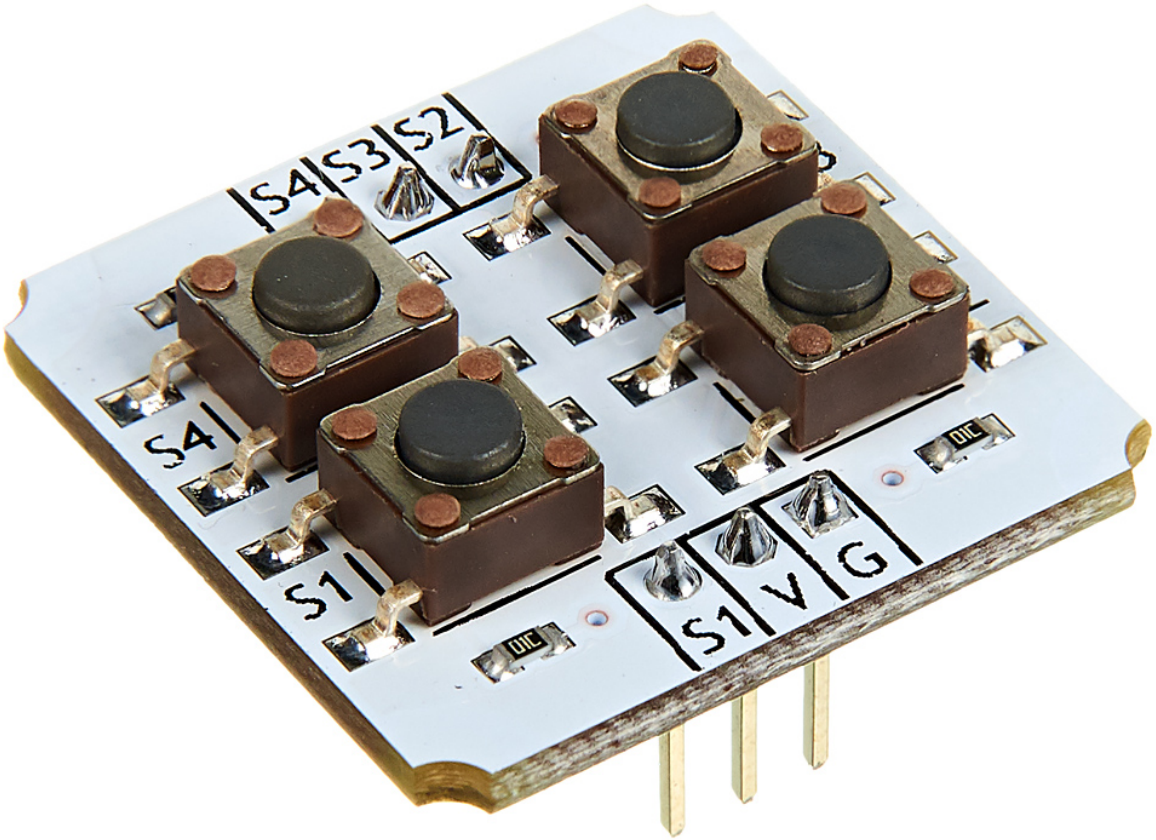
Четырёхкнопочная клавиатура (Troyka-модуль)

Клавиатурный модуль — это четыре тактовые кнопки собранные в один Troyka-модуль. Вам не придётся возиться с пайкой и беспокоиться о подтягивающих резисторах для каждой кнопки — они уже есть на модуле.

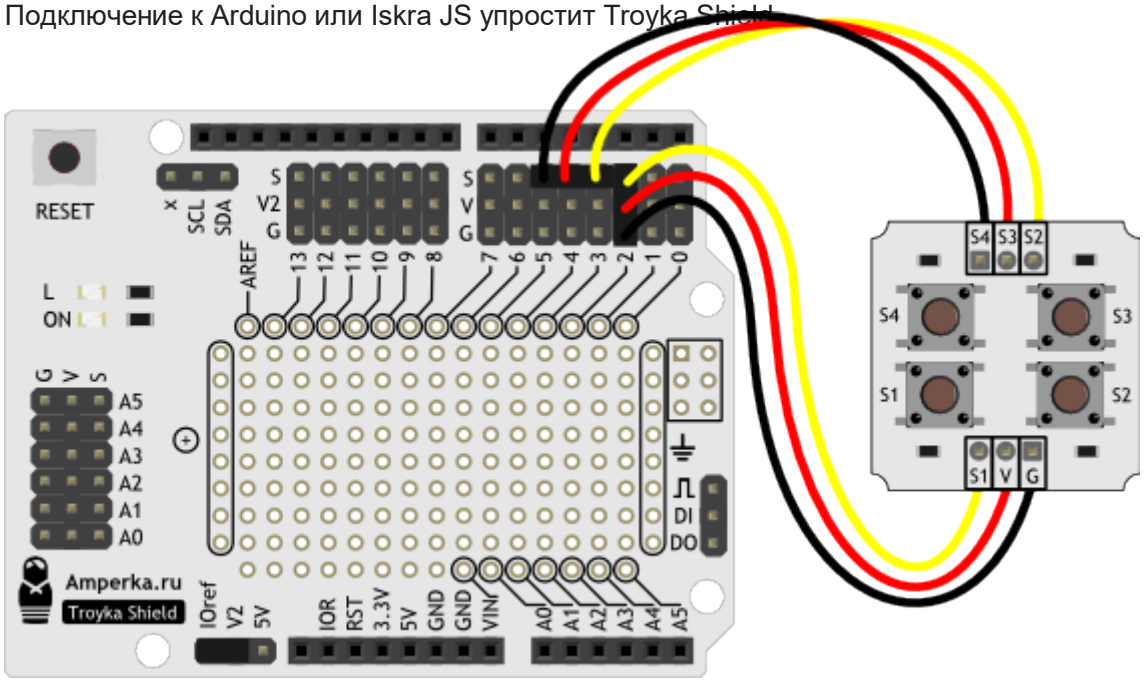


Подключение и настройка

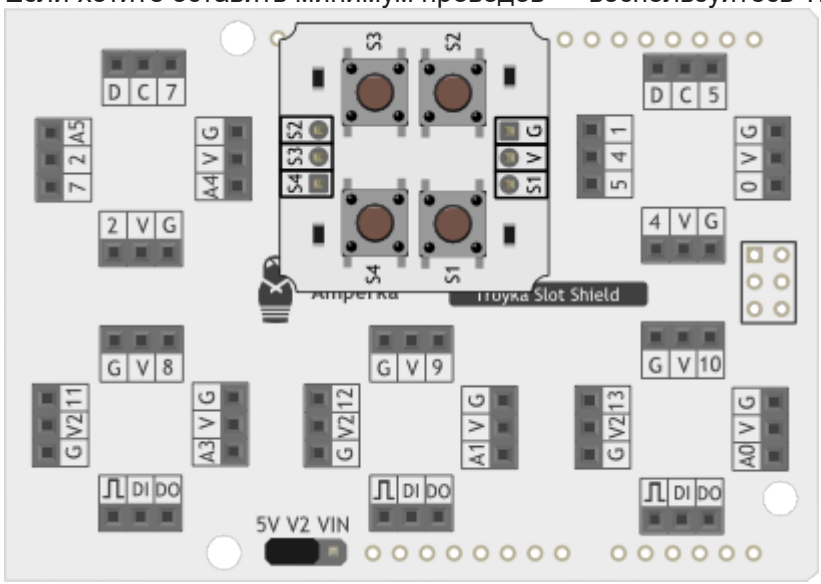
Тактовая кнопка — это простой цифровой датчик. Пока кнопка не зажата, датчик отдаёт логическую единицу, когда кнопка зажата — логический ноль. На модуле клавиатурном модуле собрано сразу четыре таких кнопки.

У кнопок общее питание и отдельные сигнальные пины. Это позволило вдвое сократить количество необходимых проводов — для подключения четырёх кнопок их понадобится всего шесть.

Подключение к Arduino или Iskra JS упростит Troyka Shield.



Если хотите оставить минимум проводов — воспользуйтесь Troyka Slot Shield.



Примеры работы

Пример работы для Arduino

Выведем в Serial-порт текущее состояние всех кнопок и будем обновлять его каждые 100 миллисекунд.

quad_switch.ino

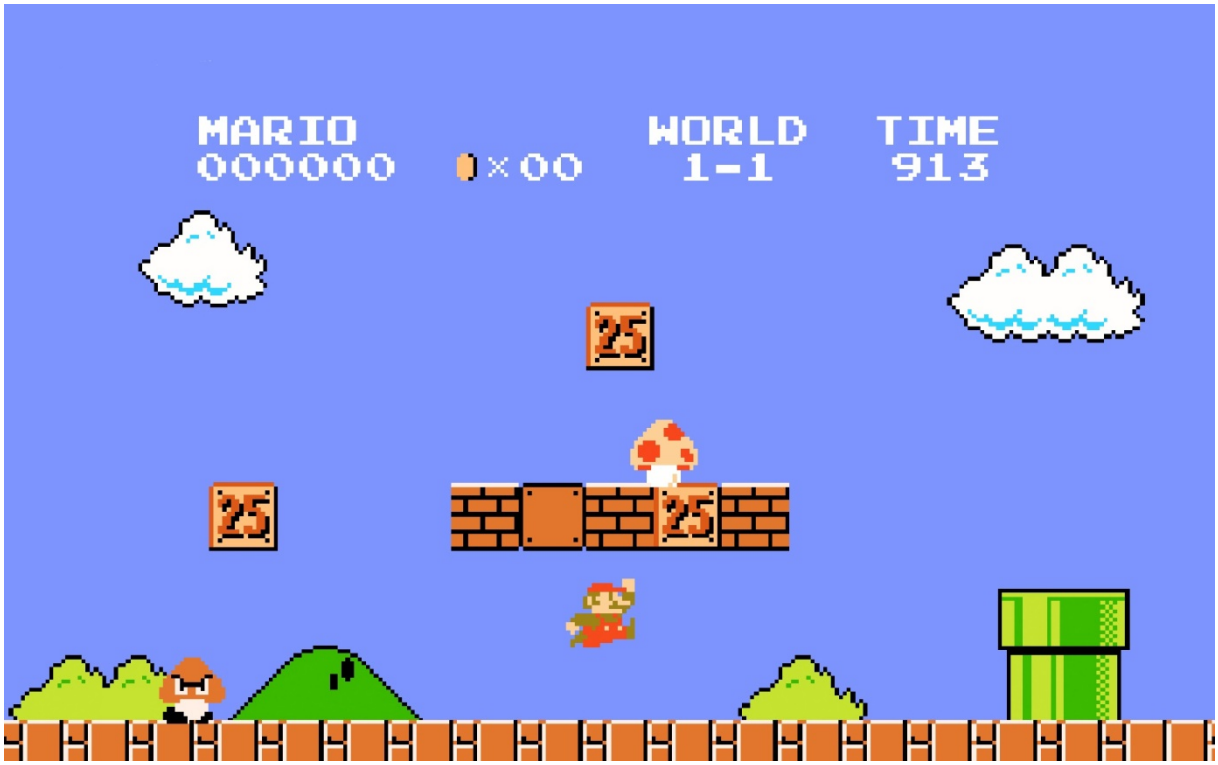
```
// даём разумные имена пинам кнопок
#define BUTTON_PIN_1  2
#define BUTTON_PIN_2  3
#define BUTTON_PIN_3  4
#define BUTTON_PIN_4  5
```

```
void setup()
{
  // открываем монитор Serial-порта
  Serial.begin(9600);
}

void loop()
{
  // считываем состояние пина
  int buttonState1 = digitalRead(BUTTON_PIN_1);
  int buttonState2 = digitalRead(BUTTON_PIN_2);
  int buttonState3 = digitalRead(BUTTON_PIN_3);
  int buttonState4 = digitalRead(BUTTON_PIN_4);
  // выводим значения состояния кнопок в Serial-порт
  Serial.print(buttonState1);
  Serial.print("\t");
  Serial.print(buttonState2);
  Serial.print("\t");
  Serial.print(buttonState3);
  Serial.print("\t");
  Serial.println(buttonState4);
  delay(100);
}
```

На экране вы увидите четыре столбика бегущих единиц. Порядковый номер столбика соответствует номеру кнопки. При нажатии на кнопки, единицы в столбиках сменятся нулями.

Эмуляция игрового джойстика на Arduino



Помните «лихие девяностые», когда Dendy и Sega открыли окно в новую реальность? С помощью наших модулей вы сможете вспомнить старые добрые игры.

Соберите игровой контроллер на Troyka Pad 1×4. Возьмите 3D-джойстик и четырёхкнопочный модуль и пару светодиодов.

Пример работает только с платами, которые определяются ПК как HID устройство.

Скетч подойдет для:

- Iskra Neo
- Arduino Leonardo
- Arduino Leonardo ETH
- Arduino Micro
- Arduino Due

joystik keyboard.ino

```
// библиотека для эмуляции клавиатуры
#include "Keyboard.h"
// номера цифровых пинов кнопок
#define BUTTON_PIN_1  2
#define BUTTON_PIN_2  3
#define BUTTON_PIN_3  4
#define BUTTON_PIN_4  5
// номера аналоговых пинов джойстика
#define JOYSTICK_X    A2
#define JOYSTICK_Y    A5
// номера цифровых пинов светодиодов
#define LED_GREEN     A0
#define LED_RED       A1

// переменные для хранения состояния сенсоров на джойстике
int joystickValueX = 0;
int joystickValueY = 0;
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;

// переменные для хранения временных интервалов
long millisJoystickX = 0;
long millisJoystickY = 0;
long millisButtom1 = 0;
long millisButtom2 = 0;
long millisButtom3 = 0;
long millisButtom4 = 0;

bool state1 = true;
bool state2 = true;
bool state3 = true;
bool state4 = true;

void setup()
{
  // открываем монитор Serial-порта
```

```
  Serial.begin(9600);
  // инициализируем эмуляцию клавиатуры
  Keyboard.begin();
  // назначаем пины светодиодов в режим выхода
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
}

void loop()
{
  // считывание состояние всех сенсоров на джойстике
  joystickRead();
  // проверка на зажигание светодиодов
  joystickLed();
  // проверка на нажатие клавиш
  joystickPress();
  // проверка на отпускания клавиш
  joystickRelease();
}

void joystickRead() {
  // считываем состояние джойстика и пинов кнопок
  joystickValueX = analogRead(JOYSTICK_X);
  joystickValueY = analogRead(JOYSTICK_Y);
  buttonState1 = digitalRead(BUTTON_PIN_1);
  buttonState2 = digitalRead(BUTTON_PIN_2);
  buttonState3 = digitalRead(BUTTON_PIN_3);
  buttonState4 = digitalRead(BUTTON_PIN_4);
}

void joystickLed() {
   // если нажата хотя бы одна кнопка
  if (!buttonState1 || !buttonState2 || !buttonState3 || !buttonState4 ) {
    // зажигаем зелёный светодиод
    digitalWrite(LED_GREEN, HIGH);
  } else {
    // гасим зелёный светодиод
    digitalWrite(LED_GREEN, LOW);
  }
  // если джойстик отклонен относительно своего обычного состояния
  if (joystickValueX > 600 || joystickValueX < 400 || joystickValueY > 600 ||
joystickValueY < 400 ) {
    // зажигаем красный светодиод
    digitalWrite(LED_RED, HIGH);
  } else {
    // гасим красный светодиод
    digitalWrite(LED_RED, LOW);
  }
}
```

```
void joystickPress() {
  // если джойстик отклонен вправо
  if (joystickValueX > 600) {
    // эмулируем нажатие и удерживание клавиши «d»
    Keyboard.press('d');
    // запоминаем текущее время
    millisJoystickX = millis();
  }
  // если джойстик отклонен влево
  if (joystickValueX < 400) {
    // эмулируем нажатие и удерживание клавиши «a»
    Keyboard.press('a');
    // запоминаем текущее время
    millisJoystickX = millis();
  }
  // если джойстик отклонен вниз
  if (joystickValueY < 400) {
    // эмулируем нажатие и удерживание клавиши «s»
    Keyboard.press('s');
    // запоминаем текущее время
    millisJoystickY = millis();
  }
  // если джойстик отклонен вверх
  if (joystickValueY > 600) {
    // эмулируем нажатие и удерживание клавиши «w»
    Keyboard.press('w');
    // запоминаем текущее время
    millisJoystickY = millis();
  }
  // если нажата кнопка «1»
  // и не прошёл заданный интервал времени с предыдущего нажатия данной кнопки
  if (!buttonState1 && state1 == 0) {
    // запоминаем состояние кнопки «1»
    state1 = true;
    // эмулируем нажатие и удерживание клавиши «k»
    Keyboard.press('k');
    // запоминаем текущее время
    millisButtom1 = millis();
  }
  // если нажата кнопка «2»
  // и не прошёл заданный интервал времени с предыдущего нажатия данной кнопки
  if (!buttonState2 && state2 == 0) {
    // запоминаем состояние кнопки «2»
    state2 = true;
    // эмулируем нажатие и удерживание клавиши «l»
    Keyboard.press('l');
    // запоминаем текущее время
    millisButtom2 = millis();
  }
  // если нажата кнопка «3»
```

```
  // и не прошёл заданный интервал времени с предыдущего нажатия данной кнопки
  if (!buttonState3 && state3 == 0) {
    // запоминаем состояние кнопки «y»
    state3 = true;
    // эмулируем нажатие и удерживание клавиши «y»
    Keyboard.press('y');
    // запоминаем текущее время
    millisButtom3 = millis();
  }
  // если нажата кнопка «4»
  if (!buttonState4 && state4 == 0) {
    // запоминимаем состояние кнопки «u»
    state4 = true;
    // эмулируем нажатие и удерживание клавиши «u»
    Keyboard.press('u');
    // запоминаем текущее время
    millisButtom4 = millis();
  }
}

void joystickRelease() {
  // если прошёл заданный интервал эмуляции нажатие клавиши
  // при отклонении джойстика влево или вправо
  if (millis() - millisJoystickX > 5) {
    // эмулируем отпускание клавиш «a» и «d»
    Keyboard.release('a');
    Keyboard.release('d');
  }
  // если прошёл заданный интервал эмуляции нажатие клавиши
  // при отклонении джойстика вверх или вниз
  if (millis() - millisJoystickY > 5) {
    // эмулируем отпускание клавиш «w» и «s»
    Keyboard.release('w');
    Keyboard.release('s');
  }
  // если прошёл заданный интервал эмуляции нажатие клавиши «k»
  if (millis() - millisButtom1 > 5) {
    // эмулируем отпускание клавиш «k»
    Keyboard.release('k');
    state1 = 0;
  }
  // если прошёл заданный интервал эмуляции нажатие клавиши «l»
  if (millis() - millisButtom2 > 100) {
    // эмулируем отпускание клавиш «l»
    Keyboard.release('l');
    state2 = 0;
  }
  // если прошёл заданный интервал эмуляции нажатие клавиши «y»
  if (millis() - millisButtom3 > 5) {
    // эмулируем отпускание клавиш «y»
```

```
    Keyboard.release('y');
    state3 = 0;
  }
  // если прошёл заданный интервал эмуляции нажатие клавиши «u»
  if (millis() - millisButtom4 > 5) {
    // эмулируем отпускание клавиш «u»
    Keyboard.release('u');
    state4 = 0;
  }
}
```

Прошейте плату и запускайте эмулятор с любимой игрой.

Пример программы для Iskra JS

Поймаем нажатие кнопок с помощью библиотеки для Iskra JS.

quad_switch.js

```
var myButton1 = require('@amperka/button')
  .connect(P2, {
  });

var myButton2 = require('@amperka/button')
  .connect(P3, {
  });

var myButton3 = require('@amperka/button')
  .connect(P4, {
  });

var myButton4 = require('@amperka/button')
  .connect(P5, {
  });

myButton1.on('press', function() {
  console.log("Button '1' is press");
});

myButton2.on('press', function() {
  console.log("Button '2' is press");
});

myButton3.on('press', function() {
  console.log("Button '3' is press");
});

myButton4.on('press', function() {
  console.log("Button '4' is press");
});
```

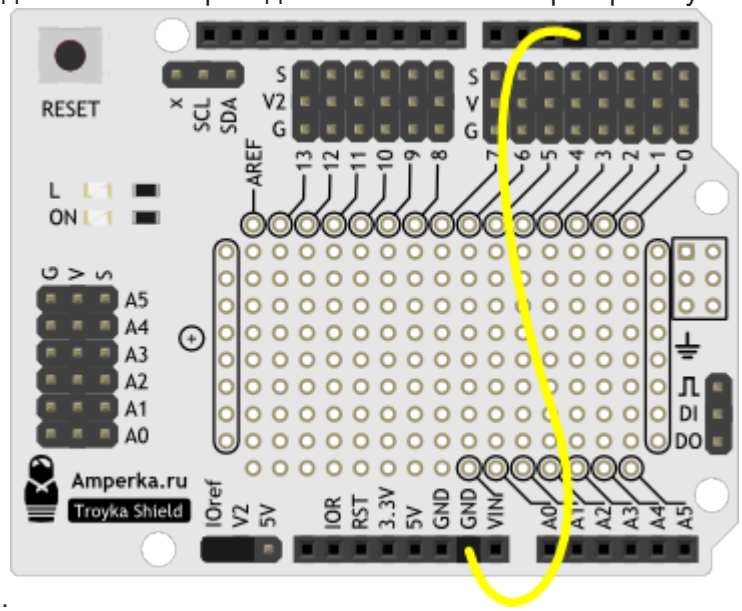
Нажмите любую кнопку — в консоли появится сообщение:

```
Button 'N' is press"
```

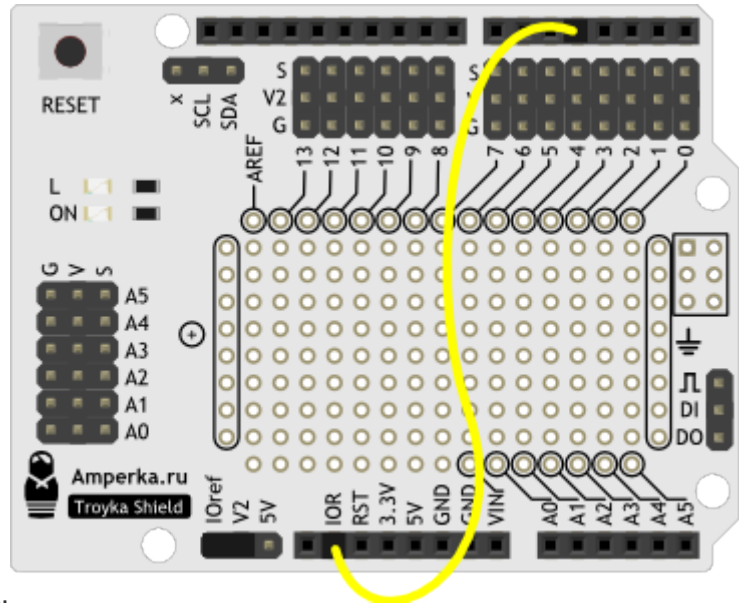
где N — номер нажатой кнопки.

Что-то пошло не так

Если у при нажатии кнопки значения не меняются, проверьте работу порта управляющей платы с помощью соединительного провода «папа-папа». Поверим работу кнопки на `4` пине.



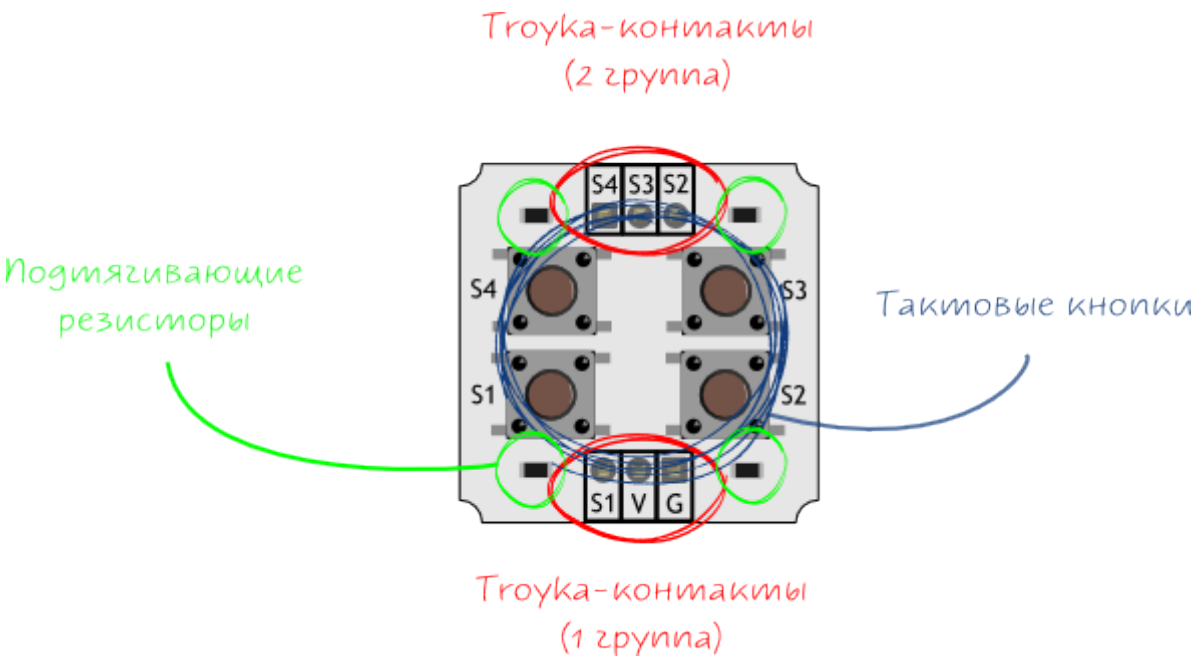
- `4+G` — кнопка нажата.



- `4+V(IOREF)` — кнопка отжата.

Если значения меняются — ваша кнопка неисправна. Обратитесь в техническую поддержку.

Элементы платы



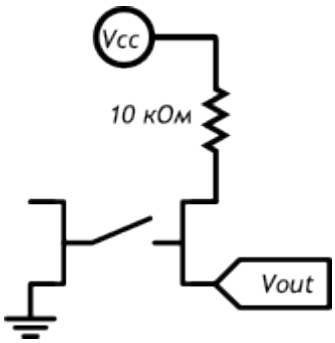
Тактовая кнопка

Тактовая кнопка без фиксатора — простой механизм, замыкающий цепь при нажатии на толкатель.

Подтягивающий резистор

В модуль входят четыре независимые кнопки подключённые по одной схеме.

Пока кнопка нажата, выходное напряжение на сигнальном пине `S` = `LOW`. Когда кнопка отпущена, провода работают как антенна и набирают наведённый сигнал — на пине `S` появляются «шумы». Эти шумы легко устранить, добавив в цепь резистор на `10 кОм`.



Контакты подключения 3-проводных шлейфов

1 группа

- Земля (G) — соедините с землёй микроконтроллера.
- Питание (V) — соедините с питанием микроконтроллера.
- Сигнальный (S1) — цифровой выход кнопки `S1`. Подключите к любому цифровому пину микроконтроллера.

2 группа

- Сигнальный (S2) — цифровой выход кнопки `S2`. Подключите к любому цифровому пину микроконтроллера.

- Сигнальный (S3) — цифровой выход кнопки `S3`. Подключите к любому цифровому пину микроконтроллера.
- Сигнальный (S4) — цифровой выход кнопки `S4`. Подключите к любому цифровому пину микроконтроллера.

Характеристики

- Сопротивление изолятора кнопки: 100 МОм
- Рабочий ток кнопки: 50 мА
- Сопротивление подтягивающих резисторов: 10 кОм
- Рабочее напряжение: 3,3–12 В
- Габариты: 25,4×25,4 мм