Методические указания по выполнению практических работ № 5-8

«Аппаратные основы, языки разметки данных, протокол MQTT»

по дисциплине «Технологические основы Интернета вещей»

Оглавление

Практическая работа №5 – Измерительные и исполнительные устройства в Интернете вещей

Физические устройства Интернета вещей

Физические устройства Интернета вещей чаще всего состоят из нескольких типов элементов:

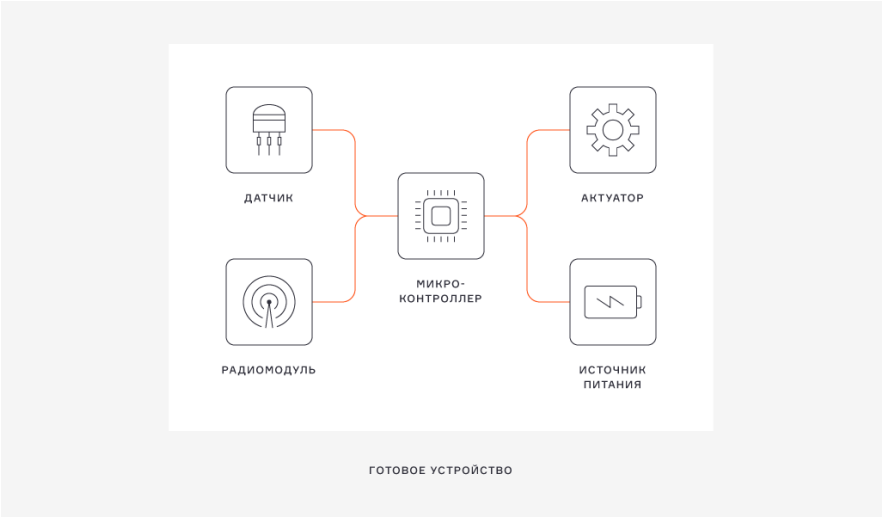


Рисунок 1 – Типовое физическое устройство Интернета вещей

Датчик – чувствительный элемент, который контактирует с окружающей средой: измеряет показания, реагирует на объекты и создает сигнал об этом.

Микроконтроллер (или гейт) – специальная микросхема, предназначенная для управления различными электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи и реализовать логику работы физического устройства.

Актуатор – исполнительное устройство, например, реле или транзистор, способные что-то переключать – по сигналу от микроконтроллера или по дистанционной команде, которую примет радиомодуль. Например, если температура выйдет за допустимый предел, актуатор может включить вентиляцию или кондиционер.

Данный тип элементов Интернета вещей предназначен для воздействия на окружающую среду или определенный объект в ней.

К актуаторам относят:

- Сервоприводы
- Динамики
- Электронные замки
- Осветительные приборы
- Реле и транзисторы
- ИК-пульты

Источник питания. Для работы микроконтроллера и подключенных к нему устройств необходимо питание. В зависимости от энергопотребления и задачи мы можем питать устройство от батарейки или от сети по проводу.

Радиомодуль – устройство для организации различных технологий беспроводной связи (например, GPRS, LTE, NB-IoT, GSM, Wi-Fi, Bluetooth и др.)

Датчики и их устройство

Датчики – устройства, измеряющие физические характеристики объектов или окружающей среды (например, температуру, давление, наличие примесей в воздухе, положение в пространстве и т. д) и преобразующие ее в вид, удобный для дальнейшей обработки.

Датчики, выполненные на основе электронной техники, называются электронными датчиками. Отдельный датчик может измерять (контролировать) одну или одновременно несколько физических величин.

В состав датчика входят чувствительные и преобразовательные элементы.

Измерительный преобразователь – средство измерений, в котором измеряемый сигнал преобразуется в сигнал другой формы, удобной для дальнейшей передачи, преобразования, обработки и хранения.

Первичный измерительный преобразователь – измерительный преобразователь, который взаимодействует непосредственно с исследуемым объектом.

Чувствительный элемент – часть преобразовательного элемента средства измерений, первый элемент в измерительной цепи, находящийся под непосредственным воздействием измеряемой величины. В преобразовательном элементе средства измерений происходит одно из ряда последовательных преобразований величины.

Датчики являются элементом технических систем, предназначенных для измерения, сигнализации, регулирования, управления устройствами или процессами. Датчики преобразуют контролируемую величину (давление, температура, расход, концентрация, частота, скорость, перемещение, напряжение, электрический ток и т. п.) в сигнал (электрический, оптический, пневматический), удобный для измерения, передачи, преобразования, хранения и регистрации информации о состоянии объекта измерений.

Исторически и логически датчики связаны с техникой измерений и измерительными приборами, например, термометры, расходомеры, барометры, прибор «авиагоризонт» и т. д. Обобщающий термин датчик укрепился в связи с развитием автоматических систем управления, как элемент обобщенной логической концепции датчик – устройство управления – исполнительное устройство – объект управления.

В автоматизированных системах управления датчики могут выступать в роли инициирующих устройств, приводя в действие оборудование, арматуру и программное обеспечение. Показания датчиков в таких системах, как правило, записываются на запоминающее устройство для контроля, обработки, анализа и вывода на дисплей или печатающее устройство. Огромное значение датчики имеют в робототехнике, где они выступают в роли рецепторов, посредством которых роботы и другие автоматические устройства получают информацию из окружающего мира и своих внутренних органов.

Основные характеристики датчиков:

- чувствительность (порог чувствительности) – наименьшее значение входной величины, приводящее к изменению измеряемой выходной;
- погрешность выходного сигнала – определена для нормальных условий эксплуатации, при изменении условий окружающей среды, погрешность увеличивается;
- диапазон измерения – минимально и максимально возможные значения величин, которые сможет детектировать датчик.

Классификация датчиков

По методу измерения:

- Активные (генераторные, сами воздействуют на окружающую среду)
- Пассивные (параметрические, окружающая среда воздействует на них)

По измеряемому параметру

- Датчики давления
- Датчики расхода
- Уровня
- Температуры
- Датчик концентрации
- Радиоактивности (также именуются детекторами радиоактивности или излучений)
- Перемещения
- Положения
- Фотодатчики
- Датчик углового положения
- Датчик вибрации
- Датчик механических величин
- Датчик влажности

По характеру выходного сигнала

- Дискретные
- Аналоговые
- Цифровые
- Импульсные

По среде передачи: проводные и беспроводные. При этом датчики могут являться частью какого-либо прибора, тогда они называются интегральные, а могут представлять собой независимое устройство, так называемые элементные датчики.

По способу питания: автономные (работают от батареи) и подключенные к сети электропитания.

Задание практической работы №5

Часть 1. Измерительные и исполнительные устройства стенда

Ознакомьтесь с устройствами в составе стенда и их характеристиками.

Рисунок 2 – Компоненты, расположенные на верхней крышке WB-demo-kit

Рисунок 3 - Компоненты, расположенные на нижней крышке WB-demo-kit

Опишите датчики и исполнительные устройства, согласно своему варианту:

№ варианта	Измеряемая величина, датчики и устройства *.
1	1. Датчик температуры 1-wire DS18B20 (1) 2. Устройство ИК-управления WB-MIR (4) 3. Концентрация CO2 в составе устройства WB-MSW v.3 (5)
2	1. Температура в составе устройства WB-MSW v.3 (5) 2. Освещенность в составе устройства WB-MS v.2 (12) 3. Многоканальный измеритель WB-MAP12H (22)
3	1. Относительная влажность в составе устройства WB-MSW v.3 (5) 2. Качество воздуха VOC в составе устройства WB-MS v.2 (12) 3. Модуль ввода-вывода WBIO-DO-R10A-8 (9)
4	1. Датчик температуры 1-wire DS18B20 (2) 2. Уровень шума в составе устройства WB-MSW v.3 (5) 3. Модуль реле 3-канальный WB-MR3 (21)
5	1. Освещенность в составе устройства WB-MSW v.3 (5) 2. Влажность в составе устройства WB-MS v.2 (12) 3. Преобразователь 1-Wire — Modbus RTU WB-M1W2 (3)
6	1. Модуль обнаружения протечек WB-MWAC (10) 2. Качество воздуха VOC в составе устройства WB-MSW v.3 (5) 3. Датчик протечки (23)
7	1. Датчик движения в составе устройства WB-MSW v.3 (5) 2. Датчик температуры в составе устройства WB-MS v.2 (12) 3. Диммер светодиодных лент на DIN-рейку WB-MRGBW-D (11)

В отчете необходимо отразить:

1. Название датчика/устройства;
2. Тип измерения (цифровой/аналоговый);
3. Измеряемые параметры и диапазон измерения;
4. Точность;
5. Напряжение питания;
6. Уникальный идентификатор датчика в веб-интерфейсе;
7. Использующийся протокол передачи данных;
8. Интерфейс управления (шина);
9. Описание входов и выходов, схема подключения.

Примечание 1. Пункты 2-4 обязательные только для датчиков.

Часть 2. Протоколы работы с устройствами

Опишите принцип работы, преимущества и недостатки, сферу применения следующих четырех технологий:

1. Modbus RTU;
2. 1-Wire;
3. I²C (IIC, англ. Inter-Integrated Circuit);
4. CAN.

Дополнительное задание практической работы №5

Часть 1. Опишите оборудование (датчики и актуаторы), необходимое для сборки аппаратных компонентов системы в вашем проекте, а также целесообразность выбора именно такого оборудования (например, на основе температурных условий его эксплуатации, точности, диапазона измерений, размера, веса и т.д.).

Если предполагается использовать программный эмулятор физического устройства, опишите набор оборудования, поведение которого вы собираетесь эмулировать.

В описании необходимо отразить:

1. Название датчика/устройства с указанием ссылки на подобное устройство в любом магазине;
2. Тип измерения (цифровой/аналоговый);
3. Измеряемые параметры и диапазон измерения;
4. Точность;
5. Напряжение питания;
6. Протокол передачи данных, которые поддерживает устройство;
7. Интерфейс управления (шина);
8. Описание входов и выходов, схема подключения.

Часть 2. Выберите технологии передачи данных от физического устройства в сеть Интернет. Приведите обоснование своего выбора (например, с точки зрения скорости передачи данных, объемов передаваемых данных, частоте, энергоэффективности и т.д.).

Практическая работа №6 – Основы работы с протоколом MQTT. Брокераж сообщений

MQTT протокол, основные особенности

MQTT (англ. message queuing telemetry transport) — упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу издатель-подписчик.

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.

Главные особенности протокола MQTT:

- Компактный и легковесный — минимальные накладные расходы на пересылку данных, для экономии трафика.
- Устойчивость к потерям — гарантированная доставка в условиях нестабильных сетевых подключений.
- Асинхронный — позволяет обслуживать большое количество устройств, и не зависит от сетевых задержек.
- Поддержка QoS — возможность управлять приоритетом сообщений и гарантировать доставку сообщения адресату.
- Динамическая конфигурация — не требует предварительно согласования полей и форматов данных, может конфигурироваться «на лету».
- Работает за NAT — клиенты могут находиться за NAT, только сервер (брокер) должен иметь реальный IP. Позволяет обойтись без VPN и пробрасывания портов.
- Удобная адресация — поля данных имеют текстовые названия, понятные для человека. Не нужно запоминать цифровые адреса и битовые смещения.

Протокол MQTT работает на прикладном уровне поверх TCP/IP и использует по умолчанию 1883 порт (8883 при подключении через SSL).

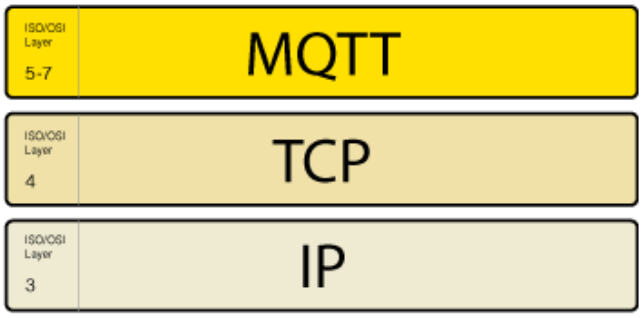


Рисунок 4 – уровни модели OSI

Принцип работы MQTT

Обмен сообщениями в протоколе MQTT осуществляется между клиентом (client), который может быть издателем или подписчиком (publisher/subscriber) сообщений, и брокером (broker) сообщений (например, Mosquitto MQTT).

Брокер (Broker) — это центральный узел MQTT, обеспечивающий взаимодействие клиентов. Обмен данными между клиентами происходит только через брокера. В качестве брокера может выступать серверное ПО или контроллер. В его задачи входит получение данных от клиентов, обработка и сохранение данных, доставка данных клиентам, и контроль за доставкой сообщений.

Publisher/Subscriber

Для понимания разницы между Publisher и Subscriber разберем простой пример: датчик влажности измеряет влажность в помещении, и? если она опустилась ниже определенного уровня, включается увлажнитель воздуха.

В данном случае датчик влажности выступает в роли Publisher: его задача сводится только к публикации данных в сторону брокера. Увлажнитель воздуха выступает в роли Subscriber: он подписывается на обновления данных о влажности и получает от брокера актуальные данные, при этом увлажнитель может сам решать, в какой момент включать увлажнение.

В этой схеме MQTT-клиенты, то есть датчик и увлажнитель, не знают о существовании друг друга, и не взаимодействуют напрямую. Брокер может получать данные из разных источников, проводить над ними манипуляции, например, рассчитывать среднее значение от нескольких датчиков, и уже обработанные данные возвращать подписчику.

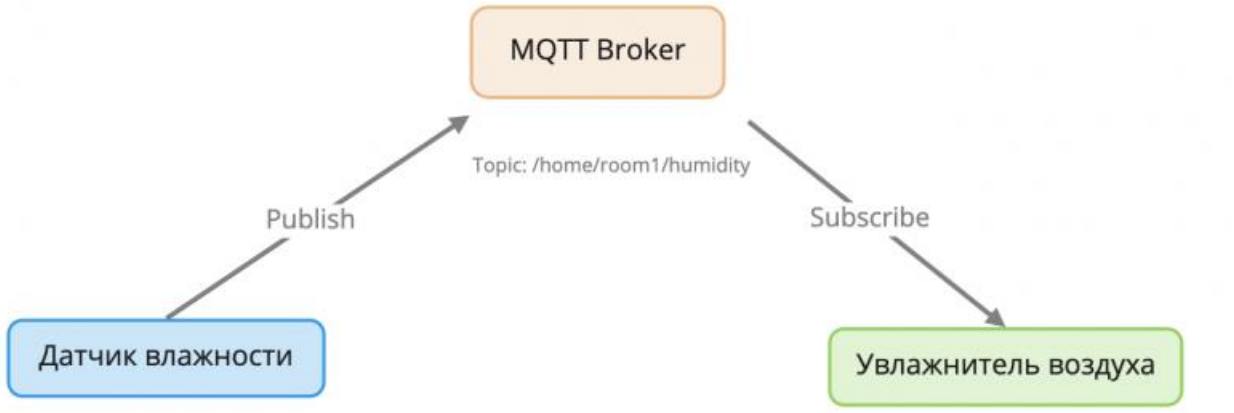


Рисунок 5 – Publisher посылает данные брокеру, Subscriber подписывается на обновления этих данных

При этом, асинхронность протокола MQTT предусматривает, что датчик и увлажнитель могут быть онлайн в разное время, терять пакеты, и быть недоступны. Брокер позаботится о том, чтобы сохранить в памяти последние данные, полученные от датчика, и обеспечить их доставку на увлажнитель.

Издатель отправляет данные на MQTT брокер, указывая в сообщении определенную тему, топик (topic). Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Устройства MQTT используют определенные типы сообщений для взаимодействия с брокером, ниже представлены основные:

- Connect – установить соединение с брокером
- Disconnect – разорвать соединение с брокером
- Publish – опубликовать данные в топик на брокере
- Subscribe – подписаться на топик на брокере
- Unsubscribe – отписаться от топика

Рисунок 6 – Схема простого взаимодействия между подписчиком, издателем и брокером

Семантика топиков

Для идентификации сущностей в MQTT используются топики (или каналы). Топики состоят из UTF8-символов, и имеют древовидную структуру, похожую на файловую систему в UNIX. Это удобный механизм, позволяющий называть сущности в человекопонятном виде.

Пример топиков в MQTT:

```
# Датчик температуры на кухне
home/kitchen/temperature
# Датчик температуры в спальне
home/sleeping-room/temperature
# Датчик освещенности на улице
home/outdoor/light
```

Такой подход позволяет наглядно видеть, какие данные передаются, и удобно разрабатывать и отлаживать код, без необходимости запоминать цифровой адрес размещения данных,

Иерархическая структура топиков имеет формат «дерева», что упрощает их организацию и доступ к данным. Топики состоят из одного или нескольких уровней, которые разделены между собой символом «/».

Пример топика, в который датчик температуры, расположенный в спальной комнате, публикует данные брокеру:

```
/home/living-space/living-room1/temperature
```

Подписчик может так же получать данные сразу с нескольких топиков, для этого существуют **wildcard**. Они бывают двух типов: одноуровневые и многоуровневые. Для более простого понимания рассмотрим в примерах каждый из них:

1. Одноуровневый wildcard. Для его использования применяется символ «+»

К примеру, нам необходимо получить данные о температуры во всех спальных комнатах:

```
/home/living-space/+/temperature
```

В результате получаем данные с топиков:

```
/home/living-space/living-room1/temperature
/home/living-space/living-room2/temperature
/home/living-space/living-room3/temperature
```

2. Многоуровневый wildcard. Для его использования применяется символ «#»

К примеру, чтобы получить данные с различных датчиков всех спален в доме:

```
/home/living-space/#
```

В результате получаем данные с топиков:

```
/home/living-space/living-room1/temperature
/home/living-space/living-room1/light1
/home/living-space/living-room1/light2
/home/living-space/living-room1/humidity
/home/living-space/living-room2/temperature
/home/living-space/living-room2/light1
```

Работа с MQTT в WirenBoard

Подробно о работе MQTT на контроллере WirenBoard вы можете ознакомиться в документации к оборудованию: https://wirenboard.com/wiki/MQTT

Задание практической работы №6

Часть 1. SHH-подключение

Подключитесь к консоли WirenBoard по протоколу SSH. Для этого используйте SSH-клиент PuTTy (или другой клиент):

1. Запустите PuTTY.

2. Выберите слева в поле **Category** ветку **Session**.
3. Ведите IP-адрес стенда
4. Укажите номер порта — 22.
5. Переключатель **Connection type** установите в положение **SSH**.
6. Нажмите кнопку **Open**.
7. При первом подключении к контроллеру появится окно запроса на приём от него ключа шифрования соединение — нажмите **Accept**.
8. Когда откроется окно консоли, в нём появится запрос имени пользователя - введите `user` и нажмите *Enter*; появится запрос пароля - введите `123123` (вводимые символы не будут отображаться) и нажмите *Enter*.
9. Появится приветственное сообщение - вы подключились к консоли контроллера.

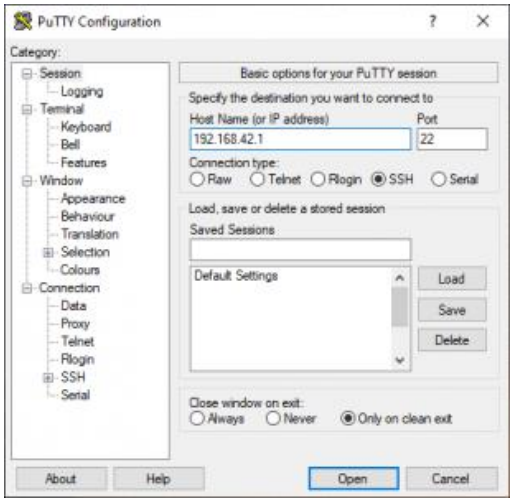


Рисунок 7 – Настройки соединения

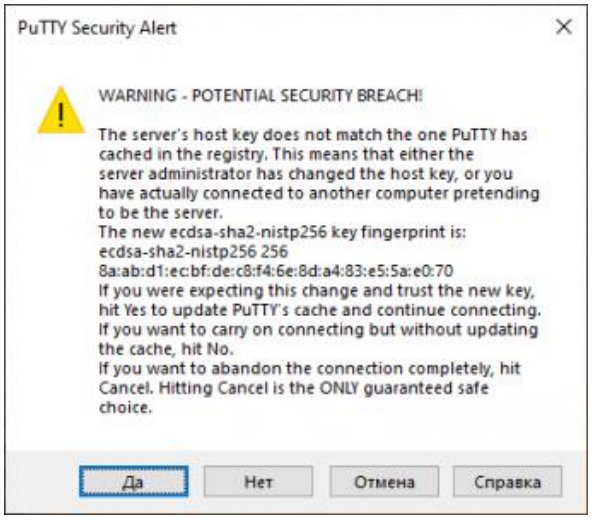


Рисунок 8 – Запрос на приём ключа шифрования

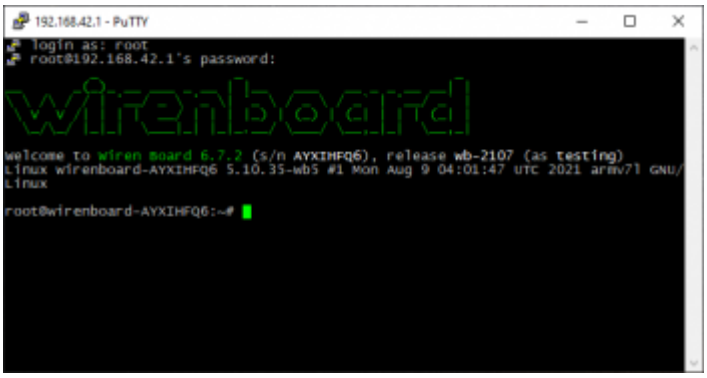


Рисунок 9 – Приветственный баннер контроллера

Часть 2. Подписка на топик

При помощи улиты **mosquitto_sub** подпишитесь на сообщения нескольких датчиков стенда, согласно варианту:

№ варианта	Датчики
1	1. Датчик температуры устройства WB-MSW v.3 (5) 2. Кнопка 27
2	1. Датчик CO_2 устройства WB-MSW v.3 (5) 2. Датчик протечки (23)
3	1. Датчик температуры 1-wire DS18B20 (2) 2. Кнопка 28
4	1. Датчик движения устройства WB-MSW v.3 (5) 2. Датчик температуры устройства WB-MS v.2 (12)
5	1. Датчик влажности устройства WB-MS v.2 (12) 2. Кнопка 29
6	1. Датчик движения устройства WB-MSW v.3 (5) 2. Качество воздуха VOC устройства WB-MS v.2 (12)
7	1. Датчик шума устройства WB-MSW v.3 (5) 2. Датчик освещенности устройства WB-MS v.2 (12)

Пример получения сообщений в результате успешной подписки:

```
~# mosquitto_sub -t '/devices/<Имя устройства>/controls/<Идентификатор датчика>' -v //команда на получение сообщения из топика

/devices//<Имя устройства>/controls/<Идентификатор датчика> 22.75 //в этой строке и ниже — вывод утилиты, полученные сообщения

/devices//<Имя устройства>/controls/<Идентификатор датчика> 22.75

/devices//<Имя устройства>/controls/<Идентификатор датчика> 22.75
```

Часть 3. Управление устройствами

Для управления устройством (изменения значения канала), необходимо отправить сообщение в топик `/devices/<device-id>/controls/<control-id>/on` (обратите внимание на /on в конце). Название топика можно также посмотреть в веб-интерфейсе контроллера во вкладке *Settings – MQTT Channels*. Это делается с помощью консольной команды mosquitto_pub. Например, данная команда отправляет сообщение «1» (логическую единицу, «включить») в топик, соответствующий подключённому по RS-485 релейном модуле WM-MRM2 с адресом 130.

```
~# mosquitto_pub -t "/devices/wb-mrm2_130/controls/Relay 1/on" -m "1"
```

Включите или измените поведение устройств посредством отправки сообщение в соответствующий топик согласно вариантам:

№ варианта	Действие
1	1. Включите звуковой сигнал 2. Включите и выключите вентилятор
2	1. Откройте и закройте шаровой кран 2. Включите индикатор 24

3	1. Включите индикатор 25 2. Включите и выключите RGB-ленту
4	1. Включите подсветку кнопки 29 2. Включите имитацию подачи воды
5	1. Включите индикатор 26 2. Включите и измените уровень громкости звукового сигнала
6	1. Включите индикатор устройства WB-MSW v.3 (5) 2. Включите подсветку кнопки 28
7	1. Включите и выключите вентилятор 2. Включите индикаторы 24 и 25

Часть 4*. Сообщения MQTT с внешнего устройства

**Примечание. Работа выполняется только в случае подключения стендов к сети Интернет.*

Настройте MQTT-мост (т.е. пересылку всех сообщений MQTT на любой облачный MQTT брокер и обратно) или создайте собственный брокер MQTT.

Обязательно верните файл конфигурации mosquito.conf контроллера к изначальному состоянию после выполнения работы!

Процесс настройки приведен в документации к контроллеру WirenBoard.

ВНИМАНИЕ! Необходимо редактировать файл конфигурации, находящийся в домашней директории пользователя **user** (а не `/etc/mosquitto/mosquitto.conf`)!

Дополнительное задание практической работы №6

Часть 1.

Рассмотрите следующие облачные платформы, их функционал и документацию и выберите облачную платформу, которую можно было бы использовать в вашей разрабатываемой системе (проекте).

- ThingsBoard;
- Rightech IoT Cloud;
- Microsoft Azure IoT;
- AdaFruit;
- HiveMQ;
- Yandex Cloud;
- OpenHAB;
- Любая IoT платформа на выбор, являющаяся свободным, бесплатным ПО и имеющая схожий с предыдущими функционал.

Часть 2.

С помощью диаграммы последовательности (UML Sequence Diagram) отобразите взаимодействие всех компонентов вашего проекта.

В качестве взаимодействующих объектов должны быть отражены:

- Пользователь;
- Все датчики, актуаторы и контроллеры физического устройства (вне зависимости от того, будет ли проект реализован с использованием физических компонентов или только с их эмуляторами);
- Облачная платформа (или объекты этой подсистемы по-отдельности);
- Пользовательский интерфейс (или объекты этой подсистемы по-отдельности);
- Другие объекты, которые вы сочтете необходимым отобразить/

Практическая работа №7 – Форматы представления данных

Синтаксическая совместимость

Системы Интернета вещей часто характеризуется большим количеством разнородных компонентов, и для корректного взаимодействия этих компонентов (т.е. для обмена данным без потерь и повреждения) необходимо поддерживать синтаксическую совместимость.

Некоторые форматы представления данных особенно хорошо подходят для обмена данными между компонентами системы Интернета вещей и между различными системами. Примером таких форматов представления данных может служить XML (расширяемый язык разметки) и JSON.

XML

XML (англ. eXtensible Markup Language) — расширяемый язык разметки. Рекомендован Консорциумом Всемирной паутины (W3C). Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Расширение XML — это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих какие атрибуты и элементы могут входить в состав других элементов. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как, собственно, XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

XML хранит данные в текстовом формате. Это обеспечивает независимый от программного и аппаратного обеспечения способ хранения, транспортировки и обмена данными. XML также облегчает расширение или обновление до новых операционных систем, новых приложений или новых браузеров без потери данных.

Преимущества XML:

Поддержка метаданных

Одним из самых больших преимуществ XML является то, что метаданные возможно помещать в теги в форме атрибутов (в JSON атрибуты будут добавлены как другие поля-члены в представлении данных, которые НЕ могут быть желательны).

Визуализация браузера

Большинство браузеров отображают XML в удобочитаемой и организованной форме. Древовидная структура XML в браузере позволяет пользователям естественным образом сворачивать отдельные элементы дерева. Эта функция будет особенно полезна при отладке.

Поддержка смешанного контента

Хорошим вариантом использования XML является возможность передачи смешанного контента в пределах одной и той же полезной нагрузки данных. Этот смешанный контент четко различается по разным тегам.

Введение в XML

Простейший XML-документ выглядит следующим образом:

```
<?xml version="1.0" encoding="windows-1251"?>
<book category="WEB">
   <title lang="en">Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price></price>
</book>
```

Первая строка — это XML декларация. Здесь определяется версия XML (1.0) и кодировка файла. На следующей строке описывается корневой элемент документа `<book>` (*открывающий тег*). Следующие 4 строки описывают дочерние элементы корневого элемента (`title, author, year, price`). Последняя строка определяет конец корневого элемента `</book>` (*закрывающий тег*).

Документ XML состоит из *элементов* (elements). Элемент начинается *открывающим тегом* (start-tag) в угловых скобках, затем идет *содержимое* (content) элемента, после него записывается *закрывающий тег* (end-teg) в угловых скобках.

Информация, заключенная между тегами, называется содержимым или значением элемента: `<author>Erik T. Ray</author>`. Т.е. элемент `author` принимает значение `Erik T. Ray`. Элементы могут вообще не принимать значения.

Элементы могут содержать *атрибуты*, так, например, открывающий тег `<title lang="en">` имеет атрибут `lang`, который принимает значение `en`. Значения атрибутов заключаются в кавычки (двойные или ординарные).

Некоторые элементы, не содержащие значений, допустимо записывать без закрывающего тега. В таком случае символ / ставится в конце открывающего тега:

```
<name first="Иван" second="Петрович" />
```

Структура XML

XML документ должен содержать корневой элемент. Этот элемент является «родительским» для всех других элементов.

Все элементы в XML документе формируют иерархическое дерево. Это дерево начинается с корневого элемента и разветвляется на более низкие уровни элементов.

Все элементы могут иметь подэлементы (дочерние элементы):

```
<корневой>
   <потомок>
     <подпотомок>.....</подпотомок>
   </потомок>
</корневой>
```

Правила синтаксиса (Валидность)

Структура XML документа должна соответствовать определенным правилам. XML-документ, отвечающий этим правилам, называется *валидным* (*англ.* Valid — правильный) или *синтаксически верным*. Соответственно, если документ не отвечает правилам, он является *невалидным*.

Основные правила синтаксиса XML:

1. Теги XML регистрозависимы — теги XML являются регистрозависимыми. Так, тег `<Letter>` не то же самое, что тег `<letter>`.

Открывающий и закрывающий теги должны определяться в одном регистре:

```
<Message>Это неправильно</message>
<message>Это правильно</message>
```

2. XML элементы должны соблюдать корректную вложенность:

```
<b><i>Некорректная вложенность</b></i>
<b><i>Корректная вложенность</i></b>
```

3. У XML документа должен быть корневой элемент — XML документ должен содержать один элемент, который будет родительским для всех других элементов. Он называется корневым элементом.

Примечание. В большинстве XML файлов отчетов для ФНС корневым элементом является `<Файл></Файл>`. После закрывающего тега `</Файл>` больше ничего быть не должно.

4. Значения XML атрибутов должны заключаться в кавычки:

```
<note date="12/11/2007">Корректная запись</note>
```

```
<note date=12/11/2007>Некорреткная запись</note>
```

Сущности

Некоторые символы в XML имеют особые значения и являются служебными. Если вы поместите, например, символ `<` внутри XML элемента, то будет сгенерирована ошибка, так как парсер интерпретирует его, как начало нового элемента.

В примере ниже будет сгенерирована ошибка, так как в значении `"ООО<Мосавтогруз>"` атрибута `НаимОрг` содержатся символы `<` и `>`.

```
<НПЮЛ ИННЮЛ="7718962261" КПП="771801001" НаимОрг="ООО<Мосавтогруз>"/>
```

Также ошибка будет сгенерирована и в слудющем примере, если название организации взять в обычные кавычки (английские двойные):

```
<НПЮЛ ИННЮЛ="7718962261" КПП="771801001" НаимОрг="ООО"Мосавтогруз""/>
```

Чтобы ошибки не возникали, нужно заменить символ `<` на его сущность. В XML существует 5 предопределенных сущностей:

Сущность	Символ	Значение
`<`	<	меньше, чем
`>`	>	больше, чем
`&`	&	амперсанд
`&apos;`	'	апостроф
`&quot;`	"	кавычки

Примечание: Только символы `<` и `&` строго запрещены в XML. Символ `>` допустим, но лучше его всегда заменять на сущность.

Таким образом, корректными будут следующие формы записей:

```
<НПЮЛ ИННЮЛ="7718962261" КПП="771801001"
НаимОрг="ООО&quot;Мосавтогруз&quot;"/>
```

или

```
<НПЮЛ ИННЮЛ="7718962261" КПП="771801001" НаимОрг="ООО«Мосавтогруз»"/>
```

В последнем примере английские двойные кавычки заменены на французские кавычки («ёлочки»), которые не являются служебными символами.

Кодировки

И еще один важный момент, который стоит рассмотреть — кодировки. Существует множество кодировок. Самыми распространенными кириллическими кодировками являются Windows-1251 и UTF-8. Последняя является одним из стандартов, но большая часть ФНС отчетности имеет кодировку Windows-1251.

В XML файле кодировка объявляется в декларации:

```
<?xml version="1.0" encoding="windows-1251"?>
```

JSON

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

Преимущества JSON

1. Более подробная структурная информация в документе

XML требует открытия и закрытия тегов, а JSON использует пары имя / значение, четко обозначенные «{«и»}» для объектов, «[«и»]» для массивов, «,» (запятую) для разделения пары и «:»(двоеточие) для отделения имени от значения.

Можно легко различить число 1 и строку "1", поскольку числа, строки (и логические значения) представлены по-разному в JSON.

Можно легко различать отдельные элементы и коллекции одного размера (используя массивы JSON).

2. Малый размер сообщения

При одинаковом объеме информации JSON почти всегда значительно меньше, что приводит к более быстрой передаче и обработке.

3. Близость к javascript

JSON является подмножеством JavaScript, поэтому код для его анализа и упаковки вполне естественно вписывается в код JavaScript.

4. Проще представить значение null

Синтаксис и структура JSON

Объект JSON имеет вид «ключ-значение» и обычно записывается в фигурных скобках. При работе с JSON все объекты хранятся в файле .json, но также они могут существовать как отдельные объекты в контексте программы.

Объект JSON выглядит так:

```
{
"first_name" : "John",
"last_name" : "Smith",
"location" : "London",
"online" : true,
"followers" : 987
}
```

Это очень простой пример. Объект JSON может содержать множество строк.

Как видите, объект состоит из пар «ключ-значение», которые заключены в фигурные скобки. Большая часть данных в JSON записывается в виде объектов.

Между ключом и значением ставится двоеточие. После каждой пары нужно поставить запятую. В результате получается:

```
"key" : "value", "key" : "value", "key": "value"
```

Ключ в JSON находится слева. Ключ нужно помещать в двойные кавычки. В качестве ключа можно использовать любую валидную строку. В рамках одного объекта все ключи должны быть уникальны. Ключ может содержать пробел (“first name”), но при программировании могут возникнуть проблемы с доступом к такому ключу. Потому вместо пробела лучше использовать подчеркивание (“first_name”).

Значения JSON находятся в правой части столбца. В качестве значения можно использовать любой простой тип данных:

- Строки
- Числа
- Объекты
- Массивы
- Логические данные (true или false)
- Ноль

Значения могут быть представлены и сложными типами данных (например, объектами или массивами JSON).

JSON поддерживает индивидуальный синтаксис каждого из перечисленных выше типов данных: если значение представлено строкой, то оно будет взято в кавычки, а если числом, то нет.

Как правило, данные в файлах .json записываются в столбик, однако JSON можно записать и в строку:

```
{ "first_name" : "John", "last_name": "Smith",  "online" : true, }
```

Так обычно записываются данные JSON в файлы другого типа.

Записывая данные JSON в столбец, вы повышаете удобочитаемость файла (особенно если данных в файле много). JSON игнорирует пробелы между столбцами, потому с их помощью вы можете разделить данные на удобное для восприятия количество столбцов.

```
{
"first_name"  :  "John",
"last_name"   :  "Smith",
"online"      :  true
}
```

Обратите внимание: объекты JSON очень похожи на объекты JavaScript, но это не один и тот же формат. К примеру, в JavaScript можно использовать функции, а в JSON нельзя.

Главным преимуществом JSON является то, что данные в этом формате поддерживают многие популярные языки программирования, потому их можно быстро передать.

Теперь вы знакомы с базовым синтаксисом JSON. Но файлы JSON могут иметь сложную, иерархическую структуру, включающую в себя вложенные массивы и объекты.

Сложные типы в JSON

JSON может хранить вложенные объекты и массивы, которые будут передаваться в качестве значения присвоенного им ключа.

Вложенные объекты

Ниже вы найдёте пример – файл users.json, в котором содержатся данные о пользователях. Для каждого пользователя

(“john”, “jesse”, “drew”, “jamie”) в качестве значения передаётся вложенный объект, который, в свою очередь, тоже состоит из ключей и значений.

Примечание: Первый вложенный объект JSON выделен красным.

```
{
" john" : {
"username"  : " John",
"location"  : "London",
"online"    : true,
"followers" : 987
},
"jesse" : {
"username"  : "Jesse",
"location"  : "Washington",
"online"    : false,
"followers" : 432
},
"drew" : {
"username"  : "Drew",
"location"  : "Paris",
"online"    : false,
"followers" : 321
},
"jamie" : {
"username"  : "Jamie",
"location"  : "Berlin",
"online"    : true,
"followers" : 654
}
}
```

Обратите внимание: фигурные скобки используются и во вложенном, и в основном объекте. Запятые во вложенных объектах используются так же, как и в обычных.

Вложенные массивы

Данные можно вкладывать в JSON с помощью массивов JavaScript, которые будут передаваться как значения. В JavaScript в начале и в конце массива используются квадратные скобки ([ ]). Массив – это упорядоченный набор данных, который может содержать данные различных типов.

Массив используют для передачи большого количества данных, которые можно сгруппировать. Для примера попробуем записать данные о пользователе.

```
{
"first_name" : "John",
"last_name" : "Smith",
"location" : "London",
"websites" : [

{

"description" : "work",
"URL" : "https://www.johnsmithsite.com/"

},
{

"desciption" : "tutorials",
"URL" : "https://www.johnsmithsite.com/tutorials"

}

],

"social_media" : [

{

"description" : "twitter",
"link" : "https://twitter.com/johnsmith"

},
{

"description" : "facebook",
"link" : "https://www.facebook.com/johnsmith"

},
{

"description" : "github",
"link" : "https://github.com/johnsmith"

}

]

}
```

Ключам «websites» и «social_media» в качестве значений присвоены массивы, которые помещаются в квадратные скобки.

При помощи вложенных массивов и объектов можно создать сложную иерархию данных.

Работа с XML и JSON

Для работы с объектами XML и JSON существуют стандартные библиотеки.

Например, для языка программирования Python это:

- Модуль json: https://docs.python.org/3/library/json.html
- Модуль ElementTree: https://docs.python.org/3/library/xml.etree.elementtree.html

Задание практической работы №7

1. С компьютера в аудитории или личного устройства подпишитесь на несколько MQTT-топиков стенда согласно вариантам.

№ варианта	Датчики
1	1. Датчик температуры устройства WB-MSW v.3 (5) 2. Датчик движения устройства WB-MSW v.3 (5) 3. Датчик шума устройства WB-MSW v.3 (5) 4. Датчик освещенности устройства WB-MS v.2 (12)
2	1. Датчик шума устройства WB-MSW v.3 (5) 2. Датчик освещенности устройства WB-MS v.2 (12) 3. Датчик CO_2 устройства WB-MSW v.3 (5) 4. Датчик температуры 1-wire DS18B20 (2)
3	1. Датчик температуры 1-wire DS18B20 (1) 2. Датчик шума устройства WB-MSW v.3 (5) 3. Датчик CO_2 устройства WB-MSW v.3 (5) 4. Качество воздуха VOC устройства WB-MS v.2 (12)
4	1. Датчик движения устройства WB-MSW v.3 (5) 2. Датчик шума устройства WB-MSW v.3 (5) 3. Датчик освещенности устройства WB-MS v.2 (12) 4. Датчик температуры устройства WB-MS v.2 (12)
5	1. Датчик CO_2 устройства WB-MSW v.3 (5) 2. Датчик шума устройства WB-MSW v.3 (5) 3. Датчик освещенности устройства WB-MS v.2 (12) 4. Датчик температуры устройства WB-MSW v.3 (5)
6	1. Датчик температуры 1-wire DS18B20 (2) 2. Датчик влажности устройства WB-MSW v.3 (5) 3. Качество воздуха VOC устройства WB-MS v.2 (12) 4. Датчик шума устройства WB-MSW v.3 (5)
7	1. Датчик движения устройства WB-MSW v.3 (5) 2. Качество воздуха VOC устройства WB-MS v.2 (12) 3. Датчик влажности устройства WB-MSW v.3 (5) 4. Датчик температуры 1-wire DS18B20 (1)

2. На любом языке программирования реализуйте программу (скрипт), которая бы каждые 5 секунд упаковывала последние полученные данные в файлы формата JSON и XML. В одной записи должно быть 6 полей: 4 показаний датчиков, время формирования файла, номер чемодана (последние две цифры IP-адреса).
3. На любом языке программирования реализуйте программу-парсер, которая бы выводила в консоль данные, полученные из сгенерированных в п.2 файлов.

В отчете приведите:

1. Команды для подписки на топики

2. Структуру данных, получаемых от стенда с описанием каждого поля. Пример структуры данных, получаемых от физического объекта:

```
id: int
temperature: float
humidity: float
name: string
```

3. Листинг с комментариями с указанием языка программирования для пунктов 2 и 3 задания.
4. Скриншоты результатов работы скриптов пунктов 2 и 3 задания
5. Краткое описание сторонних библиотек и используемых из них функций, использованных для реализации заданий пунктов 2 и 3.

Дополнительное задание практической работы №7

Часть 1.

1. Приведите структуру данных, получаемых от физического устройства разрабатываемой системы (или mock-объекта, его заменяющего), с указанием типов данных и описанием каждого поля. Пример структуры данных, получаемых от физического объекта:

```
id: int
temperature: float
humidity: float
name: string
```

2. На любом языке программирования реализуйте программу (скрипт), которая бы генерировала JSON-файл на основе данных из предыдущего пункта задания. Для демонстрации работы скрипта достаточно считать вводимые пользователем в консоли показатели.
3. На любом языке программирования реализуйте программу-парсер, которая бы выводила в консоль данные, полученные из сгенерированного в п.2 файла.

В отчете приведите структуру данных, листинги с комментариями, указанием языка программирования и результат выполнения пунктов 2 и 3.

Часть 2.

Соберите физическое устройство или реализуйте на любом языке программирования его программный эмулятор. Эмулятор должен генерировать поток данных и выводить их в консоли.

В отчете приведите схему подключения и фотографии собранного устройства для физического устройства или листинг кода с комментариями его программного эмулятора.

Практическая работа №8 – Визуализация данных в Интернете вещей

Визуализация данных

После сбора показаний с датчиков и сохранения данных в системах интернета вещей зачастую предусмотрена возможность визуализации для пользователей первичных данных или результата их агрегации и анализа.

Визуализация данных — это представление данных в виде, который обеспечивает наиболее эффективную работу человека по их изучению. Визуализация данных — важная функция средств бизнес-анализа и ключ к расширенной аналитике. Визуализация помогает оценивать значение информации или данных, создаваемых сегодня. Под визуализацией данных подразумевается представление информации в графической форме, например в виде круговой диаграммы, графика или визуального представления другого типа.

Если проводить классификацию по объектам, то стоит выделить визуализацию:

- числовых данных (детерминированные зависимости – графики, диаграммы, временные ряды; статистические распределения – гистограммы, матрицы диаграмм рассеяния)
- иерархий (диаграммы узлы-связи, дендрограммы)
- сетей (графы, дуговые диаграммы)
- геовизуализацию (карты, картограммы)

В отличие от обычного графического интерфейса, средства визуализации данных обеспечивают способность одновременного отображения большого числа разнотипных данных, возможность их сравнения, выделения выпадающих значений.

Качественная визуализация данных имеет критическое значение для анализа данных и принятия решений на их основе. Визуализация позволяет быстро и легко замечать и интерпретировать связи и взаимоотношения, а также выявлять развивающиеся тенденции, которые не привлекли бы внимания в виде необработанных данных. В большинстве случаев для интерпретации графических представлений не требуется специальное обучение, что сокращает вероятность недопонимания.

Продуманное графическое представление не только содержит информацию, но и повышает эффективность ее восприятия за счет наглядности, привлечения внимания и удержания интереса в отличие от таблиц и документов.

Основные способы визуализации (графические представления)

Выбор графического представления осуществляется с учетом типа данных и их предназначения. Некоторые представления лучше подходят для определенного типа данных, чем другие: например, гистограмма или круговая диаграмма. Большинство средств визуализации предлагает широкий выбор вариантов отображения данных, от обычных линейных графиков и столбчатых диаграмм до временных шкал, карт, зависимостей, гистограмм и настраиваемых представлений.

Графики

Линейный график

Линейные графики используются для отображения количественных показателей за непрерывный интервал или определенный период времени. Это наиболее популярный график для демонстрации трендов и соотношения показателей (при использовании нескольких линий). Линейные графики очень полезны для получения «общей картины» за определенный промежуток времени и наблюдения за развитием в этот период времени.

Чтобы нарисовать линейный график, необходимо сначала отметить точки данных на декартовой системе координат, а затем провести между этими точками линию. Как правило, на оси У отмечаются количественные значения, а на оси х либо качественные значения, либо шкала последовательностей. Отрицательные значения можно отображать ниже оси Х.

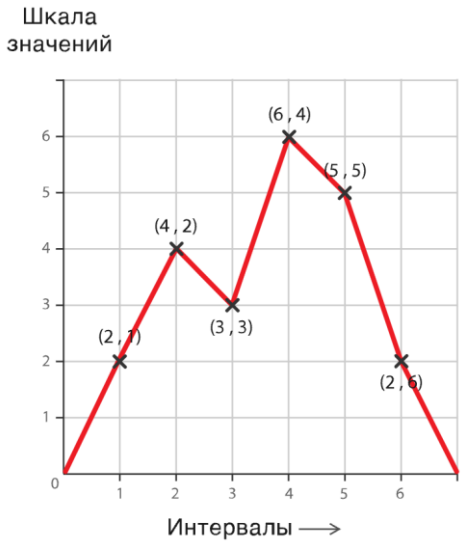


Рисунок 10 – Структура линейного графика

График плотности

Также известен как график ядра плотности, график плотности распределения вероятности.

График плотности – это инструмент визуализации распределения данных за непрерывный или определенный интервал времени. Этот график представляет собой разновидность гистограммы, где для отображения значений используется ядерное сглаживание, которое позволяет отобразить более гладкое распределение за счет сглаживания изменений параметров. Верхние точки графика плотности помогают отобразить, где именно в определенный интервал времени сконцентрирована совокупность значений.

Преимуществом графиков плотности перед гистограммами является то, что они лучше выявляют форму распределения, поскольку на них не оказывает влияние ряд используемых атрибутов (например, полосы в традиционной гистограмме). Гистограмма всего из 4 атрибутов не сможет отобразить распределение так же наглядно, как гистограмма из 20 атрибутов. Однако для графика плотности это не проблема.

Рисунок 11 – Структура графика плотности

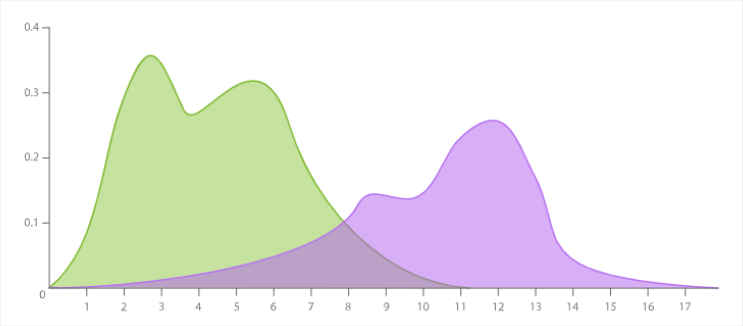


Рисунок 12 – Пример графика плотности

Диаграммы

Еще один распространенный способ визуализации. Показывают соотношения набора данных или связи внутри набора данных. В основном строятся вокруг осей, но не всегда. Также их можно построить по секторам или полярной системе координат.

Гистограмма

Гистограмма визуализирует распределение данных в рамках непрерывного интервала или ограниченного периода времени. Каждая полоса на гистограмме представляет в табличной форме частотность за определенный интервал/бин. Общая площадь гистограммы равна количеству данных.

Гистограммы помогают определить концентрацию значений, предельные значения и наличие пробелов или отклонений. Кроме того, они удобны для составления приблизительного обзора распределения вероятностей.

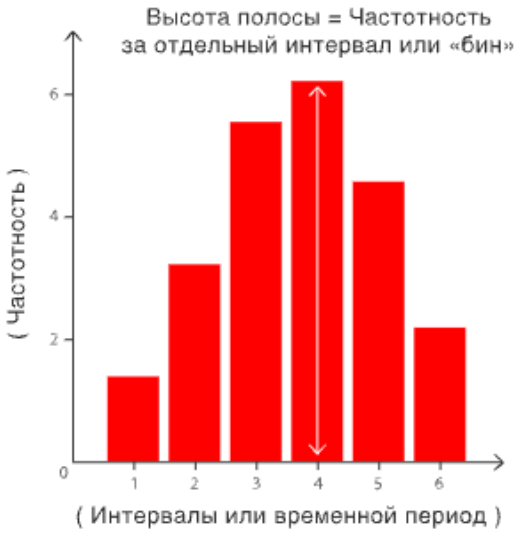


Рисунок 13 – Структура гистограммы

Столбиковая диаграмма

Также известна как линейчатая или полосчатая диаграмма.

Классическая столбиковая диаграмма оперирует горизонтальными или вертикальными столбцами для демонстрации дискретных, числовых сравнений между разными категориями. На одной оси диаграммы представлены конкретные сравниваемые категории, а на другой – шкала дискретных значений.

Столбиковые диаграммы отличаются от гистограмм тем, что не отражают непрерывное развитие в пределах определенного интервала. Дискретные данные столбиковых диаграмм – это данные по категориям, которые отвечают на вопрос: «Сколько?» - по каждой из категорий.

Единственный крупный недостаток столбиковых диаграмм – это оформление условных обозначений при большом количестве полос.

Рисунок 14 – Структура столбиковой диаграммы

Столбиковая диаграмма с группировкой

Также известна как столбиковая диаграмма с группировкой или кластерная столбиковая диаграмма.

Данный вариант столбиковой диаграммы используется, когда на одной оси располагается два или несколько наборов данных рядом друг с другом и сгруппированных по категориям.

Как и в случае со столбиковыми диаграммами, длина полос используется для отображения дискретных числовых данных, сравниваемых по категориям. Каждому набору данных присваивается определенный цвет или оттенок одной цветовой гаммы. Группы полос отделяются друг от друга промежутком.

Столбиковые диаграммы с группировкой, как правило, используются для сравнения сгруппированных показателей или категорий с другими группами тех же показателей или типов категории. Множественные полосчатые диаграммы можно также использовать для сравнения мини-гистограмм, где каждая полоса в группе представляет значительный интервал переменной.

Недостаток столбиковых диаграмм с группировкой заключается в том, что, чем больше полос в одной группе, тем сложнее читать диаграмму.

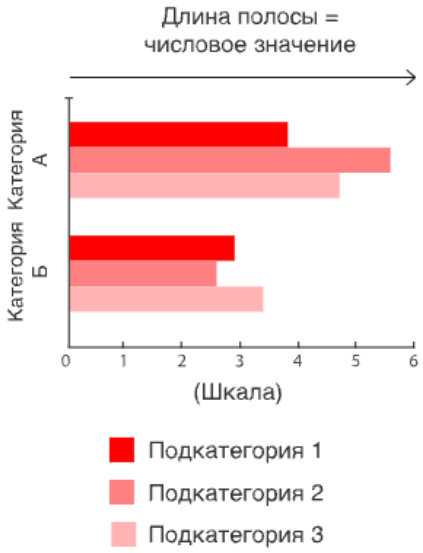


Рисунок 15 – Структура столбиковой диаграммы с группировкой

Круговая диаграмма

Известна также как секторная диаграмма в круге.

Круговые диаграммы широко используются в презентациях и офисной документации. Они позволяют показать пропорциональное и процентное соотношение между категориями за счет деления круга на пропорциональные сегменты. Длина каждой дуги представляет собой пропорциональную долю каждой категории, в то время как круг целиком представляет общую сумму всех данных, равную 100%.

Круговые диаграммы идеально подходят для того, чтобы быстро сформировать представление о пропорциональном распределении данных. Однако у этой диаграммы есть ряд существенных недостатков:

Количество отображаемых значений очень ограничено, поскольку, чем больше количество значений, тем меньше размер каждого отдельного сегмента. Соответственно, они не подходят для работы с большими объемами данных.

Они занимают больше места, чем альтернативные графики, например, 100%-ные стопочные диаграммы. Основные причины – размер и, как правило, необходимость в отдельном описании условных обозначений.

Они не очень удобны для проведения точных сравнений между показателями, поскольку визуально площадь сегментов сложнее сравнивать, нежели длину.

Несмотря на это зачастую весьма эффективно сравнивать с помощью круговых диаграмм конкретную категорию (одну часть круга) в рамках общей картины.

Рисунок 16 – Структура круговой диаграммы

Радиальная диаграмма

Также известен как паукообразная диаграмма, радарный график, полярный график.

Радиальная диаграмма – это инструмент, позволяющий проводить сравнение между множественными количественными переменными. Именно поэтому он удобен для выявления того, какие переменные имеют одинаковые значения и существуют ли среди значений этих переменных выбросы. Радиальные диаграммы также используются, чтобы выявить максимальные и минимальные значения переменных в пределах набора данных, благодаря чему этот инструмент особенно эффективен для отображения результатов деятельности.

Каждой переменной присваивается ось, имеющая начало в центре. Все оси располагаются радиально с одинаковым промежутком между друг другом, при этом на всех осях используется одинаковая шкала. Сеточные линии, которые соединяют оси между собой, зачастую используются в качестве ориентира. Значение каждой переменной отмечается на ее оси, а все переменные в наборе данных соединяются линиями, образуя многоугольник.

В то же время у радиальной диаграммы есть ряд серьезных недостатков:

При наличии множества многоугольников на одной радиальной диаграмме, она становится трудночитаемой, путанной и перегруженной. Особенно, если многоугольники заполняются цветом, поскольку верхний многоугольник перекрывает собой все остальные.

Слишком большое количество переменных приводит к появлению слишком большого количества осей, что также затрудняет прочтение диаграммы. Соответственно, лучше всего использовать простой вариант графика с ограниченным количеством используемых переменных.

Еще одним недостатком радиальных диаграмм является то, что они не очень удобны для сравнения значений переменных. Несмотря на яркий эффект, создаваемый паутинообразной сеткой, сравнивать значения на одной прямой оси гораздо легче.

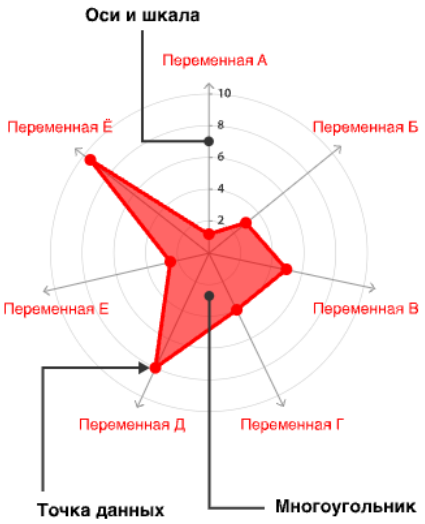


Рисунок 17 – Структура радиальной диаграммы

Инструменты визуализации данных

Существует множество инструментов визуализации данных, самые популярные библиотеки для работы с визуализацией в Python:

- Matplotlib https://matplotlib.org/
- Seaborn https://seaborn.pydata.org/
- Plotly https://plotly.com/python/

Каждая из них обладает своими преимуществами и недостаткам для использования в конкретных случаях. Далее мы будем рассматривать построение графиков и гистограмм с использованием библиотеки Matplotlib – пожалуй, самой популярной библиотекой Python для визуализации данных.

Линейные графики

Линейные графики являются самыми простыми из всех. Такой график — это последовательность точек данных на линии. Каждая точка состоит из пары значений (x, y), которые перенесены на график в соответствии с масштабами осей (x и y).

Визуализация данных в виде линейного графика — максимально простая задача. Достаточно передать объект в качестве аргумента функции `plot()` для получения графика с несколькими линиями.

```
import pandas as pd
import matplotlib.pyplot as plt

data = {'series1':[1,3,4,3,5],
```

```
        'series2':[2,4,5,2,4],
        'series3':[3,2,3,1,3]}
df = pd.DataFrame(data)
x = np.arange(5)
plt.axis([0,5,0,7])
plt.plot(x,df)
plt.legend(data, loc=2)
plt.show()
```

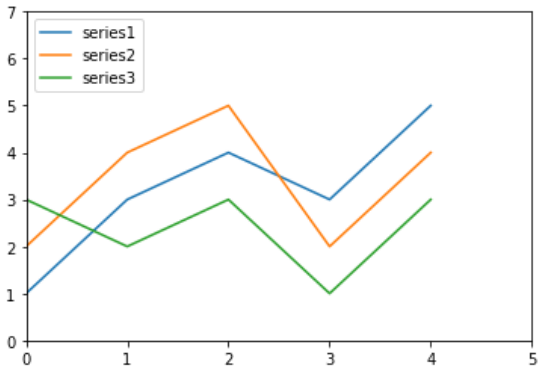


Рисунок 18 – Пример построения линейного графика

Гистрограммы

Гистограмма состоит из примыкающих прямоугольников, расположенных вдоль оси x, которые разбиты на дискретные интервалы, их называют `bins`. Их площадь пропорциональна частоте конкретного интервала. Такой способ визуализации часто используют в статистике для демонстрации распределения.

Для представления гистограммы в pyplot есть функция `hist()`. У нее также есть особенности, которых не найти у других функций, отвечающих за создание графиков. `hist()` не только рисует гистограмму, но также возвращает кортеж значений, представляющих собой результат вычислений гистограммы. Функция `hist()` может реализовывать вычисление гистограммы, чего достаточно для предоставления набора значений и количества интервалов, на которых их нужно разбить. Наконец `hist()` отвечает за разделение интервала на множество и вычисление частоты каждого. Результат этой операции не только выводится в графической форме, но и возвращается в виде кортежа.

Для понимания операции лучше всего воспользоваться практическим примером. Сгенерируем набор из 100 случайных чисел от 0 до 100 с помощью `random.randint()`.

```
pop = np.random.randint(0,100,100)
pop
array([33, 90, 10, 68, 18, 67,  6, 54, 32, 25, 90,  6, 48, 34, 59, 70, 37,
       50, 86,  7, 49, 40, 54, 94, 95, 20, 83, 59, 33,  0, 81, 18, 26, 69,
        2, 42, 51,  7, 42, 90, 94, 63, 14, 14, 71, 25, 85, 99, 40, 62, 29,
       42, 27, 98, 30, 89, 21, 78, 17, 33, 63, 80, 61, 50, 79, 38, 96,  8,
```

```
       85, 19, 76, 32, 19, 14, 37, 62, 24, 30, 19, 80, 55,  5, 94, 74, 85,
       59, 65, 17, 80, 11, 81, 84, 81, 46, 82, 66, 46, 78, 29, 40])
```

Дальше создаем гистограмму из этих данных, передавая аргумент функции `hist()`. Например, нужно разделить данные на 20 интервалов (значение по умолчанию — 10 интервалов). Для этого используется именованный аргумент `bin`.

```
n, bin, patches = plt.hist(pop, bins=20)
plt.show()
```

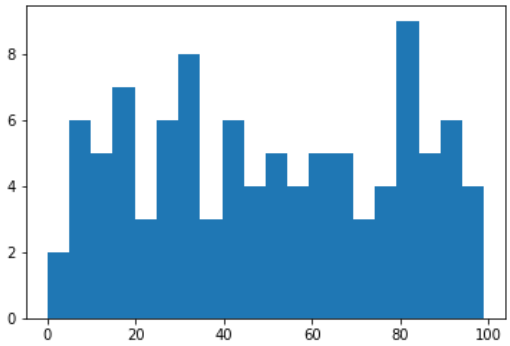


Рисунок 19 – Пример построения гистограммы

Круговая диаграмма

Еще один способ представления данных — круговая диаграмма, которую можно получить с помощью функции `pie()`.

Даже для нее нужно передать основной аргумент, представляющий собой список значений. Пусть это будут проценты (где максимально значение — 100), но это может быть любое значение. А уже сама функция определит, сколько будет занимать каждое значение.

Также в случае с этими графиками есть другие особенности, которые определяются именованными аргументами. Например, если нужно задать последовательность цветов, используется аргумент `colors`. В таком случае придется присвоить список строк, каждая из которых будет содержать название цвета. Еще одна возможность — добавление меток каждой доле. Для этого есть `labels`, которой присваивает список строк с метками в последовательности.

А чтобы диаграмма была идеально круглой, необходимо в конце добавить функцию `axix()` со строкой `equal` в качестве аргумента. Результатом будет такая диаграмма.

```
labels = ['Nokia','Samsung','Apple','Lumia']
values = [10,30,45,15]
colors = ['yellow','green','red','blue']
plt.pie(values,labels=labels,colors=colors)
plt.axis('equal')
plt.show()
```

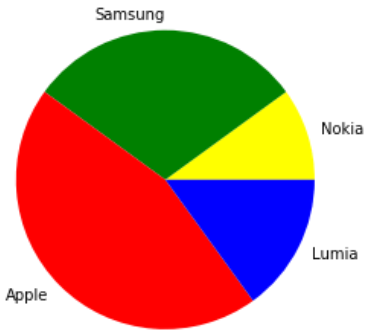


Рисунок 20 – Пример построения круговой диаграммы

Чтобы сделать диаграмму более сложной, можно «вытащить» одну из частей. Обычно это делается с целью акцентировать на ней внимание. В этом графике, например, для выделения Nokia. Для этого используется аргумент `explode`. Он представляет собой всего лишь последовательность чисел с плавающей точкой от 0 до 1, где 1 — положение целиком вне диаграммы, а 0 — полностью внутри. Значение между соответствуют среднему градусу извлечения.

Заголовок добавляется с помощью функции `title()`. Также можно настроить угол поворота с помощью аргумента `startangle`, который принимает значение между 0 и 360, обозначающее угол поворота (0 – значение по умолчанию). Следующий график показывает все изменения.

```
labels = ['Nokia','Samsung','Apple','Lumia']
values = [10,30,45,15]
colors = ['yellow','green','red','blue']
explode = [0.3,0,0,0]
plt.title('A Pie Chart')
plt.pie(values,labels=labels,colors=colors,explode=explode,startangle=180)
plt.axis('equal')
plt.show()
```

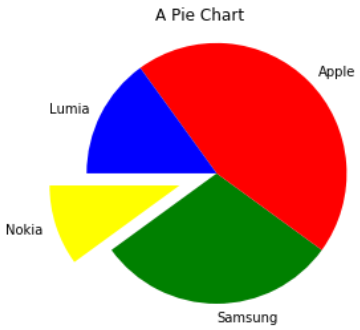


Рисунок 21 – Пример построения круговой диаграммы

Сейчас у диаграммы нет осей, поэтому сложно передать точное разделение. Чтобы решить эту проблему, можно использовать `autopct`, который добавляет в центр каждой части текст с соответствующим значением.

Чтобы сделать диаграмму еще более привлекательной визуально, можно добавить тень с помощью `shadow` со значением `True`. Результат — следующее изображение.

```
labels = ['Nokia','Samsung','Apple','Lumia']
values = [10,30,45,15]
colors = ['yellow','green','red','blue']
explode = [0.3,0,0,0]
plt.title('A Pie Chart')
plt.pie(values,labels=labels,colors=colors,explode=explode,shadow=True,autopct='%1.1f%%',startangle=180)
plt.axis('equal')
plt.show()
```

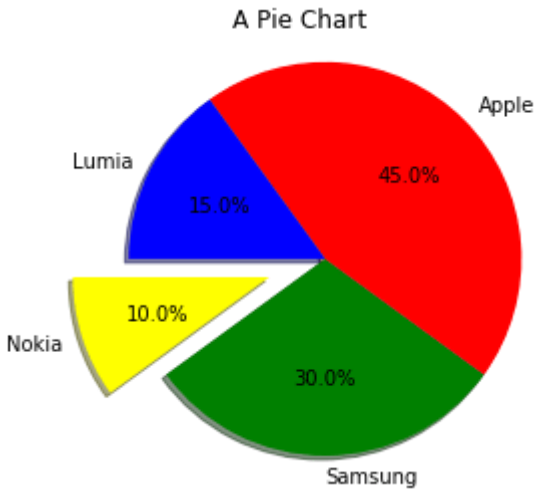


Рисунок 22 – Пример построения круговой диаграммы

Задание практической работы №8

1. Подпишитесь на несколько MQTT топиков стенда согласно вариантам с компьютера в аудитории или личного устройства и собирайте данные с датчиков в течение 10 или более минут.

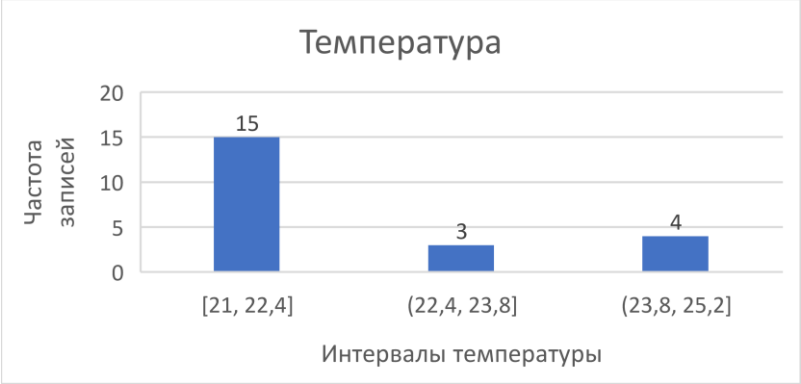
   Получаемые данные должны сохраняться в локальную базу данных или CSV-файл (на выбор).

№ варианта	Датчики
1	1. Датчик температуры устройства WB-MSW v.3 (5) 2. Датчик движения устройства WB-MSW v.3 (5) 3. Напряжение на любом устройстве стенда
2	1. Датчик шума устройства WB-MSW v.3 (5) 2. Датчик освещенности устройства WB-MS v.2 (12)

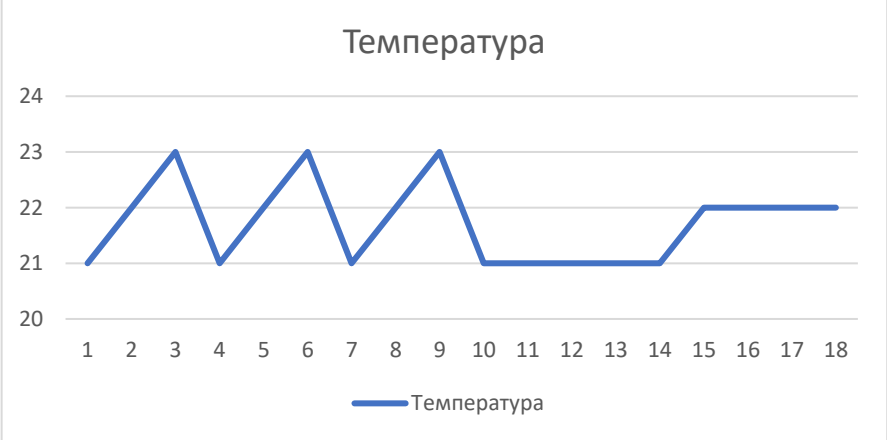
	3. Напряжение на любом устройстве стенда
3	1. Датчик шума устройства WB-MSW v.3 (5) 2. Датчик CO_2 устройства WB-MSW v.3 (5) 3. Напряжение на любом устройстве стенда
4	1. Датчик движения устройства WB-MSW v.3 (5) 2. Датчик температуры устройства WB-MS v.2 (12) 3. Напряжение на любом устройстве стенда
5	1. Датчик CO_2 устройства WB-MSW v.3 (5) 2. Датчик освещенности устройства WB-MS v.2 (12) 3. Напряжение на любом устройстве стенда
6	1. Датчик влажности устройства WB-MSW v.3 (5) 2. Датчик шума устройства WB-MSW v.3 (5) 3. Напряжение на любом устройстве стенда
7	1. Датчик влажности устройства WB-MSW v.3 (5) 2. Датчик температуры 1-wire DS18B20 (1) 3. Напряжение на любом устройстве стенда

2. Напишите скрипты, позволяющие визуализировать полученные статистические данные в виде:

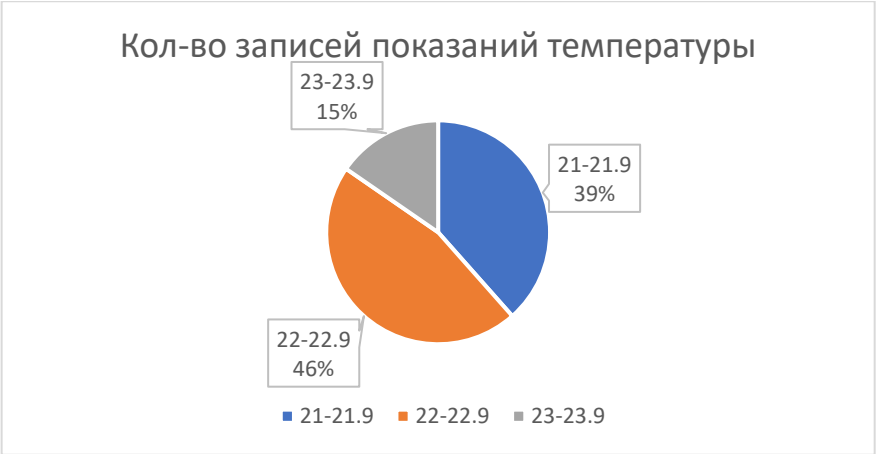
- Столбиковой диаграммы (гистограммы) (например, частоты показаний датчика)



- Линейного графика (например, показаний по времени)



- Круговой диаграммы (например, распределения показаний по времени)



Необходимо реализовать визуализацию данных всех трех перечисленных типов (т.е. для каждого датчика необходимо выбрать тип визуализации на свое усмотрение и не повторяться в типах визуализации в рамках одного варианта).

Дополнительное задание практической работы №8

Поднимите на физическом устройстве (или на устройстве, где запущен его программный эмулятор) MQTT-брокер. Показания с датчиков должны публиковаться в соответствующие им топики.

Подпишитесь с другого устройства (компьютера в аудитории или личного устройства) на топики физического устройства. Полученные от MQTT-брокера данные сохраняйте в локальную базу данных.

В отчете опишите процесс настройки MQTT-брокера (снабжая текстовое описание скриншотами), структуру топиков, процесс подключения другого устройства к MQTT-брокеру и результат получения и сохранения данных от физического устройства.

Требования к отчету по ПР №5-8:

По итогу выполнения практических работ №5-8 необходимо оформить единый отчёт, включающий:

1. Титульный лист;
2. Оглавление;
3. Описание датчиков и актуаторов, технологий передачи данных из основной части ПР №5;
4. Процесс и результат выполнения (текстовое описание выполненных действий, команды и результат их выполнения в виде скриншотов) основных заданий ПР №6;
5. Листинги с комментариями и результат выполнения работы в виде скриншотов (например, сгенерированных файлов) из основной части ПР №7;
6. Листинги и полученные графики, диаграммы и т.д. из основной части ПР №8;
7. Отчет о проекте – результаты выполнения всех дополнительных заданий ПР №5-8;
8. Выводы о проделанной работе.

Отчет по практическим работам необходимо загрузить в СДО (в случае каких-либо технических проблем отчет необходимо выслать на почту преподавателя в домене *mirea.ru*)

Литература для изучения:

1. Документация на чемодан: https://wirenboard.com/wiki/Wb-demo-kit_v.2
2. Веб-интерфейс WirenBoard: https://wirenboard.com/wiki/Wiren_Board_Web_Interface
3. Утилита для извлечения исторических данных из внутренней базы данных: https://wirenboard.com/wiki/Wb-mqtt-db-cli
4. Протокол MQTT: https://en.wikipedia.org/wiki/MQTT
5. Описание улиты mosquito_sub: http://mosquitto.org/man/mosquitto_sub-1.html
6. Описание улиты mosquito_pub: https://mosquitto.org/man/mosquitto_pub-1.html
7. Описание протокола MQTT: https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/, https://habr.com/ru/post/463669/
8. Подключение к контроллеру по SHH: https://wirenboard.com/wiki/SSH
9. Визуализация: https://tableau.pro/m11
10. Графики: https://tableau.pro/m16
11. Гистограммы: https://tableau.pro/m19
12. Круговые диаграммы: https://tableau.pro/m23
13. JSON: https://ru.wikipedia.org/wiki/JSON, https://habr.com/ru/post/554274/
14. XML: https://ru.wikipedia.org/wiki/XML, https://code.makery.ch/ru/library/javafx-tutorial/part5/ , https://habr.com/ru/post/524288/
15. Paho MQTT: https://www.emqx.com/en/blog/how-to-use-mqtt-in-python