**一、导读**

**1.1项目摘要：**此次项目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2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，研究设计一种使用便捷、实用性高、成本低廉的环境温湿度监测系统

**1.2、背景介绍**

当今现代化建设和国民经济发展迅速．社会对生产环境和生活环境意识的要求也越来越高。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，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。

**二、系统架构**

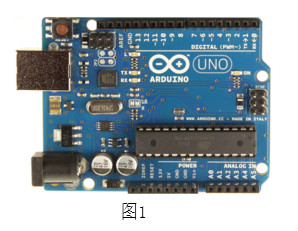
2.1硬件材料

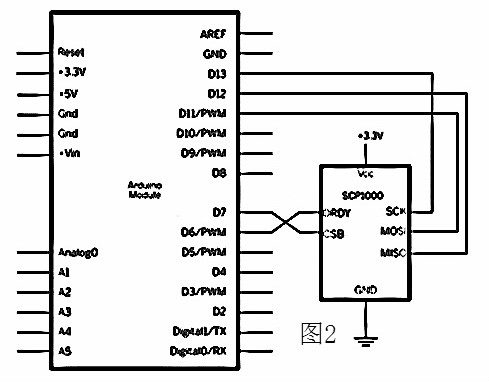
2.1.1 Arduino UNO

Arduino UNO （如图1）在具有单片机的多种优点的前提下，更是具有指令的可读性强，轻松上手，快速应用，质廉价优等特点。而且它较其他单片机而言，编

程灵活，具有丰富的可调用数据库。

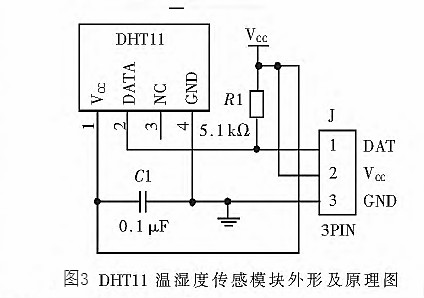
Arduino UNO是Arduino USB接口系列的最新版本，原理图如（图2）所示，作为Arduino平台的参考标准模板。UNO的处理器核心是ATmega328，同时具有14路数字输入/输出口（其中6路可作为PWM输出），6路模拟输入，一个16MHz晶体振荡器，一个USB口，一个电源插座，一个ICSP header和一个复位按钮。





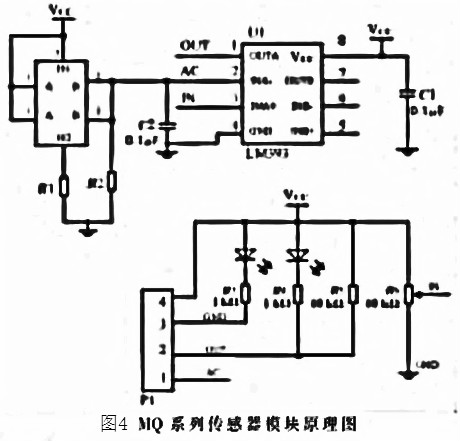
2.1.2 DHTll数字温湿度传感器

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。

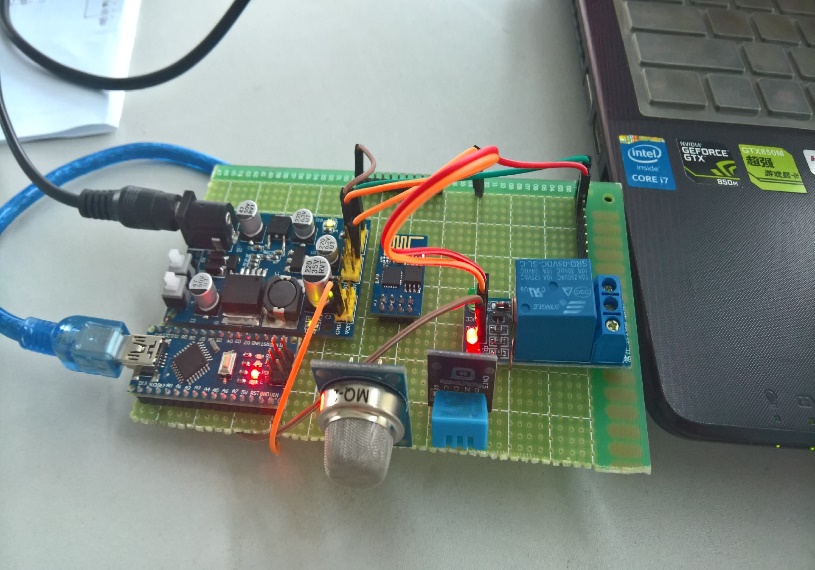


2.1.3 MQ2 气敏传感器

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[5]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。



2.1.4 硬件组成实物图





2.2软件环境

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

**三、系统制作步骤**

**3.1功能模块**

**1.建立数据模型，放在数据容器bean中，创建Node类，定义相应的方法和变量**

**程序清单1.1**

**public** **class** Node {

**private** **int** numid;//id

**private** **int** temperature;//温度

**private** **float** humidity;//湿度

**private** String date;//时间

**public** String getDate() {

**return** date;

}

**public** **void** setDate(String date) {

**this**.date = date;

}

**public** **int** getNumid() {

**return** numid;

}

**public** **void** setNumid(**int** numid) {

**this**.numid = numid;

}

**public** **int** getTemperature() {

**return** temperature;

}

**public** **void** setTemperature(**int** temperature) {

**this**.temperature = temperature;

}

**public** **float** getHumidity() {

**return** humidity;

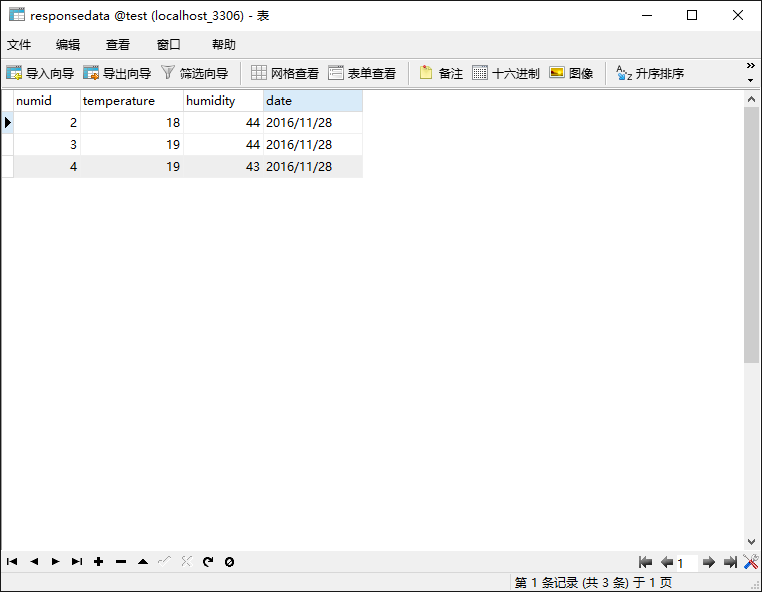
}

**public** **void** setHumidity(**float** humidity) {

**this**.humidity = humidity;

}

**2.建立MySQL数据库**



通过数据库连接类JDBC连接到MySQL数据库，

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";

private static String user="root";

private static String pass="";

public static Connection conn;

public static PreparedStatement ps;

public static ResultSet rs;

public static Statement st;

//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;

}

}

**3.1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();

}

**3.2 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。**

**3.2.1向数据库插入数据**

**public** **void** addNodeData(Node node) {

// **TODO** Auto-generated method stub

**boolean** flag;

PreparedStatement pstmt = **null**;

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**;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (pstmt != **null**) {

**try** {

pstmt.close();

} **catch** (Exception e1) {

}

}

}

}

**3.2.2从数据库获取数据**

**public** List<Node> getAllNodeData() {

// **TODO** Auto-generated method stub

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

**while** (rs.next()) {

Node node = **new** Node();

node.setNumid(rs.getInt(1));

node.setTemperature(rs.getInt(2));

node.setHumidity(rs.getFloat(3));

node.setDate(rs.getString(4));

all.add(node); // 所有的内容向集合中插入

}

rs.close();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** { // 不管如何抛出，最终肯定是要进行数据库的关闭操作的

**if** (pstmt != **null**) {

**try** {

pstmt.close();

} **catch** (Exception e1) {

}

}

}

**return** all;

}

**3.3 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。**

**public** List<Node> getAllNodeData() {

List<Node> list = **new** ArrayList<Node>();

**try** {

**this**.dao = **new** NodeDaoImpl(con);

list = dao.getAllNodeData();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**try** {

**this**.con.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**return** list;

}

**4.控制层用servlet类**

**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");

}

**5.创建简单的表格显示页面**

<center><table class=*"hovertable"*>

<tr>

<th colspan=*"5"*>仓库温湿度监测</th>

</tr>

<tr>

<th>节点号</th>

<th>温度（℃）</th>

<th>相对湿度（%）</th>

<th>时间</th>

</tr>

<%

List list = **null**;

**if**(session.getAttribute("NODE")!=**null**){

list=(List)session.getAttribute("NODE");

**if**(list.size()>0){

Node nd;

**for**(**int** i=0;i<list.size();i++){

nd=**new** Node();

nd=(Node)list.get(i);

%>

<tr onmouseover="this.style.backgroundColor='#ffff66';"

onmouseout="this.style.backgroundColor='#d4e3e5';">

<td><%=nd.getNumid() %></td>

<td><%=nd.getTemperature() %></td>

<td><%=nd.getHumidity() %></td>

<!-- <td><%//=nd.getConcentration() %></td>-->

<td><%=nd.getDate() %></td>

</tr>

<%

}

}

}

%>

</table></center>

<br>

<form action=*"servlet/ShowReport"* method=*"post"*>

<center><input type=*"submit"* value=*"最新数据"*></center>

</form>