Zabawka - programowany pojazd, część 1

AVT-5051

Nie wiem dlaczego we wszystkich pismach przeznaczonych dla elektroników obserwuje się kompletny brak zainteresowania zabawkami! Opisywane są same „strasznie mądre“ rzeczy, skomplikowane przyrządy laboratoryjne, niezliczone regulatory wszystkiego, co tylko można regulować, programatory i emulatory, a praktycznie nikt nie zauważa arcyciekawej dziedziny techniki związanej z zabawkami. Nie wiem jaka jest tego przyczyna, ale mogę jedynie domyślać się, że jest nią strach, strach przed bezlitosnymi krytykami, jakimi są dzieci. Dzieci z natury nie są grzeczne (grzeczne dziecko to chore dziecko) i taktowne, nie silą się na wymuszone komplementy i jeżeli ofiarowana im zabawka nie znajduje ich uznania, to po prostu rzucają ją w kąt! Najwyższy jednak czas przemóc ten lęk i pomyśleć o zbudowaniu ciekawej zabawki elektronicznej, jeżeli nie dla dzieci, to chociaż dla siebie. Spróbuję.

Budowanie zabawek ma jeszcze jeden sens. To właśnie od prostych zabawek wywodzi się wiele nowatorskich urządzeń i olśniewających pomysłowością rozwiązań konstrukcyjnych. Nie mam nawet zamiaru porównywać zbudowanej przeze mnie zabaweczki z bardzo ciekawym, choć z użytkowego punktu widzenia dość kontrowersyjnym wynalazkiem, jakim jest tak reklamowany w mediach Segway. Jeszcze raz podkreślam, że nie porównuję obydwóch urządzeń, ale stwierdzam, że i mnie i konstruktorowi Segway'a przyświecała ta sama idea: zbudowanie pojazdu, który byłby zdolny do sprawnego poruszania się na tylko dwóch kołach, i to w dodatku ustawionych obok siebie. Być może mój utytułowany Kolega z USA także zaczynał od budowy modelu - prostej zabawki?

Na całym świecie bowiem najtęższe umysły elektroników pracują nad wymyślaniem nowych zabawek. Przemysł, nie bacząc na wysoki stopień ryzyka, inwestuje miliony w nowe pomysły, a my ciągle jesteśmy zbyt „poważni“, aby zająć się „dziecinnymi“ sprawami. Stan naszego rynku zabawkarskiego jest więcej niż przerażający: odwiedzając sklepy z zabawkami widzimy głównie koszmarki typu Barbie lub tandetne kopie urządzeń do zabijania ludzi. Zdaję sobie sprawę, że wykonanie atrakcyjnej zabawki jest sprawą bardzo trudną, głównie ze względu na problemy związane z obudową i układami mechanicznymi. Nie mamy więc większych szans, aby w warunkach amatorskich, a nawet w dobrze wyposażonym laboratorium, wykonać coś w rodzaju słynnego Furbie. Warto jednak zająć się zabawkami o prostej konstrukcji mechanicznej, których cała „inteligencja“ umieszczona jest w układzie elektronicznym. Problemy z mechaniką zawsze można jakoś „obejść bokiem“.

Proponowany układ do zabawki jest prostym sterownikiem pojazdu mechanicznego, zrealizowanym z wykorzystaniem popularne-

go procesora AVR typu AT90S2313. Sterownik przystosowany jest do współpracy z najprostszym układem napędowym, jaki jest tylko możliwy: pojazd jest napędzany i jednocześnie kierowany za pomocą dwóch niezależnych silników. Każdy z silników napędza jedno koło, a różnica w ich prędkości obrotowej powoduje zmiany kierunku ruchu pojazdu. Rozwiązanie takie, powszechnie stosowane w pojazdach gąsienicowych, eliminuje konieczność stosowania skomplikowanego mechanizmu skręcania kół przednich pojazdu oraz mechanizmu różnicowego, bardzo trudnych do wykonania w warunkach domowego warsztatu.

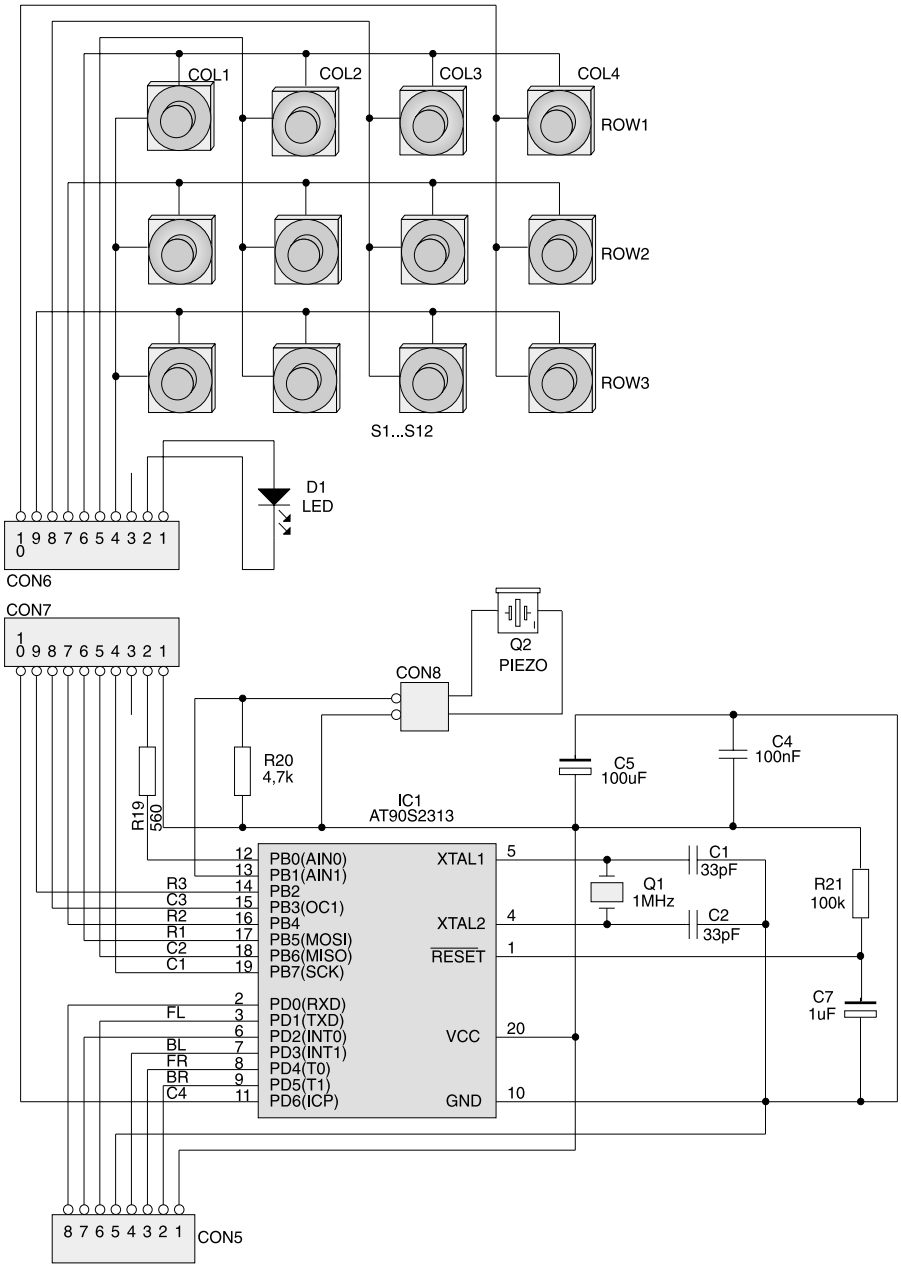
Konstrukcja mechaniczna pojazdu jest na tyle oryginalna, aby nie powiedzieć ekstrawagancka, że moi dowcipni Koledzy z redakcji Elektroniki Praktycznej nazwali ją *Raabowozem*! No cóż, niech im i tą moją krzywdę Bogowie wybaczą!

Zabawka ma rzeczywiście wyjątkowo oryginalną budowę i wygląda trochę jak pojazd ze Star Wars, co z pewnością już zauważyliście na zdjęciu. Jednak wykonanie jej zespołów mechanicznych nie powinno nikomu nastręczyć większych trudności, ponieważ największy problem, jakim jest wykonanie przekładni mechanicznej możemy tym razem ominąć, stosując w roli przekładni napędowej przerobione serwomechanizmy modelarskie.

Zabawka jest pojazdem dwukołowym, z tym że koła nie są ustawione tak, jak w pojazdach jednośladowych, ale umieszczone są jedno obok drugiego. Jak taki pojazd może się w ogóle poruszać? A może, wykorzystując do tego ogólnie znane prawa fizyki. Jednak to wyjaśnimy sobie nieco później, w części artykułu dotyczącej montażu zabawki. Teraz najwyższa pora wyjaśnić Czytelnikom, co właściwie potrafi robić pojazd, z którego opisem zapoznamy się za chwilę.

Pojazd sterowany za pomocą naszego układu może wykonywać następujące manewry:

1. Jazda do przodu.
2. Jazda do tyłu.
3. Skręt w prawo.
4. Skręt w lewo.



Rys. 1. Schemat elektryczny układu sterownika programowanego pojazdu.

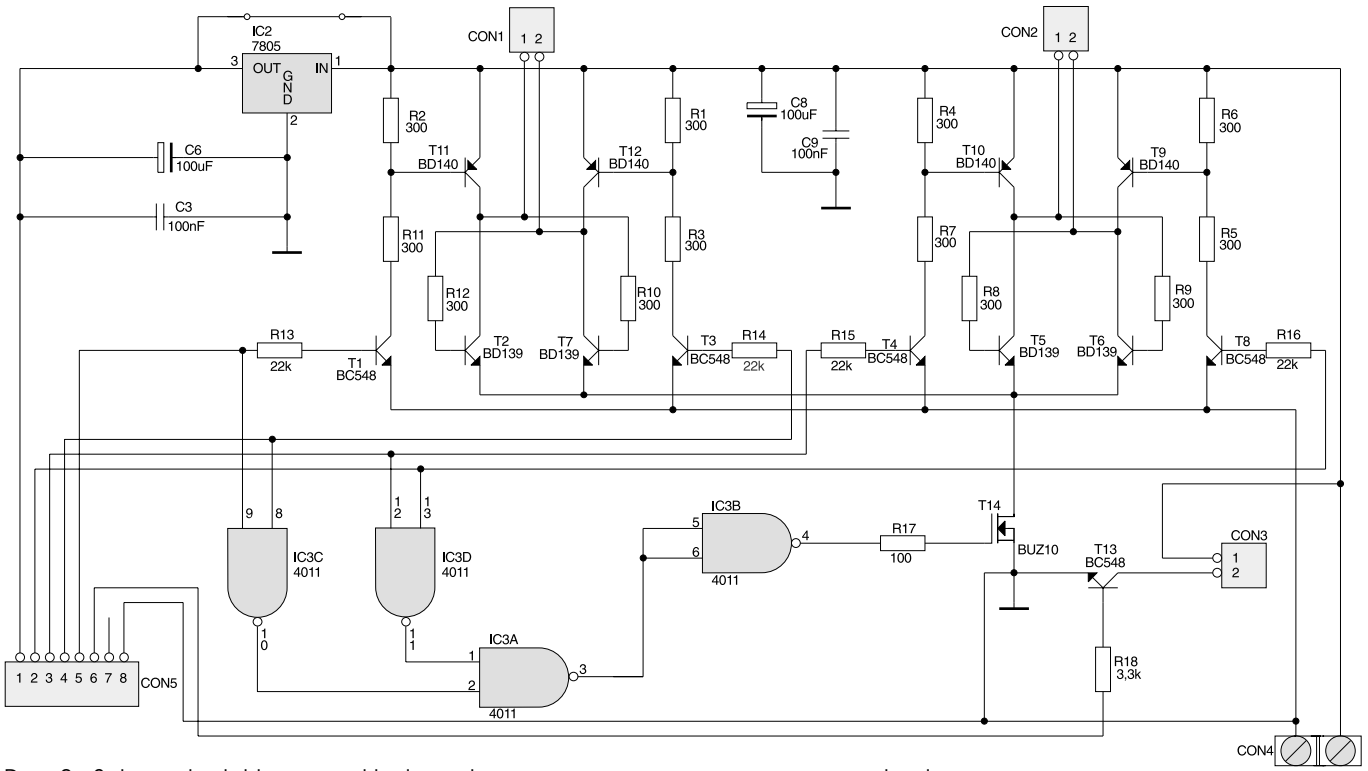
5. Skręt do tyłu w prawo.
6. Skręt do tyłu w lewo.
7. Obrót dookoła osi w prawo.
8. Obrót dookoła osi w lewo.
9. Zatrzymanie pojazdu.

Możliwe jest także korzystanie z funkcji dodatkowej, np. włączania świateł lub sygnału akustycznego.

Przed rozpoczęciem zabawy układ musi zostać zaprogramowany (co także jest niezłą zabawą), czyli „nauczony" jakie ruchy, w jakiej kolejności, i z jaką szybkością ma wykonać. Możliwe jest zaprogramowanie do 125 poruszeń, co przy najszybszym ich wykonywaniu daje całkowity czas realizacji programu równy 125 sekundom, czyli ponad 2 minuty zabawy.

Nasze pociechy bywają bardzo roztrzepane i pozostawianie przez nie zabawek z włączonym zasilaniem jest właściwie regułą. Ponieważ dobrej jakości baterie, służące do zasilania zabawki, nie należą do najtańszych, przewidziałem odpowiedni środek zaradczy: samoczynne wyłączanie się układu w przypadku braku „zainteresowania" zabawką trwającego dłużej niż kilka minut.

Zastosowanie procesora, zawierającego w sobie całą „inteligencję" sterownika, pozwoliło na



Rys. 2. Schemat elektryczny układu wykonawczego programowanego pojazdu.

znaczne uproszczenie pozostałej części układu, którą zaprojektowano z wykorzystaniem zaledwie garstki elementów dyskretnych.

Konstruując układ elektronicznej zabawki miałem jeszcze jeden cel na uwadze: bliższe zapoznanie Czytelników z programowaniem w pakiecie BASCOM AVR, za pomocą którego przygotowałem oprogramowanie sterujące pracą sterownika. Program dla procesora AT90S2313 został napisany, w całości przetestowany w symulatorze programowym i sprzętowym oraz skompilowany w środowisku BASCOM-a AVR. Praca ta nie zajęła mi więcej niż trzy godziny. Sądzę, że już tylko te zalety BASCOM-a - szybkość pracy i możliwość testowania większości programów w symulacji sprzętowej (w uruchamianym układzie) - powinny zachęcić Czytelników do korzystania z tego rewelacyjnego programu.

Aby przybliżyć Wam zasady pisania programów w MCS BASIC, opis działania zabawki został przygotowany głównie w oparciu o obszerne fragmenty kodu źródłowego programu napisanego w tym języku.

Programowanie zabawki wykonywane jest za pomocą 12-przyciskowej klawiatury. Gotowy program przechowywany jest w wewnętrznej pamięci procesora przez dowolnie długi czas, nawet po wyłączeniu zasilania. Zapisany program można zmienić tylko poprzez ponowne zaprogramowanie.

Opis działania układu

Schemat układu elektrycznego zabawki został pokazany na **rys. 1** i **2**. Na rys. 1 przedstawiono sterownik zabawki, a na rys. 2 zbudowany na tranzystorach układ wykonawczy wraz z zabezpieczeniem przeciwzwarciowym. Omawianie działania układu elektrycznego zabawki rozpoczniemy od części wykonawczej.

Część wykonawcza zabawki jest typowo skonstruowanym sterownikiem dwóch silników prądu stałego, sterowanych za pomocą czterech sygnałów cyfrowych. Silniki włączone są w przekątne mostków utworzonych przez tranzystory mocy typu BD139 i BD140. Do złącza CON5 jest dołączony układ mikroprocesorowy, przedstawiony na rys. 1, z którego budową zapoznamy się za chwilę. Rozpatrzmy teraz, co się stanie, jeżeli na przykład na styku 5 CON5 pojawi się wysoki poziom napięcia. Łatwo zauważyć, że wówczas będzie spolaryzowana baza tranzystora T1, a także tranzystorów T7 i T11. Prąd popłynie na drodze: „+“ zasilania, tranzystor T11, uzwojenie silnika dołączonego do złącza CON1, tranzystor T7 i masa zasilania. Silnik przyłączony do CON1 zacznie obracać się (umownie) w stronę obrotu wskazówek zegara. Ponieważ nasz pojazd posiada dwa silniki napędowe, zacznie on skręcać (umownie) w lewo.

Ustawmy teraz poziom wysoki na dwa wejścia złącza CON5: 5 i 3. Włączone zostaną dwa tranzystory: T1 i T4, co spowoduje przewodzenie także tranzystorów T11, T7, T10 i T8 i obracanie się dwóch silników w tę samą stronę. Nasz pojazd zacznie poruszać się do przodu (lub do tyłu - kierunek ruchu zostanie ostatecznie ustalony doświadczalnie podczas montażu pojazdu).

Sądzę, ze uważni Czytelnicy zauważyli już pewne niebezpieczeństwo, tkwiące w naszym układzie. Co bowiem się stanie, jeżeli poziom wysoki wystąpi jednocześnie na wejściach 5 i 4 CON5? Ano, będzie to piękne zwarcie w układzie, spowodowane jednoczesnym przewodzeniem wszystkich tranzystorów mostka! Oczywiście, przy poprawnie napisanym programie taka sytuacja nie powinna wystąpić, ale nie wszystkie programy napisane są od razu poprawnie...

List. 1.
```
Sub mainprogram

   Readeeprom Value, 127          'sprawdzanie zawartości adresu 127 pamięci danych

   If Value <> 44 Then            'jeżeli zapisana tam wartość nie jest równa 44 to:
      Call Record                 'wezwij program rejestrowania poleceń
   End If

   Timecounter = 0

   Do
      Call Keyscan                'wezwij podprogram przeszukiwania klawiatury
      If Digit = 10 Then          'jeżeli naciśnięty został klawisz o wartości 10 to:
         Call Ledshort            'wygeneruj sygnał akustyczny i optyczny
         Waitms 100
         Call Record              'wezwij program rejestrowania poleceń
      End If

      If Digit = 11 Then          'jeżeli naciśnięty został klawisz o wartości 11 to:
         If Value = 44 Then       'jeżeli pamięć danych została już uprzednio zaprogramowana to:
            Call Ledshort         'wygeneruj sygnał akustyczny i optyczny
            Waitms 100
            Call Replay           'wezwij podprogram odtwarzania poleceń
         End If
      End If

      Waitms 50
      Incr Timecounter

      If Timecounter = 2400 Then 'jeżeli zmienna TIMECOUNTER ma wartość 2400, to:
         Timecounter = 0          'zmienna TIMECOUNTER przyjmuje wartość 0
         For R = 1 To 20
            Call Ledshort         'wygeneruj sygnał akustyczny i optyczny
         Next R
         Powerdown                'wprowadź procesor w stan uśpienia
      End If
   Loop

End Sub
```

Aby więc zabezpieczyć się przed zwarciem i jego zwykle przykrymi konsekwencjami, do części wykonawczej zabawki dobudowany został obwód z bramkami NAND zawartymi w strukturze IC1 i tranzystorem T14 zasilającym mostki tranzystorowe od strony minusa. Bramki IC2A i IC2B wykrywają stany zabronione, które mogłyby wystąpić na wyjściach części sterującej. Wystąpienie poziomu niskiego na wyjściu jednej lub obu tych bramek powoduje natychmiastowe wyłączenie tranzystora T14 i wyeliminowanie niebezpieczeństwa powstania zwarcia w układzie.

Układ może być zasilany napięciem stałym o wartości 5...16VDC, zależnym głównie od typu zastosowanych silników. Z tego też względu stabilizator napięcia IC2 jest elementem opcjonalnym i w przypadku korzystania z napięcia o wartości zbliżonej do 5V nie musi być stosowany. Tranzystor T13 może włączać lub wyłączać układ dodatkowy: sygnalizator optyczny, akustyczny lub jakiś silnik realizujący dodatkową funkcję.

Popatrzmy teraz na rys. 1, na którym przedstawiono mikroprocesorowy sterownik odpowiedzialny za działanie całego urządzenia. Sercem układu i jednocześnie jego jedynym aktywnym elementem jest procesor typu AT90S2313.

Układ AT90S2313 jest nowoczesnym mikroprocesorem opartym na architekturze RISC. Z pozoru układ wygląda zupełnie podobnie jak znany Wam dobrze procesor AT89C2051. Rzeczywiście, procesory te posiadają identyczny rozkład wyprowadzeń i pełnią one w zasadzie identyczne funkcje. Już w tym momencie możemy zauważyć pierwszą zaletę 90S2313: może on bez większych przeróbek być zastosowany w każdym urządzeniu zaprojektowanym dla procesora '2051, zwiększając jego szybkość działania i pozwalając na rezygnację z niektórych elementów zewnętrznych (np. pamięci danych EEPROM). Jedyną modyfikacją jaką musielibyśmy wprowadzić wymieniając procesor byłaby zmiana sposobu zerowania układu: procesory rodziny AVR zerowane są bowiem niskim poziomem napięcia.

Analiza działania układu będzie jednocześnie skrótowym omówieniem sterującego nim programu, napisanego i skompilowanego za pomocą pakietu BASCOM AVR.

Program sterujący zabawką zaczyna się tak, jak każdy inny napisany w MCS BASIC: od deklaracji zmiennych i podprogramów. Należy jednak zwrócić uwagę na pewną istotną różnicę, wynikającą z odmienności architektury procesorów '51 i AVR. W programie pisanym dla procesora AVR musimy zawsze zadeklarować funkcje pełnione przez porty lub pojedyncze wyprowadzenia: czy mają być używane jako wejścia, czy jako wyjścia. W naszym przypadku deklaracja ta będzie wyglądała następująco:

```
Config Pinb.0 = Output: Config Pinb.7 = Input: Config Pinb.6 = Input: Config Pinb.3 = Input
Config Pind.6 = Input: Config
```

List. 2.
```
Sub record
   Value = 44
   Licznik = 1                'ustawienie licznika poleceń
   Digit = 255
   Do                         'wejście w pierwszą pętlę podprogramu rejestracji poleceń
      Call Keyscan            'wezwij przeszukiwanie klawiatury
      If Digit < 10 Then      'jeżeli naciśnięty został klawisz o wartości mniejszej niż 10, to:
         Writeeeprom Digit,126          'zapisz pod adresem 126 wartość opóźnienia czasowego
         For R = 1 To Digit      'tyle razy, ile sekund wynosi opóźnienie:
            Call Ledshort        'wygeneruj sygnał akustyczny i optyczny
         Next R
         Digit = 255       'zmienna DIGIT przyjmuje wstępną wartość 255
         Exit Do           'wyjdź z pierwszej pętli
      End If
   Loop

   Do                         'wejście w drugą pętlę podprogramu rejestracji poleceń
      Call Keyscan            'wezwij przeszukiwanie klawiatury
      If Digit < 10 Then      'jeżeli naciśnięty został klawisz o wartości mniejszej niż 10, to:
         Writeeeprom Digit,Licznik      'zapisz wartość tego klawisza w pamięci EEPROM
         Incr Licznik
         Call Ledshort        'wygeneruj sygnał akustyczny i optyczny
         Digit = 255
      End If

      If Licznik = 125 Then   'jeżeli zmienna LICZNIK przyjęła wartość 125 (zapisanie całej
                              'pamięci), to:
         Writeeeprom Value,127       'zapisz do pamięci wartość świadczącą o jej zaprogramowaniu
         Writeeeprom Licznik,125     'zapisz do pamięci informację o liczbie zaprogramowanych poleceń
         For R = 1 To 5
            Call Ledshort     'wygeneruj sygnał akustyczny i optyczny
         Next R
         Digit = 255
         Call Mainprogram     'powrót do programu głównego
      End If

      If Digit = 12 Then      'jeżeli naciśnięty został klawisz o wartości 12 (koniec zapisu), to:
         Writeeeprom Value,127       'zapisz do pamięci wartość świadczącą o jej zaprogramowaniu
         Writeeeprom Licznik,125     'zapisz do pamięci informację o ilości zaprogramowanych poleceń
         For R = 1 To 5
            Call Ledshort     'wygeneruj sygnał akustyczny i optyczny
         Next R
         Digit = 255
         Call Mainprogram     'powrót do programu głównego
      End If
   Loop

End Sub
```

List. 3.

```
sub replay

   For R = 1 To 3
      Call Ledshort           'wygeneruj sygnał akustyczny i optyczny
   Next R

   Digit = 0
   Licznik = 1
   Readeeprom Ddelay,126      'odczytaj z pamięci EEPROM wartość opóźnienia pomiędzy
                              'wykonywaniem kolejnych poleceń
   Readeeprom Steps, 125      'odczytaj z pamięci EEPROM, ile poleceń zostało zarejestrowanych

   Do
      Readeeprom Digit,Licznik      'odczytaj z pamięci rodzaj kolejnego polecenia
      If Licznik = Steps Then       'jeżeli zmienna LICZNIK równa jest liczbie zarejestrowanych
                                    'poleceń to:
         Reset Portd.4: Reset Portd.5: Reset Portd.3: Reset Portd.1: Reset Portd.0_
         Reset Portd.2              'ustaw stan niski na wyjściach sterujących silnikami

         For R = 1 To 3
            Call Ledshort           'wygeneruj sygnał akustyczny i optyczny
         Next R
         Exit Do                'wyjdź z pętli programowej
         Call Mainprogram       'wezwij podprogram oczekiwania na polecenie z klawiatury
      End If

      Incr Licznik
      Reset Portd.0: Reset Portd.1: Reset Portd.2: Reset Portd.3: Reset Portd.4: Reset Portd.5

      Select Case Digit          'wybierz rodzaj reakcji układu na odczytane polecenie
         Case 1 : Call Backleft     'jeżeli polecenie miało wartość 1 to skocz do podprogramu
                                    'realizującego poruszanie się pojazdu do tyłu w lewo
         Case 2 : Call Back         'jeżeli polecenie miało wartość 2 to skocz do podprogramu
                                    'realizującego poruszanie się pojazdu do tyłu
         Case 3 : Call Backright    'jeżeli polecenie miało wartość 3 to skocz do podprogramu
                                    'realizującego  poruszanie się pojazdu do tyłu w prawo
         Case 4 : Call Turnleft     'jeżeli polecenie miało wartość 4 to skocz do podprogramu
                                    'realizującego obrót pojazdu w lewo
         Case 5 : Call Sstop        'jeżeli polecenie miało wartość 5 to skocz do podprogramu
                                    'zatrzymującego ruch pojazdu
         Case 6 : Call Turnright    'jeżeli polecenie miało wartość 6 to skocz do podprogramu
                                    'realizującego obrót pojazdu w prawo
         Case 7 : Call Lleft        'jeżeli polecenie miało wartość 7 to skocz do podprogramu
                                    'realizującego poruszanie się pojazdu w lewo
         Case 8 : Call Forward      'jeżeli polecenie miało wartość 8 to skocz do podprogramu
                                    'realizującego poruszanie się pojazdu do przodu
         Case 9 : Call Rright       'jeżeli polecenie miało wartość 9 to skocz do podprogramu
                                    'realizującego poruszanie się pojazdu w prawo
      End Select

      Call Ledshort       'wygeneruj krótki sygnał akustyczny i optyczny
      Wait Ddelay         'zaczekaj zadaną zmienną DDELAY liczbę sekund
   Loop

End Sub
```

```
Pinb.5 = Output: Config
Pinb.4 = Output: Config
Pinb.2 = Output
Config Pind.0 = Output:
Config Pind.1 = Output:
Config Pind.2 = Output:
Config Pind.3 = Output
Config Pind.4 = Output:
Config Pind.5 = Output
```

W następnej kolejności deklarujemy podprogramy realizujące poszczególne funkcje naszej zabawki.

```
Declare Sub Record
'podprogram rejestrowania
'poleceń sterujących zabawką
Declare Sub Replay
'podprogram odtwarzania
'programu sterowania zabawką
Declare Sub Mainprogram
'podprogram oczekiwania na
'polecenia
Declare Sub Keyscan
'podprogram przeszukiwania
'klawiatury
Declare Sub Ledshort
'realizacja sygnalizacji
'optycznej i akustycznej
Declare Sub Forward
'ruch do przodu
Declare Sub Back
'ruch do tyłu
Declare Sub Lleft
'skręt w lewo
Declare Sub Rright
'skręt w prawo
Declare Sub Backleft
'skręt do tyłu w lewo
Declare Sub Backright
'skręt do tyłu w prawo
Declare Sub Turnleft
'obrót dookoła osi w lewo
Declare Sub Turnright
'obrót dookoła osi w prawo
Declare Sub Sstop
'zatrzymanie pojazdu
```

Bezpośrednio po włączeniu zasilania program sterujący pracą pojazdu ustala swoje parametry konfiguracyjne i następnie „wchodzi“ w MAINPROGRAM, gdzie na samym początku sprawdza zawartość komórki 127 pamięci danych EEPROM. Jeżeli wartość zapisana w tej komórce nie jest równa 44, to program przechodzi do podprogramu rejestrowania poleceń. Jeżeli wartość ta wynosi 44, co świadczy że pamięć była już zaprogramowana, to wykonywany jest podprogram wyboru trybu pracy z **list. 1**.

Należy zwrócić uwagę na rolę zmiennej pomocniczej TIME-COUNTER. Przy każdym przejściu przez pętlę programową zwiększa ona swoją wartość o 1, zliczając w ten sposób upływający czas. Jeżeli nikt nie wyda zabawce jakiegoś polecenia, to po ok. 120 sekundach (2400x50ms) przejdzie ona w stan uśpienia, z którego może się obudzić dopiero po powtórnym włączeniu zasilania. Zabezpiecza to przed wyczerpaniem baterii w przypadku porzucenia zabawki przez dziecko lub roztargnionego innego użytkownika.

Rzućmy teraz okiem na podprogramy zapisu danych i ich odtwarzania, czyli sterowania ruchem pojazdu. Rejestrowanie poleceń realizowane jest przez podprogram RECORD (**list. 2**).

Zarejestrowane polecenia możemy następnie odtworzyć we właściwej kolejności. Funkcja ta reali-

WYKAZ ELEMENTÓW

Rezystory

R1...R12: 300Ω
R13...R16: 1,5kΩ
R17: 100Ω
R18: 3,3kΩ
R19: 560Ω
R20: 4,7kΩ
R21: 100kΩ

Kondensatory

C1, C2: 33pF
C3, C4, C9: 100nF
C5, C6, C8: 100µF/16
C7: 1µF/16

Półprzewodniki

D1: dioda LED
IC1: AT90S2313
IC2: 7805
IC3: 4011
T1, T3, T4, T8, T13: BC548
T2, T5...T7: BD139
T9...T12: BD140
T14: BUZ10

Różne

Q: rezonator kwarcowy 11,059200 MHz
Q2: przetwornik piezo
S1...S12: przycisk microswitch
CON4: ARK2 (3,5mm)

zowana jest przez podprogram REPLAY pokazany na **list. 3**.

Podane fragmenty listingu programu sterującego ruchem zabawki powinny dostarczyć Czytelnikom pewnych informacji o budowie całego programu, napisanego w MCS BASIC. Program ten został napisany w najprostszy sposób, „po najmniejszej linii oporu". Zachęcam więc Wszystkich do prób jego modernizacji i ulepszania. Ciekawe mogą być eksperymenty z zastosowaniem „miękkiego startu" silników, zrealizowanego metodą PWM. Rozwiązanie takie pozwoliłoby zlikwidować „kołysanie się" pojazdu podczas rozpoczynania jazdy do przodu i do tyłu. Oczywiście, szczytem perfekcji byłoby dobudowanie do układu czujnika poziomu (np. z serii ADXL), co pozwoliłoby na precyzyjne pozycjonowanie gondoli pojazdu w stosunku do pionu i całkowite zlikwidowanie kołysania.

Zbigniew Raabe, AVT

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/luty02.htm*** *oraz na płycie CD-EP02/2002B w katalogu* ***PCB***.