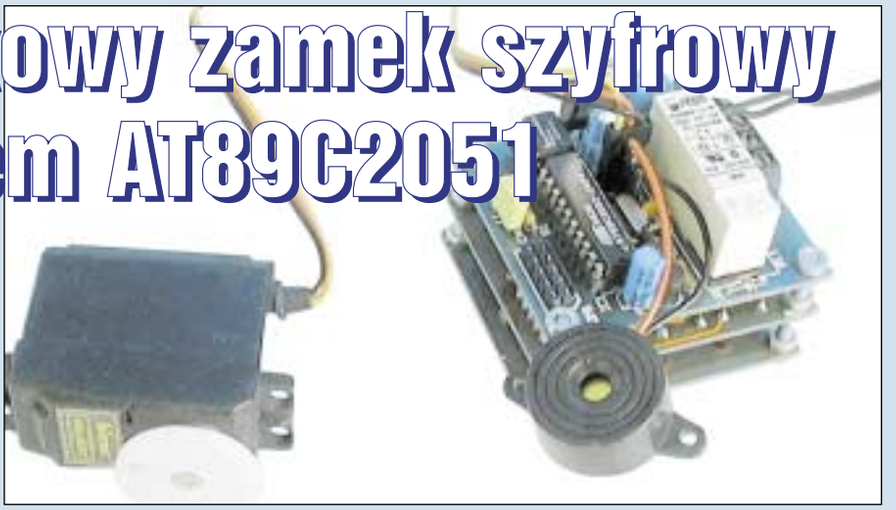
10-przyciskowy zamek szyfrowy z procesorem AT89C2051



Kit 3001 AVT

Tym projektem inaugurujemy serię „µProjekty - 3000“. To wielkie wydarzenie w dziejach EdW, dlatego koniecznie przeczytajcie najpierw „Manifest“ na stronie 17.

Do czego to służy?

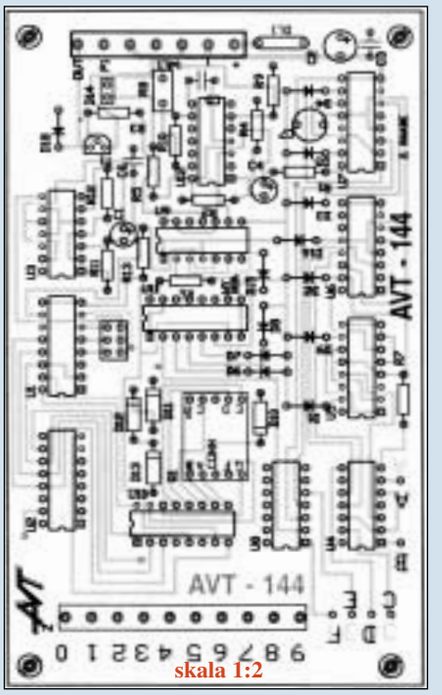
"... Moim zdaniem, stosowanie typowej wieloprzyciskowej klawiatury w nowoczesnej konstrukcji zamka szyfrowego automatycznie dyskwalifikuje takie urządzenie i naraża konstruktora na posądzenie o pójście na łatwiznę, chyba że jego konstrukcja w założeniu miała być jedynie zabawką. Złamanie kodu takiego zamka jest zawsze dziecinnie łatwe, nawet bez analizowania stanu zużycia klawiszy... "

No cóż, moi Drodzy Czytelnicy, będę strzelał do własnej bramki! Powyższy cytat pochodzi nie z niczego innego, jak z mojego własnego artykułu, który napisałem nie tak dawno temu. A zatem, muszę się Wam wytłumaczyć, dlaczego tak nagle zmieniłem zdanie, i proponuję Wam budowę niedawno wyśmianego urządzenia.

Powód zmiany moich poglądów jest bardzo prosty: zmieniły się także czasy, w Elektronice dla Wszystkich dokonano wielkiego przełomu, udostępniając Czytelnikom technikę mikroprocesorową. Pisząc powyżej cytowane słowa, miałem na myśli prymitywne konstrukcje, zbudowane z przerzutników i innych elementów zawartych w kostkach rodzin 74 i 4000. Natomiast wykorzystując technikę mikroprocesorową możemy zbudować zamek szyfrowy o "klasycznej", 10-przyciskowej konstrukcji, który wcale nie będzie zabawką, ale w pełni profesjonalnym systemem zabezpieczającym nasze mienie. Zresztą, niech parametry proponowanego urządzenia powiedzą same za siebie:

1. Nasz zamek szyfrowy umożliwia używanie jako kodu dostępu liczby o praktycznie dowolnej liczbie cyfr. Przez "praktycznie dowolnej" rozumiem, że maksymalna długość kodu wynosi aż 253 cyfr. Stosowanie kodów dłuższych niż dziesięć, no powiedzmy 20 cyfr jest nierealne, ponieważ zapamiętanie tak długiego ciągu znaków wymagałoby już (dla większości osób) zapisywania go na kartce, a tym samym wiązałoby się z ograniczeniem komfortu obsługi urządzenia. Jednak odgadnięcie nawet liczby 10-cyfrowej jest w zasadzie niemożliwe, a w każdym razie równie trudne, jak trafienie głównej wygranej w Toto Lotka. Istnieje także możliwość stosowania i zapamiętywania liczb dłuższych: po prostu jako fragmentów takiego kodu należy stosować sobie tylko znane i dobrze pamiętane dane liczbowe, np. daty urodzin członków rodziny czy numery telefonów znajomych. Nie radzę jedynie wprowadzania do kodu daty własnego ślubu lub własnych urodzin. Tę pierwszą wartość z pewnością zapomną mężczyźni, a drugą - kobiety.
2. Układ wyposażony jest w typowe wyjście przekaźnikowe, do którego dołączyć można praktycznie dowolny odbiornik energii elektrycznej. Jest to jednak tylko jedno z dwóch wyjść układu zamka. Przewidując, że proponowany układ znajdzie zastosowanie głównie jako urządzenie do blokowania dostępu do pomieszczeń, wyposażyłem je w drugie wyjście, radykalnie rozwiązujące problemy z mechanicznym układem zamka do drzwi. Powszechnie stosowane rygle elektromagnetyczne mają jedną, dość istotną wadę: zamknięte za ich pomocą drzwi można otworzyć bez najmniejszych problemów za pomocą ... nieco mocniejszego kopnięcia! Natomiast stosowanie solidnych zasuw napotykało jak dotąd na problemy z wykonaniem poruszającej je mechaniki. Nasz zamek rozwiązuje wszystkie te problemy, sterując bezpośrednio dowolną liczbą serwomechanizmów modelarskich. Serwomechanizm bez najmniejszego trudu może poruszyć nawet najcięższą i najsolidniejszą zasuwę, skutecznie chroniącą zabezpieczane pomieszczenie przed wtargnięciem niepowołanych osób. Zastosowanie kilku serwomechanizmów lub dodatkowej przekładni mechanicznej umożliwi natomiast konstruowanie skomplikowanych, wieloryglowych blokad, niemożliwych do sforsowania bez użycia czegoś w rodzaju M1A1 ABRAMS.
3. Układ może pracować w dwóch, ustawianych za pomocą jumpera, trybach:
   - w pierwszym trybie każde prawidłowe wybranie kodu zmienia stan wyjść układu na przeciwny, to znaczy, że np. pierwsze wybranie kodu otwiera drzwi, a drugie je zamyka,
   - w drugim trybie wprowadzenie poprawnego kodu powoduje otwarcie drzwi na okres ok. 20 sekund, po czym układ automatycznie zamyka strzeżone pomieszczenie.
4. Umieszczenie całej "inteligencji" układu wewnątrz zaprogramowanego procesora pozwoliło na radykalne uproszczenie urządzenia. Konstrukcja zamka jest banalnie prosta i możliwa do wykonania nawet dla zupełnie początkujących elektroników. Program umieszczony w procesorze został napisany i skompilowany z wykorzystaniem pakietu BASCOM, a opis działania układu będzie poparty obszernymi listingami, tak aby Studenci prowadzonego na łamach EdW BASCOM College mogli spróbować własnych sił i napisać własny program sterujący pracą zamka.

Rys. 1 Płytka zamka szyfrowego AVT-144 z EdW 1/96 (12x8cm, 12 układów scalonych)



Jeżeli ktoś ma jeszcze wątpliwości w ocenie możliwości techniki mikroprocesorowej, to prosimy porównać tę konstrukcję z układem zamka szyfrowego, jaki opublikowałem w EdW 1/96. Ówczesny układ miał znacznie uboższe możliwości funkcjonalne, ale zawierał 12 układów scalonych na płytce o wymiarach 14 x 8 cm (**rys. 1**).

Jak to działa?

Schemat elektryczny zamka szyfrowego został pokazany na **rysunku 2**. Jak już wspomniano, sercem układu jest dobrze nam znany procesor typu AT89C2051, wspomagany przez jeden tylko dodatkowy układ: pamięć EEPROM typu AT24C04. Zastosowanie tego układu było absolutnie konieczne, ponieważ procesor AT89C2051 nie posiada wewnętrznej nieulotnej pamięci danych. Pamięć zewnętrzna pozwoli na przechowywanie kodu zamka, bez ryzyka "wywalenia" się całej konstrukcji w przypadku wystąpienia przerwy w zasilaniu. Pozostała część urządzenia to już tylko garstka elementów dyskretnych niezbędnych do prawidłowego funkcjonowania układu, zasilacz zbudowany z wykorzystaniem scalonego stabilizatora napięcia typu 7805, i oczywiście, klawiatura z 12 przyciskami. W wykonaniu praktycznym układ został podzielony na dwie części umieszczone na osobnych płytkach obwodów drukowanych i połączonych za pomocą złącza CON2 + CON3.

Sądzę, że analizę sposobu działania układu, a właściwie zaszytego w pamięci EEPROM procesora programu, najlepiej rozpocząć w momencie pierwszego włączenia zasilania, kiedy to w pamięci EEPROM IC2 nie ma jeszcze zapisanych jakichkolwiek informacji.

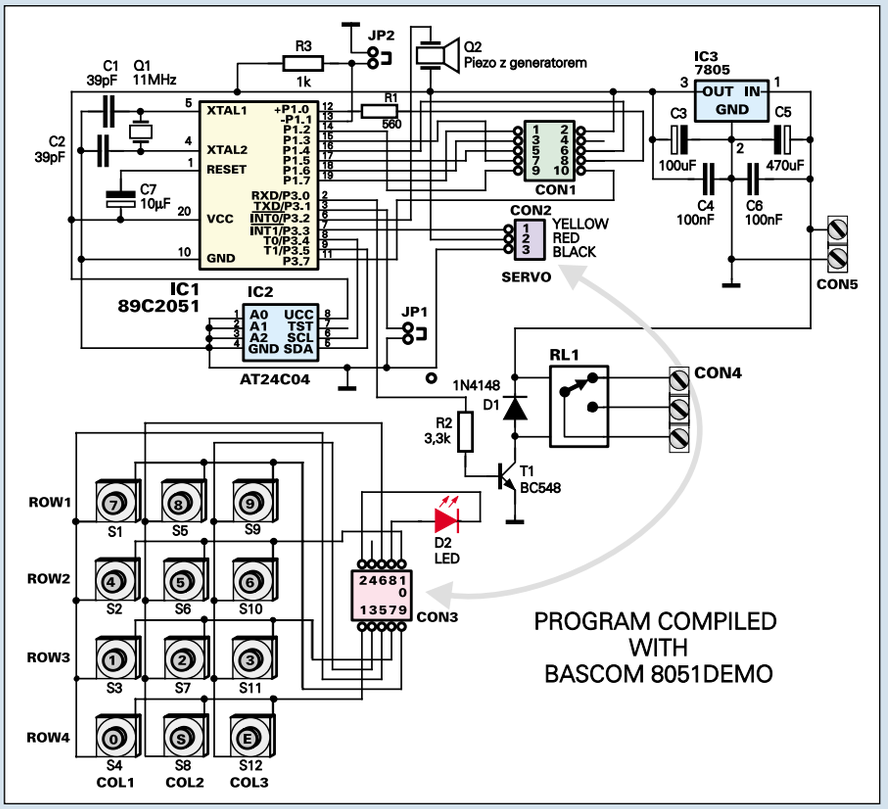
Pierwszą czynnością, jaką program ma do wykonania, będzie właśnie sprawdzenie zawartości pamięci EEPROM i próba ustalenia, czy został w niej zapisany jakikolwiek kod. Wykonanie następujących instrukcji:

```
Call Read_eeprom 255 , Value      'odczytaj
        zawartość adresu 255 pamięci EEPROM
          If Value <>44 Then      'jeżeli
        odczytana wartość nie jest równa 44 to:
            Call Rejestracja      'wezwij
                    podprogram REJESTRACJA
End If                            'koniec
                                  uwarunkowania
```

gdzie podprogram:

```
Sub Read_eeprom(adres As Byte , Value As
Byte)
  I2cstart                  'start transmisji I2C
  I2cwbyte 160              'podanie adresu
                              pamięci EEPROM
  I2cwbyte Adres            'podanie adresu
      spod jakiego mają  zostać odczytane dane
  I2cstart                  'ponowny start
                              transmisji I2C
  I2cwbyte 161              'ustawienie pamięci
                            w tryb odczytu danych
  I2crbyte Value , Ack      'odczytaj wartość
                            bajtu z potwierdzeniem
  I2cstop                   'koniec transmisji
                                    danych
End Sub
```

odczytuje wartość "VALUE" spod wskazanego adresu pamięci, pozwala na ustalenie



Rys. 2

z dużym prawdopodobieństwem, czy w pamięci zapisane są już potrzebne do pracy zamka dane. Dlaczego napisałem " z dużym prawdopodobieństwem", a nie z pewnością? Dlatego, że takie sprawdzenie zawartości EEPROM-u daje całkowitą pewność jedynie w przypadku kostki "fabrycznie nowej", której cała zawartość zapisana jest wyłącznie FF-ami (FF(HEX) = 255(DEC)). Tylko w tym przypadku nieodczytanie spod adresu 255 metafizycznej (ukłon w stronę naszego Wieszcza Narodowego) wartości 40 i 4 (zapisywanej tam podczas rejestracji szyfru) jest dowodem, że kostka nie była jeszcze nigdy użyta do zapisywania kodów zamka szyfrowego. Jeżeli jednak zastosowaliśmy w układzie używany element, w którym były już zapisywane dane, to mamy jedną szansę na 256, że układ źle zinterpretuje odczytaną wartość i będziemy musieli uruchomić go powtórnie, tym razem ze zwartym jumperem JP2. Fakt ten zostanie wykryty przez podprogram:

```
Set P1.1                          'ustaw na
                            wejściu P1.1 stan wysoki
  If P1.1 = 0 Then          'jeżeli próba
                                  nieudana, to:
    Call Rejestracja        'wezwij podprogram
REJESTRACJA
  End If                          'koniec
                                  uwarunkowania
```

Zarówno w pierwszym, jak i drugim opisywanym przypadku następuje wywołanie podprogramu REJESTRACJA, który umożliwi nam wprowadzenie do pamięci kodu, za pomocą którego będziemy mogli otwierać nasz zamek. Podprogram ten wygląda następująco:

```
Sub Rejestracja
For Licznik = 1 To 10   'dziesięciokrotnie:
Call Ledshort           'wezwij podprogram
krótkiego włączenia diody LED i sygnału akustycznego
Next Licznik            'jeszcze raz
Licznik = 0             'wyzeruj zmienną LICZNIK
Do                      'wejście w pętle DO ....
                                    LOOP
Digit = 255             'zmienna DIGIT przyjmuje
                                  wartość FF
Call Keyscan            'wezwij podprogram
                            skanowania klawiatury
If Digit < 10 Then      'jeżeli wartość odczytana
          z klawiatury jest mniejsza niż 10, to:
Call Write_eeprom Licznik , Digit  'zapisz ją w
                    pamięci pod adresem LICZNIK
Incr Licznik            'zwiększ zmienną LICZNIK
End If                  'koniec uwarunkowania
If Digit = 11 Or Licznik = 254 Then  'jeżeli
      wartość odczytana z klawiatury równa jest
 11 (klawisz 'ENTER) lub jeżeli wprowadzono już
        253 cyfry (mało 'prawdopodobne!), to:
For R = 1 To 10         'dziesięciokrotnie:
Call Ledshort           'wezwij podprogram
krótkiego włączenia diody LED i sygnału akustycznego
Next R                  ' jeszcze raz
Call Write_eeprom 255 , 44
'zapisz w EEPROM wartość 44 pod adresem 255
Call Write_eeprom , 254 , Licznik  'zapisz w
                    EEPROM ilość cyfr w kodzie
Licznik = 0             'wyzeruj zmienną LICZNIK
Exit Do                 'wyjdź z pętli
Return                  'powróć do programu
                                  głównego
End If                  'koniec uwarunkowania
Loop                    'kontynuuj pracę w pętli
programowej
End Sub                 'koniec podprogramu
```

Wykonanie powyższego podprogramu spowoduje, że w pamięci EEPROM zostanie zapisany szyfr, który może zostać wykorzystany do otwierania zamka, i informacje

o liczb i cyfr w kodzie oraz o fakcie jego zaprogramowania. Wprowadzanie kodu, podczas którego wykorzystujemy wszystkie cyfry dostępne z klawiatury, kończymy naciśnięciem klawisza S12 - ENTER.

Warto teraz wyjaśnić, w jaki sposób procesor odczytuje dane z dwunastoprzyciskowej klawiatury. Popatrzmy jeszcze raz na schemat: wszystkie klawisze zostały połączone w matrycę składającą się z czterech rzędów (ROW1 ... ROW4) i trzech kolumn (COL1 ... COL3). Każda kolumna i każdy rząd dołączone zostały do wyprowadzeń portu P1 i P3 procesora. Aby ułatwić sobie pisanie programu i uwolnić się od konieczności ciągłego spoglądania na schemat, nadajmy odpowiednim wyprowadzeniom procesora nowe nazwy, wykorzystując wygodne polecenie ALIAS. A więc:

Skanowanie klawiatury odbywa się następująco:

```
Row1 Alias P1.2          'pin P1.2 możemy
                  odtąd nazywać "ROW1"
Row2 Alias P3.7          'pin P3.7 możemy
                  odtąd nazywać "ROW2"
Row3 Alias P1.3          'pin P1.3 możemy
                  odtąd nazywać "ROW3"
Row4 Alias P1.7          'pin P1.7 możemy
                  odtąd nazywać "ROW4"
Col1 Alias P1.6          'pin P1.6
            możemy odtąd nazywać "COL1"
Col2 Alias P1.4          'pin P1.4
            możemy odtąd nazywać "COL2"
Col3 Alias P1.5          'pin P1.5
            możemy odtąd nazywać "COL3"
```

1. Procesor kolejno ustawia na wyjściach ROW1 ... ROW4 stan niski.

2. Następnie procesor stara się wymusić na kolejnych wejściach COL1 ... COL3 stan wysoki.

3. Jeżeli powyższa próba się nie udaje, to oznacza to, że naciśnięty został klawisz odpowiadający aktualnie wybranemu rzędowi i kolumnie matrycy.

Przykładowo przeanalizujmy fragmenty podprogramu KEYSCAN:

```
Sub Keyscan
Reset Row1          'ustaw stan niski na
          pierwszym rzędzie matrycy klawiatury
Set Col1            'ustaw stan wysoki
       w pierwszej kolumnie matrycy klawiatury
If Col1 = 0 Then    'jeżeli stan COL1 w dalszym
 ciągu pozostaje niski (naciśnięty klawisz S1), to:
Digit = 7           'zmienna DIGIT (wartość
    odczytana z klawiatury) przyjmuje wartość 7
Call Ledshort       'błyśnij raz diodą LED i podaj
                                sygnał akustyczny
While Col1 = 0      'poczekaj na puszczenie
                                klawisza
Wend
End If              'koniec uwarunkowania

Set Col2                'ustaw stan wysoki w
          drugiej kolumnie matrycy klawiatury
If Col2 = 0 Then    'jeżeli stan COL2 w dalszym
 ciągu pozostaje niski (naciśnięty klawisz S5), to:
Digit = 8           'zmienna DIGIT (wartość
    odczytana z klawiatury) przyjmuje wartość 8
Call Ledshort       'błyśnij raz diodą LED i podaj
                                sygnał akustyczny
While Col2 = 0      'poczekaj na puszczenie
                                klawisza
Wend
End If              'koniec uwarunkowania
.....
Set Row1 : Reset Row2  'ustaw stan niski tylko
          w drugim rzędzie matrycy klawiatury
Set Col1            'ustaw stan wysoki w
       pierwszej kolumnie matrycy klawiatury
If Col1 = 0 Then    'jeżeli stan COL1 w dalszym
 ciągu pozostaje niski (naciśnięty klawisz S2), to:
Digit = 4           'zmienna DIGIT (wartość
    odczytana z klawiatury) przyjmuje wartość 4
Call Ledshort       'błyśnij raz diodą LED i podaj
                                sygnał akustyczny
While Col1 = 0      'poczekaj na puszczenie
                                klawisza
Wend
End If
.....
End Sub             'koniec podprogramu
                    skanowania klawiatury
```

Po zarejestrowaniu kodu (należy teraz usunąć jumper JP2) układ przechodzi w stan czuwania, podczas którego nieustannie skanuje klawiaturę, oczekując na wprowadzenie prawidłowego szyfru. Program sprawdza też stan jumpera JP1, który decyduje, a w jakim trybie układ ma pracować. Jeżeli teraz naciśnięty zostanie którykolwiek z klawiszy, to program przystępuje do analizy wprowadzanych cyfr i porównywania ich z zapisanym w pamięci kodem dostępu. Badaniem poprawności wprowadzonego szyfru zajmuje się program główny:

```
Sub Mainloop
Do
Set P3.1            'ustaw stan wysoki na
                    wejściu P3.1 (jumper JP1)
If P3.1 = 1 Then    'jeżeli udało się, to:
Fl = 1              'zmienna Fl przyjmuje
                                wartość 1
Else                'w przeciwnym przypadku:
Fl = 0              'zmienna FL przyjmuje
                                wartość 0
End If              'koniec uwarunkowania
Flag1 = 1           'zmienna FLAG1
                            przyjmuje wartość 1
Call Read_eeprom 254 , Value   'odczytaj
          zawartość pamięci pod adresem 254
Digits = Value      'zmienna DIGITS (liczba
    cyfr w kodzie) przyjmuje odczytaną wartość
Call Keyscan        'sprawdź stan klawiatury
If Digit < 10 Then  'jeżeli naciśnięty został
                                klawisz 0 ... 9, to:
Call Read_eeprom Licznik , Value  'odczytaj z
                    pamięci kolejną cyfrę kodu
Incr Licznik        'zwiększ stan zmiennej
                                LICZNIK
End If              'koniec uwarunkowania
If Digit < 10 Then  'jeżeli naciśnięty został
                                klawisz 0 ... 9 to:
If Digit <> Value Then  'jeżeli odczytana z
   klawiatury cyfra nie odpowiada kolejnej cyfrze
'kodu, to:
Flag1 = 0           'zmienna FLAG1 przyjmuje
                                wartość 0
Licznik = 0         'wyzerowanie licznika
End If              'koniec uwarunkowania
End If              'koniec uwarunkowania
If Digit = 11 Then  'jeżeli odczytana z
   klawiatury wartość to 11 (klawisz ENTER), to:
If Digits = Licznik Then        'jeżeli
          wprowadzono właściwą liczbę cyfr, to:
If Flag1 = 1 Then   ',jeżeli zmienna FLAG1
równa jest 1 (wszystkie cyfry były poprawne), to:
If Fl = 1 Then      'jeżeli zmienna FL równa
                                jest 1, to:
Licznik = 0         'wyzeruj licznik
Call Otwieranie     'wezwij podprogram
                                OTWIERANIE
End If              'koniec uwarunkowań
End If
End If
End If
If Digit = 11 Then  'jeżeli odczytana z
   klawiatury wartość to 11 (klawisz ENTER), to:
If Digits = Licznik Then        'jeżeli
          wprowadzono właściwą ilość cyfr, to:
If Flag1 = 1 Then   ',jeżeli zmienna FLAG1
 równa jest 1 (wszystkie cyfry były poprawne), to:
If Fl = 0 Then      'jeżeli zmienna FL równa
                                jest 0, to:
Licznik = 0         'wyzeruj licznik
Call Otwieranie2    'wezwij podprogram
                                OTWIERANIE2
End If              'koniec uwarunkowań
End If
End If
End If
If Digit = 10 Then  'jeżeli odczytana z
   klawiatury wartość to 10 (klawisz SET) to:
If Digits = Licznik Then        'jeżeli
          wprowadzono właściwą ilość cyfr, to:
If Flag1 = 1 Then   ',jeżeli zmienna FLAG1
 równa jest 1 (wszystkie cyfry były poprawne), 'to:
Licznik = 0         'wyzeruj licznik
Call Rejestracja    'ponownie wezwij
                    podprogram rejestracji kodu
End If              'koniec uwarunkowań
End If
End If
Digit = 255         'zmienna DIGIT przyjmuje
wartość 255
Loop                'zamknięcie pętli pro-
gramowej
End Sub
```

Warto jeszcze wspomnieć o sposobie, w jaki program pozycjonuje położenie wału napędowego serwomechanizmu. Służą temu niedawno opracowane przez MCS Electronics polecenia:
CONFIG SERVOS = [liczba zastosowanych serwomechanizmów (ograniczona jedynie liczbą dostępnych wyprowadzeń procesora)]
SERVO[x] = pin (poinformowanie kompilatora, do których wyprowadzeń zostały dołączone serwa)
SERVO [numer serwa] = [kąt obrotu]

Gdzie kąt obrotu może być określony liczbą z przedziału 5 ... 24 (przy stosowaniu standardowych serw modelarskich daje to obrót wału o blisko 180°). Wał serwa możemy pozycjonować w 19 położeniach, z krokiem co około 9,5°. Ograniczało to nieco stosowanie serwomechanizmów do wykonywania precyzyjnych czynności i spowodowało konieczność modyfikacji polecenia SERVO. W kolejnej edycji BASCOM-a będzie ono umożliwiać poruszanie serwomechanizmu z krokiem co 1°. Na szczęście, w naszym programie zamka szyfrowego ograniczenie pierwszej wersji polecenia SERVO nie miało najmniejszego znaczenia,

ponieważ wał serwa ustawiany jest jedynie w dwóch skrajnych pozycjach.

Montaż i uruchomienie

Na **rysunku 3** zostały pokazane płytki drukowane, które możemy wykorzystać do budowy naszego zamka. Płytek tych jest wyjątkowo dużo, aż pięć, ale tylko dwie z nich są absolutnie niezbędne do wykonania proponowanej konstrukcji. Trzy pozostałe to płyta czołowa zamka i dwie płytki stanowiące uchwyt do zamocowania serwomechanizmu.

Montaż rozpoczniemy od wlutowania w płytkę główną wszystkich elementów elektronicznych, rozpoczynając od zamontowania podzespołów o najmniejszych gabarytach, a kończąc na kondensatorach elektrolitycznych i przekaźniku, o ile mamy zamiar go wykorzystywać. Pod układy scalone należy zastosować podstawki. Opcjonalne jest także użycie stabilizatora napięcia IC4. Jeżeli będziemy dysponować źródłem napięcia o wartości 4,5 ... 6VDC, to elementu tego nie musimy montować, a także możemy pominąć kondensatory C5 i C6.

Po zmontowaniu płytki głównej weźmy się za klawiaturę. Jednak przed rozpoczęciem tego etapu pracy warto zastanowić się nad sposobem wykonania klawisza S8, służącego do wywoływania programu ponownego ustawiania kodu. Sądzę, że klawisz ten powinien być o kilka milimetrów krótszy niż pozostałe. Uniemożliwi to jego przypadkowe naciśnięcie po wybraniu kodu, pozostawiając możliwość naciśnięcia go za pomocą np. zapałki lub śrubokręta.

Równe wlutowanie w płytkę dwunastu przycisków nie jest bynajmniej sprawą najłatwiejszą i dlatego też proponuję najpierw włożyć wyprowadzenia wszystkich przycisków oraz diody LED D2 w przeznaczone na nie otwory w punktach lutowniczych, a następnie nałożyć na całość trzecią płytkę - płytę czołową układu. Po dociśnięciu płyty czołowej do płytki klawiatury możemy mieć całkowitą pewność, że wszystkie przyciski oraz dioda LED zostaną przylutowane idealnie równo.

Zmontowane płytki musimy połączyć ze sobą za pomocą 10 odcinków srebrzanki przylutowanych do punktów lutowniczych oznaczonych jako CON1 i CON3. Jest to jedyna nieco trudniejsza czynność, jaką będziemy musieli wykonać podczas budowy zamka, ponieważ lutować musimy od strony ścieżek, a przewody powinny być jak najkrótsze.

Trzy płytki: płytkę bazową, klawiaturę i płytę czołową łączymy ze sobą za pomocą czterech śrubek M3 i tulejek dystansowych. W przypadku braku odpowiednich tulejek możemy wykorzystać dodatkowe nakrętki, mocujące poszczególne płytki (szczegóły widoczne są na fotografiach).

Płyta czołowa zamka jest nieco większa od płytek obwodów drukowanych, a umieszczone w jej rogach otwory ułatwiają zamocowanie całej konstrukcji np. w otworze wyciętym w drzwiach wejściowych do strzeżonego pomieszczenia.

Po zmontowaniu całego układu zamka wkładamy układy scalone w podstawki i dołączamy zasilanie. Jeżeli mamy zamiar wykorzystywać przekaźnik RL1, to napięcie zasilające musi wynosić około 12VDC. Jeżeli wykorzystywać będziemy jedynie serwomechanizm, to po usunięciu stabilizatora napięcia IC4 możemy zastosować zasilanie napięciem z przedziału 4,8 ... 6VDC. Tu bardzo ważna uwaga: sam układ elektroniczny pobiera bardzo mało prądu, czego niestety nie można powiedzieć o serwomechanizmie pracującym pod obciążeniem. Pobór prądu przez serwo może wynieść nawet 1,5A (przez 1 ... 2 sekundy podczas przesuwania rygla), co należy uwzględnić podczas projektowania układu zasilającego.

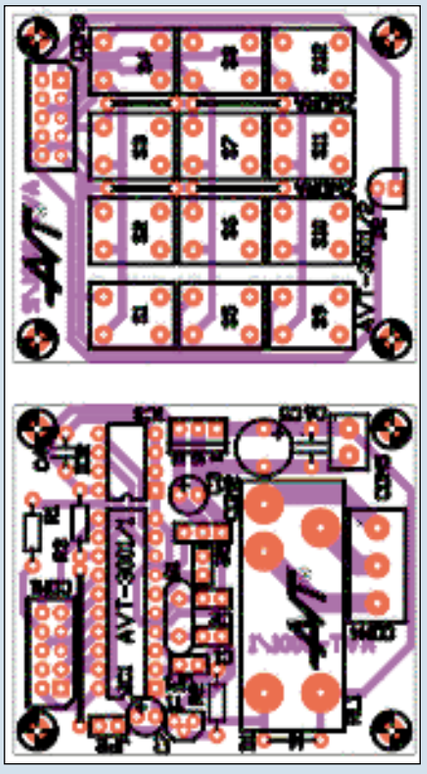
Wykaz elementów

C1, C2	39pF
C3	100µF/16
C4, C6	100nF
C5	470µF/25V
C7	10µF/16V
R1	560Ω
R2	3,3kΩ
R3	1kΩ
D1	1N4148
D2	LED f 3mm
IC1	zaprogramowany procesor 89C2051
IC2	AT24C04
IC3	7805
T1	BC548
CON1	2x5 goldpin ów
CON2	3x goldpin
CON4	ARK3
CON5	ARK2 (3,5mm)
JP1, JP2	2 x goldpin + jumper
Q1	rezonator kwarcowy 11,059MHz
Q2	Piezo z generatorem
RL1	przekaźnik typu RM96/12V
S1 ... S12	microswitch 10 mm (S8 - 3mm)

Serwomechanizm i płytki do jego mocowania nie wchodzą w skład kitu.

Komplet podzespołów i trzy płytki są dostępne w sieci handlowej AVT jako kit szkolny AVT-3001

Rys. 3



Programowanie zamka szyfrowego

Po pierwszym włączeniu zasilania układ automatycznie przechodzi w tryb programowania, co zostaje zasygnalizowane dziesięcioma sygnałami akustycznymi i błyśnięciami diody LED. Gdyby, co jest bardzo mało prawdopodobne, tak się nie stało, to należy zewrzeć jumper JP2 i ponownie włączyć zasilanie. Kod wprowadzamy korzystając z numerycznej sekcji klawiatury, wprowadzając kolejne cyfry kodu. Każde naciśnięcie klawisza potwierdzane jest sygnałem akustycznym i błyskiem diody LED. Po zakończeniu wpisywania szyfru naciskamy klawisz ENTER, co spowoduje zapisanie danych w pamięci i przejście układu w stan oczekiwania. Zakończenie wprowadzania kodu kwitowane jest, podobnie jak jego rozpoczęcie, dziesięcioma sygnałami akustycznymi i optycznymi.

Po wprowadzeniu kodu musimy jeszcze zdecydować, w jakim trybie pracy ma pracować nasz zamek.

1. Tryb pracy 1. Zwarcie jumpera JP1 spowoduje, że każde kolejne wybranie kodu będzie powodowało naprzemiennie otwieranie i zamykanie zamka.

2. Tryb pracy 2. Jeżeli jumper JP1 pozostawimy rozwarty, to po wybraniu szyfru zamek będzie pozostawał otwarty przez około 20 sekund, po czym automatycznie zostanie zamknięty.

Otwieranie zamka szyfrowego

Wprowadzamy ustawiony uprzednio kod wykorzystując dziesięć klawiszy klawiatury numerycznej. Po wprowadzeniu wszystkich zaprogramowanych cyfr, naciskamy klawisz ENTER. Podanie prawidłowego kodu skwitowane zostanie przez układ podwójnym sygnałem akustycznym i optycznym (błyśnięciami diody LED). Jeżeli zamek pracuje w trybie 2, to dioda LED pozostanie włączona przez cały czas otwarcia zamka.

Zmiana ustawionego szyfru

Istnieją dwie metody zmiany ustawionego kodu. Jedna z nich została już opisana wyżej: zwieramy jumper JP2 i po ponownym włączeniu zasilania wprowadzamy z klawiatury nowy szyfr. Jednak taka metoda nie zawsze byłaby wygodna, chociażby ze względu na brak łatwego dostępu do płytki zamka. Dlatego też istnieje druga, łatwiejsza metoda, w której wszystkie operacje wykonywane są z klawiatury.

1. Wybieramy ustawiony uprzednio szyfr.

2. Zamiast klawisza ENTER naciskamy klawisz SET. Jeszcze raz zwracam uwagę, że klawisz ten powinien być krótszy od pozostałych, tak aby nie było możliwe jego przypadkowe naciśnięcie.

3. Wprowadzamy nowy kod i potwierdzamy klawiszem ENTER.

Zbigniew Raabe
e-mail: **zbigniew.raabe@edw.com.pl**