VISOKA TEHNIČKA ŠKOLA STRUKOVNIH STUDIJA SUBOTICA

Plane Game

Projekat iz predmeta: Mikrokontroleri

Mentor: Mikloš Pot

Profesor strukovnih studija

Studenti: David Katrinka, Stepan Turicin

Studijski program: Informatika

Subotica, 2025.

Sadržaj:

Subotica, 2025.

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.

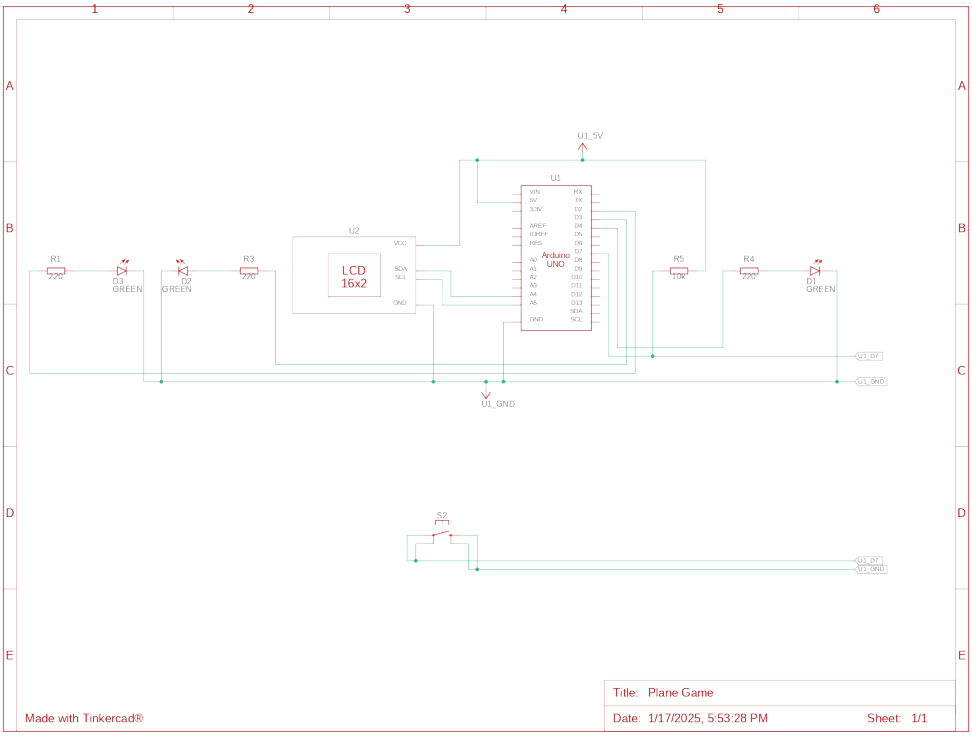
- Taster:

  Ulazna komponenta koja omogućava igraču da menja položaj aviona u igri pritiskom na dugme.

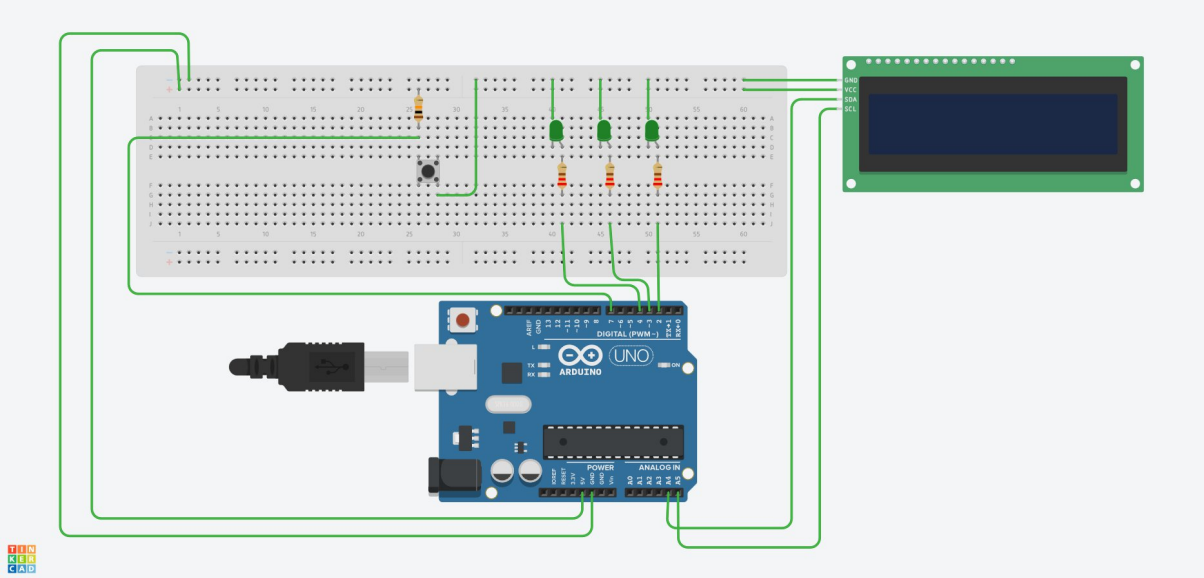
- Protobord

  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

Subotica, 2025.

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/