



**本科生课程报告**

**科 目： 专业综合设计 教 师： 李季 郑林江**

**专 业： 2018级物联网工程**

**上课时间： 2021 年 12月至 2022 年 1 月**

**学生报告成绩：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **学号** | **任务分工** | **成绩** | **阅卷评语** |
| **何丽莹**  **（李季）** | **20184302** | **GUI界面**  **数据收集**  **报告撰写**  **实时监测模块** |  |  |
| **陈露**  **（李季）** | **20184363** | **GUI界面**  **报告撰写**  **数据库模块**  **服务器连接数据库** |  |  |
| **李铉鹏**  **（郑林江）** | **20184409** | **客户端**  **报告撰写**  **工人轨迹模拟**  **累计接尘量模块** |  |  |

**教师 (签名)**

摘要

典型作业场所（煤矿、建筑工地、生产企业等）会产生大量粉尘，其中呼吸性粉尘对作业工人的危害极大。随着物联网技术的发展，粉尘的检测技术越来越成熟，典型作业场所粉尘精细化检测成为可能。为此，基于粉尘传感器采集的环境粉尘浓度，设计并开发一套基于物联网的作业场所粉尘危害监测预警系统。本文对粉尘传感器和作业工人移动轨迹进行模拟，并照5分钟频率（为了便于观察，这里修改为每3秒频率）随机产生的监测数据——粉尘浓度、环境温度、环境湿度和风力进行作业场所粉尘危害检测系统的设计和实现。

**关键词**：粉尘监测，预警，轨迹模拟，接尘量，实时监测

目录

[1 项目研究背景及意义 4](#_Toc92404542)

1.1 研究背景 4

[2.1系统结构设计 4](#_Toc92404544)

[2 系统设计 4](#_Toc92404543)

[2.1系统结构设计 4](#_Toc92404544)

[2.2系统主要功能 5](#_Toc92404545)

[2.3系统所用技术 6](#_Toc92404546)

[2.4系统运行环境 7](#_Toc92404547)

[2.5系统服务器功能分析 7](#_Toc92404548)

[2.6服务端 7](#_Toc92404549)

[2.7数据库 9](#_Toc92404550)

[2.8客户端 11](#_Toc92404551)

[3 关键技术 13](#_Toc92404552)

[3.1 数据生成 13](#_Toc92404553)

[3.2 工人轨迹 16](#_Toc92404554)

[3.3 累计接尘量 17](#_Toc92404555)

[3.4 监测预警 18](#_Toc92404556)

[4 作业场所粉尘危害监测预警系统（GUI界面实现） 19](#_Toc92404557)

[4.1 模拟工人移动轨迹模块 19](#_Toc92404558)

[4.2 实时监测模块 21](#_Toc92404559)

[4.3 接尘量分析模块 24](#_Toc92404560)

[5 遇到的问题及解决方法 25](#_Toc92404561)

[6 总结 26](#_Toc92404562)

[7 参考文献 27](#_Toc92404563)

# 1 项目研究背景及意义

## 1.1研究背景

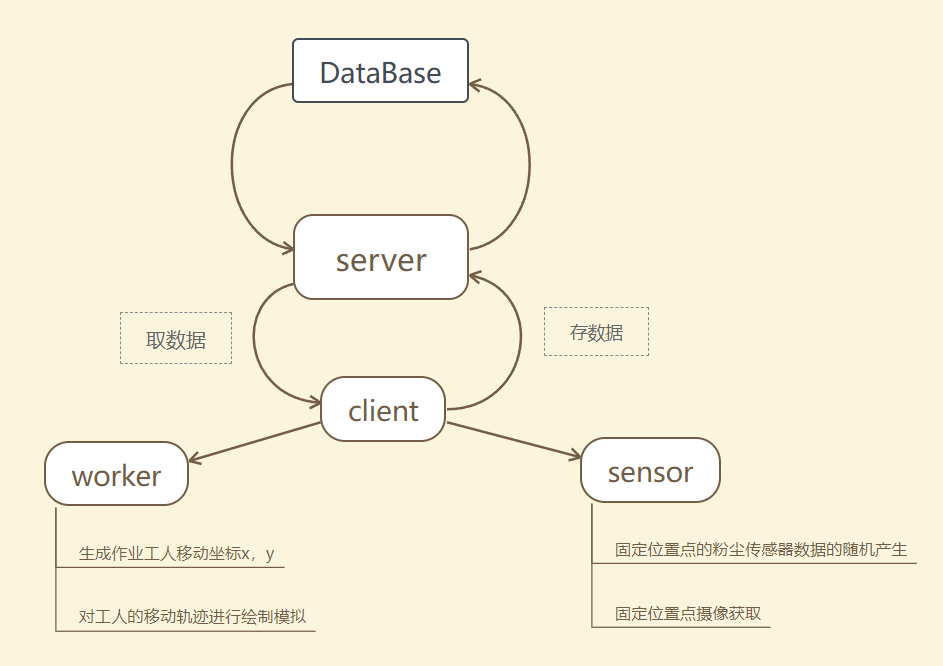
随着我国煤矿作业规模日益扩大，煤矿粉尘引发的安全隐患和作业工人职业健康问题愈加严重。煤矿粉尘是由煤炭开采的过程中产生的各种微粒组成，受微粒形状、大小、质量以及矿井内风速、湿度等多方面的影响，一部分会在矿井中形成浮尘，其中直径小于5微米的呼吸性粉尘吸入作业工人的体内后，会在工人的肺里面沉积，并对其身体健康造成很严重的危害。

## 1.2研究意义

近年来，随着物联网与粉尘分离技术的快速发展，基于物联网的煤矿粉尘浓度检测技术和监测装备逐渐出现。煤矿粉尘检测装置为粉尘实时数据采集与监控提供了重要的技术途径。在此基础上，如何有效连续采集粉尘浓度状态，并对分布式采集的粉尘数据进行管理与应用，已成为煤矿粉尘防治关注的焦点。

# 2 系统设计

## 2.1 系统结构设计



数据库、服务器、客户端之间关系：

服务端的作用是保存、处理、分发数据，客户端的作用是请求数据并显示出，数据库是通过服务端访问的，访问数据库后处理并进行包装，通过socket发送给客户端。客户端收到数据后进行解释并提取所需的数据并显示。

## 2.2系统主要功能

2.2.1 粉尘传感器模拟

实现作业场所固定位置点的粉尘传感器对于粉尘浓度、温度、湿度、风力数据的模拟，每隔一定时间随机产生对应的检测参数，同时可以采集现场信息。

2.2.2 作业工人轨迹模拟

工人在粉尘环境下进行作业，在后端由随机函数在合理范围内产生工人在一段时间内作业的行动轨迹，检测工人所在路径的粉尘浓度

2.2.3 作业场所粉尘危害监测预警系统

传感器客户端和工人客户端完成对数据的采集存储，计算出某工人的累计接尘量，对工人达到一定接尘量做出预警。

## 2.3系统所用技术

2.3.1 JAVA技术

本项目采用java的客户端与服务器通过网络协议进行通信，同时利用java自己的GUI对数据分析结果进行可视化，在基本的Java应用编程接口中有个网络应用编程接口（java net）,提供了用于网络应用编程的类库，包括URL,Connection, Socket, ServerSocket等接口。

2.3.2 TCP/IP协议

TCP/IP又称传输控制协议，由网络层的IP协议和传输层的TCP协议组成，TCP/IP是一个4层的分层体系结构，我们的客户端和服务器就是基于TCP/IP的协议基础上进行彼此的数据交换。

2.3.3 Mysql数据库

Mysql是一种关系型数据库管理系统，使用结构化查询语言（sql）进行数据库管理，并且在本项目中利用Navicat可视化应用对数据库进行管理，便于建表和进行其它操作，在本项目粉尘危害监测预警系统中使用这种数据库。

## 2.4系统运行环境

硬件环境:个人计算机

软件环境：IntelliJ IDEA, JDK 13 ,Mysql8.0,Windows 10

## 2.5系统服务器功能分析

服务器的主要功能将客户端发送过来的信息，进行存储进数据库操作，并且调用可视化界面，在可视化界面上显示我们的分析结果。在服务器端主要有一个界面，这个界面是该系统的主页面。

## 2.6服务端

建立服务器ServerSocket，监听12345端口，监听客户端连接，调用accept()方法，accept方法是一个阻塞的方法，会阻塞当前线程，对于阻塞的代码，需要放置到独立线程，因此我们监听连接的线程类ServerThread。

1. public class TcpServer {
3. public static void main(String[] args) throws IOException{
4. ServerSocket serSoc = new ServerSocket(12345); *// 监听指定端口*
5. System.out.println("-----Server Running-----");
6. DustSystemGUI serverGUI = new DustSystemGUI();
7. serverGUI.setVisible(true);
8. while(true) {
9. System.out.println("waiting clients");
10. *//监听*
11. Socket socket = serSoc.accept();
12. System.out.println("client: " + socket.getRemoteSocketAddress() + "connected");
13. Thread t = new ServerThread(socket, serverGUI);
14. t.start();
15. }
16. }
17. }

在ServerSocket类中，每当有一个客户端连接到当前的serversocket就会返回一个新的socket对象，所以当有多个的时候，就要创建一个while循环来监听来自客户端的链接。

1. //接收传感器客户端传过来的对象数据
2. while(!Thread.currentThread().isInterrupted()) {
3. //这行必须放try里面才能收到实时数据
4. ObjectInputStream ois = new ObjectInputStream(this.socket.getInputStream());
6. Object obj = ois.readObject();
7. if(obj instanceof SensorClient) {
8. storeToDatabase( (SensorClient)obj );
9. } else if(obj instanceof SensorData) {
10. int id = ois.readInt();
11. storeToDatabase( id, (SensorData)obj );
13. } else if(obj instanceof WorkerClient)
14. {
15. storeToDatabase( (WorkerClient)obj );
16. }else{
17. System.out.println("data error");
18. }
19. }

在ServerSocket类中，我们建立通信管理，搭起数据库与客户端的桥梁。我们通过实例化Database类，连接数据库，将客户端上传的数据放在数据库。这里数据库使用mysql。

将传感器历史数据传到数据库，这里我们需要获取客户端的监测点位置编号，用于上传到对应的传感器数据表中。

1. //将传感器历史数据传到数据库
2. private void storeToDatabase(SensorClient sc) {
3. //加入传感器客户端列表
4. sClients.add(sc);
5. // 获取id
6. int id=sc.getId();
7. System.out.println("you are using sensor"+id);
8. // 清空即将要插入的表格 只能在历史数据这里清空 不能在实时数据那里清空
9. myDatabase.clearSensor(id);
10. for( SensorData sd : sc.getHistoryData() ) {
11. // 插入历史数据
12. myDatabase.insertSensor(id, sd);
13. }
14. }

将传感器客户端的实时数据存储到数据库，并对当前数据进行判断，用来实现实时预警模块。

1. private void storeToDatabase(int id, SensorData sData) {
2. // 插入实时数据
3. myDatabase.insertSensor(id, sData);
4. if(sData.getDensity()>=9.5){
5. System.out.println("粉尘浓度超标，请注意！！！");
6. }
7. // gui改变显示的实时数据
8. serverGUI.change\_realtime\_data(id, sData);
9. }

## 2.7数据库

我们小组通过传感器客户端将日期，时间，粉尘浓度，温度，湿度，以及风力上传给服务器，工人客户端获取后端的工人作业日期与时间，将工人实时的坐标（x,y）上传服务器，在服务器将上传的数据存储到数据库以便进行后续分析

2.7.1 数据库字段规定

（1）Sensor表（传感器）

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 小数点 |
| date | date | 0 | 0 |
| time | Time | 0 | 0 |
| density | Double | 255 | 2 |
| temperature | Double | 255 | 1 |
| humidity | Int | 0 | 0 |
| wind | int | 0 | 0 |

（2）Workder表（工人）

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 小数点 |
| Date | date | 0 | 0 |
| Time | time | 0 | 0 |
| PosX | int | 0 | 0 |
| posX | int | 0 | 0 |

2.7.2 建立Database类

（1）Database四个属性用于java连接数据库

1. static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";
2. static final String DB\_URL = "jdbc:mysql://localhost:3306/javatest?useSSL=false&serverTimezone=UTC";
3. *// 数据库的用户名与密码*
4. static final String USER = "root";
5. static final String PASSWORD = "xxxxxxx";

（2）主要成员函数

清空sensor表，在每次客户端将后台数据传入服务器端时，在服务器端向数据库上传历史数据时，将刷新数据库内容，通过调用clearsensor（id）将该表的内容清空，导入新的数据

1. updateStr = "delete from sensor"+sensorNum;
2. //通过sql删除语言 delete frome table进行删除
3. //删除成功或者失败都会打印信息（success/error）
4. PreparedStatement st;
5. try {
6. st = conn.get().prepareStatement(updateStr);
7. int i=st.executeUpdate();
8. if(i>0){
9. System.out.println("清空sensor成功！");
10. }else{
11. System.out.println("clear sensor error！");
12. }

（3）清空worker表，原理同sensor表类似

* 1. updateStr = "delete from worker";

（4）向数据库sensor表中插入数据条目，同样利用sql插入语句

updateStr = "INSERT INTO sensor"+sensorNum+" VALUES(?,?,?,?,?,?)";

其中，插入参数包括上面数据sensor表的字段相对应

1. st.setObject( 1 , dateStr);
2. st.setObject( 2 , timeStr);
3. st.setObject( 3 , sensorObj.getDensity());
4. st.setObject( 4 , sensorObj.getTemperature());
5. st.setObject( 5 , sensorObj.getHumidity());
6. st.setObject( 6 , sensorObj.getWind());

（5）向worker表中插入数据，包括日期，时间，坐标(x,y)，代码与4类似。

String updateStr =  "INSERT INTO worker VALUES(?,?,?,?)" ;

（6）打印sensor表中的内容

新建一个Sensor类的集合allSensorData,将每一个传感器的数据添加进这个集合里面。

1. ArrayList<SensorData> allSensorData = new ArrayList<SensorData>();
2. AtomicReference<Connection> connection = new AtomicReference<>(accessDB());
3. String queryStr = null;
4. queryStr = "select \* from sensor"+sensorNum;

（7）打印worker表中的内容

新建一个路径向量PathVertex的类集合allPathVertex，将工人表中的数据存入集合中。

1. ArrayList<PathVertex> allPathVertex=new ArrayList<PathVertex>();
2. AtomicReference<Connection> connection = new AtomicReference<>(accessDB());
3. String queryStr = "select \* from worker";

（8）获取某个日期的sensor

在本项目基于粉尘的监测预警系统中，需要检测一段时间内的粉尘浓度以及其他检测环境因素，用于绘图，故考虑某一个日期的传感器数据，与前面的获取全部数据信息差别在于sql语句中多了过滤条件，加入了用户选择的日期

1. queryStr = "select \* from sensor"+sensorNum+" where Date=(?)";

（9）获取某个日期的worker

需要绘制某一个日期段的工人轨迹，以及检测他的累计接尘量、

1. String queryStr = "select \* from worker where Date=(?)";

## 2.8客户端

（1）粉尘传感器模拟（Client）

实现作业场所固定位置点的粉尘传感器相关数据的随机产生。按照5分钟频率随机产生的监测参数包括：粉尘浓度mg/m3、环境温度℃、环境湿度和其他信息（根据需要自由发挥）；同时，实时上传固定位置点的视频图像。

Class SensorClient

|  |  |  |
| --- | --- | --- |
| 成员变量 | 类型 | 含义 |
| id | int | 传感器编号 |
| xLoc | int | 传感器位置横坐标 |
| yLoc | int | 传感器位置纵坐标 |
| ssData | ArrayList<SensorData> | 传感器的数据库信息 |

|  |  |
| --- | --- |
| 成员函数 | 含义（作用） |
| getId() | 获取传感器序号 |
| openSensorCamera() | 打开传感器所在位置的摄像头 |
| geneSensors(int days) | 随机生成days天的传感器数据 |
| uploadRealTimeData(OutputStream os, int initDelay, int delay) | 周期性生成传感器数据并上传至服务器 |
| uploadData(OutputStream os) | 将传感器数据上传至服务器 |
| getX() | 获取传感器位置的横坐标 |
| getY() | 获取传感器位置的纵坐标 |

（2）作业工人轨迹模拟（Client）

在作业场所，作业工人在粉尘环境下进行作业，模拟作业工人的移动轨迹，同时可上传作视频。

Class Workerclient

|  |  |  |
| --- | --- | --- |
| 成员变量 | 类型 | 含义 |
| wName | String | 工人姓名 |
| workDays | Int | 工作天数 |
| wPaths | ArrayList<WorkerPath> | 工人路径向量 |
| date | LocalDate | 当时时间 |

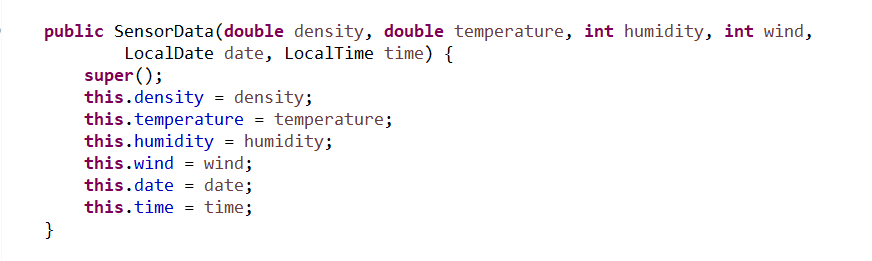
|  |  |
| --- | --- |
| 成员函数 | 含义(作用) |
| generatePath() | 产生days天的工人路径 |
| uploadWorkerPaths(OuputStream os) | 上传工人路径数据 |
| getPathsArray() | 获取工人轨迹 |
| toString() | 打印工人的轨迹信息 |
| drawPathOfDate | 根据日期绘制轨迹 |

# 3 关键技术

## 3.1 数据生成

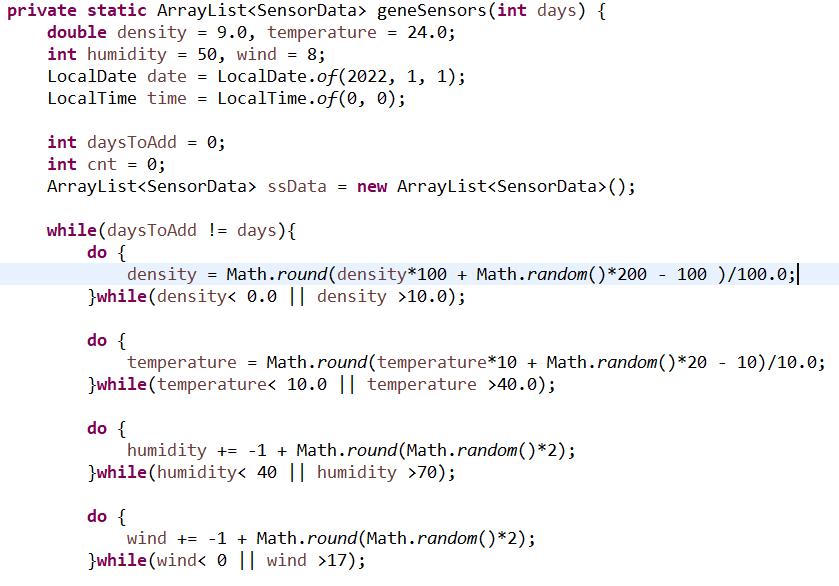
3.1.1 粉尘传感器的数据的产生方式

我们将粉尘传感器定义作一个类，同时又单独定义了一个类来专门放置粉尘的各种数据（如日期、粉尘浓度、温度、湿度，风力等），方便与服务器进行通信，具体如下图所示：

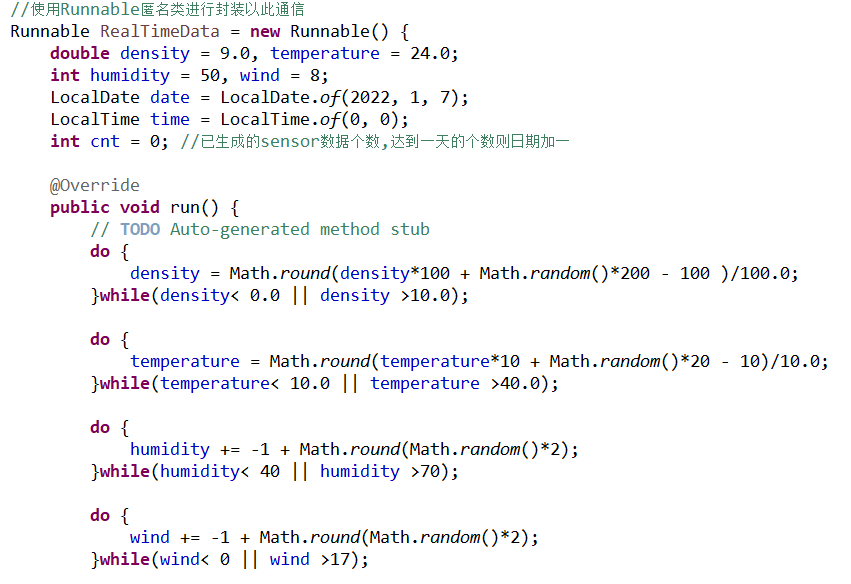


对于数据的产生采用以下的设计思路，每次开启传感器时先随机产生六天的数据（2022-1-1至2022-1-6）作为历史数据，然后设置从下一天（2022-1-7）起开始实时产生数据，由于在网上爬取的数据很难满足我们对于系统的要求，于是采用随机数的方式生成，但是也不是毫无规律的，我们先设置了一个平均粉尘浓度，所有随机产生的数据都是根据在这个平均浓度在一定的范围内进行波动的，具体实现如下图所示：

生成6天的历史数据：

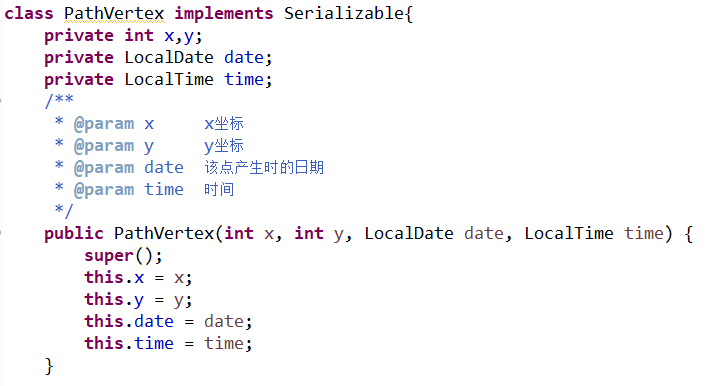


实时生成的数据也采用的是这样的方法，不过外部加了一个匿名类进行与服务器实时通信：

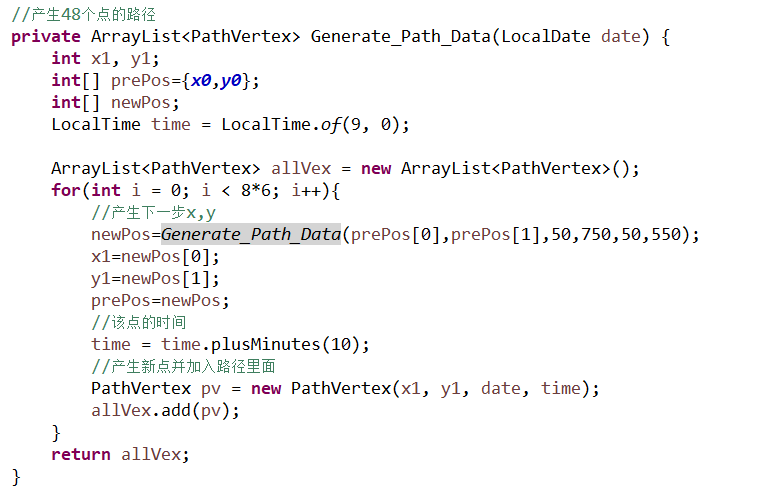


3.1.2 工人数据的产生方式

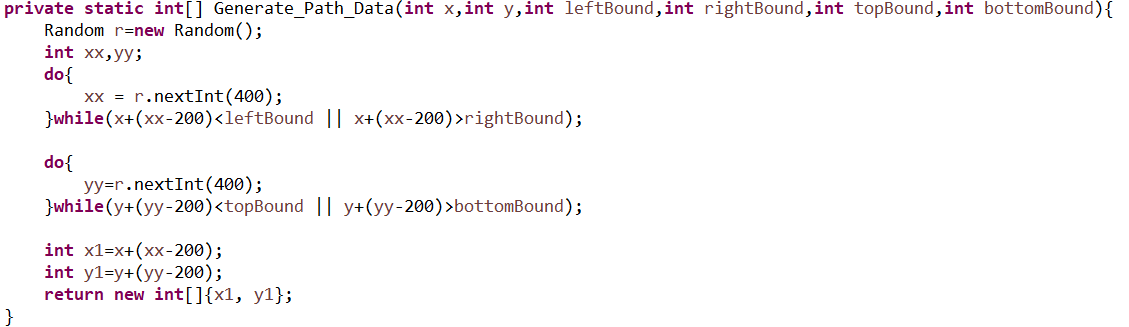
工人类与传感器类的设计类似，也是将工人设计为一个类，同时对工人的数据单独封装为一个类（包含了日期和工人的位置），如下图所示：



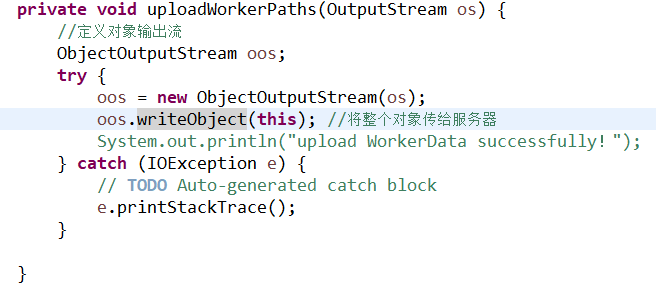
同时工人也是有历史数据的，我们希望在加载工人的client时能够同时上传工人近六天的位置信息，工人路线的设计如下图所示，我们采用递归的方式，每次产生一个点，将它加入当天的路径变量中，一条路径一共有48个点，每个点都包含了工人的位置和时间。



Generate\_Path\_Data函数的实现如下，每次工人移动的范围为±200px：



最后将整个对象上传到服务器：



3.1.3 数据之间的流通

传感器端和工人端只做了生成数据和上传数据的功能，他们将数据上传到服务器，服务器收到这些数据之后存储到数据库中，在数据库中实现对数据的分析处理，并调用GUI对数据进行显示。

## 3.2 工人轨迹

由于我们不仅需要记录工人的信息，还需要打印工人的路线图，我们的想法是按照日期来画路线图，因为我们对于工人信息的生成都是按照时间来的，同时路径其实也已经生成好了，我们只需要找到那天路径中的所有点（共48个），然后对他们的横纵坐标稍做处理就可以画出我们需要的工人路径图了，我们将它在工人端实现了，对于一个点我们用十字来表示，对于路径我们用直线来表示：

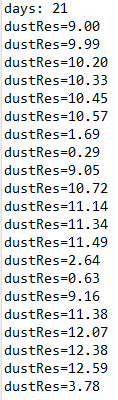


## 3.3 累计接尘量

对于工人的累计接尘量，我们先做如下的设想：工人一周上班5天，周末休息，人体自身具有对粉尘的净化功能，但是对粉尘的净化能力会降低，于是我们可以设计出如下计算累计接尘量的方法。假设每天吸入的粉尘浓度为9.00：

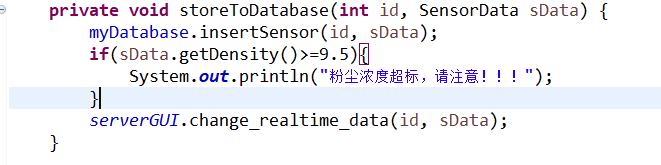


我们可以查看工人在粉尘浓度为9.00的环境中工作了三周后的接尘量：



## 3.4 监测预警

由于我们已经有了对粉尘数据的实时上传，我们想要对于粉尘浓度进行预警，只需要在server收到数据时进行一下判断，看它的浓度是否超过了我们设定的值（9.5），就可以实现实时监测预警：



如图所示，我们只需要在服务器收到数据想上传到数据库时进行判断即可，效果图如下：



# 4 作业场所粉尘危害监测预警系统（GUI界面实现）

在完成Client的粉尘传感器、作业工人轨迹数据的采集与存储，接下来我们对这些数据进行了一个可视化管理，实现作业场所粉尘危害监控、作业工人累积接尘量监控、作业工人接尘量预警、作业场所粉尘预警等功能。

## 4.1 模拟工人移动轨迹模块

查询一段时间的接尘量，调用日历面板，获取用户选择日期，点击确认按钮，响应事件监听函数，调用choose\_date\_submit()方法进行处理，在此函数中首先获得用户的输入日期，并到数据库中获取工人的数据信息。

1. //得到用户输入的日期
2. queryDate=input\_date\_box.getText();
3. //获取数据库工人信息
4. ArrayList<PathVertex> apv=myDatabase.getWorkerByDate(queryDate);

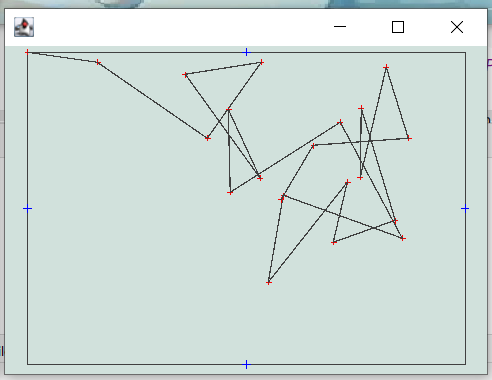


点击确认后，我们从数据库中拿到了某位工人在进行作业时的移动轨迹信息，点击模拟路径按钮，进行轨迹模拟绘制。

1. //显示数据的表格
2. String[] columnNames = { "日期", "时间","X","Y"};//表头
3. String[][] tableValues=new String[apv.size()][4];//表中数据：来自数据库
4. for(int i=0;i<apv.size();i++) {
5. PathVertex pv=apv.get(i);
6. tableValues[i][0]=pv.getDate().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
7. tableValues[i][1]=pv.getTime().format(DateTimeFormatter.ofPattern("HH:mm"));
8. tableValues[i][2]=Integer.toString(pv.getX());
9. tableValues[i][3]=Integer.toString(pv.getY());
10. }
11. show\_data\_table\_worker(tableValues,columnNames,queryDate);



在点击了模拟路径后，通过读取到的工人轨迹坐标矩阵，使用Graphics绘制工人轨迹。



## 4.2 实时监测模块

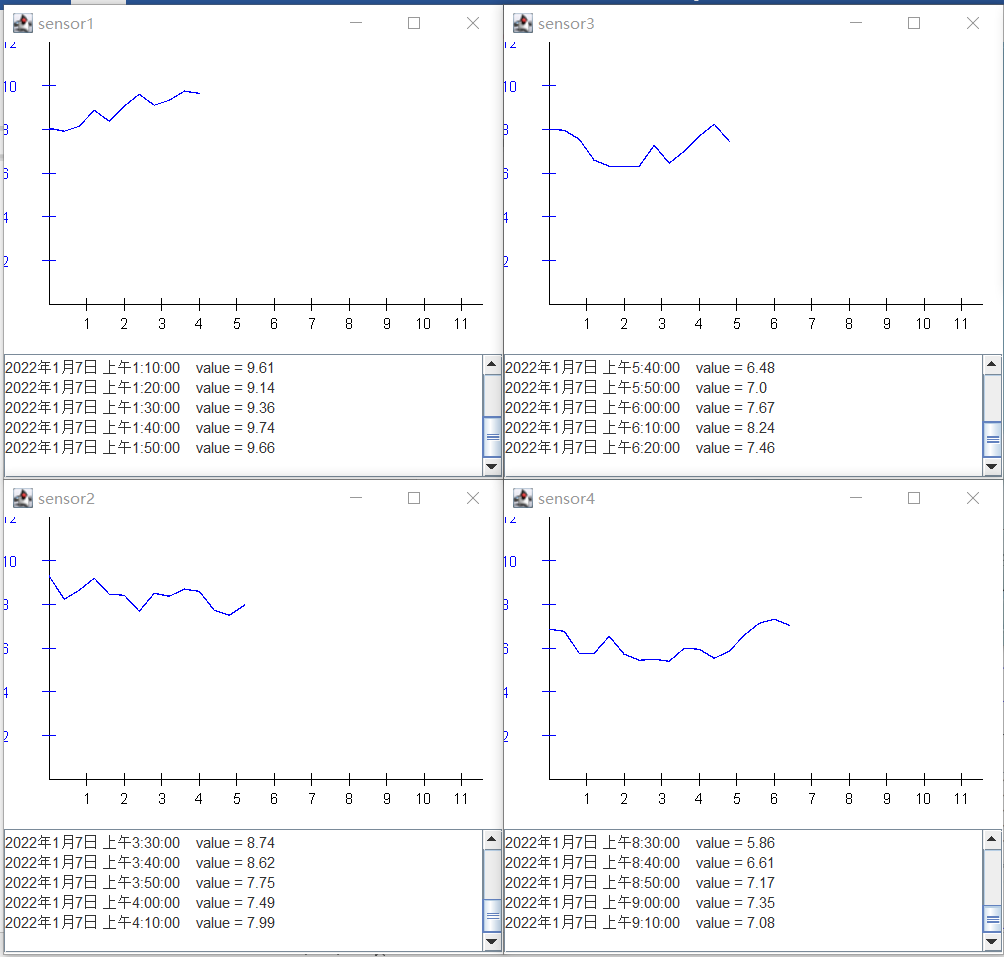
这里需显示粉尘浓度，并对其进行预警值判断，这里我们设置的预警值为8，当当前的粉尘浓度大于或等于8时，显示出来的实时浓度值用红色标记。

1. double density = sData.getDensity();
2. if(realtime\_data\_table == null) {
3. return;
4. }
5. if(density>=8){
6. DefaultTableCellRenderer dtc=new DefaultTableCellRenderer();
7. dtc.setForeground(new Color(191,27,0));
8. dtc.setHorizontalAlignment(JLabel.CENTER);
9. realtime\_data\_table.getColumnModel().getColumn(id-1).setCellRenderer(dtc);
10. realtime\_data\_table.setDefaultRenderer(Object.class,dtc);
12. }
13. else {
14. DefaultTableCellRenderer dtc=new DefaultTableCellRenderer();
15. dtc.setForeground(Color.BLACK);
16. dtc.setHorizontalAlignment(JLabel.CENTER);
17. realtime\_data\_table.getColumnModel().getColumn(id-1).setCellRenderer(dtc);
18. realtime\_data\_table.setDefaultRenderer(Object.class,dtc);
19. }

此外，我们添加选择传感器序号来查看不同的监测点的粉尘浓度变化情况，对不同的监测点，我们进行一个唯一的id编号，并通过getSensorId方法获取当前选择的id值。

1. public int getSensorId()
2. {
3. int id = choose\_sensorNumber\_box.getSelectedIndex();
4. if (id == 0)
5. {
6. id = 1;
7. }
8. return id;
9. }





当我们点击camera按钮后，可以查看监测点的现场视频。

1. public void openCamera() {
2. Webcam webcam = Webcam.getDefault();
3. webcam.setViewSize(WebcamResolution.VGA.getSize());
4. WebcamPanel panel = new WebcamPanel(webcam);
5. panel.setFPSDisplayed(true);
6. panel.setDisplayDebugInfo(true);
7. panel.setImageSizeDisplayed(true);
8. panel.setMirrored(true);
10. camera\_window=new JFrame("监控摄像头");
11. camera\_window.setLocation(820,20);
12. camera\_window.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);
13. camera\_window.add(panel);
14. camera\_window.setResizable(true);
15. camera\_window.pack();
16. camera\_window.setVisible(true);
17. camera\_window.validate();
18. }



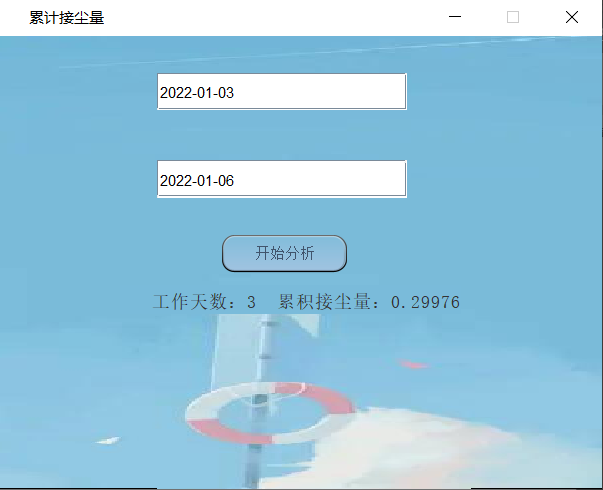
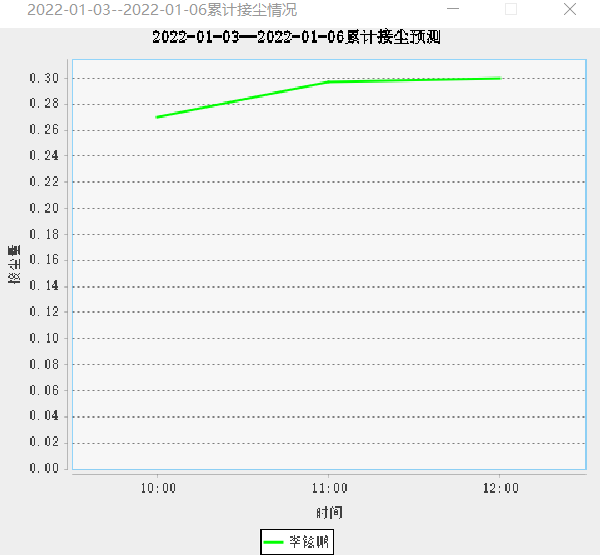
## 4.3 接尘量分析模块



如上图所示，查询一段时间的接尘量，调用日历面板，获取到日期（用户选择）。点击开始分析，响应事件监听函数，调用analyse\_sum\_dust进行处理，在此函数中首先获得用户的输入日期。

1. startDate=input\_date\_box.getText();
2. endDate=input\_date\_box2.getText();

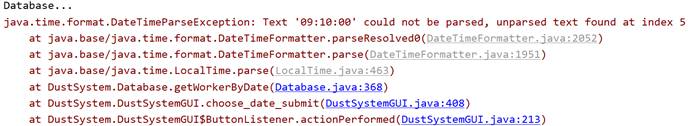
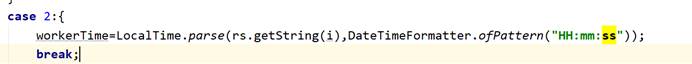
计算间隔天数，同时累加这几天的接尘量，通过java的GUI画布将接尘量通过折线图绘制出来，并且将累计接尘量的文字信息显示到原来的面板上。



界面部分代码：

1. JLabel label1= new  JLabel( "工作天数：" +Integer.toString(days)+ "累计接尘量：" +Double.toString(lastAccuDust));    *//创建标签*
2. label1.setFont( new  Font( "宋体" ,Font.PLAIN, 15 ));
3. jp.add(label1);
4. dust\_analyse\_window.add(jp);
5. dust\_analyse\_window.validate();
6. draw\_sum\_dust\_graph2(sum\_tableValues);

# 5 遇到的问题及解决方法

（1）数据库数据时，报错显示数据库字段格式不匹配  
  
经过检查，发现数据库中的时间格式是时-分-秒的格式，而代码中只给了时-分的格式，修改代码部分，成功从数据库中得到数据。  


# 6 总结

第一次接触到比较完整的物联网项目，对于整个项目框架花了较长时间来决定，一开始我们使用python进行实时数据的爬取，考虑利用python自带的库进行数据分析，但是在实时获取并异步刷新可视化界面的时候遇到了瓶颈，经过几天的探索之后，决定最后用java来实现，并且用GUI实时显示，java对于Mysql的连接使用接口很方便，便于我们的服务器端从数据库中进行存取数据，整个项目中主要分为数据的收集分析，后端对于一些参量的随机生成，还要保证生成数据的合理性，以及通过几方因素对接尘量进行数据统计，并且包括GUI的事件响应，在参阅了一些相关文献后，大多数对于该系统的预警是通过神经网络进行训练得到预测结果，但是因为能力和时间有限，我们组简化了预警的因素，只通过空气中的粉尘浓度，以及工人的累计接尘量达到一定阈值进行警报，基于物联网的粉尘监测预警平台还有需要改进的地方，包括对于累计接尘量计算模型的优化，对于随机生成数据方面也许可以通过实地考察更优化合理性，以及我们的预警精确效果有待提高。

# 7 参考文献

[1] 博客园. Socket 基础解析使用ServerSocket建立聊天服务器[EB/OL]. [20220102]. https://www.cnblogs.com/itliucheng/p/4974854.html.

[2] FwbZzzzzz. 【MySQL】客户端、服务器、数据库、表的关系[EB/OL]. [20200102]. https://blog.csdn.net/qq\_38534524/article/details/98211176?spm=1001.2101.3001.6650.5&utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~BlogCommendFromBaidu~default-5.no\_search\_link&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2~default~BlogCommendFromBaidu~default-5.no\_search\_link&utm\_relevant\_index=10.

[3] 小马达-. 数据库、服务器、客户端之间关系[EB/OL]. [20200102]. https://blog.csdn.net/linzhongqingniao/article/details/86436088.

[4] 周龙辉. 基于物联网的煤矿粉尘职业危害监测及预警技术研究[D]. 重庆大学计算机, 2018.

[5] 王杰, 赵政, 李德文, 等. 矿山作业人员呼吸尘累积接尘量计算方法及测试系统[D]. 中煤科工集团重庆研究院有限公司, 2019.

[6] 付妍文. 基于物联网的机房粉尘监控系统[D]. 深圳企管企业服务有限公司, 2017.