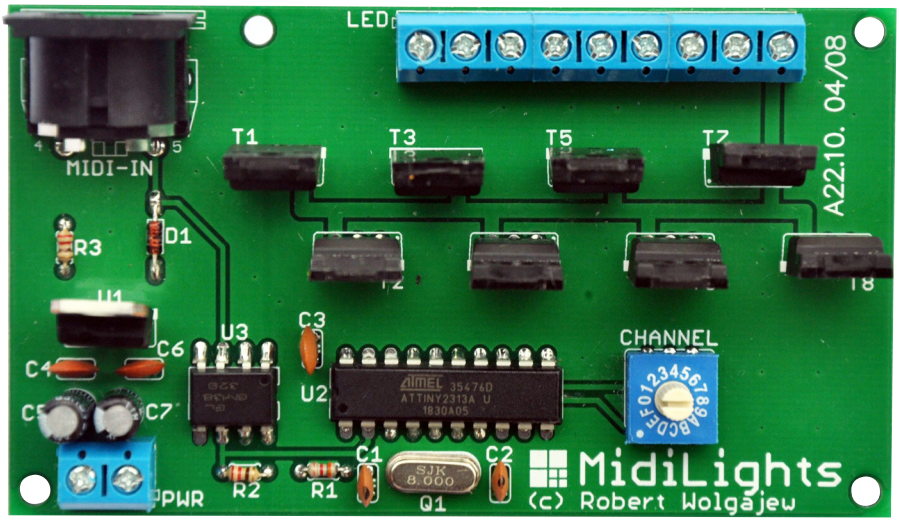
MidiLights

Instrumenty klawiszowe praktycznie zawsze wyposażone są w interfejs komunikacyjny MIDI, dzięki któremu w łatwy sposób możemy zapisać materiał nagraniowy w celu jego późniejszego odtworzenia. Co więcej, interfejs MIDI pozwala na podział poszczególnych instrumentów pomiędzy 16 aktywnych kanałów, przez co każdy z instrumentów może być niezależnie zarejestrowany. Korzystając z tej właściwości, możemy wydzielić jeden, nieużywany kanał MIDI, na którym zapiszemy informację o towarzyszących efektach świetlnych urozmaicających aspekt muzyczny wydarzenia.



Zadaniem prezentowanego urządzenia MidiLights jest odtwarzanie efektów wizualnych towarzyszących nagraniu muzycznemu zapisanych na jednej ze ścieżek MIDI. Zanim jednak przejdę do opisu urządzenia, kilka słów komentarza należy się samemu interfejsowi MIDI. Interfejs ten jest szeregowym interfejsem komunikacyjnym pracującym z szybkością 31250 (±1%) bits/s, w którym dane przesyłane są w paczkach po 8 bitów, z jednym bitem startu i jednym bitem stopu, bez bitów kontroli parzystości. Układ wejściowy interfejsu MIDI wykonuje się zwykle przy użyciu szybkiego transoptora (6N138), który zamienia prąd w linii (ok. 5 mA) na przebiegi napięciowe, zaś interfejs wyjściowy z wykorzystaniem pary rezystorów ograniczających prąd diody LED w urządzeniu odbiorczym.

W standardzie MIDI dane przesyłane są grupowo w formie tzw. komunikatów (Messages), przy czym wprowadzono bardzo prosty sposób na odróżnienie bajtów poleceń sterujących (*Status Byte*) od bajtów danych (*Data Byte*): bajty poleceń mają ustawiony najstarszy bit (0xFF...0x80) a bajty danych najstarszy bit mają wyzerowany (0x7F...0x00). Zwykle informacje przesyłane są w kolejności: bajt polecenia i po nim jeden lub dwa bajty danych (w zależności od rodzaju polecenia). Polecenia wysyłane są tylko przy zmianie danego elementu sterującego. Bajt polecenia określa jedną ze standardowo zdefiniowanych funkcji, którą instrument ma wykonać (4 najstarsze bity), np. *Note On/Off* (włącz/wyłącz nutę), *Control Change* (zmień parametr urządzenia), *Program Change* (zmień rodzaj brzmienia) oraz numer kanału MIDI, na którym informacja ma być odebrana (pozostałe 4 bity określające jeden z 16 kanałów MIDI).

Dla porządku należy wspomnieć o możliwości ograniczenia transferu danych poprzez usunięcie redundancji, z której korzysta metoda *Running Status*. Polega na wysłaniu jednego bajta polecenia i wielu bajtów danych (bez każdorazowego ponawiania bajta polecenia) w przypadku przesyłania tego samego rodzaju sygnałów sterujących jeden za drugim, np. sygnały wywołane zmianą jednego i tego samego regulatora. Przy projektowaniu programu obsługi naszego urządzenia metoda ta nie została zaimplementowana.

Co ciekawe, MIDI, choć wymyślone w latach 80., jest na tyle uniwersalne, że z powodzeniem wykorzystują je najnowsze urządzenia estradowe. W praktyce prawie każde urządzenie estradowe korzysta z dobrodziejstw tego medium transmisyjnego, zaś jednym z powodów użycia technologii MIDI jest szeroka dostępność i uniwersalność takich sterowników.

Budowa

Prezentowany układ pozwala na sterowanie ośmioma kanałami LED za pomocą dowolnego sterownika MIDI (komputer z odpowiednim oprogramowaniem, klawiatura MIDI lub sprzętowy sekwenser MIDI), przy czym w każdym kanale LED możliwa jest niezależna regulacja jasności świecenia w 128 krokach regulacji. Schemat urządzenia

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5734

Podstawowe parametry:
- 16 obsługiwanych kanałów MIDI,
- 8 obsługiwanych kanałów PWM,
- Obsługiwane komunikaty MIDI: Note On, Note Off,
- Rozdzielczość kanałów PWM: 7 bitów
- Wydajność prądowa kanałów PWM: do 10 A (zależne od zastosowanego radiatora),
- Napięcie zasilania: 7,5…12 VDC, pobierany prąd 8 mA.

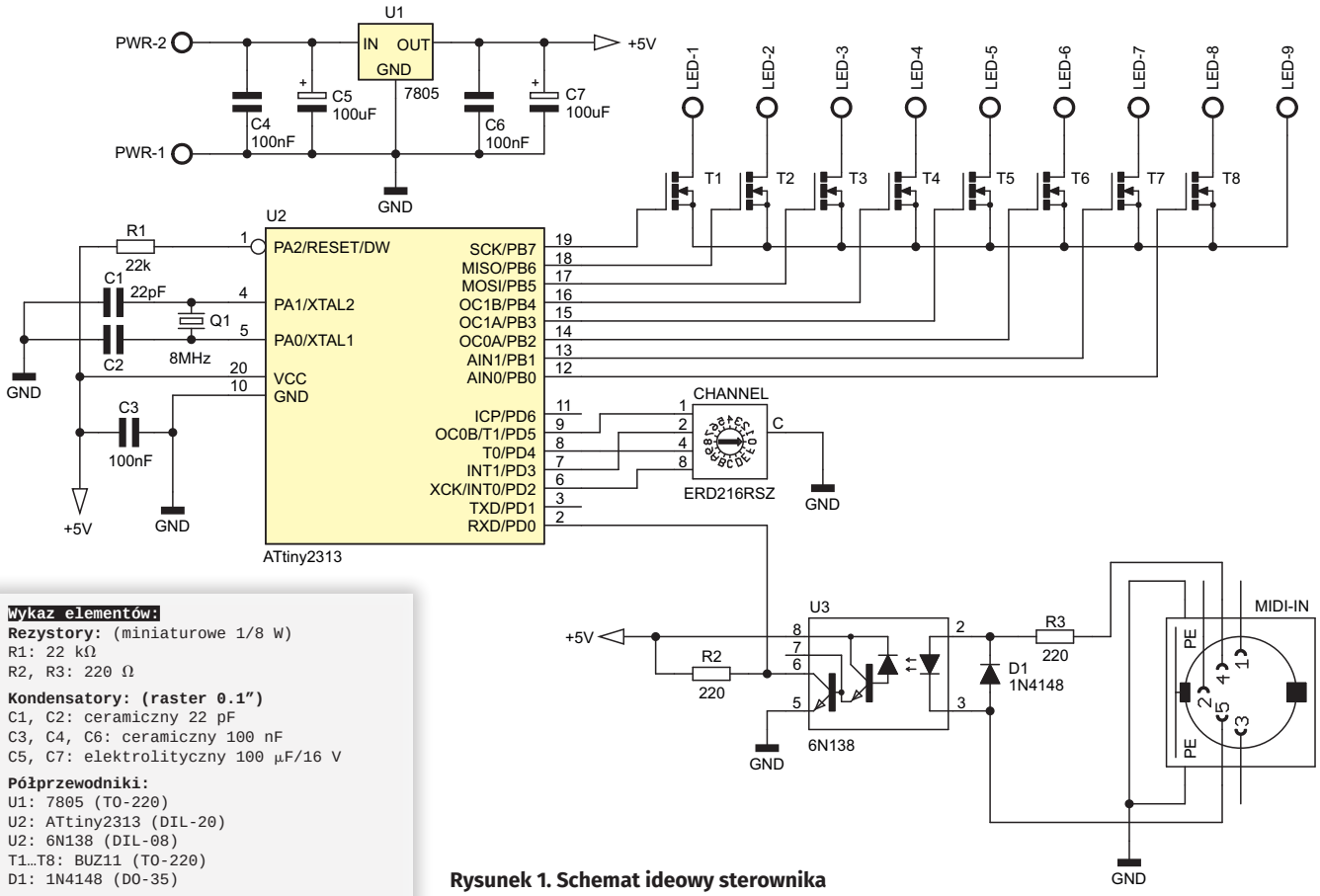
Projekty pokrewne na www.media.avt.pl:

AVT-5724	MIDIswitch – przełącznik MIDI do wzmacniacza lampowego (EP 11/2019)
AVT-5692	MidiCtrl – prosty sterownik MIDI (EP 8/2019
AVT-5594	Konwerter MIDI-SPI (EP 7/2017)
AVT-5107	Odtwarzacz plików MIDI (EP 8/2007)
AVT-960	Płyta ewaluacyjna i klawiatura muzyczna MIDI (EP 12/2006)
AVT-842	Asynchroniczny konwerter RS232-MIDI (EP 12/1999)

* **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 *http://sklep.avt.pl*, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.



Rysunek 1. Schemat ideowy sterownika

Wykaz elementów:

Rezystory: (miniaturowe 1/8 W)
R1: 22 kΩ
R2, R3: 220 Ω

Kondensatory: (raster 0.1")
C1, C2: ceramiczny 22 pF
C3, C4, C6: ceramiczny 100 nF
C5, C7: elektrolityczny 100 µF/16 V

Półprzewodniki:
U1: 7805 (TO-220)
U2: ATtiny2313 (DIL-20)
U2: 6N138 (DIL-08)
T1…T8: BUZ11 (TO-220)
D1: 1N4148 (DO-35)

Inne:
Q1: rezonator 8 MHz (raster 0.2")
PWR: złącze AK500/2 (raster 0.1")
LED: złącze AK500/9 (raster 0.1")
CHANNEL: zadajnik kodu HEX/BCD typu ERD216RSZ
MIDI-IN: gniazdo DIN-5 do montażu THT

MidiLights pokazano na rysunku **rysunku 1**. Jest to bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) ATtiny2313 taktowany zewnętrznym rezonatorem kwarcowym o częstotliwości 8 MHz. Zastosowanie rezonatora kwarcowego jako źródła taktowania mikrokontrolera wynikało z potrzeby zapewnienia dużej dokładności prędkości transmisji MIDI (±1%).

Mikrokontroler odpowiedzialny jest za obsługę wejściowego interfejsu MIDI (driver wejściowy z zastosowaniem transoptora 6N138) przy użyciu odpowiednio skonfigurowanego interfejsu USART mikrokontrolera oraz za programową realizację 8-kanałowego sterownika PWM (z użyciem układu licznikowego TIMER0 wbudowanego w strukturę mikrokontrolera).

Konieczność implementacji 8-kanałowego (7-bitowego) programowego sterownika PWM wynikała z niewystarczającej liczby sprzętowych kanałów PWM, jakie udostępnia zastosowany mikrokontroler. Z jednej strony implementacja takiego mechanizmu jest niezmiernie prosta, z drugiej angażuje sporo zasobów mikrokontrolera, gdyż przy deklarowanej rozdzielczości PWM równej 7 bitów (128 poziomów jasności) oraz obsługiwanych 8 kanałach stosowna funkcja obsługi przerwania licznika TIMER0 wywoływana jest 12800 razy na sekundę (co ok. 78 µs). Na szczęście, co pokażę za chwilę, funkcja ta jest niezmiernie prosta i krótka, więc mimo jej częstego wywoływania mikrokontroler dysponuje jeszcze sporą dawką wolnego czasu

Program aplikacji realizuje jeszcze obsługę sprzętowego interfejsu USART mikrokontrolera. Funkcja obsługi przerwania od odbioru danych USART wypełnia jedynie programowy, kołowy bufor danych MIDI, którego obsługa realizowana jest w pętli głównej aplikacji. W ten sposób zapewniono efektywną obsługę wejściowego interfejsu MIDI, minimalizując właściwie możliwość pominięcia przesyłanych danych. Warto w tym momencie podkreślić, że urządzenie Midi Lights obsługuje wyłącznie dwa rodzaje komunikatów MIDI, a mianowicie: *Note On* (0x90) i *Note Off* (0x80), przy czym odbierane są wyłącznie dane wysyłane na aktywnym dla urządzenia kanale MIDI, którego numer wybierany jest za pomocą zadajnika kodu oznaczonego jako CHANNEL. Ustawienia 0…9 odpowiadają kanałom MIDI 1…10, zaś A…F kanałom 11…16. Obsługiwany komunikat typu *Note On* lub *Note Off* ma następującą konstrukcję:

NoteOn/NoteOff – noteNr – noteVelocity

gdzie:

NoteOn/NoteOff – obsługiwany komunikat MIDI,

noteNr – numer klawisza (nuta, 0…127), przy czym obsługiwane są wyłącznie następujące wartości nut: C4, D4, E4, F4, G4, A4, B4, C5, odpowiadające kolejnym kanałom PWM (0…7),

noteVelocity – siła nacisku klawisza (0…127), odpowiadająca wartości dla wybranego kanału PWM (0…7).

Komunikat *Note On* (z towarzyszącymi danymi *noteNr* i *noteVelocity*) powoduje ustawienie jasności wybranego kanału PWM, zaś komunikat *Note Off* (z towarzyszącymi danymi *noteNr* i *noteVelocity*) powoduje wyłączenie wybranego kanału PWM. W ten prosty sposób, używając np. klawiatury MIDI, możemy sterować jasnością i zachowaniem poszczególnych kanałów PWM, tworząc tło świetlne dla prezentowanego utworu muzycznego. Co więcej, tło takie może zostać zapisane łącznie z danymi muzycznymi (np. na nieużywanym kanale MIDI), co zapewnia nam powtarzalność tak stworzonej prezentacji.

Obsługa MIDI

Na **listingu 1** pokazano funkcję przeznaczoną do inicjalizacji interfejsu USART mikrokontrolera jako sprzęgu MIDI. W celu zwiększenia przejrzystości kodu źródłowego

Ustawienia Fuse-bitów:
CKSEL3…0: 1101
SUT1…0: 11
CKDIV8: 1
CKOUT: 1

Listing 1. Funkcja inicjalizacyjna interfejsu USART mikrokontrolera do obsługi MIDI

```
//Definicja prędkości interfejsu MIDI
#define MIDI_BAUD 31250
#define CALCULATED_UBRR F_CPU/16/MIDI_BAUD-1

//Definicje bufora odbiorczego MIDI
#define MIDI_RX_BUF_SIZE 64
#define MIDI_RX_BUF_MASK (MIDI_RX_BUF_SIZE - 1)

void MIDIinit(void) {
    //Ustawienie prędkości interfejsu MIDI
    UBRRH = (CALCULATED_UBRR)>>8;
    UBRRL = CALCULATED_UBRR;
    //Załączenie odbiornika oraz aktywacja przerwania od odbioru danych
    UCSRB = (1<<RXEN)|(1<<RXCIE);
    //Ustawienie formatu ramki: 8bitów danych, 1 bit stopu
    UCSRC = (3<<UCSZ0);
}
```

Listing 3. Funkcja obsługi przerwania od odbioru danych interfejsu USART

```
ISR(USART_RX_vect) {
    uint8_t newHead, Data;
    Data = UDR;  //Pobieramy bajt danych z bufora sprzętowego USART

    //Obliczamy nowy indeks „głowy węża”
    newHead = (midiBuffer.Head + 1) & MIDI_RX_BUF_MASK;

    //Sprawdzamy, czy wąż nie zacznie zjadać własnego ogona
    if (newHead == midiBuffer.Tail) {
    //Przepełnienie bufora - można obsłużyć ten błąd
    } else {
        midiBuffer.Head = newHead;      //Zapamiętujemy nowy indeks „głowy węża”
        midiBuffer.Data[newHead] = Data;//Wpisujemy odebrany bajt do bufora FIFO
        ++midiBuffer.Bytes;
    }
}
```

Listing 4. Funkcja pobierająca dane z kołowego bufora MIDI

```
uint8_t MIDIgetByte(void){
    uint8_t Byte;

    ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
        //Obliczamy i zapamiętujemy nowy indeks „ogona węża” bufora MIDI
        midiBuffer.Tail = (midiBuffer.Tail + 1) & MIDI_RX_BUF_MASK;
        //Zwracamy bajt pobrany z bufora jako rezultat wykonania funkcji
        Byte =  midiBuffer.Data[midiBuffer.Tail];
        --midiBuffer.Bytes;
    }
    return Byte;
}
```

wprowadzamy globalną zmienną strukturalną odpowiedzialną za przechowywanie danych i obsługę kołowego bufora odbiorczego MIDI. Definicję wspomnianej zmiennej pokazano na **listingu 2**. Prosta funkcja obsługi przerwania od odbioru danych interfejsu USART (*USART_RX_vect*), której zadaniem jest napełnianie kołowego bufora odbiorczego MIDI nadchodzącymi danymi, została pokazana na **listingu 3**. Niezbędna jest prosta funkcja narzędziowa, dzięki której pobierzemy dane z tego bufora – kod pokazano na listingu **listingu 4**. Na koniec funkcja narzędziowa, dzięki której w bardzo prosty sposób sprawdzimy, czy w buforze MIDI są jakiekolwiek dane do analizy. Kod funkcji pokazano na **listingu 5**. Obsługa

Listing 5. Funkcja sprawdzająca obecność danych w buforze MIDI (zwraca wartość różną od 0, gdy w buforze znajdują się dane)

```
uint8_t MIDIisReady(void){
    return midiBuffer.Bytes;
}
```

Listing 6. Funkcja przygotowująca Timer0 do pracy jako programowy generator PWM

```
#define PWM_PORT PORTB
#define PWM_DDR DDRB

void PWMinit(void){
    PWM_DDR = 0xFF;           //Port PWM, jako wyjściowy
    TCCR0A = (1<<WGM01);      //Tryb CTC
    OCR0A = 77;               //Przerwanie 12800 razy na sekundę
    TCCR0B = (1<<CS01);       //Preskaler = 8
    TIMSK = (1<<OCIE0A);      //Aktywacja przerwania Timer/Counter0 Output Compare Match A
}
```

Listing 7. Funkcja obsługi przerwania układu Timer0 realizująca 8-kanałowy generator PWM

```
ISR(TIMER0_COMPA_vect){
    static uint8_t Counter;

    Counter = (Counter +1) & 0x7F;      //Licznik 7-bitowy

    for(uint8_t i=0; i<8; ++i){
        if(Counter < PWM[i]) PWM_PORT |= (1<<(7-i));
        else PWM_PORT &= ~(1<<(7-i));
    }
}
```

Listing 2. Definicja zmiennej odpowiedzialnej za przechowywanie danych i obsługę kołowego bufora odbiorczego MIDI

```
volatile struct{
    uint8_t Head;
    uint8_t Tail;
    uint8_t Bytes;
    uint8_t Data[MIDI_RX_BUF_SIZE];
} midiBuffer;
```

interfejsu MIDI korzysta z typowych mechanizmów obsługi programowych buforów kołowych i dzięki temu odbierane dane możemy analizować w dogodnej chwili.

Pora na przedstawienie bardzo prostej implementacji 8-kanałowego, 7-bitowego generatora PWM, za pomocą którego sterowane są kanały wyjściowe. Częstotliwość przebiegu PWM ustawiono na 100 Hz, co z powodzeniem wystarczy do sterowania diodami (taśmami) LED. Implementacja jest typowym rozwiązaniem i korzysta z wbudowanego w strukturę mikrokontrolera układu czasowo-licznikowego TIMER0 pracującego w trybie CTC i taktowanego sygnałem zegarowym mikrokontrolera podzielonym przez 8 (czyli 1 MHz). Konfigurację pokazano na listingu **listingu 6**. Tak zainicjowany układ czasowo-licznikowy Timer0 będzie zgłaszał przerwanie od porównania stanu licznika z rejestrem porównania OCR0A 12800 razy na sekundę, co przy założonej rozdzielczości kanałów PWM równej 7 bitów spowoduje generowanie przebiegu PWM o częstotliwości 100 Hz (na każdym z kanałów). Ciało obsługi funkcji odpowiedzialnej za realizację programowego generatora PWM pokazano na **listingu 7**. Funkcja jest na tyle prosta, że nawet jej częste wywoływanie nie obciąży za bardzo mikrokontrolera i nie będzie kolidowało z funkcją przerwania obsługującą interfejs MIDI.

Montaż i uruchomienie

Schemat montażowy urządzenia MidiLights pokazano na **rysunku 2**. Zaprojektowano bardzo zwarty obwód drukowany przeznaczony do montażu elementów przewlekanych. Montaż rozpoczynamy od wlutowania układów scalonych i rezonatora kwarcowego. Dalej

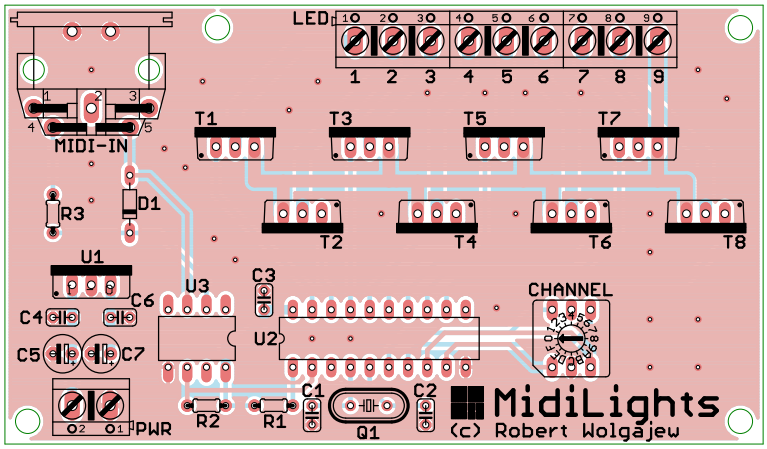
lutujemy stabilizator napięcia, tranzystory, diodę D1, rezystory i kondensatory a na samym końcu gniazda połączeniowe i zadajnik kodu CHANNEL. Poprawnie zmontowany układ nie wymaga żadnych regulacji i powinien działać zaraz po włączeniu zasilania. W przypadku przełączania dużej liczby diod lub taśm LED może zaistnieć potrzeba wyposażenia tranzystorów T1…T8 w stosowne do przełączanej (a dokładnie traconej) mocy radiatory. Dla napięcia sterującego bramką tranzystora MOSFET rzędu VGS = 5 V moc traconą możemy z pewnym przybliżeniem obliczyć ze wzoru

$$P = I^2 \cdot R_{DS(ON)}$$

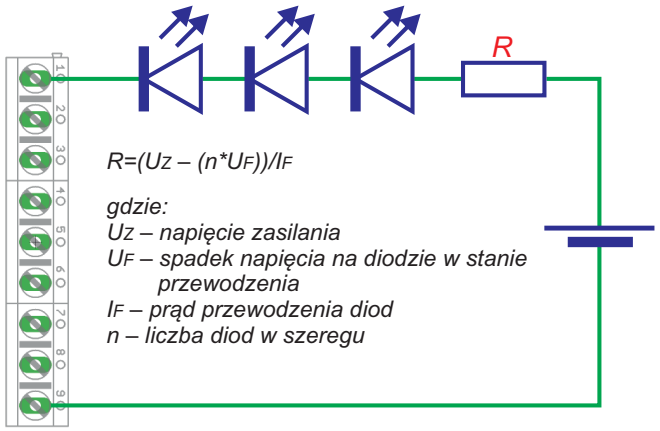
gdzie I to prąd przewodzenia diod LED, zaś RDS(ON) to rezystancja dren-źródło tranzystora w stanie załączenia równa około 0,1 Ω.

Zmontowane urządzenie MidiLights pokazano na fotografii tytułowej. Na **rysunku 3** zamieszczono przykładowy sposób podłączenia urządzenia MidiLights dla przypadku sterowania grupą diod LED. Podłączenie gotowych taśm LED będzie wyglądało nieco inaczej, gdyż taśmy takie mają zintegrowane rezystory ograniczające prąd diod LED, przy czym należy mieć na uwadze, że zaciski 1…8 złącza podłączeniowego to wyjścia tranzystorów sterujących poszczególnymi kanałami PWM, zaś zacisk 9 to połączenie masowe (wspólne).

Robert Wołgajew, EP



Rysunek 2. Schemat montażowy sterownika



Rysunek 3. Sposób podłączenia urządzenia w przypadku sterowania grupą diod LED