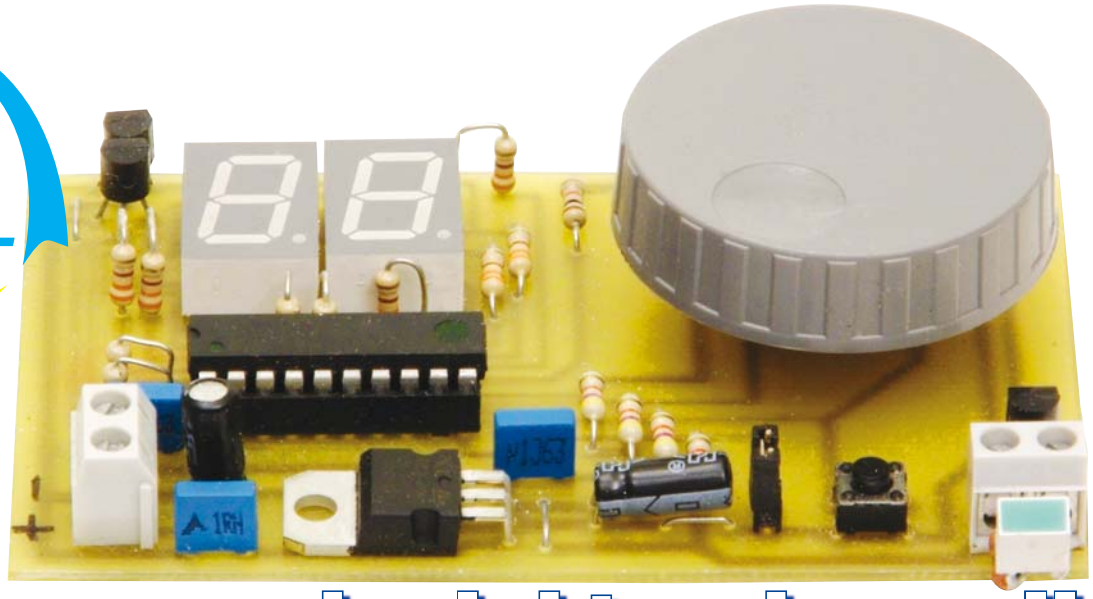
kit 2932 AVT



Sejfowy zamek elektroniczny II

Do czego to służy?

W EdW 1/2007 opublikowany był mój projekt – „Sejfowy zamek elektroniczny". Zainteresowanie tym układem było dość duże, więc postanowiłem wykonać jego kolejną, nieco zmodyfikowaną wersję. Główne ulepszenia i zmiany są następujące:

- wprowadzenie enkodera zamiast potencjometru (co eliminuje konieczność kalibracji, jak było w pierwszym układzie),
- dwucyfrowy, sześcioliczbowy kod (zwiększenie bezpieczeństwa),
- program napisany w C, skompilowany w WinAVR wersja z 25-05-2007.

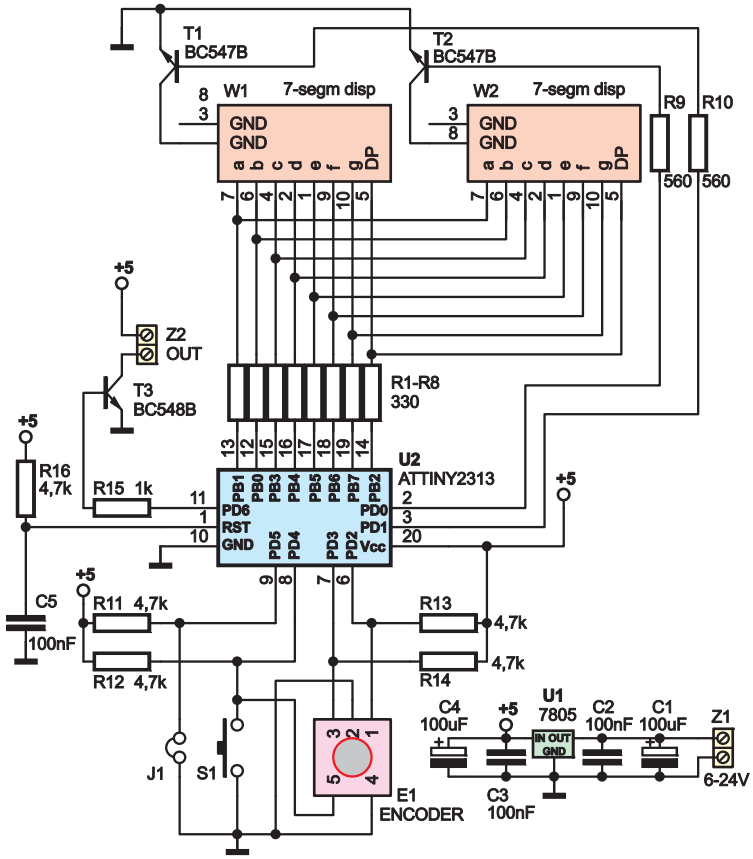
Jak to działa?

Schemat ideowy możemy zobaczyć na **rysunku 1**. Jak to bywa w układach opartych na mikroprocesorze, nie jest on zbyt skomplikowany. Do mikroprocesora ATtiny2313 podłączone są dwa wyświetlacze siedmiosegmentowe LED ze wspólną katodą. Pracują one w trybie multipleksowania. Na łamach EdW wielokrotnie opisywany był taki sposób obsługi wielu wyświetlaczy LED, dlatego nie będę się tu wgłębiał w szczegóły. Korzyść z tego taka, że wykorzystujemy tylko dziesięć pinów procesora zamiast szesnastu.

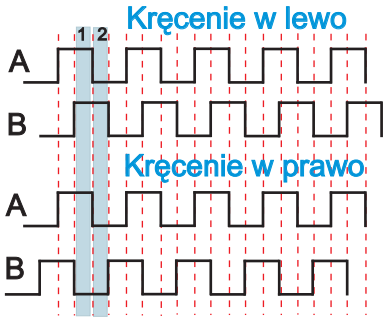
Kolejnym kluczowym elementem układu jest enkoder 28 impulsów/obrotów. Już dawno miałem ochotę, aby wykorzystać ten bardzo ciekawy element w jakimś układzie, ale najpierw nie mogłem go nigdzie dostać, a później nie miałem czasu na przeprowadzenie eksperymentów. Wiem, że wiele osób poszukuje impulsatorów w rozsądnej cenie i że w niektórych miastach są z tym problemy. Ostatecznie udało mi się nabyć dwa różne po cenie poniżej 4zł. Jeden z nich był zwykłym enkoderem w obudowie podobnej do potencjometru. Natomiast drugi – w takiej samej obudowie – miał dodatkowo wbudowany microswitch. Na początku ten drugi wydawał mi się o wiele ciekawszy, ale okazało się, że tak naprawdę to, co miało być zaletą, stało się wadą. Mianowicie podczas wciskania enkodera pojawiają się obroty, które wprowadzają tylko niepotrzebne zamieszanie. Po drugie, nie był on zbyt precyzyjny. W międzyczasie udało mi się jeszcze zbudować enkoder z elementów ze starej myszki komputerowej. Zainteresowanych takimi eksperymentami zachęcam do odwiedzenia strony **http://users.on.net/~merrifield/opto/index.html**. Ostatecznie zdecydowałem się na enkoder bez przycisku, a microswich został usytuowany na płytce. Jeśli ktoś jednak chciałby zastosować taki enkoder z przyciskiem, to jest na niego przewidziane miejsce na płytce.

Teraz łyk teorii na temat zasady działania enkodera. Taki element posiada dwa wyjścia – zazwyczaj oznaczane jako A i B. Podczas obracania pojawiają się na nich przebiegi prostokątne przesunięte w fazie o 90°. To, w którą stronę przesunięty jest jeden przebieg względem drugiego, zależy od tego, w którą stronę kręcimy enkoderem. Taką zasadę działania przedstawia **rysunek 2.** Od strony praktycznej wygląda to następująco: aby dowiedzieć się, w którą stronę kręcony jest enkoder, należy znać stan obu wyjść w dwóch „taktach". Na rysunku 2 zaznaczone są na niebiesko dwa takty: 1 i 2. Rozważmy teraz, jak rozpoznać, że obracanie następuje w lewo. W

Rys. 1



takcie 1 zarówno wyjście A, jak i wyjście B znajdują się w stanie wysokim. Zaczynamy kręcić enkoderem i wyjścia przechodzą w stan 2, gdzie wyjście A jest w stanie niskim, a wyjście B pozostało w wysokim. Takich „przejść" możemy wyróżnić po cztery dla każdego kierunku. Zostało to przedstawione w **tabeli 1.** Sposobów na programową obsługę enkodera jest wiele. Ja postanowiłem stablicować wszystkie osiem możliwości, a następnie porównywać w trakcie trwania programu z aktualnymi danymi. Możemy to zobaczyć na **listingu 1.** Funkcja „ktora_strona()" jest wywoływana za każdym razem, gdy nastąpi zmiana na PIND 2 lub PIND 3, czyli na pinach, do których podłączony jest enkoder. Zmienna nowy_stan w zerowym i pierwszym bicie przechowuje aktualny stan enkodera, natomiast zmienna stary_stan przechowuje w bicie drugim i trzecim stan poprzedni enkodera. Po „scaleniu" tych dwóch danych otrzymujemy informację, którą następnie porównujemy z danymi zapisanymi w tabelach lewo[] i prawo[].

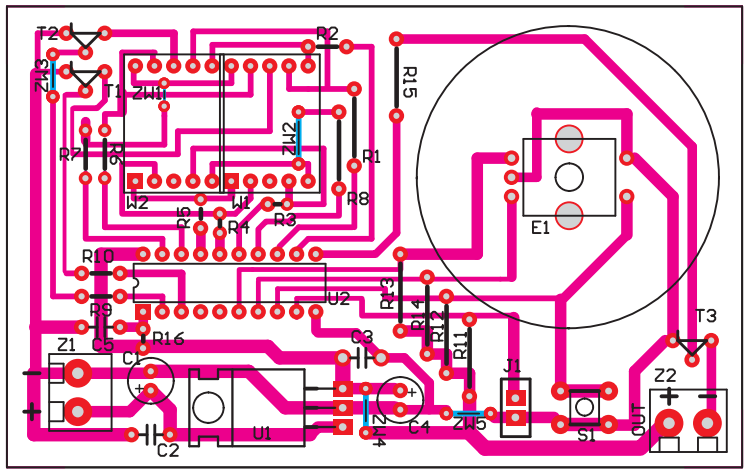


Rys. 2

Tab. 1

1		2		Kierunek
A	B	A	B	
1	1	0	1	lewo
0	1	0	0	lewo
0	0	1	0	lewo
1	0	1	1	lewo
1	0	0	0	prawo
0	0	0	1	prawo
0	1	1	1	prawo
1	1	1	0	prawo

Program jest napisany w sposób, który moim zdaniem jest najbardziej odporny na zawieszenia. Wszystko odbywa się w przerwaniu co około dwie milisekundy. Nie ma w nim żadnych poważniejszych zapętleń. W zależności od tego, w jakim stanie aktualnie znajduje się układ, wybierane są instrukcje do wykonania poprzez polecenie „switch(stan);". Mimo wszystko nad całością czuwa pies łańcuchowy – czyli znany i lubiany watchdog ☺. Stan licznika watchdoga jest zerowany w przerwaniu, czego nie powinno się robić, jednak uważam, że sposób pisania programu to usprawiedliwia. Rezystor R16 podciąga wejście reset do stanu wysokiego, żeby układ był odporny na ewentualne zakłócenia i tym samym przypadkowy reset.



Rys. 3

```
uint8_t lewo[] = {0b1011, 0b1101, 0b0100, 0b0010};
uint8_t prawo[] = {0b0111, 0b1110, 0b1000, 0b0001};
// Funkcja, która rozpoznaje w którą stronę przekręcono enkoder
 i odpowiednio wywołuje funkcję zwieksz_stan() lub zmniejsz_stan()
   void ktora_strona()
   {
   uint8_t nowy_stan = (PIND >> 2) & 0b11;

   for(uint8_t x=0; x<=3; x++)
      {
      if((nowy_stan | stary_stan) == lewo[x]) zmniejsz_stan();
      if((nowy_stan | stary_stan) == prawo[x]) zwieksz_stan();
      }

   stary_stan = PIND & 0b1100;
   }
```

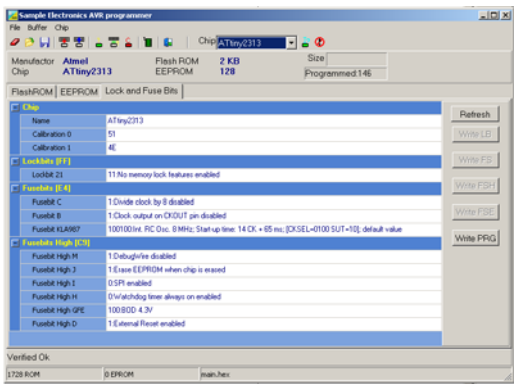
Listing 1

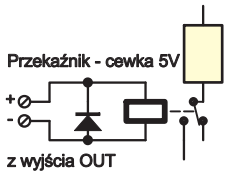
Montaż i uruchomienie

Płytka drukowana (**rysunek 3**) nie jest zbyt skomplikowana, więc nie powinno być problemów z montażem elementów. Warto zacząć od najniższych, czyli zworek, a zakończyć na włożeniu mikroprocesora do podstawki. Wcześniej oczywiście trzeba go zaprogramować programem, który możemy ściągnąć z Elportalu lub z mojej strony **www.moja-elektronika.lua.pl**. Gdy już zaprogramujemy mikroprocesor, nie zapomnijmy o poprawnym skonfigurowaniu fusebitów; najważniejsze – CKSEL3...0: 0100, BODLEVEL:100, WDTON:0, CKDIV8:1. Dodatkowo zamieszczam na **rysunku 4** zrzut z ekranu z BASCOM-owego programatora. Ustawienie fusebitów ma bardzo duże znaczenie dla działania programu, dlatego należy zwrócić szczególną uwagę na prawidłowe ich zaprogramowanie! Szczególnie ważne jest wyłączenie dzielenia taktów kwarcu, ponieważ potrafi to przysporzyć dziwnych problemów.

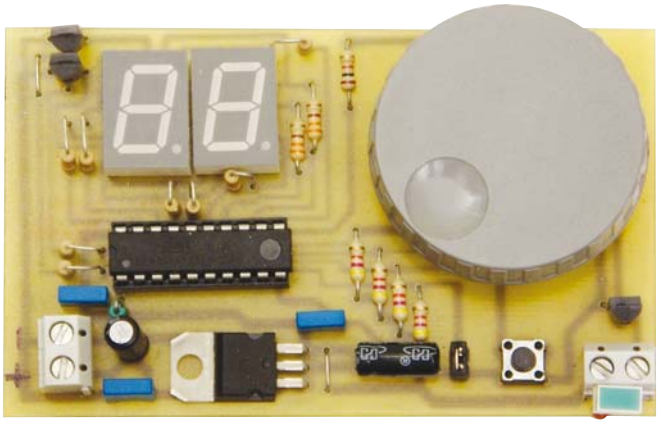
Układ można zasilać napięciem 6–24VDC. Przy napięciach powyżej 10V należy dodać radiator na stabilizator U1. Aby po poprawnym zmontowaniu móc korzystać z „Sejfowego zamka elektronicznego II", trzeba najpierw zaprogramować swój kod. Aby to uczynić, odłączamy układ od zasilania, zakładamy jumperek J1 i trzymając przycisk S1, włączamy zasilanie. Na wyświetlaczu powinno pojawić się „0" oraz powinny zaświecić się obydwie kropki. Teraz, kręcąc enkoderem, wybieramy nasze sześć liczb, każdą potwierdzając przyciskiem S1. Po wpisaniu wszystkich, zgasną kropki na wyświetlaczach i zamek przejdzie w normalny tryb. Oczywiście kod zapisywany jest w pamięci nieulotnej EEPROM, dzięki czemu nawet po odłączeniu zasilania będzie on zapamiętany. Teraz możemy, w ramach testu, wpisać nasz kod, analogicznie jak w trybie programowania, potwierdzając każdą liczbę przyciskiem S1. Jeżeli kod będzie nieprawidłowy, zgaśnie aktualnie wyświetlana liczba i zaświeci się lewa kropka. Taki stan będzie trwał około 12 sekund. Jeśli kod będzie poprawny, to także zgaśnie liczba, ale zaświeci się prawa kropka oraz zostanie podane napięcie 5V na wyjście OUT. To, na jak długo zostanie podane to napięcie, zależy od stanu jumperka J1. Jeśli będzie on założony, będzie to trwało około 4s. Natomiast jeżeli J1 będzie ściągnięty, napięcie będzie

Rys. 4





Rys. 5



dotąd podawane, aż nie zostanie wciśnięty S1. Pamiętajmy jednak, że wydajność prądowa tego wyjścia nie jest zbyt duża, ponieważ pracuje tam tranzystor BC548. Jeżeli ktoś chciałby sterować jakimś większym urządzeniem (np. ryglem elektromagnetycznym), musiałby zastosować przekaźnik, np. według **rysunku 5.**

Należy jeszcze wspomnieć o tym, że jeśli w trybie wpisywania kodu pokrętło nie zostanie poruszone przez około 10 sekund, to zamek przejdzie w tryb uśpienia i zostaną wygaszone wyświetlacze, a prawa kropka będzie cyklicznie migać, sygnalizując gotowość. Aby go „wybudzić", wystarczy poruszyć enkoderem. Jeżeli by się zdarzyło, że pomylimy jedną liczbę, to należy poczekać, aż układ przejdzie w stan uśpienia i zacząć wpisywanie kodu od początku.

Gdyby ktoś miał problem z uruchomieniem układu, niech śmiało pisze na mój e-mail, na pewno będę się starał pomóc. Czekam także na wszelkie uwagi dotyczące układu, bo to między innymi dzięki nim powstała druga wersja sejfowego zamka. Zapraszam także do obejrzenia na mojej stronie www filmiku prezentującego układ.

Możliwości zmian

Po pierwsze, enkoder można pozyskać z jakiegoś sprzętu RTV (np. wieża). Bez zmian w programie lub dodania zewnętrznego dzielnika na pewno nie można zastosować profesjonalnego enkodera, który na jeden obrót daje wiele razy więcej niż 28 impulsów. W pamięci procesora zostało też trochę miejsca, więc można je wykorzystać na dodanie dodatkowych funkcji.

Wykaz elementów

Rezystory

R1-R8	330Ω
R9,R10	560Ω
R11-R14,R16	4,7 kΩ
R15	1kΩ

Kondensatory

C1,C4	100µF
C2,C3,C5	100nF

Półprzewodniki

T1-T3	BC547 lub BC548
U1	LM7805
U2	ATtiny2313
W1,W2	wyświetlacze LED HDSP-516G, wspólna katoda

Pozostałe

J1	jumper
S1	microswitch
E1	enkoder 28imp/obrót
Z1,Z2	złącze śrubowe

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2932.

Marcin Połomski
marcin1326@tlen.pl
www.mojaelektronika.lua.pl