НП "Обучение за ИТ кариера"

Курсов проект - “Тетрис”

Изготвил: Реджеб Реджеб

Група 08

Гр.Хасково – 2024г.

GitHub:
https://github.com/RedzhebRedzheb/tetrisItKariera.git

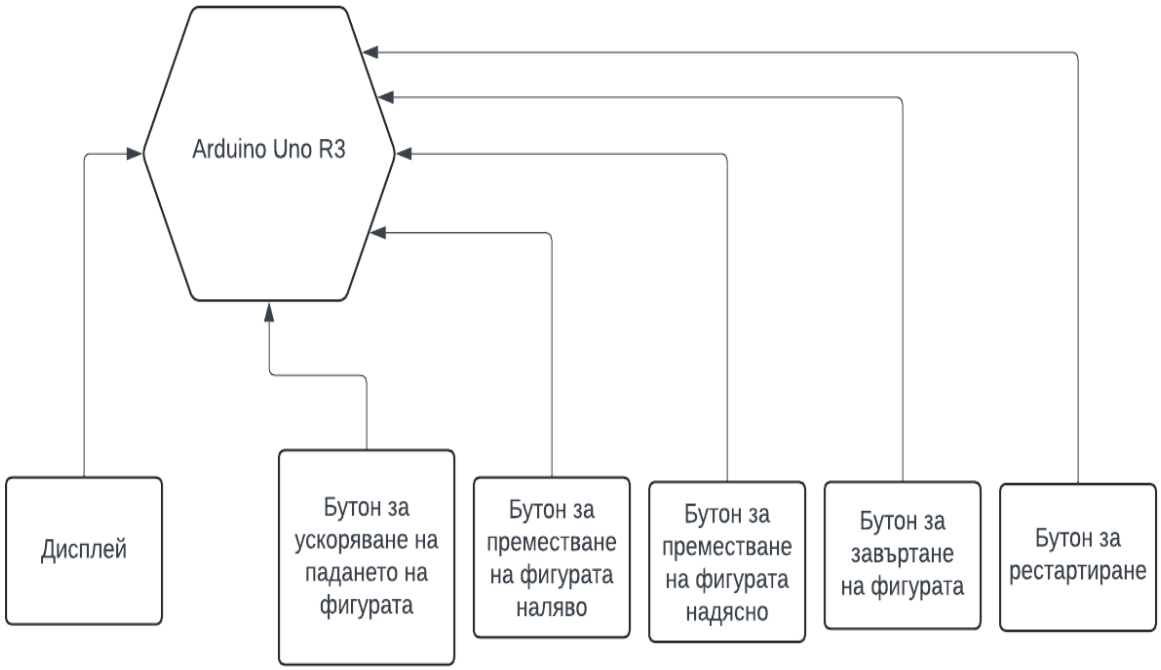
Tinkercad: https://www.tinkercad.com/things/6TPcSjbIeHz-dazzling-turing

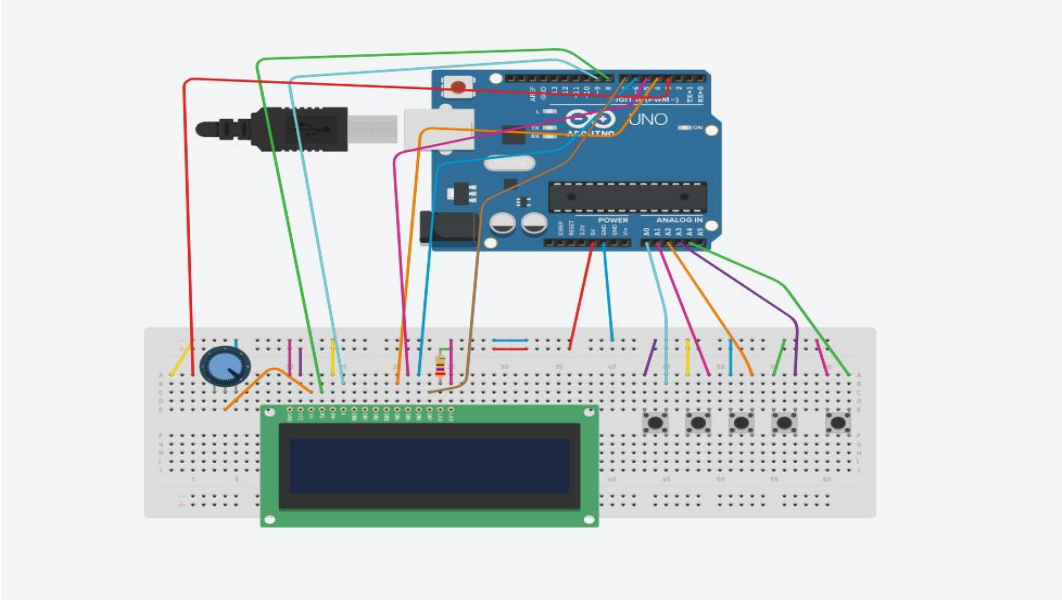
Проект - “Тетрис”

Описание на проекта:

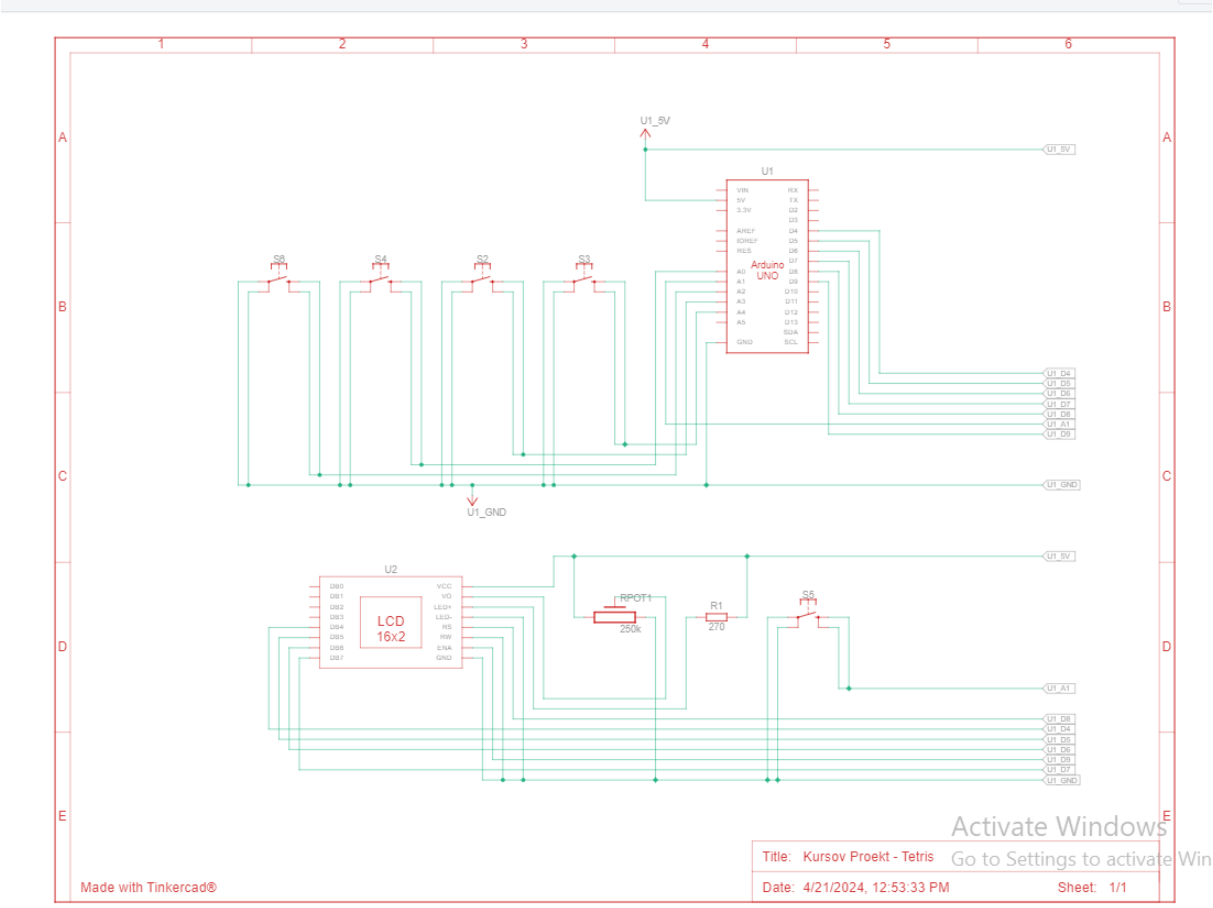
Проектът представлява игра на тетрис, реализирана за микроконтролер с използване на Arduino платформата. Играта използва LCD дисплей за визуализация и бутони за управление на движението и ротацията на падащите фигури. Допълнително, проектът включва възможност за запазване на най-висок резултат чрез използване на EEPROM памет.

Блокови схеми:





Електрическа схема:



Activate Windows
Go to Settings to activate Win

Списък със съставните части:

- Микроконтролер Arduino Uno
- Breadboard
- LCD дисплей 16x2
- 5 бутона
- Резистор 270 Ω
- Потенциометър

Сорс код:

```
#include <LiquidCrystal.h>
#include <EEPROM.h>
#define btn1      0
#define btn2      1
#define btn3      2
#define btn4      3
#define btn5      4
#define btnNone      5

#define maxShapes    3
#define maxRotations 2

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

boolean matrix[16][4];

int currentX, currentY, currentRot, currentShape, prevKey, gameSpeed, score, highScore;
unsigned long timeToMove;



byte cfull[] = {B11111, B11111, B11111, B00000, B11111, B11111, B11111, B11111};
byte ctop[] = {B11111, B11111, B11111, B11111, B00000, B00000, B00000, B00000};
byte cbottom[] = {B00000, B00000, B00000, B00000, B11111, B11111, B11111, B11111,};
byte shapes[3][2][3] = {
  {{B111, B000, B000}, {B100, B100, B100}}, //I
  {{B010, B110, B000}, {B110, B010, B000}}, //J
  {{B100, B110, B000}, {B110, B100, B000}} //L
};

void drawScreen() {
  for (int i = 0; i < 16; i++) {
    lcd.setCursor(i, 0);
    if (matrix[i][0] && matrix[i][1]) lcd.write(byte(2));
    else if (matrix[i][0]) lcd.write(byte(0));
    else if (matrix[i][1]) lcd.write(byte(1));
    else lcd.print(' ');
    lcd.setCursor(i, 1);
    if (matrix[i][2] && matrix[i][3]) lcd.write(byte(2));
    else if (matrix[i][2]) lcd.write(byte(0));
    else if (matrix[i][3]) lcd.write(byte(1));
    else lcd.print(' ');
  }
}

void newShape() {
  currentX = 1;
  currentY = 0;
  currentRot = 0;
  currentShape = rand() % maxShapes;
```

Activate Windows
Go to Settings to activate Windows.

```
}

void putShape(boolean visible, int shape, int rot, int x, int y) {
  for (int i = 0; i < 3; i++) {
    if ((shapes[shape][rot][i] >> 2) & 1) matrix[y + i][x] = visible;
    if ((shapes[shape][rot][i] >> 1) & 1) matrix[y + i][x + 1] = visible;
    if (shapes[shape][rot][i] & 1) matrix[y + i][x + 2] = visible;
  }
}

void drawShape() {
  putShape(true, currentShape, currentRot, currentX, currentY);
}

void clearShape() {
  putShape(false, currentShape, currentRot, currentX, currentY);
}

void clearDisplay() {
  for (int i = 0; i < 16; i++) {
    for (int n = 0; n < 4; n++) {
      matrix[i][n] = 0;
    }
  }
}

boolean onScreen(int x, int y) {
  if (x < 0) return false;
  if (x > 3) return false;
  if (y < 0) return false;
  if (y > 15) return false;
  return true;
}

boolean isValid(int rot, int x, int y) {
  boolean okay = true;
  for (int i = 0; ((i < 3) && (okay)); i++) {
    if ((shapes[currentShape][rot][i] >> 2) & 1) {
      if (!onScreen(x, y + i) || matrix[y + i][x]) okay = false;
    }

    if ((shapes[currentShape][rot][i] >> 1) & 1) {
      if (!onScreen(x + 1, y + i) || matrix[y + i][x + 1]) okay = false;
    }

    if (shapes[currentShape][rot][i] & 1) {
      if (!onScreen(x + 2, y + i) || matrix[y + i][x + 2]) okay = false;
    }
  }
  return okay;
```

Activate Windows
Go to Settings to activate Windows.

```
}

boolean isValid(int x, int y) {
  return isValid(currentRot, x, y);
}

boolean moveDown() {
  clearShape();
  if (isValid(currentX, currentY + 1)) {
    currentY++;
    drawShape();
    drawScreen();
    return true;
  }
  drawShape();
  return false;
}

boolean moveLeft() {
  clearShape();
  if (isValid(currentX - 1, currentY)) {
    currentX--;
    drawShape();
    drawScreen();
    return true;
  }
  drawShape();
  return false;
}

boolean moveRight() {
  clearShape();
  if (isValid(currentX + 1, currentY)) {
    currentX++;
    drawShape();
    drawScreen();
    return true;
  }
  drawShape();
  return false;
}

boolean rotate() {
  int r;
  clearShape();
  if (currentRot == maxRotations - 1) r = 0;
  else r = currentRot + 1;
  if (isValid(r, currentX, currentY)) {
    currentRot = r;
    drawShape();
    drawScreen();
    return true;
  }
  drawShape();
  return false;
}

void flashLine(int y) {
  for (int t = 0; t < 3; t++) {
    for (int i = 0; i < 4; i++) matrix[y][i] = 1;
    drawScreen();
    for (int i = 0; i < 4; i++) matrix[y][i] = 0;
    drawScreen();

  }
}

int clearLines() {
  int lineCount = 0;
  boolean tmpmatrix[16][4];
  for (int i = 0; i < 16; i++) {
    for (int n = 0; n < 4; n++) {
      tmpmatrix[i][n] = 0;
    }
  }
  int tmpY = 15;
  boolean found = false;
  for (int y = 15; y >= 0; y--) {
    if (matrix[y][0] && matrix[y][1] && matrix[y][2] && matrix[y][3]) {
      flashLine(y);
      lineCount++;
    } else {
      for (int x = 0; x < 4; x++) tmpmatrix[tmpY][x] = matrix[y][x];
      tmpY--;
    }
  }
  if (lineCount > 0) {
    for (int i = 0; i < 16; i++) {
      for (int n = 0; n < 4; n++) {
        matrix[i][n] = tmpmatrix[i][n];
      }
    }
  }
  return lineCount;
}

int getKey() {
   if (digitalRead(A0) == 0) return btn1;
   else if (digitalRead(A1) == 0) return btn2;
   else if (digitalRead(A2) == 0) return btn3;
```

Activate Windows
Go to Settings to activate Windows.

Activate Windows
Go to Settings to activate Windows.

```
   else if (digitalRead(A3) == 0) return btn4;
   else if (digitalRead(A4) == 0) return btn5;
  else return btnNone;
  delay(8);
 }

int getHighScore() {
  long two = EEPROM.read(0);
  long one = EEPROM.read(1);
  return ((two << 0) & 0xFF) + ((one << 8) & 0xFFFF);
}

void saveHighScore() {
  EEPROM.write(0, (score & 0xFF));
  EEPROM.write(1, ((score >> 8) & 0xFF));
  highScore = score;
}


void initialize() {
  clearDisplay();
  newShape();
  gameSpeed = 600;
  score = 0;
  timeToMove = millis() + gameSpeed;
  highScore = getHighScore();
}

void restart(){
    lcd.clear();
    initialize();
    lcd.setCursor(0, 0);
    lcd.print("Restarting...");
    delay(1000);
    setup();
    lcd.clear();
    drawScreen();
}


void setup() {
  pinMode(A0, INPUT);
  digitalWrite(A0, INPUT_PULLUP);
  pinMode(A1, INPUT);
  digitalWrite(A1, INPUT_PULLUP);
  pinMode(A2, INPUT);
  digitalWrite(A2, INPUT_PULLUP);
  pinMode(A3, INPUT);
  digitalWrite(A3, INPUT_PULLUP);
  pinMode(A4, INPUT);
  digitalWrite(A4, INPUT_PULLUP);
  lcd.begin(16, 2);
  lcd.createChar(0, ctop);
  lcd.createChar(1, cbottom);
  lcd.createChar(2, cfull);
  Serial.begin(9600);
  randomSeed(A1);
  initialize();
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Highscore:");
  lcd.setCursor(0, 1);
  lcd.print(highScore);
  delay(2000);
}

void loop() {
  bool gameOver = false;

  newShape();
  drawShape();
  drawScreen();

  while (!gameOver && moveDown()) {
    while (millis() < timeToMove) {
      int k = getKey();
      if (k != prevKey) {
        Serial.println(analogRead(A0));
        switch (k) {
          case btn1:
            while(moveDown());
            break;
          case btn2:
            moveLeft();
            break;
          case btn3:
            moveRight();
            break;
          case btn4:
            rotate();
            break;
          case btn5:
            if (currentY == 0) {
              restart();
              return;
            }
            break;
        }
```

Activate Windows
Go to Settings to activate Windows.

```
        prevKey = k;
      }
    }
    timeToMove = millis() + gameSpeed;
  }

  int cleared = clearLines();
  score += 10 * (cleared + 1);

  if (currentY == 0) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Game Over");
    lcd.setCursor(0, 1);
    lcd.print("Score: ");
    lcd.print(score);
    if (score > highScore) {
      saveHighScore();
    }
    gameOver = true;
  }


  while (gameOver) {
    int k = getKey();
    if (k == btn5) {
      restart();
      return;
    }
    delay(100);
  }
}
```

Activate Windows
Go to Settings to activate Windows.

Описание на функционалността:

Проектът представлява реализация на класическата игра Тетрис, която е изпълнена чрез Arduino микроконтролер, свързан с LCD дисплей и множество бутони за управление. Целта на играта е да се подреждат падащи фигури в хоризонтални редове, които се изчистват при запълването на целия ред, което увеличава резултата на играча.

Заключение:

Този проект е пример за приложение, което съчетава хардуерни и софтуерни компоненти за създаването на забавна игра. Играчите развиват своите стратегически и координационни умения, докато се състезават за подобряване на своите рекорди в класическата игра Тетрис.

Използвана литература:

-Arduino.CC

-Reddit

-SoftUni

-Tinkercad

-GitHub