

# ShIoTiny

*Контроллер для тех, кто не любит программировать, но  
любит рисовать*

2019

## Оглавление

1	Для чего или для кого? Или вместо введения.....	3
2	Что может ShIoTiny?.....	4
3	Что нужно для работы?.....	5
4	Как попасть на страницу контроллера ShIoTiny.....	9
5	Мы попали на страницу ShIoTiny.....	12
6	Первая программа.....	13
7	Усложняем задачу.....	16
8	Монитор.....	18
9	Наигрались? Теперь давайте по порядку.....	19
9.1	Узлы, связи и события.....	19
9.2	Какие бывают входы и выходы?.....	19
9.3	События: что это такое и с чем их подают к столу.....	20
10	Описание узлов (нод).....	25
10.1	Просто комментарий.....	25
10.2	Базовая логика (Basic logic).....	25
10.3	Простейшие входы и выходы (One-bit digital inputs, One-bit digital outputs).....	28
10.4	Аналого-цифровой преобразователь АЦП, он же ADC.....	31
10.5	Связь по протоколу MQTT.....	34
10.5.1	Сервер, порт, авторизация.....	35
10.5.2	Префикс.....	36
10.5.3	Безопасность (SSL).....	36
10.6	Связь по протоколу Rlink UDP.....	37
10.7	Датчики температуры и влажности.....	40
10.8	Математические функции.....	42
10.9	Функции работы со временем.....	44
10.10	Функции управления прохождением событий.....	46
10.11	Функции сетевого сервиса.....	47
10.12	Функции сохранения параметров в энергонезависимую память.....	48
10.13	Функции работы с календарем и синхронизации времени.....	49
10.14	Системные функции.....	50
10.15	Шина I2C.....	51
10.16	Модуль индикации на базе микросхемы TM1637.....	55
	Приложение А Вкладка состояния ShIoTiny (ShIoTiny info page).....	56
	Приложение Б Вкладка настройки сети и режимов работы ShIoTiny (Networking).....	58
	Приложение В Работа с редактором ElDraw.....	59
	Приложение Г Соответствие сигналов на плате ShIoTiny-07 и выводов ESP8266.....	64

## 1 Для чего или для кого? Или вместо введения

Словосочетания «умный дом» и «интернет вещей» (от же — IoT) сейчас встречаются настолько часто, что на них уже и не обращают внимания.

Оба словосочетания — корявый перевод на русский язык иностранных терминов и не отражают сути предметов, которые они приняты обозначать.

Умный дом. Почему именно дом? Почему не гараж? Не сарай? Не теплица на даче? Не собачья конура? Не серверная, наконец? Это все так же точно можно автоматизировать!

Интернет вещей. Почему именно «интернет»? Почему не локальная сеть, например?

Всё это — и «умный дом» и «интернет вещей» сводится в одно понятие - «малая автоматизация» или «бытовая автоматизация». То есть, возможность заставить бытовые вещи или устройства самостоятельно выполнять какие-то необходимые функции. А уж дома или на даче, с выходом в интернет или в гордом одиночестве — это не важно.

Исходя из этой концепции и был создан контроллер **ShIoTiny**. Делался он с вполне конкретной целью — автоматизировать освещение и водопровод в загородном доме. Но в итоге получилось то, что получилось. История его создания — это отдельная повесть, о которой я может быть когда-нибудь расскажу.

Кому этот контроллер может пригодиться? Наверное, многим.

Во-первых, **ShIoTiny** пригодится детям и взрослым, которые хотят прикоснуться к магическому миру «умных вещей», но не умеют программировать и изучать языки программирования. То есть для тех, у кого есть минимальное количество знаний и умений или для тех, кто вообще не знают с какого конца взяться за свою идею.

Во-вторых, **ShIoTiny** может вполне подойти для тех, кто хочет облегчить себе какие-то рутинные функции. Например, кто-то хочет сделать управление освещением на улице, дома или в других помещениях; кто-то хочет управлять насосом; кто-то — обогревом теплицы. Фантазии у людей много. А времени, как правило — не очень.

В-третьих, **ShIoTiny** сможет удовлетворить тех, кто хочет без особых проблем создать соединить несколько своих устройств в одну систему. Например, сделать так чтобы свет в соседней комнате включался до того, как вы в неё вошли или чтобы контроллер в подвале «знал», какая температура на улице.

В-четвёртых.. Хотя, стоп! Зачем я буду ограничивать фантазию людей? Пусть делают что хотят и как хотят. А я займусь обзором возможностей контроллера **ShIoTiny**.

## 2 Что может ShIoTiny?

Итак, что же такое **ShIoTiny**? Если коротко — то **ShIoTiny** это один из линейки универсальных контроллеров ShIoT на базе модулей ESP8266. Да, одиночная разработка вылилась в идею нескольких контроллеров, объединенных общей концепцией и философией. Поэтому то, что я расскажу, справедливо в основном для всех контроллеров ShIoT-линейки.

Одноименное название - **ShIoTiny** — имеет и «прошивка» контроллера. Заметьте, что «прошивку» **ShIoTiny** можно использовать не только на модуле ESP-07, на базе которого построен контроллер, но и на модулях ESP-01, ESP-12 и других модулях на базе чипа ESP8266. Так что если будете делать свое устройство — можете использовать прошивку. Автор ни за что не отвечает! :)

Начну с хорошего: отличительной особенностью этих контроллеров является универсальность, широкие коммуникации возможности, а самое главное — система графической конфигурации. Разумеется, что все это — с оговоркой на простоту и дешевизну.

Что это значит? Это значит, что вам, для использования контроллера **ShIoTiny** не нужно скачивать библиотеки из интернета, не нужно изучать SDK и языки программирования. Собственно, вам для программирования контроллера **ShIoTiny** не нужно ничего кроме компьютера, подключенного к сети WiFi.

Ещё одна приятная особенность — это время готовности контроллера, измеряемое секундами. То есть от включения контроллера до того, как он загрузится и ваша программа запустится — проходит несколько секунд. Это пока не по силам «малинкам-апельсинкам» с Linux на борту. Впрочем, у них и возможностей побольше.

Как происходит программирование? Очень просто! В основном - с помощью мышки!

Подключаете **ShIoTiny** к своей WiFi сети (или просто подключаетесь к **ShIoTiny** как к точке доступа, они и это умеет), затем заходите на сайт подключенного **ShIoTiny** — и всё! Программируйте, сколько влезет!

Программа для **ShIoTiny** выглядит примерно так, как показано на рисунке. Сразу оговорюсь, что слова «схема-программа», «схема» и «программа» - это, применительно к **ShIoTiny**, — синонимы, обозначающие графическую схему,

описывающую алгоритм работы устройства. А описание алгоритма работы устройства на каком-то языке программирования — это и есть программа. Поэтому эти слова будут ниже употребляться, как синонимы, если прямо не указано иное.

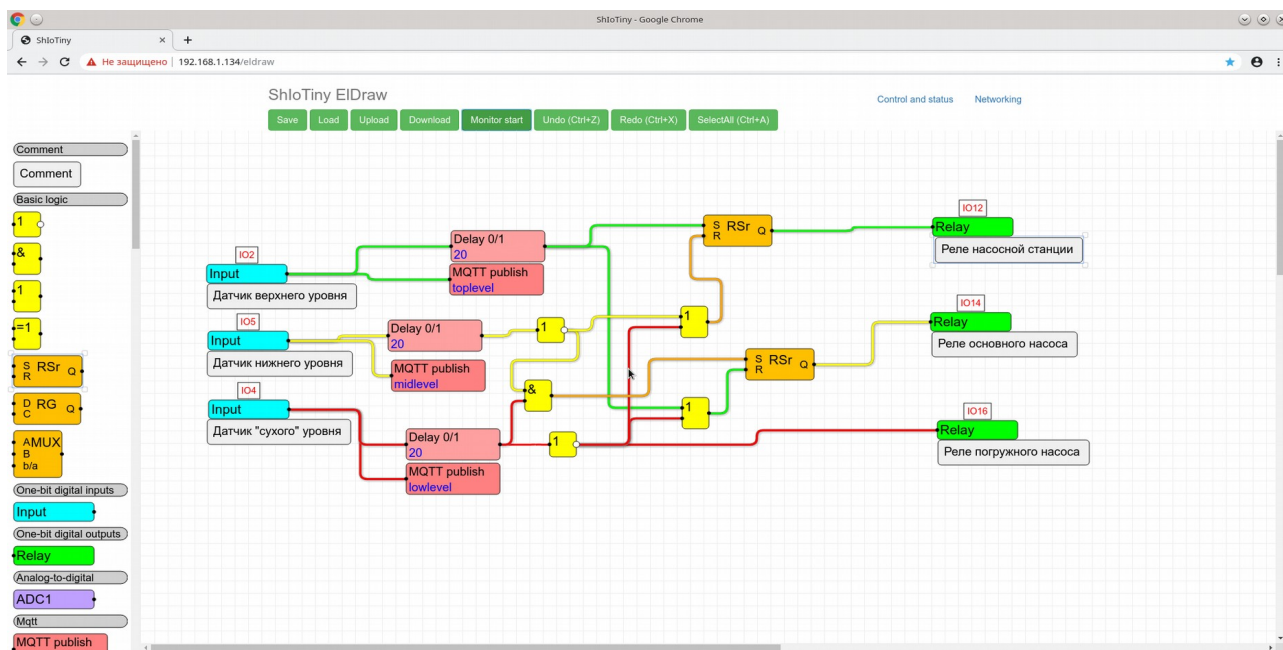


Рис. 1: Пример схемы-программы в редакторе **ShIoT Tiny EIDraw**

Просто и красиво, не правда ли?

Что умеет «программа-схема» на рисунке? Не так уж и мало: она управляет вполне реальной насосной станцией, передает на MQTT-сервер данные об уровне воды, включении и выключении насосов. Кроме того, она может по вашей команде, переданной через интернет, выключить все насосы (аварийное выключение — мало ли что?).

И это всего пара десятков квадратиков! А ведь есть еще встроенный АЦП, есть возможность подключить датчик температуры и влажности. Уже интересно?

### 3 Что нужно для работы?

Собственно для программирования, не нужно ничего кроме ноутбука или компьютера, но обязательно с WiFi.

Питается контроллер от обычного USB-блока питания напряжением 5Вольт. Строго говоря — при питании от 5 до 7 вольт всё будет работать. Но USB так популярно, так доступно и у каждого дома так много ненужных зарядок...

Кроме того, неплохо иметь светодиоды, кнопки, переменное сопротивление и несколько постоянных сопротивлений для имитации датчиков и управляемых устройств.

Подключается все это так, как показано на рисунке ниже. Не забудьте про полярность светодиодов и датчика DHT22 (или DHT11)! Рисунок специально сделан схематичным, чтобы сильно глаза не разбегались. Все нужные подробности подписаны или выделены цветом.

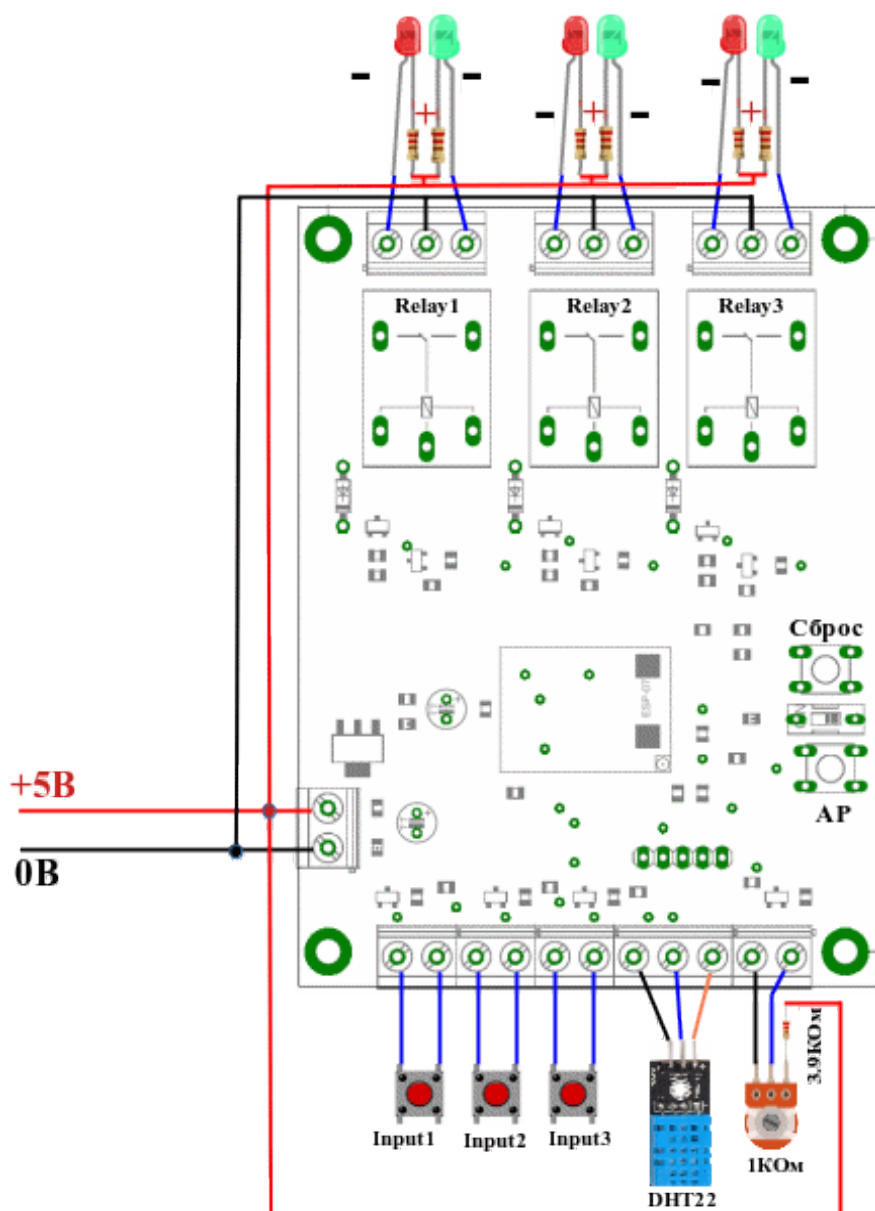


Рис. 2: Подключение датчиков и исполнительных устройств к контроллеру *ShIoTiny*

Три кнопки (Input1,2,3) имитируют датчики типа «сухой контакт». Занимательная история о том, какие контакты - «сухие», а какие «мокрые» - будет ниже. Переменный резистор 1KΩ позволяет изменять напряжение на аналоговом входе. Датчик DHT22 (или DHT11) измеряет температуру и влажность.

Три реле — Relay1,2,3 — управляют тремя красными и тремя зелеными светодиодами. Светодиоды имитируют состояние управляемых устройств —

лампочек, насосов, нагревателей, вентиляторов. Красный светодиод светится, когда реле отключено, а зеленый — когда реле включено.

Кстати, реле работают и при напряжении 220Вольт и токе до 1Ампера, что позволяет подключать реальные устройства мощностью примерно до 200Вт. Это значит, что лампочки (до 200 Вт, конечно, то есть лампочки, а не прожектора!) можно подключать напрямую к данным реле. А вот чайник — не стоит, если не хотите дымно-световых эффектов.

Кроме этого есть две служебные кнопки — «Сброс» (он же **Reset**) и «AP». Функцию кнопки «Сброс», я думаю, пояснять не надо — она все сбрасывает. Ну не совсем всё, конечно, а только перезагружает контроллер.

А вот «AP» — это очень «клевая» кнопка. Если вы ввели неправильные сетевые настройки, то надо нажать на эту кнопку и подержать её около 5 секунд. После чего ненадолго загорится красивый синенький светодиод рядом с кнопкой AP и **ShIoTiny** перезагрузится в режиме конфигурации (то есть будет виден, как открытая точка доступа WiFi). Затем можно подключиться к **ShIoTiny** с любого сотового телефона, планшета или ноутбука и исправить настройки.

*Если все совсем плохо (например программа по каким-то причинам «завешивает» **ShIoTiny**), то на этот случай есть «экстренный выход в конфигурационный режим»!*

*Для «экстренного выхода в конфигурационный режим» вы должны в течении полсекунды после отпускания кнопки **Reset** нажать кнопку **AP**.*

*То есть: нажимаем **Reset**, отпускаем его и тут же, ни мгновения не мешкая, нажимаем **AP**! Загорится на секунду синенький светодиод и погаснет — значит контролер перезагрузился в режиме конфигурации. Не загорится — повторите еще раз.*

Ну и, наконец, последний «орган управления» - переключатель, расположенный между кнопками «Сброс» и «AP». Он предназначен для входа в режим программирования, то есть для замены прошивки контроллера. В обычном, рабочем, режиме этот переключатель находится в состоянии **OFF** (отключено). Для освоения **ShIoTiny** этот переключатель нам не понадобится, но знайте, что прошивки совершенствуются и их можно без труда заменить.



## 4 Как попасть на страницу контроллера ShIoTiny

Итак, у нас есть контроллер **ShIoTiny**. Переведем его в конфигурационный режим (кнопочку AP держим около 5 секунд, пока не загорится красивый синенький светодиод).

Теперь у нас два варианта.

В первом случае (в режиме точки доступа) контроллер **ShIoTiny** сам становится точкой доступа и вы подключаетесь к нему с помощью, например, своего смартфона или ноутбука.

Во втором случае (в режиме WiFi станции) контроллер **ShIoTiny** подключается к какой-либо точке доступа WiFi точно так же как, например, смартфон или планшет. Затем, через локальную сеть или интернет вы можете зайти на **ShIoTiny** со своего компа и запрограммировать его всласть.

На самом деле — режимов работы контроллера **ShIoTiny** в сети аж 5 штук! Поэтому для удобства все они представлены в табличке. Не бойтесь, всё понятно и логично. Выбирается режим на вкладке **Networking** в поле «**ShIoTiny mode**».

№	Режим	Пояснение
1	<b>Config mode</b>	Режим конфигурации. Это режим <i>открытой</i> точки доступа с неизменным адресом <b>192.168.4.1</b> . Сюда попадаем при нажатии на кнопку AP в течении 5 секунд. Схема-программа при загрузке в режиме конфигурации <u>не запускается</u> . Название (SSID) точки доступа — всегда имеет вид <b>esp_8266_xxxxxx</b> , где <b>xxxxxx</b> — серийный номер чипа устройства.
2	<b>Station mode</b>	Режим WiFi-станции, подключенной к вашей точке доступа. В этом режиме <u>запускается при старте устройства автоматически</u> . Название (SSID) точки доступа, к которой вы хотите подключиться вводится в поле <b>Name (SSID)</b> , а пароль в поле <b>Password</b> . Кнопочка <b>ScanWiFi</b> позволяет просканировать сеть и найти доступные точки доступа.
3*	<b>AP mode</b>	Режим <i>закрытой</i> точки доступа. Отличается от <b>Config mode</b> тем, что схема-программа при загрузке в этом режиме <u>запускается при старте устройства автоматически</u> . Кроме того, точка доступа в режиме AP — всегда защищена паролем. Название (SSID) вашей точки доступа и пароль можно изменять. Базовый адрес точки доступа так же можно настроить. В режиме AP: - название (SSID) точки доступа вводится в поле <b>AP Name (SSID)</b> ; - пароль точки доступа вводится в поле <b>AP Password</b> ;

		- IP-адрес точки доступа вводится в поле <b>AP IP address</b> .
4*	<b>AP+Station mode</b>	Это одновременное включение режимов <b>AP mode + Station mode</b> .
5	<b>Single mode</b>	Работа без подключения к сети вообще. Одинокий контроллер, несущий свою тяжкую долю... Если вам нужно что-то автоматизировать без выхода в интернет, то просто выбирайте этот режим. Разумеется все сетевые функции не будут работать.

*\*Задавая пароль и имя точки доступа **AP** в режимах **AP mode** и **AP+Station mode** помните, что пароль должен быть не менее, чем из 8 символов! Иначе точка доступа работать не будет!*

Чтобы было окончательно понятно, что и куда вводить на вкладке Networking, на рисунке ниже изображена форма настройки WiFi сетей и режима работы.

Network mode and access

Name (SSID)  
ssid

Password  
\*\*\*\*\*

AP Name (SSID)  
apssid

AP Password  
\*\*\*\*\*

AP IP address  
192.168.4.1

ShIoT mode: Station mode

Lock Web in Station mode ☐

Scan WiFi

Рис. 3: форма настройки WiFi сетей и режима работы

Чтобы было понятно, что куда и как подключается во всех режимах, смотрите рисунки ниже.

## *ShIoT — сделай сложности проще*



Рис. 4: **ShIoTiny** в режимах точки доступа (Config mode и AP mode)

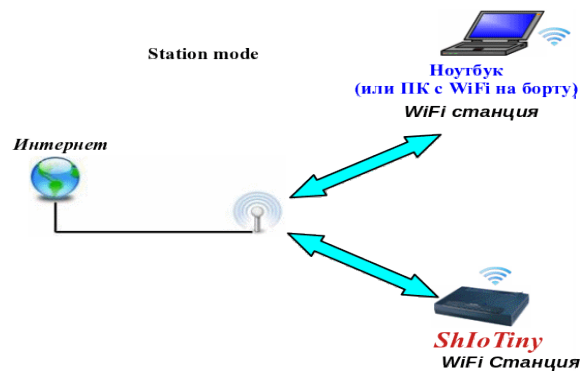


Рис. 5: **ShIoTiny** в режиме WiFi-станции (Station mode)

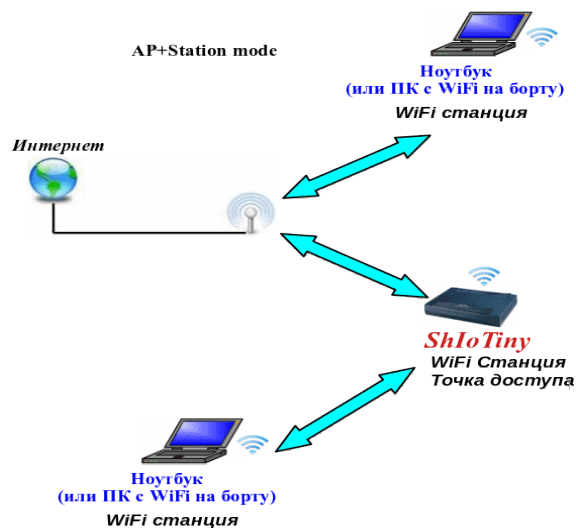


Рис. 6: **ShIoTiny** в режиме точка доступа + WiFi-станция (AP+Station mode)

Надеюсь, теперь, все понятно?

Последовательность настройки **ShIoTiny** ясна как день. Сначала переводим его в режим конфигурации (Config mode) кнопкой AP. Затем, заходим на него с любого смартфона как на точку доступа — и настраиваем его так, как нам надо. Программировать со смартфона не получится. Не удобно это. А вот сетевые настройки ввести — вполне.

Если вы работаете с **ShIoTiny** в режиме точки доступа, то попасть на него просто — вы знаете его IP адрес (в Config Mode — это всегда 192.168.4.1, в AP mode — тот, что вы ввели в поле **AP IP address** вкладки **Networking**). А как быть, если **ShIoTiny** работает в режиме WiFi станции?

Из этой ситуёвины есть выхода два. Первый — это залезть на ваш роутер-точку доступа и посмотреть какие адреса он раздал подключенным устройствам. Но это не всегда удобно.

Другой выход — использовать режим **AP+Station mode**, то есть когда **ShIoTiny** является и станцией и точкой доступа одновременно. В этом случае, вы просто заходите на **ShIoTiny** как на точку доступа с любого смартфона и на вкладке **Control and status** смотрите какой адрес ваш контроллер получил от WiFi точки доступа. Затем вы можете спокойно подсоединяться по этому адресу с любого компьютера в вашей локальной сети и работать.

## 5 Мы попали на страницу ShIoTiny

Как бы там ни было — мы попали на страницу контроллера **ShIoTiny**.

На странице всего три вкладки, из которых видны одновременно только две:

**Control and status** — показывает состояние используемых датчиков и реле, режимы работы WiFi и IP-адреса на вкладке **ShIoTiny info page**. О ней я рассказывать не буду — там все и так ясно.

**Networking** — управление всеми сетевыми настройками: в режиме станции; в режиме точки доступа; подключение к MQTT-серверу и настройки обмена по протоколу UDP между несколькими контроллерами **ShIoTiny**. По мере необходимости расскажу обо всех настройках на этой вкладке.

Особо отмечу, что существует возможность заблокировать сайт контроллера установкой крыжика **Lock Web in Station mode** на этой вкладке. Если этот крыжик установлен, то страница заблокирована во всех режимах, кроме **Config mode**.

**EIDraw** — редактор программ-схем. Это то самое, ради чего собственно и затевалась вся работа по разработке **ShIoTiny**. Можно начинать строить свою систему.

## 6 Первая программа

К сожалению, начать с классического «Здравствуй мир!» нам не светит. У нас в распоряжении лишь числа. Но это не беда. Зато у нас есть возможность при желании запустить реальный фейерверк, например, соединив «поджиг» реальной ракеты с одним из реле.

Итак, заходим на вкладку **EIDraw** и видим примерно то, что уже видели на рисунке в начале обзора. Слева - «палитра элементов». Справа — чистый лист, где будет наша схема-программа. Будем называть дальше поле для рисования схемы (то, что слева) — просто схемой.

**Сверху** — кнопки, назначение которых в основном очевидно.

**Save** — сохранить схему на ваш компьютер.

**Load** — загрузить схему с вашего компьютера в редактор (не на устройство!).

**Upload** — загрузить схему из редактора на устройство.

**Download** - загрузить схему из устройства в редактор. Вообще схема, если она имеется в устройстве, загружается автоматически. Но если вы что-то наисправляли-поменяли, то загрузка схемы из устройства позволяет все ваши исправления отменить.

**Monitor Start/Monitor Stop** — включение-выключение режима монитора. Он очень полезен для отладки схем. При включенном мониторе возле каждого входа и выхода показывается его реальное состояние.

**Undo (CTRL+Z)** — отменить действие редактирования.

**Redo (CTRL+X)** — вернуть действие редактирования.

**Select All (CTRL+A)** — выделить все, что есть на схеме.

Теперь берём в палитре элемент **Input** и мышкой тащим его на лист. Затем берём в палитре элемент **Relay** и тоже мышкой тащим его на лист.

Ну и, наконец, берём выход элемента **Input** (точка в правой части элемента) и соединяем его с входом элемента **Relay** (точка в левой части элемента).

У нас должно получиться примерно так, как на рисунке.

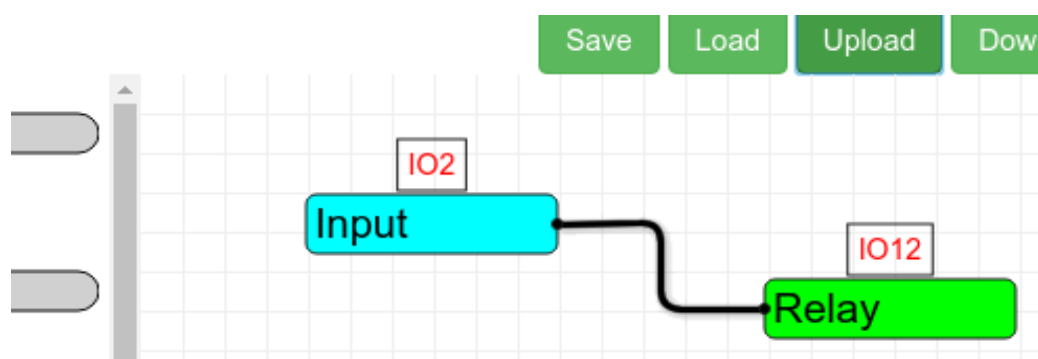


Рис. 7: Наша первая программа

Красные цифры сверху элементов означают номер вывода («ножки», «пина») **ESP8266** с которым связан данный узел (**IO** — **Input/Output**).

Разумеется, что номера выводов можно задавать только для тех узлов, которые связаны с физическими ножками **ESP8266** и которые можно изменять. Если узел не связан ни с какой «ножкой», то вместо номера вывода появится надпись «N/A».

Чтобы настроить номер физической «ножки», надо ткнуть левой кнопкой мыши на узел. Появится диалог выбора «пина», как на рис.8. Узлы, связанные с физическими интерфейсами (пинами) **ESP8266** будем называть узлами ввода-вывода.

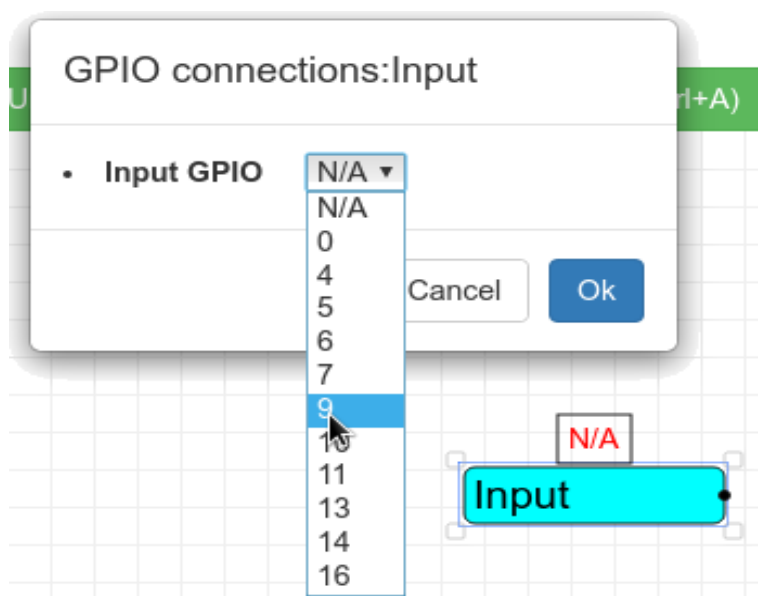


Рис. 8: Диалог выбора пина, привязанного к узлу ввода-вывода

Заметьте, что если какая-то ножка уже используется другим узлом на схеме, она будет недоступна для выбора. Чтобы отключить узел от ножек выберите пункт **N/A**.

Помните, что в устройство можно загружать только схему, у которой нет узлов с неопределенными (N/A) ножками! Но на диск выгружать такие схемы можно.

**ВНИМАНИЕ!** Выводы **GPIO9** и **GPIO10** можно выбирать только на модулях, на которых они не используются для связи с FLASH! В частности, на плате **ShIoTiny** на базе модуля **ESP-07** эти ножки выбирать нельзя: контроллер будет сбоят и перезагружаться! Если вы случайно выбрали **GPIO9** или **GPIO10**, то вам поможет «экстренный выход в конфигурационный режим» уже описанный в разделе 3.

Всё! Первая схема готова! Как нам её запустить? Очень просто! Давим кнопку Upload и на вопрос «*Are you sure to uploading scheme to device?*» («Вы уверены, что хотите загрузить схему в устройство?») отвечаем «ОК».

Если все успешно загрузилось, то появится окно-подтверждение с надписью «*Scheme uploaded to device (2 nodes).*» («Схема загружена в устройства (2 узла).»)

Всё! Первая схема работает! Проверить это проще простого: нажмите на кнопку «**Input1**» и реле «**Relay1**» послушно щелкнет и включит зеленый светодиод. Если кнопку отпустить — реле снова щелкнет и зеленый светодиод погаснет, а красный — вновь зажжется!

Подробно о номерах ножек смотрите в Приложение Г Соответствие сигналов на плате ShIoTiny-07 и выводов ESP8266.

Обратите внимание, что если вы находитесь в режиме конфигурации, то после перезагрузки схема не будет работать, пока ее вновь не загрузят на устройство кнопкой **Upload**. Во всех остальных режимах — схема стартует автоматически при включении **ShIoTiny** или после его перезагрузкой при помощи кнопки «**Сброс**».

Далее мы будем часто обозначать узлы, как это было в старых прошивках — «**Input1**» или «**Relay3**», имея ввиду то, что вы сами привяжете их к нужным «ножкам».

## 7 Усложняем задачу

Щелкать реле по нажатию кнопки — это прикольно и весело. Но для этого, конечно, не обязательно возиться с контроллером.

Попробуем создать интересную схему, которую назовем «один из двух».

Кстати, если вы хотите раскрасить линии связи между элементами — то просто надавите правой кнопкой мыши на нужную связь: выскочит меню, где можно выбрать цвет отдельной линии связи или всей группы связей, соединенной с одним выходом.

Выходы — это точки справа. Выходы отдают информацию. Входы — это точки слева. Входы принимают информацию. Это, конечно, упрощенно, но для начального понимания работы схем — достаточно.

Соединять можно только вход с выходом или выход со входом. Но не выход с выходом, и не вход со входом!

Обратите внимание, что к **выходу** можно подсоединить *много связей*. Но ко **входу** можно подсоединить только *одну связь*.

Итак, что же делает схема «один из двух»? Она позволяет включить только одно реле из двух.

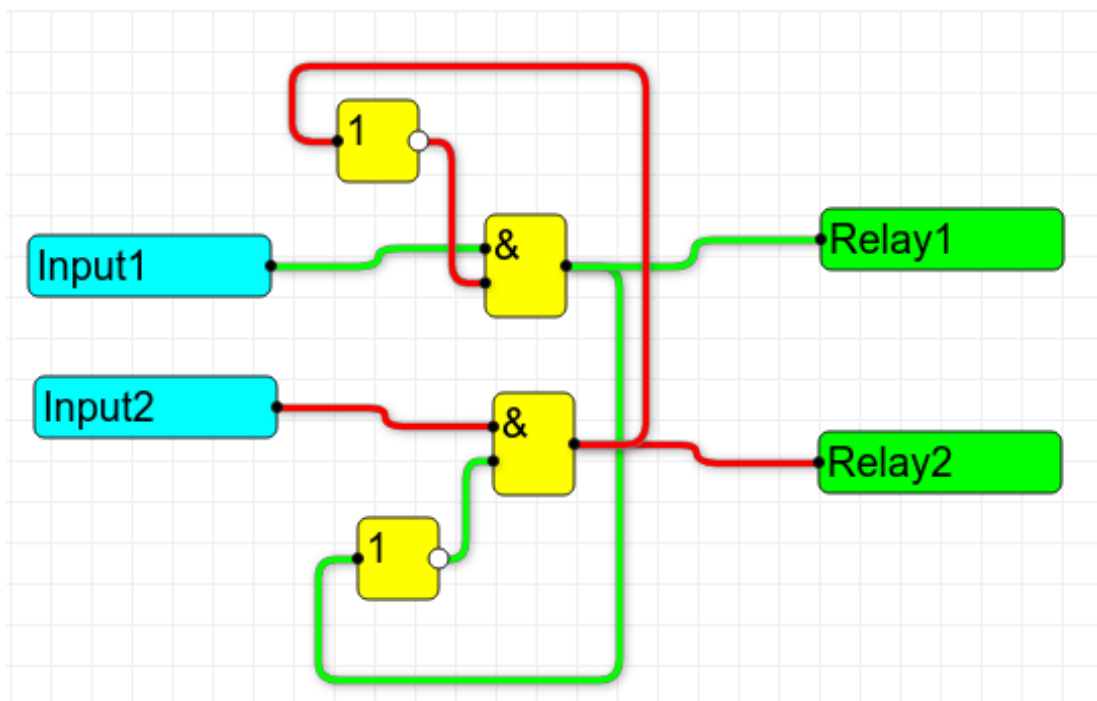


Рис. 9: Наша вторая программа



Нажмите кнопку **Input1**. Включится реле **Relay1**. Ничего особенного. Теперь, удерживая кнопку **Input1**, (реле **Relay1** включено!) нажмите кнопку **Input2**. Ничего не произошло. Реле **Relay2** заблокировано!

Если же отпустить кнопку **Input1**, то по нажатию кнопки **Input2** реле **Relay2** будет исправно срабатывать.

И, наоборот, если вначале нажать кнопку **Input2** и удерживать ее, то реле **Relay2** будет включено, а реле **Relay1** нажатием кнопки **Input1** включить не удастся, пока кнопка **Input2** остается нажатой.

Эту схему можно еще назвать «*кто успел, тот и съел*», так как какая кнопка нажата раньше, такое реле и срабатывает.

Кстати, эту схемы можно использовать как игру «Кто быстрее?». Каждому игроку дается по кнопке и по сигналу игроки нажимают эти кнопки. Кто быстрее нажал, у того и загорится зеленый светодиод.

Подумайте, как расширить эту схему до трех кнопок и трех реле сами.

## 8 Монитор

Контроллер **ShIoTiny** предоставляет встроенный монитор для отображения состояния схемы. Что это такое? Все очень просто. Если нажать на кнопку «Monitor start», то монитор включится и возле каждого входа и выхода каждого элемента появится его текущее состояние, примерно как на рисунке ниже.

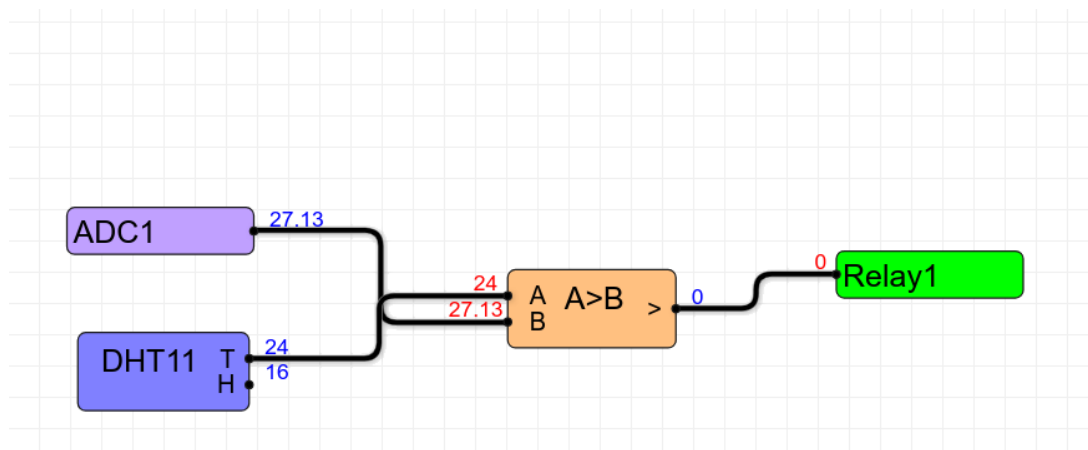


Рис. 10: В режиме монитора возле входов и выходов каждого узла-элемента появляется его состояние

Состояние элементов обновляется со скоростью примерно 10 элементов в секунду. Это не очень быстро, но позволяет подробно увидеть как изменяется состояние комбинационных схем.

В случае удаления связи, элемента или добавления элемента на схему, монитор автоматически отключается.

Заметьте, если вы не загрузили схему в устройство, то есть в устройстве и на листе у вас разные схемы — то монитор будет показывать что попало, поэтому перед запуском монитора настоятельно советую давить кнопку «Download».

## 9 Наигрались? Теперь давайте по порядку

Все это красиво и хорошо, но чтобы творить по-настоящему, необходимо немного систематизировать свои знания. Этим мы сейчас и займемся.

### 9.1 Узлы, связи и события

Итак, что представляет собой схема-программа? Достаточно взглянуть на приведенные примеры, чтобы понять, что состоит схема лишь из двух сущностей — **узлов** (или элементов) и **связей** между ними.

**Узел** или **элемент схемы** — это виртуальное представление какого-то **действия** над данными. Это может быть арифметическая операция, логическая операция или вообще какая угодно операция, какая придет нам в голову. Главное, что у узла есть **вход** и **выход**.

**Вход** — это то место, куда узел **принимает** данные. Изображения входов — это точки, находящиеся всегда с левой стороны узла.

**Выход** — это то место, откуда **извлекается** результат работы узла. Изображения выходов — это точки, находящиеся всегда с правой стороны узла.

У некоторых узлов нет входов. Такие узлы генерируют результат внутри себя. Например узел-константа или узел-датчик: им не нужны данные от других узлов, чтобы сообщить результат.

У других узлов, напротив, нет выходов. Это узлы, отображающие, например, исполнительные устройства (реле или еще какие-нибудь подобные). Они принимают данные, но не генерируют результата вычислений, доступного для других узлов.

Кроме того, есть еще уникальный узел-комментарий. Он ничего не делает, не имеет ни входов ни выходов. Его назначение — быть пояснением на схеме.

### 9.2 Какие бывают входы и выходы?

Входы и выходы бывают разные в том смысле, что данные, передаваемые от узла к узлу у нас могут быть нескольких типов: бинарные (0 и 1), целые и вещественные.

Бинарные данные — это фактически представление в цифровом виде логической величины. То есть такой величины, которая имеет только два значения: ложь (0) и истина (1).

Многие логические элементы работают именно с таким представлением данных на своих входах. С выходами всё понятно: бинарный выход может

отдавать только 0 и 1; целый выход — любое целое число в диапазоне  $[-(2^{31})...(2^{31})-1]$  и, наконец, вещественный выход — любое число в диапазоне  $[-3.4 \cdot 10^{38} .. +3.4 \cdot 10^{38}]$ , а ещё NaN (не число) и Inf (бесконечность).

А можно ли подавать на бинарные входы целые или вещественные числа? Разумеется, можно. Нужно только знать несколько незатейливых правил преобразования вещественных, целых и бинарных чисел друг в друга.

В общем случае надо помнить, что любое целое ненулевое значение бинарный выход воспринимает как 1 (истину). А нулевое — как 0 (ложь). С вещественными числами сложнее. Но таблица ниже поможет прояснить понять, как преобразуются целые, бинарные и вещественные числа друг в друга.

Или, иными словами, таблица показывает как входы и выходы различных типов воспринимают друг друга.

	Вещественное	Целое	Бинарное
1	0	0	0 (ложь)
2	$N \in [-(2^{31})...(2^{31})-1]$ ( кроме 0, NaN и Inf )	N (округлённое до целого)	1 (истина), если N, округленное до целого не равно нулю; 0 (ложь), если N, округленное до целого равно нулю.
3	NaN (не-число)	0	0 (ложь)
4	Inf (бесконечность)	$(2^{31})-1$	1 (истина)

В описании узлов, везде помечено, какой тип выхода и входа использует конкретный узел: вещественный/целый или бинарный. Разные входы и выходы одного и того же узла могут иметь разные типы.

### 9.3 События: что это такое и с чем их подают к столу

Данные от узла к узлу передаются не просто так, а при возникновении *события*.

Что такое «*событие*»? Событие — это возникновение новых данных в каком-либо узле. Например к событиям относятся: изменение состояния входа (узел **Input**), приём данных от другого устройства (узлы **MQTT** и **UDP**), истечение заданного промежутка времени (узлы **Timer** и **Delay**) и так далее.

Для чего нужны события? Да для того, чтобы определить — в каком узле возникли новые данные и состояния каких узлов необходимо изменить в связи с

получением новых данных. Событие, как бы «проходит» по цепочке узлов, пока не обойдет все узлы, состояние которых необходимо проверить и изменить.

Все узлы можно подразделить на две категории.

Узлы, которые могут генерировать события назовем «**активные узлы**».

Узлы, которые не могут генерировать события назовем «**пассивные узлы**».

Когда узел генерирует событие (то есть у него на выходе появляются новые данные), то изменяется в общем случае состояние всей цепочки узлов, подключенных к выходу узла-генератора события.

Чтобы было понятно, рассмотрим пример на рисунке.

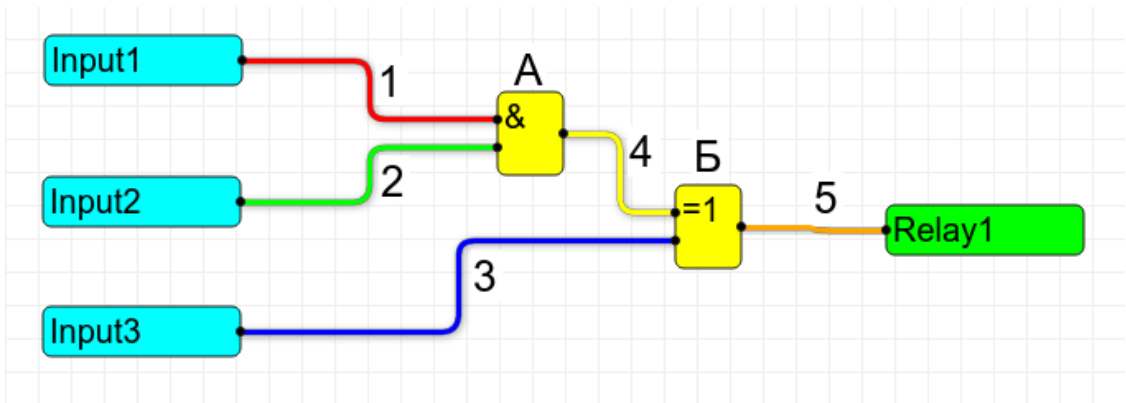


Рис. 11: Прохождение события по узлам

Активные узлы тут — **Input1**, **Input2** и **Input3**. Остальные узлы — пассивные. Рассмотрим что происходит, когда замыкается тот или иной вход. Результаты для удобства сведены в табличку.

	Узел	Путь события	Примечание
1	<b>Input1</b>	1 — 4 — 5	Изменены состояния узлов А и Б
2	<b>Input2</b>	2 — 4 — 5	Изменены состояния узлов А и Б
3	<b>Input3</b>	3 — 5	Изменено состояние узла Б

Как видим, при возникновении события, строится цепочка от узла-источника события и до окончного узла. Состояние тех узлов, которые не попадают в цепочку, не изменяется.

Возникает законный вопрос, а что же будет при возникновении двух или вообще нескольких событий *одновременно*?

Как любителя творчества Глеба Анфилова, меня так и тянет отправить любопытного вопрошателя к его книге «*Бегство от удивлений*». Это такая «теория относительности для самых маленьких», в которой хорошо рассказано, что такое «одновременно» и как с этим жить.

Но чисто практически все гораздо проще: при возникновении двух или вообще нескольких событий последовательно строятся и обрабатываются все цепочки от каждого источника события по очереди и никаких чудес.

Следующий вполне законный вопрос любопытного читателя — а что будет, если узлы соединить в кольцо? Или, как это принято говорить среди этих ваших умников — *ввести обратную связь*. То есть соединить выход одного из узлов со входом предыдущего узла так, чтобы состояние выхода этого узла влияло на состояние его же входа. Напрямую соединить выход узла с его же входом вам не позволит редактор ElDraw. А вот опосредованно, как на рисунке ниже — это сделать можно.

Итак, что же будет в этом случае? Ответ будет очень «определенный»: смотря какие узлы. Рассмотрим пример на рисунке.

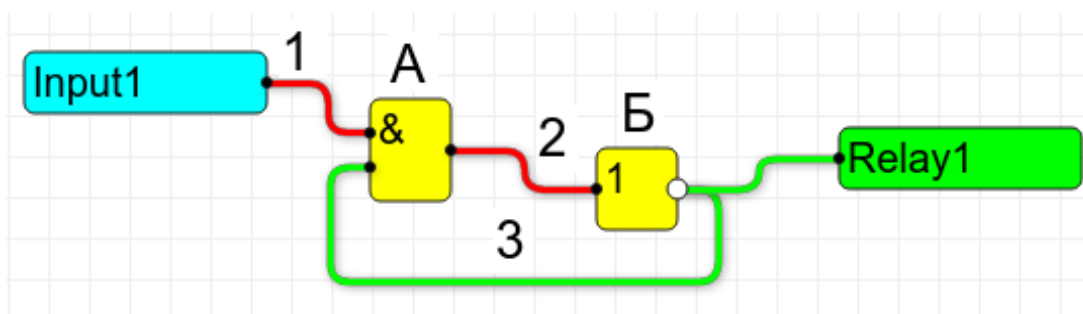


Рис. 12: Соединим узлы в кольцо, но не получим генератор!

Когда контакты входа Input1 разомкнуты на верхнем входе узла А — 0. На выходе узла А, тоже 0. На выходе узла Б — 1. И, наконец, на нижнем входе узла А — 1. Всё ясно. А кому не ясно — посмотрите ниже описание того, как работают узлы «И» и «НЕ».

Теперь замкнем контакты входа Input1, то есть подадим на верхний вход узла А единицу. Те, кто знаком с электроникой, знает, что фактически мы получим классическую схему генератора на логических элементах. И по идее, такая схема должна бесконечно выдавать на выходе элементов А и Б последовательности 1-0-1-0-1-0.... и 0-1-0-1-0-1-.... Ведь событие должно постоянно изменят состояние узлов А и Б, бегая по кругу 2-3-2-3-... !

Но на самом деле этого не происходит. Схема впадет в случайное состояние — или реле останется включенным или отключенным, а может и слегка прожужжать включаясь-отключаясь несколько раз подряд. Все зависит от погоды на южном полюсе Марса. И вот почему так происходит.

Событие с узла **Input1** изменяет состояние узла А, потом узла Б и так по кругу несколько раз. Программа определяет «зацикливание» события и принудительно прекращает этот карнавал. После этого изменение состояния узлов А и Б блокируются до возникновения нового события. Момент, в который программа решит - «хватит вертеться по кругу!» - в общем случае зависит от многих факторов и его можно считать случайным.

*Будьте осторожны, соединяя узлы в кольцо — эффекты будут не всегда очевидными!*

*Хорошо представляйте что и зачем вы делаете!*

А можно ли все же построить генератор на доступных нам узлах? Да, можно! Но для этого необходим узел, который сам умеет генерировать события. И такой узел есть — это «линия задержки». Посмотрим как работает генератор с периодом 6 секунд на рисунке ниже.

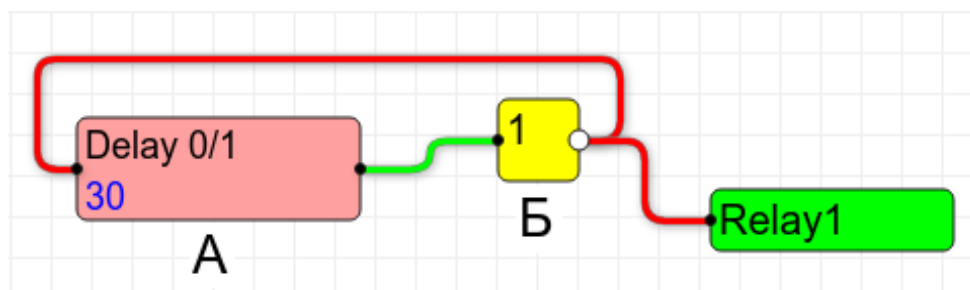


Рис. 13: Генератор с периодом 6 секунд

Ключевым элементом генератора является узел А — линия задержки. Если изменить состояние входа линии задержки с 0 на 1, то 1 на выходе появится не сразу, а только спустя заданное время. В нашем случае это 3 секунды. Точно так же, если изменить состояние входа линии задержки с 1 на 0, то 0 на выходе появится спустя те же 3 секунды. Время задержки задается в десятых долях секунды. То есть значение 30 и означает — 3 сек.

Особенностью линии задержки является то, что она генерирует событие после истечения времени задержки.

Предположим, что изначально на выходе линии задержки был 0. После прохождения узла Б — инвертора — этот 0 превращается в 1 и поступает на вход линии задержки. Сразу ничего не происходит. На выходе линии задержки как был 0 так и останется, но зато включается отсчет времени задержки. Проходит 3 секунды. И тут же линия задержки генерирует событие. На выходе у нее появляется 1. Эта единица после прохождения узла Б — инвертора — превращается в 0 и поступает на вход линии задержки. Проходит ещё 3 секунды... и процесс повторяется. То есть каждые 3 секунды состояние выхода линии задержки меняется с 0 на 1 и затем с 1 на 0. Реле щелкает. Генератор работает. Период импульсов составляет 6 секунд (3 секунды на выходе ноль и 3 секунды — единица).

Но, в реальных схемах, обычно не нужно использовать и этот пример. Существуют специальные узлы-таймеры, которые великолепно и без посторонней помощи генерируют последовательность импульсов с заданным периодом. Длительность «нуля» и «единицы» в этих импульсах равны половине периода.

*Для задания периодических действий, используйте узлы-таймеры.*

Отмечу, что такие цифровые сигналы, где длительность «нуля»  $\tau_0$  и «единицы»  $\tau_1$  равны, называются «меандром».

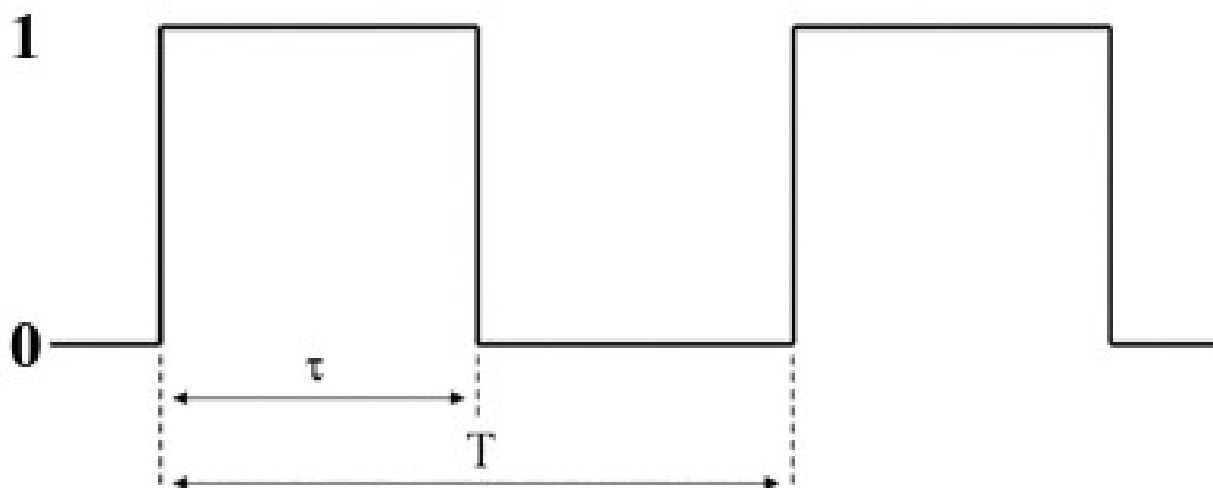



Рис. 14: Меандр. Длительность "нуля" равна длительности "единицы" и равны половине периода  $T$ .  $\tau = \tau_0 = \tau_1 = \frac{1}{2} \cdot T$

Теперь нам ясно как происходит распространение событий между узлами и чего делать не стоит. Рассмотрим теперь, какие узлы нам доступны и как они работают.



## 10 Описание узлов (нод)

### 10.1 Просто комментарий



Информация, комментарии (Comment)		
		<p>Это просто комментарий. Он ничего не делает.</p> <p>Чтобы написать свой комментарий — перетащите узел на схему и кликните по нему мышкой.</p>





### 10.2 Базовая логика (Basic logic)

Базовая логика — это то самое, что в цифровой схемотехнике называют «логическими элементами». Они являются основой основ цифровой логики. Опиерируют эти элементы только двумя цифрами 0 и 1, то есть бинарными данными.

Ниже приведена таблица, которая ясно дает понять — что будет у элемента на выходе при тех или иных значениях на входах.

Помимо базовых элементов, в этот раздел внесены ещё и регистры, триггеры и мультиплексоры. В них используются не только бинарные но и целые/вещественные входы и выходы.

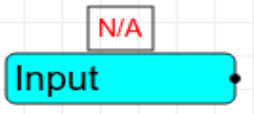
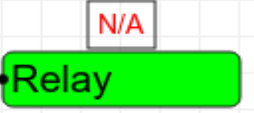
Узел (нода)	Событие	Примечание									
Основные логические элементы (Basic logic). Входы и выходы — бинарные.											
		Логический элемент НЕ (отрицание) <table><tr><th>Вход</th><th>Выход</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Вход	Выход	0	1	1	0			
Вход	Выход										
0	1										
1	0										
		Логический элемент И <table><tr><th>Вход1</th><th>Вход2</th><th>Выход</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	Вход1	Вход2	Выход	0	0	0	0	1	0
Вход1	Вход2	Выход									
0	0	0									
0	1	0									

		<table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	1	1	1														
1	0	0																				
1	1	1																				
		<p>Логический элемент ИЛИ</p> <table><tr><td>Вход 1</td><td>Вход 2</td><td>Выход</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Вход 1	Вход 2	Выход	0	0	0	0	1	1	1	0	1	1	1	1					
Вход 1	Вход 2	Выход																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	1																				
		<p>Логический элемент ИСКЛЮЧАЮЩЕЕ ИЛИ</p> <table><tr><td>Вход 1</td><td>Вход 2</td><td>Выход</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Вход 1	Вход 2	Выход	0	0	0	0	1	1	1	0	1	1	1	0					
Вход 1	Вход 2	Выход																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	0																				
		<p>Триггер RS</p> <table><tr><th>Вход S</th><th>Вход R</th><th>Выход Q</th><th>Примечание</th></tr><tr><td>0</td><td>0</td><td><math>Q_{t-1}</math></td><td>Хранение</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Установка 0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Установка 1</td></tr><tr><td>1</td><td>1</td><td><math>Q_{t-1}</math></td><td>Хранение</td></tr></table> <p><math>Q_{t-1}</math> — предыдущее состояние</p>	Вход S	Вход R	Выход Q	Примечание	0	0	$Q_{t-1}$	Хранение	0	1	0	Установка 0	1	0	1	Установка 1	1	1	$Q_{t-1}$	Хранение
Вход S	Вход R	Выход Q	Примечание																			
0	0	$Q_{t-1}$	Хранение																			
0	1	0	Установка 0																			
1	0	1	Установка 1																			
1	1	$Q_{t-1}$	Хранение																			
Основные логические элементы (Basic logic) Входы и выходы разнотипные																						
		<p>Регистр. Вход <b>D</b> и выход <b>Q</b> — <i>целые/вещественные</i>. <b>C</b> — <b>бинарный вход</b></p> <table><tr><th>Вход D</th><th>Вход C</th><th>Выход Q</th><th>Примечание</th></tr><tr><td>X</td><td>1</td><td>X</td><td>Установка Q</td></tr><tr><td>X</td><td>0</td><td><math>X_{t-1}</math></td><td>Хранение</td></tr></table> <p>Когда <math>C \neq 0</math>, <math>Q=D</math>. Когда <math>C=0</math>, данные на выходе D не</p>	Вход D	Вход C	Выход Q	Примечание	X	1	X	Установка Q	X	0	$X_{t-1}$	Хранение								
Вход D	Вход C	Выход Q	Примечание																			
X	1	X	Установка Q																			
X	0	$X_{t-1}$	Хранение																			

		изменяются (хранение)															
<div><div><div>A MUX</div><div>B</div><div>b/a</div></div></div>		<p>Мультиплексор. Входы <b>A</b>, <b>B</b> и <b>выход</b> — <i>целые/вещественные</i>.</p> <p><b>b/a</b> — бинарный вход.</p> <table><tr><th>Вход A</th><th>Вход B</th><th>Вход b/a</th><th>Выход</th><th>Примечание</th></tr><tr><td>A</td><td>B</td><td>0</td><td>A</td><td></td></tr><tr><td>A</td><td>B</td><td>1</td><td>B</td><td></td></tr></table>	Вход A	Вход B	Вход b/a	Выход	Примечание	A	B	0	A		A	B	1	B	
Вход A	Вход B	Вход b/a	Выход	Примечание													
A	B	0	A														
A	B	1	B														

### 10.3 Простейшие входы и выходы (One-bit digital inputs, One-bit digital outputs)

«Простейшие», это значит имеющие два состояния — включено/выключено, замкнуто-разомкнуто. Такие входы и выходы еще называют «бинарными».

Узел (нода)	Событие	Примечание
<b>Простейшие входы и выходы (One-bit digital inputs, One-bit digital outputs)</b>		
	ДА	<p><b>Бинарный выход.</b></p> <p>Дискретные входы, рассчитанные на подключение датчика типа «сухой контакт». То есть — кнопки, выключателя, геркона и т.п.</p> <p>На выходе узла <b>Input</b> «0», если контакты разомкнуты и «1», если контакты замкнуты.</p> <p><i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i></p>
		<p><b>Бинарный вход.</b></p> <p>Реле — это один из видов выхода типа «сухой контакт». То есть металлические провода, которые могут быть замкнуты или разомкнуты по нашему желанию.</p> <p>У <b>ShIoTiny</b> есть три абсолютно идентичных переключающих реле, рассчитанных на нагрузку до 250В 1А.</p> <p>Если на вход узла <b>Relay</b> подать «1», то реле включится, если подать «0» — то выключится.</p> <p><i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i></p>

**Лирическое отступление первое. Про «сухие контакты».** «Сухой контакт» — это контакт, который не имеет собственного источника напряжения. То есть просто два любых металлических проводника, которые можно замкнуть между собой и разомкнуть их. Под это определение попадают масса датчиков — кнопки, выключатели, поплавковые датчики уровня жидкости, герконы (датчики магнитного поля) и так далее. Очевидно, что все эти датчики, можно подключать ко входам контроллера **ShIoTiny**. На электрических схемах нормально разомкнутые «сухие контакты» обычно обозначаются так, как показано на рисунке.



Нормально разомкнутые — это значит, разомкнутые при отсутствии внешнего воздействия — кнопка не нажата, выключатель не включен, у геркона нет поблизости магнита...

Есть также и нормально замкнутые «сухие контакты». На электрических схемах они обычно обозначаются так, как показано на рисунке.

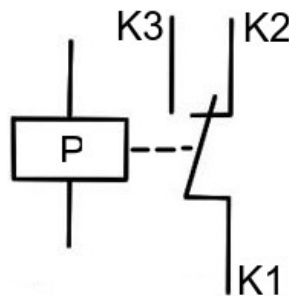


В противоположность нормально разомкнутым, нормально замкнутые «сухие контакты» в отсутствии внешнего воздействия — замкнуты.

Какие именно контакты использовать — решать вам. И те и другие «сухие контакты» можно спокойно подключать ко входам Input контроллера **ShIoTiny**.

**Лирическое отступление второе. Про реле.** Реле бывают разные — нет, не по цвету, а по назначению и характеристикам. Бывают включающие, выключающие, поляризованные, тактовые... Много их, в общем.

В контроллере **ShIoTiny** установлены три переключающих реле. Переключающие, это значит, что у них есть три контакта: один общий, один нормально разомкнутый с общим и один — нормально замкнутый с общим. Условно это показано на рисунке ниже.



Когда питание на обмотку реле не подается (контроллер вообще выключен или на вход узла Relay подается 0) — замкнуты контакты K1 и K2. Такие контакты реле, которые замкнуты в отсутствие питания на его обмотке и называются «нормально замкнутые». Если подать питание на обмотку реле (в нашем случае это значит, подать 1 вход узла Relay), то контакты K1 и K2 разомкнутся, а контакты K1 и K3 — замкнутся. Контакты K1 и K3 называются

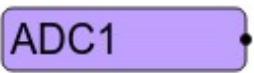
«нормально разомкнутыми», потому что в отсутствие питания на обмотке реле они разомкнуты.

Надеюсь, что теперь все ясно с контактами и реле.

## 10.4 Аналого-цифровой преобразователь АЦП, он же ADC

Аналого-цифровой преобразователь — основа основ цифровых измерений.

Что он делает? На вход АЦП подается напряжение в диапазоне от 0 до 1В. На выходе АЦП мы получаем число, пропорциональное входному напряжению. То есть, если на вход подать 0В — то с выхода АЦП получим 0. Если на вход АЦП подать 1В — то на выходе получим число 1023.

Аналого-цифровой преобразователь АЦП, он же ADC		
	ДА	<p><b>Вещественный выход.</b>  Аналого-цифровой преобразователь (аппаратный).  Значение на выходе вычисляется по формуле:</p> $X = k \cdot u_{\text{вх}} + b$ <p>где <math>u_{\text{вх}}</math> — напряжение на входе ADC (от 0 до 1В);  <b>k</b> — диапазон (<b>ADC range</b>) и <b>b</b>-смещение (<b>ADC offset</b>).</p> <p>Если напряжение больше 1В, то на выходе устанавливается значение NaN (не число).</p>

Число 1023 — максимальное, потому что АЦП **ShIoTiny** — 10разрядный. То есть имеет выходной диапазон значений от 0 до  $2^{10}-1$ . Если мы подадим на вход АЦП напряжение больше 1Вольта, то с выхода будет считываться значение с 1 в разряде ошибки переполнения. В случае ошибки, аппаратный АЦП выставляет значение 1024 в своем выходном регистре.

Все это касается аппаратного АЦП. Узел АЦП немного более сложен. Во-первых, он пересчитывает измеренное значение к диапазону 0..1. Во-вторых, он масштабирует это значение заданным коэффициентом **k** и сдвигает его на величину **b**. Зачем это надо? Очень просто: нам удобнее пересчитать входное напряжение сразу в привычные нам величины и работать уже с ними, а не с абстрактными «попугаями» - числом 0..1023.

Кроме того, если к АЦП подключен какой-то аналоговый датчик, например температуры или уровня жидкости — то можно сразу пересчитать измеряемое им значение в градусы или сантиметры.

Итак, как работает узел АЦП? Рассмотрим по шагам.

**Шаг первый.** Аппаратный АЦП измеряет входное напряжение и получает число 0..1024. (Помним, что 1024 — это переполнение АЦП!)

Если определено переполнение АЦП (подано напряжение выше 1В на вход), то на выходе модуля выставляется значение NaN (не число) и цикл изменений завершается.

Иначе, если все в порядке и никаких переполнений нет, то измеренное значение (0..1023) приводится к диапазону 0..1 (то есть число, считанное с АЦП просто делится на 1023). Это приведенное значение и называется  $u_{\text{вх}}$  в наших формулах.

**Шаг второй.** Вычисляется выходное значение узла АЦП по формуле:  $X = k \cdot u_{\text{вх}} + b$ . Обратите внимание, что коэффициенты  $k$  и  $b$  можно задавать больше единицы, меньше единицы, положительными или отрицательными. Так мы можем пересчитывать измеренное значение практически в любой нужный нам линейный диапазон.

**Шаг третий.** Если полученное значение  $X$  отличается от предыдущего измеренного значения  $X_{t-1}$  на величину, превышающую заданный процент диапазона  $k$ , то генерируется событие. О событиях мы поговорим в другом месте. Пока достаточно знать, что узел АЦП не реагирует на изменение измеренной величины  $X$ , пока изменение не станет больше заданного процента диапазона  $k$ .

Как нам задать все эти величины? Очень просто. Перетащите узел АЦП на схему и ткните на него мышкой. Появится диалог настройки АЦП.

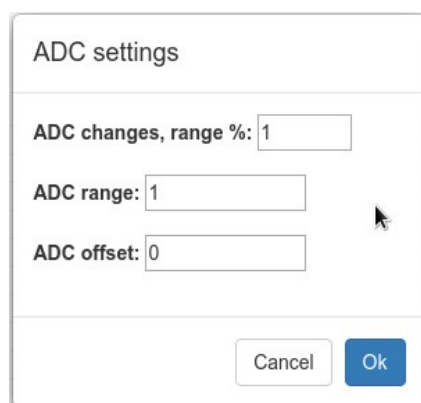


Рис. 15: Диалог настройки узла АЦП

Коэффициент  $k$  вводится в полк **ADC range** (диапазон). Коэффициент  $b$  вводится в полк **ADC offset** (смещение).

Минимальный процент изменения величины, при которой узел АЦП начинает реагировать на эти изменения задаётся в поле **ADC changes, range**.



Для чего нам надо задавать процент изменения? Да чтобы слишком часто не реагировать на события от АЦП.

Запутались? Я понимаю, сам запутался!

Для наглядности приведу пример. Например, мы ходим измерять напряжение в диапазоне от 0 до 18 вольт. Но на вход АЦП нам можно подавать только не более 1Вольта. Как быть? Конечно же — ставим на вход делитель. На рисунке это показано. Слева, на вход делителя подается напряжение в диапазоне от 0 до 18Вольт. Делитель его делит на 18 и подает на вход АЦП.

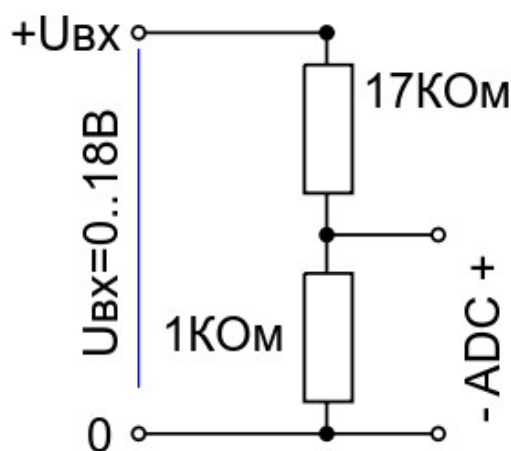


Рис. 16: Делитель напряжения на 18

Все, задача деления напряжения решена — на входе АЦП у нас теперь не больше 1Вольта. Но мы хотим, чтобы с выхода узла АЦП считывалось число 18 при 18Вольтах напряжения на входе делителя! Зачем нам какой-то 1Вольт?

Этому горю несложно помочь. Заходим в диалог настройки АЦП и устанавливаем  $k$  (**ADC range**) равным 18; устанавливаем  $b$  (**ADC offset**) равным 0.

Остается поле **ADC changes, range**. Каким его выбрать? Смотрите сами. Если это поле выставить в 1%, то АЦП будет реагировать на изменение напряжения  $\frac{18 \text{ Вольт}}{100\%} \cdot 1\%$  или 0.18Вольта, а если 10%, то на изменение напряжения  $\frac{18 \text{ Вольт}}{100\%} \cdot 10\%$  или 1.8Вольта. Выбор за вами.

Меньше 1% и больше 100% выставить нельзя. Обычно, для измерения напряжения питания достаточно реагировать на 5%-е изменение этого напряжения.

## 10.5 Связь по протоколу MQTT

MQTT (англ. message queuing telemetry transport) — упрощенный сетевой протокол, работающий поверх TCP/IP, ориентированный для обмена сообщениями между устройствами по принципу издатель-подписчик.

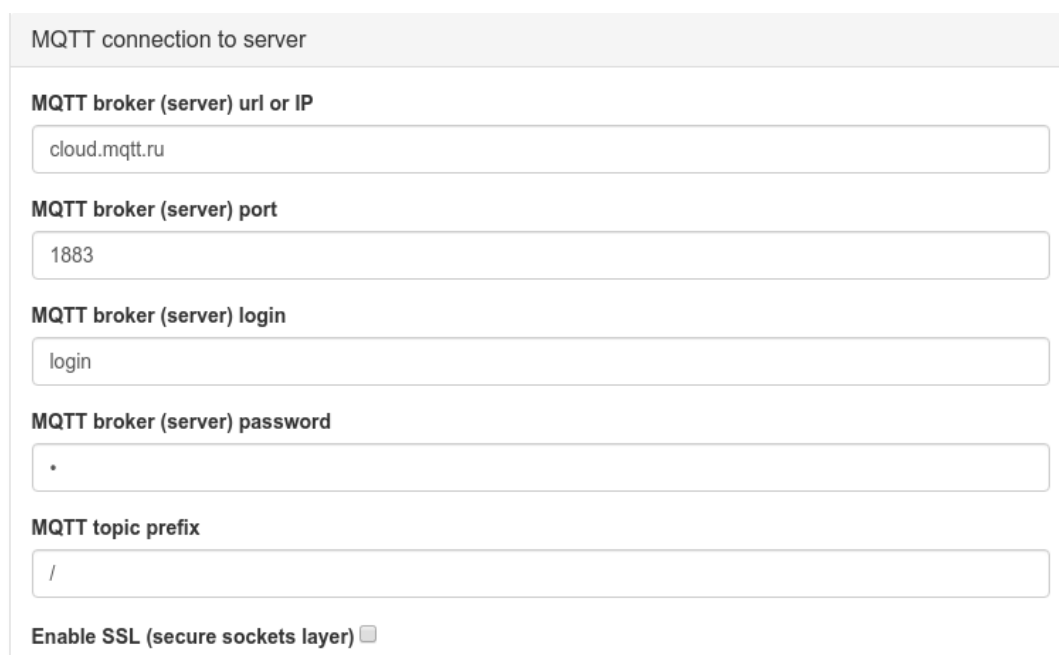
MQTT это святой грааль IoT (интернета вещей). На него сегодня молятся и уже кое-где начали приносить жертвы:)

Суть MQTT проста: одно устройство подписывается на некоторые данные (говорит серверу — какие данные ему нужны), а другое устройство — публикует эти данные. Разумеется, чтобы устройство могло публиковать данные или подписываться на них, оно должно сначала авторизоваться на сервере. Узлы для работы с протоколом MQTT показаны в таблице.

Узел (нода)	Событие	Примечание										
Публикация и подписка данных по протоколу MQTT (Mqtt)												
<div>MQTT publish topic</div>		<p><b>Вещественный вход.</b> Узел публикует на MQTT-брокере входное значение. <b>topic</b> — название темы. Его можно изменить, кликнув мышкой по названию темы (<b>topic</b>) на узле MQTT publish на схеме. Настройка подключения к MQTT-брокеру производится на вкладке <b>Networking</b>.</p> <table><tr><td>Вход</td><td>Публикуемое значение</td></tr><tr><td>Целое число N</td><td>Число N</td></tr><tr><td>Вещественное число X</td><td>Число X</td></tr><tr><td>Бесконечность (Inf)</td><td>Строка «Inf»</td></tr><tr><td>Не-число (NaN)</td><td>Строка «NaN»</td></tr></table>	Вход	Публикуемое значение	Целое число N	Число N	Вещественное число X	Число X	Бесконечность (Inf)	Строка «Inf»	Не-число (NaN)	Строка «NaN»
Вход	Публикуемое значение											
Целое число N	Число N											
Вещественное число X	Число X											
Бесконечность (Inf)	Строка «Inf»											
Не-число (NaN)	Строка «NaN»											
<div>MQTT describe topic</div>	ДА	<p><b>Вещественный выход.</b> Узел подписывается на заданную тему на MQTT-брокере и возвращает значение этой темы при новой публикации. <b>topic</b> — название темы. Его можно изменить, кликнув мышкой по названию темы (<b>topic</b>) на узле MQTT describe на схеме. Настройка подключения к MQTT-брокеру производится на вкладке <b>Networking</b>. При приеме с сервера нового значения указанной темы, генерируется событие для пересчета состояния узлов, связанных с выходом узла MQTT describe. Принимаются значения: число, бесконечность (Inf) и не-число (NaN).</p>										

Подобно рассказывать про MQTT-протокол я здесь не могу, но расскажу про настройку связи **ShIoTiny** с MQTT-сервером.

Чтобы публиковать свои данные и получать команды, контроллеру **ShIoTiny** необходимо установить соединение с MQTT-брокером. Для настройки соединения с MQTT-брокером, зайдите на вкладку «Сеть» («**Networking**») и сосредоточьтесь на разделе «Соединение с MQTT-сервером» («**MQTT connection to server**»). В нем мы можем настроить параметры доступа контроллера **ShIoTiny** к MQTT-серверу (брокеру).



The screenshot shows a web form titled "MQTT connection to server". It contains the following fields and labels:

- MQTT broker (server) url or IP**: A text input field containing "cloud.mqtt.ru".
- MQTT broker (server) port**: A text input field containing "1883".
- MQTT broker (server) login**: A text input field containing "login".
- MQTT broker (server) password**: A text input field containing a single dot ".".
- MQTT topic prefix**: A text input field containing a forward slash "/".
- Enable SSL (secure sockets layer)**: A checkbox that is currently unchecked.

Рис. 17: Параметры подключения к MQTT-серверу

Прежде всего, необходимо указать URL или IP-адрес MQTT-сервера и порт, к которым контроллер **ShIoTiny** будет присоединяться.

### 10.5.1 Сервер, порт, авторизация

Параметры подключения к MQTT-брокеру (серверу) вводятся соответственно в поля ввода «Адрес MQTT-брокера (сервера)» («**MQTT broker (server) url or IP**») и «Порт MQTT-брокера (сервера)» («**MQTT broker (server) port**»).

Для авторизации на MQTT-брокере вводится имя пользователя и пароль в поля «Имя пользователя MQTT-брокера» («**MQTT broker (server) login**») и «Пароль MQTT-брокера» («**MQTT broker (server) password**»).

Если авторизация на MQTT-брокере не требуется, то поля ввода имени пользователя и пароля для входа на MQTT-брокер не заполняются (сотрите все, что в них написано и оставьте эти поля пустыми).

### 10.5.2 Префикс

Префикс MQTT-параметров — это строка, добавляемая к названию темы при публикации и подписке на MQTT-брокере. Чтобы установить MQTT-префикс для вашего контроллера, его надо просто ввести в поле ввода **«Префикс темы MQTT»** (**«MQTT topic prefix»**). Префикс всегда начинается со слеша («/»)! Если вы не введете слэш в поле ввода — он добавится автоматически. В префиксе нельзя использовать символы «#» и «+». Других ограничений нет.

Если вы публикуете параметр **«status»** (или подписываетесь на него), а ваш префикс задан как **«/shiotiny/»**, то на брокере этот параметр будет опубликован под именем **«/shiotiny/status»**. Если у вас задан пустой префикс, то все параметры на брокере будут начинаться со слеша («/»): **«status»** будет публиковаться как **«/status»**.

### 10.5.3 Безопасность (SSL)

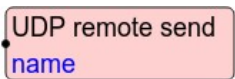
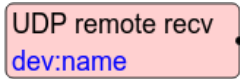
Для обеспечения безопасности, имеется возможность работать через «защищенные соединения» по технологии SSL (**Secure Sockets Layer** — уровень защищенных сокетов).

Если ваш MQTT-брокер использует SSL, то вы должны установить переключатель **«Разрешить SSL (защищенные соединения)»** (**«Enable SSL (secure sockets layer)»**) на вкладке **«Сеть»** (**«Networking»**).

## 10.6 Связь по протоколу Rlink UDP

Протокол Rlink UDP служит для обмена данными между устройствами в пределах локальной сети. Обмен данными ведется при помощи широковещательной рассылки (multicast). Разумеется, если вы создадите локальную сеть при помощи VPN, то можете обмениваться данными между устройствами, расположенными хоть на разных концах Земли.

Это значит, что все устройства «слышат» любой пакет, переданный одним из них.

Узел (нода)	Событие	Примечание
<b>Обмен данными протоколу Rlink UDP</b>		
		<p><b>Вещественный вход.</b> Передача значения по протоколу Rlink UDP с заданным именем.</p> <p>name — имя передаваемого значения. Его можно изменить, кликнув ему мышкой, когда узел <b>UDP remote send</b> находится в схеме. Имя передаваемого значения не должно содержать двоеточий!</p>
	ДА	<p><b>Вещественный выход.</b> Прием значения по протоколу Rlink UDP с заданным именем устройства и именем принимаемого значения.</p> <p>Имя устройства (dev) отделяется от имени параметра (name) двоеточием. Если имя устройства не задано (введено только имя принимаемого значения - <b>name</b>), то будут приниматься значения с указанным именем от любого устройства. Изменить имя устройства и значения можно кликнув по нему мышкой, когда узел <b>UDP remote recv</b> находится в схеме.</p> <p>Имя устройства и имя значения не должно содержать двоеточий! Двоеточия служат для отделения имени устройства от имени значения.</p>

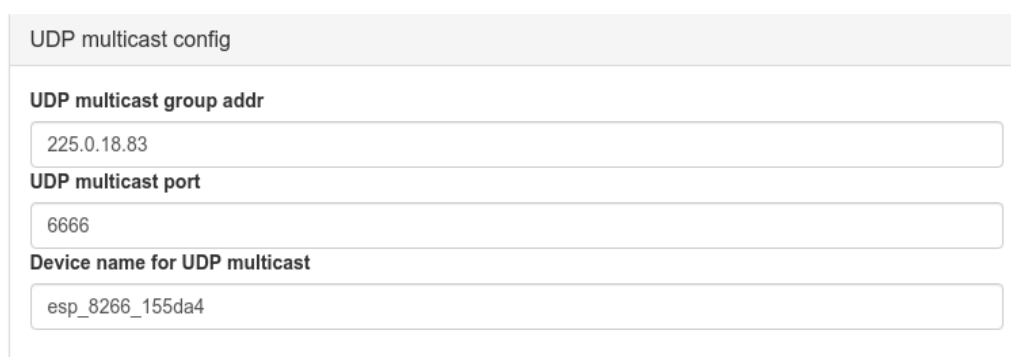
Каждое значение, которое отправляется по протоколу Rlink UDP помимо собственно значения, включает в себя имя устройства (**dev**) и имя значения (**name**).

Имя устройства задается одно для всех значений, а имя значения — непосредственно в узле.

Таким образом, разные устройства могут публиковать значения под одинаковыми именами и путаницы не произойдет.

Чтобы воспользоваться обменом данными по протоколу Rlink UDP, необходимо предварительно настроить параметры сети на вкладке **Networking**, раздел **UDP multicast config**.

Заметим, что обычно, параметры по умолчанию изменять не надо, за исключением имени устройства.



UDP multicast config

UDP multicast group addr

225.0.18.83

UDP multicast port

6666

Device name for UDP multicast

esp\_8266\_155da4

Рис. 18: Параметры передачи и приема данных по протоколу RLinkUDP

Здесь задаются три параметра:

- **адрес групповой рассылки** в пределах 224.0.0.1 — 239.255.255.254 (UDP multicast group addr);
- **порт групповой рассылки** в пределах 1000 — 65535 (UDP multicast port);
- **название (имя) устройства** (не более 16 символов) (Device name for UDP multicast). По умолчанию — имя устройства содержит серийный номер платы. Если вам захочется чего-то более красивого — введите свое имя. Имя устройства не должно содержать двоеточий!

Данные по **UDP multicast** передаются в формате JSON (в виде строки текста). Пример пакета **UDP multicast**, формируемого прошивкой **ShIoTiny** :

```
{"d":"device_name", "p":"parametr_name", "v":123.456}
```

В приведённом пакете данных передаётся число (значение) **123.456**.

Ключи JSON-объекта: «**d**» - имя устройства; «**p**» - название параметра; «**v**» - значение параметра.

**device\_name** — имя устройства (ключ «**d**»). Оно задаётся на странице настроек «**Networking/Device name for UDP multicast**» передающего устройства.

**parametr\_name** — название параметра (ключ «**p**»). Оно задаётся в поле ввода узлов «**UDP remote send**» и «**UDP remote recv**».

Для узла «**UDP remote recv**» название параметра может быть задано двумя способами:

- **parametr\_name** - в этом случае узел принимает параметр с именем **parametr\_name** от любого устройства.
- **device\_name:parametr\_name** - в этом случае узел принимает параметр с именем **parametr\_name** от только от устройства с именем **device\_name**.

Использовать двоеточие в именах устройства или названии параметра - нельзя!

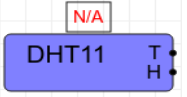
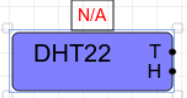
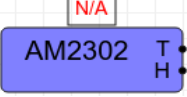
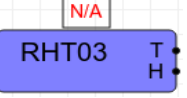
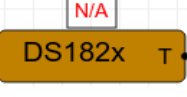
Если передаётся не-числовое значение или бесконечное значение, то вместо числа в поле значения (ключ "**v**") JSON-пакета передаётся строка "**nan**" или "**inf**". Например:

**{"d":"devname", "p":"paraname", "v":"inf"}** — передаётся **inf** (бесконечность) от устройства с именем "**devname**" и названием параметра "**paraname**".

**{"d":"mydevice", "p":"adc", "v":"nan"}** — передаётся **nan** (не-число) от устройства с именем "**mydevice**" и названием параметра "**adc**".

## 10.7 Датчики температуры и влажности

Датчики температуры и влажности воздуха, подключаются к специальному входу **DHT**. Если вы сами проектируете схему — ничего не мешает подключить к различным ножкам несколько датчиков. Но на контроллере ShIoTiny можно подключить только один датчик **DHTxx** или **DS182x**: вход **DHT** только один.

Узел (нода)	Событие	Примечание
<b>Датчики температуры и влажности</b>		
	ДА	<b>Вещественный выход.</b> Выход Т-температура воздуха Выход Н-влажность воздуха <i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i>
	ДА	<b>Вещественный выход.</b> Выход Т-температура воздуха Выход Н-влажность воздуха <i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i>
	ДА	<b>Вещественный выход.</b> Выход Т-температура воздуха Выход Н-влажность воздуха <i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i>
	ДА	<b>Вещественный выход.</b> Выход Т-температура воздуха Выход Н-влажность воздуха <i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i>
	ДА	<b>Вещественный выход.</b> Выход Т-температура воздуха Датчики <b>DS1820 / DS1822</b> <i>Номер физической ножки ESP8266, привязанной к данному узлу выбирается пользователем.</i>

Если датчик неисправен или физически отключен, то с выходов узлов датчиков Т и Н считывается значение NaN (не-число).



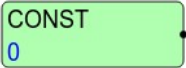

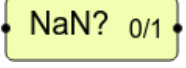
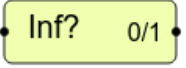
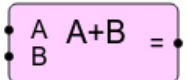
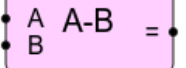
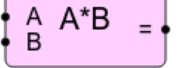
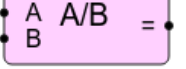
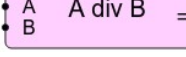
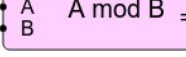


Заметьте, что хотя датчики семейства **DHTxx** и датчики семейства **DS182x** подключаются к одному и тому же входу, но работают они по абсолютно разным протоколам.

**Внимание!**

Если вы подключаете датчик семейства **DS182x**, то вы должны включить сопротивление **4,7кОм** между входом данных датчика и питанием +3.3В! Иначе датчик работать не будет!

## 10.8 Математические функции

Самые простые и понятные функции. Даже пояснять ничего почти не надо. Если кто-то их не понял — то вам нужен учебник арифметики для 2-го класса средней школы.

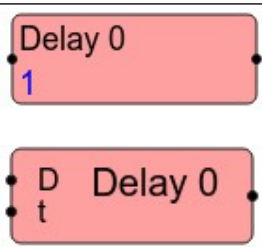
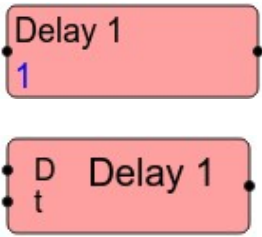
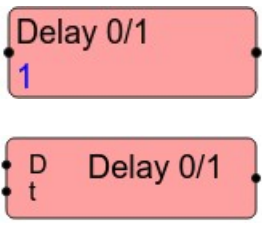
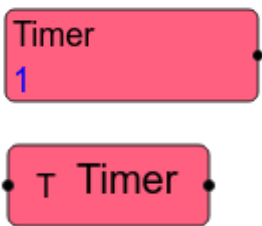
Математические функции (Math functions)		
		<b>Вещественный выход.</b> Константа. Любое число. Для изменения константы перетащите узел на схему и ткните на него мышкой.
		<b>Вещественный выход.</b> Константа «не-число». Это особая константа, которая не равна ни одному числу и даже самой себе.
		<b>Вещественный выход.</b> Бесконечность. Константа, которая больше любого числа.
		<b>Вещественные вход и выход.</b> Проверка, является ли значение на входе «не-числом». Если является — то на выходе 1. Иначе на выходе 0.
		<b>Вещественные вход и выход.</b> Проверка, является ли значение на входе бесконечностью. Если является — то на выходе 1. Иначе на выходе 0.
		<b>Вещественные входы и выход.</b> Выход — сумма чисел на входах $A + B$ .
		<b>Вещественные входы и выход.</b> Выход — разность чисел на входах $A - B$ .
		<b>Вещественные входы и выход.</b> Выход — произведение чисел на входах $A * B$ .
		<b>Вещественные входы и выход.</b> Выход — частное от деления чисел на входах $A/B$ .
		<b>Вещественные входы и выход.</b> Выход — целая часть от деления чисел на входах $A/B$ .
		<b>Вещественные входы и выход.</b> Выход — остаток от деления чисел на входах $A/B$ .
		<b>Вещественные вход и выход.</b> Выход — округление входного числа до ближайшего меньшего целого.
		<b>Вещественные вход и выход.</b> Выход — округление входного числа до ближайшего целого.

<div><div>A</div><div>B</div><div>A=B</div><div>=</div></div>		<b>Вещественные входы и бинарный выход.</b> Если A=B, то на выходе 1, иначе на выходе 0.
<div><div>A</div><div>B</div><div>A≠B</div><div>≠</div></div>		<b>Вещественные входы и бинарный выход.</b> Если A<>B, то на выходе 1, иначе на выходе 0.
<div><div>A</div><div>B</div><div>A&gt;B</div><div>&gt;</div></div>		<b>Вещественные входы и бинарный выход.</b> Если A>B, то на выходе 1, иначе на выходе 0.
<div><div>A</div><div>B</div><div>A&lt;B</div><div>&lt;</div></div>		<b>Вещественные входы и бинарный выход.</b> Если A<B, то на выходе 1, иначе на выходе 0.

## 10.9 Функции работы со временем

Нет, машину времени, несмотря на название этих функций, мы не построим.

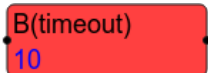
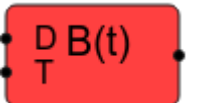

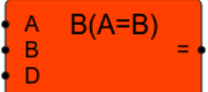
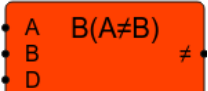
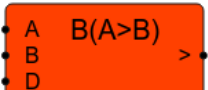
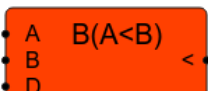
Но все системы управления работают с какими-то процессами. А процесс — это функция, аргументом которой является время. Так что фактически — это функции для измерения времени, счета, суммирования и задержки.

Узел (нода)	Событие	Примечание
<b>Функции работы со временем</b>		
	ДА	<p><b>Бинарные вход выход.</b>          Фильтр «задержка нуля». При изменении входа с 0 на 1 — немедленно выдает на выход 1. При изменении входа с 1 на 0 — выдает на выход 0 только при прошествии заданного времени. Время задержки задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом время задержки задается константой, во втором — принимается с выхода другого узла на вход <b>t</b>.</p>
	ДА	<p><b>Бинарные вход выход.</b>          Фильтр «задержка единицы». При изменении входа с 1 на 0 — немедленно выдает на выход 0. При изменении входа с 0 на 1 — выдает на выход 1 только при прошествии заданного времени. Время задержки задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом время задержки задается константой, во втором — принимается с выхода другого узла на вход <b>t</b>.</p>
	ДА	<p><b>Бинарные вход выход.</b>          Фильтр «задержка изменения состояния». При изменении входа с 0(1) на 1(0) — выдает на выход 1(0 только при прошествии заданного времени. Время задержки задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом время задержки задается константой, во втором — принимается с выхода другого узла на вход <b>t</b>.</p>
	ДА	<p><b>Бинарный выход.</b>          Таймер. Выдаст на выход попеременно нули и единицы с заданным периодом. Период задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом период таймера задается константой, во втором — принимается с выхода</p>

		другого узла на вход <b>T</b> .
 	ДА	<p><b>Бинарные вход и выход.</b>  Управляемый таймер. Выдает на выход попеременно нули и единицы с заданным периодом, если на входе 1. Если на вход подать 0, то перестает считать.  Период задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом период таймера задается константой, во втором — принимается с выхода другого узла на вход <b>T</b>.</p>
		<p><b>Бинарные входы и целый выход.</b>  Счетчик. Считает количество переходов с 0 на 1 на входе <b>C</b>. Подача 1 на вход <b>R</b> — обнулит счетчик.  На выходе <b>Q</b> — количество посчитанных импульсов.</p>
		<p><b>Бинарные входы C и R, вещественные вход D и выход Q.</b>  Сумматор. Суммирует все числа, пришедшие на вход <b>D</b>. Суммирование происходит при изменении с 0 на 1 на входе <b>C</b>.  Подача 1 на вход <b>R</b> — обнуляет счетчик.  На выходе <b>Q</b> — сумма всех чисел, пришедших на вход <b>D</b>, пока <b>R</b>=0.</p>
 		<p><b>Бинарные вход и выход.</b>  Формирователь импульса. При изменении входного значения с 0 на 1, на выходе формируется импульс единичный заданной длительностью.  Длительность импульса задается в десятых долях секунды (0.1сек) или децисекундах.</p> <p>Существует два варианта узла: в первом длительность импульса задается константой, во втором — принимается с выхода другого узла на вход <b>T</b>.</p>
 		<p><b>Бинарные вход и выход.</b>  Таймаут актуальности значения. При поступлении на вход <b>любого</b> события, на выходе появляется значение 1.  Если в течении заданного времени не приходит нового события, то выходное значение сбрасывается в 0.  Узел применяется в основном при обработке данных, принятых от других устройств (совместно с узлами UDP и MQTT).</p> <p>Существует два варианта узла: в первом время актуальности события задается константой, во втором — принимается с выхода другого узла на вход <b>T</b>.</p>

## 10.10 Функции управления прохождением событий

Узлов работы с данными у нас множество. А как же события? Разумеется, прохождением событий от узла к узлу тоже можно управлять.

Узел (нода)	Событие	Примечание
<b>Функции управления прохождением событий (Barriers and filters)</b>		
 	ДА	<p><b>Вход и выход — вещественные.</b>  Позволяет прохождение событий не чаще, чем раз в заданный интервал времени. (Фильтр частоты событий).  При поступлении события на вход, данные со входа копируются на выход. Затем прохождение событий блокируется на заданное время.</p> <p>Существует два варианта узла: в первом интервал времени задается константой, во втором — принимается с выхода другого узла на вход <b>T</b>.</p>
		<p><b>Вход и выход — вещественные.</b>  Генерирует событие на выходе при переходе входа <b>C</b> из 0 в 1.  Выход всегда равен входу <b>D</b>.</p>
		<p><b>Входы и выход — вещественные.</b>  Позволяет прохождение событий, если <math>A=B</math>.  Выход всегда равен входу <b>D</b>.</p>
		<p><b>Входы и выход — вещественные.</b>  Позволяет прохождение событий, если <math>A \neq B</math>.  Выход всегда равен входу <b>D</b>.</p>
		<p><b>Входы и выход — вещественные.</b>  Позволяет прохождение событий, если <math>A &gt; B</math>.  Выход всегда равен входу <b>D</b>.</p>
		<p><b>Входы и выход — вещественные.</b>  Позволяет прохождение событий, если <math>A &lt; B</math>.  Выход всегда равен входу <b>D</b>.</p>

## 10.11 Функции сетевого сервиса

Узел (нода)	Событие	Примечание
<b>Функции сетевого сервиса</b>		
ICMP Ping 192.168.1.1	ДА	<b>Бинарный входы и бинарный выход.</b> Пингует заданный адрес. URL задавать нельзя! Только IP!  По переходу входа из 0 в 1 — посылает пинг-пакет. По переходу входа из 1 в 0 — считает, что тайм-аут истёк.  Выход: 0 — ответа на пинг нет, 1 — ответ на пинг есть.
AP connect?	ДА	<b>Бинарный выход.</b> Проверяет, есть ли связь с точкой доступа (AP) по WiFi. Если связь есть — на выходе 1. Если связи нет — на выходе 0.

## 10.12 Функции сохранения параметров в энергонезависимую память

Узел (нода)	Событие	Примечание
<b>Функции сохранения параметров в энергонезависимую память</b>		
FLASH store name		<b>Вход вещественный</b> Узел сохраняет принятое значение в энергонезависимой памяти под заданным именем. Размер имени — не более 20 символов. При приеме значения, отличного от существующего, обновляются все значения узлов « <b>FLASH restore</b> » с заданным именем и на них генерируются события. <b>Размер имени — не более 20 символов.</b>
FLASH restore name	ДА	<b>Выход вещественный</b> Узел восстанавливает ранее сохраненное значение под заданным именем из энергонезависимой памяти.  Возможно задать значение по умолчанию в формате <b>name:1234.56</b> (имя, двоеточие, значение по умолчанию).  <b>Размер имени — не более 20 символов.</b>

Имеются некоторые особенности работы с узлами сохранения параметров в энергонезависимую память.




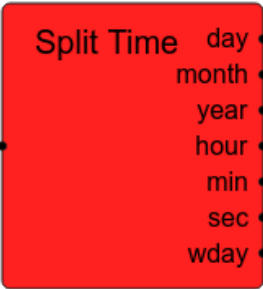
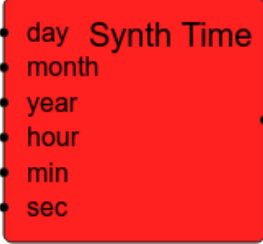
Если на вход узла FLASH store поступит значение, равное ранее сохраненному, то повторного сохранения в энергонезависимую память не произойдет и события на узлах FLASH restore с тем же именем возникать не будут. Это сделано для того, чтобы не «убить» FLASH-память при сохранении одного и того же значения много раз подряд.



## 10.13 Функции работы с календарем и синхронизации времени


Этот набор узлов предназначен для работы с календарной датой и временем.

Системное время хранится в формате UNIX time, то есть в виде количества секунд, прошедших с 01.01.1970.

Узел (нода)	Событие	Примечание
<b>Функции работы с календарем и синхронизации времени</b>		
	ДА	<b>Выход целый (количество секунд с 01.01.1970)</b> Получить системное время с внутренних часов устройства. В узле задается один параметр — часовой пояс. Может принимать значения от -12 до +12. Так как NTP-сервера отдают обычно время в GMT, то системное время также хранится в GMT. Для пересчета его в местное достаточно задать часовой пояс.
		<b>Вход целый (количество секунд с 01.01.1970)</b> Установить системное время на устройстве. Просто устанавливает внутренние часы устройства.
	ДА	<b>Выход целый (количество секунд с 01.01.1970)</b> Получение времени с заданного NTP-сервера через заданный промежуток времени. Через запятую задается url сервера и период получения времени <b>в минутах</b> . По умолчанию время с сервера запрашивается каждые <b>60 минут (1 час)</b> . Минимальный период запроса 1 минута.
		<b>Вход целый (количество секунд с 01.01.1970)</b> <b>Выходы целые</b> Разбивает входное время в формате UNIX time на компоненты: day - день месяца (1-31); mon — месяц (0-11); year — год; hour — час (0-23); min — минута (0-59); sec — секунда (0-59); wday — день неделт (0-6, 0-воскресенье).
		<b>Входы целые</b> <b>Выход целый (количество секунд с 01.01.1970)</b> Синтезирует UNIX time на основе входных данных (они аналогичны предыдущему узлу): day - день месяца (1-31); mon — месяц (0-11); year — год; hour — час (0-23); min — минута (0-59); sec — секунда (0-59); wday — день неделт (0-6, 0-воскресенье).

## 10.14 Системные функции

Узлы управления системой в целом

Узел (нода)	Событие	Примечание
<b>Системные функции</b>		
		<b>Вход бинарный</b> Любое ненулевое значение на входе вызывает немедленный рестарт контроллера

## 10.15 Шина I2C

Шина **I2C** или **TWI** является одной из популярных шин для работы с внешними устройствами, к которой можно присоединить несколько устройств одновременно. Шина **I2C** имеет два сигнальных проводника: **SDA** (двухнаправленный обмен данными) и **SCL** (линия синхронизации, управляется ведущим).

Шина **I2C** предполагает, что имеется одно управляющее устройство (ведущий, master, хозяин), которое управляет обменом по этой шине: указывает, что обмен по шине начался или закончился; посылает запросы.

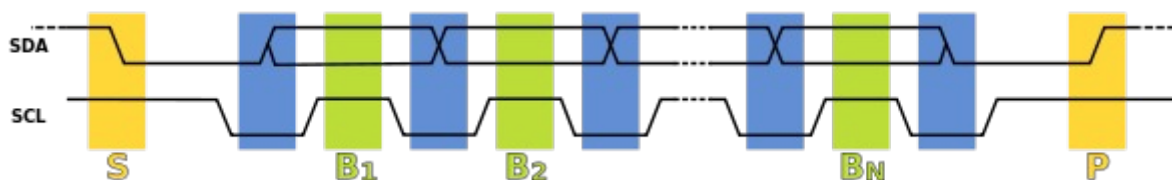
Остальные устройства на шине — управляемые (ведомые, slave, рабы). Эти устройства постоянно ждут запроса от ведущего устройства.

Шина является адресной: у каждого ведомого устройства имеется адрес, размером 7 бит. Это позволяет адресовать до 128 устройств.

Некоторые устройства позволяют выбрать их адрес из нескольких возможных; некоторые устройства имеют жёстко заданный неизменяемый адрес.

Каждый цикл обмен по шине **I2C** начинается стартовым событием **S** и завершается стоповым событием **P**. Цикл обмена показан на рисунке ниже.

**B<sub>1</sub>-B<sub>N</sub>** — биты данных.



Первый байт, который посылает на шину ведущий после того, как выставлено старт-событие **S** — включает в себя 7-битный адрес + бит чтения-записи **RW**. Адрес определяет к какому именно устройству на шине идёт обращение, а бит **RW** — будет ли информация записана в ведомое устройство или считана из ведомого устройства.

Таких циклов чтения или записи с одним и тем же устройством может быть несколько между стартовым **S** и стоповым **P** событиями на шине.

Это очень краткий экскурс в принципы работы шины **I2C**. Более подробно можете почитать соответствующую статью в **Wikipedia** или воспользоваться

поисковым запросом. Скажу лишь, что в настоящее время имеются спецификации на шину **I2C** с расширенным функционалом.

С точки зрения **ShIoTiny** для работы с шиной **I2C** необходимо сконфигурировать две линии ввода-вывода: **SDA** и **SCL**. Это делается с помощью узла **I2C bus**. Он выглядит на схеме, как показано на рисунке ниже:

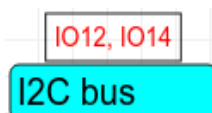
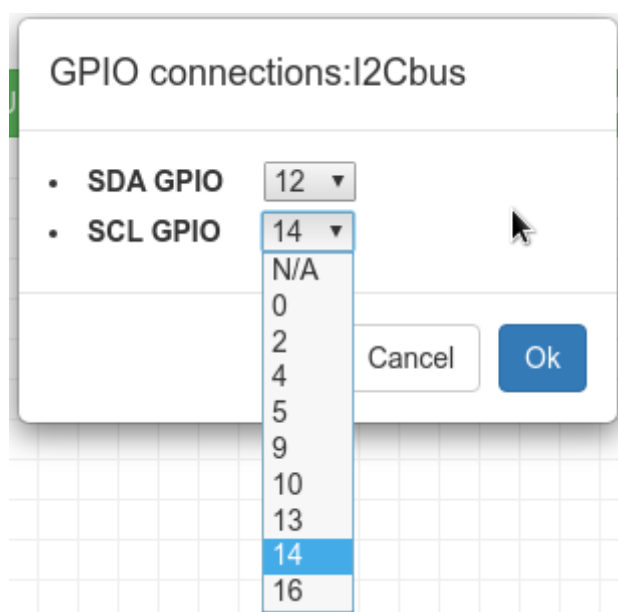


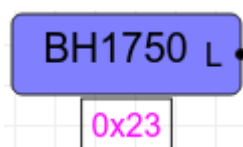
Рис. 19: Узел конфигурации сигналов шины I2C

Над узлом **I2C bus** указаны используемые им линии **GPIO**. Если ткнуть в узел мышкой, то откроется диалог выбора линий, такой как показан на рисунке ниже (и такой же, как при выборе линий **GPIO** в других узлах).

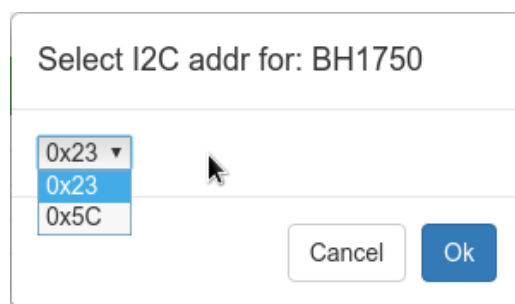


**Узел *I2C bus* обязательно должен быть на схеме, если предполагается работа с I2C-устройствами. Если данного узла нет — устройства I2C не будут работать!**

Если **I2C**-устройство имеет изменяемый адрес, то он указан снизу узла, как показано на рисунке ниже. В примере **0x23** — адрес устройства-датчика освещённости.


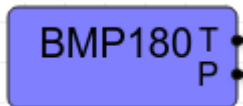
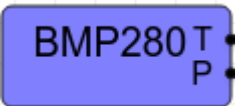


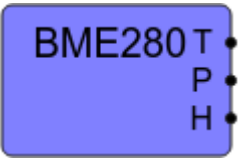

Для изменения адреса устройства необходимо ткнуть мышкой на узел этого устройства, после чего откроется диалог выбора адреса устройства (см. рис. ниже).



Выберите адрес, соответствующий реальному адресу подключённого устройства и нажмите кнопку «**Ок**».

Узлы, работающие с шиной **I2C** и устройствами, подключёнными к ней описаны в таблице ниже.

Узел (нода)	Событие	Примечание
<b>Управление шиной I2C</b>		
		Управление шиной. Позволяет выбрать линии <b>GPIO</b> для сигналов <b>SDA</b> и <b>SCL</b> . <i>Данный узел обязательно должен быть на схеме, если предполагается работа с I2C-устройствами. Если данного узла нет — устройства I2C не будут работать!</i>
<b>Устройства, подключаемые к шине I2C</b>		
	ДА	<b>Выходы вещественные</b> Датчик температуры и давления. Температура — в градусах цельсия. Давление — в Паскалях. Если не связи с датчиком, то на выходах <b>T</b> и <b>P</b> выставляется значение <b>NAN</b> (не-число). Адрес всегда равен <b>0x77</b> , не изменяется. Период опроса датчика <b>10сек.</b>
	ДА	<b>Выходы вещественные</b> Датчик температуры и давления. Температура — в градусах цельсия. Давление — в Паскалях. Если не связи с датчиком, то на выходах <b>T</b> и <b>P</b> выставляется значение <b>NAN</b> (не-число). Возможные адреса <b>0x76</b> и <b>0x77</b> . Период опроса датчика <b>10сек.</b>

	ДА	<b>Выходы вещественные</b> Датчик температуры, давления и влажности. Температура — в градусах цельсия. Давление — в Паскалях. Влажность — относительная, в процентах. Если не связи с датчиком, то на выходах <b>Т</b> , <b>Р</b> и <b>Н</b> выставляется значение <b>NAN</b> (не-число). Возможные адреса <b>0x76</b> и <b>0x77</b> . Период опроса датчика <b>10сек</b> .
	ДА	<b>Выходы вещественные</b> Датчик освещённости. Выставляет на выходе <b>L</b> освещённость в Люксах. Если не связи с датчиком, то на выходе <b>L</b> выставляется значение <b>NAN</b> (не-число). Возможные адреса <b>0x23</b> и <b>0x5C</b> . Период опроса датчика <b>5сек</b> .

## 10.16 Модуль индикации на базе микросхемы TM1637

Микросхема **TM1637** — популярная микросхема для вывода информации на 7-сегментные индикаторы (те самые, что в виде восьмёрок).

В программное обеспечение **ShIoTiny** введён узел, который позволяет управлять микросхемой **TM1637**.

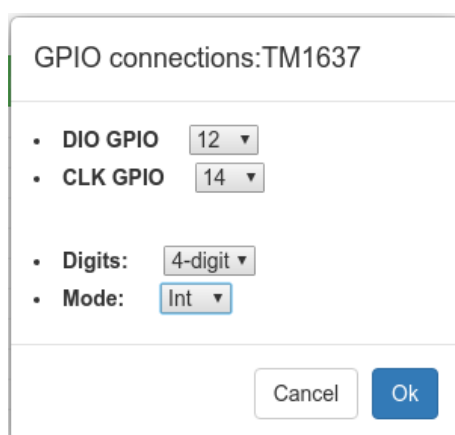
Данный узел имеет следующие параметры конфигурации (см. рис. ниже):

**DIO GPIO** — номер пина **ESP8266** (линия обмена данными с микросхемой **TM1637**);

**CLK GPIO** — номер пина **ESP8266** (линия синхронизации микросхемы **TM1637**);

**Digits** — количество 7-сегментных индикаторов, подключённых к микросхеме **TM1637** (от 1 до 6);

**Mode** — режим вывода чисел (**Int** — целые, **Hex** — шестнадцатеричные).



Диалог ввода параметров микросхемы **TM1637** доступен при клике мышкой на узел **TM1637**, установленный на схеме.

### Узлы управления микросхемой TM1637

Узел (нода)	Событие	Примечание
<b>Системные функции</b>		
• D TM1637		<b>Вход целый</b> В зависимости от режима — вывод на индикатор числа в 10м или 16м виде. Выводится от 1 до 6 цифр (определяется настройками)

## Приложение А Вкладка состояния ShIoTiny (ShIoTiny info page)

Это основная вкладка, на которую вы попадаете при входе на страницу устройства.

Здесь отображается общее состояние устройства.

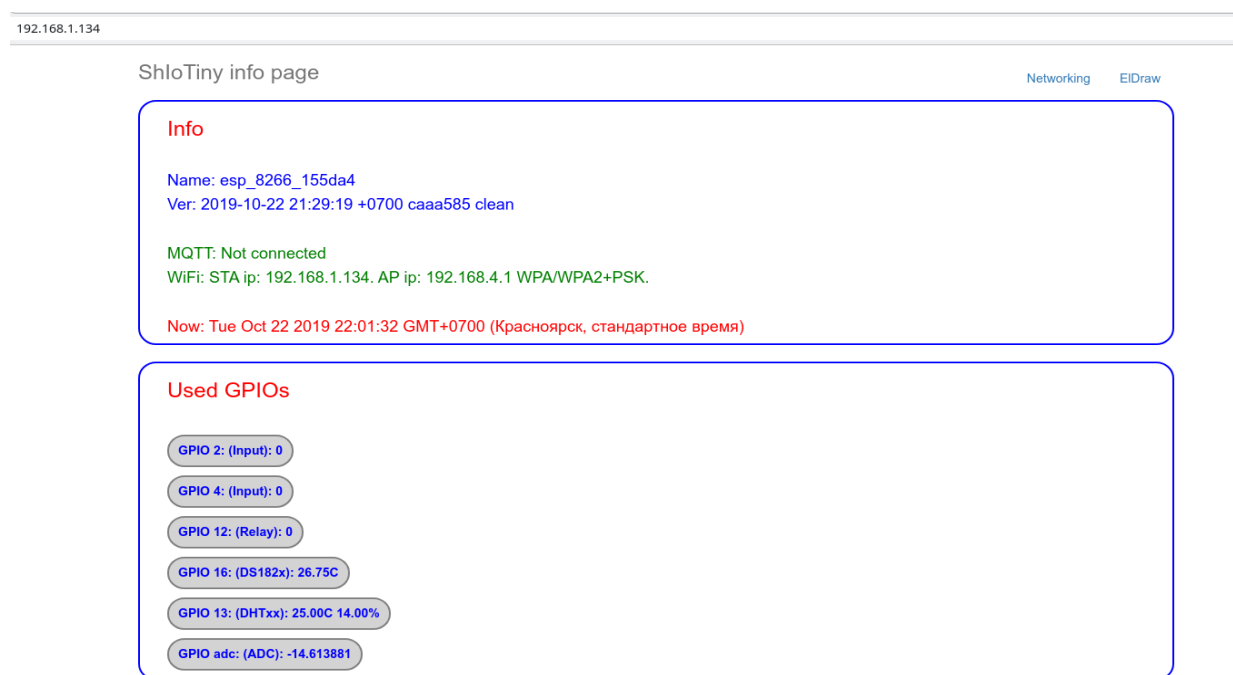


Рис. 20: Вкладка текущего состояния *ShIoTiny* (Control and status)

### Поле Info

Отображает текущие параметры сети, времени и другие.

**Name:** уникальное имя устройства, используется как имя клиента для входа на **MQTT** сервер. Изменить это имя нельзя. Кроме того, это имя устройства является именем по умолчанию для точки доступа (в режиме точки доступа) и для обмена по **UDP multicast**. Имена точки доступа и имя устройства для **UDP multicast** можно изменить на другие (вкладка Networking).

**Ver:** дата создания ПО и его версия (в нашем примере версия **caaa585**). По дате можно судить — новая версия или старая.



**MQTT:** показывает есть ли соединение с сервером **mqtt** или его нет. Если соединение установлено, то показываются его параметры: URL или IP, номер порта, используется SSL или нет.

**WiFi:** показывает параметры соединения по WiFi: IP-адреса станции и/или точки доступа, в зависимости от режима в котором находится ShIoTiny. В нашем примере включен режим **AP+Station Mode**, то есть одновременно и станция и точка доступа.

**Now:** отображает текущую системную дату и время ShIoTiny. Чтобы это работало, надо предварительно установить дату-время с помощью узла **SetTime**, например, получив дату-время по NTP. Если системные дата и время не установлены — то будет показана надпись «time is not set» («время не установлено»).

## Поле Used GPIOs

Показывает все используемые «ножки» ввода-вывода GPIO. Показывается номер или имя «ножки» GPIO и текущее значение датчика или реле (выхода), который к этой ножке подключен.

Помните, что бинарные входы и выходы работают в инверсной логике! То есть, если например на вход GPIO2 подается 0В, то на странице будет отображаться **GPIO2: (Input): 1**. Так же и для выходов (**Relay**): если на странице отображается **GPIO12: (Relay): 1**, то на выводе GPIO12 будет 0В!

## Приложение Б Вкладка настройки сети и режимов работы ShIoTiny (Networking)

Вкладка настройки сети и режимов работы **ShIoTiny** позволяет настроить: Основной режим работы; работу в режиме точки доступа WiFi; работу в режиме станции WiFi; соединение с MQTT-брокером; обмен широковещательным пакетами по локальной сети.

ShIoTiny networking and MQTT config Control and status EIDraw

Network mode and access

Name (SSID)

ssid

Password

\*\*\*\*\*

AP Name (SSID)

apssid

AP Password

\*\*\*\*\*

AP IP address

192.168.4.1

ShIoTiny mode: Station mode

Scan WiFi

Lock Web in Station mode

MQTT connection to server

MQTT broker (server) url or IP

cloud.mqtt.ru

MQTT broker (server) port

1883

MQTT broker (server) login

login

MQTT broker (server) password

\*

MQTT topic prefix

/

Enable SSL (secure sockets layer)

UDP multicast config

UDP multicast group addr

225.0.18.83

UDP multicast port

6666

Device name for UDP multicast

esp\_8266\_155da4

Save

Cancel

Reset to default

Рис. 21: Вкладка настройки сети и режимов работы **ShIoTiny** (Networking)

Подробнее о настройках смотрите в соответствующих разделах (4, 10.5, 10.6).

## Приложение В Работа с редактором ElDraw

Главная рабочая вкладка — редактор схем-программ.

*Заметьте, что переход на другую вкладку из редактора, уничтожает все ваши не сохраненные изменения! Перед тем, как закрыть редактор или перейти на другую вкладку — сохраните изменения на диск или в устройство!*

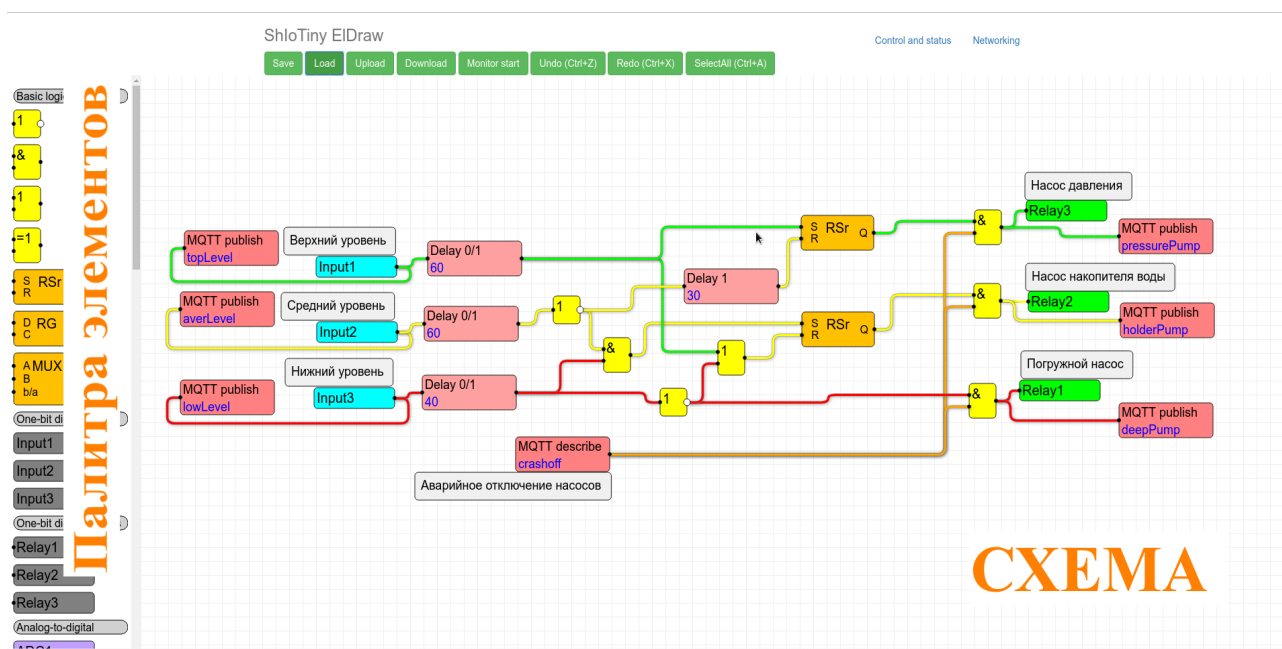


Рис. 22: Редактор схем-программ ElDraw.

Слева — палитра элементов-узлов.

Справа — схема-программа (схема).

Сверху — кнопки управления загрузкой-выгрузкой и редактированием.

### Кнопки работы с диском.

**Save** — сохранить схему на ваш компьютер.

**Load** — загрузить схему с вашего компьютера в редактор (не на устройство!).

### **Кнопки работы с устройством ShIoTiny.**

**Upload** — загрузить схему из редактора на устройство.

*Чтобы программа была загружена на устройство необходимо, чтобы все входы всех элементов были соединены!*  
*Если какой-либо вход не присоединен, будет выдано предупреждение и схема не загрузится на устройство!*

**Для удаления схемы из устройства** загрузите в устройство пустую схему (не содержащую ни одного элемента-узла).

**Download** - загрузить схему из устройства в редактор. Вообще схема, если она имеется в устройстве, загружается автоматически. Но если вы что-то наисправляли-поменяли, то загрузка схемы из устройства позволяет все ваши исправления отменить.

**Monitor Start/Monitor Stop** — включение-выключение режима монитора. Он очень полезен для отладки схем. При включенном мониторе возле каждого входа и выхода показывается его реальное состояние.

*Советуем перед каждым запуском монитора нажимать кнопку Download для синхронизации схемы и программы.*

### **Кнопки редактирования.**

**Undo (CTRL+Z)** — отменить действие редактирования.

**Redo (CTRL+X)** — вернуть действие редактирования.

**Select All (CTRL+A)** — выделить все, что есть на схеме.

### **Работа с редактором.**

**Для того, чтобы вставить узел** в схему — наведите курсор мыши на нужный узел в палитре элементов; затем нажмите левую клавишу мыши. После этого, удерживая нажатой левую клавишу мыши перетащите этот узел из палитры элементов на схему.

**Для того, чтобы выделить узел или связь** наведите курсор мыши на нужный узел или связь в схеме и нажмите левую клавишу мыши.

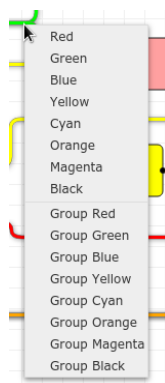
Для того, чтобы выделить группу узлов и/или связей, наведите курсор мыши на произвольную точку схемы, свободную от узлов и связей и нажмите левую клавишу мыши. Затем, удерживая нажатой левую клавишу мыши, перемещайте курсор мыши, пока не выделите прямоугольную область, включающую нужные вам узлы и связи.

Для того, чтобы выделить несколько произвольных узлов и связей, нажмите клавишу SHIFT. Затем, удерживая клавишу SHIFT, поочередно наведите курсор мыши на каждый из выделяемых элементов и связей и при наведении на каждый из них нажмите однократно левую клавишу мыши.

Для удаления узлов и связей выделите их любым способом и нажмите клавишу DEL.

Для перемещения узлов и связей выделите их любым способом. Затем наведите курсор мыши на любой выделенный узел или связь и нажмите левую клавишу мыши. После этого, удерживая нажатой левую клавишу мыши перемещайте всю выделенную группу узлов и связей, одиночный элемент или связь.

Для того, чтобы раскрасить связи между узлами наведите курсор мыши на нужную связь и нажмите правую клавишу мыши. Появится выпадающее меню, как на рисунке ниже.



Выберете курсором нужный цвет связи или группы связей и нажмите левую клавишу мыши.

Если выбрана одиночная раскраска — изменит цвет только выбранная связь.

Если выбрана раскраска группы связей — изменят цвет все связи, подключенные к тому же выходу, что и выбранная связь.

Таблица цветов приведена ниже.

№	Пункт меню	Цвет	Примечание
1	Red	Красный	Одиночная связь
2	Green	Зеленый	Одиночная связь
3	Blue	Синий	Одиночная связь
4	Yellow	Желтый	Одиночная связь
5	Cyan	Голубой	Одиночная связь
6	Orange	Оранжевый	Одиночная связь
7	Magenta	Сиреневый	Одиночная связь
8	Black	Черный	Одиночная связь
9	Group Red	Красный	Группа связей
10	Group Green	Зеленый	Группа связей
11	Group Blue	Синий	Группа связей
12	Group Yellow	Желтый	Группа связей
13	Group Cyan	Голубой	Группа связей
14	Group Orange	Оранжевый	Группа связей
15	Group Magenta	Сиреневый	Группа связей
16	Group Black	Черный	Группа связей

Раскраска связей служит для удобства пользователей и никак не влияет на работу схемы-программы.

Для того, чтобы переместить сегмент связи, выделите нужную связь. Затем подведите курсор мыши к углу нужной сегмента связи. На углах сегментов линии связи при ее выделении появляются метки, как показано на рисунке ниже.

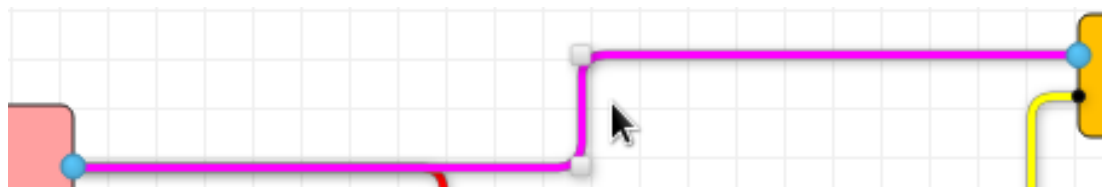


Рис. 23: При выделении линии связи, на углах ее сегментов появляются метки

При наведении на эти метки курсор мыши изменяет форму, что и показано на рисунке ниже.

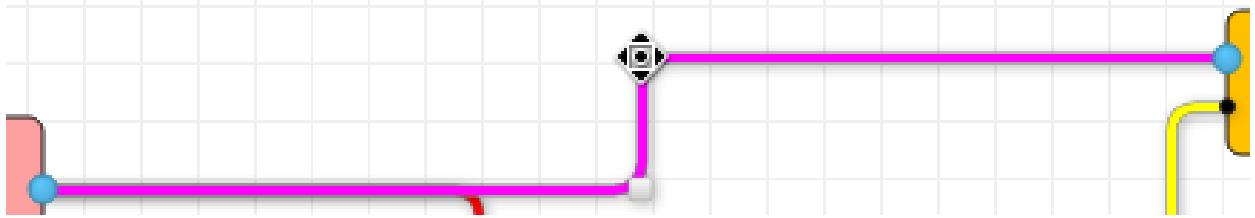


Рис. 24: При наведении на метки выделенной линии связи, курсор мыши изменяет форму

Наведя курсор, на нужный угол сегмента линии связи, нажмите левую клавишу мыши.

После этого, удерживая нажатой левую клавишу мыши перемещайте сегмент линии связи в нужное место.

## Приложение Г Соответствие сигналов на плате ShIoTiny-07 и выводов ESP8266

Дискретные входы (InputX)			
	Сигнал	ESP-07	Описание
1	Input 1	GPIO2	Сухой контакт (только замыкание) (акт. 0)
2	Input 2	GPIO5	Сухой контакт (только замыкание) (акт. 0)
3	Input 3	GPIO4	Сухой контакт (только замыкание) (акт. 0)
Дискретные выходы (OutputX)			
	Сигнал	ESP-07	Описание
1	Output 1	GPIO12	Переключающий контакт реле 220В 1А (акт. 0)
2	Output 2	GPIO14	Переключающий контакт реле 220В 1А (акт. 0)
3	Output 3	GPIO16	Переключающий контакт реле 220В 1А (акт. 0)
Аналоговый вход АЦП, 10бит, 1Вольт (ADCX)			
	Сигнал	ESP-07	Описание
1	ADC1	ADC	Полярность положительная, макс. 1В
Подключение однопроводных датчиков (DHTxx / DS1820/22 и аналогов)			
1	OneWire	GPIO13	
Специальные сигналы			
1	Reset	RST	Кнопка сброса устройства, подтяжка +3В
2	Prog	GPIO0	Кнопка перевода в режим программирования, подтяжка +3В (акт. 0), светодиод «Состояние»
3	AP	GPIO15	Кнопка перевода в режим точки доступа, подтяжка +0В (акт. 1, долгое удержание)
Интерфейс UART (для перепрограммирования устройства)			
1	RxD	RxD	Только для перепрограммирования
2	TxD	TxD	Только для перепрограммирования