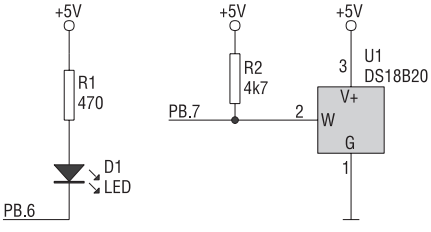
Przykłady zastosowań TCP/IP w mikrokontrolerach, część 4

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.

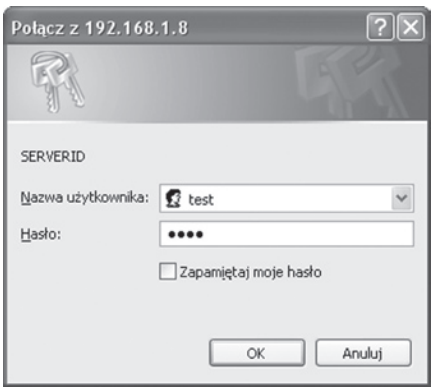
Aplikacja serwera HTTP umożliwiająca odczyt temperatury oraz sterowanie diodą LED przez przeglądarkę

Protokół HTTP (*HyperText Transfer Protocol*) jest wykorzystywany do przeglądania stron www. Jest on oparty na mechanizmie pytanie–odpowiedź. Klient żąda przesłania strony z serwera www, odpowiedzią jest przesłanie strony do klienta. Protokół HTTP wykorzystuje port o numerze 80. Po udzieleniu odpowiedzi na zapytanie klienta połączenie jest zrywane do czasu otrzymania następnego zapytania. W zapytaniach wykorzystuje się kilka metod jak: *GET*, *HEAD*, *POST*, *PUT* itp. W przykładowej aplikacji będą wykorzystywane głównie metody *GET*, *HEAD* oraz *POST*. Metoda *GET* jest wykorzystana w zapytaniu do żądania przesłania strony www o określonym adresie. Podobna do *GET* jest metoda *HEAD*, która nie zwraca do klienta treści przesyłki. Inaczej jest z metodą *POST* (wykorzystywaną podczas przesyłania danych z formularzy) pozwalająca żądać od serwera przyjęcia załączonych danych jako nowej, skierowanej do niego publikacji. Protokół HTTP jest prosty w działaniu. Serwer po otrzymaniu np. metody *GET* z adresem strony, wysyła do klienta (przeglądarki) ciąg znaków strony www napisanej np. w HTML. W ramach tego przykładu zostanie przedstawiona aplikacja serwera HTTP, który na zapytanie klienta (wysłanie adresu /index.htm) wysyła stronę powitalną, na której umieszcza zmierzoną przez niego czujnikiem DS18B20 temperaturę. Dodatkowo



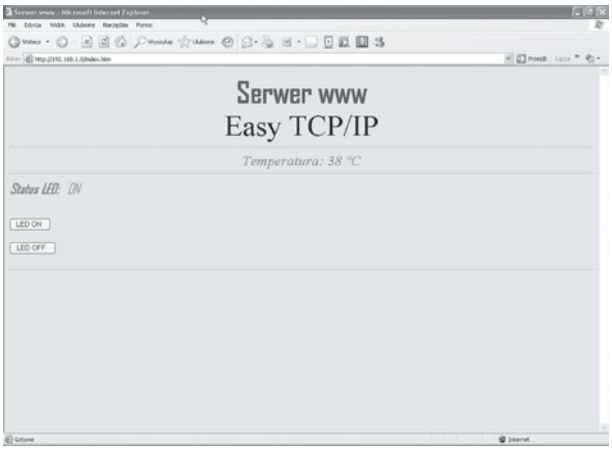
Rys. 16. Sposób dołączenia czujnika temperatury DS18B20 oraz diody LED do zestawu TCP/IP

z poziomu przeglądarki umożliwia wysłanie (z wykorzystaniem formularza i metody *POST*) danych o stanie diody LED (możliwe włączenie lub wyłączenie diody LED). W serwerze HTTP jest także możliwość włączenia autoryzacji poprzez ustawienie odpowiedniej stałej. Aby otrzymać żądaną stronę należy w takim przypadku podać nazwę użytkownika i hasło. Do zestawu TCP/IP został dołączony czujnik temperatury DS18B20 oraz dioda LED w sposób pokazany na **rys. 16**. Na wyświetlaczu LCD zestawu będzie wyświetlany komunikat powitalny oraz zmierzona temperatura. Dodatkowe informacje o pracy serwera będą także wysyłane przez interfejs RS232 do komputerowego terminala. Na **list. 8** przestawiono program pracujący jako serwer HTTP. W pierwszej kolejności konfigurowany jest wyświetlacz LCD oraz moduł TCP/IP. Linia, do której została dołączona dioda LED oraz linia będąca magistralą 1Wire jest konfigurowana jako wyjście. W programie występuje jedna procedura *Pom_temp* służąca do odczytu temperatury z termometru DS18B20. W tej procedurze w pierwszej kolejności wysyłany jest rozkaz pomiaru. Po opóźnieniu ok. 1 sekundy następuje odczyt temperatury oraz odpowiednia jej konwersja



Rys. 17. Okno autoryzacji

do dokładności równej 1°C. Wartość zmierzonej temperatury zostaje zapisana w postaci tekstowej do zmiennej *Temperature*. W programie występuje stała *Authenticate*. Jeśli ma wartość równą 1, przy dostępnie do strony www będzie dodatkowo przeprowadzana autoryzacja. Aby otrzymać stronę www, należy podać login oraz hasło. Instrukcje obsługujące protokół HTTP działają w nieskończonej pętli *do–loop*. Instrukcja wyboru *case* jest identyczna jak w poprzednich programach. Ponieważ aplikacja pracuje jako serwer, gniazdo zostaje otwarte dopiero po sprawdzeniu statusu. Jeśli gniazdo jest zamknięte (*Sock_closed*), następuje otwarcie gniazda z portem 80 charakterystycznym dla



Rys. 18. Wygląd pobranej z serwera strony WWW

protokołu HTTP. Kolejna instrukcja konfiguruje otwarte gniazdo do pracy w trybie serwera (nasłuchu). W przypadku otrzymania statusu *Sock_close_wait*, otwarte gniazdo zostaje zamknięte, co ma miejsce po wysłaniu strony www do klienta. Po otrzymaniu statusu *Sock_established* świadczącego o żądaniu klienta, sprawdzany jest jego adres IP (pobrany za pomocą instrukcji *getdstip*) z adresem znajdującym się w zmiennej *Ippon*. Jeśli adresy są różne, zmienna *Bauth* przyjmuje wartość 0. Porównanie adresów jest wykonywane, aby autoryzacja była jednorazowa dla danego klienta, a nie wymagana po każdorazowym odświeżeniu strony www w przeglądarce. W dalszej części programu sprawdzane jest czy są jakieś dane do odebrania. Jeśli są, sprawdzana jest metoda zapytania. Jeśli jest to *GET* lub *HEAD*, zmienna *BCMD* przyjmuje wartość 1 oraz następuje skok do podprogramu *page*. W tym podprogramie z otrzymanych danych z przeglądarki zostaje odczytana nazwa żądanej strony www. Nazwa żądanej strony zostaje zapisana do zmiennej *Shtml*. Jeśli autoryzacja zostaje włączona i sprawdzane adresy IP są różne (zmienna *Bauth* wynosi 0), do przeglądarki klienta jest wysyłane polecenie o kodzie 401, będące żądaniem autoryzacji. W kolejnej linii wysyłana jest nazwa serwera, który żąda autoryzacji. Będzie ona wyświetlana w okienku autoryzacji, które zostało pokazane na **rys. 17**. Po wysłaniu informacji o autoryzacji, następuje ponowne oczekiwanie na dane od klienta. Jeśli w otrzymanych danych znajduje się tekst *Authorization: Basic*, to znaczy że otrzymano z przeglądarki nazwę użytkownika i hasło kodowane w formacie *Base64*. W opisywanym przykładzie loginem jest *test* i hasłem jest również *test*. W dalszej kolejności z otrzymanych danych odczytywany jest login i hasła i dekodowane funkcją *base64dec*. Jeśli login i hasło są identyczne z tymi, które zapisano w programie, to zmiennej *Bauth* zostaje przypisana wartość 1 (wyłączenie ponownej autoryzacji). Także do zmiennej *Ipcon* zostaje zapisany adres IP klienta. Jest on zapamiętywany, aby po odświeżeniu strony nie była przeprowadzana ponowna autoryzacja, aż do zamknięcia przeglądarki www. Po przeprowadzonej poprawnie autoryzacji, do przeglądarki zostaje wysłane polecenie o kodzie 200 informujące, że będzie wysyłana strona www. W przypadku wyłączonej autoryzacji także będzie wysyłane identyczne polecenie. Po wysłaniu tego typu polecenia, następuje skok do podprogramu *struc*, w którym jest wysyłana strona www. Do klienta najpierw należy wysłać informację o tym, ile danych będzie wysyłanych, po czy można wysłać samą stronę www. Przed wysłaniem strony www, do przeglądarki jest wysyłany komunikat *Content–Type: text/html*, informujący o formie zapisu strony www (zapis w języku HTML). W pierwszej kolejności sprawdzany jest adres żądanej strony www. Jeśli jest identyczny z */index.htm*, to wysyłana jest strona www, przy czym w pierwszej kolejności obliczana jest liczba danych, jaka będzie wysyłana do przeglądarki. Liczba danych jest obliczana z wykorzystaniem instrukcji *Len*. Obliczenia są prowadzone aż do napotkania tekstu końca strony *</body></html>*. Liczba wysyłanych danych jest zapisywana w zmiennej *wsize*. Jest ona wysyłana za komunikatem *Content–Length:*. Na liczbę wysyłanych danych mają wpływ nie tylko dane zapisane w HTML samej strony, ale uwzględniana jest też wartość zmierzonej temperatury oraz stan diody LED. Strona www została zapisana w pamięci programu za pomocą instrukcji *data*. Jest ona identyfikowana etykietą www. Do odczytu tych danych wykorzystywane są instrukcje *restore* oraz *read*. Po wysłaniu do przeglądarki liczby przesyłanych danych następuje wysłanie strony www odczytywanej kolejno z pamięci programu. W kod strony podczas jej wysyłania wplatana jest wartość temperatury oraz stan diody LED. W przypadku otrzymania strony www o nazwie innej niż */index.htm*, wysyłana jest strona www, która powoduje przekierowanie strony na adres */index.htm*. Tak jak w przypadku głównej strony, najpierw obliczana jest i wysyłana liczba danych, a następnie sama strona www informująca o przekierowaniu. W kodzie HTML instrukcja:

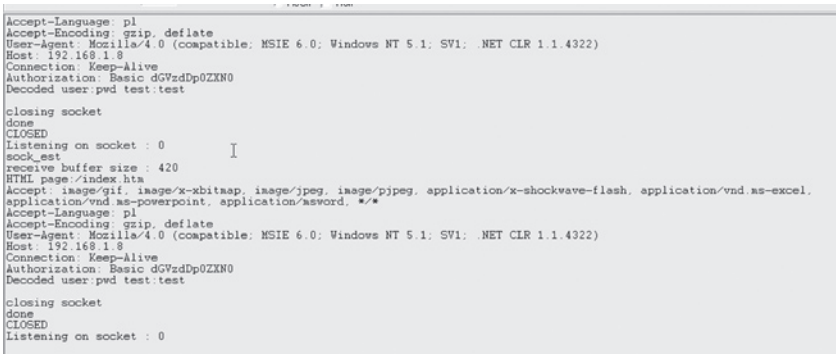
meta HTTP–EQUIV="refresh" CONTENT="5"

powoduje, że strona www będzie automatycznie odświeżana co 5 sekund. Znaki o kodzie ASCII 034 to znaki cudzysłowia ("). Wspominaliśmy wcześniej o możliwości sterowania przez serwer diodą LED z poziomu przeglądarki. Wykorzystano do tego celu przyciski wysłania formularza przy pomocy metody *POST*. Instrukcje zapisane w HTML realizujące wysłanie formularza z komunikatem zapalenia diody LED są następujące:

<form method="post">
<input type="hidden" name="ledon">
*<input type="submit value="LED ON">

</form>*

Dla przycisku wyłączającego diodę LED, instrukcje będą podobne, z tym że wyraz *ledon* zostanie zastąpiony wyrazem *ledoff*. Naciśnięcie przycisku spowoduje wysłanie do serwera zapytania z wykorzystaniem metody *POST*. Po wykryciu przez serwer tej metody wykonywany jest podprogram *page*, a zmiennej *Bcmd* zostaje przypisana wartość 2. W przypadku metody *POST* wykrywany jest komunikat *Content–Length:* z którym związana jest liczba wysłanych przez przeglądarkę danych. Liczba wysłanych danych z przeglądarki jest zapisywana do zmiennej *Bcontent*. Jeśli zmienna *Bcmd* jest równa 2, przy metodzie *POST* po instrukcjach autoryzacji następuje odczyt danych wysłanych po naciśnięciu przycisku wysyłającego formularz. Wysłane dane są odczytywane do zmiennej *Buf* leżącej w tej samej przestrzeni adresowej pamięci mikrokontrolera co zmienna *s*. Liczbę odbieranych



Rys. 19. Postać informacji wysłanych przez mikrokontroler do terminala podczas działania serwera HTTP

danych wskazuje zmienna *Bcontent*. Jeśli otrzymano tekst *ledon*, dioda LED zostaje zapalona, a jeśli *ledoff* dioda zostaje zgaszona. Temperatura jest mierzona zawsze po wysłaniu strony www. W tym czasie jest także wyświetlana na wyświetlaczu LCD. Aby pobrać stronę www z serwera, w przeglądarce należy wpisać następujący adres: *http://192.168.1.8/index.html*

Po zalogowaniu (rys. 17) zostaje wyświetlona strona www, która automatycznie jest odświeżana co 5 sekund. Na **rys. 18** przedstawiono wygląd pobranej z serwera strony www. Wyświetlana jest na niej zmierzona temperatura oraz stan diody LED. Przycisk *LED ON* włącza diodę LED, a *LED OFF* wyłącza ją. Na **rys. 19** przedstawiono informacje wysłane przez mikrokontroler do terminala podczas działania serwera HTTP. Na podstawie opisanego wyżej serwera HTTP pokazano jak wykorzystując język HTML zaprezentować na stronie www informacje uzyskane z czujnika temperatury oraz jak można sterować układem wykonawczym, którym wcale nie musi być dioda LED – równie dobrze może to być np. przekaźnik, czy triak załączający urządzenie o większym obciążeniu.

Podsumowanie

Wykorzystywanie Ethernetu oraz Internetu w wielu nietypowych aplikacjach będzie się coraz bardziej rozpowszechniało. Już dziś można w prosty sposób dołączyć do sieci mikrokontroler 8–bitowy, co zostało przedstawione w przykładach. Miejmy nadzieję, że opisane w artykule programy wykorzystujące protokoły TCP i UDP wiele wyjaśniły i że będą przydatne w wielu własnych aplikacjach (po odpowiednich przeróbkach). Dokładniejszych informacji o protokołach TCP i UDP należy szukać w dokumentach RFC. W artykule nie wyczerpano wszystkich zagadnień. Nie przedstawiono także wszystkich możliwych instrukcji związanych z modułem TCP, a dokładnie biblioteką *tcpip.lbx* jak np.: *getdstport*, która umożliwia odczyt portu podłączonego klienta, *udpwrite* wysyłająca protokołem UDP dane czy instrukcji *maketcp*, która formatuje adres IP do zmiennej LONG. Podczas testów programów bardzo pomocny okazuje się komputerowy terminal oraz program *easytcp.exe*.

Jeśli Czytelnicy będą zainteresowani tą tematyką, w przyszłości zostaną przedstawione kolejne aplikacje, jak serwer DHCP czy FTP, a także aplikacje umożliwiające połączenie się z Ethernetem czy Internetem za pomocą najprostszego mikrokontrolera wykorzystującego do tego interfejs I^2C (potrzebnych jest tylko kilka linii mikrokontrolera). Tego typu komunikację umożliwia wykorzystywany w artykule moduł IIM7000A lub IIM7010. Prosimy o listy w tej sprawie.

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