Univerza *v Ljubljani*
Fakulteta *za računalništvo*
in informatiko

Igra spomina

Vhodno izhodne naprave

Gašper Rifel, 63210280

Ljubljana, 2024

Kazalo vsebine

Uvod

Za projekt pri predmetu Vhodno izhodne naprave sem se odločil, da bom izdelal igro spomina. Hotel sem, da je igra vizualno in zvočno interaktivna, saj tako lahko uporabnik lažje in bolj razločno prepozna vzorec.

Na začetku se izvaja animacija, kjer v matriki 3x3 ledic prižigajo ledice v krožnem gibanju. S tem uporabniku sporoča, da igra čaka na pritisk kateregakoli gumba in s tem začetek igre. Ko uporabnik pritisne katerekoli gumb, se ledice začnejo posamično prižigati v naključnem vzorcu, ob tem se za vsako prižgano ledico oglasi piskač, ki s tonom pove, katera ledica se je prižgala. Če se prižge ledica zgoraj levo, je ton nizek, ko pa se premikamo navzdol od leve proti desni so toni čedalje višji.

Ko se zaporedje konča, je naloga uporabnika, da s pritiskanjem gumbov vnese prikazan vzorec. Če je ta vzorec napačen, vse ledice 5-krat utripnejo, piskač pa zaigra 5 tonov, kjer je vsak ton nižji, kar ponazarja, da je uporabnik izgubil igro. Nato pa se zopet prikaže animacija kroženja ledic, saj se je igra ponovno zagnala in spet čaka da bo igro uporabnik začel.

Če je vzorec, ki ga vnese uporabnik pravilen, potem za pol sekunde zasvetijo vse ledice, nato pa se začne prikazovati naslednje naključno zaporedje, ki je od prejšnjega večje za ena. Ko uporabnik konča igro, kjer je zaporedje 5ih utripov ledic, se prikaže novo zaporedje, kjer so v zaporedju zopet trije utripi, s tem da se ledice prižigajo hitreje in so prižgane manj časa. To se v igri zgodi 2x, ko pa uporabnik nazadnje pravilno vnese zaporedje, začenjo vse ledice utripati ter piskač zaigra tone, ki ponazarjajo zmagovalno glasbo.

GitHub: https://github.com/RiGasper/VIN

Video: https://youtu.be/UcAOf7QjIis

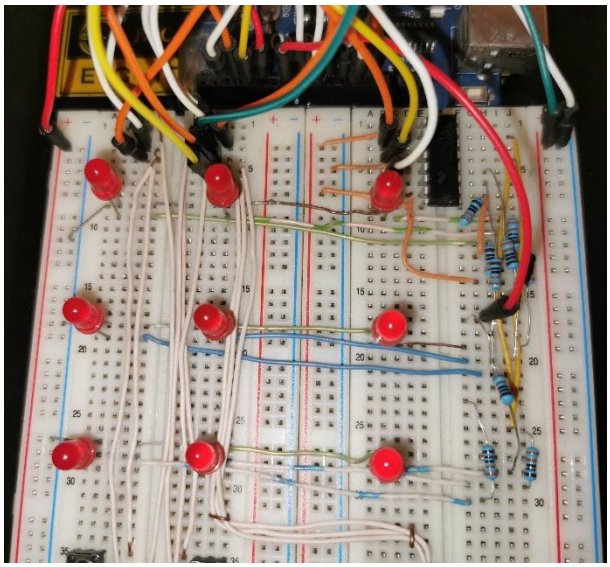
Sestava

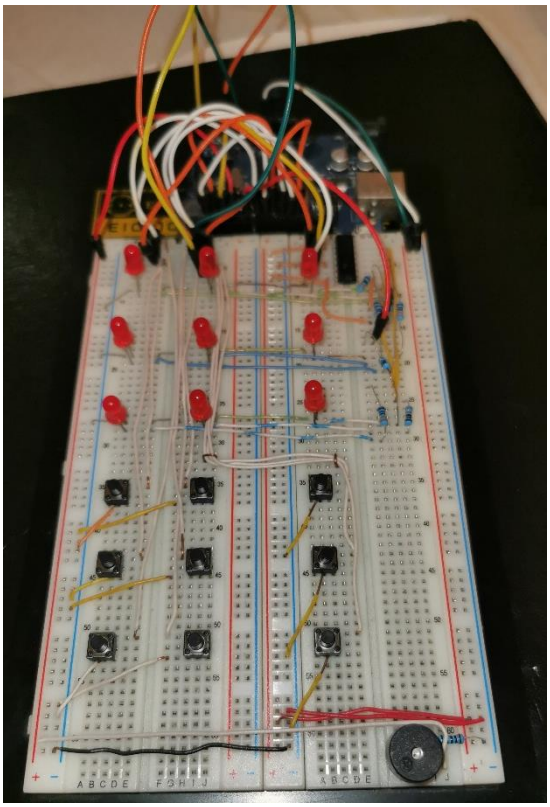
Uporabljeni elementi:

- 9x enobarvnih ledic
- 9x gumobv
- 1x piskač
- SN74HC595N čip (multiplekserski čip, ki 3 izhode pretvori v 8)
- 9x 330 ohm uporov
- 1x x 100 ohm upor

Sestavljenost

- 9 LED diod je postavljenih v 3x3 matriki, kjer so vse povezane na ozemljitev.
- SN74HC595N čip je povezan na plus, minus, ter na 3 izhode arduino plošče
- Prvih 8 LED diod dobi plus iz uporabljenega multiplekserskega čipa, 9-ta LED dioda pa je povezana na svoj izhod na arduino plošči (saj ima čip le 8 izhodov). Vsaka od teh uporablja svoj 330 ohmski upor.
- 9 gumbov je postavljenih v 3x3 matriki. Vsak je povezan na ozemljitev, ter vsak je povezan na svoj izhod na arduino plošči.
- Vsi uporabljeni izhodi/vhodi so digitalni, le trije gumbi uporabljajo analogni vhod na arduino plošči kot digitalni.
- Piskač je povezan na ozemljitev ter na svoj izhod na arduino plošči, z 100 ohmskim uporom.





Programska oprema

Najprej sem inicializiral vse potrebne spremenljivke.

```
#define b3 A0
#define b6 A2
#define b9 A5
int stcp = 4;
int shcp = 3;
int ds = 2;

int BUZZ = 5;

int b1 = 8;
int b2 = 9;
int b4 = 10;
int b5 = 11;
int b7 = 12;
int b8 = 13;
int devet = 7;

int zamik = 600;
int ponavljaj = 3;
int sequence[10];
```

b3,b6,b9 – gumbi, ki uporabljajo analogni vhode

Stcp, **shcp**, **ds** –

BUZZ – piskač, ki je povezan na digitalnem izhodu 5

b1,**b2,b4,b5,b7,b8** – gumbi, ki uporabljajo digitalne vhode

Devet – Deveta LED dioda, ki uporablja svoj digitalni izhod na Arduino plošči, namesto izhoda na multiplekserju.

Zamik – Število milisekund, ki se uporablja za zamik (delay) med prižiganjem in ugašanjem diod.

Ponavljaj – Število, ki pove dolžino zaporedja utripov ledic

Sequence – Array, kamor se shranjuje niz naključnega zaporedja.

Pomožne funkcije

Posveti

Pomožna funkcija »**posveti(int, bool)«** na podlagi podane vrednosti kliče pomožno funkcijo **posveti()** z pravilnimi parametri. Ta sprejme integer **number**, ki ponazarja številko LED diode, ki se bo prižgala, ter bool **buzz**, ki določi ali se bo piskač oglasil ali ne. Za vsako število LED diode, se kliče funkcija »**push()«**, ki sprejme višino tona v hertzih, binarno število, ki pove, katere LED diode se morajo prižgati, ter bool, ki pove, ali se mora piskač oglasiti.

```
166  void posveti(int number, bool buzz) {
167      switch (number) {
168          case 1:
169              push(500, 0B00000001, buzz);
170              break;
171          case 2:
172              push(600, 0B00000010, buzz);
173              break;
174          case 3:
175              push(700, 0B00000100, buzz);
176              break;
177          case 4:
178              push(800, 0B00001000, buzz);
179              break;
180          case 5:
181              push(900, 0B00010000, buzz);
182              break;
183          case 6:
184              push(1000, 0B00100000, buzz);
185              break;
186          case 7:
187              push(1100, 0B01000000, buzz);
188              break;
189          case 8:
190              push(1200, 0B10000000, buzz);
191              break;
192          case 9:
193              digitalWrite(stcp, LOW);
194              shiftOut(ds, shcp, MSBFIRST, 0B00000000);
195              digitalWrite(stcp, HIGH);
196              digitalWrite(devet, HIGH);
197              if (buzz) {
198                  tone(BUZZ, 1300);
199              }
200              delay(zamik);
201              digitalWrite(devet, LOW);
202              if (buzz) {
203                  noTone(BUZZ);
204              }
205              break;
206          default:
207              push(200, 0B11111111, buzz);
208              break;
209      }
210  }
```

Push

Pomožna funkcija »**push(int, int, bool)«** prižge izbrane LED diode in piskač. Ta sprejme višino tona v hertzih **buzz**, binarno število **byte**, ki pove, katere LED diode se morajo prižgati, ter bool **sound**, ki pove, ali se mora piskač oglasiti.

Funkcija najprej prižge LED diode, ki se morajo prižgati na podlagi binarne spremenljivke **byte**. Če je spremenljivka **sound =** true, poleg prižganih diod zazveni tudi piskač. Nato program počaka za toliko milisekund, kolikor je trenutno vrednost zamik. Po tem pa se izbrane LED diode ugasnejo, ter piskač utihne.

```
212  void push(int buzz, int byte, bool sound) {
213      digitalWrite(stcp, LOW);
214      shiftOut(ds, shcp, MSBFIRST, byte);
215      digitalWrite(stcp, HIGH);
216
217      if (sound) {
218          tone(BUZZ, buzz);
219      }
220      delay(zamik);
221      if (sound) {
222          noTone(BUZZ);
223      }
224
225      digitalWrite(stcp, LOW);
226      shiftOut(ds, shcp, MSBFIRST, 0B00000000);
227      digitalWrite(stcp, HIGH);
228  }
```

Flash

Pomožna funkcija **flash(int, int)** omogoča prižig in utrip vseh LED diode. Ta sprejme število **del**, ki pove koliko milisekund je med utripi vseh LED diod, ter drugo število **ponovi**, ki pove kolikokrat se bo utrip zgodil.

```
230  void flash(int del, int ponovi) {
231      for (int i = 0; i < ponovi; i++) {
232          digitalWrite(stcp, LOW);
233          shiftOut(ds, shcp, MSBFIRST, 0B11111111);
234          digitalWrite(stcp, HIGH);
235          digitalWrite(devet, HIGH);
236
237          delay(del);
238
239          digitalWrite(devet, LOW);
240          digitalWrite(stcp, LOW);
241          shiftOut(ds, shcp, MSBFIRST, 0B00000000);
242          digitalWrite(stcp, HIGH);
243
244          delay(del);
245      }
246  }
```

Izvajanje programa

Setup

Najprej sem nastavil vse vhode in izhode na Arduino plošči, ter zagnal začetno animacijo s klicem funkcija snakeAnimation().

```
22  void setup() {
23      Serial.begin(9600);
24      pinMode(stcp, OUTPUT);
25      pinMode(shcp, OUTPUT);
26      pinMode(ds, OUTPUT);
27      pinMode(BUZZ, OUTPUT);
28      pinMode(devet, OUTPUT);
29      pinMode(b1, INPUT_PULLUP);
30      pinMode(b2, INPUT_PULLUP);
31      pinMode(b3, INPUT_PULLUP);
32      pinMode(b4, INPUT_PULLUP);
33      pinMode(b5, INPUT_PULLUP);
34      pinMode(b6, INPUT_PULLUP);
35      pinMode(b7, INPUT_PULLUP);
36      pinMode(b8, INPUT_PULLUP);
37      pinMode(b9, INPUT_PULLUP);
38
39      snakeAnimation();
40  }
```

SnakeAnimation

Začetno animacijo upravlja Funkcija »snakeAnimation()«. V nizu so definirane ledice, ki se bodo ena za drugo prižigale, ter zamik je začasno nastavljen na 50ms, da bo animacija hitrejša in bolj gladka. Nato se v neskončni zanki iterira skozi vse elemente v nizu, ter se za vsak element pokliče funkcija »posveti(zaporedje[i], false)«. Tu se v funkcijo pošlje i-ti element zaporedja, ter Bool false, ki funkciji sporoča, naj se piskač ne oglasi. Ko uporabnik izbere katerikoli gumb, se sproži IF, ki **zamik** nastavi na začetno vrednost, ter se vrne iz funkcije nazaj v **Setup().**

```
248 void snakeAnimation() {
249
250   // Definicija zaporedja LED diod, ki se bodo posamično prižigale.
251     int zaporedje[] = {1, 2, 3, 6, 9, 8, 7, 4};
252     zamik = 50;
253
254     // Iteriraj skozi zaporedje, dokler uporabnik ne izbere nekega gumba.
255     while (true) {
256         for (int i = 0; i < 8; i++) {
257             posveti(zaporedje[i], false);
258
259             if (digitalRead(b1) == LOW || digitalRead(b2) == LOW || digitalRead(b3) == LOW ||
260                 digitalRead(b4) == LOW || digitalRead(b5) == LOW || digitalRead(b6) == LOW ||
261                 digitalRead(b7) == LOW || digitalRead(b8) == LOW || digitalRead(b9) == LOW)
262                 {
263                   // Nastavi zamik na začetno vrednost in se vrni v Setup().
264                   zamik = 600;
265                   delay(zamik);
266                   return;
267             }
268
269             delay(25);
270         }
271     }
272 }
```

Loop

Ko se konča funkcija **snakeAnimation(),** se ta vrne v **setup()** funkcijo, ki kliče funkcijo **loop().** To je osnovna funkcija Arduino programov, v kateri se izvaja neskončna zanka, dokler se program ne prekine.

Znotraj sta 2 FOR zanki. Prva določa kolikokrat se bo generiralo zaporedje, druga pa določa dolžino zaporedja, ki se skozi potek igre spreminja.

Najprej se generira zaporedje, ki je dolgo toliko kot je vrednost spremenljivke **ponavljaj.** Vsako iteracijo se izbere naključno število od 1 do 9, ter se jo zapiše na k-to mesto v niz **sequence**. Nato se kliče funkcija **posveti(),** ki prižge LED diodo, na katero se nanaša naključno število. Med vsako iteracijo FOR zanke, se program ustavi za število milisekund, kolikor je vrednost spremenljivke **zamik**, razen če je trenutna iteracija zadnja. Če je to res, se program ne bo ustavil in uporabnik bo lahko takoj začel vnašati zaporedje z uporabo gumbov.

Ko se prikaže zaporedje, se kliče funkcija **gumb()**, ki čaka, da uporabnik pritisne potrebne gumbe in sproti preverja, če ti gumbi sledijo prikazanemu zaporedju. Če uporabnik vnese napačen gumb, funkcija vrne vrednost 1, drugače pa vrne 0.

Če je vrnjena vrednost 1, se vse vrednosti programa ponastavijo in igra se začne znova. Drugače pa se spremenljivka **ponavljaj** poveča za eno, kar pomeni, da bo naslednje zaporedje daljše za ena.

```
42  void loop() {
43
44      // Ob spremembi hitrosti prižiga LED diod, nastavi dolžino zaporedja na 3
45      ponavljaj = 3;
46      for (int i = 0; i < 3; i++) {
47          for (int k = 0; k < ponavljaj; k++) {
48
49              // Generiraj naključno število in jo shrani v array, na k-to mesto.
50              int randomButton = random(1, 10);
51              sequence[k] = randomButton;
52              posveti(randomButton, true);
53
54              // Brez zamika v zadnji ponovitvi zanke
55              // (uporabnik lahko takoj po končanem zaporedju začne vnašati zaporedje)
56              if (k != ponavljaj - 1) {
57                  delay(zamik);
58              }
59          }
60
61          // Funkcija, ki čaka na vnos zaporedja uporabnika (vrne 1 če se uporabnik zmoti, drugače 0)
62          int checkIfWrong = gumb(ponavljaj);
63
64          // Če se uporabik zmoti, ponastavi vrednosti in ponovno zaženi igro
65          if (checkIfWrong == 1) {
66              i = -1;
67              ponavljaj = 3;
68              zamik = 600;
69              sequence[10];
70              setup();
71          }
72          // Če uporabnik uspešno vnese zaporedje, povečaj število utripov zaporedja
73              ponavljaj = ponavljaj + 1;
74      }
```

Ko se konča prva FOR zanka, pomeni da se je zaporedje z določenim zamikom zgeneriralo že 3-krat. Sedaj se **zamik** zmanjša za 200 milisekund in funkcija **loop()** se ponovi. V primeru, da je **zamik** enak 0, je uporabnik zmagal igro. Prižgejo se vse LED diode, ter piskač zaigra zaporedje tonov, ki ponazarjajo zmago. Vrednosti programa se ponastavijo in igra se začne znova.

```
75      // Ko se zgodi level-up, zmanjšaj zakasnitev med prižiganjem/ugašanjem ledic
76      zamik = zamik - 200;
77
78      // Ko je zamik 0, je konec igre
79      if (zamik == 0) {
80
81          digitalWrite(stcp, LOW);
82          shiftOut(ds, shcp, MSBFIRST, 0B11111111);
83          digitalWrite(stcp, HIGH);
84          digitalWrite(devet, HIGH);
85
86          playWinningMelody();
87
88          digitalWrite(devet, LOW);
89          digitalWrite(stcp, LOW);
90          shiftOut(ds, shcp, MSBFIRST, 0B00000000);
91          digitalWrite(stcp, HIGH);
92
93          ponavljaj = 3;
94          setup();
95      }
96  }
```

Gumb

Funkcija **gumb(int)** sprejme število, ki pove kolikšna je dolžina zaporedja. Nato se FOR zanka izvede tolikokrat, kolikor je dolžina zaporedja in za vsako iteracijo čaka na pritisk kateregakoli gumba. Ko je nek gumb pritisnjen, si ga zapomni v spremenljivko **pressedButton.**

```
98  int gumb(int in) {
99      // Zanka, ki se ponovi za vsak element zaporedja
100     for (int i = 0; i < in; i++) {
101
102         // Čakaj na pritisk gumba
103         while (digitalRead(b1) == HIGH && digitalRead(b2) == HIGH && digitalRead(b3) == HIGH &&
104                digitalRead(b4) == HIGH && digitalRead(b5) == HIGH && digitalRead(b6) == HIGH &&
105                digitalRead(b7) == HIGH && digitalRead(b8) == HIGH && digitalRead(b9) == HIGH) {}
106
107         int pressedButton = 0;
108
109         // Zapomni si gumb, ki ga je pritisnil uporabnik
110         if (digitalRead(b1) == LOW) pressedButton = 1;
111         else if (digitalRead(b2) == LOW) pressedButton = 2;
112         else if (digitalRead(b3) == LOW) pressedButton = 3;
113         else if (digitalRead(b4) == LOW) pressedButton = 4;
114         else if (digitalRead(b5) == LOW) pressedButton = 5;
115         else if (digitalRead(b6) == LOW) pressedButton = 6;
116         else if (digitalRead(b7) == LOW) pressedButton = 7;
117         else if (digitalRead(b8) == LOW) pressedButton = 8;
118         else if (digitalRead(b9) == LOW) pressedButton = 9;
```

Nato za vsako iteracijo IF stavek preveri, če je vnesen gumb pravilen, tako da preveri če je vrednost gumba enaka i-temu (števec iz FOR zanke) elementu v nizu **sequence**. Če je gumb pravilen, se prižge LED dioda, ki ta gumb ponazarja, ter FOR zanka se nadaljuje (tu se zamik nastavi na manjšo vrednost, saj sem hotel da so gumbi zelo odzivni in se LED dioda prižge le za trenutek).

Če pritisnjen gumb ni pravi, vse LED diode 5-krat utripnejo ter piskač zaigra zaporedje tonov, ki ponazarjajo, da je uporabnik izgubil. Funkcija nato vrne vrednost 1, če pa se FOR zanka brez napak izvede do konca, se bodo za trenutek prižgale vse LED diode in funkcija bo vrnila vrednost 0.

```
120         // Preveri, če je pritisnjen gumb pravilen
121         if (pressedButton != sequence[i]) {
122 
123             // Zaigraj tone piskača, ki ponazarjajo izgubo
124             // Utripaj LED diode med igranjem piskača
125             for (int j = 0; j < 5; j++) {
126                 digitalWrite(stcp, LOW);
127                 shiftOut(ds, shcp, MSBFIRST, 0B11111111);
128                 digitalWrite(stcp, HIGH);
129                 digitalWrite(devet, HIGH);
130 
131                 if (j == 0) tone(BUZZ, 1000);
132                 else if (j == 1) tone(BUZZ, 800);
133                 else if (j == 2) tone(BUZZ, 600);
134                 else if (j == 3) tone(BUZZ, 400);
135                 else if (j == 4) tone(BUZZ, 200);
136 
137                 delay(300);
138 
139                 digitalWrite(stcp, LOW);
140                 shiftOut(ds, shcp, MSBFIRST, 0B00000000);
141                 digitalWrite(stcp, HIGH);
142                 digitalWrite(devet, LOW);
143 
144                 delay(200);
145             }
146 
147             noTone(BUZZ);
148             delay(1000);
149 
150             return 1;
151         }
152 
153         // Gumb je pravilen, posveti LED diodo, katero predstavlja izbran gumb.
154         int temp = zamik;
155         zamik = 450;
156         posveti(pressedButton, true);
157         zamik = temp;
158     }
159     // Posveti vse LED diode, kar ponazarja, da je vneseno zaporedje pravilno
160     // in da prihaja novo zaporedje.
161     flash(1000, 1);
162     delay(1000);
163     return 0;
164 }
```

Zaključek

Projekt mi je bil zelo zanimiv, saj sem prvič izdelal tudi hardware, kjer se je izvajala moja koda. Do sedaj so bili vsi moji projekti interaktivni le preko spletnih aplikacij, ta projekt pa je bil fizično interaktiven in veliko bolj zanimiv. Uporaba multiplekserja mi je odprla nov svet, kjer nisem omejen z številom vhodov in izhodov. Sestava devetih gumbov in LED diod ponuja številne možnosti za nadaljnjo nadgradnjo, saj se na tej osnovi lahko implementira veliko iger. Zelo sem zadovoljen s končnim izdelkom in ga bom definitivno še nadgradil.