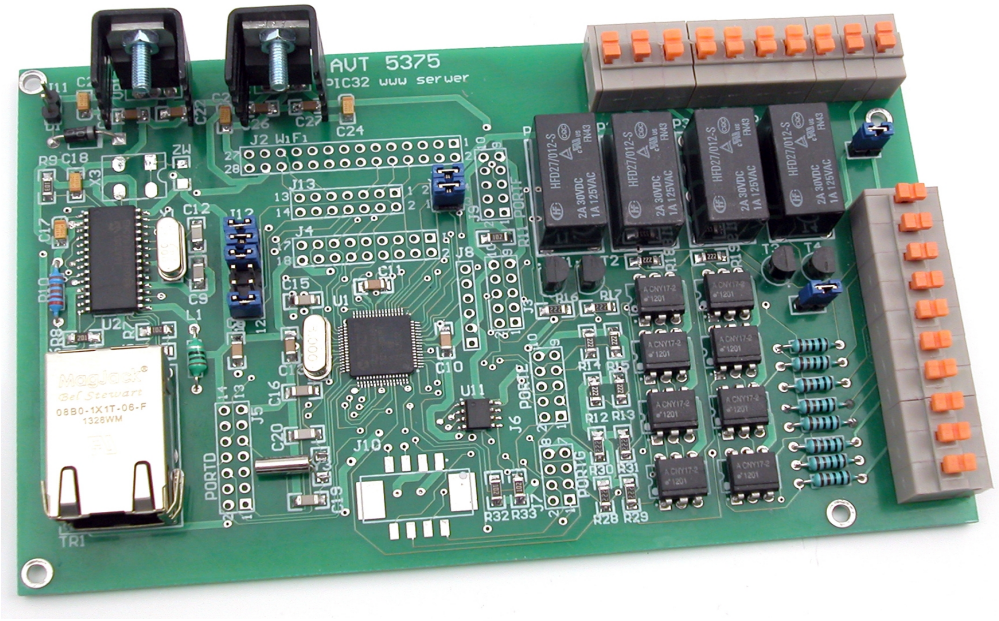
Serwer WWW

AVT 5375



Urządzenia sterowania i automatyki od bardzo dawna są realizowane w technice cyfrowej. Mikroprocesory, a właściwie mikrokontrolery, są idealnymi elementami do wszelkiego rodzaju sterowników – zdalnie sterowanych i pracujących autonomicznie. Zdalne sterowanie od zawsze wymagało zapewnienia odpowiedniego łącza do komunikacji pomiędzy sterownikiem, lub wieloma sterownikami, a panelem obsługi wykonującej czynności sterownicze, lub odczytującej stan położenia łączników i pomiary. Firma Microchip produkuje szybkie, niezawodne i dobrze wyposażone mikrokontrolery PIC32MX. Są one oferowane trochę „pod prąd" cortexowego szaleństwa, ale mocne wsparcie firmy powoduje, że stosując je nie ma się wrażenia „niszowości". W obecnej wersji serwer ma możliwość sterowania 4 przekaźnikami i odczytywania stanów 8 optoizolowanych wejść binarnych. Wbudowana konsola umożliwia konfigurowanie urządzenia za pomocą komend wpisywanych z klawiatury. ***Rekomendacje:*** *serwer może przydać się w układach automatyki przemysłowej i domowej oraz jako baza do tworzenia własnych urządzeń.*

Niegdyś połączenie na dużą odległość pomiędzy operatorem a sterownikiem było realizowane przez łącza telefoniczne i wiązało się z dużymi kosztami budowy lub dzierżawy. Zastąpienie połączenia miedzianego łączami światłowodowymi zwiększyło możliwości łącza (przepustowość, niezawodność), ale dalej było to rozwiązanie drogie. Z tego powodu zdalne sterowanie na było w zasadzie zarezerwowane dla rozwiązań komercyjnych i tylko tam, gdzie było to uzasadnione ekonomicznie (na przykład w energetyce).

Burzliwy rozwój Internetu i drastyczny spadek cen stałych łączy internetowych całkowicie zmienił ten stan. Za umiarkowaną cenę można łączyć się z dowolnym miejscem na Ziemi. Jedynym ograniczeniem jest dostęp do Internetu. No może nie jedynym: protokoły i mechanizmy przesyłania danych są ściśle zestandaryzowane, a standaryzacja dotyczy też warstwy sprzętowej. Oznacza to, że żeby użyć Internetu do przesyłania danych sterownik musi mieć wbudowany przynajmniej interfejs Ethernet i pracować w lokalnej sieci LAN z wykorzystaniem rodziny protokołów TCP/IP. Spore wymagania implementacji protokołów sieciowych odnośnie do zasobów sprzętowych i konieczność stosowania interfejsu Ethernet początkowo znacznie ostudziły zapał projektantów do stosowania tego medium w komunikacji ze sterownikami mikroprocesorowymi. Jednak oczywiste stało się, że prędzej czy później stanie się on jednym ze standardów komunikacyjnych.

Mikrokontrolery pracujące w układach sterowników mają różne zasoby i wydajności. Dobiera się je zależnie od wykonywanych funkcji. W wielu przypadkach mogą to być proste i tanie jednostki 8-bitowe, w których brakuje zasobów na zaimplementowanie protokołów z rodziny TCP/IP. Dlatego szereg firm oferuje zewnętrzne układy interfejsowe z wbudowanym stosem TCP/IP i sprzętowym Ethernetem. Mikrokontroler łączy się z nimi poprzez interfejs szeregowy np. SPI. To rozwiązanie ma swoje zalety, ale też i wady. Możliwe jest, że interfejs na przykład przestanie być produkowany.

Mikrokontrolery z większymi zasobami mogą obsługiwać stos TCP/IP samodzielnie. Chyba najbardziej znany jest bezpłatny stos

W ofercie AVT*

AVT-5375 A	AVT-5375 B
AVT-5375 C	AVT-5375 UK

Podstawowe informacje:

- Mikrokontroler PIC32MX340F512H.
- Mostek PHY z układem ENC28J60.
- Stos TCP/IP firmy Microchip.
- Oprogramowanie wykonane za pomocą MPLAB i TCP Makera.
- Wejścia i dołączone do złącz śrubowych.
- 4 wyjścia przekaźnikowe, 8 wejść binarnych.
- Interfejs Ethernet, wbudowana strona WWW

Dodatkowe materiały na CD/FTP:

ftp://ep.com.pl, user: 13621, pass: 175brjf7

- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

Projekty pokrewne na CD/FTP:

(wymienione artykuły są w całości dostępne na CD)

AVT-5366	Sterownik uniwersalny zgodny z Arduino (EP 10/2012)
AVT-5340	Konwerter Ethernet/UART (EP 4/2012)
AVT-1668	AVTduino Ethernet – moduł Ethernet dla Arduino (EP 3/2012)
AVT-5250	Karta przekaźników z interfejsem Ethernet (EP 8/2010)
AVT-5200	Uniwersalny sterownik ethernetowy (EP 9/2009)
AVT-1528	Interfejs internetowy z ENC28J60 (EP 8/2009)
AVT-5197	Pilot z WiFi (EP 8/2009)
AVT-5166	Serwer HTTP (EP 1/2009) Easy TCP/IP TWI (I²C) (EP (12/2008)
AVT-5157	Przekaźnik internetowy (EP 11/2008)
AVT-2859	Internetowy sterownik urządzeń (EdW 3/2008)
AVT-5118	Wyświetlacz ethernetowy (EP 12/2007)
AVT-5111	Zdalny system pomiarowy z interfejsem Ethernet (EP 09/2007)
AVT-974	Sterownik z interfejsem TCP/IP (EP 3/2007)
AVT-966	Karta przekaźników sterowana przez internet (EP 2/2007)
AVT-1443	Uniwersalny interfejs ethernetowy (EP 1/2007)
AVT-956	Ethernetowy sterownik (EP 11/2006)
AVT-953	Karta wejść z interfejsem Ethernet (EP 10/2006)
AVT-927	Uniwersalny interfejs internetowy (EP 4-5/2006)
AVT-5055	Internetowy interfejs dla mikrokontrolera (EP 3-5/2002)

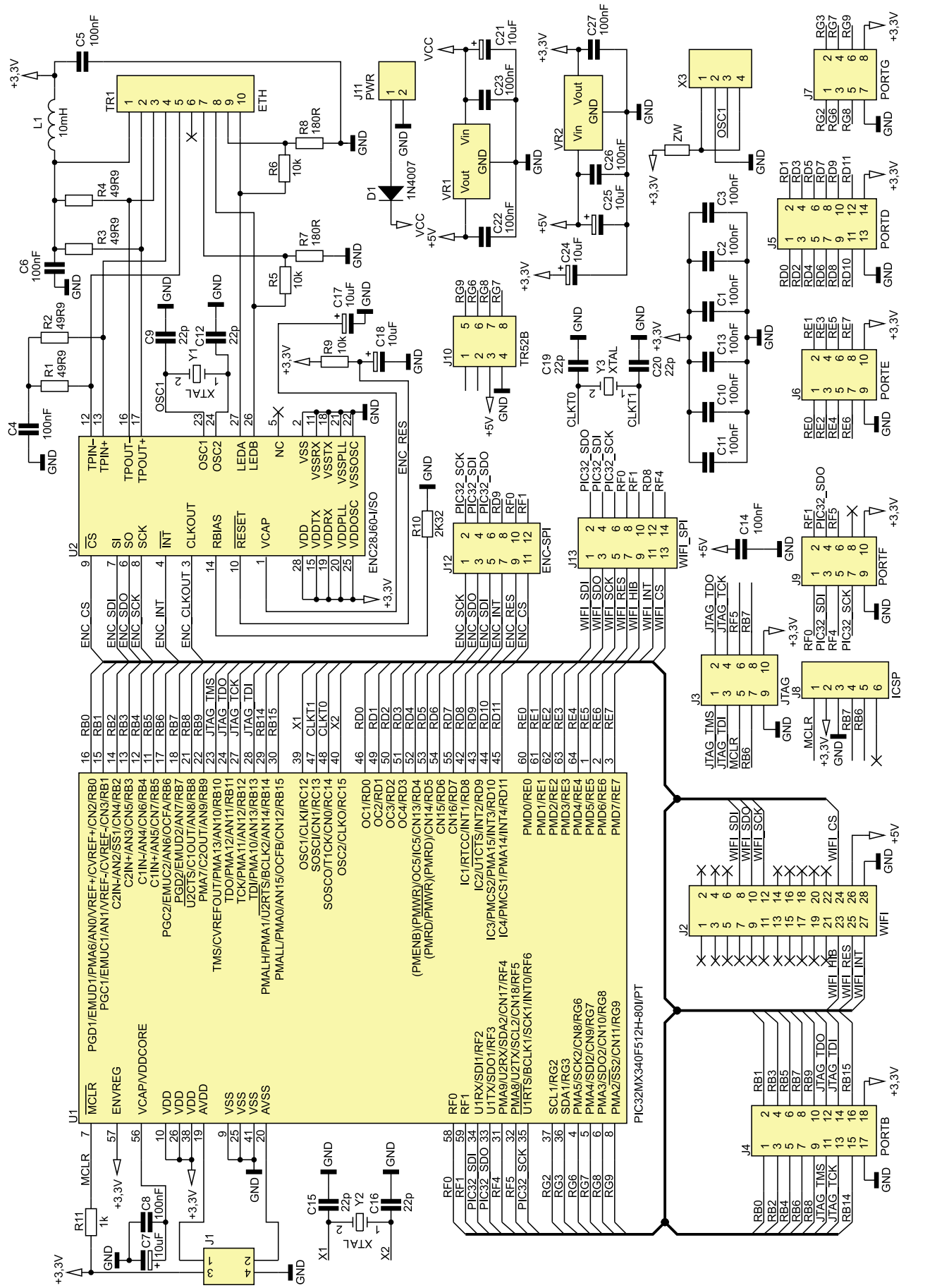
*** Uwaga:**

Zestawy AVT mogą występować w następujących wersjach:

AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymieniony w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). **http://sklep.avt.pl**

firmy Microchip, który można zaimplementować nawet na większych 8-bitowcach z rodziny PIC18. Programowa implementacja stosu pozwala na wybranie tylko potrzebnych protokółów i daje większą kontrolę nad wymianą danych bardziej zaawansowanemu użytkownikowi. Można też korzystać z narzędzi integrujących warstwę aplikacji z warstwami stosu.



Rysunek 1a. Schemat serwera WWW z mikrokontrolerem PIC32MX340

Podejmując decyzję o implementacji protokołów sieciowych w mikrokontrolerze trzeba rozważyć współpracy z układem PHY dostosowującym sygnały danych do standardu Ethernet. Tu również swoje standardy wyznacza Microchip mający w ofercie różne układy interfejsowe, w tym najstarszy i najbardziej znany ENC28J60. Produkuje też rodzinę mikrokontrolerów PIC18F z wbudowanym układem PHY.

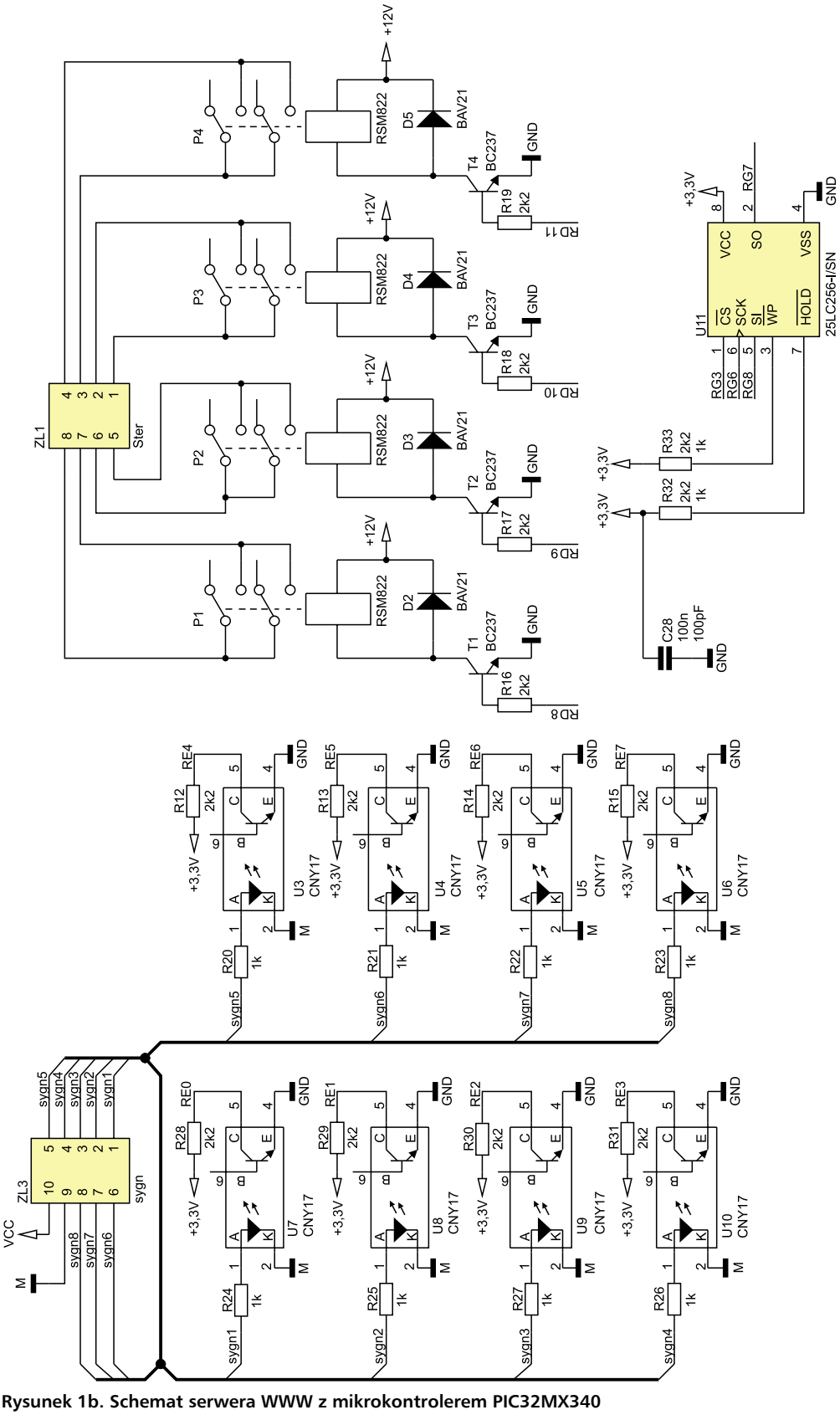
Serwer WWW – założenia projektowe

Każdy konstruktor zamierzający zbudować sterownik do zdalnego sterowania musi zastanowić się nad wyborem kanału komunikacyjnego. Możliwości technicznych jest sporo. Dla niewielkich odległości można użyć łącza radiowego z gotowymi rozwiązaniami sieciowymi, na przykład Bluetooth. Jak już wspomniałem, dla sterowania na większe odległości rozsądnym rozwiązaniem jest Internet . Jednak wybór sieci LAN i protokółów TCP/IP to coś więcej, niż wybór kanału. Niejako gratis otrzymujemy możliwość zbudowania graficznego interfejsu użytkownika w postaci strony WWW. Wystarczy tylko zaimplementować obsługę protokółu http i można tworzyć interfejs z podstronami i możliwością nawigowania pomiędzy nimi. Całość zajmuje mało miejsca w pamięci i jest łatwa do wykonania, bo większość pracy związanej z wyświetleniem tekstu i grafiki jest wykonywana przez przeglądarkę internetową, która otrzymuje ciąg instrukcji sterujących dotyczących na przykład rodzaju i koloru czcionki. Taka filozofia powoduje, że implementacja serwera ze stroną WWW bez dużych bitmap, animacji i innych „bajerów” zajmuje zaskakująco mało miejsca w pamięci danych mikrokontrolera. Nawet kiedy pracujemy tylko w sieci LAN sterownik z funkcją serwera WWW jest bardzo atrakcyjną alternatywą dla innych rozwiązań. Z tych powodów postanowiłem zaprojektować i zbudować sterownik będący jednocześnie serwerem strony WWW.

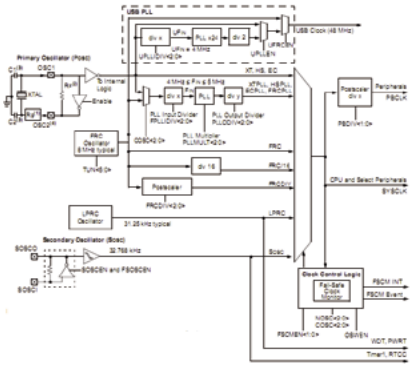
Decyzja o budowie musiała być poprzedzona wyborem mikrokontrolera. Serwer można zbudować nawet z użyciem mikrokontrolera 8-bitowego i będzie on pracował poprawnie. Żeby się o tym przekonać, wykonałem próby z mikrokontrolerem PIC18F67J60 z wbudowanym interfejsem PHY. Docelowo chciałem jednak zastosować szybki i dobrze wyposażony mikrokontroler 32-bitowy, aby nie mieć problemu z wydajnością i zasobami przy ewentualnym rozwijaniu oprogramowania i dodawaniu nowych funkcji.

Po analizie dostępnych rozwiązań zdecydowałem się na kompleksowe rozwiązanie Microchipa. Powodów wyboru było kilka, ale najważniejsze z nich to rozwijany od wielu firmowy stos i możliwość użycia rewelacyjnego narzędzia TCPMaker, opisywanego na łamach EP.

Microchip produkuje szybkie, niezawodne i dobrze wyposażone w pamięci i peryferia mikrokontrolery PIC32MX. Są oferowane



Rysunek 1b. Schemat serwera WWW z mikrokontrolerem PIC32MX340



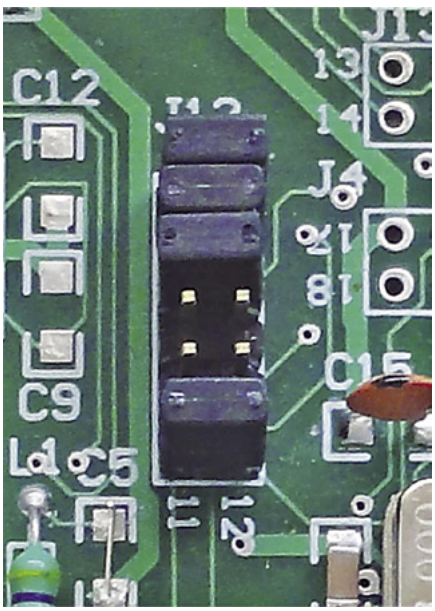
Rysunek 2. Schemat układu taktowania PIC32MX340

trochę pod prąd Cortexowego szaleństwa, ale mocne wsparcie firmy powoduje, że stosując je nie ma się wrażenia „niszowości". Wybrałem PIC32MX340F512H, mikrokontroler z rdzeniem PIPS32 M4K, taktowanym z częstotliwością maksymalną 80 MHz, wyposażony w 512 kB pamięci Flash i 32 kB pamięci danych SRAM. Układy peryferyjne również są więcej niż wystarczające: 10-bitowy przetwornik A/D, zegar czasu rzeczywistego RTC, interfejsy I²C i SPI oraz wiele innych.

Po wyborze mikrokontrolera trzeba było wybrać interfejs PHY sieci Ethernet. Tu nie było trudności – wybór padł na „klasyka" ENC28J60. Stos Microchipa ma wbudowany driver tego układu i jego implementacja jest pozbawiona jakiegokolwiek ryzyka.

Oprogramowanie serwera jest tworzone przy użyciu narzędzia TCPMaker generującego gotowy projekt dla środowiska MPLAB. Ten projekt z plikami źródłowymi stosu i obsługi wątków związanych z transmisja danych, musi być skompilowany za pomocą MPLAB-C32.

W obecnej wersji serwer ma możliwość sterowania 4 przekaźnikami i odczytywania stanów 8 optoizolowanych wejść binarnych. Wbudowana konsola umożliwia konfigurowanie urządzenia za pomocą komend wpisywanych z klawiatury.



Fotografia 3. Położenie zworek złącza J12

Rozwiązania układowe

Schemat płytki głównej pokazano na **rysunku 1**. Mikrokontroler jest zasilany napięciem +3,3 V ze stabilizatora LDO VR2. Jego rdzeń musi być zasilany napięciem +1,8V. Aby nie komplikować układu zasilania w strukturę mikrokontrolera wbudowano wewnętrzny stabilizator +1,8 V. Istnieje też możliwość zasilania rdzenia z zewnętrznego napięcia. Układ stabilizatora jest programowany poziomem wejścia *ENVREG*. Po podaniu na niego napięcia +3,3 V, napięcie zasilania rdzenia jest pobierane z wewnętrznego stabilizatora. Do wyprowadzenia *Vcore/Vcap* trzeba dołączyć kondensator o pojemności 10 µF i małym współczynniku ESR. W naszym wypadku jest to kondensator tantalowy C7 o pojemności 10 µF. Równolegle warto dołączyć do niego kondensator ceramiczny 100 nF (C8).

Wewnętrzny regulator jest włączany, gdy *ENVREG* = +3,3 V i wszystkie wejścia zasilania są dołączone do źródła. Dotyczy to również wejścia AVDD zasilającego przetwornik. Jest ono doprowadzone do złącza J1, tak aby można było go zasilić precyzyjnym, odkłóconym napięciem. Nawet kiedy nie korzystamy z przetwornika A/D, trzeba tam podać +3,3 V. W przeciwnym wypadku wewnętrzny stabilizator, a co za tym idzie mikrokontroler, nie będą działały.

Po dołączeniu wejścia *ENVREG* do masy, trzeba podać zewnętrzne napięcie +1,8 V na wyprowadzenie Vcore/Vcap.

Mikrokontroler jest taktowany przez wbudowany generator kwarcowy z zewnętrznym kwarcem Y2 o częstotliwości 8 MHz. Jak łatwo domyślić się, ta częstotliwość będzie mogła być powielana wewnątrz układu, tak aby rdzeń i układy peryferyjne mogły być taktowane z wyższymi częstotliwościami. W naszym wypadku rdzeń jest taktowany sygnałem o częstotliwości 80 MHz. Schemat układu taktowania pokazano na **rysunku 2**. Konfigurowanie częstotliwości taktowania odbywa się przez zapisanie odpowiednich bitów rejestrów konfiguracyjnych (**listing 1**).

Częstotliwość sygnału z oscylatora kwarcowego jest dzielona w wejściowym dzielniku przez 2 (FPLLIDIV = DIV_2) a następnie powielana przez PLL razy 20 (FPLLMUL = MUL_20). W wyniku tych operacji częstotliwość generatora jest powielana przez 10 i rdzeń jest taktowany sygnałem o częstotliwości 80 MHz. Dzielnik zegara taktującego układy peryferyjne nie jest włączony (FPBDIV = DIV_1), to również one są taktowane sygnałem o częstotliwości 80 MHz. Linie portów wyprowadzono na 2-rzędowe listwy goldpinów.

Do taktowania wewnętrznego modułu RTC użyto drugiego generatora kwarcowego SOSC (*Secondary Oscillator*). Jest on przystosowany do współpracy z oscylatorem kwarcowym o częstotliwości 32768 Hz (Y3).

Istotnym elementem sterownika jest układ scalony ENC28J60. Może on być taktowany wewnętrznym generatorem kwarcowym z kwarcem o częstotliwości 25 MHz lub sygnałem wyjściowym ze scalonego generatora 25 MHz (Y3). ENC28J60 komunikuje się z mikrokontrolerem poprzez interfejs SPI. Jego sygnały: ENC_SCK, ENC_SDO, ENC_SDI i ENC_CS, są doprowadzone do jednej strony dwurzędowego złącza goldpinów J12. Do drugiego rzędu doprowadzono linie interfejsu SPI mikrokontrolera. Żeby PIC32MX mógł komunikować się z ENC28J60, jest konieczne założenie zworek na J12, tak by połączyć linie interfejsu SPI (**fotografia 3**). To rozwiązanie pozwala na użycie sygnałów RF2, RF3 i RF6 jako uniwersalnych linii I/O w wypadku, gdy moduł PHY nie jest stosowany.

Najbardziej rozpowszechniony standard Ethernet 10BaseT przesyła dwoma parami

Wykaz elementów

Rezystory:
R1…R4: 49,9 Ω SMD
R5, R6, R9: 10 kΩ SMD
R7, R8: 200 Ω SMD
R12…R19, R28…R31: 2,2 kΩ SMD
R20…R27: 1 kΩ (0,6 W metalizowany)
R10: 2,32 Ω (1%; 0,6 W metalizowany)
R11, R32, R33: 1 kΩ SMD
Kondensatory: (SMD 1206)
C1…C6, C8, C10, C11, C13, C14, C22, C23, C26, C27, C28: 100 nF
C9, C12, C15, C16, C19, C20: 22 pF
C7, C17, C18, C21, C24, C25: 10 µF/10 V (SMD)
Półprzewodniki:
D1: 1N4007
D2…D5: 1N4148 SMD
T1…T4: BC237
U1: PIC32MX340F512H-80I/PT
U2: ENC28J60-I/SO
U3…U10: CNY17
UR1: 7805
UR2: LM2937 3,3
Inne:
P1…P4: przekaźnik JRC27F
L1: 10 µH
TR1: złącze Ethernet z transformatorem 08B0-1X1T-06-F
Y1: kwarc 25 MHz
Y2: kwarc 8 MHz
Y3: kwarc zegarkowy 32768 Hz
Listwa podwójnych goldpinów
Złącza TLZ24V
Radiatory

Listing 1. Definicja dotycząca bitów konfiguracyjnych

```
#elif defined(__PIC32MX__)
#pragma config FPLLODIV = DIV_1, FPLLMUL = MUL_20, FPLLIDIV = DIV_2, FWDTEN =
OFF, FPBDIV = DIV_1, POSCMOD = XT, FNOSC = PRIPLL, CP = OFF
```

skrętki dwa sygnały różnicowe. Do połączenie używa się specjalnych kabli zakończonych wtykami RJ45. Większość urządzeń sieciowych (switche, huby, karty sieciowe itp.) ma wbudowane gniazda RJ45. Dodatkowo, jest wymagana izolacja galwaniczna połączenia. Do tego celu stosuje się specjalnie zaprojektowane transformatory sygnałowe. W handlu są dostępne zintegrowane gniazda z transformatorem sygnałowym i 2 diodami LED sygnalizującymi stan połączenia sieciowego. W projekcie zastosowałem takie złącze typu MagJack 08B0-1X1T-06-F. Połączenie z transformatorem wymaga dopasowania za pomocą rezystorów R1, R2, R3 i R4 (50 Ω). Wyjścia LEDA i LEDB sterują diodami sygnalizacyjnymi umieszczonymi w złączu. Rezystory R7 i R8 ograniczają prąd diod LED.

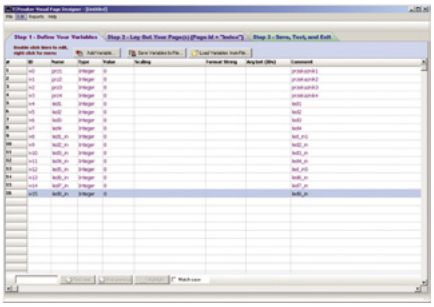
Część wykonawcza serwera składa się z 4 torów wyjściowych zbudowanych z przekaźników i 8 torów wejściowych izolowanych galwanicznie transoptorami. Sterowanie cewkami przekaźników zapewniają drivery z tranzystorem bipolarnym BC237 (T1…T4). Równie dobrze można tu użyć tranzystora unipolarnego o odpowiednich parametrach. Równolegle do każdej z cewek przekaźników dołączono diody BAV21 (D1…D4) tłumiące impulsy indukowane w cewkach w momencie wyłączania napięcia. Rezystory R16…R19 ograniczają prąd bazy. Styki przekaźników są doprowadzone do złącza ZL. Zastosowałem tu sprężynowe złącza produkowane przez firmę Wago.

Dwustanowe sygnały wejściowe maja charakter prądowy. Stan aktywny oznacza wymuszenie w obwodzie diody transoptora wejściowego (CNY17) U3…U10 przepływu prądu o natężeniu 10..20 mA. Prąd jest ograniczany przez rezystory R20…R26 o wartości 1 kΩ. Wejścia są przystosowane do napięcia wejściowego +12 V DC. Jeżeli przez diodę transoptora płynie prąd, to jego tranzystor jest nasycany i na wyjściu występuje niskie napięcie. Poziomy wejść są odczytywane prze linie RE0…RE7. Poziom wysoki na wejściu kiedy tranzystor transoptora jest zatkany wymuszają rezystory R12…R15 i R28…R31.

Mikrokontrolery PIC32MX nie maja wbudowanej nieulotnej pamięci typu EEPROM. Dla potrzeb projektu zastosowałem pamięć 25LC256 z interfejsem SPI: SI (dane wejściowe RG8), SO (dane wyjściowe RG7), SCK (zegar RG6) i CS (wybór układu RG3).

Program serwera

Jak już wspomniałem funkcje serwera zostały oprogramowane przy wykorzystaniu TCP Makera. Ten pakiet programowy generuje kompletny projekt dla środowiska MPLAB IDE ze źródłowymi plikami stosu TCP/IP

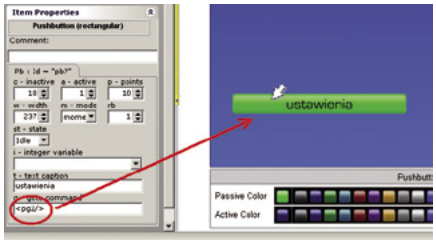


Rysunek 4. Definiowanie zmiennych serwera

Rysunek 5. Strona główna

i plikiem mtGen.c z procedurami obsługi przepływu danych pomiędzy komputerem a serwerem.

Projektowanie serwera rozpoczynamy od określenia kompilatora (tu MPLAB C-32) i zdefiniowania zmiennych przesyłanych wykorzystywanych do komunikacji. Na po-



Rysunek 6. Definiowanie przycisku nawigacji pomiędzy stronami

Listing 2. Funkcja obsługi zmian zmiennej prz3

```
#define STER1   (LATEbits.LATE0)   //sterowanie 1
#define STER2   (LATEbits.LATE1)   //sterowanie 2
#define STER3   (LATEbits.LATE2)   //sterowanie 3
#define STER4   (LATEbits.LATE3)   //sterowanie 4

// Event handler for variable:  prz3 - przekaznik3
void eventprz3(void)
{
  PosLcd(1,2);
  DispLcd("Ster->");//lokalny wyświetlacz LCD
  if(prz3==1)  //testowanie rodzaju akcji Wyłącz/Załącz
  {
    led3=1;  //sygnalizacja sterowania załączenia
    DispLcd(„PRZ3 <ZAL>");  //lokalny wyświetlacz
    STER3=1;  //załączenie przekaźnika
     EEWrite(0xff,CTRL3);  //zapisanie stanu w EEPROM
  }
  else
  {
    led3=0;  //sygnalizacja wyłączenia
    DispLcd(„PRZ3 <ODL>");
    STER3=0;  //odłączenie przekaźnika
    EEWrite(0,CTRL3);
  }
  led3TxFlag=1;
}
```

Listing 3. Funkcja odtwarzająca stany wyjść sterujących

```
void mtUserInit(void)
{
  //Initialize all TCPmaker variables here:
  prz1 = 0;
  prz2 = 0;
  prz3 = 0;
  prz4 = 0;
//odtworzenie sterowań
  if(EERead(CTRL1)==0)//przekaźnik 1 STER1=0; else STER1=1;
  if(EERead(CTRL2)==0)//przekaźnik 2 STER2=0; else STER2=1;
  if(EERead(CTRL3)==0)//przekaźnik 3 STER3=0; else STER3=1;
  if(EERead(CTRL4)==0)//przekaźnik 4 STER4=0; else STER4=1;
  It1[0] = 0;  //Init to empty string
}  // mtUserInit()
```

trzeby projektu zdefiniowałem 16 zmiennych: 4 dla poleceń sterowniczych przy1… przy4, 4 dla zwrotnego potwierdzenia wykonania polecenia sterowniczego led1…led4 i 8 dla przesyłania stanu wejść separowanych transoptorami. Definicje zmiennych wykonuje się z zakładce *Step1 -Define Your Variable* (**rysunek 4**). W następnym kroku projektuje się stronę i ewentualne podstrony. W tym projekcie zaprojektowałem stronę główną z trzema podstronami. Na stronie głównej umieszczone zostały przyciski nawigacji do stron : sterowania przekaźnikami, wyświetlania stanu wejść cyfrowych i konsoli wprowadzania komend do sterownika serwera (**rysunek 5**). Aby przycisk nawigacji *Pb Pushbutton* powodował przejście na przykład do strony *pg2*, w polu *g – goto commnds* okna *Item Properties* trzeba wpisać polecenie <pg2/> (**rysunek 6**). Jest to najważniejsza definicja obok określenia koloru przycisku (*Passive/Active color*) i ewentualnej zmiany rozmiaru ustawienia dla przycisku nawigacji. Stronę sterowania przekaźnikami pokazano na **rysunku 7**. W prawym dolnym rogu jest umieszczony przycisk *Home*, który w polu *Goto commands* okna *Properties Item* ma wpisaną komendę <index/>.

Sekwencyjne załączanie i wyłączanie przekaźnika występuje po naciśnięciu przycisku odpowiadającego danemu przekaźnikowi. Z przyciskami sterowania są związane zmienne prz1…prz4 zdefiniowane w TCP-Maker (**rysunek 8**). Po przyciśnięciu na przykład *przekaźnik 3* zmienna prz3 przyjmuje na przemian wartości 0 – wyłącz i 1 – załącz.

Powiązanie zmiennej prz3 (w trakcie edycji strony) z przyciskiem *Pbuton TCP-Maker – przekaźnik 3*, definiuje w projekcie dla mikrokontrolera zmienną unsigned char prz3 i szkielet funkcji void eventprz3(void) umieszczony w pliku mtGen.c. Ta funkcja jest wywoływana po odebraniu (poprzez sieć Ethernet) informacji o akcji naciśnięcia przycisku *przekaźnik 3*. Funkcję obsługi przycisku pokazano na **listingu 2**.

Rysunek 7. Wygląd strona sterowania przekaźnikami

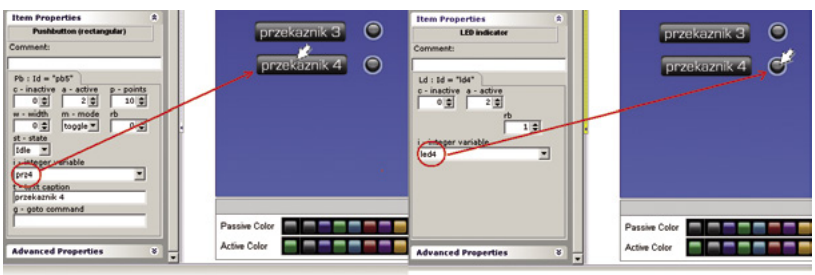
Zasadniczym zadaniem tej funkcji jest testowanie wartości zmiennej sterującej przy3 i wystawianie stanów na liniach sterujących. Operacja sterująca jest wspierana dodatkową sygnalizacją prawidłowości wykonania. Na stronie WWW pokazanej na **rysunku 9**, obok przycisków sterujących są umieszczone wirtualne diody *LED Indicator*. Zostały z nimi powiązane zmienne led1…led4. TCPMaker w projekcie dla mikrokontrolera umieszcza definicje zmiennych 8-bitowych led1…led4 i definicje flag led1TxFlag…led4TxFlag zezwalających na transmitowanie stanu tych zmiennych. Aby zmienić stan wirtualnej diody obok przycisku *przekaźnik 2* z wyłączonego na załączony, trzeba do zmiennej led2 wpisać 1 i ustawić flagę led1TxFlag. Dalej wszystko załatwia RTOS w sterowniku i przeglądarka. Dołączenie dodatkowej diody LED zwiększa pewność, że przekaźnik sterujący został wysterowany. Zmiana stanu tej diody nie oznacza, że sterowanie zostało wykonane, ale że na poziomie programowym sterownik dostał polecenie i odesłał potwierdzenie jego otrzymania. W programie dla każdego z przekaźników jest zdefiniowana podobna funkcja do tej pokazanej na listingu 2.

Każda zmiana sygnału sterującego przekaźnikami jest zapisywana w pamięci EEPROM. Po włączeniu zasilania lub zerowaniu mikrokontrolera, procedura inicjalizacji odczytuje te stany z pamięci i wysyła

Listing 4. Funkcja inicjalizacyjna mtConnect

```
#define IN1   (PORTEbits.RE0) //wejście 1
#define IN2   (PORTEbits.RE2) //wejście 2
#define IN3   (PORTEbits.RE3) //wejście 3
#define IN4   (PORTEbits.RE4) //wejście 4
#define IN5   (PORTEbits.RE5) //wejście 5
#define IN6   (PORTEbits.RE6) //wejście 6
#define IN7   (PORTEbits.RE7) //wejście 7
#define IN8   (PORTEbits.RE8) //wejście 8

void mtConnect(void)
{
  if(IN1==0) led1_in=1; else led1_in=0;
  if(IN2==0) led2_in=1; else led2_in=0;
  if(IN3==0) led3_in=1; else led3_in=0;
  if(IN4==0) led4_in=1; else led4_in=0;
  if(IN5==0) led5_in=1; else led5_in=0;
  if(IN6==0) led6_in=1; else led6_in=0;
  if(IN7==0) led7_in=1; else led7_in=0;
  if(IN8==0) led8_in=1; else led8_in=0;
  led1_inTxFlag=1;
  led2_inTxFlag=1;
  led3_inTxFlag=1;
  led4_inTxFlag=1;
  led5_inTxFlag=1;
  led6_inTxFlag=1;
  led7_inTxFlag=1;
  led8_inTxFlag=1;
}  // mtConnect()
```



Rysunek 8. Przypisanie zmiennych do przycisku i diody LED na stronie sterowania przekaźnikami

Rysunek 9. Strona z wyświetlonym stanem wejść

Rysunek 10. Strona konsoli do wprowadzani komend

na linie portów sterujących przekaźnikami. RTOS sterujący serwerem ma specjalną funkcję mtUserInit, wykonywaną po zerowaniu mikrokontrolera (**listing 3**).

Problemu zaniku sterowań, szczególnie po ponownym włączeniu sterownika, nie można pominąć. W ważnych układach sterowania urządzenia sterowane same zapamiętują czy są załączone, czy nie. Sterownik wysyła wtedy tylko impuls sterowniczy o określonej długości np. 1…2 s i wyłączanie zasilania sterownika nie zakłóca pracy układu. W prostych układach przekaźnik jest albo ciągle włączony i załącza jakieś urządzenie, albo ciągle wyłączony. Po wyłączeniu zasilania sterownika wszystkie urządzenia zostają wyłączone. Żeby za każdym razem nie trzeba było włączać ich ręcznie, zastosowałam automatyczne odtwarzanie stanu sterowań po zerowaniu mikrokontrolera.

Kolejna strona jest przeznaczona do wyświetlania stanu 8 wejść binarnych (**rysunek 9**). Wybierana jest ze strony głównej po kliknięciu na przycisk „wejścia cyfrowe”. Kiedy przez diodę transoptora wejściowego płynie prąd 10…20 mA, to wirtualna dioda LED powiązana z tym wejściem ma kolor zielony. Brak sygnału aktywnego na wejściu jest sygnalizowany kolorem szarym. Po włączeniu zasilania i nawiązaniu połączenia pomiędzy serwerem (sterownikiem) i klientem (przeglądarką) bieżące stany są przesyłane przez funkcję mtConnect (**listing 4**).

Z każdym z wejść jest związana odpowiednia zmienna led1_in…led8_in i flaga zezwolenia na transmisję led1_inTxFlag… led8_inTxFlag. Sterownik cyklicznie, co

Listing 5. Funkcja testowania zmian na wejściach i ich przesyłania do przeglądarki
```
unsigned char wzor=0;
void mtEndOfMessage(void)
{
  if(wzor!=(PORTE&0xf0)) //testowanie czy stany wejściowe zmieniły się
  {
    wzor=(PORTE&0xf0);
    if(IN1==0) led1_in=1; else led1_in=0;
    if(IN2==0) led2_in=1; else led2_in=0;
    if(IN3==0) led3_in=1; else led3_in=0;
    if(IN4==0) led4_in=1; else led4_in=0;
    if(IN5==0) led5_in=1; else led5_in=0;
    if(IN6==0) led6_in=1; else led6_in=0;
    if(IN7==0) led7_in=1; else led7_in=0;
    if(IN8==0) led8_in=1; else led8_in=0;
    led1_inTxFlag=1;
    led2_inTxFlag=1;
    led3_inTxFlag=1;
    led4_inTxFlag=1;
    led5_inTxFlag=1;
    led6_inTxFlag=1;
    led7_inTxFlag=1;
    led8_inTxFlag=1;
}
```

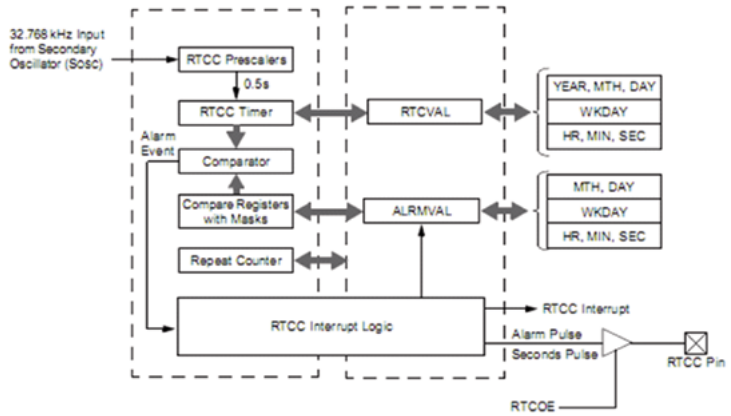
Listing 6. Funkcja obsługi konsoli tekstowej
```
// Event handler for control: It1
void eventIt1(void)
{
/* Zmienna została odebrana, niżej należy dodać kod związany z obsługą
polecenia: */
  unsigned char err;
  err=CmdPhase(It1);  //identyfikacja i obsługa funkcji konsoli
  if(err==1) memcpy(It1,"Blad formatu      ",18);
  if(err==2) memcpy(It1,"Poza zakresem     ",18);
  if(err==10) memcpy(It1,"Blad komendy      ",18);
  It1TxFlag=1;
}
```

Listing 7.Funkcja służąca do identyfikowania i wywoływania komend konsoli
```
char CmdPhase(char *buf) {
char err;
  switch (*buf)
  {
    case 't' :
      err=1;
      if(*(buf+1)=='s') err=CmdSetTime(It1); //ustawienie zegara
      if(*(buf+1)=='g') err=CmdGetTime(It1); //odczytanie zegara
      if(*(buf+1)=='c') err=CmdSetChannel(It1); //ustawianie programu kanału
przekaźnika
      if(*(buf+1)=='d') err=CmdGetChannel(It1); //wyswietlenie programu kanału
      if(*(buf+1)=='h') err=CmdHideTime(It1); //ukrywanie zegara
      return(err);
    case 'd' :
      err=1;
      if(*(buf+1)=='s') err=CmdSetData(It1); //ustawienie daty
      if(*(buf+1)=='g') err=CmdGetData(It1); //odczytanie daty
      return(err);
    default:
      return(10);//blad kodu komendy
  }
}
```

ok. 0,1 sekundy, odczytuje stany na liniach *RE0…RE7*. Jeżeli jedna lub więcej linii zmieni swój stan od ostatniego odczytania, to zmiana stanu zostaje wysłana do przeglądarki. Do tego typu działań projektanci RTOS przeznaczyli funkcję mtEndOfMessage pokazaną na **listingu 5**.

Strona wywoływana po kliknięciu ze strony głównej na belkę „ustawienia” zawiera konsolę do wprowadzania komend tekstowych (**rysunek 10**). Z elementem okna konsoli *Input Text* program TPCMaker wiąże bufor o wielkości zdefiniowanej w polu MaxStringLen. W naszym wypadku jest to It1[22]. Po wybraniu strony można wpisywać z klawiatury ciąg znaków ASCII o maksymalnej, zdefiniowanej długości (22 znaki). Naciśnięcie klawisza Enter kończy wpisywanie, a wpisany ciąg znaków zostaje przesłany przez sieć do sterownika i zapisany w buforze It1[]. Gdy program sterownika odbierze dane, to automatycznie jest wywoływana funkcja evetIt1() pokazana na **listingu 6**.

Z buforem It1 jest powiązana flaga It1xFlag zezwolenia na zwrotne przesłanie jego zwartości. Oznacza to, że program sterownika może zapisywać znakami ASCII bufor It1 i przesyłać go zwrotnie do przeglądarki. Przesłany tekst jest wyświetlany w oknie konsoli. W naszym programie ta właściwość



Rysunek 11. Schemat blokowy modułu RTCC

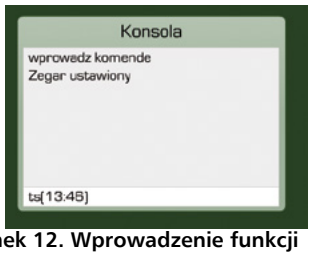
jest wykorzystywana do wyświetlania komunikatów zwrotnych wykonywanych funkcji i komunikatów o błędach.

Identyfikację i wywoływanie wprowadzanych komend wykonuje funkcja CmdPhase. Jako argument jej wywołania można przekazać wskaźnik do bufora It1 (**listing 7**).

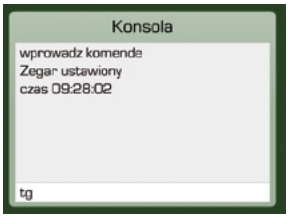
W obecnej wersji oprogramowania wszystkie komendy dotyczą zegara czasu rzeczywistego RTC i powiązania zegara z wyjściami sterującymi. W mikrokontrolerach PIC32MX jest wbudowany sprzętowy moduł zegara czasu rzeczywistego kalendarzem RTCC taktowany częstotliwością 32768 Hz. Na **rysunku 11** pokazano jego schemat blokowy. Sekcja zegara zawiera licznik godzin, minut i sekund. Sekcja kalendarza zawiera rejestry roku (zakres 2000…2099), miesiąca, dnia miesiąca i dnia tygodnia (poniedziałek wtorek itd.). Wszystkie rejestry zliczają w kodzie BCD. Zegar ma wbudowaną funkcję kalibrowania częstotliwości oscylatora (zakres 260 ppm). Można też zaprogramować funkcję alarmu. W naszym projekcie wykorzystamy funkcje zegara i kalendarza, a alarmy będą realizowane przez program sterownika.

Pierwszą opisywaną funkcją będzie ustawianie zegara ts (HH:MM). Po wpisaniu kodu komendy ustawiania ts w nawiasach okrągłych trzeba wpisać na pasku konsoli dwucyfrowo godzinę i minutę rozdzielone dwukropkiem, na przykład ts(13:46), jak na **rysunku 12**. Naciśnięcie klawisza Enter skutkuje wysłaniem ciągu ts(13:46). Po jego odebraniu sterownik zapisuje go w buforze It1 i za pomocą funkcji CmdPhase analizowane są dwa pierwsze znaki z tego bufora. Jeżeli jest to „ts", to jest wywoływana funkcja CmdSetTime (**listing 8**). Analizuje ona poprawność składni komendy i jeżeli wszystko jest w porządku, to kody ASCII cyfr godzin i minut są zamieniane na wartości w kodzie BCD. Kolejne sprawdzanie dotyczy ustawianych wartości. Jeżeli liczniki godzin mają wartości większe niż 23, a minut większe niż 59, to funkcja kończy działanie i zwraca błąd „poza zakresem".

Prawidłowo zapisane liczniki są wpisywane do rejestrów RTCC poprzez funkcję biblioteczną RtccSetTime. W momencie ustawiania zegara jest automatycznie zerowany licznik sekund. Zegar można odczytać i wyświetlić w oknie konsoli funkcją tg bez żadnych argumentów (**rysunek 13**). Podobnie



Rysunek 12. Wprowadzenie funkcji ustawiania zegara.



Rysunek 13. Odczytanie czasu

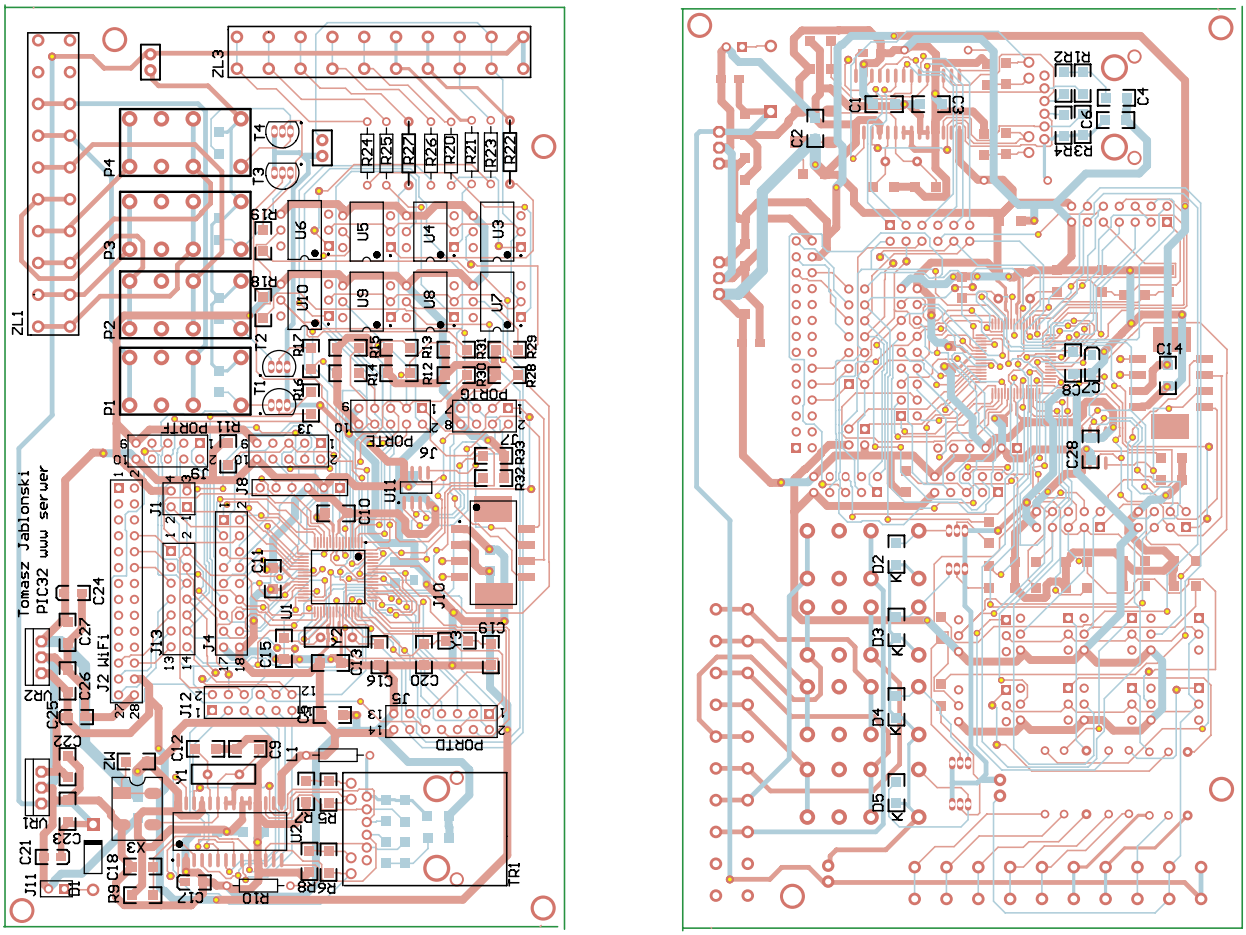
Rysunek 14. Programowanie kanału 2

Listing 8. Funkcja ustawiania zegara RTCC

```
//komenda ustawienie zegara
char CmdSetTime(char *buffer)
{
  unsigned char bcd;
  //sprawdzanie poprawności syntaktycznej komendy
  if(*(buffer+1)!='s') return(1);
  if(*(buffer+2)!='(') return(1);
  if(*(buffer+5)!=':') return(1);
  if(*(buffer+8)!=')') return(1);
  if(atoi(buffer+3)>23) return(2);
  if(atoi(buffer+6)>59) return(2);
  //konwersja bin/bcd i wpisanie godziny i minuty  do struktury czasu tm
  bcd=*(buffer+3)-0x30;
  bcd<<=4;
  bcd=bcd|(*(buffer+4)-0x30);
  tm.hour=bcd;
  bcd=*(buffer+6)-0x30;
  bcd<<=4;
  bcd=bcd|(*(buffer+7)-0x30);
  tm.min=bcd;
  tm.sec=0;  //wyzerowanie licznika sekund
  RtccSetTime(tm.l);      // ustawienie  czasu
  memcpy(It1,"Zegar ustawiony „,16);
  PosLcd(1,2);
  DispLcd(„CMD ustaw zegar „);
  return(0);
}
```

Listing 9. Funkcja ustawiania daty

```
char CmdSetData(char *buffer)
{
  unsigned char bcd;
  //sprawdzanie poprawności syntaktycznej komendy
  if(*(buffer+1)!='s') return(1);
  if(*(buffer+2)!='(') return(1);
  if(*(buffer+5)!=':') return(1);
  if(*(buffer+8)!=':') return(1);
  if(*(buffer+11)!=')') return(1);
  if(atoi(It1+3)>90) return(2);
  if(atoi(It1+6)>12) return(2);
  if(atoi(It1+9)>31) return(2);
  //konwersja ascii/bcd
  bcd=*(buffer+3)-0x30;
  bcd<<=4;
  bcd=bcd|(*(buffer+4)-0x30);
  dt.year=bcd;  //wpisanie roku (2 cyfry)
  bcd=*(buffer+6)-0x30;
  bcd<<=4;
  bcd=bcd|(*(buffer+7)-0x30);
  dt.mon=bcd;  //wpisanie miesiąca
  bcd=*(buffer+9)-0x30;
  bcd<<=4;
  bcd=bcd|(*(buffer+10)-0x30);
  dt.mday=bcd;  //wpisanie dnia miesiąca
  RtccSetDate(dt.l);  // ustawienie liczników modułu RTCC
  memcpy(It1,"Data ustawiona ",16);
  PosLcd(1,2);
  DispLcd("CMD ustaw data  ");
  return(0);
}
```



Rysunek 15. Schemat montażowy płytki mikrokontrolera i PHY

przebiega ustawienie daty i jej odczytanie. Po zidentyfikowaniu odbioru polecenia „ds”, funkcja CmdPhase wywołuje CmdSetData (**listing 9**). Datę wprowadza się podobnie jak czas. Po kodzie komedy „ds” w nawiasach okrągłych wpisuje się rok, miesiąc i dzień, na przykład ds(12:08:14), czyli 14 sierpnia 2012 roku. Datę można za pomocą „dg”.

Sterownik ma wbudowaną możliwość sterowania czterema przekaźnikami. Razem z zegarem czasu rzeczywistego daje to możliwość programowania czasowego załączania i wyłączania każdego z przekaźników – kanałów sterujących. Każdy z kanałów jest programowany komendą „tcx”, gdzie x jest numerem kanału 1…4. Argumentami tej komendy umieszczonymi w nawiasach okrągłych są: tryb pracy kanału, czas (godzina i minuta) załączenia kanału i czas (godzina i minuta) wyłączenia kanału. Tryb pracy określa czy kanał ma być wyłączony „W”, sterowanie czasowe ma się wykonać tylko raz „R” , czy co 24 godziny „C” do zmiany trybu na inny. Na przykład wysłanie komendy tc2(R,12:00,14:30) spowoduje, że przekaźnik numer dwa o godzinie 12:00 zostanie załączony, a o 14:30 wyłączony. Po wyłączaniu program zmieni automatycznie tryb na „W” i sterowanie w tym kanale się już więcej nie wykona, chyba, że przeprogramujemy tryb na „R” lub „C”. Funkcja ustawiania kanału jest podobna do ustawiania czasu i daty. Zaprogramowane tryby pracy i czasy są zapisywane w pamięci eeprom i odtwarzane po włączeniu zasilania. Na listingu 10 są pokazane procedury testowania kanałów i wykonania sterowania.

Wszystkie dane potrzebne do działania funkcji sterowania kanałami są zapisane w tablicy struktur – listing 11

Montaż i uruchomienie

Montaż płytki sterownika (**rysunek 15**) nie powinien nastręczać większych trudności . Lutowanie elementów w technice montażu powierzchniowego było już wielokrotnie opisywane. Ja z powodzeniem stosuję stację lutowniczą z grotem mini-fala i topnik w żelu. Po zmontowaniu całości trzeba podłączyć zasilanie. Układ mikroprocesorowy z układem PHY w czasie pracy pobiera ze źródła zasilania prąd o natężeniu ok. 250 mA. Przy napięciu zasilającym +12 V niezbędny był niewielki radiator na stabilizatory VR1 i VR2.

Żeby ułatwić sobie uruchomienie układu, zastosowałem w prototypie wyświetlacz LCD 2×16 znaków Jest on sterowany w trybie magistrali 4-bitowej z linii portu PORTD (J5). Linie danych to RD0…RD3, linia RS jest połączona z portem RD4, a linia EN z portem RD5. Sygnał RW (wyboru kierunku przepływu danych) jest połączony na stałe z masą.

Zmontowany układ wymaga zaprogramowania mikrokontrolera. Do tego celu przewidziano złącze J8 z wyprowadzeniami zgodnymi z programatorem PICKit3. Jak wykorzystałem do programowania mój stary programator ICD2 z przejściówką .

Pomimo że Microchip nie zleca stosowanie tego narzędzia w nowych projektach, to MPLAB IDE podpowiadał w zakładce wyboru mikrokontrolera, że powinienem programować i debugować ten typ mikrokontrolera. Niestety, nie było to możliwe, bo miałem starą wersję firmware. Po konsultacjach z firmowym wsparciem technicznym przysłano mi bezpłatnie drugi działający programator, a mój odesłałem do centrum serwisowego

Listing 10. Testowanie kanałów i procedury sterowania

```
void CheckChannel(char channel)
{
  tml.l=RtccGetTime();
  if(tml.hour==tch[channel].hour_on&&tml.min==tch[channel].min_on)
  CtrlOn(channel);
  if(tml.hour==tch[channel].hour_off&&tml.min==tch[channel].min_off)
  {
    CtrlOff(channel);
    if(tch[channel].mode=='R')
    {
      //wyłacz kanał po wykonaniu sie sterowania
      EEWrite('W',CHAN_MODE+channel);
      tch[channel].mode='W';
    }
  }
}

void CtrlOn(char ctrl)
{
  switch (ctrl)
  {
    case 0:
      STER1=1;
      break;
    case 1:
      STER2=1;
      break;
    case 2:
      STER3=1;
      break;
    case 3:
      STER4=1;
      break;
  }
}

void CtrlOff(char ctrl)
{
  switch (ctrl)
  {
    case 0:
      STER1=0;
      break;
    case 1:
      STER2=0;
      break;
    case 2:
      STER3=0;
      break;
    case 3:
      STER4=1;
      break;
  }
}
```

Listing 11. Wygląd definicji tablicy struktur kanałów

```
struct channel{
  unsigned char mode;  //tryb kanału
  unsigned char hour_on;  //czas załaczenia
  unsigned char min_on;
  unsigned char hour_off;  //czas wyłączenia
  unsigned char min_off;
}tch[4];
```

w Irlandii. Po raz kolejny okazało się, że firma bardzo dba o swoich klientów.

Plik wynikowy do programowania pamięci zwiera definicje bitów konfiguracyjnych i nie trzeba się nimi zajmować.

Zmontowany i zaprogramowany podłączamy do sieci LAN i zasilamy. Po włączeniu zasilania na wyświetlaczu pojawia się napis powitalny *PIC32MX WWW.serwer*, a potem sterownik odczytuje z pamięci EEPROM ustawienia przekaźników zapamiętane przy ostatnim sterowaniu. Jeżeli jest to pierwsze uruchomienie po zaprogramowaniu, to wszystkie przekaźniki są wyłączane.

Jeżeli wszystko zostało poprawnie zainicjowane w następnym kroku sprawdzamy działanie w sieci LAN. Prawidłowe połączenie z siecią jest sygnalizowane ciągłym świeceniem zielonej diody LED i miganiem w takt przesyłanych danych diody żółtej w zintegrowanym transformatorze 08B0--1X1T-06-F. Program sterownika ma wbudowany protokół DHCP i sam sobie pobiera informacje o przydzielonym adresie IP, bramie i masce podsieci. Przydzielenie adresu można sprawdzić w menu konfiguracyjnym routera sieci zazwyczaj dostępnym poprzez wbudowana stronę WWW. Można ten krok pominąć i przejść od razu do sprawdzenia działania serwera. Otwieramy dowolną przeglądarkę i w pasku adresu wpisujemy domyślny adres http://mchpborad. Po nawiązaniu połączenia serwer – klient (przeglądarka) miga żółta dioda LED sygnalizując ciągłe przesyłanie danych.

Sprawdzenie sterowania polega na sekwencyjnym przyciskaniu kolejnych przycisków załączających przekaźniki. Każde przyciśniecie przycisku musi spowodować sekwencyjne załączenie/wyłączenie przekaźnika i sygnalizacje stanu poprzez zapalenie/gaszenie przypisanej mu diody LED. Dodatkowo, jeżeli mamy zamontowany wyświetlacz LCD, to będą się na nim pojawiały komunikaty „Ster-> PRZ1 (ZAL)". Na tym etapie można też sprawdzić działanie inicjalizacji po włączeniu zasilania. Załączamy na przykład przekaźniki przekaźnik 1 i przekaźnik4 i wyłączamy zasilanie serwera. Po ponownym zasileniu serwer powinien załączyć te przekaźniki ponownie w procedurze inicjalizacji bez ingerencji użytkownika.

Testowanie przesyłania stanów wejść binarnych polega na podawaniu na wejścia napięcia +12 V o odpowiedniej polaryzacji. Wejścia, na które jest doprowadzone napięcie powinny zasygnalizować to „zaświeceniem się" diody LED umieszczonej na podstronie „Stan wejść". Wejścia nieaktywne są szare. Testowanie działania konsoli zaczynamy od ustawienia czasu i daty. Serwer powinien mieć ciągłe zasilanie, bo każdy zanik powoduje utracenie ustawień zegara RTCC. Zgodnie z wcześniejszym opisem wprowadzamy komendy ustawienia zegara i daty a potem komendy ustawień kanałów sterowniczych.

Uwagi końcowe

Prezentowane tu urządzenie zostało uruchomione i przetestowane. Możliwość sterowania 4 torami i przesyłania stanu 4 wejść dwustanowych można bez problemu rozszerzyć na większą ich liczbę. Ograniczeniem jest tylko liczba wolnych linii I/O. Serwer może tez przesyłać pomiary analogowe. TCPMaker pozwala na skalowanie i odpowiednie wysiedlanie wartości analogowych. W modelowym rozwiązaniu pomiary nie są przesyłane, bo nie miałem takiej potrzeby.

Układ po włączeniu pracuje w lokalnej sieci LAN. Ponieważ są obsługiwane wszystkie niezbędne protokoły TCP/IP nic nie stoi na przeszkodzie, aby dostęp do funkcji serwera był możliwy za pomocą Internetu. Tu jednak potrzebna jest dodatkowa konfiguracja routera. Szczegóły konfiguracji zależą od modelu routera i nie będę jej szczegółowo opisywał. Zazwyczaj trzeba wymusić, aby dla adresu MAC naszego serwera router przydzielał w lokalnej sieci zawsze ten sam lokalny adres IP. Potem dla tylko tego adresu i konkretnego portu (np. 8080) konfiguruje się dodatkowo router, tak aby protokół NAT „podmieniał" adresy w taki sposób, aby spoza naszej sieci serwer był widziany pod adresem routera. Trzeba tę konfiguracje wykonać starannie, by nie ułatwić hakerom opanowania innych komputerów w domowej sieci.

Użytkownicy popularnych domowych systemów dostępu do Internetu nie mają stałego adresu IP. Przy każdym łączeniu się z siecią adres jest przydzielany dynamicznie. W takim wypadku trzeba znać przydzielony adres odczytując go z routera. Jest to dość uciążliwe i można temu próbować zapobiec przez ciągłe zasilanie domowego routera. Osobiście wykonałem szereg zakończonych sukcesem prób łączenia i sterowania serwerem poprzez Internet.

Tomasz Jabłoński, EP