Connect 4 igra za dva igrača temeljena na Arduino platformi

Girotto, Ivan

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij elektrotehnike

Završni rad

CONNECT 4 IGRA ZA DVA IGRAČA TEMELJENA NA ARDUINO PLATFORMI

Rijeka, rujan 2024.

Ivan Girotto

0069092693

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij elektrotehnike

Završni rad

CONNECT 4 IGRA ZA DVA IGRAČA TEMELJENA NA ARDUINO PLATFORMI

Mentor: doc.dr.sc. Ivan Volarić

Rijeka, rujan 2024.

Ivan Girotto

0069092693

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 05.03.2024.

Zavod: Zavod za automatiku i elektroniku
Predmet: Digitalna elektronika

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Ivan Girotto (0069092693)**
Studij: Sveučilišni prijediplomski studij elektrotehnike (1030)

Zadatak: **Connect 4 igra za dva igrača temeljena na Arduino platformi / Arduino based two-player Connect 4 game**

Opis zadatka:

U sklopu završnog rada potrebno je izraditi Connect 4 igru za dva igrača temeljenu na Arduino platformi. Status igre je potrebno prikazati na 8x8 zaslonu realiziranog pomoću tzv. NeoPixel-a, a ulaz se vrši pomoću devet tipkala spojenih u pull-down konfiguraciju: osam za odabir stupca, te jedan za resetiranje igre spojenih na digitalne ulaze mikrokontrolera. Potrebno je izraditi programsku podršku koja će omogućiti funkcionalnost Connect 4 igre za dva igrača, te kada jedan od igrača pobjedi, potrebno je to indicirati na zaslonu na prikladan način.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Mentor:
doc. dr. sc. Ivan Volarić

Predsjednik povjerenstva za završni ispit:
prof. dr. sc. Dubravko Franković

Izjava o samostalnoj izradbi rada

Sukladno članku 7. Stavku 1. Pravilnika o završnom radu, završnom ispitu i završetku sveučilišnih prijediplomskih studija Tehničkog fakulteta Sveučilišta u Rijeci od 4. travnja 2023., izjavljujem da sam samostalno izradio završni rad prema zadatku preuzetom dana 20. ožujka 2024.

Rijeka, 10.09.2024.

_____

Ivan Girotto

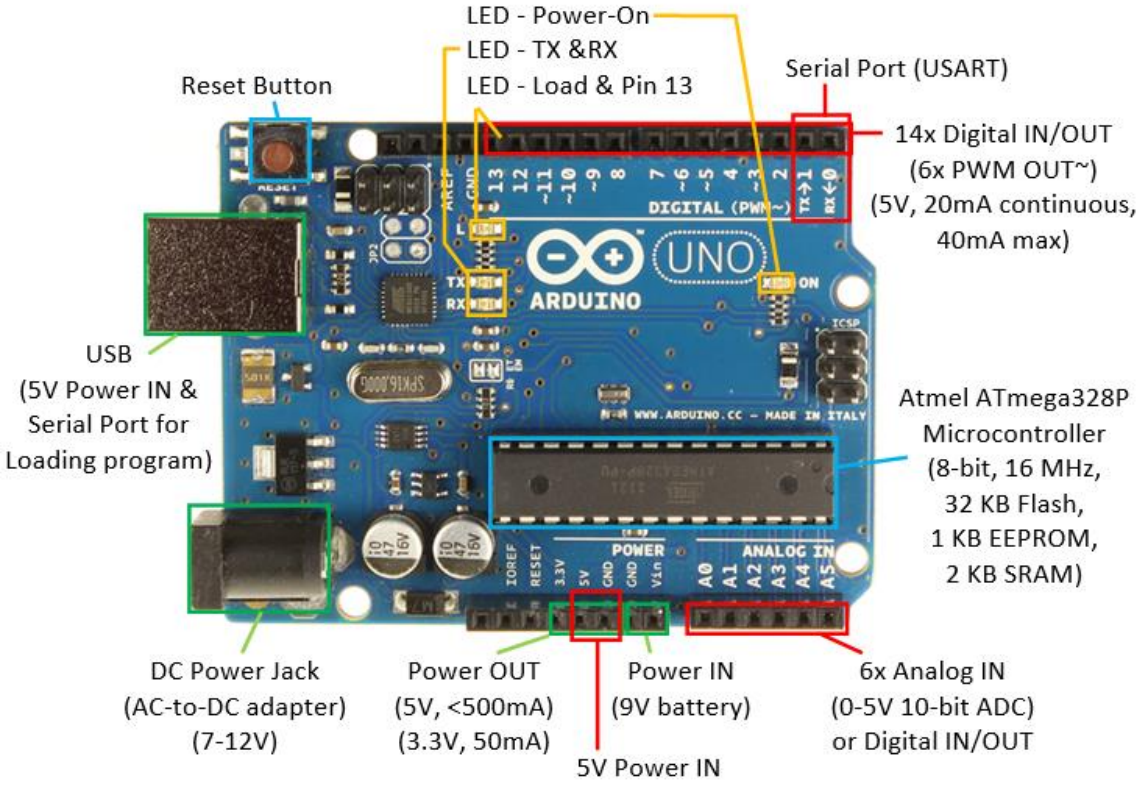
Sadržaj

1. UVOD

Cilj ovog završnog rada je implementirati digitalnu verziju Connect 4 igre koja će korisnicima omogućiti natjecanje i uživanje u igri na jednostavan i interaktivan način. Igra je implementirana pomoću Arduino Uno razvojne pločice, dok je za prikaz igre korištena 8x8 matrica programibilnih RGB LED dioda, tzv. NeoPixel-a. Upravljanje je izvedeno preko devet tipkala spojenih u *pull-down* konfiguraciju, od kojih je osam tipkala namijenjeno za odabir stupca u koji igrač želi postaviti žeton, dok deveto tipkalo služi za resetiranje igre. Osim osnovnih pravila igre, programskom podrškom su osigurane različite animacije koje poboljšavaju iskustvo igre.

2. ARDUINO UNO

Arduino Uno je razvojna pločica, prikazana na slici 2.1., koja se temelji na jednostavnom hardveru otvorenog koda koji olakšava izradu elektroničkih projekata. Pločica je opremljena setovima digitalnih i analognih ulazno/izlaznih (I/O) pinova te besplatnom programskom podrškom.



Slika 2.1. Arduino Uno razvojna pločica s istaknutim elementima.

Arduino platforma koristi razne mikrokontrolere, najčešće iz Atmel AVR linije kao što su ATmega328 i ATmega2560. Na Arduino *web*-stranicama može se preuzeti Arduino integrirano razvojno sučelje (eng. IDE – *Integrated Development Environment*), prikazano na slici 2.2., pomoću kojeg se programira mikrokontroler korištenjem varijacije programskog jezika C/C++ s dodatnim funkcijama i bibliotekama koje omogućuju jednostavno upravljanje funkcijama mikrokontrolera. Ovo uključuje funkcije za upravljanje digitalnim i analognim pinovima, komunikaciju putem serijskog porta, podršku za PWM (engl. *Pulse Width Modulation*), itd.

Slika 2.2. Arduino IDE razvojno sučelje.

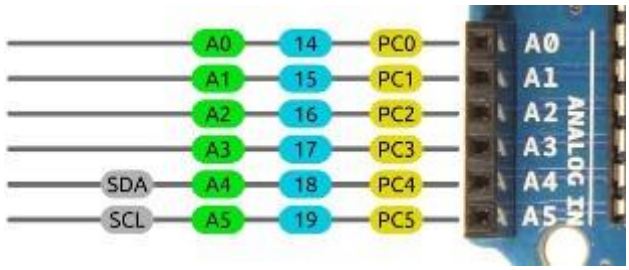
Arduino IDE pruža korisničko sučelje za pisanje, uređivanje i kompiliranje programa. Također omogućuje prenošenje programa na Arduino razvojnu pločicu preko USB veze. Iako je Arduino IDE popularan alat za programiranje Arduino razvojnih pločica, moguće je koristiti i druga razvojna sučelja poput PlatformIO i Arduino Eclipse IDE.

Tehničke specifikacije Arduino Uno razvojne pločice su:

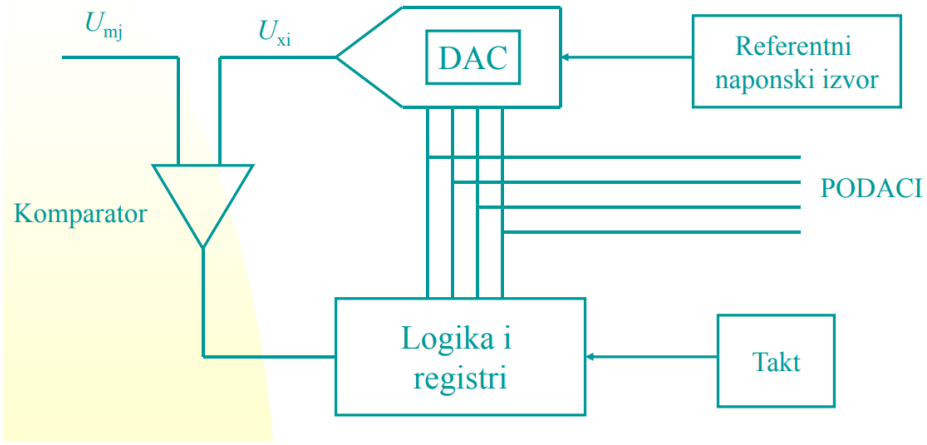
- Mikrokontroler: ATmega328P
- Napon napajanja mikrokontrolera: 5 V
- Frekvencija radnog takta mikrokontrolera: 16 MHz
- Ulazni napon: 6-20 V, preporučeno 7-12 V
- Digitalni I/O pinovi: od 1 do 13
- Digitalni pinovi s PWM podrškom: 3, 5, 6, 9, 10, 11
- Analogni I/O pinovi: od A0 do A5
- Maksimalna struja na I/O pinovima: 20 mA
- Flash memorija za pohranu programa: 32 KB od čega 0.5 KB koristi bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Duljina: 68.6 mm
- Širina: 53.4 mm
- Težina: 25 g
- Serijska komunikacija: USB priključak, tip B

2.1. Analogni pinovi

Analogni pinovi na Arduino Uno razvojnoj pločici su označeni sa: A0, A1, A2, A3, A4 i A5 kako je prikazano na slici 2.3. Ovi pinovi su samo ulazni, te se uobičajeno koriste za mjerenje napona koji je izlaz iz nekog senzora. Unutar mikrokontrolera, ti su pinovi spojeni na 10-bitni analogno-digitalni pretvornik (ADC) sa sukcesivnom aproksimacijom (SAR). Blokovski dijagram SAR ADC-a prikazan je na slici 2.4.

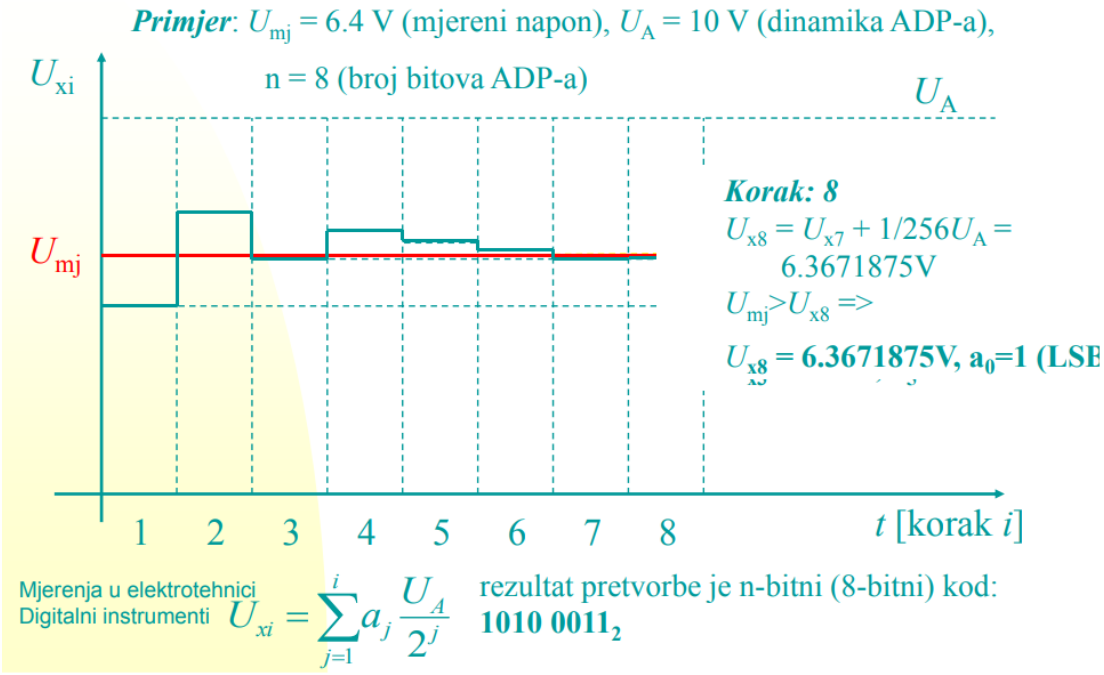


Slika 2.3. Analogni pinovi Arduino Uno razvojne pločice.



Slika 2.4. Blokovski dijagram SAR ADC-a.

Sukcesivna aproksimacija je metoda kojom se uspoređuje ulazni napon s referentnim naponom te se postupno poboljšava točnost digitalne reprezentacije ulaznog napona. Proces aproksimacije započinje postavljanjem bita najveće težine (MSB, engl. *Most Significant Bit*) digitalne vrijednosti u 1 (što za referentni napon od 5 V predstavlja napon od 2.5 V) i na temelju rezultata usporedbe s ulaznim naponom postavlja bit registra na vrijednost 1 ili 0. Ako je ulazni napon veći od referentnog, bit se postavlja na 1, dok u slučaju da je ulazni napon manji od referentnog, bit se postavlja na 0. Proces se ponavlja za svaki sljedeći bit, odnosno 10 puta kod 10-bitnog ADC-a. Primjer takve sukcesivne aproksimacije prikazan je na slici 2.5.



Slika 2.5. Primjer 8-bitne sukcesivne aproksimacije.

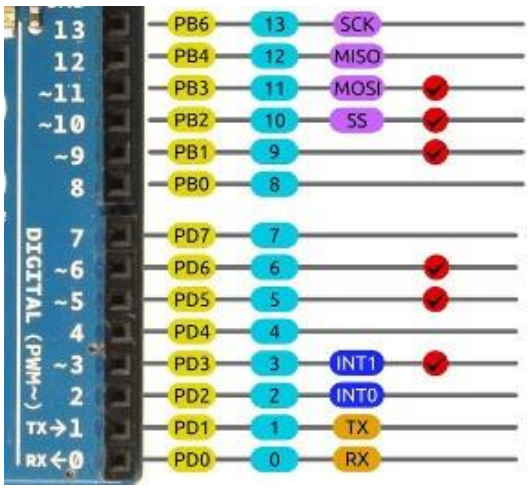
Kvantizacijski korak, Q, tj. najmanja promjena ulaznog napona koju ADC može detektirati glasi:

$$Q = \frac{U_{ref}}{2^n - 1}, \tag{2.1.}$$

gdje je U_{ref} najveći (referentni) napon analogno-digitalnog pretvornika, dok je n broj bitova.

2.2. Digitalni pinovi

Kod digitalnih pinova, Arduino razvojna pločica ulaz interpretira kao 1 kada je napon veći od 2.7 V, a u suprotnom kao 0. Pločica sadrži 14 digitalnih pinova označeni brojevima od 0 do 13 te su prikazani na slici 2.6.



Slika. 2.6. Digitalni pinovi Arduino Uno razvojne pločice.

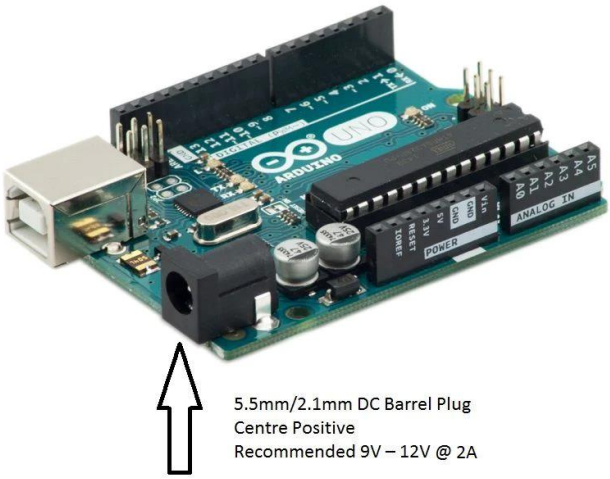
Ove pinove je moguće postaviti kao ulaz ili izlaz korištenjem funkcije `pinMode(broj pina, način rada)` najčešće u `void setup()` funkciji kako je prikazano u nastavku.

```
void setup() {
  pinMode(11, OUTPUT);     //postavljanje digitalnog pina 11 kao izlaz
  pinMode(13, INPUT);      //postavljanje digitalnog pina 13 kao ulaz
}

void loop() {
  digitalWrite(11, HIGH); //postavljanje digitalnog pina 11 na 5V (izvor)
  digitalWrite(11, LOW);//postavljanje digitalnog pina 11 na 0V (GND)

  digitalRead(13);         //očitanje stanja na 13-om pinu
}
```

Kada je pin konfiguriran kao ulazni pomoću `pinMode()` funkcije, napon na pinu se može očitati pomoću `digitalRead()` funkcije. Funkcija `digitalRead(broj pina)` očitava stanje pina i vraća rezultat kao visoka (engl. *high*) ili niska (engl. *low*) vrijednost. S druge strane, ako je pin konfiguriran kao izlazni njegov napon se postavlja preko `digitalWrite(broj pina, HIGH/LOW)` funkcije na 5 V za visoku vrijednost i 0 V za nisku vrijednost.

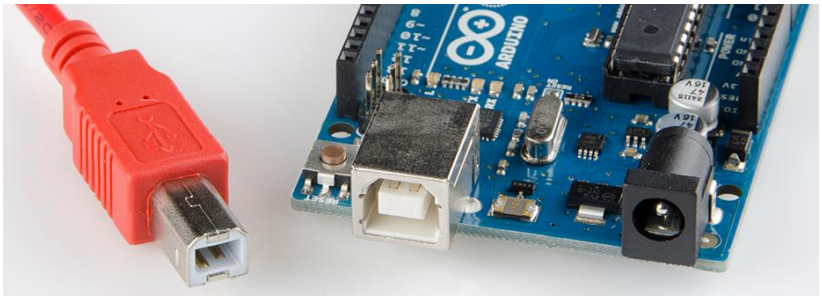
2.3. Pinovi za napajanje

Konektor za napajanje (engl. *barrel-jack*), prikazan na slici 2.7., služi kao sučelje za napajanje Arduino razvojne pločice istosmjernim naponom. Raspon ulaznog napona na ovom konektoru je 7-12 V.



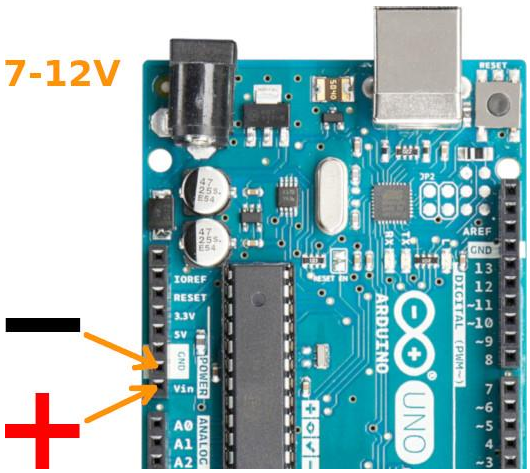
Slika 2.7. Konektor za napajanje Arduino Uno razvojne pločice.

USB priključak**,** prikazan na slici 2.8, je komunikacijski i napajajući kabel za povezivanje Arduino razvojne pločice s računalom ili drugim izvorom napajanja. Pruža stabilno napajanje od 5 V s maksimalnom strujom od 500 mA, dok istovremeno prenosi podatke između računala i Arduina.



Slika 2.8. USB priključak Arduino Uno razvojne pločice.

Pin V_{in}, prikazan na slici 2.9., je spojen paralelno s prethodno opisanim konektorom za napajanje, pa prethodno navedena ograničenja vrijede za ovaj pin. Regulator napona na razvojnoj pločici stabilizira napon na 5 V, koji se zatim koristi za napajanje mikrokontrolera i drugih komponenti.



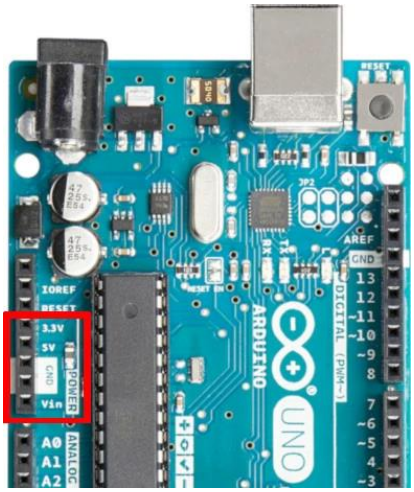
Slika 2.9. Pin V_{in} Arduino Uno razvojne pločice.

Pinovi 3.3V, 5V i GND su prikazani na slici 2.10. i oni označavaju:

- Pin 3.3V daje konstantan napon od 3.3 V te se može koristiti za napajanje komponenti koje zahtijevaju taj iznos napona. Maksimalni iznos struje koju ovaj pin može dati ovisi o mogućnostima regulatora napona, te uobičajeno iznosi 150 mA.

- Pin 5V osigurava konstantan istosmjerni napon za napajanje vanjskih komponenti od 5 V. Maksimalni iznos struje koju ovaj pin može dati iznosi 500 mA. Također može služiti kao ulazni pin na kojeg se dovodi već prethodno stabiliziranih 5 V.

- Pin GND ili uzemljenje je točka na potencijalu od 0 V i služi za zatvaranje strujnog kruga. Arduino razvojna pločica ima 3 GND pina koji su svi međusobno povezani.

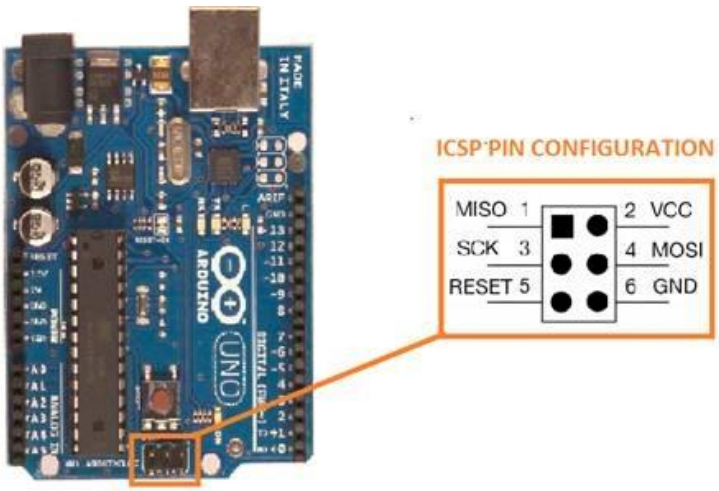


Slika 2.10. Pinovi za napajanje vanjskih komponenti Arduino Uno razvojne pločice.

2.4. ICSP konektor

ICSP (engl. *In-Circuit Serial Programming*) konektor, prikazan na slici 2.11., omogućuje programiranje mikrokontrolera kada mikrokontroler nema već prethodno isprogramirani *bootloader*.

Bootloader je mali program koji se već nalazi na mikrokontroleru na Arduino razvojnim pločicama, koji omogućuje programiranje mikrokontrolera pomoću serijskog UART (engl. *Universal Asynchronous Receiver-Transmitter*) protokola. Na Arduino razvojnim pločicama, USB konektor je spojen na integrirani krug koji pretvara USB na UART protokol, a koji je potom spojen na prva dva digitalna pina mikrokontrolera koji se uobičajeno koriste za Tx (engl. *transmit*) i Rx (engl. *receive*) serijske UART komunikacije.

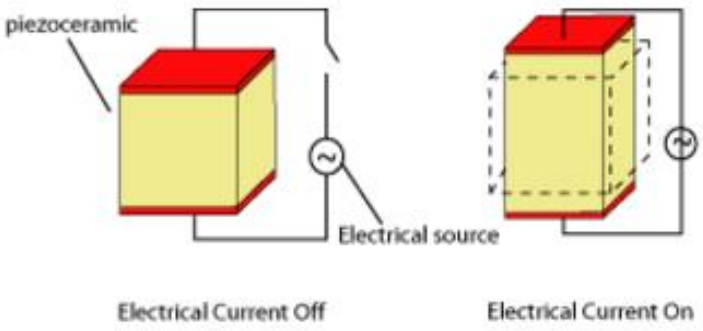


Slika 2.11. ICSP pinovi Arduino Uno razvojne pločice.

2.5. Kristalni oscilator

Kristalni oscilator je neophodna komponenta u elektroničkim sustavima koji zahtijevaju precizno vremensko upravljanje. Za mikrokontroler poput ATmega328P često se koristi 16 MHz kristalni oscilator, pomoću kojeg se generira radni takt iste frekvencije, te pruža vremensku referencu za izvršavanje instrukcija.

Kristalni oscilatori koriste kristal kao rezonantni element kako bi generirali signal stabilne i precizne frekvencije. Kristali su napravljeni od materijala koji imaju piezoelektrična svojstva, što znači da se deformiraju kad su izloženi električnom naponu. Dok obrnuto, generiraju električni napon kada su podvrgnuti mehaničkom stresu što je prikazano na slici 2.12. Kada se kristal stavi u električno polje i primijeni se odgovarajući napon, kristal vibrira na svojoj rezonantnoj frekvenciji.



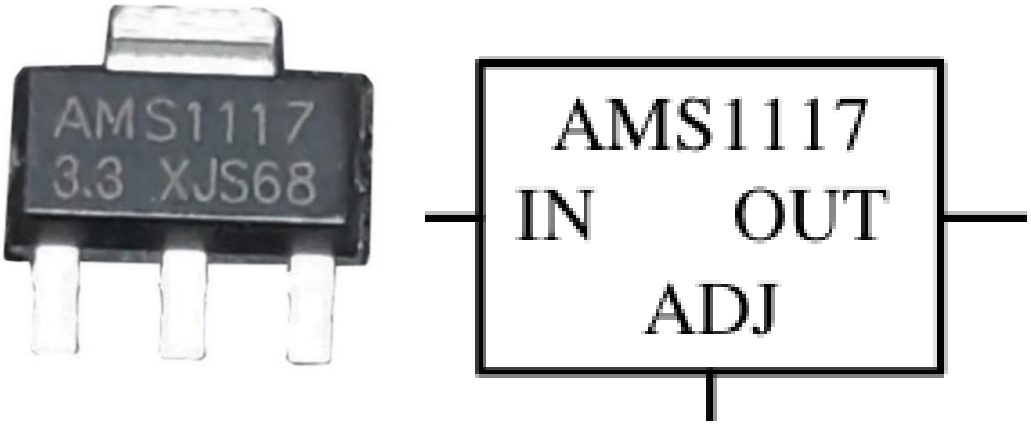
Slika 2.12. Piezoelektrični efekt.

Ovakvo svojstvo kristala omogućuje da rade kao oscilatori. Kad se kristal iskrivi, uzrokuje promjenu električnog polja unutar kristala što rezultira generiranjem električnog signala određene frekvencije. Ta frekvencija se zove rezonantna frekvencija kristala i određena je njegovim fizikalnim i geometrijskim svojstvima. Kristalni oscilatori koji se nalaze na Arduino Uno razvojnoj pločici prikazani su na slici 2.13.

Slika 2.13. Kristalni oscilatori.

2.6. Naponski regulator

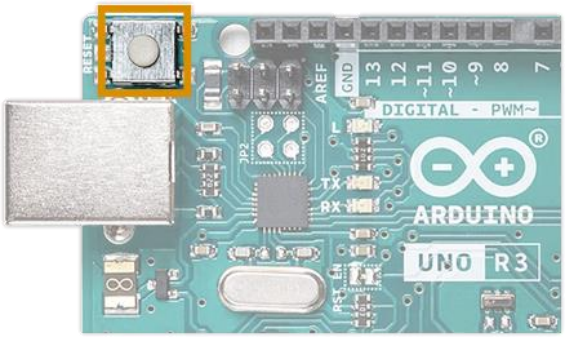
Arduino Uno razvojna pločica koristi dva naponska regulatora iz AMS1117 serije koji su jednostavni za korištenje i imaju zaštitu od kratkog spoja i toplinskog preopterećenja, tj. isključuju se ako temperatura regulatora prijeđe 165°C. Jedan regulator se koristi za stabilizaciju napona na 5 V za napajanje mikrokontrolera i ostalih vanjskih modula, dok se drugi, od 3.3 V koristi isključivo za napajanje vanjskih modula te je prikazan na slici 2.14.



Slika 2.14. Naponski regulator iz AMS1117 serije.

2.7. Reset tipkalo

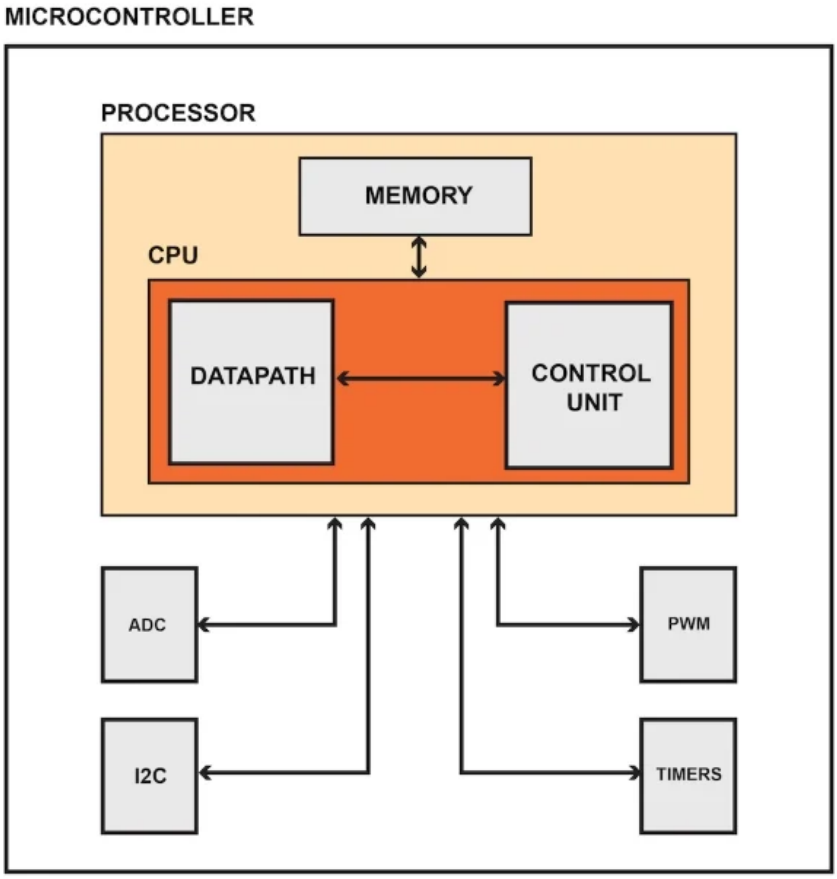
Tipkalo za *reset*, prikazano na slici 2.15., služi za ponovno pokretanje mikrokontrolera. Kada se pritisne tipkalo, Arduino Uno se ponovno pokreće i izvršava program od početka.



Slika 2.15. Reset tipkalo Arduino Uno razvojne pločice.

3. MIKROKONTROLER

Mikrokontroler, čija je tipična blokovska shema prikazana na slici 3.1., je integrirani krug (IC, engl. *Integrated Circuit*) koji se koristi za upravljanje drugim dijelovima elektroničkog sustava. Sastoji se od središnje procesorske jedinice (CPU, engl. *Central Processing Unit*), programske memorije (ROM, engl. *Read-Only Memory,* Flash), podatkovne memorije (RAM, engl. *Random Access Memory*), periferija i pomoćnih strujnih krugova.



Slika 3.1. Blokovska shema mikrokontrolera.

Atmel ATmega328P je CMOS 8-bitni mikrokontroler niske potrošnje, te se temelji na AVR poboljšanoj RISC arhitekturi. Ovaj mikrokontroler se nalazi na Arduino Uno razvojnoj pločici te je prikazan na slici 3.2.

Slika 3.2. Miktrokontroler ATmega328P.

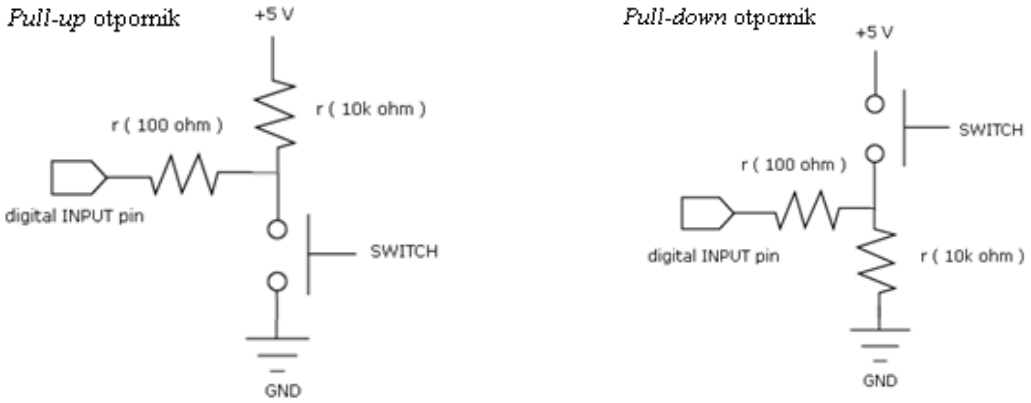
Računalo s reduciranim skupom instrukcija (RISC, engl. *Reduced Instruction Set Computer*) je računalna arhitektura koja koristi procesor s relativno malim brojem jednostavnih instrukcija. U usporedbi s instrukcijama danim računalu sa složenim skupom instrukcija (CISC, engl. *Complex Instruction Set Computer*), RISC računalo zahtjeva više strojnih instrukcija za izvršavanje istog zadatka jer procesoru nedostaju kompleksnije instrukcije. Time se postiže manja hardverska kompleksnost procesora što direktno utječe na njegovu cijenu. S druge strane, kompajler za takav procesor napisani kod u nekom višem programskom jeziku mora raščlaniti na veći broj pojedinačnih strojnih instrukcija.

4. PULL-UP I PULL-DOWN OTPORNICI

Pull-up i *pull-down* otpornici, prikazani na slici 4.1., su pasivne elektroničke komponente koje se često koriste u digitalnim elektroničkim sklopovima.

Pull-up otpornici su otpornici fiksne vrijednosti koji se koriste između izvora napajanja (Vcc) i određenog pina u digitalnom logičkom krugu. Njegova je svrha osigurati napon napajanja na izlazu kada je sklopka otvorena, izbjegavajući stanje visoke impedancije i krivog očitanja vrijednosti od strane mikrokontrolera.

Slično *pull-up* otpornicima, *pull-down* otpornici, osiguravaju aktivnu kontrolu napona na pinu mikrokontrolera kada je sklopka otvorena, osiguravajući napon od 0 V na izlazu.



Slika 4.1. Tipična aplikacija (a) pull-up, (b) pull-down otpornika.

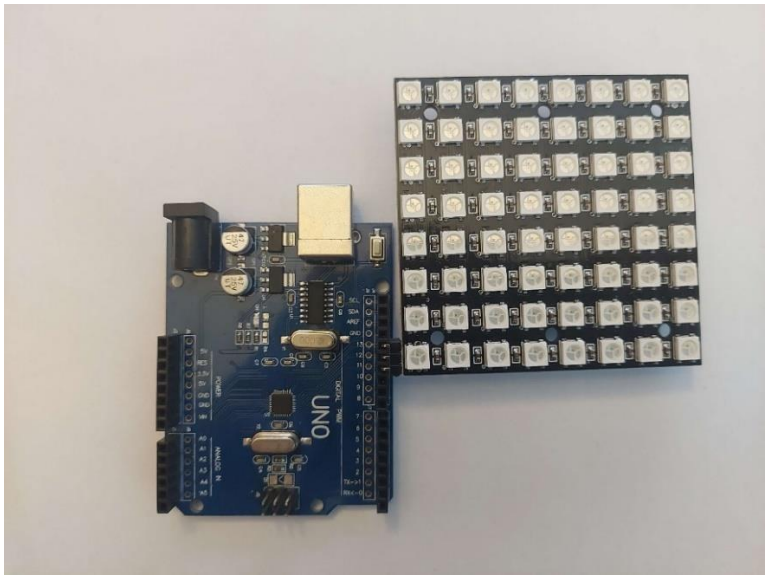
5. NEOPIXEL

U ovom završnom radu korišten je modul WS2812B-64 koji se sastoji od 64 programibilnih RGB LED dioda (poznatijih pod imenom NeoPixel), prikazan na slici 5.1. Ovakve LED diode omogućuju jednostavnu kontrolu boje i intenziteta svjetlosti pomoću jedne digitalne linije. Takvo ponašanje je moguće jer svaka LED dioda ima integrirani mikrokontroler koji upravlja LED diodom. Prva tri dobivena *byte*-a mikrokontroler koristi za upravljanje svojom LED diodom, dok preostale dobivene podatke prosljeđuje sljedećem mikrokontroleru omogućujući upravljanje velikim brojem kaskadno spojenih NeoPixel-a.

Slika 5.1. Adafruit NeoPixel modul WS2812B-64.

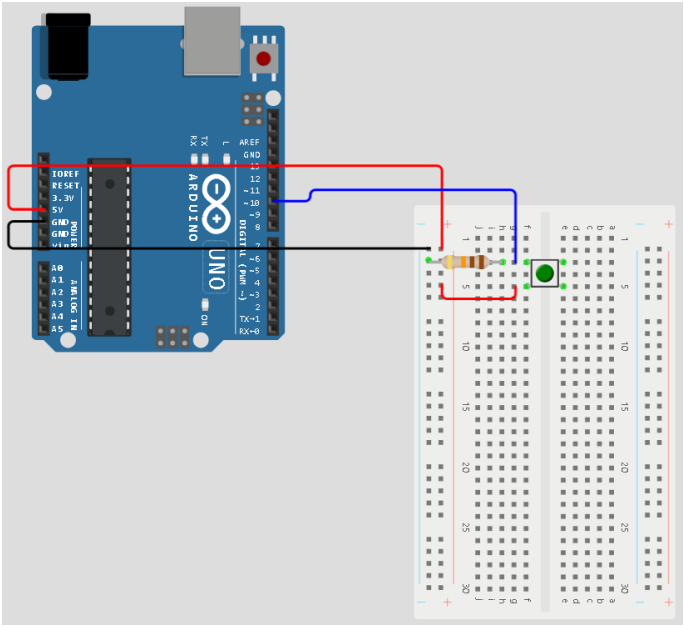
6. Spajanje svih komponenti završnog rada

NeoPixel sadrži 3 pina, V+, DIN i V-. Pin V+ se spaja na Arduino pin 11, pin DIN na Arduino pin 12 i pin V- na Arduino pin 13 kao što je prikazano na slici 6.1.



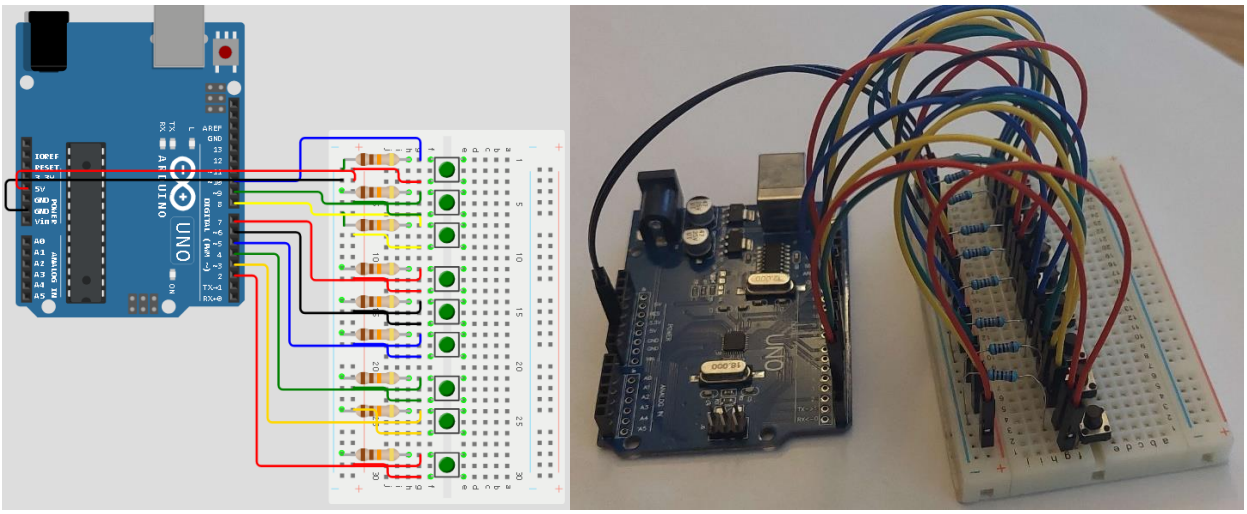
Slika 6.1. Spajanje NeoPixel matrice s Arduino Uno razvojnom pločicom.

Ulazna tipkala spojena su pomoću *pull-down* otpornika kako bi se osigurala kontrolirana vrijednost napona na digitalnim pinovima. Princip spajanja je prikazan na slici 6.2.

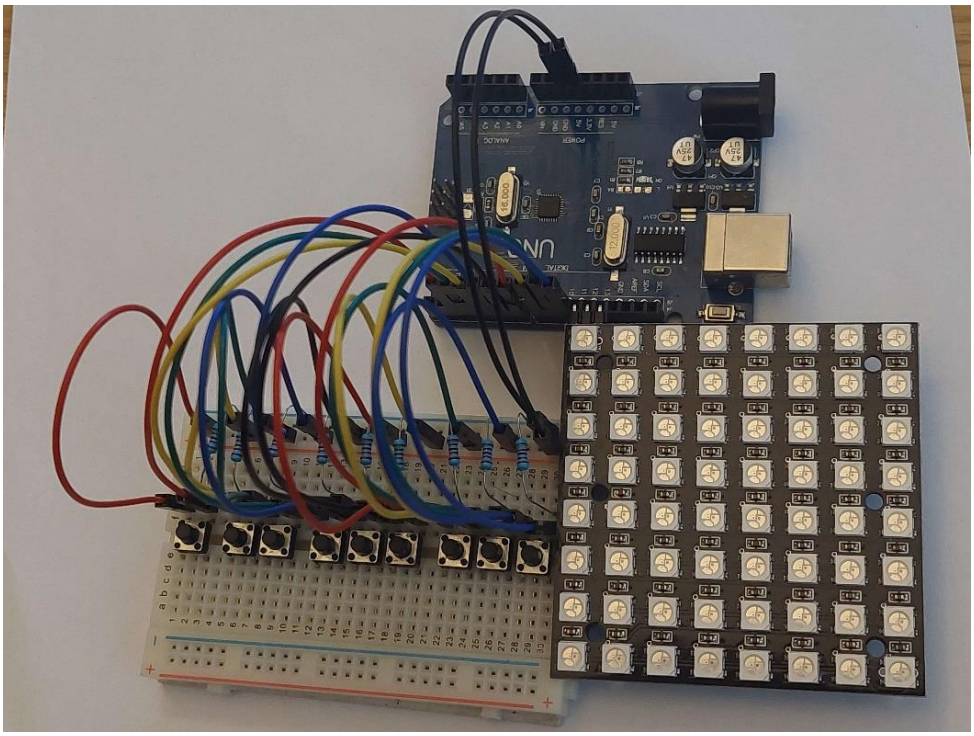


Slika 6.2. Spajanje tipkala u pull-down konfiguraciju.

Postupak spajanja prikazan na slici 6.2. je ponovljen za svako tipkalo kao što je prikazano na slici 6.3., dok je krajnji izgled hardverske konfiguracije prikazan na slici 6.4.



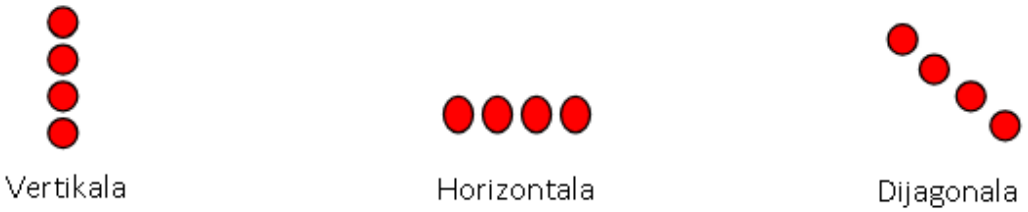
Slika 6.3. Spajanje ulaznih tipkala.



Slika 6.4. Hardverska konfiguracija igre.

7. CONNECT 4 IGRA NA ARDUINU

Connect 4 je klasična igra za dva igrača u kojoj je cilj biti prvi koji uspije napraviti liniju od četiri isto obojena žetona u okomitom, vodoravnom ili dijagonalnom smjeru na ploči za igru kao što je prikazano na slici 7.1. U sklopu ovog završnog rada, ova igra je implementirana na Arduino Uno razvojnoj pločici, gdje 8x8 LED matrica predstavlja igraću ploču na kojoj LED diode predstavljaju obojene žetone. Igrači naizmjenice pritiskom jednog od 8 tipkala odabiru stupac u koji će ispustiti svoj digitalni žeton. Taj žeton pada na dno, tj. na zadnje nepopunjeno mjesto tog stupca. Strategija ove igre je blokirati protivničke poteze i istovremeno stvarati prilike za spajanje vlastitih žetona za pobjedu.



Slika 7.1. Pobjedničke kombinacije.

7.1. Najvažnije naredbe i vrste varijabli

U Arduino programiranju postoje dvije osnovne funkcije: `void setup()` i `void loop()`.

`Void setup()` je funkcija koja se poziva jednom kada se Arduino Uno razvojna pločica uključi ili resetira. Ova funkcija obično služi za inicijalizaciju varijabli, postavljanje načina rada pinova te obavljanje drugih inicijalizacija neophodnih za ispravno funkcioniranje programa.

`Void loop()` je funkcija koja se neprestano ponavlja nakon što se funkcija `setup()` izvrši, te uobičajeno sadrži kod koji osigurava osnovnu funkcionalnost cijelog programa. Jednom kada se funkcija `loop()` izvrši, ponovno se poziva, osiguravajući kontinuirano izvođenje programa na mikrokontroleru.

U ovom završnom radu korišteni su sljedeći tipovi podataka:

Cjelobrojni tip podataka (`int`) - koristi se za predstavljanje cijelih brojeva s predznakom. Zauzima 4 *byte*-a memorijskog prostora, pa prema tome ima raspon od -32,768 do 32,767.

Logički tip podataka (`bool`) - ima dvije moguće vrijednosti: točno (engl. *true*) ili netočno (engl. *false*).

Striktno pozitivni cjelobrojni tip podataka (`unsigned long`) - koristi se za predstavljanje cijelih brojeva bez predznaka te u memoriji zauzima duplo više prostora od `int`-a, tj. 8 *byte*-ova. U ovom završnom radu ovaj tip varijable se koristi za pohranu vrijednosti povezane s vremenom, definirajući vremenske intervale za sve animacije i nesmetanu interakciju s korisnikom. Može predstaviti cijele brojeve od 0 do 4,294,867,296, što omogućuje pohranu vrijednosti do približno 50 dana u milisekundama.

Konstantni cjelobrojni tip podataka (`const int`) - kombinacija dvije ključne riječi: `const` i `int`. `Const` se koristi za deklariranje varijabli čije se vrijednosti ne mogu mijenjati nakon što su inicijalizirane, odnosno korisne su za definiranje vrijednosti koje ostaju iste tijekom izvođenja programa.

Neke od češće korištenih funkcija u ovom završnom radu:

`PinMode(broj pina, način rada)` - funkcija koja se koristi za postavljanje načina rada određenog pina na mikrokontroleru. Određuje hoće li se pin koristiti kao ulaz (engl. *INPUT*) ili izlaz (engl. *OUTPUT*).

`DigitalWrite(broj pina, vrijednost)` - funkcija koja uključuje pin koji je prethodno deklariran u `pinMode()` funkciji kao *OUTPUT* na *HIGH* (5 V) ili *LOW* (GND, odnosno 0 V).

`DigitalRead(broj pina)` - funkcija koja čita digitalni signal na određenom pinu koji je prethodno definiran u `pinMode()` kao *INPUT*.

`Strip.begin()` - funkcija koju je potrebno pozvati u `setup()` funkciji kako bi se pokrenula komunikacija s NeoPixel-om.

`Strip.show()` - funkcija koja se koristi za ažuriranje NeoPixel LED matrice.

`Strip.setBrightness(vrijednost)` - funkcija za postavljanje intenziteta svjetlosti NeoPixel LED matrice. Raspon vrijednosti intenziteta je od 0 do 255.

7.2. Analiza programskog koda

U sljedećem isječku koda su deklarirane globalne varijable koje definiraju sve važne parametre igre. Na početku je pozvana `Adafruit_NeoPixel.h` biblioteka koja olakšava upravljanje NeoPixel matricom. Uz to potrebno je i definirati parametre NeoPixel matrice. S funkcijom `Adafruit_NeoPixel(NEOPIXEL_NUM*NEOPIXEL_NUM, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);` definirane su sljedeće stavke:

- `NEOPIXEL_NUM*NEOPIXEL_NUM`: određuje ukupan broj RGB LED dioda u matrici.
- `NEOPIXEL_PIN`: specificira Arduino pin koji je spojen na komunikacijsku liniju NeoPixel-a.
- `NEO_GRB + NEO_KHZ800`: oznake koje određuju redoslijed boja (`NEO_GRB` definira redoslijed zelena, crvena pa plava) i frekvenciju komunikacije NeoPixel-a (`NEO_KHZ800`).

```
#include <Adafruit_NeoPixel.h> //biblioteka koja pojednostavljuje zadatke
poput postavljanja LED boja i definiranje intenziteta svjetlosti
#define NEOPIXEL_PIN 12           //podatkovni ulaz za NeoPixel matricu
#define NEOPIXEL_NUM 8            //broj NeoPixel-a u jednom retku/stupcu
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NEOPIXEL_NUM*NEOPIXEL_NUM,
NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
const int buttonPins[] = {3, 4, 5, 6, 7, 8, 9, 10}; // ulazna tipkala
const int resetPin = 2;           // gumb za resetiranje igre
int board[NEOPIXEL_NUM][NEOPIXEL_NUM] = {0}; // digitalna reprezentacija
NeoPixel matrice
int currentPlayer = 1;            // trenutni igrač
bool gameOver = false;            // stanje igre
unsigned long time;              // vrijeme od početka pokretanja programa
unsigned long droptime=0;       // vrijeme prethodnog pokretanja animacije
unsigned long animationStart=0;   // početak pokretanja animacije
unsigned long blinkdelay=0;       // vrijeme prethodnog kruga animacije
bool dropPieceAnimation = false;  // stanje animacije
bool skip=false;                  // stanje preskoka animacije
bool HorizontalWin=false;         // stanje horizontalne pobijede
bool VerticalWin=false;           // stanje vertikalne pobijede
bool DijagonalUpWin=false;        // stanje dijagonalne pobijede
bool DijagonalDownWin=false;      // stanje dijagonalne pobijede
bool resetGameAnimation=false;    // stanje resetiranja igre
bool secondAnimation=false;       // stanje animacije
int DropRow=0;                    // broj retka
int DropColumn=0;                 // broj stupca
int currentRow=0;                 // broj retka
int currentColumn=0;              // broj stupca
int extracolumn=0;                // dodatni broj stupaca
int extra=0;                      // dodatni broj stupaca/redova
int WinRow=0;                     // pobjednički redak
int WinColumn=0;                  // pobjednički stupac
int step=0;                       // koraci animacije
```

U sljedećem isječku koda definirana je `setup()` funkcija u kojoj su deklarirani pinovi za napajanje NeoPixel matrice, ulazni pinovi za tipkala, definiran je intenzitet svjetlosti matrice, te ulazni pin za tipkalo koje služi za resetiranje igre.

```
void setup() {
  pinMode(11, OUTPUT); // deklariranje pina kao izlaz
  pinMode(13, OUTPUT); // deklariranje pina kao izlaz
  digitalWrite(11, HIGH); // postavljanje pina 11 kao izvor +5V
  digitalWrite(13, LOW); // postavljanje pina 13 kao uzemljenje 0V
  strip.begin(); // pokretanje komunikacije s NeoPixel matricom
  strip.show(); // ažuriranje NeoPixel matrice
  strip.setBrightness(1); // podešavanje svjetline
```

```
  for (int i = 0; i < NEOPIXEL_NUM; i++) {
    pinMode(buttonPins[i], INPUT); // deklariranje tipkala kao ulaz
  }
  pinMode(resetPin, INPUT); // postavljanje reset tipkala kao ulaz

  for (int row = 0; row < NEOPIXEL_NUM; row++) {
    for (int col = 0; col < NEOPIXEL_NUM; col++) {
      board[row][col] = 0; // postavljanje matrice u početno stanje
     updateLEDMatrix(row, col, 0); // isključivanje svih LED dioda
    }}}
```

U sljedećem isječku koda definirana je `loop()` funkcija u kojoj se pozivaju funkcije koje osiguravaju funkcionalnost igre, te su detaljno opisane u nastavku poglavlja.

```
void loop() {
  time = millis(); // varijabla koja prati vrijeme
  checkForWin(currentPlayer); // funkcija za provjeru pobjede
  if (!gameOver) {
    Input(); // ulaz
  } if (digitalRead(resetPin) == HIGH) {
    resetGame(); // funkcija za resetiranje igre
  }
  Animations(); // funkcija koja sadrži animacije
  strip.show();
}
```

U sljedećem isječku koda definirana je `Input()` funkcija koja detektira pritisnuto tipkalo, provjeri ako izabrani stupac ima prazno mjesto, te ukoliko ima poziva se funkcija `dropPiece(int column)` i zabilježi vrijeme pritiska na tipkalo. To vrijeme se koristi za definiranje pauze prije mogućeg pokretanja preskoka animacije uzrokovane kada sljedeći igrač pritisne tipkalo za vrijeme trajanja animacije padanja žetona prethodnog igrača. Ako je novo tipkalo pritisnuto nakon male pauze od 300 ms animacija padanja prvog žetona se preskače i započinje se nova.

```
void Input() { // funkcija koja očitava ulaze
  for (int i = 0; i < NEOPIXEL_NUM; i++) {// detektiranje ulaznog signala
    if (digitalRead(buttonPins[i]) == HIGH) {
      if (board[0][i] == 0) { // provjera slobodnog retka matrice
        if(!dropPieceAnimation){
          animationStart=time;
          dropPiece(i);  //funkcija za određivanje koordinata žetona
        } if(dropPieceAnimation && time-animationStart >= 300){ // pauza
          skip=true;}
        return;
     }}}}
```

U sljedećem isječku koda definirana je `DropPiece(int column)` funkcija koja određuje najniži slobodni redak za ulazni stupac matrice te pokreće animaciju pada žetona, postavlja broj igrača na mjesto žetona i zapisuje konačne koordinate žetona u globalne varijable `DropRow` i `DropColumn`.

```
void dropPiece(int column){ // funkcija za određivanje koordinata žetona
  for (int row = NEOPIXEL_NUM-1; row >= 0; row--) {
    if (board[row][column] == 0) { // lociranje najnižeg retka u stupcu
      board[row][column] = currentPlayer; // postavljanje oznake igrača
      dropPieceAnimation=true; // pokretanje animacije za pad žetona
      DropRow=row; // bilježenje broja retka
      DropColumn=column; // bilježenje broja stupca
      break;
      }}}
```

U sljedećem isječku koda definiran je dio `Animations()` funkcije koji sadrži animaciju za pad žetona. Pri početku je postavljena pauza od 150 ms koja definira brzinu padanja žetona kroz retke. Kroz svaku iteraciju `loop()` funkcije žeton se spušta za jedno mjesto prema dolje, odnosno indeks retka raste od 0 do prvog slobodnog mjesta u matrici. Funkcija `updateLEDMatrix(int row, int col, int player)` određuje boju polja na matrici. Ako je novo tipkalo pritisnuto tokom animacije pada žetona, ta animacija se preskače, žeton se postavlja na odgovarajuće mjesto, te se pokreće nova animacija za novi žeton.

```
void Animations(){ // funkcija koja sadrži sve animacije
  if(dropPieceAnimation){
    currentColumn=DropColumn;
    if(time-droptime >= 150){ // pauza od 150 ms
    droptime=time;
  if(currentRow<=DropRow){ // spuštanje žetona na željeno mjesto
  updateLEDMatrix(currentRow, DropColumn, currentPlayer); // boja polja
  updateLEDMatrix(currentRow-1, DropColumn, 0);
  currentRow++;} // sljedeći red
      if(currentRow>DropRow){
      currentPlayer = 3 - currentPlayer; // promjena igrača
      dropPieceAnimation=false; // kraj animacije pada žetona
      currentRow=0;}} // vraćanje varijable retka na početno stanje
if(skip){ // ako je započet preskok animacije
  dropPieceAnimation=false; // zaustavljanje animacije pada žetona
  updateLEDMatrix(currentRow-1, currentColumn, 0); //brisanje boje polja
updateLEDMatrix(DropRow, currentColumn, currentPlayer); // boja polja
board[DropRow][DropColumn] = currentPlayer; //postavljanje oznake igrača
  currentPlayer = 3 - currentPlayer; // promjena igrača
  currentRow=0; // vraćanje varijable retka u početno stanje
  currentColumn=0; // vraćanje varijable stupca u početno stanje
      skip=false;}}
```

U nastavku `Animations()` funkcije implementirana je animacija horizontalne pobjedničke kombinacije. Na početku je deklarirana i inicijalizirana lokalna varijabla koja prati vrijednost prošlog igrača. Prvi igrač ima vrijednost 1, dok drugi igrač ima vrijednost 2. U nastavku se nalaze dvije pauze pomoću kojih je ostvareno treperenje pobjedničke kombinacije žetona. Jedna promjena boje se događa svakih 300 ms, a druga promjena boje svakih 150 ms poslije prve promijene boje što znači da se boja pobjedničkih žetona mijenja naizmjenice svakih 150 ms. U svakoj od `if` naredbi, nalazi se jedna `for` petlja koja iterira kroz žetone koji trebaju treperiti.

```
int player = 3 - currentPlayer; // praćenje prijašnjeg igrača
if(HorizontalWin){ // ako je horizontalna pobjeda
  if(time-blinkdelay >= 300){ // pauza od 300 ms
  blinkdelay=time;
    for (int m = WinColumn; m < WinColumn+extracolumn ; m++){
    updateLEDMatrix(WinRow, m, player); // promjena boje polja
    }
    strip.show();
  }
  if(time-blinkdelay >= 150){ //pauza od 150 ms
    for (int m = WinColumn; m < WinColumn+extracolumn ; m++){
    updateLEDMatrix(WinRow, m, player+2); // promjena boje polja
    }
    strip.show();
  }
}
```

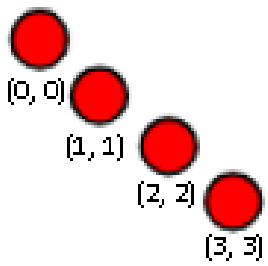
U nastavku `Animations()` funkcije, analogno prethodnom isječku koda, ostvarena je animacija vertikalne pobjedničke animacije.

```
if(VerticalWin){ // ako je vertikalna pobjeda
  if(time-blinkdelay >= 300){ //pauza 300 ms
  blinkdelay=time;
  for (int n = WinRow; n > WinRow-4 ; n--){
    updateLEDMatrix(n, WinColumn, player); // promjena boje polja
  }
  strip.show();
  }
  if(time-blinkdelay >= 150){ //pauza od 150 ms
  for (int n = WinRow; n > WinRow-4; n--){
    updateLEDMatrix(n, WinColumn, player+2); // promjena boje polja
  }
  strip.show();
  }
}
```

U sljedećem isječku koda `Animations()` funkcije implementirana je animacija pobjedničke kombinacije u smjeru dijagonalno prema dolje. Za taj smjer trenutni broj retka (n) i trenutni broj stupca (m) su povezani preko sljedećeg izraza:

$$n = WinRow + m - WinColumn, \tag{7.1}$$

gdje je *WinRow* početni broj retka, a *WinColumn* početni broj stupca. U početku trenutni i početni broj stupca su jednaki što prema gornjem izrazu možemo vidjeti da se ponište. U svakoj iteraciju `for` petlje, trenutni broj stupca se smanji za 1 što uzrokuje da se i broj retka smanji za 1 sve dok trenutni broj stupca, m, ne postigne konačnu vrijednost. Smanjenjem broja retka i stupca `for` petlja iterira kroz koordinate počevši s desna na lijevo dijagonalno prema gore kao što je prikazano na slici 7.2.



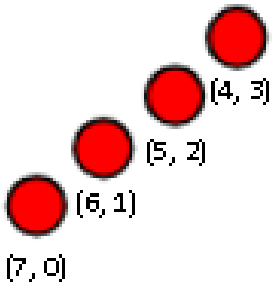
Slika 7.2. Primjer koordinata dijagonalno prema dolje pobjedničke kombinacije.

```
if(DijagonalDownWin){ // ako je dijagonalna pobjeda
if(time-blinkdelay >= 300){ //pauza od 300 ms
blinkdelay=time;
  for (int m = WinColumn; m > WinColumn-extra ; m--){
    int n;
     n = WinRow + m - WinColumn;
     updateLEDMatrix(n, m, player);} // promjena boje polja
strip.show();}
if(time-blinkdelay >= 150){ //pauza od 150 ms
  for (int m = WinColumn; m > WinColumn-extra ; m--){
    int n;
     n = WinRow + m - WinColumn;
     updateLEDMatrix(n, m, player+2);} // promjena boje polja
strip.show();}}
```

U nastavku `Animations()` funkcije, analogno prethodnom isječku koda implementirana je pobjednička animacija u smjeru dijagonalno prema gore. U ovom slučaju trenutni broj retka i stupca povezani su preko sljedećeg izraza:

$$n = WinRow - m + WinColumn. \tag{7.2.}$$

U ovom slučaju `for` petlja iterira po pobjedničkim žetonima počevši od lijevo dolje prema gore desno, kao što je to prikazano na slici 7.3.



Slika 7.3. Primjer koordinata dijagonalno prema gore pobjedničke kombinacije.

```
if(DijagonalUpWin){ // ako je dijagonalna pobjeda
  if(time-blinkdelay >= 300){ //pauza od 300 ms
  blinkdelay=time;
    for (int m = WinColumn; m < WinColumn+extra ; m++){
      int n;
      n = WinRow - m + WinColumn;
      updateLEDMatrix(n, m, player); // promjena boje polja
    }
  strip.show();}
  if(time-blinkdelay >= 150){ //pauza od 150 ms
    for (int m = WinColumn; m < WinColumn+extra ; m++){
      int n;
      n = WinRow - m + WinColumn;
      updateLEDMatrix(n, m, player+2); // promjena boje polja
    }
  strip.show();
  }
}
```

U nastavku `Animations()` funkcije implementirana je *reset* animacija koja se pokreće pritiskom tipkala za *reset*. S pauzom od 60 ms program prolazi kroz postavljene korake animacije. Kroz svaku iteraciju, definirane su koordinate 8x8 matrice koje oblikuju animaciju. Animacija ima dva dijela, gdje prvi dio sadrži 5 koraka (od 0 do 4), a drugi dio 4 koraka (od 0 do 3). U prvom dijelu animacije, prvi korak uključuje središnja četiri polja, drugi korak isključuje središnja četiri polja i uključuje susjedna polja u obliku većeg praznog kvadrata. U trećem i četvrtom koraku, analogno se isključuju prethodne LED diode i uključuju one sljedeće tvoreći sve veći i veći kvadrat. Peti korak je zadnji korak prvog dijela animacije gdje se isključuju sve LED diode, te se pokreće drugi dio animacije.

```
if(resetGameAnimation){ // ako je pritisnuto tipkalo za reset
  if(time-blinkdelay >= 60){ //pauza od 60 ms
    blinkdelay=time;
    if (step == 0) {
        updateLEDMatrix(3, 3, 7);
        updateLEDMatrix(4, 3, 7);
        updateLEDMatrix(3, 4, 7);
        updateLEDMatrix(4, 4, 7);
    } else if (step == 1) {
        for (int j = 2; j < 6; j++) {
            updateLEDMatrix(2, j, 7);
            updateLEDMatrix(5, j, 7);
        }
        for (int j = 3; j < 5; j++) {
            updateLEDMatrix(j, 2, 7);
            updateLEDMatrix(j, 5, 7);
        }
        updateLEDMatrix(3, 3, 0);
        updateLEDMatrix(4, 3, 0);
        updateLEDMatrix(3, 4, 0);
        updateLEDMatrix(4, 4, 0);
    } else if (step == 2) {
        for (int j = 1; j < 7; j++) {
            updateLEDMatrix(1, j, 7);
            updateLEDMatrix(6, j, 7);
        }
        for (int j = 2; j < 6; j++) {
            updateLEDMatrix(j, 1, 7);
            updateLEDMatrix(j, 6, 7);
        }
        for (int j = 2; j < 6; j++) {
            updateLEDMatrix(2, j, 0);
            updateLEDMatrix(5, j, 0);
        }
        for (int j = 3; j < 5; j++) {
            updateLEDMatrix(j, 2, 0);
            updateLEDMatrix(j, 5, 0);
```

```
            }
        } else if (step == 3) {
            for (int j = 0; j < 8; j++) {
                updateLEDMatrix(0, j, 7);
                updateLEDMatrix(7, j, 7);
            }
            for (int j = 1; j < 7; j++) {
                updateLEDMatrix(j, 0, 7);
                updateLEDMatrix(j, 7, 7);
            }
            for (int j = 1; j < 7; j++) {
                updateLEDMatrix(1, j, 0);
                updateLEDMatrix(6, j, 0);
            }
            for (int j = 2; j < 6; j++) {
                updateLEDMatrix(j, 1, 0);
                updateLEDMatrix(j, 6, 0);
            }
        } else if (step == 4) {
            for (int j = 0; j < 8; j++) {
                board[0][j] = 0;
                board[7][j] = 0;
                updateLEDMatrix(0, j, 0);
                updateLEDMatrix(7, j, 0);
            }
            for (int j = 1; j < 7; j++) {
                board[j][0] = 0;
                board[j][7] = 0;
                updateLEDMatrix(j, 0, 0);
                updateLEDMatrix(j, 7, 0);
            }
        }
        strip.show();
        step++;
        if(step==5){
          step=0;
          resetGameAnimation=false;
          secondAnimation=true;
        }
      }
    }
```

Drugi dio *reset* animacije je sličan prvom, ali se sada LED diode uključuju od vanjskog ruba prema središtu. Kroz animaciju se postavljaju polja matrice na vrijednost 0, odnosno brišu se prijašnje označena polja igrača.

```
if(secondAnimation){
  if(time-blinkdelay >= 60){ //Pauza od 60 ms
    blinkdelay=time;
    if (step == 0) {
        for (int j = 1; j < 7; j++) {
            board[1][j] = 0;
            board[6][j] = 0;
            updateLEDMatrix(1, j, 7);
            updateLEDMatrix(6, j, 7);
        }
        for (int j = 2; j < 6; j++) {
            board[j][1] = 0;
            board[j][6] = 0;
            updateLEDMatrix(j, 1, 7);
            updateLEDMatrix(j, 6, 7);
        }
        for (int j = 0; j < 8; j++) {
            updateLEDMatrix(0, j, 0);
            updateLEDMatrix(7, j, 0);
        }
    } else if (step == 1) {
        for (int j = 2; j < 6; j++) {
            board[2][j] = 0;
            board[5][j] = 0;
            updateLEDMatrix(2, j, 7);
            updateLEDMatrix(5, j, 7);
        }
        for (int j = 3; j < 5; j++) {
            board[j][2] = 0;
            board[j][5] = 0;
            updateLEDMatrix(j, 2, 7);
            updateLEDMatrix(j, 5, 7);
        }
        for (int j = 1; j < 7; j++) {
            updateLEDMatrix(1, j, 0);
            updateLEDMatrix(6, j, 0);
        }
        for (int j = 2; j < 6; j++) {
            updateLEDMatrix(j, 1, 0);
            updateLEDMatrix(j, 6, 0);
        }
    } else if (step == 2) {
        board[3][3] = 0;
        board[4][3] = 0;
        board[3][4] = 0;
```

```
            board[4][4] = 0;
            updateLEDMatrix(3, 3, 7);
            updateLEDMatrix(4, 3, 7);
            updateLEDMatrix(3, 4, 7);
            updateLEDMatrix(4, 4, 7);
            for (int j = 2; j < 6; j++) {
                updateLEDMatrix(2, j, 0);
                updateLEDMatrix(5, j, 0);
            }
            for (int j = 3; j < 5; j++) {
                updateLEDMatrix(j, 2, 0);
                updateLEDMatrix(j, 5, 0);
             }
        } else if (step == 3) {
            updateLEDMatrix(3, 3, 0);
            updateLEDMatrix(4, 3, 0);
            updateLEDMatrix(3, 4, 0);
            updateLEDMatrix(4, 4, 0);
        }
      strip.show();
        step++;
        if(step==4){
          step=0;
          secondAnimation=false;
        }
      }
    }
  }
```

U sljedećem isječku koda definirana je funkcija `updateLEDMatrix(int row, int col, int player)` koja kao ulazne argumente prima broj retka i stupca te redni broj igrača. Prvi igrač je predstavljen crvenom bojom, a drugi igrač plavom. Funkcija definira i druge igrače, koji zapravo u ovom kontekstu predstavljaju različite boje (žuta, cijan, zelena, magenta, bijela te isključena LED dioda) koje se koriste u svrhu uljepšavanja izgleda animacija. Pretvaranje matričnih koordinata u redni broj LED diode je izvedeno preko sljedećeg izraza:

$$pixel = col + row * NEOPIXEL\_NUM, \tag{4.3.}$$

gdje *pixel* označava redni broj LED diode na NeoPixel-u, *col* stupac u kojem se određeno polje nalazi, *row* red u kojem se određeno polje nalazi, a *NEOPIXEL_NUM* broj polja u jednom retku/stupcu.

```
void updateLEDMatrix(int row, int col, int player) {
  int pixel = col + row * NEOPIXEL_NUM;
  if (player == 1) {
    strip.setPixelColor(pixel, strip.Color(255, 0, 0));
 } else if (player == 2) {
    strip.setPixelColor(pixel, strip.Color(0, 0, 255));
  }else if (player == 3){
    strip.setPixelColor(pixel, strip.Color(255, 255, 0));
  }else if (player == 4){
    strip.setPixelColor(pixel, strip.Color(0, 255, 255));
  }else if (player == 5){
    strip.setPixelColor(pixel, strip.Color(0, 255, 0));
  }else if (player == 6){
    strip.setPixelColor(pixel, strip.Color(255, 0, 255));
  }else if (player == 7){
    strip.setPixelColor(pixel, strip.Color(255, 255, 255));
  }else if (player == 0){
    strip.setPixelColor(pixel, strip.Color(0, 0, 0));}}
```

U sljedećem isječku koda definirana je `checkForWin(int player)` funkcija koja služi za provjeru pobjede u horizontalnom, vertikalnom ili dijagonalnom smjeru. Općenito, svaka vrsta pobjede započinje s pretpostavkom da je došlo do pobjede sve dok sljedeća `if` naredba ne dokaže suprotno. U svakoj od `if` funkcija se nalazi kratki uvjet koji ispituje da li je uvjet za pobjedu neispunjen. Jednom kada je uspješno pronađena pobjednička kombinacija, u globalne varijable zapisuje se vrsta pobjede kao i početna vrijednost stupca i retka pobjedničke kombinacije, te se započinje odgovarajuća pobjednička animacija.

```
void checkForWin(int player) {
  player = 3 - player; // pracenje proslog igraca
  //horizontalna provjera
  for (int row = NEOPIXEL_NUM-1; row >= 0; row--) {
    for (int col = 4; col >=0; col--) {
    bool win = true;
      for(int i=0; i<4; i++){
        if(board[row][col+i] != player){
          win = false;
          break;}}
      if(win){
        gameOver = true;
        WinRow=row;
        WinColumn=col;
        int j;
        for (j = 4; j < 8; j++){
          if (board[row][col + j]!=player){
            break;}}
      HorizontalWin=true;
      extracolumn=j;}}}
```

```
//Vertikalna provjera
for (int col = 0; col < NEOPIXEL_NUM; col++ ) {
  for (int row = NEOPIXEL_NUM-1; row > 2; row--) {
     bool win = true;
    for(int i=0; i<4; i++){
      if(board[row-i][col] != player){
      win = false;
      break;}}
    if(win){
      gameOver = true;
      WinRow=row;
      WinColumn=col;
      VerticalWin=true;}}}
//dijagonalna dole provjera
for (int row = 3; row <= NEOPIXEL_NUM-1; row++) {
  for (int col = 3; col <= NEOPIXEL_NUM-1; col++) {
    bool win = true;
    for(int i=0; i<4; i++){
      if(board[row-i][col-i] != player){
      win = false;
      break;}}
    if(win){
      int i;
      for (i = 4; i < NEOPIXEL_NUM; i++){
        if (board[row-i][col-i]!=player){
          break;}}
      extra=i;
      gameOver = true;
      WinRow=row;
      WinColumn=col;
      DijagonalDownWin=true;}}}
//dijagonalno gore provjera
for (int row = 3; row < NEOPIXEL_NUM; row++) {
  for (int col = 4; col >= 0; col--) {
    bool win = true;
    for(int i=0; i<4; i++){
      if(board[row-i][col+i] != player){
      win = false;
      break;}}
    if(win){
      int i;
      for (i = 4; i < NEOPIXEL_NUM; i++){
        if (board[row-i][col+i]!=player){
          break;}}
      extra=i;
      gameOver = true;
      WinRow=row;
      WinColumn=col;
      DijagonalUpWin=true;}}}
return;}
```

U sljedećem isječku koda definirana je funkcija `resetGame()` koja vraća sve varijable u početno stanje nakon pritiska na *reset* tipkalo.

```
void resetGame() {
  dropPieceAnimation=false;
  DropRow=0;
  DropColumn=0;
  currentRow=0;
  currentColumn=0;
  HorizontalWin=false;
  VerticalWin=false;
  DijagonalUpWin=false;
  DijagonalDownWin=false;
  WinRow=0;
  WinColumn=0;
  resetGameAnimation=true;
  currentPlayer = 1;
  gameOver = false;
}
```

8. ZAKLJUČAK

Kroz ovaj završni rad na Arduino platformi je implementirana Connect 4 igra za dva igrača koristeći NeoPixel LED matricu za prikaz igre i tipkala za korisnički unos. S detaljno objašnjenom hardverskom i softverskom arhitekturom ostvarena je funkcionalna i interaktivna igra koju je moguće unaprijediti dodavanjem novih značajki poput umjetne inteligencije za igranje protiv računala, dodatnih animacija te mogućnosti mrežnog igranja s drugim igračima.

9. LITERATURA

[1] https://en.wikipedia.org/wiki/Arduino

[2] https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

[3] https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

[4] https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/

[5] http://www.advanced-monolithic.com/pdf/ds1117.pdf

[6] https://docs.arduino.cc/learn/microcontrollers/digital-pins/

[7] https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/

[8] https://learn.sparkfun.com/tutorials/pull-up-resistors/all

[9] https://roboticsbackend.com/arduino-uno-pins-a-complete-practical-guide/

[10] https://circuitjournal.com/arduino-power-pins#3.3v-output

[11] https://www.circuito.io/blog/arduino-uno-pinout/

[12] https://www.microchip.com/en-us/product/atmega328p#

10.SAŽETAK I KLJUČNE RIJEČI

U sklopu ovog završnog rada izrađena je Connect 4 igra za dva igrača temeljena na Arduino platformi. Kontrola igre je ostvarena putem devet tipkala spojenih u *pull-down* konfiguraciji. Igra je prikazana na 8x8 NeoPixel LED zaslonu. Programska podrška napravljena u sklopu rada omogućava osnovnu funkcionalnost Connect 4 igre, uključujući logiku za prepoznavanje pobjedničkog niza žetona u horizontalnom, dijagonalnom i vertikalnom smjeru kao i različiti broj animacija koji uljepšavaju igraće iskustvo.

Ključne riječi: Arduino, NeoPixel, Connect 4.

11. ABSTRACT AND KEYWORDS

In this bachelor thesis, the Connect 4 game for two players has been developed and implemented to the Arduino platform. The game is controlled via nine push-buttons connected in a pull-down configuration. The game is displayed on an 8x8 NeoPixel LED screen. The software developed as part of this thesis provides basic functionality for the Connect 4 game, including logic for recognizing a winning sequence of tokens in horizontal, diagonal, and vertical directions, as well as numerous animations enhancing the game experience.

Keywords: Arduino, NeoPixel, Connect 4.