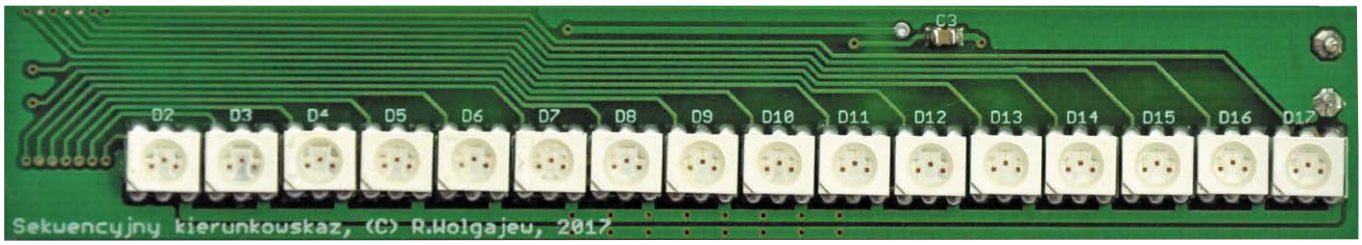


Autor składa podziękowania panu Krystianowi Krywaldowi z firmy Maritex za dostarczenie diod power LED zastosowanych w projekcie.

Sekwencyjny kierunkowskaz dynamiczny

Taki kierunkowskaz jest przez niektórych uważany za zbędny i nieładny gadżet, jednak pomijając sprawę gustu trzeba przyznać, że jego działanie przykuwa uwagę i przez to może się przyczynić do poprawy bezpieczeństwa na drodze, bo jeśli coś zwraca uwagę, to trudno to przeoczyć. Mowa o kierunkowskazach, które złożone są z linijki diod LED, w której poszczególne diody są zaświecane sekwencyjnie tworząc „falę świetlną”.

Rekomendacje: *kierunkowskaz jest przeznaczony dla miłośników tuningu swoich pojazdów.*

W trakcie swojej długoletniej przygody z techniką mikroprocesorową wielokrotnie podejmowałem wyzwanie konstruowania urządzeń związanych z szeroko rozumianą motoryzacją. Przecież jakby na to nie patrzeć, jest to dziedzina, którą interesuje się niemal każdy i która to jest świetną platformą do implementacji różnego rodzaju nowinek technologicznych. Jedną z takich nowinek, choć słowo to może nie do końca jest tutaj na miejscu, jest stosowane przez jednego z czołowych producentów motoryzacyjnych, rozwiązanie nazywane kierunkowskazami dynamicznymi. Niby nic nadzwyczajnego i przez niektórych uważane za zbędny i nieładny gadżet, jednak pomijając sprawę gustu, o którym, jak mawia przysłowie „się nie dyskutuje”, trzeba przyznać, że przykuwa uwagę a to, z kolei, może się przyczynić do wymiernego wzrostu poziomu bezpieczeństwa, bo skoro coś przykuwa uwagę to trudno to przeoczyć. Mowa o kierunkowskazach, które złożone są z linijki diod LED, w której poszczególne diody są zaświecane sekwencyjnie tworząc „falę świetlną”.

Dla przyzwoitości należy wspomnieć, iż nie jest to rozwiązanie innowacyjne, ponieważ światła kierunkowskazu załączane w sposób sekwencyjny stosował na przełomie lat 60-tych i 70-tych ubiegłego wieku amerykański producent spod znaku Mercury w modelu Cougar oraz Ford w modelu Thunderbird. Co ciekawe, w tamtym okresie było to rozwiązanie czysto mechaniczne. Nie znosząc więc nudy, postanowiłem skonstruować takie urządzenie. Na rynku, który jak wiadomo nie znosi „próżni”, jest wiele ofert „przeróbki” oryginalnych kierunkowskazów na wersję sekwencyjną. Niestety, nie są one zbyt atrakcyjne cenowo. Pora na urządzenie niedrogie, proste w konstrukcji i uniwersalne w stosowaniu.

Trzeba przyznać, że rozwiązaniem, które na samym początku przyszło mi do głowy było zastosowanie typowego, szeregowego rejestru przesuwnego, taktowanego przebiegiem o ustalonej częstotliwości, którego wyjścia poprzez elementy kluczujące (dla przykładu tranzystory MOSFET) bezpośrednio sterowałyby diodami LED. Rozwiązanie bardzo proste, jednak problem w tym, że aby zapewnić odpowiednią widzialność kierunkowskazów należy zastosować diody LED mocy, a te z kolei wymagają do zasilania źródeł prądowych. Uniemożliwia to zasilanie diod LED wprost z dodatniego bieguna zasilania przez zwykłe tranzystory MOSFET, ponieważ w takim rozwiązaniu nie

DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl
USER: 95777, PASS: 53wtjyf6

W ofercie AVT*
AVT-5611

Podstawowe informacje:

- Dwustronna płytka drukowana.
- Napięcie zasilające: 11...15 V DC.
- Maksymalny prąd zasilania: 0,9 A.
- Pomarańczowe diody LED mocy firmy CREE.
- Specjalizowany sterownik diod MBI5030 firmy Macroblock.
- Mikrokontroler AVR typu ATtiny10 (SOT23/6).

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-1945	Moduł komfortowych kierunkowskazów (EP 3/2017)
Projekt 226	Kontroler oświetlenia przyczepy ciągnika rolniczego (EP 2/2016)
AVT-1877	Automatyczny wyłącznik zasilania do instalacji samochodowej (EP 8/2015)
AVT-1850	Automatyczny zmierzchowy przełącznik świateł dziennych na mijania (EP 4/2015)
AVT-5454	Automatyczny sterownik świateł do jazdy dziennej (EP 56/2014)
AVT-3095	Komputer samochodowy (EdW 4/2014)
AVT-5395	TIDex – komputer dla samochodów z silnikiem Diesla (EP 5/2013)
AVT-1599	Softstart do żarówek samochodowych H7 (EP 11/2010)
AVT-990	Automatyczny włącznik świateł w samochodzie (EP 6/2007)
AVT-528	Inteligentny sterownik oświetlenia wnętrza samochodu (EP 12/2006)
AVT-434	Komputer samochodowy (EP 9/2005)
AVT-2733	Sterownik lampki do kabiny samochodu (EdW 12/2004)

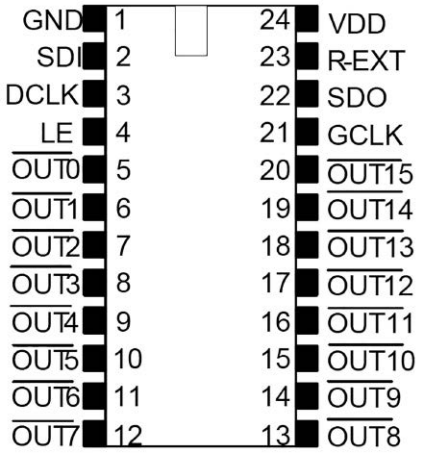
Ustawienia fusebitów:

CKOUT: 0
RSTDISBL: 0

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (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 dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają 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***



Rysunek 1. Rozmieszczenie wyprowadzeń układu MBI5030

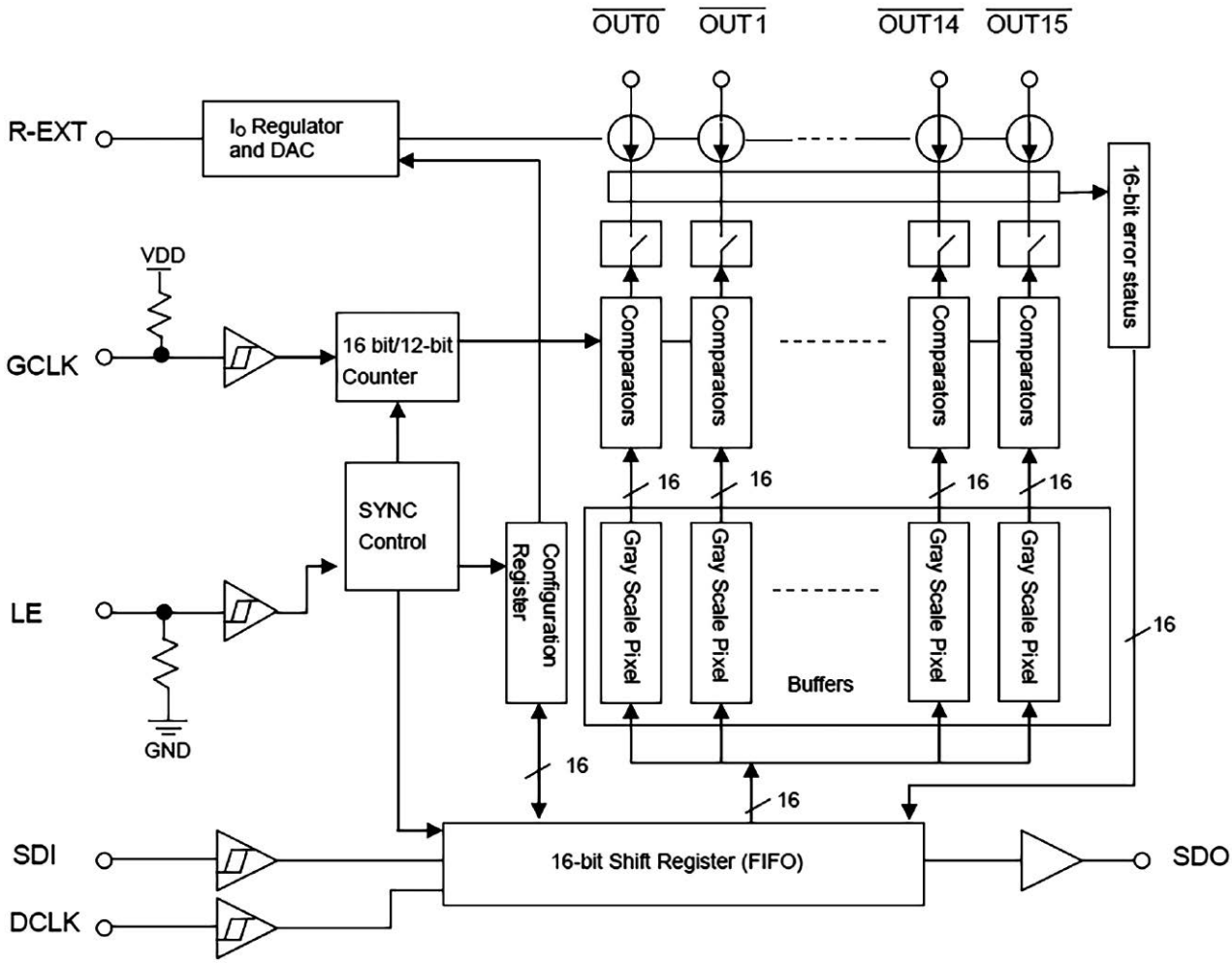
możemy zapewnić stałego prądu zasilającego diodę w miarę wzrostu temperatury jej struktury, co wcześniej czy później doprowadziłoby do jej zniszczenia. Inną sprawą jest fakt, że typowy rejestr przesuwny zapewniłby wyłącznie możliwość „zwykłego" zaświecania/gaszenia sterowanej diody nie pozwalając, dla przykładu, na kontrolowanie jasności jej świecenia w celu uzyskania bardziej realistycznego efektu „fali świetlnej" (dzięki zastosowaniu „płynnego" przechodzenia światła z diody na diodę).

Tabela 1. Nazwy i opis wyprowadzeń układu MBI5030

Nazwa	Opis
GND	Masa zasilania
SDI	Wejście danych szeregowego rejestru przesuwnego
DCLK	Sygnał zegarowy szeregowego rejestru przesuwnego
LE	Wejście „strobe" szeregowego rejestru przesuwnego
OUT0...OUT15	Wyjścia źródeł prądowych (zanegowane)
GCLK	Zewnętrzny sygnał zegarowy generatora PWM
SDO	Wyjście danych szeregowego rejestru przesuwnego
R_{EXT}	Rezystor ustalający wartość prądu wyjściowego/kanał
VDD	Napięcie zasilania układu

Kolejnym pomysłem, który przyszedł mi do głowy było użycie typowego mikrokontrolera w funkcji elementu generującego przebieg PWM dla każdej z diod linijki świetlnej, co nie powinno stanowić większego problemu z punktu widzenia oprogramowania. Jednak i w tym wypadku do rozwiązania pozostaje kwestia stałoprądowego zasilania diod LED. Nawet, jeśli przyjąć, że rozwiązanie z tranzystorem MOSFET w funkcji elementu sterującego jest wystarczające, to problematycznym wydaje się umieszczenie 16 takich elementów (powiedzmy, że z tylu elementów będzie składała się nasza linijka świetlna) na małej płytce drukowanej, jakiej oczekiwalibyśmy dla urządzenia tego typu. Pozostało poszukać rozwiązania przeznaczonego specjalnie do takich aplikacji. Tutaj z pomocą przychodzi nam firma Macroblock, która jest producentem szeregu układów-sterowników diod LED. W naszym wypadku idealnym wydaje się być zastosowanie układu MBI5030, który jest 16-kanałowym, specjalizowanym sterownikiem diod LED, charakteryzującym się następującymi, wybranymi cechami funkcjonalnymi:

- Wbudowany, 16-bitowy, szeregowy rejestr przesuwny.
- 16 wyjściowych źródeł stałoprądowych z regulacją PWM (maksymalny prąd 90 mA/kanał).



Rysunek 2. Schemat blokowy układu MBI5030

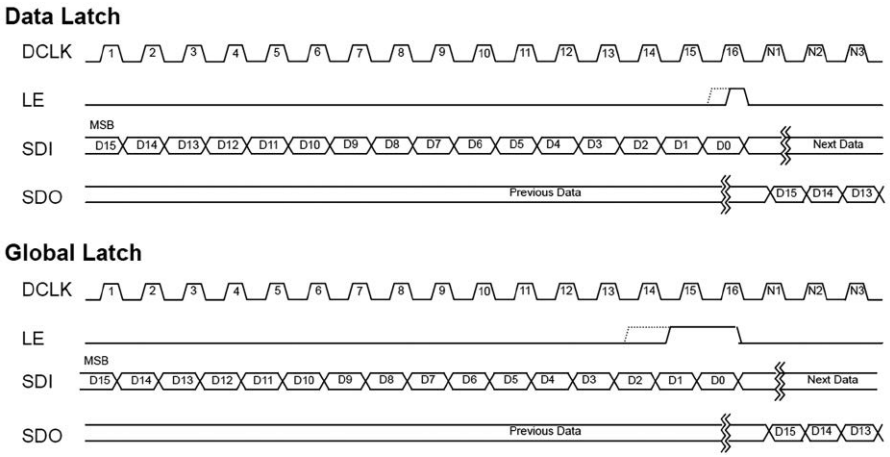
- 16- lub 12-bitowa rozdzielczość przebiegów PWM.
- Regulacja prądu wyjściowego za pomocą zewnętrznego rezystora R_{EXT} oraz programowo, przy użyciu rejestru konfiguracyjnego.
- Zastosowanie technologii scrambled-PWM, która zmniejsza zjawisko migotania, dzięki dzieleniu czasu załączenia kanału na wiele krótszych impulsów.
- Wbudowany mechanizm opóźnień między załączaniem poszczególnych wyjść dla zmniejszenia zjawiska udaru prądowego.
- Wykrywanie rozwarcia/zwarcia na każdym kanale LED.
- Wbudowane zabezpieczenie temperaturowe z funkcją alarmu.
- Niezależne wejście sygnału zegarowego generatora PWM.
- Napięcie zasilania 3…5 V.

Układ MBI5030 idealnie pasuje do naszych oczekiwań dostarczając rozwiązania „szytego na miarę". Rozmieszczenie wyprowadzeń układu pokazano na **rysunku 1**, natomiast w **tabeli 1** umieszczono ich szczegółowy opis. Tak, jak wspomniano we wstępie, układ MBI5030 jest swego rodzaju 16-bitowym rejestrem przesuwnym wyposażonym w 16 niezależnych źródeł prądowych oraz grupę wewnętrznych buforów danych, które przechowują nastawy PWM każdego ze źródeł. Schemat blokowy układu pokazano na **rysunku 2**.

Sterowanie pracą takiego układu, jak każdego rejestru przesuwnego, polega na podaniu na wejście danych SDI kolejnych bitów 16-bitowego słowa danych reprezentującego wartość wypełnienia kolejnego kanału PWM (począwszy od bitu MSB), a na wejście DCLK sygnału zegarowego o maksymalnej częstotliwości 30 MHz. W odróżnieniu jednak on „normalnego" rejestru przesuwnego wprowadzono tutaj dodatkowe wejście LE, którego zachowanie determinuje znaczenie przesłanego słowa. Stanem spoczynkowym wejścia LE jest logiczne „0". Wystąpienie logicznej „1" na wejściu LE w zależności o czasu jej trwania (w taktach zegara sygnału DCLK) oraz miejsca jej wystąpienia (w odniesieniu do zakresu przesyłanych bitów) determinuje sposób zachowania się układu MBI5030. Dostępnych jest 5 możliwości:

- odebranie sygnału „data latch",
- odebranie sygnału „global latch",
- żądanie odczytu konfiguracji układu,
- zapis konfiguracji układu,
- odczyt rejestru błędów.

Nie będę w tym miejscu szczegółowo opisywał wszystkich możliwości układu MBI5030, ponieważ stosowne informacje znajdziemy w dokumentacji producenta, a domyślna konfiguracja jest wystarczająca na potrzeby większości zastosowań. Skupię się więc na wymienionych dwóch pierwszych zdarzeniach.



Rysunek 3. Przebiegi sygnałów sterujących dla funkcji *data latch* i *global latch*

Zgodnie z tym, co napisano wcześniej, układ MBI5030 ma 16 kanałów PWM o rozdzielczości 16-bitowej, z czego wynika, że dla przesłania ustawień wszystkich kanałów jest konieczne przesłanie 16 słów 16-bitowych „zajmujących" 256 taktów zegara DCLK. Aby umożliwić jednoczesną zmianę ustawień wszystkich kanałów PWM sterownik MBI5030 wyposażono w grupę 16 wewnętrznych buforów danych, które przechowują ustawienie każdego z kanałów PWM. Przesłaniu nastaw PWM kanałów OUT15 do OUT1 powinien towarzyszyć sygnał *data latch*, który powoduje każdorazowe przepisanie wartości rejestru przesuwnego do odpowiedniego bufora danych, a przesłaniu nastaw PWM kanału OUT0 powinien towarzyszyć sygnał *global latch*, który powoduje przepisanie wartości rejestru przesuwnego do bufora danych kanału OUT0 i przepisanie wartości wszystkich buforów danych do rejestrów sterujących generatora PWM, a co za tym idzie – wprowadzenie „w życie" przesłanych ustawień. W ten sposób zmiana ustawień wszystkich kanałów PWM następuje w jednej chwili, nie licząc automatycznego mechanizmu opóźnień między załączaniem poszczególnych wyjść. Przebiegi sygnałów sterujących dla przypadku *data latch* i *global latch* pokazano na **rysunku 3**.

Jak łatwo się domyślić, wyjście SDO służy do kaskadowego łączenia wielu układów MBI5030 tak, ja ma to miejsce w przypadku łączenia „zwykłych" rejestrów przesuwnych. Wejście GCLK, jak wspomniano wcześniej, jest wejściem zewnętrznego sygnału zegarowego generatora PWM (maksymalnie 30 MHz), dzięki któremu możliwe jest w ogóle generowanie przebiegów na wyjściach OUT15…OUT0 sterownika. Częstotliwość sygnału PWM na wyjściu każdego z kanałów obliczamy według wzoru $F_{PWM}=F_{GCLK}/2^{16}$ lub $F_{PWM} = F_{GCLK}/2^{12}$ zależnie od wybranej programowo (rejestr konfiguracyjny) rozdzielczości sygnału PWM (domyślnie 16 bitów). Wejście R_{EXT} służy do przyłączenia rezystora ustalającego prąd źródła prądowego każdego z wyjść (maksymalnie 90 mA przy 5 V lub 70 mA przy 3,3 V), którego wartość obliczamy ze wzoru $I_{OUT}=(14.213\times G)/R_{EXT}$, gdzie G jest współczynnikiem korekcyjnym (domyślnie równym 1) możliwym do ustawienia z poziomu rejestru konfiguracyjnego.

To tyle, jeśli chodzi o niezbędny opis naszego sterownika LED. Pora na przedstawienie schematu urządzenia, który to pokazano na **rysunku 4**. Jest to nieskomplikowany system mikroprocesorowy, którego „sercem" jest mikrokontroler ATtiny10 w 6-wyprowadzeniowej obudowie SOT-23. Jest on odpowiedzialny za sterowanie pracą układu MBI5030, w tym dostarczając zewnętrznego sygnału taktującego generator PWM (wyjście CLKO mikrokontrolera) o częstotliwości GCLK=8 MHz, co zapewnia uzyskanie przebiegu PWM o częstotliwości ok.122 Hz. Generowanie przebiegu na wyjściu CLKO jest możliwe po ustawieniu odpowiedniego bitu konfigurującego mikrokontrolera.

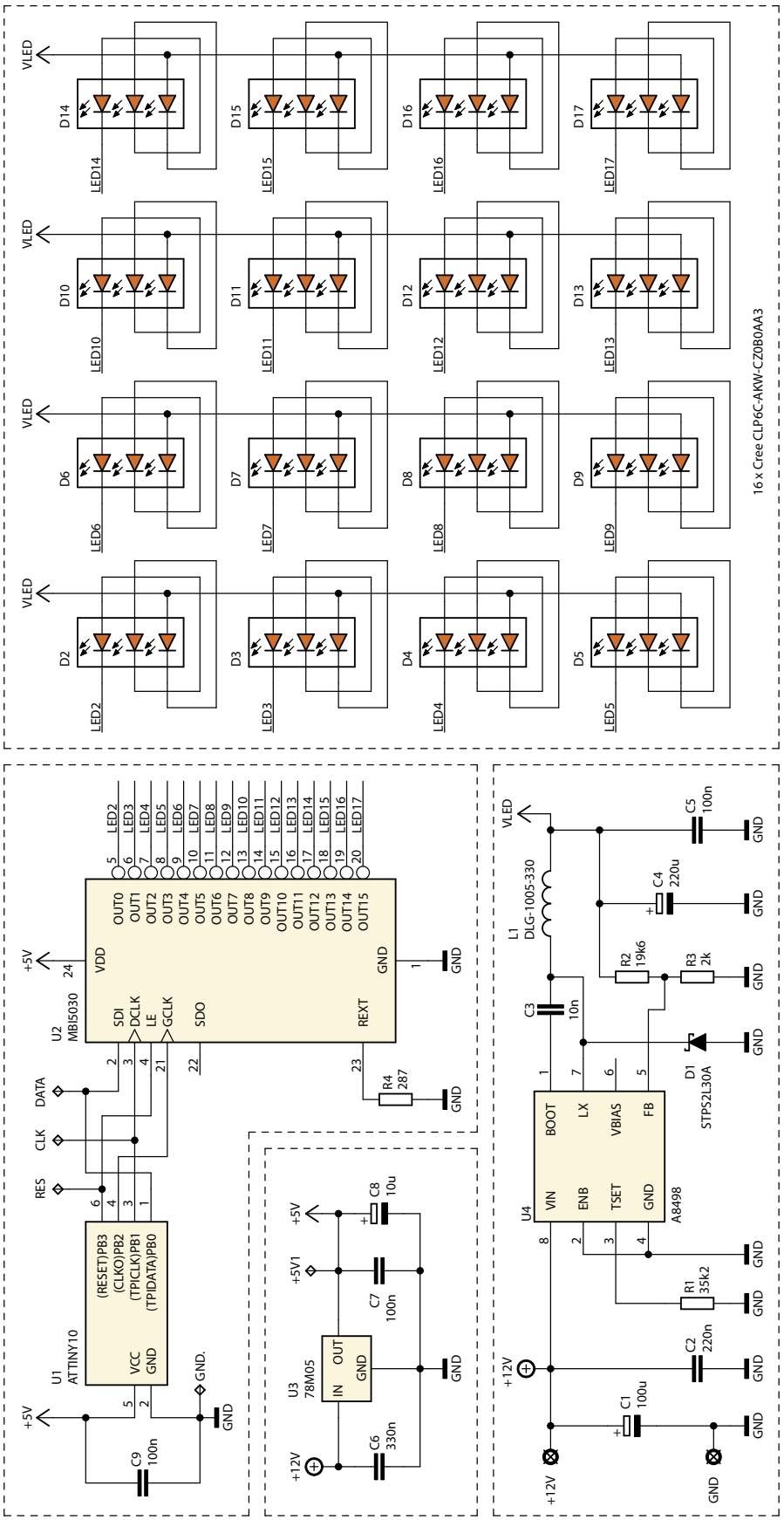
Słowo komentarza należy się sekcji zasilania, której budowa, z uwagi na dość duży sumaryczny prąd diod LED (rzędu 0,8 A) wymagała gruntownego przemyślenia. Ostatecznie zdecydowałem się na rozdzielenie napięcia zasilania mikrokontrolera i sterownika MBI5030 (w naszym wypadku 5 V) od napięcia zasilającego diody LED V_{LED}. Pierwszy z zasilaczy, z uwagi na niewielki pobór mocy, wykonałem przy użyciu zwykłego stabilizatora liniowego serii 78M05 w jego typowej aplikacji, która nie wymaga komentarza. Drugi, czyli ten o stosunkowo dużym prądzie wyjściowym, wykonałem z zastosowaniem scalonej przetwornicy step-down pod postacią układu A8498 firmy Allegro MicroSystems w jego typowej aplikacji. Zastosowanie takiego rozwiązania pozwoliło na osiągnięcie wysokiej sprawności (rzędu 86%) a co za tym idzie niewielkich strat mocy w postaci wydzielanego ciepła. Pewnej uwagi wymagał jednak dobór wartości napięcia wyjściowego tej części zasilacza, jako że jest to napięcie zasilające grupę diod LED. Dobór wartości tegoż napięcia najlepiej

ilustruje **rysunek 5**. Wynika z niego, że wartość napięcia zasilającego diody LED obliczamy ze wzoru $V_{LED}=V_{DS}+V_f+V_{Drop}$, gdzie:

- V_{DS} – minimalne napięcie na wyjściu układu MBI5030 zapewniające stabilizację prądu wyjściowego, odczytane z charakterystyki $V_{DS} = f(I_{OUT})$.
- V_f – napięcie przewodzenia zastosowanych diod LED.
- V_{Drop} – spadek napięcia na dodatkowym elemencie ograniczającym prąd (rezystor lub dioda Zenera, w naszym przypadku równy 0).

Dla ustalonego rezystorem R_{EXT} prądu wyjściowego o natężeniu 50 mA/kanał odczytana wartość napięcia V_{DS} wynosi ok. 0,8 V, co dla napięcia przewodzenia szeregu trzech diod LED rzędu 7,8 V daje napięcie zasilania diod LED wynoszące 8,6 V. Właśnie taka wartość została ustalona na wyjściu naszej przetwornicy, a do jej ustawienia służy dzielnik rezystorowy R2/R3, który określa wartość napięcia wyjściowego zgodnie ze wzorem $V_{LED}=0{,}8\ V\times(1+R2/R3)$. Oczywiście, bazując na nocie układu MBI5030, można by było przyjąć do obliczeń wyższą wartość napięcia V_{DS} (maksymalnie 17 V), aby mieć większy zapas, jeśli chodzi o potrzebę zapewnienia stabilizacji prądu wyjściowego źródeł prądowych, jednak odbiłoby się to niekorzystnie na mocy traconej w układzie sterownika diod LED, a co za tym idzie, na ilości wydzielanego ciepła. W związku z powyższym przyjęto wartość napięcia wyjściowego przetwornicy równą 8,6 V. Warto także podkreślić, iż w projekcie zastosowano wysokiej jakości, markowe, ultra-jasne, pomarańczowe diody LED produkcji firmy Cree, które przy prądzie przewodzenia 50 mA osiągają światłość około 9000 mcd.

To tyle, jeśli chodzi o opis naszego urządzenia, w związku z czym pora na przedstawienie szczegółów implementacji programowej obsługi sterownika MBI5030. Na początek przedstawię zawartość niewielkiego pliku nagłówkowego związanego z obsługą przedmiotowego układu. Wspomniany kod zamieszczono na **listingu 1**. Dalej, na **listingu 2** pokazano funkcję odpowiedzialną za wysłanie 16-bitowego słowa do układu MBI5030 z rozróżnieniem jego rodzaju z zakresu *data latch* lub *global latch* (o czym



Rysunek 4. Schemat ideowy sekwencyjnego kierunkowskazu

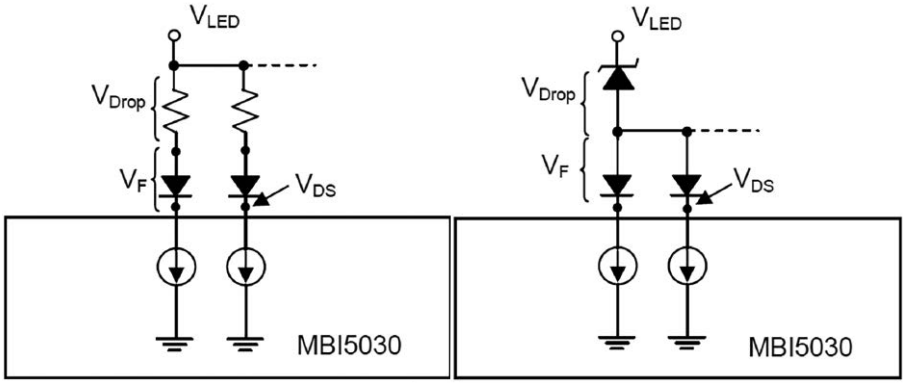
Wykaz elementów:

Rezystory: (SMD 0805, 1%)
R1: 35,2 kΩ
R2: 19,6 kΩ
R3: 2 kΩ
R4: 287 Ω

Kondensatory: (SMD 0805)
C1: 100 µF/16 V (SMD „D”)
C2: 220 nF
C3: 10 nF
C4: µF/16 V (SMD „D”)
C5, C7, C9: 100 nF
C6: 330 nF
C8: 10 µF/16 V (SMD „A”)

Półprzewodniki:
U1: ATtiny10 (SOT23/6)
U2: MBI5030GTS (TSSOP24)
U3: 78M05 (DPAK)
U4: A8498 (SOIC08)
D1: STPS2L30A (SMA)
D2...D17: dioda LED Cree CLP6C-AKW-CZ-0B0AA3 (PLCC-6)

Inne:
L1: 33 µH dławik SMD typu DLG-1005-330



Rysunek 5. Sposób doboru wartości napięcia zasilania diod LED

Listing 1. Zawartość pliku nagłówkowego obsługi sterownika MBI5030

```
#define MBI5030_PORT PORTB
#define MBI5030_DDR DDRB
#define MBI5030_SDI PB3
#define MBI5030_DCLK PB1
#define MBI5030_LE PB0
#define SET_SDI MBI5030_PORT |= (1<<MBI5030_SDI)
#define RESET_SDI MBI5030_PORT &= ~(1<<MBI5030_SDI)
#define SET_DCLK MBI5030_PORT |= (1<<MBI5030_DCLK)
#define RESET_DCLK MBI5030_PORT &= ~(1<<MBI5030_DCLK)
#define SET_LE MBI5030_PORT |= (1<<MBI5030_LE)
#define RESET_LE MBI5030_PORT &= ~(1<<MBI5030_LE)

//Definicja typu sygnału LE układu MBI5030 (wyłącznie 2 rodzaje)
#define DATA_LATCH 0
#define GLOBAL_LATCH 1
```

Listing 2. Funkcja wysyłająca słowa 16-bitowe do układu MBI5030

```
void writeWord(uint16_t Word, uint8_t latchType)
{
    for(uint8_t i=0; i<14; ++i)  //First 14 bits
    {
        if(Word & 0x8000) SET_SDI; else RESET_SDI;
        Word <<= 1;
        SET_DCLK; RESET_DCLK;  //Tick
    }
    if(latchType == GLOBAL_LATCH) SET_LE;  //We assert LE if GLOBAL_LATCH needed
    if(Word & 0x8000) SET_SDI; else RESET_SDI;
    Word <<= 1;
    SET_DCLK; RESET_DCLK;  //15th tick
    SET_LE;  //DATA_LATCH
    if(Word & 0x8000) SET_SDI; else RESET_SDI;
    SET_DCLK; RESET_DCLK;  //16th tick
    RESET_LE;
}
```

Listing 3. Funkcja wysłająca nastawy 16 kanałów PWM układu MBI5030

```
inline void setPWM(uint8_t Channels[])
{
    for(uint8_t i=0; i<16; ++i) writeWord(Channels[i]<<8, i==15? GLOBAL_LATCH:DATA_LATCH);
}
```

Listing 4. Tablica wartości funkcji korekcji Gamma

```
const uint16_t Gamma[] PROGMEM =
{
    0, 0, 1, 3, 6, 11, 16, 22, 30, 39, 49, 61,
    74, 88, 104, 121, 140, 160, 182, 205, 229, 256, 283, 313, 344, 376, 411,
    447, 484, 523, 564, 607, 651, 697, 745, 794, 846, 899, 954, 1010, 1069,
    1129, 1191, 1255, 1320, 1388, 1457, 1529, 1602, 1677, 1754, 1833, 1913,
    1996, 2081, 2167, 2255, 2346, 2438, 2533, 2629, 2727, 2827, 2930, 3034,
    3140, 3248, 3359, 3471, 3585, 3702, 3820, 3940, 4063, 4188, 4314, 4443,
    4574, 4707, 4842, 4979, 5118, 5259, 5403, 5548, 5696, 5846, 5998, 6152,
    6308, 6467, 6627, 6790, 6955, 7122, 7291, 7463, 7636, 7812, 7990, 8171,
    8353, 8538, 8725, 8914, 9105, 9299, 9495, 9693, 9893, 10096, 10301,
    10508, 10718, 10929, 11143, 11359, 11578, 11799, 12022, 12247, 12475,
    12705, 12937, 13172, 13409, 13648, 13890, 14134, 14380, 14629, 14880,
    15133, 15389, 15647, 15907, 16170, 16435, 16703, 16973, 17245, 17520,
    17797, 18076, 18358, 18642, 18929, 19218, 19509, 19803, 20099, 20398,
    20699, 21003, 21309, 21617, 21928, 22241, 22557, 22875, 23196, 23519,
    23844, 24172, 24503, 24836, 25171, 25509, 25849, 26192, 26537, 26885,
    27235, 27588, 27943, 28301, 28662, 29024, 29390, 29757, 30128, 30501,
    30876, 31254, 31634, 32017, 32403, 32791, 33181, 33574, 33970, 34368,
    34769, 35172, 35578, 35986, 36397, 36811, 37227, 37646, 38067, 38491,
    38917, 39346, 39778, 40212, 40649, 41088, 41530, 41975, 42422, 42871,
    43324, 43779, 44236, 44697, 45159, 45625, 46093, 46564, 47037, 47513,
    47992, 48473, 48957, 49443, 49932, 50424, 50919, 51416, 51915, 52418,
    52923, 53431, 53941, 54454, 54970, 55488, 56010, 56533, 57060, 57589,
    58121, 58655, 59192, 59732, 60275, 60820, 61368, 61919, 62472, 63028,
    63587, 64149, 64713, 65280
}
```

Listing 5. Funkcji wysyłająca nastawy 16 kanałów PWM z uwzględnieniem korekcji Gamma

```
inline void setPWM(uint8_t Channels[])
{
    for(uint8_t i=0; i<16; ++i) writeWord(pgm_read_word(&Gamma[Channels[i]]), i==15? GLOB-
AL_LATCH:DATA_LATCH);
}
```

pisano wcześniej). Kolejna, funkcja służy do wysłania nastaw wszystkich 16 kanałów PWM sterownika LED, a którą to zaprezentowano na **listingu 3**. Jej argumentem jest wskaźnik do tablicy złożonej z 16 elementów 8-bitowych. Uważny Czytelnik zastanowi się w tym miejscu, dlaczego 8-bitowych, skoro rozdzielczość każdego kanału wynosi 16-bitów? Odpowiedź jest nader prosta. Celowo ograniczyłem rozdzielczość każdego kanału PWM, ponieważ 8-bitowa rozdzielczość zapewniająca 256 poziomów jasności diody LED jest moim zdaniem w zupełności wystarczająca w takim zastosowaniu. Poza tym, nie bez znaczenia jest fakt, że nasz niewielki mikrokontroler ATtiny10 ma wbudowaną niewielką pamięć RAM mieszczącą zaledwie 32 bajty, więc stosowanie tablicy 16-bitowej byłoby wręcz niemożliwe. Warto w tym miejscu podkreślić, iż funkcja z list. 3 nie uwzględnia tzw. korekcji Gamma, czyli korekcji jasności diody LED biorącej pod uwagę właściwość ludzkiego wzroku i nieliniowy sposób, w jaki ludzkie oko postrzega światło widzialne. W implementacji naszego urządzenia jest to praktycznie bez znaczenia, jednak dość często wymaga uwzględnienia, gdyż w przeciwnym wypadku przy małych poziomach jasności będziemy obserwowali znaczące różnice w jasności diody LED dla kolejnych wartości wypełnienia przebiegu PWM, zaś dla większych wartości praktyczny brak różnicy jasności. Aby uwzględnić korekcję Gamma należy liniowy ciąg wartości wejściowych (0…255) „przetworzyć" według następującej zależności (korzystamy tutaj z 16-bitowej rozdzielczości kanałów PWM)

$$f(x) = 65535 \cdot \left(\frac{x}{255}\right)^{\frac{1}{\gamma}}$$

dla γ=0,45.

Mikrokontrolery AVR mają zbyt małą moc obliczeniową, aby tego typu równania obliczać „w locie", więc warto wygenerować stosowną tablicę wartości i bezpośrednio odczytywać potrzebne dane. Dla funkcji, jak wyżej przykładową tablicę zamieszczono na **listingu 4**. Aby została wspomniana korekta była wykonywana, funkcja przeznaczona do wysłania nastaw wszystkich 16 kanałów PWM sterownika LED wymaga modyfikacji – zmienioną funkcję pokazano na **listingu 5**.

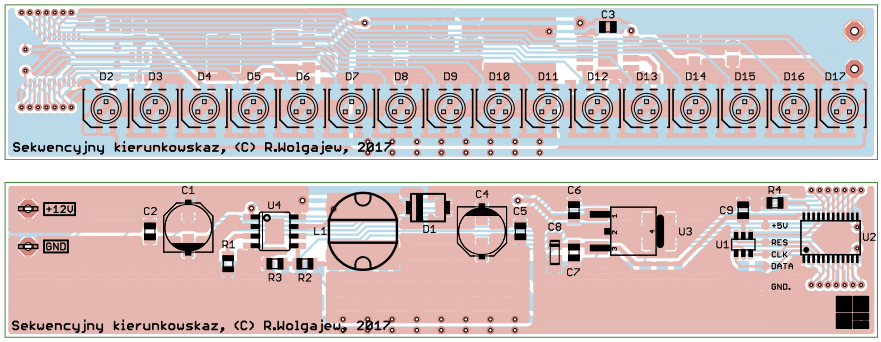
Na koniec przedstawię zawartość funkcji *main()*, która oprócz ustawienia właściwości zasobów sprzętowych mikrokontrolera, jak porty wejścia/wyjścia czy preskaler zegara taktującego, odpowiedzialna jest wyłącznie za generowanie efektu „fali świetlnej". Efekt taki uzyskano zwiększając wartość wypełnienia przebiegu PWM dla kolejnych kanałów sterownika MBI5030 w taki sposób, że wypełnienie kanału n+1 zwiększane jest od momentu, gdy wypełnienie kanału n osiąga poziom równy 25% (dobrany

eksperymentalnie). Czas pomiędzy poszczególnymi krokami regulacji dobrano w taki sposób by cały proces przebiegu „fali świetlnej" zajmował około 0,8 sekundy. Funkcję *main* pokazano na **listingu 6**.

Listing 6. Funkcja main

```
int main(void)
{
    //Porty sterujące dla układu MBI5030, jako wyjściowe ze stanem 0
    MBI5030_DDR = (1<<MBI5030_SDI)|(1<<MBI5030_DCLK)|(1<<MBI5030_LE);
    //Ustawiamy preskaler zegara taktującego na 1, by osiągnąć prędkość taktowania równą 8MHz
    CCP = 0xD8;  //Configuration Change Protection
    CLKPSR = 0x00;  //Clock Prescaler Select
    while(1)
    {
        //Symulacja „fali" świetlnej
        while(PWM[15]<255)
        {
            if(PWM[0]<255) ++PWM[0];
            for(uint8_t i=1; i<16; ++i) if(PWM[i]<255 && PWM[i-1]>64) ++PWM[i];
            setPWM(PWM);
        }
        _delay_ms(100);
        for(uint8_t i=0; i<16; ++i) PWM[i] = 0;  //Wygaszenie wszystkich diod LED
        setPWM(PWM);
        _delay_ms(300);
    }
}
```

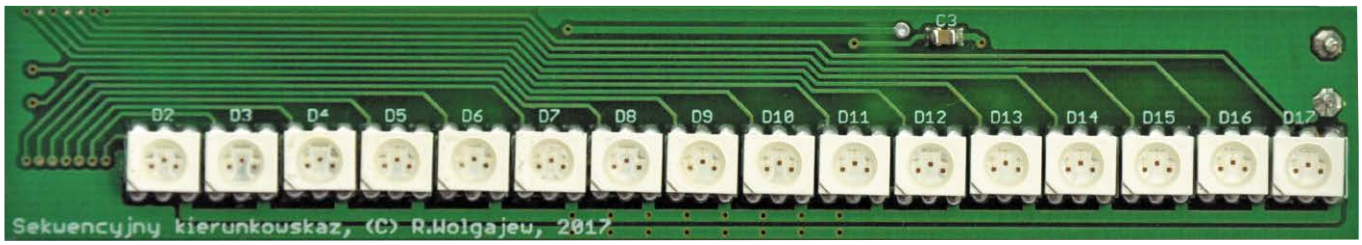
Schemat montażowy sekwencyjnych kierunkowskazów pokazano na **rysunku 6**. Zaprojektowano zwarty, dwustronny obwód drukowany przeznaczony do montażu powierzchniowego. Z uwagi na dość duży prąd szczytowy przetwornicy zasilającej diody LED mocy zadbano o odpowiednie prowadzenie sygnałów krytycznych oraz masy zasilania. Warto podkreślić, iż nieodpowiednie prowadzenie ścieżek zasilających anody diod LED mogłoby doprowadzić do powstania dość dużych indukcyjności pasożytniczych, co przy sterowaniu PWM w konsekwencji doprowadziłoby do powstawania przepięć mogących uszkodzić sterownik MBI5030. Dociekliwym Czytelnikom polecam lekturę dokumentacji firmy Macroblock opatrzoną tytułem *„Application note for 8-bit and 16-bit LED Drivers – Overshoot"*. Montaż rozpoczynamy od przylutowania wszystkich diod LED oraz kondensatora C3 od strony BOTTOM obwodu drukowanego. Następnie przechodzimy na stronę TOP, gdzie w pierwszej kolejności montujemy wszystkie półprzewodniki. Pewnej uwagi jak i wprawy wymaga wlutowywanie układu sterownika diod LED MBI5030 z powodu dość dużego zagęszczenia jego wyprowadzeń. Najłatwiejszym sposobem montażu takich elementów jednocześnie niewymagającym posiadania specjalistycznego sprzętu jest użycie typowej stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń. Jakość montażu sprawdzamy pod lupą korzystając z miernika pozwalającego sprawdzić ciągłość połączeń. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowana płytkę sterownika przemyjemy alkoholem izopropylowym. Następnie montujemy kondensatory ceramiczne i rezystory, kondensatory elektrolityczne, a na samym końcu dławik L1.



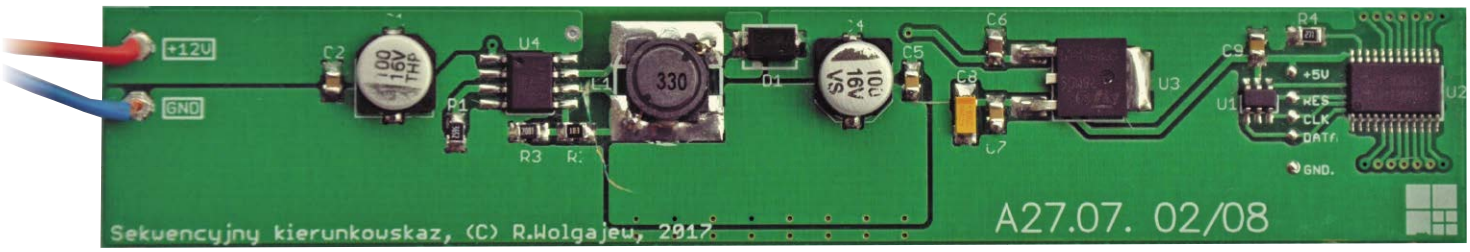
Rysunek 6. Schemat montażowy sekwencyjnych kierunkowskazów

Poprawnie zmontowany z użyciem zaprogramowanego mikrokontrolera kierunkowskaz nie wymaga jakichkolwiek regulacji i powinien działać tuż po włączeniu zasilania. Warto jednak podkreślić, że programowanie mikrokontrolera naszego urządzenia wymaga zachowania stosownej kolejności wykonywanych czynności. W pierwszej kolejności należy zaprogramować pamięć Flash mikrokontrolera a dopiero później ustawić bity konfiguracyjne. Wynika to z faktu, iż po ich ustawieniu zostanie zablokowane wejście RESET mikrokontrolera niezbędne przy programowaniu za pomocą programatora ISP. Na **fotografii 7** przedstawiono wygląd zmontowanej płytki drukowanej urządzenia widzianej od góry, a na **fotografii 8** od spodu.

Robert Wołgajew, EP



Fotografia 7. Wygląd zmontowanej płytki drukowanej urządzenia widzianej od góry



Fotografia 8. Wygląd zmontowanej płytki drukowanej urządzenia widzianej od spodu