WB-MR11 Modbus Relay 11 Channel Module

WB-MR11 модуль реле 11-канальный

Руководство по эксплуатации

Самая актуальная документация всегда доступна на нашем сайте по ссылке: https://wirenboard.com/wiki/WB-MR11_Modbus_Relay_11_Channel_Module

Этот документ составлен автоматически из основной страницы документации
и ссылок первого уровня.

Содержание

WB-MR11 Modbus Relay 11 Channel Module

Модель снята с производства.

Contents

Релейный модуль WB-MR11

Назначение

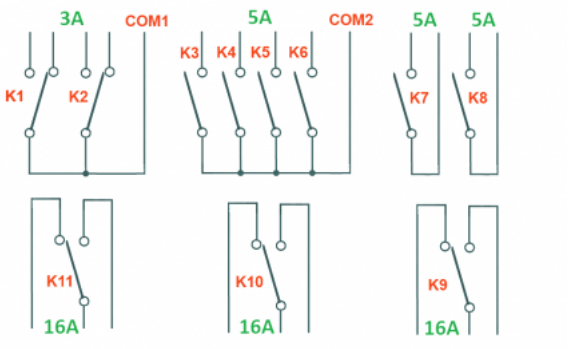
11-канальный модуль реле предназначен для систем промышленной и домашней автоматизации. Часть реле объединены в группы, что упрощает проводку при использовании модуля для автоматизации освещения или управления внешними контакторами.

Общий принцип работы

Выходы

В модуле установлены 8 реле HF32F (Datasheet (http://www.hongfa.com/pro/pdf/HF32F-G_en.pdf)), коммутирующие токи 3 и 5 А, а также 3 реле Omron G2RL-1-E (Datasheet (https://www.omron.com/ecb/products/pdf/en-g2rl.pdf)) для коммутации 16-амперных каналов. Выходы выведены на винтовые зажимы, рассчитанные на максимальный ток 20 А.

Контакты переключающих реле K1 и K2 (3 А) объединены в группу с общим проводом COM1. Контакты реле (5 А) с нормально открытыми контактами K3 — K6 объединены в группу с общим проводом COM2. Отдельно выведены контакты нормально открытых реле (5 А) K7 и K8 и переключающих реле (16 А) K9 — K11.



Контакты реле

Реле	Активная нагрузка (230 В, переменный ток)	Реактивная нагрузка (230 В, переменный ток)
K1 — K2	700 Вт	120 Вт
K3 — K6	1500 Вт	200 Вт
K7 — K8	1500 Вт	200 Вт
K9 — K11	4000 Вт	700 Вт

Максимальное постоянное напряжение, которое могут коммутировать все реле модуля — 30 В.

Для коммутации больших токов используйте внешние контакторы, рассчитанные на соответствующую нагрузку. В случае больших пусковых токов вы также можете использовать модули реле WB-MR3 и WB-MR6 соответствующих модификаций.

Каждый контакт реле защищен от образования дуги при замыкании/размыкании симметричным TVS-диодом.

В модуле может быть включен таймер безопасного режима, который при прекращении обмена данными по Modbus с модулем отключает все реле по прошествии заданного времени.

Входы

На безвинтовые зажимы модуля выведены 11 входов типа "сухой контакт", не изолированные от источника питания. Для устранения дребезга контактов на каждой входной линии применяются RC-цепочки. Входы подтянуты резисторами к линии питания +3.3V. Кнопки или выключатели с фиксацией подключаются между выводами GND и соответствующим входом Inputs. Входы могут использоваться как входы общего назначения, для счета сигналов и для прямого управления каналами реле. Программно можно выбрать три режима работы входов: управление кнопкой без фиксации или по Modbus; управление выключателем (с фиксацией) или по Modbus; управление только по Modbus. По умолчанию включен режим управления кнопкой без фиксации.

Технические характеристики

<table>
<tr><th>Параметр</th><th>Значение</th></tr>
<tr><td colspan="2">Питание</td></tr>
<tr><td>Напряжение питания</td><td>9 — 24 В постоянного тока</td></tr>
<tr><td>Потребляемая мощность</td><td>

В режиме холостого хода (со всеми выключенными реле) — 0,5 Вт
Со всеми включенными реле — 2,5 Вт
Пиковое значение — до 8 Вт в течение 20 мс

</td></tr>
<tr><td colspan="2">Выходы</td></tr>
<tr><td>Количество выходов</td><td>11 (16 с учетом переключающих реле)</td></tr>
<tr><td>Тип выходов</td><td>Контакты механического реле</td></tr>
<tr><td>Конфигурация контактов</td><td>Двухпозиционные, нормально открытые.
Состояния: разомкнут <—> замкнут Kx и COM</td></tr>
<tr><td>Конфигурация выходов</td><td>Две группы по 7 выходов, общий провод в каждой группе</td></tr>
<tr><td>Максимальное коммутируемое напряжение, AC</td><td>250 В</td></tr>
<tr><td>Максимальное коммутируемое напряжение, DC</td><td>30 В</td></tr>
<tr><td>Максимальный коммутируемый ток на каждый канал</td><td>

Для резистивной нагрузки: от 3 А до 16 А
Для ёмкостной и индуктивной нагрузки: от 120 мА до 400 мА

(в зависимости от выбранных реле)
<table>
<tr><th>Контакты</th><th>Активная нагрузка</th><th>Реактивная нагрузка</th></tr>
<tr><td>K1 — K2</td><td>700 Вт</td><td>120 Вт</td></tr>
<tr><td>K3 — K6</td><td>1500 Вт</td><td>200 Вт</td></tr>
<tr><td>K7 — K8</td><td>1500 Вт</td><td>200 Вт</td></tr>
<tr><td>K9 — K11</td><td>4000 Вт</td><td>700 Вт</td></tr>
</table>
</td></tr>
<tr><td>Сопротивление контактов</td><td>< 100 миллиом</td></tr>
<tr><td>Напряжение изоляции между контроллером и выходом</td><td>1500 В (среднеквадратичное значение)</td></tr>
<tr><td>Срок жизни: количество переключений для нагрузки
3 А — 10 А/230 В переменного тока (резистивная нагрузка)</td><td>100 000</td></tr>
<tr><td colspan="2">Входы</td></tr>
<tr><td>Количество входов (Inputs 1—11)</td><td>11</td></tr>
<tr><td>Тип входов</td><td>"Сухой контакт", не изолированные от источника питания</td></tr>
<tr><td>Функции</td><td>

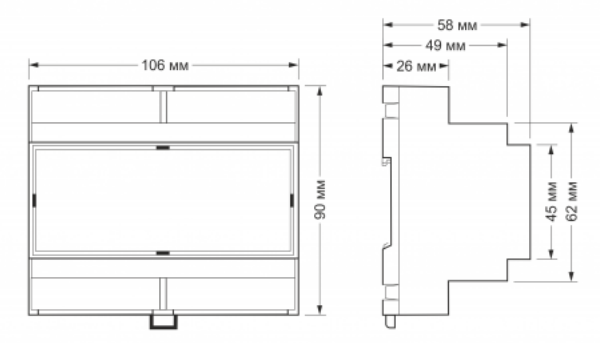
Входы общего назначения
Счет сигналов
Прямое управление каналами реле

</td></tr>
<tr><td colspan="2">Управление</td></tr>
<tr><td>Интерфейс управления</td><td>RS-485</td></tr>
<tr><td>Изоляция интерфейса</td><td>Неизолированный</td></tr>
<tr><td>Протокол обмена данными</td><td>Modbus RTU, адрес задается программно, заводские настройки указаны на наклейке</td></tr>
<tr><td>Параметры интерфейса RS-485</td><td>

Скорость: 9600 бит/сек
Данные: 8 бит
Проверка чётности: нет
Стоповых бит: 2

</td></tr>
<tr><td>Готовность к работе после подачи питания</td><td>~0,03 с</td></tr>
<tr><td colspan="2">Габариты</td></tr>
<tr><td>Ширина, DIN-юнитов</td><td>6</td></tr>
<tr><td>Габаритные размеры (Д x Ш x В)</td><td>106,25 x 90,2 x 57,5 мм</td></tr>
<tr><td colspan="2">Индикация</td></tr>
<tr><td>Индикация питания и обмена данными</td><td>Зеленый светодиод, расположенный на плате между клеммой V+ и блоком винтовых контактов</td></tr>
<tr><td>Индикация состояния каналов реле</td><td>Зеленые светодиоды 1 — 11, расположенные рядом с винтовыми зажимами контактов соответствующих реле</td></tr>
<tr><td colspan="2">Условия эксплуатации</td></tr>
<tr><td>Температура воздуха</td><td>От -40 до +50 °C</td></tr>
<tr><td>Относительная влажность</td><td>До 92%, без конденсации влаги</td></tr>
</table>

Габаритные размеры модуля



Габаритные размеры

Обмен данными

На физическом уровне модуль подключается через интерфейс RS-485. Для управления WB-MR11 используется протокол Modbus RTU. В устройствах Wirenboard данные Modbus передаются по линиям связи RS-485. Подробнее смотрите страницу Протокол Modbus. Modbus-адрес модуля задается на заводе и расположен на его боковой стороне. Адрес может быть изменен программно. Подробно смотрите в разделе #Параметры_Modbus

Монтаж

Релейный модуль монтируется на стандартную DIN-рейку шириной 35 мм и занимает ширину 6 DIN-модулей.

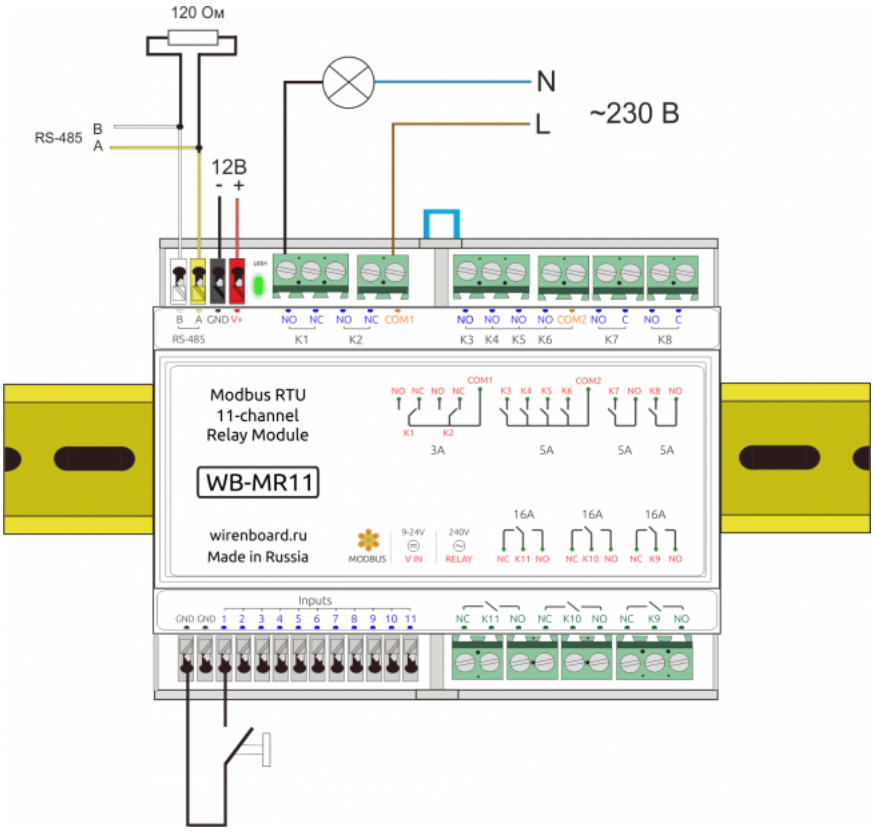
Блоки безвинтовых зажимов на плате реле служат для подключения линий питания, управления (RS-485) и входных контактов. Все выводы GND на безвинтовых зажимах (в том числе и "минус" питания) объединены на плате. При использовании при монтаже наконечников типа НШВИ необходимо, чтобы диаметр изолированных манжет не превышал 3,6 мм, сечение провода — 0,75 мм^2, а длина проводящей втулки — 5—6 мм. Винтовые зажимы принимают провод сечением 2,5—4 мм^2. Сечение провода должно соответствовать коммутируемой нагрузке.

При подключении коммутируемых устройств важно помнить, что хотя каждый из винтовых зажимов выдерживает соответствующий ток не более 20 А.

Кнопки или выключатели с фиксацией подключаются между выводами GND и соответствующим входом Inputs. Ток, протекающий при замыкании кнопки или выключателя, невелик, так что тип кнопки или выключателя может быть любым. Несмотря на встроенную защиту от дребезга, выбирайте качественные кнопки и выключатели, чтобы исключить ложные срабатывания из-за плохих контактов.

Если устройство — последнее на линии RS-485, то между его входами A и B необходимо установить резистор-терминатор сопротивлением 120 Ом. Практика показывает, что в случае стендовых испытаний при небольшой длине линии RS-485 и небольшом количестве устройств терминатор на последнем устройстве в линии можно не устанавливать.

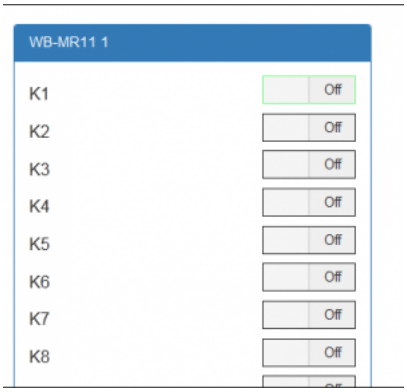
Модуль необходимо устанавливать таким образом, чтобы удовлетворять требованиям электробезопасности и не допускать случайного касания контактов, находящихся под высоким напряжением. Модуль должен эксплуатироваться при рекомендованных условиях окружающей среды.



Образец монтажа и подключения модуля (у последнего модуля на линии устанавливается терминирующий резистор). Выбор качественного блока питания очень важен для работы модуля

Добавление модуля в веб-интерфейс контроллера Wiren Board

Новое устройство добавляется в веб-интерфейс, в раздел соответствующего порта RS-485. В веб-интерфейсе на панели справа выбираем пункт Configs -> Serial Device Driver Configuration. В настройках порта /dev/ttyAPP1 добавляем новое устройство, нажав на кнопку **+ Serial device** в разделе List of Devices. Затем указываем адрес устройства и выбираем его тип. Подробно о настройке устройств в веб-интерфейсе Wiren Board смотрите страницу RS-485:Настройка_через_веб-интерфейс.



Новый модуль WB-MR11 в веб-интерфейсе

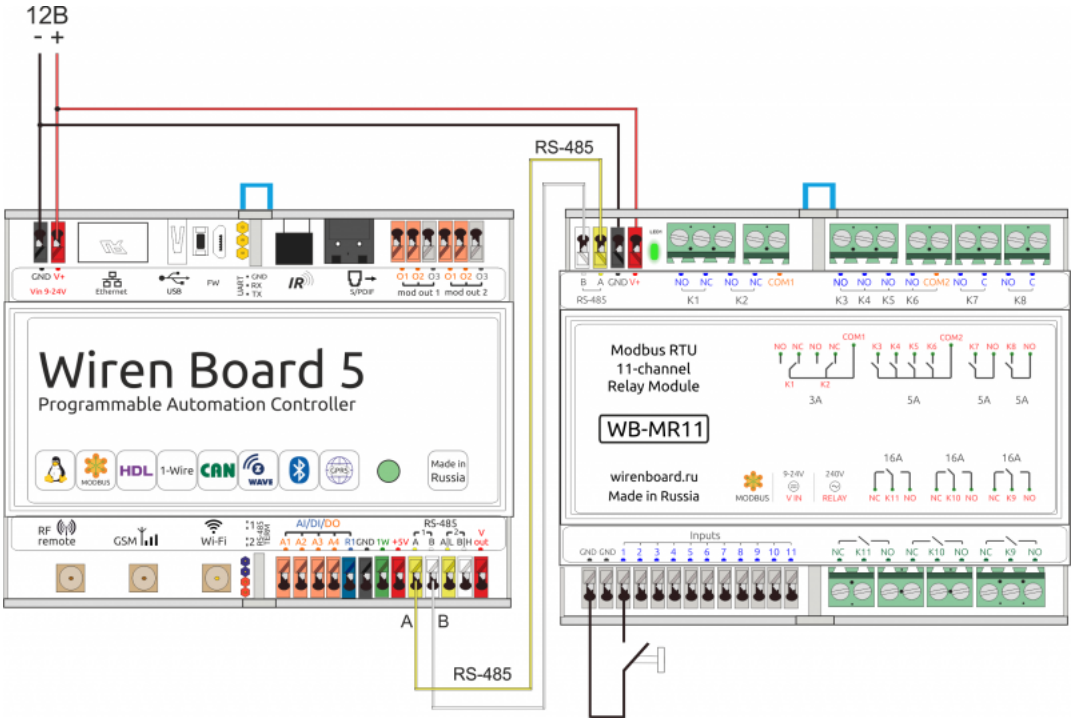
В разделе Devices появилось новое устройство, название составлено из типа устройства и адреса: **WB-MR11 1**

Обратите внимание: желтый индикатор Status на WB_MR11 начал периодически мигать, это означает, что Wiren Board обменивается данными с модулем реле. В веб-интерфейсе можно следить за параметром Supply voltage (напряжение питания модуля реле) — он меняется почти при каждом опросе модуля.

Устройство WB-MR11 стандартное, поэтому его описание задано в шаблоне, который хранится на контроллере Wiren Board в файле /usr/share/wb-mqtt-serial/templates/config-wb-mr11.json.

Управление модулем через веб-интерфейс Wiren Board

Пример управления релейного модуля WB-MR11 контроллером Wiren Board 5 и программирования сценариев управления мы рассмотрим на простом макете, когда релейный модуль является единственным устройством, подключенным к контроллеру. Нормально разомкнутая кнопка без фиксации подключена ко входу 1 и GND. Питание 12 В подается на входы V+(+) и GND (-). Входы/выходы RS-485 A и B первого порта контроллера и модуля WB-MR11 соединены.

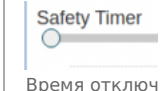


Лабораторный макет

Кнопки управления каналами реле

Флажки состояния входов

Счетчик входов



Время отключ
безопасности

С помощью виртуальных выключателей в веб-интерфейсе K1-K11 можно управлять соответствующими выходами модуля и следить за их состоянием. Если реле будет включено или выключено через внешний вход, это отразится и в веб-интерфейсе. Текущее состояние входов показывают флажки Input1 — Input11 (на них нельзя щелкнуть, чтобы изменить состояние). Счетчики нажатий/включений отображаются в полях Input 1 counter – Input 11 counter. Значения счетчиков хранятся в оперативной памяти микроконтроллера модуля реле и обнуляются при выключении/включении питания и сбросе.

Ползунок Safety Timer позволяет задать время отключения (в секундах) всех выходов при отсутствии обмена данными с Wiren Board (0 – таймер безопасности отключен). При возобновлении связи с контроллером выключившиеся реле останутся выключенными. Ползунком удобно управлять с помощью клавиш со стрелками, предварительно сфокусировавшись на нем мышью: перемещение ползунка с помощью мыши не обеспечивает достаточную точность. Таймер безопасности позволяет остановить технологические процессы в случае, если контроль над модулем утрачен (например, перебит кабель RS-485).

Управление по Modbus

Подробно о работе с модулем написано в разделе Управление модулями реле Wirenboard по протоколу Modbus. Там же можно найти карту регистров устройства.

Примеры программирования

Пишем простой скрипт приоритизации нагрузки

В качестве примера напишем скрипт, который автоматически будет отключать сильноточное реле K9 устройства wb-mr11_79 (считая, что оно управляет нагрузкой с низким приоритетом), если включены сильноточные реле K10 и K11. Такой сценарий актуален в случае ограничения выделенной мощности (например, электроплита и водонагреватель являются приоритетными нагрузками, а теплый пол можно отключить на время их совместной работы). Мы также хотим запоминать состояние низкоприоритетного реле K9 и включать его, если оно было включено до этого, как только выделенной мощности достаточно. Если мы пытаемся включить реле при уже пороговой мощности, реле не включится.

MQTT-названия устройств и их элементов управления можно найти в разделе Settings веб-интерфейса (см. MQTT).

Скрипт простой, так как не учитывает реального потребления мощности, а ориентируется только на состояние контактов реле.

```
//Создаем виртуальный выключатель, "запоминающий" текущее состояние реле K9
//Если реле не удалось включить из-за высокой нагрузки, то оно включится, как только нагрузка понизится
defineVirtualDevice("K9VirtualState", {
```

```
    title: "K9VirtualState",
    cells: {
        enabled: {
            type: "switch",
            value: false
        },
    }
});

//Создаем второй виртуальный выключатель, состояние которого хранит приоритет
//(false -- можно включать низкоприоритетное устройство, true -- нельзя)
defineVirtualDevice("K9Priority", {
    title: "K9Priority",
    cells: {
        enabled: {
            type: "switch",
            value: false
        },
    }
});

//Теперь отрабатываем переключения реле 9 и 10 и 11

//Состояние реле 9

defineRule("checkK9", {
  whenChanged: "wb-mr11_79/K9",
  then: function (newValue, devName, cellName)  {
    dev["K9VirtualState"]["enabled"] = newValue;

 //Если включены устройства с высоким приоритетом, то не даем реле 9 включиться
    if (dev["K9Priority/enabled"] == true) {dev["wb-mr11_79"]["K9"] = false; log(dev["K9Priority/enabled"])}

      }
});


//Обратите внимание: правило может следить сразу за состояниями нескольких реле одновременно (логическое "или")
defineRule("checkK1011", {
   whenChanged:
   [
    "wb-mr11_79/K10",
    "wb-mr11_79/K11",
    ],
   then: function (newValue, devName, cellName)  {
        if ((dev["wb-mr11_79/K10"] == true) && (dev["wb-mr11_79/K11"] == true))
        {log (true); dev["K9Priority"]["enabled"] = true; dev["wb-mr11_79"]["K9"] = false} else
        {log (false); dev["K9Priority"]["enabled"] = false;
        if (dev["K9VirtualState/enabled"] == true) {dev["wb-mr11_79"]["K9"] = true} else {dev["wb-mr11_79"]["K9"] = false}
        }
   }
});
```

Назад к списку периферийных устройств

RS-485

- English
- русский

Описание

RS-485 — стандарт коммуникации по двухпроводной шине.

Теоретически на шину можно подключать до 256 устройств. Длина линии может быть до 1200 метров, но она сильно влияет на скорость передачи данных.

Энциклопедия АСУ ТП. Интерфейс RS-485 — подробно про работу интерфейса.

В устройствах Wiren Board используется Протокол Modbus поверх RS-485. Пожалуйста, ознакомьтесь с ним для лучшего понимания работы устройств.

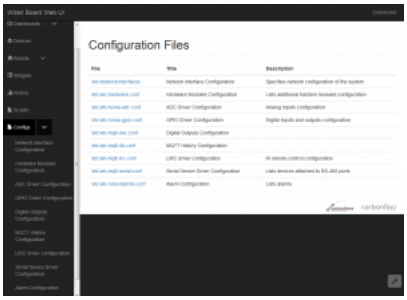
Максимальная скорость передачи данных в периферийных устройствах Wiren Board — до 115 200 бит/с.

Как правильно проложить шину

В статье RS-485:Физическое подключение описано как правильно проложить шину.

Добавление устройства в веб-интерфейс

RS-485:Настройка через веб-интерфейс — что сделать для появления устройства в веб-интерфейсе контроллера.



Настройка происходит через страницу *Configs* веб-интерфейса

Работа с портом RS-485 контроллера из собственного ПО

- Стандартно в Wiren Board с подключёнными по RS-485 устройствами работает Драйвер wb-mqtt-serial (ранее *wb-homa-modbus*). Он позволяет работать с подключёнными устройствами RS-485 через систему MQTT-сообщений.
- Если вы хотите работать с портом RS-485 напрямую, не используя этот драйвер — отключите его, иначе он будет писать в порт RS-485.
- Работа с последовательным портом из Linux
- Доступ к порту RS-485 контроллера Wiren Board с компьютера
- Настройка параметров обмена данными по RS-485 для modbus-устройств Wiren Board

Протокол Modbus

- English
- русский

Contents

Основные понятия

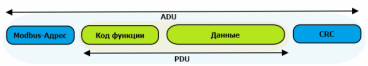
Modbus - это протокол прикладного (седьмого) уровня модели OSI. Чаще всего он служит для обмена данными между устройствами автоматизации и реализован в виде "протокола ответов на запросы (request-reply protocol)".

В устройствах Wiren Board данные Modbus передаются по последовательным линиям связи RS-485. В последовательных линиях связи протокол RS-485 полудуплексный и работает по принципу «клиент-сервер». Каждое устройство в сети (кроме ведущего см. далее) имеет адрес от 1 до 247, адрес 0 используется для широковещательной передачи данных всем устройствам, а адреса 248–255 считаются зарезервированными согласно спецификации Modbus, их использование не рекомендуется.

Существует две спецификации протокола: Modbus RTU и Modbus ASCII. В Modbus RTU передается 11-битный символ, состоящий из 1 стартового бита, 8 бит данных (начиная с младшего бита), бит четности (необязателен) и 2 стоповых бита - если бит четности не передается, или 1 стоповый бит - если бит четности передается. Такой символ передает 1 байт данных. В устройствах Wiren Board по умолчанию бит контроля четности не передается и используется 2 стоповых бита. В Modbus ASCII каждый байт передается двумя символами, представляющими ASCII-коды младшей и старшей четырехбитной группы байта (пример). Modbus RTU передает больше информации при той же скорости последовательной линии, и в устройствах Wiren Board используется именно он. Все дальнейшее описание относится к Modbus RTU.

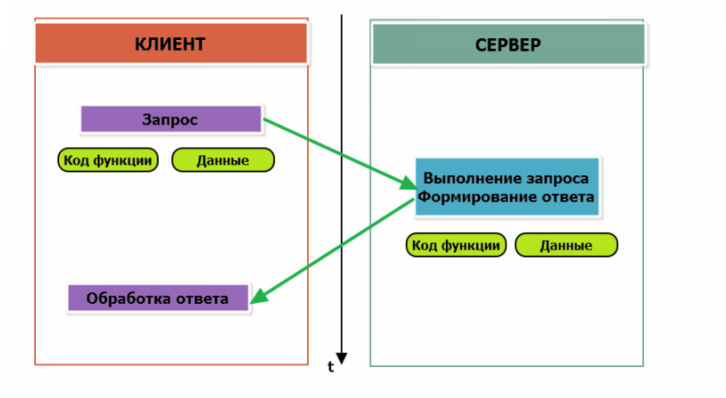
Ведущее устройство ("мастер", или "клиент") периодически опрашивает "ведомое", или "сервер". Ведущее устройство не имеет адреса, передача сообщений от устройства-сервера ведущему без запроса ведущего в протоколе не предусмотрена.

Пакет данных Modbus выглядит, как это показано на рисунке. **PDU** (Protocol Data Unit) — общая часть пакета MODBUS, включающая код функции и данные пакета. **ADU** (Application Data Unit) — полный пакет MODBUS. Включает в себя специфичную для физического уровня часть пакета и PDU. Для последовательных линий в заголовке ADU передается адрес устройства, а в конце — контрольная сумма CRC16. Максимальный размер ADU в последовательных коммуникационных линиях составляет **253 байта** (из максимальных, разрешенных спецификацией 256 байт вычитается 1 байт адреса и два байта контрольной суммы). Для справки — в Modbus TCP максимальная длина пакета составляет 260 байт.

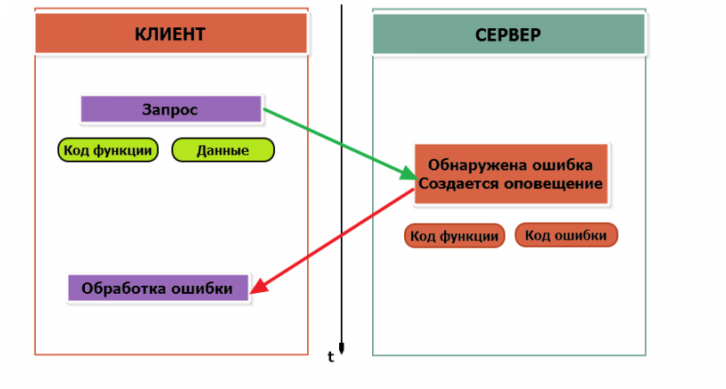


Датаграмма Modbus в общем виде

Функция кодируется одним байтом и определяет, какое действие должно выполнить устройство-сервер. Значение кодов функций лежат в диапазоне от 1 до 255, причем коды от 128 до 255 зарезервированы для сообщений об ошибках со стороны устройства-сервера. Код 0 не используется. Размер блока данных может варьироваться от нуля до максимально допустимого. Если обработка запроса прошла без ошибок, то устройство-сервер возвращает пакет ADU, содержащий запрошенные данные.



Modbus-транзакция, прошедшая без ошибок



Modbus-транзакция с ошибками

При возникновении ошибки устройством возвращается код ошибки. При обычной транзакции код функции в ответе возвращается без изменений; при ошибке старший бит кода функции устанавливается в единицу (то есть *код функции* + 0x80). Так же есть таймаут ожидания ответа от ведомого устройства — бессмысленно долго ждать ответ, который, возможно, никогда и не придет.

Структуры данных Modbus

В Modbus принято кодировать адреса и данные в формате big-endian, то есть в формате, когда байты следуют, начиная со старшего: например, при передаче шестнадцатеричного числа 0x1234 сначала устройством будет принят байт 0x12, а затем — 0x34. Для передачи данных другого типа, например, чисел с плавающей запятой (float), текстовых строк, даты и времени суток и т.п. производитель может выбрать свой собственный способ кодирования — для расшифровки получаемых данных важно ознакомится со спецификацией производителя устройства.

Модель данных Modbus

Обмен данными с Modbus-устройствами происходит через регистры. В протоколе Modbus определяется четыре типа регистров, показанных в таблице:

Таблица	Размер	Доступ
Регистры флагов (Coils)	1 бит	чтение и запись
Дискретные входы (Discrete Inputs)	1 бит	только чтение
Регистры хранения (Holding Registers)	16-битное слово	чтение и запись
Регистры ввода (Input Registers)	16-битное слово	только чтение

Регистры флагов (Coils) хранят однобитные значения - то есть могут находится в состоянии 0 или 1. Такие регистры могут обозначать текущее состояние выхода (включено реле). Название "coil" буквально и означает обмотку-актюатор электромеханического реле. Регистры флагов допускают как чтение, так и запись.

Дискретные входы (Discrete Inputs) также являются однобитными регистрами, описывающими состояние входа устройства (например, подано напряжение — 1). Эти регистры поддерживают только чтение.

Регистры хранения (Holding Registers) и **регистры ввода** (Input Registers) представлены двухбайтовым словом и могут хранить значения от 0 до 65535 (0x0000 — 0xFFFF). Регистры ввода допускают только чтение (например, текущее значение температуры). Регистры хранения поддерживают как чтение, так и запись (для хранения настроек). В настоящее время во многих устройствах, в частности в устройствах Wiren Board, эти регистры не разделяются. Команды на чтение регистра хранения N и регистра ввода N обратятся к одному и тому же значению в адресном пространстве устройства.

Адреса регистров

Регистры в стандарте Modbus адресуются с помощью 16-битных адресов. Адресация начинается с нуля. Адрес регистра, таким образом, может принимать значения от 0 до 65535.

Адресные пространства регистров, также называемые таблицами иди блоками, могут быть различны для всех четырёх типов регистров. Это значит, что значения регистров с одинаковым адресом, но разным типом, в общем случае разные.

Например, при чтении регистра флагов (coil) номер 42, регистра дискретного входа (Discrete), регистров ввода и хранения (Input и Holding) с теми же адресами, можно получить четыре разных значения.

Нестандартная адресация

В документации на некоторые, особенно старые, устройства адреса элементов (регистров) указываются в формате, не соответствующем стандарту. В этом формате тип элемента кодируется первой цифрой адреса, а адресация начинается не с нуля.

Например, регистр хранения с адресом 0 может записываться как 40001 или 400001, а Coil с адресом 0 как 000001.

В таблице перевода адресов в стандартный формат показаны диапазоны для двух разных нестандартных типов указания адресов и соответствующие им типы данных и диапазоны стандартных адресов.

Тип данных	Стандартные адреса	Стандартные адреса (hex)	Нестандартные адреса (5 цифр)	Нестандартные адреса (6 цифр)
Флагов (Coils)	0-65535	0x0000 - 0xFFFF	00001 - 09999	000001 - 065536
Дискретных входов (Discrete)	0-65535	0x0000 - 0xFFFF	10001 - 19999	100001 - 165536
Регистры входов (Input Registers)	0-65535	0x0000 - 0xFFFF	30001 - 39999	300001 - 365536
Регистры хранения (Holding Registers)	0-65535	0x0000 - 0xFFFF	40001 - 49999	400001 - 465536

Признаки использования нестандартной адресации:

- Адреса записываются в десятичном формате
- Во всех адресах пять или шесть цифр
- Адреса с недискретными данными (показания датчиков и т.п.) начинаются на 30 или 40

Часто рядом с нестандартными адресами указываются и адреса соответствующие стандарту, обычно в шестнадцатеричном формате. Стоит отметить, что физически в пакете данных передаются адреса в стандартном формате, независимо от способа представления их в документации.

Пример описания регистров в документации

В готовых шаблонах устройств для контроллера Wiren Board есть шаблон для однофазного счетчика электроэнергии SDM220 (/usr/share/wb-mqtt-serial/templates/config-sdm220.json). В документации от производителя "Eastron SDM 220 Modbus Smart Meter Modbus Protocol Implementation V1.0" перечислены регистры и соответствующие им измеряемые параметры, например:

Address (Register)	Description	Units	Modbus Protocol Start Address Hex (Hi Byte Lo Byte)
30001	Line to neutral volts.	Volts	00 00
30007	Current.	Amps.	00 06
30013	Active power	Whatts	00 0C
30019	Apparent power	VoltAmps	00 12
...	...	...	... ...

Производитель в таблице приводит и логические, и физические адреса регистров, что позволяет нам с легкостью создать шаблон устройства и проиллюстрировать связь между логическими и физическими адресами Modbus-регистров.

```
"channels" : [
        {
                "name" : "Voltage",
                "type" : "voltage",
                "reg_type" : "input",
                "address" : "0x00",
                "format" : "float"
        },
        {
                "name" : "Current",
                "type" : "current",
                "reg_type" : "input",
                "address" : "0x06",
                "format" : "float"
        },
        {
                "name" : "Active Power",
                "type" : "power",
                "reg_type" : "input",
                "address" : "0x0c",
                "format": "float"
        },
        {
                "name" : "Apparent Power",
                "type" : "power",
                "reg_type" : "input",
                "address" : "0x12",
                "format": "float"
        },
```

Фрагмент шаблона счетчика SDM220

Коды функций чтения и записи регистров

В следующей таблице приведены наиболее распространенные коды функций Modbus:

Код функции	HEX	Название	Действие
1	0x01	Read Coils	Чтение значений нескольких регистров флагов
2	0x02	Read Discrete Inputs	Чтение значений нескольких дискретных входов
3	0x03	Read Holding Registers	Чтение значений нескольких регистров хранения
4	0x04	Read Input Registers	Чтение значений нескольких регистров ввода
5	0x05	Write Single Coil	Запись одного регистра флагов
6	0x06	Write Single Register	Запись одного регистра хранения
15	0x0F	Write Multiple Coils	Запись нескольких регистров флагов
16	0x10	Write Multiple Register	Запись нескольких регистров хранения

Команды условно можно разделить по типам: чтение значений — запись значений; операция с одним значением — операция с несколькими значениями.

Формат данных запросов и ответов Modbus

Рассмотрим подробнее, как происходит обмен данными между устройством-клиентом, отправляющим запрос, и устройством-сервером, отвечающим ему. На следующем рисунке показан обмен данными контроллера с устройством с адресом 0x01. Мы хотим прочесть 8 coil-регистров, начиная с первого.

Обмен данными в Modbus

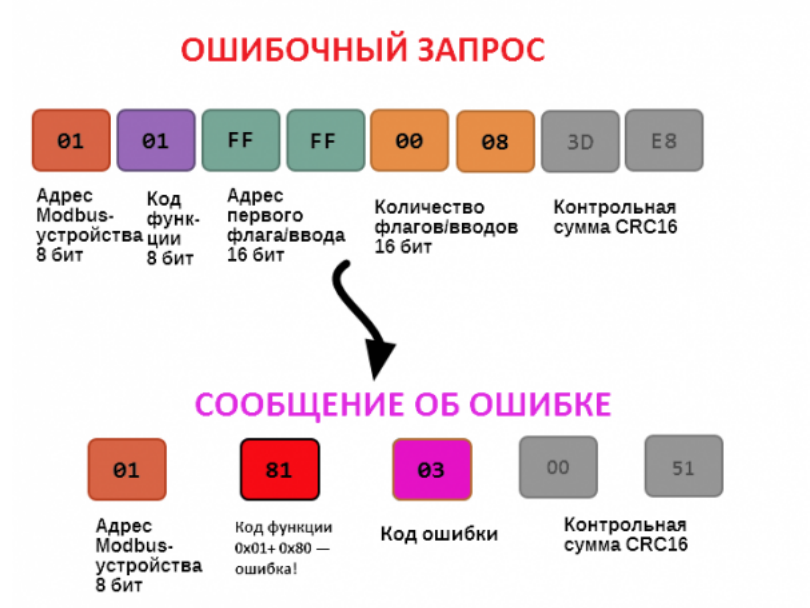
В качестве данных мы получили шестнадцатеричное число 0x2D, то есть состояние восьми coil-регистров в двоичном виде такое: 0b10110100.

В следующей таблице приведены структуры данных запросов и ответов для основных функций Modbus.

Код функции	Запрос	Ответ
1 (Read Coils) и 2 (Read Discrete Inputs)	▪ Адрес первого регистра флагов или входного регистра (16 бит) ▪ Количество данных (8 значений на байт) (16 бит)	▪ Число передаваемых байт (8 бит) ▪ Значения регистров флагов или входных регистров (8 значений на байт)
3 (Read Holding Registers) и 4 (Read Input Registers)	▪ Адрес первого регистра (16 бит) ▪ Количество регистров, которые нужно прочесть	▪ Число передаваемых байт (8 бит) ▪ Значения регистров (16 бит на 1 регистр)
5 (Write Single Coil)	▪ Адрес регистра (16 бит) ▪ Значение, которое нужно записать (0 — выключить, 0xFF00 — включить)	Ответ аналогичен запросу
6 (WriteSingle Register)	▪ Адрес регистра(16 бит) ▪ Новое значение регистра (16 бит)	Ответ аналогичен запросу
15 (WriteMultipleCoils)	▪ Адрес первого регистра флагов для записи (16 бит) ▪ Количество регистров флагов для записи (16 бит) ▪ Количество передаваемых байт данных для регистров флагов (8 бит) ▪ Данные (8 регистров флагов на байт)	▪ Адрес первого coil-регистра (16 бит) ▪ Количество записанных coil-регистров(16 бит)
16 (Write Multiple register)	▪ Адрес первого регистра хранения для записи (16 бит) ▪ Количество регистров хранения для записи (16 бит) ▪ Количество передаваемых байт данных для регистров (8 бит) ▪ Данные (16 байт на регистр)	▪ Адрес первого регистра хранения (16 бит) ▪ Количество записанных регистров хранения(16 бит)

Коды исключений (ошибки) Modbus

Если запрос не может по той или иной причине быть обработан устройством-сервером, то в ответ он отправляет сообщение об ошибке. Сообщение об ошибке содержит адрес Modbus-устройства, код функции, при выполнении которой произошла ошибка, увеличенный на 0x80, код ошибки и контрольную сумму:



Транзакция завершилась с ошибкой

В этом случае мы попытались обратиться к несуществующему адресу регистра 0xFFFF и попытались прочесть 8 регистров флагов. В результате мы получили код ошибки 0x03 — "В поле данных передано неверное значение".

Наиболее распространенные коды ошибок Modbus приведены в следующей таблице:

Код ошибки	Название ошибки	Что означает
1	Illegal Function	В запросе был передан недопустимый код функции
2	Illegal Data Address	Указанный в запросе адрес не существует
3	Illegal Data Value	Неверный формат запроса, например количество байт в запросе не соответствует ожидаемому. **Примечание**: несмотря на название, эта ошибка не говорит о том, что само значение регистра неправильное или ошибочное, и должна использоваться только для ошибок формата запроса.
4	Server Device Failure	Произошла невосстановимая ошибка на устройстве при выполнении запрошенной операции
5	Acknowledge	Запрос принят, выполняется, но выполнение потребует много времени; необходимо увеличить таймаут.
6	Server Device Busy	Устройство занято обработкой предыдущего запроса.
7	Negative Acknowledge	Устройство не может выполнить запрос, необходимо получить от устройства дополнительную диагностическую информацию. Возможно, требуется тех. обслуживание.
8	Memory Parity Error	Ошибка четности при обращении к внутренней памяти устройства.

Вычисление контрольной суммы Modbus

Для протокола Modbus RTU 16-битная контрольная сумма (CRC) вычисляется по алгоритму, описанному в спецификации Modbus, в документе "Modbus Serial Line Protocol and Implementation Guide", раздел "CRC-generation". Передающее устройство формирует два байта контрольной суммы на основе данных сообщения, а принимающее устройство заново вычисляет контрольную сумму и сравнивает с полученной. Совпадение принятой и вычисленной контрольной суммы Modbus RTU считается индикатором успешного обмена данными.

В случае ограниченных вычислительных ресурсов для вычисления контрольной суммы существует функция, использующая табличные значения (также приведена в спецификации).

Карта регистров модулей реле

- English
- русский

Регистры модулей реле

Регистр / адрес		Тип	Чтение / запись	Значение по умолчанию	Формат	Назначение	С версии прошивки
0	канал 1	coil	RW	-	1 или 0	состояние канала реле	
1	канал 2						
2	канал 3						
3	канал 4						
4	канал 5						
5	канал 6						
0	вход 1	discrete input	R	-	1 или 0	состояние входа	
1	вход 2						
2	вход 3						
3	вход 4						
4	вход 5						
5	вход 6						
7	вход 0						
5		holding	RW	0		служебный регистр, значение должно быть 0	
6		holding	RW		▪ **0**: не восстанавливать состояние реле ▪ **1**: восстанавливать состояние реле	режим работы реле при отключении питания	1.5.3
8		holding	RW	0	секунды	таймаут для безопасного режима	
9	вход 1	holding	RW	0	▪ **0**: кнопки без фиксации ▪ **1**: выключатель с фиксацией ▪ **3**: отключить взаимодействие	режим взаимодействия отдельного цифрового входа с соответствующим релейным выходом. **В регистре 5 должно стоять значение по умолчанию (0).**	
10	вход 2						
11	вход 3						
12	вход 4				▪ **2**: отключать все реле при нажатии ▪ **4**: управлять по mapping-матрице ▪ **5**: управлять по mapping-матрице, через 20 минут повторно имитировать состояние ввода **Только для WB-MWAC**		1.9.0
13	вход 5			1			1.12.0
14	вход 6						
16	вход 0			2		режим работы цифрового входа 0 для отключения всех реле	1.9.0
20	вход 1	holdreg	RW	50	0 - 100	время защиты входа от дребезга в миллисекундах	1.13.0
21	вход 2						
22	вход 3						
23	вход 4						
24	вход 5						
25	вход 6						
27	вход 0						
32	вход 1	input	R	0	16-bit unsigned int	счетчик срабатываний входа	
33	вход 2						
34	вход 3						
35	вход 4						
36	вход 5						
37	вход 6						
39	вход 0						
40	вход 1	input	R	0	16-bit unsigned int	целая часть значения частоты сигнала	1.15.0
42	вход 2						
44	вход 3						
46	вход 4						
48	вход 5						
50	вход 6						
54	вход 0						
41	вход 1	input	R	0	16-bit unsigned int	дробная часть значения частоты сигнала 1 Гц = 65536 (2 ^ 16)	
43	вход 2						
45	вход 3						
47	вход 4						
49	вход 5						
51	вход 6						
55	вход 0						
123		input	R	0	мВ	Напряжение на	1.16.0

						микроконтроллере	
124		input	R	0	°C × 100	Внутренняя температура микроконтроллера	1.16.0
384-447		holding	RW	0		регистры mapping-матрицы	1.9.0

Общие регистры

Адрес		Описание	Тип данных Modbus	Доступ	Формат	Множитель	Единица измерения	Значения		
Dec	Hex							Возможные	По умолчанию	При ошибке
104-105	0x0068 - 0x0069	Время работы с момента загрузки	Input register	RO	u32	1	с			
110	0x006E	Скорость порта RS-485. Как настроить параметры порта RS-485.	Holding register		u16	100	Боды	12 — 1200 бит/с, 24 — 2400 бит/с, 48 — 4800 бит/с, 96 — 9600 бит/с, 192 — 19 200 бит/с, 384 — 38 400 бит/с, 576 — 57 600 бит/с, 1152 — 115 200 Кбит/с	96	
111	0x006F	Настройка бита чётности порта RS-485	Holding register	RW	u16			0 — нет бита чётности (none), 1 — нечётный (odd), 2 — чётный (even)	0	
112	0x0070	Количество стоп-битов порта RS-485	Holding register	RW	u16			1, 2	2	
120	0x0078	Регистр перезагрузки устройства без сохранения состояния	Holding register	RW	u16			любое, отличное от 0		
121	0x0079	Текущее напряжение питания	Input register	RO	u16	1	мВ			
128	0x0080	Modbus-адрес устройства (подробнее)	Holding register	RW	u16				На наклейке на корпусе устройства	
129	0x0081	Регистр перевода в режим обновления прошивки на 2 минуты	Holding register	RW	u16			любое, отличное от 0	0	
200-205	0x00C8 - 0x00CD	Модель устройства	Input register	RO	null-terminated string					
220-241	0x00DC - 0x00F1	Время и дата сборки прошивки	Input register	RO	null-terminated string					
220-248	0x00DC - 0x00F8	Хэш коммита и название ветки откуда собрана прошивка (2 символа в регистре)	Input register	RO	null-terminated string					
250-265	0x00FA - 0x0109	Версия прошивки	Input register	RO	null-terminated string					
266-269	0x010A - 0x010D	Расширение уникального идентификатора	Input register	RO	u64					
270-271	0x010E - 0x010F	Уникальный идентификатор (S/N)	Input register	RO	u32					
290-301	0x0122 - 0x012D	Сигнатура прошивки	Input register	RO	null-terminated string					
330-336	0x014A - 0x0150	Версия загрузчика	Input register	RO	null-terminated string					

Регистры настройки параметров обмена данными по RS-485 поддерживаются начиная с прошивки версии 1.6.0

MQTT

Contents

Описание

MQTT — протокол обмена данными, использующийся в программном обеспечении Wiren Board. Базовая информация по MQTT на Википедии (http://en.wikipedia.org/wiki/MQTT).

Драйверы, которые отвечают за аппаратную часть контроллера (цифровые и транзисторные входы, АЦП и т.п.) и функции внешних подключенных устройств публикуют их состояние по MQTT в виде сообщений. Веб-интерфейс читает эти сообщения и на их основе отображает состояние устройств.

Действия пользователя в веб-интерфейсе также публикуются по MQTT, где их получает драйвер и передает команду пользователя устройству.

Через MQTT работают: веб-интерфейс, движок правил и встроенные драйверы. Если вы разрабатываете собственное ПО в дополнение к предустановленному — мы рекомендуем использовать MQTT.

Примеры работы через MQTT

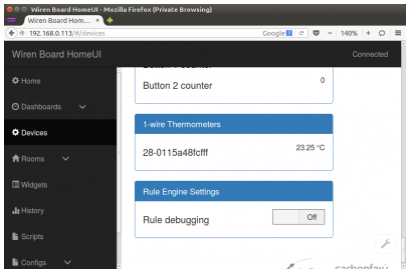
Получение значения от датчика температуры и вывод его в веб-интерфейс

К Wiren Board подключён датчик температуры по шине 1-Wire. Проследим, как данные с него через MQTT попадают в веб-интерфейс:

1. Драйвер, отвечающий за данную аппаратную функцию (wb-homa-w1 (https://github.com/contactless/wb-homa-drivers/tree/master/wb-homa-w1)) опрашивает подключённые к контролеру датчики 1-Wire.
2. При получении значения драйвер публикует по MQTT сообщение вида:

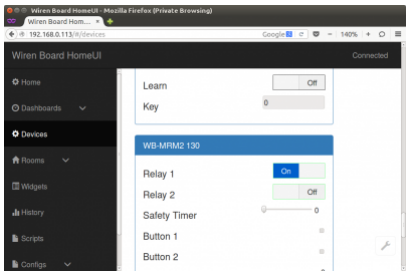
   ```
   /devices/wb-w1/controls/28-0115a48fcfff 23.25
   ```

   Оно значит, что от устройства 1-Wire с идентификатором *28-0115a48fcfff* получено значение 23.25 °C.
3. Веб-интерфейс, который подписан на все сообщения из MQTT, получает это сообщение и выводит значение датчика на страницу.



Показания датчика и его уникальный идентификатор на странице *Devices* веб-интерфейса

Нажатие кнопки в веб-интерфейсе и переключение реле на внешнем модуле



Веб-интерфейс после нажатия кнопки включения Реле 1 на подключённом по RS-485 релейном модуле WB-MRM2

К контроллеру по шине RS-485 подключён релейный модуль WB-MRM2. Пользователь в веб-интерфейсе нажимает кнопку включения реле. Проследим, как команда из веб-интерфейса попадает на внешний модуль:

1. После нажатия кнопки веб-интерфейс публикует по MQTT сообщение вида:

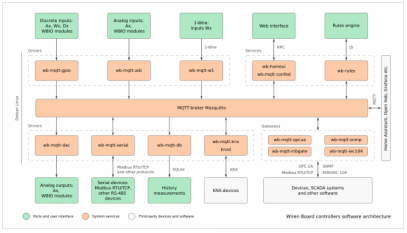
   ```
   /devices/wb-mrm2_130/controls/Relay 1/on 1
   ```

   Оно значит, что устройство WB-MRM2 с адресом *130* должно перевести *Реле 1* в состояние логической единицы — «включено».
2. Драйвер wb-mqtt-serial, отвечающий за данную аппаратную функцию, получает это сообщение (он *подписан* на все сообщения, относящиеся к подключённым по RS-485 устройствам) и посылает по шине RS-485 релейному модулю команду на включение первого реле.
3. Релейный модуль WB-MRM2 получает команду от контроллера, переключает реле и посылает обратно уведомление «Реле 1 включено».
4. Драйвер wb-mqtt-serial получает это уведомление по RS-485 и публикует по MQTT сообщение:

   ```
   /devices/wb-mrm2_130/controls/Relay 1 1
   ```

   Оно значит, что первое реле на устройстве WB-MRM2 с адресом *130* находится (уже переведено) в состоянии логической единицы — «включено».

Принцип работы MQTT


Через MQTT работают драйверы внутренних функций, внешних устройств, веб-интерфейс, система правил

Система сообщений MQTT построена по следующему принципу:

- есть иерархическая система «топиков» (как на обычных форумах в интернете).
- в эти топики клиенты (в случае Wiren Board это драйверы устройств и веб-интерфейс) могут писать сообщения и читать оттуда.
- чтобы следить за изменениями нужного топика (например, температуры на датчике), клиент может на него «подписаться» — тогда он получит все сообщения в этом топике.

Читать полное описание системы топиков и подписок (http://mosquitto.org/man/mqtt-7.html).

Отображение устройств в структуре сообщений

Логика организации топиков, соответствующих разным устройствам и их параметрам, в Wiren Board следует определённым правилам — так называемым соглашениям (Wiren Board MQTT Conventions (https://github.com/contactless/homeui/blob/master/conventions.md)).

Полный список MQTT-топиков можно увидеть на странице *Settings* веб-интерфейса в разделе *MQTT Channels* (появилось в последних версиях прошивки).

Клиенты MQTT

- драйверы внутренних аппаратных функций,
- драйверы внешних подключённых устройств,
- веб-интерфейс,
- движок правил,
- (если есть) собственные программы пользователя.

Структура сообщения о состоянии устройства

Вот сообщение от драйвера температурного датчика 1-Wire из примера выше:

```
/devices/wb-w1/controls/28-0115a48fcfff 23.25
```

Часть до пробела — название топика, после — само сообщение.

Название топика состоит из вложенных друг в друга «подтопиков»:

- `/devices` — коренной топик для всех «устройств» — как встроенных функций Wiren Board (цифровые, АЦП, ...), так и подключённых внешних (например, модулей реле).
- `/wb-w1` — подтопик, который наполняется драйвером 1-Wire.
- `/controls` — подтопик, который есть у всех устройств — именно в него записываются все их параметры, которые меняются («включено-выключено», значение датчика, ...).
- `/28-0115a48fcfff` — непосредственно сам «канал» («контрол») — топик, куда записывается значение с датчика. Его название совпадает с адресом 1-Wire датчика на шине.

Содержание сообщения:

- 23.25 — значение температуры

Если вы хотите самостоятельно написать драйвер устройства, и хотите, что оно отображалось на вкладке **Devices** и его можно было использовать в правилах, вам необходимо придерживаться такой же структуры топиков.

Структура сообщения об ошибке опроса устройства

Каждый «канал» («контрол») имеет «подтопик» `/meta/error`, в котором содержится информация о наличии ошибок взаимодействия с устройством. Ошибки получения данных (чтения) обозначаются символом **r**, ошибки записи — **w**.

Пример ошибки получения данных:

```
/devices/wb-w1/controls/28-0115a48fcfff/meta/error r
```

Это означает, что не удалось получить температуру термометра с адресом *28-0115a48fcfff*.

Драйвер wb-mqtt-serial устанавливает признак **r**, если не удалось запросить значение параметра устройства, признак **w** — не удалось передать значение устройству.

Драйвер wb-mqtt-adc (https://github.com/wirenboard/wb-homa-adc) устанавливает признак **r**, если не удалось получить значение соотвествующего канала АЦП.

Пример подписки

Клиенты, которые хотят следить за значением температуры, «подписываются» на этот топик, и им приходят все новые сообщения — меняющиеся значения температуры. Один из таких клиентов — веб-интерфейс.

Подписаться на сообщения можно и из консоли Linux при помощи утилиты `mosquitto_sub`:

```
~# mosquitto_sub -t '/devices/wb-w1/controls/28-0115a48fcfff' -v //получить сообщения из топика устройства 1-Wire с идентификатором 28-0115a48fcfff
/devices/wb-w1/controls/28-0115a48fcfff 22.75 //в этой строке и ниже - вывод утилиты, полученные сообщения
/devices/wb-w1/controls/28-0115a48fcfff 22.75
/devices/wb-w1/controls/28-0115a48fcfff 22.75
```

Полное описание работы с MQTT из командной строки смотрите ниже.

Структура сообщения — команды на изменение состояния

Подпишемся на сообщения о состоянии первого реле подключённого по RS-485 релейного модуля WB-MRM2:

```
~# mosquitto_sub -t "/devices/wb-mrm2_130/controls/Relay 1/#" -v
/devices/wb-mrm2_130/controls/Relay 1/meta/type switch
/devices/wb-mrm2_130/controls/Relay 1/meta/order 1
/devices/wb-mrm2_130/controls/Relay 1 0
```

Тут стоит отметить, что MQTT сохраняет часть сообщений (а именно те, которые при отправке были помечены флагом *retained*) вечно, поэтому после подписки вы получите даже те сообщения, которые были отправлены раньше, чем вы подписались.

Релейный модуль управляется драйвером Драйвер wb-mqtt-serial. У него есть соответствующий топик-«канал» («контрол») *Relay 1*. У него самого есть значение — *0* (реле выключено), и есть два подтопика. Один из них — служебный: в `/meta/type` записан тип «контрола». Здесь он *switch* — выключатель. Второй подтопик */on* — интереснее: в него клиенты пишут то состояние, в которое они хотят установить реле. Заметим, что оно может не совпадать некоторое время (затрачиваемое на процесс переключения) с тем состоянием, в котором реле находится. Драйвер при этом ведёт себя следующим образом: при получении сообщения в топик `/devices/wb-mrm2_130/controls/Relay 1/on` он физически включает реле на релейном модуле, а лишь затем записывает новое состояние реле в топик `/devices/wb-mrm2_130/controls/Relay`.

Например, если мы сейчас нажмём на кнопку реле в веб-интерфейсе (переключим его состояние), то получим новые сообщения:

```
/devices/wb-mrm2_130/controls/Relay 1/on 1
/devices/wb-mrm2_130/controls/Relay 1 1
```

- веб-интерфейс сначала «даёт указание» включить реле, потом драйвер его включает и записывает актуальное состояние в «канал» («контрол»).

Локальная работа с сообщениями MQTT

Программа (демон), отвечающая за рассылку сообщений от одних клиентов другим, называется брокером сообщений. В Wiren Board используется брокер сообщений Mosquitto (http://mosquitto.org/). Фактически, все драйверы и веб-интерфейс передают свои сообщения именно демону-брокеру Mosquitto.

Работа из командной строки

Управление устройствами из командной строки

Для управления устройством (изменения значения канала), необходимо отправить сообщение в топик `/devices/<device-id>/controls/<control-id>/on` (обратите внимание на /on в конце). Это делается с помощью консольной команды **mosquitto_pub**. Пример:

```
~# mosquitto_pub -t "/devices/wb-mrm2_130/controls/Relay 1/on" -m "1"
```

команда отправляет сообщение «1» (логическую единицу, «включить») в топик, соответствующий подключённому по RS-485 релейном модуле WM-MRM2 с адресом 130.

Слежение за состоянием устройства / подписка на топик

Клиенты, которые хотят следить за значением температуры, «подписываются» на этот топик, и им приходят все новые сообщения - меняющиеся значения температуры. Один из таких клиентов - веб-интерфейс.

Подписаться на сообщения можно и из консоли Linux при помощи утилиты **mosquitto_sub** (полное описание утилиты смотрите на http://mosquitto.org/man/mosquitto_sub-1.html (http://mosquitto.org/man/mosquitto_sub-1.html)):

```
~# mosquitto_sub -t '/devices/wb-w1/controls/28-0115a48fcfff' -v //получить сообщения из топика устройства 1-Wire с идентификатором 28-0115a48fcfff
/devices/wb-w1/controls/28-0115a48fcfff 22.75 //в этой строке и ниже — вывод утилиты, полученные сообщения
/devices/wb-w1/controls/28-0115a48fcfff 22.75
/devices/wb-w1/controls/28-0115a48fcfff 22.75
```

Метасимволы

Подписаться можно не только на один топик, но и на группу топиков по метасиволу. В MQTT применяется два метасимвола: **#** и **+**. Метасимвол *#* означает любое количество уровней вложенных топиков. Выполним команду

```
~# mosquitto_sub -t '/devices/wb-w1/#' -v
/devices/wb-w1/meta/name 1-wire Thermometers
/devices/wb-w1/controls/28-0115a48fcfff 22.812
/devices/wb-w1/controls/28-0115a48fcfff/meta/type temperature
/devices/wb-w1/controls/28-0115a48fcfff 22.75
```

В результате мы получили не только значения с «контрола» устройства, но и топики с метаданными — название драйвера устройства и тип «контрола» - *temperature*. Существует так же метасимвол **+**, который обозначает один уровень, а не произвольное количество,как **#**:

```
mosquitto_sub -v -t "/config/widgets/+/name"
```

В этом случае мы получим имена всех виджетов.

Полное описание системы топиков и подписок (http://mosquitto.org/man/mqtt-7.html).

Очистка очереди сообщений

Ненужные retained-сообщения могут остаться в системе MQTT после удаления неиспользуемых драйверов или отключения каких-либо устройств. Это приводит к тому, что несуществующие больше устройства могут отображаться в разделе *Devices* веб-интерфейса.

Для удаления топиков можно воспользоваться командой `mqtt-delete-retained`.

Например, удалим все топики, начинающиеся на `/devices/noolite_tx_1234/`

```
~# mqtt-delete-retained '/devices/noolite_tx_1234/#'
```

Для удаления топиков «по маске», можно циклично вызывать `runShellCommand` из правил. Таким образом, задача сводится к задаче работы со строками в js.

```
var deviceName = ['name1',.., 'nameN'];
var controlName = 'Temperature';

for (var i = 0; i<deviceName.length; i++) {
  runShellCommand ('mqtt-delete-retained /devices/'+ deviceName[i]  + '/controls/controlName/#');
}
```

Работа с MQTT из внешних программ

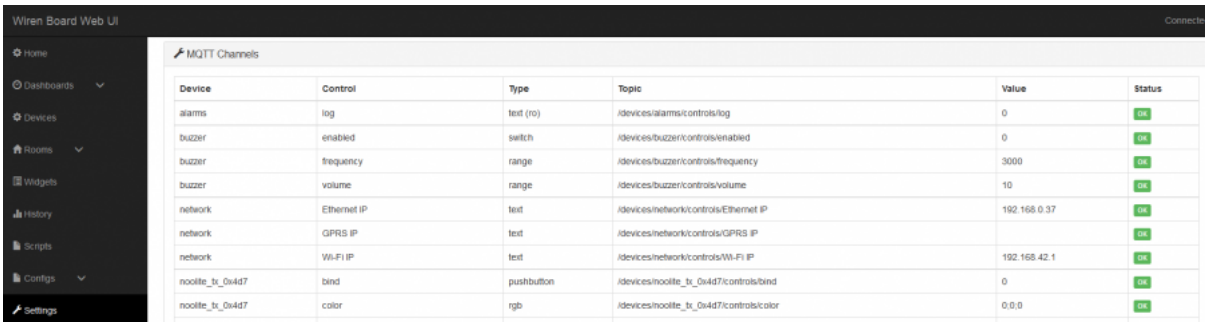
Если вы разрабатываете собственное ПО для Wiren Board, взаимодействовать с его аппаратными ресурсами лучше всего через протокол MQTT — ваша программа передаёт сообщение по MQTT, драйвер управляет устройством и вашей программе не нужно напрямую взаимодействовать с устройством на низком уровне.

Для того, чтобы отправлять сообщения MQTT, для многих языков программирования есть библиотеки:

- Python - [1] (https://github.com/contactless/mqtt-tools)
- C - [2] (http://mosquitto.org/man/libmosquitto-3.html)

Просмотр MQTT-каналов в web-интерфейсе

MQTT-названия устройств, их элементов управления и последние значения можно найти в разделе **Settings** web-интерфейса:



Device	Control	Type	Topic	Value	Status
alarms	log	text (ro)	/devices/alarms/controls/log	0	OK
buzzer	enabled	switch	/devices/buzzer/controls/enabled	0	OK
buzzer	frequency	range	/devices/buzzer/controls/frequency	3000	OK
buzzer	volume	range	/devices/buzzer/controls/volume	10	OK
network	Ethernet IP	text	/devices/network/controls/Ethernet IP	192.168.0.37	OK
network	GPRS IP	text	/devices/network/controls/GPRS IP		OK
network	Wi-Fi IP	text	/devices/network/controls/Wi-Fi IP	192.168.42.1	OK
noolite_tx_0x4d7	bind	pushbutton	/devices/noolite_tx_0x4d7/controls/bind	0	OK
noolite_tx_0x4d7	color	rgb	/devices/noolite_tx_0x4d7/controls/color	0;0;0	OK

Информация об MQTT-названиях устройств

Работа с сообщениями MQTT с внешнего устройства

Установленный на контроллер брокер mosquitto по умолчанию принимает подключения внешних клиентов по порту 1883 без пароля.

Например, если контроллер имеет адрес 192.168.0.67, его топики можно прочитать с другого компьютера с Linux, находящегося в той же сети:

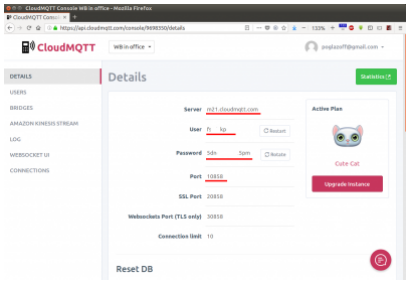
```
mosquitto_sub -h 192.168.0.67 -p 1883 -v -t "/devices/power_status/controls/Vin"
```

Настройка MQTT моста (bridge)

MQTT мост (bridge) — это функция MQTT-брокера, позволяющая пересылать все или часть сообщений на другой MQTT-брокер, и получать сообщения с другого брокера обратно.

Эту функцию удобно применять в следующей ситуации: хотя на самом контроллере уже есть MQTT-брокер, к нему часто неудобно подключаться, так как контроллер может не иметь белого IP-адреса, а иногда может быть выключен или не в сети. В таком случае удобно иметь отдельный брокер в облаке с фиксированным адресом, который будет всегда онлайн, и на который будут пересылаться сообщения с брокера контроллера.

На контроллере эта функция настраивается в конфигурационных файлах *mosquitto*. Самый простой вариант конфигурации приведён ниже.



Настройки брокера Cloud MQTT

Настройка моста с MQTT брокером Cloudmqtt

Задача: настроить пересылку всех сообщений MQTT на популярный дешёвый облачный MQTT брокер http://cloudmqtt.com/ и обратно.

Решение:

1. Зарегистрируйтесь на http://cloudmqtt.com/
2. Зайдите в свой аккаунт на http://cloudmqtt.com/ и посмотрите настройки: сервер, порт, логин, пароль.
3. Зайдите на контроллер и добавьте в конец файла `/etc/mosquitto/mosquitto.conf` следующие строки:

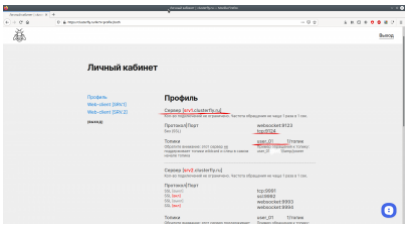
```
connection cloudmqtt
address m21.cloudmqtt.com:10858
remote_username fs_user_kp
remote_password 5dn_pass_pm
clientid pavel_test
try_private false
start_type automatic
topic # both
```

(последняя строка говорит, что нужно пересылать все сообщения (метасимвол **#**, смотрите описание выше) в обе (**both**) стороны (с брокера контроллера на облачный брокер и обратно)
Более подробное описание всех опций смотрите на https://mosquitto.org/man/mosquitto-conf-5.html.

4. Перезапустите `mosquitto`, выполнив в консоли:

```
service mosquitto restart
```

Настройка моста с MQTT брокером Clusterfly



Настройки брокера CLUSTERFLY

Задача: настроить пересылку всех сообщений MQTT на бесплатный облачный MQTT брокер https://clusterfly.ru/ и обратно.

Решение:

1. Зарегистрируйтесь на https://clusterfly.ru/
2. Зайдите в свой аккаунт на https://clusterfly.ru/ и выберите "Профиль" посмотрите настройки: сервер, порт, логин и сгенерируйте пароль. Для пересылки используйте сервер `srv1.clusterfly.ru`.
3. Зайдите на контроллер и добавьте в конец файла `/etc/mosquitto/mosquitto.conf` следующие строки:

```
connection clusterfly
address srv1.clusterfly.ru:9124
remote_username user_xxxxxx
remote_password pass_xxxxxx
try_private false
notifications true
notification_topic /client/wb_6/bridge_status
start_type automatic
topic /# both 0 "" user_xxxxxx
bridge_insecure true
cleansession false
```

   строка `'topic /# both 0 "" user_xxxxxx'` говорит, что нужно пересылать все сообщения (метасимвол **#**, смотрите описание выше) в обе (**both**) стороны (с брокера контроллера на облачный брокер и обратно) с префиксом (**user_xxxxxx**). Пример обращения к топику: `user_xxxxxx/devices/wb-mr6c_200/controls/K2`.

4. Перезапустите `mosquitto`, выполнив в консоли:

```
service mosquitto restart
```

   Потребуется подождать некоторое время пока брокер `mosquitto` сможет организовать соединение. Подписавшись на контроллере к топику `/client/wb_6/bridge_status` можно увидеть статус соединения.

```
mosquitto_sub -v -t "/client/wb_6/bridge_status"
/client/wb_6/bridge_status 0
/client/wb_6/bridge_status 1
```

Другие облачные брокеры

Список облачных брокеров, в том числе бесплатных: https://github.com/mqtt/mqtt.github.io/wiki/public_brokers (https://github.com/mqtt/mqtt.github.io/wiki/public_brokers)

Задача: настроить пересылку топика MQTT на другой контроллер. Есть два контроллера в одной сети:

1. *DestinationController* с адресом `10.0.0.40`, на этот контроллер получать топик.
2. *SourceController* с адресом `10.0.0.70`, с этого контроллера будем забирать топик.

На *SourceController* есть `/client/temp1`, но его нужно видеть на *DestinationController* в `/devices/temp1`.

Решается двумя способами, можно с *SourceController* публиковать на *DestinationController* или с *DestinationController* подписаться на топик *SourceController* и забирать изменения. От выбора стратегии зависит на каком контроллере будем проводить настройки.

Мы будем настраивать *DestinationController*.

Решение: На контроллере *DestinationController* добавьте в конфиг:

```
mcedit /etc/mosquitto/conf.d/bridge.conf
```

Строки:

```
connection wb_40
address 10.0.0.70
notifications true
notification_topic /client/wb_40/bridge_status
keepalive_interval 20
restart_timeout 20

topic /temp1/# in 2 /devices /client
```

Перезапустите `mosquitto` на *DestinationController*:

```
systemctl restart mosquitto; systemctl status mosquitto
```

ВАЖНО: перед перезапуском желательно остановить watchdog. В случае ошибки в конфигурационных файлах брокер не запустится и watchdog вызовет перезапуск контроллера.

Рассмотрим подробнее строчку `topic /temp1/# in 2 /devices /client` где:

- `/temp1/#` это топик от «корня». На брокере-источнике /client/**temp1**.
- `in` — только забираем, изменения на контроллере не передадутся на сервер.
- `/devices` — «корень» **куда** располагаем локально (на контроллере на котором **настраиваем**). На контроллере *DestinationController* это `/devices` и полный путь будет выглядеть как **/devices**/temp1.
- `/client` — «корень» откуда забираем на удаленном. На контроллере *SourceController* это `/client` и полный путь будет выглядеть как **/client**/temp1.

Проверка: Дожидаемся статуса бриджа «1» в топике `/client/wb_40/bridge_status` на контроллере *SourceController*. На нем же публикуем:

```
for i in {1..25}
do
mosquitto_pub  -t "/client/temp1/temp" -m "$i" -r
done
```

Подписавшись на контроллере *DestinationController* на целевой топик можно видеть:

```
mosquitto_sub -v -t /devices/temp1/#
/devices/temp1/temp
/devices/temp1/temp 1
/devices/temp1/temp 2
/devices/temp1/temp 3
/devices/temp1/temp 4
/devices/temp1/temp 5
/devices/temp1/temp 6
/devices/temp1/temp 7
/devices/temp1/temp 8
/devices/temp1/temp 9
/devices/temp1/temp 10
/devices/temp1/temp 11
/devices/temp1/temp 12
/devices/temp1/temp 13
/devices/temp1/temp 14
/devices/temp1/temp 15
/devices/temp1/temp 16
/devices/temp1/temp 17
/devices/temp1/temp 18
/devices/temp1/temp 19
/devices/temp1/temp 20
/devices/temp1/temp 21
/devices/temp1/temp 22
/devices/temp1/temp 23
/devices/temp1/temp 24
/devices/temp1/temp 25
```

Создание своего брокера MQTT

Вы можете создать отдельный брокер на компьютере или на VDS-сервере в интернете и собирать на нем данные с контроллеров.

Инициировать соединение будет контроллер, поэтому контроллеру не нужен «белый» IP-адрес. Если контроллеров несколько, вы можете разделить данные от них на брокере, для этого в настройках моста укажите для каждого контроллера отдельный корневой топик.

Установка брокера

1. Установите `mosquitto`:

   ```
   sudo apt update && sudo apt install mosquitto mosquitto-clients -y
   ```

2. Отключите возможность анонимного входа, для этого:

   - Откройте файл конфигурации в редакторе

     ```
     sudo nano /etc/mosquitto/mosquitto.conf
     ```

   - Добавьте в конец файла строки:

     ```
     #Turn on port listening
     listener 1883
     #Disable anonomous login:
     allow_anonymous false
     #Password file:
     password_file /etc/mosquitto/mosquitto.pwd
     ```

3. Создайте пароль для пользователя, в примере использован пользователь `test` с паролем `wbpassword`:

   ```
   sudo mosquitto_passwd -c /etc/mosquitto/mosquitto.pwd test
   ```

4. Введите пароль дважды и запомните его, он вам пригодится ниже.
5. Перезапустите `mosquitto` и проверьте его состояние:

```
sudo systemctl restart mosquitto && sudo systemctl status mosquitto
```

6. Подключитесь к брокеру для проверки, в примере адрес брокера `127.0.0.1`:

```
mosquitto_sub -v -h 127.0.0.1 -u test -P wbpassword -t "/#"
```

7. Запустите в другой консоли команду ниже и убедитесь, что топик меняется:

```
for i in {1..25}; do mosquitto_pub -h 127.0.0.1 -u test -P wbpassword -t "/client/temp1/temp" -m "$i" -r; done
```

Брокер установлен и доступен с контроллера. Для подключения нужно ввести логин и пароль.

Настройка моста на контроллере

Создайте файл конфигурации моста, для этого:

1. Создайте файл `/etc/mosquitto/conf.d/bridge1.conf`

```
nano /etc/mosquitto/conf.d/bridge1.conf
```

2. Вставьте в него строки, где `10.0.0.105` — адрес брокера:

```
connection bridge1
#address of server
address 10.0.0.105
notifications true
notification_topic /clientnotification/bridge1_status
remote_username test
remote_password wbpassword

topic /temp1/# both 2 /devices /controller
```

Содержимое топика `/devices/temp1/#` контроллера будет отображаться на брокере в `/controller`. Вместо `/controller` можете указать уникальное имя контроллера, например, серийный номер.

Центр документации

- English
- русский

Контроллеры

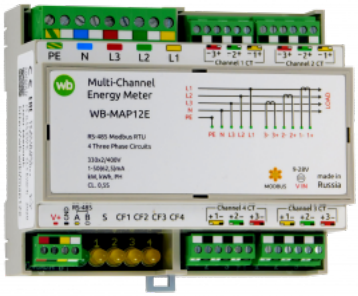
Универсальные контроллеры автоматизации, работающие под управлением свободного программного обеспечения. Применяются в задачах мониторинга серверного и климатического оборудования, диспетчеризации и сбора данных с приборов учёта, в качестве основы для «умного дома» и автоматизации производств.

Wiren Board 6

- Wiren Board 6 — универсальный контроллер для типовых задач.
- Wiren Board 7 — мощный универсальный контроллер для ресурсоёмких задач.
- Модули расширения устанавливаются внутрь корпуса контроллера, совместимы с Wiren Board 6 и Wiren Board 7.
- Модули ввода-вывода стыкуются к контроллеру Wiren Board справа через боковой разъём. Совместимы с Wiren Board 5, Wiren Board 6, Wiren Board 7.
- Поддерживаемые устройства — стороннее оборудование, работающее с контроллером Wiren Board.
- Ответы на часто задаваемые вопросы (FAQ) — сборник готовых решений и советов, полезные ссылки
- Диагностика ошибок в работе контроллера Wiren Board — сборник советов по диагностике контроллера

Счётчики электроэнергии и вольтметры

- WB-MAP12E — многоканальный счетчик электроэнергии (измерение всплесков тока и напряжения)
- WB-MAP6S — однофазный многоканальный счетчик электроэнергии
- WB-MAP3E — трехфазный счетчик электроэнергии (измерение всплесков тока и напряжения)
- WB-MAP3ET — трехфазный счетчик электроэнергии (измерение всплесков тока и напряжения) со встроенными трансформаторами
- WB-MAP3EV — трехфазный вольтметр
- WB-CT309 — сборка неразъемных трансформаторов для счетчиков MAP



WB-MAP12E

Релейные модули

О выборе модуля реле читайте в статье Рекомендации по выбору реле для нагрузки.

- WB-MR3LV/K, WB-MR6LV/K — 3- и 6-канальные модули реле общего назначения с переключаемой группой контактов
- WB-MR3LV/I, WB-MR6LV/I — мощные 3- и 6-канальные модули реле с переключаемой группой контактов
- WB-MR3LV/S, WB-MR6LV/S — очень мощные 3- и 6-канальные модули реле с нормально открытыми контактами
- WB-MRPS6 — мощный 6-канальный модуль реле без входов
- WB-MRWL3 — очень мощный 3-канальный модуль реле
- WB-MR6C — модуль реле 6-канальный
- WB-MR6C/NC — модуль реле 6-канальный с нормально-замкнутыми контактами
- WB-MR6CU v.2 — компактный модуль реле 6-канальный
- WB-MRM2-mini — компактный 2-канальный модуль реле
- WB-MRWM2 — мощный 2-канальный модуль реле с **измерением мощности**

WB-MRM-2mini

Датчики

- WB-MS — универсальный датчик температуры, влажности, освещённости, шума
- WB-MSW v.3 — датчик климата и CO2 в настенном исполнении v.3
- WB-MAI11 — модуль аналоговых входов

Диммеры

- WB-MRGBW-D — четырёхканальный диммер светодиодных лент
- WB-AMPLED — четырёхканальный усилитель для светодиодных лент
- WB-MDM3 — трёхканальный диммер светодиодных ламп и ламп накаливания 230 В

Преобразователи интерфейсов

- WB-MIO — преобразователь интерфейса I2C (WBIO) в RS-485 с поддержкой Modbus RTU
- WB-MIO-E v.2 — преобразователь интерфейса I2C (WBIO) в RS-485 с поддержкой Modbus RTU и RS-485 (Modbus) в Ethernet с поддержкой Modbus RTU over TCP и Modbus TCP
- WB-MGE v.2 — преобразователь интерфейса RS-485 (Modbus) в Ethernet с поддержкой Modbus RTU over TCP и Modbus TCP

Сетевые карты для контроллеров холодильного оборудования

- WB-REF-U-CR — сетевая карта для контроллеров Carel BASIC(PYEZ)/EASY(PJEZ)
- WB-REF-DF-178A — сетевая карта для контроллеров Danfoss EKC 202/EKC 210
- WB-REF-DF-ERC21— сетевая карта для контроллеров Danfoss ERC 211/ERC 213/ERC 214

Разное

- WB-MAO4 — модуль аналоговых выходов 0-10В 4-канальный
- WB-UPS — модуль бесперебойного питания на литий-полимерных аккумуляторах
- WB-UPS v.2 — модуль бесперебойного питания на литий-полимерных аккумуляторах
- WB-MCM8 — модуль счетных входов 8-канальный
- WB-MIR v.2 — устройство ИК-управления
- WB-M1W2 — преобразователь для термометров 1-Wire
- WB-MAI2-mini/CC — модуль измерения токового сигнала
- WB-MWAC — модуль для учета водопотребления и контроля протечек
- WB-DEMO-KIT v.3 — «Демо-чемодан»: набор интегратора, для демонстрации заказчику или самостоятельного быстрого освоения устройств Wiren Board
- Демонстрационный стенд — пример сборки демонстрационного стенда с оборудованием Wiren Board. Можно посмотреть в нашем офисе.
- **Как подключить устройство RS-485**

WB-MIR v.2

Снятые с производства устройства

- WB-MR3HV, WB-MR6HV — мощные 3- и 6-канальные модули реле
- WB-MIO-E v.1 — устройство заменено WB-MIO-E v.2
- WB-MGE v.1 — устройство заменено WB-MGE v.2
- WBC-2G v.1 — модуль заменён WBC-2G v.2
- WBC-3G — модуль заменён WBC-4G
- WB-MSW2 — датчик климата и CO2 в настенном исполнении v.2
- WB-MSGR — электрохимические датчики газа WB-MSGR с встроенным реле
- WB-MDM2 — двухканальный диммер светодиодных ламп и ламп накаливания 230 В
- WB-MCM16 — модуль счетных входов 16-канальный
- WB-MRGB — диммер светодиодных лент
- WB-MRGB-D — диммер светодиодных лент (на дин-рейку)
- WB-MSW — универсальный датчик температуры, влажности, освещённости, шума в настенном исполнении v.1
- WB-MIR v1 — устройство ИК-управления
- WB-MAP12H — многоканальный счетчик электроэнергии (измерение гармонических составляющих тока и напряжения)
- WB-MAP3H — трехфазный счетчик электроэнергии (измерение гармонических составляющих тока и напряжения)
- WB-MR6F — модуль реле для ступенчатого управления двумя вентиляторами
- WB-MR11 — модуль реле 11-канальный
- WB-MR14 — модуль реле 14-канальный
- WB-MRM2 — модуль реле 2-канальный
- WBIO-AI-DCM-4 — модуль измерения токов и напряжения, заменён модулем WBIO-AI-DV-12
- WBE2S-R-433MHZ — модуль расширения 433 MHz. Доступен по запросу
- WB_AC_rev._E2.0 — автономный/сетевой IP-контроллер доступа со встроенным считывателем карт Mifare

Wiren Board 4


Wiren Board NETMON-1

- WB-MGW — преобразователь интерфейсов WB-MGW Wi-Fi — RS-485 предназначен для создания моста между сетями Wi-Fi и RS-485
- Wiren Board NETMON-2 — контроллер для автоматизации и мониторинга в 19" стойку. Состоит из Wiren Board 5 + модуль реле + модуль для «сухих контактов» + модуль резервного питания в корпусе под 19" стойку
- Wiren Board NETMON-1 — контроллер в 19" стойку. Программное обеспечение практически полностью совпадает с таковым у Wiren Board 5. Устройства отличаются набором портов и аппаратными характеристиками
- Wiren Board 5 — предыдущая модель контроллера
- Wiren Board 4 — устаревшая версия контроллера
- Wiren Board Smart Home rev. 3.5 — устаревшая версия контроллера
- Wiren Board rev. 2.8 — устаревшая версия контроллера
