WYDZIAŁ ELEKTRYCZNY
KATEDRA INŻYNIERII SYSTEMÓW, SYGNAŁÓW i ELEKTRONIKI
NAZWA PRZEDMIOTU:
Mikrokontrolery i urządzenia wbudowane - PROJEKT
Temat projektu: Projekt sterownika wyświetlacza LCD z interfejsem RS422.
Projekt wykonał: Jóźwiak Damian
Kierunek studiów: Automatyka i Robotyka, sem. 4, rok akad.2015/2016

Prowadzący projekt: mgr inż. Tomasz Miłosławski

Data oddania projektu:
….....Ocena.....
.....

Oświadczenie
Oświadczam, że jestem wyłącznym autorem niniejszego projektu oraz, że praca ta została wykonana samodzielnie i nie narusza praw autorskich innych osób.

Podpis studenta
.....
.....

Contents

Cel projektu

Celem projektu było zaprojektowanie i wykonanie sterownika wyświetlacza LCD w wersji 16x2 znaków przez standard komunikacyjny RS422.

Założenia projektowe:

a) Wymiar płytki PCB - maksymalnie 10x8cm
b) Płytka 2 warstwowa
c) Komunikacja full duplex z urządzeniami zewnętrznymi przez interfejs RS422
d) Zasilanie z niestabilizowanego źródła napięcia stałego 12V
e) Protokół oparty o komunikację za pomocą kodów znaków ASCII
f) Zapewnienie podstawowej funkcjonalności - określenie pozycji znaku i jego przesłanie, czyszczenie ekranu.
g) Wykorzystanie wszystkich linii danych wyświetlacza
h) Niski koszt
i) Mały pobór prądu
j) Brak obudowy, podstawą są gumowe nóżki.

Wybór rozwiązania projektowego i uzasadnienie:

a) Mikrokontroler:
   Do niniejszego projektu wykorzystano mikrokontroler AVR ATmega8A-PU, ze względu na doświadczenie autora w programowaniu rodziny ATmega oraz odpowiednią liczbę portów urządzenia(przy minimalnej funkcjonalności pozostało tylko 7 wolnych pinów).
b) Taktowanie mikrokontrolera:
   Autor zaadaptował dość niskie taktowanie zewnętrznym rezonatorem kwarcowym o częstotliwości 3.686400MHz, który zapewnia całkowity dzielnik dla baud rate przy prędkości 115200kbit/s, oraz niski pobór prądu przez procesor - 2.5mA
c) Sterownik liniowy RS422:
   Jako konwerter standardów TTL <-> RS422 wykorzystano układ scalony MAX488CSA+, firmy MAXIM-DALLAS. Wybór ten był uzasadniony niską ceną układów, poborem prądu rzędu 500μA, maksymalną prędkością przesyłu 250kbit/s oraz zabezpieczeniami przeciwko: zwarciom i błędom.
d) Wyświetlacz LCD:
   W projekcie wykorzystano najmniejszy dostępny wyświetlacz o rozdzielczości 16x2 znaków, wybór był uzasadniony redukcją ceny końcowej urządzenia.
e) Regulator napięcia:
   Regulator został wybrany tak, by zapewniał na wyjściu 5V zgodne z logiką TTL oraz wymaganiami podświetlenia wyświetlacza. Typ obudowy poparty obliczeniami z punktu 4.II i 4.III to DPAK.

Część opisowa:

Projektowany sterownik ma za zadanie pośredniczyć w komunikacji dowolnego urządzenia typu master z wyświetlaczem LCD 16x2 znaków. Zainstalowany mikrokontroler ma za zadanie tylko określać czy dany pakiet danych jest adresowany do niego, oraz sprawdzać czy ciąg znaków ASCII jest zgodny z przyjętym formatem w postaci:

DELIMITER+IDENTYFICATION_NUMBER+COMMAND_IDENTYFICATOR+

+PRARMETERS+DELIMITER

A więc by wyświetlić na ekranie ciąg znaków ‘Hello world’ należy wysłać ciąg znaków

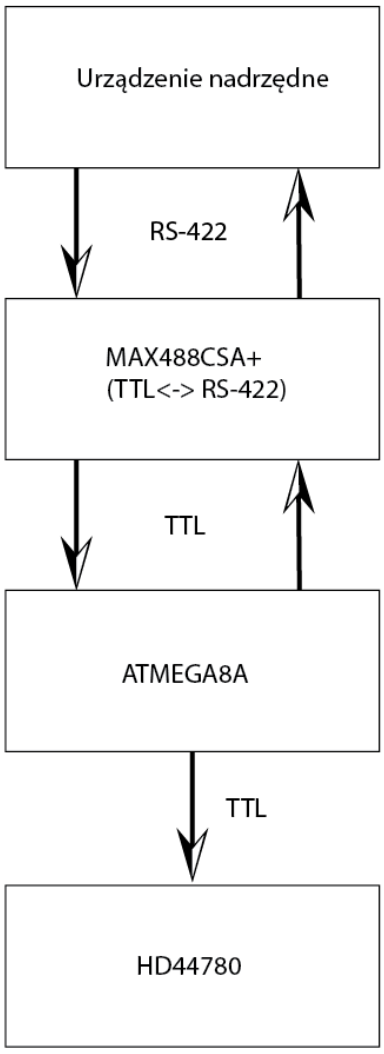
0x041pHello world0x04

Sterownik nie implementuje obsługi kontroli błędów komunikacji. Dane o niewłaściwym formacie zostają odrzucone. Nie zaleca się jego użycia gdzie owa kontrola jest ściśle wymagana. Domyślnie sterownik dostarczany jest z fabrycznymi ustawieniami:

Prędkość transmisji: 115200kbit/s

Identyfikator urządzenia: patrz naklejka na wyświetlaczu.

Schemat blokowy:



Rysunek 1: Schemat blokowy.

Źródło: Opracowanie własne.

Schemat ideowy:

Obliczenia, dobór elementów:

I. Dobór jasności diod LED:

a) Rezystor ograniczający prąd podświetlenia wyświetlacza:

ELECTRICAL RATINGS (Backlight component)

Ta = 25°C

Item	Symbol	Condition	Min	Typ	Max	Unit
Forward Voltage	VF	IF=90mA	4.0	4.2	4.4	V
Reverse Current	IR	VR=10.0V	---	90	---	uA
Luminous Intensity (Without LCD)	Lv	IF=90mA	---	160	---	cd/m^2

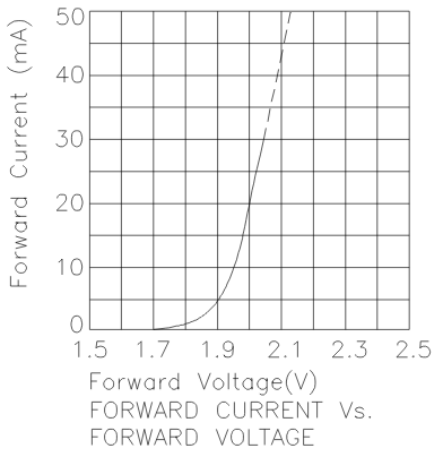
Rysunek 2: Parametry elektryczne podświetlenia wyświetlacza.

Źródło: Karta katalogowa.

Zgodnie z dokumentacją spadek napięcia na diodzie wyświetlacza to 4.2V, przy prądzie 90mA, lecz doświadczenie projektanta określa iż tak duży prąd nie jest potrzebny, by zapewnić dobrą widoczność. Nawet w całkowitej ciemności wystarczy prąd rzędu 10mA, a więc:

$$R = \frac{V_{ZAS} - V_F}{I_F} = \frac{5V - 4.2V}{0.010A} = 80 \approx 82\Omega$$

b) Rezystor ograniczający prąd diody zasilania:



Rysunek 3: Prąd przewodzenia w funkcji napięcia przewodzenia.

Źródło: karta katalogowa.

Dioda w urządzeniu będzie pełniła poboczną rolę jako wskaźnik obecności zasilania, zatem jej jasność, a więc i prąd wybrano jak najniższe - tak by nie oślepiały osoby obsługującej. Ponownie jako wyjście poparte doświadczeniem projektanta wybrano prąd 5mA, a więc:

$$R = \frac{V_{ZAS} - V_F}{I_F} = \frac{5V - 1.9V}{0.005A} = 620 \approx 680\Omega \ .$$

II. Oszacowanie poboru prądu:

a) Mikrokontroler - 4.5mA

b) Sterownik liniowy - 500 μA
c) Wyświetlacz - logika: 800 μA, podświetlenie: 10mA
d) Dioda sygnalizacyjna LED - 5mA
e) Prądy upływu, peryferia, rezystory podciągające, zmiany temperatury - max 10mA

Razem:

I_{OUT} =30.8mA

III. Obliczenia temperatury:

Stabilizator MC7805BDTG

Dane katalogowe:

- Maksymalna temperatura złącza: $T_J = 150℃$
- Rezystancja termiczna złącze-otoczenie $R_{\theta JA} = 92\frac{℃}{W}$
- Rezystancja termiczna złącze-obudowa $R_{\theta JC} = 5.0\frac{℃}{W}$

Warunki pracy:

- Maksymalna temperatura otoczenia 50℃
- Moc strat $P = \Delta U * I_{OUT} = (U \quad - U_{OUT}) * I_{OUT} = (12V - 5V) * 31mA = 7V * 31mA = 217mW$

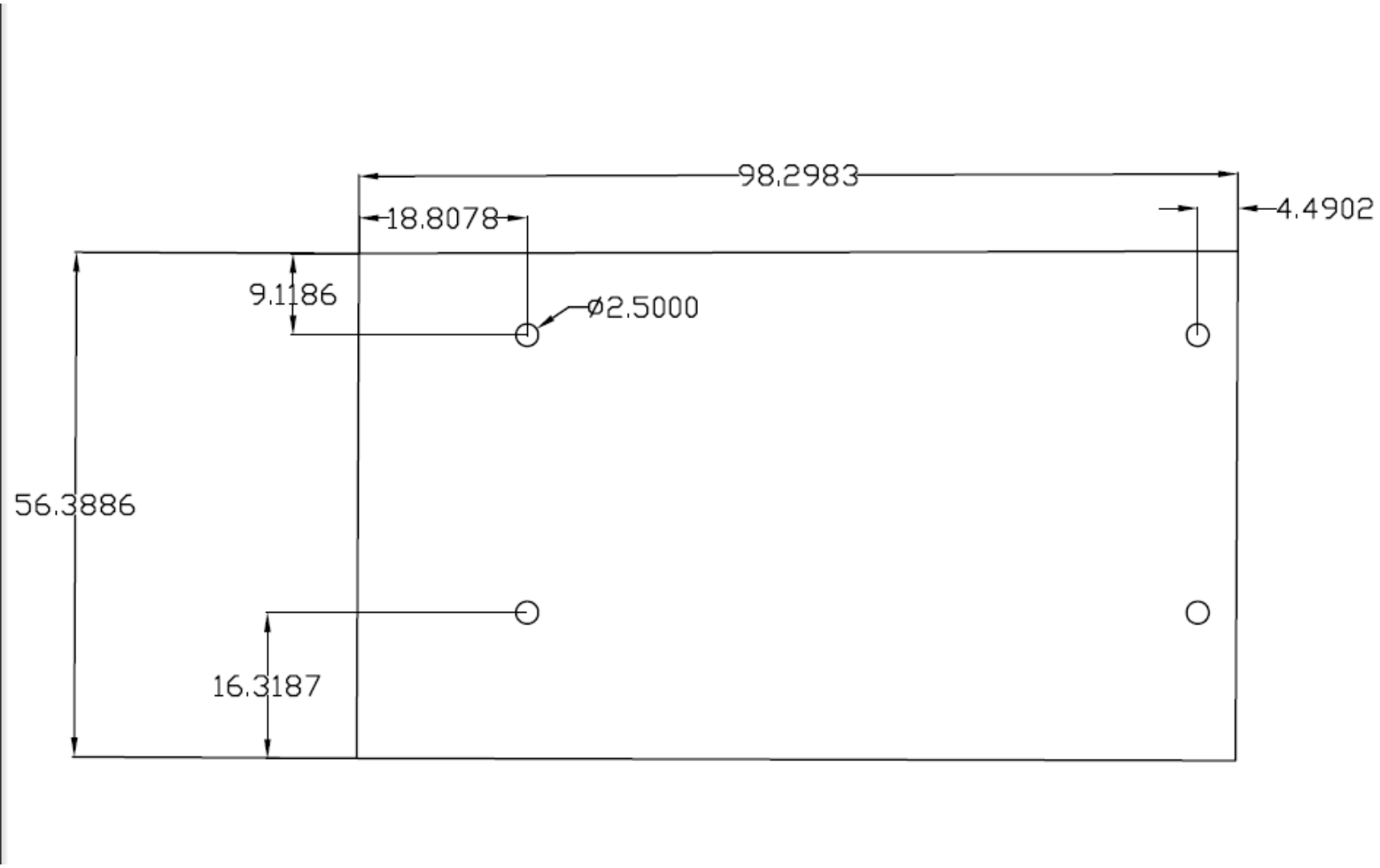
Zgodnie ze wzorem:

$R_{\theta JA} = \frac{(T_J - T_A)}{P}$ · $T_j = T_A + R_{\theta JA} * P = 50℃ + 92\frac{℃}{W} * 217mW = 50℃ + 19.97℃ \approx 70℃$

Zatem obliczona temperatura złącza jest dużo mniejsza od maksymalnej, wiec stosowanie radiatora jest zbędne.

Mozaiki druku i rozmieszczenie elementów:

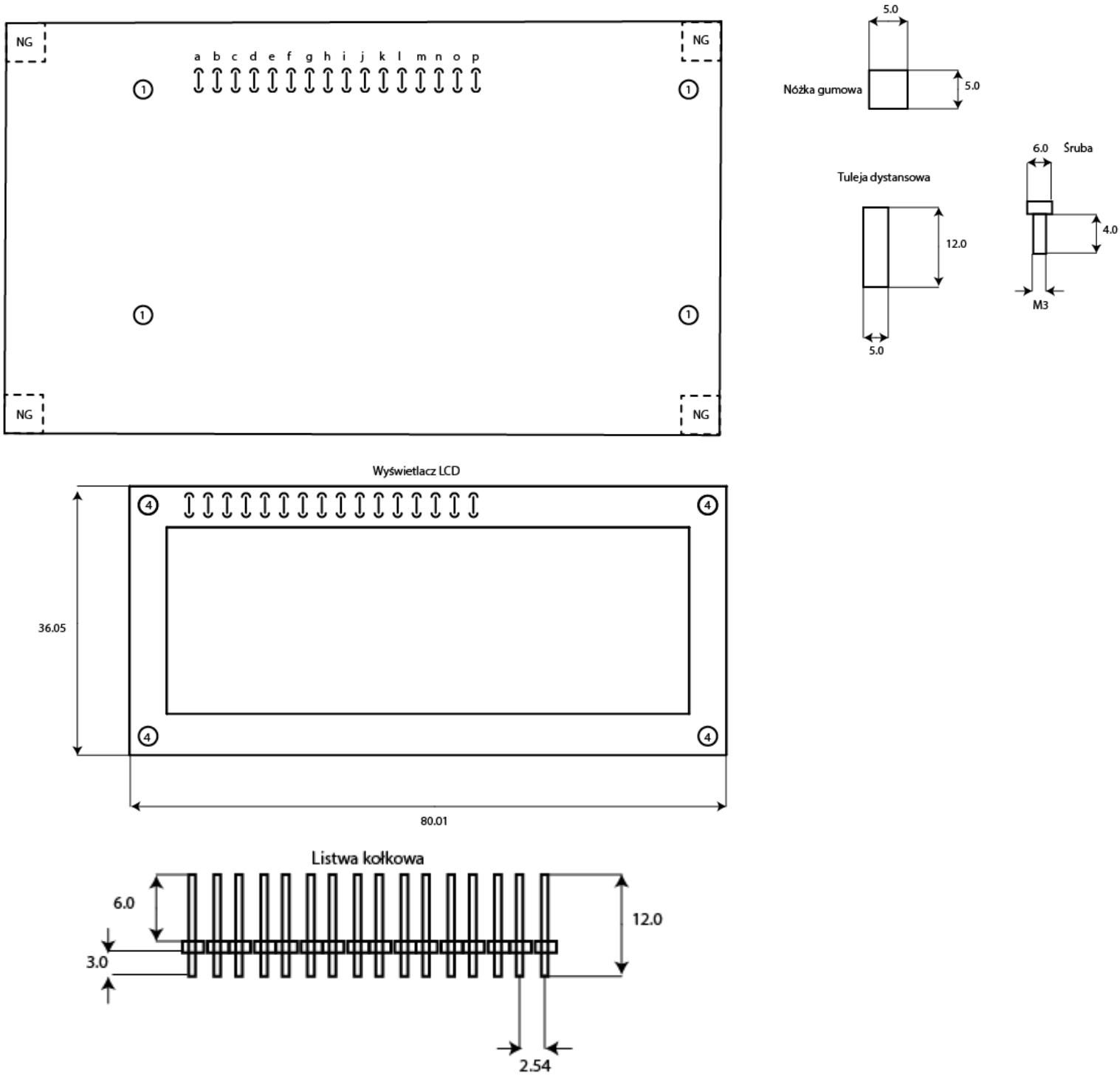
Wymiary płytki drukowanej:



Rysunek 4: Wymiary płytki oraz rozmieszczenie otworów montażowych w milimetrach.

Źródło: Opracowanie własne.

Rysunki mechaniczne:



Rysunek 5: Rysunki mechaniczne.

Źródło: Opracowanie własne.

Gdzie:

NG - docelowe miejsce przyklejenia nóżek gumowych na warstwie dolnej,

1 - miejsce na tuleje dystansowe na stronie górnej oraz na śruby od strony warstwy dolnej,

4 - miejsce na śruby mocujące wyświetlacz do tulei dystansowych,

a-p - miejsca na przylutowanie listwy kolkowej.

Opis rozwiązań konstrukcyjnych:

Płytkę wykonano na laminacie jednostronnym, z powodów czasowych oraz ograniczeń w sprzęcie amatorskim. Docelowo zastąpi go laminat dwustronny światłoczuły.

Lista elementów:

Tabela 1: Lista elementów.

Źródło: Opracowanie własne.

Lp.	Oznaczenie na schemacie	Element	Typ obudowy /montaż	Ilość	Wskazówki do zakupu
Układy scalone					
1	IC1	Mikrokontroler ATMEL ATmega8A-PU	PDIP	1	
2		Wyświetlacz LCD 16x2 DEM 16226 SYH-LY	THT	1	
3	IC4	Konwerter TTL<->RS422 MAXINTEGRATED MAX488CSA+	8-SO	1	
4	Q1	Rezonator kwarcowy IQD 3.686400MHz HC49/4H 50/50/-40 to 85C/10 FUND	SMD		
Elementy pasywne					
6	R6,R5	Rezystor YAEGO 120Ω	1206	2	
7	R3	Rezystor YAEGO 82Ω	1206	1	
8	R2	Rezystor YAEGO 680Ω	1206	1	
9	R7	Rezystor ROYAL OHM 1/4W 10kΩ	THT	1	
10	C2,C3	Kondensator ceramiczny SAMSUNG CL31C220JBCNNNC 22 pF 50V	1206	2	CL31C220JBCNNNC
11	C10,C8,C6,C7,C4,C1	Kondensator ceramiczny KEMET C1206C104J5RAC7800 100 nF 50V	1206	6	C1206C104J5RAC
12	C5	Kondensator elektrolityczny radialny 220uF 25V PANASONIC FC-A EEUFC1E221B x1	THT	1	

13	C9	Kondensator elektrolityczny radialny 47uF 16V YAEGO S5016M0047BZF-0605	THT	1	
14	R1	Potencjometr montażowy liniowy, leżący, jednoobrotowy 10k SR PASSIVES RKT6V-10K	THT	1	
Inne					
15		Listwa kołkowa wraz z zworą CONNFLY DS1027-01-2AB801	THT	2	JUMPER-KPL
21		Listwa kołkowa TE CONNECTIVITY 4-103321-8	THT	1	4-103321-8
16		Zasilacz impulsowy 12V 100mA ESPE ZSI12/0.1		1	
17	LUMBERG_NEB21R	Gniazdo zasilające DC 5.5mmx2.1mm LUMBERG NEB 21 R	THT	1	
18		Wtyk zasilający DC 5.5mmx2.1mm LUMBERG NES/J 21 GRAU		1	
5	LL-304ID2E	DIODA LED KINGBRIGHT L-934IT	THT	1	L-934IT
Elementy mechaniczne					
19		Tulejka dystansowa mosiężna, gwintowana M3 12mm TDYSFF-M3/12		4	TDYSFF-M3/12
20		Śruba M3 4mm stalowa, Ocynkowana BOSSARD B3X4/BN3334		8	B3X4/BN3334
22		Laminat 1 warstwowy FR4; 0,6mm; L:160mm; W:100mm; Pokrycie: miedź 35um		1	LAM100X160E0.6

Uwagi montażowe:

Zalecana kolejność przy montażu wyświetlacza to:

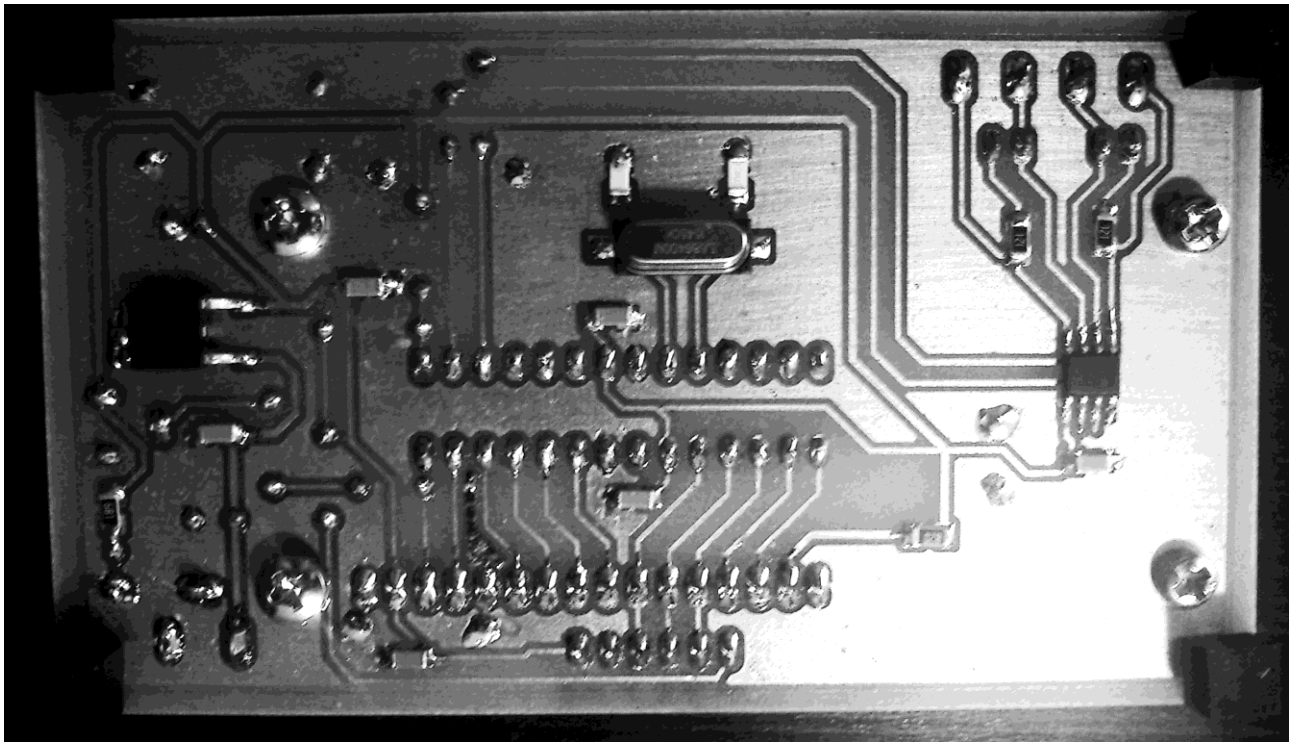
1) Przylutowanie listwy kołkowej 16 pinowej na wskazane miejsca na rysunku mechanicznym(a...p) od strony dolnej,
2) Zamocowanie tulei dystansowych na stronie górnej za pomocą 4 śrub wkręcanych na stronie dolnej,
3) Dokręcenie wyświetlacza za pomocą kolejnych 4 śrub do zamontowanych wcześniej tulei dystansowych,
4) Przylutowanie złącz sygnałowych wyświetlacza do złącz kołkowych a...f.

Opis uruchamiania projektu:

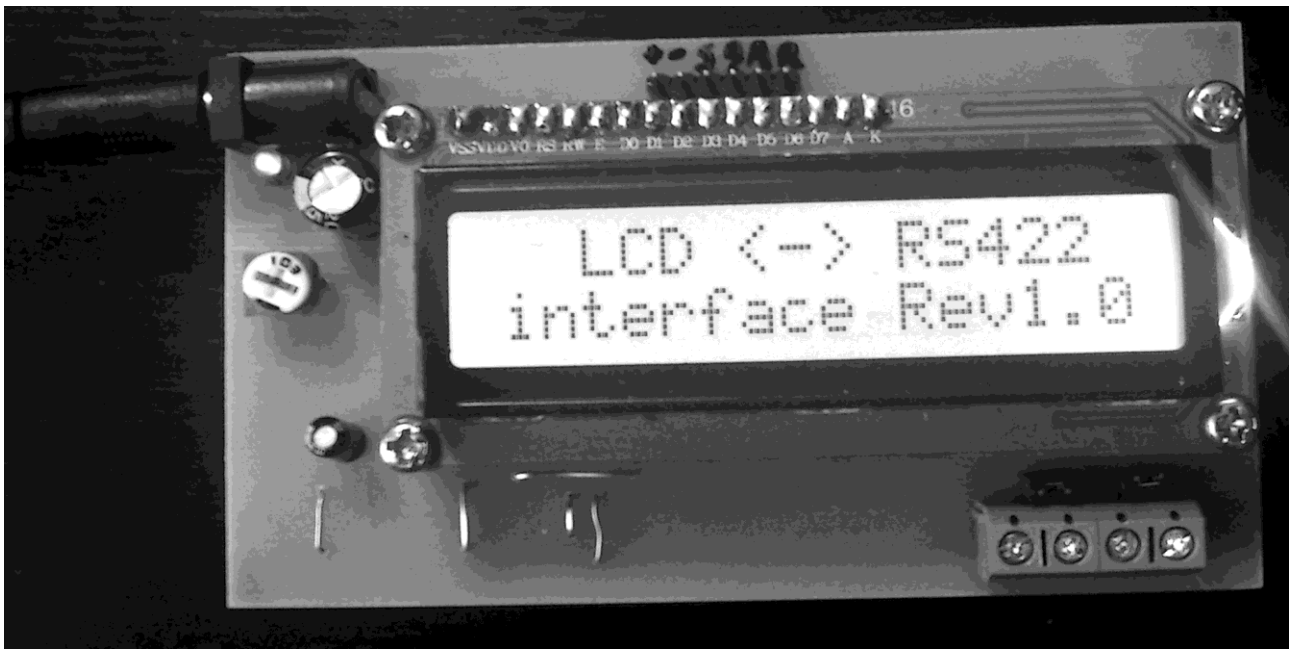
By uruchomić urządzenie należy podłączyć wtyczkę zasilająca do gniazda, poczekać do zaniku ekranu powitalnego, gdy użytkownik zobaczy migający znak zachęty może przejść do wysyłania poleceń.
By komunikacja przebiegała prawidłowo na urządzeniu nadawczym należy ustawić następujące parametry komunikacji:

- Baud rate - 115200kbit/s,
- Number of data bits - 8,
- Stop bits - 2,
- Parity - None.

Fotografia modelu:



Rysunek 6: Fotografia warstwy dolnej modelu.



Rysunek 7: Fotografia górnej części modelu.

Oprogramowanie:

Opis funkcji programu:

Algorytm:

```
1  Inicjalizuj wyświetlacz i moduł komunikacji USART.
2  Wyświetl ekran powitalny
3  Włącz przerwania globalne i USART.
4
5  Dopóki
6    Dopóki jest znak w buforze cyklicznym
7        Pobierz znak i wpisz do tablicy
8        Jeżeli aktualny znak jest znakiem kończącym bądź tablica jest pełna
9            Ustaw flagę sprawdzania tablicy
10           Opóść pętle
11       Przesuń wskaźnik tablicy
12   Jeżeli jest ustawiona flaga sprawdzania tablicy
13       Jeżeli pierwszy znak to znak kończący i drugi znak jest identyfikatorem tego urządzenia
14           Jeżeli tablica[3]==MOVE
15               Przesuń kursor wyświetlacza na zadaną pozycję
16           Jeżeli tablica[3]==PRINT
17               Wyświetl ciąg znaków na ekranie
18           Jeżeli tablica[3]==COMMAND
19               Wykonaj zadaną komendę
20           Jeżeli tablica[3]==UNKNOW
21               Przerwij
22       Resetuj flagę sprawdzania tablicy
23       Ustaw wskaźnik tablicy na pozycję zerową
```

Rysunek 8: Algorytm w postaci pseudokodu.

Źródło: Opracowanie własne.

Listing programu z komentarzami:

‘LCD.c’

```
 1 #include "lcd.h"
 2
 3 static inline void LCDSendData(uint8_t data)
 4 {
 5         if(data&(1<<0)) PORT(LCD_D0PORT) |=(1<<LCD_D0);
 6         else PORT(LCD_D0PORT) &=~(1<<LCD_D0);
 7         if(data&(1<<1)) PORT(LCD_D1PORT) |=(1<<LCD_D1);
 8         else PORT(LCD_D1PORT) &=~(1<<LCD_D1);
 9         if(data&(1<<2)) PORT(LCD_D2PORT) |=(1<<LCD_D2);
10         else PORT(LCD_D2PORT) &=~(1<<LCD_D2);
11         if(data&(1<<3)) PORT(LCD_D3PORT) |=(1<<LCD_D3);
12         else PORT(LCD_D3PORT) &=~(1<<LCD_D3);
13         if(data&(1<<4)) PORT(LCD_D4PORT) |=(1<<LCD_D4);
14         else PORT(LCD_D4PORT) &=~(1<<LCD_D4);
15         if(data&(1<<5)) PORT(LCD_D5PORT) |=(1<<LCD_D5);
16         else PORT(LCD_D5PORT) &=~(1<<LCD_D5);
17         if(data&(1<<6)) PORT(LCD_D6PORT) |=(1<<LCD_D6);
18         else PORT(LCD_D6PORT) &=~(1<<LCD_D6);
19         if(data&(1<<7)) PORT(LCD_D7PORT) |=(1<<LCD_D7);
20         else PORT(LCD_D7PORT) &=~(1<<LCD_D7);
21 }
```

W powyższej funkcji wyłuskujemy pojedyncze bity zmiennej data i wystawiamy na odpowiednie porty mikrokontrolera podłączone do wyświetlacza.

```
22 void LCDWriteByte(unsigned char _data)
23 {
24
25 #if USE_RW ==1
26         CLR_RW;
27 #endif
28         SET_E;
29         LCDSendData(_data);
30         CLR_E;
31 #if USE_RW ==1
32         //not implemented
33 #else
34         _delay_us(120);
35 #endif
36 }
```

Przed każdym zapisem do pamięci wyświetlacza należy ustawić linie E, a po wysłaniu

```
37 void LCDWriteCmd(uint8_t cmd)
38 {
39         CLR_RS;
40         LCDWriteByte(cmd);
41 }
42


48 void LCDClr(void)
49 {
50         LCDWriteCmd(LCDC_CLS);
51         #if USE_RW == 0
52                 _delay_ms(4.9);
53         #endif
54 }
```

```
55 void LCDHome(void)
56 {
57         LCDWriteCmd(LCDC_CLS|LCDC_HOME);
58         #if USE_RW==0
59                 _delay_ms(4.9);
60         #endif
61 }
62
63 void LCDCursorOn(void)
64 {
65         LCDWriteCmd(LCDC_DISPLAYON|LCDC_CURSORON);
66 }
67 void LCDCursorOff(void)
68 {
69         LCDWriteCmd(LCDC_DISPLAYON);
70 }
71
72 void LCDBlinkOn(void)
73 {
74         LCDWriteCmd(LCDC_DISPLAYON|LCDC_CURSORON|LCDC_BLINKON);
75 }
76 void LCDBlinkOFF(void)
77 {
78         LCDWriteCmd(LCDC_DISPLAYON);
79 }
80
81 void LCDStr(char * str)
82 {
83         while(*str) LCDWriteData(*str++);
84 }
85 void LCDPrintChar(char data)
86 {
87         LCDWriteData(data);
88 }
```

Funkcje odpowiednio do przesyłu ciągu znaków zakończonego bajtem NULL oraz pojedynczych bajtów(znaków).

```
89 void LCDGotoXY(uint8_t x, uint8_t y)
90 {
91         uint8_t DDRAMAddr;
92         // remap lines into proper order
93         switch(y)
94         {
95         case 0: DDRAMAddr = LCD_LINE1+x; break;
96         case 1: DDRAMAddr = LCD_LINE2+x; break;
97         case 2: DDRAMAddr = LCD_LINE3+x; break;
98         case 3: DDRAMAddr = LCD_LINE4+x; break;
99         default: DDRAMAddr = LCD_LINE1+x;
100        }
101        // set data address
102        LCDWriteCmd(LCDC_DDRAM | DDRAMAddr);
103 }
```

Pozycje x,y przekształcamy na pozycje wskaźnika w pamięci wyświetlacza, gdzie LCD_LINEn to stała zdefiniowana w pliku nagłówkowym.

```
105 void LCDInit()
106 {
107         /*
108         Setting Mega ports to output
109         */
110         DDR(LCD_D7PORT) |= (1<<LCD_D7);
111         DDR(LCD_D6PORT) |= (1<<LCD_D6);
112         DDR(LCD_D5PORT) |= (1<<LCD_D5);
113         DDR(LCD_D4PORT) |= (1<<LCD_D4);
114         DDR(LCD_D3PORT) |= (1<<LCD_D3);
115         DDR(LCD_D2PORT) |= (1<<LCD_D2);
116         DDR(LCD_D1PORT) |= (1<<LCD_D1);
117         DDR(LCD_D0PORT) |= (1<<LCD_D0);
118         DDR(LCD_RSPORT) |= (1<<LCD_RS);
119         DDR(LCD_EPORT) |= (1<<LCD_E);
120         #if USE_RW==1
121         DDR(LCD_RWPORT) |=(1<<LCD_RW);
122         #endif
123
124
125         //8bit mode inint according to datasheet
126         _delay_ms(120);
127         LCDWriteCmd(MODE_8BIT);
128         _delay_ms(5);
129         LCDWriteCmd(MODE_8BIT);
130         _delay_ms(1);
131         LCDWriteCmd(MODE_8BIT);
132         _delay_ms(1);
133         LCDWriteCmd(LCDC_FUNCTIONSET); //0011NF**
134         LCDWriteCmd(LCDC_DISPLAYOFF); //turn off the display
135         LCDClr(); //clear
136         LCDWriteCmd(LCDC_ENTRYMODE);
137         LCDWriteCmd(LCDC_DISPLAYON|LCDC_BLINKON|LCDC_CURSORON);
138 }
```

Procedura inicjalizacyjna zgodna z datasheet sterownika HD44780.

‘terminal.c’

```
1 #define BAUD 115200 //USART speed in bps
2 #define _UBRR (F_CPU/16/BAUD-1) //calculate UBRR register value for given baud and CPU clock
3
4 #include "lcd.h"
5 #include <avr/io.h>
6 #include <util/delay.h>
7 #include <avr/interrupt.h>
8 #include <stdlib.h>
9
10 #define DEVICE_ID '1'
11 #define PACKET_SIZE 36
12 #define PACKET_DELIMITER 0x04 //end of trnasmission char
13 #define USART_RX_BUF_SIZE 64 //buffer size
14 #define USART_RX_BUF_MASK ( USART_RX_BUF_SIZE -1)
15 #define PRINT 'p'
16 #define MOVE 'm'
17 #define COMMAND 'c'
```

Linie 1-8 -> definicje preprocesora parametrów komunikacji oraz dołączenie plików nagłówkowych.

Linia 10 -> identyfikator urządzenia, gdy w sieci pracują więcej niż 1 odbiornik.

Linia 11 -> definicja maksymalnego rozmiaru pojedynczego pakietu.

Linia 13 -> definicja rozmiaru bufora odbiorczego w bajtach.

Linia 14 -> definicja maski dla bufora, wykorzystywana do ograniczania wpisywania tylko do 64 znaków.

Linie 15-17 -> identyfikatory typów komand, odpowiednio: wypisywanie ciągu znaków, przenoszenie kursora na pozycję (X,Y), oraz komend serwisowych(czyszczenie ekranu, rodzaj kursora itp.).

```
19 volatile char USART_RxBuf[USART_RX_BUF_SIZE];
20 volatile uint8_t USART_RxHead;
21 volatile uint8_t USART_RxTail;
22 //function declarations
23 char USARTGetc(void);
24 void USARTInit(uint32_t _ubrr,uint8_t stopBits);
25 void EvaluatePosition(char[]);
26 void EvaluateCommand(char[]);
27 void Print(char[]);
```

Linie 19-27 -> deklaracje zmiennych globalnych oraz funkcji, tablica USART_RxBuf zdeklarowana jako volatile, ponieważ jest zmieniana w przerwaniu.

```
28 //
29 int main()
30 {
31     LCDInit(); // init LCD
32     LCDStr("  LCD <-> RS422"); //splash screen
33     LCDGotoXY(0,1);
34     LCDStr("interface Rev1.0");
35     _delay_ms(2000);
36     LCDClr();
37     LCDGotoXY(0,0);
38     USARTInit(_UBRR,2); //init USART
39     UCSRB|=(1<<RXCIE); //read interrupt on
40     sei(); // global interrupts on
41     char command[PACKET_SIZE];
42     int i=0;
43     int checkCommand=0;
```

Linie 31-37 - inicjalizacja wyświetlacza oraz wyświetlenie ekranu startowego.

Linie 38-40 - inicjalizacja USART oraz załączenie przerwań przy odbiorze znaku oraz przerwań globalnych.

Linie 41-43 - definicje zmiennych pomocniczych.

```
44      while(1)
45      {
46          while(USART_RxHead!=USART_RxTail) //something in the buffer
47          {
48              command[i]=USARTGetc(); // get that something
49              if((command[i] == PACKET_DELIMITER && i!=0) || (i>=PACKET_SIZE-1)) //if 'end of the command character'
50              //in the buffer or exceeded packet size
51              {
52                  checkCommand=1; //inform rest of the code that commad awaits
53                  break; //escape while loop
54
55              }
56              i++; //shift to the next empty slot in the command array
57          }
58
59          if(checkCommand) //evaluate command
60          {
61              if(command[0]==PACKET_DELIMITER && command[1]==DEVICE_ID) //check whether good formating is used
62              //, where @-start_byte, 1 - identifier
63              {
64
65                  switch(command[2])
66                  {
67                      case MOVE: EvaluatePosition(command); break;
68                      case PRINT: Print(command); break;
69                      case COMMAND: EvaluateCommand(command); break;
70                      default: break;
71                  }
72              }
73              checkCommand=0; //command evaluated
74              i=0; //flush command array by setting index to 1'st member
75
76          }
77      }
78  }
79
```

Linia 44 - 78 -> pętla główna

Reszta jest dobrze udokumentowana.

```
80  ISR(USART_RXC_vect)
81  {
82      uint8_t tmp_head;
83      char data;
84      data=UDR;
85      UDR=data;
86      tmp_head=(USART_RxHead +1)&USART_RX_BUF_MASK;
87      if(tmp_head==USART_RxTail)
88      {
89            //if buffer overflow then overwrite oldest data
90      }
91      else
92      {
93          USART_RxHead=tmp_head;
94          USART_RxBuf[tmp_head]=data;
95      }
96  }
97
```

Interrupt service routine dla przerwania odbioru.

```
 98 void USARTInit(uint32_t _ubrr,uint8_t stopBits)
 99 {
100     UBRRH= (uint8_t)(_ubrr>>8); // four MSBs
101     UBRRL= (uint8_t) _ubrr; // eight LSBs
102     UCSRB=(1<<RXEN); //enable  receiver
103     UCSRC=(1<<USBS)|(1<<URSEL)|(3<<UCSZ0); // (two stop bits)(URSEL set to one while
104    // writing to UCSRC)(8 bit character size)
105     if(stopBits!=2)
106     UCSRC&=~(1<USBS);
107 }
108
```

Usprawniona definicja funkcji inicjalizującej na bazie dokumentacji atmel' a.

Dobrze udokumentowana.

Linie 105-106 -> linie odpowiedzialne za wybranie ilości bitów stopu. Dla każej liczby różnej od 2 ustawiana jest wartość domyślna(1 bit stopu).

```
109 char USARTGetc(void)
110 {
111     if(USART_RxHead==USART_RxTail) return 0; //if no characters awaiting
112     USART_RxTail = (USART_RxTail+1) & USART_RX_BUF_MASK;
113     return USART_RxBuf[USART_RxTail]; //return one character from the buffer
114 }
```

```
115 void EvaluatePosition(char command[])
116 {
117
118     uint8_t position=5;
119     uint8_t x = atoi(&command[3]);
120     if(x>9)
121         position=6;
122     uint8_t y = atoi(&command[position]);
123     LCDGotoXY(x,y);
124
125 }
```

```
127 void EvaluateCommand(char command[])
128 {
129     switch(command[3])
130         {
131             case '0': LCDClr(); break;
132             case '1': LCDBlinkOn(); break;
133             case '2': LCDBlinkOFF(); break;
134             case '3': LCDCursorOn(); break;
135             case '4': LCDCursorOff(); break;
136             default: break;
137         }
138 }
```

```
139 void Print(char command[])
140 {
141     uint8_t z = 3;
142     while(command[z]!=PACKET_DELIMITER)
143     LCDWriteData(command[z++]);
144 }
145
```

‘lcd.h’

```
1 #define F_CPU 3686400UL
2 #include <avr/io.h>
3 #include <util/delay.h>
4 #include <stdio.h>
5 // definicje rozmiarÃłw wyÅ›wietlacza
6 #define LCD_Y 2
7 #define LCD_X 16
8 #define USE_RW 0 //rw do masy,USE_RW=1 nie zaimplementowane
```

Linie 1-4 -> definicje taktowania procesora(w Hz) oraz niezbędnych bibliotek.

Należy zauważyć iż biblioteka w dniu opracowywania projektu posiadała niezaimplementowaną obsługę trybu 4 bitowego oraz niepełne wsparcie dla portu RW wyświetlacza.

```
 9 //definicje wykorzystywanych portÃłw (data)
10 #define LCD_D7PORT B
11 #define LCD_D7 1
12 #define LCD_D6PORT B
13 #define LCD_D6 2
14 #define LCD_D5PORT B
15 #define LCD_D5 3
16 #define LCD_D4PORT B
17 #define LCD_D4 4
18 #define LCD_D3PORT B
19 #define LCD_D3 5
20 #define LCD_D2PORT C
21 #define LCD_D2 0
22 #define LCD_D1PORT C
23 #define LCD_D1 1
24 #define LCD_D0PORT C
25 #define LCD_D0 2
26 //definicje wykorzystywanych portÃłw(cmd)
27 #define LCD_RSPORT C
28 #define LCD_RS 4
29
30 #define LCD_RWPORT B
31 #define LCD_RW 2
32
33 #define LCD_EPORT C
34 #define LCD_E 3
```

Linie 10-34 -> definicje pinów i portów przeznaczonych do komunikacji z wyświetlaczem.

```
35 //adresy w pamiÄ™ci DDRAM
36 #if ( (LCD_Y == 4)&&(LCD_X==20))
37 #define LCD_LINE1 0x00
38 #define LCD_LINE2 0x28
39 #define LCD_LINE3 0x14
40 #define LCD_LINE4 0x54
41 #else
42 #define LCD_LINE1 0x00
43 #define LCD_LINE2 0x40
44 #define LCD_LINE3 0x10
45 #define LCD_LINE4 0x50
46 #endif
```

Linie 35-46 -> Adresy w pamięci wyświetlacza odpowiadające kolejnym liniom.

```
48 //makra upraszczajace dostep do portÄŁw
49 #define PORT(x) SPORT(x)
50 #define SPORT(x) (PORT##x)
51
52 #define PIN(x) SPIN(x)
53 #define SPIN(x) (PIN##x)
54
55 #define DDR(x) SDDR(x)
56 #define SDDR(x) (DDR##x)
57
```

Linie 48-46 -> Źródło: Mikrokontrolery AVR Język C Podstawy programowania wyd. II - M. Kardaś.

```
59 //makro operacji na sygnalach sterujacych RS, RW, E
60
61 #define SET_RS PORT(LCD_RSPORT) |= (1<<LCD_RS)
62 #define CLR_RS PORT(LCD_RSPORT) &=  ~(1<<LCD_RS)
63
64 #define SET_RW PORT(LCD_RWPORT) |= (1<<LCD_RW)
65 #define CLR_RW PORT(LCD_RWPORT) &=  ~(1<<LCD_RW)
66
67 #define SET_E PORT(LCD_EPORT) |= (1<<LCD_E)
68 #define CLR_E PORT(LCD_EPORT) &=  ~(1<<LCD_E)
69 // commands
70 #define MODE_8BIT 0x30
71 #define LCDC_DISPLAYOFF 0x08
72 #define LCDC_CLS 0x01
73 #define LCDC_DISPLAYON 0x0C
74 #define LCDC_FUNCTIONSET 0x38 //proprer setiing of the display -> D5|8bits|2rows
75 #define LCDC_ENTRYMODE 0x06 // setting the way the characters are written to
76 //display(cursor form L to R, display not shifting)
77 //funkcja do przesyŁ,u danych na wyswietlacz
78 #define LCDC_DISPLAYONOFF 0x0C
79 #define LCDC_CURSORON 0x02
80 #define LCDC_BLINKON 0x01
81 #define LCDC_HOME 0x02
82 #define LCDC_DDRAM 0x80
```

Linie 70-82 -> kombinacje bitowe w trybie hex'a na podstawie dokumentacji. Nie wszystkie up-to-date.

```
83 static inline void LCDSendData(uint8_t data);
84 void LCDWriteByte(unsigned char _data);
85 void LCDWriteCmd(uint8_t cmd);
86 void LCDWriteData(uint8_t data);
87 void LCDClr(void);
88 void LCDHome(void);
89 void LCDCursorOn(void);
90 void LCDCursorOff(void);
91 void LCDBlinkOn(void);
92 void LCDBlinkOFF(void);
93 void LCDStr(char * str);
94 void LCDGotoXY(uint8_t x, uint8_t y);
95 void LCDInit();
96 void LCDPrintChar(char data);
97
```

Deklaracje funkcji.

Bibliografia:

- Atmel, 8159-8-bit-avr-microcontroller-atmega8a_datasheet.pdf,
- Texas Instruments, RS-422 and RS-485 Standards Overview and System Configurations,
- MAXIM, MAX488 datasheet(MAX1487-MAX491.pdf),
- HITACHI, HD44780U datasheet(Rev. 0.0),
- Kardaś Mirosław ‘Mikrokontrolery AVR Język C. Podstawy programowania.’