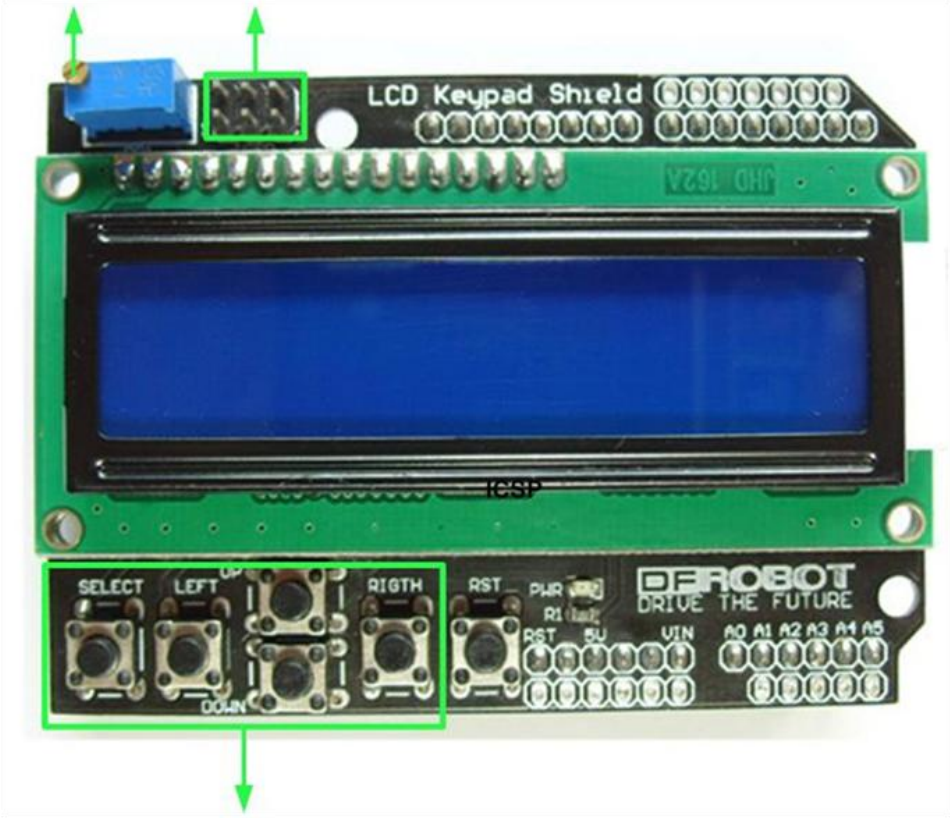
Wyświetlacz LCD. Ogólne zasady działania

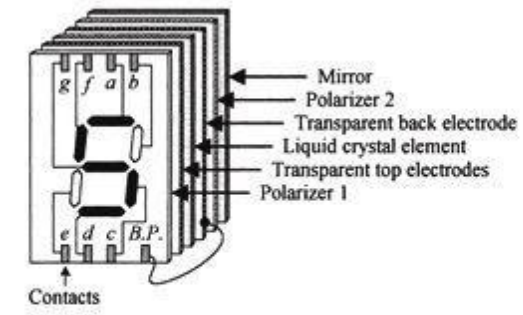
potencjometr regulacji kontrastu

ISP złącze programowan

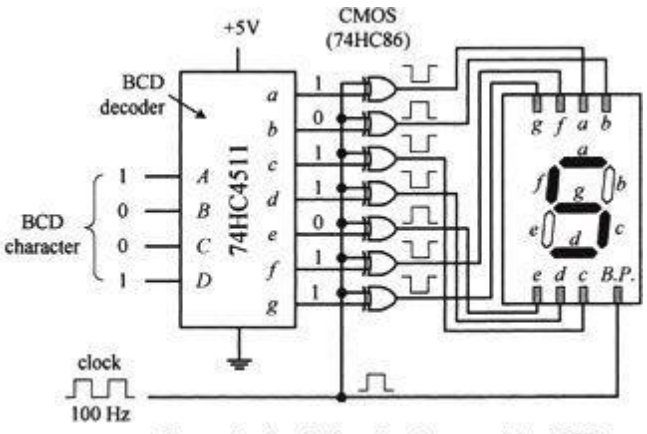


5 klawiszy przyłączonych do wejścia analogowego A0

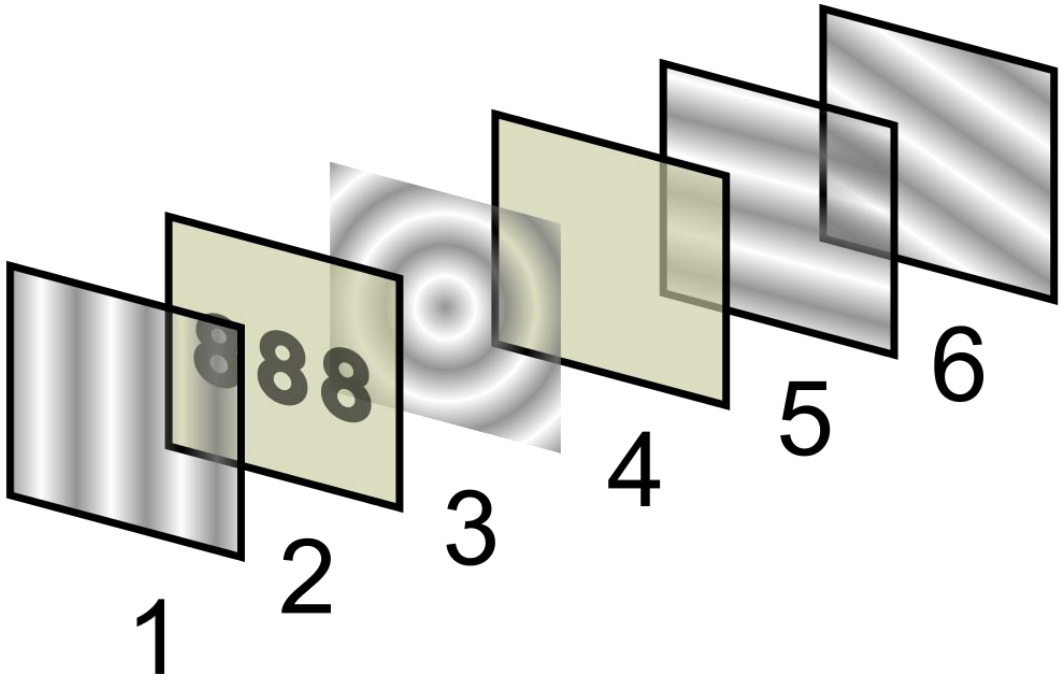
Arduino LCD Keypad Shield



Cutaway view of a reflective-type LCD



Example circuit showing how to drive LCD



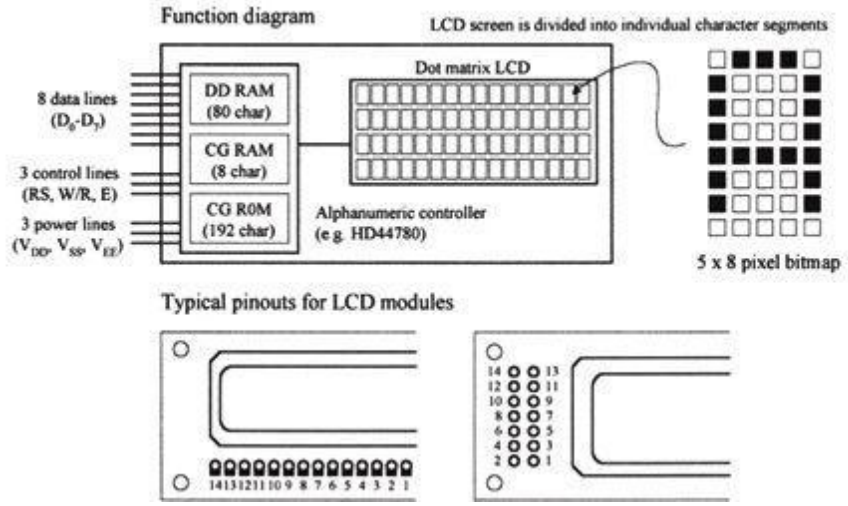
Pasywny wyświetlacz LCD oparty na skręconej fazie nematycznej (TN)

Wszystkie rodzaje wyświetlaczy ciekłokrystalicznych składają się z czterech podstawowych elementów:

1. komórek, w których zatopiona jest niewielka ilość ciekłego kryształu;
2. elektrod, które są źródłem pola elektrycznego działającego bezpośrednio na ciekły kryształ;
3. dwóch cienkich folii, z których jedna pełni rolę polaryzatora a druga analizatora;
4. źródła światła.

Zasadę działania wyświetlacza najłatwiej jest prześledzić na przykładzie pasywnego wyświetlacza odbiciowego, ze skręconą fazą nematyczną. W wyświetlaczu tym światło wnikające do niego jest wstępnie polaryzowane pionowo przez filtr polaryzacyjny (*1*), następnie przechodzi przez szklaną elektrodę (*2*) i warstwę ciekłego kryształu (*3*). Specjalne mikrorowki na elektrodach (*2* i *4*) wymuszają takie uporządkowanie cząsteczek tworzących warstwę ciekłokrystaliczną, aby przy wyłączonej elektrodzie nastąpiło obrócenie polaryzacji światła o 90°. Dzięki temu światło może przejść przez folię (*5*) pełniącą rolę analizatora światła, która przepuszcza tylko światło spolaryzowane poziomo, odbić się od lustra (*6*), przejść ponownie przez analizator (*5*), ulec ponownej zmianie polaryzacji o 90° na warstwie ciekłego kryształu i ostatecznie opuścić bez przeszkód wyświetlacz, przez górną folię polaryzacyjną. Po przyłożeniu napięcia do elektrod, generowane przez nie pole elektryczne wymusza taką zmianę uporządkowania cząsteczek w warstwie ciekłego kryształu, że nie obraca ona polaryzacji światła. Powoduje to, że światło nie przechodzi przez analizator, co daje efekt czerni.

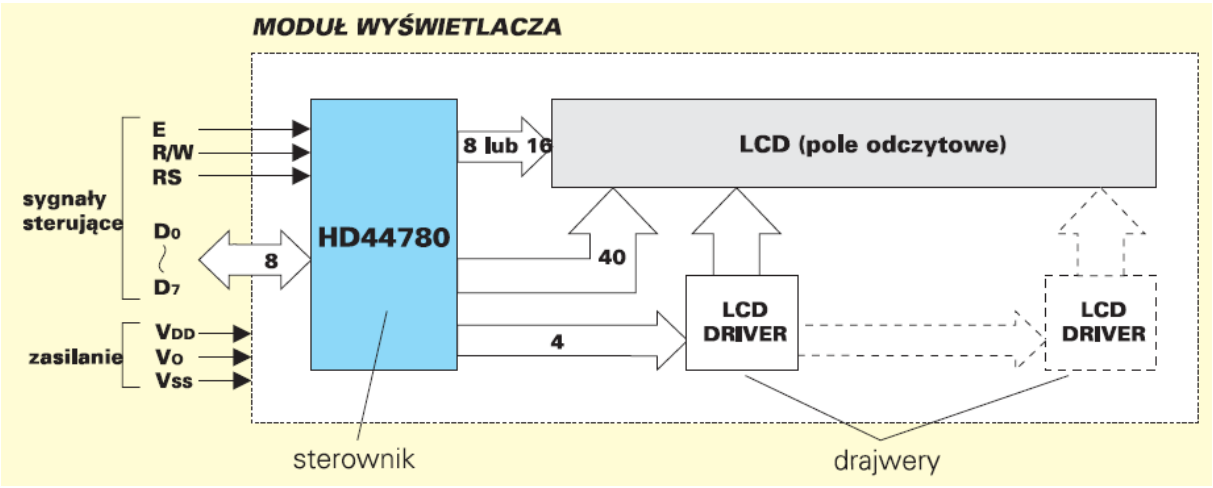
Alphanumeric LCD module



Pin functions

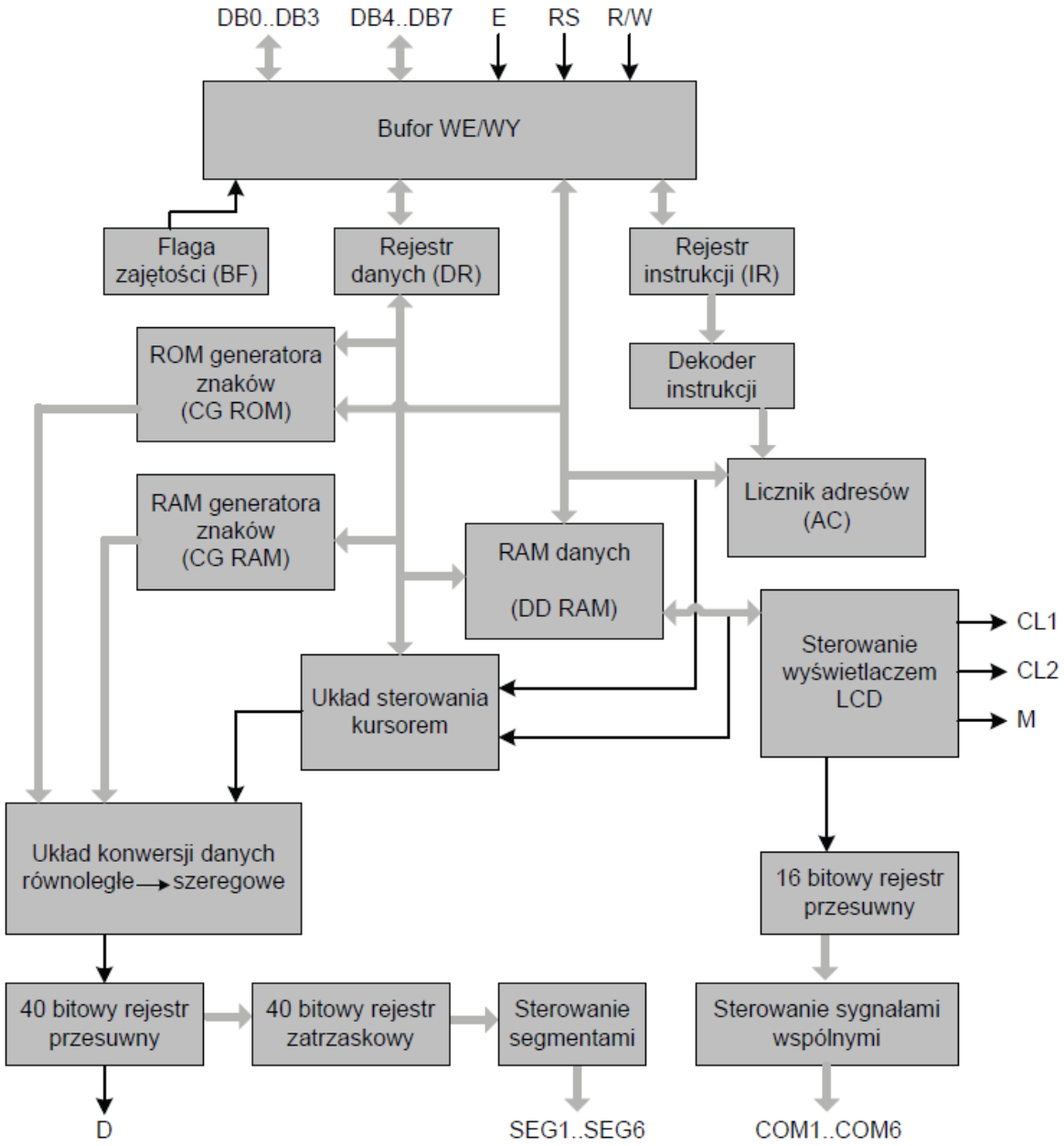
Pin	Name	Function
1	V_{SS}	Ground
2	V_{DD}	+ Supply
3	V_{EE}	Contrast
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D_0	Data bit 0
8	D_1	Data bit 1
9	D_2	Data bit 2
10	D_3	Data bit 3
11	D_4	Data bit 4
12	D_5	Data bit 5
13	D_6	Data bit 6
14	D_7	Data bit 7

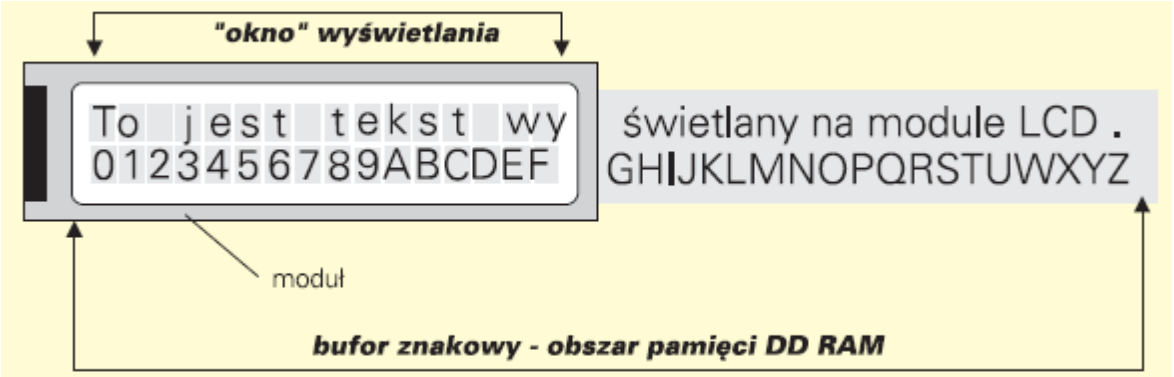
Typical pinouts for LCD modules



Typowa budowa wewnętrzna modułu LCD

Schemat blokowy wyświetlacza





Opis sygnałów wejściowych/wyjściowych wyświetlacza

1. GND (VSS) - masa
2. VCC (VDD) - zasilanie +5V
3. V0 – ustawienie kontrastu
4. RS – wybór rejestru 0-kod instrukcji, 1-dana
5. R/W – odczyt/zapis danej
6. E – sygnał wyboru
7. DB0 – linia przesyłania danych
8. DB1 – linia przesyłania danych
9. DB2 – linia przesyłania danych
10. DB3 – linia przesyłania danych
11. DB4 – linia przesyłania danych
12. DB5 – linia przesyłania danych
13. DB6 – linia przesyłania danych
14. DB7 – linia przesyłania danych
15. A – zasilanie diody LED do podświetlania ekranu
16. K – masa dla diody LED podświetlającej ekran

Dane mogą być przesyłane równolegle przez 8 linii lub tylko przez 4. Oszczędzamy w ten sposób 4 końcówki mikroprocesora.

Wyświetlacz jest kompatybilny z biblioteką **LiquidCrystal** dostarczoną z Arduino IDE. Pełna dokumentacja tej biblioteki znajduje się na stronie **arduino.cc**
Dołączenie biblioteki do programu:

#include <LiquidCrystal.h>

Udostępnia ona następujące funkcje obsługi wyświetlacza:

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

Konstruowanie obiektu

Zanim zaczniemy używać wyświetlacza należy za pomocą konstruktora obiektu utworzyć obiekt klasy *LiquidCrystal.* W bibliotece zdefiniowano kilka wersji konstruktora. Najczęściej będziemy używać konstruktora w postaci:

LiquidCrystal **lcd**(rs, enable, d4, d5, d6, d7);

który pozwala utworzyć obiekt i określić piny Arduino odpowiedzialne za komunikację. W tym przypadku wybieramy połączenie przez 4 piny do przesyłania danych (DB4-DB7).

Przykład użycia konstruktora (dla nakładki Keypad Shield):

LiquidCrystal **lcd**(8,9,4,5,6,7);

LiquidCrystal.begin()

Po utworzeniu obiektu należy określić właściwości wyświetlacza. Dokonuje się tego za pomocą funkcji ***begin(),*** dla której podajemy z reguły dwa argumenty typu int. Pierwszy to liczba znaków w wierszu, drugi to liczba wierszy na wyświetlaczu. Trzeci, opcjonalny parametr, określa rozmiar znaku (5x8 punktów). Wywołanie tej funkcji umieszczamy w funkcji ***void setup()*** naszej aplikacji:

lcd.**begin**(16,2);

Polecenia (funkcje) sterujące kursorem:

LiquidCrystal.cursor(); - włącza wyświetlanie kursora

LiquidCrystal.noCursor(); - wyłącza wyświetlanie kursora

LiquidCrystal.blink(); - włącza migotanie kursora

LiquidCrystal.noBlink(); - wyłącza migotanie kursora

LiquidCrystal.home(); - powrót kursora do pozycji wyjściowej (0, 0)

LiquidCrystal.setCursor(uint8_t **x**, uint8_t **y);** - ustawia kursor w pozycji x (0-15), y (0-1)

Pierwszy parametr informuje o kolumnie, drugi o wierszu, w których ma zostać ustawiony kursor. Kolumny i wiersze numerujemy od zera.

Przykład:

lcd.setCursor(0,1); - ustawienie kursora na początku drugiej linii wyświetlacza

Funkcje sterujące wyświetlaniem tekstów:

LiquidCrystal.clear(); - funkcja czyści cały ekran i wraca kursor do pozycji wyjściowej (0,0)

LiquidCrystal.display(); - włącza wyświetlanie treści zapisanej w buforze na wyświetlaczu

LiquidCrystal.noDisplay(); - wyłącza wyświetlanie na wyświetlaczu

LiquidCrystal.print();
Jest podstawową funkcją służącą do wyświetlania treści na ekranie. Funkcja składowa print wyświetla przekazaną do niej tablicę znakową (właściwie przekazujemy wskaźnik do tej tablicy). Zaczyna od miejsca, w którym jest ustawiony kursor. Domyślnie jest to pierwszy znak za ostatnim znakiem wypisanym na ekranie.

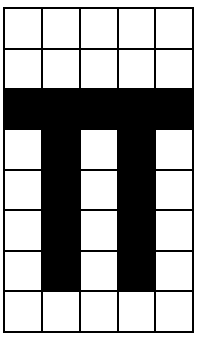
Przykłady użycia funkcji print()

```
lcd.print(”tekst”);

int liczba = 170;
lcd.print(”dziesietnie=”);
lcd.print(liczba, DEC);          // również gdy pominiemy parametr DEC

lcd.print(”binarnie=”);
lcd.print(liczba, BIN);

lcd.print(”szesnastkowo=”);
lcd.print(liczba, HEX);

float  pi = 3.141592654;
lcd.print(”pi=”);
lcd.print(pi, 3);          // 3 cyfry po przecinku (gdy pominiemy – 2)
```

Definiowanie własnych znaków

Istnieje możliwość zdefiniowania w wyświetlaczu do 8 własnych znaków.

```
float pi=3.141592654;
byte a[8] = { B00000,  //definicja własnego znaku Pi
        B00000,
        B11111,
        B01010,
        B01010,
        B01010,
        B01010,
        B00000 };
lcd.createChar(0, a);  //przypisuje tablicę a[8] do pierwszego miejsca na znaki definiowane
lcd.write(byte(0));  //wyświetlenie definiowanego znaku
lcd.print("=");
lcd.print(pi,2);
```



Arduino LCD Keypad Shield

Jest to moduł wyświetlacza LCD 16x2 I klawiszami dla Arduino.

- niebieskie podświetlenie tła z białymi literami tekstu
- korzysta z biblioteki Arduino wyświetlacza 4 bitowego LCD
- dostępne klawisze Left, Right, Up, Down oraz Select
- możliwość regulacji kontrastu wyświetlacza (potencjometr)
- dostępny klawisz Reset Arduino

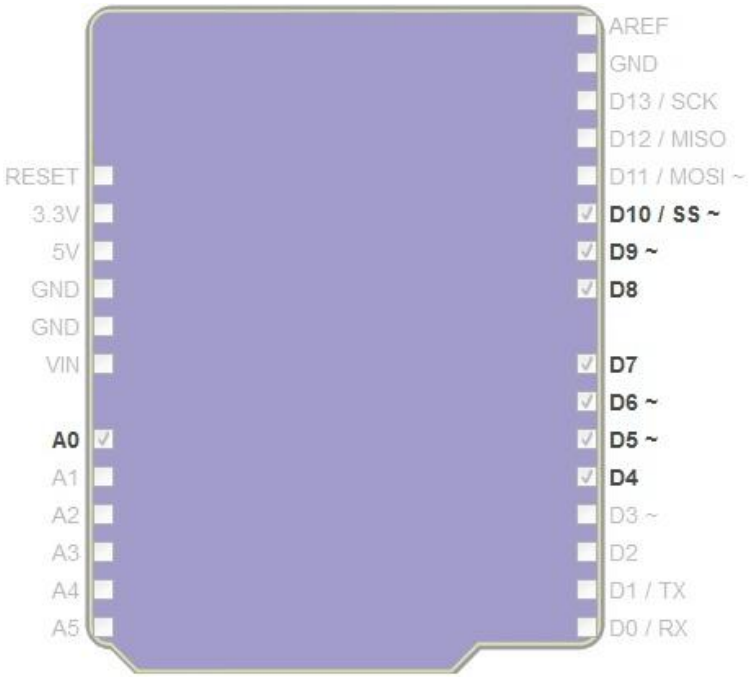
Moduł wykorzystuje wybrane piny Arduino więc deklaracja obiektu powinna być następująca:

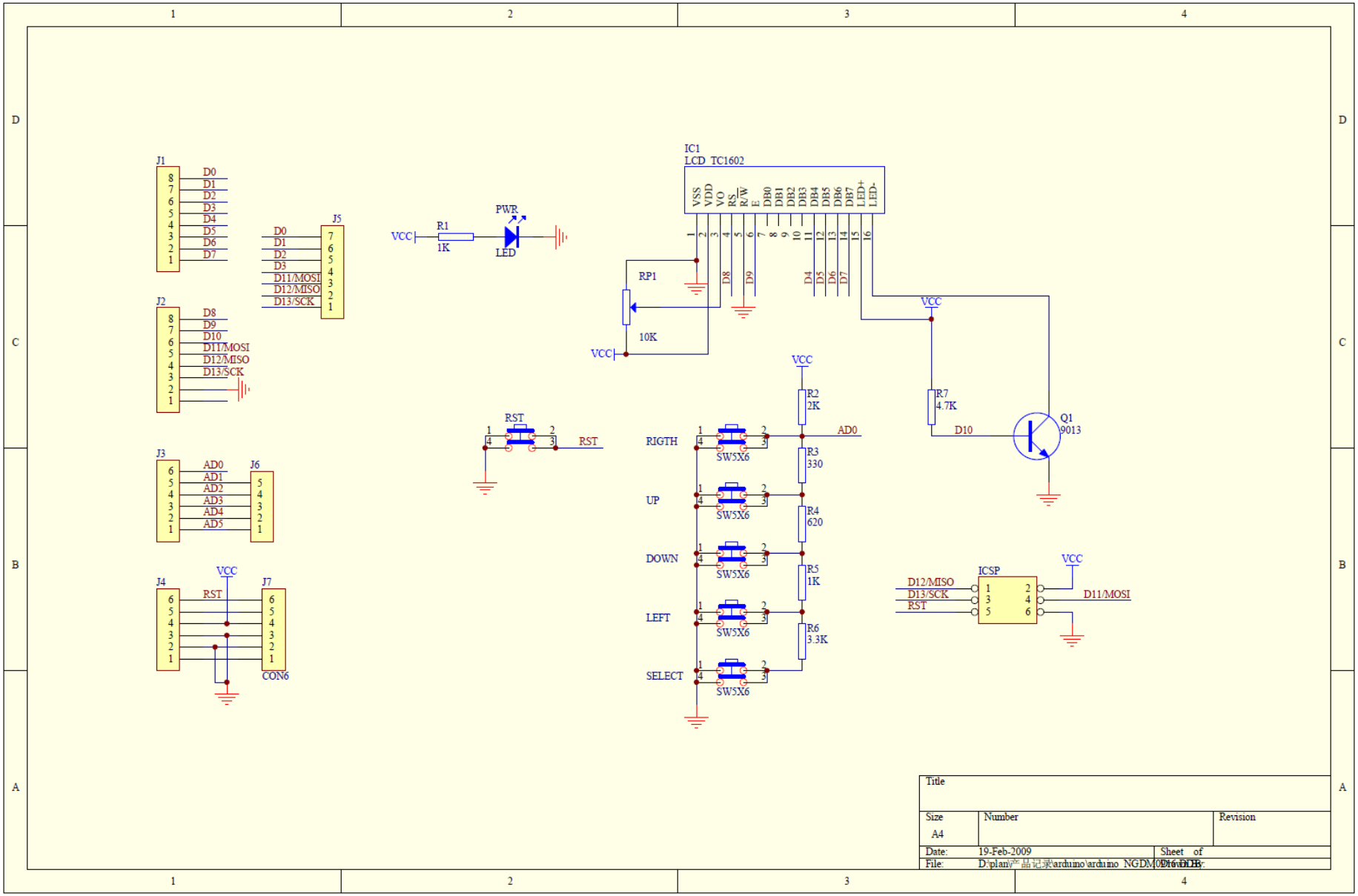
LiquidCrystal lcd(8,9,4,5,6,7);

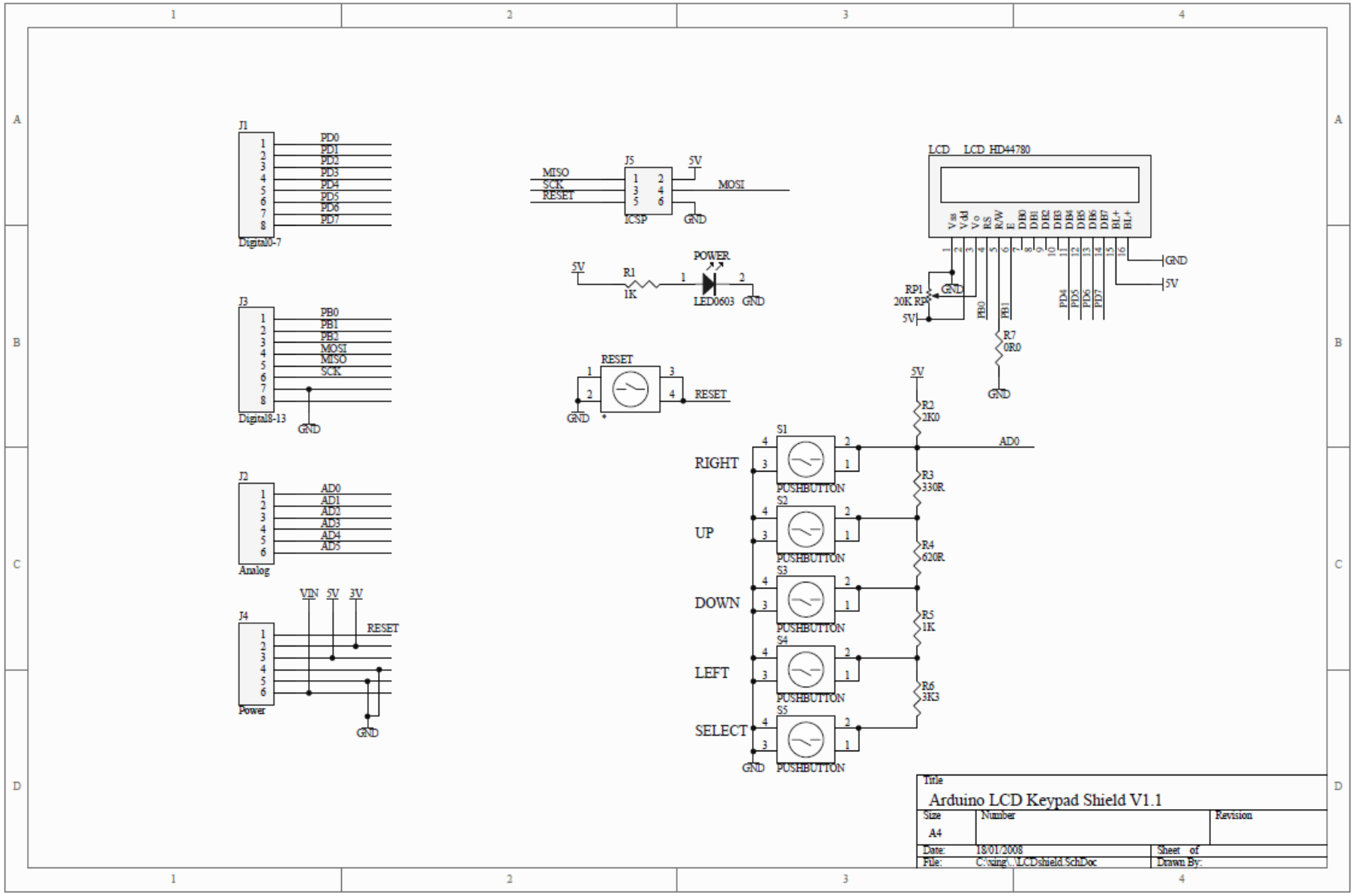
Klawisze przyłączone są do jednego wejścia analogowego (A0) i dzięki przyłączonym rezystorom (dzielnik napięcia) dostarczają różnych napięć dla każdego klawisza. Odczyt i interpretacja klawiszy pokazana jest w przykładowym programie.

Wykorzystanie końcówek Arduino

Pin	**Funkcja**
Analog 0	Buttons (select, up, right, down and left)
Digital 4	DB4
Digital 5	DB5
Digital 6	DB6
Digital 7	DB7
Digital 8	RS (Data or Signal Display Selection)
Digital 9	Enable
Digital 10	Backlit Control HIGH = Backlight on LOW = Backlight off Use PWM signal to control brightness




J1
D0
D1
D2
D3
D4
D5
D6
D7
J2
D8
D9
D10
D11/MOSI
D12/MISO
D13/SCK
J3
AD0
AD1
AD2
AD3
AD4
AD5
J4
RST
VCC
J5
D11/MOSI
D12/MISO
D13/SCK
J6
J7
CON6
R1
1K
PWR
LED
IC1
LCD_TC1602
VSS
VDD
VO
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
LED+
LED-
RP1
10K
R2
2K
R3
330
R4
620
R5
1K
R6
3.3K
R7
4.7K
Q1
9013
RIGTH
UP
DOWN
LEFT
SELECT
SW5X6
AD0
ICSP
Title
Size
A4
Number
Revision
Date:
19-Feb-2009
Sheet of
File:


J1
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
Digital0-7
J3
PB0
PB1
PB2
MOSI
MISO
SCK
Digital8-13
GND
J2
AD0
AD1
AD2
AD3
AD4
AD5
Analog
J4
VIN
5V
3V
RESET
Power
GND
MISO
SCK
RESET
J5
5V
MOSI
ICSP
GND
5V
R1
1K
POWER
LED0603
GND
RESET
RESET
GND
LCD
LCD HD44780
Vss
Vdd
Vo
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
BL+
BL-
GND
5V
RP1
20K RP
GND
5V
PB0
PB1
PD4
PD5
PD6
PD7
R7
0R0
GND
5V
R2
2K0
AD0
S1
RIGHT
PUSHBUTTON
R3
330R
S2
UP
PUSHBUTTON
R4
620R
S3
DOWN
PUSHBUTTON
R5
1K
S4
LEFT
PUSHBUTTON
R6
3K3
S5
SELECT
GND
PUSHBUTTON
Title
Arduino LCD Keypad Shield V1.1
Size
A4
Number
Revision
Date:
18/01/2008
Sheet of
File:
C:\xing\..\LCDshield.SchDoc
Drawn By:

Przykład 1:

```
//Sample using LiquidCrystal library
#include <LiquidCrystal.h>

/*********************************************************
 This program will test the LCD panel and the buttons
Mark Bramwell, July 2010
*********************************************************/

// select the pins used on the LCD panel
LiquidCrystal  lcd(8, 9, 4, 5, 6, 7);

// define some values used by the panel and buttons
int lcd_key     = 0;
int adc_key_in  = 0;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5

// read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);      // read the value from the sensor
 // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
 // we add approx 50 to those values and check to see if we are close
 if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed reasons since it will
be the most likely result
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 195)  return btnUP;
 if (adc_key_in < 380)  return btnDOWN;
 if (adc_key_in < 555)  return btnLEFT;
 if (adc_key_in < 790)  return btnSELECT;
 return btnNONE;  // when all others fail, return this...
}

void setup()
{
 lcd.begin(16, 2);              // start the library
 lcd.setCursor(0,0);
 lcd.print("Push the buttons"); // print a simple message
}
```

```
void loop()
{
 lcd.setCursor(9,1);           // move cursor to second line "1" and 9 spaces over
 lcd.print(millis()/1000);     // display seconds elapsed since power-up


 lcd.setCursor(0,1);           // move to the begining of the second line
 lcd_key = read_LCD_buttons();  // read the buttons

 switch (lcd_key)              // depending on which button was pushed, we perform an action
 {
  case btnRIGHT:
   {
    lcd.print("RIGHT ");
    break;
   }
  case btnLEFT:
   {
    lcd.print("LEFT   ");
    break;
   }
  case btnUP:
   {
    lcd.print("UP    ");
    break;
   }
  case btnDOWN:
   {
    lcd.print("DOWN  ");
    break;
   }
  case btnSELECT:
   {
    lcd.print("SELECT");
    break;
   }
   case btnNONE:
   {
    lcd.print("NONE  ");
    break;
   }
 }

}
```

Przykład 2:

```
// Wczytywanie liczby w kodzie ASCII z portu szeregowego i zamiana na zmienną typu long
#include <LiquidCrystal.h>
long liczba = 0;
int a = 0;
LiquidCrystal  lcd(8, 9, 4, 5, 6, 7);  //RS,EN, D0, D1, D2, D3
void setup()
{
 Serial.begin(9600);
 lcd.begin(16, 2);
}
void loop()
{
 liczba = 0;
 Serial.flush(); // wyczyszczenie bufora portu szeregowego przed transmisja
 while (Serial.available() == 0)
 {
  // nic nie rob dopóki znak nie będzie dostępny
 }
 // po pojawieniu znaku rozpoczynamy przetwarzanie na liczbę binarna
 while (Serial.available() > 0)
 {
  liczba = liczba * 10;

  a = Serial.read() - '0';  // odczytanie kolejnej cyfry i dodanie do liczba
  liczba = liczba + a;
  // oczekiwanie na przesłanie kolejnego znaku
  delay(5);
 }
 lcd.clear();
 lcd.print("wprowadzono:");
 lcd.print(liczba);
 lcd.setCursor(0,1);
 lcd.print(liczba, BIN);
 lcd.print("B  ");
 lcd.print(liczba, HEX);
 lcd.print("H  ");

}
```

Przykład 3:

```
#include <LiquidCrystal.h>
LiquidCrystal  lcd(8,9,4,5,6,7);
char msgs[6][15] = {"Right Key OK ", //Key message
                    "Up Key OK    ",
                    "Down Key OK  ",
                    "Left Key OK  ",
                    "Select Key OK",
                    "None Key OK  "};
int  adc_key_val[6] ={50, 195, 380, 555, 790, 1000};
int NUM_KEYS = 5;
int adc_key_in;
int key=5;
int oldkey=5;

void  setup() {
  pinMode(13, OUTPUT);       //we'll use the debug LED to output a heartbeat
  lcd.begin(16, 2);
  lcd.print("KEYPAD testing... pressing");
}

void  loop() {
  adc_key_in = analogRead(0);           // read the value from the sensor
  digitalWrite(13, HIGH);
  key = get_key(adc_key_in);            // convert into key press

  if (key != oldkey)                    // if keypress is detected
  {
    delay(50);                      // wait for debounce time
    adc_key_in = analogRead(0); // read the value from the sensor
    key = get_key(adc_key_in);          // convert into key press
    if (key != oldkey)
    {
      oldkey = key;
      if (key >=0)          // tutaj można dać switch i akcje dla klawiszy
      {
        lcd.setCursor(0, 1);  //line=2, x=0
            lcd.print(msgs[key]);
      }
    }
  }
  digitalWrite(13, LOW);
}
```

```
int get_key(unsigned int input)        //Convert ADC value to key number
{
  int k;
  for (k = 0; k < NUM_KEYS; k++)
  {
    if (input < adc_key_val[k]) return k;
  }
  return k;        // No valid key pressed k=5
}
```

Zadania:

1. Napisać program, który wyświetli na wyświetlaczu Twoje imię i nazwisko w dwóch wierszach.
2. Napisać program, który na wyświetlaczu wyświetli słowo **gęś**.
3. Napisać program, który podaną liczbę przez port szeregowy zwiększa o 1 i wyświetla na wyświetlaczu LCD.
4. Napisać program, który podane dwie liczby z klawiatury PC sumuje i wynik wyświetla na wyświetlaczu LCD.
5. Napisać program, który losuje trzy liczby z przedziału 1 do 6 (wyniki rzutu trzema kostkami do gry) wyświetla na wyświetlaczu LCD i wysyła przez port szeregowy do komputera, a po naciśnięciu klawisza SELECT operacje powtarza (losuje kolejne trzy liczby itd.).
6. Napisać program, który do bufora wyświetlacza wpisuje napis dłuższy niż 16 znaków (np. Ala ma kotka. Kotek lubi mleko), a następnie korzystając z klawiszy LEFT, RIGHT przesuwamy wyświetlane okno tak, aby można było odczytać cały napis.
7. Napisać na wyświetlaczu swoje imię, a następnie korzystając z przesuwania okna sprawić by imię przesuwało się od lewej do prawej strony wyświetlacza i z powrotem (baner). Podczas przesuwania można wykorzystać efekty specjalne np. migotanie napisu.