Лекция 3. Логическая конструкция IoT-систем

Цель лекции – изучение логической конструкции IoT-систем и её компонентов.

Введение

В данной лекции мы рассмотрим различия между физической и логической конструкциями IoT-систем. Физическая конструкция относится к узловым устройствам и протоколам, обеспечивающим функционирование IoT-экосистемы, в то время как логическая конструкция фокусируется на организации компонентов системы для выполнения определенных функций.

Различия между физической и логической конструкциями IoT-систем

Физический дизайн системы Интернета вещей относится к отдельным узловым устройствам и их протоколам, которые используются для создания функциональной экосистемы Интернета вещей. Каждое узловое устройство может выполнять такие задачи, как дистанционное зондирование, управление, мониторинг и т.д., опираясь на физически подключенные устройства. Оно также может передавать информацию с помощью различных типов беспроводных или проводных соединений. Физический дизайн фокусируется на конкретных решениях, объясняя, как они собираются или конфигурируются.

Логический дизайн для системы IoT — это фактический дизайн того, как ее компоненты (компьютеры, датчики и исполнительные механизмы) должны быть организованы для выполнения определенной функции. Он не вдается в подробности описания того, как каждый компонент будет построен с использованием низкоуровневых особенностей программирования. Физическое проектирование фокусируется на удовлетворении факторов проектирования, включая риски, требования, ограничения и допущения.

Компоненты логической конструкции IoT

Логическая конструкция IoT включает в себя: функциональные блоки IoT, коммуникационные модели IoT, API-интерфейсы связи IoT.

Функциональные блоки IoT

Системы IoT включают в себя несколько функциональных блоков, таких как устройства, коммуникация (связь), безопасность, услуги и приложения.

Функциональные блоки обеспечивают возможности обнаружения, идентификации, приведения в действие, управления и связи. Эти функциональные блоки состоят из устройств, которые обеспечивают связь между сервером и хостом, обеспечивают функции контроля и управления мониторингом, управляют передачей данных, защищают систему IoT с помощью аутентификации и различных функций, а также предоставляют интерфейс для управления и мониторинга различных условий.

Функциональные блоки:

- **Устройство: IoT-с**истема состоит из устройств, которые выполняют функции обнаружения, приведения в действие, мониторинга и управления.
- **Коммуникация:** обеспечивает связь для IoT-системы.
- **Услуги:** услуги мониторинга устройств, услуги управления устройствами, услуги публикации данных и услуги обнаружения устройств.
- **Управление:** этот блок предоставляет различные функции для управления системой IoT.

- Безопасность: этот блок защищает систему IoT, предоставляя такие функции, как аутентификация, авторизация, целостность сообщений и контента, а также безопасность данных.

- Приложение: Это интерфейс, который пользователи могут использовать для управления и мониторинга различных аспектов системы IoT. Приложение также позволяет пользователям просматривать состояние системы и просматривать или анализировать обработанные данные.

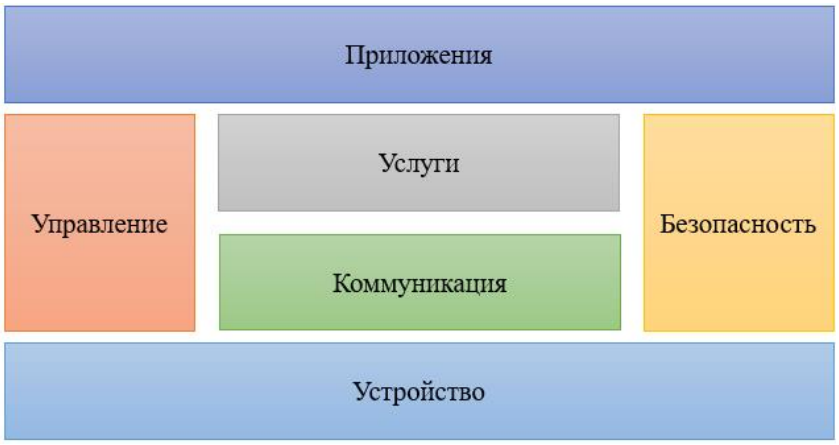

Рисунок 3.1. Функциональные блоки IoT

Коммуникационные модели IoT

В IoT-системе доступно несколько видов моделей, которые используются для связи между системой и сервером:

- **Модель «запрос-ответ» (request-response)**

Модель «запрос-ответ» – это модель связи, в которой клиент отправляет запросы серверу, а сервер отвечает на запросы. Когда сервер получает запрос, он решает, как ответить, извлекает данные, извлекает представление ресурса, подготавливает ответ, а затем отправляет ответ клиенту. Модель «запрос-ответ» — это модель связи без сохранения состояния, и каждая пара «запрос-ответ» независима от других.

HTTP работает как протокол запроса-ответа между клиентом и сервером. Клиентом может быть веб-браузер, а сервером — приложение на компьютере, на котором размещен веб-сайт. Например: клиент (браузер) отправляет HTTP-запрос на сервер; затем сервер возвращает ответ клиенту. Ответ содержит информацию о статусе запроса и может также содержать запрошенный контент.

Рисунок 3.2. Коммуникационная модель «запрос-ответ»

- Модель «издатель-подписчик» (publisher-subscriber). Эта модель состоит из трех субъектов: издателей, брокеров и потребителей. Источником данных являются издатели. Они отправляют данные в тему, которой управляет брокер. Они не знают о потребителях. Потребители подписываются на темы, которыми управляет брокер. Обязанность брокеров — принимать данные от издателей и отправлять их соответствующим потребителям. Брокер располагает только информацией о потребителе, к которому относится конкретная тема, о которой издатель не знает.

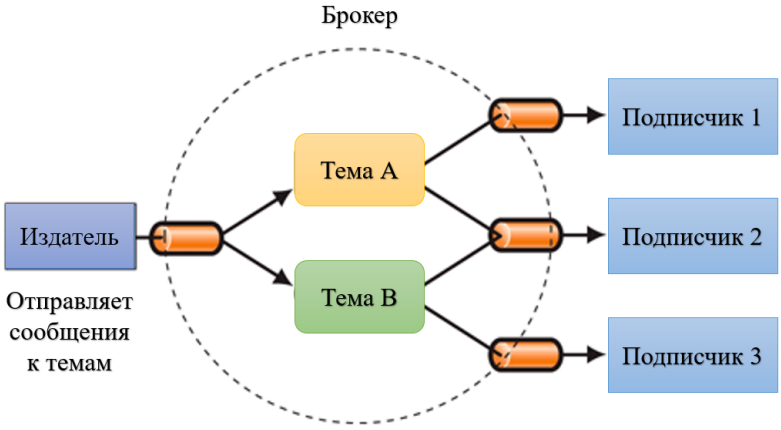

Рисунок 3.3. Коммуникационная модель «издатель-подписчик»

- Модель «тяни-толкай» (push-pull) – Модель «тяни-толкай» включает в себя издателей данных, потребителей данных и очереди данных.

Издатели и потребители не знают друг о друге. Издатели публикуют сообщение/данные и помещают их в очередь. Потребители, присутствующие на другой стороне, извлекают данные из очереди. Таким образом, очередь действует как буфер для сообщения, когда возникает разница в скорости передачи или извлечения данных на стороне издателя и потребителя. Очереди помогают разделить обмен сообщениями между производителем и потребителем. Очереди также действуют как буфер, который помогает в ситуациях, когда есть несоответствие между скоростью, с которой производители передают данные, а потребители извлекают данные.

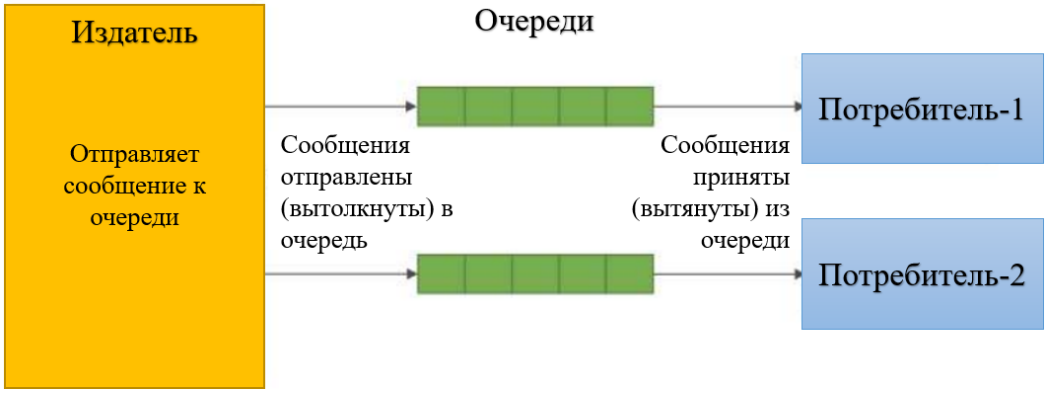

Рисунок 3.4. Коммуникационная модель «тяни-толкай»

- Эксклюзивная пара (exclusive pair) – это двунаправленная модель, включающая полнодуплексную связь между клиентом и сервером. Соединение постоянно и остается открытым до тех пор, пока клиент не отправит запрос на закрытие соединения. Сервер хранит записи всех открытых соединений. Это модель соединения с полным состоянием, и

сервер знает обо всех открытых соединениях. API связи на основе WebSocket полностью основан на этой модели.

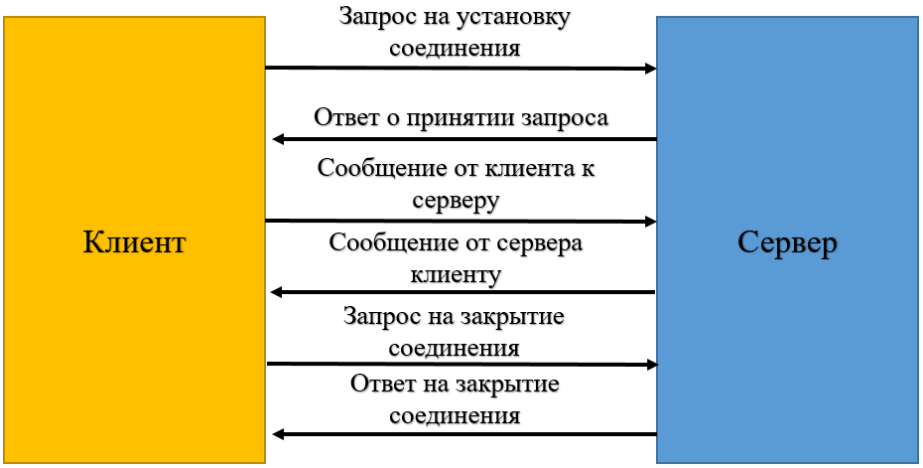

Рисунок 3.5. Коммуникационная модель «**Эксклюзивная пара»**

API-интерфейсы связи IoT

В IoT есть 2 API-интерфейса связи:

- API-интерфейс связи на основе REST;
- API-интерфейс связи на основе Web Socket.

Веб-сервис может быть реализован либо с использованием принципов REST, либо с использованием протокола Web Socket

API-интерфейс связи на основе REST

REpresentational State Transfer (REST) – это набор архитектурных принципов, с помощью которых можно проектировать веб-сервисы и веб-API, которые фокусируются на ресурсах системы и на том, как состояния ресурсов адресуются и передаются. REST API следуют модели связи запрос-ответ. Архитектурные ограничения REST применяются к компонентам, коннекторам и элементам данных в распределенной гипермедиа-системе.

API-интерфейс связи на основе Web Socket

API Web Socket обеспечивают двунаправленную полнодуплексную связь между клиентами и серверами. Она следует модели эксклюзивной парной связи. Этот API связи не требует настройки нового соединения для каждого сообщения, отправляемого между клиентами и серверами. После настройки соединения сообщения могут отправляться и приниматься непрерывно без каких-либо прерываний. API WebSocket подходят для приложений IoT с низкой задержкой или высокими требованиями к пропускной способности.

Таблица 3.1. Разница между REST API и Web Socket API

№	REST API	Web Socket API
1	Это протокол без сохранения состояния. Он не будет хранить данные.	Это протокол с отслеживанием состояния. В нем будут храниться данные.
2	Он однонаправленный. Общаться будет только либо сервер, либо клиент.	Он является двунаправленным. Сообщения могут приниматься или отправляться как сервером, так и клиентом.

3	Модель «запрос-ответ».	Это полнодуплексная модель.
4	HTTP-запрос содержит заголовки, такие как раздел заголовка и раздел названия.	Подходит для приложений реального времени. Не имеет никаких накладных расходов.
5	Для каждого HTTP-запроса будет установлено новое TCP-соединение.	Только одно TCP-соединение.
6	Горизонтальное и вертикальное масштабирование	Только вертикальное масштабирование
7	Получение данных зависит от методов HTTP	Получение данных зависит от IP-адреса и номера порта
8	Это медленнее, чем Web Socket, в отношении передачи сообщений	Web Socket передает сообщения очень быстро чем REST API
9	Ему не нужна память или буферы для хранения данных	Для хранения данных требуется память или буферы

Контрольные вопросы:

1. Что такое IoT и какие его ключевые компоненты?
2. В чем разница между физической и логической конструкциями IoT-систем?
3. Каковы основные функциональные блоки IoT?
4. Что такое модель «запрос-ответ» и как она функционирует в IoT?
5. Опишите модель «издатель-подписчик». Как она отличается от других моделей?
6. Что такое API-интерфейс связи на основе REST и как он работает?
7. Объясните, что такое API Web Socket и в каких случаях он предпочтителен?
8. Каковы ключевые различия между REST API и Web Socket API?