Klocki dla Arduino (1)

Procedury obsługi wybranych modułów dodatkowych dla Arduino – impulsator

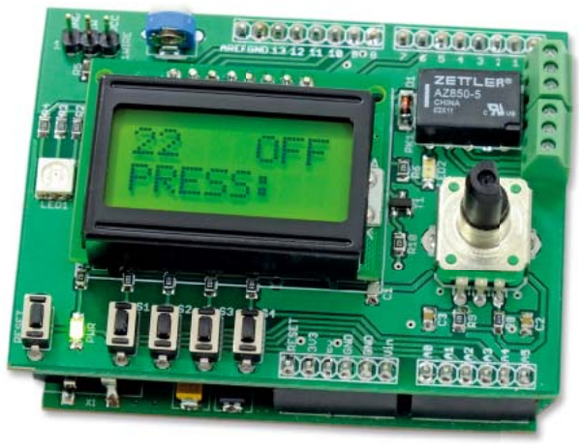
Arduino jest nie tylko świetną platformą prototypową. Ogromy wybór modułów dodatkowych umożliwia budowanie funkcjonalnych urządzeń. Unika się przy tym konieczności zaprojektowania płytki drukowanej i jej montażu. W artykule prezentujemy kilka „klocków", z których można poskładać funkcjonalne urządzenia oraz metody ich obsługi programowej.

Wakacje są świetną okazją, aby zapoznać się ze środowiskiem programistycznym Arduino. „Klocki" oferowane przez wielu niezależnych producentów oraz dostępne w Internecie, pozwalają na szybkie wykonanie zegara, woltomierza, układu sterującego i wielu, wielu innych. Przyjrzyjmy się przykładom programów wykonanych dla Arduino Duemilanove oraz modułów opisywanych w Elektronice Praktycznej.

Menu obsługiwane za pomocą enkodera

Wiele nowoczesnych urządzeń ma menu wyświetlane na ekranie LCD i obsługiwane za pomocą enkodera obrotowego o małej rozdzielczości, czasami nazywanego impulsatorem. Z takiego enkodera są wyprowadzone dwa przebiegi przesunięte w fazie o 90 stopni oraz zestyki przycisku zintegrowanego z osią. Oprócz płytki bazowej, w tym przykładzie użyjemy modułu AVT-1722 mającego zamontowany wyświetlacz LCD 2×8 znaków, 4 przyciski, przycisk zerowania, przekaźnik oraz impulsator.

W związku z niewielką liczbą znaków mieszczącą się na wyświetlaczu przyjmijmy, że pojedyncza pozycja menu będzie wyświetlana na całym ekranie. Obrót enkodera w prawo lub w lewo będzie powodował przejście do kolejnej (w prawo) lub poprzedniej (w lewo) pozycji menu. Wybór będzie zatwierdzany za pomocą naciśnięcia osi enkodera. Po jego dokonaniu będzie wyświetlany następny poziom menu lub wywoływana funkcja urządzenia. Wykonanie menu rozpoczniemy od funkcji obsługi enkodera.



Na płytce AVT-1722 enkoder jest dołączony do wyprowadzeń PD0 (przycisk), PD1 i PD2 (wyprowadzenia A i B). Prostą funkcję służącą do jego obsługi pokazano na listingu 1. Ze względu na skromne zasoby pamięci mikrokontrolera, funkcja używa zmiennych globalnych *w_lewo*, *w_prawo*, *imp_przycisk* do informowania o stanie enkodera. Dodatkowo jest używana zmienna *poprzedni*, ponieważ status enkodera jest sprawdzany tylko wtedy, gdy poprzedni stan enkodera różni się od bieżącego. Funkcja działa w ten sposób, że jeśli wykryto obrót w prawo, to zmienna *w_prawo* przyjmuje wartość logiczną TRUE (1), natomiast *w_lewo* wartość logiczną FALSE (0). W przeciwnym kierunku obrotów zmienne mają przeciwne wartości. Jeśli wykryto naciśnięcie osi, to zmienna *imp_przycisk* przyjmuje wartość logiczną TRUE. W innych przypadkach wszystkim zmiennym są nadawane wartości FALSE. Zasada działania funkcji jest bardzo prosta – jeśli zmienił się poziom na wyprowadzeniu „A" enkodera, to jest sprawdzany poziom na wyprowadzeniu „B". Jeśli na wyprowadzeniu „B" jest poziom niski, to zakładam, że enkoder obraca się w lewo. Jeśli wysoki, to w prawo. Funkcja pomimo pewnych „skrótów myślowych" działa poprawnie i świetnie radzi sobie z obsługą enkodera zamocowanego na płytce. Uzbrojeni w funkcję obsługi enkodera możemy wziąć się za menu.

W tym przykładzie programu menu jest jednopoziomowe. To znaczy, że wyboru funkcji dokonuje się na najwyższym poziomie i nie ma żadnych podmenu. Za pomocą tego menu wybiera się kolor świecenia wielokolorowej diody LED zamontowanej obok wyświetlacza. W tym przykładzie dioda może świecić w kolorach: czerwonym, zielonym, niebieskim i białym lub zostać wyłączona.

Głównymi składnikami menu są dwie funkcje: jedna wyświetlająca menu i druga, wywołująca akcję po naciśnięciu osi. Oczywiście, można to zrobić inaczej, jednak przyjąłem taką koncepcję, aby zyskać na czytelności programu. Mało tego, taki rozdział funkcji powoduje, że łatwo przerobić sposób, w jaki jest wyświetlane menu, wykonać np. przewijaną listę itp.

Za wyświetlanie menu jest odpowiedzialna funkcja *menu_glowne(int poz)*. Parametr *poz* informuje o tym, która pozycja jest aktywna. W tym przykładzie jest wyświetlany odpowiedni ekran, ale można również na bazie argumentu np. ustawiać znacznik na liście opcji. Funkcja jest nieskomplikowana i działa na zasadzie rozpatrywania argumentu *numer_linii* w klauzuli *switch – case*. Należy jedynie zwrócić uwagę na fakt, że pozycje menu są numerowane od 0! Dlatego, jeśli chcemy skorzystać z tego przykładu i wykonać menu mające np. 3 pozycje, to funkcja będzie rozpatrywała argument *poz* w zakresie 0…2, natomiast stałej *liczba_ekranow_menu* zdefiniowanej na początku programu przykładowego, należy nadać wartość 2.

Za podejmowanie odpowiedniej akcji odpowiada funkcja *akcja(int numer_linii)*. Działa ona identycznie jak funkcja wyświetlająca menu, jednak zajmuje się wywołaniem odpowiedniej akcji po wciśnięciu osi enkodera. Co ważne, przed wywołanie tej funkcji należy poczekać na zwolnienie osi (funkcja *czekaj_na_przycisk()* oczekująca aż wyprowadzenie dołączone do osi będzie miało poziom wysoki) po to, aby uniknąć wielokrotnego zadziałania wybranej opcji menu.

O numerze pozycji w menu informuje zmienna *y_markera*. Pierwotnie służyła ona do wyznaczania pozycji trójkąta wskazującego pozycję na liście i stąd wzięła się jej nazwa. W tym przykładzie służy ona do infor-

mowania o numerze aktywnego ekranu. Zmienna jest zwiększana przy obrocie w prawo i zmniejsza na przy obrocie w lewo. W wypadku przekroczenia wartości maksymalnej jest zerowana, natomiast po przekroczeniu 0 jest jej nadawana wartość maksymalna. Odczyt impulsatora i manipulacja zmienną *y_markera* są wykonywane w pętli głównej programu *void loop()*.

W zasadzie, za wyjątkiem obsługi enkodera, przykład nie wymaga obszernego komentarza. Kompletny kod źródłowy przykładu przygotowany dla Arduino Duemilanove jest zamieszczony w materiałach dodatkowych.

Listing 1. Funkcja obsługi enkodera

```
void odczyt_impulsatora()
{
  boolean imp_pin1, imp_pin2, biezacy;
  unsigned int i = 0;
//odczyt stanu przycisku enkodera
  imp_przycisk = w_prawo = w_lewo = LOW;
  imp_pin1 = digitalRead(pd0);
  if(imp_pin1 == LOW)
  {
    delay(50);
    imp_pin1 = digitalRead(pd0);
    if(imp_pin1 == LOW) imp_przycisk = HIGH;
    else imp_przycisk = LOW;
  }
//odczyt kierunku obrotu
  imp_pin1 = digitalRead(pd1);
  if(imp_pin1 == LOW) biezacy = HIGH; else biezacy = LOW;
  if(poprzedni != biezacy)
  {
    poprzedni = biezacy;
    if(imp_pin1 == LOW)
    {
      imp_pin2 = digitalRead(pd2);
      if(imp_pin2 == LOW)
      {
        w_lewo = HIGH;
        w_prawo = LOW;
      }
      else
      {
        w_lewo = LOW;
        w_prawo = HIGH;
      }
    }
  }
}
```

Generator z menu obsługiwanym enkoderem

Opisaną wcześniej funkcję obsługi enkodera zastosujemy w przykładowym generatorze. Generator będzie wykonywał zaprogramowaną liczbę cykli polegających na zwarciu przekaźnika na ustawiony czas, co ustawiany odstęp czasu. Ustawione parametry generatora zapamiętamy w pamięci nieulotnej i odtworzymy po włączeniu zasilania.

Przykład programu o takiej funkcjonalności zapamiętano w pliku *Generator_z_przekaznikiem.ino*. Jego parametry graniczne są następujące:

- liczba impulsów od 1 do 10000,
- czas trwania impulsu od 100 ms do 10 s,
- odstęp pomiędzy impulsami od 100 ms do 10 s.

Maksymalne i minimalne wartości nastaw są zapisane na początku programu, po dyrektywie *#define*. Można je zmienić w przemyślany sposób.

Po załączeniu generatora dioda LED świeci się w kolorze czerwonym, a na ekranie pojawia się możliwość wyboru opcji „URUCHOM GENER.” (przez naciśnięcie osi enkodera), po której to jest wywoływana funkcja generująca impulsy. Po przekręceniu osi enkodera w prawo lub w lewo na ekranie pojawia się opcja „PARAMETRY” umożliwiająca ustawienie parametrów generatora. Pomiędzy poszczególnymi pozycjami przechodzi się obracając oś enkodera, natomiast zmiana wartości jest możliwa po naciśnięciu osi. Wówczas kolor świecenia diody zmienia się na niebieski, a parametr można zmieniać kręcąc w prawo lub w lewo. Czasy zmieniają się z rozdzielczością 10 ms, natomiast liczba impulsów z dokładnością do pojedynczego impulsu. Tymczasowe zatwierdzenie nastawy następuje po naciśnięciu osi (zmiana koloru świecenia diody na czerwony), natomiast zapamiętanie wszystkich nastaw wymaga wyboru „OK” z menu. Wybranie „Anuluj” powoduje przywrócenie starych wartości. Po wybraniu „URUCHOM GENER.” kolor świecenia diody LED zmienia się na zielony i następuje uruchomienie funkcji generującej impulsy. Na ekranie LCD jest wyświetlana liczba cykli pozostałych do zatrzymania się. Słychać przy tym przełączające się styki przekaźnika oraz widać cykliczne zaświecanie się i gaszenie diody LED2. Po zakończeniu generowania impulsów styki są ustawiane w pozycji spoczynkowej, dioda LED2 gaśnie, a LED1 zmienia kolor świecenia na czerwony. Przyciski zamontowane pod wyświetlaczem nie są używane. Do zatrzymania generatora może posłużyć przycisk *Reset* zamontowany jako pierwszy od lewej (najbliżej rogu płytki).

Podsumowanie

Program źródłowy jest przykładem zastosowania AVT1722 i AVT5272 i może być dowolnie wykorzystywany i modyfikowany na zasadach licencji GPL. Analizując jego kod można znaleźć odpowiedzi na pytania odnośnie do sposobu odmierzania czasu, sterowania przekaźnikiem czy diodą LED.

Jacek Bogusz, EP