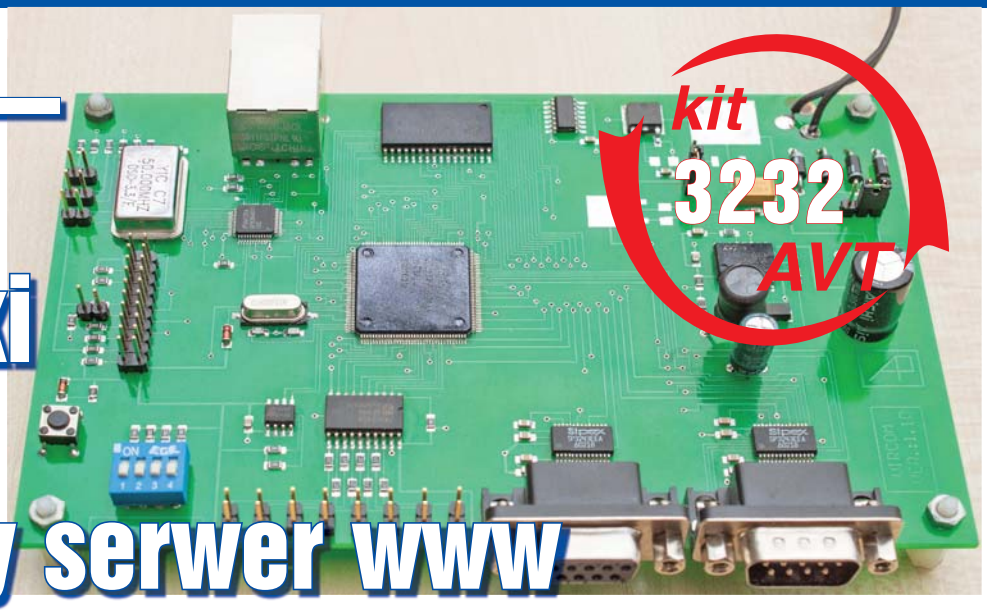
Infinity — system automatyki domowej

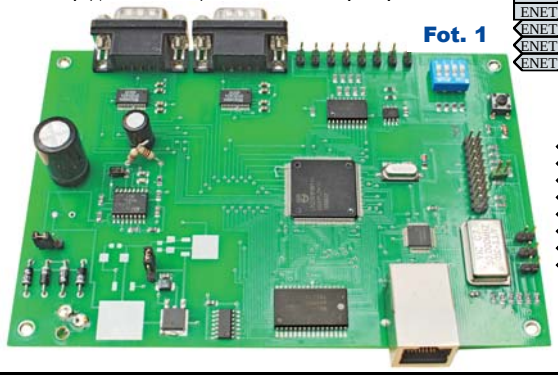
Mój własny serwer www



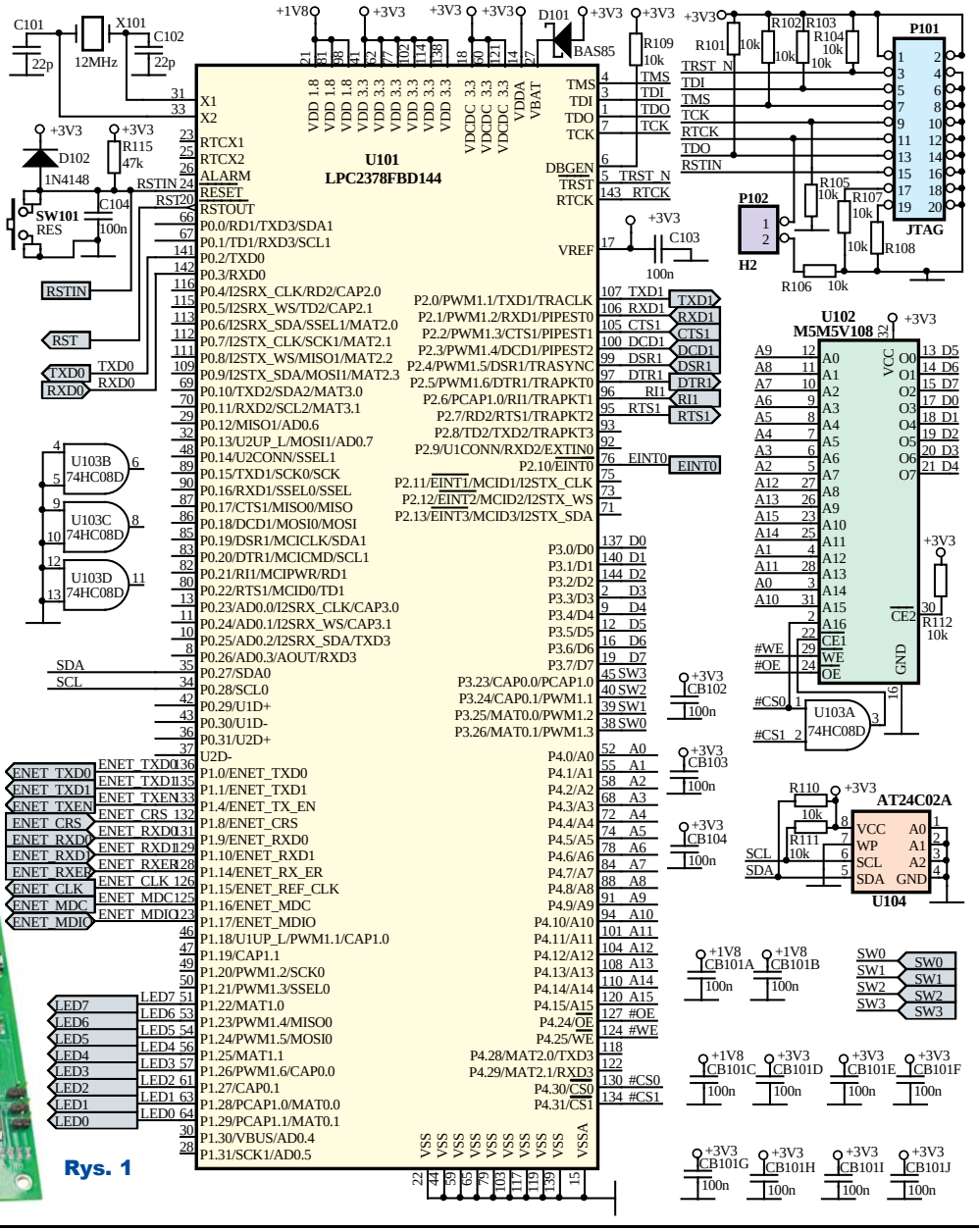
W grudniowym numerze EdW zamieszczony był ogólny opis systemu automatyki domowej, pracującego w moim domu. System jest rozbudowany, dlatego zgodnie z zapowiedzią poszczególne moduły zostaną opisane w kolejnych numerach EdW. Nie przejmuj się więc, jeśli nie wszystko zrozumiesz albo gdy nasuną Ci się pytania czy wątpliwości. Z czasem obraz się rozjaśni, a gdybyś pod koniec cyklu nadal miał jakieś pytania – napiszesz o tym do Redakcji lub wprost do mnie.

A teraz omówimy serce systemu: **serwer stron www**. **Fotografia tytułowa** pokazuje model, zmontowany specjalnie na potrzeby tego artykułu, natomiast **fotografia 1** przedstawia wcześniej zrealizowany egzemplarz. Serwer www oparty jest na mikrokontrolerze z rodziny ARM o symbolu LPC2378. Jego wybór jest podyktowany kilkoma względami, z których najważniejszym jest wbudowany blok do obsługi sieci ethernetowej. Dodatkowo bardzo istotną cechą tego mikrokontrolera jest w miarę proste wykorzystanie i pod tym względem posługiwanie się nim nie odbiega od obsługi popularnych mikrokontrolerów z rodziny AVR.

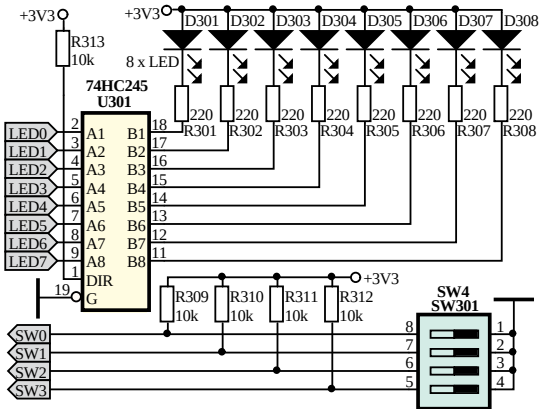
Jak widać, układ jest w sumie dość skomplikowany, ale łatwiej będzie zrozumieć jego budowę i działanie po podzieleniu



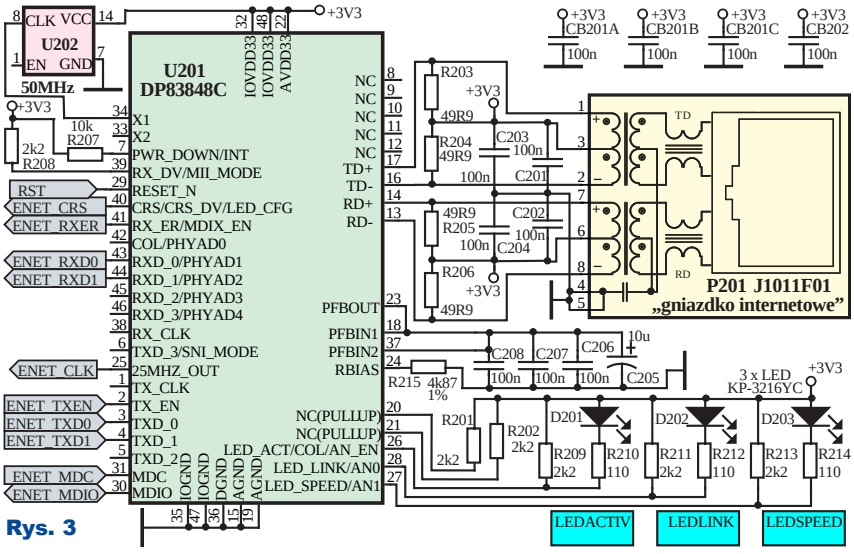
Fot. 1



Rys. 1



Rys. 2

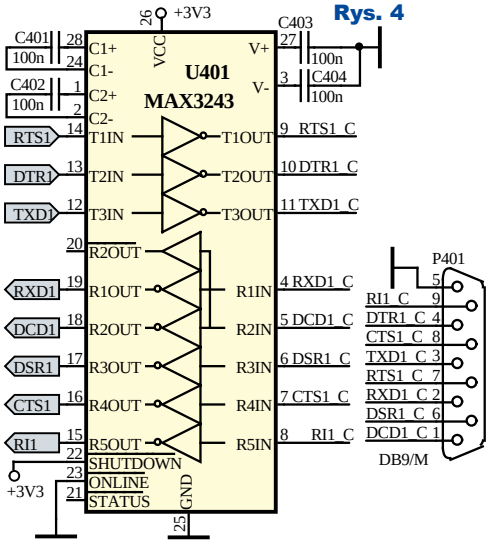


Rys. 3

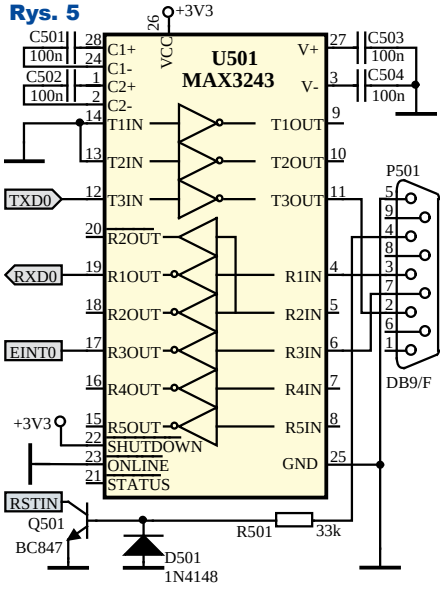
schematu na części. Schemat jednostki centralnej serwera pokazuje **rysunek 1**. **Rysunek 2** to blok kontrolek. Na **rysunku 3** widać zespół współpracy z przewodową siecią komputerową LAN z typową skrętką komputerową. Na **rysunkach 4**, **5** pokazane są dwa interfejsy RS232. **Rysunek 6** pokazuje schemat bloku zasilającego.

Wszystkie szczegóły działania urządzenia mogą wydawać się bardzo skomplikowane. Nawet sam opis konstrukcji może przestraszyć mniej zaawansowanych. Są jednak dobre wiadomości. Otóż wcale nie trzeba rozumieć wszystkich szczegółów, by taki układ wykonać (lub zakupić jako kit). Nie trzeba też być programistą. Wystarczy wgrać program (omówiony i udostępniony w Elportalu w kolejnym miesiącu). Potrzebny jest do tego program FLASHMAGIC i pełnomodemowy kabelek RS232 (więcej informacje w dalszej części).

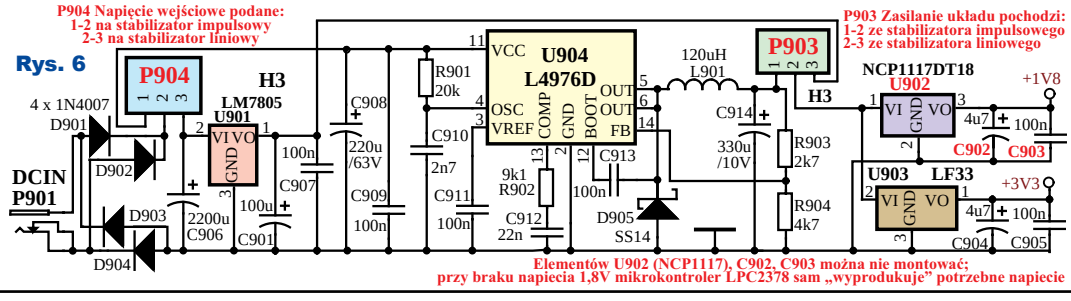
Omawianie układu zacznijmy od rysunku 1. Zastosowany mikrokontroler LPC2378 (U101) ma typową aplikację. Taktowany jest sygnałem zegarowym, uzyskiwanym w wewnętrznym generatorze współpracującym z rezonatorem kwarcowym 12MHz. Jest to typowe w mikrokontrolerach rozwiązanie, gdzie obwód rezonansowy składa się z rezonatora (X101) oraz dwóch kondensatorów C101 oraz C102. W rzeczywistości częstotliwość taktowania mikrokontrolera jest większa, gdyż istnieje możliwość programowego włączenia wbudowanego układu PLL, którego zadaniem jest kilkukrotne zwiększenie częstotliwości sygnału zegarowego. Układ LPC2378 ma dodatkowy generator do odmierzania czasu rzeczywistego (RTC), ale w tym rozwiązaniu nie jest on używany. Elementy D102, R115 oraz C104 tworzą obwód, którego zadaniem jest wygenerowanie sygnału zerowania dla mikrokontrolera w momencie włączenia napięcia zasilającego. Dodany mikroprzełącznik SW101 pozwala na asynchroniczne wyzerowanie procesora w dowolnym momencie. Dodatkowo obwód ten (sygnał RSTIN) jest przyłączony do interfejsu RS232 związanego z portem UART0, którego schemat pokazuje rysunek 5. Wyjaśnienie dla dociekliwych i zaawansowanych: połączenie to jest niezbędne podczas operacji umieszczenia kodu programu w wewnętrznej pamięci FLASH. Taka możliwość jest określana jako programowanie w trybie IAP (programowanie w układzie aplikacyjnym). W tym trybie używany jest dodatkowo sygnał zewnętrznego przerwania EINT0, który również poprzez interfejs RS232 jest wyprowadzony na złącze szeregowe. Oba te sygnały (EINT0 oraz RSTIN) są sterowane przez wyjściowe (z punktu widzenia komputera PC) linie modemowe (RTS i DTR). Te linie modemowe, obok transmitowanych danych TXD oraz RXD, to dodatkowe sygnały sterujące. Wykorzystywane są one przez wspomniany już wcześniej program FLASHMAGIC w operacji programowania (umieszczenia kodu programu w wewnętrznej pamięci FLASH). Program ten, dostępny w internecie, odpowiednio steruje liniami modemowymi, które poprzez układ interfejsu RS232 sterują liniami mikrokontrolera (sygnał RSTIN przyłączony do wejścia RESET, pin 24 oraz sygnał EINT0 przyłączony do wejścia EINT0 pin 76) i wprowadzają



Rys. 4



Rys. 5



Rys. 6

mikrokontroler do trybu IAP. Kod programu, jako zawartość pamięci FLASH, jest przesyłany w dalszej kolejności jako dane szeregowe (sygnał TXD0 oraz RXD0). Wymienione sygnały tworzą kompletny interfejs niezbędny do zaprogramowania pamięci FLASH zawartej w strukturze mikrokontrolera. Tu potwierdzam sugestię nasuwającą się uważnym Czytelnikom, do zaprogramowania mikrokontrolera **nie jest potrzebny żaden programator** lub jakiekolwiek inne urządzenie. Wymagany jest jedynie pełnomodemowy przewód do połączenia komputera z programowanym mikrokontrolerem ARM. Określenie „pełnomodemowy" oznacza, że zawiera on połączenie wszystkich styków złącza DB9 i jest to typowy element wyposażenia modemów telekomunikacyjnych. Jest zrozumiałe, że nie każdy sympatyk elektroniki musi mieć na wyposażeniu modem, by móc pożyczyć z zestawu odpowiedni przewód, ale przewód taki można wykonać samodzielnie. Niezbędne materiały to dwa złącza typu DB9 (jedno męskie i jedno żeńskie) oraz kawałek wielożyłowego przewodu. W rzeczywistości, tę „pełnomodemowość" można okroić do przewodu 5-żyłowego i połączyć (bez przeplotów) jedynie te styki w złączach, które są używane przy programowaniu (jak na rysunku 5 styki o numerach 2, 3, 4, 5 i 7).

W fazie tworzenia oprogramowania przydatne jest złącze do interfejsu JTAG (P101 wraz z kilkoma rezystorami wymuszającymi właściwy stan pasywny na liniach JTAG w sytuacji, gdy nie jest on używany). Złącze to używa dużej liczby sygnałów, toteż do docelowego programowania mikrokontrolera ARM wykorzystywany jest tryb IAP. Nie można wykluczyć modyfikacji czy rozbudowy funkcjonalności programu realizowanego przez mikrokontroler już w eksploatowanym urządzeniu, toteż wyprowadzenie złącza kanału szeregowego wykorzystywanego w trybie IAP na panel przedni obudowy zwalnia nas z konieczności każdorazowego demontażu serwera lub otwierania jego obudowy. Zastosowany mikrokontroler ma możliwość dołączenia zewnętrznej równoległej pamięci RAM (U102). Do jej obsługi dedykowane są określone wyprowadzenia portu P3 i P4. W oparciu o sygnały używane przez mikrokontroler w chwili dostępu do zewnętrznej pamięci RAM wygenerowany jest dla niej sygnał wyboru w układzie kombinacyjnym, opartym na bramkach logicznych (U103). W rozwiązaniu tym posługiwanie się zewnętrzną pamięcią RAM niczym nie różni się od używania wbudowanej pamięci operacyjnej. Obok pamięci statycznej, w układzie przewidziana jest pamięć nieulotna (U104 jako AT24C02). Pamięć ta służy do przechowywania danych konfiguracyjnych. W przypadku popularnych mikrokontrolerów z rodziny AVR tego typu pamięć jest zintegrowana razem z mikrokontrolerem. Niestety w przypadku procesorów z rodziny LPC2000 tego typu pamięć nie występuje, toteż została dobudowana jako dodatkowy element. Zastosowany układ (U104, AT24C02) to popularna pamięć EEPROM z interfejsem I2C, stąd rezystory podciągające R110 oraz R111.

Kolejnym elementem wchodzącym w skład serwera www jest zespół do sygnalizacji stanów pracy procesora za pomocą diod LED. Ze względu na ograniczoną wydajność prądową wyprowadzeń portów mikrokontrolera diody LED są sterowane za pośrednictwem układu cyfrowego 74HC245 (U301, rysunek 2), który z kolei jest przyłączony do odpowiednich wyprowadzeń portu P1 mikrokontrolera LPC2378 (rysunek 1). Znaczenie poszczególnych diod wynika wyłącznie z oprogramowania, toteż ich funkcje będą opisane w kolejnej części. Dodatkowo do mikrokontrolera przyłączony jest zespół czterobitowego przełącznika (SW301) z rezystorami podciągającymi R309 do R312. Każda z linii niezależnie może być poprzez przełącznik (SW301) przyłączona do masy, co przy odczytach przez mikrokontroler daje wartość 0. Przy manipulowaniu poszczególnymi przełącznikami istnieje możliwość przekazania programowi serwera www określonych informacji, czy wręcz poleceń. Rozpoczynając pracę program mikrokontrolera ARM może przejść do realizacji swojej podstawowej funkcji lub zainicjować jakąś specjalną akcję jak przykładowo konfiguracja. Znaczenie poszczególnych przełączników wynika z oprogramowania i szerzej będzie omówione w kolejnej części..

Oprócz wielu typowych w mikrokontrolerach peryferii (jak porty, liczniki czy zegary), LPC2378 integruje w swojej strukturze zespół EMAC (odpowiednik komputerowej karty sieciowej) i wymaga jedynie dołączenia układu określanego jako PHY, realizującego dostęp do fizycznego łącza sieci komputerowej. Idea stosowania układów PHY sprowadza się do umożliwienia wykorzystania infrastruktury elektrycznej (połączenia kablowe) lub optycznej (połączenia światłowodowe) bez znaczących zmian w układzie. W moim rozwiązaniu komunikacja bazuje na przewodach miedzianych, gdyż dom jest okablowany skrętką ethernetową. Możliwość dostępu do serwera poprzez sieć Wi-Fi wynika wyłącznie z funkcjonalności routera. Sam serwer komunikuje się z otoczeniem sieciowym wyłącznie poprzez połączenie kablowe. Układ PHY, z punktu widzenia mikrokontrolera jako jego element zewnętrzny, komunikuje się z zespołem EMAC za pośrednictwem „znormalizowanego złącza". Dla zaawansowanych i dociekliwych: najczęściej ten styk mikrokontrolera z układem PHY występuje w dwóch wariantach określanych jako **MII** (ang. Media Independent Interface – interfejs niezależny od zastosowanego medium komunikacyjnego) lub **RMII** (ang. Reduced Media Independent Interface) jako zredukowany wariant MII (użyty w serwerze układ PHY umożliwia wariant interfejsu określany jako **SNI**, który ma zastosowanie jedynie w sieci pracującej z prędkością 10MB, więc jest pominięty w opisie). W sytuacji, gdy ktoś chciałby zastąpić miedziane kable przykładowo interfejsem światłowodowym, to (poza oczywistą zmianą układu PHY na taki, który jest dopasowany do charakteru medium) nie wymaga to ingerencji w interfejs komunikacyjny pomiędzy układem PHY a zespołem EMAC. Różnica pomiędzy rozwiązaniem MII a RMII sprowadza się do liczby połączeń występujących na styku. Oprócz niezbędnych sygnałów taktujących i synchronizujących przesyłanie danych, w styku tym występują linie przesyłające transmitowane dane. W przypadku wariantu MII jednocześnie (jednym taktem zegarowym) przesyłane są cztery bity danych. W wariancie RMII dane są przesyłane po dwa bity (dwa bity do nadawania oraz dwa bity do odbierania, co daje cztery linie magistrali danych pomiędzy EMAC a układem PHY, a w wariancie MII konieczne jest użycie ośmiu linii).

Schemat ideowy przyłącza do sieci Ethernet pokazuje rysunek 3. Układem PHY jest popularny układ scalony o symbolu DP83848 (U201). Umożliwia on wariant MII, RMII i SNI, jednak ze względu na zastosowany mikrokontroler ARM, który oferuje jedynie wariant RMII, aplikacja układu PHY jest dostosowana do tego rozwiązania. Implikuje to użycie dwóch linii do przesyłania danych (sygnały ENET_TXD0 i ENET_TXD1 do nadawania oraz ENET_RXD0 i ENET_RXD1 do odbierania). Konsekwencją stosowania styku RMII jest również częstotliwość taktująca. By uzyskać prędkość transmisji sieci wynoszącą 100MB przy dwubitowej magistrali danych, wymagana częstotliwość generatora taktującego (U202) wynosi 50MHz. W układzie PHY wariant interfejsu może być wstępnie określony poprzez podanie odpowiedniego stanu na wyprowadzenie 39 układu DP83848. Gdyby to wejście pozostało nieprzyłączone (ma ono wbudowany rezystor do masy) lub

przyłączone zostało do masy, zostałby włączony wariant MII. Wtedy wyprowadzenia o numerach 3, 4, 5 i 6 tworzą magistralę danych. W przypadku przyłączenia rezystora R208 do napięcia zasilającego istotny jest dodatkowo stan na wyprowadzeniu 6 tego układu. Skoro nie jest to wariant MII (ze względu na sposób przyłączenia R208), to wyprowadzenie 6 nie tworzy magistrali danych i może być użyte do sprzętowego określenia wariantu interfejsu (pozostawienie go jako nieprzyłączonego oznacza skonfigurowanie interfejsu do wariantu RMII). Wybrany wariant (MII lub RMII) może być również zmieniany poprzez zapis do odpowiedniego rejestru układu.

Wszystkie operacje transmisyjne do i z sieci realizowane przez układ DP83848 są synchronizowane sygnałem zegarowym pochodzącym z generatora U202. Dotyczy to również synchronizacji wymiany danych z zespołem EMAC, toteż układ PHY niejako udostępnia własny sygnał zegarowy zespołowi EMAC (sygnał ENET_CLK). Sygnał ENET_TXEN aktywuje układ DP83848 do wysłania danych (stan wysoki na tej linii oznacza obecność ważnych danych). W operacjach odbierania danych z sieci Ethernet przez układ DP83848 wykorzystywane są dwie linie sygnalizacyjne. Sygnał ENET_CRS informuje o ważności odebranych danych (znajdujących się na dwubitowej magistrali danych) oraz ENET_RXER, który sygnalizuje wykrycie błędu transmisji. Sam układ interfejsu sieciowego jest konfigurowalny (ma wewnętrzne rejestry, których zawartość wpływa na jego działanie), toteż musi istnieć możliwość zapisu przez mikrokontroler odpowiednich danych do układu PHY. Jest to zrealizowane jako przesyłanie szeregowe w oparciu o dwa sygnały ENET_MDC (jako sygnał taktujący) oraz ENET_MDIO (jako sygnał danych). Tą samą drogą zrealizowany jest odczyt rejestrów statusowych z układu PHY. Tutaj mogę też uspokoić Czytelników dostrzegających konieczność złożonej programowej obsługi układu PHY. Nie, zespół EMAC w mikrokontrolerze realizuje to wszystko samodzielnie. Przykładowo zapis do określonego rejestru znajdującego się w układzie DP83848, a który jest zrealizowany jako komunikacja szeregowa bazująca na sygnałach ENET_MDC i ENET_MDIO, sprowadza się do zapisu odpowiednich kodów do właściwych rejestrów w zespole EMAC, który nakazaną czynność zrealizuje dalej samodzielnie. Podobnie przesłanie ramki ethernetowej jako bloku danych wymaga manipulacji rejestrami zespołu EMAC, który zlecone czynności wykona autonomicznie.

Jeśli nie rozumiesz szczegółów – nie przejmuj się – układ będzie działał niezależnie od stopnia Twojego zaawansowania. Więcej informacji na ten temat będzie z kolejnych częściach.

Istotnym elementem w obsłudze warstwy fizycznej jest zastosowanie odpowiedniego transformatorka separującego. Nie występuje on jawnie na schemacie, gdyż jest zintegrowany jako jeden element z 8-pinowym gniazdem (P201) do sieci Ethernet, a sam transformatorek jest symbolicznie narysowany wewnątrz złącza. Pomiędzy układem PHY a transformatorkiem występuje sieć rezystorów oraz kondensatorów, która jest zgodna z wymaganiami opisanymi w dokumentacji układu DP83848. Istotnym elementem jest rezystor (R215) przyłączony do wyprowadzenia RBIAS (pin 24). Wartość rezystancji jest precyzyjnie określona (4,87kΩ). Rezystor musi pochodzić z szeregu o tolerancji 1% lub lepszej. Zbyt duża odchyłka wartości rezystancji uniemożliwia komunikację. Zamiast tego rezystora można zastosować połączenie szeregowe dwóch rezystorów z szeregu o gorszej tolerancji (przykładowo 4,7kΩ i 170Ω), ale wtedy staje się konieczne wyselekcjonowanie rezystorów omomierzem. Również bardzo istotnym elementem jest przyłączenie kondensatorów C205...C208. Znaczenie ma nawet kształt ścieżek łączących wymienione elementy, toteż znalazło to swoje odbicie na schemacie w postaci „dziwnego" przyłączenia kondensatorów (istotne szczegóły są opisane w odpowiedniej dokumentacji producenta układu DP83848). W mojej praktyce stosowania tego układu zdarzyło się raz, że układ nie uzyskał prędkości 100MB, a jedynie 10MB i jestem przekonany, że problem dotyczył topologii ścieżek (w opisywanym serwerze www sieć Ethernet działa bez problemów z prędkością 100MB).

Zadaniem układu PHY jest również sygnalizowanie poprzez diody LED aktualnego jego stanu pracy (D201, D202 i D203). Do dyspozycji są trzy sygnały, których znaczenie jest konfigurowalne (poprzez zapis do odpowiednich rejestrów w układzie DP83848). Wybrany wariant, determinowany w oprogramowaniu, ma następujące znaczenie: włączenie diody D201 oznacza wystąpienie aktywności sieci Ethernet (nastąpiło przesłanie danych), dioda D202 sygnalizuje stan LINK (fizyczne połączenie z inną stacją gotową do wymiany danych) oraz dioda D203 sygnalizuje prędkość transmisyjną (dioda włączona oznacza prędkość 100MB, wyłączona 10MB). Tu można zauważyć, że diody LED wraz z rezystorami ograniczającymi prąd płynący przez diody są połączone równolegle z dodatkowymi rezystorami (R209, R211 i R213). Rezystory te mogą być przyłączone do +3,3V albo do masy (GND) i w momencie wystąpienia aktywnego stanu sygnału zerującego dla układu PHY, ich stan determinuje wartość początkową niektórych rejestrów konfiguracyjnych (mogą być one później zmodyfikowane przez oprogramowanie mikrokontrolera ARM). Zastosowane rezystory wymuszają wstępnie tryb pracy układu PHY określanego jako autonegocjacja. Oznacza to, że układ ten musi się dostosować do istniejącej sieci (10MB lub 100MB). W ogólnym przypadku, poprzez zapis do odpowiednich rejestrów układu PHY, można go skonfigurować do pracy jedynie w sieci Ethernet pracującej z prędkością 10MB, do pracy jedynie w sieci pracującej z prędkością 100MB albo by się dostosował do pracy w sieci do jakiej został przyłączony. Sieci 10MB powoli wychodzą z użycia i standardem stają się sieci Ethernet pracujące z prędkością 100MB.

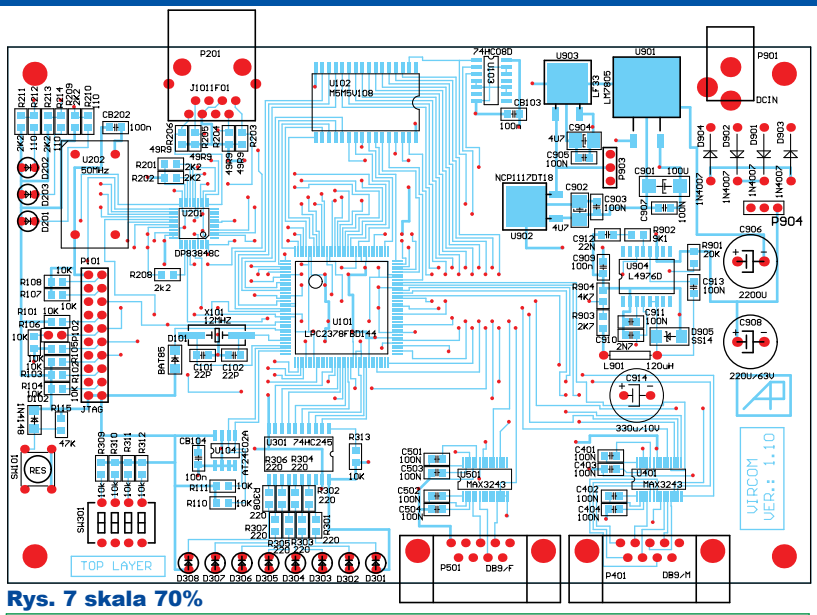
Moduł serwera www wykorzystuje dwa kanały asynchronicznej transmisji szeregowej. Jeden z nich (rysunek 4), jako zawierający wszystkie linie modemowe, jest przeznaczony do wymiany danych z przyłączonymi modułami pomiarowymi lub wykonawczymi. Zastosowany jest tam układ o symbolu MAX3243 (U401) pozwalający na transmisję danych szeregowych oraz obsługę wszystkich linii modemowych. Wyjście układu jest przyłączone do męskiego złącza DB9 (stanowi identyczne rozwiązanie funkcjonalne jakie występuje w komputerach PC). Umożliwia to przyłączenie do serwera dowolnych urządzeń, które mogą wymagać wykorzystania linii modemowych. Współpraca na tym styku z procesorem komunikacyjnym, który nie wykorzystuje linii modemowych, należy traktować jako jedną z wielu możliwości. W przypadku rezygnacji z wariantu pełnomodemowego powyższe rozwiązanie może być uproszczone poprzez zastosowanie przykładowo układu MAX3232 (odpowiednik popularnego interfejsu MAX232 przystosowanego do pracy z zasilaniem 3,3V).

Drugi interfejs szeregowy (rysunek 5) przeznaczony do programowania mikrokontrolera (jako operacji ładowania kodu do jego pamięci FLASH) jest przystosowany do współpracy z programem FLASHMAGIC (specjalizowanym programem ładującym kod do pamięci FLASH mikrokontrolerów z rodziny LPC2000). Zastosowany jest tu identyczny układ scalony MAX3243 (U501) jak w interfejsie komunikacyjnym z rysunku 4. Jak było wspomniane wyżej, program FLASHMAGIC do ładowania zawartości pamięci FLASH oprócz danych szeregowych wykorzystuje

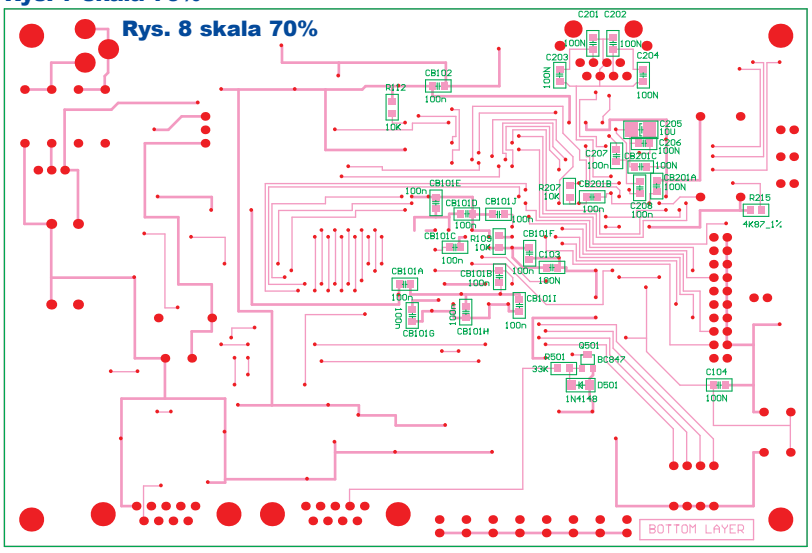
dwie linie modemowe. Dane szeregowe (nadawane oraz odbierane) są przetwarzane przez układ interfejsu w typowy sposób. Podobnie obsługiwana jest jedna z linii modemowych (sygnał EINT0). Druga linia modemowa, która ze złącza DB9 przekształca się na sygnał reset mikrokontrolera, ma odmienne rozwiązanie. Wynika to ze względu na potencjalne konflikty napięciowe w obwodzie generującym ten sygnał. Wymuszenie stanu aktywnego (zera logicznego) na tej linii może wynikać z ładowania kondensatora C104 przez rezystor R115 (rysunek 1), naciśnięcia przycisku SW101 (rysunek 1) lub wymuszenia wygenerowanego przez program FLASH-MAGIC jako odpowiednie wysterowanie linii modemowej (rysunek 5). Ponieważ układ interfejsu U501 nie ma wyjścia typu otwarty kolektor, funkcjonalność ta została odtworzona na bazie elementów dyskretnych (R501, D501, Q501). W interfejsie RS232 na liniach występują napięcia o wartościach +10...+12V lub –10...–12V. Układ interfejsu przetwarza sygnał o dodatnim napięciu na sygnał zera logicznego. Podobnie sygnały o ujemnym napięciu są konwertowane do stanu logicznej jedynki. W powyższym układzie tranzystorowym wysterowanie Q501 sygnałem o napięciu dodatnim powoduje nasycenie tranzystora, co sprawia, że na jego kolektorze wystąpi napięcie bliskie 0V. W sytuacji, gdy na linii wystąpi napięcie ujemne, tranzystor będzie zatkany a dioda D501 będzie chronić bazę tranzystora przed niewłaściwą polaryzacją. W tej sytuacji napięcie na kolektorze będzie ustalone przez elementy C101 i R115. Układ tranzystorowy również poprawnie się zachowa w sytuacji, gdy kabel RS232 będzie odłączony (tranzystor Q501 nie będzie wysterowany). Ten interfejs szeregowy zakończony jest żeńskim złączem DB9. Pozwala to na zastosowanie standardowych przewodów RS232, gdyż w trakcie ładowania kodu do pamięci FLASH moduł serwera logicznie odpowiada roli modemu podłączonego do komputera (jest urządzeniem podrzędnym). Należy pamiętać, że interfejs RS232 jest niesymetryczny (zawiera trzy linie wyjściowe oraz pięć linii wejściowych, co implikuje, że występuje strona określana jako wyposażenie komputerowe oraz strona określana jako wyposażenie modemowe). Nie wyklucza to możliwości użycia tego kanału szeregowego do funkcji spełnianej przez wcześniej opisany interfejs (przewidziany do komunikacji z modułami pomiarowymi), jednak wymaga to wykonania właściwego przewodu połączeniowego. Kabel musi być bez linii modemowych (odpowiedni stan linii modemowej spowoduje reset mikrokontrolera) oraz wymagane jest skrzyżowanie linii transmitowanych danych.

Podobnie jak w przypadku interfejsu RS232 z rysunku 4, użyty układ MAX3243 może być zastąpiony układem MAX3232.

Ostatnim elementem serwera www jest zasilacz (rysunek 6). Rozwiązanie jest zaprojektowane w dwóch wariantach: z zastosowaniem liniowego stabilizatora napięcia typu 7805 (U901) oraz stabilizatora impulsowego o symbolu L4976 (U904). Jest wybór pomiędzy prostotą rozwiązania (stabilizator liniowy) a sprawnością (impulsowy). Mając na uwadze wymóg ciągłej pracy urządzenia, zalecane jest rozwiązanie impulsowe, ale mniej doświadczeni adepci elektroniki obawiający się kłopotów z uruchomieniem impulsowego stabilizatora mają wybór w postaci aplikacji znanego i popularnego rozwiązania (w takim przypadku montaż elementów związanych z układem U904 nie jest konieczny). Do odpowiedniego skonfigurowania przeznaczone są dwa zespoły zworek (P902 oraz P903). Wstępnie napięcie zasilające urządzenie jest obniżone do wartości +5V (wszystkie moduły automatyki są zasilane przez zasilacz +12V). Finalne napięcie zasilające o wartości 3,3V jest wytwarzane przez układ U903. Sam mikrokontroler LPC2378 wymaga dwóch napięć zasilających (drugie ma wartość 1,8V), jednak w przypadku braku tego napięcia (1,8V) mikrokontroler może je sam wytworzyć.



Rys. 7 skala 70%



Rys. 8 skala 70%

Montaż i uruchomienie

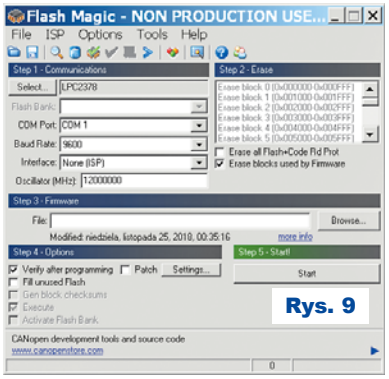
Do całego tego układu została zaprojektowana jedna płytka drukowana. Mozaikę ścieżek i rozkład elementów pokazuje **rysunek 7** dla strony TOP oraz **rysunek 8** dla strony BOTTOM. Na tych rysunkach symbolicznie występują diody LED, lecz w rzeczywistości są to GOLDPIN-owe złącza, do których przyłączone są diody fizycznie zamontowane w przednim panelu obudowy serwera.

Budowa urządzenia nie jest bardzo trudna – wymaga jednak staranności. Większość komponentów jest przeznaczona do montażu powierzchniowego, a kluczowe układy scalone mają gęsty raster wyprowadzeń. Dla początkujących majsterkowiczów może to stanowić wyzwanie, jednak przy odrobinie cierpliwości wszystkie problemy są do pokonania.

Przed pierwszym uruchomieniem należy sprawdzić, czy stabilizatory wytwarzają właściwe napięcia zasilające: usunąć zworkę ze złącza P903

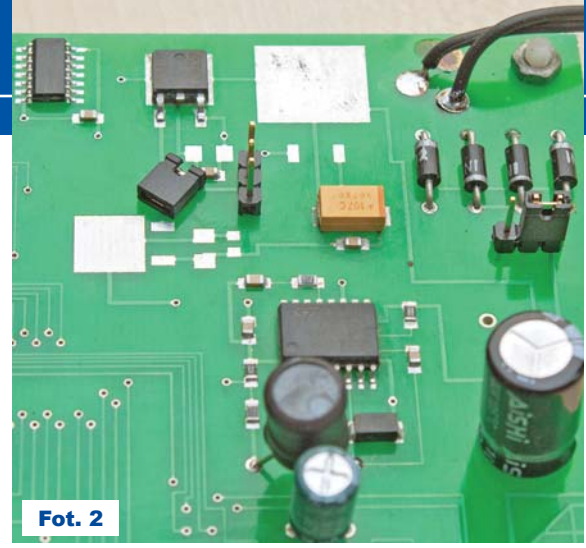
(rysunek 6, **fotografia 2**), by zabezpieczyć cały serwer przed uszkodzeniem w przypadku, gdyby w wyniku pomyłki przy montażu stabilizator nie wytworzył właściwego napięcia zasilającego i zmierzyć napięcie na wyjściu stabilizatora impulsowego (bądź liniowego). Wartość napięcia powinna być w granicach 5...6V. Przy poprawnych wartościach napięcia można zasilić układ, zakładając odpowiednio zworkę na złączu P903. Po przyłączeniu modułu do sieci (switcha 100MB), nawet bez załadowanego programu do pamięci FLASH mikrokontrolera, powinny zaświecić się dwie diody LED sygnalizujące prędkość transmisji oraz dioda sygnalizująca stan LINK (**fotografia 3**), co pkazuje, że układ PHY jest wstępnie skonfigurowany przez odpowiednie rezystory do trybu autonegocjacji prędkości komunikacyjnej oraz został fizycznie połączony z innym urządzeniem sieciowym.

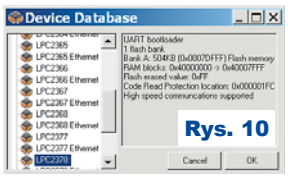
Mikrokontroler trzeba zaprogramować. Ja używam wspomnianego wyżej programu **FLASHMAGIC**. Program ten możne bezpłatnie pobrać ze strony *http://www.flashmagictool.com/download.html&d=flashmagic*. Aktualnie jest dostępna wersja 11.20. Instalacja przebiega typowo i nie powinna sprawić jakiegokolwiek problemu. Po uruchomieniu (**rysunek 9**) należy pełnomodemowym przewodem połączyć port COM komputera ze złączem P501 serwera. Uwaga! oryginały rysunków-zrzutów 9...15 dostępne są w Elportalu.



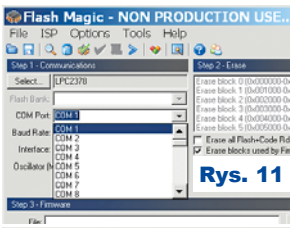
Rys. 9

Program ten jest wstępnie właściwie skonfigurowany (w zakresie parametrów sterowania liniami modemowymi) i nie należy tego modyfikować. Jedyne elementy do modyfikacji dotyczą przede wszystkim wyboru portu komunikacyjnego, wskazania pliku w formacie Intel-hex zawierającego kod programu do zapisania w pamięci FLASH mikrokontrolera. By zaprogramować mikrokontroler, należy klikając na przycisk *Select*, co spowoduje wyświetlenie listy obsługiwanych mikrokontrolerów, wybrać model użytego mikrokontrolera (**rysunek 10**). Rzecz oczywista, należy wybrać mikrokontroler LPC2378 (w przypadku wybrania niewłaściwego, oprogramowanie zwróci uwagę, że wybrana pozycja nie odpowiada użytemu w projekcie i należy dokonać korekty).


Fot. 2



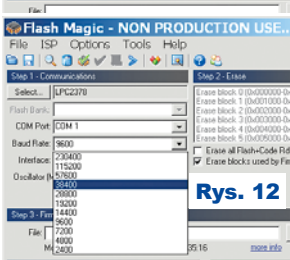
Rys. 10



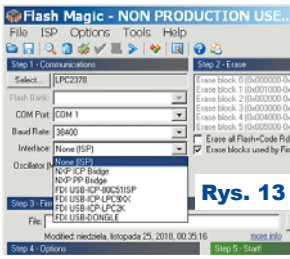
Rys. 11

W kolejnym kroku program wymaga informacji określającej port COM użytego do komunikacji z procesorem. W moim przypadku jest to port COM1 i taką wybieram z zaproponowanej listy (**rysunek 11**).

W pozycji *Baud Rate* określam prędkość transmisji poprzez wybór jednej z zaproponowanej wartości. Naturalne jest, że wybór większej prędkości implikuje krótszy czas operacji, jednak nie jest zalecane „śrubowanie" parametrów prędkościowych, gdyż rośnie prawdopodobieństwo przekłamań. Z dotychczasowej praktyki wnioskuję, że prędkość 38 400 bps jest optymalna (**rysunek 12**). Pozycja *Interface* jest przeznaczona do określenia stosowanego interfejsu. W tym konkretnym przypadku nie jest stosowany żaden, więc pole to musi zawierać wartość *None (ISP)*, **rysunek 13.**



Rys. 12



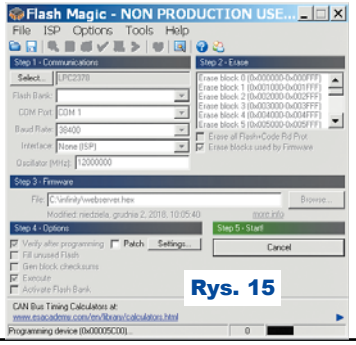
Rys. 13

W okienku *Oscillator* wpisuję częstotliwość rezonatora kwarcowego użytego do taktowania mikrokontrolera. W tym projekcie mikrokontroler jest taktowany sygnałem zegarowym o częstotliwości 12MHz, toteż wpisuję liczbę 12 000 000. W pozycji *File* jest możliwość wskazania pliku zawierającego kod programu do załadowania do pamięci FLASH, toteż po kliknięciu przycisku *Browse* powstaje możliwość wskazania właściwego pliku. Należy wyklikać właściwą kartotekę i wybrać wymagany plik w formacie hex (**rysunek 14**).

Rys. 14

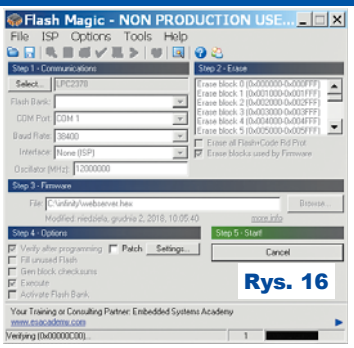
Warto mieć zaznaczone „ptaszki" w pozycjach *Erase block used by firmware*, co oznacza, że pamięć FLASH w mikrokontrolerze zostanie skasowana przed jej zaprogramowaniem w niezbędnym obszarze oraz *Verify after programming*, co implikuje, że po załadowaniu kodu programu do pamięci program FLASHMAGIC dokona weryfikacji poprawności operacji (odczyta zawartość pamięci FLASH i porówna z oryginałem). Proces programowania uruchamiany jest w wyniku kliknięcia na przycisk *Start*. W pierwszej kolejności zostanie skasowana pamięć FLASH i po jej zakończeniu nastąpi przesłanie kodu programu do mikrokontrolera (**rysunek 15**). Na dole okienka pojawi się komunikat *Programming device* i pasek postępu operacji.


Fot. 3



Rys. 15

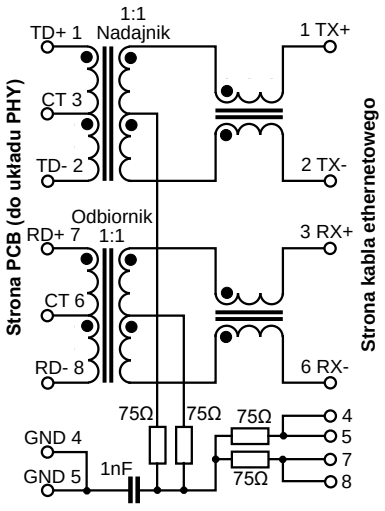
Po przesłaniu całej zawartości pamięci FLASH program FLASHMAGIC przejdzie do weryfikacji. Na dole



Rys. 16

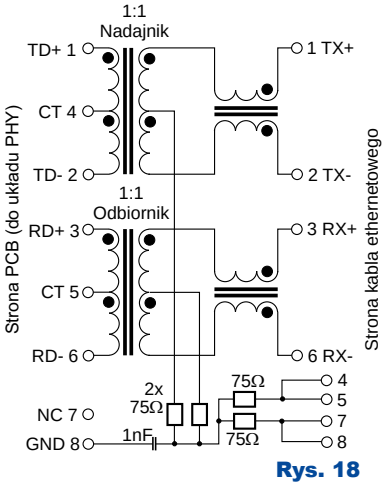
okienka pojawi się napis *Verifying* wraz z paskiem postępu (**rysunek 16**). Po jej zakończeniu mikrokontroler jest zaprogramowany i gotów do działania z nowym programem.

Na zakończenie chcę poruszyć tematykę ewentualnych zamienników do użytych komponentów elektronicznych. W przypadku pamięci RAM nie ma bezwzględnej konieczności zastosowania układu wymienionego na liście użytych elementów. W trakcie opracowywania schematu dokładnie taki „znajdował się w mojej szufladzie" i z tego powodu trafił do schematu. Wspomniany układ jest statyczną pamięcią RAM o pojemności 128k*8 i w rzeczywistości może być zastąpiony dowolnym innym kompatybilnym układem (przykładowo: CY61228, AS6C1008). Pozornie gorzej przedstawia się zamiana gniazda do sieci Ethernet z zintegrowanym wewnątrz transformatorkiem impulsowym. Dystrybutorzy elementów elektronicznych mają w swoich ofertach wiele elementów, które mogą być zastosowane, jednak istnieje tu pewien problem, mianowicie występuje różny rozkład wyprowadzeń. Nawet w sytuacji, gdy złącze mechanicznie pasuje, musi zawierać identyczną funkcjonalność poszczególnych wyprowadzeń. W konstrukcji zastosowane jest złącze o symbolu J1011F01 i takie musi zostać użyte. W swojej dotychczasowej praktyce z powodzeniem wypróbowałem kilka innych, jak przykładowo J0011D21 lub HY91116C, jednak ich użycie wymaga przeprojektowania mozaiki ścieżek na płytce drukowanej. W przypadku złącza J1011F01 tor nadajnika związany jest z wyprowadzeniami o numerach 1, 3, 2, a tor odbiornika z wyprowadzeniami 7, 6 i 8 (**rysunek 17**).



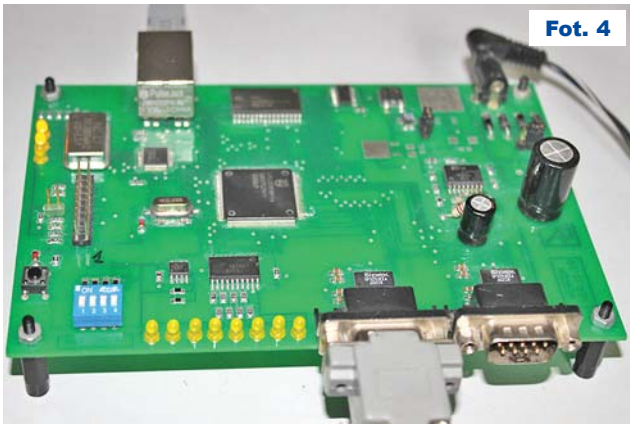
Rys. 17

Inne możliwe złącze (J0011D21, tego samego producenta) ma tor nadajnika na wyprowadzeniach 1, 4, 2 oraz tor odbiornika na wyprowadzeniach 3, 5, 6 (**rysunek 18**).



Rys. 18

Nie jest możliwe wymienienie wszystkich zamienników. W gruncie rzeczy złącze do sieci Ethernet z punktu widzenia układu PHY jest zestawem dwóch maleńkich transformatorków impulsowych. Jednym z parametrów każdego transformatora jest jego przekładnia napięciowa. W dokumentacji do układu DP83848 znajduje się informacja, że wymagana przekładnia wynosi 1:1 zarówno dla toru nadajnika, jak i odbiornika. Poza oczywistym wymogiem, by transformatorek w złączu był przystosowany do transmisji z prędkością 10MB i 100MB, przekładnia jest praktycznie jedynym kryterium, jakie muszą spełniać zintegrowane ze złączem transformatorki impulsowe.


Fot. 4

Fotografia 4 pokazuje wcześniejszy model podczas testów. W Elportalu wśród materiałów dodatkowych do tego artykułu można znaleźć dokumentację płytki, zrzuty ekranu oraz rysunki i fotografie.

W następnym miesiącu kolejna część będzie poświęcona wyjaśnieniu podstawowych pojęć związanych z siecią oraz będzie opisana implementacja programu serwera www (łącznie z udostępnieniem kodu programu).

Andrzej Pawluczuk
apawluczuk@vp.pl

Wykaz elementów

R101–R112,R207,R309–R313	10kΩ(0805)
R115	47kΩ (0805)
R201,R202,R208,R209,R211,R213	2,2kΩ (0805)
R203,R204,R205,R206	49,9Ω (0805)
R210,R212,R214	110Ω (0805)
R215	4,87kΩ (tolerancja 1%, 0805)
R301-R308	220Ω (0805)
R501	33kΩ (0805)
R901	20kΩ (0805)
R902	9,1kΩ (0805)
R903	2,7kΩ (0805)
R904	4,7kΩ (0805)
C101,C102	22pF (0805)
C103,C104,C201-C204,C206-C208,C401-C404, C501-C504,C903 (opcja),C905,C907,C909,C911, C913,CB101A-J,CB102-CB104,CB201A-CB201C, CB202	100nF (0805)
C205	10µF
C901	100µF
C902 (opcja),C904	4,7µF
C906	2200µF
C908	220µF
C910	2,7nF (0805)
C912	22nF (0805)
C914	330µF (niskoimpedancyjny)
U101	LPC2378FBD144 (LQFP144)
U102	M5M5V108 (SO32)
U103	74HC08D (SO14)
U104	AT24C02D (SO8)
U201	DP83848CVV (LQFP48)
U202	generator 50MHz +3,3V
U301	HC74C245D (SO20)
U401,U501	SP3243EEA (SSOP28)
U901 (opcja)	LM7805CD2T (D2PAK)
U902 (opcja)	NCP1117DT18 (DPAK)
U904	L4976D (SOL16)
U903	LF33CDT (DPAK)
D101	BAS85 (MINIMELF)
D102,D501	1N4148 (MINIMELF)
D201-D203,D301-D308	diody LED
D901,D902,D903,D904	1N4007
D905	SS14 (DO214)
Q501	BC847 (SOT23)
X101	12MHz
L901	120µH
P201	J1011F01
SW101	mikroprzycisk
SW301	DIP switch czterobitowy
P401	złącze D-SUB 9 pin męskie, kątowe
P501	złącze D-SUB 9 pin żeńskie, kątowe
P101	GOLDPIN dwurzędowe 10x2
P102	GOLDPIN 2 pin + jumperek
P903,P904	GOLDPIN 3 pin + jumperek

Komplet podzespołów z płytką jest dostępny w Sklepie AVT jako zestaw AVT3232

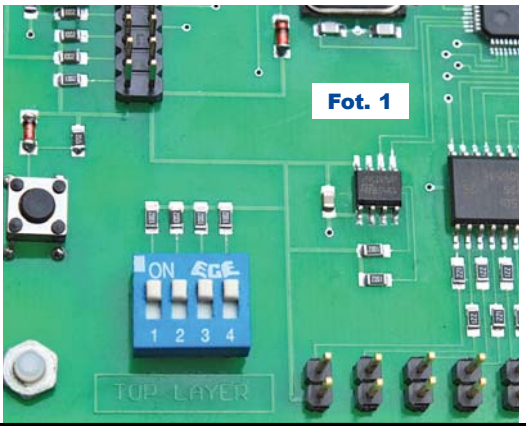
Infinity – system automatyki domowej
część 1
Oprogramowanie serwera www

Zgodnie z zapowiedzią, w tym miesiącu omówimy konfigurację i (dla chętnych) zostanie przedstawiony program działający w mikrokontrolerze LPC2378. Wszelkie odwołania do sprzętowej części serwera dotyczą opisu **serwera www** zamieszczonego w poprzednim numerze czasopisma.

Masz ogromny powód do satysfakcji, bo zrobiłeś (zrobisz albo planujesz zrobić) serwer www, czyli „pełnoprawne urządzenie sieciowe". Właśnie dlatego działanie Twojego serwera www „w pojedynkę" nie ma absolutnie żadnego sensu. Musi on być (podkreślmy z dumą: pełnoprawnym, a nie żadnym okrojonym czy zubożonym) składnikiem sieci komputerowej. Lokalnej sieci komputerowej, która może być i zapewne jest/będzie podłączona do Internetu.

Twój serwer www po zaprogramowaniu procesora jest gotowy, aż rwie się do pracy, ale w pojedynkę nic nie zdziała i musi wykorzystać możliwości, jakie daje sieć komputerowa. Aby to było możliwe, trzeba go skonfigurować, czyli sprawić, by on widział sieć, w której będzie pracował i by ta sieć też go widziała.

Z punktu widzenia systemu automatyki domowej moduł serwera www wydaje się (i słusznie) absolutnie najważniejszym składnikiem tego systemu. Owszem, ale jest dołączony do (domowej, lokalnej) sieci komputerowej, a tam jest większa demokracja. Patrząc z punktu widzenia sieci komputerowych, nasz serwer www jest (wprawdzie pełnoprawnym), ale wcale nie najważniejszym składnikiem. Jest jednym z wielu urządzeń sieciowych i w tej sieci to nie on narzuca swoje wymagania, tylko musi się dostosować do tej sieci (do jej parametrów/ustawień). Nie wgłębiając się w informatyczne szczegóły, powiemy, że do konfiguracji naszego serwera www z punku widzenia „internetowego" potrzebne są: adres MAC, adres IP, maska podsieci oraz adres IP bramy domyślnej. Ja w programie umieściłem ustawienia domyślne (które czasami nazywam „fabrycznymi", ponieważ pochodzą z fabryki, fabryki w pracowni w moim domu). Wartości tych parametrów (domyślnych) są tak dobrane, by pasowały do sieci w moim domu. Jestem przekonany, że będą odpowiednie także dla 99% Czytelników, gdyż są powszechnie używane na całym świecie. Trudno mi powiedzieć, skąd się wzięła „moda" na adresy IP postaci 192.168.0.xx. Takiej adresacji odpowiada, wręcz jako transakcja wiązana, maska podsieci 255.255.255.0, gdyż ta wartość maski zawsze wyselekcjonuje właściwie numer sieci bez względu na wartość xx (w adresie IP). O znaczeniu adresów MAC, IP, maski podsieci więcej piszę w dalszej części artykułu, więc jeżeli w tej chwili jest to trochę tajemnicze, to nie przejmuj się tym.



Fot. 1

Najprawdopodobniej możesz wykorzystać „fabryczne", domyślne wartości parametrów sieciowych. Wtedy żadna dodatkowa konfiguracja serwera i analiza nie jest konieczna. Możesz ustawić przełączniki według **fotografii 1**, dołączyć swój serwerek www do sieci komputerowej, cieszyć się, że jest w tej sieci widoczny i... czekać na następne artykuły w EdW, które pokażą, jak „z drugiej strony serwera www" dołączać kolejne moduły systemu automatyki domowej.

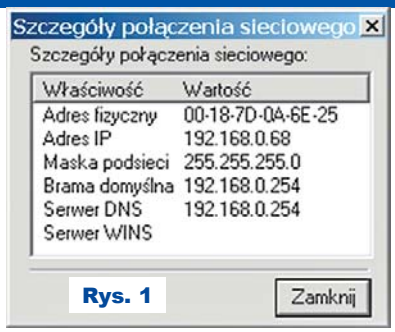
Koniec. Kropka. Amen.

W zasadzie dalej nie musisz czytać. Ale... może jednak warto wspiąć się na wyżyny i sięgnąć gwiazd, ale do tego niezbędny jest pewien wysiłek, który prowadzi do wiedzy. Można, rzecz jasna, studiować odpowiednie książki i zasoby Internetu – wysiłek ogromny. Tę wyboistą drogę mam za sobą i wiem, co sprawiło mi najwięcej problemów, toteż postaram się przedstawić ważne informacje w sposób możliwie prosty i (mam nadzieję) zrozumiały.

Może się jednak zdarzyć, że sieciowe parametry domyślne nie będą Ci odpowiadać, więc będą wymagały korekty. Na przykład jeżeli masz zamiar, drogi Czytelniku, wykorzystać więcej niż jeden egzemplarz serwera w tej samej sieci lokalnej, to musisz już zadbać, by każdy egzemplarz miał unikalny adres MAC i adres IP. W celu ewentualnego ustalenia właściwych parametrów konfiguracji sieci możesz poprosić o pomoc informatyka, znaczenie tych parametrów powinno być dla niego oczywiste. Jednak zanim to zrobisz, zachęcam Cię do samodzielnej próby konfiguracji. Jak to zrobić, opowiem za chwilę.

Rozumiem, że niektóre pojęcia na razie graniczą z czarną magią (chociaż jeżeli poznasz przeciwnika, to okaże się, że wcale tak nie jest). Jednak dla Twojego dobrego samopoczucia króciuteńkie wyjaśnienie.

Adres MAC to w sumie sprzętowy numer-adres twojej karty sieciowej (zespołu EMAC). Nie jest to informacja w jakikolwiek sposób zawarta w mikrokontrolerze czy układzie PHY w trwały sposób. Konfigurując zespół EMAC do pracy, program działający w LPC2378 nadaje mu jakiś numer. Ten numer jest zaszyty w kodzie programu mikrokontrolera na stałe i przy pierwszym uruchomieniu zostaje zapisany do pamięci EEPROM. Każde kolejne uruchomienie posiłkuje się tym numerem zapisanym w pamięci nieulotnej. Jeżeli zostanie on zmieniony, a program serwera www daje taką możliwość, to będzie obowiązywał nowy. Wracając do wartości zaszytej w programie, można zapytać: dlaczego ma taką wartość? Odpowiedź jest banalnie prosta: taki sobie „wylosowałem". Poza jednym wyjątkiem, jakim są same jedynki binarnie, czyli nie można nadać wartości FFFFFFFFFFFF hex, może być dowolny (i rzecz oczywista unikalny w danej sieci). Z punktu widzenia prawdopodobieństwa są nikłe szanse byś wylosował już używany (łatwiej jest wygrać w Lotto). Innym rozwiązaniem jest poznanie adresów MAC widocznych w sieci lokalnej. W komputerze możesz poznać „szczegóły połączenia sieciowego" (wyklikać, zaczynając od Panel sterowania



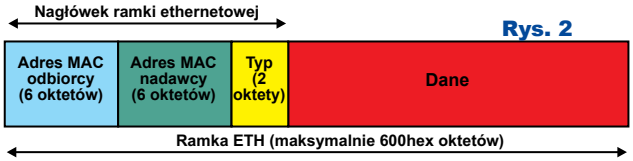
Rys. 1

→ Połączenia sieciowe), jak pokazuje to **rysunek 1**. Widoczny „adres fizyczny 00-18-7D-0A-6E-25" jest adresem MAC w moim komputerze. Jeżeli zrobisz to w każdym komputerze przyłączonym do domowej sieci, poznasz wszystkich „komputerowych domowników". Teraz pozostanie Ci wylosowanie unikalnego numeru dla serwerka. Masz jakieś obawy? Już Ci podpowiadam możliwy adres MAC: 07-19-21-33-39-44.

Natomiast adres IP Twojego serwera www, maska podsieci oraz adres IP bramy domyślnej muszą spełniać bardziej złożone kryteria. Jednak to nic skomplikowanego. Jeżeli poznałeś szczegóły połączeń sieciowych (rysunek 1), to masz już gotowe dwa parametry: maska podsieci i adres bramy domyślnej. W obrębie całej sieci lokalnej są **identyczne**. Teraz zapewne rozumiesz, dlaczego mój serwer www ma takie wartości maski podsieci oraz adres bramy domyślnej, jakie są wkompilowane w program (takie parametry ma moja domowa sieć lokalna). Teraz pozostaje wybrać adres IP dla serwera www. Musi być unikalny w Twojej sieci lokalnej. Jeżeli maska podsieci ma wartość 255.255.255.0, to oznacza, że pierwsze trzy liczby z adresu IP **muszą** być identyczne. Zostało „wylosować" jedną liczbę. Jeżeli poznałeś, Czytelniku, „komputerowych domowników", to pozostało wybrać unikalną. Ja wybrałem 44, bo to „ładna" liczba (a może zasugerowałem się słowami wieszcza Mickiewicza, który w swoim poemacie napisał: *a imię jego będzie czterdzieści i cztery*). Jeżeli musisz zmienić wartości „fabryczne" ustawień sieciowych serwera, to już wiesz, w jaki sposób uzyskać wiedzę pozwalającą określić własne wartości parametrów. Jeżeli obawiasz się, czy poprawnie je określiłeś, to możesz poprosić o pomoc informatyka. Teraz możesz przejść do śródtytułu ***Konfiguracja***, gdzie podane są dalsze najistotniejsze informacje dotyczące wpisania parametrów sieciowych w serwer (istnieje ewentualnie takie rozwiązanie, że zmienisz parametry sieciowe w programie źródłowym i całość przekompilujesz).

Garść wyjaśnień dla początkujących

Jeśli jednak jako elektronik jesteś zielony w kwestiach internetowych, chciałbym Ci wyjaśnić kilka pojęć związanych z sieciami komputerowymi. Jak było zasygnalizowane w części wprowadzającej, jednostką transmisyjną w sieci Ethernet jest ramka składająca się z maksymalnie 600 hex (0x600, czyli dziesiętnie ponad 1500) oktetów – bajtów.



Rys. 2

Rysunek 2 pokazuje format ramki ethernetowej (ETH), której wielkość wynosi maksymalnie 600 hex oktetów. Zawiera ona nagłówek ethernetowy, w którym jest adres MAC odbiorcy, adres MAC nadawcy oraz typ danych.

W poprzednim numerze zawarty był opis sprzętowy serwera www, gdzie istotnymi elementami były układ PHY (jako specjalizowany układ scalony) oraz zespół EMAC (jako integralna część mikrokontrolera ARM). Te dwa elementy są odpowiedzialne za odbiór i wyselekcjonowanie ramek adresowanych do „nas". Selekcja następuje w oparciu o adres MAC (ang. Medium Access Control) będący informacją 48-bitową (6 bajtów). Tu wielu Czytelników może ogarnąć zdziwienie, tak, wszystkie ramki ethernetowe są adresowane zawartością tej części ramki (adres MAC odbiorcy), a powszechnie używane adresy IP nie występują jako informacja adresowa w przesyłaniu ramek. Każdy interfejs ethernetowy, czyli układ PHY wraz z zespołem EMAC, ma własny, unikalny adres MAC (program serwera www nadaje go zespołowi EMAC, o czym będzie w dalszej części). Z powyższego płynie istotny wniosek: każdy komputer, switch czy budowane przez nas urządzenie musi mieć unikalny adres MAC. Panuje przekonanie, że ten adres musi być unikalny w skali świata, ale w rzeczywistości wystarczy unikalność w skali sieci lokalnej.

Drugim istotnym pojęciem związanym ze strukturą ramek ethernetowych jest ***enkapsulacja***. Sięgając do wikipedii, możemy uzyskać następujące wyjaśnienie:

Kapsułkowanie (lub enkapsulacja, od (ang.) encapsulation) – termin odnoszący się do struktury protokołu komunikacyjnego, np. opartego na modelu OSI lub modelu TCP/IP. Kapsułkowanie polega na upakowaniu danych z wyższej warstwy w warstwie niższej danego protokołu po stronie nadawczej, a więc przed wysłaniem pakietu telekomunikacyjnego (datagramu, ramki) w sieciach pakietowych.

Część Czytelników może mieć problem ze zrozumieniem sensu. Postaram się wyjaśnić to obrazowo. Nasz serwer www otrzymuje żądanie otwarcia strony od przeglądarki internetowej, toteż jakiś komputer utworzył ramkę ethernetową i zlecił ją do przekazania do adresata. Wyobraźmy sobie Jasia Wędrowniczka, który ma dostarczyć przesyłkę z miejsca A (adres MAC nadawcy) do miejsca B (adres MAC odbiorcy). Więc zapakował wszystko do plecaczka i ruszył w drogę. W plecaczku miał woreczek z jakąś zawartością (pole *dane* na rysunku 2), na którym było napisane typ = 800hex (pole *typ* na rysunku 2), które oznacza, że jest to ramka IP. W tym woreczku znajdowało się pudełko, na którym było kilka bazgrołów oraz napis TCP. W pudełeczku leżała koperta, na której obok różnych napisów widać było liczbę 80 (numer portu serwera stron www). W kopercie znajdowała się mała kartka, na której było napisane: *poproszę o stronę główną systemu infinity*. Prosta prośba o zawartość strony została upakowana jako dane warstwy najwyższej (prośba przeglądarki internetowej) z dodaniem kilku informacji (między innymi numer portu), tworząc łącznie większą strukturę (kopertę). Po dodaniu kilku szczegółów, między innymi hasła TCP (plus dodatkowych kilka szczegółów), powstaje kolejna większa struktura (pudełko). Uzupełniając zawartość pudełka o kolejne dane, powstaje struktura większa: zawartość worka. Uzupełniony o kilka kolejnych szczegółów (adresy MAC) tworzy zawartość plecaczka naszego Jasia Wędrowniczka wraz ze zleceniem „transportowym". Na tym polega idea kapsułowania, opakowanie danych dodatkowymi odpowiednimi informacjami tworzy większą strukturę (przechodząc do niższej warstwy modelu OSI). Najniższa warstwa modelu OSI, jako warstwa fizyczna, czyli układ PHY, zawiera już wszystko i jest to nasz Jaś Wędrowniczek ze swoim plecaczkiem. Analogicznie występuje termin dekapsulacja, która dotyczy rozpakowania danych. W naszej opowieści będzie to rozpakowanie zawartości plecaczka Jasia Wędrowniczka. Wyjmując z niego woreczek, widzimy, że ma napis 800 hex, czyli wiadomo, co jest w środku: pakiet IP. Rozpakowując worek, znajdujemy w nim pudełko z napisem TCP. To daje kolejną informację, czego można spodziewać się z zawartości pudełka, koperty z dodatkowymi informacjami: przede wszystkim informacji o numerze portu. W przypadku portu o numerze 80 wiemy, że zawartość koperty jest przeznaczona dla serwera www i zawiera coś w języku HTML. Dla bardziej dociekliwych Czytelników na temat modelu OSI przygotowałem dodatkowy dokument (plik o nazwie *ethernet_adresacja.pdf*, który można pobrać z Elportalu jako materiały

dodatkowe do tego numeru EdW).

W powyższej historyjce istnieje jedno ciche założenie: znany jest adres MAC serwera www. W ogólnym przypadku, gdy określone urządzenie wysyła cokolwiek do innego urządzenia, niekoniecznie musi znać adres MAC strony docelowej. Poszczególne komputery/urządzenia są identyfikowane poprzez adres IP, natomiast z przytoczonej historyjki wynika, że wszelka wymiana danych poprzez sieć Ethernet wymaga znajomości adresów MAC. W rzeczywistości w sieci wykorzystywana jest podwójna adresacja. Każda karta sieciowa znajdująca się w komputerze ma adres MAC przyporządkowany jej w fazie produkcji (najczęściej jest zapisany w niewielkiej pamięci EEPROM, którą można odnaleźć na karcie sieciowej) z wymogiem unikalności w skali świata do przesyłania ramek ethernetowych do ściśle określonego urządzenia docelowego oraz adresacja IP (w popularnej notacji czterech liczb rozdzielonych znakiem kropki) nadawana przez użytkowników.

Dlaczego stosowana jest dualna adresacja w sieci Ethernet, wyjaśni to inna krótka historyjka. Wyobraźmy sobie rodzinę: mama Wiktoria, tata Igor oraz córka Natalia. Każda z tych osób ma przydzielony sobie indywidualny adres IP i wszystkie osoby znają powiązanie: osoba ↔ adres IP. Pozwala to na swobodną wymianę informacji między dowolnymi członkami rodziny. Pewnego razu ten spokojny świat dotknęło przykre zdarzenie: komputer Natalii uległ uszkodzeniu. Natalia prowadziła ważną wymianę danych z Wiktorią, była niejako w przymusowej, awaryjnej sytuacji. Ponieważ Natalia była oczkiem w głowie tatusia, Igor oddał jej własny komputer, który po zmianie adresu IP (na taki, jaki jest przyporządkowany Natalii) pozwolił na zakończenie wymiany informacji pomiędzy Natalią a Wiktorią. Wiktoria poza chwilowym brakiem łączności nie zauważa żadnych zmian. Igor natomiast poszedł do sklepu i zakupił nowy i nadał mu adres IP przyporządkowany sobie. Ponownie istnieje harmonijna wymiana informacji pomiędzy trzema osobami. W tej historyjce, z punktu widzenia elektronicznego, zaistniały następujące zdarzenia. Znana jest początkowa adresacja MAC z nadbudową w postaci adresacji IP, to znaczy ustalone jest powiązanie adresu MAC i adresu IP. Komputer Natalii ma jakiś adres MAC (nadany przez producenta komputerów) oraz jakiś adres IP (wynikający z ustalenia pomiędzy członkami rodziny). Analogicznie komputer Wiktorii ma inny adres MAC (producent) oraz inny adres IP (ustalenie rodzinne). Identycznie jest w przypadku komputera Igora. W pewnym momencie nastąpiła zmiana, dotychczasowy adres MAC komputera Natalii wypadł z obiegu. W jego miejsce wszedł adres MAC (fizyczny, niezmienny) używany przez inną osobę (adres IP Igora) i został powiązany z nowym adresem IP (przynależnym Natalii). Finalnie w sieci pojawia się nowy adres MAC (zakup komputera), który zostaje skojarzony z adresem IP używanym przez Igora. Nastąpiła roszada adresów MAC, która nie wpłynęła na komunikację, ponieważ w instalacji występuje druga, logiczna adresacja, którą można modyfikować, a która jest używana do wskazania komputera docelowego dla każdej przesyłanej informacji. Uważny Czytelnik może zadać pytanie: skoro w rzeczywistości używana jest adresacja MAC, a informacje są przesyłane, posiłkując się adresacją IP, to musi istnieć jakiś mechanizm pozwalający powiązać adres MAC z adresem IP. Taki mechanizm istnieje i nazywa się protokół ARP (ang. **A**ddress **R**esolution **P**rotocol) pozwalający powiązać adres IP z adresem MAC. Każdy komputer/urządzenie świeżo włączone do pracy zna jedynie własny adres MAC i rzecz jasna własny adres IP. W trakcie działania w jakiś magiczny (chwilowo nieznany, ale wyjaśniony w dalszej części) sposób poznaje adresy MAC „kontrahentów" i gromadzi te informacje w swojej pamięci jako powiązania adresu IP z adresem MAC. Pierwsze przesłanie do określonego adresu IP wymaga „zdobycia" tego powiązania. Może ono zostać zapisane w pamięci i używane w kolejnych przesłaniach. W rzeczywistości ta prosta idea wymaga dodania mechanizmu automatycznego usuwania zbędnych powiązań, taki „termin przydatności do spożycia" znany w przypadku produktów spożywczych. Przeterminowane powiązania są usuwane z pamięci, toteż w pewnym momencie przymusowa przerwa w komunikacji Wiktoria ↔ Natalia doprowadzi do tego, że w komputerze Wiktorii straci swoją ważność powiązanie: stary adres MAC komputera Natalii i adres IP jej komputera. Ponowne nawiązanie łączności (po wymianie komputerów i modyfikacji adresu IP) stworzy nowe powiązanie: nowy adres MAC komputera Natalii (a dotychczasowy adres MAC komputera Igora) i adres IP przynależny Natalii. Analogicznie w przypadku komputera Igora nastąpi (po ewentualnym przeterminowaniu i usunięciu powiązania) „pozyskanie" nowego powiązania. Tu warto zauważyć, że komputer Wiktorii „nie musiał nic wiedzieć" o zaistniałej roszadzie i pomimo tego nadal prowadzić wymianę informacji. Po chwilowym zachwianiu komunikacji wszystko wróci do normy. Historyjka ta pozwala na uwypuklenie istotnej cechy sieci komputerowych, gdzie ponad techniczną, fizyczną adresacją opartą na adresach MAC istnieje druga logiczna, niejako abstrakcyjna, adresacja IP elastycznie konfigurująca sieć lokalną. Jednak najbardziej istotną cechą adresacji IP jest prostota budowania sieci lokalnej. Bazując na odpowiednich zakresach liczb występujących w adresie IP, można łatwo określić, jakie urządzenia sieciowe należą do sieci lokalnej, a jakie nie. Bazując jedynie na adresach MAC, nie jest to wykonalne ze względu na ogromną przypadkowość adresów MAC. Poza tym, kto miałby ochotę na pamiętanie 48-bitowej liczby (przykładowo 281 474 976 710 654) zamiast w miarę prostego adresu IP (przykładowo 192.168.0.44, choć w rzeczywistości wystarczy zapamiętać tylko dwucyfrową liczbę 44, gdyż pozostała część jest zawsze taka sama w całej sieci lokalnej)? Dla dociekliwych przygotowałem dodatkowy dokument opisujący budowę i ideę adresu IP (plik o nazwie *ethernet_adresacja.pdf*, który można pobrać z Elportalu jako materiały dodatkowe).

Wspomniany wcześniej „magiczny" sposób zdobywania powiązań pomiędzy adresem MAC a adresem IP zostanie wyjaśniony kolejną historyjką. Wyobraźmy sobie, że król X chce przesłać wiadomość do ściśle określonego poddanego Y. W tym celu wysyła swoich posłańców do każdej miejscowości, by odczytali oni mieszkańcom odpowiednią wiadomość. Brzmi ona następująco: ja król X (czyli określony adres IP) zamieszkały na zamku (określony adres MAC) poszukuję pana Y (określony inny adres IP). Jeżeli wśród mieszkańców nie ma pana Y, to każdy z mieszkańców po wysłuchaniu komunikatu stwierdza, że to nie do niego i wraca do domu. Jednak w sytuacji, gdy wiadomość dotrze do mieszkańca Y, to on zgłasza się do posłańca i mówi: proszę poinformować króla (adres MAC i IP znany z królewskiego obwieszczenia), że ja, pan Y (adres IP) zamieszkuję pod tym adresem (podaje adres MAC). Posłaniec wraca z wieściami do króla i jego nadworny pisarz notuje powiązanie pana Y (adres IP) z dostarczonym adresem zamieszkania (adres MAC). Od tej chwili komunikacja się upraszcza, gdyż już nie trzeba wysyłać do wszystkich miejscowości posłańców. Dla dociekliwych również przygotowałem dodatkowy dokument (plik *ethernet_transmisja.pdf*, który można pobrać z Elportalu jako materiały dodatkowe).

Rozumiejąc znaczenie adresu IP jako informacji logicznie przyporządkowanej określonemu adresowi MAC, pozostaje jeszcze wyjaśnić ideę i znaczenie takich pojęć jak: *maska podsieci* oraz *adres bramy domyślnej*. Posłużę się tutaj kolejną historyjką, która pozwoli zrozumieć znaczenie

tych pojęć. Wyobraźmy sobie mieszkańców pewnej ulicy, niech ona nazywa się ulica Ethernetowa. Kompletny adres każdej posesji będzie brzmiał: ulica Ethernetowa <jakiś numer domu>. Jeden z tych mieszkańców z okazji Nowego Roku wygenerował wiele pocztówek z życzeniami noworocznymi i poprosił sąsiada, by je rozesłał. Uczynny sąsiad, który znał jedynie najbliższe otoczenie, gdyż niedawno tu się sprowadził i zdążył poznać jedynie najbliższe sąsiedztwo, wziął plik przesyłek i ruszył zrealizować prośbę. Bierze pierwszą z góry i widzi adres: ulica Ethernetowa 7, wie gdzie to jest i dociera do właściwego miejsca. Kolejna pocztówka była adresowana do miejsca, które nie jest mu znane: ulica Sieciowa 17. Nie wiedział, gdzie to jest, ale rozwiązał problem. Mieszkając przy ulicy Ethernetowej wiedział, że w okolicy znajduje się poczta, która mieści się pod adresem Ethernetowa 24. Dodarł do tego miejsca i nadał przesyłkę na poczcie. W końcu poczta, jako instytucja, funkcjonuje od dawna i doskonale sobie radzi z dostarczaniem przesyłek pod wskazany adres, dokądkolwiek by on prowadził. Z punktu widzenia elektronicznego w powyższej historyjce występuje pełny adres: Ethernetowa 7. Uczynny sąsiad odczytuje każdy docelowy adres i wie, czy dotyczy on znanej mu okolicy, czy może lokalizacji wykraczającej poza ten teren. Decyzję podejmuje w oparcia o część adresu, samej nazwy ulicy: Ethernetowa. Ta część adresu pełni funkcję maski podsieci: jeżeli docelowy adres (ekwiwalent adresu IP) jest zgodny ze znanym wzorcem (ekwiwalent maski podsieci), to wiadomo, że adresatem jest bliższy lub dalszy sąsiad (należy do sieci lokalnej). W przypadku braku zgodności (ulica Sieciowa brzmi inaczej niż ulica Ethernetowa), to uczynny sąsiad zaniesie przesyłkę na pocztę. Dalsze doręczenie przesyłki to już problem poczty. Rzecz jasna, poczta musi się znajdować przy ulicy Ethernetowej (należeć do sieci lokalnej). Historyjka ta uzasadnia potrzebę i znaczenie takich pojęć jak: maska podsieci – informacji o charakterze adresu IP pozwalającej określić, czy docelowy adres IP należy do sieci lokalnej oraz adres bramy domyślnej – pełny adres IP miejsca, do którego są kierowane wszelkie przesyłki wychodzące poza sieć lokalną (w historyjce jest to adres poczty). ***Cały świat zewnętrzny, ujmując żartobliwie łącznie z galaktyką Droga Mleczna, niebędący siecią lokalną, jest utożsamiany z adresem bramy domyślnej***. Wszystkie pakiety sieciowe wysyłane poza sieć lokalną, dokądkolwiek są adresowane, są kierowane na adres bramy domyślnej. Podobnie, wszelkie pakiety sieciowe, skądkolwiek pochodzą (oczywiście spoza sieci lokalnej), są wprowadzane do sieci lokalnej z adresu bramy domyślnej. Ten „magiczny" adres bramy domyślnej jest adresem IP routera w naszej sieci lokalnej. Bardziej dociekliwi Czytelnicy więcej informacji na temat maski podsieci znajdą w materiałach dodatkowych (plik o nazwie *ethernet_adresacja.pdf*).

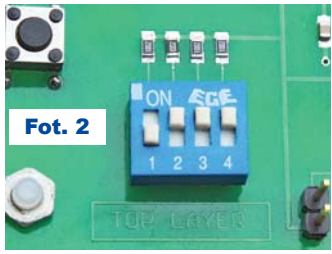
Konfiguracja

Jak już wiemy, konfiguracja naszego bohatera, czyli serwera www, z punku widzenia sieciowego wymaga wpisania do pamięci EEPROM używanej przez procesor LPC2378 jaki ma być jego własny adres MAC, jego własny adres IP oraz jakie są maska podsieci i adres IP bramy domyślnej w lokalnej sieci, gdzie będzie pracował. Jeżeli można wykorzystać moje ustawienia domyślne, nie musisz niczego robić – po dołączeniu do sieci LAN, moduł serwera będzie w niej widoczny (możesz nawet otworzyć serwowaną stronę, ale na razie niewiele z tego wyniknie z braku innych modułów systemu). Jeśli trzeba coś zmienić, również nie ma problemu! Wszystkie parametry prześlemy do naszego serwera www z jakiegoś komputera w postaci tekstowej poprzez UART mikrokontrolera ARM, a konkretnie przez złącze P501 – DSUB 9-pinowy żeński (parametry transmisji: 9600 bps transmisja bez parzystości). Do tej (jednorazowej) konfiguracji możemy wykorzystać dowolny emulator terminalu, przykładowo program HYPERTERMINAL z parametrami transmisji szeregowej podanej powyżej. Jest to jedno z możliwych rozwiązań, ale mało wygodne, gdyż należy wszystko napisać bez pomyłek (w serwerze w obsłudze konfiguracji nie ma zrealizowanej funkcjonalności związanej z klawiszem Backspace) oraz konieczny będzie kabelek bez linii modemowych (3-żyłowy łączący bez przeplotu styki 2, 3 i 5 złączki DSUB 9 męską i żeńską), gdyż ten program na starcie ustawia linie modemowe do stanu, który dla mikrokontrolera oznacza stan permanentnego resetu. Początkowo sam używałem emulatora terminalu, ale mając na uwadze Twoją wygodę, napisałem dedykowany do tego celu program (SERWCFG). Użycie mojego programu zwalnia ze stosowania kabelka 3-żyłowego (na korzyść wersji pełnomodemowej), bowiem program w sposób „świadomy" steruje stykiem 4 i tranzystorem resetującym Q501. W innym przypadku (inny program emulatora terminalu ewentualnie przewód bez linii modemowych) będzie istniała konieczność naciskania przycisku SW101. Do komunikacji programu konfiguracyjnego z serwerem wystarczy połączyć złącze P501 serwera z portem COM w komputerze. Rozumiem Twój niepokój, Czytelniku, zapewne w Twoim komputerze nie ma takiego portu. W nowoczesnych laptopach wyginęły złącza typu COM i zostały zastąpione złączami USB. Jest na to rozwiązanie: należy zastosować konwerter USB ↔ RS232. W ofercie handlowej jest szeroki wybór tego typu przejściówek. Konwertery z „normalnej półki" lub z „górnej półki" umożliwiają, oprócz samego przesyłania szeregowego danych (linie Rx, Tx), sterowanie także dodatkowymi liniami modemowymi. Tego typu konwerter będzie niezbędny w operacjach konfiguracji serwera, do załadowania kodu programu do mikrokontrolera LPC2378 i do konfiguracji wszelkich modułów stosowanych w całym systemie Infinity. Zależnie od szczegółów, co chcemy konfigurować, trzeba ustawić odpowiednią kombinację przełączników w DIPSWITCH (SW301). Program serwera rozpoznaje następujące kombinacje związane z modyfikacją zawartości pamięci EEPROM.

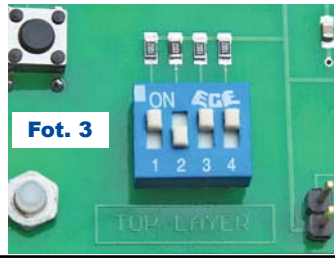
Normalna praca. Kombinacja przełączników pokazana na **fotografii 1** oznacza zwykłe uruchomienie serwera podczas jego normalnej pracy. Parametry konfiguracyjne sieci są odczytane z pamięci EEPROM i zastosowane w programie.

Ustawienia domyślne. Kombinacja pokazana na **fotografii 2** prowadzi do wpisania (przywrócenia) wkompilowanych w program domyślnych wartości parametrów konfiguracyjnych – to powrót do ustawień typu *factory default*. Po zrealizowaniu operacji zapisu do pamięci EEPROM, program zatrzymuje się, by istniała możliwość zmiany ustawień przełączników i ponownego wystartowania.

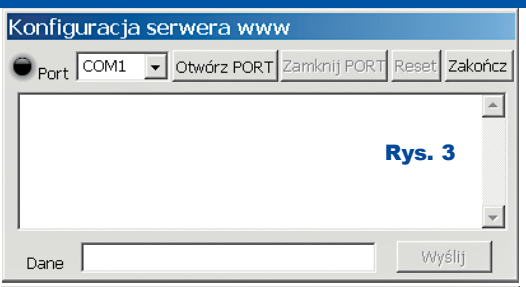
Adres MAC. Kombinacja pokazana na **fotografii 3** oznacza zmianę adresu MAC. Program oczekuje wprowadzenia z emulatora terminalu 12-cyfrowej liczby szesnastkowej (bez odstępów, przecinków, myślników itp.). Uwaga: przy użyciu emulatora terminala w przypadku liczb szesnastkowych **A...F** należy wpisywać je jako „wielkie" (nie jako **a…f**), w przypadku użycia programu SERWCFG, nie ma to znaczenia, gdyż ten program dokona zamiany małych liter na wielkie (dodatkowy



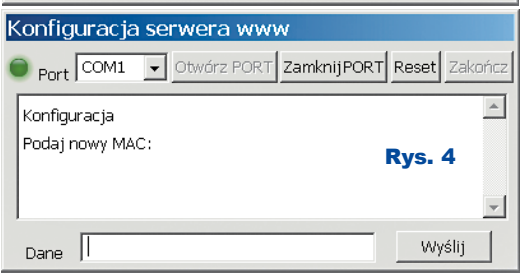
Fot. 2



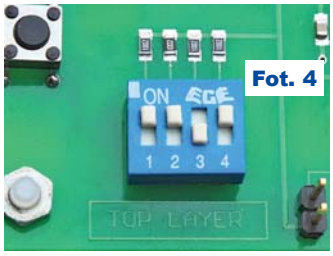
Fot. 3



Rys. 3



Rys. 4

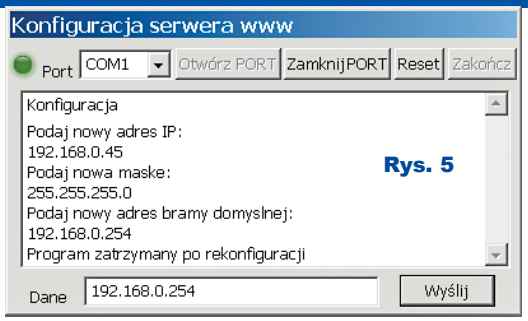


Fot. 4

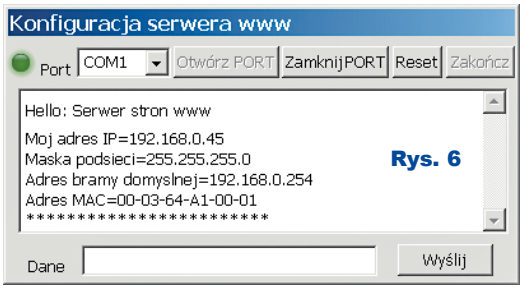
argument, by użyć dedykowanego programu). Zmiana adresu MAC jest konieczna w przypadku włączenia do sieci lokalnej więcej niż jednego serwera www (w jednej sieci lokalnej nie może być dwóch jednakowych adresów MAC). Gdy do sieci dołączony jest jeden egzemplarz, może pozostać „wartość fabryczna". Po zapisaniu danych do pamięci EEPROM program zatrzymuje się. Należy przestawić przełączniki i zresetować mikrokontroler ARM.

„Dane sieciowe". Kombinacja pokazana na **fotografii 4** oznacza zmianę parametrów adresowych. Jest ich trzy, a zapisywane są w „kropkowej" notacji adresu IP. Program wyświetla informację pozwalającą zidentyfikować parametr i oczekuje na wprowadzenie nowej wartości. Po wczytaniu kompletu, dane są zapisywane w pamięci EEPROM, a program zatrzymuje się z odpowiednim komunikatem. Tej funkcjonalności trzeba użyć w przypadku, gdy parametry „fabryczne" nie odpowiadają adresacji używanej w docelowej sieci lokalnej, zaproponowane nie muszą odpowiadać każdemu Czytelnikowi, który miałby ochotę na zbudowanie opisywanego serwera. Użycie do konfiguracji serwera wspomnianego programu (SERWCFG) pokazane jest na **rysunkach 3, 4** i **5** (program ten został utworzony w środowisku Lazarus, które jest wolnym, bezpłatnym narzędziem do tworzenia programów w języku PASCAL).

Posługiwanie się programem SERWCFG nie należy do skomplikowanych. W pierwszej kolejności należy wybrać port komunikacyjny i kliknąć na przycisk *Otwórz PORT*. Spowoduje to zaświecenie zielonej kontrolki oraz wygeneruje sygnał reset dla mikrokontrolera ARM (jako wygenerowanie impulsu na odpowiedniej linii modemowej). Po resecie, mając przykładowo przełączniki DIPSWITCH w pozycji jak na fotografii 4, w okienku zostanie wyświetlona prośba o nowy adres IP. Po wpisaniu danych w okienku *Dane* i kliknięciu *Wyślij*, dane są przesyłane do serwera. Po podaniu wszystkich parametrów program zapisuje je w pamięci EEPROM i zatrzymuje się wyświetlając odpowiedni komunikat. Od tej chwili serwer ma zapamiętane nowe parametry sieciowe.



Rys. 5



Rys. 6

Również w przypadku standardowego uruchomienia serwera (przełączniki w pozycji pokazanej na fotografii 1) generowany jest odpowiedni komunikat, który można odebrać w okienku programu konfiguracyjnego (**rysunek 6**).

Tyle informacji podstawowych. Dalsza część artykułu przeznaczona jest dla dociekliwych i nieco bardziej zaawansowanych Czytelników, którzy chcą poznać programowe szczegóły realizacji serwera www, jednak zachęcam do lektury wszystkich.

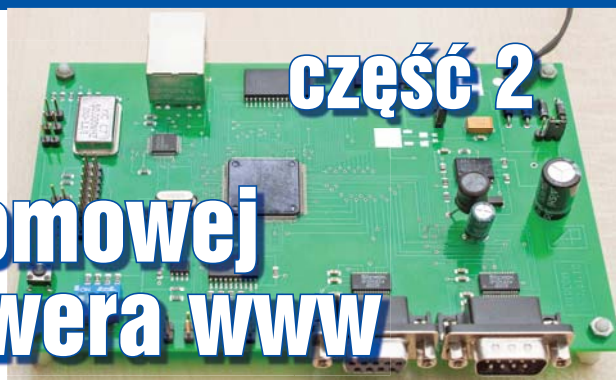
Za miesiąc przedstawione zostaną informacje dla dociekliwych.

Andrzej Pawluczuk
apawluczuk@vp.pl

Infinity – system automatyki domowej
Oprogramowanie serwera www

kit 3232 AVT

część 2



W poprzednich odcinkach omówiliśmy budowę modułu serwera www, a ostatnio jego konfigurację. W niniejszym odcinku chciałbym przedstawić programowe szczegóły realizacji serwera. Zasadniczo materiał przeznaczony jest dla bardziej zaawansowanych, jednak do lektury zachęcam wszystkich. Jak wspomniałem wcześniej, obsługa „karty sieciowej" mikrokontrolera na najbardziej podstawowym poziomie nie jest skomplikowana. Chcę to pokazać w bieżącym numerze. Prostota i elegancja wysłania lub odebrania ramki ethernetowej jest jednak okupiona zaawansowanym rozwiązaniem sprzętowym, Niech te informacje zainspirują, zachęcą Czytelników i Czytelniczki do własnych przemyśleń i eksperymentów, gdyż to jest najlepszy sposób zdobywania wiedzy.

Tylko dla dociekliwych

Podkreślam: ta część opisu systemu automatyki domowej przeznaczona jest dla Czytelników, którzy chcą i są w stanie zrozumieć szczegóły programowej realizacji serwera. Dla mniej zaawansowanych lektura może być powodem zniechęcenia i frustracji (początkujący powinni wrócić do lektury po zakończeniu cyklu, gdy zdobędą szerszą wiedzę, gdyż nie jest wykluczone, że na łamach EdW zaprezentuję kilka bardziej szczegółowych rozważań o tematyce sieciowej). A na razie wystarczy, że zaprogramują procesor gotowym wsadem.

A teraz trudniejsze informacje dla dociekliwych. W Elportalu dostępne są wszystkie wspomniane programy i pliki, w tym program źródłowy, realizujący serwer www. Zachęcam do indywidualnej analizy, ale teraz chcę zwrócić uwagę na pewne istotne i interesujące szczegóły. Ponieważ programy są bardzo obszerne (pracowałem nad nimi wiele miesięcy), nie jest możliwe przedstawienie w EdW nawet kluczowych fragmentów, dlatego wspomniane dalej w artykule listingi 1...15 zostały umieszczone w Elportalu w postaci dodatkowych 15 małych plików tekstowych. Oto omówienie ciekawszych szczegółów:

W oprogramowaniu wprowadziłem własne nazwy podstawowych typów (**listing 1**). Do najczęściej używanych należą UCHAR (liczba całkowita 8-bitowa bez znaku – bajt), USHORT (liczba całkowita 16-bitowa bez znaku) oraz ULONG (liczba całkowita 32-bitowa bez znaku).

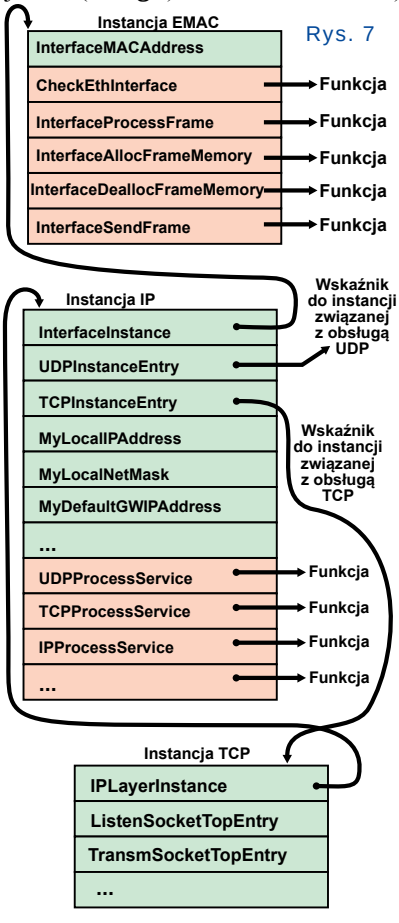
Parametry „fabryczne" są pokazane na **listingu 2**. Dane są już zapisane w formacie binarnym używanym w oprogramowaniu. Przykładowo adres IP o wartości C0A8002C w zapisie szesnastkowym po konwersji każdego bajtu do postaci dziesiętnej odpowiada adresowi IP=192.168.0.44. Podobnie zapisane są pozostałe dane o charakterze adresowym. Adres MAC zawsze jest w postaci binarnej zapisanej na 48 bitach, czyli sześciu bajtach, stąd w programie jest stosowana jako tablica sześciobajtowa.

Uruchomienie programu serwera sprowadza się do użycia funkcji *main* (**listing 3**). Nie jest tu możliwe opisanie działania wszystkich funkcji oprogramowania, ale dociekliwi Czytelnicy mogą samodzielnie przeanalizować program, gdyż komplet z programem źródłowym jest dostępny w Elportalu jako materiały dodatkowe (plik o nazwie vircom_soft.zip).

W pierwszej kolejności inicjowany jest system przerwań (wywołanie funkcji *SysInit*). Kolejna funkcja (*InitSysTime*) inicjuje działanie kolejkowania zdarzeń. Jest to istotna funkcjonalność programu odpowiedzialna za realizację pewnych czynności, które należy wykonać po określonym interwale czasu od danego momentu. Kolejne dwie funkcje (*HardwareInit* i *SoftwareInit*) konfigurują środowisko mikrokontrolera do wymaganej postaci: inicjują obsługę zewnętrznej pamięci RAM, konfigurują porty do oczekiwanej funkcji oraz inicjują zmienne programu do niezbędnej wartości początkowej. Po zainicjowaniu obu układów UART do obsługi transmisji szeregowej następuje wczytanie danych z pamięci EEPROM (funkcja *GetConfigData*). Strukturę bloku konfiguracyjnego (zapisanego w pamięci EEPROM) i realizację wspomnianej funkcji pokazuje **listing 4**, gdzie następuje wczytanie całego bloku. Program jest w stanie rozpoznać niezaprogramowaną pamięć i zainicjować pierwszy zapis do pamięci EEPROM jako domyślne parametry zastępcze („fabryczne").

W kolejnym kroku wczytany jest stan przełączników DIPSWITCH (funkcja *DIPSwitchSerwice*) i ewentualna realizacja działania wynikająca z ustawień przełączników. Obsługę pokazuje **listing 5**, gdzie po wczytaniu stanu przełączników program warunkowo przechodzi do ponownego zapisania do pamięci EEPROM parametrów fabrycznych lub przechodzi do wczytania nowej wartości adresu MAC, ewentualnie do wczytania danych o charakterze adresu IP. Po każdej operacji wykonany jest zapis nowych danych do pamięci nieulotnej oraz program jest zatrzymywany z odpowiednim komunikatem. W sytuacji, gdy nie jest rozpoznany żaden ze „stanów wyjątkowych" przełączników, program wychodzi z powyższej funkcji i realizuje kolejne działania zawarte w funkcji *main* (listing 3). Po

Rys. 7



wysłaniu w kanał szeregowy w postaci tekstowej aktualnych danych konfigurujących sieć (wywołanie funkcji *HelloMessage*), program przechodzi do zainicjowania zespołu EMAC i całej obsługi sieci (wywołane funkcji *StartNet*, która jest pokazana na **listingu 6**).

Zanim przejdę do wgłębiania się w program, muszę wyjaśnić pewien rodzaj struktury, którą określam jako instancję. W języku programowania C odpowiada to strukturze (*typedef struct*), gdzie część pól struktury zawiera specyficzne dla danej struktury dane oraz część pól jest wskaźnikami do funkcji o określonej funkcjonalności. Idea tej struktury odpowiada koncepcji obiektów w języku C++, lecz jest znacząco bardziej uproszczona. Postać instancji związanej z warstwą fizyczną pokazuje **listing 7**. Znajdują się tam klasyczne dane (jak adres MAC serwera) oraz wskaźniki do określonych funkcji związanych z odbieraniem ramek (*CheckEthInterface*), nadawaniem ramek (*InterfaceSendFrame*), przydziałem pamięci na odebrane ramki (*InterfaceAllocFrameMemory*, *InterfaceDeallocFrameMemory*) oraz przetwarzaniem odebranych ramek (*InterfaceProcessFrame*).

Kompletna obsługa sieci wymaga utworzenia dodatkowych instancji związanych z przetwarzaniem protokołów z wyższych poziomów modelu OSI. Program serwera www wymaga instancji pokazanych na **rysunku 7**, gdzie instancja EMAC (listing 7) odpowiada za fizyczną transmisję danych. Nadrzędną dla niej jest instancja związana z obsługą protokołu IP (jej strukturę można znaleźć w programach źródłowych). Zawiera ona dowiązanie do instancji EMAC oraz dowiązania do protokołów wyższych rzędów (UDP, TCP). Obok wymienionych powiązań instancja przechowuje między innymi własne parametry sieciowe (adres IP, maskę podsieci, adres IP bramy domyślnej). Wskaźniki do funkcji związanych z przetwarzaniem protokołu IP, podobnie jak w instancji EMAC, prowadzą do właściwych funkcji w programie. Przykładowo *UDPProcessService* jest odpowiedzialne za przetwarzanie ramek związanych z tym protokołem. Podobnie *TCPProcessService* prowadzi do funkcji obsługującej ramki TCP.

Powyższe rozwiązanie uelastycznia stosowanie części oprogramowania obsługującego sieć Ethernet jako niezależnej części oprogramowania mikrokontrolera ARM użytego w budowie określonego urządzenia. Dobrym tego przykładem jest właśnie serwer www, w którym nie jest używany protokół UDP (nie było takiej potrzeby). Gdyby przy rozpakowaniu odebranej ramki zostało stwierdzone, że dotyczy ona protokołu UDP, to w funkcji rozpakowującej musiałoby wystąpić wywołanie funkcji do obróbki UDP. „Sztywne" wywołanie funkcji spowoduje dodanie do kodu programu czegoś, co nie jest używane. Zamiast dodawać do programu komplet kodu, rozwiązanie z zastosowaniem wskaźników do funkcji pozwala dolinkować do programu jedynie te części, które są używane. W przypadku odebrania przez serwer pakietu UDP, gdyż nie można zabronić jakiemuś programowi mającemu dostęp do sieci lokalnej wysłania takiego pakietu do serwera, zostanie on rzecz jasna rozpoznany. Ponieważ funkcja *UDPProcessService* w tym programie ma wskaźnik pusty, taki pakiet zostanie zignorowany.

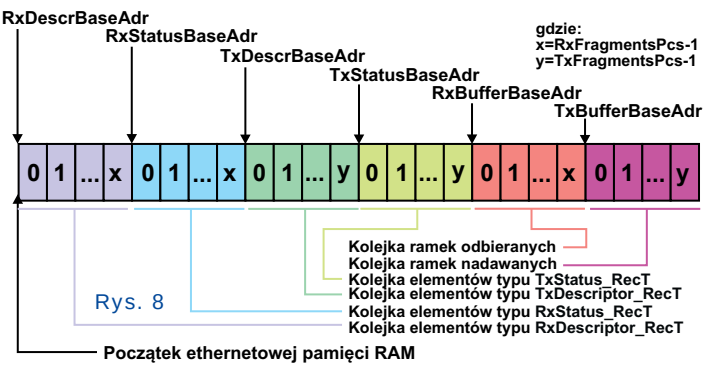
Inicjowanie i utworzenie powiązań jest zrealizowane w funkcji *StartNet* (listing 6). Instancja związana z poziomem IP jest utworzona w wyniku wywołania *InitIPLayerInstance* (tworzy dowiązanie do instancji EMAC i wypełnia wskaźniki do określonych funkcji). Niektóre z nich będą miały wskaźnik pusty jak przykładowo powiązanie z obsługą protokołu UDP. Kolejne wywołanie (w *StartNet*) funkcji *StandardAddTCPInstance* dołącza do istniejącej już instancji IP funkcje do obsługi wszystkiego co jest związane z protokołem TCP (jak na rysunku 7). Jednak cofnijmy się w analizie działania funkcji *StartNet* do początku.

Funkcja *InitPHYLayerInstance* jest odpowiedzialna na zainicjowanie obsługi sieci w warstwie fizycznej, co polega na właściwym skonfigurowaniu układu PHY oraz zespołu EMAC. Postać tej funkcji pokazana jest na **listingu 8**. Po skonfigurowaniu warstwy fizycznej sieci następuje konfiguracja „logiczna", wynikająca z adresacji IP (wywołanie funkcji *InitIPLayerInstance*), gdzie kluczową informacją jest własny adres IP oraz maska podsieci pozwalająca identyfikować „kontrahentów" należących do sieci lokalnej. Ostatnią czynnością jest zainicjowanie protokołu TCP jako wywołanie funkcji *StandardAddTCPInstance*.

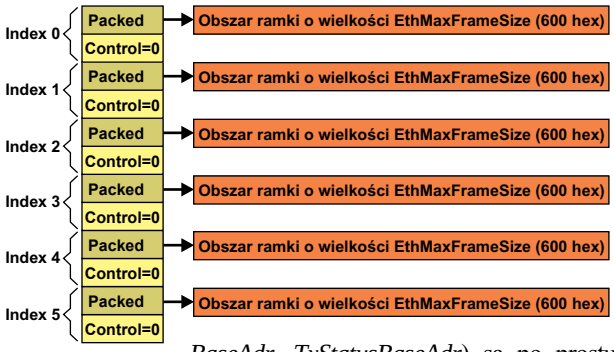
Warto przyjrzeć się bliżej procedurze inicjowania zespołu EMAC wraz z układem PHY. Zapoznanie się z obsługą zespołu sieciowego w mikrokontrolerze ARM pozwala przekonać się, że obsługa tego interfejsu na poziomie fizycznym nie jest skomplikowana, a zlecone operacje są wykonywane autonomicznie. Jedną z pierwszych czynności, jakie należy wykonać, jest włączenie zasilania dla EMAC (po sygnale reset dla mikrokontrolera domyślnie nie jest on włączony). Polega to na ustawieniu odpowiedniego bitu w rejestrze PCONP. W kolejnym kroku należy skonfigurować wyprowadzenia portu P1 do współpracy z układem PHY, czyli określić, że wybrane linie tego portu stanowią interfejs RMII do układu PHY. Polega to na odpowiednich wpisach do rejestru PINSEL2 i PINSEL3. Uważny Czytelnik zauważy, że wpis do rejestru PINSEL2 jest warunkowy. Ten mikrokontroler występuje w wersji „starej" oraz „nowej", z tego powodu realizowany jest odmienny zapis do tego rejestru. Wariant można rozpoznać po odczycie zawartości rejestru MAC_MODULEID. Po skonfigurowaniu kilku rejestrów w EMAC następuje programowe zresetowanie układu PHY jako użycie funkcji *WriteToPHY*. Przeglądając jej instrukcje, łatwo zauważyć, że zapis do PHY sprowadza się do wypełnienia określonych rejestrów EMAC (podać co i gdzie ma być zapisane) i poczekania na właściwy stan flagi potwierdzający zakończenie operacji. Podobnie przebiega odczyt z układu PHY, który można prześledzić na przykładzie odczytu identyfikatora układu PHY (wywołanie funkcji *ReadFromPHY*), czyli zapytać

układ PHY, czy „ma na imię DP83848". Analogicznie do zapisu, w operacji odczytu poprzez odpowiednie wpisy do rejestrów EMAC można zlecić wymaganą akcję i poczekać na potwierdzenie jej zakończenia. Uzyskane od układu PHY dane są do odczytania w określonym rejestrze EMAC. Prawda, że proste? W kolejnym kroku należy poinformować układ PHY o oczekiwanej prędkości komunikacyjnej (10MB/100MB). W przypadku narzuconych prędkości transmisyjnych sprowadza się to do odpowiednich zapisów do układu PHY. W przypadku autonegocjacji (jako wariantu najbardziej uniwersalnego) problem jest trochę bardziej złożony. Wstępnie układ PHY „zostaje poproszony do swoistego dogadania się" z istniejącą siecią. Z punktu widzenia elektronicznego PHY musi „złapać link" od sieci, by wywnioskować, jaka jest prędkość komunikacyjna. Ta operacja może potrwać dłuższą chwilę (z punktu widzenia programu całkiem sporą). Odczytując w pętli odpowiedni status układu PHY, wychwycony jest moment, w którym PHY zakończył autonegocjację. Pętla, w której jest odczytywany status, ma limit iteracji, gdyż w przypadku startu serwera bez połączenia z siecią (wystarczy wyjąć wtyk RJ45) program się zapętli. W dalszej kolejności program oczekuje na stan LINK (sygnalizowany również włączeniem odpowiedniej diody sterowanej bezpośrednio przez układ PHY), gdyż odczytując odpowiedni status, można uzyskać zwrotne potwierdzenie o rzeczywistej prędkości transmisyjnej. Podobna sytuacja, odczyt w pętli z limitem iteracji, przeciwdziała zapętleniu się programu. Informacja o uzyskanej prędkości jest istotna, gdyż w dalszej części wpływa na konfigurację zespołu EMAC. W sytuacji, gdy program startuje bez przyłączenia do sieci, coś trzeba założyć (że jest to sieć 100MB). Z powyższego łatwo wywnioskować, że start programu serwera www z podłączeniem do sieci przebiega trochę łagodniej. Po wpisaniu do EMAC adresu MAC (sześciobajtowego identyfikatora sieciowego na poziomie fizycznym), doprogramowaniu kilku rejestrów, zespół EMAC jest gotowy do działania. W całym ciągu instrukcji konfigurujących EMAC istotnym elementem jest wywołanie funkcji *InitEMACDescr*. Jest to bardzo istotny fragment programu, a z drugiej strony pokazuje ciekawe rozwiązanie dotyczące nadawania i odbierania ramek ethernetowych. Postać funkcji pokazuje **listing 9**.

Do obsługi sieci mikrokontroler LPC2378 ma dedykowaną pamięć RAM (poza podstawową pamięcią operacyjną) o wielkości 16kB. W obszarze tej pamięci należy programowo zainicjować odpowiednie struktury danych, które funkcjonalnie odpowiadają realizacji kolejek FIFO (**rysunek 8**). Tak, zespół EMAC mikrokontrolera ARM autonomiczne przetwarza kolejki FIFO, gdzie elementami kolejek są różne struktury danych, między innymi ramki ethernetowe o wielkości 600 hex oktetów (dla dociekliwych na temat idei kolejek FIFO przygotowałem dodatkowy dokument – *bufory cykliczne.pdf*, który można pobrać jako materiały dodatkowe z Elportalu).



Rys. 8

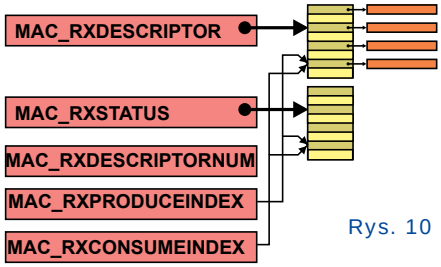


Rys. 9

Zacznę od tego, że kolejki FIFO związane z EMAC są podzielone na dwie kategorie: dotyczące nadawania danych oraz odbierania danych. Liczba elementów w buforach cyklicznych związanych z kolejkami jest określona przez dwie stałe, które można modyfikować według własnego uznania: *RxFragmentsPcs* i *TxFragmentsPcs* (listing 9). Definiują one liczbę elementów w buforach cyklicznych w odpowiednich kolejkach (tablice przechowujące elementy są indeksowane od 0 do odpowiednio *RxFragmentsPcs-1* lub *TxFragmentsPcs-1*) jak statusy czy deskryptory, których struktury definiują typy *RxDescriptor_RecT*, *RxStatus_RecT*, *TxDescriptor_RecT*, *TxStatus_RecT* (listing 9, na podstawie dokumentacji producenta UM10211 dostępnej na stronach NXP). Znając liczbę elementów w każdej kolejce oraz wielkość każdego elementu wyrażoną w bajtach, istnieje możliwość obliczenia wielkości poszczególnych obszarów i ich położenia w pamięci ethernetowej RAM.

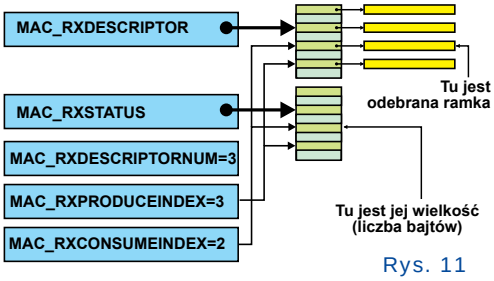
Pierwszy obszar (*RxDescrBaseAdr*, rysunek 8, **listing 10**) jest na początku obszaru pamięci RAM. Drugi (*RxStatusBaseAdr*) znajduje się o *RxFragmentsPcs*sizeof(RxDescriptor_RecT)* bajtów dalej, czyli ma adres poprzedniego obszaru (*RxDescrBaseAdr*) powiększony o wyżej określoną wielkość. Analogicznie są określone początki wszystkich pozostałych jako adres poprzedniego elementu powiększony o jego wielkość. Obszary te, w zależności od rodzaju kolejki, wymagają odpowiedniego zainicjowania zawartości. Elementy o charakterze statusowym (*RxStatusBaseAdr*, *TxStatusBaseAdr*) są po prostu wyzerowane. Pozostałe, określające bufory komunikacyjne, wymagają dowiązania do obszarów ramek (pola o nazwie *Packed*, listing 9), muszą zawierać wskazanie (adres) na obszar w pamięci, w którym jest przechowywana ramka ethernetowa. Wyjaśnia to **rysunek 9** będący kolejką FIFO ramek nadajnika (kolejki odbiornika są analogiczne). W tym przypadku jest utworzona tablica 6 elementów (taką wartość ma stała *TxFragmentsPcs*, indeksowana od 0 do 5), gdzie każde pole *Packed* wskazuje na indywidualny obszar przeznaczony na ramkę. W identyczny sposób zorganizowana jest kolejka deskryptorów odbiornika.

Tak utworzone cztery kolejki deskryptorów i statusów dla nadajnika i odbiornika są „doczepione" do rejestrów zespołu EMAC. **Rysunek 10** pokazuje powiązanie odpowiednich rejestrów (związanych z odbieraniem ramek) z właściwymi kolejkami w pamięci RAM (dowiązane są dwie kolejki, kolejne dwie dotyczące nadajnika są rozwiązane w identyczny sposób). Rejestr *MAC_RXDESCRIPTOR* wskazuje (zawiera adres) kolejkę deskryptorów określających obszary przewidziane na odbiór ramek



Rys. 10

ethernetowych (każdy element deskryptora w polu *Packed* wskazuje na indywidualny obszar przeznaczony na odebraną ramkę). Kolejny rejestr *MAC_RXSTATUS* wskazuje na kolejkę statusów związanych z operacją odbierania danych (ramek). W rejestrze *MAC_RXDESCRIPTORNUM* przechowywana jest wielkość kolejki i należy wpisać tam liczbę elementów kolejki pomniejszoną o jeden, rejestr ten dotyczy zarówno kolejki deskryptorów, jak i statusów, gdyż liczba elementów w obu kolejkach jest identyczna. Rejestr *MAC_RXPRODUCEINDEX* określa pierwsze wolne miejsce w kolejkach (tam zostanie umieszczona odebrana ramka). Podobne znaczenie ma rejestr *MAC_RXCONSUMEINDEX* wskazujący na deskryptor zawierający ramkę do przetworzenia przez program. Oba te rejestry zawierają indeks do odpowiednich tablic stowarzyszonych z kolejkami. W przypadku odbierania ramek, zespół EMAC jest „producentem" informacji zapisywanych w kolejkach (treści odebranych ramek, statusy), natomiast program serwera www jest „konsumentem" danych. W ogólnym przypadku może się tak zdarzyć, że zespół EMAC „wyprodukuje" kilka ramek, których oprogramowanie serwera nie „wygarnie". Będą one przechowywane w kolejkach (oczywiście, jeżeli zostanie przekroczona liczba nieodczytanych ramek, czyli przekroczony próg określony przez stałą *RxFragmentsPcs*, to nastąpi utrata danych). Jeżeli zawartość obu rejestrów (produkujących i konsumujących) jest identyczna, to oznacza, że kolejki są puste. Przykładowy stan rejestrów po odebraniu ramki pokazuje **rysunek 11**. Wszystkie kolejki zawierają po 4 elementy, toteż w rejestrze *MAC_RXDESCRIPTORNUM* jest wpisana liczba 3. Przed odebraniem ramki kolejki były puste, dlatego zawierały przykładowo liczbę 2 (*MAC_RXPRODUCEINDEX* i *MAC_RXCONSUMEINDEX*). Po odebraniu ramki zespół EMAC umieszcza jej zawartość w buforze określanym przez deskryptor o indeksie 2 i jej wielkość w odpowiednim miejscu w kolejce statusowej (również o indeksie 2) oraz zwiększa zawartość rejestru *MAC_RXPRODUCEINDEX* o jeden (po operacji zawiera 3). Różna zawartość rejestru „produkcyjnego" oraz „konsumenckiego" oznacza, że jest odebrana nowa ramka. Dane zawarte w buforze należy przepisać w inne miejsce i zwiększyć zawartość rejestru „konsumenckiego" o jeden, informując w ten sposób zespół EMAC, że dane zostały „sprzątnięte" z pamięci kolejkowej.



Rys. 11

Procedurę odbierania ramek pokazuje **listing 11**, gdzie następuje programowa realizacja wyżej opisanych czynności. Do pełnego zrozumienia jej działania wyjaśniam kilka szczegółów: instrukcja *EMACInstanceRec.InterfaceAllocFrameMemory* jest funkcją, która przydziela miejsce na odebraną ramkę gdzieś w pamięci operacyjnej, w pętli *for* realizowane jest przepisanie odebranej ramki z kolejki do przydzielonego obszaru (dla zwiększenia wydajności po cztery bajty), zwiększona jest zawartość rejestru konsumenckiego o jeden z uwzględnieniem ewentualnego „zawrócenia wskaźnika" na początek tablicy oraz interpretacji odebranych danych (wywołanie *EMACInstanceRec.InterfaceProcessFrame*).

Odnosząc działanie programu w zakresie dekapsulacji do historyjki Jasia Wędrowniczka, funkcja pokazana na listingu 11 odpowiada rozpakowaniu plecaczka, a wywołanie funkcji *EMACInstanceRec.InterfaceProcessFrame* prowadzi do funkcji pokazanej na **listingu 12** (wyjęciu woreczka), gdzie między innymi jest rozpoznany napis IP na woreczku (wariant instrukcji *case* o wartości *FRAME_IP*). To z kolei prowadzi do otwarcia woreczka i uzyskania pudełka. Mając napisane na pudełku hasło TCP, trafiamy poprzez wywołanie funkcji *IPLayerInstance->TCPProcessService* (**listing 13**) do funkcji pokazanej na **listingu 14**. Tam rozpoznany jest numer portu i poszukiwany jest otwarty port o odpowiednim numerze (wywołanie funkcji *LocateSocket_TCB* oraz *LocateSocket_Listen*).

Nadawanie ramek ethernetowych jest zbliżone do odbierania. Procedurę ilustruje **listing 15**, gdzie po przepisaniu zawartości nadawanej ramki do obszaru pamięci wskazanego przez odpowiedni deskryptor (odpowiedniego bufora w kolejkach związanych z EMAC), należy zwiększyć zawartość rejestru produkcyjnego związanego z funkcjonalnością nadawania. Tu program serwera jest „producentem" a zespół EMAC jest „konsumentem" danych. Podobnie jak przy odbieraniu, rejestr *MAC_TXPRODUCEINDEX* wskazuje na wolny deskryptor, który z kolei w polu *Packed* wskazuje na miejsce, do którego należy przepisać zawartość nadawanej ramki i po zakończeniu kopiowania zwiększyć zawartość rejestru (uwzględniając liczbę dostępnych deskryptorów), co z kolei jest sygnałem dla zespołu EMAC, by zabrać się do pracy.

W kolejnym odcinku będzie kontynuacja opisu działania programu serwera www.

W Elportalu w materiałach dodatkowych do tego numeru znajdują się następujące pliki:

- vircom_soft.zip – program źródłowy oraz binarny serwera www,
- serwercfg.zip – program źródłowy oraz binarny do konfiguracji serwera www,
- ramtest.zip – program źródłowy oraz binarny dla mikrokontrolera ARM realizującego test zewnętrznej pamięci RAM,
- bufory cykliczne.pdf – dokument opisujący ideę kolejek budowanych w oparciu o bufory cykliczne,
- ethernet_adresacja.pdf – dokument opisujący model OSI oraz ideę adresów IP oraz maski podsieci,
- ethernet_transmisja.pdf – dokument opisujący transmisję ramki ethernetowej.

Andrzej Pawluczuk
apawluczuk@vp.pl