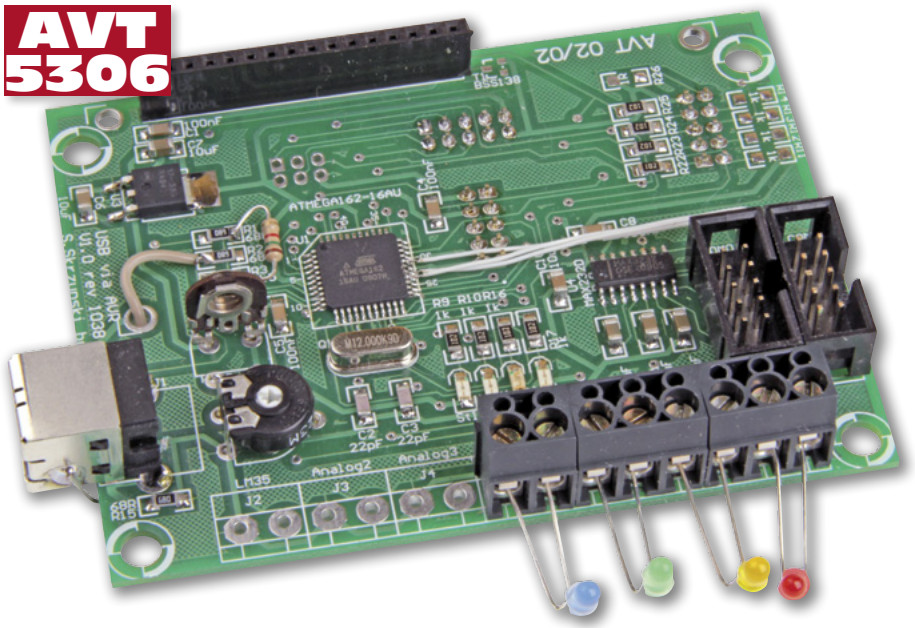
Programowe USB w AVR

Terminal znakowy z interfejsem USB

Interfejs USB niemal zupełnie wyparł z użycia inne rodzaje interfejsów urządzeń peryferyjnych, w które były wyposażone urządzenia dołączane do komputerów PC, takie jak: drukarki, modemy, kamery i inne. W związku z tym, że producenci PC przestali wyposażać je w interfejsy szeregowe RS232 czy równoległe Centronics, to aby dołączyć „mały" mikrokontroler do komputera PC, jest wymagany konwerter USB na inny rodzaj transmisji (RS232, równoległa magistrala danych itp.). Niestety, te układy nie dosyć, że zajmują miejsce na płytce, to nie należą do najtańszych. W artykule zaprezentowano rozwiązanie umożliwiające dołączenie mikrokontrolera AVR do USB w trybie device z wykorzystaniem jedynie trzech rezystorów i odpowiedniego programu.

***Rekomendacje:** oprócz użytkowych, projekt ma walory edukacyjne; polecamy lekturę wszystkim, którzy borykają się z problemem dołączenia mikrokontrolera AVR bez sprzętowego interfejsu USB do komputera PC.*



W EP wielokrotnie pojawiały się projekty, w których do mikrokontrolera z rodziny AVR był dołączony interfejs USB. Jednak przeważnie był stosowany układ konwertera (np. w programatorze z EP 2/2008), tani zamiennik układu FT232R. Czasem nie ma innego wyjścia, ale bardzo często procesor użyty w aplikacji poradzi sobie z programową obsługą USB w trybie *device*.

Pomysł rozwiązania opisywanego w artykule został zaczerpnięty ze strony internetowej *http://www.recursion.jp/avrcdc/*. Do projektu dodano kompletne kody źródłowe, dzięki czemu jest możliwe użycie opisywanego rozwiązania w jakimś własnym projekcie. Modyfikując program, można też zbudować konwerter USB-I²C czy USB-SPI lub inne.

Na wstępie zaznaczę, że aby AVR mógł komunikować się przez USB, muszą być spełnione pewne wymagania. Pierwsze to dostępność zasobów. Procedury obsługi USB zajmują około 3,5 kB pamięci programu i około 200 bajtów pamięci operacyjnej. Zajmowana jest linia przerwania zewnętrznego INT0 oraz jedna lub dwie linie I/O portu, do którego przynależy wyprowadzenie INT0. Kolejna rzecz to częstotliwość taktowania mikrokontrolera, która musi wynosić 12, 16 lub 16,5 MHz. Procedura *usbPoll()* musi być wywoływana nie rzadziej niż co 50 ms. Inne przerwania obsługiwane przez oprogramowanie muszą być deklarowane jako *interrupt*, a nie *signal* czy *isr*, które blokują możliwość wykonywania przerwania o wyższym priorytecie (INT0) podczas obsługi przerwania o priorytecie niższym. W programie nie wolno blokować obsługi przerwań, a więc mogą pojawić się kłopoty z zapisem niektórych rejestrów (np. z obsługą *WDG* czy *EEPROM*).

AVT-5306 w ofercie AVT:
AVT-5306A – płytka drukowana
AVT-5306B – płytka drukowana + elementy

Podstawowe informacje:
- Płytka dwustronna o wymiarach 92 mm × 69 mm
- Mikrokontroler ATmega162-16AU
- Wyświetlacz znakowy 16×2
- Gniazda RS232, USB, złącze modułu Ethernet
- Komunikacja przez USB bez użycia układów konwerterów
- Procedury i intefejs do wykorzystania we własnych aplikacjach

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 12040, pass: 15735862
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

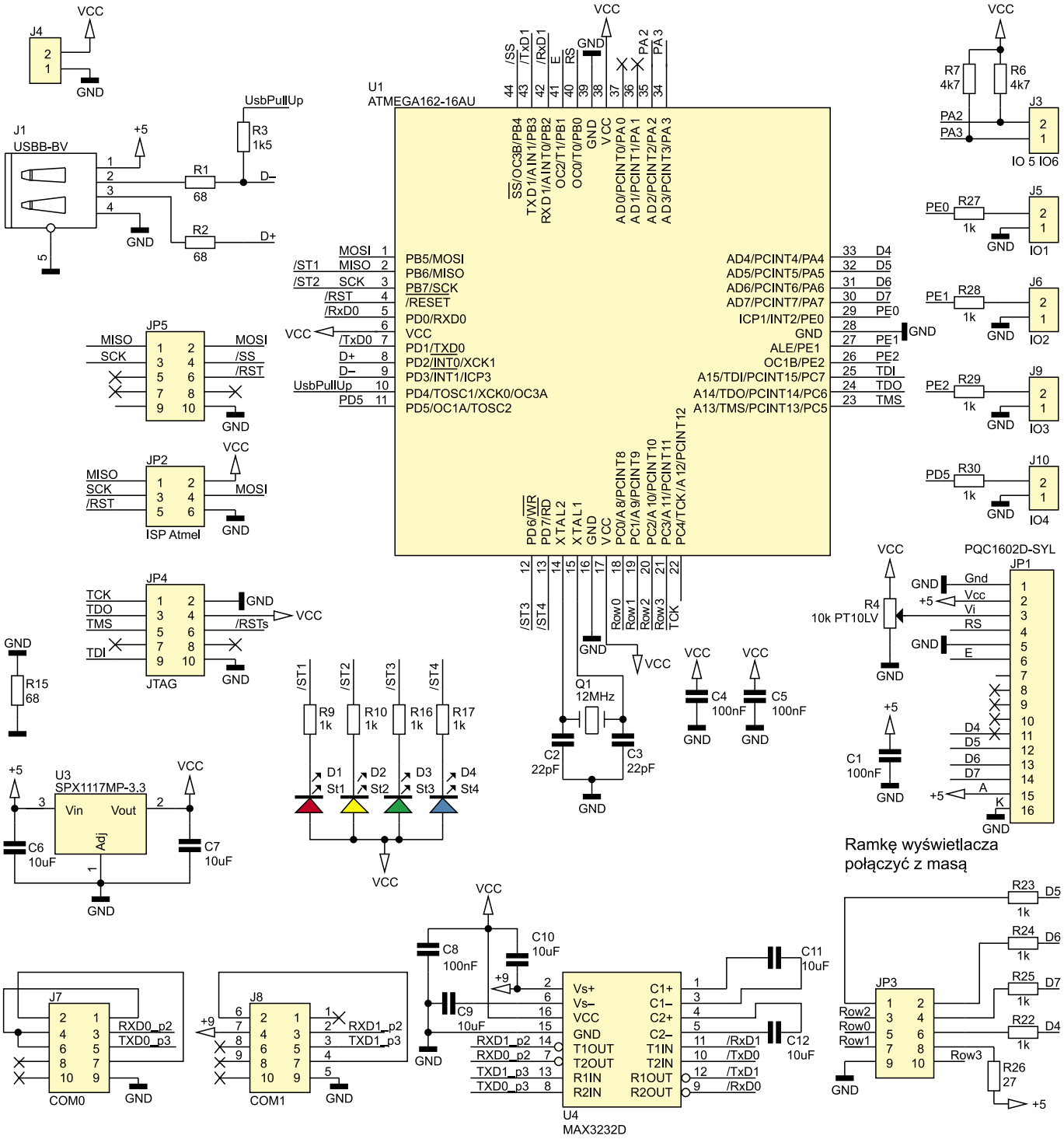
Projekty pokrewne na CD/FTP:
(wymienione artykuły są w całości dostępne na CD)

AVT-5272	AVTDuino (EP 1/2011)
AVT-1622	Minimoduł z Atmega8 (EP 6/2011)
AVT-5140	Konwerter USB-IO (EP 7/2008)
AVT-1595	Miniaturowy konwerter USB/UART (EP 10/2010)

Płytka testowa

Do zaprezentowania możliwości implementacji interfejsu USB w mikrokontrolerze AVT zaprojektowałem płytkę, na której umieszczono:

- interfejs USB zbudowany za pomocą trzech rezystorów i gniazda,



Rysunek 1. Schemat ideowy płytki testowej

- stabilizator napięcia 3,3 V zasilany ze złącza USB,
- mikrokontroler ATmega161 z interfejsami JTAG, SPI oraz 4 diody sygnalizacyjne LED,
- interfejs wyświetlacza LCD (złącze i otwory przygotowane pod wyświetlacz 2×16),
- interfejs klawiatury matrycowej 4×4,
- dwa interfejsy RS232 (COM1 przystosowany do dołączenia zegara DCF77),
- 4 cyfrowe interfejsy I/O,
- interfejs I²C,
- interfejs Ethernet (przystosowany do modułu AVT-1528).

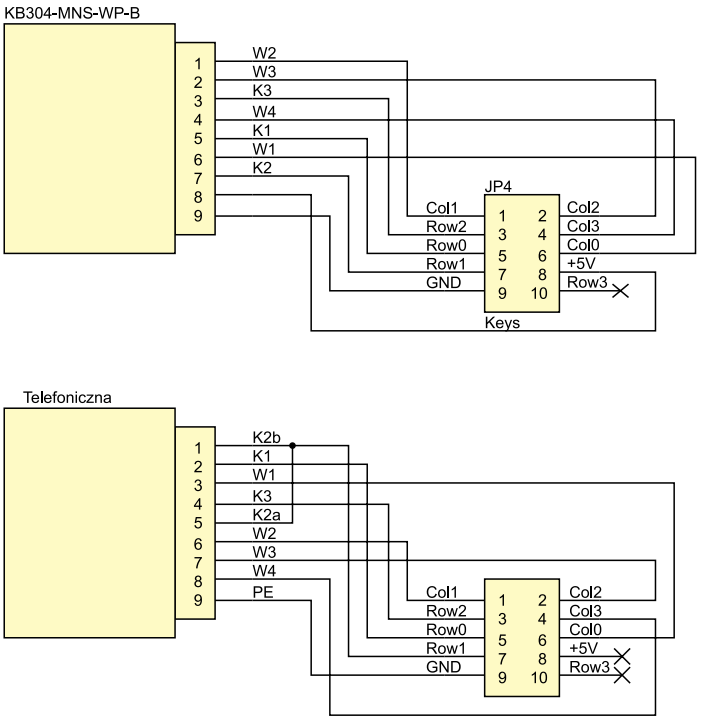
Interfejsu Ethernet nie zamontowano tam przypadkiem. Opisywane płytka będzie wykorzystana do zademonstrowania obsługi protokołów komunikacyjnych UDP i TCP (protokoły ICMP, ARP, HTTP), których implementacja będzie opisana w którymś z kolejnych numerów EP jako tania alternatywa dla modułów Ethernet.

Schemat płytki testowej pokazano na **rysunku 1**. Jest ona nieskomplikowana, podobna do innych płytek tego typu i nie wymaga jakiegoś szczególnego opisywania. Na warstwę fizyczną interfejsu USB składają się 3 rezystory (R1...R3) oraz złącze. Rezystor R3 dołączono do portu mikrokontrolera, dzięki czemu można symulować wyjęcie i włożenie wtyczki USB do komputera. We własnych rozwiązaniach, gdy zastosujemy mikrokontroler o małej liczbie wyprowadzeń, rezystor ten można na stałe dołączyć do napięcia zasilania, pamiętając o odpowiedniej deklaracji *#define* w programie źródłowym. Ze względu na to, że poziomy sygnałów USB nie powinny przekraczać 4 V, zdecydowano się na zasilanie CPU ze stabilizatora napięcia 3,3 V. Zastosowano tani stabilizator SMD typu SPX1117-3.3 Dzię-

ki temu nie trzeba wykonywać konwersji napięć na liniach USB z 5 na 3,3 V. W naszym wypadku rozwiązanie to ma pewną wadę – jest nią problem z zakupem wyświetlacza LCD zasilanego z 3,3 V. Ze względu na to, że zdecydowałem się na jednokierunkową transmisję danych w kierunku do LCD (nie jest sprawdzana flaga busy), linie danych LCD można dołączono bezpośrednio do CPU, natomiast sam wyświetlacz jest zasilany z 5 V. Pierwotnie popełniłem błąd, podłączając potencjometr regulacji kontrastu pomiędzy masę a 5 V. W konsekwencji kontrast LCD „pływał", choć zależało to w głównej mierze od typu płyty głównej komputera. W zaproponowanym tu rozwiązaniu potencjometr jest włączony pomiędzy masę a stabilizowane napięcie 3,3 V, dzięki czemu kontrast wyświetlacza jest stabilny. Przewidziano też możliwość dołączenia klawiatury 4×4 z podświetlaniem – służy do tego złącze JP3. Klawiatura i LCD mają wspólne linie danych D4...D7. Aby przypadkowe naciśnięcie kilku klawiszy klawiatury nie zakłócało pracy wyświetlacza, zastosowano rezystory separujące R22...R25. Sposoby dołączenia różnych klawiatur pokazano na **rysunku 2**.

Rezystor R26 ogranicza prąd diod podświetlających klawiaturę. Porty UART za pośrednictwem konwertera poziomów TTL/RS232 wyprowadzono na złącza J7 i J8. Zastosowano układ konwertera MAX3232 zasilany napięciem 3,3 V, a nie MAX232 lub odpowiednik zasilany napięciem 5 V.

Na złączu J7 (COM0) zapętlono wszystkie linie potwierdzeń, aby można je było połączyć zwykłym kablem typu *nullmodem*. J8 ma zapętlone tylko linie DTR i DSR ze względu na to, że przewidziano je także do współpracy z odbiornikiem DCF77, co spowodowało konieczność doprowadzenia napięcia



Rysunek 2. Schemat ideowy klawiatury

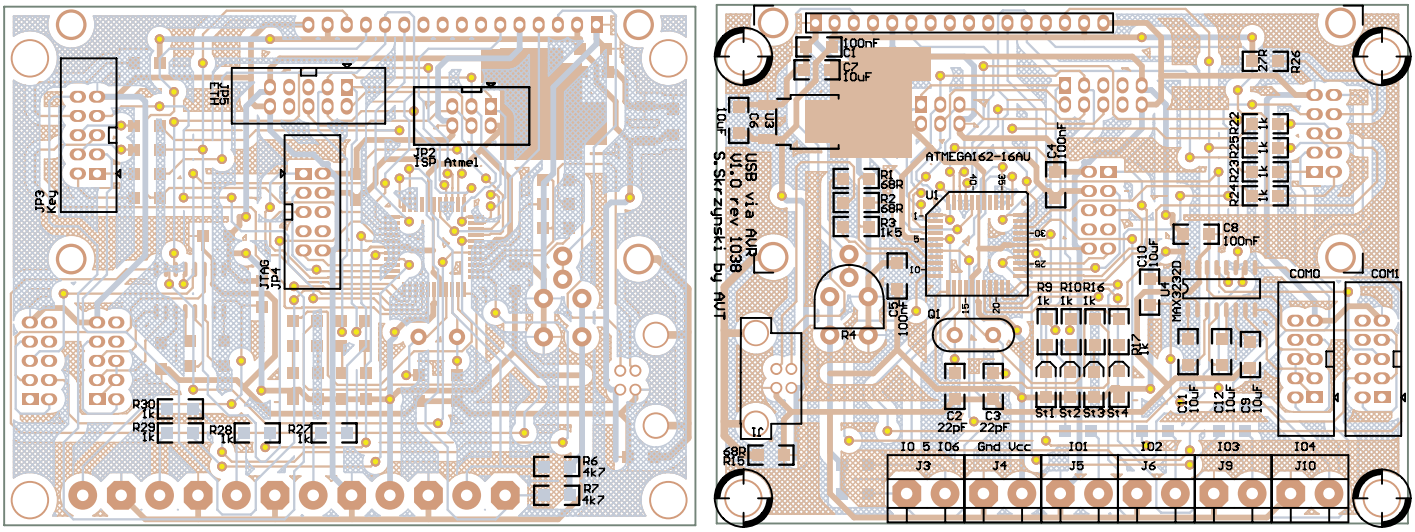
Tabela 1. Aktualne funkcje diod

Oznaczenie diody	Funkcja
D1 (czerwona)	Miga w czasie pracy.
D2 (żółta)	Gaśnie po naciśnięciu dowolnego klawisza na klawiaturze, zaświeca się po jego zwolnieniu.
D3 (zielona)	Świeci się w czasie pracy.
D4 (niebieska)	Zaświeca się po przyjęciu pierwszego znaku rozkazu (apostrof), gaśnie po odebraniu całego rozkazu.

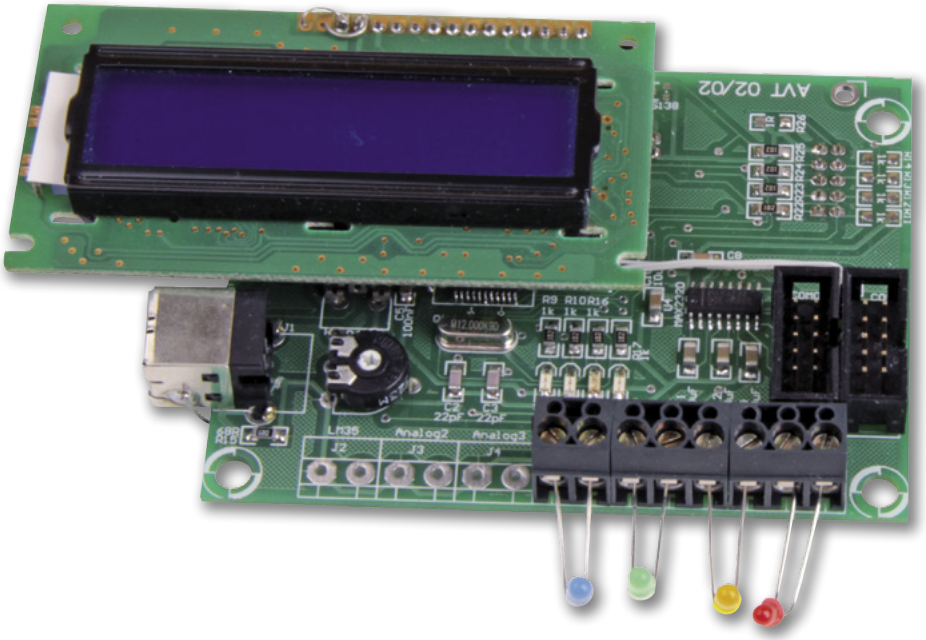
+9 V na linię RTS. Na złącza J5, J6 J9, J10 wyprowadzono porty mikrokontrolera za pośrednictwem rezystorów zabezpieczających R27...R30. Diody LED D1...D4 sygnalizują stan pracy mikrokontrolera. Po zmianach w programie można wykorzystać je w inny sposób. Opis sposobu sygnalizowania różnych stanów urządzenia za pomocą diod LED zamieszczono w **tabeli 1**.

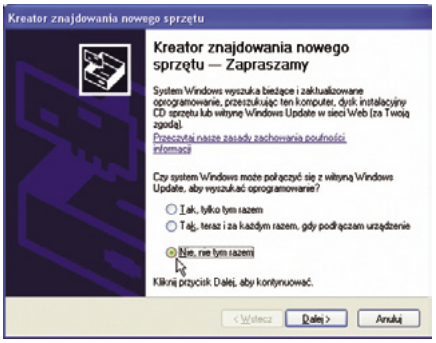
Montaż

Schemat montażowy płytki testowej pokazano na **rysunku 3**. W celu zmniejszenia gabarytów płytki złącza do programowania mikrokontrolera, klawiatury i interfejsu Ethernet umieszczono pod wyświetlaczem, a więc z konieczności od spodu płytki. Sygnały interfejsu RS232 doprowadzono do złączy IDC, dzięki czemu płytka jest mała. Jako złącza RS232 należy użyć wtyków

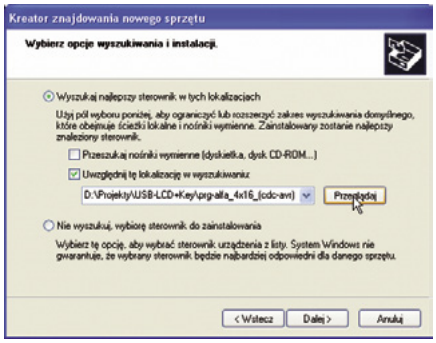


Rysunek 3. Schemat montażowy płytki testowej





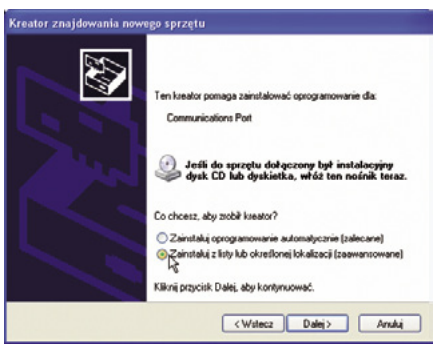
Rysunek 4. Okro kreatora wyszukiwania nowego sprzętu



Rysunek 6. Wybór sterownika

DB9M zaciskanych na taśmie przewodów, którą z drugiej strony zakończamy zaciskanym wtykiem FT10. Pod wyświetlacz należy wlutować gniazdo goldpin 1×16, natomiast w wyświetlacz wlutować listwę goldpin 1×16. Dzięki temu wyświetlacz można łatwo zdemontować, aby przeprowadzić pomiary sygnałów na mikrokontrolerze podczas uruchamiania urządzenia.

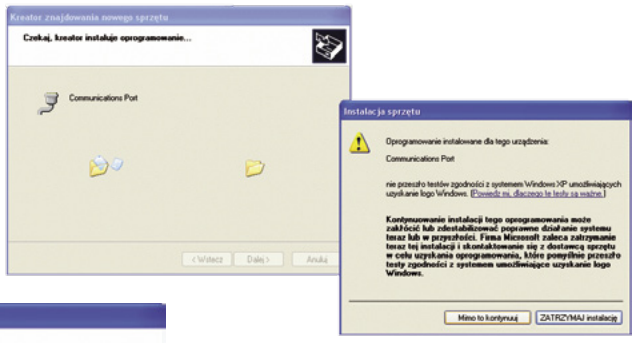
W roli klawiatury można zastosować KS304 lub telefoniczną, którą łączymy według schematu z rysunku 3.



Rysunek 5. Okno instalowania sterownika

Uruchomienie

Uruchomienie polega na kontroli napięcia zasilania i ustawieniu kontrastu wyświetlacza. Jeśli montaż jest prawidłowy, to po włączeniu zasilania i zaprogramowaniu mikrokontrolera na wyświetlaczu ukaże się komunikat zawierający numer wersji oprogramowania oraz imię i nazwisko autora. Dodatkowo, w drugim wierszu będzie wyświetlony migający kursor. Komputer (jeśli pracuje pod kontrolą sytemu system Windows XP) powinien poinformować o odnalezieniu nowego urządzenia USB i uruchomić kreatora, znajdowania nowego sprzętu (**rysunek 4**).



Rysunek 7. Instalowanie niepodpisanego sterownika

Wykaz elementów

Rezystory: (SMD 1206)
R,R2, R15: 68 Ω
R3: 1,5 kΩ
R4: 10 kΩ potencjometr
R6, R7: 4,7 kΩ
R9, R10, R16, R17, R22...R25, R27...R30: 1 kΩ
R26: 27 Ω
Kondensatory: (SMD 1206)
C1, C4, C5, C8: 100 nF
C2, C3: 22 pF
C6, C7, C9...C12: 10 µF
Półprzewodniki:
D1...D4: dioda LED (SMD 1206)
U1: ATmega162-16AU (PQFP44)
U3: SPX1117MP-3.3 (SOT-223)
U4: MAX3232D (SO-16)
Inne:
J1: gniazdo USB-B
JP3...JP5, J3...J6, J9, J10: ARK2
J7, J8: wtyk do druku IDC10
JP1: wyświetlacz LCD 16x2 (np. PQC1602D-SYL)
JP2: wtyk do druku IDC
Q1: rezonator 12 MHz (HC49S)

W kreatorze klikamy na przyciski „Nie, nie tym razem" i następnie „Dalej". W następnym oknie należy zaznaczyć opcję „Zainstaluj z listy lub określonej lokalizacji" po czym nacisnąć „Dalej" (**rysunek 5**).

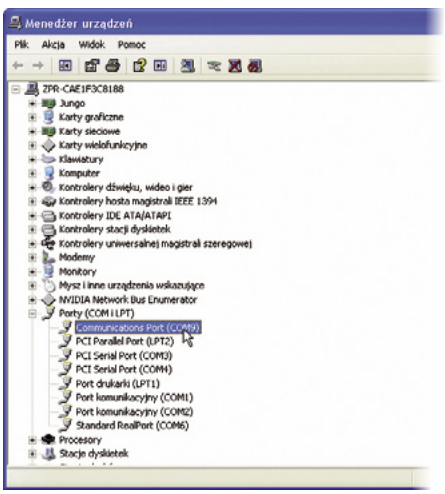
W kolejnym okienku zaznaczamy „Uwzględnij tę lokalizację w wyszukiwaniu" i naciskamy „Przeglądaj" (**rysunek 6**).

W oknie wyboru wskazujemy katalog, w którym znajduje się plik „avrcdc.inf". Plik ten jest dostępny w archiwum klasy CDC pod adresem *http://www.recursion.jp/avrcdc/avrcdc_inf.zip* oraz w materiałach źródłowych do artykułu. Po wskazaniu pliku pojawi się informacja o braku podpisu cyfrowego. Wybieramy „Mimo to kontynuuj" (**rysunek 7**).

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Tabela 2. Komendy realizowane przez oprogramowanie terminalu na płytce testowej

Komenda	Funkcja	uwagi
‘v	Wersja programu	
‘V	Nazwa programu	
‘c	Czyszczenie wyświetlacza	
‘h	Powrót kursora na pozycję 00	
‘s	Ustawienie kursora	Po komendzie należy podać dziesiętnie, dwucyfrowo pozycję w pamięci CGRAM wyświetlacza według wzoru: XY = PozycjaX + PozycjaY * 16, np.: ‘s03 pierwszy wiersz, kolumna 4, ‘s16 drugi wiersz kolumna 0
‘b	Włącz kursor	
‘B	Wyłącz kursor	
‘D	Ustawienie kierunku linii cyfrowych (zapis rejestru DDRx w AVR)	Po komendzie należy podać dwie cyfry: nr linii (zakres 1...4) oraz jej kierunek (1 wyjście, 0 wejście) np.: ‘D11 linia 1 wyjściem ‘D10 linia 1 wejściem ‘D21 linia 2 wyjściem Ustawienie linii jako wejściowej powoduje automatyczne włączenie podciągania. Podciąganie można wyłączyć komendą „‘o”
‘d	Odczyt kierunku linii (odczyt rejestru DDRx)	W odpowiedzi otrzymamy: ‘dABCD, gdzie: ABCD kierunek kolejnych linii
‘i	Odczyt linii wejściowych (odczyt rejestru PINx AVR-a)	Reguły jak dla komendy „‘d”
‘o	Ustawienie wyjścia cyfrowego (zapis rejestru PORTx w AVR)	Reguły jak dla komendy „‘D”
‘O	Odczyt rejestru linii cyfrowych (odczyt rejestru PORTx w AVR)	Reguły jak dla komendy „‘d”
‘k	Odczyt kodu klawisza z bufora klawiatury	Poza tym że naciskane znaki są wysyłane na USB w postaci kodów: ‘kx, gdzie x – znak ASCII z klawiatury są one dodatkowo gromadzone w buforze klawiatury (7 znaków). Kolejne wydawanie komendy „‘k” odczytuje jeden znak z bufora. Możliwe odpowiedzi: ‘kx, gdzie x = 1..9, 0, *, #, A, B, C, D oraz: ‘k–, bufor pusty, nie ma więcej znaków do odczytania ‘k? , uszkodzenie klawiatury (np. naciśnięto kilka klawiszy naraz)
‘I	Init wyświetlacza LCD	UWAGA! nadpisuje generator znaków w RAM i wyłącza kursor
‘R	Restart urządzenia	
‘U	Inicjalizacja interfejsu USB (efekt taki sam jak wyjęcie i włożenie wtyczki USB)	
Kody błędów (kody zwracane przez urządzenie)		
‘Synx	Błąd składni lub nieistniejący rozkaz	
Uwagi		
Po resecie procesora naciskane klawisze, poza wysyłaniem ich po USB, są także wyświetlane na LCD (tryb testu klawiatury). Po odebraniu jakiegokolwiek znaku po USB znika kursor (można włączyć komendą „‘b”), a znaki naciskane na klawiaturze są transmitowane tylko po USB.		
Jeśli w odpowiedzi na komendę „‘D”, „‘o” otrzymamy „‘DE” lub „‘oE”, oznacza to złą wartość parametru/parametrów (pierwsza cyfra większa niż 4 i/lub druga większa niż 1).		



Rysunek 8. Wirtualny port COM po zainstalowaniu

Nastąpi proces instalacji sterownika wirtualnego portu COM. Po zakończeniu instalacji naciskamy „Zakończ”. W „Menedżerze urządzeń” należy sprawdzić numer portu COM pod którym została zainstalowana płytka. Jest to nowy port COM o nazwie „Communications port”, w przypadku komputera autora był to COM9.

Gdy znamy już numer portu COM pod którym zainstalował się interfejs, wystarczy uruchomić program terminalu (Hyper Terminal, Xterm, EZterm, Bray Terminal+). Dalsze przykłady będą opisane na przykładzie „Hyper terminala”. Po uruchomieniu programu wskazujemy menu „Plik/Właściwości” i wybieramy odnaleziony wcześniej COM (**rysunek 9**). Nie ma potrzeby konfigurowania parametrów transmisji, ponieważ nie są one uwzględniane przez sterownik (pracuje on z największą prędkością ograniczoną możliwościami procesora AVR). Jeśli w oknie konfiguracji nie można wybrać portu COM, to należy przerwać połączenie.

Po skonfigurowaniu terminalu należy w jego oknie napisać jakikolwiek tekst, który natychmiast powinien zostać odebrany przez płytkę testową i pojawić się na zamontowanym na niej wyświetlaczu. to powinien pojawić się na wyświetlaczu LCD. Płytka testowa odsyła znaki (funkcja echa), więc wpisany tekst zobaczymy również na ekranie PC. Jeśli terminal nie reaguje na wpisywany tekst, należy nacisnąć przycisk „Połącz”.

Listing 1. Wysłanie znaków do komputera PC

```
                                        // Wypełniamy tablicę danymi (maksymalnie 8 liczb)
tbuf[0] = SRAMKA;                       // Pierwszy znak do wysłania
tbuf[1] = CMD_SETCURS;                  // Drugi znak do wysłania
tbuf[2] = ,\r';                         // Trzeci znak do wysłania
tbuf[3] = ,\n';                         // Kolejny znak do wysłania
                                        // Określamy liczbę danych do wysłania
tcnt    = 4;                            // Liczba znaków do wysłania (1..8)
                                        //Wysyłkę realizuje poniższy fragment
                                        //umieszczony pod koniec funkcji main()
/*  device -> host    */
    if( usbInterruptIsReady() )
    {
      if( tcnt || sendEmptyFrame )
      {
        usbSetInterrupt((uchar *)tbuf, tcnt);
        /* wysłanie bloku pustego do sygnalizacji końca wysyłania danych */
        sendEmptyFrame = tcnt==8? 1:0;
        tcnt    = 0;
      }
    }
```

Listing 2. Znaczenie dyrektyw #define w konfigurowaniu sposobu funkcjonowania USB

```
/* ------------- KONFIGURACJA PODSTAWOWA ------------- */
/* Nazwa portu, do którego dołączono linie USB. Obie linie danych USB (D+, D-) muszą być podłączone do tego samego
portu (najczęściej będzie to port B lub D zależnie od typu mikrokontrolera) */
#define USB_CFG_IOPORTNAME      D
/* Nr portu, do którego jest przyłączona linia D- */
#define USB_CFG_DMINUS_BIT      3
/* Nr portu do którego podłaczona jest linia D+. Musi to być linia przerwania INT0. */
#define USB_CFG_DPLUS_BIT       2

/* ------------- PARAMETRY OPCJONALNE ------------ */
/* Port, do którego jest przyłączony rezystor zasilający (pull up) o rezystancji 1,5 k.  Usunięcie tej linii wyłączy
zasilanie rezystora i trzeba go będzie dołączyć na stałe do napięcia zasilania. W ten sposób wyłączymy też możliwość
symulowania odłączenia urządzenia z portu USB (wyjęcia i włożenia wtyczki). //
#define USB_CFG_PULLUP_IOPORTNAME   D
/* Nazwa portu do którego przyłączono rezystor pull up */
#define  USB_CFG_PULLUP_IOPORT   PORTB
/* Numer linii, do której jest przyłączony rezystor pull up o rezystancji 1,5 k// */
#define USB_CFG_PULLUP_BIT          4
/* Definiowanie VENDOR ID */
#define  USB_CFG_VENDOR_ID       0xc0, 0x16
/* Definiowanie DEVICE ID */
#define  USB_CFG_DEVICE_ID       0xe1, 0x05
/* Definiowanie numeru wersji */
#define USB_CFG_DEVICE_VERSION   0x00, 0x01
/* Nazwa urządzenia USB */
#define USB_CFG_DEVICE_NAME     'U', 'S', 'B', '-', 'P', 'I', 'O'
/* Długość nazwy */
#define USB_CFG_DEVICE_NAME_LEN 7
/* Definiowanie VENDOR NAME */
#define USB_CFG_VENDOR_NAME     ,w', ,w', ,w', ,.', ,r', ,e', ,c', ,u', ,r', ,s', ,i', ,o', ,n', ,.', ,j', ,p'
/* Długość nazwy */
#define USB_CFG_VENDOR_NAME_LEN 16
```

Poza wyświetlaniem tekstu można wydawać komendy dla wyświetlacza. Komendy wydajemy po znaku „ ' ” (apostrof). Dostępne komendy wymieniono w tabeli 2.

Klasa CDC pracuje także z Linuksem oraz MacOS.

Program

Aby wykorzystać USB w swoim projekcie, trzeba wiedzieć, które procedury służą do wysyłania i odbierania danych. Z tego punktu widzenia istotne są następujące procedury:

- ***InitUsb()*** – inicjalizacja interfejsu.
- ***Sei()*** – włączenie przerwań.
- ***usbPoll()*** – procedura wywoływana w pętli głównej, nie rzadziej niż co 50 ms.
- ***usbFunctionWriteOut(uchar *data, uchar len)*** – wywoływana po odebraniu danych z USB. Liczba zapisanych znaków zawiera się w przedziale 1...8. Dane znajdują się w tablicy *rbuf[]*. Sposób analizy danych można obejrzeć w programach źródłowych.

Aby wysłać dane do komputera PC, należy wypełnić tablicę *tbuf[]* kodami znaków do wysłania. Ilustruje to przykład zamieszczony na **listingu 1**. **Jeśli podczas pracy komunikacja przez USB będzie się zawieszać (brak odbioru danych z komputera), oznacza to, że w programie są zawieszane przerwania lub użyto przerwania „signal” („isr”) zamiast interrupt”.**

Aby interfejs USB pracował poprawnie, należy go skonfigurować. Wykonuje się to, zmieniając postać dyrektyw #define w pliku usbconfig.h (**listing 2**).

Budując urządzenie docelowe i wykorzystując biblioteki dołączone do artykułu, należy pamiętać, że zmiany identyfikatora i nazwy dostawcy urządzenia (*VENDOR ID*, *VENDOR NAME*) należy wprowadzić także w pliku *avrcdc.inf*, aby dało się zainstalować urządzenie.

Sławomir Skrzyński, EP
slawomir.skrzynski@ep.com.pl