Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ПРОЕКТНА РОБОТА

з дисципліни «Комп’ютерна електроніка»

**на тему «Game Console»**

ВИКОНАЛИ:

студенти ІІ курсу

ФІОТ групи ІВ-81, ІО-81

Гакман Д.В.

Дрозд С.В.

Юхимчук Я.М.

ПЕРЕВІРИВ:

Виноградов Ю.М.

Київ 2020 р.

**Зміст**

[1. Опис проекту 3](#_Toc41234533)

[2. Схема проекту 3](#_Toc41234534)

[3. Бюджетний план 3](#_Toc41234535)

[4. Вихідні коди 3](#_Toc41234536)

[5. Опис процесу роботи 12](#_Toc41234537)

[6. Висновки 12](#_Toc41234538)

# 

# 1. Опис проекту

# 2. Схема проекту

# 3. Бюджетний план

# 4. Вихідні коди

**microPong.ino:**

#define GAME\_SPEED 100 // стандартний період руху кульки

#define II\_SPEED 250 // стандартний період руху суперника

#define RANDOM\_BOUNCE 1 // відбивати кульку у випадковому напрямку

// піни

#define BTN\_UP 3

#define BTN\_DWN 4

// інші налаштування

#define X\_PLAYER\_1 7

#define X\_PLAYER\_2 55

#define RACKET\_LEN 5

int8\_t ballPos[2];

int8\_t ballSpeed[2];

uint32\_t ballTimer, enemyTimer;

boolean btnFlag1, btnFlag2;

int8\_t racketPos1, racketPos2 = 0;

int8\_t prevRacketPos1, prevRacketPos2 = 0;

byte count1, count2;

byte speedIncr = 0, iiIncr = 0;

#include <Wire.h>

void setup() {

pinMode(BTN\_UP, INPUT\_PULLUP);

pinMode(BTN\_DWN, INPUT\_PULLUP);

Wire.begin();

oledInit();

oledClear();

digit(0, 0, 0);

digit(0, 20, 0);

randomSeed(analogRead(0));

newRound();

redrawRacket();

redrawRacket2();

}

void loop() {

ballRoutine();

buttonTick();

enemyTick();

}

void enemyTick() {

if (millis() - enemyTimer >= (II\_SPEED - iiIncr)) {

enemyTimer = millis();

if (racketPos2 + RACKET\_LEN / 2 > ballPos[1]) racketPos2--;

else racketPos2++;

racketPos2 = constrain(racketPos2, 0, 16 - RACKET\_LEN);

redrawRacket2();

}

}

void buttonTick() {

if (!digitalRead(BTN\_DWN) && !btnFlag1) {

btnFlag1 = true;

racketPos1 += 2;

if (racketPos1 > (16 - RACKET\_LEN)) racketPos1 = (16 - RACKET\_LEN);

redrawRacket();

}

if (digitalRead(BTN\_DWN) && btnFlag1) {

btnFlag1 = false;

}

if (!digitalRead(BTN\_UP) && !btnFlag2) {

btnFlag2 = true;

racketPos1 -= 2;

if (racketPos1 < 0) racketPos1 = 0;

redrawRacket();

}

if (digitalRead(BTN\_UP) && btnFlag2) {

btnFlag2 = false;

}

}

void redrawRacket() {

for (byte i = prevRacketPos1; i < prevRacketPos1 + RACKET\_LEN; i++) {

dotClear(X\_PLAYER\_1, i);

}

for (byte i = racketPos1; i < racketPos1 + RACKET\_LEN; i++) {

dotSet(X\_PLAYER\_1, i);

}

prevRacketPos1 = racketPos1;

}

void redrawRacket2() {

for (byte i = prevRacketPos2; i < prevRacketPos2 + RACKET\_LEN; i++) {

dotClear(X\_PLAYER\_2, i);

}

for (byte i = racketPos2; i < racketPos2 + RACKET\_LEN; i++) {

dotSet(X\_PLAYER\_2, i);

}

prevRacketPos2 = racketPos2;

}

void ballRoutine() {

if (millis() - ballTimer >= (GAME\_SPEED - speedIncr)) {

ballTimer = millis();

int8\_t prevPos[2];

for (byte i = 0; i < 2; i++) {

prevPos[i] = ballPos[i];

ballPos[i] += ballSpeed[i];

}

if (ballPos[0] < X\_PLAYER\_1) {

if (!(prevPos[1] >= racketPos1 && prevPos[1] <= (racketPos1 + RACKET\_LEN))) {

count2++;

if (count2 > 9) {

digit(count2 / 10, 20, 0);

digit(count2 % 10, 20, 1);

} else {

digit(count2, 20, 0);

}

delay(1000);

newRound();

dotClear(prevPos[0], prevPos[1]);

return;

} else {

ballPos[0] = prevPos[0];

ballSpeed[0] = -ballSpeed[0];

if (RANDOM\_BOUNCE) ballSpeed[1] \*= (random(0, 2)) ? 1 : -1;

}

}

if (ballPos[1] < 0) {

ballPos[1] = -ballPos[1];

ballSpeed[1] = -ballSpeed[1];

}

if (ballPos[0] > X\_PLAYER\_2) {

if (!(prevPos[1] >= racketPos2 && prevPos[1] <= (racketPos2 + RACKET\_LEN))) {

count1++;

if (count1 > 9) {

digit(count1 / 10, 0, 0);

digit(count1 % 10, 0, 1);

} else {

digit(count1, 0, 0);

}

delay(1000);

newRound();

dotClear(prevPos[0], prevPos[1]);

return;

} else {

ballPos[0] = prevPos[0];

ballSpeed[0] = -ballSpeed[0];

if (RANDOM\_BOUNCE) ballSpeed[1] \*= (random(0, 2)) ? 1 : -1;

}

}

if (ballPos[1] > 15) {

ballPos[1] = 15;

ballSpeed[1] = -ballSpeed[1];

}

dotClear(prevPos[0], prevPos[1]);

dotSet(ballPos[0], ballPos[1]);

}

}

void newRound() {

randomSeed(millis());

ballPos[0] = X\_PLAYER\_1 + 1;

racketPos1 = random(0, 16 - RACKET\_LEN);

ballPos[1] = racketPos1 + RACKET\_LEN / 2;

ballSpeed[0] = 2;

ballSpeed[1] = (random(0, 2)) ? 1 : -1;

//racketPos2 = 8;

redrawRacket();

redrawRacket2();

if (count1 >= 10) {

speedIncr = 10;

iiIncr = 70;

}

if (count1 >= 20) {

speedIncr = 25;

iiIncr = 150;

}

if (count1 >= 30) {

speedIncr = 40;

iiIncr = 190;

}

}

**oled.ino:**

#define OLED\_ADDRESS 0x3C

#define OLED\_COMMAND\_MODE 0x00

#define OLED\_ONE\_COMMAND\_MODE 0x80

#define OLED\_DATA\_MODE 0x40

#define OLED\_ONE\_DATA\_MODE 0xC0

const int bufsize = 128 \* 32 / 8 / 2;

byte oled\_buf[bufsize];

byte Scale = 1;

const uint8\_t \_oled\_init[] PROGMEM = {

0xAE, // oled off

0xD5, // CLOCK\_DIV\_RATIO

0x80,

0xA8, // Set multiplex

0x1F, // for 32 rows

0x8D, // Charge pump

0x14,

0x20, // Memory mode

0x01, // Vertical OLED\_ADDRESSing

0xA1, // Flip horizontally

0xC8, // Flip vertically

0x81, // Set contrast

0xCF, // brighter

0xDB, // Set vcom detect

0x40, // brighter

0xDA,

0x02,

0x20, // address

0x0, // col to row

0xAF, // Display on\*

};

const uint8\_t CharMap[][6] PROGMEM = {

{ 0x3E, 0x51, 0x49, 0x45, 0x3E, 0x00 }, // 30

{ 0x00, 0x42, 0x7F, 0x40, 0x00, 0x00 },

{ 0x72, 0x49, 0x49, 0x49, 0x46, 0x00 },

{ 0x21, 0x41, 0x49, 0x4D, 0x33, 0x00 },

{ 0x18, 0x14, 0x12, 0x7F, 0x10, 0x00 },

{ 0x27, 0x45, 0x45, 0x45, 0x39, 0x00 },

{ 0x3C, 0x4A, 0x49, 0x49, 0x31, 0x00 },

{ 0x41, 0x21, 0x11, 0x09, 0x07, 0x00 },

{ 0x36, 0x49, 0x49, 0x49, 0x36, 0x00 },

{ 0x46, 0x49, 0x49, 0x29, 0x1E, 0x00 },

};

void oledInit() {

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_COMMAND\_MODE);

for (uint8\_t i = 0; i < 20; i++) Wire.write(pgm\_read\_byte(&\_oled\_init[i]));

Wire.endTransmission();

}

void oledClear() {

for (int i = 0; i < bufsize; i++) {

oled\_buf[i] = 0;

}

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_COMMAND\_MODE);

Wire.write(0x21); Wire.write(0); Wire.write(127);

Wire.write(0x22); Wire.write(0); Wire.write(3);

Wire.endTransmission();

for (int i = 0; i < 32; i++) {

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_DATA\_MODE);

for (int i = 0; i < 32; i++) Wire.write(0);

Wire.endTransmission();

}

}

void dot(byte x, byte y, boolean state) {

byte byteY;

if (y < 4) byteY = y;

else if (y < 8) byteY = y - 4;

else if (y < 12) byteY = y - 8;

else byteY = y - 12;

byte index = x + y / 4 \* 64;

byte thisByte = oled\_buf[index];

bitWrite(thisByte, byteY \* 2, state);

bitWrite(thisByte, byteY \* 2 + 1, state);

oled\_buf[index] = thisByte;

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_COMMAND\_MODE);

Wire.write(0x21); Wire.write(x \* 2); Wire.write(127);

Wire.write(0x22); Wire.write(y / 4); Wire.write(3);

Wire.endTransmission();

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_DATA\_MODE);

Wire.write(thisByte); Wire.write(thisByte);

Wire.endTransmission();

Wire.beginTransmission(OLED\_ADDRESS);

}

void dotSet(byte x, byte y) {

dot(x, y, 1);

}

void dotClear(byte x, byte y) {

dot(x, y, 0);

}

void line(byte x0, byte y0, byte x1, byte y1) {

if (x0 == x1) {

for (byte i = y0; i < y1; i++) dotSet(x0, i);

} else if (y0 == y1) {

for (byte i = x0; i < x1; i++) dotSet(i, y0);

} else {

int sx, sy, e2, err;

int dx = abs(x1 - x0);

int dy = abs(y1 - y0);

if (x0 < x1) sx = 1; else sx = -1;

if (y0 < y1) sy = 1; else sy = -1;

err = dx - dy;

for (;;) {

dotSet(x0, y0);

if (x0 == x1 && y0 == y1) return;

e2 = err << 1;

if (e2 > -dy) {

err = err - dy;

x0 = x0 + sx;

}

if (e2 < dx) {

err = err + dx;

y0 = y0 + sy;

}

}

}

}

void digit(int c, int column, int line) {

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_COMMAND\_MODE);

// Set column address range

Wire.write(0x21); Wire.write(column \* 6); Wire.write(column \* 6 + Scale \* 6 - 1);

// Set page address range

Wire.write(0x22); Wire.write(line); Wire.write(line + Scale - 1);

Wire.endTransmission();

Wire.beginTransmission(OLED\_ADDRESS);

Wire.write(OLED\_DATA\_MODE);

for (uint8\_t col = 0 ; col < 6; col++) {

int bits = pgm\_read\_byte(&CharMap[c][col]);

if (Scale == 1) Wire.write(bits);

else {

bits = Stretch(bits);

for (int i = 2; i--;) {

Wire.write(bits);

Wire.write(bits >> 8);

}

}

}

Wire.endTransmission();

}

int Stretch (int x) {

x = (x & 0xF0) << 4 | (x & 0x0F);

x = (x << 2 | x) & 0x3333;

x = (x << 1 | x) & 0x5555;

return x | x << 1;

}

# 5. Опис процесу роботи

# 6. Висновки