PROGRAMLAB

ИННОВАЦИОННЫЕ ПРОГРАММНЫЕ КОМПЛЕКСЫ

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

ВИРТУАЛЬНЫЙ ЛАБОРАТОРНЫЙ КОМПЛЕКС
«ИНТЕРНЕТ ВЕЩЕЙ »

PL-LLC.RU

Оглавление

Введение в IoT

Интернет вещей (англ. Internet of Things, IoT) — концепция вычислительной сети физических предметов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека.

История создания и развития

Идея, что устройства могут обмениваться информацией друг с другом без участия человека появилась достаточно давно. Еще в конце 70-х обсуждалась возможность полной автоматизации передачи данных. Тогда подобный подход назывался “повсеместные вычисления” (pervasive computing). Технологиям потребовалось несколько десятилетий развития для того, чтобы наконец стало возможным заговорить об Интернете вещей.

Во второй половине девяностых британец Кевин Эштон работал на компанию Procter and Gamble и занимался оптимизацией производства. Он заметил, что оптимизация напрямую зависит от скорости передачи и обработки данных. Когда сбором и обработкой данных занимаются люди, то на это могут уйти дни. Использование радиочастотной идентификации (RFID) позволило ускорить процесс передачи данных непосредственно между устройствами. Именно тогда у него и появилась идея — а что, если вещи будут собирать, обрабатывать и передавать данные без участия человека? Как бы это называлось? “Интернет вещей”, подумал Эштон, и оказался провидцем.

Потребовалось почти десятилетие для того, чтобы словосочетание «Интернет вещей» вошло в повседневную жизнь. Вместе с искусственным интеллектом IoT стал передовым направлением развития информационных технологий. Так, в 2008 году IPSO Alliance создал союз компаний, которые поддержали разработку технологий, связанных с Интернетом вещей. Это послужило сигналом для крупных корпораций.

Летом 2010 года стало известно, что сервис Google StreetView кроме показа панорамных фотографий умеет собирать данные об используемых Wi-Fi сетях. Эксперты заговорили о разработках нового протокола передачи данных, который позволит обмениваться данными между устройствами. В том же году Китай заявил, что планирует включить Интернет вещей в список приоритетных направлений исследований на ближайшие пять лет. Стало понятно, что сбором, обработкой и хранением данных заинтересовались не

только крупные корпорации, но и правительства. В 2011 году занимающаяся исследованием рынка компания Gartner включила IoT в свой лист наиболее перспективных развивающихся технологий.

Интернет вещей завоевывал мир. В 2012 году крупнейшая европейская интернет конференция LeWeb была посвящена данной теме, а такие журналы как Forbes, Fast Company и Wired начали активно использовать термин Internet of Things. Весь мир заговорил об Интернете вещей, а компании начали гонку технологий. В 2013 году IDC опубликовало исследование, в котором спрогнозировало рост рынка IoT к 2020 году до 8.9 триллионов долларов.

В январе 2014 года Google приобретает за 3.2 миллиона долларов компанию Nest, которая занималась разработкой устройств “умного дома” и созданием систем управления зданиями. Считается, что именно тогда рынок полностью признал — за Интернетом вещей ближайшее будущее. В том же году крупнейшая американская технологическая выставка Consumer Electronics Show прошла в Лас-Вегасе под вывеской Internet of Things. Так началась эпоха IoT.

Промышленный Интернет Вещей

Составной частью Интернета Вещей и его главной на данном этапе развития технологий движущей силой является Промышленный (или Индустриальный) Интернет Вещей (Industrial Internet of Things, IIoT).

Промышленный Интернет Вещей – это система объединенных компьютерных сетей и подключенных к ним промышленных (производственных) объектов со встроенными датчиками и программным обеспечением для сбора и обмена данными, с возможностью удаленного контроля и управления в автоматизированном режиме, без участия человека.

Как устроен Промышленный Интернет Вещей

На первом этапе внедрения IIoT на промышленное оборудование устанавливают датчики, исполнительные механизмы, контроллеры и человеко-машинные интерфейсы. В результате становится возможным сбор информации, которая позволяет руководству получать объективные и точные данные о состоянии производства. Обработанные данные предоставляются всем подразделениям предприятия. Это помогает наладить взаимодействие между сотрудниками разных подразделений и принимать обоснованные решения.

Полученная информация может быть использована для предотвращения внеплановых простоев, поломок оборудования, сокращения внепланового техобслуживания и сбоев в управлении цепочками поставок, тем самым позволяя предприятию функционировать более эффективно.

При обработке огромного массива неструктурированных данных, поступающих с датчиков, их фильтрация и адекватная интерпретация становится приоритетной задачей. Поэтому особую значимость приобретает представление информации в понятном пользователю виде. Для этого используются передовые аналитические платформы, предназначенные для сбора, хранения и анализа данных о технологических процессах и событиях, работающие в реальном масштабе времени.

Промышленный Интернет Вещей позволяет создавать производства, которые оказываются более экономными, гибкими и эффективными, чем существующие. Беспроводные устройства с поддержкой протокола IP, включая смартфоны, планшеты и датчики, уже активно используются на производстве. Имеющиеся проводные сети датчиков в ближайшие годы будут расширены и дополнены беспроводными сетями, благодаря чему на предприятиях существенно расширятся зоны применения систем мониторинга и управления. Следующий этап оптимизации производственных процессов будет характеризоваться все более плотной конвергенцией лучших информационных и операционных технологий.

По мере становления цифровых экосистем производственные предприятия из изолированных систем, самостоятельно выполняющих все необходимые для производства продукции производственные и бизнес-процессы, будут преобразовываться в открытые системы, объединяющие различных участников рынка; управлять средствами производства в этих системах будет не персонал, а облачные сервисы, конечная цель всех этих трансформаций – не выпуск продукции, а предоставление услуг потребителю.

Архитектура интернет вещей

Классическая архитектура интернета вещей включает в себя:

IoT-устройства. Они собирают показания с датчиков и выполняют физические действия. Могут быть персональными, носимыми и встроенными.

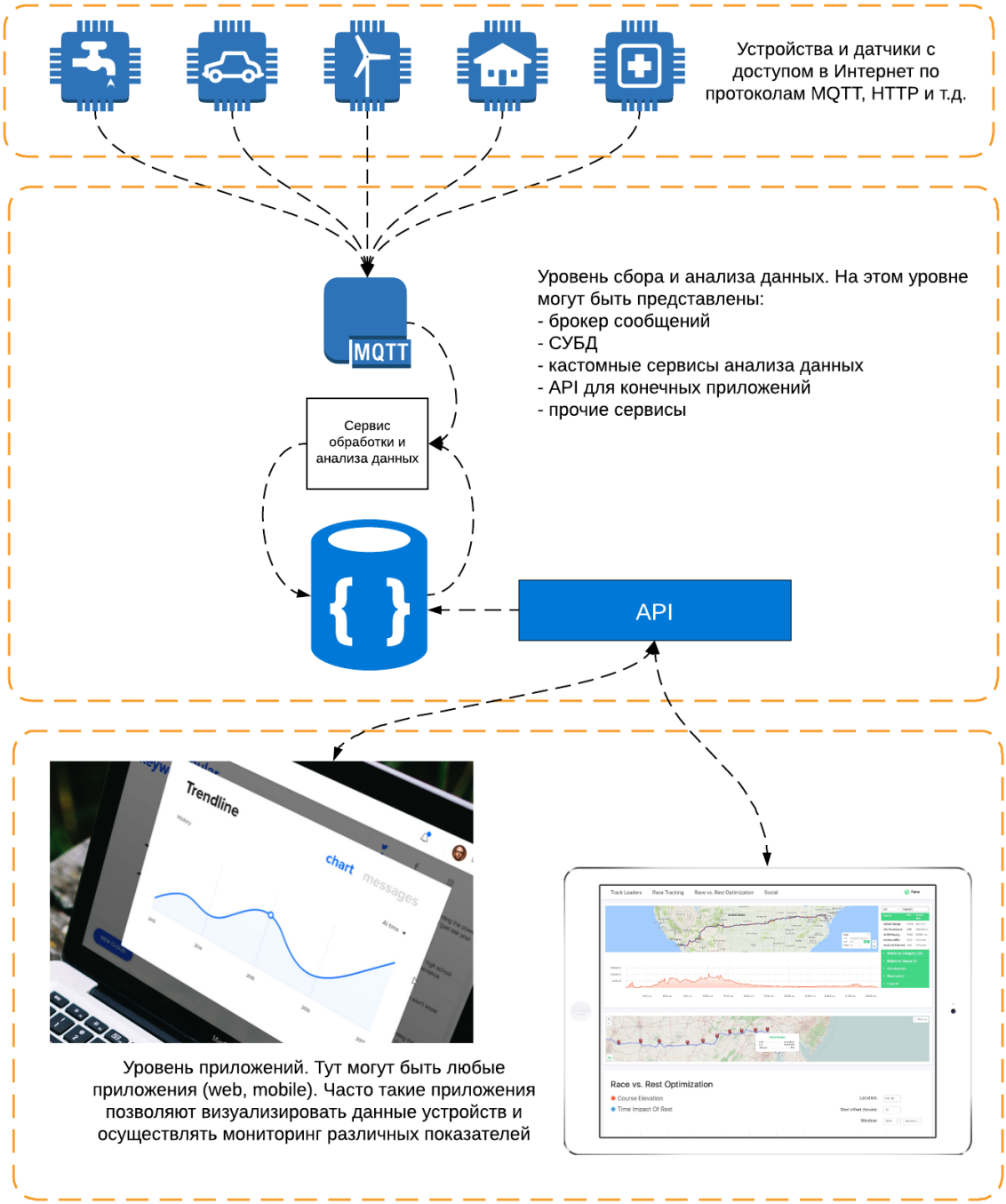


Рисунок 1 -Схема работы IoT сервиса

Встроенные системы. Здесь используется как оборудование, так и программное обеспечение. Такие системы управляют определенными функциями, связанными с более крупной системой. Встроенные системы создаются на основе микропроцессоров или микроконтроллеров.

Интеллектуальные устройства. Эти устройства могут выполнять вычисления и часто оснащены микроконтроллером.

Микроконтроллер (MCU). Эти небольшие компьютеры внедряются в микросхемы и оснащены ЦП, ОЗУ и ПЗУ. Они содержат элементы, необходимые для выполнения простых задач. Но микроконтроллеры имеют более ограниченную мощность, чем микропроцессоры.

***Микропроцессор (MPU)*.** Функции ЦП можно размещать на одной или нескольких интегральных микросхемах. Хотя для выполнения задач микропроцессорам требуются периферийные устройства, затраты на обработку значительно сокращаются, так как в них входит только стоимость ЦП.

***Устройства, не предназначенные для вычислений*.** Эти устройства только устанавливают подключение и передают данные. У них нет возможности выполнять вычисления.

Преобразователи. Физические устройства, которые преобразуют один вид энергии в другой. В контексте устройств Интернета вещей к ним относятся внутренние датчики и исполнительные устройства, передающие данные по мере того, как объекты взаимодействуют с их средой.

***Исполнительные устройства*.** Выполняют физические действия, когда центр управления дает указания. Обычно причиной являются изменения, зафиксированные датчиками.

***Датчики*.** Выявляют изменения в своей среде и создают электрические сигналы для обмена данными. Датчики обычно обнаруживают изменения окружающей среды, например, температуры, физического положения и содержания химических веществ.

Шлюзы, которые получают информацию от устройств и передают им команды выполнения действий. Как правило, представлены аппаратным маршрутизатором или программным обеспечением; используют разные протоколы.

Основная задача шлюзов – основной функции преобразования протоколов, пользователь может добавить дополнительные функции по обработке и хранению данных, управления устройствами, резервирования и другие функции.

В IoT используются промышленные протоколы и самый популярный из них – это MQTT.

Функциональные возможности шлюзов IoT:

В зависимости от решаемой задачи вы сможете выбрать готовое решение или платформу для разработки собственного решения с дополнительными функциями по управлению, обработке и хранению данных.

Шлюзы для интернета вещей (IoT) и платформы для построения систем промышленного интернета вещей (IIoT) позволяют вам реализовать масштабируемые системы сбора и обработки данных с множества устройств с функциями преобразования протоколов и управления различными устройствами.

Сервер, где хранятся, обрабатываются и анализируются показания датчиков. Может быть реализован на базе виртуального сервера, реальной машины или через облако.

Клиентская часть, реализуется через мобильное или веб-приложение. Обеспечивает доступ к данным устройств и наглядному представлению результатов анализа.

Протоколы Интернета вещей

Устройства Интернета вещей обмениваются данными с помощью протоколов Интернета вещей. Протокол IP — это набор правил, определяющих способ отправки данных в Интернет. Протоколы Интернета вещей гарантируют, что информация с одного устройства или датчика будет считываться и правильно интерпретироваться другим. Так как существует множество устройств Интернета вещей, важно использовать нужный протокол в нужном контексте.

Тип используемого протокола Интернета вещей зависит от уровня архитектуры системы, на котором должны передаваться данные. Модель OSI предоставляет карту различных уровней, на которых выполняется отправка и получение данных. Каждый протокол в архитектуре системы Интернета вещей обеспечивает взаимодействие между устройствами, между устройством и шлюзом, между шлюзом и центром обработки данных или между шлюзом и облаком, а также обмен данными между центрами обработки данных.

Уровень приложений

Уровень приложения служит интерфейсом для обмена данными между пользователем и устройством.

Расширенный протокол управления очередью сообщений (AMQP)

Уровень оборудования, на котором устанавливается взаимодействие между ПО промежуточного слоя для обмена сообщениями. Это обеспечивает совместную работу разнообразных систем и приложений. Так формируется стандартизированная система обмена сообщениями в промышленном масштабе.

Ограниченный протокол приложений (CoAP)

Сетевой протокол с ограничением пропускной способности и доступа к сети, предназначенный для устройств с ограниченной мощностью. Протокол обеспечивает обмен данными между компьютерами. Кроме того, CoAP — это протокол передачи документов, который работает с помощью протокола UDP.

Служба распределения данных (DDS)

Универсальный одноранговый протокол связи, который позволяет выполнять любые действия — от запуска небольших устройств до подключения высокопроизводительных сетей. DDS упрощает развертывание, повышает надежность и уменьшает сложность.

MQTT

Протокол обмена сообщениями, предназначенный для не интенсивного обмена данными между компьютерами. В основном используется для подключений с низкой пропускной способностью к удаленным расположениям. MQTT использует подписанный издателем шаблон. Это идеальный вариант для небольших устройств, требующих эффективной пропускной способности и использования аккумулятора.

Уровень транспортировки

Уровень транспортировки обеспечивает и защищает обмен данными между уровнями.

Протокол TCP

Главный протокол для большинства типов подключения к Интернету. Обеспечивает обмен данными между узлами. Для этого крупные наборы данных разбиваются на отдельные пакеты. При необходимости пакеты повторно отправляются и перекомпонуются.

Протокол UDP

Протокол связи для взаимодействия между процессами. Работает на основе протокола IP. Протокол UDP повышает скорость передачи данных по

протоколу TCP. Это лучший вариант для приложений, требующих передачи данных без потерь.

Уровень сети

Уровень сети помогает отдельным устройствам взаимодействовать с маршрутизатором.

6LoWPAN

Версия IPv6 с меньшей мощностью, которая сокращает время передачи.

IPv6

Это последнее обновление протокола IP для маршрутизации трафика через Интернет, а также определения и обнаружения устройств в сети.

Канальный уровень данных

На этом уровне данные передаются в пределах архитектуры системы. При этом определяются и исправляются ошибки, обнаруженные на физическом уровне.

IEEE 802.15.4

Стандарт радиочастотных устройств для беспроводного подключения с низким энергопотреблением. Он используется с Zigbee, 6LoWPAN и другими стандартами для создания беспроводных сетей.

LPWAN

Сеть такого типа обеспечивает связь на расстоянии от 500 метров. LoRaWAN — это пример сети LPWAN, оптимизированной для энергосбережения.

Физический уровень

На физическом уровне устанавливается коммуникационный канал, обеспечивающий подключение устройств в указанной среде.

Bluetooth с низким энергопотреблением (BLE)

Позволяет значительно сократить энергопотребление и затраты. Поддерживает такой же диапазон подключения, как и классическая технология Bluetooth. BLE работает напрямую в мобильных операционных системах. Эта технология быстро приобретает популярность в сфере бытовой электроники, так как она экономична и обеспечивает длительную работу от аккумулятора.

Ethernet

Это экономичное проводное соединение, которое обеспечивает быстрое подключение к данным, а также малую задержку.

"Долгосрочное развитие" (LTE)

Стандарт беспроводной широкополосной связи для мобильных устройств и терминалов данных. LTE увеличивает мощность и скорость беспроводных сетей, а также поддерживает многоадресные и широковещательные потоки.

Связь ближнего действия (NFC)

Набор протоколов связи, касающихся использования электромагнитных полей. Позволяет двум устройствам обмениваться данными на расстоянии четырех сантиметров друг от друга. Устройства с поддержкой NFC работают как карты ключей удостоверений и обычно используются для смарт-карт, бесконтактных мобильных платежей и бронирования билетов.

Радиочастотная идентификация (RFID)

Использует электромагнитные поля для отслеживания электронных тегов, сведения о которых нельзя получить иначе. Совместимое оборудование обеспечивает электропитание и взаимодействует с этими тегами, считывая информацию для идентификации и проверки подлинности.

Wi-Fi/802.11

Стандартный вариант для дома и работы. Этот вариант экономичен. Но он подходит не для всех сценариев из-за ограниченного диапазона действия и постоянного потребления энергии.

Протокол MQTT

MQTT (Message Queuing Telemetry Transport) - это протокол, сделанный конкретно для IoT. Открытый и простой он предназначен для обмена информацией между разными устройствами и модулями. Упрощает соединение каналов связи быстро, качественно и своевременно. Отвечает за безопасность соединения, скорость передачи данных и практическое функционирование систем и программ. Защищает от всевозможных сбоев и неполадок, качественно выполняя свою работу. Спектр возможностей этого протокола очень большой. Он позволяет обмениваться информацией между более масштабными "предметами", а также выполняет систематизацию локальных сетей в интернете.

MQTT протокол состоит из MQTT-брокера, MQTT-агентов подписчиков и исполнителей. Все они четко знают и выполняют запрограммированную

задачу, работая четко и слаженно. Исполнители занимаются публикацией данных предназначенных для подписчиков. Это их основная функция, без которой соединение не будет работать.

Вычислительные потребности для протокола MQTT очень маленькие, потому что он рассчитан на вмонтированные устройства с низкой мощностью. Даже если в сетях низкая пропускная возможность, MQTT сохраняет высокую качественность связи и практически не перегружает работу системы. Это один из основных плюсов этого протокола. В структуре данных, которые передаёт протокол почти нет функциональной информации, по сравнению с другими протоколами связи. Что характеризует эго с качественной стороны. К примеру HTTP передаёт все служебные данные, но в этом нет никакой срочной необходимости.

Сделав измерения в 3G-сетях и детально проверив все процессы, стало известно, что MQTT имеет в 93 раза большую пропускную возможность нежели REST (Representational State Transfer), который выполняет свою работу поверх HTTP.

Этот протокол работает по принципу "издатель-подписчик", при этом прибегая к минимальному количеству вариантов воплощения задач. Это улучшает и ускоряет функционирование самого протокола. Способы дают указания на задачи, требующие выполнения. Все они осуществляются с помощью сотрудничества с брокером и ведут к работе с разными темами и сигналами. Затем выполняющие агенты устанавливают связь с брокером и либо делают публикации сообщений и тем там, либо осуществляют подписку на темы и потом получают уведомления, которые есть в этих темах.

Примеры возможностей MQTT:
- connect (установление связи с брокером);
- disconnect (прерывание связи с брокером);
- subscribe (подписка на тему на брокере);
- unsubscribe (отписка от темы на брокере);
- publish (публикование своей темы на брокере).

Источником некой информации является публикатор, задача которого соединится с брокером, а после качественно и быстро передать нужные данные. Затем подписчик, который есть потребителем, работает по аналогичной схеме и осуществляет подписку на тему, в качестве которой мы здесь можем увидеть "/home/alarms/1/status". На примере этой темы осуществляется публикация данных об положении домашней сигнализации в

некоторой "зоне 1". Если к издателю поступает новая информация он передаёт её брокеру и осуществляет публикацию сообщения в этой теме. А задача брокера, распространить сообщение всем подписчикам этой темы.

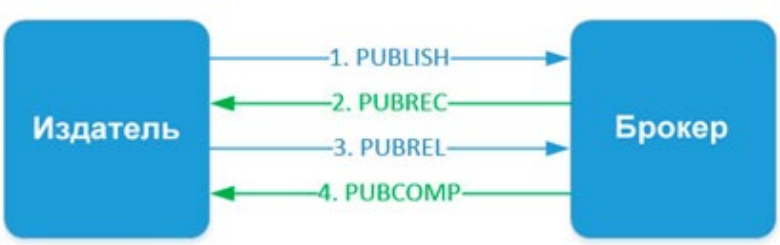
Как можно заметить, структура темы приведённой в пример есть иерархической. Этим она имеет схожесть с путём файловой системы, и возможность упростить организацию тем. А ещё такие иерархические структуры очень востребованы и популярны также и в других протоколах, к примеру в REST.

MQTT, протокол для начинающих, помимо всего вышеперечисленного разрешает использование разных символов. Что существенно отражается на процессе подписки, упрощая её выполнение.

QoS 0 At most once. На этом уровне издатель один раз отправляет сообщение брокеру и не ждет подтверждения от него, то есть отправил и забыл.

QoS 1 At least once. Этот уровень гарантирует, что сообщение точно будет доставлено брокеру. но есть вероятность дублирования сообщений от издателя. После получения дубликата сообщения, брокер снова рассылает это сообщение подписчикам, а издателю снова отправляет подтверждение о получении сообщения. Если издатель не получил PUBACK сообщения от брокера, он повторно отправляет этот пакет, при этом в DUP устанавливается "1".

QoS 2 Exactly once. На этом уровне гарантируется доставка сообщений подписчику и исключается возможное дублирование отправленных сообщений.



Издатель отправляет сообщение брокеру. В этом сообщении уникальные Packet ID, QoS=2 и DUP=0. Издатель хранит сообщение неподтвержденным пока не получит от брокера ответ PUBREC. Брокер отвечает сообщением PUBREC в котором содержит тот же Packet ID. После его получения издатель отправляет PUBREL с тем же Packet ID. До того, как брокер получит PUBREL он должен хранить копию сообщения у себя. После получения PUBREL он удаляет копию сообщения и отправляет издателю сообщения PUBCOMP о том, что транзакция завершена.

ПРОГРАММНАЯ ЧАСТЬ КОМПЛЕКСА

Интерфейс программы и управление

 — Левая кнопка мыши – действие;

 — Правая кнопка мыши – перемещение камеры по экранной плоскости;

 — Вращение колеса мыши – приближение\отдаление от экранной плоскости;

 + W — передвижение вперед по сцене;

 + S — передвижение назад по сцене;

 + A — передвижение влево по сцене;

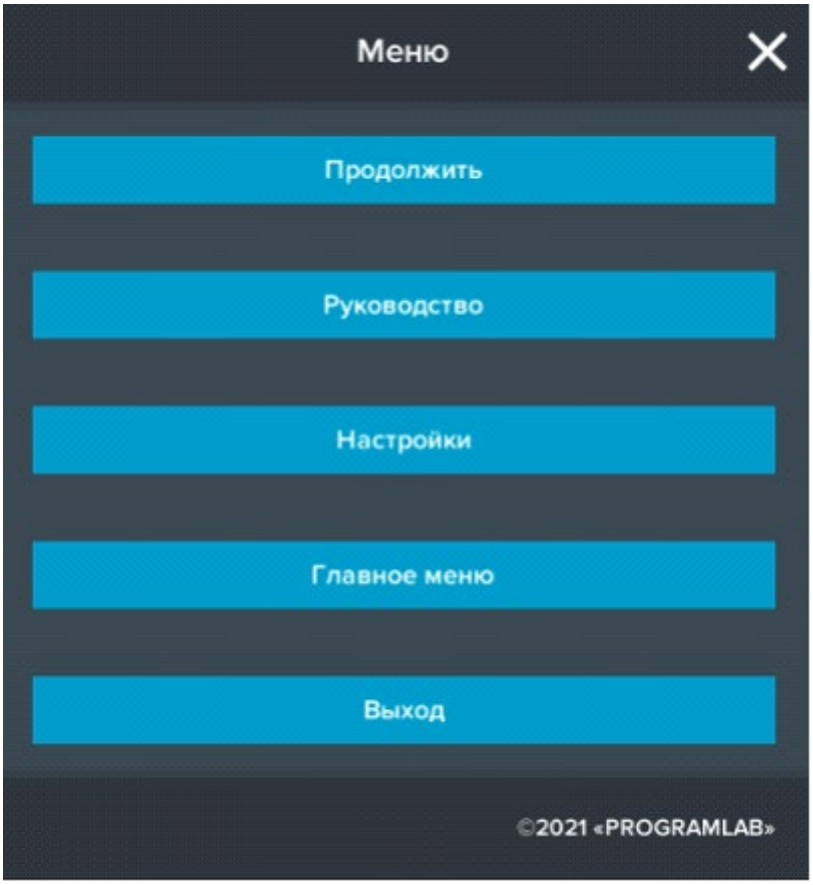
 + D — передвижение вправо по сцене;

 + E — передвижение вверх на сцене;

 + Q — передвижение вниз на сцене;

Alt +  — вращение вокруг точки;

Esc — Вызов меню программы.



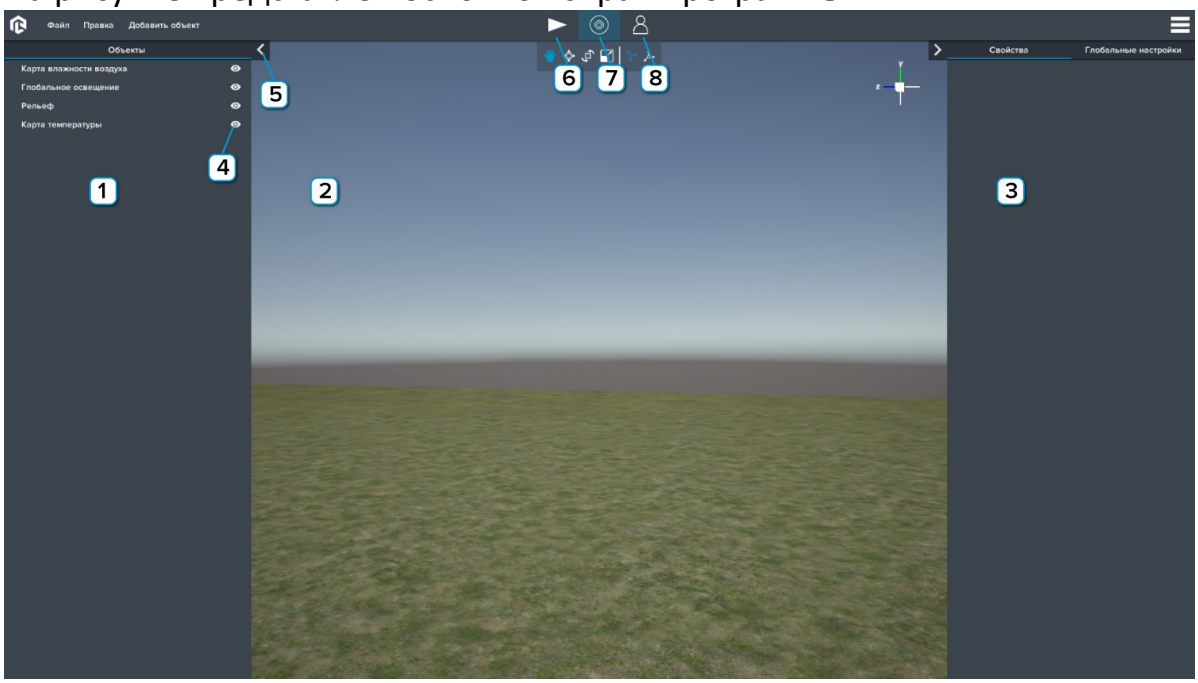
Кнопка **«Продолжить»** – вернуться в программу;
Кнопка **«Руководство»** – вызвать руководство пользователя;
Кнопка **«Настройки»** – настройки параметров графики;
Кнопка **«Главное меню»** – выход в главное меню;
Кнопка **«Выход»** – выход из программы.

– Изменение настроек графики нажмите кнопку

– Выход из программы

РАБОТА С КОМПЛЕКСОМ

На рисунке представлен основной экран программы



Интерфейс

1 – поле объектов;

2 – рабочее поле;

3 – поле свойств;

4 – нажмите чтобы скрыть объект;

5 – нажмите чтобы скрыть поле;

6 – запуск/остановка проекта;

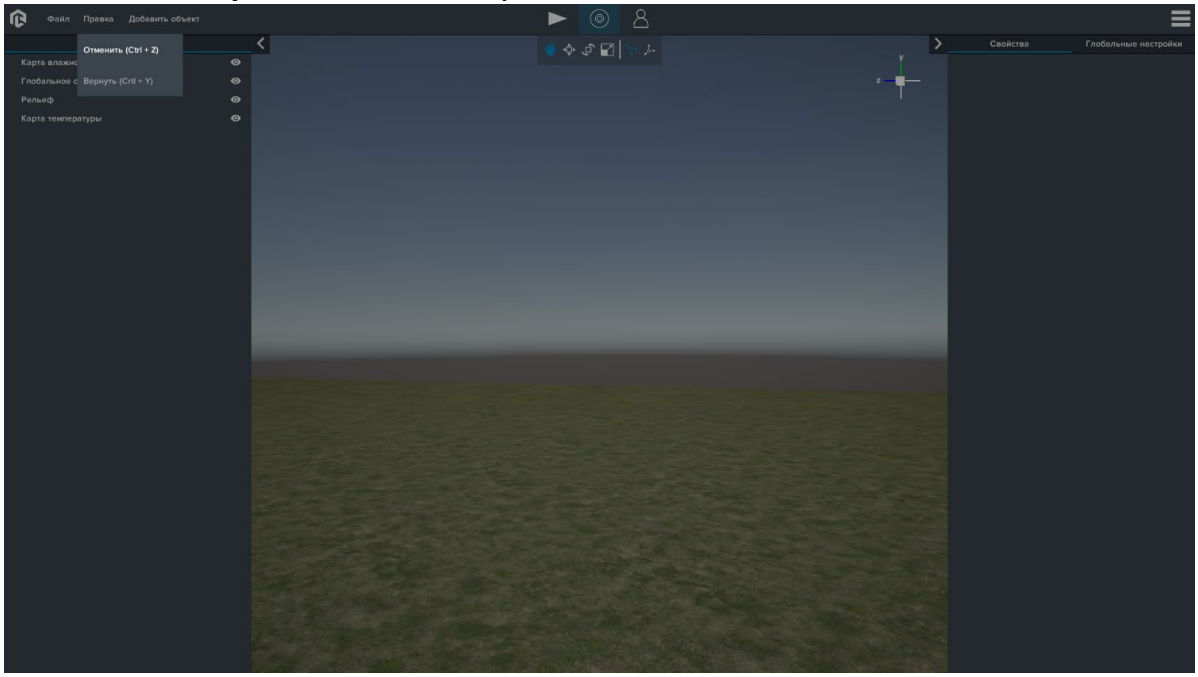
7 – управление от третьего лица;

8 – управление от первого лица.

Во вкладке **"Файл"** есть следующие возможности:

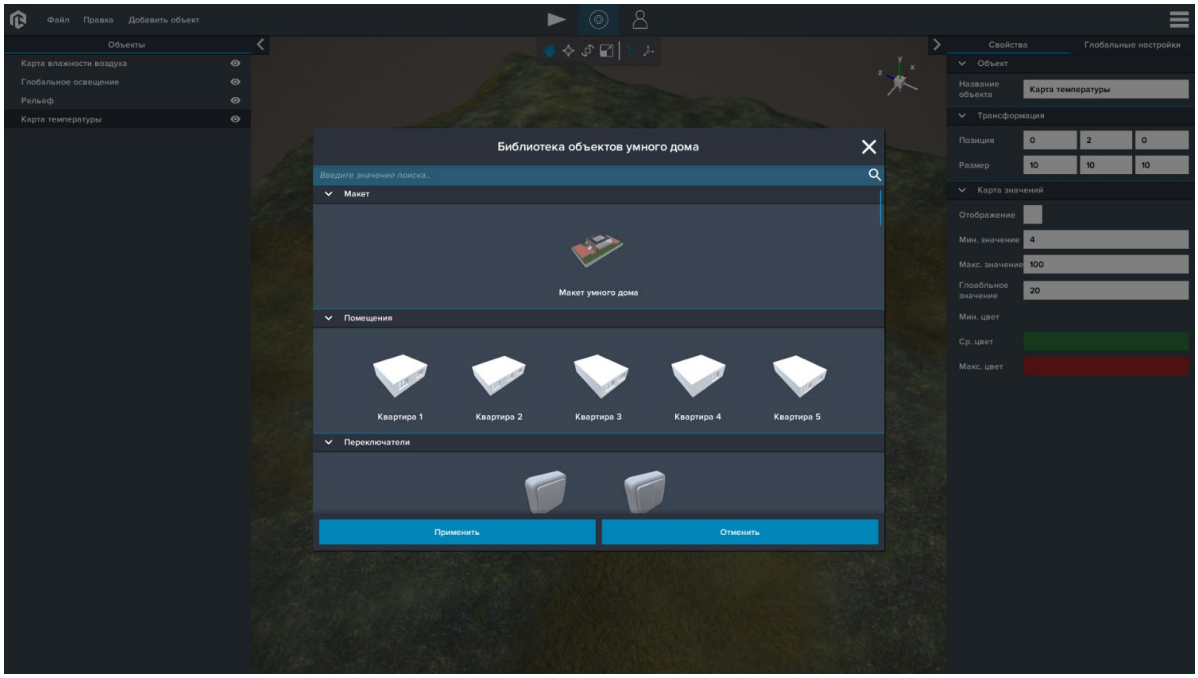
Вкладка Файл

Во вкладке **"Правка"** есть следующие возможности:



Вкладка Правка

Нажмите **"Добавить объект"**:

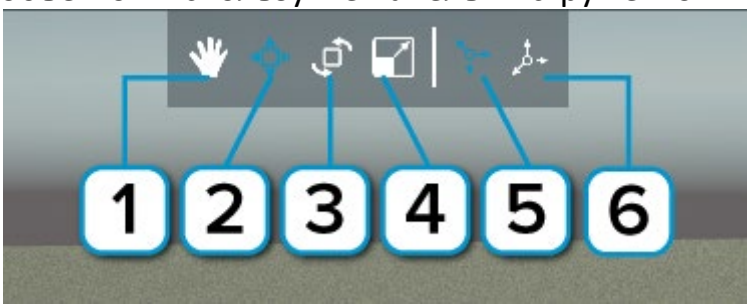


Добавление объекта

В поле свойств заполните компоненты:

Свойства объекта

Для работы с объектом используйте панель инструментов:



Панель инструментов

1 – инструмент курсор;

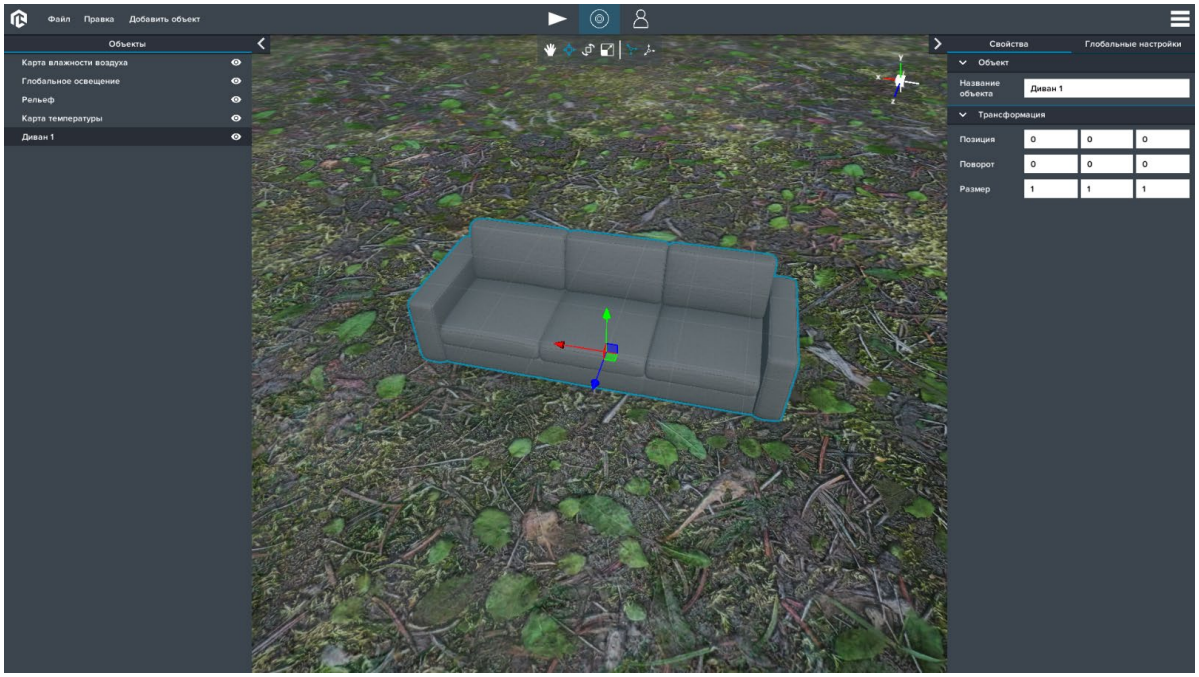
2 – инструмент передвижения объектов;

3 – инструмент поворота объектов;

4 – инструмент масштабирования объектов;

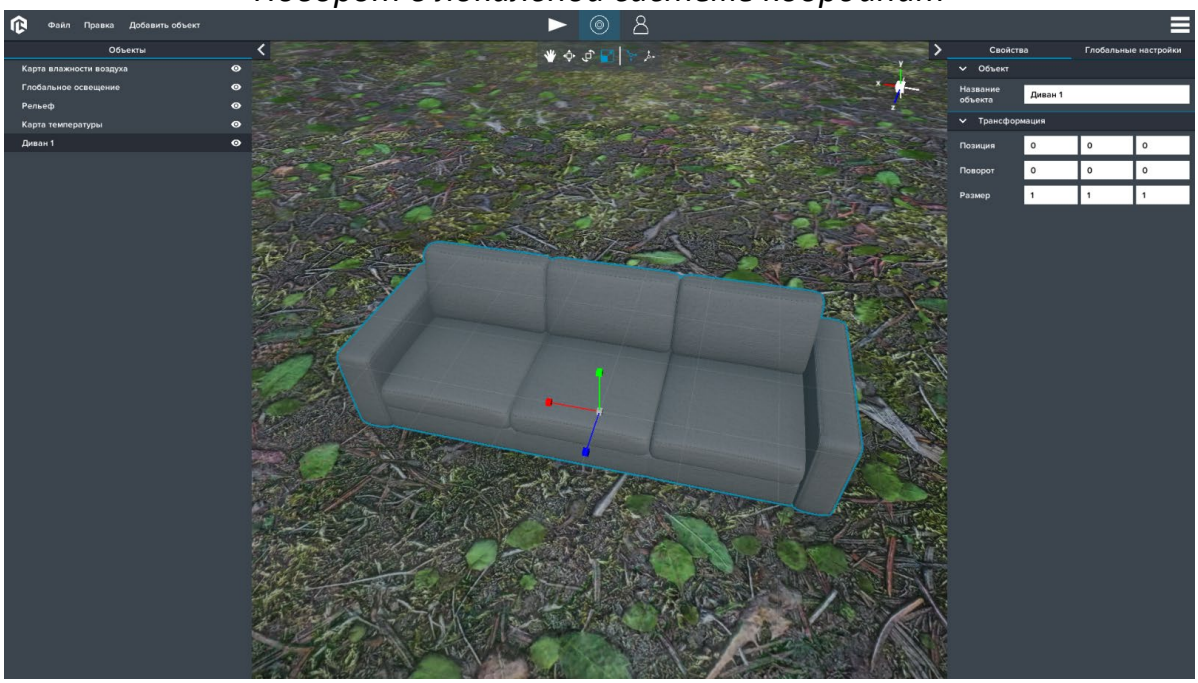
5 – локальная система координат;

6 – глобальная система координат.



Перемещение в локальной системе координат

Поворот в локальной системе координат

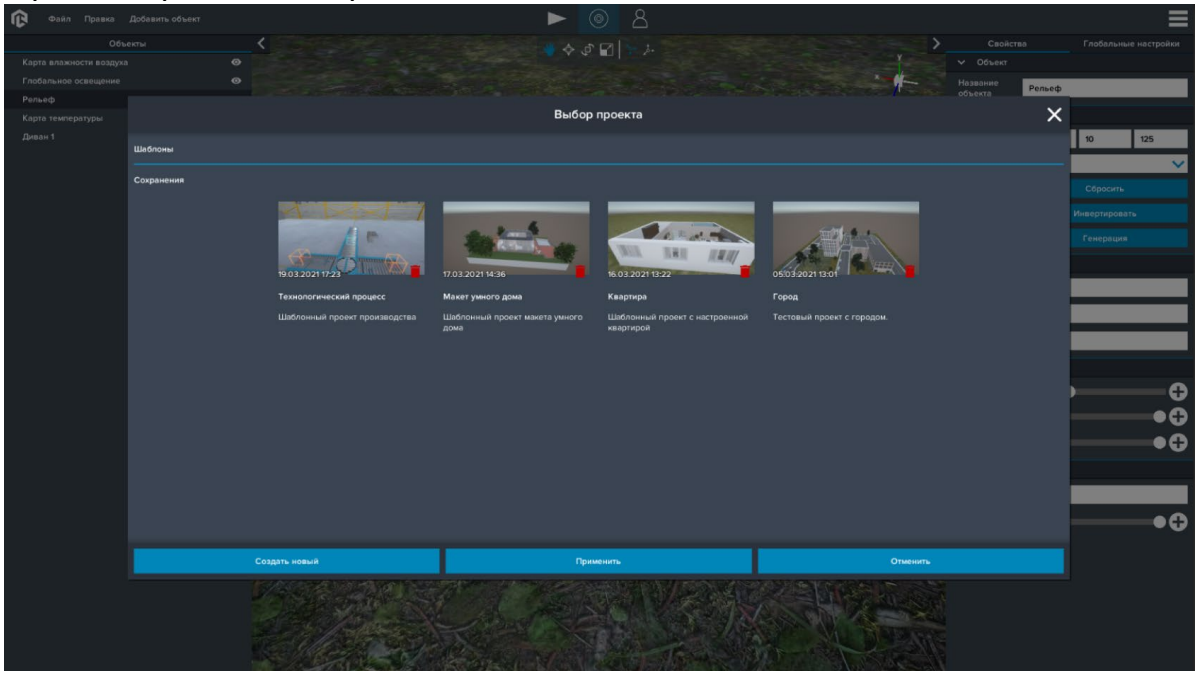


Масштабирование в локальной системе координат

При нажатии правой кнопкой мыши на объект в поле объектов появляются следующие возможности:

Свойства

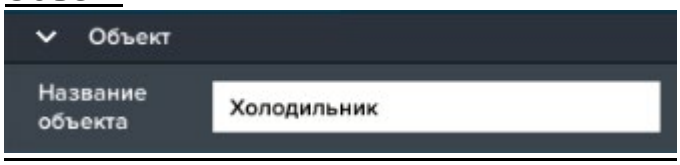
При нажатии вкладке "Файл" и выбора функции "Загрузить" можно выбирать сохраненные проекты или шаблон.



Проекты

Компоненты

Объект

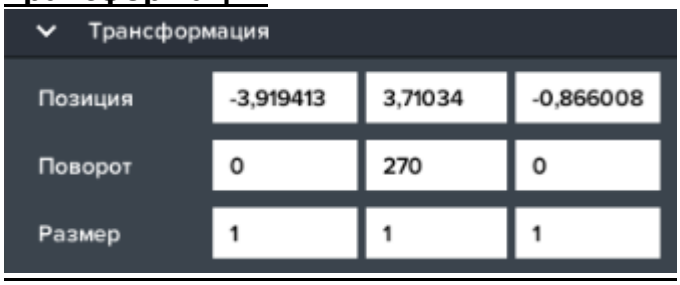


Компонент отвечает за изменение названия объекта.

Параметры:

• Название объекта.

Трансформация



Компонент отвечает за изменение положения объекта в пространстве. Можно задавать положение, вращение и размер объекта по трем осям: X, Y, Z.

Параметры:

• Позиция;

• Поворот;

• Размер.

Свет

Компонент отвечает за источник света.

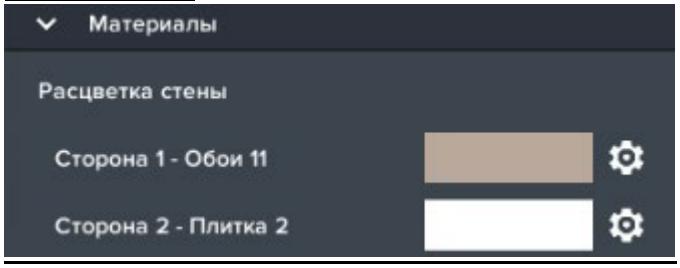
Параметры:

• Статус;

• Интенсивность;

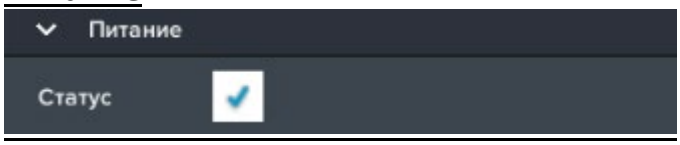
• Цвет.

Материалы



Компонент отвечает за изменение материалов и окраску объектов. Отображается название части объекта, его цвет и кнопка изменения материала. При нажатии на данную кнопку открывается библиотека доступных материалов для данного объекта.

Питание



Компонент отвечает за изменение состояния источника питания.

Параметры:

• Статус.

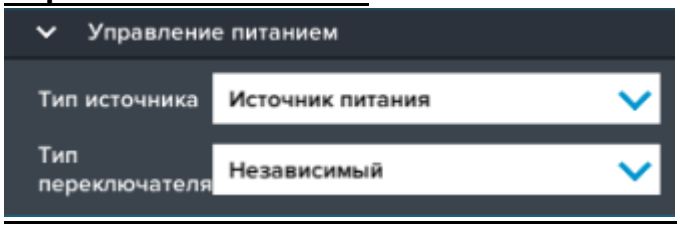
Переключатель

Компонент отвечает за изменение состояния переключателя.

Параметры:

• Статус.

Управление питанием



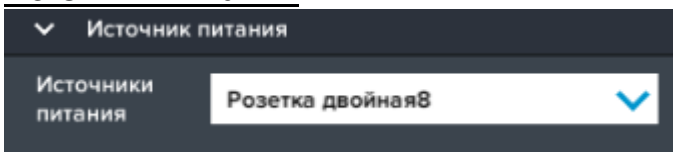
Компонент отвечает за выбор типа источника питания и типа переключателя.

Параметры:

• Тип источника питания: Независимый – объект всегда получает питание, Источник питания – объект получает питание из выбранного источника питания в появившемся компоненте ***Источник питания.***

• Тип переключателя: Независимый – объект переключается в режиме воспроизведения при нажатии по нему курсором, Переключатель – объект переключается при помощи выбранного переключателя в появившемся компоненте ***Источник переключателя***, MQTT – объект переключается при помощи MQTT сообщений из вписанных пользователем топика в появившемся компоненте ***MQTT – Переключатель***.

Источник питания

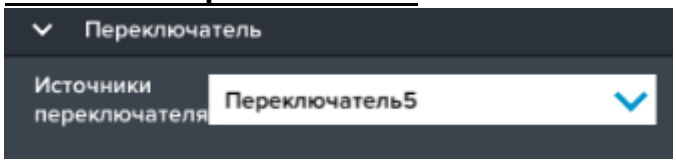


Компонент отвечает за выбор источника питания.

Параметры:

• Источник питания – список доступных источников питания.

Источник переключателя



Компонент отвечает за выбор источника переключателя.

Параметры:

• Источник переключателя – список доступных переключателей.

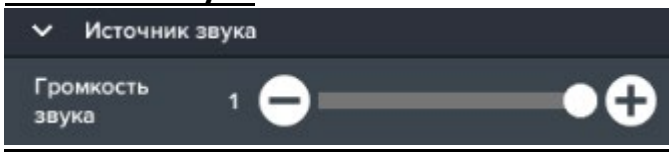
Устройство

Компонент отвечает за изменение состояния активности устройства.

Параметры:

• Статус.

Источник звука

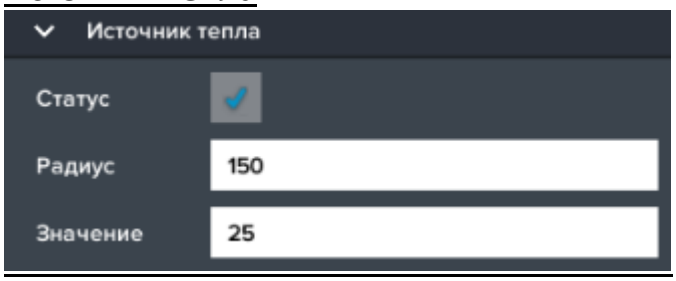


Компонент отвечает за изменение громкости звуков.

Параметры:

• Громкость звука.

Источник тепла



Компонент отвечает за изменение температуры вокруг объекта.

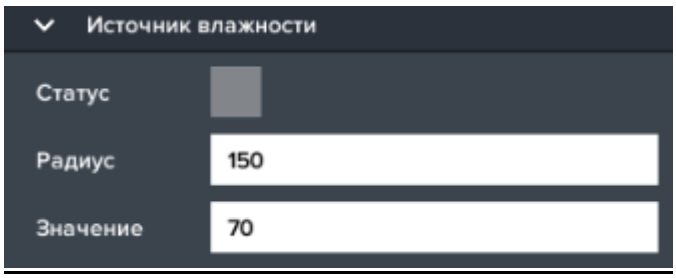
Параметры:

• Статус;

• Радиус;

• Значение.

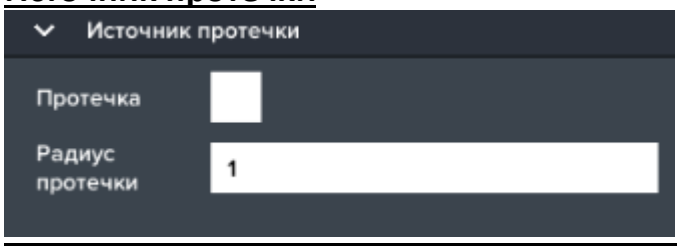
Источник влажности



Компонент отвечает за изменение влажности воздуха вокруг объекта.
Параметры:

- Статус;
- Радиус;
- Значение.

Источник протечки



Компонент отвечает за генерацию условия возникновения протечки.
Параметры:

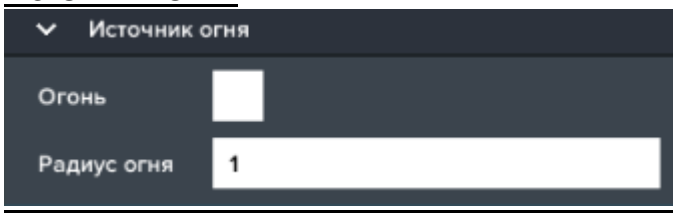
- Статус;
- Радиус.

Источник дыма

Компонент отвечает за генерацию условия возникновения дыма.
Параметры:

- Статус;
- Радиус.

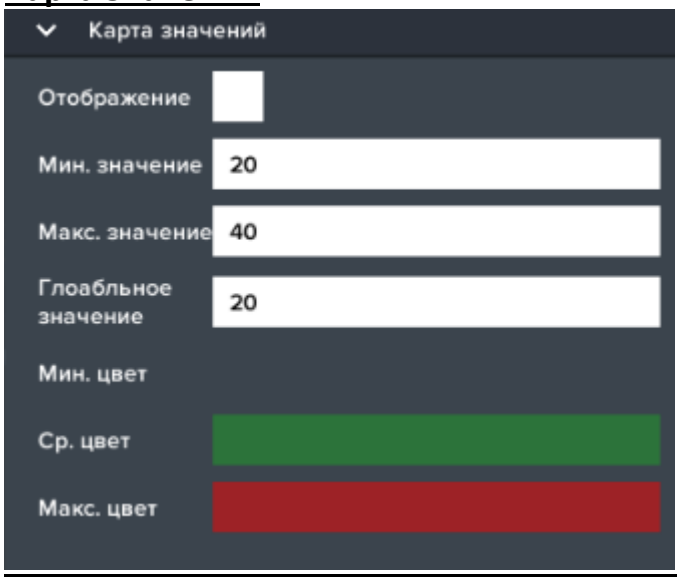
Источник огня



Компонент отвечает за генерацию условия возникновения огня.
Параметры:

- Статус;
- Радиус.

Карта значений

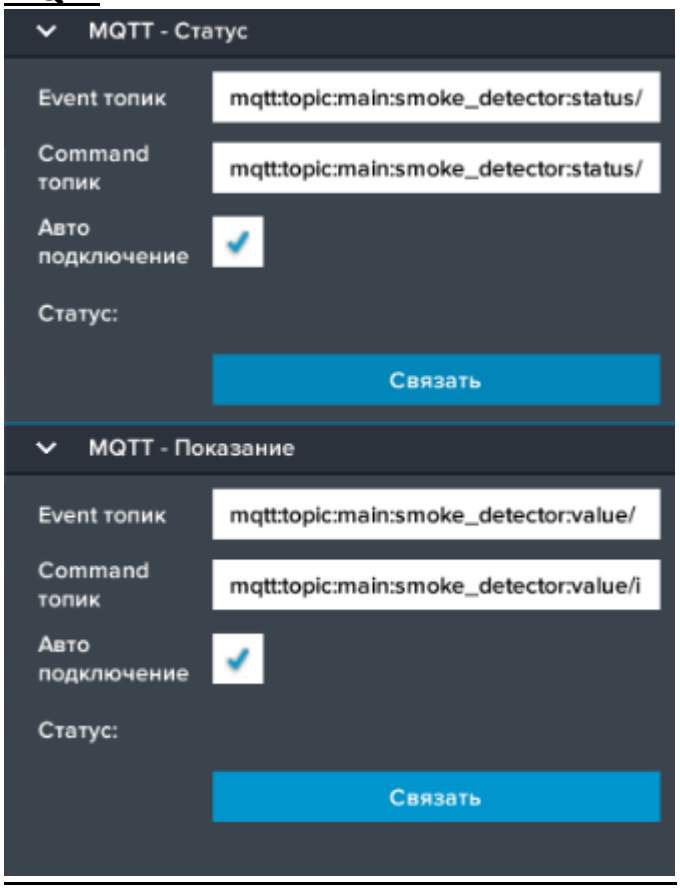


Компонент отвечает за управление картами значений, таких как ***Карта влажности воздуха*** и ***Карта температуры.***

Параметры:

- Отображение;
- Минимальное значение;
- Максимальное значение;
- Глобальное значение;
- Цвет минимального значения;
- Цвет среднего значения;
- Цвет максимального значения.

MQTT



Компонент отвечает за подключение по MQTT протоколу к платформам интернета вещей.

Параметры:

• Event топик – топик отвечает за отправку данных от виртуального устройства к облаку.

• Command топик – топик отвечает за получение данных от облака к виртуальному устройству.

• Авто подключение – служит для автоматического подключения топиков при загрузке проекта.

• Статус – состояние подключения.

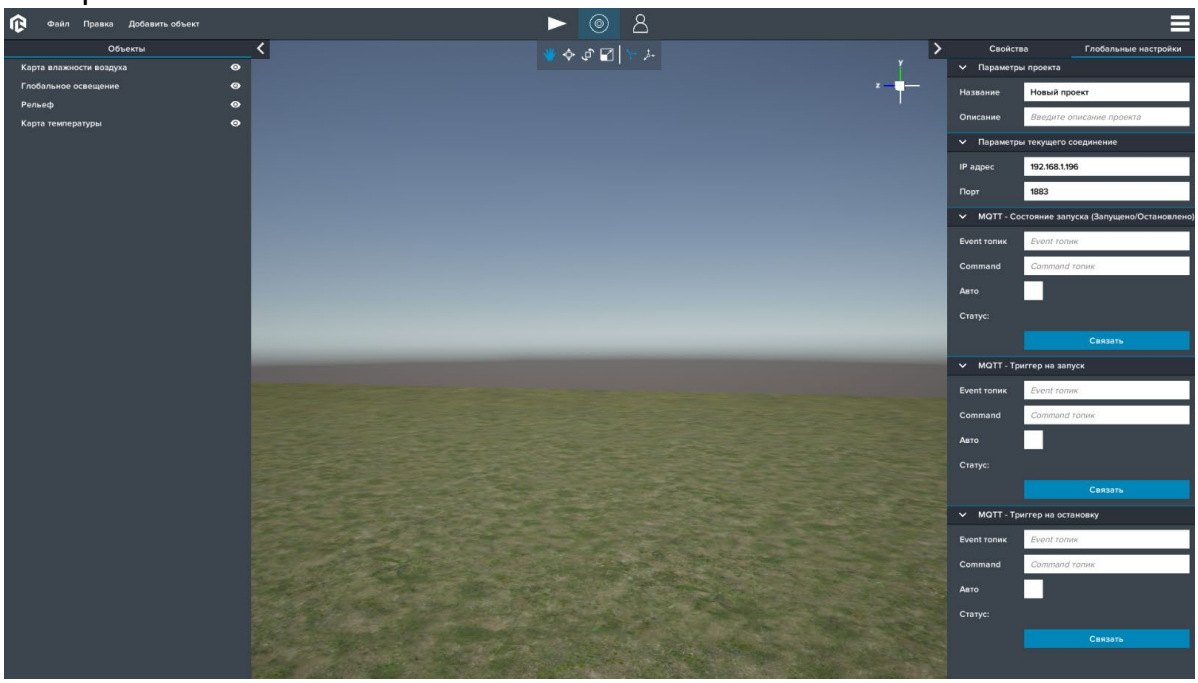
• JSON – пример структуры сообщения.

Кнопки:

• Связать – служит для подключения к MQTT топикам.

Глобальные настройки

В поле свойств во вкладке "Глобальные настройки" введите необходимые параметры:



Название проекта

Параметры проекта – необходим для редактирования названия и описания проекта.

Параметр	Описание	Значение
Название	Название проекта	string
Описание	Описание проекта	string

Параметры текущего соединение – необходим для подключения к MQTT брокеру.

Параметр	Описание	Значение
IP адрес	IP адрес или имя хоста брокера MQTT	string
Порт	Порт брокера MQTT	string

MQTT – состояние запуска (Запущенно/Остановлено) – необходим для отправки состояния запуска проекта при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

MQTT – триггер на запуск – необходим для отправки триггера на запуск проекта при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

MQTT – Триггер на остановку – необходим для отправки триггера на остановку проекта при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Глобальные объекты

В поле объектов настройте карту влажности воздуха, заполняя следующие компоненты:

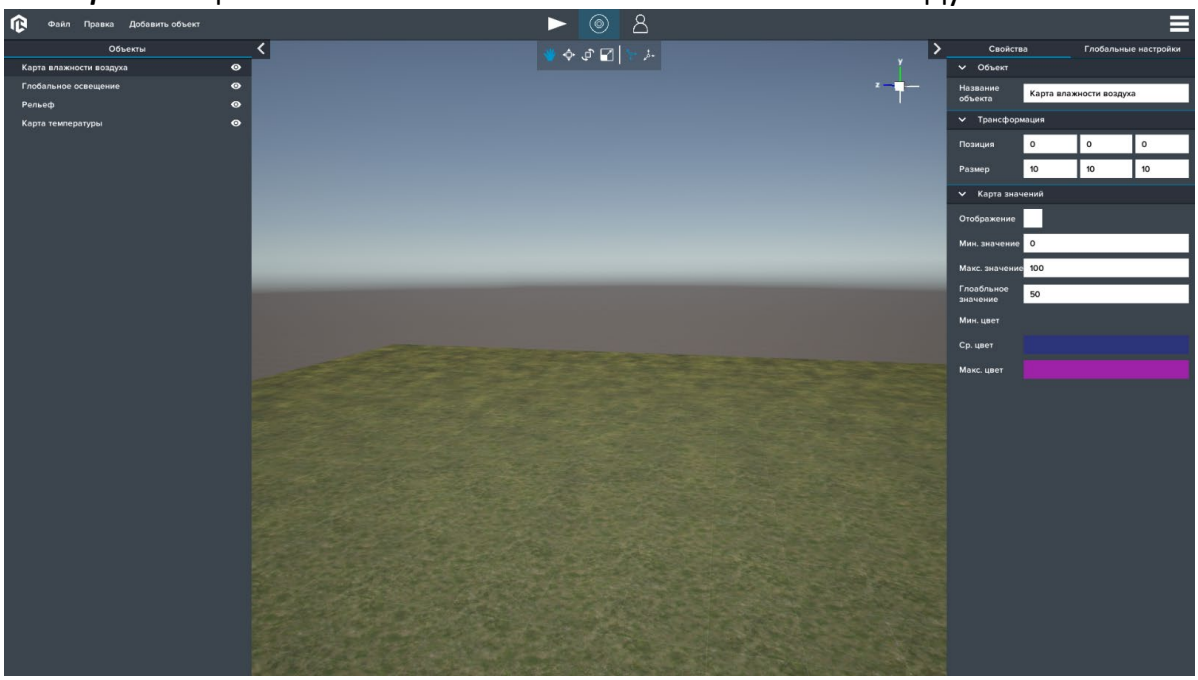
Объект – компонент отвечает за изменение названия объекта;

Трансформация – компонент отвечает за изменение положения объекта в пространстве.

Можно задавать положение и размер объекта по трем осям: X, Y, Z;

Карта значений – компонент отвечает за управление картами значений:

- ***Отображение*** – включение/отключения отображения влажности воздуха;
- ***Мин. значение*** – минимальное значение влажности воздуха;
- ***Макс. значение*** – максимально значение влажности воздуха;
- ***Глобальное значение*** – глобальное значение влажности воздуха;
- ***Мин. цвет*** – цвет минимального значения влажности воздуха;
- ***Ср. цвет*** – цвет среднего значения влажности воздуха;
- ***Макс. цвет*** – цвет максимального значения влажности воздуха.



Карта влажности воздуха

Для смены цвета нажмите на строку "Цвет":

Изменение цвета

В поле объектов настройте глобальное освещение, заполняя следующие компоненты:

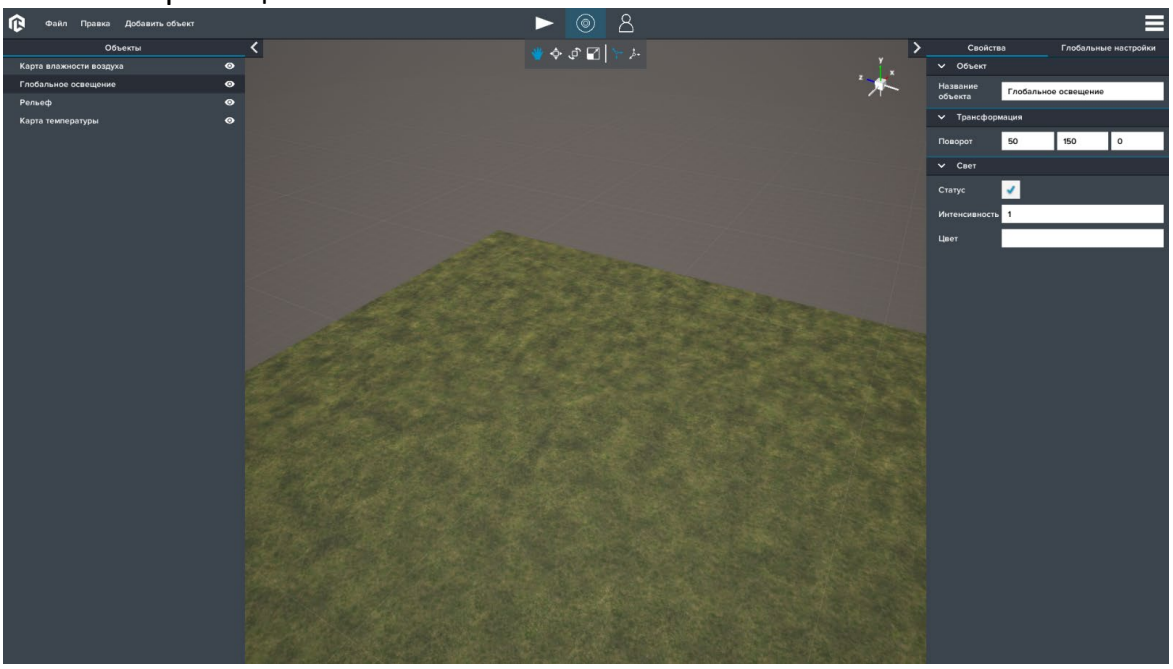
Объект – компонент отвечает за изменение названия объекта;

Трансформация – компонент отвечает за изменение положения объекта в пространстве.

Можно задавать вращение объекта по трем осям: X, Y, Z.

Свет – компонент отвечает за источник света:

- ***Статус*** – включение/отключения отображения света;
- ***Интенсивность*** – введите интенсивность света;
- ***Цвет*** – выберите цвет света.



Глобальное освещение

В поле объектов настройте рельеф, заполняя следующие компоненты:

Объект – компонент отвечает за изменение названия объекта;

Настройки местности – компонент отвечает за местность.

- ***Размер*** – задайте размер объекта по трем осям: X, Y, Z.
- ***Разрешение*** – выберите разрешение;
- ***Сбросить*** – нажмите для сброса рельефа;
- ***Инвертировать*** – нажмите чтобы инвертировать рельеф;
- ***Генерация*** – нажмите для генерации рельефа;

Шум Перлина – компонент отвечает ;

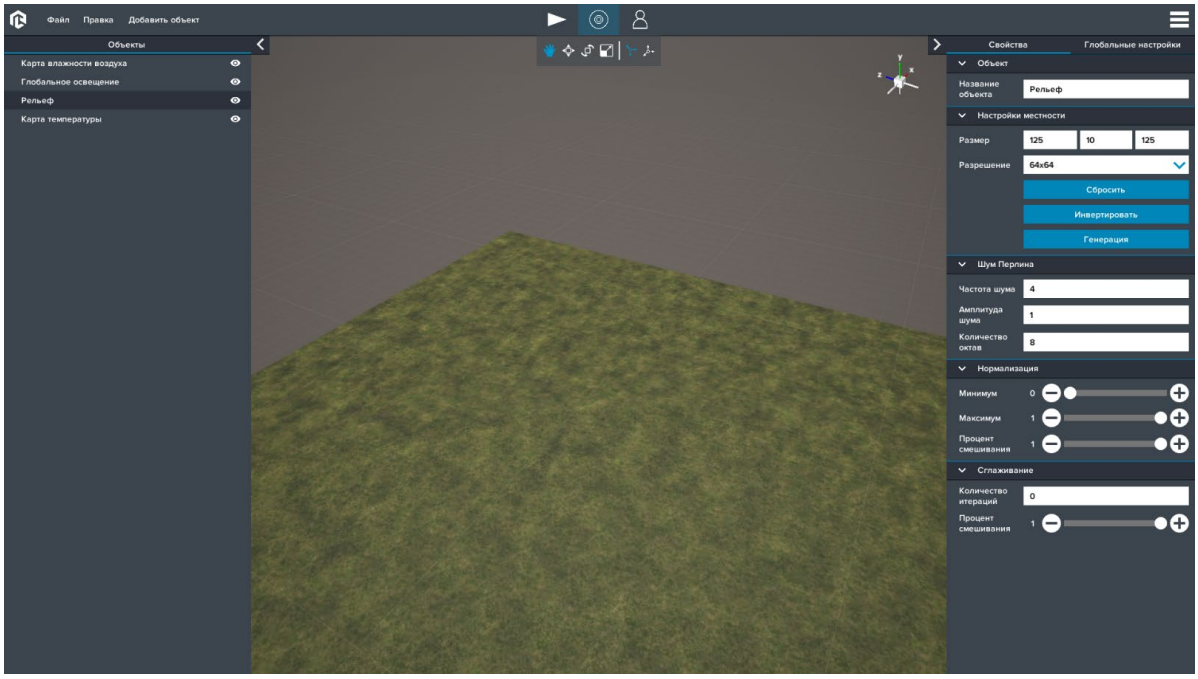
- ***Частота шума*** – установите частоту шума;
- ***Амплитуда шума*** – установите амплитуду шума;
- ***Количество актов*** – установите количество актов;

Нормализация – компонент отвечает за нормализацию объекта;

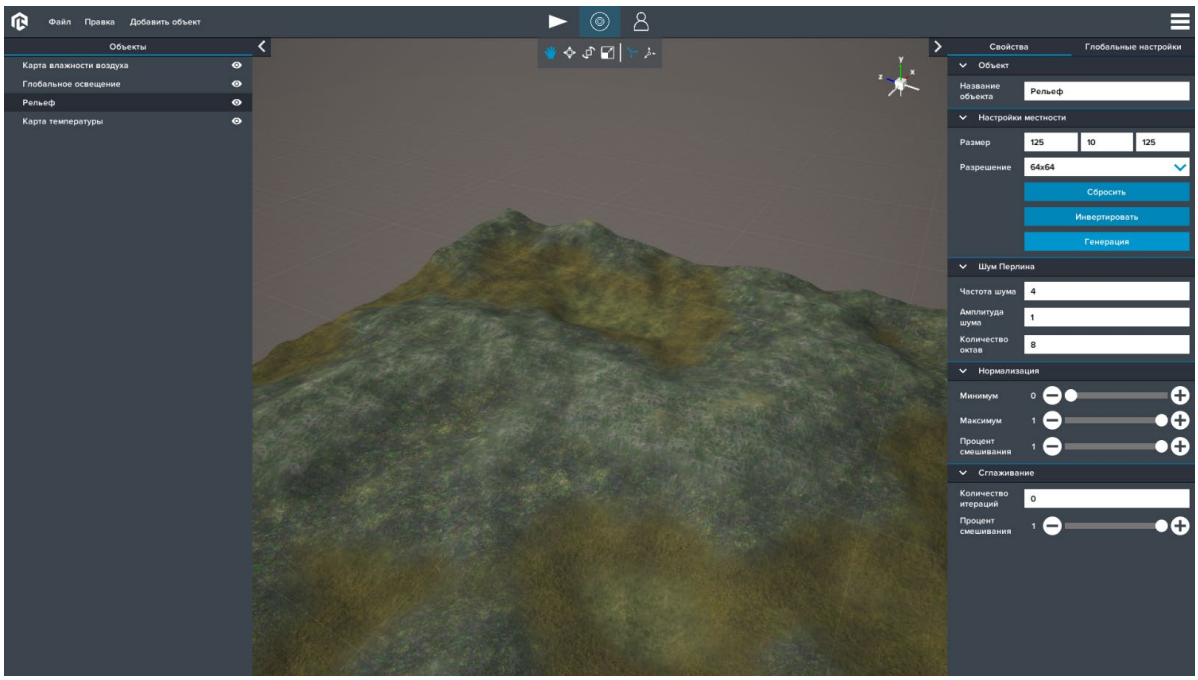
- ***Минимум*** – выставьте минимум нормализации;
- ***Максимум*** – выставьте максимум нормализации;
- ***Процент смешивания*** – выставьте процент смешивания;

Сглаживание – компонент отвечает за сглаживание объекта;

- ***Количество итераций*** – установите количество итераций;
- ***Процент смешивания*** – выставьте процент смешивания.



Свойства рельефа



Рельеф

В поле объектов настройте карту температур, заполняя следующие компоненты:

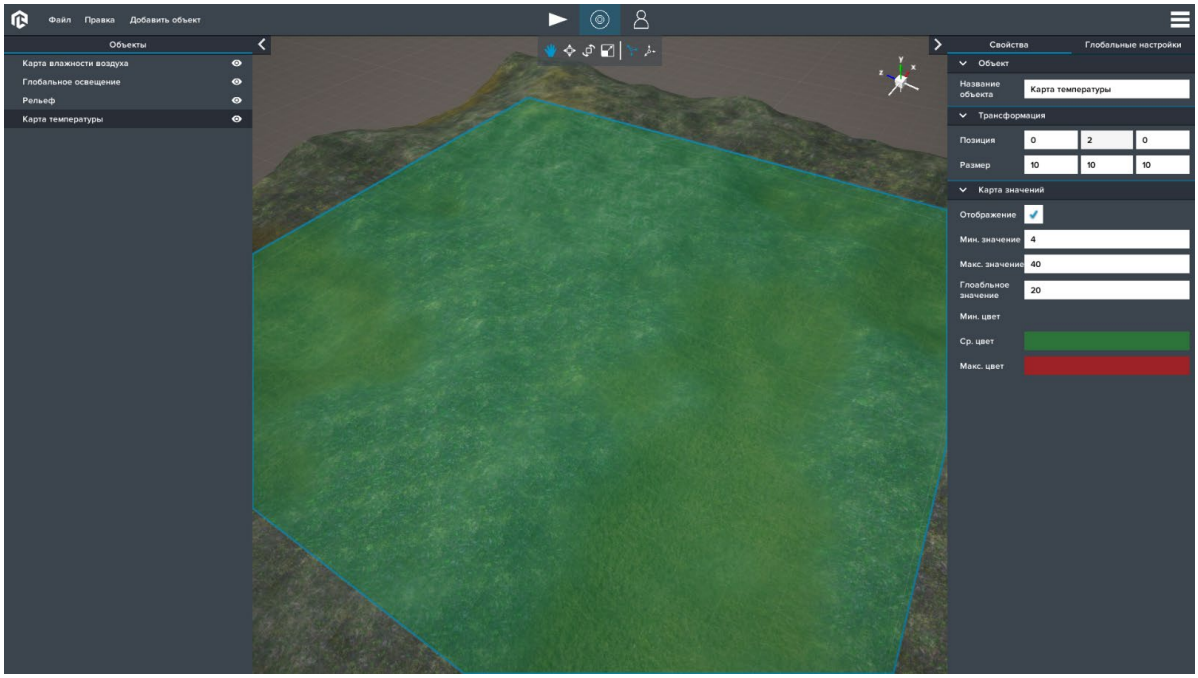
Объект – компонент отвечает за изменение названия объекта;

Трансформация – компонент отвечает за изменение положения объекта в пространстве.

Можно задавать положение и размер объекта по трем осям: X, Y, Z.

Карта значений – компонент отвечает за управление картами значений:

- ***Отображение*** – включение/отключения отображения температуры;
- ***Мин. значение*** – минимальное значение температуры;
- ***Макс. значение*** – максимально значение температуры;
- ***Глобальное значение*** – глобальное значение температуры;
- ***Мин. цвет*** – цвет минимального значения температуры;
- ***Ср. цвет*** – цвет среднего значения температуры;
- ***Макс. цвет*** – цвет максимального значения температуры.

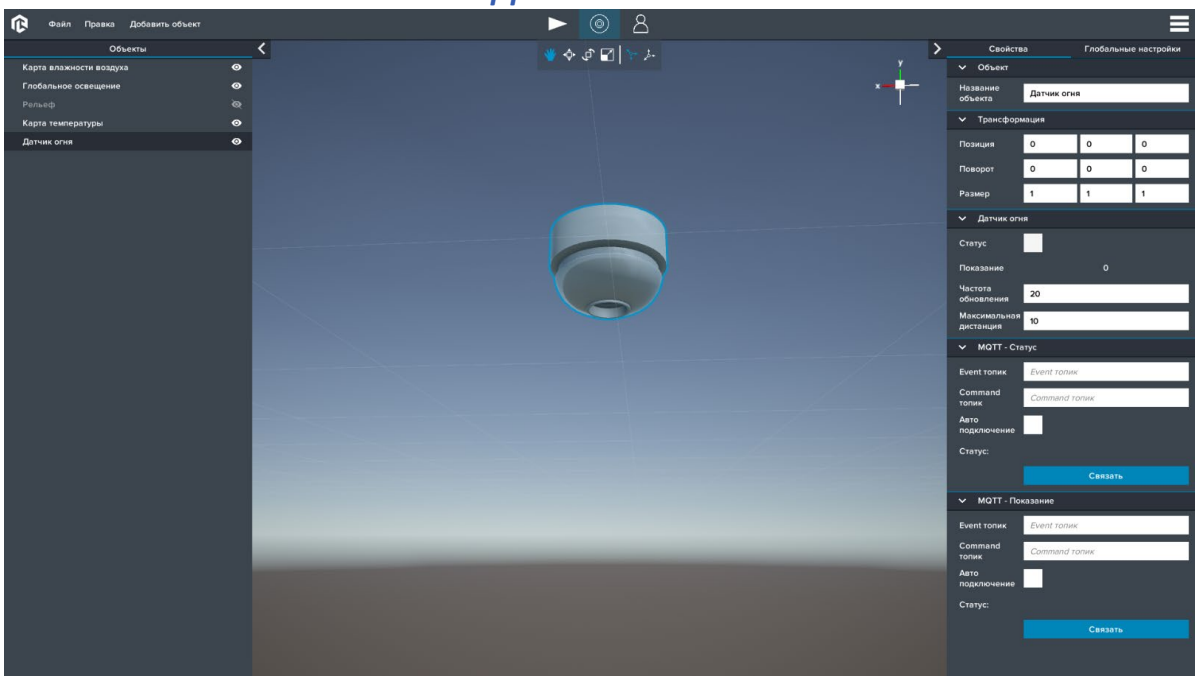


Карта температур

ОБЪЕКТЫ

Датчики

Датчик огня



Датчик огня

Компонент ***Датчик огня*** – необходим для настройки параметров работы датчика. Датчик срабатывает при обнаружении в области видимости источника огня.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	0 – нет огня 1 – есть огонь	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

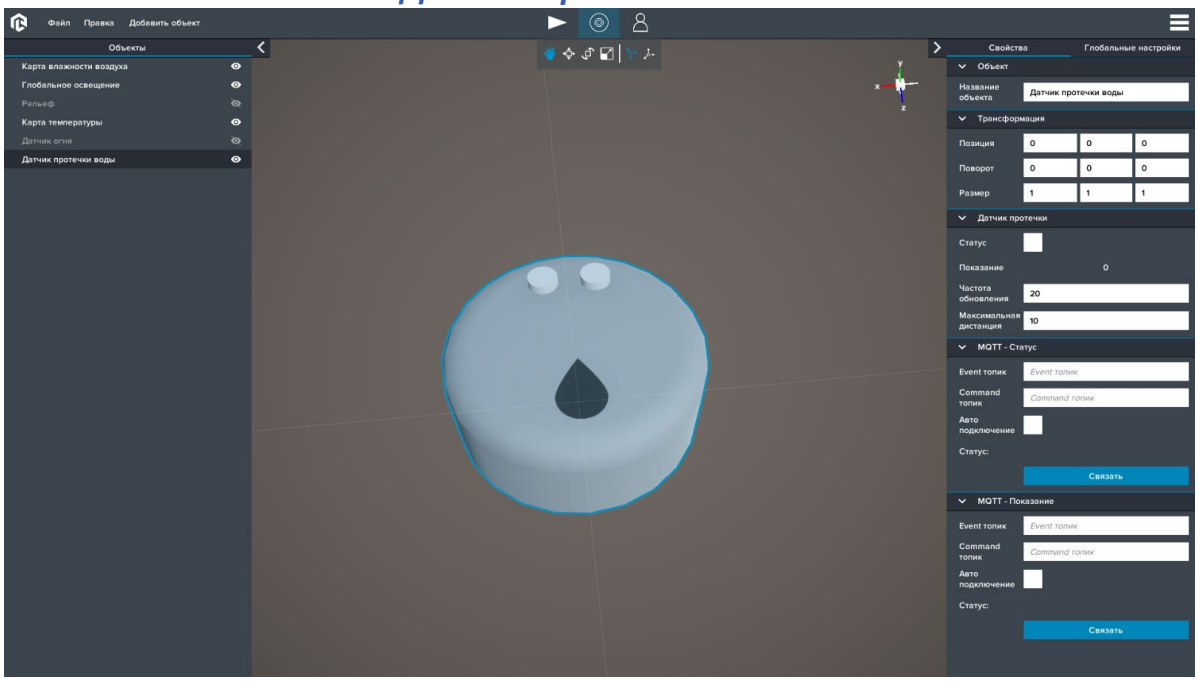
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик протечки воды



Датчик протечки воды

Компонент ***Датчик протечки воды*** – необходим для настройки параметров работы датчика. Датчик срабатывает при обнаружении в области видимости источника протечки.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	0 – нет протечки 1 – есть протечка	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

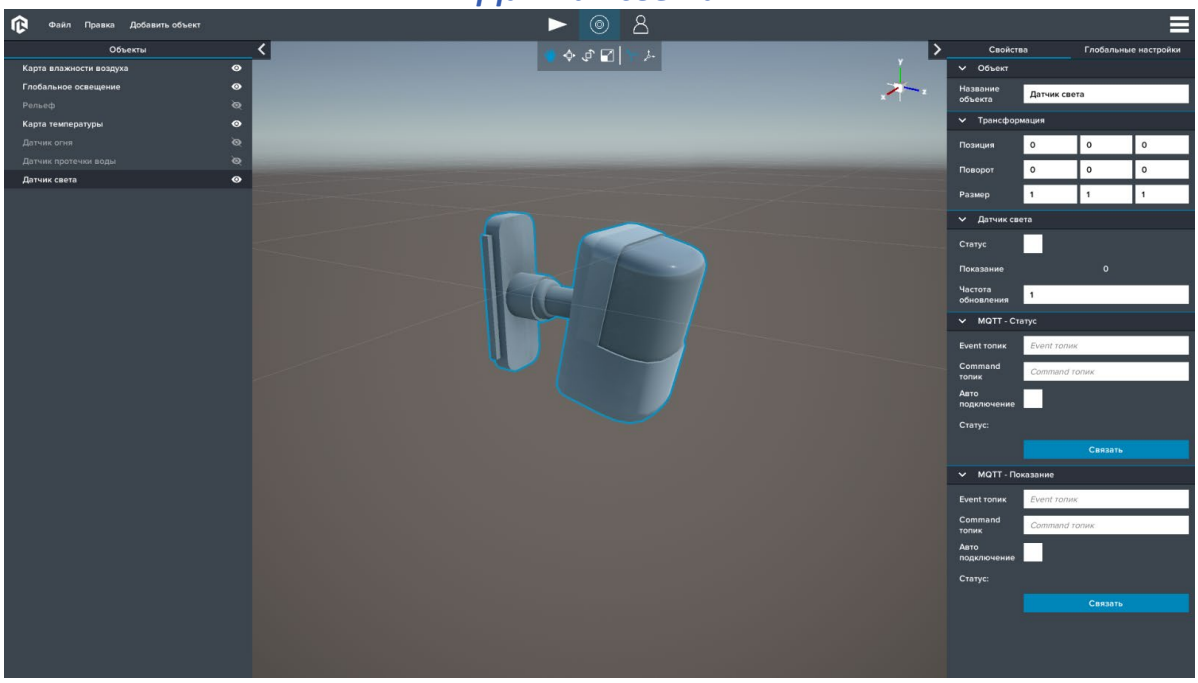
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик света



Датчик света

Компонент ***Датчик света*** – необходим для настройки параметров работы датчика. Датчик измеряет уровень освещенности в точки его местоположения.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	Значение освещенности (лм)	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

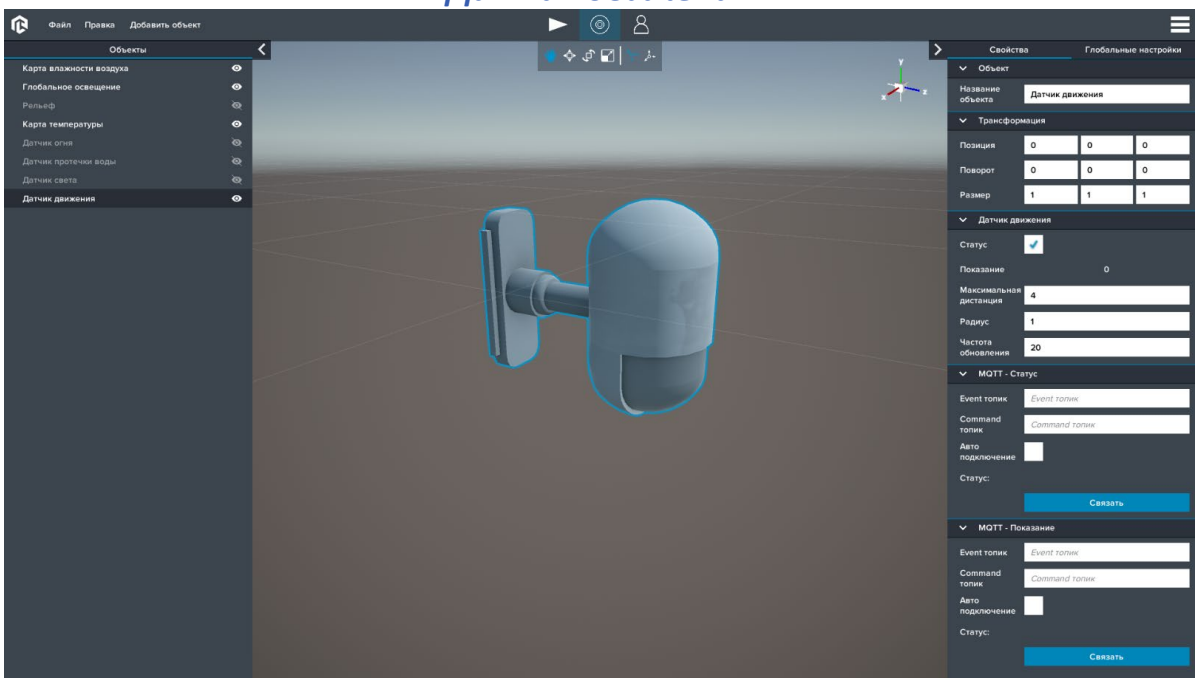
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик движения



Датчик движения

Компонент ***Датчик движения*** – необходим для настройки параметров работы датчика. Датчик срабатывает при фиксировании в области видимости перемещение объектов.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	0 – нет изменения 1 – есть изменения	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float
Радиус	Радиус действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

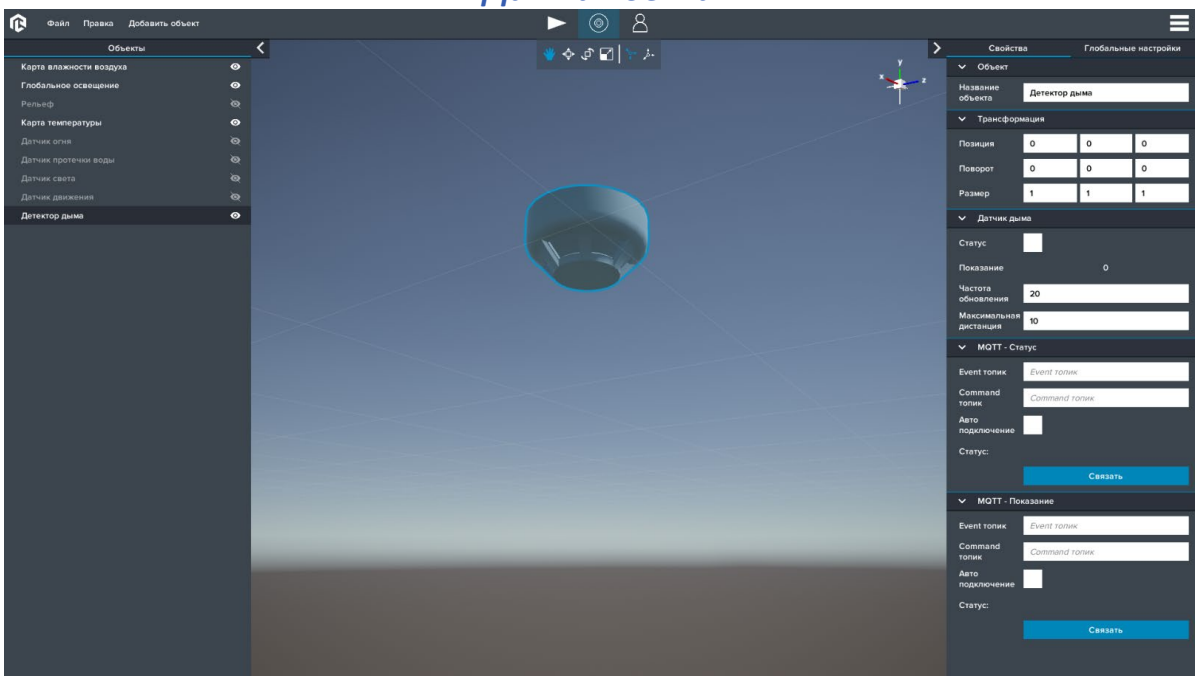
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик дыма



Датчик дыма

Компонент ***Датчик дыма*** – необходим для настройки параметров работы датчика. Датчик срабатывает при обнаружении в области видимости источника дыма.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	0 – нет дыма 1 – есть дым	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

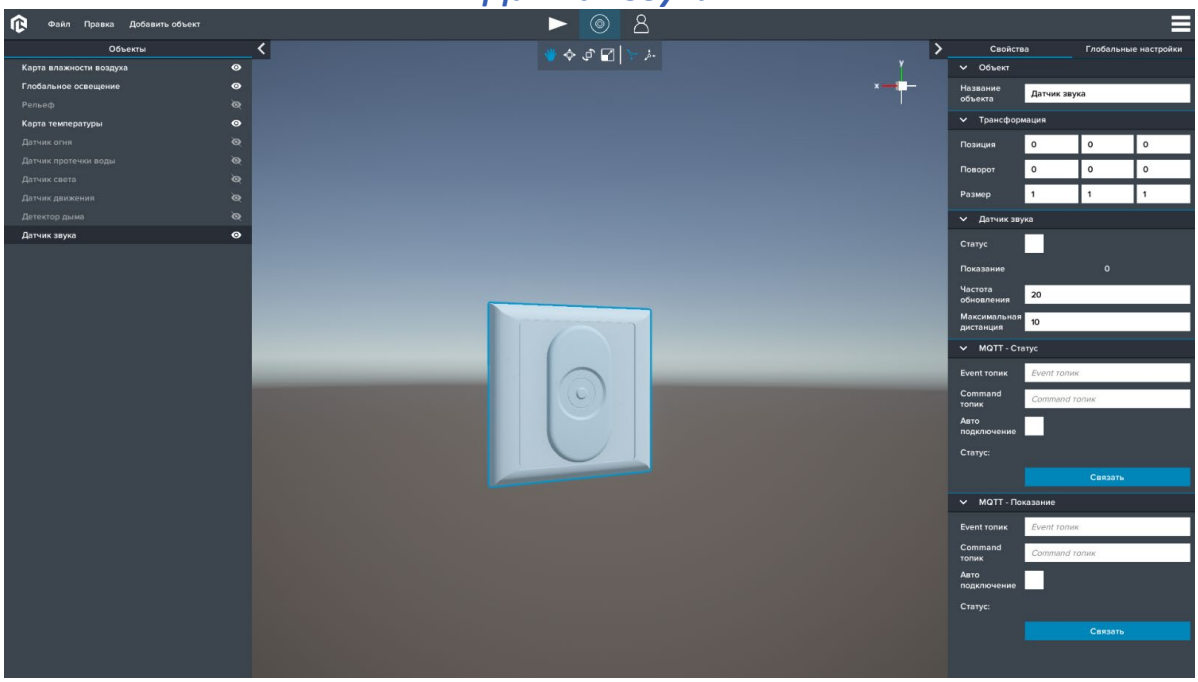
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

омпонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик звука



Датчик звука

Компонент ***Датчик звука*** – необходим для настройки параметров работы датчика. Датчик измеряет уровень шума от источников звука.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	Уровень шума (дБ)	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

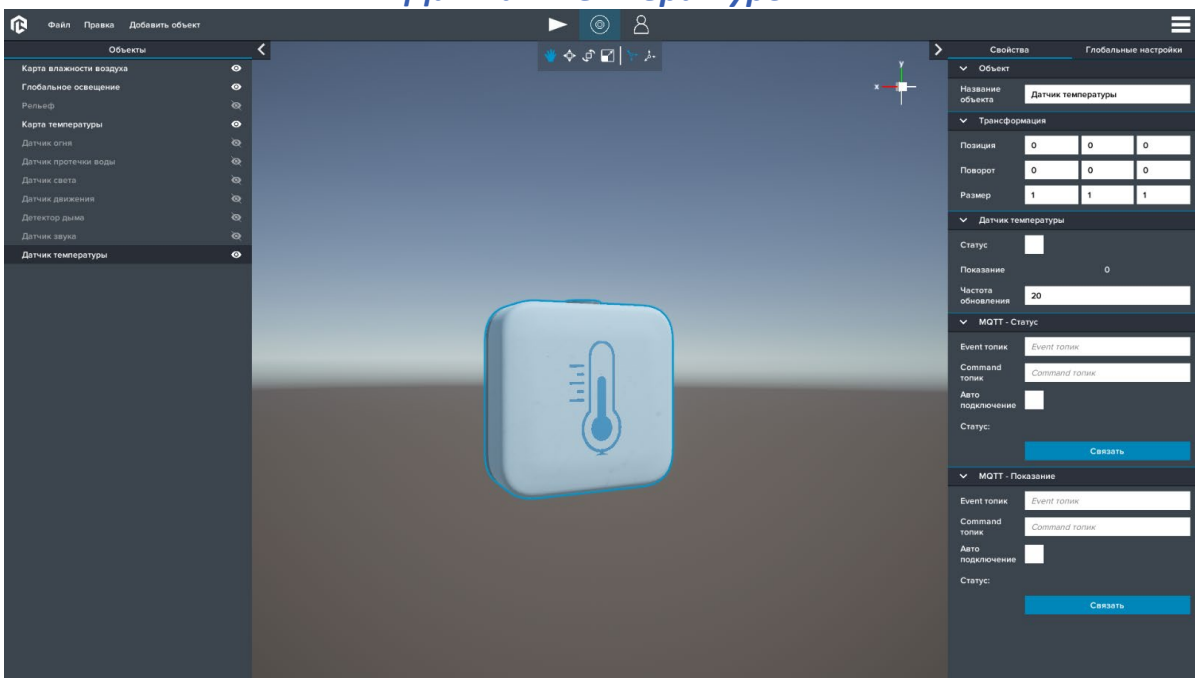
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик температуры



Датчик температуры

Компонент ***Датчик температуры*** – необходим для настройки параметров работы датчика. Датчик измеряет значение температуры в точки его местоположения.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	Значение температуры (°C)	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

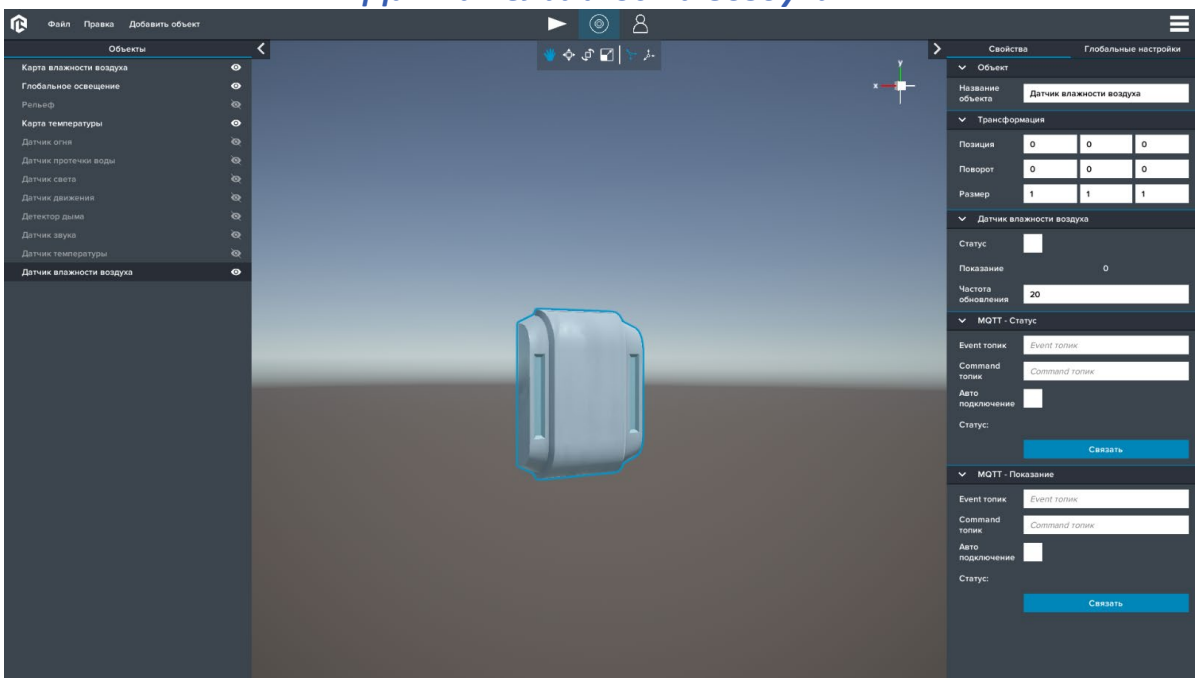
Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

Датчик влажности воздуха



Датчик влажности воздуха

Компонент ***Датчик влажности воздуха*** – необходим для настройки параметров работы датчика. Датчик измеряет значение влажности воздуха в точки его местоположения.

Параметр	Описание	Значение
Статус	Включение\выключение датчика	bool
Показание	Значение влажности воздуха (%)	float
Частота обновления	Частота обновления показания (количество раз в секунду)	float
Максимальная дистанция	Максимальна дистанция действия сенсора	float

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

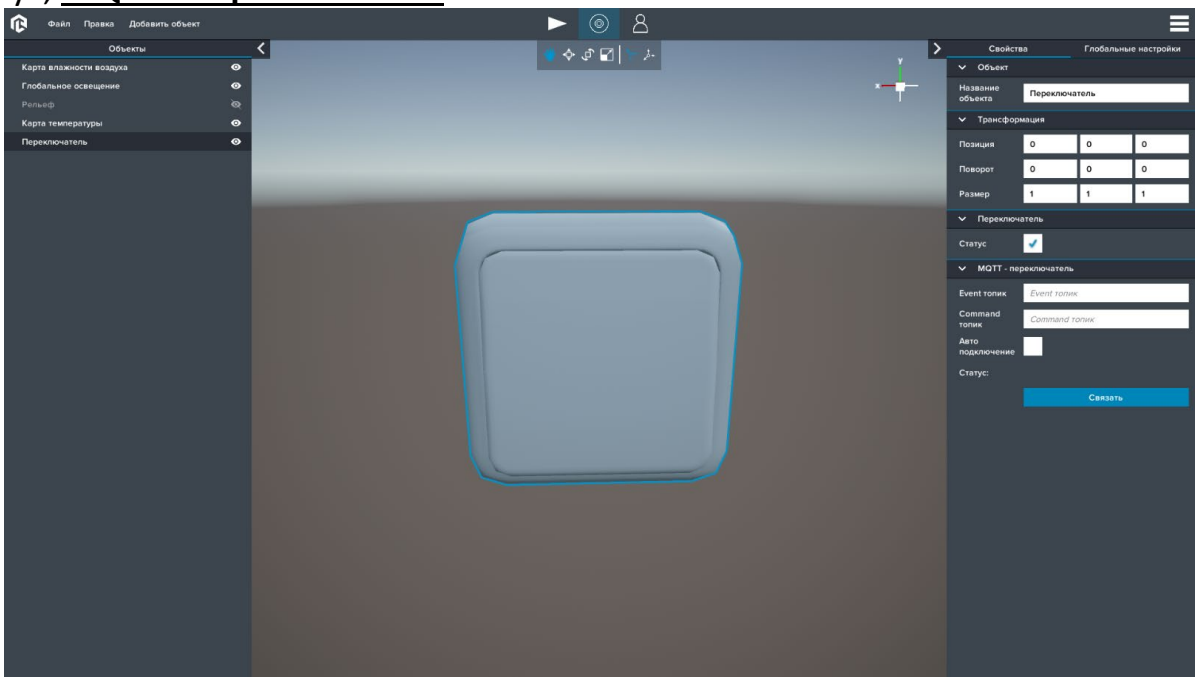
Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Показание*** – необходим для настройки управления параметром **Показание** при помощи MQTT сообщений.

Структура Command сообщения в формате JSON: {“value”: ***float***}

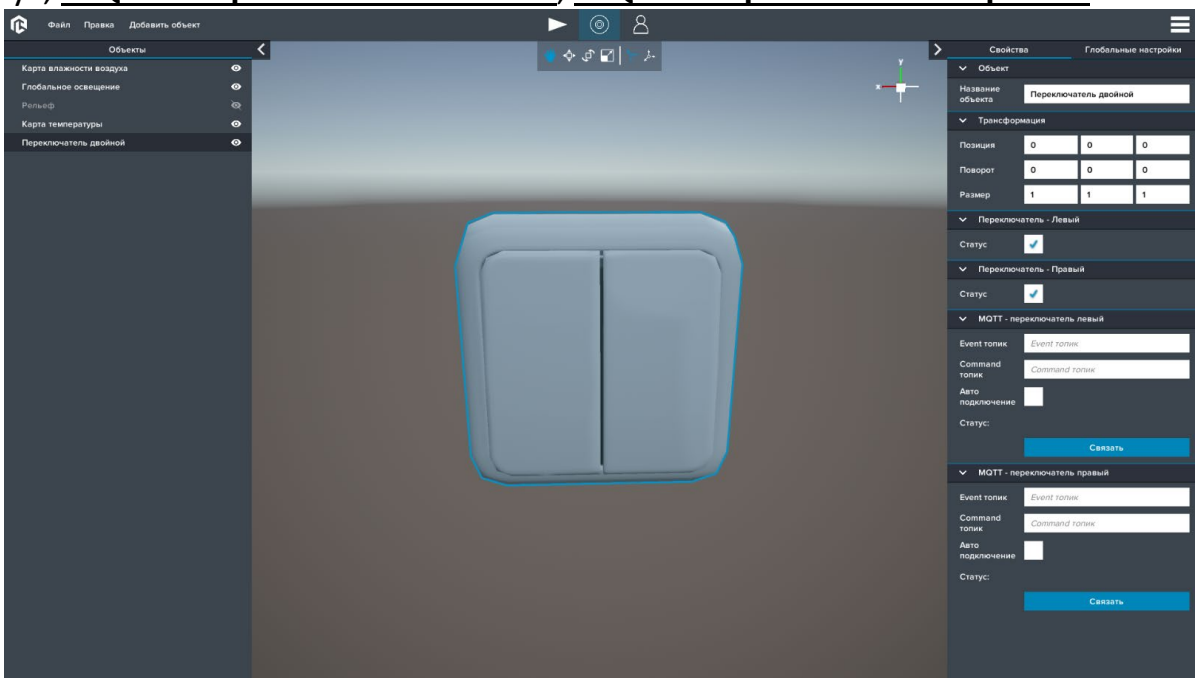
Переключатели

При добавлении *Переключателя* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Переключатель** – укажите статус, **MQTT – переключатель**.



Одноклавишный переключатель

При добавлении *Переключателя двойного* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Переключатель Левый** – укажите статус, **Переключатель Правый** – укажите статус, **MQTT – переключатель левый**, **MQTT – переключатель правый**.



Двухклавишный переключатель

Компонент ***Переключатель*** – необходим для переключения подачи питания на подключенные к нему устройства.

Параметр	Описание	Значение
Статус	Включение/выключение питания	bool

Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

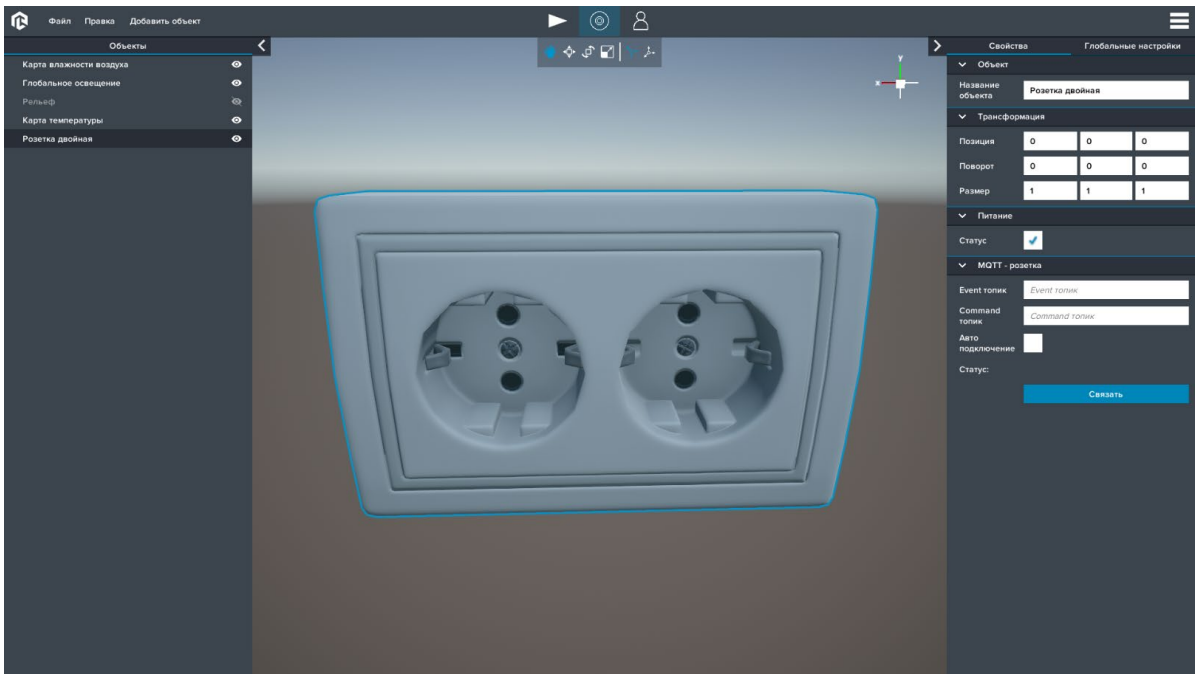
Структура Command сообщения в формате JSON: {“value”: ***bool***}

Розетки

При добавлении *Розетки* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Питание** – укажите статус, **MQTT – розетка**.

Розетка

При добавлении *Розетки двойной* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Питание** – укажите статус, **MQTT – розетка**.



Розетка двойная

Компонент ***Питание*** – необходим для подачи питания подключенным к розетке устройствам.

Параметр	Описание	Значение
Статус	Включение/выключение питания	bool

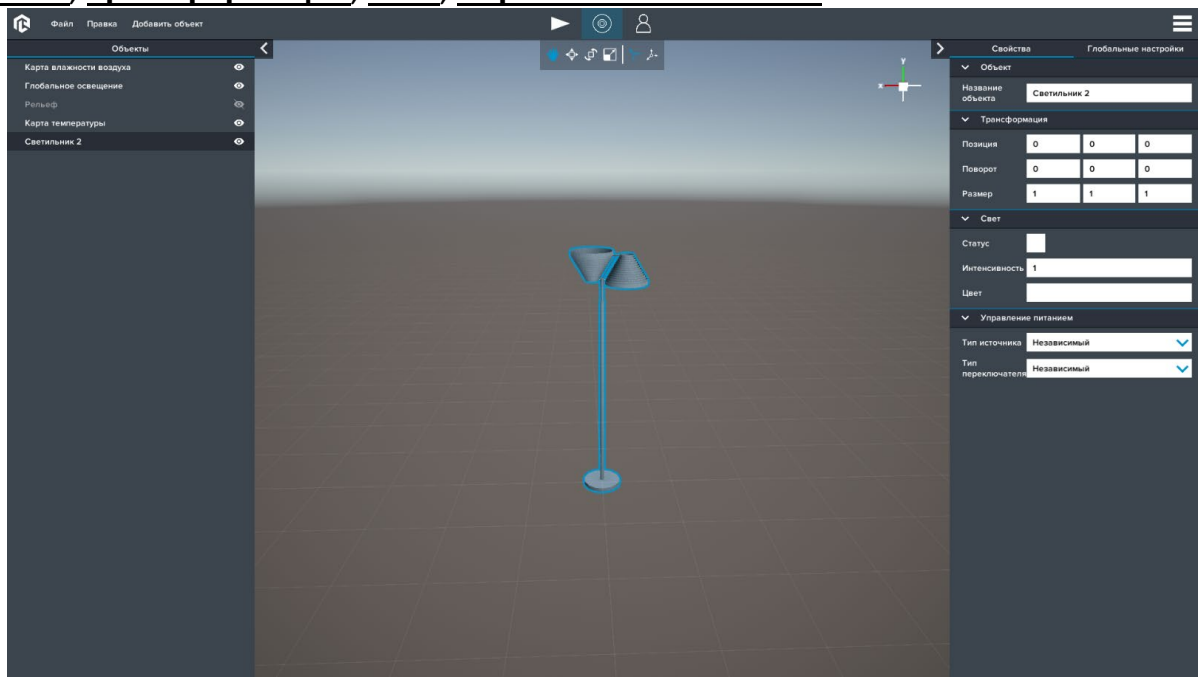
Компонент ***MQTT – Статус*** – необходим для настройки управления параметром **Статус** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Освещение

При добавлении *Светильника, Лампы потолочной или Настольной лампы* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Свет**, **Управление питанием**.



Светильник

Компонент ***Свет*** – необходим для управления параметрами источником света.

Параметр	Описание	Значение
Статус	Включение/выключение света	bool
Интенсивность	Интенсивность источника света	float
Цвет	Цвет источника света	color

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение
Тип источника	Выбор типа источника питания	Независимый/Источник питания
Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение

Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

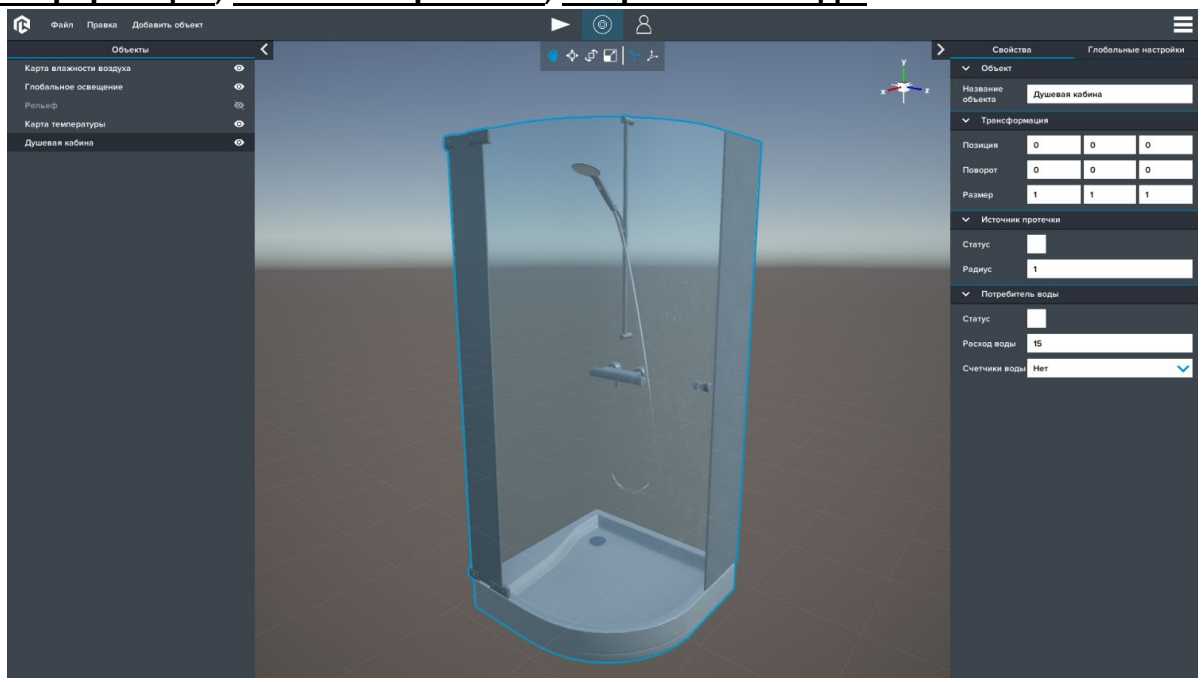
Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Водоснабжение

При добавлении *Ванны, Раковины, Душевой кабины или Туалета* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Источник протечки**, **Потребитель воды**.



Душевая кабина

Компонент ***Источник протечки*** – необходим для генерации условий возникновения протечки.

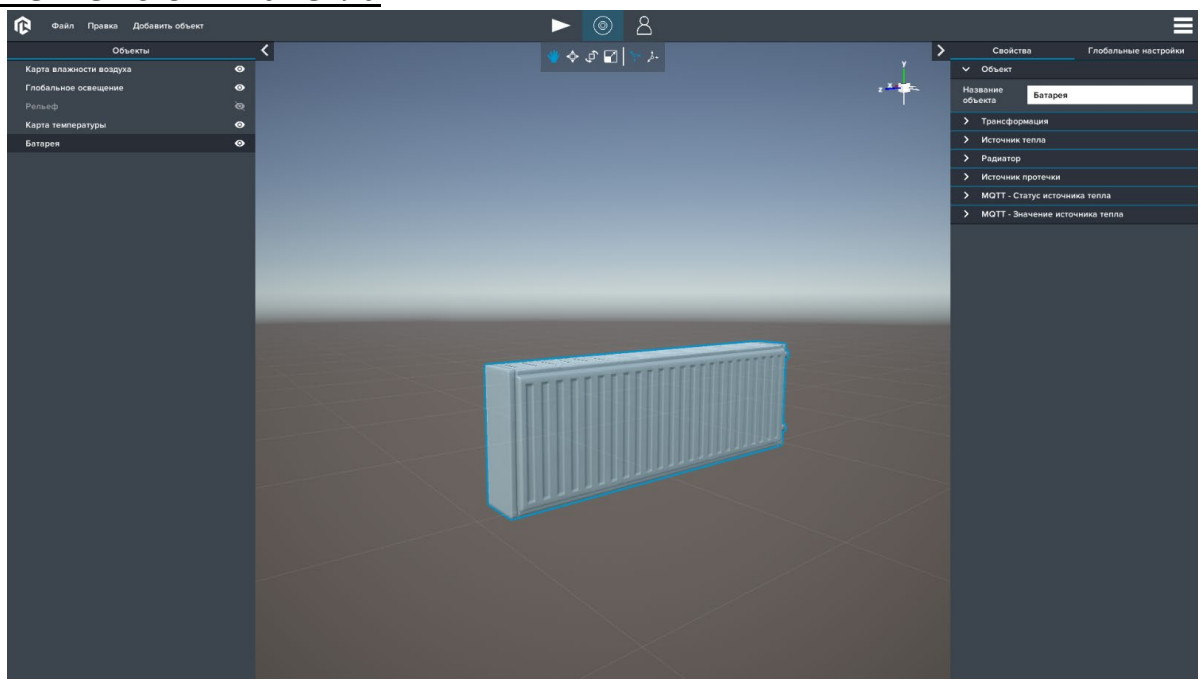
Параметр	Описание	Значение
Статус	Включение/выключение протечки	bool

Радиус	Радиус действия протечки, на который будут реагировать датчики протечки	float

Компонент ***Потребитель воды*** – необходим для потребления и учета расхода воды.

Параметр	Описание	Значение
Статус	Включение/выключение воды	bool
Расход воды	Значение расхода воды (л/мин)	float
Счётчик воды	Список доступных счётчиков воды	Счётчик воды

При добавлении *Батареи* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Источник тепла**, **Радиатор**, **Источник протечки**, **MQTT – Статус источника тепла**, **MQTT – Значение источника тепла**.



Батарея

Компонент ***Источник тепла*** – необходим для изменения температуры окружающей среды.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Радиус	Радиус действия	float

Значение	Значение температуры в центре (°C)	float

Компонент ***Радиатор*** – необходим для потребления и учета отопления.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Температура воды	Значение температуры в радиаторе (°C)	float
Расход воды	Значение расхода воды (л/мин)	float
Счётчик тепла	Список доступных счётчиков тепла	Счётчик тепла

Компонент ***Источник протечки*** – необходим для генерации условий возникновения протечки.

Параметр	Описание	Значение
Статус	Включение/выключение протечки	bool
Радиус	Радиус действия протечки, на который будут реагировать датчики протечки	float

Компонент ***MQTT – Статус источника тепла*** – необходим для настройки управления параметром **Статус** компонента **Источник тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

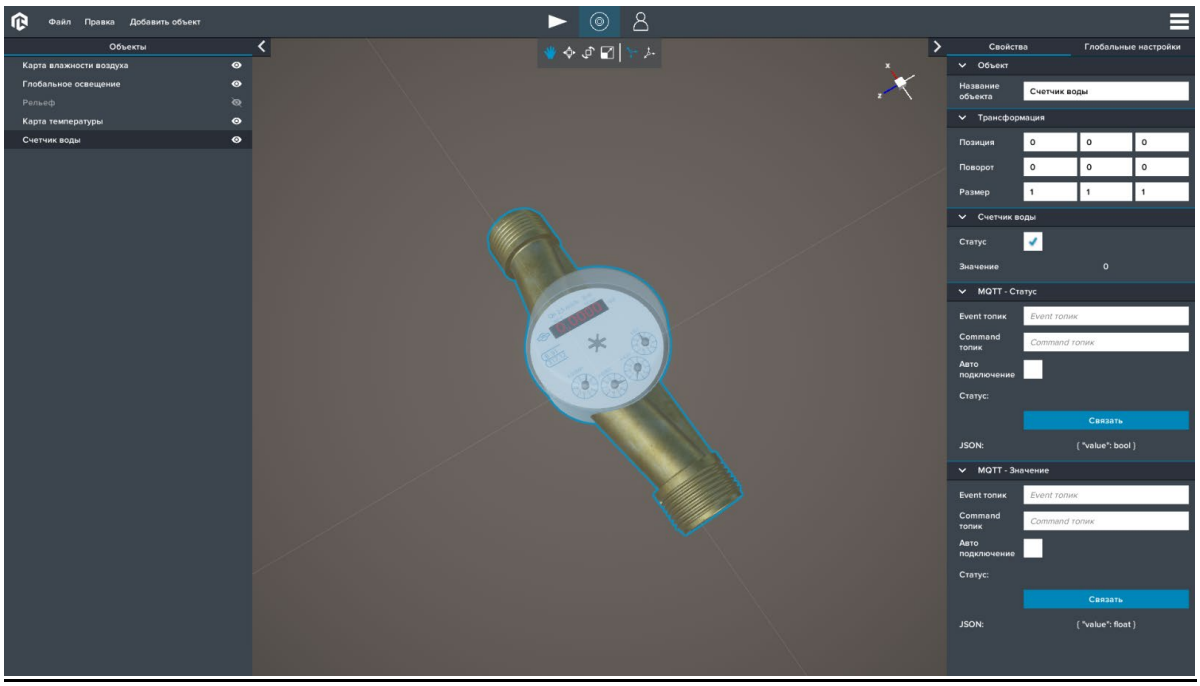
Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Значение источника тепла*** – необходим для настройки управления параметром **Значение** компонента **Источник тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***float***}

Структура Command сообщения в формате JSON: {“value”: ***float***}

При добавлении *Счетчика воды* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Счетчик воды**, **MQTT – Статус**, **MQTT – Значение**.



Счетчик воды

Компонент ***Счётчик воды*** – необходим для сбора показаний расхода воды.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Значение	Значение потребления (м³)	float

Компонент ***MQTT – Статус***– необходим для настройки управления параметром **Статус** компонента **Счётчик воды** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

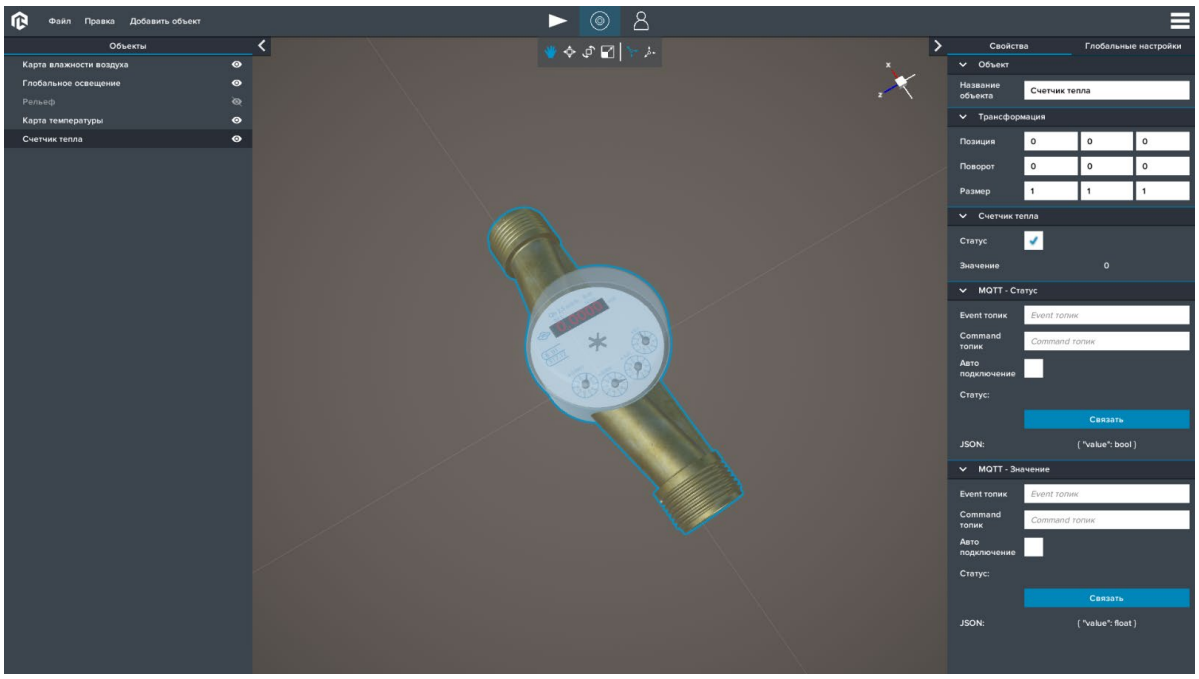
Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Значение***– необходим для настройки управления параметром **Значение** компонента **Счётчик воды** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***float***}

Структура Command сообщения в формате JSON: {“value”: ***float***}

При добавлении *Счетчика тепла* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Счетчик тепла**, **MQTT – Статус**, **MQTT – Значение**.



Счетчик тепла

Компонент ***Счётчик тепла*** – необходим для сбора показаний расхода радиаторами.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Значение	Значение потребления (МДж)	float

Компонент ***MQTT – Статус***– необходим для настройки управления параметром **Статус** компонента **Счётчик тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

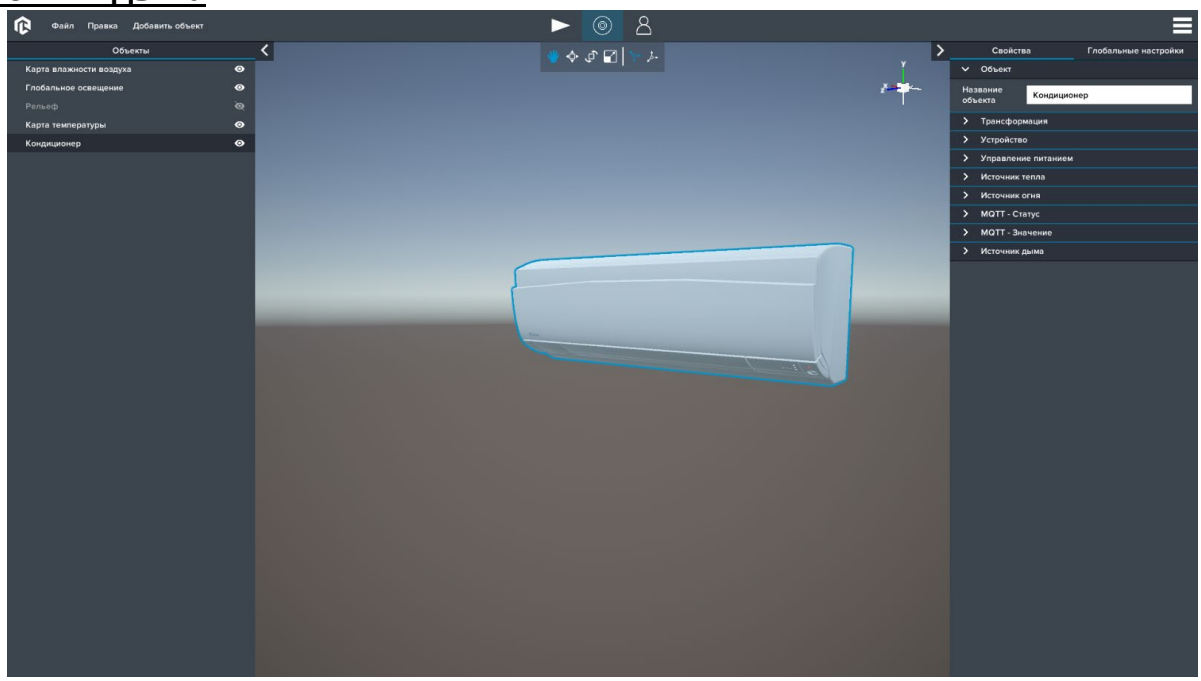
Компонент ***MQTT – Значение***– необходим для настройки управления параметром **Значение** компонента **Счётчик тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***float***}

Структура Command сообщения в формате JSON: {“value”: ***float***}

Бытовая техника

При добавлении *Кондиционера* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Устройство**, **Управление питанием**, **Источник тепла**, **Источник огня**, **MQTT – Статус**, **MQTT – Значение**, **Источник дыма**.



Кондиционер

Компонент ***Устройство*** – необходим для управления статусом работы устройства.

Параметр	Описание	Значение
Статус	Включение/выключение устройства	bool

Компонент ***Источник тепла*** – необходим для изменения температуры окружающей среды.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Радиус	Радиус действия	float
Значение	Значение температуры в центре (°C)	float

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение

Тип источника	Выбор типа источника питания	Независимый/Источник питания
Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение
Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***Источник огня*** – необходим для генерации условий возникновения огня.

Параметр	Описание	Значение
Статус	Включение/выключение огня	bool
Радиус	Радиус действия огня, на который будут реагировать датчики огня	float

Компонент ***Источник дыма*** – необходим для генерации условий возникновения дыма.

Параметр	Описание	Значение
Статус	Включение/выключение дыма	bool
Радиус	Радиус действия дыма, на который будут	float

	реагировать датчики дыма	

Компонент ***MQTT – Статус источника тепла*** – необходим для настройки управления параметром **Статус** компонента **Источник тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

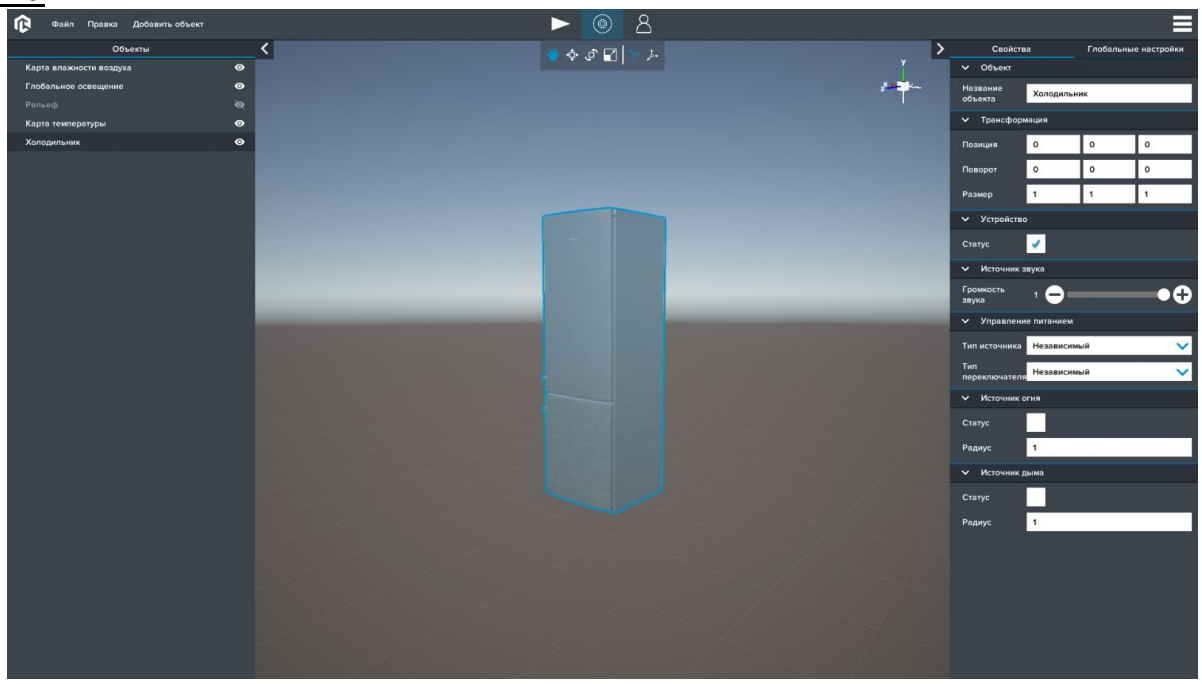
Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***MQTT – Значение источника тепла*** – необходим для настройки управления параметром **Значение** компонента **Источник тепла** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***float***}

Структура Command сообщения в формате JSON: {“value”: ***float***}

При добавлении *Холодильника, телевизора* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Источник звука**, **Устройство**, **Управление питанием**, **Источник огня**, **Источник дыма**.



Холодильник

Компонент ***Устройство*** – необходим для управления статусом работы устройства.

Параметр	Описание	Значение
Статус	Включение/выключение устройства	bool

Компонент ***Источник звука*** – необходим для управления звуком.

Параметр	Описание	Значение
Громкость звука	Уровень громкости звука	float

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение
Тип источника	Выбор типа источника питания	Независимый/Источник питания
Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение
Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***Источник огня*** – необходим для генерации условий возникновения огня.

Параметр	Описание	Значение
Статус	Включение/выключение огня	bool
Радиус	Радиус действия огня, на который будут реагировать датчики огня	float

Компонент ***Источник дыма*** – необходим для генерации условий возникновения дыма.

Параметр	Описание	Значение
Статус	Включение/выключение дыма	bool
Радиус	Радиус действия дыма, на который будут реагировать датчики дыма	float

При добавлении *Обогревателя* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Устройство**, **Управление питанием**, **Источник тепла**, **Источник огня**, **Источник дыма**.

Обогреватель

Компонент ***Устройство*** – необходим для управления статусом работы устройства.

Параметр	Описание	Значение
Статус	Включение/выключение устройства	bool

Компонент ***Источник тепла*** – необходим для изменения температуры окружающей среды.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Радиус	Радиус действия	float

Значение	Значение температуры в центре (°C)	float

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение
Тип источника	Выбор типа источника питания	Независимый/Источник питания
Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение
Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

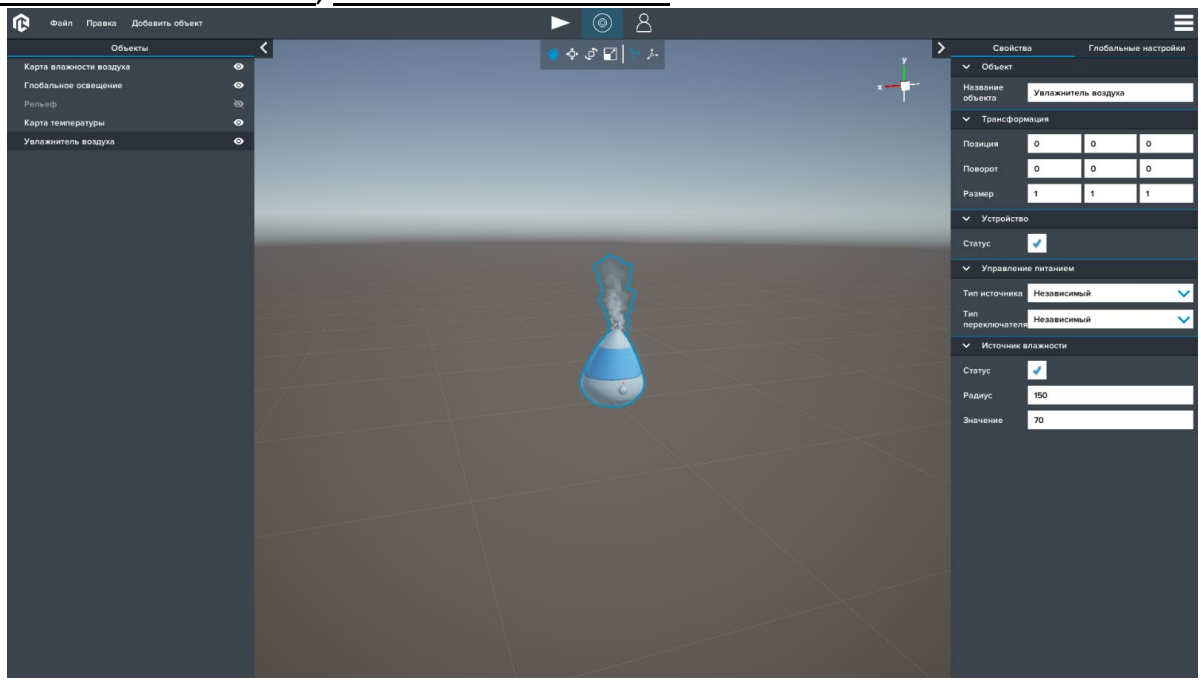
Компонент ***Источник огня*** – необходим для генерации условий возникновения огня.

Параметр	Описание	Значение
Статус	Включение/выключение огня	bool
Радиус	Радиус действия огня, на который будут реагировать датчики огня	float

Компонент ***Источник дыма*** – необходим для генерации условий возникновения дыма.

Параметр	Описание	Значение
Статус	Включение/выключение дыма	bool
Радиус	Радиус действия дыма, на который будут реагировать датчики дыма	float

При добавлении *Увлажнителя воздуха* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Устройство**, **Управление питанием**, **Источник влажности**.



Увлажнитель воздуха

Компонент ***Устройство*** – необходим для управления статусом работы устройства.

Параметр	Описание	Значение
Статус	Включение/выключение устройства	bool

Компонент ***Источник влажности*** – необходим для изменения влажности воздуха окружающей среды.

Параметр	Описание	Значение
Статус	Включение/выключение	bool
Радиус	Радиус действия	float

Значение	Значение влажности воздуха в центре (%)	float

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение
Тип источника	Выбор типа источника питания	Независимый/Источник питания
Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение
Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

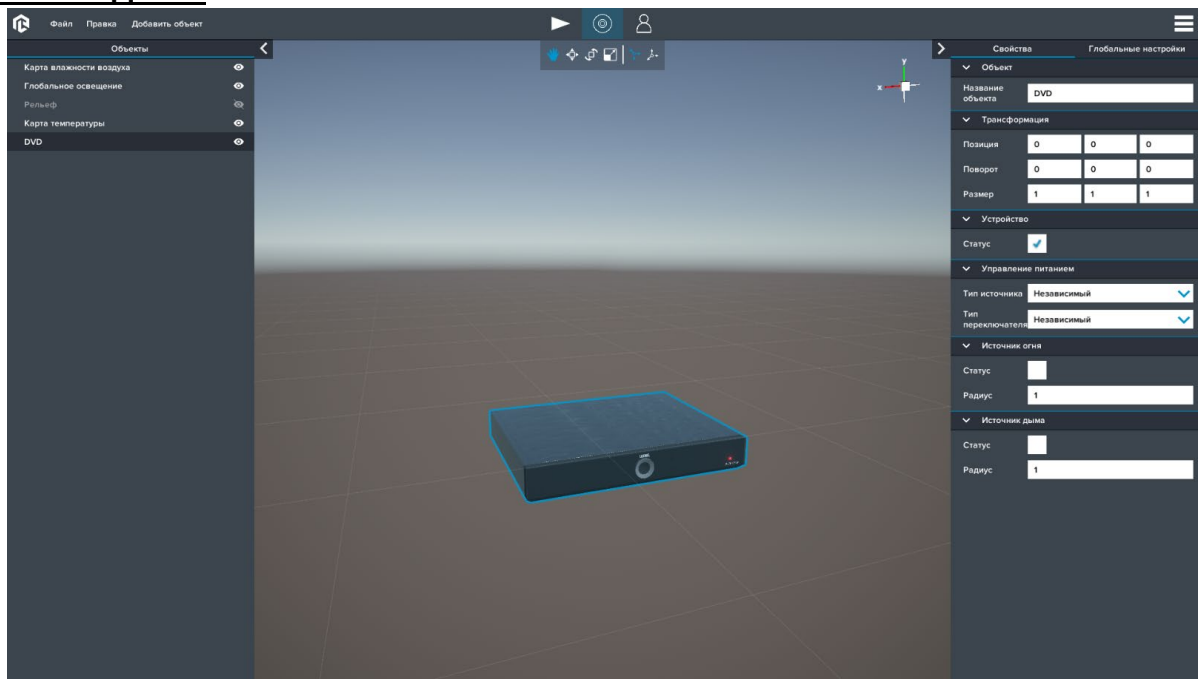
Компонент ***Источник огня*** – необходим для генерации условий возникновения огня.

Параметр	Описание	Значение
Статус	Включение/выключение огня	bool
Радиус	Радиус действия огня, на который будут реагировать датчики огня	float

Компонент ***Источник дыма*** – необходим для генерации условий возникновения дыма.

Параметр	Описание	Значение
Статус	Включение/выключение дыма	bool
Радиус	Радиус действия дыма, на который будут реагировать датчики дыма	float

При добавлении *Кофемашины, Плиты, DVD, Микроволновой печи, Печи, Стиральной машины, Чайника, Ноутбука или Компьютера* укажите необходимы свойства, заполняя следующие компоненты: **Объект**, **Трансформация**, **Устройство**, **Управление питанием**, **Источник огня**, **Источник дыма**.



DVD

Компонент ***Устройство*** – необходим для управления статусом работы устройства.

Параметр	Описание	Значение
Статус	Включение/выключение устройства	bool

Компонент ***Управление питанием*** – необходим для выбора типа источника питания и типа переключателя.

Параметр	Описание	Значение
Тип источника	Выбор типа источника питания	Независимый/Источник питания

Тип переключателя	Выбор типа переключателя	Независимый/Переключатель/MQTT

Компонент ***Источник питания*** – необходим для выбора источника питания.

Параметр	Описание	Значение
Источник питания	Список доступных источников питания	Источник питания

Компонент ***Источник переключателя*** – необходим для выбора переключателя.

Параметр	Описание	Значение
Источник переключателя	Список доступных переключателей	Переключатель

Компонент ***MQTT – Переключатель*** – необходим для настройки управления параметром **Статус** компонента **Свет** при помощи MQTT сообщений.

Структура Event сообщения в формате JSON: {“value”: ***bool***}

Структура Command сообщения в формате JSON: {“value”: ***bool***}

Компонент ***Источник огня*** – необходим для генерации условий возникновения огня.

Параметр	Описание	Значение
Статус	Включение/выключение огня	bool
Радиус	Радиус действия огня, на который будут реагировать датчики огня	float

Компонент ***Источник дыма*** – необходим для генерации условий возникновения дыма.

Параметр	Описание	Значение
Статус	Включение/выключение дыма	bool
Радиус	Радиус действия дыма, на который будут реагировать датчики дыма	float

ПРОЕКТЫ

Проект квартиры с автоматизацией

Проект квартиры

Прихожая

В прихожей находятся следующие устройства:

1. **<u>Лампа потолочная 1 прихожая</u>**

   MQTT – статус:

   Event топик – mqtt:topic:main:lamp_02:status/out

   Command топик – mqtt:topic:main:lamp_02:status/in

2. **Лампа потолочная прихожая**
   MQTT – статус:
   Event топик – mqtt:topic:main:lamp_01:status/out
   Command топик – mqtt:topic:main:lamp_01:status/in
3. **Датчик света прихожая**
   MQTT – статус:
   Event топик – mqtt:topic:main:light_sensor_2:status/out
   Command топик – mqtt:topic:main:light_sensor_2:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:light_sensor_2:value/out
   Command топик – mqtt:topic:main:light_sensor_2:value/in
4. **Детектор дыма прихожая**
   MQTT – статус:
   Event топик – mqtt:topic:main:smoke_detector_3:status/out
   Command топик – mqtt:topic:main:smoke_detector_3:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:smoke_detector_3:value/out
   Command топик – mqtt:topic:main:smoke_detector_3:value/in
5. **Датчик движения прихожая**
   MQTT – статус:
   Event топик – mqtt:topic:main:motion_sensor_2:status/out
   Command топик – mqtt:topic:main:motion_sensor_2:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:motion_sensor_2:value/out
   Command топик – mqtt:topic:main:motion_sensor_2:value/in
6. **Переключатель прихожая**
   MQTT – переключатель:
   Event топик – mqtt:topic:main:switch_01:status/out
   Command топик – mqtt:topic:main:switch_01:status/in
7. **Датчик звука прихожая**
   MQTT – статус:
   Event топик – mqtt:topic:main:sound_sensor_2:status/out
   Command топик – mqtt:topic:main:sound_sensor_2:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:sound_sensor_2:value/out
   Command топик – mqtt:topic:main:sound_sensor_2:value/in
8. **Переключатель ванна**
   MQTT – переключатель:
   Event топик – mqtt:topic:main:switch_02:status/out
   Command топик – mqtt:topic:main:switch_02:status/in
9. **Розетка двойная ванна**

MQTT – розетка:
Event топик – mqtt:topic:main:power_socket_01:status/out
Command топик – mqtt:topic:main:power_socket_01:status/in

10. **Датчик влажности воздуха прихожая**

MQTT – статус:
Event топик – mqtt:topic:main:wet_sensor:status/out
Command топик – mqtt:topic:main:wet_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:main:wet_sensor:value/out
Command топик – mqtt:topic:main:wet_sensor:value/in

Ванная комната

В ванной комнате находятся следующие устройства:

1. **Лампа потолочная ванна**

MQTT – статус:
Event топик – mqtt:topic:main:lamp_03:status/out
Command топик – mqtt:topic:main:lamp_03:status/in

2. **Лампа потолочная 1 ванна**

MQTT – статус:
Event топик – mqtt:topic:main:lamp_04:status/out
Command топик – mqtt:topic:main:lamp_04:status/in

3. **Детектор дыма ванная**

MQTT – статус:
Event топик – mqtt:topic:main:smoke_detector_4:status/out
Command топик – mqtt:topic:main:smoke_detector_4:status/in
MQTT – показание:
Event топик – mqtt:topic:main:smoke_detector_4:value/out

Command топик – mqtt:topic:main:smoke_detector_4:value/in

4. **Датчик влажности воздуха ванная**

MQTT – статус:

Event топик – mqtt:topic:main:wet_sensor_1:status/out

Command топик – mqtt:topic:main:wet_sensor_1:status/in

MQTT – показание:

Event топик – mqtt:topic:main:wet_sensor_1:value/out

Command топик – mqtt:topic:main:wet_sensor_1:value/in

5. **Раковина**
6. **Душевая кабина**
7. **Туалет**
8. **Стиральная машина**

MQTT – статус:

Event топик – mqtt:topic:main:device_01:status/out

Command топик – mqtt:topic:main:device_01:status/in

9. **Датчик протечки воды ванная**

MQTT – статус:

Event топик – mqtt:topic:main:leak_sensor:status/out

Command топик – mqtt:topic:main:leak_sensor:status/in

MQTT – показание:

Event топик – mqtt:topic:main:leak_sensor:value/out

Command топик – mqtt:topic:main:leak_sensor:value/in

Спальня

В спальне находятся следующие устройства:

1. **Лампа потолочная вторая комната**

MQTT – статус:

Event топик – mqtt:topic:main:lamp_08:status/out
Command топик – mqtt:topic:main:lamp_08:status/in

2. **Лампа потолочная 2 вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:lamp_09:status/out
   Command топик – mqtt:topic:main:lamp_09:status/in
3. **Детектор дыма вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:smoke_detector_2:status/out
   Command топик – mqtt:topic:main:smoke_detector_2:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:smoke_detector_2:value/out
   Command топик – mqtt:topic:main:smoke_detector_2:value/in
4. **Датчик света вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:light_sensor_1:status/out
   Command топик – mqtt:topic:main:light_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:light_sensor_1:value/out
   Command топик – mqtt:topic:main:light_sensor_1:value/in
5. **Датчик движения вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:motion_sensor_1:status/out
   Command топик – mqtt:topic:main:motion_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:motion_sensor_1:value/out
   Command топик – mqtt:topic:main:motion_sensor_1:value/in
6. **Датчик температуры вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:temperature_sensor_1:status/out
   Command топик – mqtt:topic:main:temperature_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:temperature_sensor_1:value/out
   Command топик – mqtt:topic:main:temperature_sensor_1:value/in
7. **Датчик звука вторая комната**
   MQTT – статус:
   Event топик – mqtt:topic:main:sound_sensor_1:status/out
   Command топик – mqtt:topic:main:sound_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:sound_sensor_1:value/out
   Command топик – mqtt:topic:main:sound_sensor_1:value/in

8. **Розетка двойная 1 вторая комната**

   MQTT – розетка:

   Event топик – mqtt:topic:main:power_socket_09:status/out

   Command топик – mqtt:topic:main:power_socket_09:status/in

9. **Переключатель вторая комната**

   MQTT – переключатель:

   Event топик – mqtt:topic:main:switch_04:status/out

   Command топик – mqtt:topic:main:switch_04:status/in

10. **Компьютер**

    MQTT – статус:

    Event топик – mqtt:topic:main:device_13:status/out

    Command топик – mqtt:topic:main:device_13:status/in

11. **Лампа настольная вторая комната**

    MQTT – статус:

    Event топик – mqtt:topic:main:lamp_10:status/out

    Command топик – mqtt:topic:main:lamp_10:status/in

12. **Розетка двойная вторая комната**

    MQTT – статус:

    Event топик – mqtt:topic:main:power_socket_08:status/out

    Command топик – mqtt:topic:main:power_socket_08:status/in

13. **Телевизор вторая комната**

    MQTT – статус:

    Event топик – mqtt:topic:main:device_12:status/out

    Command топик – mqtt:topic:main:device_12:status/in

14. **Батарея**

15. **Обогреватель**

    MQTT – статус:

    Event топик – mqtt:topic:main:device_14:status/out

    Command топик – mqtt:topic:main:device_14:status/in

Студия

В студии находятся следующие устройства:

1. **Детектор дыма 1 первая комната**

   MQTT – статус:
   Event топик – mqtt:topic:main:smoke_detector_1:status/out
   Command топик – mqtt:topic:main:smoke_detector_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:smoke_detector_1:value/out
   Command топик – mqtt:topic:main:smoke_detector_1:value/in

2. **Лампа потолочная 2 первая комната**

   MQTT – статус:
   Event топик – mqtt:topic:main:lamp_07:status/out
   Command топик – mqtt:topic:main:lamp_07:status/in

3. **Лампа потолочная 1 первая комната**

   MQTT – статус:
   Event топик – mqtt:topic:main:lamp_06:status/out
   Command топик – mqtt:topic:main:lamp_06:status/in

4. **Лампа потолочная первая комната**

   MQTT – статус:
   Event топик – mqtt:topic:main:lamp_05:status/out
   Command топик – mqtt:topic:main:lamp_05:status/in

5. **Детектор дыма первая комната**

   MQTT – статус:
   Event топик – mqtt:topic:main:smoke_detector:status/out
   Command топик – mqtt:topic:main:smoke_detector:status/in
   MQTT – показание:
   Event топик – mqtt:topic:main:smoke_detector:value/out

Command топик – mqtt:topic:main:smoke_detector:value/in

6. **Датчик огня первая комната**

MQTT – статус:
Event топик – mqtt:topic:main:fire_sensor:status/out
Command топик – mqtt:topic:main:fire_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:main:fire_sensor:value/out
Command топик – mqtt:topic:main:fire_sensor:value/in

7. **Датчик движения первая комната**

MQTT – статус:
Event топик – mqtt:topic:main:motion_sensor:status/out
Command топик – mqtt:topic:main:motion_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:main:motion_sensor:value/out
Command топик – mqtt:topic:main:motion_sensor:value/in

8. **Переключатель двойной первая комната**

MQTT – переключатель левый:
Event топик – mqtt:topic:main:switch_03:status_left/out
Command топик – mqtt:topic:main:switch_03:status_left/in
MQTT – переключатель правый:
Event топик – mqtt:topic:main:switch_03:status_right/out
Command топик – mqtt:topic:main:switch_03:status_right/in

9. **Датчик температуры первая комната**

MQTT – статус:
Event топик – mqtt:topic:main:temperature_sensor:status/out
Command топик – mqtt:topic:main:temperature_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:main:temperature_sensor:value/out
Command топик – mqtt:topic:main:temperature_sensor:value/in

10. **Датчик звука первая комната**

MQTT – статус:
Event топик – mqtt:topic:main:sound_sensor:status/out
Command топик – mqtt:topic:main:sound_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:main:sound_sensor:value/out
Command топик – mqtt:topic:main:sound_sensor:value/in

11. **Розетка двойная 4 первая комната**

MQTT – розетка:
Event топик – mqtt:topic:main:power_socket_07:status/out
Command топик – mqtt:topic:main:power_socket_07:status/in

12. **Кофемашина**

MQTT – статус:
Event топик – mqtt:topic:main:device_05:status/out
Command топик – mqtt:topic:main:device_05:status/in

13. **Чайник**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_06:status/out
    Command топик – mqtt:topic:main:device_06:status/in
14. **Плита**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_07:status/out
    Command топик – mqtt:topic:main:device_07:status/in
15. **Печь**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_08:status/out
    Command топик – mqtt:topic:main:device_08:status/in
16. **Микроволновая печь**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_09:status/out
    Command топик – mqtt:topic:main:device_09:status/in
17. **Розетка двойная первая комната**
    MQTT – розетка:
    Event топик – mqtt:topic:main:power_socket_03:status/out
    Command топик – mqtt:topic:main:power_socket_03:status/in
18. **Холодильник**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_10:status/out
    Command топик – mqtt:topic:main:device_10:status/in
19. **Кондиционер**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_11:status/out
    Command топик – mqtt:topic:main:device_11:status/in
20. **Батарея**
21. **Датчик света первая комната**
    MQTT – статус:
    Event топик – mqtt:topic:main:light_sensor:status/out
    Command топик – mqtt:topic:main:light_sensor:status/in
    MQTT – показание:
    Event топик – mqtt:topic:main:light_sensor:value/out
    Command топик – mqtt:topic:main:light_sensor:value/in
22. **Розетка двойная 1 первая комната**
    MQTT – розетка:

Event топик – mqtt:topic:main:power_socket_04:status/out
Command топик – mqtt:topic:main:power_socket_04:status/in

23. **Розетка двойная 2 первая комната**
    MQTT – розетка:
    Event топик – mqtt:topic:main:power_socket_05:status/out
    Command топик – mqtt:topic:main:power_socket_05:status/in
24. **Лампа светильник первая комната**
    MQTT – статус:
    Event топик – mqtt:topic:main:lamp_11:status/out
    Command топик – mqtt:topic:main:lamp_11:status/in
25. **Увлажнитель воздуха первая комната**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_04:status/out
    Command топик – mqtt:topic:main:device_04:status/in
26. **Телевизор первая комната**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_03:status/out
    Command топик – mqtt:topic:main:device_03:status/in
27. **DVD**
    MQTT – статус:
    Event топик – mqtt:topic:main:device_02:status/out
    Command топик – mqtt:topic:main:device_02:status/in
28. **Розетка двойная 3 первая комната**
    MQTT – розетка:
    Event топик – mqtt:topic:main:power_socket_06:status/out
    Command топик – mqtt:topic:main:power_socket_06:status/in

Проект макета дома с автоматизацией

Проект макета дома

Вид сверху

Гостиная

В гостиной находятся следующие устройства:

1. **Переключатель свет зал**

   MQTT – переключатель:

   Event топик – mqtt:topic:maket:hall_light_switch/out

   Command топик – mqtt:topic:maket:hall_light_switch/in

2. **Датчик движения 1**

   MQTT – статус:

   Event топик – mqtt:topic:maket:motion_sensor:status/out

   Command топик – mqtt:topic:maket:motion_sensor:status/in

   MQTT – показание:

   Event топик – mqtt:topic:maket:motion_sensor:value/out

3. **Датчик света 1**

   MQTT – статус:

   Event топик – mqtt:topic:maket:light_sensor:status/out

   Command топик – mqtt:topic:maket:light_sensor:status/in

   MQTT – показание:

   Event топик – mqtt:topic:maket:light_sensor:value/out

   Command топик – mqtt:topic:maket:light_sensor:value/in

4. **Детектор дыма 1**

   MQTT – статус:

   Event топик – mqtt:topic:maket:smoke_detector:status/out

   Command топик – mqtt:topic:maket:smoke_detector:status/in

   MQTT – показание:

   Event топик – mqtt:topic:maket:smoke_detector:value/out

   Command топик – mqtt:topic:maket:smoke_detector:value/in

5. **Розетка двойная3**

6. **Компьютер**
7. **Настольная лампа**
8. **Батарея**
9. **Розетка двойная4**
10. **Светильник 2**
11. **Увлажнитель воздуха**
12. **Датчик температуры 3**

    MQTT – статус:
    Event топик – mqtt:topic:maket:temperature_sensor_2:status/out
    Command топик – mqtt:topic:maket:temperature_sensor_2:status/in
    MQTT – показание:
    Event топик – mqtt:topic:maket:temperature_sensor_2:value/out
    Command топик – mqtt:topic:maket:temperature_sensor_2:value/in
13. **Детектор дыма 2**

    MQTT – статус:
    Event топик – mqtt:topic:maket:smoke_detector_1:status/out
    Command топик – mqtt:topic:maket:smoke_detector_1:status/in
    MQTT – показание:
    Event топик – mqtt:topic:maket:smoke_detector_1:value/out
    Command топик – mqtt:topic:maket:smoke_detector_1:value/in
14. **Датчик огня 2**

    MQTT – статус:
    Event топик – mqtt:topic:maket:fire_sensor_1:status/out
    Command топик – mqtt:topic:maket:fire_sensor_1:status/in
    MQTT – показание:
    Event топик – mqtt:topic:maket:fire_sensor_1:value/out
    Command топик – mqtt:topic:maket:fire_sensor_1:value/in
15. **Ноутбук**
16. **Переключатель свет коридор**

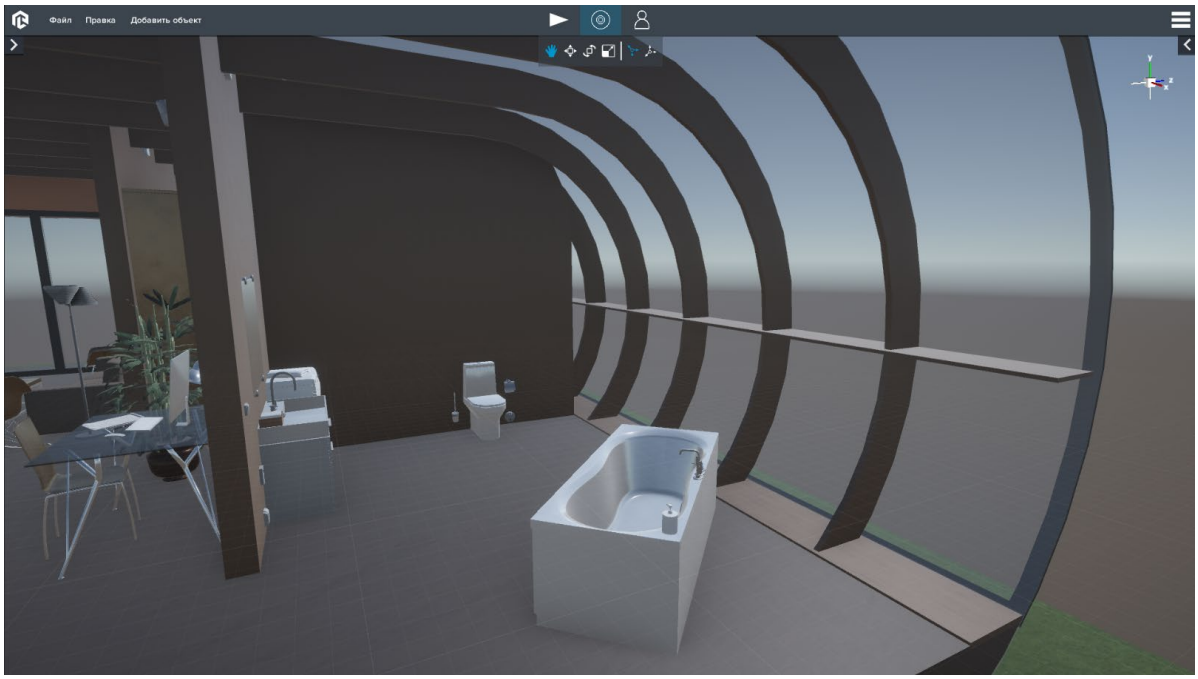
    MQTT – переключатель:
    Event топик – mqtt:topic:maket:corridor_light_switch/out
    Command топик – mqtt:topic:maket:corridor_light_switch/in
17. **Телевизор 2**
18. **DVD**
19. **Датчик света 2**

    MQTT – статус:
    Event топик – mqtt:topic:maket:light_sensor_1:status/out
    Command топик – mqtt:topic:maket:light_sensor_1:status/in
    MQTT – показание:
    Event топик – mqtt:topic:maket:light_sensor_1:value/out
    Command топик – mqtt:topic:maket:light_sensor_1:value/in

20. **Лампа потолочная 4**
21. **Лампа потолочная 4**
22. **Лампа потолочная 4**
23. **Лампа потолочная 4**
24. **Лампа потолочная 4**
25. **Лампа потолочная 4**
26. **Датчик звука**

MQTT – статус:
Event топик – mqtt:topic:maket:sound_sensor:status/out
Command топик – mqtt:topic:maket:sound_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:maket:sound_sensor:value/out
Command топик – mqtt:topic:maket:sound_sensor:value/in

Ванная комната



В ванной комнате находятся следующие устройства:

1. **Переключатель свет ванная**

MQTT – переключатель:
Event топик – mqtt:topic:maket:bathroom_light_switch/out
Command топик – mqtt:topic:maket:bathroom_light_switch/in

1. **Лампа потолочная 4**
2. **Лампа потолочная 4**
3. **Датчик температуры 4**

MQTT – статус:
Event топик – mqtt:topic:maket:temperature_sensor_3:status/out
Command топик – mqtt:topic:maket:temperature_sensor_3:status/in
MQTT – показание:

Event топик – mqtt:topic:maket:temperature_sensor_3:value/out
Command топик – mqtt:topic:maket:temperature_sensor_3:value/in

4. **Датчик влажности воздуха**

MQTT – статус:
Event топик – mqtt:topic:maket:wet_sensor:status/out
Command топик – mqtt:topic:maket:wet_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:maket:wet_sensor:value/out
Command топик – mqtt:topic:maket:wet_sensor:value/in

5. **Датчик протечки воды**

MQTT – статус:
Event топик – mqtt:topic:maket:leak_sensor:status/out
Command топик – mqtt:topic:maket:leak_sensor:status/in
MQTT – показание:
Event топик – mqtt:topic:maket:leak_sensor:value/out
Command топик – mqtt:topic:maket:leak_sensor:value/in

6. **Розетка двойная1**
7. **Раковина**
8. **Стиральная машина**
9. **Туалет**
10. **Ванна**

Спальня

В спальне находятся следующие устройства:

1. **Переключатель свет спальня**

MQTT – переключатель:
Event топик – mqtt:topic:maket:bedroom_light_switch/out

Command топик – mqtt:topic:maket:bedroom_light_switch/in

1. **Лампа потолочная 4**
2. **Лампа потолочная 4**
3. **Лампа потолочная 4**
4. **Лампа потолочная 4**
5. **Розетка двойная6**
6. **Обогреватель**
7. **Детектор дыма 4**

   MQTT – статус:
   Event топик – mqtt:topic:maket:smoke_detector_3:status/out
   Command топик – mqtt:topic:maket:smoke_detector_3:status/in
   MQTT – показание:
   Event топик – mqtt:topic:maket:smoke_detector_3:value/out
   Command топик – mqtt:topic:maket:smoke_detector_3:value/in
8. **Датчик температуры 2**

   MQTT – статус:
   Event топик – mqtt:topic:maket:temperature_sensor_1:status/out
   Command топик – mqtt:topic:maket:temperature_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:maket:temperature_sensor_1:value/out
   Command топик – mqtt:topic:maket:temperature_sensor_1:value/in
9. **Датчик движения 2**

   MQTT – статус:
   Event топик – mqtt:topic:maket:motion_sensor_1:status/out
   Command топик – mqtt:topic:maket:motion_sensor_1:status/in
   MQTT – показание:
   Event топик – mqtt:topic:maket:motion_sensor_1:value/out
10. **Кондиционер**

Кухня

На кухне находятся следующие устройства:

1. **Розетка двойная8**
2. **Холодильник**
3. **Плита 2**
4. **Печь**
5. **Микроволновая печь**
6. **Кофемашина 2**
7. **Чайник**
8. **Розетка двойная7**
9. **Датчик температуры 1**

   MQTT – статус:
   Event топик – mqtt:topic:maket:temperature_sensor:status/out
   Command топик – mqtt:topic:maket:temperature_sensor:status/in
   MQTT – показание:
   Event топик – mqtt:topic:maket:temperature_sensor:value/out
   Command топик – mqtt:topic:maket:temperature_sensor:value/in

   Детектор дыма 3
   Event топик – mqtt:topic:maket:smoke_detector_2:status/out
   Command топик – mqtt:topic:maket:smoke_detector_2:status/in
   MQTT – показание:
   Event топик – mqtt:topic:maket:smoke_detector_2:value/out
   Command топик – mqtt:topic:maket:smoke_detector_2:value/in
10. **Датчик огня 1**

    MQTT – статус:
    Event топик – mqtt:topic:maket:fire_sensor:status/out
    Command топик – mqtt:topic:maket:fire_sensor:status/in

MQTT – показание:

Event топик – mqtt:topic:maket:fire_sensor:value/out

Command топик – mqtt:topic:maket:fire_sensor:value/in

11. **Переключатель свет кухня**

MQTT – переключатель:

Event топик – mqtt:topic:maket:kitchen_light_switch/out

Command топик – mqtt:topic:maket:kitchen_light_switch/in

1. **Лампа потолочная 4**
2. **Лампа потолочная 4**
3. **Лампа потолочная 4**
4. **Лампа потолочная 4**

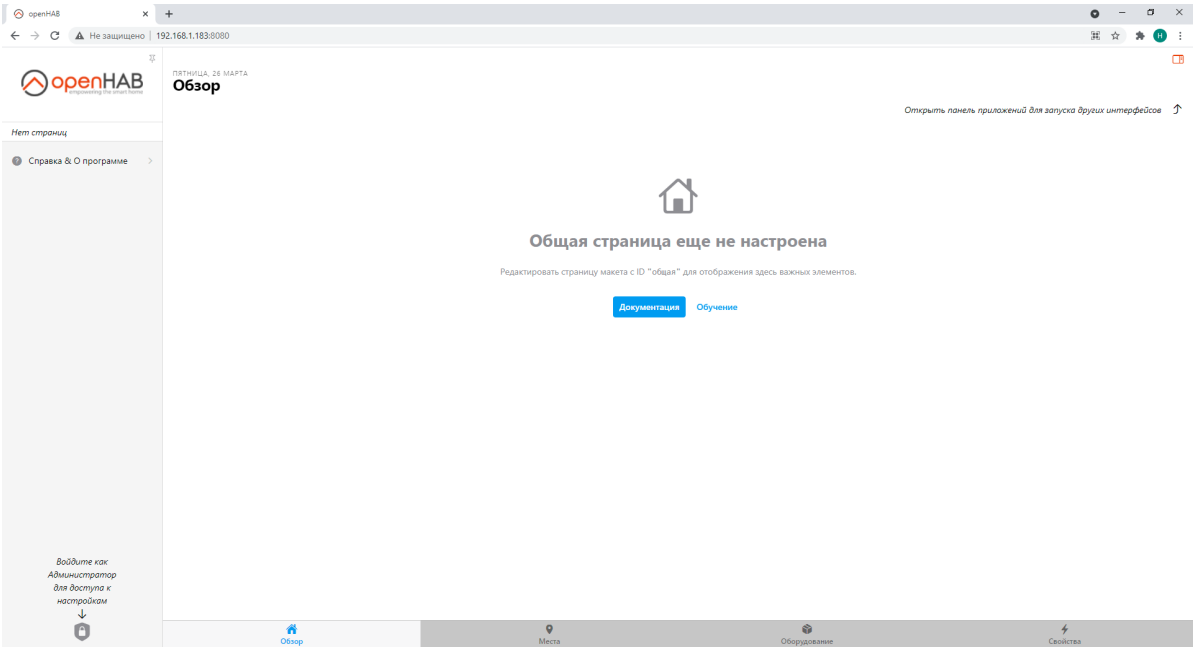
Столовая

В столовой находятся следующие устройства:

1. **Батарея**
2. **Розетка двойная2**
3. **Переключатель4**
4. **Лампа потолочная 4**
5. **Лампа потолочная 4**
6. **Лампа потолочная 4**

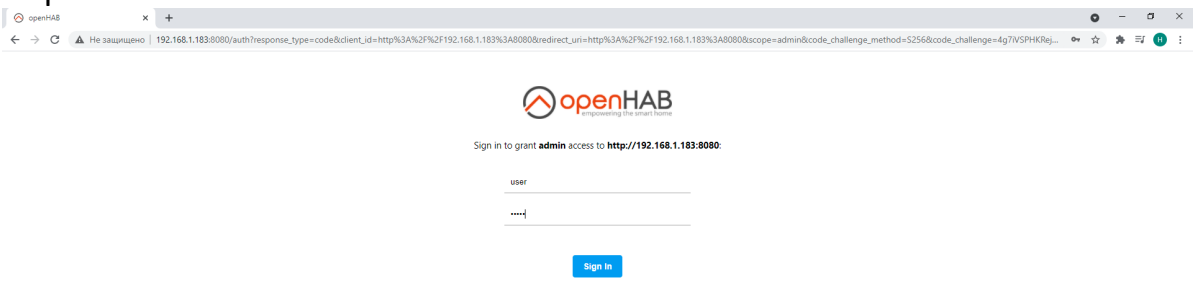
OpenHAB

Для подключения к OpenHab введите в адресной строке для макета 192.168.1.183:8080



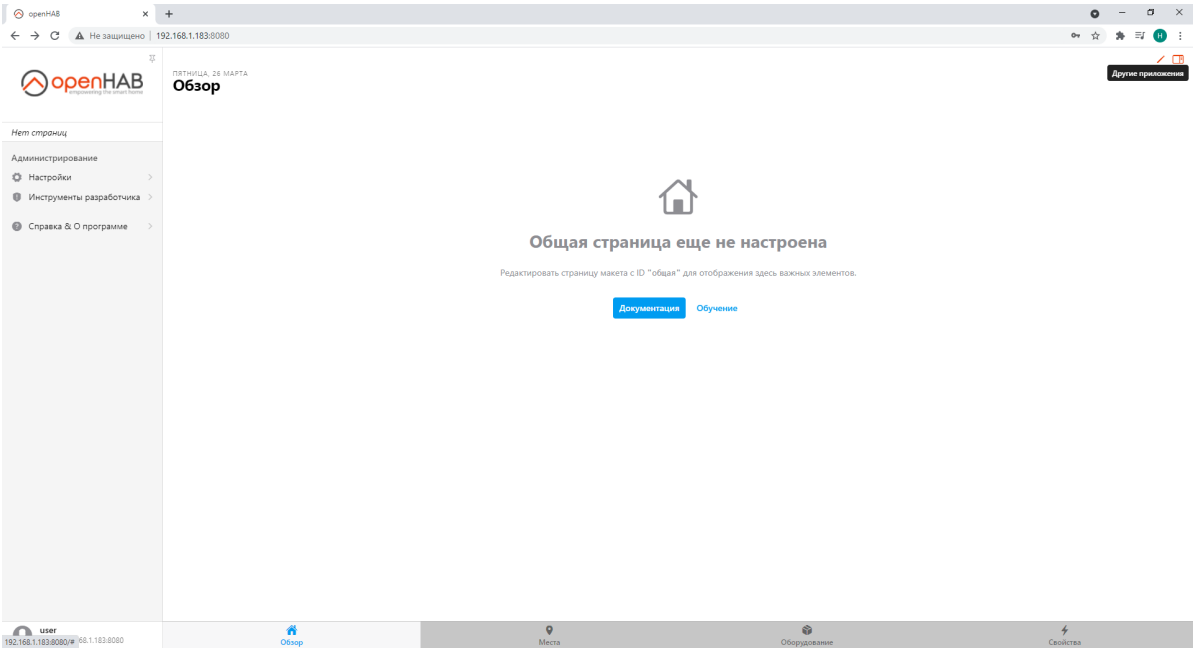
Подключение к OpenHab

Войдите как администратор для доступа к настройкам.
Логин: user
Пароль: 12345



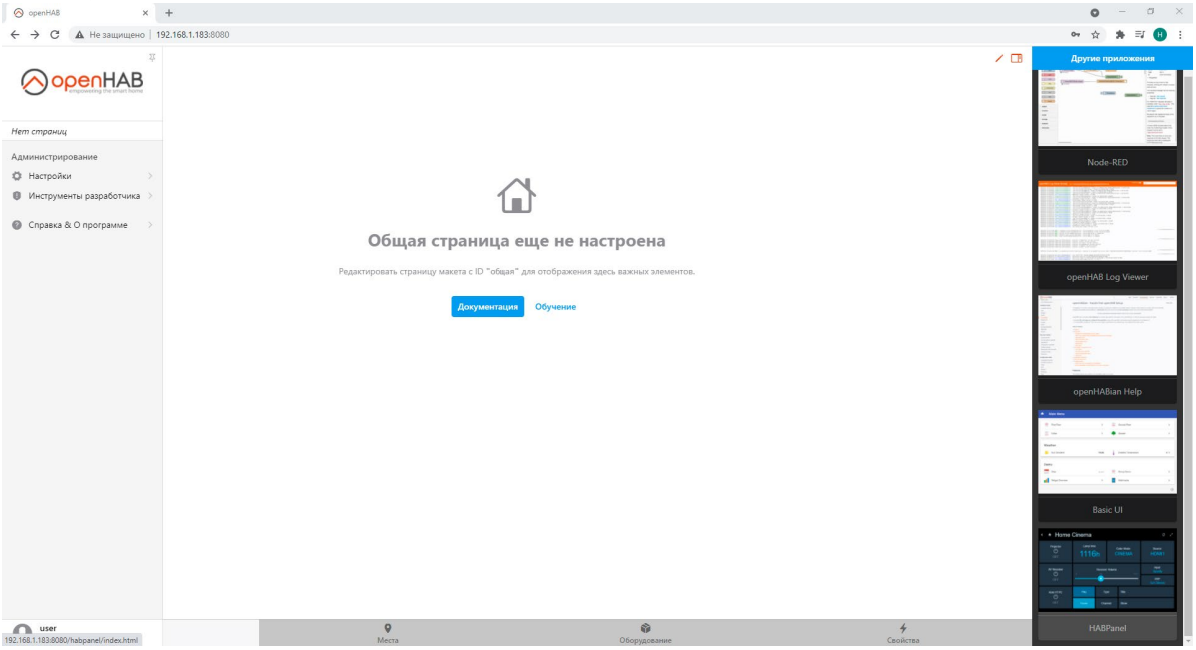
Авторизация

Основной экран представлен ниже. Нажмите на кнопку "**Другие приложения**".



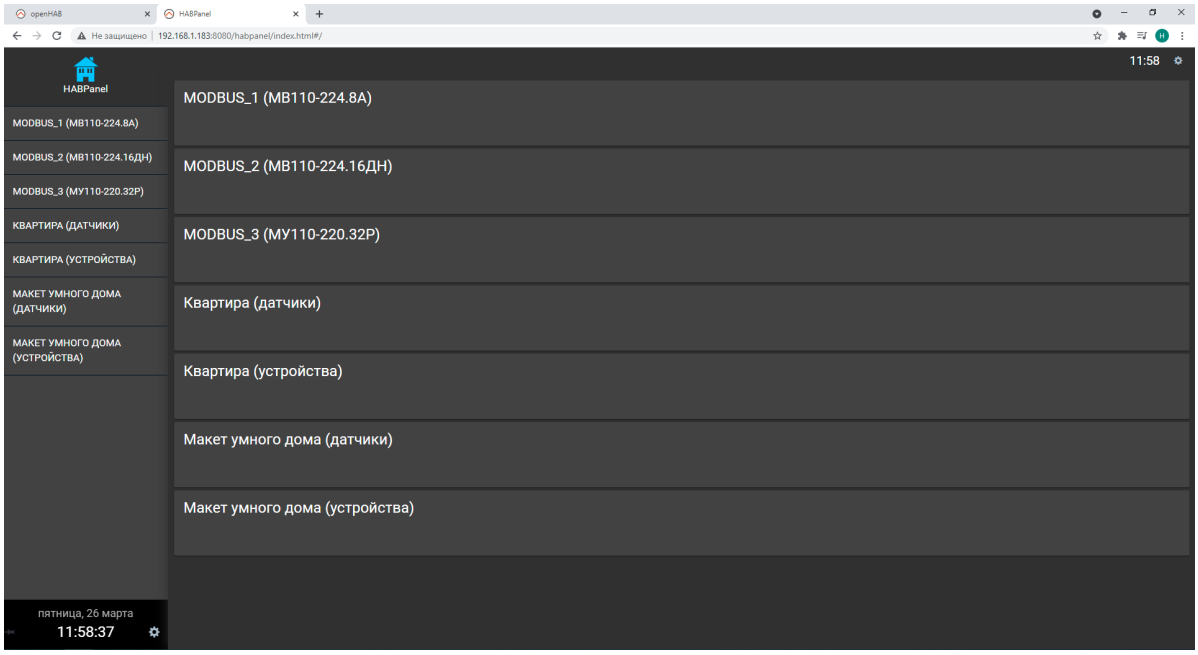
Основной экран программы

Выберите HABPanel



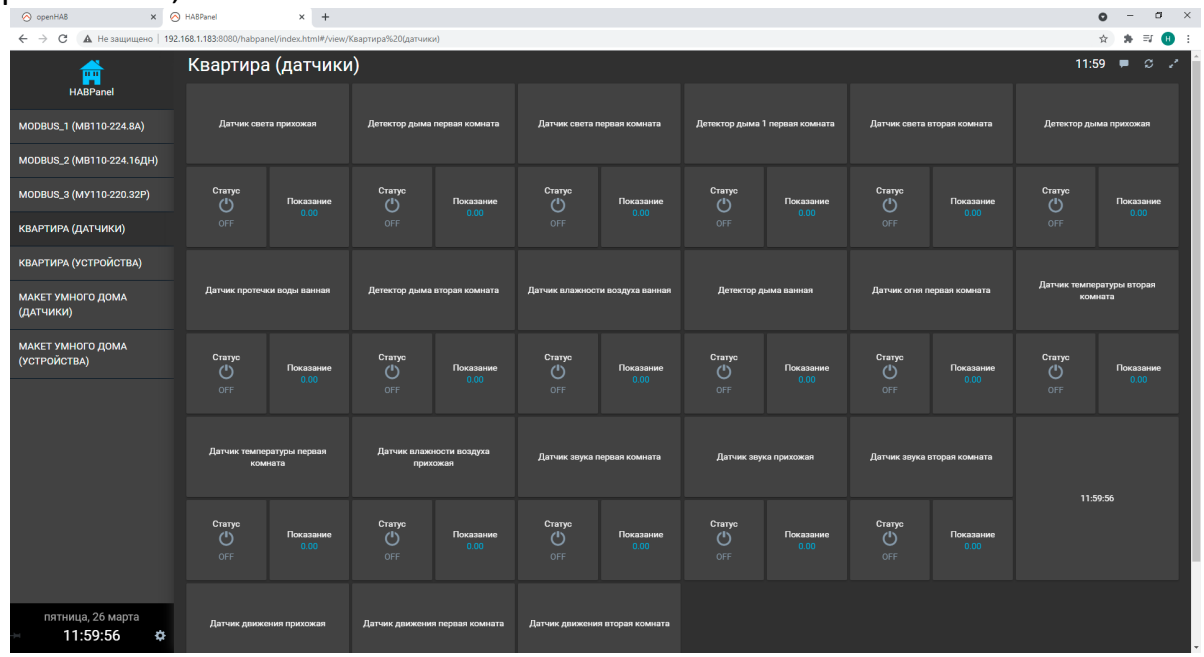
Другие приложения

Основной экран HABPanel



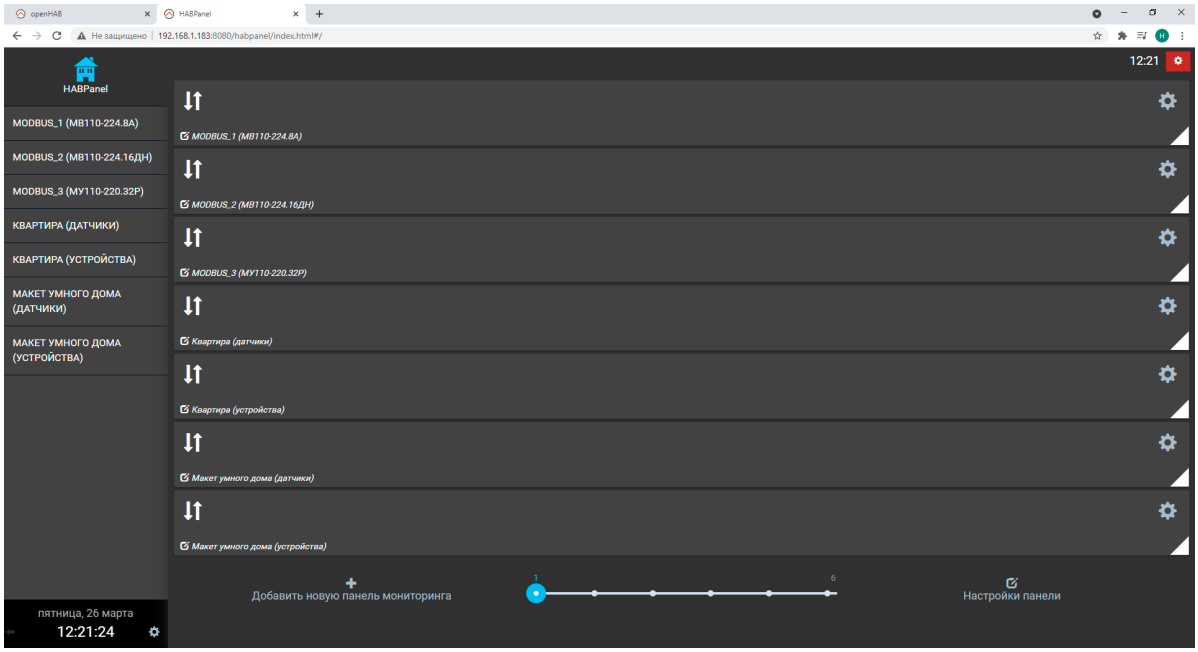
Экран HABPanel

В данном приложении вы можете управлять различными датчиками и устройствами, отслеживать показания.



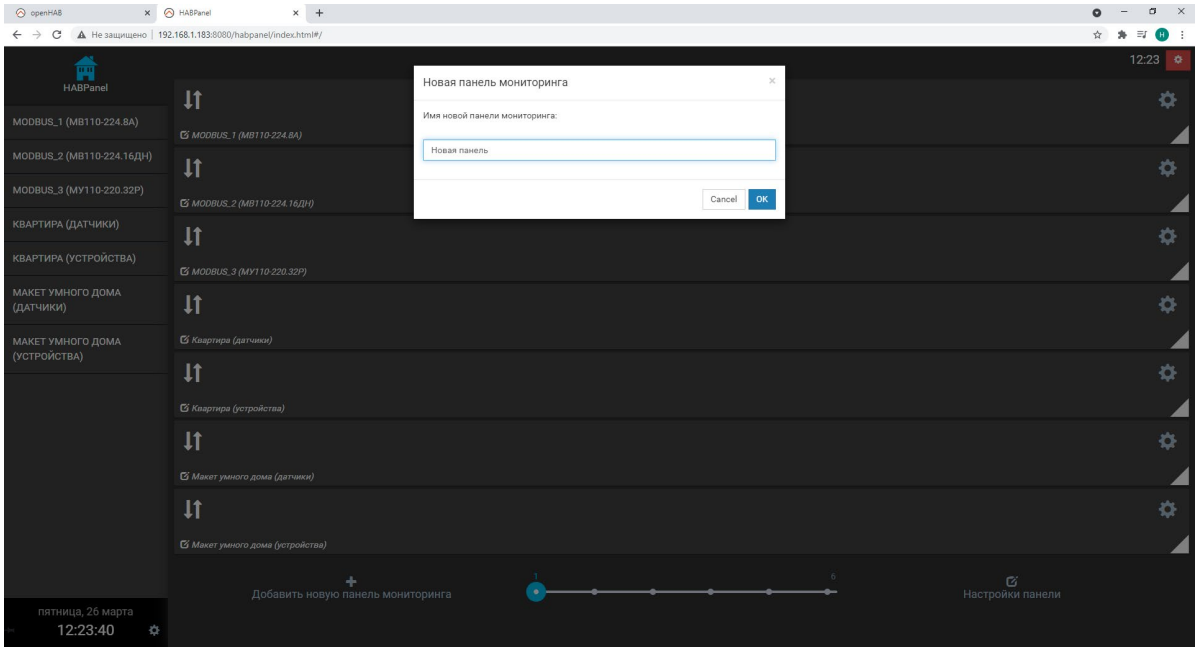
Управление датчиками

При нажатии на шестеренку в правом верхнем углу можно создать свою панель



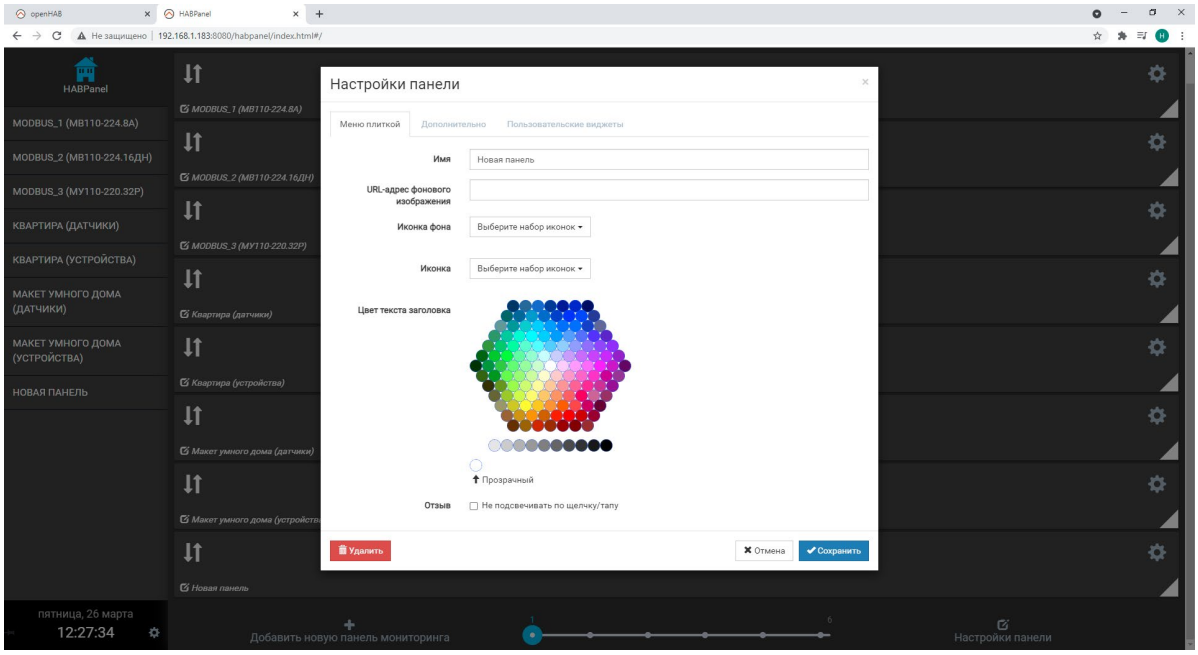
Настройки

Для добавление панели нажмите "**Добавить новую панель мониторинга**" введите имя новой панели и нажмите "**Ок**".



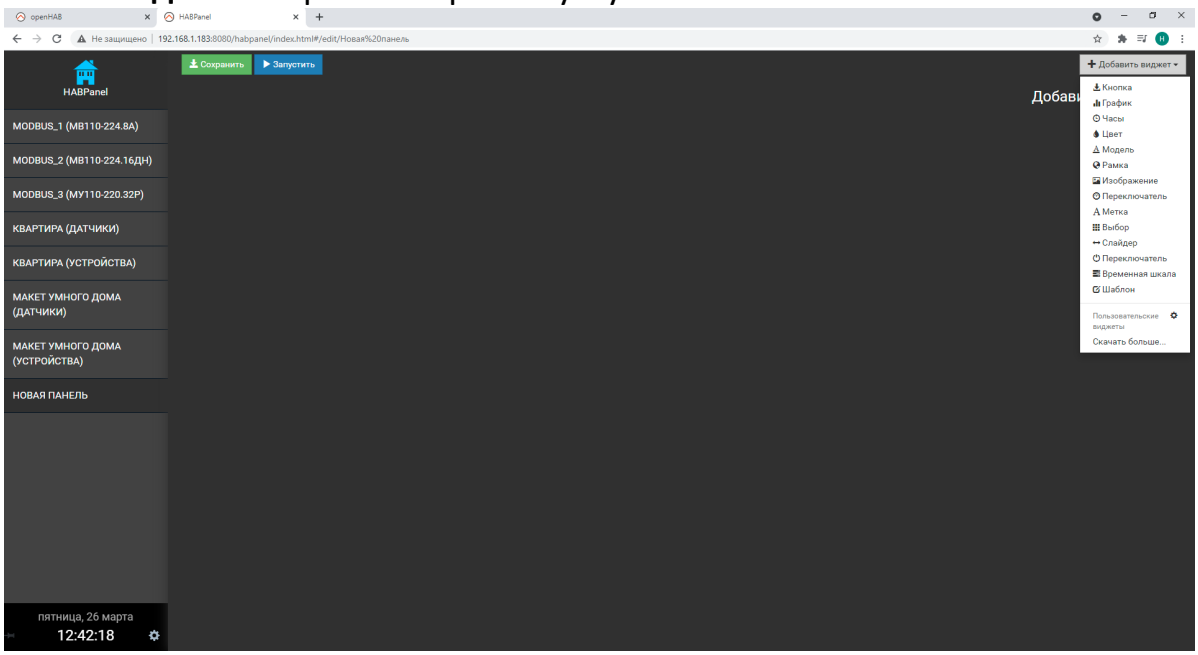
Создание новой панели мониторинга

После она добавится в общий список панелей, далее нажмите на шестеренку на созданной панели для ее редактирования. После настройки панели нажмите "**Сохранить**". Для удаления панели нажмите "**Удалить**".



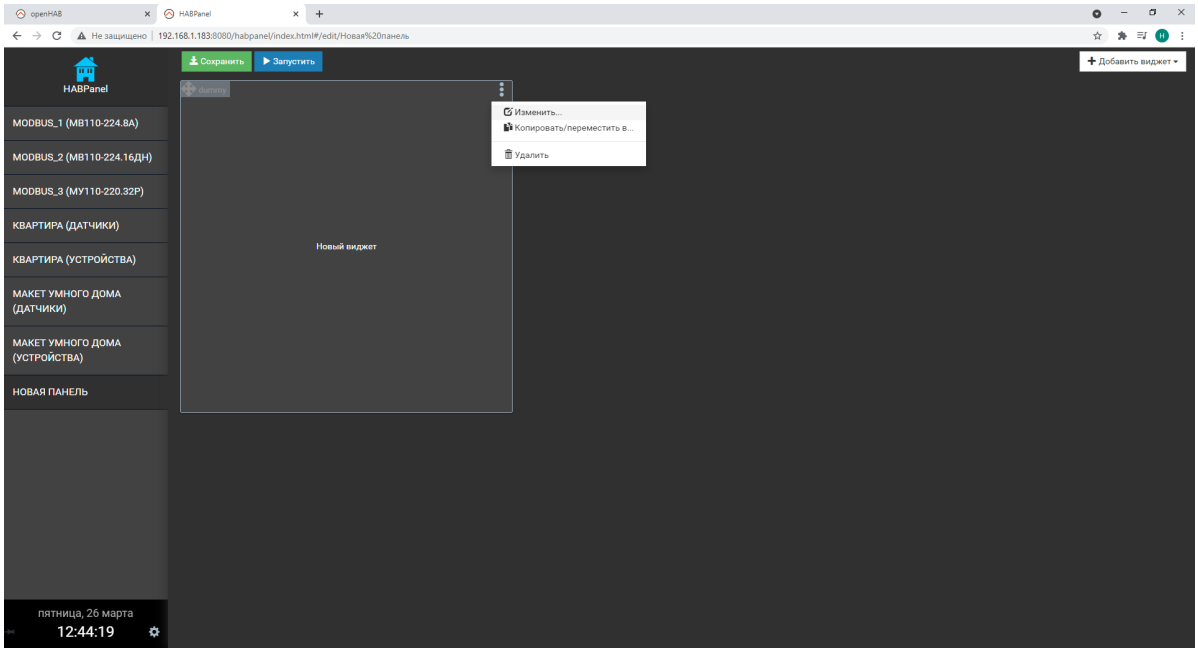
Настройки панели

После настройки зайдите в панель и добавьте виджет, для это нажмите "**+ Добавить виджет**" в правом верхнем углу.



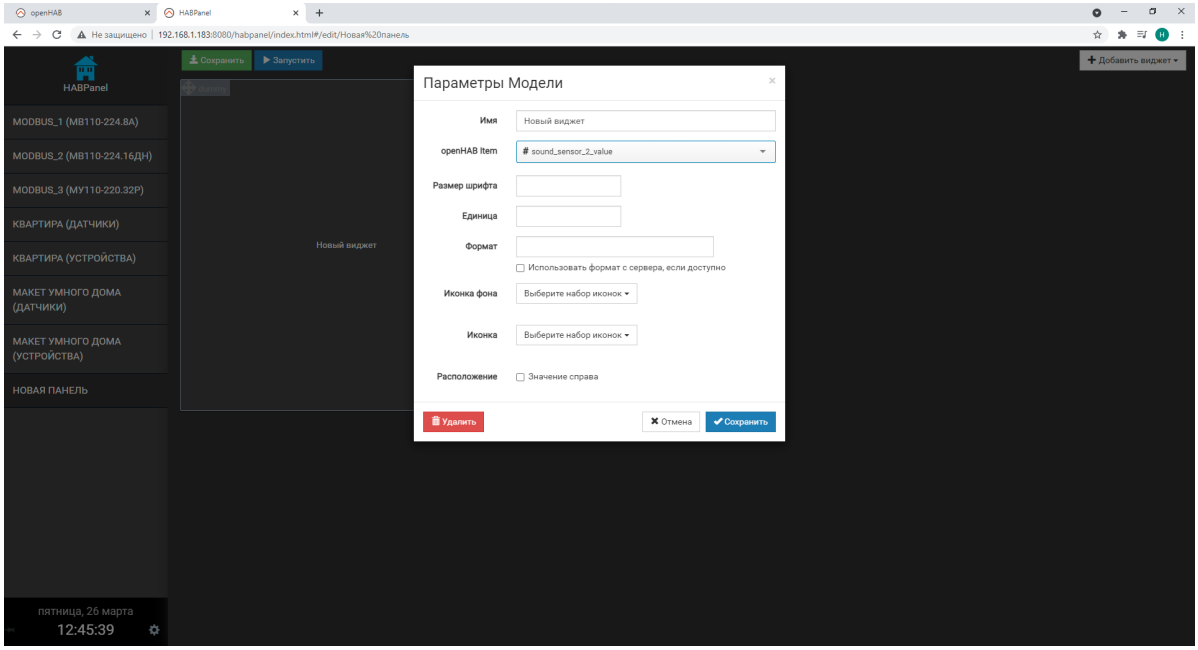
Добавление виджета

Далее измените виджет



Изменение виджета

Измените параметры и нажмите "**Сохранить**"



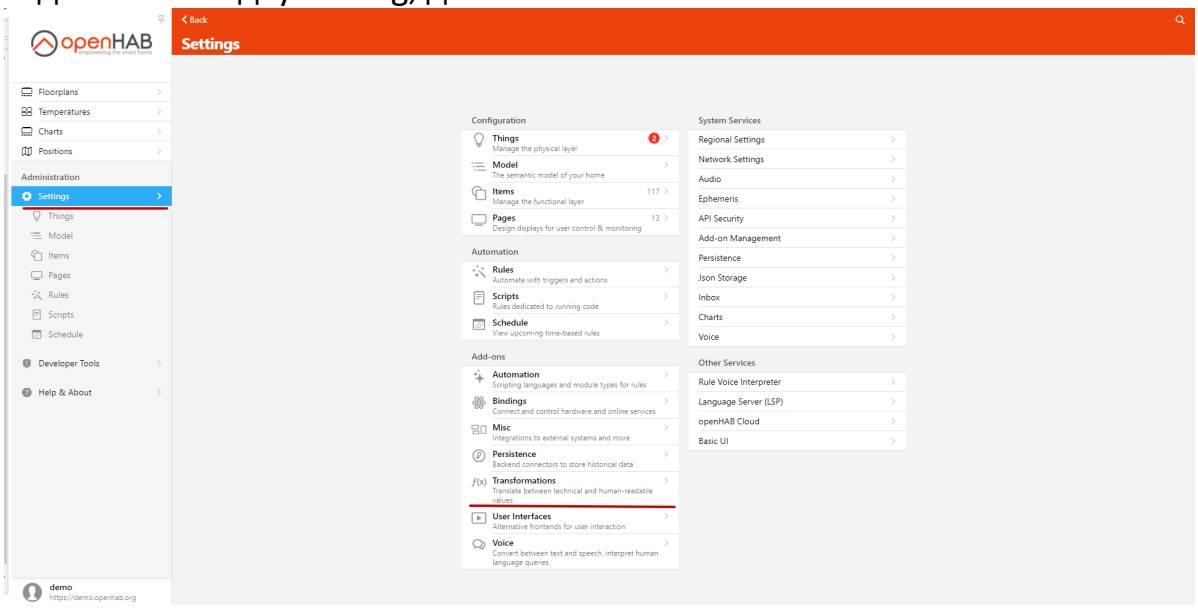
Параметры

Для самостоятельного создания новых устройств ознакомьтесь с руководством на сайте: https://openhab.org/docs/tutorial/

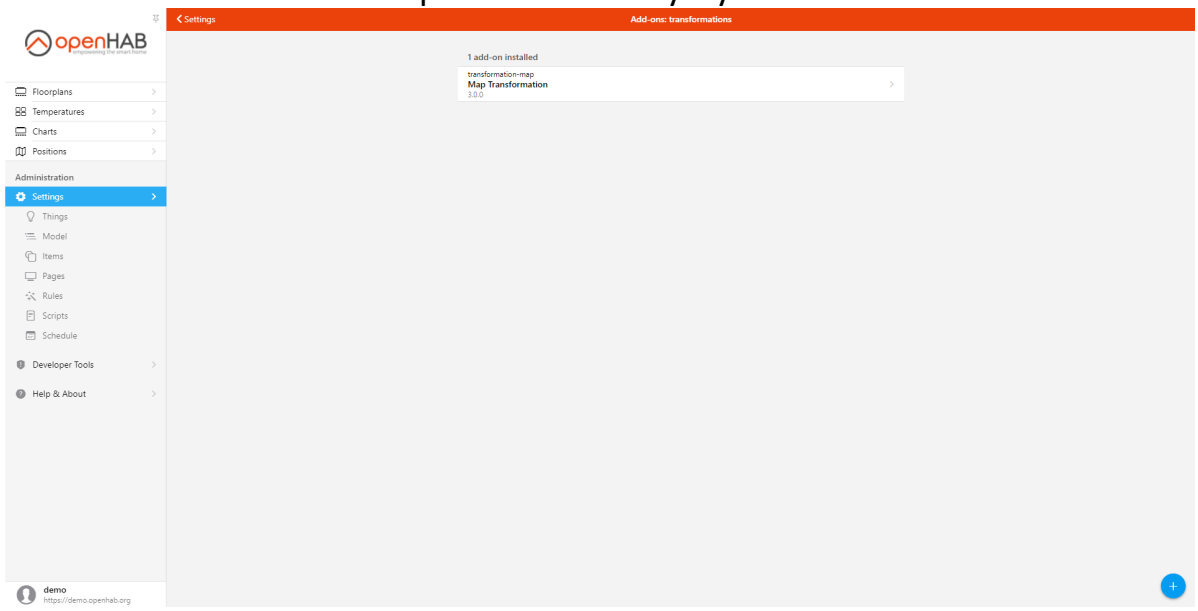
Первоначальная настройка

Для работы с JSON в OpenHAB необходимо скачать в OpenHAB **JSONPath Transformation**. Для этого:

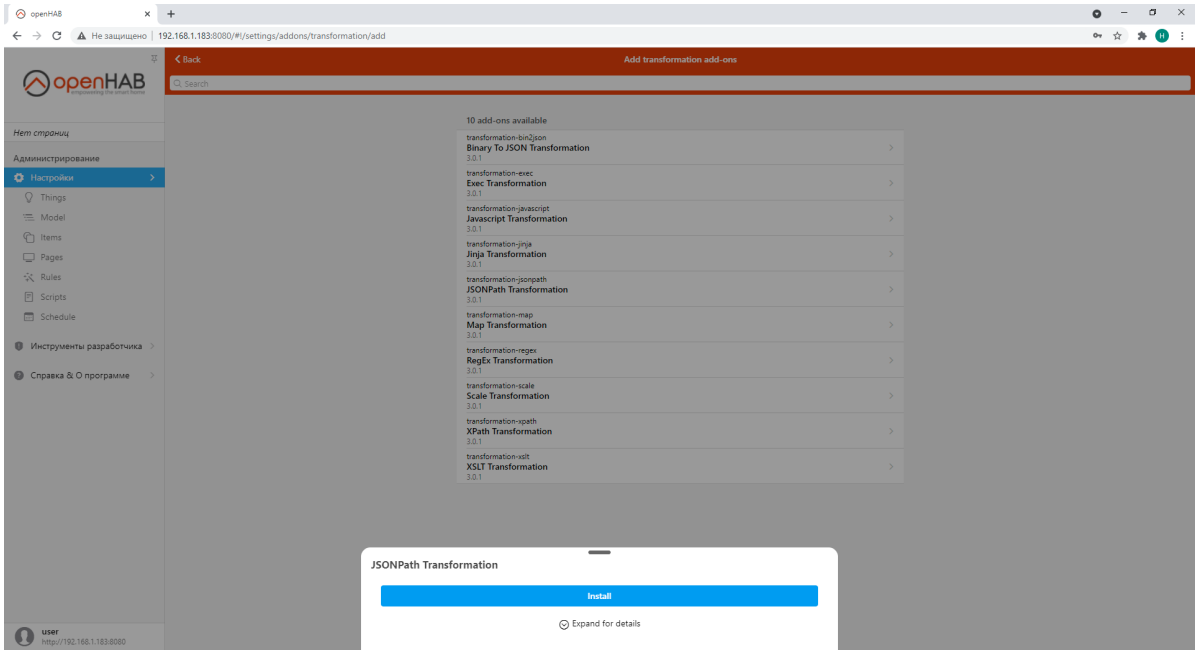
1. Зайдите во вкладку Setting, далее нажмите на Transformation:



2. После нажмите на плюс в правом нижнем углу.

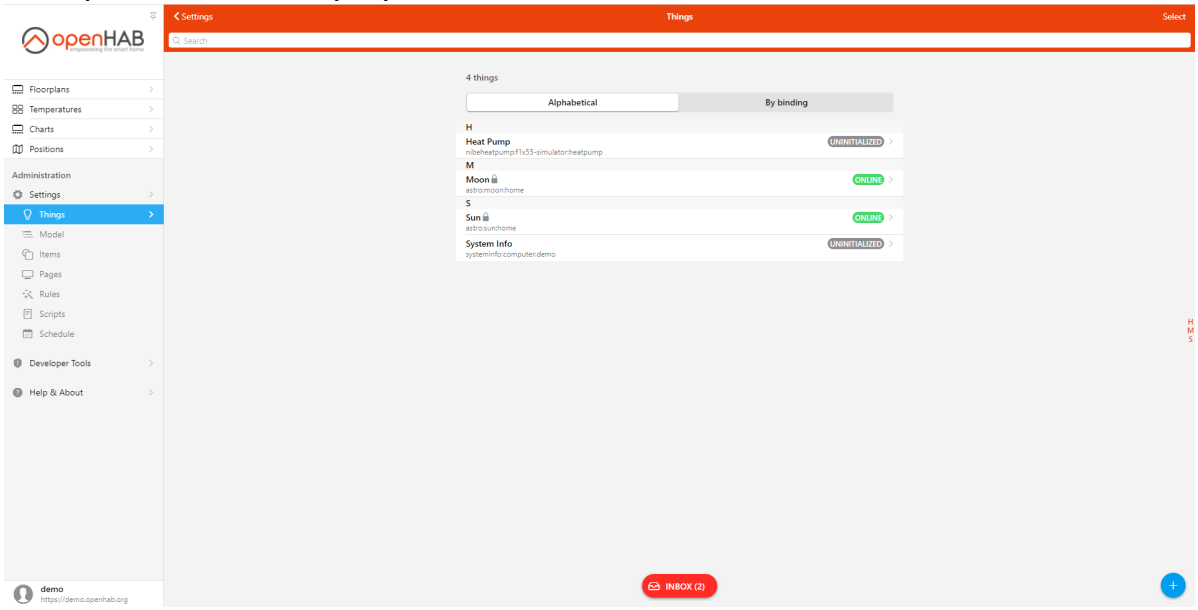


3. Далее нажмите на **JSONPath Transformation** и нажмите **Install**.

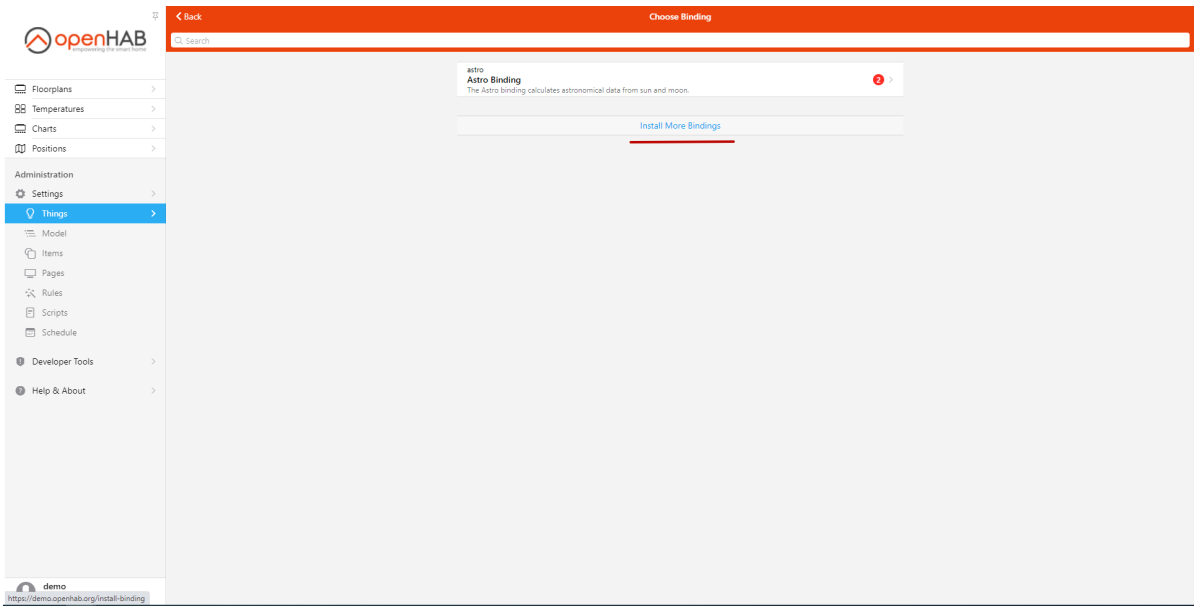


Добавление Bridge

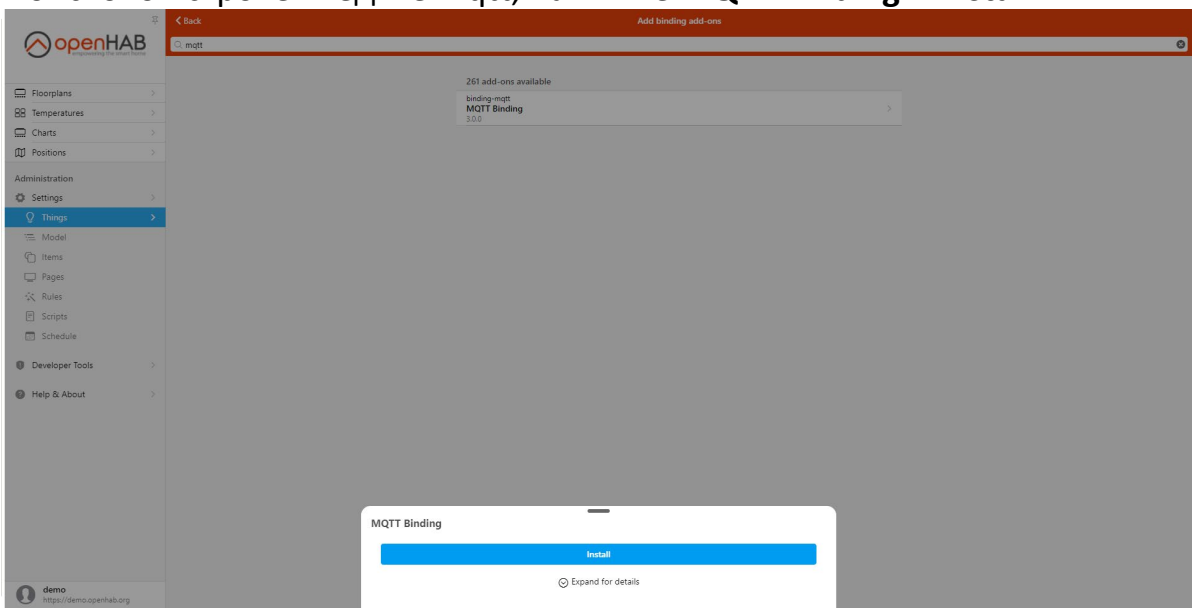
1. Переходим во вкладку Setting затем переходим в Things, далее нажмите на плюс в правом нижнем углу.



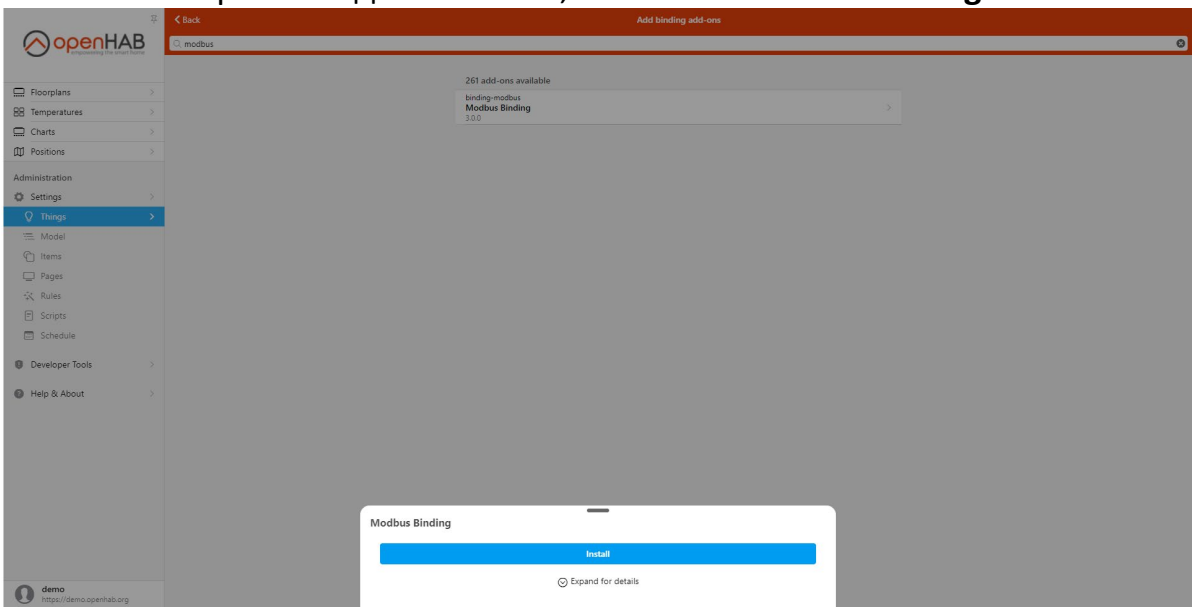
2. Далее нажмите **Install More Brindings**



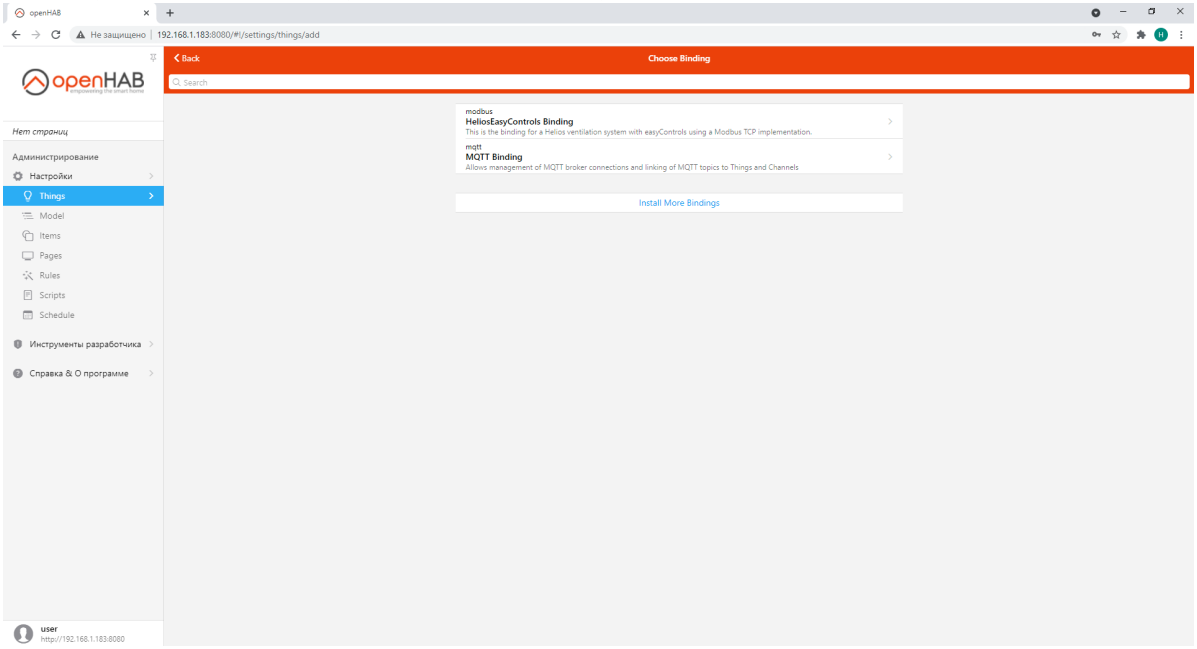
3. В поисковой строке введите mqtt, нажмите **MQTT Binding** и **Install**.



4. В поисковой строке введите modbus, нажмите **Modbus Binding** и **Install**.



После первоначальной настройки во вкладке Things появятся HeliosEasyControls Binding и MQTT Binding.

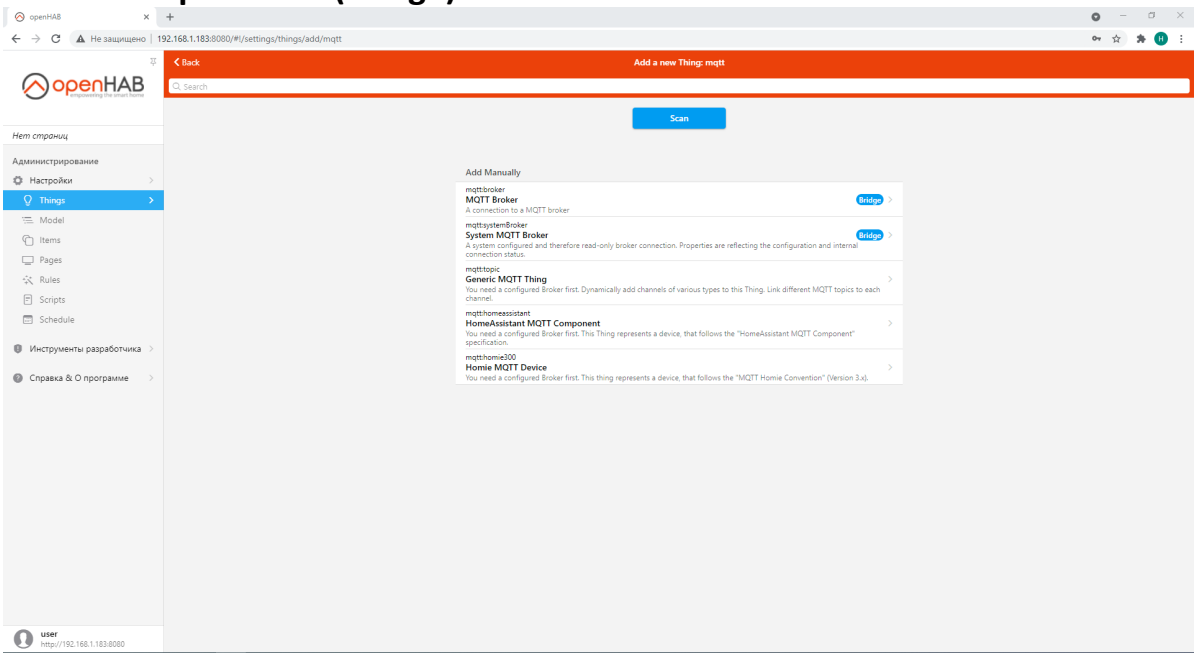


Для самостоятельного создания новых устройств ознакомьтесь с руководством на сайте: https://openhab.org/docs/tutorial/

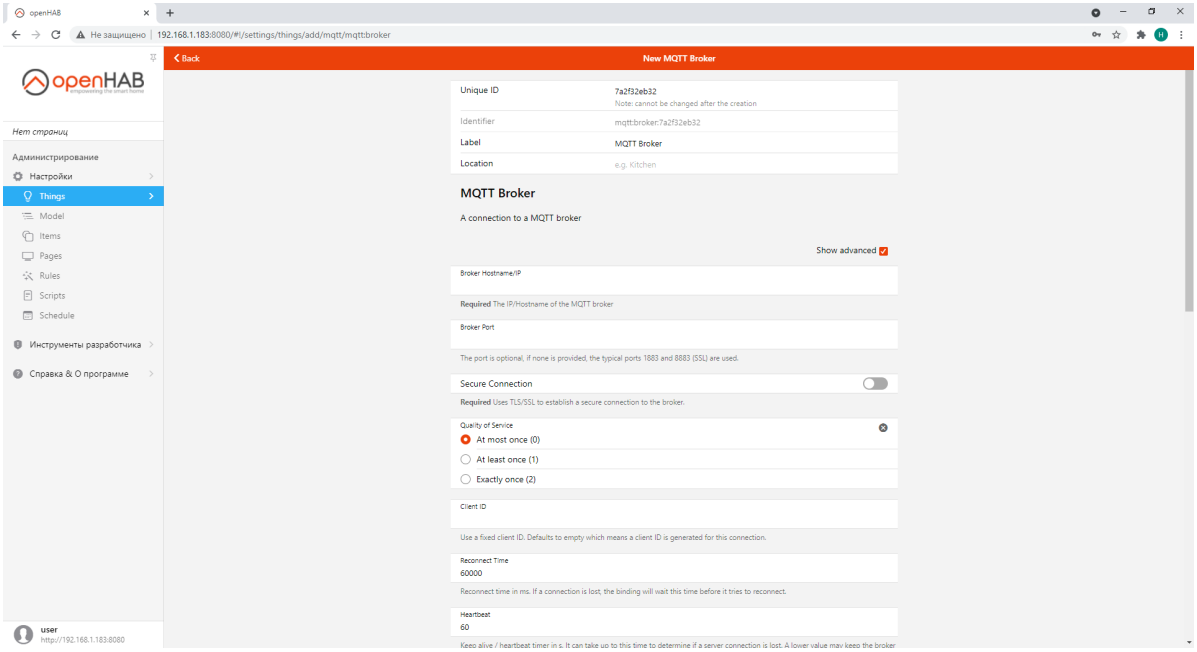
Создание Binding Bridge

Создание Mqtt Bridge

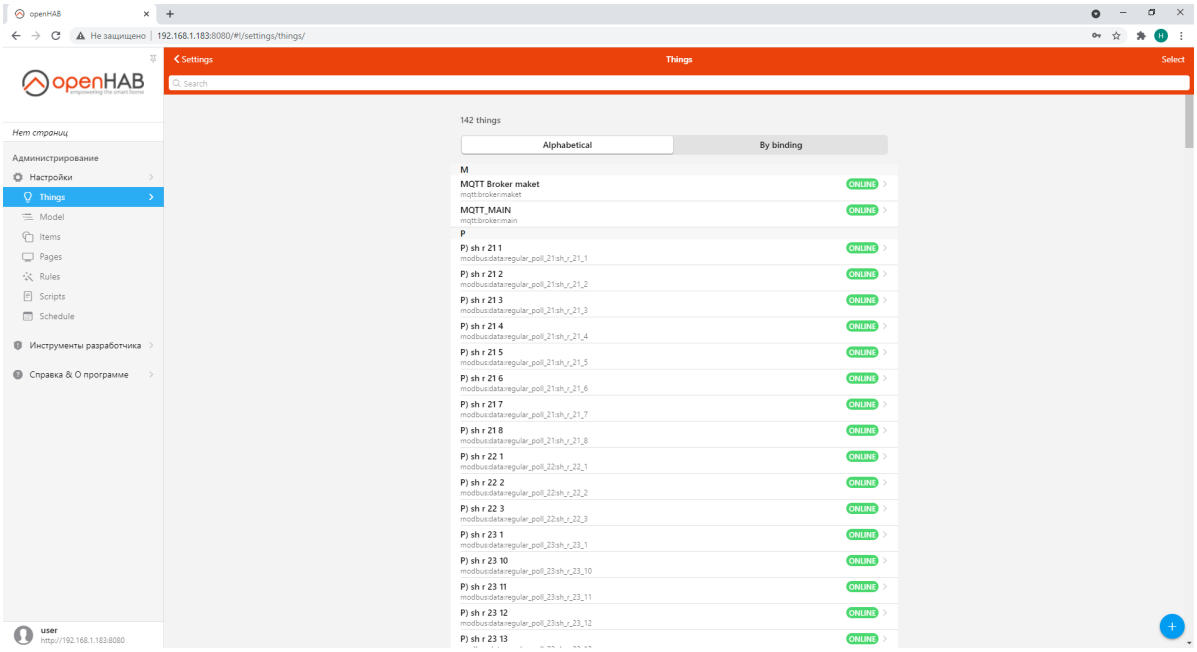
1. Зайдите во вкладку Setting, далее переходим в Things.
2. Нажмите на плюс в правом нижнем углу.
3. Нажмите **Mqtt binding**.
4. Нажмите **Mqtt broker (Bridge)**.



5. Укажите Unique ID, Label.
6. Проставьте галочка Show advanced.

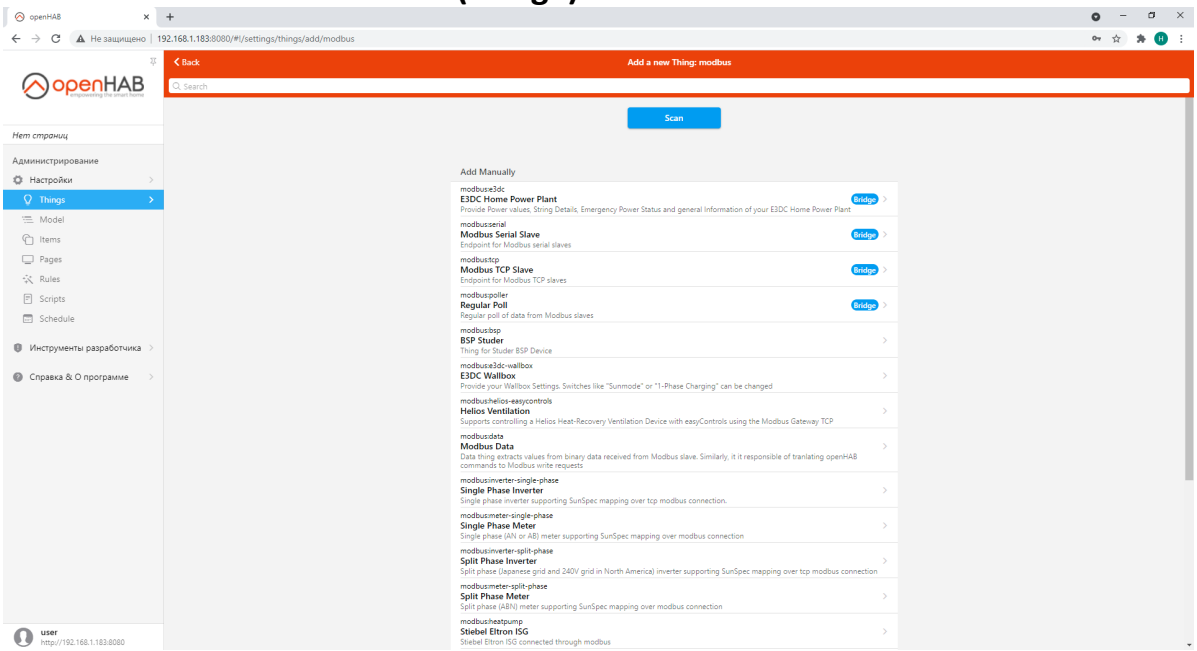


7. Укажите Broker Hostname/IP, Broker Port (1883).
8. Промотайте страничку вниз и нажмите **Create Thing**.
9. После создания в Things появится созданный Broker со знаком online, если при создании произошла ошибка, то знак будет offline красный.



Создание Modbus Bridge

1. Зайдите во вкладку Setting, далее переходим в Things.
2. Нажмите на плюс в правом нижнем углу.
3. Нажмите **HeliosEasyControls Binding**.
4. Нажмите **Modbus Serial Slave (Bridge)**.



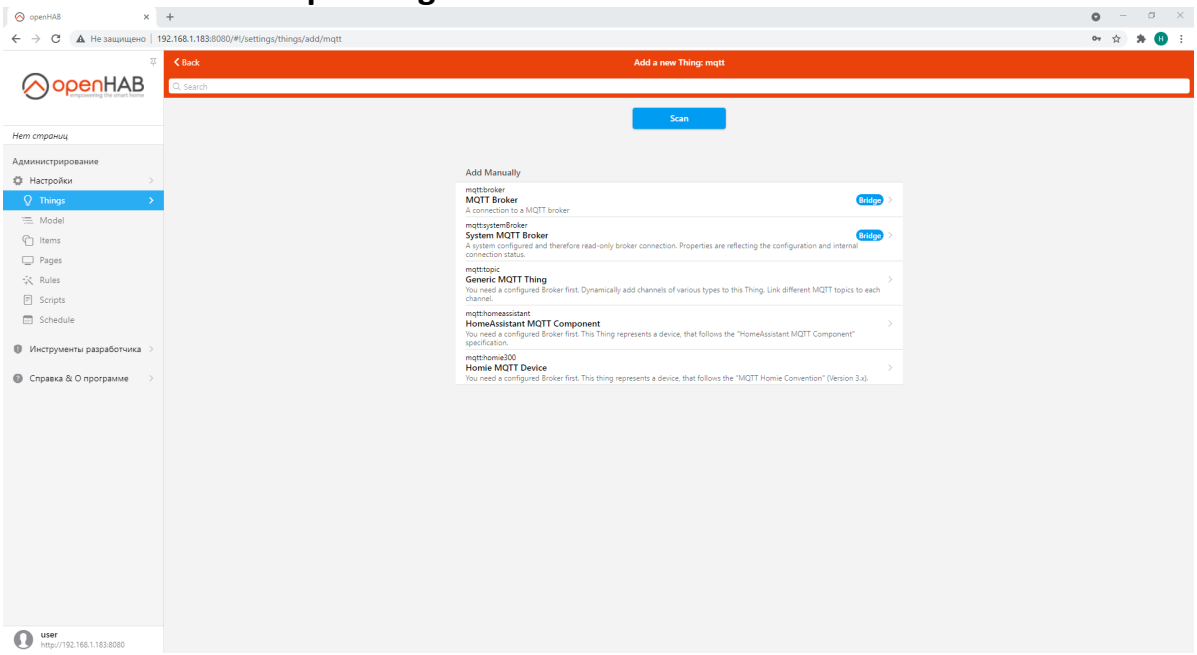
5. Укажите Unique ID, Label.
6. Укажите Serial Port, Id.
7. Установите настройки согласно вашему устройству.
8. Нажмите **Create Thing**.

Для самостоятельного создания новых устройств ознакомьтесь с руководством на сайте: https://openhab.org/docs/tutorial/

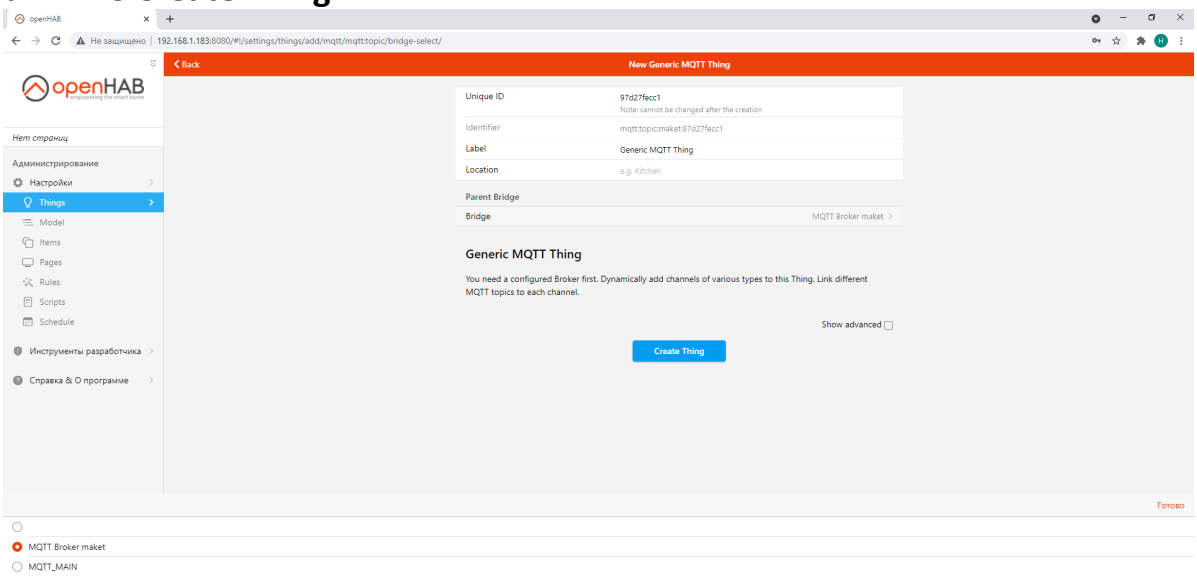
Создание Things

Создание Mqtt Things

1. Зайдите во вкладку Setting, далее переходим в Things.
2. Нажмите на плюс в правом нижнем углу.
3. Нажмите **Mqtt binding**.
4. Нажмите **Generic Mqtt Thing.**



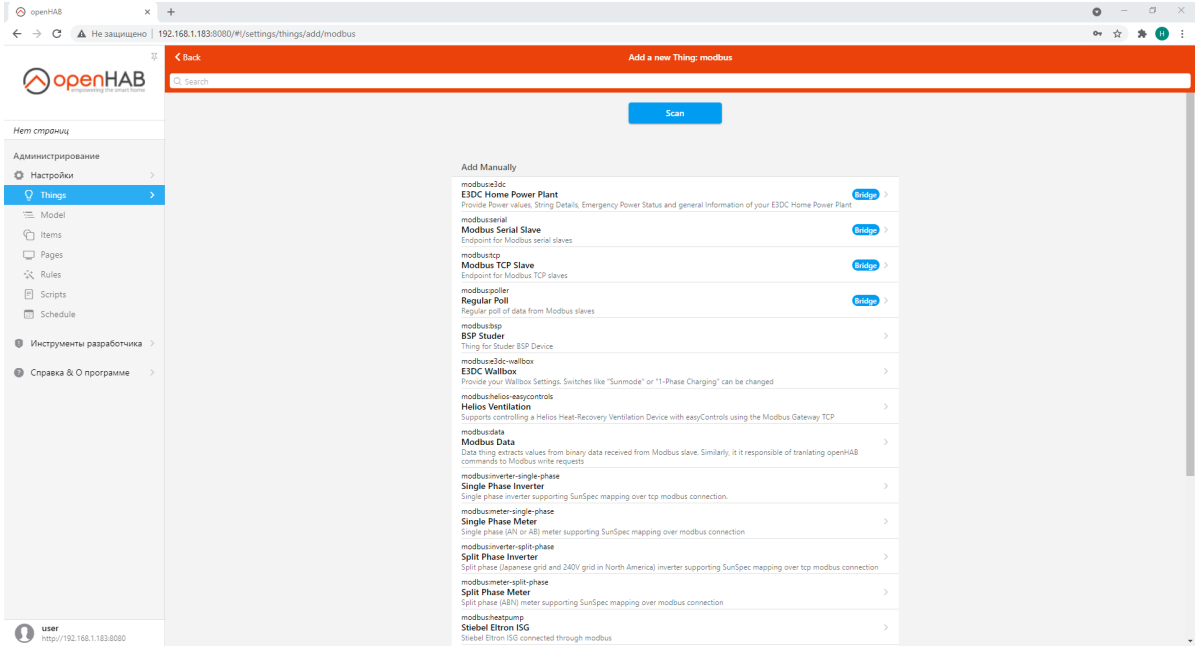
5. Укажите Unique ID, Label.
6. Нажмите **Bridge** и выберите ранее созданный Mqtt Broker.
7. Нажмите **Create Thing**.



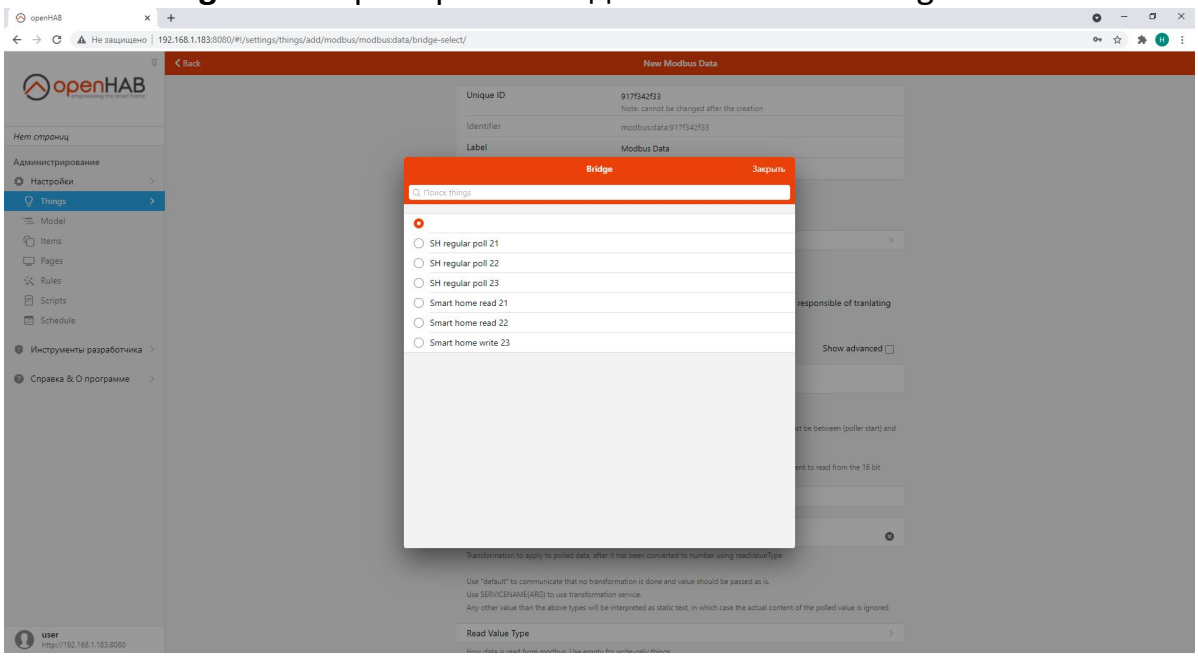
Создание Modbus Things

1. Зайдите во вкладку Setting, далее переходим в Things.

2. Нажмите на плюс в правом нижнем углу.
3. Нажмите **HeliosEasyControls Binding**.
4. Нажмите **Modbus Data.** По умолчанию настройка на запись (Write), чтобы сделать на чтение (Read) необходимо дополнительно создать Regular Poll (Bridge).



5. Укажите Unique ID, Label.
6. Нажмите **Bridge** и выберите ранее созданный Modbus Bridge.



7. Установите настройки согласно вашему устройству.
8. Нажмите **Create Thing**.

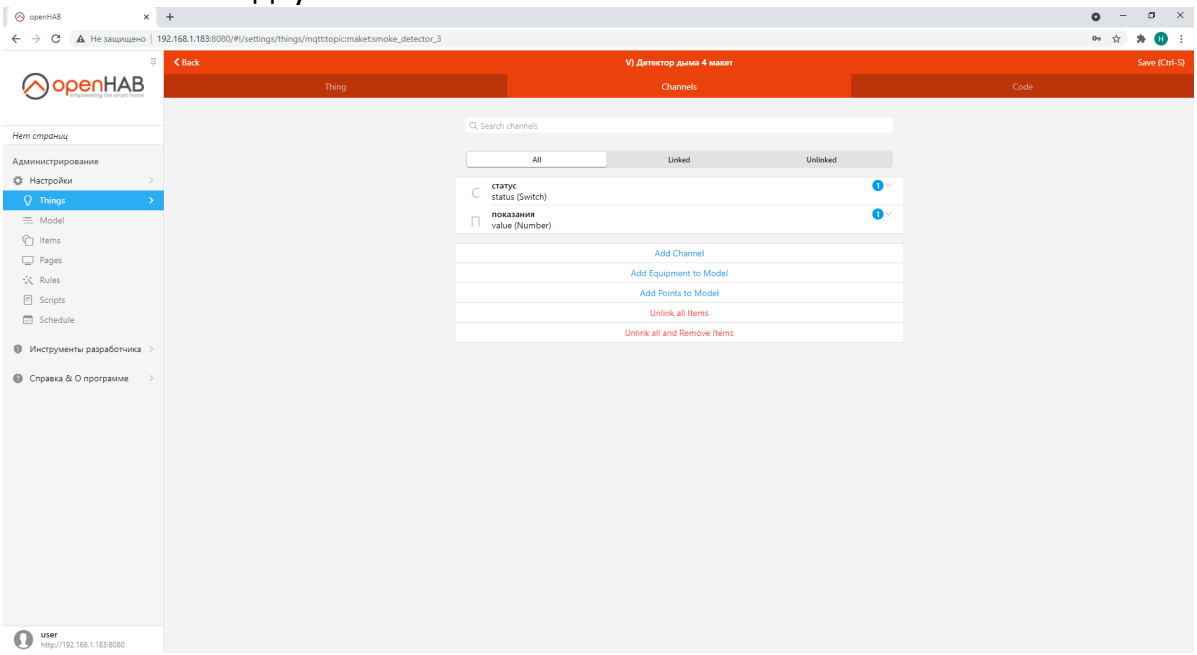
Для самостоятельного создания новых устройств ознакомьтесь с руководством на сайте: https://openhab.org/docs/tutorial/

Создание Channels

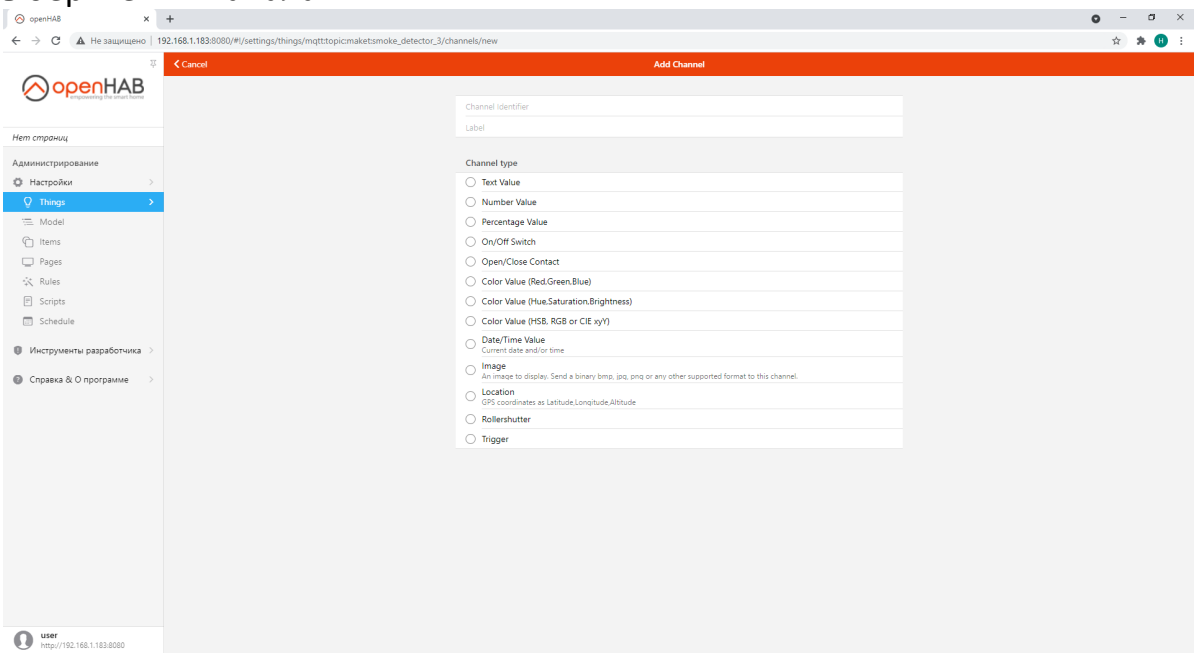
Создание Channels в Mqtt

Ниже представлен пример создания Channels в Mqtt.

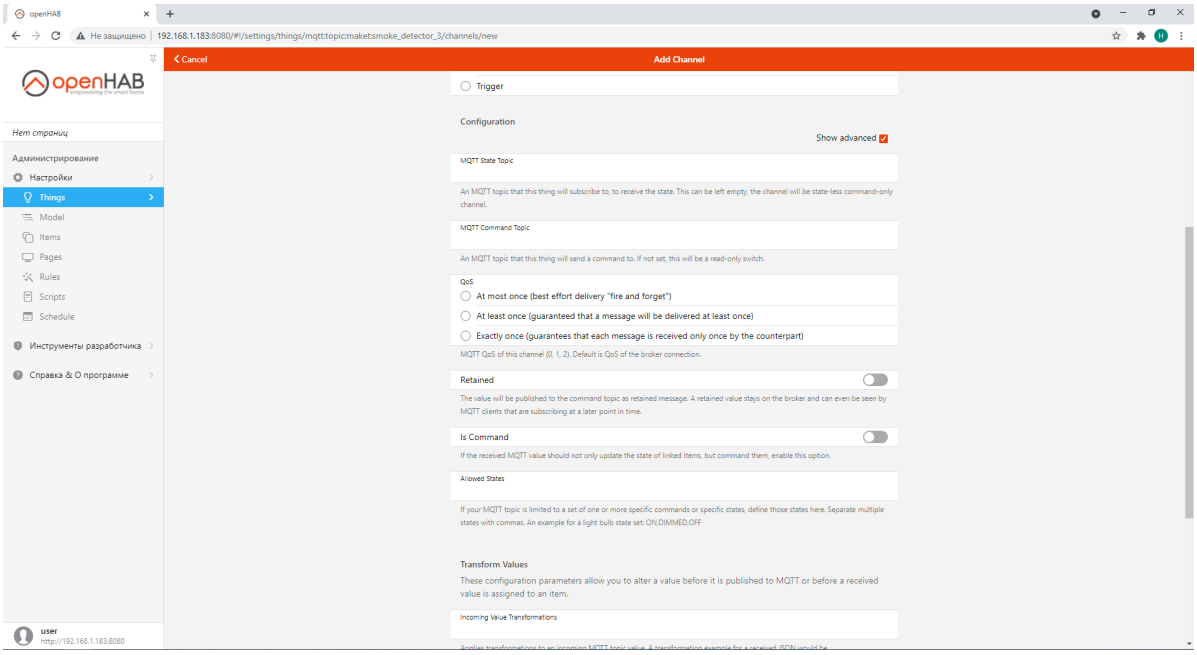
1. Зайдите во вкладку Setting, далее переходим в Things.
2. Нажмите на уже созданный Mqtt Things.
3. Нажмите на вкладку **Channels**.



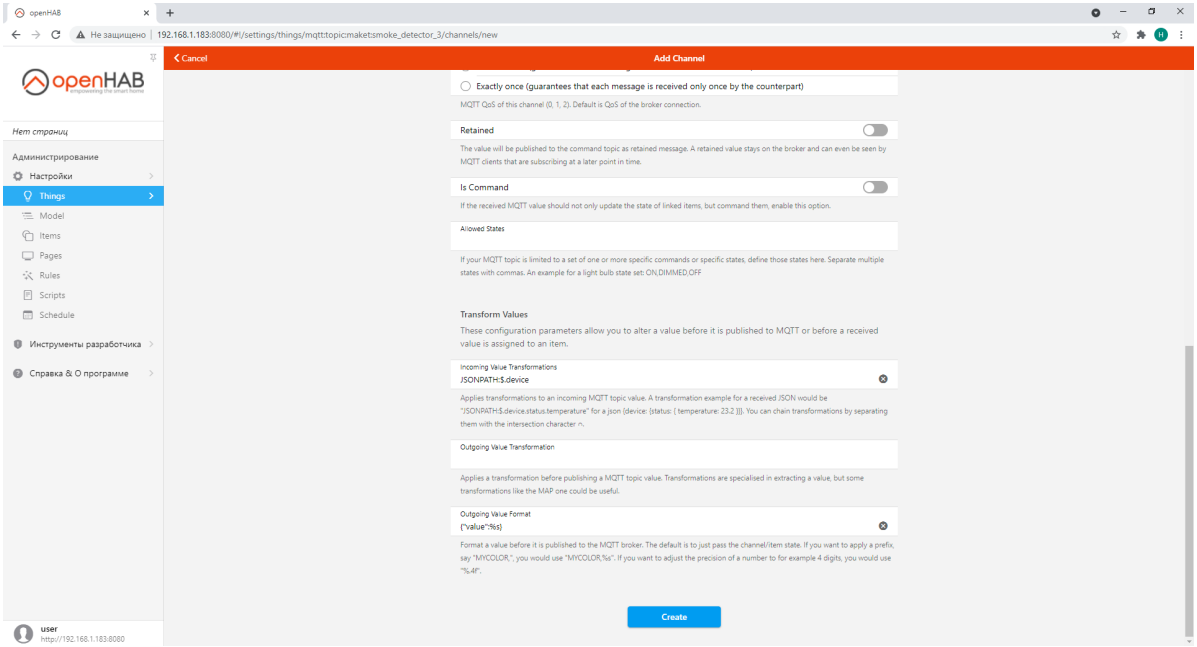
4. Нажмите на **Add Channel**.
5. Введите Channel Identifier и Label.
6. Выберите тип канала.



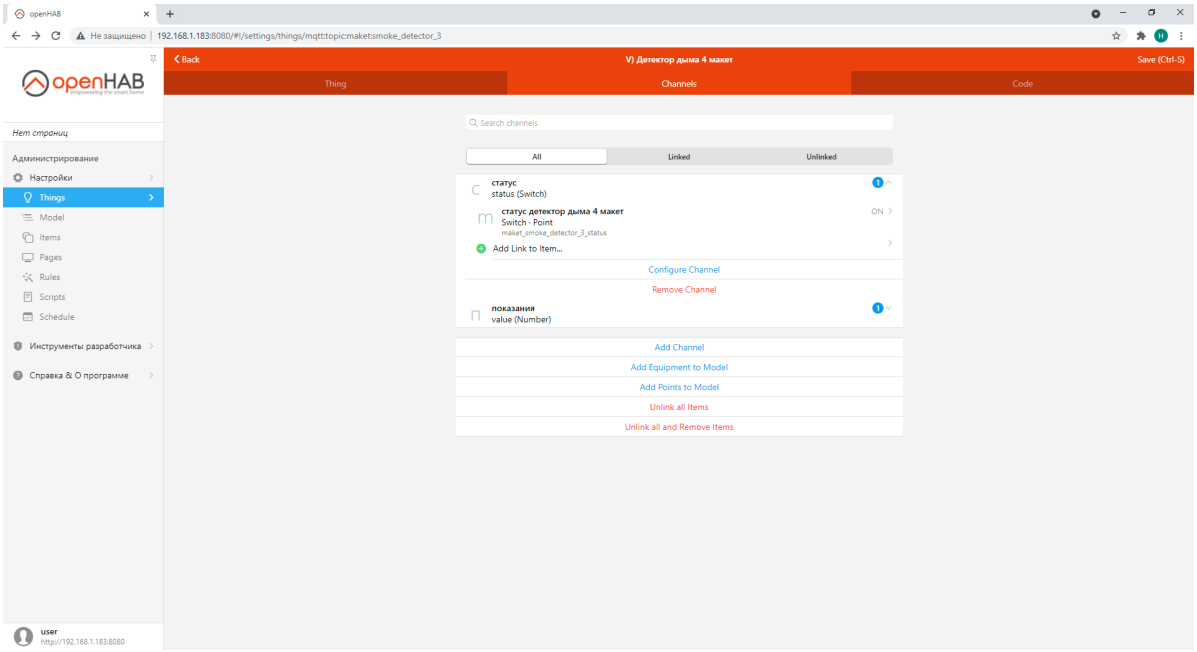
7. Проставьте галочка Show advanced.
8. Введите свои Topics на MQTT State и MQTT Command.



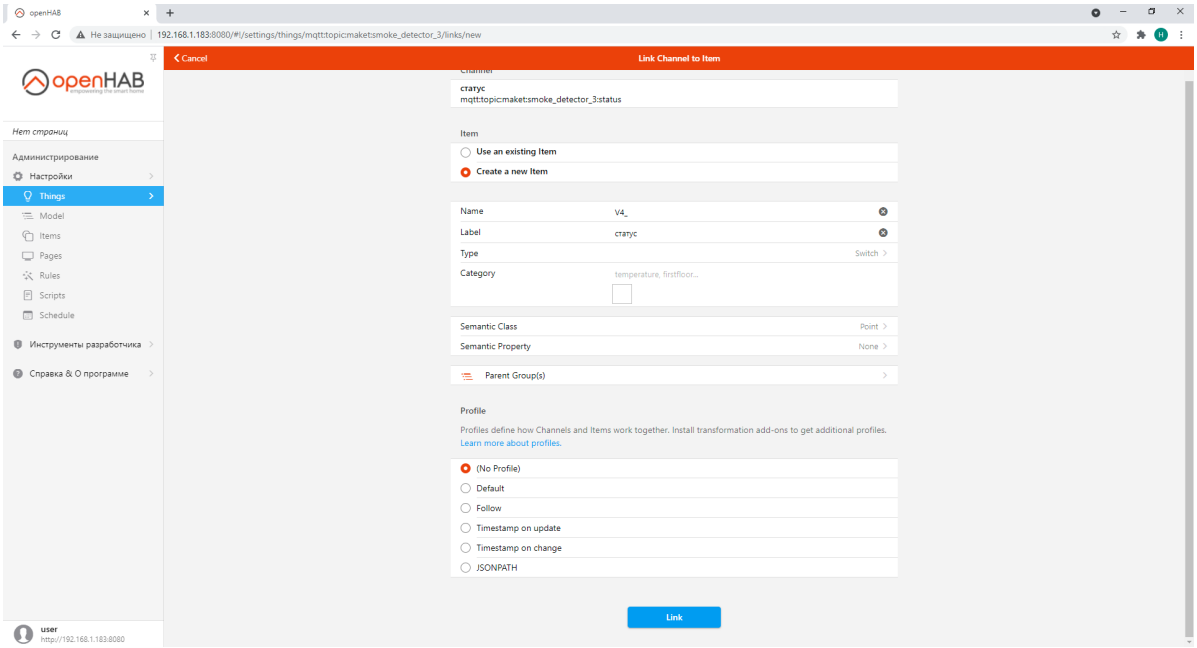
9. Чтобы данные принимать и обрабатывать в формате JSON в разделе Transform Values в графе Incoming Value Transformations введи путь JSONPATH:$.device
10. Чтобы отправлять данные в формате JSON в графе Outgoing Value Format введите формат исходящих данных согласно вашей структуре {"value":%s}
11. Нажмите **Create**.



12. В Channels появится канал.
13. Для взаимодействия с каналом нажмите **Add Link to Item**



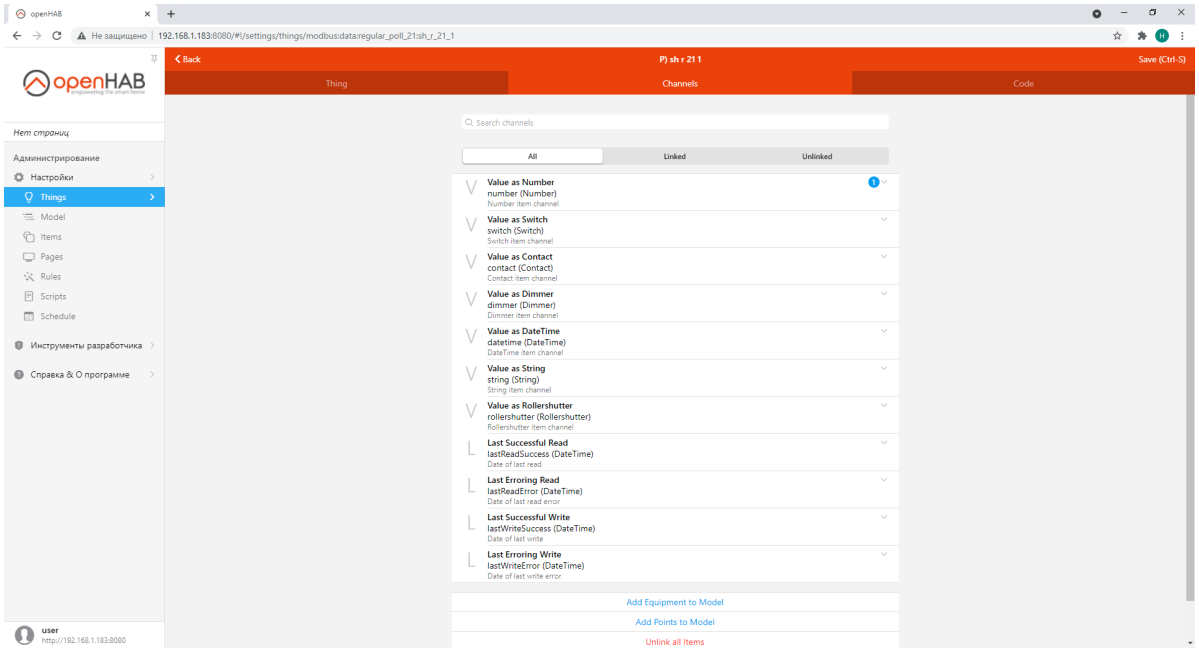
14. Нажмите **Create a new Item**.
15. Укажите Name, Label, Type.
16. Нажмите **Link**.



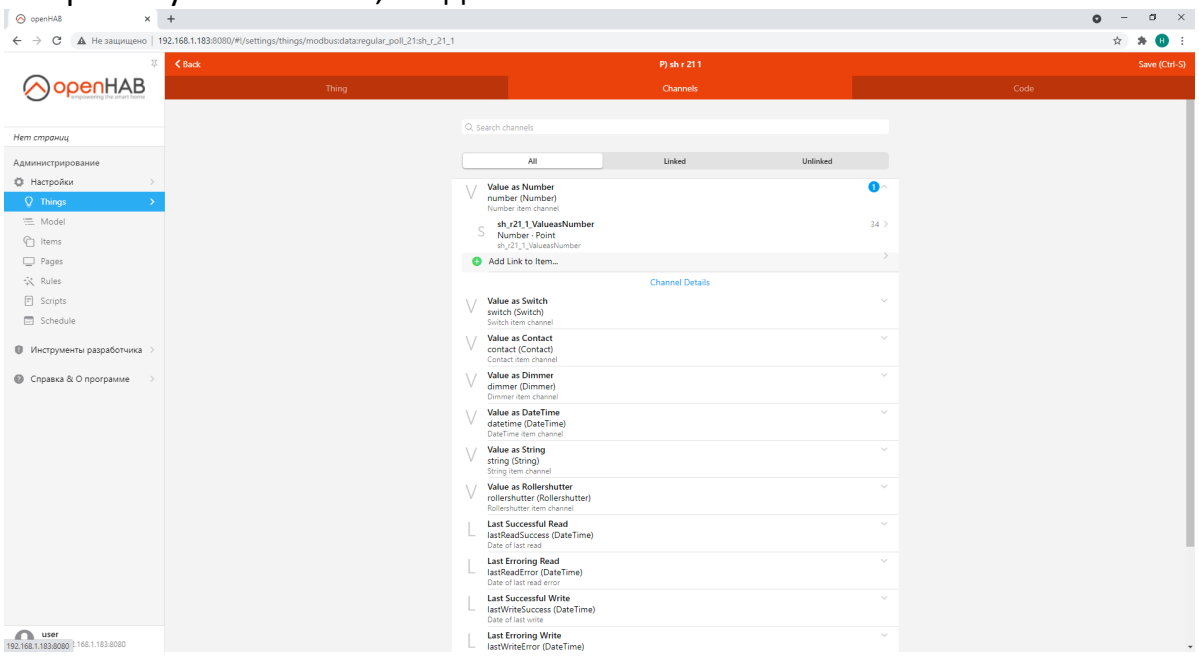
Создание Channels в Modbus

Ниже представлен пример создания Channels в Modbus.

1. Зайдите во вкладку Setting, далее переходим в Things.
2. Нажмите на уже созданный Modbus Things.
3. Нажмите на вкладку **Channels**.



4. По умолчанию уже присутствуют настроенные канал.
5. Выберите нужный канал, создайте Item.



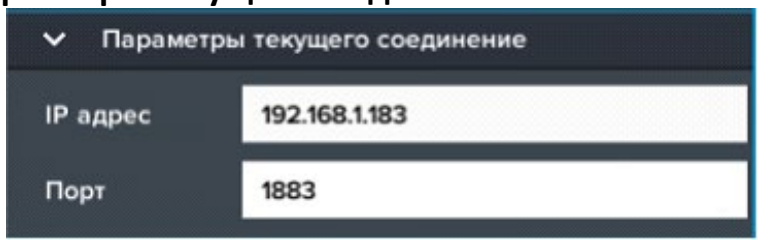
Для самостоятельного создания новых устройств ознакомьтесь с руководством на сайте: https://openhab.org/docs/tutorial/

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Пример автоматизации освещения при помощи OpenHAB.

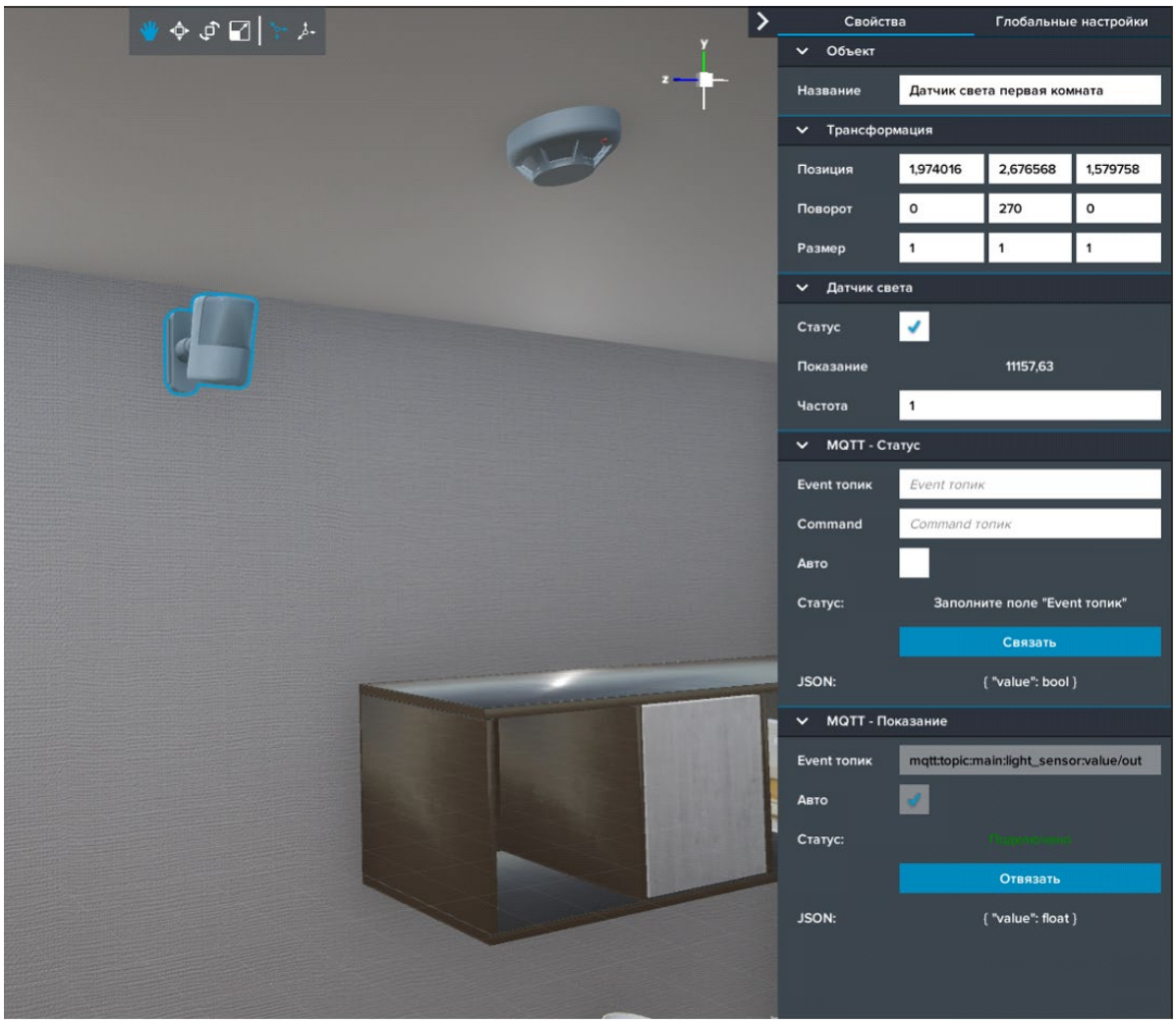
1.Создайте новый проект и разместите в нем окружение моделируемого пространства.

2.Установите IP адрес и порт MQTT брокера в панели глобальных настроек в компоненте **Параметры текущего соединения.**



Параметры текущего соединения

3.Разместите датчик света, установите статус датчика ВКЛ и пропишите Event топик в компоненте **MQTT – Показание:** mqtt:topic:main:light_sensor:value/out.

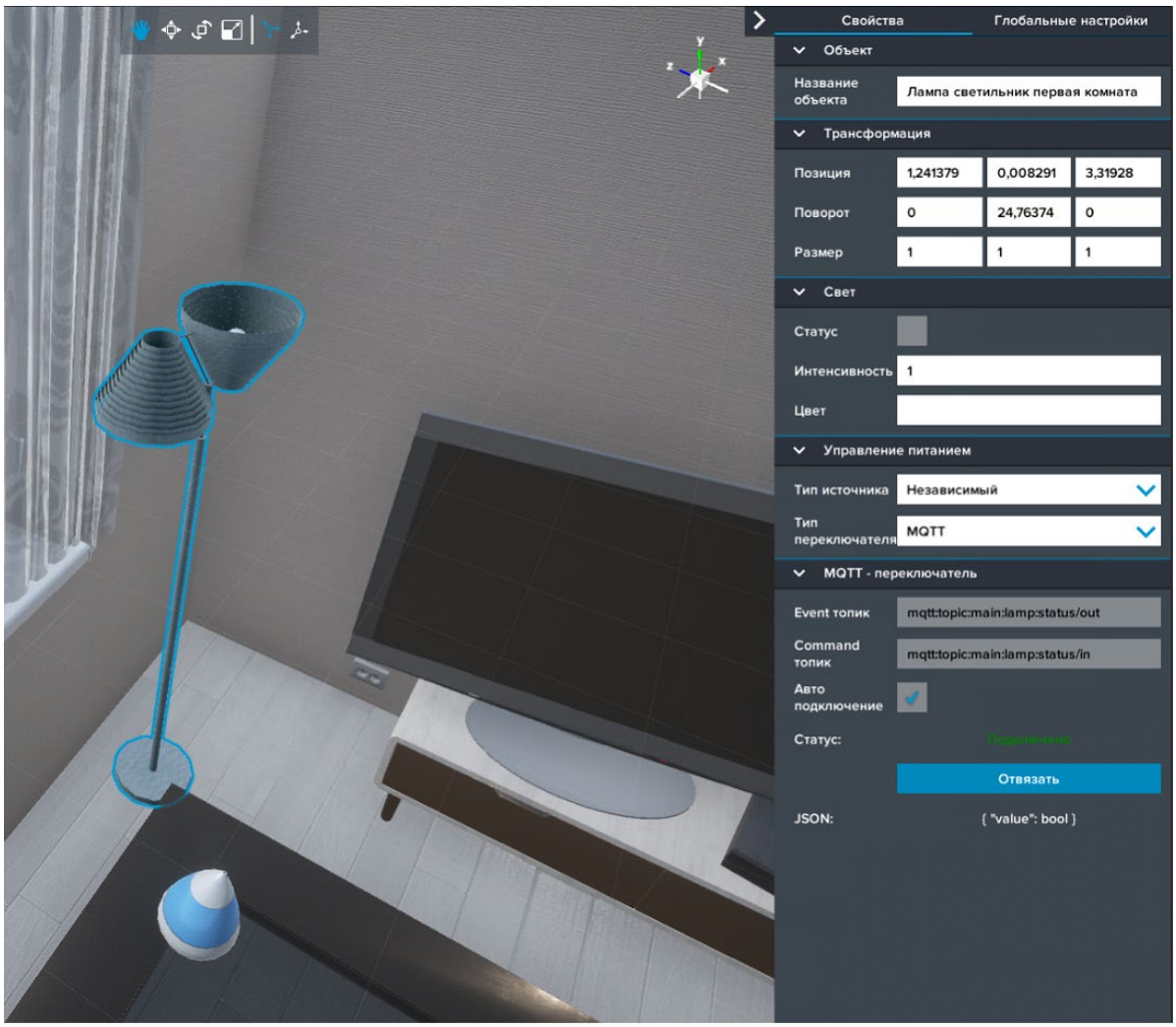


Добавление датчика света

4.Нажмите на кнопку **Связать** и дождитесь установления статуса **Подключено.**

5.Добавьте в проект объект освещения.

6.В компоненте **Управление питанием** установите тип переключателя на MQTT.



Добавление объекта освещения

7.В появившемся компоненте MQTT – Переключатель пропишите следующие топики:

a.Event топик - mqtt:topic:main:lamp:status/out,

b.Command топик - mqtt:topic:main:lamp:status/in

8.Нажмите на кнопку **Связать** и дождитесь установления статуса **Подключено.**

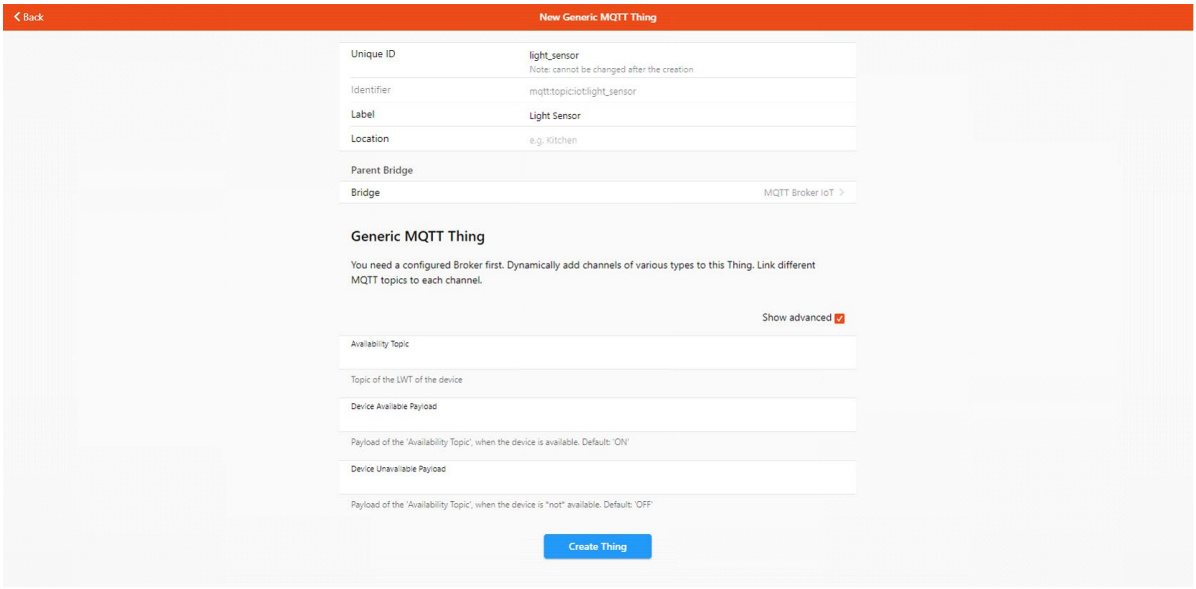
9.Откройте в браузере OpenHab.

10.В панели **Things** добавьте новую вещь **Generic MQTT Thing** (см. пункт Создание MQTT Bridge и Создание MQTT Things), которая будет связываться с виртуальных датчиком света.

11.Установите следующие параметры

a.Unique ID: light_sensor

b.Label: Light Sensor



Параметры датчика света

12.После создания датчика света, перейдите во вкладку **Channels** (см. пункт Создание Channels в MQTT) и добавьте новый канал со следующими параметрами:

a.Channel Identifier: light_sensor_value

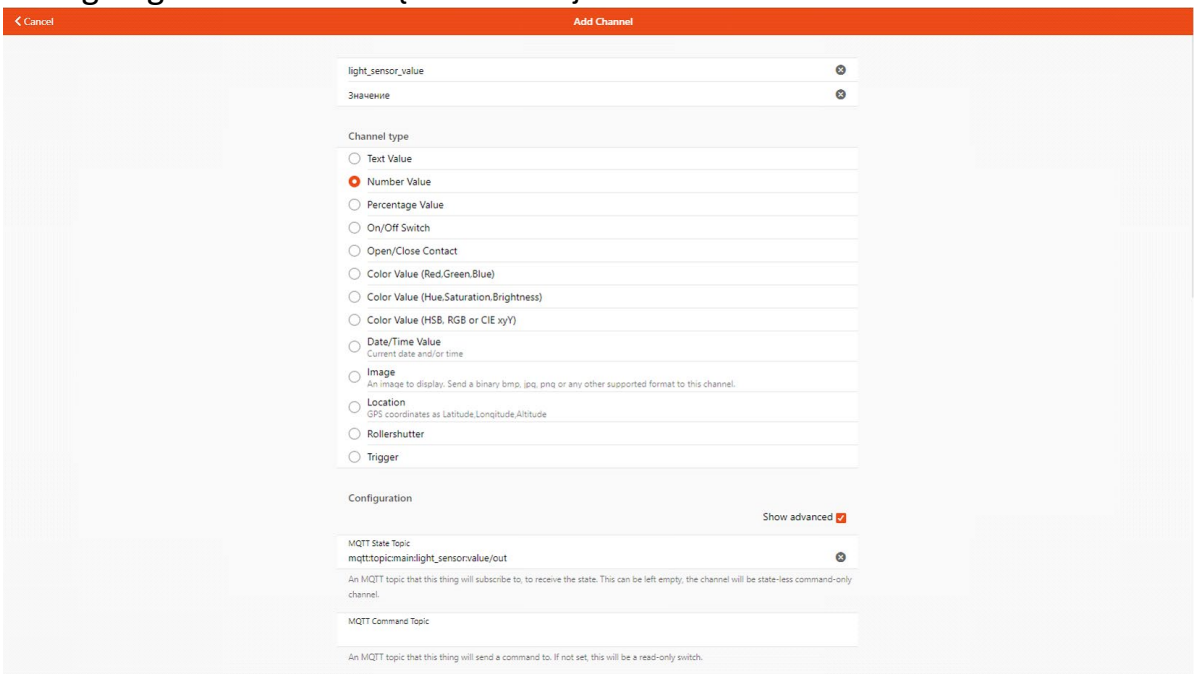
b.Label: Значение

c.Channel type: Number Value

d.MQTT State Topic: mqtt:topic:main:light_sensor:value/out

e.Incoming Value Transformations: JSONPATH:$.value

f.Outgoing Value Format: {"value":%s}



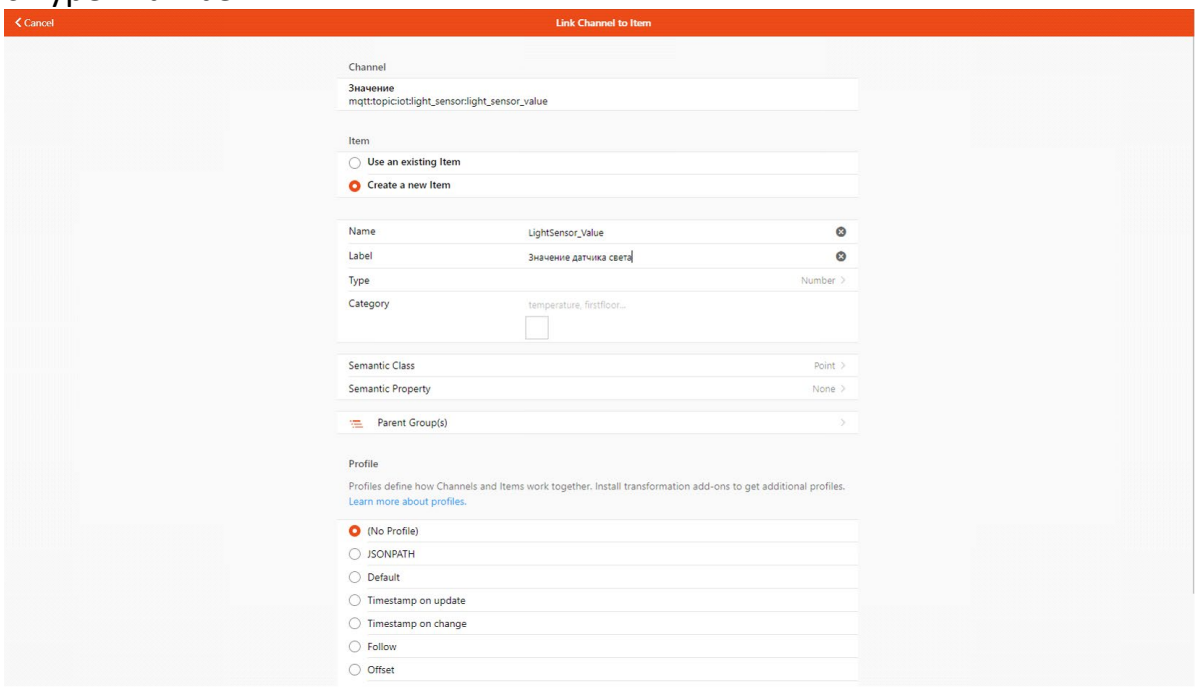
Добавление нового канала

13.Далее добавьте новый Item к созданному каналу со следующими параметрами:

a.Name: LightSensor_Value

b.Label: Значение датчика света

c.Type: Number



Добавление Item

14.Убедитесь, что значение от виртуального датчика передается в Item.

15.Далее необходимо создать новую вещь, которая будет связываться с виртуальным переключателем освещения, со следующими параметрами:

a.Unique ID: lamp

b.Label: Lamp

16.Добавьте в вещь новый канал со следующими параметрами:

a.Channel Identifier: lamp_status

b.Label: Статус

c.Channel type: On/Off Switch

d.MQTT State Topic: mqtt:topic:main:lamp:status/out

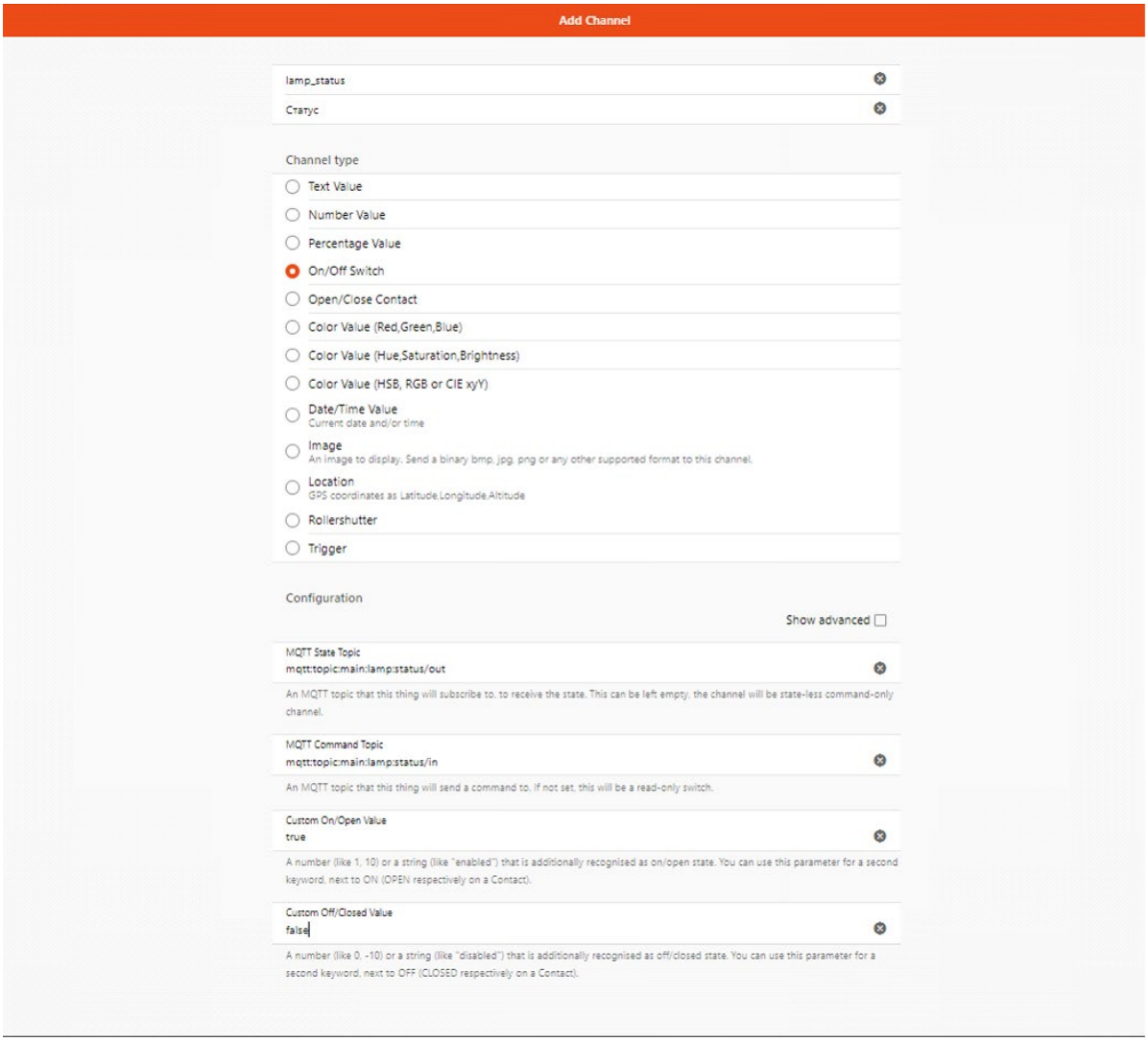
e.MQTT Command Topic: mqtt:topic:main:lamp:status/in

f.Custom On/Open Value: true

g.Custom Off/Closed Value: false

h.Incoming Value Transformations: JSONPATH:$.value

i.Outgoing Value Format: {"value":%s}



Добавление новой вещи

17.Далее добавьте новый Item к созданному каналу со следующими параметрами:

a.Name: Lamp_Status

b.Label: Статус переключателя

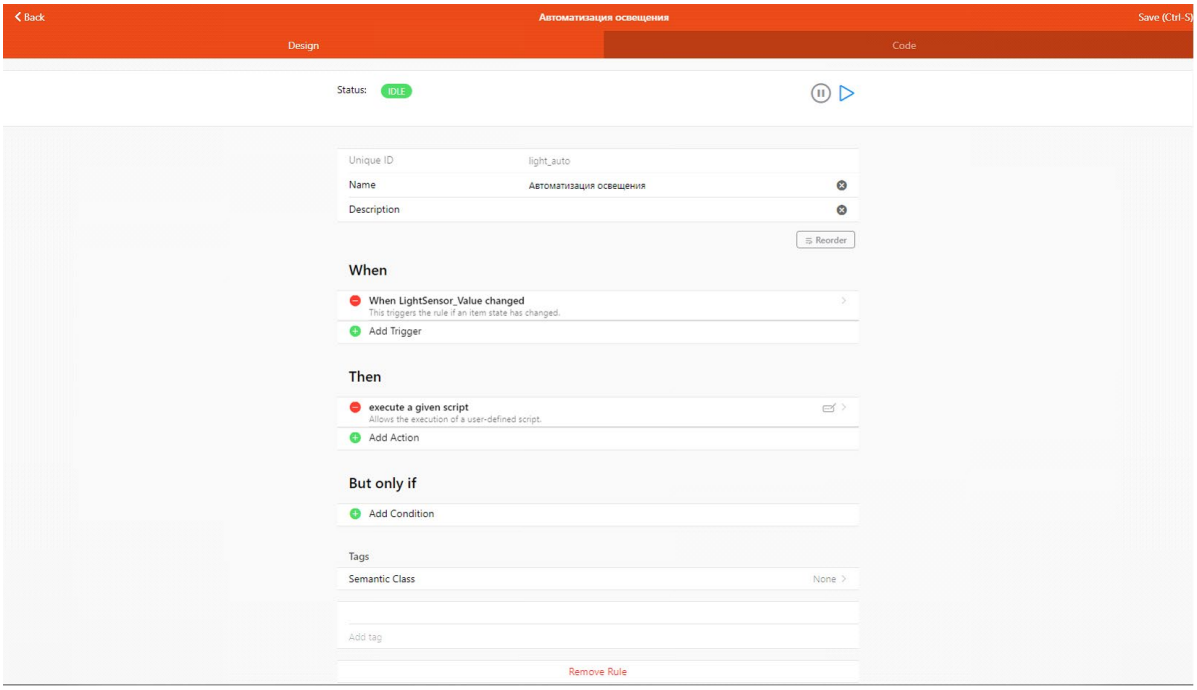
c.Type: Switch

18.Убедитесь, что статус от виртуального освещения передается в Item.

19.Далее необходимо создать новое правило во панели **Rule**, со следующими параметрами:

a.Unique ID: light_auto

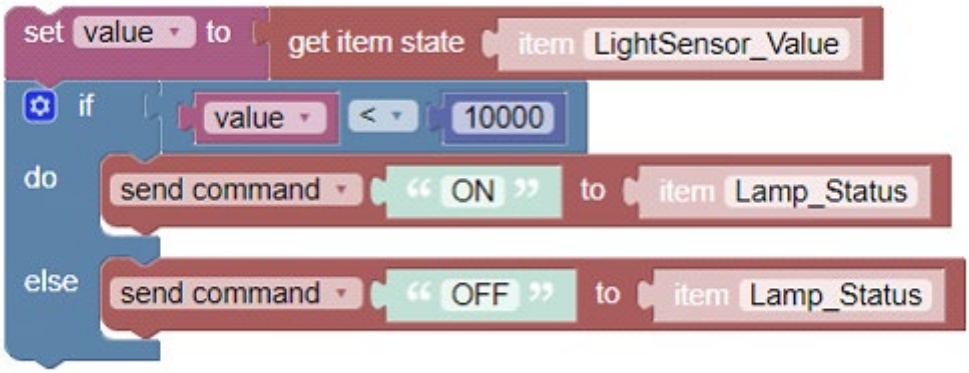
b.Label: Автоматизация освещения



Добавление Item

20.Установите срабатывание правила на триггер. Для этого в пункте **When** добавьте триггер **Item Event**, выберите **Значение датчика света,** установите пункт срабатывания **changed.**

21.В пункт **Then** добавьте действие **Run Script** и выберите **Design with Blockly** и собирите следующий скрипт:



22.После сохранения правила, перейдите в виртуальный проект и начните изменять значение глобального освещения. При значении показания датчика света меньше 10000 освещение должно включаться. При значении больше 10000 выключаться.

Показания датчика света

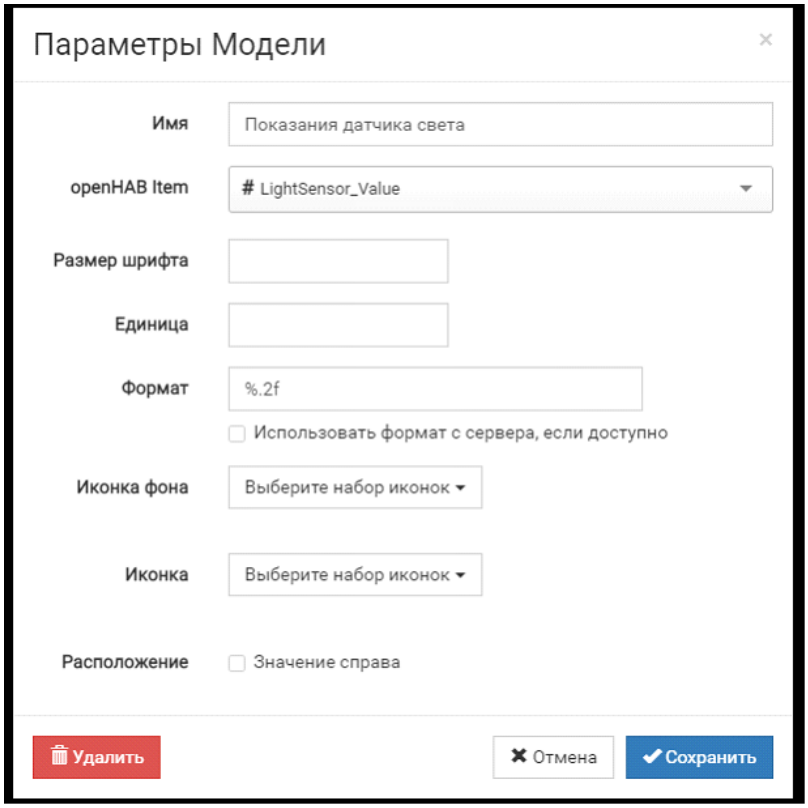
23.Далее перейдите в HABPanel (см. пункт OpenHab) и настройте дашборд.

24.Добавьте виджет **Модель** для вывода значения датчика света. В настройках установите следующие параметры:

a.Имя: Показания датчика света

b.openHAB Item: LightSensor_Value

c.Формат: %.2f

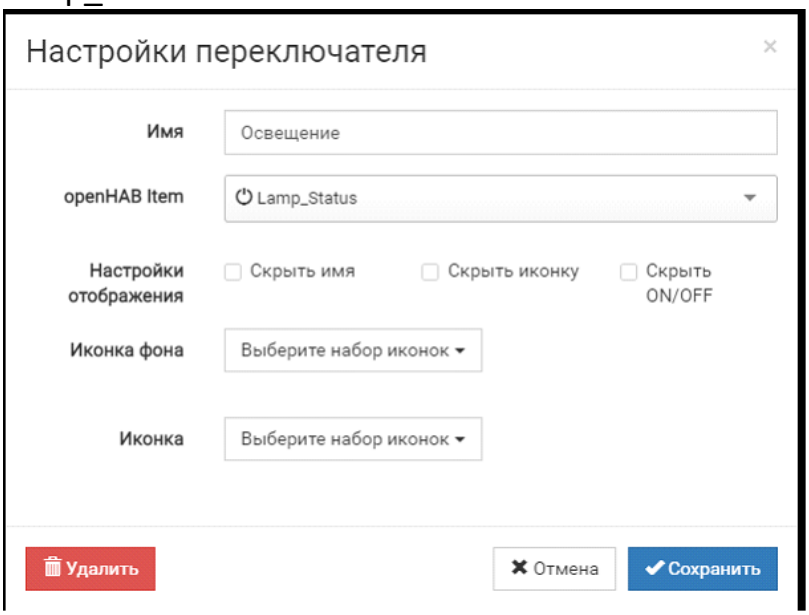


Виджет Модель

25.Добавьте виджет **Переключатель** для управления освещением со следующими параметрами:

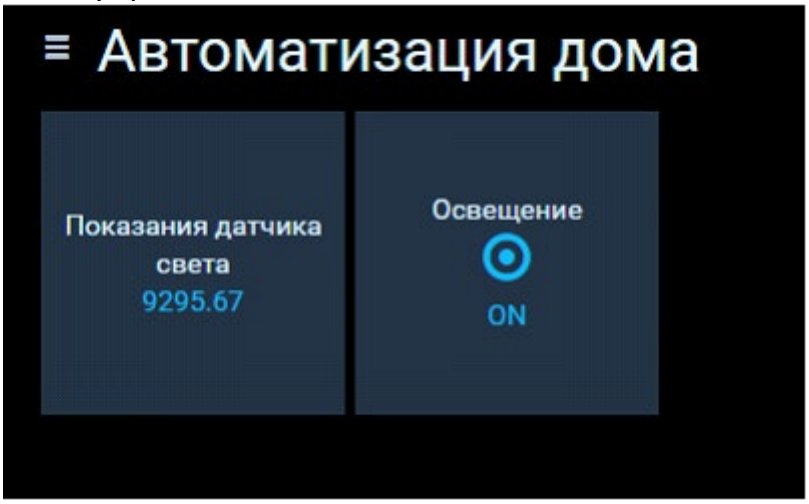
a.Имя: Освещение

b.openHAB: Lamp_Status



Виджет Переключатель

26.Убедитесь, что на дашборд выводится верное показание виртуального датчика света и идёт управление освещением.



Дашборд

PROGRAMLAB

121205, г. Москва, Территория Сколково инновационного центра,
Большой бульвар, дом 42, строение 1, помещение 13, этаж 2
Тел. 8-800-550-89-72 E-mail: info@pl-llc.ru

PL-LLC.RU