Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

ТЕХНОЛОГИИ ИНТЕРНЕТА ВЕЩЕЙ

Учебно-методическое пособие
по практическим занятиям и самостоятельной работе

Томск 2024

УДК 621.396
БКК 32.973я7
А23

Рецензент:
Рогожников Е.В., заведующий кафедрой ТОР ТУСУР, к.т.н., доцент

А23 Технологии интернета вещей: учеб.-метод. пособие по практическим занятиям и самостоятельной работе / Е.Ю. Агеев, А.В. Бусыгина – Томск : Томск. гос. ун-т систем упр. и радиоэлектроники, 2024. – 92 с.

Учебно-методическое пособие содержит указания для выполнения практических и самостоятельных работ. Цель данного пособия заключается в приобретении студентами навыков работы с инструментами, позволяющими проектировать программно-аппаратные комплексы Интернета вещей, а также формирование компетенций в области существующих IoT-технологий и умений применять их к конкретным сценариям. Практикум направлен на развитие практических навыков студентов в применении полученных теоретических знаний для решения задач, связанных с построением, проектированием, функционированием и использованием устройств Интернета вещей. Предназначено для студентов технических специальностей, также пособие может быть полезно студентам смежных специальностей и ИТ.

Одобрено на заседании ПИШ, протокол № 2 от 21.10.2023

УДК 621.396
БКК 32.973я7

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель данного практикума заключается в приобретении студентами навыков работы с инструментами, позволяющими проектировать программно-аппаратные комплексы Интернета вещей, а также формирование компетенций в области существующих IoT-технологий и умений применять их к конкретным сценариям. Практикум направлен на развитие практических навыков студентов в применении полученных теоретических знаний для решения задач, связанных с построением, проектированием, функционированием и использованием устройств Интернета вещей.

Практикум посвящен изучению следующих тем:

- Знакомство с технологией KNX, настройка виртуального стенда.
- Быстрое прототипирование решения интернета вещей в среде NODE-RED.
- Обработка данных сенсоров, полученных из базы данных MongoDB.
- Знакомство с облачным сервисом интернета вещей THINGSBOARD.
- Моделирование сети взаимодействующих устройств в среде Contiki Cooja.
- Создание беспроводной сети взаимодействующих устройств WSN, маршрутизация RPL, протоколы 6LoWPAN и CoAP.

1. Практическая работа №1
ЗНАКОМСТВО С ТЕХНОЛОГИЕЙ KNX, НАСТРОЙКА ВИРТУАЛЬНОГО СТЕНДА

Работа выполняется в программном приложении Engineering Tool Software – ETS version 6 с использованием виртуального стенда. Виртуальный стенд KNX Virtual — это отдельное приложение для Windows, имитирующее управляемые по протоколу KNX умные устройства. На рис. 1 показан внешний вид окна программы-стенда в режиме «Базовые функции. Одна комната».

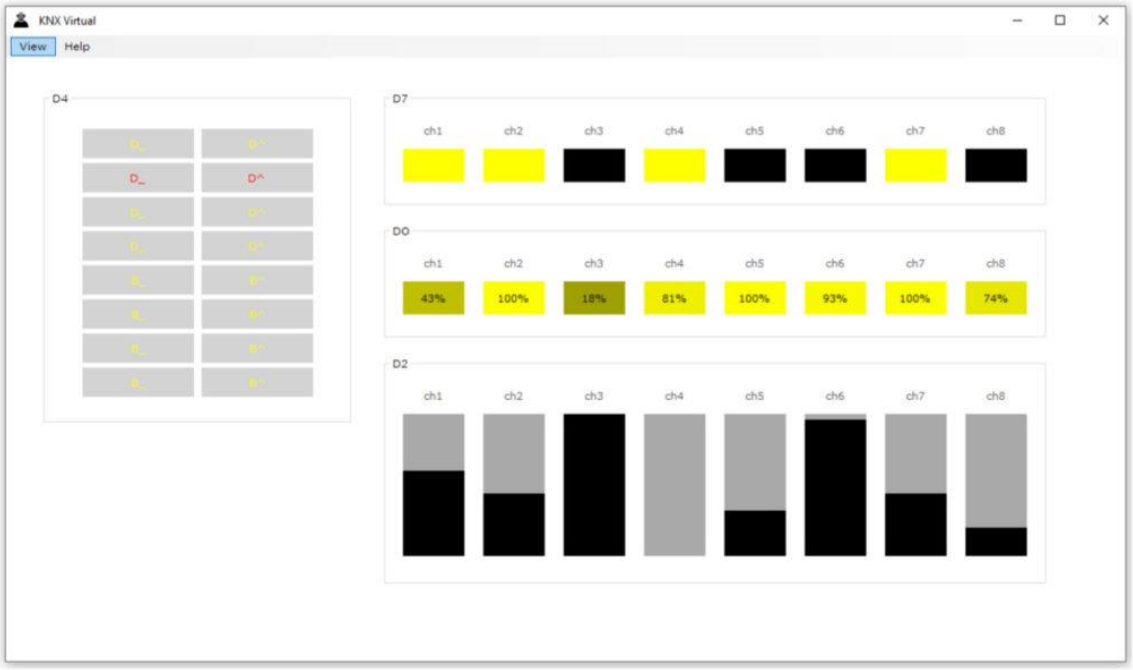

Рис. 1 Внешний вид главного окна программы

Сценариев использования стенда несколько, как показано на рис. 2. Базовые функции для нескольких комнат, продвинутый сценарий использования выключателей, управление жалюзи, управление кондиционером и т.д.

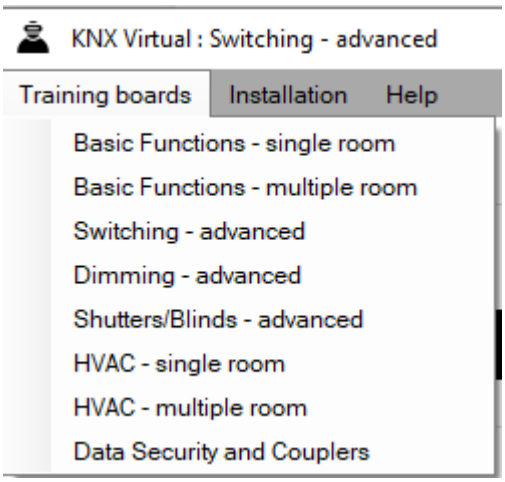

Рис. 2 Сценарии виртуального стенда

Блок D4 в базовом сценарии рис. 1 представляет набор программируемых кнопок. Реально такое устройство может выглядеть, как показано на рис. 3.

Рис. 3 Универсальный модуль управляющих кнопок

Блок D7 восьмиканальный программируемый выключатель. На панели виртуального стенда выведены «как-бы лампочки», управляемые этим выключателем. По изменению состояния этих лампочек можно судить о работе выключателя. В реальности такой переключатель может иметь вид как на рис. 4.

Рис. 4 Восьмиканальный программируемый выключатель KNX

Блок D0 восьмиканальный диммер, выключатель с возможностью регулирования яркости светильника. Устройство этого типа в реальной жизни показано на рис. 5.

Рис. 5 Четырехканальный диммер KNX

Последнее из устройств базового сценария — устройство D2 управления шторами/жалюзи. Пример подобного устройства показан на рис. 6.

Рис. 6 Программируемый многоканальный контроллер управления жалюзи.

Всего стенд KNX Virtual содержит более 20 устройств KNX различных типов, подключенных к одной информационной линии TP. Эти устройства KNX позволяют практиковаться с настройкой оборудования стандарта KNX без самого оборудования. В том числе можно протестировать такие функции как погодные модули, сигналы тревоги, сцены и даже логические операции. На рис. 7 приведена таблица с перечнем поддерживаемых устройств. Устройство D20 — реализует туннельный интерфейс между виртуальным стендом и программой настройки, оно задействовано сразу при запуске программы виртуального стенда.

Segment	ID	OrderNr	AES	Description
S1	D25	BO.ip	No	Binary Output
	D19^			Undefined: block/block
S2	D19_	C.ip.tp	No	IP/TP Coupler
	D0	DA.tp	-	Dimming Actuator
	D2	BS.tp	-	Blinds/Shutter Actuator
	D4	KX.tp	-	KliX
	D7	SA.tp	-	Switching Actuator
	D13	SC.tp	-	Scenario Controller
	D14	LM.tp	-	Logic Module
	D20	I.ip.tp	-	IP/TP Interface
	D5^			Undefined: block/block
S3	D5_	C.tp.tp	No	TP/TP Coupler
	D1	PB.tp	No	Push Button Interface
	D3	BO.tp	No	Binary Output
	D6	VA.tp	-	Valve Actuator
	D9	AM.tp	-	Alarm Module
	D10	MP.tp	-	Movement/Presence Detector
	D11	BI.tp	-	Binary Input Module
	D12	WM.tp	-	Weather Module
	D15	SP.tp	-	Setpoint Manager
	D16	HC.tp	-	Heat Controller
	D17	HE.tp	-	Heat Exchanger
	D21	RC.tp	-	Room Controller
	D22	RC.tp	-	Room Controller
	D23	RC.tp	-	Room Controller
	D24	RC.tp	-	Room Controller
	D18^			Undefined: block/block
S4	D18_	C.tp.rf	No	TP/RF Coupler
	D26	PB.rf	No	Push Button Interface

Рис. 7 Перечень устройств KNX на виртуальном стенде

Все устройства на шине KNX имеют индивидуальные адреса, однако для отправки сообщений в качестве адреса назначения чаще используются не эти индивидуальные адреса, а специальные групповые адреса, объединяющие целевые функции, выполняемые устройствами. Схема адресации может быть двух- или трехуровневая. На практике почти всегда применяют трехуровневую схему. В этом случае индивидуальный адрес записывается в виде трех чисел, разделенных точками. Первое число в адресе указывает область или зону размещения устройства, второе линию, к которой подключено устройство, последнее — индивидуальный адрес устройства, рис. 8. Максимальное значение индивидуального адреса 15.15.255.

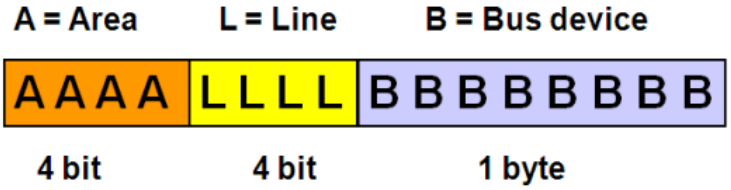

Рис. 8 Структура индивидуального адреса KNX

Небольшие системы обычно состоят из нескольких устройств, которые подключаются к одной зоне, состоящей из одной линии, рис. 9. Источник питания KNX может обеспечить энергией не более 64 устройств. Поэтому линия делится на сегменты, максимум четыре сегмента. В каждом из которых свой источник питания. Сегменты объединяются с помощью специальных устройств — линейных повторителей.

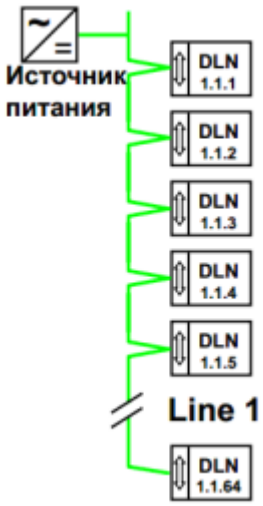

Рис. 9 Линия KNX с подключенным источником питания

Несколько линий могут быть соединены в общую зону, рис. 10. Это соединение реализуется линейными соединителями. Максимально можно соединить 15 линий, адрес 0 зарезервирован. На самом деле адрес зоны присутствует в индивидуальном адресе всегда, просто для линий в одной зоне он одинаков.

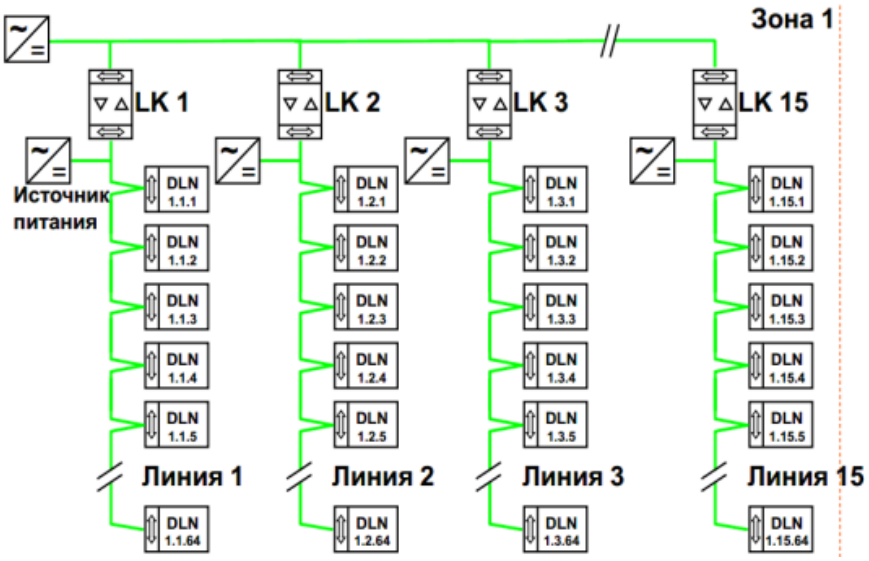

Рис. 10 Соединение нескольких линий в общую зону

В качестве зоны может выступать этаж здания, каждая комната которого будет являться линей, к которой присоединяются устройства.

С помощью системной линии можно объединить несколько зон, максимально 15, рис. 11. Максимально возможное число подключенных устройств KNX в одной системе более 58000. Нулевой адрес в поле адресации устройства зарезервирован для линейных соединителей, нулевой адрес в поле адресации линий — для магистральных соединителей.

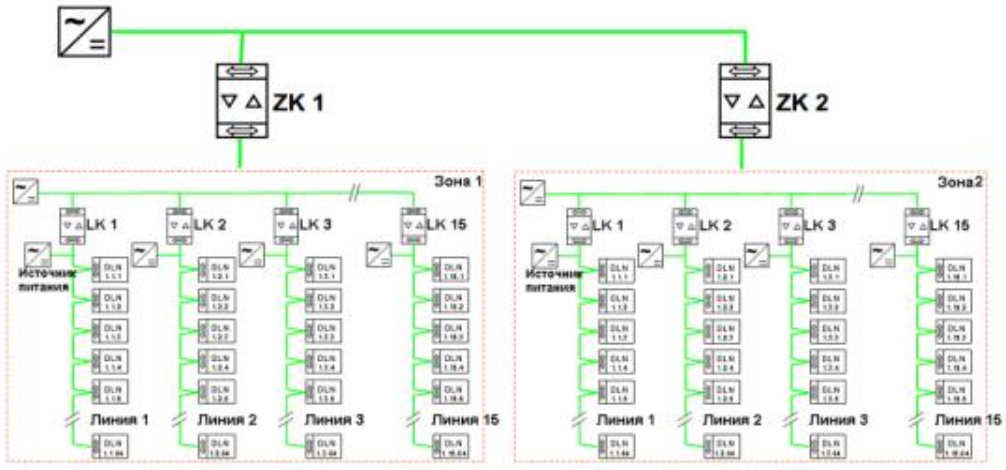

Рис. 11 Объединение зон KNX

Групповой адрес объединяет устройства по целевой функции, например, на рис. 12 два устройства — управляющая кнопка и переключатель с подключенной лампой имеют общий групповой адрес 5/2/66. Кнопка отправляет сигнал по этому адресу, а переключатель получает его. В одну группу могут быть включены множество устройств (могут иметь общий групповой адрес), но отправляет сигнал в группу всегда одно из устройств, остальные только получают его.

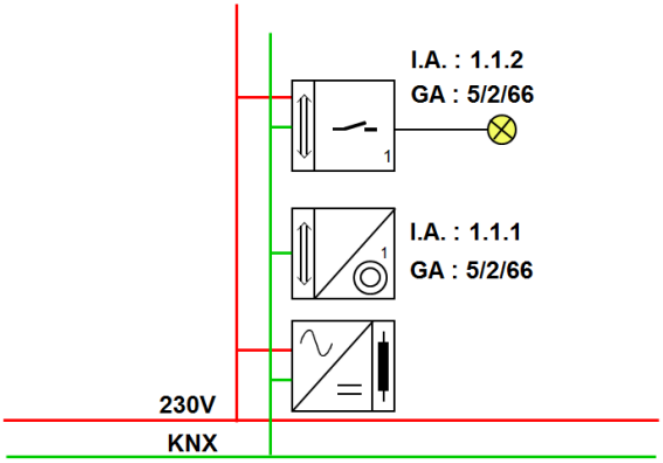

Рис. 12 Групповые и индивидуальные адреса

Групповой адрес так же, как индивидуальный, имеет длину 16 разрядов или два байта, но разделяется он по-другому. При выборе в проекте трехуровневой структуры, старшие пять разрядов адресуют главный блок иерархии (main), следующие три бита средний блок (middle), оставшиеся восемь представляют адрес группы нижнего слоя иерархии (sub). При выборе двухуровневой иерархии главный блок (main) адресуется так же пятью старшими битами, среднего уровня просто нет и далее идет сразу нижний уровень (sub) на долю которого выпадает одиннадцать разрядов. Есть еще вариант свободной структуры, в которой нет уровней вовсе, но это специальный случай, в широкой практике такая адресация применяется редко, рис. 13.

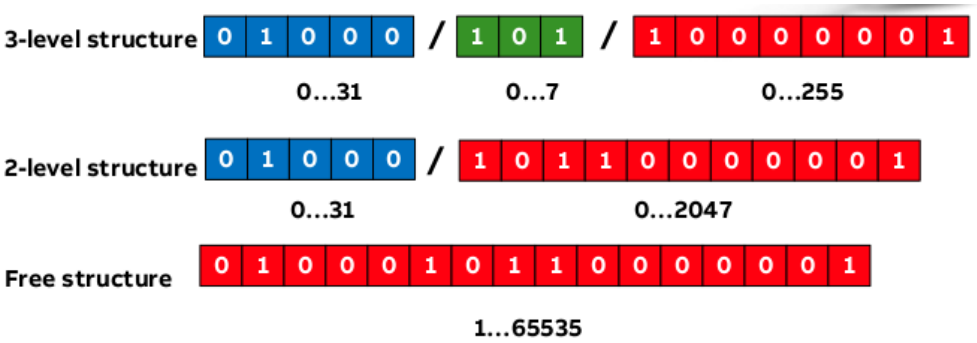

Рис. 13 Варианты группового адреса

ПРИМЕР НАСТРОЙКИ РЕГУЛИРУЕМОЙ ЦЕПИ ОСВЕЩЕНИЯ

Часть 1 – Проектирование инфраструктуры

Этап проектирования является наиболее важной частью проекта KNX. Он занимает примерно 80% времени, затраченного на проект.

Шаг 1. Создайте новый проект.

Самый первый шаг — создать новый проект. Для этого просто откройте ETS6 и выберите кнопку « + **Новый проект** » на панели инструментов, рис. 13. В демонстрационном режиме программа позволяет полноценно работать с проектами, но ограничивает разрешенное число использованных устройств — не более пяти. Имейте в виду что одно из устройств виртуального стенда — D20, включено автоматически.

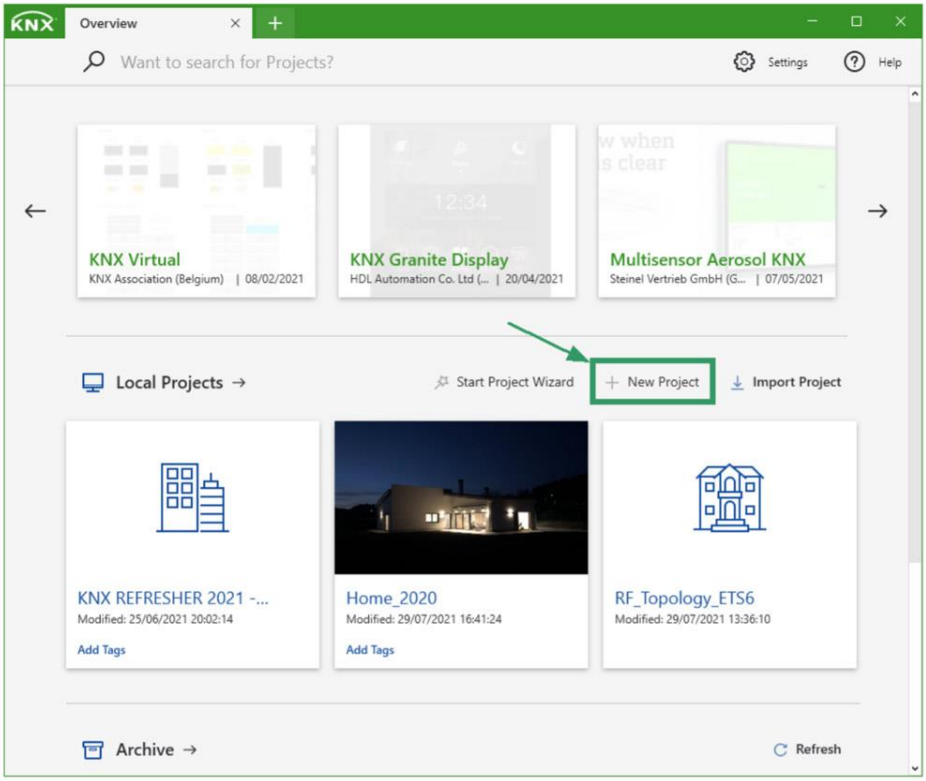

Рис. 13 Создание нового проекта.

Затем вы можете дать проекту имя. Остальные параметры оставьте по умолчанию, рис. 14. Здесь магистральная линия задана как IP, это необходимо, потому что между создаваемой топологией на основе витой пары (TP) и виртуальным стендом через интерфейс

D20 формируется туннельное соединение типа IP. Создается линия с адресом зоны 1 и адресом линии 1, и выбирается трехуровневая схема групповой адресации.

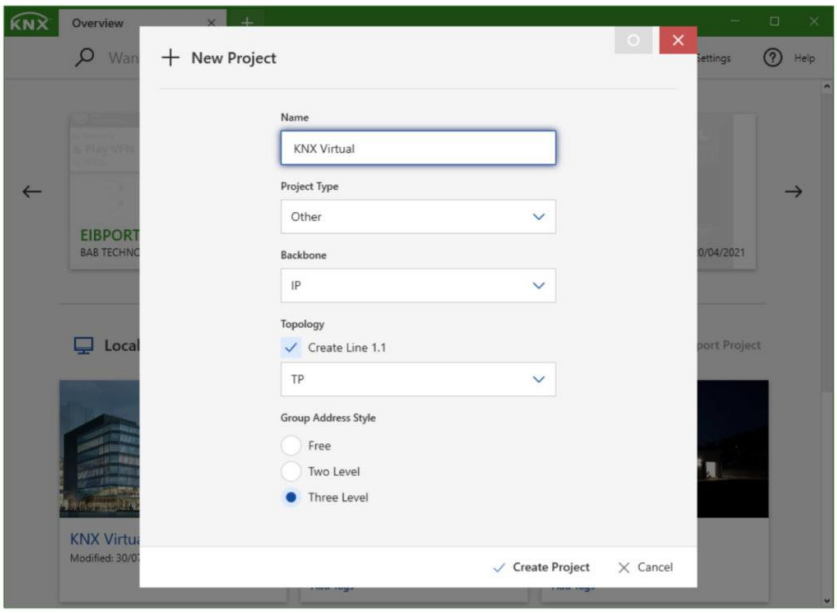

Рис. 14 Задание имени новому проекту

Шаг 2. Создайте структуру здания в представлении «Здания».

После создания проекта откроется рабочая область с окном зданий. Чтобы упорядоченно расположить устройства, необходимо создать здание, в котором они будут размещаться. В здание добавьте одну комнату для размещения управляющих элементов (кнопок). В комнате добавьте электрический шкаф для размещения элементов на DIN-рейке, рис. 15. Здесь DIN это немецкий институт стандартизации. DIN-рейка — обобщённое название металлического профиля, применяемого в электротехнике, рис. 16.

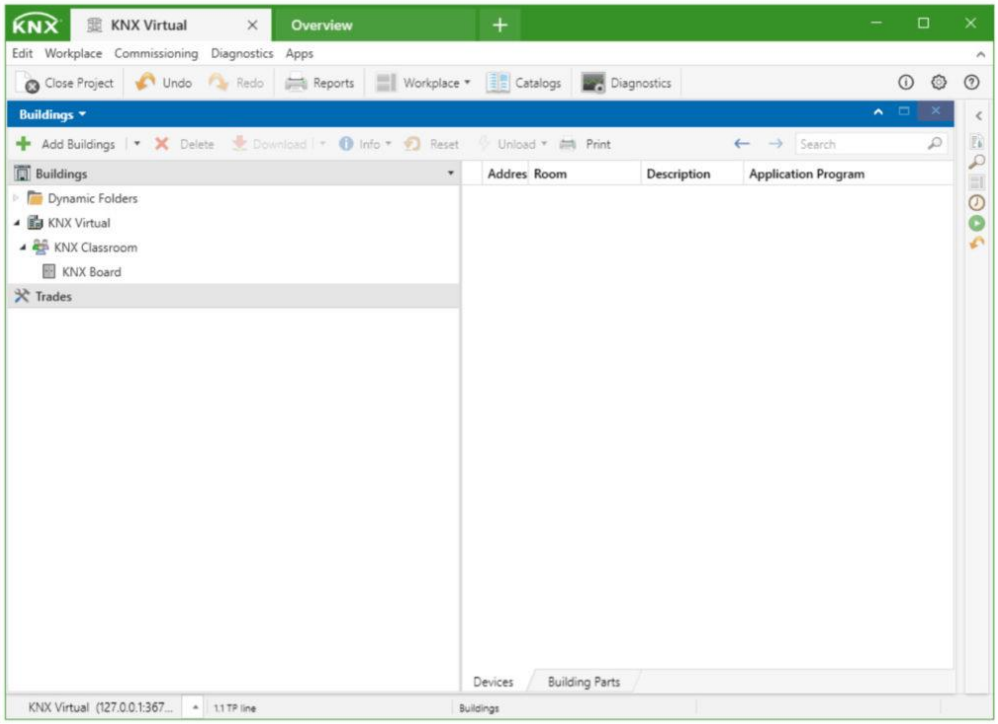

Рис. 15 Создание структуры проекта

Рис. 16 Автоматы закрепленные на DIN-рейке

Шаг 3. Добавьте виртуальные устройства KNX.

Далее добавьте в проект умные виртуальные устройства. После выбора опции «Добавить устройства» появится диалог «Каталог продукции», рис. 17.

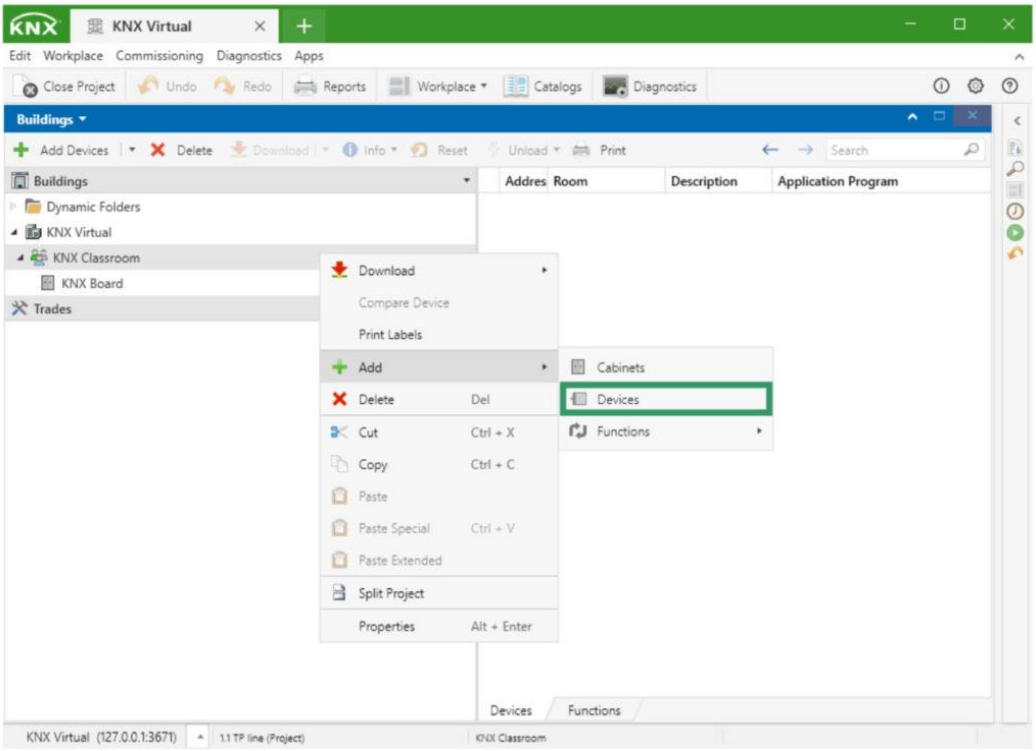

Рис. 17 Добавление устройств

Надо найти производителя « **KNX Association** » и выбрать устройства «KLiX (D4)» это набор кнопок и «Диммерный актуатор (D0)», рис. 18. Можно ориентироваться на индексы D4 и D0.

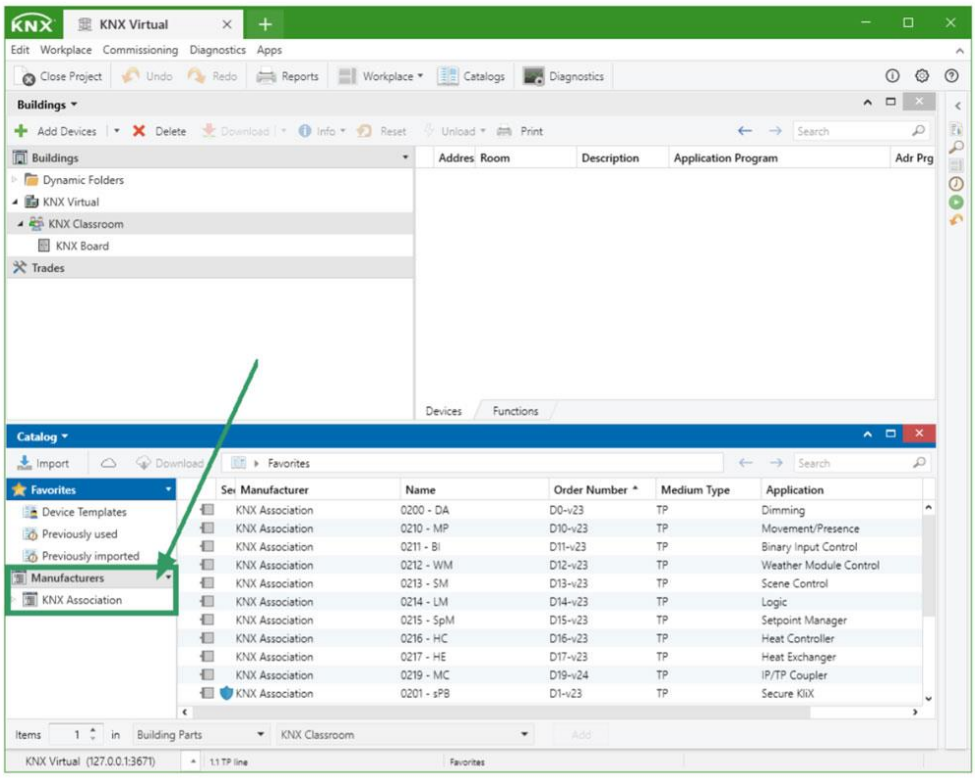

Рис. 18. Устройства KNX в каталоге

Шаг 4 – Настройка параметров

Одним из важнейших этапов является правильная настройка параметров устройства. На изображении рис. 19 вы можете увидеть конфигурацию, выбранную для первого канала кнопки и первого канала регулятора яркости. Настройте такие же параметры.

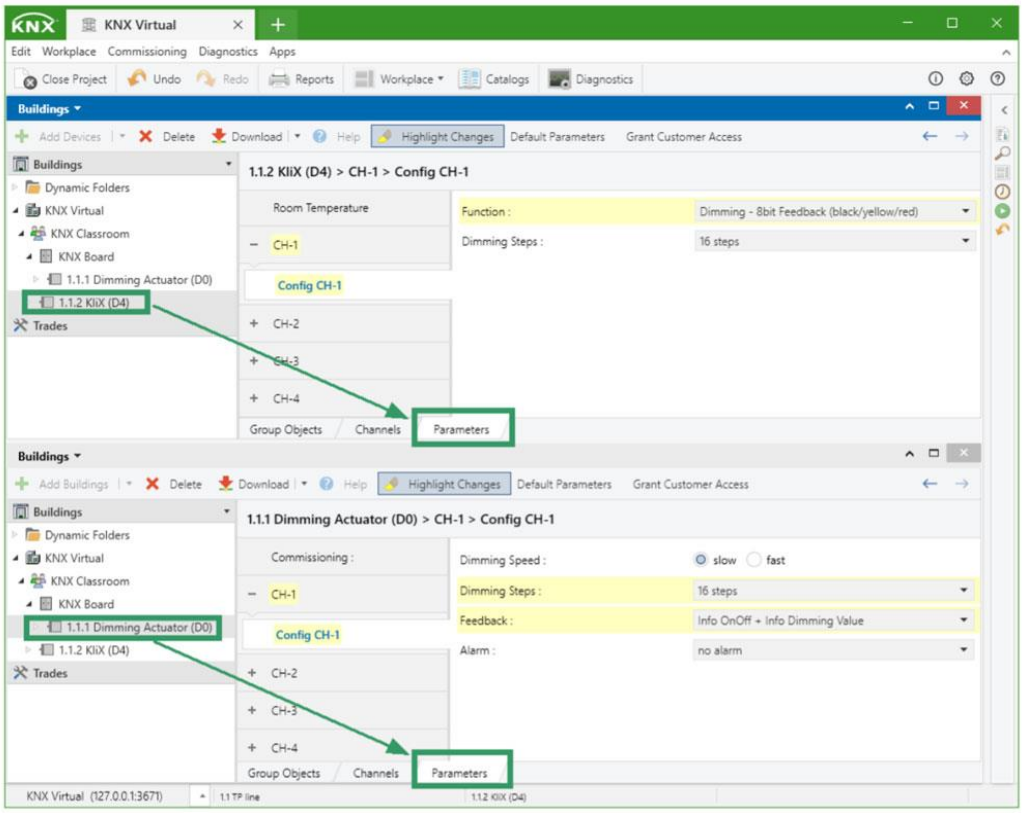

Рис. 19 Настройка параметров устройств

Шаг 5 – Создайте функцию

В «классическом» проекте KNX следующим шагом будет создание групповых адресов и соответствующая ассоциация с групповыми объектами каждого устройства. Например, как показано на рис. 20. Мы можем выбрать операцию включения/выключения на кнопке D4, нажать клавишу Ctrl и выделить операцию включения/выключения на управляемом переключателе D7. Таким образом, у нас будет одновременно выделено две позиции. После этого можно вызвать контекстное меню нажатием правой клавиши мышки и в нем выбрать строку «Связать с ...». Откроется диалоговое окно создания/назначения группового адреса.

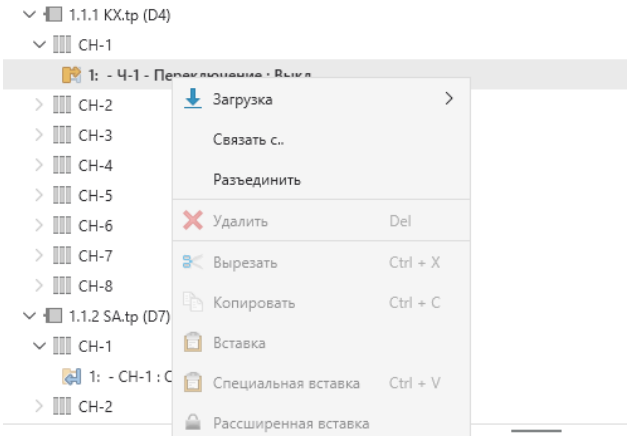

Рис. 20. Связывание двух объектов с назначением группового адреса

Однако, можно объединить эти задачи в один шаг и значительно ускорить процедуру. Для этого создайте **функцию** на том же виде здания, в данном случае для управления регулируемым освещением, рис. 21. Созданная функция автоматически добавит групповые адреса для выбранных устройств, необходимые для связи. Создаст шаблон с адресами.

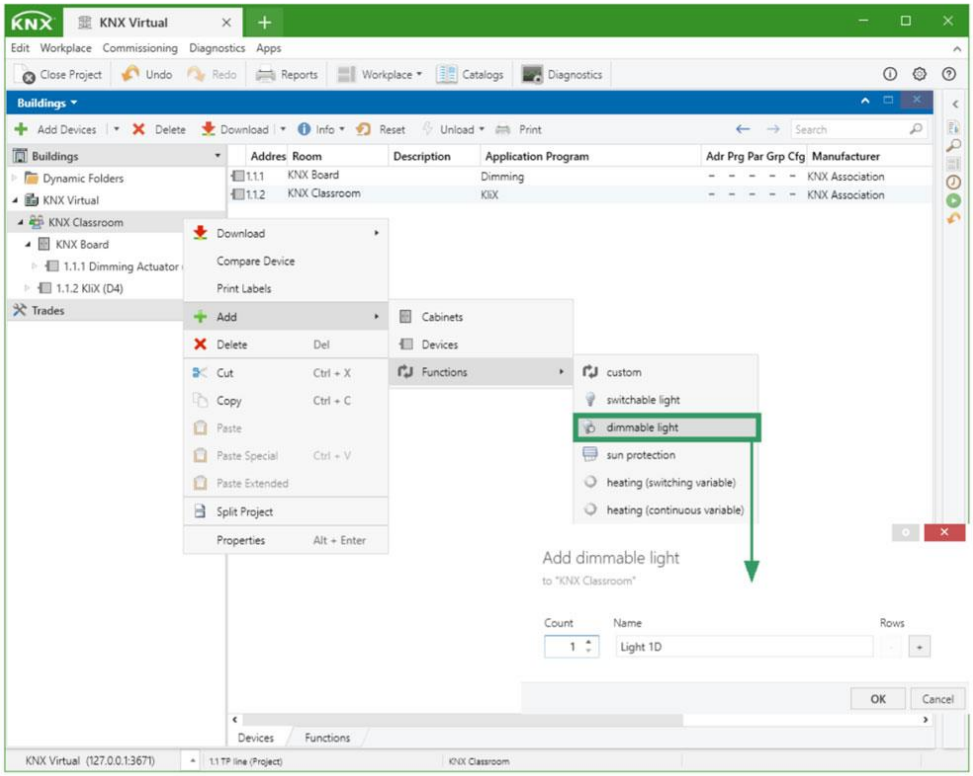

Рис. 21 Создание целевой функции

Шаг 6 – Свяжите функцию с соответствующими каналами

После создания функции нужно связать ее с первым каналом функциональной кнопки и первым каналом диммера. Это можно сделать путем перетаскивания соответствующих каналов устройств на целевую функцию в окне программы, рис. 22.

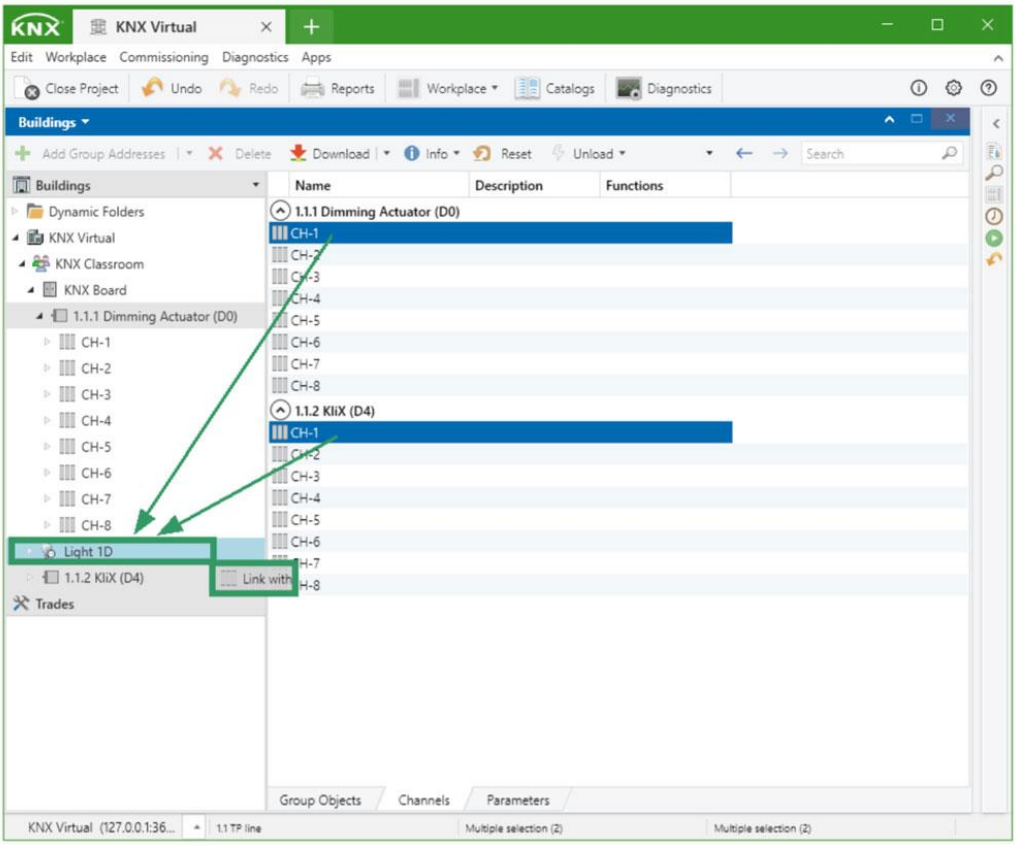

Рис. 22 Связывание функции с устройствами

В следующем диалоговом окне остается только проверить правильность связи между адресами и групповыми объектами, рис. 23. Возможно, соответствия будут установлены не все и тогда их нужно задать вручную.

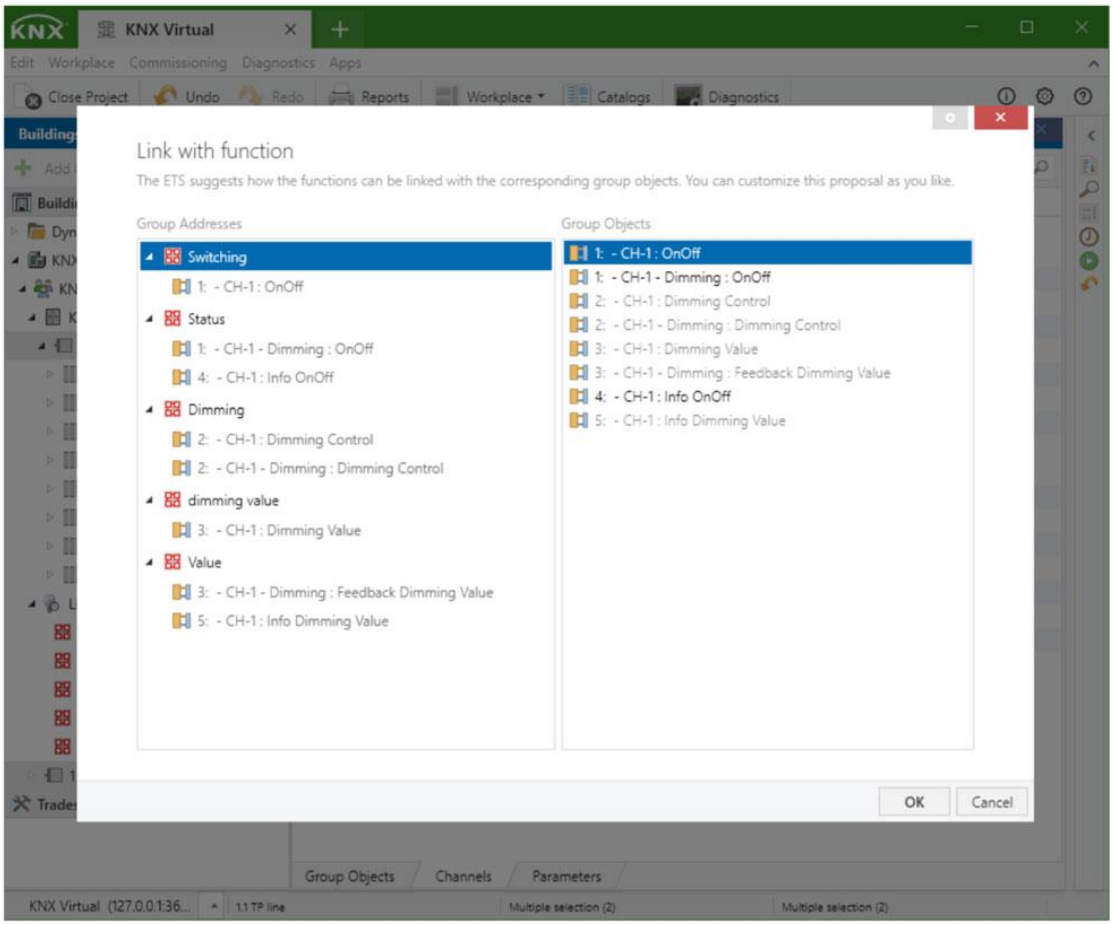

Рис. 23 Проверка связывания

Наконец, мы должны проверить в представлении здания правильность ассоциаций, установленных между групповыми адресами и групповыми объектами, рис. 24. Используйте фильтрацию по названию канала, чтобы сократить объем информации, отображаемой на экране. Убедившись в правильности ассоциаций, завершаем первую часть, посвященную созданию структуры проекта.

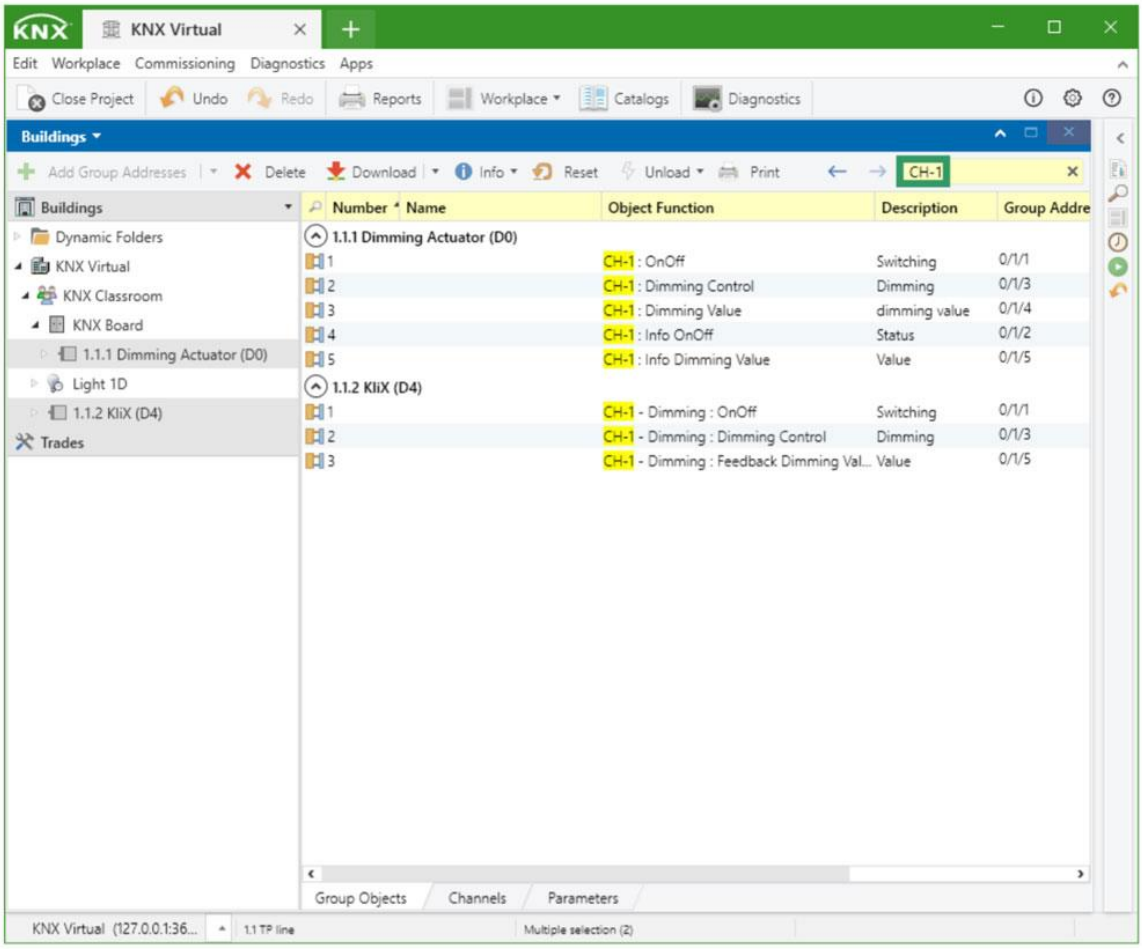

Рис. 24 проверка правильности ассоциаций

Часть 2 – Ввод устройства в эксплуатацию

После завершения проектирования проекта мы должны провести пуско-наладку устройств, т. е. загрузку на реальные устройства настроенных нами сценариев автоматизации и проверку того, что они корректно работают на реальных устройствах, выполняя все заданные алгоритмы. Процедура полностью идентична той, которую мы бы провели в реальном проекте на реальном оборудовании, но подключение к шине в этом случае будет осуществляться через **KNX Virtual** где установлено виртуальное оборудование.

Шаг 1 – Откройте виртуальный стенд KNX

Запустите приложение виртуального стенда KNX в базовой конфигурации, одна комната. Определите через меню «ETS» IP-порт какой используется порт IP-интерфейса (D20). Порт по умолчанию — 3671, но при необходимости можно выбрать другой, рис. 25.

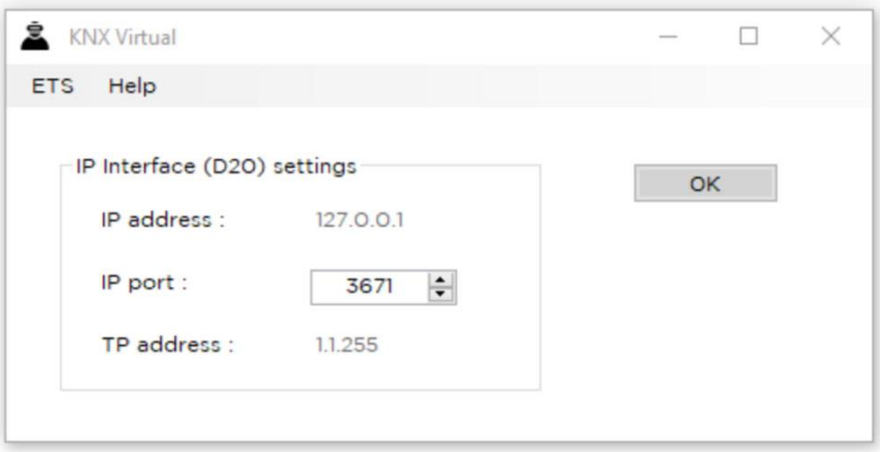

Рис. 25 Порт и адрес виртуального стенда

После нажатия кнопки «ОК» в KNX Virtual должен отобразиться вид «Устройства/Топология».

Шаг 2. Выберите виртуальный интерфейс KNX.

После запуска виртуального стенда соответствующий интерфейс для подключения к нему можно выбрать в диалоговом окне «Настройки — Диспетчер подключений» программы ETS6, рис. 26.

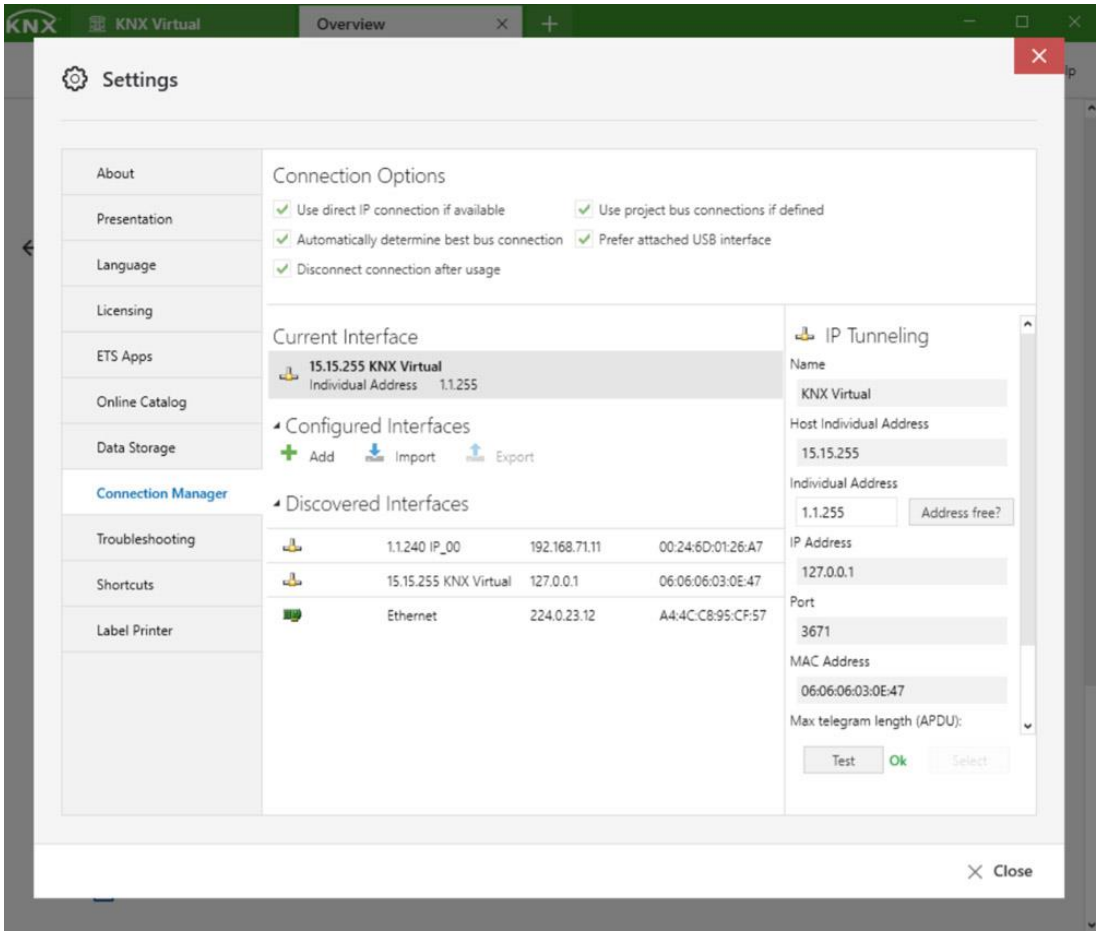

Рис. 26 Подключение ETS к виртуальному стенду

Шаг 3. Загрузите физический адрес и прикладную программу.

Настроенные сценарии автоматизации загружаются по шине KNX на соответствующие устройства, так они «программируются». Выберите первое устройство для программирования в представлении «Здания» и выберите опцию «Загрузить все». Чтобы запрограммировать индивидуальный адрес каждого устройства, нам необходимо нажать красную иконку в правом верхнем углу соответствующего устройства. Этот значок представляет собой кнопку программирования + светодиод устройства, рис. 27.

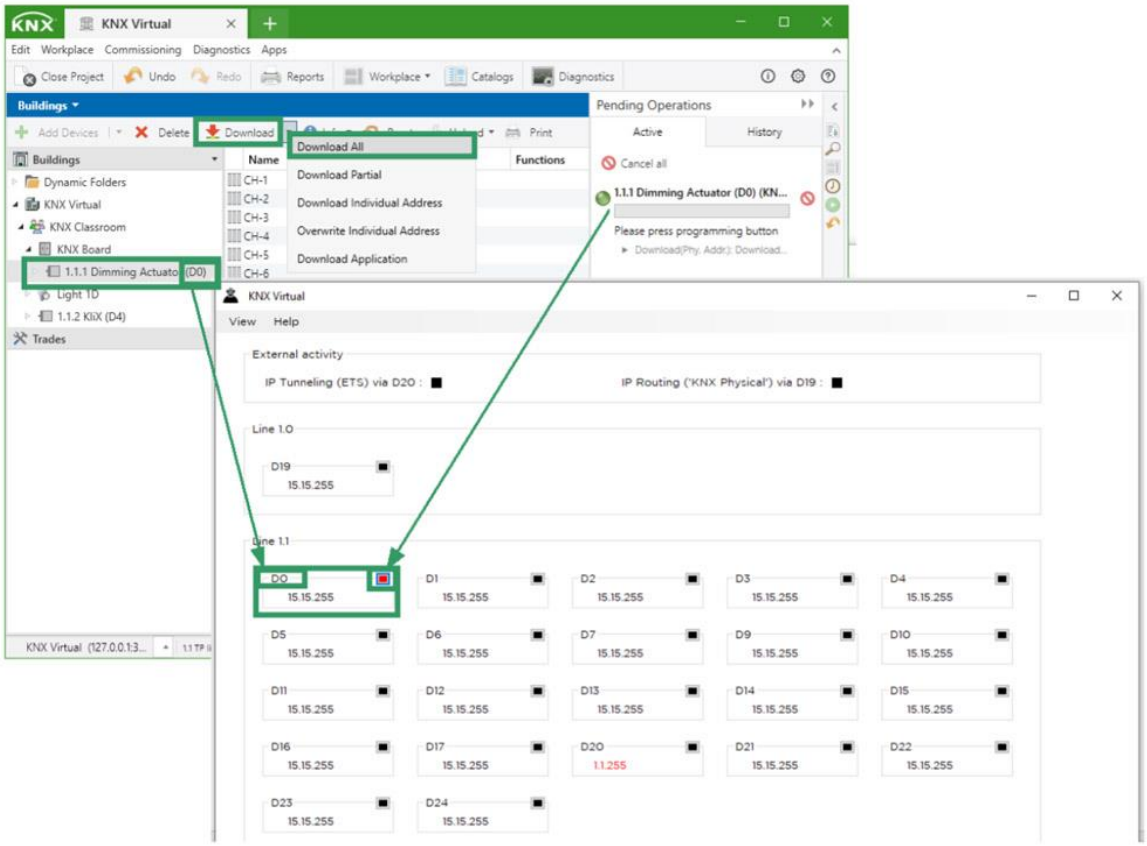

Рис. 27 Загрузка программы на устройство и программирование индивидуального адреса

Сейчас интерфейс программы изменен. В пункте меню «Installation» программы KNX Virtual нужно выбрать пункт «Configuration», рис. 28.

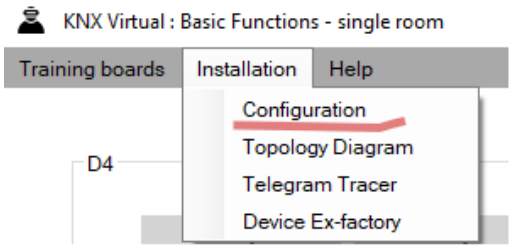

Рис. 28 Выбор пункта «Настройки» в меню KNX Virtual

Откроется окно настройки устройств, рис. 29. Кнопками переключения в режим программирования являются ячейки столбца ID. При нажатии они окрашиваются красным цветом. Когда при отправке конфигурации на устройство в программе ETS вы увидите запрос на активацию кнопки программирования, нужно открыть окно рис. 29 и нажать кнопку соответствующего устройства.

Segment	ID	OrderNr	AES	Description	ProgVersion	Status	A	L	D	SN
S1	D25	BO.ip	No	Binary Output	-	Unloaded	0	0	0	00FA-00252500
	D19^			Undefined: block/block						
S2	D19_	C.ip.tp	No	IP/TP Coupler	-	Unloaded	0	0	0	00FA-00251900
	D0	DA.tp	-	Dimming Actuator	-	Unloaded	0	0	0	00FA-00250000
	D2	BS.tp	-	Blinds/Shutter Actuator	-	Unloaded	0	0	0	00FA-00250200
	D4	KX.tp	-	KliX	-	Unloaded	0	0	0	00FA-00250400
	D7	SA.tp	-	Switching Actuator	-	Unloaded	0	0	0	00FA-00250700
	D13	SC.tp	-	Scenario Controller	-	Unloaded	0	0	0	00FA-00251300
	D14	LM.tp	-	Logic Module	-	Unloaded	0	0	0	00FA-00251400
	D20	I.ip.tp	-	IP/TP Interface	v1.0	Loaded	1	0	255	00FA-00252000
	D5^			Undefined: block/block						
S3	D5_	C.tp.tp	No	TP/TP Coupler	-	Unloaded	0	0	0	00FA-00250500
	D1	PB.tp	No	Push Button Interface	-	Unloaded	0	0	0	00FA-00250100

Рис. 29 Окно настройки устройств KNX Virtual версии 2.5

Затем выберите второе устройство для программирования в представлении «Здания» и снова опцию «Загрузить все», рис. 30 (с учетом приведенных выше комментариев).

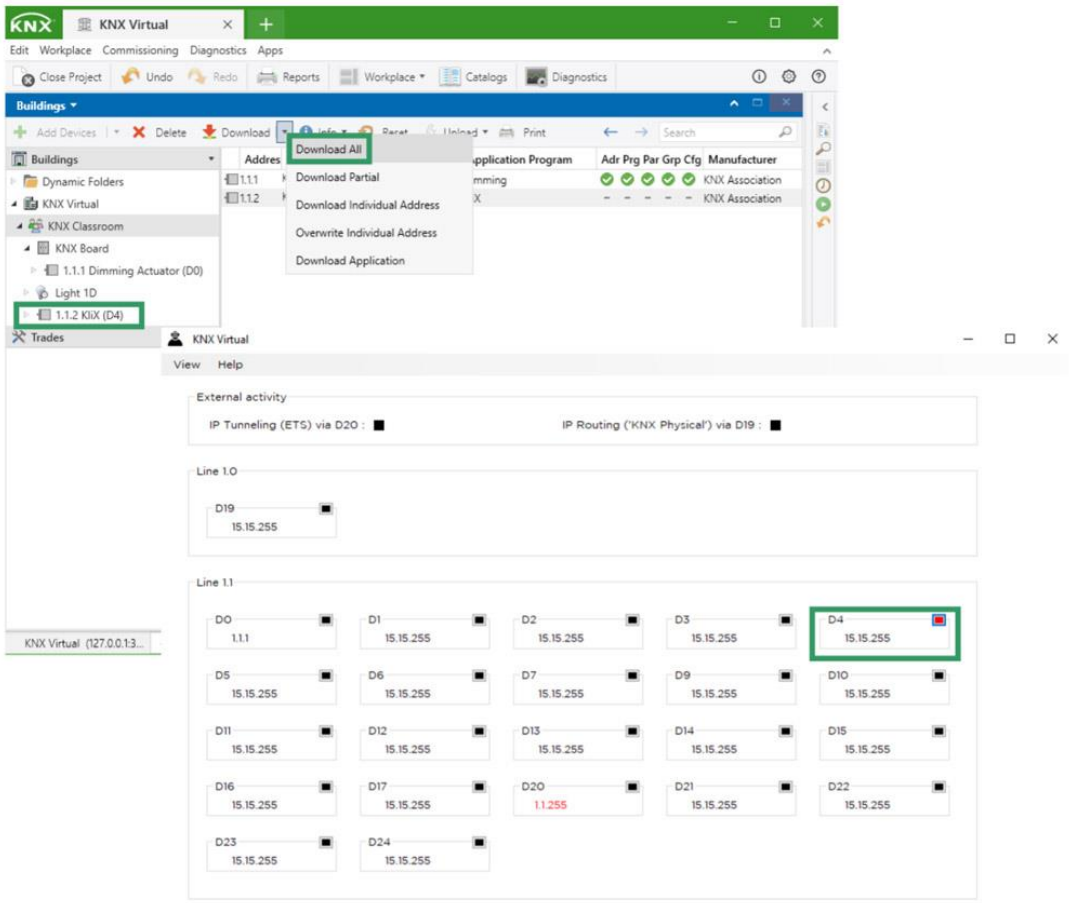

Рис. 30 Программирование второго устройства

На рис. 31 показано, что при выборе комнаты в структуре проекта мы видим, что оба устройства полностью запрограммированы.

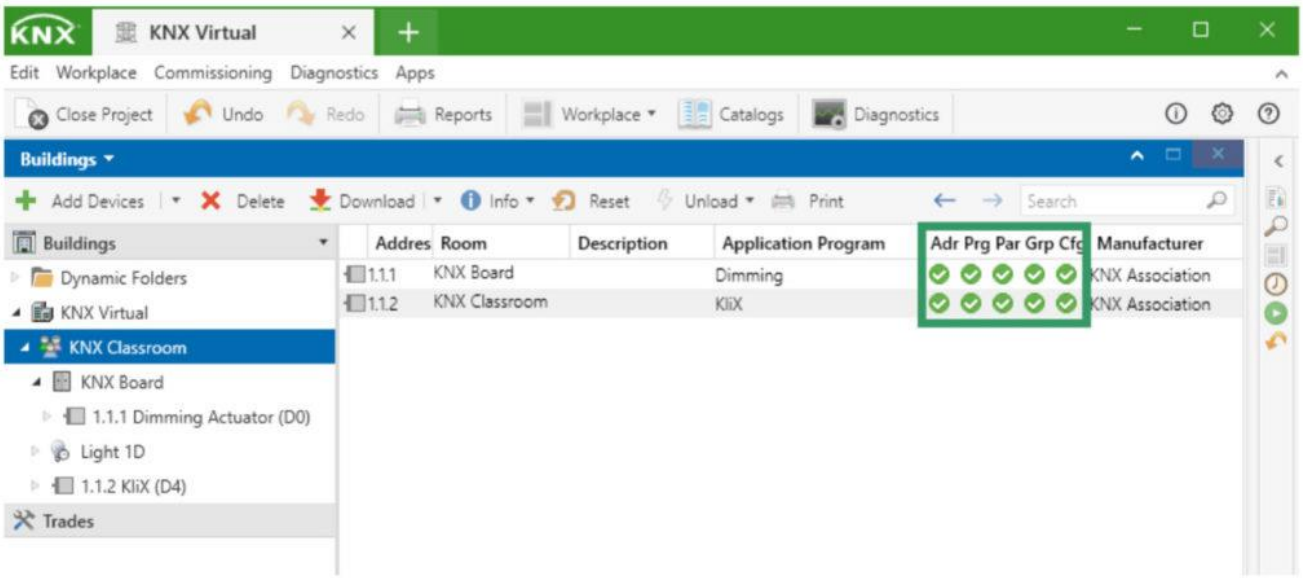

Рис. 31 Результат настройки и программирования устройств

Часть 3 – Тестирование и диагностика

KNX Virtual позволяет управлять несколькими типами устройств, от самых «традиционных» (для переключения, регулирования освещения или управления жалюзи) до более продвинутых (например, модулей сигнализации, метеостанции или логических модулей). В KNX Virtual есть несколько предварительно настроенных видов, показывающих взаимодействие между виртуальными устройствами. Выберите представление «Базовый» или «Затемнение ++», рис. 32.

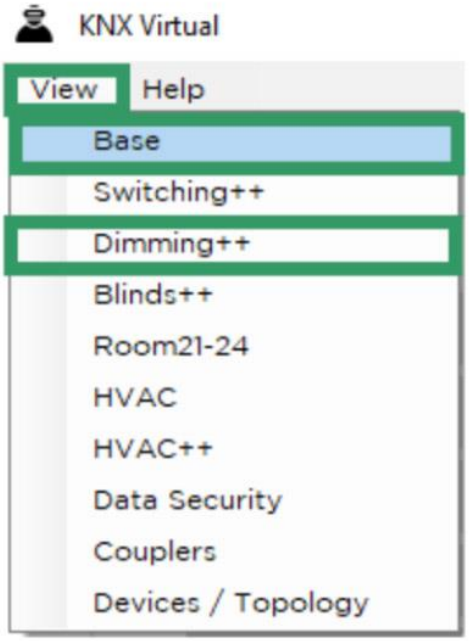

Рис. 32 Выбор представления виртуальных приборов на стенде

В окне «Базовый» вид, можно увидеть взаимодействие и поведение первых каналов настроенных устройств (KLiX и Dimming Actuator). Для кнопки возможно короткое и длительное нажатие для имитации, соответственно, переключения и изменения интенсивности. Значение состояния первого канала диммера отображается в процентах интенсивности, рис. 33.

Рис. 33 Проверка работы настроенных алгоритмов на виртуальных приборах

Соединение с KNX Virtual позволяет использовать монитор шины KNX и групповые адреса в ETS6. С помощью Group Monitor окна диагностики можно отправлять управляющие телеграммы на устройства и видеть результат как в ETS, так и по изменению состояния виртуальных устройств на стенде KNX Virtual, рис. 34.

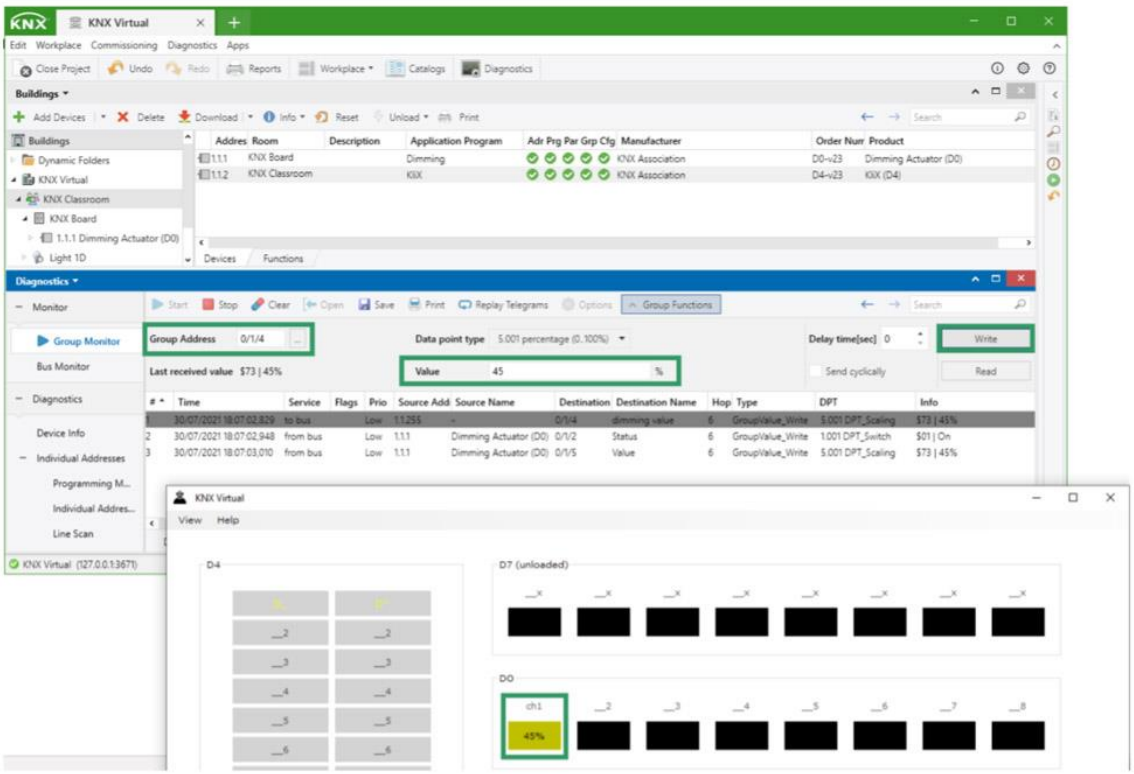

Рис. 34 Управление виртуальными устройствами программно из ETS

Варианты заданий

Вариант	Топология стенда	Что требуется настроить
1	Basic functions. Multiply room	Контроллер первой комнаты Room21 на управление включением-выключением света выключателем D7.1,

		диммером D0.1, управление жалюзи контроллером D2.1
2	Basic functions. Multiply room	Контроллер первой комнаты Room21 на управление включением-выключением света выключателем D7.2, диммером D0.2, управление жалюзи контроллером D2.1
3	Basic functions. Multiply room	Контроллер второй комнаты Room22 на управление включением-выключением света выключателем D7.3, диммером D0.3, управление жалюзи контроллером D2.2
4	Basic functions. Multiply room	Контроллер второй комнаты Room22 на управление включением-выключением света выключателем D7.4, диммером D0.4, управление жалюзи контроллером D2.2
5	Basic functions. Multiply room	Контроллер третьей комнаты Room23 на управление включением-выключением света выключателем D7.5, диммером D0.5, управление жалюзи контроллером D2.3
6	Basic functions. Multiply room	Контроллер третьей комнаты Room23 на управление включением-выключением света выключателем D7.6, диммером D0.6, управление жалюзи контроллером D2.3
7	Basic functions. Multiply room	Контроллер четвертой комнаты Room24 на управление включением-выключением света выключателем D7.7, диммером D0.7, управление жалюзи контроллером D2.4
8	Basic functions. Multiply room	Контроллер четвертой комнаты Room24 на управление включением-выключением света выключателем D7.8, диммером D0.8, управление жалюзи контроллером D2.4
9	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 первый канал, второй кнопкой D4 диммером D0 первый канал, третьей кнопкой D4 контроллером жалюзи D2 первый канал.
10	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 второй канал, второй кнопкой D4 диммером D0 второй канал, третьей кнопкой D4 контроллером жалюзи D2 второй канал.
11	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 третий канал, второй кнопкой D4 диммером D0 третий канал, третьей кнопкой D4 контроллером жалюзи D2 третий канал.
12	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 четвертый канал, второй кнопкой D4 диммером D0 четвертый канал, третьей кнопкой D4 контроллером жалюзи D2 четвертый канал.
13	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 пятый канал, второй кнопкой D4 диммером D0 пятый канал, третьей кнопкой D4 контроллером жалюзи D2 пятый канал.
14	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 шестой канал, второй кнопкой D4 диммером D0 шестой канал, третьей кнопкой D4 контроллером жалюзи D2 шестой канал.
15	Basic functions.	Управление первой кнопкой D4 выключателем света D7

	Single room	седьмой канал, второй кнопкой D4 диммером D0 седьмой канал, третьей кнопкой D4 контроллером жалюзи D2 седьмой канал.
16	Basic functions. Single room	Управление первой кнопкой D4 выключателем света D7 восьмой канал, второй кнопкой D4 диммером D0 восьмой канал, третьей кнопкой D4 контроллером жалюзи D2 восьмой канал.
17	Basic functions. Single room	Управление третьей кнопкой D4 выключателем света D7 первый канал, четвертой кнопкой D4 диммером D0 первый канал, пятой кнопкой D4 контроллером жалюзи D2 первый канал.
18	Basic functions. Single room	Управление третьей кнопкой D4 выключателем света D7 второй канал, четвертой кнопкой D4 диммером D0 второй канал, пятой кнопкой D4 контроллером жалюзи D2 второй канал.
19	Basic functions. Single room	Управление третьей кнопкой D4 выключателем света D7 третий канал, четвертой кнопкой D4 диммером D0 третий канал, пятой кнопкой D4 контроллером жалюзи D2 третий канал.
20	Basic functions. Single room	Управление третьей кнопкой D4 выключателем света D7 четвертый канал, четвертой кнопкой D4 диммером D0 четвертый канал, пятой кнопкой D4 контроллером жалюзи D2 четвертый канал.

Отчет по работе вместе с файлом проекта загрузить на проверку. Экспорт проекта в формате knxproj можно сделать на странице запуска ETS, рис. 35.

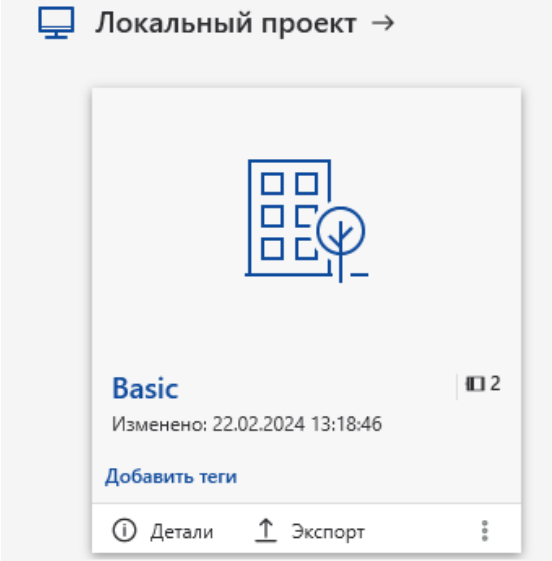

Рис. 35 Экспорт проекта

2. Практическая работа №2

БЫСТРОЕ ПРОТОТИПИРОВАНИЕ РЕШЕНИЯ ИНТЕРНЕТА ВЕЩЕЙ В СРЕДЕ NODE-RED

Node-RED — это инструмент потокового программирования, первоначально разработанный в компании IBM командой IBM Emerging Technology Services [1]. Сейчас это часть JS Foundation. В Node-RED предложена парадигма потокового программирования, которое было изобретено в 70-х Джеем Полом Моррисоном. Потоковое программирование — это способ описания поведения приложения в виде сети черных ящиков или «узлов», как они называются в Node-RED. Каждый узел имеет определенный функционал, к нему поступают некоторые данные, он что-то делает с этими данными, а затем передает их на следующий узел. Сеть отвечает за поток данных между узлами. Эта модель отлично подходит для того, чтобы представить ее визуально: так она становится более доступной для широкого круга пользователей. Если кто-то пытается разобраться в проблеме, он может разбить задачу на отдельные шаги, взглянуть на поток и понять, что он делает, без необходимости разбираться в отдельных строках кода в каждом узле. Node-RED работает в среде исполнения Node.js, а для создания или редактирования потока («Flow») используется браузер. В браузере вы можете создавать свое приложение путем перетаскивания необходимых узлов («Node») из палитры в рабочую область и соединения их. Одним кликом по кнопке «Deploy» приложение разворачивается в среде исполнения и начинает работать. Палитра узлов может быть легко расширена путем установки новых узлов, созданных сообществом, а созданные вами потоки могут быть легко переданы в виде файлов JSON. Исходный код проекта был открыт в сентябре 2013 года. С тех пор он разрабатывался в открытом виде, кульминацией развития стало признание Node-RED одним из фундаментальных проектов JS Foundation в октябре 2016 года.

Для выполнения работы используется технология Docker-контейнеров. Docker — популярная технология контейнеризации, появившаяся в 2013 году. Идея разделения ресурсов внутри операционной системы впервые появилась в 1979 году в UNIX версии 7. Ее реализацией стали Chroot jail и операция или команда Chroot. С помощью Chroot jail процесс и его дочерние элементы изолировались от основной ОС. Docker можно считать продолжением этой идеи. Спустя 20 лет появился FreeBSD Jail — механизм, позволяющий внутри одной ОС создать несколько изолированных систем, которые назвали тюрьмами. Далее технологии контейнеризации развивались стремительно. В 2001 году появился Linux VServer, который использовал chroot-подобную утилиту и применялся для безопасного разделения ресурсов. Каждый раздел назывался «контекстом безопасности», а операционная система внутри него — виртуальным частным сервером. В 2007 году компания Google предложила функцию Control Groups, cgroups, ограничивающую использование ресурсов (CPU, ROM, дисковый ввод-вывод, сеть и т.д.) на уровне групп процессов. Спустя год выпустили Linux Containers (LXC), построенную на механизмах управления пространствами имен namespaces и разделению ресурсов с помощью cgroups. Это уже практически готовая технология контейнеризации, как мы ее знаем сейчас. Docker, добавил веб-технологию взаимодействия клиента и сервиса с удобными утилитами управления.

ВЫПОЛНЕНИЕ РАБОТЫ

1. Скачайте виртуальную машину, содержащую операционную систему Linux с установленным сервисом Docker по адресу https://cloud.tusur.ru/index.php/s/Twd4TAHJCTAtwLp

2. Импортируйте образ в среду VirtualBox и запустите виртуальную машину. Обратите внимание на сетевое соединение, выберите соединение типа «Сетевой мост» или «Адаптер хоста». Для входа используйте логин docker и пароль такой же как логин.

3. Выполните команду `docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red`

Здесь `docker run` создает из локального образа и запускает контейнер, если образа нет (а его нет при первом запуске) локально, то идет обращение к реестру образов в сети и выкачивание его оттуда.

Ключи `-it` дают интерактивный доступ к командной строке той операционной системы, которая работает внутри контейнера, так что вам будут видны сообщения при запуске Node-Red.

Ключ `-p 1880:1880` реализует проброс портов из контейнера в машину, на которой он запущен. Сервис Node-Red работает на порту 1880 внутри контейнера, на этот же порт виртуальной машины выполняется перенаправление. Для обращения к сервису нужно указать IP-адрес виртуальной машины и порт 1880, например так 192.168.56.2:1880.

Ключ `-v node_red_data:/data` подключает каталог data контейнера к каталогу в виртуальной машине, реализуя такой «проброс каталогов». Данные проектов Node-Red сохраняются в каталоге data и если нет желания каждый раз все начинать сначала, такое подключение каталога будет очень полезно.

Ключ `--name mynodered` дает удобное и понятное имя контейнеру. Без него контейнер получает символьное имя автоматически, но оно значительно длинее. Обращаться к контейнеру можно и по его идентификатору, но и автоматически назначенное имя и идентификатор не так удобны, как заданное вами имя.

Последняя часть команды `nodered/node-red` содержит имя образа контейнера в онлайн реестре DockerHub.

4. Откройте браузер на компьютере и обратитесь по адресу виртуальной машины с указанием порта 1880, чтобы получить доступ к сервису Node-Red. Вы должны увидеть веб-страницу с автоматически сгенерированным первым потоком, рис. 1.

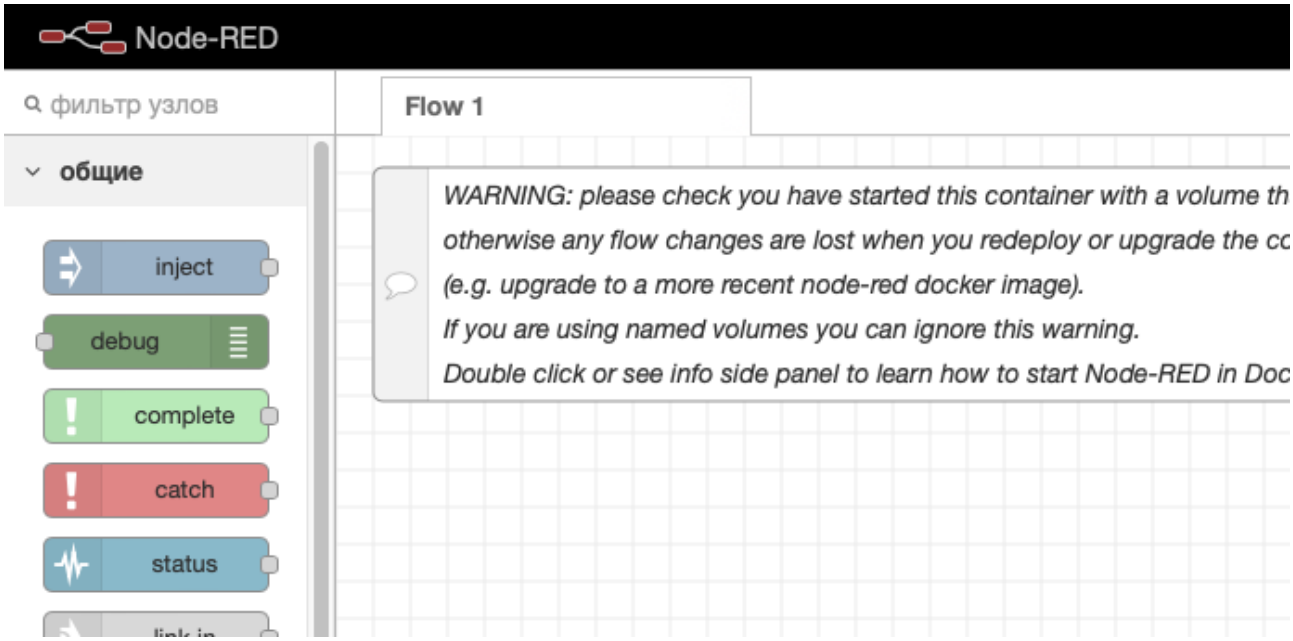

Рис. 1 Окно Node-Red при первом запуске

Установите дополнительную палитру **node-red-contrib-web-worldmap**

Это можно сделать из окна браузера. Откройте меню настройки в правом верхнем углу и выберите пункт «Управление палитрой», рис. 2.

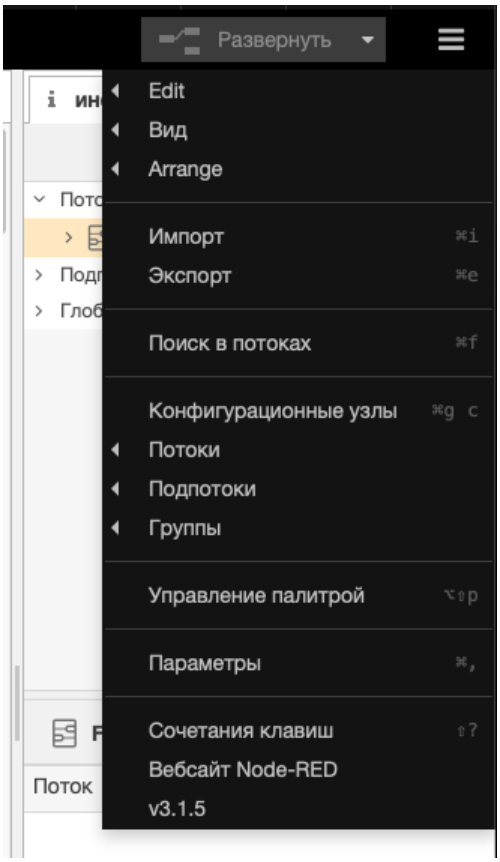

Рис. 2 Меню настройки Node-Red

Перейдите на вкладку «Установить» и введите название требуемого пакета, рис. 3

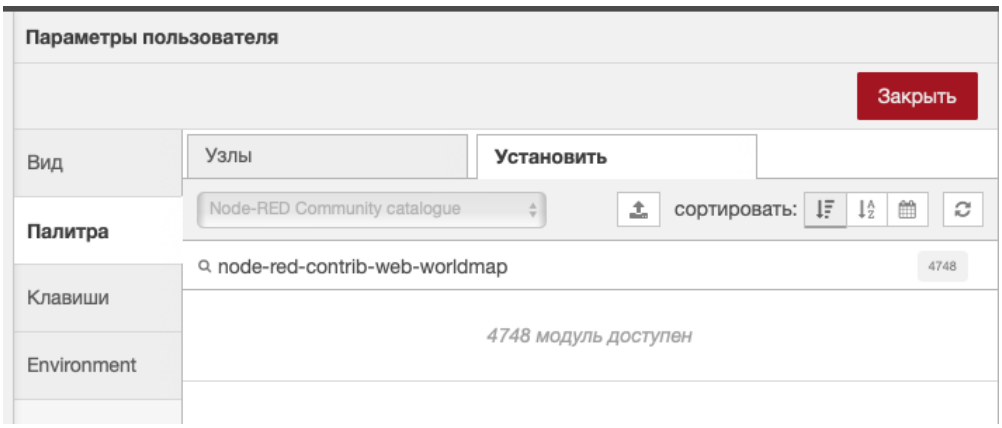

Рис. 3 Установка дополнительных пакетов

После установки этого пакета в палитре появится новый раздел под названием location с 4 новыми узлами, рис. 4. Мы задействуем узел «карта мира».

Рис. 4 Узлы, добавленные установленным пакетом

Итоговая схема узлов и соединений должна выглядеть как на рис. 5. Узел, инициирующий работу нашей программы, добавляет метку времени и может выполнять запуск в цикле с заданной периодичностью.

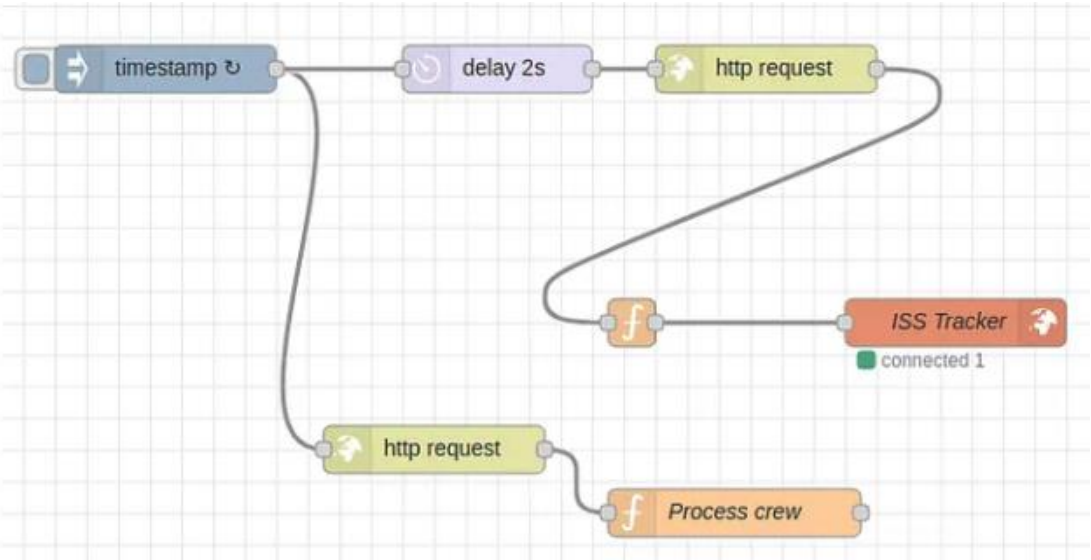

Рис. 5 Итоговая схема узлов и соединений

Далее формируются два веб-запроса к доступному API международной космической станции. Результат одного запроса отображается на карте на отдельной веб-странице, результат второго содержит информацию о составе команды космонавтов в данное время и выводится во всплывающем окне при щелчке по символу МКС на карте.

Добавьте на рабочее поле узел «Inject» из блока «Общие» и настройте его как показано на рис. 6. Добавьте два узла «http request» из блока «Сеть» и узел задержки из блока функций. Так как запросы отправляются к одному и тому же API, но сами запросы разные, мы вводим задержку, чтобы не создавать помех от одного запроса другому. Информация о составе экипажа должна быть получена раньше, потому что она используется нами при отображении положения МКС на карте.

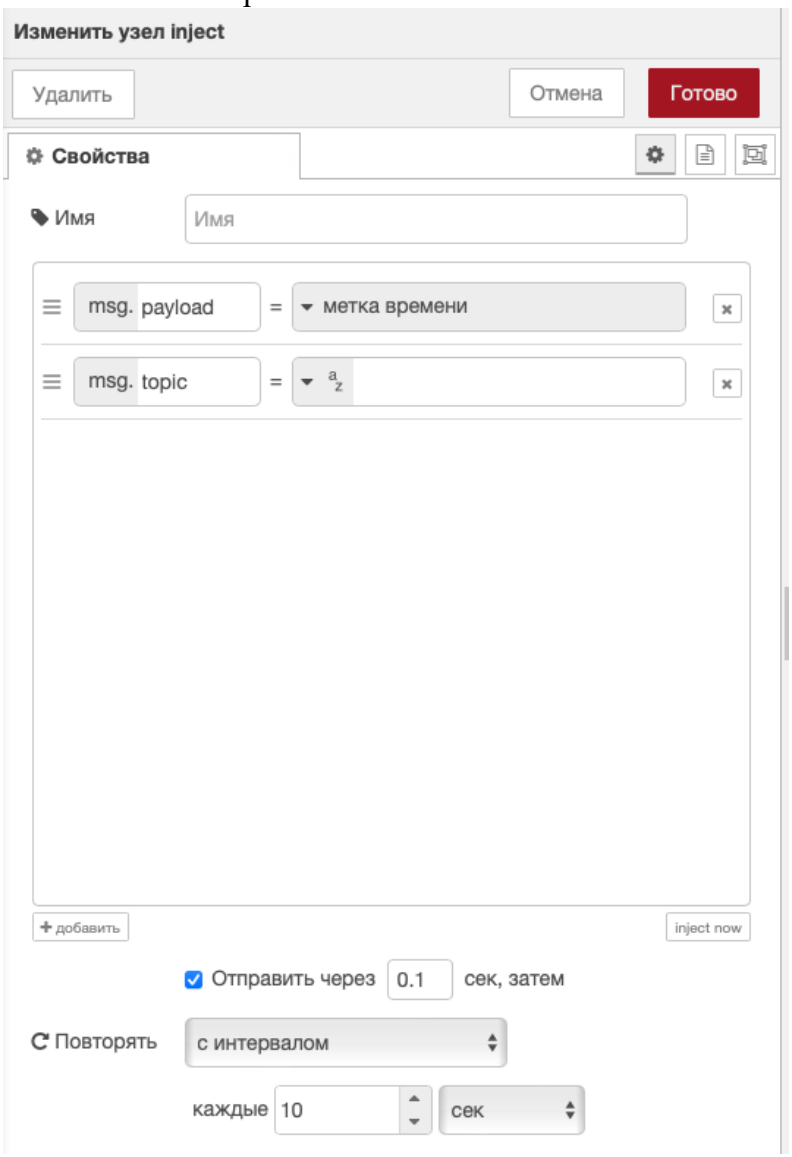

Рис. 6 Настройка узла «Inject»

Настройте запросы текущего положения и состава команды МКС, как показано на рис. 7 и 8.

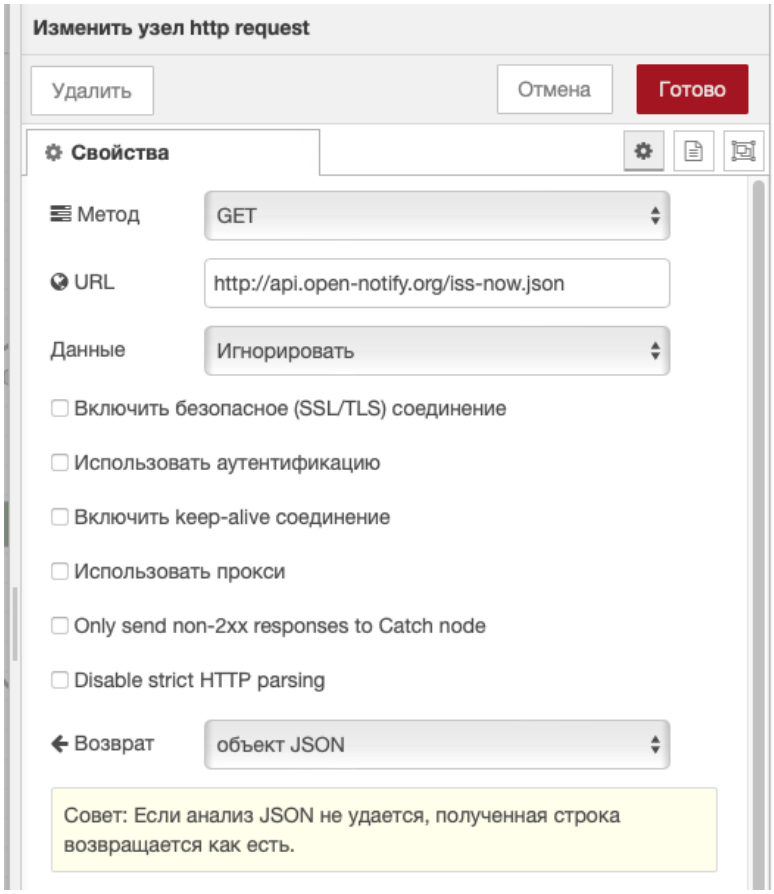

Рис. 7 Настройка запроса текущего положения МКС

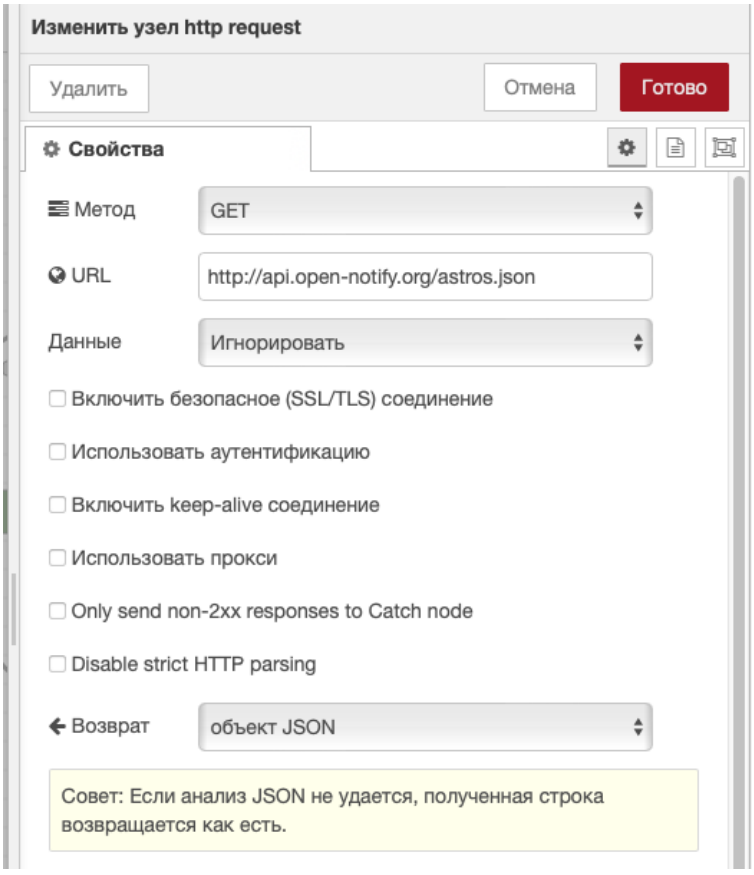

Рис. 8 Настройка запроса состава команды МКС

Создайте функцию обработки информации о составе команды для вывода ее в окне карты, как показано на рис. 9

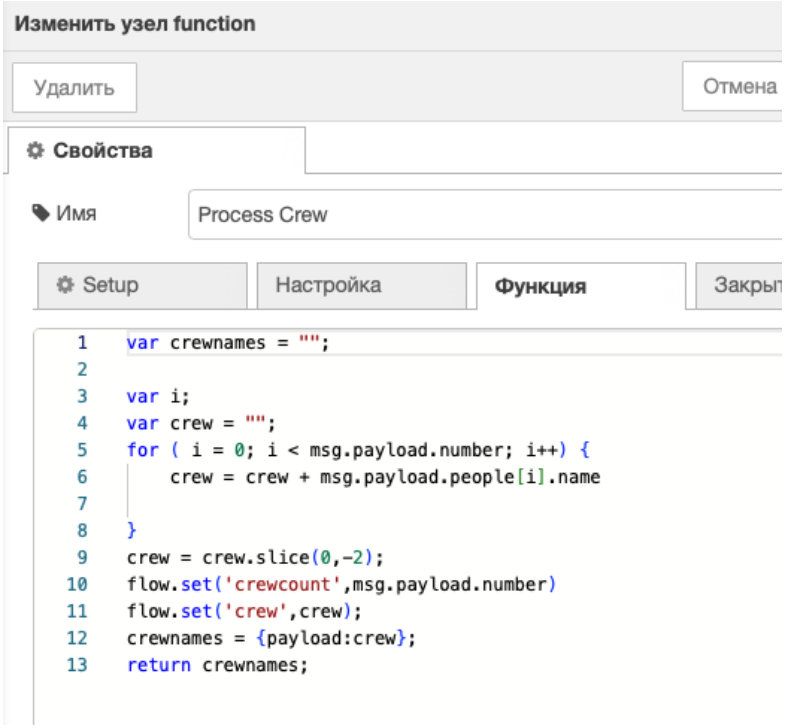

Рис. 9 Функция обработки данных состава команды МКС

Вторая функция обрабатывает данные о положении МКС, т. к. формат, требуемый для узла worldmap, отличается от формата, получаемого по запросу от API. Создайте вторую функцию, как показано на рис. 10.

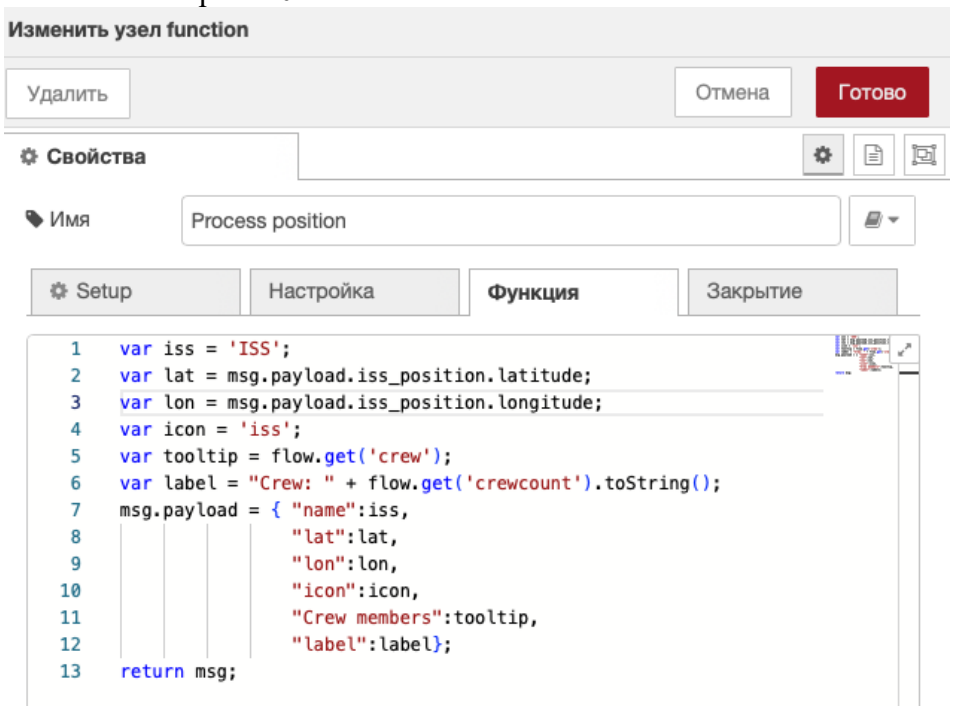

Рис. 10 Функция обработки данных положения МКС

Наконец, добавьте на рабочее поле и настройте узел карты, рис. 11.

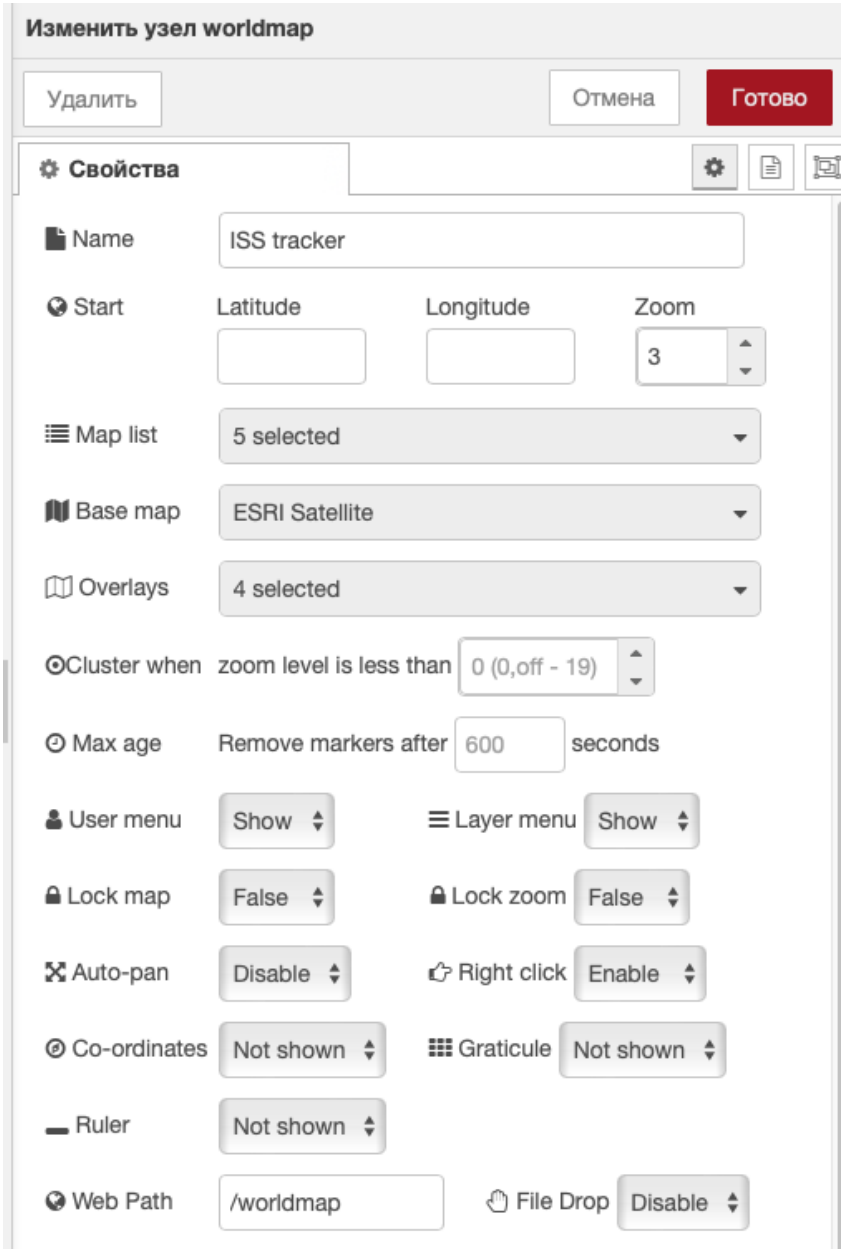

Рис. 11 Настройка узла выведения данных на карту

Выполните развертывание настроенных потоков клавишей Deploy (Развернуть) в правом верхнем углу окна Node-Red. Затем откройте страницу с тем же адресом и портом, но дополнительной папкой /worldmap, которую вы определили при настройке узла карты. Видите ли вы отображение положения МКС на карте? Выводится ли во всплывающем окне информация о составе экипажа? Меняется ли информация о положении со временем?

3. Практическая работа №3

ОБРАБОТКА ДАННЫХ СЕНСОРОВ, ПОЛУЧЕННЫХ ИЗ БАЗЫ ДАННЫХ MongoDB

Устройства Интернета вещей, как правило, оснащены датчиками, сенсорами. Объем этой информации может быть значительным, поэтому она сохраняется в базах данных. В связи с отсутствием строго заданной структуры данных у многих источников информации в Интернет, в том числе у устройств Интернета вещей, представляющих собой объединение больших сетей разнотипных датчиков, на рубеже 2000 — 2010гг появился новый тип баз данных — нереляционные базы данных, NoSQL. В данной работе мы познакомимся с MongoDB – популярным современным представителем такого рода базы данных. Если в традиционных реляционных базах данных данные хранятся в структурах табличного типа, для ячеек которых задан определенный формат, то в нереляционных базах информация хранится в структурах типа «ключ-значение», объединяемых в документы и коллекции, рис.1.

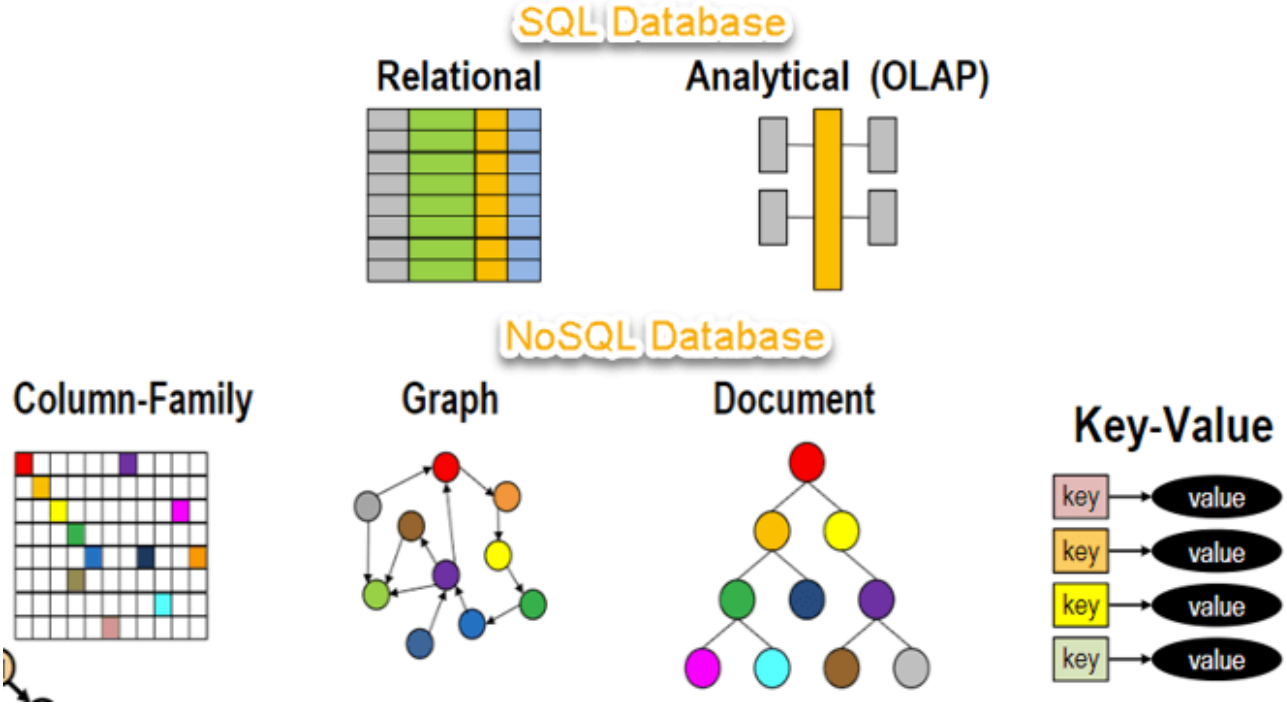

Рис. 1 Различия между реляционными и нереляционными БД

Информация в нереляционных базах не структурирована заранее, поэтому требуются дополнительные операции по ее структуризации перед обработкой. Отсутствие заданной структуры — одна из особенностей так называемых «Больших Данных», BigData. Такого рода данные создаются современными информационными системами, в том числе устройствами Интернета вещей. Большие данные получили свое имя за то, что они на самом деле большие и принципиально невозможно сохранение данных такого объема на одном физическом носителе или даже на одном компьютере, имеющем несколько физических носителей. Поэтому требуются специальные инженерные решения для хранения таких данных. Это свойство по-английски называется «размер», Volume. Упомянутое ранее свойство разнородности, неструктурированности больших данных по-английски называется «разнообразие», Variety. Часто говорят что Большие Данные характеризуются тремя «V»

(четырьмя, пятью, иногда добавляют дополнительные «V»), подразумевая три (или более) характеристики, которые описываются английскими терминами, начинающимися на букву «V». Мы отметили два свойства, третьим является скорость генерации данных, Velocity. Для Больших Данных характерным является высокая скорость создания новых данных, поступления информации. Это требует пересмотра традиционных методов обработки информации. Так как обычно собираемая информация накапливалась и сохранялась в той же базе данных, например, и уже после этого (полного сохранения) начинала обрабатываться. Большие Данные создают, во-первых, такой поток данных, который никогда не заканчивается, поэтому ждать полного сохранения бессмысленно, во-вторых, сами данные могут быстро устаревать и обрабатывать их когда-то «потом» тоже может быть бессмысленно, потому что они будут уже не актуальны. То есть такие данные должны обрабатываться в потоковом режиме, сразу. Собственно, указанные новые свойства данных: Volume, Variety и Velocity, а также разработанный инженерный инструментарий для работы с данными нового типа и послужили основанием для введения термина «Большие Данные». Здесь наглядно проявляется связь между сквозными технологиями: Интернета вещей и Больших Данных.

Для обработки данных мы используем язык Python и средство интерактивной работы с этим языком на основе веб-технологии Jupyter Notebook. Python имеет много средств, дополнительно подключаемых инструментов для обработки и визуализации данных. Технологии машинного обучения — ключевой инструмент современного анализа данных и еще одна сквозная технология. Комплексные дисциплины, подобные Интернету вещей, требуют для своего освоения владения широким инструментарием смежных дисциплин.

ВЫПОЛНЕНИЕ РАБОТЫ

Мы рассмотрим ниже разобранный детально сценарий обработки данных до получения результата, после чего вы выполните подобный анализ других данных самостоятельно.

1. Установите Jupyter Notebook с интерпретатором Python на свой компьютер. Используйте компактную версию, не требующую прав администратора https://www.portabledevapps.net/jupyter-portable.php [2]. Это самораспаковывающийся архив с расширением exe. Инсталлятору нужно указать любой каталог, на который у вас есть полный доступ, например, Документы. Установка программы завершается ее автоматическим запуском в свернутом виде. В трее появляется значок рис. 2:

Рис. 2 Символ запущенного приложения Jupyterlab

2. Запустите виртуальную машину Alpine Linux с установленным Docker-сервисом https://cloud.tusur.ru/index.php/s/Twd4TAHJCTAtwLp. Проверьте сетевые настройки виртуальной машины, выберите соединение типа «Сетевой мост», чтобы IP-адрес находился в том же диапазоне, что у основного компьютера, на котором у вас работает Jupyter Notebook.

3. Подключитесь к виртуальной машине внешним клиентом SSH, например MobaXterm - https://download.mobatek.net/2402024022512842/MobaXterm_Portable_v24.0.zip [3].

4. Скачайте архив с базой данных по адресу https://cloud.tusur.ru/index.php/s/MeJxZMLSxLYW8Ts Эти данные взяты из проекта «Открытая Африка», https://africaopendata.org/ крупнейшего независимого хранилища открытых данных африканского континента. Наборы данных для разных локаций содержат показания содержания пыли: PM (твердых частиц), температуры и влажности, полученные с помощью недорогих датчиков. Датчики измеряют концентрацию твердых частиц в воздухе, разделяя частицы диаметром менее или равным 1 микрометру (PM1), 2,5 микрометра (PM2,5) и частицы диаметром менее или равными 10 микрометрам (PM10). Набор данных также включает информацию о типе датчика, дате, времени и месте получения показаний. Обратите внимание, что P0 в данных представляет собой PM1, P2 соответствует PM2,5, а P1 представляет PM10.

5. Создайте в виртуальной машине в домашнем каталоге пользователя папку data командой **mkdir data**

6. Скопируйте в клиенте MobaXterm файл архива базы данных в домашнюю папку пользователя docker.

7. Переключитесь на пользователя root командой **su –** с указанием пароля для пользователя docker.

8. Распакуйте архив командой:

```
tar -zxvf /home/docker/AirQualityAfrica.tar.gz -C /home/docker/
```

9. Выйдите из профиля пользователя root командой **exit**. Создайте и запустите Docker-контейнер с работающим сервисом MongoDB командой:

```
docker run -d -p 27017:27017 --rm -v ~/data/:/data/db --name mongolab
                                mongo:4.0.4
```

10. Подключитесь в командной строке к работающему контейнеру командой:

```
docker exec -it mongolab bash
```

11. Выполните восстановление базы данных из архива командой:

```
mongorestore --db AirQualityAfrica /data/db/AirQualityAfrica/
```

12. Подключитесь к базе данных изнутри контейнера командой **mongo**

13. В командной оболочке клиента MongoDB выполните команду **show dbs**, убедитесь что импортированная база данных присутствует в списке.

14. Выйдите из командной оболочки клиента MongoDB командой exit

15. Выйдите из командной строки контейнера командой exit

16. Откройте окно JupyterLab, вызвав контекстное меню правой клавишей мышки на значке в трее, рис. 3.

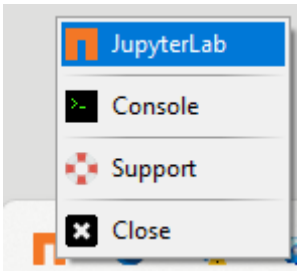

Рис. 3 Запуск JupyterLab

17. Создайте новый блокнот Юпитер, назовите его «Лабораторная работа №3», рис. 4. Для переименования найдите в меню File пункт Rename Notebook...

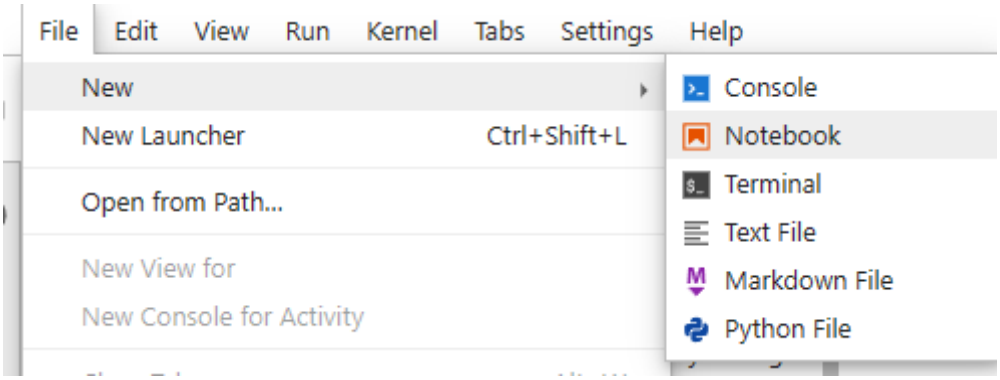

Рис. 4 Создание блокнота Jupyter Notebook

18. В первой ячейке блокнота импортируйте библиотеку Pandas и клиента для работы с базой данных MongoDB, рис. 5.

```
import pandas as pd
from pymongo import MongoClient
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15536/577788817.py in <module>
      1 import pandas as pd
----> 2 from pymongo import MongoClient

ModuleNotFoundError: No module named 'pymongo'
```

Рис. 5 Импорт библиотек

Вы получите такую же ошибку, указывающую на то, что среди доступных Python модулей отсутствует модуль pymongo. Установить недостающие модули можно из командной строки приложения JupyterLab, строка меню Console на рис. 3. Откройте командную строку и введите команду pip install pymongo, рис. 6.

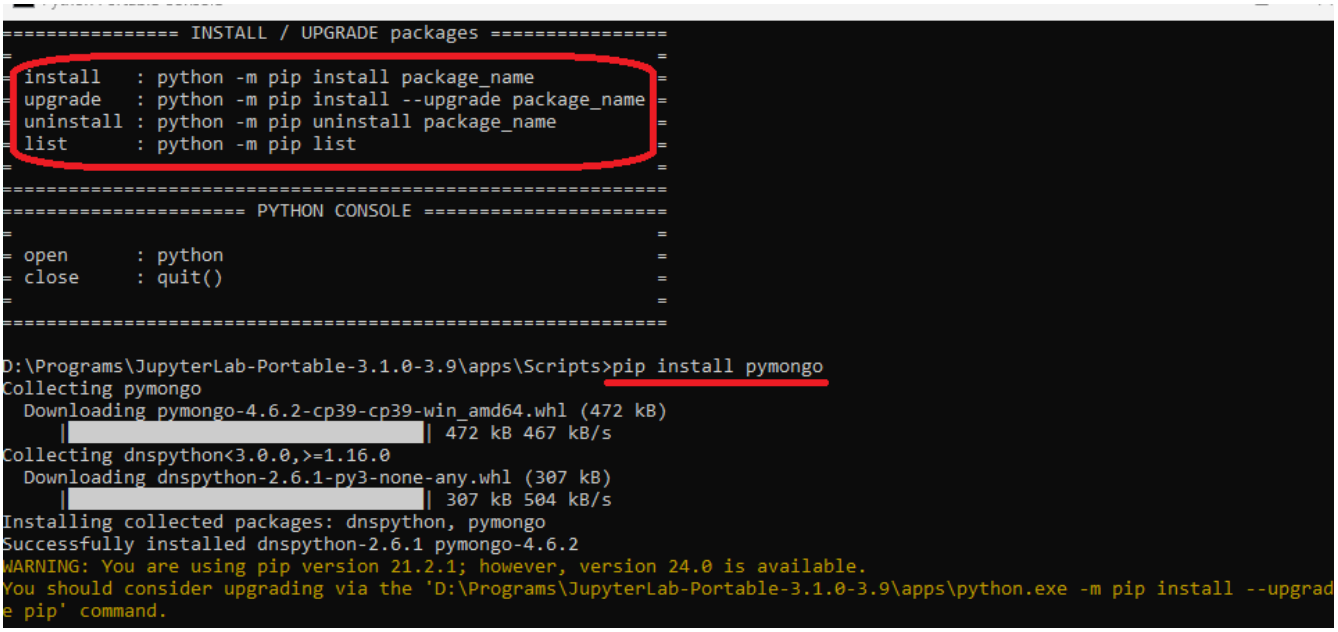

Рис. 6 Установка недостающего модуля

После этого повторно запустите ячейку блокнота на выполнение. Это можно сделать комбинацией клавиш Shift+Enter или щелчком мышкой по значку запуска в верхней части блокнота. Импортируйте дополнительно специальный виртуальный принтер форматированного вывода, он будет удобен при просмотре данных из базы. Сразу установим отступы для этого принтера в виде двух пробелов, рис. 7.

```
from pprint import PrettyPrinter

pp = PrettyPrinter(indent=2)
```

Рис. 7 Импорт PrettyPrinter

19. Для подключения к базе данных мы должны задать IP-адрес сервера и порт сервиса MongoDB. Порт по умолчанию для MongoDB 27017. Хотя наш сервис работает в контейнере, сам контейнер запущен в виртуальной машине и сетевое взаимодействие между ними прозрачное. Поэтому мы можем указать адрес виртуальной машины. Именно поэтому нам необходимо было запустить виртуальную машину с адаптером «Сетевой мост», чтобы она оказалась в той же сети, где основная машина и обе они получали адреса из одного адресного пространства, так чтобы видеть друг друга.

После подключения к MongoDB выведите список доступных баз данных, рис. 8.

```
client = MongoClient(host="192.168.0.30", port=27017)

client.list_database_names()
```

```
['AirQualityAfrica', 'admin', 'config', 'local']
```

Рис. 8 Подключение и вывод доступных баз данных

Убедитесь еще раз, что импортированная вами база данных присутствует в списке.

20. Создайте переменную для базы данных AirQualityAfrica и присвойте ей значения базы, так чтобы в программе можно было оперировать этой переменной. Прочитайте

```
db = client["AirQualityAfrica"]
```

```
db.list_collection_names()
```

```
['DarEsSalaam', 'Nairobi', 'Lagos']
```

доступные коллекции в этой базе данных, рис. 9.

Рис. 9 Создание переменной для соединения ее с базой данных

21. Присвоим отдельной переменной значение одной из коллекций и подсчитаем число документов в ней, возьмем первую локацию — DarEsSalaam, рис. 10:

```
des = db["DarEsSalaam"]
```

```
des.count_documents({})
```

```
73070
```

Рис. 10 Присвоение новой переменной значений коллекции и подсчет числа записей

22. Число записей в коллекции достаточно велико, это десятки тысяч. Выведем один документ, чтобы понять, что собой представляют записи, Рис. 11:

```
onedoc = des.find_one({})
pp.pprint(onedoc)
```

```
{ '_id': ObjectId('65eba0d429d7b76178b36ade'),
  'lat': -6.913,
  'location': 23,
  'lon': 39.29,
  'sensor_id': 31,
  'sensor_type': 'SDS011',
  'timestamp': datetime.datetime(2019, 1, 7, 7, 10, 10, 113000),
  'value': 11.5,
  'value_type': 'P2'}
```

Рис. 11 Вывод одного документа из коллекции

23. Значение «location», очевидно указывает на место установки датчика. Метод distinct позволяет определить число уникальных записей для конкретного ключа в базе. Определим число локаций установки датчиков в Дар-Эс-Саламе и число записей в коллекции для каждой локации, рис. 12. Для подсчета документов в конкретной локации мы указываем в параметрах фильтрации (в фигурных скобках) ключ и через двоеточие его значение. Видим что больше всего документов в нашей коллекции для локации 23.

```
des.distinct("location")
```

```
[23, 11, 42]
```

```
print("There are " + str(des.count_documents({"location": 23})) + " documents for the location number 23" )
print("There are " + str(des.count_documents({"location": 11})) + " documents for the location number 11" )
print("There are " + str(des.count_documents({"location": 42})) + " documents for the location number 42" )
```

```
There are 50070 documents for the location number 23
There are 40 documents for the location number 11
There are 22960 documents for the location number 42
```

Рис. 12 Определение числа локаций и числа записей для каждой из них

24. Тот же метод distinct позволит определить какие типы измерений выполняются. Для этого мы применяем метод к ключу «value_type», рис. 13. Можно видеть что измеряется температура, влажность и концентрация крупных частиц пыли: до 2,5 и 10 мкм.

```
des.distinct("value_type")
```

```
['P2', 'P1', 'humidity', 'temperature']
```

Рис. 13 Типы выполняемых измерений

25. С помощью метода find мы можем найти в базе все требуемые значения. Например, мы хотим исследовать динамику запыленности. При этом выбираем наиболее мелкие регистрируемые частицы пыли, в нашем случае — до 2.5 мкм, т. е. P2. На рис. 14 показано как с помощью метода find сформировать запрос к базе, чтобы в выводе получить только интересующие значения. Обратите внимание, что вывод намеренно ограничивается. Мы добавляем limit(3), чтобы вывести только три записи, т. к. число записей в коллекции очень велико и в данном случае мы просто хотим увидеть, что наш запрос работает корректно, нам не нужны тысячи записей.

```
result = des.find({"value_type": "P2"}).limit(3)
pp.pprint(list(result))
```

```
[ { '_id': ObjectId('65eba0d429d7b76178b36ade'),
    'lat': -6.913,
    'location': 23,
    'lon': 39.29,
    'sensor_id': 31,
    'sensor_type': 'SDS011',
    'timestamp': datetime.datetime(2019, 1, 7, 7, 10, 10, 113000),
    'value': 11.5,
    'value_type': 'P2'},
  { '_id': ObjectId('65eba0d429d7b76178b36ae2'),
    'lat': -6.913,
    'location': 23,
    'lon': 39.29,
    'sensor_id': 31,
    'sensor_type': 'SDS011',
    'timestamp': datetime.datetime(2019, 1, 7, 7, 12, 41, 106000),
    'value': 10.6,
    'value_type': 'P2'},
  { '_id': ObjectId('65eba0d429d7b76178b36ae6'),
    'lat': -6.913,
    'location': 23,
    'lon': 39.29,
    'sensor_id': 31,
    'sensor_type': 'SDS011',
    'timestamp': datetime.datetime(2019, 1, 7, 7, 15, 12, 969000),
    'value': 10.8,
    'value_type': 'P2'}]
```

Рис. 14 Фильтрация вывода по интересующему параметру

26. Мы можем комбинировать условия фильтрации, как показано на рис. 15. Здесь мы указали конкретную локацию и тип измерения. Аргумент projection позволяет нам выбрать только отдельные столбцы в результате фильтрации. Так как нам нужны только значения запыленности и метки времени, указываем для них значения 1 или True. Столбец с идентификатором документа выводится в любом случае, поэтому если он нам не нужен, для него требуется явно указать 0 или False, чтобы удалить его вывод. Обратите внимание что в этот раз вместо ограничения вывода мы применили метод next(), позволяющий вывести один очередной документ от позиции итератора.

```
result = des.find(
                {"location": 23, "value_type": 'P2'},
    projection={"value": 1, "timestamp": 1, "_id": 0}
                )
pp.pprint(result.next())
```

```
{'timestamp': datetime.datetime(2019, 1, 7, 7, 10, 10, 113000), 'value': 11.5}
```

Рис. 15 Отбор данных из базы по заданному критерию с оставлением только интересующих столбцов

27. Сохраним отобранные из базы данные в датафрейм Pandas для дальнейшей обработки. Обратите внимание на особенность работы с итератором. Переменную result нужно использовать немедленно после выполнения запроса к базе, она не хранит значение

запроса. Если вы выполните какое-то промежуточное действие: печать результата, например, то в датафрейм будет нечего передавать.

```
result = des.find(
                 {"location": 23, "value_type": 'P2'},
    projection={"value": 1, "timestamp": 1, "_id": 0}
                 )
#pp.pprint(result.next())
```

```
df = pd.DataFrame(result)
df.head()
```

	timestamp	value
0	2019-01-07 07:10:10.113	11.50
1	2019-01-07 07:12:41.106	10.60
2	2019-01-07 07:15:12.969	10.80
3	2019-01-07 07:17:44.694	10.77
4	2019-01-07 07:20:19.518	9.50

Рис. 16 Сохранение результатов запроса в датафрейм

28. Полученный датафрейм содержит отдельную колонку с индексом и колонку с временным штампом. Логично в данном случае просто использовать временной штамп в качестве индекса. Исправим это, рис. 17

```
df = pd.DataFrame(result).set_index("timestamp")
df.head()
```

	value
timestamp	
2019-01-07 07:10:10.113	11.50
2019-01-07 07:12:41.106	10.60
2019-01-07 07:15:12.969	10.80
2019-01-07 07:17:44.694	10.77
2019-01-07 07:20:19.518	9.50

Рис. 17 Задание в качестве индекса одной из колонок датафрейма

29. Также колонка, содержащая значения запыленности, имеет не очень понятное значение value. Мы знаем, что в запросе к базе отбирались значения запыленности P2, но лучше явно задать имя столбца датафрейма, соответствующее типу измерений, рис. 18.

```
df.rename(columns={"value":"P2"}, inplace=True)
df.head()
```

	P2
timestamp	
2019-01-07 07:10:10.113	11.50
2019-01-07 07:12:41.106	10.60
2019-01-07 07:15:12.969	10.80
2019-01-07 07:17:44.694	10.77
2019-01-07 07:20:19.518	9.50

Рис. 18 Переименование столбца датафрейма

30. Сформируем функцию, которая выполняет запрос с возвратом готового датафрейма, рис. 19:

```
def wr(collection):
    # Отбор из базы требуемых значений
    result = des.find(
                 {"location": 23, "value_type": 'P2'},
    projection={"value": 1, "timestamp": 1, "_id": 0}
                 )
    # Сохранение отобранных значений в датафрейме Pandas с заменой индекса
    df = pd.DataFrame(result).set_index("timestamp")

    # Переименование столбца "value" в столбец "P2"
    df.rename(columns={"value":"P2"}, inplace=True)

    return df
```

Рис. 19 Создание функции

31. Убедимся, что функция корректно работает, рис. 20:

```
df = wr(des)
df.head()
```

	P2
timestamp	
2019-01-07 07:10:10.113	11.50
2019-01-07 07:12:41.106	10.60
2019-01-07 07:15:12.969	10.80
2019-01-07 07:17:44.694	10.77
2019-01-07 07:20:19.518	9.50

Рис. 20 Проверка работы функции

32. Выведем на график значения показаний датчика, рис. 21

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(15,6))

df["P2"].plot(
    xlabel="Дата измерения",
    ylabel="Значение запыленности",
    title="Распределение запыленности PM до 2,5 мкм по измерениям");
```

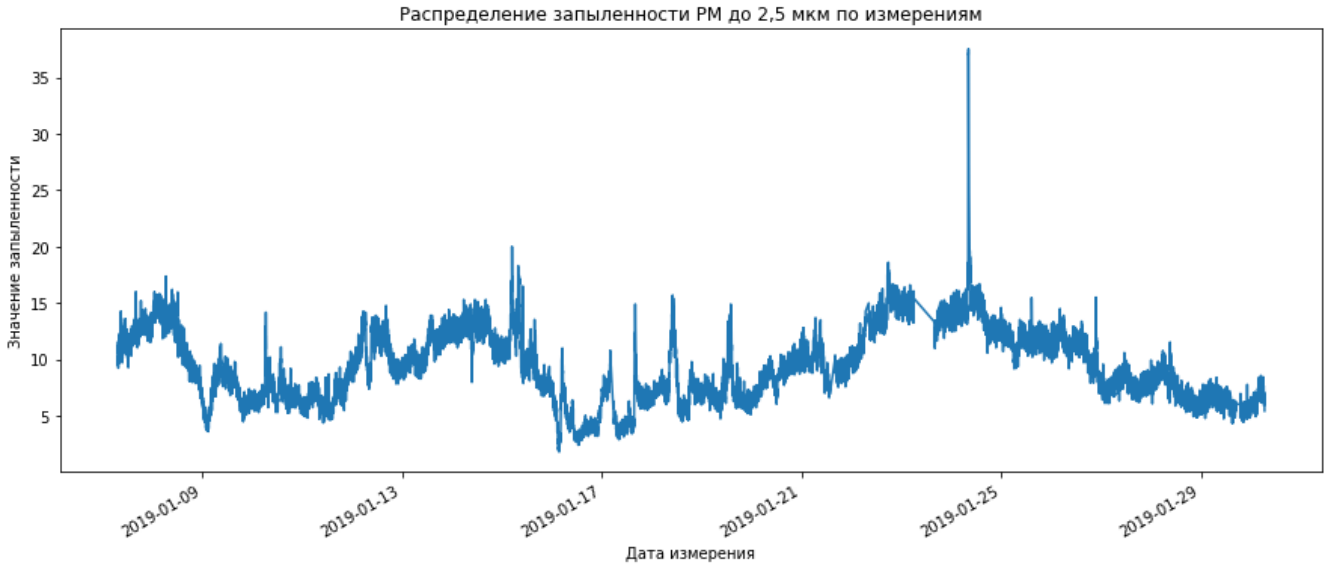

Рис. 21 Построение графика показаний датчика

Если вернуться к рис. 20 и посмотреть на метки времени, можно увидеть что измерения выполняются примерно каждые две с половиной минуты. Если мы будем строить модель, прогнозирующую значения показаний датчика на основе исторических данных, то какой прогноз нам хотелось бы иметь? Предсказывающий показания через 5 минут? Или через час, через день? Поминутный прогноз на практике мало, кому нужен. Реально максимальная детализация это почасовой прогноз. Но для того чтобы делать такой прогноз нам не нужны показания каждые две с половиной минуты. Временные серии Pandas имеют готовые механизмы пересчета для нужных нам временных промежутков. На рис. 22 показано как мы можем пересчитать показания датчиков для часовых временных интервалов с усреднением значений.

```
# Пересчет измерений для промежутков 1 час с усреднением и автоматическим заполнением нулевых значений
df = df["P2"].resample("1H").mean().fillna(method="ffill").to_frame()
```

Рис. 22 Пересчет показаний датчиков для часовых интервалов

Мы можем добавить эту строку к нашей функции обработки данных. Один столбец из датафрейма дает временную серию Pandas, методы и атрибуты временных серий и датафреймов не совпадают. Мы выполняем преобразования для временной серии и в конце преобразуем ее в датафрейм для того чтобы вернуть датафрейм, т. к. функция обработки возвращает датафрейм. На случай отсутствующих значений мы применяем метод автоматического заполнения Forward Filling.

33. Если повторно вывести тот же график, мы увидим отличия, т. к. показания датчиков усреднялись и график оказывается более сглаженным, рис. 23. Это может быть полезно для выявления тенденций, трендов в данных.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(15,6))

df["P2"].plot(
    xlabel="Дата измерения",
    ylabel="Значение запыленности",
    title="Распределение запыленности PM до 2,5 мкм по измерениям");
```

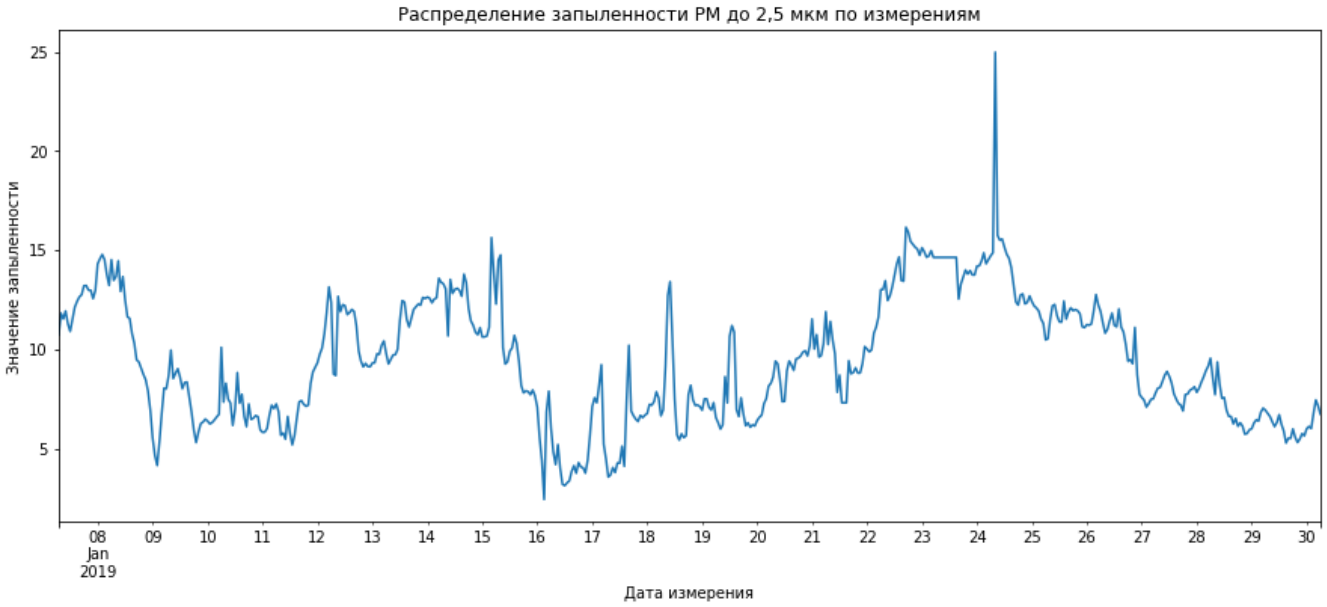

Рис. 23 Изменение графика при усреднении значений на часовых интервалах

34. Для построения модели поведения целевой функции обычно используют дополнительные характеристики, также меняющиеся во времени. Большее или меньшее влияние этих характеристик можно выявить строя корреляционные зависимости целевой функции от характеристик. Однако, даже в том случае, когда у нас есть только один ряд данных, поведение которого мы и хотели бы предсказать, и больше ничего нет, мы можем в качестве дополнительной характеристики использовать сами эти же данные, сдвинутые во времени и рассчитать функцию автокорреляции. Добавим в датафрейм столбец, повторяющий данные, измеренные датчиком, но сдвинутые на 1 шаг. Так как шаг у нас составляет один час, то это данные со сдвигом на 1 час, рис. 24. Мы видим, что первая строка при сдвиге оказывается незаполненной. При построении модели машинного обучения строки без данных приведут к ошибкам, поэтому придется удалить эту строку, рис. 25. Дополнительный столбец мы будем использовать как характеристику, атрибут наших данных для обучения модели.

```
df["P2.Lag"] = df["P2"].shift(1)
df.head()
```

timestamp	P2	P2.Lag
2019-01-07 07:00:00	10.750000	NaN
2019-01-07 08:00:00	11.834783	10.750000
2019-01-07 09:00:00	11.546818	11.834783
2019-01-07 10:00:00	11.946818	11.546818
2019-01-07 11:00:00	11.297391	11.946818

Рис. 24 Добавление в датафрейм дополнительного столбца

```
df.dropna().head()
```

timestamp	P2	P2.Lag
2019-01-07 08:00:00	11.834783	10.750000
2019-01-07 09:00:00	11.546818	11.834783
2019-01-07 10:00:00	11.946818	11.546818
2019-01-07 11:00:00	11.297391	11.946818
2019-01-07 12:00:00	10.919130	11.297391

Рис. 25 Удаление строк, не имеющих числовых значений

Проверим, есть ли корреляция между добавленным и исследуемым столбцом, рис. 26:

```
df.corr()
```

	P2	P2.Lag
P2	1.00000	0.94006
P2.Lag	0.94006	1.00000

Рис. 26 Матрица корреляции для целевого и добавленного столбцов

Построим график взаимного поведения данных в соответствующих столбцах для наглядности, рис. 27. И из матрицы корреляции, и из графика мы видим, что между данными в двух столбцах есть сильная зависимость. Это значит, что мы можем построить модель, предсказывающую значения с хорошей точностью.

```
fig, ax = plt.subplots(figsize=(6,6))
ax.scatter(x=df["P2.Lag"], y=df["P2"])
ax.plot([0, 26], [0, 26], linestyle="--", color="orange")
plt.xlabel("Значения запыленности PM2,5 сдвинутые на 1 час")
plt.ylabel("Значения PM2,5 измеренные датчиком")
plt.title("Автокорреляция между измеренными и сдвинутыми на час значениями");
```

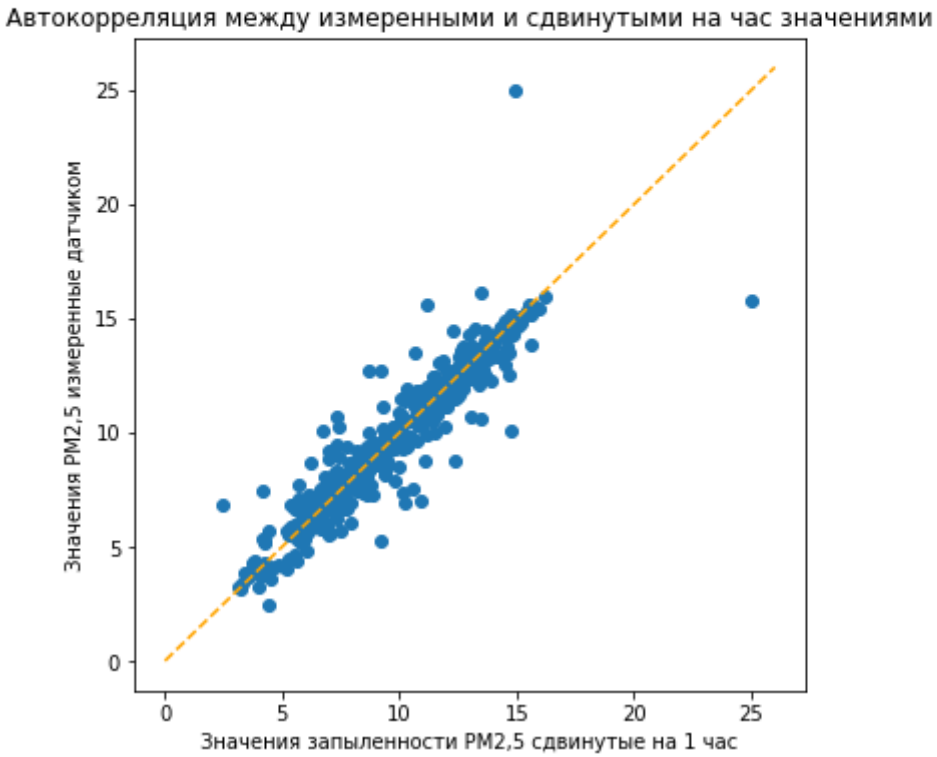

Рис. 27 Взаимозависимость добавленного столбца и исходных данных

35. Определим целевую функцию и набор данных для обучения модели, рис. 28. **y** это целевая функция, она представляет собой столбец датафрейма с данными измерений, **X** это вектор значений, которые мы используем для обучения модели. Указав имя столбца в квадратных скобках, мы выбираем только его из датафрейма, так как в датафрейме всего два столбца, отбросив столбец с целевыми значениями, мы получим второй столбец со сдвинутыми значениями.

```
target = "P2"
y = df[target]
X = df.drop(columns=target)
```

Рис. 28 Целевая функция и вектор данных для обучения

Мы должны разделить выборку данных на две части: одну будем применять для обучения модели, а вторую для ее валидации, проверки. При проверке модель должна работать с данными, которых она «не видела», иначе результат будет 100%. Деление выполняется обычно в пропорции 80% данных для обучения и 20% для валидации, рис. 29

```
cutoff = int(len(X) * 0.8)

X_train, y_train = X.iloc[:cutoff], y.iloc[:cutoff]
X_test, y_test = X.iloc[cutoff:], y.iloc[cutoff:]
```

Рис. 29 Разделение данных на тренировочную и проверочную выборки

36. Импортируем библиотеку машинного обучения. Прежде чем обучать модель нам нужно определить опорное значение, в качестве которого мы можем взять просто среднее по всем отсчетам. Если затем посчитать значение среднего отклонения для каждого отсчета (это делается автоматически специальной функцией mean_absolute_error из библиотеки машинного обучения), то мы увидим, насколько в среднем каждый отсчет в нашей выборке отклоняется от среднего значения. Это самая примитивная модель для наших данных. Построенная с помощью машинного обучения модель должна работать лучше, т. е. показывать меньшее MAE, рис. 30.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

y_pred_baseline = [y_train.mean()] * len(y_train)
mae_baseline = mean_absolute_error(y_train, y_pred_baseline)

print("Среднее значение PM2,5 по всем отсчетам", round(y_train.mean(), 2) )
print("Опорное значение среднего отклонения по выборке", round(mae_baseline, 2) )
```

```
Среднее значение PM2,5 по всем отсчетам 9.78
Опорное значение среднего отклонения по выборке 2.77
```

Рис. 30 Расчет baseline и MAE

37. Выполняем обучение модели с помощью линейной регрессии и проверку по значению среднего отклонения, рис. 31. Видим что результат значительно лучше примитивной модели. На проверочных данных он оказывается еще лучше, чем на данных обучения. Так бывает, хотя чаще результат на проверочных данных несколько хуже.

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
training_mae = mean_absolute_error(y_train, model.predict(X_train))
test_mae = mean_absolute_error(y_test, model.predict(X_test))
print("MAE для данных на этапе обучения", round(training_mae, 2))
print("MAE для проверочных данных", round(test_mae, 2))
```

```
MAE для данных на этапе обучения 0.69
MAE для проверочных данных 0.38
```

Рис. 31 Обучение модели и проверка ее качества с помощью MAE

38. Создадим датафрейм, содержащий все измеренные значения тестовой, проверочной выборки, и соответствующие значения прогноза, и выведем эти данные на график для сравнения.

```
df_pred_test = pd.DataFrame(
    {
        "y_test": y_test,
        "y_pred": model.predict(X_test)
    }
)

df_pred_test.head()
```

timestamp	y_test	y_pred
2019-01-25 16:00:00	11.900000	11.426931
2019-01-25 17:00:00	12.107917	11.762583
2019-01-25 18:00:00	11.964583	11.956888
2019-01-25 19:00:00	12.015417	11.822938
2019-01-25 20:00:00	11.960000	11.870444

Рис. 32 Создание датафрейма с двумя колонками: тестовой выборки и прогноза

```
fig, ax = plt.subplots(figsize=(15, 6))
df_pred_test.plot(ax = ax, ylabel="Измеренные значения запыленности PM2,5", title="Сравнение измеренных и предсказанных значений");
```

Рис. 33 Вывод на график значений измерений и прогноза по созданной модели

Ваша задача — взять из той же базы данных коллекцию Nairobi, проанализировать ее содержимое, выбрать локацию с наибольшим числом зарегистрированных документов и по аналогии построить для нее модель линейной регрессии. Рассчитать MAE, построить график сравнения измеренных и прогнозных значений. Для проверки работы загрузить отчет и файл блокнота Jupyter Notebook (как вариант, можно разместить файл блокнота на внешнем ресурсе — Google Colab, Github + Binder).

4. Практическая работа №4
ЗНАКОМСТВО С ОБЛАЧНЫМ СЕРВИСОМ ИНТЕРНЕТА ВЕЩЕЙ TningsBoard

Интернет вещей предполагает включение в процесс коммуникации и обмена данными миллионов устройств. Чтобы обеспечить удобный сервис управления и мониторинга большим числом разнотипных устройств, использующих зачастую различные коммуникационные технологии и протоколы, используют облачные платформы. Такие платформы позволяют накапливать информацию за длительные периоды работы, т. к. они имеют средства хранения в виде баз данных, хранилищ и озер данных. Они, как правило, содержат инструменты для обработки и визуализации накопленных данных, для представления их в графической, наглядной форме, гораздо легче воспринимаемой человеком. Широкое использование Интернета вещей совместно с облачными платформами послужило катализатором разработки и внедрения масштабируемых приложений и бизнес-моделей Интернета вещей. Облачные вычисления и Интернет вещей стали двумя очень тесно связанными интернет-технологиями будущего. Все крупные облачные провайдеры: Amazon, Microsoft, Google, в том числе отечественные Yandex, MTS и VK, предлагают свои реализации сервисов облачной поддержки технологий Интернета вещей. Например, у компании Yandex эти инструменты входят в Yandex IoT Core, у группы «Вконтакте» Cloud IoT Platform. Существует также большое число специализированных облачных сервисов, ориентированных главным образом на поддержку решений Интернета вещей, а не на большой набор универсальных сервисов, которые предлагают облачные провайдеры, например, Kaa IoT Platform, Blynk, Particle и др.

В данной лабораторной работе мы познакомимся с популярным программным решением, облачной платформой TningsBoard. Платформа предлагается как законченное программное решение, которое можно использовать как сервис подписки в облаке или развернуть самостоятельно на своих серверах. При самостоятельном развертывании предлагается два варианта: бесплатное OpenSource решение Community Edition, CE, или платная версия Proffeccional Edition, PE. Официальный сайт https://thingsboard.io сейчас заблокирован для доступа из России (при этом облачный сервис https://thingsboard.cloud/login остается доступным), но документацию по продукту и его код можно загрузить с Github без ограничений https://github.com/thingsboard/thingsboard

ВЫПОЛНЕНИЕ РАБОТЫ

Используем уже знакомый вам прием развертывания сервиса из подготовленного образа контейнера. Скачайте образ виртуальной машины по ссылке https://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-20.04.1-legacy-server-amd64-ovf.zip [4]. Распакуйте из архива и импортируйте образ в VirtualBox.

Обновите базы пакетов командой:

```
sudo apt update
```

и установите сервис docker и docker-compose командой:

```
sudo apt install docker.io docker-compose
```

Мы создадим программный скрипт на языке YAML, который запустим с помощью docker-compose — программного средства автоматизированного развертывания набора сервисов на одном компьютере. Это часть программных инструментов семейства Docker. Но, прежде чем мы начнем работать с Docker Compose, подготовим на нашем компьютере

служебные каталоги. Облачная платформа работает с данными. Если мы будем запускать ее как сервис в Docker-контейнере, то при остановке контейнера все данные будут потеряны. Чтобы данные сохранялись на основном компьютере, создадим каталог для данных в домашней папке и зададим на него правильные права доступа:

```
mkdir -p ~/.mytb-data && sudo chown -R 799:799 ~/.mytb-data
```

Команда mkdir создает каталог с заданным именем. Если имя каталога в Linux начинается с точки, то он получает атрибут «Скрытый» и обычно используется для хранения служебных файлов. Ключ -p сокращение от Path, показывает по какому пути создать новый каталог. Если мы не указываем этот ключ, то новый каталог создается внутри текущего. В данном случае в качестве пути указан знак тильды ~, этим знаком в Linux обозначается домашний каталог пользователя. Двойной знак амперсанда && позволяет создать последовательную цепочку команд. Если первая команда (создания директории) будет выполнена успешно, то затем выполняется вторая команда, идущая после знаков амперсанда. Команда chown (Change Owner) позволяет изменить владельца указанного файла или каталога. Вместе с командой дается ключ -R, указывающий что команда будет распространяться рекурсивно и на все файлы внутри каталога, в том числе вновь создаваемые. Цифры 799 это идентификатор пользователя, через двоеточие указан идентификатор группы, в которую входит пользователь. Этот идентификатор создатель контейнера Thingsboard использовал при его создании и если не сменить владельца, то сервис контейнера не сможет получить доступ к смонтированному тому из основной машины. Аналогично меняется владелец для каталога лог-файлов:

```
mkdir -p ~/.mytb-logs && sudo chown -R 799:799 ~/.mytb-logs
```

В домашнем каталоге создайте новый отдельный каталог для выполнения работы. Например, так:

```
mkdir lab4
```

Перейдите в него командой:

```
cd lab4
```

С помощью текстового редактора Vi создайте файл **docker-compose.yaml** (имя файла должно быть именно таким, это важно! Расширение может быть **yml** или **yaml**) следующего содержания:

```
version: "3"

services:
  thingsboard:
    image: "thingsboard/tb-postgres"
    container_name: mytb
    ports:
      - "8080:9090"
      - "7070:7070"
      - "1883:1883"
      - "5683-5688:5683-5688/udp"
    volumes:
      - ~/.mytb-data:/data
      - ~/.mytb-logs:/var/log/thingsboard
    restart: always
```

Docker-compose использует пары ключ-значение для задания параметров. Значение может быть задано не одним параметром, но множеством. В этом случае применяются

списки, каждый элемент которых начинается с новой строки и начинается с дефиса, как в случае портов или подключаемых томов. Важное значение имеют отступы, они формируют структуру конфигурационного файла, подобно языку Python. Например, мы определили сервис с конкретным именем thingsboard в числе сервисов, которые будут запускаться Docker-compose. Этот конкретный сервис должен быть «внутри» списка сервисов, а значит у названия сервиса должен быть отступ, по сравнению с ключом services. Все параметры сервиса thingsboard должны иметь отступ уже от позиции самого имени сервиса — thingsboard. Спецификация YAML говорит что для задания структуры «вложенная» строка должна иметь больше отступов, чем «родительская», конкретное значение отступов строго не определено, но часто используют два пробела (не табуляцию!).
Если вы открыли файл командой:

vi docker-compose.yaml

Вы попадаете в командный режим редактора vi, переключение в режим редактирования выполняется нажатием клавиши «i» на клавиатуре, переключение обратно в командный режим нажатием клавиши «Esc». Многие команды в командном режиме печатаются в окне редактора, при этом начинаются они с двоеточия. Вы можете переключиться в режим редактирования и ввести показанный выше текст строка за строкой. Можно из командного режима задать команду:

:set paste

Выделить мышкой и скопировать в буфер памяти текст, который вы хотите вставить в файл, затем перейти в окно редактора, щелкнуть правой клавишей мышки и выбрать «Вставить», содержимое вставится в редактируемый файл. Можно также после копирования для вставки использовать комбинацию клавиш «Shift + Insert». Для сохранения файла с измененным содержимым нужно из командного режима выполнить команду w - «Записать» и q - «Выйти»:

:wq

Если перед этим вы находились в режиме редактирования, т. е. вводили текст построчно, то сначала нужно нажать «Esc» чтобы вернуться в командный режим.

После сохранения файла запустите сценарий командой:

sudo docker-compose up

Эта команда ищет в текущем каталоге файл docker-compose.yaml и выполняет записанные в нем инструкции. После запуска сервиса подключитесь к нему через веб-интерфейс, доступный на порту 8080. Наберите в браузере адрес виртуальной машины с сервисом Docker и укажите порт 8080. Вы должны увидеть окно входа, рис. 1

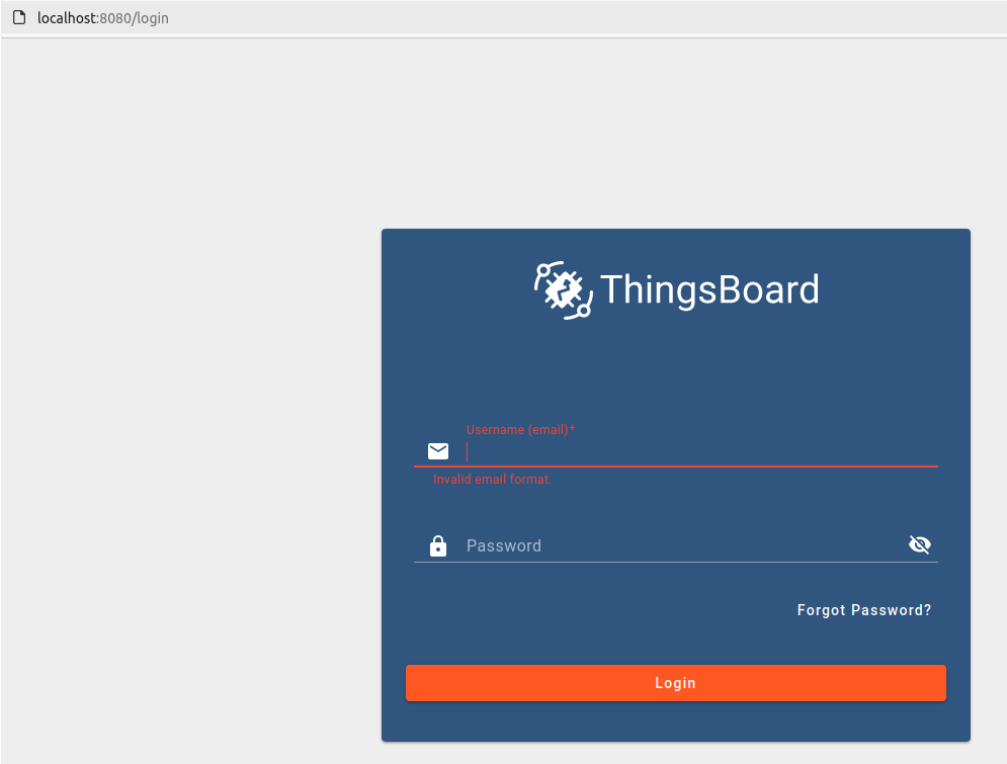

Рис. 1 Окно входа в веб-интерфейс ThingsBoard

Thingsboard CE по умолчанию имеет трех пользователей с разными уровнями привилегий: admin, tenant и customer. Admin это суперпользователь, имеющий наивысший уровень привилегий и способный создавать новые tenants, это как-бы «домовладения». Пользователь tenant, имеет такие же права, как admin, но в пределах одного, конкретного tenant. Customer не может ничего создавать и настраивать, но может просматривать уже настроенные устройства и дашборды. Войдите в систему с логином tenant@thingsboard.org и паролем tenant. Окно системы после входа выглядит так:

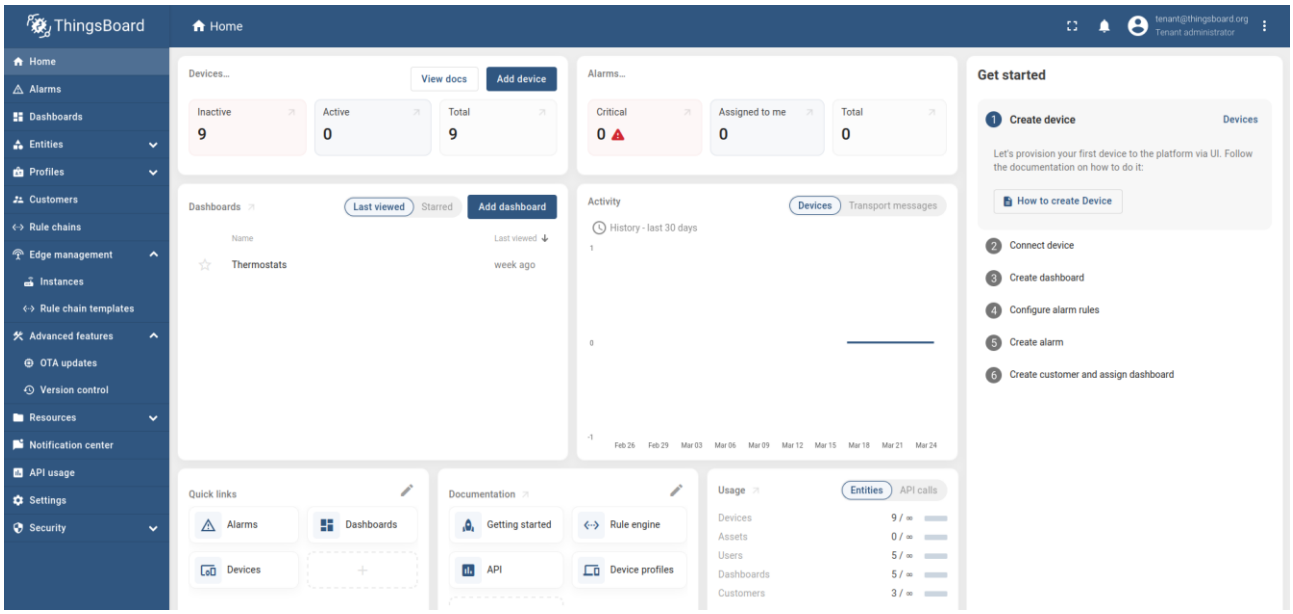

Рис. 2 Окно Thingsboard при входе пользователя tenant

Здесь уже есть несколько созданных приборов и дашбордов для демонстрации, но будем считать, что ничего нет. Создадим новое устройство. Устройство на облачной платформе это, разумеется, некоторое «представление» реального устройства, находящегося где-то, возможно очень далеко. В любом случае создание устройства это определенная

процедура, приводящая к появлению нового объекта на облачной платформе.

Добавим устройство, которое будет передавать на платформу ThingsBoard следующие данные: имя устройства и показания температуры в виде телеметрии. Чтобы добавить новое устройство, выполните следующие действия:

- перейдите в раздел «Объекты» (Entities). Затем перейдите на страницу «Устройства» (Devices), рис. 3;

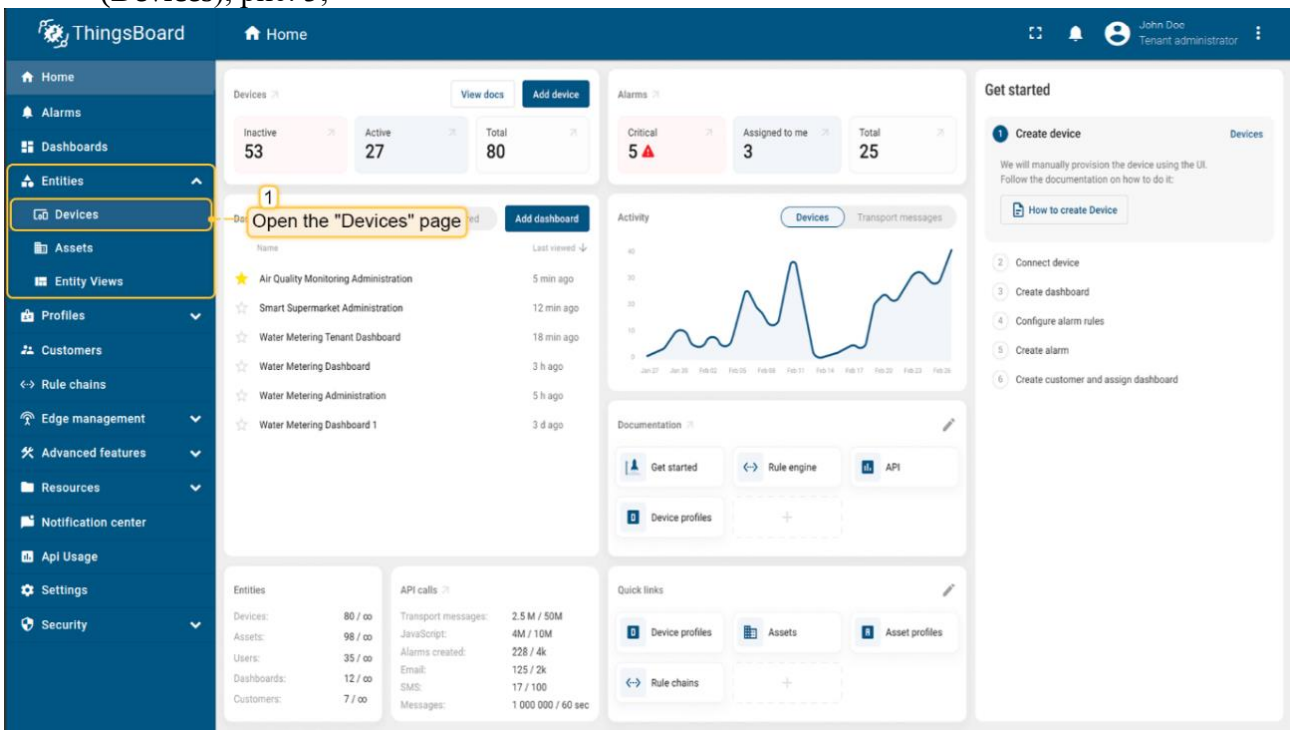

Рис. 3 Меню «Устройства»

Нажмите на значок «+» в правом верхнем углу и выберите «Добавить новое устройство», рис. 4;

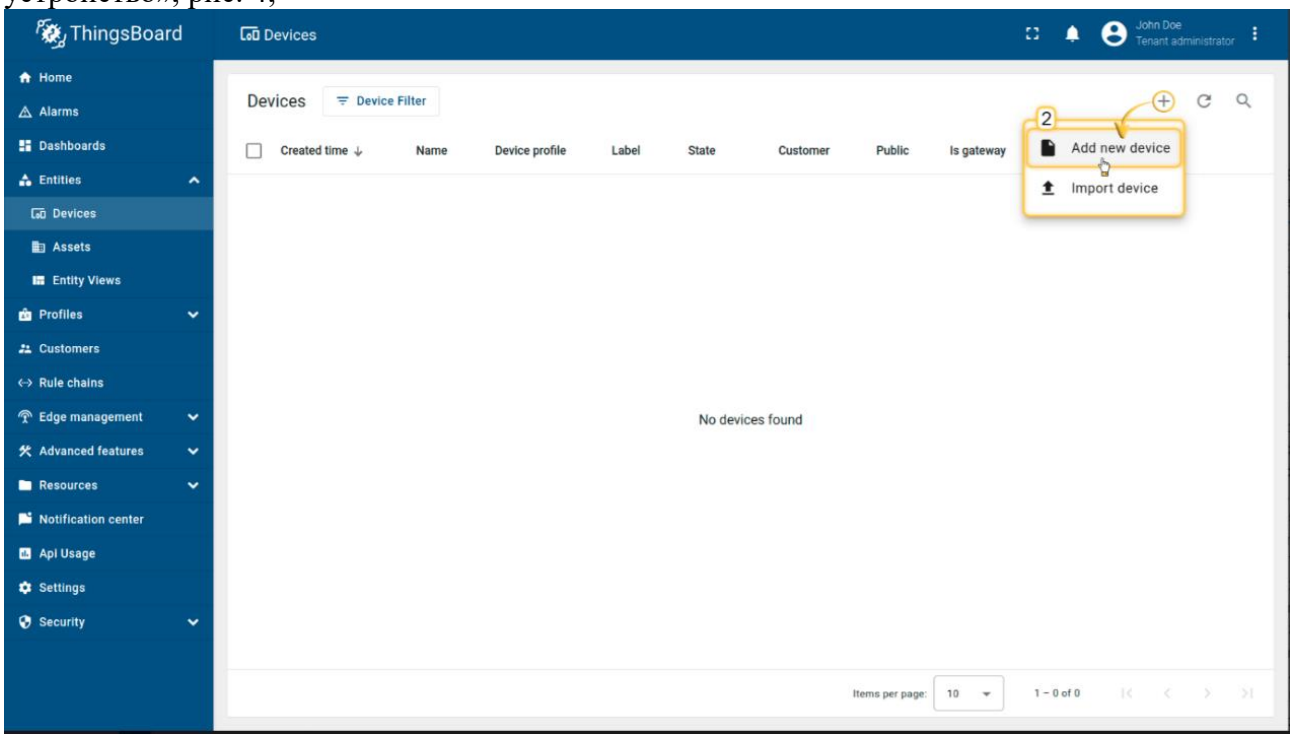

Рис. 4 Добавление устройства

В вашем случае список устройств не будет пустым, не обращайте на это внимания.

- Задайте имя устройства в форме: «устройство+ФИО». Никаких других изменений на данный момент не требуется. Нажмите «Добавить», Add, чтобы добавить устройство, рис.5;

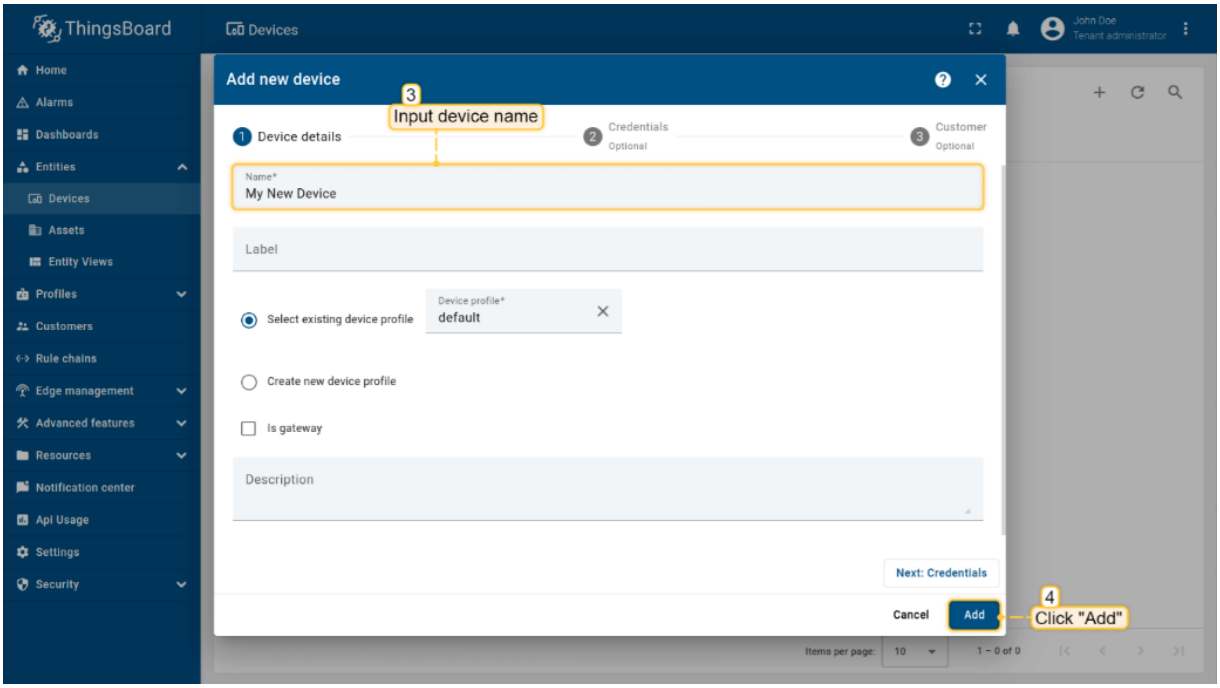

Рис. 5 Задание имени устройства

Устройство создано. Откроется окно, в котором вы сможете проверить подключение устройства к ThingsBoard. Этот шаг не является обязательным. Давайте пока закроем это окно и вернемся к проверке подключения позже, рис. 6;

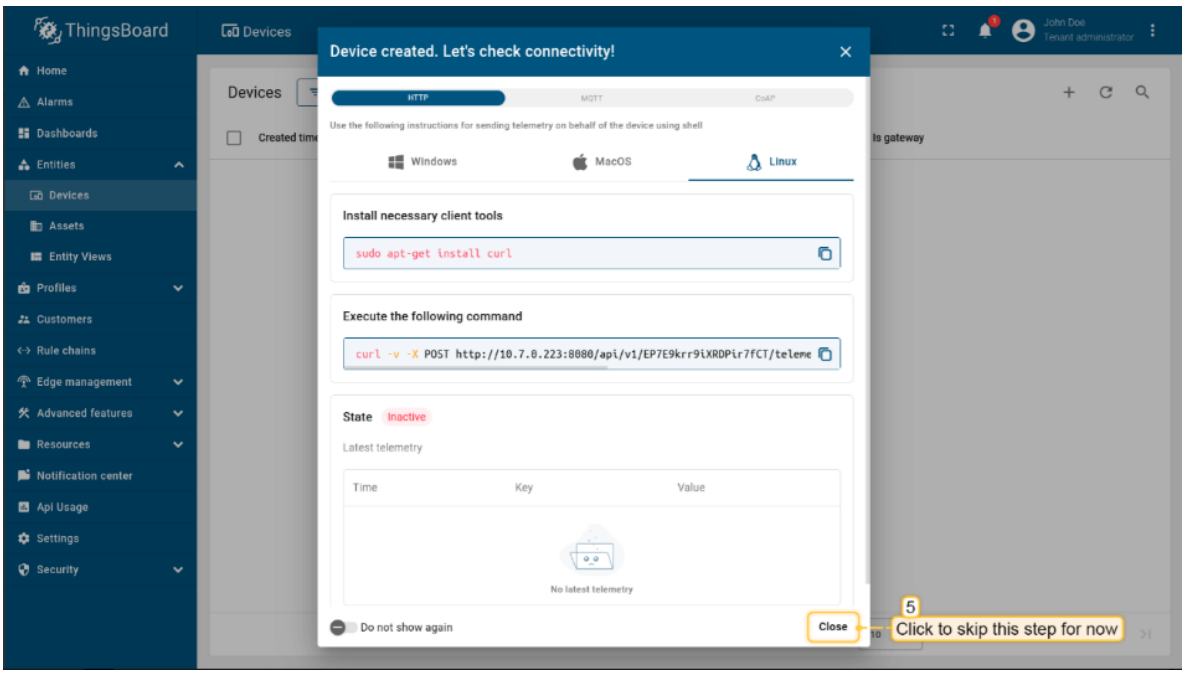

Рис. 6 Окно предложения проверить подключение

После закрытия окна вы увидите список устройств с вашим, вновь созданным устройством вверху списка. По мере добавления новых устройств они добавляются в начало таблицы, поскольку по умолчанию сортировка работает по времени создания, рис. 7.

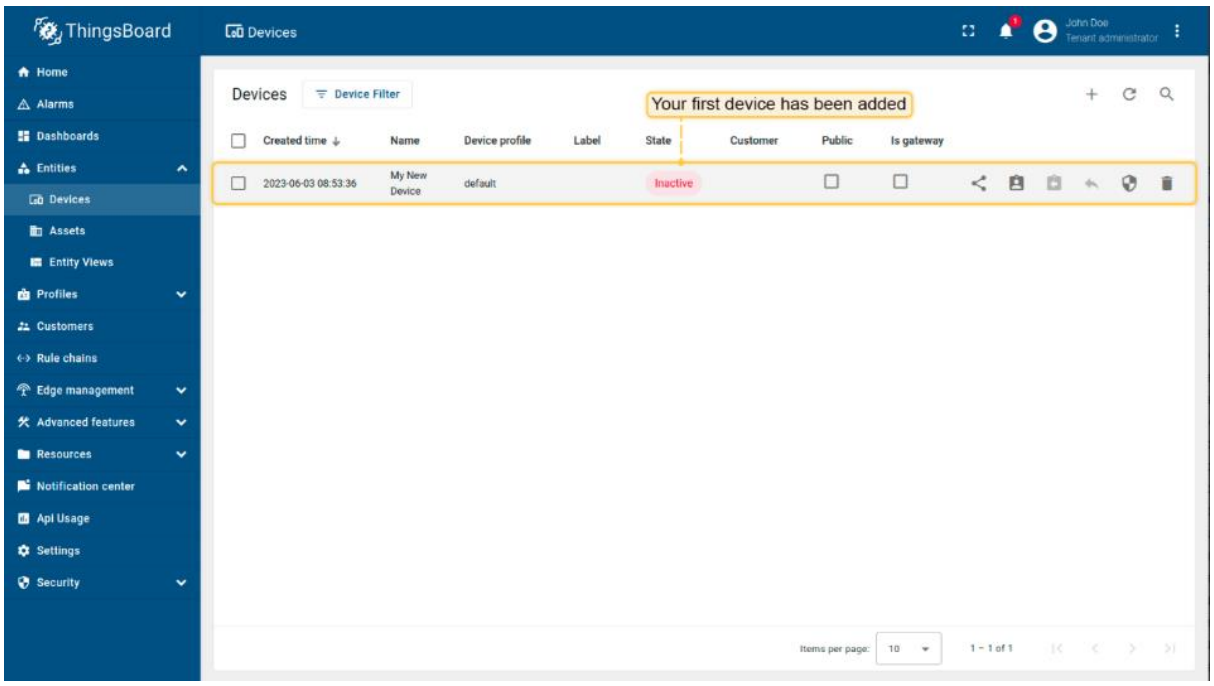

Рис. 7 Новое устройство будет вверху списка

Добавление нового устройства на платформе это системное событие, поэтому при добавлении нового устройства генерируется уведомление. Посмотреть его можно, нажав на значок «колокольчик» в правом верхнем углу, рис. 8.

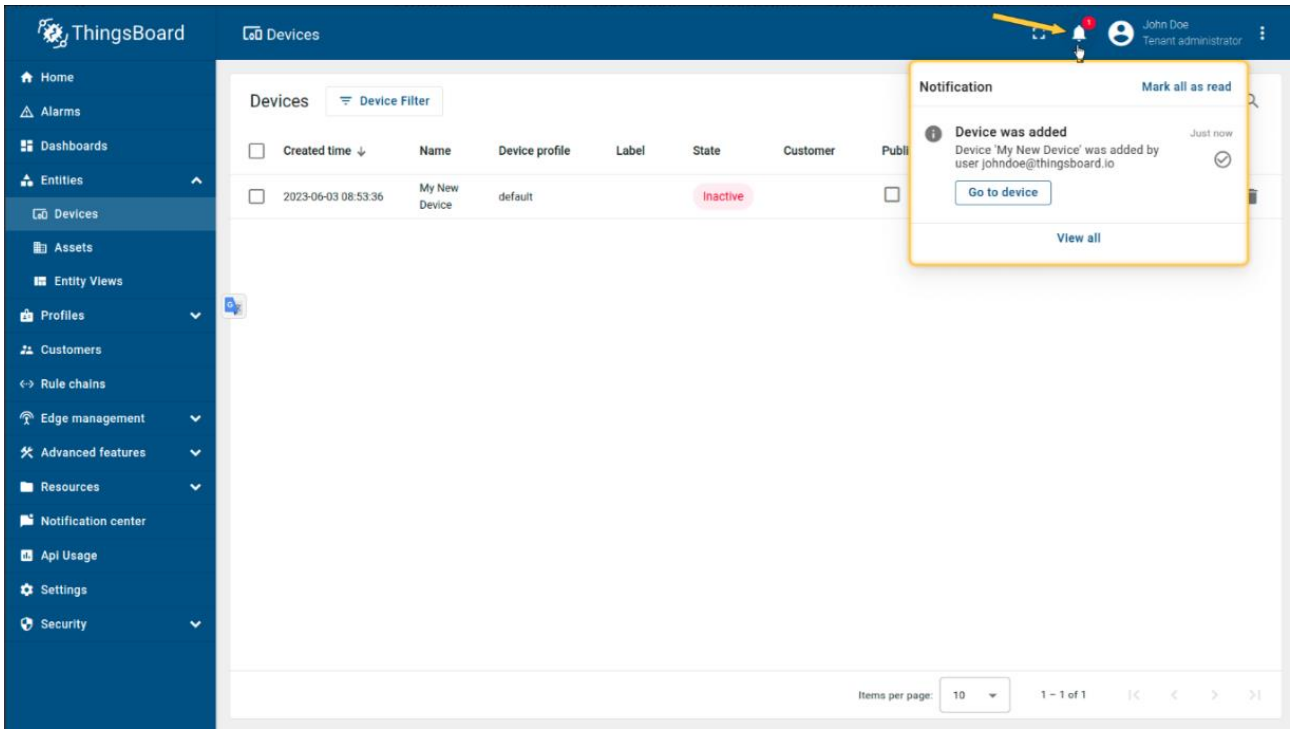

Рис. 8 Оповещение при добавлении нового устройства

Теперь проверим подключение нашего устройства к платформе ThingsBoard. Для этого используйте функцию «Проверка подключения», чтобы публиковать данные телеметрии (например, показания температуры) от имени вашего устройства. Это можно сделать как во время добавления устройства, так и после. Нажмите на свое устройство и в окне «Сведения об устройстве» (Details) нажмите кнопку «Проверить подключение», рис. 9;

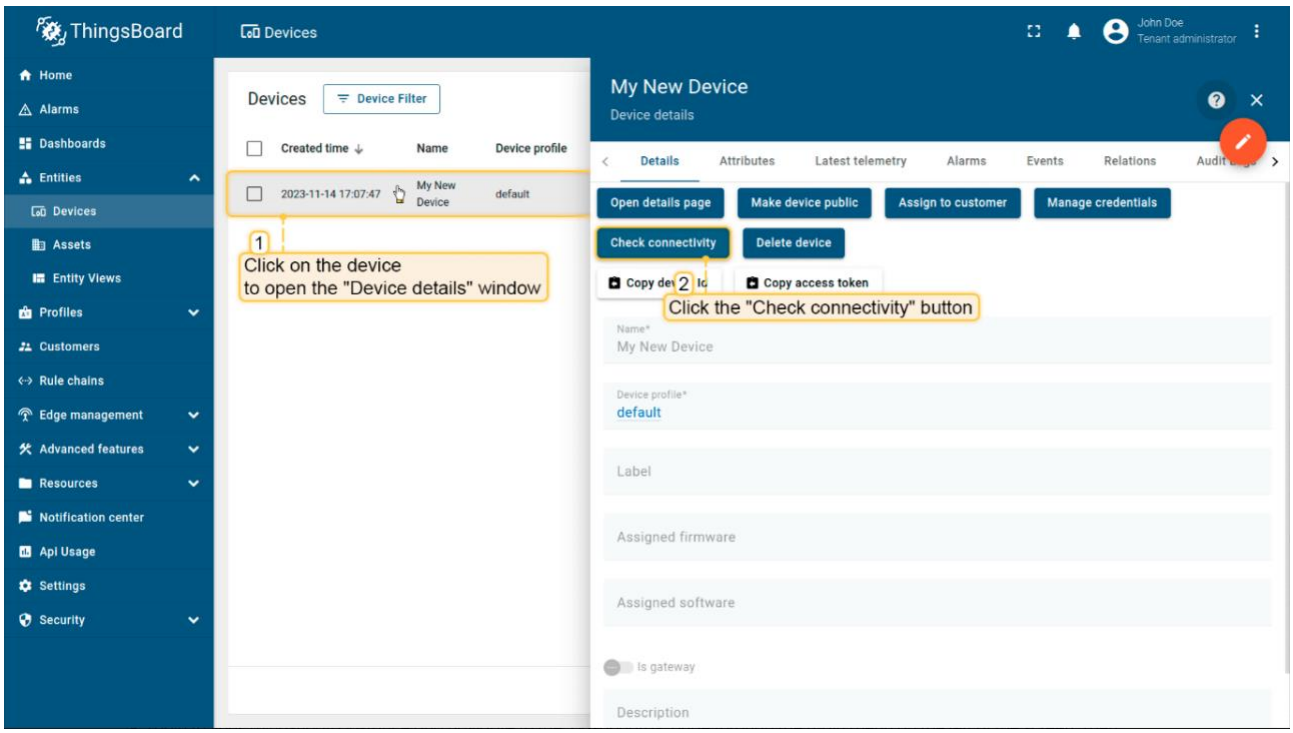

Рис. 9 Проверка подключения устройства

В открывшемся окне выберите протокол MQTT и вашу операционную систему (Linux). Установите необходимые клиентские инструменты - mosquitto-clients и скопируйте команду, рис. 10;

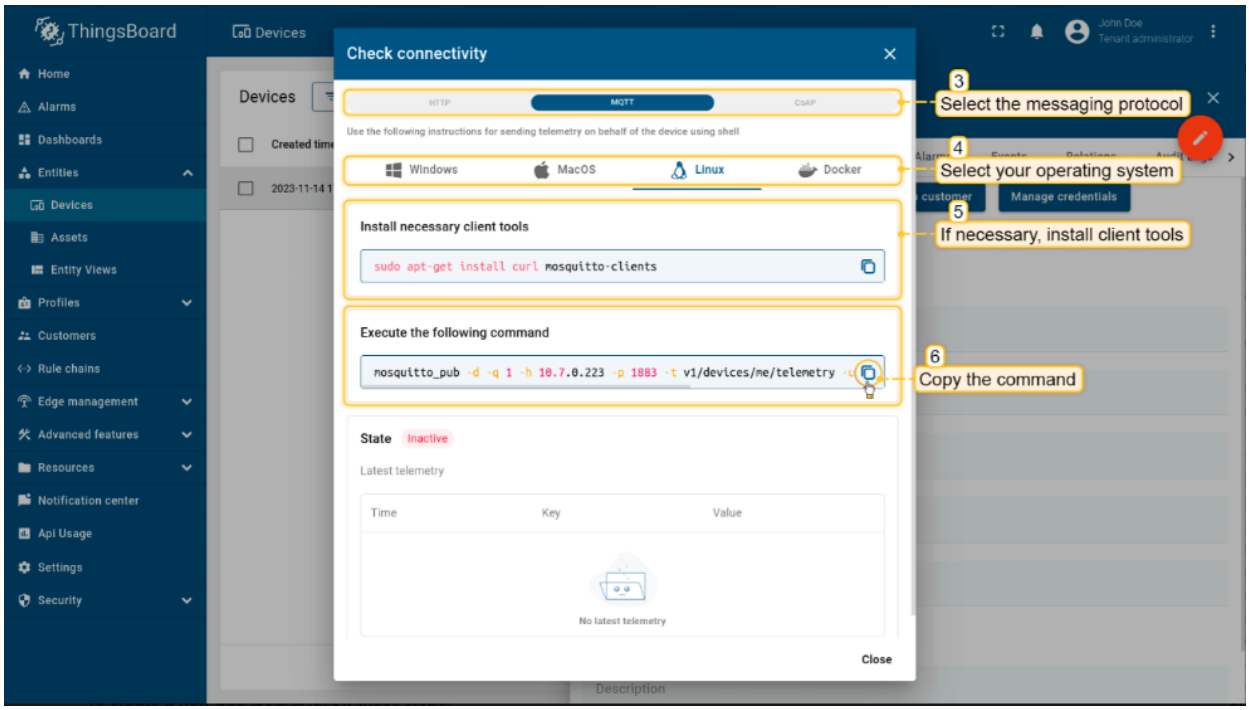

Рис. 10 Выбор протокола и команды, симулирующей отправку данных телеметрии

Команда формируется в виде:

mosquitto_pub -d -q 1 -h localhost -p 1883 -t v1/devices/me/telemetry -u "JoFMZ9slB7H0UD6OGYPB" -m "{temperature:25}"

Здесь утилита mosquitto «комарик», работает по протоколу MQTT. В протоколе MQTT предусмотрено существование трех типов устройств: брокер, издатель и подписчик. mosquitto_pub реализует функции издателя и может публиковать сообщения на брокере. Ключ -d включает режим отладки и вы будете видеть в консоли этапы процесса коммуникации, ключ -q устанавливает параметр качества сервиса. В протоколе предусмотрено три уровня качества. Самый высокий — 2, гарантирует доставку, самый низкий 0, отправляет сообщения без подтверждений со стороны брокера. Уровень 0 будет задан по умолчанию. В данном случае качество сервиса 1. Ключ -h задает адрес хоста с которым устанавливается соединение. Если вы подключаетесь к виртуальной машине с сервисом Docker клиентом из операционной системы Windows, обратите внимание на то какой IP-адрес или имя компьютера указать, чтобы ваши запросы достигали цели. Ключ -p задает порт по которому отправляется запрос. Порт 1883 брокер MQTT прослушивает по умолчанию. Это один из портов, который открывался нами для доступа в сценарии Docker Compose. Ключ -t задает топик, тему на брокере, в которую будет отправлено сообщение. Ключ -u определяет имя пользователя в протоколе MQTT, но в случае с сервисом Thingsboard здесь указан уникальный токен, который формируется для каждого созданного на платформе устройства. Ключом -m задается передаваемое сообщение. Оно состоит из пары ключ:значение. Ключ описывает параметр, в данном случае температуру, а значение показывает температуру в градусах цельсия. После того как вы выполните команду, показания температуры отобразятся в окне контроля связи, состояние устройства должно измениться с «Неактивного» на «Активное», рис. 11.

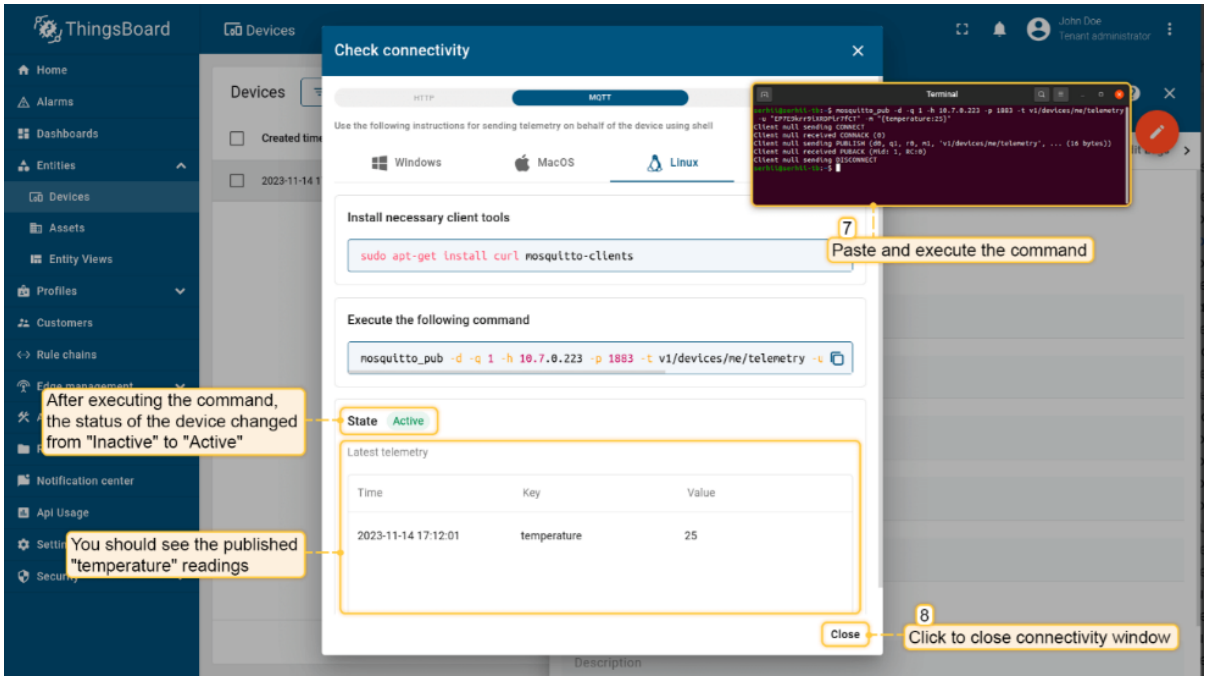

Рис. 11 Изменение статуса устройства после успешной отправки сообщения

Теперь закройте окно подключения. Последние результаты отправки данных телеметрии для устройства видны также на отдельной вкладке Latest telemetry рис. 12

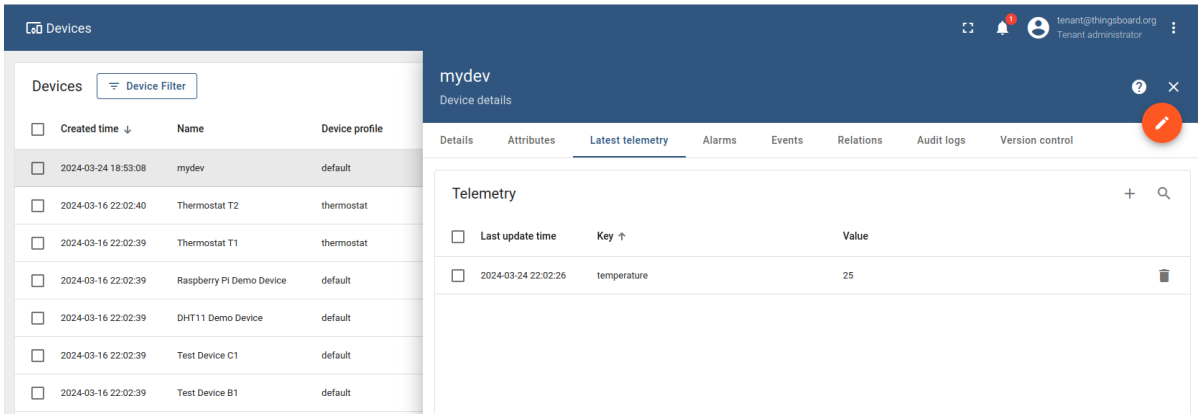

Рис. 12 Результаты отправки данных телеметрии на специальной вкладке в свойствах устройства

Просматривать полученные с датчика данные в свойствах устройства неудобно. Платформа позволяет создавать красочные дашборды и соединять их с источниками данных. Создадим такой дашборд для симулируемого датчика температуры. Процесс создания дашборда состоит из нескольких этапов. Откройте меню «Дашборды» Затем нажмите знак «+» в правом верхнем углу экрана и в раскрывающемся меню выберите «Создать новый дашборд», рис. 13;

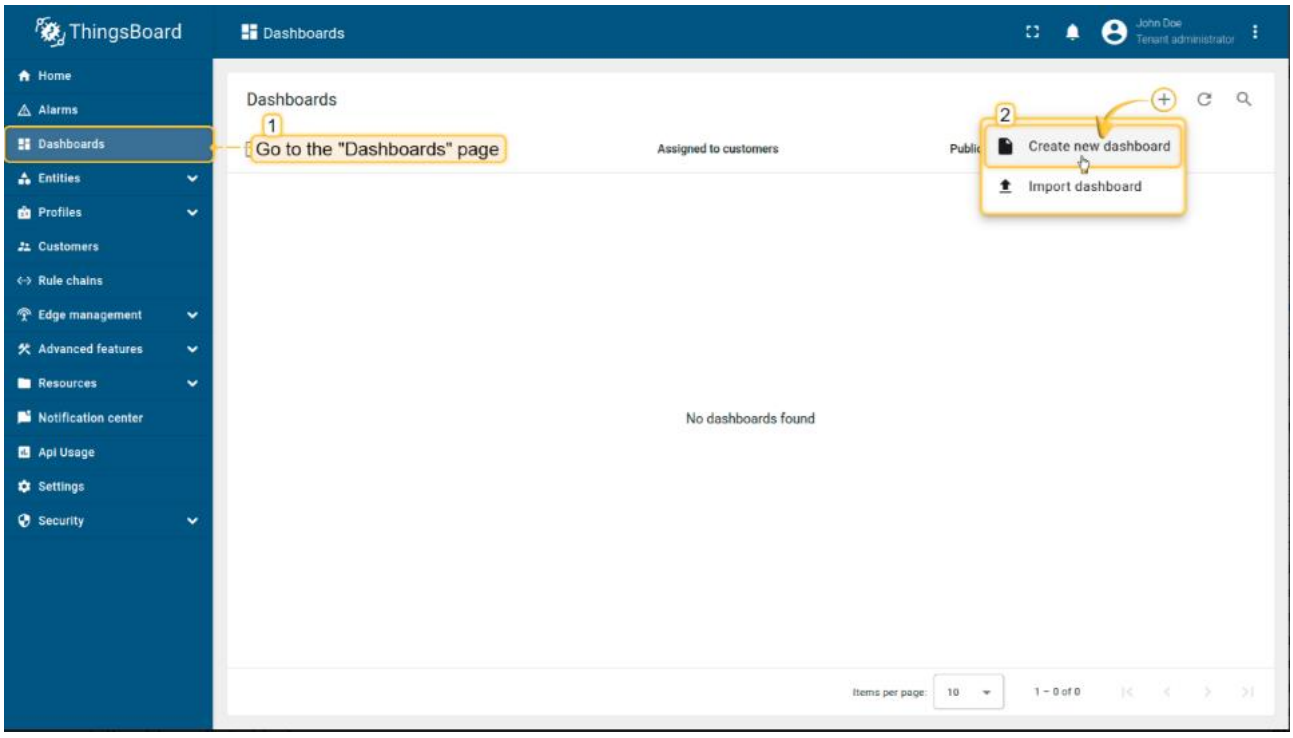

Рис. 13 Создание новой панели мониторинга (Дашборда)

В открывшемся диалоге необходимо ввести название дашборда, описание необязательно. Нажмите «Добавить», рис. 14;

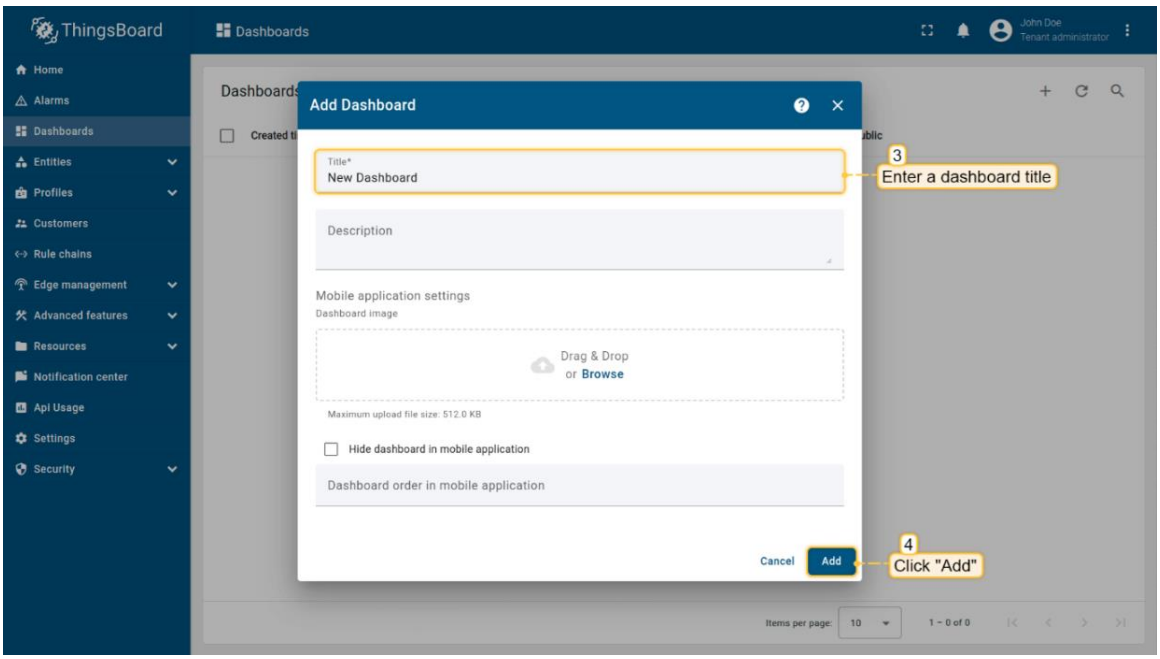

Рис. 14. Название — обязательная часть настройки дашборда

После создания дашборда он откроется автоматически, и вы сразу сможете приступить к добавлению в него виджетов. Для сохранения созданного дашборда нажмите кнопку «Сохранить» в правом верхнем углу, рис. 15;

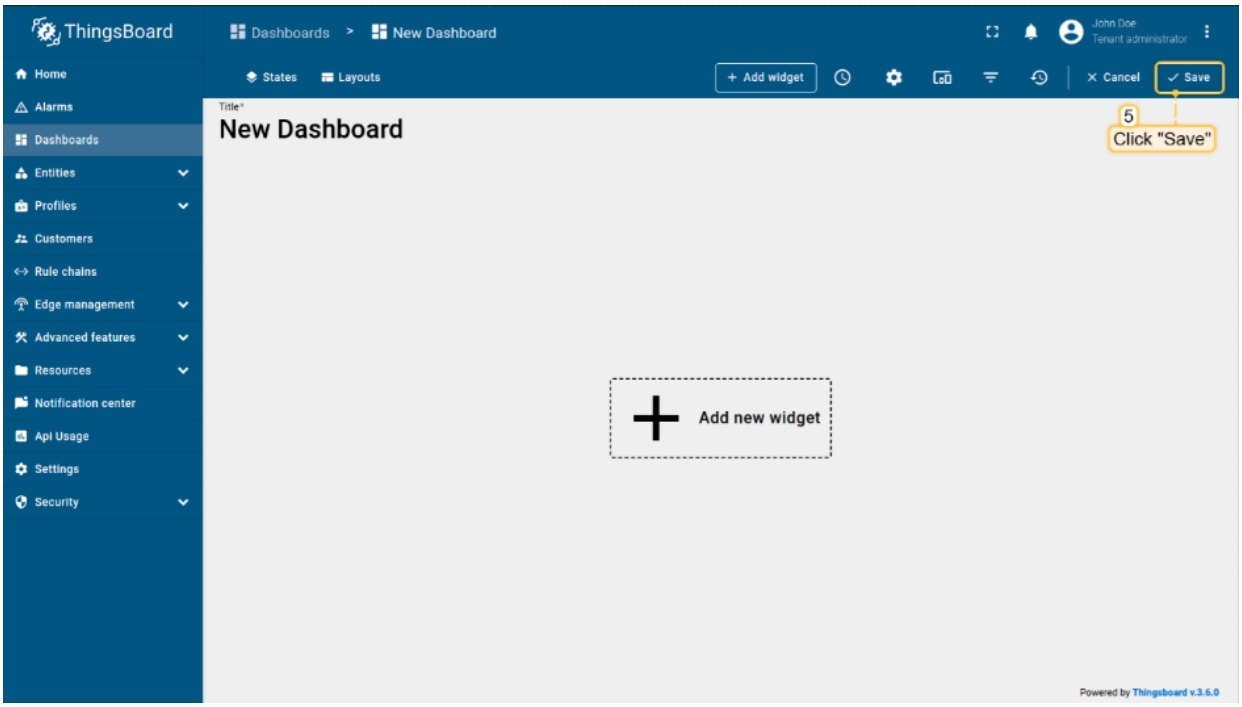

Рис. 15 Сохранение созданного дашборда

Ваша первая панель мониторинга успешно создана, правда пока абсолютно пустая. Если перейти в меню «Дашборды», вы увидите список панелей мониторинга, подобно тому, как это было со списком устройств. Новые панели мониторинга, при создании будут появляться в верхней части списка. Дашборд наполняется виджетами, а конкретный виджет вы соединяете с источником данных. Добавим виджет в виде таблицы. Для этого нужно выбрать его из библиотеки виджетов. Виджеты сгруппированы в однотипные наборы. Если вы закрыли созданный дашборд, то нужно выбрать его в списке дашбордов и перейти в

режим редактирования. Если вы находитесь на этапе, показанном на рис. 15, можно нажать на большой плюс в центре, рис. 16.

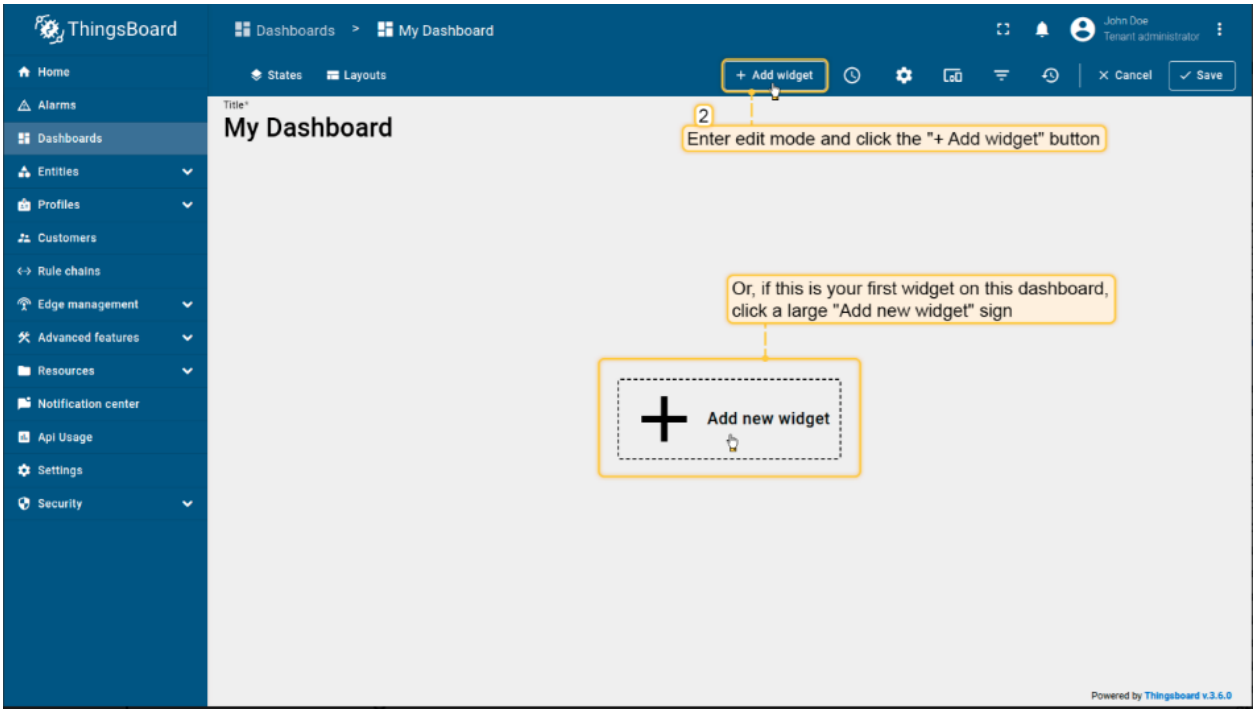

Рис. 16 Добавление виджета, первый шаг — открытие библиотеки виджетов

Выберите в библиотеке виджетов группу табличных виджетов, рис. 17

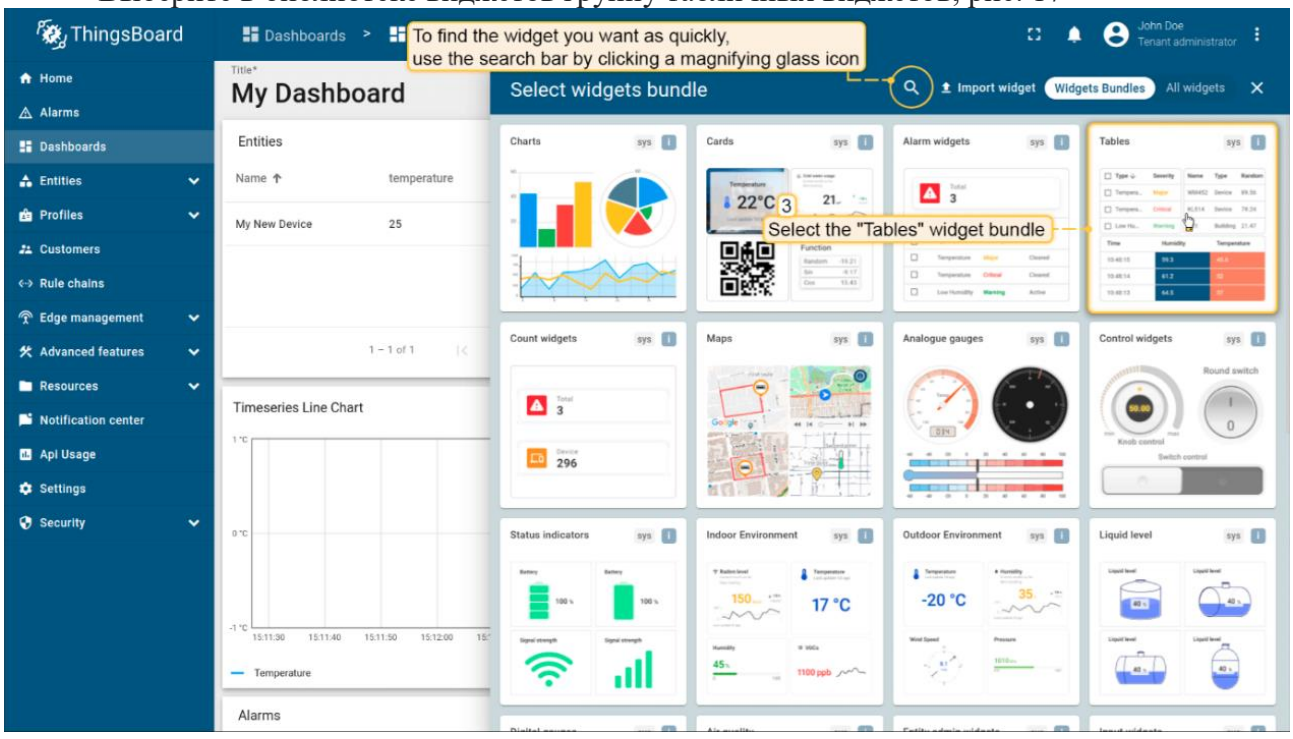

Рис. 17 Библиотека виджетов с группами по типам

Среди табличных виджетов выберите виджет «Таблица сущностей», Entities Table, рис. 18

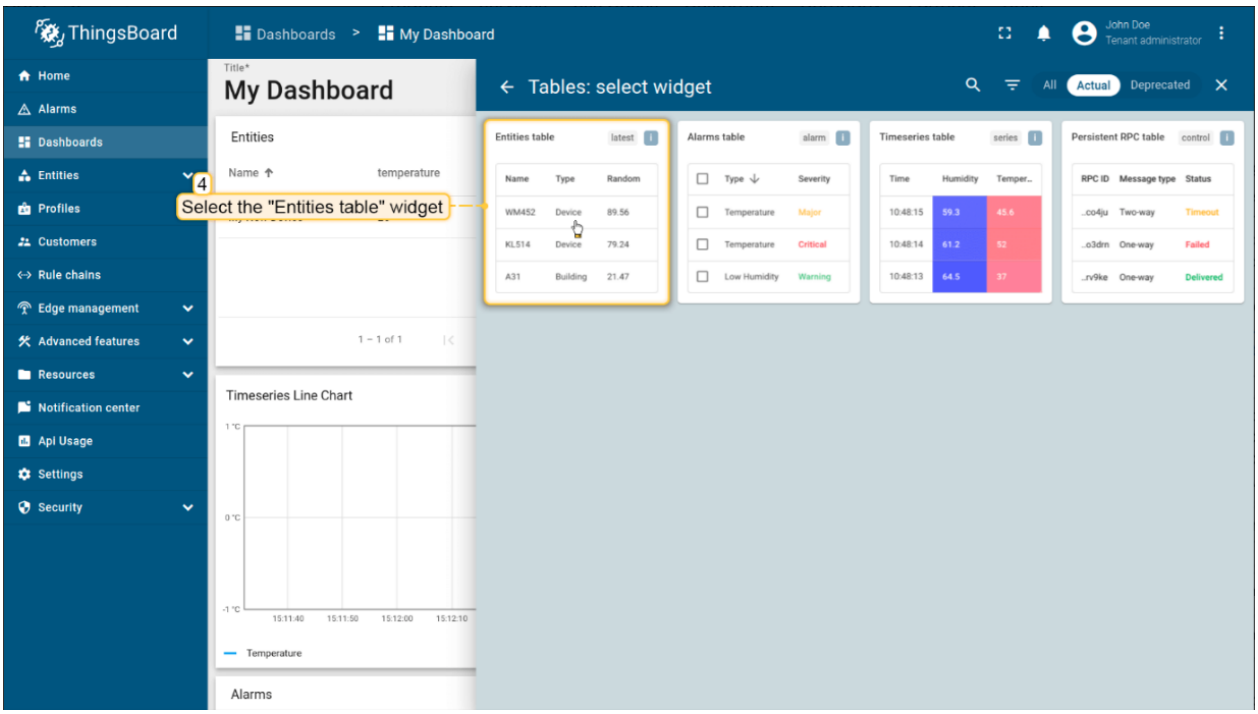

Рис. 18 Выбор виджета «Таблица сущностей»

Когда вы выбираете конкретный виджет, открывается окно его свойств, рис. 19. В поле «Источник данных», Datasource щелкните мышкой и выберите из списка то устройство, которое ранее создали. Ключевое поле name уже создано в виджете, в него выводится имя устройства, в том виде, как вы его задали.

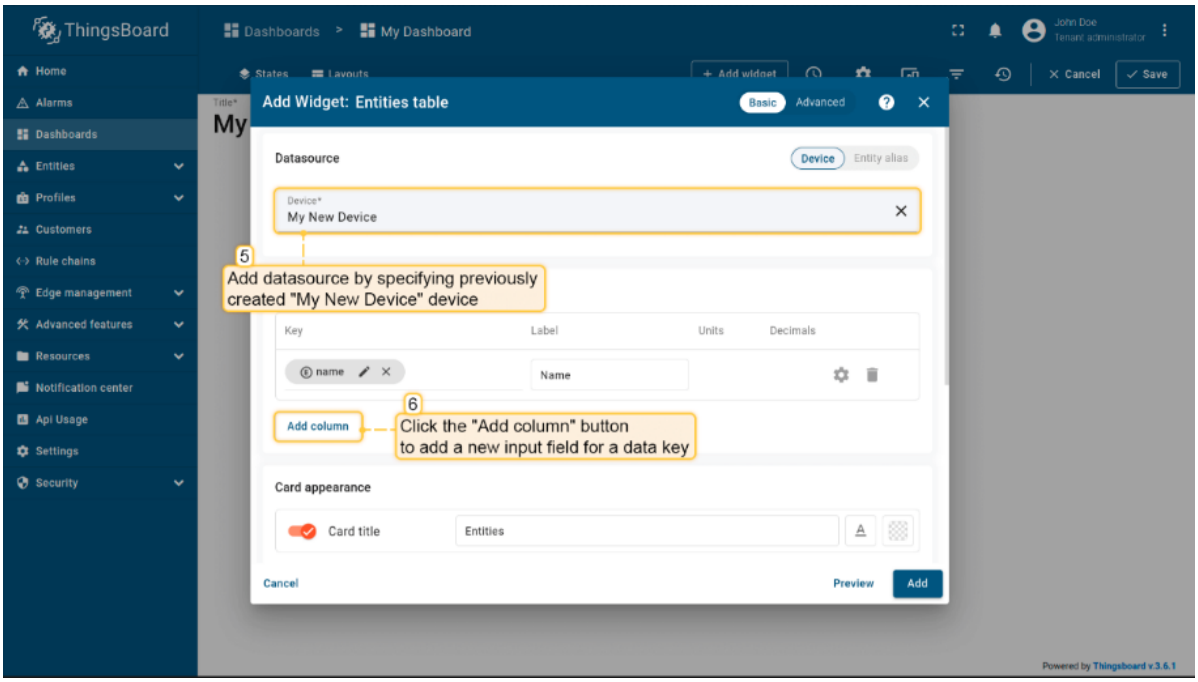

Рис. 19 Настройка параметров виджета, связь с устройством

Добавьте еще одно поле, нажатием на клавишу Add column в открывшемся дополнительном поле щелкните мышкой и из выпадающего списка выберите показания температуры, рис. 20.

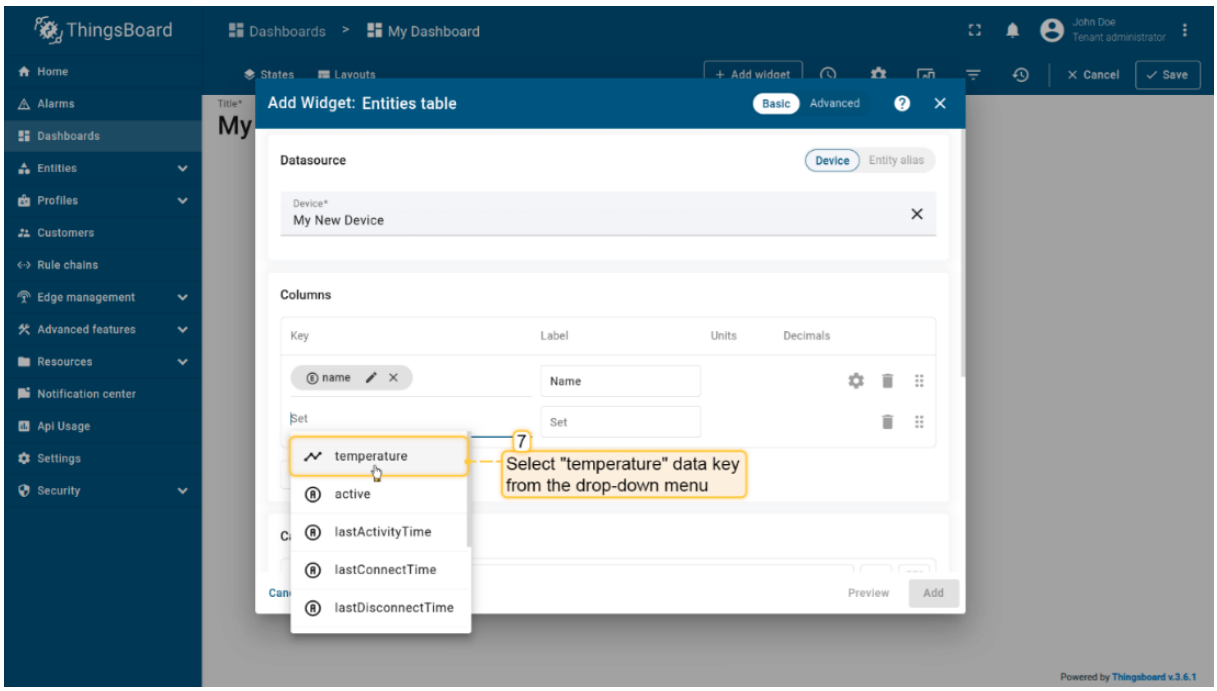

Рис. 20 Добавление поля для вывода показаний температуры

Нажатием кнопки Add в правом нижнем углу окна настройки виджета завершите конфигурирование, вы получите виджет в том виде, как он будет отображаться на веб-странице, рис. 21. Если хотите, можно изменить его размер, подведя курсор к правому нижнему углу и протянув с нажатой левой кнопкой мышки. После чего сохранить настройки можно щелкнув кнопку Save в правом верхнем углу.

В нашем виджете «Таблица сущностей» два столбца. В первом столбце отображается имя устройства, а во втором — значение ключа «температура». Вы можете вновь отправить показания телеметрии тем же способом, изменив значение температуры. Новое значение сразу появится в таблице дашборда.

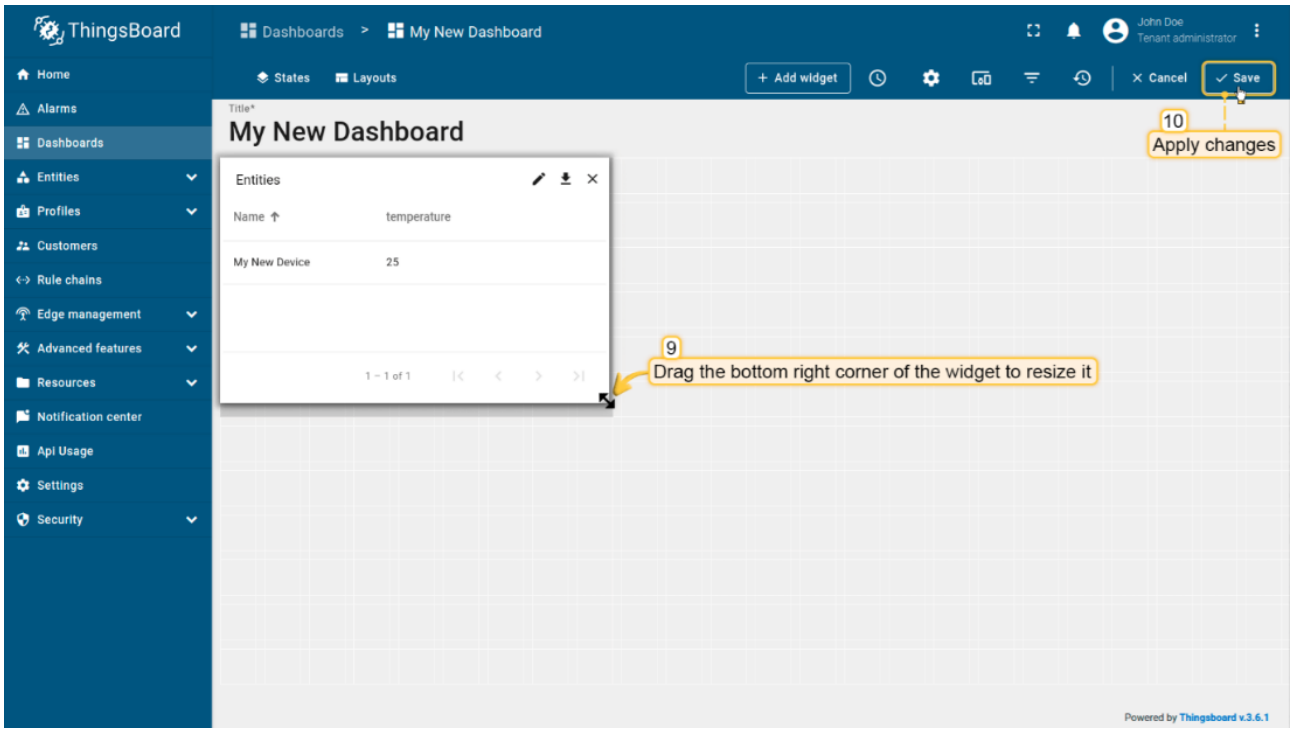

Рис. 21 Изменение размера виджета

Добавим виджет другого типа для того же источника данных. Сейчас, когда в вашем дашборде уже есть один виджет, по центру нет кнопки с плюсом. Кнопка добавления виджета есть вверху окна: «+ Add widget», нажмите ее и в окне библиотеки виджетов выберите группу «графики», рис. 22.

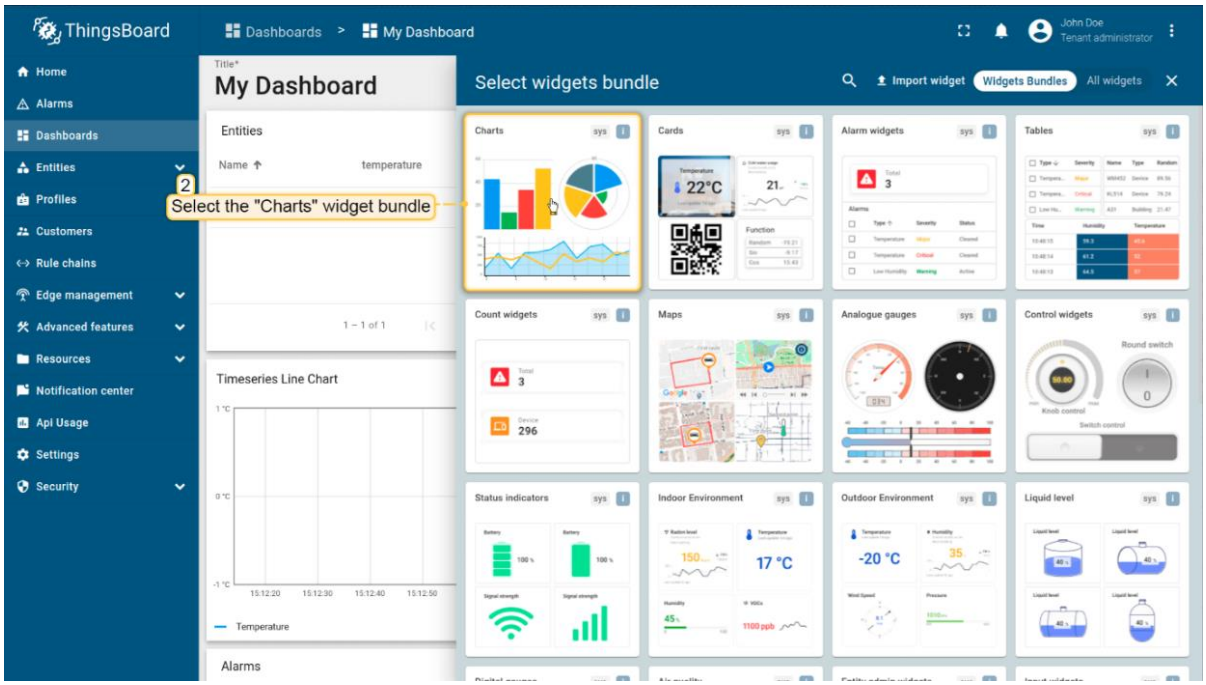

Рис. 22 Выбор группы виджетов «графики»

В этой группе выберите виджет обычного графика изменения параметра во времени, рис. 23.

Настройте параметры подобно тому, как вы это делали с виджетом таблицы.

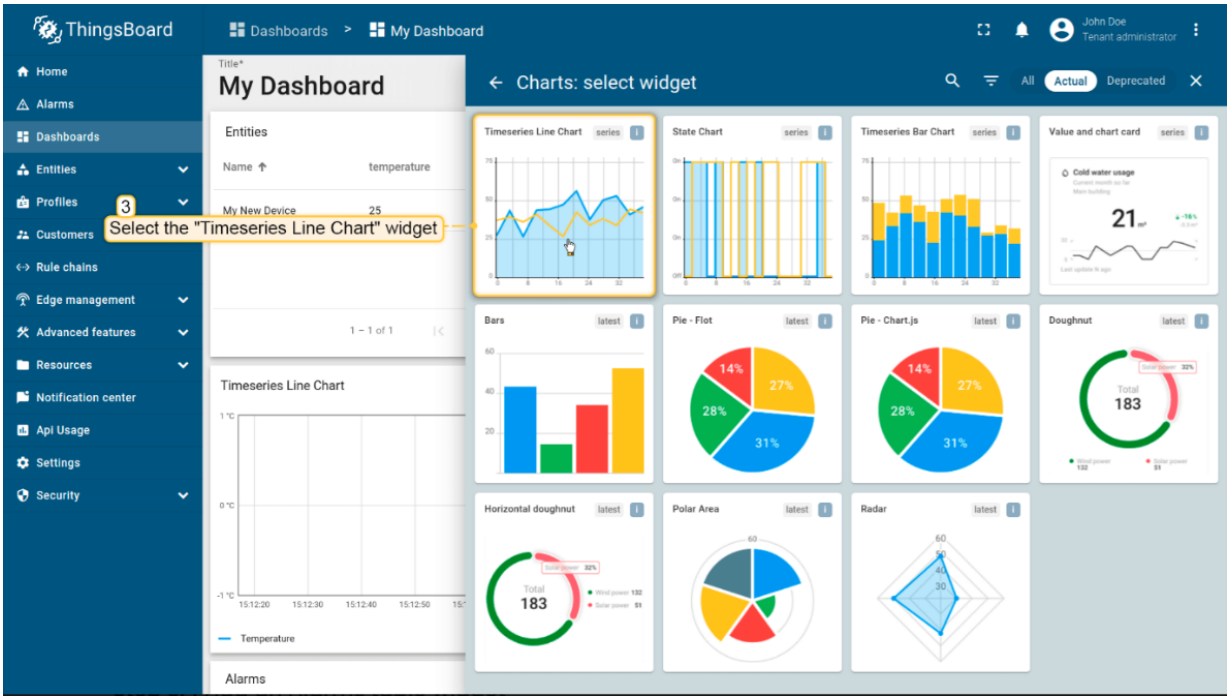

Рис. 23 Выбор графика в виде линии изменений параметра во времени

Измените размер и положение графика по своему усмотрению и сохраните изменения, рис. 24. Виджеты можно перетаскивать и располагать друг над другом или в одну линию.

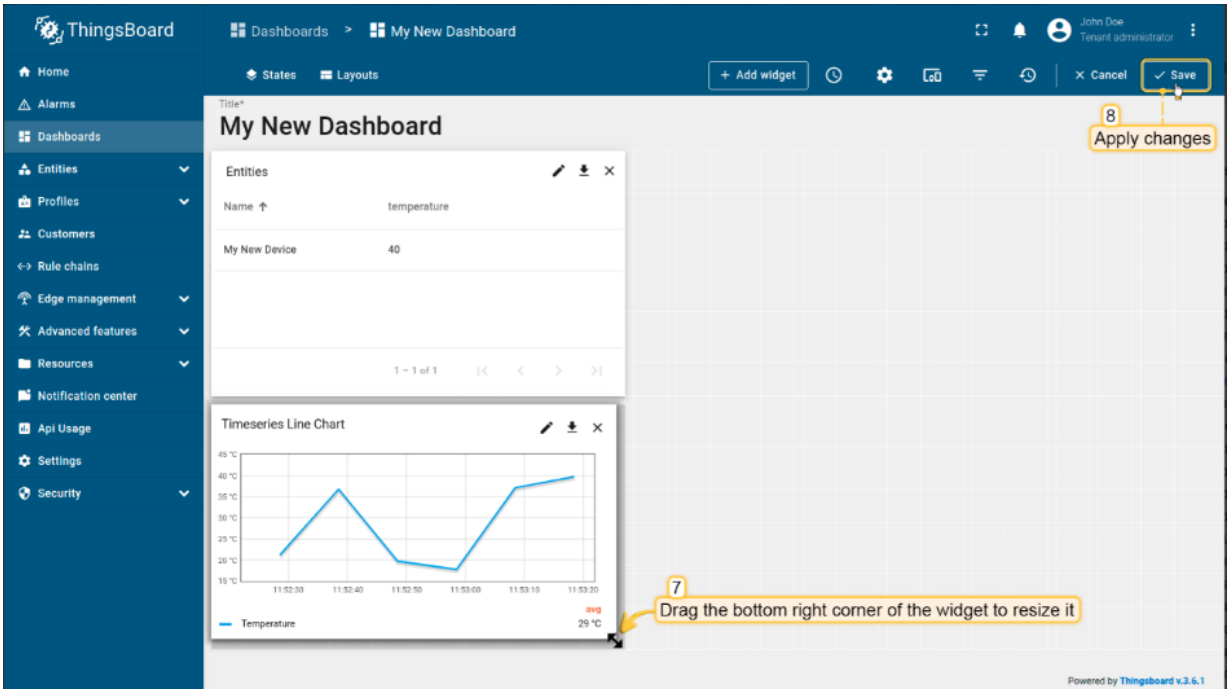

Рис. 24 Изменение размера графика

По умолчанию настройки виджета в виде линейного графика таковы, что отображаются показания, близкие к реальному времени, только за последнюю минуту. При нечастых отправках данных источником на графике будет одна точка или вовсе ничего. Перейдите в настройки временного слота и измените период, рис. 25. Можно включить параметр расчета среднего значения за выбранный временной интервал. Отправьте еще несколько раз данные через mosquitto_pub, меняя значения, чтобы получить некоторую кривую на графике.

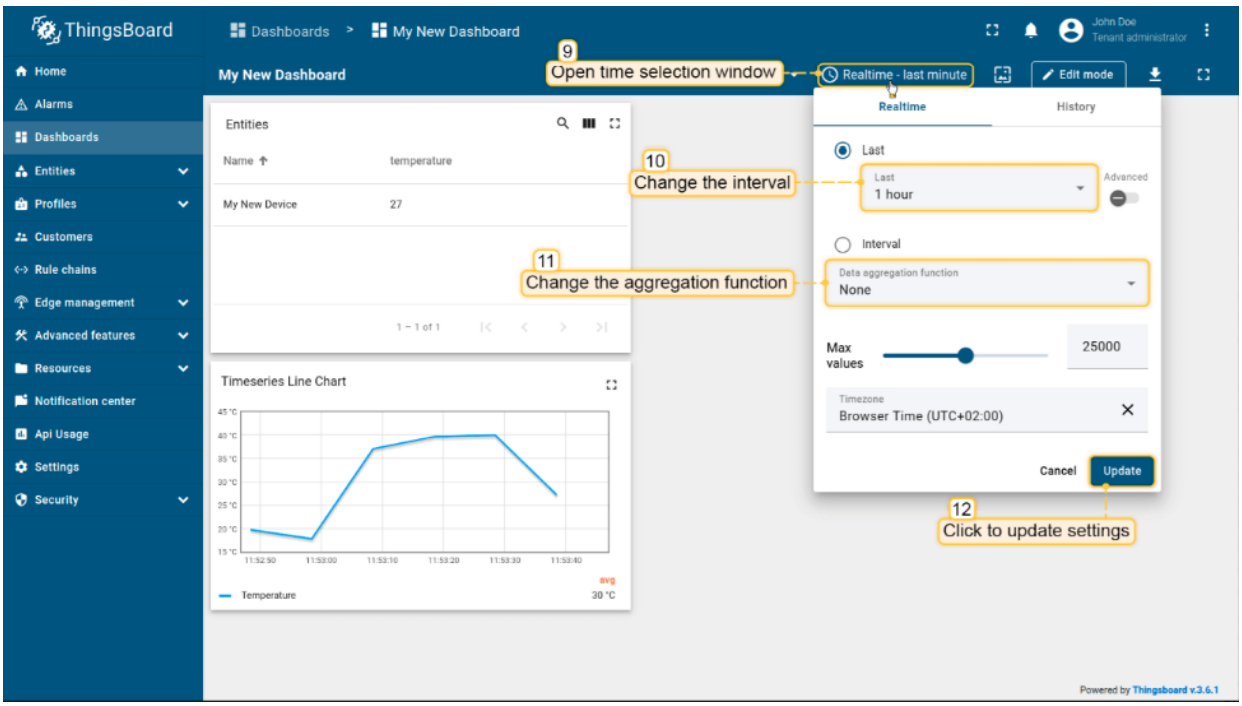

Рис. 25 Настройка временного окна для вывода на график

Модифицируем файл docker-compose.yaml. Для остановки запущенного сервиса и удаления контейнера выполните команду:

sudo docker-compose down

Откроем ранее созданный файл в каталоге lab4:

vi docker-compose.yaml

и добавим еще один сервис

```
version: "3"

services:
  thingsboard:
    image: "thingsboard/tb-postgres"
    container_name: mytb
    ports:
      - "8080:9090"
      - "7070:7070"
      - "1883:1883"
      - "5683-5688:5683-5688/udp"
    volumes:
      - ~/.mytb-data:/data
      - ~/.mytb-logs:/var/log/thingsboard

    networks:
      - node-red-net
    restart: always


  nodered:
```

```
    image: "nodered/node-red"
    container_name: mynodered
    ports:
      - "1880:1880"
    volumes:
      - node-red-data:/data
    networks:
      - node-red-net
    restart: always

volumes:
  node-red-data:
```

Запустим оба контейнера командой:

```
sudo docker-compose up -d
```

Процесс запуска сопровождается сообщениями в консоли, говорящими был ли он успешен, но можно дополнительно убедиться что оба контейнера запущены и работают, введя команду:

```
docker ps
```

Настроим отправку сообщений с данными о температуре из Node-RED в ThingsBoard. Оба сервиса размещены в одной сети и могут видеть друг друга по именам, но в настройках клиента MQTT на Node-RED нужно указать цифровой IP-адрес. Зайдем в контейнер Node-RED командой:

```
docker exec -it mynodered bash
```

и выполним эхо-запрос по имени сервиса ThingsBoard:

```
ping thingsboard
```

Запомните IP-адрес по которому выполняются эхо-запросы. Зайдите в веб-интерфейс Node-RED по адресу localhost:1880 (или адрес виртуальной машины, если контейнер запущен в виртуальной машине). Добавьте узел MQTT клиента-издателя, рис. 26

Рис. 26 Узел MQTT клиента-издателя в Node-RED

Укажите тот IP-адрес, который вы определили на предыдущем шаге для сервиса ThingsBoard. В качестве темы укажите ту же тему, куда отправлялись сообщения клиентом Mosquitto, она показана на рисунке 27.

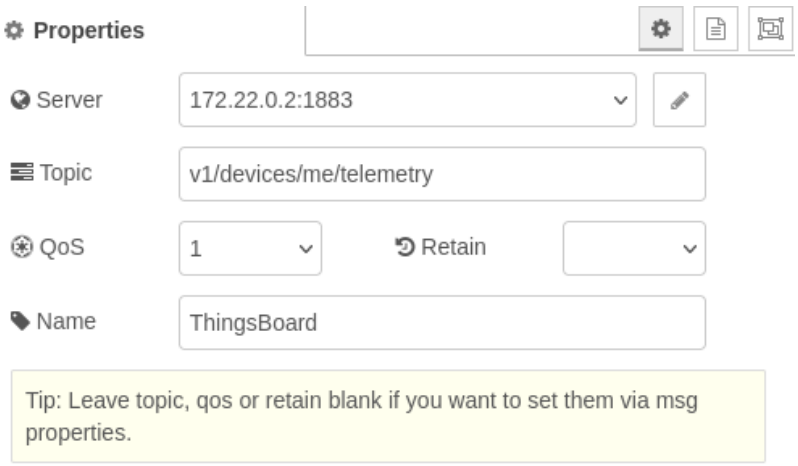

Рис. 27 Настройка клиента MQTT в Node-RED

Откройте настройки сервера (значок карандаша справа от адреса). На вкладке Security в поле имя пользователя введите токен, сгенерированный для созданного вами устройства, рис. 28

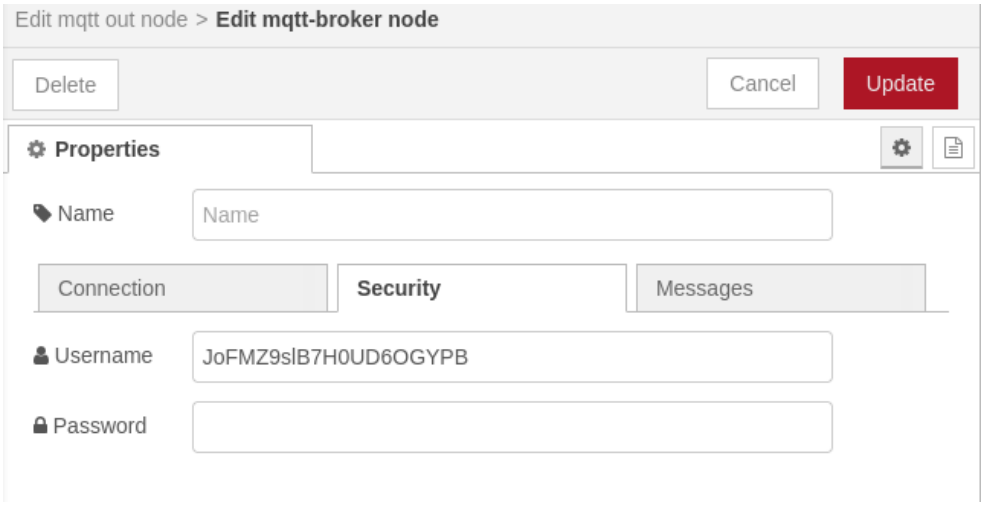

Рис. 28 Использование токена в качестве имени пользователя

Токен можно скопировать в свойствах устройства рис. 29, кнопка Copy access token.

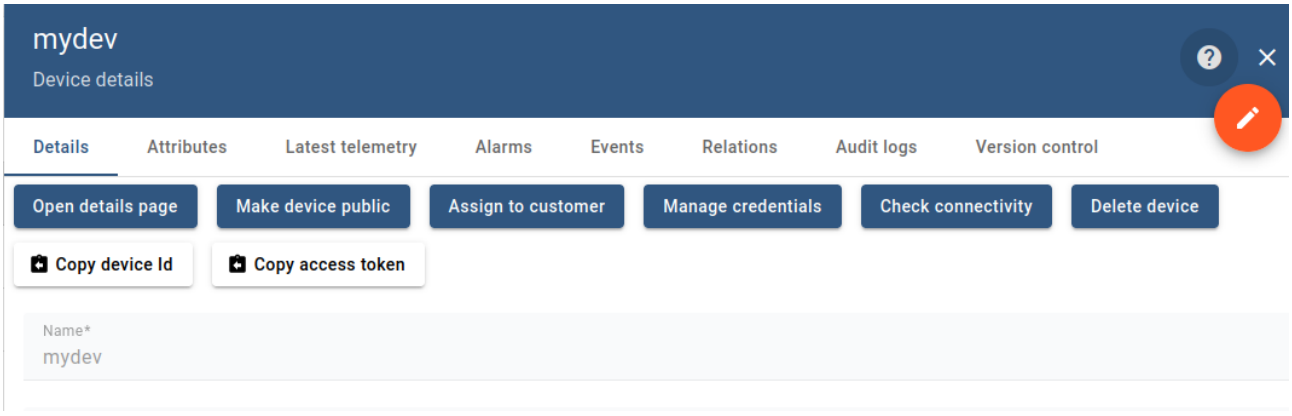

Рис. 29 Копирование токена в свойствах устройства

На вкладке «Сообщения», Messages введите новое значение температуры, рис. 30

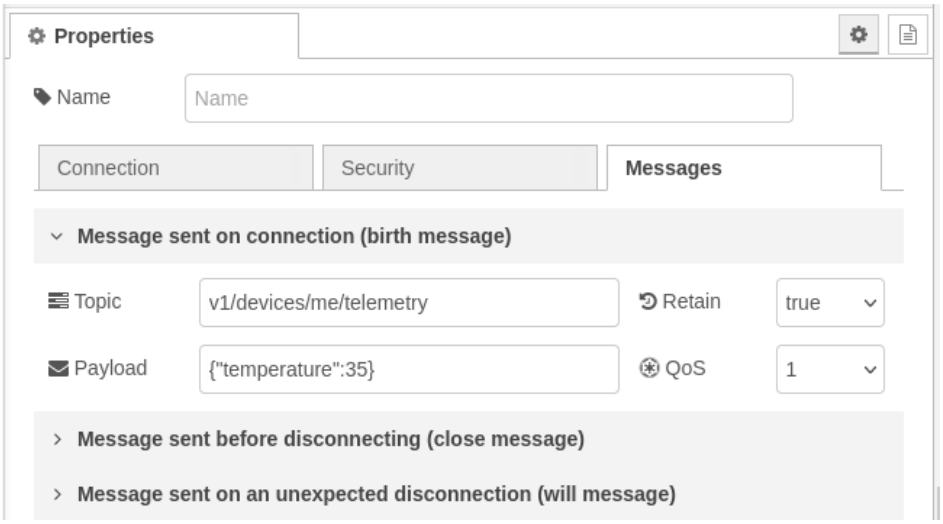

Рис. 30 Определение полезной части сообщения

Сохраните изменения и выполните деплой вашей конфигурации из одного узла. Установлено ли соединение с брокером Thingsboard? Проверьте, появилась ли новая запись в показаниях последней телеметрии и на созданных дашбордах?

По окончании работы вновь выполните команду:

sudo docker-compose down

для остановки контейнеров.

Отчет по работе загрузите для проверки.

5. Практическая работа №5
МОДЕЛИРОВАНИЕ СЕТИ ВЗАИМОДЕЙСТВУЮЩИХ УСТРОЙСТВ В СРЕДЕ Contiki Cooja

В качестве программного обеспечения для микропроцессорных систем сенсорных узлов можно использовать операционные системы Contiki, TinyOS и FreeRTOS, с открытым исходным кодом.

Contiki — это операционная система для сетевых систем с ограниченным объемом памяти, ориентированная на маломощные беспроводные устройства Интернета вещей (IoT). Contiki используется в системах уличного освещения, звукового мониторинга умных городов, радиационного мониторинга и сигнализации. Это программное обеспечение с открытым исходным кодом, выпущенное по лицензии BSD-3-Clause. Contiki создана Адамом Данкелсом в 2002 году и далее развивался всемирной командой разработчиков из Texas Instruments, Atmel, Cisco, ENEA, ETH Zurich, Redwire, RWTH Аахенского университета, Оксфордского университета, SAP, Sensinode, Шведского института компьютерных наук, ST Microelectronics, Zolertia и многих других. Операционная система приобрела популярность благодаря встроенному стеку TCP/IP и легкому упреждающему планированию по сравнению с ядром, управляемым событиями, что является очень мотивирующей функцией для Интернета вещей. Название «Контики» происходит от знаменитого плота «Кон-Тики» Тура Хейердала.

Contiki обеспечивает многозадачность, но требует всего около 10 килобайт оперативной памяти (ОЗУ) и 30 килобайт постоянной памяти (ПЗУ). Для полной системы, включая графический интерфейс пользователя, требуется около 30 килобайт оперативной памяти.

Недавно была создана новая ветка, известная как Contiki-NG: ОС для устройств IoT следующего поколения.

Contiki предоставляет три сетевых механизма: стек uIP TCP/IP, который обеспечивает работу сети IPv4, стек uIPv6, который обеспечивает работу сети IPv6, и стек Rime, который представляет собой набор специальных облегченных сетевых протоколов, предназначенных для маломощных беспроводных сетей. Стек IPv6 был предоставлен Cisco и на момент выпуска был самым маленьким стеком IPv6, получившим сертификат готовности к IPv6. Стек IPv6 также содержит протокол маршрутизации для сетей с низким энергопотреблением и потерями: Routing Protocol for Low power and Lossy Networks, RPL для сетей IPv6 с низким энергопотреблением и уровень сжатия и адаптации заголовка 6LoWPAN для каналов IEEE 802.15.4.

Rime — это альтернативный сетевой стек, который можно использовать, когда накладные расходы стеков IPv4 или IPv6 слишком велики. Стек Rime предоставляет набор примитивов связи для маломощных беспроводных систем. Примитивами по умолчанию являются одноадресная рассылка с одним переходом, широковещательная рассылка с одним переходом, одноадресная рассылка с несколькими переходами, лавинная рассылка по сети и сбор данных без адреса. Примитивы можно использовать по отдельности или комбинировать для формирования более сложных протоколов и механизмов.

Создание систем Интернета вещей требует зачастую больших финансовых затрат на развертывание интеллектуальной инфраструктуры: сетей сенсоров, исполнительных

устройств автоматизации, шлюзов между сенсорной сетью и каналом Интернет, устройств хранения и обработки информации. Поэтому важно при проектировании таких систем не только выполнять соответствующие расчеты, но иметь возможность проверить, опробовать решение на тестовом полигоне.

Система Contiki включает в себя симулятор датчиков под названием Cooja, который имитирует узлы Contiki. Узлы принадлежат к одному из трех классов:

а) эмулируемые узлы Cooja,

б) код Contiki, скомпилированный и выполняемый на хосте моделирования,

в) узлы Java, где поведение узла должно быть переопределено как класс Java.

Одна симуляция Cooja может содержать смесь сенсорных узлов любого из трех классов. Эмулированные узлы также можно использовать для включения узлов, не принадлежащих Contiki, в моделируемую сеть.

ВЫПОЛНЕНИЕ РАБОТЫ

Импортируйте файл подготовленной виртуальной машины в среду Oracle VirtualBox. Запустите виртуальную машину. Пользователь **user** имеет пароль **user**. В графическом интерфейсе нужно ввести только пароль. На рабочем столе есть ярлык симулятора Cooja, запустите его. Откроется пустое окно рабочего пространства симулятора с синим фоном. Создайте новую симуляцию, задав для нее имя, рис. 1, 2. Можно оставить имя по умолчанию.

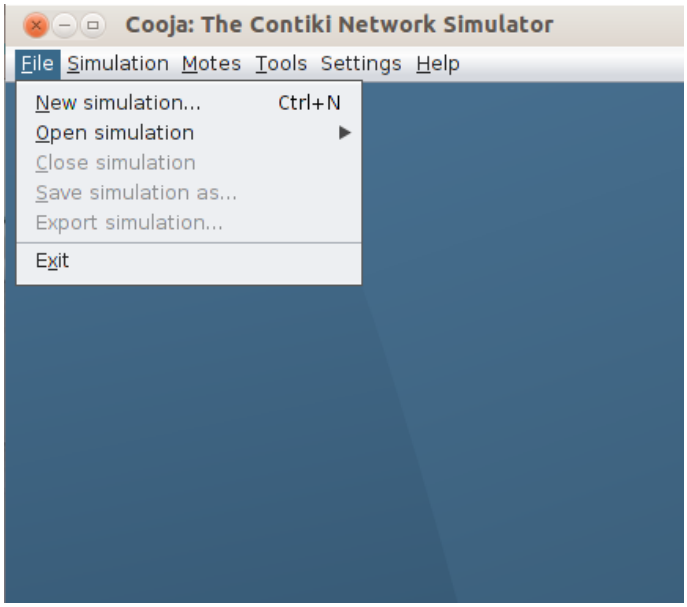

Рис. 1 Создание новой симуляции

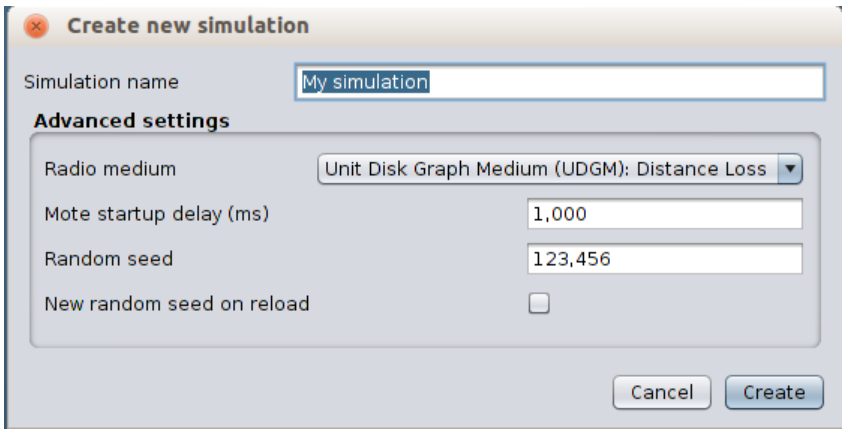

Рис. 2 Задание имени для проекта симуляции

После нажатия кнопки Create будет создано пять окон для контроля проекта, рис. 3.

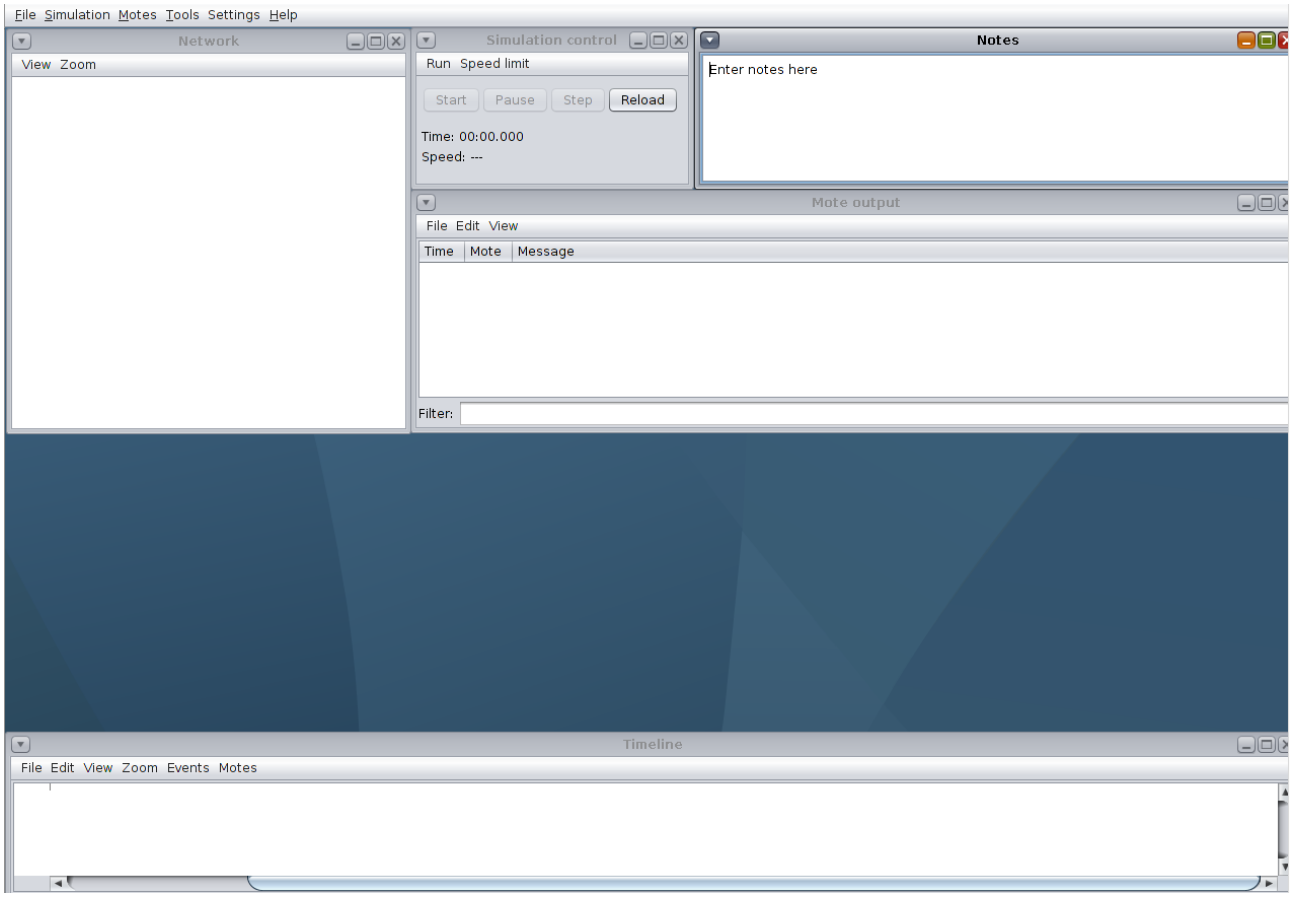

Рис. 3 Окна симуляции Cooja

В окне с подзаголовком **Network** отображается топология сети с взаимодействующими устройствами Интернета вещей. Окно с подзаголовком **Simulation control** управляет симуляцией, запускает ее, ставит на паузу, позволяет настроить скорость выполнения. Окно с подзаголовком **Notes** предназначено для заметок экспериментатора. В окне **Mote output** выводятся сообщения с одного узла сети. В терминологии симулятора узлы называются **Motes**, - пылинки или пятнышки в большой туче Интернета вещей. Нижнее окно **Timeline** выводит события в сети устройств на временной ленте. Вы можете настраивать размер окон по своему усмотрению, перетягивая их границы.

Узел Mote в Cooja — это скомпилированная и исполняемая система Contiki. Система контролируется и анализируется Cooja. Несколько разных библиотек Contiki могут быть

скомпилированы и загружены в одну и ту же симуляцию Cooja, представляющую разные типы сенсорных узлов (гетерогенных сетей). Cooja контролирует и анализирует систему Contiki с помощью нескольких функций. Например, симулятор сообщает системе Contiki о необходимости обработки события или извлекает всю системную память Contiki для анализа. Такой подход дает симулятору полный контроль над моделируемыми системами.

Создадим несколько узлов. В меню Motes выберите Add motes, затем Create new mote type и в предлагаемых типах выберите Sky mote, рис. 4. Узлы типа Sky – самые простые из набора возможных типов для использования в WSN и идеально подходят для начальных конфигураций в рамках моделирования Cooja.

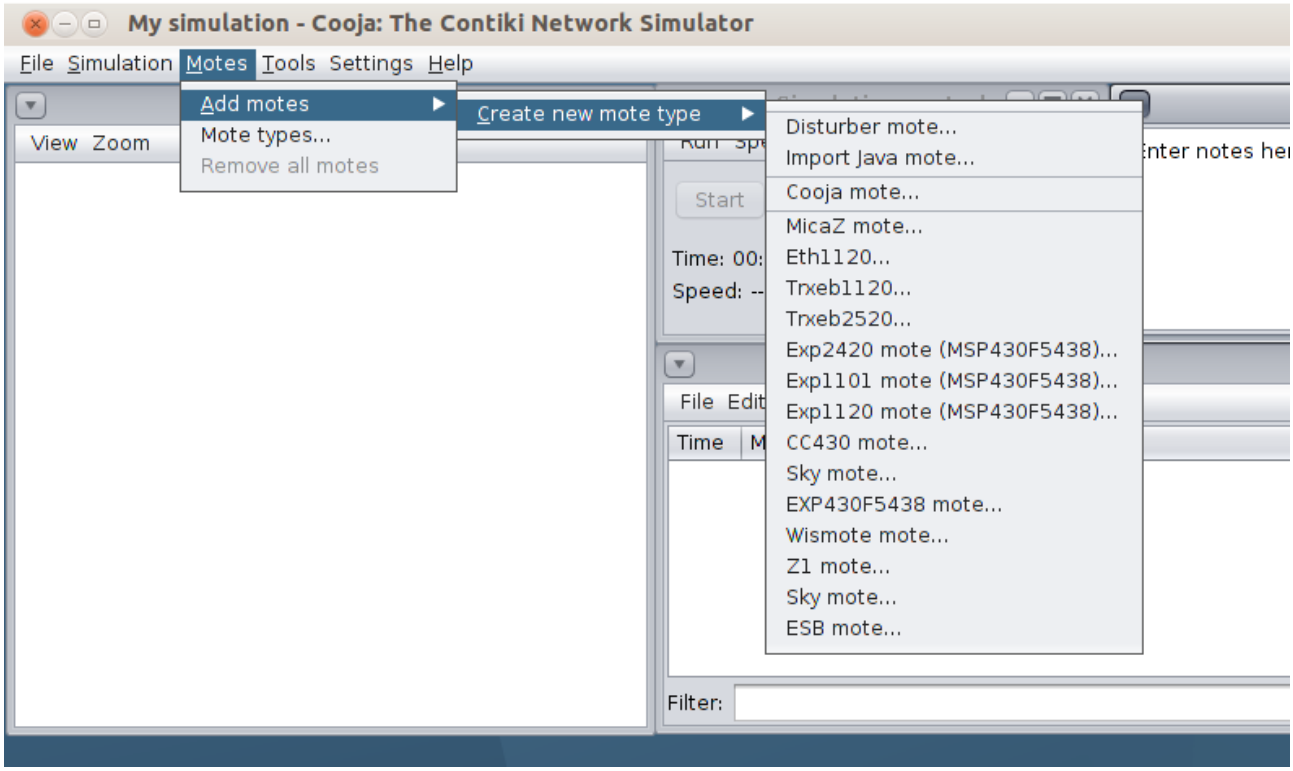

Рис. 4 Создание в Cooja узлов типа Sky

Мы создали узел, но это пока еще пустой шаблон. Для его программного наполнения нужно определить, что он будет делать. Это реализуется программно. Мы используем уже готовые программы. Нажмите кнопку Browse и выберите в папке примеров файл hello-world.c, рис. 5.

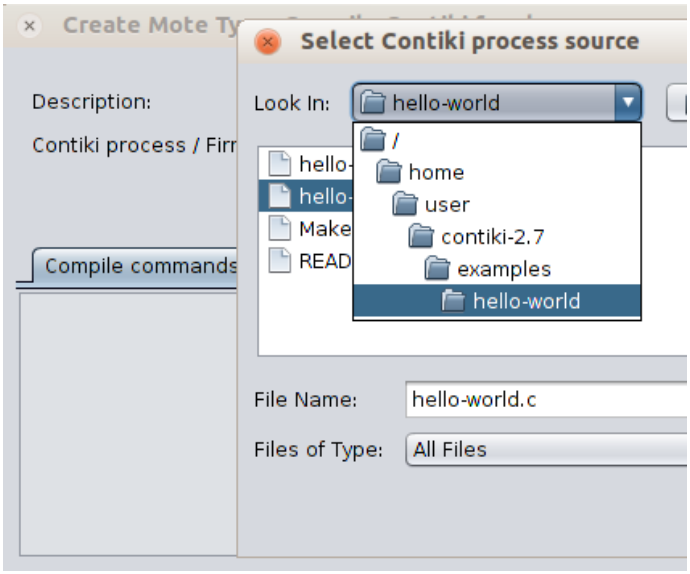

Рис. 5 Выбор файла для «начинки» созданного узла

Нажмите клавишу Compile для создания исполнимой программы, рис. 6. В процессе компиляции могут наблюдаться предупреждения, выделяемые красным цветом, рис. 7, но если в итоге не отображается ошибка, то компиляция завершилась удачно.

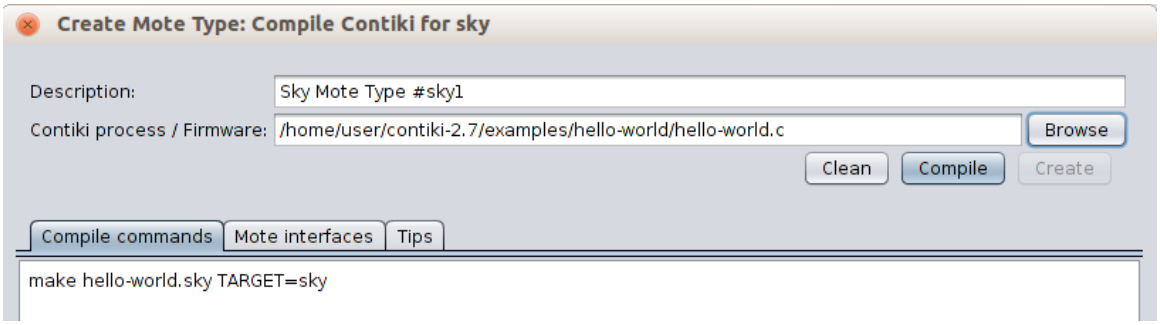

Рис. 6 Компиляция выбранного кода

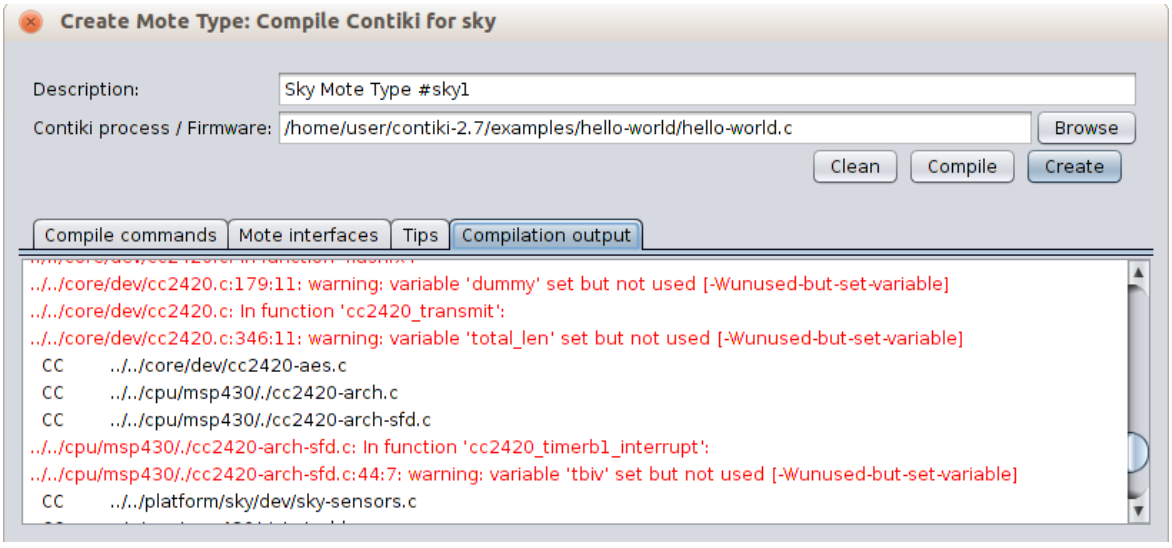

Рис. 7 Предупреждения в ходе компиляции

Индикатором успешного завершения процесса компиляции является сообщение о полученных исполнимых файлах и активация кнопки Create, рис. 8.

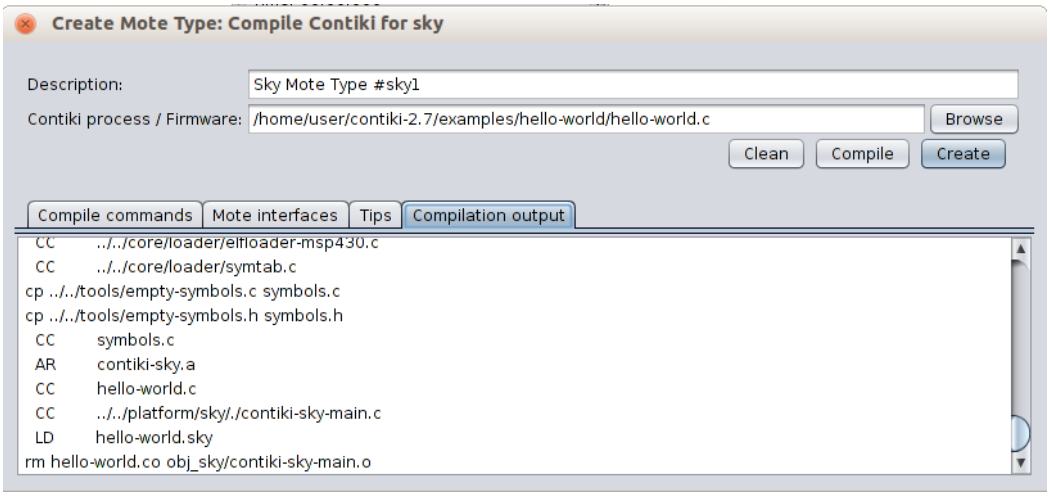

Рис. 8 Завершение процесса компиляции

Нажмите кнопку Create и создайте 5 узлов, расположив их по эллиптической кривой, рис. 9

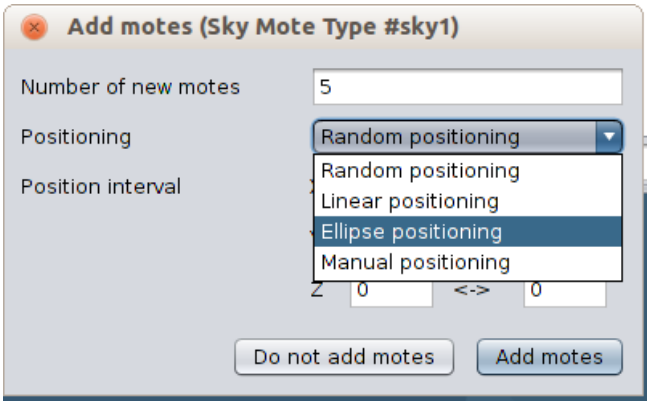

Рис. 9 Определение числа и расположения узлов

Как только мы нажмем кнопку Add motes, узлы появятся в окне топологии сети, а в окне управления симуляцией активируются клавиши управления, рис. 10.

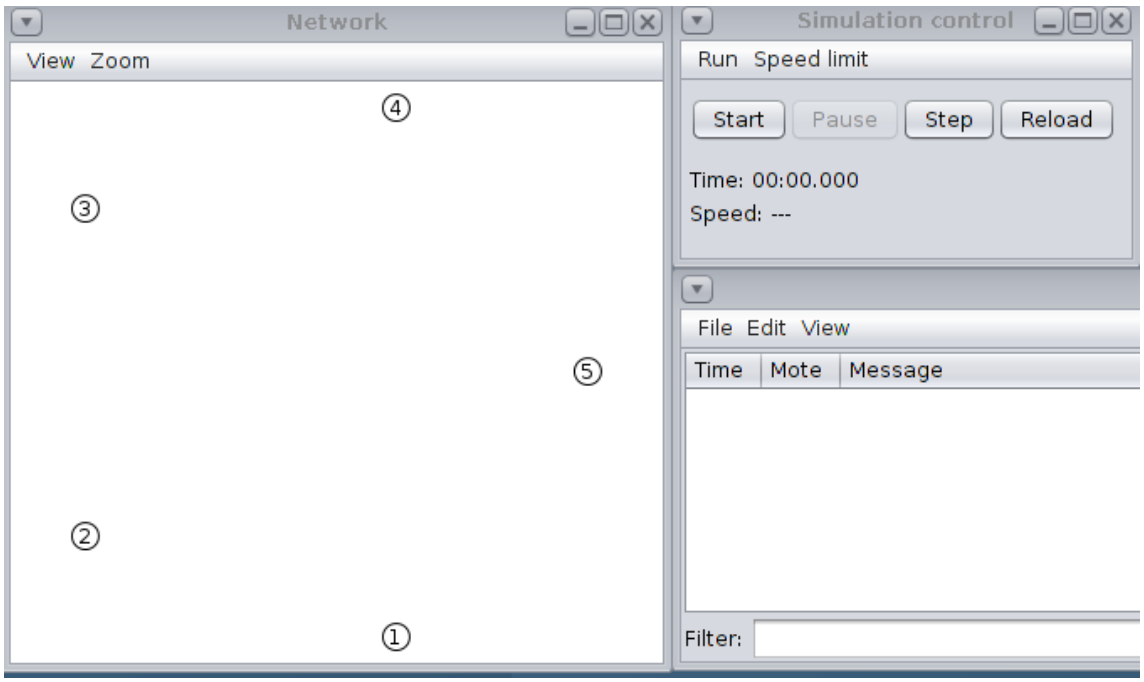

Рис. 10 Созданные узлы в топологии сети

В окне топологии сети есть подменю View и Zoom, если Zoom это говорящее название, то в меню View мы можем настроить параметры узлов-Motes для вывода этой информации в окно топологии, рис. 11: полученных IP-адресов, сигналов радио-трафика, типа устройств и т. д. Важное значение имеет параметр Radio environment, он показывает радиус действия радиосети. Зеленая область — уверенный прием, серая область — зона интерференции, рис. 12. Для отображения радиуса действия достаточно поставить галочку в меню View. Меню 10m background grid включает в окне топологии линии сетки с шагом 10 метров по горизонтали и вертикали. Узлы могут легко перемещаться в поле топологии. Как по отдельности, если щелкнуть по узлу указателем мышки и протянуть с нажатой левой клавишей, так и все вместе, если щелкнуть мышкой в пространстве между узлами и также протянуть с нажатой левой клавишей.

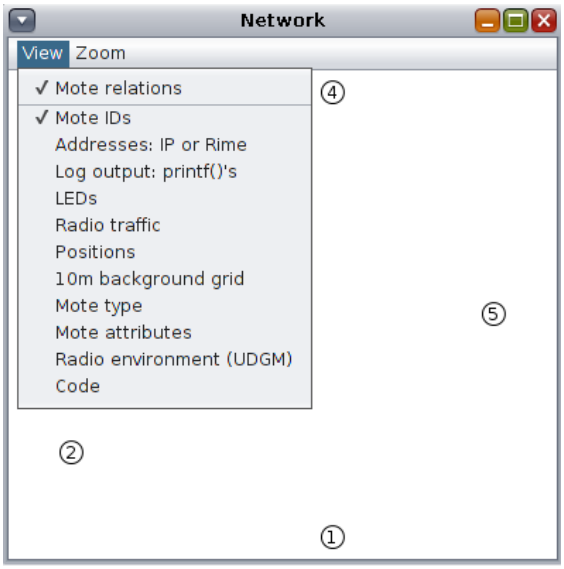

Рис. 11 Настройка выводимых в окно топологии параметров узлов

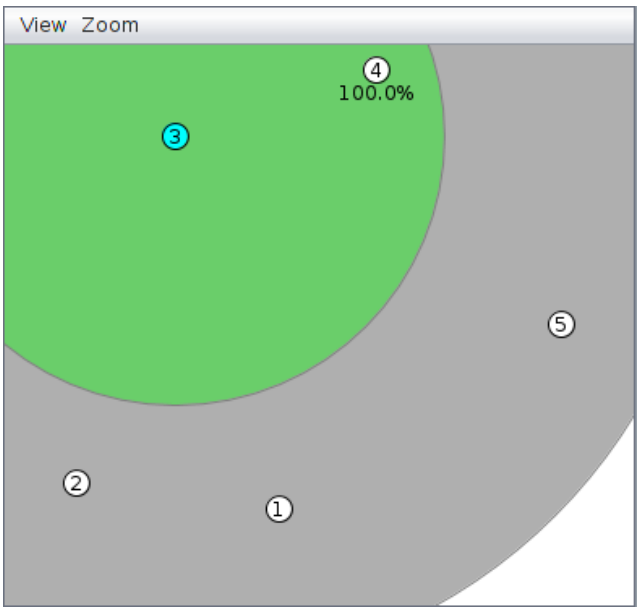

Рис. 12 Включение параметра Radio environment

Изменить радиус действия радиосети можно, вызвав контекстное меню щелчком правой клавиши мышки на узле и выбрав в нем пункт Change transmission ranges, рис. 13.

Запустите симуляцию кнопкой Start в окне управления симуляцией. Между узлами начнут передаваться пакеты, если они находятся в зоне досягаемости (в зеленой области радиуса уверенного приема). Если это не так, переместите узлы, чтобы они были достижимы для сообщений, рис. 14.

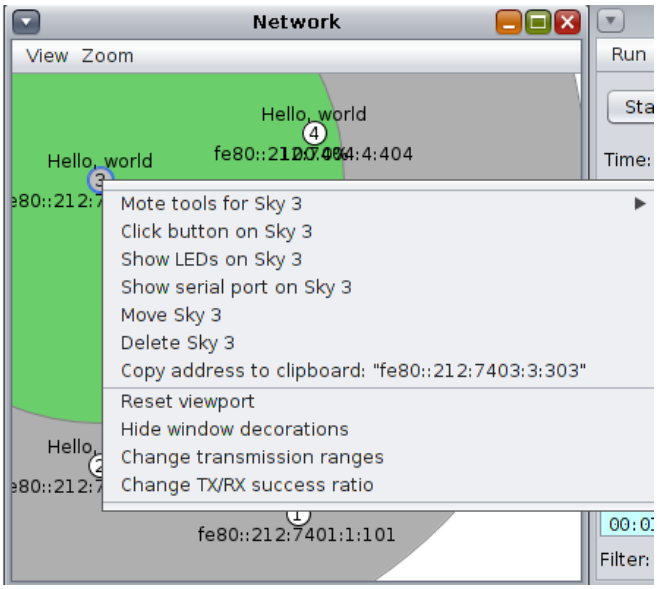

Рис. 13 Настройка радиуса действия радиосети и зоны интерференции

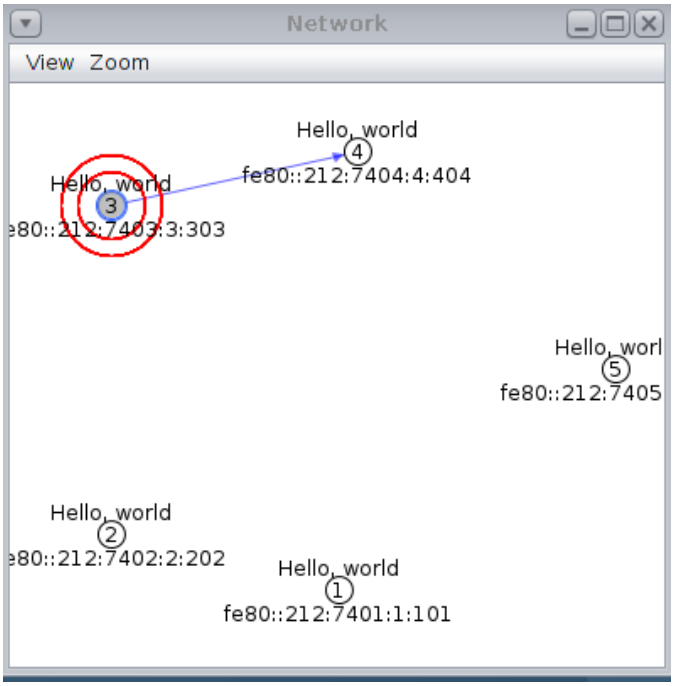

Рис. 14 Передача сообщений между узлами

Выбрав пункт в меню Tools → Radio messages, рис. 15. можно увидеть подробности передаваемых по радиоканалу сообщений. В появившемся дополнительном окне появляются радио-сообщения в момент их передачи, показана метка времени, IP-адреса узла отправителя и получателя, размер данных. При выборе конкретного пакета в этом окне, открывается дополнительное окно ниже с подробным содержимым, подобно тому, как это происходит в анализаторе пакетов Wireshark.

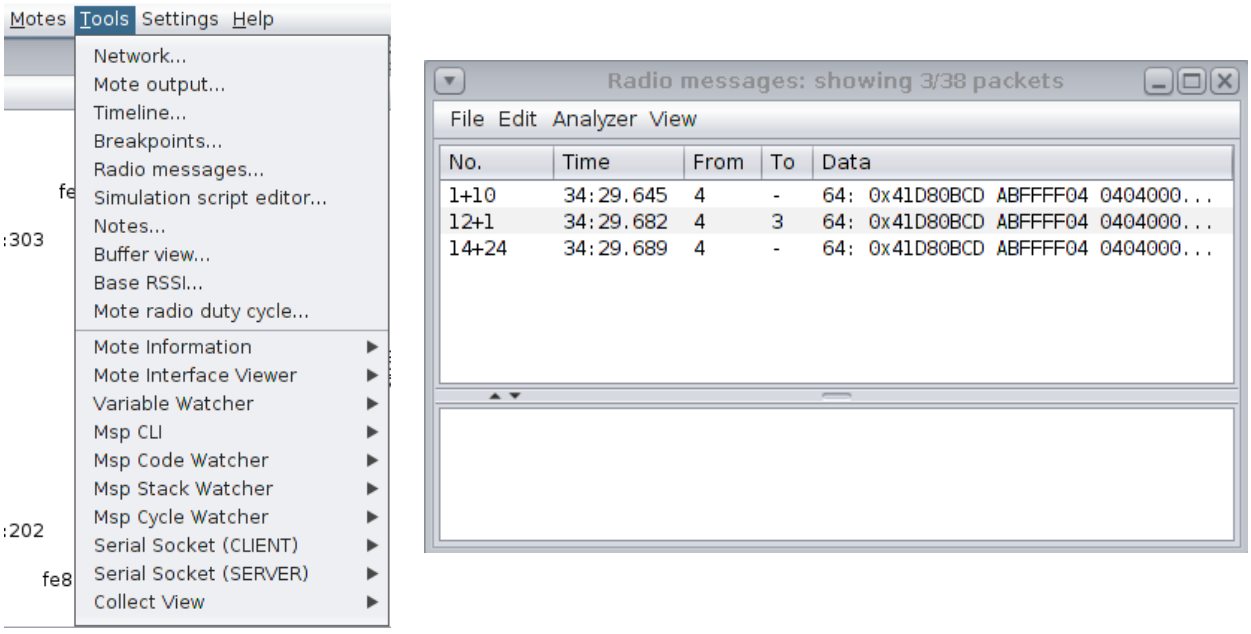

Рис. 15 Вывод радиосообщений в меню Tools в отдельном окне

В меню Mote types… можно увидеть изображение устройства, реализующего соответствующий функционал, рис. 16.

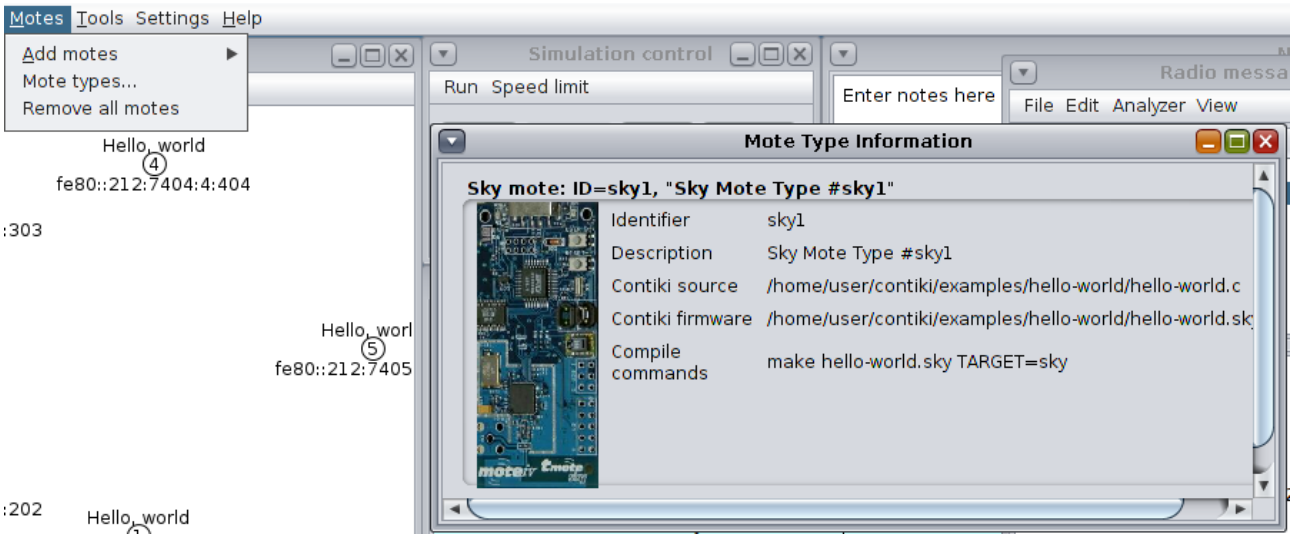

Рис. 16 Изображение устройства Sky

Сохраните вашу симуляцию через меню File → Save as в качестве названия задайте вашу фамилию, расширение устанавливается csc. Загрузите этот файл на проверку в качестве отчета по работе.

6. Практическая работа №6
СОЗДАНИЕ БЕСПРОВОДНОЙ СЕТИ ВЗАИМОДЕЙСТВУЮЩИХ УСТРОЙСТВ WSN, МАРШРУТИЗАЦИЯ RPL, ПРОТОКОЛЫ 6LoWPAN и CoAP.

Общая информация

Беспроводные сенсорные сети (WSN) играют ключевую роль в создании и развитии Интернета вещей. Большинство устройств IoT это маломощные в плане вычислительных ресурсов, малогабаритные электронные платы с подключением к Интернет. Такие устройства меняют нашу жизнь, делая ее комфортнее и безопаснее за счет качественно новых типов сервисов. Устройства Интернета вещей часто используют электрические батареи различных типов для питания электронной схемы. Важным фактором удобства и эффективности решения в этом случае оказывается длительный срок работы без замены батарей. Один из основных стандартов, который поддерживает сети с низким энергопотреблением и потерями (LLN, Low-Power and Lossy Network) — стандарт IEEE 802.15.4, который составляет основу WSN как часть Интернета вещей. Этот стандарт определяет физические и канальные уровни сети и обеспечивает основу их работы. Чтобы сделать устройства IoT частью Интернета, IETF разработал вариант протокола IPv6 — 6LoWPAN, IPv6 over Low power Wireless Personal Area Networks с уменьшенными накладными расходами на служебные заголовки и, как результат, меньшим энергопотреблением. Протокол 6LoWPAN используются в качестве уровня адаптации, связывая традиционную сеть Интернет и отличающуюся от Интернет беспроводную сеть с низкими скоростями и плохим качеством доставки сообщений.

На рис. 1 слева показан традиционный сетевой стек Интернета, а справа стек протоколов для устройств Интернета вещей.

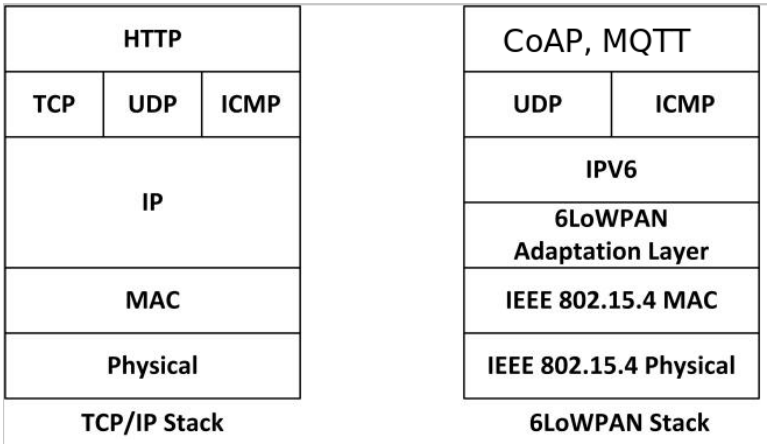

Рис. 1 Сравнение стека протоколов Интернет и Интернет вещей

Огромное число устройств в беспроводной сенсорной сети и низкая вычислительная мощность узлов не позволяет использовать в них уже известные протоколы динамической маршрутизации. Поэтому были созданы специальные протоколы маршрутизации для Интернета вещей. Сегодня наибольшую популярность завоевал протокол маршрутизации для сетей с низким энергопотреблением и потерями RPL (Routing Protocol for Low-Power and Lossy Networks). Это упреждающий протокол, основанный на векторах расстояния и

работающий на базе IEEE 802.15.4, оптимизированный для многопереходной связи и связи «многие к одному», но также поддерживающий сообщения «один к одному».

Протокол описан в RFC 6550 в 2012г и обновлен в RFC9008, 9010 в 2021г. Ряд прикладных вопросов применения протокола описан в RFC5867, 5826, 5673 и 5548. RPL может поддерживать широкий спектр канальных уровней, в том числе с ограничениями, с потенциальными потерями или которые используются в устройствах с ограниченными ресурсами. Этот протокол позволяет быстро создавать сетевые маршруты, обмениваться маршрутной информацией и эффективно адаптировать топологию. RPL строит ориентированный ациклический граф (Directed Acyclic Graph, DAG) без исходящих ребер в качестве базового элемента топологии, что гарантирует отсутствие циклов в иерархии. Первый узел создает начальную базу данных непосредственно доступных ему узлов, становясь корнем группы доступности. Другие узлы в этой группе формируют свои собственные группы доступности на основе информации о достижимых соседях, и передают эту информацию к первому узлу, создающему общую группу доступности достижимых адресов: базу данных, ориентированную на адресата (DODAG).

RPL использует ряд управляющих сообщений для построения и поддержания иерархии, рис. 2. Сообщение DIO, информационный объект DODAG, отправляется из корневого узла с информацией о ранге отправляющего узла, идентификаторе экземпляра, номере версии и DODAG-ID. Это позволяет узлам решать, действовать или нет после получения этого сообщения, а также сохранять ценную информацию о сети, которая может способствовать принятию обоснованного решения. Сообщение DAO, - Объект объявления назначения, отправляется от дочернего узла к его родительскому узлу (корню DAG или корню DODAG) и содержит пункт назначения. В этом сообщении содержится информация, что этот узел все еще доступен. При необходимости корневой узел может дополнительно отправить подтверждение DAO-ack. Сообщение DIS, Запрос информации DODAG — это еще одна форма восходящих управляющих сообщений, которая используется для запроса DIO напрямую от корневого узла. Это одна из наиболее важных и важных функций, которые RPL использует для поддержания соединения.

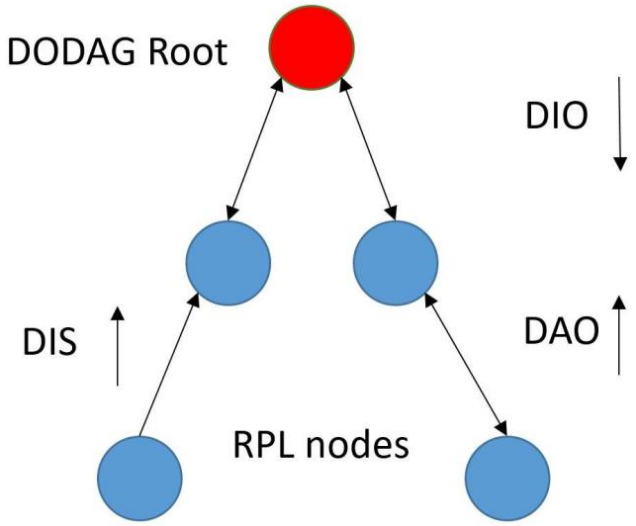

Рис. 2 Иерархическое дерево RPL

ВЫПОЛНЕНИЕ РАБОТЫ

Войдите в систему в подготовленной виртуальной машине. Логин user пароль user. Запустите симулятор Cooja, используя ярлык на рабочем столе. Создайте новый проект с именем WSN_coap.

Далее создадим Mesh-сеть беспроводных устройств, использующих протокол 6LoWPAN для коммуникации. Пограничный маршрутизатор в этой сети будет включен в нее и сможет обмениваться сообщениями с любым IoT устройством, в то же время он будет подключен к Интернет и доступен с компьютера, на котором выполняется работа.

Уже знакомый нам Sky Mote в симуляторе Cooja поддерживает протокол 6LoWPAN. Выберите этот тип Mote и загрузите исходный код пограничного маршрутизатора из каталога примеров, ipv6, рис. 3

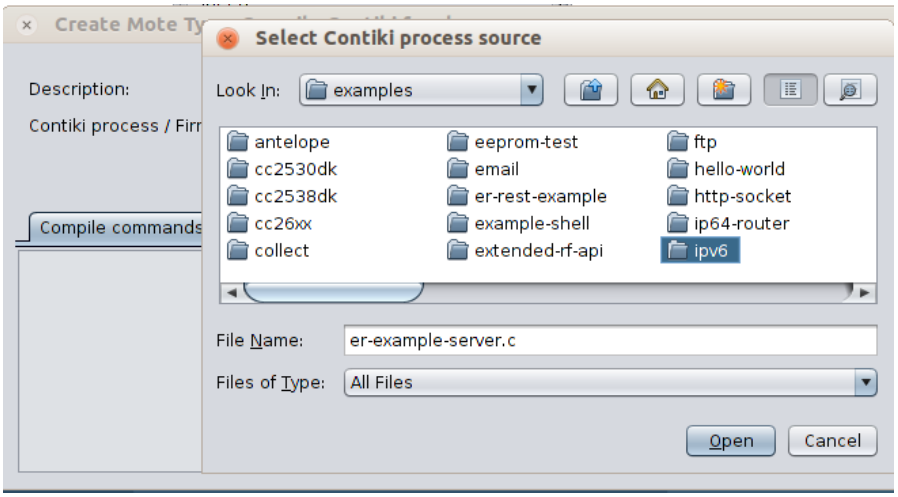

Рис. 3 Папка ipv6 в каталоге примеров

Здесь найдите каталог rpl-border-router, рис. 4

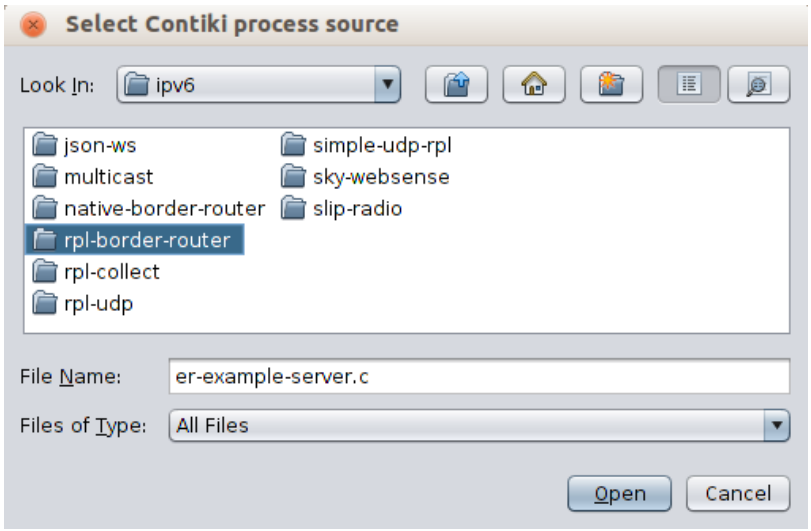

Рис. 4 Папка rpl-border-router внутри ipv6

В нем выберите файл **border-router.c**, рис. 5

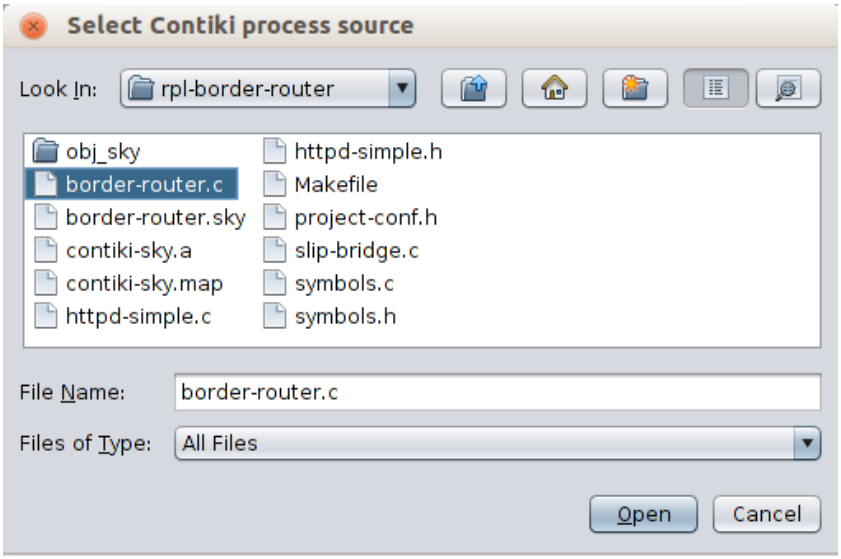

Рис. 5 Файл исходного кода пограничного маршрутизатора

Откомпилируйте файл, нажав клавишу Compile до получения исполнимых файлов и активизации кнопки Create, рис. 6.

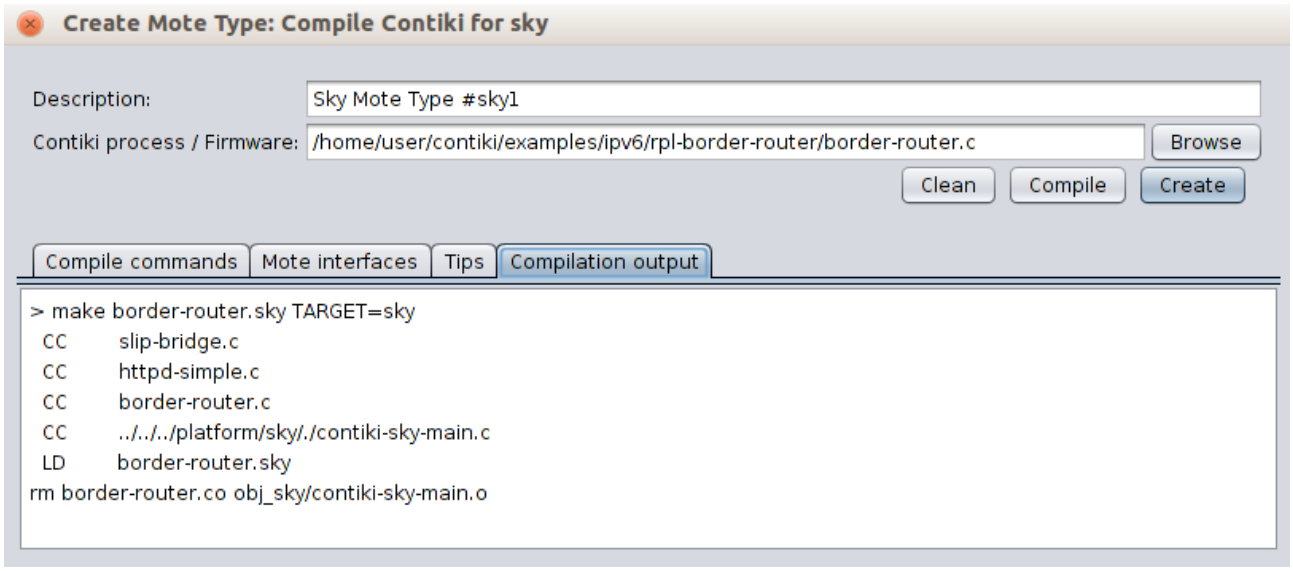

Рис. 6 Компиляция кода пограничного маршрутизатора

Создайте один объект такого типа. В окне топологии создаваемой сети включите отображение идентификатора, адреса, светодиодов, радио-трафика, сетки, типа мота, UDGM, рис. 7.

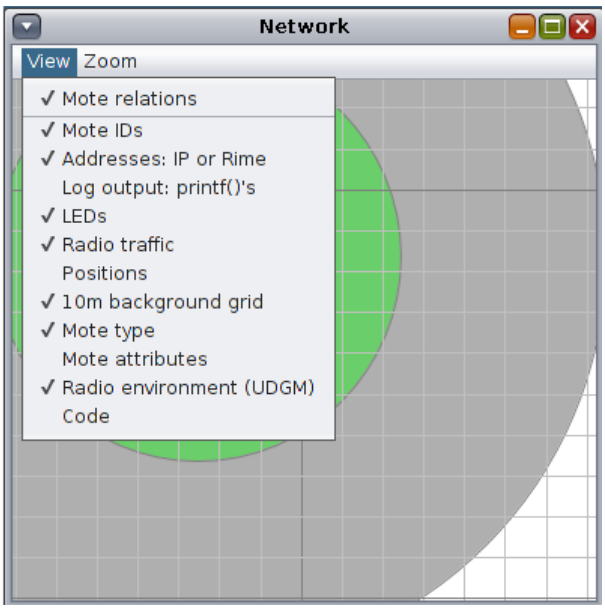

Рис. 7 Включение атрибутов, выводимых в поле топологии

Участниками сети будут устройства другого типа. Создайте еще один Sky Mote, выберите для него из каталога примеров папку **er-rest-example**, рис. 8

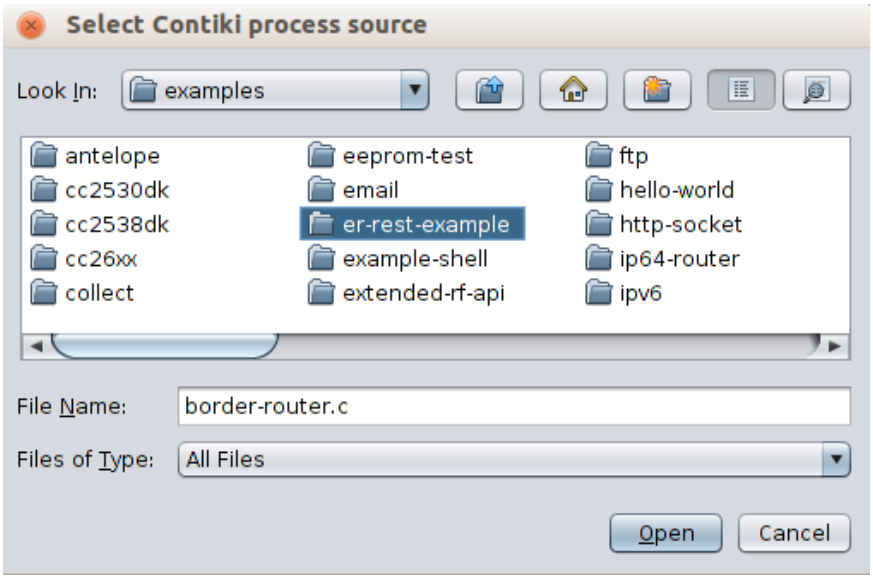

Рис. 8 Папка er-rest-example для второго типа Motes

В этой папке выберите файл **er-example-server.c**, рис. 9 и откомпилируйте его. Создайте шесть устройств такого типа, рис. 10. Расположите устройства таким образом, чтобы в зоне досягаемости пограничного маршрутизатора (зеленый круг) был только один узел, в зоне этого узла также был один или два следующих узла. Другими словами, так чтобы получилась распределенная, растянутая сеть, в которой пограничный узел не может «видеть» сразу всех напрямую и каждый отдельный узел видит только одного-двух соседей.

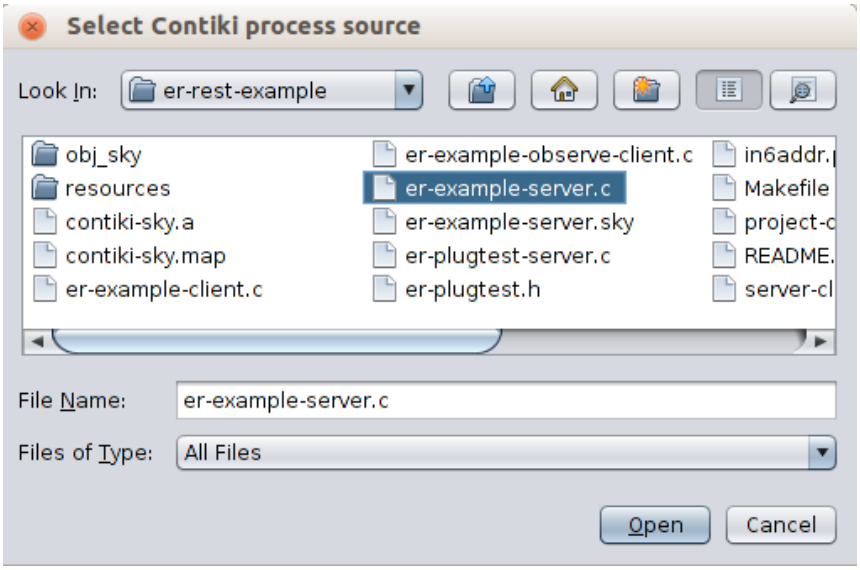

Рис. 9 Файл er-example-server.c для основных узлов меш-сети

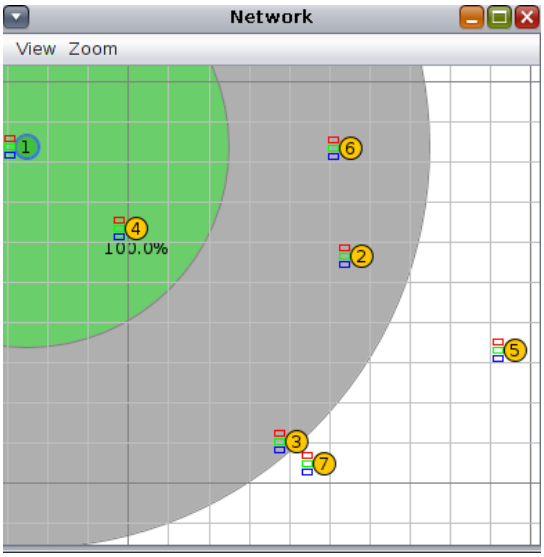

Рис. 10 Устройства двух типов в меш-сети

Для подключения пограничного маршрутизатора к сети Интернет создается туннельное соединение. Это соединение организуется в два этапа. Сначала на пограничном узле нужно активировать сокет для прослушивания. Нажмите на Sky Mote 1 правой клавишей мышки для вызова контекстного меню и выберите в нем пункт Mote tools --→ Serial Socker SERVER, рис. 11

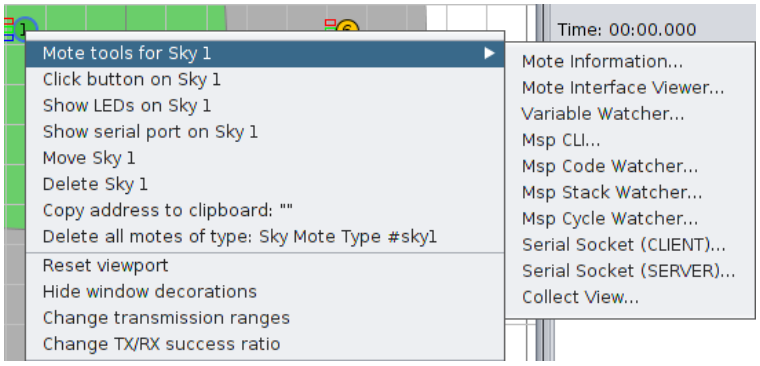

Рис. 11 Открытие сокета на пограничном маршрутизаторе

После этого в окне командной строки внутри виртуальной машины перейдите в каталог, в котором находится файл пограничного маршрутизатора, рис. 12.

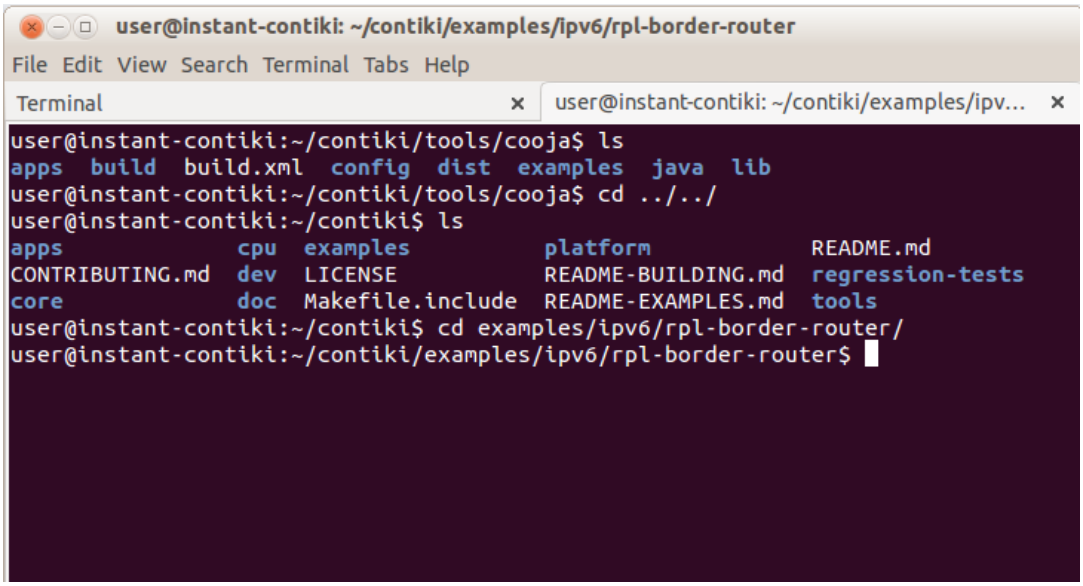

Рис. 12 Окно терминала в каталоге rpl-border-router

В этом каталоге выполните команду:

make connect-router-cooja

Как показано на рис. 13, эта команда поднимает туннельный интерфейс к созданному сокету. Пока симуляция не запущена, на интерфейсе нет реального адреса. Как только мы запускаем симуляцию, адрес появляется. Он будет отображен в том же окне, рис. 14.

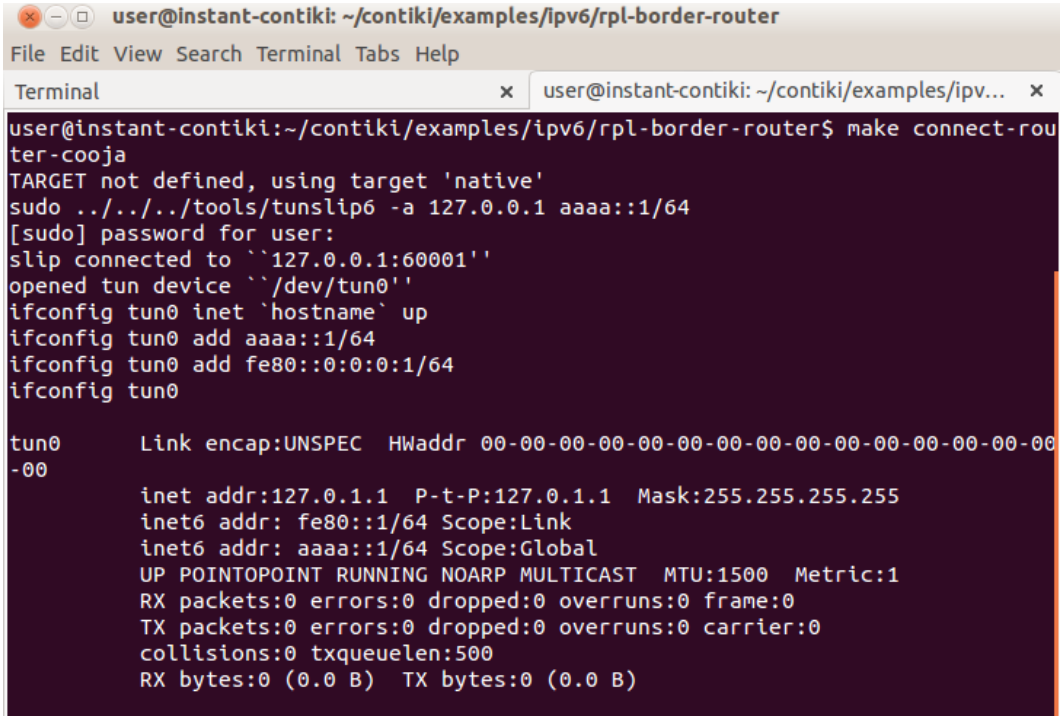

Рис. 13 Туннельный интерфейс подключения к пограничному маршрутизатору

```
Rime started with address 0.18.116.1.0.1.1.1
MAC 00:12:74:01:00:01:01:01 Contiki-2.6-2450-geaa8760 started. Node id is set to
 1.
nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshol
d -45
Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7401:0001:0101
Starting 'Border router process' 'Web server'
*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
 aaaa::212:7401:1:101
 fe80::212:7401:1:101
```

Рис. 14 Реальный адрес после запуска симуляции

Скопируйте адрес и вставьте его в окно браузера виртуальной машины. Адреса IPv6 при введении в адресную строку браузера должны заключаться в квадратные скобки, рис. 15

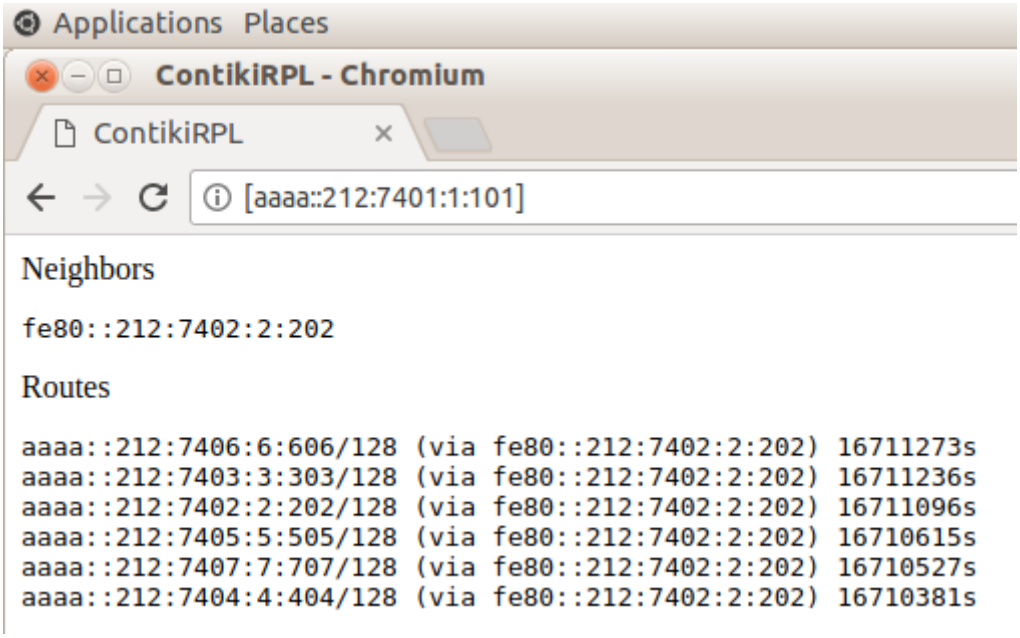

Рис. 15 Обращение к пограничному маршрутизатору через адресную строку браузера

Подзаголовок Neighbors показывает узлы, непосредственно доступные маршрутизатору. Далее идет таблица маршрутов RPL с указанием всех доступных узлов.

Откройте в меню Tools → Radio Messages окно анализа 6LoWPAN Analyzer, рис. 16. В выводе мы можем видеть перевернутую структуру отправляемых по сети данных (подобно тому, как это отображается и в анализаторе пакетов WireShark). Верхний уровень IEEE 802.15.4 является самым нижним в иерархии (рис. 1), затем следует 6LoWPAN, который применяет сжатие для заголовков IPv6, так что IPHC это сжатый формат заголовков IPv6. Далее идет уже протокол IPv6 в нормальном виде, мы можем видеть адрес отправителя и получателя. Как вышележащий протокол показан ICMPv6 — служебный протокол обработки ошибок и другой специальной информации для IPv6, он переносит сообщение протокола маршрутизации RPL типа DAO.

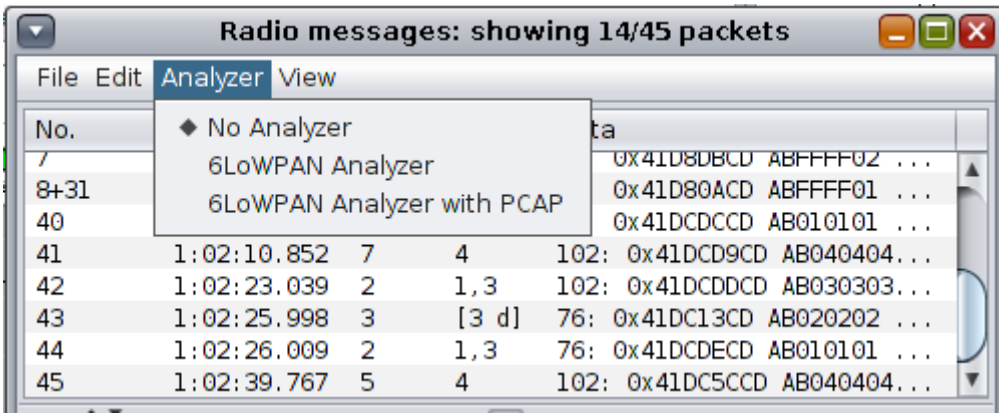

Рис. 16 Выбор анализатора 6LoWPAN

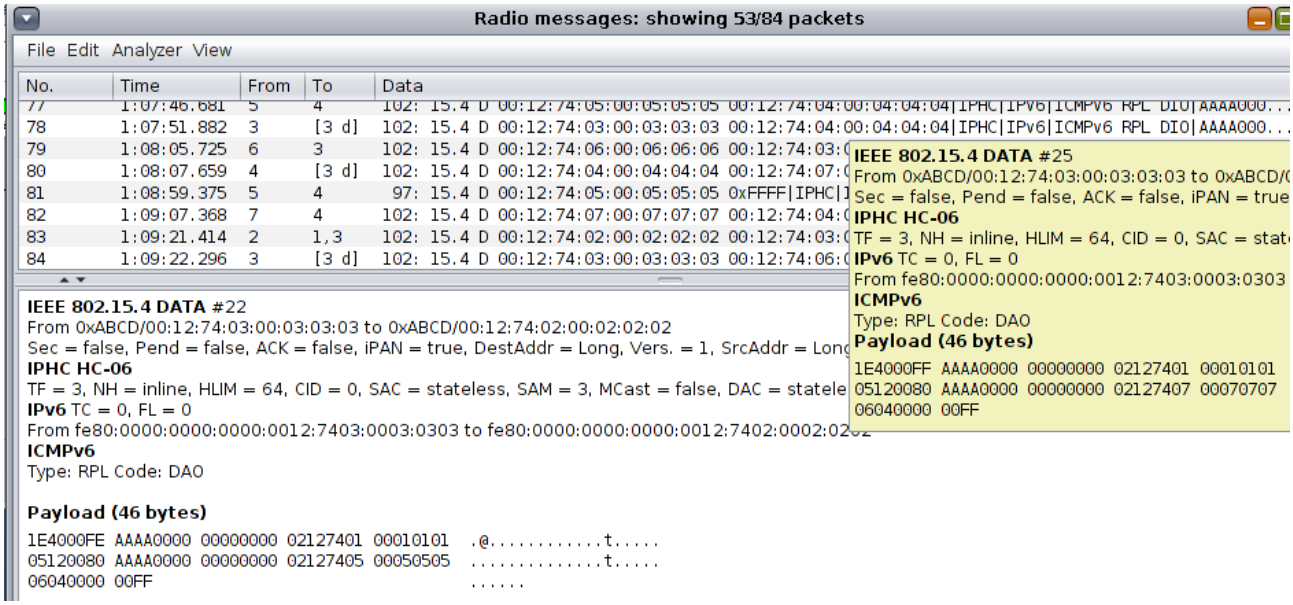

Рис. 17 Содержимое пакета 6LoWPAN

Отчет по работе загрузите на проверку.

7. Темы для самостоятельного изучения

К самостоятельному изучению предлагается изучение вопросов создания и отладки проектов умного дома на базе контроллеров Arduino и NodeMCU.

Темы:

- Установка программного обеспечения.
- Подключение датчиков.
- Отображение показаний и индикация состояния датчиков.
- Управление исполнительными устройствами.
- Создание будильников запуска исполнительных устройств по расписанию.
- Организация подключения к сети Интернет.
- Протокол MQTT – простой протокол для Интернета вещей.

Литература:

1. Петин, В. А. Создание умного дома на базе Arduino / В. А. Петин. — Москва : ДМК Пресс, 2018. — 180 с.

ЗАКЛЮЧЕНИЕ

Учебно-методическое пособие представляет собой ресурс, способствующий углубленному изучению основных аспектов цифровой обработки сигналов в контексте современных технологий мобильной связи и Интернета вещей.

Цель данного практикума заключается в приобретении студентами навыков работы с инструментами, позволяющими проектировать программно-аппаратные комплексы Интернета вещей, а также формирование компетенций в области существующих IoT-технологий и умений применять их к конкретным сценариям. Практикум направлен на развитие практических навыков студентов в применении полученных теоретических знаний для решения задач, связанных с построением, проектированием, функционированием и использованием устройств Интернета вещей.

В процессе изучения данного пособия студенты получат не только теоретические знания, но и практические навыки, которые пригодятся им в будущей профессиональной деятельности.

СПИСОК ИСТОЧНИКОВ

1. Инструмент потокового программирования Node-RED. – URL: https://nodered.org/about/. – Режим доступа: свободный (дата обращения 16.10.2023)
2. Jupyter Notebook с интерпретатором Python. – URL: https://www.portabledevapps.net/jupyter-portable.php. – Режим доступа: свободный (дата обращения 16.10.2023)
3. Внешний клиент MobaXterm. – URL: https://download.mobatek.net/2402024022512842/MobaXterm_Portable_v24.0.zip. – Режим доступа: свободный (дата обращения 16.10.2023)
4. Образ виртуальной машины. – URL: https://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-20.04.1-legacy-server-amd64-ovf.zip. – Режим доступа: свободный (дата обращения 16.10.2023)