Przykłady zastosowań TCP/IP w mikrokontrolerach, część 3

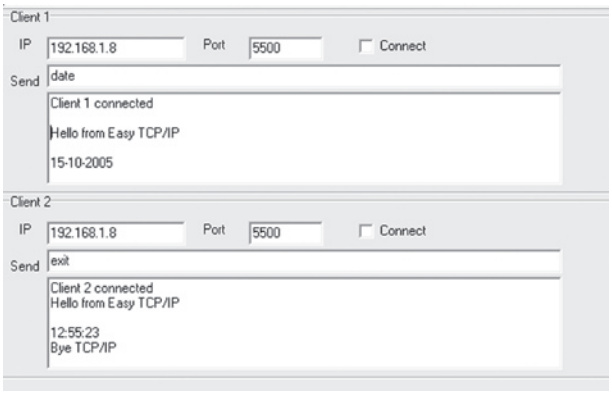
Listingi do artykułu są umieszczone na płycie CD-EP6/2006B oraz na stronie http://download.ep.com.pl

Jak mówi stare przysłowie: „co dwie głowy to nie jedna". A gdy głów tych będą tysiące, a nawet miliony? Aż strach pomyśleć. Tymczasem taką globalną siłę intelektu mamy dziś w zasięgu ręki, ba – nawet z niej korzystamy. Wszystko za pośrednictwem terabajtów informacji, które w każdej sekundzie przesyłane są olbrzymią siecią informatyczną oplatającą całą kulę ziemską.

W poprzednim odcinku kursu dowiedzieliśmy się, jak można wysłać i odebrać wiadomość e–mail. To jednak nie wszystko. Tematyka poruszona w tym odcinku będzie nie mniej frapująca, bo oto za chwilę będziemy potrafili „wystawić" własny serwer internetowy.

Aplikacja pracująca jako prosty serwer zapytań mogących obsłużyć do 4 klientów równocześnie

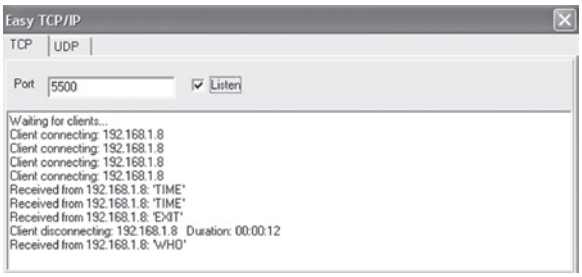
Serwerem można uznać aplikację, która odpowiada na zapytania klientów. Opisywany w artykule moduł TCP/IP może pracować jako serwer potrafiący obsłużyć jednocześnie do 4 klientów. Klienci mogą wysyłać zapytania do serwera o czas, datę oraz żądać rozłączenia. Do symulacji klientów zostanie wykorzystany program easytcpip.exe, który umożliwia dołączenie do 2 klientów. Na **list. 5** przedstawiono program realizujący prosty serwer zapytań. Działa on w nieskończonej pętli, przy czym w wewnętrznej pętli instrukcją *socketstat* sprawdzany jest status kolejno każdego z 4 gniazd. Jeżeli status informuje o zamknięciu danego gniazda (*Sock_closed*), wykonywana jest instrukcja *getsocket*, która otwiera gniazdo o zadanym numerze na porcie 5500. Kolejna instrukcja *socketlisten*, której parametrem jest numer gniazda, służy do otwarcia danego gniazda w trybie serwera. Wykonanie tej instrukcji powoduje, że wybrane gniazdo będzie nasłuchiwać portu wybranego wcześniej instrukcją *getsocket*. Jednocześnie można otworzyć do 4 gniazd. W pierwszej fazie działania programu zostają otworzone 4 gniazda w trybie serwera. Warto zauważyć że gniazda są otwierane po sprawdzeniu czy nie ma połączenia z danym gniazdem. Po otwarciu danego gniazda zerowana jest odpowiadająca mu flaga *Flags.idx*. Po przyłączeniu się klienta do serwera sprawdzany jest stan tej flagi dla gniazda, do którego dołączył się klient. Jeśli jest wyzerowana, serwer wysyła do klienta komunikat powitalny, po czym flaga jest ustawiana, aby komunikat powitalny nie był już wysyłany. Serwer reaguje na trzy polecenia otrzymane od klienta. Są to: *time*, *date* i *exit*. Jeśli dane zostały wysłane od klienta z wykorzystaniem instrukcji *getdstip*, do zmiennej LONG zwracany jest adres IP klienta. Umożliwia to rozpoznanie klienta, od którego otrzymano dane. Informacje o kliencie zostają wysłane do terminala przez RS232, przy czym adres IP klienta zostaje sformatowany do postaci tekstowej z wykorzystaniem funkcji *Ip2str*. W dalszej części programu rozpoznawane jest polecenie otrzymane od klienta. Jeśli jest to *exit*, wysyłany jest do klienta komunikat rozłączenia, po czym wykonywana jest instrukcja *closesocket* zamykająca gniazdo, do którego był dołączony klient. Jeśli otrzymano polecenie *time*, wysyłany jest do danego klienta przykładowy czas, a jeśli było to polecenie *data*, wysyłana jest przykładowa data. Skonfigurowanie modułu TCP/IP do pracy jako serwer jest dosyć proste. Na **rys. 10** pokazano widok działania programu *easytcpip* tworzącego 2 klientów. W programie należy podać adres IP serwera oraz port, na którym będą łączyć się klienci. Przykładowo po połączeniu wi-



Rys. 10. Okno programu *easytcpip* tworzącego 2 klientów

```
closed, Socket 0 0, Result 0
closed, Socket 1 1, Result 0
closed, Socket 2 2, Result 0
closed, Socket 3 3, Result 0
Connect Client 0
Connect Client 1
Client: 0(192.168.1.2): date
Client: 1(192.168.1.2): time
Client: 1(192.168.1.2): exit
closed, Socket 1 1, Result 0
close_wait
closed, Socket 0 0, Result 0
```

Rys. 11. Postać informacji wysłanych przez mikrokontroler do terminala podczas pracy układu w roli serwera



Rys. 12. Okno programu *easytcpip* pracującego w roli serwera na porcie 5500

doczny jest komunikat powitalny od serwera. Klient 1 wysłał zapytanie o datę, a drugi klient o czas i żądał rozłączenia. Na **rys. 11** przedstawiono informacje wysłane przez mikrokontroler do terminala podczas pracy układu w roli serwera. Widać w nich adres IP dołączonych klientów oraz wysyłane przez nich do serwera zapytania. Tego typu aplikacja może służyć przykładowo do zbierania danych, o które może pytać wielu klientów (jednocześnie 4).

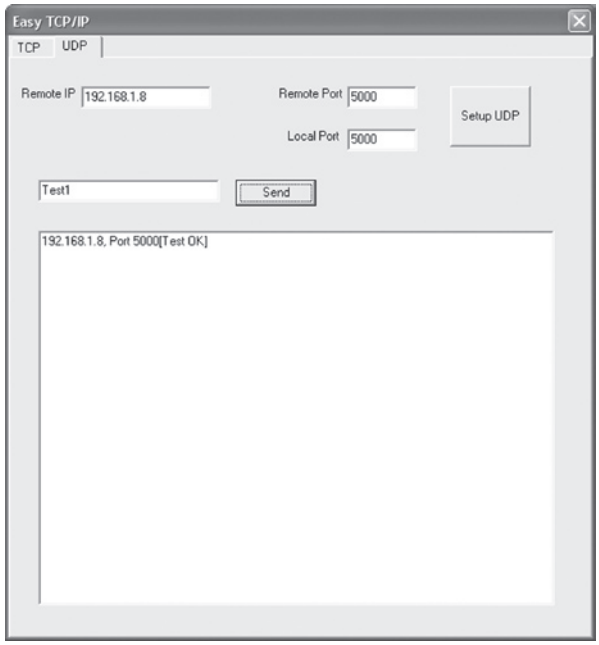
Aplikacja pracująca jako klient wysyłający zapytania do serwera

Moduł TCP/IP może pracować także jako klient, czyli pełnić funkcję aplikacji łączącej się do serwera. Klientem można uznać aplikację, która wysyła zapytania do serwera. W ramach tego przykładu zostanie przedstawiony program, który tworzy 4 klientów wysyłających do serwera zapytania takie jak: *time*, *exit* lub *who*. Wysyłanie zapytań jest możliwe przez RS232 z poziomu terminala. Serwer, do którego łączyć się będą klienci został utworzony przy pomocy programu *easytcpip*. Na **list. 6** przedstawiono program realizujący aplikację pracującą jako klient. W pierwszej kolejności następuje otwarcie 4 gniazd kolejno na portach 1001, 1002, 1003 i 1004. Jest to zrealizowane w pętli *For* wykonywanej dla każdego z gniazd.

```
Local port : 1001 Socket 0  Result 0
Local port : 1002 Socket 1  Result 0
Local port : 1003 Socket 2  Result 0
Local port : 1004 Socket 3  Result 0
Client: 0, Server:Welcome to Easy TCP/IP
Client: 1, Server:Welcome to Easy TCP/IP
Client: 2, Server:Welcome to Easy TCP/IP
Client: 3, Server:Welcome to Easy TCP/IP
Client: 0, Server:2005-11-24 21:53:50
Client: 1, Server:2005-11-24 21:53:50
close_wait
Client: 3, Server:There are 3 connected users:
Client: 3, Server:192.168.1.8:1004 2005-11-24 21:53:38
Client: 3, Server:192.168.1.8:1002 2005-11-24 21:53:38
Client: 3, Server:192.168.1.8:1001 2005-11-24 21:53:38
```

Rys. 13. Postać informacji wysłanych przez mikrokontroler do terminala podczas pracy układu w roli klient

Następuje także łączenie otwartego gniazda (klienta) do serwera o numerze IP 192.168.1.2 i porcie 5500. Rezultat otwarcia gniazda, jak i połączenia są wysyłane do terminala przez interfejs RS232. Po wykonaniu tej pętli z serwerem połączonych będzie 4 klientów. Jeśli z terminala zostanie odebrany znak o kodzie ASCII 013 (Enter), klient 0 i 1 wyślą polecenie *time*, na które serwer zwróci czas. Klient 2 wyśle polecenie *exit*, na które serwer odłączy go od siebie. Klient 3 wyśle polecenie *who*, na które dostanie odpowiedź o tym, kto jest dołączony do serwera. W tym przypadku dołączonych będzie 3 klientów, bo klient 2 wysłał polecenie rozłączenia. Do wysyłania poleceń do serwera wykorzystano instrukcję *tcpwritestr*, która różni się od instrukcji *tcpwrite* tym, że przesyła dane tekstowe. Przy ostatnim parametrze 255 instrukcja sama doda do wysyłanych danych znaki potwierdzenia CR+LF. Przy ostatnim parametrze 0 nie będą dodawane znaki CR+LF. Po wysłaniu zapytań do serwera, w kolejnej pętli For wykonywanej dla każdego z klientów, sprawdzany jest status połączenia i jeśli serwer wysłał do danego klienta dane, są one odbierane i wysyłane do terminala przez RS232 wraz z informacją o numerze klienta. Połączenie z klientami kończy się w podobny sposób jak w poprzednich aplikacjach, z wykorzystaniem instrukcji *closesocket*. Na **rys. 12** pokazano widok działania programu *easytcpip* pracującego w roli serwera na porcie 5500. Widać że przyłączyło się do niego 4 klientów, którzy wysłali wcześniej opisane zapytania, przy czym po wysłaniu zapytania *exit* przez klienta 2 został on rozłączony. Na **rys. 13** przedstawiono informacje wysłane przez mikrokontroler do terminala podczas pracy układu w roli klienta. Po podłączeniu klientów do serwera, serwer także wysłał do nich komunikaty powitalne. Widać w nich także jakie informacje dany klient otrzymał od serwera po wysłaniu zapytania. W przypadku zapytania *who* serwer zwrócił do klienta 3 informację o połączonych z nim (serwerem) klientach. Tego typu aplikacja kliencka może służyć do odbioru informacji od serwera, uzyskiwanych na podstawie zapytań. Serwer przykładowo może zbierać dane o temperaturze w kilku miejscach, którą mogą następnie odczytywać klienci.



Rys. 14. Okno programu *easytcpip* pracującego z protokołem UDP

Aplikacja komunikująca się przez protokół bezpołączeniowy UDP (prosty CHAT)

Protokół UDP umożliwia dostarczanie danych bez gwarancji odbioru. Obciążenie przesyłanych danych informacjami dodatkowymi jest niewielkie. Nie ma w tym protokole żadnego mechanizmu sprawdzającego czy aplikacja docelowa otrzymała przesyłkę. UDP zapewnia wyłącznie prostą sumę kontrolną. Pomimo braku mechanizmu sprawdzającego protokół UDP ma pewne zalety, których nie ma protokół TCP. W sieciach, w których nie ma problemów z dostarczaniem danych, wykorzystanie protokołu UDP wywołuje mniejszy ruch. Komunika-

```
Init , set IP to 192.168.0.8
Socket 0 0
192.168.1.2, Port 5000 [Test1]
Test OK
```

Rys. 15. Postać informacji wysłanych przez mikrokontroler do terminala podczas pracy układu z protokołem UDP

cja przy użyciu UDP nie wymaga nawiązywania sesji. Aplikacja źródłowa jest przygotowana do komunikacji z odpowiednim portem aplikacji docelowej. Jeżeli potrzebuje odpowiedzi, dołącza swój adres i port do nagłówka UDP. Nagłówek UDP składa się z 8 bajtów, w których znajduje się port docelowy, źródłowy, liczba danych i suma kontrolna. Popularną aplikacją wykorzystującą UDP jest serwer DNS. W opisywanym niżej przykładzie protokół UDP posłuży do realizacji prostego CHAT–a, w którym jedną stroną będzie zestaw Easy TCP/IP, a drugą stroną komputer z uruchomionym programem *easytcpip*. Komunikacja z Easy TCP/IP będzie się odbywać za pomocą interfejsu RS232 i komputerowego terminala. Na **list. 7** przedstawiono program realizujący komunikację z wykorzystaniem bezpołączeniowego protokołu UDP. W pierwszej kolejności w programie otwierane jest gniazdo instrukcją *getsocket* wykorzystujące port 5000. Gniazdo otwierane jest w trybie *Sock_dgrm* wymaganym przez protokół UDP. Po otwarciu gniazda program przechodzi do wykonywania nieskończonej pętli, w której znajdują się instrukcje wysyłające i odbierające dane za pomocą protokołu UDP. Wysłanie danych z poziomu zestawu Easy TCP/IP jest możliwe za pomocą interfejsu RS232 i komputerowego terminala, a z poziomu komputera za pomocą programu *easytcpip*. Po otrzymaniu danych z komputerowego terminala potwierdzonych enterem, są one wysyłane przez UDP za pomocą instrukcji *udpwritestr*. Pierwszym parametrem jest adres IP odbiorcy (w tym przypadku 192.168.1.2). Kolejnymi parametrami są: port, numer gniazda, nazwa zmiennej, z której dane są wysyłane oraz liczba wysyłanych bajtów. Przy liczbie bajtów równej 0, wysyłane są wszystkie dane zmiennej. Za pomocą instrukcji *socketstat*, sprawdzane jest czy są jakieś dane do odebrania. W przypadku wykorzystania tej instrukcji w protokole UDP zwracana wartość zawsze będzie większa o 8 bajtów (o wielkość nagłówka). Jeśli takowe dane są, zostają odebrane za pomocą instrukcji *udpread*, której pierwszy parametr to numer gniazda, drugim jest zmienna, do której mają być zapisane odebrane dane. Ostatni parametr określa liczbę odbieranych danych. Odbierane są wszystkie dane łącznie z nagłówkiem, który składa się z 8 bajtów. Jeśli odbierane dane są zapisywane do zmiennej tekstowej, instrukcja będzie odbierać dane aż do napotkania znaków potwierdzenia (CR+LF). Instrukcja *udpread* także rozkodowuje nagłówek i poszczególne jego składowe umieszcza w zmiennych: *peersize* (liczba danych), *peeraddress* (adres IP nadawcy) i *Peerport* (port). W programie oprócz danych otrzymanych, do terminala wysyłane są także dane o adresie IP nadawcy pobranym ze zmiennej *peeraddress*, który jest dodatkowo formatowany za pomocą instrukcji *Ip2str* oraz wykorzystywany port. Podczas wysyłania samych danych do terminala pomijany jest nagłówek. Wysyłane są tylko dane za nagłówkiem. Program działa w pętli, na bieżąco sprawdzając czy są dane do odebrania z terminala lub nadawcy z wykorzystaniem UDP. Na **rys. 14** pokazano widok działania programu *easytcpip* pracującego z protokołem UDP. W programie należy wpisać adres odbiorcy (zestawu Easy TCP/IP) i przycisnąć przycisk *Setup UDP*. Na **rys. 15** przedstawiono informacje wysłane przez mikrokontroler do terminala podczas pracy układu z protokołem UDP. Na obu rysunkach widać, że z poziomu programu *easytcpip* został wysłany tekst *test1*, a z poziomu programu terminala tekst *Test OK*. Wysłane w obu kierunkach teksty zostały poprawnie odebrane. Dodatkowo wyświetlana jest informacja o adresie IP nadawcy i wykorzystywanym porcie. Prosty w realizacji bezpołączeniowy protokół UDP można wykorzystać w wielu prostych aplikacjach. Przykładowo może to być konwerter LAN<–> RS232.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl