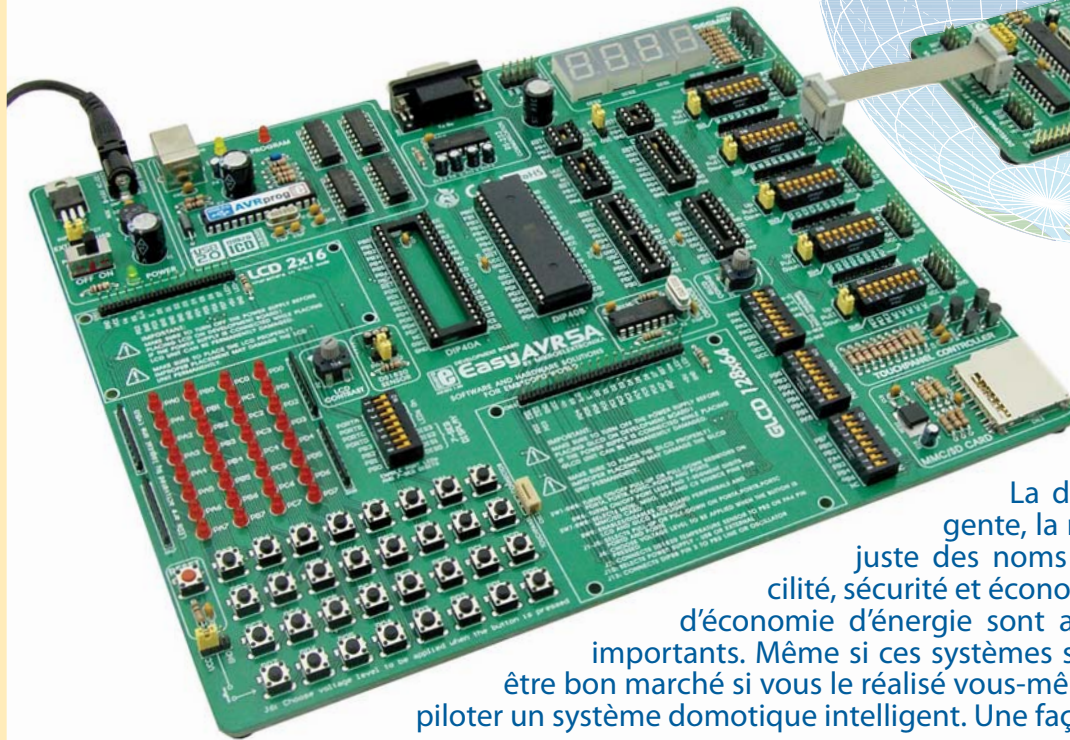


Maintenant il vous faut...

Bon. ETHERNET



Module Ethernet sériel connecté au système de développement EasyAVR5A

La domotique, la maison intelligente, la maison numérique, etc. sont juste des noms différents pour confort, facilité, sécurité et économies d'énergie. Les systèmes d'économie d'énergie sont aujourd'hui de plus en plus importants. Même si ces systèmes sont très coûteux, il peuvent être bon marché si vous le réalisez vous-même. Il y a plusieurs façons de piloter un système domotique intelligent. Une façon est d'utiliser Ethernet.

Par Srdjan Tomic
MikroElektronika - Software Department

Tout ce dont vous avez besoin est un microcontrôleur Atmega 32 et une puce Ethernet sérielle ENC28J60. Ce composant est aussi une excellente solution pour d'autres microcontrôleurs comme les PIC, dsPIC etc. Le connecteur CviLux CJCBA8HF1Y0 RJ-45 est utilisé pour la connexion avec le réseau Ethernet. Une LED connectée à PORTB.pin0 du microcontrôleur simule l'appareil ménager à piloter.

Le compilateur *mikroBASIC for AVR* possède une librairie `SPI_Ethernet` qui simplifie beaucoup le développement du programme pour le microcontrôleur. En utilisant quelques fonctions de cette librairie, il est possible de créer un programme qui permet de commander vos appareils ménagers électriques par un navigateur Internet.

Suivez les étapes suivantes dans le programme:

Étape 1. Créez une page HTML pour piloter le microcontrôleur. Insérez-la dans le programme comme une trame de caractères.

Étape 2. Saisissez les adresses IP, DNS et Passerelle et le masque Subnet fourni par votre fournisseur Internet.

Voici un exemple de paramètres d'un réseau local:

IP : 192.168.20.60 (adresse du système de pilotage)

DNS : 192.168.20.1 (adresse du serveur des noms de domaines)

GATEWAY : 192.168.20.6 (adresse Passerelle)

SUBNET : 255.255.255.0 (masque Subnet)

Étape 3. Désactivez les entrées analogiques PORTA. Configurez les broches du microcontrôleur comme sortie et mettez-les à un niveau logique bas.

Étape 4. Initialisez le module SPI du microcontrôleur Atmega 32.

Étape 5. Initialisez la puce Ethernet sériel ENC28J60.

Étape 6. Écrivez le code dans la fonction `SPI_Ethernet_userTCP` qui, après avoir reçu une commande par le navigateur Internet, allumera ou éteindra la LED connectée à PORTA.0.

Étape 7. Lisez les données reçues dans une boucle infinie.

La fonction `SPI_Ethernet_userTCP` est la plus importante du programme, c'est elle qui traite toutes les commandes reçues.

Après la réception d'une commande « GET », envoyé par le navigateur Internet sur votre ordinateur vers l'adresse IP du système de contrôle, le microcontrôleur répondra avec une page web stockée dans sa mémoire. Cette page est automatiquement affichée sur l'écran de l'ordinateur par le navigateur. Quand la commande ON est reçue, la LED connectée à PORTA.0 sera allumée. Pareil, quand la commande OFF est reçue, la LED sera éteinte. Si vous utilisez un relais à la place de la LED, il est possible de piloter toute sorte d'appareils, comme une lumière, un système d'alarme, le chauffage, etc.

Le pilotage d'un appareil se fait en saisissant l'adresse IP du système de contrôle dans le



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

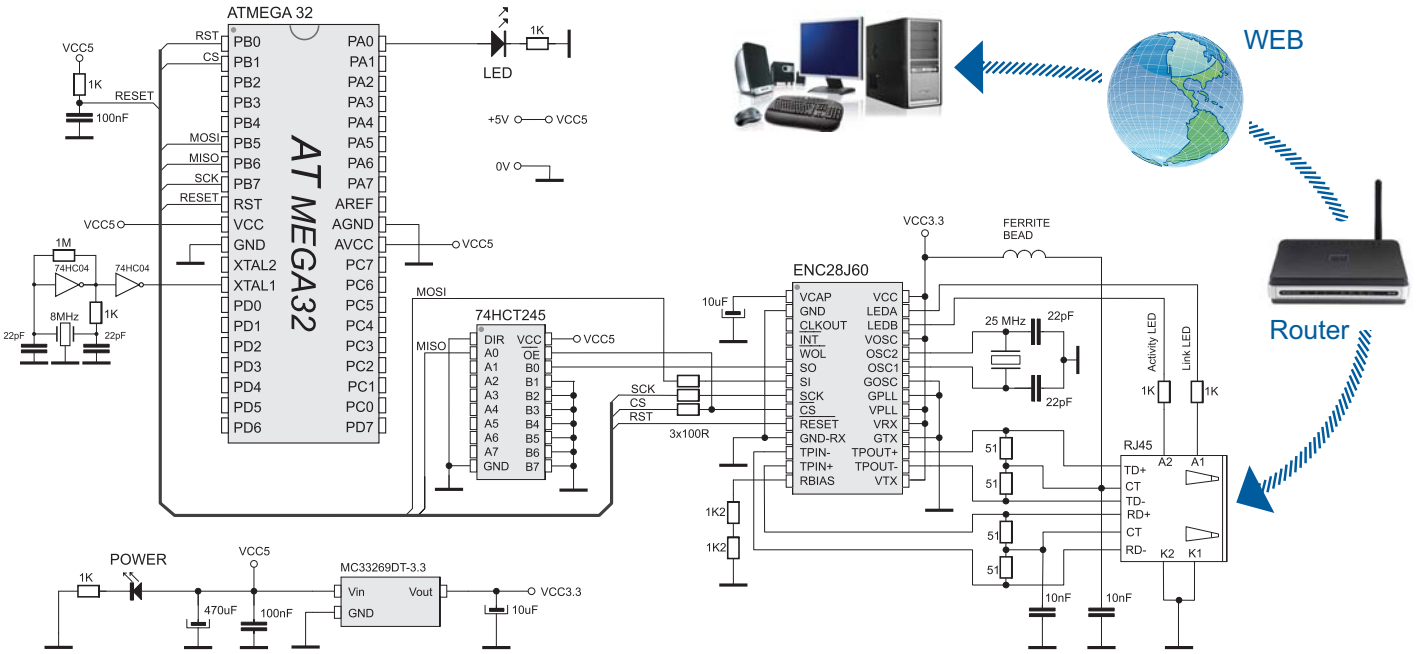
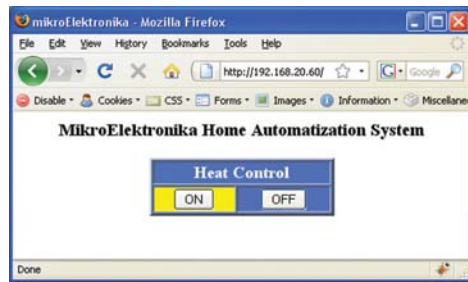


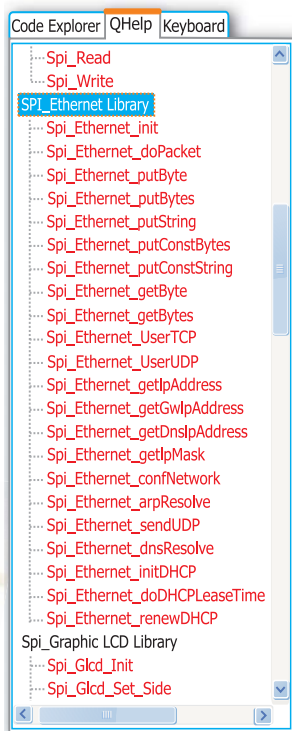
Schéma 1. Comment connecter le module Ethernet sériel à un Atmega 32

navigateur Internet et de spécifier les commandes souhaitées. Bien sûr, il est possible de piloter plus d'une broche du microcontrôleur, ce qui vous permet de commander un grand nombre d'appareils ou un système domotique complet.



La photo d'écran montre la page web affichée par le navigateur Internet après la saisie de l'adresse IP du système de contrôle. Dans notre exemple, les clics sur les boutons ON et OFF changent l'état de la LED, ainsi simulant le système de chauffage.

Voici la liste des fonctions déjà disponibles dans la librairie SPI Ethernet (ci-dessous). Cette librairie est fournie avec le compilateur mikroBASIC pour AVR.



Spi_Ethernet_Init()	Initialiser contrôleur ENC28J60
Spi_Ethernet_Enable()	Activer trafic réseau
Spi_Ethernet_Disable()	Désactiver trafic réseau
Spi_Ethernet_doPacket()	Traiter un paquet reçu
Spi_Ethernet_putByte()	Stocker un octet
Spi_Ethernet_putBytes()	Stocker plusieurs octets
Spi_Ethernet_putConstBytes()	Stocker plusieurs octets constants
Spi_Ethernet_putString()	Stocker une trame de caractères
Spi_Ethernet_putConstString()	Stocker une trame de caractères constantes
Spi_Ethernet_getByte()	Lire un octet
Spi_Ethernet_getBytes()	Lire plusieurs octets
Spi_Ethernet_UserTCP()	Gestionnaire TCP
Spi_Ethernet_UserUDP()	Gestionnaire UDP
Spi_Ethernet_getIpAddress()	Lire l'adresse IP
Spi_Ethernet_getGwIpAddress()	Lire l'adresse Passerelle
Spi_Ethernet_getDnsIpAddress()	Lire l'adresse DNS
Spi_Ethernet_getIpMask()	Lire le masque IP
Spi_Ethernet_confNetwork()	Saisir les paramètres réseau
Spi_Ethernet_arpResolve()	Envoyer une requête ARP
Spi_Ethernet_sendUDP()	Envoyer un paquet UDP
Spi_Ethernet_dnsResolve()	Envoyer une requête DNS
Spi_Ethernet_initDHCP()	Envoyer une requête DHCP
Spi_Ethernet_doDHCPLeaseTime()	Gérer temps de bail
Spi_Ethernet_renewDHCP()	Rafraîchir requête DHCP
* Fonctions de la librairie SPI Ethernet utilisées dans le programme	
Autres fonctions de mikroBASIC pour AVR utilisées dans le programme:	
Spi_Init()	Initialiser module SPI du microcontrôleur
memcpy()	Copier de la mémoire RAM du microcontrôleur
memcmp()	Comparer de la mémoire RAM du microcontrôleur

Exemple 1 : Exemple d'un programme de pilotage par 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(53)      ' default html page
getRequest 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
'mE ethernet NIC definitions

sub function putString(dim s as ^byte) as word
result = 0
while(s <> 0)
  Spi_Ethernet_putByte(s^a)
  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(memcmp(@getRequest, @tmp, 5) < 0) then ' only GET method
  result = 0
  exit
endif

tmp = "ON"
if(memcmp(@getRequest+11, @tmp, 2) = 0) then ' do we have ON command
  PORTA.B0 = 1
  set PORTA bit 0
else
  tmp = "OFF"
if(memcmp(@getRequest+11, @tmp, 3) = 0) then ' do we have OFF command
  PORTA.B0 = 0
  clear PORTA bit 0
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
with HTML MIME type
cnt = cnt + putString(@httpMimeTypeHTML)
result = cnt - 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
dim remotePort, destPort, reqLength as word
back to the library with the length of the UDP reply
end sub

main:
PORTA0_bit = 0 ' set PORTA as output
DDRA.B0 = 1 ' 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 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] = 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
myGwAddr[0] = 192 myGwAddr[1] = 168 myGwAddr[2] = 20 myGwAddr[3] = 60
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, myGwAddr, dnsIpAddr)

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

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

```

NOTE: Les codes source de cet exemple en C, BASIC et PASCAL pour microcontrôleurs AVR®, ainsi que tous les programmes écrits pour les microcontrôleurs dsPIC® et PIC® sont disponibles sur notre site Internet : www.mikroe.com/en/article/

