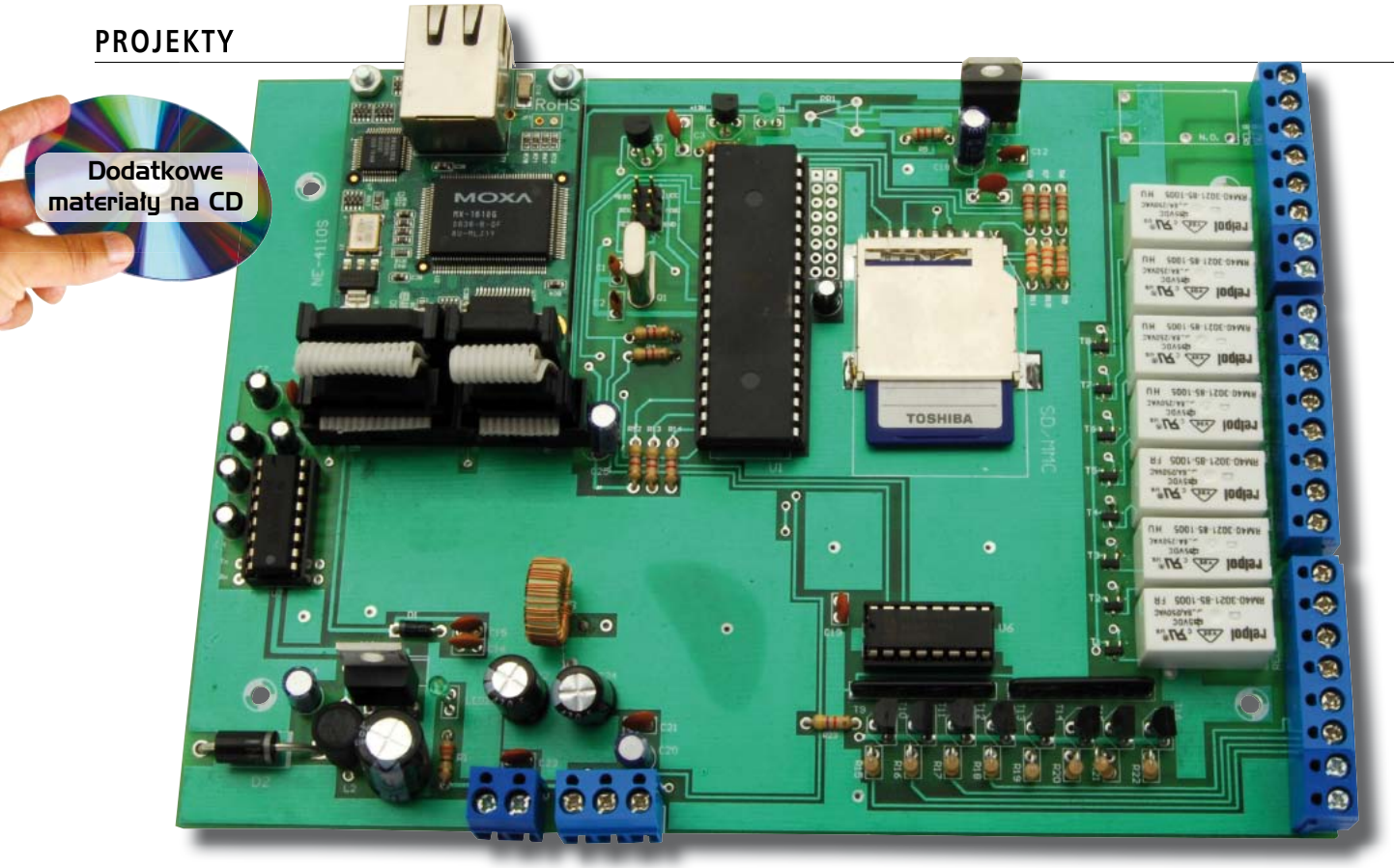


Uniwersalny sterownik Ethernetowy z modułem MOXA NE-4110S

AVT–5200

W ofercie AVT:
AVT–5200A – płytka drukowana

PODSTAWOWE PARAMETRY
• Płytka o wymiarach 179×129 mm
• Zasilanie 9...12 VDC
• Wyposażony w moduł MOXA NE-4110S (Ethernet 10/100 Mbit/s)
• Kontrolowany przez mikrokontroler AVR – ATmega32
• Sterowanie za pomocą protokołu Telnet, TCP/IP, serwera WWW oraz plików tekstowych umieszczanych w folderach wymiany
• 8 wyjść przekaźnikowych, 8 konfigurowanych wejść analogowych (pomiar napięcia) lub cyfrowych

PROJEKTY POKREWNE wymienione artykuły są w całości dostępne na CD

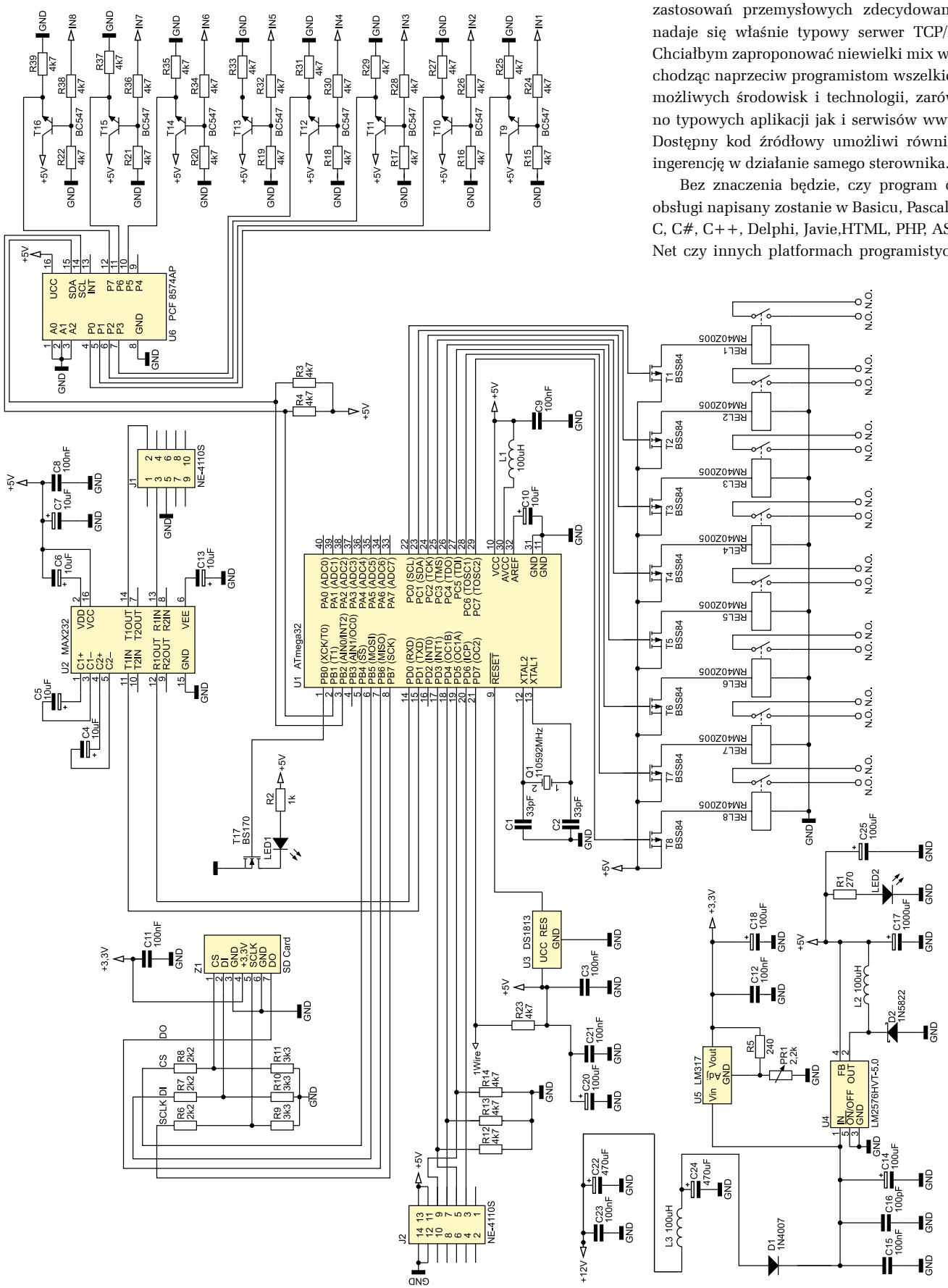
Tytuł artykułu	Nr EP/EdW	Kit
Przekaźnik internetowy	EP 11/2008	AVT-5157
Internetowy sterownik urządzeń	EdW 3/2008	AVT-2859
Internetowy interfejs dla mikrokontrolera	EP 3-5/2002	AVT-5055
Uniwersalny interfejs internetowy	EP 4-5/2006	AVT-927
Karta wejść z interfejsem Ethernet	EP 10/2006	AVT-953
Uniwersalny interfejs ethernetowy	EP 1/2007	AVT-1443
Karta przekaźników sterowana przez internet	EP 2/2007	AVT-966
Sterownik z interfejsem TCP/IP	EP 3/2007	AVT-974
Zdalny system pomiarowy z interfejsem Ethernet	EP 9/2007	---
Pilot w WiFi	EP 8/2009	AVT-5197
Ethernetowy sterownik I/O	EP 11/2006	AVT-956
Serwer HTTP	EP 1/2009	AVT-5166

Wśród Czytelników EP ogromną popularnością cieszą się wszelkiego typu rozwiązania urządzeń komunikujących się poprzez sieć Ethernet. Wiele razy publikowaliśmy projekty różnych serwerów embedded, ale ten jest szczególny. Można z nim komunikować się zarówno poprzez stronę WWW, jak i z użyciem protokołu TCP/IP. Dzięki temu sterownik można kontrolować również z poziomu własnej aplikacji napisanej w dowolnym języku programowania dla dowolnego systemu operacyjnego mającego możliwość komunikowania się poprzez sieć cyfrową Ethernet.

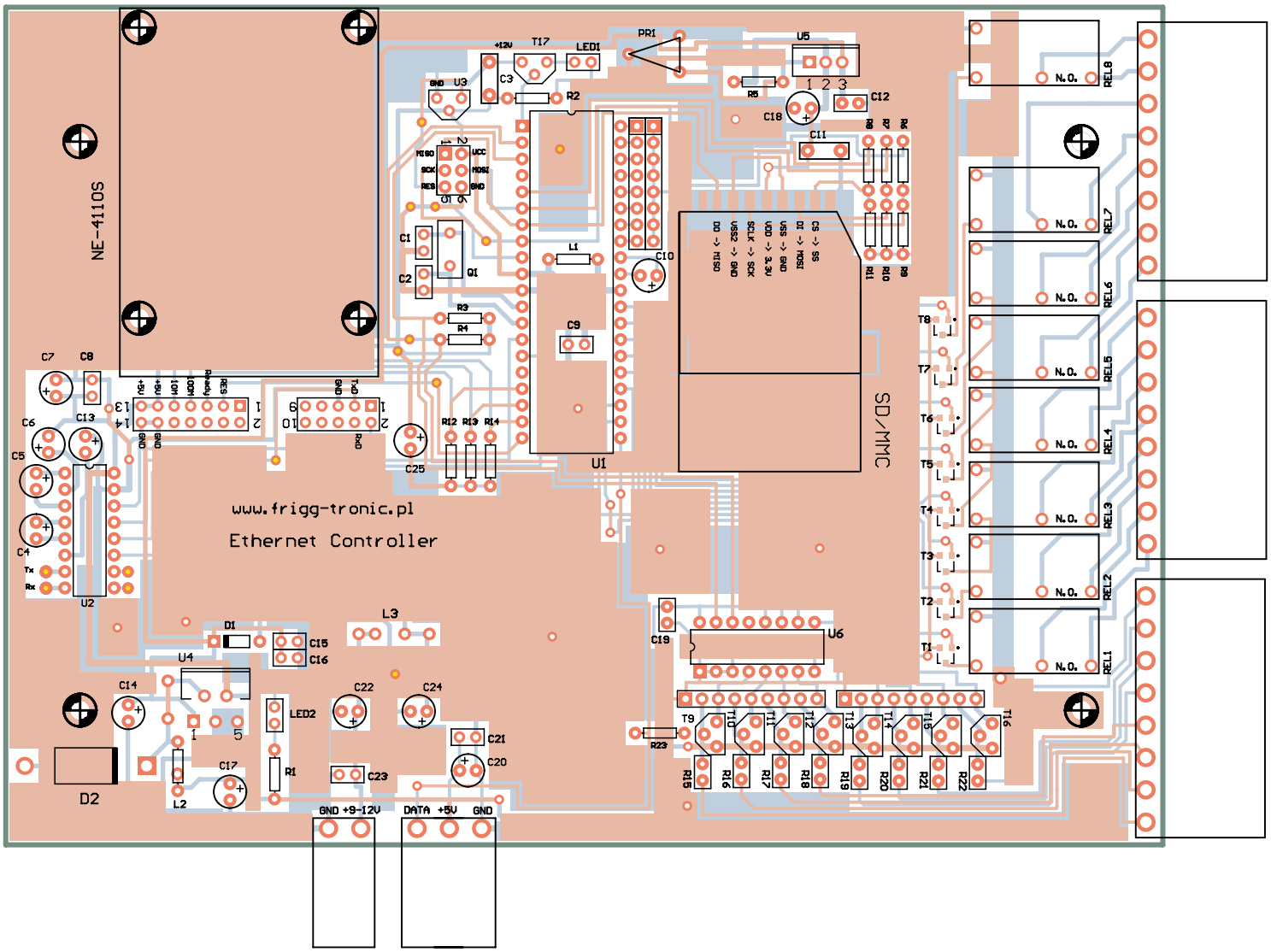
Rekomendacje: *szczególnie polecamy elektronikom-konstruktorom zajmującym się automatyką budynków i obiektów, jak również budową systemów rozproszonych.*

Do tej pory na łamach Elektroniki Praktycznej ukazało się już przynajmniej kilka sterowników pracujących z wykorzystaniem Internetu czy Intranetu. Różnica tkwiła w sposobie zapewnienia połączenia oraz w interfejsie użytkownika (serwer/klient telnetowy lub witryna Web). Ten ostatni nie wymagał instalowania dodatkowego oprogramowania, bo strona www zawarta była wewnątrz sterownika. Niewątpliwą jednak przewagą serwerów telnetowych jest ich szybkość działania. Minusem stron www zawartych wewnątrz mikrokontrolera była i jest ich spora prostota, zwłaszcza pod względem graficznym. W moim odczuciu do zastosowań przemysłowych zdecydowanie nadaje się właśnie typowy serwer TCP/IP. Chciałbym zaproponować niewielki mix wychodząc naprzeciw programistom wszelkich możliwych środowisk i technologii, zarówno typowych aplikacji jak i serwisów www. Dostępny kod źródłowy umożliwi również ingerencję w działanie samego sterownika.

Bez znaczenia będzie, czy program do obsługi napisany zostanie w Basicu, Pascalu, C, C#, C++, Delphi, Javie,HTML, PHP, ASP. Net czy innych platformach programistycz-



Rys. 1. Schemat ideowy uniwersalnego sterownika Ethernet

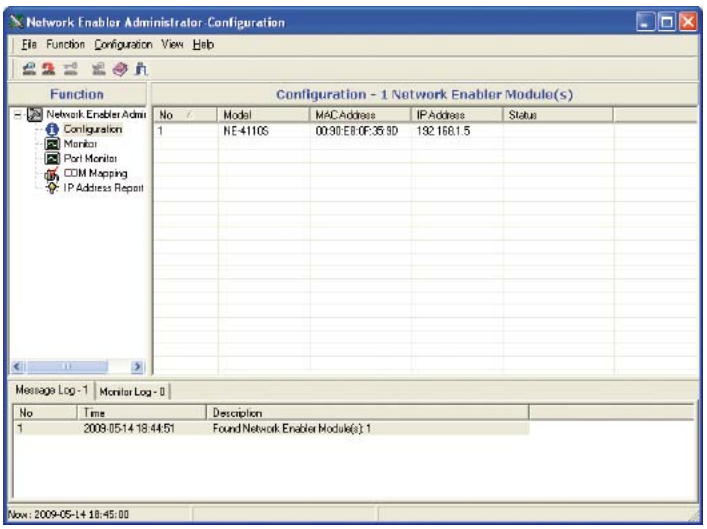


Rys. 2. Schemat montażowy sterownika

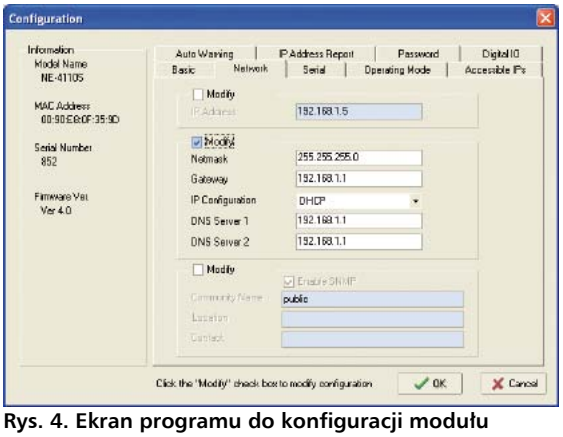
nych, których nie wymieniłem. Jawny protokół pozwala na szybką komunikację i oprogramowanie typowego klienta telnetowego. Myślę, że nawet mniej doświadczonym programistom napisanie takiego programu nie sprawi większego problemu. Alternatywnie skorzystać będzie można z dołączonej testowej aplikacji lub połączyć się ze sterownikiem wykorzystując do tego celu typowe pliki tekstowe, z tworzeniem których poradzi sobie na pewno już każdy programista przy użyciu dowolnego środowiska. Zwolennicy programów *.exe napiszą sobie własny program, natomiast fani środowisk www, będą mogli stworzyć dowolnie rozbudowaną (zwłaszcza pod względem graficznym) witrynę WWW.

Opis układu

Serwer zmontowano na płytce dwustronnej z metalizacją. Schemat ideowy umieszczono na **rys. 1**, a montażowy na **rys. 2**. Sercem sterownika jest mikrokontroler ATmega32. Zupełnie wystarczający jest ATmega16, jednak ten pierwszy daje większe możliwości rozbudowy. Taktowanie zapewnia rezonator kwarcowy o częstotliwości 11,0592 MHz. Nad prawidłowym zerowaniem mikrokontrolera po starcie czuwa DS1813-10. Komunikacja z modułem MOXA odbywa się poprzez interfejs szeregowy RS232C, dlatego do zapewnienia odpowiednich poziomów sygnałów jest potrzebny MAX232 lub jego odpowiednik, pracujący w typowej dla siebie konfiguracji. Sterowanie przekaźnikami realizowane jest bezpośrednio poprzez porty mikrokontrolera i tranzystory MOSFET z kanałem „P" typu BSS84. Wejścia podłączono do Expandera PCF8574AP o adresie odczytu 113. Takie

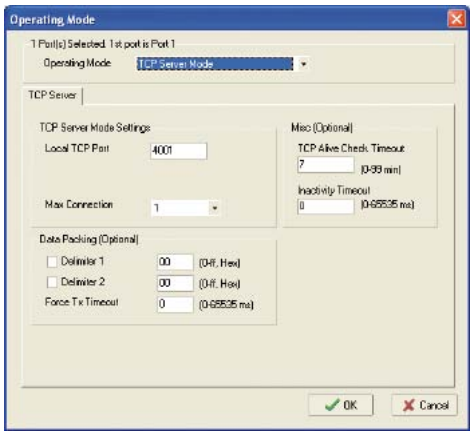


Rys. 3. Ekran programu *Network Enabler*



Rys. 4. Ekran programu do konfiguracji modułu MOXA NE-4110S

Rys. 5. Ustawienie parametrów transmisji szeregowej



Rys. 6. Ustawienie parametrów serwera TCP

rozwiązanie pozostawia wolny port A mikrokontrolera, mogący pracować jako wejścia przetwornika A/C. Rolę filtra zasilania przetwornika pełnią cewka L1 i kondensator C10.

Cały układ zasilany jest napięciem 9...12 VDC. Ze względu na straty mocy na układzie typowego stabilizatora liniowego, zastosowano stabilizator impulsowy LM2576. Oprócz niego, na schemacie znajduje się jeszcze regulowany stabilizator liniowy LM317. Przewidziany został do zasilania karty SD, której gniazdo również umieszczono na płytce sterownika. Do prawidłowej pracy sterownika nie jest ona potrzebna, podobnie jak LM317, jednak pozostawia otwartą furtkę do ewentualnej rozbudowy. Przed włączeniem karty SD do układu, trzeba będzie wcześniej przy pomocy potencjometru PR1 na wyjściu LM317 ustawić napięcie 3,3 V wymagane do zasilania karty.

Konfiguracja modułu MOXA NE-4110E

Po uruchomieniu układu i zaprogramowaniu mikrokontrolera, można przystąpić do konfiguracji modułu MOXA. W tym celu należy:

1. Ustawić adres karty sieciowej na 192.168.127.100. Moduł powinien być widoczny pod adresem 192.168.127.254. Komenda „ping 192.168.127.254” powinna dać prawidłowy rezultat.
2. Zainstalować program Network Enabler i wybrać opcję zaznaczoną na rysunku. Jeśli są do sieci podłączone jakieś inne moduły MOXA, to program automatycznie je wyszuka i wszystkie wyświetli. Można się nim więc posłużyć również do identyfikacji adresu modułu, który nie jest nam znany. Aby konfiguracja była możliwa, trzeba będzie nadać karcie sieciowej adres z tej samej puli, co nasza sieć.

Konfiguracja przebiega intuicyjnie i nie powinna nikomu sprawić problemu. Interfejs programu jest bardzo czytelny. Najważniejsze parametry do ustawienia to:

– Adres IP, maska, brama (Network). Można je wpisać ręcznie, lub zostawić do przyporządkowania serwerowi DHCP.
– Ustawienia parametrów interfejsu szeregowego (Serial) przyjmujemy domyślnie jako na 9600, 8, N, 1. Jeśli chcemy pracować przy innej prędkości, konieczna będzie również modyfikacja programu mikrokontrolera.
– Tryb pracy (Operating Mode) – od tej pory Moxa pracować będzie w trybie Serwera TCPIP z użyciem portu 4001.

Konfigurację Moxy przeprowadzić można również z poziomu typowej przeglądarki internetowej wpisując adres IP Moxy. Ten sposób konfiguracji oferuje te same możliwości, co program Network Enabler z tą różnicą, że nie daje możliwości wyszukania adresów IP – po prostu trzeba go wcześniej znać. Po zakończeniu konfiguracji, nowo nadany adres IP sprawdzić można typową komendą DOSową. W wykonanej przeze mnie aplikacji nadałem modułowi MOXA adres 192.168.1.3, a komputerowi 192.168.1.100

Praca z układem

Po uruchomieniu sterownika, możemy przystąpić do pisania własnych programów. Przykładów klientów telnetowych dla rozmaitych środowisk w sieci można znaleźć sporo i napisanie tak prostego programu nie powinno przysporzyć nikomu problemu. Protokół komunikacyjny ze sterownikiem można znaleźć poniżej. Możliwe jest oczywiście jego dowolne rozbudowywanie w ramach własnych potrzeb.

WYKAZ ELEMENTÓW

Rezystory
R1: 270 Ω
R2: 1 kΩ
R3, R4, R12...R14, R15...R23: 4,7 kΩ
R5: 240 Ω
R6...R8: 2,2 kΩ
R9...R11: 3,3 kΩ
R24...R39: 4,7 kΩ (drabinka rezystorowa – 2 szt.)
PR2: 2,2 kΩ

Kondensatory
C1...C2: 33 pF
C3, C8, C9, C11, C12, C15, C21, C23: 100 nF
C4...C7, C10, C13: 10 µF/16 V
C16: 50 pF
C14, C18, C20, C25: 100 µF/16 V
C17: 1000 µF/25 V
C22, C24: 470 µF/25 V

Półprzewodniki
U1: ATmega32
U2: MAX232
U3: DS1813-10
U4: LM2756-5.0
U5: LM317
U6: PCF 8574AP
T1...T8: BSS84
T9...T16: BC547
T17: BS170
D1: 1N4007
D2: 1N5822

Inne
Q1: rezonator kwarcowy 11,0592 MHz
L1: 100 µH
L2: 100 µH/0,68A
L3: 100 µH/2A
Z1: Gniazdo karty SD/MMC
REL1...REL8: RM40Z005
LED1, LED2: LED 3 mm
Serwer portu szeregowego Moxa NE-4110S

Format polecenia sterującego pracą przekaźników

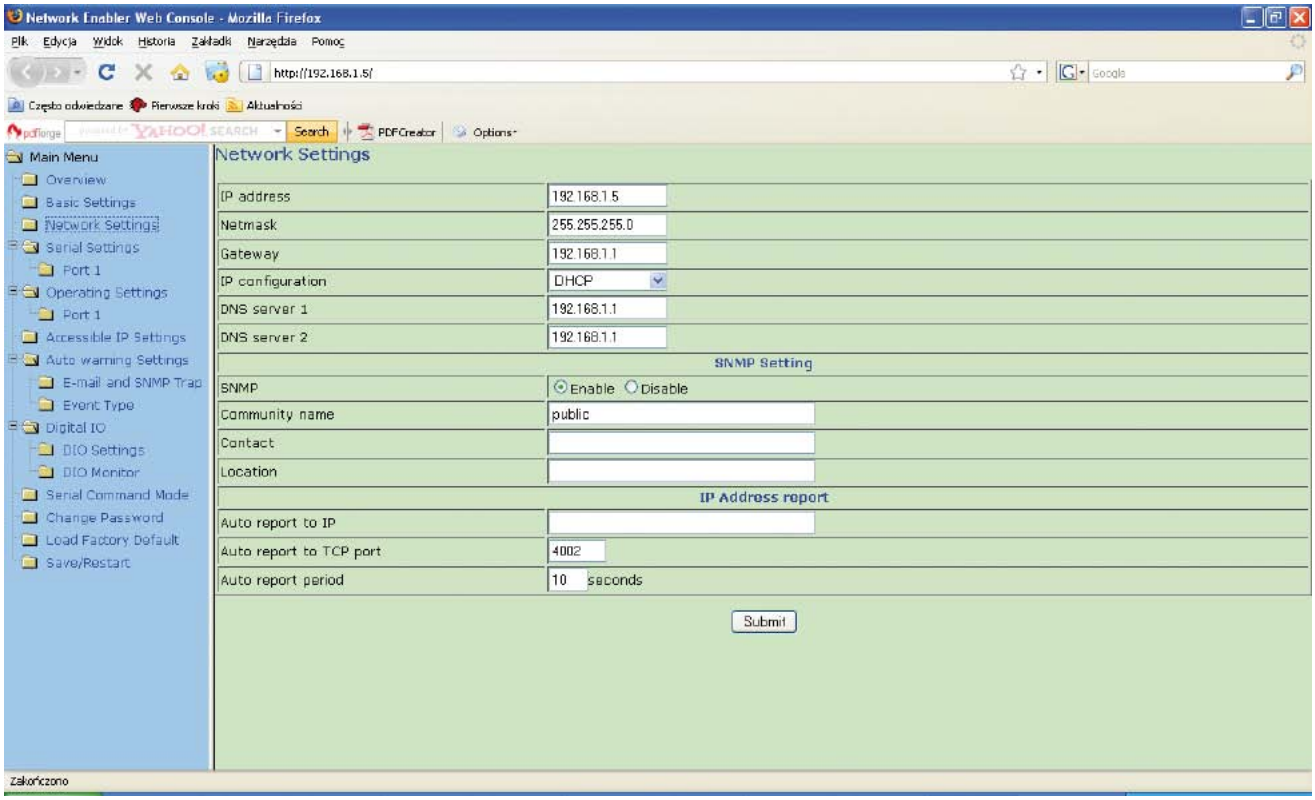
Preffix	Rozkaz	Suffix
Chr(2)	S!OUTXY	Chr(3)

Sterowanie przekaźnikami

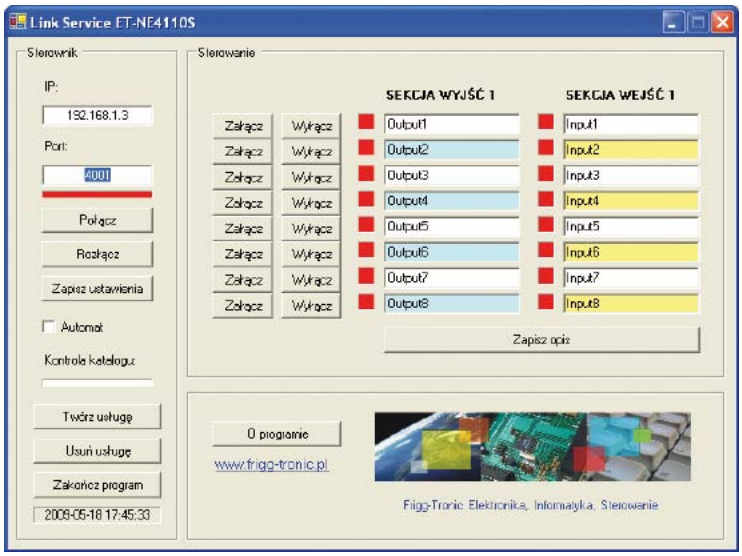
X – musi mieć wartość „S” lub „R” od słów Set oraz Reset (S – załącz, R – wyłącz)

Y – musi mieć wartość 1...8 i określa numer przekaźnika, którego dotyczy polecenie

R E K L A M A



Rys. 7. Konfiguracja MOXA NE-4110S przy użyciu przeglądarki



Rys. 8. Ekran programu *Link Service*

Format informacji zwrotnej o stanie we/wy	
Rozkaz	**Suffix**
08070605040302011817161514131211	Chr(3)

Informacja zwrotna o stanie we/wy

O8...O1 – aktualny stan wyjść. Przyjmuje wartość 0 lub jeden, przy czym „0" oznacza, że przekaźnik jest załączony.

I8...I1 – aktualny stan wejść. Przyjmuje wartość 0 lub jeden. W tym wypadku „0" oznacza stan niski na danym wejściu.

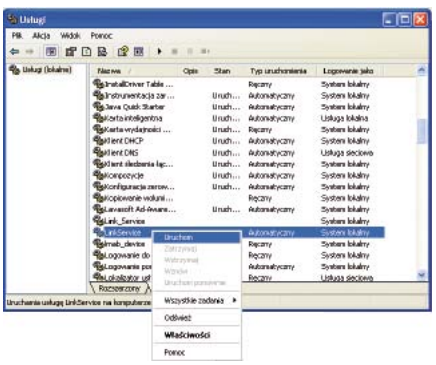
Program testowy

Do projektu dołączony został program testowy, dzięki któremu możemy przetestować uruchomiony sterownik. Umieszczono go na płycie CD_EP9/2009B. Jest typowa aplikacja napisana dla systemu Windows. Przed jego zainstalowaniem należy doinstalować *NET. Framework 1.1* oraz *NET.Framework 1.1 Service Pack 1*, które można pobrać ze strony Microsoft.

Program *Link Service* oferuje podstawowe funkcje umożliwiające sterowanie przekaźnikami, wyświetla aktualny stan we/wy, dając możliwość dowolnego ich opisania oraz zapamiętania tych opisów. To jednak nie wszystko. Na bieżąco kontroluje zadeklarowany folder wymiany danych, umożliwiając sterowanie z wykorzystaniem plików tekstowych, które mogą być generowane z poziomu innej aplikacji czy witryny WWW. Po przetworzeniu pliki te są automatycznie usuwane.

Oczywiste jest, że aby program pełnił swoje funkcje, to musi zostać uruchomiony. Można np. skrót do aplikacji umieścić w tzw. autostarcie, jednakże nawet to rozwiązanie niesie za sobą konieczność przynajmniej jednorazowego zalogowania się do komputera. Operując na rejestrach lub korzystając z uprawnień użytkownika bez hasła, komputer uruchomi się i zaloguje, jednak jest to rozwiązanie mało profesjonalne, bo trudno w tym momencie zabezpieczyć się przed niepożądanym dostępem do komputera osób trzecich.

Wady tej nie ma program testowy, ponieważ po kliknięciu jednego przycisku tworzy usługę systemu Windows. Zaletą wszelkich programów pracujących jako usługi jest to, że startują samoczynnie i działają już w momencie, kiedy widoczny jest ekran logowania, podobnie jak serwery bazodanowe i serwery ASP. Z tego powodu nie ma konieczności logowania się po uruchomieniu komputera.



Rys. 9. *Link Service* pracujący jako usługa systemu Windows

List. 1. Przykład generowania pliku tekstowego w VBScript.
```
<td width="183"><div align="center">
<input name="ZAL1" type="submit" id="ZAL1" value="Za&#322;&#261;cz
&#347;wiat&#322;o JADALNIA" />
<SCRIPT FOR="ZAL1" EVENT="onClick" LANGUAGE="VBScript">
  Set objFSO = CreateObject("Scripting.FileSystemObject")
  Set objFileNew =   objFSo.CreateTextFile("C:\Result\Output1.txt",
ForWriting)
  objFileNew.Write "OUTS1"
  objFileNew.Close
</SCRIPT>
</div></td>
```

List. 2. Przykład generowania pliku tekstowego w JavaScript
```
<input name="Przycisk3" type="submit" id="Przycisk3" value="JavaScript" />
  <SCRIPT FOR="Przycisk3" EVENT="onClick" LANGUAGE="JavaScript">
   var fso  = new ActiveXObject(„Scripting.FileSystemObject");
   var fh = fso.CreateTextFile(„C:\\Result\\Output1.txt „, true);
   fh.WriteLine(„OUTS1");
   fh.Close();
 </SCRIPT>
```

List. 3. Procedura mająca na celu optymalizację kodu programu oraz sposób jej wywołania w HTML
```
function WriteFile()
{
   var fso  = new ActiveXObject(„Scripting.FileSystemObject");
   a = ‚C:\\Result';
   c = ‚.txt';
   filename= a+b+c;   var fh = fso.CreateTextFile(filename, true);
   fh.WriteLine(value);
   fh.Close();
}

Odwołanie do procedury z poziomu html-a.
<input name="Przycisk1" type="submit" id="Przycisk1" value="JavaScript"
onClick="b='1'; value= ‚OUTS1'; WriteFile();" />
```

Wymiana informacji ze sterownikiem odbywać się będzie poprzez tworzone pliki tekstowe w zadeklarowanym przez nas katalogu. Na początku ustalmy, że będzie to katalog *Result*, który automatycznie powinien zostać utworzony przez program testowy.

Aby dokonać zmiany należy wyedytować np. za pomocą programu Notatnik plik *Service.ini* umieszczony w katalogu, w którym została zainstalowana aplikacja *Link Service*. Strukturę pliku przedstawiono niżej:

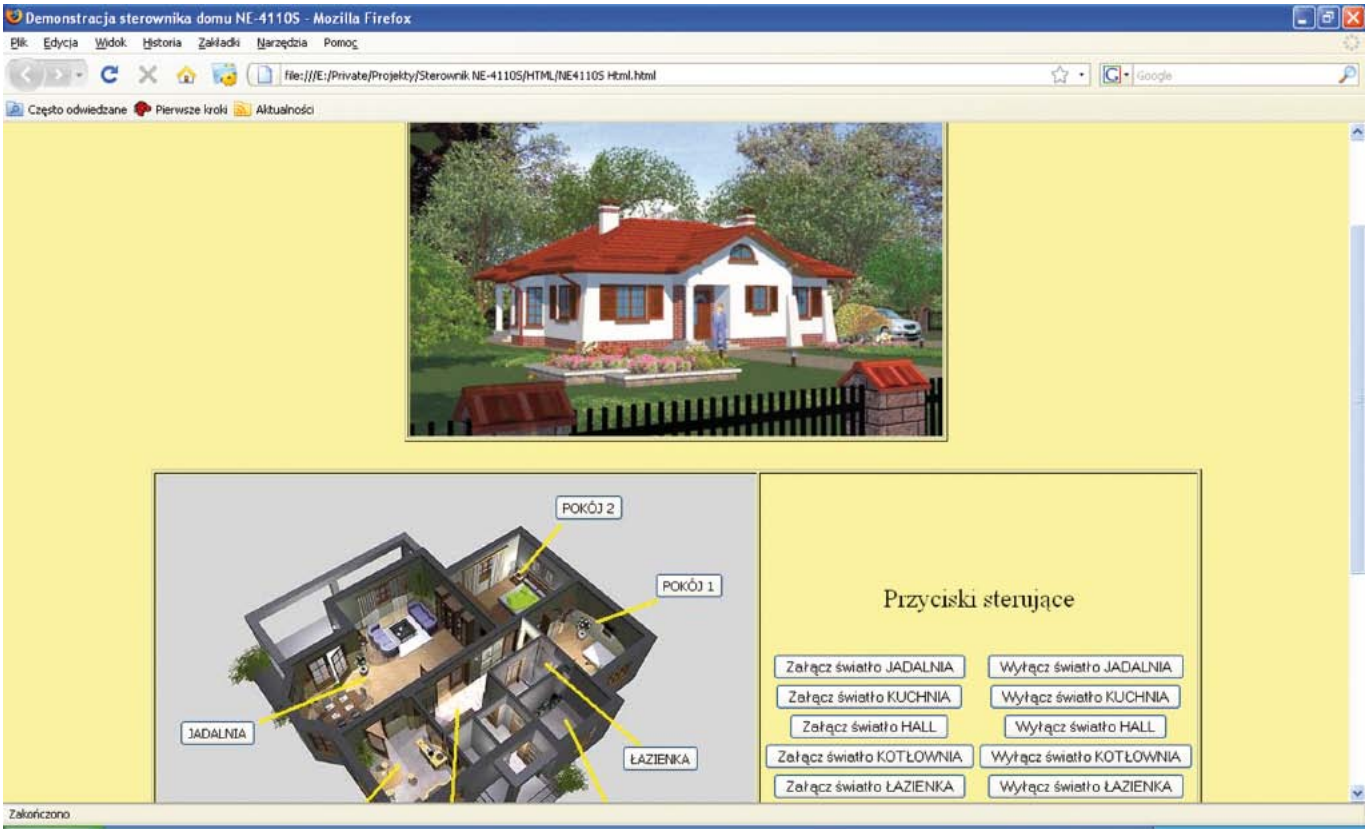
`result_path=C:\Result – określa ścieżkę dostępu dla katalogu Result`

`autostart=0 – określa, czy serwis powinien sam nawiązywać połączenie. (0 – nie, 1 – tak).`

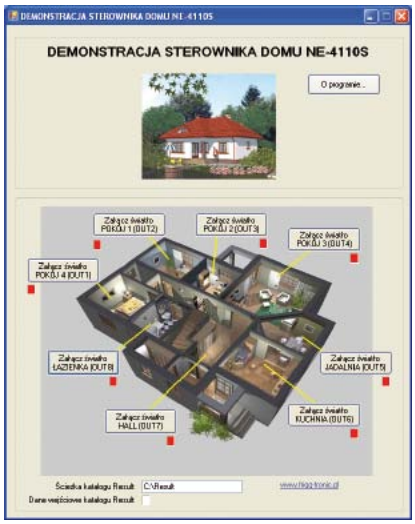
Jeśli zdecydujemy się na to, aby program działał jako usługa, to należy ustawić parametr *autostart* na wartość „1".

Po kliknięciu „Twórz usługę" program automatycznie dokona odpowiednich czynności. Uruchomione usługi można wyświetlić wydając z poziomu linii komend polecenie *services.msc*. Przykładowy rezultat zwracany przez komendę umieszczono na **rys. X**. Nasza nowo utworzona usługa powinna być widoczna. Nie pozostaje nam już nic innego jak tylko ją uruchomić. Wciskamy prawy przycisk na jej nazwie i wybieramy opcję uruchom.

Pliki *.txt do sterowania powinny być formatu OUTPUTX.txt,



Rys. 10. Przykładowa witryna WWW służąca do kontroli sterownika



Rys. 11. Przykładowy ekran aplikacji napisanej w *Visual Basic Express 2008*

gdzie X – jest numerem wyjścia. Treść zrozumiała dla programu, muis mieć następującą strukturę: *OUTXY*, gdzie:

X – przyjmuje wartość S lub R od słów Set oraz Reset (S – załącz, R – wyłącz)

Y – przyjmuje wartość 1-8 i określa numer wyjścia

Przykład strony WWW umożliwiającej sterowanie umieszczono na płycie CD_EP9/2009B w plikach *NE4110S Html JavaScript.html* oraz *NE4110S Html VBScript.html*. Zaleca się wykorzystanie przeglądarki Internet Explorer i załączonym zezwoleniem na używanie formantów ActiveX. Są to przykłady wykorzystania urządzenia do sterownia oświetleniem w domu. Oczywiście, zastosowań samego sterownika może być mnóstwo. Na **list. 1** i **list. 2** zamieszczono przykłady skryptów VBScript oraz JavaScript do wykorzystania we własnych aplikacjach.

Chcąc zoptymalizować kod, można na samym początku stworzyć bardzo prostą, sparametryzowaną procedurę i później odwoływać się do niej. Na **list. 3** umieszczono przykład takiej procedury oraz odwołania do niej w języku HTML.

Program *Demo NE-4110S* napisano w *Visual Basic Express 2008* i jest on w pewnym sensie odzwierciedleniem poprzedniego przykładu, tyle że w formie Windowsowej aplikacji. Aby przeanalizować dostępne źródła, należy pobrać bezpośrednio z witryny *Microsoftu Visual Basic Express 2008* i zainstalować. Zdecydowałem się na ten przykład ze względu na to, że oprogramowanie w wersji Express jest narzędziem darmowym. Wymagana jest tylko rejestracja. Dodatkowym atutem, jeśli chodzi o zastosowane rozwiązanie jest to, że aplikacja co jakiś ustalony czas analizuje plik tworzony przez usługę współpracującą ze sterownikiem, będący odzwierciedleniem aktualnych stanów wejść. Nie ma charakterystycznego i widocznego odświeżania się strony, które byłoby w przypadku analizy tego pliku również przez witrynę WWW.

Przykład tworzenie pliku tekstowego po naciśnięciu jednego z przycisków:

```
Private Sub Button1_Click(ByVal
sender As System.Object, ByVal
e As System.EventArgs) Handles
Button1.Click

        FileOpen(1, "C:\Result\
Output1.txt", OpenMode.Append)
        Print(1, "OUTS1")
        FileClose(1)
End Sub
```

Myślę, że proponowany sterownik będzie doskonałą alternatywą dla wcześniej opublikowanych projektów. W jego konstrukcji postawiono na uniwersalność, więc można go użyć na wiele różnych sposobów, do realizacji rozmaitych zadań.

Rafał Chromik
www.frigg-tronic.pl

Moduł Moxa NE-4110S udostępniła firma:
Elmark Automatyka Sp. z o.o.
ul. Niemcewicza 76, 05-075 Warszawa – Wesoła, *www.elmark.com.pl*