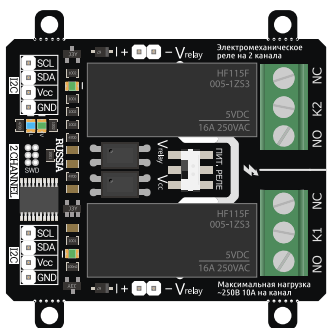


Модуль реле, 2 канала, FLASH-I2C, подключаем к Raspberry



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).

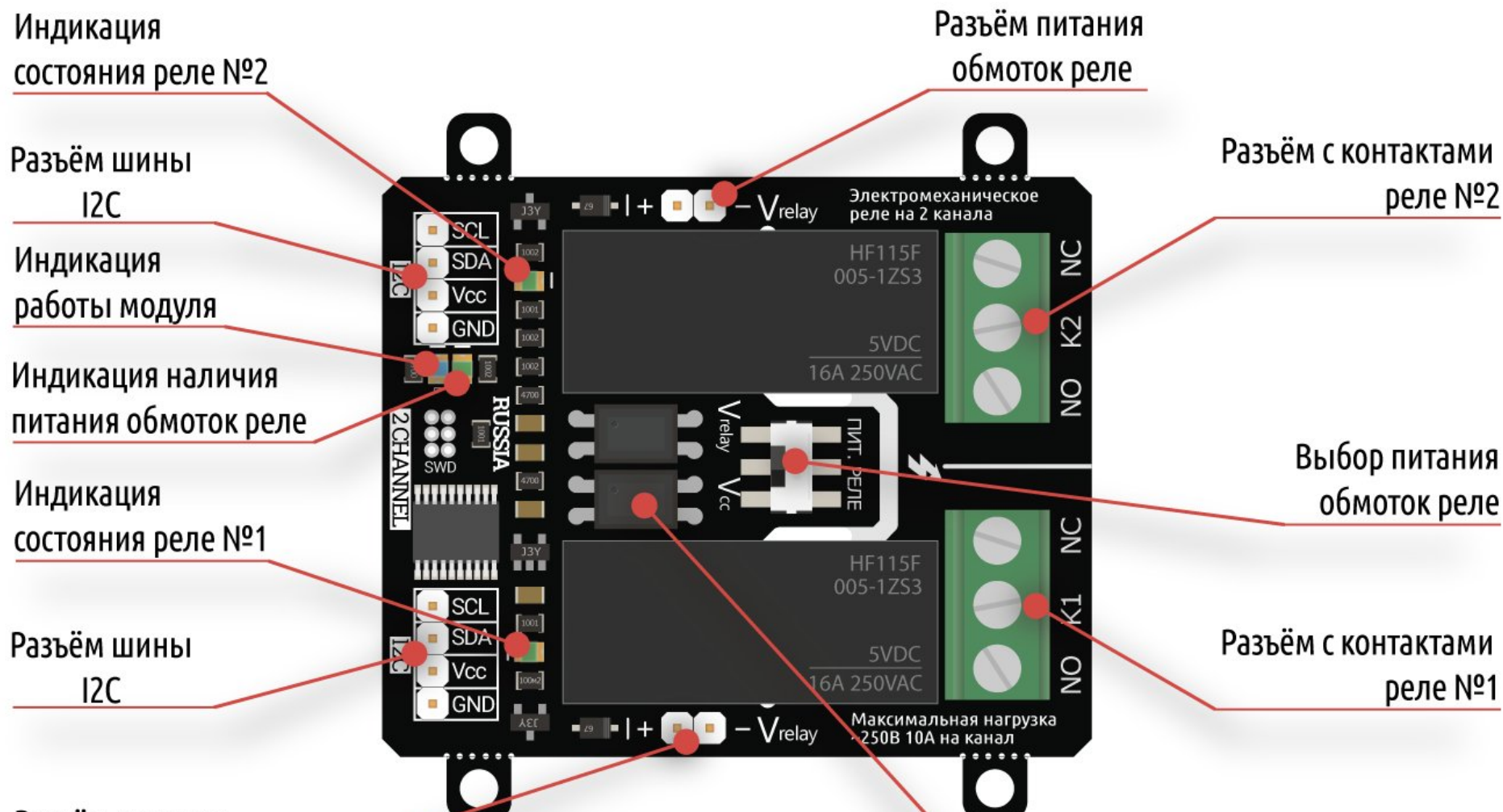
Управление модулем осуществляется по шине I2C. Модуль относится к линейке «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

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 °С.
- Габариты с креплением: 55 x 55 мм.
- Габариты без креплений: 55 x 45 мм.
- Вес: 40 г.



Разъем питания
обмоток реле

Оптоизоляция обмоток реле

Подключение:

По умолчанию все модули FLASH-I2C имеют установленный адрес 0x09.

- Перед подключением 1 модуля к шине I2C **настоятельно рекомендуется** изменить адрес модуля.
- При подключении 2 и более FLASH-I2C модулей к шине необходимо **в обязательном порядке предварительно изменить адрес каждого модуля**, после чего уже подключать их к шине.

Более подробно о том, как это сделать, а так же о многом другом, что касается работы FLASH-I2C модулей, вы можете прочесть в [этой 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 5V.
- **GND** - общий вывод питания.

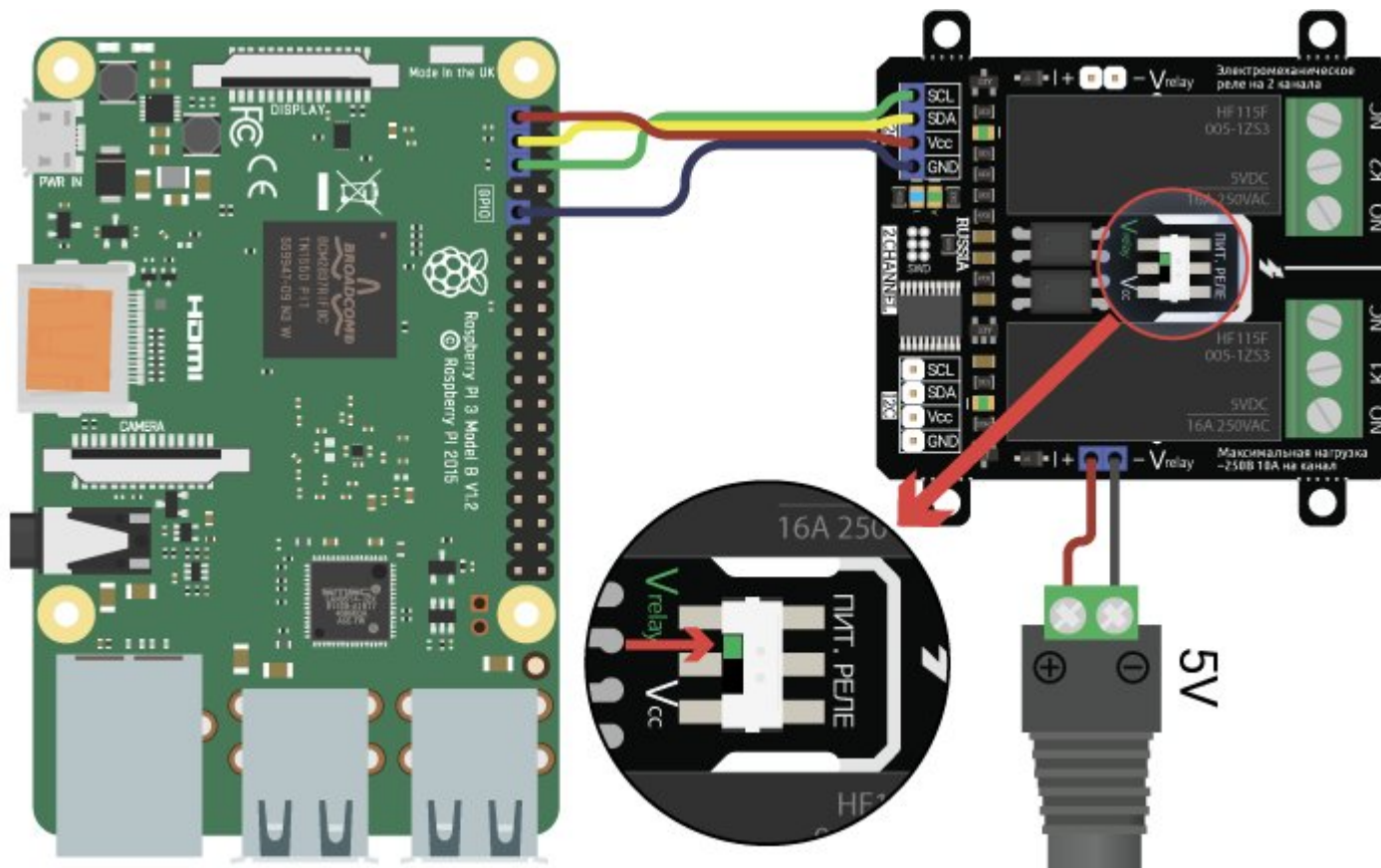
В правой части платы расположены два разъема: **K1** и **K2**, это выходы, через контакты которых подключаются силовые устройства к сети переменного тока до 250V. Устройства не должны потреблять более 10A (на каждый канал).

- **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 Raspberry Pi (питание реле в положении Vrelay)

Используя провода «[Мама – Мама](#)», подключаем напрямую к Raspberry Pi

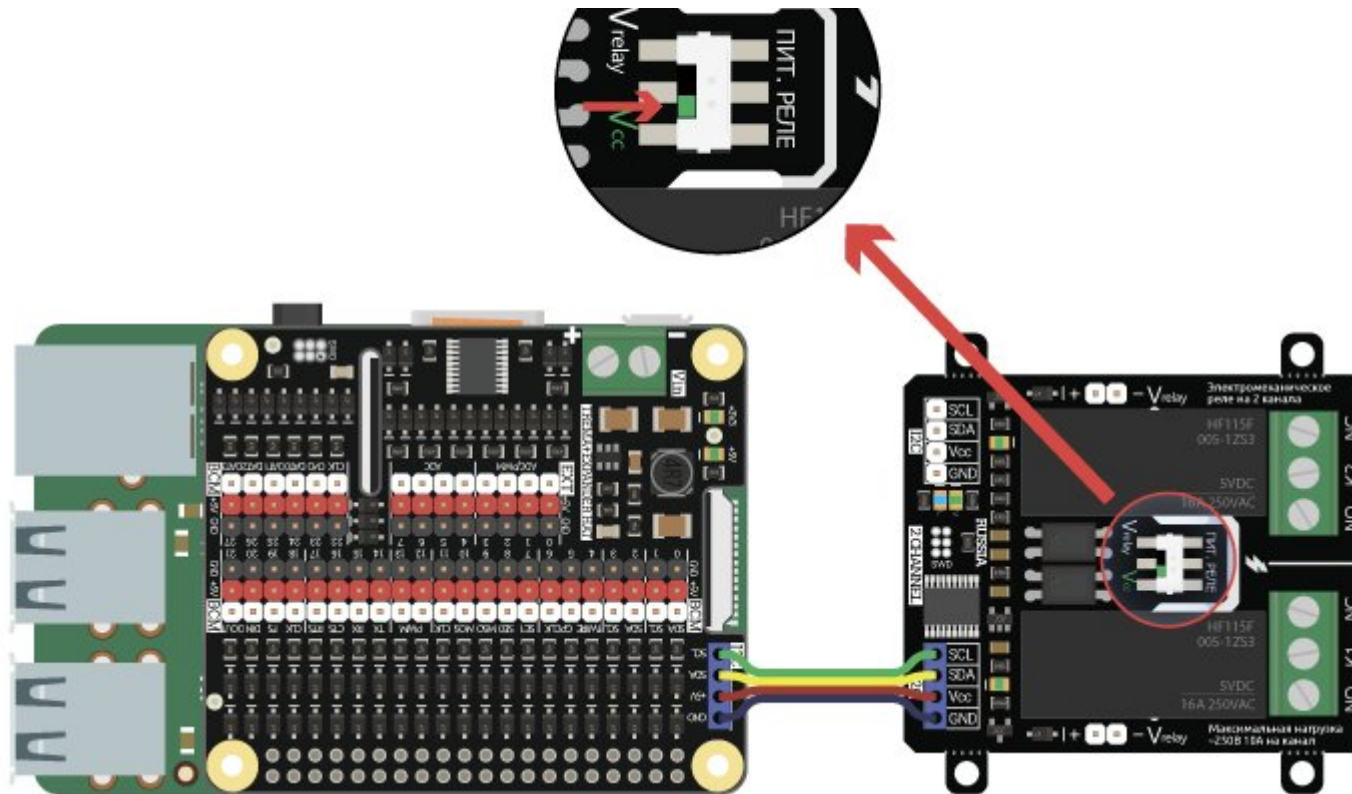
В этом случае необходимо питать логическую часть реле от 3,3 В Raspberry



Способ - 2: Используя шлейф, Trema+Expander Hat и Raspberry Pi (питание реле в положении Vcc)

Используя шлейф «[Мама – Мама](#)», подключаем к [Trema+Expander Hat](#)

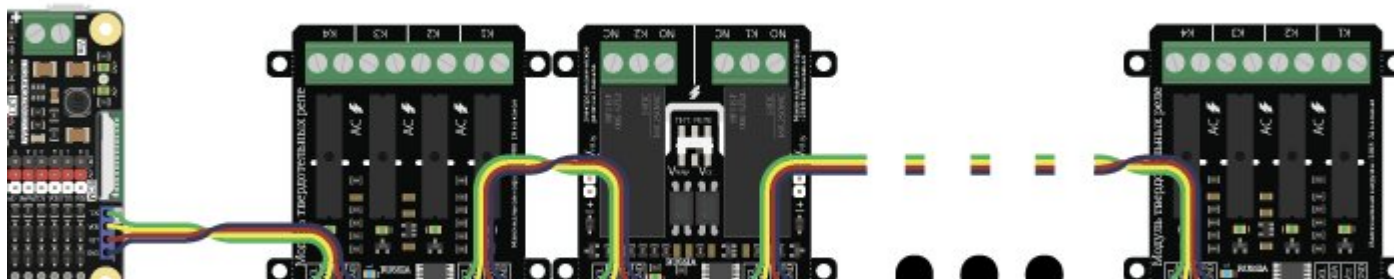




Подключение нескольких устройств

Благодаря шине I2C возможно подключение более 100 устройств, при таком подключении рекомендуем использовать дополнительный источник питания для устройств на шине:

Шина I2C:





Ведущий



Ведомый 1



Ведомый 2



Ведомый N

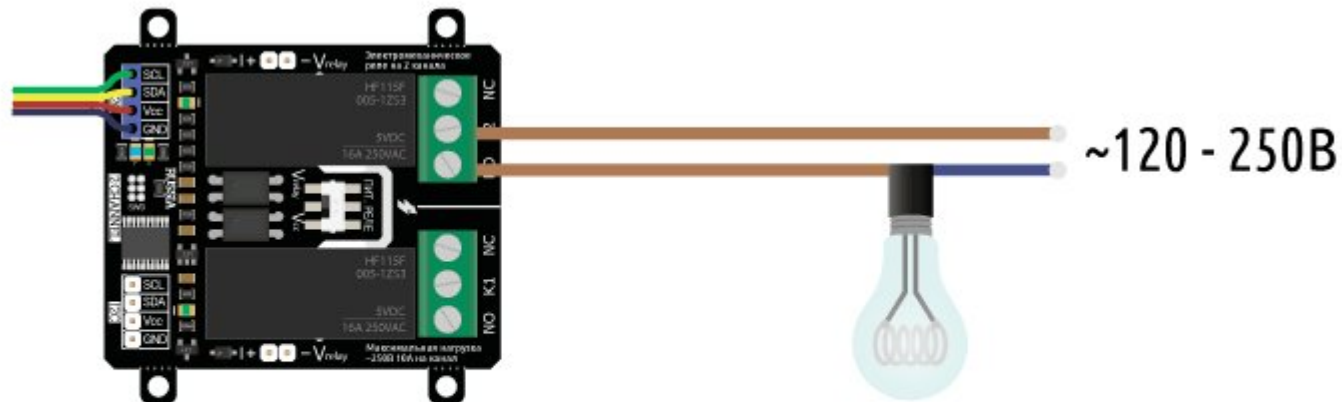
Подключение нагрузки

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.

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.

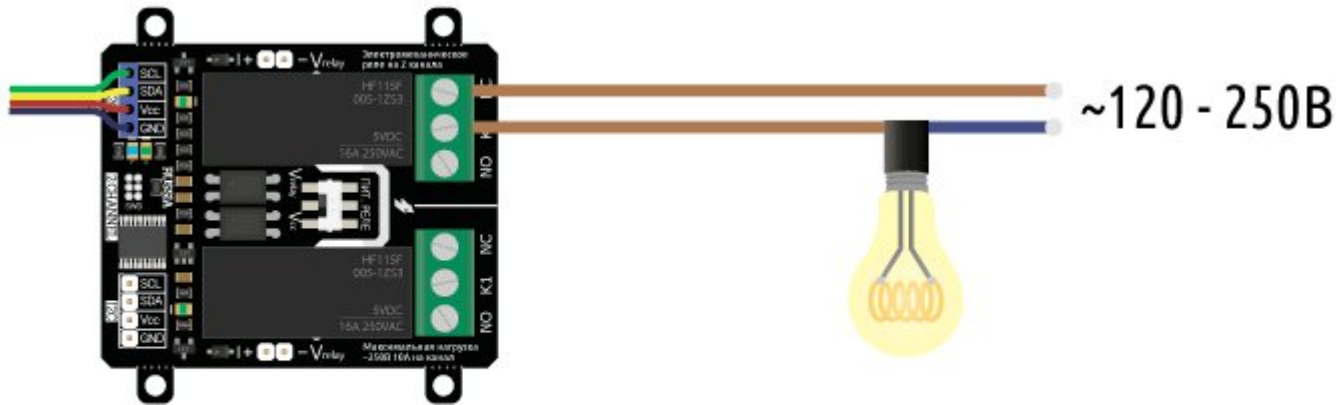
Логическая единица (реле включено) - цепь замкнута, логический ноль (реле выключено) - цепь разомкнута:



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:

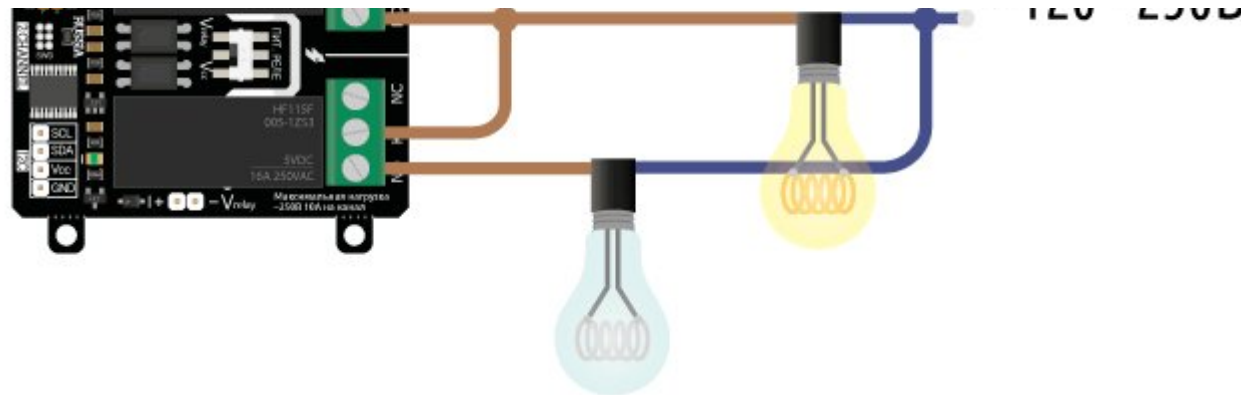


Вариант подключения:

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, 5V постоянного тока.

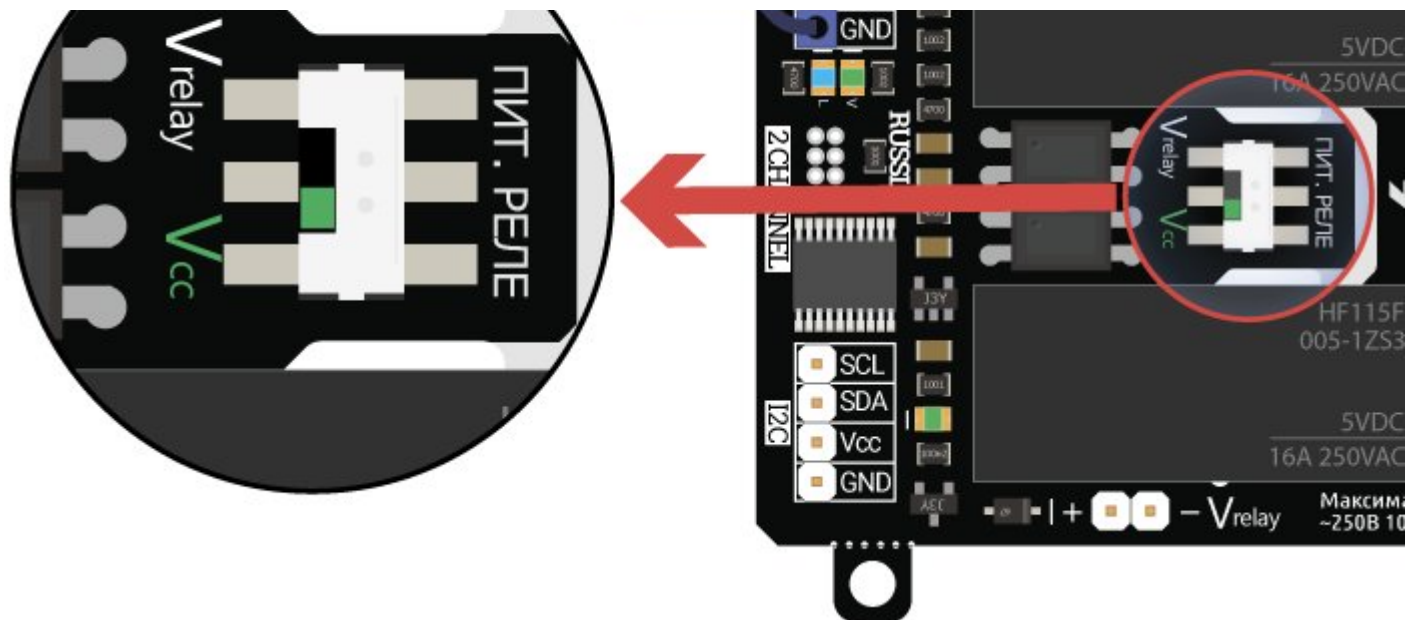
Питание обмоток реле

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 разъёма 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):





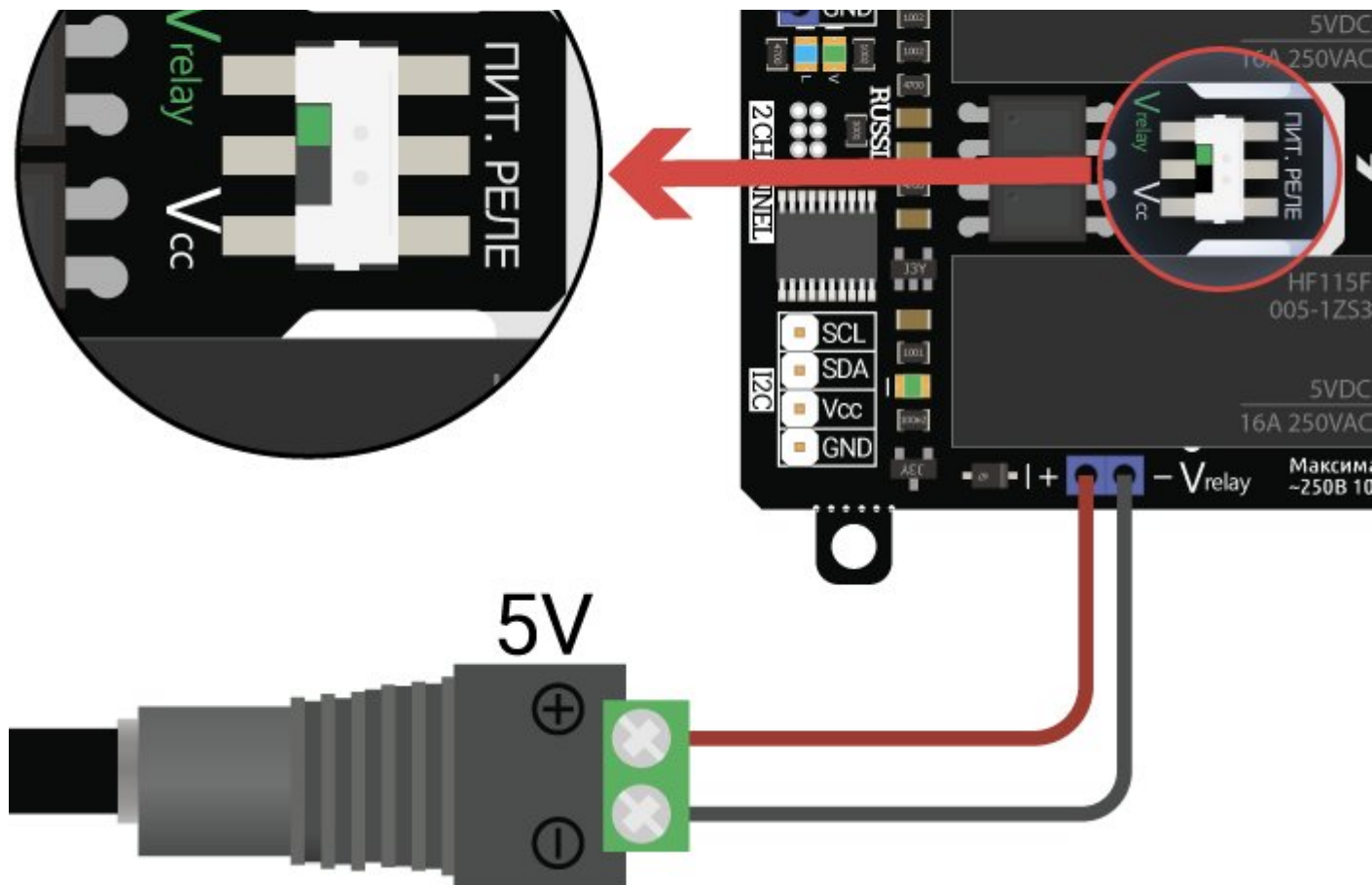
Плюсы такого подключения:

- 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.





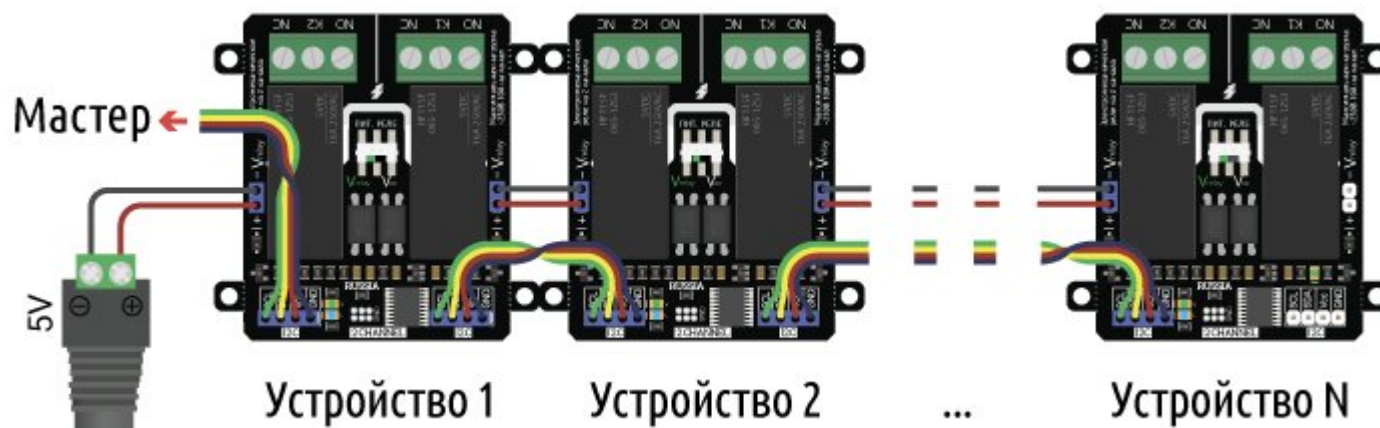
Плюсы такого подключения:

- 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:



Примеры:

Специально для работы с модулями силовых ключей и реле, нами разработана [библиотека pyiArduinoI2Crelay](#) которая позволяет реализовать все функции модуля.

Для работы с модулем необходимо включить шину I2C.

[Ссылка на подробное описание как это сделать.](#)

Внимание! Для корректной работы модулей FLASH-I2C на Raspberry Pi под управлением Raspberry OS "Buster" необходимо выключить динамическое тактирование ядра (опция **core_freq_min** должна быть равна **core_freq** в `/boot/config.txt`) [Ссылка на подробное описание.](#)

Для подключения библиотеки необходимо сначала её установить. Сделать это можно в менеджере модулей в Thonny Python IDE (тогда библиотека установится локально для этого IDE) или в терминале Raspberry (тогда библиотека будет системной) командой:

```
sudo pip3 install pyiArduinoI2Crelay
```

Подробнее об установке библиотек можно узнать в [этой статье](#).

Все примеры для Python3

Поочерёдное включение и выключение реле модуля:

```
from pyiArduinoI2Crelay import *
from time import sleep
relay = pyiArduinoI2Crelay(0x09)

relay.digitalWrite(ALL_CHANNEL, LOW)
while True:
    #Включаем и выключаем каналы модуля:
    relay.digitalWrite(1, HIGH)
    relay.digitalWrite(4, LOW)
    sleep(.5)
    relay.digitalWrite(2, HIGH)
    relay.digitalWrite(1, LOW)
    sleep(.5)
    # Подключаем библиотеку для работы с реле и силовыми
    # Подключаем метод sleep библиотеки time
    # Создаём объект relay для работы с функциями и метод
    # Если объявить объект без указания адреса (iarduino_
    # Выключаем все каналы модуля.
    # Входим в бесконечный цикл
    #
    # Включаем 1 канал
    # и выключаем 4.
    # Ждём 500 мс.
    # Включаем 2 канал
    # и выключаем 1.
    # Ждём 500 мс.
```

Данный пример будет поочерёдно включать и выключать реле.

Чтение состояний реле модуля:

```
from pyiArduinoI2Crelay import *
relay = pyiArduinoI2Crelay()

# Включаем и выключаем каналы модуля:
relay.digitalWrite(1, LOW)
# Подключаем библиотеку (модуль) для работы с реле
# Объявляем объект relay для работы с функциями и мет
# Если объявить объект без указания адреса (iarduino_
#
# Отключаем 1 канал.
```

```

relay.digitalWrite(2, HIGH) # Включаем 2 канал.
# Проверяем состояние каналов модуля в цикле:
#
for i in range(2):
    print("Канал № %s" % (i), end='')
    if relay.digitalRead(i):
        print(" включен ", end='')
    else:
        print(" отключен\t", end='')
print("-----")
# ПРИМЕЧАНИЕ: состояние всех каналов можно получить одним байтом:
# j = relay.digitalRead(ALL_CHANNEL);
# 2 младших бита переменной «j» соответствуют состояниям 2 каналов модуля.

```

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

Определение модуля на шине I2C и изменение его адреса:

```

from pyiArduinoI2Crelay import * # Подключаем библиотеку для работы с
i = 0x09 # Назначаемый модулю новый адрес (0x09)
#
choices = { # Структура для сравнения методом get
    DEF_MODEL_2RM: "электромеханическим реле на 2-канала", #
    DEF_MODEL_4RT: "твердотельным реле на 4-канала", #
    DEF_MODEL_4NC: "силовым ключом на 4 N-канала с измерением тока", #
    DEF_MODEL_4PC: "силовым ключом на 4 P-канала с измерением тока", #
    DEF_MODEL_4NP: "силовым ключом на 4 N-канала до 10A", #
    DEF_MODEL_4PP: "силовым ключом на 4 P-канала до 10A" #
} #
#
relay = pyiArduinoI2Crelay() # Объявляем объект relay для работы с
print("На шине I2C ", end='') #
if relay.begin(): # Иницируем работу с модулем реле ил

```



```

address = relay.getAddress() # Если при объявлении объекта указать
model = relay.getModel() #
model = choices.get(model, "неизвестным силовым ключом или реле") # Сравниваем модель модуля с кон
print("найден модуль с адресом %s, который является %s" % (address, model)) # Выводим текущий адрес модуля.
if relay.changeAddress(i): # Меняем адрес модуля на указанный в
    print("Адрес модуля изменён на %s" % (relay.getAddress())) # Выводим текущий адрес модуля.
else: # Если метод relay.changeAddress() ве
    print("Адрес модуля изменить не удалось!") #
else: # Если метод relay.begin() вернул fal
    print("нет ни силовых ключей, ни реле!") #

```

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

В первой строке скрипта 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 pyiArduinoI2Crelay](#) для работы с модулями реле.

Для подключения библиотеки необходимо сначала её установить. Сделать это можно в менеджере модулей в Thonny Python IDE (тогда библиотека установится локально для этого IDE) или в терминале Raspberry (тогда библиотека будет системной) командой:

```
pip3 install pyiArduinoI2Crelay
```

Подключение библиотеки:

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

```
from pyiArduinoI2Crelay import *    # Подключаем библиотеку для работы с реле
relay = pyiArduinoI2Crelay()        # Объявляем объект relay для работы с функциями и методами модуля Py_iarduino_I2C_Relay.
```

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

```
from pyiArduinoI2Crelay import *    # Подключаем библиотеку для работы с реле
relay = pyiArduinoI2Crelay()        # Объявляем объект relay для работы с функциями и методами модуля Py_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: результат инициализации (true или false).
- Примечание: Выполняется в конструкторе объекта при его объявлении.
- Пример:

```
if relay.begin():
    print("Модуль найден и инициирован!")
else:
    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():  
    print("Модуль перезагружен")  
else:  
    print("Модуль не перезагружен")
```

Функция changeAddress()

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

```
if relay.changeAddress(0x12)  
    print("Адрес модуля изменён на 0x12")  
else:  
    print("Не удалось изменить адрес")
```

Функция getAddress()

- Назначение: Запрос текущего адреса модуля на шине I2C.
- Синтаксис: getAddress();
- Параметры: Нет.
- Возвращаемое значение: 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:

```
print("Адрес модуля на шине I2C", end="")
print(hex(relay.getAddress()))
```

Функция getVersion()

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

```
print("Версия прошивки модуля")
print(hex(relay.getVersion()))
```

Функция getModel()

- Назначение: Запрос типа модуля.
- Синтаксис: getModel();
- Параметры: Нет.
- Возвращаемое значение: МОДЕЛЬ — идентификатор модуля от 0 до 255.
- Примечание: По идентификатору можно определить тип модуля (см. пример).
- Пример:

```
model = relay.getModel()           # Записываем модель
if model == DEF_MODEL_2RM:         # Сравниваем с константами
    model = "электромеханическим реле на 2-канала" # и записываем в ту же переменную
elif model == DEF_MODEL_4RT:      #
    model = "твердотельным реле на 4-канала"      #
elif model == DEF_MODEL_4NC:      #
    model = "силовым ключом на 4 N-канала с измерением тока" #
```

```

elif model == DEF_MODEL_4PC:                #
    model = "силовым ключом на 4 Р-канала с измерением тока" #
elif model == DEF_MODEL_4NP:                #
    model = "силовым ключом на 4 п-канала до 10а"           #
elif model == DEF_MODEL_4PP:                #
    model = "силовым ключом на 4 Р-канала до 10А"           #
else:                                       #
    model = "неизвестным силовым ключом или реле"           #

```

Функция digitalWrite()

- Назначение: Включение/выключение одного или всех реле модуля.
- Синтаксис: digitalWrite(РЕЛЕ , СОСТОЯНИЕ);
- Параметры:
 - РЕЛЕ — номер реле модуля, от 1 до 4, или значение ALL_CHANNEL.
 - 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:
 - РЕЛЕ — номер реле модуля, значение от 1 до 4.
- Возвращаемое значение: СОСТОЯНИЕ — значение 1 или 0.
- Примечание:

- Функция возвращает состояние установленное функцией digitalWrite().
- Пример:

```
print("Первое реле модуля", end="")
if relay.digitalRead(1):
    print("включено")
else:
    print("отключено")
```