

廈門大學



软件学院

物联网技术导论实验二

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 实验背景

随着科技的不断发展和应用范围的扩大，传感器在各个领域的重要性日益突显。传感器可以将物理世界中的各种参数转换为电信号或数字信号，以便于采集、处理和分析。从工业生产到医疗保健，从环境监测到智能家居，传感器的应用无处不在，它们为我们提供了丰富的数据，帮助我们更好地理解和控制我们周围的环境。

然而，在传感器的设计、开发和应用过程中，工程师们经常面临着一些挑战。其中之一是在实际硬件开发之前对传感器进行有效的测试和验证。比如传感器的性能受到诸如环境条件、电路设计、信号处理算法等多种因素的影响，因此在实际应用之前，对传感器进行充分的仿真和测试非常重要。

2 实验内容

2.1 模拟虚拟传感器

用 java、python 或者其他的编程语言，写一个能够定时发送 UDP 报文的虚拟传感器（包括 UDP 报文的接收服务器）。可以模拟温度传感器，温湿度传感器,气象传感器，土壤传感器，也可以模拟随机的 GPS 位置变化。传感器的采样频率可以自行设定。传感器的类型没有限制，自由发挥。

此处选用 java 语言，模拟温度传感器

(1) 客户端 (Client)

先使用 DatagramSocket 创建 socket 对象

再获取 localhost 的 InetAddress 对象，并指定端口为 12345，此为服务器接收信息的端口

使用 Random 生成随机数模拟 20 度左右的温度

创建一个 DatagramPacket 对象，用于封装要发送的数据、目标地址和端口。

循环每五秒调用 DatagramPacket 的 send 发送数据包，模拟温度的实时更新

```

public static void main(String[] args) {
    try {
        DatagramSocket socket = new DatagramSocket();
        InetAddress address = InetAddress.getByName("localhost");
        int port = 12345;

        Random random = new Random();

        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 = String.valueOf(temperature);
            byte[] sendData = data.getBytes();

            DatagramPacket packet = new DatagramPacket(sendData, sendData.length, address, port);
            socket.send(packet);

            System.out.println("Sent data: " + data);

            Thread.sleep(5000); // 采样频率设定为5秒
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

(2) 服务端 (Server)

服务端采用 SpringBoot 的架构, 使用 mysql 轻量数据库实现数据的长久化存储。

主要对象类的属性如下, 分别存储更新时间和当前温度值

```

@Data
@AllArgsConstructor
public class Message {
    private LocalDateTime date;
    private double temperature;
}

```

用于与数据库交互的 Mapper 层如下

```

@Mapper
public interface MessageMapper {

    1 个用法
    @Insert("insert into message(date,temperature) values (#{date},#{temperature})")
    void insert(Message message);

    1 个用法
    @Select("select * from message")
    ArrayList<Message> getAll();
}

```

与前端交互的 Controller 层如下，用于获取数据库中的所有温度值传递给前端

```
@RestController
public class MessageController {
    @Autowired
    MessageMapper messageMapper;

    @GetMapping("/data")
    public ArrayList<Message> getAll(){
        return messageMapper.getAll();
    }
}
```

与客户端类似，采用 DatagramSocket 创建 socket 对象

使用 DatagramPacket 创建接受数据报并接受来自客户端的数据

解析后存入数据库

```
public static void main(String[] args) {
    ApplicationContext context = SpringApplication.run(ServerApplication.class, args);
    try {

        // 创建UDP套接字并绑定到指定端口
        DatagramSocket serverSocket = new DatagramSocket( port: 12345);

        System.out.println("服务器已启动，等待客户端连接...");

        // 创建接收数据的缓冲区
        byte[] receiveData = new byte[1024];

        while (true) {
            // 创建接收数据报
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

            // 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);
            Message temperature = new Message(LocalDate.now(), Double.parseDouble(message));
            messageMapper.insert(temperature);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

数据库格式如下



字段	索引	外键	触发器	选项	注释	SQL 预览					
名						类型	长度	小数点	不是 null	虚拟	键
▶ date						datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	🔑 1
temperature						varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

2.2 数据可视化

把收到的传感器的数据能够用简单的图表给展示出来(如 echart), 或者寻找相似的 web3d 组件开源的用于可视, 使用的可视化工具不做限制。

(Ps:传感器的数据发送到服务器的数据接收以及可视化时, 如果需要用到数据存储, 可以采用小型数据库或者是文件存储, 具体也不做限制。)

数据可视化使用的是 javascript 的 Chart.js 图表库

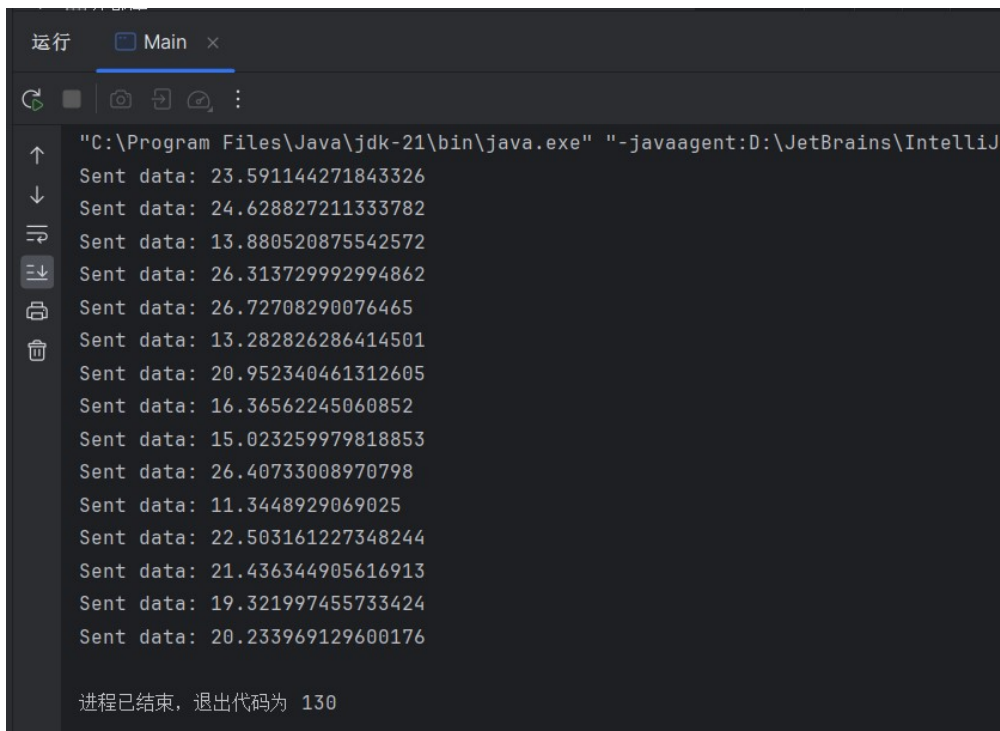
通过 fetch 方法直接从 <http://localhost:8080/data> 获得请求数据

获得数据后取得两个属性直接构建图标即可

```
9 <body>
10   <canvas id="myChart"></canvas>
11
12   <script>
13     fetch('http://localhost:8080/data')
14       .then(response => response.json())
15       .then(data => {
16         const dates = data.map(entry => entry.date);
17         const temperatures = data.map(entry => entry.temperature);
18
19         var ctx = document.getElementById('myChart').getContext('2d');
20
21         var myChart = new Chart(ctx, {
22           type: 'line',
23           data: {
24             labels: dates,
25             datasets: [{
26               label: 'Temperature',
27               data: temperatures,
28               backgroundColor: 'rgba(75, 192, 192, 0.2)',
29               borderColor: 'rgba(75, 192, 192, 1)',
30               borderWidth: 1
31             }]
32           }
33         });
34   </script>
35 </body>
```

3 实验结果

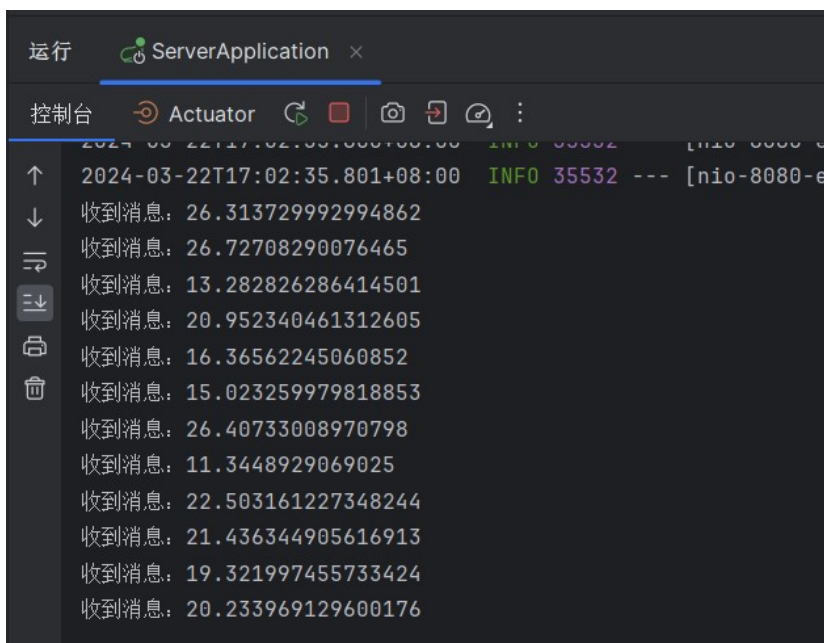
(1) 客户端发送数据



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ
Sent data: 23.591144271843326
Sent data: 24.628827211333782
Sent data: 13.880520875542572
Sent data: 26.313729992994862
Sent data: 26.72708290076465
Sent data: 13.282826286414501
Sent data: 20.952340461312605
Sent data: 16.36562245060852
Sent data: 15.023259979818853
Sent data: 26.40733008970798
Sent data: 11.3448929069025
Sent data: 22.503161227348244
Sent data: 21.436344905616913
Sent data: 19.321997455733424
Sent data: 20.233969129600176

进程已结束，退出代码为 130
```

(2) 服务端接受数据



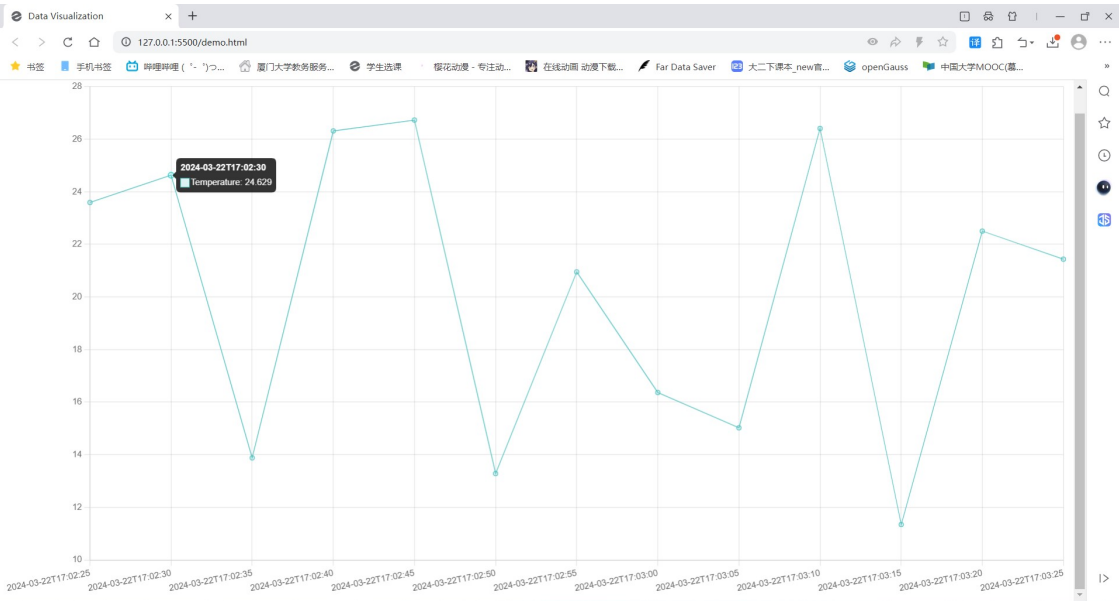
```
运行 ServerApplication x
控制台 Actuator
2024-03-22T17:02:35.801+08:00 INFO 35532 --- [nio-8080-e
收到消息: 26.313729992994862
收到消息: 26.72708290076465
收到消息: 13.282826286414501
收到消息: 20.952340461312605
收到消息: 16.36562245060852
收到消息: 15.023259979818853
收到消息: 26.40733008970798
收到消息: 11.3448929069025
收到消息: 22.503161227348244
收到消息: 21.436344905616913
收到消息: 19.321997455733424
收到消息: 20.233969129600176
```


数据库中：

WHERE

	📅 date	🌡 temperature
1	2024-03-22 17:02:25	23.591144271843326
2	2024-03-22 17:02:30	24.628827211333782
3	2024-03-22 17:02:35	13.880520875542572
4	2024-03-22 17:02:40	26.313729992994862
5	2024-03-22 17:02:45	26.72708290076465
6	2024-03-22 17:02:50	13.282826286414501
7	2024-03-22 17:02:55	20.952340461312605
8	2024-03-22 17:03:00	16.36562245060852
9	2024-03-22 17:03:05	15.023259979818853
10	2024-03-22 17:03:10	26.40733008970798
11	2024-03-22 17:03:15	11.3448929069025
12	2024-03-22 17:03:20	22.503161227348244
13	2024-03-22 17:03:25	21.436344905616913
14	2024-03-22 17:03:30	19.321997455733424
15	2024-03-22 17:03:35	20.233969129600176

(3) 可视化折线图



4 我的体会

我深刻体会到传感器在现代科技应用中的重要性和广泛应用。通过编程模拟了温度传感器，了解了如何利用可视化组件将传感器采集到的数据直观地展示出来，以便更好地理解和分析。我不仅加深了对传感器工作原理的理解，还提升了自己的编程能力和数据处理能力。同时，我也意识到了传感器在各个领域中的广泛应用，以及数据可视化在信息展示和决策支持中的重要性。