wirenboard

https://wirenboard.com/wiki/RS-485
15-03-2022 15:10

# RS-485

## Руководство по эксплуатации

Самая актуальная документация всегда доступна на нашем сайте по ссылке: https://wirenboard.com/wiki/RS-485

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

# Содержание



# RS-485

## Contents



## 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 (https://www.bookasutp.ru/Chapter2_3.aspx) — 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.

## Как ускорить опрос устройств

Для ускорения опроса устройств по шине RS-485 рекомендуем:

1. Увеличить скорость обмена до 115200 бит/с. На разумных длинах и топологии сети все должно нормально работать. Если на шине есть устройства, не поддерживающие эту скорость, см. пункт 3.
2. Отключить через веб-интерфейс в настройках устройства ненужные каналы.

3. Разделить устройства по типам и портам, контроллере 2 порта RS-485 и еще 3 можно добавить модулями расширения:
    - Устройства, не поддерживающие скорость 115200, подключите отдельно.
    - Счетчики МАР так же подключите отдельно или с оборудованием, не требующим быстрой реакции. В счетчиках очень много параметров, опрос идет медленно.
    - При большом количестве устройств разделите их на несколько портов. При прочих равных скорость вырастет кратно количеству портов.

## 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



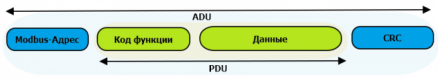
### Основные понятия

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.

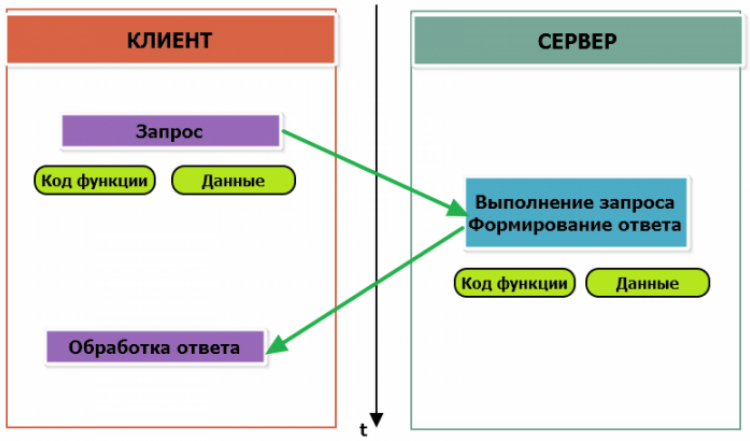
Ведущее устройство ("мастер", или "клиент") периодически опрашивает "ведомое", или "сервер". Ведущее устройство не имеет адреса, передача сообщений от устройства-сервера ведущему без запроса ведущего в протоколе не предусмотрена.



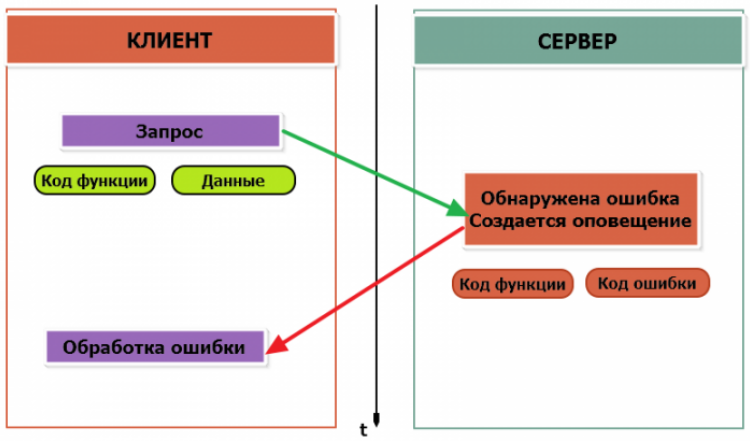
Датаграмма Modbus в общем виде

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.

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

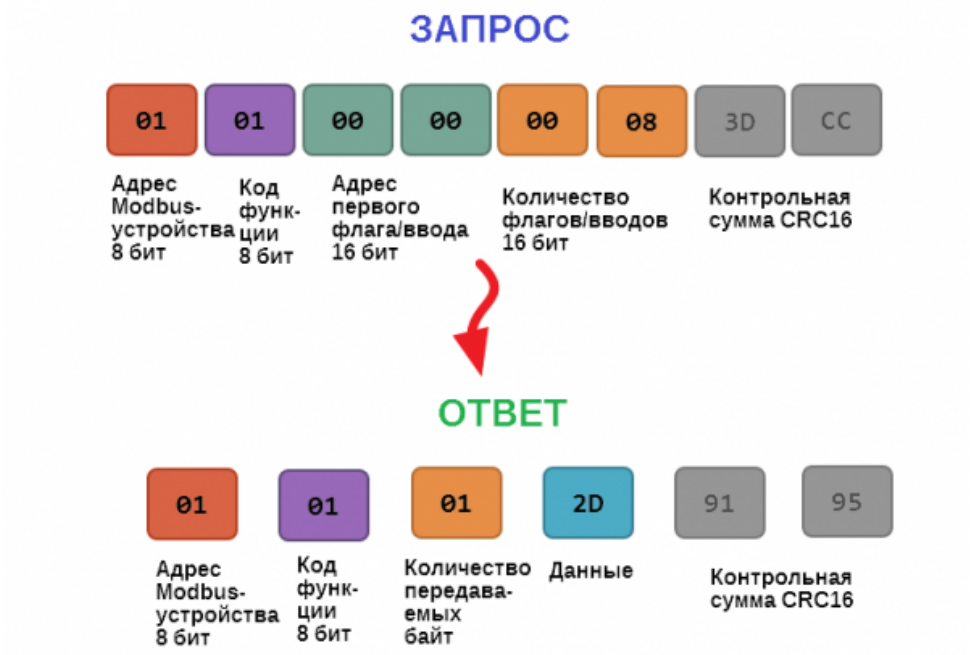
В следующей таблице приведены наиболее распространенные коды функций Modbus:

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

# Настройка параметров подключения по RS-485 для Modbus-устройств Wiren Board

- English
- русский

## Contents



## Введение

Устройства Wiren Board управляются по протоколу Modbus RTU и на физическом уровне подключаются через интерфейс RS-485.

### Параметры порта по умолчанию

| Значение по умолчанию | Название параметра в веб-интерфейсе | Параметр |
|---|---|---|
| 9600 | Baud rate | Скорость, бит/с |
| 8 | Data bits | Количество битов данных |
| None | Parity | Бит чётности |
| 2 | Stop bits | Количество стоповых битов |

## Изменение скорости обмена

> **Скоро в стабильном релизе, а пока доступно в testing**

Для ускорения отклика устройств на шине RS485 рекомендуем поднять скорость обмена до 115 200 бит/с.

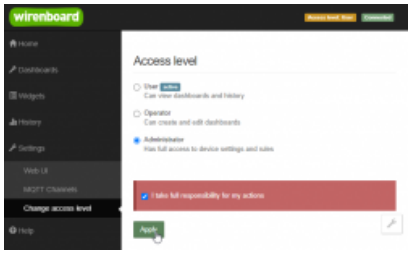
Отметим, что низкая скорость обмена прощает многие ошибки построения шины, но на высоких скоростях выполнение рекомендаций по построению шины обязательно.

### Смена уровня доступа к веб-интерфейсу

Для изменения настроек контроллера у вас должен быть уровень доступа *Administrator*.

Изменить его можно в разделе **Settings → Change access level**.

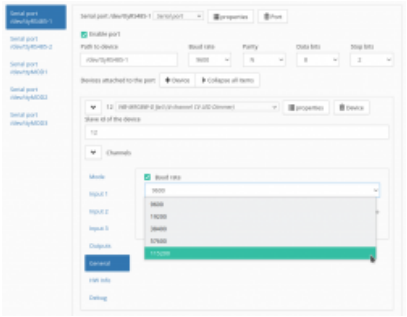
После завершения настроек рекомендуем поставить уровень доступа *User* или *Operator* — это поможет не совершить случайных ошибок при ежедневной работе с веб-интерфейсом.



Уровень «Администратор»

### Настройка

Увеличим скорость обмена в Modbus-устройствах Wiren Board со значения по умолчанию до 115 200 бит/с:



Выбор желаемой скорости обмена в настройках устройства

1. Подключите и настройте все устройства на скорости 9600 бит/с, которая стоит у них по умолчанию.
2. Убедитесь, что все работает как надо: данные идут со всех устройств, каналы не горят красным, в системном журнале нет ошибок порта.
3. Откройте веб-интерфейс контроллера и перейдите **Settings** → **Configs** → **Serial Device Driver Configuration**.
4. Выберите нужный порт, в параметрах устройства в группе **General** поставьте флажок **Baud rate** и выберите желаемую скорость обмена: 115 200 бит/с. Скорость порта пока оставьте прежней.
5. Вверху страницы нажмите на кнопку **Save**, это запишет новое значение скорости в устройство. Но так как порт работает на старой скорости, то устройства отвечать не будут.
6. Укажите в настройках порта ту же скорость, которую вы выбрали в настройках устройства: 115 200 бит/с.
7. Снова сохраните настройки. Теперь настройки устройства и порта совпадают, устройство должно начать отвечать.

## Настройка параметров обмена

Чтобы изменить параметры подключения, нам понадобится:

- знать текущие настройки подключения устройства;
- контроллер с утилитой modbus_client или компьютер с адаптером USB-RS485 и программой для работы с Modbus;
- номера регистров, которые описаны в таблице общих регистров.

Подготовка:

1. Подключите устройство по шине RS-485 к контроллеру или другому оборудованию, где будете выполнять команды.
2. Если вы выполняете команды на контроллере:
    - откройте консоль контроллера по SSH,
    - остановите драйвер wb-mqtt-serial.
3. Можно менять настройки устройств.

Допустим, у нас есть Modbus-устройство Wiren Board с заводскими параметрами подключения, Modbus-адресом 1 и подключённое к порту `/dev/ttyRS485-1`.

Изменим адрес устройства, для этого запишем в регистр 128 новый адрес, например 12:

```
modbus_client --debug -mrtu -b9600 -pnone -s2 /dev/ttyRS485-1 -a1 -t0x06 -r128 12
```

Теперь изменим скорость порта устройства с 9600 бит/с на 115 200 бит/с, для этого запишем в регистр 110 новое значение, формат которого можно посмотреть в таблице общих регистров:

```
modbus_client --debug -mrtu -b9600 -pnone -s2 /dev/ttyRS485-1 -a1 -t0x06 -r110 1152
```

Теперь устройство передаёт и принимает данные на скорости 115 200 бит/с.

Остальные параметры меняются аналогично: смотрите, в каком регистре хранится значение и записываете в него новое.

## Если параметры подключения неизвестны

Бывает так, что параметры подключения устройства неизвестны, то можно или сбросить их к заводским, или узнать перебором, для этого загрузите на контроллер скрипт Perebor.sh.tar.gz и выполните его. Если адрес, к которому подключено устройство отличается от `/dev/ttyRS485-1`, измените его в теле скрипта.

Как это работает: мы обращаемся к регистру 128, в котором во всех modbus-устройствах Wiren Board хранится modbus-адрес. Вывод скрипта будет содержать строки, подобные этим:

```
Speed:9600      Stop bits:1     Parity:none     Modbus address:0x0001
Speed:9600      Stop bits:2     Parity:none     Modbus address:0x0001
```

Для стоп-битов, скорее всего, вы получите два значения: 1 и 2. Уточнить настройку можно считав значение из регистра 112 с уже известным адресом, скоростью, четностью:

```
modbus_client --debug -mrtu -b9600 -pnone -s2 /dev/ttyAPP1 -a0x01 -t0x03 -r112
```

или

```
modbus_client --debug -mrtu -b9600 -pnone -s1 /dev/ttyAPP1 -a0x01 -t0x03 -r112
```

```
   SUCCESS: read 1 of elements:
   Data: 0x0002
```

Если при чтении из регистра 112 вы получаете ошибку — устройство не поддерживает изменение параметров подключения. В этом случае для подключения используется значение по умолчанию,2 стоп-бита.

# Драйвер wb-mqtt-serial

## Contents



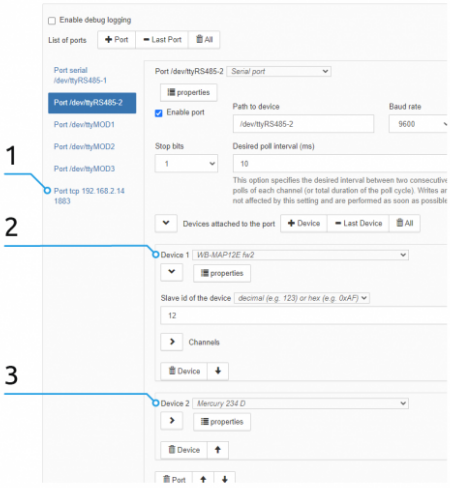
## Описание

### Общая информация

*wb-mqtt-serial* — драйвер master-slave протоколов для устройств, подключённых:

- к шине RS-485 — протокол Modbus RTU и других;
- через Ethernet — протоколы Modbus TCP, Modbus over TCP и другие;
- к разъёмам MOD1–MOD3 — при наличии модулей расширения, использующих обмен по UART.

Драйвер опрашивает serial-устройства и публикует данные в топики MQTT-брокера. Устройства настраиваются через веб-интерфейс.




Драйвер wb-mqtt-serial может одновременно опрашивать устройства, работающие по разным протоколам:
1 — виртуальный порт для устройств с протоколом Modbus TCP,
2 — устройство работает по протоколу Modbus RTU,
3 — устройство работает по протоколу DLMS

wb-mqtt-serial использует систему JSON-шаблонов, которые описывают подключённые устройства: тип протокола, номера регистров, название параметров и контролов в веб-интерфейсе контроллера. В стандартной поставке есть шаблоны для всех устройств Wiren Board, а также некоторых сторонних устройств: счётчики электроэнергии, частотные преобразователи, холодильные контроллеры и другие.

Если ваше устройство работает по поддерживаемому драйвером протоколу, но в стандартной поставке под него нет шаблона — можете написать шаблон сами.

Файлы и папки:

- `/etc/wb-mqtt-serial.conf` — файл настроек драйвера, редактировать вручную не рекомендуем;
- `/usr/share/wb-mqtt-serial/templates` — папка с предустановленными шаблонами;
- `/etc/wb-mqtt-serial.conf.d/templates` — папка для пользовательских шаблонов, которые имеют приоритет на предустановленными.

О том, как получить доступ к файлам и папкам, читайте в статье Просмотр файлов контроллера с компьютера.

Полное описание драйвера, список поддерживаемых протоколов и примеры шаблонов, смотрите **в репозитории на Github (https://github.com/contactless/wb-mqtt-serial)**.

### Особенности

При работе с Modbus-устройствами, драйвер оптимизирует запросы к устройствам: считывает несколько регистров подряд, не выдерживает некоторые задержки, рекомендованные стандартом.

Поэтому при написании шаблона для сторонних Modbus-устройств, нужно указать параметр *guard_interval_us*, который рассчитывается по формуле:

```
guard_interval_us = (3.5*11*106)/(скорость в бит/с)
```

Так же этот параметр можно установить через веб-интерфейс:

- Guard interval (us) — для порта.
- Additional delay before each writing to port (us) — для устройства.

Если при чтении регистра устройства возникла ошибка, то соответствующий контрол в веб-интерфейсе будет окрашен в красный цвет. Аналогично с устройством — если оно давно не отвечает, то все его контролы будут окрашены красным.

## Управление драйвером

Обычно драйвер запускается автоматически при загрузке контроллера и перезапускается при сохранении файла конфигурации в веб-интерфейсе.

Также можно управлять драйвером в ручном режиме — это может быть полезно для поиска ошибок в конфигурационном файле или если вам нужно освободить порт для использования modbus_client.

Для выполнения команд подключитесь к контроллеру по SSH. Доступны команды:

```
systemctl stop wb-mqtt-serial # остановить
systemctl start wb-mqtt-serial # запустить
systemctl restart wb-mqtt-serial # перезапустить
wb-mqtt-serial -c /etc/wb-mqtt-serial.conf -d # запустить в отладочном режиме с указанием пути к конфигурационному файлу
```

## Диагностика неполадок

Если возникли проблемы с запуском драйвера, например, новое устройство не появилось, то можно узнать причину: выполните команду `systemctl status wb-mqtt-serial` и в последних двух строчках ответа будет подсказка.

В примере файл конфигурации содержит синтаксическую ошибку во второй строке на 14 позиции:

```
# systemctl status wb-mqtt-serial
● wb-mqtt-serial.service - MQTT Driver for serial devices
   Loaded: loaded (/lib/systemd/system/wb-mqtt-serial.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2021-01-28 15:10:51 +04; 4s ago
  Process: 23682 ExecStart=/usr/bin/wb-mqtt-serial (code=exited, status=0/SUCCESS)
 Main PID: 23682 (code=exited, status=0/SUCCESS)

Jan 28 15:10:47 wirenboard-A6XXXT2R systemd[1]: Started MQTT Driver for serial devices.
Jan 28 15:10:51 wirenboard-A6XXXT2R wb-mqtt-serial[23682]: ERROR: [serial] Failed to parse JSON /etc/wb-mqtt-serial.conf:* Line 2, Column
14
Jan 28 15:10:51 wirenboard-A6XXXT2R wb-mqtt-serial[23682]:   Syntax error: value, object or array expected.
```

Проверить только шаблоны, в том числе и не подключённые в файле конфигурации, можно командой:

```
# wb-mqtt-serial -g
<3>ERROR: [serial config] Failed to parse /usr/share/wb-mqtt-serial/templates/config-bac-6000-series.json
Failed to parse JSON /usr/share/wb-mqtt-serial/templates/config-bac-6000-series.json:* Line 12, Column 5
  Missing ',' or '}' in object declaration
```

Проверить файл конфигурации и шаблоны на ошибки:

```
# wb-mqtt-serial -j
<3>ERROR: [serial config] Failed to parse /usr/share/wb-mqtt-serial/templates/config-wb-mdm3.json
Failed to parse JSON /usr/share/wb-mqtt-serial/templates/config-wb-mdm3.json:* Line 8, Column 9
  Missing ',' or '}' in object declaration

<3>ERROR: [serial] Can't find template for 'WB-MDM3'
```

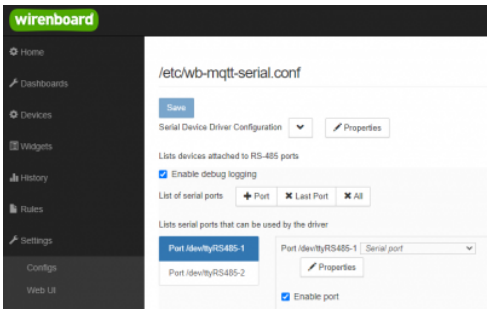
При необходимости, можно добавить путь к файлу, который нужно проверить:

```
wb-mqtt-serial -c /etc/wb-mqtt-serial.conf -j
```

# Включение отладки

Иногда нужно включить отладочный режим драйвера. Это можно сделать из командной строки или через веб-интерфейс.

> При включённой отладке размер системного журнала будет быстро расти, поэтому не забудьте отключить отладку, когда необходимость в ней отпадет.



Веб-интерфейс. Флажок *Enable debug logging* установлен, отладка включена

Включение отладки через веб-интерфейс:

1. Зайдите в веб-интерфейс контроллера
2. Если вы работаете под обычным пользователем, то смените уровень доступа
3. Перейдите **Settings → Configs → Serial Device Driver Configuration**
4. Установите флажок **Enable debug logging**
5. Нажмите на кнопку **Save**, чтобы сохранить настройки.

Теперь в системный журнал будут записываться отправленные и принятые драйвером пакеты.

Чтобы посмотреть debug-вывод драйвера, выполните в консоли контроллера команду `journalctl -e -p 7`, где `-e` — отобразить последние записи, а `-p 7` задает уровень сообщений, где 7 — это debug. Подробнее о параметрах утилиты, читайте в статье journalctl.

Пример вывода команды:

```
~# journalctl -e -p 7
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [serial port driver] channel 'Urms L1' of device 'wb-modbus-0-0' <--
224.647
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [modbus] read 2 input(s) @ 5136 of device modbus:142
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [port] Sleep 0 us
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [port] Write: 8e 04 14 10 00 02 6a c1
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [port] Sleep 10000 us
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [port] ReadFrame: 16 04 02 0f 3d 09 12
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [register handler] new val for input @ 3 of device modbus:22: f3d
Jan 29 14:27:24 wirenboard-A6ZZXT2R wb-mqtt-serial[1667]: DEBUG: [serial port driver] register value change: input @ 3 of device
modbus:22 <- 39.01
```

# Полезные ссылки

- Как подключить стороннее Modbus-устройство
- Описание wb-mqtt-serial на Github (https://github.com/contactless/wb-mqtt-serial)
- Описание протокола Modbus
- Описание шины RS-485

# 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

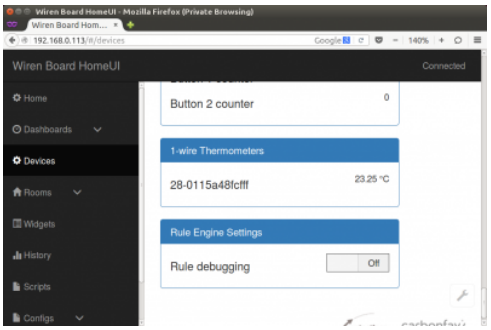
### Получение значения от датчика температуры и вывод его в веб-интерфейс

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

1. Драйвер wb-mqtt-w1 (https://github.com/wirenboard/wb-mqtt-w1), отвечающий за данную аппаратную функцию, опрашивает подключенные к контролеру датчики 1-Wire.
2. При получении значения драйвер публикует MQTT сообщение вида:

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

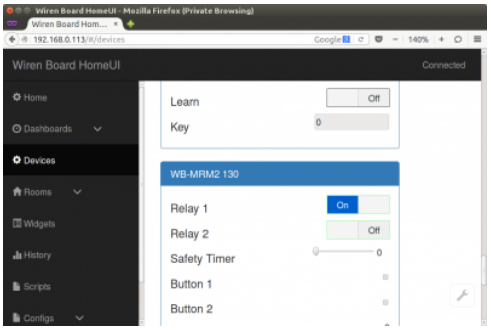
   Оно значит, что от устройства 1-Wire с идентификатором *28-0115a48fcfff* получено значение 23.25 °C.
3. Веб-интерфейс, который подписан на все сообщения из MQTT, получает это сообщение и выводит значение датчика на страницу.



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

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

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:



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

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:

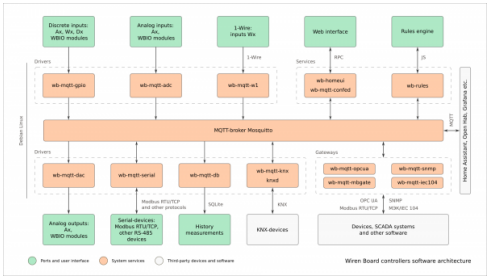
   ```
   /devices/wb-mrm2_130/controls/Relay 1 1
   ```

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

## Принцип работы MQTT

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.



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

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/wirenboard/conventions/blob/main/README.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

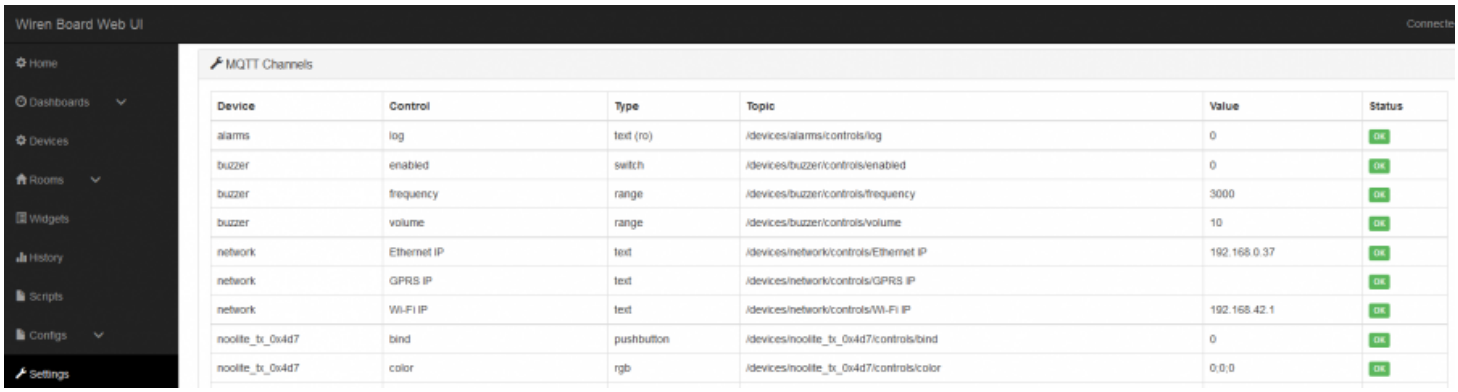
Если вы разрабатываете собственное ПО для Wiren Board, взаимодействовать с его аппаратными ресурсами лучше всего через протокол MQTT — ваша программа передаёт сообщение по MQTT, драйвер управляет устройством и вашей программе не нужно напрямую взаимодействовать с устройством на низком уровне.

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:



Информация об MQTT-названиях устройств

# Работа с сообщениями MQTT с внешнего устройства

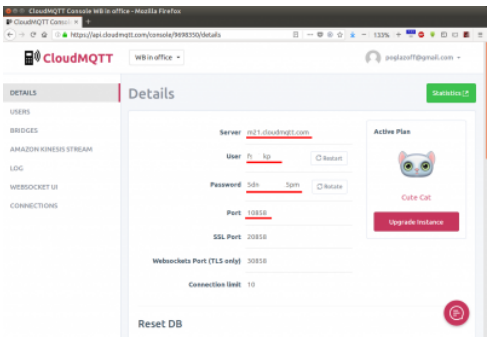
Установленный на контроллер брокер mosquitto по умолчанию принимает подключения внешних клиентов по порту 1883 без пароля.

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.



Настройки брокера Cloud MQTT

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.

### 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:

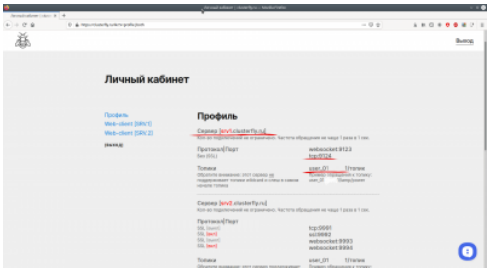
```
connection cloudmqtt
address m21.cloudmqtt.com:10858
remote_username fs_user_kp
remote_password 5dn_pass_pm
clientid pavel_test
try_private false
start_type automatic
topic # both
```

(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 выгляджеть как **/devices**/temp1.
- `/client` — «корень» откуда забираем на удаленном. На контроллере *SourceController* это `/client` и полный путь будет выгляджеть 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:

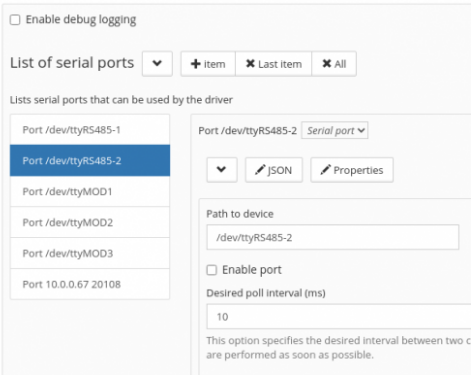
```
connection bridge1
#address of server
address 10.0.0.105
notifications true
notification_topic /clientnotification/bridge1_status
remote_username test
remote_password wbpassword

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.

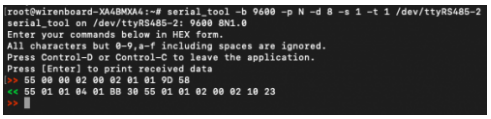
# Работа с последовательным портом (serial-портом)

## Contents

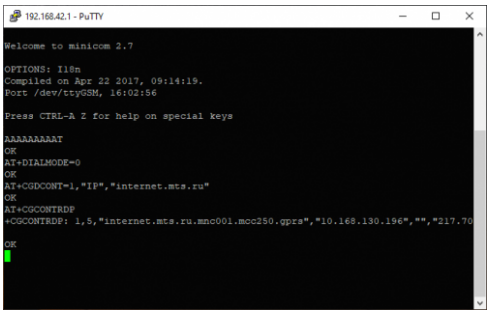




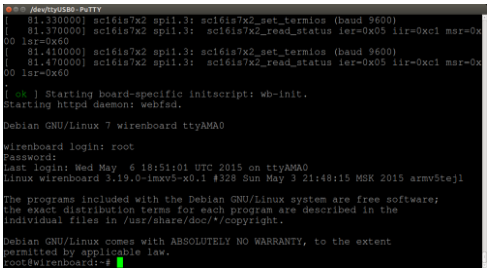

Отключение serial-порта в веб-интерфейсе контроллера Wiren Board




Пример работы с утилитой serial_tool




Отправка AT-команд для модема в терминале программы minicom




Приветственное сообщение контроллера Wiren Board при подключении к его Debug-консоли через PuTTY

## Serial-порты в контроллере Wiren Board

### Реализация

Полное описание читайте в документации (http://www.tldp.org/HOWTO/Serial-HOWTO.html), вот выжимка из неё:

- В serial-порт можно посылать данные и получать данные из него.
- В ОС Linux serial-порты — это псевдофайлы из папки /dev. Например, в контроллерах Wiren Board это могут быть /dev/ttyGSM, /dev/ttyRS485-1, /dev/ttyUSB0 — у всех в названии есть tty.

Перед началом работы с serial-портом, настройте его скорость и параметры так же как настроено подключённое к нему устройство:

- Скорость в битах в секунду, самое популярное — 9600 бит/с.
- Количество бит в символе, чётность и количество стоп-битов. Популярна конфигурация 8N1 — восемь бит в символе, без проверки чётности, один стоп-бит.
- Аппаратный и программный контроль потока — если не уверены, то поставьте «Нет» в обоих параметрах.

### Определение занятости порта и монопольное использование

Обычно, работать с serial-портом может только один процесс. Если порт «занят», то попытка передать или получить через него данные не удастся.

Используйте команду fuser для того, чтобы определить, свободен ли порт. В качестве параметра передайте порт, который нужно проверить.

Проверим, свободен ли порт /dev/ttyRS485-2. Для этого выполните команду:

```
fuser /dev/ttyRS485-1
```

Если вывод пуст — порт свободен. Иначе будет выведен процесс, который занимает порт.

В контроллерах Wiren Board порты /dev/ttyRS485-1 и /dev/ttyRS485-2 обычно заняты драйвером wb-mqtt-serial. Перед работой с этими портами — остановите драйвер одним из способов:

- Остановить драйвер из командной строки.
- В веб-интерфейсе контроллера, в настройках serial-порта снять галочку **Enable port** и сохранить настройки.

## Программы для работы с serial-портом

ОС Linux:

- Serial_tool
- Minicom
- PuTTY

ОС Windows:

- PuTTY

macOS:

- PuTTY for Mac OS X (https://www.ssh.com/ssh/putty/mac/)
- терминальный клиент cu: `cu -s 115200 -l /dev/usbmodem00001`

Android:

- USB Serial Console (https://play.google.com/store/apps/details?id=jp.sugnakys.usbserialconsole&hl=en_US) и другие аналогичные программы.

Прочее:

- Вы не можете использовать программы из этой статьи — посмотрите этот список программ (http://elinux.org/RPi_Serial_Connection).
- Serial-устройство поддерживает протокол Modbus RTU, то вы можете работать с ним с помощью утилиты modbus_client.

- Вы пишете скрипт или свою программу для работы с serial-портом — руководствуйтесь советами из Serial-Programming-HOWTO (https://tldp.org/HOWTO/Serial-Programming-HOWTO/).

# Доступ к порту RS-485 контроллера Wiren Board с компьютера

- English
- русский

socat — утилита, которая может переадресовывать сокеты с хостовой машины, на клиентскую. Будет работать только с протоколом *Modbus over TCP*.

**Внимание!** Данные инструкции следует выполнять только в закрытой подсети. Не следует давать доступ к RS-485 портам по TCP, если Wiren Board доступен по реальному IP снаружи.

Допустим, Wiren Board имеет IP 192.168.3.12.

Выполняем на Wiren Board от root'а:

```
apt-get install socat
service wb-mqtt-serial stop
```

```
socat -d -d -d -x /dev/ttyRS485-1,raw,ispeed=9600,ospeed=9600,parenb=0,cstopb=1,cs8 TCP-LISTEN:10010&
socat -d -d -d -x /dev/ttyRS485-2,raw,ispeed=9600,ospeed=9600,parenb=0,cstopb=1,cs8 TCP-LISTEN:10011&
```

В старых версиях контроллеров указывайте порты [/dev/ttyNSC0, /dev/ttyNSC1] или [/dev/ttyAPP1, /dev/ttyAPP4].

Обратите внимание на кодирование количества стоп-битов:

- cstopb=1 — 2 стоп-бита,
- cstopb=0 — 1 стоп-бит.

Параметр cstopb имеет булевский тип; подробнее смотрите в интернете **man socat**.

Вторая команда (service wb-mqtt-serial stop) — остановка драйвера Modbus во избежание конфликтов при работе с RS-485 портами.

Выполняем на компьютере (под Linux):

```
socat -d -d -d -x PTY,raw,ispeed=9600,ospeed=9600,parenb=0,cstopb=1,cs8,link=/dev/ttyRS485-1 tcp:192.168.3.12:10010&
socat -d -d -d -x PTY,raw,ispeed=9600,ospeed=9600,parenb=0,cstopb=1,cs8,link=/dev/ttyRS485-2 tcp:192.168.3.12:10011&
sudo ln -fs /tmp/ttyRS485-{1,2} /dev
```

В старых версиях контроллеров указывайте порты [/dev/ttyNSC0, /dev/ttyNSC1] или [/dev/ttyAPP1, /dev/ttyAPP4].

После выполнения этих команд в системе (на PC) появляются специальные файлы /dev/ttyRS485-1 и /dev/ttyRS485-2, соответствующие RS-485 портам Wiren Board. Последняя команда для удобства создаёт в /dev символические ссылки на файлы портов, что позволяет, например, использовать на PC конфигурацию драйвера Modbus, написанную для Wiren Board, без каких-либо изменений.

Опции -d и -x в обоих случаях можно опустить - они нужны для вывода диагностики и шестнадцатеричных дампов передаваемых данных.

---