# IGRA KRUŽIĆ-KRIŽIĆ TEMELJENA NA ARDUINO PLATFORMI

**Vincek, Gabrijel**

**Undergraduate thesis / Završni rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:190:463271

*Rights / Prava:* Attribution 4.0 International/Imenovanje 4.0 međunarodna

*Download date / Datum preuzimanja:* **2025-02-28**

*Repository / Repozitorij:*

Repository of the University of Rijeka, Faculty of Engineering

VEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

# IGRA KRUŽIĆ- KRIŽIĆ TEMELJENA NA ARDUINO PLATFORMI

Rijeka, srpanj 2024.

Gabrijel Vincek

0069086759

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

# IGRA KRUŽIĆ- KRIŽIĆ TEMELJENA NA ARDUINO PLATFORMI

Mentor: Doc. dr. sc. Ivan Volarić

Rijeka, srpanj 2024.

Gabrijel Vincek
0069086759

**IZJAVA**

Sukladno s člankom 9. stavak 1) Pravilnika o završnom radu, završnom ispitu i završetku preddiplomskih sveučilišnih studija Tehničkog fakulteta Sveučilišta u Rijeci, izjavljujem da sam samostalno izradio završni rad pod naslovom „IGRA KRUŽIĆ - KRIŽIĆ TEMELJENA NA ARDUINO PLATFORMI“, od 21.03.2023. godine, koristeći se navedenom literaturom, znanjem stečenim tijekom dosadašnjeg studija i konzultacijama s mentorom Doc. dr. sc. Ivanom Volarićem.

Rijeka, srpanj 2024.

_____

Gabrijel Vincek

Zahvaljujem se mentoru doc. dr. sc. Ivanu Volariću na nesebičnoj pomoći, prijedlozima i smjernicama prilikom izrade ovog završnog rada.

Također se zahvaljujem svojim roditeljima, užoj obitelji, djevojci i prijateljima kao i svim profesorima i asistentima na bezuvjetnoj podršci, savjetima i razumijevanju tokom cijelog školovanja.

# SADRŽAJ

1. UVOD ..... 1

2. MIKROKONTROLERI ..... 2

2.1 Uvod ..... 2

2.2 Platforme ..... 3

2.3 Arduino UNO ..... 3

3. KRUŽIĆ-KRIŽIĆ ..... 7

3.1 Povijest igre ..... 7

3.2 Hardware-ski dio sustava ..... 8

4. IGRA PROTIV RAČUNALA – MINIMAX FUNKCIJA ..... 15

5. OPIS PROGRAMSKOG KODA ..... 19

5.1 Definiranje varijabli ..... 19

5.2 Funkcija void setup() ..... 20

5.3 Funkcija void loop() ..... 21

5.4 Funkcija void citanje_start() ..... 23

5.5 Funkcija void citanje_tipki() ..... 24

5.6 Funkcija void pocetak() ..... 25

5.7 Funkcija void error() ..... 25

5.8 Funkcije void igra_P1() i void igra_P2() ..... 26

5.9 Funkcija void pretvorba_za_led() ..... 26

5.10 Funkcija void provjera() ..... 27

5.11 Funkcija void kraj_igre() ..... 27

5.12 Funkcija Move findBestMove(int borard[3][3]) ..... 28

5.13 Funkcija int evaluate(int b[3][3]) ..... 29

5.14 Funkcija bool isMovesLeft(int board[3][3]) ..... 31

5.15 Funkcija int MiniMax(int board[3][3], int depth, bool isMax) ..... 31
6. ZAKLJUČAK ..... 33
LITERATURA ..... 34
POPIS OZNAKA I KRATICA ..... 35
SAŽETAK ..... 36
ABSTRACT ..... 37

# 1. UVOD

Razvoj tranzistora u drugoj polovici prethodnog stoljeća, omogućio je sve manje i manje implementacije računalnih sustava, što je uvelike omogućilo i olakšalo njihovu primjenu u svim granama industrije pa i u svakodnevnom životu. Mikrokontroleri su samim time mala računala, koja mogu služiti za upravljanje i regulaciju različitim elementima poput motora, sklopka, zaslona, audio-video tehnike, te za jednostavne (i složene) programe i proračune.

Cilj ovog završnog rada je izrada igre kružić-križić uz pomoć Arduino razvojne pločice u dva načina igre. U prvom načinu, međusobno igraju dva igrača, dok u drugom načinu mikrokontroler preuzima ulogu drugog igrača koji igra bez greške odnosno ne može izgubiti. Ulazi su ostvareni pomoću tipkala spojenih u *pull-down* konfiguraciju, a izlaze pomoću programibilnih RGB LED dioda.

Ovaj završni rad organiziran je na sljedeći način. U drugom poglavlju opisana je Arduino UNO razvojna pločica, njene komponente, te slične platforme. Treće poglavlje opisuje igru kružić-križić, te je opisan način spajanja svih komponenti. U četvrtom poglavlju prezentiran je algoritam koji igra igru na način da ne može izgubiti, a peto poglavlje detaljno opisuje programski kod razvijen u sklopu ovog završnog rada. Zadnje, šesto poglavlje daje zaključak ovog završnog rada.

# 2. MIKROKONTROLERI

## 2.1 Uvod

Mikrokontroler se sastoji od jednog ili više procesora (eng. *central procesing unit* - CPU), memorije, te programibilnim ulaznim i izlaznim periferijama koje se nalaze unutar jednog integriranog kruga. Oni se uglavnom koriste kao centralni dio ugradbenog sustava, a mogu se pronaći u kalkulatorima, satovima, MP3-playerima, itd.

Danas se najviše upotrebljavaju 32-bitni mikrokontroleri poput STM32, ATMega328P, PIC16F877A, Attiny85, itd. [1]. Jedni od popularnijih su mikrokontroleri iz STM32 serije budući da pružaju veliku brzinu obrade podataka, široki skup perifernih uređaja te nisku potrošnju energije, čime je uvelike olakšana njihova upotreba od industrijskih pa do potrošačkih proizvoda. Jedan od češće korištenih mikrokontrolera iz te serije je STM32F0, koji se temelji na Cortex-M0 jezgri s 32-bitnoj ARMv6-M arhitekturom, te radnom taktu do 48MHz i potrošnjom do 2.4μA/MHz, zbog čega je pogodan za primjenu kod jednostavnijih sustava i sustava niske potrošnje. Osnovne značajke koje nudi su 256 kilobajta (Kb) flash memorije, 32 Kb SRAM-a (eng. *static random-access memory*), cikličke provjere redundantnosti (eng. *cyclic redundancy check* - CRC), napajanje od 3.6 V, 32 kHz oscilator za sat stvarnog vremena (eng. *real time clock* - RTC) s kalibracijom, 55 ulazno/izlaznih pinova, 5 kanalni DMA (eng. *direct memory access*) kontroler, kalendar RTC s alarmom i periodičnim buđenjem iz Stop/Standby-a, 11 timera, dva I2C (eng. *inter-integrated circuit*) i SPI (eng. *serial peripheral interface*) komunikacijska sučelja, te SWD (eng. *serial wire debug*) mogućnost [2, 3, 9].

Za ovaj završni rad korišten je ATMega328P mikrokontroler u sklopu Arduino razvojne pločice. To je 8-bitni mikrokontroler s radnim taktom od 16MHz. Pruža 32 Kb flash memorije, 1 Kb EEPROM (eng. *electrically erasable programmable read-only memory*) i 2 Kb SRAM-a. Ima dva 8-bitna timera te jedan 16-bitni timer, brojač u stvarnom vremenu s odvojenim oscilatorom, šest kanala za pulsnu širinsku modulaciju (eng. *pulse-width modulation* - PWM), jedno I2C sučelje, analogni komparator, 23 ulazno/izlazna pina, itd. Napaja se naponom od 2.7-5.5 V, s prosječnom potrošnjom u aktivnom modu od 1.5 mA na 3V i radnim taktom od 4 MHz. Može raditi na temperaturama od -40°C do 125°C [10].

## 2.2 Platforme

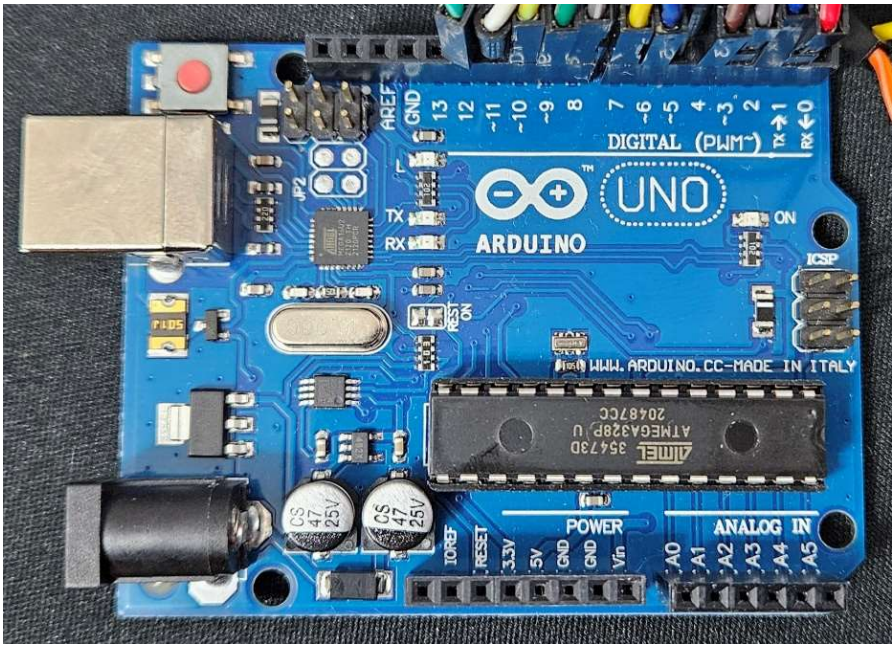
Arduino i RaspberryPI su jedne od češće korištenih platformi koje se koriste za raznovrsna upravljanja izvršnim elementima, te za upravljanje IOT (eng. *Internet Of Things*) sustavima. IOT je sustav koji omogućava komunikaciju kontrolera putem interneta za praćenje i upravljanje dijelovima sustava. Dodatno se koriste za razvoj prototipova koji služe kao dokaz mogućnosti proizvoda.

Glavne razlike između njihovih proizvoda su da je RaspberryPI samostalno računalo na kojemu je instaliran operativni sustav (OS) temeljen na Linuxu i kojemu se može pristupiti spajanjem monitora na samu pločicu, dok Arduino proizvodi razvojne pločice koje nemaju svoj vlastiti OS te je za njih potrebno posebno računalo pomoću kojeg kompajlerem pretvaramo napisani kod u njemu razumljive instrukcije.

Bitno je napomenuti da oboje platforme pružaju vlastite besplatne *open-source* programe koji iza sebe imaju ogromnu podršku online zajednice. Samim time vrlo je lako pronaći besplatna objašnjenja i upute za sve potrebe kodiranja programa te izvođenja i spajanja izvršnih elemenata [5, 6].

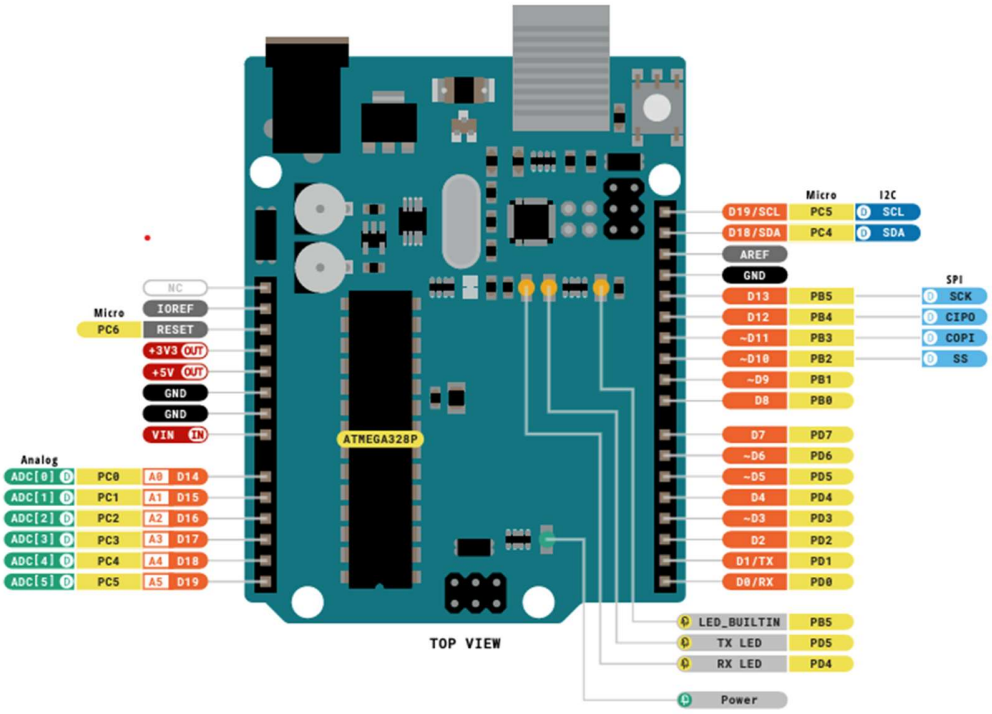
## 2.3 Arduino UNO

Arduino razvojna pločica korištena za potrebu završnog rada je Arduino UNO R3, prikazana na slici 2.1., s ATMega328P mikrokontrolerom prikazanim na slici 2.2. Arduino UNO razvojna pločica je opremljena USB tip B konektorom za komunikaciju s računalom, koji ujedno služi kao izvor napajanja, zatim 14 digitalnih ulaza/izlaza, 6 analognih ulaza, 16 MHz keramički oscilator, ISCP (eng. *in-circuit serial programming*) konektorom te tipkalo za reset kako je prikazano na slici 2.3 [6, 7]. U ovom završnom radu iskorišteno je 12 digitalnih pinova, te dva pina za napajanje (5V i GND), kako je prikazano na slici 2.4.

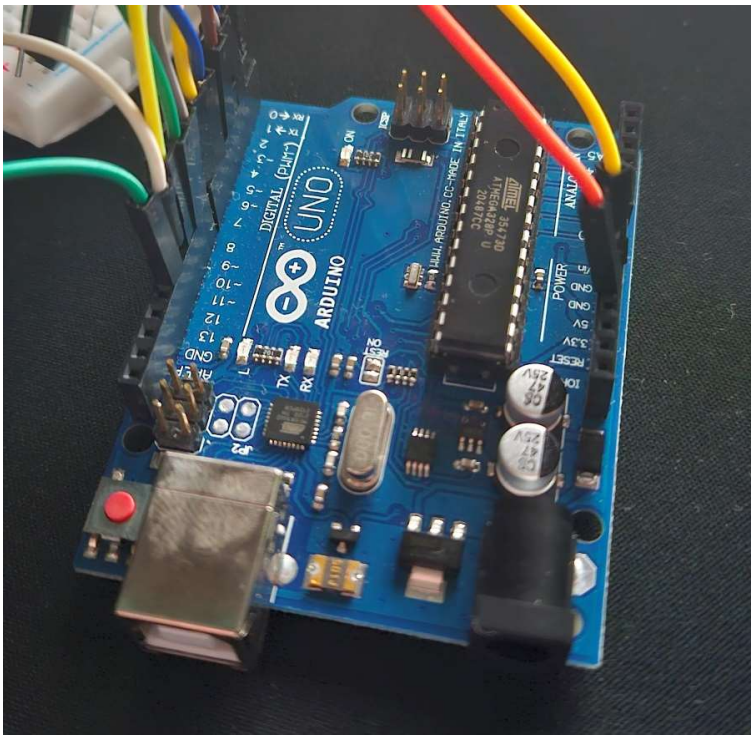



Slika 2.1. Arduino UNO R3 razvojna pločica.


Slika 2.2. ATMega328P mikrokontroler.

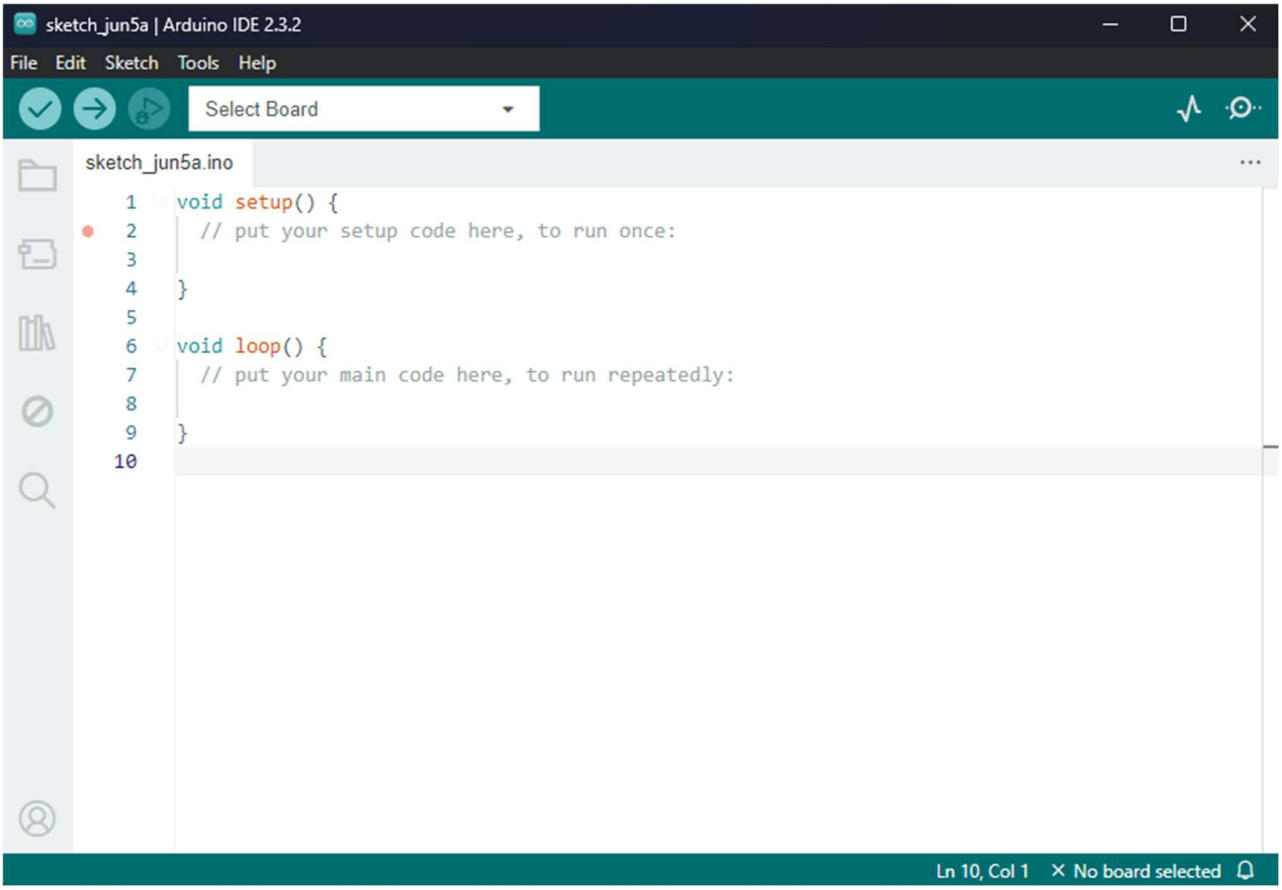



Slika 2.3. Raspored pinova na Arduino UNO R3 razvojnoj pločici [6].




Slika 2.4. Korišteni pinovi na ARDUINO UNO R3 razvojnoj pločici.

Za programiranje je korišteno razvojno sučelje Arduino IDE verzija 2.3.2, prikazano na slici 2.5., koji se temelji na C++ jeziku te omogućuje jednostavno prenošenje programskog koda na mikrokontroler i veliki pristup bibliotekama za lakše programiranje. Kao dodatna svojstva sadrži automatsko dovršavanje, navigaciju kodom, te live debugger [6].

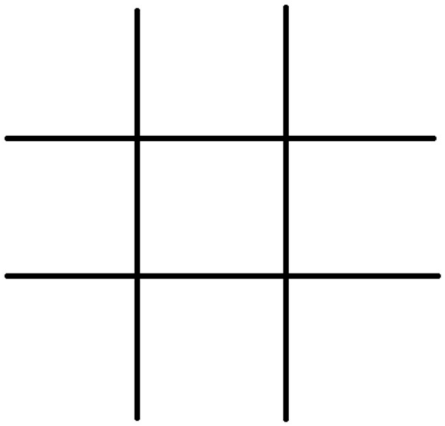



Slika 2.5. ARDUINO IDE v.2.3.2 razvojno sučelje.

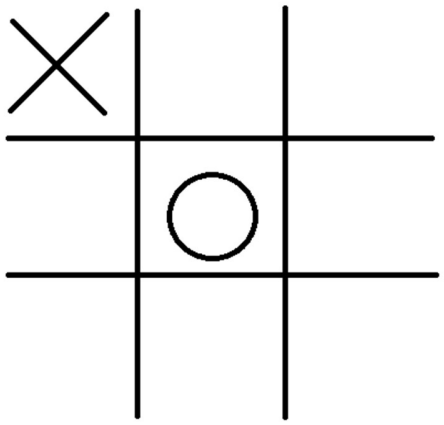
# 3. KRUŽIĆ-KRIŽIĆ

## 3.1 Povijest igre

Kako bi se mogao shvatiti zadatak prvo je potrebno opisati samu igru te je staviti u povijesni kontekst. Kružić-križić je igra za dva igrača u kojoj svaki igrač naizmjence postavlja križić „X“ ili kružić „O“ na 3x3 polje prikazano slikom 3.1. Proizvoljno odabrani prvi igrač stavi prvi „X“ na jedno od praznih polja, nakon toga drugi igrač stavlja svoj „O“ na jedno od preostalih dostupnih polja i nadalje se izmjenjuju kako je prikazano na slici 3.2. Pobjednik je onaj koji uspije spojiti tri ista znaka u horizontalnom, vertikalnom ili dijagonalnom smjeru. U slučaju kada niti jedan igrač ne uspije ostvariti taj cilj igra završava neriješeno. Igra je u suštini vrlo jednostavna, te u igri između dva iskusna igrača rezultat će uvijek biti neriješen.



Slika 3.1. Igrače polje.



Slika 3.2. Početak igre.

Prema publikaciji MO Math nacionalnog muzeja za matematiku (SAD) „Tic-Tac-Toe“: „Prvi tragovi igre kružić-križić sežu u Egipat, gdje su otkriveni ostatci 3x3 igraćih ploča na krovovima kuća koje datiraju iz 1300. godine prije Krista. Druge varijante uključivale su „Terni Lapilli“ (odnosno igra tri kamenčića odjednom) iz Rimskog Carstva, igre „Mlin“ iz raznih dijelova Azije i „Picaria“ američkih domorodaca. Naziv kružić-križić pojavio se prvi puta u znanstvenom časopisu iz 1858. godine pod nazivom „Nule i križići“. Nadalje ovo je jedna od prvih igara koja je implementirana na računalu koju je posebno razvio britanski računalni znanstvenik Alexandar S. Douglas 1952.godine, dok su 1975.

godine studenti MIT-a razvili igru kružića-križića pomoću računala Tinkertoy u kojoj računalo nije moglo izgubiti“ [11].

## 3.2 Hardware-ski dio sustava

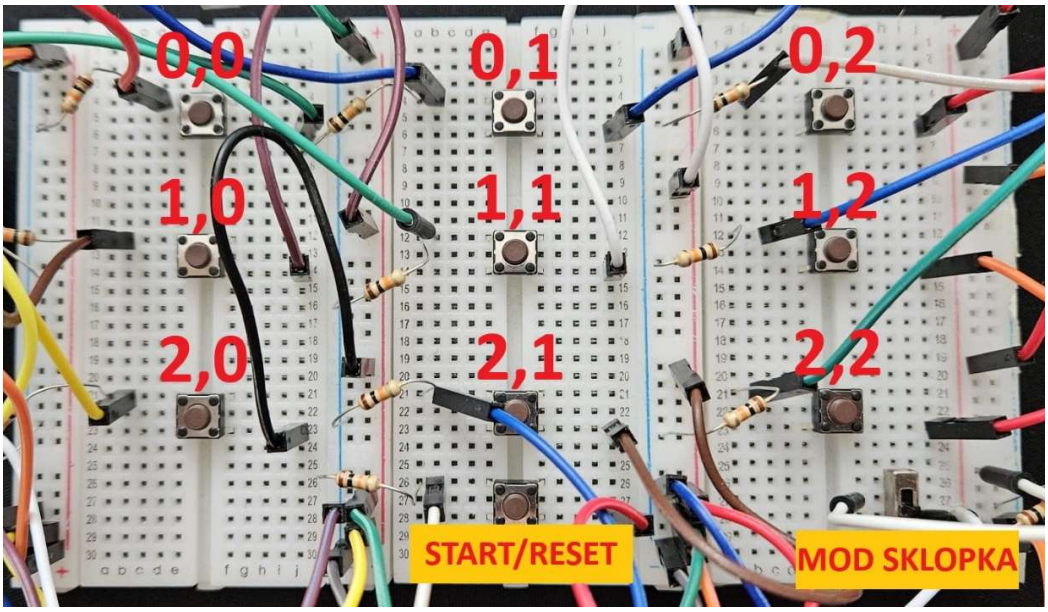
Cilj ovog završnog rada je izrada sustava za igranje igre kružić-križić putem Arduino platforme s dva načina rada, prvi način gdje dva igrača igraju međusobno, a drugi gdje igrač igra protiv računala koje ne može izgubiti. Kako bi se igra mogla preseliti iz papirnatog u digitalni oblik, potrebno je osmisliti način na koji će se stanje igre zapisati u memoriji mikrokontrolera. Zbog jednostavnosti igre križić kružić, rješenje se samo nameće u obliku 3x3 matrice kako je prikazano na slici 3.3. [12].

| | | |
|---|---|---|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

Slika 3.3. Matrica koja predstavlja stanje igre.

Ulazi korisnika izvedeni su putem tipkala spojenih u *pull-down* konfiguraciju u svrhu detekcije pozitivnog brida. Za potrebe rada, ukupno je korišteno 10 tipkala, od kojih devet za odabir polja, te jedno za resetiranje igre, kako je prikazano na slici 3.4. Prvim pritiskom tipkala za odabir polja u pripadajući element matrice zapisuje se „1“ što naznačuje da je prvi igrač postavio križić. Sljedećim pritiskom tipkala u matricu se zapisuje „2“ što naznačuje da je drugi igrač postavio kružić. Na sličan način svi sljedeći pritisci tipkala naizmjenično zapisuju „1“ ili „2“ u matricu koje predstavlja stanje igre, sve do završetka igre: bilo to do pobjede jednog igrača, ili dok se ne popune svi elementi matrice.

Za odabir načina igre korištena je sklopka prikazana na slici 3.5.



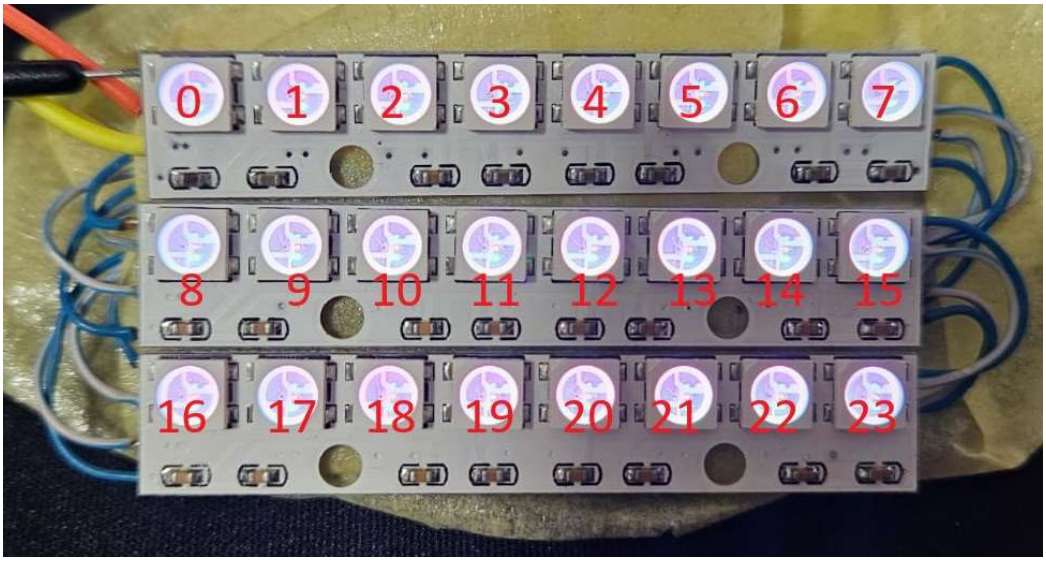

Slika 3.4. Fizička izvedba ulaznih tipkala i sklopke.


Slika 3.5. Sklopka za odabir načina igre.

Prikaz igre je implementiran pomoću tzv. *NeoPixel*-a, tj. programibilnih RGB LED dioda. Točnije rečeno, izveden je pomoću tri WS2812B-8 modula koji sadrže 8 takvih LED dioda poredanih u jednom pravcu, fizički raspoređenih tako da zajedno tvore jednu 8x3 matricu kao što je to prikazano na slici 3.6. Pločice su spojene kaskadno pomoću direktno lemljenih žica na pinove za napajanje (VCC i GND) te jedan digitalni ulaz, što je prikazano na slici 3.7. LED diode imaju opciju prikaza $256^3 = 16777216$ različitih boja s 256 različitih intenziteta svjetlosti. Za indikaciju poteza prvog igrača korištena je zelena boja, dok za drugog igrača plava boja. U tako spojenoj konfiguraciji, enumeracija LED dioda je prikazana na slici 3.8. [13].

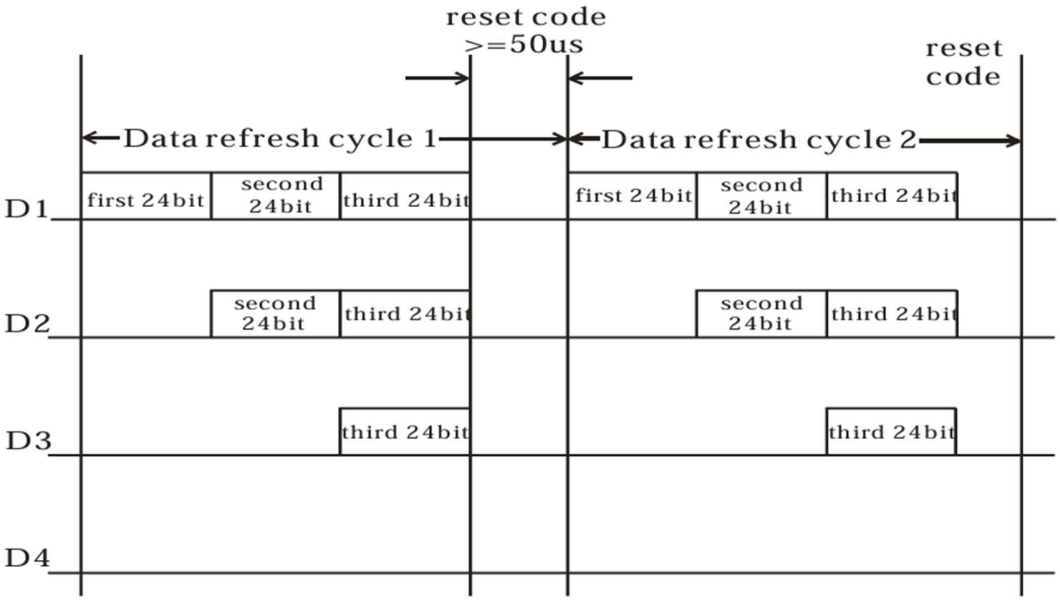
Slika 3.6. Tri WS2812B-8 modula raspoređenih u 8x3 matricu.


Slika 3.7. Prikaz WS2812B-8 modula sa stražnje strane.



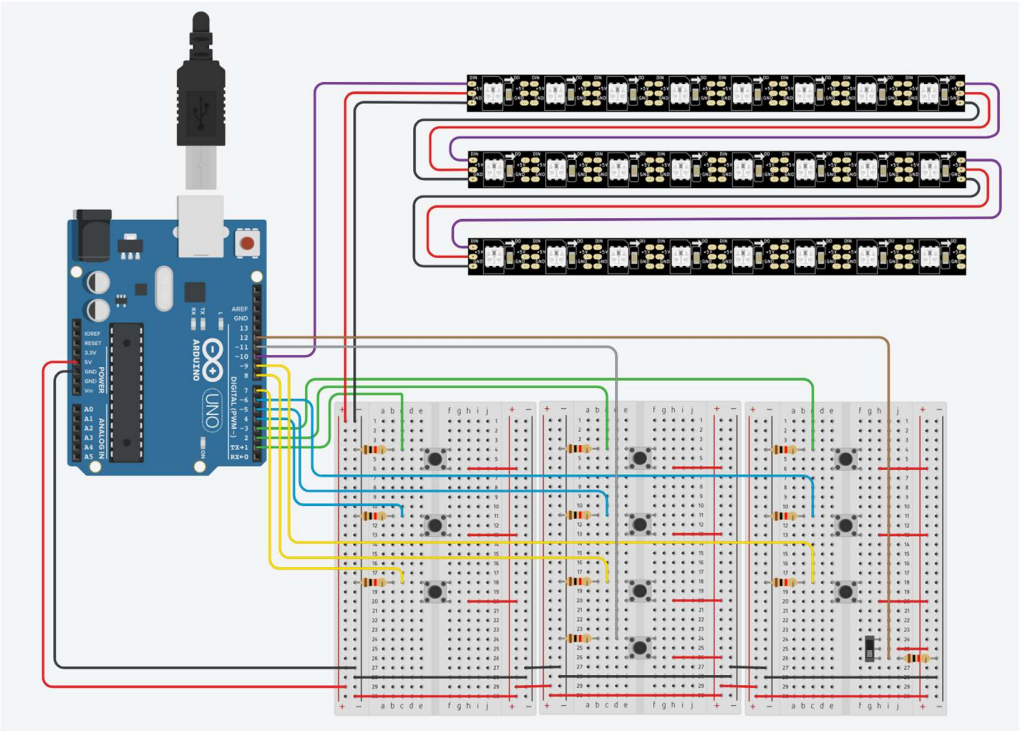

Slika 3.8. Enumeracija LED dioda.

Svaka RGB LED dioda je vezana za svoj mikrokontroler, već ugrađen u njeno kućište, koji omogućava čitanje paketa bitova koji sadrže podatke o boji i intenzitetu svjetlosti LED dioda. Za upravljanje se koristi NZR (eng. *Non-Zero-Return*) komunikacijski protokol. Arduino Uno razvojna pločica, tj. mikrokontroler koji se nalazi na pločici šalje podatkovne pakete veličine 24 bita za svaku RGB LED diodu. Mikrokontroler prve RGB LED diode uzima podatke iz prvog paketa, interpretira ih, te prema njima uključuje svoju LED diodu. Sve sljedeće pakete, prvi mikrokontroler prosljeđuje na svoj izlaz, koje zatim intepretira sljedeći mikrokontroler, i tako do kraja kaskade. Takvo ponašanje omogućuje laku kontrolu boje i jačine svjetlosti svake pojedine RGB LED diode pomoću jedne komunikacijske linije. Prijenos paketa bitova opisan je na slici 3.9 gdje *Data refresh cycle 1* predstavlja prvi set instrukcija LED diodama. *First 24 bit* označava prvi paket od 24 bitova drugi *second* treći *third* te tako nadalje do kraja kaskade, dok su diode označene s D1-D4. Prva dioda u kaskadi prima prvi 24-bitni paket, dok sve ostale prosljeđuje drugoj diodi u kaskadi. Druga dioda prima prvi 24-bitni paket koji primi (što je zapravo drugi globalni paket), te prosljeđuje sve ostale. Vrijeme od 50 mikro sekundi predstavlja minimalno vrijeme između dva seta instrukcija [13].

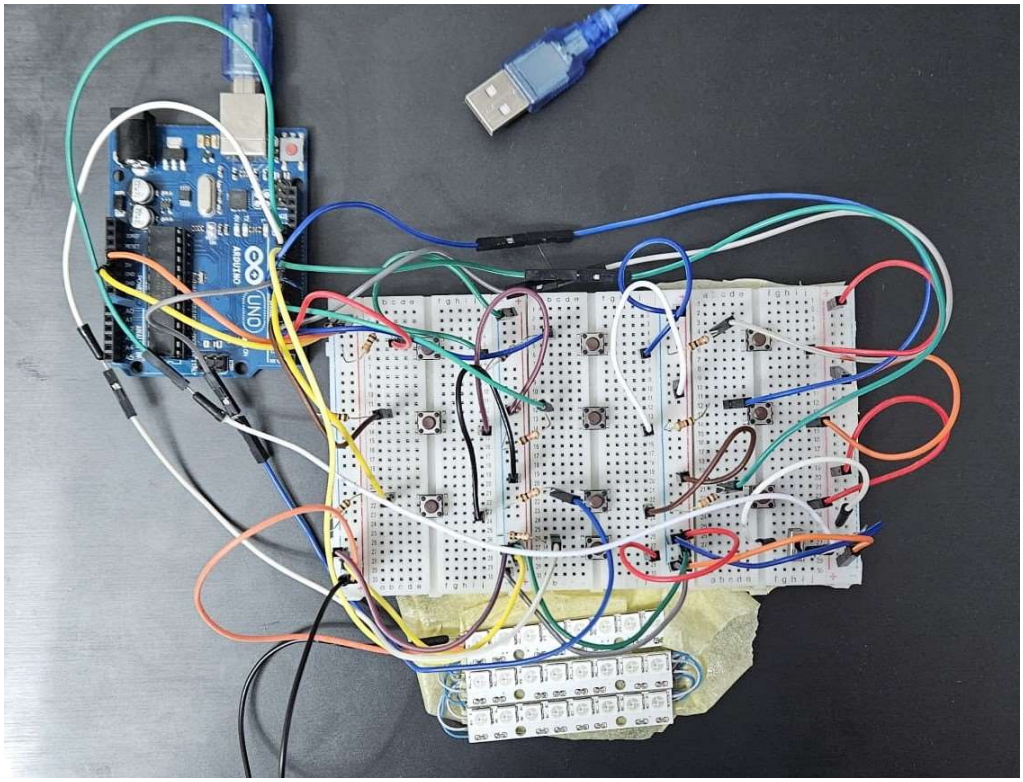



Slika 3.9. Vremenski dijagram NZR komunikacijskog protokola.

Slika 3.10. prikazuje dijagram spajanja Arduino UNO razvojne pločice sa svim komponentama potrebnim za izradu, dok slika 3.11 fizički izgled sklopa.

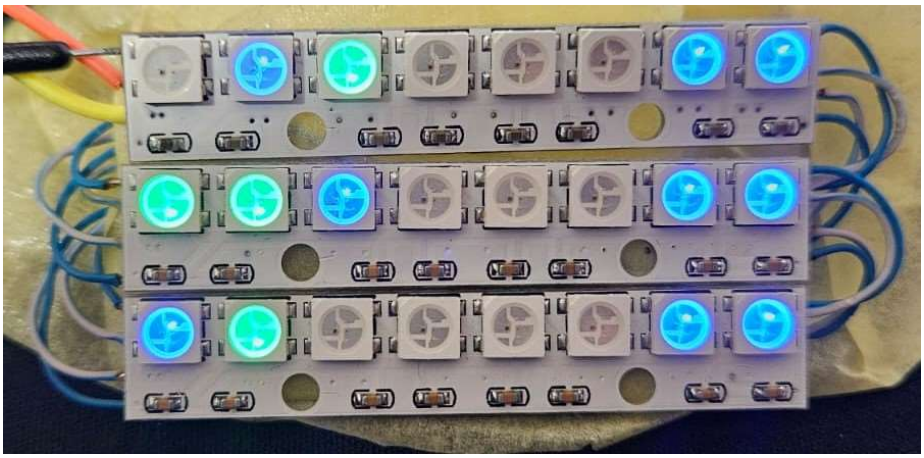


Slika 3.10. Shema cijelog spoja.



Slika 3.11. Fizička izvedba.

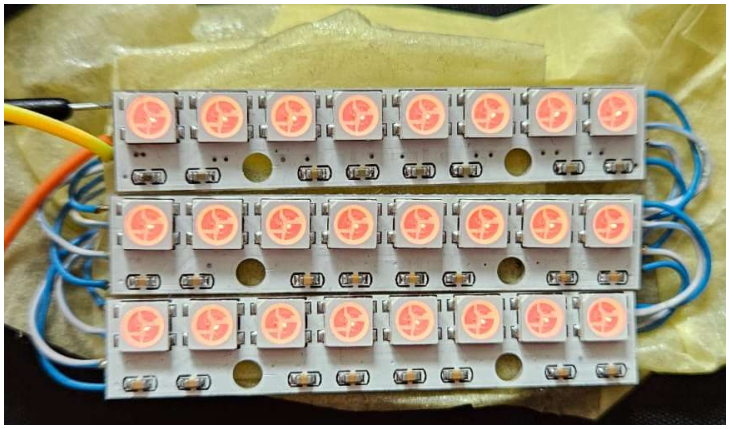
LED diode numerirane na slici 3.8. kao: 0-2, 8-10 i 16-18 korištene su za prikaz matrice stanja igre, dok su LED diode 6,7,14,15,22,23 korištene kao indikator koji je igrač na potezu. Kada prvi igrač pritisne tipkalo, odgovarajući element matrice poprimi vrijednost „1“, te se uključuje odgovarajuća LED dioda u zelenoj boji. Kada je drugi igrač na potezu, odgovarajući element matrice poprima vrijednost „2“ a odgovarajuća LED dioda se uključuje u plavoj boji, kao što je prikazano na slici 3.12.



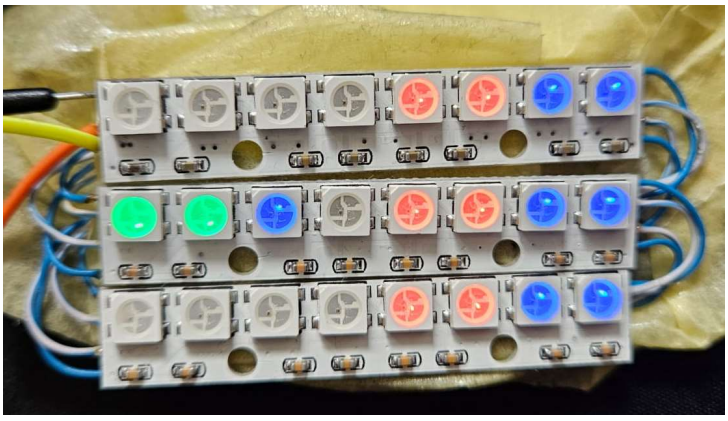
Slika 3.12. Indikacija odabranih polja prvog i drugog igrača u situaciji gdje je drugi igrač na potezu.

Kod pobjede određenog igrača sve LED diode trepere u pripadajućoj boji pobjednika što je prikazano na slici 3.13. za situaciju gdje je pobjednik igrač jedan. Ako niti jedan igrač ne pobjedi, tj. igra završi neriješeno, sve LED diode trepere u crvenoj boji kako je prikazano na slici 3.14. U slučaju kada igrač pokuša odigrati polje koje je je već zauzeto, igra se pauzira na dvije sekunde za vrijeme kojih LED diode 4-5, 12-13, 20-21 trepere crveno što je prikazano na slici 3.15.

Slika 3.13. Prikaz pobjede igrača jedan u zelenoj boji.



Slika 3.14. Prikaz kraja igre u slučaju neriješeno.



Slika 3.15. Prikaz greške prilikom odabira već popunjenog polja.

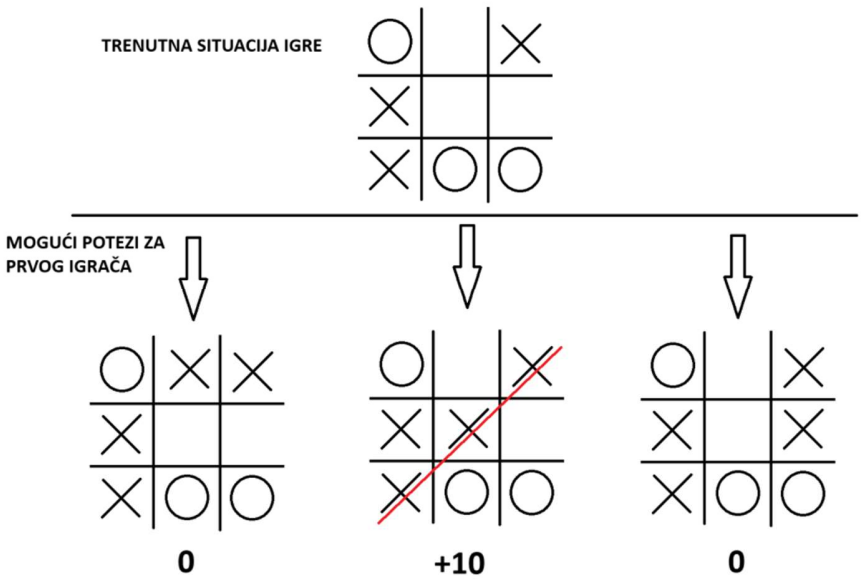
# 4. IGRA PROTIV RAČUNALA – MINIMAX FUNKCIJA

Za potrebu samostalne igre jednog igrača, potrebno je izraditi dio programa koji omogućuje igru protiv računala na način da računalo preuzme kontrolu drugog igrača. U takvom načinu rada zadan je uvjet da računalo, odnosno drugi igrač, ne može izgubiti igru. Navedeno je postignuto uz pomoć MiniMax funkcije i dodatne sklopke za odabir načina igre.

Kako bismo objasnili MiniMax funkciju, potrebno je prvo objasniti savršenog igrača kružića-križića. Savršeni igrač će ili pobijediti ili će odigrati izjednačeno protiv drugog igrača. Nadalje, ako se igra odvija između dva savršena igrača, igra će uvijek završiti neriješeno. Samim time možemo mogućim rezultatima kraja igre pridružiti prikladne bodovne vrijednosti [15]:

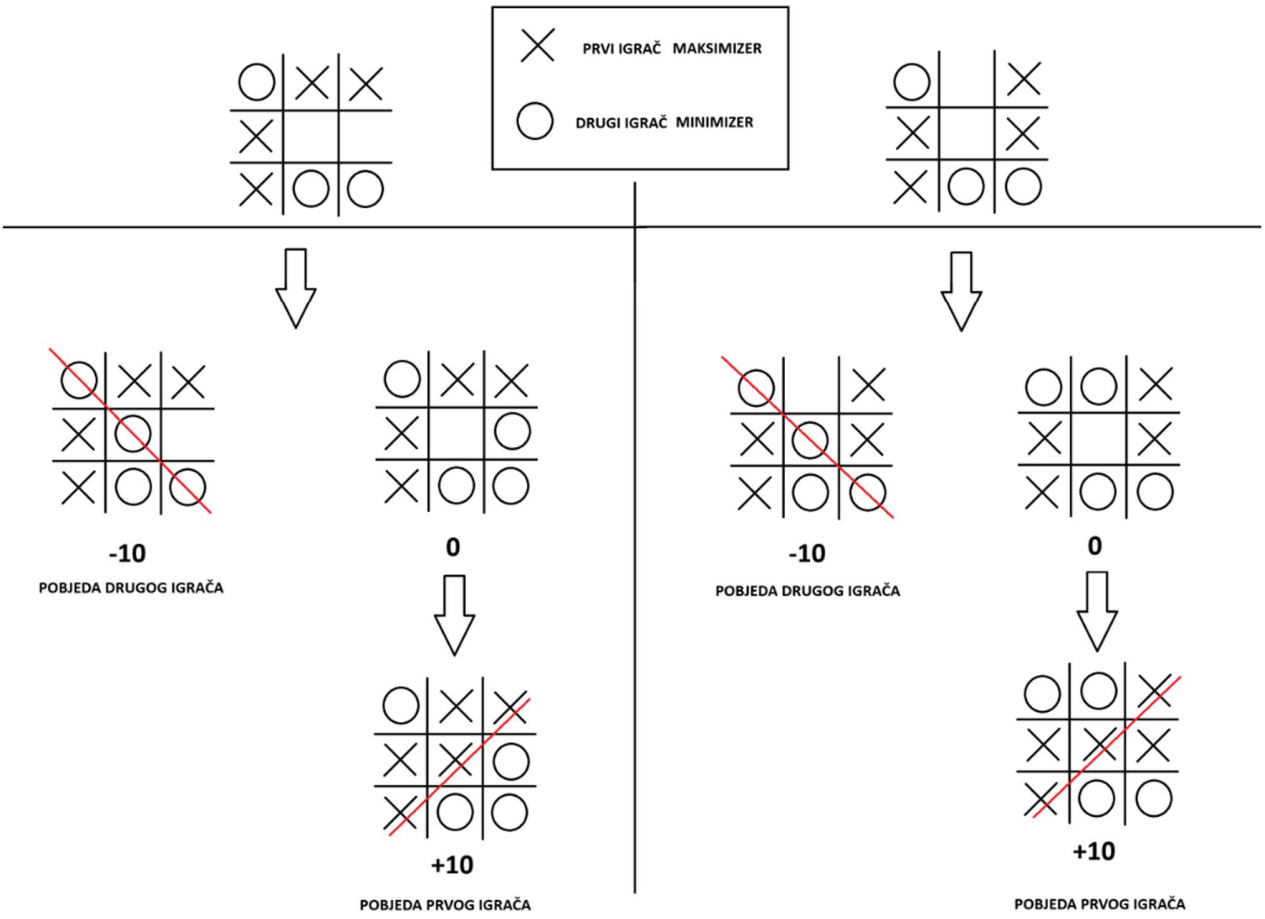
- Pobjeda: +10 bodova za prvog igrača,
- Gubitak: -10 bodova za prvog igrača odnosno +10 bodova za drugog igrača,
- Izjednačeno: 0 bodova; niti jedan igrač ne dobije bodove.

U ovom slučaju prvi igrač predstavlja maksimizera, dok drugi igrač predstavlja minimizera. Slika 4.1 prikazuje situaciju pred kraj igre gdje je na potezu prvi igrač te može odabrati jedno od tri moguća polja. Jedno od njih vodi do pobjede te se za to odabrano polje igraču jedan dodjeljuje +10 bodova. U tom slučaju prvi igrač maksimizira svoje bodove. Ako prvi igrač odabere krivo polje, drugi igrač može s lakoćom pobijediti tako da izabere srednje polje.




Slika 4.1. Prikaz mogućeg kraja igre.

Drugi igrač u ovom slučaju također igra da pobijedi u igri, ali u usporedbi s prvim igračem želi izabrati potez koji će rezultirati najgorim mogućim rezultatom za prvog igrača, dakle želi odabrati potez koji će minimizirati bodove prvog igrača. Slika 4.2 prikazuje logiku odabira drugog igrača, gdje će prvo izabrati onu poziciju za koju dodijelimo pobjedu drugom igraču odnosno vrijednost -10.

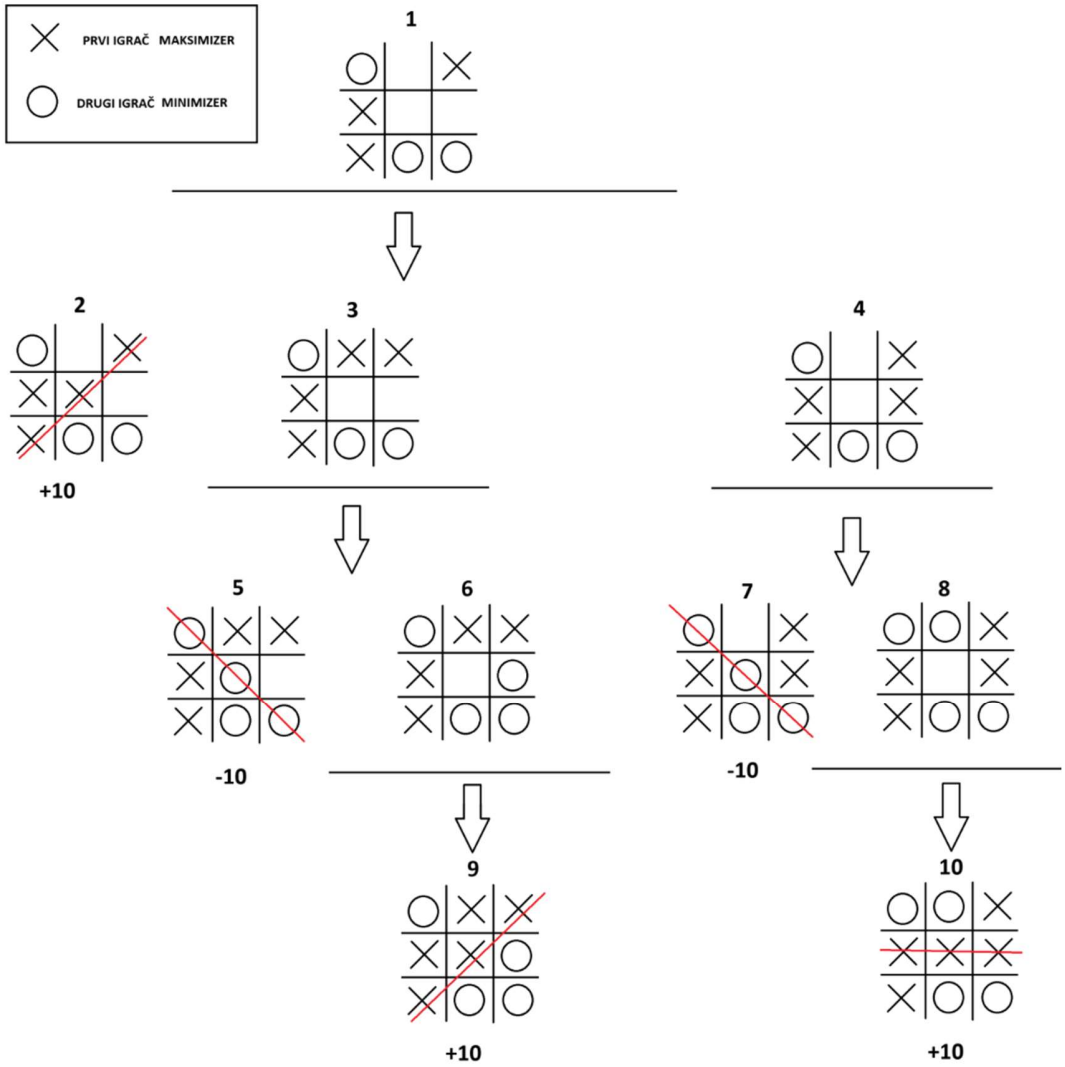



Slika 4.2. Prikaz mogućih krajeva igre za minimizera i maksimizera.

Smisao MiniMax algoritma leži u naizmjeničnoj igri prvog i drugog igrača, gdje igrač koji je na potezu odabire potez koji dovodi do maksimalnih bodova. Bodovi za svaki od dostupnih poteza su određeni od strane protivničkog igrača koji odlučuje koji od njegovih dostupnih poteza ima minimalni rezultat. S druge strane, bodovi za poteze protivničkog igrača određeni su od strane igrača na redu koji pokušava maksimizirati svoj rezultat i tako se generira stablo poteza sve do krajnjeg stanja.

Opis rada algoritma za slučaj u kojemu je križić prvi igrač, odnosno igrač koji radi potez, je sljedeći [15]:

1. Ako je igra gotova, vrati bodove iz perspektive križića
2. Inače, izradi popis novih stanja igre za sve moguće kombinacije poteza
3. Napravi popis rezultata:
   - Ako je križić na redu, vrati maksimalni rezultat s popisa
   - Ako je kružić na redu, vrati minimalni rezultat s popisa




Slika 4.3. Prikaz svih mogućih poteza kraja igre te njihove MiniMax vrijednosti.

Na slici 4.3 prikazano je sljedeće [15]:

- Na potezu je križić u stanju igre 1. Križić generira stanja 2, 3 i 4 te provodi MiniMax funkciju za svako stanje.
- Stanje 2 vraća +10 na listu bodova za stanje 1 pošto je igra završila.
- Stanja 3 i 4 nisu završetci igre, zbog toga stanje 3 generira stanja 5 i 6 i poziva MiniMax funkciju na njima, dok stanje 4 generira stanja 7 i 8 te također poziva MiniMax funkciju na njima.
- Stanje 5 vraća -10 u listu bodova za stanje 3, isto se radi i za stanje 7 koje vraća -10 na listu bodova za stanje 4.
- Stanje 6 i 8 generiraju jedine preostale poteze koji dovode do završetka s rezultatom +10 kojeg vraćaju u listu bodova za stanja 3 i 4.
- Budući da je kružić na potezu, stanja 3 i 4 će dobiti rezultat od -10, tj. minimalnu vrijednost svih stanja u koje se ona granaju.
- Na kraju se bodovna lista za stanja 2 popunjava s +10 bodova, dok se 3 i 4 popunjavaju s -10 bodova, te se odabire potez s maksimalnim brojem bodova, tj. stanje 2 s +10 bodova.

U svrhu jednostavnosti, u ovom načinu igre korisnik uvijek igra kao prvi igrač, a računalo kao drugi. Zbog toga, dobijemo ukupno manji broj mogućih grana po kojima se igra može odigrati, tj. grana koje računalo mora evaluirati. Broj mogućih grana $N$ možemo izraziti kao:

$$N = \sum_{i=0}^{d} b^i, \tag{4.1}$$

gdje $d$ predstavlja broj ukupnih budućih poteza, odnosno broj ukupnih slobodnih polja, dok $b$ prosječan broj mogućih poteza. Pošto ljudski igrač odabire već prvo polje, računalo ima preostalih osam ($d = 8$) slobodnih polja na odabir i prosječno četiri poteza ($b = 4$), pa uvrštavanje u (4.1) dobivamo $N = 87381$ različitih grana. Kada bi računalo igralo prvo, broj mogućih grana bi se povećao na $N = 2441406$ [14].

Tokom testiranja programa primijećeno je da je mikrokontroleru potrebno otprilike jedna minuta za evaluaciju prvog poteza budući da ima relativno spori radni takt (16 MHz), te uglavnom nisu predviđeni za ovakvu vrstu evaluacije. Zbog toga, promijenjen je dio koda kako bi računalo uvijek odigralo jedno od dva mogućih poteza kao prvi potez, što je značajno smanjilo vrijeme čekanja. Samim time evaluacije poteza mikrokontrolera tek počinju kada su popunjena tri polja matrice, što smanjuje ukupni broj potrebnih evaluacija na $N = 1093$ prema (4.1).

# 5. OPIS PROGRAMSKOG KODA

## 5.1 Definiranje varijabli

Na početku programa uključene su biblioteke, te su definirane sve globalne varijable koje će se koristiti [16]. Biblioteka `FastLED.h` uvodi nekoliko korisnih funkcija za upravljanje RGB LED diodama korištenih u ovom završnom radu. Početni dio koda:

```
////////////////////////////////////////////////////////////////////////
#include "FastLED.h"
#define NUM_LEDS 24
CRGB leds[NUM_LEDS];

int DI_pin[3][3]={ {1,2,3}, {4,5,6}, {7,8,9} }, DI_pin_START=11, DI_pin_MODE=12; //
vrijednosti matrice = stvarni broj pina
bool DI_Tipka[3][3]={ {false,false,false}, {false,false,false}, {false,false,false}
}, DI_Tipka_START=false;
bool M_Tipka_1[3][3]={ {false,false,false}, {false,false,false},
{false,false,false} }, M_Tipka_START_1=false;
bool M_Tipka_fp[3][3]={ {false,false,false}, {false,false,false},
{false,false,false} }, M_Tipka_START_fp=false;
bool M_Tipka[3][3]={ {false,false,false}, {false,false,false}, {false,false,false}
}, M_Tipka_START=false;

int matrica_igre[3][3]={ {0,0,0}, {0,0,0}, {0,0,0} };
int mod_igre=0; // 0= igra nije u tijeku, 1= covjek protiv covjeka, 2= covjek
protiv pc-a
int igrac=0; // 0= igra nije u tijeku, 1= P1 igra zeleni, 2= P2 igra plavi, 3=
AI/PC
bool igra_u_tijeku=false; // false= igra nije u tijeku, true= igra je u tijeku
int pom_tmp=0, poc_tmp=0, pom_broj=0, pom_AI1=0;

struct Move { int row, col; };
int player = 2, opponent = 1;
////////////////////////////////////////////////////////////////////////
```

Prvo je definirana varijabla `DI_pin`, matrica koja sadrži brojeve pinova na koja su spojena tipkala za selekciju igraćeg polja, kao i varijable `DI_pin_START` i `DI_pin_MODE` koji sadrže brojeve pinova na koje su spojeni tipkalo za start/reset i sklopka za odabir načina igre.

Varijabla `DI_Tipka` služi kao matrica detekcije pritiska na tipkalo. Varijabla `M_Tipka_1`, služi za pamćenje prošlog ciklusa programa, dok je u varijablu `M_Tipka` pohranjeno trenutno stanje tipkala. Varijabla `M_Tipka_fp` služi za detekciju pozitivnog brida tipkala. Sve navedene matrice su tipa `bool`, što znači da vrijednosti u njima jedino mogu biti istina ili laž.

Stanje igre spremljeno je u 3x3 matrici pod nazivom `matrica_igre`. U nju se upisuju vrijednosti nula, jedan ili dva koje prikladno označavaju prazno polje te prvog ili drugog igrača.

Varijabla `mod_igre` služi za detekciju položaja sklopke, odnosno za odabir načina igre. Varijabla `igrac` se koristi za detekciju igrača koji je na potezu, gdje nula označava igru koja nije u tijeku, jedan označava potez prvog igrača, dok dva označuje potez drugog igrača. Detekcija statusa igre da li je igra u tijeku ili nije vrši se pomoću varijable `igra_u_tijeku`. Varijable s početkom `pom` ili `poc` služe kao pomoćne varijable. Struktura `Move`, sadrži dva elementa `row` i `col`, te se koristi za zapisivanje poteza u načinu igre protiv računala.

Za definiranje RGB LED dioda koristi se `CRGB  leds[NUM_LEDS]`, gdje je CRGB definicija varijable `leds` koja stavlja svih 24 LED dioda u vektor.

### 5.2 Funkcija void setup()

Kod Arduino programiranja, prvo se poziva `void setup()` funkcija koja se jednom izvrši prilikom pokretanja ili resetiranja mikrokontrolera. Prema tome, ova funkcija je prikladna za definiranje ulazno/izlaznih pinova i sličnih inicijalizacija.

Funkcija `FastLED` dio je biblioteke `FastLED.h` koja služi za definiranje modela programibilnih LED dioda (WS2812), komunikacijskog pina (10), te ukupni broj kaskadno spojenih LED dioda (`NUM_LEDS`).

```
////////////////////////////////////////////////////////////////////////
void setup() {
  //Serial.begin(9600);
  FastLED.addLeds<WS2812,10>(leds,NUM_LEDS);
  FastLED.setBrightness(2);
  for (int i=0; i<3; i++) {
    for (int j=0; j<3; j++){
     pinMode(DI_pin[i][j], INPUT);
    }
  }
```

```
  pinMode(DI_pin_START, INPUT);
  pinMode(DI_pin_MODE, INPUT);}
////////////////////////////////////////////////////////////////////////
```

Zatim, unutar for petlje, pinovi na koji su spojeni tipkala za odabir igraćeg polja su definirani kao ulazni pinovi pomoću funkcije `pinMode`, a zatim izvan petlje i preostala dva pina na koje su spojeni reset tipkalo i sklopka za odabir načina igre.

### 5.3 Funkcija void loop()

Funkcija `void loop()` se izvodi ciklički, te je ona glavna funkcija svakog Arduino programa iz koje se pozivaju sve ostale funkcije.

Za početak provjerava se pomoćna varijabla `poc_tmp`, kada ima vrijednost 0 poziva se pomoćna funkcija `pocetak` u kojoj se uključuju sve LED diode i postavlja se vrijednost pomoćne varijable u 1. Zatim se poziva funkcija `citanje_start` unutar koje se provjerava da li je pritisnuto tipkalo start, te ukoliko je, pokreće se igra. Ako se ponovno pritisne tipkalo start, igra se prekida i poziva se funkcija `kraj_igre`. Ako je igra u tijeku, poziva se funkcija `citanje_tipki`, zatim s duplom for petljom provjerava se pritisnuto tipkalo, stanje pripadne matrice igre i igrač koji je na potezu. Ako je pripadno polje matrice popunjeno, signalizira se greška, te se poziva funkcija `error`. A ako je pripadno polje matrice prazno, upisuje se u matricu igre vrijednost igrača koji je na potezu, poziva se funkcija `provjera`, te se indicira pomoću funkcije `igra_P1` ili `igra_P2` potez igrača. Ako je igra u tijeku poziva se funkcija `pretvroba_za_led`. Ako je rezultat funkcije `provjera` vratio varijablu `pom_temp` s vrijednošću jedan (došlo je do pobjede), poziva se funkcija `kraj_igre`. Ako su sva polja popunjena i nema pobjednika, također se poziva ista funkcija.

Kada je varijabla `mod_igre`=2, igra je u tijeku i igra igrač dva te je `pom_AI1` jednak nuli, odigra se jedan od dva najbolja poteza koji su sredina (polje 1,1) ili gornji lijevi ugao (polje 0,0), te se vrijednost varijable `pom_AI1` postavlja u jedan. Kao što je prethodno rečeno, to je napravljeno da se izbjegne nepotrebno računanje najboljeg poteza u startu igre, dok za sve preostale poteze (kada je `pom_AI1`=1), stanje igre se preslikava u matricu `board`, te se poziva funkcija `findBestMove`, koja zatim simulira pritisak tipkala drugog igrača postavljam odgovarajuće vrijednosti u varijablu `M_Tipka`.

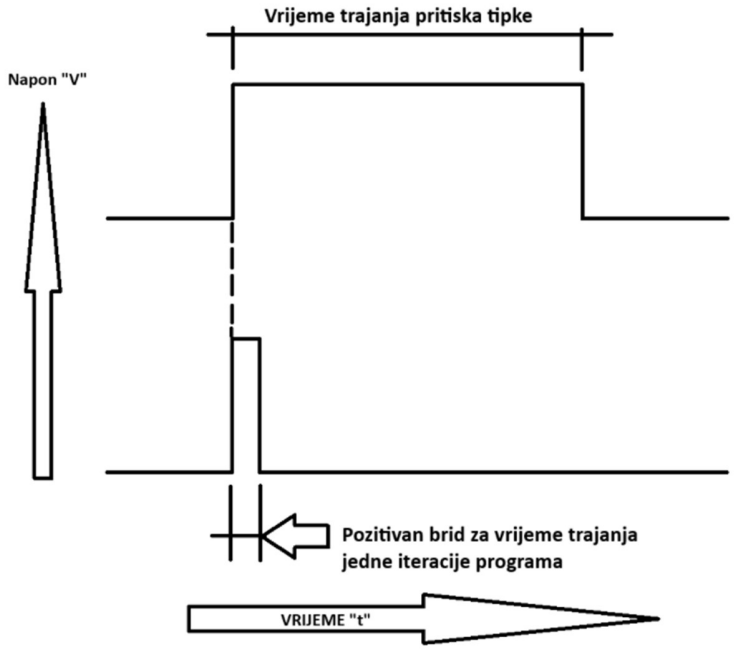
```
void loop() {
  if (poc_tmp==0){pocetak();}
  citanje_start();
  if (M_Tipka_START== true &&!igra_u_tijeku){
    igra_u_tijeku = true;
    M_Tipka_START=0;
    for (int i=0; i<24; i++){leds[i]=CRGB::Black;}
    igrac=1;
    igra_P1();
  }
  if (M_Tipka_START== true && igra_u_tijeku){
    igra_u_tijeku = false;
    M_Tipka_START=0;
    kraj_igre();
  }
  if (igra_u_tijeku == true){citanje_tipki();}
  for (int i=0; i<3; i++){
    for (int j=0; j<3; j++){
      if (M_Tipka[i][j]==true && matrica_igre[i][j]!=0 && (igrac==1||igrac==2) ){
        M_Tipka[i][j]=false;
        error();
      }
      if (M_Tipka[i][j]==true && matrica_igre[i][j]==0 && igrac==1){
        matrica_igre[i][j]=1;
        M_Tipka[i][j]=false;
        provjera();
        if(pom_tmp==0)  {igrac=2;igra_P2();}
      }
      if (M_Tipka[i][j]==true && matrica_igre[i][j]==0 && igrac==2){
        matrica_igre[i][j]=2;
        M_Tipka[i][j]=false;
        provjera();
        if(pom_tmp==0)  {igrac=1;igra_P1();}
      }
    }
  }
  if (igra_u_tijeku== true){pretvorba_za_led();}
  if (pom_tmp==1 && igra_u_tijeku==1){kraj_igre();}
  if (pom_broj==9 && igra_u_tijeku==1){kraj_igre();}

  if (igra_u_tijeku==true && igrac==2 && mod_igre==2 && pom_AI1!=0){
    int board[3][3];
    for(int i=0; i<3; i++){for(int j=0; j<3; j++){board[i][j]=matrica_igre[i][j];}}
    Move bestMove = findBestMove(board);
    delay(500);
    M_Tipka[bestMove.row][bestMove.col] = true;
  }
```

```
    if (igra_u_tijeku==true && igrac==2 && mod_igre==2 && pom_AI1==0){  // za prvi
korak AI stavi u čošak ili sredinu
      pom_AI1=1;
      if (matrica_igre[1][1]==0){
        M_Tipka[1][1]=true;
      }
      else {M_Tipka[0][0]=true;}
      delay(500);
    }
  }
```

### 5.4 Funkcija void citanje_start()

Funkcija `void citanje_start()` služi za detekciju pozitivnog brida prilikom pritiska na tipkalo za start/reset igre. Ako se tipkalo pritisne, odnosno ako je detektiran napon na ulaznom pinu, pomoćna varijabla `DI_Tipka_START` postavlja se u istinito stanje. Kako bi se postigla detekcija samo jednog pritiska, korištena je logika pozitivnog brida koja uzima vrijednost pritisnute tipke samo za prvu iteraciju programa od kada je tipka pritisnuta kako je prikazano na slici 5.1.




Slika 5.1 Prikaz pozitivnog brida u odnosu s vremenom pritiska tipke

To je postignuto tako da se uspoređuje trenutna vrijednost pritisnute tipke s vrijednosti varijable iz prošle iteracije programa. Varijabli `M_Tipka_START_FP` se dodjeljuje rezultat logičke operacije I između trenutnog stanja tipke te komplementa varijable `M_Tipka_START_1` u kojoj je zapisano stanje tipkala u prethodnoj iteraciji programa.

Dodatno se provjerava stanje sklopke za detekciju načina igre koji se zatim zapisuje u varijablu `mod igre`, gdje vrijednost 1 označava način u kojem igra igrač protiv računala, dok vrijednost 2 označava način u kojem igraju dva igrača.

```
////////////////////////////////////////////////////////////////////////
void citanje_start() {
  if (digitalRead(DI_pin_START)==HIGH) {DI_Tipka_START=true;}
  else {DI_Tipka_START=false;}
  M_Tipka_START_fp=DI_Tipka_START&&!M_Tipka_START_1;
  M_Tipka_START_1=DI_Tipka_START;
  if (M_Tipka_START_fp==true) {M_Tipka_START=!M_Tipka_START;}
  if (digitalRead(DI_pin_MODE)==HIGH) {mod_igre = 2;}
  else {mod_igre = 1;}
}
////////////////////////////////////////////////////////////////////////
```

### 5.5 Funkcija void citanje_tipki()

Funkcija `void citanje_tipki()` radi na isti način kao i prethodno opisana funkcija `void citanje_start()` te služi za detekciju pozitivnog brida pritisnutih tipkala za selekciju polja.

```
////////////////////////////////////////////////////////////////////////
void citanje_tipki() {
  for (int i=0; i<3; i++) {
    for (int j=0; j<3; j++){
      if (digitalRead(DI_pin[i][j])==HIGH) {DI_Tipka[i][j]=true;}
      else {DI_Tipka[i][j]=false;}
      M_Tipka_fp[i][j]=DI_Tipka[i][j]&&!M_Tipka_1[i][j];
      M_Tipka_1[i][j]=DI_Tipka[i][j];
      if (M_Tipka_fp[i][j]==true) {M_Tipka[i][j]=!M_Tipka[i][j];}
    }
  }
}
////////////////////////////////////////////////////////////////////////
```

### 5.6 Funkcija void pocetak()

Kada je igra u stanju mirovanja prilikom pokretanja Arduina, nakon reseta ili kraja igre, funkcija `void pocetak()` uključuje sve LED diode tako da svijetle u bijeloj boji. Koristeći dvije ugniježdene for petlje, funkcija prolazi kroz sve LED diode, a funkcijom `FastLED.show()` dajemo instrukcije da LED diode svijetle u bijeloj boji. Osim indikacije da igra nije u tijeku, ovom funkcijom možemo provjeriti i ispravnost svih LED dioda.

```
////////////////////////////////////////////////////////////////////////
void pocetak() {
  for(int i=0; i<3; i++){
    for(int j=0; j<3; j++){matrica_igre[i][j]=0;}
  }
  for (int i=0; i<24; i++) {leds[i]=CRGB::White;}
  FastLED.show();
  poc_tmp=1;}
////////////////////////////////////////////////////////////////////////
```

### 5.7 Funkcija void error()

Svrha funkcije `void error()` je dati povratnu informaciju igraču prilikom odabira polja koje je već bilo popunjeno . To je napravljeno na način da LED diode enumerirane kao 4,5,12,13,20 i 21 trepere u crvenoj boji. Unutar funkcije u matricu `leds` prvo su upisane vrijednosti koje odgovaraju crvenoj boji, zatim se poziva funkcija `FastLED.show()` koja uključi LED diode. Nakon toga slijedi pauza od 1.5 s, te na sličan način LED diode se isključuju.

```
////////////////////////////////////////////////////////////////////////
void error() {
  leds[4]=CRGB(0,255,0);  leds[5]=CRGB(0,255,0);
  leds[12]=CRGB(0,255,0); leds[13]=CRGB(0,255,0);
  leds[20]=CRGB(0,255,0); leds[21]=CRGB(0,255,0);
  FastLED.show();
  delay(1500);
  leds[4]=CRGB(0,0,0);  leds[5]=CRGB(0,0,0);
  leds[12]=CRGB(0,0,0); leds[13]=CRGB(0,0,0);
  leds[20]=CRGB(0,0,0); leds[21]=CRGB(0,0,0);
  FastLED.show();}
////////////////////////////////////////////////////////////////////////
```

### 5.8 Funkcije void igra_P1() i void igra_P2()

Funkcije `void igra_P1()` i `void igra_P2()` služe za indikaciju koji je igrač na potezu. Kada je igrač jedan na potezu, LED diode svijetle zelenom bojom u poljima 6, 7,14 ,15, 22 i 23. Kada je igrač dva na potezu, ta ista polja svijetle plavom bojom.

```
////////////////////////////////////////////////////////////////////////
void igra_P1() {
  for (int x=6; x<8; x++)   {leds[x]=CRGB(255,0,0);}
  for (int x=14; x<16; x++) {leds[x]=CRGB(255,0,0);}
  for (int x=22; x<24; x++) {leds[x]=CRGB(255,0,0);}
  FastLED.show();
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
void igra_P2() {
  for (int y=6; y<8; y++)   {leds[y]=CRGB(0,0,255);}
  for (int y=14; y<16; y++) {leds[y]=CRGB(0,0,255);}
  for (int y=22; y<24; y++) {leds[y]=CRGB(0,0,255);}
  FastLED.show();
}
////////////////////////////////////////////////////////////////////////
```

### 5.9 Funkcija void pretvorba_za_led()

Funkcija `void pretvorba_za_led()` radi kako je opisana nazivom, tj. pretvara matricu igre u izlaze za LED diode.

```
////////////////////////////////////////////////////////////////////////
void pretvorba_za_led() {
  for(int i=0; i<3; i++){
    for (int j=0; j<3; j++){
      if (matrica_igre[i][j] == 0) {leds[8*i+j]=CRGB(0,0,0);}
      if (matrica_igre[i][j] == 1) {leds[8*i+j]=CRGB(255,0,0);}
      if (matrica_igre[i][j] == 2) {leds[8*i+j]=CRGB(0,0,255);}}}
  FastLED.show(); }
////////////////////////////////////////////////////////////////////////
```

### 5.10 Funkcija void provjera()

Funkcija `void provjera()` služi za provjeru pobjednika igre. Prvo se provjerava da li u matrici stanja igre postoje retci u kojima je neki od igrača postigao pobjedu. Nakon toga se provjeravaju stupci, te obje dijagonale. U slučaju pronađene pobjede, globalna varijabla `pom_tmp` se postavlja u 1. U ovoj funkciji se također provjerava da li je igra završila neriješeno, na način da se u globalnu varijabli `pom_broj` zapiše broj popunjenih polja.

```
////////////////////////////////////////////////////////////////////////
void provjera() {
  pom_tmp=0;
  pom_broj=0;
  for (int i=0; i<3; i++){
    if (matrica_igre[i][0]>0 && matrica_igre[i][0]==matrica_igre[i][1] &&
matrica_igre[i][1]==matrica_igre[i][2]){pom_tmp=1;}
  }
  for (int i=0; i<3; i++){
    if (matrica_igre[0][i]>0 && matrica_igre[0][i]==matrica_igre[1][i] &&
matrica_igre[1][i]==matrica_igre[2][i]){pom_tmp=1;}
  }
  if (matrica_igre[0][0]>0 && matrica_igre[0][0]==matrica_igre[1][1] &&
matrica_igre[1][1]==matrica_igre[2][2]){pom_tmp=1;}
  if (matrica_igre[2][0]>0 && matrica_igre[2][0]==matrica_igre[1][1] &&
matrica_igre[1][1]==matrica_igre[0][2]){pom_tmp=1;}
  for (int i=0; i<3; i++){
    for (int j=0; j<3; j++){
      if (matrica_igre[i][j] > 0){pom_broj=pom_broj+1;}
    }
  }}////////////////////////////////////////////////////////////////////////
```

### 5.11 Funkcija void kraj_igre()

Funkcija `void kraj_igre()` pokreće završetak igre. U slučaju pobjede jednog od dvaju igrača, sve LED diode trepere u pobjedničkoj boji tri i pol sekunde nakon kojih se matrica igre i sve ostale pomoćne varijable resetiraju te se program vraća u početno stanje. U slučaju neriješenog rezultata sve LED diode trepere crvenom bojom.

```
////////////////////////////////////////////////////////////////////////
void kraj_igre() {
  if(igrac==1 && pom_tmp==1){
    delay(1000);
    for(int j=0; j<3; j++){
      for (int i=0; i<24; i++){leds[i]=CRGB(0,0,0);}
      FastLED.show(); delay(400);
      for (int i=0; i<24; i++){leds[i]=CRGB(255,0,0);}
      FastLED.show(); delay(800);
    }
  }
  if(igrac==2 && pom_tmp==1){
    delay(1000);
    for(int j=0; j<3; j++){
      for (int i=0; i<24; i++){leds[i]=CRGB(0,0,0);}
      FastLED.show(); delay(400);
      for (int i=0; i<24; i++){leds[i]=CRGB(0,0,255);}
      FastLED.show(); delay(800);
    }}
  if(pom_broj==9 && pom_tmp==0){
    delay(1000);
    for(int j=0; j<3; j++){
      for (int i=0; i<24; i++){leds[i]=CRGB(0,0,0);}
      FastLED.show(); delay(500);
      for (int i=0; i<24; i++){leds[i]=CRGB(0,255,0);}
      FastLED.show(); delay(1000);
    }}
  for(int i=0; i<3; i++){
    for(int j=0; j<3; j++){matrica_igre[i][j]=0;}
  }
  igrac=0;
  igra_u_tijeku=false;
  pom_tmp=0; poc_tmp=0; pom_broj=0;
  M_Tipka_START=false;
  pom_AI1=0;}
////////////////////////////////////////////////////////////////////////
```

### 5.12 Funkcija Move findBestMove(int borard[3][3])

Funkcija `Move findBestMove(int board[3][3])` kao argument prima matricu stanja igre, a kao rezultat vraća strukturu `Move` u kojoj je sadržana informacija o najboljem mogućem potezu, zatim radi MiniMax provjeru za svaku slobodnu ćeliju (koja nije popunjena s potezom igrača). Unutar for petlje, generiraju se svi mogući potezi igrača koji je na potezu. Za svaki mogući potez

igrača poziva se funkcija `int MiniMax(int board[3][3], int depth, bool isMax)` koja vraća ocjenu tog poteza. Konačni potez kojeg funkcija vraća je onaj potez za kojeg je funkcija MiniMax vratila najveću ocjenu.

```
/////////////////////////////////////////////////////////////////////////
// Ovo vraća najbolji mogući potez za računalo
Move findBestMove(int board[3][3])
{
  int bestVal = -1000;
  Move bestMove;
  bestMove.row = -1;
  bestMove.col = -1;
  // Prolazak kroz sve ćelije, procjena MiniMax vrijednosti za sve prazne ćelije.
  // Na kraju vrati ćeliju s optimalnom vrijednošću
  for (int i = 0; i<3; i++){
    for (int j = 0; j<3; j++){
      if (board[i][j]==0){  // Provjera prazne ćelije
        board[i][j] = player; // Odigrani potez
        // Evaluacija za odigrani potez
        int moveVal = MiniMax(board, 0, false);
        board[i][j] = 0;// poništi potez
        // Ako je vrijednost trenutnog poteza veća od vrijednosti najboljeg poteza
onda se prepiše
        if (moveVal > bestVal){
          bestMove.row = i;
          bestMove.col = j;
          bestVal = moveVal;}}}}
  return bestMove;}
/////////////////////////////////////////////////////////////////////////
```

### 5.13 Funkcija int evaluate(int b[3][3])

Za prepoznavanje kraja igre u funkciji MiniMax koristimo funkciju `int evaluate(int b[3][3])` koja vrši provjeru pomoćne matrice igre po redcima, stupcima i dijagonalama i prepoznaje ako je igrač pobijedio ili je igra završila neriješeno. U slučaju kada je igrač pobijedio dodijeli potezu vrijednost +10, kada je izgubio -10 te za neriješeno 0.

```
////////////////////////////////////////////////////////////////////////////
int evaluate(int b[3][3]){
// Ova funkcija provjerava da li je koji igrač pobijedio
  // Provjera redaka
  for (int row = 0; row<3; row++){
    if (b[row][0]==b[row][1] && b[row][1]==b[row][2]){
      if (b[row][0]==player)
        return +10;
      else if (b[row][0]==opponent)
        return -10;
    }
  }
  // Provjera stupaca
  for (int col = 0; col<3; col++){
    if (b[0][col]==b[1][col] && b[1][col]==b[2][col]){
      if (b[0][col]==player)
        return +10;
      else if (b[0][col]==opponent)
        return -10;
    }
  }
  // Provjera dijagonala
  if (b[0][0]==b[1][1] && b[1][1]==b[2][2]){
    if (b[0][0]==player)
      return +10;
    else if (b[0][0]==opponent)
      return -10;
  }
  if (b[0][2]==b[1][1] && b[1][1]==b[2][0]){
    if (b[0][2]==player)
      return +10;
    else if (b[0][2]==opponent)
      return -10;
  }
  // Ako nitko nije pobijedio vrati 0
  return 0;
}
////////////////////////////////////////////////////////////////////////////
```

### 5.14 Funkcija bool isMovesLeft(int board[3][3])

Funkcija `bool isMovesLeft(int boatd [3][3])` provjerava ako je pomoćna matrica igre u potpunosti popunjena. Vraća varijablu tipa `bool`. Ako je ploča popunjena daje vrijednost istina, ako nije laž.

```
////////////////////////////////////////////////////////////////////////
bool isMovesLeft(int board[3][3]){
// Funkcija vrati vrijednost istina ako je preostalo poteza, ako nije vrati
vrijednost false
  for (int i = 0; i<3; i++)
    for (int j = 0; j<3; j++)
      if (board[i][j]==0)
        return true;
  return false;
}
////////////////////////////////////////////////////////////////////////
```

### 5.15 Funkcija int MiniMax(int board[3][3], int depth, bool isMax)

Unutar funkcije `int MiniMax(int board[3][3], int depth, bool isMax)` prvo se poziva funkcija `int evaluate(int board[3][3])` koja mogućem potezu dodjeljuje vrijednost +10 (za pobjedu), -10 (za poraz) ili 0 (ako potez nije rezultirao niti pobjedom niti porazom). Ovo je rekurzivna funkcija, gdje funkcija iz rekurzije izlazi kada dođe do pobjede ili gubitka igrača, ili su sva polja popunjena. U slučaju da igra još traje, generiraju se svi daljnji mogući potezi s rekurzivnim pozivom na novo stanje igre. U slučaju da se generiraju potezi za protivničkog igrača traži se putanja s najmanjom ocjenom, dok se za vlastite potezi traži se putanja s najboljom ocjenom.

```
////////////////////////////////////////////////////////////////////////
int MiniMax(int board[3][3], int depth, bool isMax)
{
// Minmax funkcija prolazi kroz sve moguće situacije polja i vraća najbolju odabir
polja
  int score = evaluate(board);
  // Pobjeda za maksimizera
  if (score == 10)
    return score;
  // Pobjeda za minimizera
  if (score == -10)
    return score;
  // Slučaj izjednačeno
```

```
  if (isMovesLeft(board)==false)
    return 0;
  // Ako je maksimizer na potezu:
  if (isMax){
    int best = -1000;
    for (int i = 0; i<3; i++){
      for (int j = 0; j<3; j++){
        if (board[i][j]==0){
          board[i][j] = player;
          // Pozivanje opet MiniMax funkcije za dobivanje optimalnog poteza
          best = max( best, MiniMax(board, depth+1, !isMax) );
          board[i][j] = 0;   // Obriše se zadnji potez
        }
      }
    }
    return best;
  }
  // Minimizer potez
  else {
    int best = 1000;
    for (int i = 0; i<3; i++){
      for (int j = 0; j<3; j++){
        if (board[i][j]==0){
          board[i][j] = opponent;
          // Pozivanje opet MiniMax funkcije za dobivanje minimale vrijednosti
          best = min(best, MiniMax(board, depth+1, !isMax));
          board[i][j] = 0;
        }
      }
    }
    return best;
  }
}
////////////////////////////////////////////////////////////////////////////
```

# 6. ZAKLJUČAK

Razvoj mikrokontrolera uvelike je utjecao na našu svakodnevicu, što se očituje s raznim automatiziranim sustavima koje možemo promatrati svuda oko nas. Prvi korak u svijet mikrokontrolera je olakšan razvojnim pločicama poput Arduino koje omogućavaju visoku fleksibilnost i primjenu za razvoj prototipova. Platforma pruža ogromnu podršku online zajednice, te je samim time lako pronaći besplatne upute za izradu raznih prototipovaa i korisnih uređaja. U ovom završnom radu realizirana je igra kružić-križić uz pomoć razvojne pločice Arduino UNO.

Povijesno gledano, igra kružić-križić se razvila iz igara poput "Mlina" i "Picarie", no trenutna je njena verzija njima vrlo slična. Kružić-križić nije kompleksna igra, što je omogućilo jednostavnu implementaciju uz pomoć Arduino platforme u obliku 3x3 matrice, čiji elementi predstavljaju polja igre, a zapisane vrijednosti predstavljaju dva igrača. Pobjednik je igrač koji popuni tri polja matrice s istim znakom u nizu. Indikacija za navedeno je izvedena pomoću programibilnih RGB LED dioda koje predstavljaju prvog igrača s zelenom bojom, a drugog igrača s plavom bojom.

Drugi način igre je igra protiv računala, gdje računalo preuzima ulogu drugog igrača te samostalno odabire poteze. Za potrebe igre protiv računala, implementirana je MiniMax funkcija pomoću koje se numerički ocjenjuju putanje svih mogućih kombinacija poteza, te u konačnici se odabire potez s najboljom postignutom ocjenom. Takvom implementacijom računalo ne može izgubiti, tj. najbolji mogući ishod za igrača je neriješeni rezultat.

# LITERATURA


[1] Phil Salmony: Top Microcontrollers for Embedded Systems, s interneta, https://octopart.com/pulse/p/top-microcontrollers-embedded-systems, 2. veljače 2024.

[2] Rowsum: What Can Stm32 Do That Other Mcus Can't?, s interneta, https://www.rowsum.com/stm32-unique-advantages-comparison/, 22, listopad 2023.

[3] ST: STM32F0 Series, s interneta, https://www.st.com/en/microcontrollers-microprocessors/stm32f0-series.html, 2024

[4] Electronics-Lab.com: top 10 popular microcontrollers among makers, s interneta, https://www.electronics-lab.com/top-10-popular-microcontrollers-among-makers/, 19. lipanj 2020.

[5] RaspBerryPi: s interneta, https://www.raspberrypi.org/, 2024

[6] Arduino: s interneta, https://www.arduino.cc/, 2024

[7] Jhon Nussey, Arduino For Dummie 2nd Edition, by John Wiley & Sons, Inc., Hoboken, New Jersey, 2018

[8] Alyssa J. Pasquale, Ph.D., Microcontrollers, College of DuPage, 2024

[9] STlife.augmented, STM32F0 datasheet, DS9773 Rev 5, 2021

[10] ATMEL, ATmega328P datasheet, 7810D-AVR-01/15, 2015

[11] Peter Baum, Tic-Tac-Toe, Computer Science Department Southern Illinois University Carbondale, Illinois, 1975

[12] Alyssa S. Choi , Tic-Tac-Toe , National Museum of Mathematics, New York, 2021

[13] Worldsemi, WS2812B Intelligent control LED integrated light source(datasheet). 2020

[14] George T. Heineman, Gary Pollice & StanleySelkow, Algorithms in a nutshell second edition, O'ReillyMedia,Inc., Kanada, 2008

[15] Jason Rober, Tic Tac Toe: Understanding the Minimax Algorithm, s interneta https://www.neverstopbuilding.com/blog/minimax, 2013

[16] W3Schools: C++ Tutorial, s interneta, https://www.w3schools.com/cpp/default.asp, 2024

# POPIS OZNAKA I KRATICA

| | |
|---|---|
| RGB | Crvena, zelena, plava |
| LED | Svjetleća dioda |
| CPU | Centralna procesorska jedinica |
| SRAM | Statična memorija sa slobodnim pristupom |
| CRC | Ciklička provjera redundancije |
| RTC | Sat stvarnog vremena |
| DMA | Izravan pristup memoriji |
| I2C | Inter-integrirani krug |
| SWD | Serijsko otklanjanje pogrešaka |
| EEPROM | Memorija samo za čitanje koju se može električno izbrisati i programirati |
| PWM | Modulacija širine impulsa |
| IOT | Internet stvari |
| OS | Operativni sustav |
| USB | Univerzalno serijsko sučelje |
| ISCP | Serijsko programiranje u sklopu |
| NZR | Protokol bez automatskog poretka na nultu razinu |

# SAŽETAK

U ovome završnom radu izrađena je igra kružić-križić pomoću Arduino UNO razvojne pločice. Ukratko su opisani mikrokontroleri, te su dane tehničke karaktiristike STM32F0 i ATMega328P mikrokontrolera, kao i dostupne platforme za upravljanje IOT sustavima. Opisana je funkcionalnost Arduino UNO razvojne pločice s ATMega328P mikrokontrolerom kao i njezin raspored ulazno/izlaznih pinova. Zatim je opisana povijest igre kružić-križić, te hardverski i programski opis igre i fizička izvedba cijelog spoja. Igrač dobiva povratnu informaciju o igri putem programibilnih RGB LED. Cijelim se sustavom upravlja pomoću deset tipkala spojenih u *pull-down* konfiguraciju, te sklopkom za odabir načina igre: protiv drugog igrača ili igre protiv računala u kojoj računalo ne može izgubiti. Izvedba drugog načina igre izvedena je pomoću MiniMax algoritma koji samostalno odabire optimalan potez za računalo. U konačnici je u Arduino IDE razvojnom sučelju opisan razvijeni programski kod.

**Ključne riječi:** mikrokontroler, ATMega328P, Arduino, Arduino IDE, kružić-križić, *NeoPixel*, MiniMax funkcija.

## ABSTRACT

In this bachelor's thesis a tic-tac-toe game has been developed using an Arduino UNO development board. Microcontrollers have been briefly described, technical characteristics of STM32F0 and ATMega328P microcontrollers are given, as well as available platforms for managing IOT systems. The functionality of the Arduino UNO development board with the ATMega328P microcontroller has been described, as well as its input/output pin layout. This is followed by giving the history of the tic-tac-toe game, and detailed description of the game hardware and software. The feedback to the player is provided through programmable RGB LEDs. The entire system is controlled using ten pushbuttons connected in a *pull-down* configuration, and a switch for selecting the game mode: against another player or against the. The second game mode is implemented using the MiniMax algorithm, which independently selects the optimal move for the computer. Finally, the program code developed in Arduino IDE has been described.



**Keywords:** microcontroller, ATMega328P, Arduino, Arduino IDE, tic-tac-toe, *NeoPixel*, MiniMax function.