

# **НП "Обучение за ИТ кариера"**

**Курсов проект - "Тетрис"**

**Изготвил: Реджеб Реджеб**

**Група 08**

**Гр.Хасково – 2024г.**

**GitHub:**

**<https://github.com/RedzhebRedzheb/tetrisItKariera.git>**

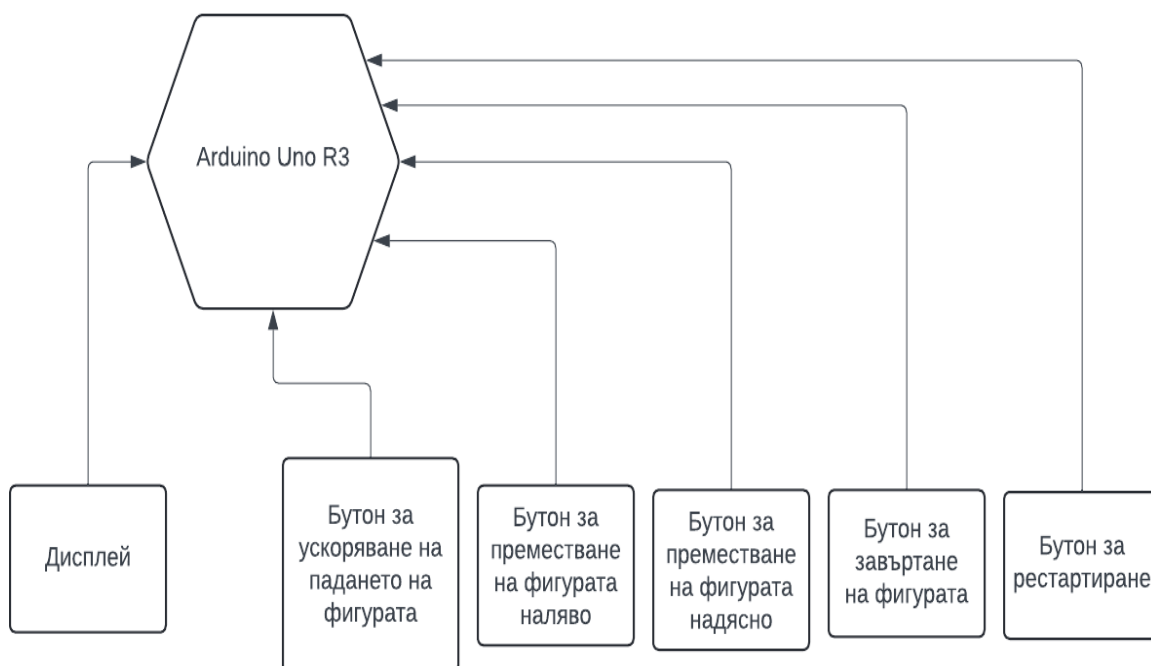
**Tinkercad: <https://www.tinkercad.com/things/6TPcSjbleHz-dazzling-turing>**

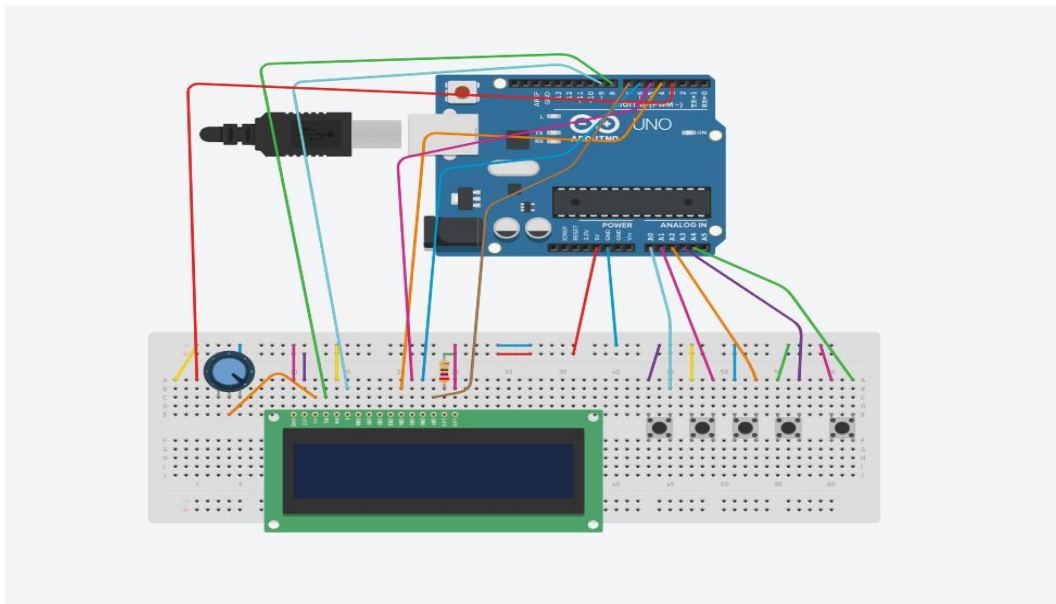
# Проект - “Тетрис”

## Описание на проекта:

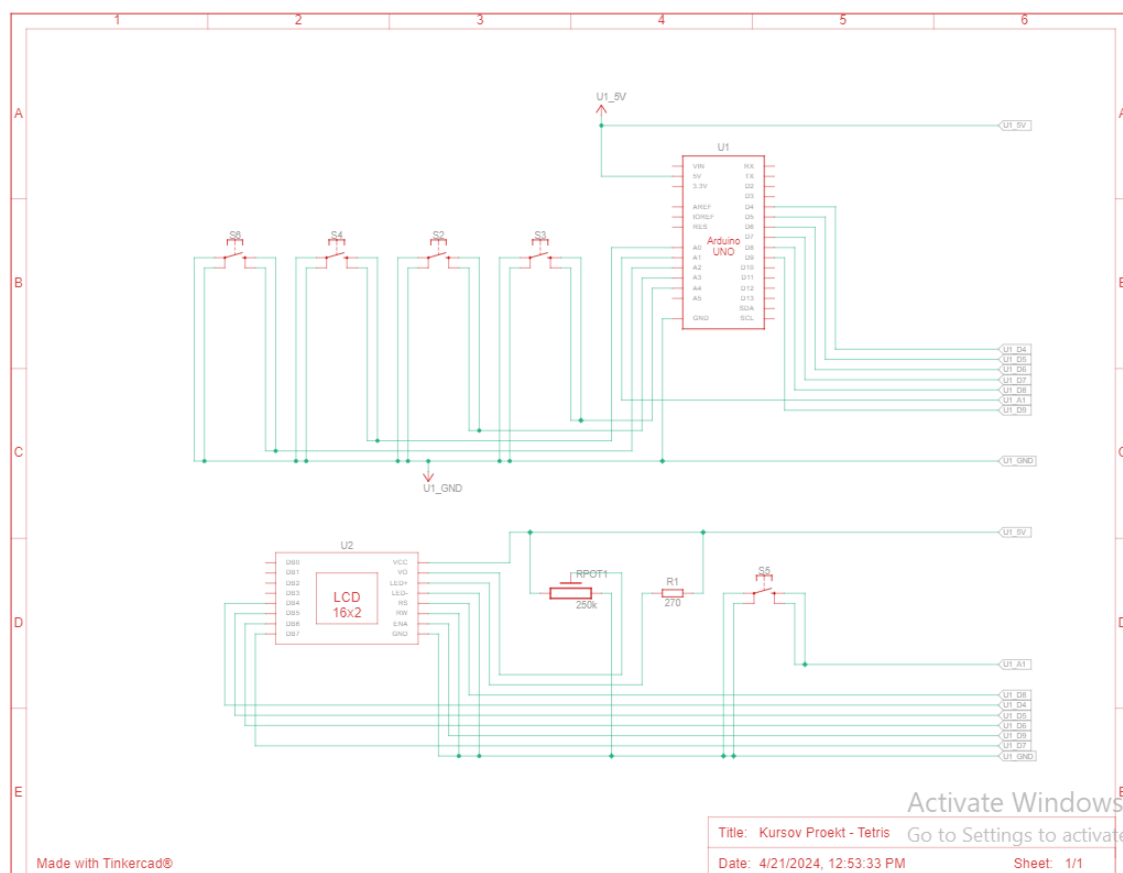
Проектът представлява игра на тетрис, реализирана за микроконтролер с използване на Arduino платформата. Играта използва LCD дисплей за визуализация и бутони за управление на движението и ротацията на падащите фигури. Допълнително, проектът включва възможност за запазване на най-висок резултат чрез използване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

## Блокови схеми:





## Електрическа схема:



## Списък със съставните части:

- Микроконтролер Arduino Uno
- Breadboard
- LCD дисплей 16x2
- 5 бутона
- Резистор 270  $\Omega$
- Потенциометър

## Сорс код:

```
1 #include <LiquidCrystal.h>
2 #include <EEPROM.h>
3 #define btn1 0
4 #define btn2 1
5 #define btn3 2
6 #define btn4 3
7 #define btn5 4
8 #define btnNone 5
9
10 #define maxShapes 3
11 #define maxRotations 2
12
13 LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
14
15 boolean matrix[16][4];
16
17 int currentX, currentY, currentRot, currentShape, prevKey, gameSpeed, score, highScore;
18 unsigned long timeToMove;
19
20
21
22 byte cFull[] = {B11111, B11111, B11111, B00000, B11111, B11111, B11111, B11111};
23 byte cTop[] = {B11111, B11111, B11111, B11111, B00000, B00000, B00000, B00000};
24 byte cBottom[] = {B00000, B00000, B00000, B00000, B11111, B11111, B11111, B11111};
25 byte shapes[3][2][3] = {
26   {{B111, B000, B000}, {B100, B100, B100}}, //I
27   {{B010, B110, B000}, {B110, B010, B000}}, //J
28   {{B100, B110, B000}, {B110, B100, B000}} //L
29 };
30
31 void drawScreen() {
32   for (int i = 0; i < 16; i++) {
33     lcd.setCursor(i, 0);
34     if (matrix[i][0] && matrix[i][1]) lcd.write(byte(2));
35     else if (matrix[i][0]) lcd.write(byte(0));
36     else if (matrix[i][1]) lcd.write(byte(1));
37     else lcd.print(' ');
38     lcd.setCursor(i, 1);
39     if (matrix[i][2] && matrix[i][3]) lcd.write(byte(2));
40     else if (matrix[i][2]) lcd.write(byte(0));
41     else if (matrix[i][3]) lcd.write(byte(1));
42     else lcd.print(' ');
43   }
44 }
45
46 void newShape() {
47   currentX = 1;
48   currentY = 0;
49   currentRot = 0;
50   currentShape = rand() % maxShapes;
```

Activate Windows  
Go to Settings to activate Windows.

```

51 }
52
53 void putShape(boolean visible, int shape, int rot, int x, int y) {
54     for (int i = 0; i < 3; i++) {
55         if ((shapes[shape][rot][i] >> 2) & 1) matrix[y + i][x] = visible;
56         if ((shapes[shape][rot][i] >> 1) & 1) matrix[y + i][x + 1] = visible;
57         if (shapes[shape][rot][i] & 1) matrix[y + i][x + 2] = visible;
58     }
59 }
60
61 void drawShape() {
62     putShape(true, currentShape, currentRot, currentX, currentY);
63 }
64
65 void clearShape() {
66     putShape(false, currentShape, currentRot, currentX, currentY);
67 }
68
69 void clearDisplay() {
70     for (int i = 0; i < 16; i++) {
71         for (int n = 0; n < 4; n++) {
72             matrix[i][n] = 0;
73         }
74     }
75 }
76
77 boolean onScreen(int x, int y) {
78     if (x < 0) return false;
79     if (x > 3) return false;
80     if (y < 0) return false;
81     if (y > 15) return false;
82     return true;
83 }
84
85 boolean isValid(int rot, int x, int y) {
86     boolean okay = true;
87     for (int i = 0; ((i < 3) && (okay)); i++) {
88         if ((shapes[currentShape][rot][i] >> 2) & 1) {
89             if (!onScreen(x, y + i) || matrix[y + i][x]) okay = false;
90         }
91
92         if ((shapes[currentShape][rot][i] >> 1) & 1) {
93             if (!onScreen(x + 1, y + i) || matrix[y + i][x + 1]) okay = false;
94         }
95
96         if (shapes[currentShape][rot][i] & 1) {
97             if (!onScreen(x + 2, y + i) || matrix[y + i][x + 2]) okay = false;
98         }
99     }
100     return okay;

```

Activate Windows  
Go to Settings to activate Windows.

```

101 }
102
103 boolean isValid(int x, int y) {
104     return isValid(currentRot, x, y);
105 }
106
107 boolean moveDown() {
108     clearShape();
109     if (isValid(currentX, currentY + 1)) {
110         currentY++;
111         drawShape();
112         drawScreen();
113         return true;
114     }
115     drawShape();
116     return false;
117 }
118
119 boolean moveLeft() {
120     clearShape();
121     if (isValid(currentX - 1, currentY)) {
122         currentX--;
123         drawShape();
124         drawScreen();
125         return true;
126     }
127     drawShape();
128     return false;
129 }
130
131 boolean moveRight() {
132     clearShape();
133     if (isValid(currentX + 1, currentY)) {
134         currentX++;
135         drawShape();
136         drawScreen();
137         return true;
138     }
139     drawShape();
140     return false;
141 }
142
143 boolean rotate() {
144     int r;
145     clearShape();
146     if (currentRot == maxRotations - 1) r = 0;
147     else r = currentRot + 1;
148     if (isValid(r, currentX, currentY)) {
149         currentRot = r;
150         drawShape();
151         drawScreen();
152         return true;
153     }
154     drawShape();
155     return false;
156 }
157
158 void flashLine(int y) {
159     for (int t = 0; t < 3; t++) {
160         for (int i = 0; i < 4; i++) matrix[y][i] = 1;
161         drawScreen();
162         for (int i = 0; i < 4; i++) matrix[y][i] = 0;
163         drawScreen();
164     }
165 }
166
167
168 int clearLines() {
169     int lineCount = 0;
170     boolean tmpmatrix[16][4];
171     for (int i = 0; i < 16; i++) {
172         for (int n = 0; n < 4; n++) {
173             tmpmatrix[i][n] = 0;
174         }
175     }
176     int tmpY = 15;
177     boolean found = false;
178     for (int y = 15; y >= 0; y--) {
179         if (matrix[y][0] && matrix[y][1] && matrix[y][2] && matrix[y][3]) {
180             flashLine(y);
181             lineCount++;
182         } else {
183             for (int x = 0; x < 4; x++) tmpmatrix[tmpY][x] = matrix[y][x];
184             tmpY--;
185         }
186     }
187     if (lineCount > 0) {
188         for (int i = 0; i < 16; i++) {
189             for (int n = 0; n < 4; n++) {
190                 matrix[i][n] = tmpmatrix[i][n];
191             }
192         }
193     }
194     return lineCount;
195 }
196
197 int getKey() {
198     if (digitalRead(A0) == 0) return btn1;
199     else if (digitalRead(A1) == 0) return btn2;
200     else if (digitalRead(A2) == 0) return btn3;
201 }

```

Activate Windows  
Go to Settings to activate Windows.

Activate Windows  
Go to Settings to activate Windows.

```

201     else if (digitalRead(A3) == 0) return btn4;
202     else if (digitalRead(A4) == 0) return btn5;
203     else return btnNone;
204     delay(8);
205 }
206
207 int getHighScore() {
208     long two = EEPROM.read(0);
209     long one = EEPROM.read(1);
210     return ((two << 8) & 0xFF) + ((one << 8) & 0xFFFF);
211 }
212
213 void saveHighScore() {
214     EEPROM.write(0, (score & 0xFF));
215     EEPROM.write(1, ((score >> 8) & 0xFF));
216     highScore = score;
217 }
218
219
220 void initialize() {
221     clearDisplay();
222     newShape();
223     gameSpeed = 600;
224     score = 0;
225     timeToMove = millis() + gameSpeed;
226     highScore = getHighScore();
227 }
228
229 void restart() {
230     lcd.clear();
231     initialize();
232     lcd.setCursor(0, 0);
233     lcd.print("Restarting...");
234     delay(1000);
235     setup();
236     lcd.clear();
237     drawScreen();
238 }
239
240
241 void setup() {
242     pinMode(A0, INPUT);
243     digitalWrite(A0, INPUT_PULLUP);
244     pinMode(A1, INPUT);
245     digitalWrite(A1, INPUT_PULLUP);
246     pinMode(A2, INPUT);
247     digitalWrite(A2, INPUT_PULLUP);
248     pinMode(A3, INPUT);
249     digitalWrite(A3, INPUT_PULLUP);
250     pinMode(A4, INPUT);
251     digitalWrite(A4, INPUT_PULLUP);
252     lcd.begin(16, 2);
253     lcd.createChar(0, ctop);
254     lcd.createChar(1, cbottom);
255     lcd.createChar(2, cfull);
256     Serial.begin(9600);
257     randomSeed(A1);
258     initialize();
259     lcd.clear();
260     lcd.setCursor(1, 0);
261     lcd.clear();
262     lcd.setCursor(0, 0);
263     lcd.print("Highscore:");
264     lcd.setCursor(0, 1);
265     lcd.print(highScore);
266     delay(2000);
267 }
268
269 void loop() {
270     bool gameOver = false;
271
272     newShape();
273     drawShape();
274     drawScreen();
275
276     while (!gameOver && moveDown()) {
277         while (millis() < timeToMove) {
278             int k = getKey();
279             if (k != prevKey) {
280                 Serial.println(analogRead(A0));
281                 switch (k) {
282                     case btn1:
283                         while (moveDown());
284                         break;
285                     case btn2:
286                         moveLeft();
287                         break;
288                     case btn3:
289                         moveRight();
290                         break;
291                     case btn4:
292                         rotate();
293                         break;
294                     case btn5:
295                         if (currentY == 0) {
296                             restart();
297                             return;
298                         }
299                         break;
300 }

```

Activate Windows  
Go to Settings to activate Windows.

Activate Windows  
Go to Settings to activate Windows.

```

301     prevKey = k;
302   }
303 }
304   timeToMove = millis() + gameSpeed;
305 }
306
307   int cleared = clearLines();
308   score += 10 * (cleared + 1);
309
310   if (currentY == 0) {
311     lcd.clear();
312     lcd.setCursor(0, 0);
313     lcd.print("Game Over");
314     lcd.setCursor(0, 1);
315     lcd.print("Score: ");
316     lcd.print(score);
317     if (score > highScore) {
318       saveHighScore();
319     }
320     gameOver = true;
321   }
322
323
324   while (gameOver) {
325     int k = getKey();
326     if (k == btn5) {
327       restart();
328       return;
329     }
330     delay(100);
331   }
332 }
333

```

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.

## Заклучение:

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