
begin
result := 0;
exit;
end;
end;

// get 10 first bytes only of the request, the rest does not matter here
for len := 0 to 14 do getRequest[len] := Spi_Ethernet_getByte();
getRequest[len] := 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
PORTD.0 := 1 // set PORTD bit 0
else
tmp := 'OFF';
if(memcmp(getRequest+11, @tmp, 3) = 0) then // do we have OFF command
PORTD.0 := 0; // clear PORTD bit 0
end;
Delay 1uS();
if (PORTD.0) then
begin
tmp := '#FFF000'; 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 := '#FFF000'; 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);
begin
result := 0; // back to the library with the length of the UDP reply
end.

```

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

