Podstawowe parametry:
- zakres regulacji tonów niskich: ±14 dB; wysokich: ±14 dB,
- zakres regulacji głośności: –47...0 dB; balansu: ±79 dB,
- zakres regulacji wzmocnienia multipleksera wejściowego: 0...28 dB,
- separacja kanałów stereo: 90 dB,
- odstęp sygnału od szumu S/N: 106 dB,
- zniekształcenia harmoniczne THD: 0,1%,
- impedancja wejściowa: 100 kΩ; wyjściowa: 30 kΩ,
- minimalna impedancja obciążenia: 2 kΩ,
- maksymalne napięcie wejściowego sygnału audio: 2 VRMS,
- napięcia zasilania: 7...10 V; prąd obciążenia: 120 mA.

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

AVT5975	Regulator barwy dźwięku, głośności i balansu (EP 3/2023)
AVT5873	Stereofoniczny aktywny regulator głośności (EP 8/2021)
AVT5851	7-pasmowy korektor graficzny (EP 4/2021)
AVT5816	Regulator balansu tonów (EP 10/2020)
AVT5637	Wielokanałowy regulator głośności VCA (EP 8/2018)
AVT5629	Cyfrowy regulator głośności z układem PT2257 (EP 6/2018)
AVT3222	Sterowany dowolnym pilotem potencjometr audio z przekaźnikiem (EdW 5/2018)
AVT1979	Korektor barwy dźwięku (EP 11/2017)
AVT1971	Stereofoniczny regulator barwy tonu zasilany z baterii (EP 9/2017)
AVT1972	Potencjometr „Panorama" audio (EP 9/2017)

W ofercie AVT*
AVT6005

* **Uwaga!** Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania! Podstawową wersją zestawu jest wersja **[B]** nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji **[B]** zawiera elementy elektroniczne (w tym **[UK]** – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu. Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- **wersja [C]** – zmontowany, uruchomiony i przetestowany zestaw **[B]** (elementy wlutowane w płytkę PCB),
- **wersja [A]** – płytka drukowana bez elementów i dokumentacji.

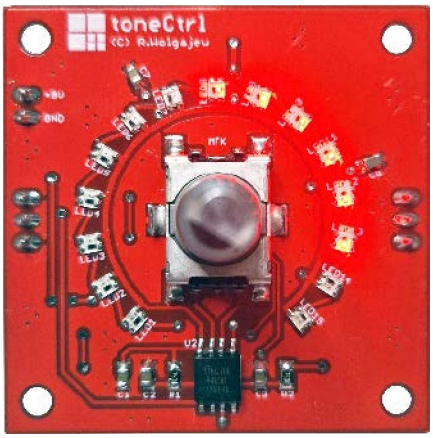
Kity, w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- **wersja [A+]** – płytka drukowana **[A]** + zaprogramowany układ **[UK]** i dokumentacja,
- **wersja [UK]** – zaprogramowany układ.

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik PDF! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! **http://sklep.avt.pl**

W przypadku braku dostępności na stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: **kity@avt.pl**.

toneCtrl (1)
– regulator barwy dźwięku

Prezentujemy nowoczesny regulator barwy dźwięku wyposażony w dodatkowe funkcjonalności, który może znaleźć zastosowanie zarówno przy implementacji nowoczesnych rozwiązań we współczesnych urządzeniach audio, jak i umożliwić modernizację i poprawę parametrów elektrycznych systemów spod znaku vintage.



Mimo że świat współczesny zdominowany jest przez wszechogarniającą technikę cyfrową, gdzie praca z sygnałem sprowadza się do stosowania zaawansowanych algorytmów z dziedziny DSP, nadal rozwijanych jest wiele nowych urządzeń audio, dla których kluczowa jest implementacja dobrej jakości rozwiązań sprzętowych. Co więcej, istnieje także spora grupa użytkowników, którzy z powodzeniem modernizują lub też utrzymują w doskonałym stanie technicznym urządzenia audio z minionej epoki, podtrzymując wydatnie bieżącą modę na systemy spod znaku vintage. Nie powinno to specjalnie dziwić, gdyż dzisiejsza wirtuozeria w domenie cyfrowej znajduje także swoje odzwierciedlenie w minionej epoce spod znaku analogu, tyle że tym razem w materii sprzętowej.

W gruncie rzeczy czasami zastanawiam się, co było większym osiągnięciem inżynierii i ludzkiej inteligencji, wszak zaawansowane rozwiązania układowe i programistyczne charakterystyczne dla obu tych epok niejednokrotnie zaskakują pomysłowością i poziomem zaawansowania technologicznego. Sam, projektując różnorakie systemy elektroniczne, nie tak rzadko spotykałem się z koniecznością rozwiązywania nietypowych problemów dla obu wspomnianych domen, cyfrowej i analogowej. Niemniej jednak zdecydowana większość moich projektów związana jest nieodłącznie ze światem mikrokontrolerów, przez co za każdym razem, gdy sięgam do rozwiązań z pogranicza „cyfry i analogu", odczuwam podwójną satysfakcję. Jako że z rozrzewnieniem wspominam początki swojej przygody z elektroniką, kiedy to, zapewne jak każdy w tym czasie, konstruowałem proste układy analogowe, tym razem postanowiłem wykonać ukłon w kierunku tego rodzaju systemów, no może z pogranicza obu domen.

Postanowiłem zbudować nowoczesny regulator barwy dźwięku wyposażony w dodatkową funkcjonalność, który w swoich założeniach ma znaleźć zastosowanie zarówno przy implementacji nowoczesnych rozwiązań we współczesnych urządzeniach audio, jak i umożliwić modernizację i poprawę parametrów elektrycznych systemów spod znaku vintage. Jako że moim celem nie była implementacja systemu złożonego wyłącznie z elementów dyskretnych, proces projektowania rozpocząłem od poszukiwania nowoczesnego układu, przy udziale którego zrealizuję założone cele.

Mając już pewne doświadczenie w realizacji tego rodzaju urządzeń, jak i odpowiednie rozeznanie w kwestii stosowanych peryferiów, dość szybko dokonałem stosownego wyboru, którym stał się zaawansowany, scalony regulator barwy dźwięku pod postacią układu TDA7440 produkowanego przez firmę STMicroelectronics. Prawdę mówiąc, element ten od dawna zagrzewał miejsce w mojej szufladzie pełnej elektronicznych

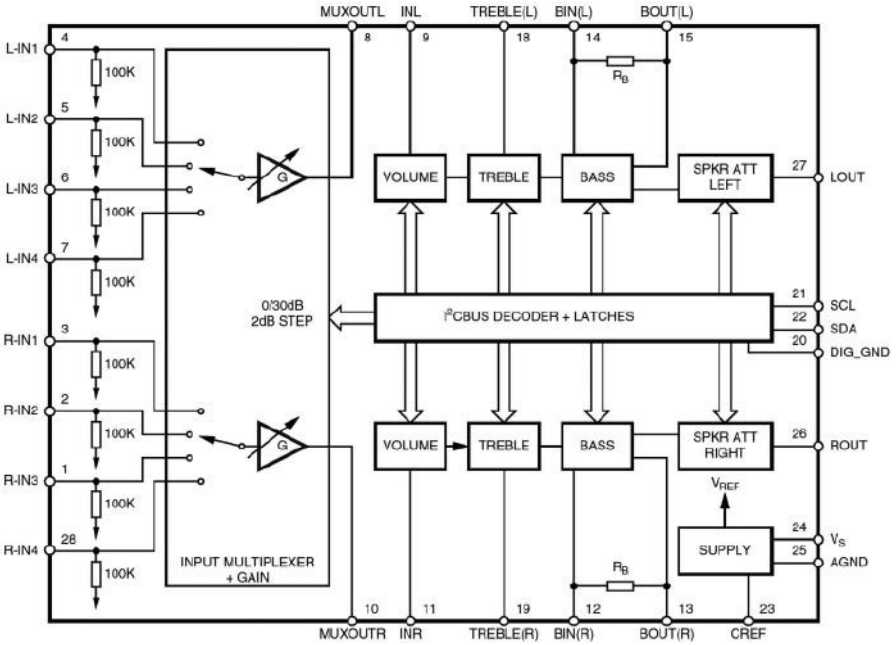
Wykaz elementów, kupuj na stronie sklep.avt.pl (Warszawa, ul. Leszczynowa 11, tel. +48222578451, e-mail: handlowy@avt.pl)

Rezystory: (SMD0805)
R1, R2: 10 kΩ
R3, R4: 4,7 kΩ
R5, R6: 5,6 kΩ

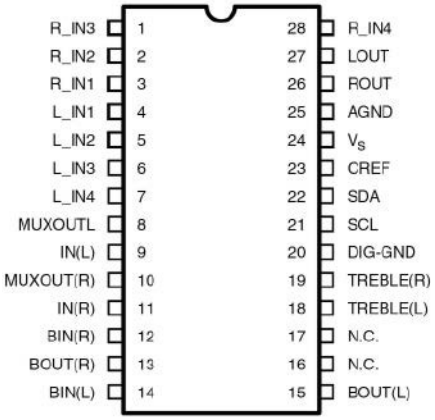
Kondensatory:
C1...C5, C7, C8, C13, C14, C17, C18, C22: 100 nF ceramiczny X7R (SMD0805)
C6: 22 µF/16 V tantalowy (B/3528-21 W)
C9, C10, C20, C21: 1 µF ceramiczny X7R (SMD0805)
C11, C15: 2,2 µF ceramiczny X7R (SMD0805)
C12, C16: 5,6 nF ceramiczny X7R (SMD0805)
C19: 10 µF ceramiczny X7R (SMD0805)

Półprzewodniki:
U1: TDA7440 (SO28)
U2: ATtiny85 (SOIC8)
U3: 78M05 (DPAK)
LED1...LED15: dioda adresowalna LED RGB typu OSTW-2020C1E (SMD2020)

Pozostałe:
IN, OUT: gniazdo męskie proste 3 piny typu NSL25-3 W
PWR: gniazdo męskie proste 2 piny typu NSL25-2 W
MFK: enkoder SMD z przyciskiem typu ALPS EC11J1524413



Rysunek 1. Uproszczony schemat funkcjonalny układu TDA7440 (za dokumentacją STMicroelectronics)



Rysunek 2. Wygląd obudowy układu TDA7440 wraz z rozmieszczeniem wszystkich wyprowadzeń (za dokumentacją STMicroelectronics)

Tabela 1. Rozkład i opis wyprowadzeń układu TDA7440

Nr	Nazwa	Opis
1	R_IN3	Wejście multipleksera wejściowego nr 3, kanał prawy
2	R_IN2	Wejście multipleksera wejściowego nr 2, kanał prawy
3	R_IN1	Wejście multipleksera wejściowego nr 1, kanał prawy
4	L_IN1	Wejście multipleksera wejściowego nr 1, kanał lewy
5	L_IN2	Wejście multipleksera wejściowego nr 2, kanał lewy
6	L_IN3	Wejście multipleksera wejściowego nr 3, kanał lewy
7	L_IN4	Wejście multipleksera wejściowego nr 4, kanał lewy
8	MUXOUT(L)	Wyjście multipleksera wejściowego, kanał lewy
9	IN(L)	Wejście regulatora głośności, kanał lewy
10	MUXOUT(R)	Wyjście multipleksera wejściowego, kanał prawy
11	IN(R)	Wejście regulatora głośności, kanał prawy
12	BIN(R)	Wejście bloku filtra tonów niskich (do podłączenia elementów zewnętrznych), kanał prawy
13	BOUT(R)	Wyjście bloku filtra tonów niskich (do podłączenia elementów zewnętrznych), kanał prawy
14	BIN(L)	Wejście bloku filtra tonów niskich (do podłączenia elementów zewnętrznych), kanał lewy
15	BOUT(L)	Wyjście bloku filtra tonów niskich (do podłączenia elementów zewnętrznych), kanał lewy
16	N.C.	Niepodłączone
17	N.C.	
18	TREBLE(L)	Wyprowadzenie bloku filtra tonów wysokich (do podłączenia elementów zewnętrznych), kanał lewy
19	TREBLE(R)	Wyprowadzenie bloku filtra tonów wysokich (do podłączenia elementów zewnętrznych), kanał prawy
20	DIG_GND	Masa części cyfrowej układu
21	SCL	Sygnał zegarowy magistrali danych I²C
22	SDA	Sygnał danych magistrali danych I²C
23	CREF	Wejście do podłączenia kondensatora filtrującego bloku zasilającego
24	VS	Napięcie zasilania (9 V)
25	A_GND	Masa części analogowej układu
26	ROUT	Wyjście układu, kanał prawy
27	LOUT	Wyjście układu, kanał lewy
28	RIN_4	Wejście multipleksera wejściowego nr 4, kanał prawy

różności. Układ ten, sterowany dzięki wbudowanemu interfejsowi standardu I²C, umożliwia regulację kilku parametrów audio i dodatkowo zapewnia utrzymanie doskonałych parametrów elektrycznych. Wspomniane peryferium charakteryzuje się następującymi, wybranymi cechami użytkowymi:

- 4-kanałowy, stereofoniczny selektor wejściowy z możliwością regulacji wzmocnienia w zakresie 0 dB…+30 dB (z krokiem 2 dB),
- regulacja tonów niskich w zakresie –14… +14 dB (z krokiem 2 dB),
- regulacja tonów wysokich w zakresie –14…+14 dB (z krokiem 2 dB),
- regulacja głośności w zakresie –47…0 dB (z krokiem 1 dB),
- regulacja balansu każdego kanału w zakresie –79…0 dB (z krokiem 1 dB),
- funkcja wyciszenia (MUTE),
- wysoki, maksymalny poziom sygnału wejściowego 2 VRMS,
- doskonały odstęp sygnału od szumu S/N równy 106 dB,
- niskie zniekształcenia harmoniczne THD rzędu 0,1%,
- niewielka liczba niezbędnych dyskretnych elementów zewnętrznych (w większości konieczne do ustalenia parametrów wbudowanych filtrów).

Budowa i działanie

Jak widać, układ TDA7440 cechuje się doskonałymi właściwościami użytkowymi oraz parametrami elektrycznymi i świetnie wpisuje się w założenia naszego projektu. Aby zrozumieć zasadę działania oraz paletę możliwości regulacyjnych układu TDA7440, najlepiej jest spojrzeć na jego uproszczony schemat funkcjonalny pokazany na **rysunku 1**. Dalej, na **rysunku 2**, pokazano wygląd obudowy układu TDA7440 wraz z rozmieszczeniem wszystkich wyprowadzeń, zaś w **tabeli 1** pokazano z kolei rozkład i opis jego wyprowadzeń.

Co oczywiste, parametry elektryczne filtrów odpowiadających za regulację

tonów niskich i wysokich (w tym topologię filtra, jeśli chodzi o filtr tonów niskich) możemy modyfikować, dobierając odpowiednie wartości (i konfigurację) elementów zewnętrznych, jednak w moim urządzeniu zastosuję parametry domyślne sugerowane przez aplikację producenta układu. Osoby chcące poeksperymentować z tego rodzaju ustawieniami odsyłam do noty aplikacyjnej, gdzie w sposób szczegółowy opisany został sposób doboru (i stosowne wzory) zewnętrznych elementów dyskretnych. Tyle na ten moment, jeśli chodzi o nasz element „regulacyjny", który, co oczywiste, wymaga współistnienia jakiegoś systemu odpowiedzialnego za przeprowadzanie i wyświetlanie wyników regulacji.

Do sterowania, jak zapewne się domyślacie, użyję jakiegoś niewielkiego mikrokontrolera, ale co z wyświetlaniem wprowadzonych nastaw? Najprościej byłoby zastosować jakiś niewielki wyświetlacz, ale że nasze urządzenie ma znaleźć zastosowanie również przy modyfikacjach istniejących systemów audio, implementacja jakiegokolwiek wyświetlacza nie zawsze będzie możliwa, a już na pewno nie zawsze pożądana, jeśli chcemy zachować wygląd w duchu retro.

Postanowiłem ostatecznie, że jako element interfejsu użytkownika prezentujący wprowadzone nastawy użyję szeregu diod LED zgrupowanych w postaci wycinka okręgu wokół elementu regulacyjnego (enkodera) co powinno wyglądać bardzo efektownie a jednocześnie nie nazbyt nowocześnie. Ale jak na linijce diod LED pokazać szereg wartości regulacyjnych? Najprostszym sposobem jest zastosowanie kilku odrębnych kolorów, każdy dla innego parametru, by już po samym kolorze użytkownik urządzenia wiedział, jaki parametr podlega regulacji. Prawda, że proste? Co oczywiste, można również zastosować kilka pokręteł, każde dla innego parametru i każde wyposażone w szereg diod LED, ale przecież chcemy skonstruować urządzenie kompaktowe, które łatwo zaadaptujemy do istniejących rozwiązań, nieprawdaż? W takim razie najprościej zastosować diody LED RGB, których kolor możemy dowolnie modyfikować przy użyciu mikrokontrolera.

W tym miejscu stanąłem przed wyzwaniem wyboru odpowiednich elementów wykonawczych, a więc sterownika diod LED, jak i samych diod. Jak wiemy, aby płynnie sterować kolorem diody LED typu RGB, należy zastosować 3-kanałowy sterownik PWM. Wynika z tego, że skoro przewiduję zastosowanie 15 elementów tego typu, to liczba niezbędnych kanałów wzrasta nam do 45. Trudno wyobrazić sobie mikrokontroler, który sprostałby tym wymaganiom, a jeszcze trudniej wyobrazić sobie projekt płytki drukowanej o niewielkiej wielkości, na której upakowalibyśmy tyle odrębnych ścieżek. Co oczywiste, można byłoby zastosować jakiegoś rodzaju sterowanie matrycowe, by ograniczyć liczbę koniecznych połączeń, ale biorąc pod uwagę liczbę wymaganych kanałów PWM i oczekiwaną rozdzielczość takiego sygnału (8 bitów), trudno mi sobie wyobrazić efektywne sterowanie bez użycia dość dużych prądów, by uzyskać wynikową jasność na akceptowalnym poziomie. Zresztą nawet w takim przypadku nie rozwiązujemy problemu ze skomplikowaniem rysunku obwodu drukowanego. Pat? Otóż nie.

Dość szybko zdałem sobie sprawę, iż jedynym sensownym rozwiązaniem tego rodzaju problemu konstrukcyjnego będzie zastosowanie adresowalnych diod LED RGB, których konstrukcja pozwala na uniknięcie wszystkich wspomnianych problemów. Diody takie, oprócz wyprowadzeń zasilających, wyposażone są w jakiś szeregowy interfejs komunikacyjny, przy użyciu którego dokonujemy ustawień koloru jej świecenia. Interfejs, o którym mowa, zaimplementowany jest w taki sposób (zarówno w kwestii sprzętowej, jak i logicznej), że diody takie, połączone w łańcuchy, mogą być indywidualnie adresowane, a co za tym idzie, każda z nich ma niezależne sterowanie. Sam przebieg PWM niezbędny do regulacji koloru jej świecenia generowany jest sprzętowo dzięki sterownikowi zabudowanemu w takie element.

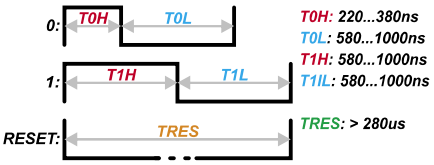
Przejdźmy zatem do konkretów. Pierwszą myślą, jaka przyszła mi w tym czasie do głowy, było zastosowanie bardzo popularnych i tanich elementów tego typu pod postacią diod z rodziny WS2811/WS2812, jednak elementy te produkowane są w dość dużych, jak na potrzeby naszej aplikacji, obudowach o rozmiarze 5×5 mm. Kierując się potrzebą znalezienia elementów możliwie najmniejszych, dość szybko natknąłem się na diody adresowalne RGB typu OSTW2020C1E firmy OptoSupply produkowane w niewielkich 4-wyprowadzeniowych obudowach SMD typu 2020 o wymiarach 2,2×2 mm, które idealnie wpisują się w założenia naszego projektu. Dioda tego rodzaju, podobnie jak popularne diody WS2811/WS2812, wyposażona jest w 4 wyprowadzenia:

- wyprowadzenia zasilające: GND i VCC,
- wejście asynchronicznego interfejsu komunikacyjnego DIN,
- wyjście asynchronicznego interfejsu komunikacyjnego DOUT.

Wygląd obudowy diody typu OSTW2020C1E z zaznaczeniem nazw wyprowadzeń pokazano na **rysunku 3**.

Jak zapewne się domyślacie, podobnie jak ma to miejsce w przypadku diod WS2811, diody typu OSTW2020C1E łączy się

Rysunek 3. Wygląd obudowy diody typu OSTW2020C1E z zaznaczeniem nazw wyprowadzeń

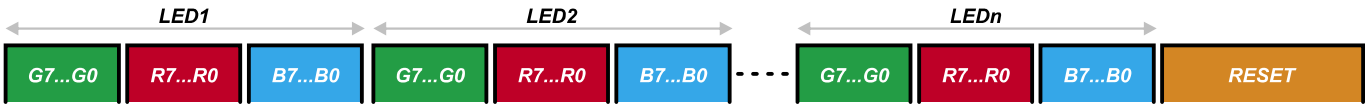


Rysunek 4. Przebiegi sygnałów interfejsu komunikacyjnego diody OSTW2020C1E w trakcie transmisji bitu logicznej „1", logicznego „0" i sygnału RESET

w łańcuchy, łącząc wyjścia interfejsu komunikacyjnego diody bieżącej z wejściami interfejsu komunikacyjnego diody kolejnej i tak dalej. Wejście interfejsu komunikacyjnego diody pierwszej, co oczywiste, łączy się z wyjściem tegoż interfejsu w mikrokontrolerze, zaś sama konstrukcja ramek danych i sposób działania sterownika zabudowanego w strukturze diody zapewnia odpowiednią synchronizację transmisji i niezbędne adresowanie. Zacznijmy więc od podstaw. Jak już wspomniałem, mamy do czynienia z interfejsem asynchronicznym, gdzie nie ma wyprowadzenia sygnału zegarowego, w związku z czym dane przesyłane przy jego użyciu muszą być w pewien sposób zakodowane, by możliwe stało się ich proste zdekodowanie i by były one odporne na zakłócenia i artefakty.

W rozwiązaniu firmy OptoSupply zastosowano mechanizmy dobrze znane z interfejsów bezprzewodowej transmisji danych stosowanych w torach podczerwieni, gdzie stany logiczne „1" i „0" zakodowane zostały długością impulsu. Dodatkowo wprowadzono tak zwany sygnał RESET (również zakodowany długością impulsu), który powoduje zresetowanie interfejsów komunikacyjnych sterowników diod LED i ich oczekiwanie na nowe dane. Na **rysunku 4** pokazano przebiegi sygnałów interfejsu komunikacyjnego diody OSTW2020C1E w trakcie transmisji bitu logicznej „1", logicznego „0" i sygnału RESET. Co ważne, pojedyncze bity danych zgrupowane w bajty danych przesyłane są w kolejności od bitu najstarszego (MSB) do najmłodszego (LSB) a każda dioda LED w łańcuchu oczekuje na 3 bajty danych odpowiedzialnych za składowe jej koloru przesyłane w kolejności G, R, B.

W tym miejscu zapewne zadacie sobie pytanie, skąd każda z diod w łańcuchu „wie", które z przesyłanych danych użytecznych przeznaczone są właśnie dla niej, a nie dla innej? Zrealizowano to w bardzo prosty, acz skuteczny sposób. Każda z diod LED w łańcuchu po włączeniu zasilania (jak również po odebraniu sygnału RESET) oczekuje na 3 bajty danych przeznaczonych wyłącznie dla niej. Do tego czasu jej wyjściowy interfejs komunikacyjny (wyjście DOUT) jest „nieprzezroczysty" dla nadchodzących danych (a dokładnie rzecz biorąc, przenosi bez zmian wyłącznie sygnał RESET). Po odebraniu wspomnianych 3 bajtów danych dioda ta staje się „przezroczysta" dla kolejnych



Rysunek 5. Kompletna ramka transmisji łańcucha diod OSTW2020C1E

Rysunek 6. Schemat ideowy urządzenia toneCtrl

nadchodzących danych, co znaczy ni mniej, ni więcej, że retransmituje je do kolejnych diod w łańcuchu (z małym opóźnieniem rzędu 300 ns). Biorąc pod uwagę, iż dokładnie tak samo zachowuje się każda dioda w łańcuchu, dość szybko zdamy sobie sprawę, że kolejne dane użyteczne przesyłane przez tak skonstruowany interfejs komunikacyjny trafiają kolejno do następujących po sobie (w sensie elektrycznym) diod w łańcuchu. Skuteczne i zarazem genialne w swojej prostocie, nieprawdaż?

Niemniej jednak, i co widać na **rysunku 5**, prezentującym kompletną ramkę transmisji łańcucha diod tego typu, przyjęty sposób transmisji stanowiący podstawę adresowania poszczególnych diod LED w łańcuchu powoduje, że nie da się zaadresować (czyli przesłać do niej danych) na przykład diody czwartej w łańcuchu bez przesłania wcześniejszych (i zdawałoby się niepotrzebnych w tym momencie) danych dla diody trzeciej, drugiej i pierwszej. Mimo tego ograniczenia jest to rozwiązanie bardzo wygodne i chętnie stosowane przez producentów wszelkiego rodzaju peryferiów o takim przeznaczeniu.

Tyle, jeśli chodzi o szczegóły dotyczące naszych podstawowych elementów wykonawczych zaangażowanych w projekt urządzenia toneCtrl.

Pora na omówienie schematu ideowego naszego urządzenia, który to pokazano na **rysunku 6**. Jest to bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) typu ATtiny85 taktowany wewnętrznym, wysokostabilnym generatorem RC o częstotliwości 8 MHz. Jest on odpowiedzialny za programową implementację interfejsu I²C, przy użyciu którego realizuje obsługę układu TDA7440. Ponadto mikrokontroler nasz realizuje obsługę szeregu adresowalnych diod LED RGB stanowiących element graficznego interfejsu użytkownika oraz obsługę switcha MFK używając w celu eliminacji drgań styków wbudowany układ czasowo-licznikowy Timer0 i stosowne przerwanie systemowe (przy okazji obsługiwane jest krótkie i długie naciśnięcie tegoż przycisku). Dodatkowo, dzięki skorzystaniu z przerwania zewnętrznego INT0, możliwa stała się efektywna obsługa kierunku obrotów enkodera, co było niezbędne podczas implementacji mechanizmów obsługi interfejsu użytkownika. Tyle w kwestii schematu.

Robert Wołgajew, EP

Podstawowe parametry:
- zakres regulacji tonów niskich: ±14 dB; wysokich: ±14 dB,
- zakres regulacji głośności: –47...0 dB; balansu: ±79 dB,
- zakres regulacji wzmocnienia multipleksera wejściowego: 0...28 dB,
- separacja kanałów stereo: 90 dB,
- odstęp sygnału od szumu S/N: 106 dB,
- zniekształcenia harmoniczne THD: 0,1%,
- impedancja wejściowa: 100 kΩ; wyjściowa: 30 kΩ,
- minimalna impedancja obciążenia: 2 kΩ,
- maksymalne napięcie wejściowego sygnału audio: 2 VRMS,
- napięcia zasilania: 7...10 V; prąd obciążenia: 120 mA.

W ofercie AVT*
AVT6005

* **Uwaga!** Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania! Podstawową wersją zestawu jest wersja **[B]** nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji **[B]** zawiera elementy elektroniczne (w tym **[UK]** – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu. Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- **wersja [C]** – zmontowany, uruchomiony i przetestowany zestaw **[B]** (elementy wlutowane w płytkę PCB),
- **wersja [A]** – płytka drukowana bez elementów i dokumentacji.

Kity, w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- **wersja [A+]** – płytka drukowana **[A]** + zaprogramowany układ **[UK]** i dokumentacja,
- **wersja [UK]** – zaprogramowany układ.

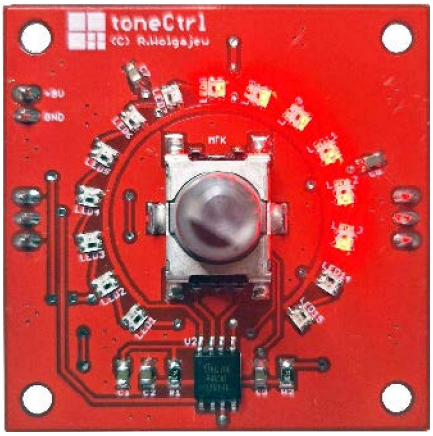
Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

AVT5975	Regulator barwy dźwięku, głośności i balansu (EP 3/2023)
AVT5873	Stereofoniczny aktywny regulator głośności (EP 8/2021)
AVT5851	7-pasmowy korektor graficzny (EP 4/2021)
AVT5816	Regulator balansu tonów (EP 10/2020)
AVT5637	Wielokanałowy regulator głośności VCA (EP 8/2018)
AVT5629	Cyfrowy regulator głośności z układem PT2257 (EP 6/2018)
AVT3222	Sterowany dowolnym pilotem potencjometr audio z przekaźnikiem (EdW 5/2018)
AVT1979	Korektor barwy dźwięku (EP 11/2017)
AVT1971	Stereofoniczny regulator barwy tonu zasilany z baterii (EP 9/2017)
AVT1972	Potencjometr „Panorama" audio (EP 9/2017)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik PDF! Podczas składania zamówienia upewnij się, którą wersję zamawiasz!
http://sklep.avt.pl

W przypadku braku dostępności na stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: **kity@avt.pl**.

toneCtrl (2)
– regulator barwy dźwięku



Prezentujemy drugą część opisu nowoczesnego regulatora barwy dźwięku. Dzięki kompaktowej budowie i intuicyjnej obsłudze może znaleźć zastosowanie zarówno przy implementacji nowoczesnych rozwiązań we współczesnych urządzeniach audio, jak i umożliwić modernizację i poprawę parametrów elektrycznych systemów spod znaku vintage.

Zanim przejdę do zagadnień montażowych nie sposób nie opisać podstawowych zagadnień implementacyjnych w zakresie oprogramowania układowego. Zacznijmy od modułu obsługi układu TDA7440.

Oprogramowanie sterujące

Zanim przejdę do schematu samego urządzenia, nie sposób nie opisać podstawowych zagadnień implementacyjnych w zakresie oprogramowania układowego. Zacznijmy od modułu obsługi układu TDA7440, a dokładnie od pliku nagłówkowego do obsługi tegoż peryferium upraszczającego i porządkującego późniejszy kod, który to plik pokazano na **listingu 1**.

Zanim przejdę do konkretnych funkcji odpowiedzialnych za realizację dostępnej funkcjonalności układu TDA7440, trzeba powiedzieć sobie, w jaki sposób, przy udziale interfejsu I²C, przeprowadzane są stosowne regulacje. Nie ma tu żadnej niespodzianki. Układ TDA7440 wyposażono w szereg rejestrów konfiguracyjnych o odrębnych adresach. Zapis do tych rejestrów stosownych wartości powoduje zmianę parametrów regulacyjnych układu. Wspomniany zapis możemy wykonać w dwóch dostępnych trybach: bez autoinkrementacji i z autoinkrementacją. Tryb pierwszy pozwala na zapis jednego rejestru konfiguracyjnego podczas jednej pełnej ramki transmisji na magistrali I²C, podczas gdy tryb drugi (z autoinkrementacją) pozwala na zapis wartości wielu rejestrów konfiguracyjnych podczas jednej pełnej ramki transmisji na magistrali I²C, co w założeniach ma zmniejszyć ruch na magistrali danych.

Kompletną ramkę transmisji układu TDA7440 dla obu trybów transmisji pokazano na **rysunku 7**. Dalej, na **listingu 2** pokazano funkcję odpowiedzialną za wybór aktywnego wejścia multipleksera wejściowego układu TDA7440, zaś na **listingu 3** pokazano funkcję odpowiedzialną za ustawienie

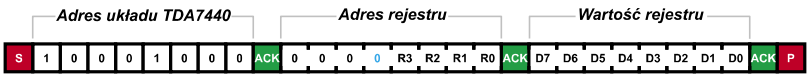
Listing 1. Plik nagłówkowy modułu obsługi układu TDA7440

```
//Adres układu TDA7440
//oraz bit autoinkrementacji
#define TDA7440_ADDR 0x88
#define AUTO_INCREMENT_MODE 0b10000

//Definicje rejestrów konfiguracyjnych
// i wybranych ustawień
#define INPUT_SELECT_REG 0x00
    #define INPUT1 0x03
    #define INPUT2 0x02
    #define INPUT3 0x01
    #define INPUT4 0x00

#define INPUT_GAIN_REG 0x01
#define VOLUME_REG 0x02
#define BASS_REG 0x03
#define UNUSED_REG 0x04
#define TREBLE_REG 0x05
#define SPEAKER_ATT_R_REG 0x06
#define SPEAKER_ATT_L_REG 0x07
```

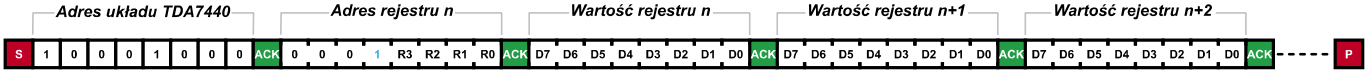
Listing 2. Funkcja odpowiedzialna za wybór aktywnego wejścia multipleksera wejściowego układu TDA7440

```
void TDA7440setInput(uint8_t Input){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(INPUT_SELECT_REG);
    //Używamy predefiniowanych stałych
    i2cWriteByte(Input);
    i2cStop();
}
```

Operacja zapisu pojedynczego rejestru:



Operacja zapisu wielu kolejnych rejestrów:



S *- sygnał Start magistrali I²C*
P *- sygnał Stop magistrali I²C*

Rysunek 7. Kompletna ramka transmisji układu TDA7440 dla obu trybów transmisji

Listing 3. Funkcja odpowiedzialna za ustawienie wzmocnienia multipleksera wejściowego układu TDA7440

```
//Gain: 0...15 odpowiada
//wzmocnieniu 0...+30 dB (krok 2 dB)
void TDA7440setInputGain(uint8_t Gain){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(INPUT_GAIN_REG);
    i2cWriteByte(Gain);
    i2cStop();
}
```

wzmocnienia dla tegoż multipleksera. Kolejną funkcją jest funkcja odpowiedzialna za regulację głośności układu TDA7440, której ciało pokazano na **listingu 4**. Czas na kluczowe funkcje odpowiedzialne za regulację wzmocnienia w zakresie tonów niskich i wysokich, których ciała pokazano odpowiednio na **listingach 5 i 6**. I na koniec zagadnień implementacyjnych związanych z obsługą układu TDA7440 funkcja odpowiedzialna za regulację balansu, której ciało pokazano na **listingu 7**.

Mamy już komplet informacji, dzięki którym jesteśmy w stanie efektywnie obsługiwać ustawienia układu TDA7440, w związku z czym pora na zaprezentowanie niezmiernie prostych funkcji odpowiedzialnych za obsługę adresowalnych diod LED RGB typu OSTW2020C1E. Zacznijmy od pliku nagłówkowego do obsługi tegoż peryferium upraszczającego i porządkującego późniejszy kod, który to plik pokazano na **listingu 8**. Dalej, na **listingu 9** pokazano ciało funkcji odpowiedzialnej za przesłanie jednego bitu danych przy udziale wspomnianego wcześniej asynchronicznego interfejsu komunikacyjnego. Warto zauważyć, że stosowną funkcję zdefiniowano, używając atrybutu `__always_inline__`, który instruuje kompilator, by ZAWSZE wstawiał rozwinięcie funkcji w miejscu wywołania, nie zaś skok do jej ciała. Jest to o tyle istotne, iż operujemy na dość rygorystycznych timingach rzędu setek nanosekund, gdzie, przy częstotliwości taktowania mikrokontrolera rzędu 8 MHz (okres 125 ns), niepotrzebne (z punktu widzenia programisty, nie kompilatora) wykonanie jakiegoś rozkazu asemblera może zmienić całą transmisję danych, uniemożliwiając jakiekolwiek sterowanie wspomnianymi elementami.

Skoro mamy już funkcję umożliwiającą przesłanie jednego bitu informacji, to pora na funkcję (także odpowiednio zadeklarowaną) umożliwiającą przesłanie kompletnego bajta danych, przy udziale wspomnianego wcześniej interfejsu danych, której ciało pokazano na **listingu 10**. Dalej, by

Ustawienia Fuse-bitów:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 1
CKOUT: 1
DWEN: 1
EESAVE: 0
RSTDISBL: 0

Listing 4. Funkcja odpowiedzialna za regulację głośności układu TDA7440

```
//Volume: 0 odpowiada MUTE, zaś 1...48
//odpowiada tłumieniu –47...0 dB (krok 1 dB)
void TDA7440setVolume(uint8_t Volume){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(VOLUME_REG);
    if(Volume == 0) i2cWriteByte(0b00111000);
    else i2cWriteByte(48-Volume);
    i2cStop();
}
```

Listing 5. Funkcja odpowiedzialna za regulację wzmocnienia w zakresie tonów niskich układu TDA7440

```
//Bass: 0...6 odpowiada tłumieniu –14...–2 dB,
//7 odpowiada 0dB, zaś 8...14 odpowiada
//wzmocnieniu +2...+14dB (krok 2dB)
void TDA7440setBass(uint8_t Bass){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(BASS_REG);
    if(Bass < 8) i2cWriteByte(Bass);
    else i2cWriteByte(22-Bass);
    i2cStop();
}
```

Listing 6. Funkcja odpowiedzialna za regulację wzmocnienia w zakresie tonów wysokich układu TDA7440

```
//Treble: 0...6 odpowiada tłumieniu –14...–2 dB,
//7 odpowiada 0dB, zaś 8...14 odpowiada
//wzmocnieniu +2...+14dB (krok 2dB)
void TDA7440setTreble(uint8_t Treble){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(TREBLE_REG);
    if(Treble < 8) i2cWriteByte(Treble); else i2cWriteByte(22-Treble);
    i2cStop();
}
```

Listing 7. Funkcja odpowiedzialna za regulację balansu układu TDA7440

```
//Balance: 0...72 odpowiada tłumieniu kanału
//prawego MUTE...-72...-1dB, 73 to balans na 0dB,
//74...146 odpowiada tłumieniu kanału
//lewego -1...-72dB...MUTE (krok 1dB)
void TDA7440setBalance(uint8_t Balance){
    i2cStart();
    i2cWriteByte(TDA7440_ADDR);
    i2cWriteByte(SPEAKER_ATT_R_REG|AUTO_INCREMENT_MODE);
    if(Balance < 73){
        if(Balance == 0) i2cWriteByte(0b01111000);
        //Tłumienie prawego kanału
        else i2cWriteByte(73-Balance);
        //Lewy na 0dB
        i2cWriteByte(0);
    } else if(Balance > 73){
        //Prawy na 0dB
        i2cWriteByte(0);
        if(Balance == 146) i2cWriteByte(0b01111000);
        //Tłumienie lewego kanału
        else i2cWriteByte(Balance-73);
    } else { //Balance = 73, czyli balans = 0
        i2cWriteByte(0); //Prawy na 0dB
        i2cWriteByte(0); //Lewy na 0dB
    }
    i2cStop();
}
```

Listing 8. Plik nagłówkowy modułu obsługi diod OSTW2020C1E

```
#define LED_PORT PORTB
#define LED_DDR DDRB
#define LED_PIN_NR PB0
#define LED_SET LED_PORT |= (1<<LED_PIN_NR)
#define LED_RESET LED_PORT &= ~(1<<LED_PIN_NR)
#define LED_PIN_AS_OUTPUT LED_DDR |= (1<<LED_PIN_NR)
```

Listing 9. Funkcja odpowiedzialna za przesłanie jednego bitu danych interfejsu diod OSTW2020C1E

```
__inline__ void __attribute__
    ((__always_inline__))ledSendBit(uint8_t Bit){
    if(Bit){
        LED_SET;
        _delay_us(0.65);
        LED_RESET;
        _delay_us(0.65);
    } else {
        LED_SET;
        _delay_us(0.25);
        LED_RESET;
        _delay_us(0.65);
    }
}
```

Listing 10. Funkcja odpowiedzialna za przesłanie jednego bajta danych interfejsu diod OSTW2020C1E

```
__inline__ void __attribute__
    ((__always_inline__)) ledSendByte(uint8_t Byte){
    ledSendBit(Byte & 0x80);
    ledSendBit(Byte & 0x40);
    ledSendBit(Byte & 0x20);
    ledSendBit(Byte & 0x10);
    ledSendBit(Byte & 0x08);
    ledSendBit(Byte & 0x04);
    ledSendBit(Byte & 0x02);
    ledSendBit(Byte & 0x01);
}
```

Listing 11. Funkcja pozwalającą na przesłanie kompletnej informacji o kolorze diody LED typu OSTW2020C1E

```
__inline__ void __attribute__
    ((__always_inline__)) ledSendColor(
    uint8_t R, uint8_t G, uint8_t B) {
    ledSendByte(G);
    ledSendByte(R);
    ledSendByte(B);
}
```

Listing 12. Funkcja odpowiedzialna za transmisję sygnału RESET interfejsu diod OSTW2020C1E

```
inline void ledReset(void){
    LED_RESET;
    _delay_us(300);
}
```

uprościć zapis danych do adresowalnych diod LED RGB, przewidziano funkcję pozwalającą na przesłanie kompletnej informacji o kolorze tejże diody LED, której ciało pokazano na **listingu 11**. Zapewne pamiętacie, że asynchroniczny interfejs komunikacyjny diod LED typu OSTW2020C1E przewiduje transmisję jeszcze jednego sygnału, a mianowicie sygnału RESET. Ciało funkcji odpowiedzialnej za transmisję sygnału RESET interfejsu diod OSTW2020C1E pokazano na **listingu 12**.

Normalnie w tym miejscu zakończyłbym opis obsługi naszych ciekawych diod LED RGB, ale nie tym razem. Prawdę mówiąc, funkcje pokazane powyżej będą działać bez problemów wyłącznie na szybszych mikroprocesorach, dla których jeden cykl zegarowy jest o rząd wielkości krótszy niż timingi odpowiedzialne za wysłanie poszczególnych bitów koloru diod LED. W innym przypadku, czyli tak jak u nas, gdzie takt zegara trwa 125 ns (dla częstotliwości taktowania 8 MHz), wykonanie każdego, dodatkowego rozkazu (jak na przykład sprawdzenie wartości każdego bitu zmiennej przeznaczonej do wysłania) może dość skutecznie zepsuć całą transmisję danych, gdyż niezachowane są wówczas rygorystyczne timingi kodujące bity informacji. Jak sobie z tym poradzić?

Dość prosto, choć nie nazbyt optymalnie. Ja utworzyłem osobne funkcje dla każdego rodzaju bitu (0 i 1) i każdego koloru, który zamierzałem przesyłać, a wszelkie warunki zastąpiłem gotowym do wykonania kodem bez zbędnych (z punktu widzenia transmisji) rozkazów. Spowodowało to powstanie wielu funkcji o bardzo prostym (i w gruncie rzeczy podobnym), acz długim kodzie, który zachował niezbędne timingi. Te uproszczone funkcje, o których mowa powyżej, pokazałem na **listingu 13**, gdzie zamieszczono odrębne funkcje dla każdego z bitów informacji (0 i 1) i przykładową funkcję wysyłającą dane dla koloru czerwonego. Dzięki takiemu rozwiązaniu bez problemu możemy transmitować dane do naszych diod LED, korzystając z mikrokontrolera taktowanego zegarem 8 MHz.

Oczywiście, i czego się zapewne domyślacie, w czasie takiej transmisji należy blokować możliwość obsługi przerwań systemowych. Można również poradzić sobie w prostszy sposób, zwyczajnie zwiększając prędkość zegara taktującego mikrokontroler (najlepiej co najmniej dwukrotnie), lecz ja nie chciałem iść tą drogą, gdyż wydatnie zwiększa się wtedy zapotrzebowanie systemu mikroprocesorowego na energię. Inną sprawą jest to, iż w przypadku naszego mikrokontrolera nie mam już wolnych portów I/O, by podłączyć niezbędny rezonator kwarcowy, w związku z czym należałoby zastosować element o większej liczbie wyprowadzeń, co z kolei nie jest pożądane z uwagi na wymiary obwodu drukowanego. Kolejną możliwością jest zastosowanie jakiegoś sprzętowego, szeregowego medium transmisyjnego wbudowanego w mikrokontroler (USART lub SPI), ale i w takim przypadku nie uciekniemy od konieczności zwiększenia częstotliwości taktowania mikrokontrolera. Uff, to tyle.

Przebrnęliśmy przez wszelkie zagadnienia implementacyjne, w związku z czym pora na omówienie schematu ideowego naszego urządzenia, który pokazano na **rysunku 7**. Jest to bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) typu ATtiny85 taktowany wewnętrznym, wysokostabilnym generatorem RC o częstotliwości 8 MHz. Jest on odpowiedzialny za programową implementację interfejsu I²C, przy użyciu którego realizuje obsługę układu TDA7440. Ponadto mikrokontroler nasz realizuje obsługę szeregu adresowalnych diod LED

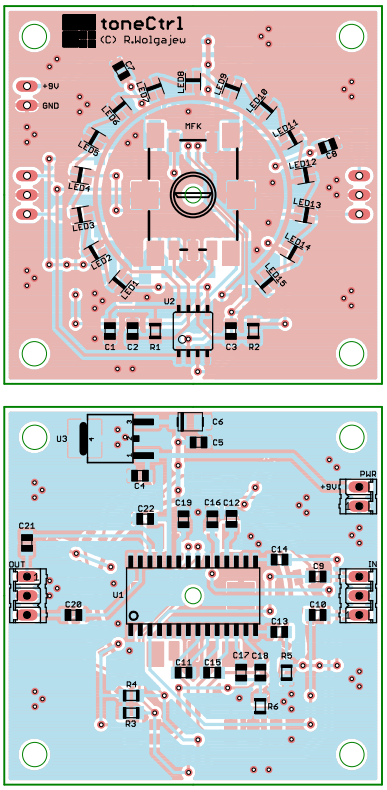
Listing 13. Uproszczone funkcje umożliwiające poprawną transmisję sygnałów sterujących interfejsu diod OSTW2020C1E dla mikrokontrolera taktowanego zegarem 8 MHz

```
#define NOP() __asm__ __volatile__ ("nop")

//Dla zegara 8MHz (NOP = 125ns)
__inline__ void __attribute__
    ((__always_inline__)) ledSendBit0(void){
    LED_SET;
    NOP();
    LED_RESET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}

//Dla zegara 8MHz (NOP = 125ns)
__inline__ void __attribute__
    ((__always_inline__)) ledSendBit1(void){
    LED_SET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    LED_RESET;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}

__inline__ void __attribute__
    ((__always_inline__)) ledRed(void){
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();

    ledSendBit1();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();

    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
    ledSendBit0();
}
```

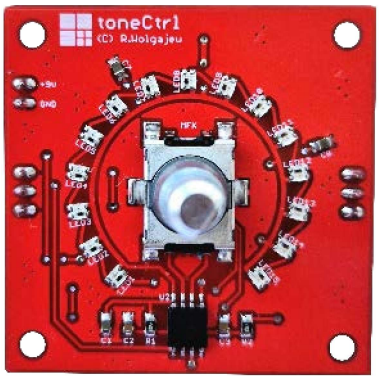
RGB stanowiących element graficznego interfejsu użytkownika oraz obsługę switcha MFK, używając w celu eliminacji drgań styków wbudowany układ czasowo-licznikowy Timer0 i stosowne przerwanie systemowe (przy okazji obsługiwane jest krótkie i długie naciśnięcie tegoż przycisku). Dodatkowo, dzięki skorzystaniu z przerwania zewnętrznego INT0, możliwa stała się efektywna obsługa kierunku obrotów enkodera, co było niezbędne podczas implementacji mechanizmów obsługi interfejsu użytkownika. Tyle w kwestii schematu, w zawiązku z czym przejdźmy do zagadnień montażowych.

Montaż i uruchomienie

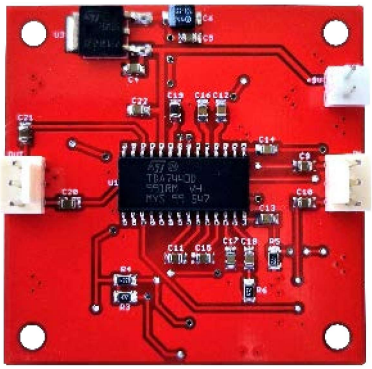
Schemat montażowy naszego urządzenia pokazano na **rysunku 8**. Jak widać, zaprojektowano bardzo kompaktową, dwustronną płytkę drukowaną ze zdecydowaną przewagą elementów SMD montowanych po obu stronach laminatu. Montaż urządzenia rozpoczynamy od warstwy TOP, gdzie w pierwszej kolejności przylutowujemy diody LED RGB. Aby ułatwić sobie nieco to zadanie,



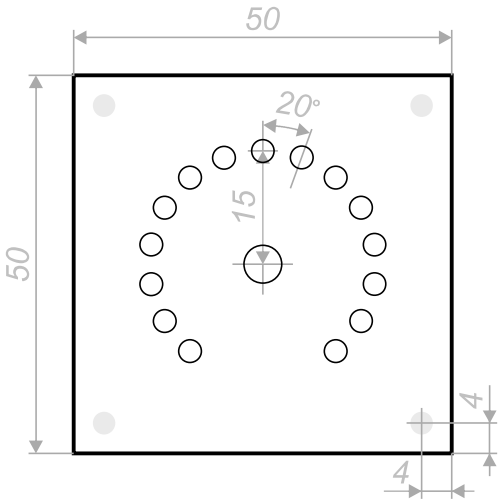
Rysunek 8. Schemat montażowy urządzenia toneCtrl



Fotografia 1. Wygląd zmontowanej płytki urządzenia toneCtrl widzianej od strony TOP (w wersji prototypowej)



Fotografia 2. Wygląd zmontowanej płytki urządzenia toneCtrl widzianej od strony BOTTOM (w wersji prototypowej)



Rysunek 9. Szablon upraszczający wykonanie otworów w płycie czołowej urządzenia toneCtrl

wspomniane elementy możemy wcześniej przykleić do obwodu drukowanego, stosując kropelkę dowolnego kleju nieprzewodzącego prąd, pamiętając jednocześnie, by klejem tym nie pokryć pól lutowniczych umieszczonych na spodzie obudowy każdej z tych diod. Warto zwrócić uwagę na polaryzację wspomnianych elementów, którą określa nadrukowany na spodzie obudowy pasek nieprzewodzącej farby. W następnej kolejności lutujemy mikrokontroler a na samym końcu elementy pasywne.

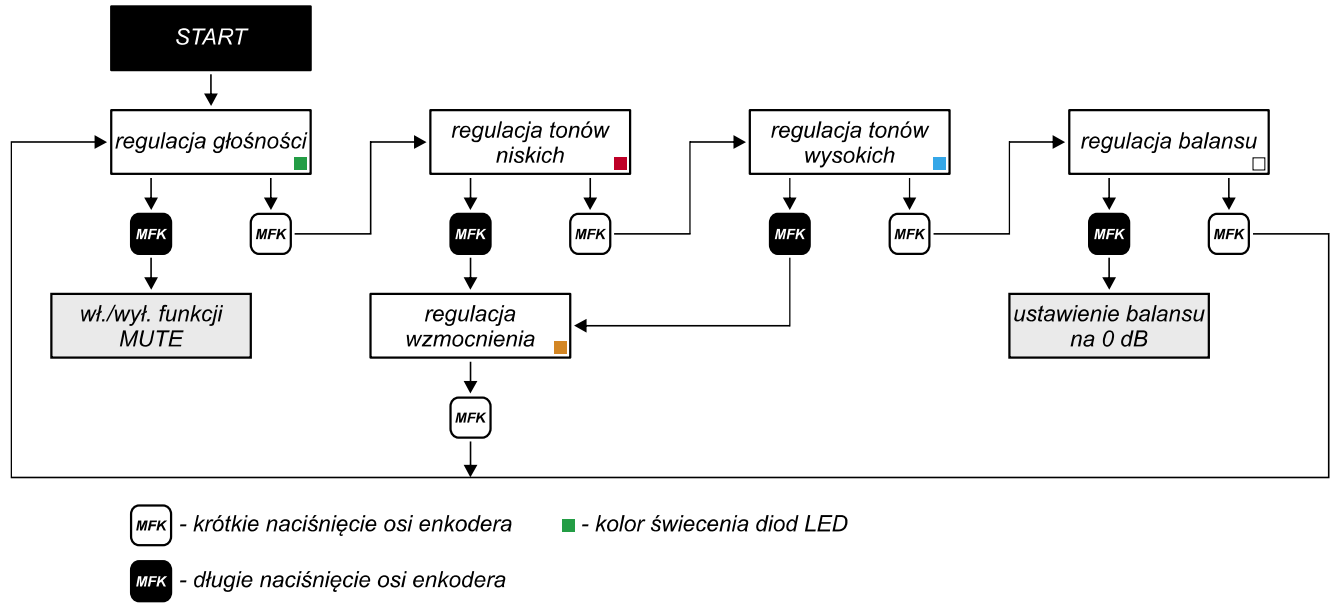
W tym momencie przechodzimy na warstwę BOTTOM, gdzie w pierwszej kolejności lutujemy elementy półprzewodnikowe (układ TDA7440 i stabilizator 78M05), dalej elementy pasywne, a na samym końcu gniazda IN, OUT oraz PWR. W tej chwili ponownie wracamy na warstwę TOP, by przylutować enkoder SMD, przy czym okazać się może, iż niezbędne stanie się usunięcie wystającego, niewielkiego plastikowego trzpienia umieszczonego na spodniej części jego obudowy (w płytce drukowanej przewidziano wyłącznie jeden otwór na środkowy trzpień o większej średnicy, podczas gdy mogą występować dwa).

Poprawnie zmontowany układ nie wymaga jakichkolwiek regulacji i powinien działać zaraz po włączeniu zasilania. Na **fotografii 1** pokazano wygląd zmontowanej płytki drukowanej urządzenia od strony TOP (w wersji prototypowej), zaś na **fotografii 2** odpowiedni widok od strony BOTTOM (także w wersji prototypowej).

Dla osób, które we własnym zakresie będą chciały wykonać panel czołowy do naszego urządzenia lub też zmodyfikować panel urządzenia, do którego zamontowany zostanie nasz regulator, przewidziałem stosowny szablon (w skali 1:1) pokazany na **rysunku 9**. Dla informacji dodam, że diody LED rozmieszczone są na bazie okręgu o promieniu 15 mm z odstępem o kąt 20° jedna od drugiej.

Obsługa

Projektując interfejs użytkownika urządzenia toneCtrl, kierowałem się zasadą maksymalnego uproszczenia sposobu obsługi układu, jak i chęcią wyposażenia go w odpowiednią paletę możliwości. W tym celu przewidziałem wyłącznie jeden element regulacyjny (enkoder z przyciskiem) oraz szereg diod LED (ułożonych po łuku wokół osi enkodera) pokazujących wartości regulowanych parametrów. Przełączenia trybu



Rysunek 10. Diagram prezentujący kompletny algorytm obsługi urządzenia toneCtrl

regulacji (czyli parametru, jaki będzie podlegał regulacji) dokonujemy poprzez krótkie naciśnięcie oski enkodera. Pokręcanie oski enkodera powoduje z kolei regulację wybranego parametru oraz wyświetlenie jego ustawień na wspomnianej wcześniej linijce LED, przy czym kolor diod LED i sposób prezentacji zależy od rodzaju regulowanego parametru (co oczywiste, w inny sposób prezentowane są ustawienia balansu, tonów niskich czy wysokich niż ustawienia wzmocnienia multipleksera wejściowego czy głośności).

Warto podkreślić, iż każdorazowej zmianie rodzaju regulowanego parametru (a więc krótkiemu naciśnięciu oski enkodera) towarzyszy zapisanie wszystkich ustawień urządzenia w nieulotnej pamięci EEPROM mikrokontrolera i ich późniejsze wczytanie podczas kolejnego uruchomienia urządzenia.

Długie naciśnięcie oski enkodera powoduje z kolei realizację dodatkowych funkcjonalności, których rodzaj zależy od aktywnego trybu regulacji. Na **rysunku 10** pokazano diagram prezentujący kompletny algorytm obsługi urządzenia toneCtrl.

Warto zauważyć, iż wejście w tryb ustawiania wzmocnienia multipleksera wejściowego dokonywane jest w nieco inny sposób niż przełączanie pozostałych trybów pracy urządzenia, co zostało wprowadzone celowo z uwagi na fakt, iż jest to ustawienie, którego dokonujemy w zasadzie wyłącznie raz, implementując nasze urządzenie w docelowym systemie audio.

Na koniec słowo uwagi na temat programowania mikrokontrolera. Uważny Czytelnik dostrzeże z pewnością, iż przycisk zintegrowany w ośce enkodera obsługiwany jest przez port PB5 mikrokontrolera będący jednocześnie portem sygnału RESET, który domyślnie nie może pełnić funkcji typowego portu I/O. Aby taką funkcję pełnił, niezbędne jest ustawienie (czyli wyzerowanie) fuse-bitu RSTDISBL. Jednak ustawienie (czyli wyzerowanie) tego bitu uniemożliwi dalsze programowanie mikrokontrolera za pomocą zwykłego szeregowego programatora, a jedyną dostępną wtedy opcją stanie się wysokonapięciowy programator równoległy. Wynika z tego, że w pierwszej kolejności należy wgrać oprogramowanie sterujące (HEX) a dopiero później ustawić docelowe fuse-bity. **Uwaga: brak ustawienia (wyzerowania) bitu RSTDISBL uniemożliwi obsługę przycisku enkodera.**

I już naprawdę na sam koniec dodam, iż włączeniu urządzenia towarzyszy pokazanie efektownej animacji za pomocą wbudowanych diod LED, która to, co oczywiste, nie wpływa na wprowadzane regulacje parametrów elektrycznych urządzenia.

Robert Wołgajew, EP