/\*

\* Игра "Саймон говорит" для Arduino

\* Полная версия с комментариями

\* Подключение:

\* - 4 основных светодиода (красный, зеленый, синий, желтый)

\* - 4 кнопки (соответствуют цветам светодиодов)

\* - 2 индикаторных светодиода (правильно/неправильно)

\* - Пьезоизлучатель (зуммер)

\*/

// ==================== НАСТРОЙКА ПИНОВ ====================

// Пины для основных светодиодов

#define LED\_RED 11 // Красный светодиод

#define LED\_GREEN 9 // Зеленый светодиод

#define LED\_BLUE 7 // Синий светодиод

#define LED\_YELLOW 5 // Желтый светодиод

// Пины для индикаторных светодиодов

#define LED\_CORRECT 4 // Зеленый (правильный ответ)

#define LED\_WRONG 2 // Красный (ошибка)

// Пины для кнопок (подключены с подтягивающими резисторами)

#define BUTTON\_RED 12 // Кнопка для красного

#define BUTTON\_GREEN 10 // Кнопка для зеленого

#define BUTTON\_BLUE 8 // Кнопка для синего

#define BUTTON\_YELLOW 6 // Кнопка для желтого

// Пин для зуммера

#define BUZZER 3

// ==================== НАСТРОЙКА ЗВУКА ====================

// Частоты тонов для каждого цвета (в Гц)

#define RED\_TONE 220 // Тон для красного

#define GREEN\_TONE 262 // Тон для зеленого

#define BLUE\_TONE 330 // Тон для синего

#define YELLOW\_TONE 392 // Тон для желтого

#define TONE\_DURATION 250 // Длительность звука (мс)

// ==================== ИГРОВЫЕ ПЕРЕМЕННЫЕ ====================

int GAME\_SPEED = 500; // Начальная скорость игры (мс)

int GAME\_STATUS = 0; // Текущее состояние игры (0-3)

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

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

int GAME\_STEP = 1; // Текущая длина последовательности

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

// ==================== ОСНОВНЫЕ ФУНКЦИИ ====================

void setup() {

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

randomSeed(analogRead(0)); // Инициализация генератора случайных чисел

// 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 подтягивающими резисторами

pinMode(BUTTON\_RED, INPUT\_PULLUP);

pinMode(BUTTON\_GREEN, INPUT\_PULLUP);

pinMode(BUTTON\_BLUE, INPUT\_PULLUP);

pinMode(BUTTON\_YELLOW, INPUT\_PULLUP);

// Настройка пина зуммера

pinMode(BUZZER, OUTPUT);

// Генерация случайной последовательности

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

GAME\_SEQUENCE[i] = random(4) + 1; // 1-красный, 2-зеленый, 3-синий, 4-желтый

}

}

void loop() {

// Главный игровой цикл - обработка текущего состояния игры

switch(GAME\_STATUS) {

case 0: // Состояние 0: Начало новой игры

startNewGame();

break;

case 1: // Состояние 1: Показ последовательности

showSequence();

break;

case 2: // Состояние 2: Ожидание ввода игрока

checkPlayerInput();

break;

case 3: // Состояние 3: Конец игры (ошибка)

gameOver();

break;

}

delay(10); // Небольшая задержка для стабильности

}

// ==================== ИГРОВАЯ ЛОГИКА ====================

//Начинает новую игру

// Сбрасывает счетчики

// Генерирует новую последовательность

// Устанавливает начальную скорость

void startNewGame() {

GAME\_STEP = 1; // Начинаем с последовательности длины 1

READ\_STEP = 0; // Сброс счетчика ввода

GAME\_SPEED = 500; // Сброс скорости

// Генерация новой случайной последовательности

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

GAME\_SEQUENCE[i] = random(4) + 1;

}

GAME\_STATUS = 1; // Переход к показу последовательности

}

//Показывает текущую последовательность светодиодов

// Поочередно включает светодиоды согласно GAME\_SEQUENCE

// Проигрывает соответствующие звуки

void showSequence() {

// Показываем всю текущую последовательность

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

int currentLED = GAME\_SEQUENCE[i]; // Получаем текущий цвет

// Включаем светодиод и звук

setLED(currentLED);

playTone(currentLED);

// Ждем заданное время

delay(GAME\_SPEED);

// Выключаем светодиод

clearLEDs();

// Короткая пауза между светодиодами

delay(200);

}

GAME\_STATUS = 2; // Переход к вводу игрока

}

// Проверяет ввод игрока

// Сравнивает нажатые кнопки с последовательностью

// Обрабатывает правильные/неправильные ответы

void checkPlayerInput() {

int buttonPressed = readButtons(); // Читаем нажатую кнопку

if(buttonPressed > 0) { // Если какая-то кнопка нажата

if(buttonPressed == GAME\_SEQUENCE[READ\_STEP]) {

// ===== ПРАВИЛЬНЫЙ ОТВЕТ =====

// Включаем зеленый индикатор

digitalWrite(LED\_CORRECT, HIGH);

// Подсвечиваем соответствующий светодиод

setLED(buttonPressed);

playTone(buttonPressed);

// Ждем 300 мс

delay(300);

// Выключаем индикаторы

digitalWrite(LED\_CORRECT, LOW);

clearLEDs();

READ\_STEP++; // Переходим к следующему шагу

if(READ\_STEP >= GAME\_STEP) {

// ===== УРОВЕНЬ ПРОЙДЕН =====

READ\_STEP = 0; // Сброс счетчика ввода

GAME\_STEP++; // Увеличиваем длину последовательности

// Увеличиваем сложность (уменьшаем время)

if(GAME\_SPEED > 200) GAME\_SPEED -= 20;

// Сигнал о переходе на новый уровень - мигание всеми светодиодами

for(int i=0; i<3; i++) {

setLEDs(true, true, true, true);

delay(100);

setLEDs(false, false, false, false);

delay(100);

}

GAME\_STATUS = 1; // Показ новой последовательности

}

} else {

// ===== НЕПРАВИЛЬНЫЙ ОТВЕТ =====

GAME\_STATUS = 3; // Конец игры

}

}

}

// Обработка конца игры (неправильный ответ)

// Мигает красный индикатор

// Проигрывает звук ошибки

// Перезапускает игру через 3 секунды

void gameOver() {

// Мигание красным светодиодом и звук ошибки

for(int i=0; i<3; i++) {

digitalWrite(LED\_WRONG, HIGH);

tone(BUZZER, 200, 200); // Низкий тон ошибки

delay(300);

digitalWrite(LED\_WRONG, LOW);

delay(200);

}

GAME\_STATUS = 0; // Начало новой игры

}

// ==================== ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ ====================

//Чтение нажатых кнопок

//Возвращает:

//1 - красная, 2 - зеленая, 3 - синяя, 4 - желтая, 0 - ничего не нажато

int readButtons() {

if(digitalRead(BUTTON\_RED) == LOW) return 1;

if(digitalRead(BUTTON\_GREEN) == LOW) return 2;

if(digitalRead(BUTTON\_BLUE) == LOW) return 3;

if(digitalRead(BUTTON\_YELLOW) == LOW) return 4;

return 0;

}

//Включает указанный светодиод

//@param led - номер светодиода (1-4)

void setLED(int led) {

switch(led) {

case 1: digitalWrite(LED\_RED, HIGH); break;

case 2: digitalWrite(LED\_GREEN, HIGH); break;

case 3: digitalWrite(LED\_BLUE, HIGH); break;

case 4: digitalWrite(LED\_YELLOW, HIGH); break;

}

}

//Выключает все основные светодиоды

void clearLEDs() {

digitalWrite(LED\_RED, LOW);

digitalWrite(LED\_GREEN, LOW);

digitalWrite(LED\_BLUE, LOW);

digitalWrite(LED\_YELLOW, LOW);

}

/\*\*

\* Управляет всеми светодиодами одновременно

\* @param red - состояние красного

\* @param green - состояние зеленого

\* @param blue - состояние синего

\* @param yellow - состояние желтого

\*/

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);

}

//Проигрывает тон для указанного цвета

void playTone(int toneNum) {

int toneFreq = 0;

switch(toneNum) {

case 1: toneFreq = RED\_TONE; break;

case 2: toneFreq = GREEN\_TONE; break;

case 3: toneFreq = BLUE\_TONE; break;

case 4: toneFreq = YELLOW\_TONE; break;

}

tone(BUZZER, toneFreq, TONE\_DURATION);

}