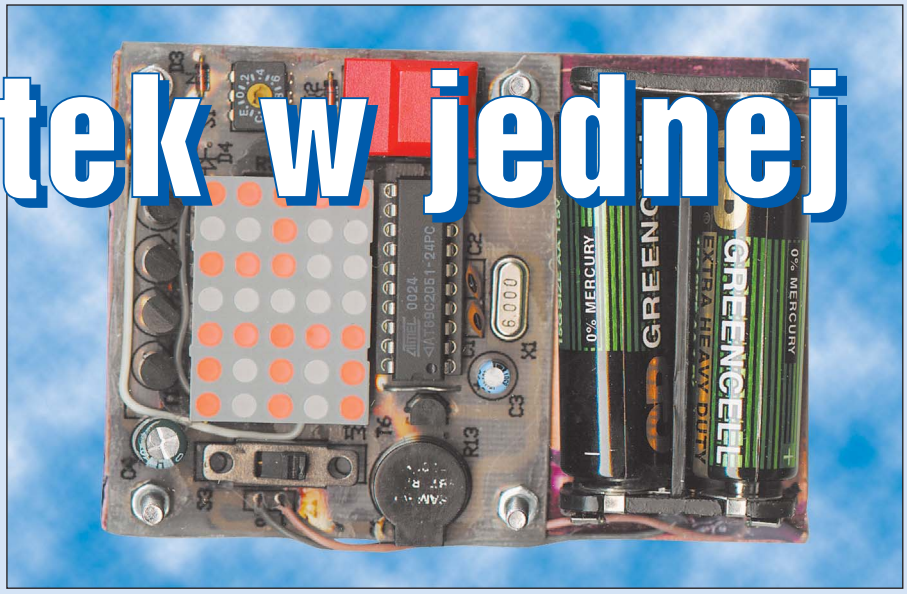
10 kostek w jednej

kit 2491 AVT



Do czego to służy?

Zacznijmy od początku – najpierw Bóg stworzył Ziemię, potem stworzył człowieka i osadził go na Ziemi. Dał człowiekowi wolną wolę, a człowiek postanowił uprościć sobie życie i stworzył mnóstwo rzeczy powodujących, że ma więcej czasu. Posiadając coraz więcej wolnego czasu człowiek, odkrył nudę i postanowił na powrót zapełnić wolny czas. W tym celu wymyślił różne gry, ale prawdziwą atrakcją było wprowadzenie do nich odrobiny przypadkowości zwanej szczęściem, losem, przeznaczeniem, hazardem... Aby szczęście mogło się przejawić, wymyślono kostkę – to był pierwszy przełom. Drugi właśnie następuje teraz i aktualnie o nim czytasz.

Tą drogą doszliśmy skrótowo od stworzenia świata do chwili obecnej, a ja kończę spekulować i zabieram się do rzeczy.

Układ jest, kolejnym zresztą, opracowanym przeze mnie „kombajnem losującym". Zawiera on w sobie 10 kostek, są to k3, k4, k6, k8, k10, k12, k20, k30, k50 i k100. Z tego typu kostkami spotkałem się w swojej karierze Mistrza Gry i Gracza.

Tak na marginesie: bardziej spostrzegawczy, ale mniej obeznani z grami fabularnymi stwierdzą może, że nie da się stworzyć figury przestrzennej o 3 ścianach, więc w rzeczywistości nie istnieje kostka 3-ścienna. Jednak w literaturze można spotkać się z takim tworem – w tym celu po prostu rzuca się kością k6, wynik dzieli przez dwa i zaokrągla w górę. W podobny sposób można w przedstawionym układzie uzyskać kostki k25, k5, k2.

Projektując układ postawiłem przede wszystkim na prostotę obsługi. Odpowiednia kostka jest wybierana po prostu za pomocą 10- pozycyjnego przełącznika kodującego BCD. Oprócz przełącznika wyboru kostki jedyne ruchome części to włącznik zasilania oraz przycisk startu losowania.

Aby uatrakcyjnić układ, dodałem do niego maleńki głośniczek wytwarzający różne stuki i trzaski podczas losowania. Symulację foniczną kostki można wyłączyć przytrzymując przycisk losowania podczas włączania układu – na początek trochę poprotestuje piszcząc, ale po chwili zamilknie (tak nawiasem pisząc, to ten pisk to nie moja sprawka – to wina procedur umieszczonych w procesorze przez kompilator).

Jakby tego było mało, wynik po puszczeniu przycisku nie ukazuje się od razu, lecz zwalnia, a ostatecznie miga trzykrotnie. Podczas losowania wyświetlacza pojawia się obiegająca go kreska. W czasie losowania oraz podczas zwalniania wyniku zmiany ustawień lub ponowne naciskanie przycisku nic nie daje. Zostało to wprowadzone, aby zapobiec ewentualnym próbom oszustwa.

Ale to nie wszystko! Kostki k3, k4 i k6 są wyświetlane w postaci kropek (jak w prawdziwej kostce), wyniki kostek k10 i k100 zaczynają się od 0, a reszty od 1 – również tak jak jest naprawdę.

Drogi Czytelniku czy trzeba Cię jeszcze przekonywać? Tak? To dodatkowo napiszę, że program został napisany w taki sposób, że posiadając BASCOM-a i programator możesz w prosty sposób zaprojektować własne typy kostek i niemal dowolny wygląd pokazywanych wyników oraz animacje inne niż obiegająca wyświetlacz kreska!

Myślę, że nie trzeba nic więcej pisać – ten układ reklamuje się sam.

Jak to działa?

Schemat ideowy układu przedstawiony jest na **rysunku 1.** Nie jest to układ zbyt skomplikowany, co udało się osiągnąć dzięki zastosowaniu mikrokontrolera AT89C2051.

Wyświetlacz pracuje na zasadzie multipleksowania z częstotliwością około 80Hz. Ze względu na niewielką liczbę wyprowadzeń układu U1 koder dziesiętny S1 został podłączony razem z wyświetlaczem – jego odczyt jest dokonywany po każdorazowym przeskanowaniu wszystkich linii wyświetlacza. Diody D1-D4 zabezpieczają nas przed sytuacją, w której koder S1 zwierałby ze sobą linie sterujące katodami wyświetlacza – nie spowodowałoby to co prawda uszkodzenia układu, jednak z pewnością uniemożliwiłoby prawidłowe wyświetlanie wyników. Tranzystor T6 steruje głośnikiem poprzez rezystor R13, który zmniejsza co prawda głośność, ale też, co jest dość istotne, pobór prądu. Dioda D5 została dodana, aby spiąć ewentualne przepięcia pojawiające się na cewce głośnika. Przełącznik S2 to przycisk startu losowania.

Układ powinien być zasilany napięciem z przedziału 2,7 – 6V co umożliwia zastosowanie dwóch baterii R6, które przy stosunkowo małym poborze prądu przez układ (średnio około 100mA) powinny wystarczyć na 8-10 godzin pracy non-stop.

Naturalnie dusza układu nie mieści się na schemacie, ponieważ jest nią program sterujący umieszczony w kostce U1. Tutaj ze względu na ograniczoną ilość miejsca omówię tylko jego fragmenty.

Dla tych, którzy chcieliby program przerobić na AVR-a mam smutną wiadomość: niestety nie będzie to łatwe ze względu na użycie przeze mnie w kodzie dużej ilości asemblera.

Program zawiera się w dwóch plikach: Kostka.bas i Dane kostki.bas. Pierwszy z nich zawiera główną część programu i jego edycje zalecałbym osobom troszkę bardziej obeznanym z tematem. Drugi natomiast zawiera podprogram *Ustaw_adres_danych*, tablice zawierającą dane animacji (w moim przypadku kreska obiegająca wyświetlacz), tablice *Data_typy_kostek* zawierającą liczbę ścianek kostki dla kolejnych pozycji kodera S1 i kilka tablic opisujących wygląd kostki.

Tablice wyglądu kostki składają się z elementów po 5 bajtów. Każdy taki element opisuje jakby kolejną ściankę kostki (wyobraźcie sobie, jaka to była robota wpisać dane dla k100), przy czym 0 oznacza diodę zapaloną, 1 – zgaszoną. Tablice są ułożone w taki sposób, że obraz będzie wyglądał na wyświetlaczu identycznie jak na tablicy. Najlepiej zobaczyć to na przykładzie danych do kostki k3 pokazanych na **Listingu 1.** Przy tej kostce program wewnętrznie losuje liczby z zakresu 0-2, gdy zostanie wylosowane 0, to pobranych zostanie 5 pierwszych bajtów z tablicy i wyświetlona zostanie pojedyncza kropka, gdy zostanie wylosowana 1 – pobrane zostaną bajty 6-10... itd.

Podprogram *Ustaw_adres_danych* jest napisany w asemblerze, ale jego działanie jest stosunkowo proste: Jest on wywoływany, gdy zostanie zmieniona pozycja S1, a jego zadaniem jest – na podstawie zawartej w akumulatorze pozycji przełącznika – wpisanie do zmiennej *W_adres_danych_kostki* wskaźnika do odpowiedniej tablicy. Zmienna ta jest następnie używana podczas wizualizowania wyniku losowania.

Po zmianie ustawienia kodera i upływie około 1 sekundy wyświetlany jest nowo wybrany typ kostki (miga trzy razy), przy czym k100 jest wyświetlana jako C (rzymskie 100), ponieważ na wyświetlaczu nie potrafiłem pokazać liczby 100 „po Polsku". Po co to opóźnienie? Dzięki temu rozwiązaniu unikniemy kilkakrotnego migania kolejnych typów kostek podczas kręcenia przełącznikiem – po wybraniu typu kostki liczba jej ścianek wyświetli się dopiero po 1 sekundzie i tylko raz. Na **Listingu 2** przedstawiłem pętle główną (zresztą nieprzyzwoicie krótką), **Listing 3** natomiast pokazuje fragment obsługi przerwania *Timera 0* odpowiedzialny za obsługę kodera. Oto krótki komentarz: Po wykryciu zmiany ustawienia S1 do zmiennej *B_stalosc_kostki* wpisywana jest pewna stała wartość, która następnie jest zmniejszana co 1/80 sekundy. Gdy jej wartość „zejdzie" do zera, wybór kostki jest „zatwierdzany", co objawia się ustawieniem bitu *Bit_kostka_zmieniona,* co z kolei powoduje w pętli głównej wyświetlenie nowej kostki i 3-krotne mignięcie wyświetlacza. Należy tutaj zaznaczyć, że chociaż typ kostki jest wyświetlany z opóźnieniem, to sama kostka jest wybierana natychmiast, co może spowodować, że gdy naciśniemy przycisk losowania zaraz po przestawieniu S1 nie zostanie wyświetlony nowy typ kostki, mimo że losowanie odbywać się będzie z kostką zgodną z pozycją S1.

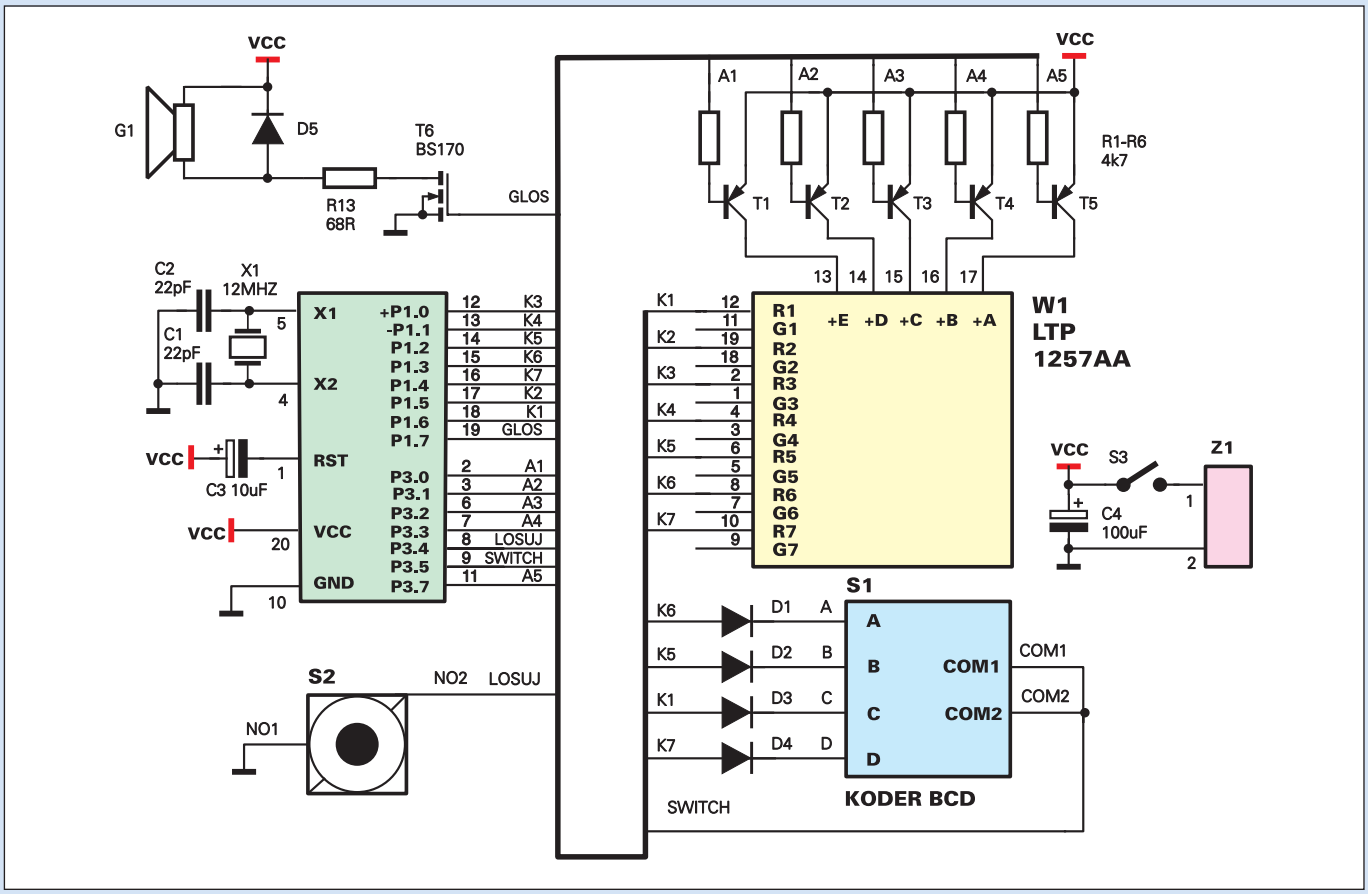
Listing 1

```
Data_k3:
'1
Data &B1111111
Data &B1111111
Data &B1110111
Data &B1111111
Data &B1111111
'2
Data &B1111111
Data &B1111111
Data &B1011101
Data &B1111111
Data &B1111111
'3
Data &B1110111
Data &B1111111
Data &B1111111
Data &B1111111
Data &B1011101
```

Listing 2

```
Pętla_główną:
Do
Debounce Losuj , 0 , Losowanie
If Bit_kostka_zmieniona = 1 Then
  B_wynik = 0        '- aby nie otrzymywać
                        bzdurnych wyników
  Reset Bit_kostka_zmieniona
                  '- aby jak najszybciej uzyskać
                        info. o ponownej zmianie
  Gosub Wyświetl_ustawioną_kostkę
  Gosub Mignij_3x
End If
Loop
```

Z ciekawszych elementów programu wypada jeszcze przedstawić fragment odpowiedzialny za losowanie – widzimy go na **Listingu 4:** Po wejściu w tryb losowania ustawiane są zmienne *Bit_losowanie_trwa* i *Bit_animacja* – pierwsza powoduje, że w obsłudze przerwanie *timera 0* przestaje być testowany koder, funkcja drugiej jest oczywista. Następnie zmienna *B_wynik* jest zmieniana w zakresie od 0 do *B_wybrana_kostka - 1* (na przykład



Rys. 1 Schemat ideowy

Listing 3

```
If B_licznik_wierszy = 0 Then
    If B_stalosc_kostki <> 0 Then
      Decr B_stalosc_kostki
      If B_stalosc_kostki = 0 Then Set
                        Bit_kostka_zmieniona
    End If
    If Bit_losowanie_trwa = 0 Then
      Gosub Przelacznik_do_acc
      cjne A, {B_ostatnie_ustawienie},
                        Zmien_ustawiona_kostke
          'Nie wiem czy if nie niszczy acc?
      sjmp Nie_zmieniaj_kostki
      !Zmien_ustawiona_kostke:
      Mov {B_ostatnie_ustawienie}, A
      Gosub Ustaw_nowa_kostke
      B_stalosc_kostki =
                  Const_opoznienie_wyboru
      !Nie_zmieniaj_kostki:
    End If
  Else
(... Obsługa wyświetlacza)
```

Listing 4

```
Losowanie:
  B_wynik = 0
  Set Bit_losowanie_trwa
  Set Bit_animacja
   If Bit_wylacz_dzwiek = 0 Then Gosub
Dzwiek_start      'Jeśli stukać to inicjacja
                                  dźwięku
  Do
    Incr B_wynik
     If B_wynik = B_wybrana_kostka Then
B_wynik = 0
  Loop Until Losuj = 1
  Gosub Wyswietl_wylosowana_wartosc
  Reset Bit_animacja
  Gosub Dzwiek_stop
  Dim Temp_for As Byte
  B_zmienna_dzwieku = 0
  If Bit_wylacz_dzwiek = 0 Then Set
        Const_bit_glosnika      'Czy stukać?
  For Temp_for = 1 To 25 Step 1
                    'Pierwszy etap zwalniania
    Incr B_wynik
     If B_wynik = B_wybrana_kostka Then
B_wynik = 0
    Waitms Temp_for
    Waitms Temp_for
    Mov A, {B_zmienna_dzwieku}
    Xrl P1, A 'i stuk jeśli zmienna dzwieku
                    odpowiednio ustawiona
    Gosub Wyswietl_wylosowana_wartosc
  Next
  For Temp_for = 1 To 240 Step 40
                                'Drugi etap
    Incr B_wynik
     If B_wynik = B_wybrana_kostka Then
B_wynik = 0
    Waitms Temp_for
    Waitms Temp_for
    Mov A, {B_zmienna_dzwieku}
    Xrl P1, A 'stuk jeśli zmienna
                    dzwieku odpowiednio ustawiona
    Gosub Wyswietl_wylosowana_wartosc
  Next
  Reset Glosnik      'Aby głośnik nie
                        pobierał więcej prądu
  Gosub Mignij_3x
  Reset Bit_losowanie_trwa
  Reset Bit_kostka_zmieniona      'Aby
uniknąć wyświetlania typu kostki zaraz po
losowanu'gdy użytkownik nie poczekał po
jej zmianie przed rozpoczęciem losowania
  Erase Temp_for
  Goto Petla_glowna
```

dla k6: 0-5), aż do puszczenia przycisku. Od tego momentu wyświetlacz wyświetla już wyniki, ponieważ zmienna *Bit_animacja* jest zerowana i program przechodzi do zwalniania losowania: Zostało ta zrealizowane w dwóch krokach w pętlach *For... Next* za pomocą instrukcji *Waitms*. W pierwszym kroku losowanie zwalnia bardzo niewiele, a w drugim zwalnia już dość mocno. Taki sposób zwalniania został dobrany na drodze eksperymentów, ponieważ po prostu ładnie się prezentuje. Po zakończeniu obu pętli niepotrzebna już zmienna *Temp_for* jest usuwana, a wynik jest wyświetlany i miga trzykrotnie. Ostatecznie bit *Bit_losowanie_trwa* jest zerowany i wykonywany jest skok spowrotem do pętli głównej. Całe zwalnianie trwa mniej więcej 3 sekundy. **Uwaga:** przy analizie tej części programu należy pamiętać, że opóźnienie instrukcji *Waitms* jest bazowane na 12MHz rezonatorze i ponieważ w układzie zastosowano rezonator 6MHz, to generowane opóźnienia będą 2 razy dłuższe.

W pliku Dane kostki.bas znajdują się odpowiednie komentarze ułatwiające stworzenie własnej kostki, dlatego mam nadzieje, że jest to czynność dość prosta.

Osoby zainteresowane dokładniejszym poznaniem programu zachęcam do analizy kodu źródłowego, którą powinny ułatwić dość bogate komentarze.

Montaż i uruchomienie:

Montaż przeprowadzamy w trochę nietypowy sposób. Zaczynamy od jednej zworki i dwóch połączeń „kabelkowych", których ze względu na zastosowanie jednostronnej płytki drukowanej nie udało mi się uniknąć. Teraz następuje właśnie ten nietypowy moment: Wszystkie rezystory oraz diody D1,D4,D5 montujemy od strony druku. Należy zwrócić szczególną uwagę na polaryzacje tych ostatnich. Dalszą część montażu przeprowadzamy w typowy sposób: zaczynając od elementów o najmniejszych gabarytach, a kończąc na największych.

Wykaz elementów

Rezystory

R1-R5 ..... 3,3-4,7kΩ
R13 ..... 68Ω
(w układzie nie przewidziano R6-R12)

Kondensatory

C1,C2 ..... 20-40pF
C3 ..... 4,7µF/16V
C4 ..... 100µF/16V

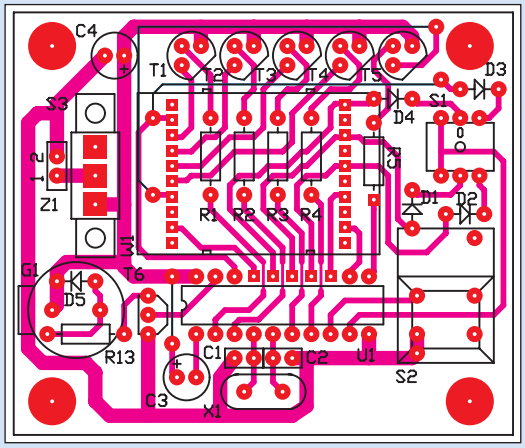
Półprzewodniki

U1 ..... AT89C2051
T1-T5 ..... BC558C
T6 ..... BS170
D1-D5 ..... 1N4148

Różne

W1 ..... wyświetlacz LED matrycowy 5x7, wspólna anoda
G1 ..... miniaturowy głośniczek
S1 ..... koder BCD
S2 ..... mikroprzełącznik z przyciskiem
S3 ..... przełącznik bistabilny do druku
X1 ..... rezonator kwarcowy 6MHz

Płytka drukowana modułu jest dostępna jako kit szkolny AVT-2491



Rys. 2 Schemat montażowy

Układ nie został zaprojektowany pod kątem umieszczenia go w konkretnej obudowie. Proponuję zamontować go wraz z pojemnikiem na baterie na jakiejś sztywnej podkładce – mnie osobiście taki wygląd układu odpowiada, ale bardziej ambitni mogą dobrać odpowiednią obudowę we własnym zakresie.

Kostka po zmontowaniu nie wymaga żadnego uruchomienia i jest od razu gotowa do zabawy.

Radosław Koppel

Uwaga! Pliki z programem można ściągnąć ze strony internetowej EdW www.edw.com.pl/library/pliki/kostkaRK.zip.