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-61-23 Филиппова Таисия Алексеевна
Астапович Ксения Александровна
Тагин Никита Вадимович

Проверил: Образцов Владимир Михайлович

2025 г.

СОДЕРЖАНИЕ

1. Цель работы

Изучить компоненты демонстрационного комплекта(датчики/устройства) и принципы работы с ними.

2. Ход выполнения работы

2.1 Практическая работа №5 - Измерительные и исполнительные устройства стенда

Часть 1. Измерительные и исполнительные устройства стенда

Опишите датчики и исполнительные устройства, согласно своему варианту для металлического чемодана:

Серый чемодан

Таблица 1 – задание варианта

№ Варианта	Измеряемая величина, датчики и устройства
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)

Таблица 2 – описание освещенности в составе устройства WB-MSW v.3 (5)

Параметр	Описание
1.Название датчика/устройства	Датчик освещенности с коррекцией под спектральную чувствительность человеческих глаз.
2.Тип измерения	Цифровой
3.Измеряемые параметры и диапазон	Освещенность, диапазон измерения от 0,02 до 100 000 люкс
4.Точность/погрешность	Погрешность до 20%
5. Напряжение питания	от 9 до 28 В постоянного тока
6.Уникальный идентификатор в веб интерфейсе	WB-MS v.2
7. Протокол передачи данных	Modbus RTU

8.Интерфейс управления (шина)	RS-485
9.Описание входов/выходов, схема подключения	Нет дискретных или аналоговых входов и выходов. Вход - это сенсоры устройств. Входы W1 и W2 устройства можно сконфигурировать как счётные входы. Выход - данные, передаваемые

Таблица 2 – описание влажности в составе устройства WB-MS v.2 (12)

Параметр	Описание
1.Название датчика/устройства	Датчик влажности
2.Тип измерения	Цифровой
3.Измеряемые параметры и диапазон	Влажность, от 0 до 98%
4.Точность/погрешность;	±3 %
5. Напряжение питания	от 9 до 28 В постоянного тока
6.Уникальный идентификатор в веб интерфейс	WB-MS v.2
7. Протокол передачи данных	Modbus RTU
8.Интерфейс управления (шина)	RS-485
9.Описание входов/выходов, схема подключения	Нет дискретных или аналоговых входов и выходов. Вход - это сенсоры устройств. Входы W1 и W2 устройства можно сконфигурировать как счётные входы. Выход - данные, передаваемые

Таблица 3 – описание преобразователя 1-Wire — Modbus RTU WB-M1W2 (3)

Параметр	Описание
1.Название датчика/устройства	WB-M1W2 v.3 преобразователь для термометров 1-Wire в RS-485
2.Тип измерения	Исполнительное устройство
3.Измеряемые параметры и диапазон	Управляемый параметр (напряжение на нагрузке)
4.Точность/погрешность;	-
5. Напряжение питания	9-28В постоянного тока
6.Уникальный идентификатор в веб интерфейсе	WB-M1W2 v.3
7. Протокол передачи данных	Modbus RTU
8.Интерфейс управления (шина)	RS-485
9.Описание входов/выходов, схема подключения	2 универсальных входа для датчиков 1-Wire или кнопок/концевиков, выход 5 В для питания датчиков

Черный чемодан

Таблица 4 – задание варианта

№ Варианта	Измеряемая величина, датчики и устройства
5	1. Трансформатор тока 5 (125) А, 9 мм WB-CT309 (TA1) 2. Импульсный счетчик расхода воды с имитацией потока (BR1) 3. Вентилятор (M2)

Таблица 5 – описание трансформатора тока 5 (125) А, 9 мм WB-CT309 (TA1)

Параметр	Описание
1.Название датчика/устройства	Трансформатора тока 5 (125) А, 9 мм *WB-CT309*
2.Тип измерения	Исполнительное устройство

3.Измеряемые параметры и диапазон	Управляемый параметр (напряжение на нагрузке)
4.Точность/погрешность;	-
5. Напряжение питания	5В
6.Уникальный идентификатор в веб интерфейсе	WB-CT309 v.2
7. Протокол передачи данных	Modbus RTU
8.Интерфейс управления (шина)	RS-485
9.Описание входов/выходов, схема подключения	Управление по Modbus RTU

Таблица 6 – описание импульсный счетчик расхода воды с имитацией потока (BR1)

Параметр	Описание
1.Название датчика/устройства	Импульсный счетчик расхода воды с имитацией потока
2.Тип измерения	
3.Измеряемые параметры и диапазон	Управляемый параметр (напряжение на нагрузке)
4.Точность/погрешность;	-
5. Напряжение питания	5В
6.Уникальный идентификатор в веб интерфейсе	BR1
7. Протокол передачи данных	Modbus RTU
8.Интерфейс управления (шина)	RS-48
9.Описание входов/выходов, схема подключения	

Таблица 7 – описание вентилятора (M2)

Параметр	Описание
1.Название датчика/устройства	Вентилятор M1
2.Тип измерения	Исполнительное устройство

3.Измеряемые параметры и диапазон	Управляемый параметр (напряжение на нагрузке)
4.Точность/погрешность;	-
5. Напряжение питания	5В
6.Уникальный идентификатор в веб интерфейсе	M1
7. Протокол передачи данных	Modbus RTU
8.Интерфейс управления (шина)	RS-485
9.Описание входов/выходов, схема подключения	Управление по Modbus RTU

Часть 2. Протоколы работы с устройствами

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

1. Modbus RTU;

Modbus — коммуникационный протокол, основан на архитектуре ведущий-ведомый (master-slave). Использует для передачи данных интерфейсы RS-485, RS-422, RS-232, а также Ethernet сети TCP/IP (протокол Modbus TCP).

Плюсы:

● Простота и распространённость: Легко внедряется, хорошо документирован, широко используется в промышленности.

● Много устройств: Поддерживает подключение большого числа устройств в одну сеть (до 247 устройств для Modbus RTU).

● Поддержка различных сред: Работает как на последовательных линиях (RS-232/RS-485), так и по TCP/IP через Ethernet.

● Надежность в промышленных условиях: Широко используется в промышленной автоматизации и SCADA системах.

Минусы:

● Ограниченная скорость: Протокол работает на относительно низких скоростях передачи данных.

● Ограниченная длина данных: Поддерживает передачу только до 256 байт в сообщении.

● Отсутствие встроенной защиты: Нет встроенных механизмов шифрования или аутентификации.

● Полудуплексный режим: RS-485 поддерживает только полудуплекс, что ограничивает одновременную передачу данных.

2. 1-Wire;

1-wire это интерфейс, позволяющий строить сети из устройств с топологией общая шина, один ведущий – много ведомых.

Плюсы:

● Экономичность: Для подключения требуется только один провод (плюс земля), что упрощает и удешевляет монтаж.

● Поддержка длинных линий связи: Может работать на расстоянии до 100 метров.

● Питание по шине: Некоторые устройства могут питаться прямо от сигнального провода.

● Малое энергопотребление: Подходит для приложений с низким энергопотреблением, таких как сенсоры температуры.

Минусы:

● Низкая скорость передачи данных: Скорость обмена очень ограничена (до 16.3 кбит/с), что не подходит для приложений с высокими требованиями к скорости.

● Ограниченное количество устройств: Хотя можно подключить много устройств, рост их количества приводит к увеличению сложности системы.

● Уязвимость к помехам: Из-за использования одного провода, система чувствительна к электромагнитным помехам.

3. I²C (IIC, англ. Inter-Integrated Circuit);

Inter-Integrated Circuit - последовательная шина обмена данными между интегральными схемами. Изобретена и в начале 80-х компанией Philips Semiconductor (теперь NXP), передача данных осуществляется по двум проводам в обе стороны.

Плюсы:

● Много ведущих-устройств и ведомых: Поддерживает несколько мастеров и до 127 рабов на одной шине.

● Двухпроводное соединение: Простота подключения — всего две линии

(SDA и SCL).

● Гибкость: Поддерживает как медленные, так и быстрые устройства, с динамическим регулированием скорости передачи данных. ● Широко используется: Популярность в интеграции с различными микроконтроллерами, сенсорами и модулями.

Минусы:

● Ограниченная длина шины: Максимальная длина линии ограничена (~1 метр) из-за ёмкости линии.

● Конфликты устройств: Нужна тщательная настройка адресов устройств, чтобы избежать конфликтов на шине.

● Ограничение скорости: Максимальная скорость обычно не превышает 3.4 Мбит/с (для высокоскоростных вариантов).

● Зависимость от емкости линии: Увеличение длины шины или количества подключённых устройств ухудшает производительность.

4. CAN.

CAN (англ. Controller Area Network — сеть контроллеров) — стандарт промышленной сети, ориентированный, прежде всего, на объединение в единую сеть различных исполнительных устройств и датчиков.

Плюсы:

● Высокая надежность: Очень устойчив к помехам, особенно в шумной среде (автомобили, промышленные объекты).

● Проверка ошибок: Встроенные механизмы проверки ошибок и исправления делают сеть очень надежной.

● Приоритетная передача сообщений: Приоритеты определяют, какое сообщение будет передано первым, что важно для систем реального времени.

● Много устройств: Поддерживает подключение до 110 устройств на одну шину без необходимости в сложных управляющих устройствах.

Минусы:

● Сложность протокола: Реализация может быть сложной, особенно в плане настройки фильтров и обработчиков ошибок.

● Ограниченная скорость передачи данных: Максимальная скорость 1 Мбит/с, что может быть недостаточно для некоторых приложений.

● Ограниченная длина линии: Хотя CAN может поддерживать линии длиной до 40 метров на максимальной скорости, при понижении скорости длину можно увеличить.

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

Часть 1. SHH-подключение

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

Рисунок 1 – ssh-подключение к консоли

Часть 2. Подписка на топик

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

Таблица 8 – задание варианта

№ Варианта	Датчики
5	1. Датчик влажности устройства WB-MS v.2 (12) 2. Кнопка SB1

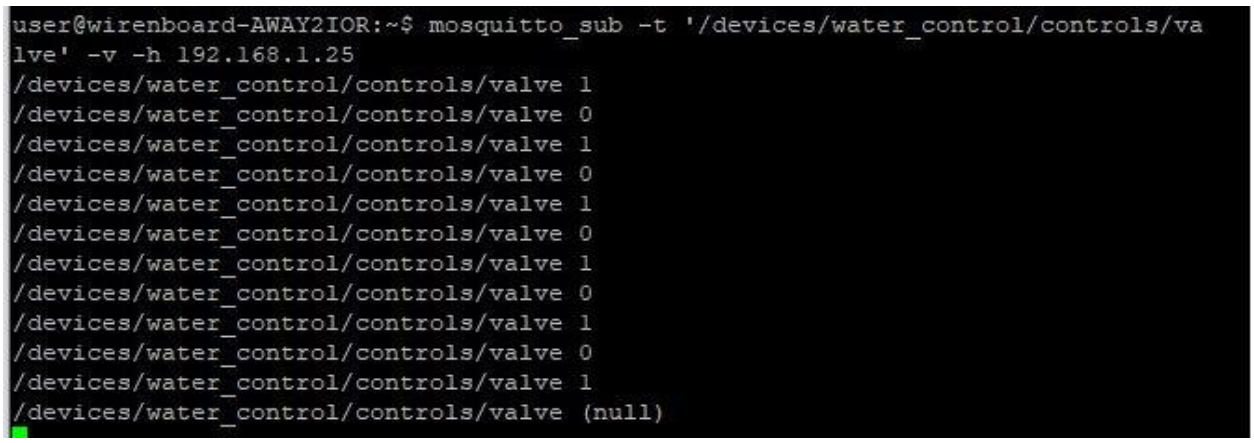

Рисунок 2 – подпись на сообщения кнопки SB1

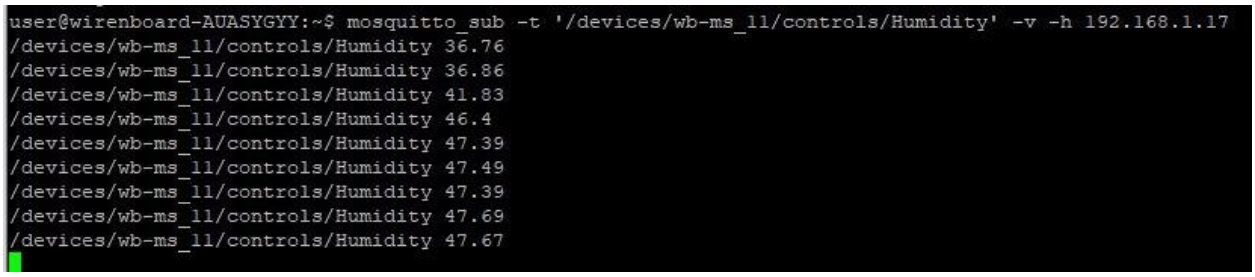

Рисунок 3 – подпись на сообщения датчика влажности устройства WB-MS v.2 (12)

Часть 3. Управление устройствами

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

Таблица 9 – задание варианта

№ Варианта	Датчики
5	1. Включите и измените уровень громкости звукового сигнала 2. Включение и выключение системы водоснабжения

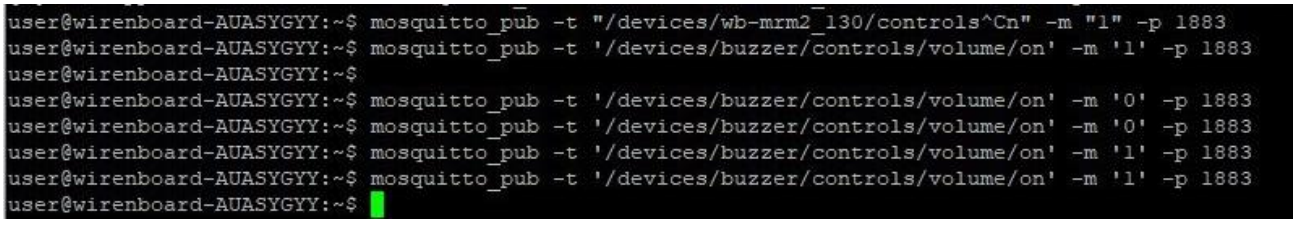

Рисунок 4 – включение датчика звукового сигнала, управление датчиком уровня громкости звукового сигнала

Рисунок 5 – управление включения и выключения системы водоснабжения

2.3 Практическая работа №7 - Форматы представления данных Синтаксическая совместимость

Часть 1(серый чемодан).

1. С компьютера в аудитории или личного устройства подпишитесь на несколько MQTT-топиков в составе стенда WB-demo-kit v.2, согласно вариантам.

Таблица 10 – задание варианта

№ 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)

Листинг 1 – подписка на MQTT топики

```
# Параметры подключения к MQTT-брокеру
HOST = "192.168.1.22" # 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-msw-v3_21/controls/Temperature':'temperature',
    '/devices/wb-msw-v3_21/controls/Illuminance': 'lux'
}
```

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.

Листинг 2 – скрипт для упаковки данных в json и вывода их в консоль

```
import paho.mqtt.client as mqtt
import json
from datetime import datetime


# Параметры подключения к MQTT-брокеру
HOST = "192.168.1.22" # 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-msw-v3_21/controls/Temperature':'temperature',
    '/devices/wb-msw-v3_21/controls/Illuminance': 'lux'
}

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 подключениии
    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 топиков

    Arguments:
    client - Экземпляр класса Client, управляющий подключением к брокеру
    userdata - Приватные данные пользователя, передаваемые при подключениии
    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
    JSON_DICT['time'] = str(datetime.now())

    JSON_LIST.append(JSON_DICT.copy())

    print(topic + " " + payload)

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

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 3 – часть содержимого файла json

```
[{"concentration": "1031", "sound_level": 0, "temperature": 0, "lux": 0, "time":
"2025-10-21 15:21:28.444493"},…
```

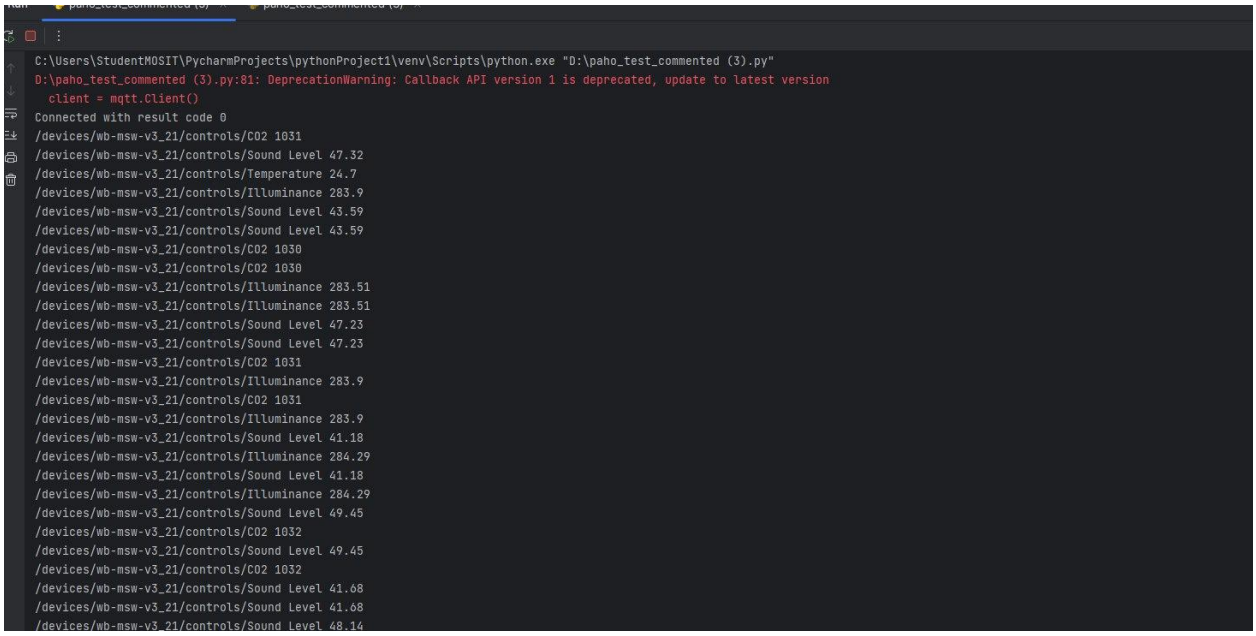

Рисунок 6 – вывод данных в консоль

Часть 2(черный чемодан).

С компьютера в аудитории или личного устройства подпишитесь на

несколько MQTT-топиков в составе стенда WB-demo-kit v.3, согласно вариантам, а далее выполните те же действия, что и в прошлой части.

Таблица 11 – задание варианта

№ Варианта	Датчики
5	1. Датчик CO2 устройства WB-MSW v.4 (A6) 2. Датчик освещенности устройства WB-MS v.4 (A6) 3. Напряжение на любом устройстве стенда

Листинг 4 – подписка на MQTT топики

```
# 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_64/controls/CO2': 'concentration',
    '/devices/wb-msw-v3_64/controls/Illuminance': 'lux',
    '/devices/battery/controls/Voltage': 'voltage'
}
```

Листинг 5 – скрипт для упаковки данных в json и вывода их в консоль

```
import paho.mqtt.client as mqtt
import json
from datetime import datetime


# 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_64/controls/CO2': 'concentration',
    '/devices/wb-msw-v3_64/controls/Illuminance': 'lux',
    '/devices/battery/controls/Voltage': 'voltage'
}

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 подключениии
    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 топиков

    Arguments:
    client - Экземпляр класса Client, управляющий подключением к брокеру
    userdata - Приватные данные пользователя, передаваемые при подключениии
    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
    JSON_DICT['time'] = str(datetime.now())

    JSON_LIST.append(JSON_DICT.copy())

    print(topic + " " + payload)

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

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()
```

```
C:\Users\StudentMOSIT\PycharmProjects\pythonProject1\venv\Scripts\python.exe "D:\paho_test_commented (3).py"
D:\paho_test_commented (3).py:73: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Connected with result code 0
/devices/wb-msw-v3_64/controls/CO2 1447
/devices/wb-msw-v3_64/controls/Illuminance 366.34
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 366.93
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 363.37
/devices/wb-msw-v3_64/controls/Illuminance 364.55
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 356.83
/devices/wb-msw-v3_64/controls/Illuminance 358.62
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 364.55
/devices/wb-msw-v3_64/controls/Illuminance 367.52
/devices/battery/controls/Voltage 4.134
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 366.34
/devices/wb-msw-v3_64/controls/Illuminance 364.55
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 363.37
/devices/wb-msw-v3_64/controls/Illuminance 362.77
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 360.99
/devices/wb-msw-v3_64/controls/CO2 1446
/devices/wb-msw-v3_64/controls/Illuminance 368.11
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 368.71
/devices/wb-msw-v3_64/controls/Illuminance 369.3
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 368.11
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 368.71
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 370.49
/devices/wb-msw-v3_64/controls/Illuminance 371.09
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 368.71
/devices/wb-msw-v3_64/controls/CO2 1445
/devices/wb-msw-v3_64/controls/Illuminance 371.68
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 375.83
/devices/battery/controls/Voltage 4.134
/devices/wb-msw-v3_64/controls/Illuminance 375.24
/devices/battery/controls/Voltage 4.134
```

Рисунок 7 – вывод данных в консоль

Листинг 6 – часть содержимого файла json

```
[{"concentration": "1447", "lux": 0, "voltage": 0, "time": "2025-10-21
15:10:35.195100"},
```

Краткое описание используемых сторонних библиотек

paho-mqtt

Библиотека для работы с MQTT-протоколом для обмена сообщениями между

устройствами.

Используемые функции:

- mqtt.Client() - создание MQTT-клиента
- client.on_connect - установка callback-функции при подключении к брокеру
- client.on_message - установка callback-функции при получении сообщения
- client.connect() - подключение к MQTT-брокеру
- client.subscribe() - подписка на MQTT-топики
- client.loop_forever() - запуск бесконечного цикла обработки сообщений

json

Стандартная библиотека для работы с JSON-форматом.

Используемые функции:

- json.dumps() - преобразование Python-объекта в JSON-строку

datetime

Стандартная библиотека для работы с датой и временем.

Используемые функции:

- datetime.now() - получение текущей даты и времени

Назначение в коде:

- **paho-mqtt** - для подключения к MQTT-брокеру и получения данных с датчиков
- **json** - для сохранения собранных данных в JSON-файл
- **datetime** - для добавления временных меток к полученным данным

2.4 Практическая работа №8

Часть 1(Серый чемодан).

Подпишитесь на несколько MQTT топиков в составе стенда WB-demo-kit v.2

1. 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).

Таблица 12 – задание варианта

№ Варианта	Датчики
5	1. Датчик CO2 устройства WB-MSW v.3 (5) 2. Датчик освещенности устройства WB-MS v.2 (12) 3. Напряжение на любом устройстве стенда

Листинг 7 – часть данных для подписки на MQTT топики

```
# Параметры подключения к MQTT-брокеру
HOST = "192.168.1.22" # 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-msw-v3_21/controls/Temperature':'temperature',
    '/devices/wb-msw-v3_21/controls/Illuminance': 'lux'
}
```

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).

Листинг 8 – скрипт для получения данных из json файла и преобразования их в график

```
from datetime import datetime
import matplotlib.pyplot as plt
import json
# Функция получения данных из json-файла
def get_data_from_json(filename):
    with open(filename, 'r') as file:
        file_data = json.load(file)

    return file_data
def create_plots(plots_data_lists):
    # Создание графиков для отрисовки данных
```

```
    fig, axs = plt.subplots(1, 2, figsize=(15,6)) # Получим окно с 1 колонкой и 2
столбцами графиков

    # fig - окно, в котором будут отрисовываться графики
    # axs содержит в себе список графиков для отрисовки на них значений

    # Задание набора точек для отрисовки
    # Первый аргумент - список значений по оси X, второй аргумент - по оси Y
    axs[0].plot(plots_data_lists['date'], plots_data_lists['motion'])

    # Задание лейблов для осей и графика
    axs[0].set_xlabel('Time')
    axs[0].set_ylabel('Motion level')
    axs[0].set_title('Motion')

    # Формирование гистограммы
    axs[1].hist(plots_data_lists['sound'])
    axs[1].set_xlabel('Sound level')
    axs[1].set_ylabel('Count')
    axs[1].set_title('Sound')

    return fig, axs


def main():
    plots_data_lists = {
        'motion': [],
        'sound': [],
        'date': [],
        'temperature': [],
        'power': []
    }

    json_data = get_data_from_json("data.json")

    # Заполнение списков с данными, с преобразованием типов
    for json_dict in json_data:
        plots_data_lists['motion'].append(int(json_dict.get('motion')))
        plots_data_lists['date'].append(datetime.fromisoformat(json_dict.get('dat
e')))
        plots_data_lists['sound'].append(float(json_dict.get('sound')))
        plots_data_lists['power'].append(float(json_dict.get('power')))
        plots_data_lists['temperature'].append(float(json_dict.get('temperature')
))

    fig, axs = create_plots(plots_data_lists)

    plt.show()


if __name__ == "__main__":
```

```
    main()
```

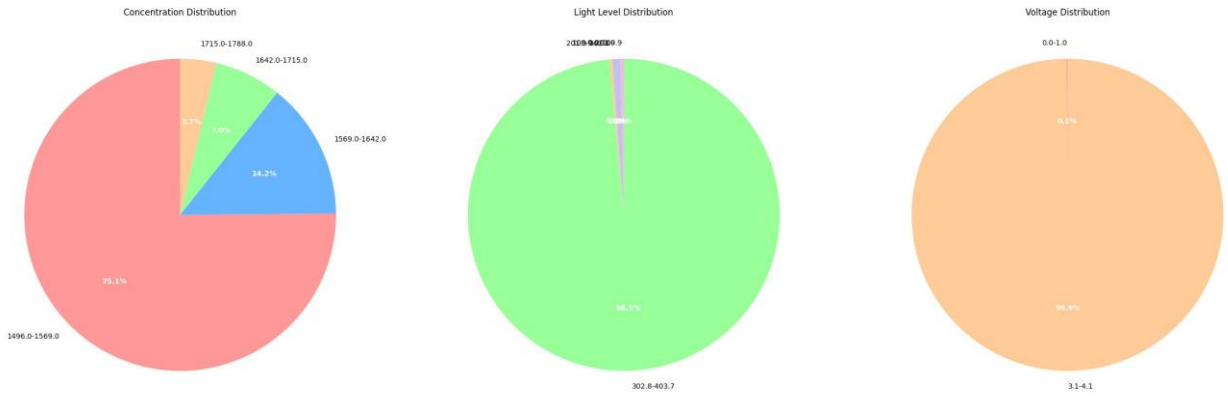

Рисунок 8 – круговые диаграммы

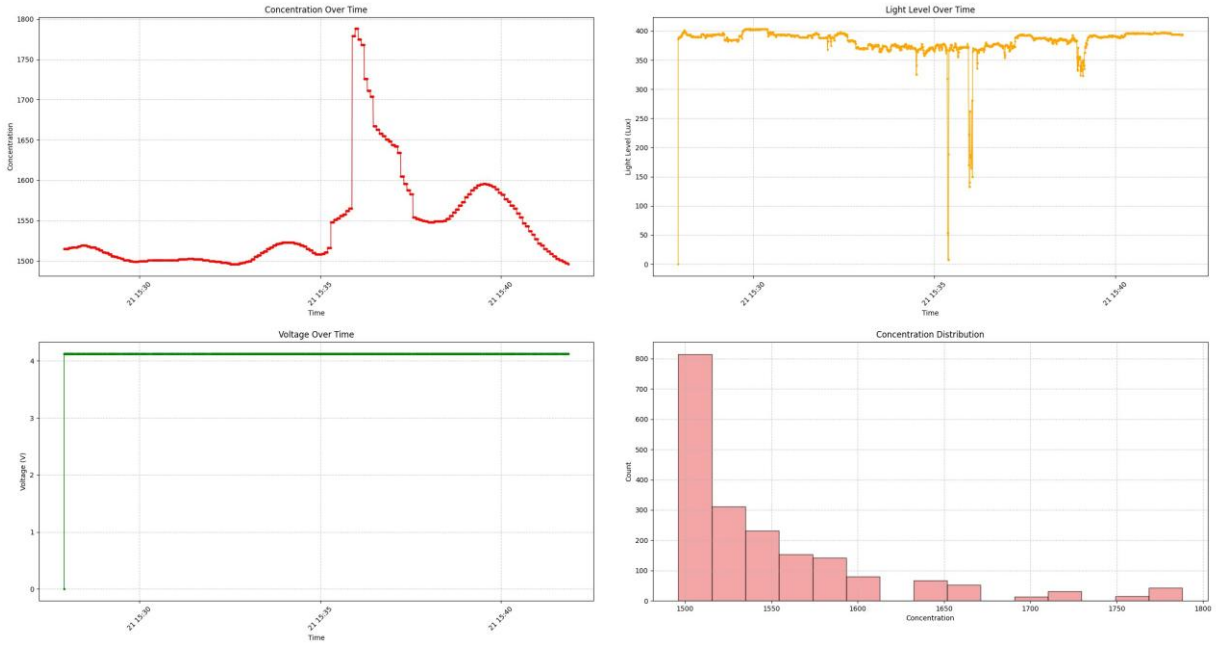

Рисунок 9 – столбиковые диаграммы

Часть 2(черный чемодан).

Подпишитесь на несколько MQTT топиков в составе стенда WBdemo-kit v.3

1. 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).

Таблица 7 – задание варианта

№ Варианта	Датчики
5	1. Датчик CO2 устройства WB-MSW v.3 (5)

	2. Датчик освещенности устройства WB-MS v.2 (12) 3. Напряжение на любом устройстве стенда

Листинг 9 – данные для подписка на MQTT топики

```
# 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_64/controls/CO2': 'concentration',
    '/devices/wb-msw-v3_64/controls/Illuminance': 'lux',
    '/devices/battery/controls/Voltage': 'voltage'
}
```

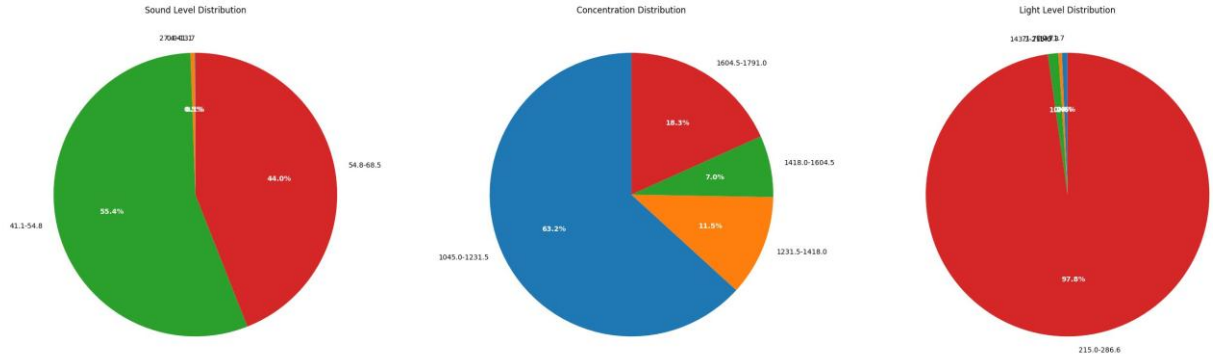

Рисунок 10 – круговые диаграммы

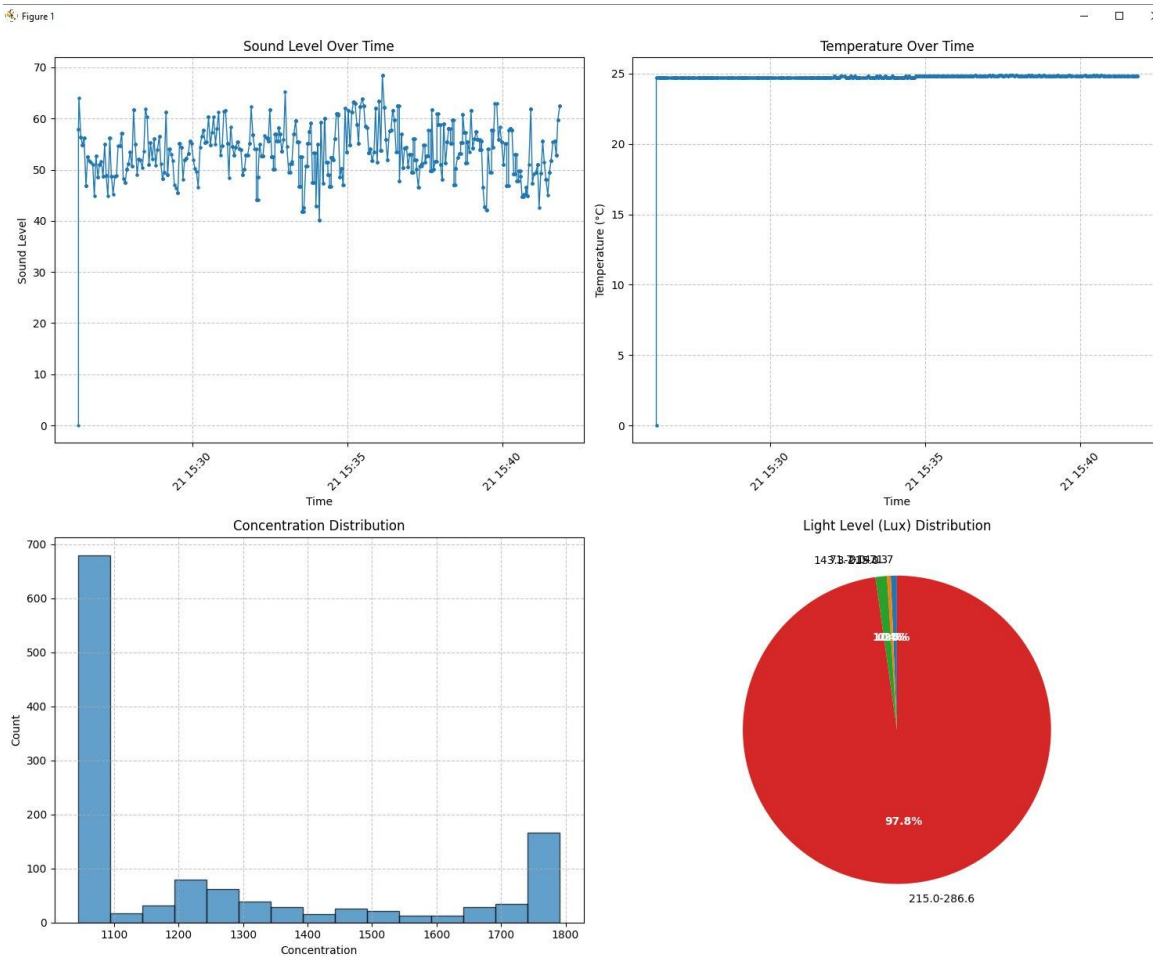

Рисунок 11 – столбиковые и круговые диаграммы

3. Вывод

В ходе работы мы познакомились с датчиками/устройствами и основными принципами работы с ними.

4. Список использованных источников

1. Технологические основы интернета вещей: Практикум : учебное пособие / А. Н. Миронов, Ю. А. Воронцов, А. В. Копылова, Е. К. Михайлова. — Москва: РТУ МИРЭА, 2022. — 147 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/239954 (дата обращения: 10.09.2025). — Режим доступа: для авториз. пользователей.

2. Документация на чемодан: https://wirenboard.com/wiki/Wb-demo kit_v.2

3. Веб-интерфейс https://wirenboard.com/wiki/Wiren_Board_Web_Interface

4. Полное описание https://github.com/wirenboard/wb-rules движка правил WirenBoard: WirenBoard: