Wydział Informatyki Politechniki Białostockiej **Katedra Mediów Cyfrowych i Grafiki Komputerowej** **Laboratorium Architektury Komputerów**	**Data prezentacji projektu:** **01.02.2023**
Dokumentacja Projektu Gry: "Simon Mówi! MSP430" **Grupa:** Adam Prus Bartosz Spiżarny Damian Richter Paweł Rudnik	Prowadzący: Dr Inż. Mirosław Omieljanowicz

Simon Mówi! - MSP430

Gra Simon Mówi! jest luźno inspirowanym portem gry "Sequence Memory "dostępnej na stronie

Projekt gry znajduję się na publicznym repozytorium serwisu github:

Cel gry

Celem gry jest zdobycie jak największej ilości punktów poprzez odgadywanie sekwencji pojawiających się na wyświetlaczu kwadratów. Odgadywanie odbywa się za pomocą przycisków mikrokontrolera. Każdy przycisk odpowiada pojawiającemu się nad nim kwadratem. Po poprawnie wprowadzonej sekwencji, użytkownik przechodzi do następnego poziomu w którym sekwencja zostanie wyświetlona ponownie lecz dodatkowo zwiększona o jeden kwadrat. Maksymalny wynik jaki może uzyskać użytkownik wynosi 50.

Możliwości gry

- Wybór 4 dostępnych graczy na których będzie zapamiętany najlepszy dotychczasowy wynik dla danego gracza.
- Generowanie losowych sekwencji dzięki użyciu obsługi przerwań i specjalnie napisanego generatora liczb pseudolosowych.
- Poprawność sekwencji jest sprawdzana po każdej odpowiedzi co oznacza, że wybór nieprawidłowego kwadratu w sekwencji zostanie uznany za porażkę.

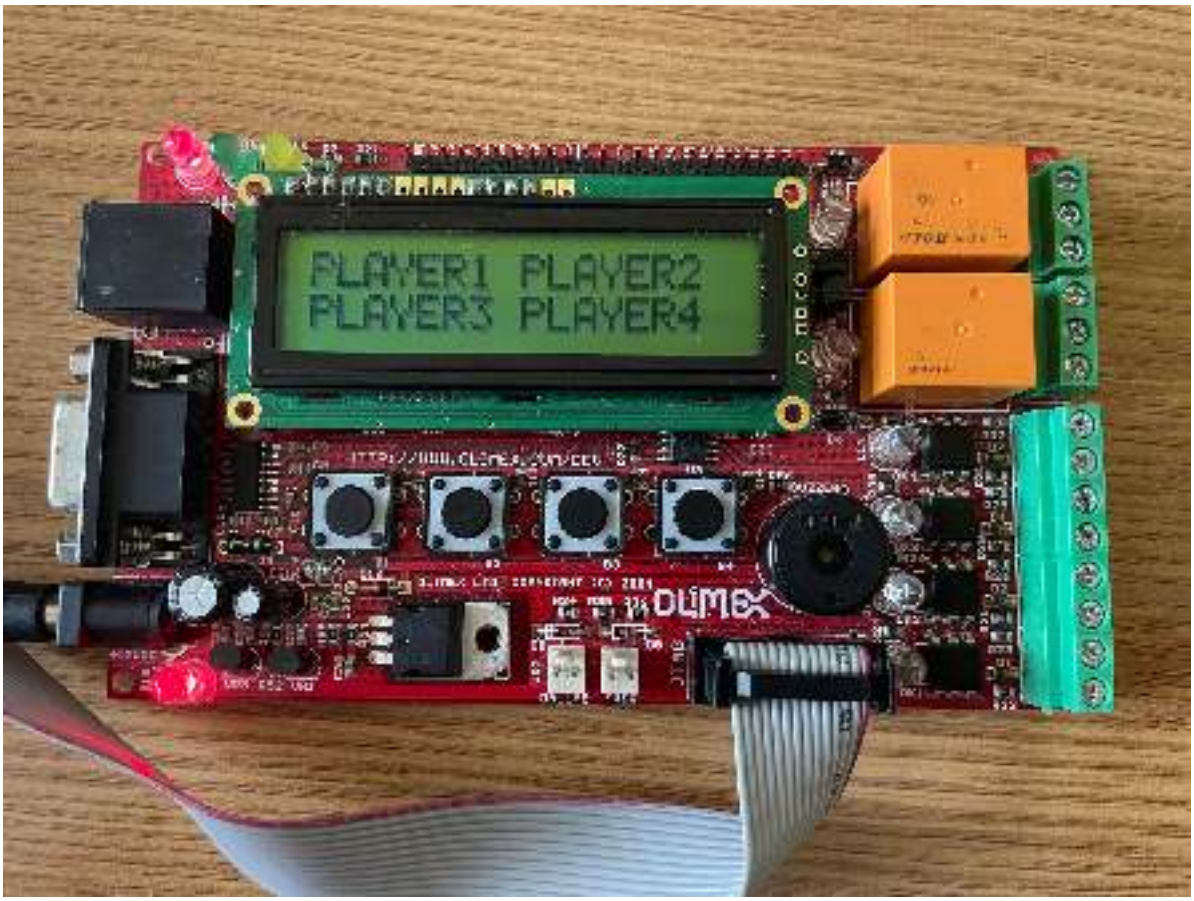
Rozgrywka

Zanim zaglądniemy do kodu źródłowego przyjrzyjmy się rozgrywce.

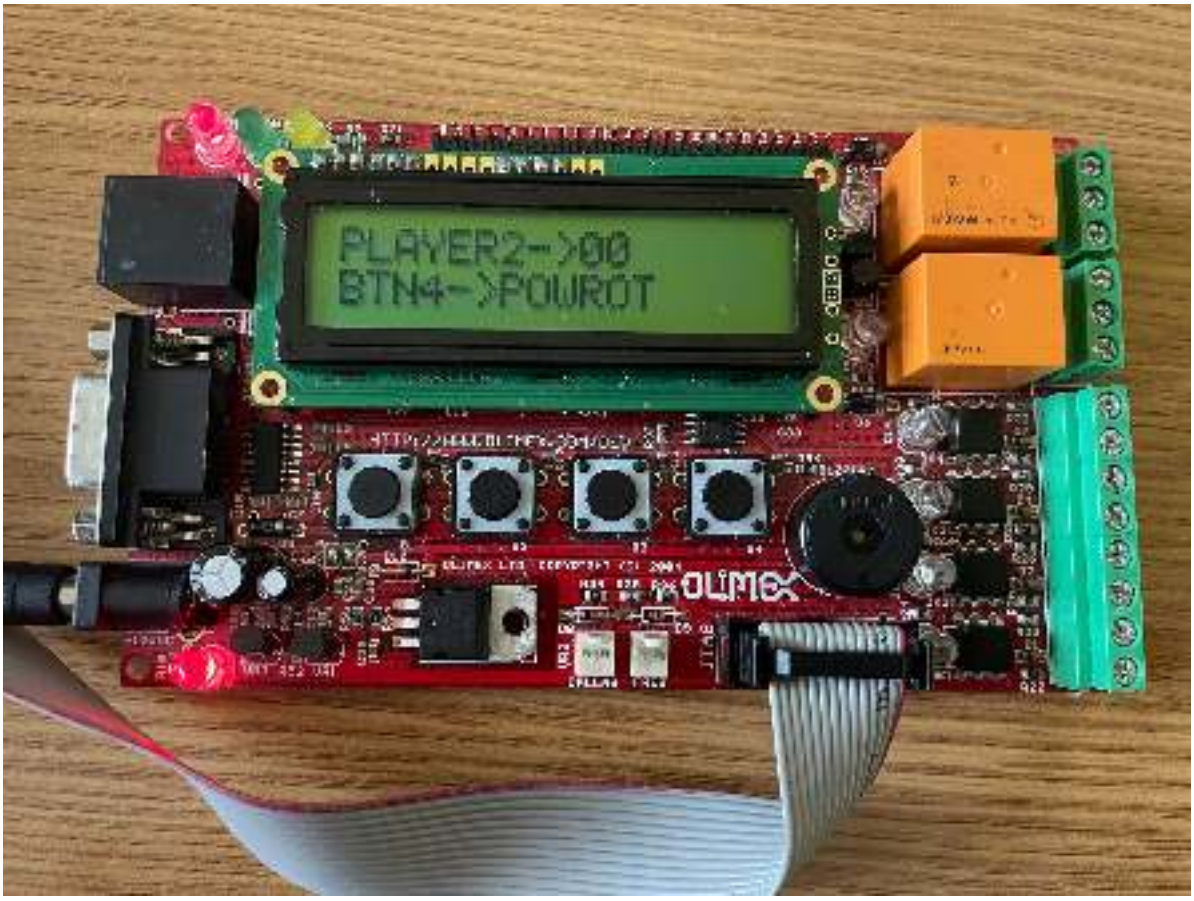
Gra rozpoczyna się od menu głównego w którym za pomocą pierwszego i drugiego przycisku (od lewej strony) możemy wybrać odpowiednią interesującą nas opcje:

- BTN1->GRA (Pierwszy przycisk od lewej przenosi nas do menu rozgrywki)
- BTN2->WYNIKI (Drugi przycisk od lewej przenosi nas do menu wyników - Scoreboard)

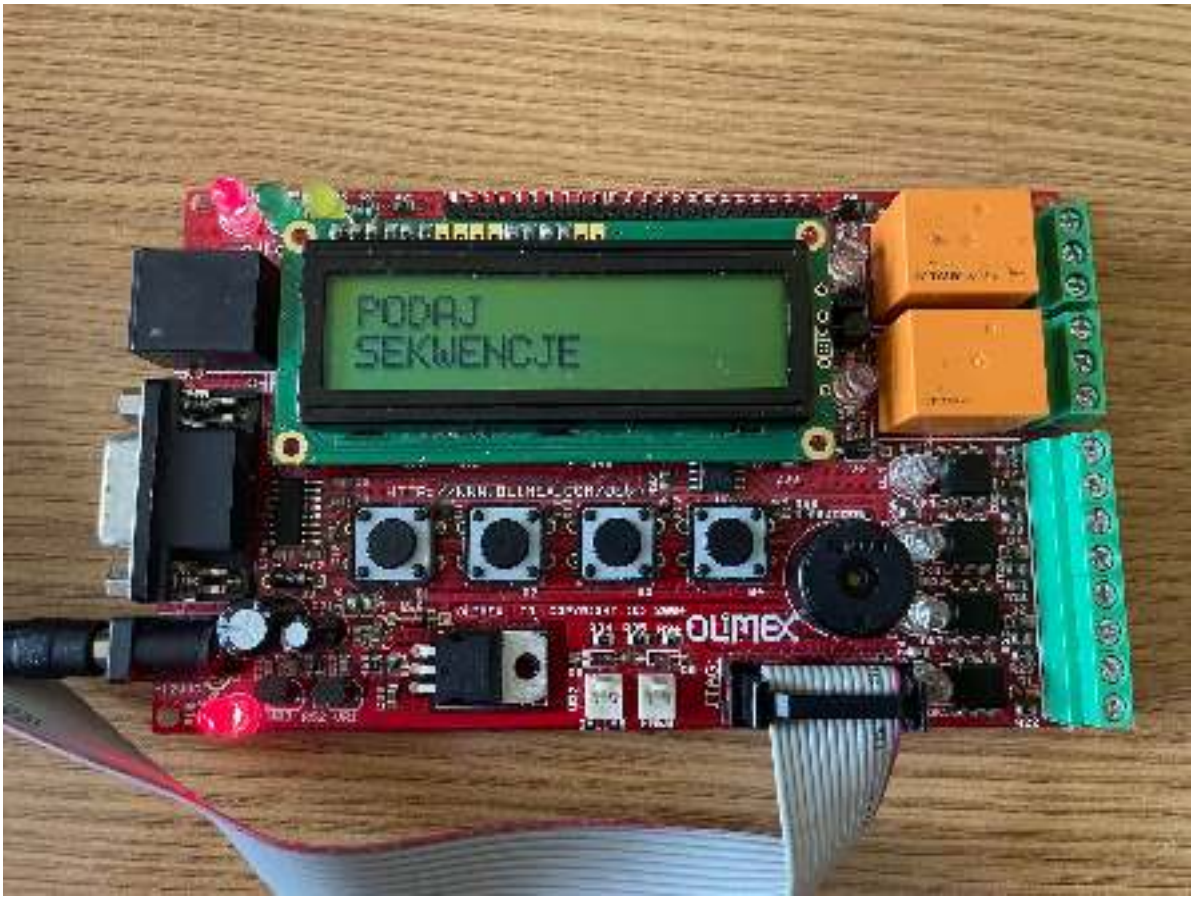
Sprawdźmy najpierw menu wyniki.

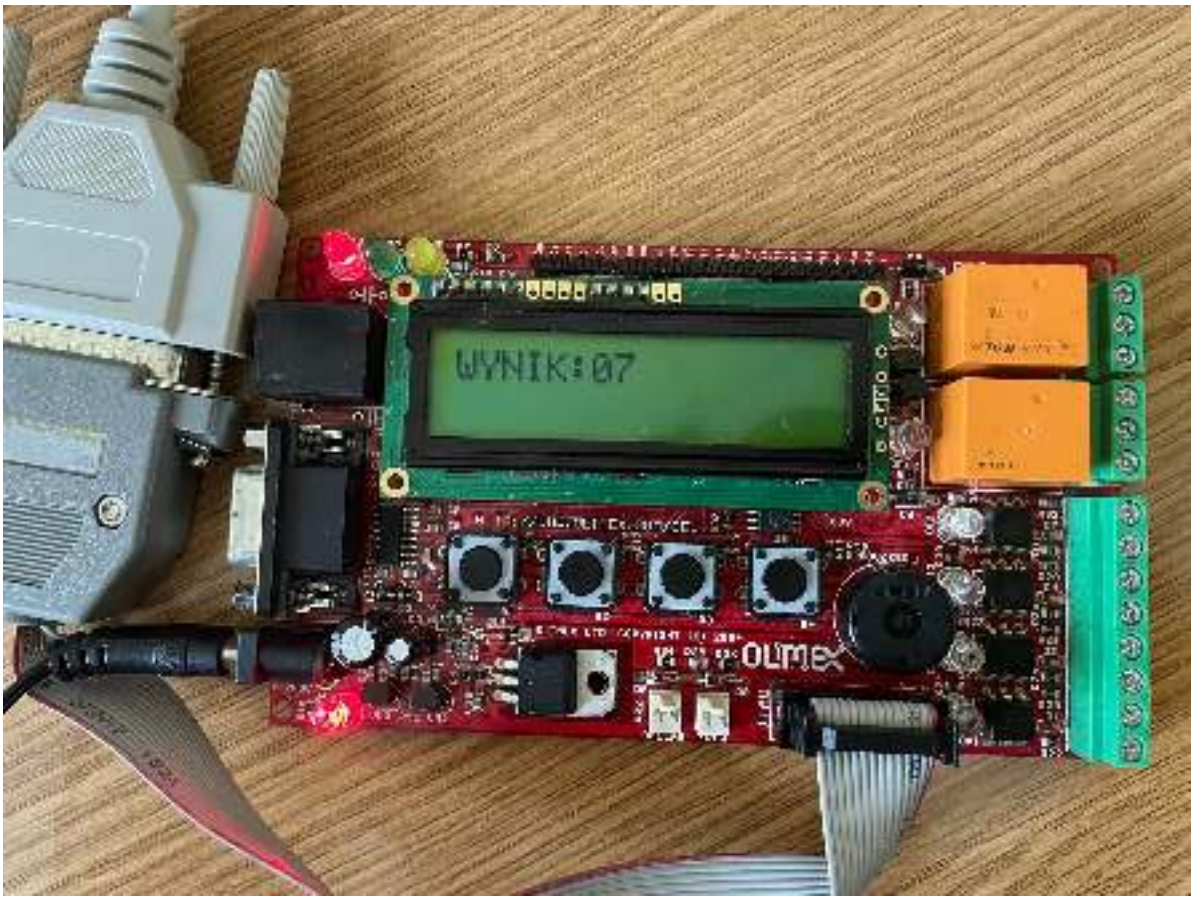


Możemy teraz sprawdzić dotychczasowe najwyższe wyniki poszczególnych graczy do wyboru. Wybierzmy jednego z nich.

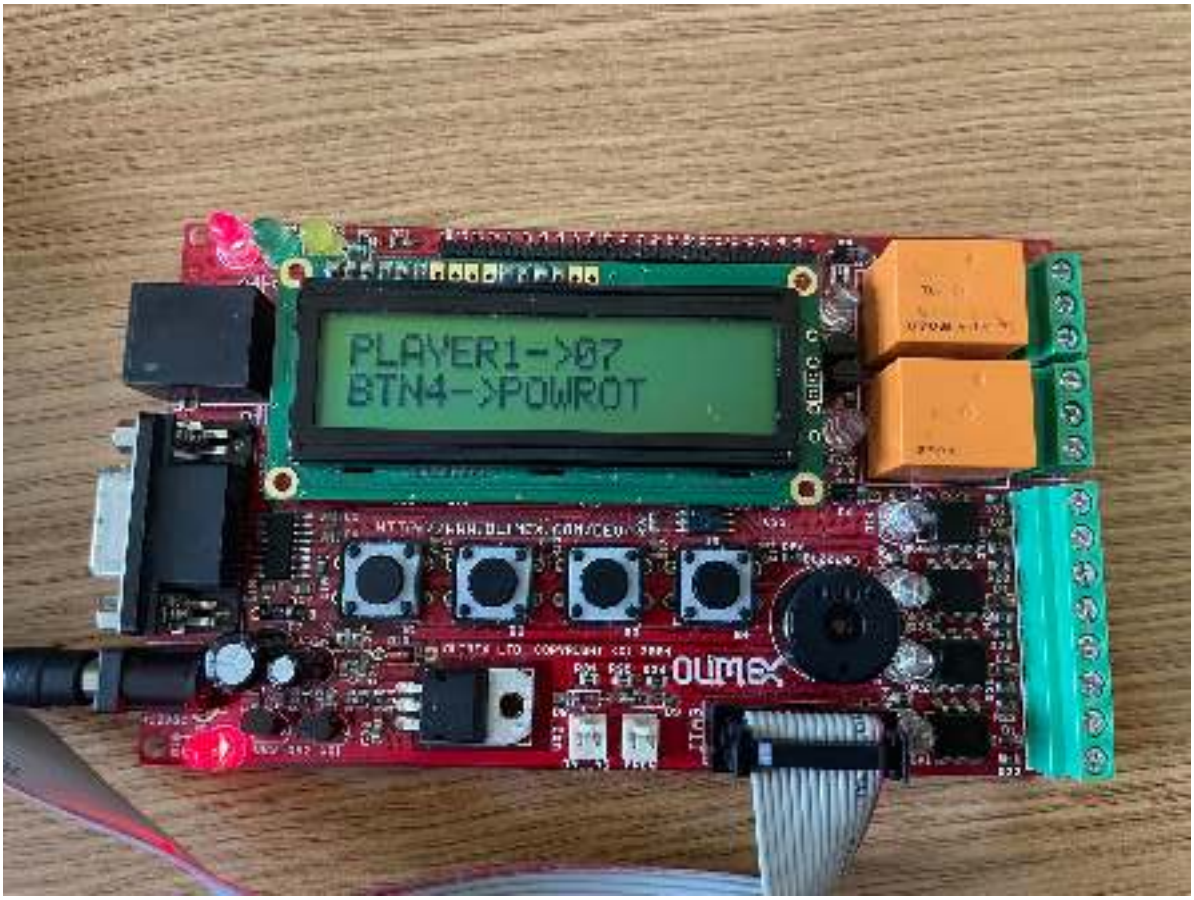


Dotychczasowy wynik będzie wynosił 0 punktów, gdyż jeszcze nie przeprowadziliśmy żadnej rozgrywki. Przejdźmy zatem z powrotem do menu głównego za pomocą przycisku czwartego i tym razem użyjmy opcji GRA. Ponownie zostaniemy poproszeni o gracza. Po jego wyborze zostanie wyświetlony jeden kwadrat rozpoczynający rozgrywkę. Po zakończeniu wyświetlania sekwencji gra poinformuje nas o możliwości podania sekwencji przez gracza.





Sprawdźmy teraz czy nowy wynik został zapisany w tabeli najwyższych wyników dla danego gracza.



Skoro już zapoznaliśmy się z ogólną rozgrywką przejdźmy do kodu źródłowego i omówienia poszczególnych funkcji.

Kod źródłowy

```
/* Dyrektywy include */
#include<msp430x14x.h>          // Główna biblioteka mikrokontrolera msp430
#include "lcd.h"        // Biblioteka obsługująca ekran ciekłokrystaliczny
#include "portyLcd.h"           // Biblioteka obsługająca porty wyświetlacza
#include <stdio.h>      // Biblioteka wejścia-wyjścia języka C
#include <stdlib.h>     // Biblioteka standardowa języka C

/* Dyrektywy define */
#define PRZYCISK0 (P4IN & BIT4) // Definicja przycisku nr. 1
#define PRZYCISK1 (P4IN & BIT5) // Definicja przycisku nr. 2
#define PRZYCISK2 (P4IN & BIT6) // Definicja przycisku nr. 3
#define PRZYCISK3 (P4IN & BIT7) // Definicja przycisku nr. 4

/* Tablice globalne */
int wyniki[4] = {0, 0, 0, 0};   // Tablica zawierajaca najwyzsze wyniki
int sekwencjaProgramu[50];      // Tablica zawierajaca sekwencje programu
int sekwencjaUzytkownika[50];   // Tablica zawierajaca sekwencje uzytkownika

/* Tablica zawierająca znaki unikalne. (Kwadraty) */
unsigned char uniq_chars[3][8] = {
        {31, 31, 31, 31, 31, 31, 31, 31},
        {31, 31, 31, 31, 31, 31, 31, 31},
        {31, 31, 31, 31, 31, 31, 31, 31}
};
```

```
/* Prototypy funkcji */
void inicjalizacjaPrzerwan(void);       // Funkcja inicjalizujaca przerwania
potrzebne do generatora liczb pseudolosowych
void pokazWynik(int);                   // Funkcja pokazujaca wynik danego
gracza (przyjmuje parametr w postaci numeru gracza)
void opoznienie(void);                  // Funkcja wprowadzajaca opoznienie
potrzebne do unikniecia kolizji r□norakich funkcjonalno□ci np. przyciskow
void menuStartowe(void);                // Funkcja uruchamiająca menu startowe
programu
void rozgrywka(void);                   // Funkcja rozpoczynajaca rozgrywke
void sekwencja(int);                    // Funkcja generujaca sekwencje i
przepytujaca gracza
int generujLosowaLiczbe(void);          // Funkcja generujaca pseudolosowa
liczbe od 1 do 4
void narysujKwadrat(int);               // Funkcja rysujaca kwadrat na
wyswietlaczu
void inicjalizacjaPrzyciskow(void);     // Inicjalizacja przyciskow
void pokazWyniki(void);                 // Funkcja obslugujaca menu wyboru
wyniku
void wyslanieKwadratowDoPamieci(void);  // Funkcja wysylajaca kwadraty do
pamieci

/* Zmienne globalne */
int licznik = 0;        // Licznik czasu jaki upłynął od startu urządzenia

/* Funkcja główna */
void main(void) {
        inicjalizacjaPrzerwan();        // inicjalizacja przerwan
        inicjalizacjaPrzyciskow();      // inicjalizacja przycisko
        wyslanieKwadratowDoPamieci();

        WDTCTL = WDTPW + WDTHOLD;       // zatrzymanie WDT

        InitPortsLcd();                 // inicjalizacja port□w
        InitLCD();                      // inicjalizacja LCD
        clearDisplay();                 // czyszczenie LCD
        SEND_CMD(CG_RAM_ADDR);          // mozliwosc wprowadzenia znakow
specjalnych

        // Start programu
        menuStartowe();
}

void inicjalizacjaPrzerwan(void) {
        /* Basic Clock Module ustawiamy na ACLK(zegar 8 MHz ) i dzielimy
częstotliwość przez 2 (4 MHz) */
        BCSCTL1 |= XTS; // ACLK = LFXT1 = HF XTAL 8MHz

        do {
                IFG1 &= ~OFIFG;                                 // Czyszczenie
flgi OSCFault
                for (unsigned int i = 0xFF; i > 0; i--);        // odczekanie
        }
        while ((IFG1 & OFIFG) == OFIFG);                        // dopóki
OSCFault jest ciągle ustawiona
```

```
        BCSCTL1 |= DIVA_1;                                      // ACLK=8
MHz/2=4 MHz
        BCSCTL2 |= SELM0 | SELM1;                               // MCLK= LFTX1
=ACLK

        /* Timer_A  ustawiamy na 500 kHz
        a przerwanie generujemy co 100 ms */
        TACTL = TASSEL_1 + MC_1 + ID_3;                         // Wybieram
ACLK, ACLK/8=500kHz,tryb Up
        CCTL0 = CCIE;                                           // włączenie
przerwań od CCR0
        CCR0 = 50000;                                           // podzielnik
50000: przerwanie co 100 ms

        _EINT();                                                // włączenie
przerwań
}

/* Inicjalizacja przycisków */
void inicjalizacjaPrzyciskow() {
        P4DIR &= ~BIT4;         // Stan wysoki bitu4 rejestru P4DIR
        P4DIR &= ~BIT5;         // Stan wysoki bitu5 rejestru P4DIR
        P4DIR &= ~BIT6;         // Stan wysoki bitu6 rejestru P4DIR
        P4DIR &= ~BIT7;         // Stan wysoki bitu7 rejestru P4DIR
}

void menuStartowe(void) {
        clearDisplay();

        /* Pierwsza linia
           - BTN1 -> GRA */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('1');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('G');
        SEND_CHAR('R');
        SEND_CHAR('A');

        /* Druga linijka
           - BTN2 -> WYNIKI */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('2');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('W');
        SEND_CHAR('Y');
        SEND_CHAR('N');
```

```
        SEND_CHAR('I');
        SEND_CHAR('K');
        SEND_CHAR('I');

        while (1) {
                if (!PRZYCISK0) {
                        /* Rozpoczecie rozgrywki */
                        opoznienie();   // Wprowadzenie opoznienia
zapobiegajacego kolizjom
                        rozgrywka();    // Przejscie do rozgrywki
                } else if (!PRZYCISK1) {
                        /* Pokazanie najlepszych wynikow graczy */
                        opoznienie();   // Wprowadzenie opoznienia
zapobiegajacego kolizjom
                        pokazWyniki();  // Przejscie do menu z najlepszymi
wynikami graczy
                }
        }
}

int wybierzUzytkownika() {
        /* Pierwsza linia
           - PLAYER1 PLAYER 2 */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('1');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('2');
        /* Druga linia
           - PLAYER3 PLAYER 4 */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('3');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
```

```
        SEND_CHAR('R');
        SEND_CHAR('4');

        while (1) {
                /* Przycisk zwraca liczbe calkowita
                odpowiadajaca odpowiedniumu graczu */
                if (!PRZYCISK0) {
                        opoznienie();
                        return 0;
                } else if (!PRZYCISK1) {
                        opoznienie();
                        return 1;
                } else if (!PRZYCISK2) {
                        opoznienie();
                        return 2;
                } else if (!PRZYCISK3) {
                        opoznienie();
                        return 3;
                }
        }
}

void rozgrywka(void) {
        /* Na sam start rozgrywki wybierany jest gracz
        poprzez funkcje wybierzUzytkownika uzytkownik
        przenoszony jest do menu z wyborem a po jego
        dokonaniu do zmiennej typu Integer przesyłana
        jest liczba calkowita odpowiednia dla danego
        uzytkownika. Co pozwoli na zapamietanie wyniku do
        jego pozycji w tabeli najlepszego wyniku. */
        int uzytkownik = wybierzUzytkownika();

        clearDisplay();         // Czyszczenie ekranu
        /* Pierwsza linia
           - BTN1->START */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('1');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('S');
        SEND_CHAR('T');
        SEND_CHAR('A');
        SEND_CHAR('R');
        SEND_CHAR('T');
        /* Pierwsza linia
           - BTN4->POWROT */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('4');
        SEND_CHAR('-');
```

```
        SEND_CHAR('>');
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('W');
        SEND_CHAR('R');
        SEND_CHAR('O');
        SEND_CHAR('T');

        while (1) {
                if (!PRZYCISK3) {
                        /* Przycisk 4
                        przenosi do menu
                        startowego gry */
                        opoznienie();
                        menuStartowe();
                } else if (!PRZYCISK0) {
                        /* Przycisk 0 przenosi do
                        faktycznej rozgrywki poprzez
                        funkcje sekwencja pobierajca
                        danego uzytkownika okreslanego
                        wartoscia zmiennej uzytkownik */
                        opoznienie();
                        sekwencja(uzytkownik);
                }
        }
}

/* Faktyczna rozgrywka:
   - Generuje sekwencje kwadratów
   - Prosi gracza o jej powtórzenie
   - Udane powtórzenie dodaje punkt
     do zmiennej wynik
   - Porazka pokazuje wynik i zapisuje
     go w najwiekszych wynikach, lecz tylko
     w przypadku gdy ten wynik jest większy
     od dotychczasowego zapisanego w tablicy */
void sekwencja(int gracz) {
        int wynik = 0;          // Przechowuje wynik gracza podczas rozgrywki
        int iteracje = 1;       // Liczba iteracji jakich dokonuje gra

        /* Nieskonczona petla rozgrywki */
        while (1) {
                sekwencjaProgramu[wynik] = generujLosowaLiczbe();       //
Zapamietanie sekwencji

                /* Pętla for generuje sekwencje zapisując ją do tablicy
                z sekwencja programu i ukazuje wizualnie kwadraty tak aby
                uzytkowik mial okazje ja zapamietac */
                for (int i = 0; i < iteracje; i++) {
                        opoznienie();
                        opoznienie();
                        clearDisplay();
                        narysujKwadrat(sekwencjaProgramu[i]);
                        opoznienie();
                        opoznienie();
```

```
                clearDisplay();
                opoznienie();
                opoznienie();
        }

        clearDisplay();

        /* Pierwsza linia
           - PODAJ */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('D');
        SEND_CHAR('A');
        SEND_CHAR('J');

        /* Druga linia
           - SEKWENCJE */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('S');
        SEND_CHAR('E');
        SEND_CHAR('K');
        SEND_CHAR('W');
        SEND_CHAR('E');
        SEND_CHAR('N');
        SEND_CHAR('C');
        SEND_CHAR('J');
        SEND_CHAR('E');

        opoznienie();

        for (int j = 0; j < iteracje; j++) {
                while (1) {
                        /* Tu zapisujemy sekwencje uzytkownika na ture*/
                        if (!PRZYCISK0) {
                                opoznienie();
                                sekwencjaUzytkownika[j] = 1;
                                break;
                        } else if (!PRZYCISK1) {
                                opoznienie();
                                sekwencjaUzytkownika[j] = 2;
                                break;
                        } else if (!PRZYCISK2) {
                                opoznienie();
                                sekwencjaUzytkownika[j] = 3;
                                break;
                        } else if (!PRZYCISK3) {
                                opoznienie();
                                sekwencjaUzytkownika[j] = 4;
                                break;
                        }
                }

                /* Kontrola - czy sekwencja uzytkownika zgadza sie z
sekwencja programu? */
```

```
                        if (sekwencjaUzytkownika[j] != sekwencjaProgramu[j]) {
                                clearDisplay();

                                SEND_CHAR('W');
                                SEND_CHAR('Y');
                                SEND_CHAR('N');
                                SEND_CHAR('I');
                                SEND_CHAR('K');
                                SEND_CHAR(':');
                                SEND_CHAR(wynik / 10 + 48);
                                SEND_CHAR(wynik % 10 + 48);

                                /* Wiekszy wynik jest zapisywany w tabeli */
                                if (wynik > wyniki[gracz]) {
                                        wyniki[gracz] = wynik;
                                }

                                for (int i = 0; i < 5; i++) {
                                        opoznienie();
                                }

                                menuStartowe();
                        }
                }

                /* Udana proba inkrementuje zmienna z wynikiem i iteracje */
                iteracje++;
                wynik++;
        }
}

void pokazWynik(int gracz) {
        /* Czyszczenie ekranu */
        clearDisplay();
        /* Pierwsza linia
           -> PLAYER(odpowiedni numer)-> (wynik) */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR((gracz + 48) + 1);
        SEND_CHAR('-');
        SEND_CHAR('>');
        /* Jako ze gra ma do 50 poziomow
        pokazujemy jednie pierwsza cyfre wyniku i druga */
        SEND_CHAR((wyniki[gracz] / 10) + 48);   // Pierwsza cyfra wyniku
        SEND_CHAR((wyniki[gracz] % 10) + 48);   // Druga cyfra wyniku
        /* Druga linia
           - BTN4->POWROT */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
```

```
        SEND_CHAR('N');
        SEND_CHAR('4');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('W');
        SEND_CHAR('R');
        SEND_CHAR('O');
        SEND_CHAR('T');
        /* Wprowadzenie opoznienia zapobiegajacego kolizjom */
        opoznienie();
        while (1) {
                if (!PRZYCISK3) {
                        menuStartowe();
                }
        }
}

void pokazWyniki(void) {
        /* Czyszczenie ekranu */
        clearDisplay();
        /* Pierwsza linia
           - PLAYER 1
           - PLAYER 2 */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('1');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('2');
        /* Druga linia
           - PLAYER 3
           - PLAYER 4 */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('3');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
```

```
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('4');

        while (1) {
                if (!PRZYCISK0) {
                        pokazWynik(0);
                } else if (!PRZYCISK1) {
                        pokazWynik(1);
                } else if (!PRZYCISK2) {
                        pokazWynik(2);
                } else if (!PRZYCISK3) {
                        pokazWynik(3);
                }
        }
}

void opoznienie(void) {
        for (int i = 0; i < 10; i++)
                Delayx100us(250);
}

void wyslanieKwadratowDoPamieci(void) {
        for (unsigned int i = 0; i < 3; i++)
                for (unsigned int j = 0; j < 8; j++)
                        SEND_CHAR(uniq_chars[i][j]);
}

/* Generowanie pseudolosowej liczby od 1 do 4 */
int generujLosowaLiczbe(void) {
        /* Opoznienie aby nie powtorzyc wygenerowanej liczby */
        for (int i = 0; i < 3; i++) {
                opoznienie();
        }

        /* Zwrocenie liczby pseudolosowej. */
        return ((licznik % 4) + 1);
}

void narysujKwadrat(int numer) {
        /* Czyszczenie ekranu */
        clearDisplay();

        /* Polozenie sa przez nas ustalane tak aby jak najdokladniej
        odpowiadaly przyciskom pod ekranem. */
        switch (numer) {
        case 1:
                /* Wypisywanie pierwszego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 2; i++)
```

```
                        SEND_CHAR(10);
                break;
        case 2:
                /* Wypisywanie drugiego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 4; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 4; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        case 3:
                /* Wypisywanie trzeciego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 8; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 8; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        case 4:
                /* Wypisanie czwartego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 12; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 12; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        }
}

/* Procedura obsługi przerwania od TimerA */
#pragma vector = TIMERA0_VECTOR
__interrupt void Timer_A(void) {
        ++licznik;              // Licznik czasu inkrementowany co jedną sekundę
od startu obsługi przerwań
        _BIC_SR_IRQ(LPM3_bits); // wyjście z trybu LPM3
}
```

Prototypy funkcji wykorzystywanych w programie

```
/* Prototypy funkcji */
void inicjalizacjaPrzerwan(void);       // Funkcja inicjalizujaca przerwania
potrzebne do generatora liczb pseudolosowych
void pokazWynik(int);                   // Funkcja pokazujaca wynik danego
gracza (przyjmuje parametr w postaci numeru gracza)
void opoznienie(void);                  // Funkcja wprowadzajaca opoznienie
potrzebne do unikniecia kolizji r�norakich funkcjonalno�ci np. przyciskow
void menuStartowe(void);                // Funkcja uruchamiająca menu startowe
programu
void rozgrywka(void);                   // Funkcja rozpoczynajaca rozgrywke
void sekwencja(int);                    // Funkcja generujaca sekwencje i
przepytujaca gracza
int generujLosowaLiczbe(void);          // Funkcja generujaca pseudolosowa
liczbe od 1 do 4
void narysujKwadrat(int);               // Funkcja rysujaca kwadrat na
wyswietlaczu
void inicjalizacjaPrzyciskow(void);     // Inicjalizacja przyciskow
void pokazWyniki(void);                 // Funkcja obslugujaca menu wyboru
wyniku
void wyslanieKwadratowDoPamieci(void);  // Funkcja wysylajaca kwadraty do
pamieci
```

Funkcja główna

```
/* Funkcja główna */
void main(void) {
        inicjalizacjaPrzerwan();        // inicjalizacja przerwan
        inicjalizacjaPrzyciskow();      // inicjalizacja przycisko

        WDTCTL = WDTPW + WDTHOLD;       // zatrzymanie WDT

        InitPortsLcd();                 // inicjalizacja port�w
        InitLCD();                      // inicjalizacja LCD
        clearDisplay();                 // czyszczenie LCD
        SEND_CMD(CG_RAM_ADDR);          // mozliwosc wprowadzenia znakow
specjalnych
        wyslanieKwadratowDoPamieci();

        // Start programu
        menuStartowe();
}
```

Program rozpoczyna się od inicjalizacji przerwań potrzebnych do generowania liczb pseudolosowych, następnie inicjalizuje przyciski wymagane do odpowiadania na sekwencje. Używamy także funkcji do wysłania specjalnych znaków wykorzystywanych do wyświetlania sekwencji. Standardową procedurą przy projektach z wykorzystaniem jest także zatrzymanie WatchDog Timera co zostało zrealizowane poprzez polecenie:

```
WDTCTL = WDTP + WDTHOLD
```

Kolejne 4 linie dotyczą funkcji potrzebnych do poprawnej obsługi wyświetlacza takich jak:

- Inicjalizacja portów
- Inicjalizacja LCD
- Czyszczenie LCD
- Możliwość wprowadzania znaków specjalnych

Po zrealizowaniu tych procedur przechodzimy do funkcji menuStartowe, zanim omówimy dalsze tajniki omówmy po kolei każdą z zastosowanych tu funkcji.

Inicjalizacja przerwań

Aby każdy etap sekwencji był unikalny potrzebujemy tzw. "nasienia" do pobrania czasu. Nie mamy możliwości skorzystania z czasu komputera dlatego jesteśmy zmuszeni wykorzystać mechanizm mikrokontrolera zwany obsługą przerwań. W czasie wykonywania programu mikrokontroler po jakimś czasie ustalonym przez użytkownika przerwie swoją pracę i wykona jakąś czynność aby następnie powrócić do poprzednio wykonywanego zadania. W naszym przypadku czynnością będzie inkrementacja licznika.

```
/* Procedura obsługi przerwania od TimerA */
#pragma vector = TIMERA0_VECTOR
__interrupt void Timer_A(void) {
        ++licznik;              // Licznik czasu inkrementowany co jedną sekundę
od startu obsługi przerwań
        _BIC_SR_IRQ(LPM3_bits); // wyjście z trybu LPM3
}
```

```
void inicjalizacjaPrzerwan(void) {
        /* Basic Clock Module ustawiamy na ACLK(zegar 8 MHz ) i dzielimy
częstotliwość przez 2 (4 MHz) */
        BCSCTL1 |= XTS; // ACLK = LFXT1 = HF XTAL 8MHz

        do {
                IFG1 &= ~OFIFG;                                 // Czyszczenie
flgi OSCFault
                for (unsigned int i = 0xFF; i > 0; i--);        // odczekanie
        }
        while ((IFG1 & OFIFG) == OFIFG);                        // dopóki
OSCFault jest ciągle ustawiona

        BCSCTL1 |= DIVA_1;                                      // ACLK=8
MHz/2=4 MHz
        BCSCTL2 |= SELM0 | SELM1;                               // MCLK= LFTX1
=ACLK

        /* Timer_A  ustawiamy na 500 kHz
        a przerwanie generujemy co 100 ms */
        TACTL = TASSEL_1 + MC_1 + ID_3;                         // Wybieram
ACLK, ACLK/8=500kHz,tryb Up
        CCTL0 = CCIE;                                           // włączenie
przerwań od CCR0
        CCR0 = 50000;                                           // podzielnik
50000: przerwanie co 100 ms

        _EINT();                                                // włączenie
przerwań
```

```
}
```

Inicjalizacja przycisków

Do obsługi przycisków warto zdefiniować pewne stałe określające dane przyciski mikrokontrolera. Użyliśmy do tego dyrektyw define języka C.

```
/* Dyrektywy define */
#define PRZYCISK0 (P4IN & BIT4) // Definicja przycisku nr. 1
#define PRZYCISK1 (P4IN & BIT5) // Definicja przycisku nr. 2
#define PRZYCISK2 (P4IN & BIT6) // Definicja przycisku nr. 3
#define PRZYCISK3 (P4IN & BIT7) // Definicja przycisku nr. 4
```

Ale to nie wszystko. Kolejnym krokiem do poprawnego działania przycisków jest zmiana stanów napięcia rejestrów za nie odpowiadających. Do tego właśnie służy napisana przez nas funkcja inicjalizacjaPrzyciskow.

```
/* Inicjalizacja przycisków */
void inicjalizacjaPrzyciskow() {
        P4DIR &= ~BIT4;         // Stan wysoki bitu4 rejestru P4DIR
        P4DIR &= ~BIT5;         // Stan wysoki bitu5 rejestru P4DIR
        P4DIR &= ~BIT6;         // Stan wysoki bitu6 rejestru P4DIR
        P4DIR &= ~BIT7;         // Stan wysoki bitu7 rejestru P4DIR
}
```

Na tym etapie możemy wykrywać wciśnięcia przycisków za pomocą zwykłego wyrażenia warunkowego z zaprzeczeniem odpowiednich portów.

Wysłanie kwadratów do pamięci

Na samym początku jedyne znaki jakie możemy wyświetlać to znaki z tabeli ascii. Dodaliśmy dodatkowe znaki aby móc wyświetlać kwadraty. Na sam początek zainicjalizowaliśmy dwuwymiarową tablicę znaków zawierającą znaki.

```
/* Tablica zawierająca znaki unikalne. (Kwadraty) */
unsigned char uniq_chars[3][8] = {
        {31, 31, 31, 31, 31, 31, 31, 31},
        {31, 31, 31, 31, 31, 31, 31, 31},
        {31, 31, 31, 31, 31, 31, 31, 31}
};
```

Kwadraty następnie wysłaliśmy do pamięci urządzenia.

```
void wyslanieKwadratowDoPamieci(void) {
        for (unsigned int i = 0; i < 3; i++)
                for (unsigned int j = 0; j < 8; j++)
                        SEND_CHAR(uniq_chars[i][j]);
}
```

Dzięki temu odwołując się do wartości 10 przy wysyłaniu znaku do wyświetlenia wyślemy cały zamalowany znak. Cztery takie znaki w polach wygenerują kwadrat ale do tego przejdziemy później. Funkcje wyświetlacza pochodzą ze specjalnych bibliotek LCD do ich zrozumienia można zajrzeć do poprzednich sprawozdań zrealizowanych w tym celu. Zezwolenie na przesyłanie znaków wywołuję się poleceniem:

```
      SEND_CMD(CG_RAM_ADDR);          // mozliwosc wprowadzenia znakow
specjalnych
```

W celu krótkich wyjaśnień co robią poszczególne linie z bibliotek wyświetlacza zamieszczam pliki nagłówkowe z komentarzami wyjaśniającymi ich działanie.

Zawartość pliku nagłówkowego lcd.h:

```
void InitLCD(void);
void clearDisplay();
void SEND_CHAR (unsigned char c);
void SEND_CMD (unsigned char e);
void MAKE_DEFINED_CHAR(unsigned char c);
void Delayx100us(unsigned char b);
```

Zawartość pliku nagłówkowego portyLcd.h:

```
#define          LCD_Data           P2OUT
#define          _100us             100              //
#define          _10us              10               //
#define          E                  3                // P2.3
#define          RS                 2                // P2.2

#define     CLR_DISP        0x01           // clear display
#define     CUR_HOME        0x02           // return home
#define     ENTRY_INC       0x06           // entry mode increment
#define     ENTRY_INC_ROL   0x07           // entry mode increment with rol
data
#define     ENTRY_DEC       0x04           // entry mode decrement
#define     ENTRY_DEC_ROL   0x05           // entry mode decrement witch rol
dat
#define     DISP_OFF        0x08           // all display off
#define     DISP_ON         0x0c           // all display on

#define         DATA_ROL_LEFT          0x18           // rol data left
#define         DATA_ROL_RIGHT         0x1c           // rol data right
#define         CUR_SHIFT_LEFT         0x10           // shift coursor left
#define         CUR_SHIFT_RIGHT        0x14           // shift coursor right

#define     DD_RAM_ADDR     0x80           // set DD_RAM
#define     DD_RAM_ADDR2    0xc0           // set CG_RAM
#define     DD_RAM_ADDR3    0x28           //
#define     CG_RAM_ADDR     0x40           //

void InitPortsLcd(void);
```

Generowanie liczby pseudolosowej

```
/* Generowanie pseudolosowej liczby od 1 do 4 */
int generujLosowaLiczbe(void) {
        /* Opoznienie aby nie powtorzyc wygenerowanej liczby */
        for (int i = 0; i < 3; i++) {
                opoznienie();
        }

        /* Zwrocenie liczby pseudolosowej. */
        return ((licznik % 4) + 1);
}
```

Za wygenerowanie liczby pseudolosowej odpowiada funkcja "generujLosowaLiczbe", która wykonuje bardzo prosty mechanizm operacji modulo 4 na liczniku - zmiennej globalnej, która jest inkrementowana przez obsługę przerwań. Jednak po wykonaniu samej operacji modulo 4 będziemy otrzymywać liczby z zakresu od 0 do 3, dlatego też na sam koniec musimy dodać jedynkę do wyniku naszej operacji. Tak wykonana operacja zostaje wracana przez tę funkcję co później posłuży nam jako definicja potrzebnego "kwadratu" w sekwencji.

Rysowanie kwadratów

Za rysowanie odpowiednich (od 1 do 4) kwadratów odpowiada dosyć długa choć prosta funkcja "narysujKwadrat". Funkcja jako parametr przyjmuję numer kwadratu jaki chcemy aby został narysowany na wyświetlaczu.

```
void narysujKwadrat(int numer) {
        /* Czyszczenie ekranu */
        clearDisplay();

        /* Polozenie sa przez nas ustalane tak aby jak najdokladniej
        odpowiadaly przyciskom pod ekranem. */
        switch (numer) {
        case 1:
                /* Wypisywanie pierwszego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        case 2:
                /* Wypisywanie drugiego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 4; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 4; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
```

```
                break;
        case 3:
                /* Wypisywanie trzeciego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 8; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 8; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        case 4:
                /* Wypisanie czwartego kwadratu od lewej */
                SEND_CMD(DD_RAM_ADDR);
                for (unsigned int i = 0; i < 12; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                SEND_CMD(DD_RAM_ADDR2);
                for (unsigned int i = 0; i < 12; i++)
                        SEND_CHAR(' ');
                for (unsigned int i = 0; i < 2; i++)
                        SEND_CHAR(10);
                break;
        }
}
```

Funkcja składa się z warunku wielokrotnego wyboru języka C jakim jest switch, który w zależności od wartości parametru numer jaki uzyskuję wykonuję inne instrukcje.

Przyjmijmy, że nasza funkcja uzyskuje parametr o wartości 4. Przed wykonaniem instrukcji odpowiednich dla tego przypadku zostanie wykonana komenda czyszcząca wyświetlacz, następnie zostaną uruchomione odpowiednie pętle rysujące białe znaki i znak naszego wcześniej zdefiniowanego znaku własnego jakim jest kwadrat. Dla przypadku 4 będzie to kwadrat wysunięty najbardziej na prawo. Każdy przypadek dzięki odpowiednim manipulacjom ze znakiem białym i znakiem kwadratu powoduje, że narysowany zostanie kwadrat o innym położeniu odpowiadającemu przyciskiem pod nim.

Implementacja rozgrywki

Wymieniliśmy i opisaliśmy już wszystkie potrzebne metody, zmienne do prawidłowego przeprowadzenia rozgrywki, zajmijmy się, więc teraz samą rozgrywką. Pętlę główną programu kończy funkcja menuStartowe zajmijmy się więc nią.

menuStartowe

```
void menuStartowe(void) {
        clearDisplay();

        /* Pierwsza linia
           - BTN1 -> GRA */
```

```
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('1');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('G');
        SEND_CHAR('R');
        SEND_CHAR('A');

        /* Druga linijka
           - BTN2 -> WYNIKI */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('2');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('W');
        SEND_CHAR('Y');
        SEND_CHAR('N');
        SEND_CHAR('I');
        SEND_CHAR('K');
        SEND_CHAR('I');

        while (1) {
                if (!PRZYCISK0) {
                        /* Rozpoczecie rozgrywki */
                        opoznienie();   // Wprowadzenie opoznienia
zapobiegajacego kolizjom
                        rozgrywka();    // Przejscie do rozgrywki
                } else if (!PRZYCISK1) {
                        /* Pokazanie najlepszych wynikow graczy */
                        opoznienie();   // Wprowadzenie opoznienia
zapobiegajacego kolizjom
                        pokazWyniki();  // Przejscie do menu z najlepszymi
wynikami graczy
                }
        }
}
```

Funkcja menu startowe czyści ekran wyświetlacza i wypisuje odpowiednie informacje jakie są potrzebne graczowi do podjęcia następnych kroków.

- BTN1 -> GRA
- BTN2 -> WYNIKI

Ponownie do wypisywania tych informacji używamy możliwości biblioteki wyświetlacza czyli funkcji SEND_CMD, która wysyła do wyświetlacza odpowiednie polecenie. Najczęściej przez nas wykorzystywanymi będą:

- SEND_CMD(DD_RAM_ADDR) **<- Umożliwia pisanie w pierwszej linii**
- SEND_CMD(DD_RAM_ADDR2) **<- Umożliwia pisanie w drugiej linii**

- SEND_CMD(*some_kind_of_char*) **<- Umożliwia wysłanie znaku**

Po wypisaniu tych informacji przechodzimy do pętli nieskończonej while zawierającą instrukcje warunkowe, które w zależności od naciśnięcia wskazanego przycisku przeniesie nas do innej funkcji odpowiadającej za rozgrywkę bądź tabele wyników.

- Wciśnięcie pierwszego przycisku przeniesie nas do rozgrywki de facto do funkcji 'rozgrywka', wykonując przedtem opóźnienie za pomocą omawianej już przez nas funkcji "opoznienie".

Na sam start zajmijmy się warstwą logicznej tej prostszej funkcjonalności jaką jest pokazywanie wyników czyli ta wywoływana po naciśnięciu przycisku drugiego w menu startowym.

Pokazywanie wyników

Funkcja "pokazWyniki" jest jedynie interfejsem do pokazywania wyników mającym na celu wyświetlenie wszystkich graczy aby następnie przejść do funkcji 'pokazWynik(Int)' przjmującej wartość całkowitą w zależności od wyboru dokonanego w pierwotnie omawianej funkcji.

PokazWynik()

```
void pokazWynik(int gracz) {
        /* Czyszczenie ekranu */
        clearDisplay();
        /* Pierwsza linia
           -> PLAYER(odpowiedni numer)-> (wynik) */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR((gracz + 48) + 1);
        SEND_CHAR('-');
        SEND_CHAR('>');
        /* Jako ze gra ma do 50 poziomow
        pokazujemy jednie pierwsza cyfre wyniku i druga */
        SEND_CHAR((wyniki[gracz] / 10) + 48);   // Pierwsza cyfra wyniku
        SEND_CHAR((wyniki[gracz] % 10) + 48);   // Druga cyfra wyniku
        /* Druga linia
           - BTN4->POWROT */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('4');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('W');
        SEND_CHAR('R');
        SEND_CHAR('O');
        SEND_CHAR('T');
        /* Wprowadzenie opoznienia zapobiegajacego kolizjom */
        opoznienie();
```

```
        while (1) {
                if (!PRZYCISK3) {
                        menuStartowe();
                }
        }
}

void pokazWyniki(void) {
        /* Czyszczenie ekranu */
        clearDisplay();
        /* Pierwsza linia
           - PLAYER 1
           - PLAYER 2 */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('1');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('2');
        /* Druga linia
           - PLAYER 3
           - PLAYER 4 */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('3');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('4');

        while (1) {
                if (!PRZYCISK0) {
                        pokazWynik(0);
                } else if (!PRZYCISK1) {
                        pokazWynik(1);
                } else if (!PRZYCISK2) {
```

```
                        pokazWynik(2);
                } else if (!PRZYCISK3) {
                        pokazWynik(3);
                }
        }
}
```

Po wypisaniu nazw graczy na wyświetlaczu przechodzimy do pętli nieskończonej while w której odbywa się sprawdzenie tego jakiego wyboru dokonał użytkownik. Przyciskowi odpowiada użytkownik o tym samym numerze co przycisk choć zmniejszony o 1 na potrzeby poprawnego indeksowania. Czyli:

- Player1 -> Przycisk0
- Player2 -> Przycisk1
- Player3 -> Przycisk2
- Player4 -> Przycisk3

Czyli po wyborze przykładowo gracza 4 do funkcji pokazWynik zostanie przesłany parametr w postaci wartości całkowitej równej (4-1)=**3**.

Przechowywanie najlepszych wyników

Przechowywanie najlepszych wyników graczy odbywa się w jednowymiarowej, czteroelementowej tablicy liczb całkowitych, która jest zdefiniowana jako zmienna globalna.

```
int wyniki[4] = {0, 0, 0, 0};   // Tablica zawierajaca najwyzsze wyniki
```

Indeksowanie tablic w języku C zaczyna się od 0, dlatego też wybierając gracza 4 w menu wyboru wyników odwołujemy się do indeksu mniejsze o 1 czyli 3.

Pokazywanie najlepszego wyniku

Po odebraniu odpowiedniego parametru z interfejsu funkcji pokazWyniki() funkcja pokazWynik(Int) rozpoczyna swoje działanie, przyjrzyjmy się mu.

PokazWynik(Int)

```
void pokazWynik(int gracz) {
        /* Czyszczenie ekranu */
        clearDisplay();
        /* Pierwsza linia
           -> PLAYER(odpowiedni numer)-> (wynik) */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR((gracz + 48) + 1);
        SEND_CHAR('-');
        SEND_CHAR('>');
        /* Jako ze gra ma do 50 poziomow
        pokazujemy jednie pierwsza cyfre wyniku i druga */
```

```
        SEND_CHAR((wyniki[gracz] / 10) + 48);   // Pierwsza cyfra wyniku
        SEND_CHAR((wyniki[gracz] % 10) + 48);   // Druga cyfra wyniku
        /* Druga linia
           - BTN4->POWROT */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('4');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('W');
        SEND_CHAR('R');
        SEND_CHAR('O');
        SEND_CHAR('T');

        /* Wprowadzenie opoznienia zapobiegajacego kolizjom */
        opoznienie();

        while (1) {
                if (!PRZYCISK3) {
                        menuStartowe();
                }
        }
}
```

Wiedząc już z poprzednich wyjaśnień co oznaczają przesyłanie znaków (jeżeli nie to warto sprawdzić nagłówek menuStartowe gdzie zostało już to wyjaśnione), jedynym co może nas zaskoczyć to sposób wyświetlania wyniku. Z racji, że gra posiada do 50 poziomów (co można zauważyć na podstawie tablic sekwencji)

```
int sekwencjaProgramu[50];      // Tablica zawierajaca sekwencje programu
int sekwencjaUzytkownika[50];   // Tablica zawierajaca sekwencje uzytkownika
```

możemy ułatwić sobie wyświetlanie cyfr wyniku poprzez operację dzielenia przez 10 dla pierwszej cyfry i modulo 10 dla drugiej cyfry. To bardzo proste operacje pozwalające rozebrać liczbę na cyfry tak aby móc przesłać znak do wyświetlacza.

```
/* Jako ze gra ma do 50 poziomow pokazujemy jednie pierwsza cyfre wyniku i druga
*/
SEND_CHAR((wyniki[gracz] / 10) + 48);   // Pierwsza cyfra wyniku
SEND_CHAR((wyniki[gracz] % 10) + 48);   // Druga cyfra wyniku
```

Jeżeli rozpoczynamy dopiero co rozgrywkę i każdy gracz ma 0 punktów to prześledźmy działanie tych dwóch komend dla gracza o numerze 4 czyli parametr o wartości 3.

- (wyniki[3] / 10) + 48 -> 0 + 48 -> 48 -> w tabeli ascii oznacza 0
- (wyniki[3] % 10) + 48 -> 0 + 48 -> 48 -> w tabeli ascii oznacza 0

Zatem naszym oczom ukażę się "PLAYER4 -> 00". Przycisk 3 przeniesie nas z powrotem do menu startowego. Niedługo poznamy mechanizm zmiany najlepszego wyniku gracza.

Logika stojąca za rozgrywką

rozgrywka

Funkcja rozgrywka odpowiada za przejście do menu rozgrywki, które z kolei odpowiada za:

- wybór uzytkownia
- obsługę wyboru co do:
  - kontynuacji sesji
    - przenosi do funkcji sekwencja odpowiadającą za faktyczną rozgrywkę (wyświetlanie kwadratów i pytanie o wyświetloną sekwencje)
  - powrotu do menu
    - realizuje w ciele instrukcję warunkową przenoszącą gracza z powrotem do menu startowego

```
void rozgrywka(void) {
        /* Na sam start rozgrywki wybierany jest gracz
        poprzez funkcje wybierzUzytkownika uzytkownik
        przenoszony jest do menu z wyborem a po jego
        dokonaniu do zmiennej typu Integer przesyłana
        jest liczba calkowita odpowiednia dla danego
        uzytkownika. Co pozwoli na zapamietanie wyniku do
        jego pozycji w tabeli najlepszego wyniku. */
        int uzytkownik = wybierzUzytkownika();

        clearDisplay();         // Czyszczenie ekranu
        /* Pierwsza linia
           - BTN1->START */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('1');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('S');
        SEND_CHAR('T');
        SEND_CHAR('A');
        SEND_CHAR('R');
        SEND_CHAR('T');
        /* Pierwsza linia
           - BTN4->POWROT */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('B');
        SEND_CHAR('T');
        SEND_CHAR('N');
        SEND_CHAR('4');
        SEND_CHAR('-');
        SEND_CHAR('>');
        SEND_CHAR('P');
        SEND_CHAR('O');
        SEND_CHAR('W');
        SEND_CHAR('R');
```

```
        SEND_CHAR('O');
        SEND_CHAR('T');

        while (1) {
                if (!PRZYCISK3) {
                        /* Przycisk 4
                        przenosi do menu
                        startowego gry */
                        opoznienie();
                        menuStartowe();
                } else if (!PRZYCISK0) {
                        /* Przycisk 0 przenosi do
                        faktycznej rozgrywki poprzez
                        funkcje sekwencja pobierajca
                        danego uzytkownika okreslanego
                        wartoscia zmiennej uzytkownik */
                        opoznienie();
                        sekwencja(uzytkownik);
                }
        }
}
```

- Czwarty przycisk -> menuStartowe() poprzedzone wywołaniem opóźnienia
- Pierwszy przycisk -> sekwencja() wraz z uprzednio pobranym użytkownikiem (wartość odpowiadająca uzytkownikowi zostala pobrana do zmiennej o tym samym imieniu dzięki funkcji obsługującej ten wybór)

WybierzUzytkownika

```
int uzytkownik = wybierzUzytkownika();
```

Aby gra wiedziała dla jakiego użytkownika mają zostać przypisane punkty, wykorzystuję ona funkcję wybierzUzytkownika(). Poniżej znajduję się jej ciało.

```
int wybierzUzytkownika() {
        /* Pierwsza linia
           - PLAYER1 PLAYER 2 */
        SEND_CMD(DD_RAM_ADDR);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('1');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('2');
        /* Druga linia
```

```
            - PLAYER3 PLAYER 4 */
        SEND_CMD(DD_RAM_ADDR2);
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('3');
        SEND_CHAR(' ');
        SEND_CHAR('P');
        SEND_CHAR('L');
        SEND_CHAR('A');
        SEND_CHAR('Y');
        SEND_CHAR('E');
        SEND_CHAR('R');
        SEND_CHAR('4');

        while (1) {
                /* Przycisk zwraca liczbe calkowita
                odpowiadajaca odpowiedniumu graczu */
                if (!PRZYCISK0) {
                        opoznienie();
                        return 0;
                } else if (!PRZYCISK1) {
                        opoznienie();
                        return 1;
                } else if (!PRZYCISK2) {
                        opoznienie();
                        return 2;
                } else if (!PRZYCISK3) {
                        opoznienie();
                        return 3;
                }
        }
}
```

Funkcja ta podobnie jak przy wyborze najlepszego wyniku danego użytkownika wyświetla ich nazwy na ekranie a za wybór odpowiada przycisk o numerze odpowiadającemu użytkownikowi.

- Pierwszy przycisk -> zwraca pierwszego użytkownika (var Integer = 0)
- Drugi przycisk -> zwraca pierwszego użytkownika (var Integer = 1)
- Trzeci przycisk -> zwraca pierwszego użytkownika (var Integer = 2)
- Czwarty przycisk -> zwraca pierwszego użytkownika (var Integer = 3)

Faktyczna rozgrywka

sekwencja

Funkcja sekwencja pobiera parametr liczby całkowitej w postaci numeru gracza. Jest on do niej wysyłany z funkcji rozgrywka, gdzie wartość ta była przedstawiona jako zmienna użytkownik.

Nagłówek:

```
void sekwencja(int gracz); // <- Przechowuje numer gracza w zmiennej gracz
```

Zmienne potrzebne do rozgrywki:

```
int wynik = 0;          // Przechowuje wynik gracza podczas rozgrywki
int iteracje = 1;       // Liczba iteracji jakich dokonuje gra
```

Zmienne te są niezbędne do:

- zapisywania wyniku gracza w celu porównania go po rozgrywce z wynikiem już istniejącym w tabeli najlepszych wyników
- manipulacja pętli (iteracje określają ile kwadratów ma być pokazywanych)
- zmienna wynik służy także do odwoływania się do indeksu tablicy, gdyż w przeciwieństwie do iteracji wynosi ona o jeden mniej, czyli zero a to właśnie od tej wartości są indeksowane tablice w języku C.

Zajmijmy się nieskończoną pętlą i tym co w niej się dzieje:

- wygenerowanie losowej liczby dla tablicy sekwencji programu

```
sekwencjaProgramu[wynik] = generujLosowaLiczbe();       // Zapamietanie
sekwencji
```

- narysowanie ich na ekranie Msp430 odpowiednio z opóźnieniem specjalnie dobranym aby te wyświetlanie było dla gracza czytelne

```
/* Pętla for generuje sekwencje zapisując ją do tablicy
   z sekwencja programu i ukazuje wizualnie kwadraty tak aby
   uzytkowik mial okazje ja zapamietac */
for (int i = 0; i < iteracje; i++) {
    opoznienie();                   //<- dwa opoznienia przed czyszczeniem
ekranu
    opoznienie();

    clearDisplay();                 //<- czyszczenie ekranu
    narysujKwadrat(sekwencjaProgramu[i]);   //<- narysowanie kwadratu w pozycji
wylosowanej liczby

    opoznienie();   //<- kolejne dwa opoznienia
    opoznienie();

    clearDisplay(); //<- czyszczenie ekranu

    opoznienie();   //<- kolejne dwa opoznienia
    opoznienie();
}

clearDisplay(); //<- czyszczenie ekranu
```

- Użytkownik zostaje informowany o możliwości podania sekwencji napisem 'PODAJ SEKWENCJE', klasycznie przy użyciu metod z biblioteki wyświetlacza przesyłających znaki odpowiednio w pierwszej i drugiej linii wyświetlacza.

```
/* Pierwsza linia
   - PODAJ */
```

```
    SEND_CMD(DD_RAM_ADDR);
    SEND_CHAR('P');
    SEND_CHAR('O');
    SEND_CHAR('D');
    SEND_CHAR('A');
    SEND_CHAR('J');

/* Druga linia
   - SEKWENCJE */
    SEND_CMD(DD_RAM_ADDR2);
    SEND_CHAR('S');
    SEND_CHAR('E');
    SEND_CHAR('K');
    SEND_CHAR('W');
    SEND_CHAR('E');
    SEND_CHAR('N');
    SEND_CHAR('C');
    SEND_CHAR('J');
    SEND_CHAR('E');

    opoznienie();
```

- Pobieranie sekwencji od użytkownika i sprawdzanie na bieżąco czy zgadza się z sekwencją znajdująca się w tablicy sekwencjaProgramu. Po wykryciu odpowiedniego przycisku, wybór użytkownika jest zpaisywany do globalnej tablicy sewkencjaUzytkownika. Po każdym kliknięciu przycisku program sprawdza czy do tego momentu sekwencja jest prawidłowa, jeżeli nie jest to na krótką chwile wyświetlany jest wynik a gracz powraca do menu, ale przed tym sprawdzane jest czy jego wynik po przegranej jest większy od najlepszego wyniku w wynikach. W przypadku, gdy tak jest wartość poprzednia jest zastępowana wartością z rozgrywki.

```
 for (int j = 0; j < iteracje; j++) {
                        while (1) {
                                /* Tu zapisujemy sekwencje uzytkownika na ture*/
                                if (!PRZYCISK0) {
                                        opoznienie();
                                        sekwencjaUzytkownika[j] = 1;
                                        break;
                                } else if (!PRZYCISK1) {
                                        opoznienie();
                                        sekwencjaUzytkownika[j] = 2;
                                        break;
                                } else if (!PRZYCISK2) {
                                        opoznienie();
                                        sekwencjaUzytkownika[j] = 3;
                                        break;
                                } else if (!PRZYCISK3) {
                                        opoznienie();
                                        sekwencjaUzytkownika[j] = 4;
                                        break;
                                }
                        }
```

```
                        /* Kontrola - czy sekwencja uzytkownika zgadza sie z
sekwencja programu? */
                        if (sekwencjaUzytkownika[j] != sekwencjaProgramu[j]) {
                                clearDisplay();

                                SEND_CHAR('W');
                                SEND_CHAR('Y');
                                SEND_CHAR('N');
                                SEND_CHAR('I');
                                SEND_CHAR('K');
                                SEND_CHAR(':');
                                SEND_CHAR(wynik / 10 + 48);
                                SEND_CHAR(wynik % 10 + 48);

                                /* Wiekszy wynik jest zapisywany w tabeli */
                                if (wynik > wyniki[gracz]) {
                                        wyniki[gracz] = wynik;
                                }

                                for (int i = 0; i < 5; i++) {
                                        opoznienie();
                                }

                                menuStartowe();
                        }
                }

                /* Udana proba inkrementuje zmienna z wynikiem i iteracje */
                iteracje++;
                wynik++;
```

Co można by było poprawić? (Dalsza perspektywa)

- Optymalizacja pamięci poprzez dynamiczne jej alokowanie
- Ciekawszy design (ręcznie stworzone awatary użytkowników)
- Płynniejsze wyświetlanie sekwencji
- Komunikaty do gracza w trakcie rozgrywki
- Integracja funkcjonalności Githuba:
  - Automatyczne testy
  - Mechanizmy ułatwiające zgłasznie błędów
  - Fora
  - Dokumentacja
- Znalezienie lepszych nazw dla zmiennych (na ten moment są niespójne)

Podsumowanie

Tworzenie projektu gry było dla całej grupy świetną zabawą choć momentami wymagającą cierpliwości. Oceniając finalnie projekt uważamy że każdy podołał powierzonemu mu zadaniu choć obyło się bez wprowadzania korporacyjnych metodyk pracy.