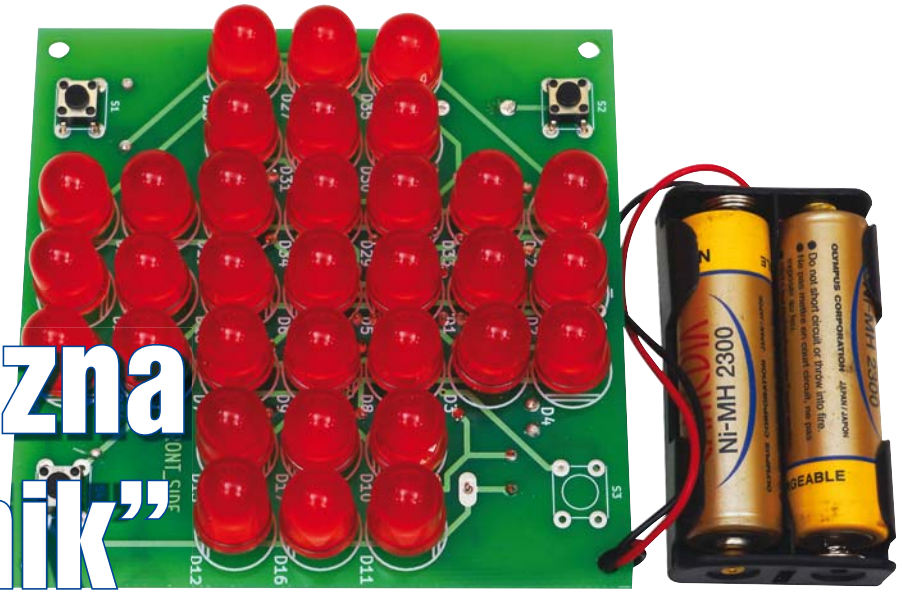
kit 3037 AVT

Gra logiczna „Samotnik”



Do czego to służy?

Jest to realizacja gry „Samotnik” w postaci układu elektronicznego. Dla nieznających zasad tej gry, przedstawiam krótkie ich wyjaśnienie. Zazwyczaj gra toczy się na planszy o trzydziestu czterech polach. Na początku gry, na trzydziestu trzech leżą pionki, natomiast jedno jest puste – środkowe. Gra polega na usunięciu wszystkich pionków z planszy; kiedy na planszy zostaje tylko jeden pionek, gra kończy się wygraną. Pionki z planszy można usunąć, „zbijając” je innym pionkiem, czyli po prostu przeskakując nad zbijanym pionkiem innym pionkiem. Ilustracja prezentująca zasady gry jest na **rysunku 1**. Widoczna jest sytuacja z początku gry z czterema możliwymi ruchami:

cztery pionki na polach oznaczonych jako „M” mogą zbić odpowiednie pionki na polach „C”. Pionki z pól „M” podczas zbijania wykonują skok nad pionkami „C”, lądując na polu „D”. Po zbiciu pionek z pola „C” jest usuwany.

O grze można się dowiedzieć więcej z wielu źródeł; jest ona znana również pod nazwą „*Peg solitaire*”.

W opisywanej tutaj wersji elektronicznej plansza gry jest zrealizowana jako wyświetlacz LED. Najważniejsze cechy urządzenia to:

- zasilanie od 2,0V do 5,0V – czyli wystarczą dwie baterie/akumulatorki AA(A),
- obsługa gry za pomocą 3 przycisków; w tym 2 przyciski na sterowanie kursorem,
- cofanie ostatniego zbicia,
- restart gry.

Jak to działa?

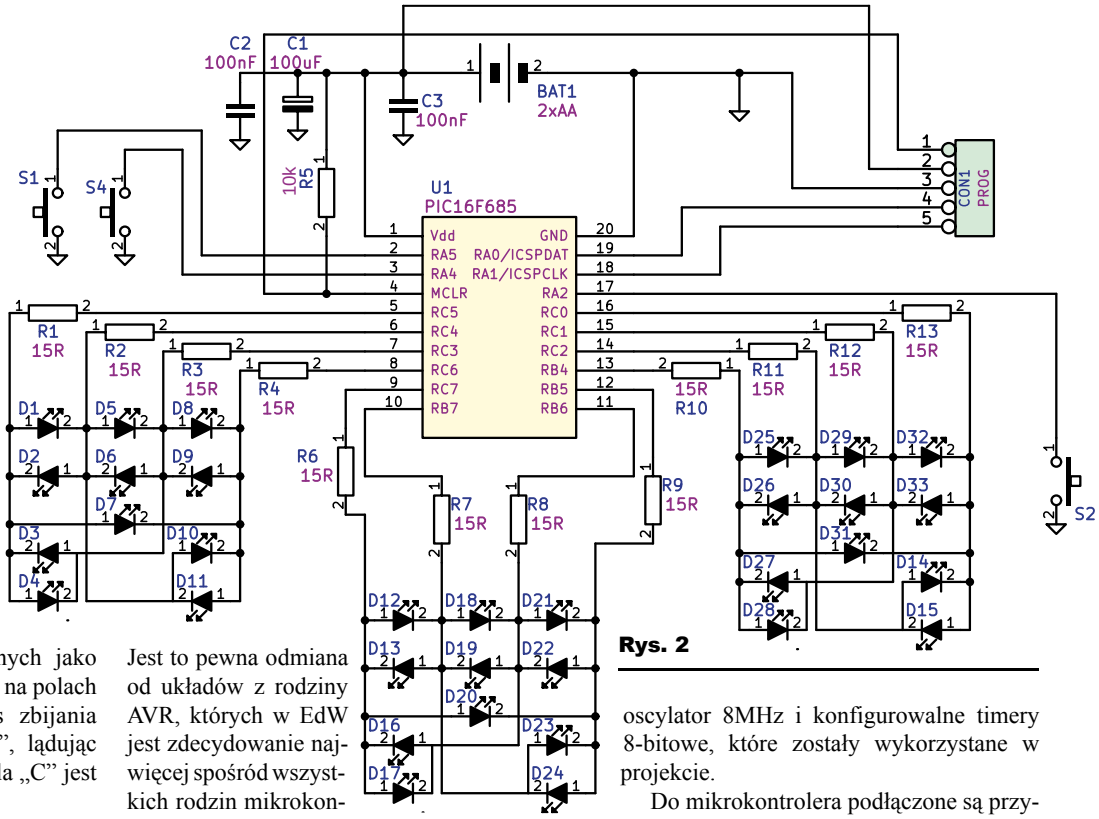
Jak pokazuje **rysunek 2**, gra została zaimplementowana na mikrokontrolerze PIC16F685. Jest to pewna odmiana od układów z rodziny AVR, których w EdW jest zdecydowanie najwięcej spośród wszystkich rodzin mikrokontrolerów. Wybór na ten układ padł również ze względu na minimalne napięcie zasilania, wynoszące 2,0V (możliwość zasilania dwiema bateriami paluszkami), a także niską cenę, łatwość dostępu oraz zasoby, pozwalające komfortowo napisać program w języku C.

Mikrokontroler ten ma m.in. wewnętrzny oscylator 8MHz i konfigurowalne timery 8-bitowe, które zostały wykorzystane w projekcie.

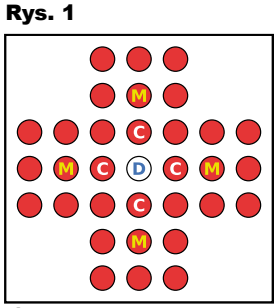
Do mikrokontrolera podłączone są przyciski S1..S3, a ich stan stanowi źródło danych sterujących przebiegiem gry. Są one podłączone do portów skonfigurowanych jako wejścia, podciągnięte do plusa zasilania.

Stan gry prezentowany jest na wyświetlaczu złożonym z diod LED D1...D33. Patrząc na połączenia między diodami LED oraz mikrokontrolerem, od razu widać niestandardowy sposób ich sterowania.

Wyświetlacz został zrealizowany w technice „charlieplexing” z podziałem na trzy sekcje składające się z jedenastu diod każda: D1...D11, D12...D22



Rys. 2



Rys. 1

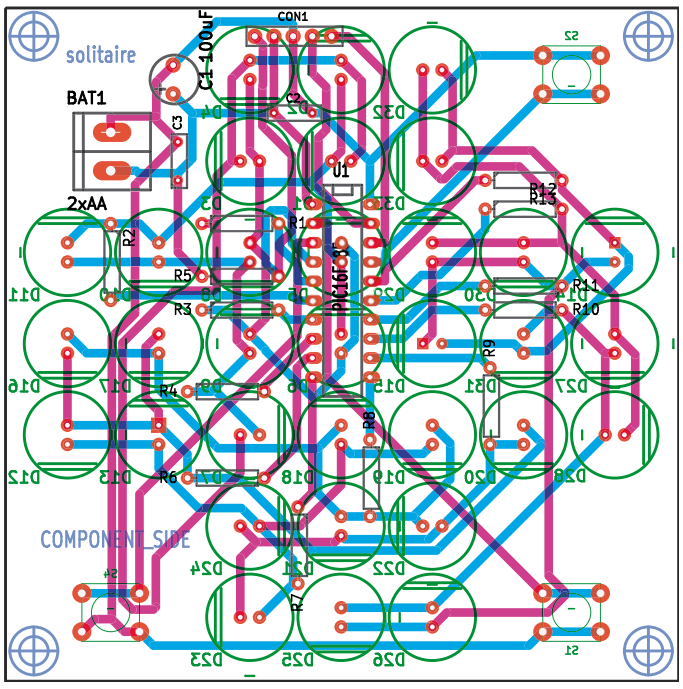
Puste pole
Pole z pionkiem
Pole z pionkiem zbijającym (skaczącym)
Pole z pionkiem zbijanym
Pole na którym stanie pionek po skoku

oraz D23...D33. Sterowanie diodami odbywa się programowo, z wykorzystaniem trzech możliwych stanów portów sterujących układu U1. Sposób sterowania diodami w każdej z sekcji polega na przydzieleniu slotu TI w pewnym czasie TD (około 0,03s) dla każdej z nich. Ponieważ w każdej sekcji jest 11 diod, każda z nich ma slot równy TD/11. W tym czasie dana dioda może być zaświecona lub zgaszona poprzez odpowiednie ustawienie portów, które nią sterują. Ustawienie portów dla danej diody polega na programowym skonfigurowaniu dwóch portów w sekcji, połączonych z daną diodą, jako wyjście i pozostałych dwóch w stan Hi-Z. Gdy dioda ma być zaświecona, dwa porty sterujące daną diodą mają przeciwne stany logiczne zgodnie z polaryzacją diody.

Podział na trzy sekcje został podyktowany chęcią maksymalnego wykorzystania liczby wyprowadzeń dostępnych w mikrokontrolerze oraz wydłużeniem czasu TI dla każdej z diod. Od czasu TI bowiem zależy subiektywna jasność świecenia diody, postrzegana przez ludzkie oko.

Rezystory R1...R4, R6...R13 mają na celu ograniczyć prąd portów do bezpiecznej wartości 20mA. W zależności od napięcia użytego źródła zasilania oraz koloru diod, wartość rezystorów należy indywidualnie dobrać, zgodnie ze wzorem: R=(Uz-Ud)/Id, gdzie: Uz – napięcie zasilania układu, Ud – napięcie na diodzie, Id – maksymalny prąd portu = 20mA. Zmianę wartości na inną niż na schemacie należy rozważyć zwłaszcza podczas zasilania układu napięciem powyżej 4V.

Rys. 3




```
void setLed(u8 n, u8 s) {
        dmasq[lmap[n]] = (dmasq[lmap[n]] != s) ? s : dmasq[lmap[n]];
}

void showMatrix(u8 * mtx, u8 ptr) {
        u8 i;
        for(i = 0; i < 33; i++) {
                if(i == ptr)
                        setLed(i, LED_STATE_2);
                else
                        setLed(i, *(mtx + i));
        }
}
```

Listing 2

```
if(++led_in_sec >= 11)
        led_in_sec = 0;

PORTB = MASQ(0, led_in_sec);
PORTC = MASQ(1, led_in_sec);
TRISB = ~(MASQ(2, led_in_sec));
TRISC = ~(MASQ(3, led_in_sec));
```

Listing 1

Natomiast samo napięcie zasilania układu jest zdeterminowane katalogowym zakresem napięcia zasilania U1 i może wynosić od 2V do 5V.

Program został napisany w języku C i skompilowany za pomocą kompilatora firmy HI-TECH, przy użyciu środowiska MPlab firmy Microchip.

Struktura programu jest wyraźnie rozdzielona na dwie warstwy:

1. Implementacja gry (silnik gry) – reagowanie na logiczną zmianę stanów przycisków, zmiana stanu planszy jako odpowiedź na posunięcia gracza, przesuwanie pionka, sprawdzanie wyniku gry;
2. Obsługa interfejsu użytkownika – sprawdzanie fizycznego stanu przycisków, generowanie zdarzeń przycisków, wyświetlanie planszy gry na wyświetlaczu LED, obsługa przerwań.

Rozdział kodu na warstwy był tutaj szczególnie pomocny, gdyż kod gry został napisany i przetestowany na komputerze PC. Gotowy kod samej gry wystarczyło połączyć z częścią programu, generującą zdarzenia od przycisków i wyświetlającą stan gry na wyświetlaczu.

Część obsługująca klawisze i wyświetlacz została zaimplementowana w przerwaniu od przepełnienia timerów T0 i T2.

Fragment kodu odpowiedzialnego za wyświetlanie planszy widoczny jest na **listingu 1**. Jest on wykonywany z każdym przepełnieniem T0, około 355 razy na sekundę. We fragmencie tym, w każdej z trzech sekcji wyświetlacza zostaje obsłużona jedna z 11 diod, zaświecona lub zgaszona.

Na **listingu 2** widoczne są funkcje wykorzystywane podczas wyświetlania planszy. Zaprezentowany kod funkcji *showMatrix()* ukazuje, jak zrealizowane jest sprawdzanie położenie kursora na planszy.

Stan przycisków jest badany w przerwaniu z przepełnienia T2. Na podstawie stanu przycisków oraz kombinacji ich stanów (np. jednoczesne naciśnięcie dwóch przycisków podczas zbijania pionka) generowane są zdarzenia. Obsługa zdarzeń z przycisków zrealizowana jest w nieskończonej pętli głównej programu, co daje namiastkę programu sterowanego zdarzeniami.

Osoby zainteresowane pozostałymi szczegółami odsyłam do zapoznania się z całością kodu źródłowego, dostępnego w Elportalu.

Instrukcja gry

Położenie kursora na planszy jest sygnalizowane miganiem diody, która wskazuje na jego aktualną pozycję na planszy.

Przesuwanie kursora odbywa się z użyciem dwóch przycisków odpowiednio:
ruch w poziomie: przycisk S4,
ruch w pionie: przycisk S1,
w sytuacji, kiedy trzymamy palec lewej ręki na S1; prawej na S4.

Aby zmienić kierunek ruchu kursora (poziomy lub pionowy) na przeciwny, należy przytrzymać około sekundy przycisk odpowiedni do osi ruchu.

Aby „zbić" dany pionek, należy przytrzymać przycisk odpowiadający kierunkowi zbijania i nacisnąć drugi przycisk służący do przesuwania w drugiej osi, np. aby zbić w poziomie, zachodząc pionek od lewej strony, należy ewentualnie (jeśli zachodzi potrzeba zmiany kierunku) ustawić ruch lewa->prawa za pomocą S4, następnie nacisnąć go i trzymając, jednocześnie wcisnąć S1. Oczywiście zbić można tylko pionek, za którym jest puste pole, a przed nim jest inny pionek, który go zbija.

Każde ostatnio wykonane zbicie pionka, z wyjątkiem kończącego grę, można cofnąć – gra powraca do stanu przed ostatnim zbiciem. Można dokonać tego, naciskając krótko S2.

Dłuższe naciskanie S2 restartuje grę, bez możliwości powrotu do zamkniętej planszy.

Montaż i uruchomienie

Układ został zmontowany na płytce, pokazanej na **rysunku 3,** z obustronnym drukiem, a także obustronnym montażem elementów. Pytka została zaprojektowana w programie KiCad za pomocą autoroutera FreeRouting ze strony *freerouting.net*. Autorouter okazał się bardzo pomocny ze względu na złożone połączenia pomiędzy diodami LED oraz U1.

Wykaz elementów

Rezystory

R1-R4,R6-R13 . . . . . 15Ω
R5 . . . . . 10kΩ

Kondensatory

C1 . . . . . 100uF/16V
C2,C3 . . . . . 100nF

Półprzewodniki

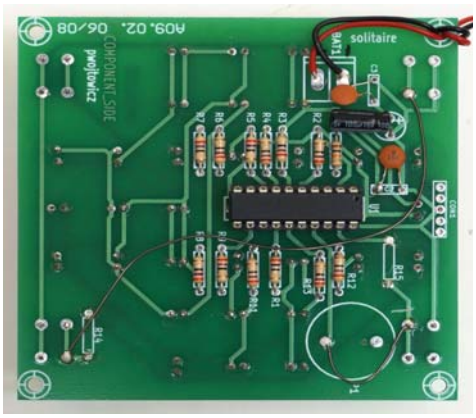
U1 . . . . . PIC16F685 (zaprogramowany)
D1-D35 . . . . . LED RED 10mm

Inne

Podstawka 20dip
koszyczek na baterie 2xAA
3 x microswitch 2mm

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-3037.

Z uwagi na specyficzne ułożenie elementów, zalecana jest odpowiednia kolejność przy montażu. Na początku należy przygotować odpowiednią podstawkę na U1. W tym celu można użyć standardowej, najlepiej precyzyjnej, podstawki DIP18, wycinając plastikowe łączenia spomiędzy obu rzędów pinów tak, aby te rzędy rozdzielić fizycznie. Można do tego również użyć listwy z gniazdami do pinów. Kolejność taka jest potrzebna do tego, aby po wlutowaniu podstawki można było



lutować diody LED pomiędzy rzędami pinów U1. Następnie lutujemy rezystory, kondensatory oraz przyciski. Na końcu lutujemy diody LED. Zakreskowana część w zarysie obudowy diody LED na PCB oznacza katodę, czyli krótszą nóżkę elementu. Należy zachować uwagę przy montażu diod, ponieważ wylutowanie którejś z nich może być kłopotliwe i stwarza ryzyko uszkodzenia płytki. Najlepiej na początku przylutować cztery najbardziej oddalone od siebie diody, lutując tylko po jednej nóżce każdej. Diody powinny być dokładnie na takiej samej wysokości w pionie. Po wyprostowaniu każdej z nich można przylutować drugą nóżkę każdej. Cztery wlutowane diody tworzą płaszczyznę odniesienia dla pozostałych, które należy włożyć w otworki dla nich przeznaczone, następnie odwrócić płytkę i pozwolić wszystkim opaść na powierzchnię. Podobnie jak na początku należy przylutować po jednej nóżce każdej, a po wyrównaniu każdej z nich przylutować drugą.

Płytka zawiera dodatkowo złącze do podłączenia programatora. Prototyp był programowany z użyciem programatora PicKit2 i złącze na płytce urządzenia jest kompatybilne z jego układem wyprowadzeń.

Baterie zasilające układ najlepiej umieścić w koszyczku i połączyć z układem. Jeśli układ ma być umieszczony w obudowie, dobrze jest zastosowań mechaniczny wyłącznik zasilania.

Piotr Wójtowicz
pw@elportal.pl