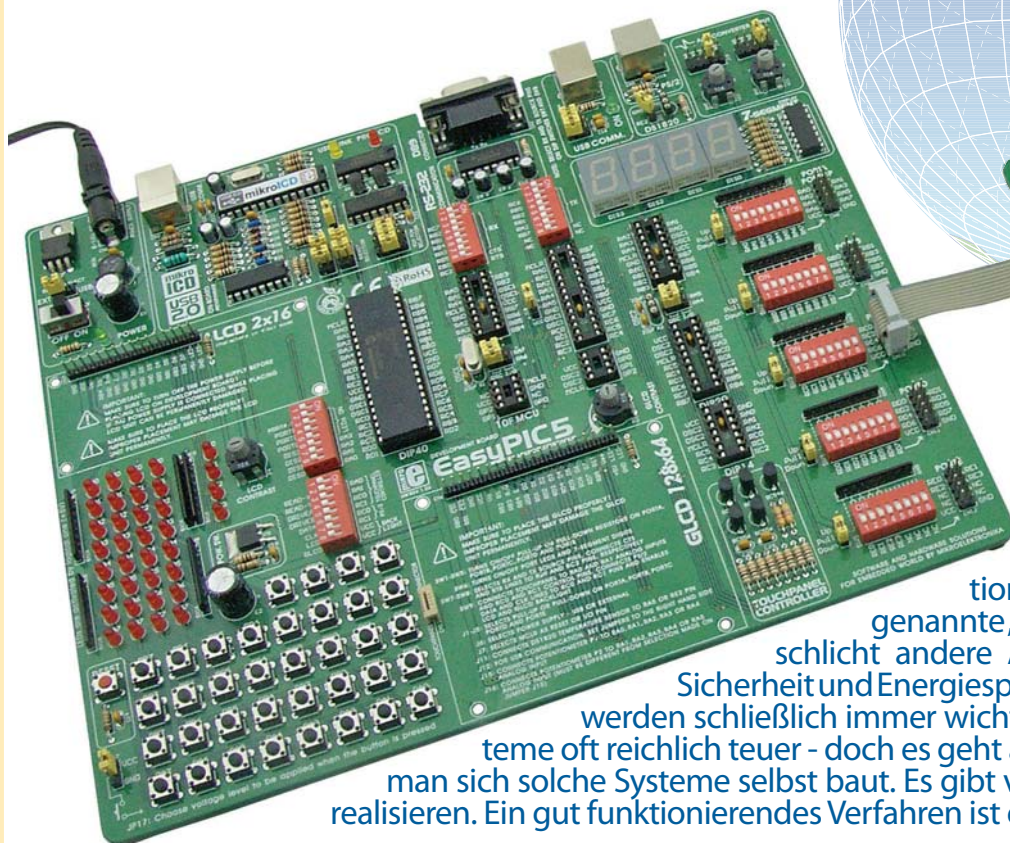


# OK. Jetzt brauchen sie... ETHERNET



Serielles Ethernet-Modul mit Anschluss an das Entwicklungs-System EasyPIC5

Heim-Automation, Heim-Steuerung oder das so genannte „smart“ oder „digital Home“ sind schlicht andere Ausdrücke für Annehmlichkeit, Sicherheit und Energiesparen - Energie sparende Systeme werden schließlich immer wichtiger. Allerdings sind solche Systeme oft reichlich teuer - doch es geht auch deutlich preiswerter, wenn man sich solche Systeme selbst baut. Es gibt viele Wege, ein „smart Home“ zu realisieren. Ein gut funktionierendes Verfahren ist es, Ethernet zu nutzen.

Von Srdjan Tomic  
MikroElektronika - Software-Abteilung

Es wird lediglich ein PIC18F4520-Mikrocontroller und ein serieller Ethernet-Chip vom Typ ENC28J60 benötigt. Dieser Chip ist eine gute Ergänzung auch für andere Mikrocontroller-Familien wie dsPIC oder die Controller von Atmel etc. Für den Anschluss an das Ethernet-Netzwerk wird eine Steckverbindung vom Typ CviLux CJCBA8HF1Y0 (RJ-45) verwendet. Eine an den Pin PORTB.0 des Mikrocontrollers angeschlossene LED simuliert das zu steuernde Gerät.

Der Compiler *mikroC for PIC* enthält die Library *SPI\_Ethernet*, welche das Schreiben von Software für den Mikrocontroller drastisch vereinfacht. Durch Verwendung nur weniger Routinen dieser Library ist es möglich, ein Programm zu erstellen, das es erlaubt, alle gesteuerten Geräte im Heim auf einfache Weise via Web-Browser fernzusteuern.

Hierzu sind folgende Schritte im Programm erforderlich:

**Schritt 1.** Erstelle eine HTML-Seite für den Mikrocontroller. Diese Seite wird dann als String importiert.

**Schritt 2.** Setzen der IP-, DNS- und Gateway-Adressen sowie der Subnetz-Maske entsprechend der Vorgaben des Internet-Providers.

Die lokalen Netzwerk-Parameter könnten zum Beispiel so aussehen:

**IP** : 192.168.20.60 (Adresse des Geräts)  
**DNS** : 192.168.20.1 (Adresse des Domain-Name-Systems)  
**GATEWAY** : 192.168.20.6 (Gateway-Adresse)  
**SUBNET** : 255.255.255.0 (Subnetz-Maske)

**Schritt 3.** Deaktivieren der analogen Eingänge von PORTB. Der jeweilige Pin wird gelöscht und als Ausgang definiert.

**Schritt 4.** Initialisieren des SPI-Moduls des PIC18F4520-Mikrocontrollers.

**Schritt 5.** Initialisieren des seriellen Ethernet-Modul-Chips ENC28J60.

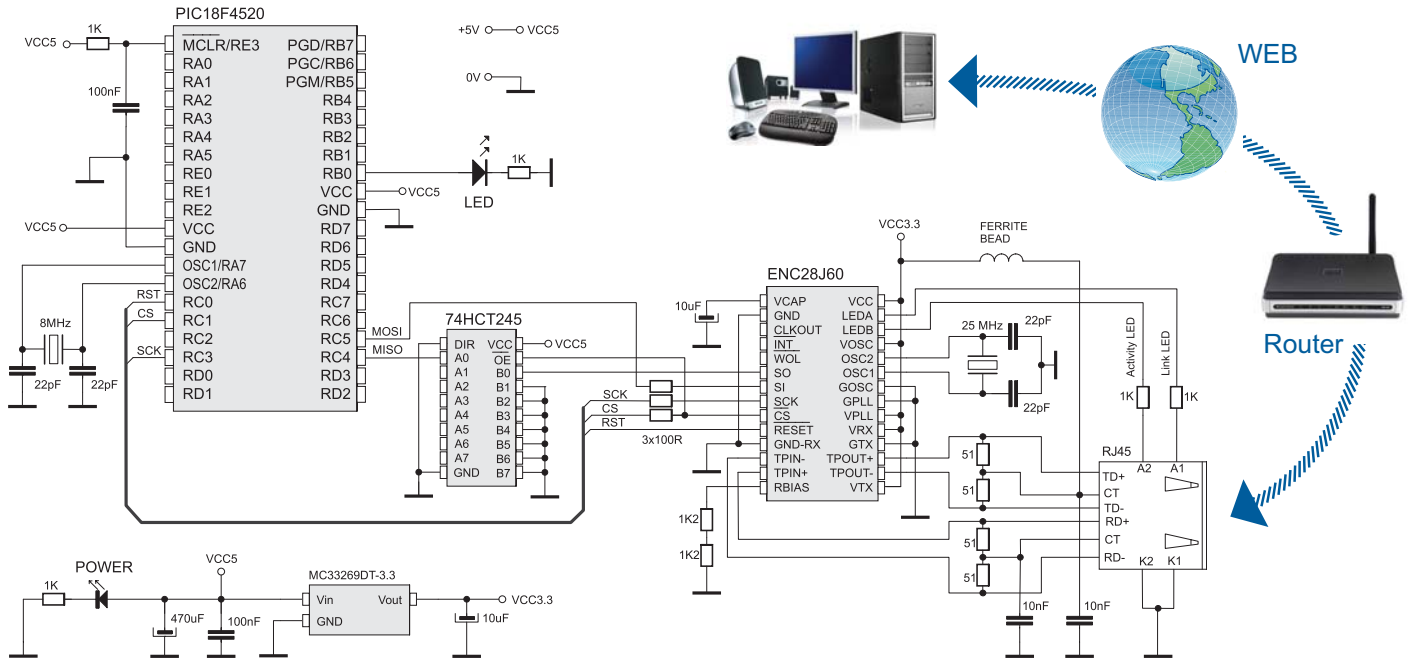
**Schritt 6.** Schreiben des Codes innerhalb der *SPI\_Ethernet\_userTCP*-Funktion, welcher nach Empfang eines Befehls via Web-Browser die an PORTB.0 angeschlossene LED ein bzw. ausschaltet.

**Schritt 7.** Lesen der zu empfangenden Daten in einer Endlos-Schleife.

Der wichtigste Teil des Programms ist die *SPI\_Ethernet\_userTCP*-Funktion, die empfangene Befehle ausführt. Nachdem eine „GET“-Anfrage des Web-Browsers empfangen wurde, die Ihr PC an die IP-Adresse des zu steuernden Geräts geschickt hat, wird der Mikrocontroller mit einer Web-Seite antworten, die in seinem Speicher abgelegt ist. Diese Web-Seite wird dann automatisch im Browser des PCs angezeigt werden. Wenn ein „ON“-Befehl empfangen wurde, wird die an PORTB.0 angeschlossene LED leuchten. Entsprechend wird ein empfangener „OFF“-Befehl die LED wieder verlöschen lassen. Ist ein Relais anstelle einer LED angeschlossen, kann das Gerät diverse andere Geräte wie Lampen, Alarmanlagen, Heizungen etc. steuern.

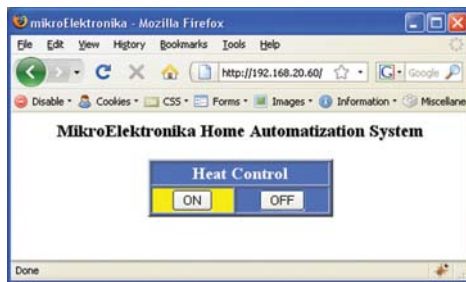


Abb. 1. MikroElektronika's Serielles Ethernet-Modul mit ENC28J60 hip



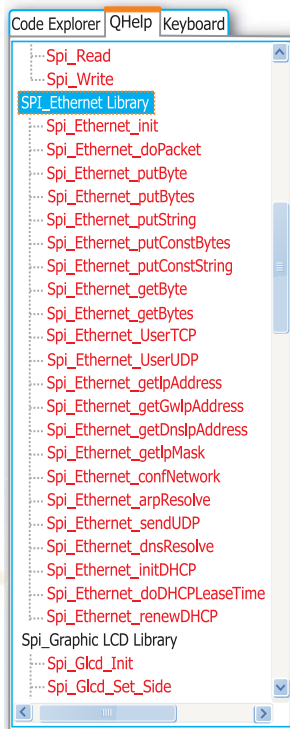
Schaltung 1. Anschluss des Serial -Ethernet-Module an einen PIC18F4520

Die Steuerung von Geräten erfolgt prinzipiell, indem die IP-Adresse des Geräts in die Adresszeile des Web-Browsers eingegeben wird und die gewünschten Befehle spezifiziert werden. Selbstverständlich kann mehr als nur ein einziger Pin eines Mikrocontrollers gesteuert werden. Auf diese Weise können viele unterschiedliche Geräte gesteuert oder auch ein komplettes Home-Automation-System realisiert werden.



Das Bildschirmfoto zeigt die Webseite, die der Web-Browser anzeigt, nachdem man die IP-Adresse eines steuerbaren Geräts eingegeben hat. In unserem Beispiel bewirken Klicks auf die Schaltflächen „ON“ und „OFF“, dass eine angeschlossene LED ein- oder ausgeschaltet wird, womit man die korrekte Funktion z.B. einer Heizung simulieren könnte.

Nachfolgend die Liste vorgefertigter Befehle, wie sie in der Library „SPI Ethernet“ enthalten sind. Diese Library ist im Compiler mikroC for PIC integriert.



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

\* Im Programm verwendete Funktionen der Library „SPI Ethernet“

**Andere im Programm verwendete Funktionen von mikroC for PIC:**

- Spi\_Init()** Initialisiere das Mikrocontroller-SPI-Modul
- memcpy()** Kopiere den RAM-Inhalt des Mikrocontrollers
- memcmp()** Vergleiche die RAM-Inhalte des Mikrocontrollers

Beispiel 1: Demo-Programm zur Steuerung via Ethernet

```

// duplex config flags
#define Spi_Ethernet_HALFDUPLEX 0x00 // half duplex
#define Spi_Ethernet_FULLLDUPLEX 0x01 // full duplex

const char httpHeader[] = "HTTP/1.1 200 OK\nContent-type: "; // HTTP header
const char httpMimeTypeHTML[] = "text/html\n\n"; // HTML MIME type
const char httpMimeTypeScript[] = "text/plain\n\n"; // TEXT MIME type

// default html page
char indexPage[] =
<html><head><title>mikroElektronika</title></head><body>\
<h3 align=center>MikroElektronika Home Automation System</h3>\
<form name="input" action="/input" method="get">\
<table align=center width=200 bgcolor=#4974E2 border=2<tr>\
<td align=center colspan=2><font size=#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>;

// network parameters
char myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f}; // my MAC address
char myIpAddr[4] = {192, 168, 20, 60}; // my IP address
char gwIpAddr[4] = {192, 168, 20, 6}; // gateway IP address
char dnsIpAddr[4] = {192, 168, 20, 1}; // dns IP address
char ipMask[4] = {255, 255, 255, 0}; // subnet mask
// end network parameters

unsigned char getRequest[20]; // HTTP request buffer

unsigned int Spi_Ethernet_UserTCP( char *remoteHost, unsigned int remotePort,
unsigned int localPort, unsigned int reqLength)
{
  unsigned int len; // my reply length
  if(localPort != 80) return(0); // I listen only to web request on port 80

  // get 10 first bytes only of the request, the rest does not matter here
  for(len = 0; len < 15; len++) getRequest[len] = Spi_Ethernet_getByte();
  getRequest[len] = 0;

  if(memcmp(getRequest, "GET ", 5)) return(0); // only GET method

  if(memcmp(getRequest+11, "ON", 2) // do we have ON command
  PORTB.F0 = 1; // set PORTB bit0
  else
  if(memcmp(getRequest+11, "OFF", 3)) // do we have OFF command
  PORTB.F0 = 0; // clear PORTB bit0

  if(PORTB.F0)
  {
    memcpy(indexPage+340, "#FFFF00", 6); // highlight (yellow) ON
    memcpy(indexPage+431, "#4974E2", 6); // clear OFF
  }
  else
  {
    memcpy(indexPage+340, "#4974E2", 6); // clear ON
    memcpy(indexPage+431, "#FFFF00", 6); // highlight (yellow) OFF
  }

  len = Spi_Ethernet_putConstString(httpHeader); // HTTP header
  len += Spi_Ethernet_putConstString(httpMimeTypeHTML); // with HTML MIME type
  len += Spi_Ethernet_putString(indexPage); // HTML page first part
  return len; // return to the library with the number of bytes to transmit

  unsigned int Spi_Ethernet_UserUDP( char *remoteHost, unsigned int remotePort,
  unsigned int destPort, unsigned int reqLength)
  {
    return 0; // back to the library with the length of the UDP reply
  }
}

void main()
{
  ADCON1 = 0x0F; // no analog inputs
  CMCON = 0x07; // turn off comparators

  PORTB.F0 = 0; // set PORTB.B0 as output (rele control pin)
  TRISB.F0 = 0;

  // 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_FULLLDUPLEX);

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

  while(1) {
    Spi_Ethernet_doPacket(); // do forever
    // process incoming Ethernet packets
  }
}

```



**HINWEIS:** Beispiel-Code für PIC®-Mikrocontroller in C, Basic und Pascal sowie andere Programme für dsPIC®- und AVR®-Mikrocontroller finden sich auf unserer Web-Seite: [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/)