FlowCode i E-blocks (3)

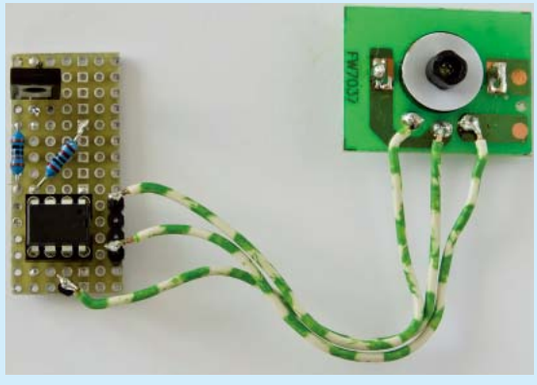
Sterownik lampki campingowej LED

Nowoczesne języki programowania ewoluują w stronę, która umożliwi łatwe napisanie programu nie tylko elicie programistów, ale dosłownie każdemu. Pozwolą na to kompilatory graficzne, których aktualnie używają z powodzeniem zarówno profesjonaliści z różnych dziedzin, jak i… dzieci programujące klocki Lego Mindstorm. Przykładem środowiska programistycznego przeznaczonego równie dobrze dla profesjonalistów, jak i amatorów, służącego do programowania graficznego różnych rodzin mikrokontrolerów, jest produkt brytyjskiej firmy Matrix Multimedia – FlowCode. W tej części kursu pokażemy praktyczny sposób użycia FlowCode do implementacji programu sterującego pracą lampki LED o regulowanym natężeniu świecenia.

Przed wakacjami kupiłem lampę kampingową. To nieskomplikowane, zasilane z baterii urządzenie jest bardzo przydatne na polu namiotowym. Kiedyś takie lampy były zasilane z akumulatora samochodowego, a jako element świecący używano żarówki. Bardziej zaawansowane konstrukcje miały wbudowaną przetwornicę zasilające małą świetlówkę. Świeciły o wiele jaśniej przy małym poborze prądu z akumulatora, lub baterii. Dzisiaj w roli elementu świecącego powszechnie wykorzystuje się białe diody świecące.

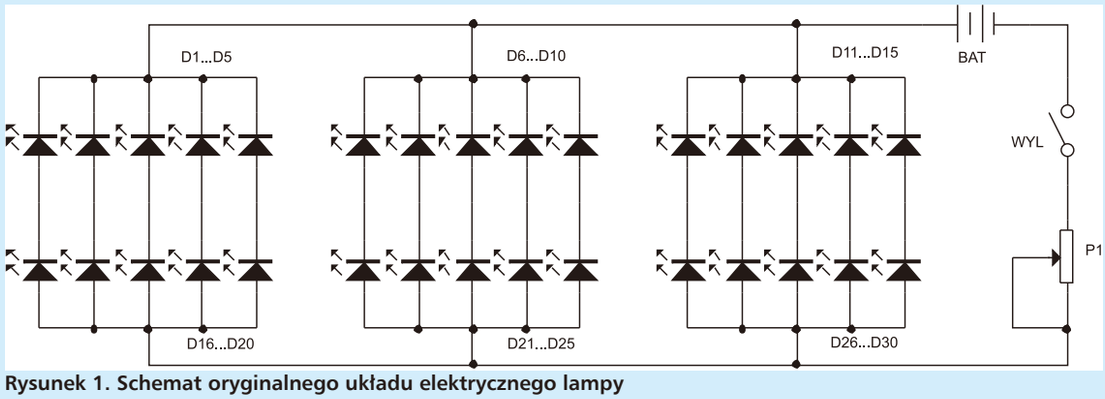
Kupiłem lampę z 30 niewielkimi diodami LED i obrotowym regulatorem jasności ich świecenia. Po zakupie 3 baterii R20 i pobieżnym sprawdzeniu działania, wybrałem się na camping. Już na początku eksploatacji zaniepokoiło mnie dość szybkie zużycie baterii, mimo iż z informacji na pudełku wynikało, że czas świecenia miał być bardzo długi. Potem część z diod LED wyraźnie zaczęła świecić słabiej. Zacząłem podejrzewać, że urządzenie ma jakąś wadę fabryczną. Po powrocie i rozebraniu lampy byłem równie zdziwiony, co zaskoczony. Przy zakupie wyobrażałem sobie regulację jasności za pomocą prostego układu elektronicznego z modulacją PWM. Przy masowej produkcji taki sterownik pewnie kosztuje przysłowiowe grosze. Jednak wewnątrz obudowy lampy niczego takiego nie znalazłem.

Schemat ideowy oryginalnej lampy narysowany przeze mnie pokazano na **rysunku 1**. 30 białych diod LED zostało podzielonych na 3 grupy po 10. Każda z grup zawiera po 5 połączonych równolegle grup złożonych się z 2 diod połączonych szeregowo. Jako element regulacyjny zastosowano… włączony szeregowo z plusem zasilania zwykły potencjometr liniowy małej mocy (!) i rezystancji 1 kΩ, zintegrowany z wyłącznikiem. Nieco zbulwersowany zacząłem analizować, dlaczego taki układ po pierwsze, wyczerpał mi tak szybko baterie, a po drugie, najprawdopodobniej uszkodził część diod LED.



Diodę LED można scharakteryzować kilkoma podstawowymi parametrami:

– **Znamionowy prąd przewodzenia** (*DC Forward Current*) – w uproszczeniu to natężenie prądu, przy którym dioda LED świeci z maksymalną światłością w temperaturze normalnej. Dla diod małej mocy najczęściej ma on natężenie rzędu 20...30 mA. Poza prądem znamionowym podaje się impulsowy prąd przewodzenia (*Peak Forward Current*). Jest to maksymalne natężenie prądu przy sterowaniu PWM o określonym współczynniku wypełnienia, najczęściej 10%. Na przykład, dla diody LED o prądzie przewodzenia 20 mA, prąd impulsowy może wynosić 150 mA.



Rysunek 1. Schemat oryginalnego układu elektrycznego lampy

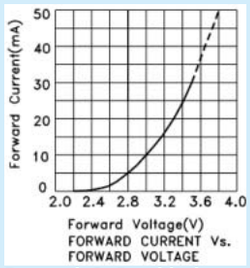
– **Znamionowe napięcie przewodzenia** jest podawane dla znamionowego prądu przewodzenia i występuje na zaciskach diody spolaryzowanej w kierunku przewodzenia. Jest ono zależne od typu diody: najczęściej stosunkowo małe dla diod świecących w podczerwieni i wysokie dla diod białych. Przykład charakterystyki zależności prądu diody zależnie od napięcia przyłożonego do diody w kierunku przewodzenia pokazano na **rysunku 2**.
– **Dopuszczalne napięcie wsteczne** to maksymalne napięcie przyłożone do diody w kierunku zaporowym.

Kluczowa do poprawnego zasilania diody LED jest znajomość charakterystyki pokazanej na rys. 2. Przyłożenie zbyt wysokiego napięcia ze źródła o małej rezystancji wewnętrznej spowoduje gwałtowny wzrost prądu przewodzenia i w konsekwencji zniszczenie diody. Ponieważ charakterystyka jest dość stroma, to mały przyrost napięcia powoduje duże przyrosty prądu przewodzenia. Dlatego stosuje się zasilanie napięciem wyższym niż znamionowe a szeregowo ze źródłem włącza się rezystor, jak na **rysunku 3**.

Wartość rezystora wylicza się znając napięcie przewodzenia i prąd znamionowy. Załóżmy, że mamy diodę o charakterystyce z rys. 2 zasilaną napięciem +4,5 V. Przy 20 mA spadek na niej wynosi 3,3 V, więc spadek na rezystorze powinien wynosić 4,5 V-3,3 V=1,2 V, a więc rezystancję opornika można wyliczyć z prawa Ohma jako R=1,2 V/0,02 A=60 Ω. Jeżeli popatrzymy na schemat lampy z rys. 1, to zauważymy podobny obwód do analizowanego. Rolę rezystora szeregowego spełnia potencjometr. Jednak jest pewna zasadnicza różnica – w skrajnym położeniu rezystancja potencjometru wynosi 0 i wszystkie diody są zasilane wprost z baterii o bardzo niskiej rezystancji wewnętrznej. Przed rozebraniem lampy zmierzyłem prąd pobierany przez układ. Kiedy potencjometr był zwarty, diody pobierały ponad 1,5 A. Przyjmując pewne założenie dla 30 diod można obliczyć, że każda z nich pobiera ok. 50 mA. Wydaje mi się, że przynajmniej dla części z nich to zdecydowanie zbyt duży prąd ciągły i dlatego uległy częściowemu uszkodzeniu. Tak duży prąd tłumaczy też bardzo szybkie wyczerpywanie się baterii.

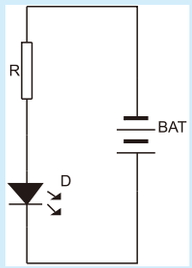
Kolejnym kontrowersyjnym rozwiązaniem jest równoległe połączenie wielu diod. Ich charakterystyki na pewno nie są idealnie powtarzalne i powinno się zastosować rezystory wyrównujące prądy w każdej z gałęzi. Projektant albo nie przyłożył się do pracy, albo z pełną świadomością zaprojektował układ w taki sposób, aby był jak najtańszy. Nawet w tym bardzo prostym układzie można było przynajmniej zapobiec uszkodzeniom włączając po jednym rezystorze wyrównującym w każdej z 3 sekcji diod LED.

Ponieważ lampa miała z założenia służyć dłużej niż jeden sezon, postanowiłem układ poprawić. Najlepszym rozwiązaniem było zaprojektowanie i wykonanie sterownika sterującego jasnością za pomocą sygnału PWM zmieniającego wypełnienie w zależności od położenia potencjometru. Zdecydowałem się na użycie mikrokontrolera, chociaż oczywiście można było zastosować np. „nieśmiertelny 555”. Do realizacji sterownika jest potrzebny mikrokontroler z przetwornikiem A/C i możliwością generowania przebiegu PWM. Miał to być układ zbudowany z jak najtańszych, ale dobrych elementów.



Rysunek 2. Przykładowa charakterystyka białej diody LED

Zdecydowałem się na użycie PIC12F675. Jest tani i ma przetwornik A/C. Nie ma sprzętowego modułu PWM, ale to nie jest problemem, bo jak pokażę dalej – przebieg PWM można równie skutecznie wygenerować innymi metodami. Jak już wspomniałem, układ ma generować przebieg PWM o wypełnieniu zależnym od położenia potencjometru o rezystancji 1 kΩ. Schemat sterownika pokazano na **rysunku 4**. Napięcie zasilania z suwaka potencjometru R3 jest podawane na wejście AN0 przetwornika A/C. Sygnał PWM z wyprowadzenia GPIO2 steruje bazą tranzystora BD139 przez rezystor R1. Rezystor R2 jest potrzebny w obwodzie zerowania mikrokontrolera.

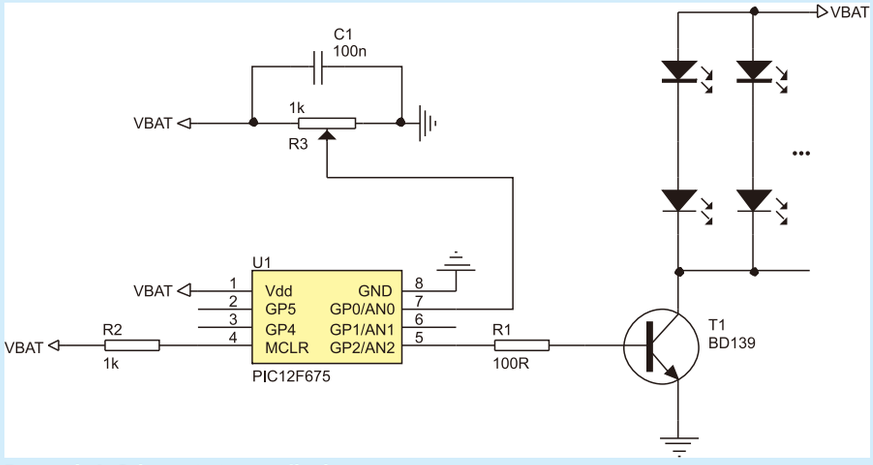


Rysunek 3. Typowy układ zasilania diody

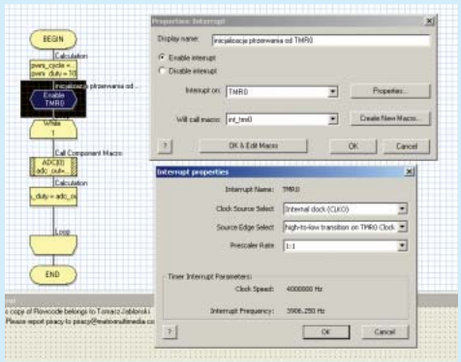
Zasada działania jest prosta: przetwornik A/C mierzy napięcia na wejściu AN0 (nóżka 7). Zależnie od poziomu tego napięcia ustawianego za pomocą potencjometru R3 zmienia się współczynnik wypełnienia przebiegu PWM generowanego na wyjściu GP2. Przyjąłem, że przy napięciu wejściowym 0 V wyjście GP2 jest wyzerowane (wypełnienie 0%), natomiast przy napięcia równym VBAT wypełnienie jest równe 95%.

Program sterujący napisałem, a w zasadzie narysowałem, we Flowcode V4. Dzięki temu jego utworzenie zajęło mi niewiele czasu. Mimo, że program nie jest skomplikowany, to jednak napisanie go w C wymaga konfigurowania układów peryferyjnych, a to zawsze jest pracochłonne.

Program zaczyna się od inicjalizowania licznika TMR0. Mikrokontroler PIC12F675 nie ma sprzętowego generatora przebiegu PWM. Do sterowania diodami LED nie potrzebujemy wysokiej częstotliwości ani sygnału o dużej rozdzielczości. Dlatego do generowania sygnału PWM można użyć 8-bitowego licznika TMR0 i mechanizmu przerwań. Inicjalizacja układu przerwań jest bardzo łatwa. Z menu ikon wybieramy element oznaczony jako INT i po przeciągnięciu na planszę otwieramy okno *Properties* (**rysunek 5**). W polu *Display name* możemy wpisać nazwę bloku. Zaznaczamy *Enable interrupt* i w oknie *Interrupt on* wybieramy TMR0. Po kliknięciu na belkę *Properties* otwiera się okno *Interrupt properties* umożliwiające skonfigurowanie licznika. Wybieramy zliczanie impulsów sygnału zegara wewnętrznego (*Internal*



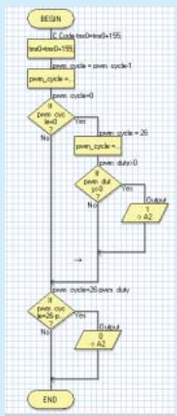
Rysunek 4. Schemat sterownika lampy



Rysunek 5. Konfigurowanie przerwania od TMR0

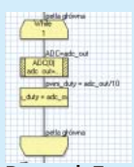
Clock) i współczynnik podziału preskalera 1:1.

Jeżeli w oknie *View -> Project Options* wpiszemy rzeczywistą częstotliwość taktowania mikrokontrolera, to w okienku zobaczymy wyliczoną częstotliwość zgłaszania przerwania po przepełnieniu się licznika liczącego od 0 do 256 (bez programowej ingerencji w jego zawartość).



Rysunek 6. Makro obsługi przerwania od przepełnienia TMR0

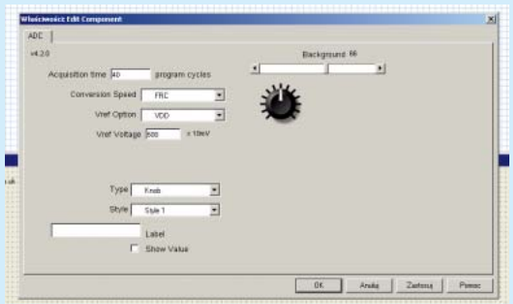
Po każdym zgłoszeniu przerwania będzie wywoływane makro o nazwie z okna *Will call macro*. Po skonfigurowaniu przerwania możemy jeszcze zainicjować nową planszę z makrem jego obsługi klikając na belkę *Create New Macro* i tam nadać mu nazwę – ja wpisałem *int_tmr0*. To makro będzie odpowiedzialne z generowanie przebiegu PWM. Dalej w pierwszej kolejności zajmiemy się częstotliwością zgłaszania przerwania. Mikrokontroler jest taktowany wewnętrznym generatorem RC o częstotliwości 4 MHz. Licznik TMR0 zlicza impulsy o częstotliwości Fclk/4=1 MHz. Aby ustalić z jaką częstotliwością będą zgłaszane przerwania musimy określić częstotliwość i rozdzielczość przebiegu PWM. Do sterowania diodami LED wystarczy tak dobrać częstotliwość, aby nie było widoczne migotanie. W praktyce wystarcza 100 Hz.



Rysunek 7. Pętla główna programu

Rozdzielczość przebiegu, to najmniejsza zmiana czasu trwania poziomu wysokiego sygnału PWM. Przyjmijmy na początek, że częstotliwość sygnału PWM wynosi 1 kHz. Żeby uzyskać zmianę czasu trwania stanu wysokiego w 9 krokach trzeba odliczać czas z dokładnością 0,1 ms. Jeżeli przerwanie będzie zgłaszane co 0,1 ms, to aby uzyskać przebieg o częstotliwości 1 kHz trzeba ustawić wyjście sygnału PWM z mikrokontrolera, odliczyć 5 przerwań co 0,1 ms, wyzerować je i odliczyć kolejnych 5 przerwań co 0,1 m. Cykliczne powtarzanie tej sekwencji da przebieg o częstotliwości 1 kHz (okres 10×0,1 ms=1 ms) i wypełnieniu 50%. Zmiana wypełnienia jest bardzo prosta: odliczamy 4 cykle poziomu wysokiego i 6 cykli poziomu niskiego, co daje 40% wypełnienia. W sterowniku tak właśnie będzie generowany przebieg PWM.

Aby licznik przepełniał się co 0,1 ms, trzeba odliczyć 100 impulsów o częstotliwości 1 MHz, więc po każdym zgłoszeniu przerwania zapisujemy do niego 155, ponieważ licznik zlicza w przód. Powoduje to, że po odliczeniu 100 impulsów osiąga wartość 255 i się przepełnia.



Rysunek 8. Okno konfigurowania przetwornika A/C

Dla potrzeb procedury generowania PWM zdefiniowałem 2 zmienne: pwm_cycle i pwm_duty. Pierwsza z nich służy do odmierzania okresu przebiegu. Czas jego trwania ustaliłem na 26 przerwań, czyli na 2,6 ms. Daje to częstotliwość ok. 385 Hz. Ponieważ odliczamy 26 przerwań, to wypełnienie przebiegu można zmieniać w 25 krokach. Ten wybór był podyktowany 8-bitową rozdzielczością pomiaru przetwornika A/C. Wartość zmiennej pwm_duty przemnożona przez 0,1 ms określa czas trwania poziomu wysokiego.

Rysunek 9. Okno wyboru wejścia przetwornika ADC

Na **rysunku 6** zamieszczono makro obsługi przerwania generującego przebieg PWM na linii GP2 (RA2). 8-bitowa wartość odczytana z przetwornika jest dzielona przez 10 i zapisywana do zmiennej pwm_duty. Zmiana współczynnika wypełnienia odbywa się przez zapisywanie zmiennej pwm_duty w pętli głównej programu (**rysunek 7**). W ten sposób kręcąc oska potencjometru zmieniamy napięcie od 0 V do VBAT (4,5 V), wartość na wyjściu przetwornika zmienia się w zakresie 0…255, a zmienna pwm_duty 0…25. Pracujące w tle programu głównego makro obsługi przerwania zmienia współczynnik wypełnienia PWM i w ten sposób odbywa się regulacja jasności świecenia diod LED.

Moduł przetwornika jest konfigurowany w oknie *Ext Properties* pokazanym na **rysunku 8**. Ustala się tam czas akwizycji (*acquistion time*), typ i częstotliwość zegara taktującego moduł A/C (*conversion speed*) oraz napięcie referencyjne. Wejście przetwornika jest wybierane w oknie *Connections* pokazanym na **rysunku 9**.

Podsumowanie

Po zamontowaniu sterownika wykonałem kilka pomiarów. Okazało się, że lampa świeci wystarczająco jasno już przy wypełnieniu ok. 50…60%. Pobór prądu wynosi wówczas ok. 400 mA. Przy wypełnieniu 95% wzrasta do 700 mA. Po zatrzymaniu PWM i ustawieniu linii GP2, pobór prądu wzrastał do ok. 870 mA. Jest on ograniczany rezystancją emiter – kolektor tranzystora BD139. Daje to ok. 27 mA na jedną diodę i można przyjąć, że jest to poziom bezpieczny. Można zastosować tranzystor unipolarny przystosowany do pracy z niskimi napięciami zasilania i mający małą rezystancją źródło – dren w czasie przewodzenia. Wtedy trzeba dobrać w układzie rezystor szeregowy ograniczający prąd diod, gdy np. na skutek awarii sterownik przestał generować przebieg PWM i ustawił wyjście GP2.

Zastosowany sterownik doskonale spełnia swoje zadanie. Jednak aby układ do końca spełniał moje oczekiwania, trzeba by było zrezygnować z diod LED włączonych równolegle i zastosować kilka diod większej mocy od znanego producenta. Wtedy można na podstawie danych katalogowych dobrać prąd każdej z nich i umożliwić optymalne warunki pracy.

Tomasz Jabłoński, EP

Redakcja Elektroniki Praktycznej dziękuje firmie TME z Łodzi, dystrybutorowi firmy Matrix Multimedia, za udostępnienie zestawu E-blocks oraz mikrokontrolerów używanych w kursie programowania FlowCode.