Sprawozdanie z zadań wykonanych w ramach przedmiotu Systemy Wbudowane

Dawid Wolniak, gr. 4, nr. Albumu: 162885

Wykonane zadania

Podczas zajęć laboratoryjnych udało mi się zrealizować kolejno zadania:

Zadanie 1 – Coś się kończy, coś się zaczyna

Zadanie 3 – Reklama

Zadanie 4 – Mam tę moc!

Zadanie 1 – Coś się kończy, coś się zaczyna

Treść:

Program przełączający cyklicznie 9 podprogramów (następny program po 9 to 1, a poprzedni względem 1 to 9).

Przycisk S4 – następny program

Przycisk S3 – poprzedni program

Podprogramy:

1. 8 bitowy licznik binarny zliczający w górę (0...255)

2. 8 bitowy licznik binarny zliczający w dół (255...0)

3. 8 bitowy licznik w kodzie Graya zliczający w górę (reprezentacja 0...255)

4. 8 bitowy licznik w kodzie Graya zliczający w dół (reprezentacja 255...0)

5. 2x4 bitowy licznik w kodzie BCD zliczający w górę (0...99)

6. 2x4 bitowy licznik w kodzie BCD zliczający w dół (99...0)

7. 3 bitowy wąż poruszający się lewo-prawo

8. Stos

9. 6 bitowy generator liczb pseudolosowych oparty o konfigurację 1110011

Wąż:

Stan	D10	D9	D8	D7	D6	D5	D4	D3
1						■	■	■
2					■	■	■	
3				■	■	■		
4			■	■	■			
5		■	■	■				
6	■	■	■					
7		■	■	■				
8			■	■	■			
9				■	■	■		
10					■	■	■	
11						■	■	■

Stos:

Stan	D10	D9	D8	D7	D6	D5	D4	D3
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								
34								
35								
36								

Wykonanie zadania

```
typedef void (*function_ptr)(void); //definicja wskaźnika funkcji



void BinaryCountUp(void); //deklaracje funkcji zdefiniowanych w późniejszym
kodzie
void BinaryCountDown(void);
void GrayCountUp(void);
void GrayCountDown(void);
void BCDCountUp(void);
void BCDCountDown(void);
void Snake(void);
void Stack(void);
void Random(void);
void SYS_Initialize ( void ) ;
```

Początkowa część kodu zadania opiera się o definicję wskaźnika dla funkcji oraz deklarację funkcji będące zaimplementowanych w późniejszej części kodu.

```
void delay(unsigned int ms) //funkcja delay, która tworzy opóźnienie w
programie
{
    unsigned int i; //deklaracje zmiennych
    unsigned char j;


    for (i = 0; i < ms; i++) { //pętla for która wykonuje się tyle razy ile
milisekund podaliśmy w argumencie


        for (j = 0; j < 200; j++) { //pętla która wykonuje 200 iteracji
            Nop(); //no operation - funkcja która nie wykonuje operacji, w tym
kodzie służy do tworzenia opóźnienia
            Nop();
            Nop();
            Nop();
            Nop();
        }
    }
}
```

W kolejnej części znajduje się funkcja delay z materiałów dostarczonych przez prowadzącego na początku zajęć. Służy ona do tworzenia opóźnienia w funkcjach w celu zaobserwowania ich działania.

```
void BinaryCountUp() //implementacja funkcji liczącej binarnie do góry od 0 do 255
{
    for(int i=0; i<256; i++) //pętla która wykonuje 256 iteracji, jest licznikiem w funkcji i na podstawie iteracji wyświetlana jest aktualna liczba
    {
        LATA = i; //wyświetlenie aktualnego stanu licznika na interfejsie LATA - diody
        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan licznika
        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6
        break; //przerwanie działania funkcji
    }
}

void BinaryCountDown() //implementacja funkcji liczącej binarnie w dół od 255 do 0
{
    for(int i=255; i>=0; i--) //pętla która wykonuje 256 iteracji, jest licznikiem w funkcji i na podstawie iteracji wyświetlana jest aktualna liczba
    {
        LATA = i; //wyświetlenie aktualnego stanu licznika na interfejsie LATA - diody
        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan licznika
        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6
        break; //przerwanie działania funkcji
    }
}
```

Funkcje binarnego liczenia w górę i w dół, których wynik jest przedstawiany binarnie na wbudowanych diodach. Przekazanie wyniku dziesiętnego w kodzie do interfejsu LATA przedstawia go na diodach w kodzie binarnym.

```
void GrayCountUp() //implementacja funkcji liczącej w kodzie Graya do góry od 0 do 255
{

    int x; //deklaracja zmiennej
    for(int i=0; i<256; i++) //pętla która wykonuje 256 iteracji
    {
        x = i; //przypisanie do zmiennej x wartości i z pętli liczącej
        x = x^x>>1; //obliczenie wartości w kodzie Graya za pomocą przesunięcia bitowego o 1 i porównanie wartości pierwotnej oraz przesuniętej za pomocą operacji XOR
        LATA = x; //wyświetlenie aktualnego stanu licznika na interfejsie LATA - diody
        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan licznika
        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6
        break; //przerwanie działania funkcji
    }

}

void GrayCountDown() //implementacja funkcji liczącej w kodzie Graya w dół od 255 do 0
{

    int x; //deklaracja zmiennej
    for(int i=255; i>=0; i--) //pętla która wykonuje 256 iteracji
    {
        x = i; //przypisanie do zmiennej x wartości i z pętli liczącej
```

```
        x = x^x>>1; //obliczenie wartości w kodzie Graya za pomocą
przesunięcia bitowego o 1 i porównanie wartości pierwotnej oraz przesuniętej
za pomocą operacji XOR

        LATA = x; //wyświetlenie aktualnego stanu licznika na interfejsie LATA
- diody

        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan
licznika

        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w
dowolnym momencie jej działania przy naciśnięciu przycisku S6

        break; //przerwanie działania funkcji

    }



}
```

Implementacja funkcji liczących w kodzie Graya do góry i do dołu, Do przetworzenia wartości na kod binarny używane jest przesunięcie bitowe o jedno miejsce wartości a później porównanie wartości pierwotnej oraz wartości po przesunięciu za pomocą bramki XOR. Wynikiem tego jest liczba w kodzie Graya. Informacje na temat tworzenia w taki sposób kodu Graya znalazłem w Internecie.

```
void BCDCountUp() //implementacja funkcji liczącej w górę w BCD od 0 do 99
{


    int x; //deklaracja zmiennej
    for(int i=0; i<100; i++) //pętla która wykonuje 100 iteracji
    {
        x = ((i/10)*16)+(i%10); //obliczanie wartości zmiennej w kodzie BCD
        LATA = x; //wyświetlenie aktualnego stanu licznika na interfejsie LATA
- diody
        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan
licznika
        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w
dowolnym momencie jej działania przy naciśnięciu przycisku S6
        break; //przerwanie działania funkcji
    }


}


void BCDCountDown() //implementacja funkcji liczącej w dół w BCD od 99 do 0
{


    int x; //deklaracja zmiennej
    for(int i=99; i>=0; i--) //pętla która wykonuje 100 iteracji
    {
        x = ((i/10)*16)+(i%10); //obliczanie wartości zmiennej w kodzie BCD
        LATA = x; //wyświetlenie aktualnego stanu licznika na interfejsie LATA
- diody
        delay(250); //opóźnienie 250 milisekund aby można było zauważyć stan
licznika
        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w
dowolnym momencie jej działania przy naciśnięciu przycisku S6
```

```
        break; //przerwanie działania funkcji
    }


}
```

Implementacja funkcji liczących w górę i w dół w kodzie BCD. Funkcja przelicza liczbę na kod BCD za pomocą wzoru, który część dziesiętną oddziela od części jedności poprzez dzielenie przez 10, a później przestawia ją o 4 miejsca w prawa za pomocą mnożenia przez 16. Dzięki takiej operacji liczba przedstawiana jest na 8 diodach z czego pierwsze 4 przedstawiają binarnie liczbę dziesiątek, a pozostałe 4 liczbę jedności. Przy rozwiązaniu tej części zadania nie napotkałem na większe trudności

```
void Snake() //implementacja funkcji wyświetlającej węża na interfejsie LATA
{



    int x; //deklaracja zmiennej

    for(int i=1; i<=32; i*=2) //// pętla która wykonuje się od 1 do 32 z krokiem równym podwojeniu aktualnej wartości i

    {

        x = i*7; //obliczenie zmiennej x

        LATA=x; //wyświetlanie węża na interfejsie LATA

        delay(500); //opóźnienie 500 milisekund aby można było zauważyć ruch węża

        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6

        break; //przerwanie działania funkcji

    }

    for(int i=16; i>=1; i/=2) //Pętla która wykonuje się od 16 do 1 z krokiem równym podzieleniu aktualnej wartości i przez 2.

    {

        x = i*7; //obliczenie zmiennej x

        LATA=x; //wyświetlanie węża na interfejsie LATA

        delay(500); //opóźnienie 500 milisekund aby można było zauważyć ruch węża

        if (BUTTON_IsPressed(BUTTON_S6)) //warunek przerwania funkcji w dowolnym momencie

                                //jej działania przy naciśnięciu przycisku S6

        break; //przerwanie działania funkcji

    }


}
```

Implementacja funkcji, której zadaniem jest wyświetlanie węża na interfejsie LATA. W zadaniu zostało sprecyzowane jak ma wyglądać ruch i łatwo można zauważyć, że jest to przesunięcie o 1 w lewo a później o 1 w prawo zapisanej binarnie 7. Efekt przejścia uzyskałem za pomocą przemnożenia liczby 7 przez dwa i przechodzenie po coraz większych wartościach. Kolejno funkcja przechodzi po

wartościach 7, 14, 28, 56, 112, 224 co odpowiada binarnie trzem bitom poruszającym się w lewo. Przy powrocie węża funkcja przechodzi odwrotnie przez wartości z pominięciem 224 aby uniknąć ponownego wyświetlenia węża w tym samym miejscu.

```
void Stack()
{

    int x = 0; //deklaracja zmiennej
    int y = 0; //deklaracja zmiennej
    int z = 6; //deklaracja zmiennej
    int count = 1; //deklaracja zmiennej

    for(int i=0; i<=7; i++) // Pętla for, która wykonuje się 8 razy dla wartości i od 0 do 7
    {
        x=0; // Resetowanie wartości zmiennej x na 0 przed każdą iteracją zewnętrznej pętli

        for(int i=0; i<=z; i++) // Pętla for, która wykonuje się zależnie od wartości zmiennej z.
        {
            if(x==0){
                x++;
            }
            else{
                x *= 2;
            }
            // jeśli x jest równe 0, zwiększa się wartość x o 1
            // w przeciwnym przypadku, wartość x jest mnożona przez 2

            count = y + x;  // obliczanie wartości zmiennej count poprzez dodanie wartości y do x
            LATA= count; // wyświetlanie stosu na interfejsie LATA
            delay(250); //opóźnienie 250 milisekund aby można było zauważyć układanie stosu
        }
```

```
        z--; // zmniejszenie wartości zmiennej z o 1 po każdej iteracji
zewnętrznej pętli

        y += x; // zwiększanie wartości zmiennej y o wartość x po każdej
iteracji zewnętrznej pętli


        if (BUTTON_IsPressed(BUTTON_S6)){ //warunek przerwania funkcji w
dowolnym

            //momencie jej działania przy naciśnięciu przycisku S6//warunek
przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu
przycisku S6

            break; //przerwanie działania funkcji

        }

    }

}
```

Implementacja funkcji która na interfejsie LATA wyświetla stos. Funkcja ma ustawiony licznik na 6 co odpowiada ilości przejść jakie musi wykonać w pętli wewnętrznej. Zewnętrzna pętla funkcji liczy od 0 do 7 co odpowiada kolejno bitom na interfejsie LATA. Funkcja wewnętrzna przelicza ustaloną liczbę razy dwójkowo i na bieżąco wyświetla wynik po czym gdy zakończy działanie ostatnia liczba licznika jest dodawana do zmiennej aby przedstawić blok, który nałożył się na stos.

```
void Random()
{

    unsigned char lcg(unsigned int x) //algorytm liniowego kongruentnego generatora liczb pseudolosowych
    {
        const unsigned int a = 251; // Wartość a w algorytmie LCG.
        const unsigned int c = 37; // Wartość c w algorytmie LCG.
        const unsigned int m = 256; // Wartość m w algorytmie LCG.
        return (a * x + c) % m; // oblicza nową wartość pseudolosową na podstawie wzoru LCG.
    }

    unsigned int x = 0b1110011; // Inicjalizacja zmiennej x jako binarnej wartości 1110011
    while(1){
        LATA = x; // wyświetlanie losowej wartości na interfejsie LATA
        delay(1000); //opóźnienie 1000 milisekund aby można było zauważyć różne wygenerowane wartości
        if (BUTTON_IsPressed(BUTTON_S6)){ //warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6//warunek przerwania funkcji w dowolnym momencie jej działania przy naciśnięciu przycisku S6
            break; //przerwanie działania funkcji
        }
        x = lcg(x); // Generowanie nowej wartości pseudolosowej na podstawie aktualnej wartości x
        // za pomocą funkcji lcg, która implementuje algorytm LCG
    };
}
```

Implementacja funkcji generującej liczby pseudolosowe w oparciu o konfigurację 1110011. Do generowania liczb pseudolosowych używany jest algorytm liniowego kongruentnego generatora liczb pseudolosowych. Informacje na temat takiego sposobu generowania liczb pseudolosowych podsunął mi ChatGPT, a później jego sposób działania odnalazłem w Internecie. Największym problemem podczas opracowania tej części zadania było odnalezienie odpowiednich składników do

generowania liczb, ponieważ pierwotne ustawienie, które wprowadziłem generowało liczby równe w zapisie bitowym czyli 2, 4, 8, 16 itd. Pomocny do wygenerowania tych składników znów okazał się ChatGPT.

```
int main ( void )
{


    SYS_Initialize(); // Inicjalizacja systemu
    int program = 0; //deklaracja zmiennej program, która określa aktualny program


    function_ptr funcs[] = {BinaryCountUp, BinaryCountDown, GrayCountUp, GrayCountDown, BCDCountUp, BCDCountDown, Snake, Stack, Random};
    //deklaracja tablicy funkcji, która przechowuje wskaźniki do funkcji odpowiadającym różnym podprogramom
    while (1) { //nieskończona pętla
        while (!BUTTON_IsPressed(BUTTON_S3) && !BUTTON_IsPressed(BUTTON_S4));
        //pętla która swoim wykonywaniem blokuje przeskok programu i ponowne odpalenie się funkcji
        if (BUTTON_IsPressed(BUTTON_S3) && program < 9) //warunek który przy wciśnięciu przycisku S3 zmienia program na następny
            program++;
        else if (BUTTON_IsPressed(BUTTON_S3))
            program = 0;
        else if (BUTTON_IsPressed(BUTTON_S4) && program > 0) //warunek który przy wciśnięciu przycisku S4 zmienia program na poprzedni
            program--;
        else if (BUTTON_IsPressed(BUTTON_S4))
            program = 9;


        if (program > 0 && program <= 9) { //warunek który sprawdza czy zmienna program posiada właściwą wartość do odpalenia podprogramu
            funcs[program - 1](); //wywołanie funkcji, która odpala aktualnie wybrany podprogram, -1 jest potrzebne aby ustawić zmienną program jako odpowiedni index dla wywołania funkcji
        }
    }
```

```
}
```

Implementacja głównej części programu, w której zawarta jest nieskończona pętla umożliwiająca nieprzerwane działanie mikrokontrolera. Podczas pisania funkcji przełączania podprogramów napotkałem problem w postaci pracy mikrokontrolera nieprzerwanie do momentu spełnienia warunku. W związku z bardzo szybkim wykonywaniem instrukcji praktycznie nie możliwe stało się przełączanie podprogramów. Z tego powodu zastosowałem pętlę while która nie wykonuje żadnej instrukcji i oczekuje na przerwanie poprzez naciśnięcie przycisku. Czas wykonywania operacji poprzez mikrokontroler pozwala na poprzez jednokrotne naciśnięcie przycisku wyjście z pętli blokującej działanie i wykonanie instrukcji przejścia do programu wyżej lub niżej w tablicy funkcji. Zastosowanie tablicy funkcji w kodzie zostało mi podsunięte przez ChatGPT i Internet po tym jak przy użyciu instrukcji switch() programy nie chciały się przełączać.

Zadanie 3 – Reklama

Wykorzystując wyświetlacz LCD 16x2 i opcjonalnie diody LED (D10...D3), należy wykonać reklamę – temat dowolny. Aby reklama była atrakcyjna, należy stworzyć proste animacje z wykorzystaniem samodzielnie zdefiniowanych symboli, tworzących obrazy na wyświetlaczu LCD.

Wykonanie zadania

```
int main ( void )

{

    SYS_Initialize(); //Inicjalizacja systemu

    char reklama[50][16]={ //tablica z zawartością reklamy

        "               ",

        "   KOMPUTERY !  ",

        "   KOMPUTERY !  ",

        "               ",

        "               ",

        "              K",

        "             K ",

        "            K O",

        "           K O ",

        "          K O M",

        "         K O M ",

        "        K O M P",

        "       K O M P ",

        "      K O M P U",

        "     K O M P U ",

        "    K O M P U T",

        "   K O M P U T ",

        "  K O M P U T E",

        " K O M P U T E ",

        "K O M P U T E R",

        " O M P U T E R Y",
```

```
    "O M P U T E R Y ",
    " M P U T E R Y !",
    "M P U T E R Y ! ",
    " P U T E R Y !  ",
    "P U T E R Y !   ",
    " U T E R Y !    ",
    "U T E R Y !     ",
    " T E R Y !      ",
    "T E R Y !       ",
    " E R Y !        ",
    "E R Y !         ",
    " R Y !          ",
    "R Y !           ",
    " Y !            ",
    "Y !             ",
    " !              ",
    "!               ",
    "                ",
    "   KOMPUTERY !  ",
    "   KOMPUTERY !  ",
    "                ",
    "                ",
    "   SPRZEDAZ!!!  ",
    "   SPRZEDAZ!!!  ",
    "                ",
    "                ",
    "    NAPRAWA!!!  ",
    "    NAPRAWA!!!  ",
};
```

```
    while ( 1 ) //nieskończona pętla
    {
        for(int i=0;i<50;i++) //pętla wykonujaca się 50 razy
        {
            LCD_PutString ( reklama[i] , 16) ; //instrukcja która wyświetla fragment tablicy z reklamą, jako argument przyjmuje zmienną i z pętli
            delay(500); //funkcja powodująca opóźnienie działania aby można było zaobserwować przejście reklamy
            LCD_ClearScreen(); // instrukcja, która czyści wyświetlać
        }

    } ;
```

Implementacja całego zadania, którego celem było stworzenie reklamy z prostą animacją na wyświetlaczu LCD. W nieskończonej pętli programu umieszczona jest pętla for która wykonuje się 50 razy, czyli tyle ile rekordów ma tablica zawierająca treść reklamy. Podczas wykonywania tego zadania nie napotkałem żadnych trudności. W kodzie znajduje się również funkcja delay, którą opisywałem we wcześniejszej części i znajdowała się w materiałach dostarczonych przez prowadzącego.

Zadanie 4 – Mam tę moc!

Kontroler kuchenki mikrofalowej – wszystkie komunikaty muszą być wyświetlone na wyświetlaczu LCD 16x2. Po upływie ustalonego czasu, należy ten fakt zasygnalizować poprzez włączenie wybranej diody LED (D10...D3). Opcjonalnie można zwizualizować wybraną moc, włączając wybrane diody LED (D10...D3). Funkcje: S3 – wybór mocy: 800W, 600W, 350W, 200W S6 – dodanie czasu 1min S5 – dodanie czasu 10s S4 – Start/Stop/Reset zegara

Wykonanie zadania

W zadaniu tak jak i w poprzednich użyłem funkcji delay() dostarczonej w materiałach przez prowadzącego.

```
void alarm(){ //implementacja funkcji alarm



    for(int i = 0; i <= 3; i++) //pętla która wykonuje się 4 razy
    {
        LATA = 1; //włączenie 1 diody na interfejsie LATA
        delay(500); //opóźnienie aby zaobserwować włączenie diody

        LATA = 0; //wyłączenie wszystkich diod na interfejsie LATA
        delay(500); //opóźnienie aby zaobserwować wyłączenie diody
    }
    LATA = 255; //ustawienie LATA na 255 powoduje odpalenie wszystkich diod na zakończenie alarmu
}
```

Implementacja funkcji alarm, która po wywołaniu powoduje czterokrotne odpalenie i wyłączenie pierwszej diody, po tym następuje odpalenie wszystkich diod.

```
int main ( void )
{
    SYS_Initialize(); //Inicjalizacja systemu
    int power = 200; //deklaracja zmiennej, wartość 200 jest wartością podstawową
    int time = 0; // deklaracja zmiennej, służy do ustalenia ilości czasu
    char str1[16] = ""; //deklaracja ciągu znaków wykorzystywany później w programie
    char str2[16] = ""; //deklaracja ciągu znaków wykorzystywany później w programie
    while ( 1 ){ //nieskończona pętla
        sprintf(str1, "MOC: %d", power); //funckja, która łączy ciąg znaków z wartością wyczytaną ze zmiennej
        LCD_PutString ( str1 , sizeof(str1) ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
        LCD_PutString ( " W      " , 11 ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
        sprintf(str2, "CZAS: %d", time); //funckja, która łączy ciąg znaków z wartością wyczytaną ze zmiennej
        LCD_PutString ( str2 , sizeof(str2) ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
        LCD_PutString ( " s" , 2 ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
        delay(500); //funkcja tworząca opóźnienie, aby możliwe było odczytanie informacji na wyświetlaczu
        LCD_ClearScreen(); //instrukcja czyszcząca wyświetlać z wporwadzonych danych
        while((BUTTON_IsPressed(BUTTON_S3))) //pętla która po wciśnięciu przycisku pozwala na zmianę wartości mocy
        {
            if(power==200) //warunki które sprawdzają jaka wartość jest ustawiona i odpowiednio ustawia następną wartość
            {
                power=350;
                break;
```

```
            }
            if(power==350)
            {
                power=600;
                break;
            }
            if(power==600)
            {
                power=800;
                break;
            }
            if(power==800)
            {
                power=200;
                break;
            }
        };
        if((BUTTON_IsPressed(BUTTON_S6))) //warunek który przy wciśnięciu
przysiku S6 dodaje 60 sekund do czasu
        {
            time+=60;
            delay(500); //opóźnienie, aby przez przypadek nie dodać dwa razy
wartości
        }
        if((BUTTON_IsPressed(BUTTON_S5))) //warunek który przy wciśnięciu
przysiku S5 dodaje 10 sekund do czasu
        {
            time+=10;
            delay(500); //opóźnienie, aby przez przypadek nie dodać dwa razy
wartości
        }
        if((BUTTON_IsPressed(BUTTON_S4))&&(time>0)) //warunek, który
rozpoczyna odliczanie czasu
```

```
        {
            delay(1000); //opóźnienie, aby po wystartowaniu odliczania, od razu go nie zatrzymać
            while(time>0)
            {
                if((BUTTON_IsPressed(BUTTON_S4))) //warunek, który zatrzymuje odliczanie czasu
                {
                    delay(1000); //opóźnienie aby nie zresotować czasu od razu po zatrzymaniu
                    while(!(BUTTON_IsPressed(BUTTON_S4))) //pętla która blokuje wykonywanie programu i wyświetla informacje do momentu wciśnięcia przycisku
                    {
                        sprintf(str1, "MOC: %d", power); //funckja, która łączy ciąg znaków z wartością wyczytaną ze zmiennej
                        LCD_PutString ( str1 , sizeof(str1) ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
                        LCD_PutString ( " W      " , 11 ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
                        sprintf(str2, "CZAS: %d", time); //funckja, która łączy ciąg znaków z wartością wyczytaną ze zmiennej
                        LCD_PutString ( str2 , sizeof(str2) ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
                        LCD_PutString ( " s" , 2 ) ; //instrukcja wyświetlająca ciąg znaków na wyświetlaczu
                        delay(500); //funkcja tworząca opóźnienie, aby możliwe było odczytanie informacji na wyświetlaczu
                        LCD_ClearScreen(); //instrukcja czyszcząca wyświetlać z wporwadzonych danych
                    };
                    while((BUTTON_IsPressed(BUTTON_S4))) //pętla, która po naciśnięciu przycisku S4 resetuje czas
                    {
                        time = 1; //ustawienie czasu na 1 po przejściu funkcji dalej czas wyzeruje się
                        delay(500); //opóźnienie
```

```
                    break; //przerwanie pętli
                };
            }
            time--; //dekrementacja wartości czasu
            sprintf(str1, "MOC: %d", power); //funckja, która łączy ciąg
znaków z wartością wyczytaną ze zmiennej
            LCD_PutString ( str1 , sizeof(str1) ) ; //instrukcja
wyświetlająca ciąg znaków na wyświetlaczu
            LCD_PutString ( " W      " , 11 ) ; //instrukcja wyświetlająca
ciąg znaków na wyświetlaczu
            sprintf(str2, "CZAS: %d", time); //funckja, która łączy ciąg
znaków z wartością wyczytaną ze zmiennej
            LCD_PutString ( str2 , sizeof(str2) ) ; //instrukcja
wyświetlająca ciąg znaków na wyświetlaczu
            LCD_PutString ( " s" , 2 ) ; //instrukcja wyświetlająca ciąg
znaków na wyświetlaczu
            delay(1000); //funkcja tworząca opóźnienie, która odlicza
czas, ustawienie na 1000 milisekund czyli 1 sekundę
            LCD_ClearScreen(); //instrukcja czyszcząca wyświetlać z
wporwadzonych danych
        };
        alarm (); //włączenie funkcji alarm w przypadku skończenia się
czasu lub
    }
};
}
```

Implementacja głównej funkcji programu umożliwia ustawienie mocy pod przyciskiem S3 oraz dodanie czasu poprzez przyciski S6 i S5. Przycisk S4 powoduje rozpoczęcie odliczania czasu oraz jego zatrzymanie i reset czasu. Po zakończeniu odliczania lub resecie czasu wywoływany jest alarm, który powoduje chwilowe miganie diody po czym odpala na chwilę wszystkie diody. Podczas rozwiązywania zadania napotkałem problem przy łączeniu ciągów znaków z wartościami całkowitymi, ale z pomocą Internetu dowiedziałem się o funkcji sprintf(), która pozwala na takie połączenia.