Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 15.03.06 Мехатроника и робототехника

**О Т Ч Е Т**

**по учебной практике**

Выполнил студент гр. ПРТ-21-1б

Пикулев Максим Александрович

(фамилия, имя, отчество)

###### 

(подпись)

Проверил:

ст. преподаватель Д.А. Карлов

(должность, Ф.И.О. руководителя по практической подготовке от кафедры)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**Пермь 2022**

**СОДЕРЖАНИЕ**

Введение………………………………………………………………………3

Основная часть………………………………………………………………..4

Заключение……………………………………………………………………19

Список использованных источников………………………………………..20

**Введение**

В рамках нашей практики, проведенной в период с 16 июня по 7 июля, мы решили сосредоточиться на исследовании и усовершенствовании игрового приложения на arduino.

Цель исследования: анализ существующего кода и его улучшение, а также создание устройства для игры.

Задачи исследования:

1) Изучение проблемы исходного проекта для выявления потенциала улучшений.

2) Составление списка комплектующих и необходимых материалов для сборки игровой консоли.

3) Сборка игровой консоли на основе Arduino или другой подходящей платформы.

4) Разработка и внедрение инновационных элементов, улучшающих игровую механику и обогащающих игровой процесс.

5) Создание новых анимаций и текстур для придания игре более привлекательного и уникального вида.

**Основная часть**

Подготовка комплектующих: Составили список необходимых компонентов и материалов для сборки игровой приставки на основе Arduino.  
Список комплектующих:  
1) Arduino UNO R3 (рис.1)  
2) Кабель USB 2.0 (A-B) (рис.2)  
3) Джойстик двухосевой (рис.3)  
4) SPI/I2C 1602 LCD yellow (рис.4)

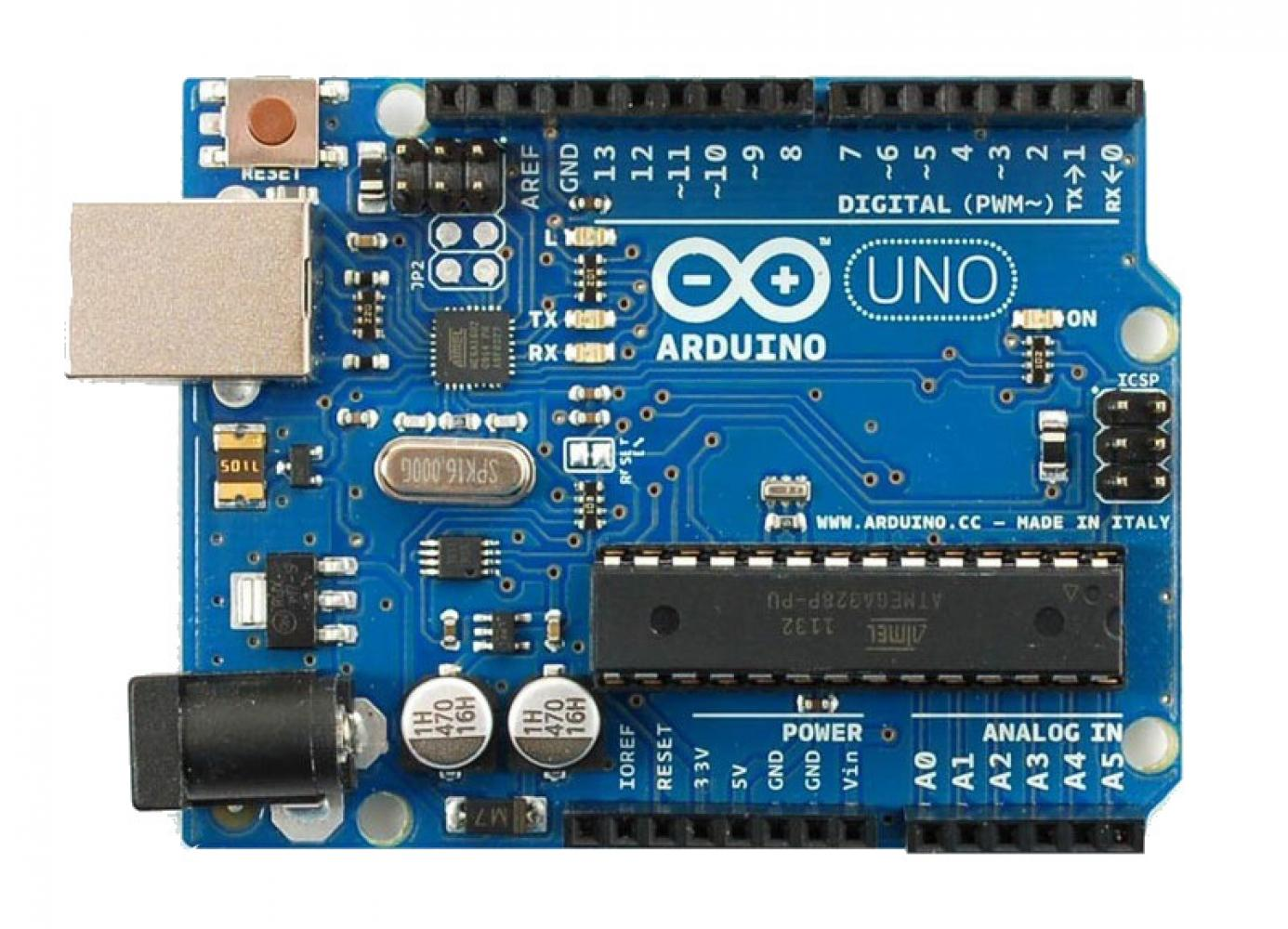
  
Рисунок 1 – Arduino UNO R3

  
Рисунок 2 - кабель USB 2.0 (A-B)

  
Рисунок 3 - джойстик двухосевой

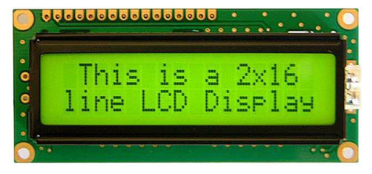


Рисунок 4 - SPI/I2C 1602 LCD yellow

В процессе практики мы провели модернизацию схемы подключения для нашей игровой приставки. Исходя из полученного опыта и дополнительных исследований, мы решили внести изменения, чтобы улучшить функциональность и удобство использования приставки, а именно добавили двухосевой джойстик вместо кнопки.

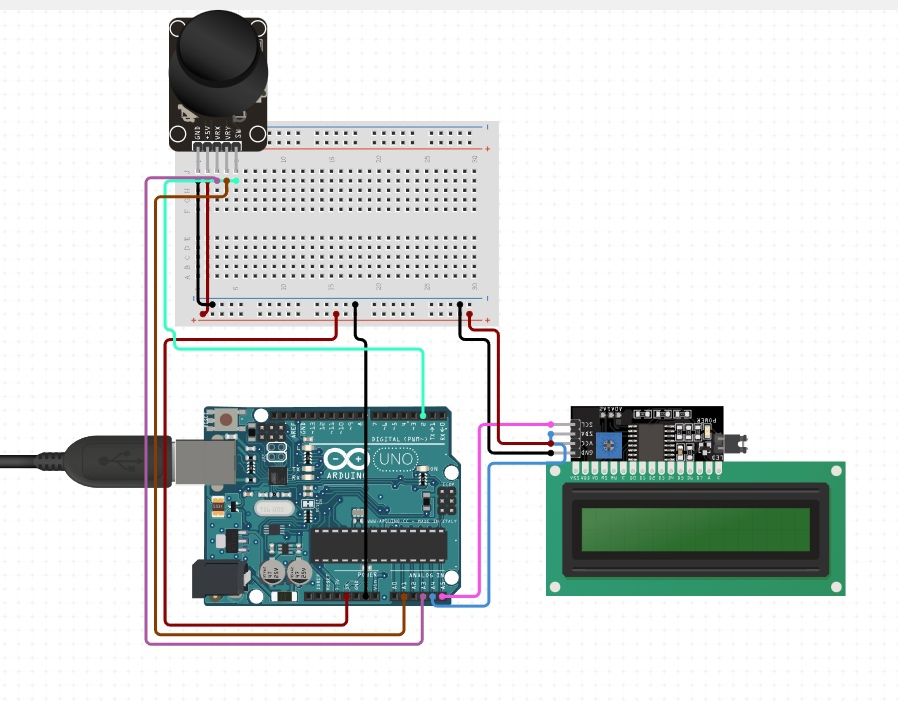
готовая схема подключения - рис. 5  


Рисунок 5 - схема подключения

Сборка физической конструкции: Мы следовали инструкциям и схемам подключения для сборки физической конструкции игровой приставки. Подключили экран, двухосевой джойстик и другие компоненты к платформе Arduino в соответствии с разработанной схемой.

Готовая модель приставки - рис.6

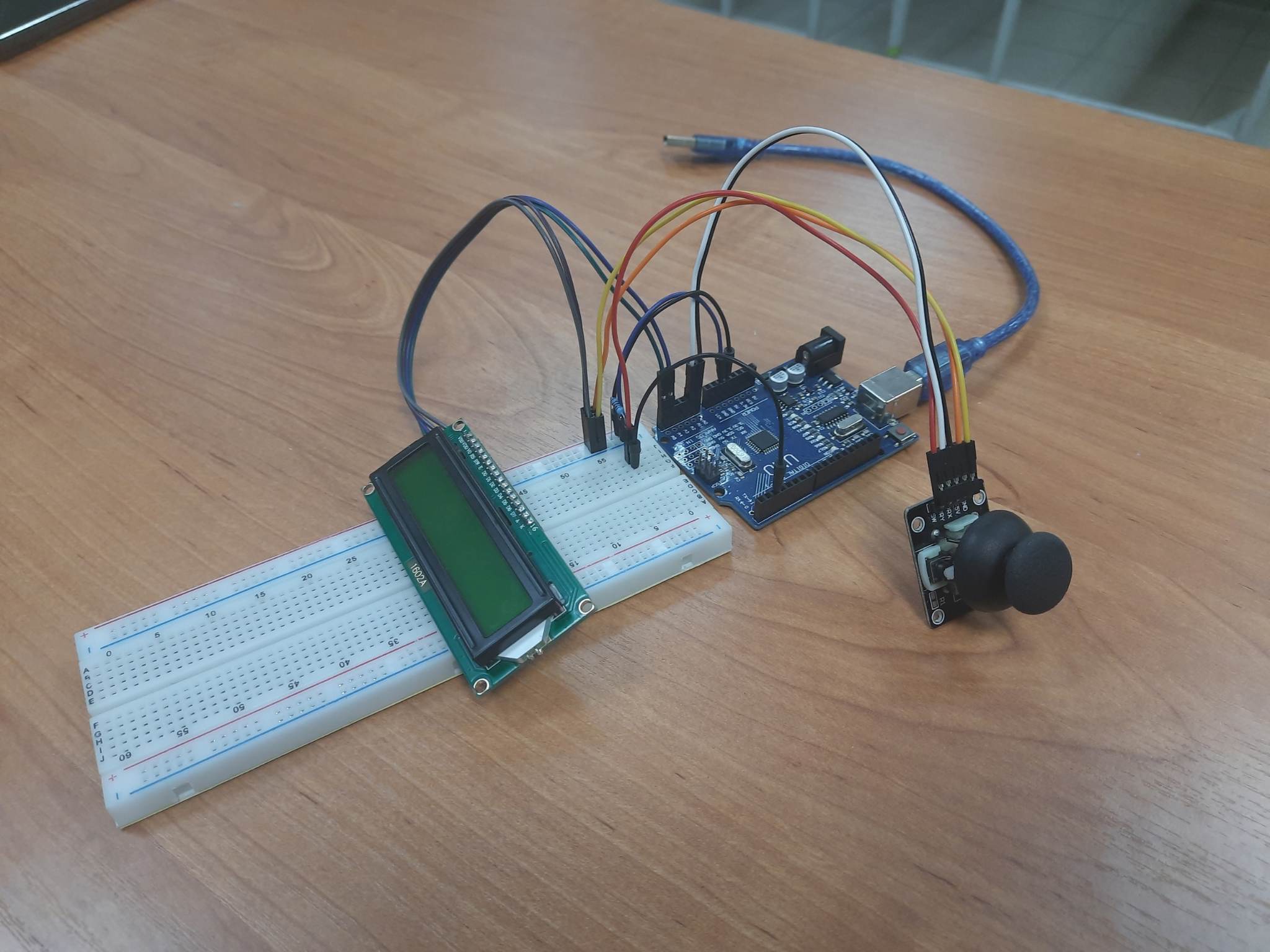


Рисунок 5 - Готовая модель игровой приставки

Проверка и тестирование: После завершения сборки, мы провели проверку и тестирование игровой приставки. Убедились, что все компоненты правильно подключены и функционируют должным образом. Выполнили необходимые настройки и исправили возможные неполадки.

В ходе нашей практики, мы поставили перед собой задачу не только собрать игровую приставку, но и обогатить игровой процесс за счет внедрения инновационных элементов. В рамках этой задачи мы провели следующие действия:

1) Анализ исходного кода: Провели тщательный анализ существующего кода, найденного в Интернете. Изучили его особенности, функции и возможности. Определили потенциал для улучшений и добавления новых элементов в игровой процесс.

Исходный код, который мы использовали:

#include <Wire.h> // библиотека для протокола IIC

#include <LiquidCrystal\_I2C.h> // подключаем библиотеку LCD IIC

LiquidCrystal\_I2C LCD(0x27, 20, 2); // присваиваем имя lcd для дисплея

int level = 1; // переменная для отсчета уровня

int pause = 400; // переменная для задержки

byte p = 0; // переменная для времени прыжка

// создаем массивы дракончика, дерева, камня и птицы

byte dracon[8] = {

0b01110, 0b11011, 0b11111, 0b11100, 0b11111, 0b01100, 0b10010, 0b11011

};

byte derevo[8] = {

0b00000, 0b00000, 0b00000, 0b11011, 0b11011, 0b11011, 0b01100, 0b01100

};

byte kamen[8] = {

0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b01110, 0b11111

};

byte ptica[8] = {

0b00100, 0b00101, 0b01111, 0b11111, 0b10100, 0b00100, 0b00000, 0b00000

};

void setup() {

pinMode (10, OUTPUT); // подключаем пьезодинамик

pinMode (A1, INPUT); // подключаем кнопку

analogWrite(A1, LOW);

LCD.init(); // инициализация LCD дисплея

LCD.backlight(); // включение подсветки дисплея

// создаем символы дракончика, дерева, камня и птицы

LCD.createChar(0, dracon);

LCD.createChar(1, derevo);

LCD.createChar(2, kamen);

LCD.createChar(3, ptica);

// начинаем игру: выводим надпись GO!

LCD.setCursor(7, 0);

LCD.print("GO!");

delay(400);

tone(10, 600);

delay(100);

noTone(10);

LCD.clear();

}

void loop() {

// первоначальное положение дракончика и препятствия

byte d = 1;

byte x = 15;

byte y = 1;

// выбираем препятствие, которое появляется, рандомно

byte i = random (1, 4);

if (i == 3) y = 0;

else y = 1;

while (x > 0) {

// очищаем экран и выводим номер уровня

LCD.clear();

LCD.setCursor(0, 0);

LCD.print(level);

// считываем данные с кнопки и учитываем количество циклов в прыжке

// если дракончик находится в прыжке долго - возвращаем его вниз

if (digitalRead(A1) == LOW) d = 1;

if (digitalRead(A1) == HIGH) d = 0;

if (p > 3) d = 1;

// выводим дракончика в нужной строке

LCD.setCursor(4, d);

LCD.print(char(0));

// выводим препятствие

LCD.setCursor(x, y);

tone(10, 50);

LCD.print(char(i));

noTone(10);

// если дракончик наткнулся на препятствие выводим надпись GAME OVER!

if (x == 4 && y == d) {

delay(400);

tone(10, 50);

delay(100);

noTone(10);

delay(100);

tone(10, 20);

delay(300);

noTone(10);

LCD.clear();

delay(200);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(600);

LCD.clear();

delay(400);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(600);

LCD.clear();

LCD.setCursor(3, 1);

LCD.print("LEVEL: ");

LCD.print(level);

delay(400);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(3000);

LCD.clear();

// начинаем игру заново, обнулив уровень игры

LCD.setCursor(7, 0);

LCD.print("GO!");

delay(400);

tone(10, 600);

delay(100);

noTone(10);

LCD.clear();

level = 0;

pause = 400;

p = 0;

y = 1;

x = 0;

break;

}

// если дракончик прыгнул, издаем звук

if (d == 0) { tone(10, 200); delay(100); noTone(10); }

else { delay(100); }

// если дракончик не столкнулся, то меняем положение препятствия

// начинаем считать сколько циклов дракончик находится в прыжке

delay(pause);

x = x - 1;

p = p + 1;

if (p > 4) p = 0;

}

// переходим на следующий уровень и сокращаем время задержки

tone(10, 800);

delay(20);

level = level + 1;

pause = pause - 20;

if (pause < 0) pause = 0;

}

2)Инновационные элементы: На основе проведенного анализа исходного кода, мы разработали инновационные элементы, которые могут значительно улучшить игровую механику и обогатить игровой процесс. Например, мы решили использовать двухосевой джойстик вместо обычной кнопки для более гибкого и точного управления.

3)Улучшение игровой механики: Внедрили новые функции и механики игры, используя инновационные элементы. Например, для стимулирования соревновательного духа и увеличения мотивации игроков мы разработали систему сбора монет, которая интегрирована в общий счёт игры. Эта система добавила новый элемент соревнования, поскольку игрокам теперь придётся успевать собирать монеты, чтобы установить новый рекорд. Для дополнительного увеличения интереса игрока к соревнованиям мы создали меню игры, в котором внедрили функцию просмотра лучшего счёта игрока. Пользователь может легко получить доступ к этой информации и увидеть наивысший достигнутый им результат. Это позволяет игроку отслеживать свой собственный прогресс и сравнивать его с другими игроками, что поддерживает их соревновательный дух и стимулирует усилия по достижению новых рекордов. Таким образом, с помощью системы сбора монет и меню игры с отображением лучшего счёта, мы создали увлекательную среду, которая способствует активному участию игроков, поддерживает соревновательный аспект и повышает их мотивацию.

4)Тестирование и оптимизация: После внедрения инновационных элементов, мы провели тестирование и оптимизацию, чтобы убедиться, что они работают стабильно и без ошибок. Исправили возможные недочеты и настроили игровую механику для достижения баланса и лучшего игрового опыта.

Итоговый код:

#include <Wire.h> // библиотека для протокола IIC

#include <LiquidCrystal\_I2C.h> // подключаем библиотеку LCD IIC

LiquidCrystal\_I2C LCD(0x3F, 16, 2); // присваиваем имя lcd для дисплея

byte m = 0;

byte c = 0;

bool isMenu = true;

bool isScore = false;

bool isPlay = false;

bool isReset = false;

int score = 0; // переменная для счета

int maxScore = 0;

int pause = 400; // переменная для задержки

byte p = 0; // переменная для времени прыжка

int xPin = A0; // Пин для оси X джойстика

int yPin = A1; // Пин для оси Y джойстика

int buttonPin = 2; // Пин для кнопки джойстика

// создаем массивы дракончика, дерева, камня и птицы

byte dracon1[8] = {

0b01110,

0b11011,

0b11111,

0b11100,

0b11111,

0b01100,

0b10011,

0b11000

};

byte dracon2[8] = {

0b01110,

0b11011,

0b11111,

0b11111,

0b11100,

0b01100,

0b11010,

0b00011

};

byte derevo[8] = {

0b00000,

0b00110,

0b00110,

0b10110,

0b10110,

0b11110,

0b00110,

0b00110

};

byte kamen[8] = {

0b00000,

0b00000,

0b00000,

0b00000,

0b00000,

0b00000,

0b01110,

0b11111

};

byte ptica1[8] = {

0b01100,

0b11100,

0b00100,

0b00100,

0b11111,

0b01110,

0b00100,

0b01010

};

byte ptica2[8] = {

0b01100,

0b11100,

0b10101,

0b11111,

0b01110,

0b00100,

0b00100,

0b01010

};

byte moneta1[8] = {

0b00000,

0b01110,

0b10001,

0b10101,

0b10101,

0b10001,

0b01110,

0b00000

};

byte moneta2[8] = {

0b01110,

0b10001,

0b10101,

0b10101,

0b10001,

0b01110,

0b00000,

0b00000

};

void setup() {

Serial.begin (9600);

LCD.init(); // инициализация LCD дисплея

LCD.backlight(); // включение подсветки дисплея

// создаем символы дракончика, дерева, камня и птицы

LCD.createChar(0, dracon1);

LCD.createChar(1, derevo);

LCD.createChar(2, kamen);

LCD.createChar(3, ptica1);

LCD.createChar(4, moneta1);

LCD.createChar(5, dracon2);

LCD.createChar(6, ptica2);

LCD.createChar(7, moneta2);

// начинаем игру: выводим надпись GO!

LCD.setCursor(5, 0);

LCD.print("HELLO!");

delay(1000);

LCD.clear();

}

void loop() {

int yAxisValue = analogRead(yPin);

if (yAxisValue < 10) c = 0;

if (yAxisValue > 900) c = 1;

int pinValue = digitalRead(buttonPin);

if (c == 1 && pinValue == LOW && isMenu) {

m = 1;

}

if (c == 1 && pinValue == LOW && (isScore || isReset)) {

m = 0;

}

if (c == 0 && pinValue == LOW && (isMenu || isReset)) {

isReset = false;

m = 2;

}

switch (m) {

case 0:

ShowMenu();

break;

case 1:

ShowScore();

break;

case 2:

Play();

break;

case 3:

ResetGame();

break;

}

}

void ShowMenu() {

isMenu = true;

isScore = false;

isPlay = false;

isReset = false;

LCD.clear();

LCD.setCursor(4, 0);

LCD.print("Play");

LCD.setCursor(4, 1);

LCD.print("Score");

Cursor();

delay(400);

}

void ShowScore() {

isMenu = false;

isScore = true;

isPlay = false;

isReset = false;

LCD.clear();

LCD.setCursor(4, 0);

LCD.print("Score:");

LCD.setCursor(10, 0);

LCD.print(maxScore);

LCD.setCursor(4, 1);

LCD.print("Back");

Cursor();

delay(400);

}

void Cursor() {

LCD.setCursor(3, c);

LCD.print(">");

delay(300);

LCD.setCursor(3, c);

LCD.print(" ");

}

void Play() {

isMenu = false;

isScore = false;

isPlay = true;

while (!isReset) {

// первоначальное положение дракончика и препятствия

byte d = 1;

byte x = 15;

byte y = 1;

byte a = 0;

// выбираем препятствие, которое появится, рандомно

byte i = random (1, 5);

if (i == 3) y = 0;

else y = 1;

if (i == 4) y = random (0, 2);

while (x > 0) {

// очищаем экран и выводим номер уровня и счет

LCD.clear();

LCD.setCursor(0, 0);

LCD.print(score);

int yAxisValue = analogRead(yPin);

// считываем данные с кнопки и учитываем количество циклов в прыжке

// если дакончик находится в прыжке долго - возвращаем его вниз

if (yAxisValue < 10) d = 0;

else d = 1;

if (p > 3) d = 1;

// выводим дракончика в нужной строке

LCD.setCursor(3, d);

if (a == 0) {

a = 3;

LCD.print(char(0));

}

else {

a = 0;

LCD.print(char(5));

}

// выводим препятствие

LCD.setCursor(x, y);

tone(10, 50);

if (i == 3 || i == 4) LCD.print(char(i + a));

else LCD.print(char(i));

// если дракончик наткнулся на препятствие выводим надпись GAME OVER!

if (x == 3 && y == d) {

if (i == 4) {

score = score + 1;

LCD.setCursor(0, 0);

LCD.print(score);

}

else {

LCD.clear();

delay(200);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(600);

LCD.clear();

delay(400);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(600);

LCD.clear();

delay(400);

LCD.setCursor(3, 0);

LCD.print("GAME OVER!");

delay(3000);

LCD.clear();

pause = 400;

p = 0;

y = 1;

x = 0;

if (maxScore < score) maxScore = score;

score = -1;

isReset = true;

break;

break;

}

}

// если дракончик не столкнулся, то меняем положение препятствия

// начинаем считать сколько циклов дракончик находится в прыжке

delay(pause);

x = x - 1;

p = p + 1;

if (p > 4) p = 0;

}

// переходим на следующий уровень и сокращаем время задержки

score = score + 1;

pause = pause - 20;

if (pause < 0) pause = 0;

}

ResetGame();

}

void ResetGame() {

isMenu = false;

isScore = false;

isPlay = false;

isReset = true;

LCD.clear();

LCD.setCursor(4, 0);

LCD.print("Retry");

LCD.setCursor(4, 1);

LCD.print("Menu");

Cursor();

delay(400);

}

**ЗАКЛЮЧЕНИЕ**

В ходе нашей практики мы успешно провели анализ существующего кода, найденного в интернете, и разработали инновационные элементы для обогащения игрового процесса. Внедрение двухосевого джойстика, системы сбора монет, меню игры, а также изменение текстур и добавление анимаций значительно улучшили игровой опыт и придали игре новый визуальный облик.

Одним из ключевых аспектов нашей работы было изменение текстур и добавление новых анимаций. Мы стремились сделать игровое окружение более привлекательным и уникальным. Путем создания новых текстур и визуальных эффектов мы достигли желаемого результата. Эти изменения позволили усилить атмосферу игры и визуальное впечатление, добавив в него элементы эстетики и стиля.

Использование анимаций, таких как переходы между игровыми состояниями, движение персонажей или объектов, придали игре динамичность и позволили создать более плавный и захватывающий игровой процесс. Анимации также способствуют улучшению интерактивности и общего ощущения присутствия в игре.

Мы уверены, что наши усовершенствования вносят значительный вклад в развитие игровой индустрии. Полученный опыт и навыки могут быть применены в будущих проектах разработки игр, способствуя созданию более захватывающих и уникальных игровых впечатлений.

Наша практика позволила нам не только расширить наши знания и навыки в области разработки игровых приложений, но и продемонстрировать нашу способность к инновациям и творческому подходу к решению задач. Мы гордимся полученными результатами и уверены, что наш вклад поможет в дальнейшем развитии игровой индустрии и созданию увлекательных игровых продуктов.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1.[https://developer.alexanderklimov.ru/arduino/button.php](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fdeveloper.alexanderklimov.ru%2Farduino%2Fbutton.php)

2.[https://arduino.ru](https://vk.com/away.php?utf=1&to=https%3A%2F%2Farduino.ru)

3.[https://3d-diy.ru/wiki/arduino-datchiki/arduino-joystick/#Connection](https://vk.com/away.php?utf=1&to=https%3A%2F%2F3d-diy.ru%2Fwiki%2Farduino-datchiki%2Farduino-joystick%2F%23Connection)