Wyższa Szkoła Handlowa w Radomiu

Systemy Wbudowane

Laboratorium 4

1 Cel zajęć

Niniejszy materiał omawia zastosowanie i implementację wyjściowego elementu komunikacyjnego dla Arduino – wyświetlacza LCD. Dodatkowo zaimplementowana zostanie komunikacja układu z urządzeniami peryferyjnymi komputera.

2 Teoria

Obecnie niemal wszystkie użytkowe urządzenia elektroniczne komunikują się z docelowym użytkownikiem za pomocą ekranu. Technika ta wydaje się być najmniej zawodna, do tego komunikaty z niej pochodzące są jednoznaczne. Z tego też powodu producenci montują wyświetlacze w sprzętach AGD, drukarkach, na płytach głównych komputera (jako alternatywna opcja wyświetlania błędów POST), w odtwarzaczach muzycznych czy nawet sprzęcie sieciowym (szczególnie w przełącznikach zarządzalnych bądź routerach, aczkolwiek wyświetlacz może posiadać także punkt dostępowy). Arduino Leonardo, dzięki zastosowaniu procesora z serii u4, umożliwia realizację komunikacji poprzez port szeregowy (w naszym przypadku USB). Wykorzystywana biblioteka Serial pozwala zarówno na odczyt danych z portu jak i zapis danych (poprzez przesył bajtowy). W przypadku używanego zestawu obiekt Serial odnosi się do USB CDC – klasy komunikacji urządzenia (communication device class). Tryb ten pozwala na emulację zachowań innych magistral (jak np. Ethernet czy RS-232) w obrębie połączenia USB. To dzięki temu możliwe jest przesyłanie informacji z Arduino na konsolę systemu operacyjnego komputera – można poznać to chociażby po szybkości transferu (typowe połączenia modemów do komputera poprzez RS-232 pozwalały na transmisję 9600 bauds). Ponieważ układ posiada dodatkowe wyprowadzenia USB w kodzie można odwołać się do nich poprzez obiekt Serial1.

Wykorzystując obiekt Serial oraz bibliotekę Mouse i Keyboard można utworzyć z Arduino zestaw szpiegowsko-automatyzujący.

2.1 Ekran

Dołączony do zestawu laboratoryjnego ekran jest jednym z najprostszych modeli. Pozwala na wyświetlanie dwóch linii tekstu o długości do 16 znaków. Ekran ten należy do rodziny układów HD44780, które mogą posiadać inne parametry (istnieje kilka wariantów ekranów wykorzystujących wspomniany sterownik).

Ekran posiada pamięć wewnętrzną, która podzielona jest na dwie części – pamięć znaków (CGROM – character generator ROM) oraz pamięć danych (DDRAM - display data RAM). Wyświetlacz na stałe posiada zaprogramowany wygląd wyświetlanych znaków. Dodatkowo jako użytkownicy mamy możliwość dopisania maksymalnie 8 znaków do wspomnianej tablicy. Ponadto układ może zapamiętać do 80 bajtów wyświetlanych znaków.

Ekran posiada 16 wyprowadzeń i może być podłączany do zestawu Leonardo na 4 możliwe sposoby:

- poprzez 8 bitową magistralę danych z odczytem flagi zajętości (operacje wejścia/wyjścia)
- poprzez 8 bitową magistralę bez odczytu flagi zajętości
- poprzez 4 bity z 8 magistrali danych z odczytem flagi zajętości (operacje wejścia/wyjścia)
- poprzez 4 bity z 8 magistrali danych bez odczytu flagi zajętości

Najprostszym trybem pracy ekranu jest 4 bitowy transfer bez odczytu zajętości flagi. Daje on możliwość przesyłania danych do wyświetlacza w dwóch krokach (po pół bajtu) bez zbędnego odczytu zajętości wyświetlacza (dane po przesłaniu wyświetlą się na wyświetlaczu zaraz po odebraniu ich przez układ). Dlatego w tym trybie flaga zapisu/odczytu może być zmasowana.

Wykorzystanie jedynie 4 wyprowadzeń danych daje możliwość podłączenia innych elementów wejścia/wyjścia do układu Arduino bez konieczności dołączania do niego układu rozszerzającego (tzw. expander).

Poniżej zaprezentowana została pełna lista instrukcji wyświetlacza wraz z odpowiadającymi im ustawieniami bitów na poszczególnych wyprowadzeniach:

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear display	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	x
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	C	B
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	x	x
Function set	0	0	0	0	1	DL	N	x	BR1	BR0
CGRAM address set	0	0	0	1	CGRAM address					
DDRAM address set	0	0	1	DDRAM address						
Address counter read	0	1	BF=0	AC contents						
DDRAM or CGRAM write	1	0	Write data							
DDRAM or CGRAM read	1	1	Read data							

x = don't care

Display clear - instrukcja ta powoduje wyczyszczenie wyświetlacza poprzez wypełnienie go spacjami, ustawienie trybu zapisu danych od pozycji w lewym górnym rogu wyświetlacza oraz wyłączenie trybu przesuwania okna

Cursor home - instrukcja powoduje ustawienie kursora na pozycji pierwszego znaku w pierwszej linii

Entry mode set - określenie trybu pracy kursora/okna wyświetlacza :

- S = 1 po zapisaniu znaku do wyświetlacza kursor nie zmienia położenia, natomiast przesuwa się cała zawartość wyświetlacza
- S = 0 po zapisaniu znaku do wyświetlacza kursor zmienia położenie, a przesuwanie okna jest wyłączone
- I = 1 kursor lub okno wyświetlacza przesuwa się w prawo (inkrementacja adresu znaku)
- I = 0 kursor lub okno wyświetlacza przesuwa się w lewo (dekrementacja adresu znaku)

Display ON/OFF control :

- D = 1 - włączenie wyświetlacza
- D = 0 - wyłączenie wyświetlacza
- C = 1 - włączenie kursora
- C = 0 - wyłączenie kursora
- B = 1 - włączenie migania kursora
- B = 0 - wyłączenie migania kursora

Cursor/Display shift :

- S = 1 - przesuwana jest zawartość okna
- S = 0 - przesuwany jest kursor
- R = 1 - kierunek przesuwu w prawo
- R = 0 - kierunek przesuwu w lewo

Function set :

- D = 1 - interfejs 8-bitowy
- D = 0 - interfejs 4-bitowy
- N = 1 - wyświetlacz dwuwierszowy
- N = 0 - wyświetlacz jednowierszowy

- F = 1 - matryca znaków 5*10 punktów
- F = 0 - matryca znaków 5*7punktów

CG RAM set - ustawia adres pamięci generatora znaków (DB5-DB0 AAALLL). AAA - 3-bitowy adres znaku, LLL - 3-bitowy numer linii składającej się na graficzne odwzorowanie znaku.

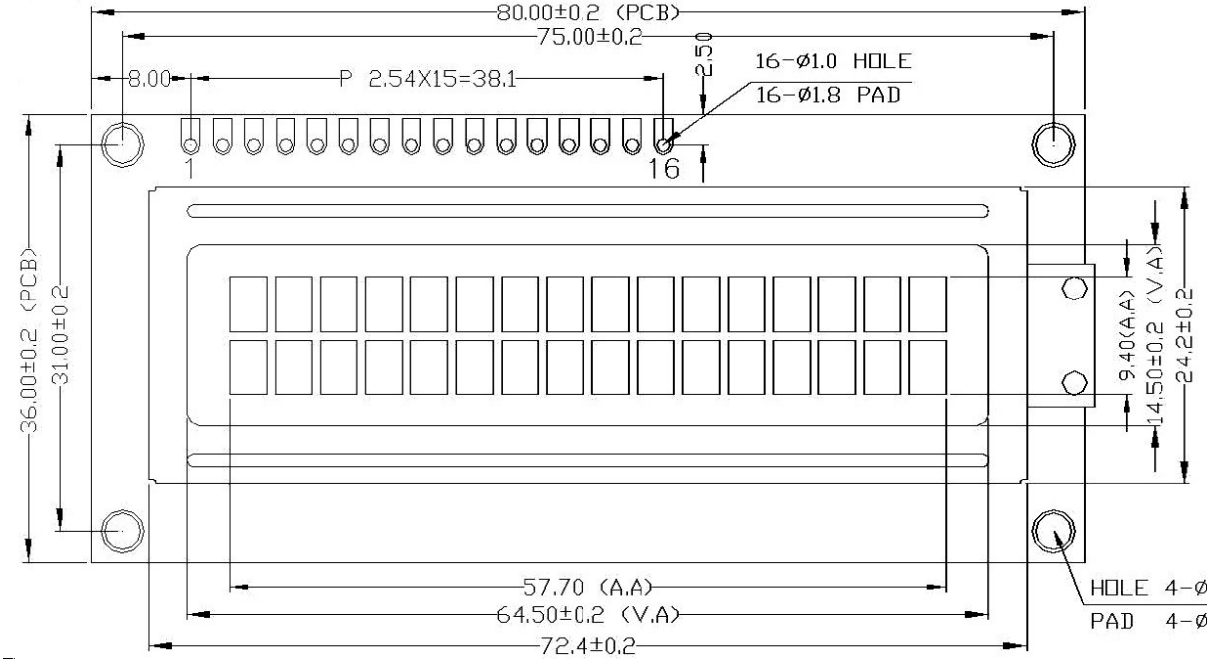
DD RAM set - ustawia adres pamięci wyświetlacza, pod który nastąpi zapis (bądź odczyt) danych operacją Data write lub Data read.

Address counter read - odczyt flagi zajętości i adresu pamięci wyświetlacza.

- B - flaga zajętości wyświetlacza

Data read - odczyt danych z pamięci wyświetlacza, bądź pamięci CG RAM (jeśli poprzednio wydano komendę CG RAM set)

Data write - zapis danych do pamięci wyświetlacza, bądź pamięci CG RAM (jeśli poprzednio wydano komendę CG RAM set)



Rysunek 1: Schemat wyświetlacza

Opis wyprowadzeń:

Nr	Nazwa	Opis
1	VSS	Masa
2	VDD	Zasilanie +5V
3	V0	Kontrast
4	RS	Wybór rejestru instrukcji wyświetlacza (stan niski) albo rejestru danych(wysoki)
5	R/W	Odczyt (stan niski) / Zapis (stan wysoki)
6	E	Odblokowanie wyświetlacza
7	DB0	Magistrala danych.
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LEDA	Zasilanie podświetlania +5V
16	LEDK	Masa podświetlania

Środowisko Arduino posiada przygotowaną bibliotekę do obsługi opisywanego wyświetlacza. Wystarczy ją dołączyć:

```
#include <LiquidCrystal.h>
```

2.1.1 Przykładowy kod wykorzystujący gotową bibliotekę.

```
1  #include <LiquidCrystal.h>
2
3  //zmienna jako globalna
4  //poszczegolne liczby to kolejne wyprowadzenia
5  //Arduino odpowiednio podlaczone do zestawu
6  //wiecej informacji w dokumentacji biblioteki
7  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
8
9  void setup()
10 {
11     //wykorzystujemy 16 kolumn tekstu i dwa wiersze
12     //sterownik wyswietlacza jest w stanie obslugiwac
13     //do 40 kolumn/4 wierszy stad wymog podania
14     //odpowiednich parametrow
15     lcd.begin(16,2);
16     //czyscimy ekran (dla pewnosci) oraz ustawiamy
17     //kurosr w gornej czesci ekranu
18     lcd.clear();
19     //przesuniecie kursora wypisujacego do pozycji 6
20     lcd.setCursor(6,0);
21     //przeslanie napisu na ekran
22     lcd.print("Arduino");
23     //ustawienie kursora w drugim wierszu
24     lcd.setCursor(0,1);
25     lcd.print("Zajecia numer 4");
26     //wylaczenie mrugajacego kursora (nie
27     //jest rownoznaczne z usunieciem go)
28     lcd.noBlink();
```

```
29  }
30
31  void loop()
32  {
33      delay(1000);
34      //petla przesuwajaca cala zawartosc
35      //ekranu; w naszym przypadku o 15 kolumn
36      for(int i = 0; i < 15; i++)
37      {
38          lcd.scrollDisplayLeft();
39          delay(50);
40      }
41      for(int i = 0; i < 30; i++)
42      {
43          lcd.scrollDisplayRight();
44          delay(50);
45      }
46      for(int i = 0; i < 15; i++)
47      {
48          lcd.scrollDisplayLeft();
49          delay(50);
50      }
51  }
52  //sumarycznie tekst powinien poruszac sie
53  //do lewej strony, nastepnie do prawej
54  //po czym wyrownywac sie do pozycji pierwotnej
```

Podłączenie zestawu należy wykonać wedle schematu dołączonego do przykładu obsługi ekranu [4]. Należy zauważyć, że w przykładzie tym linia kontrastu (V0) podpięta została pod potencjometr. W naszym wypadku dobrze byłoby zamienić potencjometr na rezystor (chyba, że chcemy wypróbować rozjaśnianie/przyciemnianie ekranu). W przypadku rezygnacji z potencjometru należy zastosować rezystor ok. 10 kohm.

2.2 Wykorzystanie klawiatury i myszy

Jak już zostało wspomniane we wstępie, gotowe biblioteki Leonardo pozwalają w dość prosty sposób wykonać komunikację układu z komputerem. W przypadku klawiatury jest to komunikacja dwustronna, w przypadku myszy – tylko w kierunku komputera (odczyt danych z myszy podłączonej do komputera nie ma większego sensu).

2.2.1 Przysłanie danych do Arduino z klawiatury (+ przykładowy kod)

Obsługa przechwytywania danych z klawiatury jest dość prosta. Po rozpoczęciu działania obiektu Serial biblioteka domyślnie działa w obu kierunkach – zarówno odczytu (pobranie z klawiatury) jak i zapisu danych (wysłanie informacji na systemową konsolę). Przykładowo wysłanie danych do Arduino Leonardo można zrealizować poprzez następujący szkic:

```
1   char bytes[256];
2
3   void setup()
4   {
5       Serial.begin(9600);
6   }
7
8   void loop()
9   {
10      if (Serial.available() > 0)
11      {
12          Serial.setTimeout(5000);
13          Serial.readBytes(bytes, 256);
```

```
14            Serial.println(bytes);
15            Serial.flush();
16        }
17    }
```

Działanie kodu jest niezwykle proste – deklarujemy zmienną tablicową, która staje się naszym buforem przechwytującym dane z portu szeregowego. Następnie budzimy komunikację szeregową poprzez konsolę szeregową. W pętli oczekujemy na nadejście jakichkolwiek danych z urządzenia wejściowego – jeżeli takie zdarzenie będzie miało miejsce to zmieniamy czas odczekiwania portu szeregowego na dane do 5 sekund. Po tym czasie dane zapisywane są do naszego bufora i wysyłane na powrót do konsoli (celem sprawdzenia czy to co zapisaliśmy do kontrolera faktycznie zostało przez niego odebrane). Na koniec czyścimy bufor szeregowy (żeby nie wystąpiły przekłamania przy kolejnym odczycie portu).

2.2.2 Arduino jako klawiatura (+przykładowy kod)

Dostępna domyślnie biblioteka Keyboard.h pozwala na zamianę układu w fałszywą klawiaturę. Poprzez USB przesyłane są kody jakie emituje prawdziwa klawiatura (każdy klawisz ma swój numeryczny kod). Przykład aplikacji przesyłającej dane z fałszywej klawiatury:

```
1   #include <Keyboard.h>
2
3
4   void setup()
5   {
6       Keyboard.begin();
7   }
8
9   void loop()
10  {
11      delay(1000);
12      Keyboard.press(KEY_RIGHT_SHIFT);
13      Keyboard.println("tekst pisany malymi literami");
14      delay(2000);
15      Keyboard.release(KEY_RIGHT_SHIFT);
16      Keyboard.println("TEKST PISANY DUZYMI LITERAMI");
17      delay(1000);
18      Keyboard.press(KEY_LEFT_CTRL);
19      Keyboard.press('a');
20      Keyboard.releaseAll();
21      Keyboard.press(KEY_DELETE);
22      Keyboard.releaseAll();
23  }
```

Powyższy przykład w pętli będzie emulował wciśnięcie prawego przycisku SHIFT (na klawiaturze powinna zapalić się odpowiednia dioda), powinien zostać wypisany tekst w aktualnie aktywnym oknie edycji (zaleca się otworzyć np. notatnik). Po dwóch sekundach przycisk SHIFT zostanie zwolniony i powinien pojawić się drugi napis. Następnie wcześniej napisany tekst powinien zostać zaznaczony i usunięty.

2.2.3 Arduino jako mysz (+ przykładowy kod)

Układ może także podszywać się pod mysz. W tym celu wykorzystuje on układ współrzędnych XY oraz emulację 3 przycisku (suwaka, tzw. dystans). Ponadto potrafi wysyłać sygnał wciśnięcia bądź zwolnienia wskazanego przycisku myszy. Przykład działania myszy:

```
1   #include <Mouse.h>
2
```

```
3  void setup()
4  {
5      Mouse.begin();
6      for (unsigned int i = 0; i < 10; i++)
7          Mouse.move(-250,-250, 0);
8      randomSeed(analogRead(0));
9  }
10
11 void loop()
12 {
13     delay(1000);
14     Mouse.move(random(-250,250), random(-250,250), 0);
15     Mouse.press(MOUSE_RIGHT);
16     delay(500);
17     Mouse.releaseAll();
18     Mouse.move(random(-250,250), random(-250,250), 0);
19     Mouse.click();
20 }
```

W pętli setup wykorzystana została pętla która przesuwa kursor w ley górny ekran. Chodzi o wyłapanie właściwych koordynat XY – Arduino nie ma możliwości ustawienia bezwzględnej pozycji kursora – wszelkie przesunięcia są względne (np. 20 punktów w prawo względem aktualnych 250 punktów – czyli nowa pozycja myszy będzie wynosić 300). Dodatkowo wykorzystano moduł losowości – wartością początkową dla losowań kolejnych wartości jest tzw. szum biały (każdorazowo inny). Ważne jest jedynie by wybrane analogowe wyprowadzenie nie było wykorzystywane jako podłączenie jakiegokolwiek modułu (wtedy liczba losowości może być stała).

W pętli głównej program co sekundę zmienia losowo pozycję kursora myszy. W wybranej pozycji klika prawym przyciskiem myszy, odczekuje pół sekundy po czym zwalania wszystkie wciśnięte przyciski. Na koniec mysz przesuwana jest na kolejną losową pozycję i następuje kliknięcie lewym przyciskiem myszy (użycie tej funkcji nie powoduje konieczności zwolnienia przycisku).

3 Zadania do realizacji

1. Zaimplementować obsługę odczytu wartości z klawiatury w taki sposób by wczytywała jedynie ilość przesłanych bajtów (naciśniętych klawiszy), nie zaś pełnego bufora 256 znaków
2. Rozwiązać problem zdeformowania przesyłanych znaków przez port szeregowych (pojedyncze znaki przesłane z powrotem na konsolę nie są tymi, które były przez nas pierwotnie przesyłane)
3. Zaimplementować rozwiązanie, które pozwoli na odbieranie znaków do chwili wystąpienia określonego zdarzenia, np. słowa END.
4. Zaimplementować obsługę urządzenia wskazującego komputera przy pomocy Arduino. Przykładowo przy wykorzystaniu 5 przycisków można osiągnąć następujący efekt:
   - jeden z przycisków określa czy kursor ma się poruszać w przód czy wstecz (znaków +/
   -)
   - dwa przyciski służą jako pozycje XY
   - 4 przycisk odpowiada za obsługę lewego przycisku myszy
   - 5 przycisk odpowiada za obsługę prawego przycisku myszy

   Oczywiście to jest tylko przykład implementacji. W realizacji można posłużyć się także fotorezystorem czy podczerwienią (czyli zbliżyć się do rozwiązań stosowanych w standardowych myszach).
5. Tekst wprowadzany przez klawiaturę należy przesłać na wyświetlacz LCD. Tekst powinien automatycznie przesuwać się w lewo w przypadku tekstu dłuższego niż 16 znaków.

6. Dodać obsługę przesuwania kursora poprzez strzałki na klawiaturze

7. Napisać aplikację emulującą tzw. dalekopis. W tym celu można wykorzystać przyciski dołączone do zestawu oraz klawiaturę. Zasadą działania od strony Arduino byłoby wychwytywanie czy użytkownik „wpisał" kropkę czy kreskę (w zależności od odstępu czasowego pomiędzy kolejnym kliknięciem w przycisk). Tak odebrany tekst powinien być przesłany jako zdarzenie klawiatury (wyświetlić się np. w notatniku systemu operacyjnego). Z kolei od strony komputera wpisywane znaki powinny być być przesyłane do Adruino. Na ekranie urządzenia w jednym rzędzie powinny wyświetlać się symbole danego znaku w postaci alfabetu Morse'a, w drugim natomiast w postaci standardowych znaków.

   UWAGA: W powyższym zadaniu należy zadbać by emulacja klawiatury przez Arduino odbywała się jedynie w określonym przez nas terminie tj. przy wciśnięciu dodatkowego przycisku na układzie bądź nadejścia odpowiedniego znaku (np. popularnego dla tego typu komunikacji STOP).

   ZADANIE: można wykonać implementując jedynie kilka pierwszych znaków alfabetu Morse'a (np. a, b oraz c)

8. Należy dodać własne znaki do pamięci wyświetlacza oraz spowodować ich wyświetlenie.

9. Należy sprawdzić szybkość działania poszczególnych funkcji wypisujących tekst na ekranie. Czy istnieje możliwość przyspieszenia działania ekranu bądź uniezależnienia działania układu od czasu obsługi wyświetlacza?

   DLA STUDENTÓW STUDIÓW NIESTACJONARNYCH:

10. Należy sporządzić sprawozdanie z przebiegu ćwiczenia według podanego szablonu (dostępny na stronie).

4 Materiały wykorzystane przy opracowaniu

https://botland.com.pl/wyswietlacze-alfanumeryczne/223-wyswietlacz-lcd-2x16-znakow.html

https://botland.com.pl/content/31-arduino-i-wyswietlacz-lcd

http://radzio.dxp.pl/hd44780/

https://www.arduino.cc/en/Tutorial/HelloWorld?from=Tutorial.LiquidCrystal

http://forum.atnel.pl/topic868.html

https://www.arduino.cc/en/Reference/Serial

https://www.arduino.cc/en/Reference/MouseKeyboard