Igra ,,Zmijica”

-projektna dokumentacija-

1.1 Osnovna logika idejnog rešenja i implementacije

Osnovna logika implementacije igre je sledeća:

1. Korisniku se prilikom prvog pokretanja igre prikazuje animacija i pozdravna poruka.

2. Potom se iscrtavaju dve prepreke na pozicijama navedenim u projektnom zadatku, a zmija kojom korisnik upravlja, prelazi u stanje kretanja DESNO i čeka komande igrača.

3. Igrač može da upravlja smerom kretanja zmije tasterima RC7 (skretanje glave zmije ulevo) i RC6 (skretanje zmije udesno).

4. Takodje, igrač može igru da zaustavi (pauzira) pritskom na taster RC5 (igra prelazi u stanje PAUZA), odnosno da je pokrene iz početka (restartuje) pritiskom na taster RC4.

5. Ukoliko se zmija sudari sa preprekom ili sa samom sobom, ekran se briše, igra prelazi u stanje KRAJ, korisniku se prikazuje slika, puštaju se četiri uzastopna zvučna signala, a nakon vremenske zdrške ekran se briše i korisniku se ispisuje poruka da je igra završena i da može ponovo da je igra pritiskom na tater RC4.

1.2 UML dijagram idejnog rešenja

Igra se sastoji od sedam stanja:

1. DESNO

2. GORE

3. LEVO

4. DOLE

5. POČETAK

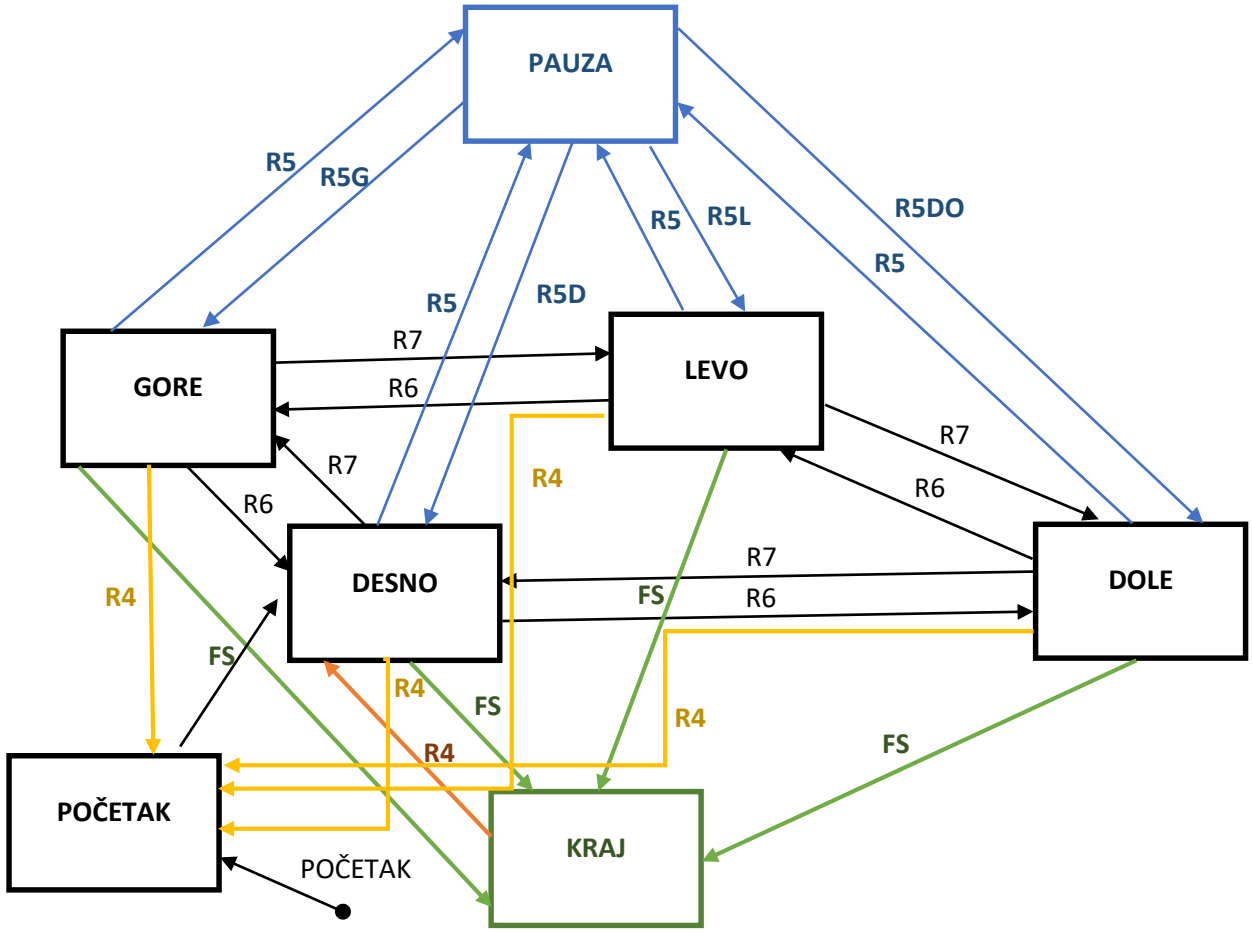
6. PAUZA

7. KRAJ

pri čemu se 5. stanje koristi samo za inicijalizaciju i igra odmah prelazi u stanje 1.

Tokom igre, korsnik može pritiskom dugmeta RC5 da iz bilo kog od prva četiri stanja pređe u stanje 6, nakon koga se, ukoliko ponovo pritisne taster RC5, vraća u stanje u kome se nalazio pre pritiska pomenutog tastera. Informacija o poslednjem stanju se neposredno posle zahteva za ulazak u stanje pauze čuva u posebnoj promenljivoj.

Ukoliko se tokom kretanja zmija sudari sa preprekom ili sa samom sobom, mašina stanja prelazi iz bilo kog trenutnog stanja u stanje KRAJ, iz koga se može preći u stanje DESNO ukoliko korisnik pritisne taster RC4 (reset igre). UML dijagram mašine stanja prikazan je na slici 1.



Slika 1. UML dijagram (dijagram konacnog automata) igre

Legenda:

R7 – detektovan pritisak dugmeta RC7 (naredba zmiji da skrene levo)
R6 – detektovan pritisak dugmeta RC6 (naredba zmiji da skrene desno)
R5 – detektovan pritisak dugmeta RC5 (pauziranje igre)
R5L – detektovan pritisak dugmeta RC5 i prethodno stanje mašine je bilo LEVO
R5D – detektovan pritisak dugmeta RC5 i prethodno stanje mašine je bilo DESNO
R5G– detektovan pritisak dugmeta RC5 i prethodno stanje mašine je bilo GORE
R5DO – detektovan pritisak dugmeta RC5 i prethodno stanje mašine je bilo DOLE
FS – fleg sudar (indikacija da je došlo do sudara)
R4 – detektovan pritisak dugmeta RC4 (reset igre)

1.3 Implementacija idejnog rešenja

Prvo definišemo promenljive tipa const code char koja predstavlja konstantni niz. U ovaj niz unosimo bitmapu slike koja treba da se prikaže korisniku. Niz je dužine 1024 bajta (128x64) gde se svaki bajt sastoji od osmobitnog registra čiji jedan bit predstavlja jedan piksel na GLCD-u.

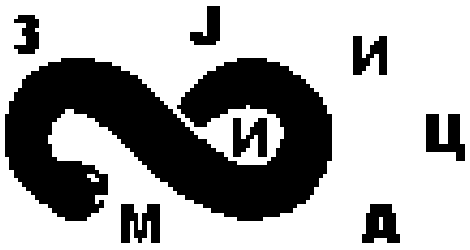
Neophodno je da učitamo sliku veličine 128x64 koja će se prikazati na kraju igre.
Prikazujemo je na slici 2.:

Slika 2. Slika koja se prikazuje na kraju igre

zatim jednu animaciju koja se sastoji od 10 uzastopnih frejmova (Slika 3.):

Slika 3. Treći frejm animacije

kao i sliku koja će se prikazati nakon završetka animacije (pozdravna poruka). Prikazujemo je na slici 4.:



Slika 4. Pozdravna slika pre prvog igranja igre

Slike su prethodno pretvorene u 128x64 monohromatske bitmape. Izvorna animacija je takođe prvo pretvorena u 128x64 animaciju a zatim je rastavljena na frejmove.

Generisanje samih bitmapa vršimo uz pomoć Mikroelektronikinog softverskog alata GLCD Bitmap Editor gde za vrstu ekrana biramo 128x64 (KS0108).

Generisan kod koristimo u programu.

Dakle, bitmape naših slika definišemo kao:

//Bitmape slika za prikaz na GLCD-u

//Slika koja se prikazuje na kraju igre

const code char slikaKrajIgre[1024] = {0, 0, 0, ...};

//Deset bitmapa uzastopnih frejmova (slika) animacije za prikaz pre prvog pokretanja igre

```
const code char s1[1024] = { 0, 0, 0, ...}; //ovde je kod skracen zarad ustede slobodnog mesta

const code char s2[1024] = { 0, 0, 0, ...};

const code char s3[1024] = { 0, 0, 0, ...};

const code char s4[1024] = { 0, 0, 0, ...};

const code char s5[1024] = { 0, 0, 0, ...};

const code char s6[1024] = { 0, 0, 0, ...};

const code char s7[1024] = { 0, 0, 0, ...};

const code char s8[1024] = { 0, 0, 0, ...};

const code char s9[1024] = { 0, 0, 0, ...};

const code char s10[1024] = { 0, 0, 0, ...};

//Poslednja slika za prikaz posle animacije pre prvog pokretanja igre

const code char s11[1024] = {0, 0, 0, ...};

const int k1 = 100; //konstanta koja se koristi za vremensku zadrsku (delay) u animaciji na pocetku igre
```

Zatim definišemo stanja tj. simbolicke oznake zarad bolje jasnosti koda:

```
//Definisanje stanja konacnog automata igre

#define POCETAK 0

#define DESNO 1

#define GORE 2

#define DOLE 3

#define LEVO 4

#define PAUZA 5

#define KRAJ 6

//Definisanje promenljivih za inicijalizaciju GLCD-a

char GLCD_DataPort at PORTD;        //deifinisanje porta za protok podataka od i ka GLCD-u


//Definisanje promenljivih tipa sbit koji nam omogucava pristup pojedinacnom bitu unutar registra

sbit GLCD_CS1 at LATB0_bit;         //definisanje Chip Select 1 linije

sbit GLCD_CS2 at LATB1_bit;         //definisanje Chip Select 2 linije

sbit GLCD_RS at LATB2_bit;          //definisanje promenljive za izbor registra

sbit GLCD_RW at LATB3_bit;          //promenljiva za odabir read (R) odnosno write (W) magistrale
```

sbit GLCD_EN at LATB4_bit; //definisanje promenljive za omogucavanje tj. setovanje bita EN GLCD-a

sbit GLCD_RST at LATB5_bit; //definisanje promenljive kojom se omogucuje reset GLCD-a

sbit GLCD_CS1_Direction at TRISB0_bit; //definisanje promenljive kojom definisemo da li je Chip Select 1 pin ulaznog odnosno izlanog tipa

sbit GLCD_CS2_Direction at TRISB1_bit; //definisanje promenljive kojom definisemo da li je Chip Select 2 pin ulaznog odnosno izlanog tipa

sbit GLCD_RS_Direction at TRISB2_bit; //definisanje promenljive kojom biramo registar

sbit GLCD_RW_Direction at TRISB3_bit; //definisanje promenljive kojom definisemo da li je pin ulazni (Write) ili izlazni (Read)

sbit GLCD_EN_Direction at TRISB4_bit; //deifnisanje promenljive kojom definisemo da li je Enable pin ulazni odnosno izlazni

sbit GLCD_RST_Direction at TRISB5_bit; //definisanje promenljive kojom definisemo da li je Reset pin ulaznog odnosno izlaznog tipa

//kraj inicijalizacije GLCD-a

Potom definišemo promenljive koje koristimo u kodu i definišemo njihov tip u zavisnosti od predviđenog opsega tj. skupa vrednosti koje mogu da imaju:

unsigned short zmijaSmer; //promenljiva u kojoj se cuva prethodno stanje zmije

unsigned short stanje; //trenutno stanje zmije

int zmijaX[20]; //promenljiva u kojoj se cuva vlicina zmije po X osi

int zmijaY[20]; //promenljiva u kojoj se cuva vlicina zmije po Y osi

//definisanje parametara prepreke 1

unsigned short prepreka1_X; //x koordinata prve tacke prepreke 1

unsigned short prepreka1_Y; //y koordinata prve tacke prepreke 1

unsigned short prepreka1_stranica; //promenjiva u kojoj se cuva duzina stranice prepreke 1

//definisanje parametara prepreke 2

unsigned short prepreka2_X; //x koordinata prve tacke prepreke 2

unsigned short prepreka2_Y; //y koordinata prve tacke prepreke 2

unsigned short prepreka2_stranica; //promenjiva u kojoj se cuva duzina stranice prepreke 2

bit staroStanje7; //promenljiva u kojoj se cuva informacija o tome da li je dugme RC7 pritisnuto/otpusteno

```
bit staroStanje6; //promenljiva u kojoj se cuva informacija o tome da li je dugme RC6 pritisnuto/otpusteno

bit staroStanje5; //promenljiva u kojoj se cuva informacija o tome da li je dugme RC5 pritisnuto/otpusteno

bit staroStanje4; //promenljiva u kojoj se cuva informacija o tome da li je dugme RC4 pritisnuto/otpusteno


bit flag7;          //promenljiva u kojoj se cuva informacija o tome da je dugme RC7 pritisnuto (komanda za upravljanje zmijicom LEVO

bit flag6;          //promenljiva u kojoj se cuva informacija o tome da je dugme RC6 pritisnuto (komanda za upravljanje zmijicom DESNO

bit flagKraj;       //promenljiva u kojoj se cuva informacija o tome da li je kraj igre ili ne

bit flagPauza;   //promenljiva u kojoj se cuva informacija o tome da li igra pauzirana ili ne

bit flagSudar;   //promenljiva u kojoj se cuva informacija o tome da li je doslo do sudara ili ne

bit flagPrekid;  //interna promenljiva u kojoj se cuva informacija o nastanku prekida (interrupt)

bit flagZmija;   //indikacija sudara  glave zmije

short i,j,k;        //interne promenljive za petlje
```

Potom prelazimo na definisanje funkcija naše igre.

Prvo definišemo funkciju prekidne rutine. Ova funkcija se poziva na svakih 50ms (usled pojave prekida koji potiče od tajmera TMR0) u kojoj resetujemo TMR0IF_bit (koji ukazuje na nastanak prekida) kao i gde setujumo naš interni fleg prekida flagPrekid koji koristimo u daljoj obradi prekida.

```
void interrupt(){
 if(TMR0IF_bit){    //Ako je nastao prekid (interrupt) od TMR0 (TMR0IF_bit = 1)
   TMR0IF_bit = 0; //resetuj fleg prekida TMR0IF_bit na 0
   TMR0H = 0x3C;  //konfigurisi TMR0 (ulazi se na svakih 50ms u masinu stanja)
   TMR0L = 0xB0;  //konfigurisi TMR0
   flagPrekid = 1;   //setuj interni fleg prekida
 }
```

Zatim definišemo funkciju uvodniEkran() kojom inicijalizujemo naš GLCD, puštamo uvodnu animaciju i na kraju prikazujemo pozdravnu sliku.

```
void uvodniEkran(){
 Glcd_init();         //Inicijalizuje se GLCD
 Glcd_Fill(0x00);   //Brisanje ekrana
```

```
Glcd_image(s1);  //Iscrtavanje prvog frejma
Delay_ms(k1);    //Vremenska zadrska od k1 ms
Glcd_Fill(0x00);   //Brisanje prethodne slike za iscrtavanje sledece
//gore opisani postupak se ponavlja deset puta
Glcd_image(s2);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s3);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s4);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s5);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s6);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s7);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s8);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s9);
Delay_ms(k1);
Glcd_Fill(0x00);
Glcd_image(s10);
Delay_ms(k1);
Glcd_Fill(0x00);
```

```
 Glcd_image(s11);    //Iscrtavanje poslednje slike (ne frejma iz animacije) sa nazivom igre
 Delay_ms(20*k1);   //k1 dvadesetrostruka vremenska zadrska
 Glcd_Fill(0x00);
 //animacija kojom se slika pomera na dole sa ekrana
 for (i=0;i<64;i++){
   Glcd_PartialImage(0,5*i,128,64,128,64,s11);  //pomeraj sliku na dole po Y osi ekrana sa korakom
5
   Glcd_Fill(0x00);
   }
}
```

Potom definišemo funkciju inicijalizacija(). Ova funkcija ima zadatak da resetuje sve flegove, inicijalizuje niz u kome smeštamo X i Y koordinate tela zmije i iscrta prepreke:

```
void inicijalizacija(){
 flag7 = 0;
 flag6 = 0;
 flagZmija = 0;
 flagPauza = 0;
 flagKraj = 1;          //ovde je iskoriscena inverzna logika
 staroStanje4 = 0;
 staroStanje5 = 0;
 staroStanje6 = 0;
 staroStanje7 = 0;
 //inicijalizacija niza u kom cuvamo nasu zmiju
 for( i = 19; i >= 0; i-- ){
   zmijaX[i]=0;
   zmijaY[i]=0;
 }
 Glcd_Init();                  //Inicijalizacija GLCD-a
 Glcd_Fill(0x00);              //brisanje ekrana
 Glcd_Rectangle(prepreka1_X, prepreka1_Y, (prepreka1_X + prepreka1_stranica), (prepreka1_Y +
prepreka1_stranica), 1);  //iscrtavanje prepreke 1
```

```
 Glcd_Box(prepreka2_X, prepreka2_Y, (prepreka2_X + prepreka2_stranica), (prepreka2_Y + prepreka2_stranica), 1);  //iscrtavanje prepreke 2
}
```

Pretposlednja funkcija koju definišemo je funkcija krajIgre(). Ukoliko se zmija sudari sa preprekom ili sa samom sobom, ova funkcija se poziva. Ona iscrtava sliku za kraj igre, pušta četiri uzastopna zvučna signala (zvučno obaveštenje korisniku o završetku igre) i na kraju ispisuje tekstualnu poruku korisniku sa informacijom o tome da je igra završena i da se može ponovo pokrenuti pritiskom na taster RC4.

```
//funkcija koja se poziva prilikom kraja igre
void krajIgre(){
 Glcd_Fill(0x00);                    //obrisi ekran
 Glcd_image(slikaKrajIgre);          //iscrtaj sliku za kraj igre
 Sound_Play(784, 250);               //pusti zvucni signal ucestanosti 784Hz u trajanju od 250ms
 Sound_Play(698, 250);               //pusti zvucni signal ucestanosti 698Hz u trajanju od 250ms
 Sound_Play(659, 250);               //pusti zvucni signal ucestanosti 659Hz u trajanju od 250ms
 Sound_Play(630, 700);               //pusti zvucni signal ucestanosti 630Hz u trajanju od 250ms
 Delay_ms(3000);                     //napravi vremensku zadrsku od ~3s
 Glcd_Fill(0x00);                    //obrisi ekran
 Glcd_Write_Text("IGRA ZAVRSENA!",25,2,1);   //ispisi poruku "IGRA ZAVRSENA!"
 Glcd_Write_Text("Pritisnite",32,5,1);       //ispisi poruku "Pritisnite"
 Glcd_Write_Text("RC4 za restart!",32,6,1);  //ispisi poruku "RC4 za restart!"
 flagKraj = 0;                       //resetuj fleg za kraj igre
}
```

Na kraju definišemo glavnu petlju programa tj. funkciju main().

Vršimo konfiguraciju registara koji su nam neophodni za ispravan rad programa. Ovde je važno da napomenemo da se obavezno moraju dozvoliti globalni prekidi, prekidi izazvani tajmerom TMR0, kao i na koju ivicu (uzlaznu ili silaznu) detektujemo prekid. Takođe definišemo dimenzije prepreka i inicijalizujemo zvuk, tajmer i pozivamo funkciju za uvodnu animaciju.

```
//Glavna petlja programa
void main(){
 ANSELB = 0;      //Konfigurisi PORTB kao digitalni (za GLCD)
 ANSELD = 0;      //Konfigurisi PORTD kao digitalni (za GLCD)
```

```
ANSELC.B7 = 0; //Konfigurisi RC7 kao digitalni pin (upravljacko dugme)
ANSELC.B6 = 0; //Konfigurisi RC6 kao digitalni pin (upravljacko dugme)
ANSELC.B5 = 0; //Konfigurisi RC5 kao digitalni pin (upravljacko dugme)
ANSELC.B4 = 0; //Konfigurisi RC4 kao digitalni pin (upravljacko dugme)
ANSELC.B3 = 0; //Konfigurisi RC4 kao digitalni pin (upravljacko dugme)
TRISC.B7 = 1;    //Konfigurisi pin RC7 kao ulazni
TRISC.B6 = 1;    //Konfigurisi pin RC6 kao ulazni
TRISC.B5 = 1;    //Konfigurisi pin RC5 kao ulazni
TRISC.B4 = 1;    //Konfigurisi pin RC4 kao ulazni
TRISC.B3 = 0;    //Konfigurisi pin RC3 kao izlazni
T0CON = 0x80;              //TMR0 stvara prekid na svakih 50ms
TMR0H = 0x3C;              //konfigurise se tajmer
TMR0L = 0xB0;              //konfigurise se tajmer
GIE_bit = 1;               //dozvoljavaju se globalni prekidi
TMR0IE_bit = 1;            //dozvoljava se prekid izazvan tajmerom TMR0
INTEDG0_bit = 1;           //prekid se detektuje na usponsku ivicu (rising edge)
prepreka1_X = 54;          //X koordinata prepreke 1
prepreka1_Y = 22;          //Y koordinata prepreke 1
prepreka1_stranica = 20;  //duzina stranice prepreke 1
prepreka2_X = 10;          //X koordinata prepreke 2
prepreka2_Y = 10;          //Y koordinata prepreke 2
prepreka2_stranica = 10; //duzina stranice prepreke 2
stanje = POCETAK;          //postavljanje pocetnog stanja na POCETAK
Sound_Init(&PORTE, 1);   //inicijalizacija zvuka na RE1 (kratkospojnk na razvojnoj ploci se
postavlja u polozaj RE1)
uvodniEkran();             //pozivanje funkcije kojom se pusta uvodna animacija
```

Unutar ove funkcije definišemo while petlju u kojoj se naš program izvršava i izlazi jedino nakon spoljnog reseta.

Prvo pišemo uslovne provere kojima detektujemo da li su pritisnuta dugmad RC7, RC6, RC5 ili RC4.

Pritom, dugmad RC7, RC6 i RC4 bivaju zaključana ukoliko je mašina u stanju PAUZA. Za vreme trajanja stanja PAUZA na pomenutim dugmadima se ne detektuje pritisak. Ovo činimo kako

bismo osigurali da zmija nastavi da se kreće u istom smeru nakon pauze bez obzira na to da li je korisnik pritisnuo dugmad za promenu smera u toku pauzirane igre.

```
while(1){
    if(flagPauza == 0){                    //u slucaju da je flagPauza == 0 promena na RC7/RC6/RC4 se ne detektuje promena promenljive stanje
      if(Button(&PORTC, 7, 1, 1)){         //definisanje dugmeta RC7 i provera da li je pritisnuto
        staroStanje7 = 1;                  //postavljanje staroStanje7 na 1
        }
      if(staroStanje7 && Button(&PORTC, 7, 1, 0)){   //ako je RC7 otpusteno
        flag7 =1;                                    //postavlja se flag7 na 1
        staroStanje7 = 0;                            //i vraca se staroStanje7 na 0
        }
      if(Button(&PORTC, 6, 1, 1)){                   //definisanje dugmeta RC6 i provera da li je pritisnuto
        staroStanje6 = 1;                            //postavljanje staroStanje6 na 1
        }
      if(staroStanje6 && Button(&PORTC, 6, 1, 0)){   //ako je RC6 otpusteno
        flag6 =1;                                    //setujemo flag6 na 1
        staroStanje6 = 0;                            //i vraca se staroStanje6 na 0
       }
      }
```

Sada proveravamo da li je korisnik pritisnuo dugme za pauzu (RC5). Ukoliko jeste, neophodno je da se trenutno stanje mašine smesti u promenljivu zmijaSmer. Ukoliko korisnik poželi da nastavi sa igrom poslednje stanje pre pauze se uzima iz promenljive zmijaSmer i smešta u promenljivu stanje.

```
if(Button(&PORTC, 5, 1, 1)){              //definisanje dugmeta RC5 i provera da li je stisnuto
      staroStanje5 = 1;                   //postavlja se staroStanje5 na 1
      }
     if(staroStanje5 && Button(&PORTC, 5, 1, 0)){   //ako je RC5 otpusteno
      staroStanje5 = 0;                             //vraca se staroStanje5 na 0
      if(stanje == PAUZA){                          //ako je stanje PAUZA
       stanje = zmijaSmer;                          //ucitaj prethodno stanje zmije u kom je masina stanja bila pre pauze
```

```
        flagPauza = 0;                          //resetuj fleg flagPauza na 0
        }
       else{                                    //ukoliko igra nije bila pauzirana
        zmijaSmer = stanje;                     //u promenljivu zmijaSmer cuva se stanje masine stanja pre pauze
        stanje = PAUZA;                         //i prelazi se u stanje PAUZA
        flagPauza = 1;                          //setujemo flagPauza na 1
        }
       }
```

Poslednja provera je provera da li je korisnik pritisnuo dugme za reset igre (RC4):

```
if(Button(&PORTC, 4, 1, 1)){            //definisanje dugmeta RC4 i provera da li je pritisnuto
       staroStanje4 = 1;                //postavlja se staroStanje4 na 1
       }
      if(staroStanje4 && Button(&PORTC, 4, 1, 0)){    //ako je RC4 otpusteno
       staroStanje4 = 0;                              //postavlja se staroStanje4 na 0
       stanje = POCETAK;                              //igra se restartuje i prelazi se u stanje POCETAK
       }
```

Sledeći deo koda ima zadatak da na svaki generisani prekid od strane tajmera TMR0 vrši proveru sudara zmije sa preprekama ili proveru sudara zmije same sa sobom. Takođe, ovde se definiše iscrtavanje kretanja zmije.

```
if(flagPrekid){                         //ako je generisan prekid
       if(flagPauza == 0){              //ako je stanje PAUZA ne proverava se sudar
        Glcd_Dot(zmijaX[19], zmijaY[19], 0);    //brisanje repa zmije


        if(flagSudar){                  //ako je prethodno nije doslo do sudara (inverzna logika)
          //provera sudara sa preprekom 1
          if((zmijaX[0] >= prepreka1_X) && (zmijaX[0] <= prepreka1_X+prepreka1_stranica)){   //ako se pozicija glave zmije nalazi između X koordinata projekcije temena prepreke 1 na X osu
            if((zmijaY[0] >= prepreka1_Y) && (zmijaY[0] <= prepreka1_Y+prepreka1_stranica)){ //ako se pozicija glave zmije nalazi izmedju Y koordinata projekcije temena prepreke 1 na Y osu
              stanje = KRAJ;   //prelazi se u stanje KRAJ
              flagSudar = 0;   //flag sudar se resetuje (indikacija da je doslo do sudara, inverzna logika)
```

```
        }
       }
      //provera sudara sa kvadratom 2
      if((zmijaX[0] >= prepreka2_X) && (zmijaX[0] <= prepreka2_X+prepreka2_stranica)){   //ako se pozicija glave zmije nalazi izmedju X koordinata projekcije temena prepreke 2 na X osu
        if((zmijaY[0] >= prepreka2_Y) && (zmijaY[0] <= prepreka2_Y+prepreka2_stranica)){ //ako se pozicija glave zmije nalazi izmedju X koordinata projekcije temena prepreke 2 na Y osu
          stanje = KRAJ;   //prelazi se u stanje KRAJ
          flagSudar = 0;   //flag sudar se resetuje (indikacija da je doslo do sudara, inverzna logika)
          }
        }
      //provera sudara zmije sa samom sobom
      for ( i = 19; i > 0; i-- ){
        if(zmijaX[0]==zmijaX[i] && zmijaY[0]==zmijaY[i] && (flagZmija)){ //ako je X koordinata glave zmije jednaka X koordinati bilo kog dela zmije
          stanje = KRAJ;       //prelazi se u stanje KRAJ
          flagSudar = 0;       //flag sudar se resetuje (indikacija da je doslo do sudara, inverzna logika)
          }
        zmijaX[i]=zmijaX[i-1];  //efekat pomeranja zmije po X osi,trenutna i prethodna koordinata glave zmije
        zmijaY[i]=zmijaY[i-1];  //efekat pomeranja zmije po Y osi,trenutna i prethodna koordinata glave zmije
      } } }
```

Princip detekcije sudara zmije sa preprekom je sledeći: koordinata X prepreke 1 kao i X koordinata poslednje tačke prepreke 1 (X + duzinaPrepreke1) projektuju se na X osu. U svakom trenutku se i X koordinata glave zmije projektuje na X osu i upoređuje da li se nalazi izmedju X koordinata krajnjih tačaka prepreke 1. Ukoliko se nalazi, upoređuje se da li se Y pozicija glave zmije nalazi između Y koordinate i krajnje Y koordinate prepreke 1. Ako je to slučaj, to znači da se zmija sudarila sa preprekom, a to povlači sa sobom promenu stanja iz trenutnog u stanje KRAJ.

Na kraju, definišemo samu mašinu stanja.

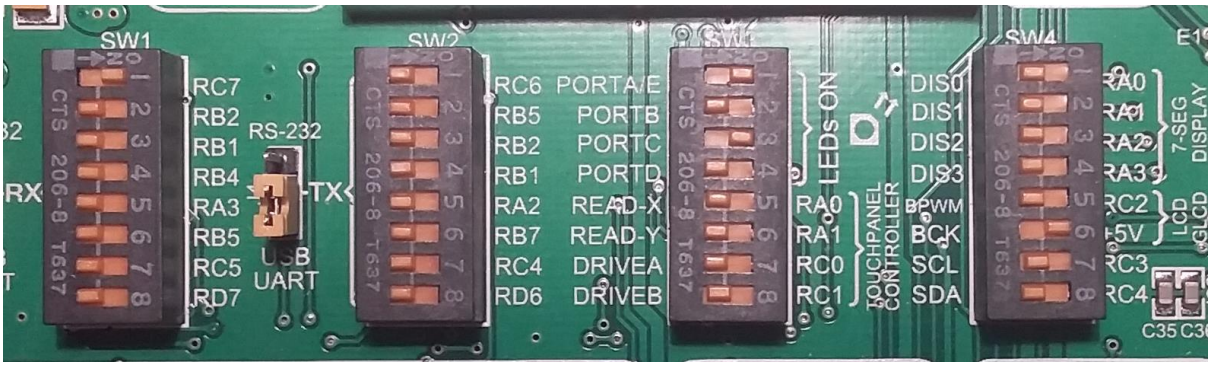
```
//masina stanja igre
    switch(stanje){
      case POCETAK:{              //stanje POCETAK od kojeg pocinje igra
```

```
 inicijalizacija();              //inicijalizacija parametara i zmije
 stanje = DESNO;                 //prvo stanje na pocetku igre
 flagSudar = 1;                  //resetovanje flega za proveru sudara
 }break;
case GORE:{                      //stanje GORE
 zmijaY[0]--;                    //smanajuje se Y kordinata glave zmije kako bi se kretala na gore
 if(zmijaY[0]<0){                //ako je zmija izasla iz okvira ekrana
  zmijaY[0] = 63;                //premesti je da izadje iz donje ivice ekrana
  }
 if(flag7){                      //ako je pritisnuto RC7 (komanda levo)
  stanje = LEVO;                 //novo stanje je LEVO
  flag7 = 0;                     //restujemo indikaciju da je komanda levo zadata
  }
 if(flag6){                      //ako je pritisnuto RC6 (komanda desno)
  stanje = DESNO;                //novo stanje je desno
  flag6 = 0;                     //resetujemo indikaciju da je komanda levo zadata
  }
 }break;
case DOLE:{                      //stanje DOLE
 zmijaY[0]++;                    //povecava se Y kordinata glave zmije
 if(zmijaY[0]>63){
  zmijaY[0] = 0;
  }
 if(flag7){
  stanje = DESNO;
  flag7 = 0;
  }
 if(flag6){
  stanje = LEVO;
  flag6 = 0;
  }
```

```
 }break;
case DESNO:{
 zmijaX[0]++;                //povecava se X kordinata glave zmije
 if(zmijaX[0]>127){
  zmijaX[0] = 0;
  }
 if(flag7){
  stanje = GORE;
  flag7 = 0;
  }
 if(flag6){
  stanje = DOLE;
  flag6 = 0;
  }
 flagZmija = 1;              //fleg se postavlja na 1 kada se glava prvi put poveca
 }break;
case LEVO:{
 zmijaX[0]--;               //smanjuje se X kordinata glave zmije
 if(zmijaX[0]<0){
  zmijaX[0] = 127;
  }
 if(flag7){
  stanje = DOLE;
  flag7 = 0;
  }
 if(flag6){
  stanje = GORE;
  flag6 = 0;
  }
 }break;
case PAUZA:{
```

```
        break;
        }
      case KRAJ:{
        if(flagKraj){              //proverava se da li je poruka ispisana na ekranu
          krajIgre();              //funkcija za isrctavanje slike na kraju igre i ispis poruke za reset igre
        }
        }break;
      }
    flagPrekid = 0;              //Resetovanje flega
     if(flagPauza == 0 && flagKraj){   //ako stanje nije PAUZA i ako nije ispisana poruka na ekranu, crta se zmija
        for (k=0; k<20; k++){
          Glcd_Dot(zmijaX[k], zmijaY[k], 1);
        } } } } }
```

1.4 Fotografije konfiguracije razvojne ploče



Slika 5. Konfiguracija mikroprekidača

Slika 6. Položaj kratkospojnika zvučnika