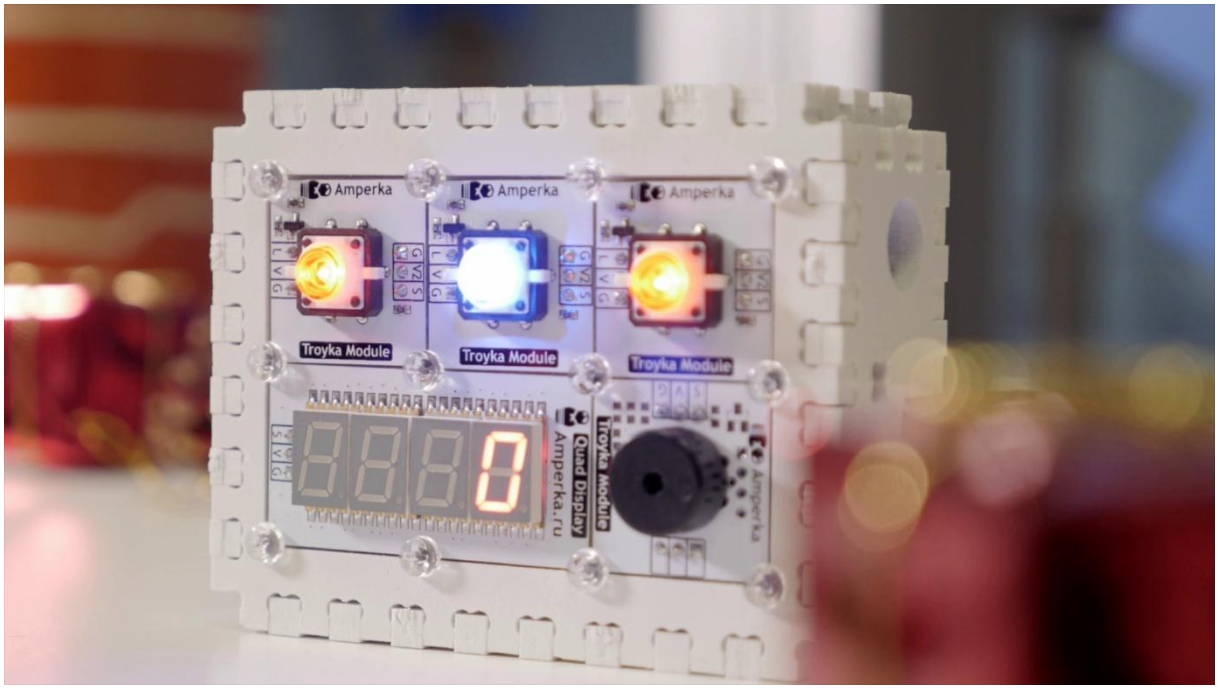
«Саймон говорит…»

Увлекательная электронная игра, направленная на развитие и тренировку памяти. Игроку предстоит последовательно повторять случайные комбинации загорающихся кнопок. Чем длиннее цепочка — тем больше счёт на экране устройства.



Что потребуется

Полный сет компонентов проекта. В сет входят:

1 x Iskra Neo

1 x Troyka-Slot Shield

1 x Structor-Slot Box

1 x Troyka-Quad Display

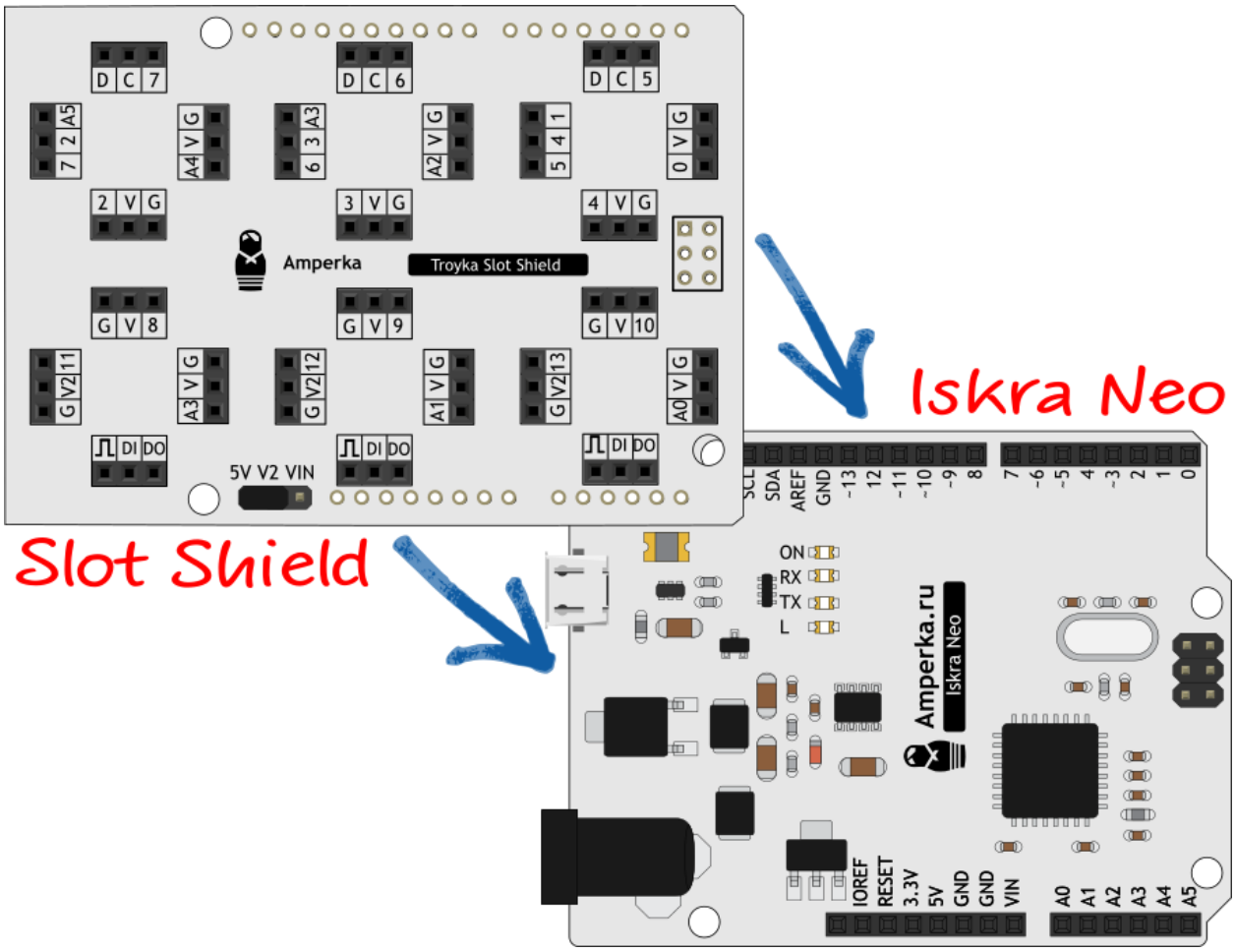
2 x Troyka-Led Button Red

1 x Troyka-Led Button Blue

1 x Troyka-Buzzer

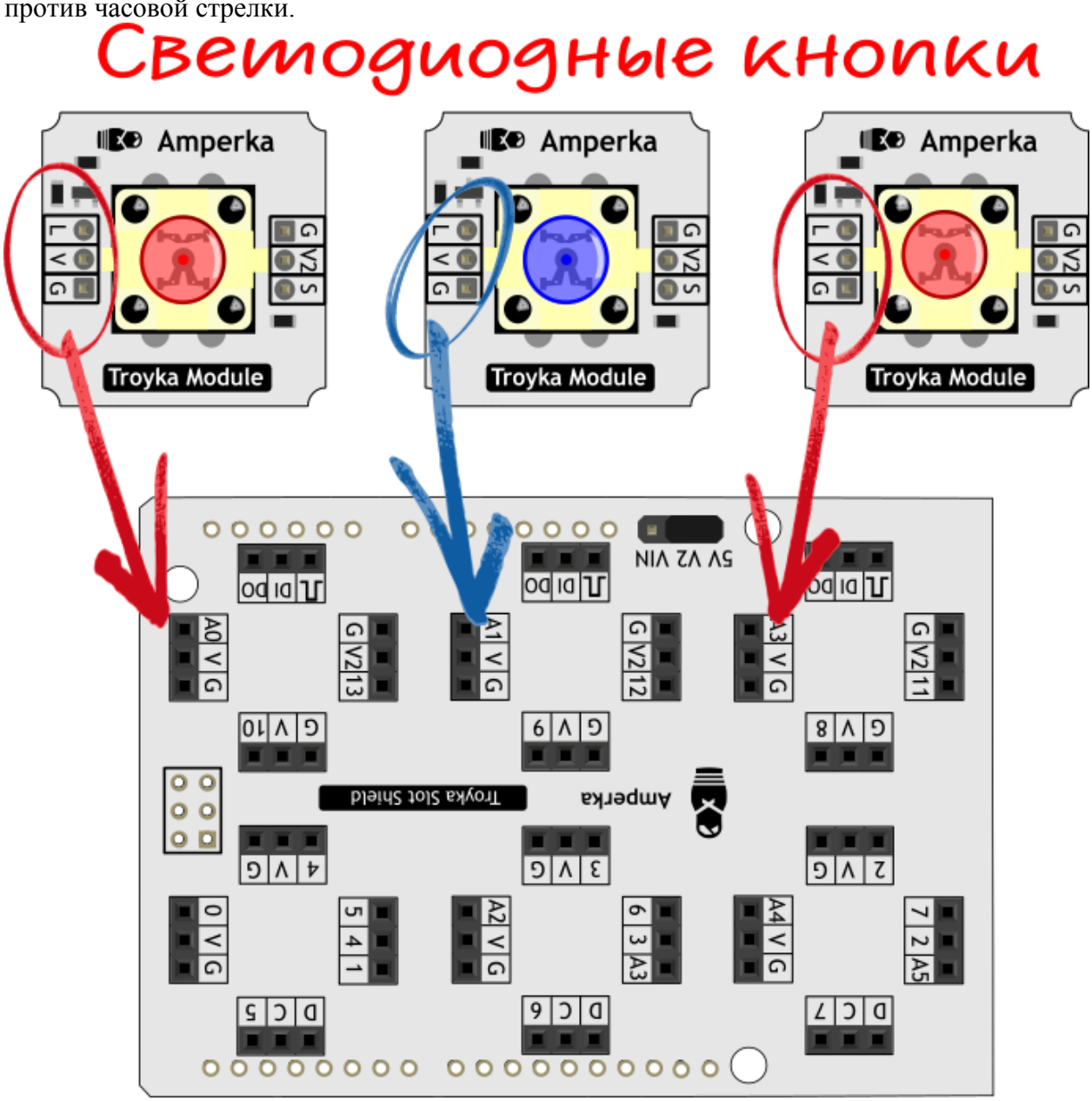
Как собрать

Установите Troyka Slot Shield на Iskra Neo

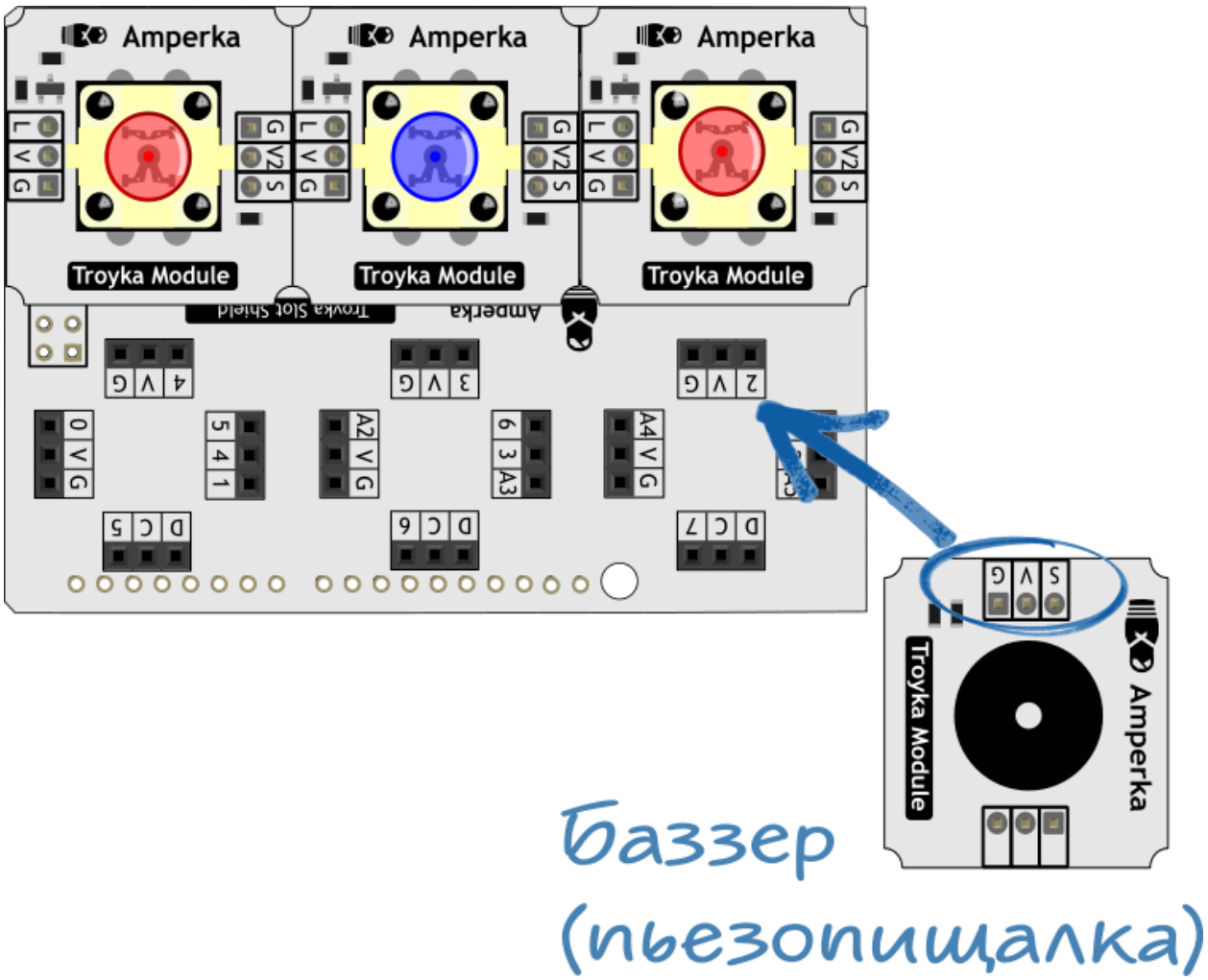


Перед следующими шагами переверните Slot Shield на 180 градусов

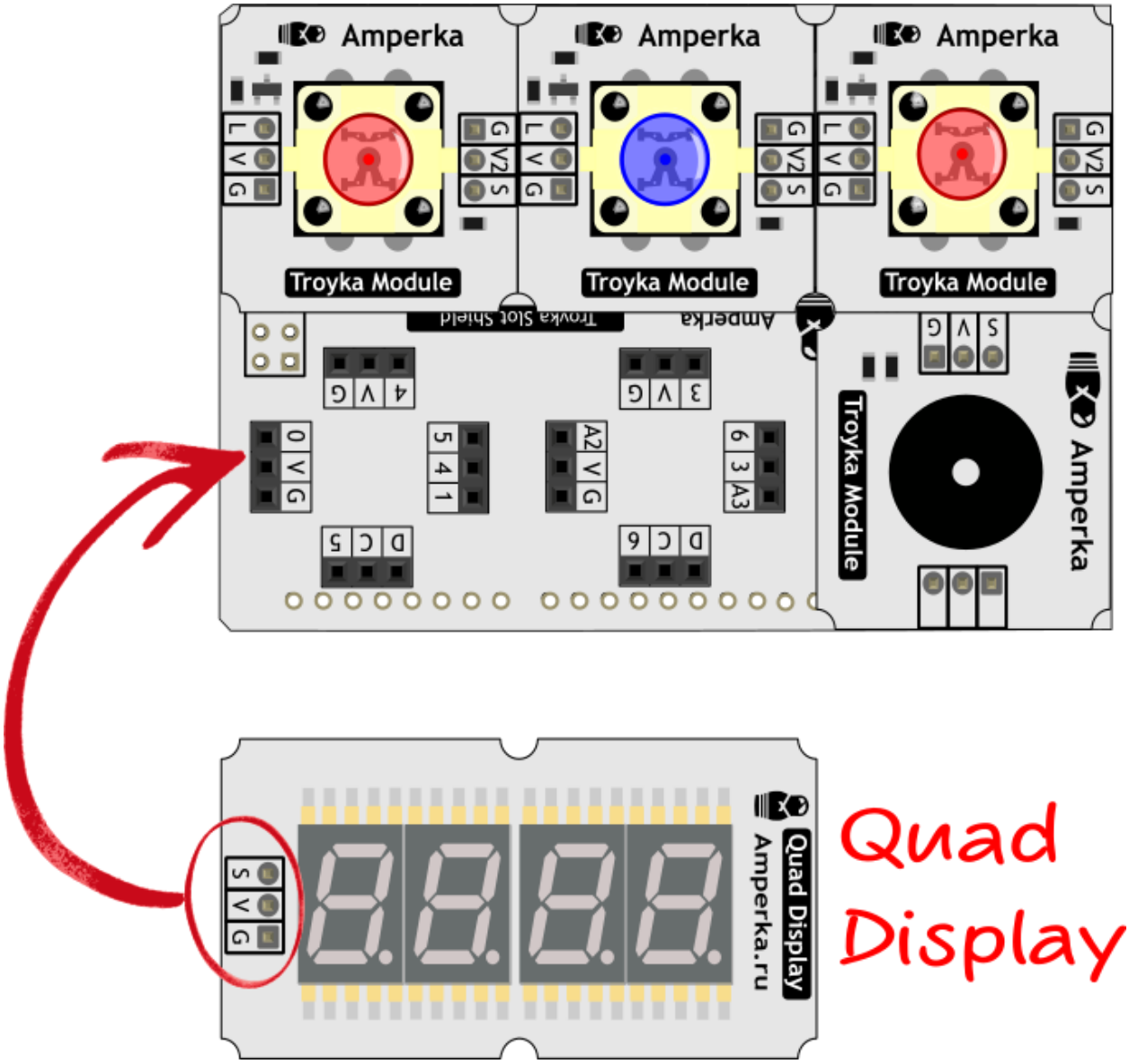
Вставьте три модуля светодиодных кнопок в верхние слоты, повернув их на 90 градусов против часовой стрелки.



Поверните модуль зуммера на 180 градусов и вставьте в левый нижний слот.



Вставьте [четырёхразрядный индикатор] в оставшиеся пустыми слоты.



Скетч

Прошейте контроллер скетчем через Arduino IDE.

SimonSays.ino

```
// Подключаем библиотеку для работы с дисплеем
#include <QuadDisplay.h>

// ноты для мелодии
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
```

```
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523

// номер цифрового пина пищалки
#define BUZZER_PIN          2
// номер цифрового пина дисплея
#define DISPLAY_PIN         0

// номера цифровых пинов кнопок
#define BUTTON_1            13
#define BUTTON_2            12
#define BUTTON_3            11

// номера цифровых пинов светодиодов
#define LED_1               A0
#define LED_2               A1
#define LED_3               A3

// зерно для генератора случайных чисел
#define ANALOG_PIN_FOR_RND  A5

// массив мелодии при загрузке программы
int startTune[] = {NOTE_C4, NOTE_F4, NOTE_C4, NOTE_F4, NOTE_C4,
NOTE_F4, NOTE_C4,
                   NOTE_F4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_F4,
NOTE_G4};
// массив длительности нот для мелодии при старте
int durationStartTune[] = {100, 200, 100, 200, 100, 400, 100, 100, 100,
100, 200, 100, 500};
// массив мелодии при выигрыше
int gameWin[] = {NOTE_C4, NOTE_C4, NOTE_G4, NOTE_C5, NOTE_G4, NOTE_C5};
// массив длительности нот для мелодии при выигрыше
int durationGameWin[] = {100, 100, 100, 300, 100, 300};

// присваиваем ноту для каждого светодиода
int ledBeep[] = {NOTE_G3, NOTE_A3, NOTE_B3};

// массив кнопок
int button[] = {BUTTON_1, BUTTON_2, BUTTON_3};
// массив светодиодов
int ledPin[] = {LED_1, LED_2, LED_3};

// переменная для хранения очков
int score = 0;
// массив случайно сгенерированых нажатий
int randomArray[100];
// массив нажатия кнопок
int inputArray[100];

void setup()
{
  // назначаем пины светодиодов и кнопок в режиме выхода
  for (int i = 0; i < 3; i++) {
```

```
    pinMode(ledPin[i], OUTPUT);
    pinMode(button[i], INPUT);
  }
  // функция для генерации случайных чисел
  randomSeed(analogRead(ANALOG_PIN_FOR_RND));

  for (int thisNote = 0; thisNote < 13; thisNote++) {
    // играем следующую ноту
    tone(BUZZER_PIN, startTune[thisNote], durationStartTune[thisNote]);
    // в зависимости от итерации цикла, включаем светодиоды
    if (thisNote == 0 || thisNote == 2 || thisNote == 4 || thisNote ==
6) {
      digitalWrite(ledPin[0], HIGH);
    } else if (thisNote == 1 || thisNote == 3 || thisNote == 5 ||
               thisNote == 7 || thisNote == 9 || thisNote == 11) {
      digitalWrite(ledPin[1], HIGH);
    } else if (thisNote == 8 || thisNote == 12) {
      digitalWrite(ledPin[2], HIGH);
    } else if (thisNote == 10) {
      digitalWrite(ledPin[3], HIGH);
    }
    delay(durationStartTune[thisNote]);
    // гасим все светодиоды
    allLedWrite(LOW);
    // задержка между нотами
    delay(25);
  }

  // ждём 1 секунду перед началом первого уровня
  delay(1000);
}

void loop()
{

  for (int y = 0; y <= 99; y++) {
    // отображаем на дисплее текущее количество очков
    displayInt(DISPLAY_PIN, score);
    // зажигаем все светодиоды
    allLedWrite(HIGH);

    // победная мелодия
    for (int thisNote = 0; thisNote < 6; thisNote ++) {
      // играем следующую ноту
      tone(BUZZER_PIN, gameWin[thisNote], durationGameWin[thisNote]);
      delay(durationGameWin[thisNote]);
      // задержка между нотами
      delay(25);
    }
    // гасим все светодиоды
    allLedWrite(LOW);

    delay(1000);

    // переход на новый уровень
    for (int i = score; i <= score; i++) {
      // генерируем массив случайных чисел в диапазоне от 0 до 2
      randomArray[i] = random(0, 3);
      // зажигаем по очереди случайный светодиод
      for (int j = 0; j <= score; j++) {
        showLed(randomArray[j]);
      }
    }
    waitPressButton();
```

```
  }
}

// функция включения или отключение всех светодиодов
void allLedWrite(bool state)
{
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledPin[i], state);
  }
}

// зажигаем светодиод на определённый период
void showLed(int led)
{
  // зажигаем выбранный светодиод
  digitalWrite(ledPin[led], HIGH);
  // подаём звуковой сигнал о нажатом светодиоде
  tone(BUZZER_PIN, ledBeep[led], 100);
  delay(400);
  // гасим выбранный светодиод
  digitalWrite(ledPin[led], LOW);
  delay(100);
}

void waitPressButton()
{
  int i = 0;
  // пока количество нажатий на кнопки не превысило количество очков
  while (i <= score) {
    for (int j = 0; j < 3; j++) {
      // вызываем функцию считывания каждой кнопки по номеру
      if (buttonIsPressed(j, i)) {
        ++i;
      }
    }
  }
  delay(500);
  // увеличиваем текущее количество очков на единицу
  score++;
}

bool buttonIsPressed(int currentButton, int currentScore)
{
  bool result = false;
  // если клавиша нажата
  if (!digitalRead(button[currentButton])) {
    // зажигаем светодиод, нажатой кнопки
    digitalWrite(ledPin[currentButton], HIGH);
    // подаём звуковой сигнал о нажатом светодиоде
    tone(BUZZER_PIN, ledBeep[currentButton], 100);
    delay(200);
    // гасим светодиод, нажатой кнопки
    digitalWrite(ledPin[currentButton], LOW);
    // присваиваем массиву кода кнопок, текущий номер кнопки
    inputArray[currentScore] = currentButton;
    delay(250);
    // если текущая нажатая клавиша не совпадает со случайно
сгенерированной
    if (inputArray[currentScore] != randomArray[currentScore]) {
      // нажата не правильная кнопка, вызываем функцию проигрыша
      fail();
    }
    // кнопка была нажата верно
    result = true;
```

```
  }
  // возвращаем результат
  return result;
}

void fail()
{
  for (int y = 0; y <= 2; y++) {
    // зажигаем все светодиоды
    allLedWrite(HIGH);
    // издаём звуковой сигнал
    tone(BUZZER_PIN, NOTE_G3, 300);
    delay(200);
    // гасим все светодиоды
    allLedWrite(LOW);
    // издаём звуковой сигнал
    tone(BUZZER_PIN, NOTE_C3, 300);
    delay(200);
  }
  delay(500);
  // присваиваем переменной score значение «-1»
  // это будет сброс и игра начнётся сначала
  score = -1;
}
```

Что дальше?

Хотите собрать другой девайс? Выберите своё будущее устройство из списка проектов на Slot Shield.

FAQ

- Где скачать библиотеку для для работы с дисплеем?
- Как установить библиотеку?