Embedded Systems SIA, VAT No LV40003411103
47. Katolu str., Riga, LV 1003, LATVIA
Phone: +371 67648888, fax: +371 67205036, e-mail: sales@openrb.com

Embedded Systems Rus
1-я ул.Энтузиастов д.3, Embedded Systems Rus, 4 этаж, ком.46
Тел.8-800-775-06-34, e-mail: support@lm.net.ru

LogicMachine5 Power KNX CANx DALI(LM5p2-KCD)

Документация продукта

Версия документа 1.0
Апрель, 2021

Техническая поддержка:
support@lm.net.ru

Авторские права

Примечание

Embedded Systems SIA. оставляет за собой права на изменение информации в этом документе в случае необходимости. Embedded Systems SIA не несет ответственности за любые ошибки, которые могут появиться в этом документе. Информация в документе предоставляется исключительно для разработчиков с целью подключения системы и использования продукции KNX/EIB LogicMachine.

Торговые марки

LogicMachine является торговой маркой Embedded Systems SIA. Все остальные наименования и торговые марки являются собственностью их соответствующих владельцев и признаются в настоящем документе.

Вступление

LogicMachine (LM) – это самый простой способ запрограммировать сложную логику в KNX/EIB, Modbus, BACnet, DALI, сетях 1-wire. LM позволяет эффективно настроить процессы автоматизации зданий, легко обеспечивая неограниченную гибкость для конечных пользователей экономически эффективным способом.

LM5 Power KCD является встраиваемой платформой с интегрированной поддержкой Ethernet, USB, KNX/EIB, DALI, 1-wire, Serial интерфейсов. LM может быть использован как межстандартный шлюз, контроллер логики, платформа визуализации, KNX/IP-роутер. Шаблоны скриптов предоставляют удобную и гибкую конфигурацию интерфейсов и интеграцию с облачными/сетевыми сервисами, устройствами сторонних производителей. Используя пользовательские скрипты, LM может одновременно действовать как термостат, панель безопасности, контроллер света и т. п. Магазин приложений LogicMachine и возможность разработки внешних приложений позволяют расширить функциональность устройства и адаптировать его к различным сегментам рынка.

LM5 Power KCD поддерживает Power-over-Ethernet. Кроме того, LM5 Power в 3 раза более производителен, чем все предыдущие версии LogicMachine, благодаря использованию более мощных CPU и RAM.

Техническая поддержка

Любое неисправное устройство должно быть возвращено Embedded Systems.

При появлении дополнительных технических вопросов по поводу продукта, пожалуйста, свяжитесь с нами с понедельника по пятницу в период времени 9:00 – 17:00 (GMT +02:00), написав письмо по адресу support@lm.net.ru.

Обновления прошивки доступны на www.openrb.com.

Внимание
Советы по безопасности

Установка и монтаж электрооборудования должны выполняться только квалифицированным специалистом. Устройства не следует использовать для работы с оборудованием, которое прямо или косвенно служит для поддержания здоровья или жизни человека, либо может подвергнуть опасности людей и животных, а также нанести ущерб имуществу.

Советы по монтажу

Устройства поставляются в рабочем состоянии. Включенные в поставку соединители для кабелей могут быть установлены на корпусе, если это необходимо.

Электрическое подключение

Устройство сконструировано для работы с защитным низким напряжением (SELV). Заземление устройства не требуется. При включении и выключении устройства следует избегать скачков напряжения.

Содержание

Спецификация устройства

Вид продукта

LogicMachine5 Power LM5Cp-KCD

Стандарты и нормы соответствия

EMC: EN61000-6-1,EN61000-6-3
CE: EMBS-CE-200312/01

Технические данные:

Источник питания: 12В-30В пост. тока на клеммных колодках или 12В-30В пост. тока Passive Power-over-Ethernet

Потребляемая мощность: 1.3Вт

Интерфейсы:

KNX/EIB TP1	1
CAN FT	1
10BaseT/100BaseTX	1
RS-485:	1
RS-485/RS-232	1 (программно переключается full-duplex=RS232, half-duplex=RS485)
DALI master	1 (до 64 балластов)
USB 2.0	1

Клеммы:

CAN FT Connection Terminal	Клеммная колодка 0.8мм2
Шина KNX	Клеммная колодка 0.8мм2
Питание	Винтовые, 1.5 мм2
Serial	Винтовые, 1 мм2

Элементы управления:

LED	1 Нагрузка CPU
	1 Работа
	Кнопка программирования 1

Корпус:

Материал	Полиамид
Цвет:	Серый
Габаритные размеры	71(Ш)x90(В)x61(Д) мм
Рабочая температура:	0С +45С

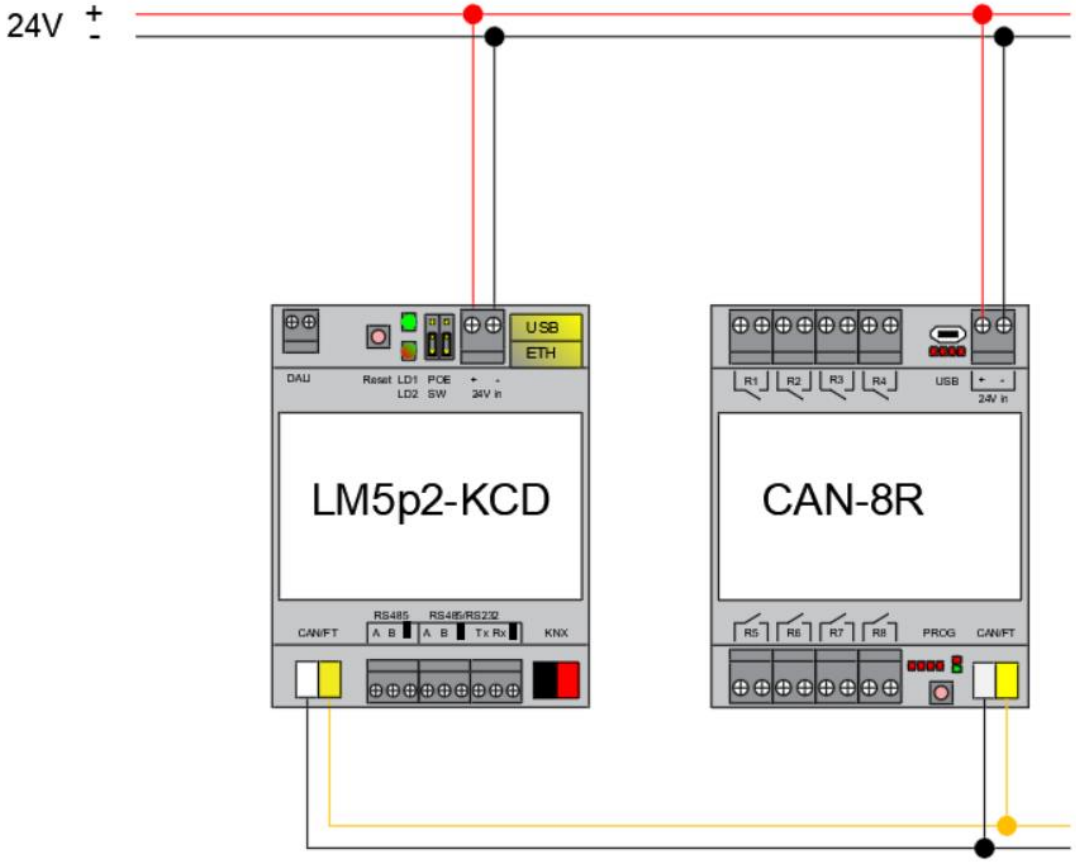
Температура хранения:	-15C ... +55C
Масса нетто:	119г
Вес брутто:	137г
Гарантия:	2 года
Относительная влажность:	10-95% без образования конденсата

LogicMachine5 Power содержит:

- Встроенная плата с предустановленным программным обеспечением;
- Пластиковый корпус на DIN-рейку.

1. Схемы подключения

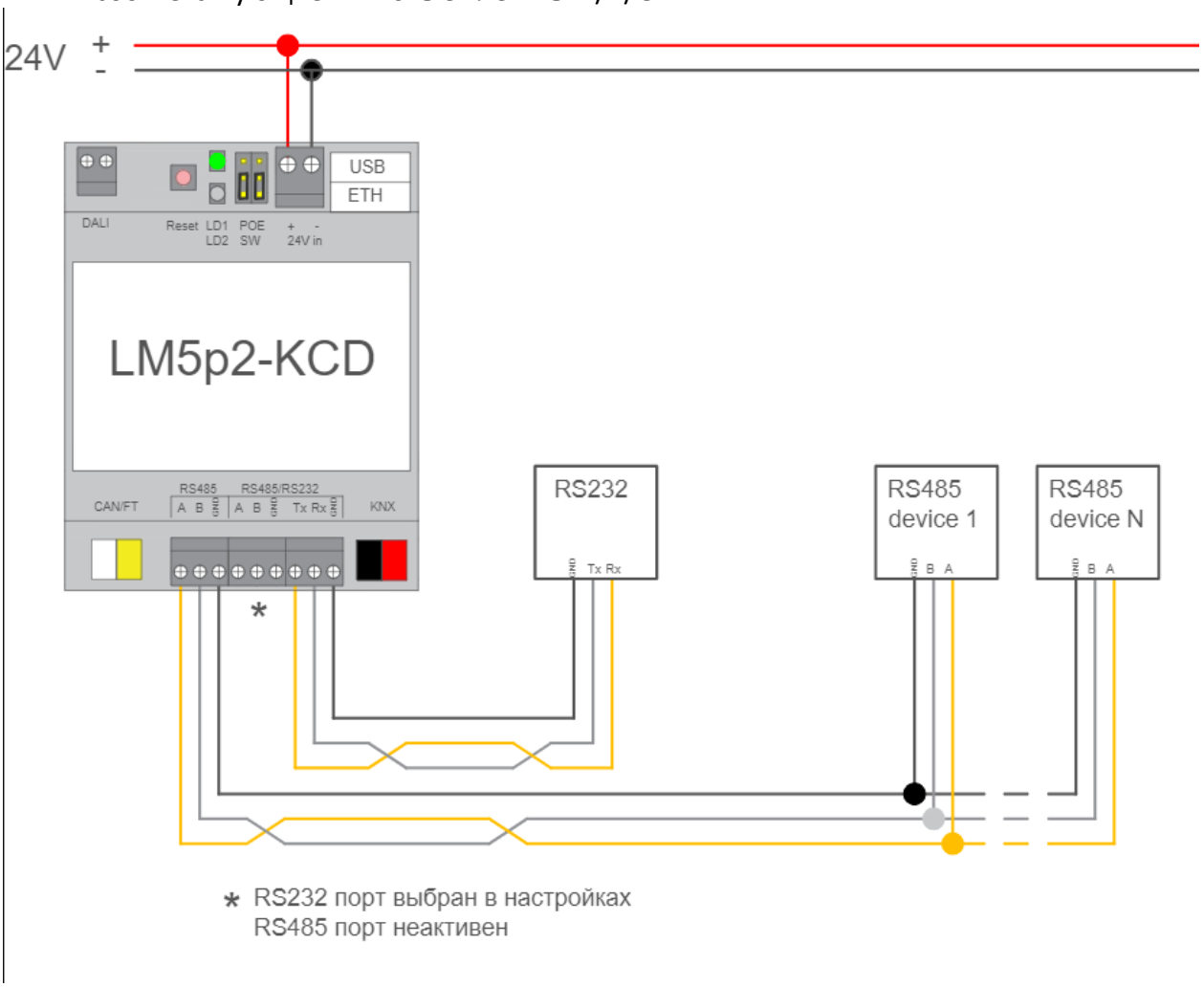
Подключение CAN



Подключение RS-485

Может быть использовано не более трех RS-485. Первый – предустановленный, второй – программно-переключаемый – работающий как RS-485 или как RS-232:

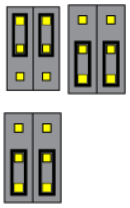
- Если включен full-duplex, он будет работать как RS-232 и должны быть использованы соответствующие винтовые клеммы TX/RX/GND;
- Если включен half-duplex (*), он будет работать как RS-485 и должны быть использованы соответствующие винтовые клеммы A/B/GND.



*В этом случае выбран RS-485, RS-232 неактивен.

Питание

LM5 поддерживает два режима питания:

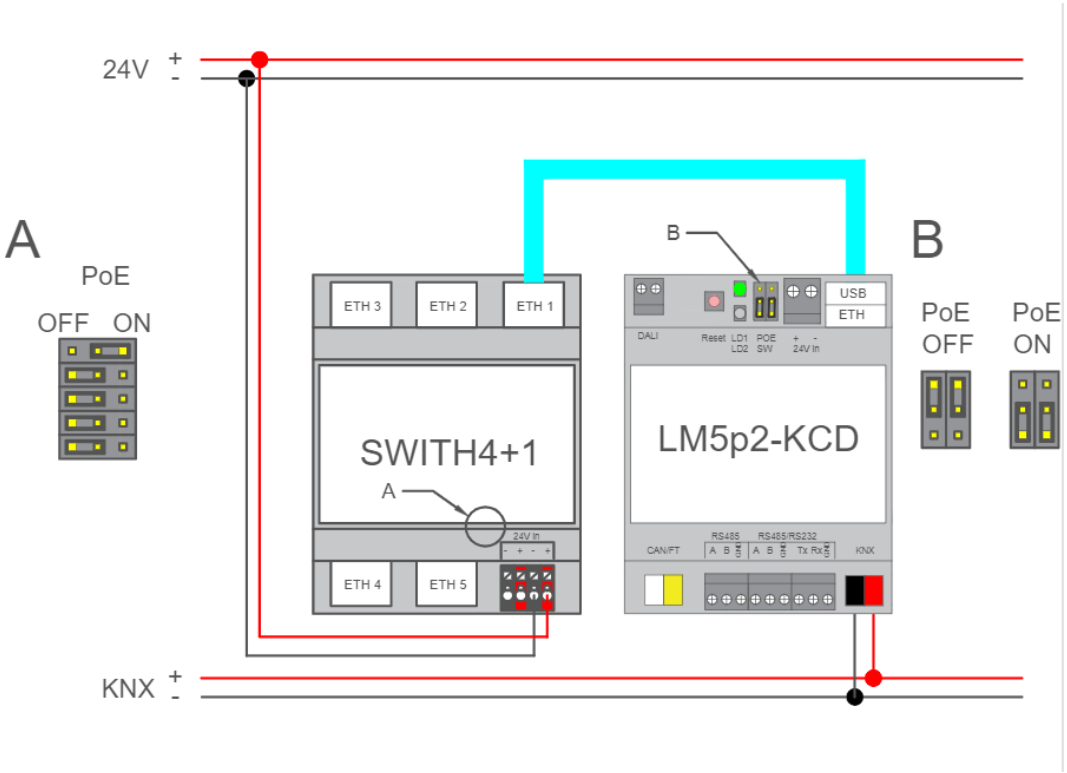


- подача питания через винтовые клеммы (перемычки вверх или вниз);
- пассивное PoE через 24V DC (перемычки вниз).

Внимание! Допускается использовать только один вид питания

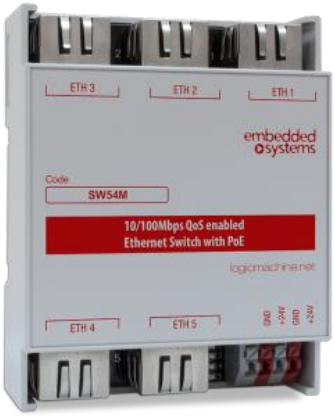
контроллера: или пассивное PoE, или через клеммы 24V! Одновременное подключение двух видов питания может привести к выходу устройства из строя, так же как и подключение к активному PoE!

Пожалуйста, обратите внимание: существует два типа PoE-переключателей/адаптеров – пассивные и активные (802.3af). В пассивном режиме 4 жилы кабеля Ethernet используются для питания, а ещё 4 жилы – для передачи данных. В активном режиме питание и передача данных могут происходить совместно.

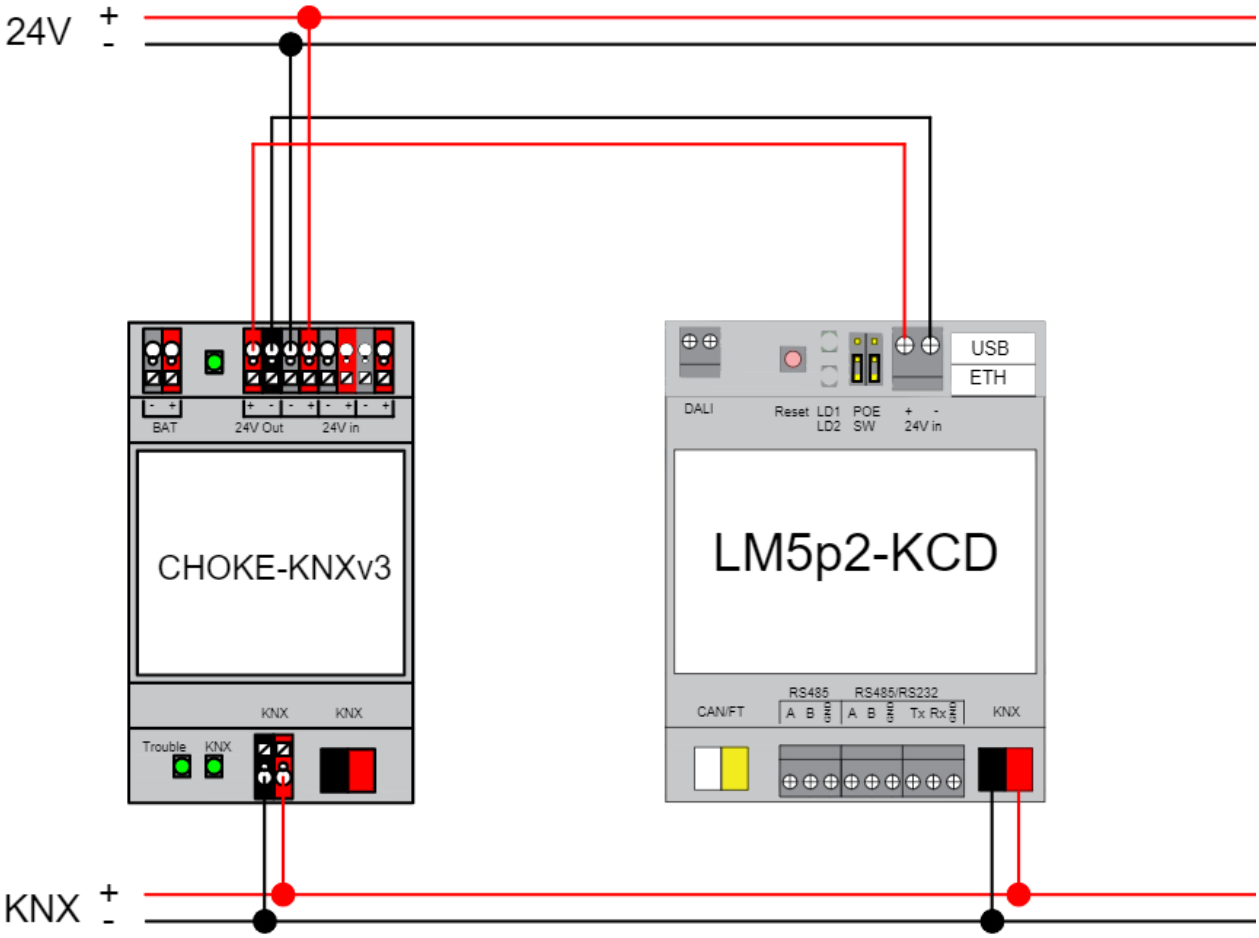


Пассивный PoE-коммутатор

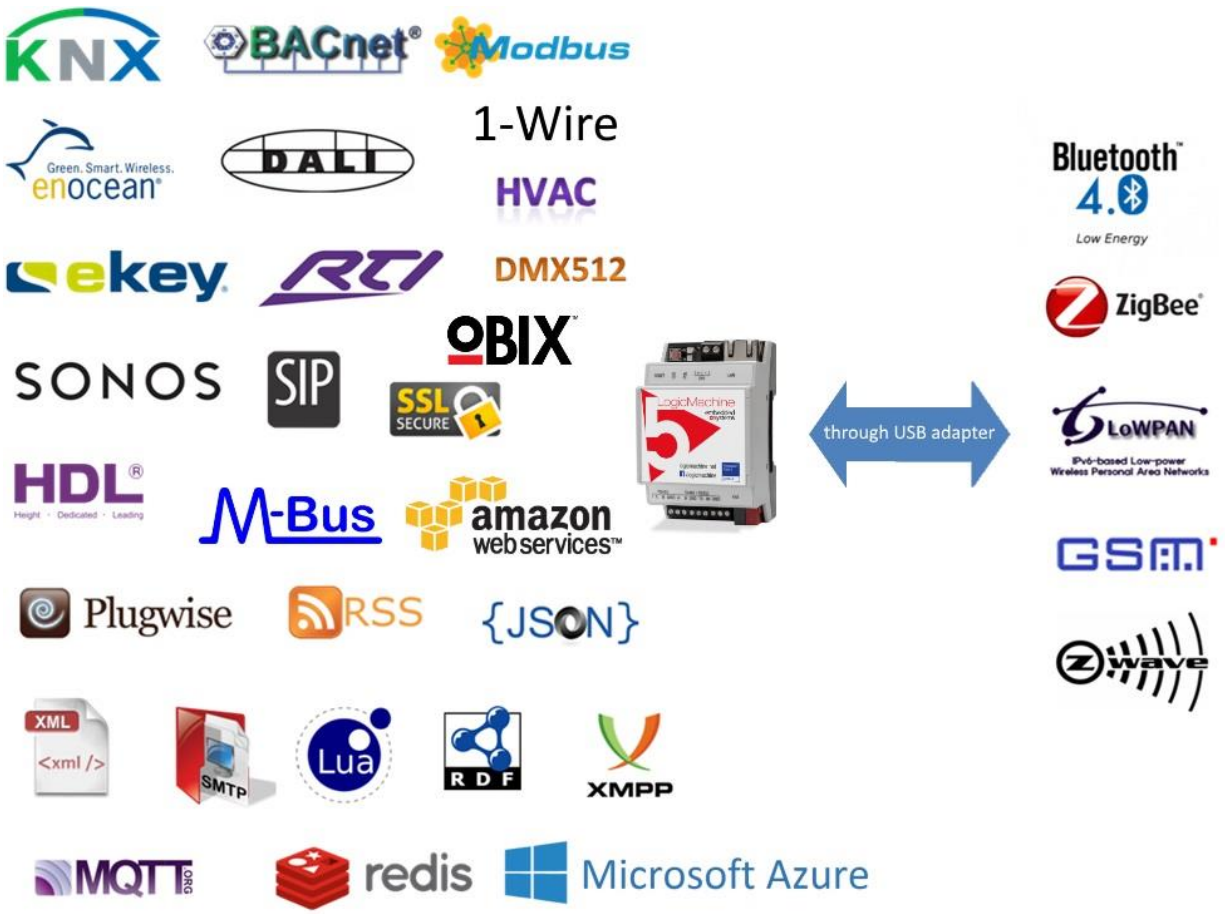
Пассивные PoE-адаптеры



Подключение KNX



2. Поддерживаемые стандарты



LogicMachine совместим со следующими стандартами:

- KNX/EIB TP, KNXnet/IP;
- Modbus TCP, Modbus RTU, Клиент/Сервер;
- BACnet IP, Клиент/Сервер;
- GSM (через USB-модем) для отправки SMS-уведомлений и управления с помощью SMS-команд;
- Bluetooth 3.0 и 4.0 (через USB-адаптер);
- DMX512 (встроенная, через RS485);
- DALI;
- 1-Wire;
- AllJoyn;
- Ekey, биометрическая система доступа (RS485);
- HVAC-системы могут управляться через RS485/Ethernet-интерфейсы с помощью скриптов;
- SMTP/Email, SSL;
- SIP;
- XML (экспорт значений объектов, предупреждений или ошибок; интеграция с Fidelio);
- RSS (чтение содержимого вкладок “Ошибки” и “Предупреждения”);
- JSON, XMPP;
- MQTT;
- REDIS;
- и другие.

Система разработана так, что стандарты могут взаимодействовать друг с другом, т. е. LogicMachine может работать как шлюз BACnet-DALI или Modbus-GSM и т.п.

3. Краткое руководство по запуску

3.1. Подключение

- Установите устройство на DIN-рейку;
- Подключите шину KNX;
- Подключите питание 24V к устройству (красная клемма – 24V+, серая клемма – GND);
- Подключите кабель Ethernet, идущий от ПК.

3.2. Параметры для входа по умолчанию

Логин	admin
Пароль	admin
IP адрес	192.168.0.10
Маска сети	255.255.255.0

Доступ к устройству можно получить, открыв веб-браузер и введя IP-адрес устройства **HTTP://IP**

Защищённое соединение с устройством доступно по адресу **HTTPS://IP:Port**

3.3. Заводские настройки

Вы можете перезагрузить устройство, либо сбросить его до заводских настроек с помощью кнопки Reset:

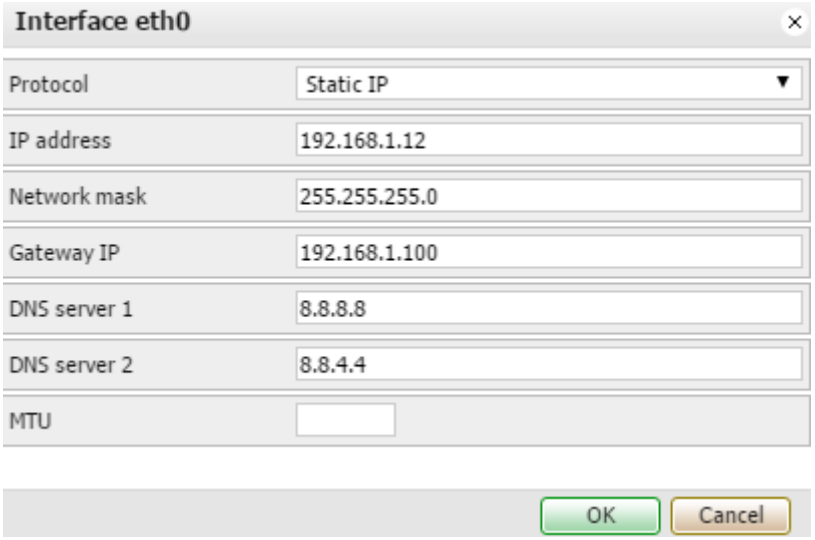
- *Нажать и держать меньше 10 секунд* – перезагрузить устройство;
- *Нажать и держать дольше 10 секунд* – сбросить настройки IP-подключения до заводских;
- *Нажать и держать дольше 10 секунд, после чего снова нажать и держать дольше 10 секунд* – полный сброс настроек до заводских.

Для получения дополнительной информации смотрите: http://openrb.com/discover-ip-of-logic-machine-or-streaming-player/

3.4. Настройки IP

В окне *System configuration* → *Network* → *Interfaces* нажмите на нужный интерфейс, чтобы

изменить настройки IP.



- ***Protocol*** – протокол для адресации:
  - ***Static IP*** – постоянный IP-адрес. По умолчанию – 192.168.0.10;
  - ***DHCP*** – использовать протокол DHCP для получения IP-адреса;
    - ***Current IP*** – IP, полученный от DHCP-сервера. Это поле видно только в том случае, если IP-адрес был получен;
- ***Network mask*** – маска сети. По умолчанию – 255.255.255.0 (/24);
- ***Gateway IP*** – IP-адрес шлюза;
- ***DNS server*** – IP-адрес DNS-сервера;
- ***MTU*** – наибольший размер пакета, который может быть передан в протоколе связи. По умолчанию – 1500.

Если настройки были изменены, в правом верхнем углу должна появиться следующая иконка.

Её следует нажать для сохранения изменений.

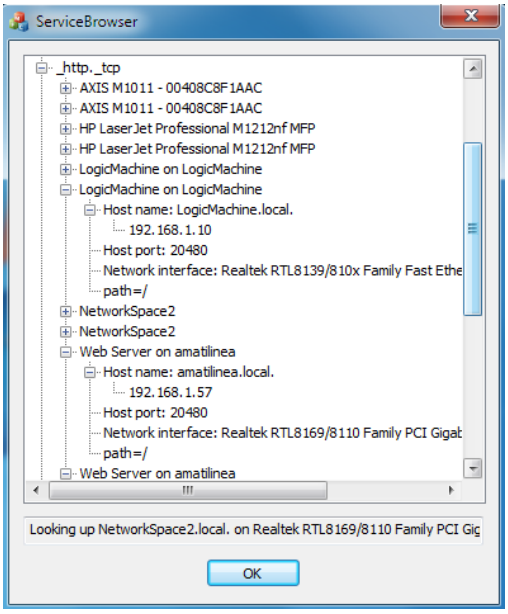
3.5. Как узнать IP-адрес LogicMachine

В LM встроена утилита zeroconf, так что, чтобы узнать IP-адрес устройства, можно использовать следующие программы:

- Windows PC – *ServiceBrowser;*
- Linux PC – *Avahi;*
- Android – *ZeroConf Browser;*
- iOS – *Discovery.*

<u>Windows PC</u>

Простейший способ узнать IP-адрес – использовать программу **ServiceBrowser**, которую можно скачать здесь:
http://marknelson.us/2011/10/25/dns-service-discovery-on-windows/



Linux PC

Утилита **Avahi**, которую можно скачать здесь:
www.avahi.org

Android

Бесплатное приложение **ZeroConf Browser**, доступное в *Play Store*:
https://play.google.com/store/apps/details?id=com.grokkt.android.bonjour&hl=en

iOS/Mac OS

Бесплатное приложение **Discovery**, доступное в *App Store*:
https://itunes.apple.com/en/app/discovery-bonjour-browser/id305441017?mt=8

Для iPad можно использовать версию приложения для iPhone.

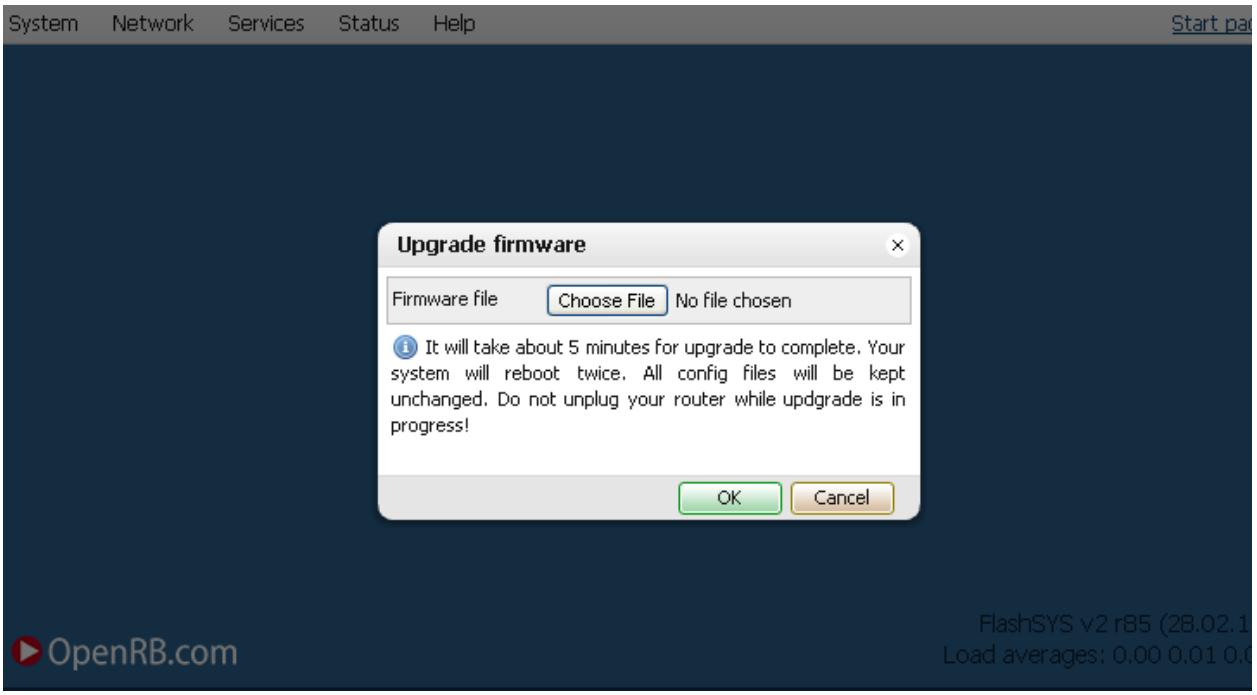
3.6. Обновление прошивки

Примечание! Перед каждым обновлением рекомендуется сделать резервную копию ваших скриптов, визуализации и объектов через *Logic Machine → Tools → Backup.*

Примечание! *После обновления мы рекомендуем очистить кэш браузера.*

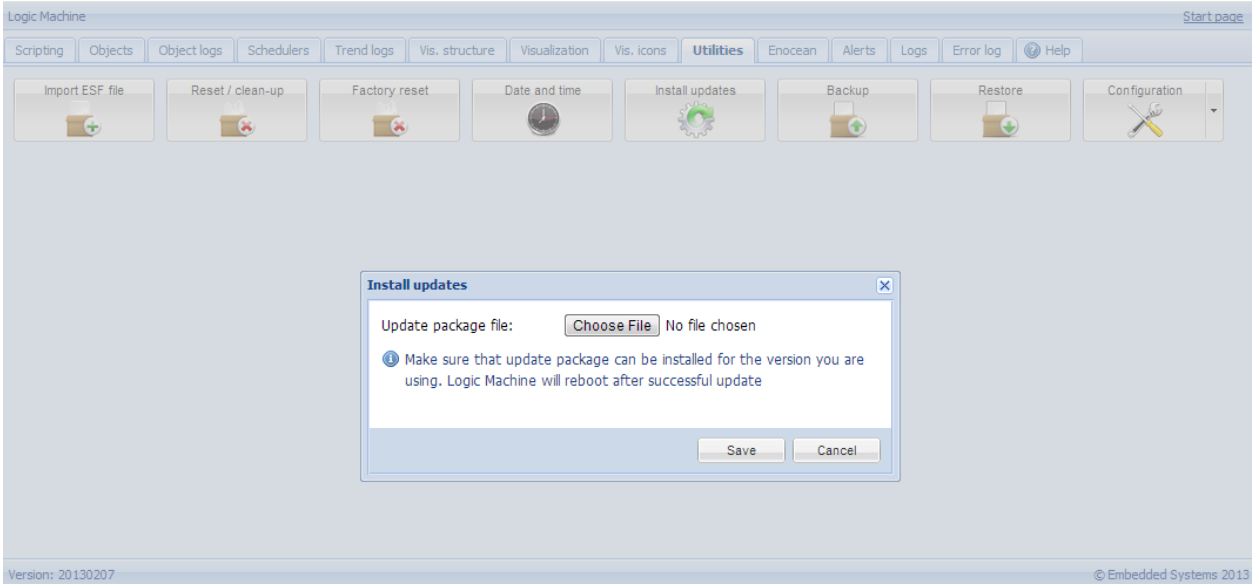
Скачайте обновления прошивки LogicMachine. Они доступны в виде образов на странице поддержки www.openrb.com.

Полное обновление системы производится в *System Configuration → System → Upgrade firmware.*



Незначительное обновление системы или визуализации проводится на вкладке *Utilities*

функцией *Install updates*. После выбора *.LMU файла нажмите кнопку *Save*. Через 5 секунд устройство перезагрузится, и обновление будет установлено.



3.7. Краткое руководство – приложение MOSAIC для быстрой визуализации

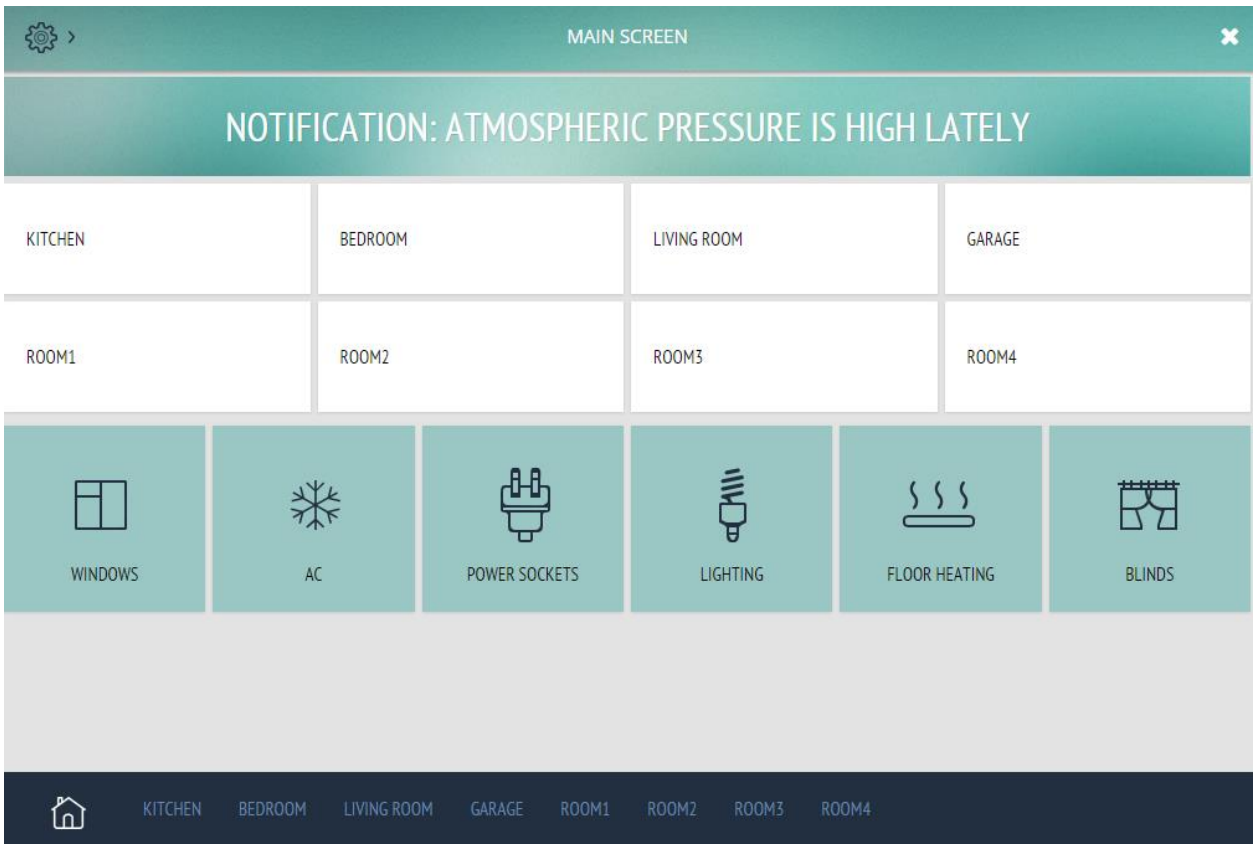
Приложение Mosaic – самый быстрый способ создать визуализацию.

<u>Начало работы</u>

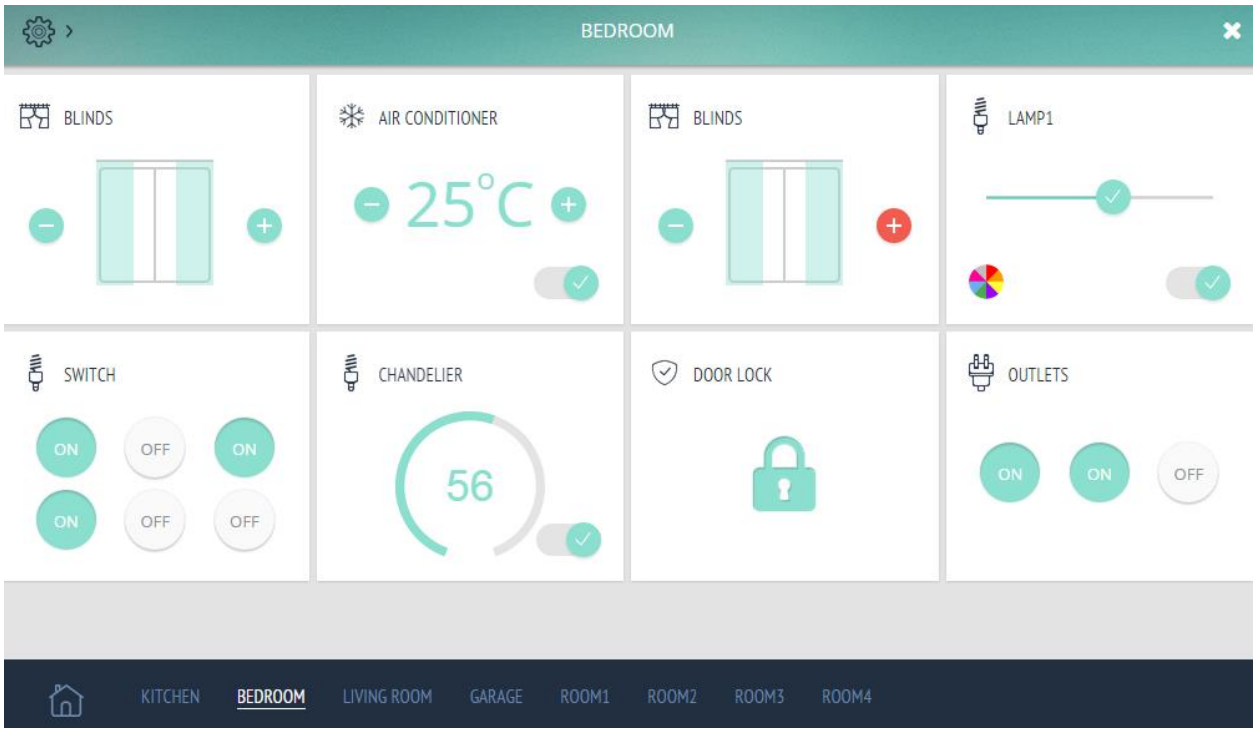
1. Откройте веб-интерфейс устройства, введя его IP-адрес в вашем веб-браузере.
2. Нажмите на Mosaic-иконку режима Разработчика. Вы увидите интерфейс контроллера с чистым шаблоном.
3. Когда вы войдёте в режим Разработчика, вы будете готовы к созданию визуализации.
4. После создания визуализации зайдите в режим Клиента, кликнув по иконке.

<u>Стартовая страница в режиме Клиента</u>

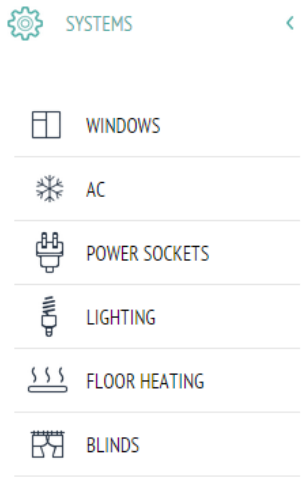
Это первый экран, который вы увидите, открыв приложение Mosaic. Первая страница содержит *Notifications* (уведомления), зоны управления по комнатам и по функциям. Объекты можно найти по комнатам, либо по выполняемым функциям.


MAIN SCREEN
NOTIFICATION: ATMOSPHERIC PRESSURE IS HIGH LATELY
KITCHEN
BEDROOM
LIVING ROOM
GARAGE
ROOM1
ROOM2
ROOM3
ROOM4
WINDOWS
AC
POWER SOCKETS
LIGHTING
FLOOR HEATING
BLINDS
KITCHEN
BEDROOM
LIVING ROOM
GARAGE
ROOM1
ROOM2
ROOM3
ROOM4

Например, перейдя в одну из комнат, вы увидите следующее:



Нажав кнопку настроек, вы увидите ярлыки для перехода по функциям:

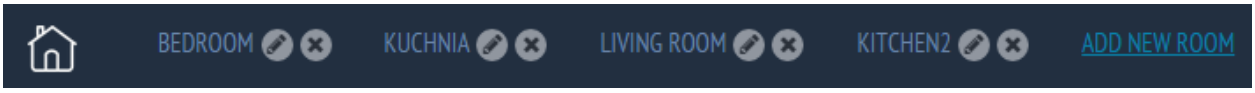


В настройках вы также можете изменить цветовую тему визуализации.

Доступ к расширенной визуализации с большим количеством виджетов будет предоставляться по ежемесячной подписке.

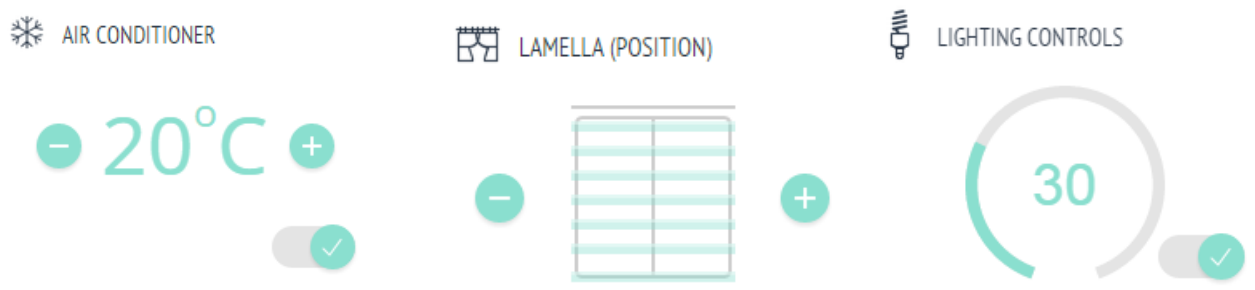
Построение структуры / Режим Разработчика

На нижней панели вы можете создать свою структуру здания, добавив в неё новые комнаты. Эта панель также предназначена для навигации между комнатами для конечного пользователя. Просто нажмите *Add new room,* затем введите название комнаты и нажмите *Enter*. Вас переместит на экран новой комнаты. Теперь вы можете начать заполнять её виджетами. Если вы захотите переименовать или удалить комнату, нажмите левой кнопкой мыши на названии комнаты и выберите, что вам нужно.

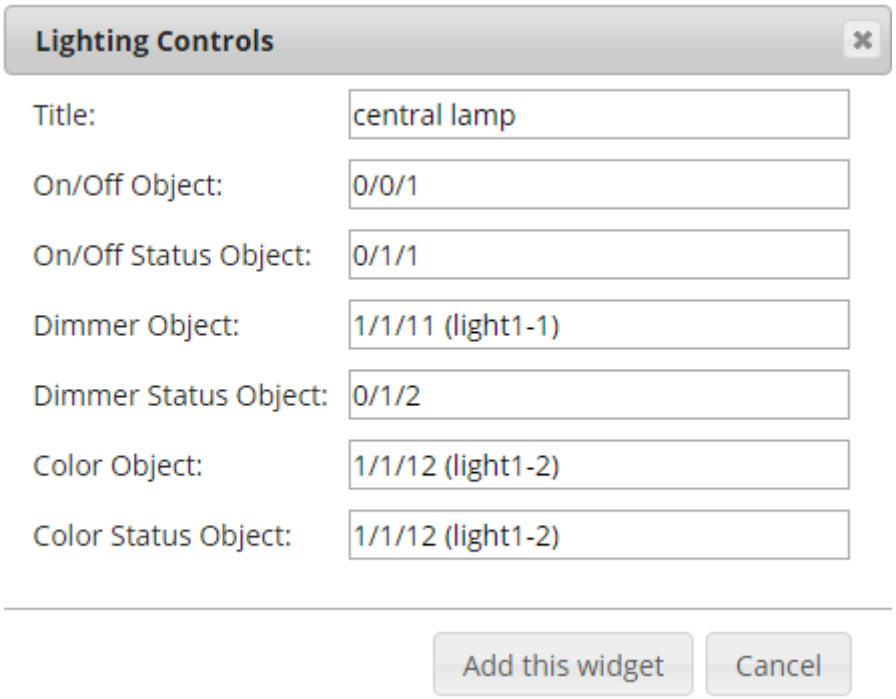


Виджеты

Для добавления нового виджета на стартовую страницу или страницу комнаты нажмите *Add new widget* в правом верхнем углу для открытия панели виджетов. На этой панели вы сможете добавить необходимый вам виджет, нажав на него. Всплывающее окно со свойствами виджета автоматически откроется при добавлении.



В этом окне вы можете задать название виджета и связи с объектами KNX для управления. После настройки виджета нажмите *Add this widget*. Виджет появился на странице. Нажмите правой кнопкой мыши, чтобы посмотреть/изменить его свойства или удалить его со страницы.



Уведомления

В режиме Клиента существует специальное поле для уведомлений, с помощью которого вы сможете посылать уведомления или предупреждения. Используйте переменную хранилища *mosaic-message* для записи уведомлений.

Пользовательские виджеты

Отсюда вы можете скачать руководство для создания пользовательских виджетов:
http://openrb.com/wp-content/uploads/2016/11/Mosaic_widgets_eng.pdf

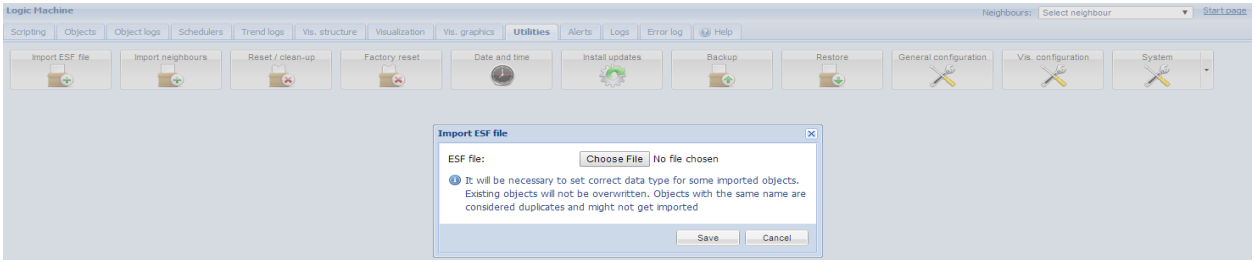
Отсюда можно загрузить несколько примеров пользовательских виджетов вместе с инструкцией по их установке:
http://forum.logicmachine.net/showthread.php?tid=122&pid=2651#pid2651

3.8. Краткое руководство – создание визуализации для iPad/PC

Вы можете загрузить готовые резервные копии LM здесь:
http://forum.logicmachine.net/showthread.php?tid=196

Импорт объектов

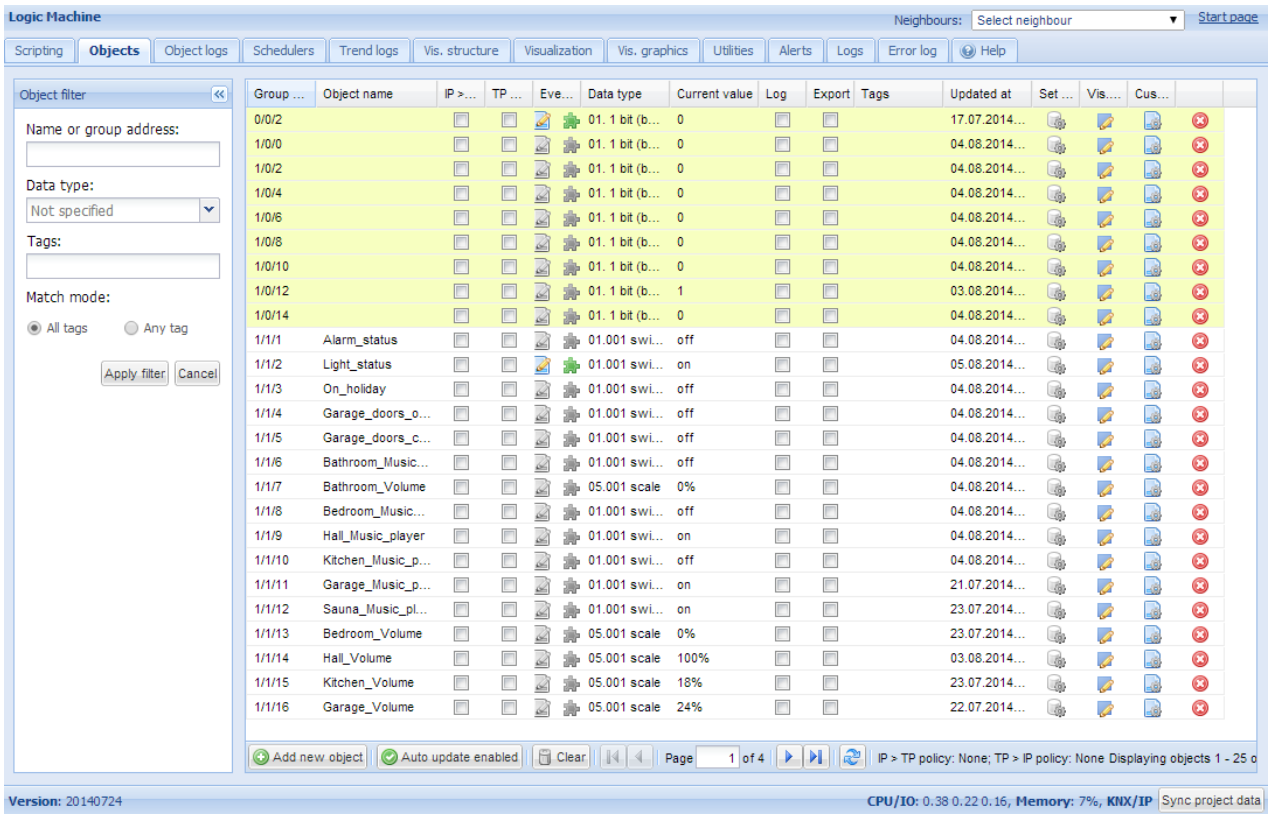
Самый быстрый способ – импортировать *.ESF файл из ETS через *Logic Machine → Utilities → Import ESF file*.



Если подключить LM к шине, устройство автоматически найдет объекты (жёлтые) на вкладке *Objects*, когда по их адресу пройдёт телеграмма. Объекты также могут быть добавлены

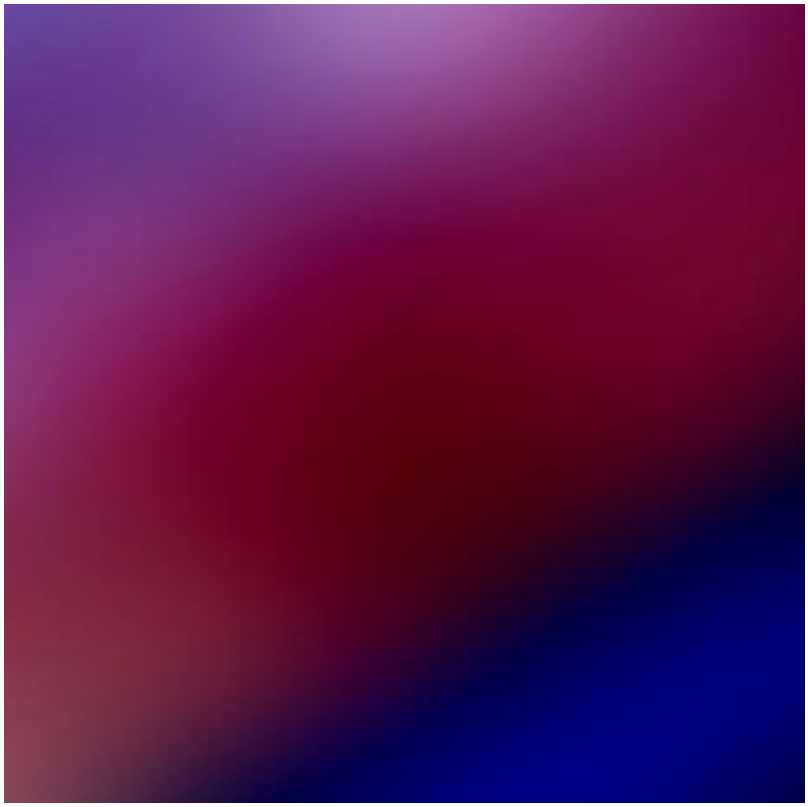
вручную.

Подготовка графики

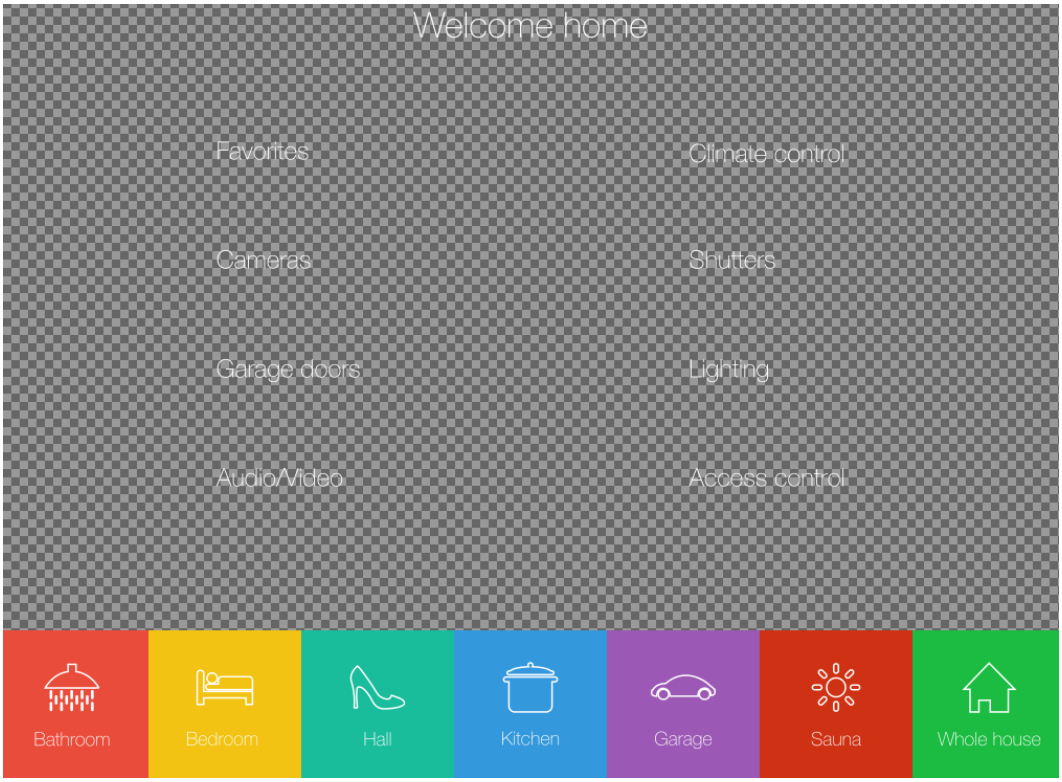


Вы можете использовать готовые изображения или создать новое в любом графическом редакторе (например, Adobe Illustrator). В этом примере мы используем созданный в Illustrator дизайн в формате SVG (для масштабируемости изображения без потери качества).

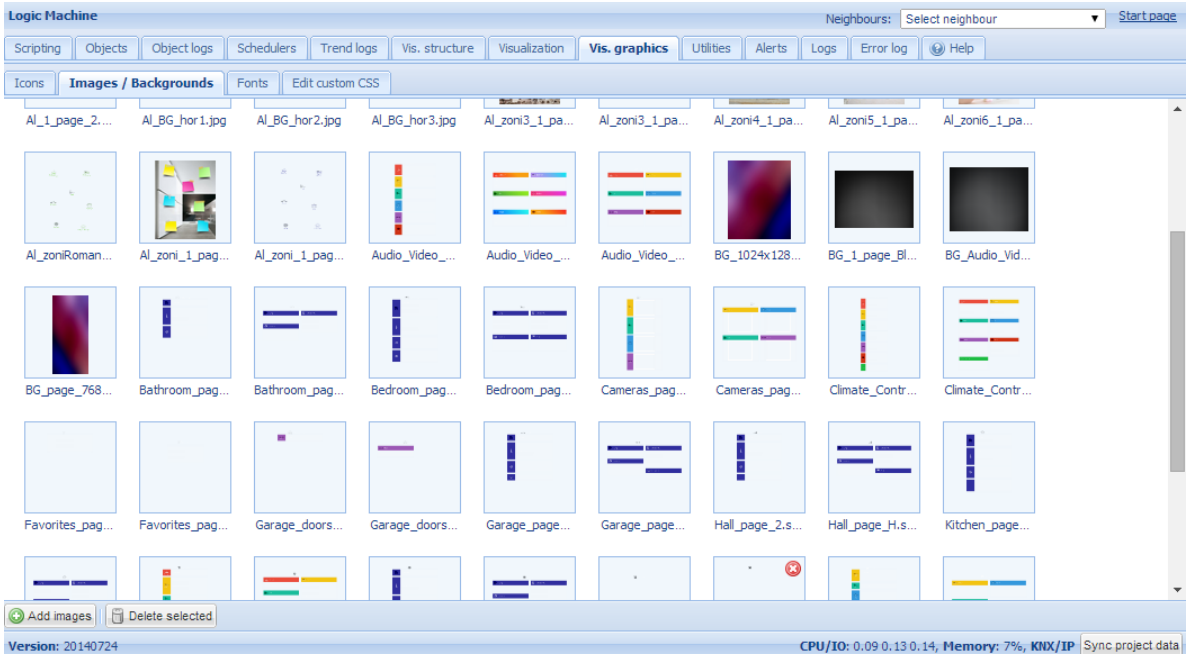
a) Основной фон, изменяемый при необходимости:



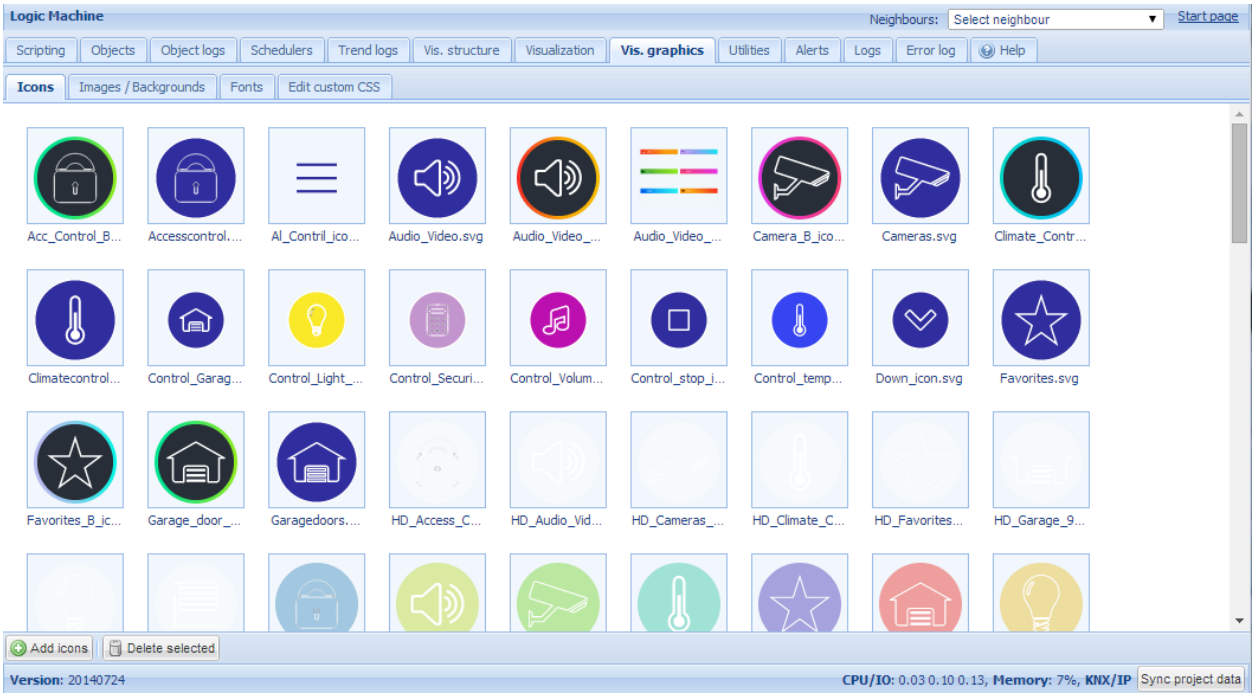
b) Передний план, который останется неизменным:



Добавьте оба файла в *Logic Machine* → *Vis. Graphics* → *Images/Backgrounds.*



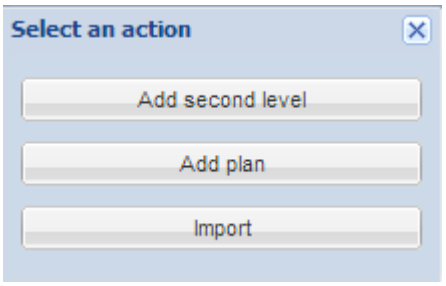
Приготовьте набор иконок (желательно в формате SVG) и добавьте их в *Logic Machine → Vis. Graphics → Icons*. Можно использовать набор стандартных иконок LogicMachine.



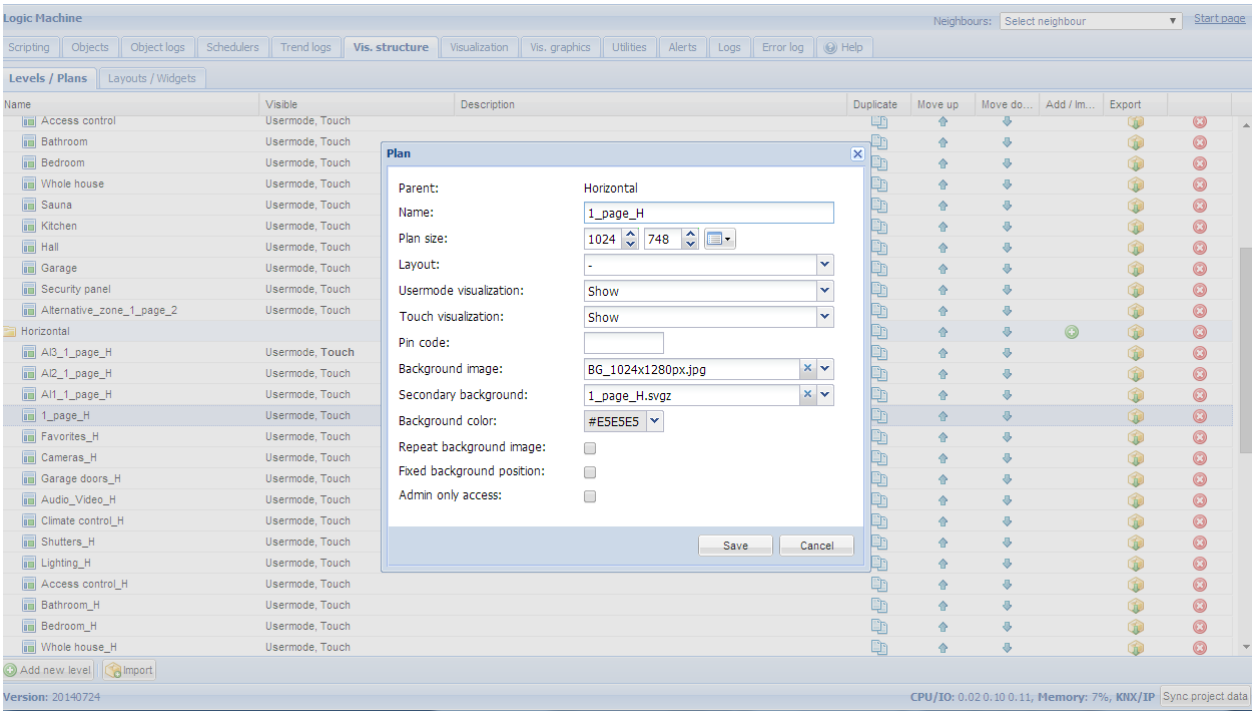
<u>Создание структуры “этажей” и добавление объектов на план</u>

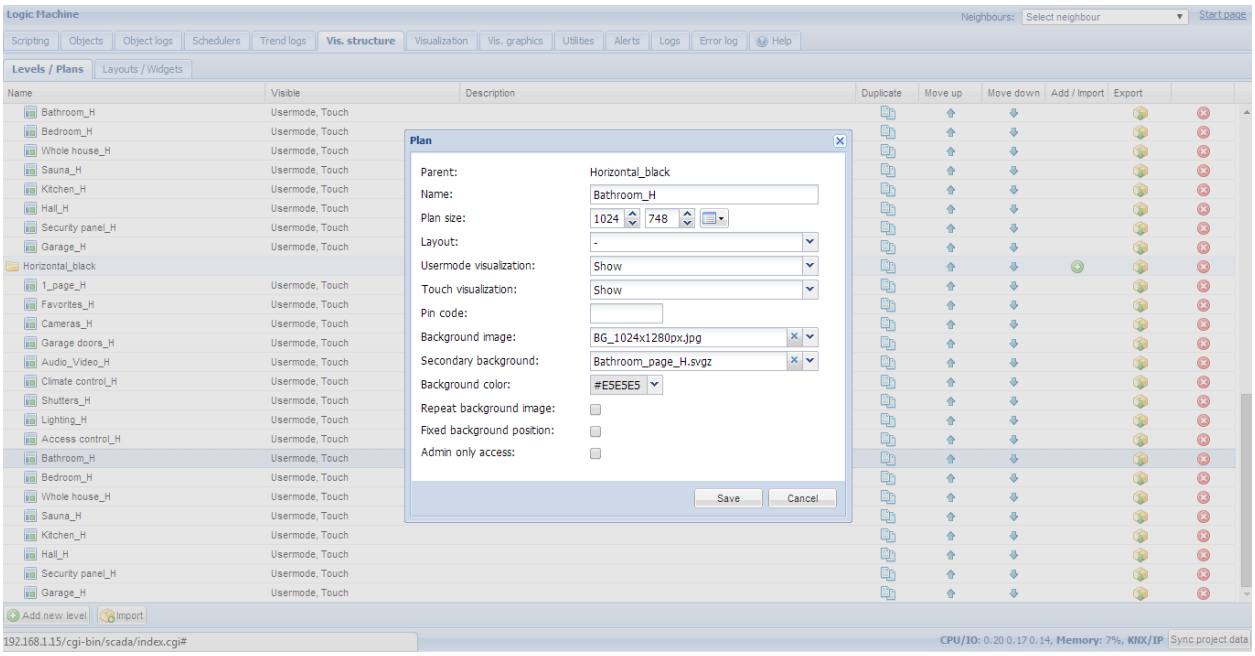
В меню *Logic Machine → Vis.structure* устанавливается структура визуализации и загружаются

фоновые изображения для планов. Используйте иконку для добавления “этажа”.



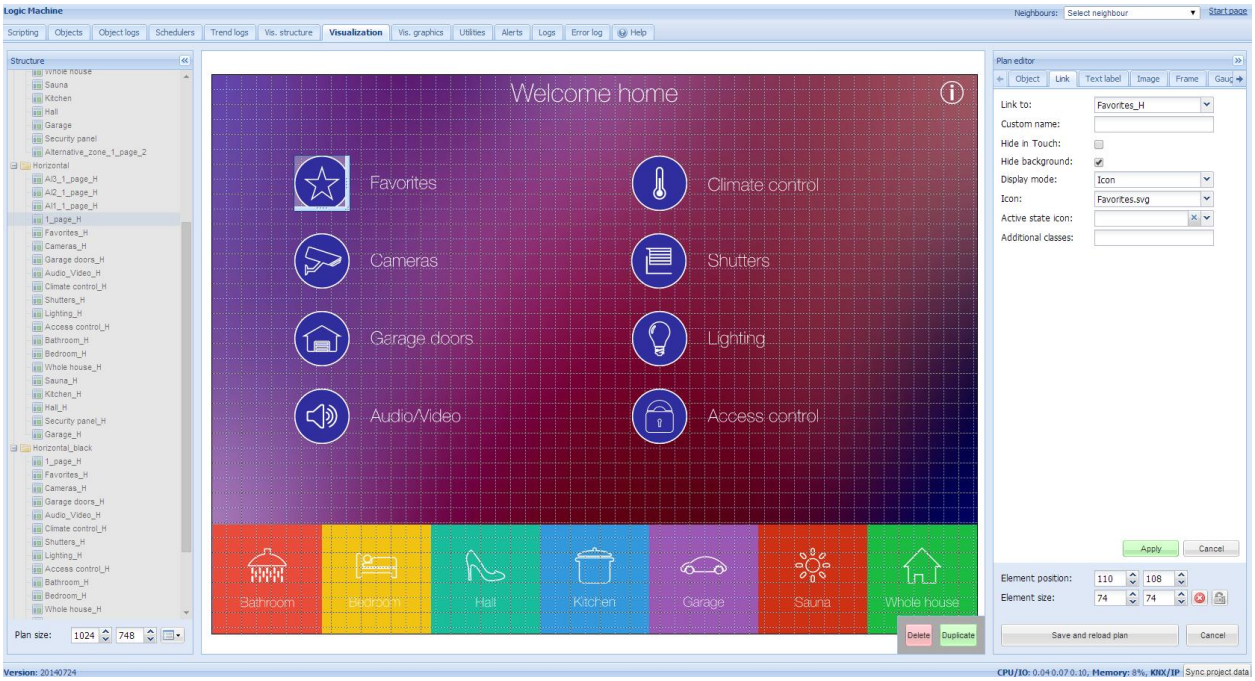
В этом примере мы создадим новые этажи с названиями *1_page_H* и *Bathroom_H*. Первая страница будет навигационной панелью со ссылками на другие комнаты и функции. Выберите разрешение экрана, для которого вы создаёте визуализацию, и фоновые изображения из тех, которые вы добавили ранее.





Добавление объектов на созданный план

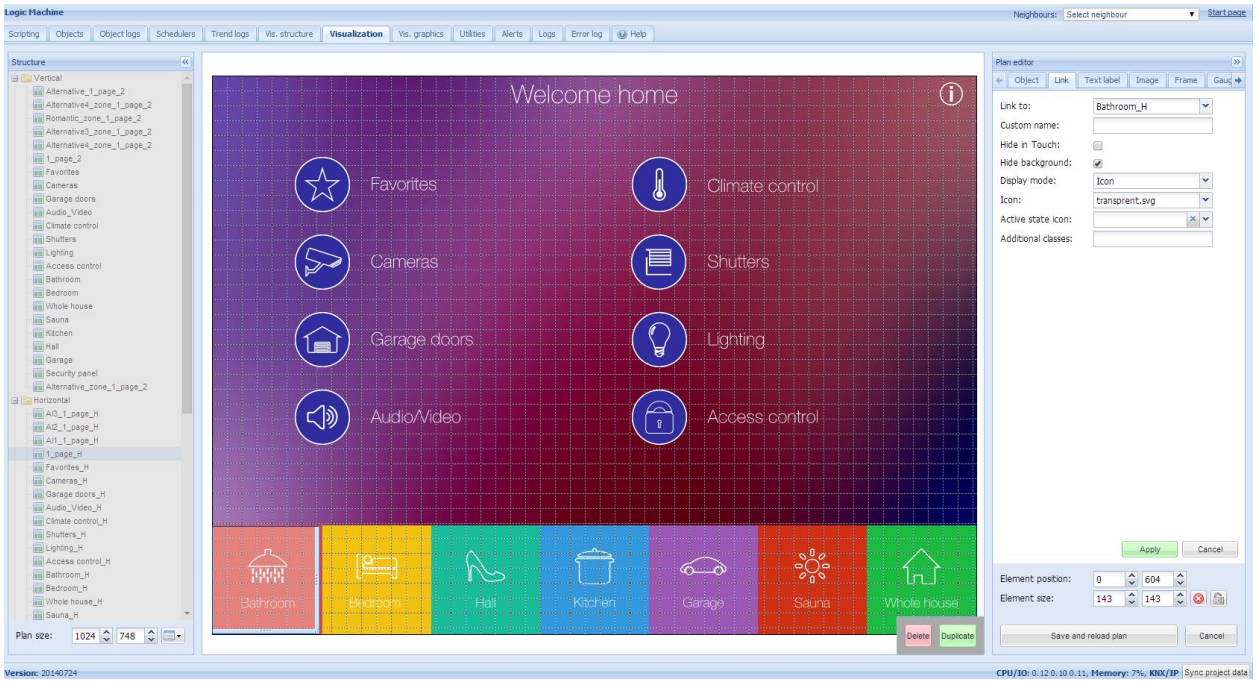
После создания структуры планы отображаются на вкладке *Visualization*, на которой можно добавлять и изменять объекты управления и мониторинга. Обе боковые панели могут быть скрыты нажатием на иконку стрелочки, освободив место для плана на маленьких экранах.



Объекты можно добавлять на план после нажатия кнопки *Unlock current floor plan for editing*. В этом примере мы создадим стартовую страницу визуализации, ведущую к комнатам и

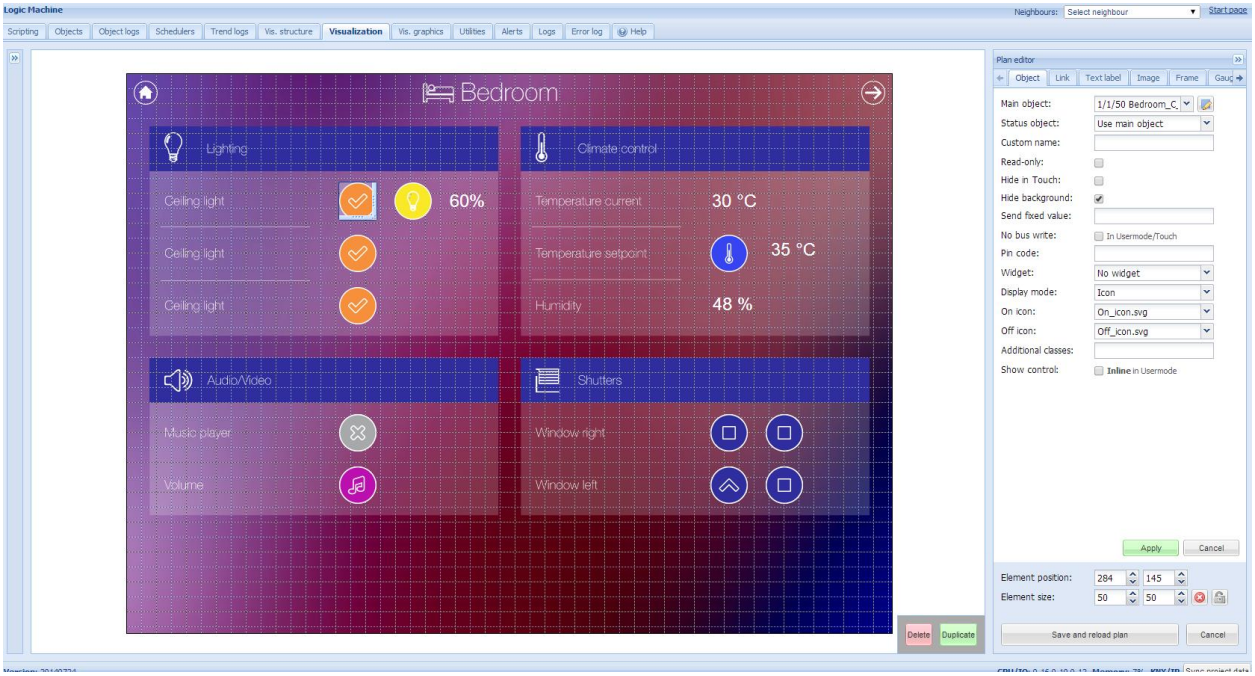
различным функциям. Чтобы добавить ссылку, нужно нажать на вкладку Link, выбрать иконку и подобрать её размер, затем поместить её в нужное место на плане.

В этом примере фон уже содержит иконки, поэтому просто добавьте ссылку с прозрачным изображением поверх иконок фона.



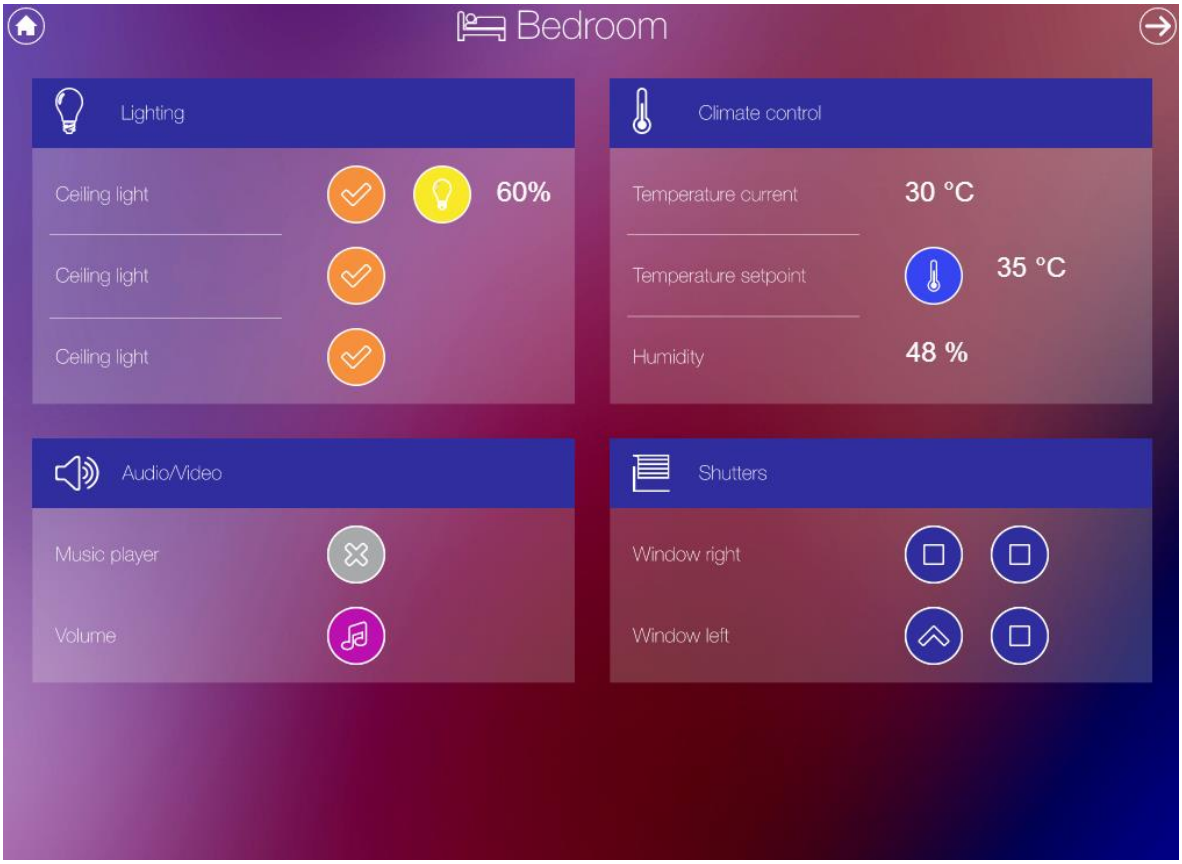
Когда все ссылки добавлены, нажмите кнопку *Save and reload floor plan*.

Таким же способ заполните план Bedroom с объектами с вкладки *Object*.

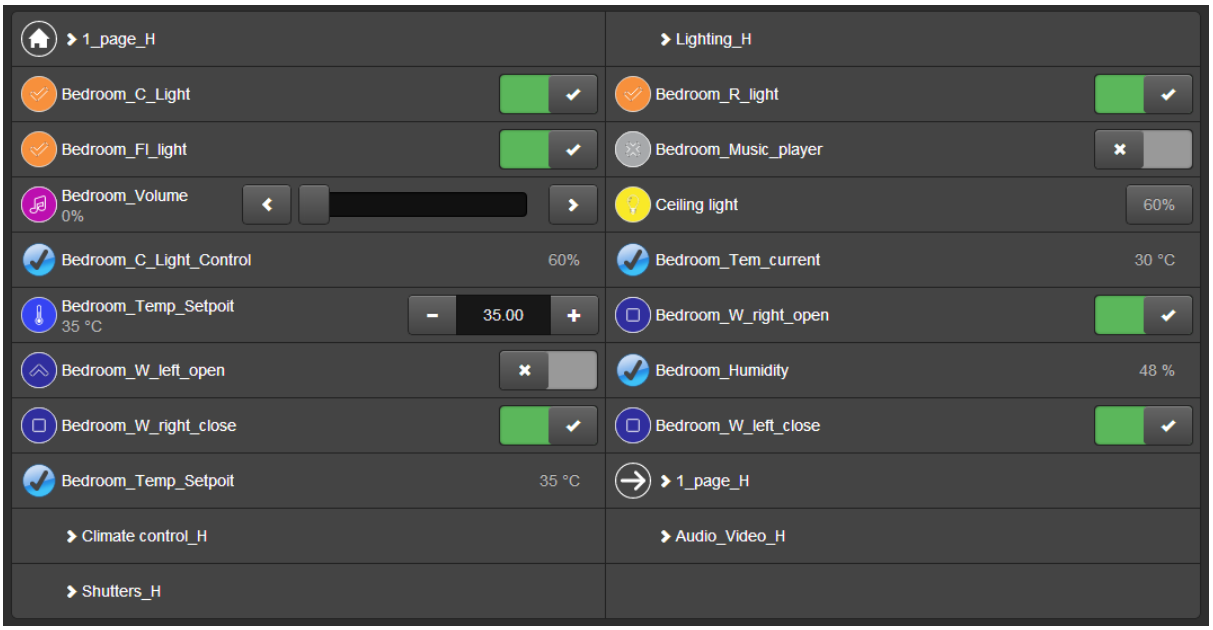


Запуск визуализации на сенсорных устройствах (на примере iPad)

- Убедитесь, что ваш iPad находится в одной сети с LogicMachine;
- В браузере введите IP-адрес Logic Machine (по умолчанию 192.168.0.10);
- Нажмите на *User mode visualization;*
- Сохраните ссылку в виде ярлыка на вашем iPad.



Touch-визуализация также автоматически создается со списком объектов плана.

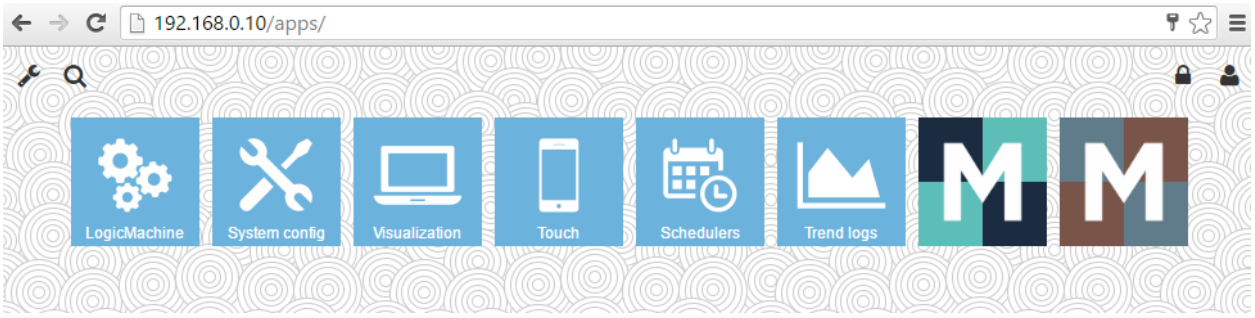

1_page_H
Lighting_H
Bedroom_C_Light
Bedroom_R_light
Bedroom_Fl_light
Bedroom_Music_player
Bedroom_Volume
0%
Ceiling light
60%
Bedroom_C_Light_Control
60%
Bedroom_Tem_current
30 °C
Bedroom_Temp_Setpoit
35 °C
35.00
Bedroom_W_right_open
Bedroom_W_left_open
Bedroom_Humidity
48 %
Bedroom_W_right_close
Bedroom_W_left_close
Bedroom_Temp_Setpoit
35 °C
1_page_H
Climate control_H
Audio_Video_H
Shutters_H

4. Авторизация в графическом интерфейсе

В KNX/EIB LogicMachine по умолчанию применяется IP-адрес 192.168.0.10. Используйте этот адрес в браузере для доступа к устройству.

Примечание! Убедитесь, что ваш ПК подключен к LogicMachine из той же подсети.

После успешной авторизации появится стартовая страница.

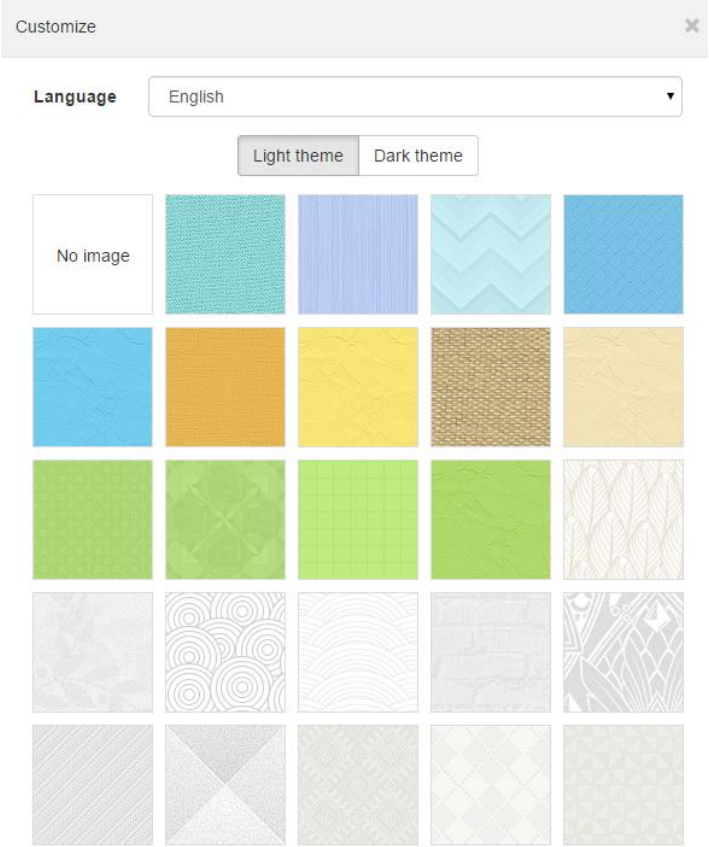


- ***Logic Machine*** – конфигуратор визуализации, скриптов, связей объектов, тревог, объектов KNX;
- ***System config*** – настройки IP и KNX;
- ***Visualization*** – разработанная визуализация;
- ***Touch*** – визуализация для сенсорных устройств (iPhone/iPod/iPad/Android);
- ***Schedulers*** – пользовательские скрипты по расписанию;
- ***Trend logs*** – графики статистики;
- ***Mosaic app*** – приложение Mosaic для создания визуализации.

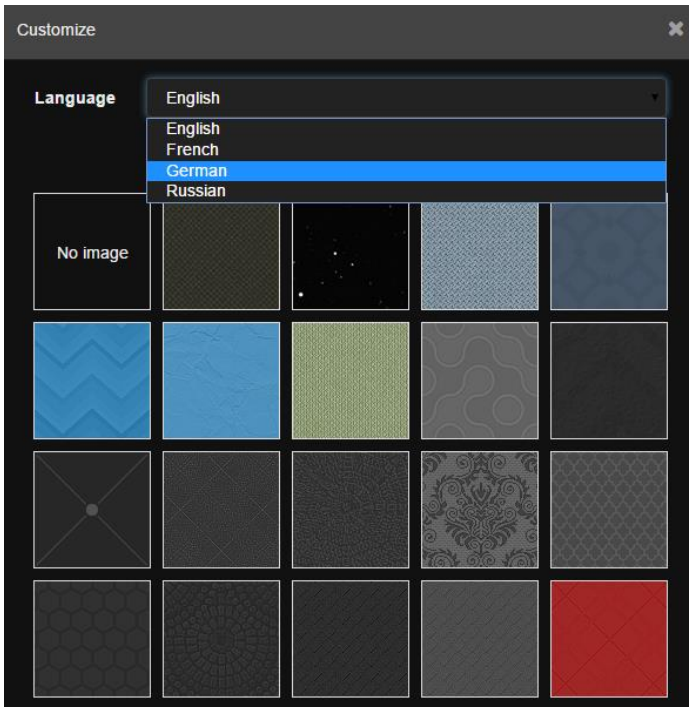
Первая страница интерфейса – конструктор приложений, в котором вы можете изменить приложения для конкретного устройства: поменять цвет фона, установить или удалить приложения и изменить их порядок. Заметьте, что эти настройки индивидуальны для каждого устройства, с которого вы подключаетесь.

4.1. Изменение фона / языка

При нажатии на иконку *Customize*  откроется окно, в котором вы сможете изменить фоновое изображение, отображаемое на вашем устройстве.



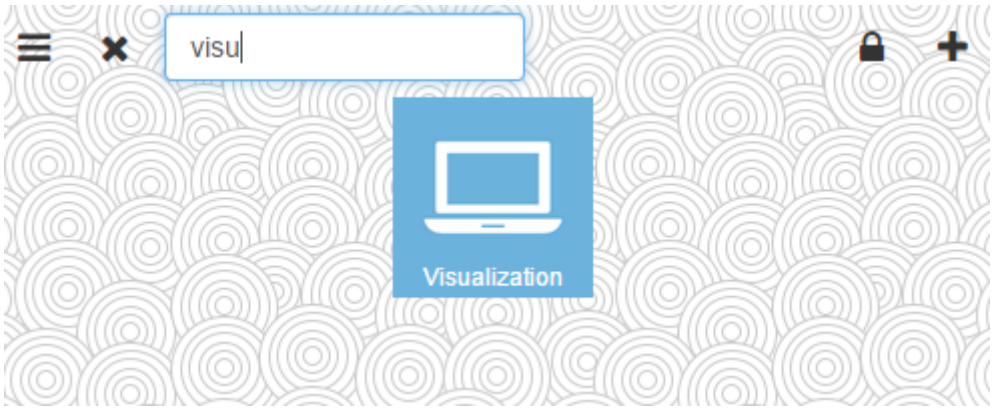
Также вы можете изменить язык интерфейса, нажав на выпадающее меню *Language*.



Выбранные изменения автоматически вступят в силу.

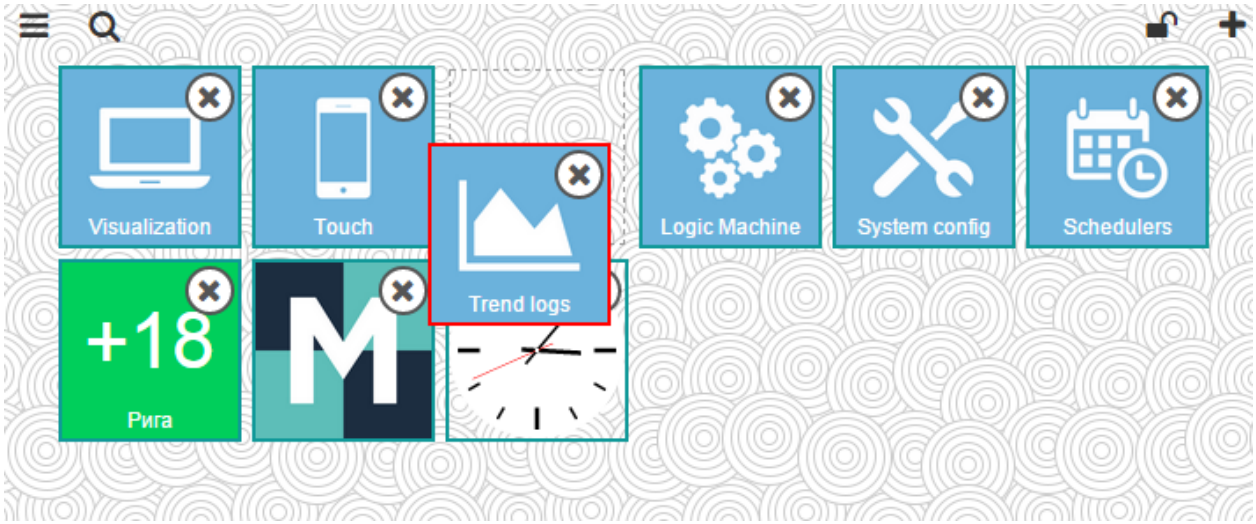
4.2. Поиск приложений

Нажав на кнопку *Filter* в левом верхнем углу, вы можете быстро найти установленные приложения по названию.



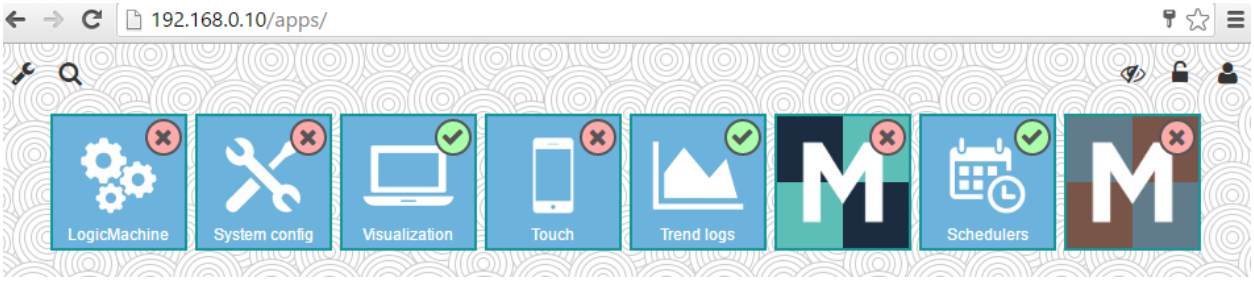
4.3. Разблокировка экрана для изменения порядка и скрытия приложений

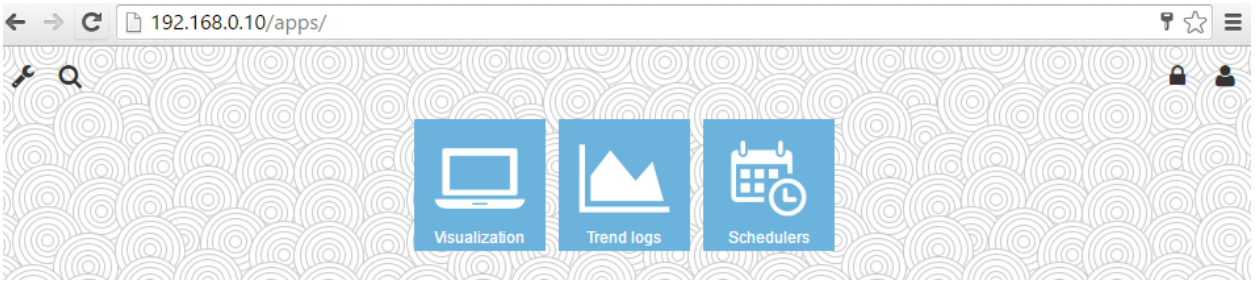
Изменение порядка расположения иконок доступно на разблокированном экране. Для этого нажмите на кнопку *Lock/unlock grid* .



Если вы нажмёте на кнопку *Show/hide apps* , вы сможете скрыть/показать приложения с вашей главной страницы. Эта функция может быть отключена администратором.

После внесения необходимых изменений снова нажмите на кнопку *Lock/unlock grid* , чтобы увидеть изменённую страницу.

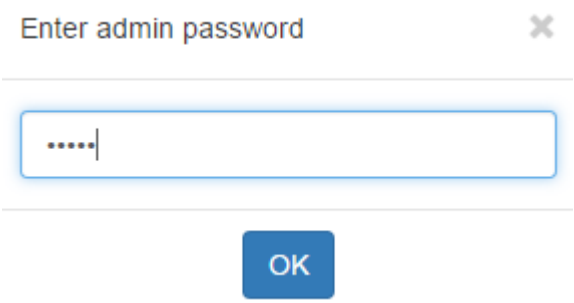




4.4. Режим Администратора: установка/удаление/настройка приложений

Нажмите на кнопку *Login* и введите пароль (по умолчанию *admin*).

Password
admin



Изменение пароля Администратора

В режиме Администратора нажмите на кнопку “Настройки” и выберите *Change admin password*.

В этом меню также находится опция *Allow users to show/hide apps,* которая включает/выключает возможность изменения видимости приложений для пользователей, как показано в 4.3.

Изменение страницы для пользователей

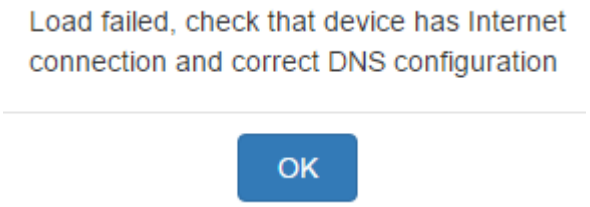
В режиме Администратора вы можете изменить фоновое изображение, изменить видимость и порядок приложений так же, как в режиме пользователя (смотрите 4.3).

После внесения изменений сохраните их с помощью кнопки *Save view*.

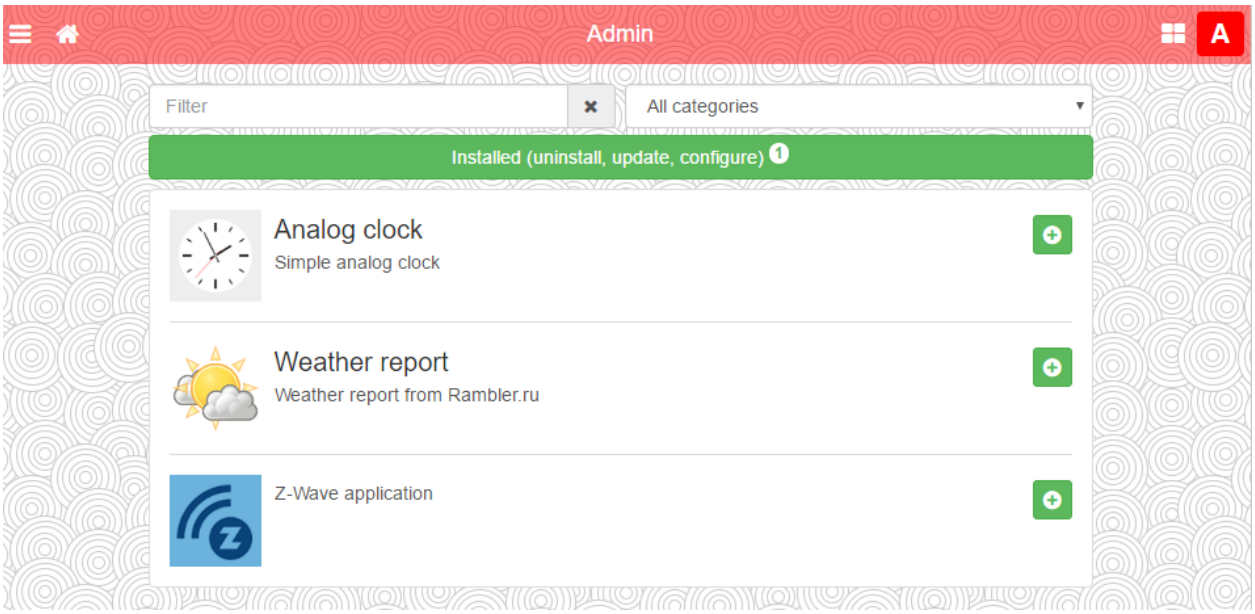
Установка / удаление приложений

Нажмите на иконку , чтобы открыть окно администрирования приложений. Если вы видите

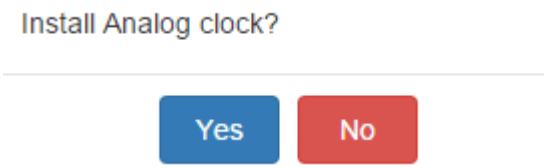
следующее окно, проверьте, правильно ли настроены IP-адрес/шлюз/DNS-сервера и имеет ли LogicMachine доступ к Интернету (*System config* → *Network* → *Interfaces*).



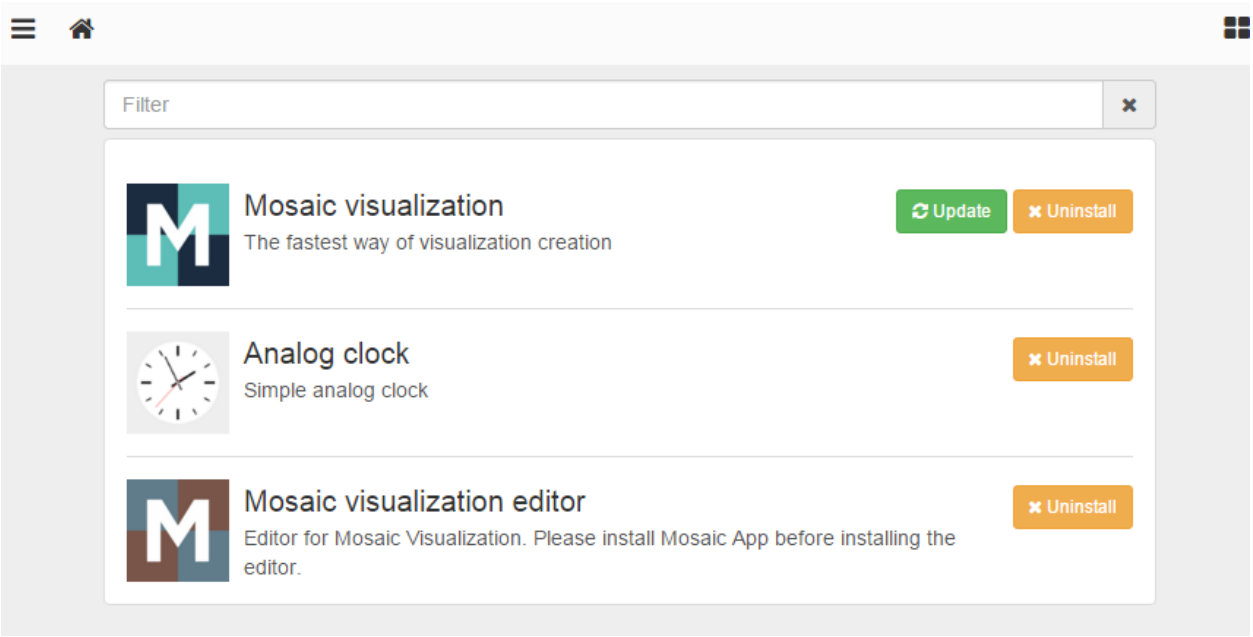
На странице управления приложениями вы увидите доступные вам приложения.



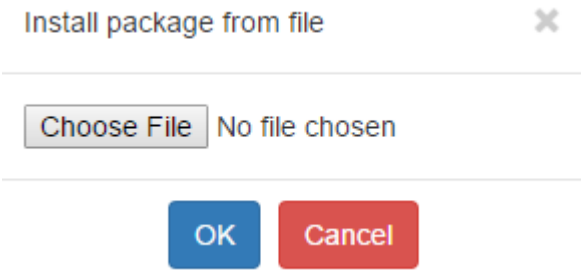
Вы можете добавить приложение на главную страницу, нажав иконку добавления  и подтвердив свой выбор.



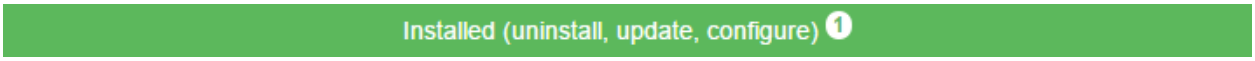
Установленные приложения отображаются во вкладке *Installed*, на которой вы при необходимости можете удалить ненужные приложения.



Также вы можете установить приложение из файла, нажав на *Install from file.*



Для обновления приложения нажмите на соответствующую иконку  . Также обновиться можно с главной страницы управления, нажав на следующую иконку:



Нажав на иконку  , вы вернётесь на стартовую страницу.

Выход из режима Администратора

Нажмите  , чтобы выйти из режима администратора.

5. Разработка приложений

Доступные библиотеки/фреймворки:

- jQuery v2 (http://jquery.com/);
- Bootstrap v3 (http://getbootstrap.com);
- Font Awesome v4 (http://fontawesome.io).

Bootstrap поставляется без *Glyphicons*, используйте *Font Awesome*.

Базовая структура папок

- */data* – приложения и виджеты хранятся здесь, доступны по адресу *http://IP/apps/data/*
- */libs* – хранилище библиотек Lua, загружаются с помощью *require('custom.lib'),* где *lib* – название библиотеки.
- */user* – хранилище пользовательских файлов и скриптов, доступно по адресу *http://IP/user/*

Структура приложений/виджетов

Название приложения (ID) должно быть уникальным и состоять только из букв, цифр, знаков подчеркивания (_) и дефисов (-). Максимальная длина – 64 символа.

Структура папки

- *index.lp* или *index.html* – необходимо для приложений. Если url не задан, то нажатие на иконку приложения откроет папку приложения. Приложения должны иметь кнопку *Назад* для возвращения на главную страницу;
- *icon.svg* или *icon.png* – необходимо для приложений. Содержит иконку приложения, рекомендуемый формат – *SVG;*
- *widget.lp* или *widget.js* – необходимо для виджетов. Может содержать код (на *JavaScript + Lua* или на чистом *JavaScript)*, который отображает виджет;
- *title* – опционально для приложений. Текстовый файл с названием приложения, которое отображается под иконкой;
- *url* – опционально для приложений. Текстовый файл с URL-адресом, который открывается при нажатии на иконку приложения;
- *style.css* – опционально для виджетов. Содержит CSS-таблицу для виджета;
- *config.lp* или *config.html* – опциональный файл конфигурации, описание ниже.

В режиме виджетов идентификатор элемента изображения совпадает с именем виджета, для всех других идентификаторов элементов HTML должен быть задан уникальный идентификатор приложения, чтобы минимизировать конфликты между различными приложениями. Это же правило применяется к селекторам CSS.

Размер виджета по умолчанию – 100×100px. Высота и ширина могут быть изменены вызовом функции *setWidgetSize(cols, rows)* элемента виджета. Формула ширины: *cols * 110 – 10,* формула

высоты: *rows * 110 – 10.*

Пример

Виджет “Часы”, имеющий двойную ширину/высоту и отображающий SVG-изображение, которое заполняет все свободное место внутри контейнера виджета:

```
(function() {
        // получение элемента виджета и задание двойной ширины/высоты
        var el = $('#clock').setWidgetSize(2, 2);
        $('<object type="image/svg+xml"></object>') // объект принимает SVG+JavaScript
          .css('width', '100%') // максимальная ширина
          .css('height', '100%') // максимальная высота
          .attr('data', '/apps/data/clock/clock.svg') // адрес SVG изображения
          .appendTo(el); // добавление в контейнер
      })();
```

Конфигурация

- В папке приложения должен находиться либо *config.lp,* либо *config.html;*
- Этот файл обязан содержать элемент *form*, ID должен быть настроен в *myapp-config* формате, где *myapp* – уникальное название приложения;
- Обмен данными происходит через события, вызываемые через элемент *form*:
  - *config-load* (в приложение) – предоставляет объект со всеми конфигурационными парами ключ/значение;
  - *config-check* (в приложение) – срабатывает, когда нажимается кнопка “Сохранить”, приложение выдаст ошибку либо выполнит *config-save;*
  - *config-save* (из приложения) – сохраняет конфигурацию на стороне сервера и закрывает всплывающее окно, приложение должно передать параметры конфигурации в виде объекта;
- Доступ к конфигурации можно получить из Lua, используя следующие функции:
  - *config.get(app, key, default)* – возвращает значение для заданного приложения, значение по умолчанию или *nil*, если ключ не найден;
  - *config.getall(app)* – возвращает *таблицу (table)* со всеми значениями конфигурации для заданного приложения или *nil*, если конфигурация пуста;
  - *config.set(app, key, value)* – добавляет пару ключ/значение или перезаписывает существующую;
  - *config.setall(app, cfg)* – перезаписывает существующую конфигурацию с данной *таблицей конфигурации (cfg table)* с парами ключ/значение;
  - *config.delete(app, key)* – удаляет существующую пару ключ/значение;
- Неопубликованные приложения с конфигурационным файлом находятся на вкладке *Dev apps* на странице администратора.

Пример (config.html)

Создайте элемент *for* с числовым вводом значений от 0 до 100.

```
<form id="myapp-config">
  <div class="form-group">
    <label for="myapp-input">Numberic input</label>
    <input type="number" name="input" id="myapp-input" class="form-control" min="0" max="100">
  </div>
</form>

<script>
(function() {
  var el = $('#myapp-config') // элемент формы
    , input = $('#myapp-input'); // элемент ввода

  // задание значений элементов, когда конфигурация загружена
  el.on('config-load', function(event, data) {
    $.each(data, function(key, value) {
      $('#myapp-' + key).val(value);
    });
  });

  // запускается по нажатию кнопки Save
  el.on('config-check', function() {
    var val = parseInt(input.val(), 10) // введённое значение
      , min = parseInt(input.attr('min'), 10) // минимальное значение
      , max = parseInt(input.attr('max'), 10); // максимальное значение

    // неверное значение
    if (isNaN(val) || val < min || max < val) {
      alert('Please enter a value between ' + min + ' and ' + max);
    }
    // всё в порядке, сохраняем конфигурацию
    else {
      el.triggerHandler('config-save', { input: val });
    }
  });
})();
</script>
```

Функции localStorage

localStorage позволяет сохранять пользовательские настройки. Некоторые функции представлены для безопасного выполнения функций *localStorage*, так как они могут не выполниться в некоторых случаях, например, в приватном просмотре на iOS. Они позволяют хранить любые значения, которые могут быть переведены с помощью *JSON.stringify*.

- *storeSet(key, value)* – сохраняет пару ключ/значение;
- *storeGet(key)* – получает значение по ключу, возвращает *null*, если ключ не найден;
- *storeRemove(key)* – удаляет ключ из хранилища.

Ключи хранилища должны иметь префикс с уникальным именем приложения, чтобы минимизировать конфликты между различными приложениями.

Примеры

Получение выбранной темы (light/dark):

```
var theme = storeGet('theme') || 'light';
```

Сохранение объектов JavaScript:

```
var user = { name: 'John', surname: 'Doe', age: 42 };
storeSet('myapp_user', user);
```

Перевод

- *$.i18n.lang* – текущий язык или *undefined*, если выбран язык по умолчанию;
- *$.i18n.add(ns, dictionary)* – добавляет перевод в заданный словарь, *ns* – название приложения;
- *$.i18n.translate(key, default, vars)* or *$.tr(key, default, vars)* – переводит заданный ключ или использует значение по умолчанию, если перевод для выбранного языка не найден. Дополнительные переменные могут быть переданы для замены переменных внутри переводимого текста.

Пример 1

```
// описание перевода для приложения “myapp”
$.i18n.add('myapp', {
  // перевод для mylang
  mylang: {
    hello: 'Hello %{username}, current temperature is %{temperature}',
    goodbye: 'Goodbye %{username}'
  }
});

var text = $.tr('myapp.hello', 'No translation', { username: 'John', temperature: 21 });

// выдаст “Hello John, current temperature is 21”, если указан язык “mylang”
// в противном случае выдаст “No translation”
alert(text);
```

Пример 2

С помощью функции *tr* можно добавить перевод для селекторов *jQuery*: все HTML-элементы, которые имеют класс *tr* или атрибут *data-tr-key,* будут заменены на переведённый текст.

HTML:

```
<div id="myapp-container">
  <span class="tr" data-tr-key="myapp.hello">Hello!</span>
</div>
```

JavaScript:

```
// перевод на французский
$.i18n.add('myapp', {
  fr: {
    hello: 'Bonjour!'
  }
});

// применить перевод ко всем элементам внутри myapp-container
$('#myapp-container').tr();
```

LP-скрипты

В одном файле можно совмещать HTML и Lua, при этом код на Lua должен быть размещен в теге *<? ?>*. Закрывающий тег в конце документа не обязателен.

Пример

Вывод текущей даты:

```
<!DOCTYPE html>
<html>
<body>Current date is <? write(os.date()) ?></body>
</html>
```

Доступные функции:

- *header(hdr)* – добавляет пользовательский заголовок к выходу;
- *getvar(name)* – возвращает переменную *GET/POST* или *nil*, если переменная не задана;
- *getvars()* – возвращает все переменные *GET/POST* в виде Lua-таблицы;
- *getcookie(name)* – возвращает содержимое *cookie* или *nil,* если cookie отсутствует;
- *print(...)* – выводит любое количество переменных, вывод оканчивает переводом строки (*CRLF);*
- *write(...)* – похожа на *print,* но не переводит строку в конце;
- *escape(val)* – выводит одинарные/двойные кавычки, знаки больше/меньше в HTML.

Библиотеки подгружаются с помощью *require('apps')* и предоставляют доступ к следующим функциям:

- Все встроенные функции LM: *alert*, *log*, *grp*, *storage* и т. д.;
- Библиотека *config;*
- Функции *vprint(...)* и *vprinthex(...)* для вывода переменных в удобной для восприятия форме;
- Библиотека *json.*

Пример

Вывод таблицы умножения. Размер может быть задан через *GET/POST* переменную в диапазоне от 1 до 20 (10 – по умолчанию).

```
<!DOCTYPE html>
<html>
<body>
<?
size = getvar('size') -- GET/POST переменная
size = tonumber(size) or 0 -- конвертация в число
if size < 1 or 20 < size then
  size = 10 --  задание значения по умолчанию, если size пуст или некорректен
end
?>
<table border="1" cellpadding="3">
<? for i = 1, size do ?>
  <tr>
    <? for j = 1, size do ?>
```

```
      <td><? write(i * j) ?></td>
    <? end ?>
  </tr>
<? end ?>
</table>
</body>
</html>
```

Полное справочное руководство по функциям Lua доступно по адресу: http://openrb.com/docs/lua.htm

Функции для работы с объектами

Большинство функций используют параметр *alias* – групповой адрес объекта или имя объекта ('1/1/1' или 'My object').

Поиск одного/нескольких объектов

grp.find(alias)

Возвращает объект по заданному alias. Значение объекта может быть автоматически расшифровано, если задан тип данных.
Возвращает *nil*, если объект не найден, или таблицу со следующими значениями:

- *address* – групповой адрес объекта;
- *updatetime* – последнее время изменения в формате временной метки UNIX. Используйте os.date() для конвертации в удобную для восприятия форму;
- *name* – уникальное название объекта;
- *datatype* – тип данных объекта;
- *decoded* – *true*, если доступно расшифрованное значение объекта;
- *value* – расшифрованное значение объекта.

grp.tag(tags [, mode])

Возвращает таблицу со всеми объектами, имеющими заданные теги. Параметр tags может быть таблицей, либо строкой. Параметр mode может иметь значение ‘*or*’ (по умолчанию) и возвращать объекты, у которых есть любой из заданных тегов, – либо ‘*and*’ – возвращать объекты, у которых есть все заданные теги. Вы можете использовать функции для работы с объектами с полученной таблицей.

grp.dpt(dpt [, strict])

Находит все объекты с заданным типом данных. *Dpt* может быть строкой (*“bool”*, *“scale”*, *“uint32”* и т. д.) или полем из таблицы *dt* (*dt.bool, dt.scale, dt.uint32*). Например, если *dpt* задан *dt.uint8*, в обычном режиме все подтипы (*dt.scale*, *dt.angle* и др.) будут включены. Если необходим поиск по конкретному типу данных, задайте *strict* – *true*.

grp.all()

Возвращает таблицу со всеми известными объектами.

grp.alias(alias)

Возвращает имя объекта по его адресу или адрес по имени (или *nil*, если объект не найден).

grp.getvalue(alias)

Возвращает значение объекта по *alias* или *nil,* если объект не найден.

Работа с шиной

grp.write(alias, value [, datatype])

Посылает объекту запрос на запись на заданный объект по *alias*. Тип данных берётся из базы данных, если не задан в функции. Возвращает результат в виде *boolean*.

grp.response(alias, value [, datatype])

Похож на *grp.write*. Посылает объекту на заданный *alias* запрос на *response*. Используется для вывода ответа на *read*-запрос из шины.

grp.read(alias)

Посылает объекту на заданный *alias* запрос на чтение. Примечание: эта функция не возвращает полученное значение. Используйте событийный скрипт.

grp.update (alias, value [, datatype])

Похож на grp.write, но не посылает значение в шину. Полезно для виртуальных объектов, используемых на визуализации.

Управление тегами

grp.gettags(alias)

Возвращает таблицу со всеми тегами объекта.

grp.addtags(alias, tags)

Добавляет теги к объекту. Параметр *tags* может быть строкой (одиночный тег) или Lua-таблицей, содержащей строки (несколько тегов).

grp.removetags(alias, tags)

Удаляет теги объекта. Параметр *tags* может быть строкой (одиночный тег) или Lua-таблицей, содержащей строки (несколько тегов).

grp.removealltags(alias)

Удаляет все теги объекта.

grp.settags(alias, tags)

Перезаписывает теги объекта. Параметр *tags* может быть строкой (одиночный тег) или Lua-таблицей, содержащей строки (несколько тегов).

Создание и изменение объектов

grp.setcomment(alias, comment)

Задаёт поле *comment* объекта.

grp.create(config)

Создаёт новый или перезаписывает существующий объект с заданной конфигурацией, которая должна быть Lua-таблицей. Возвращает либо ID объекта (в случае успеха), либо *nil* и сообщение ошибки.

Поля *config*:

- *datatype* – обязательный, тип данных объекта. Может быть строкой (*“bool”, “scale”, “uint32”* и др.) или полем из таблицы *dt* (*dt.bool, dt.scale, dt.uint32*);
- *name* – необязательный, уникальное имя объекта. Если объект с таким именем уже существует, добавляется цифра-префикс;
- *comment* – необязательный, комментарий к объекту (*string*);
- *units* – необязательный, единицы измерения объекта (*string*);
- *address* – необязательный, групповой адрес объекта (*string*). Если не задан, будет использован первый свободный адрес из настроенного диапазона;
- *tags* – необязательный, теги объекта. Может быть строкой (один тег) или таблицей Lua (несколько тегов).

Если объект с заданным групповым адресом уже существует, то будут изменены только поля *units, datatype* и *comment*. Все остальные свойства останутся без изменений.

Примеры

Создание объекта с заданным адресом:

```
address = grp.create({
  datatype = dt.float16,
  address = '1/1/1',
  name = 'My first object',
  comment = 'This is my new object',
  units = 'W',
  tags = { 'My tag A', 'My tag B' },
})
```

Создание объекта с автоматическим присвоением адреса:

```
address = grp.create({
  datatype = dt.bool,
  name = 'My second object',
})
```

Функции работы с базой данных

В качестве движка базы данных используется SQLite v3.

Примечание: Таблицы базы данных должны иметь префикс с уникальным именем приложения, чтобы минимизировать конфликты между различными приложениями.

Основные функции

- *db:execute(query)* – выполняет заданный запрос, возвращает результат запроса либо курсор базы данных;
- *db:escape(value)* – защищает заданную строку для её последующего использования в запросах;
- *db:query(query, …)* – выполняет заданный запрос, вопросительные знаки в запросе

заменяются на дополнительные параметры (смотрите примеры ниже).

Запросы INSERT/UPDATE/DELETE

Примечание: Lua-таблицы, переданные в качестве параметров запроса, не должны иметь полей, которых нет в базе данных. В противном случае запрос не выполнится.

- *db:insert(tablename, values)* – выполняет запрос *INSERT*, основанный на заданном параметре *values;*
- *db:update(tablename, values, where)* – выполняет запрос *UPDATE*, основанный на заданных параметрах *values* и *where;*
- *db:delete(tablename, where)* – выполняет запрос *DELETE*, основанный на заданном параметре *where.*

Запросы SELECT

Примечание: параметры должны быть переданы в том же виде, что и в функции *db:query()*.

- *db:getone(query, …)* – возвращает первое значение из первого подходящего ряда по заданному запросу;
- *db:getrow(query, …)* – возвращает первый подходящий ряд по заданному запросу;
- *db:getlist(query, …)* – возвращает результат запроса в виде Lua-таблицы, в которой каждое значение – первое поле из каждого ряда;
- *db:getall(query, …)* – возвращает результат запроса в виде Lua-таблицы, в которой каждое значение – Lua-таблица с соответствием field→value.

Примеры

```
-- Замена параметров
db:query('UPDATE table SET field=? WHERE id=?', 'test', 42)
-- То же, что и INSERT INTO table (id, value) VALUES (42, 'test')
db:insert('table', {
  id = 42,
  value = 'test',
})
-- То же, что и UPDATE table SET value='test' WHERE id=42
db:update('table', { value = 'test' }, { id = 42 })
-- То же, что и DELETE FROM table WHERE id=42
db:delete('table', { id = 42 })
```

6. Настройка LogicMachine

Login	**Password**
admin	admin

Это главная страница менеджера настройки LogicMachine. Главное меню содержит следующие вкладки:

- ***Scripting*** – управление скриптами;
- ***Objects*** – управление объектами KNX;
- ***Object logs*** – логи объектов KNX;
- ***Schedulers*** – интерфейс администратора для пользовательских планировщиков;
- ***Trend logs*** – интерфейс администратора для сбора статистики;
- ***Scenes*** – визуальный редактор сцен;
- ***Vis.structure*** – управление структурой визуализации;
- ***Visualization*** – создание визуализации и управление ею;
- ***Vis.graphics*** – управление иконками, фоновыми изображениями и шрифтами;
- ***Utilities*** – утилиты, в том числе импорт из ETS, сброс базы данных объектов, резервное копирование, установка обновлений системы;
- ***User access*** – управление уровнями доступа пользователей;
- ***BACnet*** – клиент BACnet со сканером;
- ***DALI*** – управление DALI;
- ***1-Wire*** – управление 1-wire;
- ***Modbus*** – управление Modbus;
- ***BLE*** – настройка Bluetooth-устройств (через USB-шлюз);
- ***Alerts*** – сообщения о тревогах, записанные функцией alert;
- ***Logs*** – логи, записанные функцией log;
- ***Error log*** – сообщения об ошибках системы или в скриптах;
- ***About*** – информация о производителе.

6.1. Вкладка Scripting

Вкладка *Scripting* позволяет добавлять и изменять скрипты различных типов. Логика программируется на языке Lua. Большинство аспектов языка Lua описано в книге "Programming in Lua", которая бесплатно доступна здесь: http://lua.org/pil/

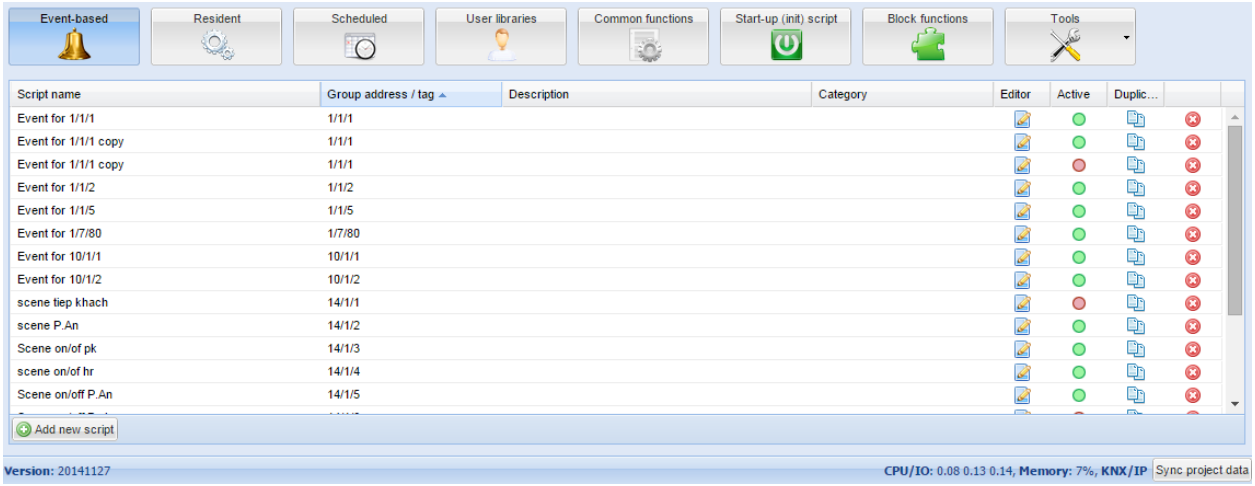
Примечание! Здесь доступно справочное руководство по Lua для LogicMachine: http://openrb.com/docs/lua.htm

Существует 6 основных типов скриптов:

- *Событийные* – скрипты, которые запускаются при появлении события на шине. Обычно используются при необходимости реагирования в реальном времени.
- *Резидентные* – скрипты, которые используют опрос для проверки изменения состояния объекта. Обычно используются с системами отопления и вентиляции, когда данные получаются более чем с одного устройства.
- *По расписанию* – скрипты, которые запускаются по расписанию. Могут использоваться для различных охранных систем и моделирования присутствия.
- *Пользовательские библиотеки* – пользовательские скрипты для вызова в других скриптах.
- *Общие функции* – распространённые функции для вызова их в других скриптах.
- *Скрипт запуска системы* – скрипт, который запускается при запуске системы.

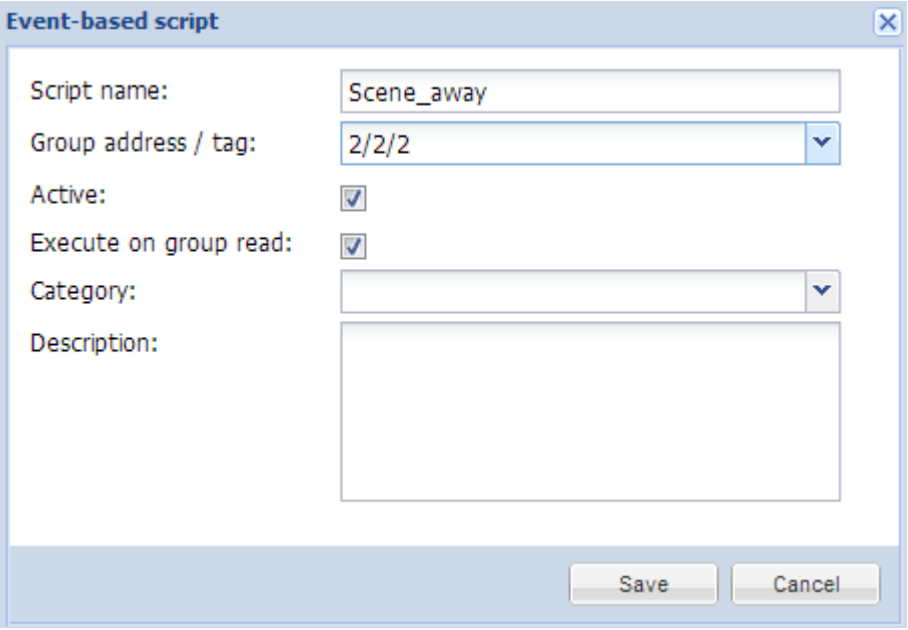
6.1.1. Добавление нового скрипта

Нажмите кнопку *Add new script* в нижней части подменю *Event-based, Resident* или *Scheduled.*



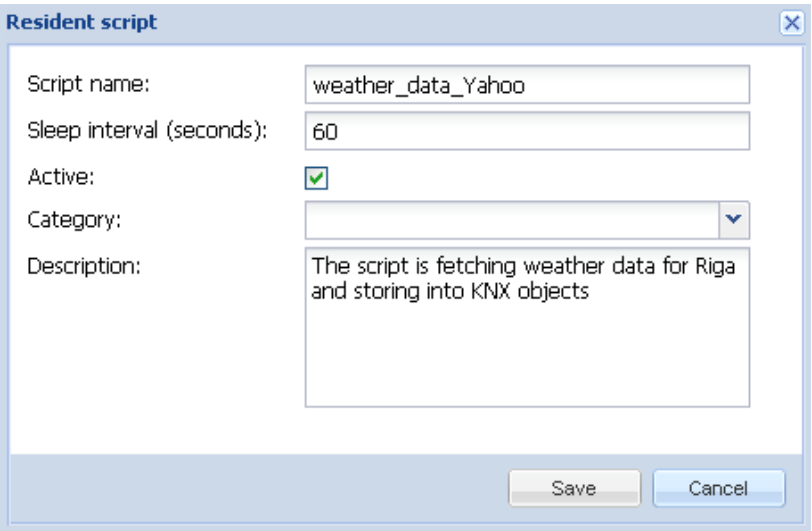
При добавлении нового скрипта нужно заполнить следующие поля:

Событийный скрипт



- ***Script name*** – название скрипта;
- ***Group address / Tag*** – групповой адрес или тег, по которому скрипт будет срабатывать;
- ***Active*** – определяет, включен (зелёный кружок) или выключен (красный кружок) скрипт;
- ***Execute on group read*** – определяет, будет ли срабатывать скрипт при KNX-телеграмме чтения;
- ***Category*** – категория скрипта. Это не влияет на работу скрипта, но помогает группировать скрипты по типу;
- ***Description*** – описание скрипта.

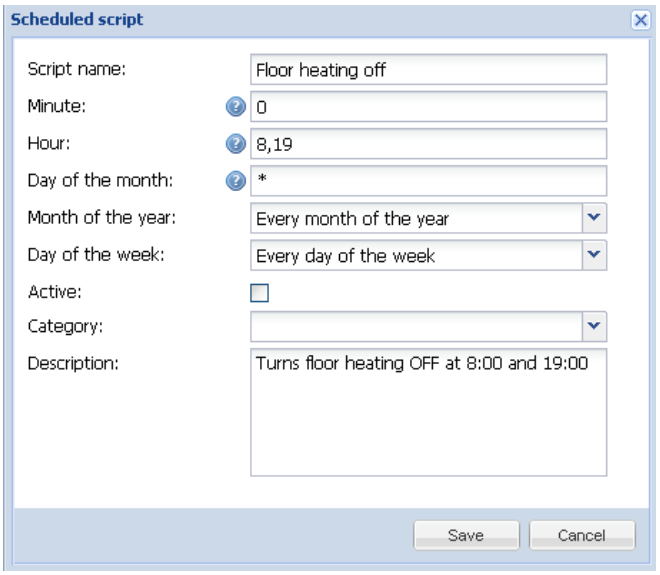
<u>Резидентный скрипт</u>



- ***Script name*** – название скрипта;
- ***Sleep interval (seconds)*** – время между окончанием предыдущего запуска скрипта и запуском нового;

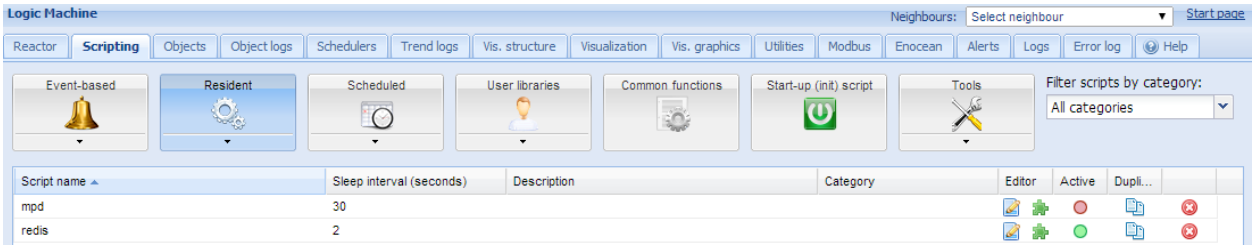
- ***Active*** – определяет, включен (зелёный кружок) или выключен (красный кружок) скрипт;
- ***Category*** – категория скрипта. Это не влияет на работу скрипта, но помогает группировать скрипты по типу;
- ***Description*** – описание скрипта.

Скрипт по расписанию



- ***Script name*** – название скрипта;
- ***Minute*** – минуты запуска скрипта;
- ***Hour*** – часы запуска скрипта;
- ***Day of the month*** – дни месяца запуска скрипта;
- ***Month of the year*** – месяцы года запуска скрипта;
- ***Day of the week*** – дни недели запуска скрипта;
- ***Active*** – определяет, включен (зелёный кружок) или выключен (красный кружок) скрипт;
- ***Category*** – категория скрипта. Это не влияет на работу скрипта, но помогает группировать скрипты по типу;
- ***Description*** – описание скрипта.

Список скриптов



Действия, которые вы можете совершить с каждым скриптом:

Duplicate – дублировать скрипт с его кодом;

Editor – войти в редактор скрипта, чтобы написать код для необходимой работы;

Active – включить (зелёный) или выключить (красный) скрипт;

Delete – удалить скрипт. При нажатии появится окно, в котором вы должны подтвердить своё действие.

6.1.2. Событийные скрипты

Событийные скрипты используются при создании дополнительной логики для группового адреса или тега. Пользовательские функции выполняются при телеграмме “записи” или “чтения” (если включено) на заданном групповом объекте. Информация о событии хранится в глобальной переменной *event*. Переменная содержит:

- *dstraw (integer)* – необработанный групповой адрес назначения;
- *srcraw (integer)* – необработанный физический адрес источника;
- *dst (string)* – расшифрованный групповой адрес назначения (1/1/4);
- *src (string)* – расшифрованный физический адрес источника (1.1.2);
- *type (string)* – тип события, может быть "groupwrite"(запись), "groupread"(чтение), "groupresponse"(ответ). На данный момент пользовательские скрипты привязаны только к событиям “запись”;
- *dataraw (integer/string)* – необработанные двоичные данные;
- *datahex (string)* – данные в шестнадцатеричном виде, которые могут быть преобразованы в переменные Lua.

*Примечание! Переменная **event** доступна только в событийных скриптах.*

Примечание! Все событийные скрипты выполняются в режиме единой очереди. Убедитесь, что ваши скрипты не содержат бесконечных циклов, вызовов sleep или других блокирующих частей.

Примечание! Для получения значения события используйте следующую команду: **a = event.getvalue()**

Примечание! Для получения имени группового адреса из события используйте следующую команду: **a = grp.alias(event.dst)**

6.1.3. Резидентные скрипты

Резидентные скрипты выполняются бесконечное количество раз. Скрипты выключаются на некоторый интервал после выполнения и по истечении таймера включаются вновь.

Примечание! Независимо от того, запускаются ли резидентные скрипты параллельно, они не должны содержать блокирующих частей, иначе вы не сможете перезагрузить скрипт после изменения.

6.1.4. Скрипты по расписанию

Скрипты по расписанию запускаются при соответствии системного времени и времени, заданного в настройках скрипта. Скрипты запускаются только один раз при совпадении времени.

Scheduled scripting date/time format

Scheduled scripting uses standard cron format for date/time parameters. Valid values are:

***** — execute script every minute, hour or day.

***/N** — execute script every N minutes, hours or days. N is an integer, script is executed when current value divided by N gives 0 in modulo. For example, script with hour parameter set to */8 will be executed when hour is 0, 8 and 16.

N — execute script exactly at N minute, hour or day.

N-K — execute script when minute, hour or day is between N-K range (inclusive).

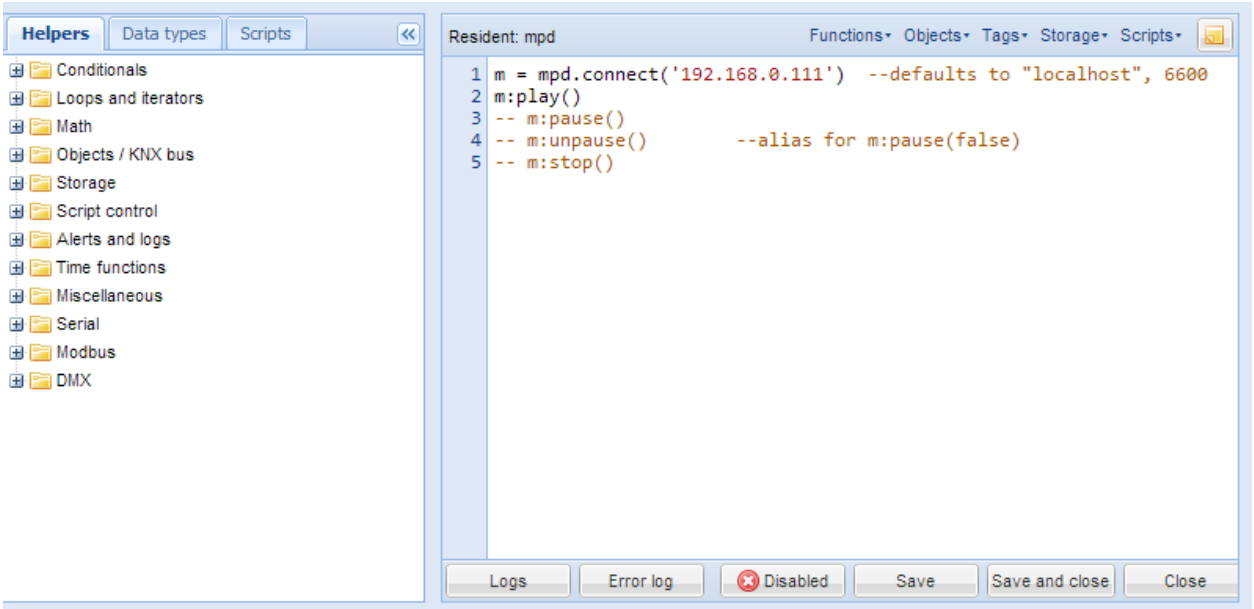
N,K — it is possible to specify several N and N-K type parameters separated by comma. For example, script with minute parameter set to 15,50-52 will get executed when minute is 15, 50, 51 and 52

Для задания параметров даты/времени скрипт по расписанию использует формат cron. Допустимые значения:

- *** – запускает скрипт каждую минуту, час или день;
- **/N* – запускает скрипт каждые N минут, часов или дней. N – целое число, скрипт запускается, когда текущее значение делится на N без остатка. Например, скрипт с параметром “Часы”, заданным как */8, станет запускаться, когда будет 0, 8 и 16 часов;
- *N* – запускает скрипт, когда минуты, часы или дни равны N;
- *N-K* – запускает скрипт, когда минуты, часы или дни находятся в диапазоне N-K (включительно);
- *N,K* – можно задать несколько параметров типа N и N-K через запятую. Например, если параметр “Минуты” задан “15,50-52”, скрипт станет запускаться, когда будет 15, 50, 51 и 52 минуты.

6.1.5. Редактор скриптов

После добавления скрипта в колонке *Editor* появляется иконка, которая позволяет открыть скрипт в редакторе скриптов и изменить его с помощью встроенных фрагментов кода.



Редактор устроен так, чтобы, даже не зная синтаксиса, вы могли бы создавать скрипты за счёт множества готовых конструкций, которые автоматически вставляются в код. Эти конструкции также экономят время и уменьшают количество опечаток, делая программирование более удобным. После нажатия на нужный фрагмент код автоматически добавляется в поле редактора.

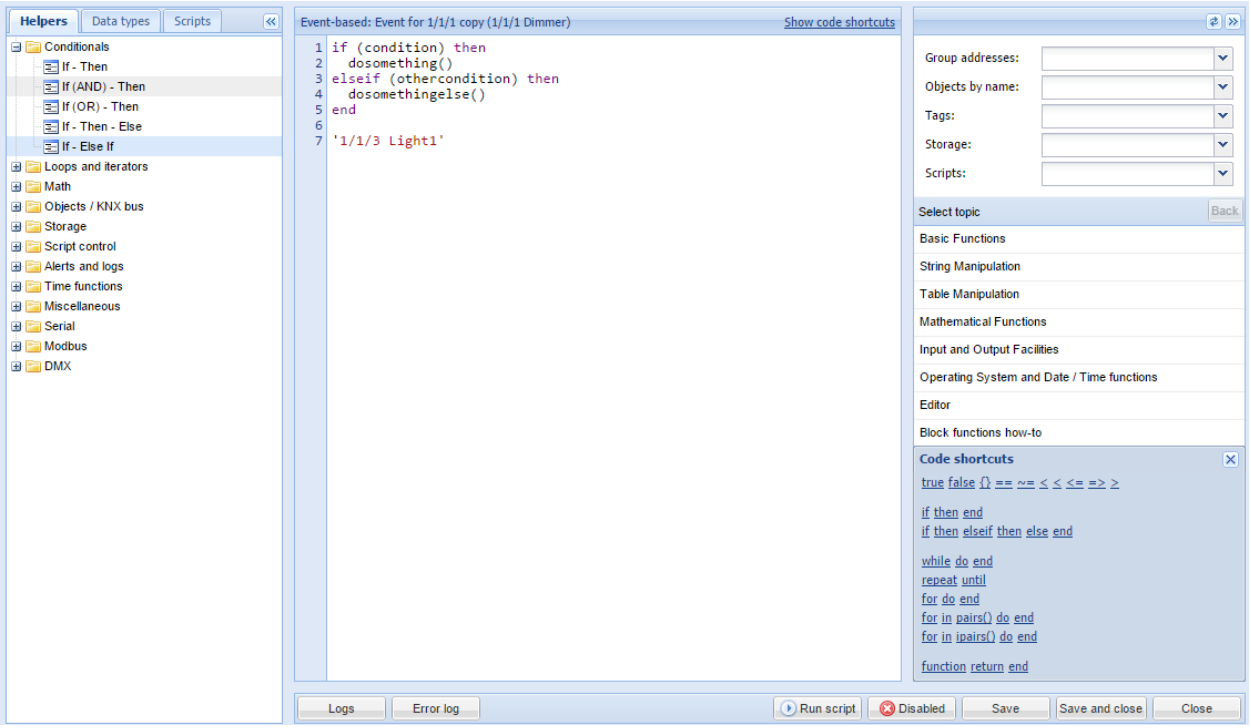
<u>Существуют три основных окна редактора скриптов:</u>

Помощники – фрагменты кода, например, оператор if-then. Помощники содержат несколько подгрупп:

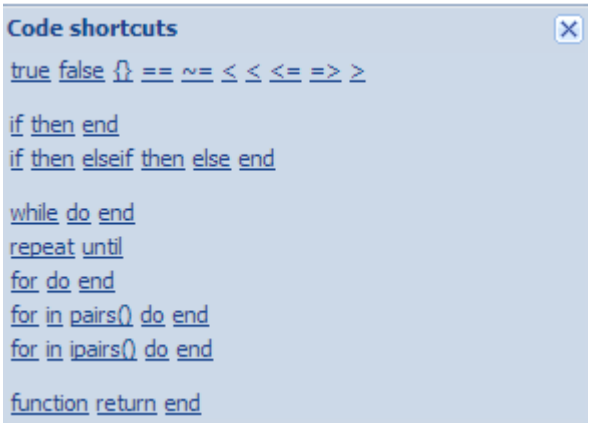
- *Conditionals* – операторы условий;
- *Loops and iterators* – циклы и итераторы;
- *Math* – математические функции;
- *Objects/KNX bus* – работа с объектами и шиной KNX;
- *Storage* – работа с хранилищем;
- *Script control* – управление скриптами;
- *Alerts and logs* – функции оповещения;
- *Time functions* – задержка в выполнении скрипта;
- *Miscellaneous* – дополнительные функции (отправка e-mail, время восхода/захода);
- *Serial* – работа со встроенными портами LogicMachine;
- *Modbus* – работа с Modbus;
- *DMX* – работа с DMX-устройствами;
- *Data types* – типы данных;
- *Scripts* – список всех скриптов в LogicMachine.

Помощники в правой части редактора

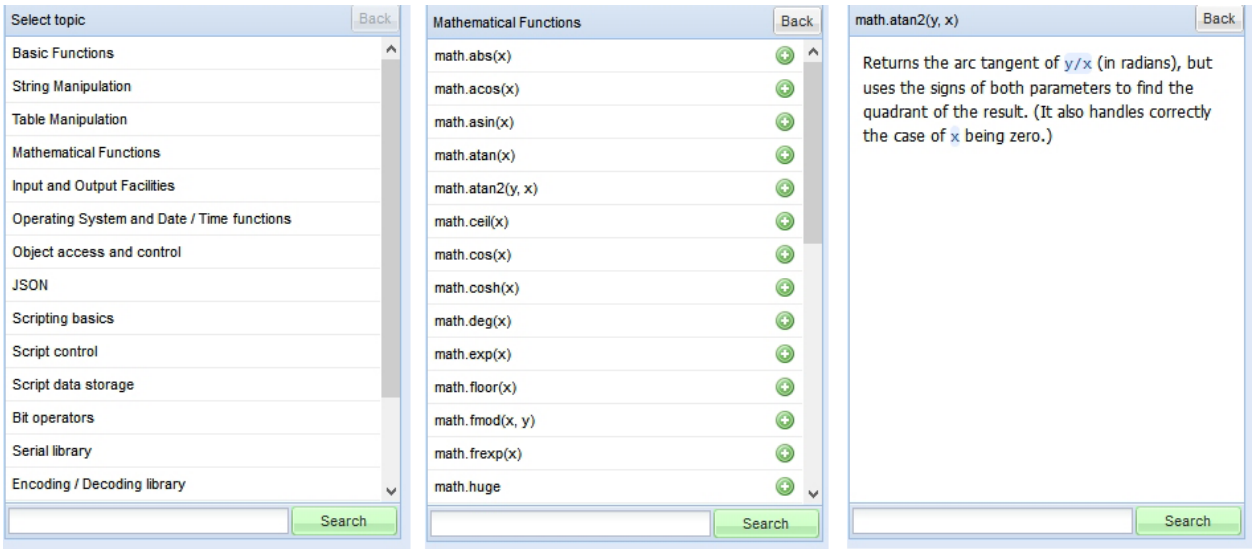
В верхнем углу правой части окна расположен специальный раздел, который помогает быстро найти в хранилище функции, объекты, теги или переменные.



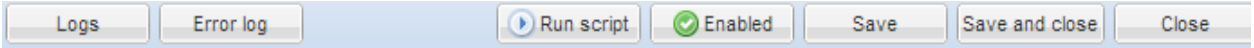
Внизу располагаются кнопки быстрого доступа к коду, которые помогают понять общую структуру функций.



Помимо этого, в правой части окна структурированы различные функции. По нажатию на тип функций, к которым нужно получить доступ (например, *Mathematical functions* для доступа к математическим функциям), открывается список функций нужного типа с описанием синтаксиса. Нажатие на имя функции открывает краткую справку по ней, а нажатие кнопки рядом с функцией добавляет её в код.



Также в окне располагаются полезные кнопки для быстрого доступа к окнам *Error log* и *Logs*, запуска скрипта или его включения/выключения.



6.1.6. Функции работы с объектами

grp предоставляет доступ к объектам, хранящимся в базе данных, и групповым адресам. Большинство функций использует параметр *alias* – групповой адрес объекта или его уникальное название. (*'1/1/1'* или *'My object'*).

grp.getvalue(alias)

Возвращает значение объекта по alias или nil, если объект не найден.

grp.find(alias)

Возвращает объект по заданному *alias*. Значение объекта может быть автоматически расшифровано, если у объекта задан тип данных.

Возвращает *nil*, если объект не найден, или таблицу со следующими полями:

- *address* – групповой адрес объекта;
- *updatetime* – последнее время изменения в формате временной метки UNIX. Используйте *os.date()* для конвертации в удобную форму восприятия;
- *name* – уникальное название объекта;
- *datatype* – тип данных объекта;
- *decoded* – true, если доступно расшифрованное значение объекта;
- *value* – расшифрованное значение объекта.

grp.tag(tags [, mode])

Возвращает таблицу со всеми объектами, имеющими заданные теги. Параметр tags может быть таблицей либо строкой. Параметр mode может иметь значение ‘or’ (по умолчанию) и возвращать объекты, у которых есть любой из заданных тегов, – либо ‘and’ – возвращать

объекты, у которых есть все заданные теги. Вы можете использовать функции для работы с объектами с полученной таблицей.

grp.alias(alias)

Возвращает имя объекта по его адресу или адрес по имени (или *nil*, если объект не найден).

6.1.7. Функции объектов

Объекты, полученные через *grp.find(alias)* или *grp.tag(tags [, mode])*, имеют следующие функции:

Примечание! Всегда проверяйте, что вы получили объект, иначе использование следующих функций вызовет ошибку. Смотрите примеры ниже.

object:write(value, datatype)

Посылает телеграмму “запись” на групповой адрес объекта. Тип данных берётся из базы данных, если не задан в качестве параметра. Возвращает результат в виде boolean.

object:response(value, datatype)

Похожа на object:write. Посылает телеграмму “ответ” на групповой адрес объекта.

object:read()

Посылает телеграмму “чтение” на групповой адрес объекта. Примечание: эта функция не возвращает значение объекта. Используйте событийные скрипты.

object:update(value, datatype)

Похожа на object:write, но не посылает значение в шину. Полезно для обновления значения виртуальных объектов, используемых на визуализации.

6.1.8. Функции коммуникации

Эти функции должны использоваться только при необходимости доступа к объекту по групповому адресу напрямую. Рекомендуем пользоваться функциями объектов.

grp.write(alias, value [, datatype])

Посылает объекту запрос на запись на заданный объект по *alias*. Тип данных берётся из базы данных, если не задан в функции. Возвращает результат в виде *boolean*.

grp.response(alias, value [, datatype])

Похож на *grp.write*. Посылает объекту на заданный *alias* запрос на *response*. Используется для вывода ответа на *read*-запрос из шины.

grp.read(alias)

Посылает объекту на заданный *alias* запрос на чтение. Примечание: эта функция не возвращает

полученное значение. Используйте событийный скрипт.

grp.update (alias, value [, datatype])

Похож на grp.write, но не посылает значение в шину. Полезно для виртуальных объектов, используемых на визуализации.

6.1.9. Примеры работы с объектами

Поиск объекта по имени и запись в него нового значения:

```
1. myobject=grp.find('My object')
2. -- grp.find вернёт nil, если объект не будет найден
3. if myobject then
4. myobject:write(1)-- задаёт объекту значение 1
5. end
```

Поиск объекта по адресу и запись в него нового значения:

```
1. myobject=grp.find('1/1/15')
2. -- проверка, точно ли объект найден
3. if myobject then
4. myobject:write(52.12, dt.float16)-- явно задаёт тип данных dt.float16 (2-byte с плавающей
   точкой)
5. end
```

Выключение всех объектов с тегом lights:

```
1. lights = grp.tag('lights')
2. lights:write(false)
```

Запись на определённый групповой адрес с заданным типом данных:

```
1. grp.write('1/1/1', true, dt.bool)-- запись 1-bit 'on' на 1/1/1
2. grp.write('1/1/2', 50, dt.scale)-- запись 1-byte 50% на 1/1/2
```

6.1.10. Функции работы с типами данных

Объект **knxdatatype** предоставляет доступ к конвертированию данных между Lua и KNX.

knxdatatype.decode(value, datatype)

Преобразует шестнадцатеричные данные в переменную Lua с заданным типом. Тип задаётся либо в виде основного типа данных KNX (число от 1 до 16), либо в виде дополнительного типа данных (число от 1000 до 16000). Возвращаемые значения:

- в случае успеха – расшифрованные данные в виде переменной Lua, длина значения в байтах;
- в случае ошибки – *nil*, строка с ошибкой.

6.1.11. Типы

данных

Следующие типы данных могут использоваться для кодирования и декодирования данных KNX. Соответствие с данными Lua и предопределёнными константами (жирным шрифтом) приведено ниже:

- *1 bit (boolean)* – ***dt.bool*** – boolean;
- *2 bit (1 bit controlled)* – ***dt.bit2*** – число;
- *4 bit (3 bit controlled)* – ***dt.bit4*** – число;
- *1 byte ASCII character* – ***dt.char*** – строка;
- *1 byte unsigned integer* – ***dt.uint8*** – число;
- *1 byte signed integer* – ***dt.int8*** – число;
- *2 byte unsigned integer* – ***dt.uint16*** – число;
- *2 byte signed integer* – ***dt.int16*** – число;
- *2 byte floating point* – ***dt.float16*** – число;
- *3 byte time / day* – ***dt.time*** – таблица со следующими полями:
    - day – число (0-7);
    - hour – число (0-23);
    - minute – число (0-59);
    - second – число (0-59);
- *3 byte date* – ***dt.date*** – таблица со следующими полями:
    - day – число (1-31);
    - month – число (1-12);
    - year – число (1990-2089);
- *4 byte unsigned integer* – ***dt.uint32*** – число;
- *4 byte signed integer* – ***dt.int32*** – число;
- *4 byte floating point* – ***dt.float32*** – число;
- *4 byte access control* – ***dt.access*** – число, на данный момент поддерживается не полностью;
- *14 byte ASCII string* – ***dt.string*** – строка, нулевые символы ('*0*') отбрасываются при расшифровке.

6.1.12. Функции хранилища

Объект **storage** предоставляет хранилище данных типа ключ/значение для пользовательских скриптов. Поддерживаются только следующие типы данных Lua:

- boolean;
- number (число);
- string (строка);
- table (таблица).

storage.set(key, value)

Задаёт новое значение по заданному ключу. Старое значение перезаписывается. Возвращает boolean в качестве результата и возможную строку ошибки.

storage.get(key, default)

Получает значение по ключу или возвращает значение *default* (*nil*, если не задано), если ключ не найден в хранилище.

Примечание: все скрипты используют общее хранилище. Убедитесь, что по одному ключу вы не храните данные разных типов.

Примеры

Следующие примеры демонстрируют работу *storage.set*.

- Этот пример вернёт в качестве результата *true*, поскольку переданные значения были верны;

```
result=storage.set('my_stored_value_1', 12.21)
```

- Этот пример вернёт *false*, потому что мы пытались сохранить функцию, а это невозможно;

```
1. testfn=function(t)
2. return t * t
3. end
4. result =storage.set('my_stored_value_2', testfn)-- вернёт ошибку
```

Следующие примеры покажут работу *storage.get*.

- Если ключ не будет найден, первый вызов вернет *nil*, а второй – число “0”, которое было задано в *default*.

```
1. result =storage.get('my_stored_value_3')-- вернёт nil, если value не будет найден
2. result =storage.get('my_stored_value_3', 0)-- вернёт 0, если value не будет найден
```

- При сохранении таблицы убедитесь в возвращённом результате. Для примера сохраним в хранилище *test_object_data.*

```
1. objectdata={}
2. objectdata.temperature=23.1
3. objectdata.scene='default'
4. result =storage.set('test_object_data', objectdata)-- сохраним переменную objectdata как
   'test_object_data'
```

- Теперь получим данные из хранилища. Тип данных проверен для корректности.

```
1. objectdata=storage.get('test_object_data')
2. if type(objectdata)=='table' then
3. if objectdata.temperature> 24 then
4. -- сделаем что-нибудь, если температура слишком высокая
5. end
6. end
```

6.1.13. Функции оповещения (уведомление)

alert(message, [var1, [var2, [var3]]])

Сохраняет сообщения о тревоге и текущее системное время в базе данных. Все сообщения доступны во вкладке *Alerts*. Эта функция работает так же, как Lua string.format.

Пример

```
1. temperature = 25.3
2. if temperature > 24 then
3. -- итоговое сообщение: 'Temperature levels are too high: 25.3'
4.   alert('Temperature level is too high: %.1f', temperature)
5. end
```

6.1.14. Функция сохранения логов

log(var1, [var2, [var3, ...]])

Конвертирует переменные в читаемый формат и сохраняет их в базе данных. Все логи доступны на вкладке *Logs*.

Пример

```
1. -- функция log принимает переменные Lua типов nil, boolean, number и table (до 5 уровней
   вложенности)
2. a ={ key1 ='value1', key2 =2}
3. b ='test'
4. c =123.45
5. -- логи всех перечисленных переменных
6. log(a, b, c)
```

6.1.15. Формат даты/времени в скриптах по расписанию

Для задания параметров даты/времени скрипт по расписанию использует формат cron. Допустимые значения:

- *** – запускает скрипт каждую минуту, час или день
- **/N* – запускает скрипт каждые N минут, часов или дней. N – целое число, скрипт запускается, когда текущее значение делится на N без остатка. Например, скрипт с параметром “Часы”, заданным как */8, станет запускаться, когда будет 0, 8 и 16 часов.
- *N* – запускает скрипт, когда минуты, часы или дни равны N;
- *N-K* – запускает скрипт, когда минуты, часы или дни находятся в диапазоне N-K (включительно);
- *N,K* – можно задать несколько параметров типа N и N-K через запятую. Например, если параметр “Минуты” задан “15,50-52”, скрипт станет запускаться, когда будет 15, 50, 51 и 52 минуты.

6.1.16. Функции времени

os.sleep(delay)

Задерживает скрипт на *delay* секунд.

os.microtime ()

Возвращает два значения: текущую временную метку в секундах и временную метку в наносекундах.

os.udifftime (sec, usec)

Возвращает разницу между текущим временем и временной меткой, переданной в параметрах функции (секунды, наносекунды), в формате с плавающей точкой.

6.1.17. Сериализация данных

serialize.encode (value)

Генерирует сохраняемое представление значения.

serialize.decode (value)

Генерирует переменную Lua из сохранённого представления.

6.1.18. Функции работы со строками

Эта библиотека предоставляет общие функции для работы со строками: например, поиск и извлечение подстрок или совпадение по шаблону. При индексации строк в Lua первый символ будет идти под номером 1 (не 0, как в C).
Индексы могут быть отрицательными и интерпретируются как индексация с конца строки. Т. е. последний символ будет идти под номером -1 и так далее.
Библиотека string предоставляет все функции внутри таблицы string. Она также задаёт мета-таблицу, в которой поле __index задаёт ссылку на таблицу string. Следовательно, функции string возможно использовать в объектно-ориентированном стиле. Например, *string.byte(s, i)* можно написать как *s:byte(i)*. Библиотека string предполагает однобайтные кодировки символов.

string.trim (str)
Обрезает начальные и конечные пробелы заданной строки.

string.split (str, sep)
Разбивает строку по заданной строке-разделителю. Возвращает таблицу Lua.

string.byte (s [, i [, j]])
Возвращает внутренние числовые коды символов s[i], s[i + 1], ···, s[j]. Значение i по умолчанию равно 1, значение j по умолчанию равно i. Следует отметить, что числовые коды не обязательно переносимы между платформами.

string.char (···)
Получает ноль или более целых чисел. Возвращает строку с длиной, равной числу аргументов, где каждый символ имеет внутренний числовой код соответствующего аргумента. Обратите внимание, что числовые коды не обязательно переносимы между платформами.

string.find (s, pattern [, init [, plain]])
Ищет первое совпадение шаблона *pattern* в строке *s*. Если совпадение находится, то find возвращает индексы *s*, в которых начинается и заканчивается совпадение; в противном случае возвращает *nil*. Третий необязательный числовой аргумент *init* указывает, с какого индекса начать поиск; его значение по умолчанию равно 1 и может быть отрицательным. Значение *true* в качестве четвёртого необязательного аргумента *plain* отключает средства сопоставления с образцом, поэтому функция выполняет обычную операцию «поиск подстроки», при этом никакие символы в шаблоне не считаются «волшебными». Обратите внимание, что если задан *plain*, то также должен быть задан *init*. Если шаблон имеет захваты, то при успешном совпадении захваченные значения также возвращаются после двух индексов.

string.format (formatstring, ···)
Возвращает отформатированную версию переданных аргументов; формат указан в его первом аргументе и должен быть строкой. Строка формата следует тем же правилам, что и семейство *printf* стандартных функций C. Единственное отличие заключается в том, что опции/модификаторы *, l, L, n, p и h не поддерживаются и присутствует дополнительная опция q. Параметр q форматирует строку в форме, удобной для безопасного считывания интерпретатором Lua: строка записывается между двойными кавычками, и все двойные кавычки, новые строки, вложенные нули и обратные косые строки в строке правильно экранируются при написании. Например, вызов

```
string.format('%q', 'a string with "quotes" and \n new line')
```

выведет строку:

*"a string with \"quotes\" and *
new line"

Опции *c, d, E, e, f, g, G, i, o, u, X,* и *x* ожидают как аргумент число, а *q* и *s* ожидают строку. Эта функция не принимает строковые значения, содержащие внедрённые нули, исключение – аргументы опции q.

string.gmatch (s, pattern)
Возвращает функцию итератора, которая при каждом вызове возвращает следующие фрагменты строки, соответствующие шаблону. Если в шаблоне нет захватов, то при каждом вызове создаётся полное совпадение. В качестве примера следующий цикл:

```
1.  s = "hello world from Lua"
2.  for w in string.gmatch(s, "%a+") do
```

```
3.  log(w)
4.  end
```

будет перебирать все слова из строки s, печатая по одному в строке. Следующий пример собирает все пары key = value из заданной строки в таблицу:

```
1.  t ={}
2.  s ="from=world, to=Lua"
3.  for k, v in string.gmatch(s, "(%w+)=(%w+)") do
4.     t[k]= v
5.  end
```

Для этой функции символ '^' в начале шаблона не работает как якорь, так как это предотвратит итерацию.

string.gsub (s, pattern, repl [, n])

Возвращает копию s, в которой все (или первые n, если заданы) вхождения шаблона были заменены на строку, указанную в repl, которая может быть строкой, таблицей или функцией. Gsub также возвращает в качестве второго значения общее количество совпадений, которые произошли.

Если repl является строкой, ее значение используется для замены. Символ % работает как escape-символ: любая последовательность в repl формы %n, где n находится между 1 и 9, обозначает значение n-й строки захвата gsub (см. ниже). Последовательность %0 означает целое совпадение. Последовательность %% означает один %.

Если repl – таблица, то таблица запрашивается для каждого совпадения, используя первый захват в качестве ключа; если в шаблоне нет захватов, то в качестве ключа используется целое соответствие.

Если repl – функция, то эта функция вызывается каждый раз, когда происходит совпадение, со всеми захваченными подстроками, переданными в качестве аргументов, по порядку; если в шаблоне нет захватов, то всё совпадение передаётся в качестве единственного аргумента.

Если значение, возвращаемое запросом таблицы или вызовом функции, является строкой или числом, то оно используется в качестве строки замены; в противном случае, если это false или nil, то замены нет (то есть исходное совпадение сохраняется в строке).

Примеры

```
x =string.gsub("hello world", "(%w+)", "%1 %1")
     --> x="hello hello world world"

x =string.gsub("hello world", "%w+", "%0 %0", 1)
     --> x="hello hello world"

x =string.gsub("hello world from Lua", "(%w+)%s*(%w+)", "%2 %1")
     --> x="world hello Lua from"

x =string.gsub("home = $HOME, user = $USER", "%$(%w+)", os.getenv)
     --> x="home = /home/roberto, user = roberto"

x =string.gsub("4+5 = $return 4+5$", "%$(.-)%$", function(s)
return loadstring(s)()
end)
     --> x="4+5 = 9"

local t ={name="lua", version="5.1"}
       x =string.gsub("$name-$version.tar.gz", "%$(%w+)", t)
     --> x="lua-5.1.tar.gz"
```

string.len (s)
Возвращает длину строки. Пустая строка "" имеет длину 0. Нулевые символы (\000 или \0) также считаются за один символ, поэтому строка "a\000bc\000" имеет длину 5.

string.lower (s)
Получает строку и возвращает копию этой строки, в которой все буквы в верхнем регистре заменяются строчными. Все остальные символы остаются без изменений. Определение того, что такое заглавная буква, зависит от текущей кодировки.

string.match (s, pattern [, init])
Ищет первое совпадение шаблона в строке s. Если он находит один, то match возвращает захваты из шаблона; иначе он возвращает nil. Если в шаблоне нет захватов, возвращает полное совпадение. Третий необязательный числовой аргумент init указывает, с чего начать поиск; его значение по умолчанию равно 1 и может быть отрицательным.

string.rep (s, n)
Возвращает строку, которая является сложением n копий строки s.

string.reverse (s)
Возвращает строку, которая является перевёрнутой строкой s.

string.sub (s, i [, j])
Возвращает подстроку s, которая начинается с индекса i и продолжается до j; i и j могут быть отрицательными. Если j отсутствует, то предполагается, что он равен -1 (что соответствует последнему символу строки). В частности, вызов string.sub (s, 1, j) возвращает первые j символов строки s, а string.sub (s, -i) возвращает последние i символов строки s.

string.upper (s)
Получает строку и возвращает копию этой строки, в которой все буквы в нижнем регистре заменяются на заглавные. Все остальные символы остаются без изменений. Определение строчной буквы зависит от текущей локали.

<u>Шаблоны</u>
Класс символов:
Классы символов используются для представления набора символов. Следующие комбинации разрешены в качестве описания класса символов:

- **x** (где x не является “волшебным” символом ^$()%.[]*+-?) представляет сам символ x;
- **.** (точка) представляет все символы;
- **%a** представляет все буквы;
- **%c** представляет все управляющие символы;
- **%d** представляет все цифры;
- **%l** представляет все буквы нижнего регистра;
- **%p** представляет все знаки препинания;
- **%s** представляет все знаки пробела;
- **%u** представляет все буквы верхнего регистра;
- **%w** представляет все буквы и цифры;
- **%x** представляет все шестнадцатеричные цифры;
- **%z** представляет все символы с представлением 0;

• **%x** (где x – любой не алфавитно-цифровой символ) представляет символ x. Это стандартный способ представления “волшебных” символов. Любые символы пунктуации (даже не “волшебные”) могут быть представлены через %, чтобы представить самих себя в шаблоне;

• **[set]** представляет класс, который является объединением всех символов в множестве. Диапазон символов можно задать, разделив конечные символы диапазона с помощью '-'. Все классы %x, описанные выше, также могут быть использованы в качестве компонентов в наборе. Все остальные символы в наборе представляют сами себя. Например, [%w_] (или [_%w]) представляет все алфавитно-цифровые символы плюс подчеркивание, [0-7] представляет восьмеричные цифры, а [0-7%l%-] представляет восьмеричные цифры, строчные буквы и знак '-';

• Взаимодействие между диапазонами и классами не определено. Поэтому шаблоны типа [%a-z] или [a-%%] не имеют смысла;

• **[^set]** представляет дополнение к множеству, где set интерпретируется, как указано выше.

Для всех классов, представленных одиночными буквами (%a,%c и т.д.), соответствующая заглавная буква представляет дополнение к классу. Например, %S представляет все символы, не являющиеся пробелами.

Определения буквенных и других групп символов зависят от текущей кодировки. В частности, класс [a-z] может быть не эквивалентен %l.

Элемент шаблона

Элементом шаблона может быть:

• Единственный символьный класс, который соответствует любому одиночному символу в классе;

• Один символьный класс, за которым следует “*” (соответствует 0 или более повторений символов в классе). Эти элементы повторения всегда будут соответствовать самой длинной последовательности;

• Один символьный класс, за которым следует «+» (соответствует 1 или более повторению символов в классе). Эти элементы повторения всегда будут соответствовать самой длинной последовательности;

• За символьным символом следует «-», который также соответствует 0 или более повторений символов в классе. В отличие от “*”, эти элементы повторения всегда будут соответствовать самой короткой последовательности;

• Один символьный класс, за которым следует “?” (означает либо 0 вхождений, либо 1 вхождение символа в класс);

• %n, где n – число от 1 до 9; такой элемент соответствует подстроке, равной n-ой захваченной строке (см. ниже);

• %bxy, где x и y – два различных символа; такой элемент соответствует строкам, которые начинаются с x и заканчиваются на y, причём количество x и y – равно. Это значит, что если кто-то читает строку слева направо, считая +1 за каждый x и -1 за каждый y, то последний y – это первый y, на котором счёт равен 0. Например, элемент %b() соответствует выражениям со сбалансированными круглыми скобками.

Шаблон

Шаблон представляет собой последовательность элементов шаблона. Символ “^” в начале шаблона привязывает совпадение к началу строки. “$” в конце шаблона привязывает совпадение к концу строки. На других позициях «^» и «$» не имеют специального значения и представляют сами себя.

Захваты

Шаблон может содержать подструктуры, заключенные в круглые скобки; они описывают захваты. Когда совпадение находится успешно, подстроки, соответствующие совпадениям, сохраняются (захватываются) для дальнейшего использования. Захваты нумеруются в соответствии с их левой скобкой. Например, в шаблоне «(a*(.)%w(%s*))» часть строки, соответствующей «a*(.)%W(%s*)», сохраняется как первый захват и поэтому имеет номер 1, совпадение символов "." записывается с номером 2, а элемент соответствия «%s*» имеет номер 3.
В специальном случае пустой захват () фиксирует текущую позицию строки (число). Например, если мы применим шаблон «()aa()» в строке «flaaap», будут два захвата: 3 и 5. Шаблон не может содержать внедрённые нули. Вместо них используйте %z.

6.1.19. Функции ввода и вывода

io.exists (path)
Проверяет, существует ли указанный путь (файл или каталог). Возвращает boolean.

io.readfile (file)
Считывает весь файл за раз. Возвращает содержимое файла в виде строки при успехе или nil при ошибке.

io.writefile (file, data)
Записывает данные в файл. Данные могут быть строкой (либо преобразовываться в строку) или таблицей. Когда данные – таблицы, каждый элемент таблицы заканчивается символом переноса строки. Возвращает в качестве результата boolean, если файл может быть открыт для записи, или nil, если файл недоступен.

Пример: запись состояния события в файл журнала, расположенный на подключенном USB-устройстве флэш-памяти:

```
1. value = knxdatatype.decode(event.datahex, dt.bool)
2. data =string.format('%s value is %s', os.date('%c'), tostring(value))
3. -- запись в конец файла с сохранением всех предыдущих данных
4. file =io.open('/mnt/usb/log.txt', 'a+')
5. file:write(data .. '\r\n')
6. file:close()
```

Вывод:

```
Mon Jan 3 05:25:13 2011 value is false
Mon Jan 3 05:25:14 2011 value is true
```

```
Mon Jan 3 05:25:32 2011 value is false
Mon Jan 3 05:25:33 2011 value is true
```

Пример: чтение данных из файла (настройки в формате key = value)

```
1. for line io.lines('/mnt/usb/config.txt')do
2. -- разделитель указан как '='
3.   items = line:split('=')
4. -- формат верный, если нашлось 2 объекта
5. if #items == 2 then
6.     key = items[1]:trim()
7.     value = items[2]:trim()
8.     alert('[config] %s = %s', key, value)
9. end
10. end
```

6.1.20. Функции управления скриптами

script.enable('scriptname')
Включает скрипт с именем scriptname.

script.disable('scriptname')
Выключает скрипт с именем scriptname.

status = script.status('scriptname')
Возвращает true/false, если скрипт найден, в противном случае возвращает nil.

6.1.21. Библиотека JSON

Примечание: json не загружена по умолчанию, используйте *require('json')* перед вызовом любой функции библиотеки.

json.encode (value)
Конвертирует переменную Lua в строку JSON. Выполнение скрипта останавливается в случае ошибки.

json.pencode (value)
Конвертирует переменную Lua в строку JSON в защищённом режиме, возвращает nil в случае ошибки.

json.decode (value)
Конвертирует строку JSON в переменную Lua. Выполнение скрипта останавливается при

ошибке.

json.pdecode (value)
Конвертирует строку JSON в переменную Lua в защищённом режиме, возвращает nil в случае ошибки.

6.1.22. Конверта ция

Уровень совместимости:
lmcore – синоним *cnv*.

cnv.strtohex (str)
Преобразует заданную двоичную строку в шестнадцатеричную строку.

cnv.hextostr (hex [, keepnulls])
Преобразует заданную шестнадцатеричную строку в двоичную строку. Нулевые символы по умолчанию игнорируются, но могут быть включены путём установки второго параметра в true.

cnv.tonumber (value)
Преобразует заданное значение в число, используя следующие правила: числа и действительные числовые строки обрабатываются как есть, boolean true равно 1, boolean false равно 0, всё остальное равно нулю.

cnv.hextoint(hexvalue, bytes)
Преобразует заданную шестнадцатеричную строку в целое число заданной длины в байтах.

cnv.inttohex(intvalue, bytes)
Преобразует заданное целое число в шестнадцатеричную строку заданной длины в байтах.

6.1.23. Битовые операторы

bit.bnot (value)
Бинарное “не”.

bit.band (x1 [, x2...])
Бинарное “и” между любым количеством аргументов.

bit.bor (x1 [, x2...])
Бинарное “или” между любым количеством аргументов.

bit.bxor (x1 [, x2...])

Бинарное “исключающее или” между любым количеством аргументов.

bit.lshift (value, shift)
Бинарный сдвиг влево.

bit.rshift (value, shift)
Бинарный сдвиг вправо.

6.1.24. Средства ввода и вывода

Библиотека ввода-вывода предоставляет два разных стиля для манипуляции файлами. Первый использует неявные файловые дескрипторы, т.е. существуют операции для установки входного файла по умолчанию и выходного файла по умолчанию, и все операции ввода/вывода выполняются над этими файлами. Второй стиль использует явные файловые дескрипторы.
При использовании неявных файловых дескрипторов все операции предоставляются таблицей io. При использовании явных файловых дескрипторов операция io.open возвращает файловый дескриптор, а затем все операции передаются как методы файлового дескриптора.
Таблица io также предоставляет три предопределенных файловых дескриптора со своими обычными значениями из C: *io.stdin*, *io.stdout* и *io.stderr*. Библиотека I/O никогда не закрывает эти файлы.
Если не указано иное, все функции ввода-вывода возвращают nil при ошибке (плюс сообщение об ошибке как второй результат и системный код ошибки в качестве третьего результата) и некоторое значение, отличное от nil, при успешном выполнении.

io.close ([file])
Эквивалентно *file:close()*. Без файла закрывает выходной файл по умолчанию.

io.flush ()
Эквивалентно *file:flush* к выходному файлу по умолчанию.

io.input ([file])
Когда вызывается с именем файла, открывает заданный файл (в текстовом режиме) и устанавливает его дескриптор в качестве входного файла по умолчанию. Когда вызывается с дескриптором файла, просто устанавливает этот дескриптор в качестве входного файла по умолчанию. Когда вызывается без параметров, возвращает текущий входной файл по умолчанию. В случае ошибок эта функция вызывает ошибку, а не возвращает код ошибки.

io.lines ([filename])
Открывает файл с заданным именем в режиме чтения и возвращает функцию итератора, которая при каждом вызове возвращает новую строку из файла. Следовательно, конструкция

for line in io.lines(filename) do body end

будет итерировать по всем строкам файла. Когда функция итератора обнаруживает конец файла, она вернет nil (чтобы завершить цикл) и автоматически закроет файл. Вызов io.lines() (без имени файла) эквивалентен io.input():lines(); то есть, перебираются строки входного файла по умолчанию. В этом случае функция не закрывает файл, когда цикл завершается.

io.open (filename [, mode])
Эта функция открывает файл в режиме, указанном в строке mode. Она возвращает новый дескриптор файла или, в случае ошибок, nil и сообщение об ошибке. Существуют следующие режимы:

- "r": режим чтения (по умолчанию);
- "w": режим записи;
- "a": режим добавления;
- "r+": режим обновления, предыдущие данные сохраняются;
- "w+": режим обновления, предыдущие данные стираются;
- "a+": режим добавления, предыдущие данные сохраняются, запись разрешена только в конец файла.

Строка mode также может иметь “b” в конце, что необходимо в некоторых системах, чтобы открыть файл в двоичном режиме. Эта строка аналогична используемой в стандартной функции C fopen.

io.output ([file])
Похожа на io.input, но работает с выходным файлом по умолчанию.

6.1.25. Математические функции

Эта библиотека является интерфейсом к стандартной библиотеке C math. Она предоставляет все функции внутри таблицы math.

math.abs (x)
Возвращает модуль x.

math.acos (x)
Возвращает arccos x (в радианах).

math.asin (x)
Возвращает arcsin x (в радианах).

math.atan (x)
Возвращает arctg x (в радианах).

math.atan2 (y, x)
Возвращает arctg y/x (в радианах), но использует знаки обоих параметров, чтобы найти квадрант результата. Она также корректно обрабатывает случай, когда переменная x равна нулю.

math.ceil (x)
Возвращает наименьшее целое число, большее или равное x.

math.cos (x)
Возвращает косинус x (x указывается в радианах).

math.cosh (x)
Возвращает гиперболический cos x.

math.deg (x)
Возвращает угол x, данный в радианах, в градусах.

math.exp (x)
Возвращает значение e^{x}
.

math.floor (x)
Возвращает наибольшее целое число, меньшее или равное x.

math.fmod (x, y)
Возвращает остаток от деления x на y, который округляет частное к нулю.

math.frexp (x)
Возвращает m и e, такие, что x = $m2^{e}$, e – целое число, а абсолютное значение m находится в диапазоне [0.5, 1) или ноль, когда x равно нулю.

math.huge
Значение HUGE_VAL, большее или равное любому другому числовому значению.

math.ldexp (m, e)
Возвращает $m2^{e}$
(e должно быть целым числом).

math.log (x)
Возвращает натуральный логарифм x.

math.log10 (x)
Возвращает десятичный логарифм x.

math.max (x, ···)
Возвращает максимальное значение среди данных аргументов.

math.min (x, ···)
Возвращает минимальное значение среди данных аргументов.

math.modf (x)
Возвращает два числа: целую часть x и дробную часть x.

math.pi
Число Пи.

math.pow (x, y)
Возвращает x^y. Для вычисления этого значения также можно использовать выражение x^y.

math.rad (x)
Возвращает угол x (данный в градусах) в радианах.

math.random ([m [, n]])
Эта функция является интерфейсом к простой функции псевдослучайного генератора rand, предоставляемой ANSI C (нет никаких гарантий относительно статистических свойств).
При вызове без аргументов возвращает равномерное псевдослучайное вещественное число в диапазоне [0,1). Когда вызывается с целым числом m, math.random возвращает однородное псевдослучайное целое число в диапазоне [1, m]. Когда вызывается с двумя целыми числами m и n, math.random возвращает равномерное псевдослучайное целое число в диапазоне [m, n].

math.randomseed (x)
Устанавливает x в качестве «начального числа» для псевдослучайного генератора: равные начальные числа производят равные последовательности чисел.

math.sin (x)
Возвращает sin x (x даётся в радианах).

math.sinh (x)
Возвращает гиперболический sin x.

math.sqrt (x)
Возвращает квадратный корень x. Для вычисления этого значения также можно использовать выражение x^0.5.

math.tan (x)
Возвращает tg x (x указывается в радианах).

math.tanh (x)
Возвращает гиперболический tg x.

6.1.26. Операции с таблицами

Эта библиотека предоставляет общие функции для работы с таблицами. Большинство функций в библиотеке таблиц предполагают, что таблица представляет собой массив или список. Для этих функций понятие «длина таблицы» подразумевает результат оператора длины.

table.concat (table [, sep [, i [, j]]])
Для массива, где все элементы являются строками или числами, возвращает

table[i]..sep..table[i+1]..sep..table[j]. Значением по умолчанию для sep является пустая строка, значение по умолчанию для i равно 1, а значение по умолчанию для j – длина таблицы. Если i больше, чем j, возвращает пустую строку.

table.insert (table, [pos,] value)

Вставляет значение элемента в позицию pos в таблице, перемещая другие элементы в открытое пространство, если это необходимо. Значение по умолчанию для pos равно n + 1, где n – длина таблицы, поэтому вызов table.insert(t, x) вставляет x в конец таблицы t.

table.maxn (table)

Возвращает наибольший положительный числовой индекс для данной таблицы или 0, если таблица не имеет положительных числовых индексов. Для выполнения задачи функция выполняет линейный обход всей таблицы.

table.remove (table [, pos])

Удаляет из таблицы элемент в позиции pos, сдвигая другие элементы, чтобы закрыть пространство, если это необходимо. Возвращает значение удаляемого элемента. Значение по умолчанию для pos равно n, где n – длина таблицы, поэтому вызов table.remove(t) удаляет последний элемент таблицы t.

table.sort (table [, comp])

Сортирует элементы таблицы в заданном порядке от table[1] до table[n], где n – длина таблицы. Если comp задано, то оно должно быть функцией, которая получает два элемента таблицы и возвращает true, когда первый меньше второго (так, чтобы после сортировки comp (a[i + 1], a[i]) не было истинным). Если comp не задано, используется стандартный оператор Lua “<”.
Алгоритм сортировки нестабилен, то есть, элементы, считающиеся равными, могут изменить свои позиции в процессе сортировки.

6.1.27. Функции операционной системы

os.date ([format [, time]])

Возвращает строку или таблицу, которая содержит дату и время и отформатирована в соответствии с заданным форматом строки. Если аргумент time присутствует, то он определяет время, которое нужно отформатировать (см. функцию os.time() для описания этого значения). В противном случае date форматирует текущее время.
Если формат начинается с символа «!», дата форматируется в «Согласованное универсальное время». Если формат – строка «*t», функция возвращает таблицу со следующими полями: год (четыре цифры), месяц (1-12), день (1-31), час (0-23), мин (0-59), сек (0-59), wday (день недели, воскресенье указано как 1), yday (день года) и isdst (флаг летнего/зимнего времени, логический).
Если формат не "*t", то функция возвращает дату в виде строки, отформатированной в соответствии с теми же правилами, что и у функции C strftime().
При вызове без аргументов date() возвращает представление даты и времени в форме, зависящей от системы и текущего языка (то есть os.date() эквивалентно os.date("%c")).

os.difftime (t2, t1)
Возвращает количество секунд, прошедших от момента времени t1 до момента времени t2. В POSIX, Windows и некоторых других системах это значение равно t2-t1.

os.execute ([command])
Эта функция эквивалентна функции C system(). Она передаёт команду, выполняемую оболочкой операционной системы, и возвращает код состояния, который зависит от системы. При отсутствии заданной команды функция возвращает ненулевое значение, если оболочка доступна, ноль в противном случае.

os.exit ([code])
Вызывает функцию C exit() с необязательным кодом для завершения главной программы. Значение по умолчанию для параметра code – код успешного завершения.

os.getenv (varname)
Возвращает значение переменной среды процесса varname или nil, если переменная не определена.

os.remove (filename)
Удаляет файл или каталог с заданным именем. Удаляемые каталоги должны быть пустыми. При неудачном удалении функция возвращает nil и строку, описывающую ошибку.

os.rename (oldname, newname)
Переименовывает файл или каталог с именем oldname в newname. При ошибке возвращает nil и строку, описывающую ошибку.

os.time ([table])
Возвращает текущее время, если вызывается без аргументов, или время, указанное в данной таблице. Эта таблица должна иметь поля year, month и day и может содержать поля hour, min, sec и isdst (описание этих полей см. в описании функции os.date).
Возвращаемое значение – число, значение которого зависит от вашей системы. В POSIX, Windows и некоторых других системах это число – количество секунд с некоторого заданного времени начала («epoch»). В других системах значение не указывается, а возвращаемое по времени число может использоваться только как аргумент date и difftime.

os.tmpname ()
Возвращает строку с именем файла, которую можно использовать для временного файла. Файл должен быть явно открыт до его использования и явным образом удален, когда он больше не нужен. В некоторых системах (POSIX) эта функция также создаёт файл с таким именем, чтобы избежать рисков. (В противном случае за время, которое пройдёт между получением имени и созданием файла, кто-то может успеть сам создать такой файл с неправильными разрешениями). Вы все равно должны открыть файл, чтобы использовать его или удалить (даже если вы его не используете).
По возможности следует использовать io.tmpfile, который автоматически удаляет файл по завершении программы.

6.1.28. Расширенн

ая библиотека функций

toboolean(value)
Преобразует заданное значение в логическое, используя следующие правила: *nil*, boolean *false*, *0*, *пустая строка*, *строка '0'* обрабатываются как false, всё остальное – как *true.*

string.split(str, sep)
Разделяет данную строку на части с помощью заданного разделителя. Возвращает таблицу Lua.

knxlib.decodeia(indaddressa, indaddressb)
Преобразует индивидуальный адрес, закодированный двоичным кодом, в строку Lua. Эта функция принимает один или два аргумента (интерпретируется как два одиночных байта).

knxlib.decodega(groupaddressa, groupaddressb)
Преобразует групповой адрес, закодированный двоичным кодом, в строку Lua. Эта функция принимает один или два аргумента (интерпретируется как два одиночных байта).

knxlib.encodega(groupaddress, separate)
Преобразует строку Lua в двоично-кодированный групповой адрес. Возвращает адрес группы для одного числа Lua, если второй аргумент равен nil или false, и два отдельных байта в противном случае.

ipairs (t)
Возвращает три значения: функцию итератора, таблицу t и 0, так что конструкция

```
for i,v in ipairs(t)do body end
```

будет перебирать пары (1, t [1]), (2, t [2]), ···, вплоть до первого целочисленного ключа, отсутствующего в таблице.

next (table [, index])
Позволяет программе перемещаться по всем полям таблицы. Первый аргумент – таблица, а второй аргумент – индекс в этой таблице. Next возвращает следующий индекс таблицы и связанное с ней значение. При вызове с nil в качестве второго аргумента next возвращает начальный индекс и связанное с ним значение. При вызове с последним индексом или с nil в пустой таблице next возвращает nil. Если второй аргумент отсутствует, то он интерпретируется как ноль. В частности, можно использовать next (t), чтобы проверить, является ли таблица пустой. Порядок, в котором перечисляются индексы, не указывается даже для числовых индексов. Чтобы пронумеровать таблицу в числовом порядке, используйте числовую функцию или функцию ipairs. Поведение next не определено, если во время обхода вы назначили какое-либо значение несуществующему полю в таблице. Однако вы можете изменять или удалять существующие поля.

pairs (t)

Возвращает три значения: функцию *next*, таблицу t и nil, так что конструкция

```
for k,v in pairs(t)do body end
```

будет перебирать все пары ключ-значение таблицы t.

tonumber (e [, base])

Пытается преобразовать аргумент в число. Если аргумент уже является числом или строкой, конвертируемой в число, то tonumber возвращает это число; в противном случае возвращается nil.

Необязательный аргумент указывает базу (система счисления) для интерпретации числа. Базой может быть любое целое число от 2 до 36 включительно. В основаниях выше 10 буква «A» (в верхнем или в нижнем регистре) представляет 10, «B» представляет 11 и так далее, с «Z», представляющим 35. В системе счисления 10 (по умолчанию) число может иметь десятичную часть, а также необязательную экспоненциальную часть. В других базах принимаются только целые числа без знака.

tostring (e)

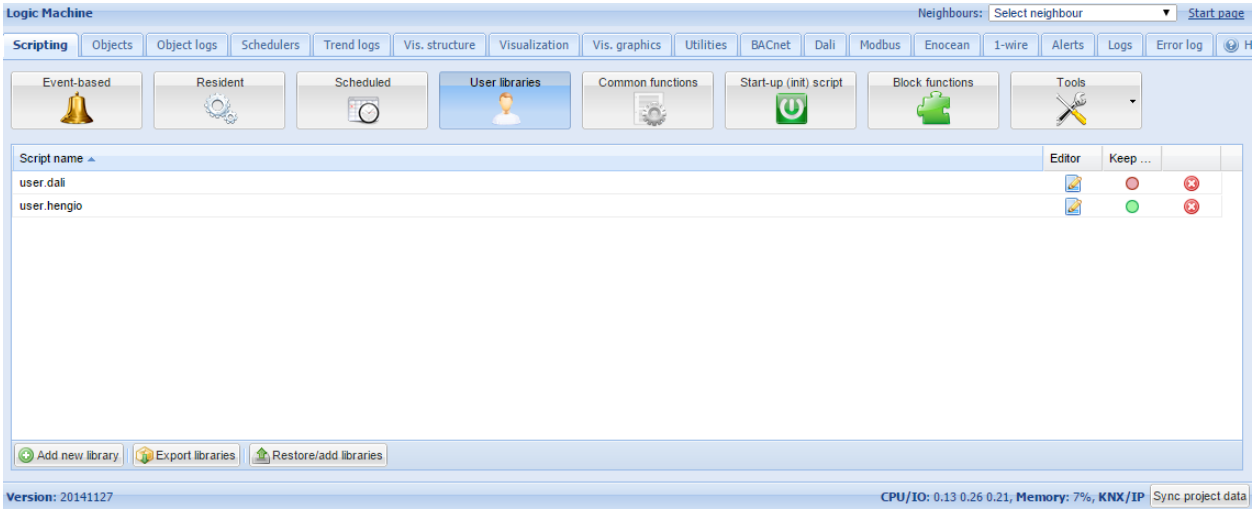
Получает аргумент любого типа и преобразует его в строку. Для более полного контроля преобразования чисел используйте string.format.

Если мета-таблица *e* имеет поле «__tostring», то tostring вызывает соответствующее значение с *e* в качестве аргумента и использует результат вызова в качестве результата.

type (v)

Возвращает тип аргумента в виде строки. Возможные результаты этой функции: «nil» (строка, а не значение nil), «number», «string», «boolean», «table», «function», «thread» и «userdata».

6.1.29. Пользовательские библиотеки



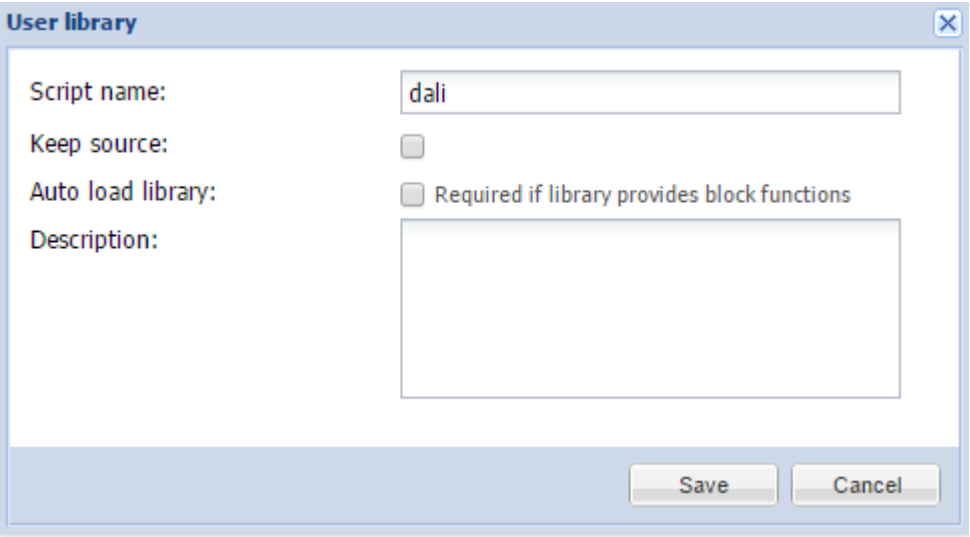
Пользовательские библиотеки обычно содержат пользовательские функции, которые можно

вызвать в скриптах.

Перед использованием библиотеки необходимо подключить её с помощью команды **require('user.test')**, если не включена функция *Auto load library* (автоматическая подгрузка библиотеки).

Защита кода

Для защиты кода библиотеки можно использовать опцию *keep source*. После отключения код компилируется в двоичной форме и не может быть просмотрен для дальнейшего редактирования. Если этот параметр включен, исходный код отображается в редакторе.



Auto load library означает, что библиотека будет загружена автоматически, и при использовании в скрипте её не нужно подключать с помощью **require().**

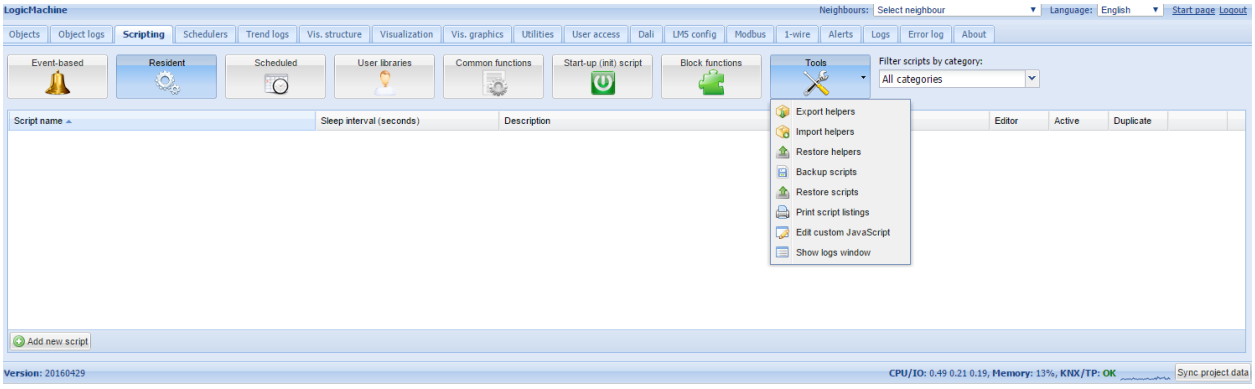
6.1.30. Общие функции

Common functions содержит библиотеку глобально используемых функций. Их можно вызывать из любого скрипта в любое время без специальных вложений, в отличие от пользовательских библиотек. В C*ommon functions* по умолчанию входят такие функции, как восход/закат, Email и т. д.

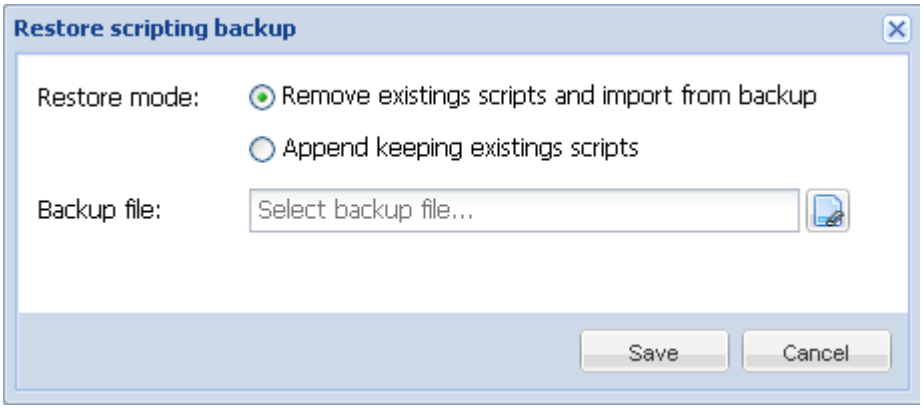
6.1.31. Скрипт запуска системы

Скрипт запуска используется для инициализации некоторых системных значений и значений шины при запуске системы. Скрипт запускается после каждой перезагрузки системы.

6.1.32. Инструменты



- ***Export helpers*** – экспорт фрагментов кода (помощников);
- ***Import helpers*** – импорт фрагментов кода;
- ***Restore helpers*** – восстановление стандартных фрагментов кода;
- ***Backup user scripts*** – резервное копирование всех скриптов в файл*.gz;
- ***Restore from archive*** – восстановление скриптов из архива (*.gz) с двумя вариантами:
  - Удаление существующих скриптов и импорт из архива;
  - Добавление скриптов из архива к существующим;



- ***Print script listings*** – показывает все скрипты с кодами в формате списка, отсортированного по категориям.

Category: Presence

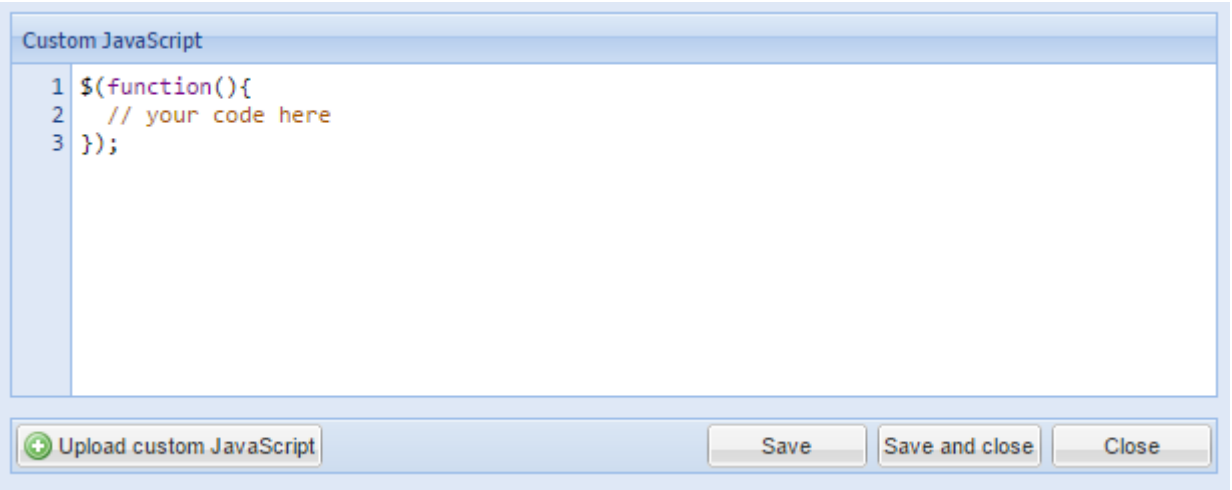
Presence simulator (id: 1)

Type: Resident
Active: Yes
Script sleep interval: 20

Synchronizes 0/0/2 value with 0/0/1

```
-- if object exists "presence" variable will be a table, nil otherwise
presence = knxobject.get('address', '0/0/1')

-- check that object exists and data has been decoded
if presence and presence.decoded then
  -- result will be either "value = true" or value = "false"
  alert('value = %s', tostring(presence.data))

  -- update 0/0/2 with the same data
  knxobject.write('0/0/2', presence.data, dt.bool)
else
  alert('read error')
end
```

- ***Show logs window*** – показывает логи в отдельном окне;
- ***Edit custom JavaScript*** – открывает окно для написания пользовательских функций на JavaScript.



С помощью пользовательских JavaScript-функций можно выполнять различные динамические задачи, например, определять короткое/длинное нажатие на визуализации или открывать конкретный план при запуске какого-либо адреса grp. Например, эта функция автоматически открывает страницу IP-камеры при нажатии кнопки Intercom:

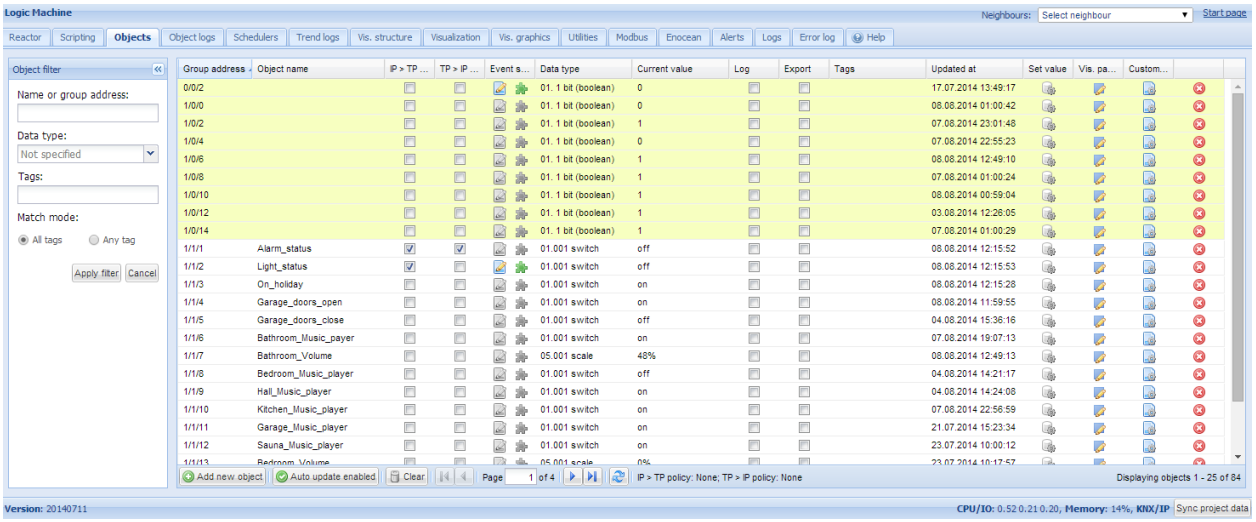
```
$(function(){
  if (typeof objectStore !== 'undefined') {
    var id = Scada.encodeGroupAddress('1/1/2');

    objectStore.addListener(id, function(object, type) {
      if (type == 'value') {
        showPlan(69);
      }
    });
  }
});
```

Посмотреть больше примеров можно здесь:
http://forum.logicmachine.net/showthread.php?tid=275

6.2. Вкладка Objects

Список сетевых объектов KNX отображается во вкладке *Objects*. Объект может попасть в список несколькими способами:

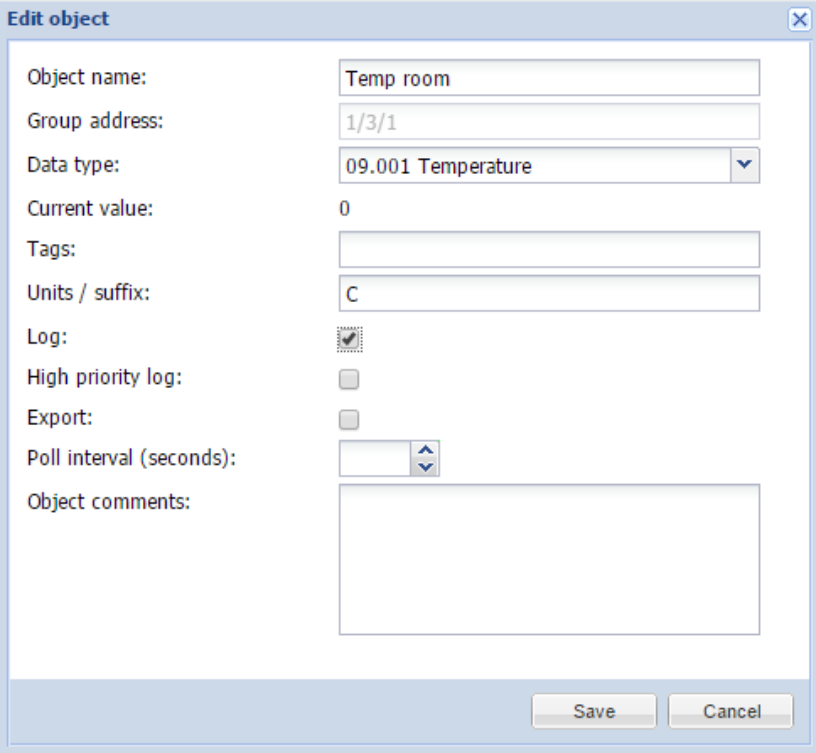
- через прослушивание телеграмм с неизвестных групповых адресов (если включено в *Utilities*);
- через добавление вручную;
- через импорт файла ESF (в *Utilities*).



6.2.1. Параметры объекта

Чтобы изменить настройки для существующего или нового объекта, нажмите на

соответствующую запись в списке.

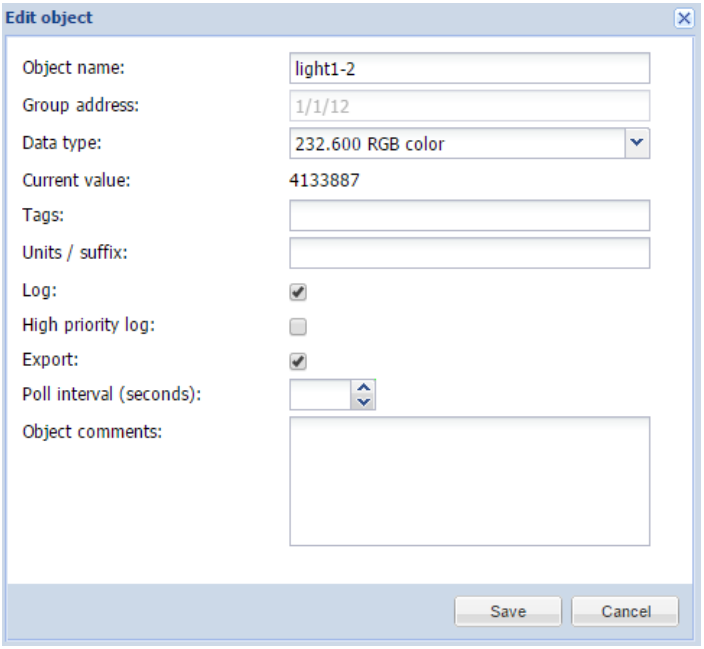


- ***Object name*** – имя объекта;
- ***Group address*** – групповой адрес объекта;
- ***Data type*** – тип данных KNX объекта. Для корректной работы его необходимо устанавливать сразу, как только LM находит новый объект;
- ***Units / suffix*** – единицы измерения, которые будут отображаться на визуализации наряду со значением;
- ***Log*** – включить ведение логов для этого объекта. Логи будут отображаться в меню «Логи объектов»;
- ***High priority log*** – пометить объект для высокоприоритетного ведения логов. Когда база данных логов переполняется, сначала очищаются стандартные логи, и только потом – логи высокого приоритета;
- ***Export*** – сделать объект видимым для удаленных запросов XML и в сети BACnet (если используется функция шлюза KNX – BACnet);
- ***Poll interval (seconds)*** – выполнять автоматическое считывание объекта через некоторый промежуток времени;
- ***Tags*** – присвоить этому объекту некоторый тег, который впоследствии может быть использован при написании скриптов, например All_lights_first_floor;
- ***Current value*** – текущее значение объекта;
- ***Object comments*** – комментарий к объекту.

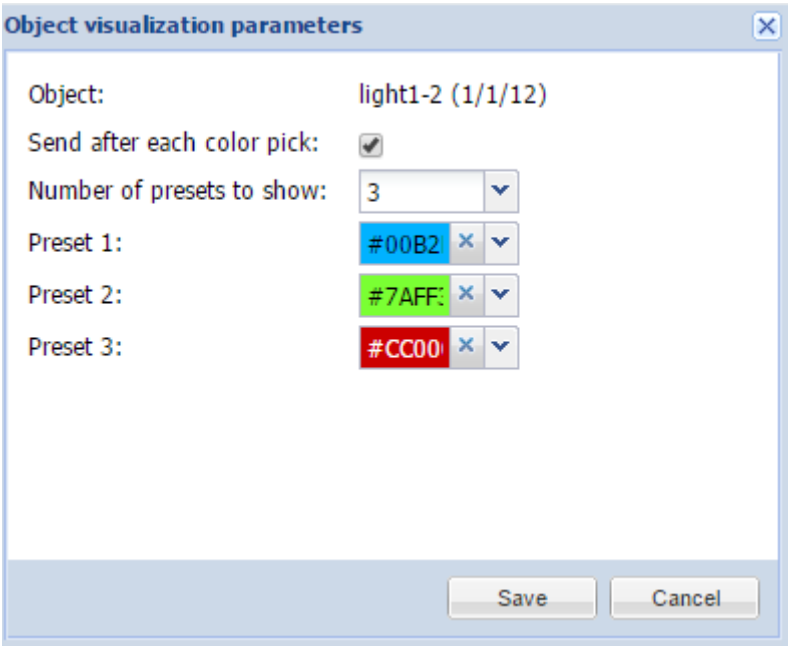
Существует возможность сортировки объектов по одному из следующих параметров: Object name, Group address, Data type, Current value, Tags, Object comments.

6.2.2. Объект группы RGB

В списке типов данных присутствует специальный тип *RGB color*.



В параметрах визуализации можно применить для объекта следующие настройки:



Send after each color pick – включает автоматическую отправку телеграммы в шину KNX после каждого выбора цвета из палитры;
Number of presets to show – количество предопределённых цветов в визуализации;
Preset 1..6 – предопределённые цвета.

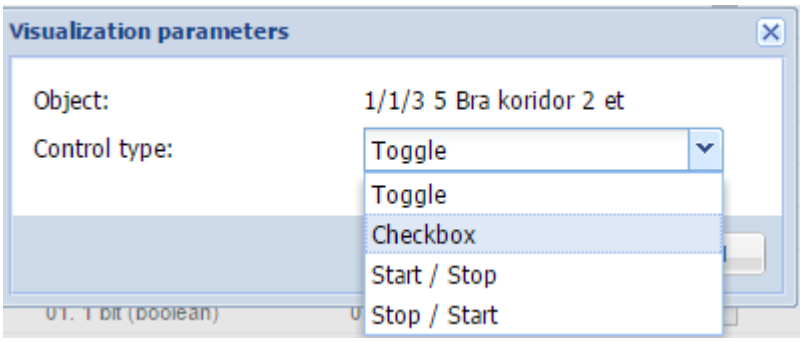
При добавлении на визуализацию объекта с типом данных RGB color появляется панель выбора

цвета с предопределёнными цветами.

6.2.3. Параметры визуализации объекта

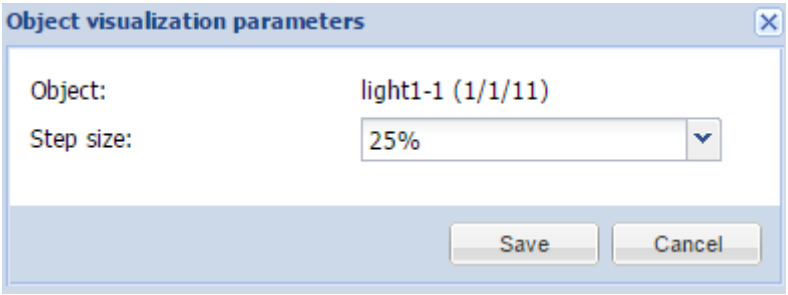
Нажав на иконку соответствующего объекта, вы можете установить конкретные параметры визуализации для этого типа объекта.

<u>1 bit (boolean)</u>



- ***Control type*** – тип визуального элемента управления, который будет отображаться в Touch Visualization
  - Toggle – переключатель; 
  - Checkbox – флажок; 
  - Start/Stop – пока иконка нажата, посылается одно значение, когда отпущена, посылается противоположное значение.

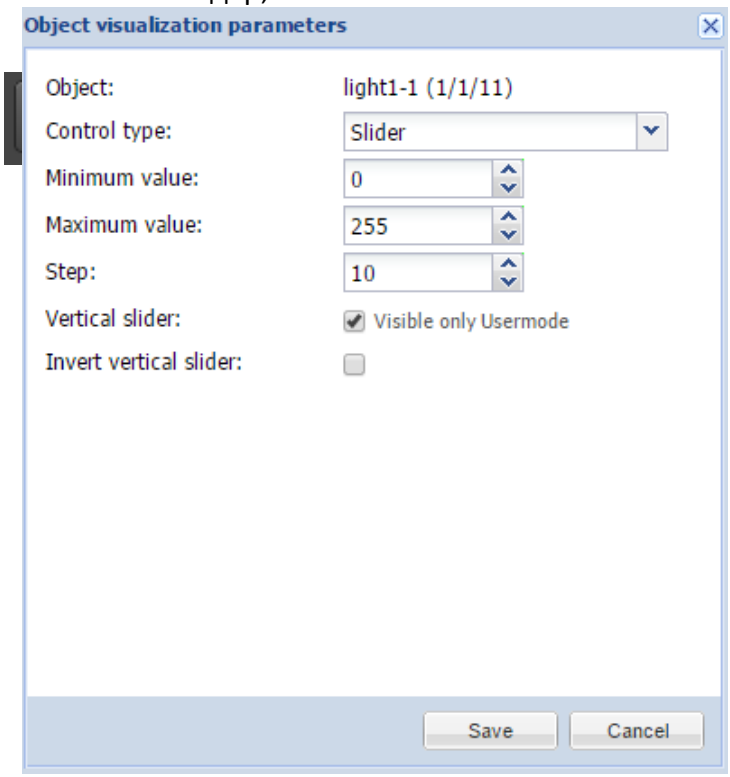
<u>4 bit (3 bit controlled)</u>



- ***Step size*** – размер шага изменения объекта, например, для управления жалюзи.

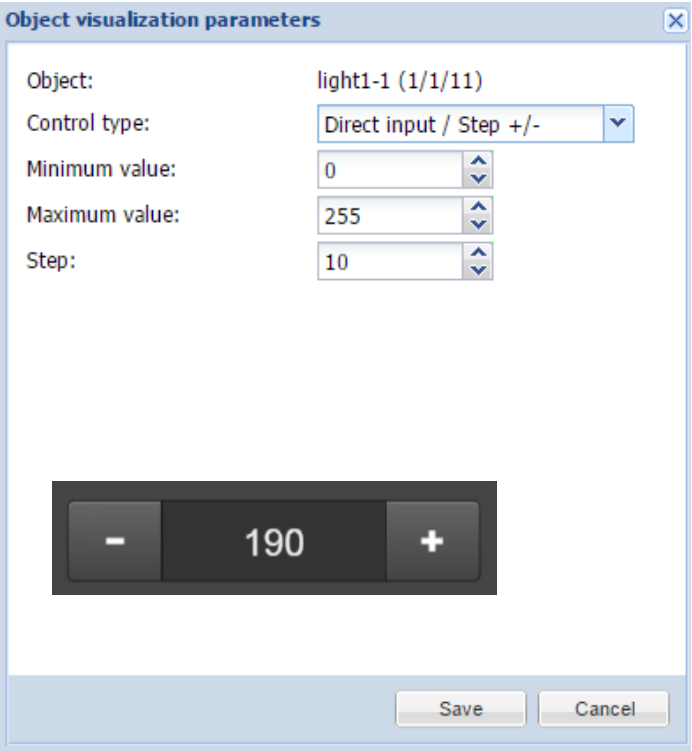
1 byte и 4 byte floating point

- Control type – тип визуального элемента управления;
  - Slider – слайдер;



    - *Minimum value* – минимальное значение на слайдере;
    - *Maximum value* – максимальное значение на слайдере;
    - *Step* – шаг за одно движение слайдера;
    - *Vertical slider* – специальная опция для визуализации в режиме пользователя;
    - *Invert vertical slider* – инвертировать вертикальный слайдер, чтобы максимум был сверху.

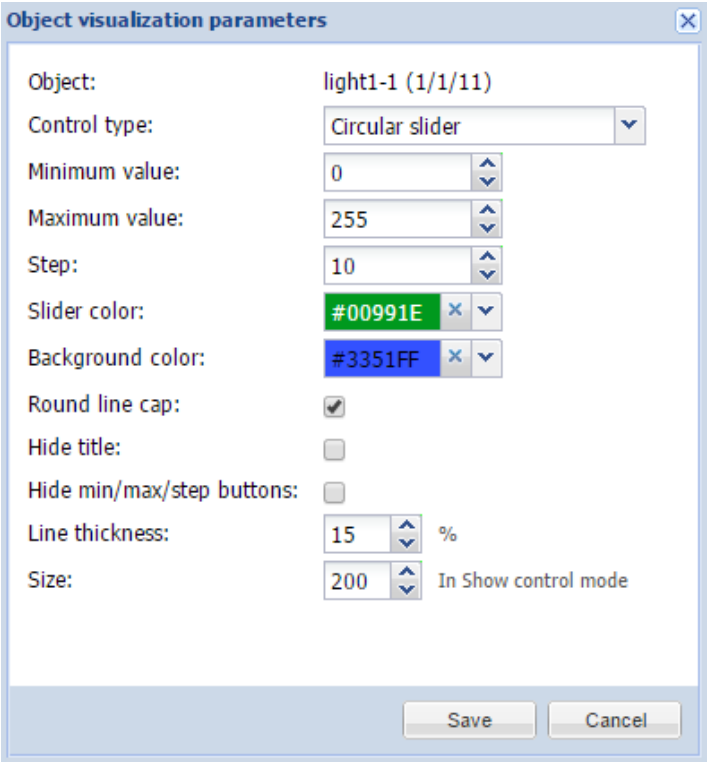
  - Direct input / Step +/- – ввод через панель управления;



- *Minimum value* – минимальное значение на панели управления;
- *Maximum value* – максимальное значение на панели управления;
- *Step* – шаг за одну смену позиции;

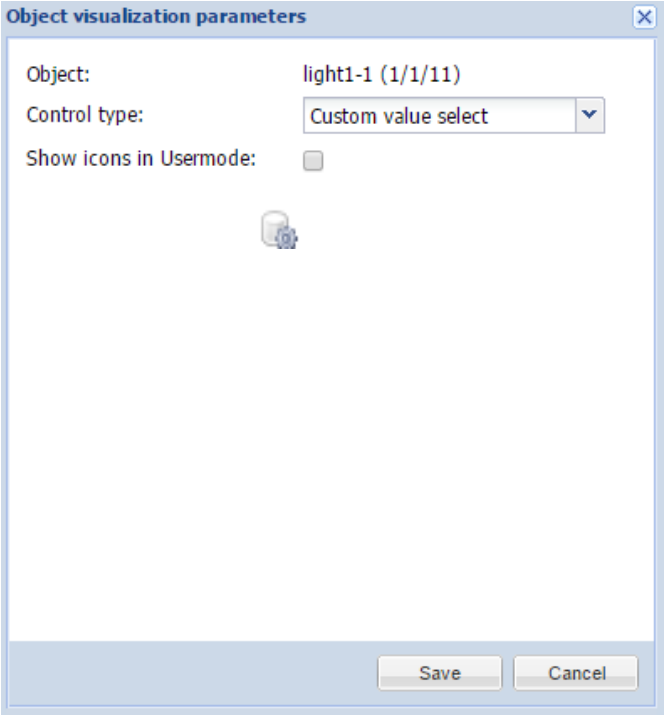
o Circular slider – круговой слайдер;

- *Minimum value* – минимальное значение на слайдере;
- *Maximum value* – максимальное значение на слайдере;
- *Step* – шаг за одну смену позиции;
- *Slider color* – цвет слайдера;
- *Background color* – цвет фона слайдера;
- *Round line cap* – закруглить концы слайдера;
- *Hide title* – спрятать заголовок;
- *Hide min/max/step buttons* – спрятать кнопки минимума, максимума, шага;
- *Line thickness* – толщина линии слайдера;
- *Size* – размер слайдера в пикселях.



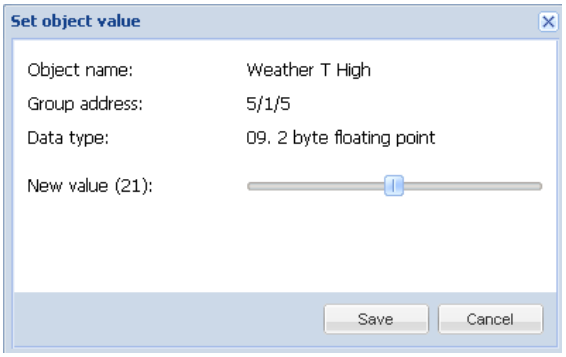
- Custom value select – выбор из списка пользовательских значений. Пользовательские значения должны быть заданы в ;

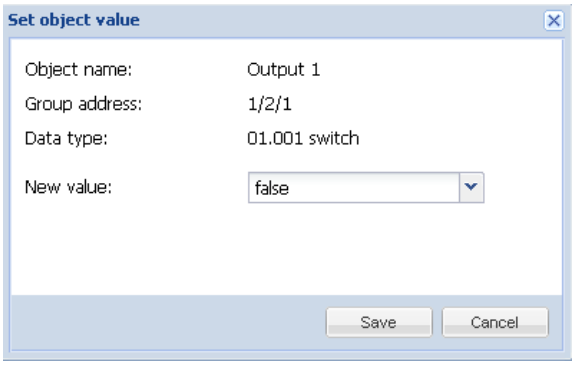
  - *Show icons in Usermode* – показывать в визуализации иконки вместо значений объекта. Можно выбрать их из заданных наборов либо загрузить пользовательские. Иконки должны быть определены в конструкторе визуализации как ***Дополнительные иконки***.



6.2.4. Изменение состояния объекта

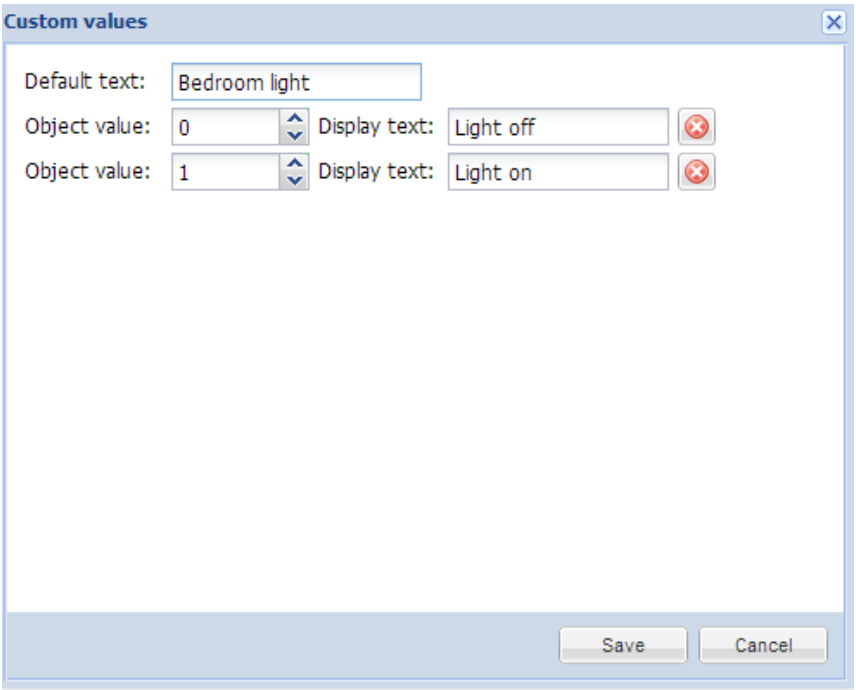
В списке объектов, нажав на кнопку , вы можете изменить состояние объекта.
Внешний вид окна изменения значения зависит от того, какие параметры визуализации заданы для конкретного объекта.





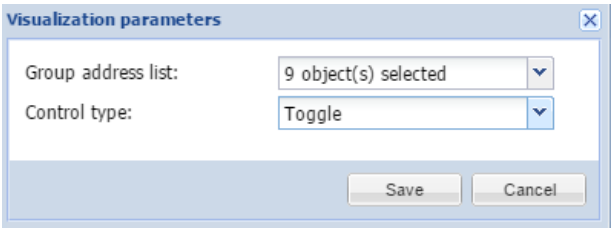
6.2.5. Пользовательские значения

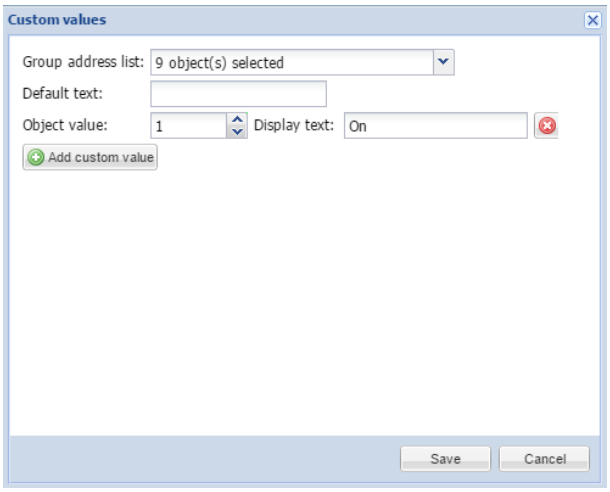
Если в объекте необходимо применить пользовательские значения, используйте значок для их настройки (только для типов данных Boolean и Integer).

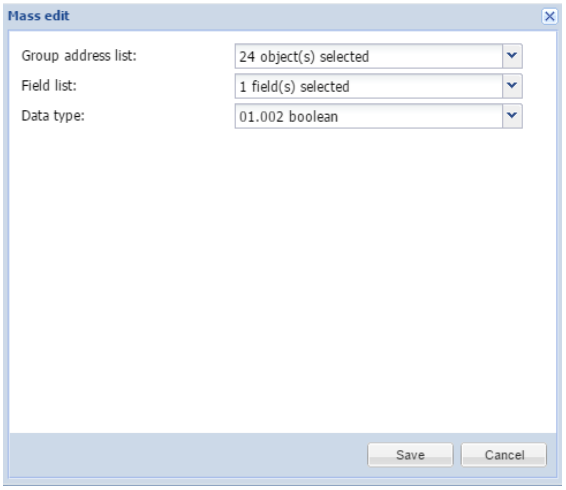


6.2.6. Панель управления объектами

- ***Add new object*** – вручную добавить новый объект в список;
- ***Auto update enabled*** – включить или отключить автоматическое обновление списка объектов;
- ***Clear*** – очистить список групповых адресов;
- ***Next/Previous page*** – перейти на следующую или предыдущую страницу;
- ***Refresh*** – обновить список объектов;
- ***Mass edit*** – отредактировать несколько объектов по текущему фильтру: свойства объекта, параметры визуализации или пользовательские значения.







- ***Mass delete*** – удалить несколько объектов по текущему фильтру либо все объекты без имени.

В нижней части экрана также есть такая панель:

- ***CPU/IO*** – средняя нагрузка. Представляет собой усреднённую нагрузку системы за определённый период времени. Она условно отображается в виде трёх чисел, которые характеризуют нагрузку системы в течение последних одно-, пяти- и пятнадцатиминутных периодов. Чем ниже число, тем лучше.

  Примечание! Проверьте запущенные задачи, если нагрузка превышает уровень 0.70!

  Подробные сведения о UNIX-подобном вычислении средней нагрузки можно узнать здесь:

  https://ru.wikipedia.org/wiki/Load_Average#Вычисление_средней_загрузки_в_UNIX

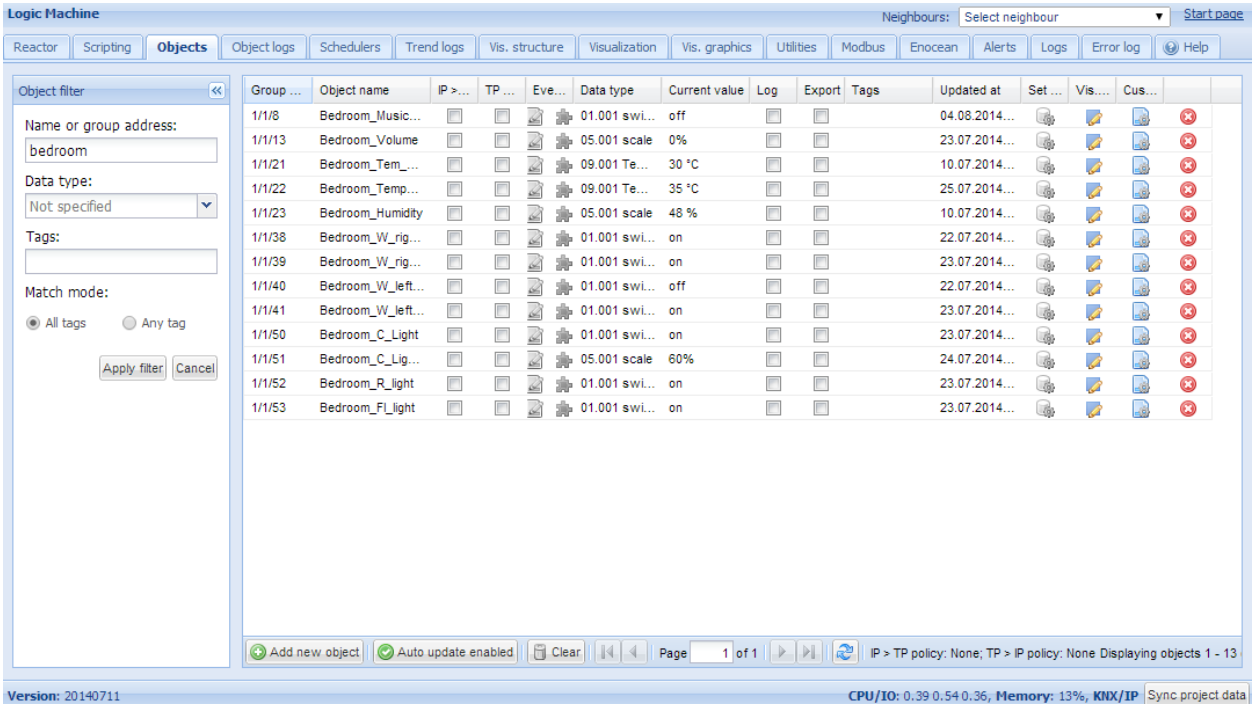
- ***Memory*** – использование памяти в %;
- ***KNX/IP / KNX/TP*** – тип соединения с шиной KNX. Если выбран KNX/TP и шина недоступна, здесь отобразится оповещение об ошибке;
- ***Sync project data*** – сохранение всех данных проекта во внутреннюю память. По

умолчанию данные сохраняются из ОЗУ во внутреннюю память один раз в 30 минут или при отправке команд Reboot или Shutdown;

- ***KNX statistics graphs*** – показывает среднюю нагрузку на шину KNX.

6.2.7. Фильтр объектов

В левой части списка можно отфильтровать объекты. Чтобы выполнить фильтрацию, укажите имя, адрес группы, тег или тип данных объекта и нажмите кнопку *Apply filter*.



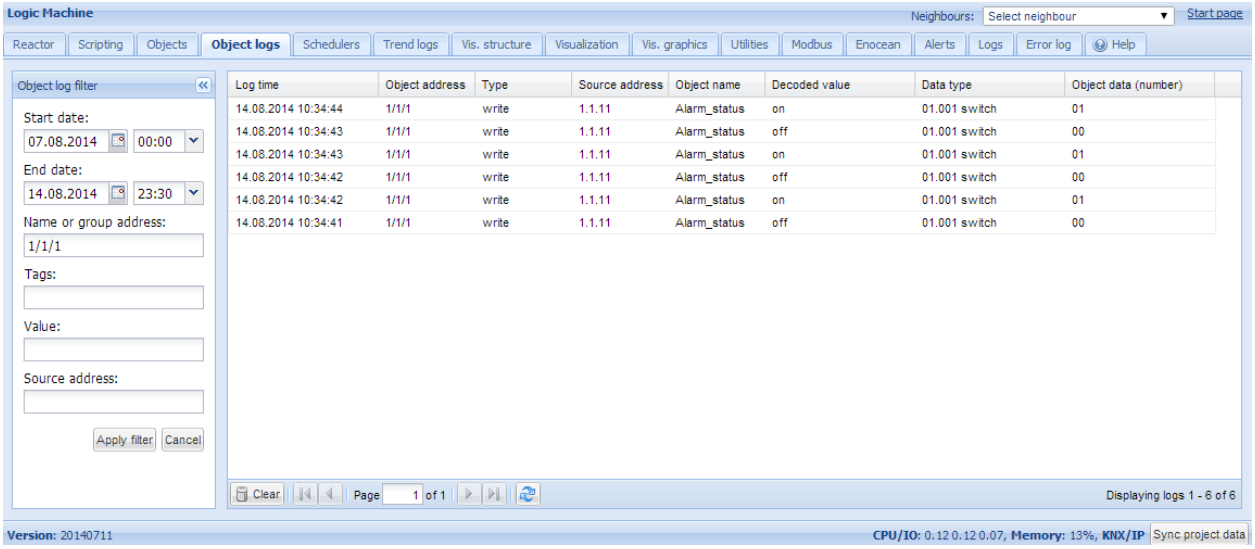
Режимы совпадения:

All tags – функция AND: объект должен иметь все указанные теги;

Any tag – функция ИЛИ: объект должен иметь хотя бы один из указанных тегов.

6.3. Вкладка Object logs

История телеграмм объектов доступна в логах объектов. После того как для объекта включено логирование, вся его дальнейшая история будет заноситься в журнал.

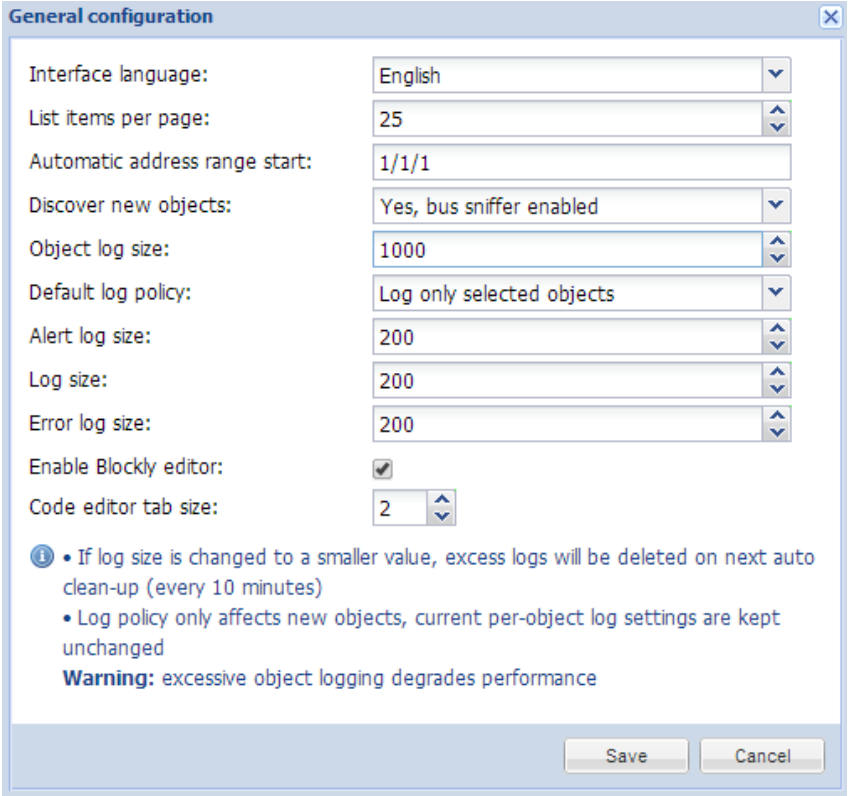


Если необходимо найти конкретную информацию о периоде или объекте, можно воспользоваться фильтрацией:

- ***Start date*** – дата и время начала поиска в журнале;
- ***End date*** – дата и время окончания поиска в журнале;
- ***Name or group address*** – конкретное имя или групповой адрес объекта;
- ***Tags*** – теги;
- ***Value*** – определённое значение;
- ***Source address*** – определённый физический адрес источника.

Можно удалить все логи, нажав на кнопку *Clear.*

Размер журнала задается *в Utilities → General Configuration.*

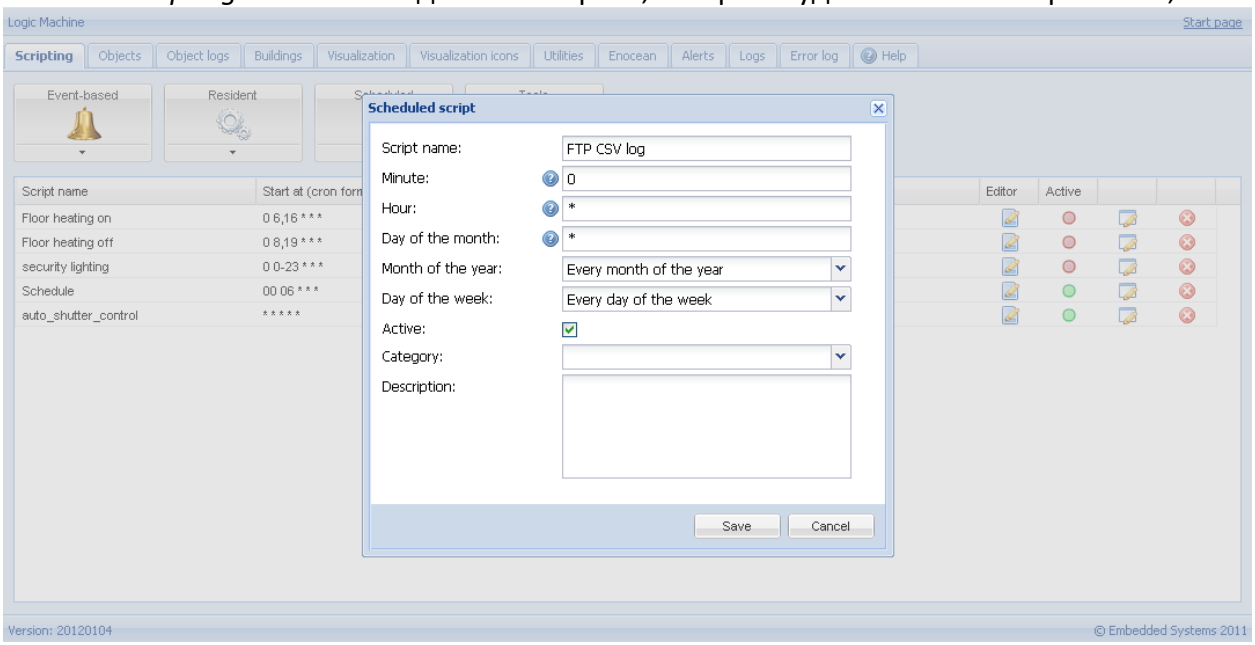


6.3.1. Экспорт логов

<u>Пример.</u>

Допустим, нужно раз в час создавать файл CSV со всеми логами объектов и отправлять на внешний ftp-сервер с IP-адресом 192.168.1.11, логином *ftplogin* и паролем *ftppassword*.

- В *Scripting -> Scheduled* добавьте скрипт, который будет выполняться раз в час;



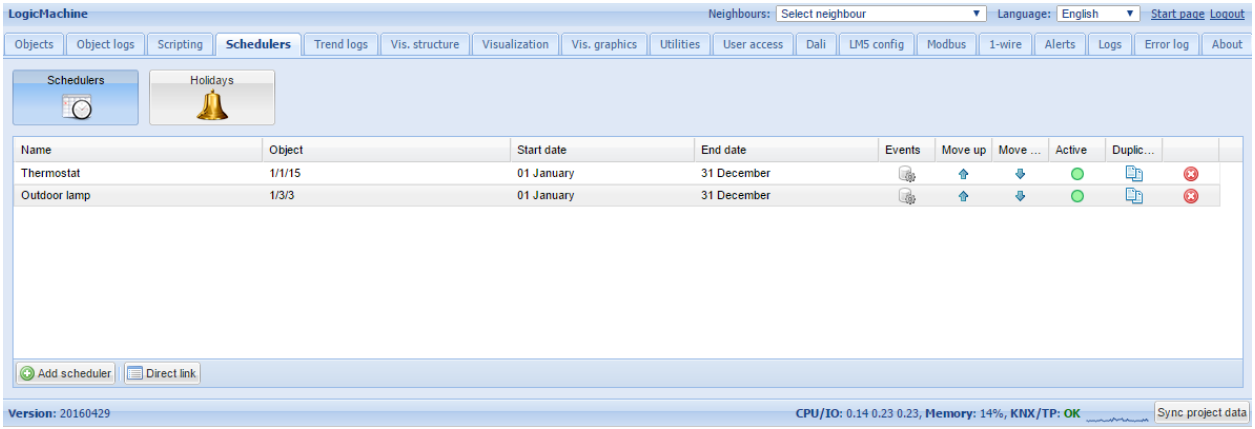
- В редакторе напишите скрипт, используя данный код:

```
1.  require('socket.ftp')
2.
3.  -- файл ftp
4.  ftpfile=string.format('ftp://ftplogin:ftppassword@192.168.1.11/%s.csv', os.date('%Y-%m-
    %d_%H-%M'))
5.  -- получение данных о прошлом часе (3600 секунд)
6.  logtime=os.time() - 60*60
7.
8.  -- список объектов по id
9.  objects ={}
10.
11. -- объекты со включенным логгированием
12. query ='SELECT address, datatype, name FROM objects WHERE disablelog=0'
13. for _, object in ipairs(db:getall(query)) do
14.   objects[tonumber(object.address)]={
15. datatype=tonumber(object.datatype),
16.     name =tostring(object.name or''),
17. }
18. end
19.
20. -- буфер csv
21. buffer ={'"date","address","name","value"'}
22.
23. -- получение логов объекта
24. query='SELECT src, address, datahex, logtime, eventtype FROM objectlog WHERE logtime>= ?
    ORDER BY id DESC'
25. for _, row in ipairs(db:getall(query, logtime))do
26.   object = objects[tonumber(row.address)]
27.
28. -- объект найден, а тип события – запись
29. if object and row.eventtype=='write' then
30. datatype=object.datatype
31.
32. -- проверка, что объект datatype задан
33. if datatype then
34. -- декодирование данных
35.       data =knxdatatype.decode(row.datahex, datatype)
36.
37. -- удаление пустых символов из символьного/строкового datatype
38. if datatype==dt.char or datatype==dt.string then
39.         data =data:gsub('%z+', '')
40. -- приведение даты к виду DD.MM.YYYY
41. elseifdatatype==dt.date then
42.         data =string.format('%.2d.%.2d.%.2d', data.day, data.month, data.year)
43. -- приведение времени к виду HH:MM:SS
44. elseif datatype==dt.time then
45.         data =string.format('%.2d:%.2d:%.2d', data.hour, data.minute, data.second)
46. end
47. else
48.       data =''
49. end
50.
51. -- форматирование ряда csv
52. logdate=os.date('%Y.%m.%d %H:%M:%S', row.logtime)
53. csv=string.format('%q,%q,%q,%q', logdate, knxlib.decodega(row.address), object.name,
    tostring(data))
54.
55. -- добавление в буфер
56. table.insert(buffer, csv)
57. end
```

```
58. end
59.
60. -- загрузка в ftp выполняется, только если в буфере есть данные
61. if #buffer > 1 then
62.   result, err =socket.ftp.put(ftpfile, table.concat(buffer, '\r\n'))
63. end
64.
65. -- если во время загрузки случилась ошибка
66. if err then
67.   alert('FTP upload failed: %s', err)
68. end
```

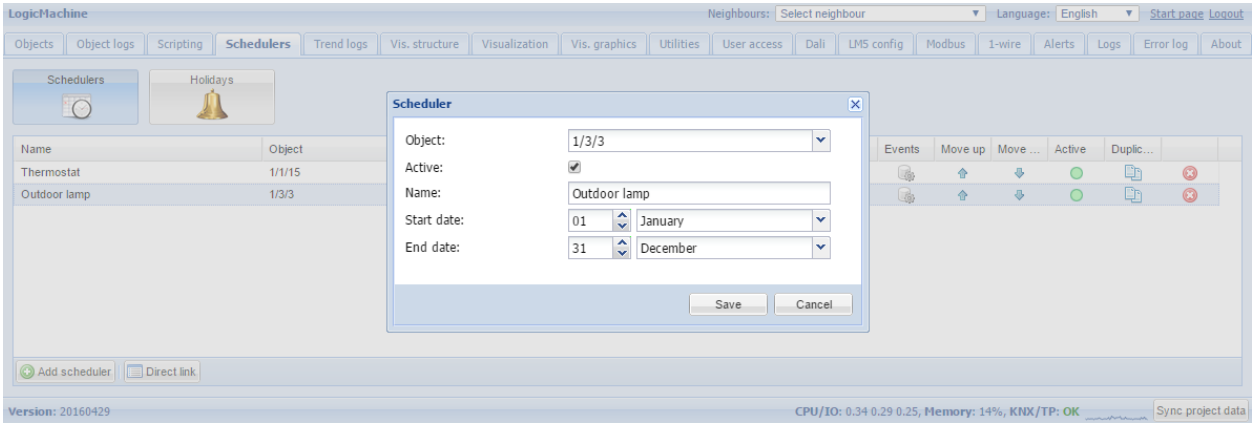
6.4. Вкладка Shedulers

На вкладке *Shedulers* настраиваются планировщики, предназначенные для конечного пользователя. Планировщики позволяют пользователю управлять значениями группового адреса KNX в зависимости от даты или дня недели.



6.4.1. Добавление нового планировщика

Нажав на *Schedulers → Add new scheduler,* вы увидите следующее окно:

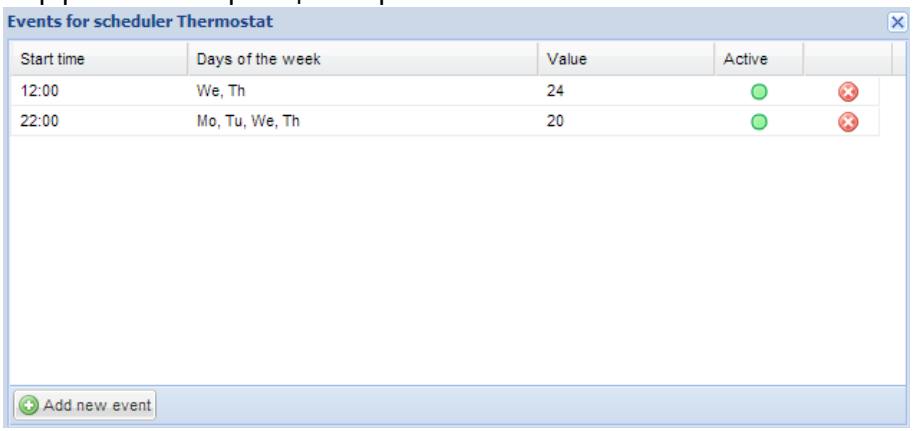


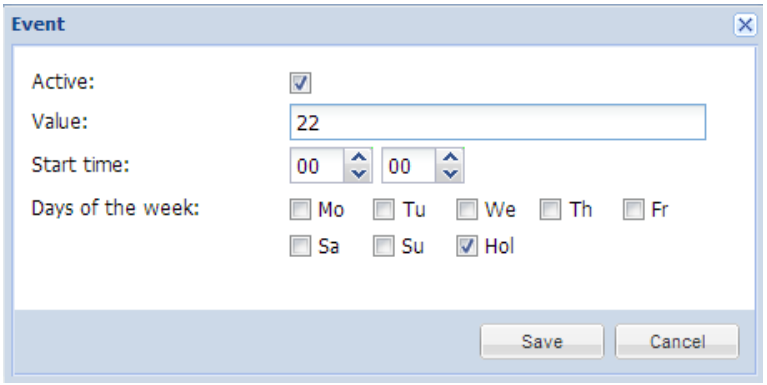
- ***Object*** – адрес группового объекта, который будет контролироваться планировщиком;

- ***Active*** – включен или выключен планировщик;
- ***Name*** – название планировщика;
- ***Start date*** – дата начала работы планировщика;
- ***End date*** – дата конца работы планировщика.

6.4.2. События планировщика

Событие может быть добавлено как в административном интерфейсе, так и в специальном интерфейсе планировщика в режиме пользователя.

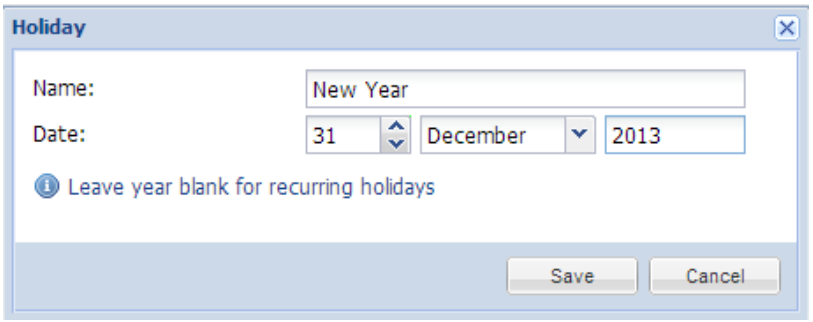




- ***Active*** – включено или выключено событие планировщика;
- ***Value*** – значение, которое будет отправлено на групповой адрес при вызове события;
- ***Start time*** – время вызова события;
- ***Days of the week*** – дни недели, в которые будет вызываться событие; ***Hol*** – праздники, которые задаются на вкладке *Holidays.*

6.4.3. Праздники

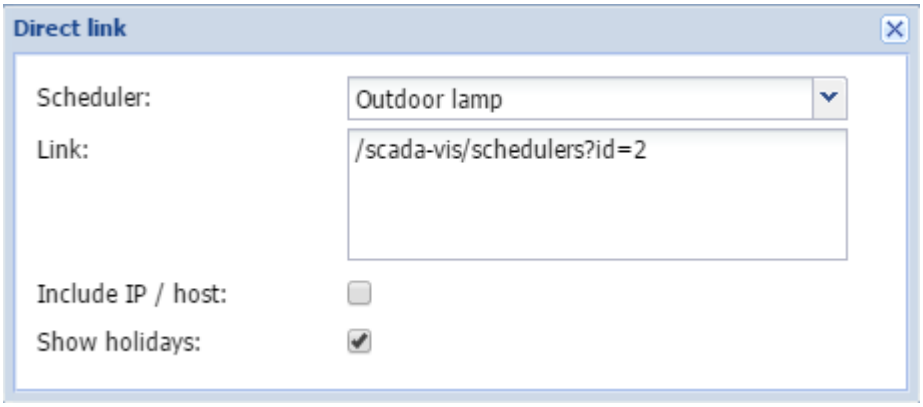
После того как событие будет помечено для запуска в *Hol*, будут активированы записи праздников.



- ***Name*** – название праздника;
- ***Date*** – дата праздника.

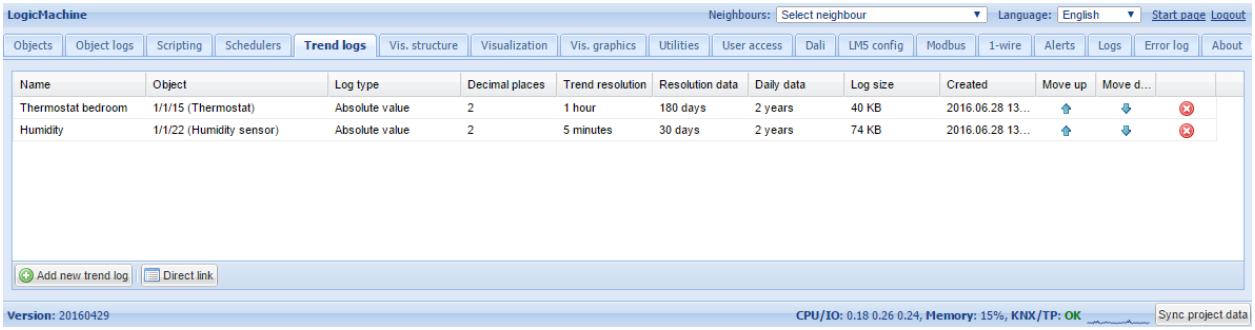
6.4.4. Прямая ссылка

Для получения прямой ссылки на конкретный планировщик нажмите на кнопку *Direct link* в левой нижней части.

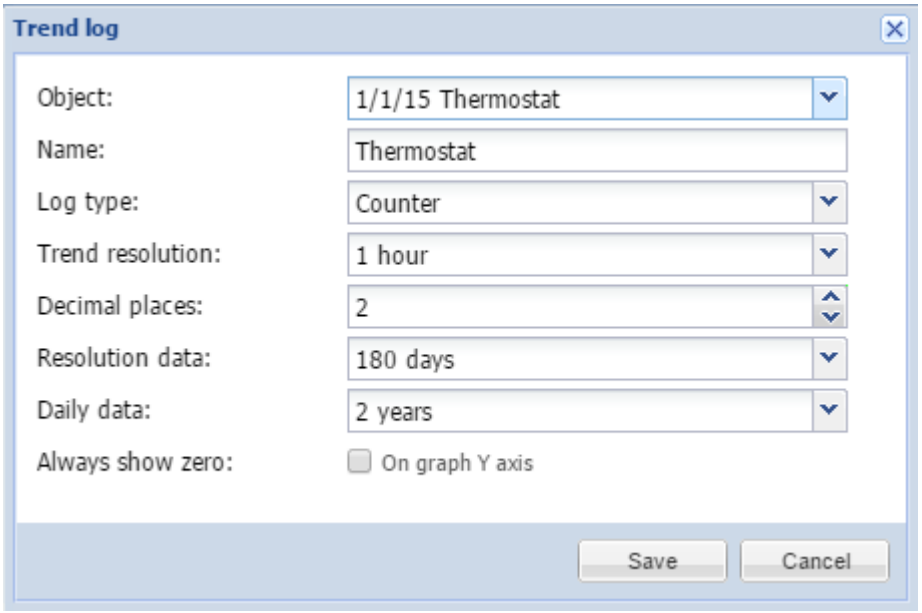


6.5. Вкладка Trend logs

Вкладка *Trend logs* предназначена для администрирования трендлогов пользовательского режима. Трендлоги используются для просмотра значений объекта в графическом виде и для сравнения с другими значениями периода.



6.5.1. Добавление нового трендлога

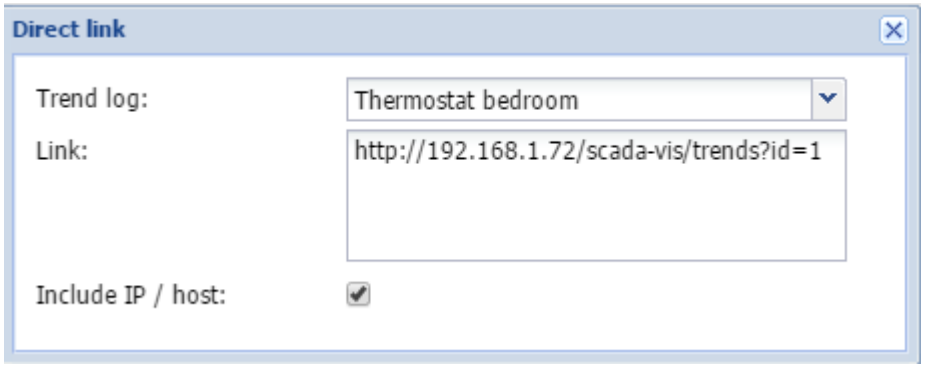


- ***Object*** – групповой адрес объекта;
- ***Name*** – название трендлога;
- ***Log type [Counter, Counter with negative delta, Absolute value]*** – тип лога. *Counter* используется для подсчёта значений, *Absolute value* – сохраняет значение;
- ***Trend resolutions [5 min .. 1 hour]*** – интервал времени, для которого в трендлоге будет отображаться среднее значение за 1 шаг. Например, если указан 1 час, то шаг будет равен 1 часу со средним показателем 60 значений;
- ***Decimal places*** – количество знаков после запятой;
- ***Resolution data*** – количество точек данных для хранения в выбранном разрешении;
- ***Daily data*** – время хранения средних значений ежедневных данных за определённый промежуток времени.

Примечание! Хранение одной точки данных тренда занимает 8 байт флэш-памяти. Например, хранение значения, которое берётся раз в 10 минут, будет потреблять ~ 0,4 МБ флеш-памяти каждый год.

6.5.2. Прямые ссылки

Для получения прямой ссылки на конкретный трендлог нажмите на кнопку *Direct link* в левой нижней части.



6.5.3. Функции трендлогов

Для обработки сохранённой информации можно использовать встроенные функции трендлогов в скриптах.

Подключите библиотеку перед вызовом функций трендлогов:

require('trends')

Получение одного или нескольких значений за данный период:

trends.fetch(name, dates, resolution)
trends.fetchone(name, dates, resolution)

Параметры:

- ***name*** – необходимый, название трендлога;
- ***dates*** – необходимый, таблица Lua с двумя полями ‘start’ и ‘end’, в которых хранится таблица с полями ‘year’, ‘month’, ‘day’;
- ***resolution*** – необязательный, разрешение трендлога; если не задан, будет использовать 86400 для получения ежедневных данных.

Возвращаемые значения:

- ***fetch*** возвращает таблицу Lua со значениями для данного периода или nil при ошибке. Количество значений зависит от настроек периода хранения, разрешения и количества данных;
- ***fetchone*** возвращает одно значение за данный период или nil при ошибке.

Пример:

```
require('trends')

-- получит данные за период с 00:00 2016.04.15 по 00:00 2016.04.16
dates = {
  ['start'] = { year = 2016, month = 4, day = 15 },
  ['end'] = { year = 2016, month = 4, day = 16 },
}
```

```
-- получит текущее значение
day = trends.fetchone('Gas', dates)

-- получит данные за прошлый год
dates = {}
dates['start'] = os.date('*t')
dates['start'].year = dates['start'].year - 1
dates['end'] = os.date('*t')

-- получит предыдущее значение
yearly = trends.fetch('Gas', dates, 86400)
```

➢ Значение ***trends.NaN*** используется для точек, которые содержат недопустимые значения или не могут быть найдены. Значение по умолчанию равно 0, но оно также может быть установлено на 0/0 (NaN – не число).

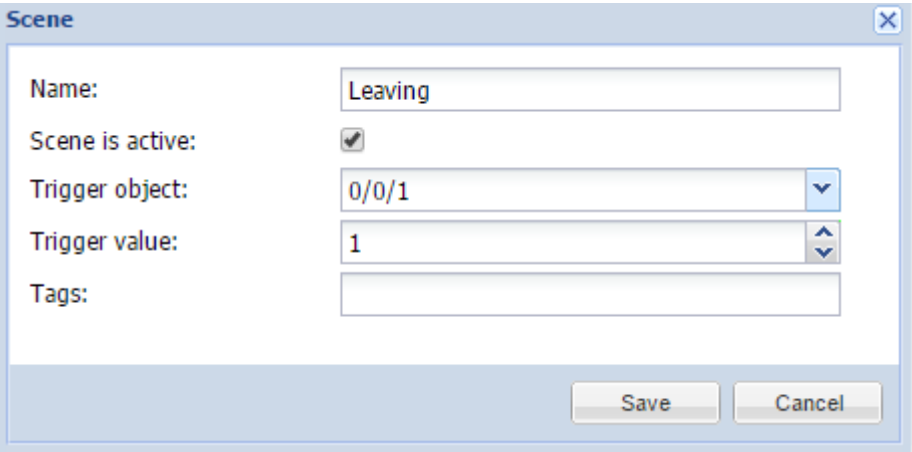
Пример:

```
require('trends')

-- использует NaN ("не число") для недопустимых значений
trends.NaN = 0 / 0

-- получит данные за прошлый год
dates = {}
dates['start'] = os.date('*t')
dates['start'].year = dates['start'].year - 1
dates['end'] = os.date('*t')

value = trends.fetchone('Hot Water', dates)

-- NaN ~= NaN означает, что значение не найдено
if value ~= trends.NaN then
  return
end
```

6.6. Вкладка Scenes

Вкладка *Scenes* позволяет создавать скрипты визуально без использования событийных скриптов.

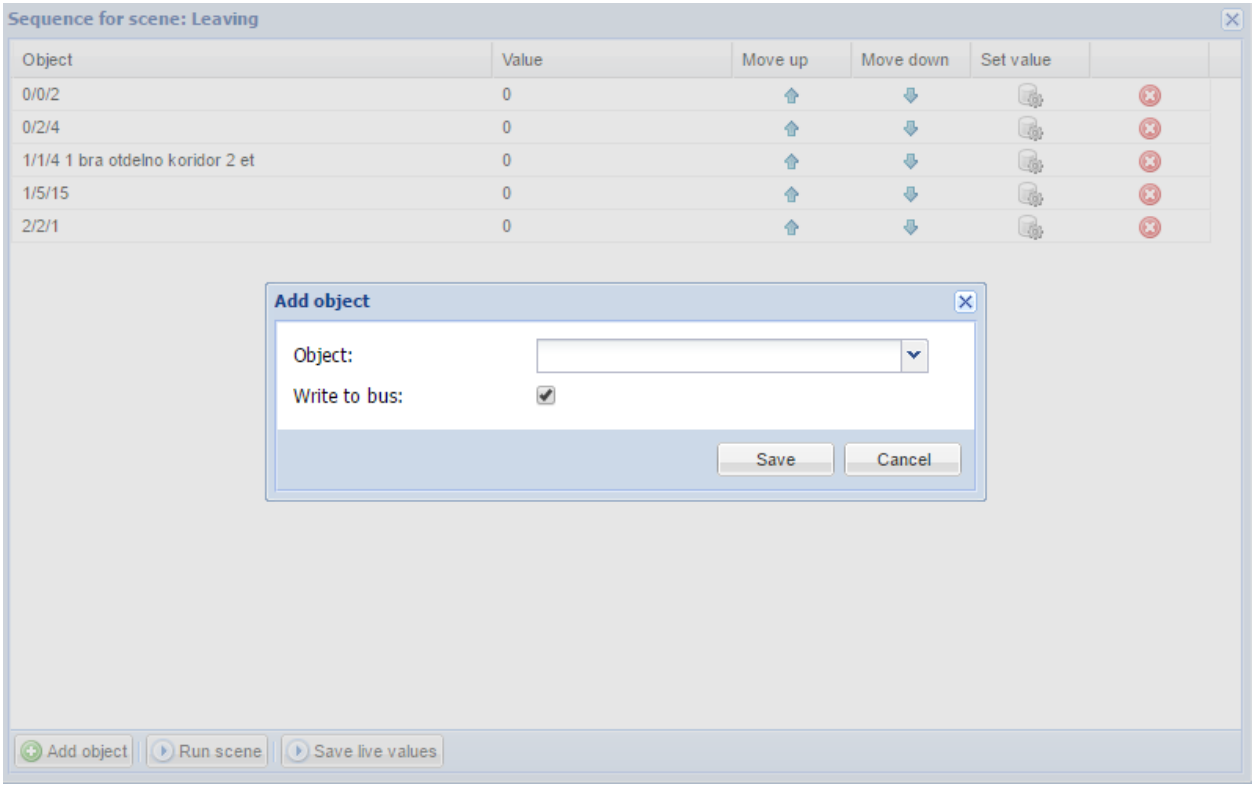
6.6.1. Добавление сцены



- ***Name*** – название сцены;
- ***Scene is active*** – определяет, включена сцена или выключена;
- ***Trigger object*** – групповой адрес для вызова сцены;
- ***Trigger value*** – значение объекта для вызова сцены.

6.6.2. Последовательность сцен

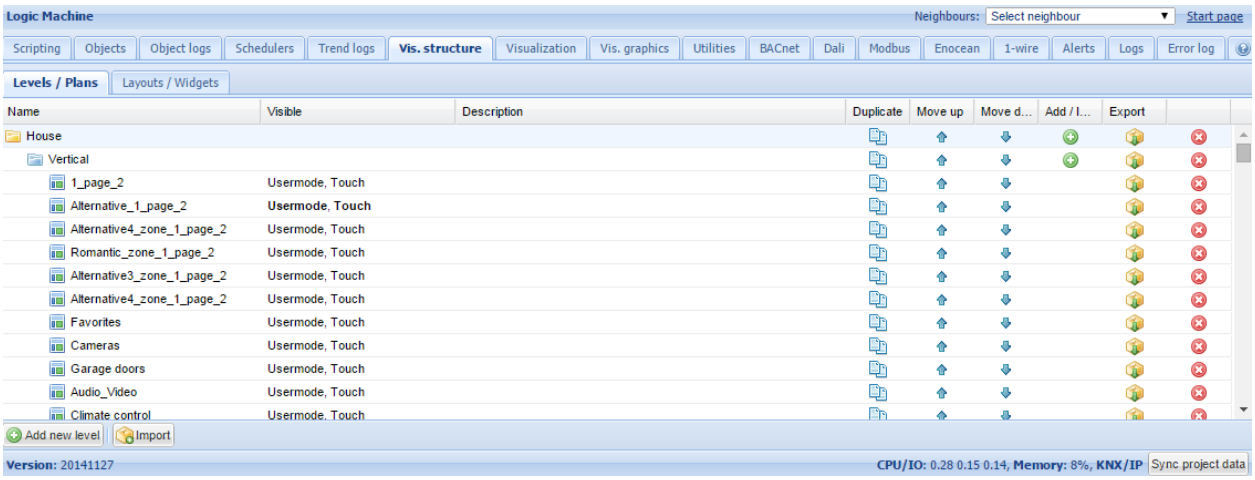
Список объектов и их последовательность задаются здесь.



- ***Object*** – групповой адрес объекта;
- ***Write to bus*** – определяет, посылать или не посылать значение в шину KNX.

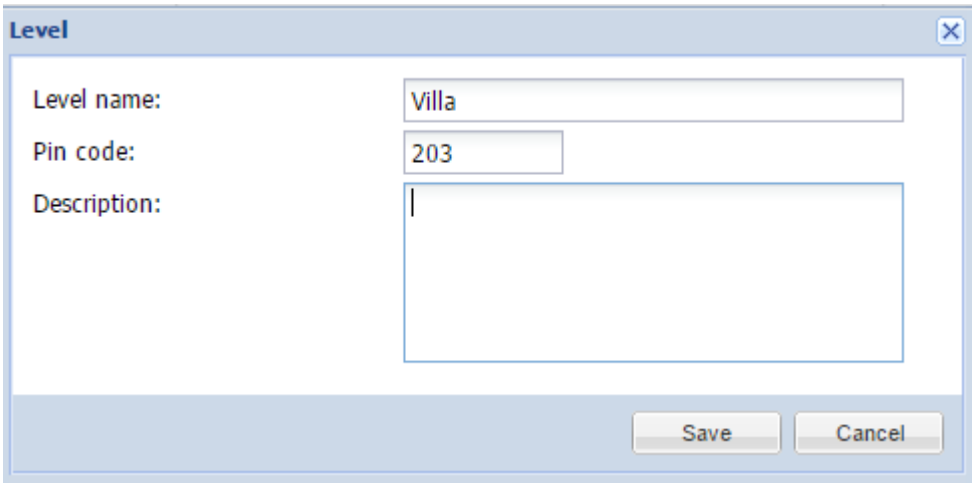
6.7. Вкладка Vis. structure

Во вкладке *Vis.structure* задаётся структура визуализации и настраиваются фоновые изображения или цвет фона для каждого плана.

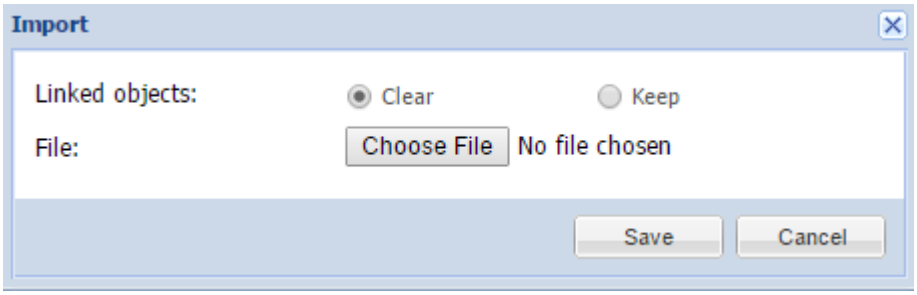


6.7.1. Levels / Plans (Уровни / Планы)

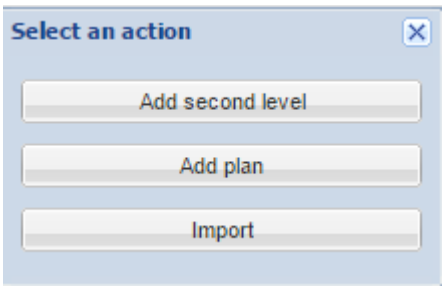
По умолчанию добавлен уровень Main. Чтобы добавить новый уровень/здание, нажмите кнопку *Add new level*. Обратите внимание, что вы можете ограничить доступ к уровню, добавив PIN-код.



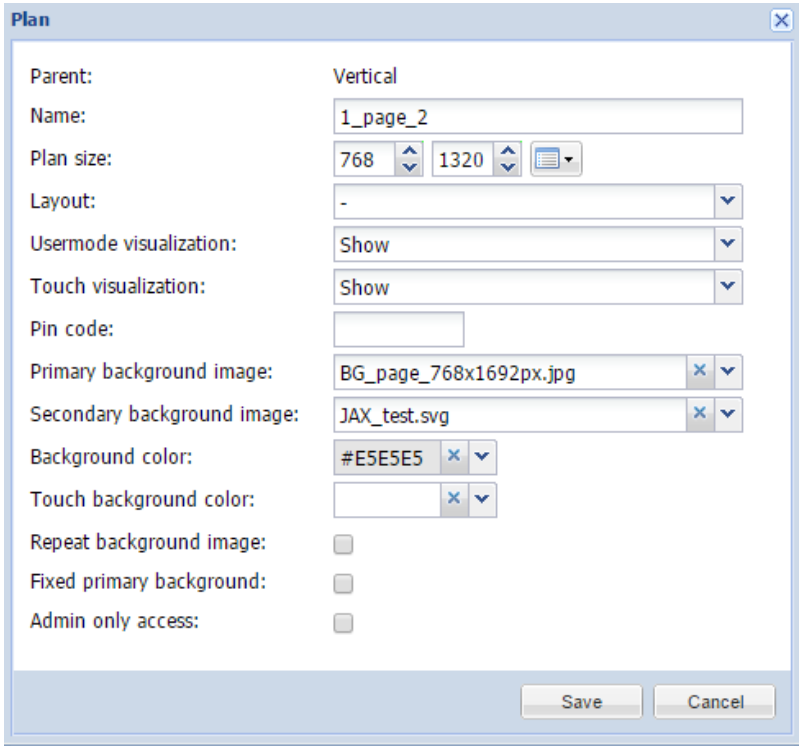
Вы также можете добавить новый уровень, импортировав его из файла (например, экспортированного из другого LM). Для этого нажмите кнопку *Import*. Связи объектов могут быть очищены или импортированы как есть.



После добавления нового уровня вы можете добавить второй уровень или создать план этажа, относящегося к данному зданию. Чтобы добавить новую запись, нажмите зелёную иконку, чтобы удалить запись, нажмите красную иконку.



При добавлении нового плана нужно указать следующие параметры:



- ***Parent*** – название уровня, на котором создается план;
- ***Name*** – название плана;
- ***Plan size*** – размер плана в пикселях. Предустановленные разрешения доступны при нажатии на значок правее этого параметра:

iPad landscape, fullscreen (XGA) 1024 x 748

iPad landscape, browser (XGA) 1024 x 672

iPad portrait, fullscreen (XGA) 768 x 1004

iPad portrait, browser (XGA) 768 x 928

Tablet landscape (WSVGA) 1024 x 600

Tablet portrait (WSVGA) 600 x 1024

Laptop / Tablet landscape (WXGA) 1280 x 800

Laptop / Tablet portrait (WXGA) 800 x 1280

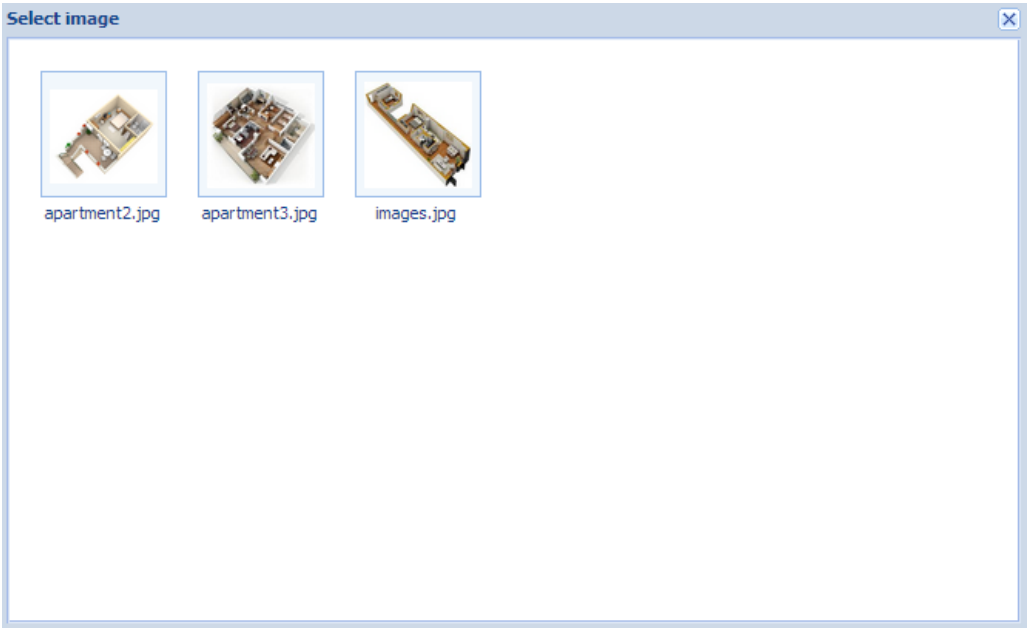
Laptop / Tablet landscape (HD) 1360 x 768

Laptop / Tablet portrait (HD) 768 x 1360

Big screen (Full HD) 1920 x 1080

- ***Layout*** – подложка для этого плана. Все подложки, включая цвет фона и изображение плана, будут дублироваться, если они не определены отдельно для этого плана;
- ***Usermode visualization [Show, Show and make default, Hide]*** – видимость плана на визуализации в режиме пользователя;
- ***Touch visualization [Show, Show and make default, Hide]*** – видимость плана на Touch-визуализации;
- ***PIN code*** – пин-код для доступа к плану;
- ***Primary background image*** – основное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Secondary background image*** – второстепенное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Background color*** – цвет фона;
- ***Touch background color*** – цвет фона на Touch-визуализации;
- ***Repeat background image*** – при включении заполняет весь план копиями изображения;
- ***Fixed primary background*** – при включении фиксирует первое фоновое изображение. Когда эта опция включена, в визуализации можно использовать эффект параллакса;
- ***Admin only access*** – разрешить доступ к этому плану только администратору.

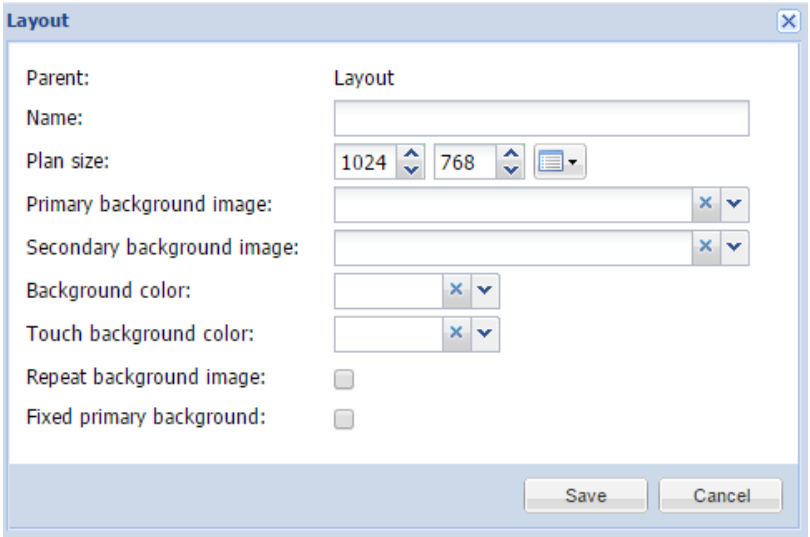
При выборе фонового изображения появится окно с изображениями, находящимися в *Vis.graphics → Images/Background*:



Вы можете продублировать план со всеми его объектами и настройками, нажав на иконку. Уровни могут быть отсортированы нажатием на иконки и . Структуру плана можно экспортировать нажатием на иконку .

6.7.2. Layouts / Widgets (Подложки / Виджеты)

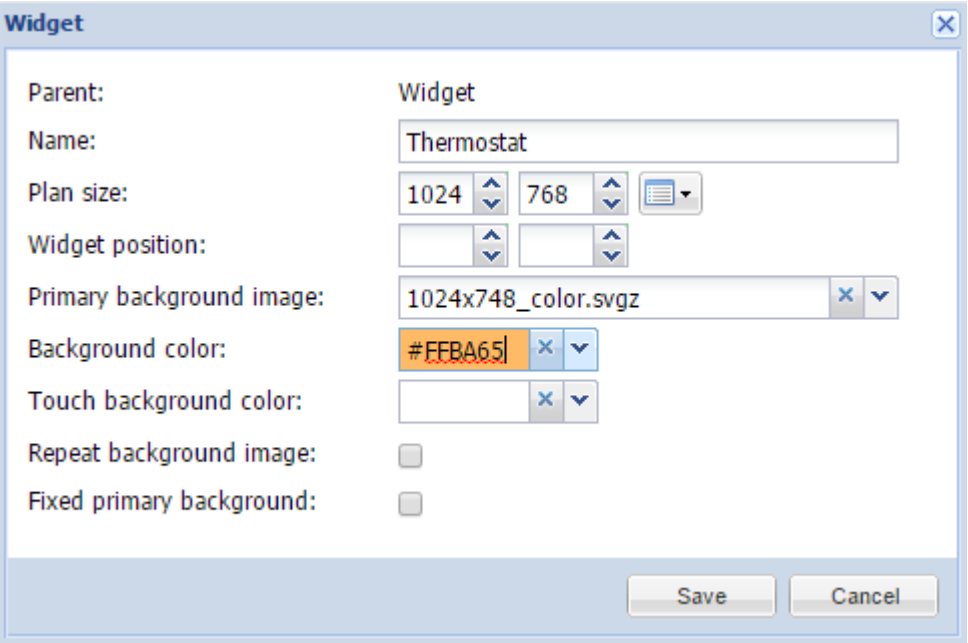
Подложки используются в качестве шаблонов для дальнейшего использования при добавлении уровней на вкладке *Levels / Plans*. Подложки не видны из визуализаций *Usermode* и *Touch*. Если вы добавите на подложку фон или объект, он автоматически появится на всех связанных уровнях.



➢ ***Parent*** – название уровня, на котором создаётся подложка;

- ***Name*** – название подложки;
- ***Plan size*** – размер подложки в пикселях. Предустановленные разрешения доступны при нажатии на значок правее этого параметра;
- ***Primary background image*** – основное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Secondary background image*** – второстепенное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Background color*** – цвет фона;
- ***Touch background color*** – цвет фона на Touch-визуализации;
- ***Repeat background image*** – при включении заполняет весь план копиями изображения;
- ***Fixed primary background*** – при включении фиксирует первое фоновое изображение. Когда эта опция включена, в визуализации можно использовать эффект параллакса.

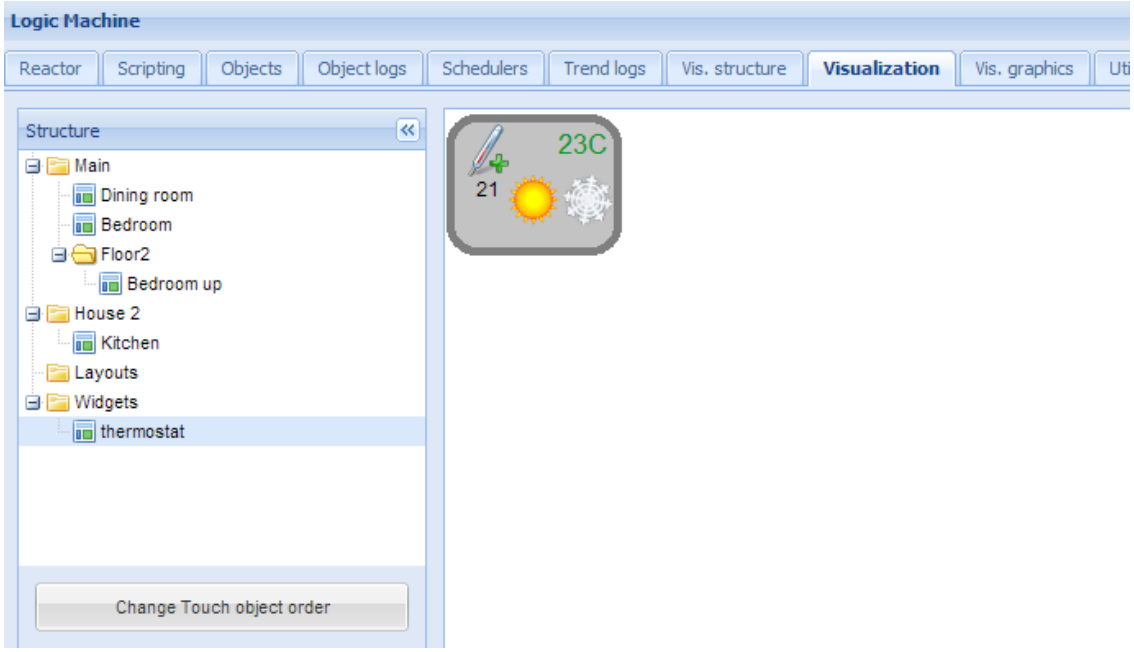
Виджеты используются для объединения нескольких объектов в один.
Фоновое изображение для виджета должно быть предварительно добавлено в *Vis.graphics → Images/Background*.



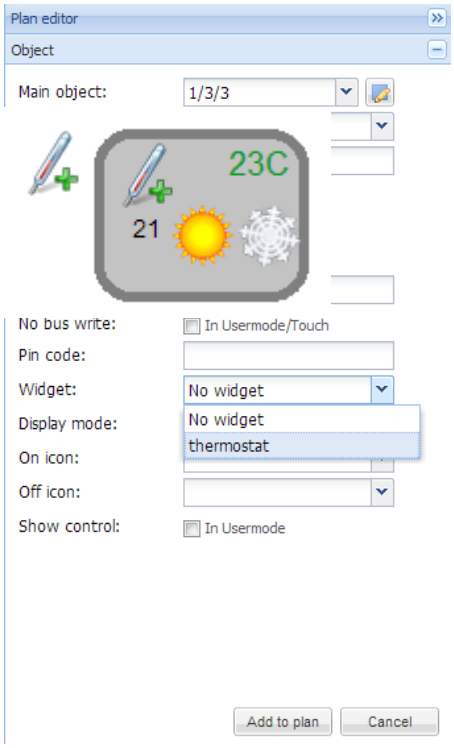
- ***Parent*** – название уровня, на котором создается виджет;
- ***Name*** – название виджета;
- ***Plan size*** – размер виджета в пикселях. Предустановленные разрешения доступны при нажатии на значок правее этого параметра;
- ***Widget position*** – положение виджета на экране по умолчанию;
- ***Primary background image*** – основное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Secondary background image*** – второстепенное фоновое изображение, загруженное в *Vis.graphics → Images/Backgrounds;*
- ***Background color*** – цвет фона;

- ***Touch background color*** – цвет фона на Touch-визуализации;
- ***Repeat background image*** – при включении заполняет весь план копиями изображения;
- ***Fixed primary background*** – при включении фиксирует первое фоновое изображение. Когда эта опция включена, в визуализации можно использовать эффект параллакса.

Когда вы создадите виджет на вкладке *Layouts/Widget*, вы сможете добавить на него объекты на вкладке *Visualization*.



После добавления в виджет необходимых элементов вы можете выбрать его при добавлении объектов на уровни, например, в спальню (*Bedroom*) на уровне Main.

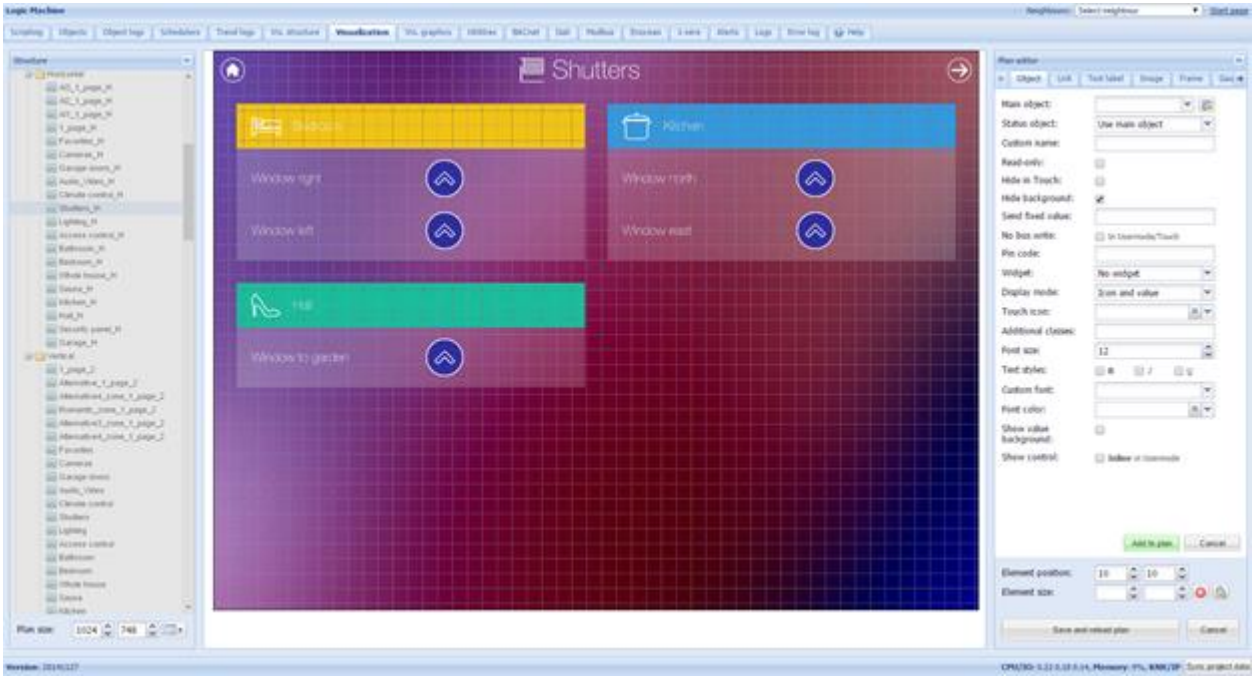


После добавления вы можете протестировать виджет в Usermode-визуализации: виджет появится при нажатии на добавленный объект (значок датчика температуры слева).

6.8. Вкладка Visualization

После того, как структура зданий и этажей смоделирована на вкладке Vis.structure, она отображается на вкладке Visualization. В этом разделе можно добавлять и удалять объекты.

Обе боковые панели можно скрыть, нажав на иконку. Это расширит основную рабочую зону, что может быть полезно на устройствах с маленьким дисплеем.



6.8.1. Редактор плана

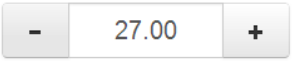
Редактор плана расположен в правой части экрана визуализации. По нажатию кнопки «Разблокировать текущий план для редактирования» на нём появляются следующие вкладки:

- ***Object*** – добавление объекта на план;
- ***Link*** – добавление перехода между планами;
- ***Text Label*** – добавление текста;
- ***Image*** – добавление изображения;
- ***Frame*** – добавление frame-объекта;
- ***Gauge*** – добавление объекта в виде стрелочного указателя;
- ***Camera*** – добавление IP-камеры;
- ***Graph*** – добавление графика в режиме реального времени для мониторинга значений объектов.

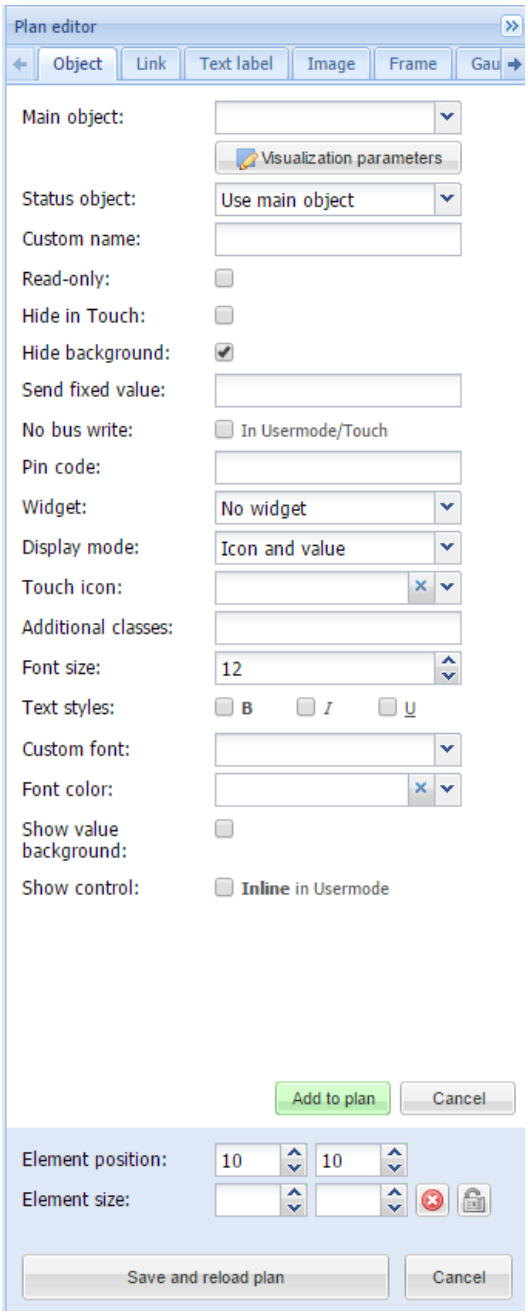
Если план открыт для редактирования, то с помощью счётчиков в левом нижнем углу окна можно изменить его размеры.

Если в режиме редактирования выбран объект, для него появляются кнопки Delete / Duplicate, позволяющие удалить или продублировать выбранный элемент.

6.8.2. Вкладка Object

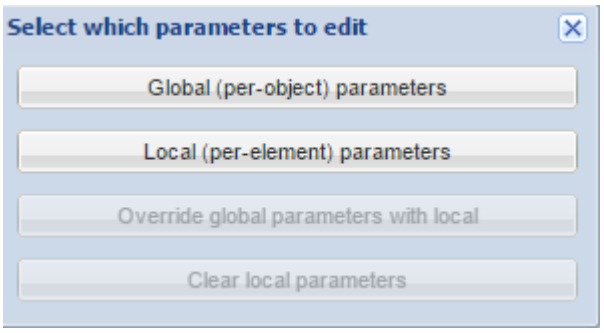
- ***Main object*** – список существующих и доступных для настройки групповых адресов на шине KNX/EIB;
- ***Status object*** – список объектов, значения которых могут отображаться на визуализации вместо основного объекта;
- ***Custom name*** – пользовательское имя объекта;
- ***Read-only*** – включение и выключение режима “только для чтения”;
- ***Hide in touch*** – не показывать в Touch-визуализации;
- ***Hide background*** – не показывать фон иконки;
- ***Send fixed value*** – позволяет задать конкретное значение, которое будет отправляться при нажатии на объект;
- ***No bus write*** – не посылать значение в шину при нажатии на объект;
- ***PIN code*** – пин-код для доступа к управлению объектом;
- ***Widget*** – виджет, который будет появляться при нажатии на объект;
- ***Display mode [icon and value; icon; value]*** – режим отображения объекта: иконка и значение, только иконка или только значение;
- ***Touch icon*** – иконка для Touch-визуализации;
- ***On icon*** – иконка состояния “Вкл” для бинарных объектов. Библиотека иконок расположена в *Vis.graphics → Icons;*
- ***Off icon*** – иконка состояния “Выкл” для бинарных объектов. Библиотека иконок расположена в *Vis.graphics → Icons;*
- ***Additional classes*** – дополнительные CSS-классы для объекта;
- ***Show control*** – опция доступна только для числовых объектов и определяет, будет ли в Usermode-визуализации отображаться элемент управления; 

- ***Show value background*** – цвет фона для значения.



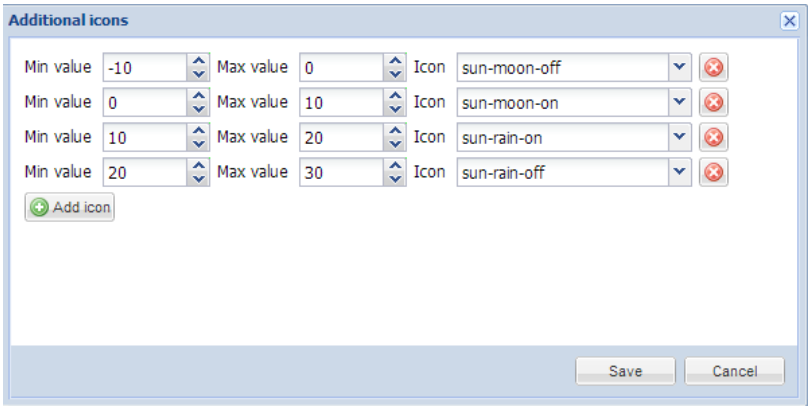
➢ **Visualization parameters (параметры визуализации)**

Кнопка *Visualization parameters* позволяет настроить глобальные и локальные параметры визуализации для определённого группового адреса.

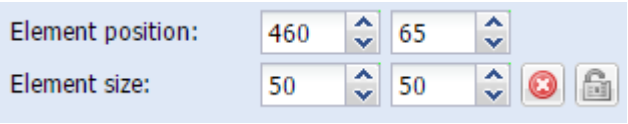


Global (per-object) parameters – задать глобальные параметры визуализации для элемента группового адреса (см. пункт 6.2.3). Глобальные параметры используются на всех элементах этого группового адреса;
Local (per-element) parameters – задать локальные параметры визуализации для элемента группового адреса (см. пункт 6.2.3). Локальные параметры используются только в одном конкретном месте;
Override global parameters with local – заменить глобальные параметры элемента на локальные.
Clear local parameters – удалить все локальные параметры

Для числовых объектов появляется дополнительная кнопка – *Additional icons.* Она открывает окно, в котором можно установить разные иконки для разных значений объекта.



Внизу находятся настройки расположения и размера элемента, которые можно свободно изменять. Нажатие на  сбрасывает размер элемента. Нажатие на  фиксирует соотношение его длины и ширины.

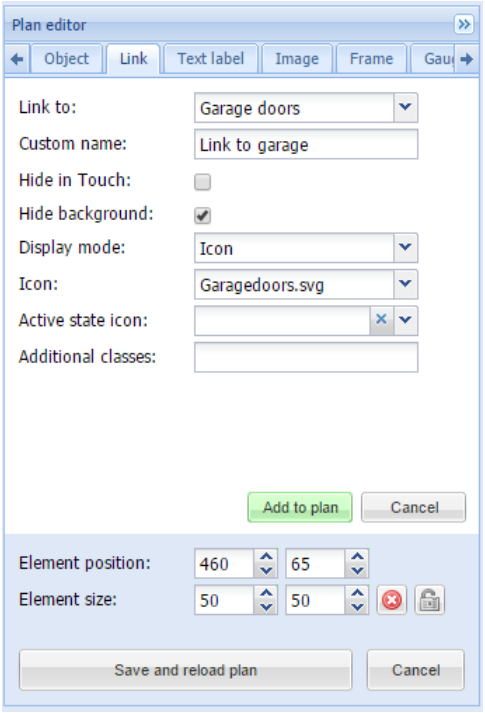


После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. Обратите внимание: в режиме редактирования объекты не будут работать. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

Вы можете изменить любой добавленный объект, нажав на него в режиме редактирования.

6.8.3. Вкладка Link

Вкладка Link управляет встроенными переходами на другие планы, которые предназначены для удобства работы с визуализацией. С их помощью на визуализацию можно добавить иконки или текст, которые перемещают вас на другой план.

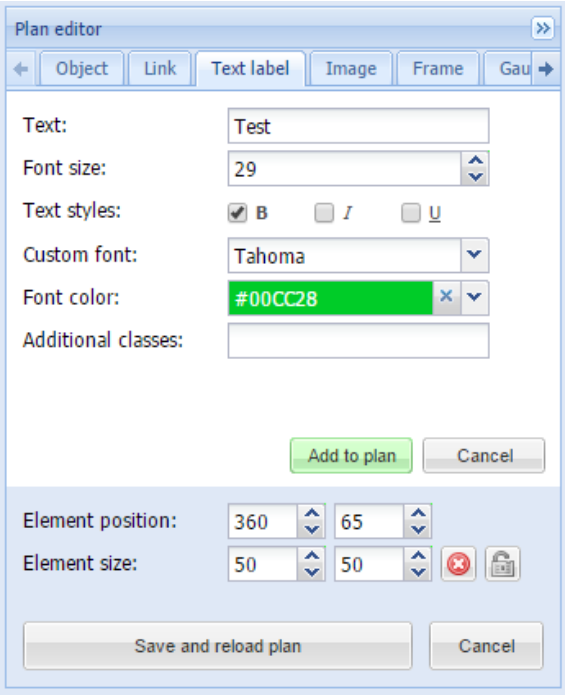


- ***Link to*** – название плана, на который будет осуществлён переход, ссылка на трендлог/планировщик или внешняя ссылка (вида http://www.openrb.com);
- ***Custom name*** – пользовательское название перехода;
- ***Hide in touch*** – не показывать в Touch-визуализации;
- ***Hide background*** – не показывать фон иконки;
- ***Display mode [Icon; Value]*** – режим отображения перехода: иконка или название;
- ***Icon*** – иконка, которая будет отображаться на визуализации (если выбрано, дальнейшие параметры отсутствуют);
- ***Active state icon*** – иконка, которая отображается, если переход ведёт на текущий план (в случае если у вас есть несколько планов одной визуализации и вы хотите отобразить текущий);
- ***Additional classes*** – дополнительные CSS-классы для объекта.

После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. Обратите внимание: в режиме редактирования объекты не будут работать. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

6.8.4. Вкладка Text label

На этой вкладке вы можете добавить на визуализацию текстовую надпись.

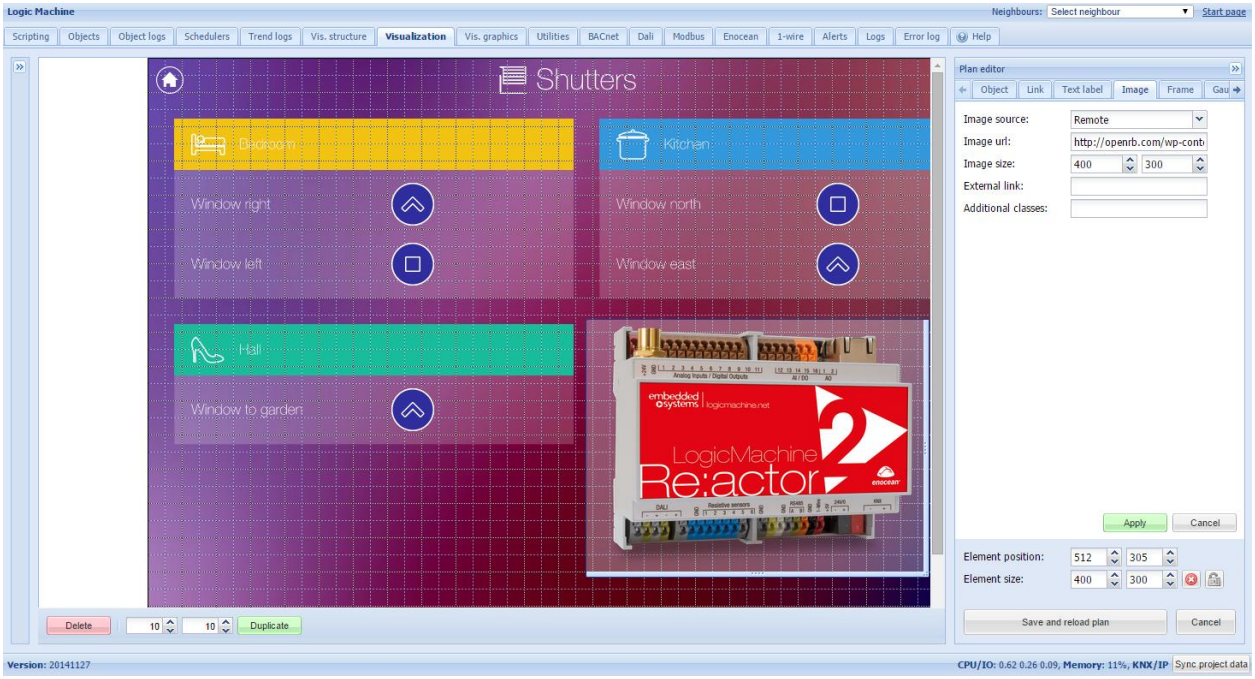


- ***Text*** – текст надписи;
- ***Font size*** – размер шрифта;
- ***Text style*** – стиль текста: полужирный, курсив, подчёркнутый;
- ***Custom font*** – название шрифта;
- ***Font color*** – цвет шрифта;
- ***Additional classes*** – дополнительные CSS-классы для объекта.

После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

6.8.5. Вкладка Image

Вкладка *Image* позволяет добавить на план картинку из библиотеки или из интернета. К примеру, можно загрузить изменяющееся изображение с прогнозом погоды.



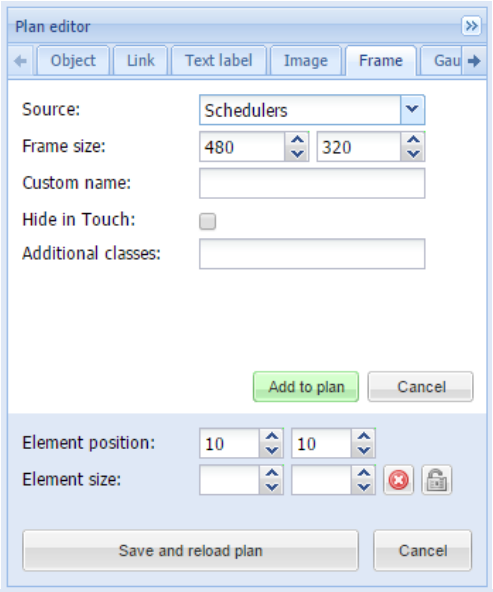
- ***Image source [Local; Remote]*** – расположение изображения (локальный или внешний источник);
- ***Source url / Select image*** – ссылка на изображение в интернете (если указан внешний источник) или выбор изображения из библиотеки (если указан локальный);
- ***Image size*** – размер изображения;
- ***External link*** – ссылка, по которой вы перейдёте при нажатии на изображение;
- ***Additional classes*** – дополнительные CSS-классы для объекта.

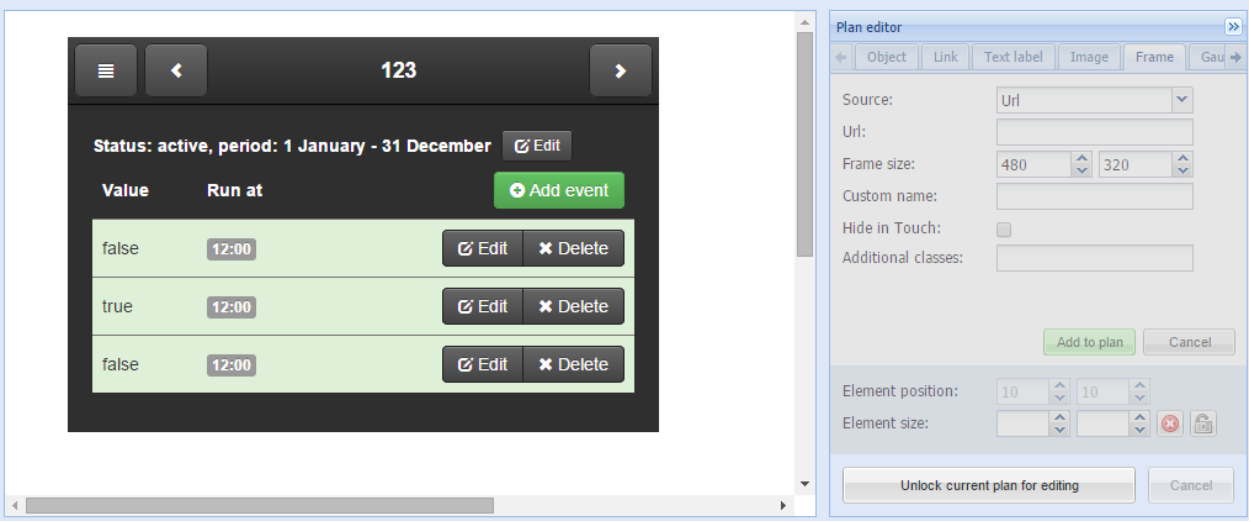
После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

6.8.6. Вкладка Frame

Фреймы позволяют добавлять на визуализацию локальные трендлоги/планировщики и приложения со сторонних ресурсов.

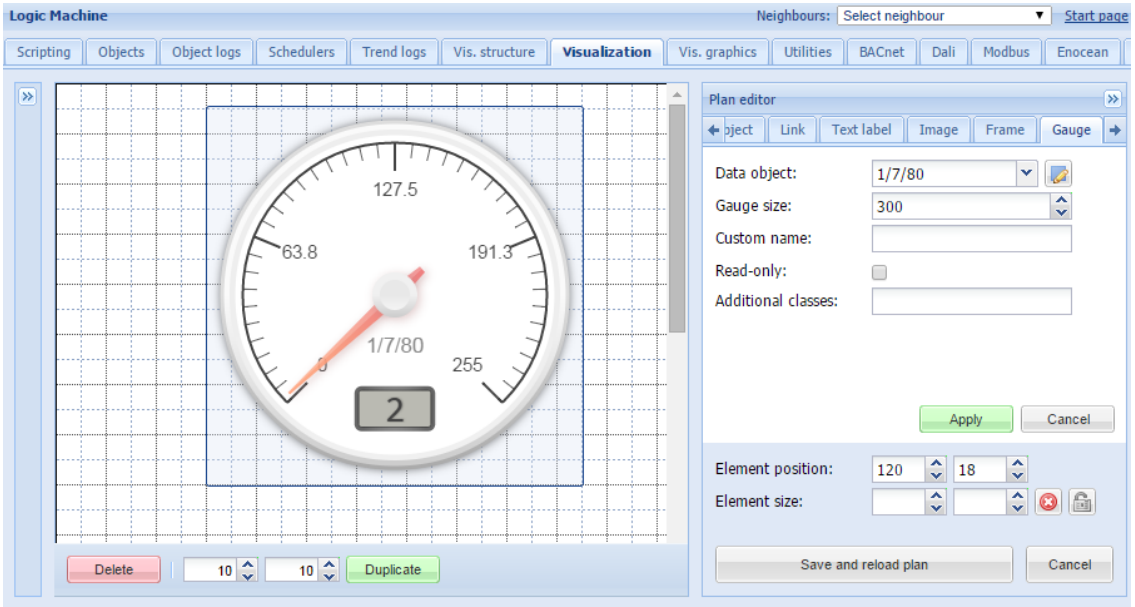
- ***Source [Url, Schedulers; Trend logs]*** – источник фрейма (URL, планировщик или трендлог);
- ***Url*** – ссылка на источник фрейма (если в предыдущем пункте выбран URL);
- ***Frame size*** – размер фрейма;
- ***Custom name*** – пользовательское имя фрейма;
- ***External link*** – ссылка, по которой вы перейдёте при нажатии на фрейм;
- ***Hide in Touch*** – не показывать в Touch-визуализации;
- ***Additional classes*** – дополнительные CSS-классы для объекта.





6.8.7. Вкладка Gauge

Вкладка *Gauge* позволяет настроить индикатор, представляющий значение объекта в виде стрелочного указателя.

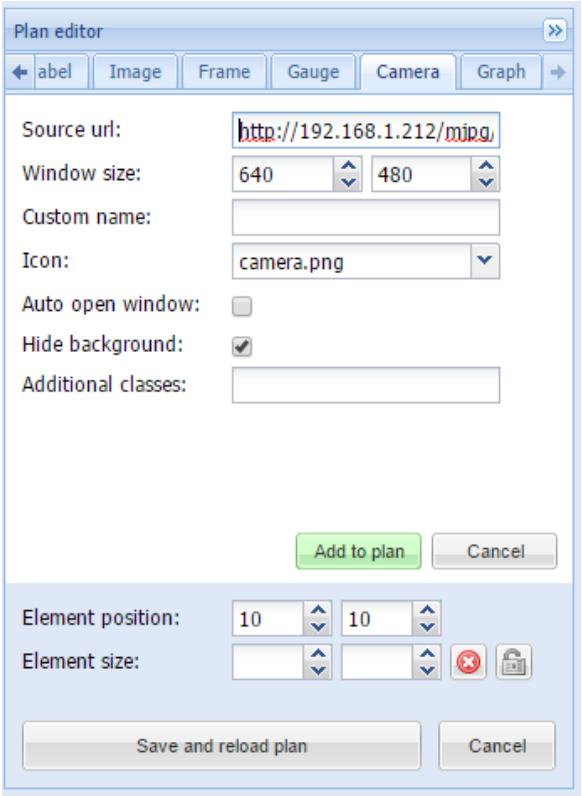


- ***Data object*** – групповой адрес KNX;
- ***Gauge size*** – размер стрелочного указателя;
- ***Custom name*** – пользовательское имя объекта;
- ***Read only*** – включение и выключение режима “только для чтения”;
- ***Additional classes*** – дополнительные CSS-классы для объекта.

После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

6.8.8. Вкладка Camera

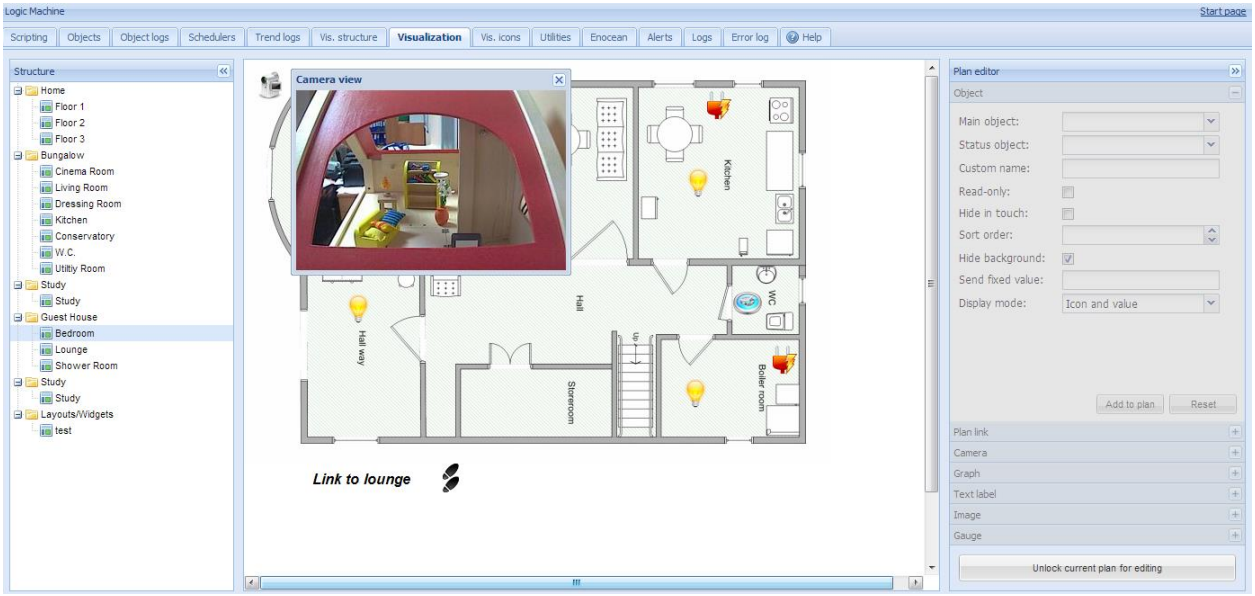
LogicMachine поддерживает интегрирование в визуализацию сторонних IP-камер. Они настраиваются через вкладку *Camera.*



- ***Source url*** – ссылка на видеострим;
- ***Window size*** – размер окна;
- ***Custom name*** – пользовательское имя объекта;
- ***Icon*** – иконка объекта;
- ***Auto open window*** – автоматически открывать окно с видео. Если эта опция отключена, окно будет открываться при нажатии на иконку;
- ***Hide background*** – спрятать фон иконки;
- ***Additional classes*** – дополнительные CSS-классы для объекта.

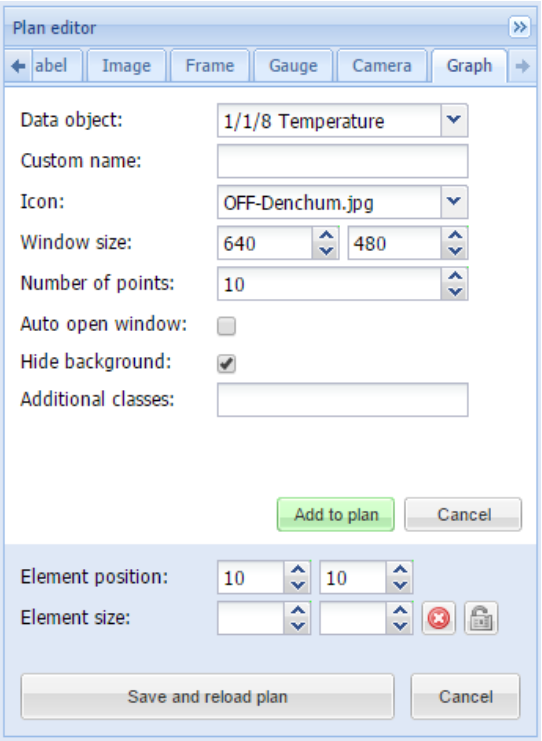
Примечание! Если камера требует авторизации, введите ссылку в форме
http://USER:PASSWORD@IP

После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. Обратите внимание: в режиме редактирования объекты не будут работать. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов. Нажатие на камеру откроет окно с изображением с этой камеры. Это окно можно свободно перемещать по плану, чтобы оно не перекрывало другие объекты.



6.8.9. Вкладка Graph

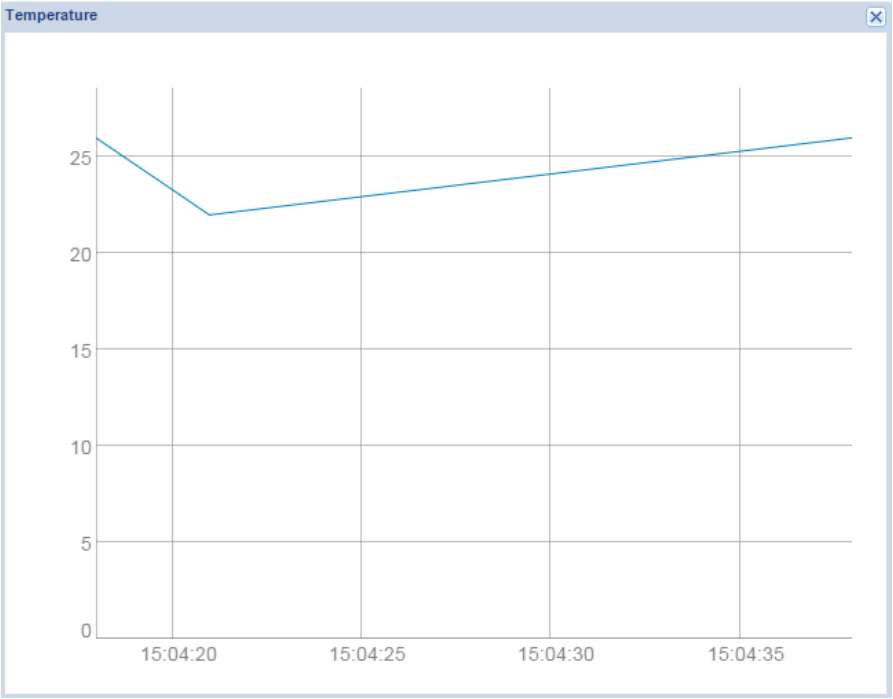
Графики в режиме реального времени используются в визуализации для отображения текущего и прошлых значений числовых объектов. Убедитесь, что на вкладке *Objects* для этих объектов включено логирование.



- ***Data object*** – групповой адрес объекта;
- ***Custom name*** – пользовательское имя объекта;
- ***Icon*** – иконка, открывающая график;
- ***Windows size*** – размер окна графика;
- ***Number of points*** – количество точек, отображаемых на графике;

- ***Auto open window*** – автоматическое открытие окна с графиком. Если эта опция отключена, окно будет открываться при нажатии на иконку;
- ***Hide background*** – скрыть фон иконки;
- ***Additional classes*** – дополнительные CSS-классы объекта.

После задания параметров нажмите на кнопку *Add to plan* для добавления нового объекта на план. Если вы неверно задали расположение объекта, можете его передвинуть. После добавления всех необходимых объектов нажмите на кнопку *Save and reload plan,* чтобы сохранить план и запустить работу объектов.

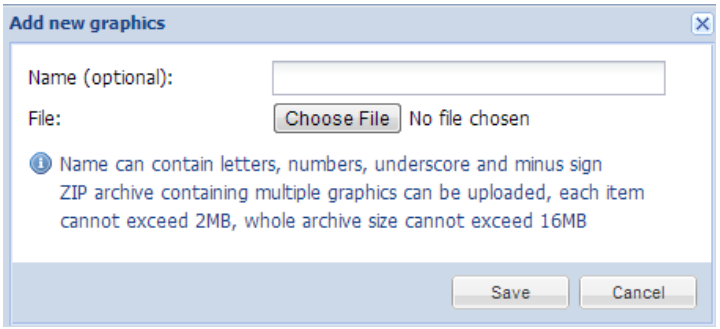


6.9. Вкладка Vis. graphics

На вкладке *Vis. Graphics* отображаются предустановленные иконки, изображения и фоны.

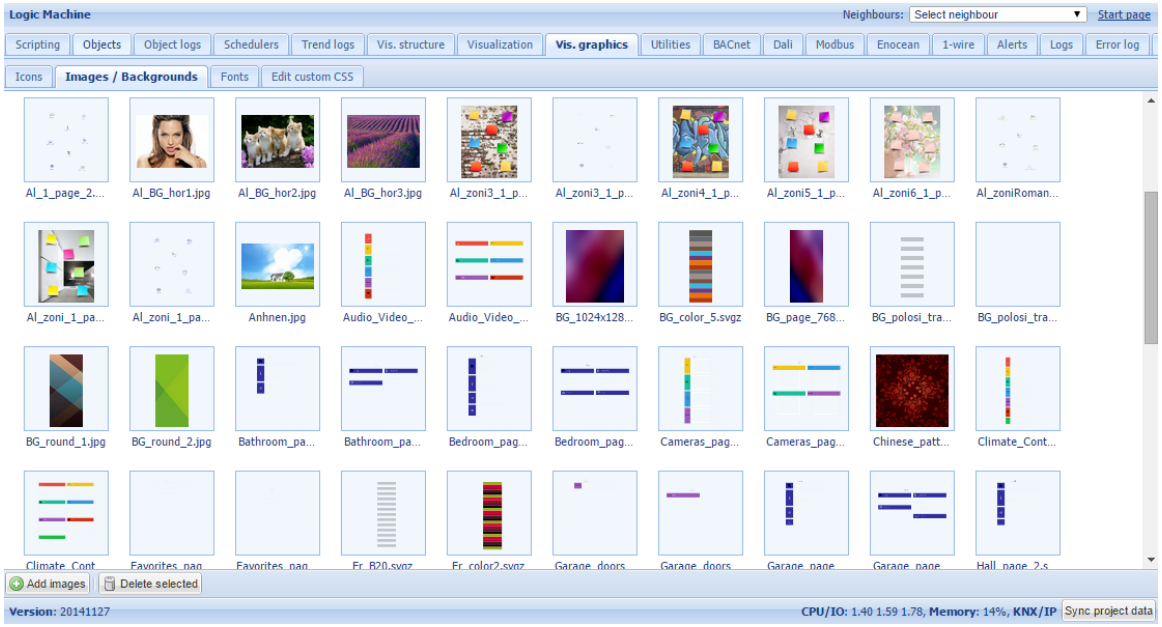
Иконки расположены на вкладке Icons.

Чтобы добавить новую иконку, нажмите на кнопку *Add icons*. LogicMachine поддерживает иконки любых размеров. Также поддерживается формат GIF и векторная графика в формате SVG.

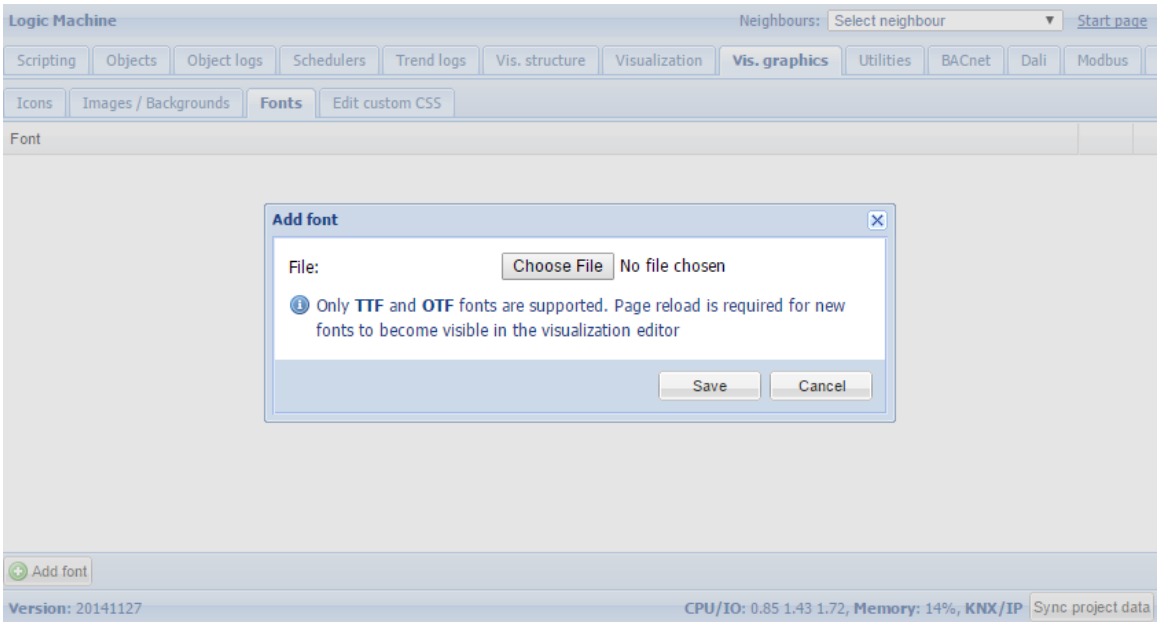


- ***Name (optional)*** – название иконки;
- ***File*** – расположение файла с иконкой.

Вкладка *Images / Backgrounds* предназначена для загрузки и хранения изображений и фонов.

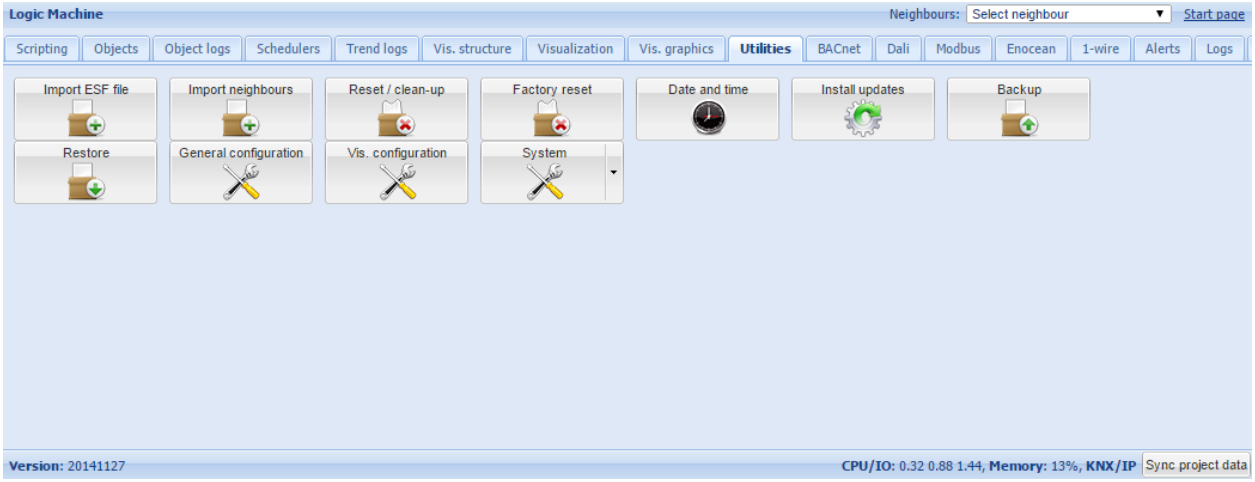


Вкладка *Fonts* позволяет добавить дополнительные шрифты форматов TTF и OTF.



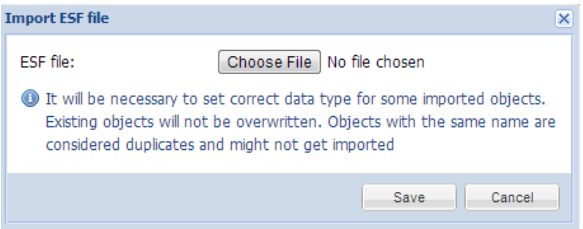
На вкладке *Edit custom CSS* можно добавить дополнительные CSS-стили, которые впоследствии можно будет использовать при отображении объектов на визуализации.

6.10. Вкладка Utilities

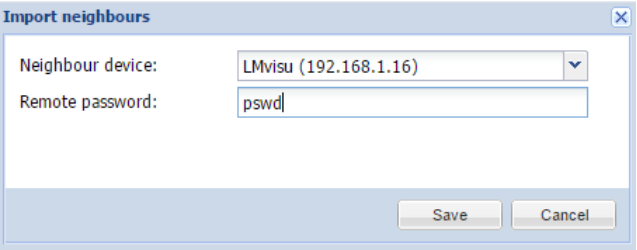


На этой вкладке доступны следующие утилиты:

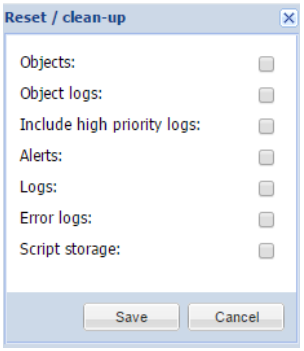
Import ESF file – импорт файла с объектами из ETS. Для некоторых импортированных объектов необходимо будет установить правильные типы данных. Существующие объекты не будут перезаписаны. Объекты с такими же именами, как и у существующих, считаются дублирующимися и не будут импортированы;



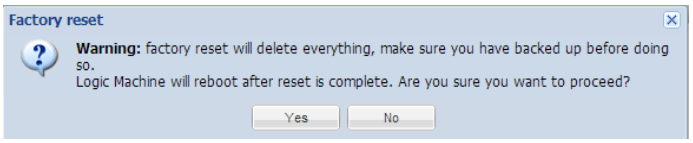
Import neighbours – импорт списка объектов с других LogicMachine в сети;



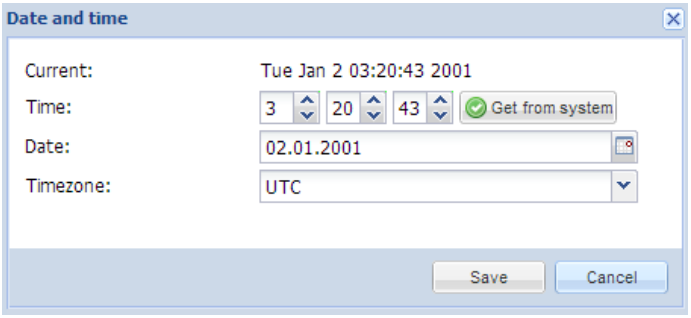
Reset / clean-up – удаление всех объектов из LogicMachine (при этом из визуализации они также удаляются);



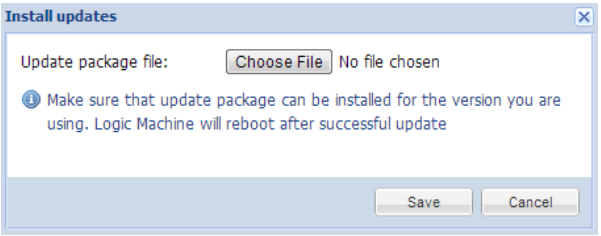
Factory reset – сброс до заводских настроек;



Date and time – настройки даты и времени;

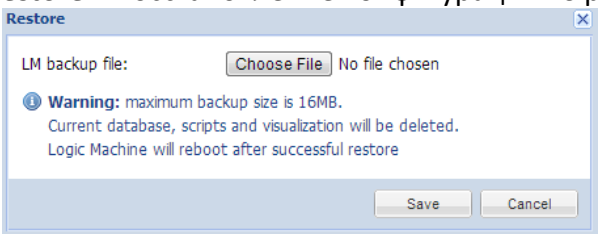


Install updates – установка обновлений LogicMachine из файла *.lmu. После успешного обновления LogicMachine перезагрузится;

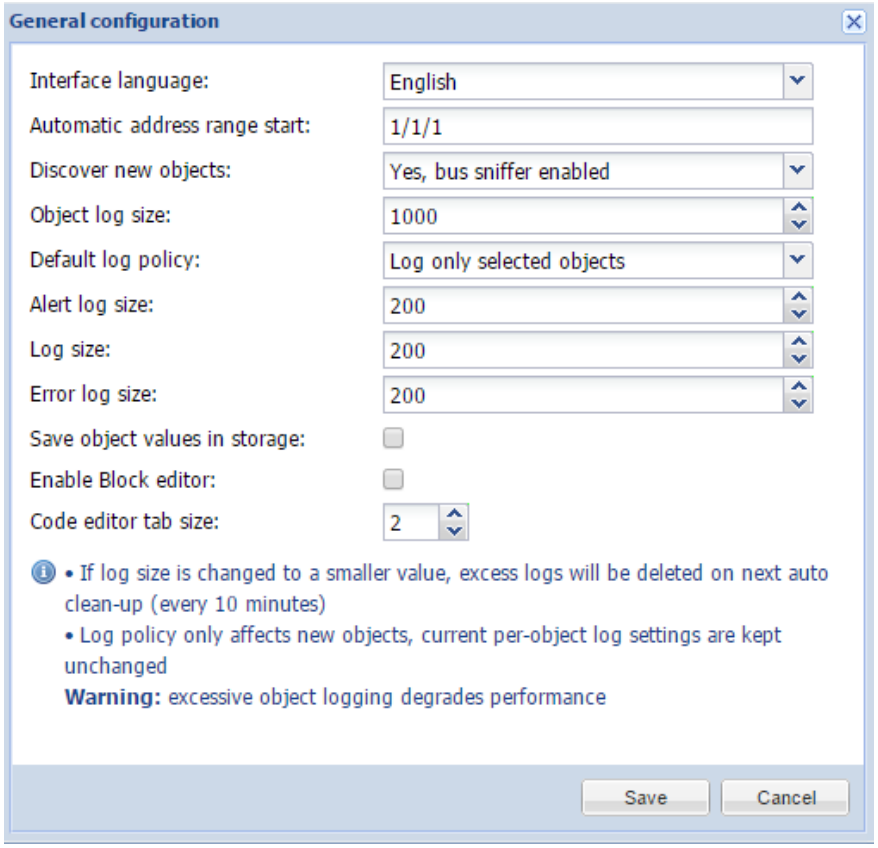


Backup – создание резервной копии всех объектов, логов, скриптов и визуализации;

Restore – восстановление конфигурации из резервной копии;



General Configuration – основные настройки системы:



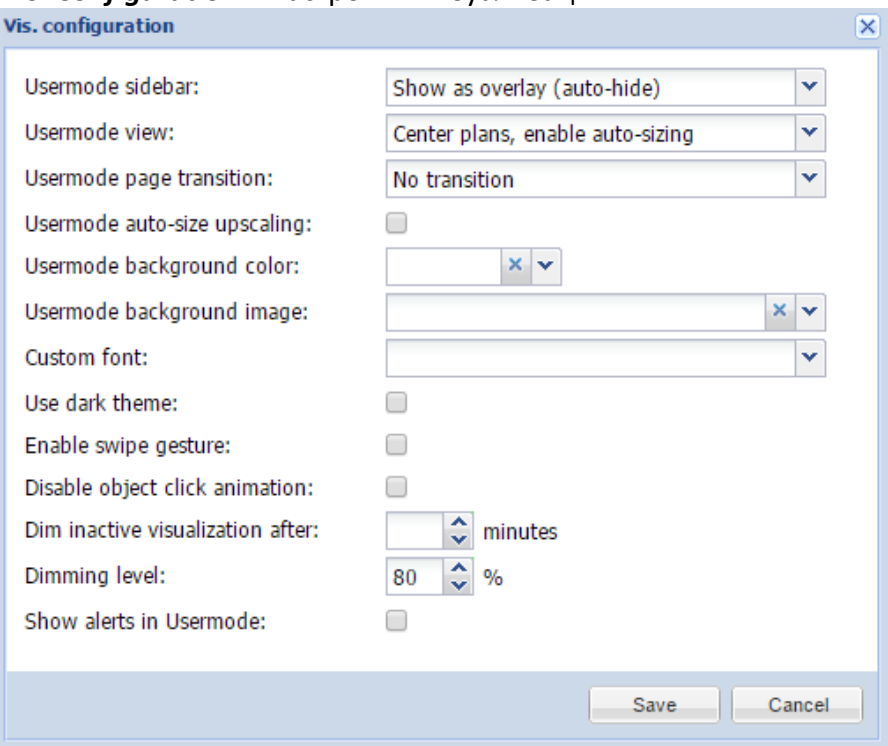
- ***Interface language*** – язык интерфейса;
- ***Automatic address range start*** – начальный групповой адрес для автоматического назначения групповых адресов;
- ***Discover new objects*** – автоматическое добавление групповых адресов при получении значения на шине;
- ***Object log size*** – максимальное количество логов объекта;
- ***Default log policy*** – настройки логов: включение логгирования для всех объектов или только для отмеченных;
- ***Alert log size*** – максимальное количество тревог;
- ***Log size*** – максимальное количество логов;
- ***Error log size*** – максимальное количество ошибок;
- ***Save object values in storage*** – сохранение значений объектов в базе данных для доступа из приложений;
- ***Code editor tab size*** – размер табуляции, который будет использоваться в редакторе скриптов.

Примечание! Если размер логов изменён на меньшее значение, то лишние логи будут удалены при следующей автоматической чистке (проводится каждые 10 минут).

Примечание! Изменение настроек логов действует только на новые объекты. Настройки логов для текущих объектов остаются без изменений.

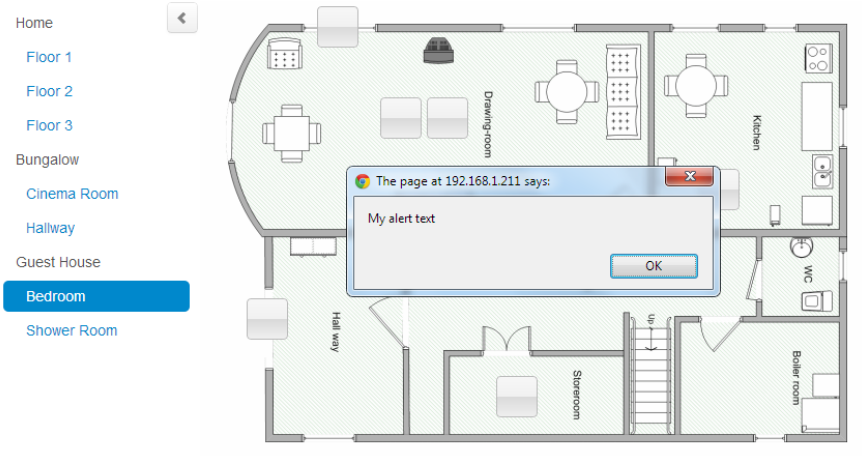
Внимание! Чрезмерное логгирование объектов ухудшает производительность LogicMachine. Рекомендуется сохранять логи на внутреннем FTP-сервере или настроить автоматический экспорт на внешний FTP-сервер. Как это сделать, описано в инструкции ниже:
http://openrb.com/example-export-last-hour-csv-object-log-file-to-external-ftp-server-from-lm2/

Vis. Configuration – настройки визуализации:

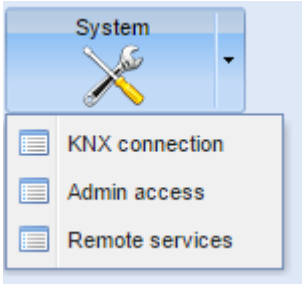


- ***Usermode sidebar [Show docked, Show as overlay (auto-hide), Hide (fullscreen mode]*** – определяет видимость боковой панели в Usermode-визуализации: видна и закреплена, видна и автоматически скрывается, не видна;
- ***Usermode view [Align plans to top left, no size limit; Center plans, limit size; Center plans, enable auto-sizing; Center horizontally, auto-size width]*** – определяет внешний вид Usermode-визуализации;
- ***Usermode page transition [Flip X; Flip Y; Shrink; Expand; Slide up; Slide down, Slide left; Slide right; Slide up big; Slide down big; Slide left big; Slide right big]*** – определяет внешний вид перехода между планами;
- ***Usermode auto-size upscaling*** – включает и отключает автоматическое масштабирование. Если эта опция включена, рекомендуется использовать иконки и изображения в формате SVG, чтобы избежать потерь качества при масштабировании;
- ***Usermode background color*** – цвет фона в Usermode-визуализации;
- ***Usermode background image*** – фоновое изображение для Usermode-визуализации;
- ***Custom font*** – шрифт, используемый в визуализации;

- ***Use dark theme*** – включает в Usermode- и Touch-визуализации тёмную тему оформления;
- ***Enable swipe gesture*** – включает жест “смахивание” для переключения между планами в Touch-визуализации;
- ***Disable object click animation*** – отключает анимацию нажатия на объекты;
- ***Dim inactive visualization after*** – время, по прошествии которого экран визуализации будет затемнён;
- ***Dim level*** – уровень затемнения экрана;
- ***Show alerts in Usermode*** – при появлении новых тревог в Usermode-визуализации будет открываться всплывающее окно.



System – стрелка рядом с кнопкой *System* открывает доступ к настройкам KNX-соединения, прав пользователей и удалённого доступа. Нажатие на кнопку *System* открывает окно настройки сети в новой вкладке браузера.

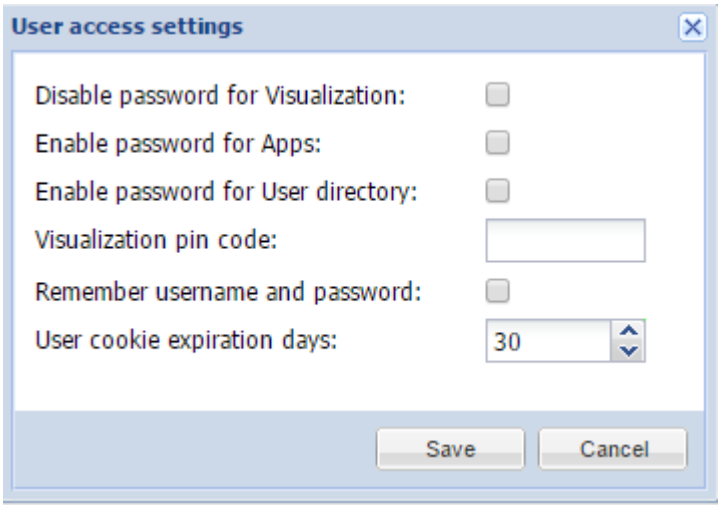


6.11. Вкладка User access

Вкладка *User access* отвечает за настройку прав пользователей.

<u>Настройки доступа пользователей</u>

Настройки доступа пользователей можно изменить с помощью кнопки *User access settings.*



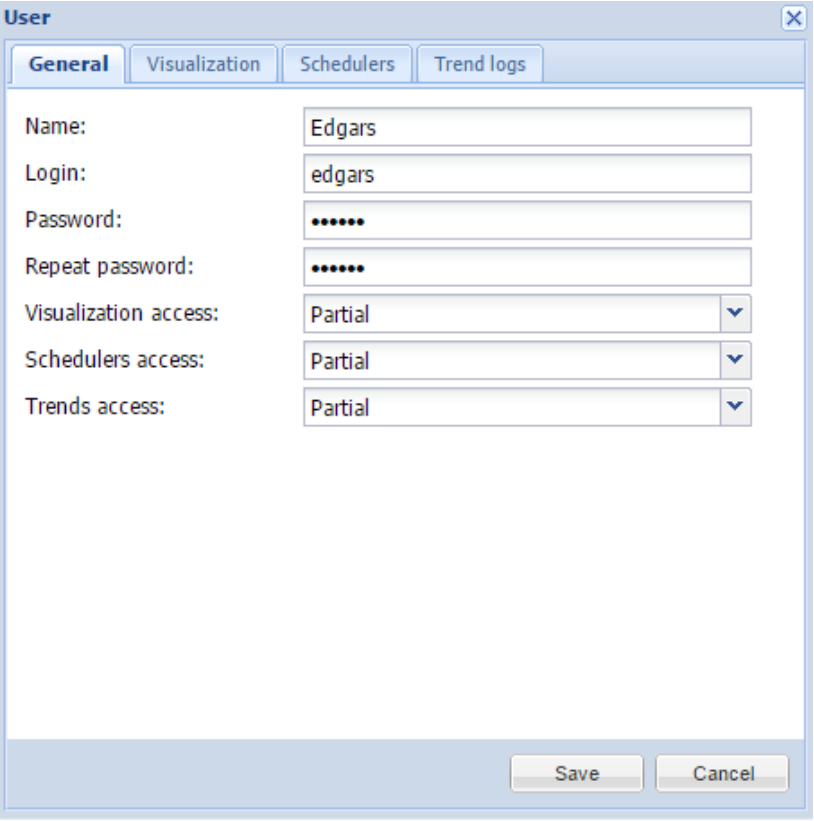
- ***Disable password for Visualization*** – отключить требование пароля для доступа к визуализации;
- ***Enable password for Apps*** – включить пароль для входа на главную страницу LogicMachine (открытие http://IP в браузере);
- ***Enable password for User directory*** – включить пароль для доступа к папке пользователей;
- ***Visualization PIN code*** – ПИН-код для доступа к визуализации;
- ***Remember username and password*** – запоминать учётные данные для входа в систему на указанный период времени;
- ***User cookie expiration days*** – срок действия пользовательских файлов cookie.

Папка User

Существует возможность загружать файлы через FTP. Для этого нужно включить FTP-сервер в *System config --> Services --> FTP server* и настроить данные для входа. После этого через него можно будет загружать файлы в папку User, которая доступна по адресу *http://IP/user*. Пароль для защиты этой папки может быть включен/отключен в *Logic Machine --> User access --> User access*.

Добавление пользователей

Осуществляется с помощью кнопки *Add new user.*

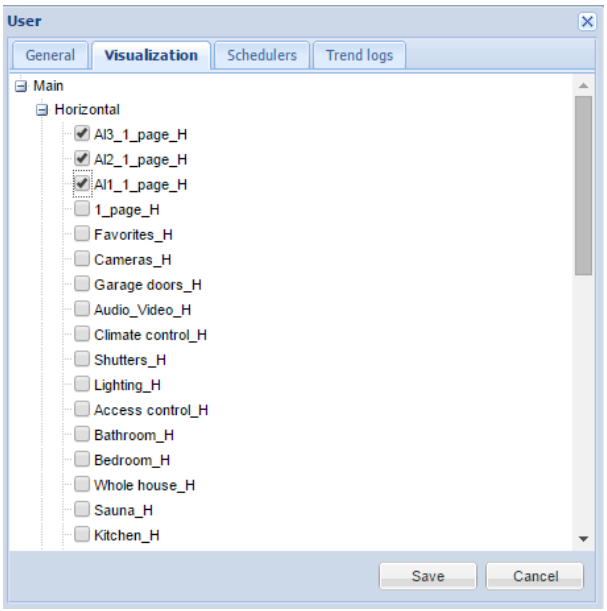


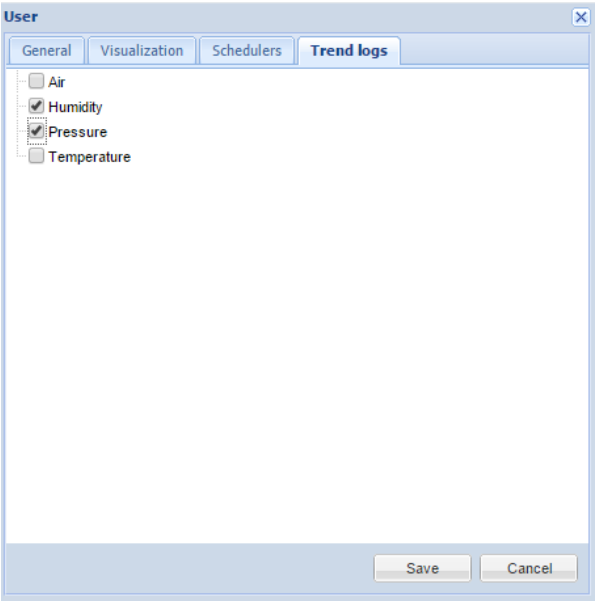
- ***Name*** – имя пользователя;
- ***Login*** – логин;
- ***Password*** – пароль;
- ***Repeat password*** – подтверждение пароля;
- ***Visualization access [None, Partial, Full]*** – уровень доступа к визуализации;
- ***Schedulers access [None, Partial, Full]*** – уровень доступа к планировщикам;
- ***Trends access [None, Partial, Full]*** – уровень доступа к трендлогам;

None – доступ запрещён;

Partial – частичный доступ: доступны определённые планы визуализации, планировщики и трендлоги;

Full – полный доступ.



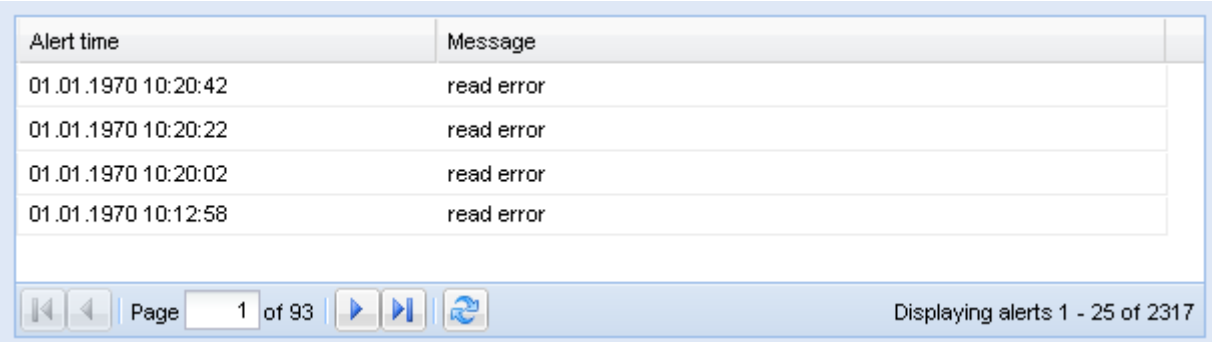


Лог доступа

Доступен по кнопке *Access logs*. Показывает время авторизации различных пользователей

6.12. Вкладка Alerts

На вкладке *Alerts* показываются сообщения, отправленные скриптами через функцию ***alert.*** Сообщения хранятся в встроенной памяти. Информация о запуске системы и статусе KNX-соединения также автоматически отображается на этой вкладке.



С помощью нижней панели вы можете перемещаться между страницами и обновлять список уведомлений.

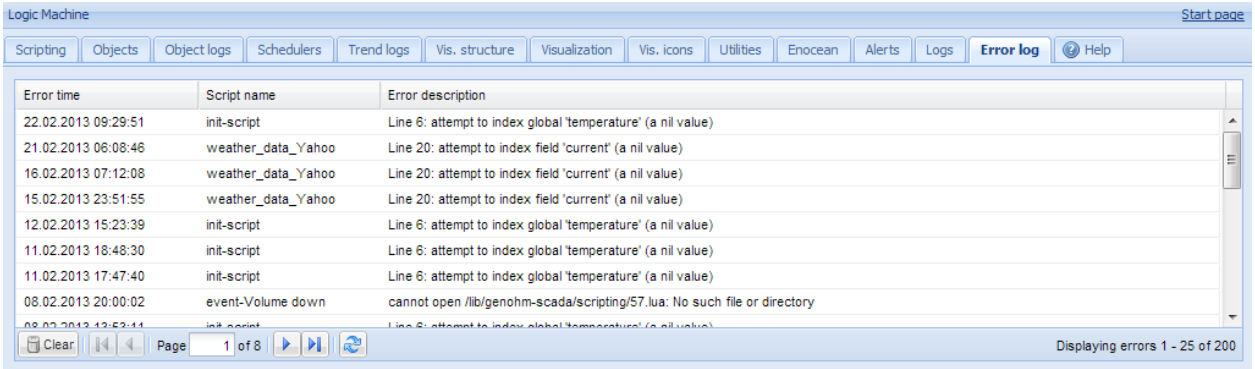
Пример

```
1.  temperature = 25.3
2.
3.  if temperature > 24 then
4.  -- отправка сообщения: 'Temperature levels are too high: 25.3'
5.  alert('Temperature level is too high: %.1f', temperature)
```

```
6. end
```

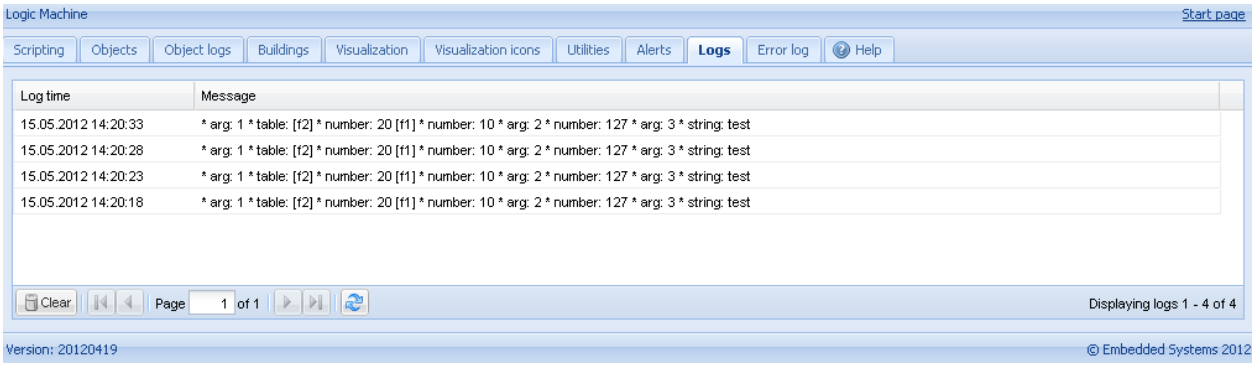
6.13. Вкладка Error log

Ошибки скриптов отображаются на вкладке *Error log.*



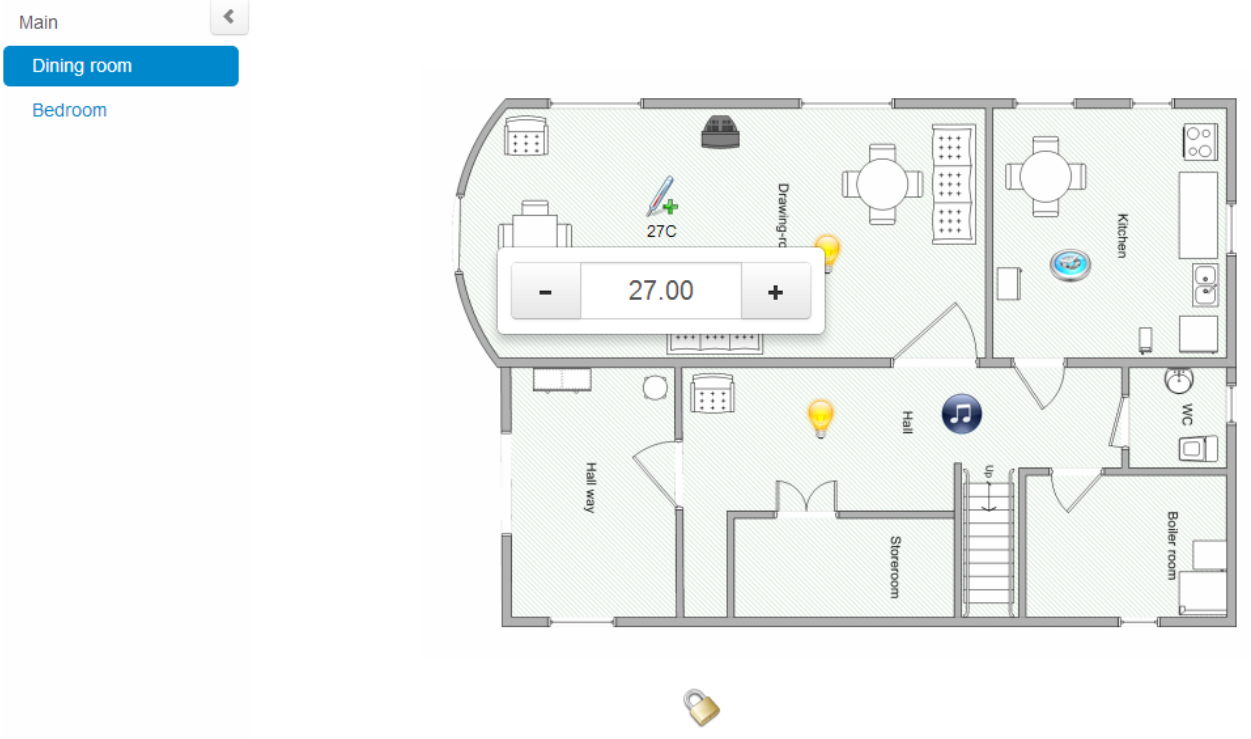
6.14. Вкладка Logs

Сообщения, отображаемые в этой вкладке, отправляются из скриптов с помощью функции ***log***. Логи можно использовать для отладки скриптов.



7. Usermode-визуализация

Usermode-визуализация отображает созданные планы визуализации. Настроить доступ пользователей к визуализации можно в *Logic Machine --> User access.*



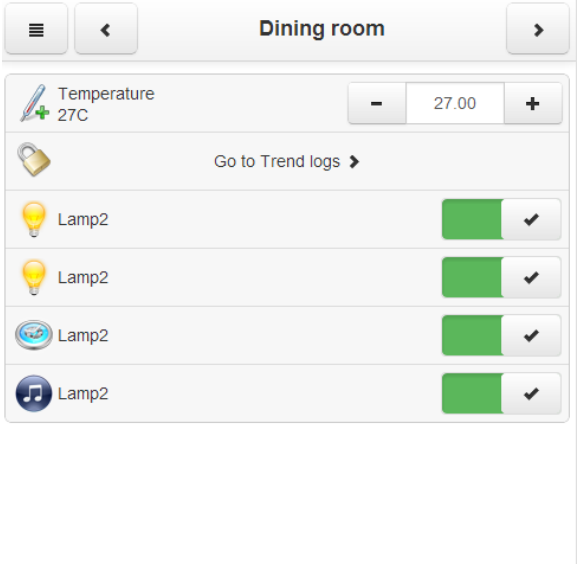
7.1. Пользовательский дизайн Usermode-визуализации

Пользовательские стили CSS позволяют создавать разнообразные креативные планы визуализации. Стили CSS задаются в *Vis. Graphics → Edit custom CSS*. Больше информации и примеров можно найти на форуме: http://forum.logicmachine.net/

8. Touch-визуализация

Touch-визуализация разработана для сенсорных устройств: iPhone/iPod/iPad/Android. Все объекты, добавленные на план Usermode-визуализации, по умолчанию видны в Touch-визуализации (если не включена опция *Hide in touch*).

Первая страница – *Building* (здание). На ней вы можете выбрать план этажа, которым хотите управлять. После выбора этажа на визуализации отобразятся все объекты, которые на нём расположены, и вы сможете ими управлять.

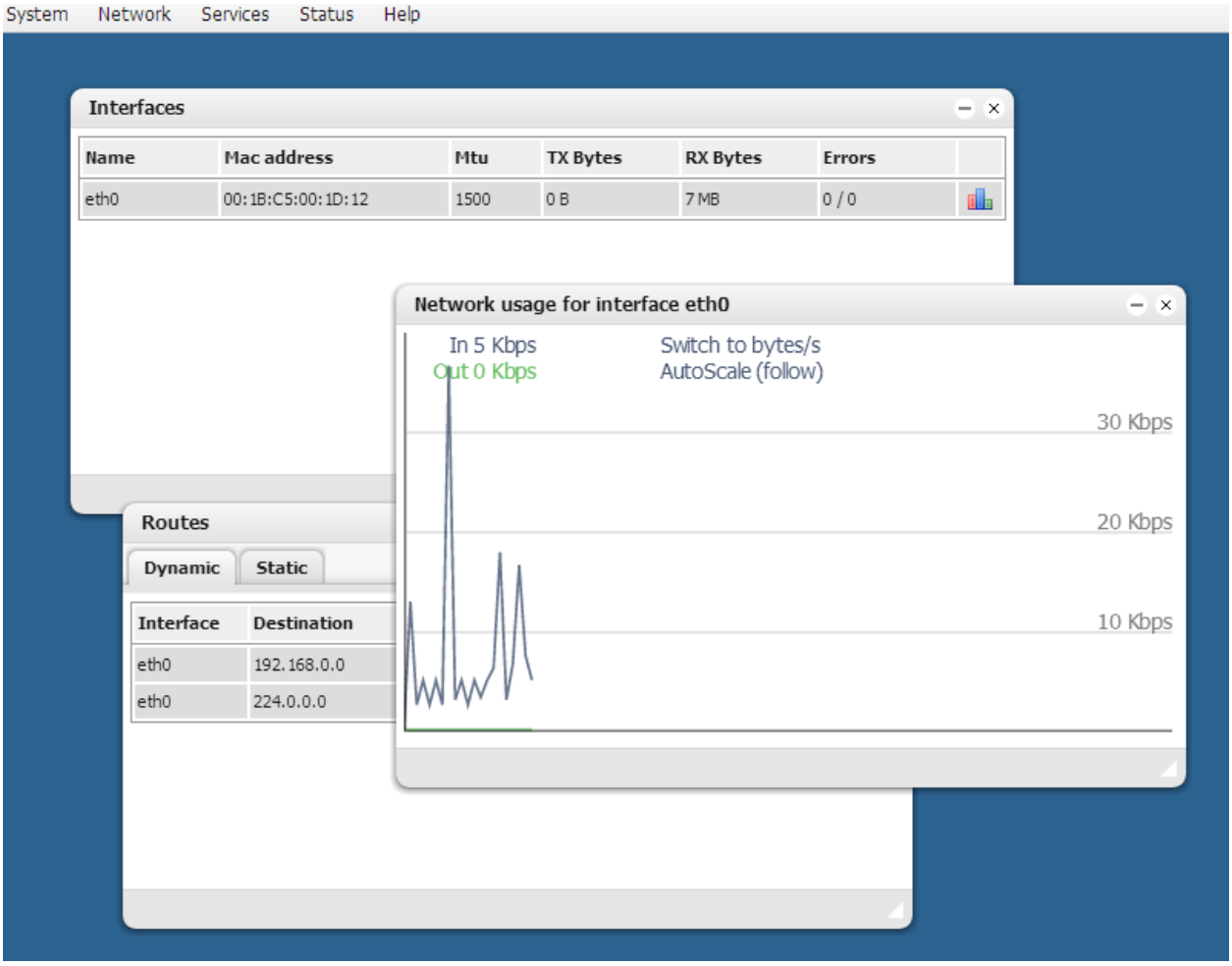


<u>Запуск визуализации на сенсорном устройстве (например iPad)</u>

- Убедитесь, что ваш iPad находится в одной сети с LogicMachine;
- В браузере введите IP-адрес LogicMachine (по умолчанию 192.168.0.10);
- Нажмите на иконку Touch Visualization;
- Сохраните приложение в виде ярлыка на вашем iPad.

9. Настройки системы

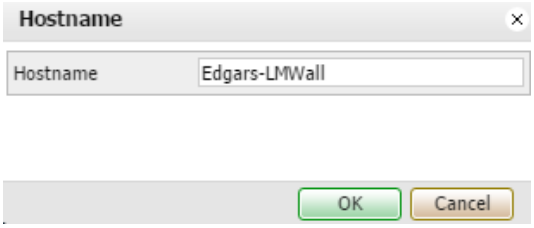
Окно *System configuration* позволяет управлять функциями маршрутизатора KNX/EIB LogicMachine, а также осуществлять управление доступом, обновлять прошивку, просматривать состояние сети или системы и т. д.



Login	**Password**
admin	admin

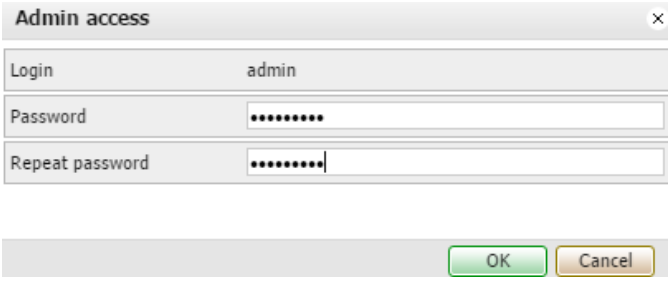
9.1. Имя хоста

Имя хоста можно изменить в *System* → *Hostname*. Это имя будет отображаться при поиске устройства с помощью приложений Zeroconf или Discovery.



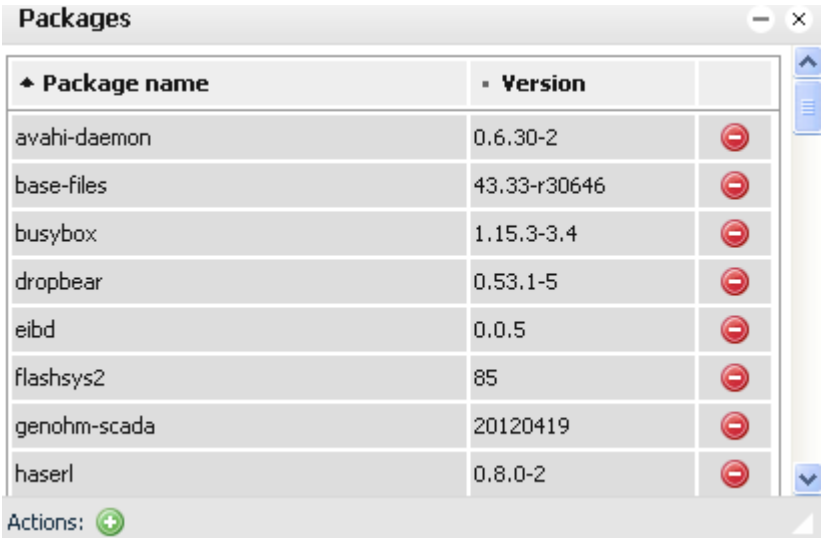
9.2. Изменение пароля администратора

Изменить пароль администратора можно в *System* → *Admin access*.



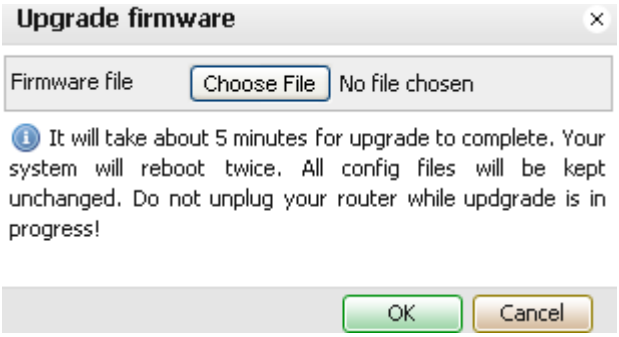
9.3. Пакеты

В *System* → *Packages* отображаются пакеты, установленные в системе. Вы можете добавить новый пакет, нажав на иконку +.



9.4. Обновление прошивки

Полное обновление системы производится в *System → Upgrade firmware.* При этом обновляется как ОС, так и LogicMachine.



9.5. Перезагрузка LogicMachine

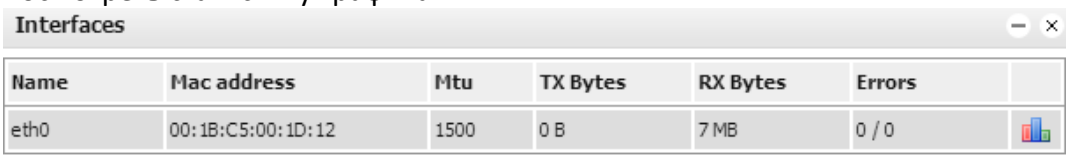
Чтобы перезагрузить LogicMachine, нажмите *System → Reboot*.

9.6. Выключение LogicMachine

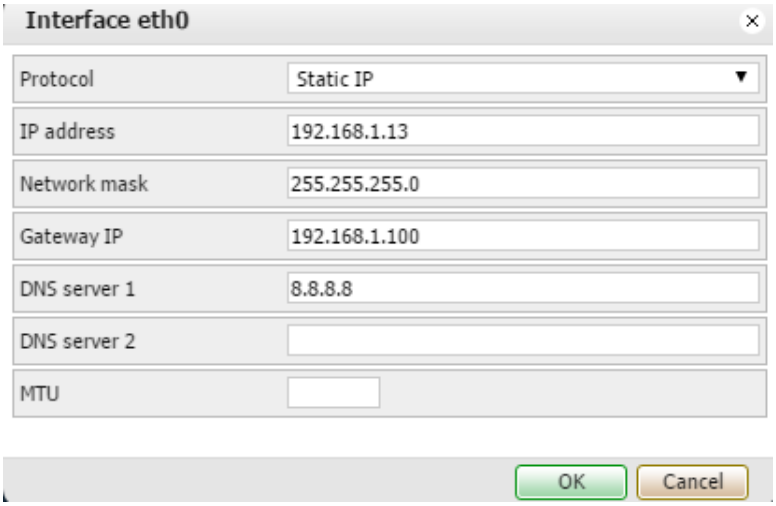
Чтобы выключить LogicMachine, нажмите *System → Shutdown*. Рекомендуется выключать систему перед отключением питания, чтобы обеспечить корректное сохранение базы данных.

9.7. Настройки интерфейсов

Настройки интерфейсов находятся в окне *Network → Interfaces.*
На первой вкладке расположен интерфейс Ethernet. Можно включить/выключить его или посмотреть статистику трафика.



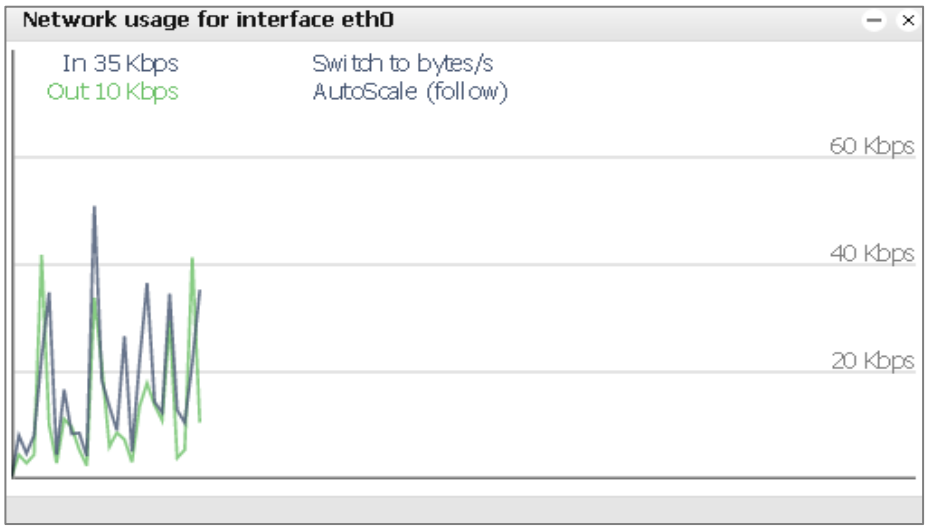
При нажатии на интерфейс откроется окно с его настройками.



- ***Protocol*** – протокол для адресации:
  - ***Static IP*** – постоянный IP-адрес. По умолчанию – 192.168.0.10;
  - ***DHCP*** – использовать протокол DHCP для получения IP-адреса;
    - ***Current IP*** – IP, полученный от DHCP-сервера. Это поле видно только в том случае, если IP-адрес был получен;
- ***Network mask*** – маска сети. По умолчанию – 255.255.255.0 (/24);
- ***Gateway IP*** – IP-адрес шлюза;
- ***DNS server*** – IP-адрес DNS-сервера;
- ***MTU*** – наибольший размер пакета, который может быть передан в протоколе связи. По умолчанию – 1500.

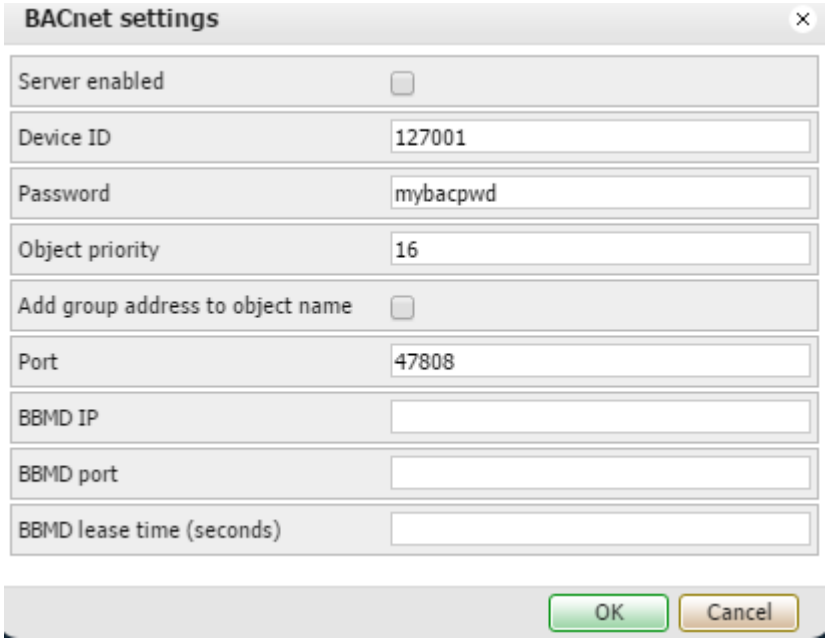
График данных интерфейса Ethernet

В главном окне вкладки Interfaces расположена кнопка . Нажатие на неё открывает новое окно, в котором отображается график трафика, проходящего через интерфейс, в режиме реального времени (отображается как входящий, так и исходящий трафик). Можно выбрать единицы измерения – байты/с или биты/с.



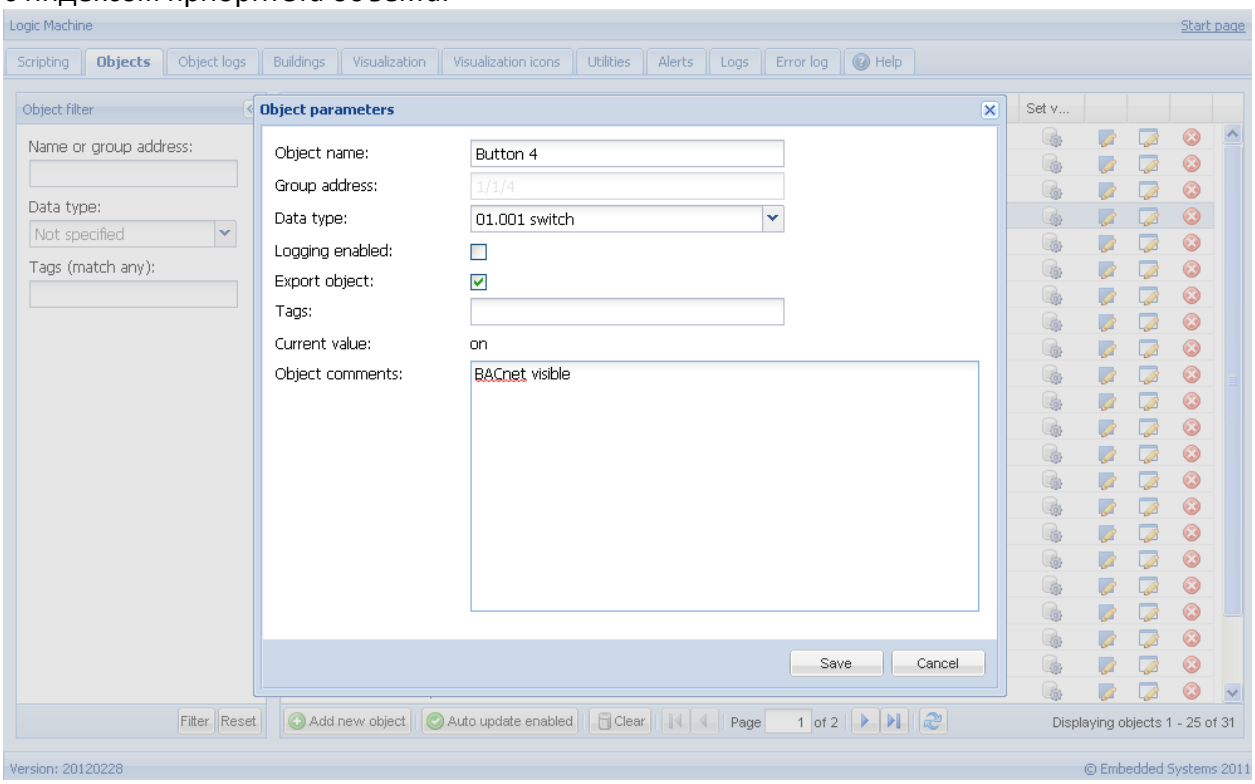
9.8. Настройки BACnet

Настройки сервера BACnet расположены в меню *Network → BACnet Settings.*



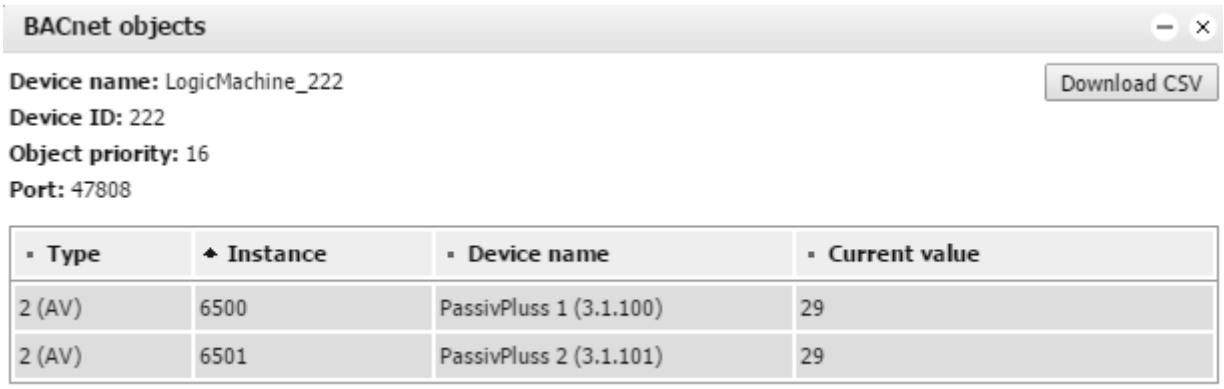
Server enabled – включить/отключить сервер BACnet;
Device ID – ID устройства в сети BACnet;
Password – пароль устройства;
Object priority – приоритет объекта;
Add group address to object name – автоматически добавлять групповой адрес к названию объекта;
Port – номер порта;
BBMD IP – IP-адрес BACnet-роутера. После установки IP-адреса и порта LM начнёт вести себя как внешнее устройство и попробует зарегистрироваться на роутере BACnet;
BBMD port – порт BACnet-роутера;
BBMD lease time (seconds) – интервал между повторными регистрациями.

Чтобы сделать объекты KNX/EIB доступными для чтения и записи в BACnet, отметьте в их настройках поле *Export object*. Бинарные объекты отобразятся как двоичные значения, остальные числовые объекты – как аналоговые. Другие типы данных на данный момент не поддерживаются. Запись в шину KNX перепишет значение массива приоритета в соответствии с индексом приоритета объекта.



9.9. Объекты BACnet

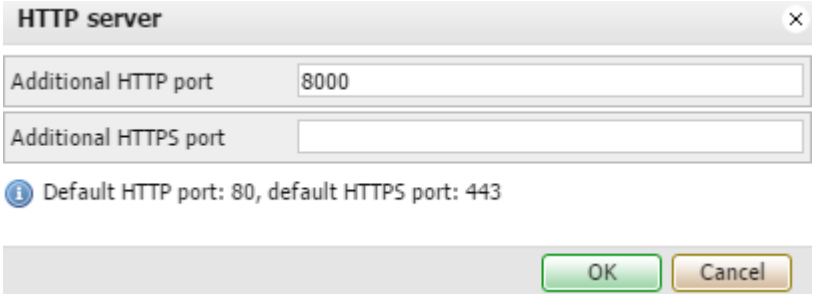
В *Network* → *BACnet objects* указаны объекты, которые посылаются в сеть BACnet.



Type	Instance	Device name	Current value
2 (AV)	6500	PassivPluss 1 (3.1.100)	29
2 (AV)	6501	PassivPluss 2 (3.1.101)	29

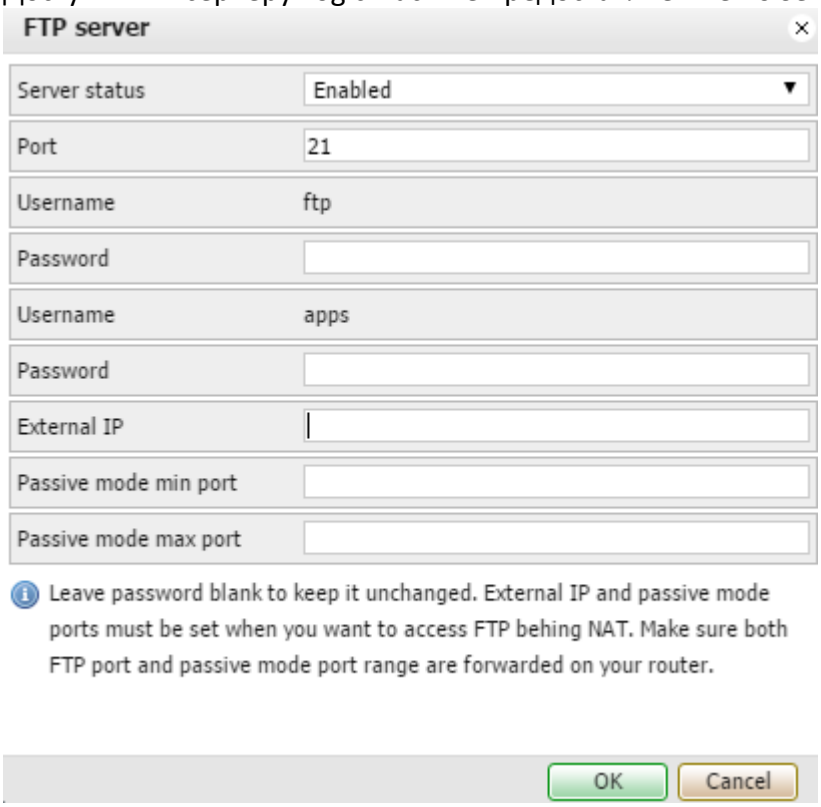
9.10. HTTP-сервер

Если вам понадобятся дополнительные порты для запуска веб-сервера, настройте их в меню *Network* → *HTTP server*. По умолчанию, порт HTTP – 80, порт HTTPS – 443.



9.11. FTP-сервер

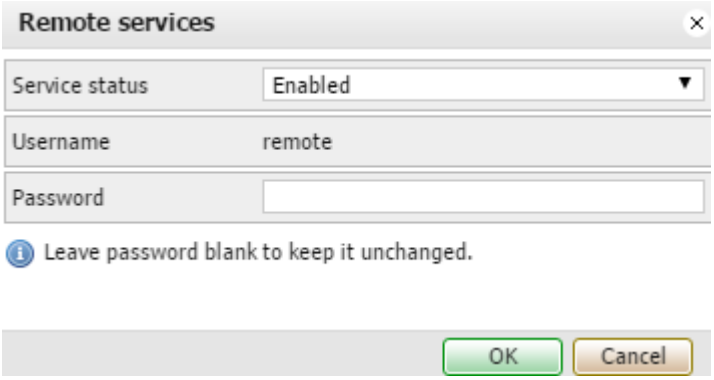
Доступ к FTP-серверу LogicMachine предоставляет меню *Service → FTP Server*.



- ***Server status*** – включение/отключение FTP-сервера;
- ***Port*** – порт сервера;
- ***Username*** – логин: *ftp;*
- ***Password*** – пароль для пользователя *ftp*, длина 4-20 символов;
- ***Username*** – логин для доступа к папке http://IP/user: *apps*. Вы можете включить или отключить авторизацию для доступа к этой папке в меню *Logic Machine → User access → User access settings;*
- ***Password –*** пароль для пользователя *apps*, длина 4-20 символов;
- ***Passive mode min port*** – минимальный порт для пассивного режима FTP;
- ***Passive mode max port*** – максимальный порт для пассивного режима FTP.

9.12. Удалённый доступ

Настройка удалённого доступа осуществляется в меню *Service → Remote services.*



- ***Service status*** – включение/отключение удалённого доступа;
- ***Username*** – логин: *remote;*
- ***Password*** – пароль.

URL

Измените IP-адрес и пароль согласно настройкам вашей LM:

```
http://remote:remote@192.168.0.10/scada-remote?m=rss&r=alerts
```

Параметры запроса

m задаёт формат возвращаемого значения:

- json;
- xml;
- rss (только для ошибок и уведомлений).

r - запрашиваемая функция:

- alerts – последние 50 уведомлений;
  Возвращаемые значения:
  - alert – текст уведомления;
  - time – время уведомления (временная метка UNIX);
  - date – дата уведомления (в формате RFC);

- errors – последние 50 ошибок;
  Возвращаемые значения:
  - error – текст ошибки;
  - script – название скрипта, в котором произошла ошибка;
  - time – время ошибки (временная метка UNIX);
  - date – дата ошибки (в формате RFC);

- objects – список объектов, отмеченных для экспорта и отсортированных по времени последнего обновления;
  Возвращаемые значения:
  - address – адрес объекта ("1/1/1");
  - name – название объекта ("My object");
  - data – расшифрованное значение объекта (42 или "01.01.2012");
  - datatype – тип данных объекта (1 или 5.001);
  - time – время обновления объекта (временная метка UNIX);
  - date – дата обновления объекта (в формате RFC);
  - comment – комментарий к объекту ("Second floor entry lights");
  - tags – необязательный массив тегов ("Light", "Second floor").

- grp – выполняет одну из функций grp;
  Параметры:
  - fn – название функции, обязательно:
    - getvalue – возвращает текущее значение объекта, если оно найдено;
    - find – возвращает информацию об объекте;
    - write – посылает в шину KNX телеграмму на запись;
    - response – посылает в шину KNX телеграмму на ответ;
    - read – посылает в шину KNX телеграмму на чтение;
    - update – обновляет значение виртуального объекта LM без использования шины KNX;
  - alias – групповой адрес или название объекта, обязательно;
  - value – значение для функций записи, обязательно для функций write / response / update (кроме типов данных time и date);
    Параметры для типа данных time:
    - day – число (0-7), день недели, необязателен;
    - hour – число (0-23);
    - minute – число (0-59);
    - second – число (0-59);

    Параметры для типа данных date:

    - day – число (1-31);
    - month – число (1-12);
    - year – число (1990-2089);
  - datatype – необязателен для функций write / response / update; тип данных берётся из базы данных, если не задан;
    Возможные значения:
    bool bit2 bit4 char uint8 int8 uint16 int16 float16
    time date uint32 int32 float32 access string

Примеры

Запись значения 50 на адрес 1/1/1:

http://remote:remote@192.168.0.10/scada-remote?m=json&r=grp&fn=write&alias=1/1/1&value=50

Запись булевого значения на адрес 1/1/2 (можно использовать true или false, а можно – 1 или 0):

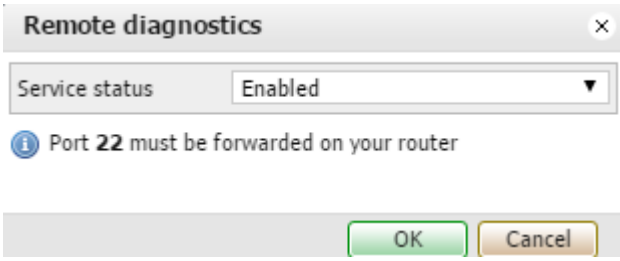
http://remote:remote@192.168.0.10/scada-remote?m=json&r=grp&fn=write&alias=1/1/2&value=true

Задание типа данных scale и запись значения 50 на адрес 1/1/1:

http://remote:remote@192.168.0.10/scada-remote?m=json&r=grp&fn=write&alias=1/1/1&value=50&datatype=scale

9.13. Удалённая диагностика

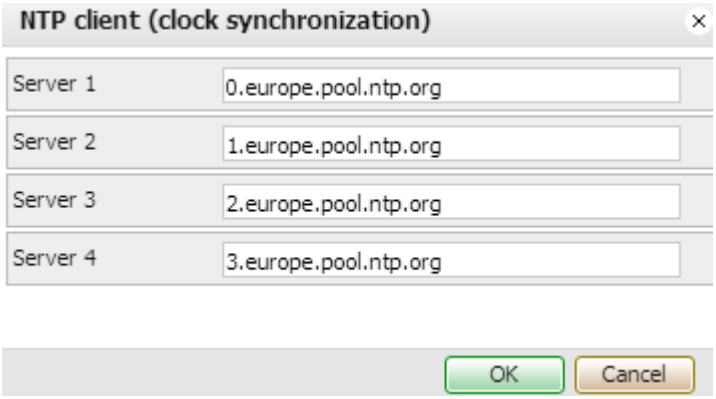
Удалённую диагностику (*Service → Remote diagnostics*) рекомендуется использовать только в случае необходимости удалённой поддержки устройства персоналом Embedded Systems. Она включает доступ к устройству по SSH.



➢ ***Service status*** – включает/отключает доступ к устройству по SSH.

9.14. NTP-клиент (синхронизация времени)

Данные NTP-серверов задаются в окне *Service → NTP client*.


NTP client (clock synchronization)
Server 1
0.europe.pool.ntp.org
Server 2
1.europe.pool.ntp.org
Server 3
2.europe.pool.ntp.org
Server 4
3.europe.pool.ntp.org
OK
Cancel

9.15. Статус системы

Общие сведения о системе (использование ЦП, памяти, разделы диска и сериальные порты) указаны в окне *Status* → *System status*.

System status

General | Memory usage | Partitions | Serial ports

Parameter	Value
CPU model	ARM926EJ-S rev 5 (v5l)
CPU BogoMips	227.12
Linux kernel version	3.18.24
System uptime	2d 2h 0m
Load averages	0.01 0.03 0.05

9.16. Сетевые утилиты

Утилиты *Ping* и *Traceroute* находятся в окне *Status* → *Network utilities*. Можно использовать как IP-адрес, так и адрес домена.

Network utilities

Ping | Traceroute

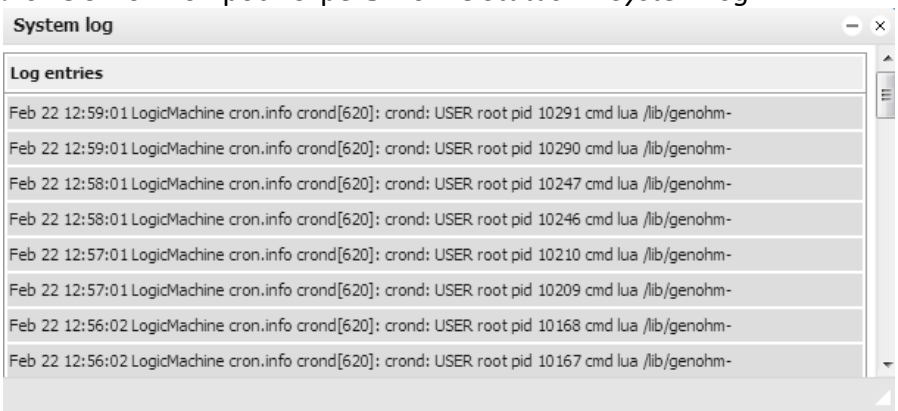
IP / Hostname 192.168.1.100

```
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=64 time=0.432 ms
64 bytes from 192.168.1.100: seq=1 ttl=64 time=0.385 ms
64 bytes from 192.168.1.100: seq=2 ttl=64 time=0.383 ms
64 bytes from 192.168.1.100: seq=3 ttl=64 time=0.385 ms
64 bytes from 192.168.1.100: seq=4 ttl=64 time=0.385 ms

--- 192.168.1.100 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.383/0.394/0.432 ms
```

OK | Cancel

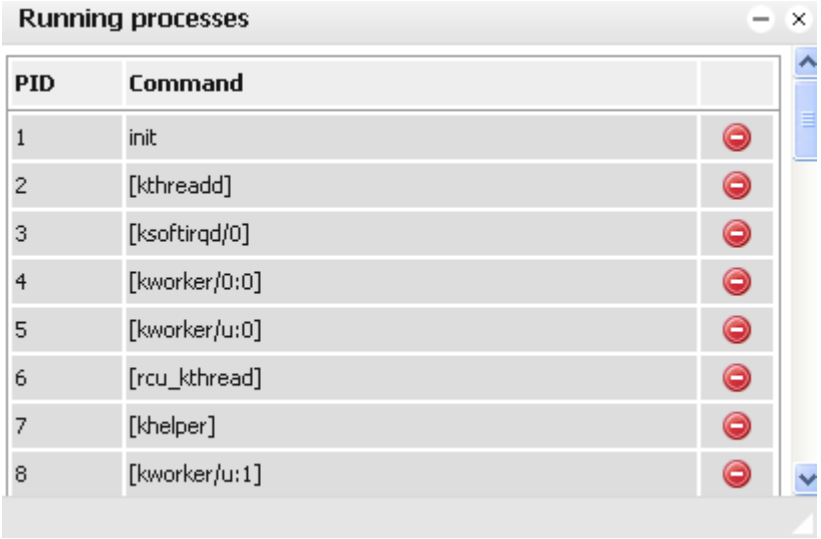
9.17. Системный лог

Лог ОС можно просмотреть в окне *Status → System log*.



9.18. Процессы

Информация о запущенных в системе процессах указана в окне *Status* → *Running processes*.

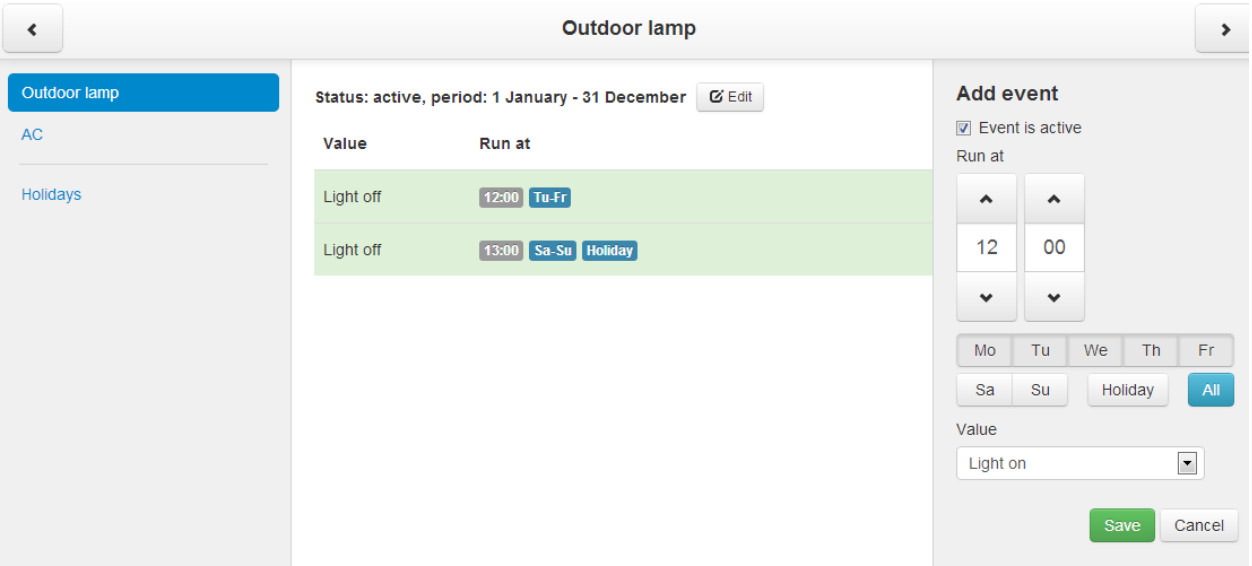


10. Пользовательский режим планировщика

Пользовательский режим планировщика обладает удобным интерфейсом, позволяющим конечному пользователю управлять задачами планировщика: например, задавать значения термостата в зависимости от дня недели, времени и наличия/отсутствия праздников.

10.1. События

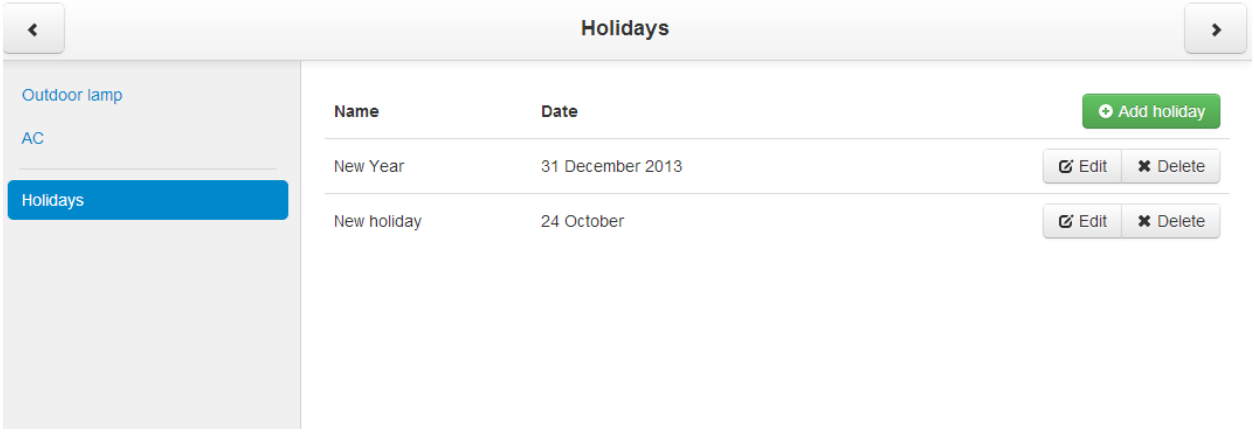
Каждый планировщик связан с определённым групповым адресом, заданным в панели администратора (*см. раздел 1.4 этой документации*).



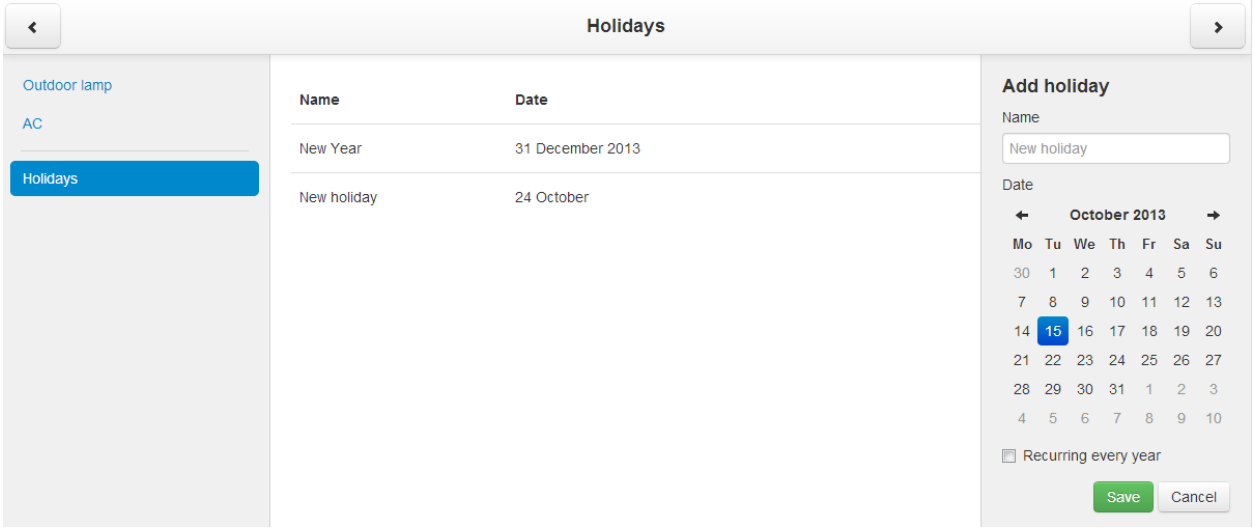
При создании новой задачи для планировщика вы можете указать дни недели, время запуска и передаваемое на объект значение.

10.2. Выходные дни

В *Holidays* задаются особые даты, которые позже можно будет использовать при создании новых событий.

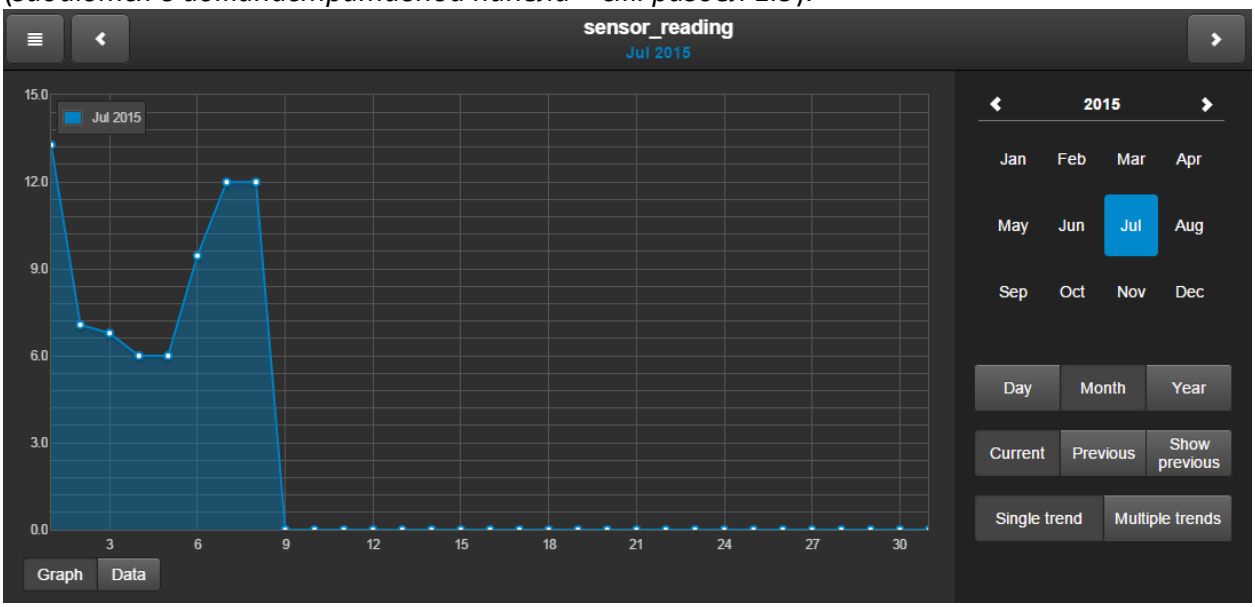


Нажмите на кнопку *Add new holiday*, чтобы добавить новый выходной день.

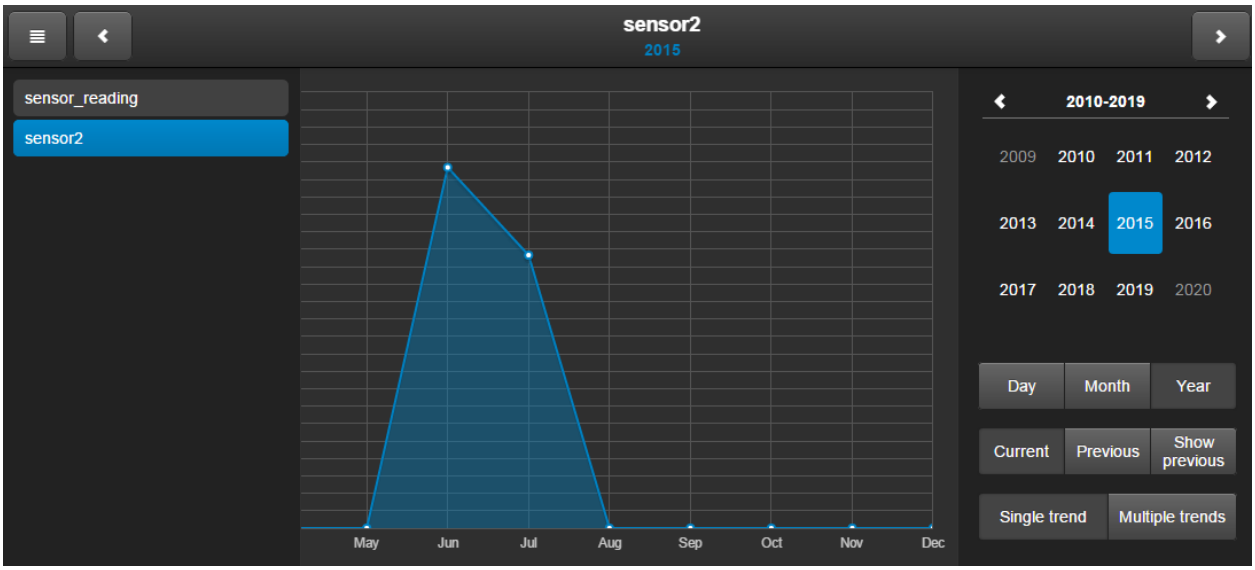


11. Трендлоги

Интерфейс трендлогов позволяет конечному пользователю просматривать динамику трендов (*задаются в административной панели – см. раздел 1.5*).

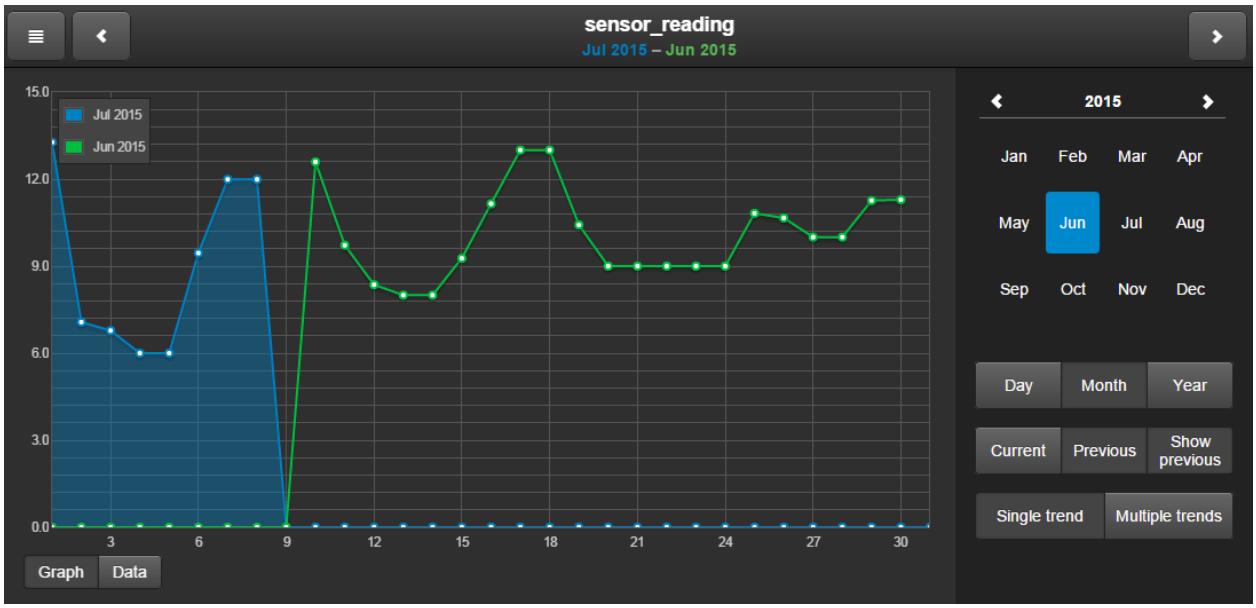


Нажав на кнопку , вы можете выбрать необходимый трендлог, который привязан к определённому групповому адресу KNX.

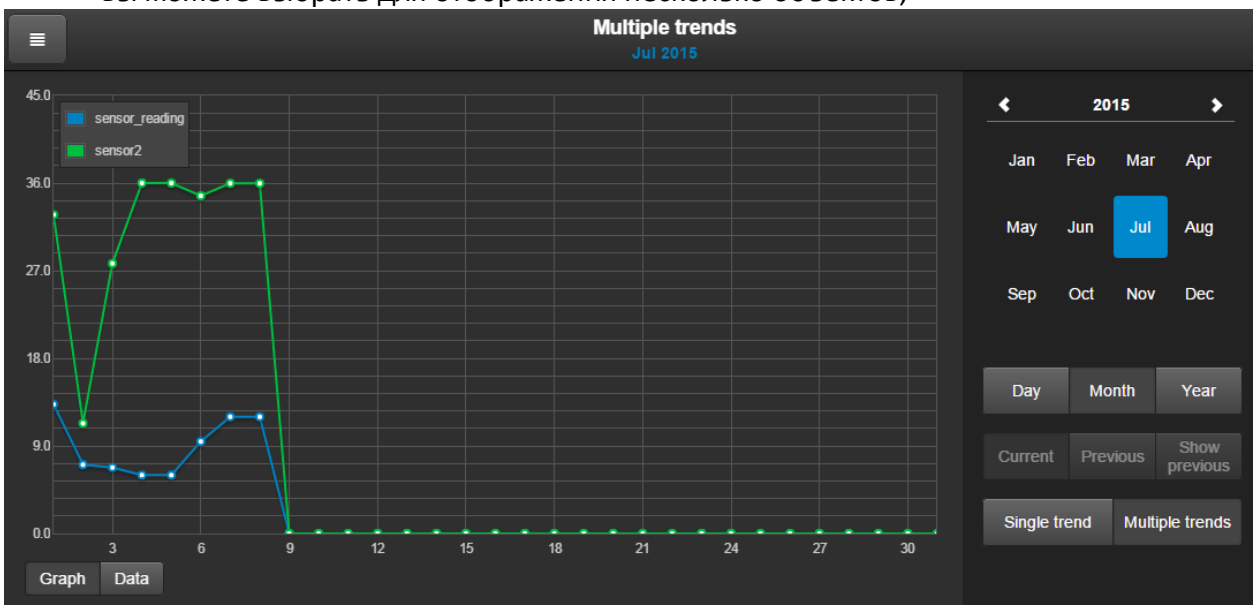


- ***Day*** – просмотреть трендлог за день;
- ***Month*** – просмотреть трендлог за месяц;
- ***Year*** – просмотреть трендлог за год;
- ***Current*** – текущий трендлог; задан синим цветом, вы можете просмотреть его за день, месяц или год;
- ***Previous*** – предыдущий временной период для трендлога; вы можете просмотреть его за день, месяц или год;
- ***Show previous*** – показывает значения трендлога за предыдущий временной

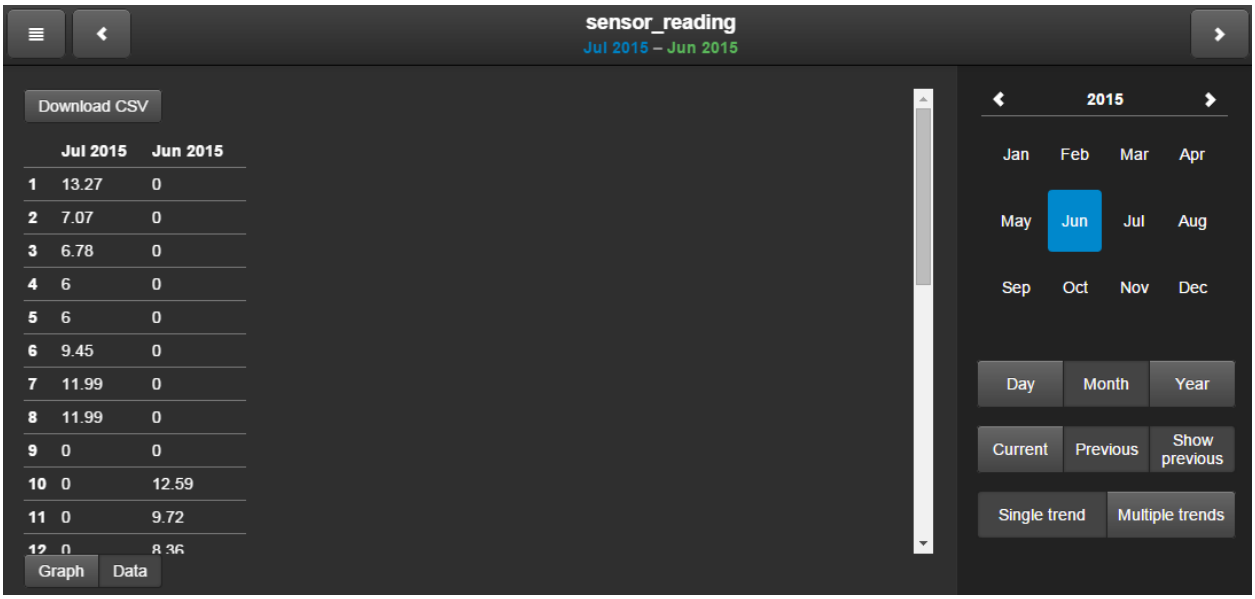
период (зелёная линия) вместе с текущим;



- ***Single trend*** – показывает один трендлог;
- ***Multiple trends*** – показывает несколько трендлогов. Когда включен этот режим, вы можете выбрать для отображения несколько объектов;

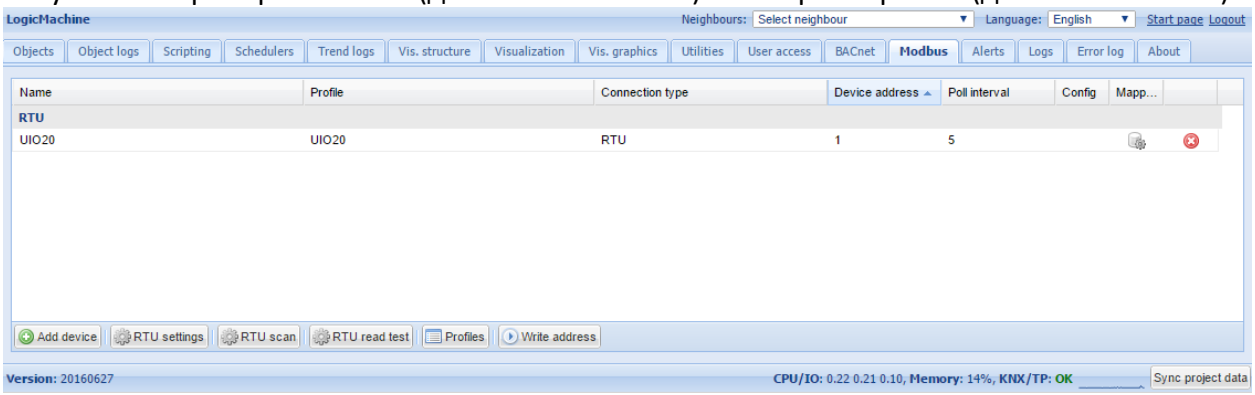


Нажатие на кнопку *Data* показывает значения точек в виде таблицы. Полученная таблица может быть экспортирована в формат CSV.



12. Соединение между Modbus RTU/TCP и LM

Поддержка Modbus TCP добавляется установкой специального пакета через меню *Sys config* → *System* → *Packages* и осуществляется через порт Ethernet. Соединение с Modbus задаётся через визуальный преобразователь (для Modbus Master) или через скрипты (для Modbus Slave).



Modbus Master – используйте графический интерфейс на вкладке Modbus;
Modbus Slave – чтобы использовать LM как Modbus Slave, отключите Modbus RTU в *Modbus* → *RTU settings* и используйте скрипты для соединения.

12.1. Профиль устройства Modbus

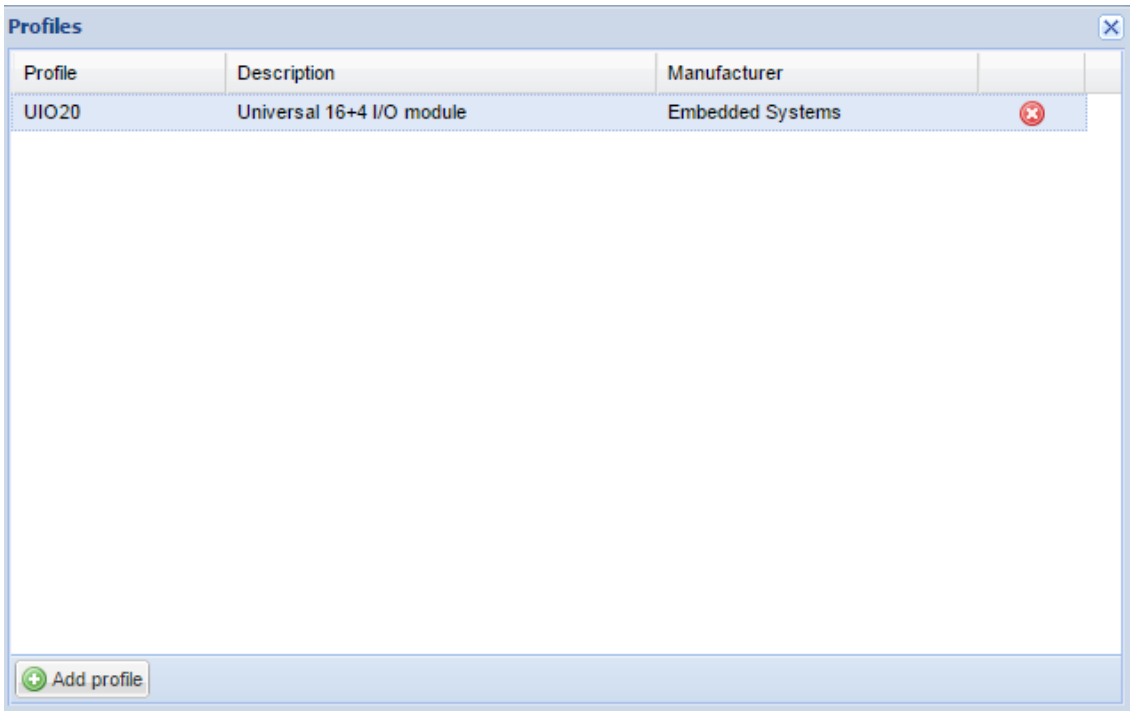
Для начала определим, что такое профиль устройства Modbus: это *.json-файл со следующей структурой (в качестве примера используем фрагмент из профиля для устройства UIO20 от Embedded Systems):

```
{
"manufacturer": "Embedded Systems",
"description": "Universal 16+4 I/O module",
```

```
"mapping": [
{ "name": "Output 1", "bus_datatype": "bool", "type": "coil", "address": 0, "writable": 1 },
{ "name": "Input 1", "bus_datatype": "float16", "type": "inputregister", "address": 0, "value_multiplier": 0.001, "units": "V" }
]
}
```

Name – название объекта: например, Output 2 (строка, обязательный);
Bus_datatype – тип данных KNX, ключ из таблицы dt: например, float32 (строка/число, обязательный);
Type – тип регистра Modbus, возможные значения: coil, discreteinput, register, inputregister (строка, обязательный);
Address – адрес регистра (число, обязательный);
Writable – значение true разрешает запись в регистр, если type – coil или register (Boolean);
Datatype – тип данных Modbus; если задан, конвертация будет произведена автоматически. Возможные значения: uint16, int16, float16, uint32, int32, float32, uint64, int64, quad10k, s10k (строка);
Value_delta – новое значение посылается только в случае, если оно отличается от старого на некоторую дельту. По умолчанию 0, отправляется после каждого запроса (число);
Value_multiplier – умножает полученное значение на указанное число, value = value_base + value * value_multiplier (число);
Value_bitmask – битовая маска; сдвиг выполняется автоматически на основе наименее значимой 1, найденной в маске (число);
Value_nan – массив 16-битных целых чисел. Если это значение определено, а операция чтения возвращает тот же массив, дальнейшая обработка значения не производится (массив);
Value_conv – применяет одну из встроенных функций преобразования (строка, Internal);
Value_custom – имя встроенной функции перечисления или список сопоставлений key->value; результирующее значение будет равно 0, если ключ не найден (строка/объект);
Internal – не отображается для пользователя, если установлено значение true, используется для scale-регистров (Boolean);
Units – единица измерения KNX-объекта (строка);
Address_scale – адрес регистра, содержащего шкалу значений, value = value * 10 ^ scale (число);
Read_count – количество регистров для чтения за один раз; используется для устройств, которые поддерживают только чтение некоторого блока регистров (число);
Read_swap – включает/выключает изменение порядка байтов при конвертации (Boolean);
Read_offset – положение первого регистра данных из блока регистров (число).

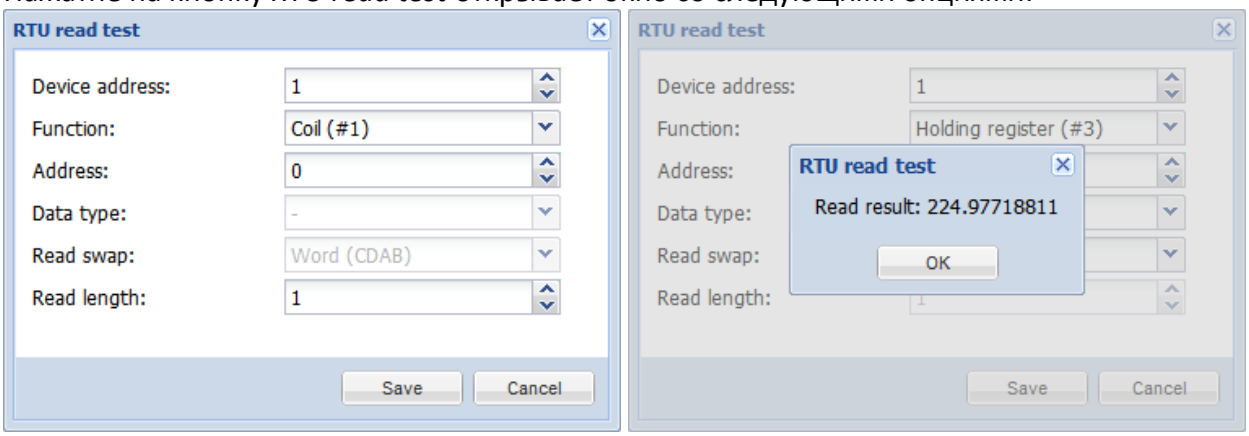
Когда профиль устройства Modbus создан, загрузите его, нажав на кнопку *Profiles*.



12.2. Чтение coil/register ModBus RTU из интерфейса

Поскольку создавать новые профили Modbus довольно неудобно, мы добавили новую функцию, которая позволяет считывать *coil* или *register* Modbus прямо из пользовательского интерфейса. Она поможет пользователям найти правильные настройки и адреса перед созданием новых профилей. Пока что функция работает только с RTU-соединением, TCP планируется реализовать позже.

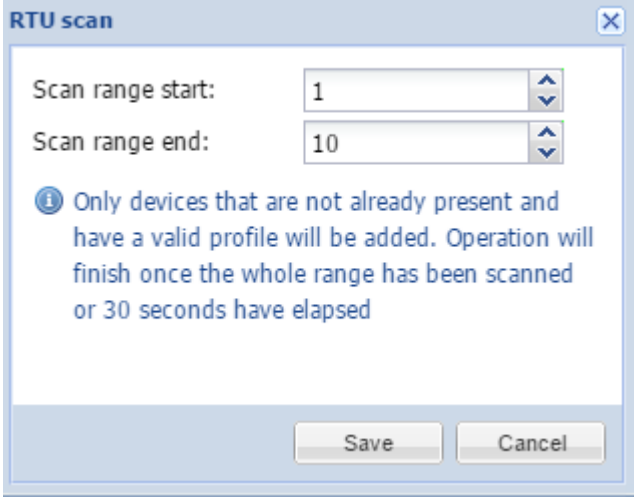
Нажатие на кнопку *RTU read test* открывает окно со следующими опциями:



- ***Device address*** – адрес устройства Modbus;
- ***Function (Coil, Discrete input, Holding register, Input register)*** – тип функции;
- ***Address*** – адрес регистра данных;
- ***Data type*** – тип данных, может быть выбран только для регистров;
- ***Read swap (None (ABCD); Word (CDAB); Byte (BADC); Byte and word (DCBA)*** – прочитанное значение изменяется в выбранном порядке;
- ***Read length*** – количество читаемых за один раз registers/coils.

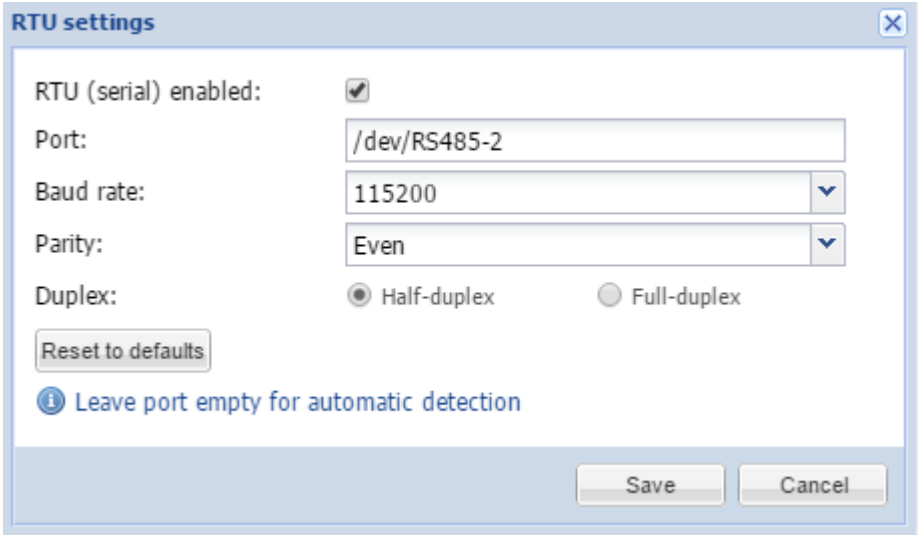
12.3. Сканирование RTU

Сканирует порт RS-485 на наличие подключённых устройств ModBus RTU. Добавляются только неопределённые устройства с корректным профилем. Операция закончится после сканирования всего диапазона или по истечении 30 секунд.



- ***Scan range start*** – начальный адрес устройства ModBus для поиска;
- ***Scan range end*** – конечный адрес устройства ModBus для поиска.

12.4. Настройки RTU

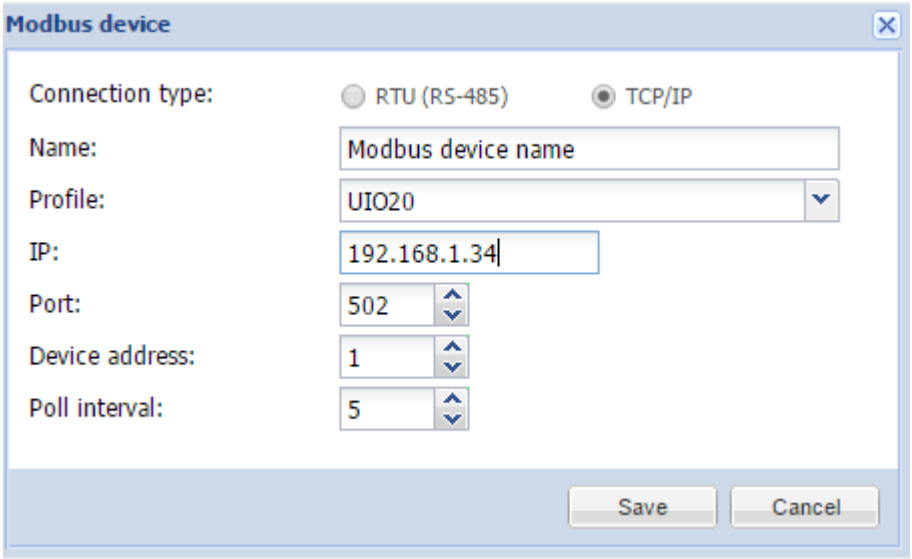


- ***RTU (serial) enabled*** – включает/выключает Modbus RTU;
- ***Port (/dev/RS485-1; /dev/RS485-2)*** – задаёт порт для связи (оставьте пустым для автоматического определения);
- ***Baud rate (1200 .. 500000)*** – скорость связи;

- ***Parity (None 1 stop bit; Odd, Even, None 2 stop bits)*** – чётность;
- ***Duplex*** – определяет тип соединения: полудуплексное или дуплексное.

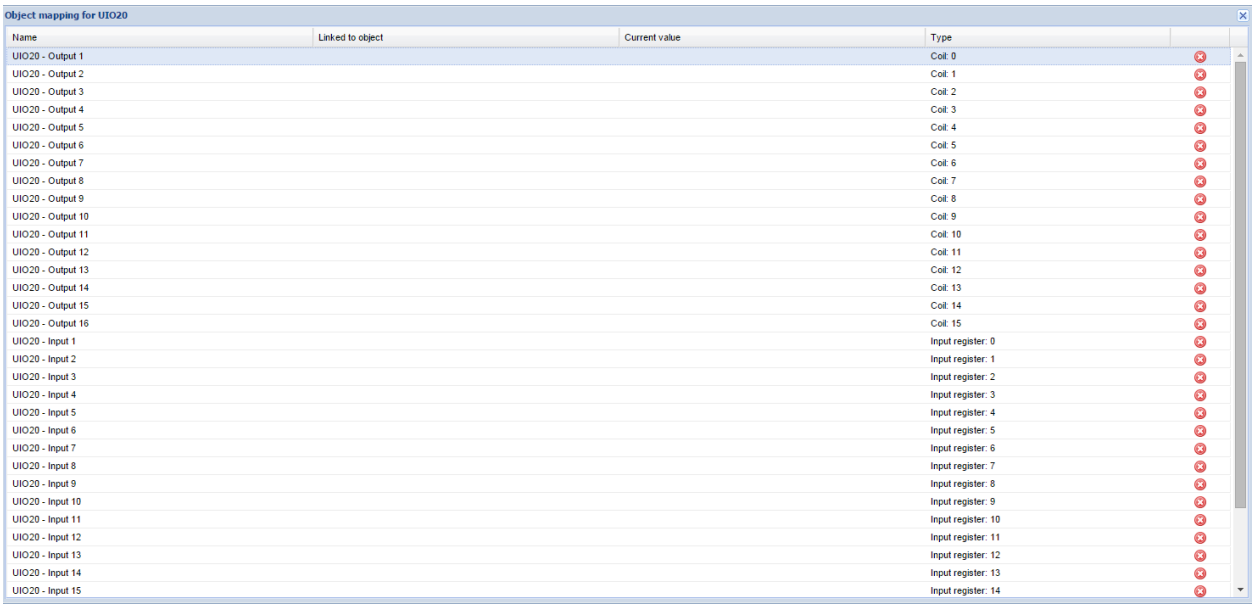
12.5. Добавление устройств Modbus

После настройки профилей добавьте устройство Modbus, нажав на кнопку *Add device*.

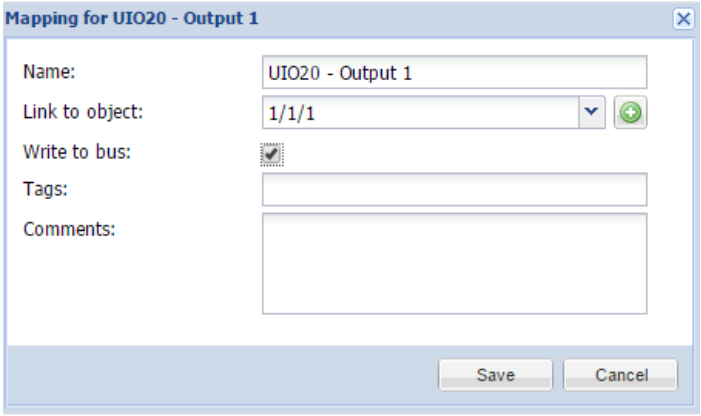


- ***Connection type*** – тип соединения (Modbus RTU или Modbus TCP);
- ***Name*** – название устройства;
- ***Profile*** – профиль устройства;
- ***Device address*** – адрес устройства;
- ***Poll interval (seconds)*** – интервал опроса устройства;
- ***IP*** – IP-адрес устройства при использовании Modbus TCP;
- ***Port*** – коммуникационный порт устройства при использовании Modbus TCP.

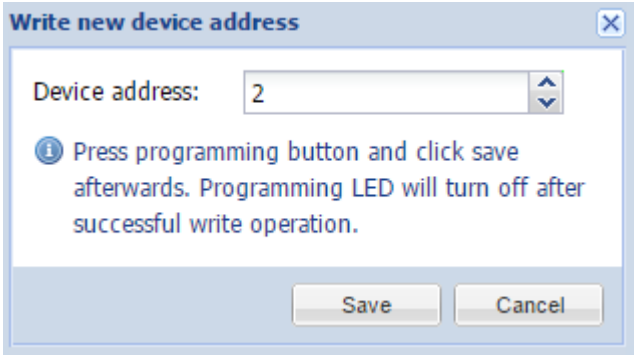
После добавления устройства вы можете задать соответствие объектов устройства Modbus с объектами KNX, нажав на иконку . Сначала вы увидите список всех объектов устройства Modbus.



Чтобы задать соответствие, нажмите на нужный объект.



12.6. Программирование адреса устройства UIO20E Modbus



Существует отдельная кнопка *Write address,* предназначенная для задания адреса устройству UIO20E. Нажмите на кнопку *Programming*, после этого кликните по *Save*. Светодиод выключится после успешной операции записи.

12.7. Примеры скриптов Modbus Slave

Добавьте следующий код в *Common functions*

```
1.  -- прокси modbus
2.  mbproxy ={
3.  -- список поддерживаемых функций
4.    functions ={
5.  'readdo',
6.  'readcoils',
7.  'readdi',
8.  'readdiscreteinputs',
9.  'readao',
10. 'readregisters',
11. 'readai',
12. 'readinputregisters',
13. 'writebits',
14. 'writemultiplebits',
15. 'writeregisters',
16. 'writemultipleregisters',
17. 'reportslaveid',
18. 'getcoils',
19. 'getdiscreteinputs',
20. 'getinputregisters',
21. 'getregisters',
22. 'setcoils',
23. 'setdiscreteinputs',
24. 'setinputregisters',
25. 'setregisters',
26. },
27. -- инициализация нового соединения
28.   new =function()
29. require('rpc')
30. local mb =setmetatable({}, { __index = mbproxy })
31.
32.     mb.slaveid =0
33.     mb.rpc = rpc.client('127.0.0.1', 28002, 'mbproxy')
34.
35. for _, fn in ipairs(mbproxy.functions)do
36.       mb[ fn ]=function(self, ...)
37. return mb:request(fn, ...)
38. end
39. end
40.
41. return mb
42. end
43. }
44.
45. -- установка локального slave id
46. function mbproxy:setslave(slaveid)
47.   self.slaveid = slaveid
48. end
49.
50. -- send rpc request for a spefic function
51. function mbproxy:request(fn, ...)
52. local res, err = self.rpc:request({
53.     fn = fn,
54.     params ={ ... },
55.     slaveid = self.slaveid or0,
56. })
57.
58. -- ошибка запроса
59. if err then
```

```
60. return nil, err
61. -- успешный запрос
62. else
63. -- ответ с ошибкой
64. if res[ 1 ]==nil then
65. return nil, res[2]
66. -- нормальный ответ
67. else
68. return unpack(res)
69. end
70. end
71. end
```

Конфигурация обработчика (резидентного скрипта с задержкой 0)

1. *mb:open()*
Открывает TCP-соединение Modbus;

2. *mb:setslave(10)*
Задаёт адрес slave-устройства;

3. *mb:setmapping(10, 10, 10, 10)*
Задаёт количество coils, discrete inputs, holding registers и input registers;

4.*mb:setwritecoilcb(function(coil, value)...*
Callback-функция, которая выполняется при записи каждого coil;

5. *mb:setwriteregistercb(function(coil, value)...*
Callback-функция, которая выполняется при записи каждого register.

Пример скрипта обработчика:

```
1.    -- инициализация modbus
2.    if not mb then
3.    require('luamodbus')
4.      mb = luamodbus.tcp()
5.      mb:open()
6.
7.    -- инициализация slave-хранилища для coils, discrete inputs, holding registers и input
      registers
8.      mb:setmapping(10, 10, 10, 10)
9.
10.   -- callback записи coil
11.     mb:setwritecoilcb(function(coil, value)
12.   if coil == 0 then
13.         grp.write('1/1/1', value, dt.bool)
14.   else
15.         alert('coil: %d = %s', coil, tostring(value))
16.   end
17.   end)
18.
19.   -- callback записи register
20.     mb:setwriteregistercb(function(register, value)
21.   if register == 0 then
22.   -- отправка значения в диапазоне 0..100
23.         grp.write('4/1/5', math.min(100, value), dt.scale)
24.   else
```

```
25.         alert('register: %d = %d', register, value)
26.   end
27.   end
28.   end
29.
30.   -- инициализация серверной части
31.   if not server then
32.   require('rpc')
33.
34.   -- обработчик входящих данных
35.   local handler = function(request)
36.   local fn, res
37.
38.       fn =tostring(request.fn)
39.
40.   if not mb[ fn ]then
41.   return{nil, 'unknown function ' .. fn }
42.   end
43.
44.   if type(request.params)=='table' then
45.   table.insert(request.params, 1, mb)
46.         res ={ mb[ fn ](unpack(request.params))}
47.   else
48.         res ={ mb[ fn ](mb)}
49.   end
50.
51.   return res
52.   end
53.
54.     server = rpc.server('127.0.0.1', 28002, 'mbproxy', handler, 0.01)
55.   end
56.
57.   mb:handleslave()
58.   server:step()
```

Пример: событийный скрипт, изменяющий modbus slave coil (адрес 0)

Должен быть сопоставлен групповому адресу с двоичным значением.

```
1.  value = event.getvalue()
2.  mb = mbproxy.new()
3.  mb:setcoils(0, value)
```

Пример: событийный скрипт, изменяющий modbus slave register (адрес 5)

Должен быть сопоставлен групповому адресу со значением (0..100).

```
1.  value = event.getvalue()
2.  mb = mbproxy.new()
3.  mb:setregisters(5, value)
```

Соединение LM с PLC через Modbus TCP

sleep time = 0. Поддерживает как coils только бинарные объекты, как registers – 1-байтные или 2-байтные целочисленные объекты. Количество coils и registers не ограничено, сопоставление

объектов возможно путём заполнения coils, registers и таблиц regdt.

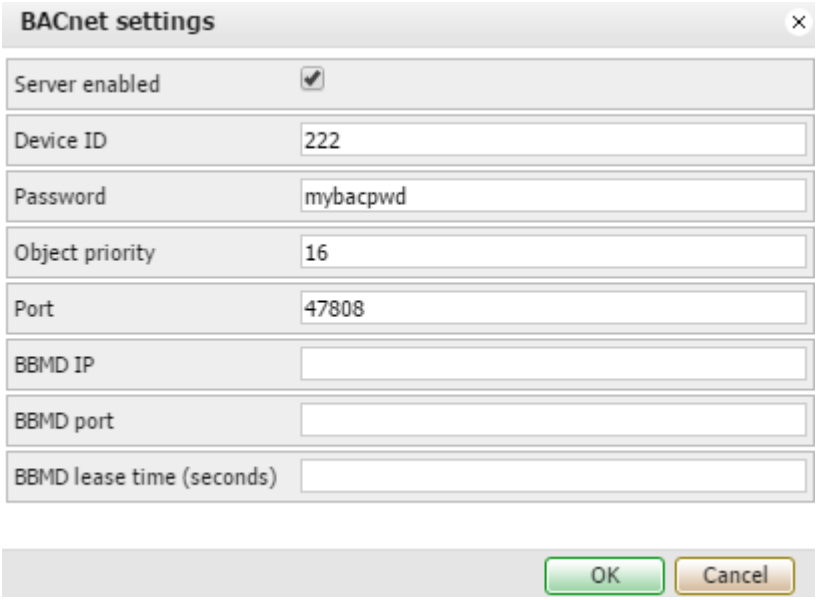
```
1.  if not mb then
2.    require('genohm-scada.eibdgm')
3.    require('luamodbus')
4.
5.    -- список сопоставления coil, начинается с 0
6.    coils = { '1/1/1', '1/1/2' }
7.
8.    -- список сопоставления register, начинается с 0
9.    registers = { '2/2/2', '3/3/3' }
10.
11.   -- список типов данных register, число элементов должно совпадать с таблицей registers
12.   regdt = { dt.int8, dt.uint16 }
13.
14.   -- callback групповой записи knx
15.   function knxgroupwrite(event)
16.     local value
17.
18.     -- попытка найти подходящий coil
19.     for id, addr in ipairs(coils) do
20.       if event.dst == addr then
21.         value = knxdatatype.decode(event.datahex, dt.bool)
22.         mb:setcoils(id - 1, value)
23.       end
24.     end
25.
26.     -- попытка найти подходящий register
27.     for id, addr in ipairs(registers) do
28.       if event.dst == addr then
29.         value = knxdatatype.decode(event.datahex, regdt[ id ])
30.         mb:setregisters(id - 1, value)
31.       end
32.     end
33.   end
34.
35.   -- callback записи coil
36.   function mbwritecoils(coil, value)
37.     local addr = coils[ coil + 1 ]
38.     if addr then
39.       grp.write(addr, value, dt.bool)
40.     end
41.   end
42.
43.   -- callback записи register
44.   function mbwriteregisters(register, value)
45.     local addr = registers[ register + 1 ]
46.     if addr then
47.       grp.write(addr, value, regdt[ register + 1])
48.     end
49.   end
50.
51.   -- проверка knx, обращение к групповым записям
52.   knxclient = eibdgm:new({ timeout = 0.1 })
53.   knxclient:sethandler('groupwrite', knxgroupwrite)
54.
55.   -- modbus slave, прослушивание интерфейсов и порта по умолчанию 502
56.   mb = luamodbus.tcp()
57.   mb:open('0.0.0.0', 502)
58.
59.   -- указание slave id (не обязательно)
60.   -- mb:setslave(1)
61.
62.   mb:setreceivetimeout(0.1)
63.   mb:setmapping(#coils, 0, #registers, 0)
64.
```

```
65.   -- инициализация coils
66.   for id, addr in ipairs(coils) do
67.     value = grp.getvalue(addr)
68.     mb:setcoils(id - 1, value)
69.   end
70.
71.   -- инициализация registers
72.   for id, addr in ipairs(registers) do
73.     value = grp.getvalue(addr)
74.     mb:setregisters(id - 1, value)
75.   end
76.
77.   -- задание коллбэков для записи coil и register
78.   mb:setwritecoilcb(mbwritecoils)
79.   mb:setwriteregistercb(mbwriteregisters)
80. end
81.
82. -- обращение к modbus и knx
83. mb:handleslave()
84. knxclient:step()
```

13. Соединение BACnet IP с LM

13.1. Режим сервера BACnet: прозрачная передача данных в сеть BACnet

Конфигурацию сервера BACnet можно настроить в *Sys Config* → *Network* → *BACnet Settings*.

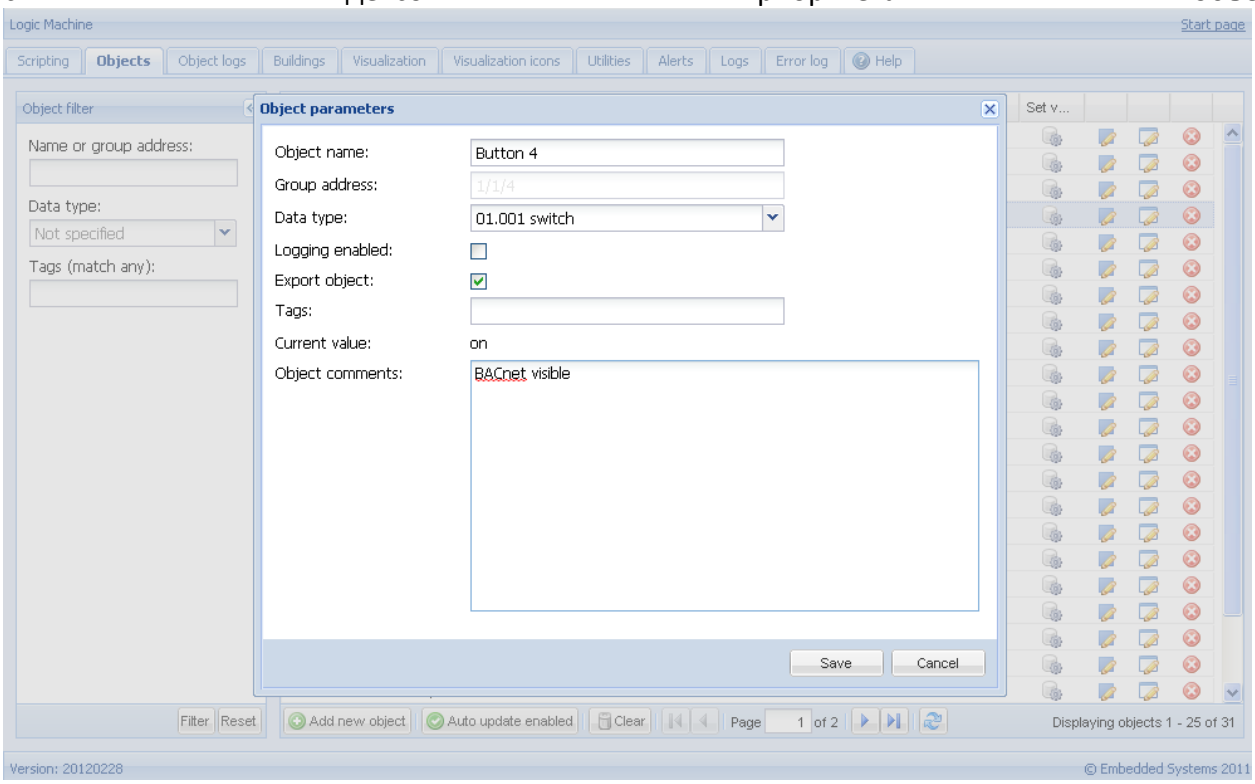


- ***Server enabled*** – включить/отключить сервер BACnet;
- ***Device ID*** – ID устройства в сети BACnet;
- ***Password*** – пароль устройства;
- ***Object priority*** – приоритет объекта;
- ***Port*** – номер порта;
- ***BBMD IP*** – IP-адрес BACnet-роутера. После установки IP-адреса и порта LM начнёт вести себя как внешнее устройство и попробует зарегистрироваться на роутере

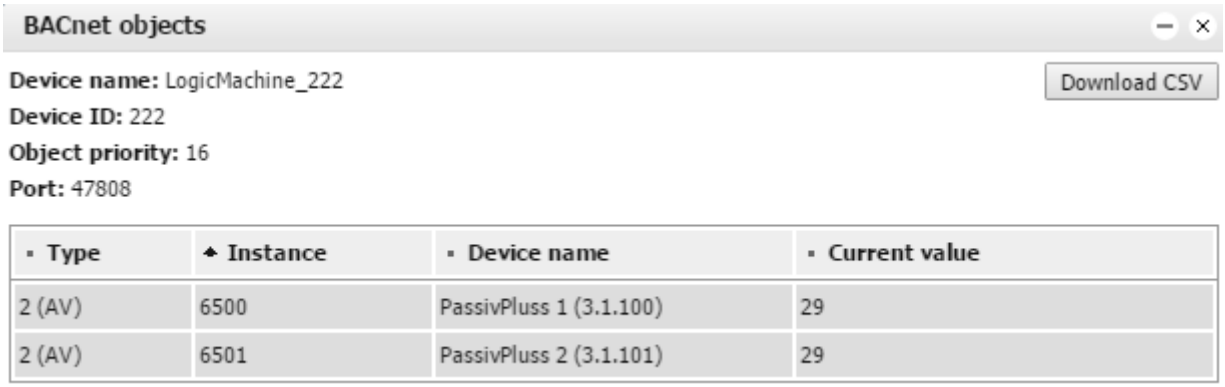
BACnet;

- ***BBMD port*** – порт BACnet-роутера;
- ***BBMD lease time (seconds)*** – интервал между повторными регистрациями.

Чтобы сделать объекты KNX/EIB доступными для чтения и записи в BACnet, отметьте в их настройках поле *Export object*. Бинарные объекты отобразятся как двоичные значения, остальные числовые объекты – как аналоговые. Другие типы данных на данный момент не поддерживаются. Запись в шину KNX перепишет значение массива приоритета в соответствии с индексом приоритета объекта.



В *Network → BACnet objects* указаны отмеченные для экспорта объекты, которые посылаются в сеть BACnet.



BACnet objects

Device name: LogicMachine_222

Device ID: 222

Object priority: 16

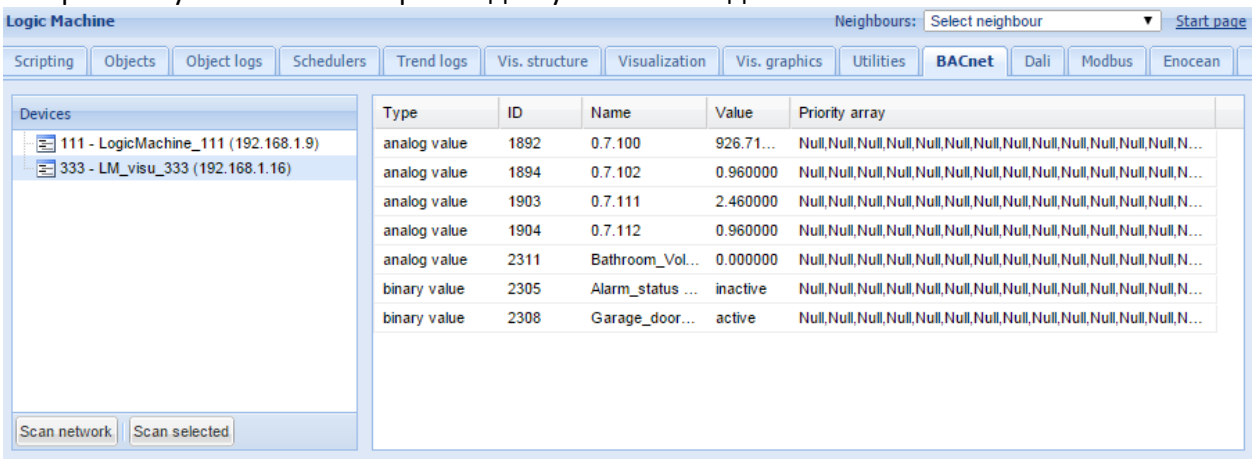
Port: 47808

Download CSV

Type	Instance	Device name	Current value
2 (AV)	6500	PassivPluss 1 (3.1.100)	29
2 (AV)	6501	PassivPluss 2 (3.1.101)	29

13.2. Режим клиента BACnet

Обычно этот режим используется, к примеру, для соединения LogicMachine с системами VRV по протоколу BACnet IP. Настройки доступны на вкладке *BACnet*.



Нажатие на кнопку *Scan Network* показывает список серверных устройств BACnet в сети. С помощью кнопки *Scan Selected* можно повторно просканировать серверные устройства BACnet для поиска необходимых объектов.

Сопоставление объектов KNX в настоящее время выполняется с помощью скриптов.

Перед использованием любой функции BACnet необходимо подключить библиотеку:
require('bacnet')

Чтение текущего значения двоичного или аналогового объекта:
bacnet.readvalue(device_id, object_type, object_id)

Чтение двоичного объекта в переменную:
value = bacnet.readvalue(127001, 'binary value', 2305)

Чтение аналогового объекта в переменную:
value = bacnet.readvalue(127001, 'analog value', 2306)

Запись нового значения в массив приоритетов двоичного или аналогового объекта:
bacnet.write = function(device_id, object_type, object_id, value, priority)

Значение может быть nil, boolean, числом или строкой чисел.
Параметр приоритета необязателен. По умолчанию используется самый низкий приоритет.

Запись значения двоичного объекта:
bacnet.write(127001, 'binary value', 2305, true)

Запись значения аналогового объекта:
bacnet.write(127001, 'analog value', 2306, 22.5)

Запись значения двоичного объекта с приоритетом 12:
bacnet.write(127001, 'binary value', 2305, true, 12)

Запись значения аналогового объекта с приоритетом 10:
bacnet.write(127001, 'analog value', 2306, 22.5, 10)

Очистка значения двоичного объекта с приоритетом 12:
bacnet.write(127001, 'binary value', 2305, nil, 12)

14. Соединение DMX с LM

Поддержка протокола DMX осуществляется через порт RS-485.

<u>Функция DMX</u>

Добавьте следующую пользовательскую библиотеку в *Scripting → User libraries*.

```
1. local luadmx = require('luadmx')
2. module('DMX', package.seeall)
3.
4. local DMX = {}
5.
6. -- параметры по умолчанию
7. local defaults = {
8.   -- ключ хранилища
9.   skey = 'dmx_line_1',
10.                                                                    -- порт
   RS-485
11.                                                                    port =
   '/dev/RS485',
12.                                                                    --
   количество вызовов в секунду
13.
   resolution = 20,
14.                                                                    -- общее
   кол-во используемых каналов
15.                                                                    channels
   = 3,
16.                                                                    -- время
   перехода в секундах, не включает в себя время перевода DMX
17.
   transition = 2,
18.                                                                 }
19.
20.                                                                 -- сеттер
   значений
21.                                                                 function
   set(chan, val, key)
22.                                                                   key =
   key or defaults.skey
23.                                                                   chan =
   tonumber(chan) or 0
24.                                                                   val =
   tonumber(val) or -1
25.
```

```
26.                                                                    --
   проверка и утверждение номера канала и значения
27.                                                                    if chan
   >= 1 and chan <= 512 and val >= 0 and val <= 255 then
28.
   storage.exec('lset', key, chan - 1, val)
29.                                                                    end
30.                                                                  end
31.
32.                                                                  -- геттер
   значения
33.                                                                  function
   get(chan, key)
34.                                                                    local
   res, val
35.                                                                    key =
   key or defaults.skey
36.                                                                    chan =
   tonumber(chan) or 0
37.
38.                                                                    --
   проверка и утверждение номера канала и значения
39.                                                                    if chan
   >= 1 and chan <= 512 then
40.                                                                      res =
   storage.exec('lrange', key, chan - 1, chan - 1)
41.                                                                      if
   type(res) == 'table' then
42.                                                                        val
   = tonumber(res[ 1 ])
43.                                                                      end
44.                                                                    end
45.
46.                                                                    return
   val
47.                                                                  end
48.
49.                                                                  --
   инициализация DMX, возвращает новый объект DMX
50.                                                                  function
   init(params)
51.                                                                    local n,
   k, v, _
52.
53.                                                                    --
   создание мета-таблицы и задание пользовательских параметров
54.                                                                    n =
   setmetatable({}, { __index = DMX })
55.                                                                    n.params
   = params or {}
56.
57.                                                                    _,
   n.conn = pcall(require('redis').connect)
58.
59.                                                                    --
   объединение параметров, заданных пользователем
60.                                                                    for k, v
   in pairs(defaults) do
61.                                                                      if
   n.params[ k ] == nil then
62.
   n.params[ k ] = v
```

```
63.                                                                 end
64.                                                               end
65.
66.
   n:reset()
67.
68.                                                               return n
69.                                                             end
70.
71.                                                             function
   DMX:reset()
72.                                                               local
   err, chan, params
73.
74.                                                               params =
   self.params
75.                                                               self.dm,
   err = luadmx.open(params.port)
76.
77.                                                               --
   ошибка при открытии
78.                                                               if err
   then
79.
   os.sleep(1)
80.
   error(err)
81.                                                               end
82.
83.                                                               --
   установка количества каналов
84.
   self.dm:setcount(params.channels)
85.
86.                                                               --
   количество тиков транзакций
87.
   self.ticks = math.max(1, params.transition * params.resolution)
88.
89.                                                               --
   расчёт времени задержки
90.
   self.sleep = 1 / params.resolution
91.
92.                                                               -- сброс
   карты каналов
93.
   self.channels = {}
94.
95.                                                               -- карта
   значений пустого канаа
96.
   self.conn:ltrim(params.skey, 1, 0)
97.
98.                                                               -- карта
   полного канала
99.                                                               for chan
   = 1, params.channels do
100.
   self.channels[ chan ] = { current = 0, target = 0, ticks = 0 }
101.
102.                                                                --
```

```
выключение по умолчанию
103.
self.conn:lpush(params.skey, 0)
104.
self.dm:setchannel(chan, 0)
105.                                                                    end
106.                                                                  end
107.
108.                                                                  --
получение новых значений
109.                                                                  function
DMX:getvalues()
110.                                                                    local
max, channels, ticks, values, val
111.
112.                                                                    max =
self.params.channels
113.                                                                    channels
= self.channels
114.                                                                    ticks =
self.ticks
115.                                                                    values =
self.conn:lrange(self.params.skey, 0, max - 1) or {}
116.
117.                                                                    --
проверка наличия новых значений для каждого канала
118.                                                                    for chan
= 1, max do
119.                                                                      val =
tonumber(values[ chan ]) or 0
120.
121.                                                                      --
целевое значение отличается; задание транзакции
122.                                                                      if val
~= channels[ chan ].target then
123.
channels[ chan ].target = val
124.
channels[ chan ].delta = (channels[ chan ].target - channels[ chan ].current)
/ ticks
125.
channels[ chan ].ticks = ticks
126.                                                                      end
127.                                                                    end
128.                                                                  end
129.
130.                                                                  --
обработчик основного цикла
131.                                                                  function
DMX:run()
132.
self:getvalues()
133.
134.                                                                    -- цикл
перехода
135.                                                                    for i =
1, self.params.resolution do
136.
self:step()
137.
self.dm:send()
138.
```

```
os.sleep(self.sleep)
139.                                                                    end
140.                                                                  end
141.
142.                                                                  -- шаг
единичной транзакции
143.                                                                  function
DMX:step()
144.                                                                    local
chan, channels, t
145.
146.                                                                    channels
= self.channels
147.
148.                                                                    --
переход для каждого канала
149.                                                                    for chan
= 1, self.params.channels do
150.                                                                      t =
channels[ chan ].ticks
151.
152.                                                                      --
переход активен
153.                                                                      if t >
0 then
154.                                                                        t =
t - 1
155.
156.
channels[ chan ].current = channels[ chan ].target - channels[ chan ].delta *
t
157.
channels[ chan ].ticks = t
158.
159.
self.dm:setchannel(chan, channels[ chan ].current)
160.                                                                      end
161.                                                                    end
162.                                                                  end
```

Скрипт обработчика DMX

Добавьте следующий резидентный скрипт с задержкой 0; в случае необходимости измените порт и количество каналов.

```
1. if not dmxhandler then
2.   require('user.dmx')
3.   dmxhandler = DMX.init({
4.     port = '/dev/RS485', -- имя порта RS-485
5.     channels = 8, -- количество используемых каналов DMX
6.     transition = 2, -- время перехода в секундах
7.   })
8. end
9.
10.                                                                  dmxhandler
:run()
```

Задатчик (используется в других скриптах)

```
DMX.set(channel, value)
```

Привязка объектов DMX

Создайте объекты с тегом DMX, где последняя часть группового адреса – адрес DMX (начинается с 1). Создайте событийный скрипт, срабатывающий на тег DMX.

```
1. require('user.dmx')
2. -- получение ID как последней части группового адреса (x/y/ID)
3. id = tonumber(event.dst:split('/')[3])
4. -- получение значения события (в пересчёте на байты)
5. value = event.getvalue()
6. -- конвертация из [0..100] в [0..255]
7. value = math.floor(value * 2.55)
8. -- задание значения ID канала
9. DMX.set(id, value)
```

Пример заранее заданной сцены:

Следующий пример нужно добавить в резидентный скрипт. Время задержки определяет время поддержания сцены (минимум 1 секунда).

```
1. if not scenes then
2.   -- 3-канальная сцена
3.   scenes = {
4.     { 255, 0, 0 },
5.     { 0, 255, 0 },
6.     { 0, 0, 255 },
7.     { 255, 255, 0 },
8.     { 0, 255, 255 },
9.     { 255, 0, 255 },
10.                                                                { 255,
   255, 255 },
11.                                                              }
12.
13.                                                              current
   = 1
14.                                                            end
15.
16.                                                            -- задание
   текущих значений сцены
17.                                                            scene =
   scenes[ current ]
18.                                                            for i, v
   in ipairs(scene) do
19.
   DMX.set(i, v)
20.                                                            end
21.
22.                                                            --
   переключение на следующую сцену
23.                                                            current =
   current + 1
24.                                                            if current
   > #scenes then
25.                                                              current
   = 1
26.                                                            end
```

Пример случайной сцены:

Следующий пример нужно добавить в резидентный скрипт. Время задержки определяет время поддержания сцены (минимум 1 секунда).

```
1. -- количество используемых шагов, например, 3 шага = { 0, 127, 255 }
2. steps = 5
3. -- количество задаваемых каналов
4. channels = 3
5. -- номер первого канала
6. offset = 1
7.
8. for i = offset, channels do
9.   v = math.random(0, (steps - 1)) * 255 / (steps - 1)
10.
   DMX.set(i, math.floor(v))
11.                                                                end
```

15. Соединение 3G модема и LM

Благодаря встроенному драйверу LogicMachine поддерживает множество стандартных 3G-модемов (в том числе от Huawei и других поставщиков).
В данный момент их можно использовать только для SMS-оповещений, получения и отправки команд. Модем должен быть подключен к USB. Рекомендуется использовать для модема внешнее питание 5V, так как стандарт USB 2.0 выдаёт на USB только 0.75A, а некоторым модемам необходимо 2A. Из-за такого несоответствия модему будет не хватать питания, и он может выключиться.

Список поддерживаемых 3G-модемов приведён здесь:
http://openrb.com/wp-content/uploads/2015/12/3G_USB_device_reference.txt

Сначала нужно понизить скорость модема. Для этого требуется добавить в скрипт запуска системы следующий код:

```
1. os.execute('echo 1 > /sys/bus/platform/devices/ci_hdrc.0/force_full_speed')
2. os.execute('echo 1 > /sys/bus/platform/devices/ci_hdrc.1/force_full_speed')
3. os.execute('usbreset /dev/bus/usb/001/001')
```

После этого вам необходимо добавить обработчик SMS-сообщений – резидентный скрипт с задержкой 0.

Примечание! “Белый список” телефонных номеров и ПИН-код SIM-карты в скрипте нужно изменить на свои значения.

```
1. -- инициализация
2. ifnot modem then
3. -- “белый список” номеров; сообщения с других номеров будут игнорироваться
4.   numbers = {'1234567890', '0123456789'}
5. -- замените 0000 на пин-код вашей SIM-карты; чтобы отключить проверку пин-кода, просто
   удалите эту строку
6. pincode ='0000'
7. -- порт коммуникации модема, для Huawei E173 значение равно ttyUSB2
8.   comport = 'ttyUSB2'
9. -- открытие последовательного порта
10.   modem = AT:init('/dev/' .. comport)
11.-- парсер команд
12.   parser = function(cmd, sender)
13.local find, pos, name, mode, offset, value, jvalue, obj
14.cmd = cmd:trim()
15.     mode = cmd:sub(1, 1):upper()
16.if mode == 'W'or mode == 'R'then
17.cmd = cmd:sub(3):trim()
18.-- парсер имени/адреса объекта
19.       find = cmd:sub(1, 1) == '"'and'"'or' '
20.       offset = find == '"'and 1 or0
21.-- дополнение пробелом в режиме чтения
22.if mode == 'R' and find == ' ' then
23.cmd = cmd .. ' '
24.end
25.-- поиск имени
26.pos = cmd:find(find, 1 + offset, true)
27.-- имя не найдено, стоп
28.if not pos then
29.return false
```

```
30. end
31. -- имя получено
32.       name = cmd:sub(1 + offset, pos - offset):trim()
33. if mode == 'W' then
34.         value = cmd:sub(pos + offset):trim()
35. if not value then
36. return false
37. end
38. -- попытка декодирования значения
39. jvalue = json.pdecode(value)
40.         value = jvalue ~= nil and jvalue or value
41. -- отправка в шину
42. grp.write(name, value)
43. -- запрос на чтение
44. else
45. obj = grp.find(name)
46. -- отправка запроса на чтение и ожидание обновления
47. if obj then
48. obj:read()
49. os.sleep(1)
50. -- чтение нового значения
51.           value = grp.getvalue(name)
52. -- значение получено, отправка ответа
53. if value ~= nil then
54. jvalue = json.pencode(value)
55. if obj.name then
56.               name = string.format('%s (%s)', obj.name, obj.address)
57. end
58. cmd = string.format('Value of %s is %s', name, jvalue)
59. modem:sendsms(sender, cmd)
60. end
61. end
62. end
63. end
64. end
65. -- обработчик входящих СМС
66.   handler = function(sms)
67.     alert('incoming sms from %s (%s)', sms.sender, sms.data)
68. -- СМС от номера из “белого списка”, вызов парсера
69. if table.contains(numbers, sms.sender) then
70.       parser(sms.data, sms.sender)
71. end
72. end
73. -- задание обработчика СМС
74. modem:setsmshandler(handler)
75. -- отправка пин-кода (если задан)
76. if pincode then
77. modem:send('AT+CPIN=' .. pincode)
78. end
79. -- перевод в режим pdu
80. modem:send('AT+CMGF=0')
81. -- включение смс-уведомлений
82. modem:send('AT+CNMI=1,1,0,0,0')
83.   alert('SMS handler started')
84. end
85. modem:run()
```

<u>Синтаксис команд:</u>

a. Запись в шину:

W ALIAS VALUE

b. Чтение из шины:

R ALIAS

При чтении скрипт отправит ответное SMS-сообщение с текущим значением объекта.

ALIAS:
a. Групповой адрес (1/1/1);
b. Название (Obj1). Если название содержит пробелы, то оно должно быть заключено в двойные кавычки ("Room Temperature").

Примечание:
a. Имя и тип данных объекта необходимо задать на вкладке *Objects*, иначе скрипт не сможет работать с этим объектом.
b. В SMS-сообщении поддерживаются только ASCII-символы.

15.1. Примеры

Запись в бинарный объект (SMS, включающее свет на кухне):

W 1/1/1 true

Запись в числовой объект (SMS, задающее красному светодиоду значение 67%):

W LED1Red 67

Запись температуры (числа с плавающей точкой) (SMS, задающее температуру уставки 22.5 градуса):

W “Room Setpoint” 22.5

Чтение (SMS для получения значения охранной сигнализации):

R 2/1/1

15.2. Отправка SMS-сообщений на заданный номер после срабатывания group-read или group-write

Задача: Предположим, что у нас есть событийный скрипт, который срабатывает при чтении и записи объекта 1/1/1. Мы хотим отправлять SMS-сообщение с актуальным статусом объекта на номера 23335555 и 23335556.

```
1. require('socket')
2.
3. client = socket.udp()
4.
5. -- номер, на который отправится SMS, должен быть определён с самого начала
6. local msg = '23335555 1/1/1 changes its value to: ' .. tonumber(event.datahex)
7. client:sendto(msg, '127.0.0.1', 12535)
8.
9. msg = '23335556 1/1/1 changes its value to: ' .. tonumber(event.datahex)
```

```
10. client:sendto(msg, '127.0.0.1', 12535)
```

15.3. Отправка SMS-сообщений без 3G-модема

Как отправить SMS с LogicMachine на мобильный телефон без внешнего 3G модема?

Можно использовать сервис Twilio, который предоставляет возможность бесплатного обмена сообщениями во время тестового периода; после его окончания это стоит $0.01. Единственный недостаток – сервис использует интернет-соединение для отправки сообщений на сервер Twilio (не через GSM, как в случае 3G модема).

Twilio-аккаунт

ID и Token, необходимые для реализации примера ниже, даются при регистрации на Twilio. При создании аккаунта убедитесь, что вы ввели правильный список номеров, или свяжитесь с нами для получения готового примера с данными нашего аккаунта.

Функция

Добавьте в *Scripting –> Common functions* следующую пользовательскую функцию:

```
1. function sms(id, token, from, to, body)
2.    local escape = require('socket.url').escape
3.    local request = require('ssl.https').request
4.    local url = string.format('https://%s:%s@api.twilio.com/2010-04-
   01/Accounts/%s/Messages.json', id, token, id)
5.    local body = string.format('From=%s&To=%s&Body=%s', escape(from),
   escape(to), escape(body))
6.
7.    return request(url, body)
8. end
```

Событийный скрипт

Добавьте событийный скрипт для необходимого объекта (в нашем примере – 1/1/2):

```
1. value = event.getvalue()
2.
3. from_nr = '+37112345679' -- номер отправителя
4. to_nr = '+37112345678' -- номер получателя
5. id_nr = 'ACe56f5' -- ваш ID
6. token_nr = '598c6ff' -- ваш token
7. sms(id_nr, token_nr, from_nr, to_nr, 'The value for 1/1/2 has changed
   to'..tostring(value))
```

16. Связь с портами RS232 / RS485

Ниже перечислены имена портов для разных версий LogicMachine.

LM4

GND	
RS485 A	RS485-1
RS485 B	
GND	
RS485 A	RS485-2
RS485 B	
GND	
RS485 A	RS485-3
RS485 B	

Reactor, LM5 - первая версия

GND	
RS485 A	RS485-1
RS485 B	
GND	
RS485 A	RS485-2
RS485 B	

Reactor V2

GND	
RS485 A	RS485
RS485 B	

Reactor V3

GND	
RS485 A	RS485
RS485 B	

LM5L, LM5-, LM5-RIO, LM5-RIOE**

RS485 A	RS485-1
RS485 B	
GND	
RS485 A	RS485-2
RS485 B	
GND	
TX	RS232
RX	
GND	

Примечание! В устройствах серии LM5 один порт – RS-485, а другой может работать и как RS-485, и как RS-232. Работать будет тот, который был открыт последним.

Следующая команда активирует второй порт RS-485:

```
port = serial.open('/dev/RS485-2', { baudrate = 115200, parity = 'even', duplex = 'half'
})
```

Следующая команда активирует порт RS-232:

```
port = serial.open('/dev/RS232', { baudrate = 9600, parity = 'even', duplex = 'full' })
```

Функции

Перед вызовом serial-функций добавьте следующую библиотеку:

```
require('serial')
```

Команда ниже открывает заданный порт и возвращает дескриптор порта или, в случае ошибки, nil и сообщение об ошибке:

```
port, err = serial.open(device, params)
```

Параметры:

- **device** – название порта, обязательный;
- **params** – таблица с параметрами, необязательный:
  - **baudrate** – 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400;
  - **parity** – "none", "even", "odd";
  - **databits** – 5, 6, 7, 8;
  - **stopbits** – 1, 2;
  - **duplex** – "full", "half" (примечание: "half" необходим для RS-485).

Данная команда считывает указанное количество байтов, выполнение блокируется до завершения чтения:

```
res, err = port:read(bytes)
```

Параметр:

- **bytes** – количество байтов для чтения.

Данная команда считывает значение, пока не наступит тайм-аут или не будет получено указанное количество байтов, в зависимости от того, что произойдёт раньше.
Возвращает данные и количество прочитанных байтов или, в случае ошибки, nil и сообщение об ошибке.

```
res, err = port:read(bytes, timeout)
```

Параметры:

- **bytes** – количество байтов для чтения;
- **timeout** – максимальное время ожидание окончания чтения; минимальное значение и шаг – 0.1 секунда.

Команда, сбрасывающая все прочитанные / неотправленные байты:

```
port:flush()
```

Команда, закрывающая последовательный порт, после чего другие функции порта становится нельзя вызвать:

```
port:close()
```

Примеры:

Запись в порт:

```
port:write('test data')
```

Чтение с блокировкой (скрипт блокирует чтение до 10 символов):

```
data = port:read(10)
```

Чтение до тайм-аута (скрипт будет ждать 10 символов в течение 20 секунд):

```
data = port:read(10, 20)
```

Закрытие порта:

```
port:close()
```

Резидентный скрипт, эхо-тест RS-485:

```
-- открытие порта при первом вызове
if not port then
require('serial')
port = serial.open('/dev/RS485-1', { baudrate = 9600, parity = 'even', duplex = 'half' })
port:flush()
end

-- порт готов
if port then
  -- чтение одного байта
char = port:read(1, 1)
  -- возвращение данных назад, если чтение успешно
if char then
port:write(char)
end
end
```

Запись шестнадцатеричного значения в RS-485:

```
require('serial')
port = serial.open('/dev/RS485-1', {
baudrate = 4800,
parity = 'none',
duplex = 'half'
})

cmd = string.char(0xAB, 0xF1, 0xFF, 0xFF, 0xFF, 0xFF, 0xBE, 0xD1, 0x01, 0xFE, 0xFF, 0xFF,
0x0A, 0x24)
cmd = 'ABF1FFFFFFFFBED101FEFFFF0A24'

port:write(cmd)
```

Проверьте, какой вариант cmd работает в вашем случае: это могут быть либо шестнадцатеричные читаемые данные, либо шестнадцатеричное представление двоичных данных. Вам также может потребоваться изменение конфигурации чётности: http://openrb.com/docs/serial.htm

17. Интеграция Bluetooth 4.0

Bluetooth можно интегрировать с помощью USB-Bluetooth-адаптера.

Некоторые из поддерживаемых адаптеров:

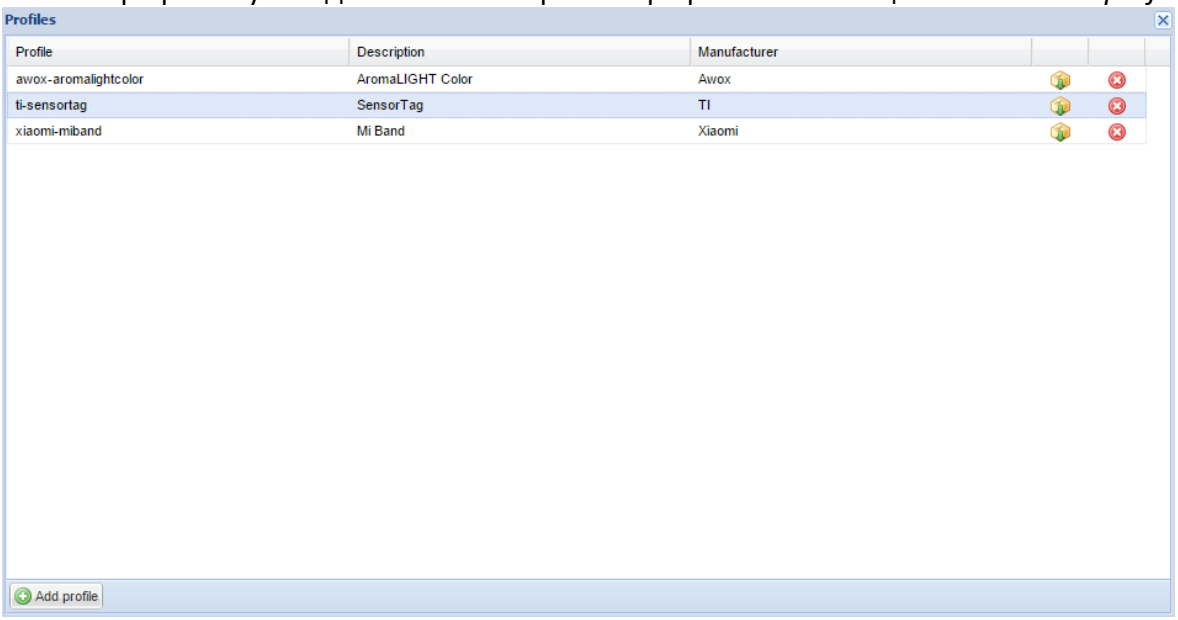
- Broadcom BCM20702A0;
- Trust 18187;
- Belkin F8T065bf;
- Plugable USB Bluetooth 4.0;
- Laird BT820.

Настройки Bluetooth находятся на вкладке *LogicMachine → BLE*.

Можно добавить поддержку любого устройства BLE, если будет предоставлен протокол связи, который не изменится в будущей версии программного обеспечения BLE.

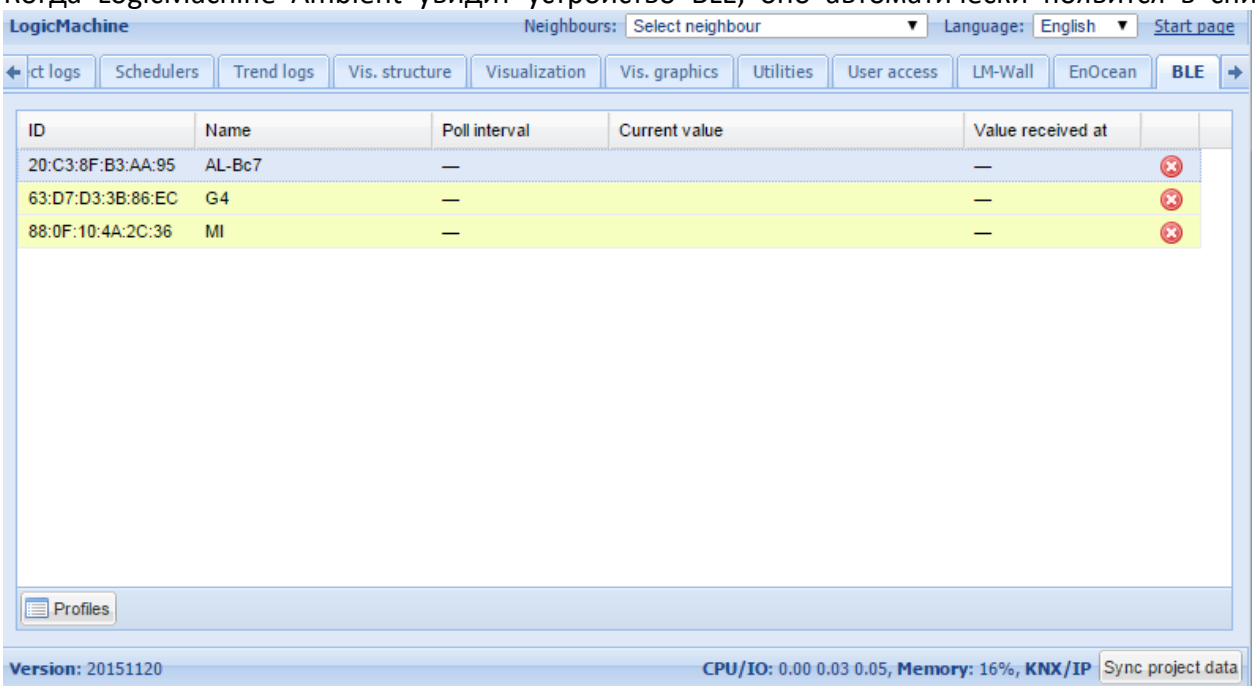
Профили

Список поддерживаемых BLE-устройств открывается нажатием на кнопку *Profiles*. Для создания нового профиля нужно добавить *.lua-файл с профилем с помощью кнопки *Add profile*.

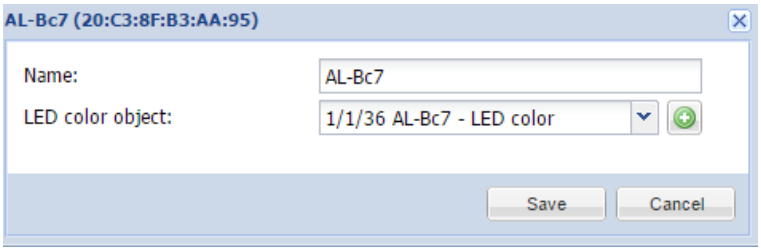


Сопоставление функциональности с адресами групп KNX

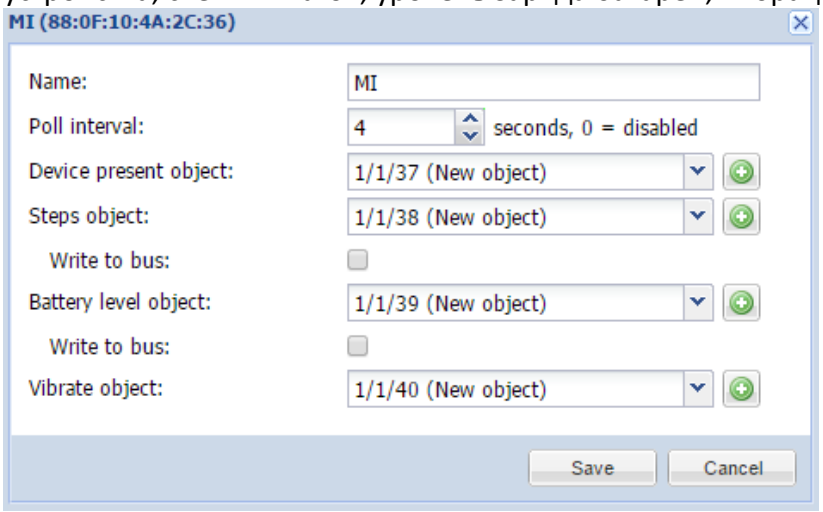
Когда LogicMachine Ambient увидит устройство BLE, оно автоматически появится в списке.



По нажатию на определённое устройство можно сопоставить его функциональность с групповыми адресами KNX.
Для лампы AWOX AromLight Color BLE можно сопоставить с KNX цветной светодиодный объект.



Для браслетов Xiaomi Mi Band существуют следующие объекты отображения: наличие устройства, счётчик шагов, уровень заряда батареи, вибрация браслета.



Пример:

Датчик сердцебиения часов Alpha MIO BLE связан с соответствующим объектом. Приведённый ниже событийный скрипт следит за сердцебиением: включает вентиляцию при пульсе выше 80 и выключает при пульсе ниже 80.

```
18.   value = event.getvalue()
19.   if value > 80 then
20.     grp.write('2/2/2', true)
21.   else
22.     grp.write('2/2/2', false)
23.   end
```

20. SIP-сервер на LogicMachine

Задача: как объединить системы входа SIP с проектом автоматизации зданий? LogicMachine обладает SIP-регистратором, который может отправлять SIP-запросы конечным SIP-клиентам. Например, можно установить на сенсорные устройства, используемые для управления визуализацией, клиентское приложение Linphone SIP. Если система входа SIP даст запрос, LogicMachine переадресует этот запрос соответствующему клиенту/получателю SIP. На устройстве клиента появится окно, в котором можно будет принять или отклонить вызов. При ответе на вызов система передаст на устройство клиента видео и аудио. Когда вызов будет завершён, приложение Linphone перейдет в фоновый режим.

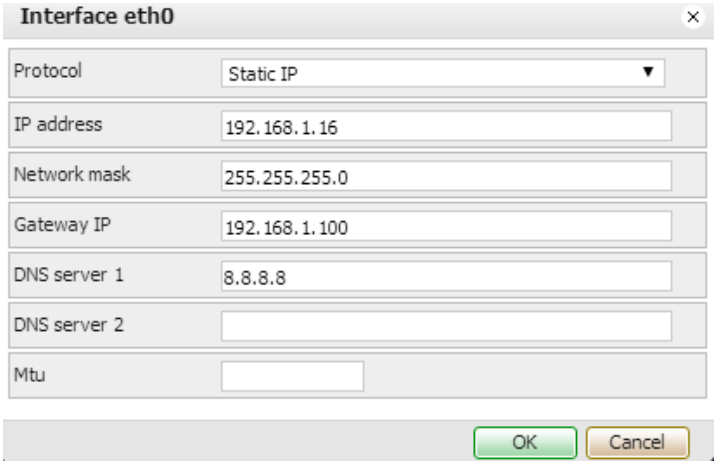
Установка пакета SIP на LM:

Добавьте следующий резидентный скрипт с задержкой 60, запустите его один раз:

```
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/terminfo_5.7-5_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/libncurses_5.7-5_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/libreadline_5.2-2_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-maxfwd_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-registrar_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-rr_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-sl_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-tm_3.3.7-1_mxs.ipk')
os.execute('opkg --force-depends install http://dl.openrb.com/pkg/kamailio/kamailio3-mod-usrloc_3.3.7-1_mxs.ipk')

os.execute('/etc/init.d/kamailio enable')
os.execute('/etc/init.d/kamailio start')
```

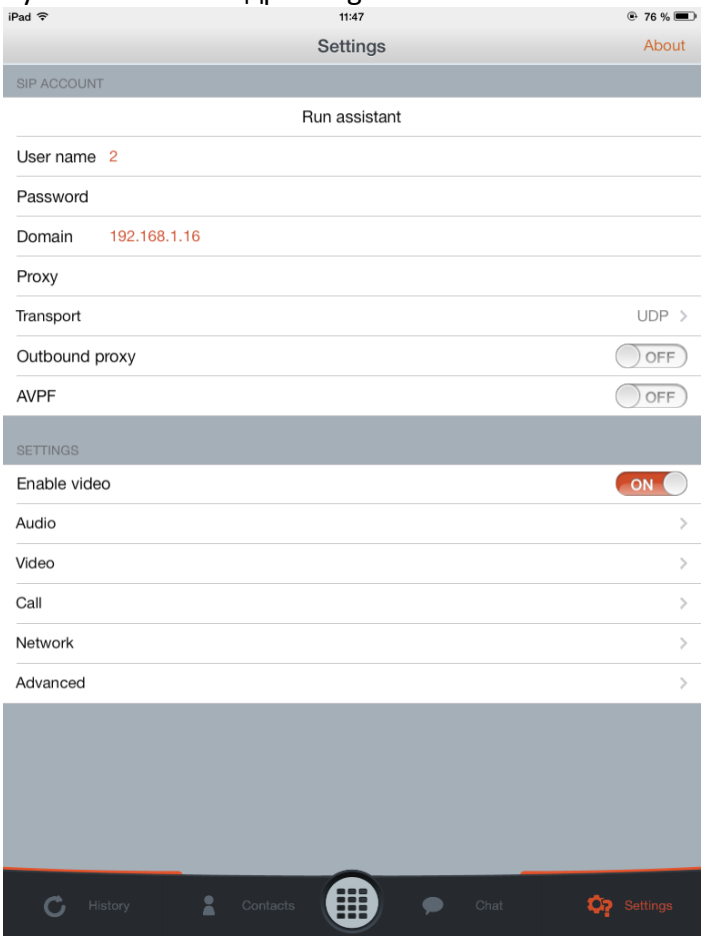
Проверьте, есть у LM доступ в интернет:

Убедитесь, что IP, шлюз, подсеть, DNS настроены правильно.



Клиентское приложение SIP

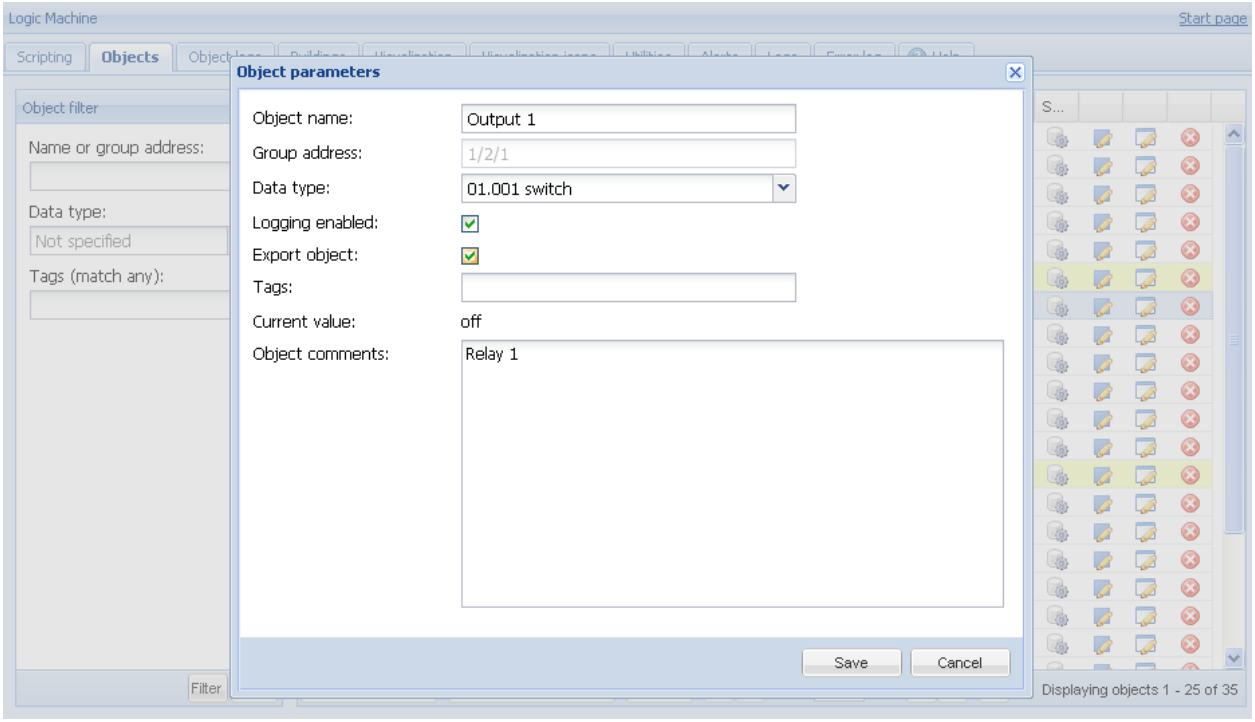
В качестве SIP-клиента можно использовать, например, Linphone. Для этого в его настройках нужно ввести IP-адрес LogicMachine.



21. Экспорт значений объектов в XML

Как сделать объекты KNX доступными для чтения XML:

Во вкладке *Objects* выберите объекты, значения которых вы хотите получать. Отметьте поле *Export*.



XML-запрос с внешнего ПК

Запрос XML выглядит следующим образом::

http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=xml&r=objects

Параметры:

- **address** – адрес объекта ("1/1/1");
- **name** – название объекта ("My object");
- **data** – расшифрованное значение объекта (42 или "01.01.2012");
- **datatype** – тип данных объекта (1 or 5.001);
- **time** – время обновления объекта (временная метка UNIX);
- **date** – дата обновления объекта (в формате RFC);
- **comment** – комментарий к объекту ("Second floor entry lights");
- **tags** – необязательный массив тегов ("Light", "Second floor").

Примечание! Для получения списка объектов, обновлённых по прошествии некоторого

времени, вы можете передать необязательный параметр “updatetime”.

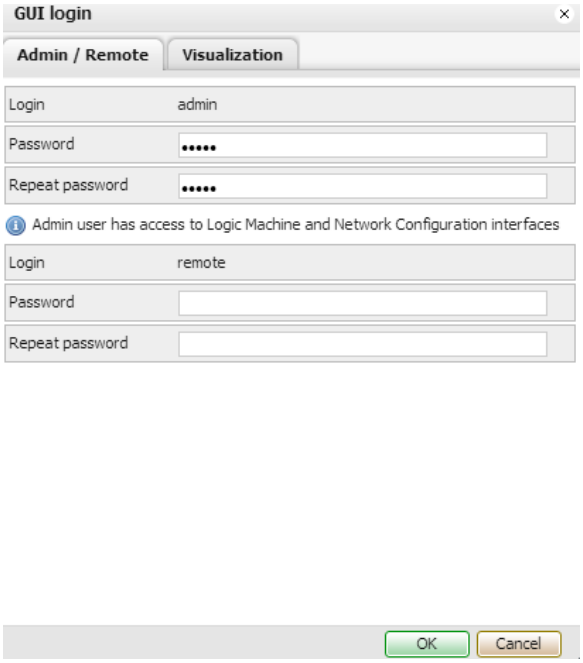
192.168.1.211/cgi-bin/scada-remote/request.cgi?m=xml&r=objects

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<objects>
  ▼<object>
     <comment/>
     <name>Weather Temperature</name>
     <address>5/1/2</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-4</data>
     <datatype>9</datatype>
   </object>
  ▼<object>
     <comment/>
     <name>Weather T Low</name>
     <address>5/1/4</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-13</data>
     <datatype>9</datatype>
   </object>
  ▼<object>
     <comment/>
     <name>Weather T High</name>
     <address>5/1/5</address>
     <date>Tue, 14 Feb 2012 23:41:45 -1000</date>
     <time>1329298905</time>
     <data>-8</data>
     <datatype>9</datatype>
   </object>
 </objects>
```

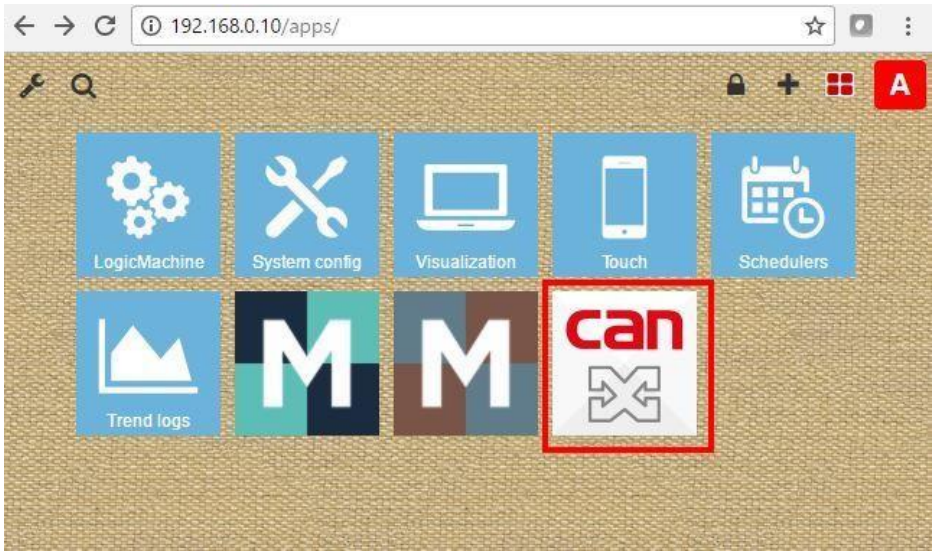

Логин и пароль для XML-запроса

Логин и пароль можно изменить во вкладке *System Config → Service → Remote services*.



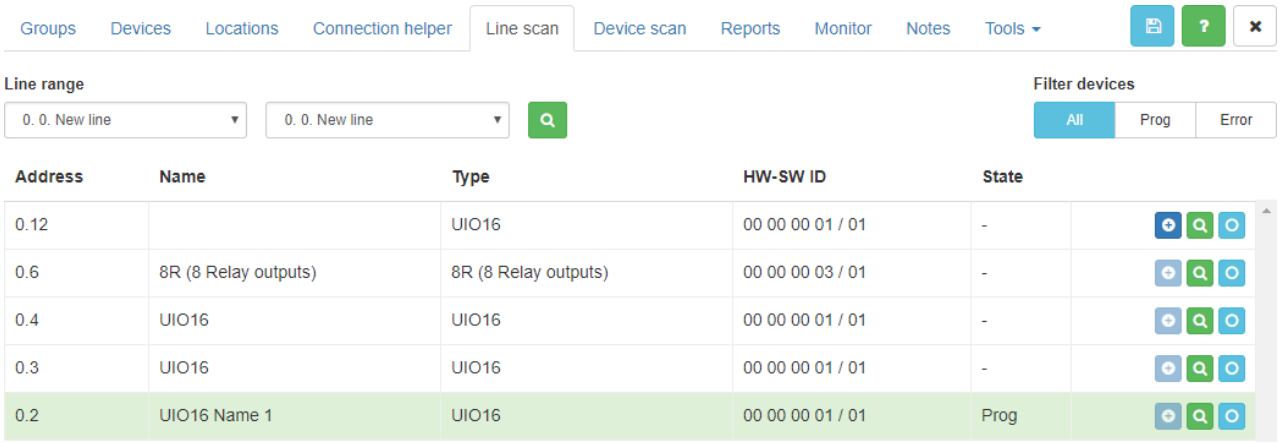
22. CANx конфигурация

Приложение CANx на LogicMachine позволяет настраивать и контролировать устройства на основе протокола CANx. Приложение CANx доступно в магазине приложений LogicMachine.



22.1 Сканирование CAN FT линии

Если уже есть устройства, подключенные в онлайн-режиме, перейдите на вкладку **Line Scan**, укажите **Line range** (или оставьте значение по умолчанию) и щелкните значок .



После обнаружения устройств процесс сканирования можно отменитьпомощью значка .

Address – физический адрес устройства

Name – название устройства. Можно изменить на вкладке «Devices».

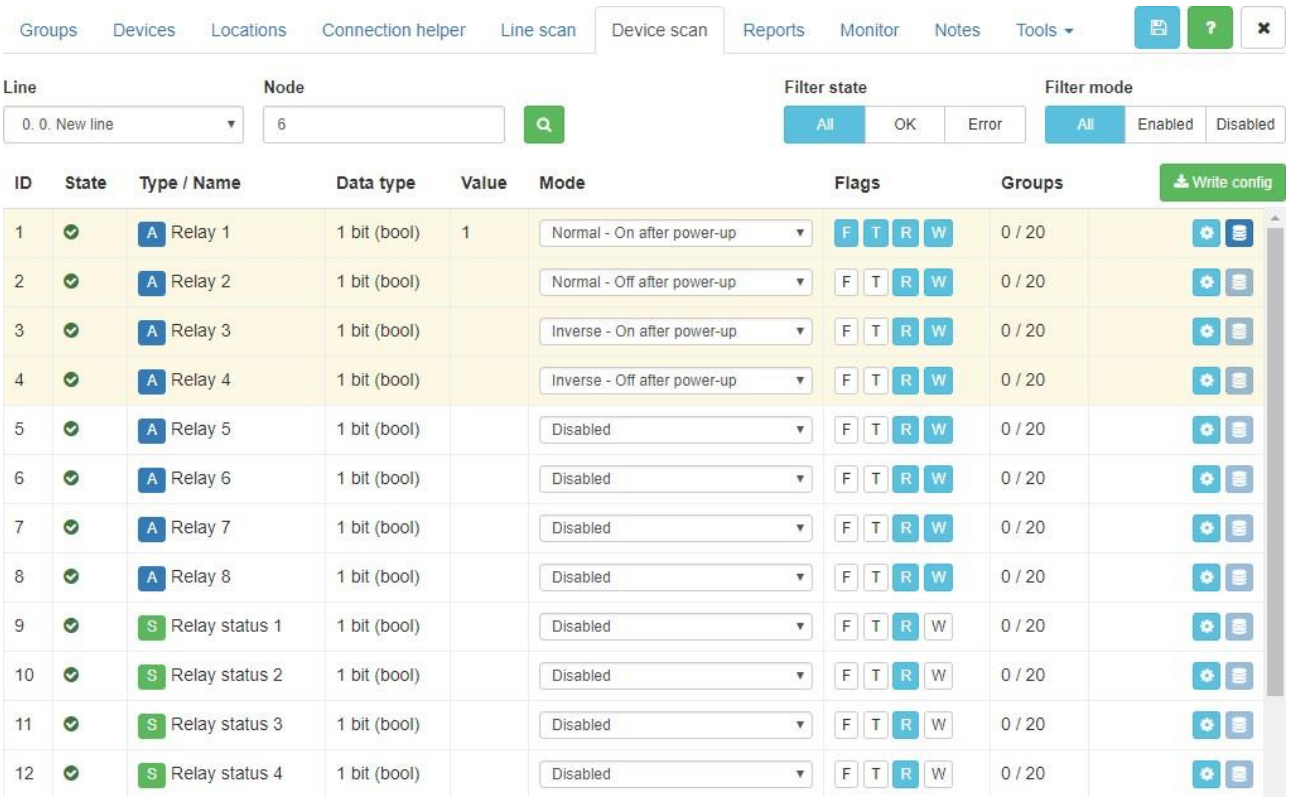
Type – тип устройства (профиль)

HW-SW ID –идентификатор оборудования и идентификатор программного обеспечения.

State – Определяет, находится ли устройство в состоянии программирования или нет (кнопка программирования нажата)

22.2Сканирование устройств

Сканирование устройства показывает все объекты конкретного устройства.



ID – ID номер объекта
State – внутреннее состояние устройства, определяет, нет ли ошибок с устройством
Type / Name – имя обьекта
Data type – тип данных объекта
Value – текущее значение объекта
Mode – режим объекта

Relay / Output modes:

Disabled отключено

«Normal, off after power-up» - нормальный (1 – on, 0 – off) режим работы, выключен после включения

«Inverse, off after power-up» - инвертированный (1 – off, 0 – on) режим работы, выключен после подачи питания

«Normal, on after power-up» - нормальный (1 – on, 0 – off) режим работы, включен после включения

«Inverse, on after power-up» - инвертированный (1 – off, 0 – on) режим работы, включен после подачи питания

Relay status modes:

«Disabled» - отключено
«Normal» - нормальный (1 – on, 0 – off) статус
«Inverse» - инвертированный (1 – off, 0 – on) статус

Input modes:

Switch on/off
Switch off/on (inverse)
Switch Toggle
Button Toggle (optional long press)
Button On (optional long press)
Button Off (optional long press)
Button Start/Stop
Button Stop/Start (inverse)

Input Long Press modes:

Long press Toggle
Long press On
Long press Off

Флаги

F – Фильтрация. Определяет, может ли устройство фильтровать телеграммы (для CAN-CAN роутеров)
T – Передача. Определяет, может ли устройство инициировать связь.
R – Чтение. Определяет, разрешена ли команда чтения устройству
W – Запись. Определяет, разрешена ли команда записи на устройство

Groups – количество групп, связанных с конкретным объектом

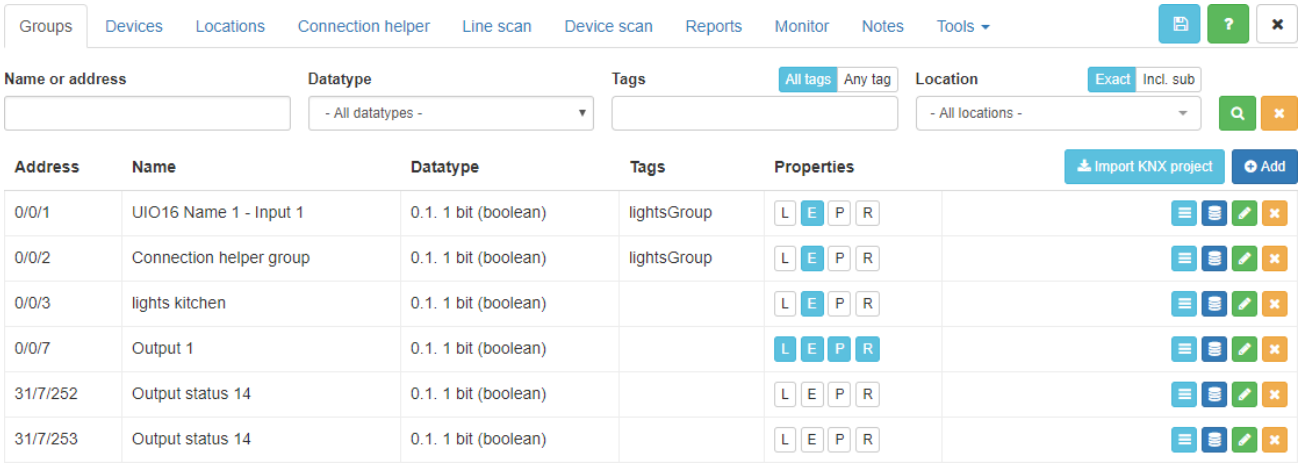
Настроить режим объекта устройства, флаги и группу z

Установить значение объекта устройства

После внесения некоторых изменений кнопка Write config становится активной. После завершения всех изменений и нажатия на эту кнопку вся обновленная конфигурация будет записана на соответствующее устройство.

22.3Конфигурация групповых адресов

Групповые адреса могут быть настроены как в настройках конфигурации объекта, так и непосредственно во вкладке "Groups".



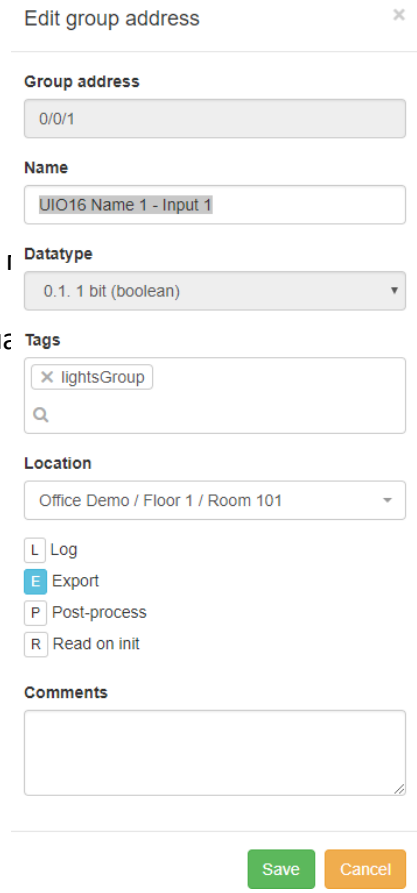
Address – групповой адрес
Name – название группы
Datatype – тип данных
Tags – теги для специальных групп
Properties – свойства групп
L – Регистрировать каждое изменение группы
E – Экспорт. Используется для автоматического сопоставления этой г
(вкладка Objects в LogicMachine)
P – Пост-процесс. Выполнить событийный скрипт при изменении зна
R – Чтение при инициализации (запуск устройства)
Location - расположение объекта / группы


Список устройств, которые используют этот групповой адрес


Чтение / запись значения адреса группы

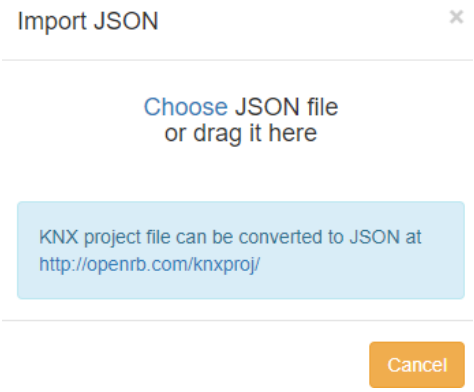

Изменить адрес группы


Удалить адрес группы

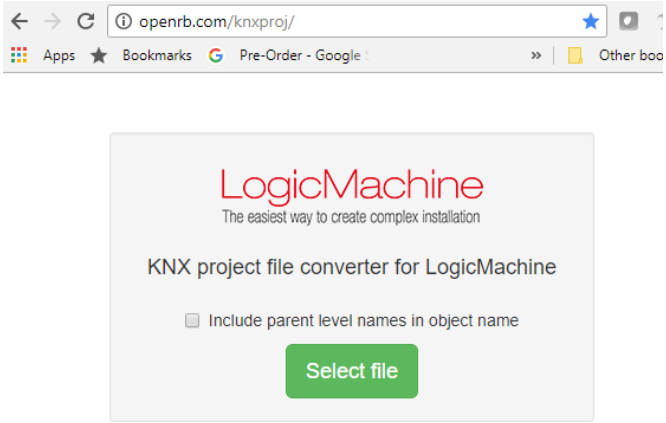


Импорт групповых адресов из файла проекта ETS

Import KNX project кнопка позволяет импортировать файл проекта KNX.

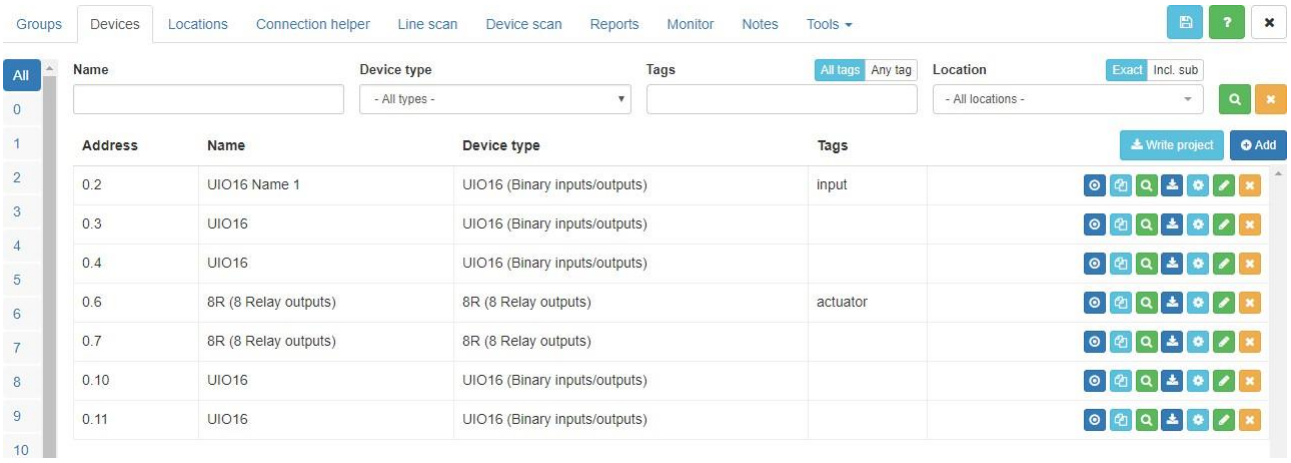


Перед импортом файл проекта ETS необходимо преобразовать в файл JSON с помощью этой онлайн-утилиты.



22.4 Конфигурация устройств

Раздел Devices используется для настройки устройств CANx. Конфигурацию можно выполнить,

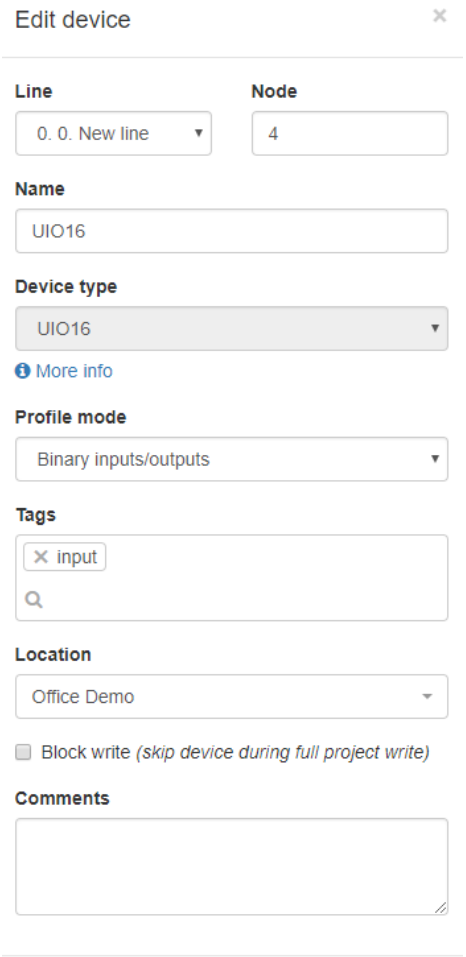


даже если устройство еще физически не подключено к линии (офлайн режим).

Address – физический адрес устройства
Name – название устройства
Device type – тип / профиль устройства
Tags – теги, связанные с устройством

Проверка, подключено ли устройство к шине
Скопировать конфигурацию устройства
Сканировать и показать конфигурацию устройства
Записать конфигурацию в устройство
Настроить параметры устройства
Изменить базовую конфигурацию устройства
Удалить устройство

При нажатии на конкретную запись появляются настройки устройства.



Line [0..15] – ID линии
Node [1..255] – ID узла

Name – имя устройства

Device type – автоматическое определения типа устройства

Profile mode – режим программного профиля для устройства. Для одного типа возможно несколько профилей, например. для UIO16 - двоичные входы / выходы, двоичные выходы, двоичные входы. Это упрощает настройку.

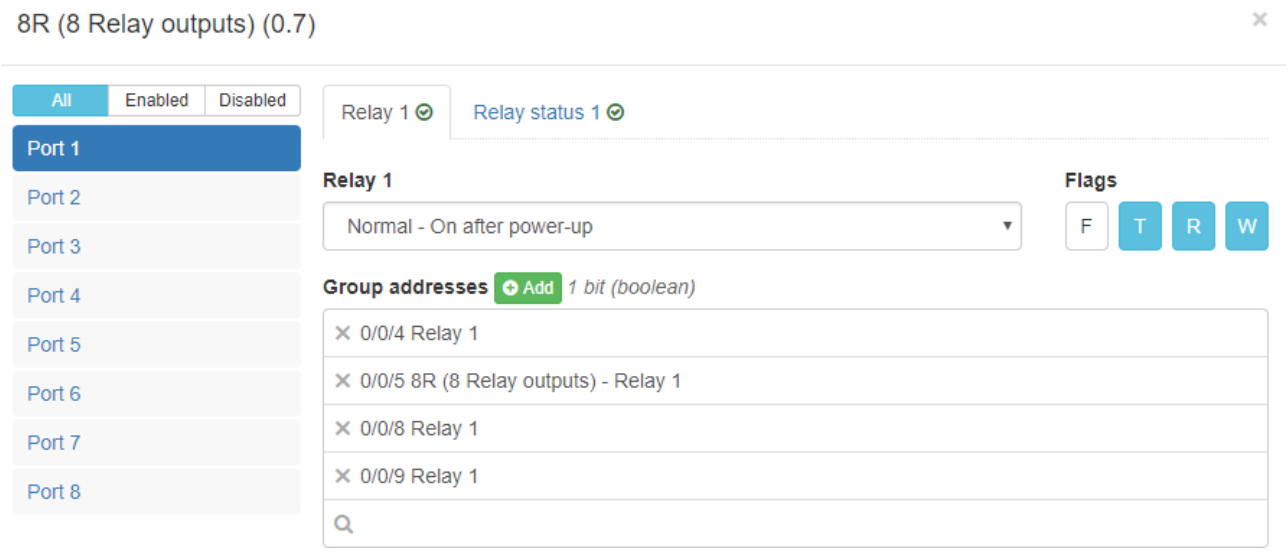
Tags – теги, связанные с этим устройством

Location – расположение устройства

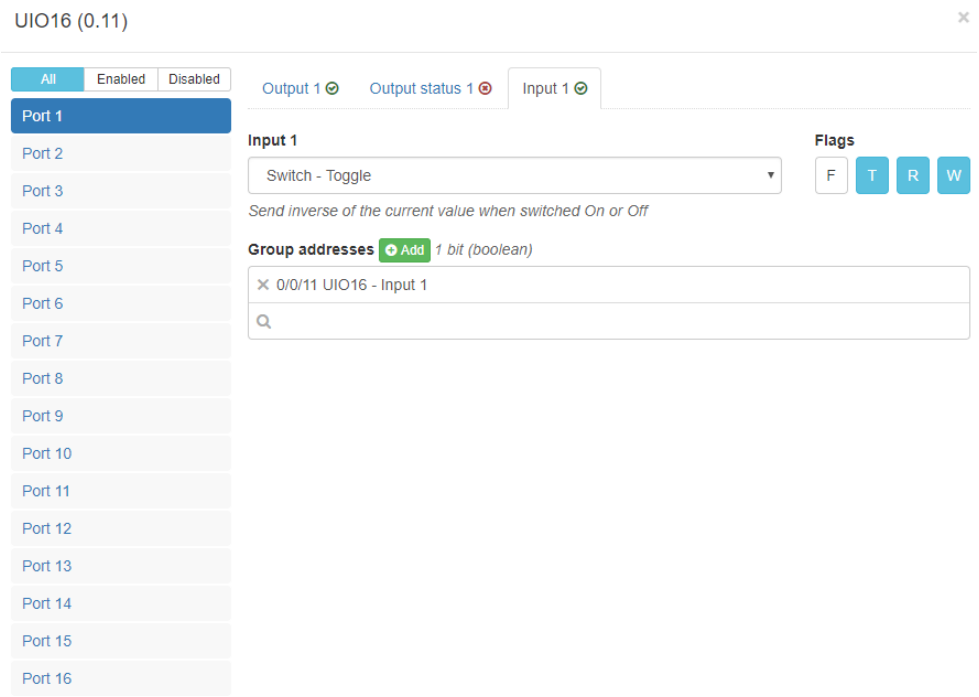
Block write – пропускать устройство при полной заливке проекта

При н Save Cancel являются настройки объектов. В зависимости от выбранного профиля программного обеспечения вы видите соответствующий параметр управления для каждого канала.

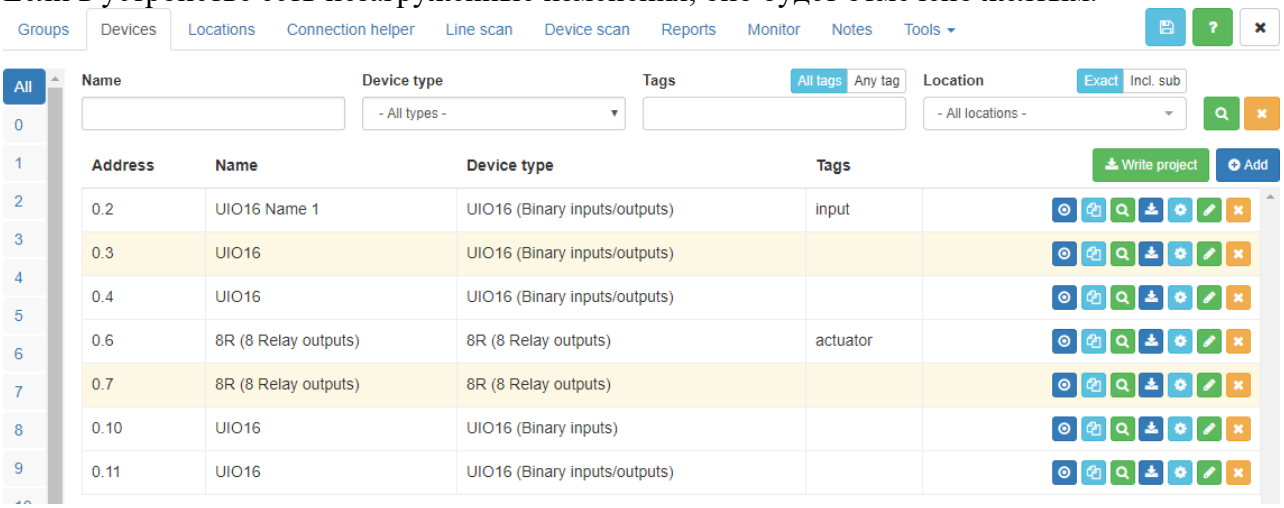
8R - 8 релейных каналов:



UIO16 16-канальный универсальный модуль ввода / вывода:

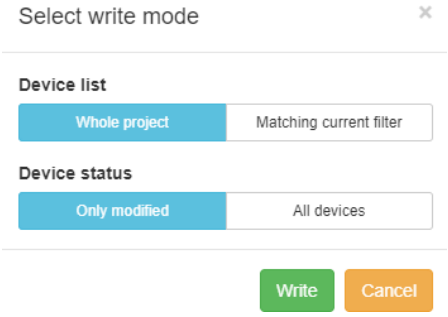


Если в устройстве есть незагруженные изменения, оно будет отмечено желтым.



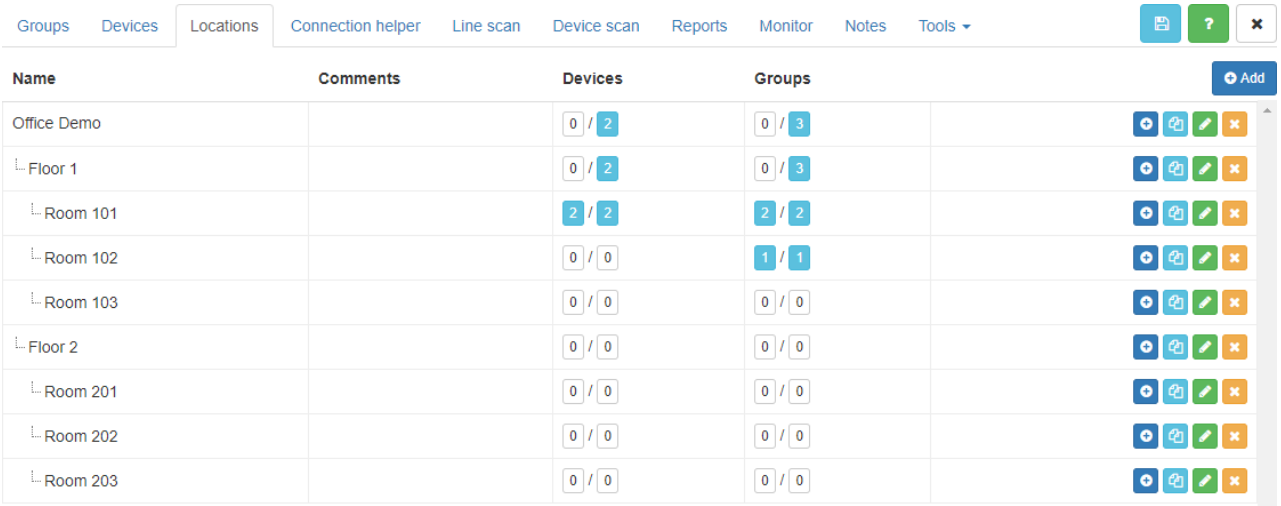
Нажмите эту 

кнопку, чтобы загрузить новые настройки на “**Измененные** / **Все**” устройства.



22.5Местоположение

Структура проекта может быть определена на вкладке Locations.



Name – название локации

Devices – устройства, связанные с определенным местоположением. Первое число обозначает количество конкретных устройств в конкретной комнате. Второй - общее количество устройств.

Groups – групповые адреса, связанные с определенным местоположением. Первое число обозначает количество определенных групп в конкретной комнате. Второй - общее количество групп.

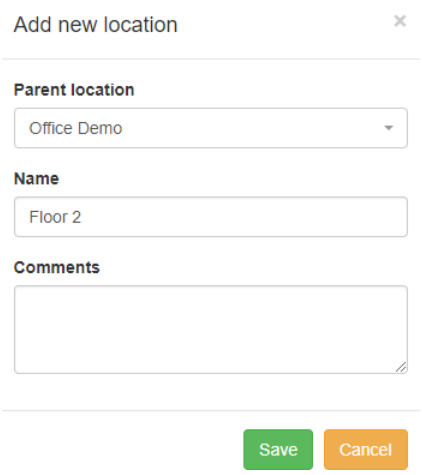
Добавить новую локацию

Повторяющаяся локация, включая устройства и групповые адреса

Редактировать локацию

Удалить локацию и под-локации

Нажав на кнопку Add, вы добавите новое место.



Parent location – выберите родительскую локацию

Name – название новой локации

22.6Отчеты

В разделе «Отчеты» перечислены все устройства в проекте с указанием, какие устройства в какой строке расположены, сколько групповых адресов связано с конкретным устройством, сколько места требуется в шкафу.

Groups | Devices | Locations | Connection helper | Line scan | Device scan | Reports | Monitor | Notes | Tools

Report for project Office Demo

Print

Address	Name	Device type	Group associations	Width (units)
0. New line				
0.2	UIO16 Name 1	UIO16 (Binary inputs/outputs)	2	3
0.3	UIO16	UIO16 (Binary inputs/outputs)	6	3
0.4	UIO16	UIO16 (Binary inputs/outputs)	1	3
0.6	8R (8 Relay outputs)	8R (8 Relay outputs)	0	4
0.7	8R (8 Relay outputs)	8R (8 Relay outputs)	5	4
0.10	UIO16	UIO16 (Binary inputs)	0	3
0.11	UIO16	UIO16 (Binary inputs/outputs)	0	3
Line total (7 devices)			14	23
Project total (7 devices)			**14**	**23**

22.7Мониторинг линии полевой шины.

На вкладке «Монитор» вы можете видеть активность на линии шины, отправлять запросы чтения / записи на определенное устройство / группу / объект.

Groups | Devices | Locations | Connection helper | Line scan | Device scan | Reports | Monitor | Notes | Tools

Command type: Group | Command: GroupValueWrite | Group address: 0/0/8 | Data (HEX, space-separated) 1-8B: 00

Filter messages: All | Individual | Group | Filter direction: All | RX | TX | Name or address

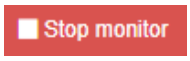
Stop monitor | Export CSV | Clear

#	Time	Type	Address	Name	Command	Data
50	14:24:02.868	RX Group	0/0/10	Relay status 1	GroupValueResponse	00
49	14:24:02.866	TX Group	0/0/8	Relay 1	GroupValueWrite	00
48	14:23:57.421	RX Group	0/0/10	Relay status 1	GroupValueResponse	01
47	14:23:57.418	TX Group	0/0/8	Relay 1	GroupValueWrite	01
46	14:23:24.270	RX Object	0.7.1	8R (8 Relay outputs) (Relay 1)	GAListResponse	09 00 08 00 04
45	14:23:24.267	TX Object	0.7.1	8R (8 Relay outputs) (Relay 1)	GAListRead	-
44	14:23:17.910	RX Device	0.7.0	8R (8 Relay outputs)	DeviceModeResponse	00
43	14:23:17.907	TX Device	0.7.0	8R (8 Relay outputs)	DeviceModeSet	00

Time – время получения телеграммы
Type (RX/TX; Object/Device/Group/Control) – тип телеграммы

Address – групповой или физический адрес устройства, на который получена телеграмма
Name – название устройства
Command – команда получена
Data – данные получены

Запустите монитор, чтобы видеть сообщения в реальном времени



Остановить мониторинг

Экспорт результатов мониторинга в файл CSV

Очистить список

Заполните поля поиска соответствующими данными из выбранной записи

Отправить команду на шину

Выберите тип команды и другие параметры и нажмите зеленую стрелку, чтобы отправить запрос.

<u>Типы команд</u>

Обьекты

ObjectValueRead - Запрос значения объекта
ObjectValueResponse (1..8 bytes) – Ответ на команду ObjectValueRead, возвращает значение для данного объекта
ObjectValueWrite (1..8 bytes) – Записать значение в данный объект
ObjectInfoRead - Запрос на чтение адреса EEPROM с адресом в качестве параметра для команды
ObjectInfoResponse (1 byte) - Ответ для ReadEEprom, возврат значения для данного адреса EEPROM
ObjectConfigWrite (1 byte) - Записать EEPROM addr. = идентификатор объекта (ответ с сообщением ResponseObjectEpromConfig в случае успеха)
GAListAdd (4 bytes) – Добавить групповой адрес в список для данного объекта
GAListDel (4 bytes) - Удалить данный групповой адрес из списка для Object, групповой адрес как параметр для команды удаления
GAListClear - Удалить полный список групповых адресов, связанных с данным объектом
ObjectCmdResponse -
GAListRead (1 byte) - Счетчик партии групповых адресов как параметр (1 для первого 2 адресс., 2 для второго 2адрес и т.д.)
GAListResponse (8 bytes) - Ответ на команду GetListGA, вернуть 2 групповых адреса. подключен к объекту (2 x 4 байта)

Устройство

DeviceInfoRead – Считывание идентификатора оборудования и программного обеспечения для данного устройства, состояния устройства и количества объектов устройства

DeviceInfoResponse (*8 bytes*) – Ответ на…, возвращает ObjQty (байт), статус (байт), 5 байт для идентификатора HW и 1 байт для идентификатора SW

DeviceAddrRequest (*8 bytes*) **-** Запрос адреса для этого устройства (текущий идентификатор линии и идентификатор узла) с идентификатором HW + SFT в качестве данных сообщения

DeviceAddrWrite **-** Возвращает: идентификатор линии (d [1]), идентификатор узла (d [0]) и мастер-код для устройства 6 байтов (d [2] - d [7])

DeviceConfigRead (*2 bytes*) – Команда чтения EEPROM для данного адреса (2 байта DATA)

DeviceConfigResponse (*8 bytes*) **-** Возвращает 8 байт данных из EEPROM для заданного начального адреса

DeviceConfigWrite (*8 bytes*) – Записать 6 байтов данных с 2-байтовым адресом

DeviceConfigWriteResponse (*1 byte*) – 1, если запись прошла успешно, 0 если не удалось

DeviceModeSet – установить режим устройства

DeviceModeResponse – ***...***

DeviceLedOn – Перевод зеленого светодиода устройства в состояние ВКЛ.

DeviceLedOff – Перевод зеленого светодиода устройства в состояние ВЫКЛ.

DevicePing – Ping-запрос к устройству с заданным идентификатором линии и идентификатором узла

DevicePong – Ответ на запрос Ping с ошибками TX и RX шины CAN FT и максимальное количество GA на объект

Группа

GroupValueRead **–** Запрос значения для данного группового адреса

GroupValueResponse (*1..8 bytes*) **–** Ответ на команду GroupValueResponse, возврат значения для данного адреса группы

GroupValueWrite (*1..8 bytes*) **–** Записать значение на указанный адрес группы

Управление

DirectMsgOn (*8 bytes*) **–** Включить прямую связь с кодом (master_code = Fx (code))

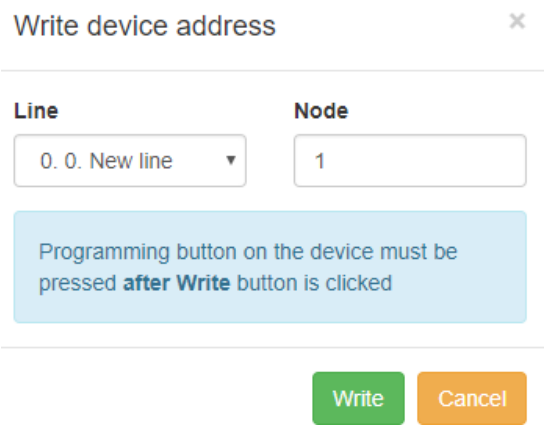
DirectMsgOff (*8 bytes*) **-** Отключить прямую связь с кодом (master_code = Fx (code))

SecureGAOn (*8 bytes*) **–** Перейти в режим Secure GA с помощью кода (master_code = Fx (code))

SecureGAOff (*8 bytes*) **–** Отключить безопасный режим GA с помощью кода (master_code = Fx (code))

22.8 Запись устройства / Физический адрес

Чтобы записать устройству другой физический адрес, перейдите в **Tools → Write device address**. Выберите номер линии и узла и нажмите кнопку «**Write**». Затем нажмите и отпустите кнопку программирования на устройстве. Окно записи закроется после того, как будет записан новый адрес.

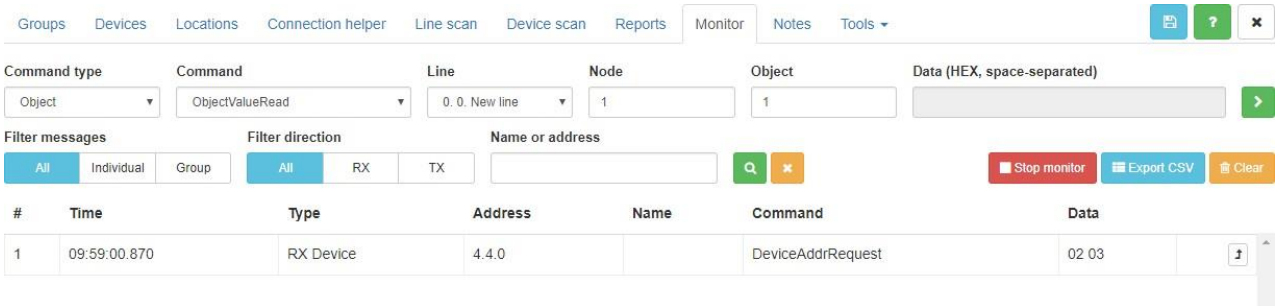


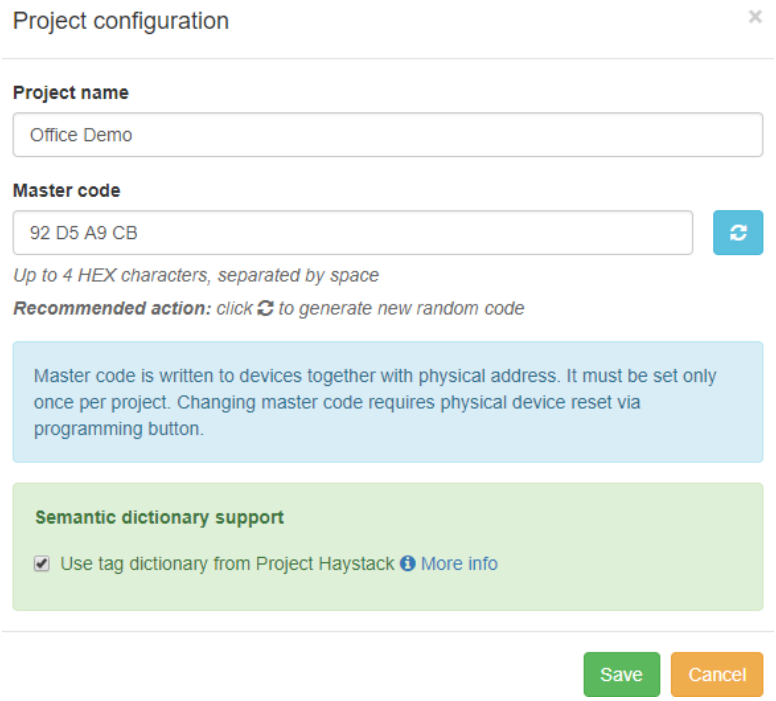
22.9 Посмотреть текущий физический адрес устройства

Чтобы увидеть текущий физический адрес, перейдите на вкладку «**Monitor**», запустите монитор. Затем нажмите кнопку программирования на устройстве, вы увидите текущий физический адрес.

Конфигурация проекта

Общие настройки проекта доступны в **Tools → Project configuration**.





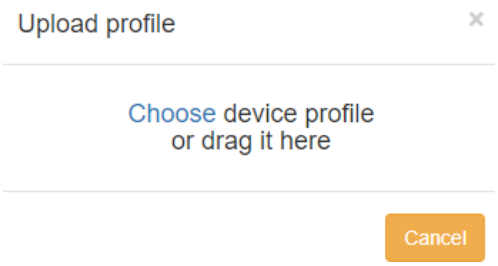
Project name – название проекта

Master code – мастер-код записывается в устройства вместе с физическим адресом. Его необходимо установить только один раз для каждого проекта. Изменение мастер-кода требует физического сброса устройства с помощью кнопки программирования. Мастер-код — это защита установщика от возможности чтения или программирования устройств CANx третьими лицами. Если необходимо заблокировать полный доступ к устройству CANx, необходимо отключить прямую связь / сообщения в **Tools → Disable direct communication** или через **Monitor**.

Semantic dictionary support – определяет, используется ли словарь тегов из Project Haystack: https://project-haystack.org/. Project Haystack - это инициатива с открытым исходным кодом для оптимизации работы с данными из Интернета вещей. Haystack стандартизирует модели семантических данных и веб-сервисы с целью облегчения извлечения выгоды из огромного количества данных, генерируемых интеллектуальными устройствами.

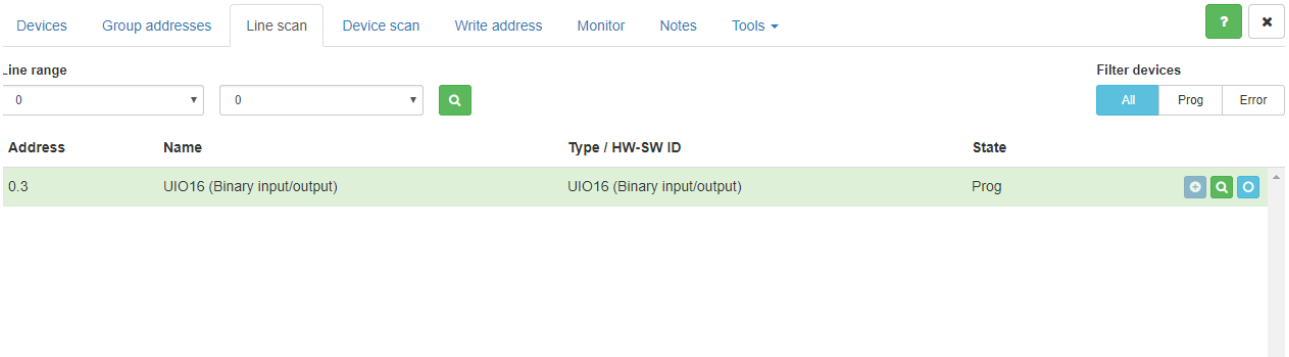
22.10 Новые профили устройств

Чтобы добавить новые профили устройств CANx в приложение CANx, перейдите по ссылке **Tools → Upload device profile**



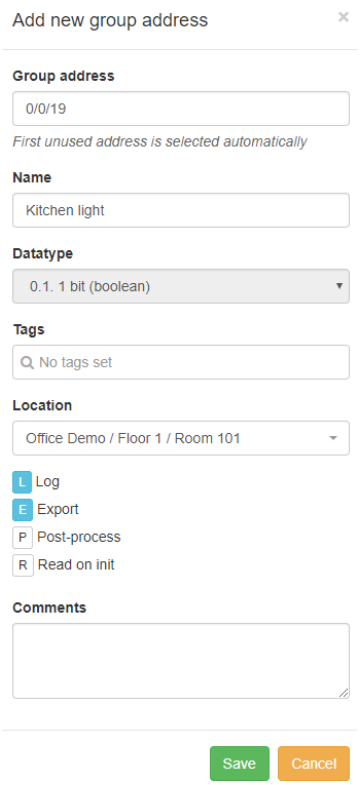
22.11 Поиск устройства, если их несколько в очереди

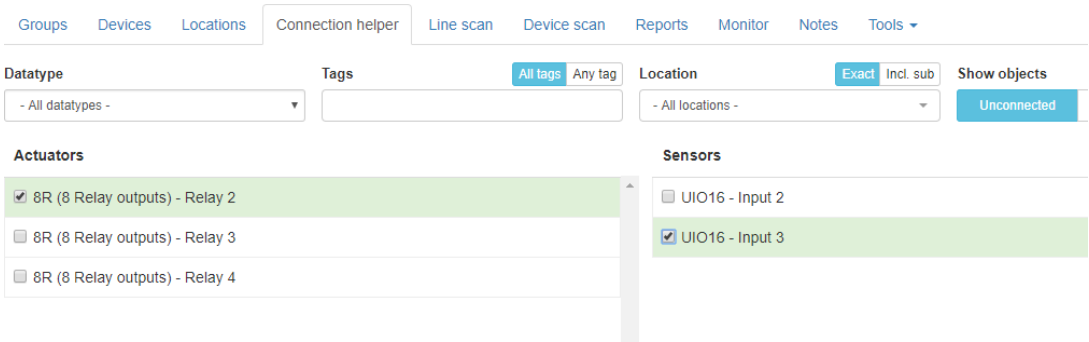
В Line Scan при сканировании вы можете нажать на кнопку программирования устройства, и соответствующее поле станет зеленого цвета.



22.12 Помощник по подключению

Помощник по подключению позволяет упростить установление связи между неиспользуемыми объектами ввода и вывода.





После нажатия кнопки «Сохранить соединение» открывается новое окно адреса группы.

Address – групповой адрес
Name – название группы
Datatype – тип данных группы
Tags – теги для определенной группы
Properties – свойства группы

L – Регистрировать каждое изменение группы
E – Экспорт. Используется для автоматического сопоставления этой группы с группой KNX (вкладка Objects в LogicMachine)
P – Почтовый процесс. Выполнить сценарий изменения адреса группы
R - Читать при инициализации (запуск устройства)

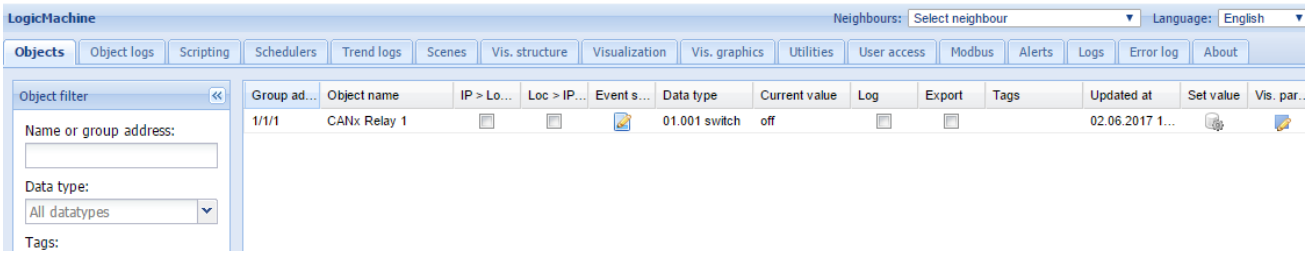
Location – расположение объекта / группы

22.13 Сбросить устройство

Нажмите кнопку программирования на 3-5 секунд, и отпустите, когда на устройстве дважды мигнет красный светодиод. Подождите, пока зеленый светодиод мигнет один раз. После того, как это будет сделано, желательно также перезапустить питание.

22.14 Доступ к объектам CANx из скриптов LogicMachine

Чтобы использовать конфигурацию CANx из сценариев LogicMachine, используйте скрипты. Например:



Управление через скрипт через grp-адрес:

```
can = require('canx')
can.sendrequest({
data = event.getvalue(),
command = can.cmds.group.valuewrite,groupaddress = 1,
})
```

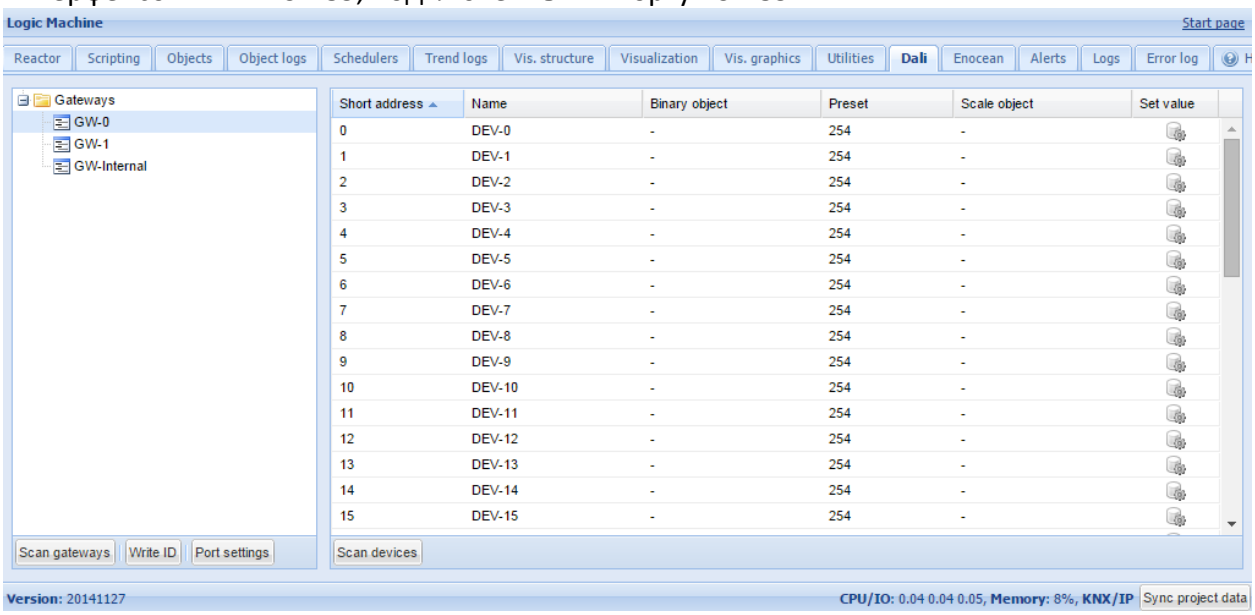
Контроль по физическому адресу:
Изменение идентификатора линии / идентификатора узла в соответствии с настройками вашего устройства ретрансляции.

```
can.sendrequest({lineid = 0,
nodeid = 1,
objectid = 1,
command = can.cmds.object.valuewrite,
```

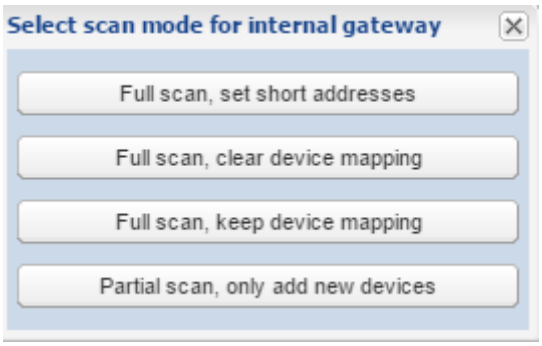
data = grp.getvalue(),
})

23 Настройка DALI

В устройстве есть 1 master-интерфейс DALI. К интерфейсу можно подключить до 64 устройств DALI. Если необходимо присоединить больше устройств, можно воспользоваться интерфейсом DALI-RS-485, подключенным к порту RS-485.



- ***Scan gateways*** – сканирование подключенных шлюзов; сопоставление адресов для отсутствующих устройств автоматически удаляется;
- ***Write ID*** – установка уникального адреса для каждого шлюза;
- ***Scan devices*** – сканирование подключенных к выбранному шлюзу устройств DALI. Существует 4 варианта сканирования:



Full scan, set short addresses – сканирует все подключенные к выбранному шлюзу устройства DALI и автоматически присваивает короткий адрес, начиная с 0,1,2…
Full scan, clear device mapping – сканирует все устройства DALI, в настоящее время подключенные к выбранному шлюзу, без назначения коротких адресов, очищает сопоставление групповых адресов KNX для устройств;

Full scan, keep device mapping – сканирует все текущие подключенные к выбранному шлюзу устройства DALI без назначения коротких адресов, сохраняет привязку устройства к групповым адресам KNX;
Partial scan, only add new devices – сканирует только недавно добавленные на выбранный шлюз устройства DALI, не назначая короткие адреса. Отсутствующие устройства не удаляются из списка;

- ***Port settings*** – имя серийного порта (если подключены внешние интерфейсы DALI-RS-485).

23.2Задание соответствия для объектов DALI

Каждому устройству DALI можно дать собственное имя и указать соответствие для двоичного включения/выключения и масштабирования объекта. Это позволяет осуществлять визуализацию и связь с устройствами DALI с шины KNX без дополнительных скриптов.
После сканирования объектов DALI вы можете нажать на нужный объект и настроить его.

- ***Device name*** – имя устройства;
- ***Binary (ON/OFF) object*** – соответствие с бинарным объектом KNX;
- ***Preset for binary ON*** – предустановка для двоичного ВКЛ;
- ***Scale (0-100%) object*** – соответствие с численным объектом KNX.

Вы можете установить определённое значение, нажав на иконку .

23.3Доступ к шине DALI из скриптов

Для доступа к устройствам DALI из скриптов можно использовать функцию **dalicmd**.

res, err = dalicmd(gwid, cmd, params)

Параметры:

- gwid *(число/строка)* – id шлюза: номер шлюза или internal (при использовании встроенного DALI);
- cmd *(строка)* – команда, которую требуется отправить (см. таблицу команд для

возможных значений);
- `params` *(таблица)* – параметры команды.

params (таблица Lua):

- `addrtype` *(строка)* – тип адреса, необходим только для адресуемых команд; возможные значения: `short, group, broadcast;`
- `address` *(число)* – short или group адрес;
- `value` *(число)* – дополнительное значение для отправки.

Поддерживаются 3 режима адресации:

- `broadcast` – должны отреагировать все slaves: `{ addrtype = 'broadcast' }`
- `short` – должен отреагировать только один slave с соответствующим short-адресом: `{ addrtype = 'short', address = SLAVE_ID }`
- `group` – должны отреагировать несколько slaves, принадлежащих к группе: `{ addrtype = 'group', address = GROUP_ID }`

Типы команд

Если команда адресуемая, тип адреса и адрес можно передать в таблице `params`.

Если команда подразумевает ответ, то она должна адресоваться только одному устройству, которое может ответить, иначе произойдёт коллизия. В случае успеха ответ вернётся в виде бинарной строки, обычно содержащей один байт. Его можно конвертировать в число таким образом:

```
-- запрос статуса slave с short-адресом 5 на встроенной шине DALI
res, err = dalicmd('internal', 'querystatus', { addrtype = 'short', address = 5 })
-- чтение успешно
if res then
  status = res:byte()
end
```

Если у команды есть диапазон значений, таблица params должна содержать поле value с числом в указанном диапазоне. Например, команда arc принимает значения от 0 до 254:

```
-- установить уровень 42 для всех slave-устройств на встроенной шине DALI
dalicmd('internal', 'arc', { addrtype = 'broadcast', value = 42 })
```

Настройка DTR

В случае с командами, для которых требуется DTR, используйте команду `setdtr` для установки значения:

```
-- установка значения dtr в 200 для ballast 5
dalicmd('internal', 'setdtr', { addrtype = 'short', address = 5, value = 200 })
```

Пример (используя шлюз с id 1, выключает все подключённые устройства и включает устройство с short-адресом 5):

```
require('user.dali')

dalicmd(1, 'arc', { addrtype = 'broadcast', value = 0 })
dalicmd(1, 'arc', { addrtype = 'short', address = 5, value = 254 })
```

Пример (задаёт устройству с адресом 5 максимальное значение, равное 200; устройство подключено к внутреннему шлюзу LogicMachine):

```
require('user.dali')

dalicmd('internal', 'setdtr', { addrtype = 'short', address = 5, value = 200 })
dalicmd('internal', 'storemax', { addrtype = 'short', address = 5 })
```

Пример (выдаёт все подключённые к внутреннему шлюзу устройства с short-адресом):

```
require('user.dali')
res, err = dalicmd('internal', 'queryshortaddr', { addrtype = 'broadcast' })
if res then
   log(res:byte())
else
   log(err)
end
```

Пример (добавляет в группу с номером 7 4 устройства DALI с заданным short-адресом):

```
require('user.dali')
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 1, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 2, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 3, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 4, value = 7 })
```

Задание некоторого значения группе 7:

```
require('user.dali')
value = event.getvalue()
value = math.floor(value * 2.54)
dalicmd('internal', 'arc', { addrtype = 'group', address = 7, value = value })
```

Пример (функции для вызова и сохранения сцен, могут быть использованы не только с DALI):

Сначала нужно задать сцену с помощью таблицы Lua, где каждое поле – таблица с двумя полями: групповой адрес и значение. Каждая сцена имеет уникальный идентификатор, который задаётся строкой или числом.

callscene(id) вызывает сцену с заданным идентификатором и объектами. Сначала команда ищет сцену в хранилище; если она не найдена, использует стандартные значения.

savescene(id) получает текущие значения всех объектов из сцены и сохраняет сцену в хранилище.

```
scenes = {}
scenes[1] = {
   { '15/1/1', 50 },
   { '15/1/2', 75 },
   { '15/1/3', 90 },
}
function callscene(id)
  local key, scene
  key = 'scene_' .. id
  scene = storage.get(key, scenes[ id ])
  if type(scene) ~= 'table' then
    alert('Scene ' .. id .. ' not found')
    return
  end
  for _, item in ipairs(scene) do
    grp.write(item[ 1 ], item[ 2 ])
  end
end

function savescene(id)
  local key, scene
  scene = scenes[ id ]
  if type(scene) ~= 'table' then
    alert('Scene ' .. id .. ' not found')
    return
  end
  for i, item in ipairs(scene) do
    scene[ i ][ 2 ] = grp.getvalue(item[ 1 ])
  end
  key = 'scene_' .. id
```

```
  storage.set(key, scene)
end
```

Пример (бинарный диммер для DALI-ламп, который может приглушать свет ламп с физического выключателя):

1) Добавьте пользовательскую функцию *bindimmer*:

```
function bindimmer(up, down, out, event)
  local main, rev, step, val, new, delay
  step = 10 -- в %
  delay = 0.5 -- в секундах
  -- игнорирование команды “стоп”
  val = tonumber(event.datahex, 16)
  if val == 0 then
    return
  end
  -- увеличение яркости, нормальный режим
  if event.dst == up then
    main, rev = up, down
  -- уменьшение яркости, обратный шаг
  elseif event.dst == down then
    main, rev = down, up
    step = -step
  -- неверный объект
  else
    return
  end
  -- текущее выходное значение объекта
  val = grp.getvalue(out) or 0
  while true do
    -- основной объект в состоянии “стоп”
    if not grp.getvalue(main) then
      return
    end
    -- обратный объект в состоянии “старт”
    if grp.getvalue(rev) then
      return
    end
```

```
    -- получение нового значения
    new = math.min(100, val + step)
    new = math.max(0, new)
    -- без изменений, стоп
    if new == val then
      return
    end
    -- запись нового значения
    val = new
    grp.write(out, new, dt.scale)
    -- ожидание следующего запуска
    os.sleep(delay)
  end
end
```

2) Создайте 3 объекта:
1/1/1 – binary (dim up);
1/1/2 – binary (dim down);
1/1/3 – 1-byte scale (output);

3) Создайте событийный скрипт для каждого бинарного объекта:

```
bindimmer('1/1/1', '1/1/2', '1/1/3', event)
```

4) Для регулировки скорости диммирования (изменения яркости) можно изменить переменные *step* и *delay* в функции *bindimmer*.

<u>Команды DALI</u>

Команда	**Описание**	**Адресуемост**	**Отве**	**Значени**
arc	прямой контроль питания	+		0..254
off	выключить	+		
up	включить	+		
down	down	+		
stepup	увеличить яркость	+		
stepdown	уменьшить яркость	+		
recallmin	запросить максимальный урове	+		
recallmax	запросить минимальный уровен	+		
stepdownoff	уменьшить яркость и выключит	+		
stepupon	включить и повысить напряжен	+		
gotoscene	перейти к сцене			0..15
reset	сбросить	+		
storeactual	сохранить текущий уровень в dt	+		

storemax	запомнить dtr как макс. уровень	+		
storemin	запомнить dtr как мин. уровень	+		
storesystemfailure	запомнить dtr как уровень отказ системы	+		
storepoweron	запомнить dtr как уровень включения	+		
storefadetime	запомнить dtr как время затухан	+		
storefaderate	запомнить dtr как уровень затухания	+		
storescene	запомнить dtr как сцену	+		0..15
removescene	удалить из сцены	+		0..15
addtogroup	добавить в группу	+		0..15
removefromgroup	удалить из группы	+		0..15
storeshortaddress	запомнить dtr как короткий адр	+		
querystatus	запрос статуса	+	+	
queryballast	запрос балласта	+	+	
querylampfailure	запрос отключения лампы	+	+	
querylamppoweron	запрос включения лампы	+	+	
querylimiterror	запрос ошибки диапазона	+	+	
queryresetstate	запрос сброса состояния	+	+	
querymissingshort	запрос потери короткого адреса	+	+	
queryversion	запрос номера версии	+	+	
querydtr	запрос содержания dtr	+	+	
querydevicetype	запрос типа устройства	+	+	
queryphysicalmin	запрос уровня физ. минимума	+	+	
querypowerfailure	запрос отказа питания	+	+	
queryactual	запрос текущего уровня	+	+	
querymax	запрос максимального уровня	+	+	
querymin	запрос минимального уровня	+	+	
querypoweron	запрос уровня включения	+	+	
querysystemfailure	запрос уровня отказа системы	+	+	
queryfadetimerate	запрос времени / уровня затуха	+	+	
queryscene	запрос уровня сцены (сцены 0-1	+	+	0..15
querygroupslow	запрос групп 0-7	+	+	
querygroupshigh	запрос групп 8-15	+	+	
queryrandomaddrh	запрос случайного адреса (h)	+	+	
queryrandomaddrm	запрос случайного адреса (m)	+	+	
queryrandomaddrl	запрос случайного адреса (l)	+	+	
terminate	завершить			
setdtr	задать регистр переноса данных (dtr)			0..255
initialise	инициализировать			
randomise	рандомизировать			

compare	сравнить		+	
withdraw	отозвать			
searchaddrh	задать адрес поиска (h)			0..255
searchaddrm	задать адрес поиска (m)			0..255
searchaddrl	задать адрес поиска (l)			0..255
programshortaddr	короткий адрес программы			0..63
verifyshortaddr	проверить короткий адрес		+	0..63
queryshortaddr	короткий адрес запроса		+	
physicalselection	физическое выделение			
enabledevicetype	сделать доступным устройство типа x			0..255

24 Уведомления и ошибки

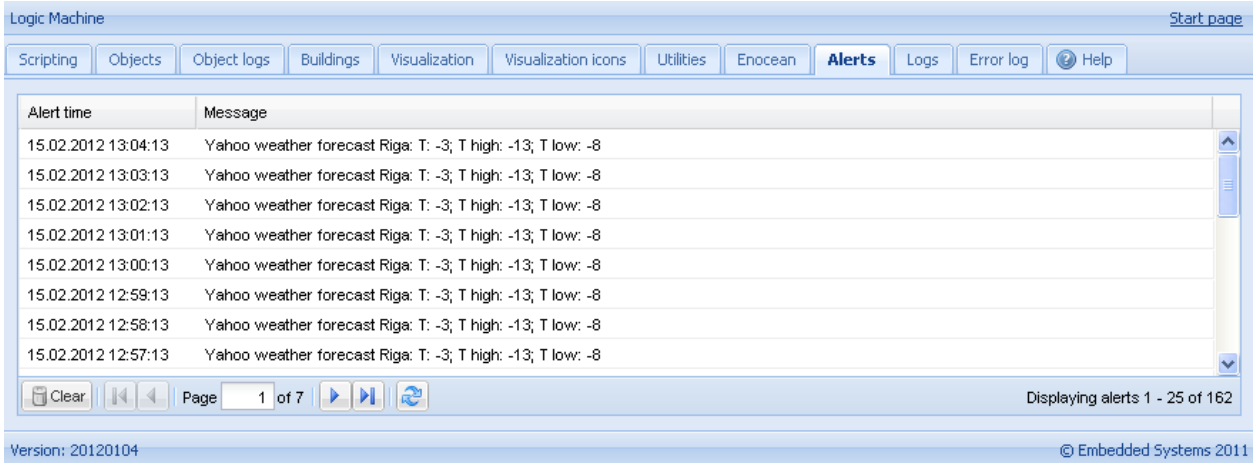
Похожим образом с помощью XML-запроса можно получать уведомления и ошибки.

Запрос уведомлений:
http://remote:remote@192.168.0.10/cgi-bin/scada-remote/request.cgi?m=xml&r=alerts

Запрос ошибок:
http://remote:remote@192.168.0.10/cgi-bin/scada-remote/request.cgi?m=xml&r=errors

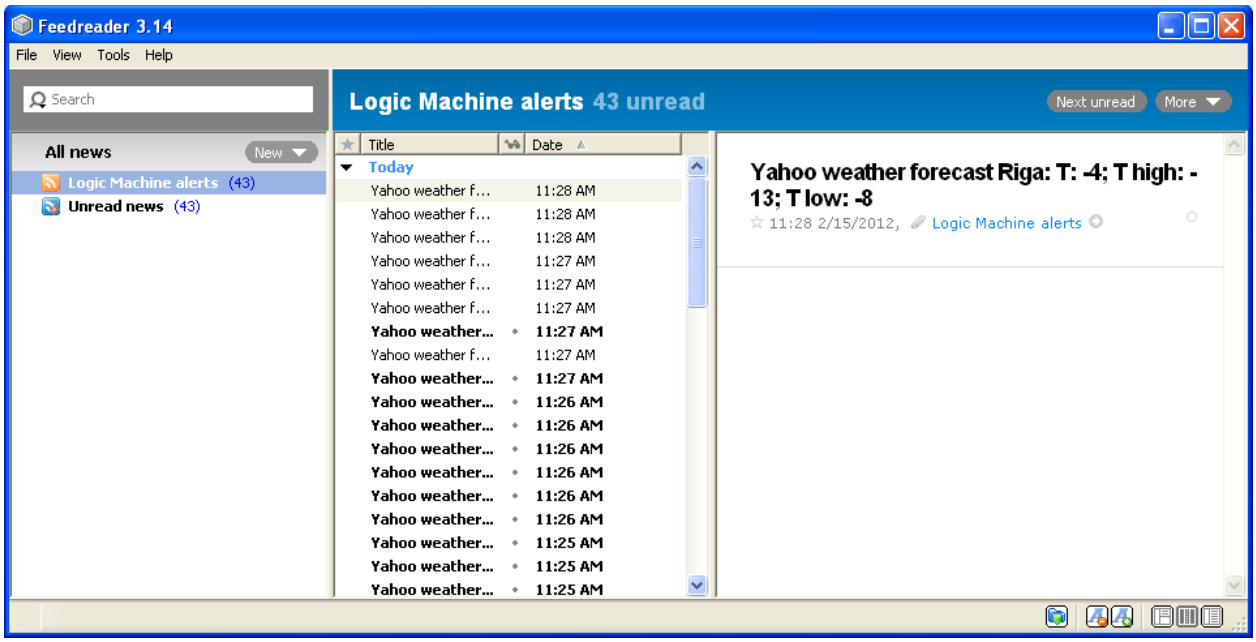
25 Чтение уведомлений с помощью RSS

Прочитать уведомления и ошибки можно с помощью RSS-ридеров.



<u>Добавление новой RSS-ленты в RSS-ридер</u>

- Отправьте следующий запрос:
- http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=rss&r=alerts
- Будут показаны последние 50 уведомлений.
- Время уведомления будет отображено в формате временной метки UNIX, дата уведомления отобразится в формате RFC.



<u>Чтение вкладки *Alerts* с помощью RSS</u>

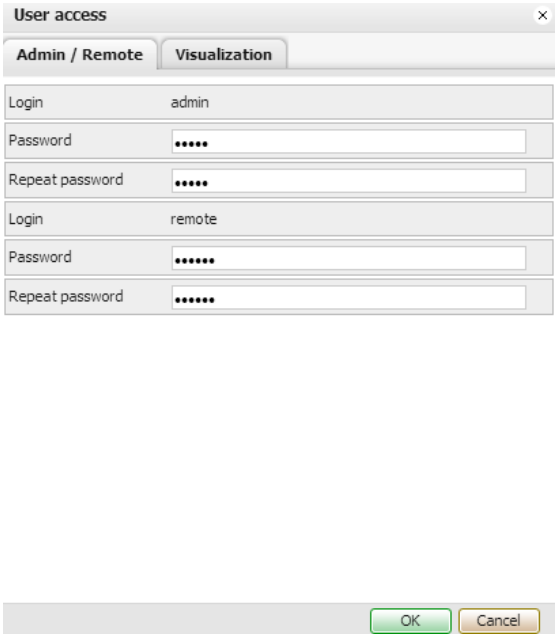
RSS можно также использовать для чтения ошибок из вкладки *Alerts*. В этом случае запрос

будет выглядеть следующим образом:

http://remote:remote@192.168.1.211/cgi-bin/scada-remote/request.cgi?m=rss&r=errors

Логин и пароль для RSS запроса

Логин и пароль можно изменить во вкладке *System Config → Service → Remote services*.



26 Другие примеры

Различные примеры, протоколы интеграции сторонних разработчиков и другие полезные приложения можно найти здесь:

http://openrb.com/all-examples/
http://forum.logicmachine.net/