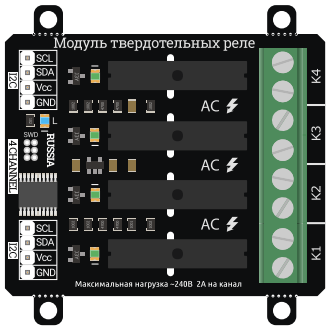
Модуль твердотельное реле, 4 канала, FLASH-I2C подключаем к Raspberry



Общие сведения:

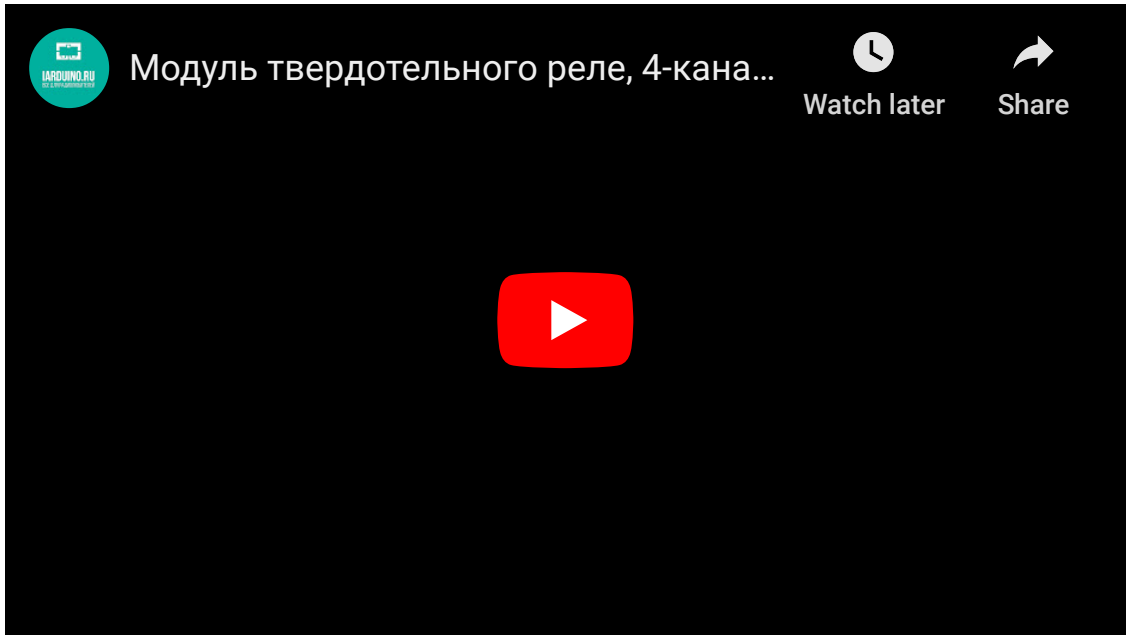
Модуль твердотельных реле на 4 канала, I2C, Flash - является устройством коммутации, которое позволяет подключать и отключать устройства к сети переменного тока от 120В до 240В. При этом устройства подключённые через выходные контакты модуля, не должны потреблять более 2А переменного тока (на каждый 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 можно использовать в любых проектах где требуется управлять устройствами с напряжением питания от 120В до 240В и потреблением переменного тока до 2А.

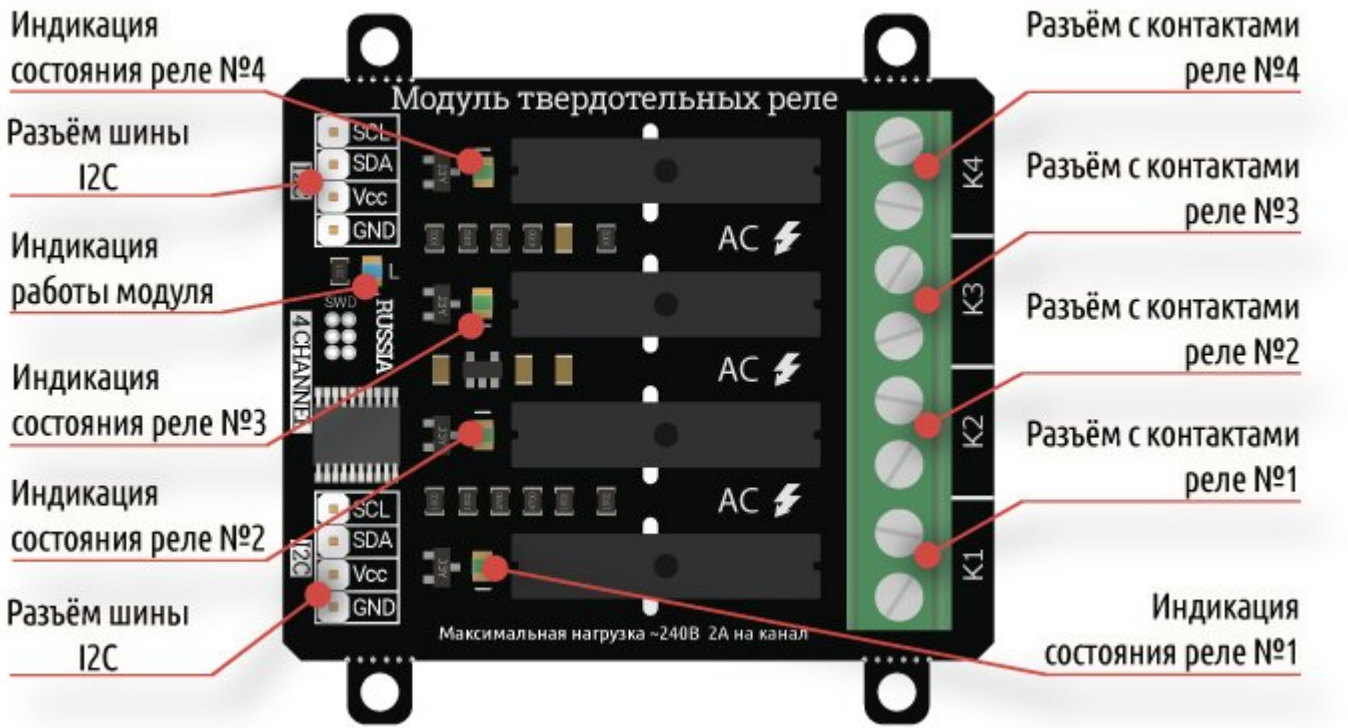
Видео:



Спецификация:

- Напряжение питания: 5 В (постоянного тока).
- Потребляемый ток: до 20 мА.
- Коммутируемое напряжение: от 120 В до 250 В (переменного тока).
- Коммутируемый ток: до 2 А (на каждый канал).
- Количество каналов: 4.
- 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).
- Рабочая температура: от -40 до +65 °C.
- Габариты с креплением: 55 х 55 мм.
- Габариты без креплений: 55 х 45 мм.
- Вес: 30 г.



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 твердотельных 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.

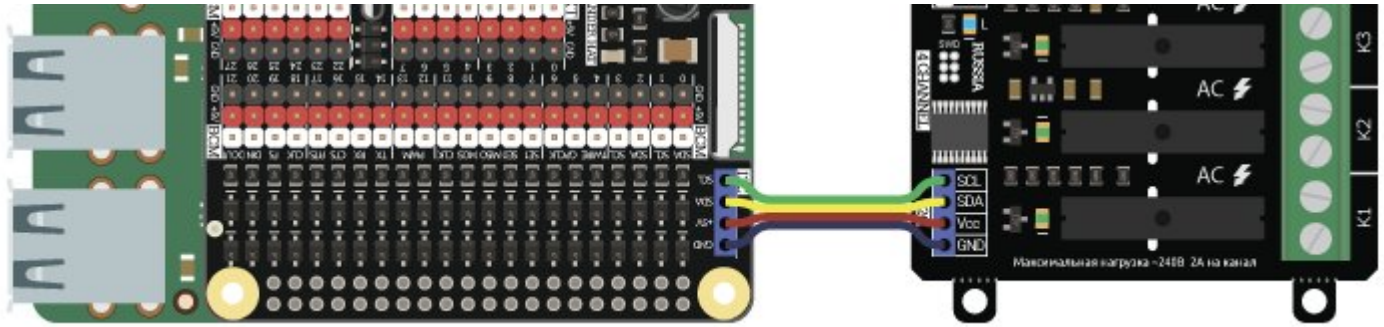
В правой части платы расположены четыре разъема: **K1**, **K2**, **K3**, **K4**, это выходы, через контакты которых подключаются силовые устройства к сети переменного тока от 120В до 250В. Устройства не должны потреблять более 2А (на каждый канал).

- **K1** - разъём первого реле с нормально разомкнутыми «NO» (Normally Open) контактами.
- **K2** - разъём второго реле с нормально разомкнутыми «NO» (Normally Open) контактами.
- **K3** - разъём третьего реле с нормально разомкнутыми «NO» (Normally Open) контактами.
- **K4** - разъём четвёртого реле с нормально разомкнутыми «NO» (Normally Open) контактами.

Для удобства подключения к Arduino воспользуйтесь Trema Shield, Trema Power Shield или Motor Shield.

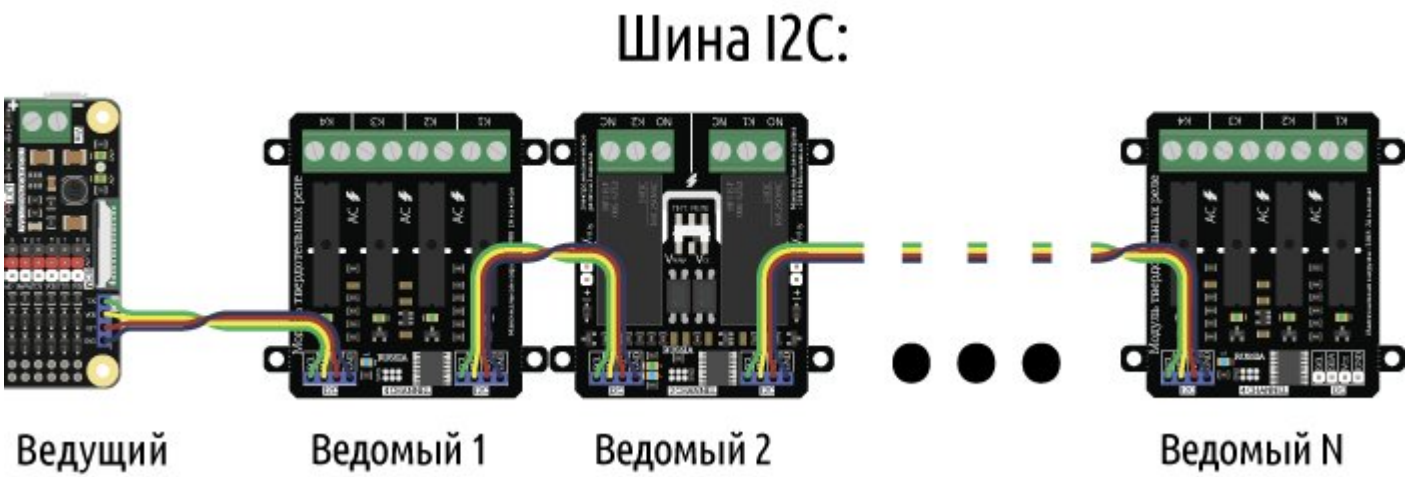
Способ подключения: Используя шлейф, Trema+Expander Hat и Raspberry Pi 3 model B

Используя 4-х проводной шлейф «Мама — Мама» подключаем к Trema+Exander Hat:



Подключение нескольких устройств

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



Питание:

Входное напряжение питания логической части модуля подаётся на выводы Vcc и GND любого разъёма шины I2C.

Подробнее о модуле:

Модуль твердотельных реле на 4 канала, I2C, Flash построен на базе микроконтроллера STM32F030F4, снабжен собственным

стабилизатором напряжения и четырьмя твердотельными реле.

Силовые устройства подключаются к сети переменного тока (от 120В до 240В) через контакты разъемов «K1», «K2», «K3», «K4».

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

- Если светодиод выключен, значит реле отключено и его выводы разъединены.
- Если светодиод светится, значит реле включено и его выводы замкнуты.

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.

```
#encoding=utf-8
```

```
                                                                                #
from Py_iarduino_I2C_Relay import *                                             #   Подключаем библиотеку для работы с реле и силовыми
from time import sleep                                                          #   Подключаем метод sleep библиотеки time
relay = Py_iarduino_I2C_Relay relay(0x09);                                      #   Создаём объект relay для работы с функциями и мето
                                                                                #   Если объявить объект без указания адреса (iarduino
relay.digitalWrite(ALL_CHANNEL,LOW);                                           #   Выключаем все каналы модуля.
while True:                                                                     #   Входим в бесконечный цикл
#Включаем и выключаем каналы модуля:                                           #
    relay.digitalWrite(1,HIGH)                                                  #   Включаем 1 канал
    relay.digitalWrite(4,LOW)                                                   #   и выключаем 4.
    sleep(.5)                                                                   #   Ждём 500 мс.
    relay.digitalWrite(2,HIGH)                                                  #   Включаем 2 канал
    relay.digitalWrite(1,LOW)                                                   #   и выключаем 1.
    sleep(.5)                                                                   #   Ждём 500 мс.
    relay.digitalWrite(3,HIGH)                                                  #   Включаем 3 канал
    relay.digitalWrite(2,LOW)                                                   #   и выключаем 2.
    sleep(.5)                                                                   #   Ждём 500 мс.
    relay.digitalWrite(4,HIGH)                                                  #   Включаем 4 канал
    relay.digitalWrite(3,LOW)                                                   #   и выключаем 3.
    sleep(.5)                                                                   #   Ждём 500 мс.
```

Данный пример будет поочерёдно включать и выключать реле.

Чтение состояний реле модуля:

```
#encoding=utf-8                                                                 # Устанавливаем кодировку скрипта
from pyiArduinoI2Crelay import *                                                # Подключаем библиотеку (модуль) для работы с реле
relay = pyiArduinoI2Crelay()                                                    # Объявляем объект relay для работы с функциями и мето
                                                                                # Если объявить объект без указания адреса (iarduino_I
#  Включаем и выключаем каналы модуля:                                          #
relay.digitalWrite(1, LOW);                                                     # Отключаем 1 канал.
relay.digitalWrite(2,HIGH);                                                     # Включаем  2 канал.
```

```
relay.digitalWrite(3, LOW);                                    # Отключаем 3 канал.
relay.digitalWrite(4,HIGH);                                    # Включаем  4 канал.
#  Проверяем состояние каналов модуля в цикле:                 #
for i in range(4):                                             # Проходим по всем каналам модуля.
        print("Канал № %s" % (i), end='')                      # Выводим номер очередного канала.
        if relay.digitalRead(i):                               # Если функция digitalRead() вернула True
                print(" включен ", end='')                     # значит канал включён.
        else:                                                  # Если функция digitalRead() вернула False
                print(" отключен\t", end='')                   # значит канал отключён.
print("--------")
#  ПРИМЕЧАНИЕ: состояние всех каналов можно получить одним байтом:
#  j = relay.digitalRead(ALL_CHANNEL);
#  4 младших бита переменной «j» соответствуют состояниям 4 каналов модуля.
```

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

Определение модуля на шине I2C и изменение его адреса:

```
#encoding=utf-8                                                #  Устанавливаем кодировку скрипла
from pyiArduinoI2Crelay import *                               #  Подключаем библиотеку для работы с
i = 0x09                                                       #  Назначаемый модулю новый адрес (0x0
                                                               #
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():                                                              #  Инициируем 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
```

- 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.
```

Подключение библиотеки:

- 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 P-канала с измерением тока"   #
elif model == DEF_MODEL_4NP:                                   #
  model == "силовым ключом на 4 n-канала до 10a"               #
elif model == DEF_MODEL_4PP:                                   #
  model == "силовым ключом на 4 P-канала до 10A"               #
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")
```