Igra memorije sa svijetlećim diodama

Pećar, Vjekoslav

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IGRA MEMORIJE SA SVIJETLEĆIM DIODAMA

Završni rad

Vjekoslav Pećar

Osijek, 2023.

SADRŽAJ

1. UVOD

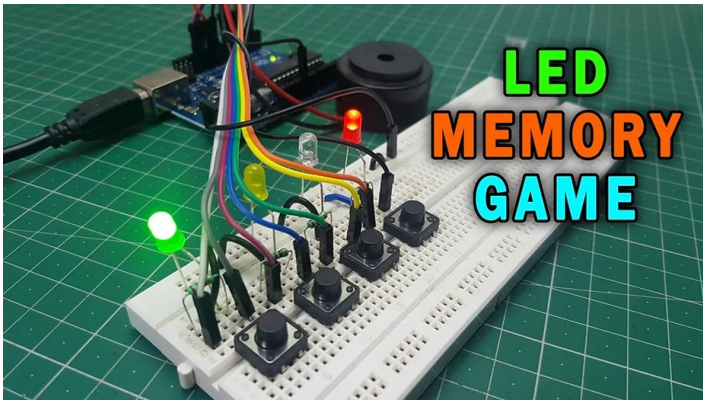
Zadatak za završni rad je izrada igre memorije sa svijetlećim diodama pri čemu će kao platforma biti korišten Arduino mikroupravljački sustav. Igra se zasniva na testiranju vlastitog pamćenja pokušavajući rekreirati prikazani niz boja. Postoji po jedno tipkalo za svaku od četiri moguće boje, koje se prikazuju pomoću svijetleće diode. Na početku se niz sastoji od tri nasumično izabrane boje. Ako igrač uspješno ponovi zadani niz boja koristeći tipkala, prelazi u novi krug, u kojem se na postojeći niz boja dodaje još jedna, također nasumično izabrana. Sa svakom novom bojom dodanom u niz, povećava se brzina prikazivanja boja. Ukoliko igrač napravi pogrešku pri unosu boja, igra prestaje. Cilj igre je generirati što dulji niz boja bez pogreške.

1.1. Zadatak završnog rada

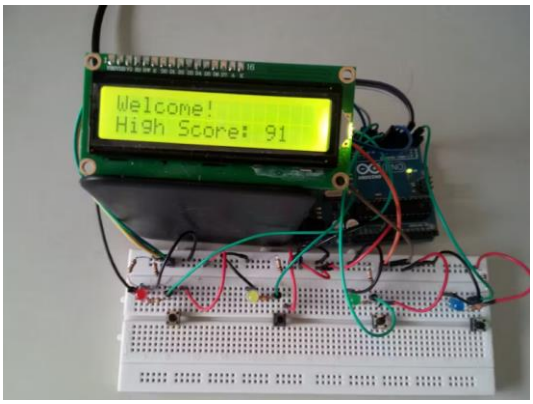
Potrebno je dizajnirati i izraditi PCB (eng. *Printed Circuit Board*) te na nju implementirati igru memorije sa svijetlećim diodama pomoću Arduina.

2. PREGLED PODRUČJA TEME

Na ovu temu je već napisan velik broj radova, pri čemu mnogo njih ima isti koncept s manjim izmjenama. Većina dostupnih radova, projekata, tutoriala i sličnih izvora se razlikuje samo po kôdu koji drugačije postiže istu funkcionalnost, ali moguće je pronaći i izmjene (umjesto svijetlećih dioda koriste se zujalice tako da signali budu zvučni, a ne svjetlosni, slika 2.1.) i proširenja (npr. LCD ekran prikazuje rezultat, slika 2.2.) danog koncepta, kao i potpuno drugačiji tip igre (npr. igra memorije s naizmjeničnim otvaranjem para koji se treba poklapati) [1]. Skoro svaki od dostupnih projekata koristi Arduino te ne bi bilo neutemeljeno pretpostaviti da je razlog tome Arduinova jednostavnost u pogledu implementacije i kodiranja (budući da je Arduino programski jezik utemeljen na jeziku C++, s kojim su mnogi već upoznati kada se počnu baviti Arduinom). Postoje različite komercijalne verzije ove igre koje se mogu pronaći i naručiti (slike 2.3. i 2.4.). Oba navedena proizvoda kombiniraju svjetlosne i zvučne signale te je njihov oblik najveća razlika između njih [2]. Ovakva igra memorije može se također pronaći na mnogim mjestima i u digitalnom obliku. Jedna verzija igre dostupna je aplikaciji *Microsoft Store* potpuno besplatno, te sadrži različite mogućnosti: igra se može igrati u proširenom izdanju (sa šest boja, slika 2.5.) kao i u originalu (sa četiri boje), pri čemu je za svaku opciju vidljiva ljestvica najboljih igrača. Osim toga, postoje tri različite teksture za prikaz boja na ekranu te se mogu promijeniti i zvučni signali, s mogućnošću podešavanja tako da zvuče kao zvona i čak kao mjaukanje mačaka [3]. Od ostalih inačica igre, jedna je implementirana sa sličnim funkcionalnostima, ali tako da se može igrati i ocijeniti unutar web preglednika (u trenutku pisanja sveukupna ocjena je 61/100, slika 2.6.) [4] .



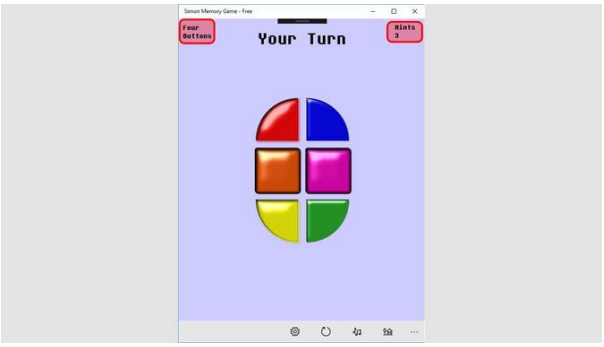
Sl. 2.1. Igra memorije sa zujalicom.



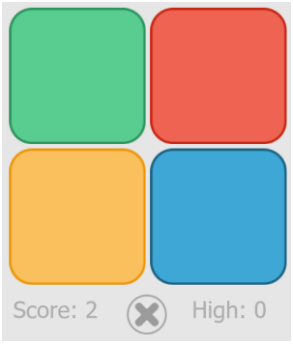
Sl. 2.2. Igra memorije s LCD ekranom.

Sl. 2.3. Komercijalna verzija igre memorije.

Sl. 2.4. Komercijalna verzija igre memorije.



Sl. 2.5. Aplikacija *Simon Memory Game*.



Sl. 2.6. Igra memorije u web pregledniku.

3. IZRADA UREĐAJA

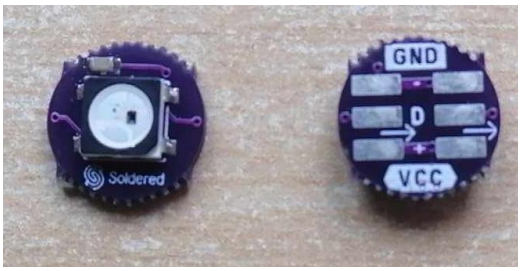
3.1. Sklopovlje

Kako bi bilo moguće izraditi igru i ostvariti njenu potpunu funkcionalnost, potrebno je upotrijebiti određene fizičke komponente koje će obavljati procese potrebne kako bi igra mogla pravilno funkcionirati. Za izradu igre bit će potrebne slijedeće komponente:

- RGB svijetleća dioda
- tipkalo
- baterija
- držač za bateriju
- prekidač
- eksperimentalna pločica
- Arduino Pro Micro.

3.1.1. RGB svijetleća dioda

Za ovu igru potrebno je moći prikazati 4 različite boje, za što je korištena WS2812B SMD LE dioda (slika 3.1.). Ova dioda ima ugrađeni integrirani krug (eng. *Integrated Circuit*) koji joj omogućuje ostvarivanje cjelokupne funkcionalnosti RGB diode preko samo jedne žice tj. samo jednog signalnog pina. Za rad osnovnih funkcija diode potrebno je povezati samo tri od ukupno šest kontakata na poleđini diode: bilo koji od dva gornja kontakta treba povezati na *Ground*, bilo koji od dva donja treba povezati na napon do 5V, a kontakt s lijeve strane oznake D treba povezati na jedan od digitalnih pinova mikroupravljača [5].



Sl. 3.1. WS2812 SMD LE dioda.

3.1.2. Tipkalo

Tipkalo je element koji omogućuje povezivanje i prekidanje strujnog kruga tako što na digitalni ulaz dovodi logičku jedinicu ili logičku nulu. Također je koristan dodatak to što tipkala imaju različito obojane nastavke koji se mogu po potrebni nataknuti ili skinuti, što će olakšati korištenje jer nije potrebno razmišljati o nekom drugom načinu da se prikaže koje tipkalo predstavlja koju boju [6].

Sl. 3.2. Tipkalo.

3.1.3. Baterija

Baterija koja će se upotrijebiti je NCR 18650A litij-ionska baterija koja daje struju napona od 3.7V (slika 3.3.) [7].

Sl. 3.3. NCR 18650A baterija.

3.1.4. Držač za bateriju

Nije preporučljivo lemiti na određene vrste baterija jer pri visokim temperaturama gube svoj kapacitet. Zbog toga je preporučljivo imati držač za bateriju (slika 3.4.) koji omogućuje jednostavno korištenje baterija te ostavlja mogućnost vađenja i punjenja ili zamjene baterija [8].

Sl. 3.4. Držač za bateriju.

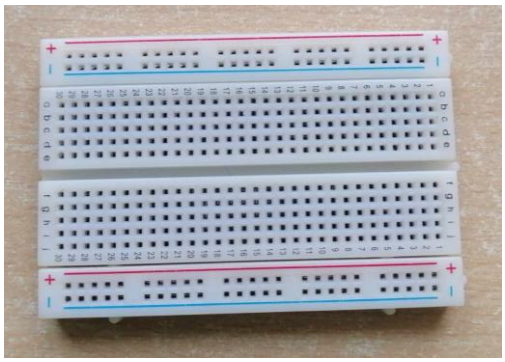
3.1.5. Prekidač

Kako bi bilo moguće raditi sa sklopom bez neprestanog spajanja i odvajanja baterije, dobro je imati prekidač koji omogućuje da baterija bude cijelo vrijeme spojena, ali svejedno se može prekinuti strujni krug u bilo kojem trenutku da se ne bi bez prestanka trošila. Prekidač koji će se koristiti u tu svrhu je modela KCD11 JSH (slika 3.5.) [9].

Sl. 3.5. KCD11 JSH prekidač.

3.1.6. Eksperimentalna pločica

Cilj završnog rada je implementacija igre na PCB, ali zbog jednostavnosti testiranja prototipa uređaja za povezivanje elemenata korištena je eksperimentalna pločica EIC-801 (slika 3.6.). Pločica je strukturirana tako da ima s vanjskih strana po dva reda kontakata s po 25 povezanih utora, a s unutrašnje strane dva stupca s po 30 redova koji sadrže po 5 utora [10].



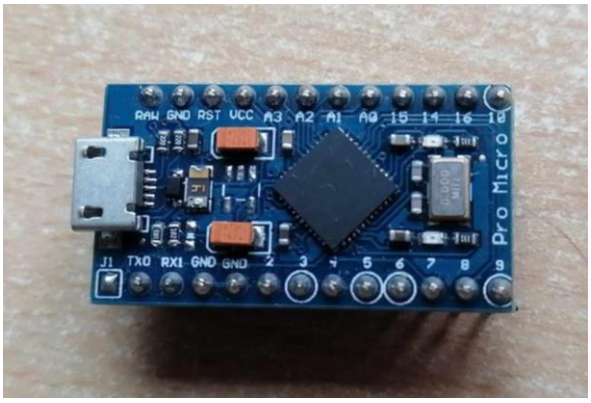
Sl. 3.6. Eksperimentalna pločica EIC-801.

3.1.7. Arduino Pro Micro

Za izvedbu završnog rada bit će iskorišten Arduino Pro Micro (slika 3.7.) koji sadrži 24 nožice:

- 14 nožica koje predstavljaju digitalne pinove (0-10 i 14-16, pri čemu je pin 0 predstavljen nožicom TX0, a pin 1 nožicom RX1)
- četiri nožice koje predstavljaju analogne pinove (A0-A3)
- šest nožica za napajanje.

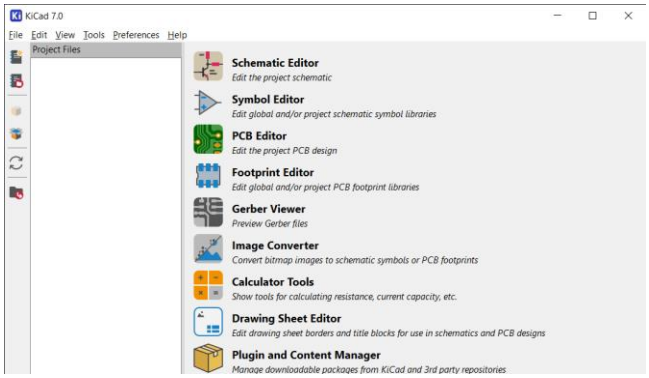
Arduino Pro Micro temelji se na ATmega32U4 8-bitnom mikroupravljačkom sustavu koji ima integrirani mikro USB ulaz tipa B. Postoje dvije varijante ovog mikroupravljača: jedna koja radi na naponu od 3.3V i jedna koja radi na naponu od 5V. Frekvencija sustava ovisi o varijanti, pri čemu jedna varijanta ima 8MHz, a druga 16 MHz [11].



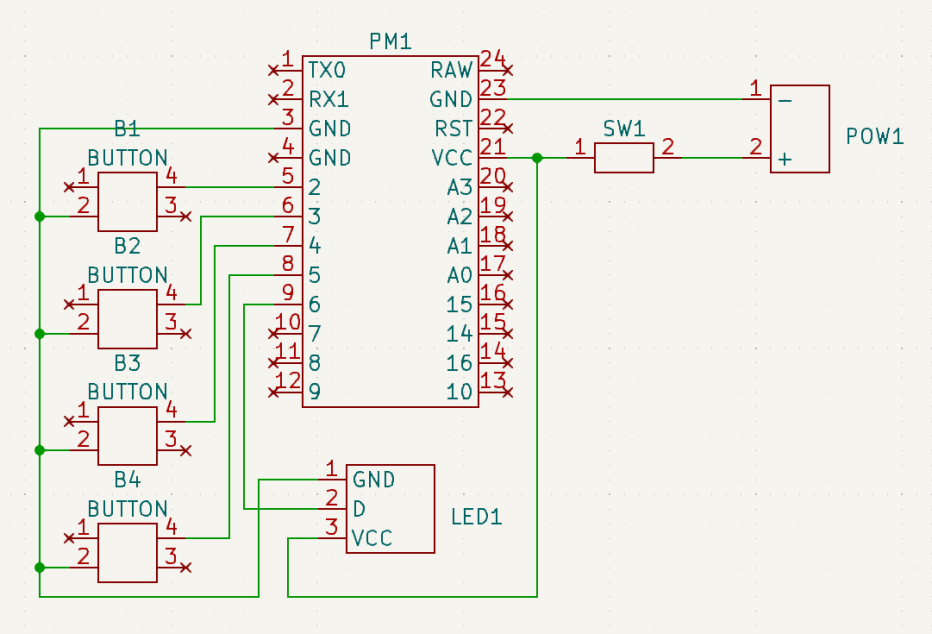
Sl. 3.7. Arduino Pro Micro upravljačka pločica.

3.1.8. Shema

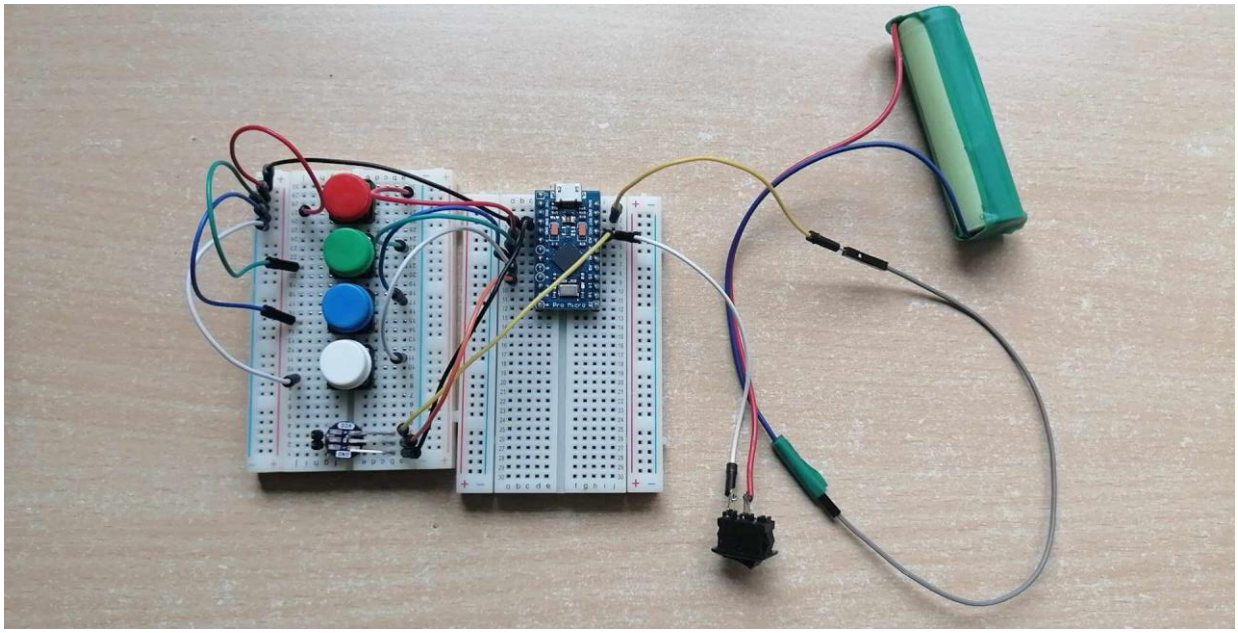
Za izradu PCB-a odabran je KiCad alat, što je jedan od specijaliziranih programa za PCB dizajn (slika 3.8.) [12]. Prvi korak kod razvoja PCB-a je stvaranje sheme koja prikazuje električnu povezanost komponenata sklopa. Shema se uređuje unutar programa *Schematic Editor* kojemu se može pristupiti preko KiCad izbornika. Simboli za pojedine elemente mogu se urediti unutar programa *Symbol Editor*.



Sl. 3.8. KiCad izbornik.



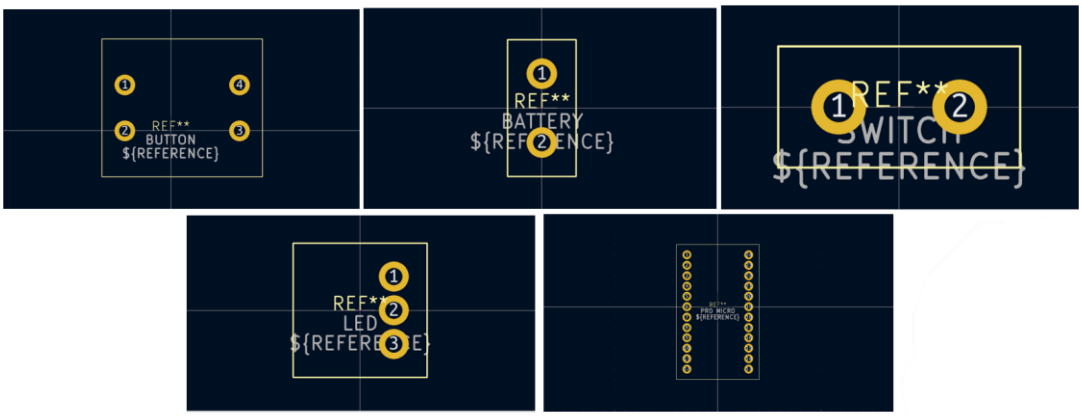
Sl. 3.9. Potpuna shema sustava.



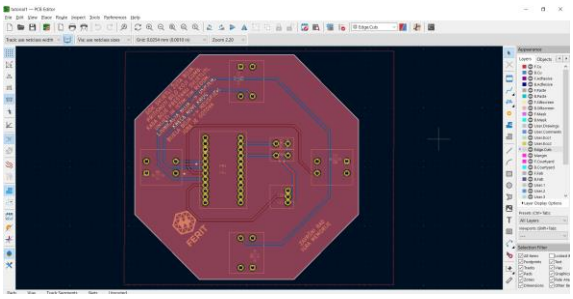
Sl. 3.10. Fizički dio prototipa.

3.1.9. Dizajn tiskane pločice

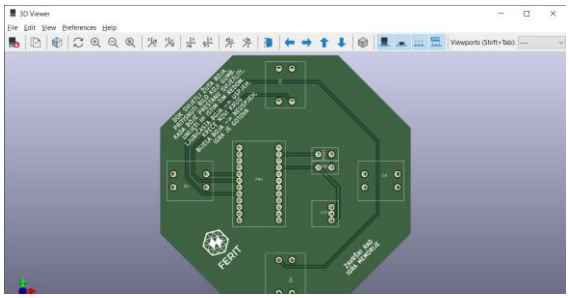
Nakon izrade sheme uređaja, potrebno je dizajnirati strukturu PCB-a. Ovdje je riječ o postavljanju elemenata i vodiča onako kako će biti postavljeni na fizičkoj pločici. Mora postojati barem po jedan otisak (eng. *footprint*) za svaki simbol u shemi (slika 3.11.). Pojedini otisak se može urediti u programu *Footprint Editor*, a *PCB Editor* služi za postavljanje i uređivanje PCB-a (slika 3.12.). Nakon postavljanja i povezivanja otisaka vodičima i određivanja oblika PCB-a, pomoću 3D preglednika se može vidjeti kako će pločica izgledati kada bude fizički izrađena (slika 3.13.).



Sl. 3.11. Otisci za pojedinačne elemente.



Sl. 3.12. PCB unutar PCB Editora.



Sl. 3.13. PCB u 3D pregledniku.

Kada je PCB u potpunosti dizajniran, potrebno je generirati Gerber datoteke i poslati ih na izradu. Unutar glavnog izbornika nalazi se tipka *Plot* koja otvara podprozor s raznim opcijama, među kojima je i generiranje potrebnih datoteka. Tako dobivene datoteke je onda potrebno sažeti u .zip datoteku koja je spremna za slanje na izradu. Za postavljanje narudžbe je korištena stranica JLCPCB koja pruža mnoge mogućnosti za izmjenu PCB-a ili same narudžbe (npr. boja PCB-a, debljina, količina, materijal, …). Nakon postavljanja narudžbe potrebno je nekoliko dana da pločica bude izrađena, nakon čega slijedi period varijabilne duljine (ovisno o odabranim opcijama pri naručivanju) prije nego što PCB bude dostavljen.

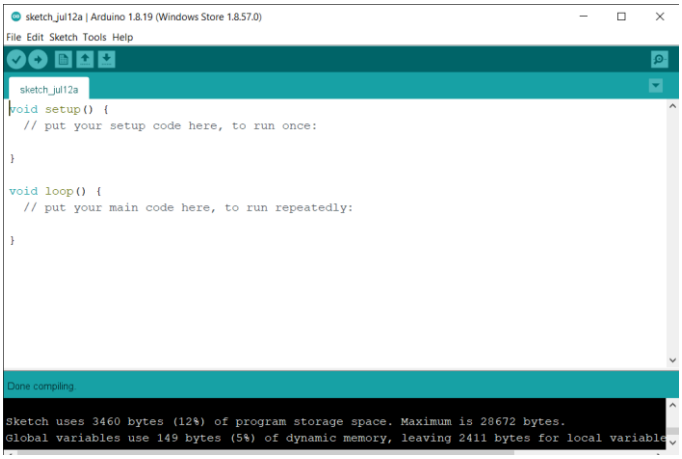
3.2. Programska podrška

3.2.1. Razvojno okruženje

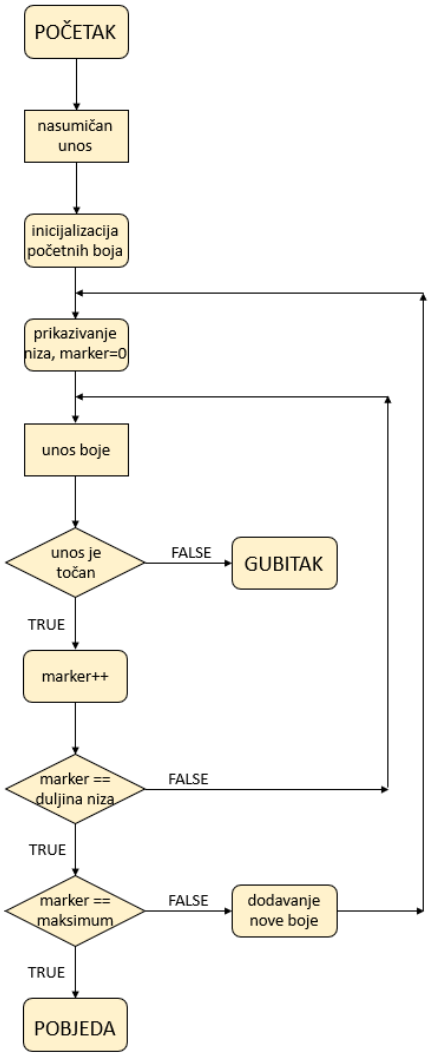
Za razvoj programa ugrađenog računalnog sustava koristi se programsko okruženje Arduino IDE (slika 3.14.) [13]. Sastoji se od dvije alatne trake (jedna za brzi pristup i jedna za pristup proširenim funkcionalnostima), dijela za pisanje kôda i trake za ispis.

3.2.2. Algoritam

Radi jednostavnosti i jasnoće prilikom pisanja kôda, izrađen je dijagram toka (slika 3.15.) koji ugrubo prikazuje kako se program izvodi.



Sl. 3.14. Arduino IDE.



Sl. 3.15. Dijagram toka programa za igru memorije.

3.2.3. Kôd

Kôd se može ugrubo podijeliti na četiri segmenta:

- inicijalizacija biblioteka i globalnih varijabli
- korisnički definirane funkcije
- *setup* dio
- *loop* dio.

Inicijalizacija biblioteka i globalnih varijabli

Linija ***Kod***

```
1:      #include <Adafruit_NeoPixel.h>
2:      #ifdef __AVR__
3:        #include <avr/power.h>
4:      #endif
5:
6:      #define R     2
7:      #define G     3
8:      #define B     4
9:      #define W     5
10:
11:     int period = 250;
12:     int memory[100];
13:     int marker = 0;
14:     int max_value = 100;
15:     int init_value = 3;
16:     bool turn = false;
17:     bool testmode = false;
18:     Bool var = false;
19:     Adafruit_NeoPixel LED(1, 6, NEO_GRB + NEO_KHZ800);
```

Sl. 3.16. Inicijalizacija biblioteka i globalnih varijabli.

Prva linija kôda prikazuje uključivanje biblioteke specifično kreirane za korišteni tip LED diode, nakon čega redovi 2-4 koji služe za poboljšanje rada te biblioteke. Redovi 6-9 predstavljaju oznake specifičnih tipkala za odgovarajuće boje, te su definirani pomoću slova, a ne brojeva kako bi bilo lakše po potrebi mijenjati njihove vrijednosti.

Globalne varijable su redom:

- *period* – označava vremensko trajanje između prikazivanja dvaju boja
- *memory* – polje cijelih brojeva koje predstavlja niz boja
- *marker* – varijabla koja označava boju koja se trenutno unosi/pogađa
- *max_value* – predefinirana maksimalna veličina niza
- *init_value* – predefinirana početna veličina niza
- *turn* – varijabla koja onemogućuje igraču da unosi boje za vrijeme njihovog prikazivanja
- *testmode* – varijabla korištena pri razvoju programa, nije neophodna za rad igre
- *var* – varijabla koja kontrolira duljinu prikaza boja pri unosu.

Zadnja linija kôda je funkcija iz uključene biblioteke. Pomoću nje je deklarirana varijabla LED koja predstavlja niz od N dioda (u ovom slučaju je samo jedna, moguće ih je imati proizvoljan broj) na digitalnom izlazu M (u ovom slučaju je to 6).

Korisnički definirane funkcije

Funkcija *Initialize()*

Linija ***Kod***

```
1:      void Initialize()
2:      {
3:        LED.clear();
4:        LED.show();
5:        Serial.println("Initializing...");
6:        turn=false;
7:        marker=0;
8:        for(int i=0;i<max_value;i++) memory[i]=-1;
9:        for(int i=0;i<init_value;i++) Add();
10:       Play();
11:       turn=true;
12:     }
```

Sl. 3.17. Funkcija *Initialize()*.

Funkcija *Initialize()* je funkcija koja postavlja varijable na njihove početne vrijednosti te triput poziva funkciju *Add()* koja dodaje nasumično izabranu boju u niz (budući da je varijablom *init_value* predefinirano da početni niz ima tri boje). Nakon toga, funkcija *Play()* prikazuje cijeli zadani niz. Linije kôda koje sadrže *Serial.println* ili *testmode* korištene su samo pri razvoju i testiranju programa, te se mogu zanemariti jer nemaju utjecaj na igru.

Funkcija *Check()*

Linija ***Kod***

```
1:      void Check(int input)
2:      {
3:        turn=false;
4:        if(memory[marker]==input)
5:        {
6:          var = true;
7:          Serial.println("yes");
8:          var = false;
9:          Light(input);
10:         if(marker==max_value-1) Success();
11:         else if(memory[marker+1]<0)
12:         {
13:           RoundEnd();
14:           Add();
15:           Play();
16:           marker=-1;
17:         }
18:         marker++;
19:       }
20:       else GameOver();
21:       turn=true;
22:     }
```

Sl. 3.18. Funkcija *Check()*.

Druga bitna korisnički definirana funkcija je *Check()* funkcija koja se poziva pri svakom pritisku nekog tipkala dok je vrijednost varijable *turn* jednaka *true* tj. red je na igraču. Ova funkcija uspoređuje boju koja je na mjestu *marker* u nizu i pritisnuto tipkalo koje odgovara određenoj boji te na taj način provjerava je li pritisnuto točno tipkalo. Ako nije, igra je gotova i poziva se funkcija *GameOver()*. U suprotnom, prikazuje se pritisnuta boja te se izvršava dodatna provjera: ako je marker jednak maksimalnoj veličini niza, poziva se funkcija *Success()*, a ako nije, provjerava se je li marker na zadnjem mjestu u nizu koje ima definiranu boju. Ako je, dodaje se nova boja i počinje novi krug, u suprotnom se marker povećava za jedan. Unutar funkcije *RoundEnd()* implementirana je funkcija *UpdateTime()* koja smanjuje vremenski period između prikazivanja boja ovisno o trenutnoj duljini niza.

Funkcija *setup()*

Linija ***Kod***

```
1:      void setup()
2:      {
3:        Serial.begin(9600);
4:        Serial.println(marker);
5:        pinMode(R, INPUT_PULLUP);
6:        pinMode(G, INPUT_PULLUP);
7:        pinMode(B, INPUT_PULLUP);
8:        pinMode(W, INPUT_PULLUP);
9:        LED.begin();
10:       LED.clear();
11:       LED.show();
12:       LED.setPixelColor(0, LED.Color(255, 255, 0));
13:       LED.show();
14:       while(true)
15:       {
16:         randomSeed(millis()+1);
            if(digitalRead(R)==0 || digitalRead(G)==0 || digitalRead(B)==0
17:     || digitalRead(W)==0)
18:         {
19:           LED.clear();
20:           LED.show();
21:           delay(1000);
22:           break();
23:         }
24:       }
25:       Initialize();
26:     }
```

Sl. 3.19. Funkcija *setup()*.

Kao što je ranije spomenuto, *setup* dio služi za sve operacije i procese koji se trebaju izvršiti jednom pri pokretanju programa. Prvo se inicijalizira komunikacija sa serijskim monitorom, nakon čega se definiraju ulazi i izlazi. Sva četiri tipkala definirana su kao *INPUT_PULLUP*, a LED diodu nije bilo potrebno definirati, budući da smo ju deklarirali još na početku kôda, zajedno s globalnim varijablama. Nadalje, dioda se resetira te počinje neprestano svijetliti žutim svjetlom. Pri pritisku bilo kojeg tipkala, svijetlo se gasi i poziva se funkcija *Initialize()* te igra počinje.

Funkcija *loop()*

Linija ***Kod***

```
1:      void loop()
2:      {
3:        if(turn==true)
4:        {
5:          else if(digitalRead(R)==0) {Check(0);}
6:          else if(digitalRead(G)==0) {Check(1);}
7:          else if(digitalRead(B)==0) {Check(2);}
8:          else if(digitalRead(W)==0) {Check(3);}
9:        }
10:     }
```

Sl. 3.20. Funkcija *loop()*.

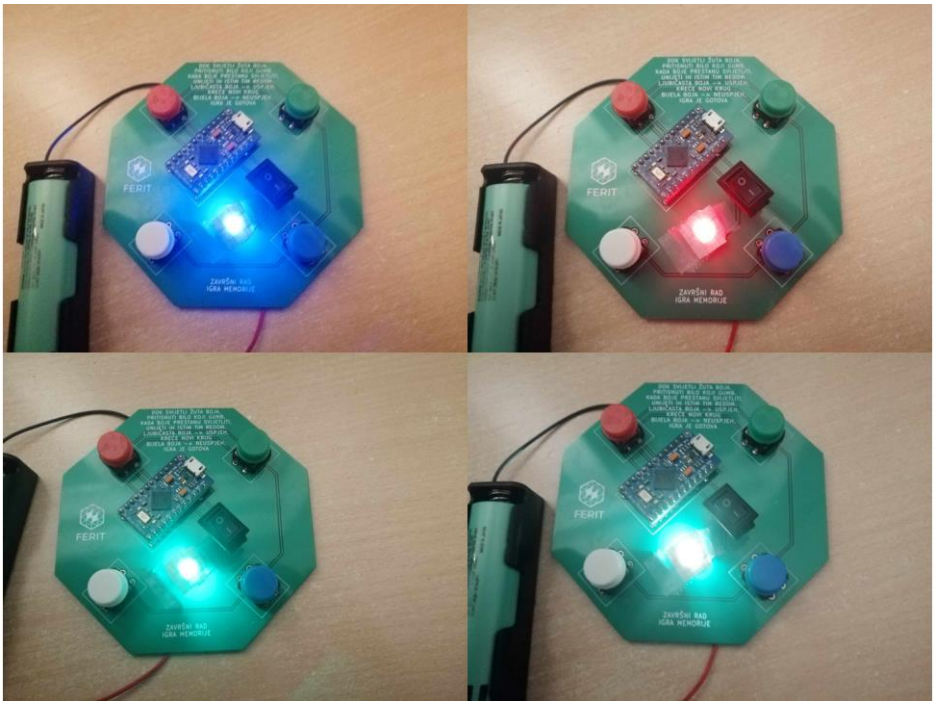
U *loop* dijelu programa se nalazi kôd koji se beskonačno ponavlja nakon što se završi *setup*. Prvo se izvodi provjera varijable *turn* (da se provjeri je li red na igraču), a nakon toga se provjerava je li pritisnuto neko tipkalo. Ako je, poziva se funkcija *Check()* uz vrijednost pritisnutog tipkala kao argument. Funkcija *loop* će se prestati ponavljati kad sustav bude isključen ili kad igrač napravi pogrešku pri unosu boja. Pri unosu pogrešne boje program ulazi u funkciju *GameOver()* iz koje ne može izaći, a za to vrijeme će izvršavanje funkcije *loop* biti zaustavljeno.

3.3. Testiranje rada

Sklop je tijekom razvoja često bio testiran, pogotovo pri razvoju kôda. Većina testova je ukazala na postojanje jednostavnih previda (npr. boje se izmjenjuju prebrzo ili presporo) uz manji broj testova koji otkrivaju propuste u programu (npr. mogućnost povratka u glavni dio programa nakon gubitka igre). Također, program je testiran s različitim načinima postizanja nasumičnosti:

- početne varijable ovise o funkciji *millis()* → početne varijable su uvijek iste
- klik na gumb određuje početak inicijalizacije → početne varijable su uvijek istog rasporeda
- nasumičnost pomoću funkcije *random()* → boje su pseudoslučajne tj. niz je uvijek isti
- klik na gumb inicijalizira *randomSeed* pomoću funkcije *millis()* → nasumične boje

Nakon završetka izrade igre, funkcionalnost je testirana s različitim igračima. Prosječni rezultat iznosi 10.25 bodova (pri čemu je jedan bod za svaki uspješno završeni krug), a prosječno vrijeme trajanja igre je 2m 49s. Slika 4.1. prikazuje kako izgleda završeni uređaj dok svijetli svaka od 4 boje koje se mogu pojaviti u nizu.



Sl. 4.1. Sklop svijetli u svakoj od 4 moguće boje: a) plavo b) crveno c) zeleno i d) bijelo.

4. ZAKLJUČAK

Zadatak je dizajnirati i izraditi PCB te implementirati igru memorije sa svijetlećim diodama pomoću Arduina. Igra počinje s nizom od tri boje. Svaki put kada niz bude točno unesen, povećava se za jednu boju. Brzina prikazivanja boja se postepeno povećava ovisno o duljini generiranog niza.

Kako bi bilo moguće izraditi igru pamćenja, prvo je bilo potrebno pronaći fizičke komponente. Povezivanjem tih komponenti dobiven je prototip sklopa koji je korišten radi jednostavnosti testiranja programa. Nakon upoznavanja s KiCad-om, izrađen je PCB na koji su komponente postavljene i zalemljene. Konačno, u programskom jeziku Arduino razvijen je kôd koji implementira igru na mikroupravljački sustav.

Unatoč tome što je zadatak završen, funkcionalnost igre se još uvijek može mijenjati. U slučaju da se želi promijeniti način na koji se računa varijabla *period*, implementirati drugačiji mehanizam za postizanje nasumičnosti ili povećati početnu veličinu niza, može se jednostavno vratiti i izmijeniti kôd. Postoje i mnogi drugi načini na koji bi se igra mogla izmijeniti, ali sve te ideje i zamisli predstavljaju ciljeve za potencijalno unaprjeđenje igre.

LITERATURA

[1] M., Geddes, *Arduino project handbook*. No Starch Press: San Francisco, 2016.

[2] „Electronic Memory Game - Memory Buzzer, Classic Memory Maze Educational Toy | Mini Memory Maze Educational Toy with LED Light and Sound: Amazon.de: Toys“ [online]. Dostupno na: https://www.amazon.de/-/en/Electronic-Memory-Game-Classic-Educational/dp/B0CBJPL4LM. [Pristupljeno: 31.8.2023.].

[3] „Get Simon Memory Game - Free - Microsoft Store“ [online]. Dostupno na: https://www.microsoft.com/en-us/p/simon-memory-game-free/9n5k21lpp66v?activetab=pivot:overviewtab. [Pristupljeno: 13.9.2023.].

[4] „Play Simon Says Online for Free: Kids Memory Game treZimon Inspired by Simon Says“ [online]. Dostupno na: https://www.calculators.org/games/simon-says/. [Pristupljeno: 13.9.2023.].

[5] „Pametni LED WS2812B Pixel“ [online]. Dostupno na: https://soldered.com/hr/proizvod/pametni-led-ws2812b-pixel/. [Pristupljeno: 31.8.2023.].

[6] M., Banzi, *Getting Started with Arduino*, 2. Aufl. O'Reilly & Associates: Sebastopol, CA, 2009.

[7] „[UMIROVLJEN] 18650 Li-ion baterija Panasonic NCR18650PF 2900mAh 10A“ [online]. Dostupno na: https://soldered.com/hr/proizvod/18650-li-ion-baterija-panasonic-ncr18650pf-2900mah-10a/. [Pristupljeno: 31.8.2023.].

[8] „Držač za 18650 litijsku bateriju“ [online]. Dostupno na: https://soldered.com/hr/proizvod/drzac-za-18650-litijsku-bateriju/. [Pristupljeno: 31.8.2023.].

[9] „KCD11 Rocker Switch 3A (4 Pack) - Micro Robotics“ [online]. Dostupno na: https://www.robotics.org.za/KCD11. [Pristupljeno: 31.8.2023.].

[10] „Kysan Electronics“ [online]. Dostupno na: http://www.kysanelectronics.com/Products/cat_pro_rev.php?recordID=2601. [Pristupljeno: 31.8.2023.].

[11] „Pro Micro - 3.3V/8MHz - DEV-12587 - SparkFun Electronics“ [online]. Dostupno na: https://www.sparkfun.com/products/12587. [Pristupljeno: 31.8.2023.].

[12] J., Evans, „Documentation | KiCad“ [online]. Dostupno na: https://docs.kicad.org/. [Pristupljeno: 31.8.2023.].

[13] E., Verberne, „Arduino documentation“ [online]. Dostupno na: https://docs.arduino.cc/. [Pristupljeno: 31.8.2023.].

SAŽETAK

Tema ovog završnog rada je izrada igre pamćenja sa svijetlećom diodom na mikroupravljačkoj platformi Arduino. Korišten je specifično model Arduino Pro Micro, a funkcionalnost je isprogramirana u razvojnom okruženju Arduino IDE. Programski kôd se sastoji od četiri glavna dijela: inicijalizacije biblioteka i globalnih varijabli, korisnički definiranih funkcija, funkcije *setup()* i funkcije *loop()*. Na početku igre dioda svijetli žutom bojom, nakon čega pritiskom bilo kojeg tipkala igra započinje te se prikazuju tri nasumično izabrane boje. Pri točnom unosu tih boja istim redoslijedom krug završava, pri čemu se u niz dodaje nova nasumično izabrana boja te započinje novi krug. Igra traje dok jedna od boja ne bude nepravilno unesena ili dok veličina niza boja ne prijeđe 100 (u teoriji bi pravilnim unosom boja igra trebala trajati beskonačno dugo, ali zbog praktičnih razloga potrebno je imati unaprijed definiranu maksimalnu veličinu niza boja). Testiranjem igre utvrđeno je da nema pogrešaka u radu, što znači da je ostvareno potpuna funkcionalnost.

Ključne riječi: Arduino, igra pamćenja, RGB LED dioda, tipkalo

ABSTRACT

Arduino-based Memory Game with a LED diode

The topic of this bachelor thesis is creating a memory game with a LED diode on the Arduino microcontroller platform. The model which was used is specifically Arduino Pro Micro, and the game was programmed and compiled in the Arduino IDE development environment. The code consists of four main parts: library inclusions and global variables, user defined functions, the *setup()* function and the *loop()* function. In the beginning the diode lights up yellow, after which pressing any button stars the game and shows the sequence of three randomly generated colours. The correct input of these colours ends the round and a new randomly generated colour is added to the sequence before the next round begins. The game ends once there is an incorrect input or the sequence size reaches 100 (theoretically, the game should be endless if inputs are correct, but for practical reasons there has to be a predefined maximum of the size of the sequence). Testing the game was successful and no bugs were detected, which means the full functionality was achieved.

Keywords: Arduino, memory game, RGB LED diode, button

PRILOZI

U prilozima se nalaze:

- Arduino projekt završnog rada
- KiCad projekt završnog rada
- kopija obrasca Z1S/Z1P
- izjava o originalnosti završnog rada
- izjava o pohrani i objavi ocjenskog rada u repozitoriju u sustavu Dabar.