



Geekbrains

**Создание системы визуализации данных с использованием
MQTT-протокола в качестве транспорта с оконечного
оборудования**

Программа:
Разработчик – Инженер умных устройств
Шуляк Олег Эдуардович

Минск
2024 г.

Содержание

Содержание	2
Введение	3
Глава 1. Основы разработки системы визуализации как решение IoT	4
1.1 Платформа интернет вещей IoT	4
1.2 Протоколы передачи данных и сетевые протоколы в IoT	6
1.3 Выбор протоколов для реализации поставленной задачи	9
1.4 Виртуализация. Выбор приложений для развертывания виртуальной машины	11
1.5 Состав и описания решения по системе визуализации	14
1.6 Выбор оконечного оборудования	17
1.7 Заключение по главе 1	18
Глава 2. Развертывание системы визуализации IoT. Описание работы системы	19
2.1 Регистрация доменного имени. Создание зависимостей зон	19
2.2 Развертывание виртуальной машины	21
2.3 Изменение настроек загрузчика VM. Настройка системы VM	23
2.4 Установка программных компонентов системы визуализации	25
2.5 Настройка роутера. Проброс портов виртуальной машины	28
2.6 Настройка и подключение оконечного оборудования к системе визуализации	32
2.7 Создание страницы на хостинге для быстрого доступа к приложениям	33
2.8 Авторизация в приложениях. Настройка приложений. Создание зависимостей в системе визуализации	36
2.9 Тест узлов системы визуализации	41
2.10 Заключение по главе 2	44
Заключение	45
Список использованных источников	46
Приложения	47

Введение

Тема проекта: Создание системы визуализации данных с использованием MQTT-протокола в качестве транспорта с оконечного оборудования.

Цель: Создать систему визуализации данных с использованием MQTT-протокола в качестве транспорта оконечного оборудования с возможностью расширения функционала системы.

Какую проблему решает: в процессе проекта создается скрипт с быстрым развертыванием системы визуализации, которая будет пригодна как для тестовых реализаций решений IoT, так и для создания реальных проектов.

Задачи:

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные принципы развертывания систем визуализации в IoT.
3. Определиться с основными компонентами/приложениями для создания собственной системы визуализации, ознакомиться с их функционалом.
4. Разработать структуру взаимодействия между компонентами системы визуализации.
5. Собрать систему с закреплением полученных результатов, описание работы системы визуализации.

Инструменты: хостинг-сервис my.hostfly.by, VMware Workstation 17 Player, Termius V9.1.1, Debian 12.x 64-bit, MQTT broker Mosquitto, Telegraf, Node-Red, InfluxDB, Grafana, Nginx, Wokwi - Online ESP32 Simulator

Состав команды: Шуляк Олег Эдуардович (Разработчик)

Глава 1. Основы разработки системы визуализации как решение IoT

1.1 Платформа интернет вещей IoT

Интернет вещей (Internet of Things, IoT) – новейший этап длительной и еще не закончившейся революции в области вычислительных систем и средств связи.

IoT – это термин, которым обозначается постоянно разрастающийся комплекс подключенных друг к другу интеллектуальных устройств, которые варьируются от обычных бытовых предметов до сложных промышленных инструментов и комплексов.

Интернет вещей (IoT) описывает сеть физических объектов – "вещей", – которые оснащены датчиками, программным обеспечением и технологиями для подключения и передачи данных, как другим устройствам, так и прикладными системами через сеть, с целью обработки полученных данных, информирования пользователей о различных событиях или автоматизации действий.

Основной темой здесь является дооснащение мобильными передатчиками различного радиуса действия, с микроконтроллером на борту, разнообразных физических вещей, предметов, промышленных установок, что открывает новые формы коммуникации между людьми и разными вещами.

Когда устройства IoT собирают и передают данные, их конечная цель состоит в том, чтобы извлечь из получаемых метрик как можно больше полезной информации, стремясь предоставлять пользователям данные, на основании которых можно строить различную логику по дальнейшим действиям.

Процесс работы с данными в IoT:

1. Сбор данных. С помощью датчиков IoT-устройства собирают данные из окружающей среды. Это может быть как простая температура, сложная видеозапись в реальном времени, так и параметры производственного станка через подключение по промышленному интерфейсу.

2. Обмен данными. Используя доступные сетевые соединения, как проводные, так и беспроводные, IoT-устройства отправляют эти данные в общедоступную или частную облачную систему (устройство-система-устройство) или на другое устройство (устройство-устройство).

3. Обработка данных. На этом этапе программное обеспечение программируется на то, чтобы сделать что-то на основании полученных данных – например, включить вентилятор или отправить предупреждение.

4. Действие на основе данных. Накопленные данные со всех IoT-устройств в сети анализируются. Это позволяет получать достаточно полезной и важной информации

для принятия действий и построения различной бизнес-логики. Например, предиктивное обслуживание.

"Вещами" в интернете вещей являются, главным образом, глубоко встроенные устройства с такими отличительными особенностями, как:

- узкая полоса пропускания;
- сбор данных с низкой повторяемостью;
- и малый объем используемых данных.

Несмотря на все приведенные качества, некоторые IoT-устройства, например охранные видеокамеры высокого разрешения, которые тоже являются частью IoT, требуют для работы довольно большой пропускной способности и это нормально. Но абсолютное большинство других устройств требует передачи пакетов данных всего лишь время от времени.

1.2 Протоколы передачи данных и сетевые протоколы в IoT

Протоколы данных и сетевые протоколы необходимы для IoT, поскольку они обеспечивают связь между устройствами и системами стандартизированным и эффективным образом.

- Протоколы данных:

Протоколы передачи данных, такие как MQTT, CoAP, AMQP и iBeacon, определяют формат, структуру и правила обмена данными между устройствами, приложениями и сервисами IoT. Они обеспечивают безопасную, надежную и эффективную передачу данных независимо от типа данных, местоположения устройств и состояния сети. Протоколы данных также поддерживают различные схемы передачи данных, такие как публикация-подписка, запрос-ответ и событийно-ориентированные, что очень важно для приложений IoT, требующих обработки данных в реальном времени, аналитики и принятия решений.

Протоколы передачи данных являются важной частью экосистемы Интернета вещей (IoT), поскольку они позволяют устройствам обмениваться данными и общаться друг с другом. Существует несколько типов протоколов передачи данных, используемых в IoT, каждый из которых имеет свои уникальные особенности и преимущества.

Одним из наиболее широко используемых протоколов передачи данных в IoT является MQTT (Message Queuing Telemetry Transport). MQTT – это легкий протокол обмена сообщениями, который разработан для обеспечения эффективности, надежности и безопасности. Он использует схему публикации/подписки сообщений, при которой сообщения отправляются в определенные темы, а устройства, заинтересованные в этих темах, могут подписаться на получение этих сообщений. В качестве примера можно привести облачное дисковое хранилище, где есть группа людей с доступом к какой-либо папке. Как только администратор загрузит файл в эту папку, он сразу же станет доступен всем участникам. Только в mqtt вместо папки - топик, а вместо файла - сообщение с данными. Если говорить о самом mqtt, то представьте себе систему умного дома, светильники которого подключены к различным топикам, каждый из которых ответственен за определенную комнату. Вы в приложении нажимаете на кнопку “включить свет в гостиной”, и все устройства в гостиной включают свет.

Другим широко используемым протоколом передачи данных является CoAP (Constrained Application Protocol). CoAP – это легкий протокол, предназначенный для использования в средах с ограниченными ресурсами, таких как устройства IoT с ограниченной вычислительной мощностью и памятью. Он использует модель клиент/сервер и поддерживает различные методы взаимодействия с ресурсами, такие как GET, PUT, POST и DELETE.

AMQP (Advanced Message Queuing Protocol) – еще один популярный протокол передачи данных, который используется в IoT. Это протокол обмена сообщениями, который разработан для обеспечения надежности и совместимости с широким спектром платформ и языков программирования. AMQP поддерживает различные схемы обмена сообщениями,

включая "точка-точка" (напрямую устройству) и "публикация/подписка", и может использоваться как с TCP/IP, так и с другими транспортными протоколами.

iBeacon – это собственный протокол, разработанный компанией Apple, который используется для предоставления услуг, основанных на определении местоположения. Он разработан для работы с Bluetooth Low Energy (BLE) и использует широкоэвещательную схему передачи сообщений, когда устройство посылает сигнал, который может быть принят другими устройствами, находящимися поблизости. iBeacon обычно используется в розничной торговле для отправки целевой рекламы и рекламных акций покупателям на основе их местоположения в магазине.

- Сетевые протоколы:

Сетевые (и беспроводные) протоколы, такие как Wi-Fi, Bluetooth, ZigBee, LoRaWAN, SigFox, NB-IoT и RFID, обеспечивают инфраструктуру и средства для подключения устройств IoT к Интернету, облаку или другим устройствам. Они определяют правила и стандарты передачи данных по различным типам беспроводных и проводных сетей, таких как сотовая связь, Wi-Fi, Ethernet и LPWAN. Сетевые протоколы также обеспечивают связь устройств IoT друг с другом и с другими устройствами независимо от производителя, операционной системы или приложения.

Сетевые протоколы используются для обеспечения связи между устройствами в сети IoT. Существуют различные типы сетевых протоколов, включая проводные, беспроводные и протоколы маломощных глобальных сетей (LPWAN).

Проводные сетевые протоколы, такие как Ethernet, используют физические кабели для соединения устройств. Ethernet – это популярный протокол для подключения устройств в локальных сетях (LAN). Он обеспечивает надежную и высокоскоростную связь, что делает его пригодным для использования в приложениях, требующих обмена данными в реальном времени, таких как промышленная автоматизация, видеонаблюдение, потоковое аудио и видео.

Протоколы беспроводных сетей, с другой стороны, используют радиоволны для обеспечения связи между устройствами. Некоторые из наиболее распространенных протоколов беспроводных сетей включают сотовую связь, Wi-Fi, Bluetooth, ZigBee, NFC и RFID.

Сотовые протоколы, такие как 2G, 3G и 4G, широко используются для подключения мобильных устройств к Интернету. Они обеспечивают высокоскоростную передачу данных и широко доступны в городских и пригородных районах. Сотовые протоколы широко используются в таких приложениях, как подключенные автомобили, отслеживание активов и удаленный мониторинг.

Wi-Fi – это популярный беспроводной протокол, использующий радиоволны для подключения устройств к Интернету. Он обеспечивает высокоскоростную передачу данных и широко используется в домах, офисах и общественных местах. Wi-Fi широко используется в таких приложениях, как "умные дома", автоматизация офиса и общественные точки доступа Wi-Fi.

Bluetooth и BLE (Bluetooth Low Energy) – это беспроводные протоколы, обычно используемые для связи между устройствами на коротких расстояниях. Они широко используются в таких приложениях, как носимые устройства, мониторинг здоровья и устройства "умного дома". Bluetooth обеспечивает надежную и безопасную передачу данных,

а BLE – связь с низким энергопотреблением, что делает его пригодным для использования в устройствах с питанием от батарей.

ZigBee – это беспроводной протокол, разработанный для приложений с низкой скоростью передачи данных, требующих низкого энергопотребления, таких как системы домашней автоматизации и управления зданиями. Он обеспечивает надежную и безопасную связь и подходит для использования в крупномасштабных сетях.

NFC (Near Field Communication) и RFID (Radio Frequency Identification) – беспроводные протоколы, обычно используемые для связи между устройствами на коротких расстояниях. NFC обычно используется в таких приложениях, как бесконтактные платежи и продажа билетов, а RFID широко применяется в управлении цепочками поставок, отслеживании активов и управлении запасами.

Протоколы LPWAN, такие как LoRaWAN, Sigfox и NB-IoT, предназначены для обеспечения связи на большие расстояния с низким энергопотреблением. Они широко используются в таких приложениях, как "умные" города, "умное" сельское хозяйство и отслеживание активов. Протоколы LPWAN обеспечивают надежную и безопасную связь, что делает их пригодными для использования в устройствах с низкой пропускной способностью и питанием от батарей.

1.3 Выбор протоколов для реализации поставленной задачи

В п.п. 1.2 были описаны наиболее распространенные протоколы передачи данных и сетевые протоколы, используемые для реализации решений платформы IoT. Исходя из вышеперечисленного выбраны следующие протоколы:

- 1) **Сетевой протокол Ethernet** – это протокол связи, который существует с 1970-х годов и широко используется в проводных локальных сетях (LAN) и глобальных сетях (WAN). Это надежный и эффективный протокол, который позволяет устройствам взаимодействовать друг с другом по сети. Ethernet широко используется в приложениях IoT, поскольку это зрелая технология, стандартизированная в течение многих лет.

Ethernet работает на различных частотах, в зависимости от типа кабеля и используемой сети. В сетях Ethernet обычно используются следующие частоты: 10 Мбит/с, 100 Мбит/с и 1 Гбит/с. Ethernet может использовать как лицензированный, так и бесплатный спектр в зависимости от сети и местоположения. Ethernet обладает высокой пропускной способностью, что позволяет быстро и эффективно передавать данные между устройствами. Он также имеет низкое энергопотребление, что делает его идеальным для использования в устройствах IoT, работающих от батарей.

Задержка – это время, необходимое для передачи пакета данных от одного устройства к другому по сети. Ethernet имеет низкий уровень задержки, что означает, что данные передаются быстро и эффективно. Это важно для приложений IoT, где требуется связь в режиме реального времени, например, в системах промышленной автоматизации и управления.

Ethernet – это высоконадежный протокол, в котором предусмотрены механизмы проверки и исправления ошибок для обеспечения правильной передачи данных. Это также безопасный протокол, который может быть зашифрован для защиты данных от несанкционированного доступа. Однако одним из потенциальных недостатков Ethernet является то, что он уязвим к помехам от других электронных устройств и может быть подвержен влиянию шумов в сети.

- 2) **Беспроводной протокол Wi-Fi**, сокращение от "Wireless Fidelity" – это широко используемый протокол беспроводных сетей, который позволяет устройствам подключаться к Интернету и общаться друг с другом без использования кабелей. Wi-Fi работает в различных частотных диапазонах, включая 2,4 ГГц и 5 ГГц, в зависимости от устройства и его возможностей.

Доступная полоса пропускания для Wi-Fi зависит от частотного диапазона и количества используемых каналов. Обычно диапазон 2,4 ГГц обеспечивает пропускную способность до 20 МГц на канал, а диапазон 5 ГГц – до 160 МГц. Wi-Fi становится все более популярным протоколом связи в IoT благодаря своей надежности, низкой задержке и высокой пропускной способности.

Устройства с поддержкой Wi-Fi могут быстро и беспрепятственно взаимодействовать друг с другом, что делает его идеальным для приложений реального времени, таких как потоковое видео или онлайн-игры.

Однако энергопотребление Wi-Fi может быть относительно высоким по сравнению с другими протоколами беспроводной связи, что может быть недостатком для IoT-устройств с питанием от батарей. Кроме того, на сигналы Wi-Fi могут влиять помехи от других устройств, работающих в том же частотном диапазоне, что может привести к снижению надежности.

Несмотря на эти ограничения, Wi-Fi остается популярным выбором для IoT-приложений, требующих высокоскоростного и надежного беспроводного соединения. Некоторые распространенные случаи использования Wi-Fi в IoT включают домашнюю автоматизацию, "умные города" и промышленные приложения.

- 3) **Протокол передачи данных MQTT (Message Queuing Telemetry Transport)** – это легкий протокол обмена сообщениями, разработанный для IoT-устройств с ограниченной вычислительной мощностью и сетями с низкой пропускной способностью. MQTT – это протокол публикации-подписки, в котором устройства, подключенные к сети, могут публиковать сообщения на определенные темы или подписываться на получение сообщений на эти темы.

MQTT широко используется в приложениях IoT благодаря низким накладным расходам и эффективному использованию пропускной способности сети. Он работает поверх TCP/IP и может работать с несколькими транспортными протоколами, такими как Wi-Fi, Ethernet, сотовые и спутниковые сети. MQTT использует небольшой заголовок, состоящий всего из 2 байт, что делает его идеальным для устройств с ограниченной вычислительной мощностью.

Одной из ключевых особенностей MQTT является низкая задержка. Он разработан для минимизации задержек между отправкой и получением сообщений, что делает его идеальным для связи в реальном времени в таких приложениях, как домашняя автоматизация, промышленный контроль и мониторинг окружающей среды.

MQTT также отличается высокой надежностью благодаря встроенным функциям для обеспечения доставки сообщений. Он использует уровни качества обслуживания (QoS), которые позволяют устройствам определять, насколько важно сообщение и сколько раз оно должно быть доставлено для обеспечения надежности. MQTT может работать с уровнями QoS 0 (не более одного раза), 1 (не менее одного раза) и 2 (точно один раз).

Одной из сильных сторон MQTT является его гибкость. Он хорошо настраивается и может использоваться в различных приложениях, от маломощных датчиков до высокопроизводительных промышленных систем управления. MQTT также поддерживает двунаправленную связь, что означает, что устройства могут как публиковать, так и подписываться на сообщения, обеспечивая более сложное взаимодействие между устройствами.

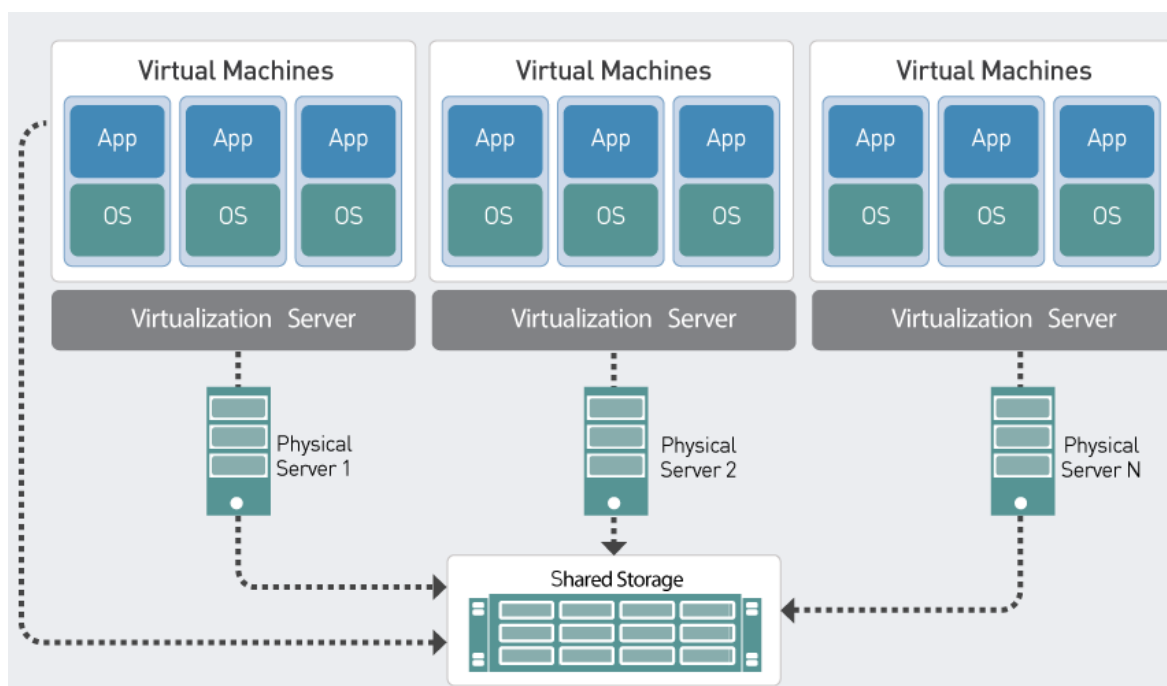
Однако одним из потенциальных недостатков MQTT является отсутствие встроенных функций безопасности, что означает, что он может быть уязвим для таких атак, как подслушивание или фальсификация данных. Чтобы решить эту проблему, MQTT можно использовать вместе с Transport Layer Security (TLS) или другими протоколами безопасности для обеспечения безопасного соединения между устройствами.

Некоторые примеры использования MQTT включают в себя устройства "умного дома", мониторинг окружающей среды и промышленную автоматизацию. Например, умный термостат в доме может использовать MQTT для связи с облачной службой, позволяя владельцу дома дистанционно управлять температурой в своем доме. В промышленной автоматизации MQTT может использоваться для мониторинга и управления машинами на заводе, позволяя анализировать данные в режиме реального времени и принимать решения.

MQTT является популярным и широко используемым протоколом в сфере IoT благодаря своим низким накладным расходам, низкой задержке и высокой надежности. Гибкость и настраиваемые функции делают его подходящим для широкого спектра приложений, от маломощных датчиков до высокопроизводительных промышленных систем управления.

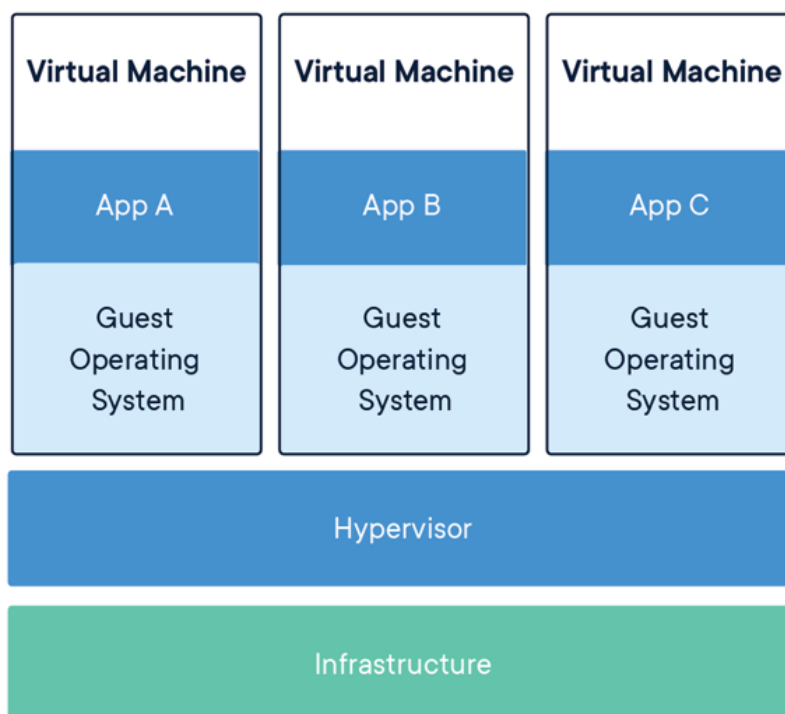
1.4 Виртуализация. Выбор приложений для развертывания виртуальной машины

Виртуализация представляет собой современную технологию, которая позволяет создавать несколько виртуальных компьютеров на одном физическом сервере. Её основной концепцией является эффективное распределение ресурсов одного физического компьютера между разными средами, что позволяет размещать различные операционные системы и приложения в одной общей среде. Этот подход устраняет ограничения, связанные с физическими и географическими ограничениями. Виртуализация способствует оптимальному использованию аппаратных ресурсов, снижению издержек, обеспечивает повышенную доступность, облегчает управление и улучшает систему восстановления данных в случае критических ситуаций.



Пример виртуализации на сервере

Важной составной частью виртуализации серверов являются виртуальные машины (VM). Они представляют собой виртуальные компьютеры, обладающие всеми характеристиками реальных физических компьютеров. Операционные системы и приложения работают на виртуальных машинах так же, как на настоящих компьютерах. Однако, вместо использования физического оборудования, виртуальные машины представлены в виде компьютерных файлов, которые запускаются на физическом сервере. Их поведение идентично поведению реальных компьютеров, что позволяет использовать виртуальные машины в качестве отдельных компьютерных систем.



Архитектура виртуальных машин

Основная цель виртуализации заключается в разделении программного обеспечения и аппаратной части. В рамках информационных технологий виртуализация предоставляет абстракцию вычислительных ресурсов и создает систему, которая скрывает свою физическую реализацию от пользователя. Это позволяет пользователю взаимодействовать с объектами, используя удобное для него представление, не беспокоясь о том, как объект физически устроен.

Принцип работы виртуализации серверов заключается в гибком распределении ресурсов физического сервера между виртуальными машинами. Каждая виртуальная машина воспринимает только те ресурсы, которые ей выделены, и функционирует, будто бы она работает на отдельном физическом сервере. Этот метод позволяет одновременно обеспечивать производительность, доступность и безопасность приложений на сервере в режиме "один сервер – множество приложений".

Широко применяемая виртуализация дает возможность запускать различные операционные системы на отдельных виртуальных машинах, имитируя их запросы к аппаратным ресурсам сервера.

Гипервизор, специальная программа (по сути, операционная система), отвечает за создание и управление виртуальными машинами. Он обеспечивает изоляцию между операционными системами, гарантирует безопасность и управляет распределением ресурсов между работающими операционными системами. Гипервизор может работать непосредственно с аппаратным оборудованием, без основной операционной системы на хост-машине (аппаратная виртуализация), либо через основную операционную систему (программная виртуализация). Обычно на домашних компьютерах используется второй тип гипервизора.

В качестве гипервизора принято решение использовать **VMware Workstation 17 Player** (*источник 2*) — бесплатный для некоммерческого использования программный продукт, на основе виртуальной машины VMware Workstation, но с ограниченной функциональностью, предназначенный для запуска образов виртуальных машин, созданных в других продуктах

VMware, а также в Microsoft VirtualPC и Symantec LiveState Recovery. Начиная с версии 3.0 VMware Player позволяет также создавать образы виртуальных машин. Ограничение функциональности теперь касается в основном функций, предназначенных для IT-специалистов и разработчиков ПО. Например, отсутствует возможность тонкого настраивания виртуальных сетевых адаптеров через Virtual Network Editor.

Операционная система на виртуальной машине **Debian 12.x 64-bit (источник 3)** — операционная система, состоящая из свободного ПО с открытым исходным кодом. В настоящее время Debian GNU/Linux — один из самых популярных и важных дистрибутивов GNU/Linux, в первичной форме оказавший значительное влияние на развитие этого типа ОС в целом.

Для удаленного подключения с основной операционной системы используем **Termius (источник 4)** — это кроссплатформенный клиент SSH, который работает на настольных компьютерах и мобильных устройствах. Этот клиент SSH позволяет организовывать хосты в группы. Группы позволяют вам обмениваться настройками, хотя каждый хост может иметь свои собственные предпочтения.

1.5 Состав и описания решения по системе визуализации

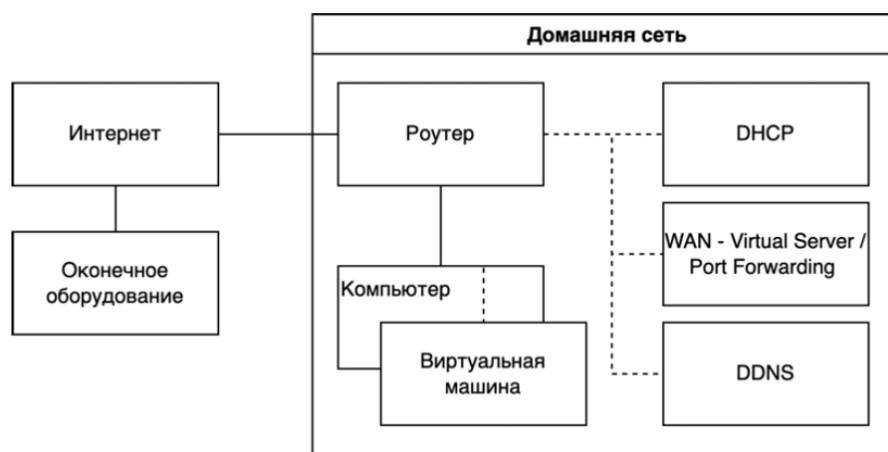


Схема решения

На данном рисунке представлена общая схема решения, включающая в себя оконечное оборудование с доступом в интернет, компьютер, с развернутой на нем виртуальной машиной и доступом в домашнюю локальную сеть. Центром домашней локальной сети является роутер, который может обладать дополнительным функционалом в части Port Forwarding'а и DDNS'а, частных случаев применения. Далее рассмотрим назначение этих программных компонентов.

Функция "**Динамический DNS**" (*Dynamic DNS / DDNS*) позволяет присвоить постоянное доменное имя (адрес для доступа из интернета) вашему публичному IP-адресу, который роутер получает от интернет-провайдера. Публичный IP-адрес может поменяться, а доменное имя, которое вы регистрируете – нет. (Есть сервисы, позволяющие сделать это бесплатно и вне ПО роутера, рассматриваемого в рамках занятия). Это будет необходимо для того, чтобы получать доступ к устанавливаемым сервисам через интернет с любого устройства по доменному имени. И порядком упростить конфигурацию, потому что не нужно прописывать конкретный IP адрес в различных конфигурациях ПО.

Далее, **Port Forwarding (или Virtual Server)** – настройка для WAN-интерфейса (который отвечает за подключение к интернету), позволяющая извне (через интернет) обращаться к конкретным клиентам локальной сети (например, к нашей виртуальной машине) по определенному порту приложения. Таким образом, можно на любом устройстве открыть браузер, ввести зарегистрированное доменное имя и указать порт необходимого приложения для входа на приложение своего сервера.

DHCP — протокол прикладного уровня модели TCP/IP, служит для назначения IP-адреса клиенту. Это следует из его названия — Dynamic Host Configuration Protocol. IP-адрес можно назначать вручную каждому клиенту, то есть компьютеру в локальной сети. Но в больших сетях это очень трудозатратно, к тому же, чем больше локальная сеть, тем выше возрастает вероятность ошибки при настройке. Поэтому для автоматизации назначения IP был создан протокол DHCP.

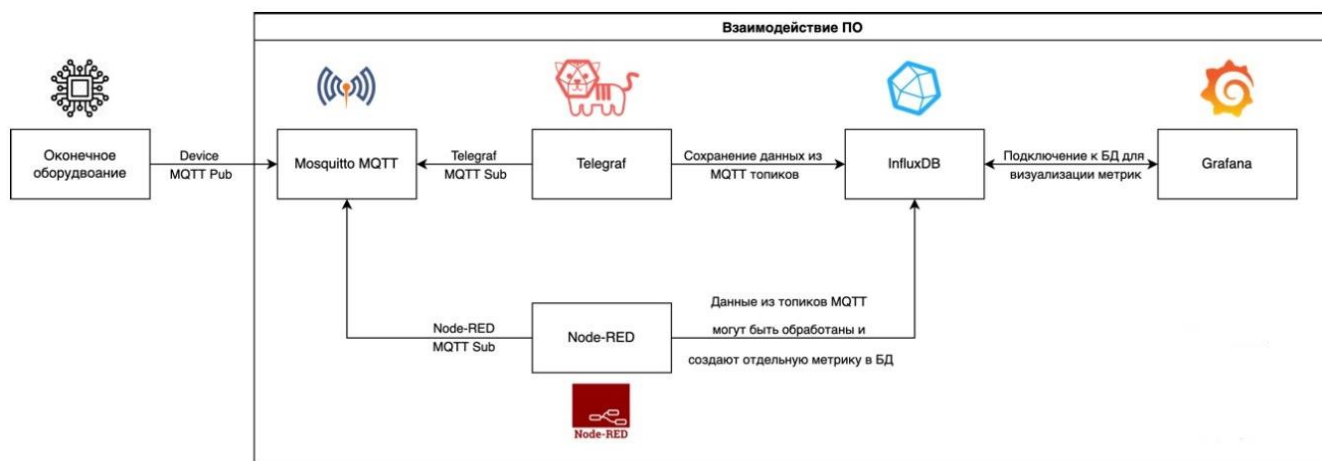


Схема взаимодействия программных компонентов решения по визуализации

1. Оконечное оборудование.

Собирает телеметрию, доступом в интернет, разрабатываемое студентами индивидуально. На конечном оборудовании реализуется MQTT клиент, который публикует в определённый топик MQTT брокера некую информацию, например, температуру окружающей среды в градусах цельсия.

2. Mosquitto MQTT broker.

MQTT брокер служит неким посредником, который позволяет собирать (принимать) информацию от устройств в определенные топики и сразу же «отдавать» ее клиентам, которые эти топики слушают для сбора, обработки, анализа, визуализации получаемых данных.

3. Telegraf.

Чтобы получить информацию от MQTT брокера, также необходим клиент, который подписывается на топики брокера для получения данных, реализуя тем самым паттерн Издатель-подписчик (англ. publisher-subscriber или англ. pub/sub). Для этого и используется Telegraf. Он, как клиент-подписчик, прослушивает заданные MQTT топики и, получив сообщение, сразу же записывает информацию в InfluxDB. Вообще, Telegraf – это не только MQTT клиент. Telegraf – это серверный агент для сбора, обработки, агрегирования и записи метрик из различных стеков, датчиков и систем в InfluxDB.

4. InfluxDB.

InfluxDB – это система баз данных, оптимизированная для предоставления данных временных рядов и их хранения в привязке ко времени и значению (иными словами база данных временных рядов (time series database – TSDB)), разработанная компанией InfluxData. Временной ряд – это совокупность наблюдений за четко определенными элементами данных, полученных в результате повторяющихся измерений в течение определенного времени. Данные временного ряда индексируются во временном порядке, который представляет собой последовательность точек данных. В InfluxDB есть встроенный конструктор дашбордов для визуализации – Chronograf, на котором можно было бы и остановиться, но он больше сделан для быстрого прототипирования дашбордов и проверки отображения тех или иных метрик в том или ином виде в самой InfluxDB. Гораздо удобнее использовать платформу Grafana, которая является плюс минус стандартном в вопросах визуализации (особенно в сфере IoT). Для этого будет настраиваться интеграция в Grafana на подключение к InfluxDB.

5. Grafana.

Grafana – программное обеспечение с открытым исходным кодом, позволяющее запрашивать, визуализировать и исследовать метрики (данные), где бы они ни хранились. По сути, предоставляет пользователю инструменты для преобразования информации из базы данных временных рядов (TSDB) в удобные графики и визуализации.

6. Node-RED.

Это инструмент визуального программирования для интернета вещей, позволяющий подключать друг к другу устройства, API и онлайн-сервисы. Выступает в роли многофункционального бекэнда с поддержкой множества интерфейсов и протоколов. Звездочка здесь стоит только потому, что данный компонент не обязателен, но всегда должна быть альтернатива, и Node-RED позволяет собирать и передавать данные альтернативным от Telegraf способом, имея мощный функционал промежуточной обработки.

Для развертывания программных компонентов используем платформу для контейнеризации **Docker (источник 5)** – это платформа для разработки, доставки и запуска контейнерных приложений. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

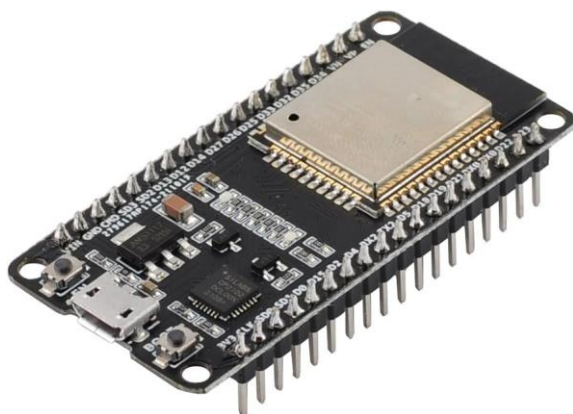
1.6 Выбор окончечного оборудования

В качестве микроконтроллера для обработки и передачи данных будет использоваться плата **ESP32** – “классическая” серия ESP32. Выпущенная в 2016 году на базе двухъядерного процессора Xtensa® 32-bit LX6 (или одноядерного в некоторых вариантах) с максимальной тактовой частотой 240 МГц. Работает с использованием относительно старых протоколов WiFi 2.4 МГц. Технические характеристики платы:

- Два (или одно) ядра ЦП с изменяемой тактовой частотой 80 / 160 / 240 МГц.
- Выходная мощность WiFi +19,5 дБм обеспечивает достаточную дальность связи.
- Классический Bluetooth с поддержкой L2CAP, SDP, GAP, SMP, AVDTP, AVCTP, A2DP (SNK) и AVRCP (CT).
- Поддержка Bluetooth Low Energy (Bluetooth LE), включая профили L2CAP, GAP, GATT, SMP и GATT, такие как BluFi, SPP-подобные и т. д.
- Ток сна составляет менее 5 мкА, что делает его пригодным для приложений с батарейным питанием и носимой электроники.
- Периферийные устройства включают емкостные сенсорные датчики, датчик Холла, интерфейс SD-карты, Ethernet, высокоскоростной SPI, UART, I2S и I2C.
- Внешняя RAM и флэш-память может быть подключена только через относительно небольшое “окно” в адресном пространстве (без DMA).

К микроконтроллеру будет подключен датчик **DHT22 Arduino** – это цифровой датчик влажности и температуры на микросхеме AM2302. Датчик предназначен для измерения с высокой точностью, температуры и влажности окружающей среды. Датчик DHT22 выполнен в виде модуля и содержит всю необходимую обвязку, для подключения его к контроллеру Arduino. Технические характеристики датчика:

- Напряжение питания: 3 .. 6 В.
- Диапазон измерения влажности: 0 .. 100%.
- Диапазон измерения температуры: -40°C .. +80°C.
- Погрешность измерений влажности: 2%.
- Погрешность измерений температуры: 0.5°C.
- Шаг измерения влажности: 0.1%.
- Шаг измерения температуры: 0.1 °C.
- Период измерений: 1 сек.
- Интерфейс: onewire, 1-проводной.
- Шаг выводов: 2,54 мм.



Плата ESP32



Датчик DHT22 Arduino

1.7 Заключение по главе 1

В данной главе были описаны основные протоколы передачи данных и сетевые протоколы, а также выбраны протоколы для решения поставленной задачи:

- *Сетевой протокол Ethernet.*
- *Беспроводной протокол Wi-Fi.*
- *Протокол передачи данных MQTT (Message Queuing Telemetry Transport).*

Далее разработана схема для разработки системы визуализации и схема взаимодействия программных компонентов, которые разворачиваются на виртуальную машину с операционной системой Debian 12.x при помощи платформы Docker. Программные компоненты следующие:

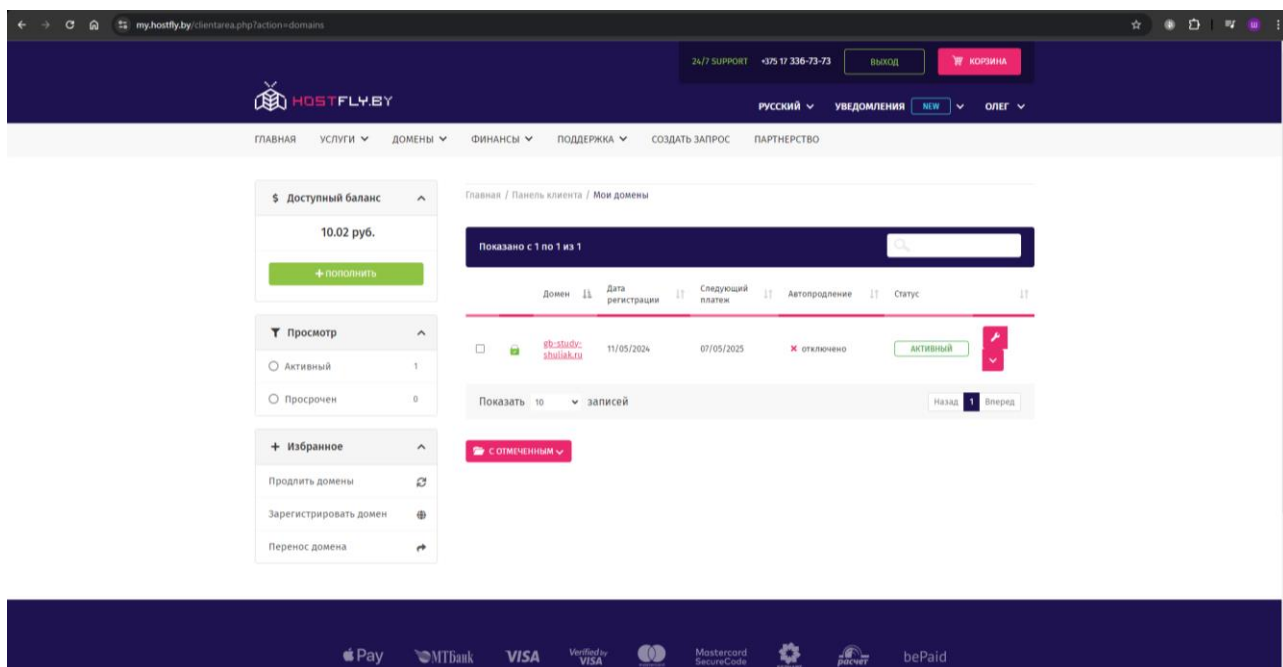
- *Mosquitto MQTT broker.*
- *Telegraf.*
- *InfluxDB.*
- *Grafana.*
- *Node-RED.*

В качестве окончательного оборудования выбран датчик DHT22 подключенный к плате ESP32.

Глава 2. Развертывание системы визуализации IoT. Описание работы системы

2.1 Регистрация доменного имени. Создание зависимостей 30Н

Для регистрации доменного имени был выбран хостинг-сервис *my.hostfly.by*. Доменное имя *gb-study-shuliak.ru*.



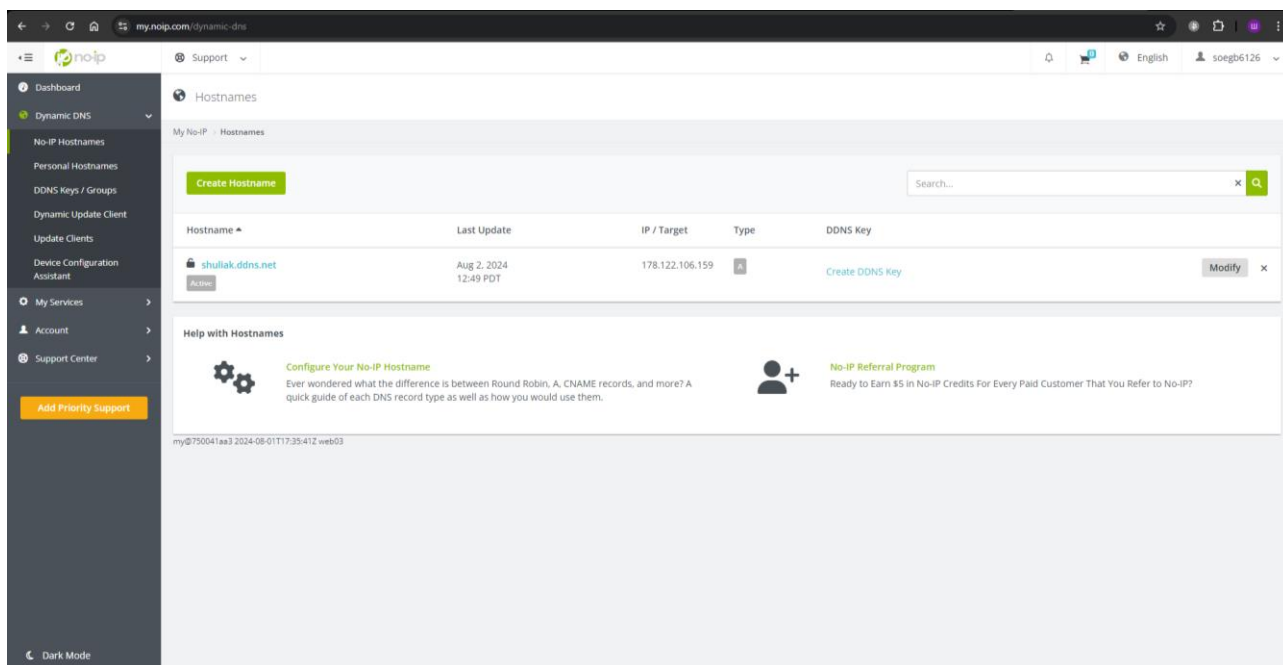
Личный кабинет хостинг-сервиса *my.hostfly.by*

Далее был приобретено место на хостинге с привязкой к вышеуказанному доменному имени, на котором будут отредактированы зоны для доступа к приложениям виртуальной машины через сеть интернет.

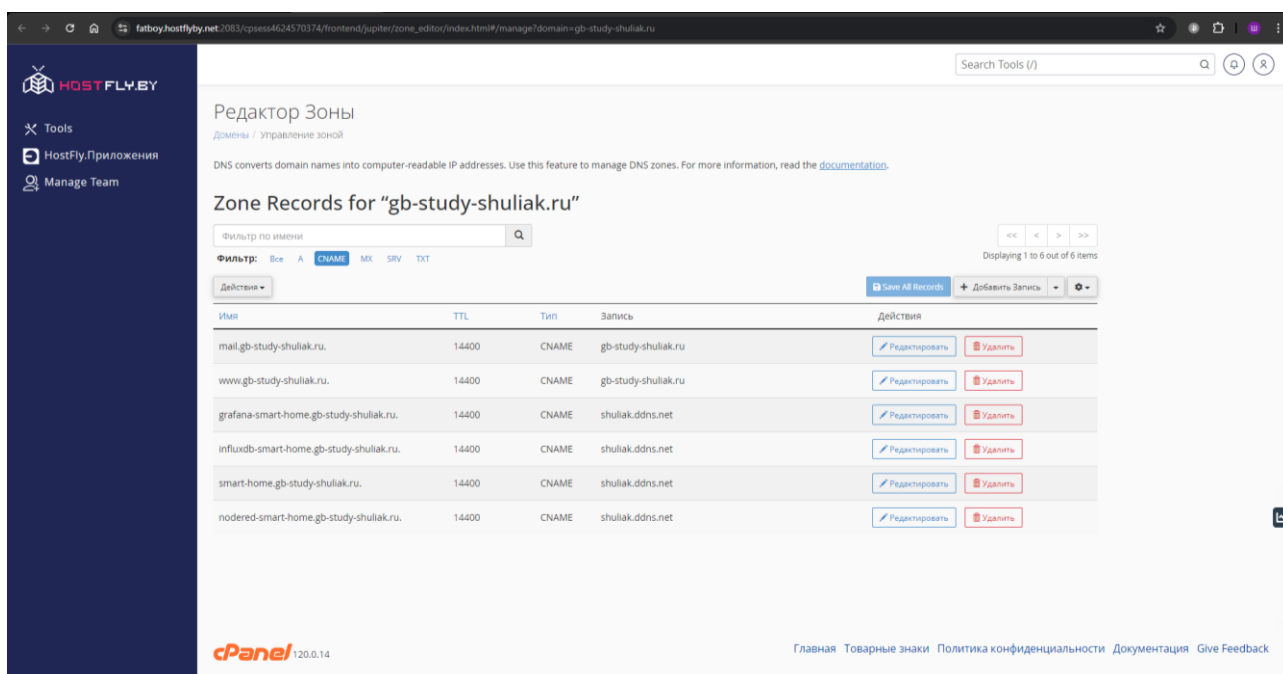
Также был создан аккаунт на сайте www.noip.com для создания собственного доменного имени, для возможности использования функции DDNS. Полученное доменное имя *shuliak.ddns.net*, к которому присвоен текущий IP роутера (текущий IP узнаем на сайте <https://2ip.ru/>). В дальнейшем Функция DDNS будет подтягивать актуальные IP адреса роутера, в случае их изменения.

На хостинг-сервисе *my.hostfly.by* создаем записи соответствия вида CNAME в редакторе зон домена с доменными именами для доступа к приложениям системы визуализации с функцией DDNS (соответствия с доменным именем *shuliak.ddns.net*):

- *grafana-smart-home.gb-study-shuliak.ru*.
- *influxdb-smart-home.gb-study-shuliak.ru*.
- *nodered-smart-home.gb-study-shuliak.ru*.



Личный кабинет с доменным именем на сайте noip.com



Создание соответствий CNAME

2.2 Развертывание виртуальной машины

Для развертывания виртуальной машины предварительно необходимо скачать приложения (в нашем случае для Windows):

- *VMware Workstation 17 Player (источник 2).*
- *Termius (источник 4).*

Устанавливаем данные приложения. Далее необходимо скачать файл образ операционной системы *Debian 12.x 64-bit (источник 3).*

Запускаем виртуальную машину VMware Workstation 17 Player и начинаем установку операционной системы Debian, при установке указываем скачанный файл образа операционной системы.

При установке были заданы следующие технические характеристики виртуальной машины:

- *Оперативная память: 2 GB.*
- *Количество ядер процессора: 2 ядра.*
- *Максимальный объем жесткого диска: 20 GB.*
- *Сетевой адаптер: Bridged (с указанием реального адаптера с выходом в Интернет).*

В процессе установки была выбрана экспертная установка с отменой установки ненужных компонентов для уменьшения занимаемого объема виртуальной машины на жестком диске. В ходе установки:

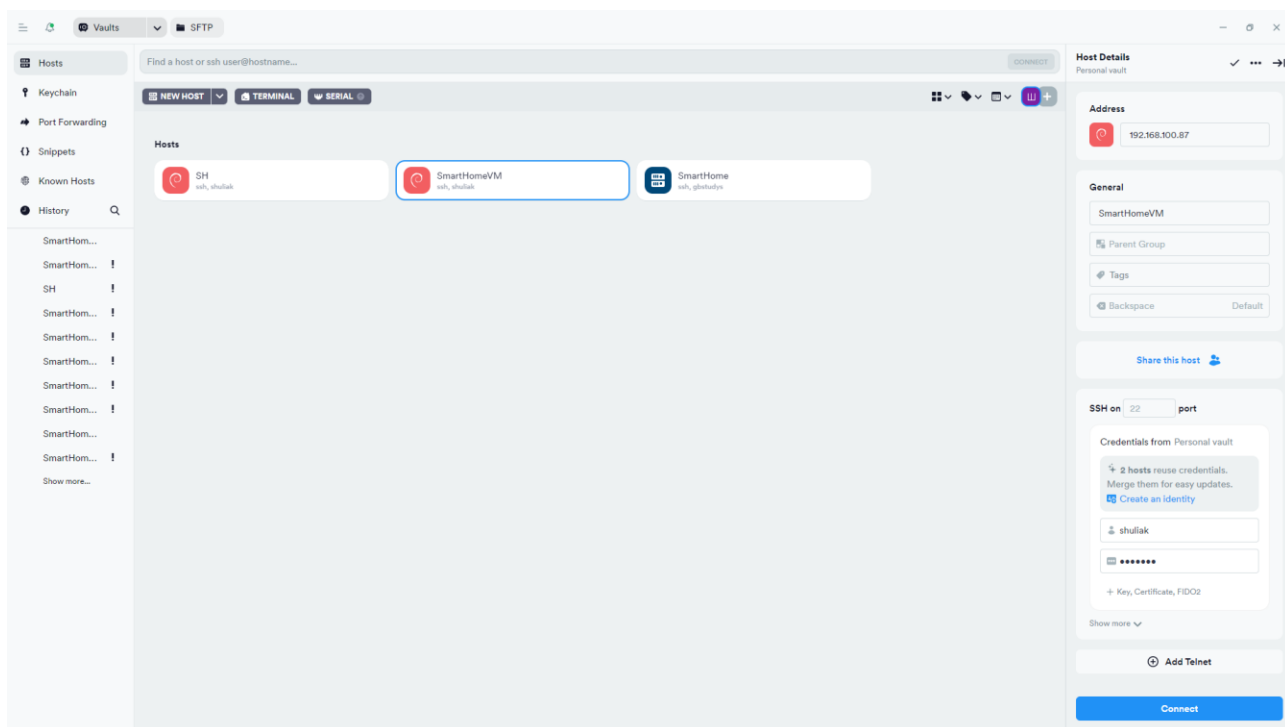
- *Задаем и сохраняем имя пользователя и пароль к нему (приложение 1).*
- *Устанавливаем SSH клиент.*
- *Указываем на автоматическую настройку сетевых параметров.*
- *Указываем установку low-мемори для экономии памяти.*

После окончания установки запускаем виртуальную машину, вводим имя пользователя и пароль. Далее вводим команду *ip a* для получения адреса виртуальной машины (см. строку *ens33*), сохраняем его в файл (*приложение 1*). В нашем случае IP : 192.168.100.87.

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:90:f0:e1 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.100.87/24 brd 192.168.100.255 scope global dynamic ens33
        valid_lft 258960sec preferred_lft 258960sec
    inet6 fe80::20c:29ff:fe90:f0e1/64 scope link
        valid_lft forever preferred_lft forever
```

Первый вход в систему Debian, результат команды ip a

Запускаем ранее установленное приложение Termius. Создаем новое подключение по SSH выбрав в меню кнопку NEW HOST. Далее вводим необходимые данные: IP виртуальной машины, название подключения, логин и пароль виртуальной машины, подключаемся.

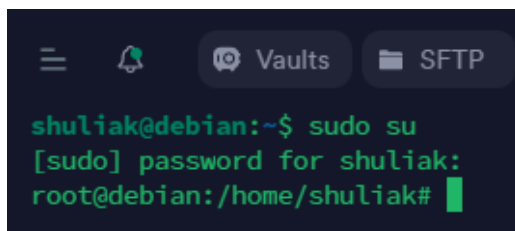


Настройка подключения по SSH через клиент Termius

Дальнейшую настройку ВМ и развертывание программных компонентов осуществляем через SSH-клиент Termius.

2.3 Изменение настроек загрузчика ВМ. Настройка системы ВМ.

Для изменения настроек системы необходимо авторизоваться под пользователем *root* (суперпользователь) командой `sudo su`, далее вводим логин и пароль вашего пользователя. После успешного ввода в терминале пользователь должен смениться на *root*.



Подключение с правами root

Создаем файл конфигурации загрузчика блоком команд (при помощи утилиты `cat`) (источник 2):

```
root@debian:/home/shuliak# cat > /etc/default/grub <<EOF
GRUB_DEFAULT=0
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet text mitigations=off nowatchdog processor.ignore_ppc=1 cpufreq.default_governor=performance debug=-1"
GRUB_CMDLINE_LINUX=""
GRUB_DISABLE_LINUX_RECOVERY=true
GRUB_DISABLE_OS_PROBER=true
GRUB_TERMINAL=console
EOF
root@debian:/home/shuliak#
```

Блок команд для записи файла grub

В созданном файле конфигурации мы отключаем таймауты загрузки операционной системы, отключаем поиск других операционных систем и переводим режим работы с системой в терминал.

Далее выполняем блок команд для установки недостающих компонентов и настройки системы (источник 2):

```
root@debian:/home/shuliak# echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
echo "net.ipv4.conf.all.forwarding=1" >> /etc/sysctl.conf && \
sysctl -p /etc/sysctl.conf && \
systemctl stop cron && \
systemctl disable cron && \
systemctl set-default multi-user.target && \
mkdir -m 0755 -p /etc/apt/keyrings && \
apt install -y ca-certificates curl wget gnupg gnupg2 systemd-timesyncd http lm-sensors lsb-release unzip && \
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg && \
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null && \
sudo wget -qO- https://repo.mosquitto.org/debian/mosquitto-repo.gpg.key | gpg --dearmor -o /etc/apt/keyrings/mosquitto-repo.gpg && \
sudo wget -O /etc/apt/sources.list.d/mosquitto-bookworm.list https://repo.mosquitto.org/debian/mosquitto-bookworm.list && \
apt-get update && \
apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin mosquitto-clients=2.0.18-mosquitto-bookworm && \
sed -i 's/#NTP=ntp.ubuntu.com/NTP=i.debian.pool.ntp.org/' /etc/systemd/timesyncd.conf && \
systemctl restart systemd-timesyncd && \
update-grub && \
reboot
```

Блок команд для установки недостающего программного обеспечения и настройки системы

Описание команд блока:

- `echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && |`
`echo "net.ipv4.conf.all.forwarding=1" >> /etc/sysctl.conf && |`
`sysctl -p /etc/sysctl.conf && |` - задаем настройки для переадресации сетевых интерфейсов.

- ***systemctl stop cron && |
systemctl disable cron &&*** | - отключаем системную службу cron.
- ***systemctl set-default multi-user.target &&*** | - перевод в серверный режим работы приложений
- ***apt install -y ca-certificates curl wget gnupg gnupg2 systemd-timesyncd htop lm-sensors lsb-release unzip &&*** | - устанавливаем приложения для работы некоторых служб (названия приложения содержатся в данной строке).
- ***mkdir -m 0755 -p /etc/apt/keyrings && |
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg && |
echo "deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null && |
sudo wget -qO- https://repo.mosquitto.org/debian/mosquitto-repo.gpg.key | gpg --dearmor -o /etc/apt/keyrings/mosquitto-repo.gpg && |
sudo wget -O /etc/apt/sources.list.d/mosquitto-bookworm.list https://repo.mosquitto.org/debian/mosquitto-bookworm.list*** - создание репозитория для пакетного менеджера для установки последней версии Docker и клиента Mosquitto.
- ***apt-get update &&*** | - применение добавления репозитория.
- ***apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin mosquitto-clients=2.0.18-0mosquitto1~bookworm1 && |
sed -i 's/#NTP=/NTP=1.debian.pool.ntp.org/' /etc/systemd/timesyncd.conf && |
systemctl restart systemd-timesyncd && |
update-grub && |
reboot*** — установка и настройка ранее прописанных компонентов с последующей перезагрузкой виртуальной машины.

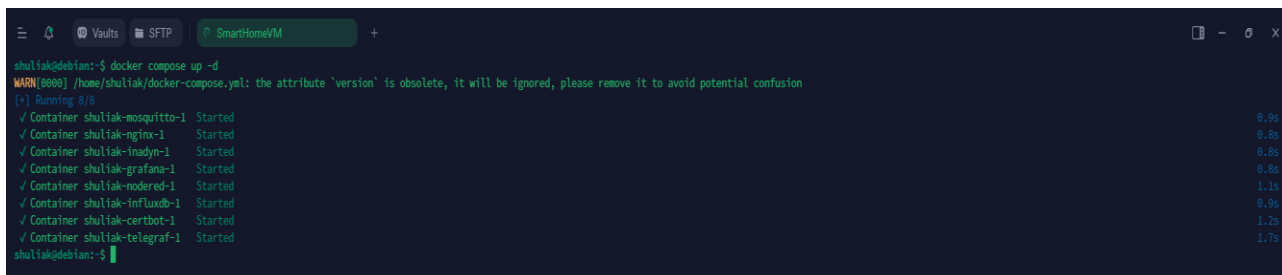
После перезагрузки приступаем к установке программных компонентов системы визуализации.

2.4 Установка программных компонентов системы визуализации

Для установки программных компонентов создадим файл `install.sh` (приложение 3) командой **`nano install.sh`**. Запишем команды скрипта (приложение 2) в созданном файле. Данный скрипт выполняет следующие операции (описание к коду скрипта в (приложение 2)):

- Записывает IP адрес машины, на которой запускается скрипт в переменную, которая далее используется в скрипте.
- Выводит информацию о версии Docker и IP адреса машины.
- Записывает имя текущего пользователя в переменную, для дальнейшего использования в скрипте (настройка прав доступа, создание директорий и т.д.).
- Генерирует уникальные порты пользователя для приложений:
 - `mosquitto_port=1883`
 - `influxdb_port=8086`
 - `grafana_port=3000`
 - `nodered_port=1880`
 - `telegraf_port1=8092`
 - `telegraf_port2=8094`
 - `telegraf_port3=8125`
 - `nginx_port1=80`
 - `nginx_port2=443`
- Задаёт доменное имя для конфигурации сервера nginx и данные для подключения сервиса DDNS No-IP.
- Создает `docker-compose.yml` (приложение 4) для текущего пользователя.
- Создает необходимые каталоги с разрешениями только для текущего пользователя.
- Создает конфигурационные файлы для созданного `docker-compose.yml` типа `prog.conf` с подстановкой переменных доменного имени, текущего пользователя и IP текущей машины.
- Создает файл `settings.sh` (приложение 5) со скриптом для настройки пользовательских параметров приложений.
- Устанавливает владельца и разрешения для созданных файлов конфигурации.
- Проверяет, является ли текущий пользователь уже частью группы Docker, если нет, то добавляет текущего пользователя в группу Docker.

Запускаем файл скрипта командой **`bash install.sh`**. После завершения работы скрипта (установки программных компонентов) поднимаем установленные контейнеры командой **`docker compose up -d`**.



```
shuliak@debian:~$ docker compose up -d
WARN[0000] /home/shuliak/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 0/0
 ✓ Container shuliak-mosquitto-1 Started 0.9s
 ✓ Container shuliak-nginx-1 Started 0.8s
 ✓ Container shuliak-inady-1 Started 0.8s
 ✓ Container shuliak-grafana-1 Started 0.8s
 ✓ Container shuliak-nodered-1 Started 1.1s
 ✓ Container shuliak-influxdb-1 Started 0.9s
 ✓ Container shuliak-certbot-1 Started 1.2s
 ✓ Container shuliak-telegraf-1 Started 1.7s
shuliak@debian:~$
```

Первый запуск контейнеров

Запускаем файл скрипта для настроек пользовательских параметров командой **bash settings.sh**. Далее в терминале будут выводиться сообщения и подсказки для задания логинов и паролей приложений, а также для настройки некоторых конфигураций приложений. Все заданные логины, пароли и токены сохраняем в файл (*приложение 1*).

Также важный этап при выполнении скрипта настроек – получение сертификатов ssl при помощи клиента certbot. При получении необходимо ввести свой действующий email, который указывался при регистрации доменного имени (см. п.п. 2.1), далее даем согласие на создание сертификатов, и создаем записи вида TXT в редакторе зон своего домена **gb-study-shuliak.ru**. Сертификаты будут установлены автоматически.

```
SmartHomeVM
(Y)es/(N)o: y
Account registered.
Requesting a certificate for *.gb-study-shuliak.ru and gb-study-shuliak.ru

-----
Please deploy a DNS TXT record under the name:
_acme-challenge.gb-study-shuliak.ru.
with the following value:
6uV7nHhEH1KH8aLQ1NkFLPPYvIghcWu_b53w6n2FU
Press Enter to Continue

-----
Please deploy a DNS TXT record under the name:
_acme-challenge.gb-study-shuliak.ru.
with the following value:
Q37aK0n7eapz195Ls_Mr2usXeoX8PvJpDnU4YuColw
(This must be set up in addition to the previous challenges; do not remove,
replace, or undo the previous challenge tasks yet. Note that you might be
asked to create multiple distinct TXT records with the same name. This is
permitted by DNS standards.)

Before continuing, verify the TXT record has been deployed. Depending on the DNS
provider, this may take some time, from a few seconds to multiple minutes. You can
check if it has finished deploying with aid of online tools, such as the Google
Admin toolbox: https://toolbox.googleapps.com/apps/dig/#TXT/_acme-challenge.gb-study-shuliak.ru.
Look for one or more bolded line(s) below the line 'ANSWER'. It should show the
value(s) you've just added.

-----
Press Enter to Continue

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/gb-study-shuliak.ru/fullchain.pem
Key is saved at: /etc/letsencrypt/live/gb-study-shuliak.ru/privkey.pem
This certificate expires on 2024-10-29.
These files will be updated when the certificate renews.

NEXT STEPS:
- This certificate will not be renewed automatically. Autorenewal of --manual certificates requires the use of an authentication hook script (--manual-auth-hook) but one was not provided. To renew this certificate, repeat this same certbot
command before the certificate's expiry date.

-----
If you like Certbot, please consider supporting our work by:
- Donating to EFF: https://letsencrypt.org/donate
- Donating to EFF: https://eff.org/donate-le

shuliak@debian:~$
```

Запрос на добавление записей TXT в редакторе зон домена от клиента Certbot

The screenshot shows the 'Редактор Зоны' (Zone Editor) interface on the HostFly.by website. The page title is 'Редактор Зоны' and the subtitle is 'Домены / Управление зоной'. Below the header, there is a search bar and a list of tools. The main content area is titled 'Zone Records for "gb-study-shuliak.ru"'. It features a filter bar with options for 'Все', 'A', 'CNAME', 'MX', 'SRV', and 'TXT'. The 'TXT' filter is selected. Below the filter, there is a table with three columns: 'Имя' (Name), 'TTL', 'Тип' (Type), 'Запись' (Record), and 'Действия' (Actions). The table contains three records, all of type 'TXT' with a TTL of 14400. Each record has a 'Редактировать' (Edit) button and a 'Удалить' (Delete) button. The records are: 1. '_acme-challenge.gb-study-shuliak.ru.' with value '09ZnqYTD032k26Cjx59aP3j5ZXLuj5Hn2oIoti', 2. '_acme-challenge.gb-study-shuliak.ru.' with value '6uV7nHhEH1KH8aLQ1NkFLPPYvIghcWu_b53w6n2FU', and 3. '_acme-challenge.gb-study-shuliak.ru.' with value 'Q37aK0n7eapz195Ls_Mr2usXeoX8PvJpDnU4YuColw'. At the bottom of the page, there is a footer with the cPanel logo and version 120.6.14, and a navigation bar with links to 'Главная', 'Товарные знаки', 'Политика конфиденциальности', 'Документация', and 'Give Feedback'.

Имя	TTL	Тип	Запись	Действия
_acme-challenge.gb-study-shuliak.ru.	14400	TXT	09ZnqYTD032k26Cjx59aP3j5ZXLuj5Hn2oIoti	Редактировать Удалить
_acme-challenge.gb-study-shuliak.ru.	14400	TXT	6uV7nHhEH1KH8aLQ1NkFLPPYvIghcWu_b53w6n2FU	Редактировать Удалить
_acme-challenge.gb-study-shuliak.ru.	14400	TXT	Q37aK0n7eapz195Ls_Mr2usXeoX8PvJpDnU4YuColw	Редактировать Удалить

Записи TXT в редакторе зон домена

После завершения всех настроек, необходимо перезапустить все контейнеры, чтобы все изменения вступили в силу. Останавливаем контейнеры командой ***docker compose down***, далее поднимаем их снова командой ***docker compose up -d***. Проверяем состояние запущенных контейнеров командой ***docker ps***.

```
shuliak@shulian:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8d6e7ee773a4	telegraf:alpine	"/entrypoint.sh tele..."	23 minutes ago	Up 23 minutes	0.0.0.0:8092->8092/tcp, 0.0.0.0:8092->8092/udp, :::8092->8092/tcp, :::8092->8092/udp, 0.0.0.0:8094->8094/tcp, 0.0.0.0:8094->8094/udp	shuliak-telegraf-1
94->8094/udp, e2d8f8bdc298	certbot/certbot:latest	"/bin/sh -c 'trap ev..."	23 minutes ago	Up 23 minutes	80/tcp, 443/tcp	shuliak-certbot-1
73a4cad1f202	influxdb:alpine	"/entrypoint.sh infl..."	23 minutes ago	Up 23 minutes	0.0.0.0:8086->8086/tcp, :::8086->8086/tcp	shuliak-influxdb-1
a777e416ad24	nodored/node-red:latest-minimal	"/.entrypoint.sh"	23 minutes ago	Up 23 minutes (healthy)	0.0.0.0:1880->1880/tcp, :::1880->1880/tcp	shuliak-nodored-1
ae61e66d11b	trglabit/inadyn:latest	"/usr/sbin/inadyn --..."	23 minutes ago	Up 23 minutes		shuliak-inadyn-1
254acbe453e	grafana/grafana	"/run.sh"	23 minutes ago	Up 23 minutes	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	shuliak-grafana-1
bfe1cda44a	eclipse-mosquitto	"/docker-entrypoint..."	23 minutes ago	Up 23 minutes	0.0.0.0:1883->1883/tcp, :::1883->1883/tcp	shuliak-mosquitto-1
f2d43b3e3a75	nginx:mainline-alpine-slim	"/docker-entrypoint..."	23 minutes ago	Up 23 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp	shuliak-nginx-1

```
shuliak@shulian:~$
```

Результат выполнения команды *docker ps*

В результате выполнения видно, что в столбце STATUS все контейнеры помечены как UP (подняты). Это свидетельствует об успешной установке и настройке программных компонентов системы визуализации.

2.5 Настройка роутера. Проброс портов виртуальной машины

В настоящей системе используется роутер HUAWEI HG8245H-256M. Для доступа к роутеру необходимо ввести его IP в адресной строке браузера. IP роутера – 192.168.100.1. В появившемся меню ввести:

- *Account* – **root**.
- *Password* – **admin**.



Стартовое меню роутера HUAWEI HG8245H-256M

После входа в интерфейс настройки заходим в раздел *Forward Rules > Port Mapping Configuration*. Создаем записи для соответствующих приложений системы визуализации.

Mapping Name	WAN Name	Internal Host	External Host	Enable
<input type="checkbox"/> NODE-RED	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> Grafana	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> Telegraf	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> Mosquitto	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> Influx-DB	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> HTTP	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/> HTTPS	4_INTERNET_R_VID_10	192.168.100.87	--	Enable

Type: ☒ User-defined ☐ Application

Application: Select...

Enable Port Mapping: ☒

Mapping Name: NODE-RED

WAN Name: 4_INTERNET_R_VII

Internal Host: 192.168.100.87 DESKTOP-TL4E390


External Source IP Address:

Protocol: TCP Internal port number: 1880 - 1880

External port number: 1880 - 1880 External source port number: 0 - 0

Delete Add Apply Cancel

Конфигурация для доступа к Node-Red


HG8245H-256M
Logout

Status
WAN
LAN
IPv6
WLAN
Security
Forward Rules
Network Application
System Tools

DMZ Configuration
Forward Rules > Port Mapping Configuration

Port Mapping Configuration
Port Trigger Configuration

On this page, you can configure port mapping parameters to set up virtual servers on the LAN network and allow these servers to be accessed from the Internet.
Note: The well-known ports for voice services cannot be in the range of the mapping ports.

	Mapping Name	WAN Name	Internal Host	External Host	Enable
<input type="checkbox"/>	NODE-RED	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Grafana	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Telegraf	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Mosquitto	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Influx-DB	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTP	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTPS	4_INTERNET_R_VID_10	192.168.100.87	--	Enable

Type:
☒ User-defined
☐ Application

Application:
Select...

Enable Port Mapping:
☒

Mapping Name:
Grafana

WAN Name:
4_INTERNET_R_VII

Internal Host:
192.168.100.87
DESKTOP-TL4E390

External Source IP Address:


Protocol:
TCP
Internal port number:
3000
3000

External port number:
3000
3000
External source port number:
0
0

Delete

Add

Apply
Cancel

 Copyright © Huawei Technologies Co., Ltd. 2009-2016. All rights reserved.

Конфигурация для доступа к Grafana


HG8245H-256M
Logout

Status
WAN
LAN
IPv6
WLAN
Security
Forward Rules
Network Application
System Tools

DMZ Configuration
Forward Rules > Port Mapping Configuration

Port Mapping Configuration
Port Trigger Configuration

On this page, you can configure port mapping parameters to set up virtual servers on the LAN network and allow these servers to be accessed from the Internet.
Note: The well-known ports for voice services cannot be in the range of the mapping ports.

	Mapping Name	WAN Name	Internal Host	External Host	Enable
<input type="checkbox"/>	NODE-RED	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Grafana	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Telegraf	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Mosquitto	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Influx-DB	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTP	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTPS	4_INTERNET_R_VID_10	192.168.100.87	--	Enable

Type:
☒ User-defined
☐ Application

Application:
Select...

Enable Port Mapping:
☒

Mapping Name:
Telegraf

WAN Name:
4_INTERNET_R_VII

Internal Host:
192.168.100.87
DESKTOP-TL4E390

External Source IP Address:

Protocol:
TCP
Internal port number:
8092
8092

External port number:
8092
8092
External source port number:
0
0

Delete

Protocol:
TCP
Internal port number:
8094
8094

External port number:
8094
8094
External source port number:
0
0

Delete

Protocol:
TCP
Internal port number:
8125
8125


External port number:
8125
8125
External source port number:
0
0

Delete

Add

Apply
Cancel

Конфигурация для доступа к Telegraf


HG8245H-256M
Logout

Status
WAN
LAN
IPv6
WLAN
Security
Forward Rules
Network Application
System Tools

DMZ Configuration
Port Mapping Configuration
Port Trigger Configuration

On this page, you can configure port mapping parameters to set up virtual servers on the LAN network and allow these servers to be accessed from the Internet.
Note: The well-known ports for voice services cannot be in the range of the mapping ports.

	Mapping Name	WAN Name	Internal Host	External Host	Enable
<input type="checkbox"/>	NODE-RED	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Grafana	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Telegraf	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input checked="" type="checkbox"/>	Mosquitto	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Influx-DB	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTP	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTPS	4_INTERNET_R_VID_10	192.168.100.87	--	Enable

Type:
☒ User-defined
☐ Application

Application:
Select...

Enable Port Mapping:
☒

Mapping Name:
Mosquitto

WAN Name:
4_INTERNET_R_VID_10

Internal Host:
192.168.100.87
DESKTOP-TL4E390

External Source IP Address:

Protocol:
TCP
Internal port number:
1883
1883

External port number:
1883
1883
External source port number:
0
0


Delete

Add

Apply
Cancel

Copyright © Huawei Technologies Co., Ltd. 2009-2016. All rights reserved.

Конфигурация для доступа к Mosquitto


HG8245H-256M
Logout

Status
WAN
LAN
IPv6
WLAN
Security
Forward Rules
Network Application
System Tools

DMZ Configuration
Port Mapping Configuration
Port Trigger Configuration

On this page, you can configure port mapping parameters to set up virtual servers on the LAN network and allow these servers to be accessed from the Internet.
Note: The well-known ports for voice services cannot be in the range of the mapping ports.

	Mapping Name	WAN Name	Internal Host	External Host	Enable
<input type="checkbox"/>	NODE-RED	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Grafana	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Telegraf	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	Mosquitto	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input checked="" type="checkbox"/>	Influx-DB	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTP	4_INTERNET_R_VID_10	192.168.100.87	--	Enable
<input type="checkbox"/>	HTTPS	4_INTERNET_R_VID_10	192.168.100.87	--	Enable

Type:
☒ User-defined
☐ Application

Application:
Select...

Enable Port Mapping:
☒

Mapping Name:
Influx-DB

WAN Name:
4_INTERNET_R_VID_10

Internal Host:
192.168.100.87
DESKTOP-TL4E390

External Source IP Address:

Protocol:
TCP
Internal port number:
8086
8086

External port number:
8086
8086
External source port number:
0
0

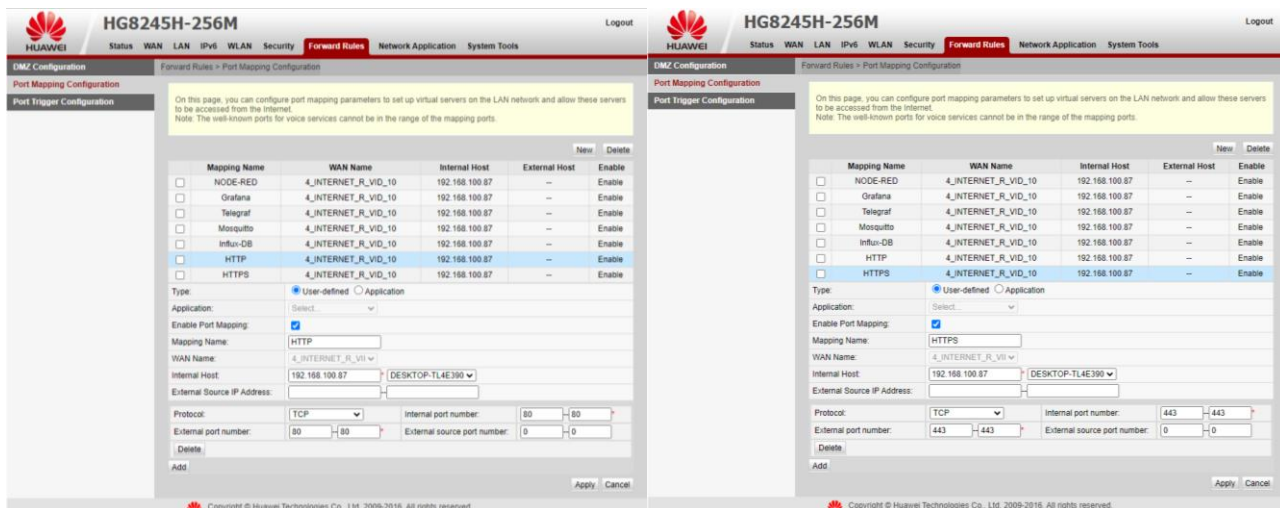
Delete

Add

Apply
Cancel

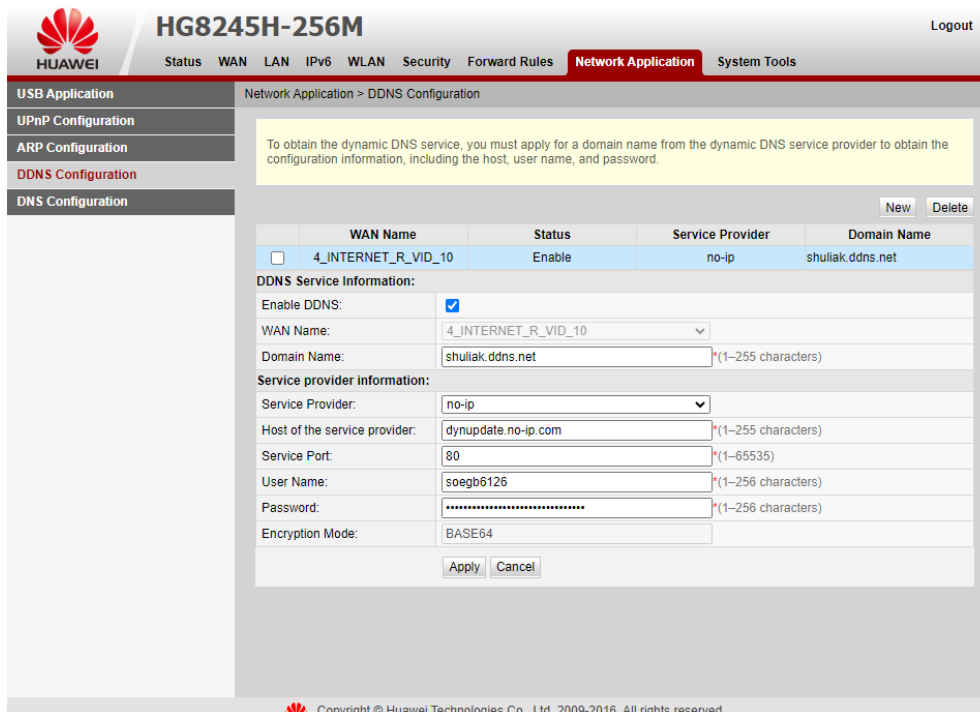
Copyright © Huawei Technologies Co., Ltd. 2009-2016. All rights reserved.

Конфигурация для доступа к Influx-DB



Проброс портов 80 и 443 для надстройки Nginx, которая позволит подключаться к приложениям через зашифрованный протокол HTTPS

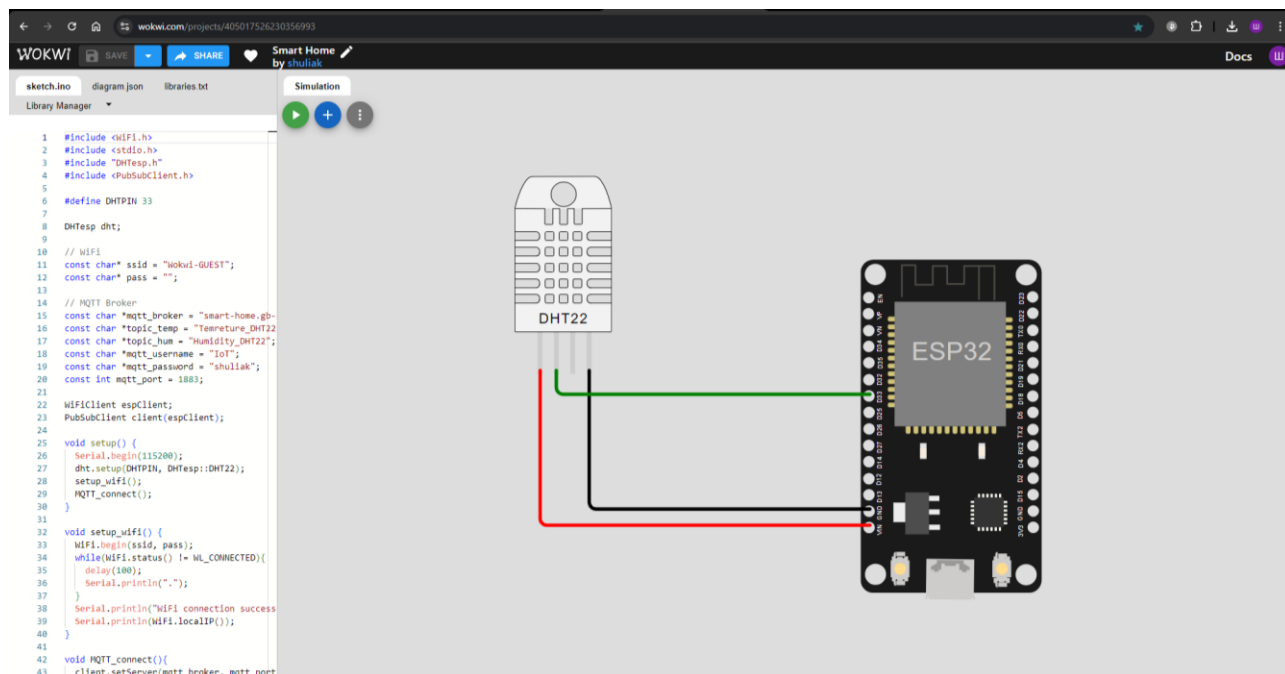
Далее включим функцию DDNS на роутере. Переходим в подраздел *Network Application* > *DDNS Configuration*. Создаем новую запись и вводим все данные и соответствия, которые были внесены при регистрации доменного имени *shuliak.ddns.net* на сайте www.noip.com.



Включение функции DDNS на роутере

2.6 Настройка и подключение оконечного оборудования к системе визуализации

Для имитации работы и подключения оконечного оборудования было принято решение воспользоваться сервисом *Wokwi - Online ESP32 Simulator*. Согласно п.п. 1.6 в качестве оконечного оборудования была выбрана плата *ESP32* с подключенным датчиком температуры и влажности *DHT22 Arduino*.



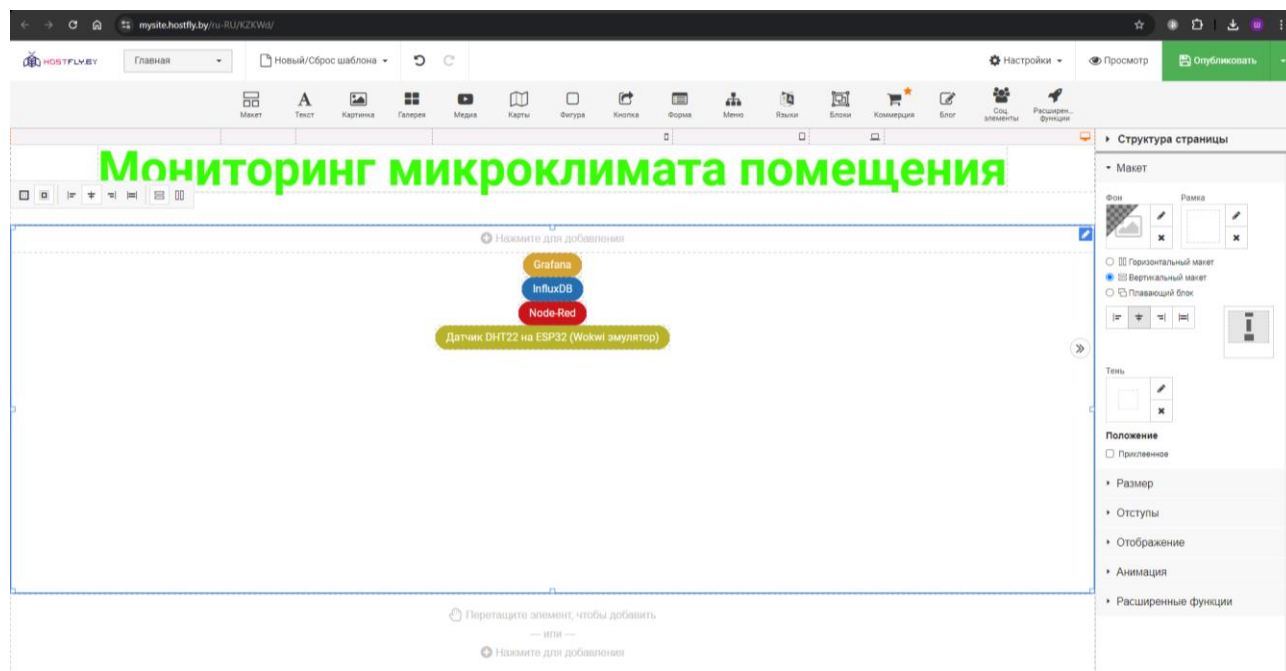
Визуализация подключения датчика DHT22 к плате ESP32

Логика работы написана в файле *sketch.ino* (приложение 6) на языке C++. Для написания использовались следующие библиотеки:

- **<WiFi.h>** - библиотека *WiFi* используется для подключения *Arduino* по протоколу *WiFi* к локальной сети, интернету или другим устройствам, а так же для создания сервера, что бы другие устройства и компьютеры могли подключаться к вашей ардуинке «по воздуху».
- **"DHTesp.h"** – библиотека для работы датчика *DHT* с платой *ESP*.
- **<PubSubClient.h>** - библиотека для работы с *MQTT*.

2.7 Создание страницы на хостинге для быстрого доступа к приложениям

Для создания страницы использовалось приложение хостинга *my.hostfly.by* «Конструктор сайтов».



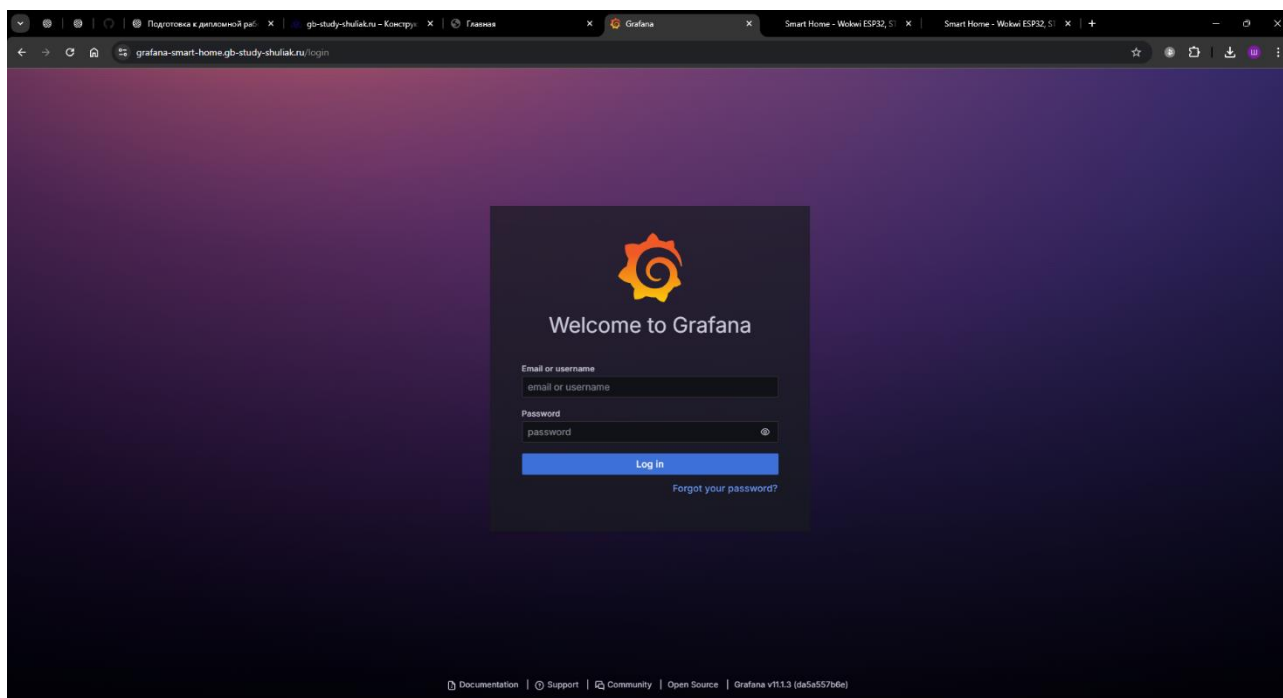
Приложение «Конструктор сайтов», хостинг *my.hostfly.by*

Доступ к главной странице осуществляется по адресу <https://gb-study-shuliak.ru/>.

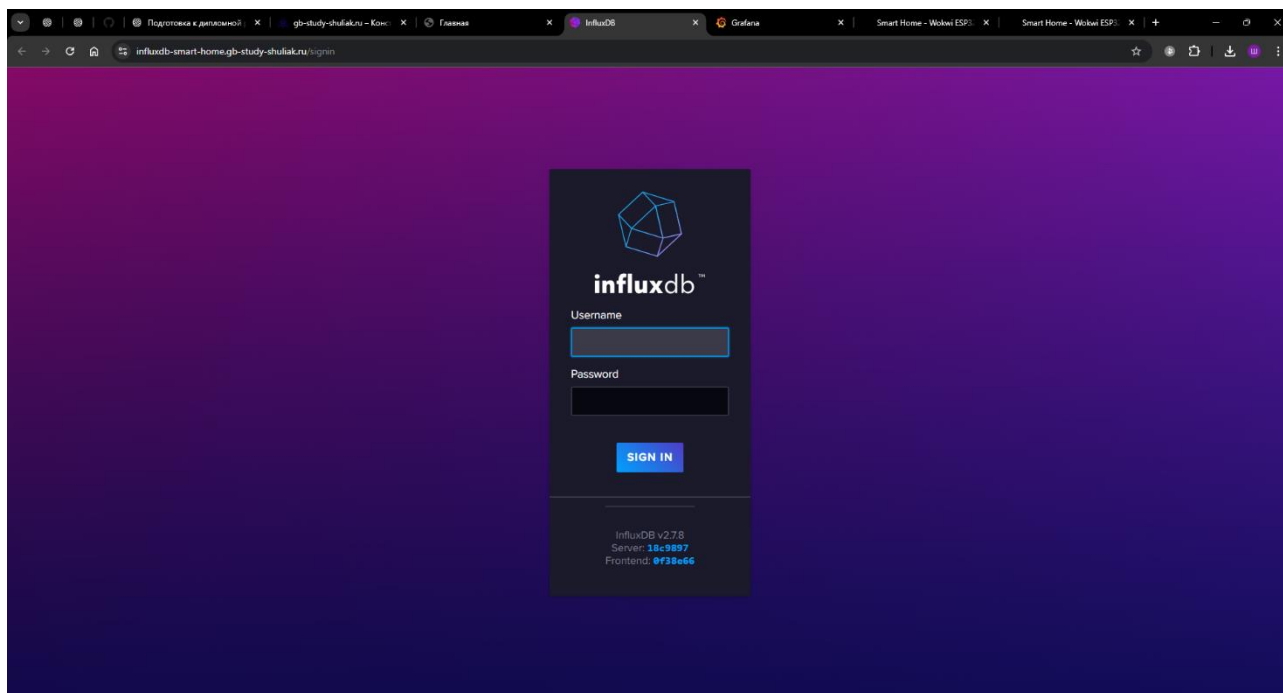


Главная страница сайта <https://gb-study-shuliak.ru/>

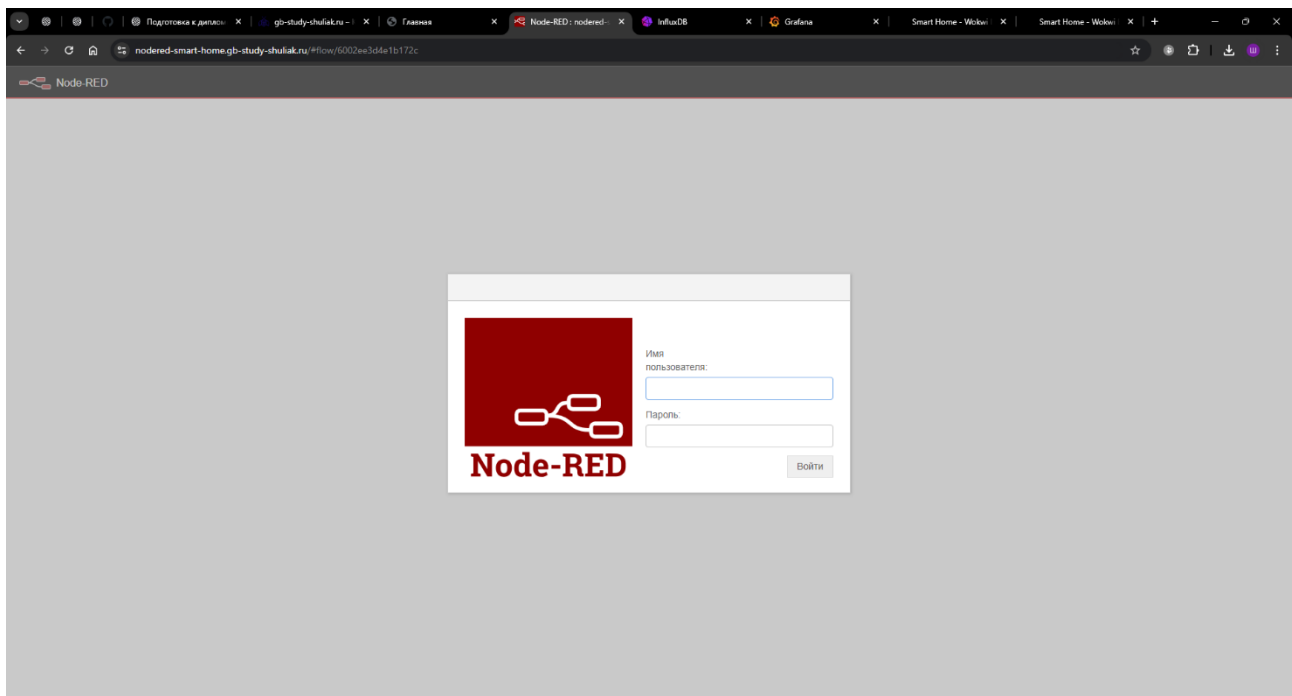
Подключение к приложениям осуществляет при нажатии кнопок с названием приложения, открытие приложений происходит в новой вкладке по протоколам HTTPS.



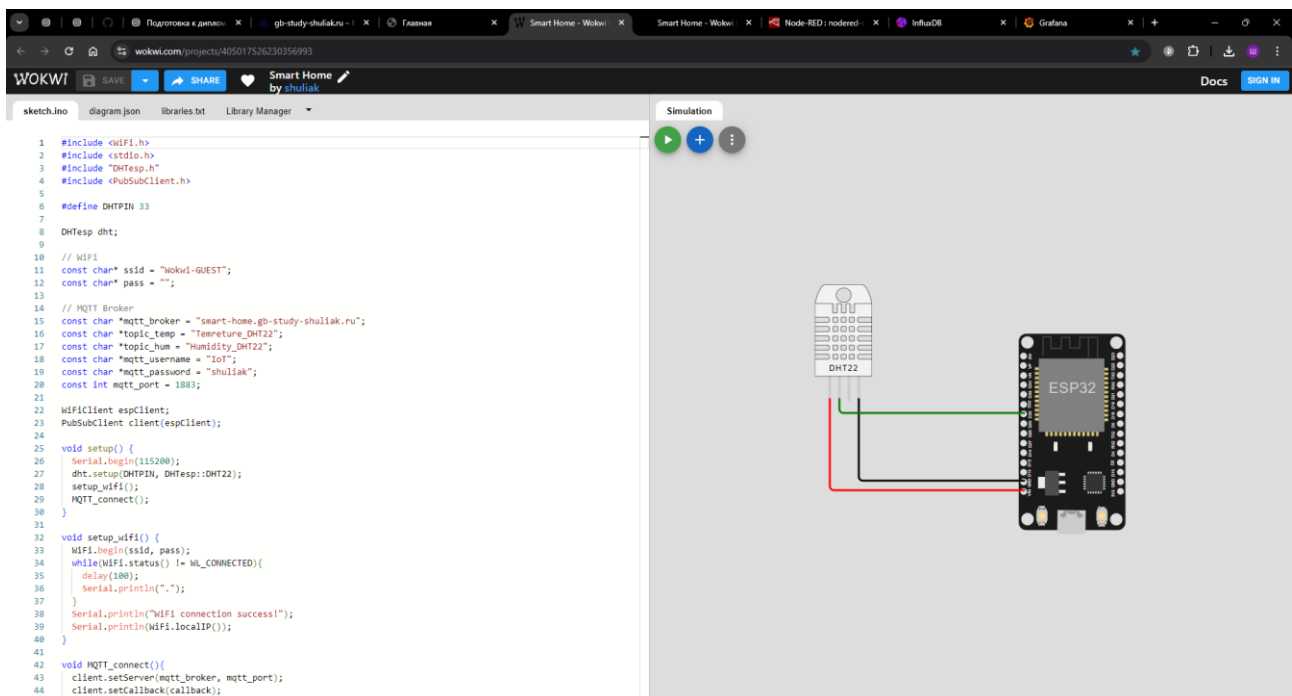
Доступ к Grafana, прямая ссылка: <https://grafana-smart-home.gb-study-shuliak.ru>



Доступ к InfluxDB, прямая ссылка: <https://influxdb-smart-home.gb-study-shuliak.ru>



Доступ к Node-RED, прямая ссылка: <https://nodered-smart-home.gb-study-shuliak.ru>



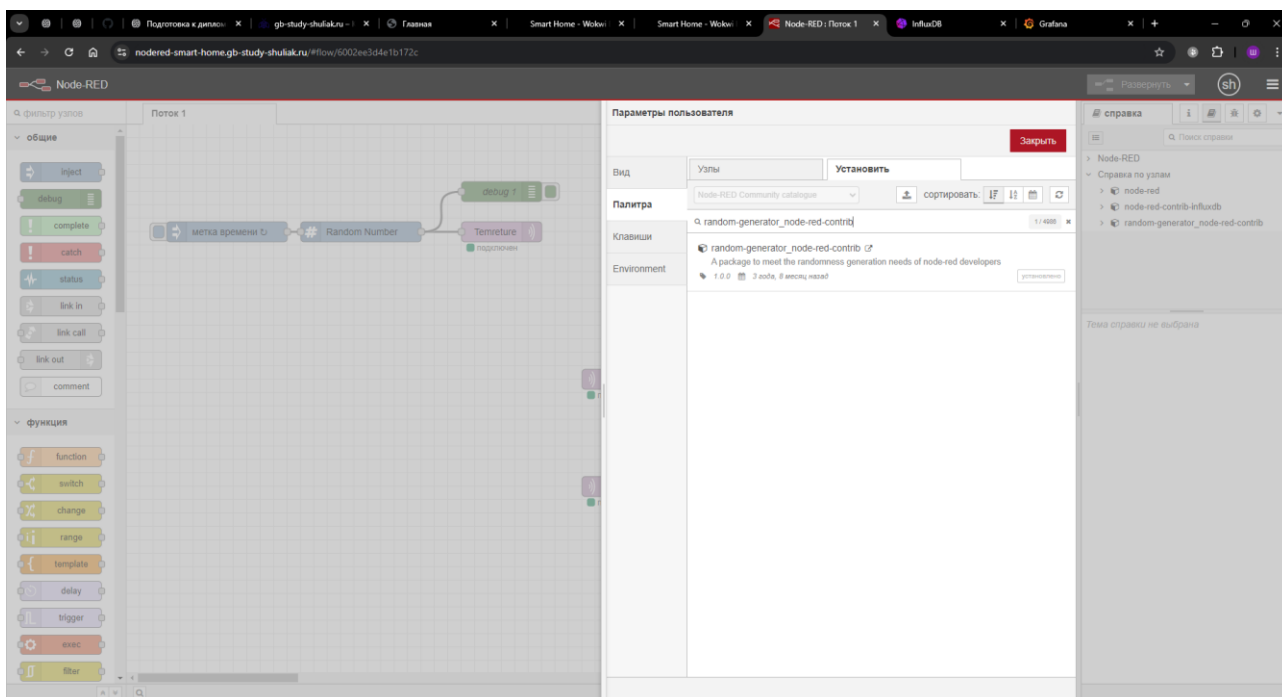
Доступ к проекту оконечного оборудования на Wokwi, прямая ссылка
<https://wokwi.com/projects/405017526230356993>

2.8 Авторизация в приложениях. Настройка приложений. Создание зависимостей в системе визуализации

Данный для авторизации в приложениях прописаны в файле AutorizationData.txt (приложение 1).

1) *Авторизация и настройка Node-RED.* Для возможности проведения тестов устанавливаем дополнительные палитры. Вызываем окно настроек палитры сочетанием клавиш Shift+Alt+p, далее переходим во вкладку «Установить», в поисковой строке ищем и устанавливаем дополнительные палитры:

- *node-red-contrib-influxdb.*
- *random-generator_node-red-contrib.*

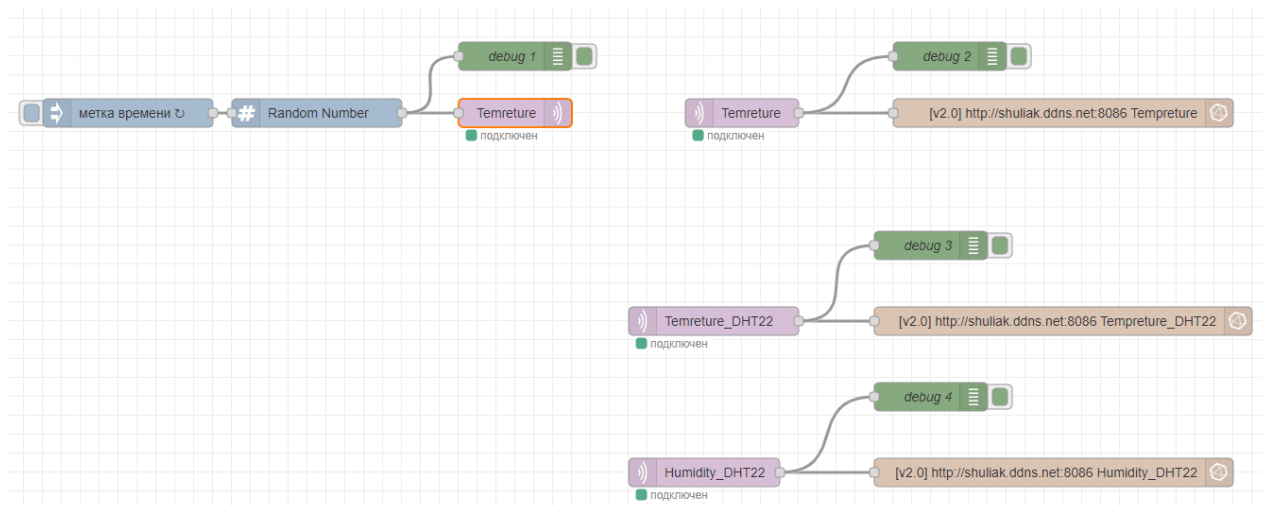


Вкладка настройки палитры в Node-Red

Далее в меню палитр выбираем следующие компоненты:

- блок «inject» в палитре «общие» – 1 блок.
- блок «Number» в палитре «Random Generator» - 1 блок.
- блок «mqtt in» в палитре «сеть» - 1 блок.
- блок «mqtt out» в палитре «сеть» - 3 блока.
- блок «debug» в палитре «общие» - 4 блока.
- блок «influxdb out» в палитре «хранилище» - 1 блок.

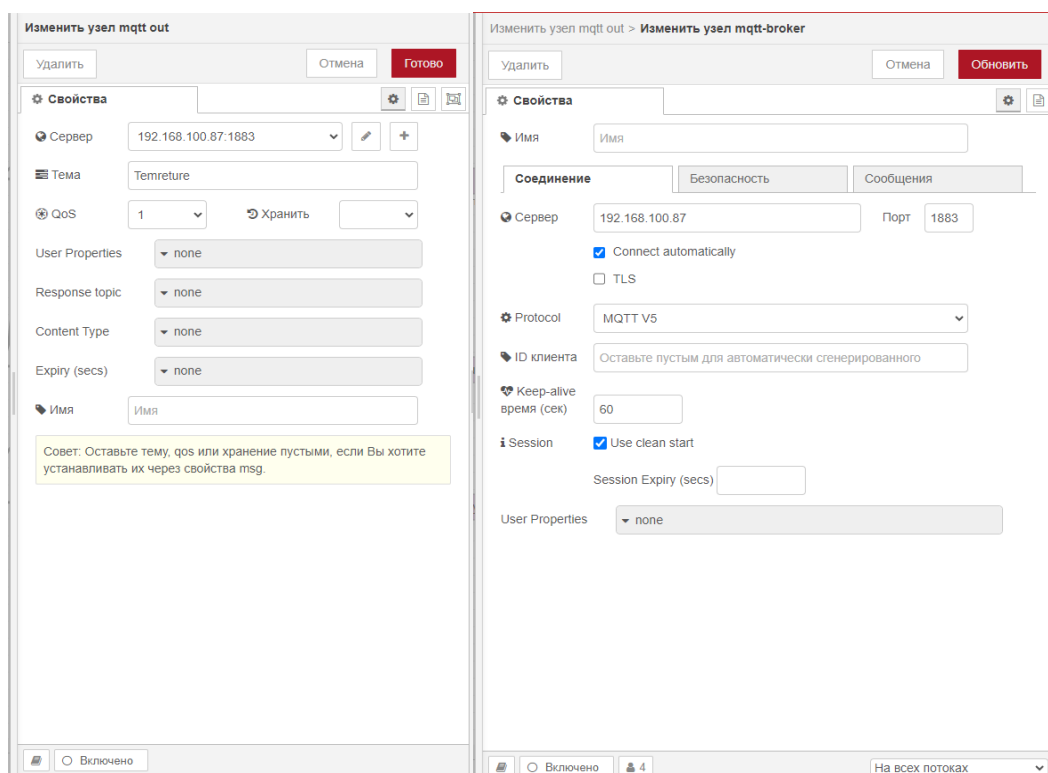
Соединяем компоненты согласно схеме взаимосвязей компонентов.



Отслеживать будем 3 топика:

- Tempreture – топик с генерацией случайных значений, реализованный функционалом рандомайзера Node-RED.
- Tempreture_DHT22 – топик с генерацией показаний температура датчика DHT22, реализованный сервисом Wokwi.
- Humidity_DHT22 – топик с генерацией показаний влажности датчика DHT22, реализованный сервисом Wokwi.

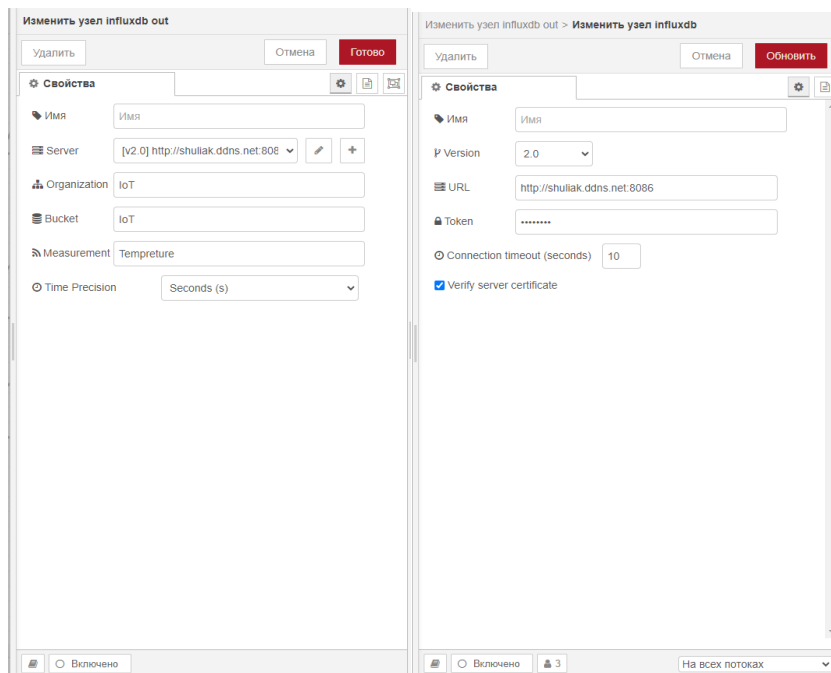
Настройка MQTT осуществляется путем двойного клика по соответствующему блоку.



Настройки узлов MQTT

Настройка всех узлов MQTT выполняется по аналогии с примером, единственным отличием будет названия отслеживаемых топиков.

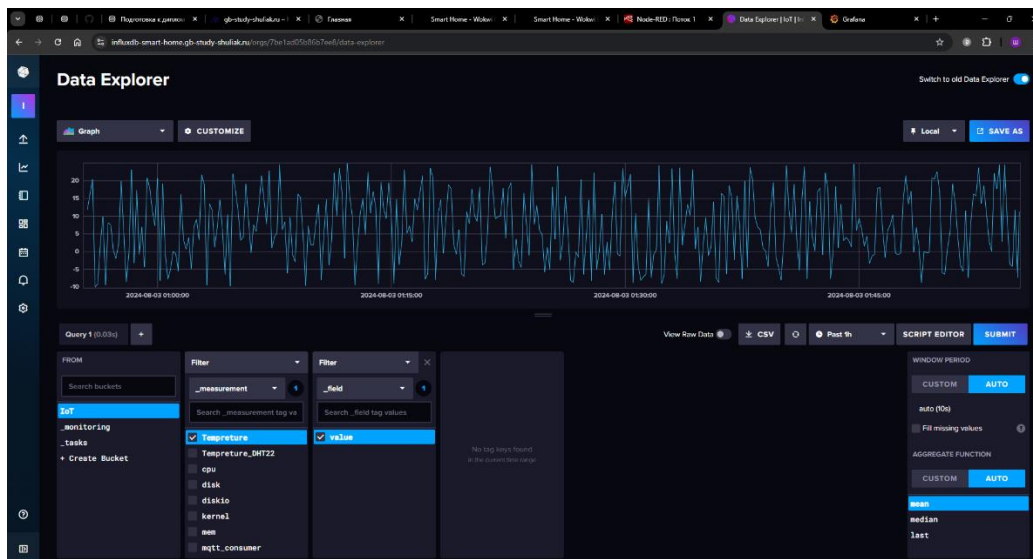
Далее произведем настройку узлов InfluxDB, вызов меню настройки осуществляется путем двойного клика по соответствующему блоку.



Настройки узлов InfluxDB

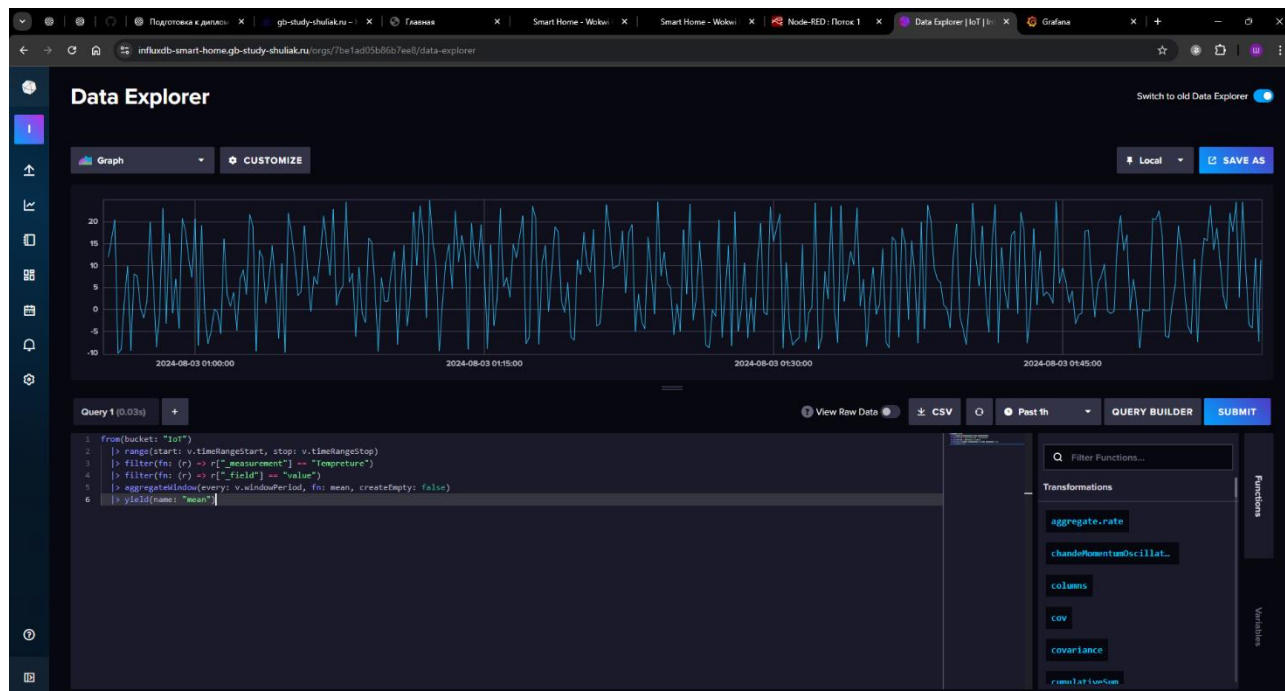
Настройка всех узлов InfluxDB выполняется по аналогии с примером, единственным отличием будет названия отслеживаемых топиков.

2) **Авторизация и настройка InfluxDB.** Настройка будет представлена на примере одного топика «Temperature». Заходим во вкладку «Data Explorer», выбираем соответствующие фильтры, настройки отображения и параметры визуализации, нажимаем кнопку «Submit».



Настройка предварительной визуализации данных в InfluxDB, топик «Temperature»

Так же полезной функцией InfluxDB является автоматическое формирование запроса в виде кода, который в дальнейшем необходим для настройки панелей визуализации в Grafana. Запрос формируется при нажатии кнопки «Script editor».



Формирования запроса на топик «Temperature» в Influx DB

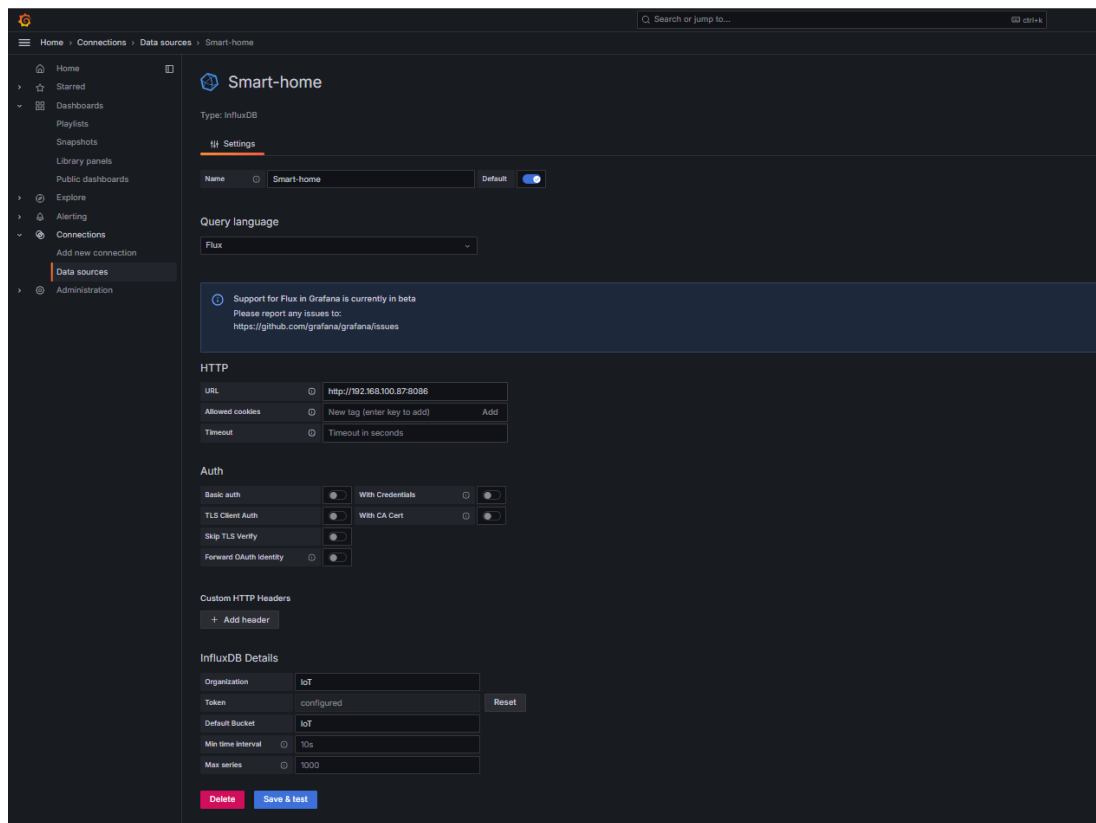
3) **Авторизация и настройка Grafana.** Первая авторизация осуществляется под:

- *Email or username* – **admin**.
- *Password* – **admin**.

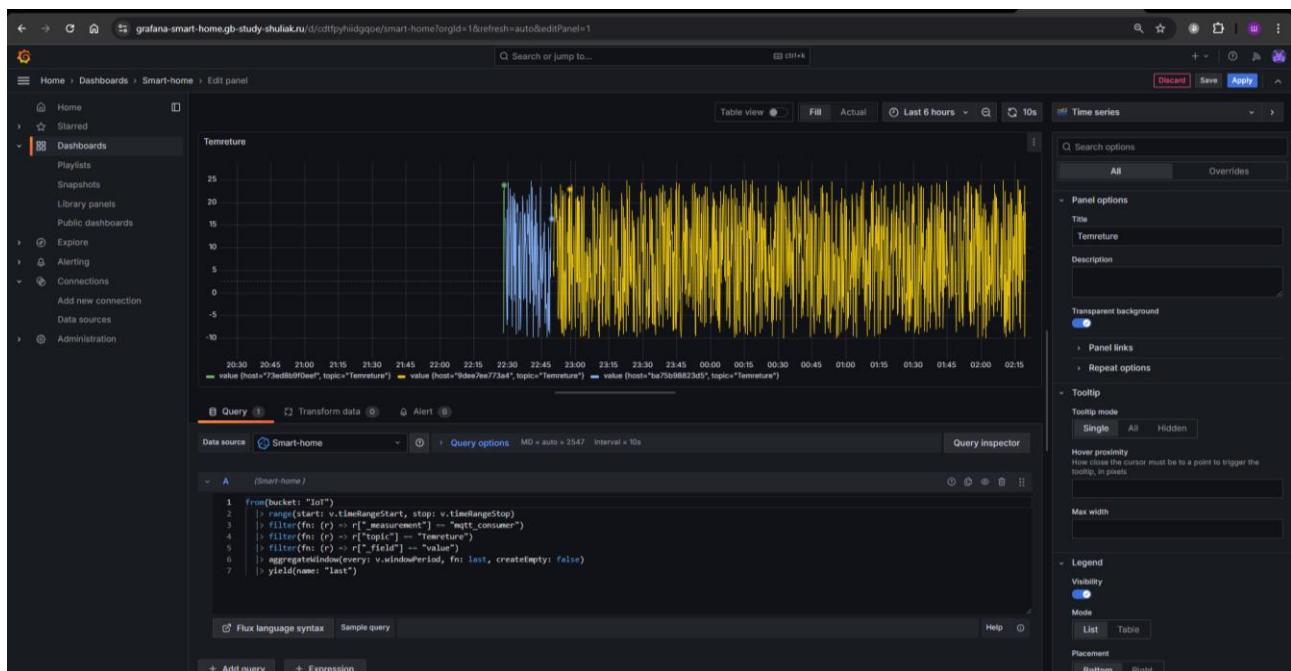
Далее система предложит установить свой пароль, изменяем его, после входа прописываем email адрес и меняем имя пользователя (*приложение 1*).

После авторизации заходим во вкладку *Connections* > *Add new connection* > *InfluxDB*. Далее прописываем все необходимые данные и настройки, согласно рисунку «Меню настройки Grafana».

Настройка визуализации данных производится во вкладке *Dashboards* > *New*. Далее вставляем запрос из InfluxDB (на примере топика «Temperature»). Устанавливаем необходимые нам параметры визуализации и сохраняем изменения.



Меню настройки Grafana



Настройка визуализации данных в Grafana, топик «Tempreture»

2.9 Тест узлов системы визуализации

Тест 1. Проводим в терминале приложения Termius. Создаем два терминала, подключенных к нашей виртуальной машине. Один терминал будет выступать в роли брокера, второй в роли подписчика. Команды для проверки:

БРОКЕР - `mosquitto_pub -h 192.168.100.87 -p 1883 -t "GB" -m "21.3" -u "IoT" -P "shuliak"`

ПОДПИСЧИК - `mosquitto_sub -h 192.168.100.87 -p 1883 -t "GB" -u "IoT" -P "shuliak"`



```
shuliak@debian:~$ mosquitto_pub -h 192.168.100.87 -p 1883 -t "GB" -m "21.3" -u "IoT" -P "shuliak"
shuliak@debian:~$ mosquitto_sub -h 192.168.100.87 -p 1883 -t "GB" -u "IoT" -P "shuliak"
21.3
```

Отправка сообщений

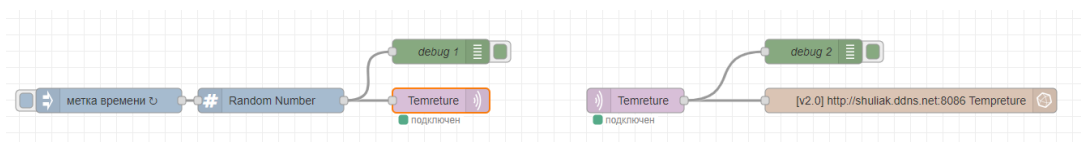


```
Linux debian 6.1.0-23-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Aug 2 22:18:51 2024
shuliak@debian:~$ mosquitto_sub -h 192.168.100.87 -p 1883 -t "GB" -u "IoT" -P "shuliak"
21.3
25.6
23.1
```

Прием сообщений

Тест 2. Сообщения генерируются при помощи блока «random» и тайминга блока «ingest» в Node-RED, топик теста «Temperature».



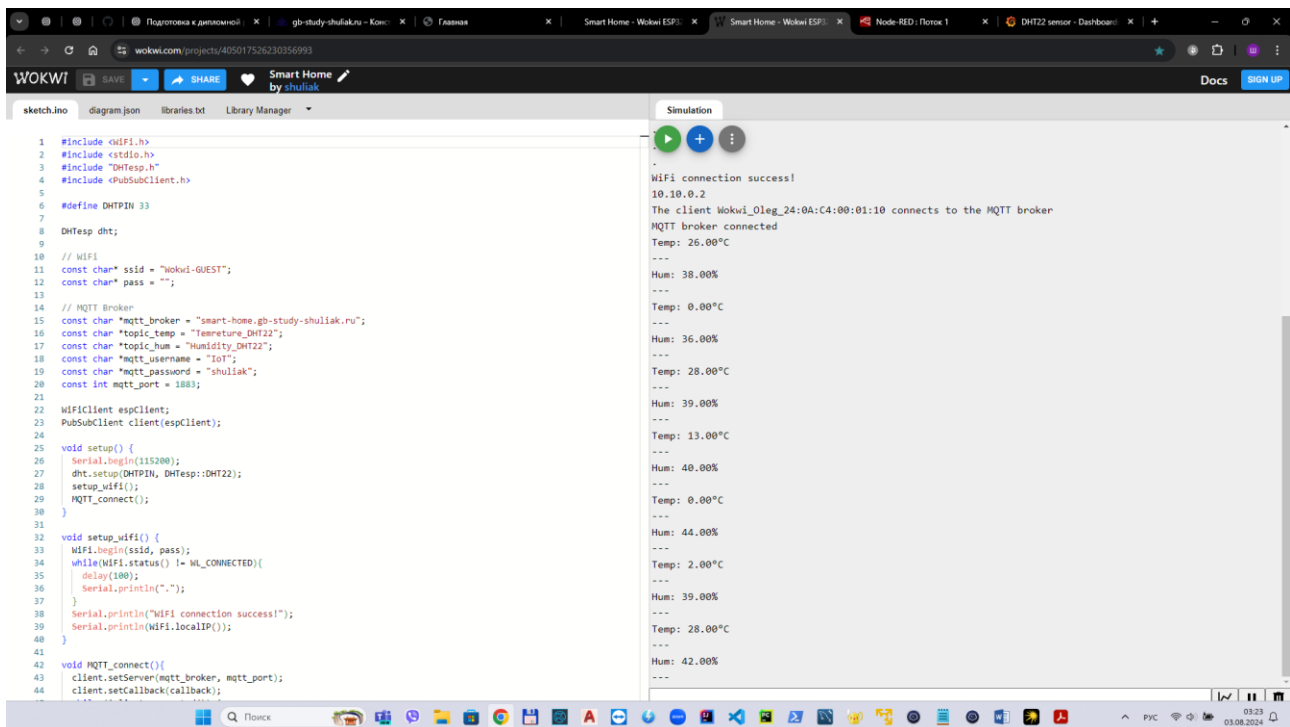
Связка для второго теста, топик «Tempreture»

Было сгенерировано 5 значений:

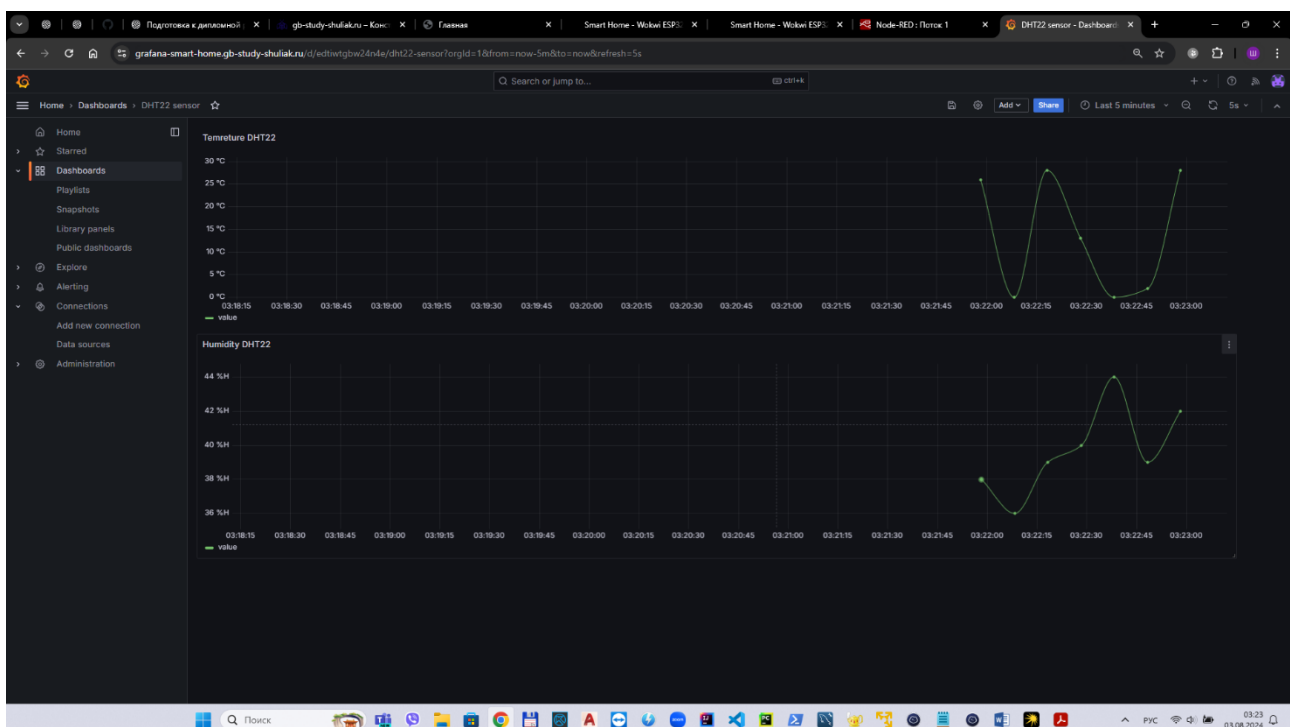
- 03:16:15 – значение 17.5.
- 03:16:25 – значение 3.4.
- 03:16:35 – значение 1.9.
- 03:16:45 – значение -0.2.
- 03:16:55 – значение 1.1.

Согласно таймингу на визуализации значения приняты успешно (см. рисунки ниже).

Эмулированная система DHT22 + ESP32 успешно подключилась к сети Wi-Fi, далее к клиенту MQTT, сгенерировано 7 значений температуры и 7 значений влажности.



Генерация данных эмулятором Wowki



Визуализация данных в Grafana

Все три теста пройдены, система виртуализации работоспособна.

2.10 Заключение по главе 2

В данной главе была описана проделанная работа по развертыванию системы визуализации данных с использованием MQTT-протокола в качестве транспорта с оконечного оборудования, а именно:

- Зарегистрировано доменное имя для системы, подключена функция DDNS для исключения возможности падения системы при изменении IP адреса роутера провайдером. Применены протоколы шифрованной передачи данных HTTPS. Разработан веб-сайт для быстрого доступа к приложениям системы.
- Была развернута виртуальная машина с использованием контейнеризации сервиса Docker для установки программных компонентов. Написаны файлы скриптов для быстрого развертывания системы.
- Настроена система передачи, сбора и визуализации данных, принимаемых от оконечного оборудования.
- Запрограммировано оконечное оборудования и подключено к разработанной системе виртуализации.
- Проведены тесты для подтверждения работоспособности разработанной системы визуализации.

Заключение

В ходе проделанной работы были закреплены полученные навыки по разработке системы визуализации данных с использованием MQTT-протокола в качестве транспорта с окончательного оборудования.

Разработанная система может использоваться как для тестов различных сборок окончательного оборудования, так и для реальной интеграции в различные системы IoT. Однако данная система имеет ряд недостатков и достоинств.

Достоинства:

- Быстрое развертывание системы с помощью имеющегося скрипта.
- Малый объем занимаемого места на жестком диске.
- Использование контейнеризации, что позволяет приложениям работать независимо от общих системных ресурсов.
- Использование зашифрованной передачи данных через протокол HTTPS.

Недостатки:

- Сама система развернута на ПК, что может привести к недоступности системы в сети.
- Используемые бесплатные протоколы SSL необходимо обновлять каждые 3 месяца.
- Использование бесплатного доменного имени для DDNS также следует продлять каждый месяц.

Из вышеперечисленного следует, что данная система находится данная система может быть усовершенствована, и ее разработка будет вестись дальше.

Доступ к актуальной системе по адресу - <https://gb-study-shuliak.ru> .

Список использованных источников

- 1) <https://gb.ru>
- 2) <https://www.vmware.com/info/workstation-player/evaluation>
- 3) <https://cdimage.debian.org/debian-cd/current/amd64/iso-cd>
- 4) <https://termius.com/free-ssh-client-for-windows>
- 5) <https://selectel.ru/blog/what-is-docker>

Приложения

1. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/AutorisationData.txt> - AutorisationData.txt.
2. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/CommandsDebian.sh> - CommandsDebian.sh
3. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/install.sh> - Install.sh
4. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/docker-compose.yml> - docker-compose.yml
5. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/settings.sh> - settings.sh
6. <https://github.com/OlegShuliak/GraduateWork.Shuliak.Group6126/blob/main/sketch.ino> - sketch.ino