Wyświetlacz widmowy

kit 2651 AVT

Jest to urządzenie, które umożliwia wyświetlanie w przestrzeni napisów oraz prostych obrazków, może też pełnić funkcję efektownego zegara. Siedem wirujących diod LED daje efekt zawieszonych w powietrzu cyferek. Jest to iluzja, ale jest bardzo interesująca i wygląda miło. Zaprojektowane urządzenie posiada naprawdę duże możliwości. Umożliwia m.in. włączanie i zatrzymanie przesuwu napisów, regulację: prędkości przesuwu napisów, szerokości napisów, szerokości okna, w którym wyświetlane są napisy, programowanie haseł reklamowych, cykliczne wyświetlanie pojedynczego hasła lub zdania, kolejne wyświetlanie wszystkich zaprogramowanych haseł reklamowych. Pełny opis funkcji oraz sposób programowania zamieszczę na końcu niniejszego tekstu.

Wyświetlacz widmowy jest podobny do opisanego w EdW (numer 7/2002) µProjektu 3000 o nazwie *Widmo, albo magiczna różdżka.* Podobny w tym sensie, że wykorzystuje ten sam efekt stroboskopowy, czyli powstały efekt bazuje na wadzie ludzkiego oka, jaką jest niewątpliwie inercja – bezwładność. Urządzenie sterowane jest za pomocą dowolnego pilota pracującego w standardzie RC5. Jest to naprawdę bardzo wygodny sposób komunikacji, zwłaszcza że urządzenie podczas pracy jest w ciągłym ruchu (wykonuje ruch obrotowy).

Jak to działa?

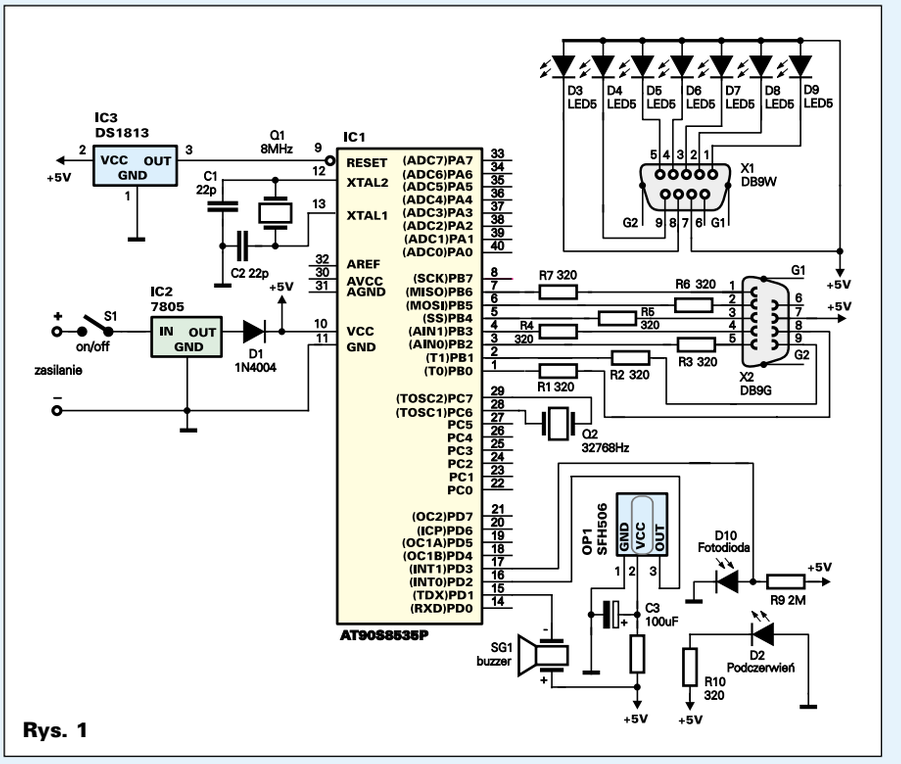
Schemat ideowy urządzenia widoczny jest na **rysunku 1**. Jest to w miarę prosty układ elektryczny, ale zastosowany w nim mikroprocesor AT90S8535AVR umożliwia realizację naprawdę skomplikowanych operacji.

Mikrokontroler ten sprostał zadaniu, które mu powierzyłem, ze względu na swoje możliwości sprzętowe.

Za pracę wyświetlacza odpowiedzialny jest program napisany w języku Bascom AVR ver.1.11.6.4. Program można ściągnąć ze strony internetowej EdW ***www.edw.com.pl*** z działu FTP.

Wykonane przeze mnie urządzenie posiada dwa niezależne źródła zasilania. Część elektroniczna zasilana jest z 9V baterii, która umieszczona została w elemencie wirującym. Jest to bardzo korzystne rozwiązanie, ponieważ pozbyłem się problemu z doprowadzeniem zasilania do części ruchomej, a stosunkowo ciężka bateria zapewniła przeciwwagę dla modułu wyświetlającego. 3 baterie 4,5 V umieszczone w podstawie posłużyły do zasilania silnika napędowego. Opcjonalnie zamiast baterii można dołączyć zewnętrzny zasilacz 12–14 V. Jako element napędowy wykorzystałem silnik z odtwarzacza wideo.

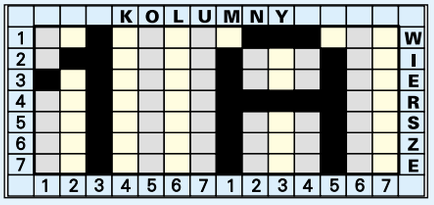
Jak wynika z przedstawionego schematu ideowego, oprócz mikrokontrolera znajdują się tam też inne elementy. Jednym z nich jest odbiornik podczerwieni SFH506, bez którego niemożliwa byłaby komunikacja z urządzeniem za pomocą pilota. Na płytce umieściłem też złącze SPI umożliwiające programowanie mikroprocesora. W praktyce nie jest ono potrzebne, ale na etapie projektowania urządzenia i modyfikacji parametrów może okazać się bardzo pomocne.



Rys. 1

Tu właściwie kończy się sprzęt a zaczyna oprogramowanie. Aby zrozumieć działanie programu, konieczna jest wiedza, jak zbudowane są znaki (literki, cyfry, obrazki). Sposób budowy przedstawia **rysunek 2.** Czarny kwadracik oznacza zapaloną diodę w kolumnie. Poszczególne znaki zostały zapisane w pamięci flash ROM mikrokontrolera. Każdy z nich został zaprojektowany w oparciu o matrycę 7x7. Tak naprawdę kształt opisany jest na pięciu kolumnach, ale dołożenie jeszcze dwóch wyeliminowało problem odstępu pomiędzy znakami. Przykładowa deklaracja kształtu cyfry **1** oraz litery **A** pokazana jest na **listingu 1.** Tak więc wyświetlanie znaków będzie polegało na wysłaniu na port, do którego podpięte są diody, po kolei takich kombinacji liczbowych zsynchronizowanych czasowo.

Rys. 2



Listing 1

```
Dat1:
Data 125 , 126 , 0 , 127 , 127 , 127 , 127
Dat_a:
Data 1 , 118 , 118 , 118 , 1 , 127 , 127
```

Program obsługujący wyświetlacz widmowy po skompilowaniu końcowej wersji zajmuje całą pamięć flash ROM, calutkie 8kB! Składa się on z 26 zadeklarowanych zmiennych, ośmiu procedur (napisy, opoznienie, czytaj_czas, czytaj_napis, zapisz_napis, dźwięk, przesuw_napisu, obsługi przerwania – fotodioda), głównej pętli programowej oraz dwóch bloków danych opisujących kształty znaków.

Po uruchomieniu wyświetlacza program analizuje założenia początkowych wartości zmiennych, zapisanych w pamięci EEPROM, które potrzebne są do „rozruchu" pozostałej części programu. Są to m.in.; czas, od którego zegar startuje zaraz po włączeniu zasilania, długość okna, w którym, wyświetlane są znaki, szerokość znaków, itp. – **listing 2.**

Listing 2

```
Time$ = "12:01:15"
Czas = ""
Readeeprom Zwloka , 0
Waitms 10
If Zwloka = 255 Then
                  Zwloka = 20
                  End If
Readeeprom Dl_okna , 1
Waitms 10
If Dl_okna = 255 Then
                  Dl_okna = 56
                  End If
Rolowanie = 0
Timer_start = 0
Zegar = 1
Ustaw=0
```

W głównej pętli programu, pomiędzy słowami kluczowymi *Do...Loop* umieszczona została funkcja *Getrc5*, której zadaniem jest dekodowanie kodów z pilota oraz same warunki uzależnione od zmiennej *Command*, której przypisywane są wartości kodu. W pętli głównej znajduje się aż 14 różnych warunków. Są to między innymi: ustaw zegar, wpisz tekst, roluj napisy, przełącz zegar/tekst, wywołaj zaprogramowany napis, wywołaj hasło początkowe, zmniejsz/zwiększ prędkość przesuwu napisu, zmniejsz/zwiększ szerokość okna, włącz/wyłącz przesuw napisu, zwęż/rozciąg napis, zapisz do pamięci eeprom. Funkcję Getrc5 wraz z przykładową instrukcją warunkową, która umożliwia nam wejście do procedury „ustaw zegar", widzimy na **listingu 3.**

Po wejściu do tego trybu – **listing 4**, program zaczyna analizę naciskanych kolejno na pilocie klawiszy.

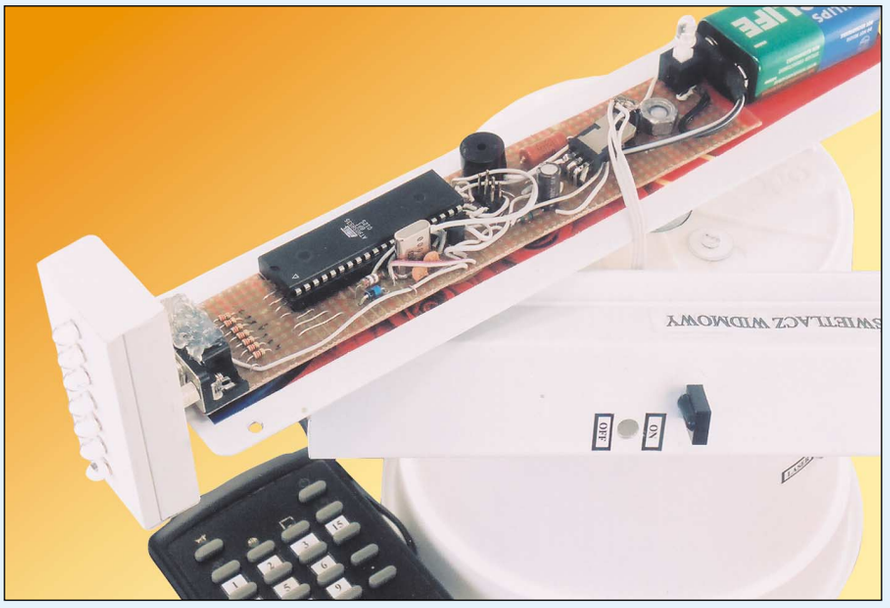
Listing 3

```
Pętla główna:
Do
Getrc5(address       ,       Command)
'czytanie kodu
If Address = 0 Or Address = 10 Then
Command = Command And &B10111111
End If

If Command = 15 And Zegar = 1 Then
'ustawianie zegara
Call Dźwięk
Ustaw = 1
Index = 1
Czas = ""
Portb = &B11111111
Disable Int1
End If
...
Loop
```

Listing 4

```
If Ustaw = 1 And Command < 255 And
Command <> 15 And Zegar = 1 Then
 Call Dźwięk
 Czas = Czas + Str(command)
Wait 1
Select Case Index
Case 1 : Portb = &B11111110
Case 2 : Portb = &B11111100
Case 3 : Portb = &B11111000
Case 4 : Portb = &B11110000
Case 5 : Portb = &B11100000
Case 6 : Portb = &B11000000
Ustaw = 0
Time$ = Mid(czas , 1 , 2) + ":" +
Mid(czas , 3 , 2) + ":" + Mid(czas ,
5 , 2)
Enable Int1
End Select
Incr Index
Else
If Ustaw = 0 And Zegar = 1 Then
Call Czytaj_czas
End If
```



Każde wprowadzenie danej sygnalizowane jest krótkim sygnałem dźwiękowym oraz zapalającą się diodą. Zmienna *Czas* przyjmuje wartości zmiennej *Command* po każdorazowym naciśnięciu klawisza. Następnie końcową postać tej zmiennej (*Czas*) przypisujemy wewnętrznej zmiennej *Time$,* czyli aktualny czas.

Po wprowadzeniu ostatniej cyfry zegara program automatycznie przechodzi do wykonywania procedury „Czytaj_czas" – **listing 5**, która odpowiedzialna jest za pobranie z pamięci danych opisujących kształty cyfr.

Tutaj znowu, tylko w odwrotnej kolejności, zmienna *Czas* równa jest zmiennej *Time$*. Zabiegi tego typu pozwalają na ograniczenie liczby wprowadzanych do programu zmiennych. Następnie dzięki funkcji *Mid*, która znajduje się w „obrębie pracy" dwóch pętli powtórzeń, każdy znak opisujący

zmienną *Czas* przetwarzany jest na ciąg liczb (patrz: listing 1). Zmienna *Dane(index)* równa jest kolejno sczytywanym liczbom z takiego ciągu. Umożliwia nam to zastosowanie dwóch współpracujących ze sobą funkcji *Restore* oraz *Read.*

Listing 5
```
Podprogram dekodowania czasu
Sub Czytaj_czas:
Disable Int1
Index = 1
Czas = Time$
Portb = &B1111111
For I = 1 To 8
G = Mid(czas , I , 1)
Select Case G
Case "0" : Restore Dat0
Case "1" : Restore Dat1
Case "2" : Restore Dat2
Case "3" : Restore Dat3
Case "4" : Restore Dat4
Case "5" : Restore Dat5
Case "6" : Restore Dat6
Case "7" : Restore Dat7
Case "8" : Restore Dat8
Case "9" : Restore Dat9
Case ":" : Restore Dat_dw
End Select
For J = 1 To 7
Read Segment
Dane(index) = Segment
Incr Index
Next J
Next I
Enable Int1
End Sub Czytaj_czas
```

Wyświetlanie znaków odbywa się w przerwaniach, które zgłaszane są za pomocą dwóch współpracujących ze sobą diod. Procedura obsługi przerwania o nazwie *Fotodioda* – **listing 6**, zawiera dwa warunki, uzależnione od zmiennych *Wskaznik* oraz *Dl_slowa*. Ta druga, jak sama nazwa wskazuje, równa jest liczbie znaków w danym słowie, natomiast pierwsza odpowiedzialna jest za przesuw okna, w którym wyświetlane są napisy (gdy niewłączona jest opcja przesuwu napisu, ma ona stałą wartość). Następnie w każdym z tych warunków znajdują się trzy pętle powtórzeń. Jedna odpowiedzialna jest za wysyłanie na port liczby, która sprawi, że diody zgasną, natomiast pozostałe dwie wysyłają na port kolejne wartości zmiennej *Dane (index),* czyli kombinacje liczbowe opisujące kształty znaków.

Teraz chciałbym przybliżyć zasadę działania pozostałych, równie ciekawych procedur, które w swojej budowie są nieco mniej skomplikowane od omawianych wcześniej. I tak procedura *Przesuw_ napisu* podobnie jak procedura *Fotodioda* „pracuje" w przerwaniach, tylko że w tym przypadku przerwanie zgłaszane jest przez wewnętrzny licznik *Timer1*. Wejście do tego trybu jest możliwe po naciśnięciu klawisza na pilocie o kodzie 4. Tak naprawdę włączenie bądź wyłączenie przesuwu napisu zależeć będzie od zmiennej *Timer_start*,

Listing 7
```
If Command = 4 And Ustaw = 0 Then
'wł/wył przesuw
                                Call
Dźwięk
If Timer_start = 0 Then
Wskaznik = 0
Start Timer1
Else
Stop Timer1
Wskaznik = Dl_okna
End If
Timer_start = Not Timer_start
Waitms 255
Waitms 255
Else
```

która, jak wynika z założeń, może przyjmować dwie wartości 0 lub 1. Na początku programu zmienna ta przyjmuje wartość równą 0 – (patrz: listing 2). Zmiana jej wartości realizowana jest za pomocą funkcji *Not,* czyli negacji – **listing 7.**

Wyżej wspomniany licznik na „potrzeby" programu został skonfigurowany jako Timer – **listing 8,** o Prescalerze równym 8, co w

Listing 8
```
On Timer1 Przesuw_napisu
Rem Konfiguracja Portu, Zegara, Rc5
Config Clock = Soft
Config Rc5 = Pind.2
Config Timer1 = Timer, Prescale = 8
```

Listing 6
```
Fotodioda:
Disable Int1

If Wskaznik < Dl_slowa Or Wskaznik = Dl_slowa Then  'warunek 1
               If Wskaznik < Dl_okna Or Wskaznik = Dl_okna Then
                                 Przesuw = Dl_okna - Wskaznik
                                 For Index = 1 To Przesuw
                                 Portb = &B11111111
                                 Call Opoznienie
                                 Next Index
                                 For Index = 1 To Wskaznik
                                 Portb = Dane(index)
                                 Call Opoznienie
                                 Next Index
                                 Portb = &B11111111
                                                          End If

               If Wskaznik > Dl_okna Then
                                 Przesuw = Wskaznik - Dl_okna
                                 For Index = Przesuw To Wskaznik
                                 Portb = Dane(index)
                                 Call Opoznienie
                                 Next Index
                                 Portb = &B11111111
                                       End If

                                       End If                'end warunku 1

If Wskaznik > Dl_slowa Then                                    'warunek 2
               If Wskaznik < Dl_okna Or Wskaznik = Dl_okna Then
                                   Przesuw = Dl_okna - Wskaznik
                                   For Index = 1 To Przesuw
                                   Portb = &B11111111
                                   Call Opoznienie
                                   Next Index
                                   For Index = 1 To Dl_slowa
                                   Portb = Dane(index)
                                   Call Opoznienie
                                   Next Index
                                   Portb = &B11111111
                                                          End If

               If Wskaznik > Dl_okna Then
                                   Przesuw = Wskaznik - Dl_okna
                                   For Index = Przesuw To Dl_slowa
                                   Portb = Dane(index)
                                   Call Opoznienie
                                   Next Index
                                   Portb = &B11111111
                                   Call Opoznienie
                                                        End If
                                 End If                   'end warunk 2

Enable Int1
Return
```

praktyce oznacza, że „przekręci się” po upływie 0,5 sekundy.

Przesuw napisu będzie polegał na stopniowej zmianie położenia okna, w którym wyświetlane są napisy – **listing 9**. Odpowiedzialna za to jest zmienna *Wskaznik*, która po każdorazowym wywołaniu tej procedury będzie zwiększana o 1. Przy okazji omawiania tego fragmentu zdradzę tajemnicę prostego rozwiązania zmiany prędkości przesuwu. Mianowicie, aby szybciej zwiększać wartość zmiennej, *Wskaznik* dodaje do niej zmienną *Predkosc_napisu*, którą zwiększam, naciskając klawisz 5 – **listing 10.**

Listing 9
```
Przesuw_napisu:
Wskaznik = Wskaznik + Predkosc_napisu
Zak_wskaznika = Dl_okna + Dl_slowa
If Wskaznik > Zak_wskaznika Then
          Wskaznik = 0
               If Rolowanie = 1 Then
                    Call Napisy
                    End If
          End If
Return
```

Listing 10
```
If Command = 5 And Ustaw = 0 Then
                    'zwiększa prędkość
          Call Dzwiek
          Incr Predkosc_napisu
          Waitms 250
          Waitms 250
          If Predkosc_napisu > 3 Then
               Predkosc_napisu = 1
                              End If
```

Kolejna procedura o nazwie *Opoznienie* – **listing 11**, odpowiedzialna jest za szerokość wyświetlanych znaków. Niestety, regulacja wysokości znaku jest niemożliwa w realizacji, gdyż ogranicza ją liczba diod (7).

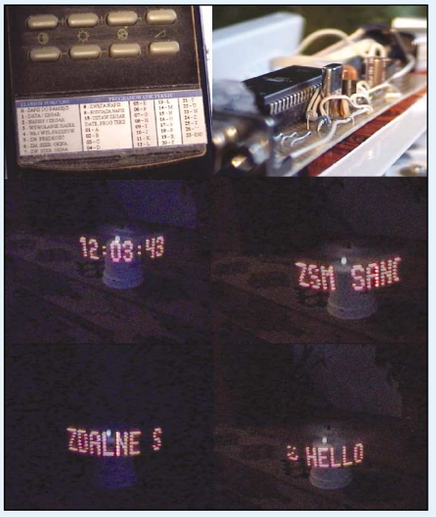
Listing 11
```
Rem Procedura Opoznienie
Sub Opoznienie:
Waitus 250
Waitus 25
For I = 1 To Zwloka
Waitus 1
Next I
End Sub Opoznienie
```

Znajdują się tam dwie „popularne” funkcje *Wait* oraz stosowana już w wielu przypadkach pętla powtórzeń *For...To*. Procedura ta wywoływana jest podczas pracy *Fotodiody* (patrz; listing 5), gdy na port „wyrzucane” są znaki. Zasada jej działania jest bardzo prosta, polega na powtarzaniu w pętli funkcji *Wait* do wartości zmiennej *Zwloka*.

Ostania procedura, którą chcę omówić, różni się od pozostałych tym, że efektu jej pracy nie widzimy, tylko słyszymy. Mowa oczywiście o procedurze *Dzwiek* – **listing 12**. Jak widać, zastosowałem tam funkcję *Sound*, która umożliwia załączenie dźwięku na określony czas trwania. Procedura wywoływana jest przy każdym naciśnięciu klawisza na pilocie.

Tak właśnie to działa. W razie jakiejś niejasności polecam jeszcze raz dokładnie przeanalizować listingi.

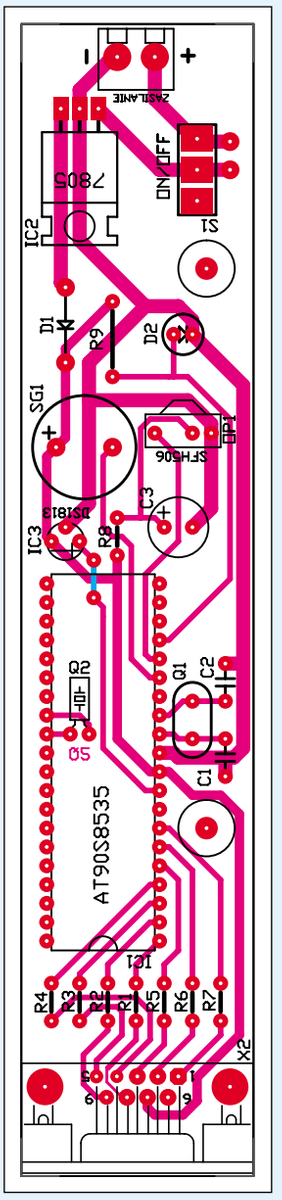
Listing 12
```
Sub Dzwiek:
Sound Bzyczek , 800 , 10
Waitms 200
Set Bzyczek
End Sub Dzwiek
```



Montaż i uruchomienie

W tej części artykułu postaram się Wam udzielić paru wskazówek co do wykonania takiego urządzenia metodą „chałupniczą”. Jak widać na schemacie ideowym, główna część elektroniczna połączona jest z linijką diod za pomocą gniazda DB9. Moja propozycja: płytkę główną wykonać według **rysunku 3**, natomiast diody wraz z wtykiem DB9 połączyć na przysłowiowego „pająka” i umieścić w plastykowej rynience na przewody o wymiarach 32mmx15mm (szerokość x grubość). Wymiary idealnie pasują do wspomnianego już wtyku, poza tym łatwo możemy tam wkomponować diody, wiercąc w przedniej części rynienki symetrycznie rozmieszczone otwory, o średnicy diody LED. Ja osobiście użyłem diod ultrajasnych o średnicy 5mm. Wiercąc otwory pod diody, należy robić to w miarę starannie, aby były one pasowane na wcisk. Po zamocowaniu diod obcinamy rynienkę na długość około 70mm, przy odstępie jednej od drugiej 6mm. Proponuję takie wymiary ze względu na dość ładny kształt znaków i małe opory powietrza. Zatyczki możemy wykonać z twardej tektury lub, jak w moim przypadku, z silikonu białego, który podgrzałem do temperatury, w której staje się płynny, a następnie wypełniłem nim otwory. Linijkę z diodami mamy już gotową. Odkładamy ją i przygotowujemy się do „zabawy” wymagającej umiejętności w posługiwaniu się lutownicą, a mianowicie do wykonania płytki głównej. Jak wiadomo, ta część wyświetlacza posiada własne zasilanie, więc proponuję na początku przylutować stabilizator, a następnie

Rys. 3 Schemat montażowy



doprowadzić do niego zasilanie z baterii i sprawdzić poprawność działania, aby uniknąć przykrych niespodzianek. Baterię możemy umieścić w specjalnym uchwycie, który jest ogólnie dostępny w sklepach elektrycznych lub tak jak w moim przypadku, wykorzystać kabel ze starego zasilacza kostkowego z możliwością ładowania tego typu baterii. Po prostu obciąłem kawałek tego przewodu wraz ze specjalną końcówką. Baterię mocujemy po przeciwnej stronie niż płytkę, za osią obrotu. Dalszy montaż elementów przeprowadzamy w sposób typowy: rezystorki, kondensatorki... i tak do układu IC1. Gotową płytkę trzeba będzie też zabudować, dla estetycznego wyglądu całego urządzenia. Posłuży nam do tego również rynienka na przewody, tylko o innych wymiarach. Na rynku dostępnych jest wiele rodzajów, proponuje dobrać taką, w której „zamkniemy" baterię 9V. Jeżeli ktoś posiada cienką blachę aluminiową, może pokusić się o wykonanie obudowy w kształcie przypominającym przebieg prostokątny. Na górnej części rynienki trzeba jeszcze umocować w miarę blisko osi obrotu odbiornik podczerwieni, który możemy umocować za pomocą kleju, a do płytki podpiąć cieniutkimi kabelkami. Nie ma znaczenia, czy umocujemy go w pionie, czy położymy - nie będzie to miało wpływu na pracę elementu. Jeszcze tylko trzy otwory: jeden na włącznik zasilania, drugi wywiercony u spodu rynienki na fotodiodę, trzeci do mocowania całości na osi wirnika i gotowe! Za nami już półmetek. Kolejnym etapem będzie zdobycie silniczka, najlepiej prądu stałego (ja wykorzystałem stary silniczek z magnetowidu) oraz jakiejś obudowy, w której go umieścimy. Znów propozycja z mojej strony – plastikowa doniczka na kwiatek!, oczywiście + podstawka do niej, która posłuży nam do „zamknięcia" całości od spodu. Wielkość doniczki zależy od konstruktora, który zdecyduje, czy zasilanie będzie pochodziło z baterii, czy z zasilacza. Bieguny napięcia zasilania silnika powinny być tak dobrane, aby wirnik obracał się przeciwnie do ruchu wskazówek zegara. Doniczka jest niewątpliwie dobrym rozwiązaniem ze względu na łatwość dostępu (sklepy gospodarcze, itp.), wiercenia otworów, a dla mnie najważniejsze, dająca stabilną podstawę (mowa o doniczce typu ścięty stożek). Oprócz silnika w doniczce musimy umieścić diodę podczerwoną wraz z rezystorem ograniczającym prąd, ale w takim miejscu, aby podczas obrotu części ruchomej „spotykała się" z fotodiodą. Diodę możemy zabudować w zatyczkę od długopisu i przykleić do doniczki. Teraz musimy jakoś umocować część ruchomą na osi wirnika.

Ja użyłem do tego specjalnie zaprojektowanej tulejki, która ma z jednej strony otwór pasowany na wcisk osi wirnika, natomiast z drugiej strony trzpień gwintowany. Mając już wszystkie te elementy, zaczynamy montować je w jedną całość. Należy pamiętać, aby przykręcić linijkę z diodami do płytki głównej, ponieważ siła odśrodkowa może sprawić nam niespodziankę w postaci jej „odfrunięcia". Ostatnim etapem tej części będzie wyważenie elementu wirującego, proponuję zamocować po stronie baterii śrubę M8, którą będziemy mogli wkręcać lub wykręcać. Po złożeniu w całość i wyważeniu mamy niekonwencjonalną zabawkę.

Obsługa układu

Po włączeniu zasilania części wirującej diody powinny zapalić się na krótki moment oraz dodatkowo powinien pojawić się sygnał dźwiękowy.

Aby ustawić czas 12:23:25, wciskamy klawisz o kodzie 15 (gdy wyświetlany jest zegar), a następnie 1;2;2;3;2;5 *uwaga*: po każdym naciśnięciu klawisza należy odczekać ok. 0,5s!

Aby zaprogramować tekst „ABC", wciskamy klawisz o kodzie 15 (gdy wyświetlany jest dowolny napis), a następnie 0;1;0;2;0;3 (A- 01; B- 02; C-03), *uwaga*: kod 3;3 (33) kończy wprowadzanie tekstu!

Na pewno niektórzy patrząc na schemat urządzenia, zastanawiali się, po co użyłem diody prostowniczej na ścieżce „+" zasilania mikrokontrolera. Otóż okazało się, że wewnętrzny oscylator układu pracuje przy napięciu zasilania poniżej 4,5V (w moim przypadku) i zadaniem tej diody jest obniżenie go do tej wartości.

Na koniec podam Wam ciekawostkę, którą możecie wykorzystać przy ulepszeniu własnego wyświetlacza widmowego, mianowicie zmiana kierunku obrotów silnika na zgodny z ruchem wskazówek zegara, powoduje efekt lustrzany. Po zastosowaniu modułów laserowych napis wyświetlany może być np. na ścianie.

Łukasz Hrapek

Wykaz elementów

Rezystory

R1-R7,R10	320Ω
R8	220Ω
R9	2MΩ

Kondensatory

C1,C2	22pF
C3	100µF

Półprzewodniki

D1	1N4004
D2	dioda podczerwona
D3-D9	diody LED 5mm
D10	fotodioda
IC1	AT90S8535
IC2	7805
IC3	DS1813
OP1	SFH506

Różne

Q1	8MHz
Q2	32768Hz (zegarkowy)
S1	switch dwupozycyjny
Zasilanie1-2	łącznik
X1	wtyk DB9
X2	gniazdo kątowe DB9
SG1	Buzzer
G1	Bateria 9V
Podstawka pod IC1	

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2651