



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практическим работам №5-8

по дисциплине «Технологические основы Интернета вещей»

Выполнили:

Студенты группы ИКБО-11-22

Голованев Никита Алексеевич
Гришин Андрей Валерьевич
Андрусенко Лада Дмитриевна
Егоров Илья Эдуардович

Проверил:

Кандидат технических наук, доцент

Жматов Дмитрий Владимирович

2024 г.

Оглавление

Практическая работа №5	3
Практическая работа №6	7
Практическая работа №7	12
Практическая работа №8	21
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	28

Практическая работа №5

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.

Часть 1. Датчики и устройства для wireboard 6

1. Освещенность в составе устройства WB-MSW v.3 (5)

- **Название:** Датчик освещенности в составе WB-MSW v.3
- **Тип измерения:** Комбинированный
- **Измеряемый параметр и диапазон:** Освещенность, от 0 до 65 535 люкс
- **Точность:** $\pm 3\%$
- **Напряжение питания:** 5 В
- **Уникальный идентификатор в веб-интерфейсе:** sensor.illumination_wb_msw_v3_5
- **Протокол передачи данных:** Modbus RTU
- **Интерфейс управления:** RS-485 (Modbus)

- **Входы и выходы, схема подключения:** Датчик подключается к шине RS-485 и передает данные об освещенности в цифровом виде через Modbus RTU. Требуется подключение по стандарту RS-485 с использованием линии A/B для обмена данными и общей земли.

2. Влажность в составе устройства WB-MS v.2 (12)

- **Название:** Датчик влажности в составе WB-MS v.2
- **Тип измерения:** Аналоговый
- **Измеряемый параметр и диапазон:** Относительная влажность, от 0% до 100%
- **Точность:** $\pm 2\%$
- **Напряжение питания:** 5 В
- **Уникальный идентификатор в веб-интерфейсе:** sensor.humidity_wb_ms_v2_12
- **Протокол передачи данных:** Modbus RTU
- **Интерфейс управления:** RS-485 (Modbus)
- **Входы и выходы, схема подключения:** Датчик влажности передает данные в сеть через шину RS-485 по Modbus RTU, подключение производится через стандартную линию A/B для передачи сигнала и общую землю.

3. Преобразователь 1-Wire — Modbus RTU WB-M1W2 (3)

- **Название:** Преобразователь 1-Wire — Modbus RTU WB-M1W2
- **Тип измерения:** Цифровой
- **Напряжение питания:** 12 В
- **Уникальный идентификатор в веб-интерфейсе:** converter.1w_modbus_rtu_wb_m1w2_3

- **Протокол передачи данных:** Modbus RTU, 1-Wire
- **Интерфейс управления:** RS-485 (Modbus), 1-Wire
- **Входы и выходы, схема подключения:** Подключение 1-Wire датчиков происходит через интерфейс преобразователя, данные конвертируются в Modbus RTU и передаются на шину RS-485. Входы: контакты для подключения 1-Wire устройств. Выходы: стандартная шина RS-485 для передачи данных в Modbus сети.

Часть 2. Протоколы работы с устройствами

1. Modbus RTU

- **Принцип работы:** Используется мастер-слейв архитектура, где один мастер управляет одним или несколькими слейвами, передавая данные в последовательном формате.
- **Преимущества:** Высокая надежность, хорошая совместимость с оборудованием, широкая поддержка.
- **Недостатки:** Ограничение по скорости, сложность с увеличением количества устройств.
- **Сфера применения:** Промышленная автоматизация, управление оборудованием.

2. 1-Wire

- **Принцип работы:** Однопроводная шина, где данные и питание идут по одному проводу, используется уникальный ID для связи с устройством.
- **Преимущества:** Экономия на кабелях, простота установки.
- **Недостатки:** Невысокая скорость передачи, ограниченное расстояние.
- **Сфера применения:** Датчики температуры, небольшие системы мониторинга.

3. I²C (Inter-Integrated Circuit)

- **Принцип работы:** Мульти-мастер/слейв интерфейс, где мастер управляет обменом данных по двухпроводной линии.
- **Преимущества:** Поддержка множества устройств, простота подключения, высокая скорость обмена на короткие расстояния.
- **Недостатки:** Ограниченное расстояние, уязвимость к помехам.
- **Сфера применения:** Сенсоры, микроконтроллеры, системы мониторинга.

4. CAN (Controller Area Network)

- **Принцип работы:** Протокол передачи сообщений с высокой скоростью, использующий дифференциальную шину для передачи данных.
- **Преимущества:** Высокая надежность и помехозащищенность, способность работать в реальном времени.
- **Недостатки:** Сложность конфигурации, более высокая стоимость.
- **Сфера применения:** Автомобильные сети, системы управления в транспорте, промышленные сети.

Практическая работа №6

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.

Часть 1. SSH-Подключение

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

1. Запустите PuTTY.
2. Выберите слева в поле Category ветку Session.

3. Введите IP-адрес станда
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 Ассерпт.
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

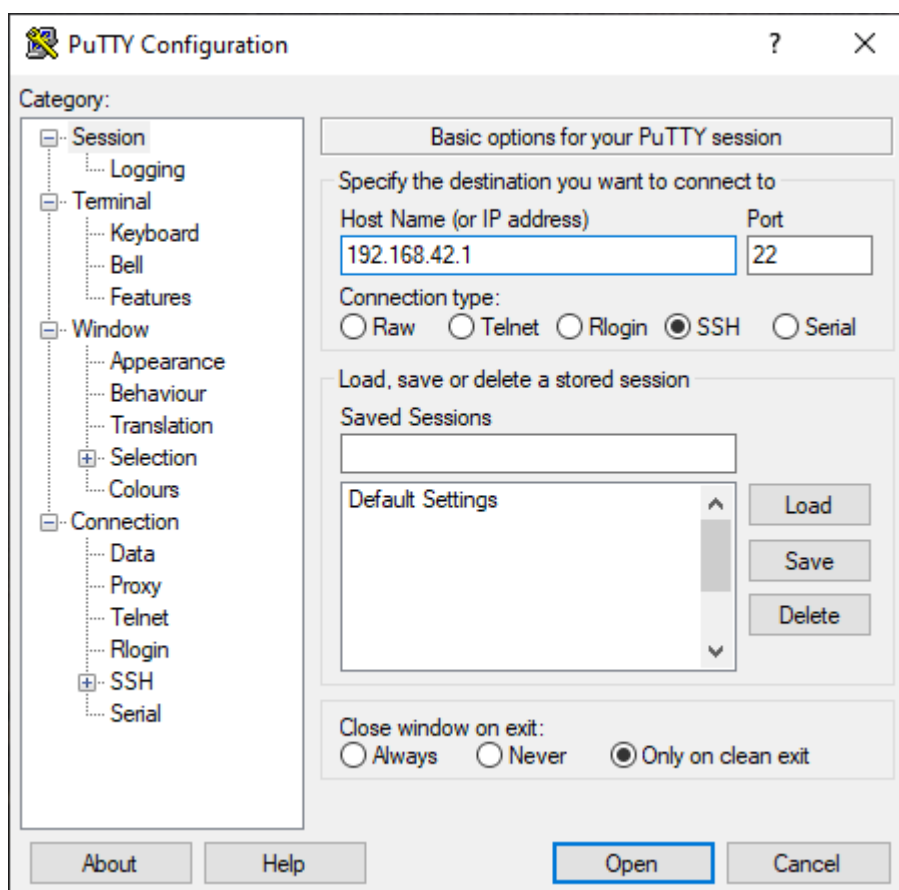


Рисунок 1 – Настройки соединения

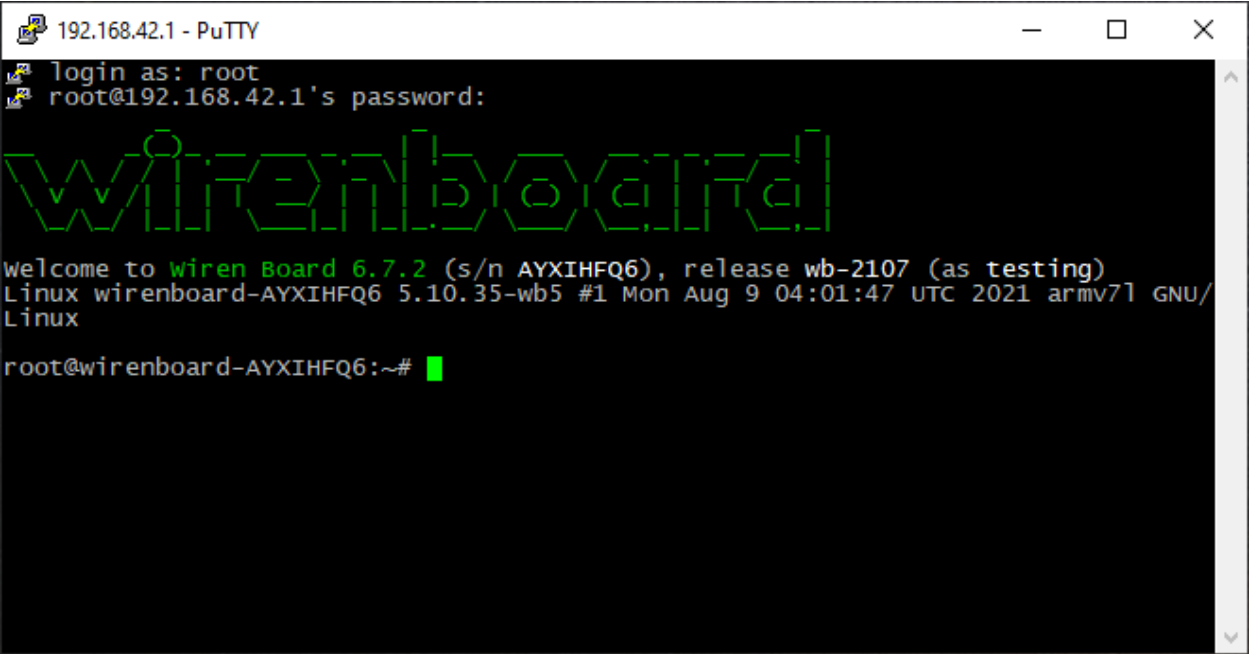


Рисунок 2 – Приветственный баннер контроллера

Часть 2. Подписка на топик

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

№ Варианта	Датчики
5	1. Датчик влажности устройства WB-MS v.2 (12) 2. Кнопка 29

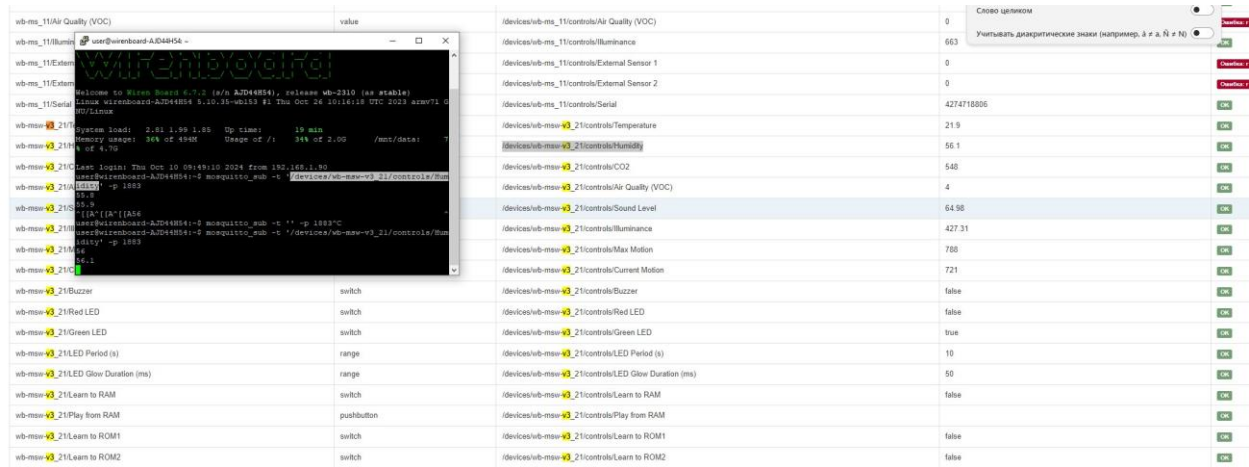


Рисунок 3 – Подписка на первый датчик

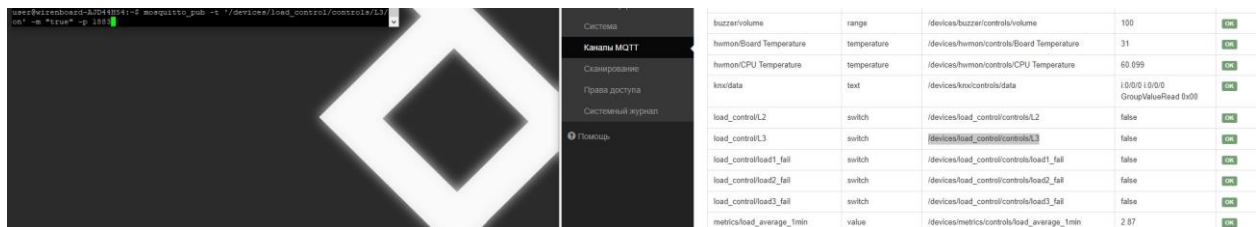


Рисунок 5 – Состояние первого датчика до переключения

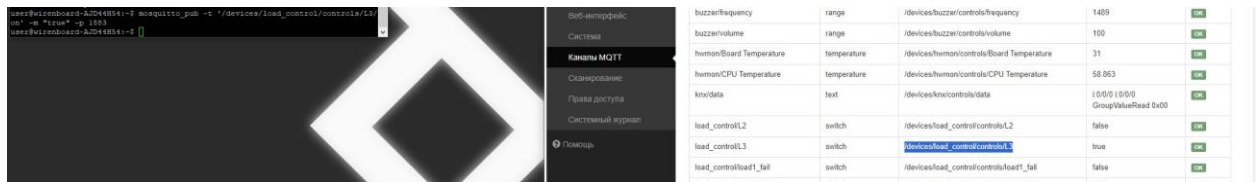


Рисунок 6 – Состояние первого датчика после переключения

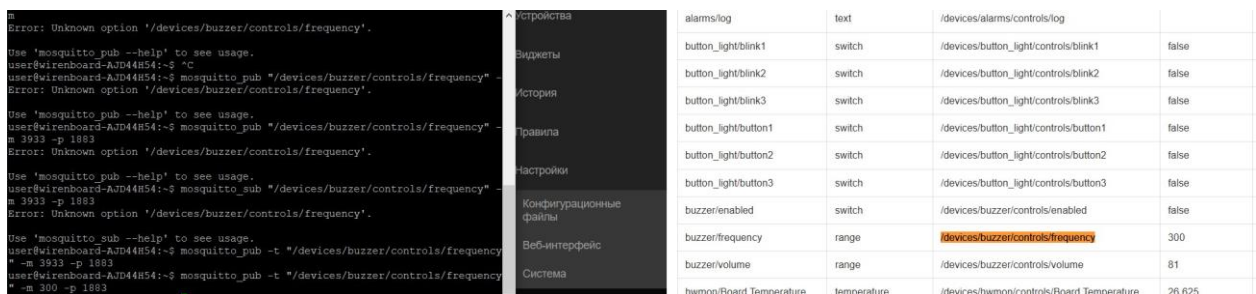


Рисунок 7 – Изменение уровня громкости

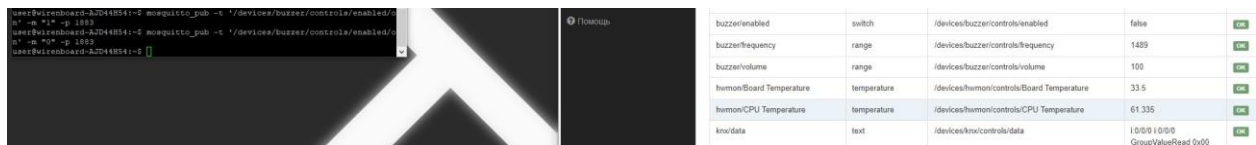


Рисунок 8 – Оповещение звукового сигнала

Практическая работа №7

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.

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.

Задание

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

№ Варианта	Датчики
5	1. Датчик CO2 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)

C:\Windows\System32\cmd.exe - python paho_test commented.py

```
/devices/wb-ms-v3_21/controls/Illuminance 1106
/devices/wb-ms-v3_21/controls/Sound Level 63.1
/devices/wb-ms-v3_21/controls/Illuminance 1105
/devices/wb-ms-v3_21/controls/Sound Level 60.6
/devices/wb-ms-v3_21/controls/Temperature 21.8
/devices/wb-ms-v3_21/controls/Sound Level 61.85
/devices/wb-ms-v3_21/controls/CO2 722
/devices/wb-ms-v3_21/controls/Illuminance 1099
/devices/wb-ms-v3_21/controls/Sound Level 64.2
/devices/wb-ms-v3_21/controls/Illuminance 1070
/devices/wb-ms-v3_21/controls/Sound Level 61.51
/devices/wb-ms-v3_21/controls/Illuminance 1062
/devices/wb-ms-v3_21/controls/Sound Level 69.38
/devices/wb-ms-v3_21/controls/Illuminance 1077
/devices/wb-ms-v3_21/controls/Sound Level 68
/devices/wb-ms-v3_21/controls/Illuminance 1131
/devices/wb-ms-v3_21/controls/Sound Level 68.7
/devices/wb-ms-v3_21/controls/CO2 723
/devices/wb-ms-v3_21/controls/Illuminance 1104
/devices/wb-ms-v3_21/controls/Sound Level 61.56
/devices/wb-ms-v3_21/controls/CO2 724
/devices/wb-ms-v3_21/controls/Illuminance 1176
/devices/wb-ms-v3_21/controls/Sound Level 48.95
/devices/wb-ms-v3_21/controls/Illuminance 1268
/devices/wb-ms-v3_21/controls/Sound Level 47.43
/devices/wb-ms-v3_21/controls/CO2 725
/devices/wb-ms-v3_21/controls/Illuminance 1288
/devices/wb-ms-v3_21/controls/Sound Level 61.31
/devices/wb-ms-v3_21/controls/CO2 726
/devices/wb-ms-v3_21/controls/Illuminance 1293
/devices/wb-ms-v3_21/controls/Sound Level 48.92
/devices/wb-ms-v3_21/controls/Illuminance 1246
/devices/wb-ms-v3_21/controls/Sound Level 57.94
/devices/wb-ms-v3_21/controls/CO2 727
/devices/wb-ms-v3_21/controls/Illuminance 1289
/devices/wb-ms-v3_21/controls/Sound Level 75.33
/devices/wb-ms-v3_21/controls/CO2 728
/devices/wb-ms-v3_21/controls/Illuminance 1286
/devices/wb-ms-v3_21/controls/Sound Level 66.82
/devices/wb-ms-v3_21/controls/Illuminance 1283
/devices/wb-ms-v3_21/controls/Sound Level 67.6
/devices/wb-ms-v3_21/controls/CO2 729
/devices/wb-ms-v3_21/controls/Illuminance 1291
/devices/wb-ms-v3_21/controls/Sound Level 64.34
/devices/wb-ms-v3_21/controls/CO2 730
/devices/wb-ms-v3_21/controls/Illuminance 1189
/devices/wb-ms-v3_21/controls/Sound Level 64.42
/devices/wb-ms-v3_21/controls/Illuminance 1187
/devices/wb-ms-v3_21/controls/Sound Level 73.29
/devices/wb-ms-v3_21/controls/CO2 731
/devices/wb-ms-v3_21/controls/Illuminance 1182
/devices/wb-ms-v3_21/controls/Sound Level 71.92
/devices/wb-ms-v3_21/controls/Illuminance 1173
/devices/wb-ms-v3_21/controls/Sound Level 70.52
/devices/wb-ms-v3_21/controls/Illuminance 1162
/devices/wb-ms-v3_21/controls/Sound Level 61.92
/devices/wb-ms-v3_21/controls/Illuminance 1129
/devices/wb-ms-v3_21/controls/Sound Level 70.26
/devices/wb-ms-v3_21/controls/Illuminance 1152
/devices/wb-ms-v3_21/controls/Sound Level 64.23
/devices/wb-ms-v3_21/controls/Illuminance 985
/devices/wb-ms-v3_21/controls/Sound Level 73.55
/devices/wb-ms-v3_21/controls/CO2 732
/devices/wb-ms-v3_21/controls/Illuminance 1173
/devices/wb-ms-v3_21/controls/Sound Level 64.62
/devices/wb-ms-v3_21/controls/Illuminance 1205
/devices/wb-ms-v3_21/controls/Sound Level 68.64
```

Рисунок 9 – Подписка на все датчики

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)

Листинг на языке Python

```
import paho.mqtt.client as mqtt
import json
from datetime import datetime
from time import sleep
import xml.etree.ElementTree as ET

# Параметры подключения к MQTT-брокеру
HOST = "192.168.1.16" # IP чемодана
PORT = 1883 # Стандартный порт подключения для Mosquitto
KEEPALIVE = 60 # Время ожидания доставки сообщения, если при отправке
оно будет превышено, брокер будет считаться недоступным

# Словарь с топиками и собираемыми из них параметрами
SUB_TOPICS = {
    '/devices/wb-msw-v3_21/controls/CO2': 'concentration',
```

```

        '/devices/wb-msw-v3_21/controls/Sound Level': 'sound_level',
        '/devices/wb-ms_11/controls/Illuminance': 'lux',
        '/devices/wb-msw-v3_21/controls/Temperature': 'temperature'
    }

JSON_LIST = []

# Создание словаря для хранения данных JSON
JSON_DICT = {}
for value in SUB_TOPICS.values():
    JSON_DICT[value] = 0

def on_connect(client, userdata, flags, rc):
    """ Функция, вызываемая при подключении к брокеру

    Arguments:
        client - Экземпляр класса Client, управляющий подключением к
        брокеру
        userdata - Приватные данные пользователя, передаваемые при
        подключении
        flags - Флаги ответа, возвращаемые брокером
        rc - Результат подключения, если 0, всё хорошо, в противном случае
        идем в документацию
    """
    print("Connected with result code " + str(rc))

    # Подключение ко всем заданным выше топикам
    for topic in SUB_TOPICS.keys():
        client.subscribe(topic)

def on_message(client, userdata, msg):
    """ Функция, вызываемая при получении сообщения от брокера по
    одному из отслеживаемых топику
    """

    Arguments:
        client - Экземпляр класса Client, управляющий подключением к
        брокеру

```

```
    userdata - Приватные данные пользователя, передаваемые при
подключении

    msg - Сообщение, приходящее от брокера, со всей информацией
    """

    payload = msg.payload.decode() # Основное значение, приходящее в
сообщение, например, показатель температуры

    topic = msg.topic # Топик, из которого пришло сообщение, поскольку
функция обрабатывает сообщения из всех топиков

    param_name = SUB_TOPICS[topic]
    JSON_DICT[param_name] = payload
    timenow = str(datetime.now())
    JSON_DICT['time'] = timenow
    JSON_DICT['num'] = HOST[-2:]

    JSON_LIST.append(JSON_DICT.copy())

    print(topic + " " + payload)

    # Запись данных в файл JSON
    with open('data.json', 'w') as file:
        json_string = json.dumps(JSON_DICT, indent=4) # Формирование
строки JSON из словаря
        file.write(json_string)

    # Запись данных в файл XML
    with open('data.xml', 'wb') as file:
        data = ET.Element('data')
        a = json.loads(json.dumps(JSON_DICT))
        for key in a:
            param = ET.SubElement(data, key)
            param.text = a.get(key)
        tree = ET.ElementTree(data)
        ET.indent(tree, space='\t', level=0)
        tree.write(file)

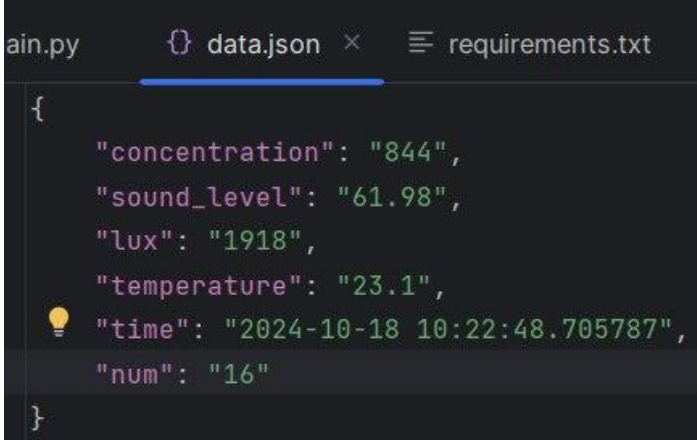
    sleep(5)
```



```
def main():
    # Создание и настройка экземпляра класса Client для подключения в
    Mosquitto
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(HOST, PORT, KEEPALIVE)

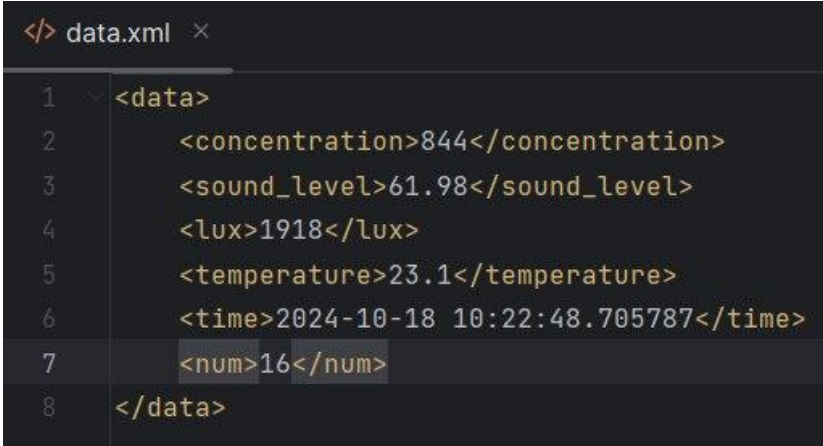
    client.loop_forever() # Бесконечный внутренний цикл клиента в
    ожидании сообщений

if __name__ == "__main__":
    main()
```



```
{
  "concentration": "844",
  "sound_level": "61.98",
  "lux": "1918",
  "temperature": "23.1",
  "time": "2024-10-18 10:22:48.705787",
  "num": "16"
}
```

Рисунок 10 – Сгенерированный файл в формате JSON



```
<?xml version="1.0" encoding="UTF-8" ?>
<data>
  <concentration>844</concentration>
  <sound_level>61.98</sound_level>
  <lux>1918</lux>
  <temperature>23.1</temperature>
  <time>2024-10-18 10:22:48.705787</time>
  <num>16</num>
</data>
```

Рисунок 11 – Сгенерированный файл в формате XML

На любом языке программирования реализуйте программу-парсер,

которая бы выводила в консоль данные, полученные из сгенерированных в п.2 файлов.

Структура получаемых данных

concentration: string

sound_level: string

lux: string

temperature: string

time: datetime

num: string

Листинг чтения файла

```
import json
import xml.etree.ElementTree as ET

with open('data.json', 'r') as f:
    json_data = json.load(f)

with open('data.xml', 'rb') as f:
    xml_data = ET.parse(f)

print('JSON:')
for data in json_data:
    print(f'{data}: {json_data[data]}')

print('\n\nXML:')
root = xml_data.getroot()
for data in root:
    print(f"{data.tag}: {data.text}")
```

```
JSON:
concentration: 844
sound_level: 61.98
lux: 1918
temperature: 23.1
time: 2024-10-18 10:22:48.705787
num: 16

XML:
concentration: 844
sound_level: 61.98
lux: 1918
temperature: 23.1
time: 2024-10-18 10:22:48.705787
num: 16
```

Рисунок 12 – Результат выполнения программы

Описание используемых библиотек

Название библиотеки	Описание библиотеки
paho.mqtt.client (как mqtt)	Основная библиотека для работы с протоколом MQTT (Message Queuing Telemetry Transport), предназначенного для обмена данными между устройствами через брокер. Позволяет создавать клиента MQTT для подключения, подписки и получения сообщений от брокера, а также для настройки функций, выполняющихся при подключении и получении сообщений.
json	Библиотека Python для работы с JSON (JavaScript Object Notation) — удобным форматом хранения и обмена данных в виде пар ключ-значение. Здесь

	используется для преобразования Python-словарей в строки JSON и записи их в файл.
<code>datetime</code>	Библиотека для работы с датой и временем в Python. В этом скрипте используется для получения текущего времени, чтобы добавлять его к собранным данным в качестве временной метки.
<code>time.sleep</code>	Функция из модуля <code>time</code> , которая приостанавливает выполнение программы на указанное количество секунд. В коде используется для ограничения частоты записи данных в файлы, чтобы не перегружать программу.
<code>xml.etree.ElementTree</code> (как ET)	Библиотека для создания, чтения и изменения XML-документов. Здесь она используется для преобразования данных в формат XML и записи их в файл.

Практическая работа №8

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.

Листинг сборки csv-файла

```
import paho.mqtt.client as mqtt

import csv

from datetime import datetime, timedelta

# Параметры подключения к MQTT-брокеру

HOST = "192.168.1.16" # IP чемодана

PORT = 1883 # Стандартный порт подключения для Mosquitto

KEEPALIVE = 60 # Время ожидания доставки сообщения, если при отправке
оно будет превышено, брокер будет считаться недоступным

# Словарь с топиками и собираемыми из них параметрами

SUB_TOPICS = {

    '/devices/wb-msw-v3_21/controls/CO2': 'concentration',

    '/devices/wb-ms_11/controls/Illuminance': 'lux',

    '/devices/power_status/controls/Vin': 'voltage'

}
```

```
tmp_dict = {value: '0' for value in SUB_TOPICS.values()}

count = 0

start_time = None


def on_connect(client, userdata, flags, rc):

    print("Connected with result code " + str(rc))

    for topic in SUB_TOPICS.keys():

        client.subscribe(topic)


def on_message(client, userdata, msg):

    global count, start_time

    if start_time is None:

        start_time = datetime.now()    # Фиксируем время начала сбора
данных

    current_time = datetime.now()

    # Проверяем, прошло ли 10 минут

    if current_time - start_time >= timedelta(minutes=1):

        print("10 минут прошло, завершение сбора данных.")

        client.disconnect()    # Отключаемся от MQTT-брокера

        return
```

```
count += 1

payload = msg.payload.decode() # Основное значение, приходящее в
сообщение, например, показатель температуры

topic = msg.topic # Топик, из которого пришло сообщение

param_name = SUB_TOPICS[topic]

tmp_dict[param_name] = payload

timenow = str(datetime.now())

tmp_dict['time'] = timenow

current_dict = tmp_dict.copy()

print(topic + " " + payload)

# Запись данных в CSV

with open('data.csv', mode='a', newline='') as file:

    writer = csv.writer(file)

    # Если это первая запись, пишем заголовок

    if count == 1:

        header = list(current_dict.keys())

        writer.writerow(header)

    # Запись данных строки

    row = list(current_dict.values())

    writer.writerow(row)
```



```
def main():

    # Создание и настройка экземпляра класса Client для подключения в
    Mosquitto

    client = mqtt.Client()

    client.on_connect = on_connect

    client.on_message = on_message

    client.connect(HOST, PORT, KEEPALIVE)

    client.loop_forever()    # Бесконечный внутренний цикл клиента в
    ожидании сообщений

if __name__ == "__main__":

    main()
```

Листинг генерации графиков

```
import matplotlib.pyplot as plt

import pandas as pd

data = pd.read_csv("data.csv")

concentration = data['concentration']

lux = data['lux']
```

```
voltage = data['voltage']

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.bar(range(len(concentration)), concentration)
plt.xlabel('Номер измерения')
plt.ylabel('concentration')
plt.title('CO2')

plt.subplot(1, 3, 2)
plt.plot(range(len(lux)), lux)
plt.ylim(300, 700)
plt.xlabel('Номер измерения')
plt.ylabel('lux')
plt.title('Освещенность')

plt.subplot(1, 3, 3)
plt.pie(voltage.value_counts(), labels=voltage.value_counts().index)
plt.title('Распределение напряжений')

plt.tight_layout()

plt.show()
```

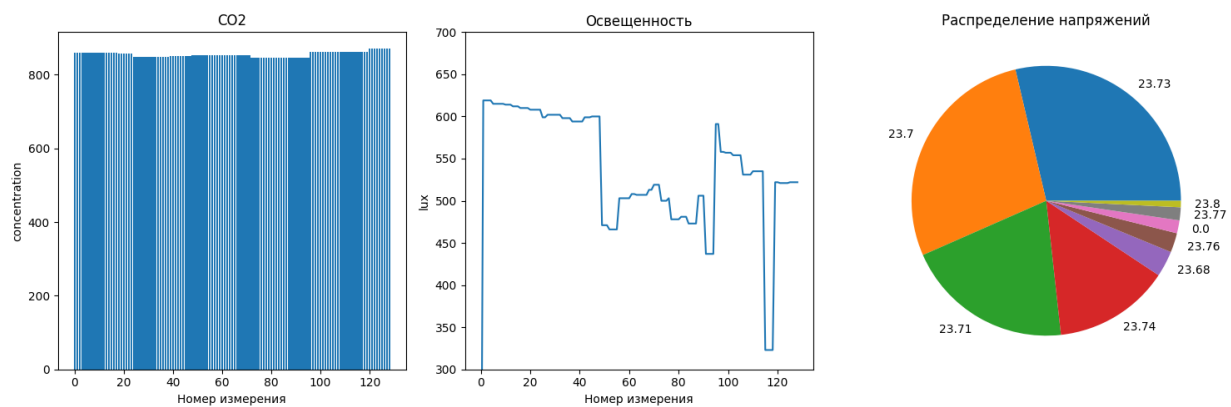


Рисунок 13 – Итоговые диаграммы

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

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 SSH: <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/javafxtutorial/part5/> ,
<https://habr.com/ru/post/524288/>
15. Paho MQTT: <https://www.emqx.com/en/blog/how-to-use-mqtt-in-python>