Стандартный образ ФС

- English
- русский

Contents

Настройки по-умолчанию
Ethernet MAC
Wi-Fi
Скачать

Настройки по-умолчанию

Логин: **root**, Пароль: **wirenboard**

Ethernet MAC

MAC-адрес Ethernet устанавливается программно. Адрес указан в файле **/etc/network/interfaces**

Wi-Fi

По-умолчанию создаётся точка доступа **Wiren Board**. Чтобы отключить это поведение, необходимо

- Изменить настройки интерфейса wlan0 беспроводной сети в **/etc/network/interfaces**
- Отключить hostapd - создание программной точки доступа с помощью файла **/etc/default/hostap**

Скачать

https://github.com/contactless/wirenboard/releases См. Создание microSD-карты с образом

wirenboard

Wiren Board 4

Wiren Board 4

Руководство по эксплуатации

Самая актуальная документация всегда доступна на нашем сайте по ссылке:
https://wirenboard.com/wiki/Wiren_Board_4

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

Содержание

Wiren Board 4
Программное обеспечение Wiren Board
Устройства, протоколы и программы, с которыми может работать контроллер Wiren Board
Питание USB-портов
Доступ к порту RS-485 контроллера Wiren Board с компьютера
Подключение периферийных устройств к контроллеру Wiren Board
Уникальные идентификаторы
Debug UART
Создание microSD-карты с образом
Стандартный образ ФС
Работа с GPIO
Работа с последовательным портом (serial-портом)
Zabbix
RS-485
SC16IS752
Wi-Fi
Зуммер (звуковой излучатель)
Device Tree
Сборка ядра Linux

Wiren Board 4

Wiren Board 4 - универсальный контроллер для автоматизации с открытым ПО на базе Linux, ведущая модель линейки Wiren Board.

Предназначен для домашней и промышленной автоматизации и мониторинга: опроса датчиков и счетчиков, использования в качестве УСПД, в системах АСКУЭ, в системах “умного дома”.

Wiren Board 4

Contents

Первое включение
Полное техническое описание
Краткое описание
Поддерживаемое оборудование
Отличия от Wiren Board Smart Home 3.5
Список основных статей
- Hardware
- Software

Первое включение

Если вы впервые запускаете контроллер, прочитайте Wiren Board 4:Первое включение.

Полное техническое описание

- Wiren Board 4:Аппаратная часть
- Программное обеспечение Wiren Board

Краткое описание

- Операционная система: Debian Linux 7.0. Собственный веб-интерфейс.
- Процессор и память: 454 МГц ARM9, 64 МБ RAM, 8 ГБ microSD
- Беспроводная связь: Wi-Fi, GSM/GPRS, 433 МГц
- Порты: Ethernet 10/100, USB 2.0, 2xRS-485, 1-wire, опции: RS-232, CAN
- Входы/выходы:
  - 5 входов АЦП до 30 В
  - 2 входа для резистивных датчиков
  - 4 входа для датчиков “сухой контакт”
  - 9 выходов “открытый коллектор”
  - 2 реле 250В/2А
- Корпус: 103x87x20 мм, на DIN-рейку
- Условия эксплуатации: 0..70C (-40..85C по запросу)
- Watchdog, динамик, часы реального времени RTC, влагозащитное покрытие, разъём для подключения Li-Ion аккумулятора

Поддерживаемое оборудование

Категории поддерживаемых устройств:

- водо- и энергосчётчики с импульсными выходами и RS-485
- устройства управления подачей питания (релейные модули) с интерфейсом RS-485
- цифровые и аналоговые датчики, в том числе удалённые по RS-485. Измеряемые параметры: температура, влажность, освещённость, наличие газов, давление
- устройства управления освещением: релейные модули, диммеры, контроллеры светодиодных лент, работающие по RS-485, радио 433 Мгц

Список протестированного оборудования: Поддерживаемые устройства. Если у вас есть вопросы по выбору/подключению устройств из списка, задавайте их на форуме (http://wirenboard.com/forums/).

Отличия от Wiren Board Smart Home 3.5

Добавлены:

- оптоизолированный порт RS-485
- входы для сигналов “сухой контакт” с защитой от помех
- возможность установки модулей расширения
- независимый сторожевой таймер (watchdog)
- влагозащитное покрытие
- защита зарядки аккумулятора при отрицательной температуре
- самозажимные клеммники
- диапазон питания расширен до 9-24В

Список основных статей

Hardware

Wiren Board 4:Аппаратная часть

Файл:WB 4.3.pdf

Файл:WB 4.2 чертеж.pdf

Аппаратные ошибки/особенности Wiren Board 4 rev. 4.3

Питание USB-портов

Доступ к RS-485 портам с PC

Управление низковольтной нагрузной

Уникальные идентификаторы - для идентификации устройства, привязки софта

Software

Программное обеспечение Wiren Board

Debug UART

Создание microSD-карты с образом

Стандартный образ ФС

Работа с GPIO

Wiren Board 4:Список GPIO

Работа с последовательным портом из Linux

Zabbix support

Подсистемы:

- ADC#Software
- Управление низковольтной нагрузкой#Пример работы в Linux
- GSM/GPRS
- RS-485 через SC16IS752
- Wi-Fi
- Buzzer

Пересборка Device Tree

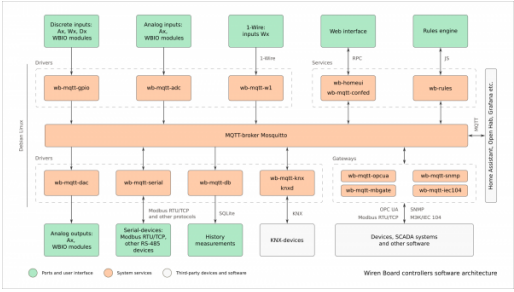
Сборка ядра

Примеры от Olimex (https://github.com/OLIMEX/OLINUXINO/tree/master/SOFTWARE/iMX233)

Программное обеспечение Wiren Board

Архитектура ПО Wiren Board

Wiren Board работает под управлением стандартной сборки Debian Linux 9 Stretch. Для архитектуры используемого процессора есть официальный порт (https://www.debian.org/ports/arm/). Поэтому почти любой пакет найдётся в стандартном репозитории, и его можно установить одной командой `apt-get install имя_пакета`.



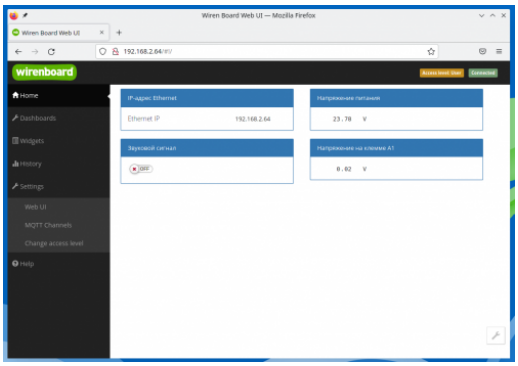
Структура ПО контроллера. В центре очередь сообщений MQTT, которая используется для обмена информацией между разными частями ПО

Есть две ветки ПО Wiren Board: **stable** и **testing**.

Исходный код программного обеспечения доступен на GitHub (https://github.com/contactless/). Там можно почерпнуть примеры для разработки собственного ПО.

Очередь сообщений MQTT — «скелет» программной архитектуры Wiren Board.

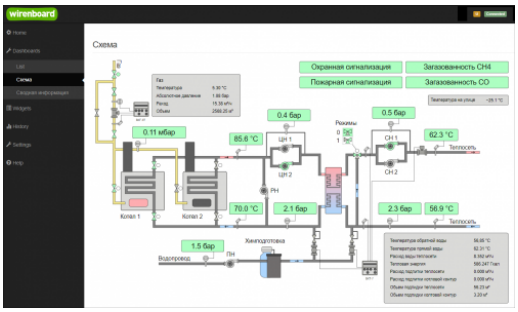
Веб-интерфейс Wiren Board работает непосредственно на контроллере. В нём можно:



Главная страница веб-интерфейса

- следить за состоянием контроллера и подключённых устройств и управлять ими,
- подключать устройства к контроллеру,
- настраивать контроллер и обновлять его ПО,
- писать правила на встроенном движке,
- настраивать SMS- и email-уведомления,
- смотреть графики истории значений параметров: температуры, напряжения и т.п.

Движок правил wb-rules позволяет создавать собственные правила для контроллера, например: «Если температура датчика меньше 18°C, включи нагреватель». Правила создаются через веб-интерфейс и пишутся на простом Javascript-подобном языке.



Пример графического SVG-дашборда

Для работы с SCADA-системами есть:

- Агент Zabbix
- Шлюз Modbus TCP/RTU
- Шлюз OPC UA
- Шлюз МЭК 104
- Агент SNMP

Дополнительно:

- Node-RED — инструмент визуального программирования.
- Home Assistant — open-source платформа для автоматизации.
- Docker — программное обеспечение для запуска приложений в изолированной среде.

Полезные ссылки

- Обновление прошивки контроллера
- Как разрабатывать ПО для Wiren Board — статья для программистов.
- Обновление прошивок в Modbus-устройствах Wiren Board

Список сервисов и их назначение

Список сервисов, запущенных на контроллере, их статус и описание можно получить командой:

```
systemctl list-units --type=service
```

Про управление сервисами читайте в статье Диагностика ошибок в работе контроллера.

Имя сервиса	**Описание**
avahi-daemon.service	Avahi mDNS/DNS-SD Stack
bluetooth.service	Bluetooth service

cgmanager.service	Cgroup management daemon
cron.service	Regular background program processing daemon
dbus.service	D-Bus System Message Bus
dnsmasq.service	dnsmasq - A lightweight DHCP and caching DNS server
getty@tty1.service	Getty on tty1
hostapd.service	LSB: Advanced IEEE 802.11 management daemon
kmod-static-nodes.service	Create list of required static device nodes for the current kernel
knxd.service	KNX Daemon
mosquitto.service	Mosquitto MQTT v3.1/v3.1.1 Broker
netplug.service	LSB: Brings up/down network automatically
networking.service	Raise network interfaces
nginx.service	A high performance web server and a reverse proxy server
ntp.service	LSB: Start NTP daemon
rsyslog.service	System Logging Service
serial-getty@ttymxc0.service	Serial Getty on ttymxc0
ssh.service	OpenBSD Secure Shell server
systemd-fsck-root.service	File System Check on Root Device
systemd-fsck@dev-mmcblk0p6.service	File System Check on /dev/mmcblk0p6
systemd-journal-flush.service	Flush Journal to Persistent Storage
systemd-journald.service	Journal Service
systemd-logind.service	Login Service
systemd-modules-load.service	Load Kernel Modules
systemd-random-seed.service	Load/Save Random Seed
systemd-modules-load.service	Load Kernel Modules
systemd-random-seed.service	Load/Save Random Seed
systemd-remount-fs.service	Remount Root and Kernel File Systems
systemd-sysctl.service	Apply Kernel Variables
systemd-tmpfiles-setup-dev.service	Create Static Device Nodes in /dev
systemd-tmpfiles-setup.service	Create Volatile Files and Directories
systemd-udev-trigger.service	udev Coldplug all Devices
systemd-udevd.service	udev Kernel Device Manager
systemd-update-utmp.service	Update UTMP about System Boot/Shutdown
systemd-user-sessions.service	Permit User Sessions
user@0.service	User Manager for UID 0
watchdog.service	watchdog daemon
wb-configs-early.service	prepare mounts and symlinks to config files
wb-configs.service	watch config files
wb-gsm-rtc.service	LSB: initscript to use GSM modem integrated RTC
wb-homa-ism-radio.service	LSB: MQTT driver for WB HomA for RFM69 ISM radio
wb-hwconf-manager.service	LSB: Hardware configuration with Device Tree overlays
wb-init.service	LSB: board-specific initscript
wb-mqtt-adc.service	MQTT Driver for ADC
wb-mqtt-confed.service	LSB: Configuration Editor Backend
wb-mqtt-db.service	Wiren Board database logger
wb-mqtt-gpio.service	MQTT Driver for GPIO-controlled switches

wb-mqtt-knx.service	LSB: : Wiren Board MQTT KNX bridge
wb-mqtt-logs.service	Wiren Board journald to MQTT gateway
wb-mqtt-mbgate.service	Wiren Board MQTT to Modbus TCP gateway
wb-mqtt-opcua.service	Wiren Board MQTT to OPC UA gateway
wb-mqtt-serial.service	MQTT Driver for serial devices
wb-mqtt-w1.service	Kernel 1-Wire MQTT driver for WB-HomA
wb-prepare.service	initialize filesystems at first boot
wb-repart.service	prepare partitions at first boot
wb-rules.service	MQTT Rule engine for Wiren Board
wb-systime-adjust.service	Compensation of systime in PPM from value, stored in device-tree (with opposite sign)
wb-watch-update.service	LSB: Firmware update monitor

Устройства, протоколы и программы, с которыми может работать контроллер

которыми может работать контроллер Wiren Board

Поддерживаемые контроллером Wiren Board протоколы, устройства и системы верхнего уровня	
Поддерживаемые протоколы	
Опрос датчиков и работа	1-Wire • DLMS/COSEM • Modbus RTU/TCP Master • ГОСТ МЭК 61107 • СПОДЭС

с устройствами (в базовой комплектации)	1-Wire • DLMS/COSEM • Modbus RTU/TCP Master • PCO MSK 01107 • SNMPv3 (ГОСТ Р 58940-2020)
Опрос датчиков и работа с устройствами (с помощью модулей расширения)	KNX • eBUS • OpenTherm • Z-Wave • Zigbee
Системы верхнего уровня	KNX • Modbus RTU/TCP Slave • MQTT • OPC UA • SNMP • Zabbix • МЭК 104 • SmartWeb
ПО верхнего уровня	Grafana • MasterSCADA • Nagios • Rapid SCADA • SAYMON • Zabbix • IntraSCADA • IntraHouse • IRidium Server
Протестированные устройства сторонних производителей	
Датчики климата	DS18B20 и клоны • Kvadro 1WIRE-RS485 • RLDA NL-3DPAS-M • RLDA NL-1S111 • Wellpro WP3066ADAM • РД MSU21 • РД MSU24 • РД MSU34+TLP • РД MSU34+THLP • Эксис ИВТМ-7 М 3
Датчики уровня	ЭСКОРТ ДБ-2
Диммеры	Шлюз DALI GW2 • Philio PAD07-RU • Uniel UCH-M131RC/0808 • Uniel UCH-M141RC/0808 • РД DDL04R • РД DDL24 • РД DDL84R-V • РД DDM845R
Конвекторы	Varmann QTherm
Кондиционеры	Haier YCJ-A002 • Z-Wave ИК-передатчик PAR01-RU
Контроллеры вентиляции и климата	Mautomatics JL204C5 (Breezart 550 Lux) • GTC (General Thermo Controllers) Syberia 5.0 • SystemAir VR 300
Контроллеры холодильного оборудования	Carel BASIC(PYEZ)/EASY(PJEZ) • Danfoss EKC 204A1 / AK-CC 210 • Danfoss EKC 202B • Danfoss EKC 202D • Danfoss ERC 211/ERC 213/ERC 214 • Eliwell IDPlus 974
Метеостанции	Netatmo Urban Weather Station
Модули ввода-вывода	Wellpro WP8026ADAM • Wellpro WP8027ADAM • Wellpro WP8028ADAM • Wellpro WP9038ADAM
Модули реле	РД DRB88 • Rubetek TZ78 • ICP DAS tM-P3R3 • ICP DAS LC-103 • Uniel UCH-M111RX/0808 • Uniel UCH-M121RX/0808
Моторы для штор/Электрокарнизы	Akko AM82 • Dooya DT82 • WinDeco • Somfy SDN • SunFlower KT82TV • Somfy RS485 RTS transmitter
Преобразователи частоты	Vacon/Danfoss 10 • Danfoss VLT Microdrive FC51 • T13-400W-12-H
Счётчики воды	Пульсар • Пульсар-М • Элехант СВД-15 • Элехант СВД-20 • Счётчики с импульсным выходом
Счётчики тепла	Пульсар
Счётчики электроэнергии	CSQ PD561Z-9SY • Peacefair PZEM-016 • Eastron SDM120M • Eastron SDM220M • Меркурий 200 • Меркурий 201 • Меркурий 203.2T • Меркурий 204 • Меркурий 206 • Меркурий 208 • Меркурий 230 • Меркурий 231 • Меркурий 234 • Меркурий 236 • Меркурий 238 • Милур 104 • Милур 105 • Милур 107 • Милур 305 • Милур 307 • Нева МТ 113 • Нева МТ 123 • Нева МТ 124 • Нева МТ 323 • Нева МТ 324 • Энергомера CE301 • Энергомера CE102M • Энергомера CE303 • Энергомера CE308
Термостаты	BAC-6000 Series • BHT-6000 Series • Cityron ПУ-3 (Modbus) • Heatit Z-TEMP2 • Hessway • Siemens RDF302
Увлажнители	CAREL Humisonic
Прочее	DIY • Shelly UNI • Tasmota • ESPHome
Устройства с аналоговым или цифровым выходом	
Низковольтная нагрузка	Реле с управляющим напряжением 12–24 В • Светодиоды • Низковольтные вентиляторы • Низковольтные сигнальные лампы
Датчики с аналоговым выходом	Датчики температуры, давления и другие, имеющие на выходе ток или напряжение
Счётчики с импульсным выходом	Счётчики электроэнергии, воды, тепла и другие с импульсным выходом
Устройства с выходом «открытый коллектор»	Устройства с выходом «открытый коллектор»
Устройства с питанием 220 В	Лампы • Контакторы • Другое оборудование, питающееся от 220 В

Питание USB-портов

- English
- русский

пакет hubpower из нашего репозитория.

Работа:

```
root@wirenboard:~# hubpower 1:2 status
Port  1 status: 0503  High-Speed Power-On Enabled Connected
Port  2 status: 0100  Power-On
Port  3 status: 0100  Power-On
Port  4 status: 0100  Power-On
Port  5 status: 0503  High-Speed Power-On Enabled Connected
```

Включение/выключение порта:

```
root@wirenboard:~# hubpower 1:2 power 4 off
Port  4 status: 0000  Power-Off
root@wirenboard:~# hubpower 1:2 power 4 on
Port  4 status: 0100  Power-On
```

Contents

Wiren Board 6

Wiren Board 5

Wiren Board 4

Wiren Board Smart Home rev. 3.5

Wiren Board 6

Второй внешний USB-порт:

```
# выключить
$ hubpower `lsusb  | grep "0424:2514" | sed 's/^Bus 0*\([[:digit:]]*\) Device 0*\([[:digit:]]*\).*$/\1:\2/g'` power 4 off
# включить
$ hubpower `lsusb  | grep "0424:2514" | sed 's/^Bus 0*\([[:digit:]]*\) Device 0*\([[:digit:]]*\).*$/\1:\2/g'` power 4 on
```

Модуль Wi-Fi:

```
# выключить
$ hubpower `lsusb  | grep "0424:2514" | sed 's/^Bus 0*\([[:digit:]]*\) Device 0*\([[:digit:]]*\).*$/\1:\2/g'` power 1 off
# включить
$ hubpower `lsusb  | grep "0424:2514" | sed 's/^Bus 0*\([[:digit:]]*\) Device 0*\([[:digit:]]*\).*$/\1:\2/g'` power 1 on
```

Вместо

```
$ hubpower `lsusb  | grep "0424:2514" | sed 's/^Bus 0*\([[:digit:]]*\) Device 0*\([[:digit:]]*\).*$/\1:\2/g'` ...
```

в большинстве случаев можно писать

```
$ hubpower 2:2 ...
```

Wiren Board 5

Внешний USB-порт:

```
# выключить
$ hubpower 1:1 power 1 off
# включить
```

```
# включить
$ hubpower 1:1 power 1 on
```

Модуль Wi-Fi:

```
# выключить
$ hubpower 2:1 power 1 off
# включить
$ hubpower 2:1 power 1 on
```

Wiren Board 4

Номера портов (для управления питанием):

```
4 - встроенный WiFi
3 - порты USB-Hub, выход 5V
1 - EThernet-часть LAN9514
```

Пример (отключение Wi-Fi):

```
hubpower 1:2 power 4 off
```

Wiren Board Smart Home rev. 3.5

Номера портов (для управления питанием):

```
4 - встроенный WiFi
3 - порты USB-Hub
1 - EThernet-часть LAN9514
```

Доступ к порту RS-485 контроллера Wiren Board с компьютера

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'a:

```
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.

Подключение периферийных устройств к контроллеру Wiren Board

контроллеру Wiren Board

Contents

Управление низковольтной нагрузкой
- Технические подробности
- Подключение нагрузки

Датчики с аналоговым выходом по напряжению

Датчики с аналоговым токовым выходом

Датчики/счетчики с импульсными выходами/кнопки

Устройства с выходом открытый коллектор

Контакторы с управляющим напряжением 220В

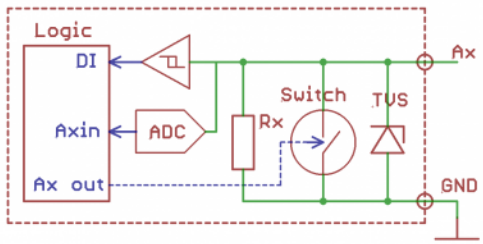


Схема входов/выходов A1-A4

Управление низковольтной нагрузкой

Технические подробности

Для управления низковольтной нагрузкой в Wiren Board предназначены так называемые «транзисторные выходы», они же FET или **открытый коллектор**. С их помощью можно управлять включением низковольтных ламп, светодиодных лент, внешних блоков реле и т.п.

Транзисторными выходами можно управлять из веб-интерфейса, там они называются соответственно клеммам: **A1_OUT** — **A4_OUT**.

Для сокращения общего числа клеммников каналы управления низковольтной нагрузки совмещены с каналами АЦП. Поэтому выходы имеют большое, но конечное сопротивление — 100кОм. Это может вызывать, например, слабое свечение светодиодных лент, но проблему можно решить — подтянуть вывод резистором к питанию.

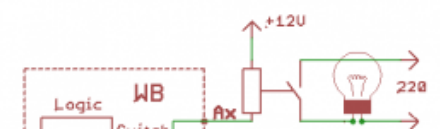
В контроллерах Wiren Board 6 выходы защищены от импульсных перенапряжений, короткого замыкания и перегрева.

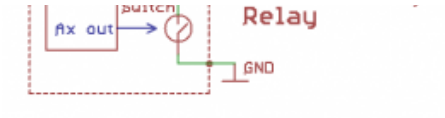
Подключение нагрузки

Чтобы подключить нагрузку, подключите «плюс» нагрузки к источнику питания, а «минус» к транзисторному входу. Нагрузка включается подачей высокого уровня на выход. Если суммарный ток на канале превышает 2 А — дополнительно подключите клемму **GND** к минусу источника питания.

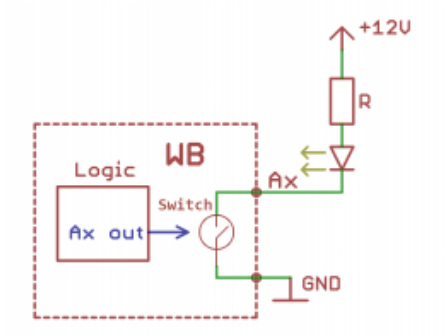
При управлении индуктивной нагрузкой (реле), возникают всплески напряжения. Для защиты от перенапряжения в контроллер встроены защитные диоды — внешних защитных элементов не требуется.

Также для управления низковольтной нагрузкой можно использовать модуль дискретных выходов WBIO-DO-HS-8.





Пример подключения реле к выходам A1-A4

- 

Пример подключения светодиода к выходам A1-A4

Датчики с аналоговым выходом по напряжению

Клеммы A1-A4 могут измерять напряжение, поэтому к ним можно подключить датчики с аналоговым выходом по напряжению, например, температурные сенсоры.

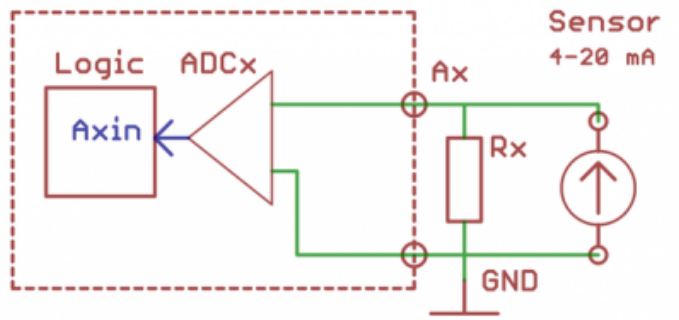
Подключите землю устройства к клемме GND, или соедините с общей земляной шиной. Выход датчика подключите к одной из клемм **Ax**.

Для точного измерения напряжения можно использовать модуль ввода-вывода WBIO-AI-DV-12_I/O_Module или модуль аналоговых входов WB-MAI11.

Датчики с аналоговым токовым выходом

Специальных токовых входов в WB нет, но можно, используя резистор Rx= 100-300Ом, ток преобразовать в напряжение и подключить по аналогии с датчиком, имеющим аналоговый выход по напряжению.

Так же можно использовать модуль ввода-вывода WBIO-AI-DV-12_4-20MA или модуль аналоговых входов WB-MAI11.



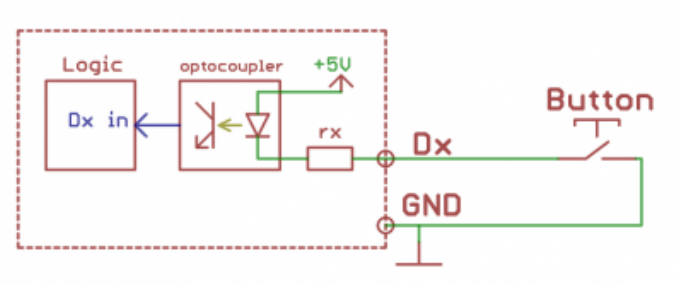
Датчики/счетчики с импульсными выходами/кнопки

Такие устройства формируют сигнал, замыкая подходящие к ним два провода.

Способы подключения к контроллеру:

1. С помощью клемм **Ax** контроллера. Подключите один провод к источнику питания 5-24 В, второй провод к клемме **Ax**. Подробнее смотрите на странице Подключение устройств с импульсными выходами.
2. С помощью модуля ввода-вывода WBIO-DI-WD-14 (14 каналов). Один из проводов подключите к GND, второй к клемме **Dx** модуля.
3. С помощью модуля расширения WBE2-DI-DR-3 (3 канала). Один из проводов подключите к GND, второй к клемме **Ox** соответствующей модулю расширения.

Некоторые счетчики имеют импульсный выход на оптроне, тогда два провода имеют полярность — «плюс» и «минус». В таком случае минус подключается к **GND**, а «плюс» ко входу. Либо для первого способа — «плюс» к питанию, а «минус» к **Ax**.



Устройства с выходом открытый коллектор

Есть три способа подключить такие устройства к контроллеру:

1. С помощью модуля ввода-вывода WBIO-DI-WD-14 (14 каналов). Выход «открытый коллектор» подключите к клемме **Dx** модуля. Землю устройства к iGND модуля.
2. С помощью модуля расширения WBE2-DI-DR-3 (3 канала). Выход «открытый коллектор» подключите к клемме **Ox** соответствующей модулю расширения, а землю устройства к GND контроллера.
3. Можно подключать к клеммам **A1-A4**, при этом нужно также подключить внешний подтягивающий резистор между **5Vout** и соответствующей клеммой **Ax** номиналом около 10 кОм. Соедините земли устройства и контроллера.

Контакторы с управляющим напряжением 220В

Используйте модуль ввода-вывода с релейными выходами, например WBIO-DO-R1G-16.

Подключите управляющую катушку контактора через реле модуля расширения, схему подключения смотрите в разделе «Монтаж» на странице используемого модуля.

Модуль WBIO-DO-R1G-16 содержит TVS, защищающий контакты реле от искрения. Внешние защитные компоненты не требуются.

Уникальные идентификаторы

- English
- русский

Contents

Серийный номер (MAC) Wiren Board
Уникальный идентификатор процессора
Уникальный идентификатор чипа памяти eMMC
IMEI GSM-модуля
MAC-адрес Wi-Fi модуля

Серийный номер (MAC) Wiren Board

Серийный номер Wiren Board хранится в файле

```
/var/lib/wirenboard/serial.conf
```

он генерируется при первом старте из MAC Wi-Fi, если его нет, то из IMEI GSM-модема, либо из уникального идентификатора процессора.

Этот же идентификатор используется в качестве MAC-адреса интерфейса Ethernet

Уникальный идентификатор процессора

Доступен в файле /proc/cpuinfo в графе Serial.

Уникальный идентификатор чипа памяти eMMC

только для Wiren Board 5,6,7

```
cat /sys/class/mmc_host/mmc0/mmc*/serial
```

IMEI GSM-модуля

```
wb-gsm imei
```

Внимание: не рекомендуется допускать попадание IMEI в распоряжение посторнним лицам

MAC-адрес Wi-Fi модуля

```
cat /sys/class/net/wlan*/address
```

Debug UART

Возможно, для специалистов это будет излишним, но я думаю,что можно привести ссылки на документацию по началу работы с Debug UART и описать, зачем вообще это нужно. Быстрый поиск в интернете меня не удовлетворил.

Создание microSD-карты с образом

- English
- русский

Contents

Запись готового образа на карточку
- Выбор нужного образа
- Для Windows
- Для Linux

Создание образа по частям
- Сборка вместе
- Таблица разделов
- Загрузчик
- Создание фс
- Копирование образа на раздел

Пример

Запись готового образа на карточку

Выбор нужного образа

- Зайдите на страницу готовых образов в Github и выберите нужный образ:
  - для Wiren Board 4 - название оканчивается на *_wb4*
  - для Wiren Board Smart Home 3.5 - название оканчивается на *_wb3*
    - если серийный номер вашего Wiren Board Smart Home 3.5 больше 300 - используйте образ с *newwifi* в названии
  - для WB rev. 2.8 - название оканчивается на *_wb28*

    У образа будет расширение .dd, либо .dd.gz, либо img.zip
- Распакуйте архив
- Следуйте инструкции для вашей операционной системы

Для Windows

- скачайте программу для записи образов(например, Win32DiskImager)
- вставьте microSD-карту в ридер
- узнайте букву, под которой она появилась (например "F:")
- проигнорируйте сообщения о необходимости отформатировать диск перед использованием, если такое появится
- убедитесь, что другие программы не используют флеш-карту
- в Win32DiskImager выберите распакованный образ карты, выберите букву диска и нажмите кнопку *Write*

Для Linux

- вставьте microSD-карту в ридер
- узнайте название устройства, соответствующего карте. Обычно это /dev/mmcblk0 или /dev/sdX (где X - буква). В этом может помочь команда

```
dmesg | tail
```

Не перепутайте название устройства! Неправильно указав название устройства, вы навсегда потеряете все данные на вашем компьютере!

- отмонтируйте разделы карты, которые Linux примонтировал автоматически:
  - если устройство называется /dev/mmcblk0, то разделы называются /dev/mmcblk0p1, /dev/mmcblk0p2 и т.д.
  - если устройство называется /dev/sdb, то разделы называются /dev/sdb1, /dev/sdb2 и т.д.

  Пример команды:

  ```
  umount /dev/mmcblk0p1
  ```

- запишите образ на карту:

```
sudo dd if=sdcard.dd of=/dev/mmcblk0 bs=4M
```

, где "sdcard.dd" - путь к ранее скачанному распакованному файлу с образом.

Пример процесса целиком:

```
wget https://github.com/contactless/wirenboard/releases/download/0.6-20140614/sdcard_20140614.img.zip
unzip sdcard_20140614.img.zip
umount /dev/mmcblk0p2
umount /dev/mmcblk0p1
sudo dd if=sdcard_20140614.img of=/dev/mmcblk0 bs=4M conv=fdatasync
sync
```

Создание образа по частям

Внимание! Это сложный вариант самостоятельной подготовки образа карточки. Лучше воспользуйтесь вариантом, описанным выше.

Внимание! На 6 ноября 2015 инструкция ниже ещё и безнадёжно устарела.

Сборка вместе

Согласно [1]

- разбить флешку на два раздела
- записать u-boot на первый раздел
- создать фс на втором разделе
- скопировать rootfs на второй раздел

Сначала надо выяснить имя устройства с флеш-картой. Воспользуйтесь поиском. Можно, к примеру, попробовать запустить gparted и посмотреть в нём. Устройство может быть /dev/sdb, а может выглядеть и как /dev/mmcblk0

Найдя устройство создаем переменную чтобы облегчить себе использование нижеследующих комманд

```
MYDISK="/dev/sdb"
```

Между кавычками пишем название своего устройства с флеш-картой.

Таблица разделов

Перед запуском убедитесь, что разделы на карточке не примонтированы.

Теперь с помощью скрипта создадим разделы на карте. [2]

Скачиваем скрипт:

```
wget [https://raw.github.com/contactless/wirenboard/master/image/create_partitions.sh]
```

Запускаем скрипт и указываем ему имя устройства с флеш-картой:

```
sudo bash create_partitions.sh $MYDISK
```

Загрузчик

См. Сборка U-Boot.

Готовый образ u-boot для записи в раздел: u-boot

Скачиваем образ

```
wget "https://github.com/contactless/wirenboard/blob/master/contrib/u-boot/u-boot.sb.cl25?raw=true" -O u-boot.sb
```

Теперь выясняем имена разделов на карте. Скрипт выше отработал и разделил карту на разделы, нам надо узнать название первого из этих разделов. Если название устройства microSD-карты имело вид **/dev/sdX**, то первый раздел будет иметь название **/dev/sdX1**. Если устройство называлось **/dev/mmcblkX** , то первый раздел - **/dev/mmcblkXp1** (обратите внимание на **p** перед номером раздела).

Найдя имя первого раздела укажите его тут:

```
MYDISK1="/dev/sdb1"
```

Теперь заливаем образ раздела на карту

```
sudo dd if=u-boot.sb of=$MYDISK1 bs=512 seek=4
```

Создание фс

Находим название второго раздела на флешке и прописываем его в переменную

```
MYDISK2="/dev/sdb2"
```

rootfs станет названием этого раздела.

```
sudo mkfs.ext4 $MYDISK2 -L rootfs
```

Копирование образа на раздел

Готовый образ (включая ядро, dtbs, модули и прошивки): releases

См. также Сборка образа

```
wget https://github.com/contactless/wirenboard/releases/download/0.1/rootfs.tar.gz
```

Теперь надо примонтировать созданную файловую систему созданную нами ранее. *Как это сделать в терминале?*

По-умолчанию в Ubuntu она монтируется в **/media/$USER/rootfs/**. Найдите куда система

смонтировалась на вашем компьютере.

Распаковываем образ на раздел:

```
 sudo tar xfpz rootfs.tar.gz -C /media/$USER/rootfs/
```

Отмонтируем файловую систему:

```
umount /media/user/rootfs
```

Пример

ОС Ubuntu, свежая SD-карта подключенная к встроенному считывателю и определяющаяся как /dev/mmcblk0. Репозиторий скачан, мы находимся в его корне.

Образ rootfs.tar.gz находится внутри папки rootfs.

```
cd image
sudo umount /dev/mmcblk0p1
sudo ./create_partitions.sh  /dev/mmcblk0
sudo dd if=../contrib/u-boot/u-boot.sb  of=/dev/mmcblk0p1 bs=512 seek=4
sudo ./create_fs.sh /dev/mmcblk0p2

# Ubuntu automount:
udisksctl mount -b /dev/mmcblk0p2

#extract rootfs
sudo tar xfpz ../rootfs/rootfs.tar.gz -C /media/$USER/rootfs/

umount /dev/mmcblk0p2
```

См. также Стандартный образ ФС.

Работа с GPIO

ВНИМАНИЕ: статья рассчитана на разработчиков или опытных пользователей и даёт общие рекомендации того, как использовать gpio в обход официального ПО WirenBoard.

Если вам нужно работать напрямую с gpio, то мы рекомендуем делать это через драйвер wb-mqtt-gpio (https://github.com/wirenboard/wb-homa-gpio).

Описание доступных ножек gpio для конкретной ревизии контроллера можете посмотреть в статье GPIO.

Contents

Меры предосторожности

Именование gpio

- Вычисление номера gpio

Работа из userspace

- Bash
  - Интерфейс sysfs
    - Чтение и запись
    - Работа с прерываниями
  - Работа через chardev
- Python
- Прямое обращение через память процессора

Работа в ядре Linux

- Рекомендации по Device Tree
  - Пример device-tree node

Меры предосторожности

- **Убедитесь, что вашу задачу нельзя решить стандартными средствами программного обеспечения Wiren Board.**
- **Все порты Wiren Board, в том числе и GPIO, работают с напряжением 3.3V.**
- **Подключение сигнала с напряжением большим 3.3V к ножке GPIO грозит выходом из строя процессорного модуля.**

В случае необходимости подключения устройств, работающих с более высоким напряжением, необходимо использовать схемы согласования или подключать (для 5V) через резистор в 20 кОм и более.

Именование gpio

К сожалению, четкого стандарта по именованию gpio не существует, но при работе с контроллерами WirenBoard стоит придерживаться следующих правил:

- выводы gpio сгруппированы по банкам (*banks*; эквивалентно *gpiochips*)
- каждый банк содержит 32 gpio. Нумерация банков начинается с 0.

Вычисление номера gpio

Для управления ножкой gpio нужно знать её номер. В рассматриваемых примерах будем работать с gpio *A1_IN* контроллера WB6.7 (*номер: 109; gpiochip 3, offset 13*): Вычислим банк gpio и offset, зная номер (109):

```
# Поделим 109 на 32. Целая часть — номер банка, остаток - offset:
109.0 / 32.0 = 3, остаток 13
```

То же самое справедливо и наоборот. Зная банк и offset (3 и 13, соответственно), можно вычислить номер gpio:

```
# Умножим номер банка на 32 и прибавим offset:
3 * 32 + 13 = 109
```

Работа из userspace

Перед началом работы из userspace, необходимо убедиться, в том, что нужный gpio — свободен. Для этого можно посмотреть на вывод команды

```
cat /sys/kernel/debug/gpio
```

В выводе команды видим примерно следующее:

```
gpiochip0: GPIOs 0-31, parent: platform/209c000.gpio, 209c000.gpio:
 gpio-0   (                    |sysfs               ) in  hi IRQ
 gpio-10  (                    |?                   ) in  lo
 gpio-11  (                    |w1                  ) in  hi
 gpio-13  (                    |w1 strong pullup    ) out lo
 gpio-26  (                    |sysfs               ) out lo
 gpio-27  (                    |sysfs               ) out hi
```

Это значит, что gpio 0, 26 и 27 уже экспортированы в sysfs и доступны для управления. Gpio 11 и 13 заняты ядерным драйвером onewire и недоступны для использования. Остальные gpio банка 0 — свободны.

Если нужный gpio — занят, то можно остановить драйвер:

```
lsmod | grep w1 # узнаем название драйвера
rmmod w1_gpio # выгружаем драйвер, название которого узнали
```

ВНИМАНИЕ: остановка драйверов может привести к неожиданному поведению контроллера. Теперь нужный gpio свободен до следующей перезагрузки.

Bash

В настоящий момент, для работы с gpio в userspace доступны 2 интерфейса: *sysfs* и *chardev* (начиная с версии ядра 4.8).

Различия между *chardev* и *sysfs* хорошо описаны в этой статье (https://embeddedbits.org/new-linux-kernel-gpio-user-space-interface/). Sysfs имеет статус deprecated, поэтому, по возможности, стоит работать через chardev.

Интерфейс sysfs

Для работы через sysfs с определённым GPIO его надо экспортировать:

Здесь и далее N — номер gpio

```
echo N > /sys/class/gpio/export
```

Экспортированные gpio появляются в каталоге `/sys/class/gpio`:

```
~# ls -1 /sys/class/gpio/
export
gpio32
gpiochip0
gpiochip120
gpiochip32
gpiochip64
unexport
```

В директории `/sys/class/gpioN` теперь находятся файлы для работы с GPIO (где N — номер GPIO, как и было сказано ранее):

```
~# ls -1 /sys/class/gpio/gpioN/
active_low
device
direction
edge
power
subsystem
uevent
value
```

Установка направления GPIO (ввод/вывод) производится с помощью записи в файл `direction`

```
echo in > /sys/class/gpio/gpioN/direction # установим GPIO номер N на ввод
echo out > /sys/class/gpio/gpioN/direction # установим GPIO номер N на вывод
```

Чтение и установка значения GPIO производится с помощью файла `value`.

Чтение и запись

Чтение:

```
echo in > /sys/class/gpio/gpioN/direction # установим GPIO номер N на ввод
cat /sys/class/gpio/gpioN/value # вернёт 1 или 0
```

Запись:

```
echo out > /sys/class/gpio/gpioN/direction # установим GPIO номер N на вывод
echo 0 > /sys/class/gpio/gpioN/value # установим логический 0  (низкое напряжение) на GPIO номер N
echo 1 > /sys/class/gpio/gpioN/value # установим логический 1  (высокое напряжение) на GPIO номер N
```

Пример:

1. Находим номер GPIO, соответсвующий вашей версии контролера нужному клеммнику в таблице WB2.8. Для клеммника номер 2 в версии 2.8 это GPIO 32.
2. Экспортируем GPIO в sysfs

   ```
   echo 32 > /sys/class/gpio/export
   ```

3. Устанавливаем GPIO в режим вывода для управления транзистором. Это обязательно, т.к. GPIO может находится в режиме ввода и иметь высокий импенданс, оставляя транзистор в неопределённом состоянии.

   ```
   echo out > /sys/class/gpio/gpio32/direction
   ```

4. Открываем транзистор, подавая логический высокий уровень на затвор:

```
echo 1 > /sys/class/gpio/gpio32/value
```

5. Закрываем транзистор, подавая логический ноль на затвор:

```
echo 0 > /sys/class/gpio/gpio32/value
```

Работа с прерываниями

Через интерфейс sysfs можно запросить прерывания по изменению состояния процессора.

Установка прерывания производится путём записи значения в файл "edge". Значения могут быть:

- `none` — отключить прерывание
- `rising` — включить прерывание по нисходящему фронту
- `falling` — включить прерывание по восодящему фронту
- `both` — включить прерывание по обеим фронтам.

Пример работы с прерываниями:

```
~# echo 3 >  /sys/class/gpio/export # экспортируем GPIO номер 3 (TB10 на WB3.3)
~# cat /sys/class/gpio/gpio3/edge   # проверяем состояние прерывания
none
~# echo falling > /sys/class/gpio/gpio3/edge # устанавливаем прерывание по нисходящему фронту
~# cat /proc/interrupts  | grep gpiolib # прерывание появилось в списке. 26 - внутренний номер прерывания, 0 - количество событий
 26:          0  gpio-mxs   3  gpiolib
~# cat /proc/interrupts  | grep gpiolib # после нескольких собтий, 76 - количество событий
 26:         76  gpio-mxs   3  gpiolib
```

Прерывания можно ловить из userspace с помощью системного вызова epoll() и select() на файл value. Пример работы см. [1] (https://github.com/contactless/wiegand-linux-sysfs)

См. также elinux.org (http://elinux.org/GPIO)

Работа через chardev

Представленный в ядре 4.8 интерфейс chardev имеет C/Python библиотеку *libgpiod* и userspace-утилиты для работы с gpio. Исходный код библиотеки и документация доступны в репозитории libgpiod (https://github.com/brgl/libgpiod).

Утилиты распространяются в составе debian-пакетов *gpiod* и *libgpiod-dev* для debian buster и новее. К сожалению, **для stretch пакетов в официальных репозиториях нет**.

Если нужно установить libgpiod в debian stretch, можно воспользоваться сторонними репозиториями (например, этим (https://github.com/rcn-ee/repos)). **Используйте сторонние репозитории на свой страх и риск; компания WirenBoard не контролирует их содержимое**.

Для работы с gpio из bash в пакете *gpiod* поставляются следующие утилиты:

- `gpiodetect` — информация обо всех банках gpio в системе
- `gpioinfo` — подробная информация обо всех линиях gpio определённого банка
- `gpioget <чип> <линия>` — возвращает значение определённого gpio
- `gpioset <чип> <линия1>=<значение1> <линия2>=<значение2>` — устанавливает состояние на определенные линии gpio
- `gpiofind <название>` — возвращает номер gpio
- `gpiomon` — отслеживание событий gpio

Примеры использования gpiod можно посмотреть в [2] (https://www.acmesystems.it/gpiod) и [3] (https://github.com/brgl/libgpiod)

Python

Для управления gpio из python был написан модуль

```
wb_common.gpio
```

Модуль представляет собой обёртку вокруг sysfs. Исходный код доступен на нашем github. (https://github.com/wirenboard/wb-common/blob/master/wb_common/gpio.py)

Модуль позволяет работать с gpio в синхронном и асинхронном (с регистрацией коллбэков) режимах.

Прямое обращение через память процессора

Этот метод настоятельно НЕ РЕКОМЕНДУЕТСЯ для использования без достаточных оснований. Для работы из C/C++ стоит использовать работу через файлы в sysfs или chardev, как описано в предыдущих разделах.

Управлять GPIO можно с помощью прямого доступа к регистрам процессора, в обход Linux, через интерфейс /dev/mem. При этом, по сравнению с работой через sysfs минимизируются накладные расходы. Этот метод можно использовать, если вам необходим очень быстрый доступ к GPIO, например bitbang протоколов или ШИМ. Стоит иметь в виду, что планировщик процессов всё ещё может вносить в работу программы значительные задержки. Рекомендуется выносить критичные ко времени задачи в ядро.

См. [4] (http://olimex.wordpress.com/2012/09/11/imx233-olinuxino-gpios-faster-and-faster/) , [5] (https://github.com/OLIMEX/OLINUXINO/blob/master/SOFTWARE/iMX233/gpio-mmap.h)

Работа в ядре Linux

Ознакомиться с ядром Linux, использующимся в контроллерах WirenBoard можно в нашем репозитории ядра (https://github.com/wirenboard/linux).

Рекомендации по Device Tree

Device-tree, использующиеся на контроллерах WirenBoard, доступны в репозитории ядра. Разные аппаратные ревизии контроллера используют разные dts (например, dts для WB6.X можно найти здесь (https://github.com/wirenboard/linux/blob/dev/v4.9.x/arch/arm/boot/dts/imx6ul-wirenboard61.dts))

Указывать GPIO в Device Tree необходимо для настройки работы GPIO в режиме программного SPI, I2C, для использования GPIO в качестве источника прерываний и т.д. Так, например, на пин 10@UEXT1 (CS) и пины 5@UEXT2 (SCL), 6@UEXT2 (SDA), 10@UEXT2 (CS) выведены линии GPIO процессора. Их можно сконфигурировать для использования, например, в качестве chip-select для SPI или в качестве I2C.

GPIO процессора и периферийных устройств разбиты на банки (gpiochip). GPIO процессора разбиты на 3 банка по 32 GPIO: gpio0, gpio1, gpio2. Адресация GPIO в Device Tree происходит по номеру банка и номеру GPIO *внутри* банка.

Пример device-tree node

Определим сигнал 6@UEXT2 (SDA) в качестве источника прерываний для драйвера mrf24j40. Согласно таблице Список GPIO, сигнал соответствует GPIO 53 процессора. 53 принадлежит второму банку gpio (от 32 до 63). Номер GPIO внутри банка 53-32=21 :

```
6lowpan@0 {
    compatible = "microchip,mrf24j40";
    spi-max-frequency = <100000>;
    reg = <6>;
    interrupt-parent = <&gpio1>;
    interrupts = <21 0>;
};
```

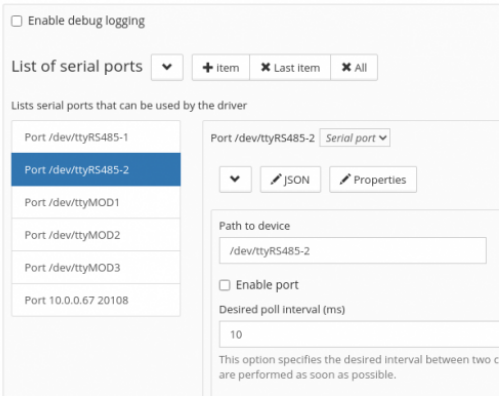
Работа с последовательным портом (serial-портом)

Contents

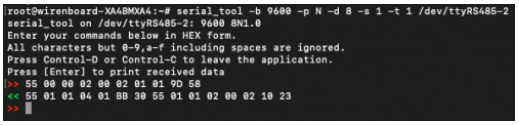
Serial-порты в контроллере Wiren Board

- Реализация
- Определение занятости порта и монопольное использование

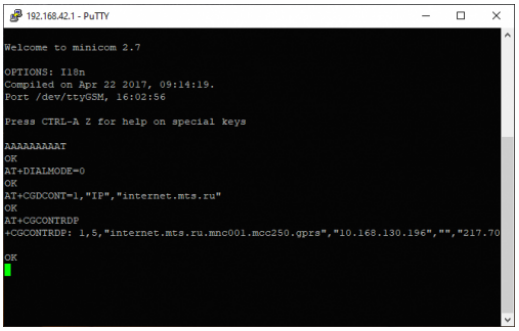
Программы для работы с serial-портом



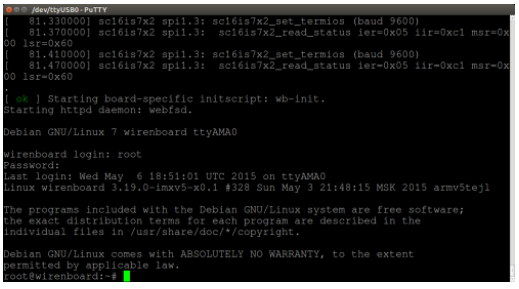
Отключение serial-порта в веб-интерфейсе контроллера Wiren Board



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/).

Zabbix

Смотрите также:

- Репозиторий с устаревшим нативным мостом в Zabbix (https://github.com/contactless/wb-mqtt-zabbix). Пакеты в разделе Releases (https://github.com/wirenboard/wb-mqtt-zabbix/releases).
- Статья на Хабре «Zabbix + Wirenboard: мониторинг производства (https://habr.com/ru/post/525852/)» — подробное описание создания системы с Wiren Board и Zabbix. В статье используется плагин, написанный пользователем — zbx_mqtt (https://github.com/v-zhuravlev/zbx_mqtt)

Подготовка

Нужно поставить zabbix-agent:

```
apt-get install zabbix-agent -y
```

Настройка Zabbix Agent

Для получения значений переменных в Zabbix используется чтение retained-значений каналов.

Для настройки `zabbix-agent` нужно создать файл `zabbix-mqtt.conf` в директории `/etc/zabbix/zabbix_agentd.conf.d/` и записать в него строку:

```
UserParameter=mqtt.value[*],mosquitto_sub -t '$1' -C 1
```

Это можно сделать одной командой:

```
echo "UserParameter=mqtt.value[*],mosquitto_sub -t '\$1' -C 1" > /etc/zabbix/zabbix_agentd.conf.d/zabbix-mqtt.conf
```

Использование

Обращение к каналам в Zabbix происходит следующим образом:

```
mqtt.value[topic]
```

где topic - топик соответствущего канала в MQTT

например:

```
mqtt.value[/devices/sht1x/controls/humidity]
mqtt.value[/devices/sht1x/controls/temperature]
```

Пример теста агента, запускать на контроллере:

```
zabbix_agentd -t mqtt.value[/devices/wb-adc/controls/5Vout]
```

RS-485

Contents

Описание

Как правильно проложить шину

Добавление устройства в веб-интерфейс

Как ускорить опрос устройств

Работа с портом RS-485 контроллера из собственного ПО

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

SC16IS752

- English
- русский

SC16IS752 - микросхема, управляющая последовательными портами в некоторых версиях Wiren Board.

Contents

Описание
- GPIO

В Wiren Board Smart Home 3.5
- UART0
- UART1

В Wiren Board 2.8
- UART0
- UART1
- GPIO

Описание

SPI-to-UART bridge IC - NXP SC16IS752

Два полных порта UART, До 8 линий GPIO.

Подключен к SPI. Используется GPIO в качестве chip-select.

Для работы в Linux используется драйвер sc16is7x2, который пока не портирован в upstream. Его можно найти в нашем репозитории Linux [1]. Драйвер включён в стандартный образ.

GPIO

Драйвер sc16is7x2 экспортирует 8 GPIO в стандартный sysfs-интерфейс Linux под номерами 120-127.

В Wiren Board Smart Home 3.5

UART0

Устройство в Linux: **/dev/ttyNSC0**. Полудуплексный RS-485-трансивер. Порт RS-485-1 (зелёная пара клеммников).

Направление работы трансивер задаётся чипом с помощью сигнала RTS автоматически, аппаратный контроль потока должен быть отключен. В следующих версиях драйвера, режим RS-485 возможно будет необходимо активировать из userspace.

UART1

Устройство в Linux: **/dev/ttyNSC1**. Полудуплексный RS-485-трансивер. Порт RS-485-2 (синяя пара клеммников).

Управляющие сигналы/GPIO, а также линии RX/TX выведены на разъём WB SH 3.5: UEXT2. Разъём предназначен для использования GPIO, а также для подключения модуля расширения RS232.

B Wiren Board 2.8

UART0

Устройство в Linux: **/dev/ttyNSC0**. TX/RX линии порта выведены на разъём UEXT UEXT2. Параллельно к ним подключен полудуплексный RS-485-трансивер (если есть). Направление работы трансивер задаётся чипом с помощью сигнала RTS автоматически, аппаратный контроль потока должен быть отключен. В следующих версиях драйвера, режим RS-485 возможно будет необходимо активировать из userspace.

UART1

Порт **/dev/ttyNSC1** в Linux. К TX/RX-линиям порта подключен GPS-модуль SIM68V.

GPIO

7 линий GPIO выведены на отверстия с шагом 2.54 *на плате Wiren Board.* К одной из линий GPIO подключен сигнал PPS с GPS-модема.

Драйвер sc16is7x2 экспортирует 8 GPIO в стандартный sysfs-интерфейс Linux под номерами 120-127.

Wi-Fi

Contents

Режимы работы

Первое подключение по Wi-Fi

- Антенны
- Подключение к точке доступа

Настройка Wi-Fi на контроллере Wiren Board

- Настройка в режиме точки доступа
- Установка пароля на подключение к точке доступа
- Настройка в режиме точки доступа и клиента одновременно
- Отключение режима точки доступа
- Настройка в режиме клиента
- Подключение к Wi-Fi точке доступа вручную
- Универсальный файл настроек Wi-Fi
- Автоматическое переподключение при проблемах с соединением

Режимы работы

Wi-Fi в Wiren Board можно настроить на работу в одном из двух или трёх режимов:

- Режим точки доступа (включён по умолчанию). Работает относительно медленно. Скорости вполне хватит для работы с веб-интерфейсом, но не стоит использовать как замену роутера.
- Режим клиента.
- Одновременная работа в режиме и точки доступа, и клиента.

В очень редких случаях возможна несовместимость адаптера Wi-Fi в Wiren Board с некоторыми другими устройствами Wi-Fi. Это общая проблема реализаций Wi-Fi на чипсетах разных производителей. Если вы столкнулись с необъяснимыми проблемами при работе, рекомендуем поменять настройки шифрования, ширины канала и т.п.

Первое подключение по Wi-Fi

Антенны

Прикрутите антенну к разъёму для антенны Wi-Fi.

Без антенны Wi-Fi в контроллерах Wiren Board работает на расстоянии не более одного метра. Чтобы получить стандартный для Wi-Fi радиус работы, нужно подключить к соответствующему разъёму контроллера антенну. Если контроллер находится в щитке (особенно в металлическом) или отдельной комнате, лучше использовать выносную антенну. Разъём для антенны — стандартный для Wi-Fi RP-SMA (https://en.wikipedia.org/wiki/SMA_connector#Reverse_polarity_SMA) ("гнездо", у GSM-антенн - наоборот).

Сравнение разъёмов для антенн Wi-Fi (RP-SMA) и GSM (SMA)

Подключение к точке доступа

Контроллер создает Wi-Fi точку доступа и мы можем подключиться к ней:

- Откройте на ноутбуке или телефоне список WiFi точек доступа.
- Выберите из списка точку доступа с именем `WirenBoard-XXXXXXXX`. Где XXXXXXXX - серийный номер контроллера.

При подключении по Wi-Fi контроллер будет доступен по IP-адресу **192.168.42.1**.

По умолчанию, для подключения к контроллеру по Wi-Fi не требуется пароль, но вы можете это изменить.

Настройка Wi-Fi на контроллере Wiren Board

Настройка производится стандартным для Linux Debian способом - через файл `/etc/network/interfaces`. Краткие инструкции для типовых задач даны ниже, на сайте Linux Debian есть подробная документация (https://wiki.debian.org/ru/NetworkConfiguration).

Настройка в режиме точки доступа

Режим точки доступа включён по умолчанию. Работа в режиме точки доступа обеспечивается демоном **hostapd (https://wireless.wiki.kernel.org/en/users/documentation/hostapd)**.

Сперва настраиваем демон `hostapd`:

1. в файле `/etc/default/hostapd` раскомментируйте строку (то есть удалите знак **#** в начале строки)

```
DAEMON_CONF="/etc/hostapd.conf"
```

2. отредактируйте файл `/etc/hostapd.conf`, чтобы он выглядел так:

```
interface=wlan0
#driver=nl80211 # оставьте эту строку закомментированной
ssid=WirenBoard # вместо WirenBoard можете подставить другое имя для создаваемой точки доступа
channel=1
```

Теперь нужно настроить сам интерфейс. Настройка делается в файле `/etc/network/interfaces`:

1. раскомментируйте и отредактируйте (или добавьте, если их не было) строки, относящиеся к настройке в режиме точки доступа:

```
iface wlan0 inet static
      address 192.168.42.1 # здесь 192.168.42.1  - адрес, по которому в новой сети будет находиться Wiren Board; можете
указать другой адрес
      netmask 255.255.255.0
```

2. закомментируйте строки, относящиеся к работе в режиме клиента:

```
#auto wlan0
#iface wlan0 inet dhcp
#                wpa-ssid {ssid}
#                wpa-psk  {password}
```

Выполните команду

```
/etc/init.d/hostapd restart
```

В итоге у нас получилась открытая точка доступа, для подключения к которой не требуется пароль.

Установка пароля на подключение к точке доступа

Подключитесь к контроллеру по SSH и откройте файл настроек `/etc/hostapd.conf`, для этого введите команду:

```
nano /etc/hostapd.conf
```

Добавьте в конец файла строки:

```
wpa=2
wpa_passphrase=your_password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=TKIP CCMP
```

Придумайте свой пароль и замените в файле `your_password` на него. Сохраните файл нажатием клавиш `Ctrl+O` и выйдете из редактора `Ctrl+X`.

После этого выполните команду:

```
/etc/init.d/hostapd restart
```

Контроллер применит новые настройки и связь с ним будет потеряна. Нужно будет заново подключиться к контроллеру по WiFi с указанным паролем. Если изменения настроек вы не можете подключиться к контроллеру по WiFi — подключитесь к нему по Ethernet и проверьте настройки в файле `/etc/hostapd.conf`.

Настройка в режиме точки доступа и клиента одновременно

Режим одновременной работы модуля Wi-Fi и в режиме точки доступа, и в режиме клиента, называется *Concurrent Mode* или *STA+SoftAP*, и поддерживается не всеми Wi-Fi модулями. Он работает на всех версиях Wiren Board 6 и на некоторых ревизиях WB5. Проверено, что он работает из коробки на Wiren Board с чипом Realtek 8723BU и ядром Linux 4.1.15. Чтобы проверить, выполняются ли эти условия, выполните команды:

```
uname -a
lsmod | grep 8723bu
```

Если условия не выполнены, возможно, на вашем Wiren Board, всё равно, можно настроить Concurrent Mode. В качестве отправной точки используйте инструкцию (http://randomstuffidosometimes.blogspot.ru/2016/03/rtl8192cu-and-rtl8188cus-in-station-and.html).

Если условия выполнены:

1. Выполните команду

   ```
   iwconfig
   ```

   В её выводе должны быть показаны два интерфейса Wi-Fi: *wlan0* и *wlan1*.

2. Настройте по двум предыдущим инструкциям подключение в режиме клиента и подключение в режиме точки доступа, но используйте для них разные интерфейсы. Например, оставьте *wlan0* для точки доступа, а клиента сделайте на *wlan1*. Соответствующая часть файла */etc/network/interfaces* должна выглядеть так:

   ```
   # Wireless interfaces
   auto wlan1
   iface wlan1 inet dhcp
       wpa-ssid {ssid} # вместо {ssid} подставьте имя точки доступа
       wpa-psk {password} # вместо {password} подставьте пароль

   auto wlan0
   ```

```
iface wlan0 inet static
  address 192.168.42.1
  netmask 255.255.255.0
```

Отключение режима точки доступа

Если вы хотите перевести адаптер в режим клиента, подключиться к Wi-Fi точке доступа в ручном режиме или совсем отключить Wi-Fi на контроллере — отключите режим точки доступа:

1. Отключите автоматический запуск сервиса `hostapd`:

```
systemctl disable hostapd
```

2. Остановите демон `hostapd`

```
service hostapd stop
```

3. Теперь закомментируйте настройки точки доступа и задайте настройки WiFi-клиента:
   - откройте файл для редактирования

```
mcedit /etc/network/interfaces
```

   - закомментируйте строки, относящиеся к настройке в режиме точки доступа:

```
#allow-hotplug wlan0
#iface wlan0 inet static
#  address 192.168.42.1
#  netmask 255.255.255.0
```

4. Сохраните и закройте файл настроек.
5. Запретите раздачу IP-адресов, для этого остановите DHCP-сервер:

```
systemctl disable dnsmasq
service dnsmasq stop
```

Режим точки доступа отключен, чтобы его включить, выполните инструкции из раздела Настройка в режиме точки доступа.

Настройка в режиме клиента

> После настройки точки доступа в режиме клиента, контроллер будет подключаться к точке доступа автоматически при каждой загрузке операционной системы.

Вы можете настроить автоматическое подключение контроллера к Wi-Fi точке доступа:

1. Отключите точку доступа по инструкции в разделе Отключение режима точки доступа
2. Откройте файл настроек:

```
mcedit /etc/network/interfaces
```

3. Раскомментируйте и отредактируйте строки (или добавьте, если их не было):

```
auto wlan0
iface wlan0 inet dhcp
  wpa-ssid ssid # вместо ssid подставьте имя точки доступа
  wpa-psk  password # вместо password подставьте пароль
```

4. Если точка доступа скрыта, то добавьте параметр:

```
  wpa-scan-ssid 1
```

5. Сохраните и закройте файл настроек.
6. Завершите настройку, для этого перезапустите беспроводной интерфейс командами:

```
ifdown wlan0 && ifup wlan0
```

Подключение к Wi-Fi точке доступа вручную

> Подключение в ручном режиме будет разорвано после перезагрузки контроллера.

Если у вас возникла проблема с настройкой автоматического подключения, то вы можете попробовать подключиться к Wi-Fi точке доступа вручную:

1. Отключите точку доступа по инструкции в разделе Отключение режима точки доступа
2. Запустите поиск доступных точек доступа с помощью команды `iwlist wlan0 scanning`:

```
~# iwlist wlan0 scanning | grep -i essid
                    ESSID:"DIR-615"
                    ESSID:"MTSRouter_2.4GHz_072433"
                    ESSID:"Smart_box-40B598"
                    ESSID:"TP-Link_0E5AW"
                    ESSID:"TP-LINK_78DC"
```

   в примере контроллер «видит» пять точек доступа.

3. Этот шаг зависит от типа сетевой аутентификации, выбранной в настройках точки доступа, к которой вы хотите подключиться:

   - WPA-PSK:

   1. Задайте параметры подключения:

```
iwconfig wlan0 essid ИмяТочкиДоступа key ПарольОтТочкиДоступа
```

   2. Запустите сетевой интерфейс:

```
ifconfig wlan0 up
```

   - WPA2-PSK:

   1. Сгенерируйте файл с учётной записью для подключения к точке доступа:

```
wpa_passphrase ИмяТочкиДоступа ПарольОтТочкиДоступа > /root/wpa.conf
```

   2. Установите подключение с использованием сгенерированного файла:

```
wpa_supplicant -Dwext -iwlan0 -c/root/wpa.conf &
```

4. Подождите 15 секунд и проверьте подключение командой `iwconfig wlan0`:

```
~# iwconfig wlan0 | grep -i essid
wlan0     IEEE 802.11bgn  ESSID:"DIR-615"  Nickname:"<WIFI@REALTEK>"
```

   в примере контроллер подключён к точке доступа с именем `DIR-615`. Если в строке будет `unassociated`, то контроллер не смог подключиться.

5. Если контроллер успешно подключился к точке доступа и на ней запущен DHCP-сервер, то запустите `dhcpclient`:

```
dhclient wlan0
```

6. Проверьте, получил ли контроллер IP адрес, для этого используйте команду `ip a`:

```
~# ip a | grep wlan0
5: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    inet 192.168.2.83/24 brd 192.168.2.255 scope global wlan0
```

   в примере контроллер получил ip-адрес `192.168.2.83`.

Настройка подключения контроллера к точке доступа завершена.

Универсальный файл настроек Wi-Fi

Ниже приведен текст файла с настройками для подключения к сетям с разными параметрами шифрования. Оригинал файла можно посмотреть на сайте www.raspberrypi.org (https://www.raspberrypi.org/forums/viewtopic.php?t=7592).

```
####################
#; start of wireless bits
#; this command stays for all configs
auto wlan0
####################
#; comments indicated by #;
#; commands indicated by #
#; remove the # to enable the command
####################
#; if using static IP then....#
#iface wlan0 inet static
# address UR_IP
#gateway UR_ROUTER_IP
#netmask 255.255.255.0
##################
#; otherwise use dhcp #
#iface wlan0 inet dhcp
##################
#; OPEN wireless config #
#wireless-essid UR_ESSID
#wireless-mode managed

##################
#; WEP wireless config #
#wireless-essid UR_ESSID
#wireless-key UR_KEY
#; end of WEP config

########################
#; WPA and WPA2 wireless config #
#; all command config lines above HERE to be #'ed except the entry auto wlan0
########################
wpa-driver wext
wpa-ssid UR_ESSID
#; wpa-ap-scan is 1 for visible and 2 for hidden hubs
wpa-ap-scan 1
#; wpa-proto is WPA for WPA1 (aka WPA) or RSN for WPA2
wpa-proto WPA
#; wpa-pairwise and wpa-group is TKIP for WPA1 or CCMP for WPA2
wpa-pairwise TKIP
wpa-group TKIP
wpa-key-mgmt WPA-PSK
#; use "wpa_passphrase UR_ESSID UR_KEY" to generate UR_HEX_KEY
#; enter the result below
```

```
wpa-psk UR_HEX_KEY
#########################
# end of wireless bits
```

Автоматическое переподключение при проблемах с соединением

Способ заимствован на сайте alexba.in (http://alexba.in/blog/2015/01/14/automatically-reconnecting-wifi-on-a-raspberrypi/).

Допустим, контроллер подключён к роутеру с адресом `192.168.0.1` через интерфейс `wlan1`:

1. Создайте в папке `/root` скрипт `wifi_autoconnect.sh`:

   ```
   mcedit /root/wifi_autoconnect.sh
   ```

   с содержанием:

   ```
   #!/bin/bash

   # Подставьте имя интерфейса
   WLANINTERFACE=wlan1
   # Подставьте адрес роутера или сервера в интернете, доступ к которому будет проверяться
   SERVER=192.168.0.1

   PATH="/bin:/sbin:/usr/local/sbin:/usr/sbin:$PATH"
   # Only send two pings, sending output to /dev/null
   ping -I ${WLANINTERFACE} -c2 ${SERVER} > /dev/null

   # If the return code from ping ($?) is not 0 (meaning there was an error)
   if [ $? != 0 ]
   then
   # Restart the wireless interface
   ifdown --force ${WLANINTERFACE}
   ifup ${WLANINTERFACE}
   fi
   ```

2. Сделайте файл исполняемым, выполнив команду

   ```
   chmod +x /root/wifi_autoconnect.sh
   ```

3. Запланируйте выполнение скрипта каждую минуту:

   Добавьте в конец файла `/etc/crontab` строку

   ```
   *  *    * * *   root    /root/wifi_autoconnect.sh
   # Обязательно добавьте пустую строку в конец файла
   ```

Зуммер (звуковой излучатель)

Contents

Описание

Управление из веб-интерфейса

Управление из движка правил

Управление из python

Низкоуровневая работа

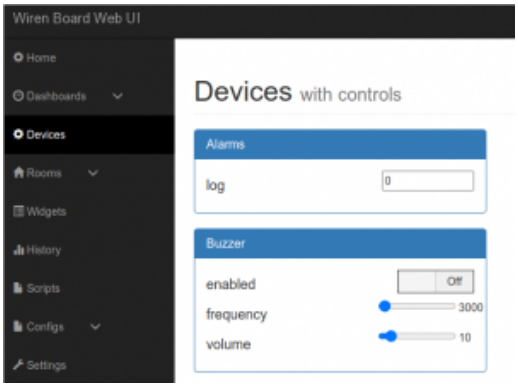
- О ШИМ и пересчёт параметров
- Номер pwm-порта для sysfs
- Работа из sysfs

Описание

Контроллер Wiren Board имеет на борту Зуммер (звуковой излучатель). Зуммер питается от 5В и управляется ножкой gpio процессора в режиме ШИМ. Управлять зуммером можно через sysfs-интерфейс ядра и различное ПО поверх него. Сейчас реализовано управление из веб-интерфейса, движка правил wb-rules и python.

Управление из веб-интерфейса

В веб-интерфейсе контроллера управление зуммером доступно во вкладке *"Devices"*. Параметр *"Frequency"* - частота звука в Гц. *"Volume"* - громкость (в условных единицах, шкала линейная). Параметры сохраняются при перезагрузке контроллера.



Управление зуммером

Управление из движка правил

Управление зуммером, выведенное в веб-интерфейс - это виртуальное устройство, созданное системным правилом wb-rules при старте контроллера. Исходный код правила доступен на нашем github (https://github.com/wirenboard/wb-rules-system/blob/master/rules/buzzer.js).

О том, для чего нужны виртуальные устройства, можно узнать подробнее в описании движка правил.

Системное правило внутри реализует пересчёт тональности и громкости (см раздел о пересчёте) и работу с pwm через sysfs (см соответствующий раздел). Наружу пользователю доступно устройство *"buzzer"*, имеющее несколько mqtt-контролов:

Device	Control	Тип	Максимальное значение	Описание
Buzzer	Frequency	Range	7000	Частота звука
	Volume	Range	100	Громкость, %
	Enabled	Switch		Включение/отключение

Контролы устройства можно использовать в собственных правилах. Подробнее о структуре mqtt-топиков виртуальных и физических устройств можно узнать из нашей mqtt-конвенции (https://github.com/wirenboard/conventions/blob/main/README.md).

Управление из python

На контроллерах Wiren Board работать с зуммером можно из python с помощью модуля *beeper* из пакета *wb_common*. Это обёртка вокруг интерфейса sysfs. Модуль предустановлен на все контроллеры в составе deb-пакета *python-wb-common*. Исходный код доступен на нашем github (https://github.com/wirenboard/wb-common/blob/master/wb_common/beeper.py).

Пример работы из python:

```
from wb_common import beeper

beeper.beep(0.5, 2)
```

Поддерживаются все настройки sysfs-интерфейса (пересчёт нужно проводить вручную; см раздел о пересчёте).

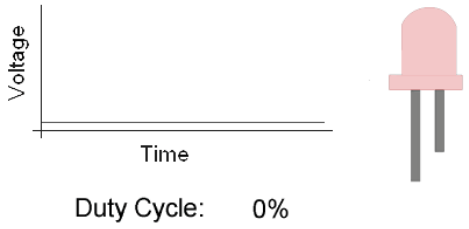
Низкоуровневая работа

О ШИМ и пересчёт параметров

ШИМ (PWM) - это распространённый способ управления мощностью, подаваемой к нагрузке.

В контексте управления зуммером, нас интересуют 2 параметра PWM:

- Коэффициент заполнения (duty cycle) - влияет на громкость звука. Обычно, считается в процентном соотношении от периода сигнала.
- Частота PWM (frequency) - влияет на высоту звука (чем выше частота, тем выше и звук). Единица, обратная периоду сигнала.



Duty cycle управляет яркостью светодиодов / громкостью Зуммера

Ядро Linux предоставляет интерфейс sysfs для pwm, который принимает частоту pwm и duty cycle в **наносекундах (10^{-9}C)**! Поэтому, для низкоуровневого управления Buzzer'ом нужно производить пересчёт желаемой частоты из kHz в период в наносекундах по формуле: **T(ns) = 1 000 000 / f(kHz)**

Номер pwm-порта для sysfs

Ножка gpio настраивается, как выход PWM в dts ядра linux. Подробнее можно посмотреть [на нашем github (https://github.com/wirenboard/linux/blob/ef2d87e222b365848fe7262c022ca887b6449432/arch/arm/boot/dts/imx6ul-wirenboard61.dts#L495)].

- Для контроллеров WB6.X.X номер порта = 0,(для всех контроллеров до WB6.X.X номер порта = 2)
- Номер порта можно узнать, выполнив команду

```
echo $WB_PWM_BUZZER
```

Во всех примерах далее будем считать, что номер pwm-порта = 0.

Работа из sysfs

Для работы с pwm через sysfs нужно:

1. Экспортировать порт

```
echo 0 > /sys/class/pwm/pwmchip0/export
```

   После этого появляется директория /sys/class/pwm/pwmchip0/pwm0
2. Записать период pwm в наносекундах

```
echo 250000 > /sys/class/pwm/pwmchip0/pwm0/period # устанавливаем период в 250 000 нс, т.е. в 250мкс, что соответствует частоте 4кГц
```

3. Записать громкость (пересчитав из duty-cycle)

```
echo 125000 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle # максимальная громкость достигается при duty_cycle = period / 2 => устанавливаем duty_cycle в 125 000 нс
```

4. Включить выход PWM

```
echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```

Для выключения зуммера, нужно записать 0:

```
echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
```

Пример bash-скрипта для работы с pwm (https://github.com/contactless/wirenboard/tree/master/examples/beeper)

Установка периода в наносекундах. Пересчёт из частоты (в килогерцах) в период (в наносекундах) производится по формуле: **T(ns) = 1 000 000 / f(kHz)**

Device Tree

- English
- русский

Device Tree - файлы описания аппаратной конфигурации. Они используются в Linux и, следовательно в Wiren Board. Модификация Device Tree-файлов может понадобится при переназначении портов, подключении некоторых UEXT-устройств, при подключении 1-wire датчиков и т.д.

Подробнее про Device Tree: https://en.wikipedia.org/wiki/Device_tree , http://elinux.org/Device_Tree

Device Tree для Wiren Board находится в файле imx23-wirenboard28.dts. В нём также используется файл описания процессора, imx23.dtsi

Файлы Device Tree бывают в текстовом формате dts, который компилируется в бинарный формат dtb

Contents

Загрузка

Пересборка

- Загрузка
- Компилятор
- Компиляция
- Установка

Загрузка

В стандартом образе Wiren Board загрузчик U-Boot считывает dtb-файл описания Device Tree и передаёт его ядру. Имя dtb-файла задаётся в файле /boot/uEnv.txt :

```
#These are the default settings for some useful u-boot variables:
fdt_file=/boot/dtbs/imx23-wirenboard28.dtb
```

Файл dtb находится в /boot/dtbs/imx23-wirenboard28.dtb (для Wiren Board rev. 2.8), в /boot/dtbs/imx23-wirenboard32.dtb (для Wiren Board Smart Home rev. 3.5)

Пересборка

Для внесения изменений в Device Tree надо скачать Device Tree в текстовом формате, скомпилировать файл imx23-wirenboard28.dts и записать результат (imx23-wirenboard28.dtb) в /boot/dtbs/

Другой способ - Сборка ядра

Загрузка

Необходимо скачать из [1] файл imx23-wirenboardXX.dts, соответствующий версии устройства и зависимости. Зависимости на настоящий момент это файлы imx23.dtsi, skeleton.dtsi.

Основной DTS-файл:

imx23-wirenboard32.dts для Wiren Board Smart Home rev. 3.5

imx23-wirenboard28.dts для Wiren Board rev. 2.8

```
$ mkdir dts
$ cd dts

$ #export WB_BRANCH=v3.12-rc3-imxv5-x0.3  # для ядра 3.12
$ export WB_BRANCH=v3.13-imxv5-x0.1

$ wget https://raw.github.com/contactless/linux/$WB_BRANCH/arch/arm/boot/dts/mxs-pinfunc.h

$ wget https://raw.github.com/contactless/linux/$WB_BRANCH/arch/arm/boot/dts/imx23-wirenboard28.dts

2013-11-17 04:24:28 (37.9 MB/s) - «imx23-wirenboard28.dts» сохранён [5255/5255]

$ wget https://raw.github.com/contactless/linux/$WB_BRANCH/arch/arm/boot/dts/imx23.dtsi

2013-11-17 04:24:33 (272 KB/s) - «imx23.dtsi» сохранён [13052/13052]

$ wget https://raw.github.com/contactless/linux/$WB_BRANCH/arch/arm/boot/dts/skeleton.dtsi

$ wget https://raw.github.com/contactless/linux/$WB_BRANCH/arch/arm/boot/dts/imx23-pinfunc.h

$ ls
imx23.dtsi       imx23-wirenboard28.dts  skeleton.dtsi
imx23-pinfunc.h  mxs-pinfunc.h
```

Компилятор

Компилятор Device Tree в Ubuntu и Debian находится в пакете device-tree-compiler. Установим его:

```
$ sudo apt-get install device-tree-compiler
```

Также требуется gcc

```
$ sudo apt-get install gcc
```

Компиляция

```
$ gcc -E -Wp,-MD,imx23-wirenboard28.dtb.d.pre.tmp -nostdinc -I. -undef -D__DTS__ -x assembler-with-cpp -o .imx23-
wirenboard28.dtb.dts.tmp  imx23-wirenboard28.dts

$ cat .imx23-wirenboard28.dtb.dts.tmp | grep -v "^#" | dtc -I dts -O dtb -o imx23-wirenboard28.dtb
DTC: dts->dtb  on file "-"
```

Установка

Скомпилированный файл необходимо записать в /boot/dtsb. **Обязательно сделайте резервную копию существущего в /boot/dtsb файла!**

Сборка ядра Linux

Сборка ядра Linux вручную может понадобиться, например, если нужно включить в ядро модули, отсутствующие в стандартной поставке Wiren Board. Если вы не знаете, для чего вам это нужно, то, скорее всего, вам не нужно собирать ядро вручную.

Начиная с 8 апреля 2021 года, скрипты для сборки deb-пакетов ядра Linux для Wiren Board добавлены в репозиторий с кодом ядра: http://github.com/wirenboard/linux.

Contents

Подготовка сборочной машины

Получение исходного кода

Сборка

Сборка deb-пакета

Разные модели контроллеров

Настройка ядра

Сборка вручную

Подготовка сборочной машины

Сборка ядра должна производиться на настольном компьютере, ноутбуке или сервере под управлением Linux. Собирать ядро на самом Wiren Board не стоит - у контроллера не хватит дискового пространства для получения репозитория, а также вычислительной мощности. Даже на настольном компьютере сборка ядра может занять десятки минут.

Сборочные скрипты писались с расчётом на дистрибутивы Debian и Ubuntu, инструкции в этой статье приводятся также из расчёта использования этих дистрибутивов. Инструкция была проверена в Ubuntu 18.04.

Для сборки ядра понадобится установить пакеты с необходимым для сборки ПО:

```
$ sudo apt update && sudo apt install build-essential libncurses5-dev fakeroot lzop bc git
```

Если вы собираете ядро для **Wiren Board 6** и новее, нужно установить компилятор для **arm-linux-gnueabihf**:

```
$ sudo apt install gcc-arm-linux-gnueabihf
```

Для **Wiren Board 5** и более старых понадобится другой компилятор:

```
$ sudo apt install gcc-arm-linux-gnueabi
```

Получение исходного кода

Исходный код ядра Linux с правками от команды Wiren Board хранится в репозитории на Github. Чтобы получить его на свой компьютер, выполните команды:

```
$ git clone https://github.com/wirenboard/linux
$ cd linux
$ git submodule update --init --recursive
```

Сборка

Все промежуточные и конечные артефакты сборки - объектные файлы, dtb, модули ядра .ko, zImage и т.д. - будут находится в поддиректории **.build-wbX**, где X зависит от модели контроллера. В этой же директории находится конфигурация ядра (см. ниже).

Чаще всего для использования на контроллере удобней всего собрать deb-пакет с файлами ядра.

Если вам нужны только некоторые бинарные файлы (zImage, модули и dtbs), то после успешной сборки deb-пакета их можно будет найти в сборочной поддиректории **.build-wbX**.

Сборка deb-пакета

```
$ make mrproper
$ KERNEL_FLAVOUR=wb6 VERSION_SUFFIX="~my~test~kernel" ./scripts/package/wb/do_build_deb.sh  # Wiren Board 6
```

Можно поменять значение VERSION_SUFFIX на свой вкус согласно правилам оформления версий пакетов в Debian. Это значение будет добавлено в конец номера версии пакета и поможет отличить собранное вручную ядро от ядра из репозитория Wiren Board.

После сборки в корне появятся файлы пакетов (пример для Wiren Board 6):

- linux-image-wb6_4.9.22-wb1~my~test~kernel_armhf.deb - образ ядра, модули и dtbs;
- linux-headers-wb6_4.9.22-wb1~my~test~kernel_armhf.deb - нужен для разработки
- linux-libc-dev_4.9.22-wb1~my~test~kernel_armhf.deb - нужен для разработки

На контроллер достаточно скопировать файл пакета linux-image-wb6, в нём уже содержится всё необходимое.

Разные модели контроллеров

Модель контроллера	KERNEL_FLAVOUR	файл defconfig	сборочная директория	переменная CROSS_COMPILE	файлы dts и dtb
Wiren Board 6	wb6	imx6_wirenboard_defconfig	.build-wb6	CROSS_COMPILE=arm-linux-gnueabihf-	imx6ul-wirenboard6*
Wiren Board 5 и ниже	wb2	mxs_wirenboard_defconfig	.build-wb2	CROSS_COMPILE=arm-linux-gnueabi-	imx28-wirenboard5*
Wiren Board 7	wb7	wirenboard7_defconfig	.build-wb7	CROSS_COMPILE=arm-linux-gnueabihf-	sun8i-r40-wirenboard7*

Настройка ядра

При запуске скрипт сборки deb-пакета спрашивает, использовать ли конфигурацию по-умолчанию:

```
$ KERNEL_FLAVOUR=wb7  scripts/package/wb/do_build_deb.sh
Building kernel packages for wb7 (Wiren Board 7)
Revision: -wb100
Architecture: armhf
Config: wirenboard7_defconfig
.config already present
Use wirenboard7_defconfig instead? (y/N)
```

Конфигурация по-умолчанюи при этом берётся из файлов в **arch/arm/configs/**, например **imx6_wirenboard_defconfig**.

Чтобы поменять конфигурацию, запустите **make** с необходимыми параметрами:

```
$ make KBUILD_OUTPUT=.build-wb6 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig # для Wiren Board 6
```

вместо **menuconfig** можно использовать графический **xconfig**.

После сохранения, конфигурация заишется в сборочную директорию: **.build-wbX/.config**.

Теперь вы можете собрать ядро с новой конфигурацией, выполнив обычную команду

```
$ KERNEL_FLAVOUR=wb7  scripts/package/wb/do_build_deb.sh
```

и ответив N, чтобы использовать новую конфигурацию, вместо конфигурации по-умолчанию:

```
Use wirenboard7_defconfig instead? (y/N)
```

Когда вы полностью довольны результатом, можно посмотреть изменения относительно исходной версии:

```
# для Wiren Board 6
$ make KBUILD_OUTPUT=.build-wb6 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- savedefconfig # приводит конфигурацию к стандартному виду и записывает в .build-wb6/defconfig
$ diff -u arch/arm/configs/imx6_wirenboard_defconfig .build-wb6/defconfig  # посмотреть её отличия от исходной конфигурации
```

Чтобы заменить исходную версию настроек своей (например, при подготовке патча или pull request):

```
# для Wiren Board 6
$ make KBUILD_OUTPUT=.build-wb6 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- savedefconfig
$ cp .build-wb6/defconfig  arch/arm/configs/imx6_wirenboard_defconfig
```

После этого скрипт **do_build_deb.sh** будет использовать обновлённую конфигуацию при ответе "y".

Сборка вручную

Этот этап не требуется для стандартных задач. Если вы хотите выполнить вручную какой-либо этап сборки, то можно запустить **make** вручную.

```
# для Wiren Board 6
$ make KBUILD_OUTPUT=.build-wb6 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage dtbs modules -j4
```

(-j4 запускает сборку в 4 потока, можно это убрать или поменять значение на более подходящее вашему компьютеру, идеальное значение - количество ядер CPU)

В результате появятся нужные нам файлы:

- .build-wb6/arch/arm/boot/zImage - образ ядра, который нужно скопировать в /boot/zImage на контроллере;
- .build-wb6/arch/arm/boot/dtb/* - файлы device tree, используемые для настройки оборудования при запуске, копируются в директорию /boot/dtbs/.

Файлы модулей можно найти с помощью команды:

```
$ find . -name '*.ko'
```

Далее файлы модулей копируются на контроллер в директорию /lib/modules/<версия ядра>/kernel/ с сохранением исходного пути, как было при сборке.

- Privacy policy
- About Wiren Board
- Disclaimers
-