|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 группыИКБО-20-21 | Квашнин Ю. В.  Хитров Н. С.  Мухаметшин А. Р.  Емельянов И. В. |
| **Проверил:** | Веремейчик Р. М. |

2023 г.

Оглавление

[Практическая работа 5 3](#_Toc148946366)

[Практическая работа 6 12](#_Toc148946367)

[Практическая работа 7 14](#_Toc148946368)

[Практическая работа 8 23](#_Toc148946369)

Практическая работа 5

**Задание на пятую практическую работу:**

Опишите датчики и исполнительные устройства, согласно своему варианту

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)

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

**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.

**Ход работы:**

**Относительная влажность в составе устройства WB-MSW v.3 (5)**

1. Название датчика/устройства - Настенный комбинированный

датчик WB-MSW v.3

2. Тип измерения (цифровой/аналоговый) - аналоговый

3. Измеряемые параметры и диапазон измерения - Измеряет относительную влажность. Диапазон измерения относительной влажности 5-95%

4. Точность - базовая погрешность при измерении относительной влажности +-3%

5. Напряжение питания - 9 В – 28 В постоянного тока

6. Использующийся протокол передачи данных - Modbus RTU

7. Интерфейс управления (шина) - RS-485

8. Описание входов и выходов, схема подключения

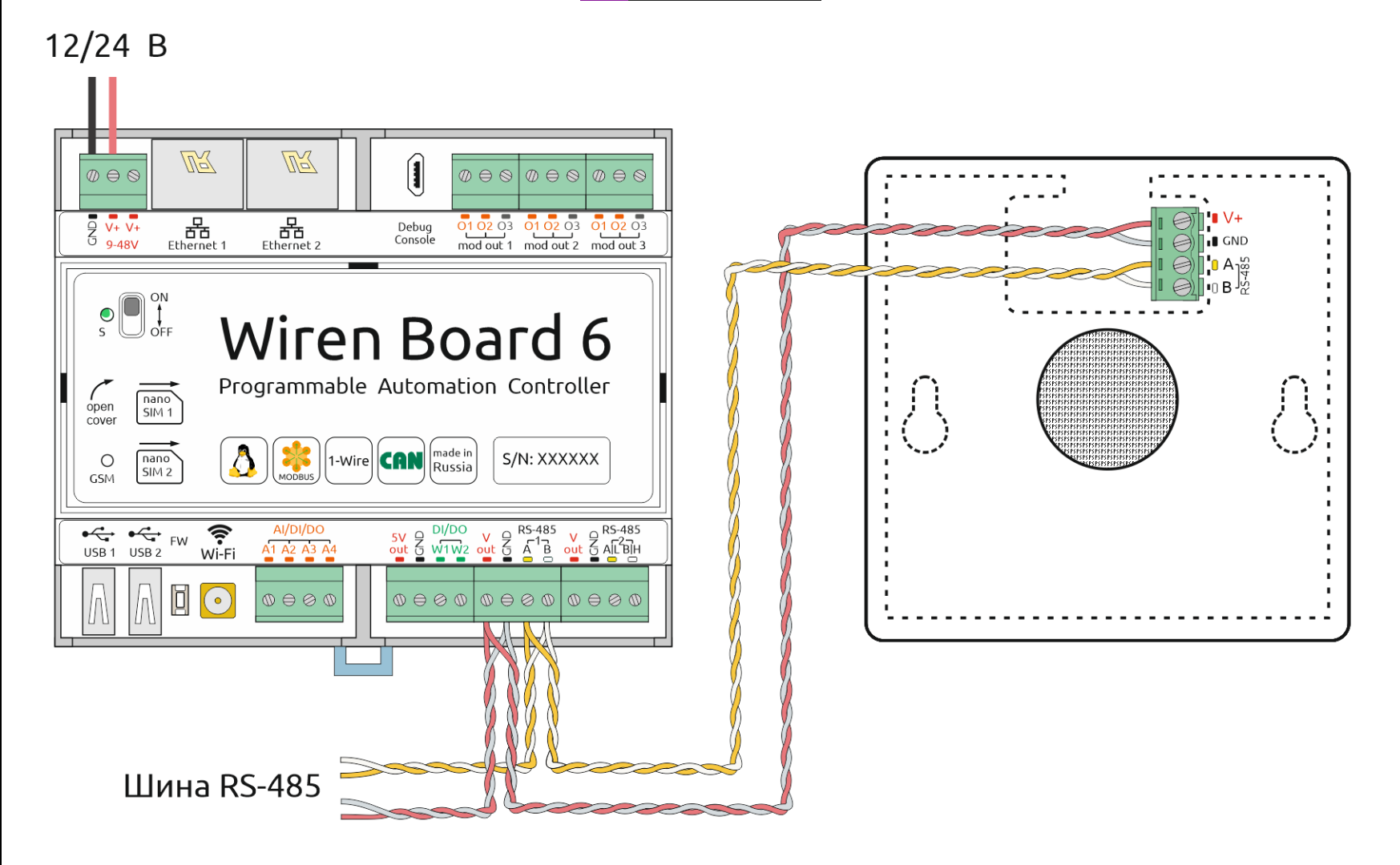


Рисунок 1 – Схема подключения настенного комбинированного датчика WB-MSW v.3

Клеммный блок «V+ GND A B» с шагом 3.5 мм служит для подключения питания и управления по шине RS-485

**Качество воздуха VOC в составе устройства WB-MS v.2 (12)**

1. Название датчика/устройства - Универсальный датчик WB-MS v.2

2. Тип измерения (цифровой/аналоговый) - аналоговый

3. Измеряемые параметры и диапазон измерения - измереяет качество воздуха VOC. Диапозон измерений 0 ppm - 60000 ppb (миллиардных долей)

4. Точность - +-15%

5. Напряжение питания - 9 – 28 В DC

6. Использующийся протокол передачи данных - Modbus RTU

7. Интерфейс управления (шина) - RS-485

8. Описание входов и выходов, схема подключения.

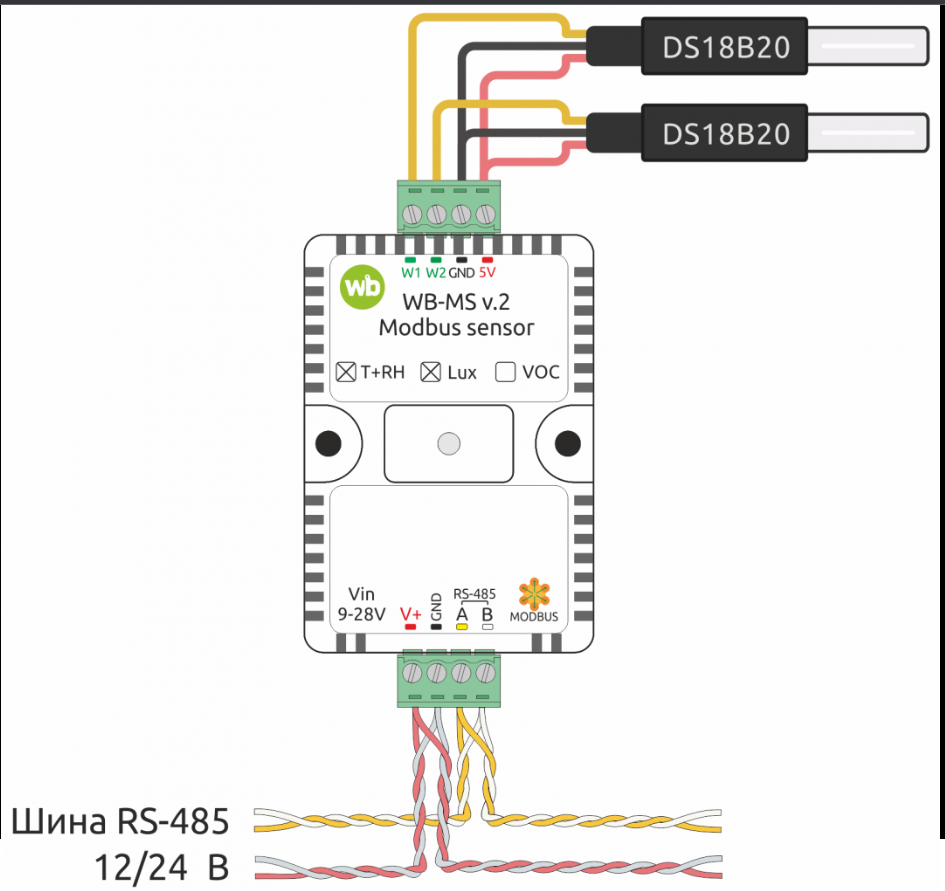


Рисунок 2 – Схема подключения универсального датчика WB-MS v.2

Клеммный блок «V+ GND A B» с шагом 3.5 мм служит для подключения питания и управления по шине RS-485. Верхний разъемный клеммник (W1, W2. GND и 5V) служит для подключения двух внешних цифровых датчиков температуры. Датчики температуры DS18B20.

**Модуль ввода-вывода WBIO-DO-R10A-8 (9)**

1. Название датчика/устройства - Модуль ввода-вывода WBIO-DO-R10A-8 (9)

2. Тип измерения (цифровой/аналоговый) - цифровой

3. Измеряемые параметры и диапазон измерения - коммутирует 8 выходов.

4. Точность -

5. Напряжение питания - Максимальное коммутируемое напряжение, переменный ток 240 В

6. Использующийся протокол передачи данных -

7. Интерфейс управления (шина) -

8. Описание входов и выходов, схема подключения

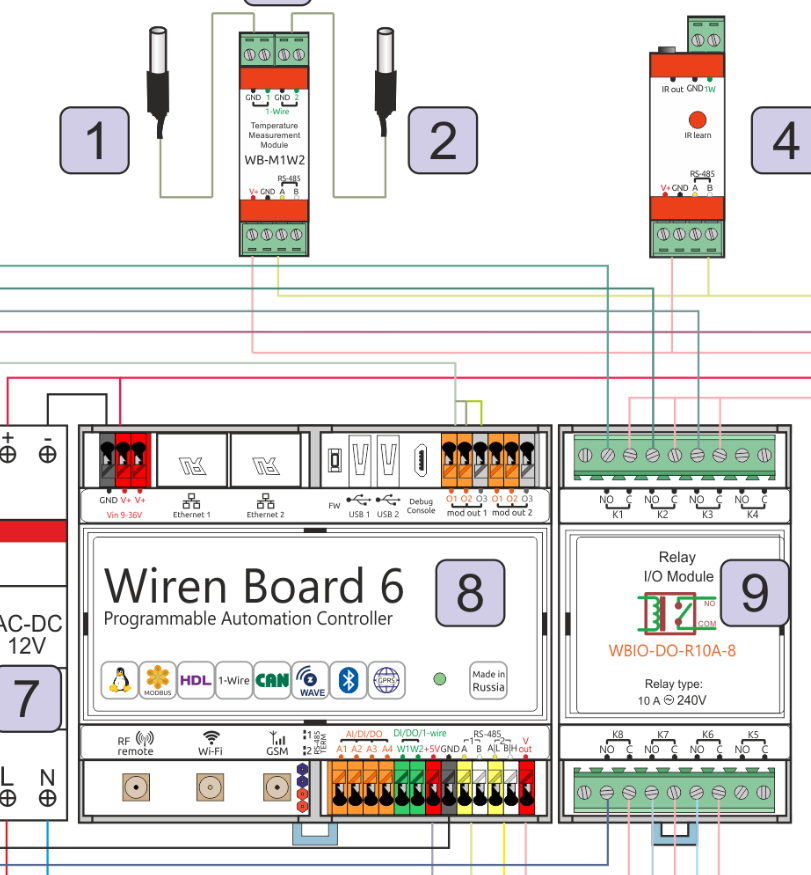


Рисунок 3

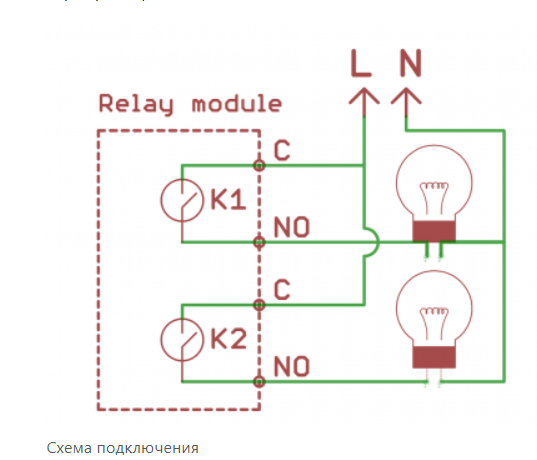


Рисунок 4

Содержит 8 реле реле HF32F-G с конфигурацией контактов SPST (два контакта на реле: NO/COM), коммутирующие соответствующие 8 выходов. Выходы выведены на винтовые зажимы. Последовательно можно подключать до 8 штук: до 4 модулей ввода (типа «I») и до 4-х модулей вывода (тип «O»). Исключение WBIO-AI-DV-12 — этот модуль можно подключить только один.

**1. Modbus RTU**

**Принцип работы**

Modbus RTU - это полудуплексный протокол связи, основанный на передаче по последовательному интерфейсу RS-485. Данные передаются в виде байтов, каждый из которых состоит из 8 битов. Протокол использует два провода для передачи данных и один провод для питания.

**Преимущества**

Простота реализации

Низкая стоимость оборудования

Широкая распространенность

**Недостатки**

Ограниченная скорость передачи данных (до 19,2 кбит/с)

Небезопасность

**Сфера применения**

Modbus RTU широко используется в системах автоматизации, где требуется обмен данными между устройствами, расположенными на относительно небольшом расстоянии друг от друга. Например, для подключения датчиков, исполнительных механизмов, контроллеров и других устройств.

**2. 1-Wire**

**Принцип работы**

1-Wire - это однопроводной протокол связи, предназначенный для подключения небольших, энергонезависимых устройств. Протокол использует один провод для передачи данных и питания.

**Преимущества**

Низкая стоимость оборудования

Простота реализации

Возможность подключения большого количества устройств

**Недостатки**

Ограниченная скорость передачи данных (до 125 кбит/с)

Небезопасность

**Сфера применения**

1-Wire используется для подключения датчиков температуры, влажности, движения и других устройств в системах автоматизации, а также в бытовой электронике.

**3. I²C**

**Принцип работы**

I²C - это двухпроводной протокол связи, предназначенный для подключения устройств в пределах одной платы или небольшого корпуса. Протокол использует два провода для передачи данных и управления.

**Преимущества**

Высокая скорость передачи данных (до 400 кбит/с)

Простота реализации

Безопасность

Недостатки

Ограниченное количество устройств на шине (до 127)

**Сфера применения**

I²C используется для подключения датчиков, исполнительных механизмов, микроконтроллеров и других устройств в системах автоматизации, а также в бытовой электронике.

**4. CAN**

**Принцип работы**

CAN - это последовательный протокол связи, предназначенный для подключения устройств в пределах одной платы или небольшого корпуса. Протокол использует два провода для передачи данных и управления.

**Преимущества**

Высокая скорость передачи данных (до 1 Мбит/с)

Безопасность

Возможность подключения большого количества устройств

**Недостатки**

Сложность реализации

Высокая стоимость оборудования

**Сфера применения**

CAN используется для подключения устройств в системах автоматизации, а также в автомобильной промышленности.

Практическая работа 6

1. **Подписка на топики**

Согласно варианту, было необходимо подписаться на сообщения следующих датчиков, а именно:

1. Датчик температуры 1-wire DS18B20 (2),
2. Кнопка 28.

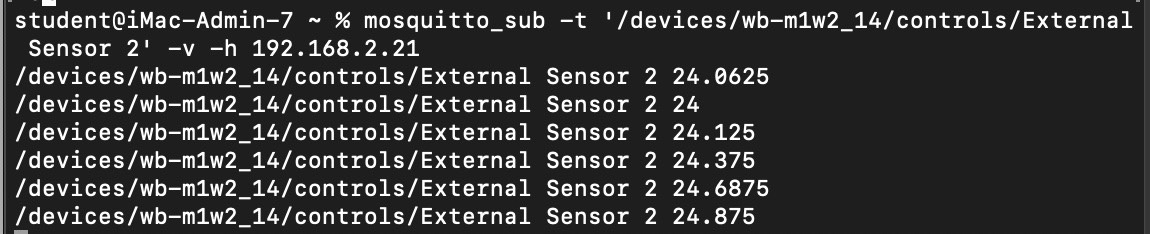


Рисунок 1 – Подписка на сообщения датчика температуры

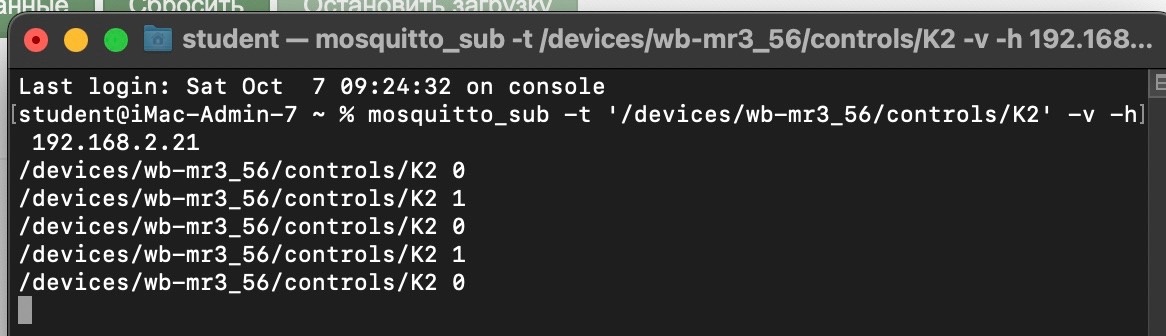


Рисунок 2 – Подписка на сообщения кнопки 28

1. **Управление устройствами**

Согласно варианту, было необходимо подписаться на сообщения следующих датчиков, а именно:

1. Включите индикатор 25
2. Включите и выключите RGB-ленту.

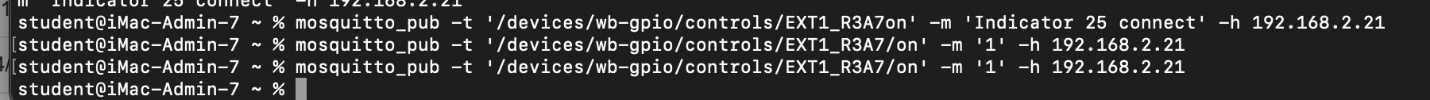


Рисунок 3 – Включение индикатора устройства

https://sun9-4.userapi.com/impg/3IAWQzaA-5uvEpKeW-3PxdxvK_yuG4_9cBgDHg/JRQq2Q1HSYc.jpg?size=1870x70&quality=95&sign=335e9db580bbbac60d0f24d6cb005ce1&type=album

Рисунок 4 – Включение подсветки кнопки

1. **Сообщения MQTT с внешнего устройства**

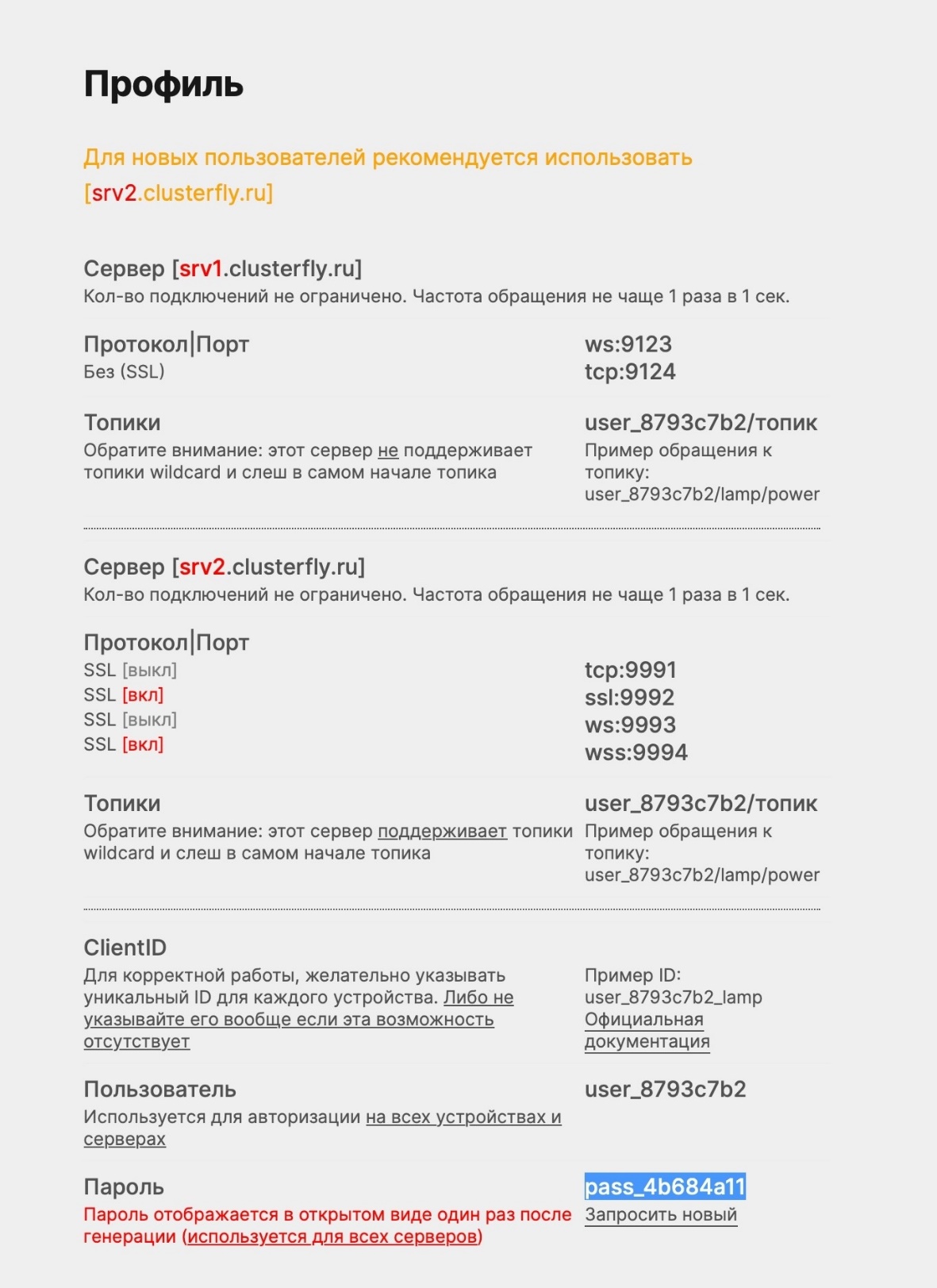


Рисунок 5 –брокер clusterfly

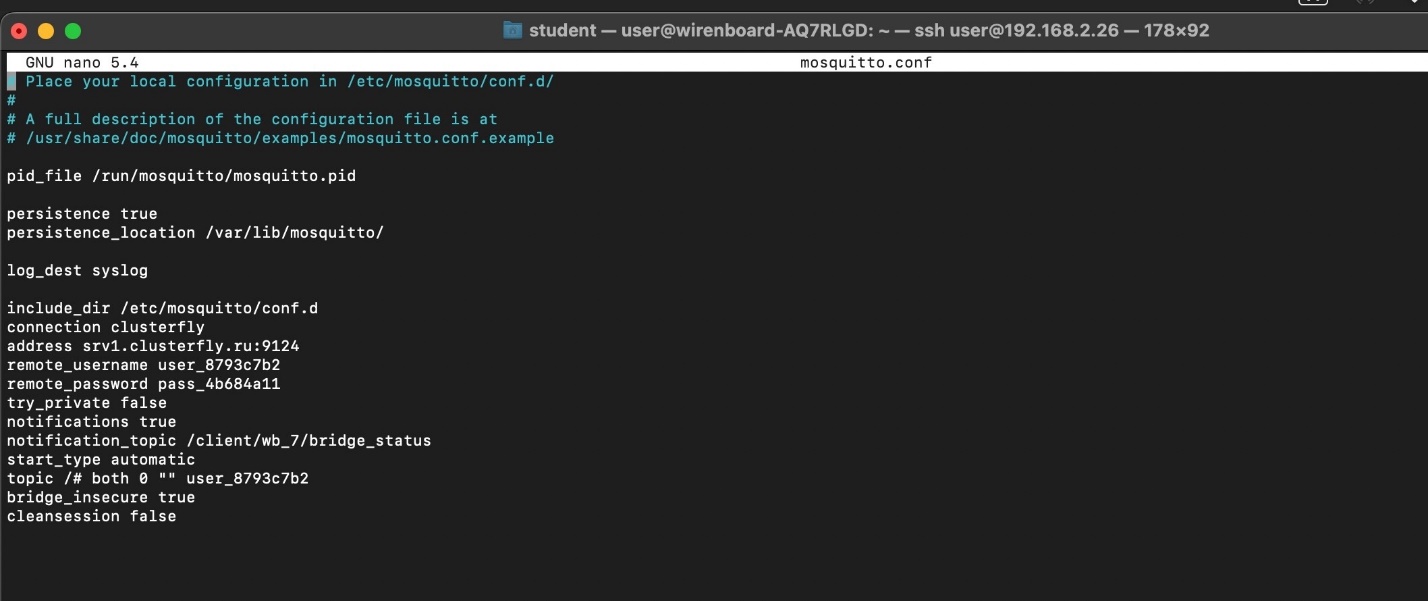


Рисунок 6 – mosquito.conf

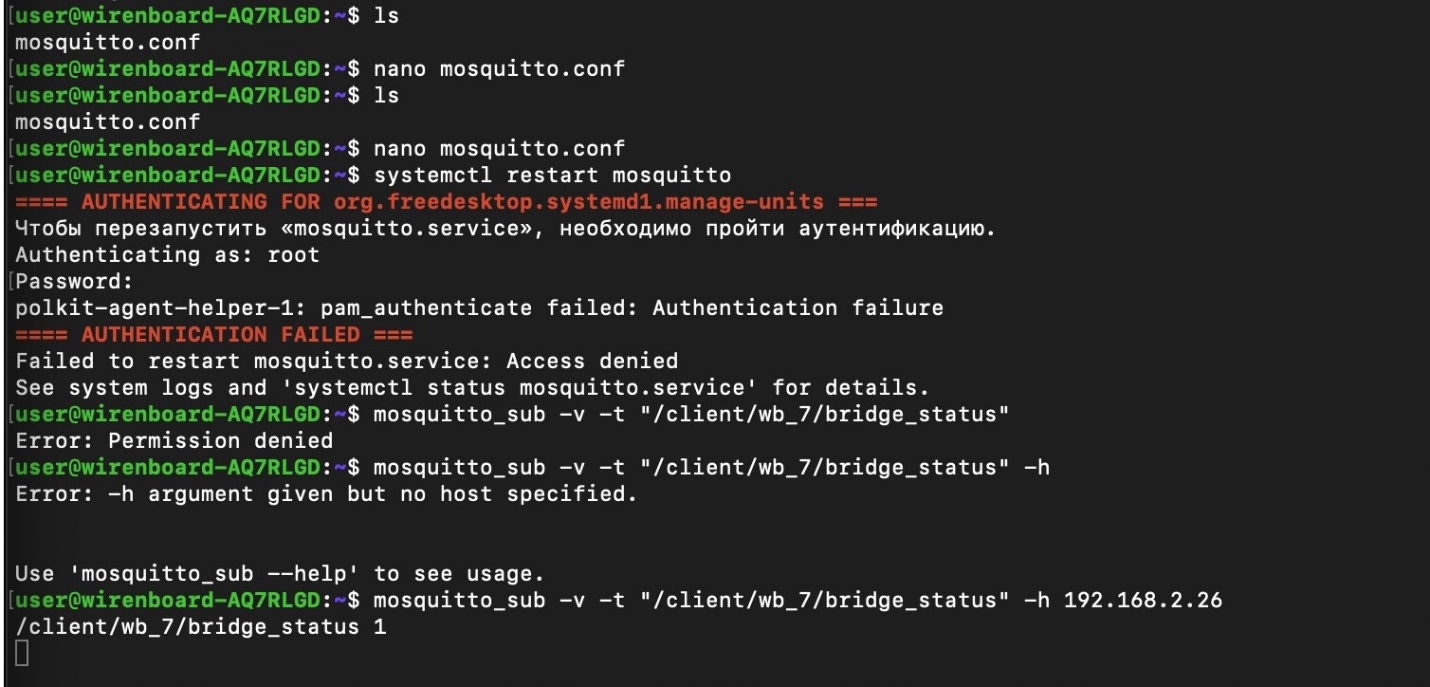


Рисунок 5 – Получение статуса настроенного моста

Практическая работа 7

С помощью программы-парсера подписаться на несколько топиков для отображения данных в консоли, а также их упаковки в JSON и XML файлы.

1. **Команды для подписки на топики**

Листинг 1 – Подписка на топики

SUB\_TOPICS = {

'/devices/wb-ms\_11/controls/Temperature': 'temperature',

'/devices/wb-msw-v3\_21/controls/Sound Level': 'sound',

'/devices/wb-msw-v3\_21/controls/CO2': 'concentration',

'/devices/wb-msw-v3\_21/controls/Air Quality (VOC)': 'VOC'

}

1. **Структура получаемых данных**

Листинг 2 – Структура получаемых данных

"temperature": float

"sound": float

"concentration”: number

"VOC": number

"time": date

"ip": string

1. **Листинги реализованных программ**
   1. **Программа для парсинга данных в формат JSON и XML**

Листинг 3 – Код программы для парсинга данных в формат JSON и XML

import time

import paho.mqtt.client as mqtt

import json

from datetime import datetime

HOST = "192.168.2.27"   
PORT = 1883   
KEEPALIVE = 60   
  
SUB\_TOPICS = {  
 '/devices/wb-ms\_11/controls/Temperature': 'temperature',  
 '/devices/wb-msw-v3\_21/controls/Sound Level': 'sound',  
 '/devices/wb-msw-v3\_21/controls/CO2': 'concentration',  
 '/devices/wb-msw-v3\_21/controls/Air Quality (VOC)': 'VOC'  
}

Листинг 4 – Продолжение Листинга 3, часть 1

HOST = "192.168.2.27"   
PORT = 1883   
KEEPALIVE = 60   
  
SUB\_TOPICS = {  
 '/devices/wb-ms\_11/controls/Temperature': 'temperature',  
 '/devices/wb-msw-v3\_21/controls/Sound Level': 'sound',  
 '/devices/wb-msw-v3\_21/controls/CO2': 'concentration',  
 '/devices/wb-msw-v3\_21/controls/Air Quality (VOC)': 'VOC'  
}  
  
JSON\_LIST = []  
  
JSON\_DICT = {}  
for value in SUB\_TOPICS.values():  
 JSON\_DICT[value] = 0  
  
def on\_connect(client, userdata, flags, rc):  
 print("Connected with result code " + str(rc))  
  
 for topic in SUB\_TOPICS.keys():  
 client.subscribe(topic)  
  
last\_check\_time = time.time()  
  
def on\_message(client, userdata, msg):  
 global last\_check\_time  
 payload = msg.payload.decode()   
 topic = msg.topic   
 param\_name = SUB\_TOPICS[topic]  
 JSON\_DICT[param\_name] = payload  
  
 if(time.time() - last\_check\_time >= 5):  
  
 JSON\_DICT['time'] = str(datetime.now())  
 JSON\_DICT['ip'] = '192.168.2.27'  
 JSON\_LIST.append(JSON\_DICT.copy())  
  
 print(JSON\_DICT)

Листинг 5 – Продолжение Листинга 3, часть 2

with open('data.json', 'w') as file:

json\_string = json.dumps(JSON\_LIST)

file.write(json\_string)

with open('data.xml', 'ab') as file1:

root = ET.Element('root')

tree = ET.ElementTree(root)

data = ET.SubElement(root, 'data')

temperature = ET.SubElement(data, 'temperature')

sound = ET.SubElement(data, 'sound')

concentration = ET.SubElement(data, 'concentration')

voc = ET.SubElement(data, 'VOC')

time1 = ET.SubElement(data, 'time')

ip = ET.SubElement(data, 'ip')

temperature.text = JSON\_DICT['temperature']

sound.text = JSON\_DICT['sound']

concentration.text = JSON\_DICT['concentration']

voc.text = JSON\_DICT['VOC']

time1.text = JSON\_DICT['time']

ip.text = JSON\_DICT['ip']

tree = ET.ElementTree(root)

tree.write(file1)

last\_check\_time = time.time()

def main():

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(HOST, PORT, KEEPALIVE)

client.loop\_forever()

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Результаты работы**

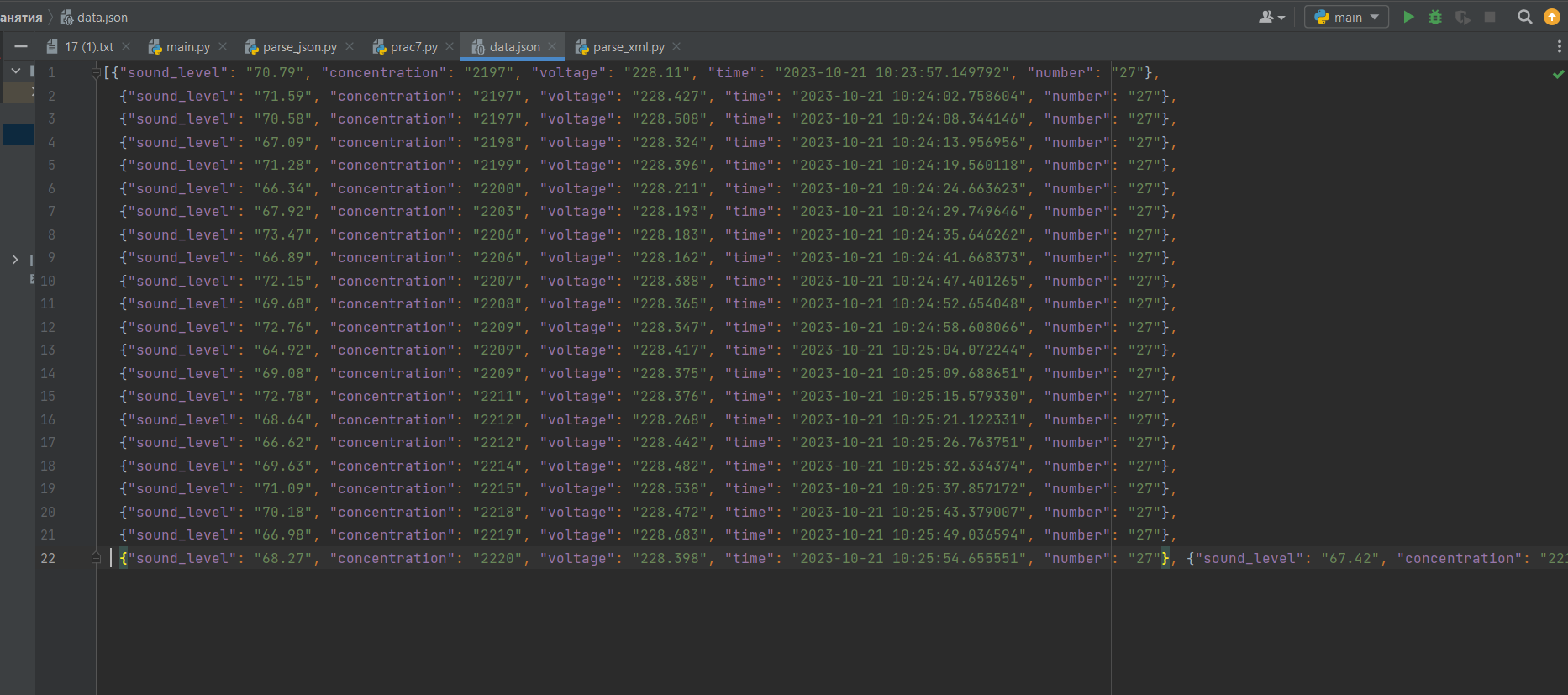


Рисунок 1 – Результат парсинга данных в формате JSON

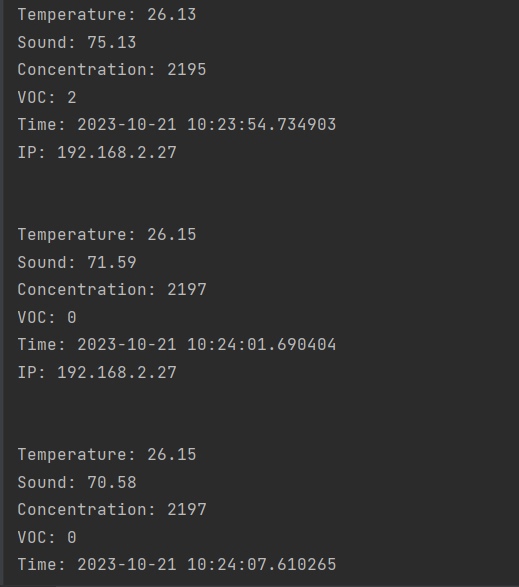


Рисунок 2 – Результат парсинга данных в формате XML

1. **Описание использованных библиотек**
2. paho.mqtt.client (MQTT библиотека):

Это библиотека для работы с протоколом MQTT (Message Queuing Telemetry Transport), который часто используется для передачи сообщений и данных между устройствами и серверами в интернете в реальном времени.

Используется для создания MQTT клиента, подключения к брокеру MQTT, публикации (отправки) сообщений и подписки на топики (каналы) для приема сообщений.

Примеры функций:

mqtt.Client(): Создает новый MQTT клиент.

connect(): Устанавливает соединение с брокером MQTT.

publish(): Публикует сообщение в указанный топик.

subscribe(): Подписывается на указанный топик для приема сообщений.

on\_connect(): Метод-обработчик, вызываемый при успешном соединении с брокером.

on\_message(): Метод-обработчик, вызываемый при получении нового сообщения.

1. xml.etree.ElementTree (XML библиотека):

Эта библиотека предоставляет инструменты для работы с XML-документами. Она позволяет парсить (анализировать) XML-документы и создавать их.

Используется для чтения и записи данных в формате XML.

Примеры функций:

ElementTree.Element(): Создает новый элемент XML.

ElementTree.SubElement(): Создает подэлемент внутри элемента.

ElementTree.parse(): Анализирует XML-документ из файла.

ElementTree.ElementTree(): Создает объект для работы с деревом XML.

element.text: Получает текстовое содержимое элемента.

element.attrib: Получает атрибуты элемента.

1. datetime (Дата и время):

Эта библиотека предоставляет функциональность для работы с датой и временем в Python. Используется для создания, форматирования и операций с датами и временем.

Примеры функций:

datetime.datetime.now(): Получает текущую дату и время.

datetime.datetime.strptime(): Преобразует строку в объект даты и времени по заданному формату.

datetime.timedelta(): Позволяет выполнять арифметические операции с датами и временем.

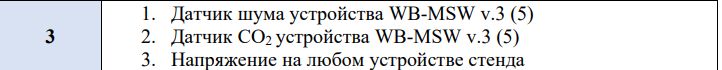
strftime(): Форматирует дату и время в строку в соответствии с заданным форматом.

date(): Получает только дату из объекта даты и времени.

time(): Получает только время из объекта даты и времени.

Практическая работа 8

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



1. **Напишите скрипты, позволяющие визуализировать статистические данные в виде диаграмм.**

Листинг1 – Написанный скрипт

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, 3, figsize=(15, 6))

# 2023-10-21 10:23:57.149792

# Линейный график для CO2

axs[0].plot(plots\_data\_lists['time'], plots\_data\_lists['concentration'])

axs[0].set\_xlabel('Time')

axs[0].set\_ylabel('CO2 level')

axs[0].set\_title('CO2')

# Формирование кругового графика для sound\_level

range1\_count = 0

range2\_count = 0

range3\_count = 0

range4\_count = 0

for sound\_level in plots\_data\_lists['sound\_level']:

if 45 <= sound\_level < 55:

range1\_count += 1

if 55 <= sound\_level < 65:

range2\_count += 1

if 65 <= sound\_level < 75:

range3\_count += 1

if 75 <= sound\_level < 85:

range4\_count += 1

total\_records = len(plots\_data\_lists)

percentage\_range\_1 = (range1\_count / total\_records) \* 100

percentage\_range\_2 = (range2\_count / total\_records) \* 100

percentage\_range\_3 = (range3\_count / total\_records) \* 100

percentage\_range\_4 = (range4\_count / total\_records) \* 100

percentages = [percentage\_range\_1, percentage\_range\_2, percentage\_range\_3, percentage\_range\_4]

labels = ['45-55', '55-65', '65-75', '75-85']

colors = plt.cm.Paired(range(len(labels)))

explode = (0.1, 0, 0, 0)

Листинг 2 – Продолжение Листинга 1

axs[2].pie(percentages, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

axs[2].set\_title('sound\_level')

axs[2].axis('equal')

# Формирование гистограммы для voltage

axs[1].hist(plots\_data\_lists['voltage'])

axs[1].set\_xlabel('voltage level')

axs[1].set\_ylabel('Count')

axs[1].set\_title('voltage')

return fig, axs

def main():

plots\_data\_lists = {

'concentration': [],

'sound\_level': [],

'time': [],

'voltage': [],

}

json\_data = get\_data\_from\_json("data.json")

for json\_dict in json\_data:

plots\_data\_lists['concentration'].append(int(json\_dict.get('concentration')))

plots\_data\_lists['time'].append(datetime.fromisoformat(json\_dict.get('time')))

plots\_data\_lists['sound\_level'].append(float(json\_dict.get('sound\_level')))

plots\_data\_lists['voltage'].append(float(json\_dict.get('voltage')))

print(datetime.fromisoformat(json\_dict.get('time')).time())

fig, axs = create\_plots(plots\_data\_lists)

plt.show()

main()

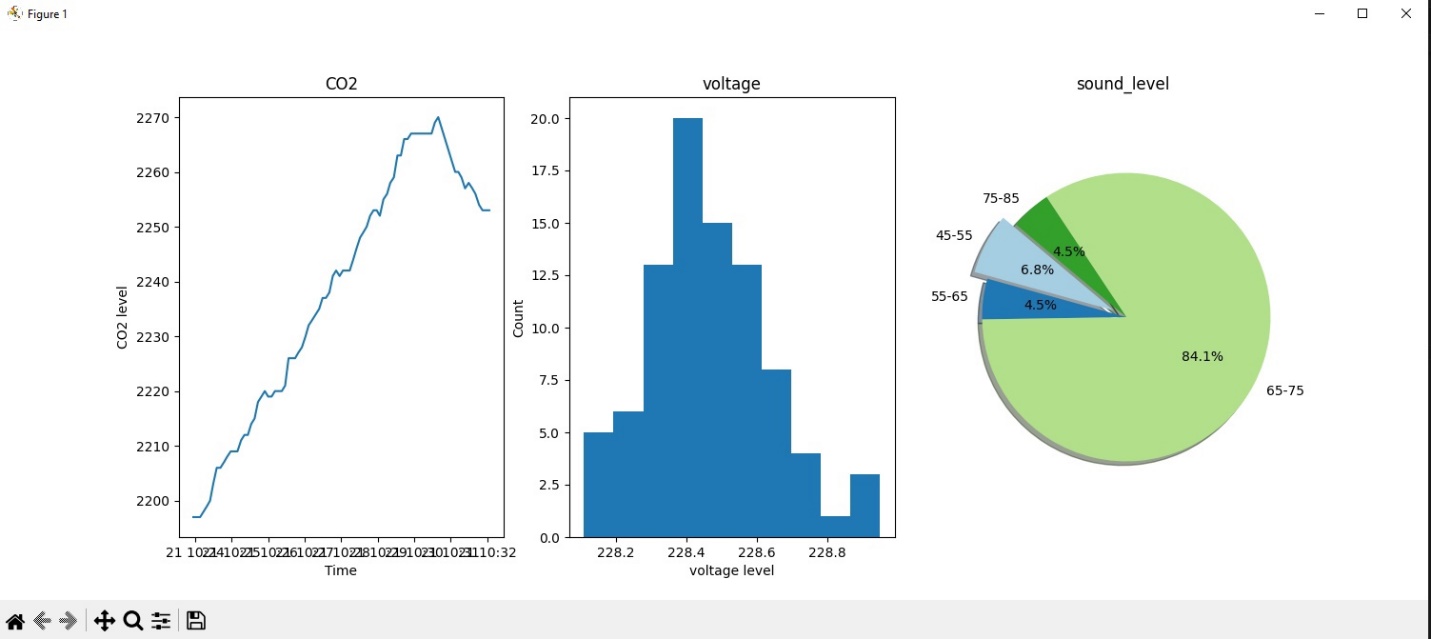


Рисунок 1 – Созданные диаграммы