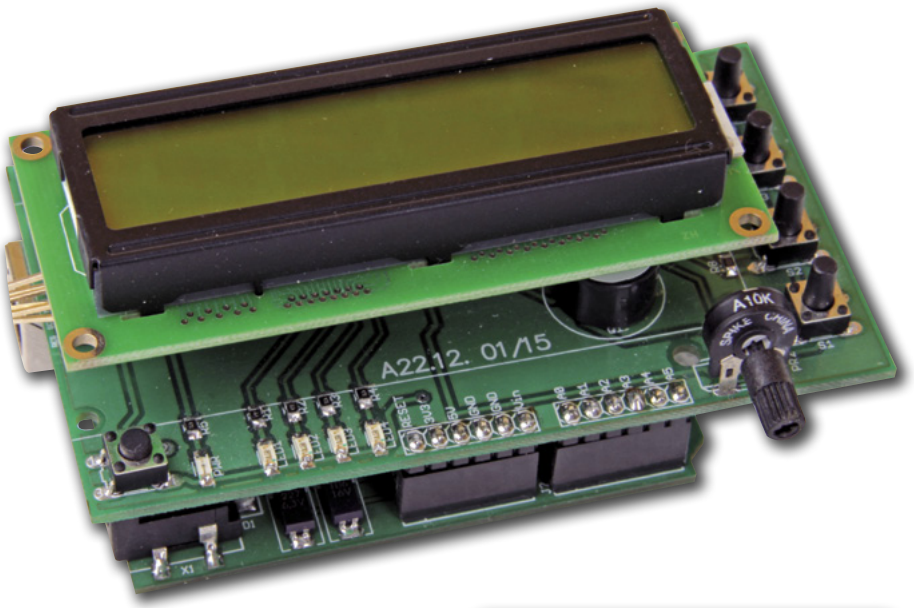
Kurs Arduino (3)

Obsługa modułu LCD

W EP 4/2011 opublikowaliśmy opis modułu LCD z przyciskami (AVT1615) współpracującego z płytką Arduino oraz kompatybilną – AVTduino. W tym artykule pokażemy sposób obsługi programowej tego modułu oraz komponentów interfejsu użytkownika, które on zawiera: przycisków, potencjometru, generatora piezzo, diod LED oraz czujnika temperatury LM35.



Zacznijmy od przykładów praktycznych rozwiązań, których będzie można użyć w samodzielnie opracowywanych aplikacjach. Na **listingu 1** zamieszczono przykładowy program dla modułu AVT1615. Program ten wyświetla komunikat na wyświetlaczu LCD, mierzy napięcie na suwaku potencjometru, odczytuje temperaturę, wyświetla umowny numer naciśniętego przycisku oraz zaświeca umieszczoną koło niego diodę LED. Dodatkowo, naciśnięcie przycisku jest sygnalizowane dźwiękiem brzęczyka piezzo.

W Arduino do obsługi modułów wyświetlaczy z kontrolerem HD44780 jest przeznaczona biblioteka *LiquidCrystal (LCD)* która umożliwia sterowanie modułem wyświetlacza za pomocą interfejsu z 4- lub 8-bitową szyną danych. W **tabeli 1** zamieszczono komendy dostępne w tej bibliotece.

Niektóre z komend wymienionych w tabeli 1 zastosowano w przykładowym programie z listingu 1. Jak można zauważyć, w programie w pierwszej kolejności dołączono bibliotekę obsługi LCD – *LiquidCrystal.h*. Następnie zadeklarowano stałe definiujące wyprowadzenia, do których dołączono diody LED, klawisze i brzęczyk piezzo. Dla lepszej czytelności programu nadano im łatwe do zapamiętania nazwy. Numery wyprowadzeń są zgodnie z opisem na płytkach Arduino Uno i Avtduino LCD. Dalej, za pomocą komendy *LiquidCrystal(rs, enable, d4, d5, d6, d7) zgodnie ze schematem ideowym* przypisano wyprowadzenia, do których został dołączony LCD. W dalszej części programu utworzono zmienne wykorzystywane do obliczenia wartości zmierzonego napięcia z suwaka potencjometru dołączonego do nóżki A0 oraz do obliczenia temperatury zmierzonej przez termometr LM35 dołączony do nóżki A1. Dzięki 8-bajtowej tablicy *byte st[8]={…}* zdefiniowano symbol stopnia, który jest używany jako jednostka temperatury.

W funkcji *setup()* za pomocą komendy *lcd.begin(16, 2)* zdefiniowano rozdzielczość zastosowanego wyświetlacza LCD. Pierwszy parametr określa liczbę znaków w wierszu (kolumn) a drugi liczbę wierszy. Jak łatwo domyślić się, w przykładzie zastosowano wyświetlacz 2×16 znaków. Do utworzenia znaku stopnia służy komenda *lcd.createChar(0, st), której argumentami są numer znaku (kod) oraz bajty definicji (w naszym wypadku jest to tablica st[])*. Komenda *analogReference(DEFAULT)* ustala napięcie odniesienia dla wewnętrznego przetwornika A/C mikrokontrolera na napięcie zasilające (w tym wypadku 5 V). Przetwornik mikrokontrolera będzie mierzył sygnały analogowe

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 10925, pass: 87thc181
- poprzednie części kursu

Tabela 1. Komendy obsługi wyświetlacza LCD

LiquidCrystal()	Definiuje piny do których został dołączony LCD
begin()	Definiuje rozdzielczość zastosowanego LCD
clear()	Czyści ekran LCD
home()	Ustawia kursor na początku ekranu LCD
setCursor()	Ustawia kursor w zadanym miejscu LCD
write()	Zapisuje znak do LCD
print()	Zapisuje znak lub znaki do LCD
cursor()	Włącza kursor
noCursor()	Wyłącza kursor
blink()	Włącza miganie kursora
noBlink()	Wyłącza miganie kursor
display()	Włącza ekran LCD
noDisplay()	Wyłącza ekran LCD
scrollDisplayLeft()	Przesuwa zawartość LCD w lewo
scrollDisplayRight()	Przesuwa zawartość LCD w prawo
autoscroll()	Automatyczne przesuwanie zawartości na LCD
noAutoscroll()	Wyłączenie automatycznego przesuwania zawartości na LCD
leftToRight()	Ustawia kierunek zapisu tekstu od prawej do lewej
rightToLeft()	Ustawia kierunek zapisu tekstu od lewej do prawej
createChar()	Umożliwia zdefiniowanie własnego znaku

Listing 1. Przykładowy program dla modułu AVT11615

```
/*
Przykład programu do obsługi modułu AVT1615 z:
- wyświetlaczem LCD 2x16 znaków
- 4 diodami LED
- 4 przyciskami
- brzęczykiem piezzo
- czujnikiem temperatury LM35
*/


#include <LiquidCrystal.h>  //biblioteka obsługi LCD

const int Led1 =  13;       //przypisanie aliasów do pinów portów
const int Led2 =  12;
const int Led3 =  11;
const int Led4 =  10;
const int SW1 =  3;
const int SW2 =  2;
const int SW3 =  1;
const int SW4 =  0;
const int Buzzer =  A5;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);  //konfigurowanie linii do których został
dołączony LCD

int wart_pot;          //zmienna na wartość zmierzona z potencjometru
int wart_czuj;         //zmienna na wartość zmierzonej z czujnika temperatury
float wart_nap;        //zmienna na wartość zmierzonego napięcia
float temperatura;     //zmienna na wartość zmierzonej temperatury

byte st[8] = {         //tablica znaku stopnia dla wyświetlacza LCD
  B00100,
  B01010,
  B00100,
  B00000,
  B00000,
  B00000,
  B00000,
};

void setup() {             //funkcja inicjalizacji
  lcd.begin(16, 2);        //konfigurowanie rozdzielczości LCD
  lcd.createChar(0, st);  //funkcja utworzenia własnego znaku z tablicy st
o kodzie 0
  analogReference(DEFAULT); //konfigurowanie napięcia odniesienia
                            //dla przetwornika A/C - domyślnie 5V
  pinMode(Led1, OUTPUT);    //konfigurowanie I/O, do których są
                            //dołączone diody LED
  pinMode(Led2, OUTPUT);
  pinMode(Led3, OUTPUT);
  pinMode(Led4, OUTPUT);
  pinMode(Buzzer, OUTPUT);  //konfigurowanie I/O, do której jest
                            //dołączony brzęczyk piezzo
  pinMode(SW1, INPUT);      //konfigurowanie I/O, do których są
                            //dołączone przyciski
  pinMode(SW2, INPUT);
  pinMode(SW3, INPUT);
  pinMode(SW4, INPUT);
  digitalWrite(SW1, HIGH);  //dołączenie wewnętrznych rezystorów
                            //zasilających
  digitalWrite(SW2, HIGH);
  digitalWrite(SW3, HIGH);
  digitalWrite(SW4, HIGH);
  digitalWrite(Led1, HIGH);    //wyłączenie diod LED
  digitalWrite(Led2, HIGH);
  digitalWrite(Led3, HIGH);
  digitalWrite(Led4, HIGH);
  digitalWrite(Buzzer, HIGH);  //wyłączenie brzęczyka piezzo
}

void loop() {              //pętla główna programu
  lcd.clear();             //czyszczenie LCD
  lcd.setCursor(4, 0);     //ustawienie kursora w 5 kolumnie
                           //pierwszego wiersza
  lcd.print(„Arduino”);    //wyświetlenie na LCD napisu Arduino
//pomiar napięcia z potencjometru i dodanie wyniku do wart_pot
  for (int i = 0; i < 20; i++) {    //pętla wykonywana 20 razy
    wart_pot = wart_pot + analogRead(A0);
  }
//obliczenie średniej arytmetycznej z 20 pomiarów
  wart_pot = wart_pot / 20;
//przeliczenie odczytanej wartości na napięcie
  wart_nap=(5.0*wart_pot)/1024.0;
//ustawienie kursora w pierwszej pozycji drugiego wiersza LCD
  lcd.setCursor(0, 1);
  lcd.print(„U=”);         //wyświetlenie napisu U=
  lcd.print(wart_nap);     //wyświetlenie napięcia
  lcd.print(„V”);          //wyświetlenie znaku V

  for (int i = 0; i < 20; i++) {    //pętla wykonywana 20
//pomiar napięcia z czujnika temperatury i dodawanie wart_czuj
    wart_czuj = wart_czuj + analogRead(A1);
  }
//obliczenie średniej arytmetycznej z 20 pomiarów
  wart_czuj = wart_czuj / 20;
//przeliczenie wartości na stopnie Celsjusza
temperatura=(5.0*wart_czuj*100)/1024.0;
//ustawienie kursora na pozycji 9 drugiego wiersza LCD  lcd.setCursor(9, 1);
  lcd.print(„T=”);             //wyświetlenie napisu T=
  lcd.print((long)temperatura);
//wyświetlenie wartości temperatury zaokrąglonej do pełnych stopni
  lcd.write(0);                //wyświetlenie znaku stopnia
```

o napięciu od 0 V do 5 V z rozdzielczością 10-bitow. Komendy *pinMode()* definiują sposób pracy portów I/O mikrokontrolera. Linie, które sterują diodami LED oraz generatorem piezzo skonfigurowano jako wyjścia, a linie, do których dołączono przyciski jako wejścia. Za pomocą funkcji *digitalWrite()* dołączono rezystory zasilające pull-up, które polaryzują wejścia. Przyciśnięcie przycisku zmienia poziom dołączonego wejścia na niski.

W funkcji *loop()* znajduje się program główny. W pierwszej kolejności za pomocą komendy *lcd.clear()* jest czyszczony ekran LCD. Następnie, za pomocą komendy *lcd.setCursor(4, 0)* kursor jest ustawiany w pierwszym wierszu i 5 kolumnie LCD (współrzędne numerowane są od 0). Komenda *lcd.print(„Arduino”)* wyświetla od ustawionej pozycji kursora komunikat *Arduino*. Jako parametr funkcji *lcd.print()* można użyć stałych w cudzysłowie lub zmiennych typu: char, byte, int, long, string. Funkcja ta ma również drugi opcjonalny parametr, który umożliwia formatowanie wyświetlanych liczb. Dozwolone są następujące ich formaty: DEC (dziesiętny), BIN (binarny), OCT (ósemkowy) i HEX (szesnastkowy). Następnie, w programie przykładowym za pomocą komendy *analogRead(A0)* jest odczytywane napięcie zmierzone przez przetwornik A/C na wejściu *A0*. *Jak pamiętamy, jest to napięcie z suwaka potencjometru*. Dla uniknięcia błędów pomiar jest wykonywany 20 razy w pętli *FOR*, jego wynik jest sumowany w zmiennej *wart_pot*, a następnie jest obliczana średnia arytmetyczna z wyników pomiarów. Za pomocą wyrażenia *wart_nap=(5.0*wart_pot)/1024.0* wartość obliczonej średniej jest zamieniana na napięcie. Stała „5.0” to napięcie odniesienia dla przetwornika, natomiast „1024” to jego rozdzielczość. Tak przeliczona wartość zmiennej zostaje wyświetlona w drugiej linii wyświetlacza LCD.

Podobnie w dalszej części programu jest wykonywany pomiar temperatury za pomocą pomiaru napięcia na wyjściu czujnika LM35. Zmiana temperatury o 1ºC powoduje wzrost napięcia na wyjściu czujnika temperatury o 10 mV. Łatwo wywnioskować, że zmiana temperatury o 10ºC będzie odpowiadała zmianie napięcia wyjściowego czujnika o 100 mV. Temperatura jest odczytywana za pomocą funkcji *analogRead(A1)* mierzącej równoważne jej napięcie z czujnika LM35 na wejściu *A1*. Również w tym wypadku jest wykonywane 20 pomiarów, z których jest obliczana średnia arytmetyczna. Następnie za pomocą wyrażenia *temperatura=(5.0*wart_czuj*100)/1024.0* liczba odczytana z rejestru przetwornika A/C jest zamieniana na temperaturę. Wartość temperatury jest wyświetlana na wyświetlaczu w pozycji wskazywanej przez komendę *lcd.setCursor(9, 1)*. Komenda *lcd.write(0)* powoduje wyświetlenie po wartości temperatury symbolu stopnia.

W dalszej części programu umieszczono cztery bloki podobnych instrukcji obsługujących przyciski S1...S4 z diodami oraz brzęczyk piezzo. Ich działanie omówimy na przykładzie instrukcji odczytujących stan przycisku S1, ponieważ pozostałe działają w ten sam sposób.

Za pomocą funkcji *digitalRead(SW1) jest* odczytywany stan wejścia, do którego dołączono przycisk *S1*. Następnie program sprawdza, czy odczytano poziom niski. Jeśli tak, to naciśnięto przycisk i zostają wykonane instrukcje zawarte w warunku *IF*. Jako pierwsza jest zaświecana dioda LED1 za pomocą komendy *digitalWrite(Led1, LOW). Kolejna instrukcja – digitalWrite(Buzzer, LOW) – powoduje załączenie brzęczyka piezzo. Następnie jest* czyszczony ekran LCD, pozycjonowany kursor oraz wyświetlany komunikat informujący o numerze wciśniętego przycisku. W pętli *while(digitalRead(SW1) == LOW)* następuje oczekiwanie na zwolnienie przycisku. Po jej zakończeniu są wykonane instrukcje *digitalWrite(Led1, HIGH), digitalWrite(Buzzer, HIGH)* wyłączające diodę LED oraz brzęczyk piezzo.

Kolejne bloki programowe w taki sam sposób obsługują pozostałe przyciski i diody LED. Analogicznie, naciśnięcie przycisku S2 spowoduje zaświecenie się LED2, S3 zapali LED3 itd. Na ekranie LCD będą wyświetlane odpowiednie komunikaty, a naciśnięcie każdego przycisku będzie sygnalizowane przez brzęczyk piezzo. Program kończy się instrukcją *delay(300), która powoduje zwłokę o czasie trwania* 300 ms.

Więcej informacji na temat komend obsługujących wyświetlacz LCD można znaleźć na stronach internetowych Arduino w informacjach dotyczących zastosowania biblioteki *LiquidCrystal*.

Listing 1. c.d.

```
  lcd.print(„C");                 //wyświetlenie znaku C

//sprawdzenie czy naciśnięto przycisk S1
  if (digitalRead(SW1) == LOW) {
    digitalWrite(Led1, LOW);     //zaświecenie LED1
    digitalWrite(Buzzer, LOW);   //włączenie brzęczyka
    lcd.clear();                 //czyszczenie LCD
//ustawienie kursora w pierwszym rzędzie i drugiej kolumnie    lcd.
setCursor(2, 0);
    lcd.print(„Przycisk S1");    //wyświetlenie nazwy przycisku
//oczekiwanie na zwolnienie przycisku S1
    while(digitalRead(SW1) == LOW);
 } else {                        //w przeciwnym razie
    digitalWrite(Led1, HIGH);    //wyłączenie diody LED1
    digitalWrite(Buzzer, HIGH); //wyłączenie brzęczyka
  }

 if (digitalRead(SW2) == LOW) { //sprawdzenie czy naciśnięto S2
    digitalWrite(Led2, LOW);
    digitalWrite(Buzzer, LOW);
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print(„Przycisk S2");
    while(digitalRead(SW2) == LOW);
  } else {
    digitalWrite(Led2, HIGH);
    digitalWrite(Buzzer, HIGH);
  }

 if (digitalRead(SW3) == LOW) { //sprawdzenie czy naciśnięto S3
    digitalWrite(Led3, LOW);
    digitalWrite(Buzzer, LOW);
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print(„Przycisk S3");
    while(digitalRead(SW3) == LOW);
  } else {
    digitalWrite(Led3, HIGH);
    digitalWrite(Buzzer, HIGH);
  }

 if (digitalRead(SW4) == LOW) { //sprawdzenie czy naciśnięto S4
    digitalWrite(Led4, LOW);
    digitalWrite(Buzzer, LOW);
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print(„Przycisk S4");
    while(digitalRead(SW4) == LOW);
  } else {
    digitalWrite(Led4, HIGH);
    digitalWrite(Buzzer, HIGH);
  }

  delay(300);                    //opóźnienie o 300ms
}                                //koniec pętli głównej
```

Podsumowanie

Moduł LCD zapewne będzie jednym z najczęściej stosowanych we własnych aplikacjach. Uniwersalny, umożliwiający wyświetlanie liczb, komunikatów tekstowych, prostych ikon i pasków postępu na pewno będzie chętnie używany do różnych zadań. Przyciski, w które jest wyposażony zestaw AVT1615 umożliwiają realizację interfejsu użytkownika umożliwiającego na przykład wprowadzanie nastaw. Niewątpliwym atutem są również diody LED i brzęczyk, których można użyć do sygnalizowania stanu budowanego przez siebie urządzenia, sygnalizowania alarmów itp. Mam nadzieję, że zaprezentowane przykłady obsługi wyjaśniają jak można użyć tych elementów we własnym projekcie.

W kolejnym odcinku kursu omówimy podobny moduł, jednak wyposażony w wyświetlacze LED. Umożliwia on na przykład zbudowanie miernika panelowego, zegara i innych urządzeń. Jednymi z najważniejszych cech Arduino są bowiem prostota użycia i niesamowita wręcz elastyczność platformy umożliwiająca różnorodne jej zastosowanie.

Marcin Wiązania
marcin.wiazania@ep.com.pl