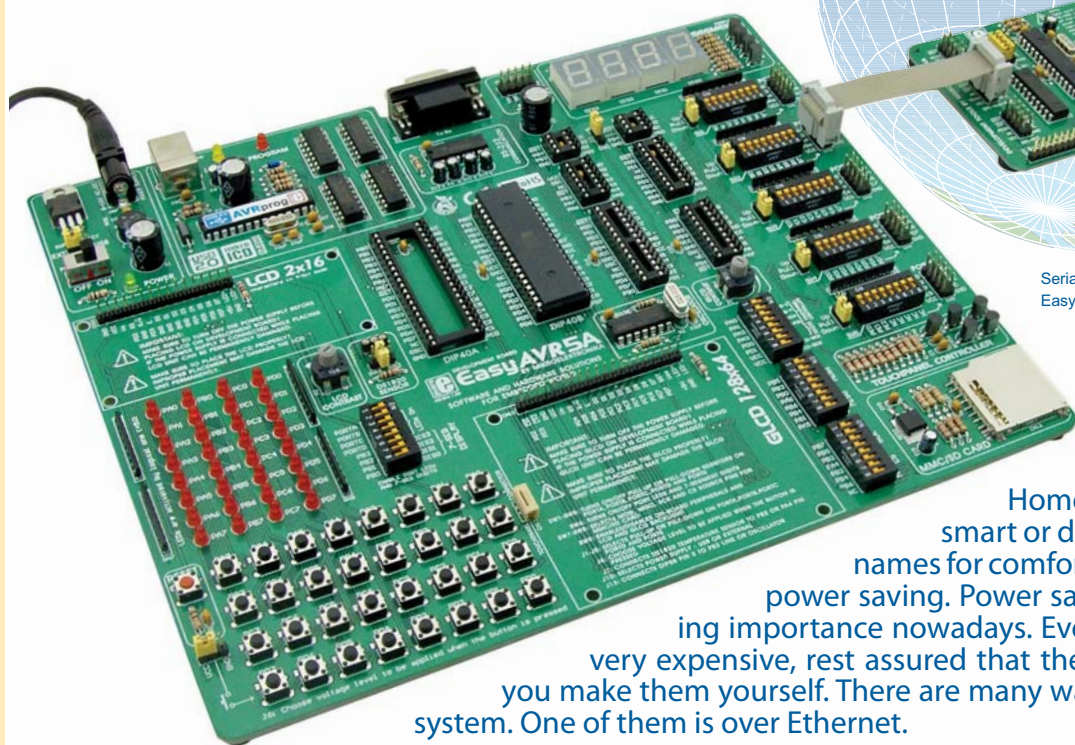


OK. Now you need ... ETHERNET



Serial Ethernet module connected to EasyAVR5A Development System

Home automation, home control, smart or digital home are just different names for comfort, convenience, security and power saving. Power saving systems are of increasing importance nowadays. Even though such systems are very expensive, rest assured that they can also be very cheap if you make them yourself. There are many ways to control a smart home system. One of them is over Ethernet.

By Srdjan Tomic
MikroElektronika - Software Department

All you need is a Atmega 32 microcontroller and an ENC28J60 serial Ethernet chip. This chip is a great solution for other microcontroller families as well such as PIC, dsPIC etc. The CviLux CJCBA8HF1Y0 RJ-45 connector is used for connection to the Ethernet network. An LED connected to the microcontroller PORTB. pin0 simulates a home appliance we want to control.

The *mikroBASIC for AVR* compiler contains the *SPI_Ethernet* library that will considerably simplify the process of writing a program for the microcontroller. By using a few routines from this library, it is possible to create the program that will enable electrical appliances in your home to be controlled via a web browser.

It is necessary to perform the following operations within the program:

- Step 1.** Create an html page to run the microcontroller through. Import it in the code as a string.
- Step 2.** Set IP, DNS, Gateway addresses and Subnet mask obtained from your internet provider.

For example, our local network parameters are as follows:

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

- Step 3.** Disable PORTA analogue inputs. The microcontroller pin should be cleared and configured as an output.
- Step 4.** Initialize the SPI module of the Atmega 32 microcontroller.
- Step 5.** Initialize the Serial Ethernet module chip ENC28J60.
- Step 6.** Write the code within the *Spi_Ethernet_userTCP* function that will, after receiving command via web browser, turn on/off the LED connected to the PORTA.0.
- Step 7.** Read received data in an endless loop.

The most important part of the program is the *Spi_Ethernet_userTCP* function, processing all received commands

After the web browser "GET" request is received, sent from your computer to the control system IP address, the microcontroller will respond with a web page stored in its memory. This page will then be automatically displayed on the computer screen by the browser.

When the ON command is received, the LED connected to the PORTA.0 will be turned on.

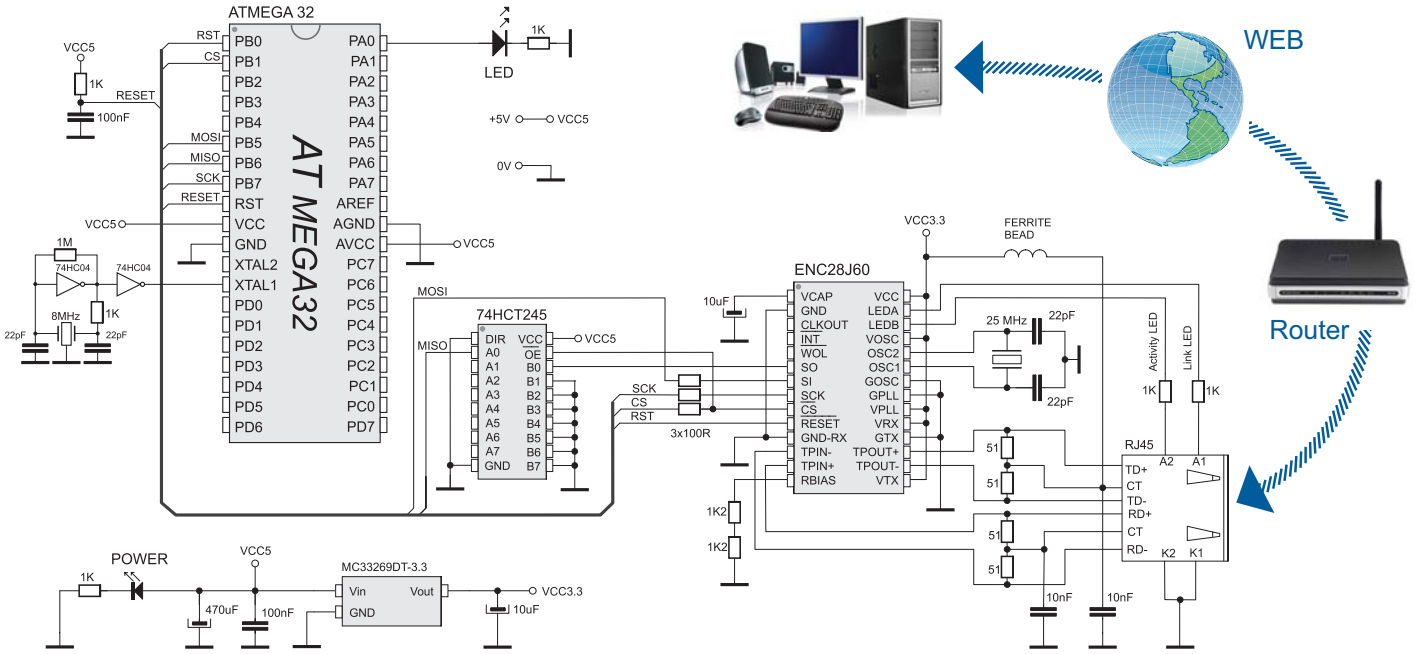
Likewise, when the OFF command is received the LED will be turned off. If you have a relay instead of LED, it is possible to control any appliance such as lighting, security system, heating system etc.

The control of any home appliance consists of entering control system IP address in the web browser and specifying the desired commands.

Of course, it is possible to control more than

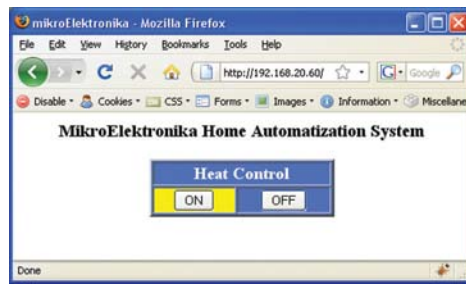


Figure 1. MikroElektronika's Serial Ethernet module with ENC28J60 chip



Schematic 1. Connecting the Serial Ethernet module to a Atmega 32

one microcontroller pin, which enables you to govern a large number of appliances or complete home automation system as well.



The screendump illustrates the web page displayed by the web browser after entering the control system IP address. In our example, ON and OFF button clicks cause the LED to be turned on and off, thus simulating the heating control system.

Example 1: Program to demonstrate control over Ethernet

```

program enc_ethernet
dim myMacAddr as byte(6)
myIpAddr as byte(4)
gwIpAddr as byte(4)
ipMask as byte(4)
dnsIpAddr as byte(4)
indexPage as string(523)
getRequest as byte(20)
httpHeader as string(30)
httpMimeTypeHTML as string(13)
httpMimeTypeScript as string(14)

' my MAC address
' my IP address
' gateway (router) IP address
' network mask (for example : 255.255.255.0)
' DNS server IP address
' default html page
' HTTP request buffer
' HTTP header
' HTML MIME type
' 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^a)
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
dim cnt as word
tmp as string(10)

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

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

tmp = "GET/"
if(memcpy(@getRequest, @tmp, 5) > 0) then
' only GET method
result = 0
exit
endif

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

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
endif

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

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

main:
PORTA bit 0 = 0
DDRA.B0 = 1
' set PORTA as output
' clear PORTA.B0
' set PORTA.B0 as output (rele 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 Automatization 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>ON</td><td align=center colspan=2><font size=4 color=white>OFF</td></tr></table>"+
"</td></tr><td align=center bgcolor=#4974E2><input name="+chr(34)+"tst1"+
chr(34)+" 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] = 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

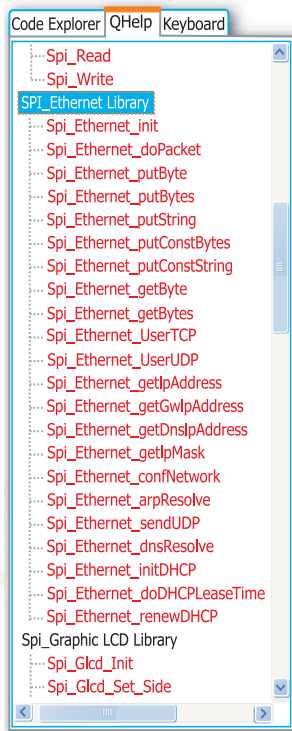
' starts ENC28J60 with: reset bit on PORTB.F0, CS bit on PORTB.F1,
' my MAC & IP address, full duplex
SPI_Init_Advanced(SPI_MASTER, SPI_FCY_DIV4, SPI_CLK_LO_LEADING)
SPI_Read_Ptr = @SPI_Read
SPI_Ethernet_UserTCP_Ptr = @SPI_Ethernet_UserTCP
SPI_Ethernet_UserUDP_Ptr = @SPI_Ethernet_UserUDP
SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX)

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

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

```

Below is a list of ready to use functions contained in the SPI Ethernet Library. This library is integrated in mikroBASIC for AVR compiler.



Spi_Ethernet_Init(*)	Init ENC28J60 controller
Spi_Ethernet_Enable()	Enable network traffic
Spi_Ethernet_Disable()	Disable network traffic
Spi_Ethernet_doPacket(*)	Process received packet
Spi_Ethernet_putByte()	Store a byte
Spi_Ethernet_putBytes()	Store bytes
Spi_Ethernet_putConstBytes()	Store const bytes
Spi_Ethernet_putString(*)	Store string
Spi_Ethernet_putConstString(*)	Store const string
Spi_Ethernet_getByte(*)	Fetch a byte
Spi_Ethernet_getBytes()	Fetch bytes
Spi_Ethernet_UserTCP(*)	TCP handling code
Spi_Ethernet_UserUDP()	UDP handling code
Spi_Ethernet_getIpAddress()	Get IP address
Spi_Ethernet_getGwIpAddress()	Get Gateway address
Spi_Ethernet_getDnsIpAddress()	Get DNS address
Spi_Ethernet_getIpMask()	Get IP mask
Spi_Ethernet_confNetwork(*)	Set network parameters
Spi_Ethernet_arpResolve()	Send an ARP request
Spi_Ethernet_sendUDP()	Send an UDP packet
Spi_Ethernet_dnsResolve()	Send an DNS request
Spi_Ethernet_initDHCP()	Send an DHCP request
Spi_Ethernet_doDHCPLeaseTime()	Process lease time
Spi_Ethernet_renewDHCP()	DHCP renewal request

* SPI Ethernet Library functions used in program

Other mikroBASIC for AVR functions used in program:	
Spi_Init()	Initialize microcontroller SPI module
memcpy()	Copy microcontroller RAM memory locations
memcmp()	Compare microcontroller RAM memory locations

NOTE: Code for this example written for AVR microcontrollers in C, Basic and Pascal as well as the programs written for dsPIC and PIC microcontrollers can be found on our web site: www.mikroe.com/en/article/

