kit 2870 AVT

Wielofunkcyjny licznik rowerowy czyli pierworodne dziecko Konkursu C

Większość urządzeń elektronicznych możemy wykonać samodzielnie, jednak są i takie, których budowa jest nieopłacalna. Powody są różne: większe koszty, wymiary, pobór prądu. Wydawałoby się, że liczniki rowerowe należą do tej drugiej grupy. Czy aby na pewno? Poniższy artykuł jest dowodem tego, że tak nie jest. Od kiedy mikrokontrolery stały się tanie i łatwo dostępne, możemy sobie pozwolić na budowę takich układów. Jakby tego było mało, prezentowany licznik przewyższa parametrami fabryczne urządzenia i jest dosyć tani. Ponadto, istnieje możliwość samodzielnego programowania i dodawania własnych funkcji. Tak więc, znając język C, możemy dostosować program do swoich potrzeb.

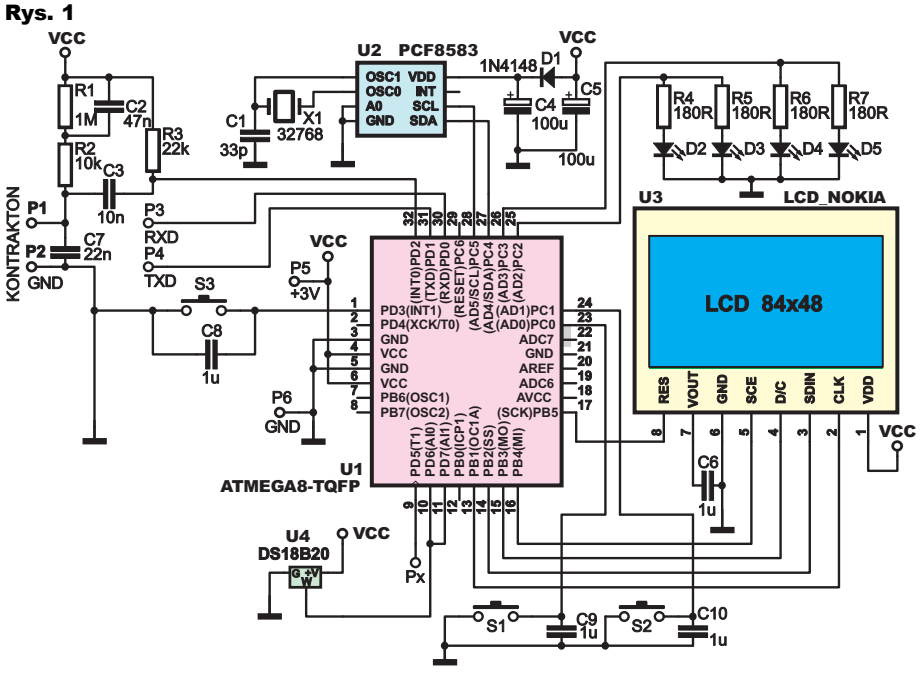
Interfejs RS-232 pozwala na komunikację z komputerem lub z innymi urządzeniami. Można dopisać odpowiedni podprogram, który sprawi, że dzięki temu interfejsowi nasz licznik staje się monitorem np. dla modułu nawigacji GPS. Rozdzielczość 84x48 pikseli wystarcza na wyświetlenie prostej mapki, nazw ulic i miast. Proponuję teraz porozglądać się za licznikami z najwyższej półki. Co prawda wytrzymałością i estetyką nie będzie łatwo dorównać takim maszynom. Za to funkcje będziemy mieli takie, o jakich tylko można pomarzyć. Graficzny wyświetlacz pozwala na wiele dodatkowych możliwości oraz zapewnia wysoki komfort pracy. Pewnie myślisz, drogi Czytelniku, że taki licznik jest niezwykle skomplikowany i drogi. Zapomnij o tym, co przed chwilą widziałeś w sklepach. Prezentowany układ zbudowany jest z tanich i łatwo dostępnych części, a jego cena jest niższa od podobnych fabrycznych urządzeń. Niestety coś za coś. Teraz muszę Cię zmartwić. Budowa solidnego licznika, a w szczególności jego obudowy, nie jest prostym zadaniem i wymaga dużo pracy. Jednak, cokolwiek by to nie było, warto się do tego zabrać. Chyba nie muszę mówić ile satysfakcji daje własny projekt. Nawet jeśli nie jesteś fanem rowerów, to i tak myślę, że zainteresują Cię użyte tutaj rozwiązania.

Parametry prezentowanego licznika rowerowego:
- 12 funkcji pomiarowych:
  aktualny czas, data, stoper, czas jazdy, prędkość chwilowa, prędkość maksymalna, całkowita prędkość średnia, prędkość średnia jazdy, dystans całkowity, dystans dzienny, dystans do zrobienia, temperatura
- manualna oraz automatyczna zmiana funkcji
- graficzny wyświetlacz z podświetleniem
- interfejs RS-232
- wymiary 65x46x15mm
- wodoodporny, wstrząsoodporny
- czas pracy baterii:
  w trybie aktywnym – 200 godzin (1,5mA),
  w stanie spoczynku – > 6lat (3µA, zależy od warunków otoczenia).

Opis układu

Schemat ideowy licznika rowerowego znajduje się na **rysunku 1**. Najważniejszą rolę pełni mikrokontroler Atmega8L. Jest to minimum, jakie trzeba zastosować. Słabsze mikroprocesory nie poradzą sobie z tak dużą ilością danych do przetworzenia i formowaniem grafiki. Natomiast lepsze pobierają więcej prądu, a ich możliwości nie są w pełni wykorzystane. Częstotliwość pracy też nie powinna być zbyt duża. Atmega8 taktowany jest wewnętrznym oscylatorem RC o częstotliwości 1MHz. Jako wyświetlacz został użyty moduł Nokia 3310. Jest on bardzo tani, energooszczędny i nadaje się do pracy w trudnych warunkach. Ramka tego wyświetlacza pozwala dodać podświetlenie, co niezmiernie przydaje się podczas nocnych podróży. Składają się na nie cztery diody. Ze względu na duży pobór prądu sterowane są z dwóch nóżek mikrokontrolera. Rezystory R4-R7 ograniczają prąd każdej diody do około 7mA, co daje w sumie

Rys. 1



28mA. A więc podświetlenia używajmy rozważnie, aby nie doprowadzić do szybkiego rozładowania baterii. Klawiaturę tworzą trzy przyciski. Jeden z nich (S3) podłączony jest do wejścia INT1 procesora. Pozwala to na szybką reakcję (funkcja stopera) oraz budzenie z trybu *power down*. Klawiatura jest sprawdzana co 0,2 sekundy (nie dotyczy S3 dla stopera). Może to powodować, że nie każde naciśnięcie będzie wykrywane. Tę niedogodność likwidują kondensatory C8, C9 i C10. W przypadku S3 są eliminowane drgania zestyków. To tyle, jeśli chodzi o interfejs użytkownika. Czas zająć się pozostałymi układami. Zegar czasu rzeczywistego (RTC), czyli PCF8583, pracuje w standardowej aplikacji z kondensatorem C1 i rezonatorem kwarcowym 32768Hz. Komunikuje się on z mikrokontrolerem przez I^{2}C. Ten interfejs tworzą układy z wyjściami typu open-drain, czyli otwarty kolektor. Wymagane jest więc podciąganie szyn SDA i SCL do plusa zasilania. Jest to realizowane w sposób programowy rezystorami podciągającymi w Atmega8. Komplikuje to trochę program, ale za to znacznie upraszcza ścieżki na płytce. Zasilanie układu PCF8583 doprowadzone jest przez diodę D1. Zapobiega to „cofaniu się prądu" i szybkiemu rozładowaniu kondensatora C4. Dotyczy to wymiany baterii. Nawet jeśli przed wyłożeniem baterii procesor znajdował się w trybie *power down*, to i tak po ponownym uruchomieniu startuje normalnie i pobiera znaczny prąd. A nie ma żadnej gwarancji na to, że przy zamykaniu obudowy sprężynka będzie dobrze dociskała nową baterię. Obwód D1, C4 eliminuje takie przerwy dla układu PCF8583, a pojemność kondensatora wystarcza na pracę zegara przez kilkadziesiąt sekund. Mamy więc pewność, że czas, data i inne ważne parametry zostaną zachowane. No dobrze, w takim razie po co drugi taki kondensator przed diodą? W trakcie jazdy rowerem wytwarzane są drgania, co może prowadzić do chwilowego „oderwania" się baterii od styków. Taki impuls na pewno zakłóciłby pracę procesora, gdyby nie równolegle włączony C5. Przecież licznik musi być niezawodny w każdych warunkach. Kolejnym układem na schemacie jest termometr DS18B20. Do komunikacji służy interfejs 1Wire. Tu również wymagane jest podciąganie. Żeby znowu nie komplikować programu, rezystor podciągający pochodzi z sąsiedniego wyprowadzenia procesora. Nie trzeba wtedy zmieniać ustawienia przy wysyłaniu każdego bitu, jak w przypadku I^{2}C. Wystarczy jednorazowe ustawienie podciągania na początku programu. Pozostał jeszcze jeden blok do omówienia, składający się z rezystorów R1, R2 i R3 oraz kondensatorów C2, C3 i C7. Elementy R2, R3, C3 i C7 formują odpowiedni impuls pochodzący z kontaktronu. R1 i C2 ograniczają prąd w przypadku pozostawienia na dłuższy czas zwartego kontaktronu.

```
czas[0]=_odPCF(4);//czytaj tylko godziny. Bity 7-8 są cały czas wyzerowane(tryb 24h)
        7//w przypadku gr tryb ma być 12h, to trzeba dodać &3f
czas[1]=_odPCF(3);//czytaj minuty
czas[2]=_odPCF(2);//czytaj sekundy
data[0]=_odPCF(5);//czytaj dzień miesiąca oraz bit roku
data[1]=_odPCF(6)&0x1f;//czytaj miesiąc bez dnia tygodnia

//rejestr 0x5 w PCF - rok | rok | 10dzień | 10dzień | dzień | dzień | dzień | dzień
//                  rok - 0-3        10dzień - 0-3          dzień - 0-9
if(data[0]&0x40) //odczytanie, czy rejestr roku w PCF jest większy od 0
{
++data[2];   //jeśli tak, to zwiększenie zmiennej roku
doPCF((data[0] & 0x3f),5); //wyzerowanie roku w rejestrze PCF bez naruszania dnia miesiąca
}
```

Listing 1

```
if(!sstart)  //start
{
bS=aS; //wpisanie odczytanej zmiennej do zmiennej początkowej
sstart=1;  //ustawienie stopera na START
}else{       //stop
wS+=aS-bS; //obliczenie odcinka czasu, zmienna odczytana (naciśnięto stop)
        minus zmienna początkowa i dodanie do zmiennej przechowującej odcinki mierzonego czasu
sstart=0; //ustawienie stopera na STOP
}
if(pr2==2) //wyzerowanie
{
wS=0;  //wyzerowanie całego zmierzonego czasu
sstart=0; //ustawienie na STOP
}
```

Listing 2

```
if(sstart){  //wykonywane w czasie pracy stopera
             //sto – stoper, zmienna, która wędruje na wyświetlacz
sto=bin(_odPCF(1))+(bin(czas[2]))*100+(bin(czas[1]))*6000;//zamiana trzech liczb bcd na jedną liczbę
sto+=wS-bS; //odjęcie składowej stałej (odjęcie zmiennej początkowej i dodanie odcinków zmierzonych wcześniej)
}else{ //wykonywane, jeśli stoper zatrzymany
sto=wS; //wpisanie odmierzonego czasu
}
stoperW(sto);//wyświetl zmienną stopera
```

Listing 3

Program

Pełny program i jego opis znajduje się na Elportalu. Tutaj przedstawię tylko działanie funkcji licznika. Pomoże to przy próbie ich modyfikacji. Całość składa się z pliku głównego – *Licznik.c*, pliku z funkcjami niższego poziomu *Function.c* i *Function.h*, pliku definicji portów mikrokontrolera *harddef.h* oraz pliku ułatwiającego dostęp do portów *makra.h*. Ogólnie program jest pisany dość sztywno, aby uzyskać jak największy stopień optymalizacji kodu wynikowego. Dotyczy to w szczególności pliku *Function.c*. Funkcje znajdujące się w głównym pliku *Licznik.c* dają więcej swobody. Właśnie tam znajdują się wszystkie funkcje pomiarowe i tylko to nas będzie za chwilę interesować. Na początek ogólny zarys programu. Funkcje obsługi wyświetlacza są nieco zmodyfikowaną wersją tych z 13 części kursu C. Pisząc obsługę I^{2}C i 1Wire też opierałem się na gotowych rozwiązaniach. Czas na główny plik. Funkcja *main* składa się z dwóch dużych pętli *while*. Pierwsza dla ustawień i druga dla właściwej pracy licznika. Po każdym przebiegu pętli program przechodzi w stan *idle* dla oszczędności energii. Praca wznawiana jest co 0,2 sekundy przy przerwaniu od timera0 obsługującego klawiaturę lub innych przerwań, np. od kontaktronu czy przycisku S3. Wartość przycisków S1 i S2 zwracana jest jako jedna liczba (0-5) po zwolnieniu klawiszy. Dla S3 wartość ta zwracana jest jednorazowo zaraz po naciśnięciu. Gdy zostanie przytrzymany po pewnym czasie (ok. 1 sekundy) zwracana jest cyklicznie wartość długiego naciśnięcia. Kody klawiszy znajdują się w dalszej części artykułu w rozdziale obsługa. Nad niezawodnością czuwa wachtdog. Zerowany jest w każdej pętli programu. Czas na funkcje pomiarowe. Omówimy je po kolei.

Zegar i data. Wprawdzie z pomiarem niewiele ma wspólnego, ale występuje w większości liczników. Tu wystarczy tylko odczytać wartości rejestrów układu PCF8583. Pokazuje to **listing 1**. Odczytanie roku jest nieco inne. W rejestrze może znajdować się liczba od 0 do 3, a nam potrzebne są większe wartości. Dlatego w mikrokontrolerze jest dodatkowa zmienna przechowująca rok. Jeśli nastąpi zmiana w rejestrze PCF8583, zmienna roku jest inkrementowana, a rejestr zerowany.

Stoper. Algorytm jest inny niż w typowych urządzeniach tego typu. Musimy pamiętać o tym, że zliczanie ma się odbywać także w trybie *power down*. Znowu z pomocą przychodzi PCF8583. Naciśnięcie S3 powoduje wpisanie aktualnego czasu wraz z setnymi sekundami z tego układu, do zmiennych w procesorze. Następnie te zmienne konwertowane są do jednej długiej liczby (aS), co pozwala łatwo wykonywać obliczenia. Potem w zależności od stanu stopera wykonywane są odpowiednie instrukcje. Jednorazowo **listing 2**, oraz z każdym przebiegiem pętli **listing 3**. Zasadę działania najlepiej wyjaśni **rysunek 2**. Nad osią podane są instrukcje wykonywane raz, a pod osią wykonywane w pętli. W nawiasach są przykładowe wartości, jakie będą wyświetlane w danym momencie. Najprościej mówiąc, program mierzy odcinek czasu. Ponowne uruchomienie stopera bez resetowania powoduje pomiar następnego odcinka i dodanie tego pierwszego. W taki sposób zachowana jest ciągłość. Kolejna funkcja ma podobne zadanie, ale jest o wiele prostsza.

```
if((czas[2]!=aczas)&&(!startl)) //sprawdzenie, czy nastąpiła zmiana sekundy w czasie rzeczywistym
{
//dla czasu jazdy
aczas=czas[2]; //odświeżenie zmiennej dla sprawdzenia kolejnej sekundy
if(++czj[2]>59) //jeśli przepełnienie sekund
{
        czj[2]=0; //wyzeruj sekundy
        if(++czj[1]>59) //zwiększ minuty i jeśli przepełnienie
        {
                czj[1]=0; //wyzeruj minuty
                if(++czj[0]>0x99)czj[0]=0; //zwiększ godziny i jeśli przepełnienie to wyzeruj
        }
}
//dla prędkości średniej jazdy
static uint8_t aatj; //zmienna pomocnicza dla odliczania czasu w prędkości średniej jazdy
if(++aatj>(2*pr_sr)){++atj;aatj=0;} //jeśli odliczono odpowiedni odcinek czasu
}
```

Listing 4

Czas jazdy. Jak sama nazwa wskazuje, liczy czas podczas gdy się poruszamy. Tutaj wystarczy tylko inkrementować zmienną czasu jazdy, gdy prędkość chwilowa jest większa od zera. Prawda, że proste? **Listing 4** pokazuje jej realizację i odmierzanie 1 sekundy. Na dole jest kod dla prędkości średniej, ale o tym za chwilę.

Prędkość chwilowa. To chyba najważniejsza funkcja. Tu z pomocą przychodzi Timer1 mikrokontrolera. Od prędkości poruszania się rowerzysty zależy szybkość zwierania kontaktronu przez magnes. Czas pomiędzy kolejnymi impulsami odlicza Timer1. Czym większa prędkość, tym mniej zliczy, więc trzeba tę wartość przetworzyć na km/h z uwzględnieniem obwodu koła. Pokazuje to **listing 5**.

Prędkość maksymalna również korzysta z listingu 5, ale jako parametr podaje najmniejszą wartość zmierzoną przez Timer1. Reszta jest taka sama jak przy obliczaniu prędkości chwilowej.

```
uint16_t predkosc(uint16_t aapr) //aapr- wartosc odczytana z Timera1
{
uint16_t i; //zmienna pomocnicza
aapr=aapr/20;   //pierwotne podzielenie, aby dalsze operacje były wykonywane
                  na liczbach 16- a nie 32-bitowych
i=kalib*28/aapr;  //kalib - obwód koła w milimetrach
if(i>999)i=999; //zapobiega wyjściu poza obszar tablicy znaków cyfr
return(i);  //zwraca wynik w km/h
}
```

Listing 5

Prędkość średnia całkowita liczona jest cały czas, nawet jeśli rower stoi w miejscu, a licznik jest w stanie spoczynku. Oblicza się ją ze wzoru v=s/t. Trzeba więc zmierzyć dystans (o tym będzie dalej) oraz czas i podzielić te wartości. Czas jest mierzony w taki sam sposób jak w stoperze, tylko bez zmiennej przechowującej odcinki czasu (wS w stoperze). Uniemożliwia to zatrzymanie liczenia, możliwe jest tylko wyzerowanie. To w zupełności wystarcza do przeciętnych pomiarów.

Prędkość średnia jazdy liczona jest tylko w trakcie poruszania się. Pomiar czasu wygląda tak samo jak dla funkcji czas jazdy. Pokazuje to listing4. Aby uzyskać większą dynamikę pomiarów, został wprowadzony dodatkowy dzielnik (zmienna pr_sr). Przy dalszych dystansach i dłuższym czasie powoduje wolniejsze naliczanie i pozwala bez utraty dokładności dokonywać przeliczeń na liczbach 16-bitowych, a nie 32-bitowych.

Dystans dzienny. Nie ma w tym nic trudnego, wystarczy przy każdym przerwaniu z kontaktronu dodawać obwód koła do zmiennej przechowującej ten dystans. W programie są to trzy zmienne 8-bitowe, co pozwala w prosty sposób je wyświetlać. Z tego względu potrzebna jest jeszcze jedna zmienna licząca 10 metrów, i dopiero wtedy są inkrementowane i dekrementowane dystanse.

Dystans całkowity. Wygląda tak samo jak dystans dzienny, tylko nie da się go skasować. Jego zmienne przechowywane są w pamięci PCF8583. Jak wiadomo, układ ten

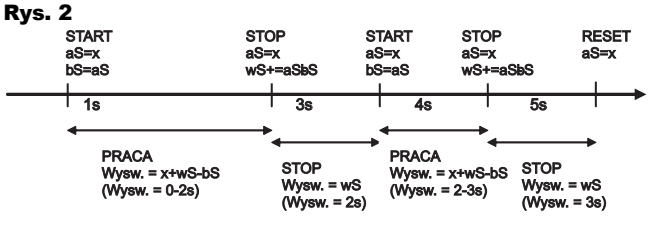
może pracować przy napięciu od 1V, co pozwala zachować dystans całkowity i inne ważne parametry nawet po rozładowaniu baterii lub jej wymianie.

Dystans do zrobienia. Jest zliczany do zera od ustawionej wartości. Całość pokazuje **listing 6**. Są tam również instrukcje do obliczania prędkości średnich oraz inicjacja procesora i wyświetlacza do normalnej pracy po wyjściu z trybu uśpienia.

Temperatura. To już ostatnia funkcja do omówienia. Cała trudność sprowadza się do odczytania temperatury z DS18B20. Potem tylko kilka operacji arytmetycznych i logicznych oraz sprawdzenie, czy jest ujemna. Przedstawia to **listing 7**.

To tyle o funkcjach licznika. Mam nadzieję, że choć trochę z tego rozumiesz. Cały program

Rys. 2



x – aktualny czas z układu PCF minuty, sekundy, setne sekundy
aS – do tej zmiennej zapisywany jest aktualny czas przy każdym naciśnięciu S3
bS – tutaj przepisywana jest zmienna aS przy starcie
wS – tu przechowywane są odcinki już Zmierzonego czasu
Wysw. – wyświetl, w programie zmienną wyświetlaną jest sto

jest bogaty w komentarze i jego działanie nie powinno być aż takie trudne do zrozumienia. Powyższy tekst dodatkowo ułatwi przedzieranie się przez gąszcz instrukcji.

Montaż i uruchomienie

Układ został zmontowany na jednostronnej płytce drukowanej z **rysunku 3**. Większość elementów jest typu SMD. Przyda się do tego stacja lutownicza. Zacznijmy od przylutowania złącza pod wyświetlacz. Będą to goldpiny obcięte na 1/3 ich długości. Lutujemy je tak, aby cyna nie wystawała ponad ich wysokość. Dalszy montaż jest tradycyjny, zaczynamy od najmniejszych elementów, a kończymy na układach scalonych, diodach LED i przyciskach. Pod układem PCF8583 jest otwór. Po co? Jak widać, układ ten jest bardzo gruby i nie można by potem prawidłowo położyć na płytce wyświetlacza z Nokii. Dotyczy to również innych części, oprócz przycisków. Elementy mogą być wyższe od złącza wyświetlacza o co najwyżej 0,1mm. Tę maksymalną wysokość ma układ Atmega8. Tak więc musimy odgiąć delikatnie nóżki układu zegara i dopasować go tak, aby jego spód był prawie na równo z drugą stroną płytki. A jak poradzić sobie z procesorem ATMEGA8 w obudowie TQFP? Można posłużyć się cienkim grotem i lutować jego nóżki po kolei albo grotem typu minifala lutować naraz całą stronę. Ja wybrałem tę drugą opcję, z tą różnicą, że użyłem zwykłego grotu, a zamiast specjalnego topnika kalafonii w spirytusie. Najpierw lutujemy po jednym wyprowadzeniu z każdego rogu, żeby procesor się nie zsunął. Potem smarujemy jedną stronę rozpuszczoną kalafonią. Następnie grotem i cyną przeciągamy przez cały rząd nóżek. Robimy to na styku końcówek z drukiem. Powtarzamy te czynności dla wszystkich stron. Jeśli zleją się nam nóżki, to przejeżdżamy jeszcze raz grotem. Gorzej, jeśli stało się to zaraz przy obudowie mikrokontrolera. Wtedy cienkim grotem przeciągamy od miejsca zwarcia w dół. Cyna powinna spłynąć niżej. Po takim zabiegu trzeba znowu nałożyć trochę kalafonii i ponownie przeciągnąć grotem przez cały rząd wyprowadzeń. Atmega8 tak łatwo się nie przegrzeje, jednak temperatura grota nie powinna przekraczać 260°C. Wygląda to całkiem nieźle, tylko trzeba trochę więcej się pomęczyć, nie mając profesjonalnych narzędzi. Kondensatory elektrolityczne i rezonator kwarcowy lutujemy z drugiej strony płytki. Tam również musi znaleźć się złącze. W moim przypadku jest to listwa goldpinów połączona z płytką przewodami. Pozwala to na odłączanie płytki od tylnej części obudowy. Połączenie z baterią najlepiej wykonać ze sprężynki. Drugi biegun będzie wtedy zwyczajną blaszką. Nie zapew-

Listing 6

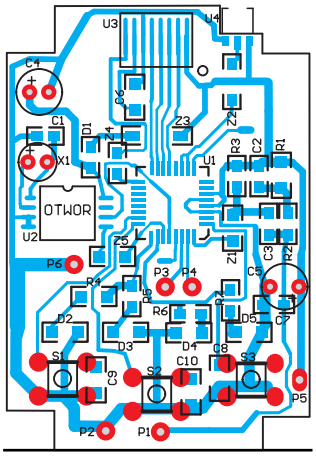
```
SIGNAL(SIG_INTERRUPT0) //przerwanie int0 od kontaktronu (wykonywane co jeden obrót przedniego koła)
{
static uint16_t liczd; //zmienna licząca 10 metrów
if(!startl)apr=TCNT1; //Jeśli nie było przpełnienia, to zapisz Timer1 w apr
TCNT1=0; //wyzeruj Timer1
startl=0;  //skasuj flagę przepełnienia
liczd +=kalib;  //dodaj do zmiennej pomocniczej wartość obwodu koła
if(liczd>10000) //jeśli przekroczy 10 metrów
{
        liczd-=10000;  //odejmnij 10 metrów
        if(++dc[2]>99) //inkrementuj dystans całkowity, jeśli przepełnienie
        {
                dc[2]=0;  //wyzeruj zmienną metrów
                if(++dc[1]>99)  //zwiększ kilometry i jeśli większe od 99
                {
                        dc[1]=0;  //wyzeruj kilometry
                        if(++dc[0]>99)dc[0]=0; //zwiększ setki kilometrów, jeśli przekroczy 99, to zeruj
                }
        }
        if(++dd[2]>99)   //tak jak wyżej
        {
                dd[2]=0;
                if(++dd[1]>99)
                {
                        dd[1]=0;
                        if(++dd[0]>99)dd[0]=0;
                }
        }
        if(--dzr[2]<0)   //tak jak wyżej
        {
                dzr[2]=99;
                if(--dzr[1]<0)
                {
                        dzr[1]=99;
                        if(--dzr[0]<0)dzr[0]=99;
                }
        }
        adj+=180/pr_sr;  //dodawanie dystansu do obliczenia prędkości średniej jazdy, dopasowanie
        adc+=360;  //dodawanie dystansu do obliczenia całkowitej prędkości średniej
}
if(apmx>apr)apmx=apr;  //jeśli prędkość chwilowa większa od maksymalnej, to zaktualizuj maksymalną
if(sleep>200)  //jeśli sleep większe od 200, to znaczy, że procesor się obudził
{
        lcd_Command(LCDC_DC|LCDC_DC_NORMAL);  //włącz wyświetlacz do normalnej pracy
        lcd_Command(LCDC_FS|LCDC_FS_BASIC|LCDC_FS_HORADDR);  //ustaw potrzebne parametry
}
sleep=0;  //wyzeruj zmienną uśpienia

}
```

```
if(zmtemp==4)init1w();  //wykonywane raz na 8 przebiegów pętli, rozpocznij pomiar temperatury
static uint16_t temper;  //zmienna temperatury
static uint8_t znak_temp;  //zmienna znaku temperatury (dodatnia czy ujemna)
if(zmtemp++>7) //wykonywane raz na 8 przebiegów pętli, czyli co 1,6 sekundy
{
znak_temp=0;  //skasuj znak
temper=temperatura(); //odczytaj temperaturę
zmtemp=0; //wyzeruj licznik pętli
if(temper & 0xfe00)  //jeśli temperatura ujemna
{
        temper=~temper;  //przekonwertuj na postać binarną liczbę z uzupełnieniem do 2
        ++temper;
        znak_temp=1;  //ustaw znak minus
}
}
```

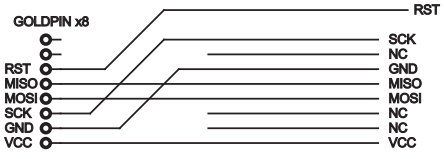
Listing 7

nia to solidnego połączenia przy dużych wstrząsach, ale od tego jest wyżej wspomniany kondensator C5. Mamy gotową płytkę, ale jak to teraz zaprogramować? Nieprzypadkowo na złączu wyświetlacza wyprowadzony jest cały interfejs SPI. Do programatora wystarczy podpiąć prosty adapter – wstążkę przewodów. Jego schemat znajduje się na **rysunku 4**. Przewód resetu lutujemy do specjalnie wyprowadzonego pola miedzi na płytce drukowanej. Resztę kabelków przykładamy do odpowiednich goldpinów w złączu i ładujemy program do mikrokontrolera. Na koniec trzeba polakierować płytkę (oprócz złącza do wyświetlacza!). Zabezpieczy to przed nadmiernym poborem prądu przy dużej wilgotności lub gwałtownej zmianie temperatury.



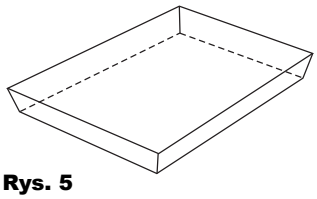
Rys. 3

Rys. 4



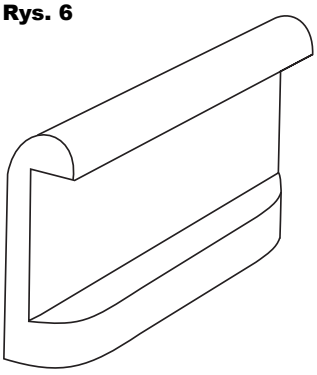
Obudowa

Czas na najtrudniejsze zadanie – zrobienie obudowy. Najlepszym rozwiązaniem jest zastosowanie fabrycznej obudowy z zepsutego licznika. Jednak nie może to być obudowa z byle jakiego taniego urządzenia. Potrzebna jest mocna i szczelna z miejscem na duży wyświetlacz. Takie mają tylko drogie liczniki. Problem w tym, że one się nie psują, a nowego przecież nie będziemy niszczyć. Pozostaje nam tylko wybór spośród uniwersalnych obudów. Niestety i tu nie ma łatwo. Ja wybrałem Z-24A i dalszy opis będzie dotyczył jej przeróbki. Najlepiej zaopatrzyć się od razu w dwie takie obudowy. Jedna przyda się do ćwiczeń i eksperymentów. No dobrze, co z nią nie tak? Przede wszystkim jest za gruba. Skracamy obie części, czołową na wysokość ok. 7mm, a tylną na ok. 6mm. W tym celu można wykorzystać np. nożyk do tapet lub gruboziarnisty papier ścierny. Na koniec wygładzamy drobnoziarnistym papierem i delikatnie zaokrąglamy nim rogi ścianek. Jednak zanim to zrobimy, wycinamy tulejki na śrubki, przydadzą się dłuższe. W następnej kolejności wycinamy otwór na szybkę lub pleksi. Najlepiej gdy jego krawędzie nie będą prostopadłe do płaszczyzny obudowy, tylko nachylone pod pewnym kątem. Również szybka powinna być wycięta na kształt trapezu, według **rysunku 5**. Zapobiega to jej wpadnięciu do środka, przy mocnym ściśnięciu obudowy. Zastanówmy się teraz, co wybrać, szkło czy pleksi? Obydwa rozwiązania mają swoje zalety i wady. Szybka jest mniej odporna na silne wstrząsy i upadki, a pleksi można łatwo porysować. W moim modelu jest szybka. Na razie jej nie przyklejamy, zrobimy to na końcu. Kolejny otwór robimy pod klawiaturę. Rozmieszczenie przycisków na płytce drukowanej odpowiada ostatniemu rzędowi przycisków z klawiatury do Nokii 3410. Wycięcie w obudowie nieregularnych kształtów na te klawisze nie jest proste. Warto poćwiczyć najpierw na drugiej obudowie. Najpierw odrysowujemy na niej kształt przycisków, potem wycinamy np. wiertarką, a na koniec wyrównujemy papierem ściernym. Jednocześnie sprawdzamy czy klawiatura mieści się w otworze, ewentualnie gdzie jeszcze trzeba doszlifować. Gumka spod tych klawiszy będzie dobrą uszczelką. Gdy już zrobimy otwór, przyklejamy ją od środka obudowy dużą ilością butaprenu. Na początku trochę się odkształci, ale po wyschnięciu będzie już w porządku. Nie trzeba co do milimetra dopasowywać miejsca klawiatury, otwór powinien znajdować się mniej więcej w środku pomiędzy wyświetlaczem a końcem obudowy. Teraz trzeba wyżłobić rowek na gumkę uszczelniającą dwie części obudowy. Wykorzystałem do tego lutownicę z grotem o temperaturze 200°C. Żeby było równo, musimy wypalać rowek od linijki (metalowej) lub innego równego przedmiotu np. radiatora. Celowo napisałem radiatora, gdyż dobrze odprowadza ciepło i nie pozwala na zniekształcenie zewnętrznych części obudowy. Zagłębienia robimy na obydwóch jej częściach.



Rys. 5

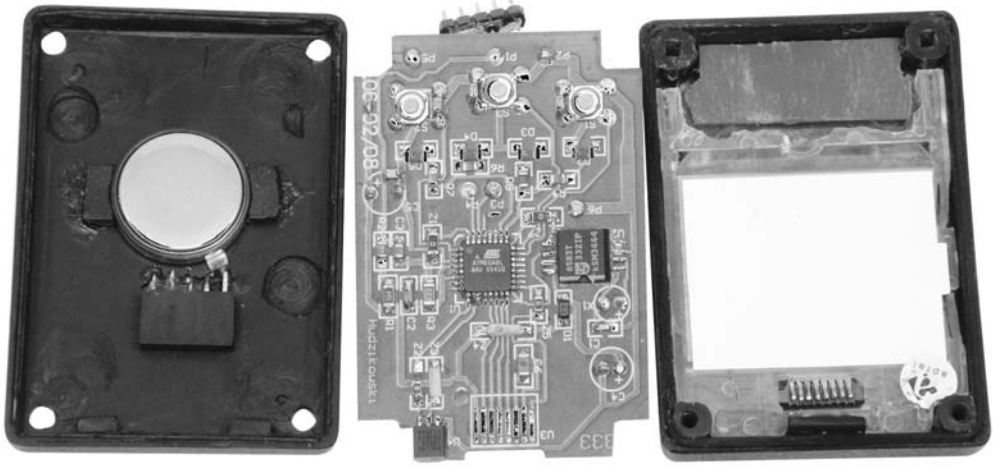
Rys. 6

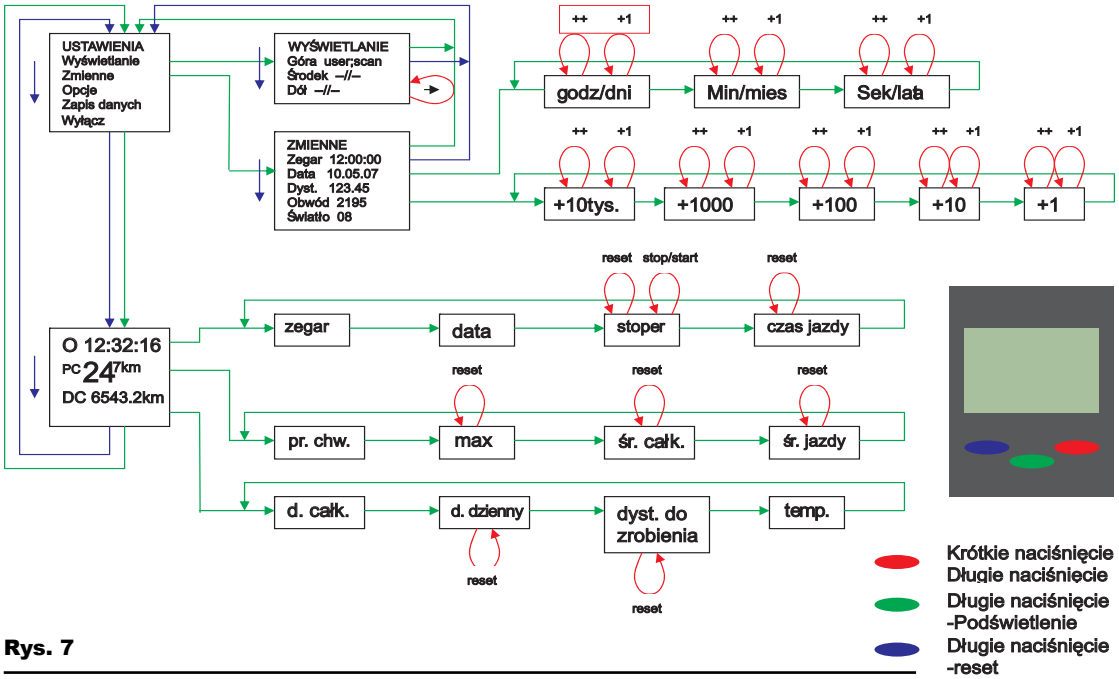


Kolejną niedogodnością są tulejki na śrubki. Przy takim rozstawie jest mało miejsca na płytkę. Nie przyklejamy więc ich w to samo miejsce, ale w samym rogu. Dodatkowo trochę ścinamy tak, aby powstały dwie ścianki pod kątem prostym, co pozwoli mocniej przykleić tulejki do brzegów obudowy. Nie musimy stosować żadnego droższego kleju, bardzo dobrze sprawdza się Butapren-A. Trzeba tylko poczekać jedną dobę na całkowite wyschnięcie. Tulejki z części czołowej powinny wystawać około 1,5mm ponad wysokość ścianek. Będzie tam zamocowana gumka uszczelniająca. W tylnej części obudowy tulejki są zbędne. Należy tylko pamiętać o prawidłowym miejscu wiercenia dziur na śrubki. Płytka drukowana powinna się trzymać obudowy i dobrze dociskać złącze wyświetlacza. Musimy więc ją jakoś przymocować. Na śrubki nie ma miejsca, więc wykonujemy element według **rysunku 6. P**rzyklejamy go na bocznej ściance u góry obudowy. Takie mocowanie wystarczy. Dobrze, gdyby płytka wchodziła trochę ciasno do obudowy i przy wymianie baterii nie wypadła. Dodatkowo w tylnej części obudowy przyklejamy dwa kawałki tworzywa, które będą trzymać płytkę podczas naciskania przycisków. Te elementy wykonujemy np. z kawałka wyciętej obudowy pod szybkę. W tylnej części znajdować się będzie złącze z interfejsem RS-232 i wejściem z kontaktronu. Najlepiej sprawdza się gniazdo goldpinów. Jednak jest ono stosunkowo grube. Robimy więc wgłębienie w obudowie, tak aby gniazdo było nachylone lekko pod skosem i wystawało tylko do połowy. Ostre brzegi można oszlifować. Samo złącze jest za słabe, żeby utrzymać licznik. W tym celu do górnej części tylnej obudowy przyklejamy małe kliny. Podstawkę widoczną na fotografii tytułowej wziąłem z zepsutego licznika. Ta część może pochodzić nawet z najtańszego urządzenia. Z reguły są one dość mocne. W podstawce dobrze jest wyprowadzić złącze pod RS-232, jeśli chcemy dołączać podczas jazdy dodatkowe moduły do naszego licznika.

Obsługa

Dużo funkcji i rozbudowany panel sterowania powoduje, że na początku można się pogubić. Pomocą będzie graf przejść pokazany na **rysunku 7**. Obsługa jest intuicyjna. Wystarczy przyjąć takie zasady:





Rys. 7

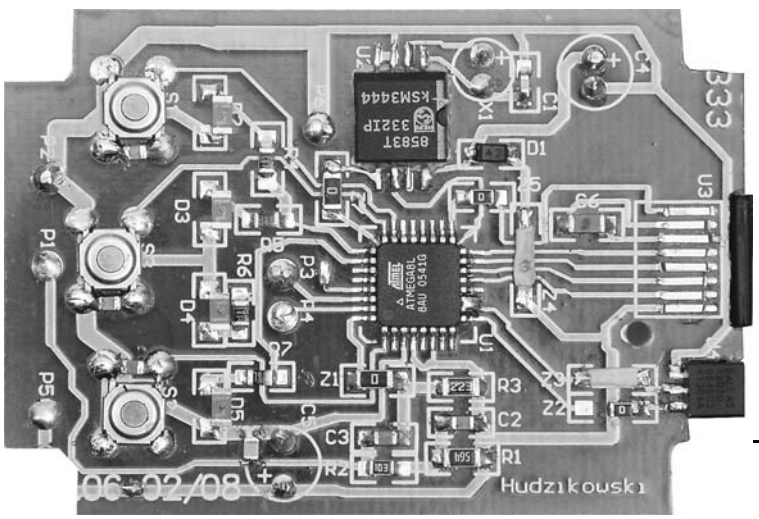
S1 – przycisk nawigacyjny, przemieszczanie się w dół (kod – pr1=1),
S2 – przycisk nawigacyjny, poruszanie się w prawo (kod – pr1=2),
S1 i S2 razem – wyjście z podprogramu (kod – pr1=3),
S3 – ustawianie/kasowanie danych i parametrów, dla stopera start, stop i reset (kod – pr2=1/2),
S1 długie naciśnięcie – reset procesora (kod – pr1=4),
S2 długie naciśnięcie – włączenie podświetlenia (kod – pr1=5).

Obok znajdują się kody klawiszy zwracane w programie. Po włączeniu układ znajduje się w podprogramie USTAWIENIA. Na początku wchodzimy na zakładkę ZMIENNE i ustawiamy kolejno czas, datę, obwód koła i czas pracy podświetlenia. Dystans do zrobienia nie będzie nam na razie potrzebny. Jak widać, wszystkie zmienne ustawiane są parami. Takie założenie bardzo uprościło procedury ustawień w programie. Wyjaśnienia wymaga czas pracy podświetlenia. Nie są to sekundy, tylko 1/5 sekundy. Tak więc wartość np. 20 oznacza, że podświetlenie będzie włączone na czas 4 sekund. Jeśli wprowadzimy wartość powyżej 99 to włączone będzie stale. Po ustawieniu tych wartości licznik jest gotowy do pracy. Naciskamy więc razem S1 i S2, potem jeszcze raz S1 i S2, aby wyjść z panelu ustawień. Jak chcemy do niego powrócić, też naciskamy te dwa przyciski.

Możliwości zmian

Zmieniać można wszystko, od programu po płytkę drukowaną i obudowę. Skupmy się teraz na programie. Zmiana funkcji licznika jest stosunkowo prosta. Pewną trudnością jest to, że nie są one spójne, ale ich fragmenty znajdują się w różnych miejscach programu. Zależy to od tego, jakie działanie ma dana funkcja i z jakich danych korzysta. Najłatwiejszą zmianą dającą wyraźne efekty jest inne ustawienie prescalera timera0 z 1024 na 256. Zwiększa się szybkość reakcji na naciśnięcie przycisku oraz wyświetlania. Najlepiej jest to widoczne dla funkcji stopera. Również zmniejsza się czas, po którym zwracana jest wartość długiego naciśnięcia, ale to też można dostosować do swoich potrzeb. Wystarczy wpisać większe stałe, do których porównywana jest zmienna *ln*. Te dwa miejsca znajdują się w procedurze obsługi przycisków, czyli przerwaniu Timera0. Wadą takich zmian jest aż dwukrotny wzrost poboru prądu. Zastanówmy się, co lepiej wybrać: większy komfort czy dłuższą pracę baterii. Na płytce jest jedno wolne pole lutownicze od portu PD.5. W obudowie zmieści się jeszcze jakiś malutki układzik np. odbiornik. W rowerze może być wtedy zainstalowany alarm z nadajnikiem. Gdy włączy się alarm, na PD.5 zostanie podany odpowiedni sygnał i będziemy o tym wiedzieli. Wtedy też przyda się jakiś sygnał dźwiękowy. To tylko przykładowe możliwość wykorzystania tego wolnego wyprowadzenia. Drugim ciekawym rozwiązaniem jest podłączyć tam płytkę piezo i dodać funkcję budzika do programu lub sygnalizatora dla innych funkcji. Zanim jednak zaczniemy dokładanie kodu do programu, musimy zoptymalizować istniejący. Na pewno da się go jeszcze porządnie zmniejszyć. Zajmijmy się teraz płytką, ją też można nawet dwukrotnie zmniejszyć, gdyby udało się zdobyć obudowę z fabrycznego licznika. Płytka z rysunku 3 przystosowana jest pod obudowę Z-24 i dlatego jest taka duża. Mimo wszystko obudowa wcale nie jest taka wielka, niektóre liczniki są nawet jeszcze większe, ale większy problem stanowi chyba jednak estetyka. No ale co tu poradzić, jak tylko takie prostokątne obudowy produkują... Po jakimś czasie można się do tego przyzwyczaić. Na koniec, gdy już będziemy mieli gotowy licznik, warto go jeszcze przetestować, sprawdzić, jak sobie radzi przy dużych wstrząsach i czy jest wodoodporny. W moim modelu były trudności z uszczelnieniem szybki, butapren widocznie nie radzi sobie ze szkłem, ale przeciekał tylko przy mocnym strumieniu z prysznica. W praktyce oznacza to, że przy niebyt silnym deszczu nie ma się czym martwić. Gdyby pojawiły się jakieś problemy czy trudności podczas przeróbki programu można pisać na adres e-mailowy.

Arkadiusz Hudzikowski
a-r-o@o2.pl



Wykaz elementów

Rezystory
R1 ..... 1MΩ
R2 ..... 10kΩ
R3 ..... 22kΩ
R4-R7 ..... 180Ω

Kondensatory
C1 ..... 33pF
C2 ..... 47nF
C3 ..... 10nF
C4,C5 ..... 100μF
C6,C8-C10 ..... 1μF
C7 ..... 22nF

Półprzewodniki
D1 ..... prostownicza SMD
D2-D5 ..... LED SMD
U1 ..... Atmega8 TQFP
U2 ..... PCF8583 SMD
U3 ..... LCD Nokia3310
U4 ..... DS18B20

Pozostałe
P1-P6 ..... goldpin*
S1-S3 ..... microswitch SMD
X1 ..... rezonator kwarcowy 32768Hz
Listwa żeńskich i męskich goldpinów*
*patrz opis

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2870.