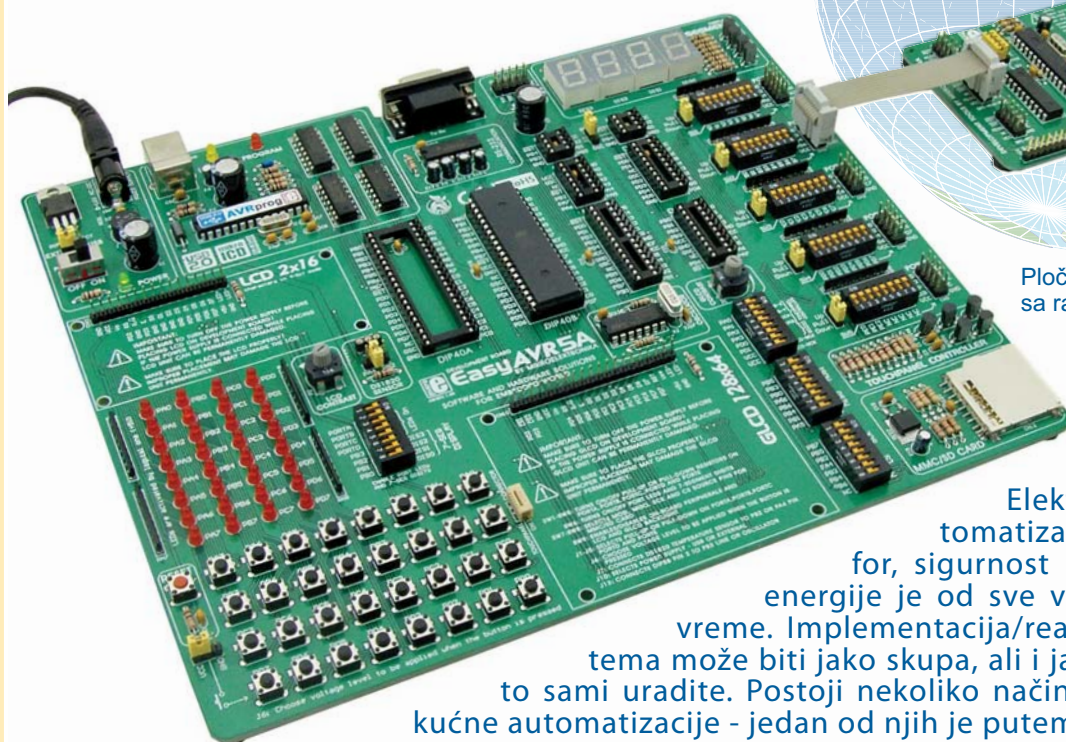


OK. Vama je potreban ... ETHERNET



Ploča serijskog etneta povezana sa razvojnim sistemom EasyAVR5A

Elektronski sistem kućne automatizacije je sinonim za komfor, sigurnost i uštedu energije. Ušteda energije je od sve većeg značaja u poslednje vreme. Implementacija/realizacija/ugradnja ovih sistema može biti jako skupa, ali i jako jeftina ako odlučite da to sami uradite. Postoji nekoliko načina da upravljate sistemom kućne automatizacije - jedan od njih je putem Eterneta.

Srdan Tomić
MikroElektronika - Software Department

Sve što je potrebno je jedan mikrokontroler Atmega 32 i serijski ethernet čip ENC28J60. Serijska komunikacija čini ovaj čip odličnim rešenjem i za mikrokontrolere iz drugih familija kao što su dsPIC, PIC itd. Za povezivanje na ethernet mrežu korišćen je RJ-45 konektor firme CviLux pod oznakom CJC8 A8HF1Y0. LED dioda povezana na PORTA.0 mikrokontrolera predstavljaće uređaj u kući kojim želimo da upravljamo.

Kompajler *mikroPASCAL for AVR* sadrži ethernet biblioteku koja će nam uveliko olakšati pravljenje samog programa za mikrokontroler. Uz pomoć svega par rutina iz ove biblioteke, napravićemo program koji će nam omogućiti paljenje i gašenje uređaja u kući putem web browser-a.

U programu je potrebno uraditi sledeće:

- Korak 1.** Napraviti html stranicu preko koje će se upravljati mikrokontrolerom i ubaciti je u kod u vidu string-a.
- Korak 2.** Setovati IP, DNS, Gateway adrese i Subnet masku dobijenu od strane internet provajdera.

Primeru radi, parametri naše lokalne mreže su:

IP: 192.168.20.60 (adresa kontrolnog sistema)
DNS: 192.168.20.1 (adresa Domain Name Sistema)
GATEWAY: 192.168.20.6 (adresa Gateway-a)
SUBNET: 255.255.255.0 (subnet maska)

Korak 3. Ugasiti analogne ulaze na portu A mikrokontrolera. Postaviti nulu na pin PORTA.0 i setovati ga kao izlazni.

Korak 4. Inicijalizovati SPI modul mikrokontrolera Atmega 32.

Korak 5. Inicijalizovati serijski ethernet modul ENC28J60.

Korak 6. Unutar funkcije `Spi_Ethernet_userTCP` napisati kod koji će po prijemu komande poslate putem web browser-a upaliti/ugasiti LED diodu.

Korak 7. U neprekidnoj petlji iščitavati primljene podatke.

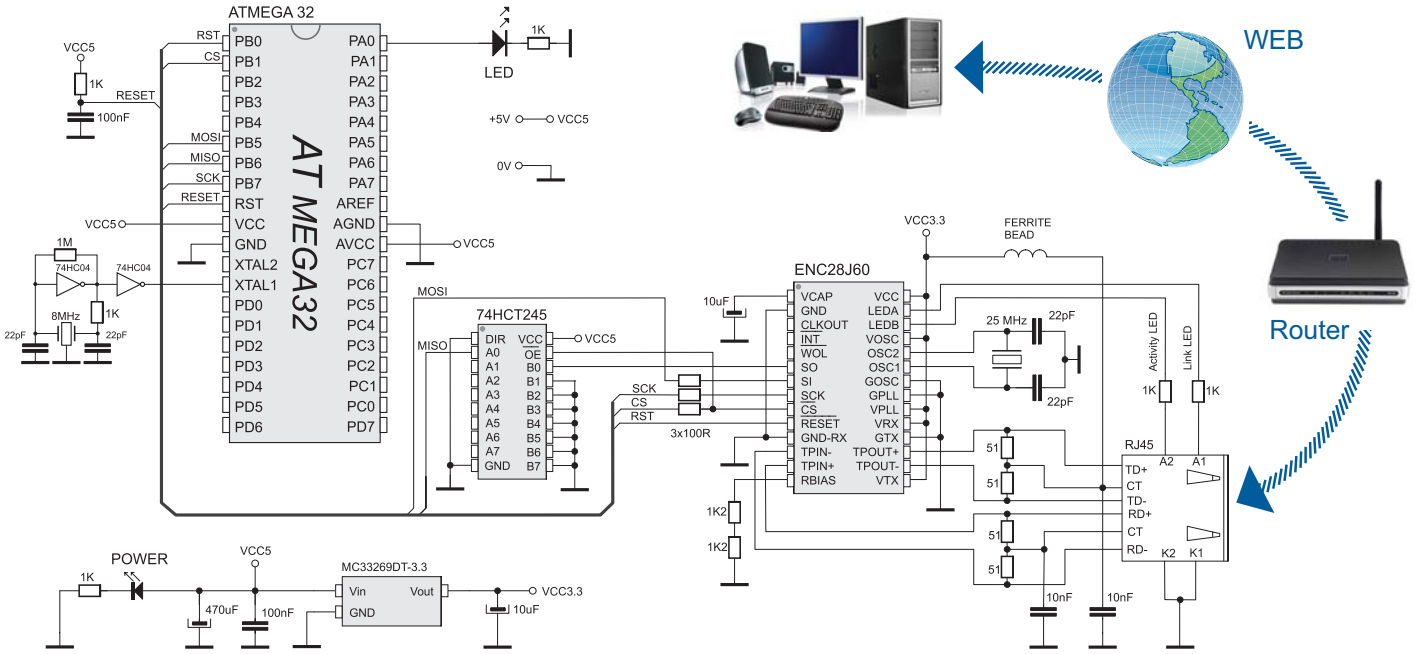
Najvažnija funkcija je `Spi_Ethernet_userTCP` u kojoj se odvija procesiranje svih primljenih komandi. Nakon prijema 'GET' zahteva poslatog sa vašeg računara putem web browser-a na IP adresu

kontrolnog sistema, mikrokontroler će browser-u vratiti web stranicu koja se nalazi u njegovoj memoriji. To je upravo ono što će biti prikazano na monitoru vašeg računara. Po prijemu ON komande LED dioda se pali, a po prijemu OFF komande ista se gasi. Ukoliko se umesto diode stavi relej može se ostvariti kontrola bilo kog uređaja kao što je svetlo, alarm, grejanje itd.

Sama kontrola se svodi na kucanje IP adrese kontrolnog sistema koji upravlja uređajima u kući u web browser i zadanje željenih komandi. Naravno, kako se kontroliše jedan pin mikrokontrolera tako se može kontrolisati i više pinova čime se dolazi do kompleksne kontrole većeg broja kućnih uređaja, odnosno, do sistema kućne automatizacije.



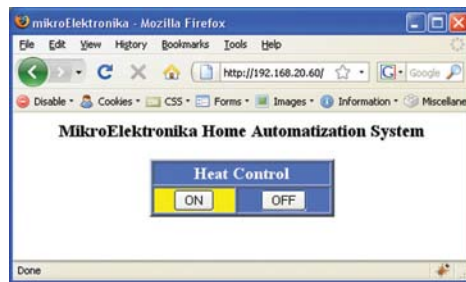
Slika 1. Pločica serijskog etneta sa ENC28J60 čipom



Šema 1. Povezivanje pločice serijskog ethernet i mikrokontrolera Atmega 32

Primer 1: Program demonstracije rada serijskog ethernet

Na slici 2 je prikazana stranica koja se dobija u web browser-u nakon unošenja IP adrese kontrolnog sistema. U opisanom primeru, klikom na tastere ON i OFF dolazi do paljenja i gašenja LED diode čime se simulira upravljanje sistemom za grejanje.



Slika 2. Stranica web browser-a

```

program enc_ethernet;
var myMacAddr : array[6] of byte ; // my MAC address
myIpAddr : array[4] of byte ; // my IP 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

// mEethernet NIC pinout
SPI_Ethernet_Rst : sbit at PORTB.B0;
SPI_Ethernet_CS : sbit at PORTB.B1;
SPI_Ethernet_Rst_Direction : sbit at DDRB.B0;
SPI_Ethernet_CS_Direction : sbit at DDRB.B1;
// end ethernet NIC definitions

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

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(s : byte) : word;
begin
result := 0;
while(s <> 0) do
begin
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);
var len : word ; // my reply length
tmp : string[10];
begin
result := 0; exit; // listen only to web request on port 80
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
result := 0; exit;
end;

tmp := 'ON';
if(memcmp(getRequest+11, @tmp, 2) = 0) then // do we have ON command
PORTA.B0 := 1 // set PORTA bit0
else
begin
tmp := 'OFF';
if(memcmp(getRequest+11, @tmp, 3) = 0) then // do we have OFF command
PORTA.B0 := 0; // clear PORTA bit0
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);
begin
result := 0; // back to the library with the length of the UDP reply
end;

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

indexPage =
<html><head><title>mikroElektronika</title></head><body>+
<h3 align=center>MikroElektronika Home Automation 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=#FFFF00 +
'input name='tst2' type='submit' value='OFF'></td></tr></table>+
</form></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;
myIpAddr[0] := 192; myIpAddr[1] := 168; myIpAddr[2] := 20; myIpAddr[3] := 60;
gwIpAddr[0] := 192; gwIpAddr[1] := 168; gwIpAddr[2] := 20; gwIpAddr[3] := 6;
dnsIpAddr[0] := 192; dnsIpAddr[1] := 168; dnsIpAddr[2] := 20; dnsIpAddr[3] := 1;

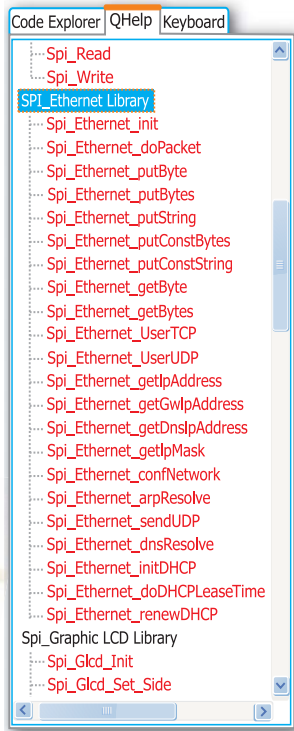
// starts ENC28J60 with: reset bit on PORTB.F0, CS bit on PORTB.F1,
myIpAddr and IP address, full duplex
SPI_Init_Advanced(SPI_MASTER, SPI_FCY_DIV4, SPI_CLK_LO_LEADING);
SPI_Rd_Ptr := @SPI_Read;
SPI_Write_UserTCP_Ptr := @SPI_Ethernet_UserTCP;
SPI_Ethernet_UserUDP_Ptr := @SPI_Ethernet_UserUDP;
SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX);

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

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

```

SPI Ethernet Library biblioteka sa gotovim funkcijama sadržana u kompajleru mikroPASCAL for AVR.



Spi_Ethernet_Init(*)	Inicijalizuje ENC28J60 kontroler
Spi_Ethernet_Enable()	Omoгуuje mrežni saobraćaj
Spi_Ethernet_Disable()	Onemogućuje mrežni saobraćaj
Spi_Ethernet_doPacket(*)	Obraduje dobijeni paket podataka
Spi_Ethernet_putByte()	Upisuje bajt
Spi_Ethernet_putBytes()	Upisuje zahtevane bajtove
Spi_Ethernet_putConstBytes()	Upisuje zahtevane const bajtove
Spi_Ethernet_putString(*)	Upisuje string
Spi_Ethernet_putConstString(*)	Upisuje const string
Spi_Ethernet_getByte(*)	Uzima bajt
Spi_Ethernet_getBytes()	Uzima zahtevane bajtove
Spi_Ethernet_UserTCP(*)	Obraduje TCP
Spi_Ethernet_UserUDP()	Obraduje UDP
Spi_Ethernet_getIpAddress()	Uzimanje IP adrese
Spi_Ethernet_getGwIpAddress()	Uzimanje Gateway adrese
Spi_Ethernet_getDnsIpAddress()	Uzimanje DNS adrese
Spi_Ethernet_getIpMask()	Uzimanje IP maske
Spi_Ethernet_confNetwork(*)	Postavlja mrežne parametre
Spi_Ethernet_arpResolve()	Šalje zahtev za ARP
Spi_Ethernet_sendUDP()	Šalje UDP
Spi_Ethernet_dnsResolve()	Šalje zahtev za DNS
Spi_Ethernet_initDHCP()	Šalje zahtev za DHCP
Spi_Ethernet_doDHCPLeaseTime()	Obrada DHCP validnosti
Spi_Ethernet_renewDHCP()	Zahtev za DHCP obnovu

* SPI Ethernet biblioteka korišćena u programu:

Spisak dodatnih funkcija korišćenih u programu:

- Spi_Init() Inicijalizacija modula na mikrokontroleru
- memcpy() Kopira RAM lokacije mikrokontrolera
- memcmp() Upoređuje RAM lokacije mikrokontrolera

NOTE: Pored proširene verzije ovog programa koji je napisan za AVR mikrokontrolere, sa našeg sajta možete preuzeti i verzije pisane za dsPIC i PIC mikrokontrolere. www.mikroe.com/en/article/

