Zegar „wskazówkowy”

2973 AVT

Na łamach EdW opisywanych była już może nie niezliczona, ale z pewnością duża liczba rozmaitych zegarów. Jak dotąd żaden z nich nie mógł się poszczycić prawdziwymi wskazówkami ;-) Opisany projekt powstał w większości już 3 lata temu, ale z powodu trudności w uzyskaniu pożądanego efektu końcowego, poszedł na bardzo długi czas w odstawkę. Minęło sporo czasu, technika poszła nieco do przodu, podobnie jak doświadczenie i umiejętności autora. Inspiracją do dokończenia projektu, jak to często bywa, była lektura jednego z odcinków Szkoły Konstruktorów, a konkretnie zadania 174 związanego z wykorzystaniem diod LED RGB. Tak więc odkurzony i odświeżony powraca jako zegar z nie jedną, ale aż z czterema tarczami i czterema wskazówkami. Bo jak już zapewne domyślasz się, drogi Czytelniku, z fotografii tytułowej, medium, za pośrednictwem którego przekazywana jest informacja o czasie, są cztery wskazówkowe mierniki elektryczne.

Opis układu

Urządzenie można podzielić na dwa bloki: blok główny (**rysunek 1**), stworzony w pierwszej fazie rozwoju projektu, oraz blok pomocniczy (**rysunek 2**), zaprojektowany po wznowieniu prac, który zawiera wszystkie elementy, których zabrakło w bloku głównym, wzbogacony o nowe pomysły.

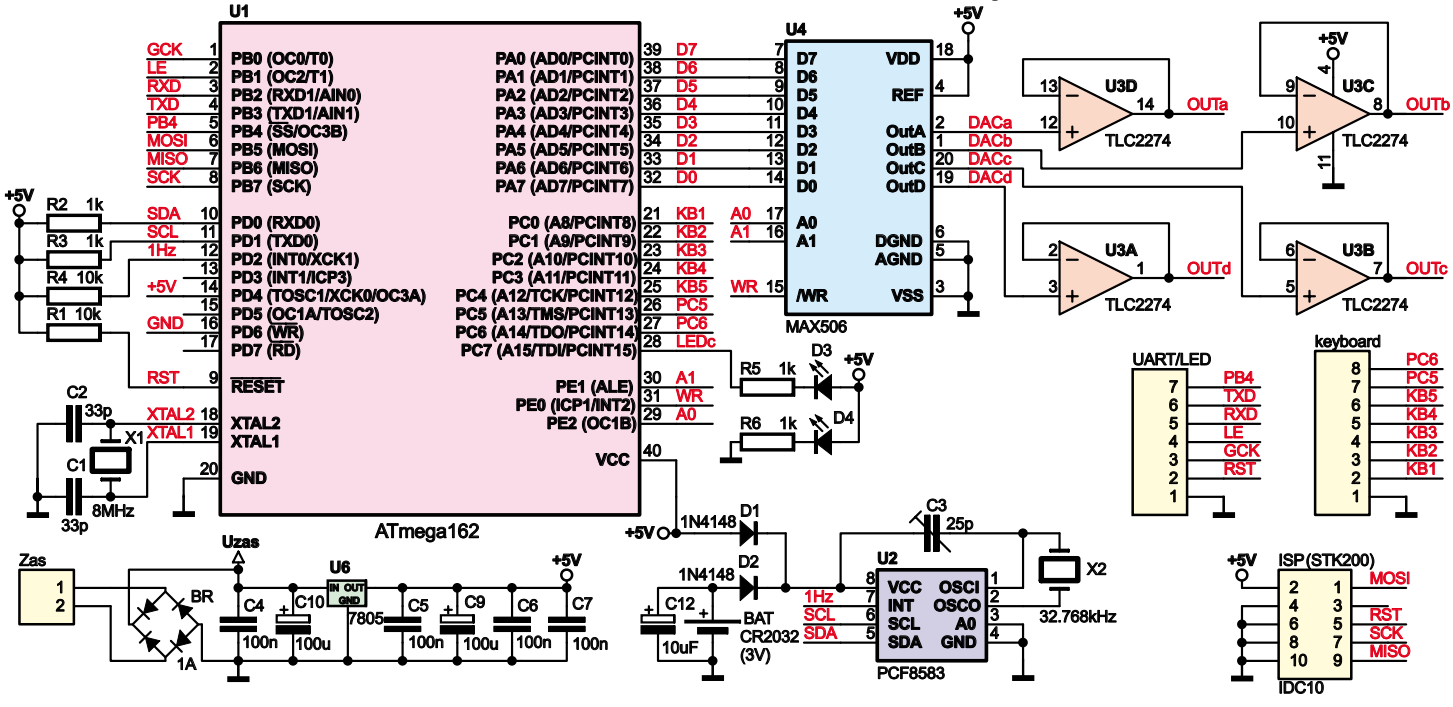
Sercem bloku głównego jest mikrokontroler AVR typu ATmega162 (U1 – 16KB Flash + 1KB RAM) pracujący z zewnętrznym kwarcem 8MHz, w obudowie DIP40.

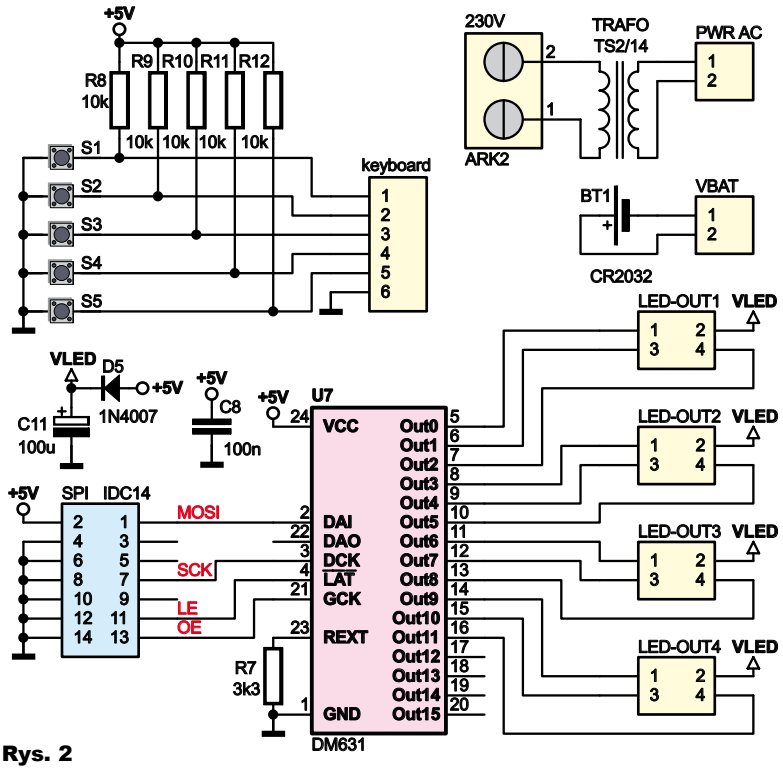
Układ ten ma inny układ wyprowadzeń niż większość AVR-ów. Nie ma to zupełnie znaczenia w urządzeniu, ale tylko przy chęci zmiany mikrokontrolera (w pewnym stopniu kompatybilny będzie jedynie układ ATmega8515).

Jak na porządny zegar przystało, nie mogło zabraknąć układu RTC. W tym przypadku rolę tę pełni układ U2 typu PCF8583 w klasycznej aplikacji z podtrzymywaniem bateryjnym zrealizowanym na dwóch diodach – D1 i D2. Układ ten łączy się z procesorem za pośrednictwem magistrali I^{2}C, a dodatkowo wysyła sygnał o częstotliwości 1Hz. Ze względu na brak w mikrokontrolerze sprzętowego układu TWI (*two wire interface* – atmelowski odpowiednik I^{2}C), jest on emulowany programowo. Rezystory R2 i R3 są konieczne do poprawnego działania magistrali. Rezystor R4 został dodany ze względu na to, że wyjście INT układu U2 ma budowę typu otwarty dren i wobec tego do generowania poprawnego sygnału wymaga podciągnięcia. Teoretycznie wystarczyłby pull-up wbudowany w strukturę procesora, ale nie zaszkodzi się zabezpieczyć.

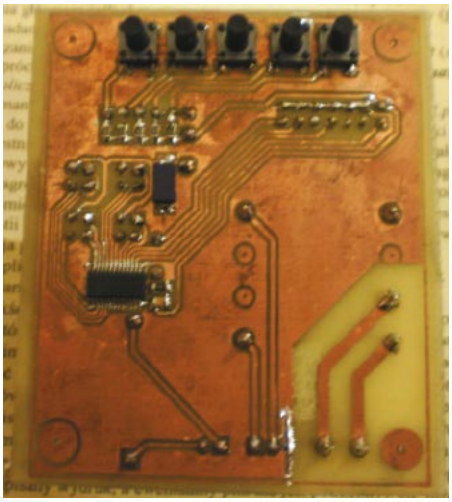
Aby zaprezentować czas na miernikach wychyłowych, potrzebny był przetwornik co najmniej 3 kanałowy (sekundy, minuty, godziny). Wybór padł na czterokanałowy 8-

Rys. 1





Rys. 2



Fot. 1

-bitowy przetwornik MAX506 (U4) firmy Maxim Dallas. Jest on bardzo prosty w obsłudze. Dzięki równoległej magistrali danych, wystarczy w jawnej formie podać wartość bitową napięcia na jego wejścia, wybrać adres jako jedną z czterech kombinacji stanów na dwóch wejściach adresowych i zatrzasnąć dane sygnałem na linii /WR. Wadą tego układu jest wysoka cena. O alternatywach można przeczytać w podrozdziale „Możliwości zmian".

Wyjścia DAC-a zostały zbuforowane w klasyczny sposób za pomocą czterech wtórników na wzmacniaczach U3A-U3D. Zabezpiecza to układ przetwornika przed przeciążeniami i innymi niespodziankami, jakie mogą pojawić się szczególnie podczas prób. Układ U3 typu TLC2274 jest wzmacniaczem operacyjnym Rail-to-Rail, co oznacza, że może pracować w pełnym zakresie napięcia zasilania. Dzięki temu, możliwe było wykorzystanie napięcia zasilania +5V w roli napięcia referencyjnego.

Kilka słów komentarza na temat samych mierników. Jak widać, są to cztery identyczne wskaźniki, pokazujące sekundy, minuty, godziny i dni tygodnia. Pierwotnie były to amperomierze prądu stałego o zakresie... 6A! Spośród mnóstwa identycznych z wyglądu amperomierzy i woltomierzy wskazówkowych, wiele ma identyczne „serce" w postaci mikroamperomierza i są jedynie odpowiednio przeskalowywane. Użyte przeze mnie mierniki są łatwo dostępne i tanie (10-15 PLN/szt.), jednak trudno mi podać konkretny model, bo pudełka zaginęły dawno temu.

Na potrzeby zegara, oryginalne rezystory, ustalające zakres miernika, zostały usunięte i zastąpione 20-obrotowym potencjometrem montażowym 10kΩ (przyklejonym do obudowy miernika), którym precyzyjnie można dobrać odpowiednią czułość. Na wszelki wypadek w tylnej ściance miernika zostały wywiercone niewielkie otwory, umożliwiające kalibrację także po zamknięciu obudowy miernika. Pozostała część układu głównego zawiera między innymi diody kontrolne, złącza do programowania i podłączenia płytki pomocniczej oraz klasyczny układ pracy stabilizatora 7805 wraz z mostkiem prostowniczym.

Płytka pomocnicza

Jak wspomniałem na wstępie, płytka pomocnicza powstała znacznie później. Znalazł się na niej między innymi transformator zasilający typu TS 2/14 oraz gniazdo baterii CR2032, podtrzymującej działanie układu RTC. Ponadto znajduje się tu 5 przycisków sterujących wraz z rezystorami podciągającymi. Przyciski te wyjątkowo zostały zamontowane od spodu płytki, jak pokazuje **fotografia 1**, dzięki czemu można je było wyprowadzić na tylnej ściance obudowy zegara – **fotografia 2**. Jak widać, większość niemechanicznych komponentów to elementy SMD.

Dochodzimy teraz do przyczyny wznowienia projektu, czyli do podświetlenia. Stara, pierwotna koncepcja zakładała użycie dwukolorowych dwukońcówkowych diod RG 5mm i programowego PWM-a do ich sterowania. Niestety, ówczesne diody tego typu nie oferowały zbyt dużej jasności, a dodatkowo wystąpiły trudności w wykonaniu półprzezroczystej skali. Jednakże przez 3 lata technika LED rozwinęła się i powszechnie dostępne są diody RGB o bardzo przyzwoitych parametrach. Koncepcja uległa zmianie, więc i podejście do niej musiało się zmienić. Zrezygnowałem z programowego PWM-a na rzecz układu U7 typu DM631. Jest to specjalizowany układ, mający 16 kanałów ze źródłami prądowymi, specjalnie do sterowania diod LED. Ponadto każdym kanałem można osobno sterować za pomocą wbudowanego 12-bitowego PWM-a. Wadą układu jest słaba dostępność, ale chętnym do powielenia konstrukcji polecam bliski odpowiednik, tj. MBI5031, który jest nóżkowo kompatybilny z DM631 i można go bez problemu kupić w jednym ze sklepów internetowych. Dla osób, które widzą problem w lutowaniu układu o tak małym rastrze (0,635mm) i precyzyjnym trawieniu płytki, dobrą wiadomością będzie fakt, że układ można dostać także w obudowach o większych rastrach, w tym klasycznym SOP24 (1,27mm). Prąd diod ustalany jest jednym wspólnym rezystorem R7. Przy wartości jak na schemacie, wynosi on około 18mA, co jest wartością aż nadto wystarczającą. Dodatkowa dioda prostownicza D5 wprowadza pewien

Fot. 2

Fot. 3

spadek napięcia, dzięki czemu układ nie musi „zbijać" całej różnicy między napięciem zasilania a napięciem przewodzenia diod, co prowadzi do mniejszych strat cieplnych.

Układ łączy się z procesorem po magistrali szeregowej zbliżonej do SPI, wzbogaconej o linię zatrzaskującą wprowadzone dane. Dodatkowo do układu MBI5031 należy doprowadzić sygnał zegarowy dla PWM-a – linia GCK. Wyjścia układu połączone są przewodowo bezpośrednio ze strukturami RGB diod podświetlających mierniki. Diody te zostały umieszczone na niewielkich płytkach przyklejonych do tylnej ścianki miernika. Przez dodatkowy otwór wyprowadzone zostały goldpiny do podłączenia przewodów sterujących. Schemat tego prostego układu widoczny jest na **rysunku 3**. Zastosowałem tu diody RGB w obudowach PLCC6, które umożliwiają dowolną konfigurację np. tak jak w tym przypadku ze wspólną anodą.

Plik	Opis zadań
zegar.main.c	Główny plik programu, przerwania od klawiatury i sygnału 1Hz, obsługa klawiatury i ustawiania czasu.
uart.c	Plik obsługi interfejsu UART, przydatny głównie podczas prób.
dac.c	Zawiera funkcje obsługi przetwornika CA.
rtc.c	Zawiera funkcje transmisji z i do układu RTC wraz z obliczeniami.
sw_spi.c	Plik programowej emulacji interfejsu SPI.
backlight.c	Zawiera obsługę drivera LED i funkcje efektów podświetlenia.
i2cmaster.s	Plik programowej emulacji interfejsu I²C.

Tab.1. Opis funkcji plików źródłowych

Program

Program dla mikrokontrolera został napisany prawie w całości w języku C (poza drobnymi wstawkami i biblioteką emulacji interfejsu I²C, napisanymi w asemblerze). Wszystko zostało skompilowane w środowisku AVR Studio z dołączonym WinAVR. Program po skompilowaniu zajmuje około 30% pamięci Flash (niecałe 5KB), co daje duże możliwości rozbudowy. Użycie pamięci RAM jest bardzo niewielkie, na poziomie kilku procent, więc i tu można poszaleć. Cały program składa się z kilku plików (można je ściągnąć z Elportalu), których krótki opis funkcjonalny znajduje się w **tabeli 1**. W programie każdy plik źródłowy ma odpowiadający mu plik nagłówkowy *.h, gdzie umieszczone są deklaracje funkcji, definicje stałych i kilka makr. Przy okazji polecam lekturę dodatkowego pliku nazwanego tao.h ;-)

Zanim przejdę do szczegółów związanych z poszczególnymi plikami źródłowymi, warto jeszcze wspomnieć o ciekawej konstrukcji umieszczonej w pliku zegar_main.h, którą zobaczyć można na **listingu 1**. Jest to element, który zapożyczyłem dawno temu z książki „Mikrokontrolery AVR w praktyce" autorstwa Jarosława Dolińskiego, a który stosowałem z powodzeniem nie tylko z mikrokontrolerami AVR. Cała rzecz dotyczy łatwego dostępu bitowego do rejestrów. W strukturze o nazwie *pole_bitowe* zdefiniowane są poszczególne bity bajtu. Następnie korzystając z makra *DAJ_BIT(adr),* możemy przypisać dowolną nazwę do jednego bitu z wybranego rejestru. Teraz do takiej nazwy wystarczy przypisać 1 lub 0, aby zmienić stan bitu w rejestrze! Do poprawnego działania, makro wymaga podania adresu danego rejestru, który najwygodniej jest sobie zdefiniować jako stałą, a odnaleźć go można w zestawieniu rejestrów, pod koniec noty katalogowej procesora. Taki sposób dostępu do poszczególnych bitów rejestru może nie jest najwydajniejszy, ale w niekrytycznych miejscach warto go zastosować, bo bardzo poprawia czytelność programu i jest po prostu wygodny.

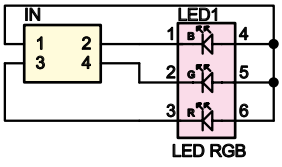
Przejdźmy teraz do opisu kilku fragmentów właściwego programu, rozpoczynając od pliku sw_spi.c Przedrostek sw sugeruje od razu realizację software'ową, czyli programową tegoż interfejsu. Być może zastanawiasz się drogi Czytelniku, dlaczego zdecydowałem się na ten krok skoro ogromna większość mikrokontrolerów AVR ma sprzętowe SPI?! Już tłumaczę. Ma to związek ze specyficzną obsługą układu DM631. Jak wspomniałem wcześniej, przyjmuje on po 12 bitów na każdy kanał. W procesorach AVR nie można zmienić długości słowa w interfejsie SPI, która jest na sztywno ustalona na 8 bitów. Tak więc chcąc wysłać dane do 12 kanałów (tyle jest wykorzystywane w urządzeniu), należałoby je najpierw odpowiednio przekształcić na postać 8-bitową. Ponieważ transmisja ta nie wymagała znacznej prędkości, łatwiejsze okazało się zasymulowanie działania interfejsu drogą programową. Jeżeli ktoś chciałby wykorzystać układ MBI5031, to wspomnę tutaj, że choć także ma on 12-bitowy PWM, to przyjmuje po 16 bitów na kanał, co stwarza możliwość prostszego wykorzystania interfejsu sprzętowego. Do programu dołączam także plik spi.c, który obsługuje interfejs sprzętowy i posiada funkcje o identycznym dla reszty programu działaniu, więc w takim wypadku wystarczy podmienić plik sw_spi.c na spi.c.

Pliki obsługi interfejsu I²C zostały znalezione w Internecie i okazały się bardzo przyjemne w użyciu. Cały pakiet składa się z pliku nagłówkowego i2cmaster.h oraz dwóch plików źródłowych: i2cmaster.c i i2cmaster.s. Dołączając pierwszy z nich, uzyskujemy obsługę interfejsu sprzętowego, natomiast drugi umożliwia emulację programową. Wystarczy jedynie zmienić w nim definicje pinów SDA i SCL i gotowe.

Z interfejsem I²C wiąże się ściśle także plik rtc.c, gdzie na początku zdefiniowane są dwie funkcje zbierające razem wszystkie potrzebne działania na magistrali – *i2c_wyslij* i *i2c_odbierz*. Dalsza część pliku zawiera funkcje konwertujące wartość w kodzie BCD na wartość dziesiętną i odwrotnie, oraz funkcje wysyłania i odbioru czasu z układu RTC. Funkcję zapisu można zobaczyć na **listingu 2** (druga funkcja jest analogiczna). Wyjaśnienia wymagają zawiłe przeliczenia czasu. Otóż w celu wykorzystania możliwie jak największej liczby poziomów przetwornika dla płynnego ruchu wskazówek (poza sekundami, których skokowa zmiana wydaje się bardziej naturalna) konieczne jest przechowywanie aktualnego czasu w sposób możliwie zbliżony do rozdzielczości przetwornika. I tak dla sekund czas jest przechowywany jako liczba dziesiętna od 0 do 59, ale już dla minut i godzin jako liczba od 0 do 239, a dla dni tygodnia od 0 do 209.

Wartości te są przechowywane w strukturze o nazwie *czas*. Zawiera ona ponadto 3 zmienne, przechowujące wartości pośrednie czasu. Dochodzimy tutaj do najistotniejszego fragmentu programu, tj. obsługi przerwania INT0, które jest wyzwalane sygnałem 1Hz z układu RTC (**listing 3**). W przerwaniu tym na bieżąco zliczane są kolejne sekundy oraz sekundy pośrednie. Te ostatnie służą do odpowiedniej inkrementacji minut. Po upływie 15 sekund inkrementowana jest wartość „minut" i zerowane są sekundy pośrednie. Teraz już łatwo się domyślić, dlaczego „minuty" przyjmują wartości od 0 do 239 –> 60s/15s = 4, 4*60min = 240. Wraz z inkrementacją „minut" zwiększane są także minuty pośrednie. Po upływie 12 „minut pośrednich" (czyli 3 prawdziwych minut -> 12/4=3) następuje inkrementacja „godzin pośrednich". Ta zmienna jest tak naprawdę miarą upływu nie godzin, tylko okresów równych 3 minut,

Rys. 3



Listing 1. Przykład definicji dostępu bitowego do rejestrów procesora

```
typedef struct _bit_struct
{
  unsigned char bit0: 1;
  unsigned char bit1: 1;
  unsigned char bit2: 1;
  unsigned char bit3: 1;
  unsigned char bit4: 1;
  unsigned char bit5: 1;
  unsigned char bit6: 1;
  unsigned char bit7: 1;
}pole_bitowe;
#define DAJ_BIT(adr) (*((volatile pole_bitowe*) (adr)))
// okreslenie adresow rejestrow portow przestrzeni adresowej pamieci RAM
#define _PORTC 0x35
#define _DDRC                    0x34
#define _PINC                    0x33
#define LED_CON DAJ_BIT(_PORTC).bit7  // dioda kontrolna LED_CON = 1/0
```

```
void wyslij_godzine(void)
{
  unsigned char bufor[5];                          //bufor dla I2C
  unsigned char x1;
  bufor[0] = byte2bcd(czas.sek);                   //sekundy
  bufor[1] = byte2bcd(czas.min >> 2);              //minuty
  x1 = czas.godz / 20; 00 na 12
    if(x1 == 0)                                    //zamien godzine
  x1 = 12;
  bufor[2] = byte2bcd(x1) | 0x80;                  //tryb 12h, godziny
  x1 = (czas.dni / 15) & 0x01;                     //polowa dnia tygodnia      (AM/PM)
  bufor[2] = (bufor[2] & 0xBF) | (x1 << 6);        //AM/PM
  bufor[3] = 0;                                    //data
  bufor[4] = byte2bcd(czas.dni / 30) << 5;         //dzien tygodnia
  i2c_wyslij(RTC_ADR,2,5,bufor);                   //wyslij 5 bajtow od adresu 0x02
}
```

Listing 2. Funkcja zapisu czasu do RTC

a takich jest w prawdziwej godzinie 20. Na tarczy zegara mamy 12 godzin więc z prostego działania 20*12 otrzymujemy odpowiedź, skąd wziął się zakres „godzin". I ostatecznie po upływie 16 pośrednich godzin (co przekłada się na 16*3min = 48min = 1/30 doby) inkrementowana jest liczba „dni". Ta z kolei jest w istocie miarą okresów równych 48 minut a tych jest w dobie 30. Mamy 7 dni tygodnia, co daje 7*30 = 210. I w ten sposób rozwiązała się ostatnia zagadka.

Dzięki tym z pozoru skomplikowanym i „kombinowanym" zabiegom, możliwy jest płynny ruch wskazówek minutowej, godzinowej i wskazującej dzień tygodnia, podobnie jak ma to miejsce w klasycznym zegarze. Obliczone wyżej wartości są ładowane do przetwornika, gdzie przekładają się na odpowiednie wartości napięcia. Jeżeli zdarzy się, że jedna lub więcej wartości przekroczy swój zakres i wskazówki muszą wrócić do położenia na początku skali, zostanie wprowadzona procedura w miarę szybkiego, a zarazem „bezbolesnego" zmniejszenia napięcia na przetworniku, tak aby wskazówka nie uderzyła w ściankę miernika.

Ostatnią istotną operacją, wykonywaną w przerwaniu, jest załadowanie odpowiedniego efektu podświetlenia. Definicje efektów znajdują się w pliku backlight.c. Umieszczone są tu także funkcje związane z fizyczną obsługą drivera LED: *backlight_init, backlight_send* i *backlight_off*. Właśnie te trzy funkcje należy zmienić, chcąc użyć układu MBI5031, między innymi usuwając procedurę włączenia wewnętrznego generatora i zamieniając ją na włączenie generacji sygnału zegarowego na jednym z liczników.

Aby ułatwić dodawanie kolejnych efektów, stworzona została tablica *efekty[]*. Jest to tablica zawierająca wskaźniki do funkcji sterujących podświetleniem, a zmienna *efekt*, którą jest indeksowana tablica, określa numer aktualnego efektu. Dodanie kolejnego efektu sprowadza się (oczywiście oprócz napisania samej funkcji) do zwiększenia stałej *EFEKT_MAX* w pliku zegar_main.h i do przypisania kolejnej funkcji do tablicy w funkcji *backlight_init*. To wszystko! W podstawowej wersji zdefiniowane są trzy efekty: wyłączenie podświetlenia (jeżeli można to nazwać efektem ;-)), chwilowe podświetlenie na 8 sekund kolorem białym z efektem fade-in i fade-out (przydatne w nocy) oraz efekt niezależnego podświetlenia każdego miernika kolorem uzależnionym od jego obecnego wskazania. W planach są kolejne interesujące efekty, a działanie obecnych można zobaczyć na plikach wideo, umieszczonych w Elportalu.

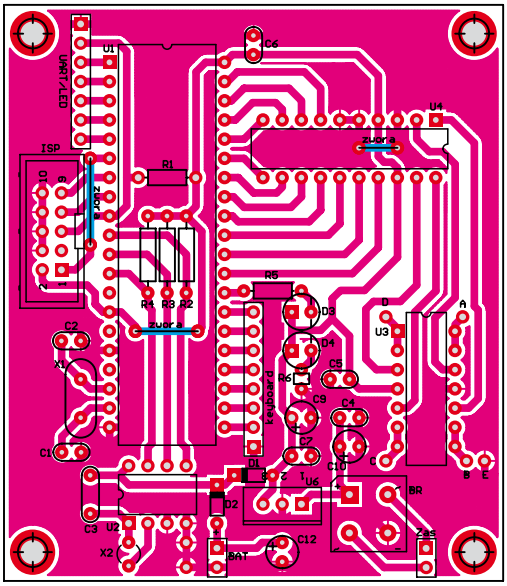
```
//=============================================================
//===== Przerwanie 1Hz ========================================
//=============================================================
ISR(INT0_vect)
{
  unsigned char i1,i2;
  LED_CON ^= 1;                    //miganie kontrolne
  i2 = 0;
  czas.sek++;                      //zwieksz sekundy
  czas.sek_p++;                    //zwieksz posrednie sekundy
  if(czas.sek_p > 14)              //minela 1/4 minuty
  {
    czas.sek_p = 0;                //wyzeruj posrednie sekundy
    czas.min++;                    //zwieksz minuty
    czas.min_p++;                  //zwieksz posrednie minuty
    if(czas.min_p > 11)            //minela 1/20 godziny (3 minuty)
    {
      czas.min_p = 0;              //wyzeruj posrednie minuty
      czas.godz++;                 //zwieksz godziny
      czas.godz_p++;               //zwieksz posrednie godziny
      if(czas.godz_p > 15)         //minela 1/30 doby (48 minut)
      {
        czas.godz_p = 0;           //wyzeruj posrednie godziny
        czas.dni++;                //zwieksz dni
      }
    }
  }
  if(czas.sek > 59)                //minela minuta
  {
    czas.sek = 0;                  //wyzeruj sekundy
    i2 |= SEK;
  }
  if(czas.min > 239)               //minelo 60 minut
  {
    czas.min = 0;                  //wyzeruj minuty
    i2 |= MIN;
  }
  if(czas.godz > 239)              //minela doba
  {
    czas.godz = 0;                 //wyzeruj godziny
    i2 |= GODZ;
  }
  if(czas.dni > 209)               //minal tydzien
  {
    czas.dni = 0;                  //wyzeruj dni
    i2 |= DNI;
  }
  if(i2 != 0)                      //jezeli nalezy wygasic ktorykolwiek z miernikow
  {
    for(i1=209;i1>0;i1--)
    {
      laduj_DAC(i2,i1);            //stopniowe wygaszanie wszystkich wymaganych wartosci
      _delay_ms(2);
    }
  }
  laduj_DAC(SEK,czas.sek << 2);    //ladowanie sekund
  laduj_DAC(MIN,czas.min);         //ladowanie minut
  laduj_DAC(GODZ,czas.godz);       //ladowanie godzin
  laduj_DAC(DNI,czas.dni);         //ladowanie dni
  (*efekty[efekt])();              //wywolaj odpowiedni efekt podswietlenia
  (...)
}
//=============================================================
```

Listing 3. Obsługa przerwania 1Hz

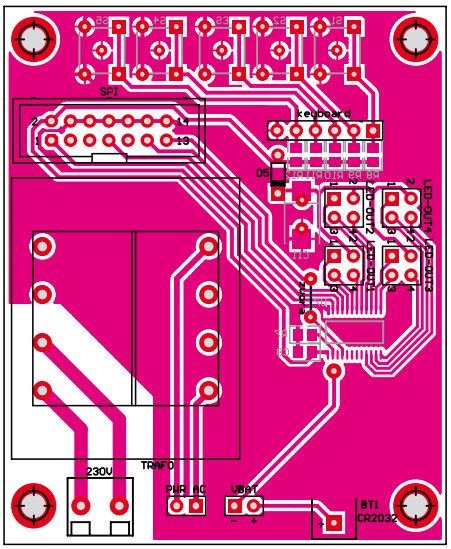
Montaż i uruchomienie

Sposób montażu płytki głównej (**rysunek 4**) jest klasyczny. Rozpocząć należy od wlutowania 3 zworek, a następnie podstawek pod układy scalone i kontynuować w wielokrotnie opisywany sposób, kończąc na elementach największych gabarytowo. Przed montażem płytki pomocniczej można ją wykorzystać jako szablon do wywiercenia otworów w obudowie, przez które wyprowadzone będą przyciski. Do tego celu służą dodatkowe pady umieszczone dokładnie pod geometrycznym środkiem każdego z przycisków.

Płytka pomocnicza (**rysunek 5**) wymaga nieco większej precyzji ze względu na obecność elementów SMD, które najlepiej zamontować na początku. W drugiej kolejności lutujemy przyciski, które jak wspomniałem umieszczone są na spodniej stronie płytki. Następnie lutujemy elementy przewlekane, kończąc na transformatorze. Łączymy obie płytki ze sobą odpowiednimi przewodami. Na tym etapie najważniejszy jest przewód zasilający płytkę główną. Po dostarczeniu zasilania przystępujemy do zaprogramowania procesora i ustawieniu fusebitów. Po tej czynności układ powinien od razu zacząć pracować, czego oznaką będzie miganie diody kontrolnej

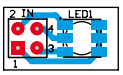


Rys. 4



Rys. 5

Rys. 6



D3 z częstotliwością 0,5Hz. Układ w zasadzie nie wymaga regulacji, choć jeżeli pod ręką jest dokładny miernik częstotliwości, można spróbować wyregulować układ RTC trymerem C3, mierząc sygnał na nóżce nr 7 układu U2. Powinna ona wynosić dokładnie 1Hz.

Nieco szerszego komentarza wymaga przygotowanie mierników. Jak już wcześniej wspomniałem, należy usunąć oryginalne rezystory i zastąpić je potencjometrem, którego śrubkę regulacyjną umieszczamy w wywierconym niewielkim otworze. Ponadto należy nawiercić otwór o średnicy minimalnej około 6,5mm, przez który wyprowadzone zostanie złącze niewielkiej płytki z diodą LED RGB – **rysunek 6**. Płytkę należy wkleić tak, aby dioda znajdowała się możliwie pod środkiem skali.

Oryginalną skalę należy usunąć przez odkręcenie dwóch śrub przytrzymujących całą mechanikę miernika. Na jej miejsce zostanie zamontowana ***nowa skala, wykonana z kawałka pleksi z naklejoną białą folią***. Brak oryginalnej skali może spowodować chwiejne mocowanie mechaniki, dlatego warto zastosować podkładki pod śruby mocujące. Szablon do laserowego wycięcia skali znajduje się w pliku ***plexi.pdf***, choć firma świadcząca takie usługi prawdopodobnie będzie wolała skorzystać z pliku ***plexi.cdr***. W moim przypadku ta sama firma wykonała wydruk skali na samoprzylepnej białej folii wraz z plotowaniem. Potrzebne pliki można ściągnąć z Elportalu. Cała usługa wycinania i drukowania dwóch kompletów kosztowała mnie równo 30PLN wraz z materiałami (folia i pleksi). Porządna skala to kluczowy element. Właśnie brak możliwości technicznych w domu, a może raczej brak wiedzy o łatwym dostępie do nich był główną przyczyną zniechęcenia i odłożenia projektu na tak długi czas.

Montowanie skali odbywa się poprzez przykręcenie jej na dystansowniku 20mm, ewentualnie z dodatkową nakrętką podwyższającą go o kolejne 2mm. Oczywiście należy wcześniej wywiercić w tylnej ściance miernika otwór na śrubkę. Precyzyjne wywiercenie tego otworu może być trudne i prawdopodobnie konieczne będzie wypiłowanie podłużnego otworu umożliwiającego regulację położenia dystansownika. Wszystko to jest warte zachodu, bo światło diody LED rozproszone na tak skonstruowanej skali robi naprawdę wspaniałe wrażenie, co trudno oddać na fotografiach.

Obsługa

Całe sterowanie zegarem odbywa się za pomocą 5 przycisków. Przycisk nr 5 jest odpowiedzialny za zmniejszenie jasności podświetlenia. Jego naciśnięcie powoduje obniżenie jasności o 1 stopień, aż do osiągnięcia minimum po 7 naciśnięciach. Po przekroczeniu minimalnej wartości jasność wraca do poziomu maksymalnego.

Przycisk nr 4 służy do zmiany efektu podświetlenia. Jego naciśnięcie zmienia efekt na kolejny, a po przekroczeniu indeksu ostatniego efektu wraca do efektu 1, czyli wyłącza podświetlenie. Jedynym efektem, którego nie można włączyć w ten sposób, jest efekt 0 zarezerwowany dla chwilowego podświetlenia, które może zostać wywołane przez naciśnięcie przycisku 1 lub 2, o ile nie jest włączony inny rodzaj podświetlenia.

Naciśnięcie przycisku nr 3 wprowadza zegar w tryb ustawiania godziny. W tym trybie przyciski 1 i 2 służą do inkrementacji i dekrementacji wartości na danym mierniku. Po ustawieniu sekund, należy ponownie nacisnąć przycisk nr 3, co spowoduje przejście do ustawiania minut. Aktualnie ustawiany miernik będzie podświetlony kolorem ustawionym w programie, a pozostałe będą wygaszone. Ustawianie wszystkich wartości jest skokowe, przy czym dla sekund i minut występuje rzecz jasna 60 położeń, dla godzin 12, a dla dni tygodnia 14. Tych ostatnich jest 14, a nie 7 ze względu na konieczność ustalenia, czy wprowadzona godzina odpowiada wartości przed czy po południu (AM/PM).

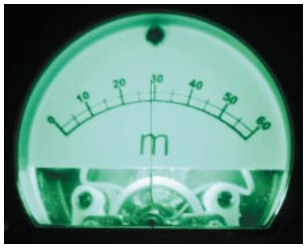
Odpowiednie „dotrymowanie" wskazań nastąpi automatycznie po wyjściu z trybu ustawiania, czyli po naciśnięciu przycisku nr 3 w momencie ustawiania dni tygodnia. Jeżeli żaden przycisk nie zostanie naciśnięty dłużej niż 20 sekund, zegar automatycznie powróci do normalnej pracy, przywracając poprawny

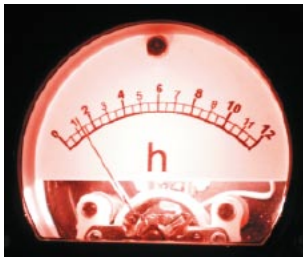
czas przez załadowanie go z układu RTC.

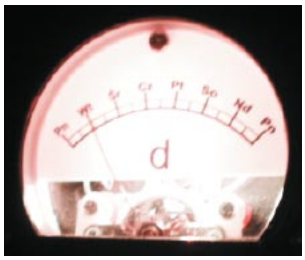
Możliwości zmian

Najważniejszą i w zasadzie konieczną do wykonania zmianą jest zamiana trudno dostępnego układu DM631 na MBI5031. Nie trzeba przy tym nic zmieniać od strony sprzętowej, a zmiany programowe zostały wyjaśnione wcześniej. Po szczegóły odsyłam do noty katalogowej tego układu.

Kolejną istotną sprawą jest wymiana przetwornika na tańszą kostkę. Bardzo zbliżony do układu MAX506 jest układ MAX5100, przy czym jest on ponad trzykrotnie tańszy. Istnieje między nimi kilka różnic w wyprowadzeniach, a najważniejsza z nich to fakt, że z jakiegoś powodu (ten sam!) producent postanowił zamienić miejscami nóżkę zasilania z nóżką masy. Należy je więc pozostawić poza podstawką i odpowiednio dołączyć przewodami. Ponadto konieczne będzie odłączenie nóżki nr 6 od masy i połączenie jej z nóżką 15 lub zamiana tych wyprowadzeń miejscami. Tu także odsyłam po szczegóły do noty.







Może okazać się, że napięcie referencyjne dla przetwornika w postaci napięcia zasilania jest niewystarczająco stabilne, co spowoduje np. zauważalne różnice we wskazaniach przy kolejnych uruchomieniach. W takim wypadku najlepiej zastosować dodatkowe źródło referencyjne, najlepiej o niższym napięciu i odpowiednio wyregulować mierniki. Pozwoli to także na zastosowanie tańszej kostki ze wzmacniaczami operacyjnymi, gdyż nie będzie już potrzebny wzmacniacz rail-to-rail, a jedynie taki, który potrafi pracować od poziomu masy np. LM324.

Alternatywą dla stosowania gotowych przetworników C/A jest zastosowanie odpowiednich sygnałów PWM. Można spróbować generować je programowo, ale trzeba będzie zadbać, aby nie znikały na zbyt długi czas, żeby wskazówki nie zaczęły opadać. Pomocny w takim wypadku będzie odpowiednio dobrany układ RC, podtrzymujący chwilowo napięcie, ale nie może on także zbytnio wydłużać przesuwania wskazówek.

Ciekawą opcją mogłoby okazać się wykorzystanie nieużywanych 4 kanałów drivera LED. Wyjścia te należałoby obciążyć odpowiednimi rezystorami ze względu na ich specyficzną budowę – źródła prądowe. Wystarczyłoby jeszcze dodać odpowiedni układ RC i gotowe – uzyskujemy ten sam efekt bez konieczności stosowania przetwornika C/A i być może bez dodatkowego bufora (wydajność prądowa wyjść drivera jest aż nadto wystarczająca do wysterowania mierników).

Jeżeli chodzi o zmiany programowe, to poza opisywaną możliwością dodania nowych efektów można także poeksperymentować, np. z nieco zmienionymi skalami. W pliku ze skalami dodane są także dwie dodatkowe, jedna to skala dla dni tygodnia przesunięta o pół dnia, a druga to skala 24-punktowa dla godzin. Ponadto polecam wyprowadzenie na zewnątrz sygnałów UART-a wraz z masą i ewentualnie sygnału reset i zastosowanie dołączonego bootloadera. Pozwoli to na łatwą zmianę oprogramowania zegara bez jego rozbierania. Bootloader został skonstruowany tak, że uruchamia się zawsze po włączeniu zasilania lub po resecie procesora i czeka przez 5 sekund na odpowiednią komendę. Dołączam także zmodyfikowany przeze mnie program AVROSP do obsługi tego sposobu programowania. Po wgraniu bootloadera do pamięci Flash procesora nie należy zapomnieć o ustawieniu odpowiedniego fusebitu, aby po resecie procesor zaczynał pracę od bootloadera. Wspomnę jeszcze o ułatwiającej pracę z bootlo-aderem funkcji, a mianowicie o możliwości zdalnego zresetowania mikrokontrolera przez wysłanie po UARC-ie znaku „r". Rzecz jasna, aby połączyć układ z komputerem, należy zastosować odpowiednią przejściówkę zmieniającą poziomy napięć np. w postaci układu MAX232.

Jeżeli okaże się, że w układzie pojawiają się nieliniowości (najczęściej związane z samym miernikiem) objawiające się nierównym przesuwaniem wskazówek, najprostszym rozwiązaniem będzie dodanie tablicy LUT (Look-Up Table), która będzie zawierała odpowiednie korekty w postaci wartości ładowanych do przetwornika, a jej indeksowanie będzie następowało liniowymi wartościami dotychczas wysyłanymi bezpośrednio na przetwornik. Tak więc tablice takie dla poszczególnych mierników w najprostszej postaci miałyby 60, 240, 240 i 210 elementów.

Filip Rus
filip.rus@livelights.pl

Wykaz elementów

Rezystory
R1,R4,R8-R12 ..... 10kΩ
R2,R3 ..... 1kΩ...4,7kΩ
R5,R6 ..... 1kΩ
R7 ..... 3,3kΩ* (zmienić dla MBI5031)

Kondensatory
C1,C2 ..... 33pF
C3 ..... 25pF (najlepiej trymer)
C4-C8 ..... 100nF
C9-C11 ..... 100µF
C12 ..... 10µF

Półprzewodniki
D1,D2 ..... 1N4148
D3,D4 ..... LED
D5 ..... 1N4007
BR ..... mostek 1A
U1 ..... ATmega162
U2 ..... PCF8583
U3 ..... TLC2274
U4 ..... MAX506
U6 7805
U7 ..... DM631/MBI5031

Pozostałe
X1 ..... kwarc 8MHz
X2 ..... kwarc 32,768kHz
BT1 ..... podstawka pod baterię CR2032
S1-S5 ..... microswitch 6x6
TRAFO ..... TS 2/14 lub większy
ARK2
Goldpiny pojedyncze i podwójne
4 miliamperomierze wskazówkowe
4 diody LED RGB np. PLCC4

Płytka drukowana jest dostępna w sieci handlowej AVT jako kit szkolny AVT-2973.