厦門大學

软件学院

物联网技术导论实验四

班　　级 软工三班

学　　院 信息学院

专　　业 软件工程

年　　级 2021 级

学　　号 32420212202930

姓　　名 陈澄

1 实验内容

延续实验三的内容，在对比MQTT和UDP基础上：

1. 实现1台服务器接受3个以上时间序列传感器数据（频次相同），并可视化显示——单独显示趋势和曲线、组合显示（需要归一化处理）。
2. 设置报警条件，单传感器数据超过异常告警范围的，需要页面提示报警。最终推送手机报警——采用钉钉的开放Webhook接口，需要能够在钉钉中可以查看传感器的数据和报警内容（钉钉机器人webhook接口）。

2 实验环境

1、编译环境：IDEA Intellij、Visual Studio Code

2、前端：html、javascript

3、后端：

(1)服务端：Springboot

(2)客户端：java+maven

3 实验步骤

1. 实现1台服务器接受3个以上时间序列传感器数据（频次相同），并可视化显示——单独显示趋势和曲线、组合显示（需要归一化处理）。

(1)消息发送：使用特定的字段区分各项数据并发送

```java
while (true) {
    // 模拟温度数据
    double temperature = 20 + random.nextGaussian() * 5; // 均值为20，标准差为5
    String data = "t" + String.valueOf(temperature);
    byte[] sendData = data.getBytes();
    DatagramPacket packet = new DatagramPacket(sendData, sendData.length, address, port);
    socket.send(packet);
    System.out.println("Sent temperature: " + data);

    //模拟压力数据
    double pressure = 100000 + random.nextGaussian() * 20;
    data = "p" + String.valueOf(pressure);
    sendData = data.getBytes();
    packet = new DatagramPacket(sendData, sendData.length, address, port);
    socket.send(packet);
    System.out.println("Sent pressure: " + data);

    //模拟体积数据
    double volume = 100 + random.nextGaussian() * 10;
    data = "v" + String.valueOf(volume);
    sendData = data.getBytes();
    packet = new DatagramPacket(sendData, sendData.length, address, port);
    socket.send(packet);
    System.out.println("Sent volume: " + data);

    Thread.sleep( millis: 5000); // 采样频率设定为5秒
}
```

(2)消息接收：解析数据后存入数据库

```java
while (true) {
    // 创建接收数据报
    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
et serverSocket = new DatagramSocket(12345)
    // 接收数据报
    serverSocket.receive(receivePacket);

    // 解析接收到的数据
    String message = new String(receivePacket.getData(), offset: 0, receivePacket.getLength());
    System.out.println("收到消息: " + message);

    MessageMapper messageMapper = context.getBean(MessageMapper.class);

    if(message.charAt(0)=='t') {
        TemperatureMessage temperature = new TemperatureMessage(LocalDateTime.now(), Double.parseDouble(message.s
        messageMapper.insertTemperature(temperature);
        if(Double.parseDouble(message.substring( beginIndex: 1))>24.0) CustomRobotGroupMessage.send( s: "温度超过阈值24
    }
    else if(message.charAt(0)=='p'){
        PressureMessage pressure = new PressureMessage(LocalDateTime.now(), Double.parseDouble(message.substring(
        messageMapper.insertPressure(pressure);
        if(Double.parseDouble(message.substring( beginIndex: 1))>100020.0) CustomRobotGroupMessage.send( s: "压力超过阈
    }
    else if(message.charAt(0)=='v'){
        VolumeMessage volume = new VolumeMessage(LocalDateTime.now(), Double.parseDouble(message.substring( beginIn
        messageMapper.insertVolume(volume);
        if(Double.parseDouble(message.substring( beginIndex: 1))>110.0) CustomRobotGroupMessage.send( s: "体积超过阈值11
    }
}
```

(3)可视化显示：使用 https://cdn.jsdelivr.net/npm/chart.js 的图表工具，通过 fetch 方法从服务端获取 json 数据解析后以折线图的形式显示。（此处是其中一种数据）

```
fetch('http://localhost:8080/pressure')
    .then(response => response.json())
    .then(data => {
        const dates = data.map(entry => entry.date);
        const pressure = data.map(entry => entry.pressure);
                                                    any
        var ctx = document.getElementById('Chart2').getContext('2d');

        var myChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: dates,
                datasets: [{
                    label: 'Pressure',
                    data: pressure,
                    backgroundColor: 'rgba(54, 162, 235, 0.2)',
                    borderColor: 'rgba(54, 162, 235, 1)',
                    borderWidth: 1
                }]
            }
        });
    });
```

(4)归一化处理：获取每个传感器时间序列中数据的最大值最小值，重新映射到 0-1 的范围内，并将三种数据展现在一张图表中。

```
// 归一化函数
function normalizeData(data) {
    const max = Math.max(...data);
    const min = Math.min(...data);
    return data.map(value => (value - min) / (max - min));
}

// 在获取数据后，对数据进行归一化处理
fetch('http://localhost:8080/temperature')
    .then(response => response.json())
    .then(temperatureData => {
        fetch('http://localhost:8080/pressure')
            .then(response => response.json())
            .then(pressureData => {
                fetch('http://localhost:8080/volume')
                    .then(response => response.json())
                    .then(volumeData => {
                        // 获取日期
                        const dates = temperatureData.map(entry => entry.date);
                        // 获取温度、压力和体积数据
                        const temperature = normalizeData(temperatureData.map(entry => entry.temperature));
                        const pressure = normalizeData(pressureData.map(entry => entry.pressure));
                        const volume = normalizeData(volumeData.map(entry => entry.volume));

                        // 创建图表
                        var ctx = document.getElementById('CombinedChart').getContext('2d');

                        var myChart = new Chart(ctx, {
                            type: 'line',
                            data: {
                                labels: dates,
                                datasets: [{
                                    label: 'Temperature',
                                    data: temperature,
                                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                                    borderColor: 'rgba(75, 192, 192, 1)',
                                    borderWidth: 1
                                },
                                {
                                    label: 'Pressure',
                                    data: pressure,
                                    backgroundColor: 'rgba(54, 162, 235, 0.2)',
                                    borderColor: 'rgba(54, 162, 235, 1)',
                                    borderWidth: 1
                                },
                                {
                                    label: 'Volume',
                                    data: volume,
                                    backgroundColor: 'rgba(255, 206, 86, 0.2)',
                                    borderColor: 'rgba(255, 206, 86, 1)',
                                    borderWidth: 1
                                }
                                ]
                            }
                        });
                    });
            });
    });
```

3. 设置报警条件，单传感器数据超过异常告警范围的，需要页面提示报警。最终推送手机报警——采用钉钉的开放 Webhook 接口，需要能够在钉钉中可以查看传感器的数据和报警内容（钉钉机器人 webhook 接口）。

(1) 前段数据检测

设置阈值并添加变量进行判断，如果超越阈值则以新的样式显示在图表中。

```
fetch('http://localhost:8080/temperature')
    .then(response => response.json())
    .then(data => {
        const dates = data.map(entry => entry.date);
        const temperature = data.map(entry => entry.temperature);
        const threshold = 25; // 你可以根据需要设置阈值

        var ctx = document.getElementById('Chart1').getContext('2d');

        // 添加报警信息的显示逻辑
        const alertData = temperature.map(temp => temp > threshold ? temp : null);

        var myChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: dates,
                datasets: [{
                    label: 'Temperature',
                    data: temperature,
                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                },
                {
                    label: 'Alert',
                    data: alertData,
                    backgroundColor: 'rgba(255, 0, 0, 0.2)', // 红色背景表示报警
                    borderColor: 'rgba(255, 0, 0, 1)', // 红色边框表示报警
                    borderWidth: 3
                }]
            }
        });
    });
```

(2) 后端数据检测

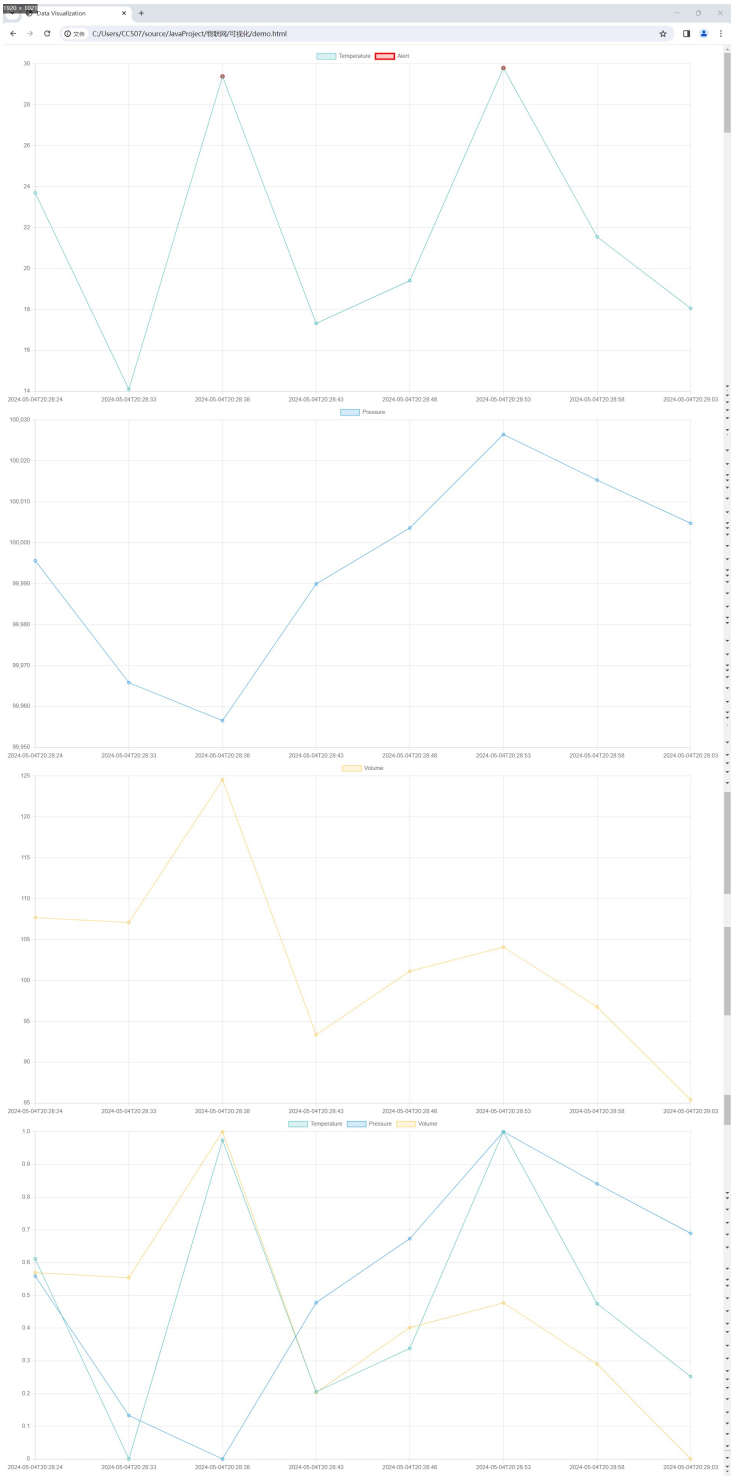
每次接收新的数据都进行一次判断，是否超越阈值，是则调用 Webhook 接口发送钉钉消息。

```
if(message.charAt(0)=='t') {
    TemperatureMessage temperature = new TemperatureMessage(LocalDateTime.now(), Double.parseDouble(message.subs
    messageMapper.insertTemperature(temperature);
    if(Double.parseDouble(message.substring( beginIndex: 1))>24.0) CustomRobotGroupMessage.send( s: "温度超过阈值24!");
}
```

消息发送代码：

```java
public class CustomRobotGroupMessage {

    1 个用法
    public static final String CUSTOM_ROBOT_TOKEN = "f7c84318d549069e9582e1ed71ccceffa9d55fb68d2fe7ffe3e4eb9f9623e

    0 个用法
    public static final String USER_ID= "<you need @ group user's userId>";

    1 个用法
    public static final String SECRET = "";

    3 个用法
    public static void send(String s) {
        try {
            Long timestamp = System.currentTimeMillis();
            System.out.println(timestamp);
            String secret = SECRET;
            String stringToSign = timestamp + "\n" + secret;
            Mac mac = Mac.getInstance( algorithm: "HmacSHA256");
            mac.init(new SecretKeySpec(secret.getBytes( charsetName: "UTF-8"), algorithm: "HmacSHA256"));
            byte[] signData = mac.doFinal(stringToSign.getBytes( charsetName: "UTF-8"));
            String sign = URLEncoder.encode(new String(Base64.encodeBase64(signData)), enc: "UTF-8");
            System.out.println(sign);

            //sign字段和timestamp字段必须拼接到请求URL上，否则会出现 310000 的错误信息
            DingTalkClient client = new DefaultDingTalkClient( serverUrl: "https://oapi.dingtalk.com/robot/send?sign=
            OapiRobotSendRequest req = new OapiRobotSendRequest();
            /**
             * 发送文本消息
             */
            //定义文本内容
            OapiRobotSendRequest.Text text = new OapiRobotSendRequest.Text();
            text.setContent(s);
            //定义 @ 对象
            //OapiRobotSendRequest.At at = new OapiRobotSendRequest.At();
            //at.setAtUserIds(Arrays.asList(USER_ID));
            //设置消息类型
            req.setMsgtype("text");
            req.setText(text);
            //req.setAt(at);
            OapiRobotSendResponse rsp = client.execute(req, CUSTOM_ROBOT_TOKEN);
            System.out.println(rsp.getBody());
        } catch (ApiException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException(e);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        } catch (InvalidKeyException e) {
            throw new RuntimeException(e);
        }
    }
}
```

4 实验结果


Data Visualization
C:/Users/CC507/source/JavaProject/物联网可视化/demo.html
Temperature
Alert
Pressure
Volume
Temperature
Pressure
Volume

5 我的体会

通过这个实验，我对物联网系统的设计和实现有了更深入的了解，特别是在数据传输、处理和可视化，以及实时监控和异常处理方面。这些技能对于未来在物联网领域的工作和研究都将非常有用。