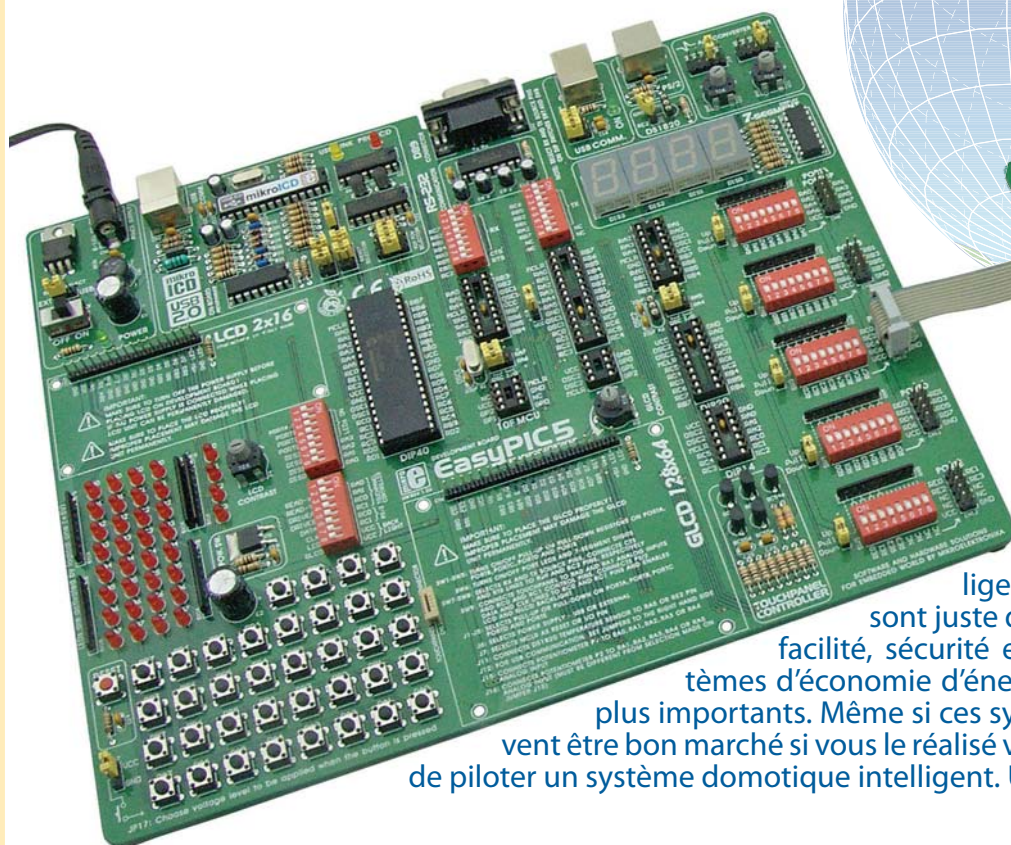


# Bon. ETHERNET

## Maintenant il vous faut...



Module Ethernet sériel connecté au système de développement EasyPIC5

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 PIC18F4520 et une puce Ethernet sérielle ENC28J60. Ce composant est aussi une excellente solution pour d'autres microcontrôleurs comme les AVR, dsPIC etc. Le connecteur CviLux CJCBA8HF1Y0 RJ-45 est utilisé pour la connexion avec le réseau Ethernet. Une LED connectée à PORTB.0 du microcontrôleur simule l'appareil ménager à piloter.

Le compilateur *mikroPASCAL for PIC* 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 PORTB. Configurez les broches du microcontrôleur comme sortie et mettez-les à un niveau logique bas.
- Étape 4.** Initialisez le module SPI du microcontrôleur PIC18F4520.
- Étape 5.** Initialisez la puce Ethernet sérielle 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 PORTB.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 PORTB.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.



Figure 1. MikroElektronika's Module Ethernet sériel avec une puce ENC28J60

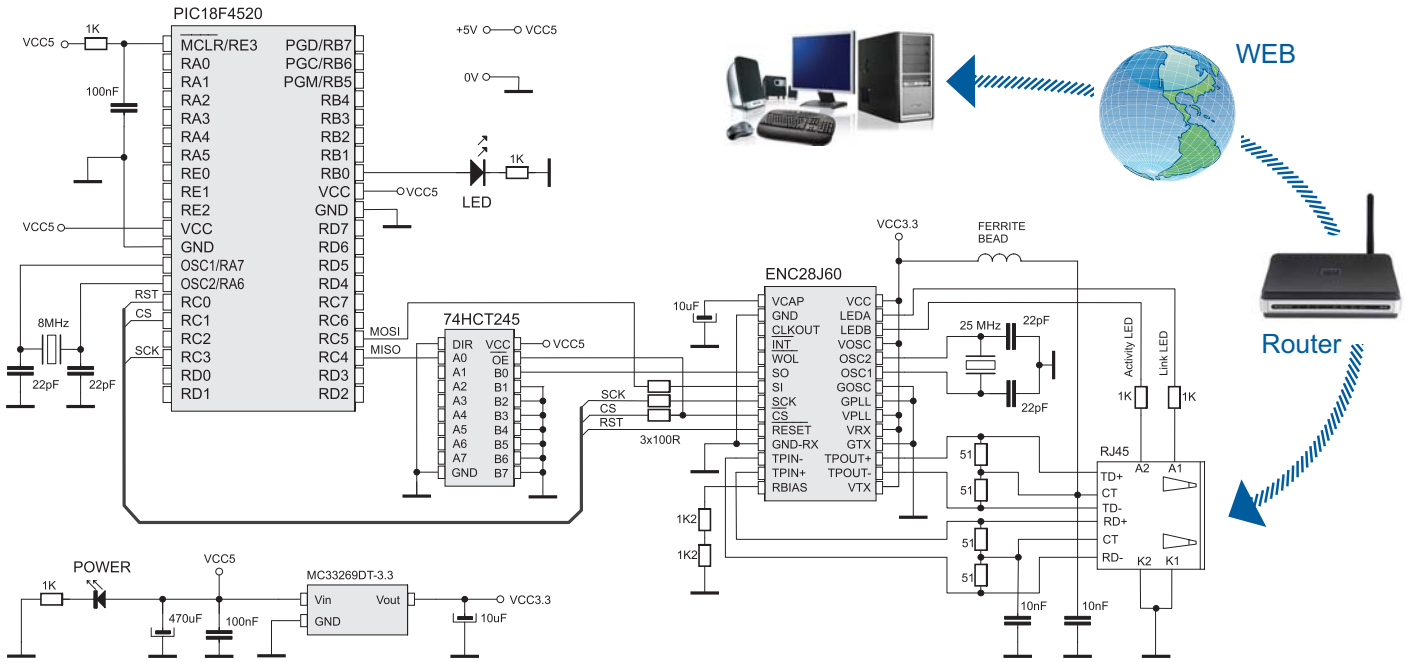
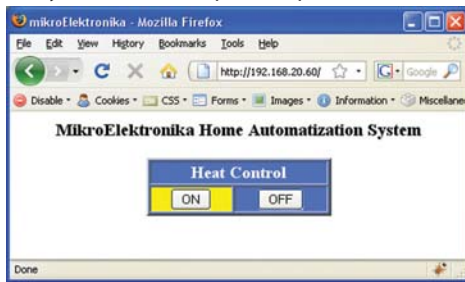


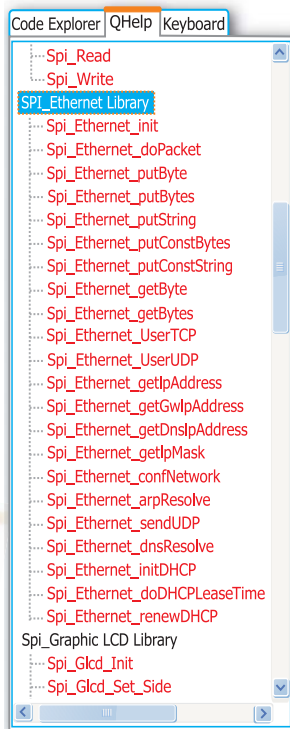
Schéma 1. Comment connecter le module Ethernet sériel à un PIC18F4520

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



<b>Spi_Ethernet_Init()</b>	Initialiser contrôleur ENC28J60
<b>Spi_Ethernet_Enable()</b>	Activer trafic réseau
<b>Spi_Ethernet_Disable()</b>	Désactiver trafic réseau
<b>Spi_Ethernet_doPacket()</b>	Traiter un paquet reçu
<b>Spi_Ethernet_putByte()</b>	Stocker un octet
<b>Spi_Ethernet_putBytes()</b>	Stocker plusieurs octets
<b>Spi_Ethernet_putConstBytes()</b>	Stocker plusieurs octets constants
<b>Spi_Ethernet_putString()</b>	Stocker une trame de caractères
<b>Spi_Ethernet_putConstString()</b>	Stocker une trame de caractères constantes
<b>Spi_Ethernet_getByte()</b>	Lire un octet
<b>Spi_Ethernet_getBytes()</b>	Lire plusieurs octets
<b>Spi_Ethernet_UserTCP()</b>	Gestionnaire TCP
<b>Spi_Ethernet_UserUDP()</b>	Gestionnaire UDP
<b>Spi_Ethernet_getIpAddress()</b>	Lire l'adresse IP
<b>Spi_Ethernet_getGwIpAddress()</b>	Lire l'adresse Passerelle
<b>Spi_Ethernet_getDnsIpAddress()</b>	Lire l'adresse DNS
<b>Spi_Ethernet_getIpMask()</b>	Lire le masque IP
<b>Spi_Ethernet_confNetwork()</b>	Saisir les paramètres réseau
<b>Spi_Ethernet_arpResolve()</b>	Envoyer une requête ARP
<b>Spi_Ethernet_sendUDP()</b>	Envoyer un paquet UDP
<b>Spi_Ethernet_dnsResolve()</b>	Envoyer une requête DNS
<b>Spi_Ethernet_initDHCP()</b>	Envoyer une requête DHCP
<b>Spi_Ethernet_doDHCPLeaseTime()</b>	Gérer temps de bail
<b>Spi_Ethernet_renewDHCP()</b>	Rafraîchir requête DHCP

**\* Fonctions de la librairie SPI Ethernet utilisées dans le programme**

**Autres fonctions de mikroPASCAL for PIC 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 (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
    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

begin
ADCON1 := 0x0F; // no analog inputs
CMCON := 0x07; // turn off comparators

PORTB0 := 0; // set PORTB0 as output (relé 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=#FF0000>+
<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]:=60;
dnsIpAddr[0]:=192; dnsIpAddr[1]:=168; dnsIpAddr[2]:=20; dnsIpAddr[3]:=6;

// starts ENC28J60 with: reset bit on PORTC.F0, CS bit on PORTC.F1
// my MAC & IP address, full duplex
Spi_Init();
// full duplex, CRC + MAC Unicast + MAC Broadcast filtering
Spi_Ethernet_Init(PORTC.0, PORTC.1,
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(); // 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 Spi_Ethernet_UserTCP(var remoteHost : array[4] of byte;
remotePort, localPort, reqLength : word) : word;
var len : word; // my reply length
tmp : string[10];
begin
if(localPort <> 80) then // I listen only to web request on port 80
result := 0;
exit;
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
PORTB0 := 1; // set PORTB bit 0
else
begin
tmp := 'OFF';
if(memcmp(getRequest+11, @tmp, 3) = 0) then // do we have OFF command
PORTB0 := 0; // clear PORTB bit 0
end;

if (PORTB0) 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 := Spi_Ethernet_putConstString(@httpHeader);
len := len + Spi_Ethernet_putConstString(@httpMimeTypeHTML);
len := len + Spi_Ethernet_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) : word;
begin
result := 0; // back to the library with the length of the UDP reply
end.

```

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

