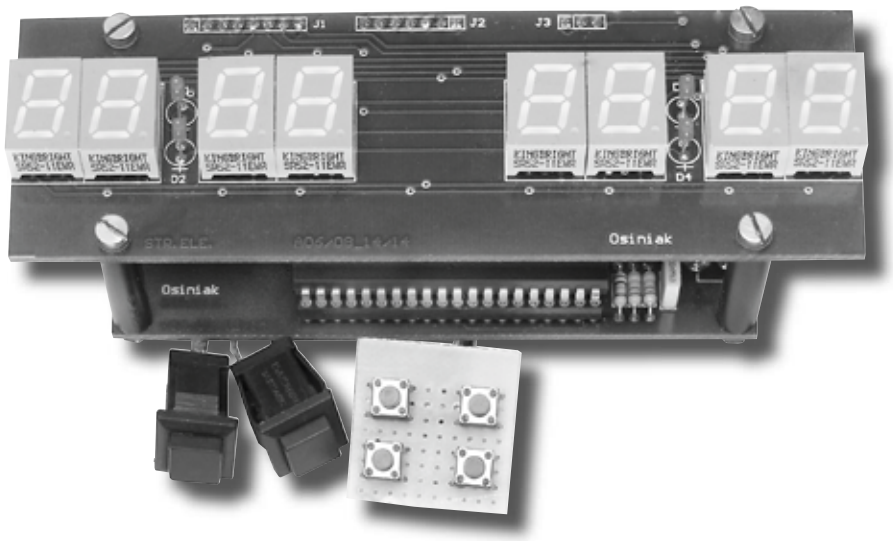
Zegar szachowy

AVT-378



Na łamach Elektroniki Praktycznej wielokrotnie już mogliśmy spotkać opisy przeróżnych zegarów. Były wszelkiego rodzaju timery, budziki, stopery, zegary cyfrowe oraz analogowe, jednym słowem wszystko co tylko można sobie wymarzyć. W niniejszym artykule pojawi się opis kolejnego licznika czasu – zegara szachowego.

Rekomendacje:

projekt polecany jest tym miłośnikom szachów, którzy chcieliby własnoręcznie wykonać zegar aby uatrakcyjnić rozgrywki i dodać do nich szczyptę emocji.

Dodatkowe materiały do artykułu publikujemy na CD-EP oraz www.ep.com.pl

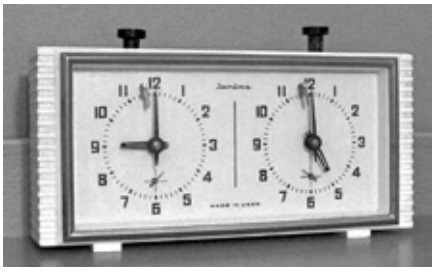
Kolejnym układem przeznaczonym do odliczania czasu publikowanym w Elektronice Praktycznej jest zegar szachowy. W niniejszym artykule zostanie opisany układ posiadający wszystkie funkcje tradycyjnych zegarów tego typu, który został rozszerzony o dodatkowe opcje, jak np.: możliwość gry na tzw. „przewagi czasowe", czy też zliczanie całkowitego czasu gry niezależnie dla obu graczy.

Na rynku zegarów szachowych wersje elektroniczne ustępują miejsca swoim „wskazówkowym" odpowiednikom (jedno z takich właśnie rozwiązań można zobaczyć na **fot. 1**). Jednak w domowym warsztacie elektronika hobbysty dużo łatwiej jest wykonać zegar cyfrowy niż wskazówkowy.

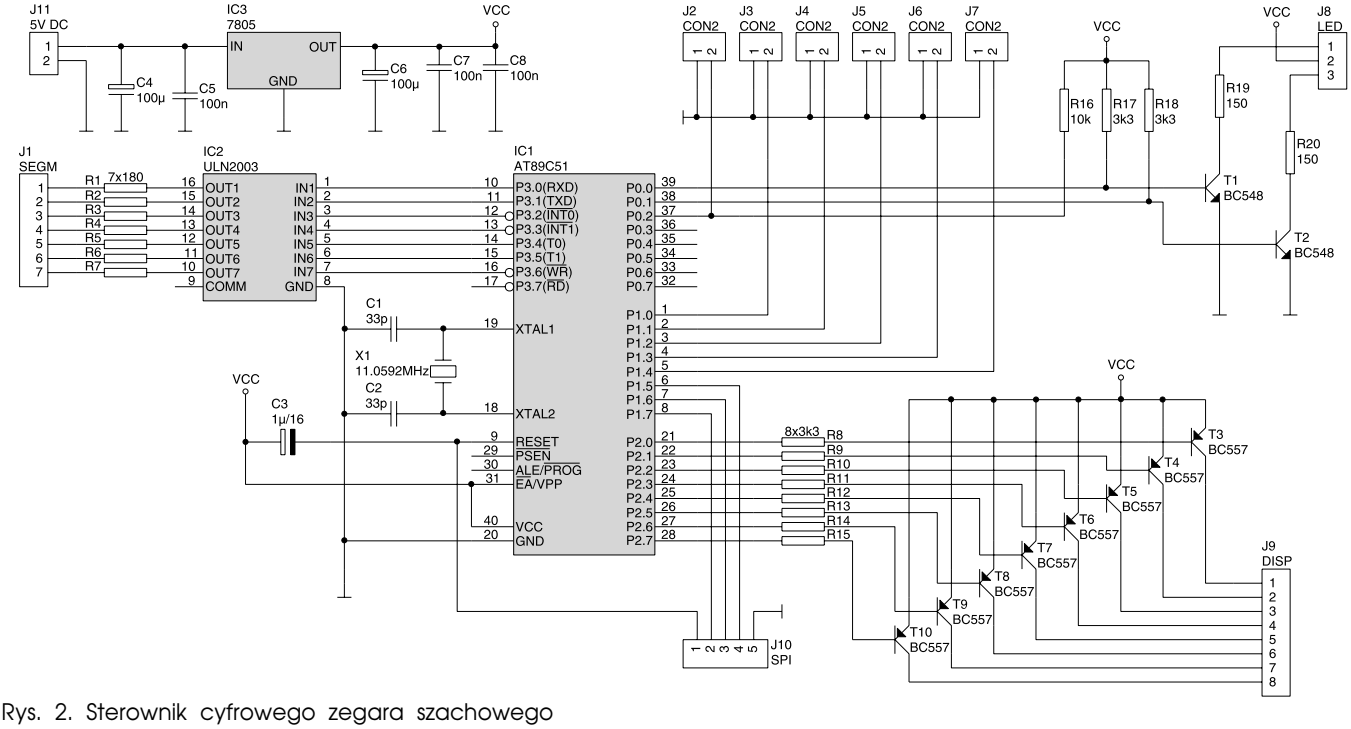
Typowy zegar składa się z dwóch identycznych mechanizmów zamkniętych razem we wspólnej obudowie, każdy z nich odmierza czas indywidualnie dla każdego z graczy. Przed rozpoczęciem rozgrywki zostaje ustalony limit czasu, jeżeli gracz nie zdąży z wykonaniem ruchu w ustalonym czasie - automatycznie przegrywa grę. Zegar odlicza w dół od zadanej wartości i jednocześnie wyświetla czas pozostały do końca tury. Po wykonaniu ruchu gracz wciska przycisk umieszczony nad „swoim zegarem", który powoduje zatrzymanie i wyzerowanie własnego licznika oraz jednoczesny początek odliczania czasu dla przeciwnika.

Prezentowany w artykule układ zegara można podzielić na dwa zasadnicze bloki: sterownik, którego schemat znajduje się na **rys. 2** oraz moduł wyświetlaczy (schemat na **rys. 3**). Sterowanie wyświetlaczem odbywa się z wykorzystaniem multipleksowania, co pozwoliło do minimum ograniczyć liczbę niezbędnych połączeń między płytkami. Jako dwukropki zostały wykorzystane cztery diody świecące o średnicy soczewki 1,8 mm połączone parami równolegle (D1+D2 oraz D3+D4), które sterowane są za pomocą 3-stykowego złącza J3.

Moduł sterownika z pozoru może wydawać się skomplikowany, jednak rzeczywistość jest nieco prostsza. Przede wszystkim można go podzielić na kilka mniejszych grup: zasilacz (kondensatory C5...C8 oraz stabilizator IC3), bufor segmentów (oporniki R1...R7 oraz układ IC2), bufor „wspólnych anod" (rezystory R8...R15 wraz z tranzystorami T3... T10), bufor dwukropków (elementy R17...R20 oraz T1 i T2) oraz na mikrokontroler wraz z niezbędnymi



Fot. 1. Tradycyjny zegar szachowy



Rys. 2. Sterownik cyfrowego zegara szachowego

do jego pracy elementami (IC1, X1, C1...C3). Postaram się teraz pokrótce omówić każdy z wymienionych bloków w kolejności, w jakiej były wymieniane.

Zasilacz całego układu został zbudowany w oparciu o bardzo popularny układ 7805. Zastosowanie scalonego stabilizatora pozwoliło rozszerzyć granice, między którymi powinna znajdować się wartość napięcia zasilającego. Kondensatory C4...C8 wygładzają to napięcie zapewniając poprawną pracę mikrokontrolera. Kolejnym blokiem jest bufor segmentów wyświetlaczy, składający się z rezystorów ograniczających prąd pojedynczej diody oraz układu ULN2003. Zastosowanie oporników o wartości 180 Ω sprawiło, że wyświetlane cyfry są nieco ciemniejsze (efekt multipleksowania), jednak ryzyko uszkodzenia wyświetlacza podczas testowania programu jest minimalne, gdyż przez diody nie popłynie prąd większy niż 28 mA.

Następnym opisywanym przeze mnie elementem sterownika jest bufor wspólnych anod wyświetlaczy. Część ta składa się z ośmiu tranzystorów PNP oraz rezystorów o wartości 3,3 kΩ, których zadaniem jest ograniczenie prądu bazy tranzystorów T3...T10. Podanie stanu niskiego na bazę tranzystora za pośrednictwem odpowiedniego opornika powoduje podłączenie wybranego wyświetlacza do plusa zasilania.

Kolejny układ, będący ostatnią już częścią składową sterownika, ma za zadanie zasilić dwukropki. Został on zrealizowany w typowy sposób na dwóch tranzystorach: T1, T2 wraz z rezystorami R17...R20. Dwa z tych oporników (R19 i R20) ograniczają prąd diod, natomiast pozostałe dwa są niezbędne do wymuszenia stanu wysokiego na bazach tranzystorów.

Program sterujący

Program sterujący pracą zegara został napisany w języku C i skompilowany przy użyciu darmowego kompilatora SDCC (*http://sdcc.sourceforge.net*). Opisywanie szczegółowo całego programu sterującego pracą mikrokontrolera w tej części artykułu byłoby dość kiepskim pomysłem, dlatego też skupię się jedynie na najważniejszych fragmentach kodu. Na początek najlepiej zerknąć na schemat blokowy algorytmu, który został przedstawiony na **rys. 4**. W pierwszym etapie ustawiane są odpowiednie parametry układu przerwań oraz timerów takie, jak: włączenie obsługi przerwań, ustawnie licznika 0 w tryb 2, a licznika 1 w tryb 0, itp. Przygotowanie portów procesora sprowadza się głównie do wygaszenia wyświetlaczy wraz z dwukropkami oraz podania logicznej jedynki na wyprowadzenia klawiatury. Na tym etapie programu bardzo pomocne okazały się dyrektywy „define", dzięki którym kod stał się bardziej przejrzysty.

Za odświeżanie zawartości wyświetlacza odpowiedzialna jest funkcja *Multiplexing* (**list. 1**), która została „podczepiona" pod przerwanie pochodzące od licznika T1. Bezpośrednio po przepełnieniu się licznika i wygenerowaniu przerwania, do rejestru TH1 zapisywana jest wartość B8h, co powoduje ustawienie częstotliwości odświeżania na około 50 Hz. W celu jej zwiększenia należy zmienić tę wartość na odpowiednio większą. Każde przepełnienie timera generuje przerwanie, a procesor wykonuje funkcję *Multiplexing* za każdym razem wyświetlając kolejną cyfrę, od lewej do prawej strony. Numer aktualnie wyświetlanej cyfry przechowywany jest w zmiennej *active_display*. Przed wyświetleniem nowego znaku, wszystkie wyświetlacze są wygaszane, a następnie po ustawieniu danej na „porcie segmentów" zostaje podane zasilanie na anodę odpowiedniej pozycji. Taki sposób sterowania pozwala uniknąć sytuacji, w której na jednej pozycji wyświetlane są dwie cyfry, z czego jedna znacznie słabiej od drugiej.

Kolejną ważną, z punktu widzenia działania programu, funkcją jest *Clock* (kod na **list. 2**), której zadaniem jest zliczanie sekund od chwili uruchomienia timera nr 0. Mikrokontroler zaczyna wykonywać instrukcje zawarte w ciele funkcji w momencie otrzymania przerwania od licznika T0. Timer ten został ustawiony w tryb 2, co oznacza, że pracuje jako licznik 8-bitowy z automatycznym przeła-

List. 1.
```
void Multiplexing () interrupt TF1_VECTOR
// multipleksowanie „podczepione” pod prze-
rwanie T0
{
  TH1 = 0xE2;
                  // zaladuj nowa wartosc do
licznika
  segment_port = 0;
            // wylacz wyswietlana cyfre
  if ((powered_display & 3) && (powered_di-
splay & 4))      // jezeli wyswietlacz ma
cokolwiek wyswietlic...
  {
    switch (powered_display & 3)
              // dokonaj identyfikacji tylko
pierwszych 3 bitow
    {
      case 1:
        if (active_display > 3) active_display
= 0;      //tylko lewa czesc wyswietlacza
      break;
      case 2:
        if (active_display > 7) active_display
= 4;         //tylko prawa czesc wyswietlacza
      break;
      case 3:
        if (active_display > 7) active_di-
splay = 0;        //odswiezaj wszystkie osiem
pozycji
      break;
    }
    display_port = display[active_display];
// ustaw zasilanie segmentow dla nowej cyfry
    segment_port = buffer[active_display];
// wlacz zasilnie kolejnego wyswietlacza
    ++active_display;
// zwieksz numer wyswietlacza
  }
}
```

List. 2.
```
void Clock () interrupt TF0_VECTOR
// sekundnik „podczepiony” pod przerwanie T1
{
  ++clock_temp;                    //
zwieksz stan licznika przerwan timera
  switch (clock_temp)
  {
    case 1:
      if (powered_colon & 1) colon_left = 1;
// wlacz lewy dwukropek
      if (powered_colon & 2) colon_right = 1;
// wlacz prawy dwukropek
      if (powered_display & 8) powered_di-
splay |= 4;     // wlacz wyswietlacze jezeli
wlaczono miganie
    break;
    case 1800:
      if (powered_colon & 1) colon_left = 0;
      if (powered_colon & 2) colon_right = 0;
      if (powered_display & 8) powered_di-
splay &= 251;      // wylacz wyswietlacze
jezeli wlaczono miganie
    break;
    case 3600:
            // wyzeruj licznik przerwan gdy
osiagnie 3600
      clock_temp = 0;
      ++seconds_counted;
// zwieksz o jeden licznik odliczonych sekund
    break;
  }
}
```

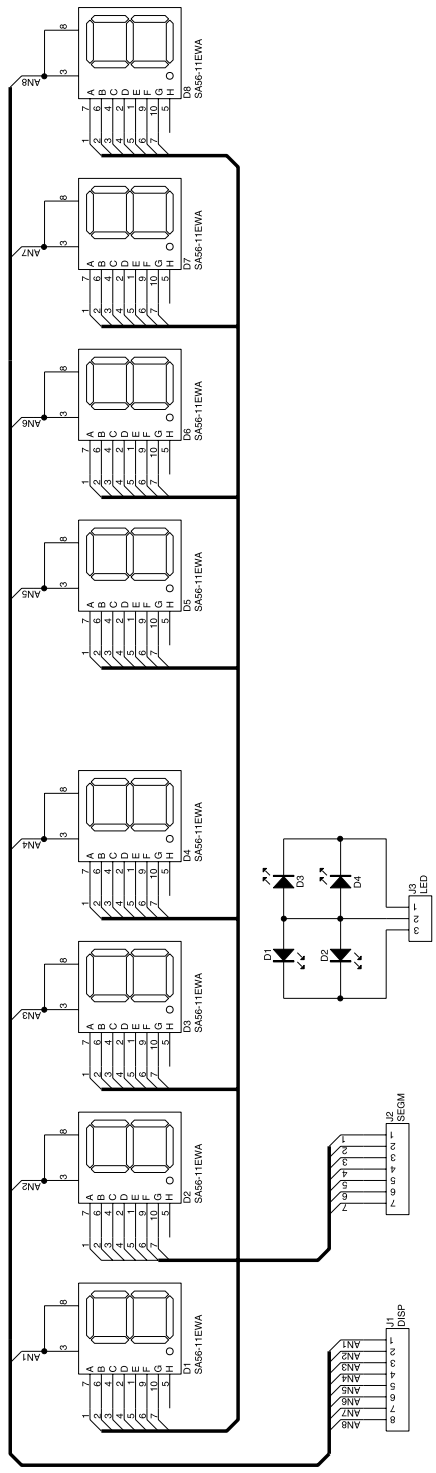
dowaniem. Przy częstotliwości rezonatora 11,0592 MHz timer generuje przerwanie dokładnie 3600 razy na sekundę, dlatego też niezbędne było wprowadzenie pomocniczej zmiennej *clock_temp*, która pełni rolę licznika przerwań modulo 3600, a w momencie przepełnienia zwiększa o 1 licznik sekund. W funkcji dodatkowo zaimplementowano miganie dwukropków oraz wyświetlacza. O ile dwukropek powinien migać z częstotliwością 1 Hz, to nic nie stoi na przeszkodzie by np.: „koniec czasu” był sygnalizowany miganiem wyświetlaczy z większą częstotliwością. W takim przypadku należy jedynie dopisać potrzebne nam warunki do instrukcji *switch* przy odpowiednich wartościach zmiennej *clock_temp* (np.: 900 i 2700 dla 2 Hz).

Warto także poświęcić nieco uwagi obsłudze klawiatury, podczas ustawiania początkowych parametrów zegara, realizowanej przez funkcję *Settings*, której fragmenty zamieszczono na **list. 3**. Można rzec, że całe jej wnętrze to jedna wielka pętla, która wykonuje się tak długo, aż nie zostanie naciśnięty przycisk *mode_button* służący do wejścia/wyjścia z trybu ustawień. Do tego czasu procesor cały czas sprawdza stan pozostałych klawiszy, czy przypadkiem któryś nie został wciśnięty. Jeżeli np.: został naciśnięty klawisz *mode_button* procesor zwiększy stan odpowiedniej zmiennej, pilnując jednocześnie by nie osiągnęła ona niedozwolonej wartości. Jeżeli natomiast wciśnięto przycisk *mode_button* program przejdzie w drugą pętlę, tym razem jednak oczekując na zwolnienie styków. Gdyby zrezygnować z oczekiwania na zwolnienie klawisza, procesor mógłby się bardzo łatwo zapętlić, uznając wciśnięcie klawisza jednocześnie jako znak do opuszczenia i ponownego wejścia w tryb ustawień. Wywołania *WaitMs (250)* odpowiadają za tzw. powtarzanie klawisza oraz zapobiegają zbyt szybkiemu zwiększaniu się stanów liczników w sytuacji gdy dany klawisz został wciśnięty i przytrzymany.

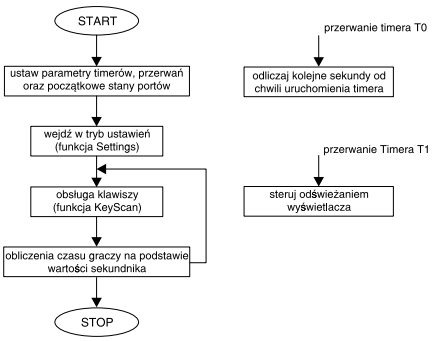
Funkcja *KeyScan* wywoływana jest bezpośrednio z głównej pętli programu, a jej zadaniem jest między innymi obsługa klawiszy końca tury podczas normalnej pracy zegara. Dodatkowo wspomniana przed chwilą główna pętla programu zawiera szereg instrukcji przeznaczonych do obliczania aktualnej wartości czasu. Tak obliczony czas, inaczej dla każdego trybu pracy, jest następnie przekształcany z liczby sekund na postać *h:m:s* i umieszczany w buforze wyświetlacza, z którego już zostanie bezpośrednio wyświetlony przy najbliższym przerwaniu licznika T1.

Obsługa zegara

Bezpośrednio po podłączeniu napięcia zasilającego zaszyty w pamięci procesora program rozpocznie swoje działanie, czego dowodem powinno być 6 cyfr pokazanych na wyświetlaczu. Pierwsza, patrząc od lewej strony, cyfra informuje o sposobie odmierzania czasu wskazując jednocześnie na tryb pracy zegara. Następna cyfra informuje nas o tym, który gracz rozpocznie partię; 1 jest przypisana do lewej części wyświetlacza, zaś 2 do prawej. Pozostałe 4 cyfry reprezentują wartość czasu niezbędną podczas pracy w trybie 1 i 2. Zegar może odmierzać czas gry pracując w jednym z 3 dostępnych trybów. W pierwszym z nich czas jest odliczany w dół od zadanej wartości, drugi tryb to gra na tzw.



Rys. 3. Schemat elektryczny wyświetlacza cyfrowego zegara szachowego



Rys. 4. Algorytm działania programu zegara cyfrowego

przewagi czasowe, w której zegar pilnuje by różnica całkowitych czasów gry każdego z graczy nie przekroczyła nastawionej wartości. W trybie 3 zegar pełni rolę licznika czasu, odmierzając całkowity czas ruchów dla każdego gracza. Przy takiej pracy nastawiona wartość czasu nie jest brana pod uwagę. Do wyboru sposobu odmierzania czasu służy *przycisk mode_button*, którego każdorazowe naciśnięcie powoduje przejście na następny w kolejności tryb pracy. Jeżeli chcemy aby grę rozpoczynał zawodnik, którego wyświetlacz znajduje się z prawej strony wystarczy, że będąc w ustawieniach, naciśniemy jego przycisk końca tury. Jeżeli wszystko się udało, na wyświetlaczu powinien się zmienić numer gracza z 1 na 2. Można wielokrotnie zmieniać numer rozpoczynającego zawodnika przez wciśnięcie odpowiedniego dla danego gracza przycisku końca tury. Następnie za pomocą klawiszy *minutes_button* oraz *seconds_button* nastawiamy odpowiednio dobrany czas (nie jest to wymagane przy pracy w trybie 3). Kiedy już wszystko zostało ustawione i gracze są gotowi do rozpoczęcia partii, wystarczy nacisnąć przycisk *settings_button*, a zegar automatycznie rozpocznie odmierzanie czasu.

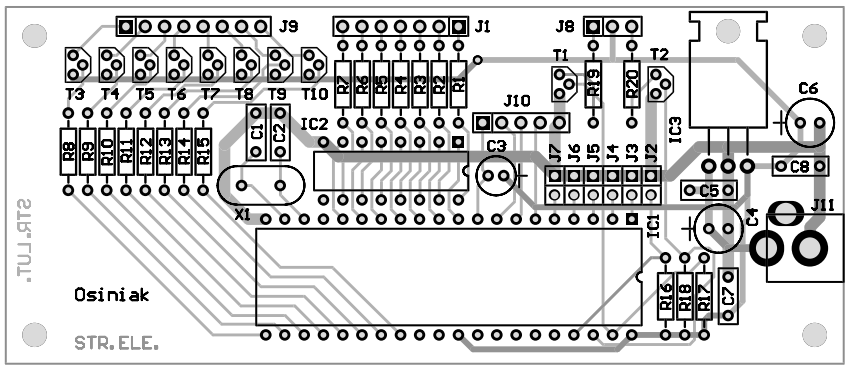
Montaż i uruchomienie

Układ zegara zmontowano na dwóch osobnych płytkach obwodu drukowanego (sterownik - **rys. 5** oraz wyświetlacz - **rys. 6**). Montaż elementów powinien rozpocząć się od elementów najmniejszych, a zakończyć na zainstalowaniu procesora w podstawce oraz połączeniu płytek ze sobą. Połączenie takie najwygodniej jest wykonać srebrzanką, jednak nic nie stoi na przeszkodzie by wykorzystać inny materiał, jak np. popularny miedziany drut „telefoniczny". Jeżeli wszystko zostało poprawnie zmontowane, to po podaniu napięcia zasilania można przystąpić do zaprogramowania procesora o ile nie zostało to wykonane wcześniej. Tak zmontowany w całość układ zegara jest gotowy do pracy i powinien „startować" zaraz po podaniu napięcia zasilania na złącze J11.

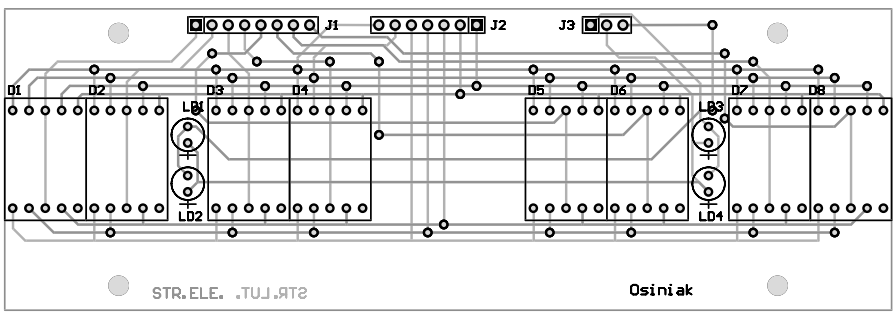
Marcin Osiniak

List. 3.

```
void Settings ()
{

 [...]

  while (settings_button)
// wykonuj petle dopoki nie wcisnieto ponownie guzika
  {
    if (!mode_button)
// obsluz przycisk zmiany trybu pracy zegara
    {
      if (++clock_mode == MODES_NUMBER)
clock_mode = 0;    // zwieksz i jesli „przepelniono", to wyzeruj
      buffer[0] = digits[clock_mode + 1];
// pokaz nowa wartosc na wyswietlaczu
      WaitMs (250);
    }
    if (button_left)
                // jezeli wcisnieto lewy przycisk konca tury
    {
      player_turn = 0;
                // zmien rozpoczynajacego gracza na pierwszego
      buffer[2] = 0x06;
                // wyswietl odswiezona wartosc numeru gracza
    }
    if (button_right)
// jezeli wcisnieto prawy przycisk konca tury
    {
      player_turn = 1;
                // zmien rozpoczynajacego gracza na drugiego
      buffer[2] = 0x5B;
                // wyswietl odswiezona wartosc numeru gracza
    }
    if (!minutes_button)
    {
      if (++initial_time.m == 60) initial_time.m = 0;      // zwieksz licznik minut i wyzeruj jezeli osignie 60
      LoadBuffer (4, initial_time.m);
// pokaz nowa wartosc minut na wyswietlaczu
      WaitMs (250);
                // oporznienie powtarzania klawisza
    }
    if (!seconds_button)
    {
      if (++initial_time.s == 60) initial_time.s = 0;      // zwieksz licznzik sekund i wyzeruj jezeli osignie 60
      LoadBuffer (6, initial_time.s);
      WaitMs (250);
    }
  }
  WaitMs (10);
                // poczekaj z powodu stykow wlacznika
  while (!settings_button);
// czekaj az przycisk zostanie zwolniony

[...]

}
```



Rys. 5. Schemat montażowy płytki drukowanej sterownika zegara szachowego



Rys. 6. Schemat montażowy płytki drukowanej wyświetlacza cyfrowego zegara szachowego

WYKAZ ELEMENTÓW

Rezystory
R1...R7: 180 Ω
R8...R15, R17, R18: 3,3 kΩ
R16: 10 kΩ
R19, R20: 150 Ω

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

Półprzewodniki
IC1: AT89C51
IC2: ULN2003
IC3: 7805
T1, T2: BC548
T3...T10: BC557
X1: 11,0592 MHz
D1...D8: SA56-11EWA
LD1...LD4: dioda LED 1,8 mm

Inne
J2...J7: goldpin 1x2
J10: goldpin 1x5
J11: gniazdo zasilania