Numele proiectului: The Maze Game

Autor: Bucătariu Lavinia-Maria

Scop: Jucătorul trebuie să găsească o cale de a ieși dintr-un labirint într-un anumit număr de secunde.

Piese:

- 1 Arduino Mega
- 4 matrici de leduri 32x8
- 4 butoane pentru controlul direcțiilor
- un display numeric
- un buzzer
- 4 rezistori de 10 k-ohmi
- fire pentru conexiuni

Descrierea proiectului:

Jocul “Labirintul” a fost realizat folosind simulatorul online Wokwi. Acesta este alcătuit din 3 niveluri de dificultăți diferite, iar scopul pe care trebuie să-l atingă jucătorul este acela de a ieși din labirint în timp util (20-30 de secunde). Design-ul labirinturilor a fost făcut manual respectând câteva modele disponibile pe internet. Pereții labirintului sunt reprezentați prin leduri aprinse de care jucătorul nu poate trece, iar la finalul unui nivel se va auzi un sunet corespunzător, în funcție de câștigul nivelului (a ieșit din labirint) sau la pierderea acestuia (a expirat timpul). O rundă de joc e alcătuită din 3 niveluri de dificultate, de care jucătorul trebuie să treacă pentru a i se calcula scorul final. Dacă jucătorul nu reușește să iasă din labirintul corespunzător unui anumit nivel, acesta nu poate trece la nivelul următor.

Conexiuni:

Conexiunile sunt realizate în fișierul diagram.json:


```
{
  "version": 1,
  "author": "Bucatariu Lavinia-Maria",
  "editor": "wokwi",
```

```
  "parts": [
    { "type": "wokwi-arduino-mega", "id": "mega", "top": 77.4, "left": 92.4, "attrs":
{} },
    {
      "type": "wokwi-max7219-matrix",
      "id": "matrix1",
      "top": -105.4,
      "left": 122.96,
      "rotate": 180,
      "attrs": { "chain": "4" }
    },
    {
      "type": "wokwi-max7219-matrix",
      "id": "matrix2",
      "top": -181.8,
      "left": 122.64,
      "attrs": { "chain": "4" }
    },
    {
      "type": "wokwi-max7219-matrix",
      "id": "matrix3",
      "top": -259,
      "left": 122.96,
      "rotate": 180,
      "attrs": { "chain": "4" }
    },
    {
      "type": "wokwi-max7219-matrix",
      "id": "matrix4",
      "top": -335.4,
      "left": 122.64,
      "attrs": { "chain": "4" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 83,
      "left": 624,
      "attrs": { "color": "blue", "key": "ArrowUp" }
    },
    {
      "type": "wokwi-pushbutton",
```

```
  "id": "btn2",
  "top": 227,
  "left": 624,
  "attrs": { "color": "red", "key": "ArrowDown" }
},
{
  "type": "wokwi-pushbutton",
  "id": "btn3",
  "top": 150.2,
  "left": 547.2,
  "attrs": { "color": "yellow", "key": "ArrowLeft" }
},
{
  "type": "wokwi-pushbutton",
  "id": "btn4",
  "top": 150.2,
  "left": 700.8,
  "attrs": { "color": "green", "key": "ArrowRight" }
},
{
  "type": "wokwi-resistor",
  "id": "r1",
  "top": 90.35,
  "left": 825.6,
  "attrs": { "value": "10000" }
},
{
  "type": "wokwi-resistor",
  "id": "r2",
  "top": 157.55,
  "left": 825.6,
  "attrs": { "value": "10000" }
},
{
  "type": "wokwi-resistor",
  "id": "r3",
  "top": 195.95,
  "left": 825.6,
  "attrs": { "value": "10000" }
},
{
  "type": "wokwi-resistor",
```

```
    "id": "r4",
    "top": 234.35,
    "left": 825.6,
    "attrs": { "value": "10000" }
  },
  {
    "type": "wokwi-tm1637-7segment",
    "id": "sevseg1",
    "top": -230.44,
    "left": 535.03,
    "attrs": { "color": "red" }
  },
  {
    "type": "wokwi-buzzer",
    "id": "bz1",
    "top": -122.4,
    "left": 577.8,
    "attrs": { "volume": "0.1" }
  }
],
"connections": [
  [ "matrix1:GND", "mega:GND.2", "black", [ "h-67.2", "v288", "h209.05" ] ],
  [ "matrix1:CLK.2", "matrix2:CLK", "blue", [ "h9.44", "v-38.4" ] ],
  [ "matrix2:CS", "matrix1:CS.2", "gold", [ "h19.2", "v57.6" ] ],
  [ "matrix1:DOUT", "matrix2:DIN", "green", [ "h28.64", "v-76.8" ] ],
  [ "matrix2:GND", "matrix1:GND.2", "black", [ "h38.4", "v96" ] ],
  [ "matrix1:V+.2", "matrix2:V+", "red", [ "h47.84", "v-115.2" ] ],
  [ "matrix2:CLK.2", "matrix3:CLK", "blue", [ "h-56.74", "v-110.5" ] ],
  [ "matrix3:CS", "matrix2:CS.2", "gold", [ "h-47.8", "v89.07" ] ],
  [ "matrix2:DOUT", "matrix3:DIN", "green", [ "h-38.2", "v69.87" ] ],
  [ "matrix3:GND", "matrix2:GND.2", "black", [ "h-28.6", "v50.67" ] ],
  [ "matrix2:V+.2", "matrix3:V+", "red", [ "h-19", "v31.47" ] ],
  [ "matrix3:CLK.2", "matrix4:CLK", "blue", [ "h9.44", "v-38.4" ] ],
  [ "matrix4:CS", "matrix3:CS.2", "gold", [ "h19.2", "v57.6" ] ],
  [ "matrix3:DOUT", "matrix4:DIN", "green", [ "h28.64", "v-76.8" ] ],
  [ "matrix4:GND", "matrix3:GND.2", "black", [ "h38.4", "v96" ] ],
  [ "matrix3:V+.2", "matrix4:V+", "red", [ "h47.84", "v-115.2" ] ],
  [ "matrix1:V+", "mega:5V", "red", [ "h-57.6", "v268.8", "h192" ] ],
  [ "matrix1:DIN", "mega:11", "green", [ "h-19.2", "v48", "h134.4", "v9.6" ] ],
  [ "matrix1:CS", "mega:10", "yellow", [ "h-28.8", "v67.2", "h153.6" ] ],
  [ "matrix1:CLK", "mega:13", "blue", [ "h-38.4", "v86.4", "h134.4" ] ],
  [ "mega:4", "btn4:1.l", "green", [ "v-38.4", "h399" ] ],
```

```
    [ "mega:5", "btn2:1.l", "magenta", [ "v-28.8", "h216.5", "v172.8" ] ],
    [ "mega:6", "btn3:1.l", "yellow", [ "v-19.2", "h235.6" ] ],
    [ "mega:7", "btn1:1.l", "blue", [ "v-9.6", "h322.4" ] ],
    [ "btn1:1.r", "r1:1", "gray", [ "v0" ] ],
    [ "r2:1", "btn4:1.r", "gray", [ "v0" ] ],
    [ "r3:1", "btn3:1.r", "gray", [ "v0", "h-172.8", "v-19.2" ] ],
    [ "r4:1", "btn2:1.r", "gray", [ "v0" ] ],
    [ "r1:2", "r2:2", "black", [ "v0" ] ],
    [ "r2:2", "r3:2", "black", [ "v0" ] ],
    [ "r4:2", "r3:2", "black", [ "v0" ] ],
    [ "r4:2", "mega:GND.3", "black", [ "v86.4", "h-606" ] ],
    [ "btn3:2.r", "btn1:2.r", "red", [ "h0" ] ],
    [ "btn1:2.r", "btn4:2.r", "red", [ "h0" ] ],
    [ "btn4:2.r", "btn2:2.r", "red", [ "h0.2", "v77" ] ],
    [ "btn2:2.r", "mega:5V", "red", [ "h0.2", "v96.2", "h-432" ] ],
    [ "sevseg1:DIO", "mega:19", "limegreen", [ "h0", "v201.6", "h-297.6" ] ],
    [ "sevseg1:CLK", "mega:20", "magenta", [ "h19.2", "v230.4", "h-307.2" ] ],
    [ "sevseg1:GND", "bz1:1", "black", [ "v48", "h-163.2", "v86.4" ] ],
    [ "bz1:2", "mega:21", "red", [ "v28.8", "h-184.9" ] ]
  ],
  "dependencies": {}
}
```


Am folosit 4 matrici de leduri de 32x8 pe care le-am conectat între ele și respectiv la placa Arduino Mega la pinii digitali 10, 11, 13 și la pinul GND. Am realizat conexiunea și între matricea de leduri a întrecului joc și butoanele pe care le va folosi jucătorul pentru a-și controla mișcările prin labirint, prin intermediul pinul 5V. Butoanele au fost conectate la pinii 4, 5, 6 și 7, iar buzzer-ul și display-ul numeric la pinii 19, 20 și 21.

Codul arduino (sketch.ino):

```
#include <TM1637Display.h>
#define CLK_TM1637 20
#define DIO 19
#define CLK 13
#define DIN 11
#define CS  10
#define X_SEGMENTS   4
#define Y_SEGMENTS   4
#define NUM_SEGMENTS (X_SEGMENTS * Y_SEGMENTS)
```

```
// a framebuffer to hold the state of the entire matrix of LEDs
// laid out in raster order, with (0, 0) at the top-left
byte fb[8 * NUM_SEGMENTS];
int two_power = 1;
int current_segment = 1;
int end_segment = 1;
int next_level = 1;
int current_level = 1;
int levels_won = 0;
int score_per_level = 0;
int total_score = 0;

TM1637Display display(CLK_TM1637, DIO);

unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 1000;  // Update period in milliseconds
int timeLeft; // Remaining time in seconds


void shiftAll(byte send_to_address, byte send_this_data)
{
  digitalWrite(CS, LOW);
  for (int i = 0; i < NUM_SEGMENTS; i++) {
    shiftOut(DIN, CLK, MSBFIRST, send_to_address);
    shiftOut(DIN, CLK, MSBFIRST, send_this_data);
  }
  digitalWrite(CS, HIGH);
}

void setup() {

  Serial.begin(115200);
  pinMode(CLK, OUTPUT);
  pinMode(DIN, OUTPUT);
  pinMode(CS, OUTPUT);
  pinMode(4, INPUT); // butonul pentru dreapta
  pinMode(5, INPUT); // butonul pentru jos
  pinMode(6, INPUT); // butonul pentru stânga
  pinMode(7, INPUT); // butonul pentru sus
```

```
  display.setBrightness(0x0f);
  display.clear();

  // Setup each MAX7219
  shiftAll(0x0f, 0x00); //display test register - test mode off
  shiftAll(0x0b, 0x07); //scan limit register - display digits 0 thru 7
  shiftAll(0x0c, 0x01); //shutdown register - normal operation
  shiftAll(0x0a, 0x0f); //intensity register - max brightness
  shiftAll(0x09, 0x00); //decode mode register - No decode
}

void show() {
  for (byte row = 0; row < 8; row++) {
    digitalWrite(CS, LOW);
    byte segment = NUM_SEGMENTS;
    while (segment--) {
      byte x = segment % X_SEGMENTS;
      byte y = segment / X_SEGMENTS * 8;
      byte addr = (row + y) * X_SEGMENTS;

      if (segment & X_SEGMENTS) {
        shiftOut(DIN, CLK, MSBFIRST, 8 - row);
        shiftOut(DIN, CLK, LSBFIRST, fb[addr + x]);
      } else { // even rows of segments
        shiftOut(DIN, CLK, MSBFIRST, 1 + row);
        shiftOut(DIN, CLK, MSBFIRST, fb[addr - x + X_SEGMENTS - 1]);
      }
    }
    digitalWrite(CS, HIGH);
  }
}

// design realizat de mine după un model de labirint
void first_level()
{
  for(int i = 0; i < 8 * NUM_SEGMENTS; i++) fb[i] = 0;
  fb[127]=B11111111; fb[126]=B11110000; fb[125]=B11111111;
  fb[124]=B11111111; fb[3]=B11111111; fb[2]=B11111111;
  fb[1]=B10000111; fb[0]=B11111111;

  //primul dreptunghi 32x8
  fb[123]=B10000000; fb[119]=B10000000; fb[115]=B10000000;
```

fb[111]=B10000000; fb[107]=B10001111; fb[103]=B10000000;
fb[99]=B10000000; fb[95]=B10000000; fb[91]=B10000000;
fb[87]=B10001111; fb[83]=B10000000; fb[79]=B10000000;
fb[75]=B10000000; fb[71]=B10000000; fb[67]=B11111111;
fb[63]=B10000000; fb[59]=B10000000; fb[55]=B10000000;
fb[51]=B10000000; fb[47]=B10000000; fb[43]=B10000100;
fb[39]=B10000100; fb[35]=B10000100; fb[31]=B10000100;
fb[27]=B10000100; fb[23]=B10000111; fb[19]=B10000000;
fb[15]=B10000000; fb[11]=B10000000; fb[7]=B10000000;

//al doilea dreptunghi 32x8
fb[122]=B00010000; fb[118]=B00010000; fb[114]=B00010000;
fb[110]=B00010000; fb[106]=B11110000; fb[102]=B00000000;
fb[98]=B00000000; fb[94]=B00000000; fb[90]=B00000000;
fb[86]=B11111111; fb[82]=B00000000; fb[78]=B00000000;
fb[74]=B00000000; fb[70]=B00000000; fb[66]=B11110000;
fb[62]=B00010000; fb[58]=B00010000; fb[54]=B00010000;
fb[50]=B00010000; fb[46]=B00010000; fb[42]=B00010000;
fb[38]=B00000000; fb[34]=B00000000; fb[30]=B00000000;
fb[26]=B00000000; fb[22]=B11111111; fb[18]=B00010000;
fb[14]=B00010000; fb[10]=B00010000; fb[6]=B00010000;

//al treilea dreptunghi 32x8
fb[121]=B10000000; fb[117]=B10000000; fb[113]=B10000000;
fb[109]=B10000000; fb[105]=B10000111; fb[101]=B10000100;
fb[97]=B10000100; fb[93]=B10000100; fb[89]=B10000100;
fb[85]=B10000111; fb[81]=B00000000; fb[77]=B00000000;
fb[73]=B00000000; fb[69]=B00000000; fb[65]=B11111111;
fb[61]=B10000000; fb[57]=B10000000; fb[53]=B10000000;
fb[49]=B10000000; fb[45]=B10000000; fb[41]=B10000111;
fb[37]=B10000000; fb[33]=B10000000; fb[29]=B10000000;
fb[25]=B10000000; fb[21]=B11111111; fb[17]=B00000000;
fb[13]=B00000000; fb[9]=B00000000; fb[5]=B00000000;

//ultimul dreptunghi 32x8
fb[120]=B00000001; fb[116]=B00000001; fb[112]=B00000001;
fb[108]=B00000001; fb[104]=B11110001; fb[100]=B00000001;
fb[96]=B00000001; fb[92]=B00000001; fb[88]=B00000001;
fb[84]=B11111111; fb[80]=B00000001; fb[76]=B00000001;
fb[72]=B00000001; fb[68]=B00000001; fb[64]=B11110001;
fb[60]=B00000001; fb[56]=B00000001; fb[52]=B00000001;
fb[48]=B00000001; fb[44]=B00000001; fb[40]=B11111111;

```
  fb[36]=B00000001; fb[32]=B00000001; fb[28]=B00000001;
  fb[24]=B00000001; fb[20]=B11110001; fb[16]=B00000001;
  fb[12]=B00000001; fb[8]=B00000001; fb[4]=B00000001;
}

// design realizat de mine după un model de labirint
void second_level()
{
  for(int i = 0; i < 8 * NUM_SEGMENTS; i++) fb[i] = 0;
  for(int i=0; i<4; i++) fb[i]=255;
  for(int i=124; i<128; i++) fb[i]=255;

  //primul dreptunghi 32x8
  fb[123]=B10000000; fb[119]=B10000000; fb[115]=B10000000;
  fb[111]=B10001111; fb[107]=B10000001; fb[103]=B10000001;
  fb[99]=B10000001; fb[95]=B11110001; fb[91]=B10010001;
  fb[87]=B10010000; fb[83]=B10010000; fb[79]=B10010000;
  fb[75]=B10011111; fb[71]=B10000000; fb[67]=B10000000;
  fb[63]=B10000000; fb[59]=B10001111; fb[55]=B00000001;
  fb[51]=B00000001; fb[47]=B00000001; fb[43]=B11110001;
  fb[39]=B10010000; fb[35]=B10010000; fb[31]=B10010000;
  fb[27]=B10011111; fb[23]=B10000000; fb[19]=B10000000;
  fb[15]=B11111111; fb[11]=B10000000; fb[7]=B10000000;

  //al doilea dreptunghi 32x8
  fb[122]=B00000001; fb[118]=B00000001; fb[114]=B00000001;
  fb[110]=B11110001; fb[106]=B00010001; fb[102]=B00010000;
  fb[98]=B00010000; fb[94]=B00010000; fb[90]=B00011111;
  fb[86]=B00000001; fb[82]=B00000001; fb[78]=B00000001;
  fb[74]=B11110001; fb[70]=B00000001; fb[66]=B00000001;
  fb[62]=B00000001; fb[58]=B11111111; fb[54]=B00000000;
  fb[50]=B00000000; fb[46]=B00000000; fb[42]=B00001111;
  fb[38]=B00000001; fb[34]=B00000001; fb[30]=B00000001;
  fb[26]=B11110001; fb[22]=B00010001; fb[18]=B00010001;
  fb[14]=B00010001; fb[10]=B00000001; fb[6]=B00000001;

  //al treilea dreptunghi 32x8
  fb[121]=B00000000; fb[117]=B00000000; fb[113]=B00000000;
  fb[109]=B00011111; fb[105]=B00010000; fb[101]=B00010000;
  fb[97]=B00010000; fb[93]=B00010001; fb[89]=B11110001;
  fb[85]=B00000001; fb[81]=B00000001; fb[77]=B00000001;
  fb[73]=B00011111; fb[69]=B00000001; fb[65]=B00000001;
```

```
  fb[61]=B00000001; fb[57]=B11110001; fb[53]=B00010000;
  fb[49]=B00010000; fb[45]=B00010000; fb[41]=B11110001;
  fb[37]=B00010001; fb[33]=B00010001; fb[29]=B00010001;
  fb[25]=B00000000; fb[21]=B00000000; fb[17]=B00000000;
  fb[13]=B11110001; fb[9]=B00000000; fb[5]=B00000000;

  //ultimul dreptunghi 32x8
  fb[120]=B00000001; fb[116]=B00000001; fb[112]=B00000001;
  fb[108]=B11110001; fb[104]=B00000001; fb[100]=B00000001;
  fb[96]=B00000001; fb[92]=B11110001; fb[88]=B00000001;
  fb[84]=B00000001; fb[80]=B00000001; fb[76]=B00011111;
  fb[72]=B00010000; fb[68]=B00010000; fb[64]=B00010000;
  fb[60]=B00010001;fb[56]=B11110001; fb[52]=B00000001;
  fb[48]=B00000001; fb[44]=B00000001; fb[40]=B11111111;
  fb[36]=B00000001; fb[32]=B00000001; fb[28]=B11110001;
  fb[24]=B00000001; fb[20]=B00000001; fb[16]=B00000001;
  fb[12]=B11111111; fb[8]=B00000001; fb[4]=B00000001;
}

// design realizat de mine după un model de labirint
// pe acest nivel l-am realizat prima dată, dar ulterior mi-am dat seama că pot
utiliza
// reprezentarea binară a numerelor cu ajutorul cărora am realizat pereții
labirintului
void third_level()
{
  for(int i = 0; i < 8 * NUM_SEGMENTS; i++) fb[i] = 0;
  for(int i=0; i<4; i++) fb[i]=255;
  for(int i=125; i<127; i++) fb[i]=255;

  //primul dreptunghi 32x8
  fb[127]=241; fb[123]=144; fb[119]=144; fb[115]=144; fb[111]=129;
  fb[107]=129; fb[103]=129; fb[99]=143; fb[95]=136; fb[91]=136;
  fb[87]=136; fb[83]=143; fb[79]=129; fb[75]=129; fb[71]=129;
  fb[67]=241; fb[63]=129; fb[59]=129; fb[55]=241; fb[51]=129;
  fb[47]=129; fb[43]=143; fb[39]=128; fb[35]=128;
  fb[31]=128; fb[27]=255; fb[23]=129; fb[19]=143;
  fb[15]=128; fb[11]=128; fb[7]=128;

  //al doilea dreptunghi 32x8
  fb[110]=241; fb[106]=1; fb[102]=1; fb[98]=1; fb[94]=63;
  fb[90]=32; fb[86]=32; fb[82]=224; fb[78]=7; fb[66]=63;
```

```
  fb[62]=32; fb[58]=32; fb[54]=36; fb[50]=4; fb[46]=4;
  fb[42]=252; fb[38]=4; fb[34]=4; fb[30]=4; fb[26]=36;
  fb[22]=36; fb[18]=36; fb[14]=39; fb[10]=32; fb[6]=32;

  //al treilea dreptunghi 32x8
  fb[109]=31; fb[93]=255; fb[89]=1; fb[85]=1; fb[81]=1;
  fb[77]=241; fb[73]=17; fb[69]=17; fb[65]=145; fb[61]=145;
  fb[57]=145; fb[53]=144; fb[49]=144; fb[45]=144; fb[41]=255;
  fb[25]=31; fb[21]=16; fb[17]=16; fb[13]=241; fb[9]=1; fb[5]=1;

   //ultimul dreptunghi 32x8
  fb[4]=17; fb[8]=17; fb[12]=17; fb[16]=17; fb[20]=17;
  fb[24]=241; fb[40]=255; fb[44]=17; fb[48]=17; fb[52]=17;
  fb[56]=17; fb[60]=17; fb[64]=17; fb[68]=1; fb[72]=1;
  fb[76]=1; fb[80]=241; fb[84]=17; fb[88]=17; fb[92]=17;
  fb[96]=1; fb[100]=1; fb[104]=1; fb[108]=255; fb[112]=1;
  fb[116]=1; fb[120]=1; fb[124]=255;
}

void winning_sound()
{
  tone(21, 392); delay(80); noTone(21);
  tone(21, 440); delay(80); noTone(21);
  tone(21, 494); delay(80); noTone(21);
  tone(21, 523); delay(250); noTone(21);
}

void losing_sound()
{
  tone(21, 392, 200); delay(200);
  tone(21, 330, 200); delay(200);
  tone(21, 262, 400);
}

void updatePlayerPosition() {
  int new_segment = current_segment;
  int new_two_power = two_power;

  if (digitalRead(4) == HIGH) { // Dreapta
    if (two_power > 1) new_two_power /= 2;
    else {
      new_two_power = 128;
```

```
      new_segment--;
    }
  }

  if (digitalRead(6) == HIGH) { // Stânga
    if (two_power < 128) new_two_power *= 2;
    else {
      new_two_power = 1;
      new_segment++;
    }
  }

  if (digitalRead(5) == HIGH) { // Jos
    new_segment -= 4;
  }

  if (digitalRead(7) == HIGH) { // Sus
    new_segment += 4;
  }

  // verifică dacă LED-ul nou este stins înainte de a muta jucătorul
  if ((fb[new_segment] & new_two_power) == 0 && (new_segment <= 127) &&
(new_segment >= 0)) {
    // pentru a actualizează poziția jucătorului
    two_power = new_two_power;
    current_segment = new_segment;
    fb[current_segment] |= two_power; // se face un sau pe biți pentru a reprezenta
                                          //atât jucătorul cât și pereții labirintului de
care
                                         //acesta este înconjurat
  }

  if(current_level == 1)
  {
    if(new_segment > end_segment)
      {
        winning_sound();
        Serial.print("Ai castigat! Scor: " + String(score_per_level) + " puncte");
        Serial.print("\n\n");
        levels_won++;
        next_level = 1;
        current_level++;
```

```
      total_score += score_per_level;
    }
  }

  if(current_level == 2)
  {
    if((current_segment == end_segment || current_segment == (end_segment -
4) || current_segment == (end_segment - 8)) && new_two_power == 1)
    {
      winning_sound();
      Serial.print("Ai castigat! Scor: " + String(score_per_level) + " puncte");
      Serial.print("\n\n");
      levels_won++;
      next_level = 1;
      current_level++;
      total_score += score_per_level;
    }
  } else
  if(current_level == 3)
  {
    if(new_segment > end_segment)
      {
        winning_sound();
        Serial.print("Ai castigat! Scor: " + String(score_per_level) + " puncte");
        Serial.print("\n\n");
        levels_won++;
        next_level = 1;
        current_level++;
        total_score += score_per_level;
      }
  }
}

void displayTime(int time) {
  int displayMinutes = time / 60;
  int displaySeconds = time % 60;

  uint16_t displayValue = displayMinutes * 100 + displaySeconds;
  display.showNumberDecEx(displayValue, 0x80, true, 4, 0);
}

void smiley_face()
```

```
{
  for(int i = 0; i < 8 * NUM_SEGMENTS; i++) fb[i] = 0;
    fb[37]=B00111100;
    fb[45]=B10000001;
    fb[41]=B01000010;
    fb[76]=B10000000;
    fb[77]=B00100100;
    fb[78]=B00000001;

    fb[72]=B10000000;
    fb[73]=B00000000;
    fb[74]=B00000001;

    fb[68]=B10000000;
    fb[69]=B00000000;
    fb[70]=B00000001;

    fb[64]=B10000000;
    fb[65]=B00000000;
    fb[66]=B00000001;

    fb[60]=B10000000;
    fb[61]=B00100100;
    fb[62]=B00000001;

    fb[56]=B10000000;
    fb[57]=B00100100;
    fb[58]=B00000001;

    fb[52]=B10000000;
    fb[53]=B00011000;
    fb[54]=B00000001;

    fb[85]=B10000001;
    fb[89]=B01000010;
    fb[93]=B00111100;
}

void loop() {

  if (current_level == 1)
    {
```

```
    if(next_level == 1) // după ce trec la un nou nivel fac inițializările
    {
      Serial.println("Nivelul 1:");
      for(int i = 0; i < 8 * NUM_SEGMENTS; i++)
          fb[i] = 0;
          two_power = 16;
          current_segment = 1;
          end_segment = 127;
          next_level = 0;
          startMillis = millis();
          timeLeft = 31;
          score_per_level = 31;
    }
    else // sunt deja la nivelul curent, doar mă joc
    {
      first_level();
      updatePlayerPosition();

      currentMillis = millis();
      if (currentMillis - startMillis >= period) {
          startMillis = currentMillis;

          if (timeLeft > 0) {
            timeLeft--;
            score_per_level--;
          }
          else{
            losing_sound();
            Serial.println("Ai pierdut! Mai încearcă o dată!");
            next_level = 1;
            score_per_level = 0;
          }

          displayTime(timeLeft);
          }
    }
  }

if (current_level == 2)
  {
    if(next_level == 1)
    {
```

```
      Serial.println("Nivelul 2:");
      for(int i = 0; i < 8 * NUM_SEGMENTS; i++)
          fb[i] = 0;
          two_power = 1;
          current_segment = 68;
          end_segment = 55;
          next_level = 0;
          startMillis = millis();
          timeLeft = 31;
          score_per_level = 31;
    }
    else
    {
      second_level();
      updatePlayerPosition();
      currentMillis = millis();
      if (currentMillis - startMillis >= period) {
        startMillis = currentMillis;

      if (timeLeft > 0) {
        timeLeft--;
        score_per_level--;
      }
      else{
        losing_sound();
        Serial.println("Ai pierdut! Mai încearcă o dată!");
        next_level = 1;
        score_per_level = 0;}

        displayTime(timeLeft);
      }
    }
  }

if(current_level == 3)
{
    if(next_level == 1)
    {
      Serial.println("Nivelul 3:");
      for(int i = 0; i < 8 * NUM_SEGMENTS; i++)
          fb[i] = 0;
          two_power = 1;
```

```
          current_segment = 32;
          end_segment = 128;
          next_level = 0;
          startMillis = millis();
          timeLeft = 31;
          score_per_level = 31;
      }
      else
      {
       third_level();
       updatePlayerPosition();
       currentMillis = millis();
       if (currentMillis - startMillis >= period) {
         startMillis = currentMillis;

       if (timeLeft > 0) {
         timeLeft--;
         score_per_level--;
       }
       else
       {
         losing_sound();
         Serial.println("Ai pierdut! Mai încearcă odată!");
         next_level = 1;
         score_per_level = 0;
       }

       displayTime(timeLeft);
      }
    }
  }

  if(current_level == 4)
  {
    if(levels_won == 0)
    Serial.print("Nu ai castigat niciun nivel :(");
    if(levels_won == 1)
    Serial.print("Felicitari, ai castigat " + String(levels_won) + " nivel din 3! Scor
total: " + String(total_score) + " de puncte");
    if(levels_won > 1)
    Serial.print("Felicitari, ai castigat " + String(levels_won) + " niveluri din 3! Scor
total: " + String(total_score) + " de puncte");
```

```
    current_level = 5;
  }

  if(current_level == 5)
    smiley_face();

  show();
}
```

Funcții importante folosite în cod:
- design-ul nivelurilor: acestea sunt realizate manual în funcțiile first_level(), second_level() și third_level(), inițializând corespunzător cele 128 de segmente ale matricei de leduri (fiecare segment este de fapt un număr reprezentat pe 8 biți)

- updatarea jucătorului pe matricea de leduri este realizată în funcția updatePlayerPosition() în următorul mod: se verifică inițial dacă poziția pe care jucătorul vrea să se deplaseze este liberă (jucătorul fiind reprezentat de un led aprins, nu se poate deplasa pe un led deja aprins și nici nu va putea trece peste el). Această verificare va fi realizată folosind un ȘI pe biți între segemntul pe care se află jucătorul și segmentul care conține poziția lui viitoare, și doar dacă rezultatul va fi 0 jucătorul va putea înainta. Dacă poate înainta se va face un SAU pe biți pentru a reprezenta atât noua poziție a jucătorului, cât și pereții labirintului.

- în funcția loop() se realizează logica întregului joc astfel: folosesc 2 variabile, current_level și next_level, pentru a controla afisarea corespunzătoare unui anumit nivel. Variabila current_level îmi indică nivelul curent la care se află jucătorul, iar variabila next_level îmi indică momentul când se trece la nivelul următor, pentru a putea face inițializările aferente nivelului doar o singură dată.

- timpul pe care îl are jucătorul pentru a putea ieși din labirint e reprezentat în funcția „displayTime” și pentru a o implementa m-am folosit de librăria „TM1637Display.h”. Buzzer-ul este activat dacă jucătorul va ieși din labirint, acesta oferind pentru câteva secunde un suntet de victorie, sau atunci când va expira timpul, având setat un sunet de înfrângere.

- în monitorul serial se va afișa evoluția jucătorului pe parcursul rundei curente, fiind afișate mesaje corespunzătoare: faptul că jucătorul a câștigat și punctele aferente sau faptul că a pierdut și trebuie să o ia de la capăt cu nivelul curent.