**仓库温湿度环境监测系统**

|  |
| --- |
| 本 章 导 读 |
| 本章主要从环境温湿度的监测角度出发，以仓库或中小型库房存储环境监控为背景，通过采用嵌入式开发技术，监测、监控环境温度和湿度两大指标为基础，通过阿里云服务器在网页上实时远程监测仓库温湿度，并且如果烟雾浓度超过警戒值，报警灯快速闪烁，提醒管理员迅速救火报警。着重向读者展示的是温湿度传感器、烟雾浓度传感器数据的采集、传输、显示的基本实现，并在一个现实的应用场景中来完成此次项目  本章将学习以下内容：   * 了解温湿度传感器和烟雾传感器的基本原理及硬件设计方案 * 掌握无线传感网的搭建及数据传输方法 * 掌握简单的数据库设计及连接方法 * 掌握通过Java Web及阿里云远程获取数据并显示的方法 |

1.1、项目简介

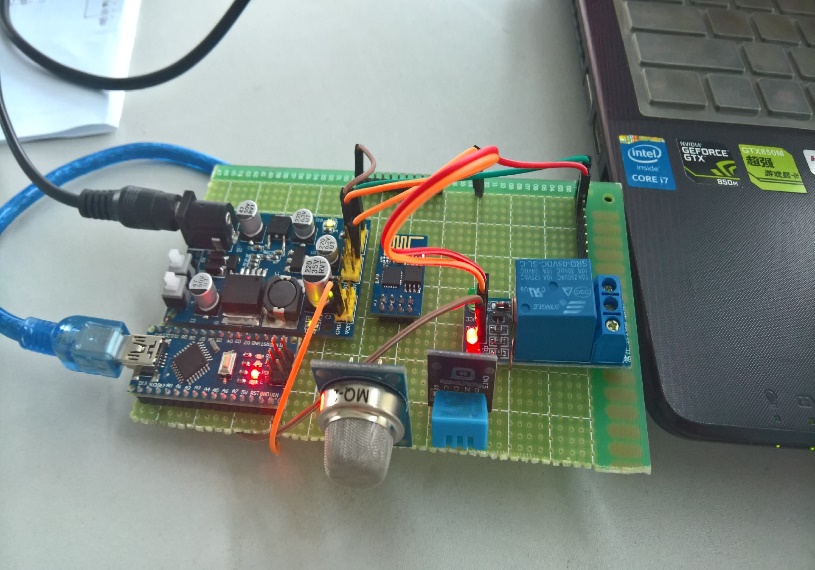
1.1.1背景介绍

人们的日常生活和周围环境的温湿度息息相关，石油、化工、航天、制药、档案保管、粮食存储等领域对温湿度也有着较高的要求。因此，对温湿度的监测和控制已成为生产过程中非常重要的技术。随着我国经济的快速稳定发展，在库房或仓库管理中不管是针对食品还是其他物资的存放环境都提出了非常高的要求。存储条件或环境的好坏直接影响到存储质量与环境

安全。因此，库房或仓库环境的实时监控与智能化管理对社会的发展有着较为重要的现实意义。大气环境测量的要素主要包括：温度、湿度、风速、风向、雨量、辐射等。环境温度与湿度通常是衡量存储环境的一个重要参数。通常大型的仓库都具有一套复杂的环境监测系统，这类监控系统主要采用计算机复杂控制系统，存在系统庞大、成本高等问题，不利于推广和应用。针对当前各类中小型仓库或库房存储环境，如何通过监测环境温度和湿度这两个重要的指标，并在第一时间获悉火灾的发生，设计一款低成本、高效能、使用便捷的监测系统，以实现对库房或仓库的存储环境的监控与管理是此次项目研究工作的目标。

1.2 项目设计

1.2.1运行流程

运用嵌入式开发技术，通过温湿度传感器监测、监控环境温度和湿度，通过阿里云服务器在网页上实时远程监测仓库温湿度，并且如果烟雾浓度超过警戒值，报警灯快速闪烁，提醒管理员迅速救火报警。

1.2.2系统功能运行流程



在货品仓库放置事先设置好的温湿度传感器以及烟雾传感器，获取温湿度信息，通过WiFi模块连接万维网上传取得的数据信息，将信息通过Socket套接字方法传送到云服务器，云服务器获取信息后将数据存储到MySQL数据库中，最后通过servlet/jsp技术在网页界面中展示温湿度信息。

1.2.2数据流程详解

(1)数据获取



(2)后台数据抽取

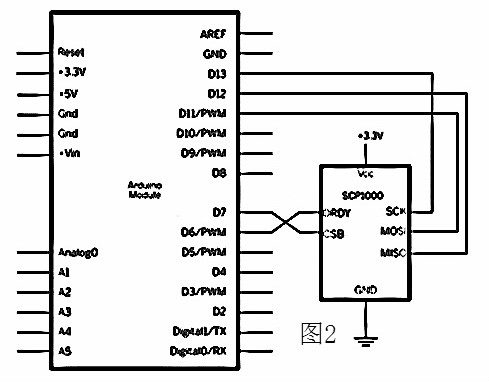
(3)前台展示

1.3项目开发

1.3.1材料准备

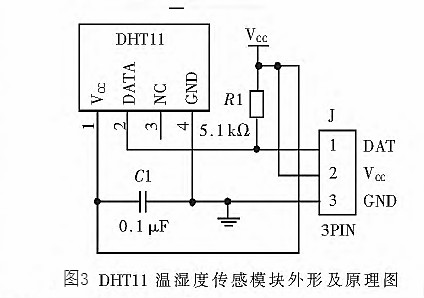
(1) Arduino UNO

Arduino UNO是Arduino USB接口系列的最新版本，原理图如（图2）所示，作为Arduino平台的参考标准模板。UNO的处理器核心是ATmega328，同时具有14路数字输入/输出口，6路模拟输入，一个16MHz晶体振荡器，一个USB口，一个电源插座，一个ICSP header和一个复位按钮。



(2) DHTll数字温湿度传感器

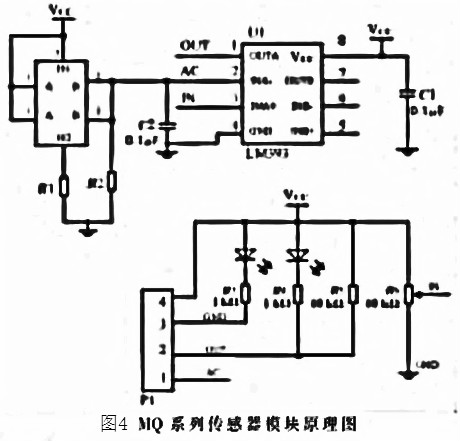
DHTll数字温湿度传感器（图3）是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包含一个电容式感湿元件和一个NTC测温元件，并与高性能8位单片机相连接。因此该产品具有品质卓越、响应超快、抗干扰能力强、性价比极高等优点。每个DHTl1都在极为精确的湿度校验室进行校准。校准系数以程序的形式储存的OTP内在中，传感器内部在检测信号的处理过程中要调用这些校准系数。



(3) MQ2 气敏传感器

气敏传感器（图4）是一种能够检测气体的特定成分或者浓度，然后将其转换成电信号的电子装置。MQ2 气体传感器所使用的材料二氧化锡，是一种重要的半导体传感器材料，也是第一个作为商业用途的透明导电材料。当MQ2 气体传感器所处的环境中存在可燃性气体，因二氧化锡晶体缺陷，即O 空位提供的电子产生载流子，会随可燃气体的浓度增加导电率增强。此时，使用简易的电路就可以将导电率直接转换成该气体浓度或比重的相关电信号输出。MQ2 传感器对于各种常见可燃气体如丙烷、甲烷、一氧化碳等灵敏度较高，对气体可燃蒸汽的检测也较为灵敏，其本身材料成本低，稳定性强，可存在于各种环境中,通过3P传感器连接线直接连接到arduino传感器拓展板，和蜂鸣器模块结合，实现烟雾报警功能。





1.3.2开发环境搭建

操作系统环境：Windows;

数据库选择：MySql;

开发工具：MyEclipse10;

本地调试服务器：Tomcat;

应用服务器：阿里云

开发语言：Java

1.3.3开发过程

1.3.3.1 DHT11温湿度模块



(1)模块描述

1 可以检测周围环境的湿度和温度  
2 传感器采用DHT11  
3 湿度测量范围：20%-95%（0度-50度范围）湿度测量误差：+-5%  
4 温度测量范围：0度-50度 温度测量误差：+-2度  
4 工作电压3.3V-5V  
5 输出形式 数字输出           
6 设有固定螺栓孔，方便安装  
7 小板PCB尺寸：3.2cm \* 1.4cm  
8 电源指示灯（红色）  
9 每套重量约为8g

(2)模块接口说明（3线制）

1 VCC 外接3.3V-5V

2 GND 外接GND

3 DO 小板开关数字量输出接口 接单片机IO口

(3)接口说明

建议连接线长度短于20米时用5K上拉电阻,大于20米时根据实际情况使用合适的上拉电阻

(4)电源引脚

DHT11的供电电压为 3－5.5V。传感器上电后，要等待 1s 以越过不稳定状态在此期间无需发送任何指令。电源引脚（VDD，GND）之间可增加一个100nF 的电容，用以去耦滤波。

(5)测量分辨率

测量分辨率分别为 8bit（温度）、8bit（湿度）。

(6)电气特性

VDD=5V，T = 25℃，除非特殊标注

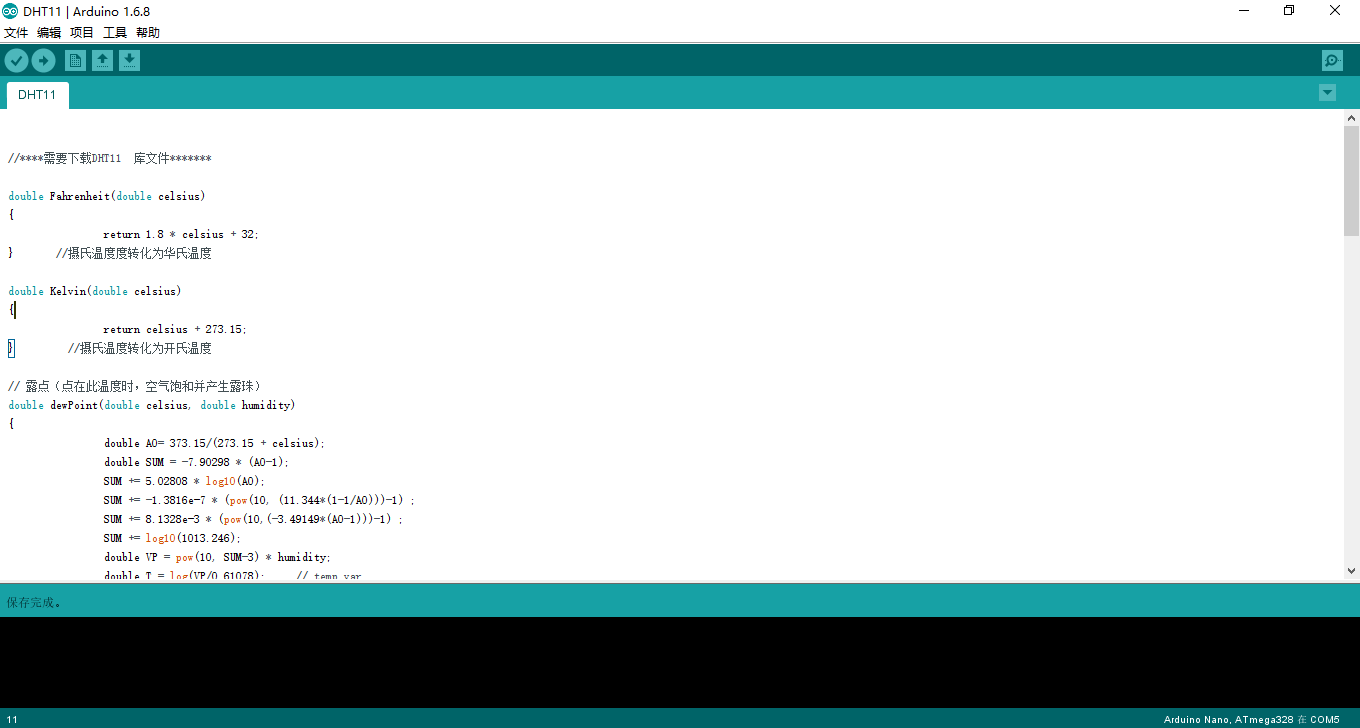
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **条件** | **min** | **typ** | **max** | **单位** |
| **供电** | DC | 3 | 5 | 5.5 | V |
| **供电电流** | 测量 | 0.5 |  | 2.5 | mA |
| 平均 | 0.2 |  | 1 | mA |
| 待机 | 100 |  | 150 | uA |
| **采样周期** | 秒 | 1 |  |  | 次 |

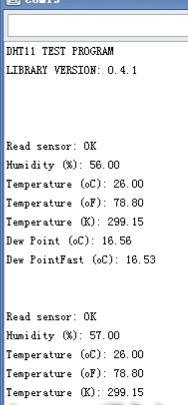
注:采样周期间隔不得低于1秒钟。

(7) DHT11引脚说明

|  |  |  |
| --- | --- | --- |
| Pin | 名称 | 注释 |
| 1 | VDD | 供电 3－5.5VDC |
| 2 | DATA | 串行数据，单总线 |
| 3 | NC | 空脚，请悬空 |
| 4 | GND | 接地，电源负极 |

(8)注意事项：气体的相对湿度，在很大程度上依赖于温度。因此在测量湿度时，应尽可能保证湿度传感器在同一温度下工作。如果与释放热量的电子元件共用一个印刷线路板，在安装时应尽可能将DHT11远离电子元件，并安装在热源下方，同时保持外壳的良好通风。长期保存条件：温度10－40℃，湿度60％以下。

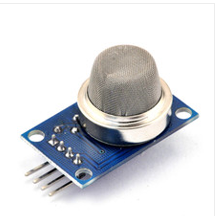




(9)源代码

|  |  |
| --- | --- |
|  | //\*\*\*\*需要下载DHT11 库文件\*\*\*\*\*\*\*  double Fahrenheit(double celsius)  {          return 1.8 \* celsius + 32;  }    //摄氏温度度转化为华氏温度    double Kelvin(double celsius)  {          return celsius + 273.15;  }     //摄氏温度转化为开氏温度    // 露点（点在此温度时，空气饱和并产生露珠）  double dewPoint(double celsius, double humidity)  {          double A0= 373.15/(273.15 + celsius);          double SUM = -7.90298 \* (A0-1);          SUM += 5.02808 \* log10(A0);          SUM += -1.3816e-7 \* (pow(10, (11.344\*(1-1/A0)))-1) ;          SUM += 8.1328e-3 \* (pow(10,(-3.49149\*(A0-1)))-1) ;          SUM += log10(1013.246);          double VP = pow(10, SUM-3) \* humidity;          double T = log(VP/0.61078);   // temp var          return (241.88 \* T) / (17.558-T);  }    // 快速计算露点，速度是5倍dewPoint()  double dewPointFast(double celsius, double humidity)  {          double a = 17.271;          double b = 237.7;          double temp = (a \* celsius) / (b + celsius) + log(humidity/100);          double Td = (b \* temp) / (a - temp);          return Td;  }   #include <dht11.h>  dht11 DHT11;  #define DHT11PIN 2  void setup()  {    Serial.begin(9600);  **········**  }  void loop()  {  **···············**    } |

1.3.3.2 MQ-2 烟雾传感器



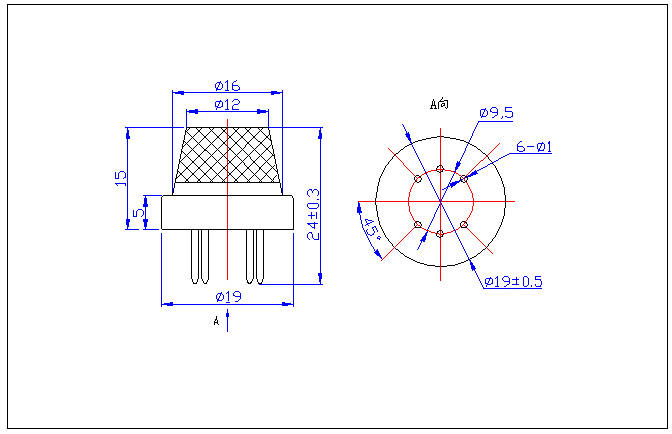
(1)模块描述  
1、采用优质双面板设计，具有电源指示

和TTL信号输出指示；  
2、具有DO开关信号（TTL）输出和AO模拟信号输出；  
3、TTL输出有效信号为低电平。（当输出低电平时信号灯亮，可直接接单片机或继电器模块）  
4、模拟量输出电压随浓度越高电压越高。  
5、对液化气，天然气，城市煤气，烟雾有较好的灵敏度。  
6、有四个螺丝孔便于定位；  
7、产品外形尺寸：32(L)\*20(W)\*22(H)  
8、具有长期的使用寿命和可靠的稳定性  
9、快速的响应恢复特性

(2)模块接口说明

1. VCC:接电源正极（5V）  
   2、GND:接电源负极  
   3、DO:TTL开关信号输出  
   4、AO:模拟信号输出

(3)模块原理图



(4)MQ2引脚说明

|  |  |  |
| --- | --- | --- |
| Pin | 名称 | 注释 |
| 1 | VDD | 供电5VDC |
| 2 | DO | TTL开关信号输出 |
| 3 | A0 | 模拟信号输出 |
| 4 | GND | 接地，电源负极 |

(5)电气特性

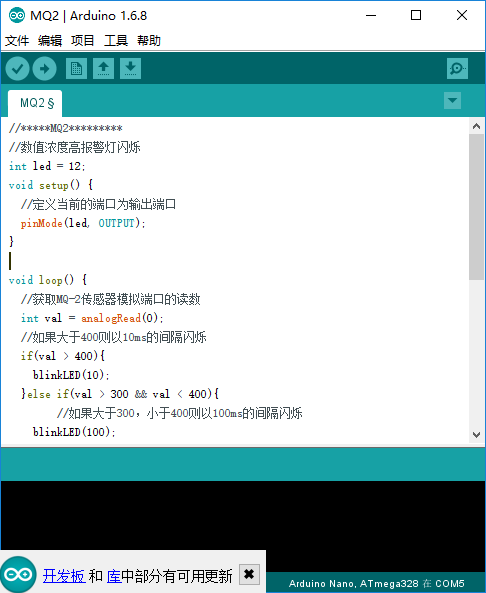
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **条件** | **min** | **typ** | **max** | **单位** |
| **供电** | DC | 3 | 5 | 5.5 | V |
| **供电电流** | 测量 | 0.5 |  | 2.5 | mA |
| 平均 | 0.2 |  | 1 | mA |
| 待机 | 100 |  | 150 | uA |
| **采样周期** | 秒 | 20 |  |  | 次 |
| **A0输出** | DC | 0.1 |  | 0.4 | V |

(6)注意事项：①传感器通电后，需要预热20S左右，测量的数据才稳定，传感器发热属于正常现象，因为内部有电热丝。

②加热电压的改变会直接影响元件的性能，所以在规定电压范围内使用最佳

③环境温湿度的变化会给元件电阻带来小的影响，可进行湿度补偿，最简便的方法是采用热敏电阻补偿。

④传感器要避免暴露其在硅粘接剂、发胶、硅橡胶、腻子或其它含硅塑料添加剂可能存在的地方。



(7)源代码

|  |  |
| --- | --- |
|  | /\*当烟雾浓度越大时LED闪烁频率越高，烟雾浓度降低时LED闪烁频率降低，烟雾浓度为标准浓度时LED灯关闭。\*/  \*\*\*\*\*\*\*\*\*/  //\*\*\*\*\*MQ2\*\*\*\*\*\*\*\*\*  //数值浓度高报警灯闪烁  int led = 12;  void setup() {  //定义当前的端口为输出端口  pinMode(led, OUTPUT);  }  void loop() {  //获取MQ-2传感器模拟端口的读数  int val = analogRead(0);  //如果大于400则以10ms的间隔闪烁  if(val > 400){  blinkLED(10);  }else if(val > 300 && val < 400){  //如果大于300，小于400则以100ms的间隔闪烁  blinkLED(100);  }else if(val < 300 && val > 200){  blinkLED(1000);  }else{ //否则关闭  digitalWrite(led, LOW);  }  }  //闪烁函数  unsigned int blinkLED(int period){  unsigned long time = millis();  unsigned int signal;  if((time / period) % 2 == 0){  digitalWrite(led, HIGH);  }else{  digitalWrite(led, LOW);  }  } |

1.3.3.3 软件功能模块

(1).运用Socket套接字从传感器获取信息

①客户端线程，每当有一个传感器节点连接服务器打开一个新线程，发送接受数据

|  |  |
| --- | --- |
|  | BufferedReader in=**new** BufferedReader(**new** InputStreamReader(client.getInputStream()));  **boolean** flag=**true**;  **while**(flag){  String str=in.readLine();  System.*out*.println("接收数据："+str);  String []handle=str.split("\_");  **if**(handle.length<4){  System.*out*.println("拆分结果："+handle.length+"");  **continue**;  }  **for**(**int** i=0;i<3;i++){  **if**(!handle[i].matches("\\d+")){  c=**false**;  System.*out*.println("数据异常："+str);  }  }  **if**(c){  System.*out*.println("保存数据："+str);  Node data=**new** Node();  data.setNumid(Integer.*parseInt*(handle[0]));  data.setHumidity(Float.*parseFloat*(handle[1]));  data.setTemperature(Integer.*parseInt*(handle[2]));  NodeDaoProxy pro = **new** NodeDaoProxy();  pro.addNodeData(data);  }}}}} |

②调用ServletContextListener 接口监听Web应用的生命周期

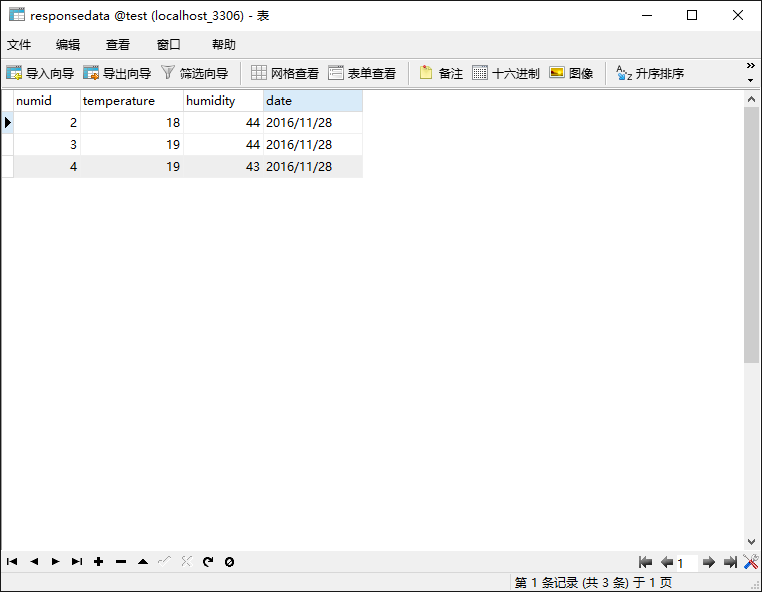
|  |  |
| --- | --- |
|  | **public** **class** myServerListener **implements** ServletContextListener {  **public** **void** contextDestroyed(ServletContextEvent arg0) {  }  **public** **void** contextInitialized(ServletContextEvent arg0) {  **new** server();  }  } |

③Socket服务器

|  |  |
| --- | --- |
|  | **public** server(){  **try**{  server=**new** ServerSocket(8888);  System.*out*.println("等待链接。。。");  **while**(**true**){  client=server.accept();  System.*out*.println("链接成功！！！");  **new** Thread(**new** clientThread(client)).start();  }} |

(2)建立数据模型，创建一个拥有对属性进行set和get方法的类（Node）放在数据容器bean中，并且私有化的属性必须通过public类型的方法暴露给其他程序。

(3)建立MySQL数据库,上传到服务器的温湿度数据，存储在数据库中



通过Java数据库连接（Java Database Connectivity）简称JDBC，连接到MySQL数据库，需要下载MySQL JDBC驱动Connector/J，在程序中需要声明且加载数据库驱动

|  |  |
| --- | --- |
|  | public class JdbcConn {  private static String driver = "com.mysql.jdbc.Driver";  private static String url="jdbc:mysql://localhost:3306/数据库名?useUnicode=true&characterEncoding=UTF-8";  public static Connection conn;  //创建一个连接数据库的方法  public static Connection getConnection(){//加载驱动  try {  Class.forName(driver);  conn = DriverManager.getConnection(url, user, pass);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  return conn;  }} |

(4)业务逻辑层DAO:定义操作的接口，定义一系列数据库的原子性操作标准，如增加、修改、删除、按ID查询等。

|  |  |
| --- | --- |
|  | **public** **interface** NodeDao {  **public** **void** addNodeData(Node node);  **public** List<Node> getAllNodeData();  } |

(5)Dao.impl:DAO接口的真实实现类，完成具体的数据库操作，但是不负责数据库的打开和关闭。

①向数据库插入数据

|  |  |
| --- | --- |
|  | String sql = "INSERT INTO ResponseData(numid,temperature,humidity,date) VALUES (?,?,?,?) ";  **try** {  pstmt = **this**.conn.prepareStatement(sql);  pstmt.setInt(1,node.getNumid());  pstmt.setInt(2,node.getTemperature());  pstmt.setFloat(3,node.getHumidity());  pstmt.setString(4,node.getDate());  **if** (pstmt.executeUpdate() > 0) {// 至少已经更新了一行  flag = **true**;  }  } |

②通过sql语句从数据库查找并获取数据

|  |  |
| --- | --- |
|  | List<Node> all = **new** ArrayList<Node>();  PreparedStatement pstmt = **null**;  String sql = "SELECT numid,temperature,humidity,date FROM ResponseData";  **try** {  pstmt = **this**.conn.prepareStatement(sql);  ResultSet rs = pstmt.executeQuery(); // 执行查询操作  **}** |

(6) Dao.Proxy：代理实现类，主要完成数据库的打开和关闭，并且调用真实实现类对象的操作。

|  |  |
| --- | --- |
|  | List<Node> list = **new** ArrayList<Node>();  **try** {  **this**.dao = **new** NodeDaoImpl(con);  list = dao.getAllNodeData();  } **catch** (Exception e) {  e.printStackTrace();  }  **return** list; |

(7)控制层用Servlet

Servlet运行在服务器端，是由Web服务器负责加载的，是独立于平台和协议的Java应用程序。JSP改变了Servlet提供HTTP服务时的编程方式。HTTP协议的特点是每次连接只完成一个请求，其处理过程为：建立连接、发送请求、提供服务、发送响应，最后关闭连接。

|  |  |
| --- | --- |
|  | **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  NodeDaoProxy pro = **new** NodeDaoProxy();  Node node = **new** Node();  pro.addNodeData(node);  }  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  List list;  NodeDaoProxy service = **new** NodeDaoProxy();  list = service.getAllNodeData();  request.getSession().setAttribute("NODE",list);  response.sendRedirect(request.getContextPath()+"/index.jsp"); } |

(8)创建简单的表格显示页面,运用JSP技术通过在使用HTML编写的网页中添加一些专有标签以及脚本程序就可以实现网页中动态内容的显示，并且可以在任何Web或应用程序服务器上运行。



1.5.应用展望

**1.5.1温湿度传感器应用于烟叶发酵过程中**

在烟草行业，烟叶的仓储是关系到成品烟质量问题重要因素之一，良好的发酵工艺是做出好烟的前提。



由于烟叶自然发酵的时间很长，整个发酵过程要经历4-6个雨季。为保证自始至终不出问题达到理想的醇化结果，必须加强工艺管理，尤其是温湿度的管理。做好防潮、防霉工作。温度最好在20℃左右，相对湿度最好保持在60%-70%。这样一来，无线温湿度传感器便很适用于对温湿度的监测与控制。

如果烟叶含水量高，烟堆体积过大，堆积紧密，当外界气温较高，烟包内会发热增温，引起烟叶碳化（即为烧包）。在具备自然通风的情况下，采用开窗通风的方法，排除库内不适宜的热湿空气，防止烟叶发热。

　　初烤烟和晒黄烟入库时水分在16.0%-18.0%，少数烟包达18%以上。对含水量超标的烟叶，为了尽快降低其水分，减少或防止碳化、霉变，在入库后应采取隔离措施，松包通风摊晾，并用排气扇将湿气排除库外，使水分降低到15%以下。在不具备通风除湿的情况下，应采用除湿机除湿，降低库内空气的相对湿度，使烟叶水分散失至安全范围以内。

利用无线温湿度传感器可与除湿机控制系统连接，自动监测温湿度，达到预设值便能启停除湿机。以后，通过除湿机除湿保证库内温度的正常。

在自然发酵过程中，应注意控制烟叶的含水量和烟包温度，经常对烟包进行检查，使用无线温湿度传感器系统，可提高烟叶的发酵质量。

**1.5.2温湿度传感器在地铁环境中的应用**

地铁乘客流量大，所需新风量变化大。因此地铁的空调负荷变化大，要实现节能必须借助于自动控制的手段。

对此，在地铁车站的站厅和站台区等公共区域，以及重要设备房内，设置室内温湿度传感器，从而得以监测车站实时的温度及湿度。这些参数可帮助运营人员对车站各系统工况进行合理调整，以保持车站公共区始终处于较为舒适的环境、确保设备房一直处于合适的温度之下。



与此同时，在车站的新风室和回风室安装管道温湿度传感器，以此来监测室外新风的和车站内的温度以及湿度。环境控制系统可以根据温湿度传感器采集到的数据来判断车站的环境质量，并根据预先设计好的各种工况来进行自动切换，以实现自动控制系统对车站环境的自动控制，使得车站环境始终处于较为舒适的环境之中，并最终实现节能的减排的目的。