



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-03-22

Пашков Д.С.

Чернов А.А.

Шанаев А.В.

Проверил:

Образцов В.М.

2024 г.

СОДЕРЖАНИЕ

ПРАКТИЧЕСКАЯ РАБОТА №5	3
1. Введение.....	3
2. Основная часть	3
2.1 Описание устройств стенда.....	3
2.2 Описание протоколов	6
3. Заключение	9
ПРАКТИЧЕСКАЯ РАБОТА №6	10
1. Введение.....	10
2. Основная часть	10
2.1 Подключение к стенду по SSH	10
2.2 Подписка на топики	11
2.2.1 Датчик движения.....	11
2.2.2 Датчик температуры	11
2.3 Управление устройствами.....	12
2.3.1 Подсветка кнопки 29.....	12
2.3.2 Имитация подачи воды.....	13
3. Заключение	14
ПРАКТИЧЕСКАЯ РАБОТА №7	15
1. Введение.....	15
2. Основная часть	15
2.1 Получение данных	15
2.2 Результаты.....	18
3. Заключение	19
ПРАКТИЧЕСКАЯ РАБОТА №8	20
1. Введение.....	20
2. Основная часть	20
3. Заключение	23
ВЫВОД.....	24

ПРАКТИЧЕСКАЯ РАБОТА №5

1. Введение

Тема: “Измерительные и исполнительные устройства в Интернете вещей”

Цель работы: изучить измерительные и исполнительные устройства в Интернете вещей.

Постановка задачи:

1. Описать измерительные и исполнительные устройства стенда;
 - a. Датчик температуры 1-wire DS18B20 (2)
 - b. Уровень шума в составе устройства WB-MSW v.3 (5)
 - c. Модуль реле 3-канальный WB-MR3 (21)
2. Описать принцип работы, преимущества и недостатки, сферу применения следующих четырех технологий:
 - a. Modbus RTU;
 - b. 1-Wire;
 - c. I²C (IIC, англ. Inter-Integrated Circuit);
 - d. CAN.

2. Основная часть

2.1 Описание устройств стенда

Датчик температуры 1-wire DS18B20

- Название датчика/устройства: Датчик температуры DS18B20
- Тип измерения: Цифровой
- Измеряемые параметры и диапазон измерения:
 - Измеряемый параметр: температура
 - Диапазон измерения: -55°C до +125°C
- Точность: $\pm 0.5^{\circ}\text{C}$ в диапазоне от -10°C до +85°C
- Напряжение питания: 3.0 В до 5.5 В (обычно используется 5 В)

- Уникальный идентификатор датчика в веб-интерфейсе: Обычно отображается в виде уникального 64-битного адреса (например, 28-0000076030a9)

- Используемый протокол передачи данных: 1-Wire
- Интерфейс управления (шина): 1-Wire
- Описание входов и выходов, схема подключения:
 - Входы: Один контакт для подключения к 1-Wire шине.
 - Выходы: Цифровой сигнал с данными о температуре.
 - Схема подключения: Подключение через резистор 4.7 кОм (pull-up) к шине 1-Wire, а также к питанию и земле.

Уровень шума в составе устройства WB-MSW v.3

- Название датчика/устройства: Уровень шума WB-MSW v.3
- Тип измерения: Аналоговый
- Измеряемые параметры и диапазон измерения:
 - Измеряемый параметр: уровень шума (дБ)
 - Диапазон измерения: 30 дБ до 120 дБ
- Точность: Обычно ± 2 дБ
- Напряжение питания: 5 В до 30 В (в зависимости от версии устройства)

- Уникальный идентификатор датчика в веб-интерфейсе: Обычно отображается в виде уникального идентификатора устройства (например, WB-MSW-v3-xxxx)

- Используемый протокол передачи данных: Modbus RTU (по RS-485)
- Интерфейс управления (шина): RS-485
- Описание входов и выходов, схема подключения:
 - Входы: Аналоговый вход для измерения уровня шума.
 - Выходы: Цифровой сигнал уровня шума.

- Схема подключения: Подключение через RS-485, требуется подключение к питанию и общему проводу.

Модуль реле 3-канальный WB-MR3

- Название датчика/устройства: Модуль реле WB-MR3
- Тип измерения: Цифровой
- Измеряемые параметры и диапазон измерения:
 - Измеряемый параметр: состояние реле (включено/выключено)
 - Диапазон: Включение/выключение нагрузки до 16А на каждый канал (в зависимости от модели)
- Точность: Не применимо (измеряются состояния реле)
- Напряжение питания: 5 В до 24 В
- Уникальный идентификатор датчика в веб-интерфейсе: Обычно отображается в виде уникального идентификатора устройства (например, WB-MR3-xxxx)
- Используемый протокол передачи данных: Modbus RTU (по RS-485) или через 1-Wire (в зависимости от настройки)
- Интерфейс управления (шина): RS-485 или 1-Wire
- Описание входов и выходов, схема подключения:
 - Входы: Контакты для подключения к нагрузке (выходные реле).
 - Выходы: Контакты для подключения питания и управляющие сигналы.
 - Схема подключения: Подключение нагрузки к выходам реле, подключение питания и управление через RS-485 или 1-Wire.

2.2 Описание протоколов

Modbus RTU (Remote Terminal Unit)

Принцип работы: Modbus RTU — это промышленный протокол обмена данными между устройствами по последовательному интерфейсу (RS-232, RS-485). Он основан на мастер-слейв (ведущий-ведомый) архитектуре, где один ведущий (контроллер) может управлять несколькими ведомыми устройствами (например, датчиками или исполнительными механизмами). Каждое устройство имеет уникальный адрес, что позволяет ведущему отправлять команды определенному ведомому и получать от него данные.

Преимущества:

- Простота реализации и использования.
- Широко распространен в промышленной автоматизации.
- Поддержка как последовательных интерфейсов (RS-232, RS-485), так и Ethernet (Modbus TCP).
- Высокая надежность и устойчивость к помехам при использовании RS-485.

Недостатки:

- Ограниченная скорость передачи данных (до 115.2 кбит/с).
- Отсутствие механизмов шифрования или защиты данных.
- Мастер-слейв архитектура не поддерживает взаимодействие между ведомыми устройствами.
- Низкая гибкость в расширении сети.

Сферы применения:

- Автоматизация в промышленности.
- Управление электросетями и энергетическими объектами.
- Системы управления HVAC (отопление, вентиляция, кондиционирование).
- Мониторинг и управление производственными процессами.

1-Wire

Принцип работы: 1-Wire — это протокол передачи данных с использованием всего одного сигнального провода (плюс общий провод). Устройства могут подключаться к общей шине, передавая данные поочередно. Устройства могут питаться как от отдельного источника, так и непосредственно от сигнальной линии, что позволяет снизить количество необходимых проводов.

Преимущества:

- Простота подключения устройств (один сигнальный провод).
- Поддержка множества устройств на одной шине.
- Возможность питания устройств через сигнальную линию.
- Низкая стоимость реализации.

Недостатки:

- Низкая скорость передачи данных (обычно до 16 кбит/с).
- Ограниченная длина шины и количество подключенных устройств.
- Подверженность электромагнитным помехам.

Сферы применения:

- Домашняя автоматизация (системы "умный дом").
- Датчики температуры (например, DS18B20).
- Идентификационные системы (iButton).
- Системы мониторинга в небольших сетях.

I²C (Inter-Integrated Circuit)

Принцип работы: I²C — это синхронная двухпроводная шина передачи данных, где используются линии данных (SDA) и синхронизации (SCL). Протокол поддерживает связь между несколькими устройствами на одной шине, при этом каждое устройство имеет уникальный адрес. Мастер-устройство может передавать команды слейв-устройствам, а те, в свою очередь, отвечают, передавая данные.

Преимущества:

- Поддержка нескольких устройств на одной шине.

- Простота подключения и минимальное количество линий.
- Возможность работы с разными скоростями передачи данных (обычно до 400 кбит/с, но есть расширенные версии до 3,4 Мбит/с).
- Широко используется в микроконтроллерах и встраиваемых системах.

Недостатки:

- Ограниченная длина шины (до 1 метра без специальных решений).
- Неустойчивость к электромагнитным помехам.
- Низкая скорость передачи по сравнению с более современными протоколами.

- Сложности с синхронизацией при большом количестве устройств.

Сферы применения:

- Управление периферийными устройствами в микроконтроллерах (датчики, дисплеи, память).
- Встроенные системы и бытовая электроника.
- Цифровые датчики и системы сбора данных.
- Коммуникация между устройствами на одной плате (например, внутри микроконтроллера).

CAN (Controller Area Network)

Принцип работы: CAN — это многомастерная последовательная шина передачи данных, которая использует два провода для связи (CAN_H и CAN_L). Все устройства на шине равноправны и могут передавать сообщения по сети без центрального ведущего. Каждое сообщение имеет уникальный идентификатор, который определяет приоритет сообщения в случае коллизии.

Преимущества:

- Высокая устойчивость к электромагнитным помехам.
- Поддержка нескольких ведущих устройств на одной шине.
- Высокая надежность передачи данных благодаря встроенной системе арбитража и обнаружения ошибок.

- Скорость передачи данных до 1 Мбит/с.

Недостатки:

- Ограниченная длина шины при высоких скоростях (до 40 метров при 1 Мбит/с).
- Сложность реализации и настройки в сравнении с простыми протоколами, такими как UART или SPI.
- В некоторых случаях требуется аппаратная поддержка (специальные микроконтроллеры или чипы).

Сферы применения:

- Автомобильные сети (например, для управления двигателем, ABS и других систем).
- Промышленные сети и автоматизация.
- Управление роботизированными системами.
- Железнодорожные и авиационные системы.

3. Заключение

В процессе выполнения практической работы были изучены измерительные и исполнительные устройства в Интернете вещей.

ПРАКТИЧЕСКАЯ РАБОТА №6

1. Введение

Тема: “Основы работы с протоколом MQTT. Брокераж сообщений”

Цель работы: получить навыки работы с протоколом MQTT.

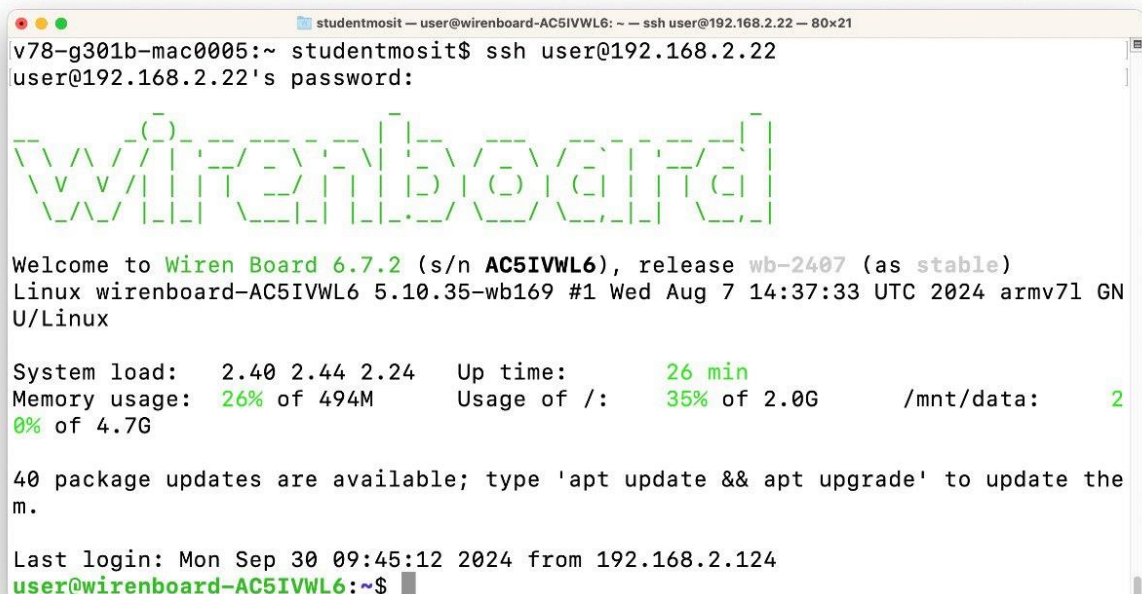
Постановка задачи:

1. Подключиться к стенду по SSH;
2. Подписаться на топики;
 - a. Датчик движения устройства WB-MSW v.3 (5)
 - b. Датчик температуры устройства WB-MS v.2 (12)
3. Управлять устройствами.
 - a. Включить подсветку кнопки 29
 - b. Включить имитацию подачи воды

2. Основная часть

2.1 Подключение к стенду по SSH

На рисунке 1 показан процесс подключения к стенду.



```
studentmosit — user@wirenboard-AC5IVWL6: ~ — ssh user@192.168.2.22 — 80x21
v78-g301b-mac0005:~ studentmosit$ ssh user@192.168.2.22
user@192.168.2.22's password:

w(
i
r
e
n
B
o
a
r
d

Welcome to Wiren Board 6.7.2 (s/n AC5IVWL6), release wb-2407 (as stable)
Linux wirenboard-AC5IVWL6 5.10.35-wb169 #1 Wed Aug 7 14:37:33 UTC 2024 armv7l GN
U/Linux

System load:  2.40 2.44 2.24   Up time:       26 min
Memory usage: 26% of 494M     Usage of /:   35% of 2.0G    /mnt/data:   2
0% of 4.7G

40 package updates are available; type 'apt update && apt upgrade' to update the
m.

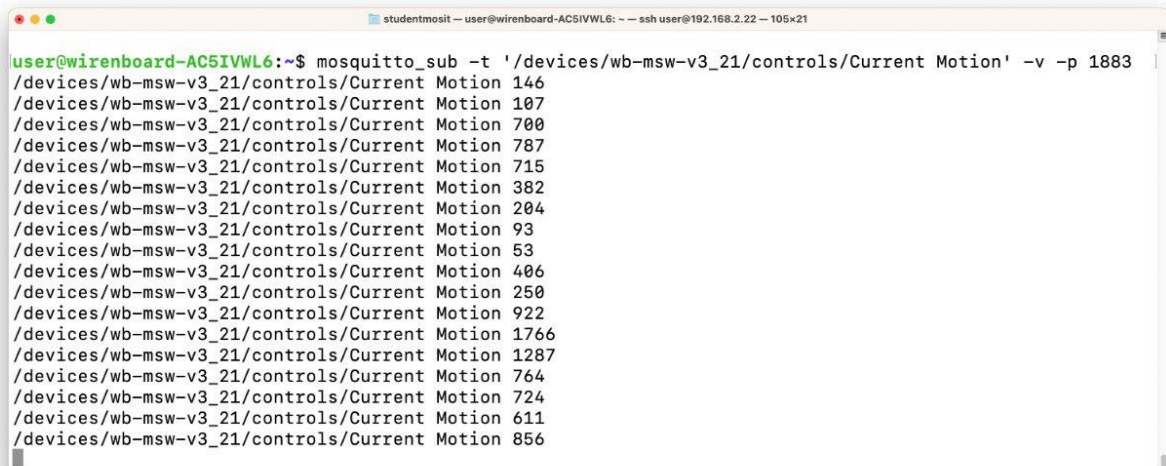
Last login: Mon Sep 30 09:45:12 2024 from 192.168.2.124
user@wirenboard-AC5IVWL6:~$
```

Рисунок 1 – Подключение к стенду по SSH

2.2 Подписка на топики

2.2.1 Датчик движения

На рисунке 2 показан процесс подписки на датчик движения и вывод соответствующих параметров в консоль.



```
studentmosit — user@wireboard-AC5IVWL6: ~ — ssh user@192.168.2.22 — 105x21
user@wireboard-AC5IVWL6:~$ mosquitto_sub -t '/devices/wb-msw-v3_21/controls/Current Motion' -v -p 1883
/devices/wb-msw-v3_21/controls/Current Motion 146
/devices/wb-msw-v3_21/controls/Current Motion 107
/devices/wb-msw-v3_21/controls/Current Motion 700
/devices/wb-msw-v3_21/controls/Current Motion 787
/devices/wb-msw-v3_21/controls/Current Motion 715
/devices/wb-msw-v3_21/controls/Current Motion 382
/devices/wb-msw-v3_21/controls/Current Motion 204
/devices/wb-msw-v3_21/controls/Current Motion 93
/devices/wb-msw-v3_21/controls/Current Motion 53
/devices/wb-msw-v3_21/controls/Current Motion 406
/devices/wb-msw-v3_21/controls/Current Motion 250
/devices/wb-msw-v3_21/controls/Current Motion 922
/devices/wb-msw-v3_21/controls/Current Motion 1766
/devices/wb-msw-v3_21/controls/Current Motion 1287
/devices/wb-msw-v3_21/controls/Current Motion 764
/devices/wb-msw-v3_21/controls/Current Motion 724
/devices/wb-msw-v3_21/controls/Current Motion 611
/devices/wb-msw-v3_21/controls/Current Motion 856
```

Рисунок 2 – Подписка на датчик движения

2.2.2 Датчик температуры

На рисунке 3 показан процесс подписки на датчик температуры и вывод соответствующих параметров в консоль.



```
studentmosit — user@wireboard-AC5IVWL6: ~ — ssh user@192.168.2.22 — 105x21
user@wireboard-AC5IVWL6:~$
user@wireboard-AC5IVWL6:~$
user@wireboard-AC5IVWL6:~$ mosquitto_sub -t '/devices/wb-ms_11/controls/Temperature' -v -p 1883
/devices/wb-ms_11/controls/Temperature 25.96
/devices/wb-ms_11/controls/Temperature 25.93
/devices/wb-ms_11/controls/Temperature 25.96
/devices/wb-ms_11/controls/Temperature 25.93
/devices/wb-ms_11/controls/Temperature 25.97
/devices/wb-ms_11/controls/Temperature 26
/devices/wb-ms_11/controls/Temperature 26.01
/devices/wb-ms_11/controls/Temperature 26.03
/devices/wb-ms_11/controls/Temperature 26.04
/devices/wb-ms_11/controls/Temperature 26.03
/devices/wb-ms_11/controls/Temperature 26.04
/devices/wb-ms_11/controls/Temperature 26.05
/devices/wb-ms_11/controls/Temperature 26.08
/devices/wb-ms_11/controls/Temperature 26.07
/devices/wb-ms_11/controls/Temperature 26.08
/devices/wb-ms_11/controls/Temperature 26.1
```

Рисунок 3 – Подписка на датчик температуры

2.3 Управление устройствами

2.3.1 Подсветка кнопки 29

На рисунке 4 показан процесс изменения состояния подсветки кнопки через консоль.

```
studentmosit — user@wirenboard-AC5IVWL6: ~ — ssh user@192.168.2.22 — 105x21
/devices/wb-ms_11/controls/Temperature 26.08
/devices/wb-ms_11/controls/Temperature 26.1
2/devices/wb-ms_11/controls/Temperature 26.08
_2/devices/wb-ms_11/controls/Temperature 26.1
/devices/wb-ms_11/controls/Temperature 26.08
/devices/wb-ms_11/controls/Temperature 26.1
/devices/wb-ms_11/controls/Temperature 26.08
/devices/wb-ms_11/controls/Temperature 26.05
/devices/wb-ms_11/controls/Temperature 26.07
/devices/wb-ms_11/controls/Temperature 26.05
^Cuser@wirenboard-AC5IVWL6:~$
user@wirenboard-AC5IVWL6:~$
user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/button_light/controls/button3/on" -m "1" -p 1883
user@wirenboard-AC5IVWL6:~$
user@wirenboard-AC5IVWL6:~$
user@wirenboard-AC5IVWL6:~$
user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/button_light/controls/button3/on" -m "0" -p 1883
user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/button_light/controls/button3/on" -m "1" -p 1883
user@wirenboard-AC5IVWL6:~$
```

Рисунок 4 – Переключение подсветки кнопки

На рисунках 5-6 показаны изменения подсветки на стенде.



Рисунок 5 – Включенная подсветка



Рисунок 6 – Выключенная подсветка

2.3.2 Имитация подачи воды

На рисунке 7 показан процесс изменения состояния водяного счётчика.

```
[user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/wb-mwac_68/controls/K2/on" -m "1" -p 1883
[user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/wb-mwac_68/controls/K2/on" -m "0" -p 1883
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ 
[user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/wb-mwac_68/controls/K2/on" -m "1" -p 1883
[user@wirenboard-AC5IVWL6:~$ mosquitto_pub -t "/devices/wb-mwac_68/controls/K2/on" -m "0" -p 1883
[user@wirenboard-AC5IVWL6:~$ █
```

Рисунок 7 – Включение счётчика

На рисунках 8-9 показаны изменения счётчика.



Рисунок 8 – Включенный счётчик



Рисунок 9 – Выключенный счётчик

3. Заключение

В процессе выполнения практической работы были получены навыки работы с протоколом MQTT.

ПРАКТИЧЕСКАЯ РАБОТА №7

1. Введение

Тема: “Форматы представления данных”

Цель работы: получить навыки получения данных от стенда и формирования их представления в виде XML и JSON файлов.

Постановка задачи:

1. Подписаться на топики
 - a. Датчик движения устройства WB-MSW v.3 (5)
 - b. Датчик шума устройства WB-MSW v.3 (5)
 - c. Датчик освещенности устройства WB-MS v.2 (12)
 - d. Датчик температуры устройства WB-MS v.2 (12)
2. На любом языке программирования реализовать 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 реализовать программу-парсер, которая бы выводила в консоль данные, полученные из сгенерированных в п.2 файлов.

2. Основная часть

2.1 Получение данных

Для выполнения поставленных задач был написан Python-скрипт, который выполняет подключение к стенду, получает указанные показания и записывает их в файлы XML и JSON формата. Вместе с этим реализован парсинг записанных данных.

На листинге 1 показан полный код Python-скрипта *main.py*, выполняющий поставленные задачи.

Листинг 1 – main.py

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

# Параметры подключения к MQTT-брокеру
HOST = "192.168.2.22" # IP чемодана
PORT = 1883 # Стандартный порт подключения для Mosquitto
KEEPALIVE = 60 # Время ожидания доставки сообщения
# Словарь с топиками и собираемыми из них параметрами
SUB_TOPICS = {
    '/devices/wb-msw-v3_21/controls/Current Motion': 'value',
    '/devices/wb-msw-v3_21/controls/Sound Level': 'sound_level',
    '/devices/wb-ms_11/controls/Illuminance': 'lux',
    '/devices/wb-ms_11/controls/Temperature': 'temperature',
}

def get_case_number(ip_address):
    return ip_address.split('.')[-1]

sensor_data = {}
for value in SUB_TOPICS.values():
    sensor_data[value] = 0
sensor_data['time'] = ""
sensor_data['case_number'] = get_case_number(HOST)

def save_to_json(data):
    file_path = 'data.json'
    if os.path.exists(file_path):
        try:
            with open(file_path, 'r', encoding='utf-8') as json_file:
                existing_data = json.load(json_file)
            if not isinstance(existing_data, list):
                existing_data = [existing_data]
        except json.JSONDecodeError:
            existing_data = []
    else:
        existing_data = []
```



```

existing_data.append(data)
with open(file_path, 'w', encoding='utf-8') as json_file:
    json.dump(existing_data, json_file, ensure_ascii=False, indent=4)

def save_to_xml(data):
    file_path = 'data.xml'
    if os.path.exists(file_path):
        tree = ET.parse(file_path)
        root = tree.getroot()
    else:
        root = ET.Element("SensorDataCollection")
    sensor_data_element = ET.SubElement(root, "SensorData")
    for key, value in data.items():
        element = ET.SubElement(sensor_data_element, key)
        element.text = str(value)
    tree = ET.ElementTree(root)
    tree.write(file_path, encoding='utf-8', xml_declaration=True)

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):
    payload = msg.payload.decode()
    topic = msg.topic
    print(topic, payload)

    param_name = SUB_TOPICS.get(topic, None)
    if param_name:
        sensor_data[param_name] = payload

    sensor_data['time'] = str(datetime.now())

    print(f"Получено сообщение: {param_name} = {payload}")

def main():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(HOST, PORT, KEEPALIVE)

```

```

client.loop_start()
try:
    while True:
        current_data = sensor_data.copy()
        save_to_json(current_data)
        save_to_xml(current_data)
        print("Данные сохранены в файлы JSON и XML.")

        time.sleep(5)
except KeyboardInterrupt:
    client.loop_stop()
    print("Остановка программы.")

main()

```

Процесс выполнения скрипта показан на рисунке 10 (вместе с получением данных выполняется их сохранение и вывод в консоль).

```

...
client = mqtt.Client()
...
Данные сохранены в файлы JSON и XML.
Connected with result code 0
/devices/MQ-em-v2.12/controls/Current Notion 1234
Выполнен: сохранено: value = 1234
/devices/MQ-em-v2.12/controls/Sound Level 71.79
Выполнен: сохранено: sound_level = 71.79
/devices/MQ-em-v2.12/controls/FillLevel 467
Выполнен: сохранено: Lvl = 467
/devices/MQ-em-v2.12/controls/Temperature 26.31
Выполнен: сохранено: temperature = 26.31
/devices/MQ-em-v2.12/controls/FillLevel 431
Выполнен: сохранено: Lvl = 431
/devices/MQ-em-v2.12/controls/Sound Level 69.68
Выполнен: сохранено: sound_level = 69.68
/devices/MQ-em-v2.12/controls/Current Notion 462
Выполнен: сохранено: value = 462
/devices/MQ-em-v2.12/controls/FillLevel 442
Выполнен: сохранено: Lvl = 442
/devices/MQ-em-v2.12/controls/Sound Level 69.79
Выполнен: сохранено: sound_level = 69.79
/devices/MQ-em-v2.12/controls/Current Notion 377
Выполнен: сохранено: value = 377
Данные сохранены в файлы JSON и XML.
/devices/MQ-em-v2.12/controls/FillLevel 443
Выполнен: сохранено: Lvl = 443
/devices/MQ-em-v2.12/controls/Temperature 26.31
Выполнен: сохранено: temperature = 26.31
/devices/MQ-em-v2.12/controls/Sound Level 68.24
Выполнен: сохранено: sound_level = 68.24
/devices/MQ-em-v2.12/controls/Current Notion 253
Выполнен: сохранено: value = 253
/devices/MQ-em-v2.12/controls/FillLevel 449
Выполнен: сохранено: Lvl = 449
/devices/MQ-em-v2.12/controls/Temperature 26.33
Выполнен: сохранено: temperature = 26.33
Остановка программы.
Process finished with exit code 0

```

Рисунок 10 – Выполнение скрипта

2.2 Результаты

После выполнения написанного скрипта были сформированы файлы *data.xml* и *data.json*. Их содержимое показано на листингах 2-3, соответственно.

Листинг 2 – data.xml

```
<?xml version='1.0' encoding='utf-8'?>
<SensorDataCollection>
<SensorData>
<value>377</value>
<sound_level>59.79</sound_level>
<lux>442</lux>
<temperature>26.03</temperature>
<time>2024-10-11 13:56:35.658293</time>
<case_number>22</case_number>
</SensorData>
</SensorDataCollection>
```

Листинг 3 – data.json

```
[
  {
    "value": "377",
    "sound_level": "59.79",
    "lux": "442",
    "temperature": "26.03",
    "time": "2024-10-11 13:56:35.658293",
    "case_number": "22"
  }
]
```

3. Заключение

В процессе выполнения практической работы были получены навыки получения данных от стенда и формирования их представления в виде XML и JSON файлов.

ПРАКТИЧЕСКАЯ РАБОТА №8

1. Введение

Тема: “Визуализация данных в Интернете вещей”

Цель работы: получить навыки визуализации данных, получаемых от стенда, в виде диаграмм разных видов.

Постановка задачи:

1. Подписаться на топики
 - a. Датчик движения устройства WB-MSW v.3 (5)
 - b. Датчик температуры устройства WB-MS v.2 (12)
 - c. Напряжение на любом устройстве стенда
2. Написать скрипты визуализации в виде данных диаграмм

2. Основная часть

Для получения данных в формате JSON был использован Python-скрипт, разработанный в практической работе 7 (были изменены топики).

В результате 10-минутной работы скрипта был сформирован JSON-файл, используемый при построении диаграмм.

Для построения диаграмм был написан Python-скрипт, выполняющий парсинг данных из файла *data.json* и дальнейшее построение диаграмм разных видов. На листинге 4 показан код скрипта.

Листинг 4 – Python-скрипт для построения диаграмм

```
from collections import Counter

import matplotlib.pyplot as plt
import json

def load_data_from_json(file_path):
    with open(file_path, 'r') as file:
        data = json.load(file)
    return data

def extract_data(data):
    values = [int(item['value']) for item in data]
    temperatures = [float(item['temperature']) for item in data]
```

```

voltages = [float(item['voltage']) for item in data]
return values, temperatures, voltages

def plot_value_bar_chart(values):
    plt.figure(figsize=(10, 6))
    plt.bar([i for i in range(0, len(values))], values, color='blue', label='Уровень
движения')
    plt.xlabel('Шаг')
    plt.ylabel('Уровень движения')
    plt.title('Столбиковая диаграмма уровня движения')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

def plot_temperature_line_chart(temperatures):
    plt.figure(figsize=(10, 6))
    plt.plot([i for i in range(0, len(values))], temperatures, color='green', la-
bel='Температура')
    plt.xlabel('Шаг')
    plt.ylabel('Температура')
    plt.title('Линейная диаграмма температуры')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

def plot_voltage_circle_chart(voltages):
    voltage_counts = Counter(voltages)

    labels = [f'{voltage} V' for voltage in voltage_counts.keys()]
    sizes = voltage_counts.values()

    plt.figure(figsize=(10, 6))
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
    plt.title('Круговая диаграмма напряжения')
    plt.axis('equal')
    plt.show()

data = load_data_from_json("data.json")
values, temperatures, voltages = extract_data(data)

plot_value_bar_chart(values)

```

```
plot_temperature_line_chart(temperatures)
plot_voltage_circle_chart(voltages)
```

Для показаний датчика движения была построена диаграмма, показанная на рисунке 11.

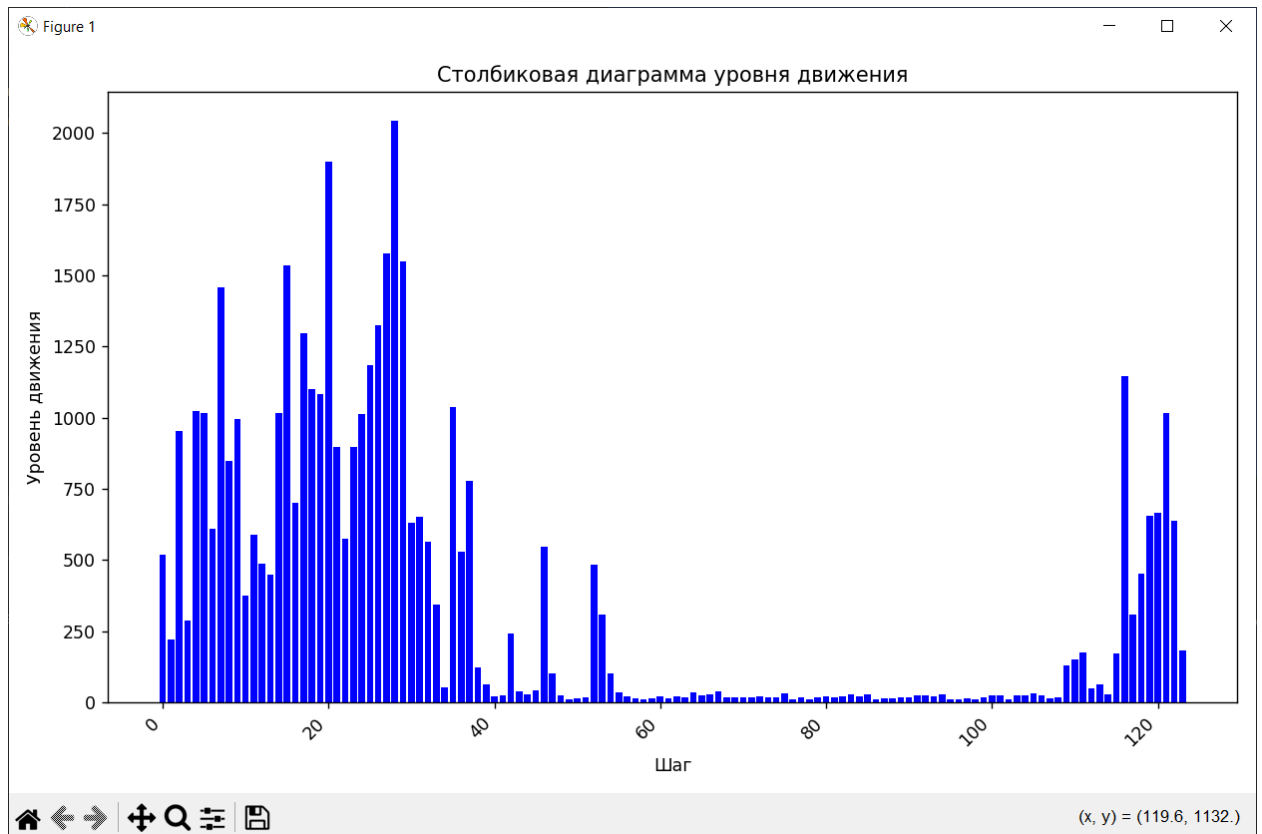


Рисунок 11 – Уровень движения

Для показаний датчика температуры была построена диаграмма, показанная на рисунке 12.

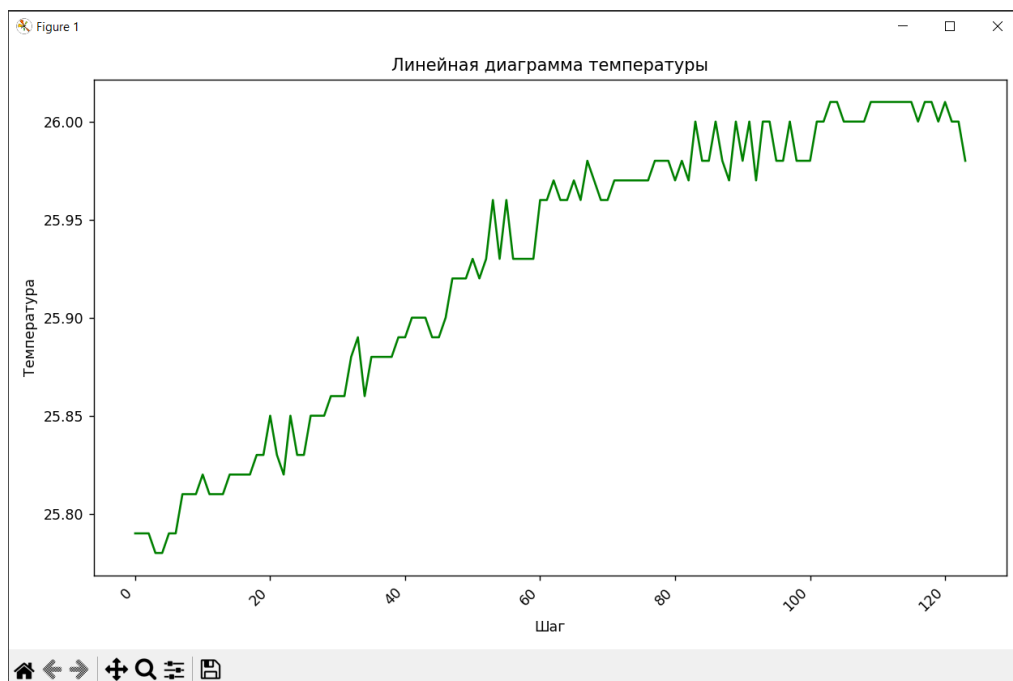


Рисунок 12 – Температура

Для показаний датчика напряжения была построена диаграмма, показанная на рисунке 13.

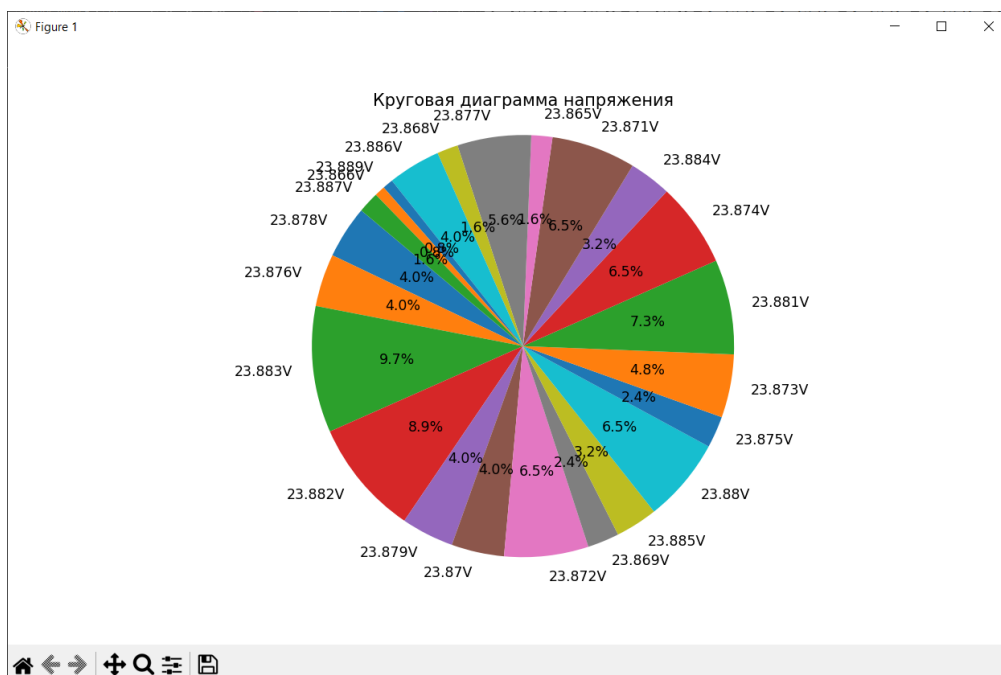


Рисунок 13 – Напряжение

3. Заключение

В процессе выполнения практической работы были получены навыки визуализации данных, получаемых от стенда, в виде диаграмм разных видов.

ВЫВОД

В ходе выполнения практических работ №5-8 были изучены и применены различные технологии и протоколы, используемые в IoT-системах.

Был получен опыт работы с измерительными и исполнительными устройствами, такими как датчики температуры, реле и устройства контроля шума, а также изучены принципы их подключения и взаимодействия через различные протоколы передачи данных (Modbus RTU, 1-Wire, I²C, CAN). Кроме того, был освоен протокол MQTT для брокеража сообщений, что позволило научиться управлять устройствами и подписываться на топики для мониторинга различных параметров.

Практическая работа с форматами представления данных дала возможность понять, как данные передаются и обрабатываются в IoT-системах, а также как их можно эффективно использовать для принятия решений и автоматизации процессов.

В результате этих работ был закреплён навык интеграции измерительных устройств с исполнительными системами, что является основой для создания эффективных IoT-решений.