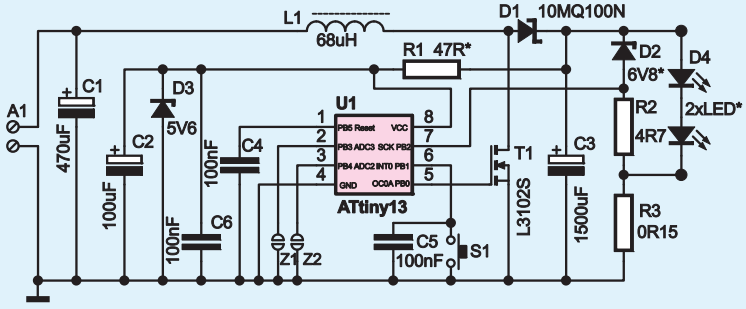
Sterownik LED mocy do latarek

Do czego służy?

Prezentowany układ jest bardzo interesującym przykładem wykorzystania popularnego, małego procesora AVR między innymi w roli... przetwornicy impulsowej. A tak w ogóle, to diody LED mocy stają się coraz popularniejsze, tańsze oraz bardziej efektywne. Taką diodę należy zasilać ze źródła prądowego. Jeżeli mamy zasilacz, to problemu nie ma. Wystarczy prosty układ źródła prądowego na tranzystorach i nie musimy przejmować się stratami mocy. Ale co zrobić, gdy chcemy zasilić takie LED-y bateriami? Przecież nie będziemy do latarki wkładać czterech lub więcej paluszków, aby zapewnić odpowiednią pracę źródła prądowego. Można wykorzystać zalecane do tego sterowniki, ale… Po pierwsze, taki sterownik jest droższy od samej diody, po drugie, musimy zaprojektować układ włącznika. Przecież nikt dzisiaj nie będzie stosował zwykłych mechanicznych przełączników. Ale byłby obciach! Teraz odchodzi się nawet od przycisków na rzecz klawiatur dotykowych. To jednak ma swoje wady. Tak więc pozostańmy przy standardowym włączniku jednoprzyciskowym. Zawsze można dodać więcej opcji, na przykład dwa rodzaje jasności, mruganie itp. No to bierzmy się do roboty! Projektujemy prosty układ włącznika na najtańszym mikrokontrolerze AVR, jeszcze jakiś tranzystorek, najlepiej MOSFET, bo w końcu płyną tam dość spore prądy. I gotowe! Czyli mamy sterownik przycisku na AVR, fabryczny sterownik LED oraz samą diodę mocy. Tylko pytanie, po co tyle tego? Mikrokontroler przez większość czasu będzie się „nudził", sterownik LED jest drogi… A gdyby tak odchudzić cały układ? Na przykład pozbyć się specjalizowanej przetwornicy i zastąpić ją przez kontroler PWM oraz przetwornik ADC mikrokontrolera. W tej roli świetnie się sprawdzi ATtiny13. Jest mały i ma wszystko, co potrzebne. Wystarczy mu tylko klucz na tranzystorze MOSFET, a reszta jak w standardowej przetwornicy Step-Up. Właśnie taki prosty układ chciałbym zaprezentować. Jest on nieco bardziej skomplikowany ze względu na zasilanie sterownika od strony wyjścia, za to można zasilać go już z dwóch akumulatorków 1,2V. Najważniejsze, że całość jest tańsza i prostsza od samego fabrycznego sterownika, który nie daje większych możliwości, jak tylko utrzymywać stały prąd na diodzie. Natomiast prezentowany układ ma trochę więcej opcji. Ma 4 tryby pracy. Dwa z nich służą do pracy z diodami 1W, a dwa pozostałe do 3W. Pierwszy tryb posiada 3 stany: wyłącz oraz dwa stopnie jasności świecenia. Drugi tryb posiada 4 stany: wyłącz/świecenie/szybkie mruganie (stroboskop)/wolne mruganie. Kolejne dwa tryby są analogiczne, tylko służą do sterowania diody 3W. Kolejną ważną cechą jest możliwość pracy z większą liczbą diod. Można zasilać 1, 2 lub nawet 3 LED-y mocy naraz, zarówno 1W, jak i 3W. I to wszystko przy zasilaniu z dwóch akumulatorków 1,2V. Wymiary też nie są jakieś wielkie. Płytka jest odrobinę mniejsza od średnicy baterii R20. Jedyne wady, to niewygórowana sprawność, wynosząca 70–85% oraz duży dławik i kondensatory ze względu na niską częstotliwość kluczowania wynoszącą najwyżej 20kHz. Tyle był w stanie „wycisnąć" ATtiny13 taktowany 4,8MHz.

Rys. 1



Jak to działa?

Sercem tego sterownika jest oczywiście mikrokontroler ATtiny13. Odpowiada on za generowanie odpowiedniego przebiegu PWM sterującego pracą przetwornicy. Kolejne zadanie to pomiar prądu jaki płynie przez diodę LED i ewentualna korekcja wypełnienia PWM. Ostatnie zadanie to obsługa przycisku i sterowanie jasnością świecenia diody, w tym także całkowite wyłączanie przetwornicy. Schemat sterownika znajduje się na **rysunku 1**. Na początek weźmy samą przetwornicę Step-Up, czyli podwyższającą napięcie. Składają się na nią elementy L1, D1, T1 oraz C3. W stanie spoczynku C3 ładuje się do napięcia zasilania, nie licząc niewielkiego spadku na diodzie Schottky'ego. Gdy klucz zostanie zamknięty (tranzystor T1 przewodzi), w cewce zaczyna gromadzić się energia, a więc płynie przez nią coraz większy prąd. W momencie otwarcia klucza cewka próbuje utrzymać stały prąd, wymuszając tym samym odwrotne napięcie na swoich zaciskach. Powoduje to dalsze ładowanie kondensatora C3, w ten sposób cewka oddaje swoją energię. Stosunek czasu przewodzenia tranzystora do czasu zatkania jest wprost proporcjonalny do ilości przekazywanej energii na wyjście przetwornicy. Ten stosunek to nic innego jak współczynnik wypełnienia PWM. Tak więc regulując wypełnienie, sterujemy pośrednio jasnością świecenia diody LED. Dla ułatwienia spójrz na **rysunek 2**. Pokazano na nim, jak wyglądają przebiegi na cewce i na wyjściu dla różnych wartości wypełnienia. Indukcyjność dławika wcale nie musi być ściśle określona, im większa, tym lepiej. Natomiast nie może być zbyt mała, gdyż prąd

narastałby bardzo szybko i w efekcie doprowadziłoby to do nasycenia rdzenia. W prezentowanym sterowniku zalecam stosowanie dławika o indukcyjności przynajmniej 47µH. Powinien mieć jak najmniejszą rezystancję uzwojenia (poniżej 0,1Ω) oraz bez problemu radzić sobie z prądami powyżej kilku amperów. Po co aż tyle, skoro diody 3W pobierają tylko 700mA? Załóżmy, że zasilamy 3 takie diody naraz. Napięcie na nich będzie się wahać w granicach 9–9,5V. Przyjmując napięcie zasilania 2V oraz uwzględniając sprawność 75%, średni prąd wejściowy, jaki będzie płynął, to prawie 4,5A. O wiele lepiej ma się sytuacja dla jednej diody o mocy 1W. Sprawność przetwornicy jest wtedy trochę niższa, co zależy głównie od zastosowanego tranzystora – o czym poniżej. W każdym razie przy zasilaniu napięciem 2V prąd wejściowy nie powinien przekraczać 800mA. Gdy będzie większy, to znaczy, że jakość zastosowanych elementów jest kiepska. Jeżeli sprawność dla kilku diod jest duża, a dla jednej niska, to wina nieodpowiedniego tranzystora. Musi to być MOSFET z serii *logic level*, na przykład potężny IRL3803. Ja zastosowałem do tego tranzystory z płyty głównej komputera L3102S oraz 76137S w prezentowanym układzie. Te MOSFET-y sprawdzą się najlepiej, gdyż większość z nich może pracować już przy 1,8–2V, co jest wymagane do poprawnego startu przetwornicy. Ale czemu do startu? I co to ma wspólnego z liczbą diod LED? Już wyjaśniam. Spójrz jeszcze raz na schemat układu. Czy już widzisz, jak zasilana jest przetwornica? Tak! Zasilana jest napięciem wyjściowym, które sama wytwarza! Przy pracy z jedną diodą to napięcie wynosi około 3,15V, przy dwóch 6,3V, a przy trzech dochodzi do 9,5V. To pozwala bez problemu sterować tranzystorem. Dlatego, gdy zasilamy tylko jedną diodę, tranzystor nie może się w pełni otworzyć ze względu na niskie napięcie. Na szczęście tranzystory uzyskane z płyty głównej przy takim napięciu na bramce mogą pracować z prądami rzędu kilkunastu amperów, co skutkuje tylko niewielkim pogorszeniem sprawności w stosunku do pracy z kilkoma diodami mocy. Wtedy napięcie zasilania przetwornicy przekracza 5V i tym można w pełni wysterować takiego MOSFET-a. Aby nie spalić mikrokontrolera, zastosowałem prosty stabilizator z diodą Zenera i elementami C2, D3 i R1. Rezystor R1 jest oznaczony gwiazdką, gdyż jego wartość zależy od liczby zasilanych diod. Dla jednej diody LED może on mieć kilkanaście omów. Dla dwóch diod wartość powinna wynosić 100Ω, a dla trzech 470Ω. Pewnie zapytasz, po co taki stabilizator, jeśli będzie pracować tylko jedna dioda mocy? Cóż, na przykład przerwie się przewód, którym ta dioda podłączona jest do przetwornicy. I co wtedy? Przetwornica pracuje jako źródło prądowe, tak więc za wszelką cenę będzie próbowała utrzymać stały prąd. A jeśli obwód został przerwany, to napięcie wyjściowe będzie się zwiększać aż do granic możliwości. Aby temu zapobiec, dołożyłem kolejną diodę Zenera D2 i rezystor R2. Jest to zabezpieczenie nadnapięciowe. Mimo to nagłe odłączenie diody LED w trakcie pracy przetwornicy może spowodować pojawienie się impulsu o dużo wyższym napięciu. Zanim procesor zareaguje i zmniejszy odpowiednio wypełnienie PWM, może ulec zniszczeniu. Dlatego właśnie należy zostawić elementy D3 i R1 na swoim miejscu. Warto wspomnieć jeszcze parę słów o stabilizacji prądu i zabezpieczeniu nadnapięciowym, gdyż układ z elementów D2, R2 i R3 nie do końca objaśnia zasadę działania na pierwszy rzut oka. Wyobraźmy sobie, że nie ma elementów D2 i R2, gdyż nie mają na nic wpływu. Od razu wszystko widać. Prąd jest stabilizowany poprzez odczyt napięcia, jakie powstaje na rezystorze R3, gdy przepływa przez niego prąd. To napięcie trafia na przetwornik ADC mikrokontrolera i dzieją się rzeczy, które są opisane w dalszej części artykułu. Efektem tych „skomplikowanych" czynności jest możliwość zmiany współczynnika wypełnienia impulsów sterujących pracą tranzystora. No dobrze, a jak działa zabezpieczenie? W trakcie normalnej pracy elementy D2, R2 nie grają żadnej roli. Dopiero gdy nie ma obciążenia na wyjściu (dioda LED jest odłączona), wzrasta napięcie wyjściowe. W końcu zaczyna przewodzić dioda Zenera. W ten sposób zostaje oszukany procesor, który „myśli", że płynie odpowiedni prąd na wyjściu i nie zwiększa dalej współczynnika PWM. Układ staje się wtedy stabilizatorem napięcia z ograniczeniem prądowym. Na schemacie jest jeszcze para najważniejszych elementów S1 oraz C5. Do czego służy S1, chyba nie muszę tłumaczyć… C5 eliminuje oczywiście drgania styków przycisku.

Zworki	Z1	Z2
Dwie jasności LED 1W	Połączenie	Połączenie
Świecenie, miganie LED 1W	Przerwa	Połączenie
Dwie jasności LED 3W	Połączenie	Przerwa
Świecenie, miganie LED 3W	Przerwa	Przerwa

Tabela 1

Program

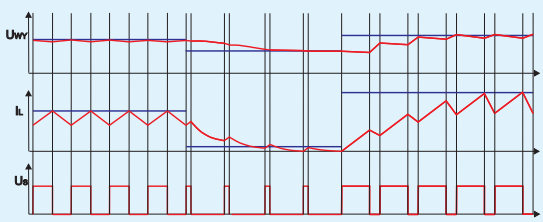
ATtiny13 ma zaledwie 1kB pamięci FLASH, tak więc program (można go ściągnąć z Elportalu) nie jest zbyt obszerny. Opiera się on o przerwania, pętla główna zawiera tylko instrukcje wprowadzające mikrokontroler w stan uśpienia IDLE. Zmniejsza to nieco pobór prądu, poprawiając całkowitą sprawność urządzenia o ułamek procenta. Zawsze coś…

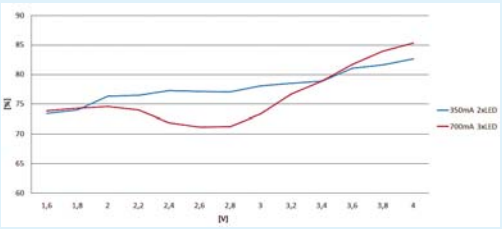
Na początku programu inicjowane są piny, Timer0 do pracy w trybie Fast PWM, przerwanie INT0 oraz częściowo przetwornik ADC. Odczytywany jest też stan pinów PB3 i PB4 i ustawiany odpowiedni tryb. Do wyboru są opcje przedstawione w **tabeli 1**. Służą temu zworki Z1 i Z2. Jak już wcześniej wspomniałem, przebieg PWM jest generowany z częstotliwością prawie 20kHz, gdyż procesor pracuje z wewnętrznym zegarem 4,8MHz. Jak łatwo policzyć 4,8MHz/256 = 18,5kHz. W egzemplarzu modelowym częstotliwość wynosi jednak około 20kHz, co jest spowodowane słabą stabilnością wewnętrznego generatora RC. Nie trzeba się tym przejmować, co najwyżej będzie słychać cichy pisk podczas pracy przetwornicy. Przetwornik ADC pracuje w trybie ciągłym i obsługiwany jest w przerwaniu. Również tam przeprowadzone są niezbędne obliczenia związane z utrzymaniem stabilnego prądu na wyjściu. Najpierw obliczana jest różnica między zmierzonym napięciem na rezystorze a ustalonym w programie. Następnie w zależności, jak wielka jest ta różnica, odpowiednio zwiększany lub zmniejszany jest współczynnik wypełnienia. Jeśli różnica jest duża, to współczynnik jest zmieniany o większą wartość. Pokazuje to **listing 1**. Taka procedura pozwala szybko reagować na zmiany napięcia wejściowego oraz zmianę zaprogramowanego prądu. Na przykład gdy przetwornica pracuje jak stroboskop, wtedy trzeba szybko zmieniać wartość prądu płynącego przez diodę LED od zera do 375mA lub 850mA i odwrotnie.

```
int8_t PWM=0; //zmienna pomocnicza do zapisania szybko-
   ści zmian współczynnika PWM
int16_t rej=CURRENT-ADCW; //odczytaj różnicę prądu za-
   programowanego od wyjściowego
if(rej>50)PWM=6;  //szybkośc zmian współczynnika PWM za-
   leżna od różnicy prądu zaprogramowanego a wyjściowego
else if(rej>20)PWM=5;
else if(rej>10)PWM=4;
else if(rej>5)PWM=3;
else if(rej>2)PWM=2;
else if(rej>0)PWM=1;
else if(rej<-50)PWM=-10;
else if(rej<-20)PWM=-7;
else if(rej<-10)PWM=-4;
else if(rej<-5)PWM=-3;
else if(rej<-2)PWM=-2;
else if(rej<0)PWM=-1;
uint8_t ocr=OCR0A; //zmienna pomocnicza do ustalenia
   współczynnika wypełnienia
ocr-=PWM; //zmień współczynnik PWM o odpowiednią wartość
//sprawdzenie czy współczynnik mieści się w przedziale
   0 - 89%, można spróbowac więcej, ale może to pogor-
   szyc sprawę
if(ocr<10)ocr=255;
if(ocr<28)ocr=28; //sprawdzenie czy ocr <28  nie można
   zejsc zbyt nisko, wszysko zależy od parametrów ele-
   mentów TLC
OCR0A=ocr; //przepisanie zmiennej do właściwego rejestru
```

Listing 1

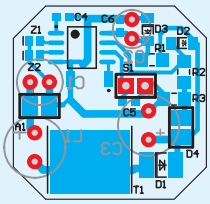
Rys. 2





Rys. 4

Rys. 3



Skąd te dziwne wartości? Otóż gdy dioda pracuje impulsowo, może być zasilana większym prądem, co wykorzystano w prezentowanym urządzeniu. Przycisk obsługiwany jest poprzez przerwanie INT0. Tam znajdują się procedury zmiany stanów pracy. Dodam, że obsługa stanów migania diodą znajduje się w przerwaniu od ADC, gdyż wymaga ciągłej kontroli. Mikrokontroler cyklicznie zwiększa wartość zmiennej *licznik* i odpowiednio do jej wartości gasi i zaświeca diodę LED. Przerwanie od przycisku jest wykonywane jednokrotnie po naciśnięciu i nadaje się tylko do obsługi pozostałych stanów, czyli świecenie ciągłe, wyłączenie. No i doszliśmy do „wyłączenia". Wywoływana jest wtedy funkcja PWR_DOWN(). Wykonuje ona czynności niezbędne do wyłączenia przetwornicy i przejścia do trybu Power down. Dodatkowo wyłączany jest także ADC, aby zmniejszyć pobór prądu procesora poniżej 1µA. Jest on z powrotem włączany w przerwaniu INT0 po wybudzeniu mikrokontrolera, ponieważ zmienna *stan* ma wtedy wartość 0. Dodatkowo jest zmieniana reakcja przerwania na zbocze. Przy przejściu do trybu Power down należy zmienić reakcję na wyzwalanie poziomem. W przeciwnym razie procesor by się nie obudził. Natomiast podczas normalnej pracy powodowałoby to ciągłe zmienianie stanów pracy, gdy przytrzymamy przycisk na dłużej niż kilkadziesiąt mikrosekund. Dlatego zaraz po wybudzeniu należy od razu zmienić na wyzwalanie zboczem opadającym.

Montaż i uruchomienie

Układ został zmontowany na jednostronnej płytce drukowanej pokazanej na **rysunku 3**. Wymiarami nie przekracza średnicy baterii R20, co odbyło się kosztem gęstszego upakowania elementów. Jednak nie sprawia to problemów przy lutowaniu. Na początek proponuję przylutować tranzystor. Szczęśliwi posiadacze stacji na gorące powietrze nie będą mieli z tym problemu. Nie panikuj, jeśli nie masz takiej stacji. Alternatywą jest… żelazko. Mocujemy je w odwrotnej pozycji i kładziemy na jego stopę płytkę z naniesioną cyną na pole drenu tranzystora. Trzeba poczekać, aż cyna będzie płynna, a następnie położyć tranzystor i po chwili całość ochłodzić. Reszta elementów może zostać przylutowana zwykłą lutownicą. Najpierw rezystory, następnie kondensatory SMD, diody, procesor, a na koniec elektrolity i dławik. Kondensatory elektrolityczne powinny być przystosowane do pracy impulsowej. W przeciwnym razie będą się grzały, a sprawność przetwornicy będzie fatalna. Również dławik musi być odpowiedniej wielkości. Ja wykorzystałem rdzeń dławika z płyty głównej PC. Z reguły na płycie są 3 takie dławiki. Po nawinięciu 40 uzwojeń jego indukcyjność powinna wynosić około 68µH. Oczywiście można kupić gotowy, najlepiej toroidalny, tylko nie żaden miniaturowy! Taki z pewnością szybko się nasyci i pewnie po kilku sekundach się spali (choć nie testowałem :). Nie bez znaczenia jest też prąd, jaki może przez niego płynąć oraz rezystancja uzwojenia, o czym wspomniałem powyżej w opisie działania. Ogólna sprawność przetwornicy zależy przede wszystkim od tranzystora, a także od jakości dławika i kondensatorów C1 i C3. Dioda Schottky'ego też ma znaczenie. Do tego dochodzi jeszcze liczba zasilanych diod, gdyż sprawność jest większa dla dwóch i trzech LED-ów. Można jednak uznać, że to wina tranzystora, o czym już wspomniałem (przypomnij sobie, jak zasilany jest mikrokontroler). **Rysunek 4** pokazuje wykres efektywności przetwornicy. Takie charakterystyki uzyskałem zasilając dwie diody 1W oraz trzy diody 3W napięciem z przedziału 1,6–4V. Oczywiście, aby mierzyć przy napięciach poniżej 2V, trzeba najpierw podać wyższe, żeby przetwornica „zaskoczyła". Pozostaje jeszcze kwestia zaprogramowania mikrokontrolera. Aby ograniczyć wymiary płytki, nie zastosowałem złącza do programowania. Zanim procesor zostanie przylutowany, można go połączyć przewodami programatora na pająka. W pierwszej kolejności ustawiamy bity konfiguracyjne. **CKSEL1..0** na **01** oraz **CKDIV8** na **1** (wewnętrznie 4,8MHz). Dodatkowo aby mikrokontroler poprawnie startował, należy ustawić bity **BODLEVEL** na **10** (1,8V). Teraz spokojnie wrzucamy do niego program i lutujemy do płytki sterownika. Na koniec dodam jeszcze, aby pamiętać o odpowiednim chłodzeniu diod mocy oraz nie używać akumulatorów NiMH, tylko NiCd, jeśli pracuje więcej niż jedna dioda 3W lub trzy diody 1W. **Fotografia tytułowa** przedstawia jedną z moich latarek, trzy diody, każda o mocy 3W, zasilanie – dwie baterie (akumulatory) R20. Niech nikogo nie przeraża ten potężny dławik, po prostu tylko taki miałem pod ręką. Ten z płyty głównej okazał się niewystarczający, choć dostarczał pełnej mocy, to strasznie się grzał (wina cienkiego uzwojenia). W każdym razie widać, co potrafi ten mały układzik. Przy tym prawie się nie grzeje, jedynie dioda Schottky'ego jest ciepła.

Wykaz elementów

Rezystory
R1 ..... 100Ω *
R2 ..... 4,7Ω
R3 ..... 0,15Ω

Kondensatory
C1 ..... 470µF impulsowy
C2 ..... 100µF
C3 ..... 1500µF impulsowy
C4-C6 ..... 100nF

Półprzewodniki
D1 ..... 10MQ100N Schottky'ego
D2 ..... 6,8V Zenera *
D3 ..... 5,6V Zenera
T1 ..... L3102S, logic level *
U1 ..... AVR ATtiny13V

Pozostałe
L1 ..... 68µH dławik *
S1 ..... dowolny przycisk
*patrz opis

Płytka drukowana jest dostępna w sieci handlowej AVT ako kit szkolny AVT-2929.

Możliwości zmian

Jak wspomniałem wyżej, przetwornica może pracować przy napięciach o wiele niższych niż potrzebne do włączenia. To może być groźne dla akumulatorów. Na szczęście jeśli zasila się kilka diod, to widać spadek mocy wyjściowej przy napięciu poniżej 1,3–1,6V. Jednak gdy mamy podłączoną tylko jedną diodę, to przetwornica może wejść „na pełne obroty", czyli uzyskać maksymalny współczynnik wypełnienia dopiero przy zasilaniu poniżej 1V. To na pewno nie wyjdzie na zdrowie akumulatorkom. W takiej sytuacji można pokusić się o modyfikację kodu. Zamiast wyboru trybu pracy, wykorzystać jedno z wejść do monitorowania napięcia zasilania i wyłączać przetwornicą po przekroczeniu dopuszczalnego progu. Jeżeli wyrzucić z programu obsługę tych trybów, to powinno wystarczyć miejsca w szczupłej pamięci FLASH mikrokontrolera na dodatkowe procedury obsługi ADC. To tylko jedna z wielu możliwości zmian.

Arkadiusz Hudzikowski
a-r-o@o2.pl