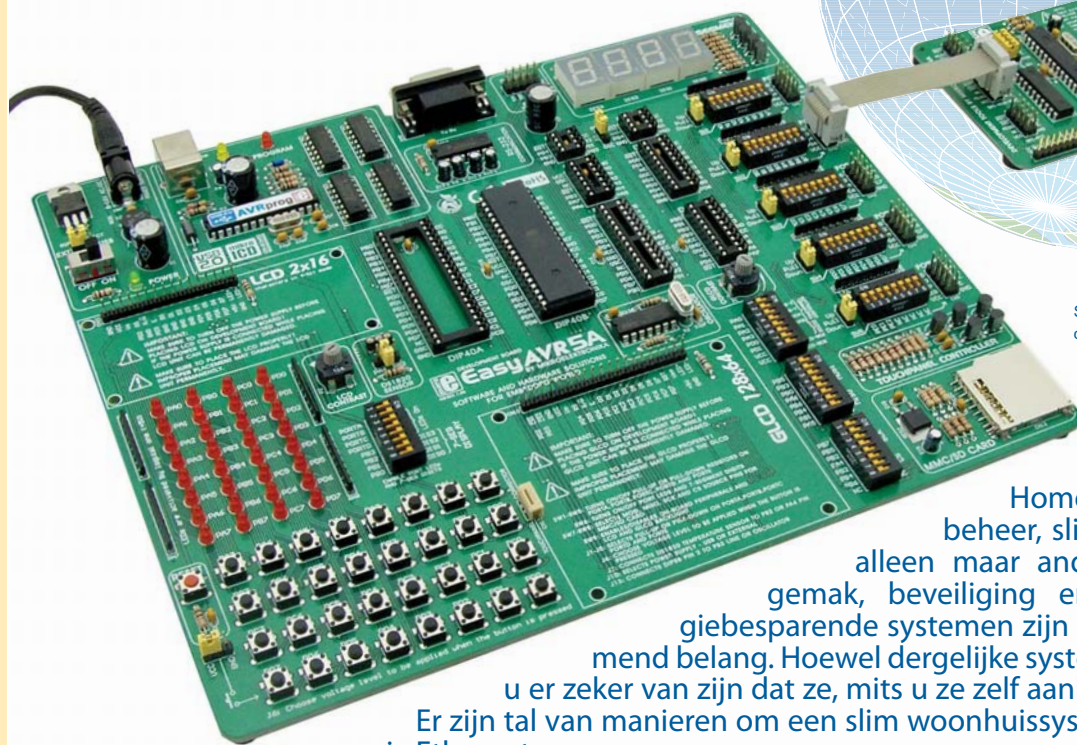


Nu hebt u ... Oké. Ethernet nodig



Seriële Ethernet-module, aangesloten op EasyAVR5A Development System

Home automation, woonhuis-beheer, slim of digitaal woonhuis zijn alleen maar andere namen voor comfort, gemak, beveiliging en energiebesparing. Energiebesparende systemen zijn vandaag de dag van toenemend belang. Hoewel dergelijke systemen erg kostbaar zijn, kunt u er zeker van zijn dat ze, mits u ze zelf aanlegt, ook erg goedkoop zijn. Er zijn tal van manieren om een slim woonhuissysteem te sturen. Eén ervan is via Ethernet.

Srdjan Tomic
MikroElektronika - Software Department

Alles wat u nodig hebt, is een Atmega 32 microcontroller en een ENC28J60 seriële Ethernet-chip. Deze chip is trouwens ook een prima oplossing voor andere microcontroller families, zoals PIC, dsPIC enzovoort. De aansluiting op het Ethernet-netwerk komt tot stand via een CviLux CJCBA8HF1Y0 RJ-45 connector. De op de PORTA.0 aangesloten LED simuleert een huishoudelijk apparaat dat we willen regelen.

De *microPASCAL for AVR* -compiler bevat de SPI_Ethernet-bibliotheek die het schrijven van een programma voor de microcontroller aanzienlijk vereenvoudigt. Met enkele routines uit deze bibliotheek kunt u een programma opstellen waarmee u de elektrische huishoudelijke apparaten in uw huis via een webbrowser bestuurt.

Dit programma moet de volgende bewerkingen uitvoeren:

- Stap 1.** Aanmaken van een html-pagina om de microcontroller aan te sturen. Deze wordt in de code geïmporteerd in de vorm van een tekenreeks.
- Stap 2.** Instellen van de door uw internetprovider verstrekte IP-, DNS- en gateway-adressen en het subnetmasker.

De parameters van uw lokale netwerk kunnen bijvoorbeeld luiden:

IP: 192.168.20.60 (Controlesysteem-adres)
DNS: 192.168.20.1 (Domain Name System-adres)
GATEWAY: 192.168.20.6 (gateway-adres)
SUBNET: 255.255.255.0 (subnetmasker)

- Stap 3.** Uitschakelen van de analoge PORTA ingangen. De microcontroller-pen moet vrijgemaakt en als output geconfigureerd worden.
- Stap 4.** Initialiseren van de SPI-module van de Atmega 32 microcontroller.
- Stap 5.** Initialiseren van de ENC28J60-chip op de module.
- Stap 6.** Schrijven van de code binnen de functie Spi_Ethernet-userTCP die, na ontvangst van een opdracht via de webbrowser, de op de PORTA.0 aangesloten LED in- of uitschakelt.
- Stap 7.** De ontvangen gegevens in een eindeloze lus inlezen.

Het belangrijkste onderdeel van het programma is de functie Spi_Ethernet-UserTCP die alle ontvangen opdrachten verwerkt.

Ontvangt de webbrowser een "GET"-verzoek dat

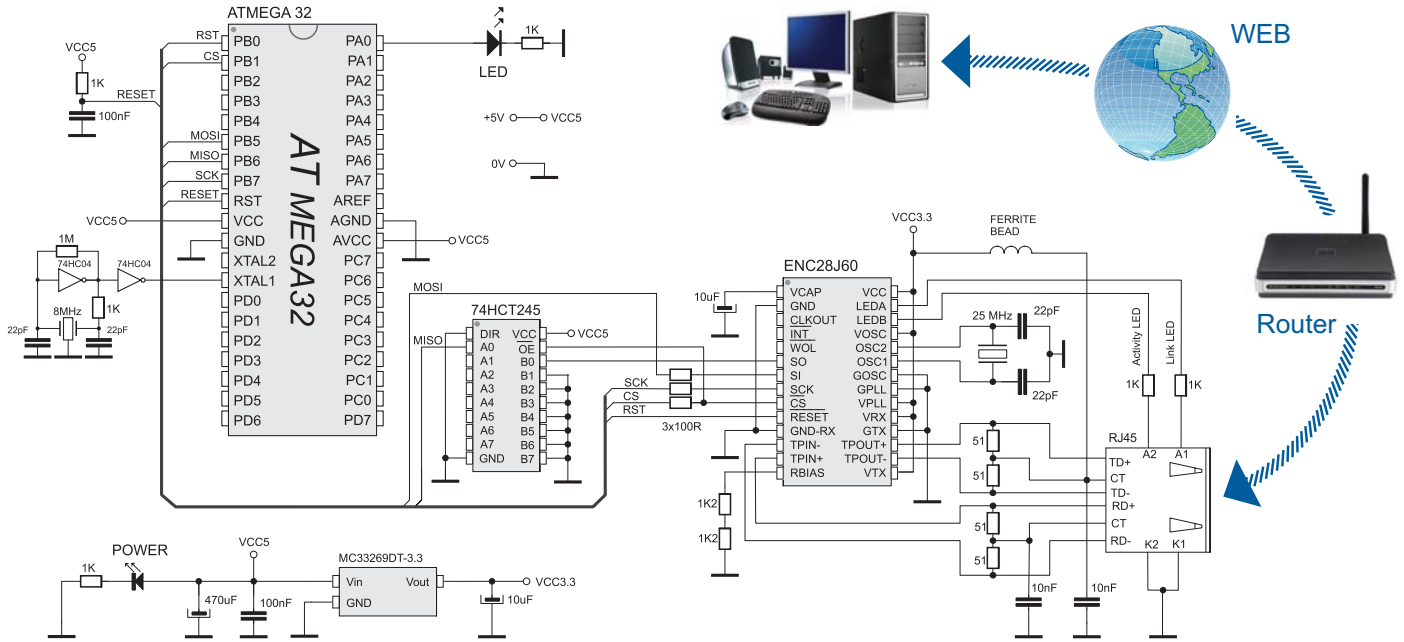
door uw computer naar het IP-adres van het besturingssysteem is gezonden, dan reageert de microcontroller door een in zijn geheugen opgeslagen webpagina af te geven. Deze pagina wordt dan door de browser automatisch afgebeeld op het computerscherm.

Wordt de opdracht "ON" ontvangen, dan wordt de op PORTA.0 aangesloten LED ingeschakeld. Op dezelfde manier wordt bij ontvangst van de opdracht "OFF", de LED uitgeschakeld. Sluit u een relais in plaats van een LED aan, dan kunt u een of ander huishoudelijk apparaat besturen, bijvoorbeeld de verlichting, de beveiliging, de verwarming enzovoort.

Het besturen van een of ander huishoudelijk apparaat bestaat uit het in de webbrowser invoeren van het IP-adres van het controlesysteem en het specificeren van de gewenste opdrachten. Uiteraard is het mogelijk meer dan een microcontrollerpen aan te sturen, zodat u ook een



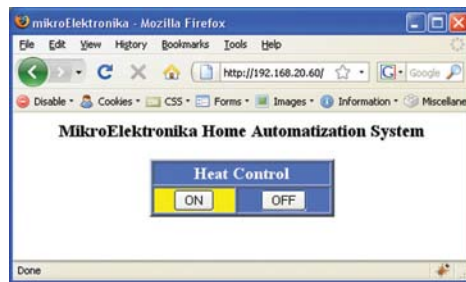
Figuur 1. MikroElektronika's Seriële Ethernet module met ENC28J60-chip



Schema 1. Aansluiten van de Seriële Ethernet-module op een Atmega 32

Voorbeeld 1: Programma om sturing via Ethernet te demonstreren

groot aantal huishoudelijke apparaten als een compleet automatiseringssysteem kunt aansturen.



De schermafdruk illustreert de door de webbrowser getoonde webpagina als het IP-adres van het controlesysteem in ons voorbeeld wordt ingevoerd. Op de ON- of OFF-knop klikken schakelt de LED in en uit en simuleert op die manier het verwarmingssysteem.

```

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
    getReq : array[20] of byte ; // HTTP request buffer

// mE Ethernet NIC pinout
SPI_Ethernet_Rst : sbit at PORTB.0;
SPI_Ethernet_CS : sbit at PORTB.1;
SPI_Ethernet_Rst_Direction : sbit at DDRB.0;
SPI_Ethernet_CS_Direction : sbit at DDRB.1;
// 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
        SPI_Ethernet_putByte(s^); s := s + 1; result := result + 1;
    end;
end;

function putString(s : ^byte) : word;
begin
    result := 0;
    while(s^ <> 0) do
        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) : word;
var len : word ; // my reply length
begin
    result := string(10);
    if(localPort <= 80) then // I listen only to web request on port 80
        begin
            result := 0; exit;
        end;
    // get 10 first bytes only of the request, the rest does not matter here
    for len := 0 to 14 do getReq[RequestLen] := SPI_Ethernet_getByte();
    getReq[RequestLen] := 0;

    tmp := 'GET /';
    if(memcmp(getReq, @tmp, 5) < 0) then // only GET method
        begin
            result := 0; exit;
        end;
    tmp := 'ON';
    if(memcmp(getReq+11, @tmp, 2) = 0) then // do we have ON command
        PORTA.B0 := 1 // set PORTA bit 0
    else
        begin
            tmp := 'OFF';
            if(memcmp(getReq+11, @tmp, 3) = 0) then // do we have OFF command
                PORTA.B0 := 0; // clear PORTA bit 0
            end;
        end;
    if(PORTA.B0) then
        begin
            tmp := "#FFFFFF0"; memcpy(@indexPage+340, @tmp, 6); // highlight (yellow) ON
            tmp := "#4974E2"; memcpy(@indexPage+431, @tmp, 6); // highlight (yellow) OFF
        end;
    else
        begin
            tmp := "#4974E2"; memcpy(@indexPage+340, @tmp, 6); // clear ON
            tmp := "#FFFFFF0"; memcpy(@indexPage+431, @tmp, 6); // highlight (yellow) OFF
        end;
    len := putConstString(httpHeader); // with HTML MIME type
    len := len + putConstString(httpMimeTypeHTML); // HTML page first part
    result := len + putString(indexPage); // 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) : word;
begin
    result := 0; // back to the library with the length of the UDP reply
end;

begin
    PORTA.B0 bit = 0; // set PORTA as output
    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></td></tr>+
    <tr><td align=center colspan=2><input type="button" value="ON" width=60 +
    <type="submit" value="ON"></td><td align=center bgcolor=#FFFFFF0; +
    <input type="button" value="OFF" width=60 +
    </tr></tbody></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;

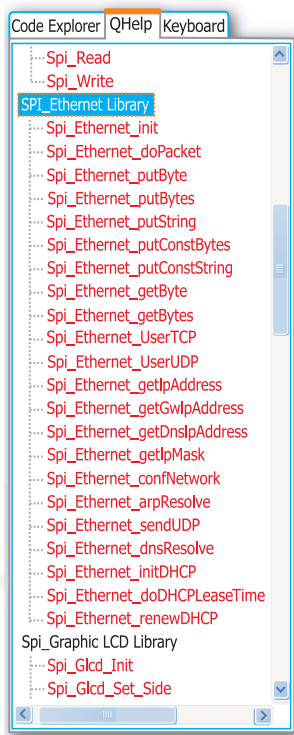
// starts ENC28J60 with: reset bit on PORTB.F0, CS bit on PORTB.F1,
SPI_Init_Advance(SPI_MASTER, SPI_FCY_DIV4, SPI_CLK_LO_LEADING);
SPI_Rd_Prt := @SPI_Read;
SPI_Ethernet_UserTCP_Prt := @SPI_Ethernet_UserTCP;
SPI_Ethernet_UserUDP_Prt := @SPI_Ethernet_UserUDP;
SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX);

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

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

```

Onderstaande lijst van kant en klare functies is opgenomen in de SPI Ethernet Library. Deze bibliotheek maakt deel uit van de microPASCAL for AVR compiler.



Spi_Ethernet_Init()	ENC28J60-controller initialiseren
Spi_Ethernet_Enable()	Netwerkverkeer inschakelen
Spi_Ethernet_Disable()	Netwerkverkeer uitschakelen
Spi_Ethernet_doPacket()	Ontvangen pakket verwerken
Spi_Ethernet_putByte()	Een byte opslaan
Spi_Ethernet_putBytes()	Bytes opslaan
Spi_Ethernet_putConstBytes()	Bytes continu opslaan
Spi_Ethernet_putString()	Tekenreeks opslaan
Spi_Ethernet_putConstString()	Tekenreeks continu opslaan
Spi_Ethernet_getByte()	Een byte ophalen
Spi_Ethernet_getBytes()	Bytes ophalen
Spi_Ethernet_UserTCP()	TCP-code afhandelen
Spi_Ethernet_UserUDP()	UDP-code afhandelen
Spi_Ethernet_getIpAddress()	IP-adres ophalen
Spi_Ethernet_getGwIpAddress()	Gateway-adres ophalen
Spi_Ethernet_getDnsIpAddress()	DNS-adres ophalen
Spi_Ethernet_getIpMask()	IP-masker ophalen
Spi_Ethernet_confNetwork()	Netwerkparameters instellen
Spi_Ethernet_arpResolve()	ARP-verzoek verzenden
Spi_Ethernet_sendUDP()	UDP-pakket verzenden
Spi_Ethernet_dnsResolve()	DNS-verzoek verzenden
Spi_Ethernet_initDHCP()	HCP-verzoek verzenden
Spi_Ethernet_doDHCPLeaseTime()	Verwerk lease time
Spi_Ethernet_renewDHCP()	Verzoek tot DHCP-vernieuwing

In het programma gebruikte *.SPI Ethernet Library-functies

Spi_Init()	Microcontroller SPI-module initialiseren
memcpy()	Microcontroller RAM-geheugenplaatsen kopiëren
memcmp()	Microcontroller RAM-geheugenplaatsen vergelijken

NOTE: De voor dit voorbeeld in C, Basic en Pascal voor AVR® microcontrollers geschreven code staan, evenals de voor dsPIC® en PIC® microcontrollers geschreven programma's, op onze website: www.mikroe.com/en/article/

