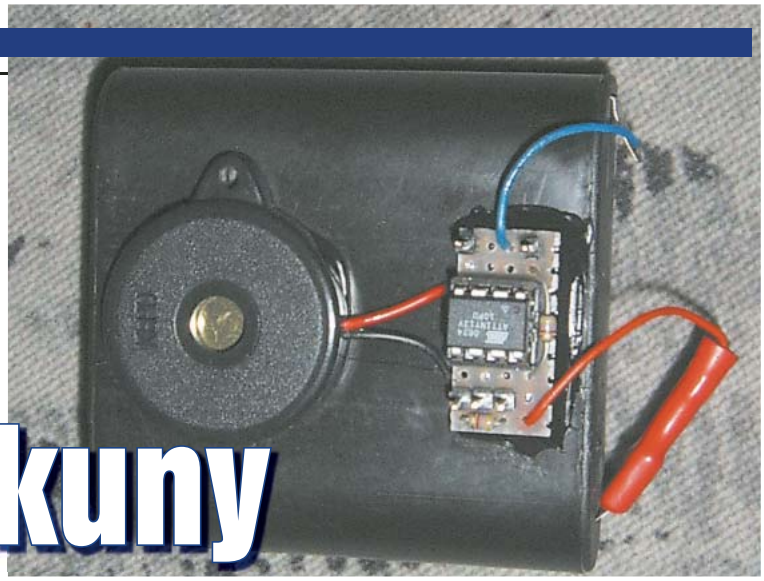


Strach na kuny

Piękna wycieczka krajoznawcza, pogoda świetna, ale zbliża się wieczór. Niedaleko za miastem mały hotelik zaprasza neonem. Chyba skorzystamy, zwłaszcza że nazajutrz mamy jeszcze sporo do przejechania. Las niedaleko, ptaszki śpiewają do późna, istna sielanka. Poranek piękny, wsiadam do samochodu i... nie chce zapalić, sprawdzam – paliwo jest, otwieram maskę, oj bardzo niedobrze, kable zapłonowe wyglądają jak na **fotografii 1**.

Kto nam tak pokrzyżował plany? Oto prawdopodobny sprawca (**fotografia 2**) – kuna, sympatyczne zwierzątko z rodziny łasicowatych, zmora automobilistów. Nawiasem mówiąc, już kilka razy zdarzyło mi się, że znalazłem pod maską samochodu, w okolicy bloku silnika... kurze jajko, ale innych poważniejszych szkód do tej pory nie miałem.

O dalszych kłopotach ze zdobyciem odpowiednich kabli już nie piszę, ale po powrocie do domu zmobilizowałem się bardzo i szybko przystąpiłem do budowy „straszaka na kuny". Oczywiście dużo gotowych urządzeń tego typu można kupić w sklepach elektronicznych i marketach, ale ponieważ części u mnie dostatek, wykonałem sobie coś podobnego i, jak się na razie okazało, bardzo skutecznego.


Fot. 1

przymocowane jest za pomocą rzepa w komorze silnika (**fotografia 4**), wyłącznikiem jest samochodowa wsuwka na biegunie dodatnim.


Fot. 2


Fot. 3

I to już wszystko? A gdzie wartość dydaktyczna artykułu? Bardzo proszę, postanowiłem przy tej okazji nauczyć Czytelników, jak samemu zaprogramować mikroprocesor, zmienić dźwięki, zmodyfikować program itp.

Napisałem w tym celu bardzo prosty program do wytwarzania dźwięków, korzystając z języka Bascom AVR, opisy przy komendach wyjaśniają dokładnie, co w danym momencie program robi.

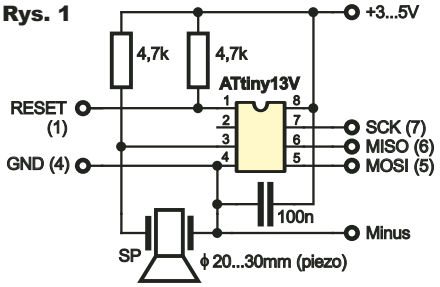
Co nam będzie potrzebne? Komputer z gniazdem portu równoległego LPT, system może być nawet Windows 98, program, który ściągniemy z Elportalu, BASCOM AVR Demo 1.11.7.7 lub nowszą wersję (znajdziemy przez Google). Musimy wykonać też bardzo prosty programator wg schematu z **rysunku 2**, zawiera, oprócz wtyku trzy rezystory i jeden kondensator. Gotowy programator przedstawiony jest na **fotografii 5**. Zastosujemy metodę tzw. ISP (In System Programming), czyli programowanie w układzie. Jest to bardzo wygodny sposób, ponieważ wszystkie zmiany w programie prawie natychmiast możemy wprowadzić do mikroprocesora i sprawdzić w działaniu bez wyłączania zasilania!

A zatem do dzieła, uruchamiamy program i otwieramy plik naszego źródłowego programu (ściągnięty z Elportalu) komendą open file; jego listing powinien pokazać się w dodatkowym oknie. W tym oknie będziemy później program modyfikować. Okno, jeśli trzeba, powiększamy. Następną operacją będzie skompilowanie programu (o odpowiednim

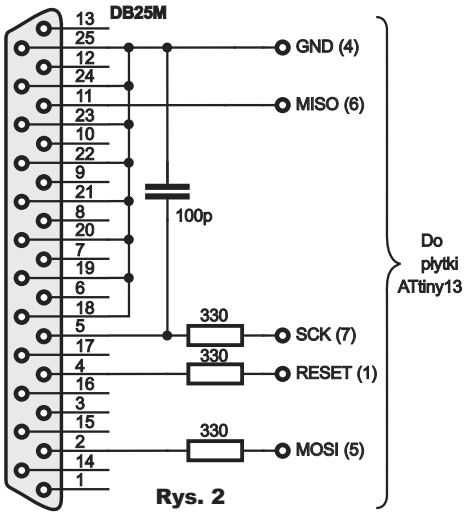
Opis układu

Patrząc na schemat z **rysunku 1** widzimy, że potrzebnych jest niewiele elementów: jeden układ scalony, dwa rezystory, kondensator i głośniczek z płytką piezoceramiczną. Na dobrą sprawę to nawet płytki drukowanej do tego urządzenia nie trzeba robić, tylko elegancko można zmontować układ na małej płytce uniwersalnej. Gotowy „strach" przedstawiony jest na **fotografii 3**. Ale jak to działa? Układ scalony – procesor ATtiny13 generuje kilka dźwięków (3) o różnych częstotliwościach, między dźwiękami są kilkusekundowe przerwy. Następnie głośniczek odtwarza te dźwięki, bardzo skutecznie odstraszając nieproszonych gości. W mojej wersji układ zasilany jest wprost z baterii płaskiej 4,5V, pobór prądu jest mniejszy niż 6mA (bateria starcza na ponad tydzień ciągłej pracy), ale można dodać mały stabilizator typu 78L05 i podpiąć się do instalacji 12V naszego wehikułu. Urządzenie

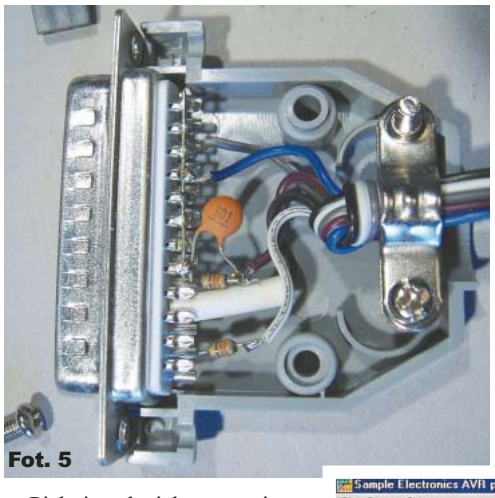
Rys. 1

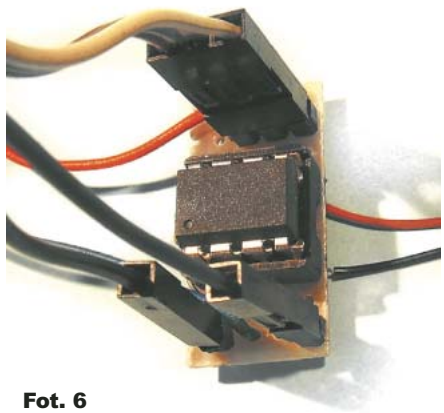



Fot. 4



Rys. 2


Fot. 5


Fot. 6

przetworzeniu) tak, aby stał się zrozumiały dla procesora, robimy to, wciskając F7. Jeżeli nie było błędów (pokazują się one na dole ekranu) podłączamy wsuwki kabelków programatora na odpowiednie goldpiny naszej płytki, jak na **fotografii 6**. Teraz włączamy zasilanie do układu i naciskamy klawisz F4. Jeżeli wszystko jest w porządku, to otwiera się nowe okno obsługi programatora, widać skompilowany program, który będzie wpisany do FlashROM-u. W małym okienku Chip zostanie znaleziony przez program i programator nasz ATtiny13, musimy jeszcze tylko wejść (jest to czynność jednorazowa) do zakładki Lock and Fuse Bits i zmienić fabryczne ustawienie Fusebit E na OFF (wyłączamy dzielnik częstotliwości zegara przez 8) oraz zatwierdzić podświetlonym klawiszem Write FS widać to na **rysunku 3**.

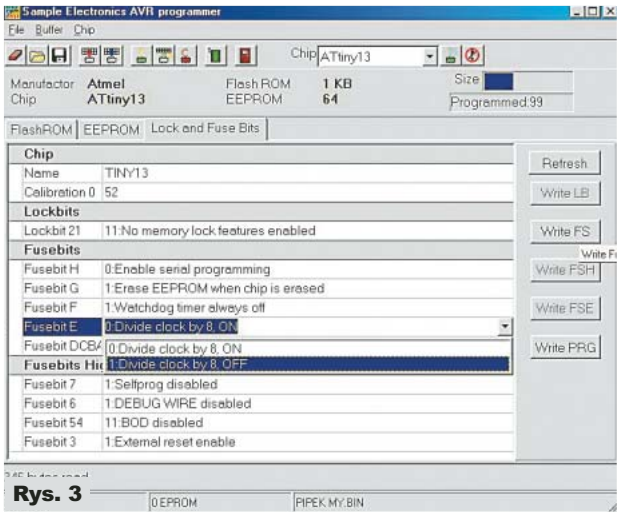
Właściwe programowanie inicjujemy myszką, naciskając na czwartą ikonkę (scalaczek z czerwoną strzałką) rozpoczyna się programowanie mikroprocesora. W głośniczku po chwili będzie słychać owoc naszej dotychczasowej pracy, trzy różne dźwięki z przerwami ok. 5s.

Pięknie, ale jak powstają te dźwięki?

Ależ bardzo prosto! Spójrzmy jeszcze raz na **listing 1** (opisy obok linii). Procesor włącza i wyłącza głośnik (Sp) na określony czas w (us), po czym w trzech pętlach powtarza to po kilka (kilkadziesiąt) tysięcy razy. Tak wytworzony przebieg prostokątny podawany jest potem przez port 4 bezpośrednio na głośnik. Zmieniać możemy do woli te czasy, również czasy przerwy między dźwiękami. Możemy też dodać jeszcze kilka pętli wg uznania (zostało jeszcze sporo wolnej pamięci). Podaję odpowiednio przekształcony wzór dla obliczenia wartości ***waitus*** dla konkretnej częstotliwości. Oto on:

wartość ***waitus*** [us] = 1000000 : 2f [Hz]

Pamiętajmy, że po każdej zmianie w programie, trzeba go ponownie kompilować klawiszem F7, potem wchodzimy przez F4, a przed kolejnym wpisem (czwartą ikonką) do procesora, nacisnąć myszką ósmą ikonkę (czerwony scalaczek z literką C) – czyszczenie pamięci ze starego programu. Jak widzisz, Drogi Czytelniku, możesz sam dokonywać zmian w programie, dowolnie go modyfikować, a efekty praktycznie natychmiast podziwiać. Jeżeli z efektów jesteśmy zadowoleni odłączamy nasz programator od płytki i montujemy „stracha" w pojeździe. W Bascom AVR istnieje też specjalna komenda SOUND, ale jest bardzo pamięciożerna i dla nas nie bardzo użyteczna. Ale można samemu popróbować!



Rys. 3

Listing 1

```
program: „strach na kuny"
'język: Bascom AVR
'napisał: Piotr Świerczek
'data: 10.05.2010
$regfile = „ATtiny13.dat"       'ustawiamy typ procesora
$crystal = 9600000                'częstotliwość zegara
$hwstack = 64                             'ustawienie wielkości stosu
Dim A As Integer                  'miejsce w pamięci dla zmiennej A
Config Portb.4 = Output         'port 4 (nóżka 3) ustawiamy jako wyjście
Portb.4 = 0                               'wartość początkowa
Sp Alias Portb.4                  'port 4 otrzymuje nazwę Sp(głośnik)
   'tutaj zaczyna się właściwy program
For A = 1 To 3000                 'początek pierwszej pętli
Sp = 0                                    'ustaw na głośniku stan 0
Waitus 1000                       'czekaj 1000 us
Sp = 1                                    'ustaw na głośniku stan 1
Waitus 1000                       'czekaj 1000 us
Next A                                    'dodaj do A 1 i rób tak aż doliczysz do 3000
Wait 5                                    'czekaj 5 sekund
For A = 1 To 3000                 'początek drugiej pętli
Sp = 0                                    'jak wyżej tylko
Waitus 1500                       'czasy są inne i właśnie
Sp = 1                                    'zmieniając te wartości
Waitus 1500                       'uzyskujemy inne
Next A                                    'częstotliwości(wzór w tekście)
Wait 5
'początek trzeciej pętli
For A = 1 To 20000                'tutaj np. trzeba było wydłużyć
Sp = 0                                    'czas trwania dźwięku ponieważ włączanie i wyłączanie
Waitus 100                        'trwa bardzo krótko, jak łatwo zauważyć trzeba było zwiększyć
Sp = 1                                    'ilość powtórzeń (okresów) przebiegu prostokątnego
Waitus 100
Next A
Wait 5
Return                                    'wróć na początek pierwszej pętli i tak
'po kolei wykonuj wszystko bez końca.
```

Piotr Świerczek
sp9egm@wp.pl

Wykaz elementów

330Ω x 3 szt.
4,7kΩ x 2szt.
100pF
100nF
ATtiny13V
Piezo
DB25M