Okrągły wyświetlacz OLED (2)

W pierwszej części artykułu opublikowanej w rubryce „Kurs” zaprezentowano sposób obsługi programowej okrągłego wyświetlacza OLED. W części drugiej zamieszczamy opis płytki ewaluacyjnej, która pozwoli na zrealizowanie wielu interesujących projektów.

Rekomendacje: *zestaw przyda się do wykonania prototypu lub funkcjonującego urządzenia typu smartwatch lub innego, o podobnym kształcie.*

Schemat ideowy zestawu ewaluacyjnego pokazano na **rysunku 3**. Jest to system mikroprocesorowy, którego sercem jest niewielki mikrokontroler ATmega88 taktowany wewnętrznym, wysokostabilnym generatorem RC o częstotliwości 8 MHz. Mikrokontroler ten, podobnie jak wyświetlacz, jest zasilany napięciem 3 V pozyskiwanym ze stabilizatora liniowego LDO typu TC-1015-3.0VCT. Całe urządzenie, w zamyśle przenośne, zaprojektowano jako zasilane

Listing 1. Funkcje odpowiedzialne za wysyłanie rozkazów lub danych obrazu

```
void writeCommand(uint8_t Command)
{
    RESET_DC;  //DC=0 wskazuje na rozkaz sterujący
    RESET_WR;
    DATA_PORT = Command;
    SET_WR;  //Zapis danych następuje na rosnącym zboczu sygnału WR
    SET_DC;  //DC domyślnie ustawiine, by przyspieszyć transfer danych obrazu
}

void writeData(uint8_t Data)
{
    RESET_WR;
    DATA_PORT = Data;
    SET_WR;  //Zapis danych następuje na rosnącym zboczu sygnału WR
}
```

Listing 2. Funkcja inicjalizacyjna sterownika SSD1327 wyświetlacza WEO128128

```
void initOLED(void)
{
    //Port danych jako wyjściowy, równy 0x00
    DATA_DDR = 0xFF;
    //Port sterujący jako wyjściowy, równy 0xFF
    CTRL_PORT |= (1<<WR_PIN)|(1<<RD_PIN)|(1<<DC_PIN)|(1<<RST_PIN);
    CTRL_DDR |= (1<<WR_PIN)|(1<<RD_PIN)|(1<<DC_PIN)|(1<<RST_PIN);
    RESET_RST;  //Zerowanie sprzętowe sterownika SSD1327
    _delay_ms(10);
    SET_RST;
    _delay_ms(10);
    writeCommand(SET_COLUMN_ADDR);
    writeCommand(0x00);  //Start Column Address
    writeCommand((OLED_WIDTH/2)-1);  //End Column Address
    writeCommand(SET_ROW_ADDR);
    writeCommand(0x00);  //Start Row Address
    writeCommand(OLED_HEIGHT-1);  //End Row Address
    writeCommand(SET_CONTRAST_CTRL);
    writeCommand(0x9B);  //Contrast Level
    writeCommand(SET_REMAP);
    writeCommand(COLUMN_ADDR_REMAP|COM_REMAP|COM_SPLIT_ODD_EVEN);
    writeCommand(SET_DISP_START_LINE);  //Set Display Start Line
    writeCommand(0x00);
    writeCommand(SET_DISP_OFFSET);
    writeCommand(0x00);
    writeCommand(SET_DISP_MODE_NORMAL);
    writeCommand(SET_MULTIPLEX_RATIO);
    writeCommand(0x7F);       //Multiplex = 128
    writeCommand(SET_FUNC_SELECTION_A);
    writeCommand(ENABLE_INTERNAL_VDD);
    writeCommand(SET_PHASE_LENGTH);
    writeCommand(0xF1);  //105
    writeCommand(SET_DISP_CLOCK);  //Set Display Clock Divide Ratio/Oscillator Frequency
    writeCommand(0x00);
    writeCommand(SET_PRECHARGE_VOLTAGE);  //Set Prechange Voltage
    writeCommand(0x07);  //0.613 x VCC
    writeCommand(SET_VCOMH_VOLTAGE);  //Set VCOMH Voltage
    writeCommand(0x07);  //0.86 x VCC
    writeCommand(SET_SEC_PRECHARGE_PERIOD);  //Set Second Pre-charge period
    writeCommand(0x0F);
    writeCommand(SET_FUNC_SELECTION_B);
    writeCommand(INTERNAL_VSL|ENABLE_SECOND_PRECHARGE);
    writeCommand(SET_GRAY_SCALE_TABLE);
    writeCommand(0x01);  //A[2:0]
    writeCommand(0x01);  //B[2:0]
    writeCommand(0x10);  //B[6:4]
    writeCommand(0x01);  //C[2:0]
    writeCommand(0x10);  //C[6:4]
    writeCommand(0x01);  //D[2:0]
    writeCommand(0x10);  //D[6:4]
    writeCommand(0x01);  //E[2:0]
    writeCommand(0x10);  //E[6:4]
    writeCommand(0x01);  //F[2:0]
    writeCommand(0x10);  //F[6:4]
    writeCommand(0x01);  //G[2:0]
    writeCommand(0x10);  //G[6:4]
    writeCommand(0x01);  //H[2:0]
    writeCommand(0x10);  //H[6:4]
    writeCommand(SET_DISPLAY_ON);  //Set Display On
}
```

DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl
USER: 92822, PASS: 37euo8qf

Podstawowe informacje:

- Rozdzielczość 128×128 pikseli.
- Możliwość wyświetlania 16 poziomów szarości, co poprawia estetykę wyświetlanych piktogramów.
- Średnica obszaru aktywnego 30 mm.
- Wymiary modułu 37 mm×41 mm×2 mm (tylko 2 mm grubości!).
- Wbudowany kontroler typu SSD1327ZB.
- Zasilanie 3 V.
- Szeroki wybór dostępnych interfejsów sterujących: równoległy 6800/8080, szeregowe I^2C oraz SPI.
- Mały pobór mocy.
- Dostępne kolory świecenia: żółty, biały, niebieski.
- Długi czas bezawaryjnej pracy (rzędu 50 tys. godzin)
- Jasność 100 cd/m^2.
- Kontrast 2000:1.
- Szeroki kąt patrzenia w pionie i poziomie rzędu 160°.

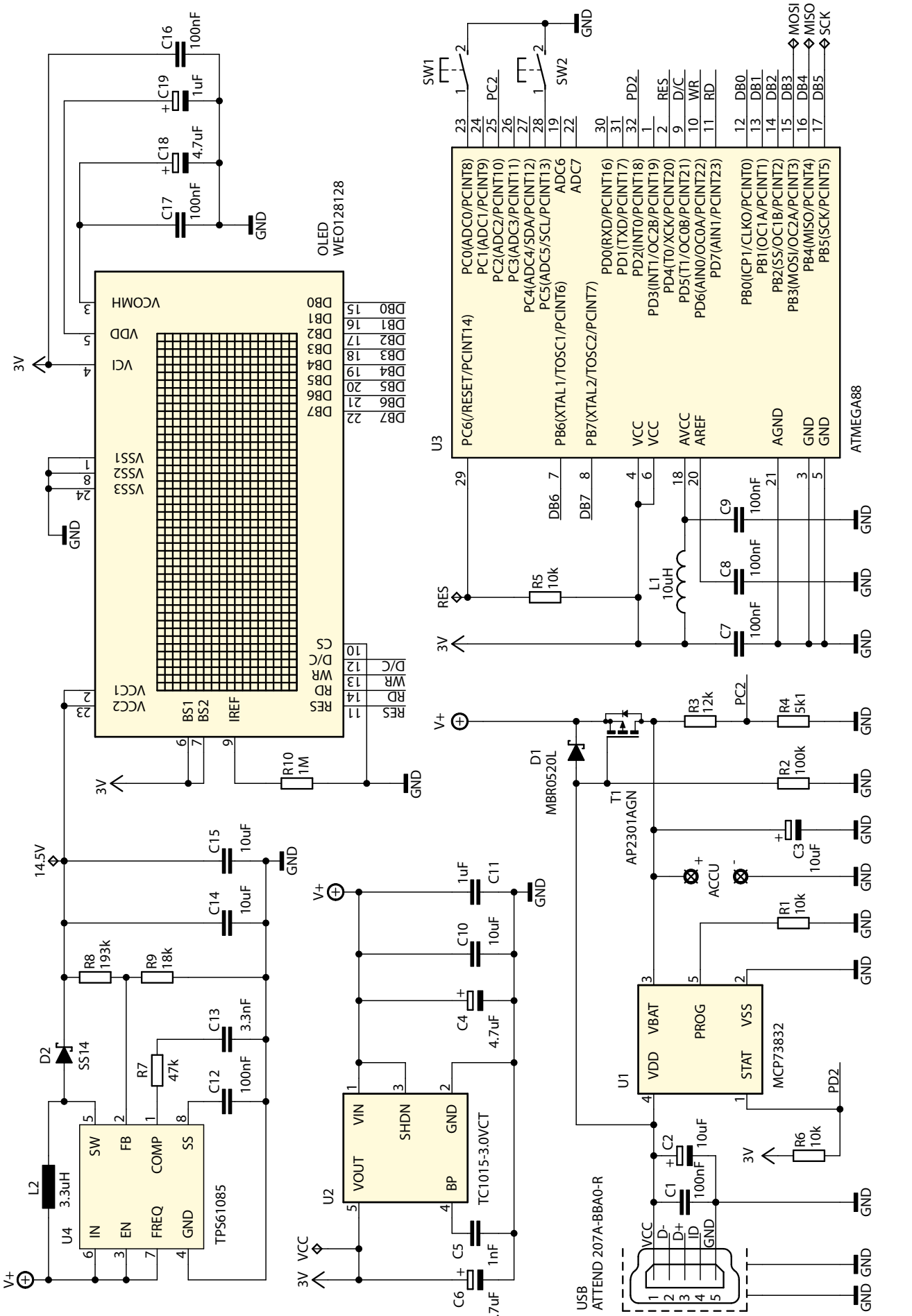
Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5525	Watch – zegarek naręczny (EP 12/2015-01/2016)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] płytka drukowana bez elementów i dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają następujące dodatkowe wersje:
- wersja [**A+**] płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***



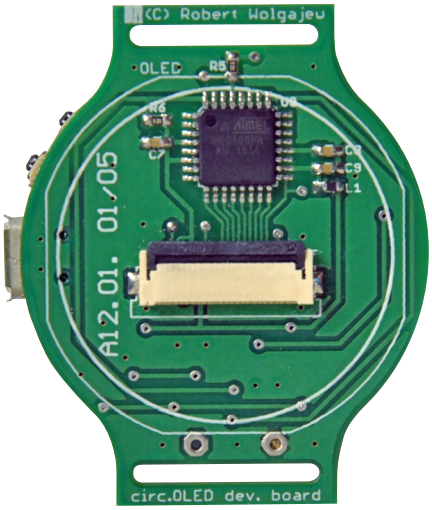
Rysunek 3. Schemat ideowy zestawu ewaluacyjnego dla okrągłych wyświetlaczy OLED z rodziny WEO128128B

z wbudowanego akumulatora litowego (zaciski ACCU), dla którego przewidziano specjalizowaną sekcję ładowania z wykorzystaniem napięcia pozyskiwanego z umieszczonego na płytce złącza micro-USB. Sekcję, o której mowa, zbudowano z użyciem specjalizowanego układu ładowania akumulatorów litowych typu MCP73832 firmy Microchip. Ten układ idealnie nadaje się do zastosowania w aplikacjach ładowarek, ponieważ w pełni automatycznie nadzoruje proces ładowania, wybierając odpowiedni tryb ładowania oraz mechanizm kontroli, zaś jedynym zmartwieniem użytkownika jest wybór prądu ładowania szybkiego, którego dokonujemy, dobierając wartość rezystora podłączonego pomiędzy wyprowadzenie PROG a masę. Prąd ten dobieramy według zależności $I_{REG}=1000\ V/R_{PROG}$, gdzie wielkości I_{REG} wyrażono w mA, natomiast R_{PROG} w kΩ.

W zestawie rezystor R_{PROG} (R1) ma wartość 10 kΩ, co ustawia prąd ładowania szybkiego na wartość 100 mA. Nie bez powodu ustawiono tego typu prąd ładowania, wszak należy pamiętać, iż dla wygody do zasilania przewidziano podłączenie go do portu USB komputera lub popularnego zasilacza ze złączem USB, a ten pozwala na maksymalny pobór prądu rzędu 500 mA w trybie high-power, zaś typowo 100 mA.

Komentarza wymaga także opcjonalny układ współdzielenia obciążenia zbudowany przy użyciu tranzystora T1 typu MOSFET z kanałem P, diody Schottky'ego D1 oraz rezystora R2. Dlaczego niezbędna była implementacja tego rodzaju rozwiązania? Otóż układ MCP73832 ma bloków funkcjonalnych odpowiedzialnych za współdzielenie obciążenia, to znaczy, odpowiedzialnych za uwzględnienie w procesie ładowania faktu, że w czasie, gdy układ nadzoruje proces ładowania akumulatora, to tenże akumulator zasila urządzenie, które pobiera z niego prąd. Taka sytuacja, po pierwsze, powoduje w najlepszym wypadku wydłużenie samego procesu ładowania, zaś w skrajnych wypadkach może go zaburzyć czy spowodować, że proces ładowania nigdy nie dobiegnie końca. Aby temu zapobiec, zastosowano prosty „przełącznik" w postaci tranzystora MOSFET, którego bramkę dołączono bezpośrednio do napięcia USB zasilającego ładowarkę. W wypadku obecności napięcia USB (czyli de facto poziomu wysokiego na bramce tranzystora) tranzystor T1 przechodzi w stan wyłączenia, odłączając tym samym ładowany akumulator od obciążenia. W tym samym czasie obciążenie, którym jest urządzenie, zostaje zasilone bezpośrednio z napięcia portu USB, a dokładnie rzecz ujmując, poprzez diodę D1. W razie odłączenia urządzenia od napięcia zasilającego USB, bramka tranzystora T1 zostaje ściągnięta do masy (poprzez rezystor R2), powodując przewodzenie tegoż tranzystora, a więc tym samym zasilenie naszego urządzenia z akumulatora ACCU. W tym przypadku dioda D1 pełni nieco inną funkcję, a mianowicie zabezpiecza przed przepływem prądu wstecznego tj. z akumulatora w kierunku źródła napięcia zasilającego (USB). W ten prosty sposób zbudowano prosty i w pełni funkcjonalny układ współdzielenia obciążenia, który czasami występuje w innych typach scalonych kontrolerów ładowania produkcji firmy Microchip.

Dodatkowo, wyświetlacz potrzebuje napięcia rzędu 14,5 V zasilającego panel OLED, więc w bloku zasilającym tenże panel wykorzystano nowoczesną przetwornicę typu step-up o oznaczeniu TPS61085 firmy Texas Instruments. Przetwornica ta odznacza się doskonałymi parametrami elektrycznymi: sprawność dochodząca do 93%, szeroki zakres napięcia wejściowego (2,3…6 V), miękki start, wbudowane zabezpieczenia (termiczne, przed spadkiem napięcia), częstotliwość przełączania 1,2 MHz. Dodatkowo układ TPS61085 ma wejście EN (Enable), za którego pomocą możemy wyłączyć przetwornicę, jeśli zajdzie taka potrzeba (np. wygaszenie wyświetlacza) i tym samym ograniczyć pobór prądu ze źródła napięcia zasilającego. W celu uczynienia naszego zestawu jeszcze bardziej uniwersalnym, na schemacie urządzenia dodano dwa, uniwersalnie mikroprzyciski oznaczone SW1/SW2 przeznaczone do dowolnego wykorzystania przez aplikację użytkownika.



Ustawienia fusebitów (ważniejszych)

CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1

Wykaz elementów:

Kondensatory: (obudowy SMD 0603, jeśli nie zaznaczono inaczej)
C1, C7...C9, C12, C16, C17: ceramiczny X7R 100 nF
C2, C3: tantalowy 10 µF/16 V (obudowa typ A/3216-18W)
C4, C6, C18: tantalowy 4,7 µF/16 V (obudowa typ A/3216-18W)
C19: tantalowy 1 µF/16 V (obudowa typ A/3216-18W)
C5: ceramiczny X7R 1 nF
C10, C14, C15: ceramiczny X7R 10 µF (obudowa SMD 0805)
C11: ceramiczny X7R 1 µF
C13: ceramiczny X7R 3,3 nF

Rezystory: (obudowy SMD 0603)
R1, R5, R6: 10 kΩ
R2: 100 kΩ
R3: 12 kΩ
R4: 5.1 kΩ
R7: 47 kΩ
R8: 193 kΩ 1%
R9: 18 kΩ 1%
R10: 1 MΩ

Półprzewodniki:
U1: MCP73832 (obudowa SOT-23-5)
U2: TC1015-3.0VCT (obudowa SOT-23-5)
U3: ATMEGA88 (obudowa TQFP32)
U4: TPS61085 (obudowa TSSOP8)
D1: MBR0520L (obudowa SOD123)
D2: SS14 (obudowa SMA)
T1: AP2301AGN (obudowa SOT23)
OLED: wyświetlacz graficzny OLED typu Winstar WEO128128BWPP3N00000

Inne:
L1: dławik 10 µH (obudowa SMD 0603)
L2: dławik 3,3 µH DLG-0504-3R3 (obudowa DLG-0504)
USB: gniazdo USB-B micro ATTEND 207A-BBA0-R
SW1, SW2: microswitch SMD typu B3U-3000PM
ZIF: gniazdo ZIF (24 pin, raster 0,5 mm, styki górne)
ACCU: akumulator Li-Po 3,7 V/250 mA CELLEVIA BATTERIES L502030 lub podobny

Montaż

Schemat montażowy zestawu ewaluacyjnego pokazano na **rysunku 4**. Zaprojektowano niewielką płytkę drukowaną, która swoim kształtem przypomina zegarek naręczny, dając możliwość zamocowania typowego paska utrzymującego ją na nadgarstku. Z uwagi na fakt, że moduł wyświetlacza OLED jest dołączony do płytki z użyciem gniazda ZIF o bardzo gęstym rastrze (0,5 mm), montaż płytki ewaluacyjnej rozpoczynamy właśnie

Listing 3. Plik nagłówkowy sterownika SSD1327 wyświetlacza WEO128128

```
#define DATA_PORT PORTB
#define DATA_DDR DDRB
#define CTRL_PORT PORTD
#define CTRL_DDR DDRD
#define WR_PIN PD6
#define RD_PIN PD7
#define DC_PIN PD5
#define RST_PIN PD4
#define RESET_DC CTRL_PORT &= ~(1<<DC_PIN)
#define SET_DC CTRL_PORT |= (1<<DC_PIN)
#define RESET_WR CTRL_PORT &= ~(1<<WR_PIN)
#define SET_WR CTRL_PORT |= (1<<WR_PIN)
#define RESET_RST CTRL_PORT &= ~(1<<RST_PIN)
#define SET_RST CTRL_PORT |= (1<<RST_PIN)
//Rozdzielczość ekranu (pozioma i pionowa)
#define OLED_WIDTH 128
#define OLED_HEIGHT 128
//Definicje komend sterujących
#define SET_COLUMN_ADDR 0x15
#define SET_ROW_ADDR 0x75
#define SET_CONTRAST_CTRL 0x81
#define SET_REMAP 0xA0
    #define COLUMN_ADDR_REMAP (1<<0)
    #define NIBBLE_REMAP (1<<1)
    #define HORIZONTAL_ADDR_INCR (0<<2)
    #define VERTICAL_ADDR_INCR (1<<2)
    #define COM_REMAP (1<<4)
    #define COM_SPLIT_ODD_EVEN (1<<6)
#define SET_DISP_START_LINE 0xA1
#define SET_DISP_OFFSET 0xA2
#define SET_DISP_MODE_NORMAL 0xA4
#define SET_DISP_MODE_INVERSE 0xA7
#define SET_MULTIPLEX_RATIO 0xA8
#define SET_FUNC_SELECTION_A 0xAB
    #define SELECT_EXTERNAL_VDD (0<<0)
    #define ENABLE_INTERNAL_VDD (1<<0)
#define SET_DISPLAY_OFF 0xAE
#define SET_DISPLAY_ON 0xAF
#define SET_PHASE_LENGTH 0xB1
#define SET_DISP_CLOCK 0xB3
#define SET_SEC_PRECHARGE_PERIOD 0xB6
#define SET_GRAY_SCALE_TABLE 0xB8
#define SET_PRECHARGE_VOLTAGE 0xBC
#define SET_VCOMH_VOLTAGE 0xBE
#define SET_FUNC_SELECTION_B 0xD5
    #define INTERNAL_VSL (0x60|(0<<0))
    #define ENABLE_EXTERNAL_VSL (0x60|(1<<0))
    #define DISABLE_SECOND_PRECHARGE (0x60|(0<<1))
    #define ENABLE_SECOND_PRECHARGE (0x60|(1<<1))
```

Listing 4. Funkcja ustawiająca aktywny obszar ekranu

```
//X1 i X2: 0...127 (tylko parzyste), Y1 i Y2: 0...127
void setActiveWindow(uint8_t X1, uint8_t Y1, uint8_t X2, uint8_t Y2)
{
    writeCommand(SET_COLUMN_ADDR);  //Set Column Address
    writeCommand(X1>>1);  //Adres startowy aktywnego obszaru pamięci ekranu dla osi X
    writeCommand(X2>>1);  //Adres końcowy aktywnego obszaru pamięci ekranu dla osi X
    writeCommand(SET_ROW_ADDR);  //Set Row address
    writeCommand(Y1);  //Adres startowy aktywnego obszaru pamięci ekranu dla osi Y
    writeCommand(Y2);  //Adres końcowy aktywnego obszaru pamięci ekranu dla osi Y
}
```

Listing 5. Funkcje wyświetlające wypełniony i pusty prostokąt

```
void drawFilledBox(uint8_t X1, uint8_t Y1, uint8_t X2, uint8_t Y2, uint8_t Color)
{
    uint16_t Pixels = ((X2-X1+1)/2)*(Y2-Y1+1);
    setActiveWindow(X1, Y1, X2, Y2);
    while(Pixels--) writeData(Color);
}

void drawRectangle(uint8_t X1, uint8_t Y1, uint8_t X2, uint8_t Y2, uint8_t Color)
{
    drawFilledBox(X1 , Y1 , X2 , Y1, Color);
    drawFilledBox(X1 , Y2 , X2 , Y2, Color);
    drawFilledBox(X1 , Y1 , X1 , Y2, Color);
    drawFilledBox(X2 , Y1 , X2 , Y2, Color);
}
```

od przylutowania tego gniazda. Najprostszym sposobem montażu elementów o tak dużym zagęszczeniu wyprowadzeń, niewymagającym jednocześnie posiadania specjalistycznego sprzętu, jest użycie typowej stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń złącza. Należy przy tym uważać, by nie uszkodzić termicznie tego elementu. Jakość tak wykonanego połączenia sprawdzamy pod lupą, korzystając z miernika pozwalającego sprawdzić ciągłość połączeń. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej. Następnie lutujemy mikrokontroler oraz wszystkie elementy bierne przeznaczone do montażu na warstwie TOP obwodu drukowanego.

W kolejnym kroku przechodzimy do warstwy BOTTOM, gdzie w pierwszej kolejności lutujemy elementy półprzewodnikowe, następnie rezystory i kondensatory SMD, zaś na samy końcu dławiki L1/L2 oraz elementy mechaniczne, jak mikroprzyciski SW1/SW2 oraz gniazdo micro-USB. Na samym końcu podłączamy taśmę wyświetlacza OLED do złącza ZIF umieszczonego na płytce naszego modułu, dbając o odpowiednie zablokowanie zatrzasku tego złącza, po czym przyklejamy go do obwodu drukowanego, korzystając z dwustronnej taśmy klejącej.

Listing 6. Funkcja odpowiedzialna za odczytanie, dekompresję i wyświetlenie obrazka

```
void drawPicture(uint8_t X1, uint8_t Y1, const uint8_t *Picture)
{
    register uint8_t byteA, byteB;
    uint16_t bytesToSend, repeats, i=2;
    //Obliczamy liczbę bajtów jakie należy wysłać do sterownika SSD1327
    bytesToSend = (pgm_read_byte(&Picture[0])>>1)*pgm_read_byte(&Picture[1]);
    //Ustawiamy aktywne okno pamięci obrazu sterownika SSD1327 by uprościć zapis danych
    setActiveWindow(X1, Y1, X1+pgm_read_byte(&Picture[0])-1, Y1+pgm_read_byte(&Pic-
ture[1])-1);
    do
    {
        if(bytesToSend == 1)  //Dla ostatniego bajta danych
        {
            writeData(pgm_read_byte(&Picture[i]));
            bytesToSend--;
        }
        else
        {
            byteA = pgm_read_byte(&Picture[i]);  //Odczytujemy bajt „i”
            byteB = pgm_read_byte(&Picture[i+1]);  //Odczytujemy bajt „i+1”
            if( byteA != byteB)  //Sprawdzamy, czy kolejne bajty są różne
            {
                writeData(byteA);
                bytesToSend--;
                i++;
            }
            else
            //Jeśli kolejne bajty są takie same, to ustalamy liczbę powtórzeń danego bajta
(+2 powtórzenia
            //obligatoryjne) wynikające ze sposobu zapisu kompresji typu mRLE (zmodyfikow-
ane RLE)
            {
                repeats = pgm_read_byte(&Picture[i+2])+2;  //Odczytujemy liczbę powtórzeń
danego bajta + 2 bajty obligatoryjne
                while(repeats--) {writeData(byteA); bytesToSend--;} //Wysyłamy liczbę
powtórzeń bajta
                i +=3;  //Przesuwamy indeks o 3 miejsca
            }
        }
    } while(bytesToSend);
}
```

Listing 7. Definicja typu danych przechowującego parametry czcionki

```
typedef struct  //Deklaracja struktury przechowującej parametry bieżącej czcionki
{
    uint8_t Width;  //Rzeczywista szerokość znaku (px)
    uint8_t Height;  //Rzeczywista wysokość znaku (px)
    uint8_t Interspace;  //Odstęp pomiędzy znakami (px)
    uint8_t BytesPerChar;  //Liczba bajtów danych tablicy wzorców na 1 znak
    uint8_t FirstCharCode;  //Kod ASCII pierwszego znaku
    const uint8_t *Bitmap;  //Wskaźnik do tablicy zawierającej wzorce poszczególnych znaków
} fontDescription;
```

Listing 8. Funkcja wybierająca bieżącą czcionkę ekranową

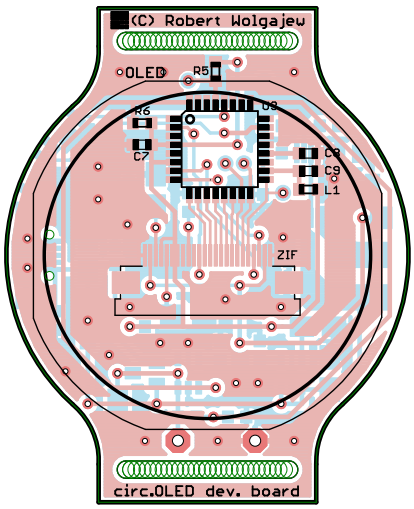
```
void setFont(const fontDescription *Font)
{
    CurrentFont.Width = pgm_read_byte(&Font->Width);  //Rzeczywista szerokość czcionki
    CurrentFont.Height = pgm_read_byte(&Font->Height);  //Rzeczywista wysokość czcionki
    CurrentFont.Interspace = pgm_read_byte(&Font->Interspace);  //Odstęp pomiędzy znakami
    CurrentFont.BytesPerChar = pgm_read_byte(&Font->BytesPerChar);  //Liczba bajtów
na definicje pojedyn. znaku
    CurrentFont.FirstCharCode = pgm_read_byte(&Font->FirstCharCode);  //Kod ASCII definicji
pierwszego znaku
    CurrentFont.Bitmap = (uint8_t*)pgm_read_word(&Font->Bitmap);  //Wskaźnik do tablicy
wzorców tej czcionki
}
```

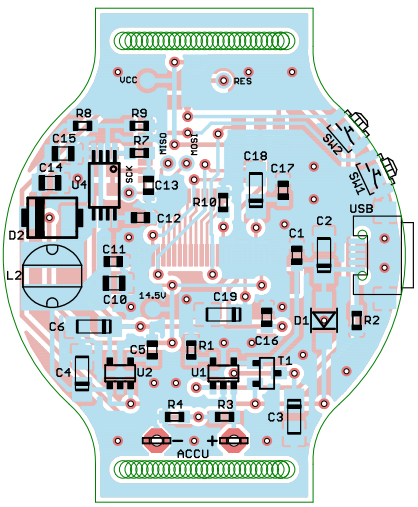
Listing 9. Funkcja odpowiedzialna za rysowanie znaków przy użyciu czcionki ekranowej

```
void drawText(uint8_t X1, uint8_t Y1, char *Text, uint8_t Color, uint8_t Background)
{
    register char Character;
    register uint16_t byteIndex, offset;
    register uint8_t corrWidth, bitIndex, readByte;
    //Korygujemy docelową szerokość w celu poprawnego ustawienia okna zapisu, które za-
okrąglamy do pełnych bajtów. Ma to uprościć
    //sprawdzenie poszczególnych bitów bajta i tak np. dla szerokości = 6, skorygowana sze-
rokość będzie równa 8, dla 12 równa 16 itd.
    corrWidth = ((CurrentFont.Width/8) + (CurrentFont.Width%8 !=0 ? 1 : 0)) *8;
    while ((Character = *(Text++)))  //Kolejne znaki napisu wejściowego
    {
        setActiveWindow(X1, Y1, X1+corrWidth-1, Y1+CurrentFont.Height-1);
        offset = (Character-CurrentFont.FirstCharCode)*CurrentFont.BytesPerChar;  //Offset
położenia wzorca w tablicy wzorców znaków
        for( byteIndex = offset; byteIndex < (offset + CurrentFont.BytesPerChar); byteIn-
dex++)
        {
            //Odczytujemy kolejny bajt definicji znaku umieszczony w tablicy Bitmap
            readByte = pgm_read_byte(CurrentFont.Bitmap+byteIndex);
            //Kolejne pary bitów bajta: 0-1, 2-3, 4-5, 6-7 by utworzyć bajt wynikowy zawi-
erający dane 2 pixeli w 16 odcieniach
            for( bitIndex=0; bitIndex<6; bitIndex+=2 )
            {
                if(readByte & (1<<bitIndex)) corrWidth= Color <<4; else corrWidth = Back-
ground <<4;  //corrWidth użyte jako temp
                if(readByte & (1<<(bitIndex+1))) corrWidth |= Color; else corrWidth |=
Background;
                writeData(corrWidth );
            }
        }
        X1+=(CurrentFont.Width+CurrentFont.Interspace);  //Przesuwamy się o rzeczywistą
szerokość znaku plus odstęp pomiędzy znakami
    }
}
```

Akumulator zasilający podłączamy do wyprowadzeń ACCU (zachowując, co oczywiste, odpowiednią polaryzację) i podklejamy do płytki urządzenia od strony BOTTOM. Warto zauważyć, że na płytce drukowanej urządzenia, po stronie BOTTOM, przewidziano, dość duże punkty lutownicze, za których pomocą podłączymy nasz system do programatora mikrokontrolerów AVR (oznaczone VCC, SCK, MOSI, MISO, RES).

Robert Wołgajew, EP





Rysunek 4. Schemat montażowy zestawu ewaluacyjnego dla okrągłych wyświetlaczy OLED z rodziny WEO128128B