Univerzitet Singidunum

Tehnički fakultet

Predmet:

Sistemi za rad u realnom vremenu

Profesor:
Prof. dr Marko Tanasković
Predmetni asistent:
Uroš Dragović

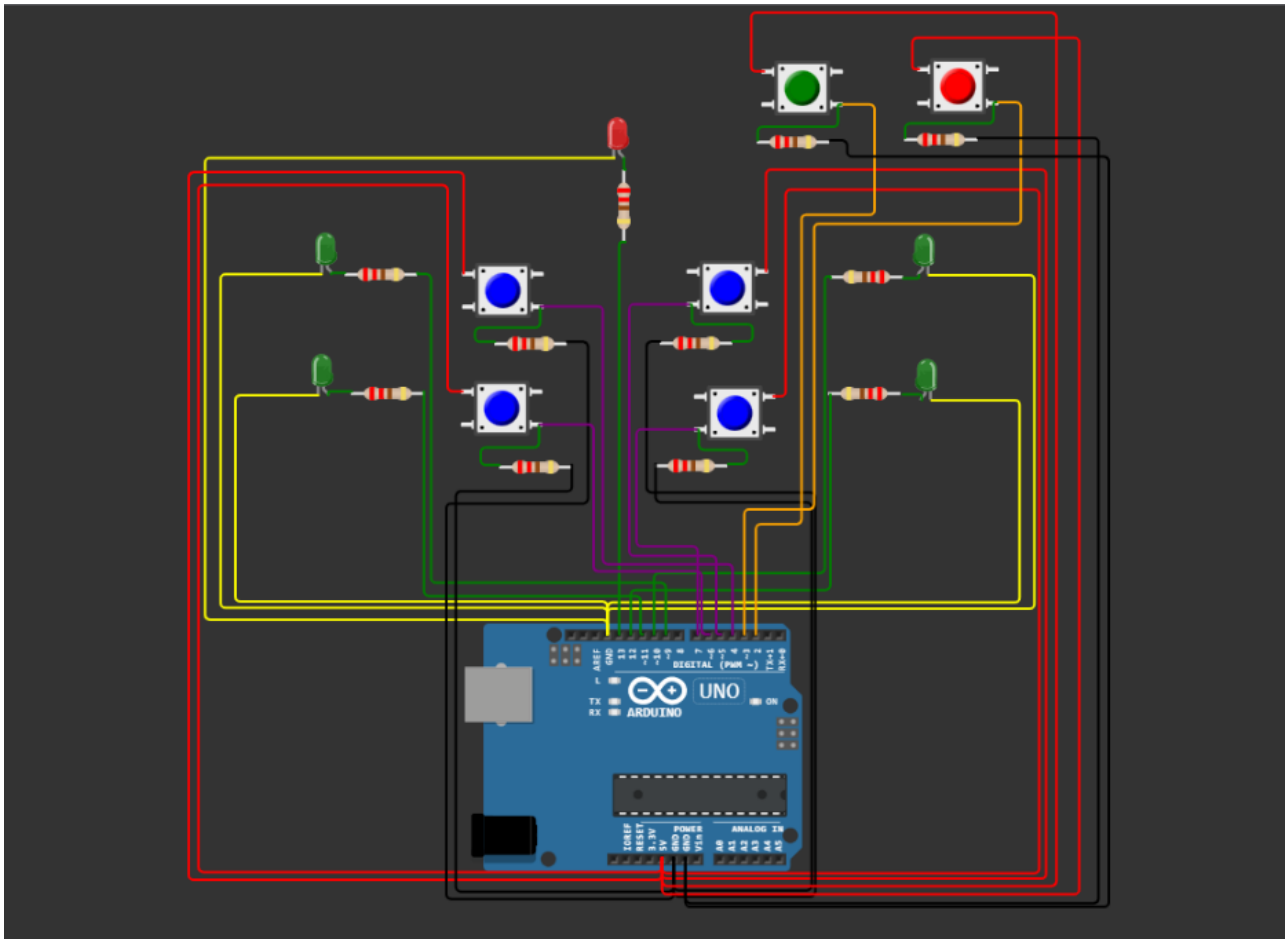
Student:
Božidar Mladenović 2018240254

Beograd, 2022. Godina

1. Uvod

U ovoj dokumentaciji biće prikazana implementacija i šematski prikaz ispitnog zadatka „Igra ponavljanja sekvenci“ iz predmeta „Sistemi za rad u realnom vremenu“. Projekat je realizovan uz pomoć arduino simulatora „wokwi“.

2. Šematski Prikaz



(Slika 1. Šematski prikaz zadatka)

Kao što je prikazano na slici 1, u šemi zadatka se nalaze četiri zelene diode koje služe za prikazivanje sekvence za pogađanje, jedna crvena dioda koja uz pomoć vremenskog prekida treperi kada korisnik napravi grešku i svetli ako korisnik izgubi igru, zatim četiri tastera koji služe za pogađanje za svaku zelenu diodu i dva tastera koji služe za pokretanje i prekidanje same igre i ta dva tastera su eksterni prekidi.

- Diode su žutom žicom povezane na uzemljenje, a zelenom žicom na otpornik čija je otpornost 220 oma i na digitalne pinove od 9 do 13
- Tasteri za pogađanje sekvence su povezani crvenom žicom na napon od 5V, ljubičastom žicom na digitalne pinove od 4 do 7 i crnom žicom na otpornike (takođe 220 oma) i uzemljene
- Tasteri za pokretanje i prekidanje igre su takođe crvenom žicom povezani na napon od 5V, narandžastom žicom na pinove 2 i 3 jer jedino ta dva pina mogu da čitaju eksterni prekid i na kraju crnom žicom na otpornik (220 oma) i uzemljenje.

3. Implementacija

U ovom delu su prikazani i detaljno objašnjeni delovi koda zadatka.

```
1   //Ubacivanje TimerOne biblioteke u projekat
2   #include <TimerOne.h>
3
4   //Nizovi u koje čuvamo pinove led dioda i pinove tastera
5   const int leds[] = {9, 10, 11, 12};
6   const int buttons[] = {4, 5, 6, 7};
7
8   //Definišemo konstantu koja će biti korišćena za pin 13 (dioda za greške)
9   #define LED_WRONG 13
10
11  //Definišemo konstantu koja će biti korišćena za pin 2 (dioda za startovanje sekvenci)
12  #define BUTTON_START 2
13  //Definišemo konstantu koja će biti korišćena za pin 3 (dioda za prekid igre)
14  #define BUTTON_END 3
15
16  //Definisanje konstanti za stanja radi čitljivijeg koda
17  #define START_SEQUENCE    0
18  #define USER_GUESSING     1
19  #define PROGRAM_CHECKING  2
20  #define SHOW_NEW_SEQUENCE 3
21  #define GAME_OVER         4
22
23  //Definisanje state varijablje
24  int state;
25
26  //Definisanje nizova za sekvencu igre i za sekvencu korisnika koje se kasnije upoređuju
27  int game_sequence[100] = {0};
28  int user_sequence[100] = {0};
29
30  //Definisanje početnog nivoa, da u prvoj sekvenci zasvetle 2 diode
31  int sequence_length = 2;
32
33  //Definisanje varijable za greške korisnika
34  int mistakes = 0;
35  //Definisanje varijable za broj mogućih grešaka korisnika
36  int available_mistakes = 3;
37  //Definisanje varijable za osvojene poene korisnika
38  int score = 0;
39  //Postavljamo pocetnu vrednost crvene lampice na LOW odnosno 0 odnosno iskljuceno
40  int red_led = LOW;
41  //Pokazivač na trenutni element igračeve sekvence
42  int seq_pointer = 0;
43
44  //Definisanje i setovanje varijable na false, služi sa pokretanje sekvence klikom na dugme
45  bool start_game = false;
```

(Slika 2. Definisanje Varijabli)

Na slici 2 se vidi definisanje svih promenljivih i svih konstanti koje se koriste u projektu, nizovi koji se koriste za led diode i tastere, nizovi koji se koriste za sekvence programa i korisnika i konstante koje se koriste za stanja mašine stanja.

```
148  void setup() {
149
150    //Omogucava ispis na konzoli
151    Serial.begin(9600);
152
153    //Omogucava arduinu da koristi nasumicno biranje elemenata u nizu
154    randomSeed(analogRead(A0));
155
156    //Za svaki broj u leds i buttons nizu postavlja se mod
157    for (int i = 0; i < 4; i++) {
158      pinMode(leds[i], OUTPUT);
159      pinMode(buttons[i], INPUT);
160    }
161
162    //Postavlja se mod Izlaz za crvenu diodu
163    pinMode(LED_WRONG, OUTPUT);
164    //Inicializuje se tajmer na 0.8 sekundi
165    Timer1.initialize(800000);
166    //Timeru se povezuje prekid koji ce da pozove funkciju za vremenski prekid
167    Timer1.attachInterrupt(timeInterrupt);
168
169    //Postavlja se mod ulaz za dugme koje pokrece igru
170    pinMode(BUTTON_START, INPUT);
171    //Tasteru za pokretanje igre se povezuje prekid koji ce da pozove funkciju za
172    //spoljasnji prekid
173    attachInterrupt(digitalPinToInterrupt(BUTTON_START), button_interrupt_start, RISING);
174
175     //Postavlja se mod ulaz za dugme koje zaustavlja igru
176    pinMode(BUTTON_END, INPUT);
177    attachInterrupt(digitalPinToInterrupt(BUTTON_END), button_interrupt_end, RISING);
178
179    //Na pocetku je stanje inicijalzovano na START_SEQUENCE
180    state = START_SEQUENCE;
181  }
```

(Slika 3. Funkcija Setup)

Funkcija Setup je ugradjena funkcija koja se prva pokreće i koja služi za inicjalizaciju celog projekta, na slici 3 se mogu videti komande kao sto su “pinMode” koje postavljaju modove za pinove podrazumevano u arduinu svi digitalni pinovi imaju INPUT mod koji služi za unos, pored INPUT moda u ovom projektu je korišćen i OUTPUT mod koji služi za čitanje. “Serial.begin” komanda je tu da bi se omogućio ispis na konzoli, zatim “randomSeed” komanda koja omogućava nasumično biranje elemenata u nizu jer bez toga Arduino nema tu mogućnost. Što se tiče komande “attachInterrupt” ona sadrži tri argumenta, prvi je za pin koji želimo da pokreće interrupt (Na ploči za koju je napisan ovaj kod “Arduino UNO” pinovi koji mogu da čitaju prekide su samo pin 2 i pin 3), drugi argument je funkcija prekida i treći je to kada će se aktivirati prekid u ovom slučaju to je RISING što znači da će se pri promeni napona sa LOW na HIGH aktivirati, “digitalPinToInterrupt” ovim pinu koji stavimo kao prvi argument prethodnoj funkciji stavljamo do znanja da će se koristiti kao prekid. Ovo su bili eksterni prekidi, a u ovom projektu je korišćen i vremenski prekid koji je moguć uz pomoć bilbioteke TimeOne, na slici 3 se vidi inicializacija brojača “Timer1” na 0,8 sekundi na koji će se vezati vremenski prekid ponovo funkcijom “attachInterrupt” s tim što funkcija u tom slučaju ima samo jedan argument I to je funkcija prekida koja treba da se izvrši. (U nastavku su prikazane sve tri funkcije prekida, 2 eksternog prekida i jedna vremenskog prekida)

```
47  //Funkcija za eksterni prekid koji služi za startovanje igre
48  void button_interrupt_start(){
49    start_game = true;
50  }
```

(Slika 4. Funkcija prvog eksternog prekida za pokretanje igre)

Funkcija za prvi eksterni prekid prikazana je na slici 4 i služi za pokretanje igre tako što će promenljivu start_game setovati na true.

```
52  //Druga funkcija za eksterni prekid koja sluzi da prekine igru
53  //(menja stanje igre u kraj igre)
54  void button_interrupt_end(){
55    state = GAME_OVER;
56  }
```

(Slika 5. Funkcija drugog eksternog prekida za prekidanje igre)

Funkcija za drugi eksterni prekid prikazana je na slici 5 i služi za prekidanje igre tako što će promeniti stanje programa u kraj igre. (Više o stanjima programa u nastavku)

```
58  //Funkcija koja se poziva pri vremenskom prekidu koji je definisan TimerOne bibliotekom
59  //sluzi za paljenje crvene led diode
60  void timeInterrupt(){
61    //ako ima gresaka onda da ce crvena dioda treperiti
62    if(mistakes > 0){
63      red_led = !red_led;
64    }else{
65      red_led = LOW;
66    }
67  }
```

(Slika 6. Funkcija vremenskog prekida)

Na slici 6 prikazana je funkcija vremenskog prekida i ona radi tako što, ako korisnik napravi grešku u pogađanu sekvence, menja stanje crvene diode sa LOW na HIGH to jest uključuje je, a ako nema grešaka samo postavlja crvenu diodu na LOW to jest isključuje je .

```
183 void loop() {
184   //Masina stanja koja kontrolise stanja programa
185   switch(state){
186     //Pocetno stanje za pokretanje sekvence
187     case START_SEQUENCE:
188 
189       //Ako je uz pomoc spoljasnjeg prekide klikom na dugme start, setovano
190       //start_game na true onda pokrenuti igru
191       if(start_game == true){
192 
193         //Iskljucuje crvenu diodu
194         digitalWrite(LED_WRONG, LOW);
195 
196         //Kada je igra pokrenuta vracamo stanje ove varijable na false
197         start_game = false;
198 
199         //Kreira se nova sekvenca brojeva, u ovom slucaju od dva broja
200         create_new_sequence();
201 
202         //Pokrece se sekvenca
203         start_sequence();
204 
205         //Setuje se stanje za pogadjanje sekvence za korisnika
206         state = USER_GUESSING;
207 
208       }
209       break;
```

(Slika 7. Funkcija loop sa mašinom stanja i prvo stanje programa)

Funkcija loop je takođe ugrađena funkcija koja se pokreće odmah nakon setup funkcije, u ovom programu u loop funkciji nalazi se mašina stanja i na slici 7 prikazano je prvo stanje programa koje je setovano uz pomoć state varijable i konstanti za stanja u setup funkciji, a to je „START_SEQUENCE“ . Kao što se vidi na slici kada program dođe u ovo stanje on čeka sve dok se ne desi eksterni prekid kliknom na start dugme da bi se promenljiva start_game setovala na true, kada se to desi program prvo isključuje crvenu diodu ako je uključena, zatim setuje start_game ponovo na false, nakon toga pokreće funkcije „create_new_sequence“ koja kreira sekvencu i funkciju „start_sequence“ koja pokreće sekvencu i na kraju setuje stanje na „USER_GUESSING“.

```
69  //Funkcija koja sluzi za kreiranje sekvence to jest
70  void create_new_sequence(){
71    for(int i=0; i < sequence_length; i++){
72      //u game_sequence ubacije random indekse dioda
73      game_sequence[i] = random(0, 4);
74    }
75  }
```

(Slika 8. Funkcija za kreiranje sekvence)

Funkcija za kreiranje sekvence radi tako što uzme trenutnu dužinu sekvence (Na početku to je broj 2) i onda u niz „game_sequence“ generiše onliko nasumično odabranih indexa dioda kolika je dužina sekvence.

```
85  //Funkcija koja sluzi za pokretanje gore napravljene sekvence
86  void start_sequence(){
87    delay(500);
88    for (int i = 0; i < sequence_length; i++) {
89      //za svaki diodu iz niza pokrece funkciju blink_led koja sluzi za treptanje diode
90      blink_led(leds[game_sequence[i]]);
91    }
92  }
```

(Slika 9. Funkcija za pokretanje sekvence)

Funkcija za pokretanje sekvence radi tako što za svaku diodu iz niza dioda uz pomoć niza kreirane sekvence poziva funkciju „blink_led“.

```
77  //Funkcija koja sluzi da pokrene treptanje diode koja joj je prosledjena
78  void blink_led(int led) {
79    digitalWrite(led, HIGH);
80    delay(250);
81    digitalWrite(led, LOW);
82    delay(250);
83  }
```

(Slika 10. Funkcija za treptanje diode)

Funkcija za tepranje diode uz pomoć „digitalWrite“ funkcije pokreće treptanje diode i tako korisnik vidi koju sekvencu treba da ponovi.

```
210     //U ovom stanju masina ceka input korisnika
211     case USER_GUESSING:
212
213       //Stanje crvene diode u odnosu na broj gresaka
214       digitalWrite(LED_WRONG, red_led);
215
216       //Pozivanje funkcije za pogadjanje sekvence
217       user_guessing_function();
218
219       //Program ceka sve dok korisnik ne klikne onoliko tastera
220       //kolika je dužina sekvence programa
221       if(seq_pointer == sequence_length){
222         //Kada korisnik zavrsi za unosom prelazi se u stanje provere sekvence
223         state = PROGRAM_CHECKING;
224       }
225
226       break;
```

(Slika 11. Stanje u kome korisnik ponavlja sekvencu)

Drugo stanje u koje ulazi program nakon pokretanja sekvence jeste stanje u kome se čeka odgovor korisnika na pokazanu sekvencu, znači poziva „user_guessing_funkciju“ i čeka dok korisnik ne ponovi celu sekvencu, kada korisnik završi sa unosom program prelazi u sledeće stanje a to je „PROGRAM_CHECKING“.

```
94    //Funkcija koja sluzi
95    void user_guessing_function(){
96      //Ako je korisnik kliknuo taster 1
97      if(digitalRead(buttons[0]) == HIGH){
98        //U njegovu sekvencu upisati broj diode
99        user_sequence[seq_pointer] = 0;
100       //Ovo sluzi da didoa trepne da bi korisnik znao koju je izabrao
101       blink_led(leds[0]);
102       //pointer se povecava za 1
103       seq_pointer++;
104     }
105     if(digitalRead(buttons[1]) == HIGH){
106       user_sequence[seq_pointer] = 1;
107       blink_led(leds[1]);
108       seq_pointer++;
109     }
110     if(digitalRead(buttons[2]) == HIGH){
111       user_sequence[seq_pointer] = 2;
112       blink_led(leds[2]);
113       seq_pointer++;
114     }
115     if(digitalRead(buttons[3]) == HIGH){
116       user_sequence[seq_pointer] = 3;
117       blink_led(leds[3]);
118       seq_pointer++;
119     }
120   }
```

(Slika 12. Funkcija za pogađanje korisnika)

Funkcija za pogađanje korisnika radi tako što se proveravaju tasteri da li su pritisnuti i onda za taster koji je pritisnut u niz „user_sequence“ se upisuje broj diode, zatim dioda zatreperi kako bi korisniku dala povratnu informaciju i „seq_pointer“ služi kao pokazivač gde će se taj input korsnika upisati u niz.

```
227     //Stanje programa u kojem program proverava sekvencu koja je trebalo da se pogodi
228     // i sekvencu koju je uneo korisnik
229     case PROGRAM_CHECKING:
230
231       //Ako je duzina sekvence manja ili jednaka od 100 pokrenuti proveru,
232       //ako je duza zavrsiti igru.
233       if(sequence_length <= 100){
234
235         //Ako su sekvenca koju je napravio program i sekvenca koju je uneo korisnik iste
236         if (array_cmp(game_sequence, user_sequence) == true){
237
238           //Povecati duzinu sekvence za 1, nesto kao predji na sledeci nivo
239           sequence_length++;
240
241           //Povecavanje poena korisnika za 1
242           score++;
243
244           //Setovanje stanja koje sluzi za kreiranje nove sekvence
245           state = SHOW_NEW_SEQUENCE;
246
247         }else{
248
249           //Ukoliko se sekvenca korisnka razlikiuje od sekvence programa,
250           //Povecati broj gresaka za 1
251           mistakes++;
252
253           //Ako korisnik ima manje gresaka od 3 moze da nastavi
254           if(mistakes < available_mistakes){
255
256             Serial.print("Wrong Sequence! Mistake: ");
257             Serial.print(mistakes);
258             Serial.print(" of ");
259             Serial.println(available_mistakes);
260
261             //Ponovo se prebacuje u stanje za generisanje nove sekvence
262             state = SHOW_NEW_SEQUENCE;
263
264           }else{
265
266             Serial.print("Wrong Sequence! Mistake: ");
267             Serial.print(mistakes);
268             Serial.print(" of ");
269             Serial.println(available_mistakes);
270
271             //Ukoliko korisnik ima 3 greske prebacuje se u stanje za kraj igre
272             state = GAME_OVER;
273
274           }
275         }
276       }else{
277         state = GAME_OVER;
```

(Slika 13. Stanje u kome program poredi sekvence)

Treće stanje u koje program ulazi je stanje gde se poredi sekvenca korisnika sa sekvencom programa, ukoliko je dužina sekvence manja ili jedanka od 100 program proverava sekvence uz pomoć funkcije „array_cmp“ i ukoliko se sekvence poklapaju znači da je korisnik pogodio sekvencu programa i onda se dužina sekvence povećava, zatim se korisniku dodaje jedan poen i program prelazi u stanje u kome se kreira nova sekvenca. Ukoliko je korisnik pogrešio povećava se broj grešaka korisnika i ukoliko je broj grešaka manji od tri prikazuje mu se nova sekvenca u suprotnom korisnik gubi igru i program prelazi u stanje kraj igre. Takodje ako je dužina sekvence veća od 100 program takoće prelazi u stanje kraj igre.

```
122   //Funkcija koja se koristi za poredjenje dva niza
123   boolean array_cmp(int *a, int *b){
124
125       for (int n=0;n<sequence_length;n++) if (a[n]!=b[n]) return false;
126
127       return true;
128   }
```

(Slika 14. Funkcija „array_cmp“ koja služi za poređenje)

```
280     //Stanje koje kreira nove sekvence
281     case SHOW_NEW_SEQUENCE:
282
283         create_new_sequence();
284
285         start_sequence();
286
287         //Postavljanje pokazivaca na 0
288         seq_pointer = 0;
289
290         state = USER_GUESSING;
291
292       break;
```

(Slika 15. Stanje koje kreira i pokreće novu sekvencu)

Četvrto stanje koje kreira i pokreće novu sekvencu radi na sličnom pricipu kao i prvo stanje samo što ovde ne postoji uslov za pokretanje igre i ovde se promenjliva „seq_pointer“ ponovo setuje na nulu.

```
293     case GAME_OVER:
294
295       //Ukljucuje crvenu diodu
296       digitalWrite(LED_WRONG, HIGH);
297
298       //Pozivanje funkcije za kraj igre
299       game_over();
300
301       break;
302
303   }
304 }
```

(Slika 16. Stanje za kraj igre)

Na kraju peto stanje koje služi za kraj igre uključuje crvenu diodu i poziva funckiju „game_over“.

```
130   //Funkcija koja se pokrece kada je igra zavsena bilo da je korisnik pogresio 3 puta,
131   //presao igru ili iskoristion dugme za prekid igre
132   void game_over(){
133     Serial.print("Games is Over. Your Score was: ");
134     Serial.println(score);
135
136     //Vracamo varijable na pocetno stanje
137     sequence_length = 2;
138     seq_pointer = 0;
139     mistakes = 0;
140     score = 0;
141     red_led = LOW;
142     start_game = false;
143
144     //Vraca se stanje na start_sequence da bi mogli ponovo da pokrenemo igru
145     state = START_SEQUENCE;
146   }
```

(Slika 17. Funkcija „game_over“)

Kao što se vidi na slici 17 funkcija za kraj igre generiše korisniku poruku o kraju i pokazuje mu osvojene poene, zatim postavlja sve promenjlive na početne vrednosti i setuje stanje programa u „START_SEQUENCE“.

4. Literatura

- Wokwi - Online Arduino and ESP32 Simulator
- wokwi-led Reference | Wokwi Docs
- wokwi-pushbutton Reference | Wokwi Docs
- Materijali sa predavanja i vežbi