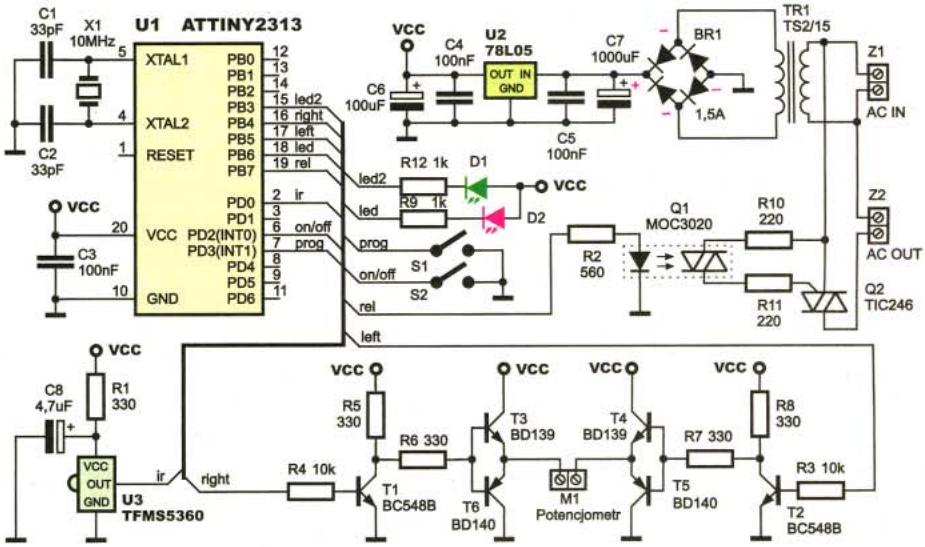
Uniwersalny sterownik wzmacniacza audio

Na łamach EdW wielokrotnie pojawiały się projekty różnych końcówek mocy i wzmacniaczy audio. Były to projekty bardzo proste w realizacji (wykonane na układach scalonych) oraz znacznie trudniejsze i kosztowniejsze wzmacniacze tranzystorowe i lampowe. Nie spotkałem się natomiast ze sterownikiem do takiego wzmacniacza. Nie chodziło o przedwzmacniacz, tylko o układ, który między innymi zarządzałby zasilaniem. Przede wszystkim zależało mi na możliwości sterowania wzmacniaczem za pomocą pilota. Najlepiej, gdyby łatwo było zainstalować taki sterownik w sprzęcie fabrycznym. Wiadomo, potrzeba jest matką wynalazków. Dlatego, Drogi Czytelniku, masz teraz okazję czytać ten tekst.

Opis układu

Na **rysunku 1** został przedstawiony schemat ideowy sterownika. Na **rysunku 2** widoczny jest schemat połączenia sterownika ze wzmacniaczem audio. Sterowanie odbywa się za pomocą dowolnego, fabrycznego pilota IR. Stwierdziłem, że bezcelowe jest budowanie kolejnego pilota specjalnie na potrzeby tego projektu. Prawie pewne jest, że w pomieszczeniu oprócz wzmacniacza znajdować się będzie inne urządzenie sterowane za pomocą własnego pilota. Można także kupić tanie piloty uniwersalne. Takie podejście jest bardziej estetyczne i profesjonalne.

Jako regulator głośności zastosowałem potencjometr z silnikiem. O jego wyborze zadecydował szereg czynników:
– separacja galwaniczna obwodów sterownika i części audio,
– bardzo duża precyzja regulacji głośności (płynna regulacja),



Rys. 1 Schemat ideowy

– tańsze potencjometry elektroniczne o niezadowalającej jakości,
– trudności z dostępem do lepszych potencjometrów elektronicznych i ich koszt,
– trudność wykonania dobrego potencjometru elektronicznego z elementów dyskretnych,
– łatwość sterowania,
– możliwość zastosowania sterownika w sprzęcie fabrycznym z minimalną ingerencją w oryginalną elektronikę,
– subiektywna opinia o takim rodzaju sterowania głośnością.

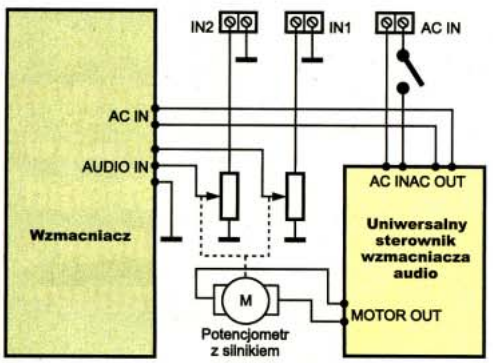
Część układowa sterownika nie jest skomplikowana. Sercem układu jest mikrokontroler ATTINY2313. Układ U3 TFMS5360 jest odbiornikiem podczerwieni i pracuje w typowej konfiguracji. Mikroprocesor i odbiornik IR zasilane są poprzez transformator sieciowy i stabilizator U2 78L05. Sterowanie silnikiem potencjometru odbywa się za pomocą 6 tranzystorów i 6 rezystorów. Taki sposób podłączenia umożliwia obrót w dowolnym kierunku i jest ekonomiczny. Gwarantuje także niewystąpienie zwarcia na wypadek podania wysokiego potencjału na obydwie gałęzie sterujące.

Program mikrokontrolera został napisany w języku Basic ze względu na łatwość programowania i można go ściągnąć z Elportalu. O wyborze środowiska Bascom zadecydował także wbudowany w kompilator programator. Chcąc bardzo ogólnie opisać działanie programu, wyglądałoby to następująco. Procesor jest cały czas w stanie Idle. Co ok. 200µs przerwanie *timera* budzi układ. Następuje sprawdzenie, czy pojawiła się zmiana stanu

na wyjściu odbiornika IR (U3), oznaczająca rozpoczęcie nadawania. Jeśli tak, to podczas kolejnych przerwań następuje zbieranie informacji o nadawanym kodzie. Trwa to do momentu stwierdzenia zakończenia nadawania (dłuższy brak zmian na wyjściu odbiornika IR). Następnie zebrane informacje są porównywane z informacjami zapisanymi w pamięci EEPROM. Jeśli kod odebrany pasuje do kodu w pamięci, zostanie wykonane odpowiednie działanie.

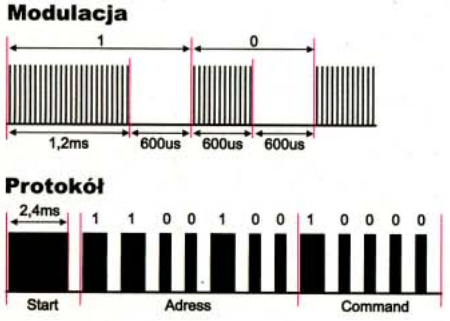
Omówię teraz najważniejszą część programu, czyli sposób gromadzenia i przetwarzania informacji o odbieranym kodzie pilota IR. Program początkowo był pisany pod obsługę jednego standardu kodowania – Sony SIRC, a to ze względu na popularność i powszechność sprzętu elektronicznego firmy Sony. Protokół ten został zaprezentowany na **rysunku 3**. Jednak ostatecznie program jest bardzo uniwersalny i umożliwia pracę z praktycznie dowolnym pilotem. Jest to możliwe, gdyż program nie interpretuje kodów, tylko zapamiętuje długość pojedynczych impulsów i odstępów pomiędzy nimi.

Uniwersalność sterownika okazała się kłopotliwa. Pojawił się problem z formatami innych standardów kodowania niż Sony SIRC, o bardzo długim protokole, np. NEC, popularnego w sprzęcie produkcji japońskiej. Protokół NEC został zaprezentowany na **rysunku 4**. W pamięci EEPROM muszą znaleźć się 3 różne rozkazy: włączenie/wyłączenie zasilania, obrót potencjometru w prawo, obrót potencjometru w lewo. Natomiast mikrokontroler ATTINY2313 posiada tylko 128 bajtów nieulotnej pamięci EEPROM. Próba interpretacji i odrzucania powtarzającej się części kodu (patrz protokół NEC) nie wchodzi w grę, gdyż godziłoby to w postulat uniwersalności urządzenia. W tym wypadku mogłem zastosować większy mikrokontroler lub pamięć zewnętrzną. Oznaczało to niestety konieczność przeprojektowania płytki i oczekiwanie na jej ponowne wytworzenie. Zrezygnować z uniwersalności lub… zastosować prostą kompresję danych. Zrealizowałem, z powodzeniem, drugą możliwość.



Rys. 2 Schemat blokowy przyłączenia do wzmacniacza

Rys. 3 Protokół Sony SIRC



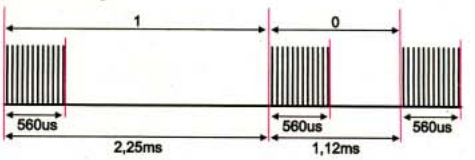
Tylko dla dociekliwych

Sygnał próbkowany jest co ok. 200µs. Dlatego standardowo długość impulsu lub przerwy nie będzie dłuższa niż 16 próbek (dla protokołu NEC długość przerwy dla logicznej jedynki będzie równa ok. 8,5x200µs). Oznacza to, że można wykorzystać do zapisu takiej informacji tylko pół bajtu (4 bity). Trzeba oczywiście jeszcze mieć możliwość zapisu i rozpoznania, czy w danym bajcie znajdują się 2 różne informacje 4-bitowe, czy pojedyncza 8-bitowa. Na przykład impuls inicjalizujący transmisję rozkazu w protokole NEC ma długość 9ms = 45x200µs i musi być zapisany na 8 bitach. W tym celu w pamięci EEPROM została wydzielona przestrzeń, w której zapisywany jest format bajtów zawierających właściwą informację. Na **rysunku 5** można zobaczyć mapę pamięci EEPROM. Rysunek ten pozwoli łatwiej zrozumieć ideę pomysłu. Każdemu bajtowi rozkazu jest przyporządkowany jeden bit formatu. W wyniku obliczeń optymalizacyjnych okazało się, że pamięć jest najlepiej wykorzystana, gdy informacja o rozkazie ma długość 37 bajtów. W takim wypadku tylko 2 bajty pamięci EEPROM są niewykorzystywane.

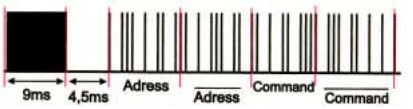
Kompresja wykonywana jest w locie. Do pamięci RAM trafiają już zakodowane informacje. Procedura kompresująca przedstawiona została na **listingu 1.** Zmienna *Licznik_pom* zawiera ilość przerwań timera przypadających na czas trwania określonego stanu logicznego (przerwy lub impulsu). Przyjrzyjmy się w jaki sposób program dokonuje kompresji na przykładzie protokołu NEC. Impuls inicjujący trwa 9ms. Przerwanie *timera* występuje co 200µs, czyli zmienna *Licznik_pom* zawiera wartość 45±1. Ponieważ jest to pierwszy odebrany impuls, indeksy wskazują na pozycję „–1" tablicy danych. Następuje preinkrementacja indeksów, ustawienie formatu bajtu danych na 8-bitowy, przepisanie wartości zmiennej *Licznik_pom* do tablicy, postinkrementacja indeksów, wyzerowanie kolejnego bajtu i zmiana jego formatu na 4-bitowy. Teraz, kiedy znów nastąpi zmiana stanu na wyjściu odbiornika IR, w zmiennej *Licznik_pom* będzie znajdować się długość przerwy: 4,5ms, czyli 22±1. Znów następuje zmiana formatu na 8-bitowy, kopiowanie i postinkrementacja indeksów, czyszczenie kolejnego bajtu i zmiana jego formatu na 4-bitowy. Kolejny impuls ma długość 560µs (zmienna *Licznik_pom* zawiera wartość 2±1) i jest zapisywana do tablicy bez zmiany formatu i postinkrementacji. Załóżmy, że odbieramy logiczną 1. Długość przerwy wynosi 1,69ms (zmienna *Licznik_pom* zawiera wartość 8±1). Następuje przesunięcie zawartości bajtu o 4 bity w lewo i wprost dodanie nowej informacji do bieżącego bajtu. Na końcu postinkrementacja indeksów, wyzerowanie kolejnego bajtu i zmiana jego formatu na 4-bitowy. Cykl powtarza się, aż indeks tablicy danych przekroczy wartość 37 lub długość przerwy pomiędzy impulsami będzie większa niż 25ms.

Kolejne działanie zależy od wcześniejszych ustawień (zapis do pamięci EEPROM czy porównanie). Jeśli jest to zapis, następuje przepisanie zawartości pamięci RAM wprost do pamięci EEPROM. Jeśli jest to porównanie, to następuje porównywanie (3-krotne, na każdą instrukcję osobno) bajt po bajcie zawartości pamięci RAM i EEPROM (z uwzględnieniem odpowiedniej tolerancji). Jeśli weryfikacja przebiegła pomyślnie, następuje wykonanie właściwej procedury obsługi np. obrót potencjometru. Podczas trwania

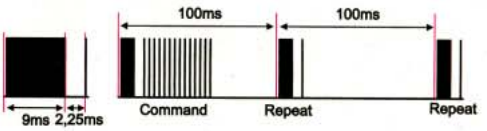
Modulacja



Protokół



Kod powtarzania



Rys. 4 Protokół NEC

Rys. 5 Mapa pamięci EEPROM

obrotu układ sprawdza dodatkowo, czy nie zostały odebrane kody powtórzenia (patrz protokół NEC, **rysunek 4**). Wyłączenie silnika następuje dopiero po dłuższej nieaktywności odbiornika IR.

Montaż i uruchomienie

Układ należy zmontować na płytce przedstawionej na **rysunku 6**. Na początku montujemy elementy najniższe, kończymy na wlutowaniu transformatora i włożeniu zaprogramowanego procesora do podstawki. Bardzo ważne jest odpowiednie skonfigurowanie Fuse Bitów procesora ATTINY2313. W przeciwnym wypadku układ nie będzie pracował lub będzie pracował błędnie. Następujące bity należy zmienić:

- Fusebit C: 1: Divide clock by 8 disabled
- Fusebit A987: 1111: 1111 external XTAL
- Fusebit GFE: 100: BOD 4.3V

Pozostałe Fuse Bity pozostawiamy bez zmiany. Przycisk S1 montujemy wprost na płytce, natomiast S2 łączymy z płytką za pomocą krótkiego kabla dwużyłowego (jest to przycisk zasilania, który zostanie umieszczony na obudowie). Potencjometr z silnikiem powinien być łączony z płytką poprzez złącze śrubowe. Ułatwi to późniejszy demontaż i serwisowanie. Podobnie przewody zasilania.

Układ możemy zamontować we wnętrzu już gotowego wzmacniacza, zgodnie z koncepcją z rysunku 2. Należy rozciąć przewód zasilania wzmacniacza i włączyć pomiędzy nasz sterownik. Oryginalny potencjometr wzmacniacza można zastąpić nowym, zdalnie sterowanym, lub dodać nowy, dodatkowy potencjometr, włączając go na wejściu wzmacniacza. Układ sterownika można także zabudować i umieścić w oddzielnej obudowie, dodając odpowiednie gniazda wejścia i wyjścia.

Po włączeniu zasilania należy przygotować układ do pracy z wybranym pilotem. W tym celu trzeba przycisnąć przycisk S1 (ten na płytce). Następnie zbliżyć pilot i nacisnąć wybrany przycisk. Układ trzy razy zaświeci zieloną diodę, co oznacza odebranie i zapamiętanie kodu. Po czym przyciskamy inny przycisk pilota itd. Układ uczy się komend w następującej kolejności:

– włączenie/wyłączenie zasilania,
– obrót potencjometru w lewo,
– obrót potencjometru w prawo.

Być może trzeba będzie powtórzyć proces programowania w wypadku, kiedy po naciśnięciu przycisku *Głośniej* potencjometr obróci się w nieprawidłową stronę. Należy wtedy ponownie zaprogramować układ, zamieniając kolejność odpowiednich przycisków. Teraz trzeba już tylko sprawdzić, czy układ prawidłowo reaguje na wszystkie zaprogramowane komendy.

Jeszcze jedna mała uwaga: trzeba mieć świadomość, że zastosowany w modelu potencjometr z silnikiem wymaga minimalnego napięcia zasilania 2V. Gdyby ktoś zastosował

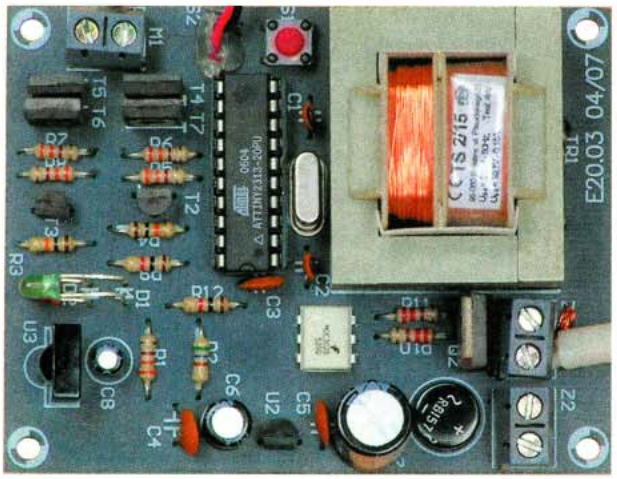
Listing 1 Listing procedury kompresującej dane

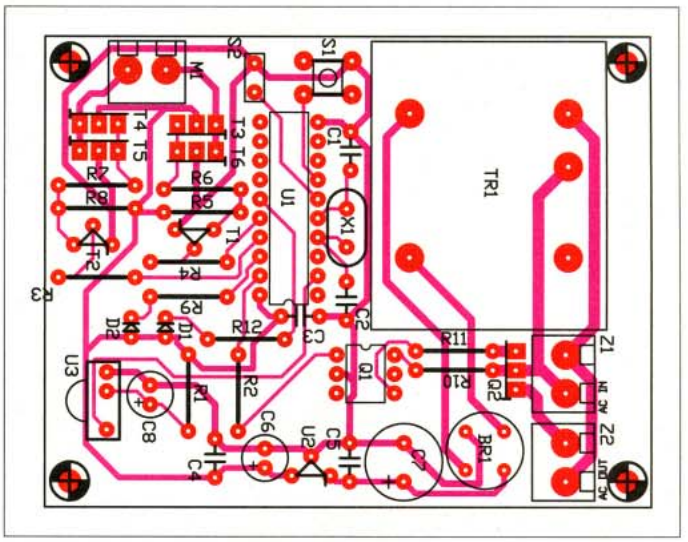
```
'-------------------------------------------------------------
' funkcja przesuwa indeksy o "jeden" w gore,
' znaczenia indeksow zostaly opisane przy okazji
' omawiania funkcji Zapis

Shift_indexs:
          Incr J
          If J = 8 Then
             J = 0
             Incr I
          End If
          Incr Index
Return

'-------------------------------------------------------------
' funkcja "zeruje" bajt danych umozliwiajac wpisanie poprawne
' nowej informacji

Clear_bajt:
          A(index) = 0                          'wyczysc caly bajt
          V_i(i).j = 1                          'formatujemy bajt do zapisyu dwoch liczb 4 bitowych
Return

'-------------------------------------------------------------
' funkcja koduje odebrana dlugosc impulsu (lub przerwe pomiedzy
' impulsami) i zapisuje tymczasowo do pamieci RAM. Zapis i
' weryfikacja z pamiecia EEPROM trwa za dlugo i bedzie
' wykonana po odczytaniu calego rozkazu.
'  - zmienna Licznik_pom zawiera dlugosc impulsu
'    (lub przerwy pomiedzy impulsami)
'  - tablica A(index) reprezentuje przestrzen w pamieci RAM,
'    do ktorej nastepuje tymczasowy zapis rozkazu
'  - tablica V_i(i).j zawiera informacje czy
'    biezacy bajt z tablicy A(index) zawiera 2 krotkie
'    liczby, czy jedna dluga (kazdy bit tej tablicy odpwiada
'    odpowiedniemu bajtowi tablicy A(index))

Zapisz:

                                                'jesli dlugosc impulsu (lub przerwy) przekracza 16
                                                jednostek czasu, to musi on zajac caly bajt
   If Licznik_pom >= 16 Then

      If A(index) = 0 Then                      'jesli biezacy bajt w pamieci RAM zawiera 0

         A(index) = Licznik_pom                 'wpisz "odebrany" bajt informacji
                                                 V_i(i).j = 0
                                                'formatujemy bajt do zapisu jednej liczby 8-bitowej
         Gosub Shift_indexs                     'przejdz do kolejnego bajtu
         Gosub Clear_bajt                       'wyczysc ten bajt

      Else                                      'jesli biezacy bajt zawiera już jakąś informację

         Gosub Shift_indexs                     'przejdz do nastepnego bajtu
         A(index) = Licznik_pom                 'zapisz zawartosc zmiennej pomocniczej do tablicy
         V_i(i).j = 0                           'formatujemy bajt do zapisu jednej liczby 8-bitowej
         Gosub Shift_indexs                     'przejdz do kolejnego bajtu
         Gosub Clear_bajt                       'wyczysc ten bajt

      End If

                                                'jesli dlugosc impulsu (lub przerwy) NIE przekracza
                                                16 jednostek czasu, to moze on zajac tylko 4 bity
   Else

         'poniewaz jesli bajt zostaje calkowicie zapelniony, to zawsze nastapuje przejscie
         'do nastepnego pustego bajtu, dlatego nie ma potrzeby sprawdzena bajtu na wypadek,
         'w ktorym caly bajt bylby pelny

         If A(index) < 16 And A(index) > 0 Then 'jesli biezacy bajt jest czesciowo zapelniony

            Shift A(index) , Left , 4           'przesuń stara czesc do gory (przesuniecie
                                                'biezacej zawartosci bajtu o 4 pozycje w lewo)
            A(index) = A(index) + Licznik_pom   'dodaj do biezacego bajtu nowa informacje
            Gosub Shift_indexs                  'przejdz do kolejnego pustego bajtu
            Gosub Clear_bajt

         Else                                   'jesli bajt jest pusty (zawiera 0)

           A(index) = Licznik_pom               'wpisz liczbe do bajtu
                                                '!!! brak przejscia do kolejnego bajtu, bo bajt
                                                moze pomiescic jescze jedna liczbe 4-bitowa

         End If

   End If

   Licznik_pom = 0                              'zacznij zliczanie czasu od nowa
Return
```

Wykaz elementów

Kondensatory	D2 ..... LED czerwona
C1,C2 ..... 33pF	T1,T2 ..... BC548B
C3-C5 ..... 100nF	T3,T4 ..... BD139
C6..... 100µF	T5,T6 ..... BD140
C7 ..... 1000µF/16V	U1 ..... ATTINY2313
C8 ..... 4.7µF	U2..... 78L05
Rezystory	U3 ..... TFMS5360
R1 ..... 330Ω	Q1 ..... MOC3020
R2 ..... 560Ω	Q2 ..... TIC246
R3,R4 ..... 10kΩ	**Pozostałe**
R5-R8 ..... 330Ω	S1,S2 .... mikroswitch
R9,R12..... 1kΩ	TR1 ..... TS2/15
R10,R11..... 220Ω	X1 ..... kwarc 10MHz
Półprzewodniki	Z1,Z2,M1:..... ARK2
BR1..... mostek okrągły	Potencjometr z silnikiem
D1 ..... LED zielona	należy zakupić oddzielnie

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2829.





Rys. 6 Schemat montażowy

inny model, o większym wymaganym napięciu pracy, trzeba będzie wprowadzić zmiany na płytce.

Mimo iż układ jest przystosowany do pracy z praktycznie dowolnym pilotem, istnieją wyjątki. Urządzenie słabo współpracuje z pilotami firmy Technics. Zdarzało się, że reagowało na nieprzypisany przycisk lub nie chciał współpracować z pewną grupą przycisków. Układ został przetestowany i pracuje prawidłowo z pilotami od urządzeń następujących producentów: Sony, Pioneer, Samsung, Nexus, Sanyo, DVB, PixelView. Niezwykle rzadko zdarzały się przypadki, kiedy nie reagował na przycisk. Były one na tyle rzadkie, iż nie powinno to stanowić problemu przy normalnej eksploatacji.

Paweł Knioła
pawel.kn@wp.pl