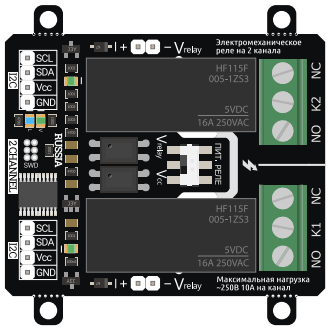
Модуль реле, 2 канала, FLASH-I2C



Общие сведения:

Модуль реле на 2 канала, I2C, Flash - является устройством коммутации, которое позволяет подключать и отключать устройства к сети переменного тока до 250В. При этом устройства подключённые через выходные контакты модуля, не должны потреблять более 10А переменного тока (на каждый канал).

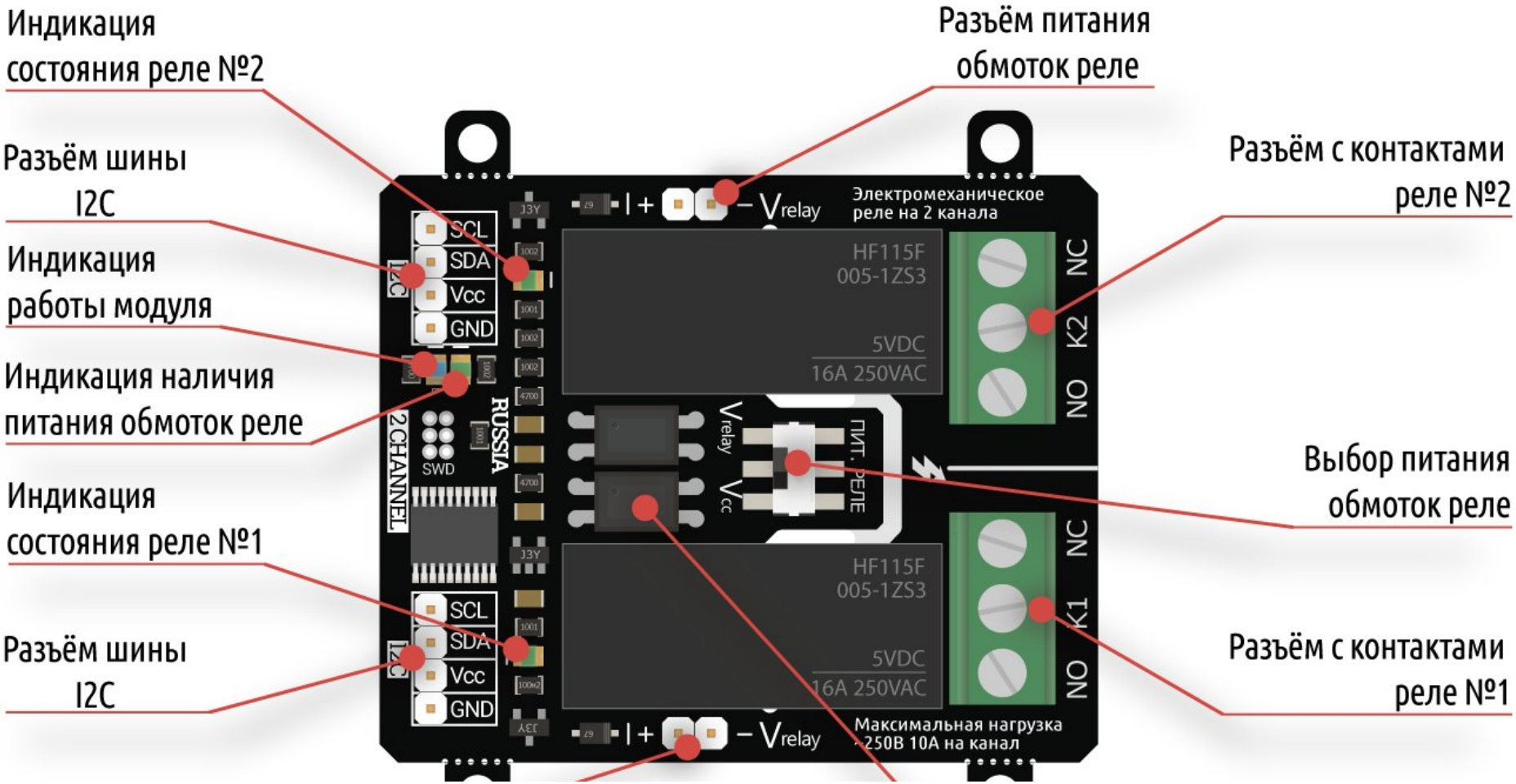
Управление модулем осуществляется по шине I2C. Модуль относится к линейке «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

Модуль построен на базе микроконтроллера STM32F030F4, снабжен собственным стабилизатором напряжения, двумя электромеханическими реле и переключателем выбора питания обмоток реле. Модуль можно использовать в любых проектах где требуется управлять устройствами с напряжением питания до 250В и потреблением переменного тока до 10А.

Спецификация:

- Напряжение питания логики: 5 В (постоянного тока).
- Напряжение питания обмоток реле: 5 В (постоянного тока).
- Ток потребляемый логикой модуля: до 20 мА.
- Ток потребляемый обмоткой реле: до 80 мА (на каждый канал).
- Коммутируемое напряжение: до 250 В (переменного тока).
- Коммутируемый ток: до 10 А (на каждый канал).
- Количество каналов: 2.

- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: 3,3 В (толерантны к 5 В, подтянуты к 5 В).
- Рабочая температура: от -40 до +65 °C.
- Габариты с креплением: 55 x 55 мм.
- Габариты без креплений: 55 x 45 мм.
- Вес: 40 г.



Подключение:

Перед подключением модуля ознакомьтесь с разделом "Смена адреса модуля на шине I2C" в данной статье.

В левой части платы расположены два разъема для подключения модуля к шине **I2C**. Шина подключается к любому разъему **I2C**, а второй разъем можно использовать для подключения следующего модуля реле, или других устройств.

- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **Vcc** - вход питания модуля 5В.
- **GND** - общий вывод питания.

В правой части платы расположены два разъема: **K1** и **K2**, это выходы, через контакты которых подключаются силовые устройства к сети переменного тока до 250В. Устройства не должны потреблять более 10А (на каждый канал).

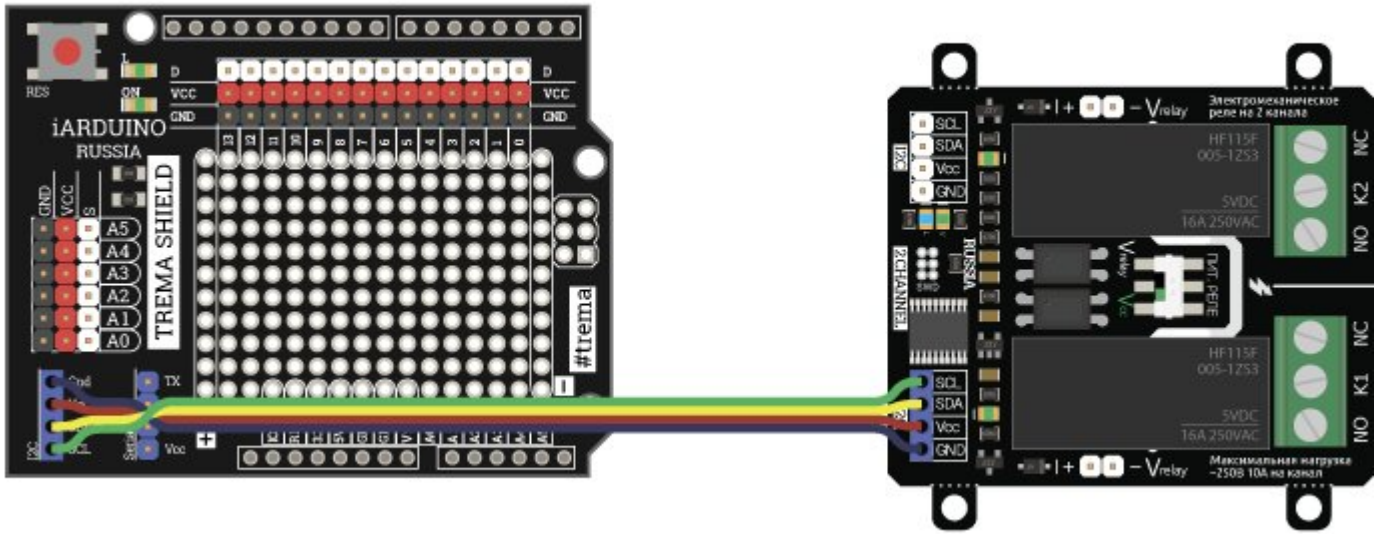
- **K1** - разъём первого реле с контактами «NC» (Normally Closed) - нормально замкнуты и «NO» (Normally Open) - нормально разомкнуты.
- **K2** - разъём второго реле с контактами «NC» (Normally Closed) - нормально замкнуты и «NO» (Normally Open) - нормально разомкнуты.

Способ - 1: Используя провода и Piranha UNO

Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO, переключатель питания реле в положении Vcc - питание микроконтроллера и обмоток реле общее

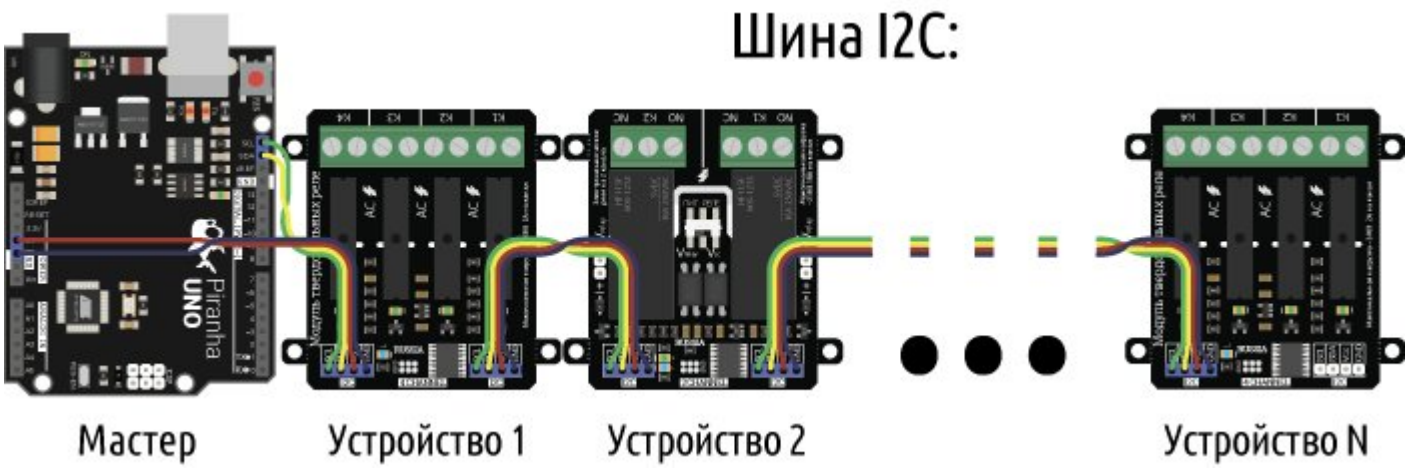
Способ - 2: Используя проводной шлейф и Trema Shield

Используя 4-х проводной шлейф подключаем к Trema Shield, переключатель питания реле в положении Vcc - питание микроконтроллера и обмоток реле общее



Подключение нескольких устройств

Благодаря шине I2C возможно подключение более 100 устройств:



Подключение нагрузки

Внимание! При включении и отключении нагрузок в сети возникают импульсные помехи, которые могут приводить к зависанию управляющего микроконтроллера или шины I2C при использовании некачественных блоков питания или блоков питания без синфазных дросселей.

Силовые устройства подключаются к сети переменного тока (до 250В) через контакты разъемов «K1» (первое реле) или «K2» (второе реле). У каждого разъема имеются контакты «NC» - нормально замкнутые и «NO» - нормально разомкнутые.

О состоянии реле можно судить по светодиодам расположенным рядом с реле.

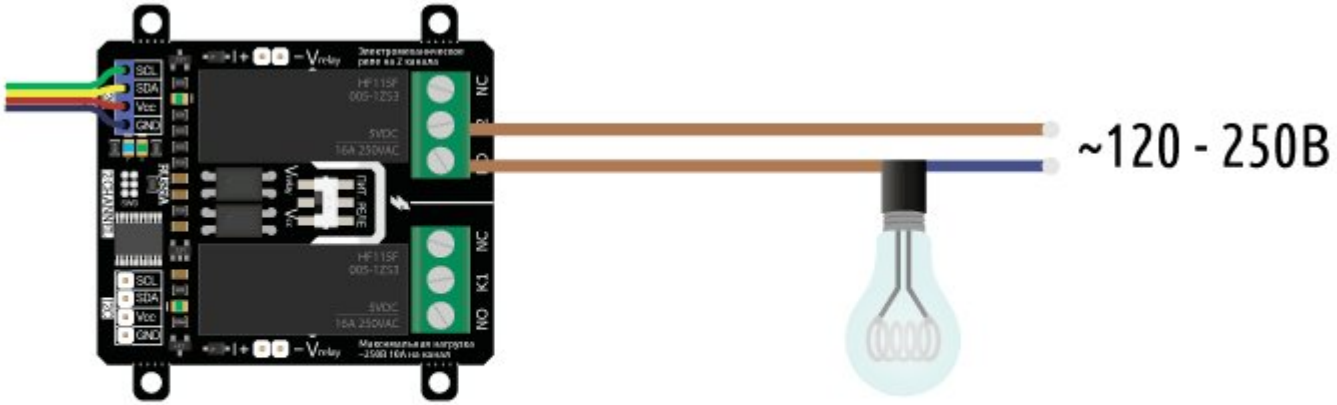
- Если светодиод выключен, значит обмотка соответствующего реле обесточена, следовательно, выводы «NC» (Normally Closed) данного реле — замкнуты, а выводы «NO» (Normally Open) - разомкнуты.
- Если светодиод светится, значит на обмотку соответствующего реле подано напряжение, следовательно, выводы «NC» данного реле —

разомкнуты, а выводы «NO» - замкнуты.

Использование нормально закрытых контактов реле:

Можно подключать нагрузку к нормально-закрытым или нормально-открытым контактам реле. Например, если нагрузку нужно иногда включать, то её лучше подключить к нормально открытым контактам.

Логическая единица (реле включено) - цепь замкнута, логический ноль (реле выключено) - цепь разомкнута:



Использование нормально открытых контактов реле:

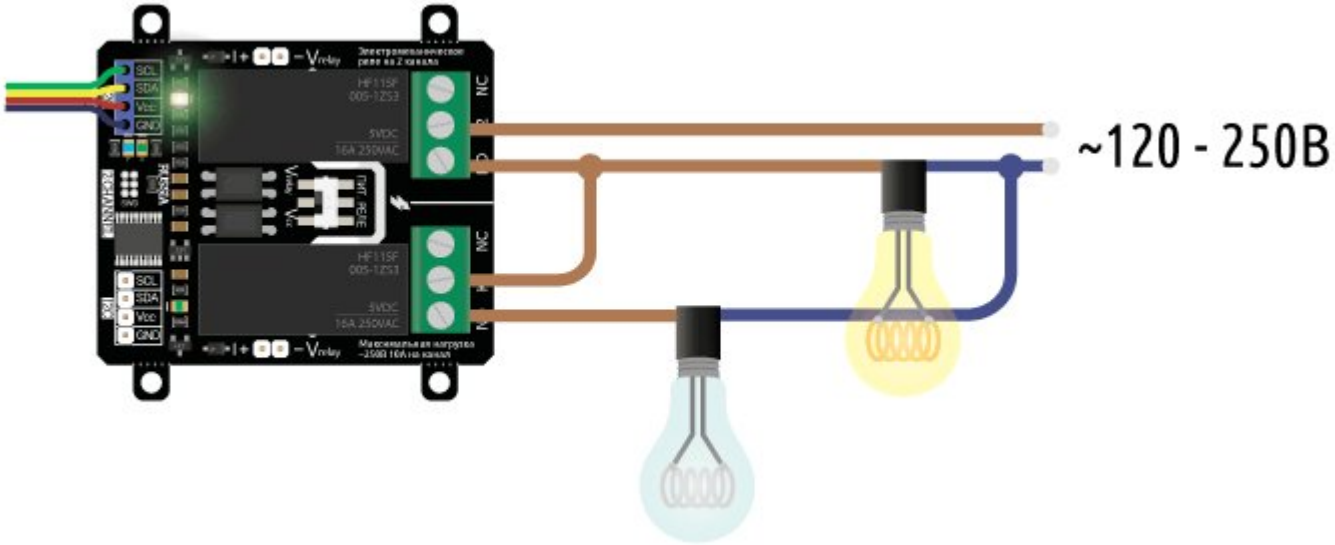
Если нагрузку нужно иногда выключать - её лучше подключить к нормально-закрытым контактам, чтобы катушка реле не потребляла ток большую часть времени:

Логическая единица (реле включено) - цепь разомкнута, логический ноль (реле выключено) - цепь замкнута:

Вариант подключения:

Можно управлять нагрузкой одного реле в зависимости от состояния второго реле

- Одно реле включено, другое выключено — горит одна лампа
- Оба реле включены — горят обе лампы
- Первое реле включено, второе выключено — обе лампы не горят



Питание:

Питание логической части

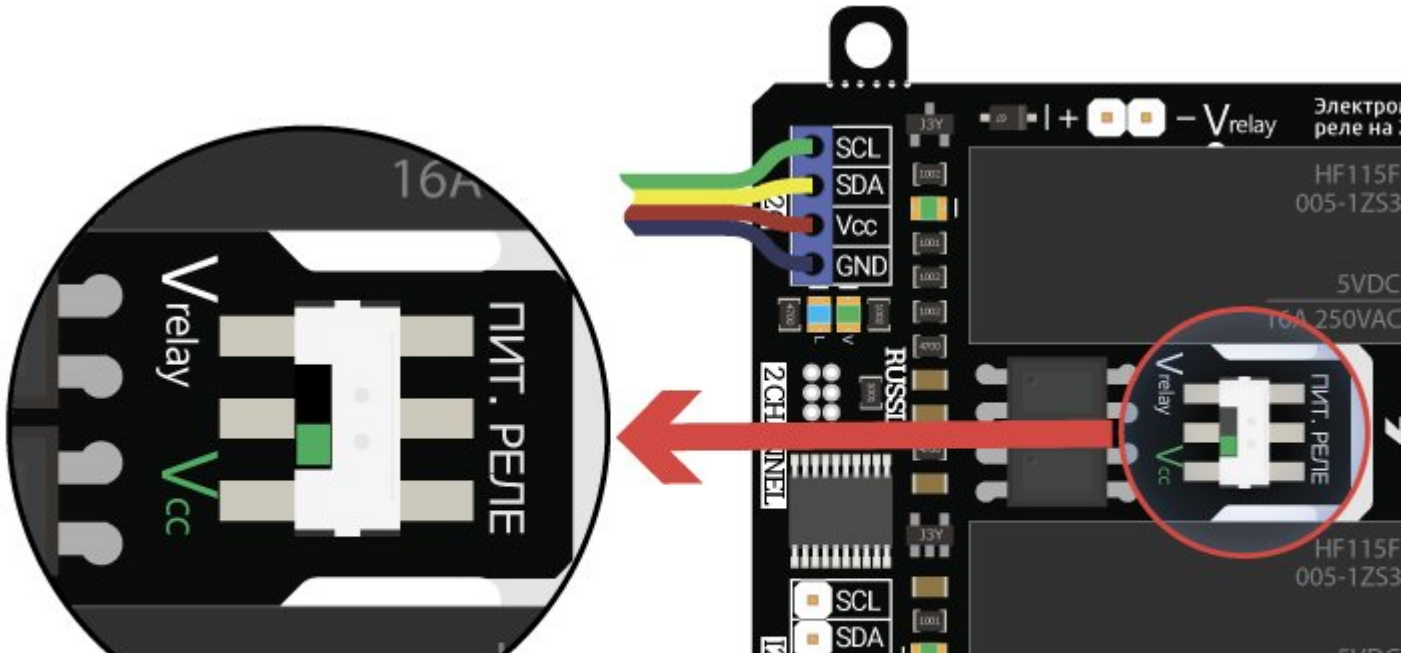
Логическая часть модуля питается от разъёма I2C, 5В постоянного тока.

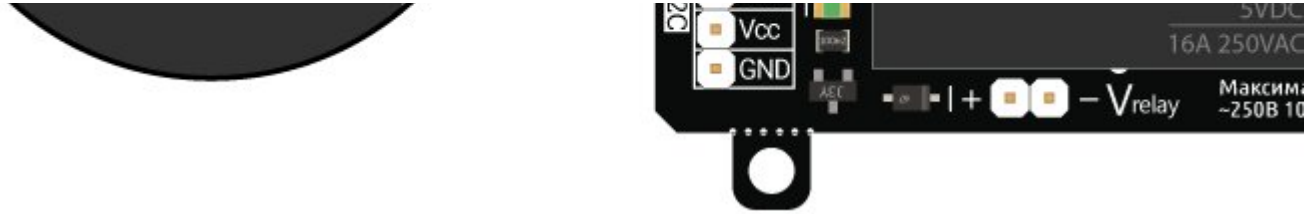
Питание обмоток реле

На модуле есть возможность выбирать способы питания обмоток реле при помощи установленного переключателя. Если переключатель находится в положении «Vrelay», то обмотки реле питаются от напряжения поступающего со входа «Vrelay», а если переключатель находится в положении «Vcc», то обмотки реле питаются от напряжения питания логики поступающего с разъема шины I2C.

Тип питания 1: общее питание логики и катушек реле

Переключатель питания реле находится в положении **Vcc**, обмотки реле питаются от разъёмов Vcc и GND шины I2C, подавать питание на **Vrelay** не требуется. Такое подключение имеет смысл, когда нужно подключить не более трёх модулей (зависит от максимального тока регулятора на управляющем контроллере):



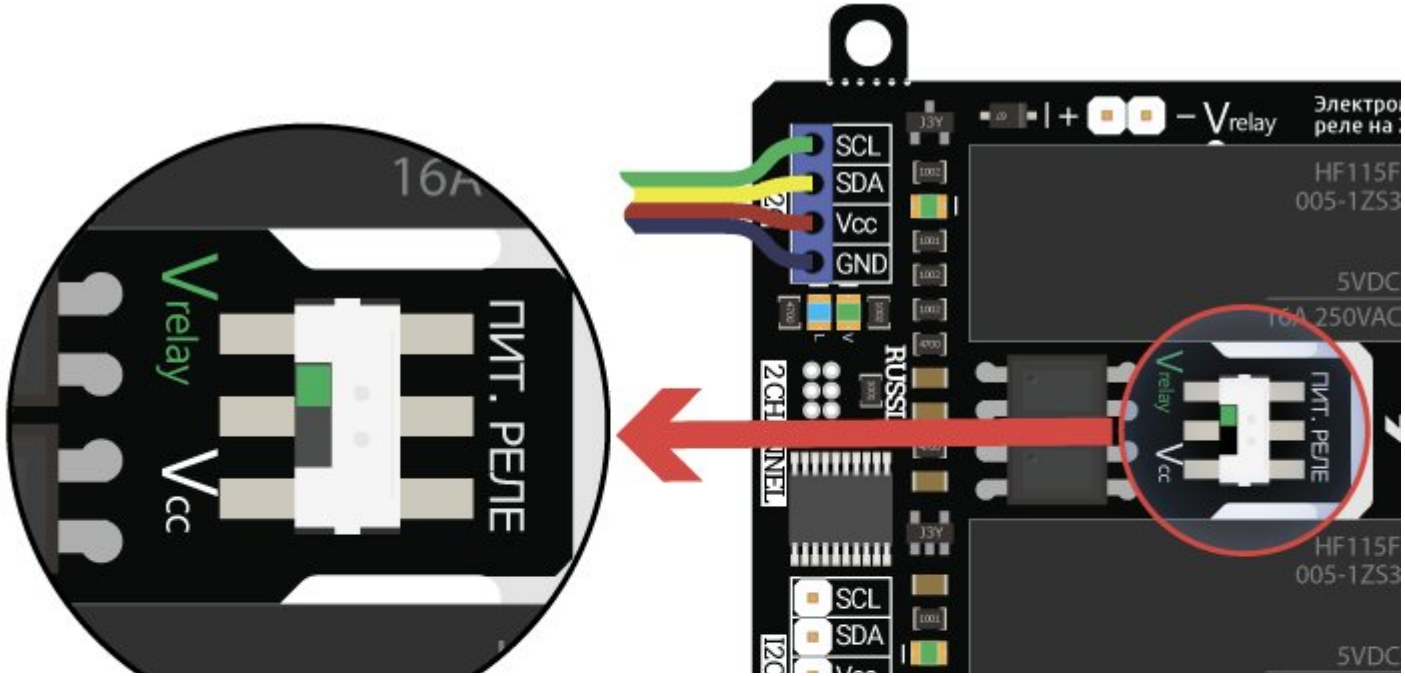


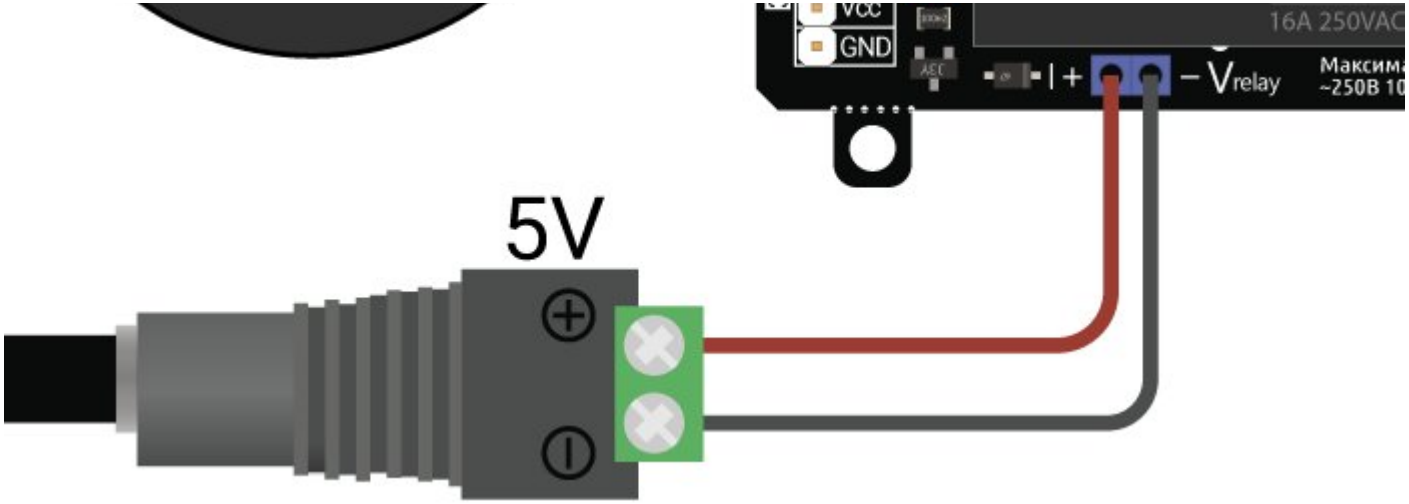
Плюсы такого подключения:

- *Подключение без дополнительного источника питания*
- *Простота и быстрота подключения*

Тип питания 2: раздельное питание логики и катушек реле

Если переключатель питания реле находится в положении **Vrelay**, то обмотки реле питаются от любого разъема Vrelay, для работы реле необходимо подать 5 В постоянного тока на разъём **Vrelay.** Такой вариант подключения оправдан, если на линиях питания Vcc и GND шины I2C находятся модули чувствительные к помехам, возникающим при включении и отключении обмоток реле.





Плюсы такого подключения:

- *Цепь питания **Vrelay** оптически развязана с питанием шины I2C (у них нет общих выводов). Шина питания контроллера не нагружена обмотками реле, самоиндукция обмоток не вызывает скачков на шине питания контроллера.*
- Можно подключить большее количество модулей реле, подобрав блок питания по необходимой нагрузке.
- Возможно увеличение расстояния от управляющего микроконтроллера до модуля

Объединение питания обмоток модулей

Благодаря тому, что выводы питания обмоток реле находятся с двух сторон платы, упрощается подключение нескольких модулей к одному источнику:

Смена адреса модуля на шине I2C:

По умолчанию все модули FLASH-I2C имеют установленный адрес 0x09. Если вы планируете подключать более 1 модуля на шину I2C, необходимо изменить адреса модулей таким образом, чтобы каждый из них был уникальным. Более подробно о том, как изменить адрес, а также о многом другом, что касается работы FLASH-I2C модулей, вы можете прочесть в **этой статье**.

В первой строке скетча необходимо записать в переменную **newAddress** адрес, который будет присвоен модулю. После этого подключите модуль к контроллеру и загрузите скетч. **Адрес может быть от 0x07 до 0x7F**.

```
uint8_t newAddress = 0x09;                                          //   Назначаемый модулю адрес (0x07 < адрес < 0x7F).
                                                                    //
#include <Wire.h>                                                   //   Подключаем библиотеку для работы с аппаратной шиной I
#include <iarduino_I2C_Relay.h>                                     //   Подключаем библиотеку для работы с модулем
iarduino_I2C_Relay relay;                                           //   Объявляем объект для работы с функциями и методами би
                                                                    //   Если при объявлении объекта указать адрес, например,
void setup(){                                                       //
    Serial.begin(9600);                                             //
    if( relay.begin() ){                                            //   Инициируем работу с модулем.
        Serial.print("На шине I2C найден модуль с адресом 0x");     //
        Serial.print( relay.getAddress(), HEX );                    //   Выводим текущий адрес модуля.
        Serial.print(" который является реле\r\n"); //
        if( relay.changeAddress(newAddress) ){                      //   Меняем адрес модуля на newAddress.
            Serial.print("Адрес модуля изменён на 0x");             //
            Serial.println( relay.getAddress(), HEX );              //   Выводим текущий адрес модуля.
        }else{                                                      //
            Serial.println("Адрес модуля изменить не удалось!");    //
        }                                                           //
```

```
    }else{                                                          //
        Serial.println("модуль не найден!");          //
    }                                                               //
}                                                                   //
                                                                    //
void loop(){}
```

Примеры:

Специально для работы с модулями силовых ключей и реле, нами разработана библиотека iarduino_I2C_Relay которая позволяет реализовать все функции модуля.

Подробнее про установку библиотеки читайте в нашей инструкции.

Поочерёдное включение и выключение реле модуля:

```
#include <Wire.h>                                       // Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_Relay.h>                         // Подключаем библиотеку для работы с реле и силовыми ключами.
iarduino_I2C_Relay relay(0x09);                         // Объявляем объект relay для работы с функциями и методами библи
                                                        // Если объявить объект без указания адреса (iarduino_I2C_Relay r
void setup(){                                           //
    relay.begin();                                      // Инициируем работу с модулем.
    relay.digitalWrite(ALL_CHANNEL,LOW);                // Выключаем все каналы модуля.
}                                                       //
                                                        //
void loop(){                                            //
    relay.digitalWrite(1,HIGH); relay.digitalWrite(2,LOW);  // Включаем 1 канал и выключаем 2.
    delay(500);                                         // Ждём   500 мс.
    relay.digitalWrite(2,HIGH); relay.digitalWrite(1,LOW);  // Включаем 2 канал и выключаем 1.
    delay(500);                                         // Ждём   500 мс.
}                                                       //
```

Данный пример будет поочерёдно включать и выключать реле.

Чтение состояний реле модуля:

```
#include <Wire.h>                                          // Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_Relay.h>                            // Подключаем библиотеку для работы с реле и силовыми ключами.
iarduino_I2C_Relay relay(0x09);                            // Объявляем объект relay для работы с функциями и методами библи
                                                           // Если объявить объект без указания адреса (iarduino_I2C_Relay r
void setup(){                                              //
    Serial.begin(9600);                                    // Инициируем передачу данных в монитор последовательного порта н
    relay.begin();                                         // Инициируем работу с модулем.
//  Включаем и выключаем каналы модуля:                    //
    relay.digitalWrite(1,  LOW);                           // Отключаем 1 канал.
    relay.digitalWrite(2, HIGH);                           // Включаем  2 канал.
//  Проверяем состояние каналов модуля в цикле:            //
    for(int i=1; i<=2; i++){   Serial.print  ("Канал N ");  // Проходим по всем каналам модуля.
                               Serial.print  (i);          // Выводим номер очередного канала.
      if(relay.digitalRead(i)){Serial.println(" включен ");} // Если функция digitalRead() вернула HIGH значит канал включён.
      else                    {Serial.println(" отключен");} // Если функция digitalRead() вернула LOW  значит канал отключён.
    }                          Serial.println("---------");  //
}                                                          //
                                                           //
void loop(){}                                              //
                                                           //
//  ПРИМЕЧАНИЕ:                                            //
//  состояние всех реле модуля можно получить одним байтом:  //
//  uint8_t j = relay.digitalRead(ALL_CHANNEL);            // 2 младших бита переменной «j» соответствуют состояниям 2 канал
```

Данный пример считывает состояние реле и выводит их в монитор последовательного порта.

Защита от зависаний Arduino и отключения шины I2C:

```ino
#include <Wire.h>                                                    // Подключаем библиотеку для работы с аппаратной шиной
#include <iarduino_I2C_Relay.h>                                      // Подключаем библиотеку для работы с реле и силовыми к
iarduino_I2C_Relay relay(0x09);                                      // Объявляем объект relay для работы с функциями и мето
                                                                     // Если объявить объект без указания адреса (iarduino_I
void setup(){                                                        //
    Serial.begin(9600);                                              // Инициируем передачу данных в монитор последовательно
    relay.begin();                                                   // Инициируем работу с модулем.
    relay.enableWDT(5);                                              // Разрешаем работу сторожевого таймера модуля, задав в
}                                                                    // Отключить работу сторожевого таймера модуля можно фу
                                                                     //
void loop(){                                                         //
//  Переключаем 1 и 2 каналы модуля:                                 //
    relay.digitalWrite(1,HIGH); relay.digitalWrite(2,LOW); delay(500); // Включаем 1 канал, выключаем 2 канал и ждём 500 мс.
    relay.digitalWrite(2,HIGH); relay.digitalWrite(1,LOW); delay(500); // Включаем 2 канал, выключаем 1 канал и ждём 500 мс.
//  Сбрасываем (перезапускаем) сторожевой таймер модуля:             //
    relay.resetWDT();                                                // Теперь таймер модуля начнёт заново отсчитывать 5 сек
//  Сообщаем, что сработал сторожевой таймер:                        //
    if( relay.getStateWDT() == false ){ Serial.println("ERROR"); }   // Если таймер отключился, значит он досчитал до 0 и пе
}                                                                    // Если модуль не отвечает (отключилась шина I2C), то ф
```

При нормальной работе данного примера, в первую половину секунды включено первое реле, а во вторую половину секунды включено второе реле.

Если отключить от модуля вывод SDA или SCL шины I2C, то реле перестанут переключаться, но одно из реле останется включённым. Подождав от 4 до 5 секунд, сработает сторожевой таймер модуля и все реле отключатся. Время ожидания зависит от того, в каком месте выполнения кода был отключён вывод.

Примечание: Время назначенное сторожевому таймеру функцией `enableWDT()` должно быть больше чем время между вызовами функции `resetWDT()` .

Определение модуля на шине I2C и изменение его адреса:

```
uint8_t i = 0x09;                                       // Назначаемый модулю новый адрес (0x07 < адрес < 0x7F).
#include <Wire.h>                                       // Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_Relay.h>                         // Подключаем библиотеку для работы с реле и силовыми ключами.
iarduino_I2C_Relay relay;                               // Объявляем объект relay для работы с функциями и методами библи
                                                        // Если при объявлении объекта указать адрес, например, relay(0xE
void setup(){
    Serial.begin(9600);             Serial.print( "На шине I2C "                                    );
//  Инициируем работу с модулем:    (метод relay.begin(); вернёт true при успехе).
    if( relay.begin() ){            Serial.print( "найден модуль с адресом 0x"                     );
                                    Serial.print( relay.getAddress(), HEX                           );
                                    Serial.print( ", который является "                             );
//      Выводим информацию о модуле:
        switch( relay.getModel() ){
            case DEF_MODEL_2RM:     Serial.print( "электромеханическим реле на 2-канала"           );
            case DEF_MODEL_4RT:     Serial.print( "твердотельным реле на 4-канала"                 );
            case DEF_MODEL_4NC:     Serial.print( "силовым ключом на 4 N-канала с измерением тока" );
            case DEF_MODEL_4PC:     Serial.print( "силовым ключом на 4 P-канала с измерением тока" );
            case DEF_MODEL_4NP:     Serial.print( "силовым ключом на 4 N-канала до 10A"            );
            case DEF_MODEL_4PP:     Serial.print( "силовым ключом на 4 P-канала до 10A"            );
            default:                Serial.print( "неизвестным силовым ключом или реле"            );
        }                           Serial.print( ".\r\n"                                           );
//      Меняем адрес модуля на «i»: (метод relay.changeAddress(новый адрес); вернёт true при успехе).
        if(relay.changeAddress(i)){ Serial.print( "Адрес модуля изменён на 0x"                      );   // Меняем адрес модуля
                                    Serial.print( relay.getAddress(), HEX                           );   // Выводим текущий адр
                                    Serial.print( " и сохранён в flash память модуля."              );
        }else{                      Serial.print( "Адрес модуля изменить не удалось!"               ); } // Если метод relay.ch
    }else{                          Serial.print( "нет ни силовых ключей, ни реле!"                 ); } // Если метод relay.be
                                    Serial.print( "\r\n"                                            );
}
void loop(){}
```

Для работы этого примера необходимо что бы на шине I2C был только один модуль.

В первой строке скетча определяется переменная «i» с указанием адреса которой будет присвоен модулю (это значение Вы можете изменить на то, которое требуется Вам).

Данный пример определяет тип модуля, его текущий адрес и присваивает модулю новый адрес на шине I2C.

Описание функций библиотеки:

В данном разделе описаны функции библиотеки iarduino_I2C_Relay для работы с модулями реле.

Данная библиотека может использовать как аппаратную, так и программную реализацию шины I2C. О том как выбрать тип шины I2C рассказано в статье Wiki - расширенные возможности библиотек iarduino для шины I2C.

Подключение библиотеки:

- Если адрес модуля известен (в примере используется адрес 0x09):

```
#include <iarduino_I2C_Relay.h>   // Подключаем библиотеку для работы с модулями реле.
iarduino_I2C_Relay relay(0x09);   // Создаём объект relay для работы с функциями и методами библиотеки iarduino_I2C_Relay, ука
```

- Если адрес модуля неизвестен (адрес будет найден автоматически):

```
#include <iarduino_I2C_Relay.h>   // Подключаем библиотеку для работы с модулями реле.
iarduino_I2C_Relay relay;         // Создаём объект relay для работы с функциями и методами библиотеки iarduino_I2C_Relay.
```

При создании объекта без указания адреса, на шине должен находиться только один модуль.

Функция begin();

- Назначение: Инициализация работы с модулем.
- Синтаксис: begin();

- Параметры: Нет.
- Возвращаемое значение: bool - результат инициализации (true или false).
- Примечание: По результату инициализации можно определить наличие модуля на шине.
- Пример:

```
if( relay.begin() ){ Serial.print( "Модуль найден и инициирован!" ); }
else               { Serial.print( "Модуль не найден на шине I2C" ); }
```

Функция reset();

- Назначение: Перезагрузка модуля.
- Синтаксис: reset();
- Параметры: Нет.
- Возвращаемое значение: bool - результат перезагрузки (true или false).
- Пример:

```
if( relay.reset() ){ Serial.print( "Модуль    перезагружен" ); }
else               { Serial.print( "Модуль не перезагружен" ); }
```

Функция changeAddress();

- Назначение: Смена адреса модуля на шине I2C.
- Синтаксис: changeAddress(АДРЕС);
- Параметры:
  - uint8_t АДРЕС — новый адрес модуля на шине I2C (целое число от 0x08 до 0x7E)
- Возвращаемое значение: bool - результат смены адреса (true или false).
- Примечание: Текущий адрес модуля можно узнать функцией getAddress().
- Пример:

```
if( relay.changeAddress(0x12) ){ Serial.print( "Адрес модуля изменён на 0x12" ); }
```

```
else                            { Serial.print( "Не удалось изменить адрес"    ); }
```

Функция getAddress();

- Назначение: Запрос текущего адреса модуля на шине I2C.
- Синтаксис: getAddress();
- Параметры: Нет.
- Возвращаемое значение: uint8_t АДРЕС — текущий адрес модуля на шине I2C (от 0x08 до 0x7E)
- Примечание: Функция может понадобиться если адрес модуля не указан при создании объекта, а обнаружен библиотекой.
- Пример:

```
Serial.print( "Адрес модуля на шине I2C = 0x");
Serial.println( relay.getAddress(), HEX );
```

Функция getVersion();

- Назначение: Запрос версии прошивки модуля.
- Синтаксис: getVersion();
- Параметры: Нет
- Возвращаемое значение: uint8_t ВЕРСИЯ - номер версии прошивки от 0 до 255.
- Пример:

```
Serial.print( "Версия прошивки модуля ");
Serial.println( relay.getVersion(), HEX );
```

Функция getModel();

- Назначение: Запрос типа модуля.
- Синтаксис: getModel();
- Параметры: Нет.
- Возвращаемое значение: uint8_t МОДЕЛЬ — идентификатор модуля от 0 до 255.

- Примечание: По идентификатору можно определить тип модуля (см. пример).
- Пример:

```
switch( relay.getModel() ){
  case DEF_MODEL_2RM: Serial.print( "электромеханическое реле на 2-канала" );         break;
  case DEF_MODEL_4RT: Serial.print( "твердотельное реле на 4-канала" );               break;
  case DEF_MODEL_4NC: Serial.print( "силовой ключ на 4 N-канала с измерением тока" ); break;
  case DEF_MODEL_4PC: Serial.print( "силовой ключ на 4 P-канала с измерением тока" ); break;
  case DEF_MODEL_4NP: Serial.print( "силовой ключ на 4 N-канала до 10A" );            break;
  case DEF_MODEL_4PP: Serial.print( "силовой ключ на 4 P-канала до 10A" );            break;
  default:            Serial.print( "неизвестный модуль" );                          break;
}
```

Функция digitalWrite();

- Назначение: Включение/выключение одного или всех реле модуля.
- Синтаксис: digitalWrite(РЕЛЕ , СОСТОЯНИЕ);
- Параметры:
  - uint8_t РЕЛЕ — номер реле модуля 1, 2, или значение ALL_CHANNEL.
  - uint8_t СОСТОЯНИЕ — значение 1 (включено) или 0 (отключено).
- Возвращаемое значение: Нет.
- Примечание: Функция включает или отключает реле.
- Пример:

```
relay.digitalWrite( ALL_CHANNEL, 0 ); // Отключить все реле.
relay.digitalWrite( 2, 1 );           // Включить второе реле.
```

Функция digitalRead();

- Назначение: Чтение состояния одного из реле модуля.
- Синтаксис: digitalRead(РЕЛЕ);

- Параметры:
  - uint8_t РЕЛЕ — номер реле модуля, значение 1 или 2.
- Возвращаемое значение: uint8_t СОСТОЯНИЕ — значение 1 или 0.
- Примечание:
  - Функция возвращает состояние установленное функцией digitalWrite().
- Пример:

```
                            Serial.print( "Первое реле модуля ");
if( relay.digitalRead( 1 ) ){ Serial.println( "включено " ); }
else                        { Serial.println( "отключено "); }
```

Дополнительные функции библиотеки:

Данные функции не являются основными, но могут быть полезны.

Функция enableWDT();

- Назначение: Запуск (разрешение работы) сторожевого таймера модуля.
- Синтаксис: enableWDT(ВРЕМЯ);
- Параметры:
  - uint8_t ВРЕМЯ - количество секунд от 1 до 254.
- Возвращаемое значение: bool - результат запуска сторожевого таймера (true или false).
- Примечание:
  - После выполнения функции, сторожевой таймер начнёт отсчитывать время, от указанного до 0. Обнуление сторожевого таймера приведёт к перезагрузке модуля и, как следствие, отключению всех его каналов.
  - Если в процессе выполнения скетча, постоянно обращаться к функции enableWDT() (или resetWDT(), см. ниже), то таймер не сможет досчитать до 0, пока корректно работает скетч и шина I2C.
- Пример:

```
relay.enableWDT( 10 ); // Через 10 секунд модуль перезагрузится.
```

Функция disableWDT();

- Назначение: Отключение (запрет работы) сторожевого таймера модуля.
- Синтаксис: disableWDT();
- Параметры: Нет.
- Возвращаемое значение: bool - результат отключения сторожевого таймера (true или false).
- Примечание:
  - Обращение к функции отключает сторожевой таймер без перезагрузки модуля.
- Пример:

```
relay.disableWDT(); // Отключение сторожевого таймера.
```

Функция resetWDT();

- Назначение: Сброс (перезагрузка) сторожевого таймера.
- Синтаксис: resetWDT();
- Параметры: Нет
- Возвращаемое значение: bool - результат перезагрузки сторожевого таймера (true или false).
- Примечание:
  - Функция сбрасывает время сторожевого таймера в значение которое было определено ранее функцией enableWDT(), значит таймер начнёт отсчёт времени заново.
  - К функции resetWDT() можно обращаться только если сторожевой таймер уже запущен, в противном случае функция resetWDT() вернёт false.
- Пример:

```
relay.enableWDT( 10 ); // Через 10 секунд модуль перезагрузится;
delay(9000);           // Ждём 9 секунд.
relay.resetWDT();      // Модуль перезагрузится не через 1 секунду, а через 10.
```

Функция getStateWDT();

- Назначение: Чтение состояния сторожевого таймера.
- Синтаксис: getStateWDT();
- Параметры: Нет.
- Возвращаемое значение: bool - выполняется отсчёт времени (true или false).
- Примечание:
  - Если таймер запущен функцией enableWDT() и не отключался функцией disableWDT(), а функция getStateWDT() вернула false, значит таймер досчитал до 0, и перезагрузил модуль.
  - Если модуль не отвечает, например, отключилась шина I2C, то функция getStateWDT() так же вернёт false.
- Пример:

```
if( relay.getStateWDT() ){ Serial.println( "таймер выполняет отсчёт времени." ); }
else                     { Serial.println( "сторожевой таймер отключен."); }
```