Pacman - Badea Sebastian-Mihail

Autor: Badea Sebastian-Mihail
Grupa: 341C4
Îndrumător: Florin Stancu

Introducere

Proiectul consta in realizarea jocului Pacman controlat din butoane cu afisajul pe un ecran LCD. Tu esti protagonistul, Pacman, care trebuie sa obtina un scor cat mai mare fara a se lasa atacat de inamici. Scopul jocului este de a te relaxa si cred ca este util pentru mine intrucat ma va ajuta sa inteleg cat mai bine ceea ce presupune un astfel de proiect.

Descriere generală

Pentru deplasarea personajului se vor folosi 4 butoane. Personajul va avea 3 vieti la fiecare joc care vor fi reprezentate pe cele 3 led-uri (verde – reprezinta o viata ok, rosu inseamna ca s-a pierdut viata). La finalul jocului este afisat scorul si se mentioneaza daca acesta este un highscore. Jocul se termina cand protagonistul pierde toate cele 3 vieti.

Hardware Design

Lista de componente:

- OLED SSD1306
- 5 x push button
- Arduino UNO
- 3 x LED
- rezistente
- fire
- buzzer

Cum am legat pinii: OLED SSD1306:

- VCC la 3.3V
- GND la GND
- SDA la pinul digital A4

- SCL la pinul digital A5

LED-urile:

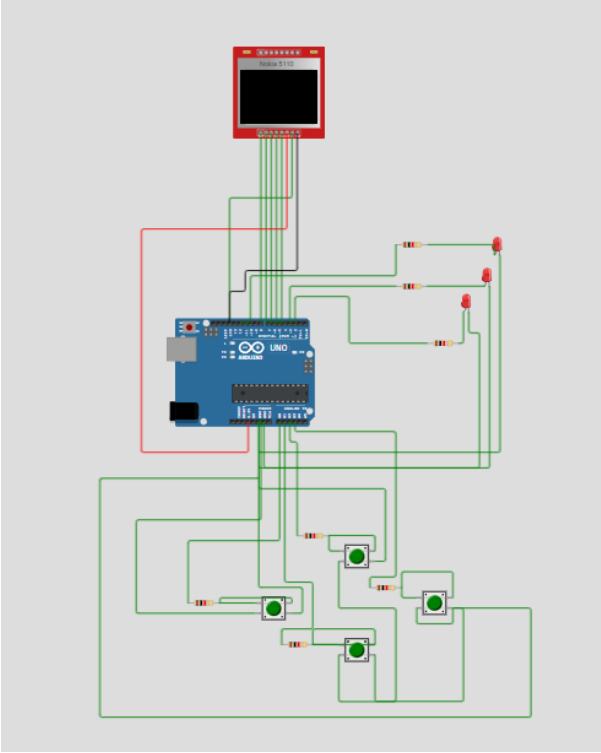
- LED1 anod la digital 3, catod la GND
- LED2 anod la digital 5, catod la GND
- LED3 anod la digital 6, catod la GND

Butoanele:

- Buton1 un pin la digital 9, celalalt pin la GND
- Buton2 un pin la digital 10, celalalt pin la GND
- Buton3 un pin la digital 11, celalalt pin la GND
- Buton4 un pin la digital 12, celalalt pin la GND
- Buton5 un pin la digital 8, celalalt pin la GND

Buzzer:

- Buzzer plus la digital 7, minus la GND



Software Design

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define UP_BUTTON 9
#define DOWN_BUTTON 10
#define LEFT_BUTTON 11
#define RIGHT_BUTTON 12
#define PAUSE_BUTTON 8

#define BUZZER_PIN 7

#define LIFE_LED_1 3
#define LIFE_LED_2 5
#define LIFE_LED_3 6

int pacmanX = 10;
int pacmanY = 10;
const int pacmanRadius = 3;

int lives = 3;
int score = 0;

struct Wall {
  int x, y, w, h;
};

#define NUM_WALLS 12
Wall walls[NUM_WALLS] = {
  {0, 0, 128, 5},
  {0, 59, 128, 5},
  {0, 0, 5, 64},
  {123, 0, 5, 64},
  {20, 15, 5, 35},
  {40, 5, 5, 15},
  {40, 45, 5, 20},
  {60, 15, 5, 35},
  {80, 5, 5, 15},
  {80, 45, 5, 20},
  {100, 15, 5, 35},
  {20, 30, 85, 5}
};

#define DOT_SPACING 10
bool dots[SCREEN_WIDTH / DOT_SPACING][SCREEN_HEIGHT / DOT_SPACING];

struct Ghost {
  int x, y;
  int dx, dy;
  int stepCount;
  int radius;
};
```

```
#define NUM_GHOSTS 3
Ghost ghosts[NUM_GHOSTS];

bool isValidPosition(int x, int y, int radius) {
  for (int i = 0; i < NUM_WALLS; i++) {
    if (x + radius > walls[i].x && x - radius < walls[i].x + walls[i].w &&
        y + radius > walls[i].y && y - radius < walls[i].y + walls[i].h) {
      return false;
    }
  }
  return true;
}

bool checkDotCollision(int x, int y) {
  int cx = x / DOT_SPACING;
  int cy = y / DOT_SPACING;
  if (cx >= 0 && cx < SCREEN_WIDTH / DOT_SPACING && cy >= 0 && cy <
SCREEN_HEIGHT / DOT_SPACING) {
    if (dots[cx][cy]) {
      dots[cx][cy] = false;
      score += 10;
      tone(BUZZER_PIN, 1000, 100);
    }
  }
}

void updateLEDs() {
  if (lives >= 1) analogWrite(LIFE_LED_1, 255); else analogWrite(LIFE_LED_1,
0);
  if (lives >= 2) analogWrite(LIFE_LED_2, 180); else analogWrite(LIFE_LED_2,
0);
  if (lives >= 3) analogWrite(LIFE_LED_3, 100); else analogWrite(LIFE_LED_3,
0);
}

void fadeOutLed(int pin) {
  for (int brightness = 255; brightness >= 0; brightness -= 1) {
    analogWrite(pin, brightness);
    delay(2);
  }
  analogWrite(pin, 0);
}

void loseLife() {
  if (lives > 0) {
    if (lives == 3) fadeOutLed(LIFE_LED_3);
    else if (lives == 2) fadeOutLed(LIFE_LED_2);
    else if (lives == 1) fadeOutLed(LIFE_LED_1);

    lives--;
```

```
    updateLEDs();
    tone(BUZZER_PIN, 500, 300);
  }
}

void respawnGhost(Ghost &g) {
  int x, y;
  do {
    x = random(10, SCREEN_WIDTH - 10);
    y = random(10, SCREEN_HEIGHT - 10);
  } while (!isValidPosition(x, y, g.radius));
  g.x = x;
  g.y = y;
  g.stepCount = 0;
  g.dx = 0;
  g.dy = 0;
}

void moveGhost(Ghost &g) {
  if (g.stepCount <= 0) {
    int possibleDirs[3] = {-1, 0, 1};
    g.dx = possibleDirs[random(3)];
    g.dy = possibleDirs[random(3)];
    g.stepCount = random(10, 30);
  }

  if (random(100) < 20) {
    if (pacmanX > g.x) g.dx = 1;
    else if (pacmanX < g.x) g.dx = -1;
    else g.dx = 0;

    if (pacmanY > g.y) g.dy = 1;
    else if (pacmanY < g.y) g.dy = -1;
    else g.dy = 0;
  }

  int newX = g.x + g.dx;
  int newY = g.y + g.dy;

  if (isValidPosition(newX, newY, g.radius) &&
      newX - g.radius >= 0 && newX + g.radius <= SCREEN_WIDTH &&
      newY - g.radius >= 0 && newY + g.radius <= SCREEN_HEIGHT) {
    g.x = newX;
    g.y = newY;
  } else {
    g.stepCount = 0;
  }

  g.stepCount--;
}
```

```
bool checkGhostCollision(Ghost &g) {
  int dx = pacmanX - g.x;
  int dy = pacmanY - g.y;
  int distSq = dx * dx + dy * dy;
  int radiiSum = pacmanRadius + g.radius;
  return distSq <= radiiSum * radiiSum;
}

void drawGhost(int x, int y, int radius) {
  display.fillCircle(x, y - radius / 2, radius / 1.5, SSD1306_WHITE);
  display.fillRect(x - radius, y - radius / 2, radius * 2, radius,
SSD1306_WHITE);
  for (int i = -radius; i < radius; i += radius / 3) {
    display.fillTriangle(x + i, y + radius, x + i + radius / 3 / 2, y +
radius - 3, x + i + radius / 3, y + radius, SSD1306_WHITE);
  }
  display.fillRect(x - radius / 2, y - radius, radius / 3, radius / 3,
SSD1306_BLACK);
  display.fillRect(x + radius / 6, y - radius, radius / 3, radius / 3,
SSD1306_BLACK);
}

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(A0));

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  display.clearDisplay();
  display.display();

  pinMode(UP_BUTTON, INPUT_PULLUP);
  pinMode(DOWN_BUTTON, INPUT_PULLUP);
  pinMode(LEFT_BUTTON, INPUT_PULLUP);
  pinMode(RIGHT_BUTTON, INPUT_PULLUP);
  pinMode(PAUSE_BUTTON, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);

  pinMode(LIFE_LED_1, OUTPUT);
  pinMode(LIFE_LED_2, OUTPUT);
  pinMode(LIFE_LED_3, OUTPUT);

  updateLEDs();

  for (int i = 0; i < NUM_GHOSTS; i++) {
    ghosts[i].radius = 3;
    respawnGhost(ghosts[i]);
  }
```

```
  for (int x = 0; x < SCREEN_WIDTH / DOT_SPACING; x++) {
    for (int y = 0; y < SCREEN_HEIGHT / DOT_SPACING; y++) {
      if (isValidPosition(x * DOT_SPACING, y * DOT_SPACING, pacmanRadius)) {
        dots[x][y] = true;
      }
    }
  }
}

void loop() {
  if (digitalRead(PAUSE_BUTTON) == LOW) {
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(20, 30);
    display.print(F("SCOR: "));
    display.print(score);
    display.display();
    delay(500);
    return;
  }

  if (lives <= 0) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(20, 25);
    display.println(F("GAME OVER"));
    display.display();
    delay(2000);

    lives = 3;
    pacmanX = 10;
    pacmanY = 10;
    updateLEDs();
    score = 0;

    for (int i = 0; i < NUM_GHOSTS; i++) {
      respawnGhost(ghosts[i]);
    }

    for (int x = 0; x < SCREEN_WIDTH / DOT_SPACING; x++) {
      for (int y = 0; y < SCREEN_HEIGHT / DOT_SPACING; y++) {
        if (isValidPosition(x * DOT_SPACING, y * DOT_SPACING, pacmanRadius))
{
          dots[x][y] = true;
        }
      }
    }
    return;
  }
```

```
  int newX = pacmanX;
  int newY = pacmanY;

  if (digitalRead(UP_BUTTON) == LOW) newY -= 2;
  if (digitalRead(DOWN_BUTTON) == LOW) newY += 2;
  if (digitalRead(LEFT_BUTTON) == LOW) newX -= 2;
  if (digitalRead(RIGHT_BUTTON) == LOW) newX += 2;

  if (newX - pacmanRadius < 0) newX = pacmanRadius;
  if (newX + pacmanRadius > SCREEN_WIDTH) newX = SCREEN_WIDTH - pacmanRadius;
  if (newY - pacmanRadius < 0) newY = pacmanRadius;
  if (newY + pacmanRadius > SCREEN_HEIGHT) newY = SCREEN_HEIGHT -
pacmanRadius;

  if (isValidPosition(newX, newY, pacmanRadius)) {
    pacmanX = newX;
    pacmanY = newY;
    checkDotCollision(pacmanX, pacmanY);
  }

  for (int i = 0; i < NUM_GHOSTS; i++) {
    moveGhost(ghosts[i]);
    if (checkGhostCollision(ghosts[i])) {
      loseLife();
      respawnGhost(ghosts[i]);
    }
  }

  display.clearDisplay();

  for (int i = 0; i < NUM_WALLS; i++) {
    display.fillRect(walls[i].x, walls[i].y, walls[i].w, walls[i].h,
SSD1306_WHITE);
  }

  for (int x = 0; x < SCREEN_WIDTH / DOT_SPACING; x++) {
    for (int y = 0; y < SCREEN_HEIGHT / DOT_SPACING; y++) {
      if (dots[x][y]) {
        display.drawPixel(x * DOT_SPACING, y * DOT_SPACING, SSD1306_WHITE);
      }
    }
  }

  display.fillCircle(pacmanX, pacmanY, pacmanRadius, SSD1306_WHITE);

  for (int i = 0; i < NUM_GHOSTS; i++) {
    drawGhost(ghosts[i].x, ghosts[i].y, ghosts[i].radius);
  }

  display.setTextSize(1);
```

```
  display.setCursor(100, 0);
  display.print(score);

  display.display();
  delay(50);
}
```

Librarii si surse 3rd-party utilizate

- **Adafruit_GFX** - pentru desenarea formelor grafice.
- **Adafruit_SSD1306** - pentru controlul afisajului OLED SSD1306.

Concepte din laboratoare utilizate in proiect

- **I2C** - utilizat pentru comunicarea cu display-ul OLED SSD1306
- **PWM** - folosit pentru controlul intensitatii LED-urilor care indica vietile jucatorului
- **GPIO** - utilizat pentru gestionarea butoanelor de control (sus, jos, stanga, dreapta, pauza)

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF