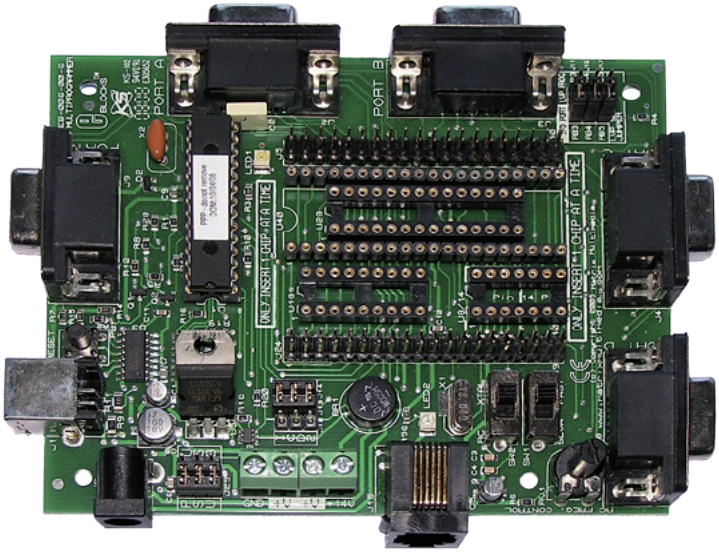
Flowcode w sieci

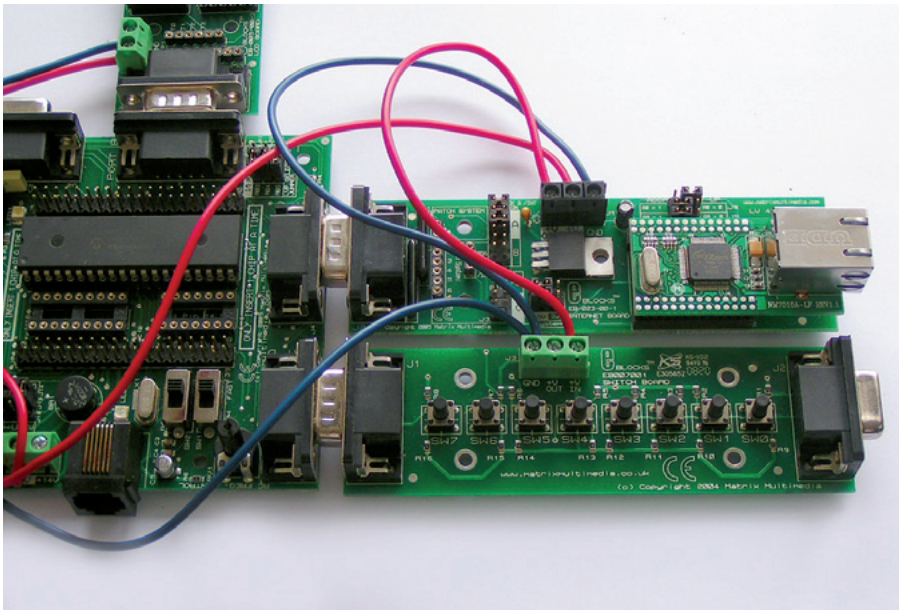
Aplikacje sieciowe dla mikrokontrolerów są obecnie jednym z gorących tematów. Chociaż rozwiązania stosowane w sieciach LAN nie należą do prostych, to producenci mikrokontrolerów i specjalizowanych układów peryferyjnych robią wszystko, żeby dołączenie mikrokontrolera do sieci było tak łatwe, jak to tylko możliwe.

Aby programistom korzystanie z Ethernetu i jednocześnie zachęcić ich do stosowania swoich wyrobów niektórzy producenci udostępniają bezpłatnie biblioteki gotowych stosów TCP/IP. Tak postępuje na przykład firma Microchip, w której rozwiązaniu zaimplementowano większość protokołów z rodziny TCP/IP. Jednak mimo bardzo dobrze przygotowanych przykładów i materiałów szkoleniowych, korzystanie z programowego stosu wymaga od programisty pewnej wiedzy i doświadczenia. Początkującemu raczej trudno będzie użyć tej biblioteki.

Innym rozwiązaniem są układy peryferyjne z wbudowanym, sprzętowym stosem TCP/IP. Troszczą się one o warstwę sieciową, transportu oraz mają wbudowane obwody interfejsu Ethernet. Dzięki temu programista może skupić się tylko na tworzeniu warstwy aplikacji. Jednym z takich układów jest W3100A firmy WIZnet. Oprócz zaimplementowanej obsługi protokołów internetowych TCP/IP, UDP, ICMP i ARP, układ wspiera protokoły DLC i MAC, a komunikuje się z mikrokontrolerem za pomocą interfejsu I²C lub przez magistralę równoległą.



Fotografia 1. Płyta główna e-block przeznaczona dla mikrokontrolerów PIC



Fotografia 2. Sposób dołączenia modułów e-block do płyty głównej

Układ W3100A zastosowano w jednym z modułów systemu e-block oferowanym przez firmę Matrix Multimedia, producenta pakietu Flowcode. System e-block dla mikrokontrolerów PIC składa się płyty głównej, na której umieszczono układ zasilania, mikrokontroler i programator umożliwiający zaprogramowanie mikrokontrolera przez port USB z menu Flowcode (**fotografia 1**). Linie portów są dołączone do złącz żeńskich DSUB9. Przy ich użyciu można do płyty głównej dołączyć układy peryferyjne systemu e-block: wyświetlacz LCD, przyciski, diody LED, przekaźniki, gniazdo kart SD, układy Bluetooth, RFID i wiele innych.

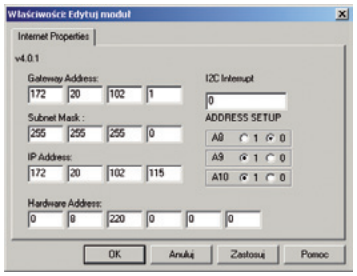
Moduł jest dołączany do płyty głównej za pomocą D-SUB9 (**fotografia 2**). Do dołączenia do sieci LAN służy standardowy kabel ethernetowy z wtykiem RJ-45.

Oprócz modułu z układem W3100A, na płytce jest umieszczony stabilizator LDO napięcia 3,3 V, złącze zasilania oraz zworki łączące linie sygnałów sterujących z liniami portów oraz służące do ustawiania adresu slave interfejsu I²C. Zastosowanie modułu ze stosem sprzętowym umożliwia dołączanie układów interfejsu Ethernet również innych producentów.

Mimo iż W3100A zawiera kompletny stos protokółów TCP/IP, to jego programowa obsługa nie jest łatwa, jednak Flowcode zdejmuje z programisty ciężar wnikliwego wczytywania się w dokumentację, poznawania wszystkich rejestrów sterujących i algorytmów ich programowania.

Układ modelowy

Mam do dyspozycji płytę główną e-block dla mikrokontrolerów PIC i kilka modułów peryferyjnych, w tym ethernetowy EB-023, więc postanowiłem sprawdzić czy można łatwo i szybko dołączyć mikrokontroler PIC do sieci. Do prób wybrałem mikrokontroler PIC16F877A i dodatkowo moduły e-block wyświetlacza LCD EB-005 i styków EB-007, które dołączyłem do złącz portów B i D mikrokontrolera. Moduł ethernetowy korzysta z interfejsu I²C i dlatego dołączyłem go do portu C.



Rysunek 3. Okno właściwości komponentu *TCP_IP*

Obsługa modułu ethernetowego jest powiązana z komponentem *TCP_IP* dostępnym w najnowszej wersji *Flowcode v4*. Wybiera się go z zakładki *Peripherial*. Komponent *TCP_IP* zawiera wiele makr sprzętowych i aby go efektywnie używać, trzeba mieć wiedzę na temat protokołów internetowych.

Wysyłanie instrukcji PING

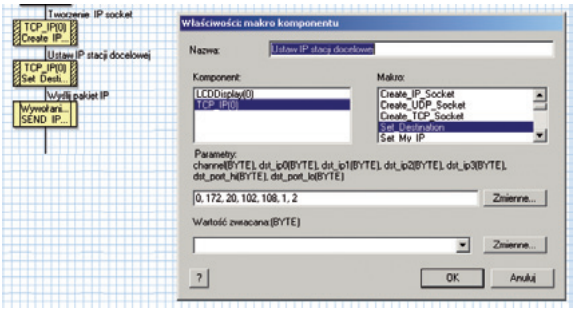
Polecenie *ping* jest używane do testowanie połączeń pomiędzy dwoma stacjami. Przesyła się je z użyciem protokołu ICMP. Ramka protokółu ICMP składa się 8-bitowych pól *type* i *code*, 16-bitowej sumy kontrolnej CRC i opcjonalnego pola danych.

Popularne „pingowanie" jest żądaniem odesłania echa. Stacja wysyłająca żądanie przesyła pakiet ICMP typu 0x08 (*Echo Request*). Stacja pytana odsyła potwierdzenie typu 0x00 (*Echo Reply*). Kod ma wartość 0x00, a pole danych zawiera 2-bajtowy identyfikator i 2-bajtowy numer sekwencyjny pakietu oraz dowolne, inne dane użytkownika. Stacja , która odebrała żądanie echa umieszcza w polu danych te same dane, które odebrała. Identyfikator i numer sekwencyjny mogą posłużyć do sprawdzenia czy odebrane potwierdzenie *Echo Reply* jest potwierdzeniem wysłanego żądania.

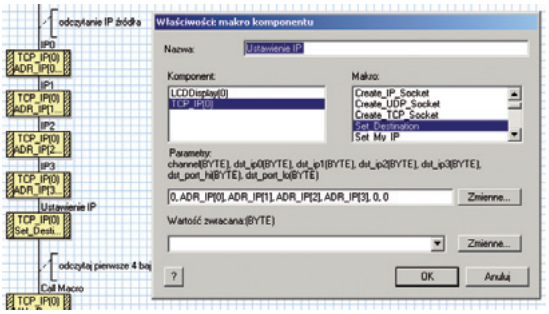
Ramka protokołu ICMP jest umieszczana w polu danych ramki IP. W ramce protokołu IP interesuje nas pole *Protocol*, czyli typu protokołu, którym przesyłane są dane. Dla protokołu ICMP pole *Protocol* ma wartość 0x01, a dla protokołu TCP ma wartość 0x06.

Rysowanie programu zaczynamy od postawienia na panelu elementu *TCP_IP* i kliknięciu na nim prawym klawiszem. Z menu rozwijanego wybieramy *Ext Properties*. W oknie *Properties* (**rysunek 3**) musimy wpisać wartości adresów bramy domyślnej i IP modułu oraz maskę podsieci. Dodatkowo, ustawia się adres MAC oraz *address setup*. Ta ostatnia wartość jest adresem slave interfejsu I²C układu W3100A. Ustawienia *address setup* muszą być takie same, jak ustawienia zworek J8, J9 i J10 w module EB-023.

„Pingowanie" można wykonać łącząc bezpośrednio moduł z komputerem PC. Można też podłączyć naszą płytkę do routera lub switcha sieciowego. Należy przy tym pamiętać, że proste switche nie mają swojego adresu IP, więc musi być do niego dołączone co najmniej jedno urządzenie mające adres IP i obsługujące polecenie PING, np. komputer PC z uruchomionym systemem Windows lub Linux. Dodatkowo, należy zwrócić uwagę na przeplot kabla Ethernet, który jest konieczny przy bezpośrednim połączeniu z komputerem PC.



Rysunek 5. Argumenty funkcji *Set_Destination*

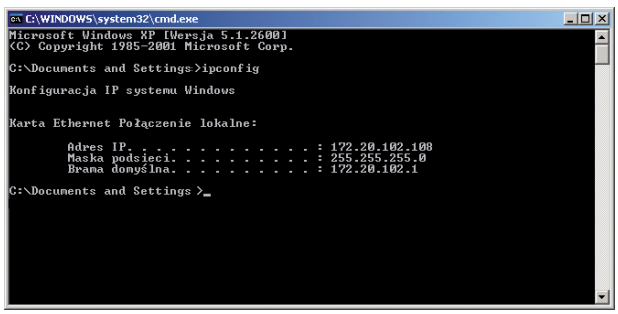


Rysunek 6. Fragment programu odczytującego adres IP z pakietu *Echo Request*

Ustawienia sieci (brama, maska i adres IP komputera) odczytujemy z ekranu komputera po wysłaniu komendy *ipconfig* (**rysunek 4**). W polu *IP address* w oknie konfigurowania komponentu trzeba wpisać dowolny, wolny adres w podsieci, inny niż adres wyświetlony *ipconfig*. Nie jest zaimplementowany protokół DHCP, więc trzeba to zrobić ręcznie. Po wpisaniu danych w oknie *Properties* klikamy Ok. Od tego momentu można już używać makr komponentu *TCP_IP* w programie.

Najpierw trzeba wywołać makro *Initialize*. Inicjalizuje ono układ W3100A i ustawia wcześniej wpisane w oknie *Properties* parametry.

Przesyłanie danych musi być poprzedzone zainicjowaniem gniazd transmisji. Można zdefiniować do 4 takich gniazd (*sockets*). My będziemy przesyłać ramki ICMP protokołem IP i dlatego musimy zdefiniować *socket IP*. Do tego celu jest używana funkcja *Create IP_Socket* z parametrami: gniazdo, protokół, rozgłaszanie (*broadcast*). Parametr *gniazdo* może mieć wartość od 0 do 3. W pole protokół wpisujemy 0x01, bo taką wartość ma mieć pole *Protocol* w nagłówku ramki protokołu IP dla danych ICMP. Rozgłaszanie włącza się wpisując parametr 0x01, a wyłącza się wpisując 0x00. Na koniec pozostaje zdefiniowanie adresu IP stacji, do której będziemy wysyłać żądanie echa. Najlepiej jest podać adres komputera PC, na którym wcześniej uruchamialiśmy polecenie *ipconfig*. Adres IP stacji docelowej ustawia funkcja *Set_Destination*. Jej argumentami są: adres IP i 2 bajty określające numer portu.



Rysunek 4. Okno programu *ipconfig*

Po wywołaniu wymienionych funkcji program jest gotowy do przesyłania danych z użyciem ICMP. Wysyłanie danych umieszczonych w polu danych protokołu IP musi być poprzedzone wywołaniem funkcji *Tx_start* z argumentem określającym numer kanału, w którym dane będą transmitowane. Następnie można wysyłać kolejne bajty ramki ICMP. Pamiętamy, że stacja inicjująca żądanie echa wysyła ramkę z typem *Echo Request* (0x08). Program wysyła bajty wywołując funkcję *Tx_sendbyte* z argumentami określającymi numer kanału i przesyłany bajt.

Wysłanie żądania echa będzie polegało na wysłaniu sekwencji:

- bajtu 0x08 – typ *Echo Request*,
- bajtu 0x00 – kod,
- 2 bajtów sumy CRC,
- 2 bajtów identyfikatora na przykład 0x10,
- 2 bajtów numeru sekwencyjnego,
- 32 bajtów danych użytkownika.

Jeżeli podczas konfigurowania nie popełniliśmy błędów, to pakiet powinien być odebrany przez stację o adresie ustalonym za pomocą funkcji *Set_Destination*. Stacja odeśle do naszego modułu potwierdzenie z typem *Echo Reply* (0x00). Moduł musi to potwierdzenie odebrać, aby stwierdzić czy realizacja polecenia przebiegła prawidłowo.

W procedurze odbioru musi być również odbieranie żądania echa wysłanego przez inne stacje. Jeżeli pierwszy odebrany bajt ma wartość 0x08, to oznacza, że moduł odbiera żądanie echa *Echo Request* wysłane przez inną stację. Jeżeli odebrany bajt jest ze-

rowy, to zaczynamy odbierać potwierdzenie *Echo Reply* na przesłane przez nas żądanie. Ponieważ wiemy jaki wysłaliśmy identyfikator i numer sekwencji, to procedura odbioru może te wartości testować. Można też sprawdzać prawidłowość sumy CRC pakietu. W czasie testu użyłem procedury, która odbiera cała ramkę, ale nie testuje ani identyfikatorów, ani sumy CRC, tylko wyświetla pierwszych 8 znaków ASCII z pola danych.

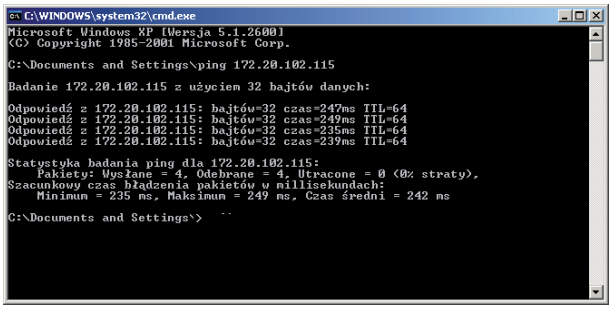
Trochę bardziej skomplikowany jest fragment procedury odbioru *Echo Request* i wysyłania potwierdzenia do nadawcy żądania. Moduł odbiera pakiet ICMP i umieszcza go w buforze odbiorczym. Żaby wysłać potwierdzenie *Echo Request*, trzeba prawidłowo wyliczyć 2 bajty sumy kontrolnej CRC, a do tego są potrzebne dane z pakietu ICMP. Liczbę bajtów danych można odczytać funkcją *Rx_readheader*. Ta funkcja odczytuje informacje dla każdego ze zdefiniowanych socketów. Dla trybu IP funkcją *Rx_readheader* z identyfikatorem idx=1 można odczytać liczbę bajtów danych (w naszym przypadku całego pakietu ICMP) i adres IP źródła pakietu IP.

Obliczaniem sumy kontrolnej zajmuje się makro programowe CRC. Suma kontrolna jest wyliczana na podstawie odebranych danych pakietu z tym, że pierwsze 4 bajty są wyzerowane, bo w polu typu jest bajt zerowy (*Echo Reply*), w polu kodu jest bajt zerowy i nie liczymy CRC z dwóch bajtów odebranego CRC.

Drugim ważnym elementem jest odczytanie adresu IP stacji, która wysłała *Echo Request* po to, by móc do niej odesłać *Echo Reply*. Ten adres jest zapisany w nagłówku pakietu IP jako *Destination Address*. Jak już wspomniałem, do jego odczytania używa się funkcji *Rx_readheader* z identyfikatorami idx od 2 do 5. Na **rysunku 6** pokazano fragment programu odczytywania adresu IP z pakietu *Echo Request* wysłanego do naszej płytki. Ten adres jest zapamiętywany w tablicy *ADR_IP*, a potem funkcją *Set_Destination* wpisywany w polu *Destination Address* pakietu IP, który zostanie wysłany z pakietem ICMP w polu danych.

Mamy już wyliczoną sumę kontrolną i ustawiony adres źródła. Teraz trzeba skompletować pakiet ICMP. W poleceniu *Ping* w odsyłanym *Echo Reply* pole danych powinno mieć taką samą zawartość, jak w odebranym *Echo Request*. Po doczytaniu cały pakiet ICMP jest zapamiętywany w buforze odbiorczym, który jest czytany sekwencyjnie przez funkcję *Rx_readbyte*. Wykorzystamy to do odesłania tych samych danych, które odebraliśmy. Żeby ustawić się na odebrane dane odczytujemy zawartość bufora i dodajemy pierwsze 4 bajty, aby licznik adresowy bufora odbiorczego ustawił się na pierwszy bajt odebranego pola danych. Potem wysyłamy 2 bajty zerowe, 2 bajty wcześniej wyliczonej sumy kontrolnej CRC i wszystkie bajty pola danych. Po wysłaniu danych trzeba wywołać funkcję *Rx_flush_data*. Po jej wykonaniu jest możliwe odebranie następnego pakietu.

Na **rysunku 7** pokazano okno z wynikiem realizacji komendy PING do naszego modułu mającego adres 172.20.102.115.



Rysunek 7. Wynik pingowania do aplikacji

Przykład użycia komponentu WebServer

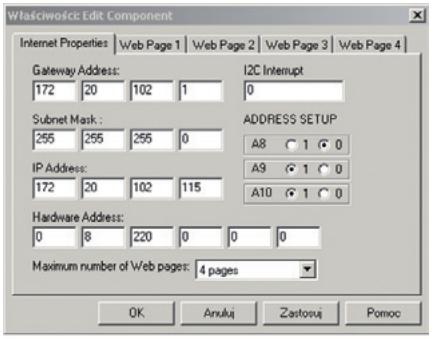
Mając do dyspozycji moduł EB-023 można użyć innego komponentu dla aplikacji sieciowych – *WebServer*. Umożliwia on zbudowanie własnego serwera stron WWW funkcjonującego w oparciu o mikrokontroler.

Komponent *WebServer* jest łatwy w użyciu i nawet niezbyt doświadczony programista jest w stanie go użyć. Pokażę jak skonfigurować *WebServer* i narysować własną aplikację wyświetlającą wartość zmiennej w oknie przeglądarki internetowej. Docelowo może to być na przykład temperatura lub napięcie, a moduł może posłużyć do wykonywania zdalnych pomiarów.

Z komponentem *WebServer* są związane trzy makra sprzętowe: *Initalize*, *CreateSeverSocket*, *CheckSocketActivity*.

Makro *Initialize* zeruje i inicjalizuje moduł EB-023 oraz ustawia parametry sieciowe: domyślny adres bramy, maskę podsieci adres IP i adres MAC (**rysunek 8**). Ustawiana jest też maksymalna liczba stron z zakresu 1...4 i adres interfejsu I²C układu W3100A.

Makro *CreateServerSocket* tworzy gniazdo (*socket*) do wysyłania i odbierania stron internetowych. Argumentami makra są: numer kanału i numer portu. Dla gniazda można zdefiniować do 4 kanałów, które mogą być



Rysunek 8. Okno właściwości komponentu *WebServer*

jednocześnie otwierane. Numer portu jest 16-bitowy (zakres 0...65535). Dla przesyłania stron WWW w protokole HTTP jest zarezerwowany port 80.

Strona umieszczona na serwerze wymaga stałego monitorowania żądania dostępu, na przykład dla stwierdzenia czy przeglądarka nie żąda odświeżenia strony. Do tego celu służy makro *CheckSocketActivity*. Argumentem wejściowym jest numer kanału. Zależnie od skonfigurowania połączenia sieciowego, przesłanie strony może wymagać jednego lub wielu żądań dostępu.

W zakładkach Web Page 1...4 trzeba umieścić kod strony napisany w języku

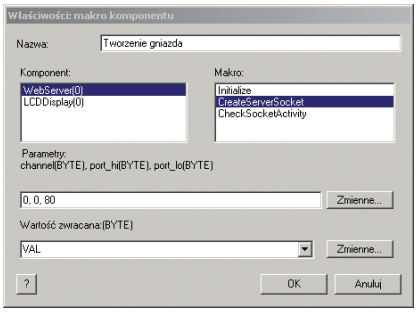
Listing 1. Kod źródłowy strony głównej serwera

```
<HTML>
<HEAD>
  <TITLE>Main page</TITLE>
</HEAD>
<BODY>

<P><CENTER><B>Test Webserver Flowcode PIC Micro</B></CENTER></P>

<P><CENTER><TABLE WIDTH="450" BORDER="1" CELLSPACING="2" CELLPADDING="2">
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/">Index</A></TD>
    <TD WIDTH="50%">    Home</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page2">Page 2</A></TD>
    <TD WIDTH="50%">   Wyświetl wartość zmiennej .</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page3">Page 3</A></TD>
    <TD WIDTH="50%">    Test strony 3.</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page4">Page 4</A></TD>
    <TD WIDTH="50%">    Test strony 4.</TD>
  </TR>
</TABLE></CENTER></P>

</BODY>
</HTML>
```



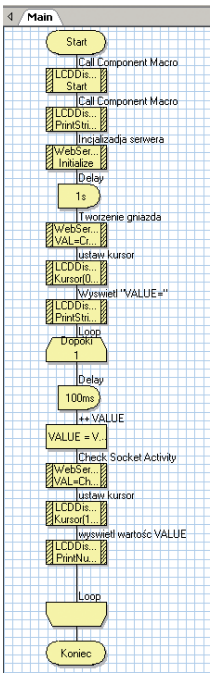
Rysunek 9. Okno komponentu Webserwer z argumentami makra *CreateServerSocket*

Listing 2. Kod źródłowy strony *„Page 2”*

```
<!DOCTYPE HTML PUBLIC „-//W3C//DTD HTML 4.0 Transitional//EN”>
<html>
<head>
<title>Internet E-Block Component</title>
</head>
<P><CENTER><TABLE WIDTH=”450” BORDER=”2” CELLSPACING=”4” CELLPADDING=”2”>
<body BGCOLOR=”#FF3333”>
<font face=”arial”>
 <TR BGCOLOR=”#3333FF”>
<TD WIDTH=”50%”>
  <H1>Wartosc zmiennej  = %VALUE%</H1>
</TD WIDTH=”50%”>
  </TR>
<TR BGCOLOR=”#6666FF”>
    <TD WIDTH=”50%”>
    <A HREF=”http://172.20.102.115/”>Index</A></TD>
    </TD WIDTH=”50%”>
  </TR>
</font>
</body>
</TABLE></CENTER></P>
</html>
```

HTML. Do prób z serwerem wykorzystałem kod źródłowy pokazany na **listingu 1**.

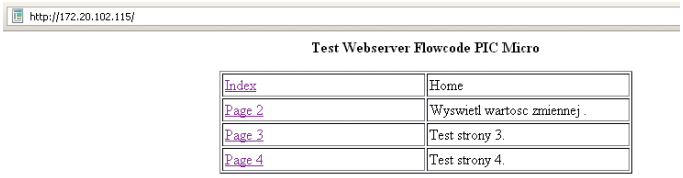
Oprócz modułu internetowego użyłem modułu wyświetlacza LCD do częściowej kontroli poprawności danych przesyłanych poprzez sieć. Na początku programu jest inicjalizowany wyświetlacz LCD i jest na nim wyświetlany napis „*Webserwer Test*”. Po tym jest wywoływane opisywane wcześniej makro *Intialize* komponentu *WebServer*. Ponieważ na zainicjowanie modułu potrzebny jest czas, to program odlicza opóźnienie 1 sekundy. W kolejnym kroku jest tworzone gniazdo (*socket*) do komunikacji z portem 80 przez wywołanie makra *CreateServerSocket* z argumentami numer kanału=0 i numer portu=80 (**rysunek 9**).



Rysunek 10. Program *Webserwera*

Teraz możemy sprawdzić czy klient (przeglądarka) nie żąda dostępu do strony na naszym serwerze. Aby nie było to sprawdzeniem niczego, to w podstronę *page2* został wbudowany mechanizm przesyłania do klienta wartości 8-bitowej zmiennej inkrementowanej co około 100 ms. Kodu źródłowy *page2* pokazano na **listingu 2**.

Żeby dana mogła być przesyłana z serwera do przeglądarki, to najpierw należy zdefiniować w programie nazwę zmiennej. W naszym przypadku jest to 8-bitowa zmienna *VALUE*. Wartość tej zmiennej jest przesyłana z użyciem protokołu HTTP do klienta (przeglądarki) po umieszczeniu jej nazwy w otoczeniu znaków %. W przykładzie będzie to linijka *<H1>Wartosc zmiennej = %VALUE%</H1>*. To wystarczy, aby przeglądarka otrzymała polecenie HTML „wyświetl wartość zmiennej VALUE”. Ten bardzo prosty z punktu widzenia użytkownika mechanizm pozwala na odczytywanie z serwera wartości dowolnych zmiennych zadeklarowanych w programie po każdym poleceniu odświeżenia strony w przeglądarce.



Rysunek 11. Okno główne programu serwera stron WWW

Rysunek 12. Fragment strony *Page2* umożliwiającej odczyt wartości zmiennej

Na **rysunku 10** przedstawiono kompletny program narysowany we *Flowcode v4* dla mikrokontrolera PIC18F458. Wartość zmiennej inkrementowanej w nieskończonej pętli jest wyświetlana na wyświetlaczu LCD, tak aby można było sprawdzić czy program wyświetlania na stronie działa prawidłowo.

Do testowania działania Webserwera użyto mikrokontrolera PC18F458. Po jego zaprogramowaniu trzeba połączyć moduł z siecią Ethernet i w przeglądarce wpisać adres IP 172.20.102.115 ustalony dla modułu w trakcie inicjalizacji. Jeżeli wszystko zostało zrobione prawidłowo, to powinno wyświetlić się okno strony głównej.

Z poziomu strony głównej można przejść do 3 stron nazwanych *Page2*, *Page3* i *Page4*. Wartość *VALUE* z modułu serwera jest odczytywana na stronie otwierającej się po kliknięciu na *Page2*. Fragment tej strony pokazano na **rysunku 11**. Zmienna jest odświeżana po kliknięciu na przycisk odświeżania strony w przeglądarce. Do strony głównej można wrócić klikając na Index.

Uwagi końcowe

Dzięki pracy wykonanej przez twórców pakietu Flowcode i układu WM3100A aplikacje sieciowe są tak łatwe, że może je wykonać nawet mało zaawansowany programista. Dowodzą tego dwa zademonstrowane tutaj przykłady. Pierwszy z nich (*ping*) wymaga trochę szerszej wiedzy na temat protokółów sieciowych, ale nie jest to wiedza trudna do opanowania. Z kolei projekt *WebSerwera* jest tak pomyślany, że jest potrzebna tylko znajomość podstaw języka HTML i pakietu Flowcode.

Tomasz Jabłoński, EP
tomasz.jablonski@ep.com.pl