СЕТЕВОЙ ПРОТОКОЛ MQTT

СОДЕРЖАНИЕ

СЕТЕВОЙ ПРОТОКОЛ MQTT

ВВЕДЕНИЕ

MQTT / Message Queuing Telemetry Transport

Упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу «издатель — подписчик» (publisher — subscribber).

Обмен данными между устройствами — централизованный - выполняется через специальный сервер (программное обеспечение, MQTT-брокер).

Протокол ориентируется на простоту в использовании, невысокую нагрузку на каналы связи, работу в условиях постоянной потери связи, лёгкую встраиваемость в любую систему. Основное предназначение — работа с телеметрией от различных датчиков и устройств.

Использование шаблона подписчика обеспечивает возможность устройствам выходить на связь и публиковать сообщения, которые не были заранее известны или предопределены, в частности, протокол не вводит ограничений на формат передаваемых данных.

Перечисленные особенности позволяют применять протокол в следующих сегментах:
- интернет вещей (Internet of Things, IoT)
- промышленный интернет вещей (Industrial Internet of Things, IIoT)
- машинно-машинное взаимодействие (M2M)

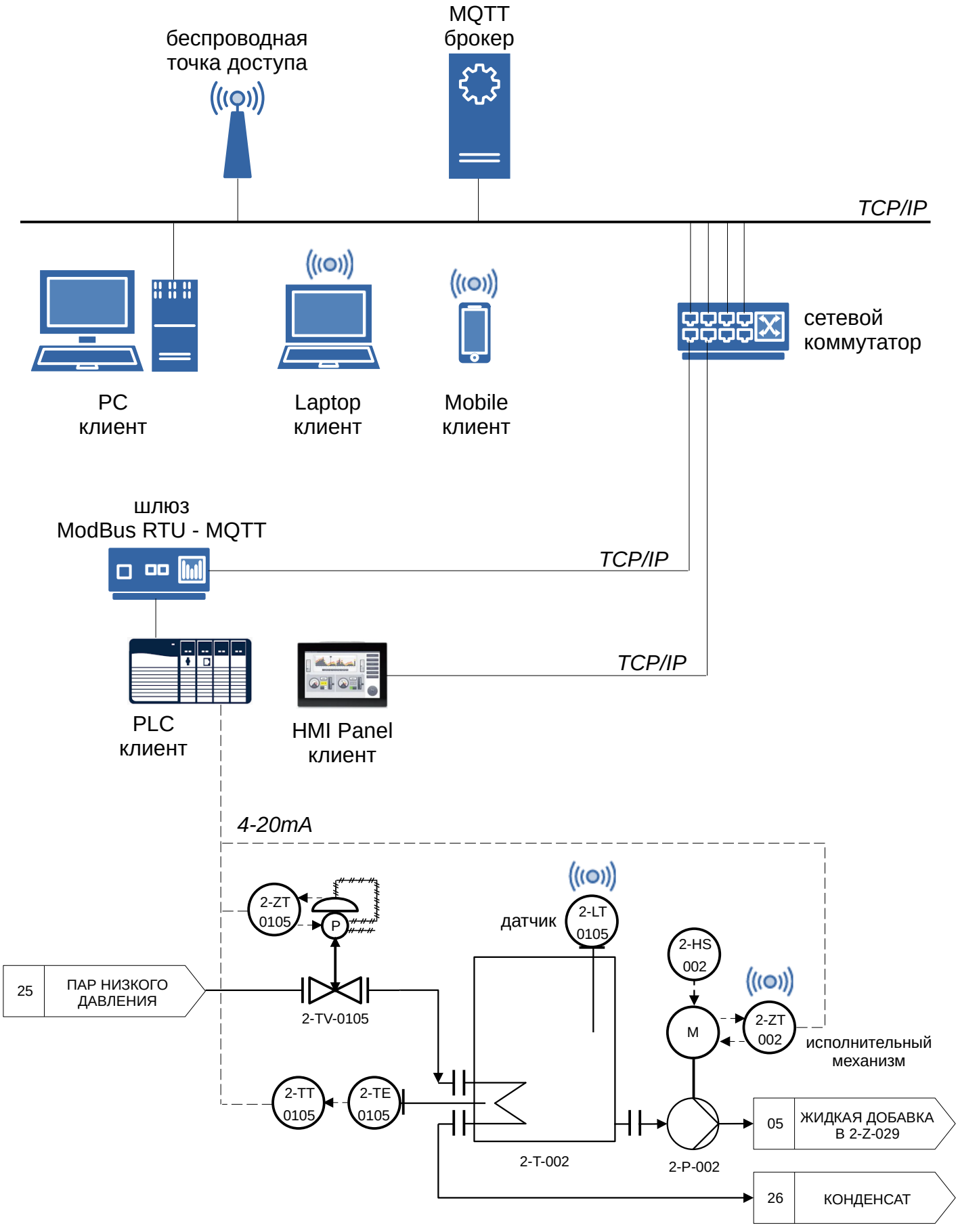
Для встраиваемых беспроводных устройств, которые не поддерживают TCP/IP-сети (например, ZigBee и Bluetooth) специально разработана отдельная версия протокола MQTT-SN (MQTT for Sensor Networks).

Энди Стэнфорд-Кларк (IBM) и Арлен Ниппер (в то время работавший в Eurotech, Inc.) создали первую версию протокола в 1999 году. MQTT в то время использовался для мониторинга нефтепроводов в системе промышленного управления SCADA.

Несмотря на то, что MQTT изначально был закрытым, в 2010 году версия 3.1 вышла по лицензии royalty-free. В 2014 году MQTT стал официально утверждённым стандартом OASIS. Сейчас последней версией протокола является версия 5.0.

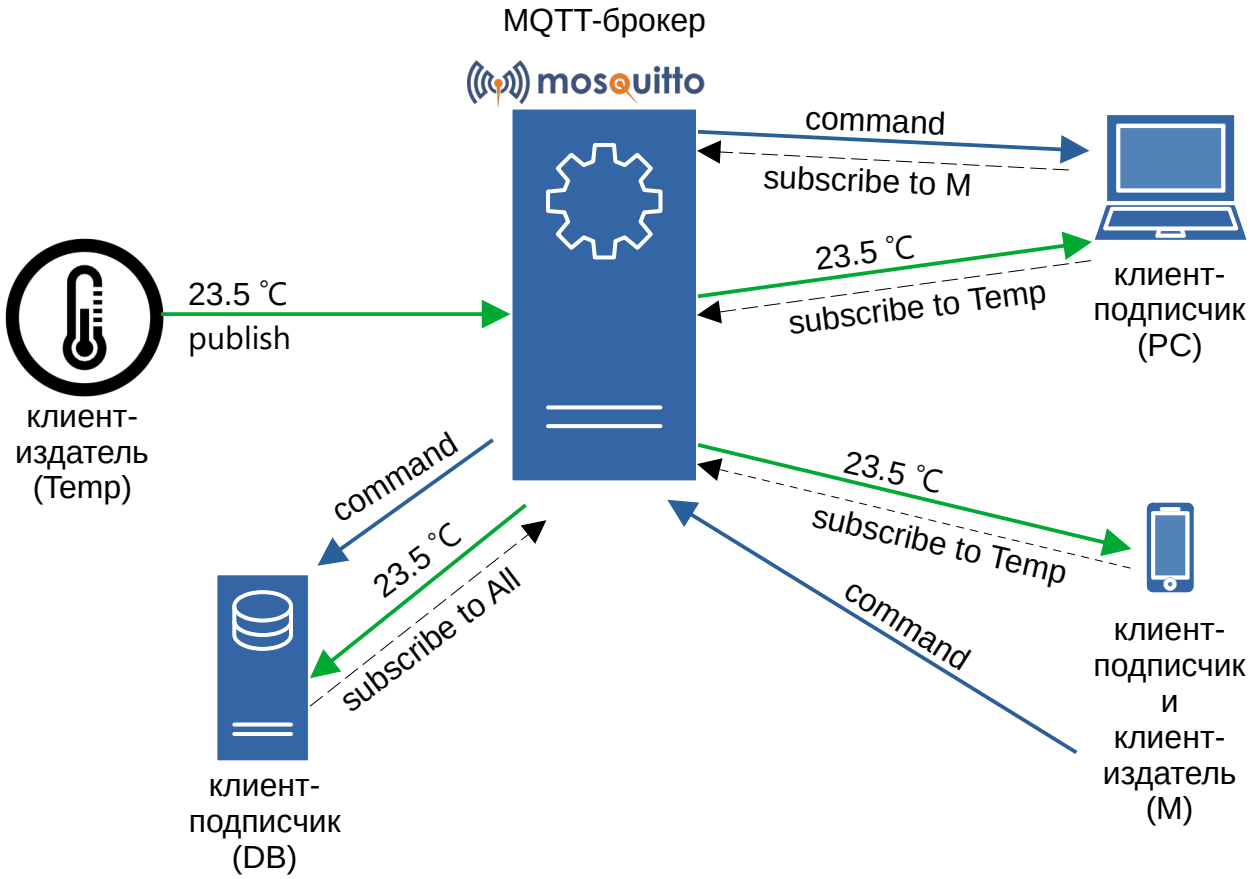
СЕТЕВОЙ ПРОТОКОЛ MQTT

ТОПОЛОГИЯ СЕТИ



СЕТЕВОЙ ПРОТОКОЛ MQTT

МОДЕЛЬ ОБЩЕНИЯ



Участники в системе связи MQTT

- сервер-брокер (один или несколько)
- клиет (один или несколько)
  - издатель, подписчик

Брокер (сервер)

- программное обеспечение, поддерживающее протокол MQTT
- работает на любой аппаратной платформе
  - ПК, ноутбук, сервер, одноплатный компьютер, микроконтроллер
- функции
  - получение данных от издателя
  - обработка данных
  - сохранение данных (если установлен флаг «retain»)
  - передача данных подписчикам
  - контроль доставки

Получаемые (публикуемые) данные имеют уникальные идентификаторы (ID / topic / slot):

- Temp/data
- M/command

MQTT-брокеры: Eclipse Mosquitto, Yandex-cloud, VK-cloud

СЕТЕВОЙ ПРОТОКОЛ MQTT

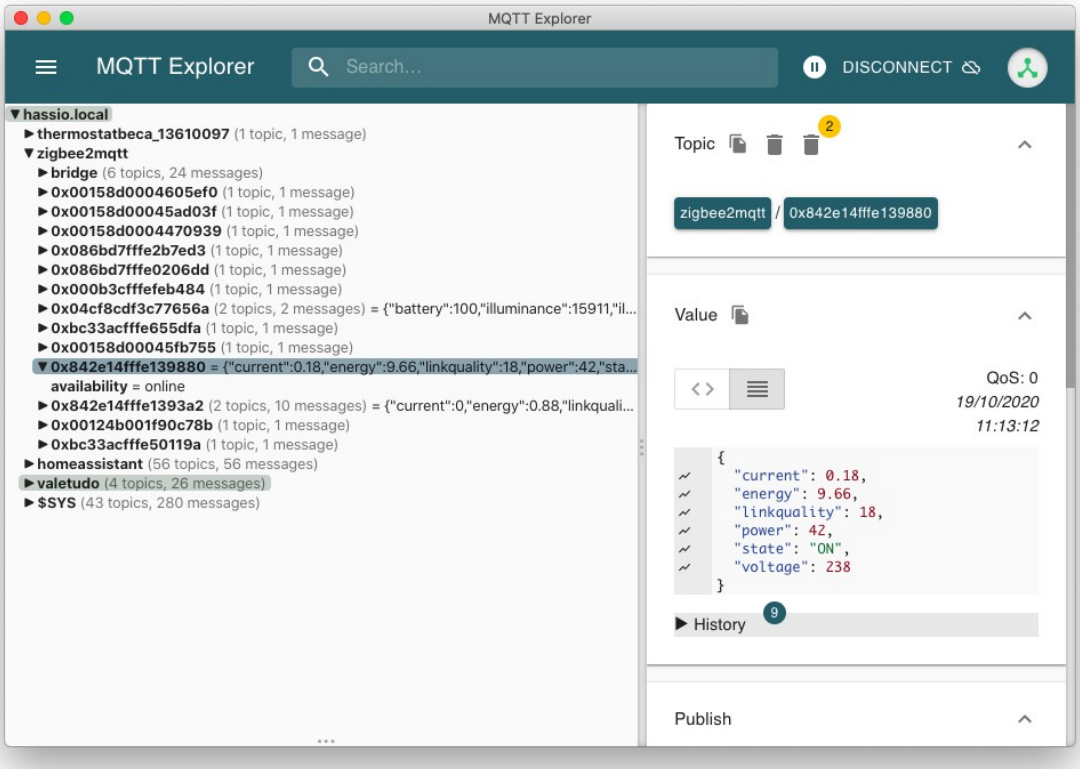
МОДЕЛЬ ОБЩЕНИЯ

Сообщения для взаимодействия с брокером
- Connect — подключение
- Disconnect — отключение
- Publish — публикация информации (по идентификатору)
- Subscribe — подписка (по идентификатору)
- Unsubscribe — отписка (по идентификатору)

Клиент-издатель (отправитель данных)
- программное обеспечение, поддерживающее протокол MQTT
- работает на любой аппаратной платформе
  - ПК, ноутбук, сервер, одноплатный компьютер, микроконтроллер
  - вторичный элемент датчика
- функции
  - публикация / отправка данных брокеру (с указанием идентификатора данных)

Клиент-подписчик (получатель данных)
- программное обеспечение, поддерживающее протокол MQTT
- работает на любой аппаратной платформе
  - ПК, ноутбук, сервер, одноплатный компьютер, микроконтроллер
  - вторичный элемент исполнительного механизма
- функции
  - подписка на данные брокера (с указанием идентификатора данных)
  - получение данных от брокера по подписке

Для отладки и эмуляции можно использовать программу **MQTT-Explorer**, которая может выступать в роли клиента (отправитель/получатель): Windows, Linux, macOS версии.



СЕТЕВОЙ ПРОТОКОЛ MQTT

ОСОБЕННОСТИ

Компактный, легковесный бинарный протокол

При передаче данных в условиях медленной и нестабильной связи нужно экономить каждый бит. Поэтому, данные пересылаются в максимально компактном виде.

Асинхронный

Позволяет обслуживать большое количество устройств и не зависит от сетевых задержек.

Механизмы защиты

Протокол не имеет встроенных механизмов защиты данных при передаче, но для обхода этого ограничения успешно применяется SSL (и прочие).

Степень важности сообщения

Степень фажности сообщения определяется сервисным флагом QoS (Quality of Service — качество обслуживания), устанавливаемом в теле сообщения:

= 0

- доставка сообщения осуществляется не более одного раза
- при неудачном исходе сообщение теряется

= 1

- доставка сообщения осуществляется не менее одного раза
- отправка повторяется до тех пор, пока не будет получено подтверждение от адресата

= 2

- доставка сообщения осуществляется только один раз
- если на линии возникают проблемы, доставка задерживается
- адресат в любом случае получит сообщение, когда связь восстановится

Работа за NAT

Клиенты могут находиться за NAT.

Только брокер должен иметь реальный IP-адрес.

Позволяет обойтись без VPN и «пробрасывания» портов.

Флаг RETAIN

Брокер всего лишь посредник между разными клиентами. Клиент отправил сообщение брокеру, брокер переслал это сообщение всем клиентам, которым интересно это сообщение и тут же забыл про это сообщение. Новые клиенты которые подключаются к брокеру уже не получат это сообщение.

При отправке сообщения клиент может сказать что это сообщение нужно сохранять, установив в теле сообщения сервисный флаг Retain. В этом случае брокер получит сообщение, переправит его всем подписанным клиентам, но не забывает про это сообщение, а оставляет его у себя и будет отправлять это же сообщением всем новым клиентам которые подключились к MQTT серверу.

СЕТЕВОЙ ПРОТОКОЛ MQTT

ОСОБЕННОСТИ

Опция LAST WILL (LWT, Last Will and Testament)

Стандартная работа, когда клиент подключается к брокеру и все время остается подключенным к нему (брокер "видит" момент подключения и может понять когда клиент отключился).

Допустим, есть клиент, который должен быть постоянно подключен к брокеру. Если этот клиент, вдруг, не подключен — это авария (что-то случилось либо с клиентом, либо с каналом связи). Как узнать об этом происшествии (например, какому-то другому клиенту — сервисному-клиенту, клиенту-дублеру)?

Решение следующее: клиент при подключении сообщает брокеру: "слушай, когда я отключусь, ты, пожалуйста, вот в этот «топик» положи вот такое сообщение". Брокер принимает такое "завещание" от клиента и когда этот клиент отключается — выполняет его "последнюю волю".

СЕТЕВОЙ ПРОТОКОЛ MQTT

ТИПЫ СООБЩЕНИЙ

Сеанс разделен на следующие этапы:

1) ПОДКЛЮЧЕНИЕ (CONNECT/CONNACK)
2) АУТЕНТИФИКАЦИЯ (AUTH)
3) ПЕРЕДАЧА ДАННЫХ
4) ОТКЛЮЧЕНИЕ (DISCONNECT)

Клиент начинает с создания соединения с Брокером (т.е. Клиент — инициатор). При этом, Брокер может продолжить старый сеанс, если он не был до этого закрыт Клиентом.

Аутентификация — это когда Клиент передает Брокеру свое имя и пароль. Имя и пароль, по-умолчанию, передаются в виде открытого текста. Поэтому, для защиты данных необходимо использовать зашифрованный канал связи (например, SSL, TLS).

Брокер может допускать подключение анонимных пользователей — т. е. без их аутентификации (пример открытых и доступных в сети Интернет). При этом, сообщение аутентификации не отменяется — имя пользователя и пароль передаются пустыми.

СЕТЕВОЙ ПРОТОКОЛ MQTT

ФОРМАТ СООБЩЕНИЯ

Поле	Размер	Описание
FIXED HEADER	**2 Байта**	Управляющий заголовок фиксированного размера (для всех пакетов)
. type	4 бита	- тип сообщения (код)
. flag	4 бита	- управляющие флаги
. len	1 байт	- размер блока данных: VAR HEADER + DATA (байт)
VARIABLE HEADER	**N Байт**	Расширенный заголовок переменного размера (не для всех пакетов)
DATA	**M Байт**	Данные переменного размера (не для всех пакетов)

Типы сообщения

Код	Описание
0	Резерв
1	CONNECT - запрос подключения *от Клиента к Серверу*
2	CONNACK - подключение подтверждено *от Сервера к Клиенту*
3	PUBLISH - публикация сообщения *от Клиента к Серверу ИЛИ от Сервера к Клиенту*
4	PUBACK - публикация сообщения подтверждена (QoS 1) *от Сервера к Клиенту ИЛИ от Клиента к Серверу*
5	PUBREC - публикация получена (QoS 2 delivery part 1) *от Сервера к Клиенту ИЛИ от Клиента к Серверу*
6	PUBREL - публикация сброшена (QoS 2 delivery part 2) *от Сервера к Клиенту ИЛИ от Клиента к Серверу*
7	PUBCOMP - публикация завершена (QoS 2 delivery part 3) *от Сервера к Клиенту ИЛИ от Клиента к Серверу*
8	SUBSCRIBE - запрос подписки *от Клиента к Серверу*
9	SUBACK - подписка подтверждена *от Сервера к Клиенту*
10	UNSUBSCRIBE - запрос отписки *от Клиента к Серверу*
11	UNSUBACK - отписка подтверждена *от Сервера к Клиенту*

СЕТЕВОЙ ПРОТОКОЛ MQTT

ФОРМАТ СООБЩЕНИЯ

Типы сообщений

Код	Описание
12	PINGREQ - запрос пинга *от Клиента к Серверу*
13	PINGRESP - ответ пинга *от Сервера к Клиенту*
14	DISCONNECT - сообщение об отключении *от Клиента к Серверу ИЛИ от Сервера к Клиенту*
15	AUTH - аутентификационные данные *от Клиента к Серверу ИЛИ от Сервера к Клиенту*

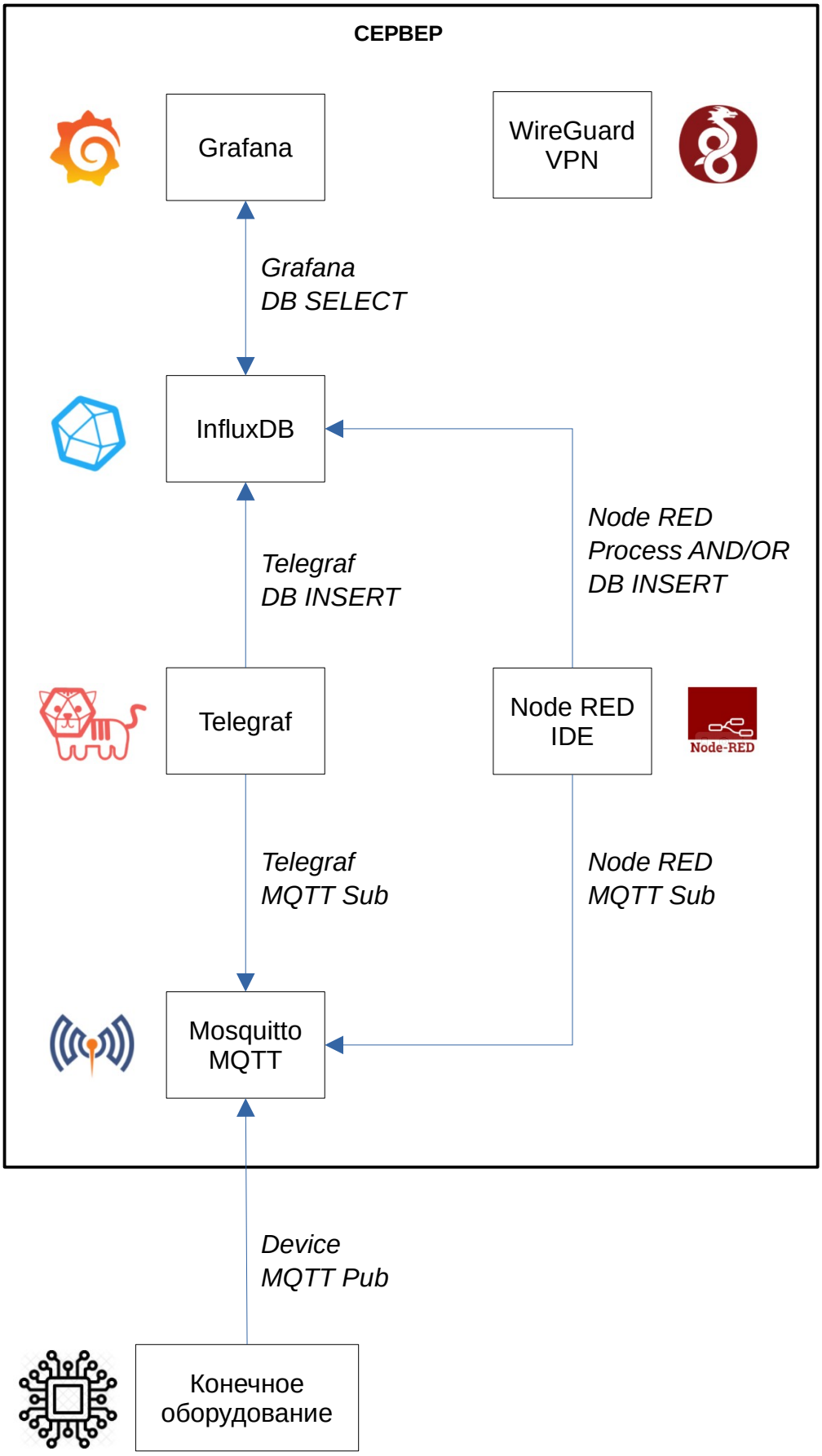
Управляющие флаги

Тип сообщения	биты			
	3	2	1	0
CONNECT	0	0	0	0
CONNACK	0	0	0	0
PUBLISH	D	Q	Q	R
PUBACK	0	0	0	0
PUBREC	0	0	0	0
PUBREL	0	0	1	0
PUBCOMP	0	0	0	0
SUBSCRIBE	0	0	1	0
SUBACK	0	0	0	0
UNSUBSCRIBE	0	0	1	0
UNSUBACK	0	0	0	0
PINGREQ	0	0	0	0
PINGRESP	0	0	0	0
DISCONNECT	0	0	0	0
AUTH	0	0	0	0

PUBLISH.D — DUP = дублирующая отправка PUBLISH-пакета (1 бит)
.Q — QoS = настройки QoS (Quality of Service, 2 бита)
.R — RETAIN = хранимая публикация (1 бит)

СЕТЕВОЙ ПРОТОКОЛ MQTT

ПРИМЕР СИСТЕМЫ СБОРА, ХРАНЕНИЯ И ПРЕДСТАВЛЕНИЯ



СЕТЕВОЙ ПРОТОКОЛ MQTT

ПРИМЕР СИСТЕМЫ СБОРА, ХРАНЕНИЯ И ПРЕДСТАВЛЕНИЯ

Конечное оборудование
- собирает телеметрию (например, опрос датчиков)
- имеет доступ в сеть (в том числе может быть доступ к Интернет)

Mosquitto MQTT
- MQTT брокер
- принимает данные от устройств (конечное оборудование)
- передает данные клиентам

Telegraf
- MQTT клиент
- принимает данные от брокера
- выполняет обработку полученных данных
- записывает полученные и/или обработанные данные в базу данных

InfluxDB
- система управления базами данных (СУБД)
- для хранения временных рядов (данные с метками времени)

Grafana
- система представления (визуализации) данных
- реализовано как web-приложение в стиле «приборных панелей» (диаграммы, графики, таблицы, предупреждения)
- может подключаться к различным источникам данных (в том числе к СУБД)

Node RED
- среда программирования (язык потоковых диаграмм)
- реализовано как web-приложение
- поддерживает множетсво интерфейсов и протоколов

WireGuard
- коммуникационный протокол
- реализует зашифрованные виртуальные частные сети (VPN)

Описание процедуры настройки сервера:
https://habr.com/ru/articles/680902/