VISOKA TEHNIČKA ŠKOLA STRUKOVNIH STUDIJA SUBOTICA



**Plane Game**

Projekat iz predmeta: Mikrokontroleri

Mentor: Mikloš Pot Studenti: David Katrinka, Stepan Turicin

Profesor strukovnih studija Studijski program: Informatika

**Sadržaj:**

[**Uvod** 2](#_Toc1)

[**Korišćene komponente** 3](#_Toc2)

[**Šema** 4](#_Toc3)

[**Kod** 5](#_Toc4)

[**Uključivanje biblioteka i deklaracija i definicija promenljivih**  5](#_Toc5)

[**Deklaracija niza za displej igre, definicija strukture za objekat u igri i deklaracija niza za objekte** 6](#_Toc6)

[**Funkcija za stvaranje novog objekta u igri** 6](#_Toc7)

[**Funkcija za brisanje celog prikaza igre** 6](#_Toc8)

[**Funkcija za prikazivanje objekata igre na LCD displeju** 7](#_Toc9)

[**Funkcija za prikaz poruke dobrodošlice** 8](#_Toc10)

[**Funkcija za prikaz poruke za kraj igre i rezultata** 8](#_Toc11)

[**Deklaracija i definicija osnovne razdaljine neprijatelja od igrača** 9](#_Toc12)

[**Stvaranje novih karaktera (za igrača, neprijatelje i srce)** 9](#_Toc13)

[**Deklaracija i definicija statistike igrača i srca za pomoć** 11](#_Toc14)

[**Definisanje objekata za srce, neprijatelje i igrača** 11](#_Toc15)

[**Funkcija za neprijatelje** 12](#_Toc16)

[**Funkcija za srce** 13](#_Toc17)

[**Setup** 14](#_Toc18)

[**Loop** 15](#_Toc19)

[**Literatura** 16](#_Toc20)

# **Uvod**

Ovim projektom realizujemo jednostavnu igru, čiji je cilj izbegavanje neprijatelja, čime se postiže bolji rezultat (engl. score).

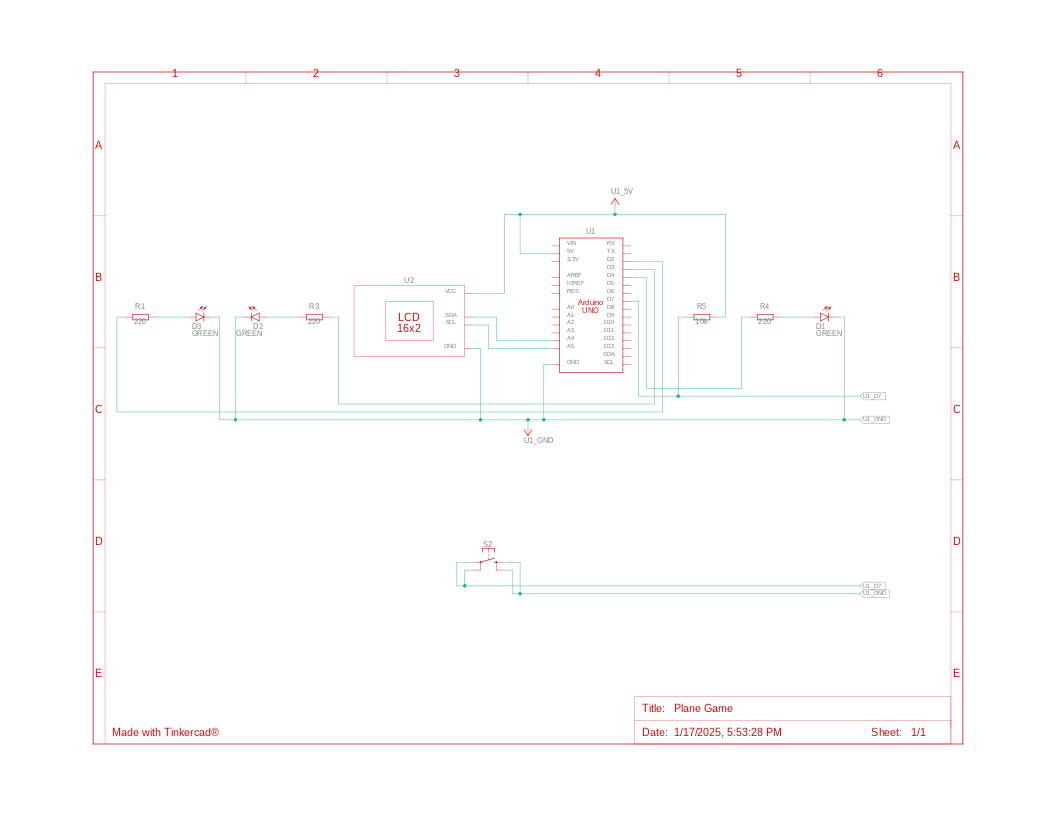
Igrač je avion koji izbegava neprijatelje, koji su vanzemaljci. Uvek se nalazi sa leve strane ekrana i ima opciju da ide dole ili gore. Pozicija se menja pritiskom dugmeta (dok nije pritisnuto igrač je u gornjem položaju, a dok je pritisnuto u donjem). Nakon što igrač izbegne neprijatelja njegov rezultat se povećava. Ako se igrač sudari sa neprijateljom on gubi jedan život, a ako izgubi sva tri života izgubio je igru. Igrač može da vrati živote sakupljajući srca koja se, isto kao neprijatelji, pojavljuju nasumično (ali su ređa nego neprijatelji). Igrač ne može da ima više od 3 života i ako sakupi srce dok već ima maksimalan broj života neće se ništa dogoditi.

Ovaj dokument služi kao dodatak projektu izrađenom uz pomoć Arduino Uno mikrokontrolera. U nastavku su opisani šema projekta i kompletan kod.

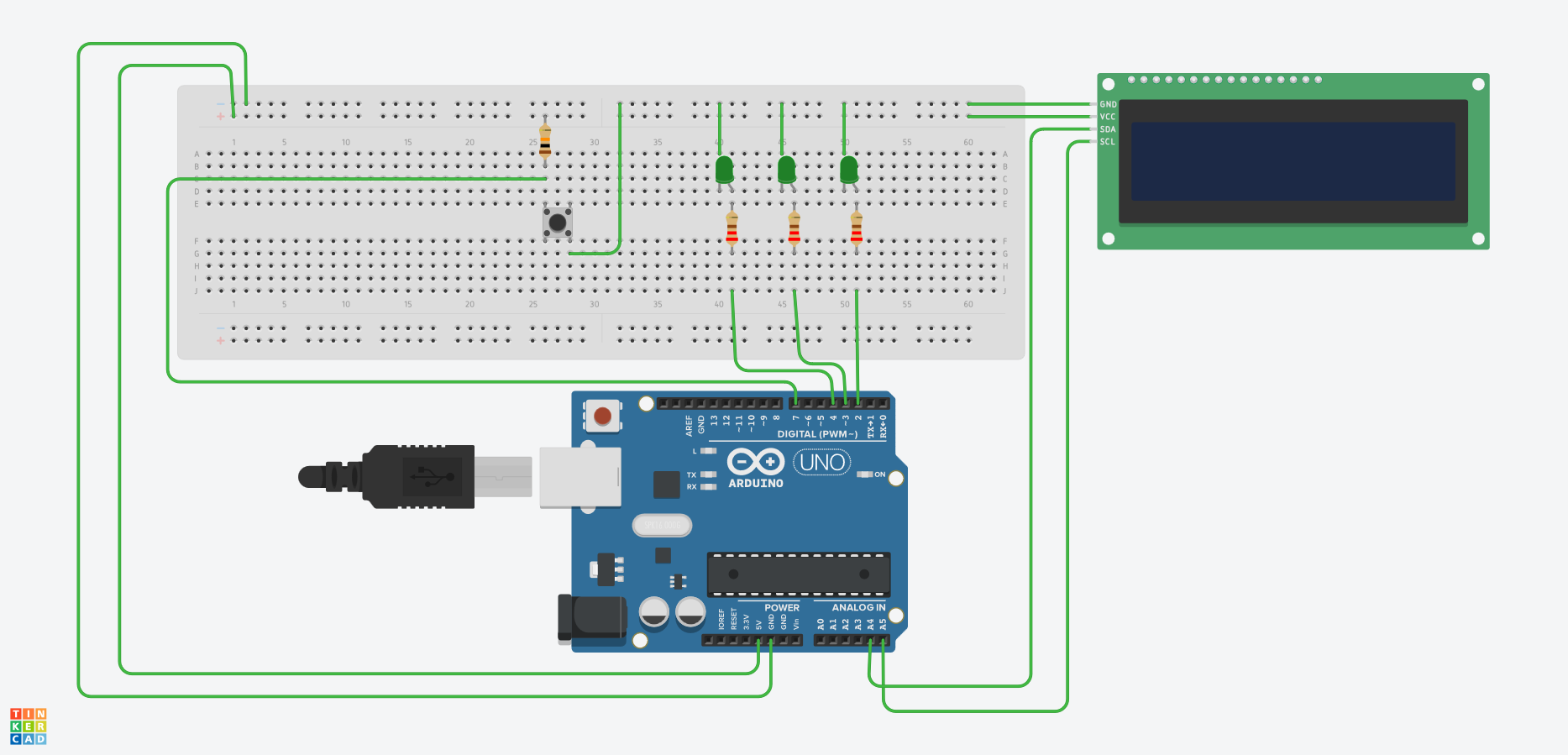
# **Korišćene komponente**

* Arduino Uno R3:   
    
  Mikrokontroler koji služi kao osnovni kontrolni sistem za upravljanje igrom, čitajući ulaze i upravljajući izlazima.
* LCD Displej (I2C):   
    
  LCD displej prikazuje rezultate i status igre. Zahvaljujući I2C interfejsu, broj potrebnih pinova za povezivanje je smanjen u odnosu na klasični LCD, što olakšava povezivanje i štedi prostor na ploči.
* 3 Zelene LED:   
    
  Svetlosni indikatori koji se koriste za vizualizaciju broja preostalih života.
* 3 Otpornika od 220 Ω :   
    
  Otpornici ograničavaju struju kroz LED kako bi ih zaštitili od oštećenja.
* Otpornik od 10 kΩ:   
    
  Koristi se kao pull-up otpornik za stabilizaciju signala sa tastera i sprečavanje lažnih ulaza.
* Taster:  
    
  Ulazna komponenta koja omogućava igraču da menja položaj aviona u igri pritiskom na dugme.
* Protobord  
    
  Komponenta koja omogućava brzo i jednostavno pravljenje prototipa električnih kola bez potrebe za lemljenjem. Koristi se za testiranje i povezivanje električnih komponenti, jer ima metalne trake koje povezuju rupe, omogućavajući lako povezivanje žica i komponenti.

# **Šema**



*Slika 1. Šematski prikaz iz Tinkercad-a*



*Slika 2. Prikaz kola iz Tinkercad-a*

# **Kod**

## **Uključivanje biblioteka i deklaracija i definicija promenljivih**

Promenljive za prikaz redova i kolona igre, maksimalan broj objekata, brzinu i ubranje igre i promenljivih za kraj igre.

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

/////////////////////////

// Game Engine

// Config

const int rowDisplay = 2;

const int columnDisplay = 32;

const int maxGameObjects = 10;

const int gameSpeedIncrease = 1;

const int minGameSpeed = 50;

int gameSpeed = 200;

int lengthGameObjects = 0;

int gameOver = 0;

char\* gameOverText = "Game Over";

int \_\_gameOverOn = 0;

## **Deklaracija niza za displej igre, definicija strukture za objekat u igri i deklaracija niza za objekte**

// LCD Display 2x16 (Game border is 2x32)

char gameDisplay[rowDisplay][columnDisplay];

// GameObject

struct GameObject {

// Position

int x;

int y;

// Render char

char render;

};

// All objects on scene

struct GameObject objects[maxGameObjects];

## **Funkcija za stvaranje novog objekta u igri**

// New game object on scene

GameObject\* create\_object(int x, int y, byte render) {

GameObject obj = {x, y, (char)render};

objects[lengthGameObjects++] = obj;

return &objects[lengthGameObjects - 1];

}

## **Funkcija za brisanje celog prikaza igre**

// gameDisplay is empty (not LCD)

void clear\_display() {

for(int r = 0; r < rowDisplay; r++)

for(int c = 0; c < columnDisplay; c++)

gameDisplay[r][c] = ' ';

}

## **Funkcija za prikazivanje objekata igre na LCD displeju**

// Render gameDisplay and objects on LCD

void render\_display() {

clear\_display();

// Place gameObjects on gameDisplay

for(int i = 0; i < lengthGameObjects; i++) {

gameDisplay[objects[i].y][objects[i].x] = objects[i].render;

}

// Set gameDisplay on LCD

lcd.setCursor(0, 0);

lcd.print(gameDisplay[0]);

lcd.setCursor(0, 1);

lcd.print(gameDisplay[1]);

lcd.setCursor(0, 0);

// delay gameSpeed

gameSpeed -= gameSpeedIncrease;

if(minGameSpeed > gameSpeed)

gameSpeed = minGameSpeed;

delay(gameSpeed);

}

## **Funkcija za prikaz poruke dobrodošlice**

// only on setup()

void welcome\_screen(char\* welcome\_row\_1, char\* welcome\_row\_2) {

lcd.setCursor(0, 0);

lcd.print(welcome\_row\_1);

lcd.setCursor(0, 1);

lcd.print(welcome\_row\_2);

lcd.setCursor(0, 0);

}

## **Funkcija za prikaz poruke za kraj igre i rezultata**

void game\_over\_screen(int score) {

// If gameOver is already on

if(\_\_gameOverOn)

return;

// gameOver is on

\_\_gameOverOn = 1;

// clear all gameObjects from gameDisplay

clear\_display();

// set cleared gameDisplay on LCD

lcd.setCursor(0, 0);

lcd.print(gameDisplay[0]);

lcd.setCursor(0, 1);

lcd.print(gameDisplay[1]);

// print gameOverText on LCD

lcd.setCursor(0, 0);

lcd.print(gameOverText);

// print score

if(score > 9999)

score = 9999;

char textScore[15];

sprintf(textScore,"Score: %d", score);

lcd.setCursor(0, 1);

lcd.print(textScore);

}

## **Deklaracija i definicija osnovne razdaljine neprijatelja od igrača**

// Distance from right side of display

int enemy1Distance = 3;

int enemy2Distance = 7;

int enemy3Distance = 12;

int enemy4Distance = 16;

## **Stvaranje novih karaktera (za igrača, neprijatelje i srce)**

// Display character for player

int byteEnemyNum = 1;

byte enemyRender[8] = {

B01110,

B11111,

B10101,

B11111,

B11111,

B11111,

B10101,

B10001

};

int bytePlayerNum = 2;

byte playerRender[8] = {

B00000,

B00000,

B00100,

B10110,

B11111,

B10110,

B00100,

B00000

};

int byteAidNum = 3;

byte aidKitRender[8] = {

B00000,

B11011,

B11111,

B11111,

B11111,

B01110,

B00100,

B00000

};

## **Deklaracija i definicija statistike igrača i srca za pomoć**

// Player Stats

const int maxPlayerHealth = 3;

int playerHealth = maxPlayerHealth;

int score = 0;

// AidKit

const int maxTicksKit = 10;

int ticksKit = maxTicksKit;

int isKitDropped = 0;

int healthLED[maxPlayerHealth] = {

2, 3, 4

};

## **Definisanje objekata za srce, neprijatelje i igrača**

// Create Kits

GameObject\* aidKit = create\_object(16, random(2), byte(byteAidNum));

// Create enemies from right side and randomize row

GameObject\* enemy1 = create\_object(columnDisplay - enemy1Distance, random(2), byte(byteEnemyNum));

GameObject\* enemy2 = create\_object(columnDisplay - enemy2Distance, random(2), byte(byteEnemyNum));

GameObject\* enemy3 = create\_object(columnDisplay - enemy3Distance, random(2), byte(byteEnemyNum));

GameObject\* enemy4 = create\_object(columnDisplay - enemy4Distance, random(2), byte(byteEnemyNum));

// Create player

GameObject\* player = create\_object(0, 0, byte(bytePlayerNum));

void enemy\_walk(GameObject\* enemy);

## **Funkcija za neprijatelje**

Hodanje neprijatelja, vraćanje neprijatelja nazad kada izađe iz prikaza i nasumično biranje njegovog reda, smanjenje života igrača ako udari u neprijatelja, dizanje zastave za kraj igre ako životi igrača dođu do 0 i uvećavanje rezultata igrača za uspešno izbegavanje neprijatelja.

void enemy\_walk(GameObject\* enemy) {

// Move

enemy->x -= 1;

// If enemy behind scene, back to right screen corner

if(enemy->x < 0) {

enemy->x = 16;

enemy->y = random(2);

}

// If an enemy bumps into a player

if(enemy->x == player->x && enemy->y == player->y) {

// Hit

playerHealth -= 1;

// Display health

digitalWrite(healthLED[playerHealth], LOW);

// Die

if(playerHealth == 0)

gameOver = 1;

}

// If player and enemy in one column (and not hit) - score++

else if (enemy->x == player->x) {

score++;

}

## **Funkcija za srce**

Pomeranje srca, vraćanje srca nazad kada izađe iz prikaza i nasumično biranje njegovog reda i povećavanje života igrača kada pokupi srce.

void dropAidKit() {

aidKit->x -= 1;

// If kit behind scene, back to right screen corner

if(aidKit->x < 0) {

aidKit->x = 16;

aidKit->y = random(2);

isKitDropped = 0;

ticksKit = maxTicksKit;

}

// Pick up kit

if(aidKit->x == player->x && aidKit->y == player->y) {

playerHealth++;

if(playerHealth > maxPlayerHealth)

playerHealth = maxPlayerHealth;

// Display health

for(int i = 0; i < playerHealth; i++) {

digitalWrite(healthLED[i], HIGH);

}

}

}

## **Setup**

Inicijalizacija displeja, pinova za LED-ove, pina za taster, seed-a za nasumične brojeve, serial monitora za debug i prikaz poruke dobrodošlice.

void setup() {

lcd.begin(16, 2);

lcd.backlight();

lcd.noCursor();

// Create char for GameObjects

lcd.createChar(byteEnemyNum, enemyRender);

lcd.createChar(bytePlayerNum, playerRender);

lcd.createChar(byteAidNum, aidKitRender);

// Health display

for(int i=0; i < playerHealth; i++) {

pinMode(healthLED[i], OUTPUT);

digitalWrite(healthLED[i], HIGH);

}

pinMode(7, INPUT);

randomSeed(analogRead(0));

Serial.begin(9600);

welcome\_screen("You are airplane!", "Avoid the UFO!");

delay(2000);

}

## **Loop**

Kontrola igrača (promena reda), hodanje neprijatelja, stvaranje srca, prikaz frejma igre i prikaz poruke za kraj igre ako je igrač izgubio.

void loop() {

// Control player (change row)

player->y = !digitalRead(7);

if(!gameOver) {

// Move enemies

enemy\_walk(enemy1);

enemy\_walk(enemy2);

enemy\_walk(enemy3);

enemy\_walk(enemy4);

// Move kit

ticksKit--;

if(ticksKit < 0)

isKitDropped = 1;

if(isKitDropped)

dropAidKit();

// Render game frame

render\_display();

}

else {

// Game end

game\_over\_screen(score);

}

}

# **Literatura**

[W1] Šeme napravljene koristeći Tinkercad - <https://www.tinkercad.com/>

[W2] I2C LCD displej - <https://projecthub.arduino.cc/arduino_uno_guy/i2c-liquid-crystal-displays-5eb615>

[W3] <https://people.vts.su.ac.rs/~pmiki/>