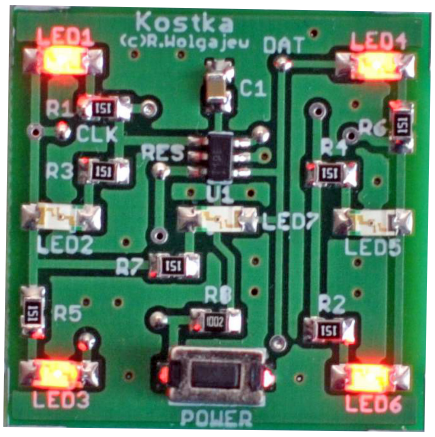
Kostka do gry

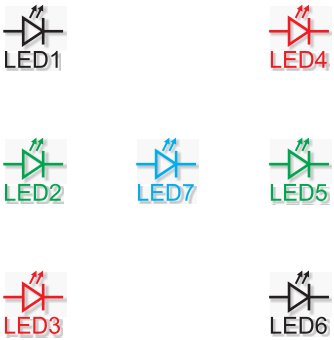


Przeglądając zasoby Internetu, natrafiłem ostatnio na bardzo prosty, a zarazem niezmiernie ciekawy projekt elektronicznej kostki do gry. Nie mam w swoim dorobku żadnego urządzenia przeznaczonego do szeroko rozumianej gry, więc postanowiłem skonstruować własną wersję kostki. To z pozoru bardzo łatwe zadanie postanowiłem nieco skomplikować, stosując najmniejszy, dostępny mikrokontroler AVR w 6-nóżkowej obudowie, który, poza wyprowadzeniem RESET i zasilania, udostępnia jedynie trzy porty wejścia/wyjścia. Tego rodzaju założenia spowodowały konieczność wysterowania 7 diod LED, bo tyle potrzeba do pokazania wszystkich kombinacji wylosowanej liczby „oczek" kostki, za pomocą wyłącznie 3 portów I/O.

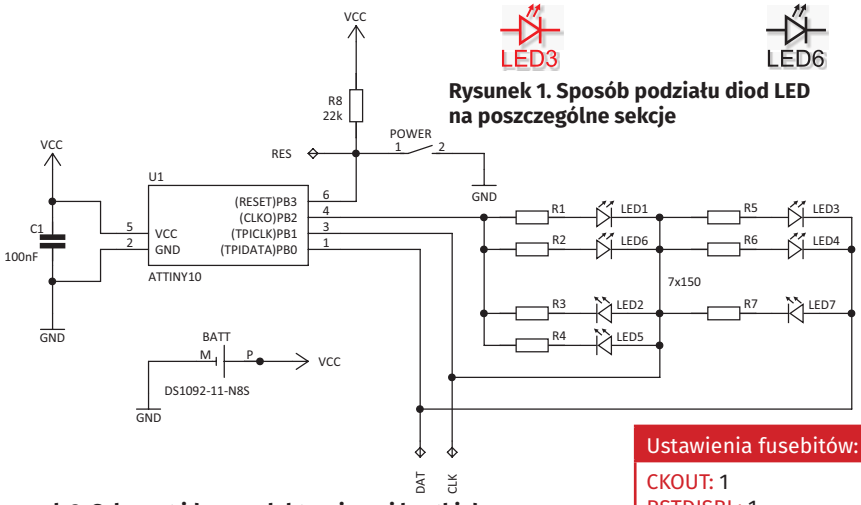
Rekomendacje: *użyteczny gadżet, przy którego budowie można nauczyć się, jak sterować diodami LED mając do dyspozycji niewiele wyprowadzeń mikrokontrolera.*

Zasadę działania kostki, o której mowa we wstępie, pokazano na **rysunku 1**. Jak widać, wyodrębniono następujące sekcje: **LED1/LED6**, **LED3/LED4**, **LED2/LED5** i **LED7**, przy czym w ramach każdej sekcji diody połączone są równolegle (poza sekcją LED7, gdzie występuje wyłącznie jedna dioda LED). Po drugie i najważniejsze, wspomniane sekcje diod LED połączono w dość specyficzny sposób, korzystając z topologii nazywanej charlieplexing, opracowanej w latach 90. przez Charliego Allena z firmy Maxim Integrated. W założeniach swoich metoda ta pozwalała na sterowanie pracą matryc diod LED przy użyciu techniki multipleksowania oraz ograniczonej liczby trójstanowych portów I/O. W naszym przypadku nie będziemy korzystać z możliwości ustawienia portu I/O w tryb wysokiej impedancji, tylko ograniczymy się do dwóch stanów logicznych. Sposób konfiguracji wspomnianych wcześniej sekcji diod LED pokazano na schemacie na **rysunku 2**. Jak widać, jest to nieskomplikowany system mikroprocesorowy, którego sercem jest, najmniejszy z rodziny AVR, mikrokontroler ATtiny10 realizujący całą, założoną funkcjonalność. Podstawowym zadaniem wspomnianego układu jest sterowanie grupą 4 sekcji diod LED, czego dokonuje poprzez ustawienie odpowiednich stanów logicznych na wyprowadzeniach PB2…PB0, których kombinacje pokazano w **tabeli 1**.

Uważny czytelnik z łatwością zauważy, że w tego rodzaju układzie połączeń nie ma możliwości wyświetlenia wszystkich kombinacji wylosowanego zestawu „oczek" naszej elektronicznej kostki, gdyż nie istnieje sposób na jednoczesne załączenie dwóch lub więcej, wybranych kombinacji sekcji diod LED, które utworzyłyby wspomnianą liczbę „oczek". Jak poradzić sobie z tym problemem? Najprościej, jak się da…stosując multipleksowanie poszczególnych sekcji diod LED, które umożliwi z kolei wyświetlenie wynikowej liczby „oczek". W **tabeli 2** przedstawiono



Rysunek 1. Sposób podziału diod LED na poszczególne sekcje



Rysunek 2. Schemat ideowy elektronicznej kostki do gry

Ustawienia fusebitów:
CKOUT: 1
RSTDISBL: 1

DODATKOWE MATERIAŁY DO POBRANIA ZE STRONY:
www.media.avt.pl

W ofercie AVT*
AVT-5624

Podstawowe informacje:
- Napięcie zasilania: 3 V (bateria CR2032).
- Maksymalny prąd obciążenia (włączony/tryb power-down): 5 mA/0,1 µA
- Mikrokontroler ATtiny10 (w 6-nóżkowej obudowie!).

Projekty pokrewne na www.media.avt.pl:

AVT-3124	Kostka do gry (EdW 6/2015)
AVT-1713	Kostka RGB (EP 11/2012)
AVT-1661	Elektroniczna kostka do gry (EP 1/2012)
AVT-5028	Elektroniczna gra w kości (EP 8/2001)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] płytka drukowana bez elementów i dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają następujące dodatkowe wersje:
- wersja [**A+**] płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***

Tabela 1. Kombinacje stanów logicznych na wyprowadzeniach PB2...PB0 mikrokontrolera odpowiadające załączeniu poszczególnych sekcji diod LED

Aktywna sekcja	PB2	PB1	PB0
LED1/LED6	1	0	0
LED3/LED4	1	1	0
LED2/LED5	0	1	1
LED7	0	0	1

Wykaz elementów:
Rezystory: (SMD 0805)
R1...R7: 150 Ω
R8: 22 kΩ
Kondensatory: (SMD 0805)
C1: 100 nF
Półprzewodniki:
U1: ATtiny10 (SOT-23-6)
LED1...LED7: czerwona dioda LED (obudowa 1206)
Inne:
BATT: koszyczek baterii CR2032 typu CONNFLY DS1092-11-N8S
POWER: mikroprzełącznik SMD typu NINIGI TACTM-34N-F

kombinacje sekcji diod LED odpowiadające wyświetleniu poszczególnych liczb wylosowanych „oczek". Stan 1 oznacza załączenie poszczególnej sekcji w procesie multipleksowania, zaś stan 0 oznacza wyłączenie tej sekcji.

Tabela 2. Kombinacje sekcji diod LED odpowiadające wyświetleniu poszczególnych liczb wylosowanych „oczek"

	Załączone sekcje diod LED			
Liczba oczek	LED1/LED6	LED3/LED4	LED2/LED5	LED7
1	0	0	0	1
2	1	0	0	0
3	1	0	0	1
4	1	1	0	0
5	1	1	0	1
6	1	1	1	0

Podczas multipleksowania są zaświecane kolejno od 1 do 3 sekcji diod LED w zależności od docelowej liczby „oczek" jaka ma zostać pokazana. Prawda, że proste? A do tego angażuje wyłącznie 3 dostępne wyprowadzenia mikrokontrolera do wysterowania aż (!) 7 diod LED. Pozostaje kwestia „przypadkowości" procesu losowania liczby oczek oraz oczekiwanego, niskiego poboru mocy z niewielkiej baterii zasilającej typu CR2032. Pierwszy z problemów rozwiązano w sposób sprzętowy, a mianowicie przez wykorzystanie wbudowanego w mikrokontroler przetwornika ADC. Jak wiadomo, każdy pomiar wykonywany przy użyciu tegoż peryferium obarczony jest pewnym błędem, którego źródło związane jest z zakłóceniami, jakie mogą się pojawiać na wejściu pomiarowym ADC, a które materializują się w postaci szumu, powodując zmiany wartości na najmłodszych bitach wyników pomiaru. To, w większości, niekorzystne zjawisko wykorzystamy do skonstruowania prostego, lecz skutecznego generatora liczb losowych. Zwyczajnie dokonamy 8 pomiarów dowolnego, „wiszącego w powietrzu" kanału A/C, przesuwając każdorazowo wartość poprzedniego wyniku pomiaru o jedno miejsce w lewo i dokonując na nim operacji XOR z wynikiem poprzednim. W ten sposób dostaniemy losową wartość pomiaru, którą wykorzystamy, po odpowiednim skalowaniu, do pokazania

Listing 1. Definicje tablic ułatwiających manewrowanie stanami portów PB2…PB0 w odniesieniu do losowanych wyników liczby „oczek" kostki

```
//Definicje stanów portu B odpowiadające zapaleniu poszczególnych segmentów
#define SEG_1_6 0b100
#define SEG_3_4 0b110
#define SEG_2_5 0b011
#define SEG_7 0b001

//Tablica przechowująca definicje odpowiadające poszczególnym segmentom kostki
const uint8_t Segments[4] PROGMEM = {SEG_1_6, SEG_3_4, SEG_2_5, SEG_7};
//Tablica przechowująca kombinacje bitów odpowiadające liczbie oczek kostki (zapaleniu segmentów)
//Bity odpowiadają następującym segmentom: SEG_7, SEG_2_5, SEG_3_4, SEG_1_6
const uint8_t Points[6] PROGMEM = {0b1000, 0b0001, 0b1001, 0b0011, 0b1011, 0b0111};
```

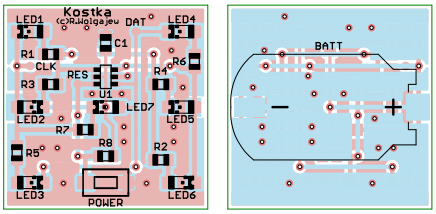
Listing 2. Konfiguracja licznika Timer0 (wraz ze stosownym przerwaniem) odpowiedzialnego za mechanizm multipleksowania sekcji diod LED

```
//Uruchomienie Timera0 odpowiedzialnego za multipleksowanie diod LED
TCCR0B = (1<<WGM02)|(1<<CS01)|(1<<CS00);          //Tryb CTC, preskaler = 64
OCR0A = 77;                                       //Przerwanie wywoływane 200 razy na sekundę
TIMSK0 = (1<<OCIE0A);                             //Timer0 Output Compare A Match Interrupt Enable

ISR(TIM0_COMPA_vect)
{
    static uint8_t Bit;
    if(pgm_read_byte(&Points[Dots]) & (1<<Bit)) PORTB = pgm_read_byte(&Segments[Bit]); else PORTB = 0;
    Bit = (Bit+1) & 0x03;
}
```

Listing 3. Funkcja main urządzenia

```
int main(void)
{
    //Wyłączenie komparatora analogowego dla zmniejszenia poboru mocy
    ACSR |= (1<<ACD);
    //Uruchomienie Timera0 odpowiedzialnego za multipleksowanie diod LED
    TCCR0B = (1<<WGM02)|(1<<CS01)|(1<<CS00);              //Tryb CTC, preskaler = 64
    OCR0A = 77;                                           //Przerwanie wywoływane 200 razy na sekundę
    TIMSK0 = (1<<OCIE0A);                                 //Timer0 Output Compare A Match Interrupt Enable
    //Uruchomienie przetwornika ADC, preskaler=8 (125kHz)
    ADCSRA = (1<<ADEN)|(1<<ADPS1)|(1<<ADPS0);
    //Wykonanie 8 pomiarów na domyślnym i niepodciągniętym do VCC kanale ADC
    for(uint8_t i=0; i<8; ++i) Dots = (Dots<<1) ^readVoltage();
    //Zawężenie zakresu zmiennej do 0...5
    Dots = Dots % 6;
    DDRB = 0xFF;                                          //PortB, jako wyjściowy z domyślnym stanem 0x00
    //Zezwolenie na obsługę przerwań a tym samym wyświetlenie (w funkcji ISR) wylosowanej liczby oczek (przez 5s)
    sei(); _delay_ms(5000);
    TIMSK0 = 0x00;                                        //Wyłączenie przerwania multipleksowania
    PORTB = 0x00;                                         //Wyłączenie diod LED
    ADCSRA  = 0x00;                                       //Wyłączenie przetwornika ADC dla zmniejszenia poboru mocy
    //Redukcja poboru mocy przez wyłączenie zasilania nieużywanych modułów mikrokontrolera
    PRR = (1<<PRTIM0)|(1<<PRADC);
    //Uśpienie mikrokontrolera (do czasu użycia przycisku POWER powodującego zresetowanie mikrokontrolera)
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_enable();
    sleep_cpu();
    while(1){}
}
```



Rysunek 3. Schemat montażowy elektronicznej kostki do gry

wylosowanej liczby „oczek". Zagadnienie niskiego poboru mocy, tak ważne przy zasilaniu bateryjnym, rozwiążemy, wykonując poszczególne kroki, jak niżej:

- wyłączamy wszystkie, nieużywane peryferie mikrokontrolera,
- ustawiamy wszystkie porty jako porty wyjściowe,
- losujemy wynik gry (pomiar ADC) i pokazujemy go przez 5 s,
- wyłączamy pozostałe, używane dotąd peryferie mikrokontrolera (ADC i Timer0),
- wprowadzamy mikrokontroler w stan uśpienia (tryb power-down), z którego wybudzony być może wyłącznie przez użycie przycisku POWER (czyli, w tym przypadku, reset mikrokontrolera).

W ten sposób realizujemy wymóg niskiego poboru mocy, gdyż tak skonfigurowany

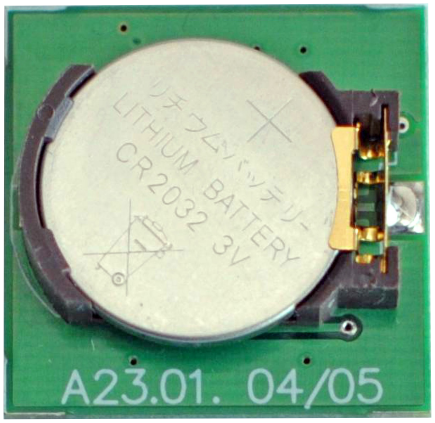
Listing 4. Funkcja odpowiedzialna za pomiar napięcia przetwornika A/C

```
inline uint8_t readVoltage(void)
{
    ADCSRA |= (1<<ADSC);                    //Uruchomienie konwersji
    while(ADCSRA & (1<<ADSC));              //Czekamy na zakończenie konwersji - ok. 116us
    return ADCL;
}
```

mikrokontroler w trybie uśpienia pobiera prąd rzędu 0,1 µA! Na koniec przedstawię kilka szczegółów implementacyjnych związanych z aplikacją sterującą. Na początek definicje tablic ułatwiających manewrowanie stanami portów PB2…PB0 w odniesieniu do losowanych wyników „oczek" kostki i samego mechanizmu multipleksowania. Wspomniane definicje przedstawiono na **listingu 1**. Na **listingu 2** pokazano z kolei konfigurację licznika Timer0 (wraz ze stosownym przerwaniem) odpowiedzialnego za mechanizm multipleksowania sekcji diod LED. Na koniec, na **listingu 3**, przedstawiono funkcję main realizujące całą, założoną funkcjonalność urządzenia. Dla porządku, na **listingu 4** zamieszczono funkcję readVoltage() odpowiedzialną za wykonanie pomiaru napięcia na wybranym kanale A/C.

Montaż

Na **rysunku 3** pokazano schemat montażowy urządzenia. Zaprojektowano niewielki i bardzo zwarty obwód drukowany przeznaczony do montażu powierzchniowego. Montaż urządzenia rozpoczynamy od warstwy TOP, gdzie



w pierwszej kolejności wlutowujemy wszystkie półprzewodniki, następnie elementy bierne a na samym końcu mikroprzełącznik POWER. W tym momencie przechodzimy na warstwę BOTTOM, gdzie wlutowujemy gniazdo baterii CR2032, po czym w gnieździe umieszczamy samą baterię zasilającą. Poprawnie zmontowany układ powinien działać po włączeniu zasilania.

Robert Wołgajew, EP