// Определение пинов для светодиодов

#define LED\_RED 11

#define LED\_GREEN 9

#define LED\_BLUE 7

#define LED\_YELLOW 5

#define LED\_CORRECT 4 // Зеленый светодиод для правильного ответа

#define LED\_WRONG 2 // Красный светодиод для неправильного ответа

// Определение пинов для кнопок

#define BUTTON\_RED 12

#define BUTTON\_GREEN 10

#define BUTTON\_BLUE 8

#define BUTTON\_YELLOW 6

// Определение пина для зуммера

#define BUZZER 3

// Определение тонов для каждого цвета

#define RED\_TONE 220

#define GREEN\_TONE 262

#define BLUE\_TONE 330

#define YELLOW\_TONE 392

#define TONE\_DURATION 250 // Длительность звука в миллисекундах

// Игровые переменные

int GAME\_SPEED = 250; // Начальная скорость игры

int GAME\_STATUS = 0; // Текущее состояние игры (0 - начало, 1 - воспроизведение последовательности, 2 - ожидание ввода, 3 - конец игры)

const int GAME\_MAX\_SEQUENCE = 50; // Максимальная длина последовательности

int GAME\_SEQUENCE[GAME\_MAX\_SEQUENCE]; // Массив для хранения последовательности

int GAME\_STEP = 0; // Текущий шаг в последовательности

int READ\_STEP = 0; // Текущий шаг для ввода игрока

void setup() {

Serial.begin(9600); // Инициализация последовательного порта для отладки

// Инициализация генератора случайных чисел

randomSeed(analogRead(0));

// Настройка пинов светодиодов как выходы

pinMode(LED\_RED, OUTPUT);

pinMode(LED\_GREEN, OUTPUT);

pinMode(LED\_BLUE, OUTPUT);

pinMode(LED\_YELLOW, OUTPUT);

pinMode(LED\_CORRECT, OUTPUT);

pinMode(LED\_WRONG, OUTPUT);

// Настройка пинов кнопок как входы с подтягивающим резистором

pinMode(BUTTON\_RED, INPUT\_PULLUP);

pinMode(BUTTON\_GREEN, INPUT\_PULLUP);

pinMode(BUTTON\_BLUE, INPUT\_PULLUP);

pinMode(BUTTON\_YELLOW, INPUT\_PULLUP);

// Настройка пина зуммера как выход

pinMode(BUZZER, OUTPUT);

}

void loop() {

// Основной цикл игры - проверяем текущее состояние игры

switch(GAME\_STATUS) {

case 0: // Начало новой игры

resetGame();

break;

case 1: // Воспроизведение последовательности

playSequence();

break;

case 2: // Ожидание ввода игрока

readSequence();

break;

case 3: // Конец игры

gameOver();

break;

}

}

// Функция сброса игры и создания новой последовательности

void resetGame() {

// Сброс шагов

READ\_STEP = 0;

GAME\_STEP = 0;

// Создание случайной последовательности

for(int i = 0; i < GAME\_MAX\_SEQUENCE; i++) {

GAME\_SEQUENCE[i] = random(4) + 1; // Генерация чисел от 1 до 4

}

// Переход к состоянию воспроизведения последовательности

GAME\_STATUS = 1;

}

// Функция воспроизведения последовательности

void playSequence() {

// Воспроизведение текущего шага последовательности

for(int i = 0; i < GAME\_STEP; i++) {

Serial.print("Set LED ");

Serial.println(GAME\_SEQUENCE[i]);

delay(GAME\_SPEED \* 2); // Пауза перед включением светодиода

// Включение соответствующего светодиода и звука

setLED(GAME\_SEQUENCE[i]);

playTone(GAME\_SEQUENCE[i]);

delay(GAME\_SPEED); // Длительность свечения светодиода

clearLEDs(); // Выключение всех светодиодов

}

// Переход к состоянию ожидания ввода игрока

GAME\_STATUS = 2;

}

// Функция чтения ввода игрока

void readSequence() {

// Чтение состояния кнопок

int button\_value = readButtons();

if(button\_value > 0) { // Если нажата какая-то кнопка

if(button\_value == GAME\_SEQUENCE[READ\_STEP]) { // Если нажата правильная кнопка

// Включение соответствующего светодиода и звука

setLED(button\_value);

playTone(button\_value);

// Включение зеленого светодиода (правильный ответ)

digitalWrite(LED\_CORRECT, HIGH);

delay(GAME\_SPEED);

clearLEDs(); // Выключение всех светодиодов

digitalWrite(LED\_CORRECT, LOW);

// Увеличение скорости игры (усложнение)

if (GAME\_SPEED > 100) {

GAME\_SPEED = GAME\_SPEED - 15;

}

Serial.println("Correct!");

if(READ\_STEP == GAME\_STEP) { // Если игрок завершил текущую последовательность

// Сброс шага чтения

READ\_STEP = 0;

// Увеличение длины последовательности

GAME\_STEP++;

// Переход к воспроизведению новой последовательности

GAME\_STATUS = 1;

Serial.println("Next step");

// Мигание всеми светодиодами для индикации перехода на новый уровень

setLEDs(true, true, true, true);

delay(GAME\_SPEED);

setLEDs(false, false, false, false);

} else {

READ\_STEP++; // Переход к следующему шагу в последовательности

}

delay(10);

} else { // Если нажата неправильная кнопка

// Переход к состоянию конца игры

GAME\_STATUS = 3;

Serial.println("Game Over!");

}

}

delay(10);

}

// Функция обработки конца игры

void gameOver() {

// Включение красного светодиода

digitalWrite(LED\_WRONG, HIGH);

// Воспроизведение звукового сигнала проигрыша

tone(BUZZER, 98, TONE\_DURATION);

delay(TONE\_DURATION);

tone(BUZZER, 93, TONE\_DURATION);

delay(TONE\_DURATION);

tone(BUZZER, 87, TONE\_DURATION);

delay(TONE\_DURATION);

delay(GAME\_SPEED);

// Выключение красного светодиода

digitalWrite(LED\_WRONG, LOW);

// Сброс скорости игры

GAME\_SPEED = 250;

// Переход к состоянию новой игры

GAME\_STATUS = 0;

}

// Вспомогательные функции

// Функция чтения состояния кнопок

int readButtons() {

if (digitalRead(BUTTON\_RED) == LOW) {

return 1; // Красная кнопка

} else if (digitalRead(BUTTON\_GREEN) == LOW) {

return 2; // Зеленая кнопка

} else if (digitalRead(BUTTON\_BLUE) == LOW) {

return 3; // Синяя кнопка

} else if (digitalRead(BUTTON\_YELLOW) == LOW) {

return 4; // Желтая кнопка

} else {

return 0; // Ни одна кнопка не нажата

}

}

// Функция включения определенного светодиода

void setLED(int led) {

switch(led) {

case 1: // Красный

digitalWrite(LED\_RED, HIGH);

break;

case 2: // Зеленый

digitalWrite(LED\_GREEN, HIGH);

break;

case 3: // Синий

digitalWrite(LED\_BLUE, HIGH);

break;

case 4: // Желтый

digitalWrite(LED\_YELLOW, HIGH);

break;

}

}

// Функция включения всех светодиодов

void setLEDs(bool red, bool green, bool blue, bool yellow) {

digitalWrite(LED\_RED, red ? HIGH : LOW);

digitalWrite(LED\_GREEN, green ? HIGH : LOW);

digitalWrite(LED\_BLUE, blue ? HIGH : LOW);

digitalWrite(LED\_YELLOW, yellow ? HIGH : LOW);

}

// Функция выключения всех светодиодов

void clearLEDs() {

digitalWrite(LED\_RED, LOW);

digitalWrite(LED\_GREEN, LOW);

digitalWrite(LED\_BLUE, LOW);

digitalWrite(LED\_YELLOW, LOW);

}

// Функция воспроизведения тона для определенного цвета

void playTone(int toneNum) {

switch(toneNum) {

case 1: // Красный

tone(BUZZER, RED\_TONE, TONE\_DURATION);

break;

case 2: // Зеленый

tone(BUZZER, GREEN\_TONE, TONE\_DURATION);

break;

case 3: // Синий

tone(BUZZER, BLUE\_TONE, TONE\_DURATION);

break;

case 4: // Желтый

tone(BUZZER, YELLOW\_TONE, TONE\_DURATION);

break;

}

}