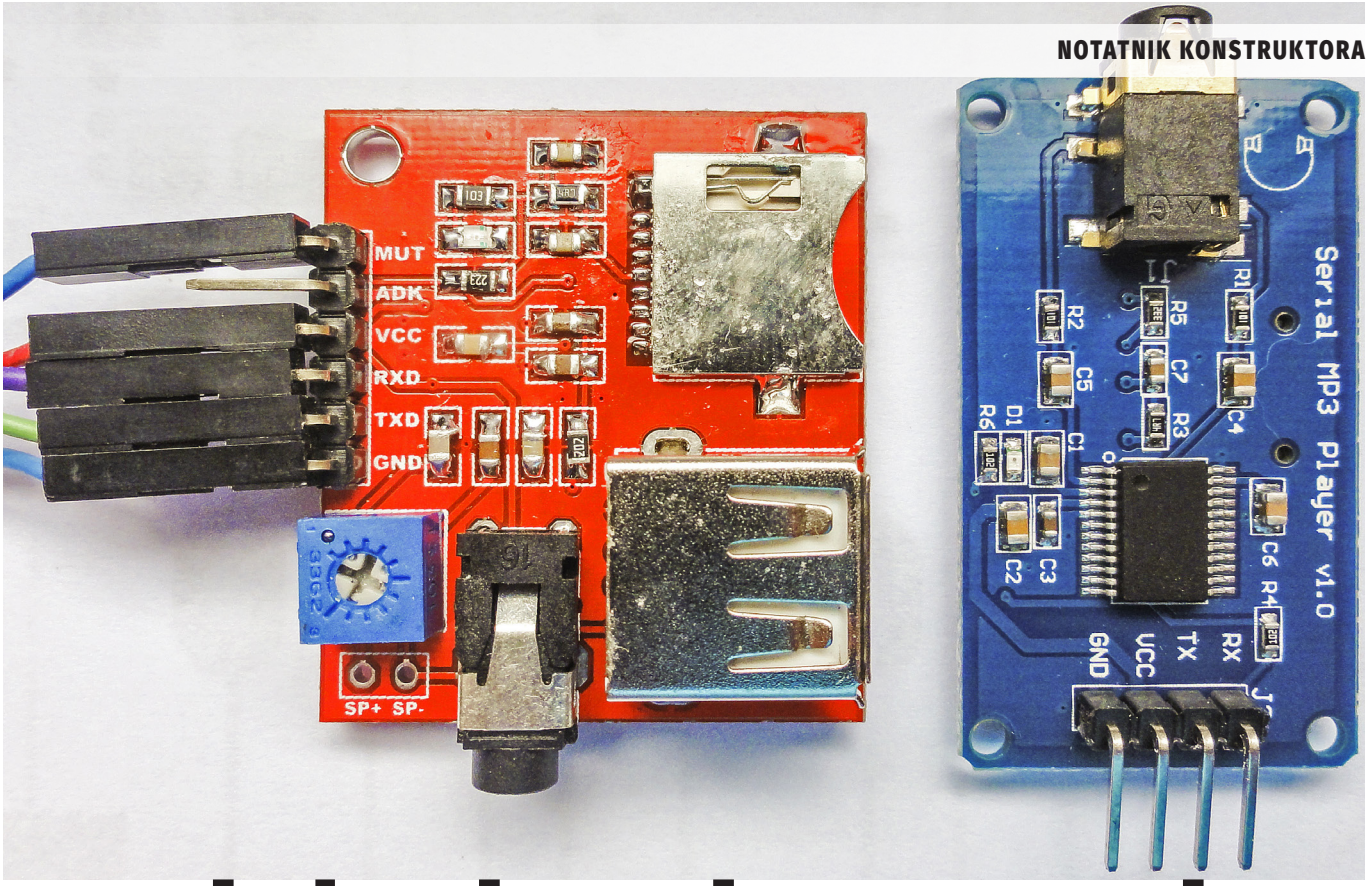



# Moduły do odtwarzania plików mp3

*Niegdyś na łamach Elektroniki Praktycznej był opisywany słynny odtwarzacz Yampp. Yampp został zaprojektowany przez Jespera Hansena, a oprogramowanie do niego było rozwijane przez Romualda Białego. Dzisiaj budowanie takich odtwarzaczy plików mp3 nie ma większego sensu. Każdy komputer, tablet czy smartfon potrafi je odtwarzać z lepszą lub gorszą jakością. Są jednak zastosowania, w których wykorzystanie urządzeń mobilnych do odtwarzania zapisanych wcześniej materiałów dźwiękowych jest niemożliwe lub bardzo utrudnione.*

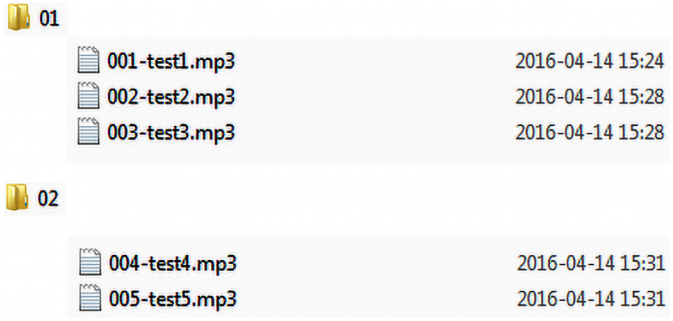
Załóżmy, że mamy zbudować urządzenie, które musi informować obsługę o ważnych zdarzeniach za pomocą komunikatów głosowych. W przypadku, kiedy to urządzenie będzie wykorzystywać komputer, lub moduł embeded z wbudowanym systemem operacyjnym np. Linux, to sprawa jest w miarę prosta. Ale kiedy nie mamy do dyspozycji takich środków, to nawet przy zastosowaniu prostego mikrokontrolera można sobie poradzić dołączając kartę pamięci SD i sprzętowy kodek. Jednak wtedy potrzebne będzie napisanie oprogramowania odtwarzania. Będzie to zadanie tym trudniejsze im „słabszy" to będzie mikrokontroler. Najprostszym i zarazem najtańszym rozwiązaniem może się okazać zastosowanie specjalizowanego modułu. Jednym z takich modułów jest moduł firmy Catalex z układem YX5300.

## Moduł z układem YX5300

Moduł z układem YX5300 jest oferowany przez firmę CATALEX. Po krótkim przeszukaniu Internetu nie znalazłem ani strony tej firmy, ani dokumentacji samego układu YX5300. Jest za to dostępny dokument Serial MP3 Player Manual zupełnie wystraczający do zaaplikowania układu.

Za kompletną płytka z układem YX5300 , stereofonicznym gniazdem słuchawkowym Jack 3,5 mm i złączem karty microSD zapłaciłem razem z przesyłką ok. 5 USD. Jest to cena bardzo atrakcyjna zwarzywszy na możliwości układu:

- Możliwość odtwarzania plików skompresowanych w formacie mp3 i nieskompresowanym WAV.
- Wspierane częstotliwości próbkowania: 8 kHz, 11,025 kHz, 12 kHz, 16 kHz, 22,05 kHz, 24 kHz, 32 kHz, 44,1 kHz i 48 kHz.
- Możliwość użycia kart Micro SD (maksymalnie 2 GB) i Micro SDHC (maksymalnie 32 GB). Format plików FAT16 lub FAT32.
- Sterowanie odtwarzaniem z użyciem interfejsu szeregowego UART pracującego z prędkością 9600 b/s.
- Poziomy logiczne na magistrali UART 3,3 V lub 5 V.




**Rysunek 1. Struktura plików na karcie SD**

- Napięcie zasilania 3,2...5,2 V. Maksymalny pobór prądu 200 mA przy 5 V DC.
- Wbudowany wzmacniacz słuchawkowy.
- Cyfrowa regulacja poziomu sygnału audio w 30 krokach.

Oprócz wspomnianego gniazda słuchawkowego Jack na płytce umieszczono 4-pinową listwę golpinów przeznaczona do doprowadzenia zasilania: piny GND i VCC oraz linii interfejsu UART (RX i TX). Kierunek przesyłania danych jest oznaczony z punktu widzenia modułu. Zielona dioda D1 sygnalizuje gotowość do odtwarzania, lub pauzę odtwarzania (ciągłe świecenie), lub stan odtwarzania pliku (świecenie przerywane).

## Komendy sterujące

Odtwarzacz wymaga, by pliki i katalogi na karcie SD miały nazwy zaczynające się od cyfr. Nazwa katalogu może być wyłącznie liczbą dwucyfrową na przykład 01, 02, 11, 23 itp. Nazwa każdego pliku musi

**Tabela 1. Format komend**

| | Bajt (hex) | opis |
|---|---|---|
| $S | 0x7E | Bajt startu zawsze 0x7E |
| VER | 0xFF | Informacja o wersji |
| LEN | 0x06 | Ilość bajtów komendy bez bajtów startu o końca |
| CMD | -- | Kod komendy |
| Feedback | 0x00, lub 0x01 | Potwierdzenie komendy 00- bez potwierdzenia, 01 z potwierdzeniem |
| Data[0]<br>Data[1]<br>....<br>Data[n] | -- | Parametry komendy – zazwyczaj 2 bajty, ale są dłuższe |
| $0 | 0xEF | Bajt końca komendy |

**Tabela 2. Komendy sterujące**

| Komendy odtwarzania plików | Bajty komendy bez sumy CRC | Opis |
|---|---|---|
| Następny utwór | 7E, FF, 06, 01, 00, 00, 00, EF | Odtwarzaj następny utwór |
| Poprzedni utwór | 7E, FF, 06, 02, 00, 00, 00, EF | Odtwarzaj poprzedni utwór |
| Odtwarzanie pliku z indeksem | 7E, FF, 06, 03, 00, 00, 01, EF<br>7E, FF, 06, 03, 00, 00, 02, EF | Odtwarzaj pierwszy plik<br>Odtwarzaj drugi plik |
| Odtwarzaj pliki na podstawie numeru folderu i pliku w folderze | 7E, FF, 06, 0F, 00, 01, 01, EF<br>7E, FF, 06, 0F, 00, 01, 02, EF | Odtwarzaj plik 001-test1.mp3 z katalogu 01<br>Odtwarzaj plik 002-test2.mpe z katalogu 01<br>Itp. |
| Cykliczne odtwarzanie plików w folderze | 7E, FF, 06, 17, 00, 00, 01, EF | Cykliczne odtwarzanie plików w folderze 01 |
| Grupowe odtwarzanie plików z indeksem | 7E, FF, 09, 21, 00, 05, 01, 02, 03, 04, EF<br>7E, FF, 0C, 21, 00, 05, 01, 02, 03, 04, 06, 07, 08, EF | Odtwarzanie 5 plików w kolejności 05, 01, 02, 04<br>Odtwarzanie 5 plików w kolejności 05, 01, 02, 04, 06, 07, 08 |
| Stop | 7E, FF, 06, 16, 00, 00, 00, EF | Zatrzymaj odtwarzanie |
| Play | 7E, FF, 06, 0D, 00, 00, 00, EF | Wznów odtwarzanie |
| Pause | 7E, FF, 06, 0E, 00, 00, 00, EF | Wstrzymaj odtwarzanie |
| Shuffle Play | 7E, FF, 06, 18, 00, 00, 00, EF | Odtwarzanie w przypadkowej kolejności |
| Play with volume | 7E, FF, 06, 22, 00, 1E, 01, EF | Ustaw głośność 1E (30) i odtwarzaj plik 01 |
| Set volume | 7E, FF, 06, 06, 00, 00, 1E, EF | Ustaw głośność 1E (30) |
| Volume Up | 7E, FF, 06, 04, 00, 00, 00, EF | Zwiększ głośność o 1 |
| Volume down | 7E, FF, 06, 05, 00, 00, 00, EF | Zmniejsz głośność o 1 |
| **Komendy dodatkowe** | | |
| Sleep mode | 7E, FF, 06, 0A, 00, 00, 00, EF | Wprowadź stan uśpienia |
| Wake Up | 7E, FF, 06, 0B, 00, 00, 00, EF | Wybudź układ |
| Reset | 7E, FF, 06, 0C, 00, 00, 00, EF | Zerowanie układu |
| Set DAC | 7E, FF, 06, 1A, 00, 00, 00, EF<br>7E, FF, 06, 1A, 00, 00, 01, EF | Włączenie wyjścia DAC<br>Wyłączenie wyjścia DAC |

**Tabela 3. Ramki spontaniczne i komendy statusowe**

| Komenda | Odpowiedź | opis |
|---|---|---|
| spontanicznie | 7E, FF, 06, 41, 00, 00, 00, FE, BA, EF | Dane odebrano prawidłowo |
| spontanicznie | 7E, FF, 06, 3A, 00, 00, 02, FE, BF, EF<br>7E, FF, 06, 3B, 00, 00, 02, FE, BE, EF | Włożono kartę pamięci<br>Wyjęto kartę pamięci |
| spontanicznie | 7E, FF, 06, 3D, 00, 00, 04, FE, BA, EF | Zakończono odtwarzać plik 04 |
| | | |
| 7E, FF, 06, 42, 00, 00, 00, EF | 7E, FF, 06, 42, 00, 00, 00, FE, B6, EF | Stan – odtwarzanie zatrzymane (stop) |
| 7E, FF, 06, 42, 00, 00, 00, EF | 7E, FF, 06, 42, 00, 00, 01, FE, B6, EF | Stan – odtwarza (play) |
| 7E, FF, 06, 42, 00, 00, 00, EF | 7E, FF, 06, 42, 00, 00, 02, FE, B6, EF | Stan – pauza |
| 7E, FF, 06, 48, 00, 00, 00, EF | 7E, FF, 06, 48, 00, 00, 07, FE, AC, EF | Znaleziono 7 plików |
| 7E, FF, 06, 4C, 00, 00, 00, EF | 7E, FF, 06, 4C, 00, 00, 04, FE, AB, EF | Odtwarzany plik nr 4 |
| 7E, FF, 06, 43, 00, 00, 00, EF | 7E, FF, 06, 43, 00, 00, 1E, FE, 99, EF | Poziom siły głosu =1F |

się zaczynać od liczby trzycyfrowej na przykład 001-test1.mp3, 002-test2.mp3. Przykładową strukturę katalogów i nazw plików pokazano na **rysunku 1**.

Odtwarzacz identyfikuje pliki na dwa sposoby. Komenda odtwarzania może podać numer katalogu i numer pliku w katalogu. Można też identyfikować pliki za pomocą indeksu. Indeks jest nadawany w momencie wgrywania na kartę. Inaczej mówiąc – o kolejności odtwarzania decyduje kolejność nagrania plików na kartę.

Sterowanie odtwarzaczem odbywa się poprzez wysyłanie do niego komend. Komendy mają format pokazany w **tabeli 1**. Zestaw najważniejszych komend umieszczono w **tabeli 2**. Jeżeli bajt feedback (rysunek 2) ma wartość 01, to po każdym wysłaniu komendy moduł odpowiada sekwencją 7E, FF, 06, 41, 00, 00, FE, BA EF. Wyzerowany szósty bajt oznacza, że dane zostały odebrane prawidłowo. Oprócz potwierdzenia przesłania komendy mogą być wysyłane przez moduł ramki spontaniczne, niebędące reakcją na komendę, ale na zdarzenie, np. zakończenie odtwarzania konkretnego pliku, lub włożenie/wyjęcie karty SD. Ponadto, można wysyłać komendy pytające o status urządzenia. Te komendy umieszczono w **tabeli 3**.

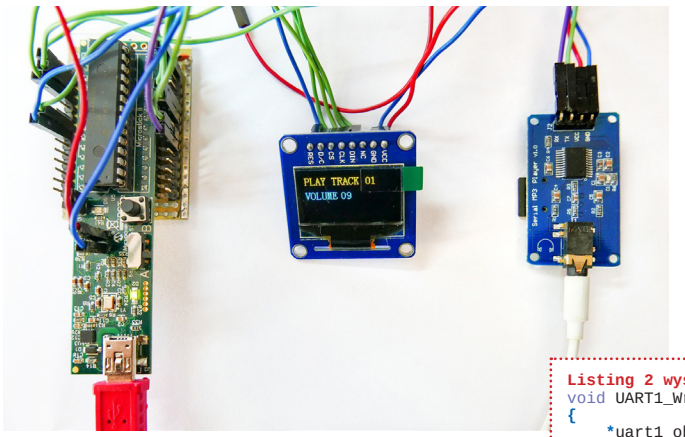
## Testy praktyczne

Do testowania odtwarzacza użyłem modułu ewaluacyjnego Microchip Microstick II z 16-bitowym mikrokontrolerem PIC24FJ64GB002. Moduł ma wbudowany programator/debugger i jest wspierany przez wtyczkę MCC środowiska MPLAB X IDE. Oprócz modułu mp3 dołączonego do mikrokontrolera przez interfejs UART, użyłem niewielkiego wyświetlacza OLED z interfejsem SPI oraz impulsatora ze stykiem zwieranym przyciśnięciem osi (**fotografia 2**).

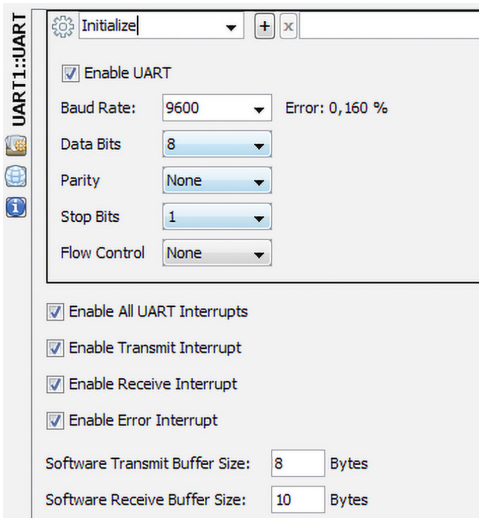
Za pomocą MCC (MPLAB Code Configurator) szybko skonfigurowałem interfejs UART. Obsługa jest oparta na systemie przerwań i ma programowy bufor nadajnika i odbiornika (**rysunek 3**). MCC na podstawie częstotliwości taktowania mikrokontrolera (8 MHz) i zadanej prędkości transmisji (9600 bps) konfiguruje rejestry liczników odpowiedzialnych za prędkość transmisji interfejsu UART. Przy okazji jest wyliczany błąd. Ten błąd jest nie do uniknięcia, jeśli częstotliwość taktowania nie jest specjalnie dobrana do oczekiwanych prędkości transmisji. Procedurę inicjalizacji UART wygenerowaną przez MCC pokazano na **listingu 1**.

Do sterowania modułem będą nam potrzebne procedury wysyłania (**listing 2**) i odbierania (**listing 3**) bajtu przez UART. Wyposażeni w nie możemy przystąpić do napisania procedur wysyłających komendy do modułu odtwarzacza. Podstawowa funkcja wysłania komendy WriteCmdYX ma trzy argumenty: kod komendy, parametr 1 i parametr 2 (**listing 4**). Jeżeli po wysłaniu komendy moduł ma odpowiedzieć potwierdzeniem, to po bajcie komendy cmd trzeba wysłać bajt 0x01. Za pomocą funkcji WriteCmdYX można już sterować wszystkimi funkcjami odtwarzacza. Aby było wygodniej, można napisać szereg funkcji realizujących poszczególne komendy – **listing 5**.

Testowe odtwarzanie odbywa się w pętli nieskończonej. Na początku pętli jest sprawdzany warunek przyciśnięcia osi i zwarcia styku. Jeżeli oś jest wciśnięta, to program najpierw czeka na jej zwolnienie, a potem analizuje stan dwóch zmiennych: cont i pauza. Zależnie od ich stanu przyciśnięcie powoduje wysłanie komendy Pauza (PauzaP()), lub kontynuuj (ContP()). Obrócenie oski enkodera




**Fotografia 2. Układ testowy**




**Rysunek 3. Konfiguracja modułu UART**

**Listing 1. Inicjalizacja modułu UART**

```
void UART1_Initialize(void)
{
    // RTSMD enabled; BRGH enabled; STSEL 1; UARTEN enabled;
PDSEL 8N; LPBACK disabled; WAKE disabled; USIDL disabled; RXINV
disabled; ABAUD disabled; IREN disabled; UEN TX_RX;
    U1MODE = 0x8808;
    // UTXEN disabled; UTXINV disabled; URXISEL RX_ONE_CHAR;
ADDEN disabled; UTXISEL0 TX_ONE_CHAR; UTXBRK COMPLETED; OERR NO_
ERROR_cleared;
    U1STA = 0x0000;
    // U1TXREG 0x0000;
    U1TXREG = 0x0000;
    // U1RXREG 0x0000;
    U1RXREG = 0x0000;
    // Baud Rate = 9600; BRG 103;
    U1BRG = 0x0067;
    IEC0bits.U1RXIE = 1;
    U1STAbits.UTXEN = 1;
    uart1_obj.txHead = uart1_txByteQ;
    uart1_obj.txTail = uart1_txByteQ;
    uart1_obj.rxHead = uart1_rxByteQ;
    uart1_obj.rxTail = uart1_rxByteQ;
    uart1_obj.rxStatus.s.empty = true;
    uart1_obj.txStatus.s.empty = true;
    uart1_obj.txStatus.s.full = false;
    uart1_obj.rxStatus.s.full = false;
}
```

**Listing 2 wysyłanie bajtu przez interfejs UART**

```
void UART1_Write(const uint8_t byte)
{
    *uart1_obj.txTail = byte;

    uart1_obj.txTail++;
    if (uart1_obj.txTail == (uart1_txByteQ + UART1_CONFIG_TX_BYTEQ_LENGTH))
    {
        uart1_obj.txTail = uart1_txByteQ;
    }
    uart1_obj.txStatus.s.empty = false;
    if (uart1_obj.txHead == uart1_obj.txTail)
    {
        uart1_obj.txStatus.s.full = true;
    }
    if (IEC0bits.U1TXIE == false)
    {
        IEC0bits.U1TXIE = true;
    }
}
```

**Listing 3. Odbieranie bajtu przez UART**

```
uint8_t UART1_Read(void)
{
    uint8_t data = 0;
    data = *uart1_obj.rxHead;
    uart1_obj.rxHead++;
    if (uart1_obj.rxHead == (uart1_rxByteQ + UART1_CONFIG_RX_BYTEQ_LENGTH)) {
        uart1_obj.rxHead = uart1_rxByteQ;
    }
    if (uart1_obj.rxHead == uart1_obj.rxTail) {
        uart1_obj.rxStatus.s.empty = true;
    }
    uart1_obj.rxStatus.s.full = false;
    return data;
}
```

powoduje zmianę poziomu sygnału audio ustawianą komendą VolSet z argumentem zmieniającym się od 0 do 32.

Po sprawdzeniu stanu procedury obsługi enkodera program sprawdza czy moduł nie wysłał do mikrokontrolera przez UART danych. W programie wyłączono wysyłanie potwierdzeń i możemy się spodziewać tylko ramki z danymi sygnalizującymi koniec odtwarzania utworu. Jeżeli dane zostały odebrane i jest to ramka sygnalizująca koniec odtwarzania, to jest wysyłana komenda odtwarzająca kolejny utwór z katalogu – **listing 6**.

Moduł testowany w ten sposób działa bez problemu. Wszystkie funkcje są wykonywane prawidłowo. Po włączeniu potwierdzenia każde wysłanie komendy skutkuje przesłaniem ramki z danymi. W aplikacjach wymagających dużej niezawodności wskazane byłoby włączenie potwierdzenia i przesyłanie bajtów CRC. Dla mniej wymagających zastosowań nie jest to konieczne.

**Listing 4. Procedura wysyłania komendy do YX5300**

```
//*********************************************************
//wysłanie komendy
//*********************************************************
void WriteCmdYX(uint8_t cmd,uint8_t par1, uint8_t par2)
{
    UART1_Write(0x7e);
    UART1_Write(0xff);
    UART1_Write(0x06);
    UART1_Write(cmd);
    UART1_Write(0);//0x00 bez odpowiedzi, 0x01 z potwierdzeniem
    UART1_Write(par1);
    UART1_Write(par2);
    UART1_Write(0xef);
}
```

**Listing 5. funkcje komend odtwarzacza**

```
//zerowanie
void ResYX(void)
{
    WriteCmdYX(0x0c,0,0);
    Delay_ms(500);
}
//wybierz karte
void SelSdYX(void)
{
    WriteCmdYX(0x09,0,2);
    Delay_ms(200);
}
//odtwarzaj plik o numerze bezwzglednym nrp
void PlayBp(uint8_t nrp)
{
    WriteCmdYX(0x03,0,nrp);
}
/odtwarzaj plik o numerze nrp w folderze nrf
void PlayP(uint8_t nrf,uint8_t nrp)
{
    WriteCmdYX(0x0f,nrf,nrp);
}
//Nastepny utwor
void NextP(void)
{
    WriteCmdYX(0x01,0,0);
}
//Poprzedni utwor
void PrevP(void)
{
    WriteCmdYX(0x02,0,0);
}
//Pauza
void PauzaP(void)
{
    WriteCmdYX(0x0e,0,0);
}
//Wznow
void ContP(void)
{
    WriteCmdYX(0x0d,0,0);
}
//Zatrzymaj odtwarzanie
void StopP(void)
{
    WriteCmdYX(0x16,0,0);
}
//VOL++
void VolUp(void)
{
    WriteCmdYX(0x04,0,0);
}

//VOL--
void VolDwn(void)
{
    WriteCmdYX(0x05,0,0);
}
//VOL setup
void VolSet(uint8_t vol)
{
    WriteCmdYX(0x06,0,vol);
}
//DAC ON
void DACOn(void)
{
    WriteCmdYX(0x1a,0,0);
}
void SetupYX(void)
{
     ResYX();
}
```

## Moduł z portem USB

Zachęcony pozytywnymi doświadczeniami z modułem YX5300 zakupiłem również w bardzo przystępnej cenie kolejny moduł audio potrafiący odtwarzać pliki z karty SD, ale również z pamięci z interfejsem USB, czyli z tak zwanego pendrivea. Tego rodzaju pamięć jest dużo wygodniejsza od karty SD, ponieważ nie wymaga dodatkowego czytnika, a port USB jest w prawie każdym komputerze.

Po dostarczeniu modułu okazało się, że jest problem z dokumentacją. Sprzedawca przesłał mi link do dokumentacji po chińsku… Pomimo wielu poszukiwań nie udało mi się dotrzeć do dokumentacji napisanej po angielsku. Z konieczności musiałem posłużyć się translatorem Google. Tłumaczenie z chińskiego na polski, mówiąc oględnie, nie zachwyciło. Tłumaczenie z chińskiego na angielski było lepsze, ale tylko trochę. Pozostało posiłkować się częściowymi informacjami i uzupełniać je metodą eksperymentów z oprogramowaniem sterującym. Taka metoda okazała się bardzo pracochłonna, ale pozwoliła na zweryfikowanie działania przynajmniej najważniejszych komend pozwalających na uruchomienie odtwarzania.

Jak już wspomniałem, moduł może odtwarzać pliki dźwiękowe z dwóch nośników: karty SD i pamięci pendrive (USB). Podstawowe właściwości modułu umieszczono w **tabeli 4**, a parametry w **tabeli 5**. Zasilanie i sterowanie odbywa się przez złącze goldpinów – **fotografia 4**. Wyprowadzenia mają następujące funkcje:

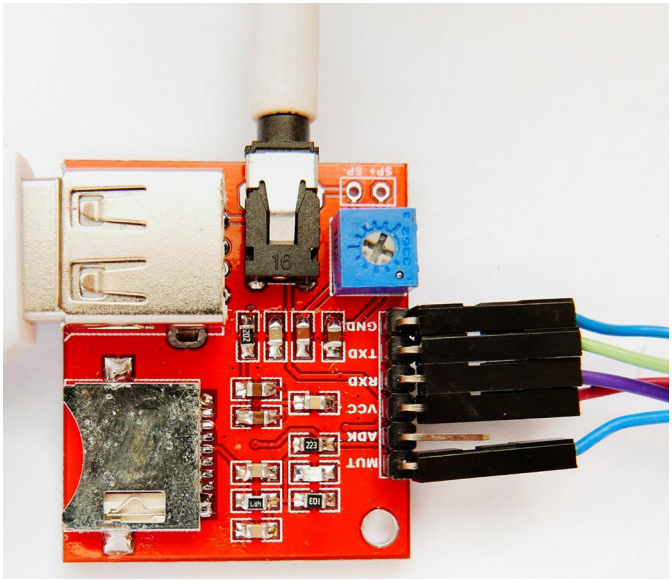
- MUTE – sprzętowe wyciszenie – aktywne, gdy na wejściu jest poziom wysoki.
- ADKEY – sprzętowe sterowanie funkcjami odtwarzania.

**Listing 6.** Pętla testowania modułu YX5300

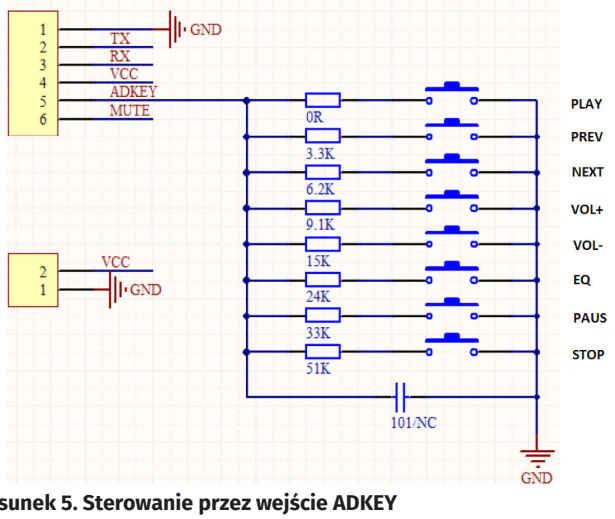
```
while(1)
{
    kod=GetEncoder();//odczytaj stan enkodera
    if(kod==KOD_IMP_ST)
    {
        while(ST==0);  //przyciśnięto oś enkodera
        if(pauza)
        {
            PauzaP();//komenda Pauza
            DispTxt(0,0,"PAUSE     ", 16,1);
            RefreshRAM();
            pauza=0; cont=1;
            continue;
        }
        if(cont)
        {
            ContP();//komenda kontynuuj (wznów)
            DispTxt(0,0,"PLAY TRACK ", 16,1);
            RefreshRAM();
            pauza=1; cont=0;
            continue;
        }
    }
    if(kod==KOD_IMP_UP)
    {
        ++vol;//głośniej
        if(vol==31) vol=30;
        VolSet(vol);
        DispHex(vol,52,20,16);
        RefreshRAM();
    }
    if(kod==KOD_IMP_DWN)
    {
        --vol;//ciszej
        if(vol==0xff) vol=0;
        VolSet(vol);
        DispHex(vol,52,20,16);
        RefreshRAM();
    }
    status=UART1_TransferStatusGet();//czy odebrano z modułu
jakieś dane
    if((status&1)==1)
    {
        UART1_ReadBuffer(buf,10);
        if(buf[0]==0x7e&&buf[3]==0x3d)
        {
            //koniec odtwarzania utworu
            //DispTxt(0,0,"STOP", 16,1);
            PlayP(nk,++np);//odtwarzaj następny utwór
            DispHex(np,88,0,16);
            RefreshRAM();
        }
    }
}
```

| Tabela 4. Podstawowe właściwości modułu | |
|---|---|
| Parametr | |
| Częstotliwość próbkowania | 8/11,025/12/16/22,05/24/32/44,1/48 kHz |
| Rozdzielczość przetwornika | 24 bity |
| Zakres dynamiczny | 90 dB, 85 dB SNR |
| Format | FAT16, FAT32 |
| Foldery/pliki | 255 folderów po 1000 plików |
| Interfejs sterujący | UART lub sterowanie przyciskami |
| Regulacja poziomu sygnału | Cyfrowa (UART) lub za pomocą potencjometru na płytce modułu |
| Charakterystyka częstotliwościowa | 5 ustawień equalizera FLAT, POP, ROCK, JAZZ, CLASIC, BASS |

| Tabela 5. Podstawowe parametry | |
|---|---|
| Format plików dźwiękowych | mp3 11172-3 i ISO13813-3 Layer3 |
| Interfejs USB | 2.0 |
| Interfejs sterujący | UART 9600, 1, N poziom TTL |
| Napięcie zasilania | 3,3 V....5,4 V |
| Pobór prądu | 15 mA (bez pendrive) |
| Temperatura pracy | –40...+70˚C |
| Wilgotność | 5...95% |



**Fotografia 4. Płytka modułu ze złączem sterującym**




**Rysunek 5. Sterowanie przez wejście ADKEY**

- VCC – napięcie zasilające (3,3…5,4 V).
- RxD – linia danych odbieranych przez moduł.
- TxD – linia danych wysłanych przez moduł.
- GND – masa.

Poza złączem sterującym można podłączać słuchawki stereofoniczne do standardowego złącza Jack 3,5 mm lub głośnik do dwóch wyprowadzeń: „SP–” i „SP+”.

Przed opisem interfejsu sterującego warto wspomnieć, że moduł ma wbudowaną możliwość sterowania podstawowymi funkcjami poprzez dołączenie 8 styków zwiernych do specjalnego wejścia ADKEY tak jak to zostało pokazane na rysunku 5. Zwieranie styków połączonych szeregowo z różnymi rezystancjami powoduje wymuszenie różnych napięć na ADKEY. Te napięcia są mierzone przez wewnętrzy przetwornik analogowo cyfrowy i na podstawie tych pomiarów wykonywane są poszczególne komendy. Ponieważ nie testowałem tego trybu pracy, a nie mogłem skopiować chińskich znaków z rysunku do translatora, to przypisanie funkcji do poszczególnych styków należy traktować orientacyjnie. Prawidłowość przypisania funkcji sterujących do styków można łatwo zweryfikować po zbudowaniu układu z **rysunku 5**. Sterowanie poprzez wejście ADKEY można włączyć lub wyłączyć komendą przesyłaną interfejsem UART. Domyślnie ta funkcja jest włączona, co pozwala na pracę w tym trybie bez konieczności użycia sterownika mikroprocesorowego.

## Interfejs sterujący UART

Komendy sterujące modułem można przesyłać wykorzystując standardowy interfejs sterujący UART. Parametry transmisji to: prędkość 9600Bd, jeden bit stopu, brak bitu parzystości i brak sterowania przepływem danych. Podobnie jak w wypadku YX5300, do sterowania wystarczą dwie linie: RxD i TxD.

Format ramki sterującej pokazano na **rysunku 6**. Od razu rzucają się w oczy podobieństwa z układem YX5300, co sugeruje tego samego producenta układu scalonego.

| Bajt startu 7Eh | Liczba bajtów | Komenda | Parametr1 | Parametr 2 | Bajt stopu EFh |
|---|---|---|---|---|---|

- Liczba bajtów = komenda+parametr1+parametr2+bajt stopu.
- Pola Parametr 1 i/lub Parametr2 nie musza występować.

**Rysunek 6. Format ramki sterującej**

Ramka rozpoczyna się od bajtu startu 7Eh. Po nim następuje bajt określający ilość przesyłanych bajtów od bajtu komendy do bajtu stopu lub liczba przesyłanych bajtów tj. wszystkie bajty w ramce bez bajtu startu i stopu. Pola parametru mogą nie występować w ogóle lub może ich być różna liczba, zależnie od rodzaju komendy. Każda ramka kończy się bajtem stopu o wartości EFh.

## Komendy odtwarzacza

Zestaw podstawowych umieszczono w **tabeli 6**. W tym przypadku ramka sterująca nie zawiera pola parametrów. Na przykład żeby wykonać komendę PLAY trzeba wysłać sekwencję 7E 02 01 EF, a komendę PAUSE sekwencję 7E 02 02 EF. Po wysłaniu komendy moduł odpowiada wysłaniem sekwencji znaków ASCII:

- „OK” dla prawidłowo rozpoznanej i zdekodowanej komendy.
- „ERR” dla nierozpoznanej komendy.

Kolejny zestaw komend umożliwia zapytanie modułu o ustawione parametry, na przykład: o status odtwarzania, ustawioną głośność, ilość plików itp. Po wysłaniu komendy moduł odpowiada maksymalnie 16-bitową liczbą zapisaną za pomocą 4 znaków ASCII. Na przykład, aby odczytać ustawiony poziom głośności trzeba wysłać 7E 02 11 EF. Moduł może odpowiedzieć znakami ASCII: 30h, 30h, 30h,39h (0009h), co oznacza liczbę 9, jeżeli komenda jest prawidłowo rozpoznana (nie jest zwracane „OK”). Kiedy komenda nie zostanie prawidłowo rozpoznana, jest zwracany kod błędu „Err”.

**Tabela 6. Podstawowe komendy odtwarzania**

| Kod komendy | Komenda |
|---|---|
| 01h | PLAY |
| 02h | PAUZA |
| 03h | Następny utwór (plik) |
| 04h | Poprzedni utwór (plik) |
| 05h | Głośniej (VOL+) |
| 06h | Ciszej (VOL–) |
| 07h | Tryb obniżonego poboru energii (STANBY –ON) |
| 09h | Praca normalna (STANBY-OFF) |
| 0ah | Szybkie przewijanie do przodu (FORWARD) |
| 0bh | Szybkie przewijanie do tyłu (REWIND) |
| 0eh | STOP |

W **tabeli 7** umieszczono zestaw komend z 8-bitowym argumentem (hex) przeznaczonych do konfigurowania pracy modułu. Ustawienie poziomu głośności może wyglądać następująco: 7E 03 31 1E EF.

Ostatni zestaw komend ma argument 2-bajtowy (16-bitowy) i jest przeznaczony do wybierania pliku do odtwarzania. Komenda 41h wybiera numer pliku (utworu) do odtwarzania. Na przykład, aby odtworzyć plik numer 8 trzeba wysłać komendę 7E 04 41 00 08 EF. Komendy wyboru pliku działają podobnie, jak w module z YX5300. Można identyfikować pliki za pomocą indeksu lub podając numer folderu i numer pliku w folderze. W razie komendy „Wybierz plik z folderu" (42h) folder musi mieć nazwę od 01 do 99, a pliki nazwę 001do 255.mp3 (lub 1do 255wav). Jeżeli foldery i pliki nie będą tak nazywane, to funkcja wyboru plików 42h nie będzie działała poprawnie. Na **rysunku 7** pokazano strukturę folderów i plików zapisanych na pendrive w trakcie testowania modułu odtwarzacza.

Żeby odtworzyć plik nr 002 z katalogu 01 trzeba wysłać komendę 7E 04 42 01 02 EF.

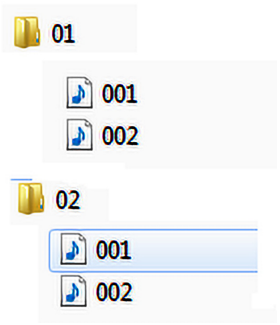
## Testy praktyczne

Napięcia zasilające i interfejs sterujący jest taki sam, jak w module YX5300, więc zastosowałem taki sam układ testowy z modułem Microstick II, wyświetlaczem OLED i impulsatorem. Dokładnie w taki sam sposób został skonfigurowany układ UART oraz obsługa wyświetlacza i impulsatora. Funkcje komend

**Listing 7. Komenda PLAY**

```
void PlayMp3(void)
{
    BufCmd[0]=0x7e; //bajt startu
    BufCmd[1]=0x02; //bajt liczby danych
    BufCmd[2]=0x01; //bajt komendy PLAY
    BufCmd[3]=0xef; //bajt stopu
    DispTxt(0,0,"PLAY TRACK ", 16,1);
    RefreshRAM();
    WriteCmdMp3(BufCmd,4);
}
```

zapisują bufor BufCmd kolejnymi bajtami przesyłanymi przez UART do modułu. Na **listingu 7** pokazano funkcję PlayMp3(), wysyłającą komendę PLAY o kodzie 01h. Najpierw do bufora BufCmd[] są zapisywane bajty: startu, liczba danych, kodu komendy i stopu. Potem na ekranie wyświetlacza jest wyświetlana informacja o uruchomieniu odtwarzania „PLAY TRACK" i jest wywoływana funkcja WriteCmdMp3() z argumentami: wskaźnikiem




**Rysunek 7. Wymagane nazwy plików i katalogów**

**Listing 8. Funkcja wysłania bajtów komendy**

```
uint8_t WriteCmdMp3(uint8_t *buf,uint8_t len)
{
    uint8_t i,status;
    uint8_t bufor[10];
    for(i=0;i<len;i++)  //wysłanie zadanej liczby bajtów
    UART1_Write(buf[i]);
    Delay_ms(50);  //odczekanie na odpowiedź
    status=UART1_TransferStatusGet();
    if((status&2)==2)  //czy odebrano jakieś znaki
    {
        UART1_ReadBuffer(bufor,8);//zapisanie bufora odebranymi znakami
        if(bufor[0]=='O'&&bufor[1]=='K')//czy komenda prawidłowa
        {
            DispTxt(0,20,"ok!", 16,1);
            RefreshRAM();
            return(0);
        }
        if(bufor[0]=='E')//czy komenda nie rozpoznana
        {
            DispTxt(0,20,"err", 16,1);
            RefreshRAM();
            return(1);
        }
        else
        {
            DispTxt(0,20,"NACK", 16,1);//odebrano inne znaki (nie potwierdzenie)
            RefreshRAM();
            return(2);
        }
    }
    DispTxt(0,20,"NOE", 16,1);//nie odebrano żadnych znaków
    RefreshRAM();
    return(3);
}
```

**Tabela 7. Komendy zapytania o ustawienia modułu**

| Kod komendy | Komenda | Odpowiedź (4znaki ASCII) |
|---|---|---|
| 10h | status odtwarzania | 0-STOP, 1-PLAY, 2-PAUZA, 3 –FFORW, 4-FREW |
| 11h | poziom głośności | 0-30 ustawiony poziom głośności |
| 12h | ustawienia equalizera | 0-5 (FLAT\POP\ROCK\JAZZ\CLASSIC\BASS) |
| 13h | tryb odtwarzania | 0-4 Wszystkie\folder\jeden\RANDOM |
| 14h | numer wersji | 1,0 |
| 15h | ilość plików na karcie SD | 1-65536 |
| 16h | ilość plików na pendrive | 1-65536 |
| 18h | Aktualnie używana pamięć | 0-USB 1-SD |
| 19h | Aktualnie odtwarzany plik z karty SD | 1-65536 |
| 1Ah | Aktualnie odtwarzany plik z pendrive | 1-65536 |
| 1Ch | Całkowity czas odtwarzanego utworu w sekundach | Czas w sekundach |
| 1Dh | Bieżący czas odtwarzanego utworu w sekundach | Czas w sekundach |
| 1Eh | Nazwa odtwarzanego utworu | |

**Listing 9. Komedy PAUZA i STOP**
```
uint8_t PauseMp3(void)
{
    BufCmd[0]=0x7e;
    BufCmd[1]=0x02;
    BufCmd[2]=0x02;
    BufCmd[3]=0xef;
    DispTxt(0,0,"PAUSE     ", 16,1);
    RefreshRAM();
    return(WriteCmdMp3(BufCmd,4));
}
uint8_t StopMp3(void)
{
    BufCmd[0]=0x7e;
    BufCmd[1]=0x02;
    BufCmd[2]=0x0e;
    BufCmd[3]=0xef;
    return(WriteCmdMp3(BufCmd,4));
}
```

**Listing 10. Ustawianie głośności**
```
uint8_t VolMp3(uint8_t vol)
{
    BufCmd[0]=0x7e;
    BufCmd[1]=0x03;//liczba bajtów
    BufCmd[2]=0x31;//kod komendy
    BufCmd[3]=vol;  //ustawiana wartość
    BufCmd[4]=0xef;//bajt stopu
    return(WriteCmdMp3(BufCmd,5));
}
```

**Listing 11. Funkcja wyboru pliku do odtwarzania**
```
uint8_t PlayTrackMP3(uint8_t fld,uint8_t plk )
{
    BufCmd[0]=0x7e;
    BufCmd[1]=0x04;
    BufCmd[2]=0x42;//funkcja wybierz folder i plik
    BufCmd[3]=fld;//numer folderu
    BufCmd[4]=plk;//numer pliku
    BufCmd[5]=0xef;//bajt stopu
    DispTxt(0,0,"Play ", 16,1);
    DispHex(fld,40,0,16);//wyświetl numer folderu
    DispHex(plk,60,0,16);//wyświetl numer pliku
    RefreshRAM();
    return(WriteCmdMp3(BufCmd,6));
}
```

na bufor z bajtami do wysłania i ilością danych do wysłania (**listing 8**). Po wysłaniu bajtów komendy program odczekuje 50 ms na odebranie potwierdzenia. To o wiele dłużej, niż potrzeba. Każdy bajt to 10 bitów przesyłanych z prędkością 9600 b/s, więc przesłanie jednego bajtu trwa 1/9600×10=1,04 ms. Trzy bajty potwierdzenia to ok. 3 ms, 4 bajty odpowiedzi po komendach o status modułu to ok. 4 ms, jednak podczas obserwacji transmisji na oscyloskopie widać było, że moduł odpowiada dopiero po pewnym czasie.

Program sprawdza czy bajty zostały odebrane testując status transmisji funkcją *UART1_TransferStatusGet();*. W innych implementacjach użytkownik musi sobie zapewnić możliwość odbierania batów i sygnalizację odebrania. Po odebraniu odpowiedzi następuje analiza odebranych znaków. Wykrycie znaków „OK" oznacza prawidłowo odebrana komendę, wykrycie znaku „E" oznacza niezidentyfikowane polecenie. Wykrycie innych znaków jest sygnalizowane brakiem potwierdzenia NACK. Komendy PAUZA i STOP pokazano na **listingu 9**.

Z zestawu komend konfigurujących najczęściej używana będzie komenda ustawiania głośności – pokazano ją na **listingu 10**. W tym wypadku, po kodzie komendy jest wysyłany jeden bajt argumentu mogący mieć wartość z zakresu 0…30. Do przedstawionych wyżej komend trzeba dołączyć komendę wyboru pliku na podstawie numeru katalogu i numeru pliku (**listing 11**) i można zbudować prosty, ale funkcjonalny odtwarzacz plików mp3.

## Podsumowanie

Pokazane tu moduły odtwarzania plików dźwiękowych są głównie przeznaczone do zastosowań przemysłowych. Zbudowanie „konsumenckiego" odtwarzacza plików muzycznych mp3 jest również możliwe, ale kłopotliwe. Przeszkodą mogą być narzucone nazwy plików, ale głównie brak możliwości odczytania rzeczywistej nazwy pliku będącej jednocześnie nazwą utworu. Żeby skutecznie wybierać utwory do odtwarzania, trzeba za każdym razem zmieniać ich nazwę, aby zaczynała się od trzycyfrowej liczby.

Ta niedogodność jest jednocześnie zaletą, gdy chcemy wybierać komunikaty głosowe nie na podstawie dowolnej nazwy składającej się ze znaków ASCII, ale na podstawie wartości cyfrowej. Takie rozwiązanie bardzo ułatwia odtwarzanie komunikatów zależnie od jakiejś wartości- na przykład wartość napięcia, mocy, temperatury, itp.

Podczas testów okazało się, że oba moduły radzą sobie z podanymi częstotliwościami próbkowania, ale gorzej jest z przepływnością. Na początku przez zupełny przypadek umieściłem na nośniku pliki z przepływnością 320 kb/s. Moduł YX5300 jakoś sobie poradził. Moduł z interfejsem USB odtwarzał materiał dźwiękowy ze sporym poziomem szumów i chwilowymi zniekształceniami. Widać było, że wbudowany mikrokontroler nie bardzo radził sobie z takim plikiem. Materiał nagrany z przepływnością od 64 kB/s do 128 kb/s był odtwarzany z dobrą jakością. Zważywszy na przeznaczenie modułów trudno to zakwalifikować jako wadę, bo przepływność 128 kb/s daje już bardzo dobrą jakość odtwarzanego dźwięku.

Na pewno ze względu na bardzo korzystny stosunek cena/możliwości warto się zainteresować zastosowaniem tego rozwiązania we własnych konstrukcjach, szczególnie tych amatorskich. W zastosowaniach profesjonalnych problemem może być stabilność dostaw.

**Tomasz Jabłoński, EP**

**Tabela 8. Komendy konfiguracji modułu**

| Kod komendy | Komenda | Argument (8bit hex) |
|---|---|---|
| 31h | Ustaw głośność | 0-30 |
| 32h | Ustaw equalizer | 0-5 (FLAT\POP\ROCK\JAZZ\CLASSIC\BASS) |
| 33h | Ustaw tryb odtwarzania | 0-4 Wszystkie\folder\jeden\RANDOM |
| 34h | Przełącz folder | Kolejny podfolder |
| 35h | Przełącz nośnik | 0-USB, 1 karta SD |
| 37h | Wejście ADK | 1-aktywne, 0-nie aktywne |
| 38h | Wejście MUTE | 1-wyciszenie stan wysoki, 0-wyciszenie stan niski |

**Tabela 9. Komendy wyboru pliku do odtwarzania**

| Kod komendy | Komenda | Argument (16bit hex) |
|---|---|---|
| 41h | Wybierz utwór | Numer utworu |
| 42h | Wybierz plik z folderu | 8 starszych bitów określa numer folderu, 8 młodszych numer pliku (utworu) |
| 43h | Wybierz steram track | |
| 44h | Wybierz folder dla stream track | |