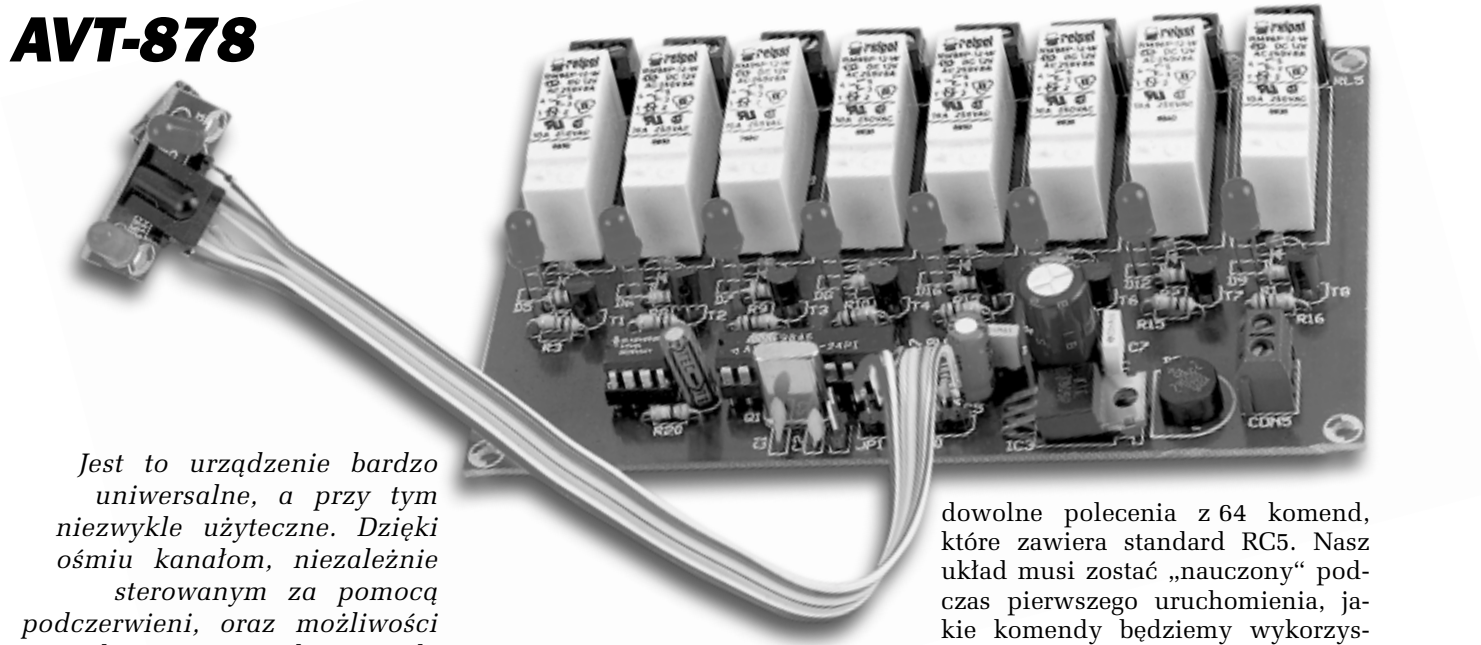
Zdalnie sterowany moduł I/O z programowanym kodem dostępu

AVT-878

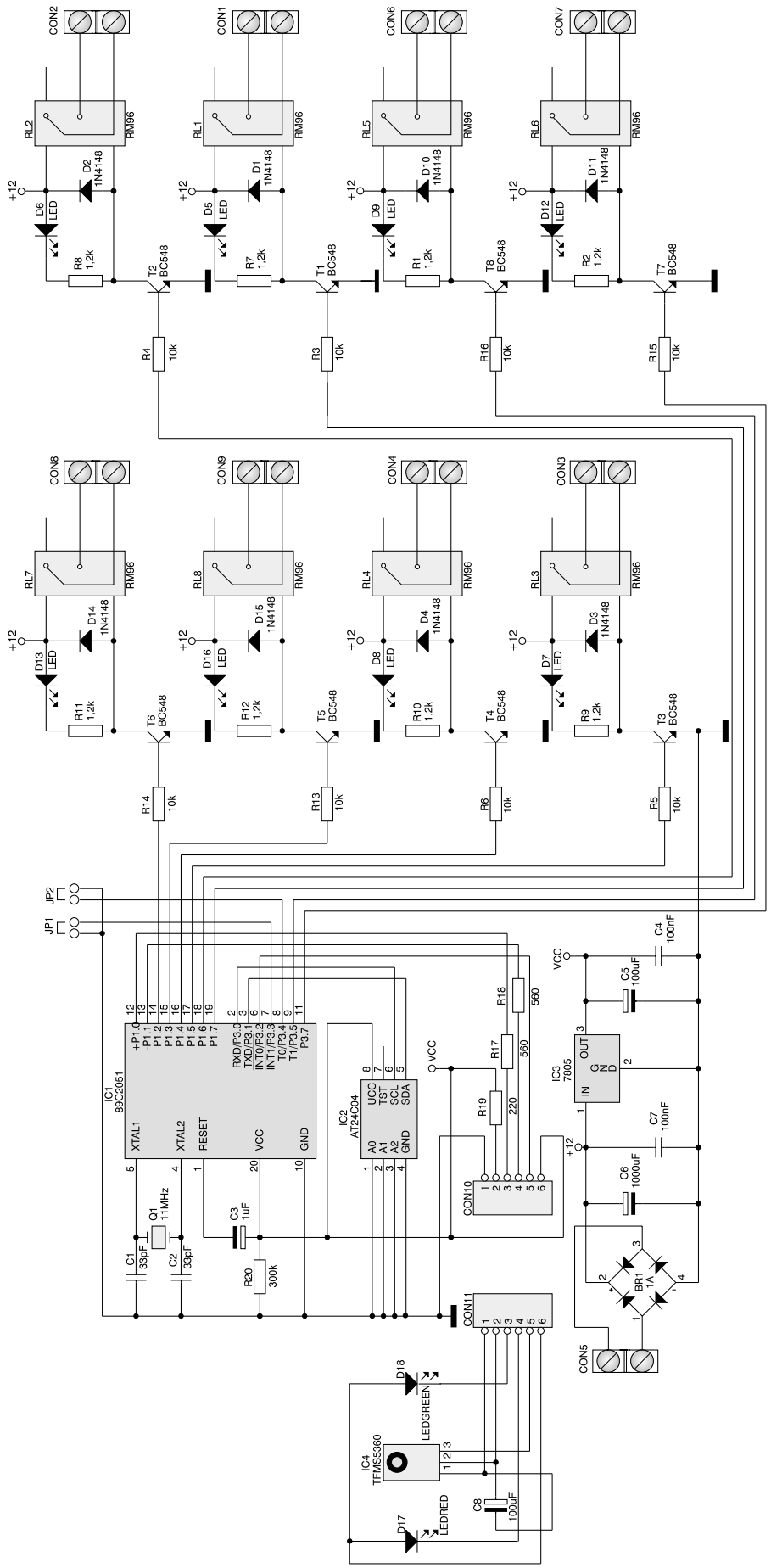


Jest to urządzenie bardzo uniwersalne, a przy tym niezwykle użyteczne. Dzięki ośmiu kanałom, niezależnie sterowanym za pomocą podczerwieni, oraz możliwości zabezpieczenia dostępu do nich hasłem, sterownik można dostosować nawet do najbardziej specjalistycznych zastosowań.

W jednym z poprzednich numerów Elektroniki Praktycznej pozwoliłem sobie zaprezentować Czytelnikom opis budowy uniwersalnego pilota, który może współpracować z dowolnym odbiornikiem kodu RC5. Wspomniałem przy tym, że opisywany układ może okazać się użyteczny nie tylko do sterowania pracą fabrycznego sprzętu RTV, ale także do współpracy z najróżniejszymi układami domowej automatyki, które w najbliższym czasie skonstruujemy. Najwyższa więc pora na dotrzymanie obietnicy i opisanie Czytelnikom jednego z układów sterowanych kodem RC5, który może znaleźć zastosowanie jako uniwersalny sterownik praktycznie dowolnych urządzeń zasilanych energią elektryczną.

Proponowany układ nie jest związany z jakimkolwiek konkretnym pilotem i może współpracować z dowolnym nadajnikiem kodu RC5, w tym oczywiście z pilotem AVT-849. Po drugie, nie ma najmniejszego znaczenia, jakie komendy wykorzystamy do sterowania naszym układem. Mogą to być dowolne polecenia z 64 komend, które zawiera standard RC5. Nasz układ musi zostać „nauczony" podczas pierwszego uruchomienia, jakie komendy będziemy wykorzystywać i pod jaki adres będą one wysyłane. Zapamiętane informacje przechowywane są w pamięci EEPROM i mogą być stamtąd usunięte lub zmienione jedynie w wyniku naszego celowego działania. To jednak nie wszystkie cechy odróżniające proponowany układ od innych, o podobnym działaniu.

Nie zawsze jesteśmy zadowoleni z faktu, że do urządzeń sterowanych za pomocą kodu RC-5 może mieć dostęp praktycznie każdy, kto tylko posiada odpowiedni nadajnik, którym najczęściej jest standardowy pilot od sprzętu RTV. Nie tylko małe dzieci potrafią narobić w domu niezłego bałaganu, ale i w wielu firmach jest pożądane, aby do pewnych urządzeń miały dostęp tylko uprawnione osoby. Dostęp do sterowanych urządzeń możemy zablokować za pomocą szyfru o praktycznie dowolnej liczbie cyfr (do 210), a tym samym całkowicie uniemożliwić korzystanie z niego przez dzieci czy też pracowników „niższego szczebla". Oczywiście, kod dostępu może być w każdej chwili zmieniony,



Rys. 1. Schemat elektryczny pilota.

a blokowanie nim dostępu do układu jest opcją, którą możemy wykorzystywać lub nie.

Opis działania układu

Na **rys. 1** znajduje się schemat elektryczny proponowanego układu sterownika. Układ jest wyjątkowo prosty: poza procesorem 89C2051 i pamięcią AT24C04 układ składa się z ośmiu przekaźników wykonawczych, odbiornika kodu RC5 - TFMS5360 i garstki elementów dyskretnych. Procesor 89C2051 posiada liczne zalety, ale także i jedną wadę: nie jest wyposażony w wewnętrzną nieulotną pamięć danych, co spowodowało konieczność zastosowania dodatkowego układu przechowującego informacje o kodach poszczególnych komend sterujących pracą urządzenia i ewentualnie szyfru blokującego do niego dostęp.

Sposób połączenia procesora z biernymi elementami układu najlepiej prześledzić na podstawie fragmentu programu, w którym za pomocą arcywygodnego polecenia ALIAS nadano wyprowadzeniom procesora nazwy zgodne z pełnionymi przez nie funkcjami. Dodatkowo w systemie utworzono magistralę I²C, która służy obsłudze zewnętrznej pamięci danych - IC2.

```
Relay1 Alias P1.7
Relay2 Alias P1.6
Relay3 Alias P1.5
Relay4 Alias P1.4
Relay5 Alias P3.5
Relay6 Alias P3.7
Relay7 Alias P1.2
Relay8 Alias P1.3
Ledgreen Alias P1.0
Ledred Alias P1.1
Jumper1 Alias P3.3
Jumper2 Alias P3.4
Config Sda = P3.1
Config Scl = P3.0
```

Omówienie działania układu przeprowadzimy w oparciu o fragmenty programu uruchamianego w momencie włączenia zasilania urządzenia. Pierwszą czynnością, jaką procesor wykonuje po inicjalizacji i konfiguracji systemu, jest sprawdzenie stanu dwóch jumperów: JP1 i JP2. Od wyniku tego badania zależeć będzie dalszy bieg wypadków.

```
Set Jumper1 : Set Jumper2
If Jumper1 = 0 Then
   If Jumper2 = 1 Then
     Call Warning
```

```
      Call Commands_registration
    End If
End If
If Jumper1 = 0 Then
    If Jumper2 = 0 Then
      Call Warning
      Call Code_registration
    End If
End If
If Jumper1 = 1 Then
    If Jumper2 = 0 Then
      Call Waitingforcode
    End If
End If
If Jumper1 = 1 Then
    If Jumper2 = 1 Then
      Call Waitingforcommand
    End If
End If
```

Z analizy powyższego listingu wynika, że stan jumpera JP1 decyduje o tym, czy program ma przystąpić do rejestrowania wydawanych z pilota poleceń (JP1 zwarty), czy też od razu przejść do wykonywania wyznaczonych mu za pomocą jumpera JP2 zadań. Cztery możliwe sytuacje pokazane są w **tab. 1**.

Przypadek 1

Poleceniem CALL WARNING zostaje wywołany podprogram generujący sygnał ostrzegawczy - 10 błysków czerwonej diody LED, informujący o przystąpieniu do programowania komend sterujących. Następnie rozpoczyna się realizacja podprogramu rejestrującego polecenia wysyłane z pilota i zapisującego je w pamięci EEPROM. Pojedynczy błysk zielonej diody jest sygnałem zachęty do podania z pilota komendy sterującej pierwszym urządzeniem - przekaźnikiem RL1.

```
Sub Commands_registration
New = 0
Licznik = 1
Call Ledgreenshort
Do
If New = 1 Then
    Disable Int0
    New = 0
    Call Write_eeprom(licznik,
        Command)
    Call Write_eeprom(10,
        Subaddress)
    Incr Licznik
    If Licznik = 9 Then
      Disable Int0
      Call Write_eeprom(255,133)
      Reset Ledgreen
      Wait 3
      Set Ledgreen
      Exit Do
      Call Waitingforcommand
    End If
    For R|= 1 To Licznik
      Call Ledgreenshort
    Next R
    Enable Int0
End If
Loop
End Sub
```

Powyższy podprogram pracuje cały czas w pętli, oczekując na wysłanie z pilota kolejnej komendy. Znacznikiem świadczącym o jej odebraniu jest zmienna NEW, która może przyjąć stan 1 podczas realizacji podprogramu wywoływanego wystąpieniem przerwania na wejściu INT0. Właśnie do tego wejścia dołączone jest wyjście odbiornika kodu RC5 - IC4. Warto zwrócić uwagę na umieszczoną na samym początku programu głównego dyrektywę obsługi przerwania:

```
On Int0 Receiverc5
Enable Interrupts
Enable Int0
```

powodującą, że w przypadku wystąpienia przerwania, czyli odebrania transmisji w podczerwieni, wykonywany jest następujący podprogram:

```
Receiverc5:
    Getrc5(subaddress, Command)
    New = 1|
Return
```

Polecenie GETRC5 jest jednym z licznych „fajerwerków" BASCOM-a, a w wyniku jego działania otrzymujemy liczbowe wartości odebranej komendy RC5 i adresu, pod jaki została wysłana.

Obie te wartości zapisywane są w zewnętrznej pamięci EEPROM, za pomocą wywoływanego poleceniem CALL WRITE_EEPROM podprogramu:

```
Sub Write_eeprom (adres As Byte,
Value As Byte)
    I2cstart
    I2cwbyte 160
    I2cwbyte Adres
    I2cwbyte Value
    I2cstop
    Waitms 10
End Sub
```

z tym, że dla uproszczenia programu wartość, oznaczająca adres wysyłany przez pilota, zapisywana jest zawsze w tym samym miejscu w pamięci.

Każdy kolejny prawidłowy zapis jest potwierdzany błyskami zielonej diody LED. Liczba tych błysków sygnalizuje, którą z kolei komendę mamy teraz zarejestrować. Zarejestrowanie wszystkich komend, czyli osiągniecie przez zmienną LICZNIK wartości 8, potwierdzane jest włączeniem na okres 3 sekund (WAIT 3) zielonej diody, po czym program przechodzi w stan oczekiwania na odebranie poleceń sterujących przekaźnikami.

Tab. 1. Możliwe konfiguracje jumperów i związane z nimi funkcje.

Case	JP1	JP2	Działanie programu
1	0	1	Przejście do rejestrowania tylko poleceń sterujących
2	0	0	Przejście do rejestrowania kodu dostępu
3	1	1	Przejście w tryb pracy bez zabezpieczenia układu szyfrem
4	1	0	Przejście w tryb pracy z blokadą dostępu za pomocą szyfru

Uwaga: opisane wyżej procedury rejestracji poleceń zostaną przeprowadzone nie tylko w przypadku wykrycia podanej kombinacji stanów wejść Jumper1 i Jumper2, ale także po pierwszym uruchomieniu układu, w którym znajduje się niezaprogramowana jeszcze pamięć EEPROM.

Zwróćmy uwagę na polecenie Call Write_eeprom(255, 133), wykonywane przed wyjściem z podprogramu rejestracji komend. Zapisuje ono w wolnej komórce pamięci, pod adresem 255, umowną wartość 133, której obecność pod tym adresem świadczy teraz o uprzednim zaprogramowaniu komend. Na początku działania program sprawdza zawartość tej komórki pamięci i jeżeli odnajduje w niej wartość inną, niż podana, to automatycznie przechodzi do opisanego wyżej podprogramu rejestracji komend.

Przypadek 2

Wykrycie przez program stanów niskich na wejściach Jumper1 i Jumper2 procesora (pamiętajmy o poleceniach ALIAS) powoduje rozpoczęcie przez procesor rejestrowania kodu dostępu do sterowanych przez układ urządzeń. Należy zauważyć, że kod dostępu możemy wprowadzić za-

wsze, nawet jeżeli w najbliższej przyszłości nie mamy zamiaru z niego korzystać. W każdym bowiem przypadku o sposobie działania programu decydować będzie późniejsze ustawienie jumperów i jeżeli zostaną one ustawione w pozycji Jumper1 = 1 i Jumper2 = 1, to program nie będzie żądał podania kodu dostępu.

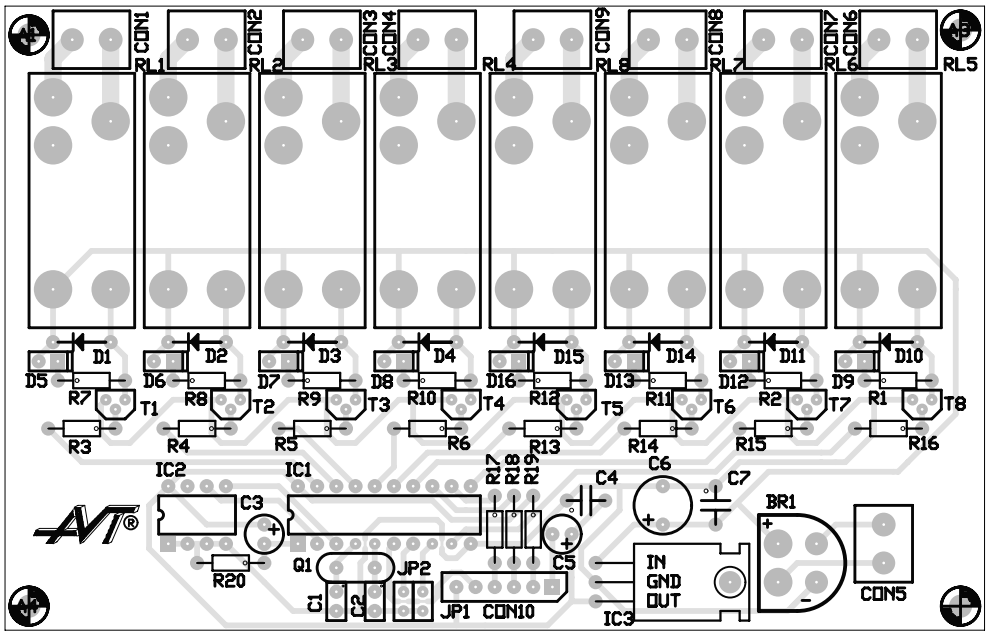
Podobnie jak w przypadku 1, rozpoczęcie rejestracji kodu sygnalizowane jest dziesięcioma błyskami czerwonej diody LED. Także struktura podprogramu rejestrującego kod jest bardzo podobna do programu rejestracji komend i nie ma sensu go tu szczegółowo opisywać. Jedyną istotną różnicą jest to, że wprowadzić możemy do 210 cyfr (lub innych komend wysyłanych z pilota, będących składnikami kodu). Rejestracja kodu dostępu może zostać zakończona na dwa sposoby:

1. Po przekroczeniu maksymalnej liczby składników kodu program automatycznie kończy rejestrację i zawiadamia o tym fakcie włączeniem zielonej diody LED na okres 3 sekund.

2. Podczas rejestrowania kodu dostępu program ustawicznie sprawdza stan wejścia Jumper1. W przypadku wykrycia na nim stanu wysokiego, czyli faktu usunięcia zworki, kończy rejestrację. Umożliwia to wprowadzenie mniejszej niż maksymalna liczba składników kodu.

Należy tu wspomnieć jeszcze o jednej sprawie: opisując rejestrację kodu używałem określenia „składniki kodu", a nie „cyfry kodu". Wynika to z faktu, że jako elementy kodu dostępu mogą być użyte **wszystkie polecenia wysyłane z dowolnego pilota RC5**, a nie tylko cyfry z klawiatury numerycznej.

Przed wyjściem z podprogramu rejestracji kodu dostępu sprawdzany jest jeszcze stan wejścia Jumper2, decydującego o trybie pracy: z kodem dostępu lub bez. Wykrycie na tym wejściu stanu niskiego powoduje przejście do oczekiwania na podanie właściwego kodu. Stan wysoki na tym wejściu spowoduje, że pomimo zaprogramowania kodu dostępu, program nie będzie żądał jego podawania i od razu przejdzie do oczekiwania na odebranie komendy sterującej.



Rys. 2. Rozmieszczenie elementów na płytce drukowanej dekodera.

Przypadek 3

Wykrycie stanów wysokich na wejściach Jumper1 i Jumper2 procesora powoduje wywołanie podprogramu oczekiwania na podanie z pilota komendy sterującej pracą przekaźników:

```
Sub Waitingforcommand
Count = 0
Enable Int0
New = 0
Do
   If New = 1 Then
     Disable Int0
     Call Reading
     New = 0
     Enable Int0
   End If
   Incr Count
   If Count = 100 Then
     If Flag2 = 1 Then
       Call Waitingforcode
     End If
   End If
   Reset Ledgreen
   Waitms 10
   Set Ledgreen
   Wait 1
Loop
End Sub
```

Po ustawieniu wartości zmiennej NEW na 0 program wchodzi w pętlę, w której oczekuje na odebranie transmisji kodu RC5. Jeżeli sygnał taki zostanie wykryty przez odbiornik, to zmienna NEW przyjmuje wartość 1 (patrz podprogram RECEIVERC5) i program przystępuje do analizy odebranego sygnału, przeprowadzanej w podprogramie READING:

```
Sub Reading
Flag1 = 0
Call Read_eeprom (10 , Value)
Adres2 = Value

For Licznik = 1 To 8
   Call Read_eeprom(licznik,
        Value)
   If Adres2 = Subaddress Then
     If Value = Command Then
       Flag1 = 1
       Call Ledgreenshort
       Count = 0
       Call Switch
     End If
   End If
Next Licznik

If Flag1 = 0 Then
   Call Ledredshort
End If
End Sub
```

Pierwszą czynnością wykonywaną przez ten podprogram jest odczytanie z pamięci EEPROM (spod adresu 10) informacji o adresie, pod jaki powinny być wysyłane polecenia nadawane z pilota (wartość tego adresu została uprzednio zapisana w pamięci podczas rejestracji komend). Od tego momentu wartość ta będzie porównywana z adresem aktualnie

odebranego polecenia RC5. Następnie podprogram READING odczytuje, pracując w pętli FOR ... NEXT ze zmienną LICZNIK, wszystkie zapisane w pamięci EEPROM komendy. Po wykryciu zgodności jednej z nich z odebranym poleceniem, generowany jest krótki błysk zielonej diody LED, a następnie wykonywany jest podprogram SWITCH, sterujący już bezpośrednio przekaźnikami.

```
Sub Switch
Select Case Licznik
   Case 1|: Relay1 = Not Relay1
   Case 2|: Relay2 = Not Relay2
   Case 3|: Relay3 = Not Relay3
   Case 4|: Relay4 = Not Relay4
   Case 5|: Relay5 = Not Relay5
   Case 6|: Relay6 = Not Relay6
   Case 7|: Relay7 = Not Relay7
   Case 8|: Relay8 = Not Relay8
End Select
End Sub
```

Każdorazowe odebranie zarejestrowanego polecenia powoduje więc zmianę stanu odpowiadającego mu wyjścia na przeciwny, a tym samym naprzemienne włączanie i wyłączanie dołączonego do niego przekaźnika.

Przypadek 4

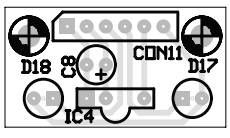
Wykrycie stanu niskiego na wejściu Jumper2, a wysokiego na wejściu Jumper1 procesora świadczy o tym, że zamierzamy zabezpieczyć dostęp do sterownika za pomocą ustawionego wcześniej kodu i program rozpoczyna wykonywanie podprogramu WAITINGFORCODE.

```
Sub Waitingforcode
Do
   Licznik = 20
   Call Warning
   Reset Ledred
   Call Read_eeprom(251 , Value)
   Licznik2 = Value
   Enable Int0
   Count = 0
   Do
     If New = 1 Then
       New = 0
       Call Ledgreenshort
       Waitms 250
       Disable Int0
       Call Read_eeprom(licznik,
         Value)
       Incr Licznik
     End If

     If Value<>Command Then
       Flag1 = 0
     Else
       Count = 0
     End If

     If Flag1 = 1 Then
       If Licznik = Licznik2 Then
         Reset Ledgreen
         Set Ledred
         Wait 2
         Set Ledgreen
         Flag2 = 1
         Call Waitingforcommand
       End If
     End If

     Enable Int0
     Incr Count
     If Count = 60000 Then
       Exit Do
     End If
   Loop
Loop
End Sub
```



Rys. 3. Rozmieszczenie elementów na płytce drukowanej odbiornika sygnału podczerwieni.

Wejście do podprogramu analizy podawanego z pilota kodu dostępu do sterownika sygnalizowane jest dziesięciokrotnym błyskiem czerwonej diody LED, po czym dioda ta zostaje włączona na stałe. Następnie program odczytuje z pamięci EEPROM zapisaną tam podczas rejestracji kodu liczbę elementów występujących w kodzie (Call Read_eeprom(251 , Value)) i po ustawieniu wartości zmiennych pomocniczych wchodzi w kolejną pętlę, w której oczekuje na wprowadzenie szyfru.

Odebranie kodu RC5 nadanego z pilota potwierdzone zostaje krótkim włączeniem zielonej diody LED, a następnie program odczytuje wartość pierwszego elementu kodu zapisaną w pamięci EEPROM i porównuje ją z odebraną wartością. Jeżeli wynik porównania jest negatywny, to zmienna FLAG1 przyjmuje wartość 0. Łatwo zauważyć, że już w tym momencie uzyskanie dostępu do sterownika stało się niemożliwe, nawet jeżeli dalsze elementy kodu zostaną poprawnie podane. Warunkiem „otworzenia zamka" jest bowiem to, aby wszystkie elementy kodu zostały podane poprawnie, we właściwej kolejności i aby liczba wysłanych z pilota sygnałów ściśle odpowiadała liczbie elementów kodu.

Jednak zawsze istnieje możliwość pomyłki i błędnego wprowadzenia kodu, nawet przez znającego go, uprawnionego użytkownika. Zwróćmy teraz uwagę na zmienną COUNT, której wartość jest zwiększana o 1 przy każdorazowym wykonaniu wewnętrznej pętli programu i jest zerowana podczas odebrania kodu RC5 nadanego przez pilota. Po wprowadzeniu błędnego kodu program nadal pracuje w pętli wewnętrznej i jeżeli nie będziemy teraz już więcej naciskać przycisków pilota, to po pewnym czasie zmienna COUNT osiągnie wartość 60000, co spowoduje wyjście programu z pętli wewnętrznej i ponowne przygotowanie zmiennych pomocniczych do analizowania wprowadzanego kodu. Ponowne błyski czerwonej diody zasygnalizują nam, że możemy dokonać kolejnej próby podania kodu dostępu do sterownika.

Jeżeli jednak wprowadzony przez nas kod był prawidłowy, to wywołany zostanie opisany już wyżej podprogram WAITINGFORCOMMAND, dający nam możliwość sterowania dołączonymi do układu urządzeniami. Tu jednak pojawia się pewien problem: wybraliśmy prawidłowy kod dostępu, dokonaliśmy potrzebnych operacji włączając lub wyłączając odpowiednie urządzenia i co dalej? Przecież nie możemy pozostawić sterownika nie zabezpieczonego, dostępnego dla każdego posiadacza pilota RC5! Na szczęście ten problem został rozwiązany stosunkowo prosto, metodami programowymi. Zauważmy, że w momencie prawidłowego podania kodu dostępu, zmienna pomocnicza FLAG2 przyjęła wartość 1. Wróćmy jeszcze na chwilę do podprogramu WAITINGFORCOMMAND. Zauważmy, że i tam występuje zmienna pomocnicza COUNT zerowana w momencie wejścia do podprogramu i zwiększająca swoją wartość przy każdym przejściu przez pętlę programową. W momencie kiedy zmienna ta osiągnie wartość 100 (czyli po ok. 100 sekundach, ze względu na zasto-

WYKAZ ELEMENTÓW

Rezystory
R1, R2, R7..R12: 1,2kΩ
R3..R6, R13..R16: 10kΩ
R17, R18: 560Ω
R19: 220Ω
R20: 300kΩ

Kondensatory
C1, C2: 33pF
C3: 1µF/16V
C4, C7: 100nF
C5, C8: 100µF/10V
C6: 1000µF/16V

Półprzewodniki
BR1: mostek prostowniczy 1A
D1..D4, D10, D11, D14, D15: 1N4148
D5..D9, D12, D13, D16..D19: diody LED
IC1: zaprogramowany procesor AT89C2051
IC2: AT24C04 lub odpowiednik
IC3: 7805
IC4: TFMS5360
T1..T8: BC548

Różne
CON1..CON9: ARK2
JP1, JP2: dwa goldpiny + jumper
Q1: rezonator kwarcowy 11,059MHz
RL1..RL8: RM96

** Diody D5..D9, D12, D13, D16 nie wchodzą w skład kitu*

sowanie opóźnienia WAIT 1) nastąpi powrót do podprogramu WAITINGFORCODE, czyli do ponownego zablokowania układu i oczekiwania na wprowadzenia kodu dostępu. Stanie się jednak tak wtedy i tylko wtedy, kiedy zmienna pomocnicza FLAG2 ma wartość 1.

Montaż i uruchomienie

Na **rys. 2** znajduje się schemat montażowy płytek drukowanych wykonanych na laminacie jednostronnym. Na większej płytce umieszczona została główna część układu, zawierająca procesor, przekaźniki, zasilacz i diody sygnalizujące stan przekaźników. Na mniejszej płytce znajduje się tylko odbiornik kodu RC5 i dwie diody sygnalizacyjne. Takie rozmieszczenie elementów i podział układu na dwa moduły umożliwia zamontowanie głównej części urządzenia w obudowie, która niekoniecznie musi spełnić wysokie wymagania estetyczne, i umieszczenie całości pod przysłowiową szafą. Jedynie część układu zawierająca odbiornik RC5 musi być zlokalizowana w widocznym miejscu, ale sądzę, że z łatwością znajdziecie jakąś estetyczną obudowę dla tej malutkiej płytki.

Montaż układu wykonujemy typowo, rozpoczynając od wlutowania elementów o najmniejszych gabarytach, a kończąc na kondensatorach elektrolitycznych i przekaźnikach. Pod układy scalone należy zastosować podstawki. Dyskusyjna jest sprawa stosowania diod LED sygnalizujących włączenie poszczególnych przekaźników. Na etapie testowania układu mogą one okazać się użyteczne, tak jak użyteczne były podczas pisania programu. Natomiast podczas eksploatacji urządzenia, po umieszczeniu go w obudowie, diody te przestaną pełnić jakąkolwiek użyteczną funkcję i ich stosowanie wydaje się być zbędne. Dlatego też te elementy nie wchodzą w skład kitu.

Obie płytki należy połączyć ze sobą za pomocą sześciożyłowego kabla, najlepiej tzw. taśmowego. Układ prototypowy testowany był z kablem o długości 2,5 m i nie stwierdzono jakichkolwiek zakłóceń w przekazywaniu informacji z odbiornika RC5 do procesora.

Obsługa

1.Programowanie komend obsługujących osiem sterowanych urządzeń

Aby zaprogramować komendy, należy wyłączyć zasilanie układu i zewrzeć jumper JP1. Jeżeli programowanie wykonujemy po raz pierwszy, to zwieranie jumpera nie jest konieczne, ponieważ procesor sprawdza zawartość pamięci EEPROM i po stwierdzeniu, że nie była zaprogramowana, automatycznie przechodzi w tryb programowania komend. Następnie ponownie włączamy zasilanie układu.

Bezpośrednio po włączeniu zasilania układ zasygnalizuje dziesięcioma błyskami czerwonej diody konieczność wysłania z pilota dziesięciu różnych komend, odpowiednio dla każdego ze sterowanych urządzeń. Następnie pojedynczy błysk diody zielonej stanie się zachętą do podania pierwszej komendy. Po naciśnięciu przycisku pilota, zarejestrowanie pierwszej komendy zostanie potwierdzone dwoma błyskami zielonej diody, co jednocześnie jest zachętą do podania drugiej komendy, a następnie trzeciej (trzy błyski zielonej diody) i tak dalej, aż do zarejestrowania wszystkich ośmiu poleceń. Jest całkowicie obojętne, które klawisze wykorzystamy do przesyłania poleceń do sterownika, tak jak obojętne jest, pod jaki adres będą wysyłane. Jednak adres ustawiony w pilocie musi być jednakowy dla wszystkich komend, a także kolejnych elementów kodu dostępu, o ile taki będzie stosowany.

Zarejestrowanie wszystkich ośmiu poleceń zostanie przez sterownik potwierdzone włączeniem na 3 sekundy zielonej diody LED. Następnie układ przechodzi w stan oczekiwania na polecenia, w którym pozostanie (o ile nie stosujemy opcji z kodem dostępu) aż do ewentualnego wyłączenia zasilania. Po zaprogramowaniu poleceń należy zawsze pamiętać o usunięciu jumpera JP1, ponieważ w przeciwnym przypadku układ wchodziłby w tryb programowania komend po każdym wyłączeniu zasilania (nawet przypadkowym).

2. Programowanie kodu dostępu

Aby zaprogramować kod dostępu, należy wyłączyć zasilanie układu i zewrzeć jumper JP1 i JP2. Po zgłoszeniu się programu rejestrującego kod (10 błysków czerwonej diody LED) wprowadzamy kolejno składniki kodu, czyli najlepiej polecenia wysyłane z klawiatury numerycznej pilota. Po wprowadzeniu kodu usuwamy jumper JP1 lub, aby przejść do pracy bez blokowanie dostępu do układu, obydwa jumpery i ponownie włączamy zasilanie układu.

Zbigniew Raabe, AVT
zbigniew.raabe@ep.com.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/pcb.html*** *oraz na płycie CD-EP09/2000 w katalogu* ***PCB****.*

Pliki źródłowe programu znajdują się na płycie CD-EP09/2000 oraz na naszej stronie www.