УДК 004

Создание игры «Snake» на базе микроконтроллера семейства Arduino

Болтовский Гавриил Александрович
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
Целью данной статьи является разработка игры змейка на базе микроконтроллера семейства Arduino. Прошивка написана на языке программирования C++ и библиотеки Adafruit_PCD8544 для работы с дисплеем Nokia 5110. Результатом исследования станет устройство и его прошивка с подробным описанием их реализации.
Ключевые слова: Arduino, видеоигры, встроенная разработка

Creating the game "Snake" based on the microcontroller of the Arduino family

Boltovsky Gavriil Alexandrovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this article is to develop a snake game based on the microcontroller of the Arduino family. The firmware is written in the C++ programming language and the library Adafruit_PCD8544 for working with the Nokia 5110 display. The result of the study will be the device and its firmware with a detailed description of their implementation.
Keywords: Arduino, video games, embedded development

1. Введение

1.1 Актуальность исследования

Игра «Snake» является классической аркадной игрой. Она проста в управлении и не требует больших вычислительных мощностей. «Snake» может быть воссоздана с помощью микроконтроллера Atmel ATmega168 или ATmega328, которые установлены в платах Arduino Uno и Arduino Nano. Создание игры «Snake» может использоваться на уроках робототехники, а описываемые методы применимы для создания других игр.

1.2 Обзор исследований

А. В. Турантаев описал алгоритмы и технологии в программировании игр [1]. История появления и развития игр аркадного направления расписана в работе М. И. Андерсона [2]. Рассматриваются основные вопросы,

возникающие при работе с ЖКИ дисплеем Nokia 5110 в работе А. Ю Ивойлова [3].

1.3 Цель исследования

Создать устройство, на котором можно будет играть в «Snake».

1.4 Постановка задачи

На первом этапе было создано устройство. Затем был осуществлён поиск подходящих библиотек для работы с прошивкой. Конечным этапом является непосредственное создание прошивки игры.

2. Методы исследования

Для сборки используется плата Arduino UNO.

Устройство, позволяющее запускать игру «Snake», должно иметь кнопки ввода и дисплей. Ввод реализуется модулем-клавиатурой, игровой процесс предаётся на дисплей Nokia 5110.

В проекте используется библиотеки EncButton [4] и AnalogKeyboard, они позволяют обрабатывать нажатия на клавиатуру.

Схема устройства представлена на рисунке (рис. 1).

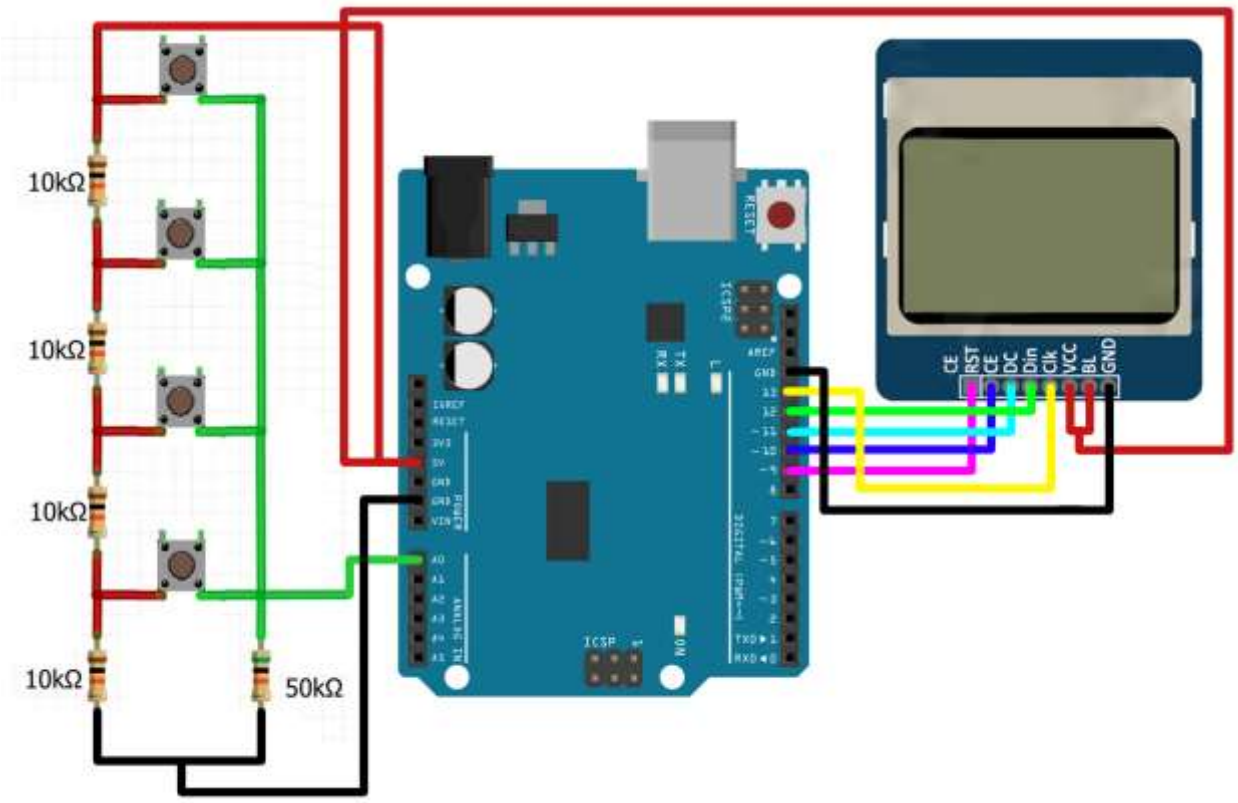

Рисунок 1 – Схема подключения модулей

Для взаимодействия с дисплеем выбрана библиотека Adafruit_PCD8544. Она обладает достаточным функционалом для вывода графики, необходимой для игры.

В начале программы объявляются необходимые переменные и объекты, включая экземпляр класса Adafruit_PCD8544 для управления дисплеем. Затем в функции setup() (рис. 2) выполняется инициализация дисплея, отображение заголовка игры "SNAKE" в течение пяти секунд, инициализация параметров игры, отрисовка начального положения змейки и цели.

```
#include <EncButton.h>
#include <AnalogKey.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <SPI.h>
#define KB_PIN 0

Adafruit_PCD8544 display = Adafruit_PCD8544(13, 12, 11, 10, 9);
boolean dl = false, dr = false, du = false, dd = false;
int x[50], y[50], i, slength, tempx = 10, tempy = 10, xx, yy;
unsigned int high;
uint8_t bh, bl;
int xegg, yegg;
int freq, tb;
unsigned long time = 280, beeptime = 50;
int score = 0, flag = 0;

EncButton<EB_TICK, VIRT_BTN> btnRight;
EncButton<EB_TICK, VIRT_BTN> btnLeft;
EncButton<EB_TICK, VIRT_BTN> btnUp;
EncButton<EB_TICK, VIRT_BTN> btnDown;

// создаём массив значений сигналов с кнопок
  int16_t sigs[4] = {
    393, 732,
    1023, 538
  };

// указываем пин, количество кнопок и массив значений
  AnalogKey<KB_PIN, 4, sigs> keys;

void setup(){
  Serial.begin(9600);
  display.begin();
  display.setContrast(50);
  display.clearDisplay();
  display.drawRoundRect(0, 0, 84 , 25, 1, 2);
  display.setTextSize(2);
  display.setTextColor(BLACK);
  display.setCursor(12, 6);
  display.println("SNAKE");
  display.setTextSize(1);
  display.setTextColor(BLACK);
  display.display();
```

Рисунок 2 – Начало программы

Основная логика игры находится в функции loop(). Она вызывает функцию movesnake() (рис. 3), которая обрабатывает движение змейки. Функция movesnake() вызывается с определенной периодичностью (задержка определяется переменной time). Внутри функции проверяется текущее направление змейки, обновляются координаты змейки и проверяется столкновение с границами поля или собственным телом. Также проверяется, съела ли змейка цель (egg), в таком случае увеличивается длина змейки и

увеличивается счетчик очков. Если игра окончена, выводится сообщение об окончании игры и отображаются результаты (счет и рекорд).

```
77  void movesnake()
78  {
79
80    if (flag == 0)
81    {
82      direct();
83    }
84
85    if (millis() % time == 0)
86    {
87      if (flag == 0)
88      {
89        if (dr == true) {
93        if (dl == true) {
97        if (du == true) {
101       if (dd == true) {
105     }
106     flag = 0;
107     checkgame();
108     checkegg();
109
110     //изменение координат
111     for (i = 0; i <= slength; i++)
112     {
120     //перерисовка змейки и цели
121     drawsnake();
122   }
123 }
```

Рисунок 3 – Код функции movesnake()

Функция checkgame() проверяет столкновение змейки с границами игрового поля и самой собой. Если столкновение произошло, игра считается оконченной, отображается сообщение об окончании игры, выводятся результаты и игра перезапускается (рис. 4).

```
125 void checkgame()
126 {
127   for (i = 1; i < slength; i++)
128   {
129     //Проверяет, что рекорд побит и вывод нового результата
130     high = (((0xff00 + bh) << 8) + bl);
131     if (score > high)
132     {
140     //Проверка касания границ игрового поля
141     if ((x[0] <= 1 || x[0] >= 83) || (y[0] <= 12 || y[0] >= 47)
142     {
174   }
175 }
```

Рисунок 4 – Код функции checkgame()

Функция checkegg() проверяет, съела ли змейка цель (egg). Если координаты змейки и цели совпадают, увеличивается длина змейки, обновляется счетчик очков, цель перемещается в новое место на игровом поле (рис. 5).

```
125  void checkgame()
126  {
127    for (i = 1; i < slength; i++)
128    {
129      //Проверяет, что рекорд побит и вывод нового результата
130      high = (((0xff00 + bh) << 8) + bl);
131      if (score > high)
132      {
140      //Проверка касания границ игрового поля
141      if ((x[0] <= 1 || x[0] >= 83) || (y[0] <= 12 || y[0] >= 47)
142      {
174    }
175  }
```

Рисунок 5 – Функция checkegg()

Функция direct() отвечает за изменение направления движения змейки. Она определяет, какая кнопка была нажата, и, в зависимости от этого, изменяет направление движения змейки.

Функция drawsnake() отрисовывает текущее положение змейки и цели на дисплее(рис. 6).

```
204  void direct()
205  {
206    btnRight.tick(keys.status(0));
207    btnLeft.tick(keys.status(1));
208    btnUp.tick(keys.status(2));
209    btnDown.tick(keys.status(3));
210
211    //изменение движения если нажимаем на кнопки
212    if (btnRight.click() and dr == false) //право
213    {
219    else if (btnLeft.click() and dl == false)//лево
220    {
226    else if (btnUp.click() and dd == false)// вверх
227    {
233    else if (btnDown.click() and du == false) //вниз
234    {
240
241  }
242
243  void drawsnake()
244  {
245    display.fillRect(xegg, yegg, 3, 3, BLACK);
246    display.drawCircle(x[0], y[0], 1, BLACK);
247    display.drawCircle(x[slength], y[slength], 1, WHITE);
248    display.display();
249  }
```

Рисунок 6 – Функции direct() и drawsnake()

Функция redraw() выполняет перезапуск игры, сбрасывая все параметры в исходное состояние, отображает начальное положение змейки и цели (рис. 7).

```
void redraw()
{
  display.drawRect(0, 0, 84, 48, 1);
  display.drawRect(0, 0, 84, 48, 1);
  display.setCursor(4, 2);
  display.print("S:     R:");
  display.drawRect(0, 0, 84, 11, 1);
  display.fillRect(21, 1, 20, 9, 0);
  display.setCursor(22, 2);
  display.print(score);
  display.fillRect(61, 1, 20, 9, 0);
  display.setCursor(65, 2);
  display.print(high);

  xegg = (display.width()) / 2;
  yegg = (display.height()) / 2;
  dl = false, dr = false, du = false, dd = false;
  dr = true;
  display.setCursor(4, 2);
  display.print("S:     R:");
  display.drawRect(0, 0, 84, 11, 1);
  //возвращаем начальные координаты
  for (i = 0; i <= slength; i++)
  {
    x[i] = 25 - 3 * i;
    y[i] = 30;
  }
  tempx = 33 - 3 * i;
  tempy = 30;
  display.display();
}
```

Рисунок 7 – Функция redraw()

Весь цикл игры (движение змейки, проверка столкновений, обновление дисплея) повторяется бесконечно в функции loop().

Внешний вид собранного устройства представлен на рисунке (рис. 8).

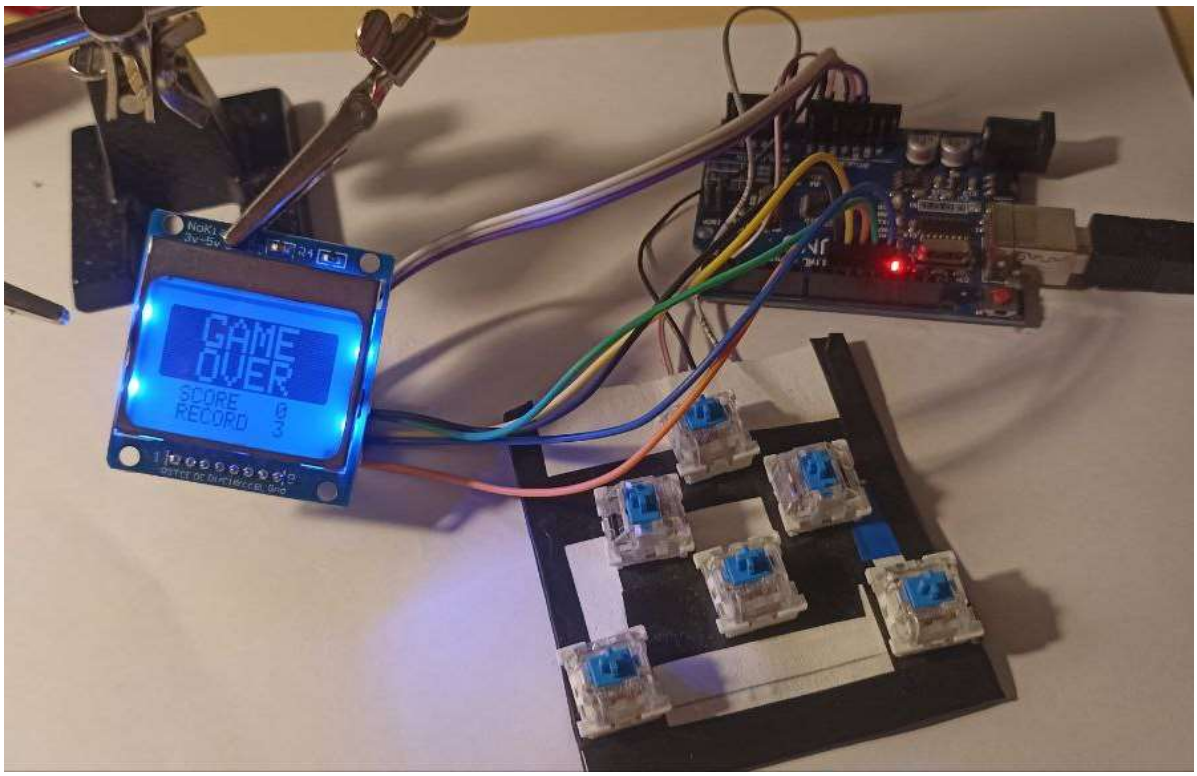

Рисунок 8 – Внешний вид устройства

В репозитории проекта на GitHub [5] можно найти код прошивки и схему устройства, а также дополнительные изображения, демонстрирующие его работу.

3. Выводы

Таким образом, было создано устройство, позволяющее запускать игру «Snake». Описанные методы могут использоваться на уроках робототехники и при обучении алгоритмам, а также для создания других игр.

Библиографический список

1. Турантаев А. В. Программирование игр, алгоритмы и технологии // Инновации. Наука. Образование. 2021. № 28. С. 1183-1192
2. Андерсон М. И. Влияние игр аркадного направления на развитие индустрии компьютерных игр // Молодой исследователь Дона. 2019. № 2(17). С. 85-90.
3. Ивойлов А. Ю Особенности работы с ЖКИ дисплеем NOKIA 5110 /. // Автоматика и программная инженерия. 2013. № 4(6). С. 8-13
4. Github [Электронный ресурс]. URL: https://github.com/GyverLibs/EncButton (дата обращения: 4.07.2023)
5. Github [Электронный ресурс]. URL: https://github.com/Gavriilbolt/Ardurobo/tree/master/ArduinoGames/Snake (дата обращения: 5.07.2023)