|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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-24-21 | Ращиков К.А.  Новиков А.А.  Константинова С.В.  Конценберг И.  Шкред В.С. |
| **Проверил:** | Веремейчик Роман Максимович |

2023 г.

**Оглавление**

[1 ПРАКТИЧЕСКАЯ РАБОТА №5 3](#_Toc148953739)

[1.1 Описание задания 3](#_Toc148953740)

[1.2 Выполнение задания 3](#_Toc148953741)

[2 ПРАКТИЧЕСКАЯ РАБОТА №6 7](#_Toc148953742)

[2.1 Описание задания 7](#_Toc148953743)

[2.2 Выполнение задания 8](#_Toc148953744)

[3 ПРАКТИЧЕСКАЯ РАБОТА №7 12](#_Toc148953745)

[3.1 Описание задания 12](#_Toc148953746)

[3.2 Выполнение задания 12](#_Toc148953747)

[4 ПРАКТИЧЕСКАЯ РАБОТА №8 18](#_Toc148953748)

[4.1 Описание задания 18](#_Toc148953749)

[4.2 Выполнение задания 18](#_Toc148953750)

[5 ВЫВОД ПО ПРАКТИЧЕСКИМ РАБОТАМ 23](#_Toc148953751)

1. ПРАКТИЧЕСКАЯ РАБОТА №5
   1. Описание задания

Часть 1. Измерительные и исполнительные устройства стенда

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)

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.

* 1. Выполнение задания

1. Датчик №1
   1. Название датчика/устройства: Универсальный настенный датчик WB-MSW v.3
   2. Тип измерения: Цифровой
   3. Измеряемые параметры и диапазон измерения:
   4. Относительная влажность: 5% до 95%
   5. Точность: ±3% для относительной влажности
   6. Напряжение питания: 9-28 В
   7. Уникальный идентификатор датчика в веб-интерфейсе: wb-msw-v3\_21/Humidity
   8. Использующийся протокол передачи данных: Modbus RTU, адрес задается программно, заводские настройки указаны на наклейке
   9. Интерфейс управления (шина): RS-485
   10. Описание входов и выходов, схема подключения: Этот датчик предназначен для измерения температуры и относительной влажности внутри помещения. Его можно подключить к контроллеру через интерфейс RS-485.
2. Датчик №2
   1. Название датчика/устройства: WB-MS v.2
   2. Тип измерения: Цифровой
   3. Измеряемые параметры и диапазон измерения:
   4. Качество воздуха VOC (органические летучие соединения): 0 - 60000 ppm.
   5. Точность: ±15 % (тип) ±40 % (макс) погрешность
   6. Напряжение питания: 9 В – 28 В постоянного тока
   7. Уникальный идентификатор датчика в веб-интерфейсе: wb-ms\_11/Air Quality (VOC)
   8. Использующийся протокол передачи данных: Modbus RTU, адрес задается программно, заводские настройки указаны на наклейке  
      Интерфейс управления (шина): RS-485
   9. Описание входов и выходов, схема подключения: Этот датчик предназначен для измерения качества воздуха в помещении. Подключается к контроллеру через интерфейс RS-485.
3. Устройство №1
   1. Модуль ввода-вывода WBIO-DO-R10A-8:
   2. Название датчика/устройства: WBIO-DO-R10A-8
   3. Напряжение питания: 9-30 В постоянного тока
   4. Уникальный идентификатор датчика в веб-интерфейсе: WBIO-DO-RxA-8
   5. Использующийся протокол передачи данных: Modbus RTU и Ethernet (при помощи модуля WB-MIO).
   6. Интерфейс управления (шина): RS-485
   7. Описание входов и выходов, схема подключения: Модуль предоставляет 8 цифровых выходов. Он используется для управления различными устройствами и контроля их состояния в системе автоматизации. Подключение осуществляется через интерфейс RS-485.

1. **Modbus RTU** - это протокол передачи данных, который используется для связи между электронными устройствами. Он использует RS-485 для передачи данных и может подключаться до 32 устройств к одной линии связи. Преимущества Modbus RTU включают простоту использования, низкие затраты на оборудование и возможность подключения большого количества устройств к одной линии связи. Недостатки включают ограниченную скорость передачи данных и необходимость использования дополнительного оборудования для подключения к сетям Ethernet или Wi-Fi. Modbus RTU широко используется в промышленности для управления и мониторинга процессов.

2. **1-Wire** - это протокол передачи данных, который используется для связи между электронными устройствами. Он использует один провод для передачи данных и питания устройства. Преимущества 1-Wire включают простоту использования, низкие затраты на оборудование и возможность подключения большого количества устройств к одной линии связи. Недостатки включают ограниченную скорость передачи данных и необходимость использования дополнительного оборудования для подключения к сетям Ethernet или Wi-Fi. 1-Wire широко используется в системах автоматизации зданий, системах безопасности и системах контроля доступа.

3. **I²C (Inter-Integrated Circuit)** - это последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов. Она использует две двунаправленные линии связи (SDA и SCL SDA (Serial Data) — линия передачи данных, а SCL (Serial Clock) — линия тактового сигнала) и применяется для соединения низкоскоростных периферийных компонентов с процессорами и микроконтроллерами. Преимущества использования I²C включают простоту подключения, возможность подключения нескольких устройств к одной шине, а также возможность использования длинных линий связи. Недостатки включают ограниченную скорость передачи данных и ограниченное количество устройств, которые можно подключить к одной шине. I²C широко используется во многих областях, таких как автомобильная промышленность, медицинская техника, промышленный контроль и автоматизация, а также в бытовых.

4. **CAN** - это протокол передачи данных, который используется для связи между электронными устройствами. Он использует две провода для передачи данных и может подключаться до 110 устройств к одной линии связи. Преимущества CAN включают высокую скорость передачи данных, возможность подключения большого количества устройств к одной линии связи и возможность обнаружения ошибок при передаче данных. Недостатки включают более высокую стоимость оборудования по сравнению с другими протоколами передачи данных и более сложную настройку. CAN широко используется в автомобильной промышленности, системах автоматизации зданий и системах контроля доступа.

1. ПРАКТИЧЕСКАЯ РАБОТА №6
   1. Описание задания

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.

Часть 2. Подписка на топик

При помощи улиты mosquitto\_sub подпишитесь на сообщения нескольких датчиков стенда, согласно варианту:

1. Датчик температуры 1-wire DS18B20 (2)

2. Кнопка 28

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.

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

1. Включите индикатор 25

2. Включите и выключите RGB-ленту

Часть 4\*. Сообщения MQTT с внешнего устройства

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

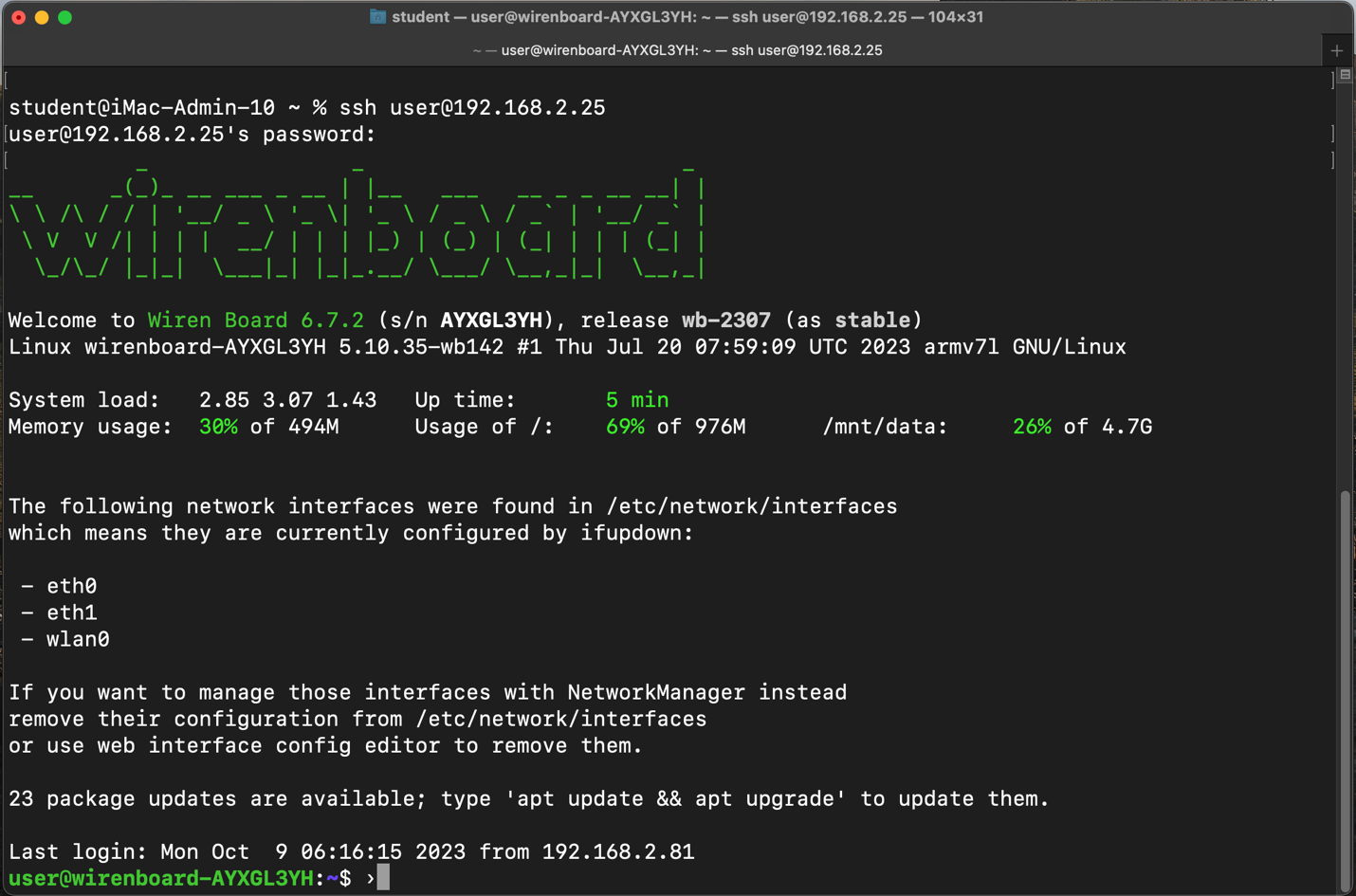
Настройте MQTT-мост (т.е. пересылку всех сообщений MQTT на любой облачный MQTT брокер и обратно) или создайте собственный брокер MQTT.

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

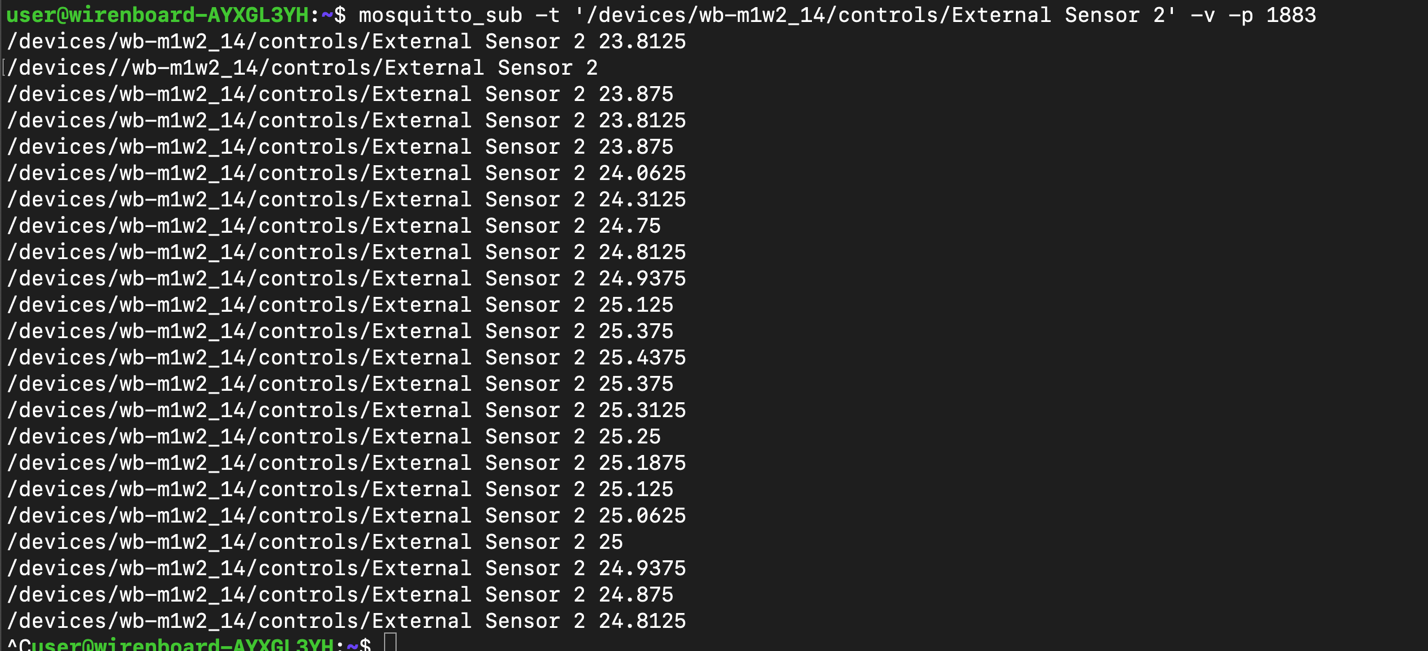
Процесс настройки приведен в документации к контроллеру WirenBoard.

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

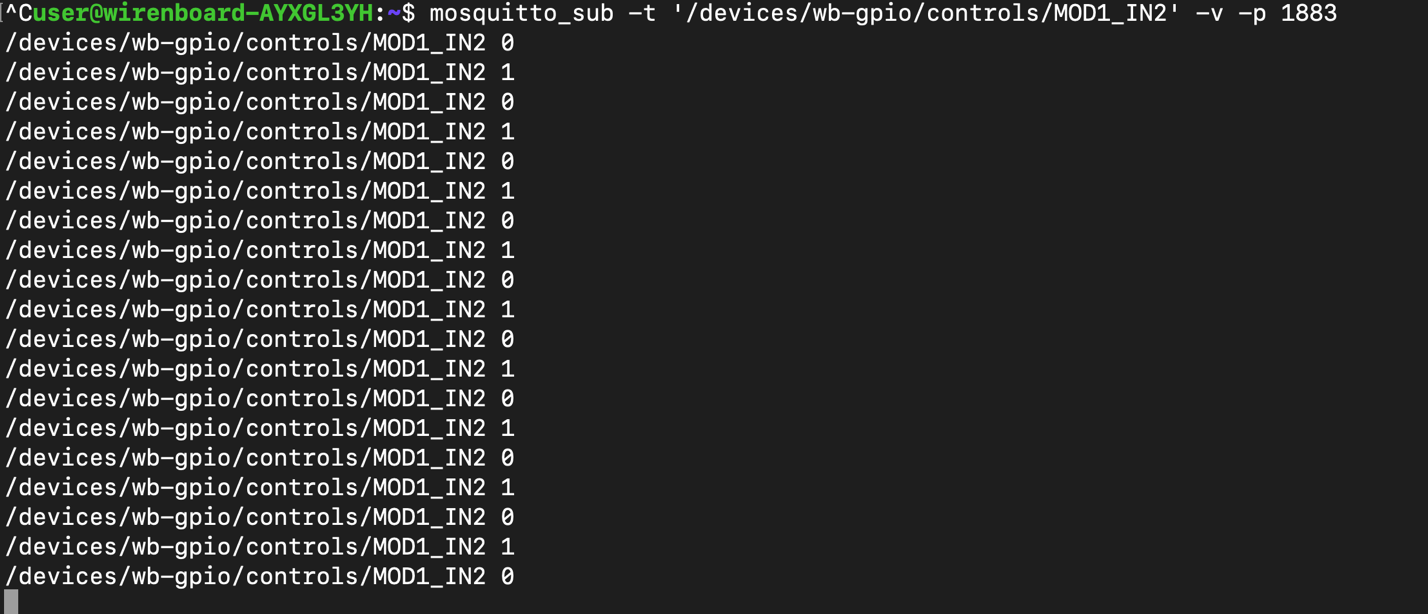
* 1. Выполнение задания



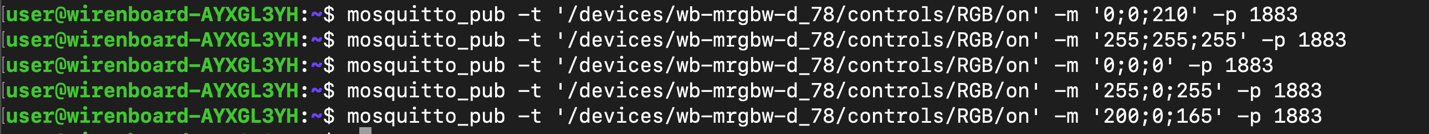
**Рисунок 2.1 – Подключение к консоли**



**Рисунок 2.2 – Подписка на сообщения**



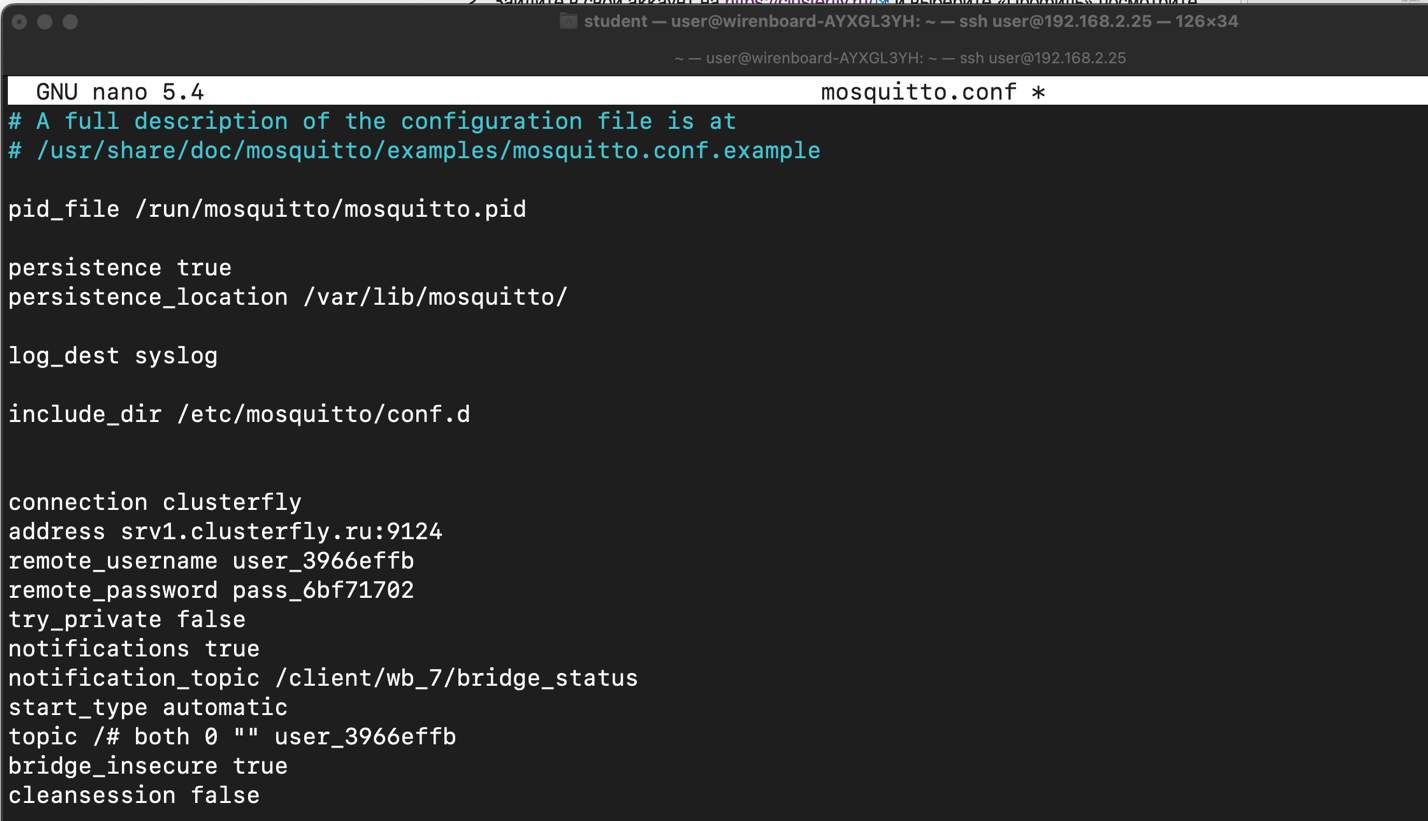
**Рисунок 2.3 - Подписка на сообщения**

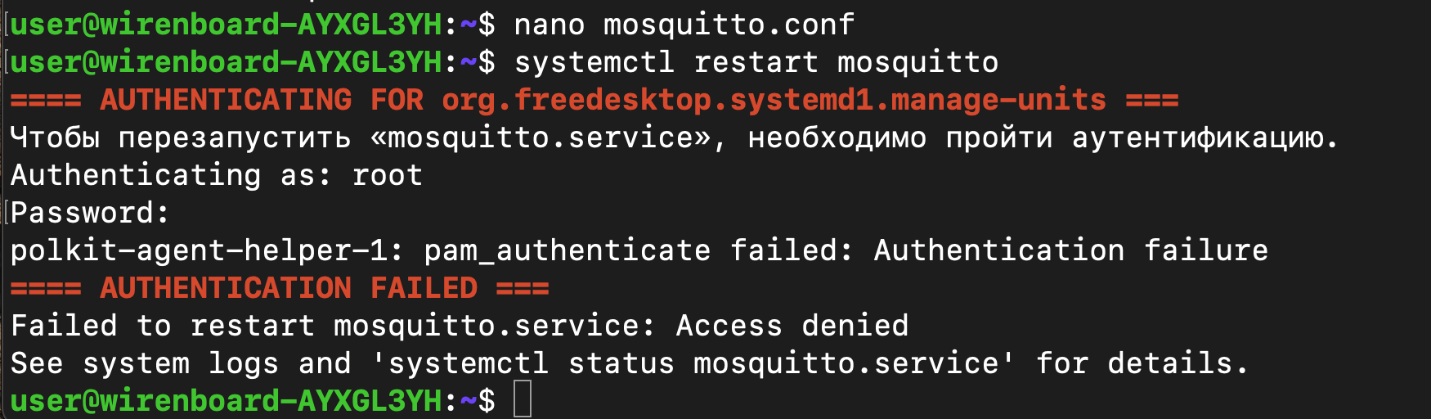


**Рисунок 2.4 – Управление устройством**

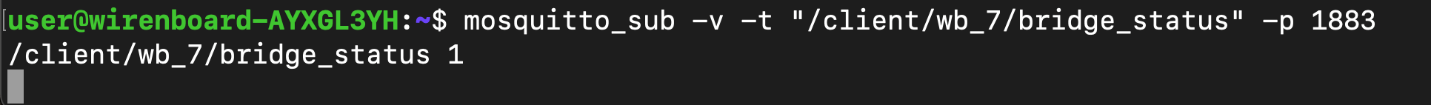


**Рисунок 2.5 - Управление устройством**

**Рисунок 2.6 - Файл конфигурации mosquito.conf**



**Рисунок 2.7 – Сохранение файла и перезагрузка**



**Рисунок 2.8 – Результат после перезагрузки**

1. ПРАКТИЧЕСКАЯ РАБОТА №7
   1. Описание задания

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

1. Датчик температуры 1-wire DS18B20 (1)

2. Датчик шума устройства WB-MSW v.3 (5)

3. Датчик CO2 устройства WB-MSW v.3 (5)

4. Качество воздуха VOC устройства WB-MS v.2 (12)

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).

На любом языке программирования реализуйте программу-парсер, которая бы выводила в консоль данные, полученные из сгенерированных в п.2 файлов.

* 1. Выполнение задания

Команды для подписки на топики:

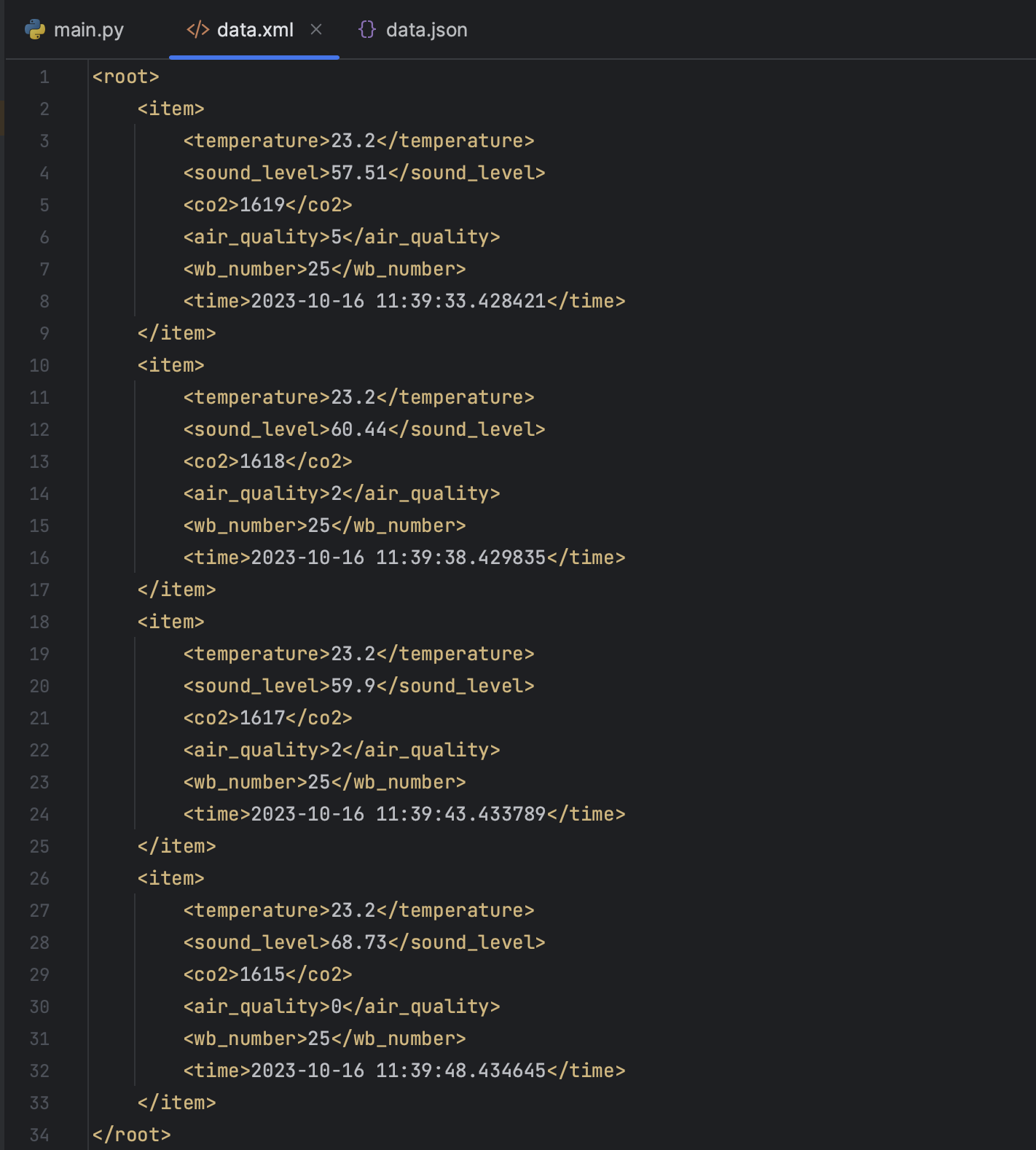
'/devices/wb-msw-v3\_21/controls/Temperature': 'temperature',

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

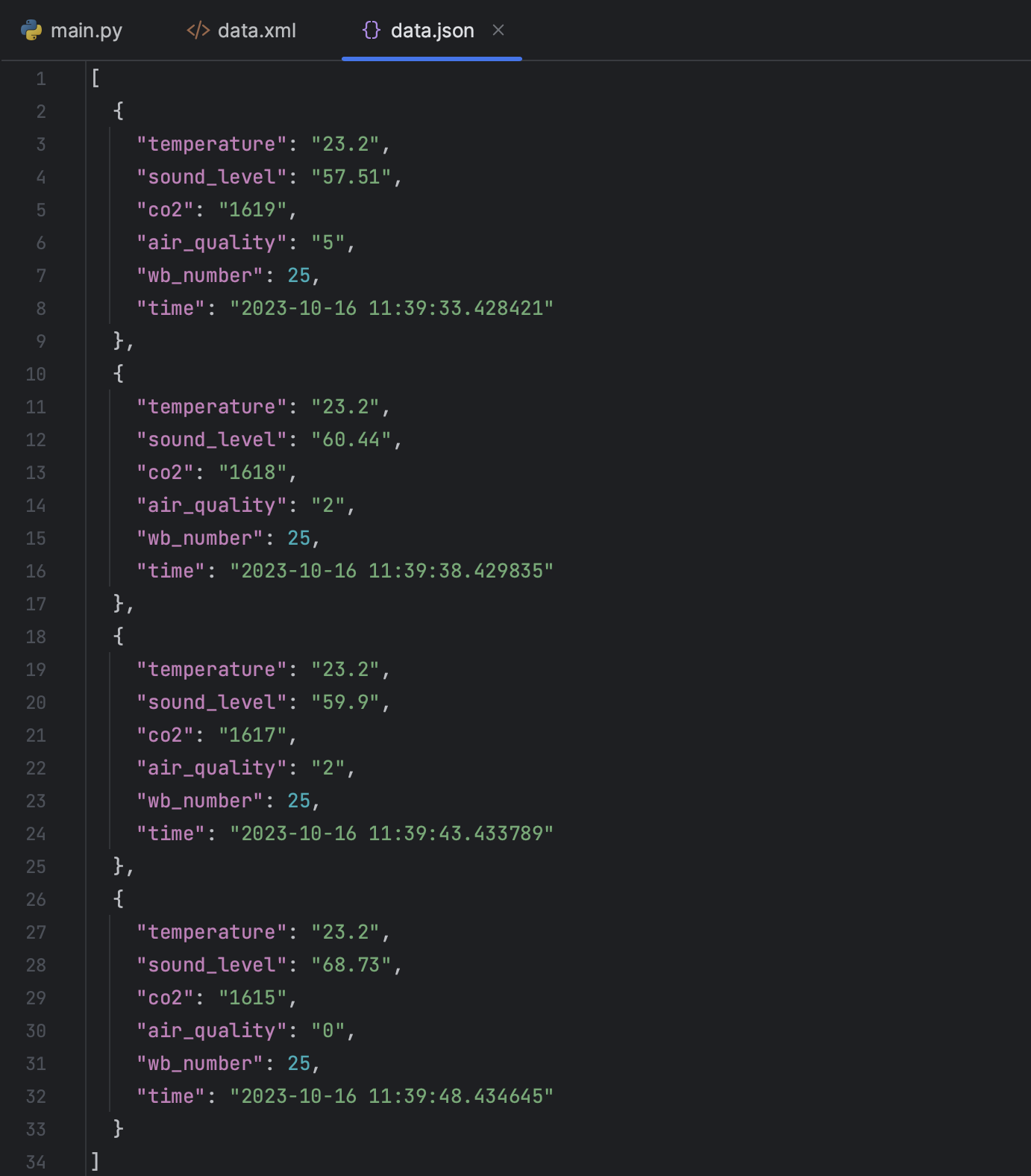
'/devices/wb-msw-v3\_21/controls/CO2': 'co2',

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

Структура данных, получаемых от стенда с описанием каждого поля:



**Рисунок 3.1 – Структура Data.xml**



**Рисунок 3.2 – Структура Data.json**

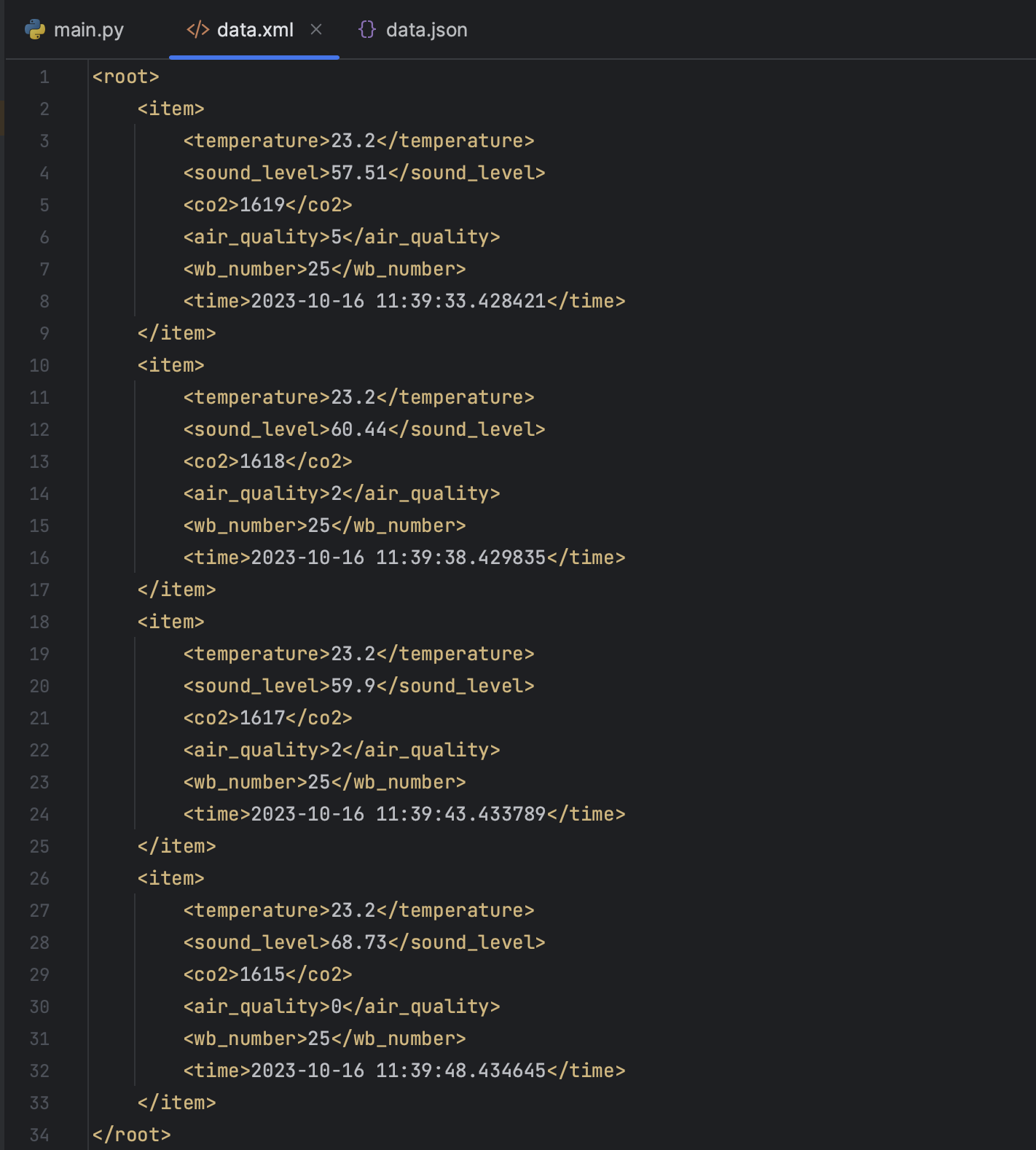
Листинг с комментариями с указанием языка программирования

Данный код был написан на языке Python (листинг 3.1).

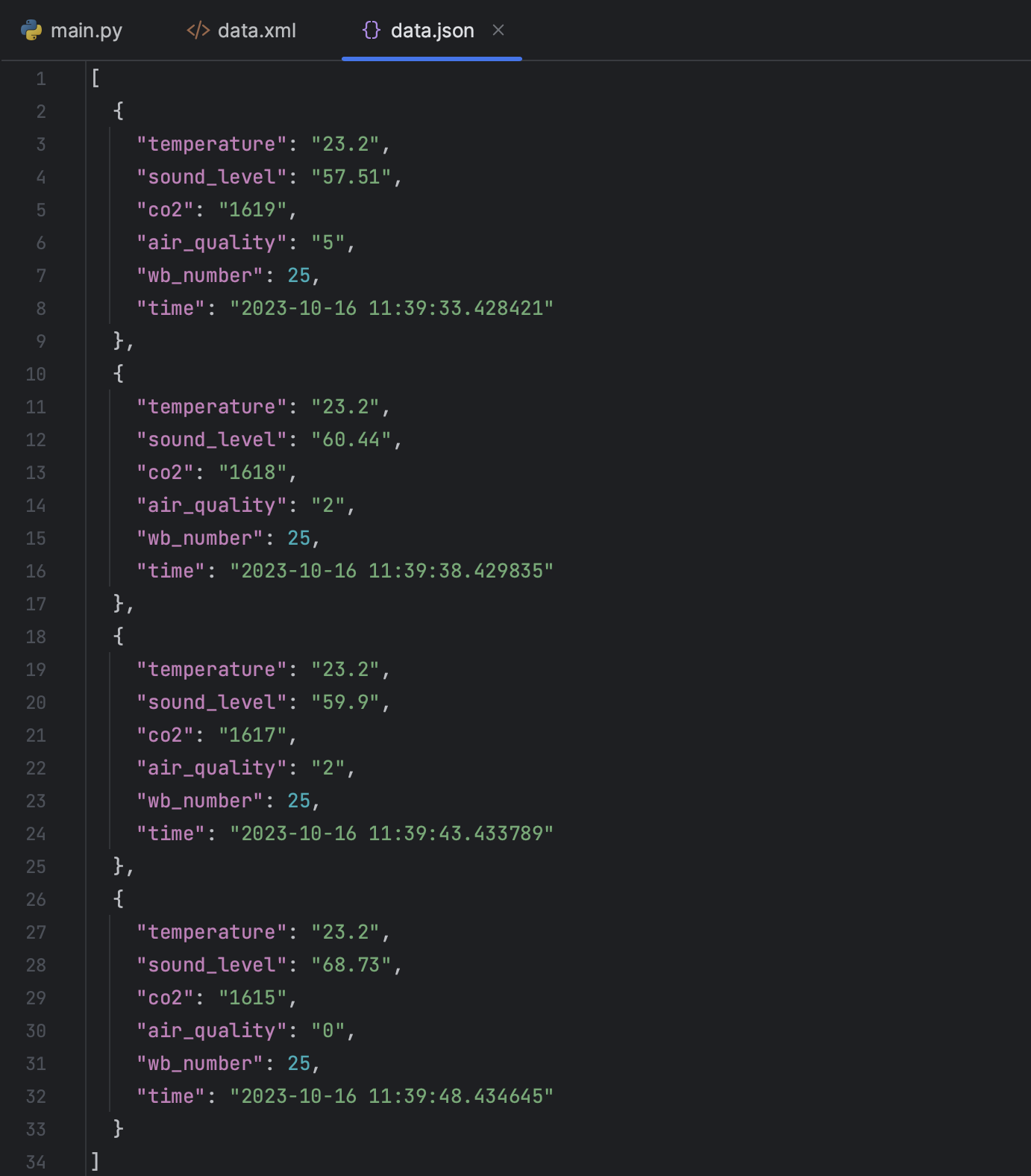
Листинг 3.1 – Изменение цвета диодной ленты по концентрации CO2

|  |
| --- |
| import time from datetime import datetime import paho.mqtt.client as mqtt import json import xml.etree.ElementTree as ET  HOST = "192.168.2.25" PORT = 1883 KEEPALIVE = 20  WB\_NUMBER = 25  SUB\_TOPICS = {  '/devices/wb-msw-v3\_21/controls/Temperature': 'temperature',  '/devices/wb-msw-v3\_21/controls/Sound Level': 'sound\_level',  '/devices/wb-msw-v3\_21/controls/CO2': 'co2',  '/devices/wb-msw-v3\_21/controls/Air Quality (VOC)': 'air\_quality' }  JSON\_LIST = [] 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))   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 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  JSON\_DICT["wb\_number"] = WB\_NUMBER  print(param\_name)   def write\_data\_in\_json():  JSON\_DICT['time'] = str(datetime.now())   JSON\_LIST.append(JSON\_DICT.copy())   with open('data.json', 'w') as file:  json\_string = json.dumps(JSON\_LIST) # Формирование строки JSON из словаря  file.write(json\_string)   def write\_data\_in\_xml():  root = ET.Element("root")   for item in JSON\_LIST:  sub\_element = ET.SubElement(root, "item")  for key, value in item.items():  sub\_sub\_element = ET.SubElement(sub\_element, key)  sub\_sub\_element.text = str(value)   tree = ET.ElementTree(root)  tree.write("data.xml")  def main():   client = mqtt.Client()  client.on\_connect = on\_connect  client.on\_message = on\_message  client.connect(HOST, PORT, KEEPALIVE)  client.loop\_start()   while True:  time.sleep(5)  write\_data\_in\_json()  write\_data\_in\_xml()  print("Данные записаны")   if \_\_name\_\_ == "\_\_main\_\_":  main() |

Скриншоты результатов работы скриптов



**Рисунок 3.3 –Data.xml**



**Рисунок 3.4 –Data.json**

Также был разработан обработчик данных в отформатированном виде (листинг 3.2).

Листинг 3.2 – Разработанный обработчик

|  |
| --- |
| import json  from datetime import datetime  # Открываем файл JSON для чтения  with open('data.json', 'r') as file:  data = json.load(file)  # Выводим отформатированные записи  for record in data:  print('Время:', datetime.strptime(record['time'], "%Y-%m-%d %H:%M:%S.%f").strftime("%H:%M:%S"))  print('\tУровень шума:', record['sound\_level'])  print('\tУровень CO2:', record['co2'])  print('\tНапряжение:', record['voltage'])  print() |

Также показан пример вывода программы на рисунке 3.5.

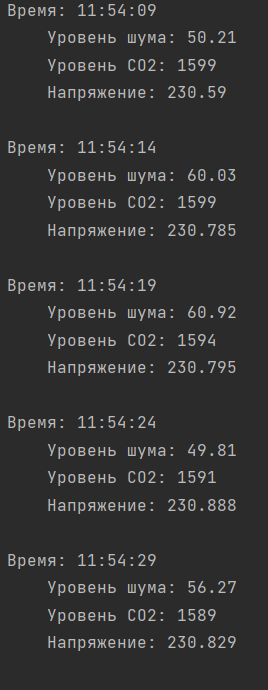


Рисунок 3.5 – Тестирование обработчика

1. ПРАКТИЧЕСКАЯ РАБОТА №8
   1. Описание задания

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.

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

1. Датчик шума устройства WB-MSW v.3

2. Датчик CO2 устройства WB-MSW v.3

3. Напряжение на любом устройстве стенда

Напишите скрипты, позволяющие визуализировать полученные статистические данные в виде: столбиковой диаграммы, линейного графика, круговой диаграммы.

* 1. Выполнение задания

Были написаны скрипты для разных представлений статических данных (листинг 4.1-4.3), а также было протестирована их работоспособность (рис. 4.1-4.3).

Листинг 4.1 – Столбиковые диаграммы

|  |
| --- |
| import json  import matplotlib.pyplot as plt  # Загрузка данных из JSON-файла  with open('data.json', 'r') as file:  data = json.load(file)  # Извлечение значений из данных  time = [entry['time'] for entry in data]  sound\_level = [float(entry['sound\_level']) for entry in data]  co2 = [int(entry['co2']) for entry in data]  voltage = [float(entry['voltage']) for entry in data]  # Разбивка данных на группы по 10 элементов  time = time[::10]  sound\_level = sound\_level[::10]  co2 = co2[::10]  voltage = voltage[::10]  # Создание графиков  fig, axes = plt.subplots(3, 1, figsize=(12, 18))  # График Sound Level  axes[0].bar(time, sound\_level, width=0.2, align='center', label='Sound Level')  axes[0].set\_xlabel('Time')  axes[0].set\_ylabel('Sound Level')  axes[0].set\_title('Sound Level')  # График CO2  axes[1].bar(time, co2, width=0.2, align='center', label='CO2')  axes[1].set\_xlabel('Time')  axes[1].set\_ylabel('CO2')  axes[1].set\_title('CO2')  # График Voltage  axes[2].bar(time, voltage, width=0.2, align='center', label='Voltage')  axes[2].set\_xlabel('Time')  axes[2].set\_ylabel('Voltage')  axes[2].set\_title('Voltage')  # Настройка осей и легенд  for ax in axes:  ax.set\_xticks([])  ax.legend()  # Отображение графиков  plt.tight\_layout()  plt.show() |

Листинг 4.2 – Линейные графики

|  |
| --- |
| import json  import matplotlib.pyplot as plt  import datetime  # Загрузка данных из JSON файла  with open('data.json', 'r') as json\_file:  data = json.load(json\_file)  # Инициализация списков для хранения данных  time\_values = []  sound\_level\_values = []  co2\_values = []  voltage\_values = []  # Разбор данных из JSON и заполнение списков  for entry in data:  time\_str = entry['time']  time\_obj = datetime.datetime.strptime(time\_str, "%Y-%m-%d %H:%M:%S.%f")  time\_values.append(time\_obj)  sound\_level\_values.append(float(entry['sound\_level']))  co2\_values.append(int(entry['co2']))  voltage\_values.append(float(entry['voltage']))  # Создание трех отдельных графиков  fig, (ax1, ax2, ax3) = plt.subplots(3, 1, sharex=True, figsize=(12, 8))  # График для sound\_level  ax1.plot(time\_values, sound\_level\_values, label='Sound Level')  ax1.set\_ylabel('Sound Level')  ax1.legend()  # График для co2  ax2.plot(time\_values, co2\_values, label='CO2', color='orange')  ax2.set\_ylabel('CO2')  ax2.legend()  # График для voltage  ax3.plot(time\_values, voltage\_values, label='Voltage', color='green')  ax3.set\_xlabel('Время')  ax3.set\_ylabel('Voltage')  ax3.legend()  # Настройка заголовков  plt.suptitle('Графики звукового уровня, CO2 и напряжения')  # Отображение графиков  plt.grid(True)  plt.tight\_layout()  plt.show() |

Листинг 4.3 – Круговые диаграммы

|  |
| --- |
| import json  import matplotlib.pyplot as plt  from datetime import datetime  # Загрузка данных из JSON-файла  with open('data.json', 'r') as file:  data = json.load(file)  # Извлечение данных, взятие записи в диапазоне с 20 по 40  times = [datetime.strptime(entry['time'], "%Y-%m-%d %H:%M:%S.%f").strftime("%H:%M:%S") for i, entry in enumerate(data) if 20 <= i <= 40]  sound\_levels = [float(entry['sound\_level']) for i, entry in enumerate(data) if 20 <= i <= 40]  co2\_levels = [int(entry['co2']) for i, entry in enumerate(data) if 20 <= i <= 40]  voltage\_levels = [float(entry['voltage']) for i, entry in enumerate(data) if 20 <= i <= 40]  # Функция для построения круговой диаграммы  def create\_pie\_chart(data, labels, title):  plt.figure(figsize=(6, 6))  plt.pie(data, labels=labels, autopct='%1.1f%%', startangle=140)  plt.title(title)  # Построение круговой диаграммы по уровню звука  create\_pie\_chart(sound\_levels, times, 'Распределение по уровню звука')  # Построение круговой диаграммы по уровню CO2  create\_pie\_chart(co2\_levels, times, 'Распределение по уровню CO2')  # Построение круговой диаграммы по напряжению  create\_pie\_chart(voltage\_levels, times, 'Распределение по напряжению')  # Отображение диаграмм  plt.show() |

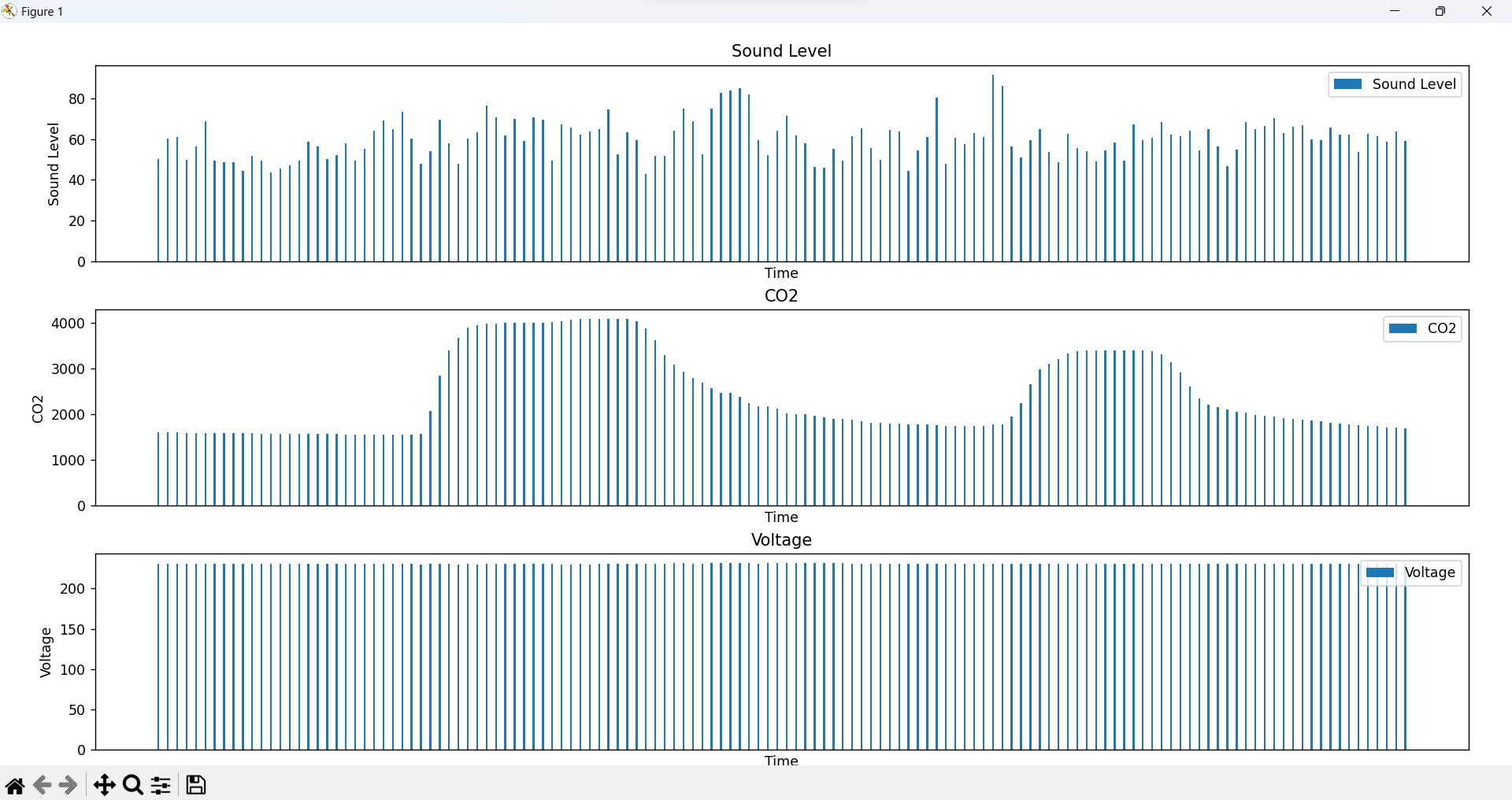


Рисунок 4.1 – Столбиковые диаграммы

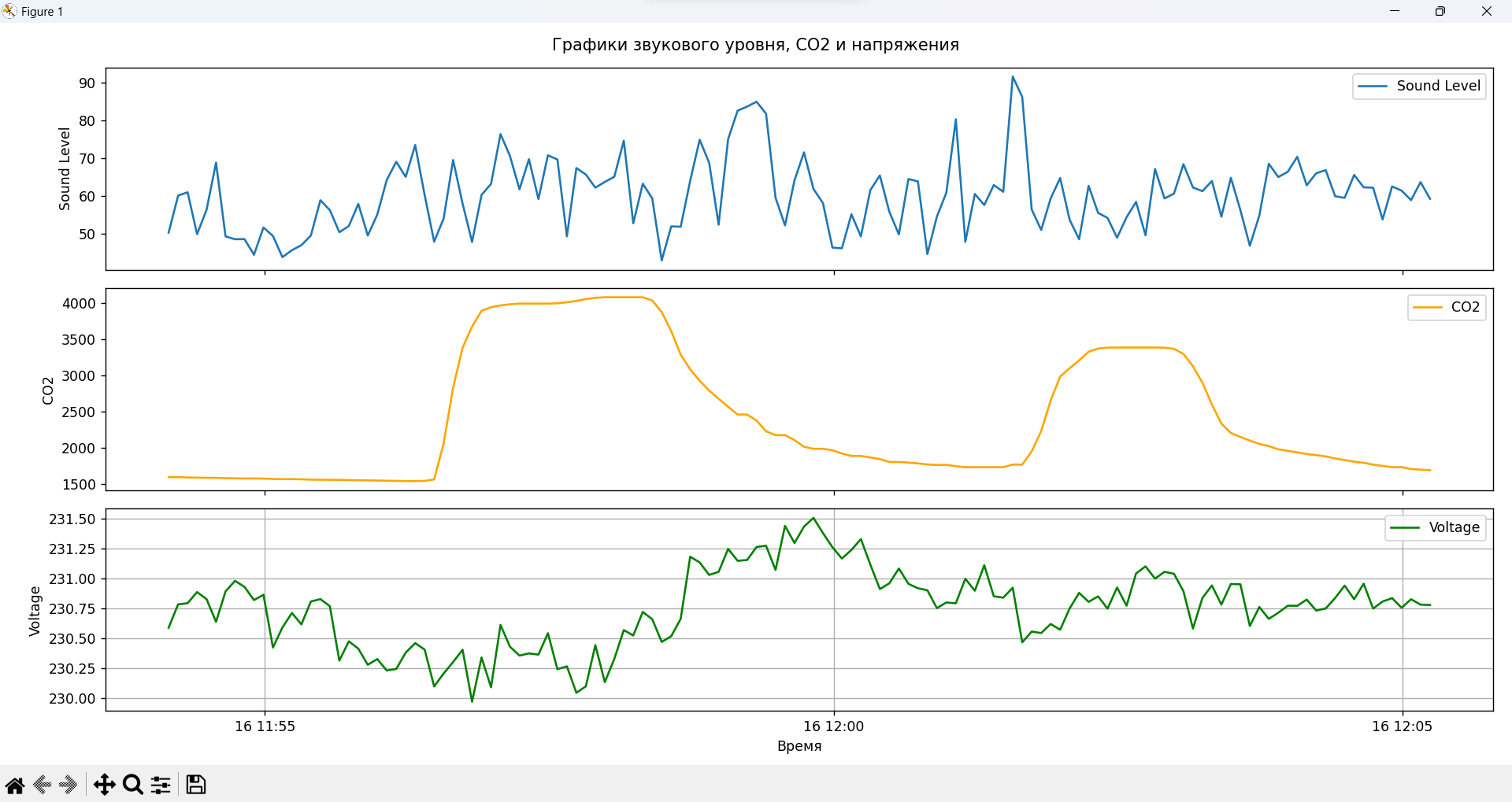


Рисунок 4.2 – Линейные графики

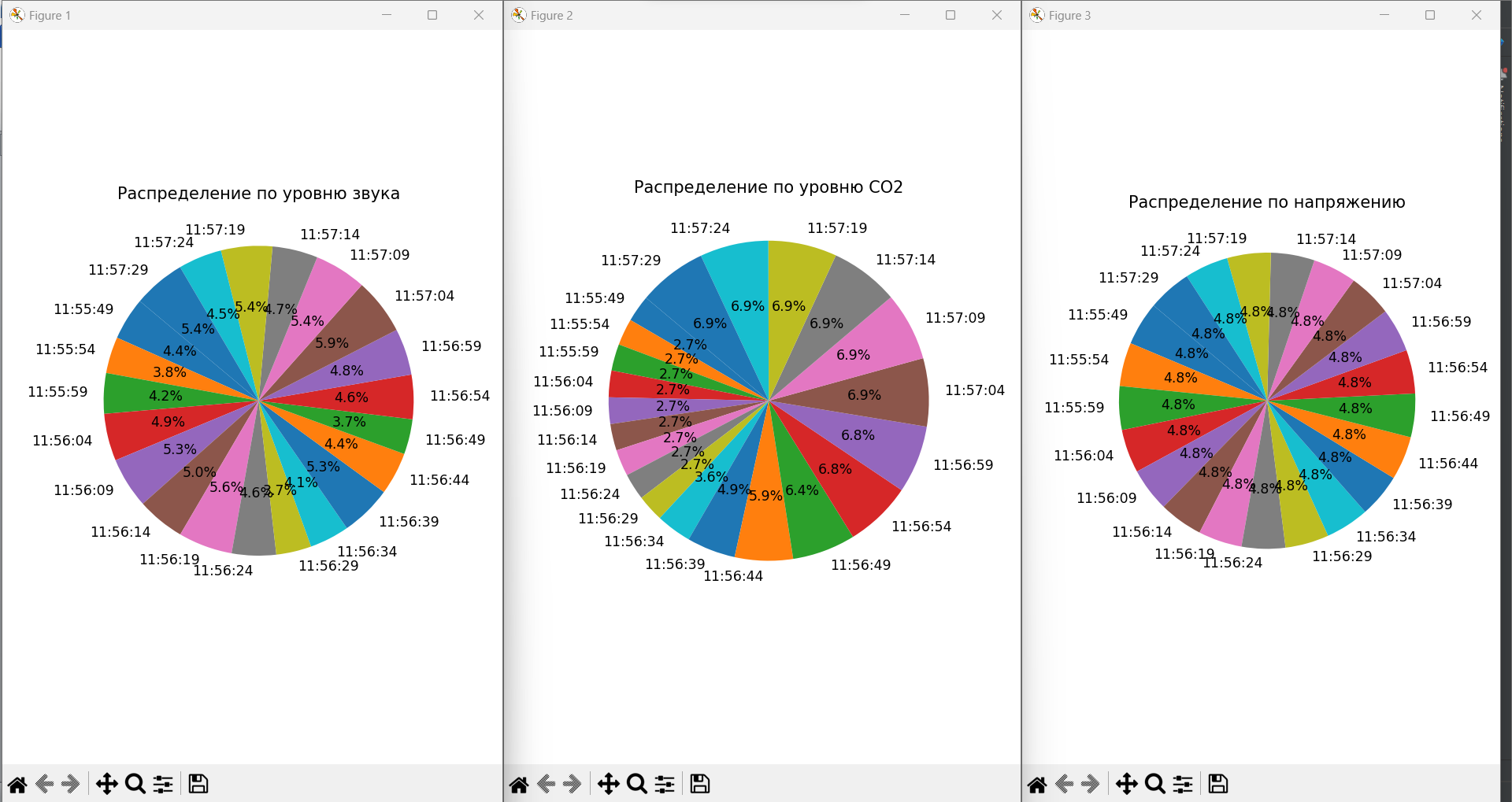


Рисунок 4.3 – Круговые диаграммы

1. ВЫВОД ПО ПРАКТИЧЕСКИМ РАБОТАМ

В ходе выполнения практических работ 5-8 были выполнены все поставленные в условии задачи. А именно: описание 2х датчиков и устройств по шаблону (Название датчика/устройства, Тип измерения, Измеряемые параметры и диапазон измерения, Относительная влажность, Точность, Напряжение питания, Уникальный идентификатор датчика в веб-интерфейсе, Использующийся протокол передачи данных, Интерфейс управления (шина), Описание входов и выходов, схема подключения). Далее мы подключались к консоли WirenBoard по протоколу SSH, подписывались на топик и управляли устройствами, кроме того получали сообщения MQTT с внешнего устройства. В следующей работе, мы подписались на 4 топика в ходе написания программы на языке Python, которая каждые 5 секунд упаковывала последние полученные данные в файлы формата JSON и XML. После чего в последней работе этого раздела была реализована визуализация данных из задания 7 в виде графиков и диаграмм.