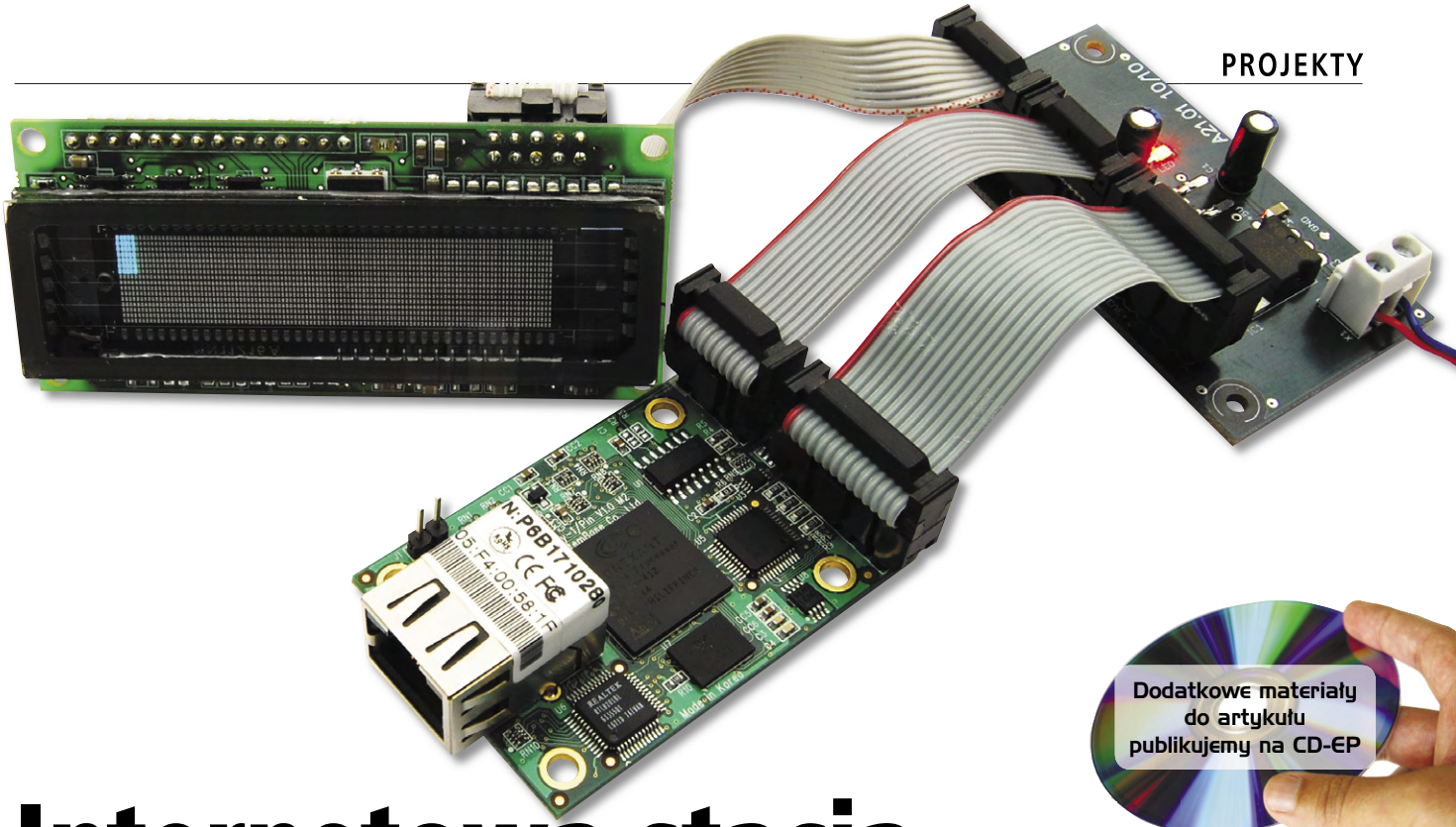


Internetowa stacja pogodowa Yahoo (2)

W drugiej części artykułu, poświęconej czynnościom wykonawczym, zagadnienia stricte elektroniczne, związane z montażem układu, potraktowane są opisem zaledwie dwuzdaniowym. Cała reszta artykułu, czyli ponad 95% jego treści, dotyczy działań software'owych. Są to proporcje znamienne dla tego działu elektroniki. Zamiast lutować i mierzyć uruchamia się program, co w tym przypadku sprowadza się do uruchamiania kolejnych aplikacji linkusowych.

Rekomendacje:

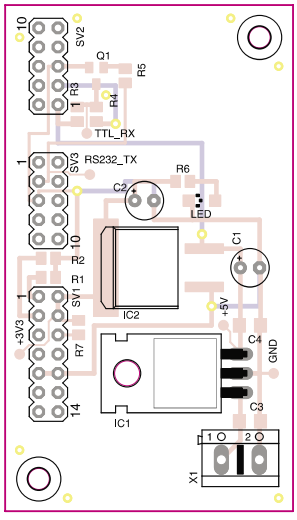
propozycja dla nowoczesnych elektroników nie stroniących od projektów opartych na rozwiązaniach programistycznych.

Montaż

Montaż części elektronicznej stacji pogodowej Yahoo jest bardzo prosty. Na płytce znajduje się tylko kilka gniazd, do których dołączamy wyświetlacz i moduł ethernetowy. Schemat montażowy przedstawiono na **rys. 6**.

Oprogramowanie

Aplikację sterującą napisano w języku C za pomocą dostarczonego przez producenta pakietu SDK (*Software Development Kit*). Środo-



Rys. 6. Schemat montażowy stacji Yahoo

AVT-5141

W ofercie AVT jest dostępna:
– [AVT–5141A] – płytka drukowana

PODSTAWOWE PARAMETRY

- Wyświetlanie komunikatów pogodowych z serwisu *weather.yahoo.com* przez moduł z wyświetlaczem VFD 2x16 znaków
- Zasilanie 9...15 VDC
- Połączenie z lokalną siecią Ethernet za pomocą modułu Eddy S1/PIN

PROJEKTY POKREWNE wymienione artykuły są w całości dostępne na CD

Tytuł artykułu	Nr EP/EdW	Kit
Higrometr elektroniczny	EdW 11/2001	AVT-2607
Domowa stacja meteo ze zdalnym pomiarem temperatury	EP 4-5/2002	AVT-5060
Wilgotnościomierz cyfrowy	EP 1/2006	AVT-914
PC-termometr termometr internetowy	EdW 5/2006	AVT-2787
Domowa stacja pogodowa	EP 12/2006	AVT-961

wisko uruchomieniowe jest oparte o kompilator gcc, zaś przygotowane programy uruchamiane są w systemie operacyjnym *uClinux* z jądrem w wersji 2.4. Środowisko to jest przeznaczone dla szeroko pojętych zastosowań *embedded*. Razem z systemem dostajemy komplet aplikacji, jakie tylko możemy sobie wyobrazić, gdy pomyślimy o sieciowym zastosowaniu modułu. Mamy, zatem do dyspozycji:

- **busybox** – jest to zestaw typowych narzędzi UNIX-owych skompresowanych do postaci niewymagającej sporych zasobów sprzętowych. Busybox zawiera następujące aplikacje: *cat, chgrp, chmod, chown, cp, date, dmesg, echo, free, hostname, kill, ln, ls, mkdir, mknod, mound, mv, ping, ps, pwd, rm, rmdir, route, stty, chroot, getty, ifconfig, insmod, lsmod, rmmod, telnet, ftp, wget*.
- **tinylogin** – zestaw aplikacji związanych z dostępem do systemu. Dostarcza takich poleceń jak: *adduser, login, passwd*.
- **boa** – serwer WWW przygotowany z myślą o systemach wbudowanych. Aplikacja jest uruchamiana automatycznie przy starcie systemu.
- **ftpd** – okrojona wersja demona ftp. Startuje zawsze z systemem. Dostępny jest tylko użytkownik *anonymous*. Upload plików jest możliwy jedynie do folderu */var* z racji tego, iż jest on tworzony w pamięci RAM. Konsekwencją jest to, że jego zawartość znika po każdym restarcie systemu.
- **telnetd** – demon usługi Telnet. Uruchamiany przy starcie systemu na standardowym porcie.
- **dhcpcd** – demon klienta *dhcp*. Umożliwia dynamiczną konfigurację urządzenia w sieci, w której pracuje serwer *dhcp*.

Moduł jest dostarczany w postaci gotowej do użycia, jako most RS232<–>Ethernet. Domyślne oprogramowanie umożliwia konfigurację urządzenia poprzez interfejs WWW oraz Telnet. Konfiguracja obejmuje wszystkie zasoby sprzętowe, a więc interfejs sieciowy (konfigurację IP), port szeregowy oraz linie GPIO. Moduł Eddy może pracować w jednym z trzech trybów (*Execution Mode*):

- „0” (Tryb Eddy – *Eddy Mode*) – moduł startuje z domyślnym firmwarem (opisanym powyżej). Nie ma możliwości wykonywania jakiejkolwiek aplikacji użytkownika. Domyślny adres IP modułu to 192.168.0.223.
- „1” (Tryb Użytkownika – *User Mode*) – tryb ten domyślnie uruchamia aplikację zlokalizowaną w katalogu */usr* i nazwaną *userapp*. W tym trybie jest dostępny interfejs telnet modułu, natomiast adres IP jest ustalany na stałe na 192.168.0.224. W trybie użytkownika zostaje uruchomiony również serwer WWW boa pod warunkiem, iż w katalogu */usr/htdocs* są zlokalizowane pliki *html*.
- „2” (Tryb Eddy + Użytkownika). Moduł będzie zachowywał się tak, jak w przypadku trybu „0”, ale zostanie też uruchomiona aplikacja użytkownika.

Oprócz adresu IP modułu wynikającego z pracy w poszczególnych trybach, zawsze jest on dostępny pod adresem 10.10.1.1. Zabieg ten umożliwia dostęp do modułu, np. w wyniku utraty konfiguracji IP wynikającej z błędnego działania aplikacji użytkownika.

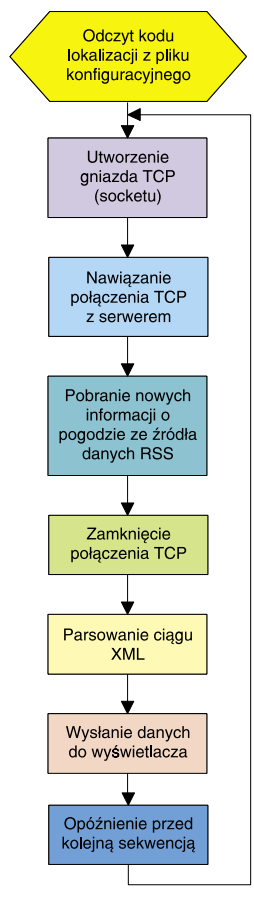
Ponieważ standardowy firmware modułu nie był wystarczający, aby zrealizować założenia projektu, należało posłużyć się środowiskiem rozwojowym (SDK) dostarczonym przez producenta. Pakiet ten pracuje wyłącznie pod kontrolą Linuksa (nie istnieje wersja windowsowa SDK). Po instalacji oprogramowania w katalogu */opt* (domyślnie) zostanie stworzony katalog uClinux. W katalogu */opt/uClinux/bin* zostaną umieszczone linki do narzędzi *gcc* dla modułu, zaś w */opt/uClinux/arm-elf* wykonywalne pliki gcc. Ponadto zostanie zainstalowana biblioteka *binutils* (*arm-elf-binutils*), przykładowe aplikacje oraz narzędzia do tworzenia obrazu aplikacji. Jednym z przykładów dostarczanych przez Sysbas jest kod źródłowy domyślnej aplikacji uruchamianej w trybie pracy *Eddy*. W stacji pogodowej Yahoo skorzystano z tych źródeł w części odpowiedzialnej za konfigurację modułu. Kompletna implementacja stacji pogodowej zawiera się zatem w dwóch niezależnych programach uruchamianych w trybie użytkownika.

Klient Yahoo Weather

Jest to właściwy kod służący do uzyskiwania informacji pogodowej ze źródła danych RSS i prezentacji wyniku na wyświetlaczu VFD. Aplikacja zakłada, że konfiguracja IP jest poprawna oraz że moduł posiada dostęp do Internetu.

Algorytm jego działania przedstawiono na **rys. 7**. Program działa w nieskończonej pętli. Na początku otwierany jest plik z danymi konfiguracyjnymi systemu (*/var/config.cfg* – plik tworzony zawsze przy starcie systemu z danych zaszytych w pamięci NVM. Należy pamiętać, iż operacje zapisu do pliku są możliwe dzięki jego lokalizacji w katalogu */var* w obszarze RAM). W pliku zapisana jest struktura zawierająca element *location*. Pole to jest wypełniane podczas konfigurowania urządzenia, a odbywa się to z poziomu interfejsu WWW (a więc aplikacji WebManagera). Jest to tablica elementów typu `char`. Ponieważ kod lokalizacji ma stałą długość, kopiujemy z pola zawsze 8 znaków do zmiennej lokalnej. Następnie tworzymy ciąg stanowiący treść zapytania, które wyślemy do serwera po uzyskaniu połączenia TCP Ciąg ten ma postać:

```
"GET http://weather.yahooapis.com/forecastrss?p=PLXX0029&u=c\n"
```

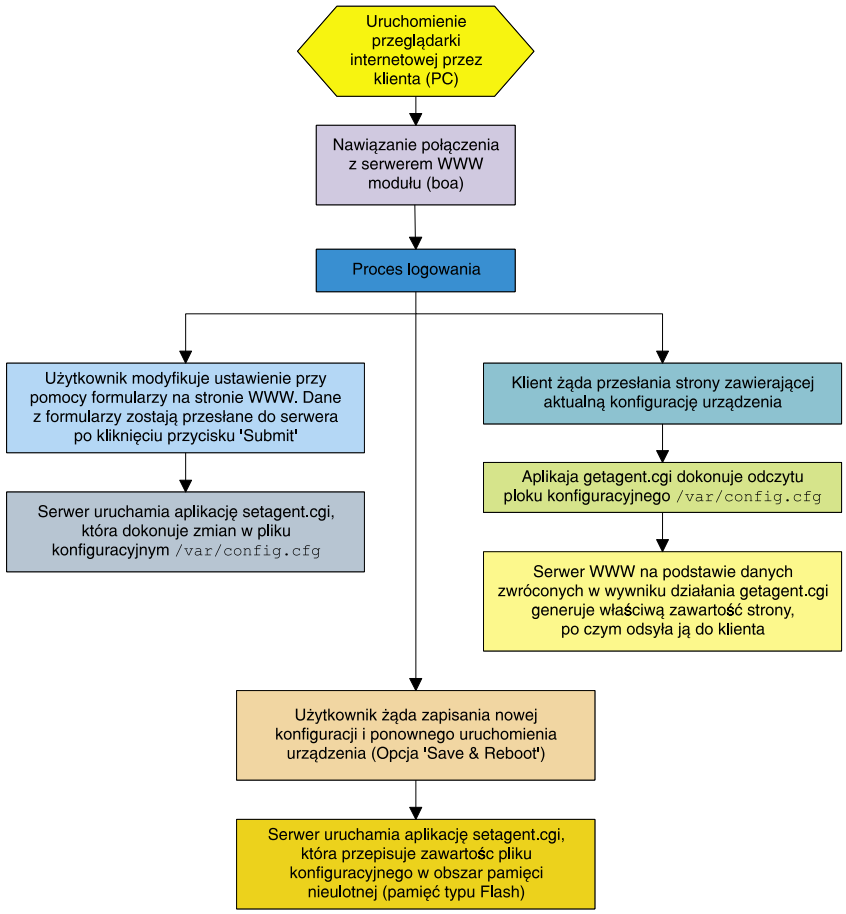


Rys. 7. Algorytm działania aplikacji klienta

gdzie: `PLXX0029` to kod odczytany z pliku konfiguracyjnego.

Mając taki zestaw danych wejściowych możemy przystąpić do próby nawiązania połączenia z serwerem. W tym celu tworzymy gniazdo TCP przy pomocy funkcji *socket()* oraz przygotowujemy dane do nawiązania połączenia. Nazwa hosta, z którym chcemy nawiązać łączność to „weather.yahooapis.com". Ponieważ dysponujemy jedynie jego nazwą (nie ma potrzeby polegania w tym przypadku na jawnych adresach IP), musimy użyć systemowego klienta DNS do wyłuskania adresu IP hosta. Możemy tego dokonać wywołując funkcję *gethostbyname(),* z parametrem w postaci nazwy serwera. Funkcja (rozwiązując nazwę przy użyciu *resolvera* systemowego) w rezultacie zwróci wskaźnik na strukturę opisującą zdalnego hosta. Jednym z pól struktury jest jego adres IP. Po jego wyłuskaniu, za pomocą funkcji *connect()* dokonujemy próby nawiązania połączenia z hostem. Funkcja w argumencie przyjmuje uchwyt do utworzonego socketu, adres hosta oraz port, na którym ma zostać nawiązane połączenie. Wynik działania funkcji jest sygnalizowany poprzez wartość zwracaną przez funkcję. W przypadku powodzenia, do serwera zostaje wysyłany ciąg znaków stanowiący żądanie danych pogodowych dla zdefiniowanej lokalizacji (przy pomocy funkcji *write()*). W następnym kroku, w pętli, odczytujemy przy użyciu funkcji *read()* dane napływające do gniazda, gromadząc je w lokalnym buforze. Odczyt trwa do momentu przekroczenia czasu operacji. Podejście takie jest konieczne ze względu na to, iż klient nie wie, jakiej liczby bajtów powinien się spodziewać (ze względu na tekstowy charakter danych). Po zakończeniu operacji odczytu następuje zamknięcie gniazda (funkcja *close()*).

Dysponując nowymi danymi w postaci ciągu XML możemy przystąpić do jego analizy w celu wyodrębnienia interesujących nas informacji, które następnie zostaną przekazane do wyświetlacza. Aktualizacja danych pogodowych w serwisie Yahoo odbywa się co 30 min. Opisywana w artykule stacja pogodowa aktualizuje w tym czasie swoje dane kilkukrotnie.



Rys. 8. Sposób dynamicznego generowania zawartości stron WWW przez serwer boa

WebManager

Jest to aplikacja bazująca na oryginalnym oprogramowaniu firmowym uruchamianym w trybie „0". Jej funkcjonalność obejmuje między innymi pełną konfigurację IP urządzenia. Właśnie z tej części skorzystano w opisywanym projekcie. Do dyspozycji mamy interfejs WWW, a w nim dwie zakładki: *Network Settings* (odpowiedzialna za konfigurację sieci) oraz *Yahoo! Weather*. Ta ostatnia jest odpowiedzialna za konfigurację lokalizacji, dla jakiej będą prezentowane dane pogodowe.

Serwer WWW (boa) dostarczany wraz z modułem Eddy obsługuje dynamicznie generowane strony HTML. System korzysta z technologii CGI (*Common Gateway Interface*). Serwer po otrzymaniu żądania od klienta dotyczącego dynamicznie generowanej strony WWW (w naszym przypadku są to strony WebManagera) uruchamia jeden z dwóch programów CGI (*getagent.cgi*). Po zmianie ustawień i ich zatwierdzeniu przez użytkownika, wywoływana jest druga aplikacja - *setagent.cgi*. Konfiguracja urządzenia jest przechowywana w pliku */var/config.cfg*, który jak pamiętamy, znajduje się w obszarze pamięci RAM. Po kliknięciu przez użytkownika na przycisk linku *Save & Reboot* aplikacja *setagent.cgi* przepisuje zawartość pliku do obszaru nieulotnej pamięci urządzenia (NVM). Na **rys. 8** przedstawiono ten proces w postaci grafu. Mając do dyspozycji kod źródłowy obu aplikacji CGI możemy dowolnie modyfikować zachowanie się interfejsu WWW urządzenia. Dodatkowo możemy edytować same pliki HTML dostarczające statycznej zawartości stron. Aplikacje CGI w module Eddy stanowią jedyne powiązanie systemu z interfejsem WWW urządzenia. Ich zaletą jest to, że są to normalne aplikacje pisane w języku C, co pozwala zachować spójność kodu lub używać wspólnej funkcjonalności przez zasadniczą aplikację modułu oraz przez jego interfejs WWW.

BootTime Configurator

Ostatnią aplikacją, która pozwoli na samodzielne działanie całego sytemu jest program, który tuż po starcie systemu utworzy plik konfiguracyjny zapisany w pamięci Flash oraz odczytując z niego konfigura-

cję dokona wszelkich niezbędnych zmian, aby urządzenie zachowywało się tak, jak spodziewał się tego użytkownik. Kod źródłowy takiej aplikacji jest również dostarczany przez producenta modułu. Jest to dokładnie ta sama aplikacja, która jest używana w domyślnym firmwarze modułu. Ze względu na to, że podczas pracy w trybie użytkownika moduł ma predefiniowaną konfigurację (np. adres IP modułu to 192.168.0.224), należy tę aplikację uruchomić jako jedną z pierwszych. Kod źródłowy znajduje się w piku *porter.c*.

I wreszcie... userapp.c

Udało nam się dotrzeć do końca. Jak założyliśmy na początku, moduł będzie startował w trybie „1" (tryb użytkownika). Poza statyczną konfiguracją IP i uruchomieniem wszystkich standardowych aplikacji (np. *busybox*), system uruchomi też plik wykonywalny */usr/userapp*. Reszta procesu bootowania zależy od nas (a właściwie od kodu źródłowego *userapp.c*). W stacji pogodowej Yahoo wygląda on tak, jak to pokazano na **list. 2**.

Kolejne aplikacje uruchamiane są za pomocą funkcji wbudowanej *system()*, która zachowa się tak, jakby użytkownik wydał polecenie korzystając z konsoli syatemowej. W wyniku działania *userapp*, w systemie zostną utworzone trzy nowe procesy.

W ten oto dość nieświadomy sposób skorzystano z dobrodziejstw posiadania systemu operacyjnego. Fakt, że systemem tym jest uClinux spowodował, iż tworzenie nowych wątków programu sprowadziło się do uruchamiania kolejnych aplikacji. Żeby system stał się w pełni samodzielny, należy jeszcze nasze aplikacje połączyć w ten sposób, aby otrzymać jeden obraz pamięci Flash urządzenia. W tym celu należy skopiować wszystkie pliki wykonywalne (przede wszystkim aplikację *userapp*) do folderu utworzonego podczas procesu instalacji SDK: */opt/uClinux/romdisk/usr*. Następnie uruchamiamy aplikację */opt/uClinux*/romdisk/mkfs, w wyniku działania której zostanie utworzony plik *user.img*. Tak stworzony plik binarny umieszczamy w folderze */var* modułu, korzystając przy tym z połączenia FTP. Kiedy plik obrazu znajduje się w pamięci modułu, otwieramy nową sesję Telnet i wydajemy polecenie *#upfirm /var/user.img*. Urządzenie samo nadpisze odpowiedni obszar pamięci nieulotnej, po czym poprosi o restart (komenda *#reboot*). Dodatkowo należy pamiętać o tym, iż moduł musi być przestawiony w tryb pracy użytkownika (komenda *#modechange 1*).

Obsługa stacji pogodowej Yahoo

Po podłączeniu urządzenia do segmentu sieci LAN (najczęściej będzie to po prostu sieć domowa) musimy nawiązać próbę połączenia z modułem. Domyślnie startuje on z uaktywnionym klientem DHCP. Jeśli w naszej sieci działa serwer DHCP i jest skonfigurowany tak, aby przyjmować nowe urządzenia (a nie np. tylko te ze znanym adresem MAC), wówczas poprzez jakikolwiek interfejs statusowy serwera możemy odnaleźć nowo przydzielone IP modułu Eddy. Dodatkowo, moduł tuż po starcie wyświetla swój obecny adres IP. Jeśli z jakichś powodów tak nie jest, wówczas musimy skomunikować się z modułem przy pomocy alternatywnego adresu IP (10.10.1.1). Po wpisaniu tego adresu, w przeglądarce powinniśmy zobaczyć ekran logowania, taki jak na **rys. 9**.

List. 2. Kod źródłowy aplikacji *userapp.c*

```
#include <p_define.h>
#include <eddylib.h>

int main(void)
{
  /* Reconfigure system. */
  system("/usr/porter");

  /* Start boa. */
  system("/sbin/boa start");
  /* Start Yahoo Weather client. */
  system("/usr/client &");

  while(1)
  {
    sleep(10);
  }
  return 0;
}
```

Rys. 9. Ekran logowania stacji pogodowej Yahoo

Po podaniu domyślnych danych logowania (użytkownik: Eddy, hasło: 99999999) powinniśmy ujrzeć główną stronę interfejsu WWW. Dwie najważniejsze dla nas podstrony to: *Network Settings* i *Yahoo! Weather*. Na pierwszej z nich ustalamy konfigurację sieciową urządzenia (statyczną, bądź dynamiczną – **rys. 10**). Najprostszym rozwiązaniem jest tutaj zastosowanie dynamicznej konfiguracji sieci (*Dynamic Host Configuration Protocol*). Ustawienie to spowoduje, iż stacja stanie się całkowicie bezobsługowa, i na przykład po zmianie architektury sieci LAN nie będzie musiała być przekonfigurowana. Klient DHCP (w naszym systemie jest to demon *dhcpcd – DHCP Client Daemon*) nada urządzeniu adres IP, maskę podsieci oraz bramę domyślną (wszystkie te ustawienia pochodzą z serwera DHCP). W przypadku, gdy nasza sieć nie dysponuje serwerem DHCP, możemy skorzystać z możliwości statycznej konfiguracji (z rozwijanego menu *Line Type* należy wybrać opcję *Static IP*).

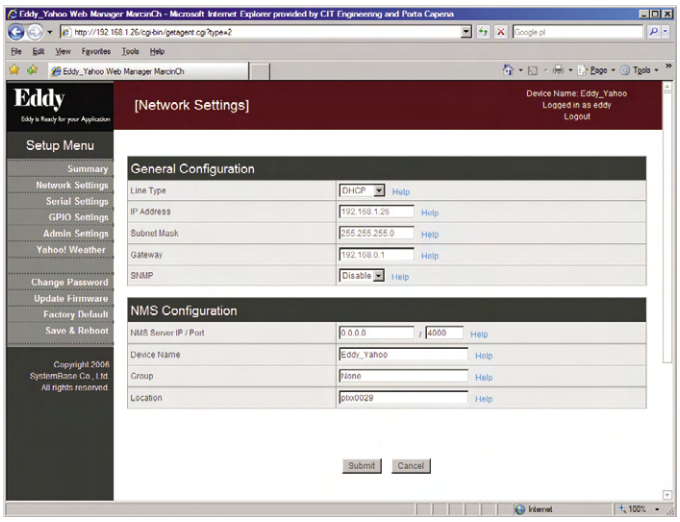
Na drugiej, ważnej dla nas podstanie (*Yahoo! Weather*), wpisujemy kod lokalizacji, dla której interesują nas dane pogodowe. Kiedy już wszystkie ustawienia są poprawne, klikamy na przycisk *Submit*, a następnie wybieramy opcję *Save & Reboot*. Po chwili powinniśmy ujrzeć na wyświetlaczu aktualną temperaturę oraz jeden z opisów warunków pogodowych, które przestawiono w tab. 2 (EP7/2008).

Dostępne są też pozostałe opcje ze standardowej aplikacji modułu, takie jak chociażby zmiana hasła logowania. Należy pamiętać,

iż podczas pracy z interfejsem WWW, programy CGI wykonują operacje na pliku znajdującym się w obszarze pamięci RAM. Dopiero po wybraniu opcji *Save & Reboot* treść pliku zostanie zachowana w pamięci Flash.

Podsumowanie

Moduł EddyT V1.x okazał się przyjaznym urządzeniem. Do jego prawidłowego wykorzystania potrzebna była minimalna wiedza z zakresu elektroniki. W pewnym momencie przestaje on funkcjonować w świadomości użytkownika jako procesor i pamięć, a zaczyna istnieć jako kolejna konsola Terminalu (tak jak PC z zainstalowanym Linuxem). Moduł nadaje się idealnie dla pasjonatów i profesjonalistów stosujących Linuxa w systemach wbudowanych. Podczas tworzenia projektu miałem okazję sprawdzić możliwość przenoszenia aplikacji pisanych w języku C pomiędzy różnymi platformami. Ponieważ pojawił się problem parsowania ciągu XML, postanowiłem poszukać w sieci projektów tego typu, udostępnianych na zasadach *open-source*. Jeden z nich udało się nawet skompilować i uruchomić w module, ale niestety okazało się, że potrzebuje on do działania znacznie więcej zasobów, niż te którymi dysponuje moduł. Konieczne było stworzenie własnego, prostego parsera.



Rys. 10. Podstrona ustalająca konfigurację sieciową urządzenia

Dużo poważniejszym, moim zdaniem, problemem jest brak możliwości debugowania aplikacji innymi metodami niż wywołania funkcji *printf*. Dla jednych będzie to wada (będą to raczej profesjonalni wymagający programiści), dla innych będzie to nieznaczący fakt (elektronicy-programiści przyzwyczajeni to takich warunków pracy). Problem ten zdaje się być dostrzegany przez twórców modułu, jako że jego nowe wersje są dostarczane z kompletnym środowiskiem uruchomieniowym opartym o Eclipse, którego jednym z narzędzi jest debbuger.

Marcin Chruściel, EP
marcin.chrusciel@ep.com.pl

R E K L A M A