kit 3124 AVT

Kostka do gry

Miniaturowa, sześcienna elektroniczna wersja tradycyjnej kostki do gry, wyposażona w czujnik wstrząsów. Kolorowe diody LED, efektowne animacje oraz energooszczędność – spodoba się każdemu!

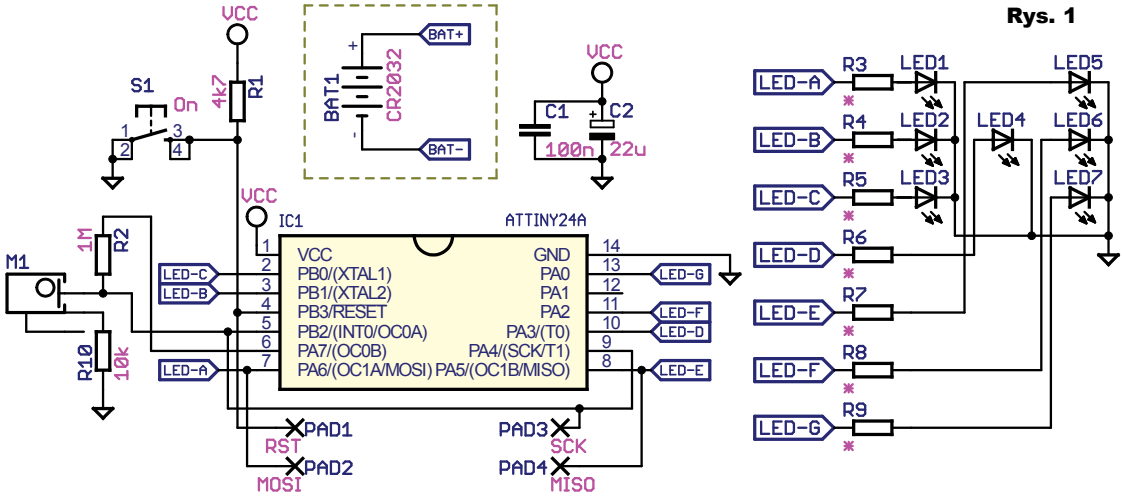
W różnych czasopismach i w Internecie zamieszczonych jest mnóstwo projektów elektronicznych kostek do gier. Osobiście nie mogłem jednak znieść wielkich pudełek z 9-woltowymi bateriami, wyzwalanych przyciskiem. Przedstawiony tutaj gadżet jest bardzo bliski tradycyjnej, sześciennej kostce do gry. Elektroniczna wersja ma podobny kształt, wynik prezentowany jest w postaci oczek, a wirtualny rzut wykonywany jest przez potrząsanie urządzeniem. Towarzyszą temu ciekawe animacje, automatyczne, płynne ściemnianie diod, ciekawy i estetyczny wygląd całości. Źródłem zasilania jest mała bateria litowa, a elektronika cechuje się dużą oszczędnością energii, co właśnie było dla mnie priorytetem przy projektowaniu tej kostki.

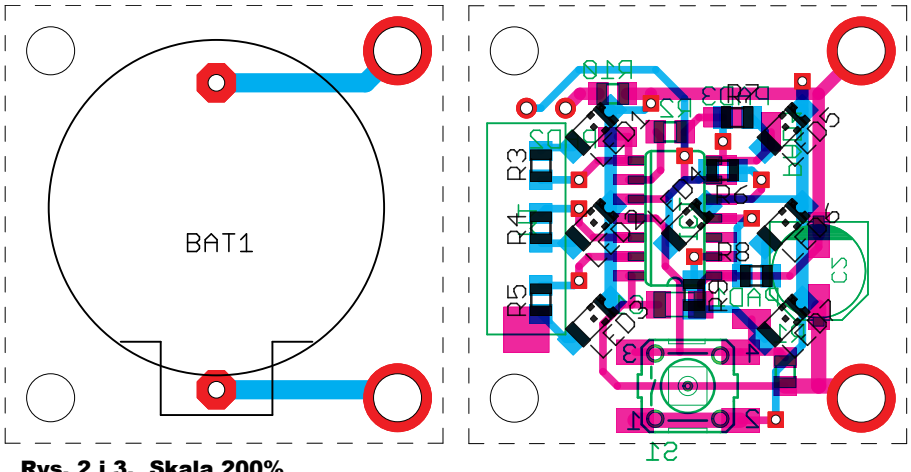
Urządzenie to dedykuję szczególnie miłośnikom gier planszowych i tradycyjnych RPG-ów, którzy chcą połączyć przyjemność grania z elektronicznym hobby. Może to być również świetny prezent dla bliskiej osoby. Choć w tradycyjnych planszówkach zwykło się pozostać przy tradycyjnych kostkach, to czasem miło jest pozwolić sobie na jakąś odmianę, co zawsze może być źródłem nowych wrażeń. Warto wykonać sobie takie elektroniczne kostki, zwłaszcza z diodami o różnych barwach – z doświadczenia mogę zapewnić, że spodobają się każdemu i przyciągną jego uwagę na dłuższą chwilę!

Opis układu

Budowa urządzenia jest tak prosta, jak to możliwe (i nie prostsza, jak ostrzegał Einstein). Jak wspomniałem, projektując schemat ideowy, kierowałem się przede wszystkim założeniem minimalizacji poboru energii. Ukazałem go na **rysunku 1**. Jak widać, źródłem zasilania jest tu jedna, 3-woltowa bateria litowa CR2032, a najważniejszym elementem mikrokontroler AVR ATtiny24A, wyposażony w 2kB pamięci programu. Wybrałem go, ponieważ ma niewielką obudowę, odpowiednio dużą liczbę uniwersalnych wyprowadzeń, obsługuje przerwanie zewnętrzne, ma dwa liczniki-timery, pracuje w szerokim zakresie napięć 1,8–5,5V oraz, odpowiednio wykorzystany, może pobierać bardzo mały prąd. Element M1 to czujnik wstrząsów, ma on kształt walca, wewnątrz którego swobodnie porusza się metalowa kulka. Na jednym z jego końców znajdują się styki, które mogą być zwarte elektrycznie przez tę kulkę. Megaomowy rezystor R2 polaryzuje wejście czujnika i piny PB2, PA4 mikrokontrole-

Rys. 1





Rys. 2 i 3. Skala 200%

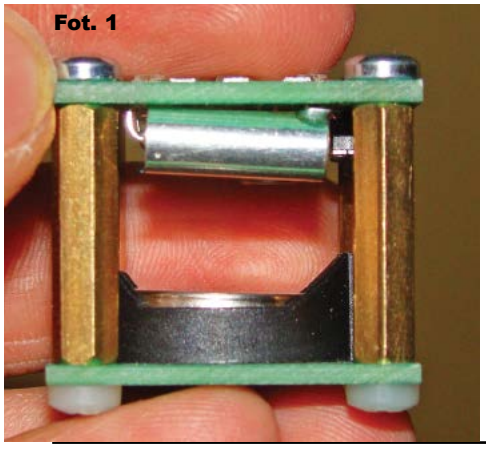
ra, które sprawdzają stan logiczny i mają możliwość wywołania przerwania sprzętowego INT0 (PB2) oraz zliczania impulsów przez Timer1 (PA4), co znacznie ułatwia pisanie programu.

A dlaczego rezystora R2 nie dołączyłem bezpośrednio do plusa zasilania, tylko do innej uniwersalnej nogi IC1? Bo dzięki temu mogę wyłączyć czujnik wstrząsów (logiczne 0) i w razie zwarcia przez kulkę nie będzie płynął przez niego prąd (niewielki, bo ok. 3µA, ale zawsze), co było jednym z zabiegów mającym na celu zmniejszenie poboru prądu. Czujnik wstrząsów jest wyłączany razem z całym urządzeniem, a mikrokontroler wprowadzany wtedy w tryb powerdown. Całość pobiera wówczas niewielki ułamek mikroampera. Mogłem sobie na to pozwolić, bo miałem wolne linie I/O. Rezystor R10 o małej wartości 10kΩ pełni funkcję zabezpieczającą na wypadek, gdyby ktoś przez pomyłkę skonfigurował PB2/PA4 jako wyjście i ustawił na nim logiczne 1. Wtedy zwarcie przez kulkę mogłoby zniszczyć mikrokontroler lub czujnik.

Uwagę zwraca również przycisk S1. Jego naciśnięcie powoduje reset mikrokontrolera, a po zwolnieniu zaczyna on program od początku. Postanowiłem, że kostkę będzie można włączyć tylko przyciskiem, nie przez potrząsanie – ma to bardzo duży wpływ na pobór energii. Transport urządzenia w kieszeni, plecaku, samochodem lub rowerem, bez przerwy powodowałby wybudzanie mikrokontrolera z trybu powerdown i losowanie. Bateria padłaby natychmiast, więc byłoby to kiepskie rozwiązanie. Przełącznik został umieszczony na górnej płytce przy krawędzi, na dolnej warstwie.

Diody dołączone są bezpośrednio do wyprowadzeń, oczywiście przez rezystory ograniczające prąd. Są one multiplekso-wane (działają na zmianę 4 i pozostałe 3 diody), aby zmniejszyć średnie zapotrzebowanie na energię. Pady PAD1-PAD4 służą do wygodnego dołączenia przewodów programatora ISP, kiedy mikrokontroler zostanie już zamontowany na płytce.

Źródło zasilania – bateria litowa – jest podłączone do VCC i GND. Oznaczenia BAT+ i BAT– wzięły się stąd, że postanowiłem umieścić ją na osobnej płytce drukowanej, a połączenia między nimi będą zapewniały nam metalowe tuleje. W ten sposób optymalnie wykorzystałem przestrzeń i zapewniłem estetyczny wygląd, pozbawiony jakichkolwiek przewodów oraz wygodne montowanie i złożenie całości.



Fot. 1

Montaż i uruchomienie

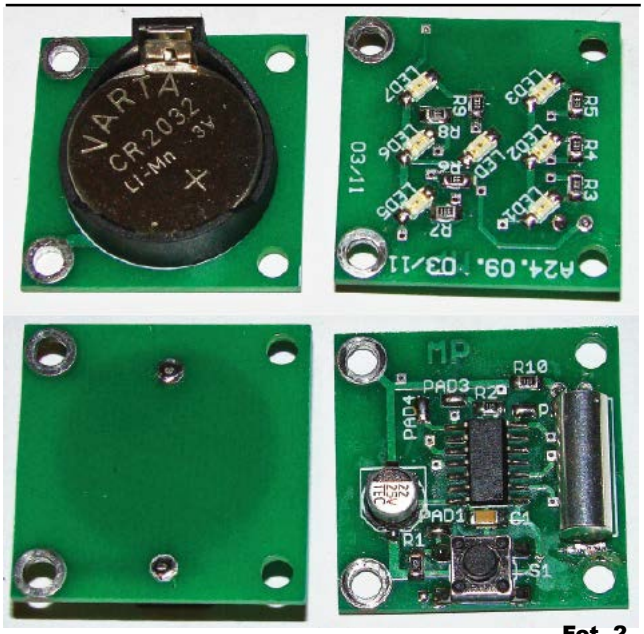
Kostka do gry została zmontowana na dwóch płytkach drukowanych, które są kwadratami o boku 29mm. Starałem się, aby były one tak małe, jak to możliwe – ich rozmiar narzucił koszyk na baterię. Widoczna na **rysunku 2** płytka dolna nic poza nim nie zawiera, ale musi być dwustronna z metalizacją. Jak widać na fotografiach, obie te płytki są skręcone metalowymi tulejami, tworząc zwartą konstrukcję. Dwie z tych tulei służą dodatkowo za przewodniki, dostarczające zasilanie do górnej płytki. Nie ma tu żadnych lutów, są one dociskane do metalizowanych otworów przez śruby. Nie mogłem zastosować metalowych śrub, na których stałaby kostka, ponieważ położenie jej na przewodzącej powierzchni spowodowałoby zwarcie styków baterii. Postanowiłem zatem użyć poliamidowych śrub na dół, więc gdyby nie było metalizacji, a ścieżki byłyby na dole, poliamid nie zapewniłby kontaktu elektrycznego. A gdyby ścieżki były na górze, połączenie tulejek ze ścieżkami byłoby zapewnione, lecz nie dałoby się przylutować (solidnie i estetycznie) koszyka na baterie. Dlatego zdecydowałem się na PCB dwustronną z metalizacją, a ścieżki poprowadziłem po stronie górnej.

Druga płytka, widoczna na **rysunku 3**, zawiera całą elektronikę i potrzebuje z zewnątrz jedynie zasilania. Również jest ona dwustronna (z metalizacją), na górnej warstwie znalazły się tylko diody i rezystory ograniczające ich prąd oraz kilka ścieżek. Na dolnej jest mikrokontroler, czujnik wstrząsów, kondensatory i kilka pojedynczych drobiazgów. Przycisk, umieszczony przy krawędzi na spodzie, jest łatwo i wygodnie dostępny, nawet dla osób z grubymi palcami.

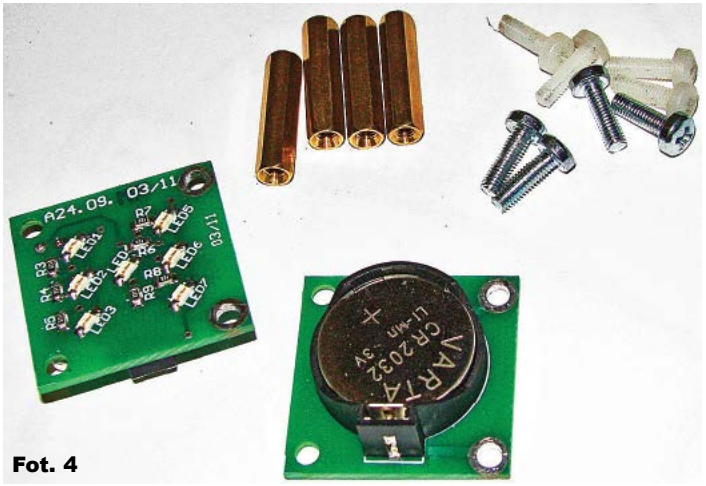
Jedyne, co można doradzić w kwestii montażu płytki z baterią, to zalecić skrócenie odstających nóżek koszyka, aby płytka stała na łbach śrub. Natomiast montaż górnej zaczynamy od przylutowania mikrokontrolera. Jest to prostsze, niż się wydaje. Bardzo pomocny jest w tym topnik, bo zwiększa napięcie powierzchniowe cyny i „rozrywa" zwarcia między nóżkami. Z resztą części SMD pójdzie łatwiej. Jako przedostatni element trzeba zamontować czujnik wstrząsów. Ma on specyficzną cechę, mianowicie elektrody zwierane przez kulkę znajdują się na jednym końcu rurki. W praktyce, kiedy kulka sturla się tam, kostka będzie nadwrażliwa. Dlatego czujnik warto zamontować poziomo, ale pod lekkim kątem, jak widać na **fotografii 1**. Tym sposobem, po postawieniu urządzenia na stole, kulka sturla się do drugiego końca i wszystko będzie działać jak należy. Do ustalenia kąta i umocowania tego końca stosujemy pad lutowniczy i grubą kroplę lutowia (tylko nie przegrać obudowy, bo odpadną nóżki!). Na samym końcu lutujemy przycisk.

Kiedy dolna strona jest już gotowa, bierzemy się do strony diod. Wybieramy barwę i odpowiednie rezystory ograniczające prąd. Dla czerwonych i żółtych polecam 470Ω, zielonych 220Ω, niebieskich i białych 330Ω. Dobrałem je doświadczalnie, aby jasność była podobna. Dobrze jest chwycić płytkę w jakieś małe imadło lub uchwyt montażowy, aby nie przeszkadzały elementy po przeciwnej stronie. Zmontowane płytki można zobaczyć na **fotografii 2**.

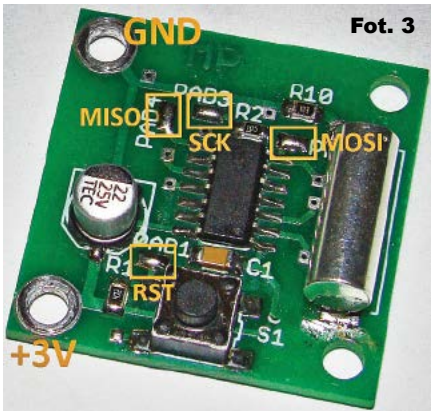
Teraz trzeba zaprogramować mikrokontroler ATtiny24A. Używamy do tego specjalnych padów, opisanych na **fotografii 3**, do których lutujemy krótkie kabelki



Fot. 2



Fot. 4



Fot. 3

ISP. Trzeba pamiętać też o dołączeniu zasilania – można przykręcić przewód krótką śrubą do otworów na tuleje, wystarczy tutaj litowa bateria 3V. Skompilowane wsady oraz listing można znaleźć wśród materiałów na Elportalu. Nie trzeba nic zmieniać w ustawieniach fuse-bit. Każdy może dowolnie modyfikować program lub napisać swój od nowa – wrócę do tego później.

Po zaprogramowaniu usuwamy przewody ISP i możemy skręcić konstrukcję. Używamy do tego celu metalowych, sześciokątnych tulei z gwintem M3 o wysokości 20mm. Przy tulejach 25mm kostka była wysokim prostopadłościanem i przypominałaby blok mieszkalny. Na spód stosujemy obowiązkowo śruby z tworzywa sztucznego (poliamid) o długości gwintu 8mm, a na górę metalowe (ładniejsze i tańsze) długości nie większej niż 12mm. Zestaw widoczny jest na **fotografii 4**. Chciałbym jeszcze zwrócić uwagę na specyficzną budowę metalowych tulejek. Jak zauważyłem, gwint nie biegnie u nich od początku do końca, ale mają w środku swego rodzaju bardzo cienką blokadę. Sęk w tym, że nie leży ona dokładnie pośrodku! To akurat dla nas duży plus, bo może być trudno dostać krótką (<12mm) śrubę metalową M3, a poliamidową – bez problemu (plastik można łatwo skrócić). Zmontowałem 4 kostki, najpierw na tulejach 25mm, potem zmieniłem na 20mm i wszystkie bez wyjątku miały tę asymetryczność. Na **fotografii 5** można zobaczyć z bliska jedną z moich kostek z krótko przyciętymi końcówkami koszyka na baterie.

Aby włączyć kostkę, naciskamy przycisk – wszystkie diody powinny błysnąć. Jeśli potrząśniemy urządzeniem, wykonamy wirtualny rzut. Kolejne wyniki będą szybko zmieniały się na diodach, stopniowo zwalniając i w końcu zatrzymując się w miejscu. Wynik będzie wyświetlany przez 6 sekund, po czym płynnie zostanie zmniejszona jasność (oszczędność energii). Po kolejnych 10 sekundach diody całkowicie się wyłączą. Od tego momentu jeszcze przez 70 sekund kostka pozostanie w stanie czuwania, gotowa do wykonania kolejnego „rzutu". Dopiero po tym czasie czujnik zostanie wyłączony, a mikrokontroler wejdzie w tryb powerdown.

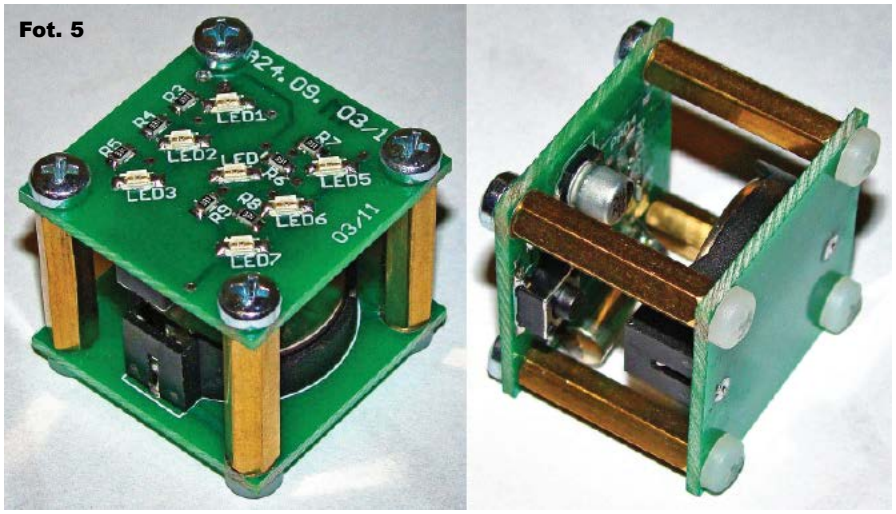
A jak można kostkę szybko wyłączyć, skoro przycisk powoduje reset? Ktoś mógłby zaproponować wykorzystanie pamięci EEPROM, ale ona ma ograniczoną liczbę zapisów i by się zużyła. Ja to zrobiłem inaczej: jeśli po resecie przez 3 sekundy nie potrząśniemy kostką (na tyle intensywnie, aby wykonać wirtualny rzut), urządzenie natychmiast się wyłączy, co też potwierdza błyśnięcie LED-ów. Na **fotografii 6** można zobaczyć działające kostki (w rzeczywistości jasność diod pozwala na grę nawet w silnym oświetleniu).

Jeśli chodzi o pobór prądu, to przy wyświetlaniu „6" na pełnej jasności (zaraz po wylosowaniu), układ czerpie z baterii ok. 8mA (wersja czerwona). Po wyłączeniu diod, w trybie oczekiwania, działa tylko mikrokontroler w stanie aktywnym i pobiera 500µA. Po wyłączeniu urządzenia i wejściu w tryb powerdown, pobór prądu wynosi ok. 0,25µA. Typowa bateria CR2032 ma deklarowaną pojemność ok. 220mAh. Litowe ogniwo pozwoli więc na bardzo długą eksploatację kostki.

Oprogramowanie

Program dla mikrokontrolera w kostce do gry napisałem w BASCOM-ie i zajmuje on 60% pamięci FLASH układu. Jest on oczywiście dostępny dla Czytelników w Elportalu wśród materiałów dodatkowych, listing zawiera komentarze, a wszelkie parametry, jak czas do przyciemnienia, wyłączenia itp. są zadeklarowane jako stałe, więc każdy może dowolnie je zmienić, dostosowując do swoich preferencji.

Taktowanie mikrokontrolera wynosi 8MHz (wewnętrzny oscylator RC), ale włączony jest fuse-bit dzielący tę częstotliwość przez 8, stąd wartość 1MHz. Następnie w programie konfigurowane są uniwersalne linie I/O, jedyne, na co chciałbym teraz zwrócić uwagę, to usta-



Fot. 5

wienie PA1 jako wyjścia – ta noga „wisi w powietrzu”, domyślnie są one ustawiane jako wejścia bez wewnętrznego pull-upu rezystorem do plusa. Nie można tak tego zostawić, dziwię się, czemu w książkach o programowaniu µC tak niedbale pomija się tę kwestię. Wspomina o niej producent w danych katalogowych. Niepodłączony pin działa jak antena, częściowo otwierając oba komplementarne tranzystory CMOS na wejściu, powodując dodatkowy przepływ prądu (marnotrawienie!). Dlatego skonfigurowałem je jako wyjście i ustawione zostało na nich logiczne 0.

Następnie skonfigurowane są peryferia, co pokazałem na **listingu 1**. Timer0 wywołuje przerwania co określoną ilość czasu, natomiast Timer1 jest ustawiony jako licznik, który zlicza do swego 16-bitowego rejestru zbocza opadające na PA4, czyli tam, gdzie podłączony jest czujnik wstrząsów. Ostatnie 3 wiersze na tym listingu wyłączają nieużywane analogowe peryferie, jak komparator i ADC (Brown-Out Detector jest zablokowany fuse-bitem). Kiedy je wyłączymy, automatycznie wyłączane jest wewnętrzne źródło napięcia odniesienia i pobór prądu spada – w trybie power-down kilkadziesiąt razy!

W dalszym ciągu programu definiowane są aliasy (wygodniejsze nazwy) dla różnych wyprowadzeń I/O oraz odblokowywane przerwania i timery. W pętli głównej nie dzieje się nic, poza sprawdzaniem, czy należy wejść w tryb power-down. Złą praktyką jest usypianie mikrokontrolera w jakimś innym przerwaniu lub podprogramie, bo lubią się wtedy zawieszać. W przypadku kostki to ma mniejsze znaczenie, bo i tak budzimy ją resetem, ale trzeba trzymać się dobrych nawyków.

Wszystkie operacje wykonywane są w przerwaniu *Czas*, wywoływanym przez Timer0 co około 2ms. Obok każdego bloku kodu znajduje się odpowiedni komentarz,

```
Config Timer0 = Timer , Prescale = 8
Config Timer1 = Counter , Edge = Falling
On Timer0 Czas

Config Aci = Off
Stop Ac
Stop Adc
```

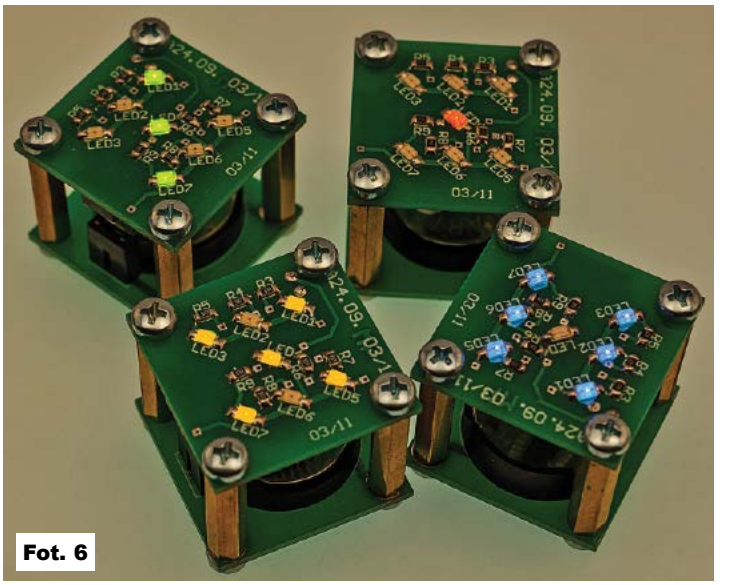
Listing 1

myślę, że każdy, kto siądzie do tego kodu, szybko go rozpracuje i łatwo wprowadzi swoje poprawki.

Do losowania potrzebujemy zmiennej losowej, a ona z kolei źródła entropii, dostarczającego dane o przypadkowym charakterze. Źródłem zaburzenia jest tutaj czujnik wstrząsów – urządzenie zlicza do zmiennej bajtowej każdy impuls wygenerowany przez czujnik, a jeśli w jednostce czasu wystąpi ich więcej lub tyle, ile wynosi wartość stałej *Czulosc*, nastąpi losowanie (ta zmienna losowa modulo 6). Co każde wystąpienie przerwania *Czas*, zawartość Timer1 jest porównywana ze stałą *Czulosc*, dodawana do zmiennej losowej, a rejestr licznika zerowany. I tak w kółko. Zmienna losowa jest bajtem, czyli jej wartość może być w zakresie 0–255, ale pamiętajmy, że przy dodawaniu, gdy przekroczymy jej wartość, zaczynamy liczyć od zera (licznik się przekręca).

Bit *Turnoff* pamięta, czy po uruchomieniu (resecie) kostki wystąpił już przynajmniej jeden wirtualny rzut. Jeśli nie, po upływie 3 sekund urządzenie się wyłącza.

Chciałbym zwrócić również uwagę na możliwość napisania własnego programu – urządzenie ma bardzo prostą i uniwersalną konstrukcję. Nie wykorzystałem w swoim programie zewnętrznego przerwania INT0, wyzwalanego czujnikiem wstrząsów. Nic nie stoi na przeszkodzie, aby wybudzać kostkę przez potrząsanie! Na pewno jest to swego rodzaju wygoda, nie trzeba bowiem wciskać przycisku, ale trzeba liczyć się z większym apetytem na prąd. Można też wymyślić i wprowadzić jakieś nowe efekty graficzne – nawet w Bascomie, przy dobrym optymalizowaniu kodu, da się ich wcisnąć tam mnóstwo!



Fot. 6

Przedstawiona tutaj elektroniczna odmiana tradycyjnej kostki to przyciągający uwagę gadżet. Jeszcze ciekawiej jest, gdy mamy ich kilka, w różnych barwach. Zapewniam, że fajnie jest zagrać w dobrym towarzystwie w chińczyka lub monopoly, używając dla odmiany własnoręcznie zbudowanych kostek z procesorem i diodami!

Michał Pędzimąż
mpedzimaz@gmail.com

Wykaz elementów

R1	4,7kΩ SMD 805
R2	1MΩ SMD 805
R3–R9	*patrz tekst SMD805
R10	10kΩ SMD 805
C1	100nF SMD 1206
C2	elektrolityczny 22µF SMD
LED1–LED7	diody LED 1206
IC1	ATtiny24A
S1	microswitch SMD
BAT1	koszyk na baterię CR20XX poziomy
M1	czujnik wstrząsów (w AVT MER4)
Tuleja metalowa 20mm z gwintem M3 (w AVT TFF-M3/20)	4 szt.
Śruba poliamidowa M3 8mm (w AVT CRS-7045-M3-8)	4 szt.
Śruba metalowa M3 12mm (w AVT B3X12/BN1435)	4 szt.

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-3124.

Fot. 7