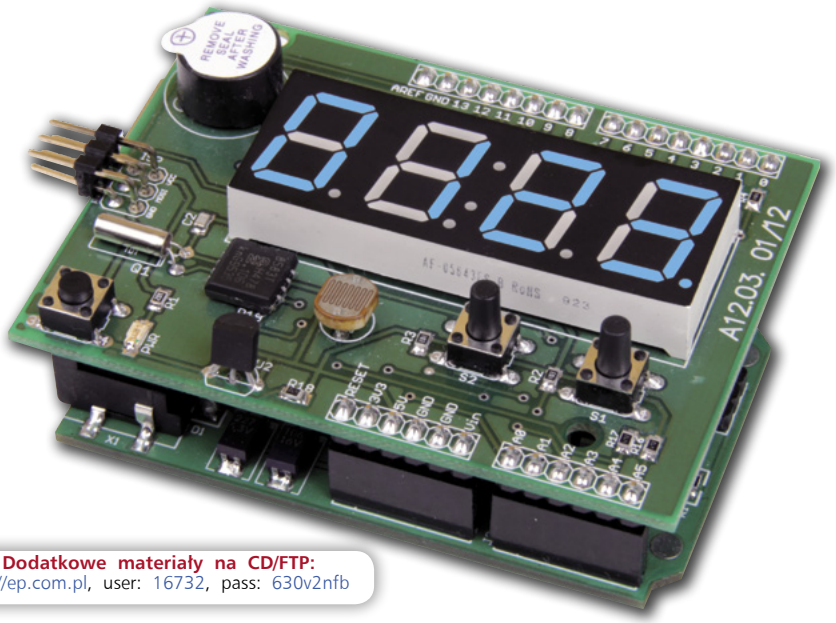
Kurs Arduino (4)
Obsługa modułu LED

W „Elektronice Praktycznej” 5/2011 opisaliśmy moduł LED dla popularnego zestawu Arduino. W tym artykule pokażemy sposoby wyświetlania znaków oraz obsługi elementów, z których składa się ten moduł: dwóch przycisków, zegara RTC PCF8583, generatora piezo, czujnika temperatury DS18B20 oraz fotorezystorowego czujnika światła.



Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 16732, pass: 630v2nfb

Środowisko programistyczne Arduino zawiera gotowe funkcje obsługi zegara PCF8583 oraz termometru DS18B20, co po zaimplementowaniu obsługi wyświetlacza LED pozwala na szybkie, samodzielne zbudowanie funkcjonalnego zegara z termometrem lub obu tych przyrządów niezależnie.

Na **listingu 1** pokazano przykładowy program testowy dla modułu AVTduino LED. Po krótkiej demonstracji tj. wyświetleniu znaków „0”–„9” jest wyświetlany czas odczytany z zegara RTC i migocze kropka dziesiętna, sygnalizując w ten sposób pracę zegara. Po naciśnięciu przycisku S1 jest wyświetlana temperatura otoczenia zmierzona za pomocą czujnika DS18B20 z interfejsem 1-Wire. Po naciśnięciu przycisku S2 jest wyświetlana data odczytana z wewnętrznego kalendarza układu RTC. Po zaciemnieniu fotorezystora jest włączany generator piezo. Nie jest to być może użyteczna funkcja, ale służy ona jedynie do demonstracji sposobu odczytu napięcia z czujnika światła.

Do obsługi programowej układów i komunikacji za pomocą I²C (zegar RTC) oraz 1-Wire (termometr) zastosowano funkcje dostępne w bibliotekach. Środowisko widzi je po skopiowaniu do podkatalogu *Library* oprogramowania Arduino IDE. Są to OneWire i PCF8583 (dostępne na stronie domowej Arduino oraz w materiałach dodatkowych). Do obsługi wyświetlacza modułu AVTduino LED brak gotowej biblioteki – tę należy wykonać samodzielnie lub wykorzystać opisane dalej rozwiązanie.

Wyświetlacz jest multipleksowany. Jego obsługa odbywa się w przerwaniu generowanym przez *Timer1* co 100 µs. Do generowania samych przerwań zastosowano bibliotekę *Timerone* (dostępna w materiałach dodatkowych).

Na początku programu głównego instrukcja *OneWire* ustala linię komunikacyjną

Listing 1. Przykład programu obsługującego moduł AVTduino LED

```
#include „TimerOne.h”  //biblioteka obslugi timera
#include <Wire.h>  //biblioteka i2c
#include <PCF8583.h>  //biblioteka RTC
#include <OneWire.h>  //biblioteka 1Wire

OneWire  ds(A3); // konfigurowanie 1Wire
byte present = 0;
byte data[12];
byte addr[8];
int wart_analog;
int HighByte, LowByte, SignBit, temp, Fract, TReading, Tc_100;
PCF8583 p (0xA0);  //konfigurowanie RTC
const int Buzzer =  13;  //aliasy
const int DP =  12;
const int SW1 =  A1;
const int SW2 =  A2;
const int sens_sw =  A0;
int groundPins[8] = {0, 1, 2, 3, 4, 5, 6, 7};  //wiersze LED
int digitPins[4] = { 8, 9, 10, 11}; //kolumny LED
int wys=0;
int digit[4]; //tablica znakow dla wyswietlacza LED
int kr[4];   //tablica przechowujaca znak kropki
byte temp_sec;
byte wsk_sek=1;
int number[13][7] = { //tablica znakow dla LED
  {0,0,0,0,0,0,1}, //zero
  {1,0,0,1,1,1,1}, //jeden
  {0,0,1,0,0,1,0}, //dwa
  {0,0,0,0,1,1,0}, //trzy
  {1,0,0,1,1,0,0}, //cztery
  {0,1,0,0,1,0,0}, //piec
  {0,1,0,0,0,0,0}, //szesc
  {0,0,0,1,1,1,1}, //siedem
  {0,0,0,0,0,0,0}, //osiem
  {0,0,0,0,1,0,0}, //dziewiec
  {1,1,1,1,1,1,1}, //wylaczenie LED
  {0,0,1,1,1,0,0}, //znak stopnia
  {0,1,1,0,0,0,1}  //znak C
};

void setup()  //procedura konfigurujaca
{
  Timer1.initialize(100);  //inicjalizacja timera 1
  Timer1.attachInterrupt(int_wys);  //uruchomienie przerwania
  for(int i=0; i < 8; i++)  //wyjscia obslugi wierszy LED
  {
    pinMode(groundPins[i], OUTPUT);
    digitalWrite(groundPins[i], HIGH);
  }
  pinMode(Buzzer, OUTPUT);  //konfigurowanie linii piezo
  digitalWrite(Buzzer, HIGH);
  pinMode(DP, OUTPUT);  //konfigurowanie dwukropka
  digitalWrite(DP, HIGH);
```

Listing 1. Przykład programu obsługującego moduł AVTduino LED

```
  for(int i=0; i < 4; i++)  //konfigurowanie kolumn LED
  {
    pinMode(digitPins[i], OUTPUT);
    digitalWrite(digitPins[i], HIGH);
  }
  p.hour = 14;  //inicjalizacja RTC
  p.minute = 30;
  p.second = 0;
  p.year = 2011;
  p.month = 4;
  p.day = 1;
  p.set_time();
  pinMode(SW1, INPUT);  //linie przyciskow
  pinMode(SW2, INPUT);
  digitalWrite(SW1, HIGH);  //zalaczenie pull-up
  digitalWrite(SW2, HIGH);
  analogReference(DEFAULT);   //nastawy A/C

}

//funkcja przerwania, w ktorej jest obslugiwany LED
void int_wys() {
  for(int i=0; i < 4; i++)
  {
    digitalWrite(digitPins[i], HIGH);  //wylaczenie LED
  }
  for(int g=0; g < 7; g++)  //LED=znak z digit
  {
    digitalWrite(groundPins[g], number[digit[wys]][g]);
  };
  digitalWrite(groundPins[7], !(kr[wys])); //kropka
  digitalWrite(digitPins[wys], LOW);  //wlaczenie wyswietlacza
  wys++;  //zwiekszenie wartosci wys
  if (wys>4) wys=0;  //jesli był obslugiwany 4 wyswietlacz
                     //to przejscie do 1
};

void loop()  //petla glowna programu
{
  digitalWrite(DP, LOW);   //wlaczenie LED
  kr[0]=HIGH;
  kr[1]=HIGH;
  kr[2]=HIGH;
  kr[3]=HIGH;
  for(int k=0; k < 10; k++)   //wyswietlenie „0” do „9”
  {
    digit[0] = k;
    digit[1] = k;
    digit[2] = k;
    digit[3] = k;
    delay(200);          //opoznienie 200ms
  };

  digitalWrite(DP, HIGH);  //wylaczenie LED
  kr[0]=LOW;
  kr[1]=LOW;
  kr[2]=LOW;
  kr[3]=LOW;
  digit[0] = 10;
  digit[1] = 10;
  digit[2] = 10;
  digit[3] = 10;
  delay(2000);
  while(1)      //petla programu
  {
    p.get_time();  //odczyt RTC
    digit[3]=  p.minute%10;  //wyswietlenie minut
    digit[2] = p.minute / 10;
    digit[1]=  p.hour%10;  //wyswietlenie godzin
    digit[0] = p.hour / 10;

    if (temp_sec!=p.second)  //miganie dwukropka 1s
    {
      temp_sec=p.second;  //zapisanie wartosci sekund
//wlaczenie lub wylaczenie dwukropka
      if (wsk_sek==1)
        digitalWrite(DP, HIGH);
      else digitalWrite(DP, LOW);
      wsk_sek = !wsk_sek;  //zmiana stanu zmiennej
    }

    if (digitalRead(SW1) == LOW) {  //Wcisniety S1?
      digit[0] = 10;  //wylaczenie LED
      digit[1] = 10;
      digit[2] = 10;
      digit[3] = 10;
      digitalWrite(DP, HIGH);
      while(digitalRead(SW1) == LOW)
      {
        getTemp();              //odczyt temperatury
        digit[1]=  temp%10;     //wyswietlenie temperatury
        digit[0] = temp / 10;
        digit[2] = 11;          //wyswietlenie °
        digit[3] = 12;          //wyswietlenie C
        delay(500);            //opoznienie 500 ms
      }
    }
    if (digitalRead(SW2) == LOW) {  //czy nacisniety S2?
      digit[0] = 10;  //wylaczenie LED
      digit[1] = 10;
      digit[2] = 10;
      digit[3] = 10;
```

interfejsu 1-Wire, którą w tym przypadku jest linia *A3*. Instrukcja *PCF8583 pcf(0xA0)* z biblioteki *PCF8583* umożliwia zapisanie adresu układu RTC, który ma wartość *0xA0*. Tablica *groundPins[8]* zawiera linie wierszy wyświetlacza, natomiast tablica *digitPins[4]* linie kolumn wyświetlacza LED. Do tablicy *digit[]* będą zapisywane wartości wyświetlane na wyświetlaczu LED w taki sposób, że każdemu wyświetlaczowi odpowiada pojedynczy element tablicy *digit[]*. Tablica *kr[]* jest używana do obsługi kropek wybranego wyświetlacza LED. Zapis do tej tablicy wartości „1” powoduje zapalenie się kropki danego wyświetlacza, a „0” jej zgaszenie. W tablicy *numer[]* zapisano wyświetlane na LED znaki, czyli cyfry od „0” do „9”. Zapisanie liczby „10” powoduje wyłączenie wyświetlacza, „11” wyświetlenie znaku stopnia, natomiast „12” powoduje wyświetlenie litery „C”. Tablice znaków można dowolnie rozbudować o własne znaki.

W procedurze *setup* instrukcja *Timer1.initialize(100)* konfiguruje Timer1 w taki sposób, aby zgłaszał przerwanie co 100 µs. Natomiast instrukcja *Timer1.attachInterrupt(int_wys)* konfiguruje nazwę wywoływanej procedury podczas przerwania. Co 100 µs będzie wykonywana procedura *int_wys* obsługująca wyświetlacz LED.

Procedura obsługi przerwania w pierwszej kolejności wyłącza wyświetlacz na danej pozycji, wystawia na port mikrokontrolera wartość odczytaną z tablicy *digit[]*, a następnie załącza ten wyświetlacz. Przy kolejnym wywołaniu przerwania jest obsługiwany kolejny wyświetlacz LED. W procedurze przerwania jest także włączana kropka wyświetlacza w zależności od wartości zapisanych w tablicy *kr[]*.

Na pozycję aktualnie obsługiwanego wyświetlacza wskazuje zawartość zmiennej *wys*. Po obsłudze ostatniego jest ona zerowana i następuje obsługa pierwszego wyświetlacza. Obsługa wyświetlaczy LED jest wykonywana tak szybko, że dla obserwatora będzie widoczne jednoczesne świecenie wszystkich cyfr LED.

Instrukcje *p.hour = 14*, *p.minute = 30*, *p.second = 0*, *p.year = 2011*, *p.month = 4* oraz *p.day = 1* ustawiają domyślne wartości czasu w układzie zegara RTC. Ich zapis do układu RTC powoduje instrukcja *p.set_time()*, natomiast odczyt następuje z użyciem instrukcji *p.get_time()*. Opis komend obsługi zegara RTC można znaleźć w dokumentacji biblioteki *PCF8583*.

W dalszej części procedury nastaw są konfigurowane te linie mikrokontrolera, do których dołączono przyciski. Są one ustalane jako wejściowe z włączonymi rezystorami podciągającymi. W pętli głównej programu *loop* jest włączany za pomocą instrukcji *digitalWrite(DP, LOW)* dwukropek *DP* dołączony do linii 12 systemu Arduino (wartość *LOW*

włącza dwukropek, a *HIGH* wyłącza). Kolejne instrukcje włączają kropki wyświetlaczy. W tym przypadku wpisanie wartości *HIGH* włącza kropkę, a *LOW* ją wyłącza.

W pętli *FOR* co 200 ms zapisywane są do wszystkich wyświetlaczy cyfry od „0" do „9" (wartość zmiennej *k*). Po zakończeniu się pętli *FOR* wyświetlacz jest wyłączany (wyłączane są kropki, znak DP oraz dzięki zapisaniu liczby 10 do tablicy *digit[]* wszystkie wyświetlacze) na 2 s. Następnie w nieskończonej pętli *while* za pomocą instrukcji *p.get_time()* jest pobierana z układu RTC godzina oraz data, które są zapisywane do zmiennych *p.hour*, *p.minute*, *p.second*, *p.year*, *p.month* oraz *p.day*. Za pomocą instrukcji *digit[3]= p.minute%10* oraz *digit[2] = p.minute / 10* do dwóch ostatnich wyświetlaczy są zapisywane wartości minut (dziesiątki oraz jednostki), natomiast za pomocą instrukcji *digit[1]= p.hour%10* oraz *digit[0] = p.hour / 10* do dwóch pierwszych wyświetlaczy zapisywana jest godzina. W warunku *if (temp_sec!=p.second)* jest obsługiwany dwukropek wyświetlacza. Odczytana wartość sekund jest porównywana z poprzednią wartością zapisaną w zmiennej *temp_sec*. Jeśli jest różna, to jest zmieniany stan dwukropka. Zmiana jest uzależniona od stanu zmiennej *wsk_sek*, której wartość zmienia się co 1 s. Jeśli zmienna ma wartość „0", to na „1", jeśli „1", to „0". Tak sterowany dwukropek wyświetlacza będzie migał z częstotliwością 1 Hz.

Za pomocą instrukcji *if (digitalRead(SW1) == LOW)* jest sprawdzany stan przycisku S1. Jeśli został on naciśniety (o czym świadczy poziom niski), wykonywane są instrukcje wyłączające wyświetlacz LED. Następnie instrukcja *while(digitalRead(SW1) == LOW)* sprawdza, czy nadal jest naciśniety przycisk S1. Jeśli tak, to jest wykonywana procedura *getTemp()*, w której następuje odczyt temperatury z czujnika DS18B20 (za pomocą funkcji dostępnych w bibliotece *OneWire*). Jako pierwsze procedura *getTemp()* wykonuje zerowanie magistrali 1-Wire za pomocą instrukcji *ds.reset()*. W dalszej kolejności, zgodnie ze specyfikacją układu DS18B20, są wykonywane instrukcje *s.write(0xCC,1)* oraz *ds.write(0x44,1)* inicjujące pomiar temperatury. Kompletna procedura odczytu temperatury wygląda następująco:

```
ds.reset()
ds.write(0xCC,1);
ds.write(0xBE);
 for ( i = 0; i < 9; i++) {
  data[i] = ds.read();
 }
```

Listing 1. Przykład programu obsługującego moduł AVTduino LED

```
      digitalWrite(DP, HIGH);
      while(digitalRead(SW2) == LOW)       //jesli S2 nacisniety
      {
        digit[3]=  p.day%10;               //wyswietlenie dnia miesiaca
        digit[2] = p.day / 10;
        digit[1]=  p.month%10;             //wyswietlenie miesiaca
        digit[0] = p.month / 10;
        kr[1]=HIGH;                        //wlaczenie kropki 2. LED
        delay(500);                        //opoznienie 500 ms
      }
      kr[1]=LOW;                           //wylaczenie kropki
    }
//wlaczenie lub wylaczenie buzzera
    if ((wart_analog = analogRead(A0))<80)
      digitalWrite(Buzzer, LOW);
    else digitalWrite(Buzzer, HIGH);
  }
}

void getTemp() {                           //odczytu temperatury DS18B20
  int foo, bar, i;                         //definicje zmiennych
  ds.reset();                              //zerowanie 1-Wire
  ds.write(0xCC,1);                        //start pomiaru temperatury
  ds.write(0x44,1);
  present = ds.reset();                    //odczyt temperatury
  ds.write(0xCC,1);
  ds.write(0xBE);
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  LowByte = data[0];                       // obliczenie temperatury
  HighByte = data[1];                      //wynik do temp
  TReading = (HighByte << 8) + LowByte;
  SignBit = TReading & 0x8000;             //obliczenie znaku temperatury
  if (SignBit) {
    TReading = -TReading;
  }
  Tc_100 = (6 * TReading) + TReading / 4;
  temp = Tc_100 / 100;
  Fract = Tc_100 % 100;
  if (Fract > 49) {
    if (SignBit) {
      --temp;
    }
    else {
      ++temp;
    }
  }
}
```

Kolejną instrukcje procedury obliczają odczytana temperaturę w stopniach Celsjusza i zapisują ją do zmiennej *temp*. Dokładny algorytm obsługi układu DS18B20 oraz sposobu obliczenia temperatury na podstawie odczytanych danych można znaleźć w specyfikacji termometru DS18B20.

Odczytana temperatura jest wyświetlana na dwóch pierwszych wyświetlaczach z wykorzystaniem instrukcji *digit[1]= temp%10* oraz *digit[0] = temp / 10*. Na trzecim wyświetlaczu, za pomocą instrukcji *digit[2] = 11*, jest wyświetlany znak stopnia, a na czwartym znak *„C"*. Pomiar temperatury jest wykonywany z opóźnieniem 500 ms wprowadzonym przez komendę *delay(500)*.

Kolejne instrukcje dotyczą obsługi przycisku S2, po naciśnięciu którego na wyświetlaczu zostanie wyświetlona data odczytana z układu zegara RTC. Dodatkowo, na drugim wyświetlaczu za pomocą instrukcji *kr[1]=HIGH* zostanie zaświecona kropka dziesiętna.

Dalsze instrukcje programu dotyczą obsługi rezystorowego czujnika światła, którego rezystancja zmniejsza się wraz ze wzrostem natężenia oświetlenia. Wartość z czujnika światła jest odczytywana przez przetwornik A/C mikrokontrolera. Im większy jest poziom jasności, tym większa wartość napięcia zmierzona przez przetwornik. Z użyciem instrukcji warunkowej *if ((wart_analog = analogRead(A0))<80)* mikrokontroler sprawdza, czy reprezentacja liczbowa napięcia odczytanego z czujnika jest mniejsza od 80. Jeśli tak, to za pomocą instrukcji *digitalWrite(Buzzer, LOW) jest* włączany brzęczyk piezo. Należy zauważyć, że *analogRead()* to funkcja, nie zmienna. Jej parametrem jest numer wejścia analogowego.

Podsumowanie

Dzięki użytecznym komponentom: wyświetlaczowi, kilku przyciskom, zegarowi RTC i czujnikom, będzie to moim zdaniem jeden z najbardziej popularnych i najczęściej używanych modułów. Z jego wykorzystaniem można w łatwy sposób wykonać czytelny wyświetlacz zegara czy termometru z jasności świecenia regulowaną za pomocą fotorezystora.

Marcin Wiązania
marcin.wiazania@ep.com.pl