BlueIR – wzmacniacz multimedialny

kit 2901 AVT

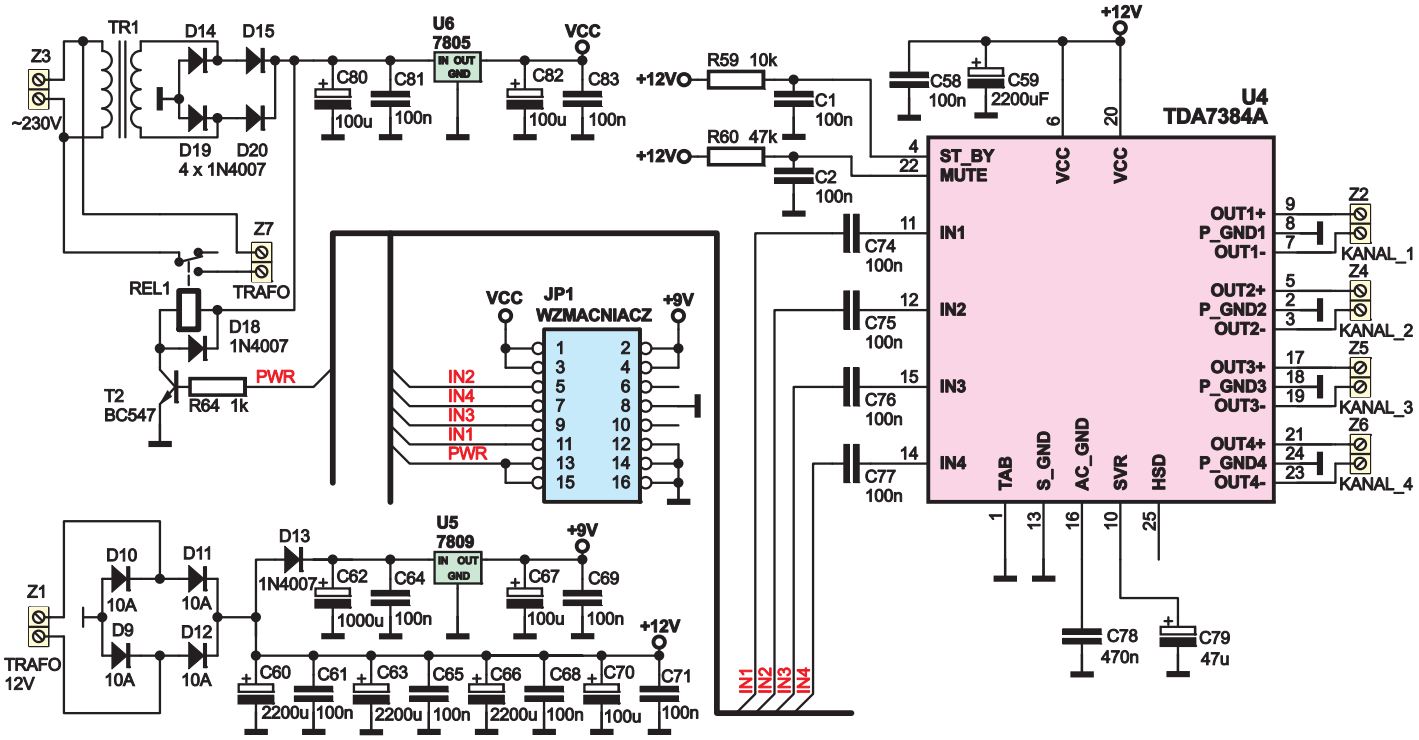
część 1

W ciągu kilkunastu lat istnienia EdW opisane zostały różnorodne projekty audio – od najprostszych po bardzo złożone. Czy projektowanie kolejnego takiego urządzenia ma sens? Zależy czego się od niego oczekuje. Gdybyś miał, Czytelniku, zażyczyć sobie wzmacniacz mocy, to czego byś zażądał?

Ja proponuję budowę wersji 4-kanałowej. Czy lubisz mieć kontrolę nad barwą dźwięku? BlueIR ma dwa korektory graficzne z pięcioma pasmami, sterowane niezależnie – każdy z nich przypisany jest do dwóch kanałów. Do tego dodana została wspólna regulacja głośności. Każdy z kanałów zapewnia moc na poziomie 20W, co wbrew pozorom jest aż nadto wystarczające do nagłośnienia pokoju. Podejrzewam, że lubisz spędzać czas przy komputerze i nie zawsze będziesz miał ochotę się od niego oderwać, żeby wyregulować poziom czy barwę dźwięku. W Elportalu znajdziesz prostą w obsłudze aplikację, która pozwala na sterowanie wszystkimi parametrami wzmacniacza bez odchodzenia od komputera – wystarczy myszka. Co zrobić, gdy jednak wygodnie leżysz w łóżku i nie masz ochoty z niego wychodzić? Wystarczy użyć pilota RC5 i możesz załadować ostatnią playlistę WinAMP-a, rozpocząć odtwarzanie, wstrzymać je, przeskakiwać po utworach... Jedynym wymaganiem jest podłączenie BlueIR-a do włączonego komputera. Może uważasz, że podczerwień jest już niemodna – trzeba celować pilotem, szukać go w pokoju, itd. Za to z komórką się nie rozstajesz, prawda? W Elportalu znajdziesz oprogramowanie napisane w Javie – jeżeli Twój aparat ma Bluetooth, możesz za jego pomocą sterować wzmacniaczem. Po uruchomieniu midletu będziesz miał możliwość wybrania urządzenia, z którym chcesz się połączyć i po nawiązaniu połączenia komórka umożliwi dostęp do wszystkich funkcji: korektorów graficznych, regulacji głośności i funkcji oferowanych przez WinAMP-a (otwarcie ostatniej playlisty, rozpoczęcie odtwarzania, wstrzymanie, etc.). Zgodnie z wymogami współczesnej mody, wzmacniacz został wyposażony w funkcję stand-by umożliwiającą wyłączenie głównego transformatora, gdy nie jest potrzebny (również za pomocą komórki, pilota lub komputera).

Prawdopodobnie nie wspomniałem o najważniejszym – o jakości dźwięku. Nie

Rys. 1 Schemat ideowy wzmacniacza



BlueIR – wzmacniacz multimedialny

kit 2901 AVT

część 2

Oprogramowanie

MIKROKONTROLER. Oprogramowanie dla mikrokontrolera zostało napisane w darmowym środowisku WinAVR, a jego pełny kod źródłowy dostępny jest w Elportalu. Wykorzystuje ono dobrodziejstwa języka C++ i zawiera kilka klas.

Pierwszą z nich, prawdopodobnie najciekawszą, jest klasa *btm222*, która odpowiada za obsługę modułu Bluetooth. Do zadań konstruktora należy przygotowanie modułu do pracy. W pierwszej kolejności resetowany jest układ BTM-222 za pomocą sygnału *RESET* podłączonego do mikrokontrolera, konfigurowane są porty I/O mikrokontrolera oraz układ USART wykorzystywany do komunikacji z modułem. Użyte tu zostały ustawienia domyślne, czyli prędkość 19 200, jeden bit stopu i brak kontroli parzystości. Po uruchomieniu wszystkich pozostałych komponentów wywoływana jest jeszcze funkcja *setupBTM* dokonująca końcowych ustawień. Teoretycznie mogła ona wejść w skład konstruktora, jednakże program nie chciał wtedy poprawnie pracować. Wewnątrz tej funkcji wywoływane są trzy komendy AT konfigurujące sposób zachowania się modułu. Pierwszym krokiem jest wyłączenie echa komendą „ate0". Jest to bardzo ważne, gdyż w innym wypadku, przed ustanowieniem połączenia, moduł odsyła każdy znak, jaki został wysłany. Było to powodem bardzo kłopotliwego błędu, którego wykrycie zabrało mi trochę czasu. Objawiał się on w sytuacji, gdy nie był uruchomiony midlet na komórce – wtedy dane przesyłane do komórki były odsyłane (bo tak działa echo) i podlegały normalnej „obróbce". Część pakietów okazywała się poprawna i w efekcie w urządzeniu samoistnie przestawiał się korektor graficzny oraz poziom głośności. Było to bardzo uciążliwe.

Po wyłączeniu echa wydawana jest komenda *atn=BlueIR*, która ustawia nazwę modułu. Dzięki temu, w czasie wykrywania go przez komórkę, ta właśnie nazwa zostanie wyświetlona. Pomijając walory estetyczne ułatwia to znacznie określenie, z jakim modułem się połączyć. Szczerze powiedziawszy, nawet nie pamiętam, jaka jest domyślna nazwa modułu BTM-222 :).

Ostatnia komenda AT (*atp=1234*) ustawia PIN umożliwiający autoryzację modułu Bluetooth. Po wykonaniu każdej z tych komend BTM-222 odsyła potwierdzenia: OK, jeżeli polecenie zostało przyjęte, albo ERROR jeżeli nie zostało ono przyjęte. W czasie testowania okazało się, że kolejne znaki tych komend muszą być wprowadzane z dużymi odstępami czasu, co objawia się obecnością relatywnie dużych opóźnień w tej części programu. W programie zaszyty został specjalny moduł testowy sprawdzający, czy polecenia zostały zaakceptowane – szczegóły za chwilę.

Po nawiązaniu połączenia między BTM-222 i komórką tor radiowy staje się „przezroczysty" i otrzymujemy bezprzewodowe łącze między telefonem i mikrokontrolerem.

Wartą uwagi funkcją jest *interrupt()*, która jest wywoływana w przerwaniu układu USART (po nadejściu nowego znaku). Jej rola sprowadza się do buforowania transmisji. Po pierwsze, sprawdzane jest, czy dostępne jest jeszcze miejsce w buforze, a jeżeli nie, to odebrany znak nie jest zapisywany. Jest to konieczne, gdyż w innym przypadku mogłoby dojść do nadpisania danych innych klas i spowodować niepoprawne działanie całego oprogramowania. Błędy tego typu są bardzo trudne do wykrycia, więc należy zawsze przeciwdziałać ich powstawaniu.

Warto zadać sobie pytanie, jak pracuje taki bufor. Jest to typowa kolejka FIFO – pierwszy zapisany do niej znak zostanie również odczytany jako pierwszy. Jest z nią związany wskaźnik **wsk*, który wskazuje początek tablicy stanowiącej kolejkę. Przychodzący bajt jest zapisywany do wskaźnika, czyli do pierwszej wolnej komórki tablicy i wskaźnik jest przesuwany o jedną pozycję – wskazuje na kolejny element tablicy. Następny bajt również zapisywany jest do wskaźnika, ale już w innym miejscu tablicy. Taka organizacja danych pozwala od razu określić, gdzie znajduje się pierwszy (najwcześniej odebrany) bajt – jest on na pierwszej pozycji tablicy. Oprócz zapisu odebranego bajtu wykonywana jest jeszcze jedna operacja – zwiększenie o jeden wartości zmiennej *odebrano*. Dzięki temu w każdej chwili można określić, ile nieprzeczytanych znaków znajduje się w naszym buforze. Ze zmienną tą związana jest funkcja *bytesToRead()*, która nie robi nic innego jak tylko zwraca wartość zmiennej *odebrano*.

Wiedząc, jak zapisywać znaki do bufora, należy zastanowić się, w jaki sposób je stamtąd pobrać. Służy do tego funkcja *readBuffer()* (**listing 1**), która przyjmuje dwa argumenty. Pierwszy z nich określa miejsce w pamięci, do którego należy pobrać znaki z bufora (np. jakaś tablica), a drugi argument informuje, ile znaków pobrać. Okazuje się, że proces pobierania znaków z bufora jest bardziej złożony niż zapisywanie do niego. Pierwszym krokiem jest utworzenie jeszcze jednego, pomocniczego wskaźnika **cpy*, wskazującego obszar pamięci, z którego ma się odbywać kopiowanie. Następnie utworzono pętlę *for* wykonującą się

Listing 1

```
//odczytaj dane z bufora
void btm222::readBuffer(char *dest, unsigned char ile){
  char *cpy = bufor ;
    //wskaznik pomocniczy
  for(unsigned char t=0 ; t<ile ; t++){
    *(dest+t) = *(cpy+t) ;
        //operacje na wskaznikach
    *wsk-- ; //przesun wskazniki, tak aby pokazywaly na wlasciwe miejsce
    odczytano-- ; //zmniejsz liczbe odczytanych znakow
  }
    //przesun dane w buforze
  for(unsigned char t=0 ; t<(BTM222_BUF_SIZE-ile) ; t++){
    bufor[t] = bufor[t+ile] ;
  }
}
```

tyle razy, ile znaków ma być odczytanych z bufora. Wewnątrz tej pętli następuje kopiowanie znaków z miejsca wskazywanego przez wskaźnik **cpy* do miejsca docelowego wskazywanego przez wskaźnik **dest (*argument funkcji). Adresy tych wskaźników są sukcesywnie zwiększane. Po każdym skopiowanym bajcie zmniejszana jest zmienna *odebrano*, co odzwierciedla zmniejszenie liczby bajtów, jakie jeszcze są w buforze. Oprócz tego przesuwany jest także wskaźnik **wsk* „w dół" tak, aby wskazywał na zwolnione miejsce w buforze odbiorczym. Pozostaje jednak problem „dziury", jaka tworzy się po pobraniu danych z początku tablicy. Jest on rozwiązywany za pomocą drugiej pętli *for* operującej na buforze, która wykonuje wewnętrzne kopiowanie. Działanie to sprowadza się do umieszczenia na pierwszej pozycji tablicy bufora pierwszego, nieodczytanego bajtu, na drugiej pozycji umieszczany jest drugi, nieczytany bajt, itd. Uzyskujemy w ten sposób proste przesunięcie zawartości bufora tak, aby trzymać się zasady, że w pierwszej komórce jest najwcześniej odebrany, nieczytany znak. Rozwiązanie to nie jest może optymalne, ale w przypadku tak wolnej transmisji, z jaką mamy do czynienia w urządzeniu, nie stanowi to problemu. Znacznie bardziej optymalne jest tworzenie buforów pierścieniowych, ale w tym wypadku nie było to konieczne.

Do wysyłania danych wykorzystywana jest funkcja *write*, która również przyjmuje dwa argumenty – pierwszy określa obszar pamięci, z którego należy pobrać informacje do wysłania, a drugi z parametrów definiuje, ile znaków przesłać. Dane te są przesyłane do komórki i tam zostają poddane obróbce. Wysyłanie odbywa się w pętli przy użyciu sprzętowego układu USART.

Podobnie działa klasa *PCcomm*, która również buforuje transmisję danych, ale nie do komórki, tylko do komputera PC. Funkcje *write*, *bytesToRead, readBuffer* oraz *discarBuffer* mają działanie identyczne jak te z klasy *btm222*. Różnicę stanowi sposób transmisji, gdyż mikrokontroler ma, niestety, tylko jeden port szeregowy. Konieczne zatem było opracowanie transmisji programowej. Zdecydowałem, że optymalną prędkością będzie 4800 b/s, gdyż zbyt duża szybkość transmisji mogłaby zbyt mocno obciążyć procesor. Wysyłanie i odbieranie danych do komputera jest buforowane w obie strony i oparte na przerwaniach, które pochodzą od liczników T0 oraz T2. Dane nie są wysyłane bezpośrednio, ale najpierw następuje ich umieszczenie w buforze nadawczym, skąd są pobierane i sukcesywnie przesyłane do komputera. W przerwaniu sprawdzane jest, czy bufor jest pusty i gdy tak nie jest, rozpoczyna się procedura transmisji znaku. W czasie trwania przerwania wysyłany jest tylko jeden bit i następuje powrót do programu głównego. W czasie następnego przerwania wysyłany jest kolejny bit, itd. do czasu całkowitego zwolnienia bufora.

Bardzo ważną klasą jest klasa *constr*, która stanowi pojemnik do przechowywania ustawień wzmacniacza – poziomów korektorów graficznych, poziomów głośności oraz informacji o zasilaniu (wł/wył). Kopia tej klasy znajduje się zarówno w aplikacji komputerowej, jak i midlecie dla komórki. Dzięki temu łatwiej jest zarządzać wymianą danych pomiędzy wszystkimi tymi urządzeniami. Sprowadza się to do przesłania bieżącej konfiguracji do komputera i do komórki. Przykładowo, zmieniając ustawienie za pomocą telefonu, zawartość klasy *constr* zmieni się i zostanie ona najpierw wysłana do mikrokontrolera, tam opracowana, następnie przesłana do komputera i aplikacja BlueIR automatycznie skoryguje swoje wskazania.

Chcąc sobie ułatwić taką wymianę danych, przygotowałem dwie funkcje: *send_to_pc()* oraz *send_to_mobile()*, które w działaniu są bardzo zbliżone. Ich rola sprowadza się do przesłania bieżącej konfiguracji do, odpowiednio, komputera lub komórki. Obie funkcje „pakują" zmienne bitowe, takie jak informacja o rozpoczęciu odtwarzania, zmiana utworu, zatrzymanie czy stan (wł/wył) głównego transformatora. W czasie wysyłania pojemnika do komputera dodatkowo wyznaczana jest suma kontrolna CRC w oparciu o funkcję biblioteczną *_crc_ibutton_update()* i dołączana do przesyłanych danych. Przed wysłaniem dane są umieszczane w podręcznym buforze klasy *constr* i dopiero po skompletowaniu całego pakietu wysyłane za pomocą funkcji *write* omówionej powyżej. Działanie polecenia *opracuj()* jest odwrotne – na podstawie danych zapisanych w podręcznym buforze *buf* odtwarzane są poszczególne wartości i zapisywane w obiekcie *pojemnik*.

Klasa *tda7317* odpowiada za wysterowanie układów TDA stanowiących korektor graficzny oraz regulator głośności. Układy te mają wejście adresowe, co pozwala na podłączenie dwóch takich elementów do jednej magistrali I^2C. W czasie przesyłania danych, w ramce znajduje się stosowny bit określający, który z układów ma odebrać przesyłane dane. Konstruktor tej klasy konfiguruje jedynie sprzętowy blok odpowiedzialny za transmisję. Klasa zawiera dwie funkcje, pierwsza z nich: *put_byte* zgodnie z nazwą pozwala wysłać jeden bajt do układu TDA. Druga klasa, *update*, jest bardziej złożona. Odpowiada ona za pobieranie bieżącej konfiguracji z klasy *constr* i przesyłanie jej do układów TDA. Zapis obu tych układów odbywa się sekwencyjnie, tzn. najpierw jeden, potem drugi. Scenariusz transmisji nie odbiega zbytnio od utartych schematów – najpierw wysyłany jest bit startu, następnie adres docelowego układu, parametry korektora i poziom głośności, a na końcu bit stopu. Można tu jednak spotkać dwie nietypowe rzeczy. Pierwszą z nich jest zapobieganie zawieszeniu transmisji,

```
while(port.bytesToRead()>=11){ //czy odebrano wystarczajaca liczbe
bajtow ?
  //pobierz pierwszy znak
  port.readBuffer(pojemnik.recBuf, 1) ;
  //sprawdz, czy to znak specjalny
  if(pojemnik.recBuf[0] == CONSTR_SEND_FRAME){
    //pobierz pozostale dane
    port.readBuffer(&pojemnik.recBuf[1], 10) ;
    //wyznacz sume kontrolna
    unsigned char suma_crc = 0 ;
    for(unsigned char w=0 ; w<10 ; w++){
      suma_crc = _crc_ibutton_update(suma_crc, pojemnik.recBuf[9-w]) ;
    }
    //sprawdz, czy jest zgodna
    if(suma_crc == pojemnik.recBuf[10]){
      pojemnik.send_to_pc() ; //odeslij ramke do komputera
    }
  }
  //sprawdz, czy to znak specjalny
  if(pojemnik.recBuf[0] == CONSTR_UPDATE){
    //pobierz pozostale dane
    port.readBuffer(&pojemnik.recBuf[1], 10) ;
    //wyznacz sume kontrolna
    unsigned char suma_crc = 0 ;
    for(unsigned char w=0 ; w<10 ; w++){
      suma_crc = _crc_ibutton_update(suma_crc, pojemnik.recBuf[9-w]) ;
    }
    //sprawdz, czy jest zgodna
    if(suma_crc == pojemnik.recBuf[10]){
      pojemnik.opracuj() ;
      if(pojemnik.pwr){sbi(PORTC, 3) ;} //wlaczono zasilanie ?
      _delay_ms(50) ;
      pojemnik.send_to_mobile() ; //wyslij pojemnik do telefonu
      korektor.update() ; //ustaw uklady TDA7317
    }
  }
  //wyszukiwanie ciagu testowego
  if(pojemnik.recBuf[0] == '*'){
    //odczytaj reszte znakow
    port.readBuffer(&pojemnik.recBuf[1], 10) ;
    //sprawdz, czy sa 4 gwiazdki pod rzad
      if( (pojemnik.recBuf[1] == '*') && (pojemnik.recBuf[2] == '*') ¬
                                      && (pojemnik.recBuf[3] == '*') ){
      //wyslij komunikat o rozpoczeciu testu
      port.write_string("TEST MODE\r\n") ;
      port.discardBuffer() ;
      //wyonaj polecenia AT
      _delay_ms(100)  ;
      _delay_ms(100)  ;
      btm.setupBTM() ;
      //przesylaj dane z komorki
      while(1){
        //odsylanie danych do komorki
        if(btm.bytesToRead()>0){
          char t[10] = {0};
          btm.readBuffer(t, 1) ;
          port.write(t, 1) ;
        }
        //mmozliwosc opuszczenia trybu testowego
        if(port.last_char() == '#'){
          port.discardBuffer() ;
          break ;
        }
      }
      port.write_string("\r\nRUN MODE\r\n") ;
    }
  }
}
```

Listing 2

mianowicie przed wysłaniem kolejnej informacji konieczne jest odczekanie, aż moduł TWI zakończy pracę. Można to zrealizować w prosty sposób, tzn. sprawdzając stan flagi w rejestrze instrukcją *while*. Problem pojawił się w momencie testowania samej płytki sterownika, która w ogóle nie reagowała na polecenia wysyłane z komputera, pilota czy modułu Bluetooth. Winę za ten stan rzeczy ponosił właśnie moduł TWI, który czekał na zakończenie zadania, które nie mogło się zakończyć, gdyż układy TDA nie były dołączone i nie odpowiadały. W związku z tym pętla *while* została wzbogacona o instrukcję warunkową *if*, która po upływie założonego czasu przerwie dalsze oczekiwanie na odpowiedź. Drugim nietypowym rozwiązaniem jest obecność tablicy *rzutowanie*. Jest ona wynikiem kolejnego uproszczenia, mianowicie sposób reprezentacji poziomów korektora w aplikacji oraz w układach TDA jest inny, więc niezbędne było dokonanie stosowanego przekształcenia. Zrealizowano to w najprostszy możliwy sposób – poprzez odniesienie się do komórki tej tablicy za pomocą wartości otrzymanej z aplikacji odczytywana jest wartość adekwatna dla korektora graficznego.

Na koniec warto przyjrzeć się zawartości pliku *main.cpp*, w którym znajduje się główna pętla programu. Przed wejściem do niej konfigurowane są porty mikrokontrolera oraz wysyłana jest konfiguracja do komputera i komórki. Jest to niezbędne, aby po włączeniu zasilania wszystkie urządzenia posiadały identyczne informacje o konfiguracji. Z tego samego powodu wywoływana jest procedura *update*, która zapisuje zawartość klasy *constr* do układów TDA.

Pierwszym elementem pętli głównej jest fragment kodu odpowiedzialny za wymianę danych z komputerem – **listing 2**. Znajduje się tam pętla *while*, która wykonywana jest dopóty, dopóki w buforze odbiorczym znajduje się przynajmniej 11 bajtów, czyli potencjalnie cała ramka zawierająca konfigurację przesłaną z komputera. Pierwszym krokiem jest odczyt jednego bajtu z bufora i sprawdzenie, czy odpowiada jednej z wartości stałych: *CONSTR_SEND_FRAME* lub *CONSTR_UPDATE*. Stanowią one rodzaj preambuły i jeden z mechanizmów sprawdzania poprawności transmisji. Jest to swoistego rodzaju nagłówek informujący, że pobrany bajt jest początkiem nowej ramki danych. Pierwsza z tych stałych stanowi prośbę o wysłanie bieżącej konfiguracji, a sytuacja taka ma miejsce po uruchomieniu aplikacji na komputerze. Zapobiega to ustawieniu wartości domyślnych, które mogą być różne od ustawień przechowywanych przez mikrokontroler. Druga wartość stała informuje o tym, że w buforze znajduje się ramka zawierająca nową konfigurację. Jest ona wysyłana za każdym razem, gdy użytkownik zmieni w aplikacji jakiś parametr, np. poziom głośności

Gdyby się okazało, że pobrany znak nie reprezentuje żadnego z tych dwóch nagłówków, to nic się nie dzieje i ponownie sprawdzana jest pętla *while*. Gdy w buforze jest minimum 11 znaków, proces pobierania pierwszego bajtu powtarza się. W sytuacji, w której odebrano wadliwy pakiet, zostanie on sukcesywnie usunięty z bufora po odebraniu właściwego pakietu dzięki kolejnym pobraniem bajtów z bufora do czasu natrafienia na właściwy bajt nagłówka.

Kiedy odczytany zostanie właściwy nagłówek, wykonywana jest jedna z dwóch instrukcji warunkowych – zależnie od tego, który z nagłówków odebrano. Następuje wtedy pobranie pozostałej części pakietu z bufora (czyli 10 bajtów) i wyznaczenie sumy kontrolnej CRC. Po stwierdzeniu, że jest ona zgodna z tą, która została wysłana, pakiet zostaje uznany za poprawny i podlega opracowaniu. Wykonywana jest wtedy jedna z dwóch akcji: wysłanie konfiguracji do komputera lub zapisanie konfiguracji odebranej z komputera. W tym drugim przypadku należy jeszcze sprawdzić, czy nie włączono zasilania głównego transformatora – gdyby tak się stało, należy natychmiast włączyć przekaźnik, aby zasilić układy TDA. W innym wypadku nie da się do nich zapisać ustawień korektorów i poziomu głośności. Wysyłana jest tu jeszcze konfiguracja do komórki, żeby ona również miała aktualną konfigurację.

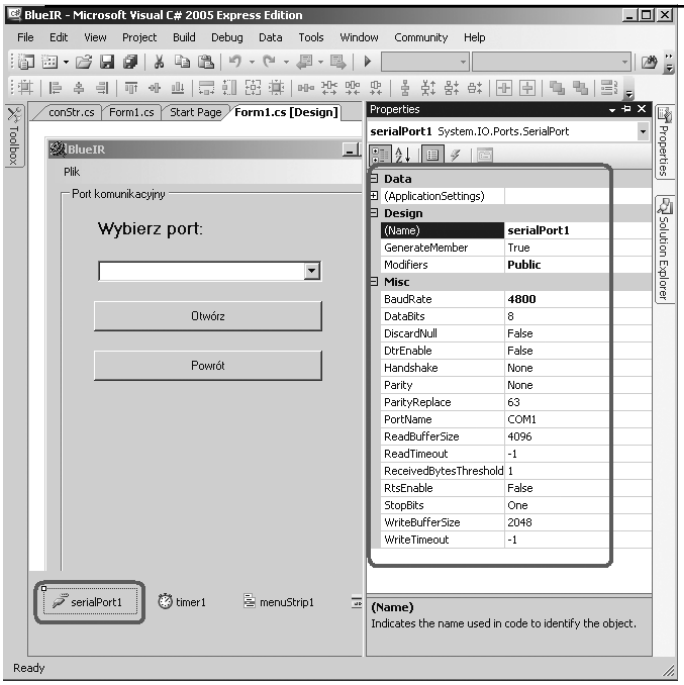
Jest jeszcze trzecia instrukcja *if*, która sprawdza, czy w buforze nie pojawił się znak * (gwiazdka), co jest informacją o wejściu do trybu testowego. Polega on na zapętleniu bufora odbiorczego BTM-222 oraz bufora nadawczego portu szeregowego, w wyniku czego wszystkie znaki pochodzące od modułu Bluetooth są od razu przesyłane do komputera. Dzięki temu można sprawdzić, czy akceptowane są komendy AT. Po pojawieniu się znaku # pętla *while* jest przerywana i urządzenie powraca do normalnego trybu pracy.

Obsługa wymiany danych z komórką przebiega prawie identycznie, tzn. pobieranie pierwszego bajtu, sprawdzanie nagłówka, etc. jest analogiczne. Różnica polega na pominięciu sumy kontrolnej CRC i trybu testowego. Dodatkowo pojawia się jeszcze jedną instrukcję *if*. Jej rola sprowadza się do poszukiwania ciągu znaków „*CONNECT@*”, który pojawia się w buforze w momencie połączenia modułu BTM-222 z komórką. Dzięki temu można łatwo stwierdzić, kiedy taka sytuacja ma miejsce i przesłać do telefonu bieżącą konfigurację. Sam proces wyszukiwania nie ingeruje w bufor odbiorczy, tylko wykorzystuje fakt, iż wcześniejsza procedura wyszukiwania nagłówka pobiera z bufora kolejne znaki. Dzięki temu można analizować, czy kolejno pobierane znaki tworzą poszukiwaną wartość, czy też nie.

Dalsza część programu koncentruje się na obsłudze pilota pracującego w standardzie RC5. Dane o odebranych znakach są odczytywane z klasy *pilot*. W pierwszej kolejności następuje sprawdzenie, czy nie naciśnięto przycisku odpowiedzialnego za rozpoczęcie odtwarzania utworu, otwarcie odtwarzacza, zatrzymanie, etc. Odbywa się to z wykorzystaniem prostych instrukcji warunkowych. W momencie, gdy odebrany kod zgadza się ze stosowaną wartością stałą, następuje ustawienie odpowiedniej flagi w klasie *constr*. Kolejna instrukcja *if* sprawdza, czy choć jedna z tych flag ma wartość różną od zera (false) i gdy tak się dzieje, to następuje wywołanie funkcji *send_to_pc*, która przesyła, oprócz konfiguracji, informację o takim zdarzeniu. Komputer na tej podstawie podejmuje stosowne działanie. Kolejny fragment programu odpowiada za obsługę poziomu głośności oraz sterowanie pracą transformatora. Wykorzystano ponownie instrukcje *if* działające na identycznej zasadzie, tzn. przez porównanie ostatnio odebranego kodu z pilota z wartością odpowiedniej stałej. Po stwierdzeniu równości zmieniane są poziomy głośności na obu kanałach i następuje uaktualnienie konfiguracji układów TDA i wysłanie tak zmienionej konfiguracji do komputera i komórki. Wspomniane stałe mają następujący zapis:
RC5_COMMAND_XXX
gdzie XXX jest nazwą polecenia, z jakim ta stała jest stowarzyszona. Są one utworzone w oparciu o instrukcje *#define* zawarte w pliku *ir.h*. Za nazwą stałej znajduje się kod przypisany do polecenia. Zmieniając te kody i następnie kompilując program, można dostosować działanie pilota do własnych oczekiwań.

KOMPUTER Zasadniczo opis oprogramowania komputerowego odbiega od głównego nurtu zainteresowań czasopisma, jakim jest EdW, jednakże warto zwrócić uwagę na jeden z aspektów. Mianowicie chciałbym pokazać, w jaki sposób zorganizować wysyłanie i odbieranie danych z portu szeregowego na komputerze PC. Rozważania te dotyczyć będą darmowego środowiska programistycznego Visual Studio 2005 Express Edition, w którym została napisana aplikacja BlueIR.

Osobom niezorientowanym w temacie przypominam, że układ FT232 jest „widziany” od strony komputera jako zwyczajny port COM, którego obsługa w Visual Studio jest



Rys. 11

obecnie bardzo prosta. Wszystko co potrzebne mieści się w kontrolce *SerialPort*, którą należy dodać do projektu. Po kliknięciu na niej, w oknie *Properties* (**rysunek 11**), zyskujemy dostęp do szeregu parametrów umożliwiających elastyczną konfigurację takiego portu. Warto tu wspomnieć o prędkości pracy (*BaudRate*), liczbie przesyłanych bitów (*DataBits*), kontroli parzystości (*Parity*), itp. W tym miejscu można również ustawić nazwę portu, który chcemy wykorzystać do komunikacji, ale lepszym rozwiązaniem jest stworzenie okienka dialogowego i pozostawienie wyboru użytkownikowi aplikacji. Dzięki temu program staje się bardziej uniwersalny, a po zadeklarowaniu portu, jaki użytkownik chce wykorzystać, możemy określić jego nazwę za pomocą prostej instrukcji:

serialPort1.PortName = „COMx”;

gdzie *serialPort1* to nazwa utworzonej kontrolki, natomiast *COMx* to nazwa używanego portu (np. COM4).

Po ustawieniu parametrów pracy oraz nazwy port należy otworzyć przed rozpoczęciem korzystania z niego. Służy do tego instrukcja:

serialPort1.Open();

Należy pamiętać, aby przed zamknięciem aplikacji zamknąć port, gdyż w innym wypadku nie będzie można go otworzyć ponownie do czasu zrestartowania komputera lub odłączenia i ponownego podłączenia urządzenia. Dokonuje się tego instrukcją:

serialPort1.Close();

Wiedząc, w jaki sposób skonfigurować port i go otworzyć, pora przyjrzeć się metodzie odczytu i zapisu do niego danych. Tutaj czeka nas miła niespodzianka, gdyż proces ten jest bardzo podobny do omówionego powyżej.

Odczytu dokonuje się poleceniem *serialPort1.Read(dest, offset, ile)*, gdzie:

– *dest* – oznacza miejsce pamięci, do którego dane mają być skopiowane, może to być np. tablica,
– *offset* – określa przesunięcie, tzn. od którego elementu tablicy dane mają być zapisywana,
– *ile* – liczba znaków, jakie chcemy wczytać do np. tablicy.

Co więcej, sposób obsługi portu jest bardzo podobny do tego, jaki zastosowano w pętli głównej mikrokontrolera – **listing 3**. Bardzo podobnie przebiega także określenie, ile nieczytanych bajtów znajduje się w buforze – sprawdzamy wartość atrybutu *BytesToRead*. Jak można wyczytać z przytoczonego listingu, dane są odczytywane w zdarzeniu, jakie zgłasza licznik (*timer1*). Funkcja *timer1_Tick* jest wywoływana co kilka milisekund, wtedy następuje sprawdzenie, czy w buforze znajduje

```
private void timer1_Tick(object sender, EventArgs e) {
  //odczyt danych z mikrokontrolera
  if (serialPort1.IsOpen) {
    try {
      //sprawdz rozmiar ramki
      while (serialPort1.BytesToRead>=11) {
        //pobierz dane
        byte[] bufor=new byte[20];
        //poszukiwanie znaku synchronizacji - informacji o
ramce
        serialPort1.Read(bufor, 0, 1);
        //czy to ramka type 'updateFrame' ?
        if (bufor[0]==conStr.updateFrame) {
          //pobierz pozostale znaki do bufora
          serialPort1.Read(bufor, 1, 10);
          //sprawdzenie sumy kontrolnej
          if (bufor[10]==CRC_module.crc.wyznacz(bufor, 10)) {
            //zapisz do zmiennych odebrane dane
            pojemnik.get(bufor);
            //odswiez kontrolki
            przeladowanie();
            generujEtykiety();
          }
        }
      } catch {
        ;
      }
    }
}
```

Listing 3

się już jakaś cała ramka i gdy tak jest, podejmowane są znane nam już kroki: sprawdzenie, czy mamy do czynienia z nagłówkiem, a jeżeli tak, to z jakim. Sprawdzana jest suma kontrolna po pobraniu całej ramki i następuje uaktualnienie konfiguracji oraz ustawień kontrolek (np. suwaków korektora graficznego). Pewną różnicą jest brak nagłówka z prośbą o odesłanie konfiguracji. Wynika to z faktu, że rolę „serwera” pełni mikrokontroler i tylko jemu przysługuje prawo określania, jakie dane są aktualne i jak mają być ustawione układy TDA.

Zapis do portu również nie jest trudny, przebiega praktycznie identycznie jak odczyt, ale korzysta z funkcji:

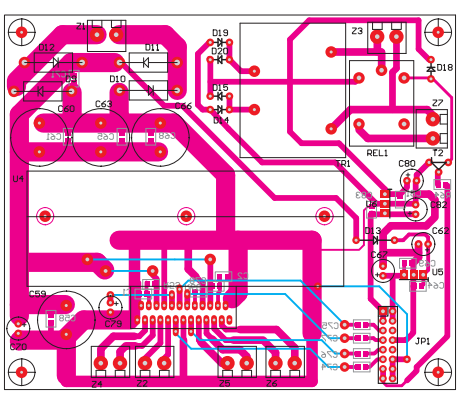
serialPort1.Write(dest, offset, ile), gdzie:

– *dest* – określa fragment pamięci, z którego chcemy pobrać dane do wysłania, może to być np. tablica,
– *offset* – określa, od którego miejsca (np. komórki tablicy) rozpocząć pobieranie danych do wysłania,
– *ile* – za pomocą tego parametru określamy ile bajtów ma zostać wysłanych.

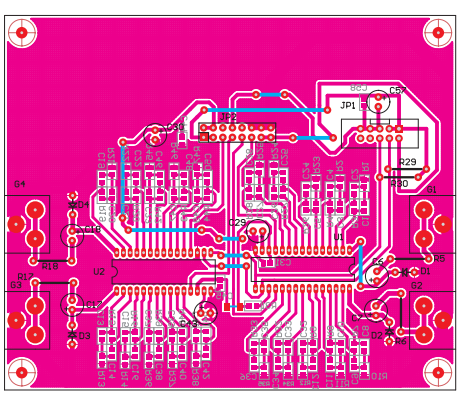
Montaż i uruchomienie

Układ można zmontować na płytkach drukowanych pokazanych na **rysunkach 12–14**. Montaż niczym nie odbiega od klasycznych reguł. Lutowanie warto rozpocząć od elementów SMD, a następnie małych, przewlekanych komponentów. Wszystkie trzy płytki mają identyczną wielkość, co pozwala na stworzenie „wafelka” za pomocą gwintowanego pręta o średnicy 3mm i tulejek dystansowych. Na samym dole warto przykręcić drewnianą podstawkę z transformatorem sieciowym. Zalecam zastosowanie transformatora toroidalnego (moc = 50W * liczba kanałów). Należy również zwrócić uwagę na

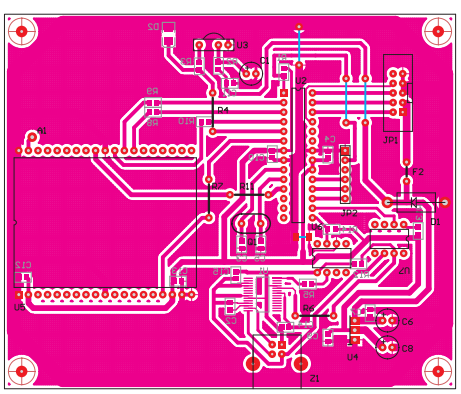
Rys. 12. Schemat montażowy wzmacniacza. Skala 50%



Rys. 13. Schemat montażowy korektora. Skala 50%



Rys. 14. Schemat montażowy sterownika. Skala 50%



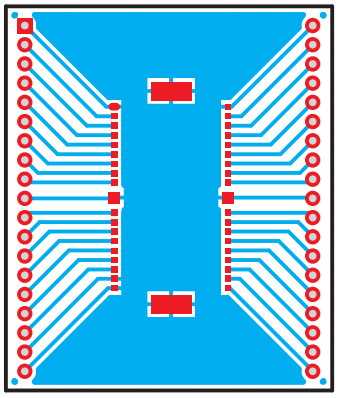
temperaturę wzmacniacza TDA7384A i zapewnić radiator adekwatny do rozpraszanej mocy. W przypadku pełnego wysterowania na trzech lub czterech kanałach należy liczyć się z koniecznością zapewnienia chłodzenia z użyciem wentylatora.

Przypominam o względach bezpieczeństwa – w urządzeniu występuje groźne, wręcz zabójcze, napięcie 230V. Z tego powodu pracujący wzmacniacz musi być osłonięty obudową (np. z pleksiglasu – należy pamiętać o otworach wentylacyjnych!!!) lub stać w miejscu, w którym nie może zostać przypadkowo dotknięty (np. wysoko na szafie).

Jak wspomniano już w artykule, istnieje możliwość sterowania pracą WinAMP-a z użyciem pilota RC5 lub komórki. Konieczne jest jednakże uprzednie włączenie obsługi klawiszy multimedialnych w tym odtwarzaczu. Można to zrobić poprzez kliknięcie prawym przyciskiem na okienku WinAMP-a i wybranie *Options->Preferences* i następnie zakładki *Global Hotkeys*. Odznaczamy w oknie obie pozycje i przy okazji usuwamy skrót ALT+L, aby można było z klawiatury wprowadzać normalnie literę „ł".

Warto jeszcze przeprowadzić test modułu Bluetooth przed podłączeniem komórki. Sprowadza się to do uruchomienia aplikacji BlueIR i wybrania *Plik->Panel testowy*. Następnie klikamy *Wykonaj test*. W okienku powinien pojawić się napis *TEST MODE* i po chwili trzy razy słowo *OK*. Będzie to oznaczało, że wszystko jest w porządku (znaczenie tych komunikatów opisane jest w części poświęconej oprogramowaniu). W przeciwnym wypadku należy kliknąć *Zakończ tryb testowy* i po paru sekundach ponownie *Wykonaj test*. Wszystkie te czynności można wykonać dopiero po ustawieniu port (*Plik->Ustawienia*).

Rys. 15 Schemat montażowy przejściówki do modułu Bluetooth



Do poprawnej pracy urządzenia konieczne jest także skonfigurowanie fuse bitów:
LFUSE = 0xEE
HFUSE = 0xD5
Zwracam jeszcze uwagę, że moduł Bluetooth jest dołączony do płytki sterownika za pomocą specjalnej przejściówki (**rysunek 15**). Jest ona wykonana w formie płytki PCB z dwoma rzędami złączy do goldpinów, a na płytce sterownika są właśnie dwa rzędy goldpinów. Wzór tej płytki można znaleźć w Elportalu razem z pozostałymi materiałami. Należy zwrócić uwagę, że „kropka" na obudowie (w formie wgniecenia na blaszce) wskazuje ostatnie, a nie pierwsze wyprowadzenie modułu BTM-222. Kropka ta powinna być zwrócona w stronę padu przeznaczonego do ewentualnego podłączenia anteny (na warstwie opisowej oznaczony jako A1). Wykonując przejściówkę, należy umieścić ścieżki na warstwie Top Layer i przylutować dwa rzędy gniazd na goldpiny tak, aby moduł Bluetooth znajdował się na górze.

W przypadku płytki wzmacniacza konieczne będzie przylutowanie kilku zworek, ale od strony druku.

Możliwości zmian

Udostępnienie pełnego oprogramowania w wersji źródłowej umożliwia wprowadzanie własnych modyfikacji i dodawanie nowych funkcji. Oryginalny projekt BlueIR, opisany w jednym z wydań Szkoły Konstruktorów, miał jeszcze analizator widma złożony z trzech diod LED. Niestety brak czasu i sensownej koncepcji na budowę takiego układu zaowocowały usunięciem go z projektu. Wykończenie urządzenia zabrało znacznie więcej czasu, niż mogłem przypuszczać i nie znalazłem czasu na walkę z analizatorem. Proste rozwiązanie oparte o filtr aktywny niestety nie zdało u mnie egzaminu, gdyż efekt końcowy zbyt mocno zależał od poziomu dźwięku na wejściu urządzenia. Zachęcam Czytelników do prowadzenia eksperymentów we własnym zakresie, np. z diodami RGB dużej mocy (tzw. Power LED).

Jest jeszcze jeden drobny mankament wzmacniacza, którego nie udało mi się wyeliminować, mianowicie słyszalny trzask przy zmianie poziomu głośności. Niestety nie miałem pomysłu na wyeliminowanie tej przypadłości, ale może Czytelnicy znajdą jakąś prostą receptę...

Jakub Borzdyński
jakub.borzdynski@elportal.pl

Wykaz elementów

Wzmacniacz

Rezystory

R59	10kΩ 2012[0805]
R60	47kΩ 2012[0805]
R64	1kΩ 2012[0805]

Kondensatory

C1,C2,C58,C61,C64,C65,C68,C69,C71,C74-C77,C81,C83	100nF 2012[0805]
C59,C60,C63,C66	2200µF
C62	1000µF
C67,C70,C80,C82	100µF
C78	470nF 2012[0805]
C79	47µF

Półprzewodniki

D9- D12	dioda 10A
D13,D14,D15,D18-D20	1N4007
T2	BC547
U4	TDA7384A
U5	LM7809
U6	LM7805

Pozostałe

TR1	transformator (szczegóły w artykule)
REL1	JQC-3FF (10A/277VAC)
Z1-Z7	ARK2

Korektor

Rezystory

R1,R10,R12,R13,R21,R23,R36,R45	6,8kΩ 2012[0805]
R2,R11,R14, R22,R24,R34,R37,R46	5,6kΩ 2012[0805]
R3,R7,R9,R15,R19,R26,R39,R42	39kΩ 2012[0805]
R4,R8,R16,R20,R27,R32,R40,R43	33kΩ 2012[0805]
R5,R6,R17,R18	100Ω
R25,R35,R38,R47	8,2kΩ 2012[0805]
R28,R33,R41,R44	47kΩ 2012[0805]
R29,R30	4,7kΩ
R31	10kΩ 2012[0805]

Kondensatory

C1,C2,C8, C9,C13,C14,C19,C20	1nF 2012[0805]
C3,C4,C10,C11,C15,C16,C21,C22	47nF2012[0805]
C5,C12,C23,C24,C32,C37,C38,C46	10nF 2012[0805]
C6,C7,C17,C18	47µF
C25,C26, C33,C34,C39,C40,C47,C48	3,3nF 2012[0805]
C27,C28,C31,C35,C36,C41,C42,C44,C45,C49,C50,C58	100nF 2012[0805]
C29,C30,C43,C57	10µF

Półprzewodniki

D1-D4	Zenera 2,4V
U1,U2	TDA7317

Pozostałe

G1-G4	gniazda cinch
JP1	goldpin 5x2
JP2	goldpin 8x2

Sterownik

Rezystory

R1,R3,R5,R8-R10,R12-R14	1kΩ 2012[0805]
R2	220Ω 2012[0805]
R4,R6,R7,R11	1kΩ

Kondensatory

C1	4,7µF
C2,C4,C7,C9-C15	100nF 2012[0805]
C3,C5	22pF 2012[0805]
C6,C8	10µF

Półprzewodniki

D1	Zenera 5,6V 5W
D2	Zenera 3,3V SMD
U1	FT232RL
U2	ATmega48
U3	TSOP1736
U4	SPX1117
U5	BTM-222 Bluetooth
U6,U7	4N25

Pozostałe

F2	bezpiecznik 0,5A
JP1	goldpin 5x2
JP2	goldpin 6x1
Q1	3,6864MHz
Z1	USB PORT

Płytki drukowane są dostępne w sieci handlowej AVT jako kit szkolny AVT-2901.

Uwaga! Podłączanie i odłączanie przewodów, a także inne czynności MUSZĄ ODBYWAĆ SIĘ PRZY ODŁĄCZONYM NAPIĘCIU!!! Na płytce wzmacniacza obecne jest pełne napięcie sieci i chwila uwagi może skończyć się tragicznie. NIE RYZYKUJ ŻYCIA, ŻEBY ZAOSZCZĘDZIĆ 5 MINUT!!!

Niepełnoletni i niedoświadczeni użytkownicy mogą budować, uruchamiać i wykorzystywać wzmacniacz WYŁĄCZNIE pod opieką dorosłych, wykwalifikowanych opiekunów.

Wzmacniacz bez obudowy musi stać w miejscu niedostępnym dla dzieci i poza zasięgiem ręki, aby uniknąć porażenia prądem elektrycznym.

prowadziłem pomiarów, jednakże testy odsłuchowe wypadły bardzo przyzwoicie.

Takiego wzmacniacza jeszcze w EdW nie było, więc mam nadzieje, że wzbudzi on zainteresowanie Czytelników i okaże się ciekawą alternatywą dla innych tego typu konstrukcji. Zapraszam do zapoznania się ze szczegółami.

Pragnę zaznaczyć, że całe niezbędne oprogramowanie (aplikacja komputerowa, midlet dla komórki oraz program dla mikrokontrolera) jest dostępne bezpłatnie w Elportalu, część również w wersji źródłowej. Wystarczy wgrać do procesora, zainstalować w komórce (za chwilę podam, jak to można zrobić) i uruchomić na komputerze. Nie są potrzebne żadne modyfikacje, instalowanie środowisk programistycznych ani nic w tym guście. Zainstaluj i używaj :).

Budowa części sprzętowej

Umieszczenie całego urządzenie na jednej płytce drukowanej uznałem za nieoptymalne. Zdecydowałem się na budowę modułową, która w moim odczuciu nadała też całości bardziej interesujący wygląd. Urządzenie składa się z trzech modułów, które łączy się standardowymi złączami zaciskanymi na taśmie – IDC-10 oraz IDC-16.

ZASILACZ. Schemat pokazany na **rysunku 1**, pokazuje dwa zasilacze oraz końcówkę mocy. Główny zasilacz, zbudowany w oparciu o transformator toroidalny (200W dla czterech kanałów lub 100W, gdy używane będą tylko dwa), służy do zasilania korektora oraz samej końcówki mocy. Zaciski pierwotne tego transformatora powinny być dołączone do złącza Z7. Są one przyłączane do sieci 230V przez przekaźnik sterowany z mikrokontrolera. Dioda D13 nie pozwala na rozładowanie kondensatora C62, aby zagwarantować rezerwę energii dla scalonego stabilizatora. Duże, szczytowe pobory prądu mogłyby spowodować wahania napięcia na wejściu U5 i prowadzić do nieprawidłowej pracy układów TDA7317.

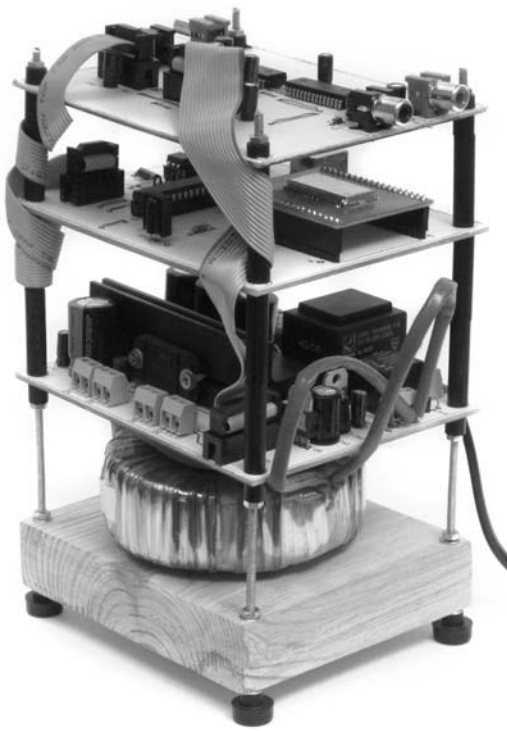
Końcówkę mocy stanowi scalona kostka TDA7384A, która zapewnia moc około 20W na każdym z czterech kanałów.

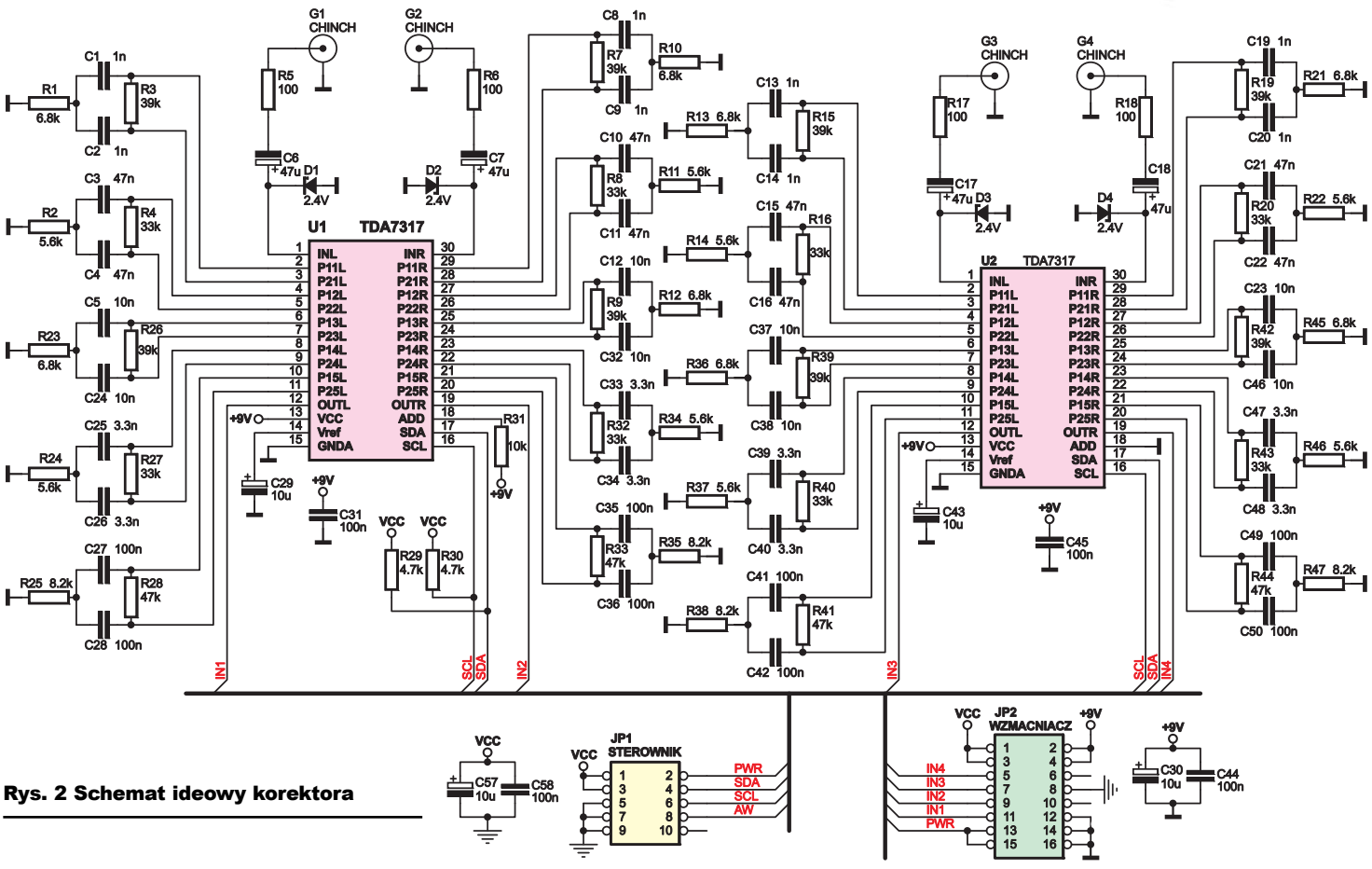
Dodatkowy, zawsze włączony, niewielki transformator TR1 służy do zasilania części cyfrowej – sterownika, by zapewnić nieprzerwaną pracę i tym samym umożliwić uruchomienie głównego transformatora w dowolnej chwili za pomocą pilota, komórki lub komputera.

Złącze JP1 służy do podłączenia płytki korektora. Z niego są również pobierane cztery sygnały akustyczne (IN1..IN4) poddane obróbce przez korektor graficzny i układ regulacji głośności. Znajdująca się tu linia PWR pochodząca od mikrokontrolera (umieszczonego na płytce sterownika) umożliwia sterowanie przekaźnika zasilającego główny transformator.

Chciałbym wspomnieć jeszcze o problemie z masą, który dość mocno dał się we znaki podczas konstrukcji wzmacniacza. Masa cyfrowa pochodząca ze stabilizatora U6 oraz masa analogowa końcówki mocy i stabilizatora U5 spotykają się tylko w jednym miejscu na płytce drukowanej. Dalej są prowadzone osobno i nie łączą się ponownie ani na płytce korektora, ani na płytce sterownika. Ma to kluczowe znaczenie!

KOREKTOR graficzny (schemat na **rysunku 2**) zrealizowany jest w oparciu o specjalizowane układy scalone TDA7317. Zawierają one stereofoniczny 5-kanałowy korektor graficzny oraz układ kontroli głośności. Konieczne jest dołączenie filtru dla każdego kanału i pasma, co sprowadza się do dołączenia 10 filtrów do każdego układu TDA7317. Parametry filtrów zostały pobrane z noty katalogowej układu i nieznacznie zmodyfikowane tak, aby łatwiej było kupić potrzebne elementy w sklepie.





Rys. 2 Schemat ideowy korektora

Zaproponowane pasma equalizera to: 60Hz, 250Hz, 1kHz, 3kHz oraz 10kHz.

Sterowanie układu TDA7317 odbywa się za pomocą magistrali I^{2}C, do czego wykorzystany został sprzętowy interfejs TWI wbudowany w mikrokontroler. Obecność wejścia adresowego ADD pozwala z wykorzystaniem jednej magistrali obsłużyć dwa układy, co w tym projekcie sprawdziło się bardzo dobrze.

Sygnały wejściowe są dołączane do złącz G1...G4 przez obwody ochronne.

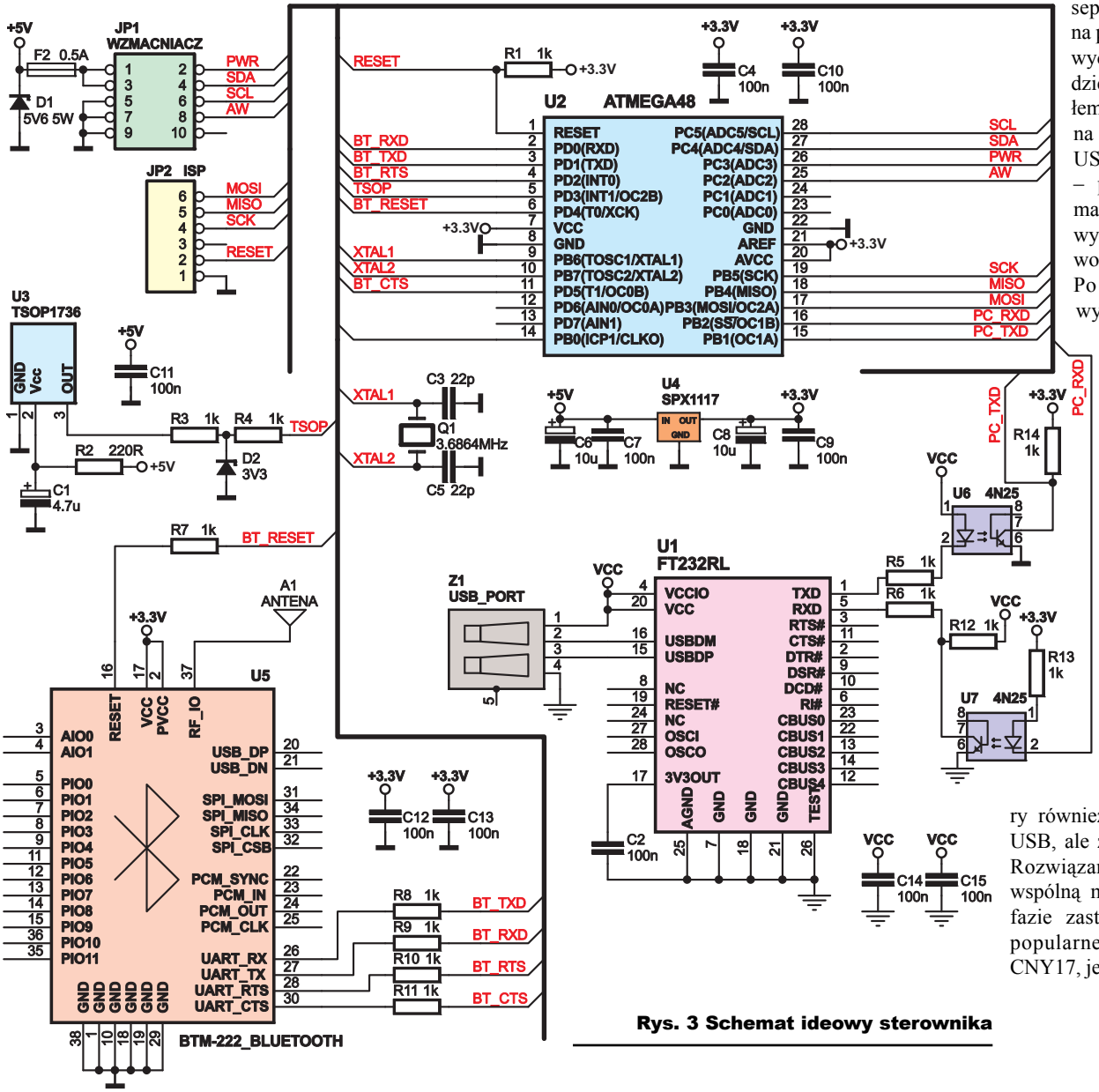
Na płytce znajdują się jeszcze dwa złącza: JP1 oraz JP2. Pierwsze z nich służy do podłączenia płytki sterownika. Za jego pośrednictwem przekazywane jest napięcie zasilające część cyfrową urządzenia oraz sygnał sterujące pracą przekaźnika. Obecna jest tutaj również magistrala I^{2}C kontrolująca pracę układów TDA.

Do drugiego złącza (JP2) należy podłączyć płytkę wzmacniacza, z której pobierane są napięcia zasilające płytkę sterownika, układy TDA oraz wyprowadzany jest sygnał sterujący przekaźnik.

STEROWNIK Najważniejszym elementem sterownika, którego schemat pokazano na **rysunku 3**, jest mikrokontroler ATmega8. Oprócz procesora obecny jest moduł Bluetooth – BTM-222, który służy do ustanowienia połączenia z telefonem komórkowym. Dostępny jest on w sprzedaży detalicznej w kilku sklepach internetowych (autor kupił go w cenie około 45zł). Od strony mikrokontrolera jest on widziany jak zwyczajny port szeregowy. Po uruchomieniu znajduje się on w trybie konfiguracyjnym i można ustawić parametry transmisji, adresy, etc. komendami AT przesyłanych przez układ USART procesora. Po ustanowieniu połączenia staje się zupełnie przezroczysty, co oznacza, że zyskujemy bezprzewodowy port szeregowy do wymiany danych z komputerem lub komórką. Jego wadą jest praca z napięciem 3,3V i niemożność podania napięć w standardzie 5V. Nie chciałem stosować różnorodnych konwerterów napięć, więc zdecydowałem, że cała część cyfrowa będzie pracować z napięciem 3,3V. Dostarcza go scalony stabilizator napięcia zbudowany na układzie U4.

Układ U3 jest popularnym odbiornikiem podczerwieni typu TSOP1736 i służy tu jako odbiornik sygnałów RC-5. Elementy C1, C11 oraz R2 tworzą filtr zasilania. Jedynie odbiornik podczerwieni nie może być zasilany napięciem 3,3V i wymaga do poprawnej pracy 5V. Wymusiło to zastosowanie prostego konwertera napięcia, który tworzą elementy R3, R4 i D2. Czytelnicy, którzy zdecydują się nie korzystać z tej drogi sterowania, mogą nie lutować elementów U3, C1, R2, R3, R4 i D2.

Na schemacie widoczny jest także układ scalony odpowiedzialny za konwersje USB<->RS232, konkretnie jest to FT232RL. Umożliwia on podłączenie BlueIR-a do komputera, który może sterować pracą wzmacniacza i odbierać polecenia sterujące pracą WinAMP-a. W tym miejscu ponownie dał o sobie znać problem mas. Do testów dźwięk był pobierany z karty dźwiękowej komputera, do którego podłączono również wzmacniacz przez port USB. W efekcie masa analogowa (karty dźwiękowej) i cyfrowa (portu USB) łączy się w urządzeniu w dwóch miejscach. Jakość dźwięku jest wtedy koszmarna – jest on tłumiony i pojawia się „dzwonienie". Po kilku godzinach spędzonych na wyszukiwaniu tego błędu i próbach jego eliminacji doszedłem do wniosku, że jedynym rozwiązaniem jest separacja galwaniczna portu USB. Sprawa wydawała się beznadziejna, gdyż nie znałem łatwego sposobu na oddzielenie portu USB od urządzenia – przesyłany sygnał ma charakter różnicowy i do tego częstotliwość liczoną w MHz. Po jakimś czasie wymyśliłem rozwiązanie, widoczne na schemacie. Za pomocą transoptorów oddzieliłem port szeregowy, który ma sygnał cyfrowy o małej częstotliwości. Zasilanie układu FT232 jest pobierane z portu USB, gdyż w innym wypadku nadal masy byłyby ze sobą połączone. Dopiero sygnały RXD i TXD są wprowadzane na transoptory również zasilane z portu USB, ale z „drugiej strony". Rozwiązanie to eliminuje wspólną masę. W pierwszej fazie zastosowałem bardzo popularne optoizolatory CNY17, jednakże okazały się

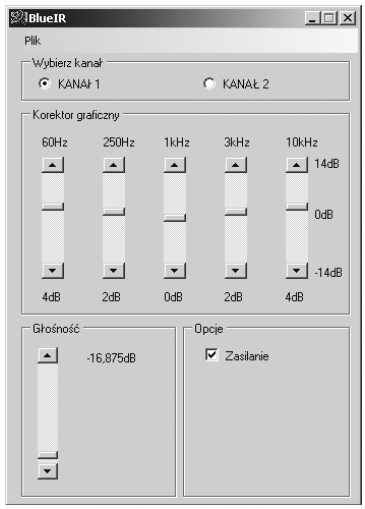


Rys. 3 Schemat ideowy sterownika

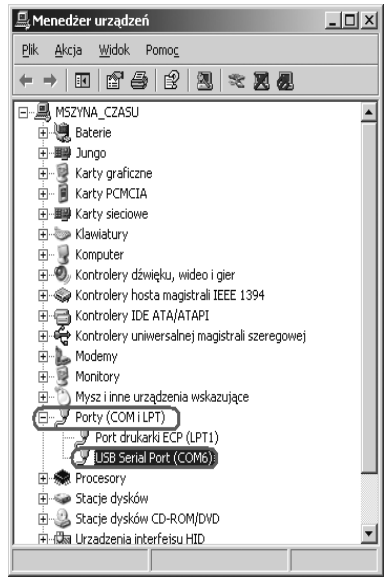
one za wolne i transmisja działała bardzo zawodnie. Dopiero zastosowanie szybszej wersji, konkretnie 4N25, pozwoliło uzyskać separację galwaniczną i dobrą jakość transmisji.

Na schemacie widoczne są jeszcze dwa złącza. Pierwsze z nich, JP2, służy do podłączenia programatora ISP. Zwracam uwagę, że po jego podłączeniu może ponownie pojawić się problem mas. Możesz się, Czytelniku, sam wtedy przekonać, co się dzieje, gdy nieprzestrzegana jest zasada mówiąca o łączeniu mas w jednym punkcie. Oczywiście wszystkie te uciążliwości ustaną po odłączeniu programatora.

Złącze JP1 służy do przyłączenie płytki korektora. Pobierane z niej napięcie 5V zasila układ TSOP oraz stabilizator U4. Umożliwia ono również wyprowadzenie magistrali I^2C służącej do sterowania układów TDA7317, a także portu włączającego przekaźnik głównego transformatora.



Rys. 4



Rys. 5

Rys. 6

Obsługa wzmacniacza

Praca wzmacniacza może być kontrolowana za pomocą komputera PC, pilota pracującego w kodzie RC5, bądź telefonem komórkowym obsługującym aplikacje napisane w Javie i komunikację przez Bluetooth.

Zacznijmy jednak od komputera. Zostało dla niego napisane specjalne oprogramowanie dostępne w Elportalu. Pierwszym krokiem powinno być umieszczenie do niego skrótu w autostarcie. Pozwoli to automatycznie uruchamiać program po włączeniu systemu i zapewni możliwość sterowania pracą WinAMP-a za pomocą pilota lub komórki. Okno tej aplikacji widoczne jest na **rysunku 4**. Aby zapewnić komunikację z urządzeniem, należy wybrać z menu *Plik* pozycję *Ustawienia* i w wyświetlonym okienku skonfigurować port szeregowy. Działanie to ogranicza się do wskazania, do którego portu podłączono urządzenie. Po kliknięciu *Otwórz* nastąpi próba otwarcia portu i gdy zakończy się powodzeniem zostanie wyświetlony stosowny komunikat, a wybrany port zapamiętany w pliku. Do głównego okna powraca się, klikając przycisk Powrót. Oczywiście wybór portu jest możliwy dopiero po uprzednim podłączeniu wzmacniacza.

W razie problemów z określeniem, który port wybrać, można wybrać *Start->Ustawienia->Panel sterowania->System*. W otwartym oknie należy kliknąć zakładkę *Sprzęt* i przycisk *Menedżer urządzeń*. Na liście urządzeń należy następnie rozwinąć zakładkę *Porty* i poszukać wpisu podobnego do przedstawionego na **rysunku 5**.

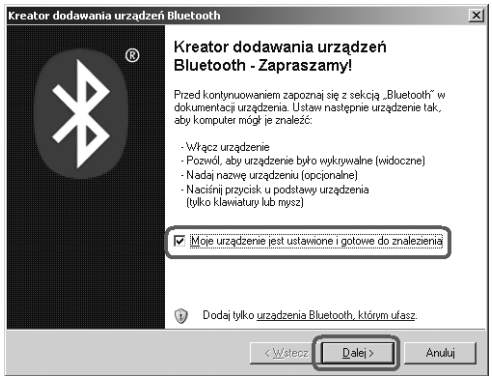
Okno aplikacji zostało podzielone na cztery części. Pierwsza z nich, nosząca nazwę *Wybierz kanał*, pozwala określić, który układ TDA ma być konfigurowany. Zmieniając parametry danego kanału, wpływamy na dźwięk obecny na dwóch wyjściach. Dla dźwięku stereo można podłączyć kartę muzyczną laptopa (lub innego odtwarzacza), tak aby jeden kanał trafiał do jednego układu TDA, a drugi do drugiego, co zapewni niezależną regulację barwy dźwięku i głośności w obu kanałach.

W polu *Korektor graficzny* istnieje możliwość tłumienia lub uwypuklenia dźwięków o danych częstotliwościach. W panelu *Głośność* można regulować siłę dźwięku dla danego kanału. Układ TDA zapewnia regulację w zakresie od 0 do około –18dB. Nie jest to dużo i nie jest możliwe całkowite wytłumienie sygnału przy skrajnej wartości. Z tego względu należy wstępnie wyregulować siłę dźwięku w systemie Windows lub podłączonym odtwarzaczu.

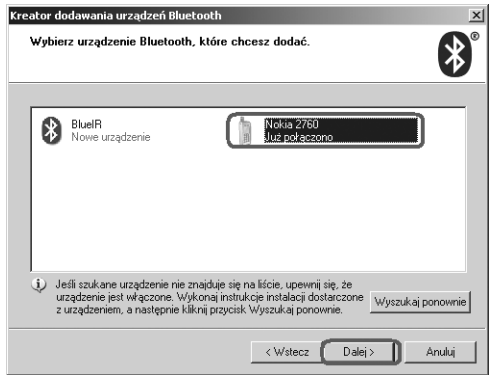
Ostatnie pole *Opcje* umożliwia włączenie bądź wyłączenie głównego transformatora. Uwaga! Należy korzystać z tej opcji do wyłączenia zasilania wzmacniacza. Dlaczego? Są dwa powody. Po pierwsze, mikrokontroler przechowuje bieżącą konfigurację układów TDA i po odłączeniu zasilania konieczne będzie dokonanie ponownych ustawień. Po drugie, tracimy w ten sposób możliwość jakiejkolwiek kontroli nad urządzeniem – czy to za pomocą pilota czy komórki. Jeżeli komputer pracuje przez większość dnia, to właściciel, siedząc na kanapie i czytając gazetę, w dowolnej chwili można sięgnąć po telefon i włączyć muzykę. Bez zasilania jest to niemożliwe.

Chcąc sterować wzmacniaczem z użyciem komórki, należy pobrać z Elportalu oprogramowanie napisane w Javie i załadować je do telefonu. Interesuje nas konkretnie plik *BlueIR.jar*. Trudno podać uniwersalną receptę, w jaki sposób przesłać program do komórki, dlatego pokażę, krok po kroku, jak dokonać takiego manewru, posiadając Bluetootha w laptopie, Nokię 2760 i Windows XP. Zaczynamy od włączenia w telefonie modułu Bluetooth, w moim wypadku sprowadziło się to do wybrania z menu *Ustawienia->Łączność->Bluetooth->Bluetooth* i zmiany opcji z *nie* na *tak* (na samej górze ekranu pojawiła się niebieska ikonka).

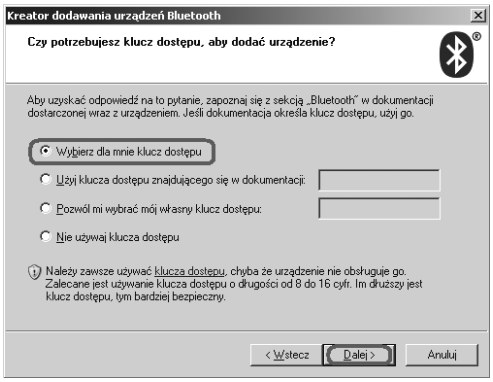
Rys. 7



Rys. 8

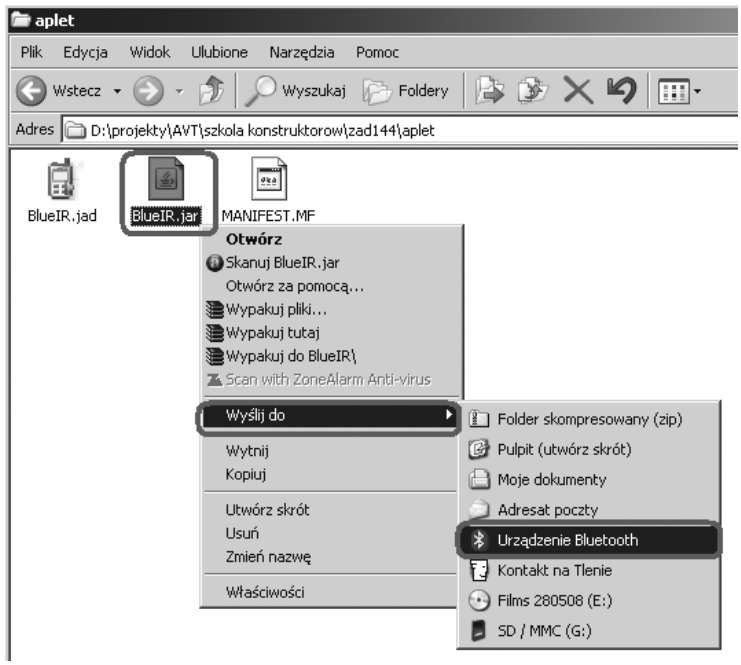


Teraz pozostało odnaleźć telefon i dokonać autoryzacji w Windowsie. Sprowadza się to do wybrania *Start->Ustawienia->Panel sterowania->Urządzenia Bluetooth*. Naszym oczom powinno ukazać się okienko z **rysunku 6**, w którym klikniemy, a jakże, przycisk Dodaj. Efektem będzie pojawienie się kolejnego okienka (**rysunek 7**). w którym zaznaczamy opcję *Moje urządzenie jest ustawione i gotowe do znalezienia* i klikamy Dalej. Jak nietrudno się domyślić, otworzy się kolejne okienko, w którym będą na bieżąco wyświetlane wyszukane urządzenia. Należy uzbroić się w cierpliwość, gdyż może to zająć parę chwil. Po odszukaniu komórki należy kliknąć jej symbol (**rysunek 8** – zwróć uwagę, że stojący w pobliżu wzmacniacz również został wykryty!) i nacisnąć przycisk Dalej. W kolejnym oknie skorzystamy z domyślnej opcji i poprosimy system o wygenerowanie dla nas klucza – **rysunek 9**. Klikamy *Dalej* i na ekranie komórki pojawia się pytanie, czy dokonać uwierzytelnienia, zgadzamy się i wprowadzamy wygenerowany przez komputer klucz. Akceptujemy i twierdząco odpowiadamy również na pytanie o automatyczne nawiązywanie połączenia. Tym samym proces uwierzytelniania został zakończony (w ostatnim okienku kreatora klikamy po prostu Zakończ).



Rys. 9

Jak teraz przesłać program do komórki? Prościej niż mogłoby się wydawać – wystarczy kliknąć prawym przyciskiem myszy na pliku *BlueIR.jar* i wybrać kolejno *Wyślij do*, a następnie *Urządzenie Bluetooth* (**rysunek 10**). W otwartym oknie upewniamy się, że wysyłamy plik do komórki (pole *Wyślij do*) i po tym klikamy Dalej. To wszystko! Komórka powinna poinformować stosownym komunikatem i dźwiękiem o odebraniu aplikacji.



Rys. 10

Pozostało już tylko uruchomić program i zacząć go używać. Po uruchomieniu aplikacji rozpocznie się wyszukiwanie urządzeń Bluetooth pracujących w okolicy. Wszystkie odnalezione urządzenia są dodawane do listy, po której można się przemieszczać. Po pojawieniu się wpisu *BlueIR* wybieramy go, a następnie *OK*. Prawdopodobnie komórka wyświetli pytanie, czy użytkownik zezwala na wykorzystywanie łącza, oczywiście zgadzamy się na to. Po chwili zapyta także, czy dokonać uwierzytelnienia, na które wyrażamy zgodę i poprosi o hasło (stanowi je numer *1234*). Po tych operacjach połączenie powinno zostać nawiązane i ukaże się główne menu. Warto zauważyć, że w prawym górnym rogu każdego pola wyświetlana jest cyfra. Wybranie jej z klawiatury spowoduje wykonanie stosownego polecenia – szczegółowy opis znajduje się w **tabelce 1**.

Sterowanie za pomocą pilota RC5 jest zbliżone do sterowania komórką, tzn. odbywa się z wykorzystaniem przycisków z cyframi od 0...9. Polecenia pokrywają się w większości z tymi dostępnymi dla komórki – patrz **tabela 2**. Są jednak pewne różnice. Po pierwsze, pilot nie pozwala na sterowanie korektorem graficznym. Drugą różnicą jest sterowanie głośnością – wprowadzane zmiany oddziałują na OBA kanały jednocześnie.

Dźwięk do wzmacniacza jest wprowadzany przez złącza chinch oznaczone symbolami G1..G4 (płytka korektora). Głośniki są podłączane do złączy śrubowych Z2, Z4, Z5 oraz Z6 zlokalizowanych na płytce wzmacniacza. Podłączanie i odłączanie przewodów MUSI ODBYWAĆ SIĘ PRZY ODŁĄCZONYM NAPIĘCIU!!! Na płytce wzmacniacza obecne jest pełne napięcie sieci i chwila nieuwagi może skończyć się tragicznie.

Jakub Borzdyński
jakub.borzdynski@elportal.pl

Ciąg dalszy w następnym numerze.

Tabela 1. Polecenia w głównym menu telefonu

Klawisz	Podejmowana akcja
1	Otwiera okienko, w którym za pomocą suwaków ustawia się głośność pierwszego i drugiego kanału. Do głównego okna wraca się, wybierając polecenie *Powrót*.
2	Otwiera domyślny odtwarzacz systemu Windows (w przypadku WinAMP-a razem z ostatnią playlistą).
4	Otwiera menu korektora graficznego. Znajduje się tu pięć suwaków do kontroli pięciu pasm. Na samej górze znajduje się pole pozwalające wybrać kanał, którego mają dotyczyć wprowadzone zmiany. Do głównego okna wraca się, wybierając polecenie *Powrót*.
6	Włączenie bądź wyłączenie głównego transformatora (zależnie od aktualnego stanu).
7	Poprzedni utwór z playlisty (PREV).
8	Wstrzymanie bądź rozpoczęcie odtwarzania (PLAY/PAUSE).
9	Następny utwór z playlisty (NEXT).
0	Zatrzymaj odtwarzanie (STOP).

Tabela 2. Polecenia pilota RC5

Klawisz	Podejmowana akcja
1	Zwiększa głośność w obu kanałach.
2	Otwiera domyślny odtwarzacz systemu Windows (w przypadku WinAMP-a razem z ostatnią playlistą).
4	Zmniejsza głośność w obu kanałach.
6	Włączenie bądź wyłączenie głównego transformatora (zależnie od aktualnego stanu).
7	Poprzedni utwór z playlisty (PREV).
8	Wstrzymanie bądź rozpoczęcie odtwarzania (PLAY/PAUSE).
9	Następny utwór z playlisty (NEXT).
0	Zatrzymaj odtwarzanie (STOP).