Rowerowy wyświetlacz widmowy

Do czego to służy?

Kto nie chciałby urozmaicić wyglądu swojego roweru o efektywny wyświetlacz na kole? Chyba każdy. Taki efektowny gadżet przyciąga oko, bawi i sprawia, że nasz bicykl jest wyjątkowy.

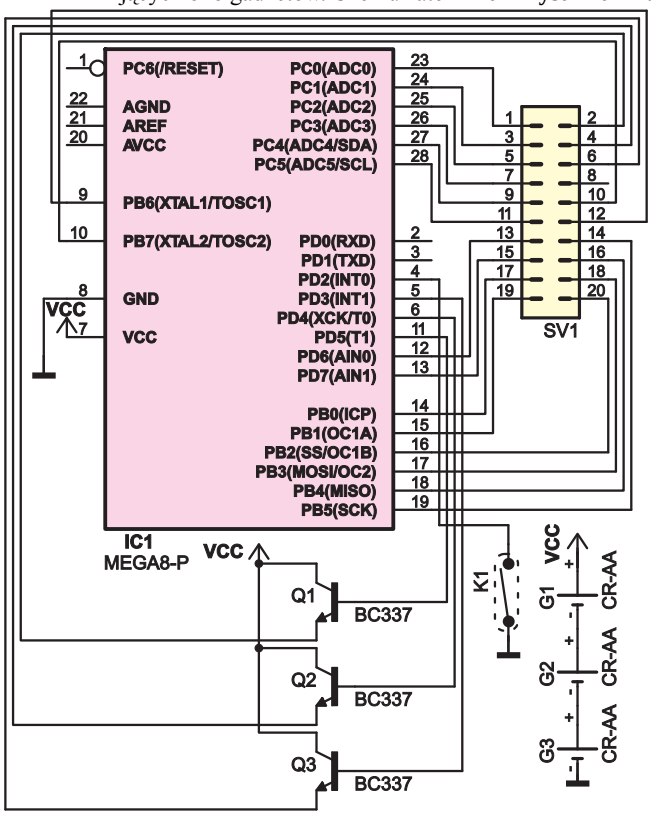
Zainspirowani artykułem *Wyświetlacz diodowy na kole rowerowym* z EdW 9/2008 i zachęceni do pracy nad podobnym projektem w ramach zajęć na uczelni postanowiliśmy zbudować podobny wyświetlacz. Naszym celem było zachęcenie dzieci i młodzieży do licznych wypraw rowerowych. Jak to osiągnąć? Czyniąc z bicykla niepowtarzalny pojazd, przykuwający uwagę przechodniów. W dzisiejszych czasach młodzież przywiązuje dużą wagę do efektownych, przyciągających oko gadżetów. Czemu zatem nie rozświetlić roweru kolorowym, wyjątkowym napisem lub obrazem?

Oto proste urządzenie, wzorowane na układzie z EdW 9/2008, zawierające znacznie większą liczbę diod i oparte na mikrokontrolerze ATmega8L. Stworzyliśmy też interfejs na komputer PC, który umożliwia przetwarzanie obrazka na formę zrozumiałą przez mikrokontroler. Nasz układ charakteryzuje się: prostotą wykonania, małym kosztem eksploatacji, prostotą programowania, energooszczędnością, prostotą montażu.

Jak to działa?

Cały układ składa się z czterech części: modułu sterownika, modułu listwy z diodami, programu napisanego w języku C oraz interfejsu użytkownika na komputer PC. Na **rysunku 1** zaprezentowany jest schemat ideowy sterownika, a na **rysunku 2** schemat ideowy listwy z diodami. Sercem układu jest mikrokontroler ATmega8L. Zdecydowaliśmy się użyć wersji przewlekanej zamiast SMD, gdyż jest ona dużo prostsza w lutowaniu. Ponadto jest zdecydowanie wygodniejsza przy testowaniu układu na uniwersalnej płytce stykowej w początkowych fazach projektu. Duża liczba programowalnych wyjść (23) dała nam możliwość sterowania szesnastoma diodami RGB, a stosunkowo duża pamięć programu FLASH (8KB) oraz danych EEPROM (512B) pozwoliła zrealizować projekt bez problemu. Jako zasilanie wykorzystaliśmy trzy popularne baterie LR6 AA, które zapewniły zasilanie 4,5V. Wybraliśmy zatem wersję z literką L, gdyż pracuje ona w niższym zakresie napięć (2,75– 5V) niż zwykła ATmega8 (4,55–5V).

Rys. 1



Postanowiliśmy ograniczyć się do siedmiu odcieni uzyskiwanych za pomocą łączenia barw diod RGB: czerwonego, zielonego, niebieskiego, żółtego, różowego, seledynowego, białego. Aby to osiągnąć, posłużyliśmy się trzeba tranzystorami średniej mocy NPN BC337 (0,8A). Przestrzegam przed zastosowaniem BC548 i odpowiedników, gdyż przy tak dużej liczbie diod nieuchronnie dojdzie do spalenia tranzystora i w efekcie nie będą świecić diody koloru, za który odpowiadał spalony tranzystor. Taki efekt możecie zobaczyć na **fotografii 1**, na której widać, że brakuje koloru zielonego.

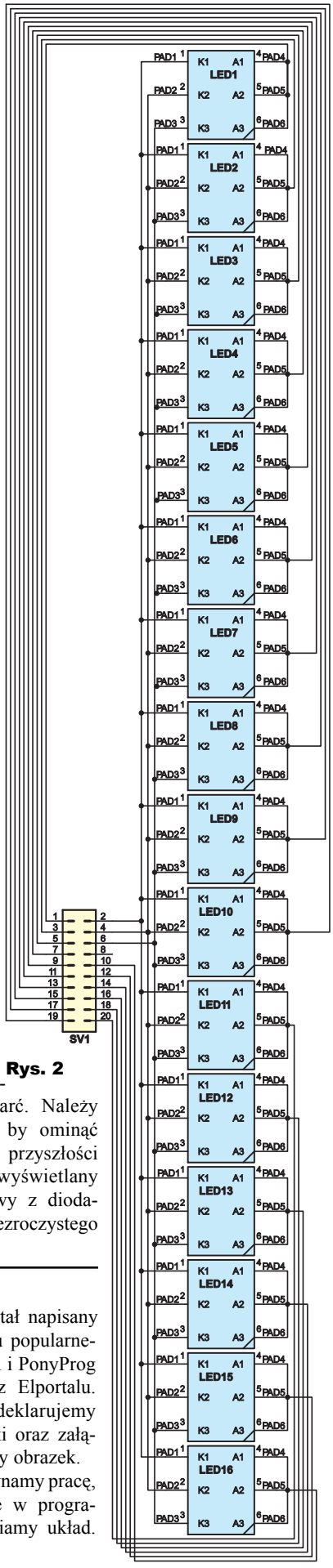
Istotny jest właściwy dobór diod. Przy szesnastu diodach każdego z kolorów podstawowych (czerwony, zielony, niebieski) jest możliwe wykorzystanie zwykłych diod, jednak my pokusiliśmy się o zastosowanie diod RGB SMD w obudowie PLCC6. Obudowa jest na tyle duża, że można ją lutować w warunkach domowych, a jednocześnie nie zajmuje wiele miejsca na płytce. Strzałem w dziesiątkę okazały się diody firmy Itswell, dostępne na www.maritex.com.pl. W praktyce świecą bardzo jasno i dają wspaniały efekt na kole. Jako ciekawostkę dodam, że diody mają tak dużą sprawność, że zaczynają lekko świecić po podłączeniu do nich omomierza, a nawet podczas lutowania (co zresztą pokazuje stan uziemienia lutownicy).

Najbardziej newralgicznym momentem doboru elementów okazał się wybór kontaktronu. Nasz magnes umieściliśmy na ramie roweru jak na **fotografii 2**. Kontaktron musiał być zatem wystarczająco czuły, by wykryć przejście obok magnesu, a jednocześnie na tyle wytrzymały, by się nie zewrzeć na stałe pod wpływem dużego prądu. Na szczęście nie okazało się to trudnym zadaniem i trafnie postawiliśmy na dość duży kontaktron o długości 3,5cm.

Ponieważ płytka sterownika w obudowie jest oddzielona od listwy z diodami, należa-

ło je połączyć. Do tego celu wykorzystaliśmy popularną taśmę do dyskietek przeciętą na pół oraz dwa zestawy 20-pinowych złączy w rastrze 2,54mm. Dla uzupełnienia dodaliśmy prosty przełącznik dwustanowy, odłączający zasilanie od układu w czasie dłuższego postoju.

Z uwagi na zastosowane zasilanie, które wymogło użycie obudowy o głębokości co najmniej 3cm, zdecydowaliśmy się wykorzystać powszechnie dostępną obudowę Z-79. W celu stabilnego i bezpiecznego przymocowania płytki drukowanej układu sterującego do obudowy, powiększyliśmy płytkę, aby umieścić na niej otwory mocujące. Dzięki temu możliwe jest solidne przymocowanie płytki do obudowy, co zabezpieczy część sterującą przed groźnymi wstrząsami. Płytkę można zabezpieczyć roztworem kalafonii w spirytusie albo dedykowanym środkiem chemicznym ze sklepu dla elektroników. Należy jednak wykonać to po uprzednim upewnieniu się, że układ na pewno działa, a zastosowany materiał izolacyjny nie spowoduje niebezpiecznych zwarć. Należy jednocześnie pamiętać, by ominąć mikrokontroler, aby w przyszłości można było zmienić wyświetlany obrazek. Obudowę listwy z diodami wykonaliśmy z przezroczystego pudełka po pędzlu.

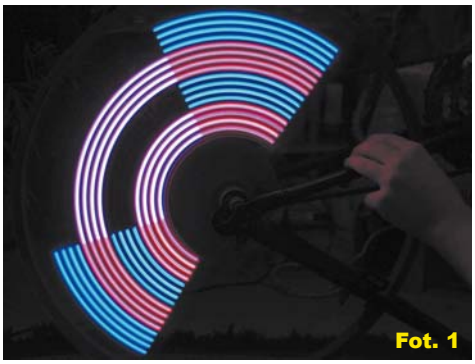


Rys. 2

Program

Program sterownika został napisany w języku C przy użyciu popularnego kompilatora WinAVR i PonyProg i można go ściągnąć z Elportalu. Najpierw oczywiście deklarujemy potrzebne nam biblioteki oraz załączamy nasz przetworzony obrazek.

W głównej pętli zaczynamy pracę, zerując wszystkie użyte w programie zmienne oraz usypiamy układ.



Fot. 1

Fot. 2

Następnie określamy, jaki port będzie sterował diodami. Program zaświeca najpierw zewnętrznych osiem diod, następnie wewnętrznych sześć, a na końcu dwie pozostałe w środku. Następnie odblokowujemy przerwania, ustalamy opóźnienie ok. 0,5 sekundy, które ma wyeliminować drgania kontaktronu. Odblokowujemy przerwanie zewnętrzne – sygnał z kontaktronu – i ustalamy je na narastające zbocze.

Zmienna *blok* ma nam pomóc w określaniu, czy łapiemy przerwania z zewnątrz, czy też nie. Pierwszy przypadek to sytuacja, kiedy nie łapiemy przerwań, a układ śpi. Jeśli coś go obudziło, ma to zapisać w rejestrze. Drugi przypadek to obudzenie układu przed chwilą – nic się nie dzieje. Ostatnia sytuacja – układ został trwale obudzony i zaczyna dokonywać operacji logicznych na portach, by wyświetlić kolory – najpierw zielony, potem niebieski, a na końcu czerwony.

Druga funkcja obsługuje przerwanie z kontaktronu. Na czas tej operacji blokujemy przerwania i w zależności od stanu dokonujemy następujących operacji:

- Jeśli układ spał, to go budzimy, zerujemy timer1 i ustawiamy prescaler timera1.
- Jeśli układ był przed chwilą obudzony, to budzimy go na dłużej i stopujemy timery. Na podstawie stanu naszego licznika dokonujemy obliczenia prędkości obrotu koła i dzielimy je na fragmenty, a więc określamy czas świecenia diod. Jednocześnie, bez względu na poprzednią pozycję obrazu, każemy mikroprocesorowi rysować go zawsze od początku.

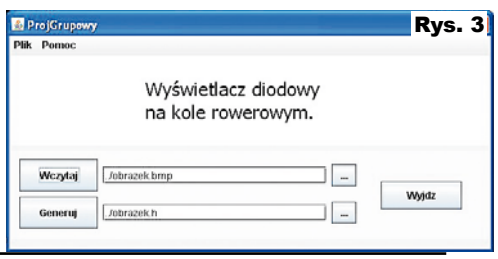
Na końcu włączamy obsługę przerwań. Gdyby doszło do przepełnienia timera (co nie powinno się zdarzyć), urządzenie ma się zresetować. Podobnie ma się zachować w przypadku, kiedy staniemy i koło przestanie się obracać (nie będzie docierał sygnał z kontaktronu).

Ostatnia funkcja określa sposób przechodzenia pomiędzy kolejnymi pikselami obrazu. Operacja przejścia ma miejsce tylko wówczas, gdy układ jest obudzony i polega na sczytywaniu wartości z bitów z pamięci programu (czyli naszego pliku „czacha.h").

Interfejs użytkownika

Za pomocą środowiska NetBeans napisaliśmy prosty program w języku Java (**rysunek 3**). NetBeans jest narzędziem darmowym, bardzo dopracowanym i przyjaznym w użyciu. Ma bogatą funkcjonalność, pozwalającą tworzyć kod efektywnie i bezbłędnie. Środowisko to można rozszerzać wtyczkami, dzięki temu można je dobrze przystosować do własnych potrzeb. Jedyną wadą Javy są niekiedy problemy z jej uruchomieniem, związane ze ścieżką i wersją programu.

Najpierw należy mieć oczywiście zainstalowaną wirtualną maszynę Javy – JRE (niezależny od platformy system uruchomieniowy) dostępną na oficjalnej stronie Java SE Downloads. Po zainstalowaniu JRE możemy uruchamiać program, ale nie damy rady go skompilować, w tym celu należy ściągnąć całe JDK dostępne na wspomnianej stronie.



Rys. 3

Aby skompilować program, trzeba mu określić ścieżkę położenia pliku javac.exe – w pliku kompiluj.bat domyślnie jest ustawiona ścieżka:

cd engine
„C:\Program Files\Java\jdk1.6.0_14\bin”\
javac Main.java
cd ..

Należy ją ustawić na właściwą dla swojego umiejscowienia pliku javac.exe.

Program uruchamiamy plikiem ProjGrupowy.bat.

W celu stworzenia pliku nagłówkowego dołączanego do programu mikrokontrolera należy:

- stworzyć obrazek o wymiarach 80x16 pikseli (do tego może posłużyć nawet Paint) i zapisać go w formacie bezstratnym (polecany bmp, ale może być każdy obsługiwany przez standardowe biblioteki Javy), przykładem może być **rysunek 6** (należy pamiętać o czarnym tle i ograniczonej palecie kolorów),
- wczytać go w aplikacji w Javie,
- przetworzone wartości zapisać w pliku nagłówkowym (domyślnie obrazek.h),
- wczytać projekt z programem mikrokontrolera i załączyć plik z przetworzonym obrazkiem.

Montaż i uruchomienie

Na **rysunkach 4** i **5** zamieszczone są schematy montażowe płytek drukowanych. Montaż płytek jest klasyczny.

Doświadczenie pokazało, że obudowa Z-79 wchodzi między szprychy koła o promieniu 14” (**fotografie 3** i **4**) i klinuje się tam na tyle mocno, że nie były potrzebne dodatkowe mocowania. Czytelników chcielibyśmy jednak uprzedzić, iż montaż tej obudowy nie jest łatwy i bez odrobiny wprawy może sprawiać trudności. Warto zatem poszukać jakiegoś innego rozwiązania, nawet korzystając z przedmiotów codziennego użytku, tak jak to zrobiliśmy z obudową na listwę z diodami.

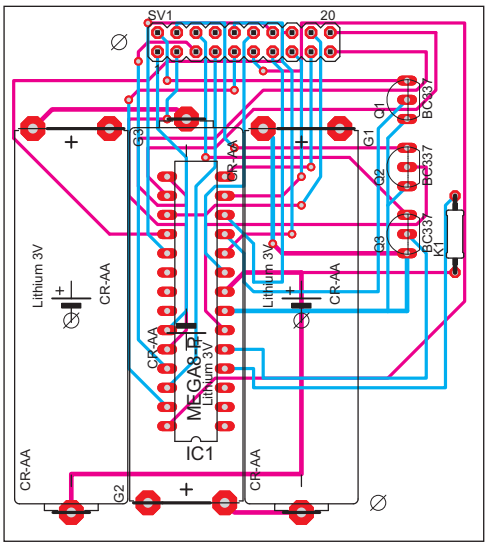
W obu obudowach zrobiliśmy podłużne szczeliny, umożliwiające użycie taśmy. Ponadto w obudowie listewki zrobiliśmy cztery malutkie otwory, które posłużyły do przymocowania obudowy do szprych (**fotografie 5** i **6**). W naszym rozwiązaniu płytka z listwami jest przyklejona do jednej wewnętrznej strony obudowy zwykłą taśmą, ale aby ją zabezpieczyć przed ewentualnymi wstrząsami i ocieraniem się o obudowę, włożyliśmy w miejsce styku cienką gąbkę służącą powszechnie do zabezpieczania przewożonych pakunków. Efekt naszych prac możecie podziwiać na fotografii tytułowej.

Wykaz elementów

K1	kontaktron
LED1-LED16	dioda RGB PLCC6
Q1-Q3	BC337
IC1	ATmega8L
SW1*	przełącznik
Obudowa Z-79	
2×złącze 20-pinowe w rastrze 2,54mm	
Podstawka precyzyjna DIL28	
*elementy opcjonalne	

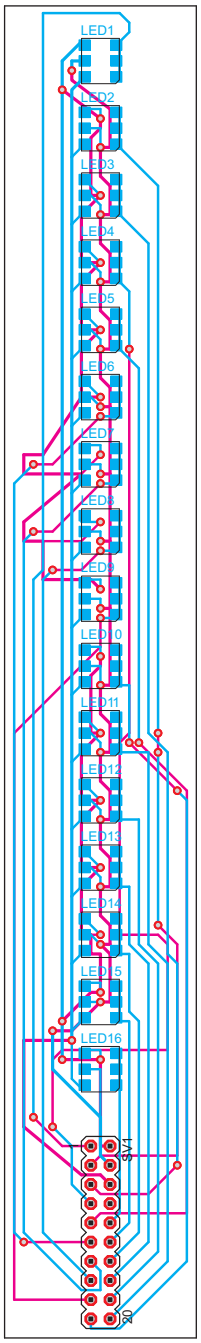
Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2945.

Rys. 6



Rys. 4

Rys. 5



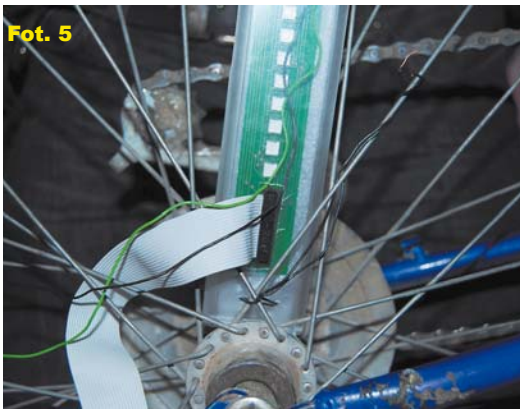
Możliwe modyfikacje

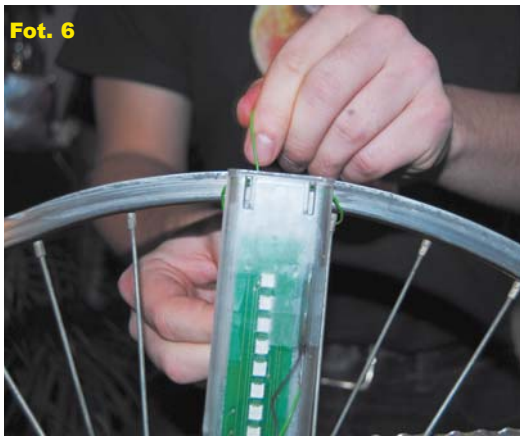
Najprostszą modyfikacją jest dodanie zewnętrznej pamięci Flash. Dzięki temu można, wykorzystując wyprowadzenia Rx i Tx, pokusić się o programowanie mikrokontrolera przez złącze służące łączeniu modułów, a nawet za pomocą interfejsu USB i bootloadera. Ta druga modyfikacja, choć dużo bardziej skomplikowana, umożliwiłaby zmianę wyświetlanego obrazka poprzez włożenie zwykłego pendrive’a do złącza umieszczonego na płytce. Takie rozwiązanie wyeliminowałoby konieczność wyjmowania mikrokontrolera z układu i użycia programatora.

Liczymy, że już niedługo na łamach EdW znajdziemy takie rozwiązania. Do dzieła!


Fot. 3


Fot. 4


Fot. 5


Fot. 6

Warto dodać jeszcze, że kontaktron był połączony ze sterownikiem dwoma przewodami. Można by zamiast takiego rozwiązania połączyć wszystko za pomocą złącza, ale wówczas trzeba by zwiększyć złącze lub zmniejszyć liczbę diod.

Sylwia Babicz
sylwiababicz@gmail.com