IRIM - Institut za razvoj i inovativnost mladih

Generacija NOW

Strukovna škola Đurđevac

# Multimedijalni toranj

## Tehnički opis rada

svibanj 2022., Đurđevac

## SADRŽAJ

# 1. PROJEKTNI TIM

Mentor :

Željko Brček mr.ing.el.

Učenici:

Luka Vuković

Gordan Pecikozić

Ivan Žufika

Gabrijel Đurišević

Informacije o školi:

Strukovna škola Đurđevac

Dr. Ivana Kranjčeva 5

http://ss-strukovna-djurdjevac.skole.hr/

## 2. KOMPONENTE KORIŠTENE ZA IZRADU

RGB LED diode 192 kom

Arduino MKR 1000 – 1kom

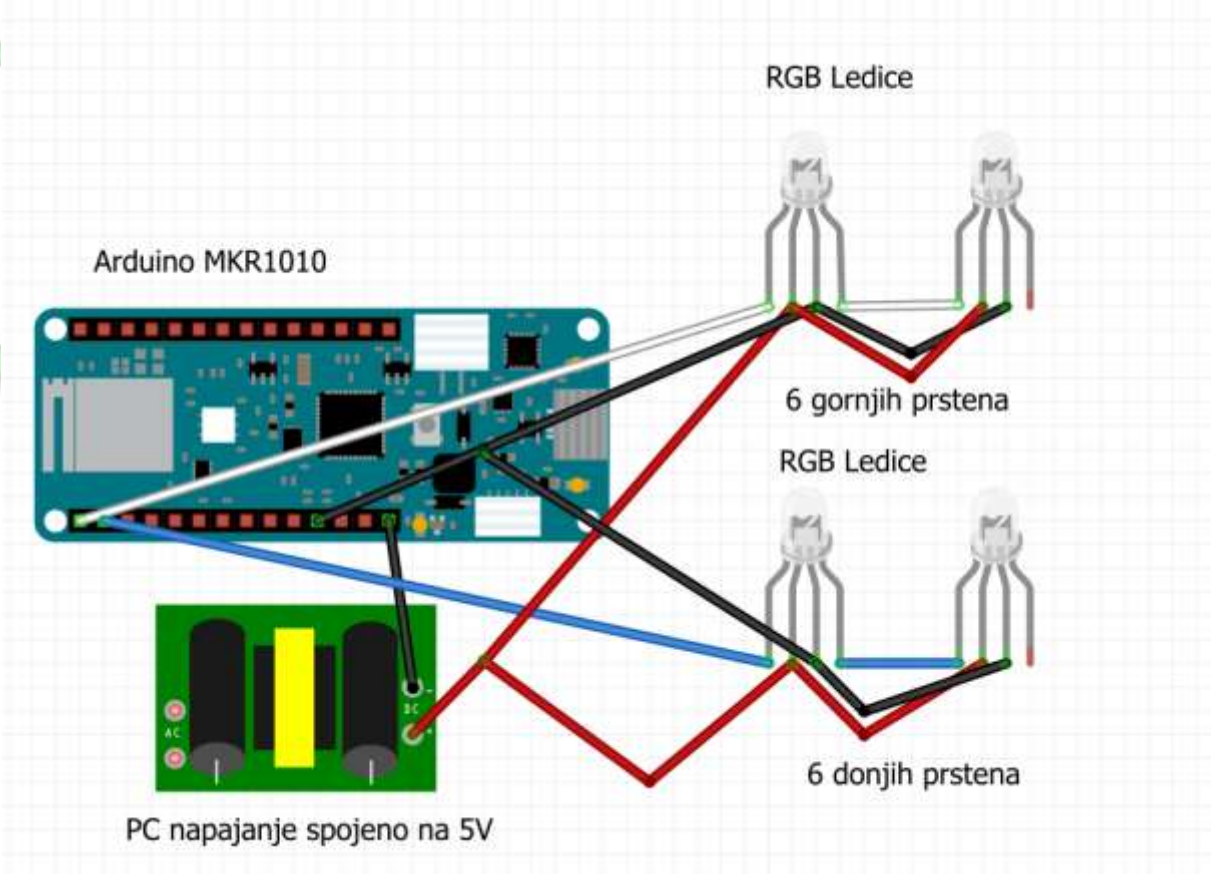
bakrena žica 1,5 mm2

plastični prsteni (3d ispis)

plastične poveznice (3d ispis)

PC napajanje

# 3. SHEME SPAJANJA



# 4. FAZE IZRADE

Projekt smo započeli izradom 12 bakrenih prstena promjer 15.5 cm, nakon čega smo 3D printali plastične prstene u koje smo stavili 16 LED dioda po prstenu. Nakon stavljanja LED dioda smo rasporedili njihove nožice tako da bi ulazi i izlazi dioda bili pravilno raspoređeni. Poslije uspješnog raspoređivanja smo zalemili nožice međusobno kao i na bakrene prstene. Prstene smo spojili pomoću 3D printanih stupića kako bi sve stajalo na svom mjestu. Napajanje smo stavili u 3D printanu kutiju i spojili ga s Arduino pločicom MKR-1000. Izrada/dorada arduino koda.

U konačnici smo spojili toranj s napjanjem i na Arduino pločicu smo prenijeli programski kod.

# 5. PROGRAMSKI KOD

Glavni:

```
// glavna petlja je samo popis poziva raznim animacijama


void loop() {
 sinusniVal();
 triPrstena2();
 spiralniDemo();
 velikaSpirala();
 toranjPrijelaz();
 sfere();
 rotacijaCrvenih();
 zicaniOkvir();
 sjenaPokret();
 dupliPrijelaz();
 ravninaAnimacija();
```

```
  toranjPad();
  triPrstena1();
  rotacijskiOval();
  triPrstena3();
  prijelaznaRavnina();
  softwareReset();
}
```

# Mutimedijalni toranj:

```
// za LED diode u tornju 24 stupca okolo i 12 redova visoko
// #include <avr/wdt.h>
//postavka za FastLED biblioteku
#include "FastLED.h"
#define NUM_LEDS 144  // LED diode u pola tornja
#define PIN1 6  // kontrolira gornjih šest prstenova
#define PIN2 7  // kontrolira donjih šest prstenova

// moja paleta standardnih boja, s prilagodbama za ono što najbolje izgleda na APA106 LED diodama
#define Crvena CRGB::Red
#define Narancasta CRGB:: OrangeRed
#define Zuta CRGB:: Gold
#define Zelena CRGB::Green
#define SvijetloPlava CRGB::Aqua
#define Plava CRGB::Blue
#define Purpurna CRGB::Purple
#define Ljubicasta CRGB::DarkViolet
#define Bijela CRGB::White
```

```
#define Crna CRGB::Black


CRGB ledGore[NUM_LEDS];  // niz koji kontrolira gornju polovicu tornja

CRGB ledDolje[NUM_LEDS];  // niz koji kontrolira donju polovicu tornja

CRGB mojaBoja;

byte mojaNijansa;


// ova tablica sadrži grupe od 12 brojeva, koji zajedno predstavljaju status tornja.

// Svaki broj predstavlja prsten odozdo prema vrhu, a najnižih 12 bitova predstavljaju svaku polovicu prstena, pri čemu je svaka polovica simetrična.

// Postoji 18 slika tornja, od kojih svaka prikazuje presjek ravnine pod različitim kutom.

const int ravnina[18][12] PROGMEM ={

  // vrh ---------------------------------------------------- ---------------------------------------- ----------
---------------------------- ----------> dno (prstenovi)

  // 0&23 ----- 11&13 (stupci)

  // ovdje gore je kut 0 (ravno)

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b111111111111},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000111111,
0b111111000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000111, 0b000011111000,
0b111100000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000000, 0b000000000111, 0b000000111000, 0b000111000000,
0b111000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000111, 0b000000011000, 0b000011100000, 0b001100000000,
0b110000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000011, 0b000000001100, 0b000000110000, 0b000011000000, 0b001100000000,
0b110000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000111, 0b000000011000, 0b000001100000, 0b000010000000, 0b001100000000,
0b110000000000},
```

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000011, 0b000000001100, 0b000000110000, 0b000001000000, 0b000110000000, 0b001000000000, 0b110000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000111, 0b000000011000, 0b000000100000, 0b000011000000, 0b000100000000, 0b111000000000, 0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000111, 0b000000011000, 0b000000100000, 0b000001000000, 0b000110000000, 0b001000000000, 0b110000000000, 0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000111, 0b000000001000, 0b000000010000, 0b000000100000, 0b000011000000, 0b000100000000, 0b001000000000, 0b110000000000, 0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000111, 0b000000001000, 0b000000010000, 0b000000100000, 0b000001000000, 0b000010000000, 0b000100000000, 0b001000000000, 0b110000000000, 0b000000000000},

{0b000000000000, 0b000000000110, 0b000000001000, 0b000000010000, 0b000000100000, 0b000000100000, 0b000001000000, 0b000010000000, 0b000100000000, 0b000100000000, 0b011000000000, 0b000000000000},

{0b000000000000, 0b000000001100, 0b000000001000, 0b000000010000, 0b000000100000, 0b000000100000, 0b000001000000, 0b000001000000, 0b000010000000, 0b000100000000, 0b000100000000, 0b000000000000},

{0b000000000000, 0b000000001000, 0b000000010000, 0b000000010000, 0b000000100000, 0b000000100000, 0b000001000000, 0b000001000000, 0b000010000000, 0b000010000000, 0b000100000000, 0b000100000000},

{0b000000010000, 0b000000010000, 0b000000010000, 0b000000100000, 0b000000100000, 0b000000100000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000010000000, 0b000010000000, 0b000010000000},

{0b000000100000, 0b000000100000, 0b000000100000, 0b000000100000, 0b000000100000, 0b000000100000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000},

{0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000, 0b000001000000}

};

// ovdje dolje je kut 17 (okomito)

// ova tablica sadrži grupe od 12 brojeva, koji zajedno predstavljaju status tornja.

// Svaki broj predstavlja prsten odozdo prema vrhu, a najnižih 12 bitova predstavljaju svaku polovicu prstena, pri čemu je svaka polovica simetrična.

// postoji 9 skupova brojeva, od kojih svaki predstavlja fazu u procesu prolaska kugle kroz cilindar

const int sfera[9][12] PROGMEM ={

```
{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b100000000000, 0b100000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b110000000000,
0b010000000000, 0b010000000000, 0b110000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b011000000000, 0b001100000000,
0b000100000000, 0b000100000000, 0b001100000000, 0b011000000000, 0b000000000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000110000000, 0b000011000000, 0b000001000000,
0b000001000000, 0b000001000000, 0b000001000000, 0b000011000000, 0b000110000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000010010000, 0b000001100000, 0b000000000000, 0b000000000000,
0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000001100000, 0b000010010000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000011000, 0b000000110000, 0b000000100000,
0b000000100000, 0b000000100000, 0b000000100000, 0b000000110000, 0b000000011000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000110, 0b000000001100,
0b000000001000, 0b000000001000, 0b000000001100, 0b000000000110, 0b000000000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000011,
0b000000000010, 0b000000000010, 0b000000000011, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000},

{0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000001, 0b000000000001, 0b000000000000, 0b000000000000, 0b000000000000, 0b000000000000,
0b000000000000}

};

// 2 sekunde sa svim isključenim - osigurava sigurno razdoblje za preuzimanje ako je potrebno


void setup() {
  FastLED.setBrightness(25);
  FastLED.addLeds<APA106, PIN1, RGB>(ledGore, NUM_LEDS);
  FastLED.addLeds<APA106, PIN2, RGB>(ledDolje, NUM_LEDS);
  ocistiSve();
  delay(2000);
}
```

# P_Animacije:

```
// evo animacija koje sam uključio u svoju emisiju
// ovdje nema puno komentara o pojedinačnim animacijama,
// ali su potprogrami u sljedećoj kartici vrlo dobro dokumentirani
// ako ste zainteresirani za pokušaj izrade vlastitih animacija


void sfere() {
 for (int protuteza = 0; protuteza < 23; protuteza++) {
  for (int k = 0; k < 9; k++) {
   for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
     if (pgm_read_word_near(&sfera[k][11 - j]) & (1 << i)) {
      int noviRed = i + protuteza; if (noviRed > 24) noviRed -= 24;
      int noviRed2 = i - protuteza; if (noviRed2 < 0) noviRed2 += 24;
      postaviNijansu(j, noviRed, mojaNijansa);
      postaviNijansu(j, 23 - noviRed2,  mojaNijansa);
     }
    }
   }
   mojaNijansa+= 5;
   ocistiIzgled(25);
  }
  for (int k = 8; k > -1; k--) {
   for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {

     if (pgm_read_word_near(&sfera[k][11 - j]) & (1 << i)) {
      int noviRed = i + protuteza; if (noviRed > 24) noviRed -= 24;
      int noviRed2 = i - protuteza; if (noviRed2 < 0) noviRed2 += 24;
      postaviNijansu(j, noviRed, mojaNijansa);
      postaviNijansu(j, 23 - noviRed2,  mojaNijansa);
```

```
        }
       }
      }
      mojaNijansa+= 5;
      ocistiIzgled(25);
    }
  }
  stanka();
}



void sinusniVal() {
  svjetlina(40);
  for (int k = 1; k < 13; k++) {
    for (int j = 0; j < 12; j++) {
      for (int i = 0; i < 12; i++) {
        if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
          postaviPaletu(j, i, Plava);
          postaviPaletu(j, 23 - i, Plava);
        }
      }
    }
    for (int i = 0; i < 23; i++) {
      for (int m = 0; m < 12; m++) {
        rotiraj(m, 1);
      }
      odrziIzgled(30 + k / 2);
    }
    ocistiSve();
  }
  float adj = 3.8;
  svjetlina(50);
  for (int zbroj = 0; zbroj < 15; zbroj++) {
```

```
  for (int i = 0; i < 40; i++) {
    for (byte xx = 0; xx < 24; xx++) {
      byte z = ((byte)(6 + cos((xx / adj) + (float)i / 6.28) * 6)); // stvarni z izračun
      if (z<12) postaviPaletu(z, xx, Plava); // stavi z na odgovarajuće mjesto za svaku poziciju u kvadrantu 1
    }
    ocistiIzgled(5);
  }
  adj -= .2;
  if (adj < 1.9)adj = 1.9;
 }
 for (int zbroj = 0; zbroj < 12; zbroj++) {
  for (int i = 0; i < 40; i++) {
    for (byte xx = 0; xx < 24; xx++) {
      byte z = ((byte)(6 + cos((xx / adj) + (float)i / 6.28) * 6)); // stvarni z izračun
      if (z<12)postaviPaletu(z, xx, Plava); // stavi z na odgovarajuće mjesto za svaku poziciju u kvadrantu 1
    }
    ocistiIzgled(5);
  }
  adj += .2;
  if (adj > 3.8)adj = 3.8;
 }
 svjetlina(40);
 for (int k = 12; k > 0; k--) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
        postaviPaletu(j, i, Plava);
        postaviPaletu(j, 23 - i, Plava);
      }
    }
  }
```

```
    for (int i = 0; i < 23; i++) {
      for (int m = 0; m < 12; m++) {
        rotiraj(m, 1);
      }
      odrziIzgled(30 + k / 2);
    }
    ocistiSve();
  }
  svjetlina(25);
  stanka();
}


void sjenaPokret() {
  mojaNijansa= 0;
  napuniToranj(mojaNijansa);
  for (int zbroj = 0; zbroj < 2; zbroj++) {
    for (int k = 1; k < 13; k++) {
      for (int j = 0; j < 12; j++) {
        for (int i = 0; i < 12; i++) {
          if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
            postaviPaletu(j, i, Plava);
            postaviPaletu(j, 23 - i, Plava);
          }
        }
      }
      for (int i = 0; i < 23; i++) {
        for (int m = 0; m < 12; m++) {
          rotiraj(m, 0); // u smjeru kazaljke na satu
        }
        odrziIzgled(10 + k / 2);
      }
      napuniToranj(mojaNijansa);
```

```
    mojaNijansa+= 1;
   }
   for (int k = 11; k > 1; k--) {
    for (int j = 0; j < 12; j++) {
     for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
       postaviPaletu(j, i, Plava);
       postaviPaletu(j, 23 - i, Plava);
      }
     }
    }
    for (int i = 0; i < 23; i++) {
     for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
     }
     odrziIzgled(10 + k / 2);
    }
    napuniToranj(mojaNijansa);
    mojaNijansa+= 1;
   }
  }
 for (int zbroj = 0; zbroj < 15; zbroj++) {
  for (int i = 0; i < 40; i++) {
   for (byte xx = 0; xx < 24; xx++) {
    byte z = ((byte)(6 + cos((xx / 1.9) + (float)i / 6.28) * 6)); // stvarni z izračun
    if (z<12) postaviPaletu(z, xx, Plava); // stavi z na odgovarajuće mjesto za svaku poziciju u
kvadrantu 1
   }
   odrziIzgled(20);
   napuniToranj(mojaNijansa);
  }
  mojaNijansa-= 3;
 }
```

```
  stanka();
}

void triPrstena1() {
  for (int zbroj = 0; zbroj < 18; zbroj++) {
    for (int prsten = 0; prsten < 24; prsten++) {
      int prsten2, prsten3, prsten4;
      if (prsten < 12) prsten2 = prsten;
      else prsten2 = 23 - prsten;
      if (prsten > 7  && prsten < 20) prsten3 = prsten - 8;
      if (prsten < 9) prsten3 = 8 - prsten;
      if (prsten > 19) prsten3 = 31 - prsten;
      if (prsten > 15) prsten4 = prsten - 16;
      if (prsten < 4) prsten4 = prsten + 8;
      if (prsten > 3 && prsten < 16) prsten4 = 15 - prsten;
      postaviPrsten(prsten2, 0);
      if (zbroj > 4) postaviPrsten(prsten3, 167);
      if (zbroj > 9) postaviPrsten(prsten4, 83);
      ocistiIzgled(15 + zbroj * 4);
    }
  }
  stanka();
}

void triPrstena2() {
  int pos = 0;
  int dir = 1;
  for (int zbroj = 0; zbroj < 48; zbroj++) {
    for (int j = 0; j < 12; j++) {
      for (int i = 0; i < 12; i++) {
        if (pgm_read_word_near(&ravnina[8][11 - j]) & (1 << i)) {
          postaviPaletu(j + 2, i, Crvena);
```

```
        postaviPaletu(j + 2, 23 - i, Crvena);
       }
      }
     }

     for (int i = 0; i < 23; i++) {
      for (int m = 0; m < 12; m++) {
       rotiraj(m, 0);
      }
      odrziIzgled(5);
     }

     ocistiSve();
     postaviPrsten(pos, 83);
     postaviPrsten(11 - pos, 167);
     pos = pos + dir;
     if (pos == -1) {
      pos = 1;
      dir = +1;
     }
     if (pos == 12) {
      pos = 10;
      dir = -1;
     }
     odrziIzgled(1);
   }
   stanka();
}


void triPrstena3() {
  njihanje(1);
  njihanje(2);
```

```
 njihanje(3);
 stanka();
}

void njihanje(int pos) {
 if (pos == 1)mojaNijansa= 0;
 if (pos == 2)mojaNijansa= 167;
 if (pos == 3)mojaNijansa= 83;
 postavi2Prstena(pos);
 for (int k = 1; k < 10; k++) {
  for (int j = 0; j < 10; j++) {
   for (int i = 0; i < 12; i++) {
    if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
     postaviNijansu(j + 2, i, mojaNijansa);
     postaviNijansu(j + 2, 23 - i, mojaNijansa);
    }
   }
  }
  for (int i = 0; i < 23; i++) {
   for (int m = 0; m < 12; m++) {
    rotiraj(m, 0);
   }
   odrziIzgled(5 + k / 2);
  }
  ocistiSve();
  postavi2Prstena(pos);
 }
 for (int k = 8; k > 0; k--) {
  for (int j = 0; j < 10; j++) {
   for (int i = 0; i < 12; i++) {
    if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
     postaviNijansu(j + 2, i, mojaNijansa);
     postaviNijansu(j + 2, 23 - i, mojaNijansa);
```

```
      }
     }
    }
    for (int i = 0; i < 23; i++) {
     for (int m = 0; m < 12; m++) {
      rotiraj(m, 0);
     }
     odrziIzgled(5 + k / 2);
    }
    ocistiSve();
    postavi2Prstena(pos);
  }
}

void postavi2Prstena(int pos) {
  if (pos == 1) {
    postaviPrsten(1, 167);
    postaviPrsten(0, 83);
  }
  else if (pos == 2) {
    postaviPrsten(1, 83);
    postaviPrsten(0, 0);
  }
  else if (pos == 3) {
    postaviPrsten(1, 0);
    postaviPrsten(0, 167);
  }
}



void prijelaznaRavnina() {
  int mojeZakasnjenje = 5;
  for (int zbroj = 0; zbroj < 2; zbroj++) {
```

```
for (int k = 1; k < 9; k++) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
        postaviNijansu(j, i, mojaNijansa);
        postaviNijansu(j, 23 - i, mojaNijansa);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(mojeZakasnjenje);
  }
  ocistiSve();
  mojaNijansa+= 3;
}
for (int k = 1; k < 4; k++) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[8][11 - j]) & (1 << i)) {
        postaviNijansu(j + k, i, mojaNijansa);
        postaviNijansu(j + k, 23 - i, mojaNijansa);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(mojeZakasnjenje);
  }
```

```
  if (k < 3)ocistiSve();
  mojaNijansa+= 3;
 }
 for (int i = 0; i < 12; i++) {
  for (int m = 0; m < 12; m++) {
   rotiraj(m, 0); // u smjeru kazaljke na satu
  }
  odrziIzgled(mojeZakasnjenje);
 }
 ocistiSve();
 mojaNijansa+= 3;
 for (int k = 7; k > 1; k--) {
  for (int j = 0; j < 12; j++) {
   for (int i = 0; i < 12; i++) {
    if (pgm_read_word_near(&ravnina[k][j]) & (1 << i)) {
     postaviNijansu(j, i, mojaNijansa);
     postaviNijansu(j, 23 - i, mojaNijansa);
    }
   }
  }
  for (int i = 0; i < 23; i++) {
   for (int m = 0; m < 12; m++) {
    rotiraj(m, 0); // u smjeru kazaljke na satu
   }
   odrziIzgled(mojeZakasnjenje);
  }
  ocistiSve();
  mojaNijansa+= 3;
 }

 for (int k = 1; k < 8; k++) {
  for (int j = 0; j < 12; j++) {
   for (int i = 0; i < 12; i++) {
```

```
      if (pgm_read_word_near(&ravnina[k][j]) & (1 << i)) {
        postaviNijansu(j, i, mojaNijansa);
        postaviNijansu(j, 23 - i, mojaNijansa);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(mojeZakasnjenje);
  }
  if (k < 7)ocistiSve();
  mojaNijansa+= 3;
}

for (int i = 0; i < 12; i++) {
  for (int m = 0; m < 12; m++) {
    rotiraj(m, 0); // u smjeru kazaljke na satu
  }
  odrziIzgled(mojeZakasnjenje);
}
ocistiSve();
mojaNijansa+= 3;


for (int k = 3; k > 0; k--) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[8][11 - j]) & (1 << i)) {
        postaviNijansu(j + k, i, mojaNijansa);
        postaviNijansu(j + k, 23 - i, mojaNijansa);
      }
    }
```

```
    }
    for (int i = 0; i < 23; i++) {
      for (int m = 0; m < 12; m++) {
        rotiraj(m, 0); // u smjeru kazaljke na satu
      }
      odrziIzgled(mojeZakasnjenje);
    }
    ocistiSve();
    mojaNijansa+= 3;
  }

  for (int k = 8; k > 1; k--) {
    for (int j = 0; j < 12; j++) {
      for (int i = 0; i < 12; i++) {
        if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
          postaviNijansu(j, i, mojaNijansa);
          postaviNijansu(j, 23 - i, mojaNijansa);
        }
      }
    }
    for (int i = 0; i < 23; i++) {
      for (int m = 0; m < 12; m++) {
        rotiraj(m, 0); // u smjeru kazaljke na satu
      }
      odrziIzgled(mojeZakasnjenje);
    }
    ocistiSve();
    mojaNijansa+= 3;
  }
 }
}

void ravninaAnimacija() {
```

```
for (int k = 1; k < 18; k++) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
        postaviPaletu(j, i, Crvena);
        postaviPaletu(j, 23 - i, Crvena);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(10 + k / 2);
  }
  ocistiSve();
}
mojaNijansa= 0;
for (int k = 0; k < 5; k++) {
  mojaNijansa+= 34;
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[17][j]) & (1 << i)) {
        postaviNijansu(j, i, mojaNijansa);
        postaviNijansu(j, 23 - i, mojaNijansa);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(18);
```

```
  }
  ocistiSve();
}

for (int k = 16; k > 0; k--) {
  for (int j = 0; j < 12; j++) {
    for (int i = 0; i < 12; i++) {
      if (pgm_read_word_near(&ravnina[k][j]) & (1 << i)) {
        postaviPaletu(j, i, Plava);
        postaviPaletu(j, 23 - i, Plava);
      }
    }
  }
  for (int i = 0; i < 23; i++) {
    for (int m = 0; m < 12; m++) {
      rotiraj(m, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(10 + k / 2);
  }
  ocistiSve();
}

for (int m = 11; m >= 1; m--) {
  for (int i = 0; i < 23; i++) {
    postaviNijansu(m, i, 167 + 4 * (11 - m));
  }
  ocistiIzgled(75);
}
for (int m = 0; m < 12; m++) {
  for (int i = 0; i < 23; i++) {
    postaviNijansu(m, i, 211 + 4 * m);
  }
  ocistiIzgled(75);
```

```
}
svjetlina(45);
mojaNijansa= 0;
for (int rot = 0; rot < 12; rot++) {
 for (int k = 1; k < 18; k++) {
  for (int j = 0; j < 12; j++) {
   for (int i = 0; i < 12; i++) {
    if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
     postaviNijansu(11 - j, i, mojaNijansa);
     postaviNijansu(11 - j, 23 - i, mojaNijansa);
    }
   }
  }
  for (int i = 0; i < rot; i++) {
   for (int m = 0; m < 12; m++) {
    rotiraj(m, 0); // u smjeru kazaljke na satu
   }
  }
  ocistiIzgled(5);
  if (k % 2) mojaNijansa++;
 }

 for (int k = 17; k > -1; k--) {
  for (int j = 0; j < 12; j++) {
   for (int i = 0; i < 12; i++) {
    if (pgm_read_word_near(&ravnina[k][11 - j]) & (1 << i)) {
     postaviNijansu(11 - j, 11 - i, mojaNijansa);
     postaviNijansu(11 - j, 12 + i, mojaNijansa);
    }
   }
  }
  for (int i = 0; i < rot; i++) {
   for (int m = 0; m < 12; m++) {
```

```
        rotiraj(m, 0); // u smjeru kazaljke na satu
       }
      }
      ocistiIzgled(5);
      if (k % 2) mojaNijansa++;
    }

  }
  svjetlina(25);
  for (int m = 11; m >= 1; m--) {
    for (int i = 0; i < 23; i++) {
      postaviNijansu(m, i, 167 + 7 * (11 - m));
    }
    ocistiIzgled(75);
  }
  stanka();
}


void toranjPad() {
  for (int i = 0; i < 300; i++) {
    mojaNijansa+= 2;
    int myran = random(24);
    int myran2 = random(24);
    postaviNijansu(11, myran, mojaNijansa);
    postaviNijansu(11, myran2, random(255));
    odrziIzgled(100);
    copyDolje();
    if ((i < 50) || (i > 250)) {
      postaviPaletu(11, myran, Crna);
      postaviPaletu(11, myran2, Crna);
    }
  }
```

```
  stanka();
}



void zicaniOkvir() {
  byte k1, k2, k3;
  for (int k = 0; k < 300; k++) {
    mojaNijansa+= 5;
    k1 = k % 24;
    k2 = k % 24 + 8; if (k2 > 23) k2 = k2 - 24;
    k3 = k % 24 + 16; if (k3 > 23) k3 = k3 - 24;
    postaviPrsten(11, mojaNijansa);
    postaviPrsten(0, mojaNijansa);
    postaviStup(k1, mojaNijansa);
    postaviStup(k2, mojaNijansa);
    postaviStup(k3, mojaNijansa);
    postaviPaletu(11, k1, Crna);
    postaviPaletu(11, k2, Crna);
    postaviPaletu(11, k3, Crna);
    postaviPaletu(0, k1, Crna);
    postaviPaletu(0, k2, Crna);
    postaviPaletu(0, k3, Crna);
    for (int j = 0; j < 12; j++) {
      rotiraj(j, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(75);
  }
  stanka();
}

void toranjPrijelaz() {
  for (int j = 0; j < 300; j++) {
```

```
    mojaNijansa++;
    if (mojaNijansa% 2 == 0)postaviPrsten(random(12), 21 * random(12));
    postaviStup(random(24), 21 * random(12));
    odrziIzgled(100);
    prijediSve(150);
  }
  stanka();
}

void rotacijskiOval() {
  byte mojaNijansa= 0;
  int o = 0; // protuteza
  for (int i = 0; i < 30; i++) {
    if (i < 9)o = i;
    if (i > 8 && i < 24) o = 16 - i;
    if (i > 23) o = i - 30;
    if (o + 8 < 12)postaviNijansu(o + 8, 0, mojaNijansa);
    if (o + 8 < 12)postaviNijansu(o + 8, 1, mojaNijansa);
    if (o + 7 < 12)postaviNijansu(o + 7, 2, mojaNijansa);
    if (o + 7 < 12)postaviNijansu(o + 7, 3, mojaNijansa);
    if (o + 6 < 12)postaviNijansu(o + 6, 4, mojaNijansa);
    if (o + 5 < 12)postaviNijansu(o + 5, 5, mojaNijansa);
    if (o + 5 < 12)postaviNijansu(o + 5, 6, mojaNijansa);
    if (o + 4 < 12)postaviNijansu(o + 4, 7, mojaNijansa);
    if (o + 3 > -1)postaviNijansu(o + 3, 8, mojaNijansa);
    if (o + 3 > -1)postaviNijansu(o + 3, 9, mojaNijansa);
    if (o + 2 > -1)postaviNijansu(o + 2, 10, mojaNijansa);
    if (o + 2 > -1)postaviNijansu(o + 2, 11, mojaNijansa);

    if (o + 2 > -1)postaviNijansu(o + 2, 12, mojaNijansa);
    if (o + 2 > -1)postaviNijansu(o + 2, 13, mojaNijansa);
    if (o + 3 > -1)postaviNijansu(o + 3, 14, mojaNijansa);
    if (o + 3 > -1)postaviNijansu(o + 3, 15, mojaNijansa);
```

```
    if (o + 4 < 12)postaviNijansu(o + 4, 16, mojaNijansa);
    if (o + 5 < 12)postaviNijansu(o + 5, 17, mojaNijansa);
    if (o + 5 < 12)postaviNijansu(o + 5, 18, mojaNijansa);
    if (o + 6 < 12)postaviNijansu(o + 6, 19, mojaNijansa);
    if (o + 7 < 12)postaviNijansu(o + 7, 20, mojaNijansa);
    if (o + 7 < 12)postaviNijansu(o + 7, 21, mojaNijansa);
    if (o + 8 < 12)postaviNijansu(o + 8, 22, mojaNijansa);
    if (o + 8 < 12)postaviNijansu(o + 8, 23, mojaNijansa);
    for (int k = 0; k < 23; k++) {
     for (int j = 0; j < 12; j++) {
       rotiraj(j, 1); // u smjeru kazaljke na satu
     }
     odrziIzgled(12);
    }
    mojaNijansa+= 9;
    ocistiSve();
  }
  stanka();
}

void goreDoljePrsteni() {
  for (int i = 0; i < 20; i++) {
   for (int j = 0; j < 12; j++) {
     for (int k = 0; k < 24; k++) {
       postaviPaletu(j, k, Plava);
     }
     ocistiIzgled(40);
    }
    for (int j = 10; j > 0; j--) {
     for (int k = 0; k < 24; k++) {
       postaviPaletu(j, k, Plava);
     }
     ocistiIzgled(40);
```

```
    }
  }
  stanka();
}

void rotacijaCrvenih() {
  postaviPaletu(0, 0, Crvena);
  postaviPaletu(1, 0, Crvena);
  postaviPaletu(2, 0, Crvena);
  postaviPaletu(3, 0, Crvena);
  postaviPaletu(4, 0, Crvena);
  postaviPaletu(5, 0, Crvena);
  postaviPaletu(6, 0, Crvena);
  postaviPaletu(7, 0, Crvena);
  postaviPaletu(8, 0, Crvena);
  postaviPaletu(9, 0, Crvena);
  postaviPaletu(10, 0, Crvena);
  postaviPaletu(11, 0, Crvena);
  odrziIzgled(1);
  for (int k = 400; k < 800; k++) {
    for (int j = 0; j < 12; j++) {
      if (j % 2)rotiraj(j, 1); // u smjeru kazaljke na satu
      else rotiraj(j, 0);
    }
    odrziIzgled(110 - (k >> 3));
  }
  stanka();
}

// ovo je spirala koja premašuje 1 punu revoluciju, tako da // nije bilo moguće napraviti spiralnu rutinu
void velikaSpirala() {
  postaviPaletu(0, 0, Zelena);
```

```
  postaviPaletu(1, 3, Zelena);
  postaviPaletu(2, 6, Zelena);
  postaviPaletu(3, 9, Zelena);
  postaviPaletu(4, 12, Zelena);
  postaviPaletu(5, 15, Zelena);
  postaviPaletu(6, 18, Zelena);
  postaviPaletu(7, 21, Zelena);
  postaviPaletu(8, 0, Zelena);
  postaviPaletu(9, 3, Zelena);
  postaviPaletu(10, 6, Zelena);
  postaviPaletu(11, 9, Zelena);
  odrziIzgled(1);
  for (int k = 0; k < 300; k++) {
    for (int j = 0; j < 12; j++) {
      rotiraj(j, 0); // u smjeru kazaljke na satu
    }
    odrziIzgled(25);
  }
  stanka();
}


// Prstenovi se okreću u suprotnim smjerovima i blijede
void dupliPrijelaz() {
  for (int i = 0; i < 15; i++) {
    for (int k = 0; k < 24; k++) {
      for (int j = 0; j < 12; j++) {
        if (j % 2)postaviPaletu(j, k, Zelena);
        else postaviPaletu(j, 23 - k, Plava);
      }
      odrziIzgled(50);
      prijediSve(150);
    }
```

```
  }
  stanka();
}



// spiralni demo
// parametri: stupanj spirale, boja, broj rotacija, kašnjenje, smjer rotacije, smjer spirale
// radi do jednog punog okretaja; npr. stupanj 7 sa 7 prstenova int(49/2)=24; 4 sa 12 prstenova (4*12)/2=24
void spiralniDemo() {
  spiral(1, Zuta, 5, 25, 0, 1);
  spiral(2, Zelena, 5, 25, 0, 0);
  spiral(3, SvijetloPlava, 5, 25, 0, 1);
  spiral(4, Crvena, 5, 25, 0, 0);
  spiral(4, Plava, 5, 75, 1, 1);
}
```

# Podprogrami:

```
// Sve su to rutine niske razine koje koriste sve animacije.

// imajte na umu da glavna petlja ne upućuje izravno na FastLED upute
// sve su sadržane u mojim vlastitim funkcijama i definicijama boja

// postavlja maksimalnu svjetlinu svih LED dioda.
void svjetlina(int bright){  // nominalna je 25
  FastLED.setBrightness(bright);
}

// kopira sadržaj svakog prstena prsten ispod njega
// gornji prsten je očišćen.
void copyDolje() {
```

```
  for (int i = 0; i < 24; i++) {
    for (int j = 0; j < 11; j++) {
      if (j > 5) {
        ledGore[(j - 6) * 24 + i] = ledGore[(j - 6) * 24 + i + 24];
      }
      else if (j < 5) {
        ledDolje[j * 24 + i] = ledDolje[j * 24 + i + 24];
      }
      else {  // j=5
        ledDolje[j * 24 + i] = ledGore[(j - 6) * 24 + i + 24];
      }
    }
  }
}

// postavlja cijeli prsten u nijansu
void postaviPrsten(int prsten, byte nijansa) {
  for (int k = 0; k < 24; k++) {
    postaviNijansu(prsten, k, nijansa);
  }
}

// postavlja cijeli okomiti stupac u nijansu
void postaviStup(int stup, byte nijansa) {
  for (int k = 0; k < 12; k++) {
    postaviNijansu(k, stup, nijansa);
  }
}
// generira širok izbor rotirajućih spirala
void spiral(int stupanj, CRGB mojaBoja, int rotations, int mojeZakasnjenje, byte rotdir, byte spirdir){
  for (int prsten=0; prsten<12; prsten++){
    if (spirdir) postaviPaletu(11-prsten, prsten*stupanj/2, mojaBoja);
    else postaviPaletu(prsten, prsten*stupanj/2, mojaBoja);
```

```
  }
   for (int k = 0; k < rotations*24; k++) {
    for (int j = 0; j < 12; j++) {
      rotiraj(j, rotdir);
    }
    odrziIzgled(mojeZakasnjenje);
  }
  ocistiSve();
}

// sve blijedi brzinom koju određuje x, što je bliže 255, to je blijeđenje sporije
void prijediSve(byte x) {
  for (int i = 0; i < NUM_LEDS; i++) {
    ledDolje[i].nscale8(x);
    ledGore[i].nscale8(x);
  }
}

// postavlja LED na duginu boju od 0 - 255
// određuje LED svojim redom (prstenom) i stupcem (pozicijom u tom krugu)
void postaviNijansu(byte red, byte stupac, byte nijansa) {
  if (red < 6) ledDolje[red * 24 + stupac] = CHSV(nijansa, 255, 255);
  else ledGore[(red - 6) * 24 + stupac] = CHSV(nijansa, 255, 255);
}

// postavlja LED na jednu od standardnih boja definiranih mojom paletom
// specificira LED svojim redom (prstenom) i stupcem (pozicijom u tom krugu
void postaviPaletu(byte red, byte stupac, CRGB mojaBoja) {
  if (red < 6) ledDolje[red * 24 + stupac] = mojaBoja;
  else ledGore[(red - 6) * 24 + stupac] = mojaBoja;
}

// rotira sadržaj jednog prstena za 1 korak naprijed (1) ili natrag (0)
```

```
void rotiraj(int red, bool Forward) {
  CRGB temp;
  if (Forward) {
    if (red < 6) {
      temp = ledDolje[red * 24 + 0];
      for (int j = 1; j < 24; j++) {
        ledDolje[red * 24 + j - 1] = ledDolje[red * 24 + j];
      }
      ledDolje[red * 24 + 23] = temp;
    }
    else {
      temp = ledGore[(red - 6) * 24 + 0];
      for (int j = 1; j < 24; j++) {
        ledGore[(red - 6) * 24 + j - 1] = ledGore[(red - 6) * 24 + j];
      }
      ledGore[(red - 6) * 24 + 23] = temp;
    }
  }
  else {
    if (red < 6) {
      temp = ledDolje[red * 24 + 23];
      for (int j = 22; j > -1; j--) {
        ledDolje[red * 24 + j + 1] = ledDolje[red * 24 + j];
      }
      ledDolje[red * 24 + 0] = temp;
    }
    else {
      temp = ledGore[(red - 6) * 24 + 23];
      for (int j = 22; j > -1; j--) {
        ledGore[(red - 6) * 24 + j + 1] = ledGore[(red - 6) * 24 + j];
      }
      ledGore[(red - 6) * 24 + 0] = temp;
    }
```

```
  }
}


// prikazuje za x ms i ne briše niz
void odrziIzgled(int mojeZakasnjenje) {
  FastLED.show();
  delay(mojeZakasnjenje);
}

// prikazuje za x ms, zatim briše niz
void ocistiIzgled(int mojeZakasnjenje) {
  FastLED.show();
  delay(mojeZakasnjenje);
  ocistiSve();
}

// postavlja sve LED diode na jednu nijansu za x ms bez utjecaja na niz uopće
void showAll(int mojeZakasnjenje, byte nijansa) {
  FastLED.showColor(CHSV(nijansa, 255, 255));
  delay(mojeZakasnjenje);
}

// slično kao i show All, osim što postavlja niz na jednu boju i nema vremenskog odgoda
void napuniToranj(byte nijansa) {
  for ( int i = 0; i < 144; i++) {
    ledDolje[i] = CHSV(nijansa, 255, 255);
    ledGore[i] = CHSV(nijansa, 255, 255);
  }
}

// isključiti sve LED diode
void ocistiSve() {
```

```
  FastLED.clear();  // briše niz LED dioda
  FastLED.show();
}

// isključite sve i pričekajte 1 sek.
void stanka() {
  ocistiSve();
  delay(500);
}

  // Reset je dodan za čišćenje dodjele memorije. Negdje u mom kodu, stog
   // se ne prazni, pa ponavljanje bez resetiranja na kraju dovodi do maksimalne potrošnje RAM-a!
   // Nije tako dobro kao pronalaženje problema, ali radi!
void softwareReset() {
//  wdt_enable(WDTO_60MS);
  while(1) {}
}
```

## 6. Fotografije gotovog uređaja

## 7. ODRŽAVANJE

Sve elemente i dijelove potrebno je redovito kontrolirati i održavati. U slučaju oštećenja ili nekog njegovog dijela (kućišta, konstrukcije, podloge ili drugog elementa opreme), popravak se može izvršiti zamjenom oštećenog elementa isključivo istim elementom.

## 8. FOTO ALBUM RADA NA PROJEKTU