目录

一．问题描述

我们周围的环境对生活有很大的影响，设计一个 Zigbee 的智能农业大棚环境检测系统，自动检测农业生态信息，可以自动开启或关闭指定设备。并且设计一个显示界面，在 PC 机上实时显示相应的环境变量。

解决方案以协调器接收到各个传感节点或者路由节点收到的数据为输入，最终 PC 机上的 GGUI 为输出。程序编写包括对输入数据的 C/C++/Java 编程。此外还涉及了数据库，移动开发平台等技术，几种方案的系统基于 2019 年 1 月物联网课程设计系统进行开发，以及增加对手机端的实现，此次实现的功能如下：

（1）基于 Zigbee 的智能农业大棚环境监测

系统能够自动检测农业生态信息即对接收监测传感节点数据，可以自动开启或关闭指定设备，能在 PC 机以及手机端上实时显示相应的环境变量。

（2）身份验证以及数据存储与查询

系统提供登录界面，可以实现对管理人员的注册和登录。系统 PC 端可以实现对历史数据的查询与显示。

（3）WSN 传输组网

系统能够实现对含路由器的网络拓扑的支持，并且对于树-簇型网络，系统能够对网络拓扑 中各种设备进行拓扑跟踪，实现拓扑绘制。

二．设计需求及分析

2.1 抽象数据类型

作用：抽象数据类型可以使我们更容易描述现实世界。

定义：一个数学模型以及定义在该模型上的一组操作。

关键：使用它的人可以只关心它的逻辑特征，不需要了解它的存储方式。
定义它的人同样不必要关心它如何存储。

2.1.1 数据转换类

```
ADT Tran{
    数据对象：D1={String, byte,StringBuffer}
    基本操作：
    bytesToHex();//字节转为字符串
    hexStringToByte();//字符串转为字节
    toByte();//16 进制字符转为字节
    }
```

2.1.2 设备数据类

```
ADT SerialTool{
        数据对象:D2={Tran,serialTool, Enumeration<CommPortIdentifier>,
ArrayList<String>, Exception, serialPort}
```

数据关系：R2={< sendToPort,Tran>,< serialPort, readFromPort>,< serialPort ,sendToPort>…< SerialPort, closePort>}

基本操作：

findPort ();//发现可用端口

openPort ();//打开串口

closePort ();//关闭串口

sendToPort();//向串口发送数据

readFromPort();//从窗口读取数据

addListener (SerialPort port, SerialPortEventListener listener);

}

2.1.3 交互界面类

ADT View{

数据对象：D3={ TextField, Label, List<String> , SerialPort, Connect, Tran, Button, Choice, ShowLED, PHOTO ,History, Smoke, Graphics }

数据关系：R3={< Graphics, TextField >……< Graphics , Button >}

基本操作：

setTitle(String title);//设置界面标题

setBounds(int x, int y, int width, int height);//设置界面边界

setBackground(Color bgColor);////设置背景颜色

addWindowListener(WindowListener l);//设置串口监听

addActionListener(ActionListener l);//Button 事件监听

setFont(Font f);//设置字体

SerialListener();//数据基础处理显示并传入各个设备处理

……

}

2.1.4 数据库处理类

ADT Connect{

数据对象：D4={String, sql.Statement, Connection, ResultSet,}

数据关系：R4={<String, Statement ,ResultSet>}

基本操作：

getConnection().createStatement();// 连接数据库 Creates a Statement object for sendingSQL statements to the database.

st.executeQuery(sql);//执行 sql 语句返回结果 Executes the given SQL statement, which returns a single ResultSet object.

getConnection().prepareStatement(sql);// Creates a PreparedStatement object for sendingparameterized SQL statements to the database.

redDate();//读取数据库

judge(String username, String password);//身份判定

rejudge(String username, String password);//身份注册判定

```
        showData();//数据库内容显示
        add(double d, double e, int led, int y, int p, int g);//添加
    }
```

2.1.5 设备类型

```
ADT Device{
        数据对象：D5={ Connect , SerialPort,}
        数据关系：R5={< SerialPort ,Connect>}
        基本操作：
        Tosend();//经过逻辑判断对设备发出相应控制信息
    }
```

2.1.6 报错类

```
ADT serialException{
    数据对象：D6={String,serialTool}
    数据关系：R6={< SerialPort ,Connect>}
    基本操作：
    ExceptionWriter;// 负责将传入的 Exception 中的错误信息提取出来并转
换成字符串；
    NoSuchPort;//返回没有串口
    NotASerialPort;//
    ReadDataFromSerialPortFailure;//数据读取失败
    SendDataToSerialPortFailure;//发送数据失败
    SerialPortInputStreamCloseFailure;// 设备串口输入流关闭失败
    SerialPortOutputStreamCloseFailure;// 设备串口输出流关闭失败
    SerialPortParameterFailure;//
    TooManyListeners;//

}
```

2.1.7 组网类

```
ADT DrawMap{
    数据对象：D7={int[][],int[],String[]}
    数据关系：R7={<int[] change,String[] IDname>,<int[][] Mymap,int []
vis>}
    基本操作：
    Update(String s);//判断目前组网架构又没由新增节点或者连接方式有变
    Show();//将图中存储的信息以显示的界面
    DrawMap()//初始化图形界面与存储结构初值;
  }
```

2.2 主程序流程

本程序拥有六个模块

2.2.1 登录模块

实现：身份验证，管理员登录与注册 流程图如图 2-1

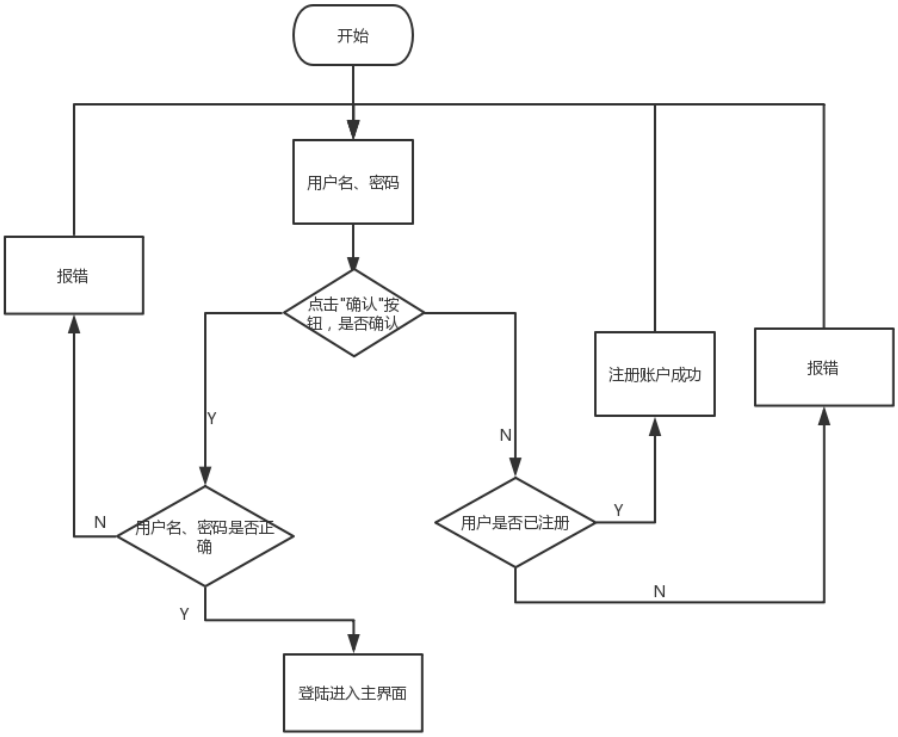

图 2-1 身份验证流程图

模块伪代码：

```
Enter(){
    //主要功能实现界面的构造
    //六个构件 JTextField JPasswordField JLabel*2 JButton*2
    JLabel.init();//设置在界面显示方式
    //主要为 setBounds(),setBackground(),setFont(),setForeground()
    JPasswordField.init();
    JLabel.init();
    //对于 JButton 按钮添加监听
    check.addActionListener(this);
    register.addActionListener(this);
}
actionPerformed (ActionEvent e){
    //对于按下按钮“确认”“注册”
    if(check){//确认
    //连接数据库 判断数据库中是否有该用户信息,并且该用户信息密码和
```

用户名是否相匹配

```
            if（OK）{//
                new Client().launchFrame();
                this.setVisible(false);
            }
            else 报错;
        }
        else{//注册
            判断是否以存在与库中
            if(不存在) 插入，注册成果
            else 报错;
        }
    }
```

2.2.2 主界面模块

实现：版本显示与系统基本信息介绍，跳转显示主界面

模块伪代码：

```
launchFrame(){
    主界面信息显示;
    Enter 键监听;
}
KeyMonitor(){
    setVisible(false);     //隐去欢迎界面
     dataview.setVisible(true);     //显示监测界面
     dataview.dataFrame();     //初始化监测界面
}
```

2.2.3 设备控制模块

实现：对温湿度信息进行相应处理

模块伪代码：

```
wait_my(){}//延时函数，为了保证数据发送有时间间隔
ShowLED.toSend(){//wendu
    if(temp>tu||humm>hu) 控制电机正转 LED1 亮 返回温湿度不合适
    else if(temp<td||humm<hd) 控制电机正转 LED1 亮 返回温湿度不合适
    else 停止电机转动 LED3 亮返回正常
    //判断上一条数据库信息
    //更新数据库
}
PHOTO.toSend(){
    if(pp<350) LED2 亮 返回亮度过小
    else LED2 灭 返回正常
    //判断上一条数据库信息
    //更新数据库
```

}

2.2.4 监测模块

实现：信息显示，数据预处理转向设备处理模块，提供数据调控与查询接口主要数据采集相关处理实现如图 2-2

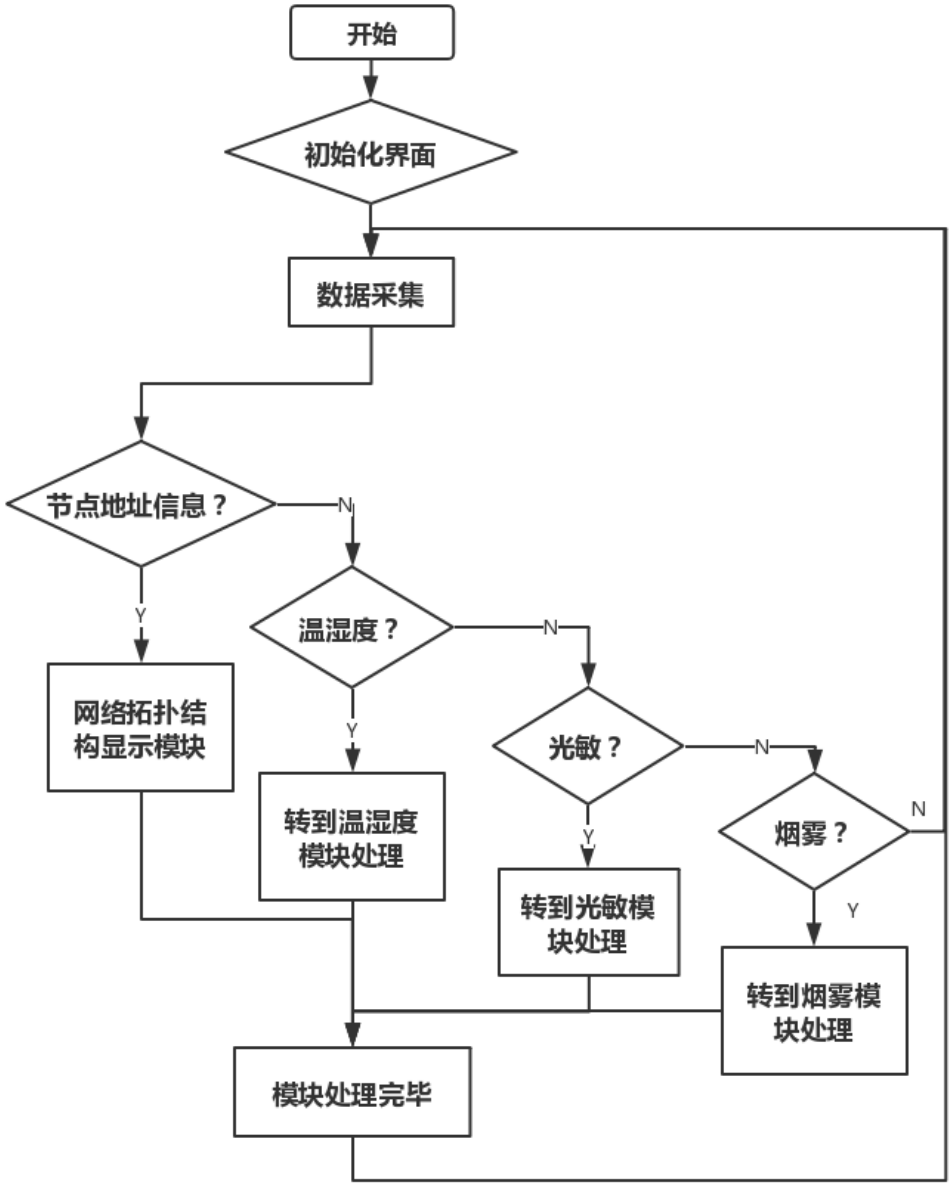

图 2-2 数据采集流程图

模块伪代码：

```
dataFrame(){
    界面初始化，添加控件
}
```

```
paint(){
    界面初始化，添加必要文字信息
//主体如下形式
    Graphics.setColor(Color c)
    Graphics.setFont(Font font)
    Graphics.drawString(String str, int x, int y)
}
wait_my(){}//延时函数，为了保证数据发送有时间间隔
SerialListener(){

    data = SerialTool.readFromPort(serialPort); // 读取数据
    String dataOriginal = Tran.bytesToHex(data); // 将字节数组数据
转换位为保存了原始数据的字符串

    //采取界面的显示信息
    if (flag.matches("eecc010301")){//判断数据为温湿度数据
       String showL = showled.tosend();//进行相应控制
    if (showL != null) pa.setText(showL);//更新显示数据
    }
    //同理对于光敏烟雾的信息数据的更新和相应处理
}
```

2.2.5 数据库管理模块

```
实现：数据更新查询插入
模块伪代码
//数据库连接
conn = DriverManager.getConnection(url, user, password);
redDate(){//读取数据库
    String sql = "select * from MyDate_W";//sql 语句中的查询 数据表
    st = getConnection().createStatement();
    ResultSet rs = st.executeQuery(sql); // 用于返回结果
    //这里主要为判断最新信息与当前想要 add 的信息是否相似
    返回灯状态
}
judge(String username, String password){ ;//身份判定
    //查询数据库用户表
    if(name==username&&pass==password) return 1; //如果库中存在此用
户名和密码相匹配
    return 0;
}
int rejudge(String username, String password){;//身份注册判定
    if (username.equals(name.substring(0, lenu))) {
          rs.close();return 0;// 已经有该用户了
    }
```

```
        没有该用户，则在数据库 Manage 表中插入数据
        PreparedStatement ps = getConnection().prepareStatement(sql2);
        ps.setString(1, username); // 给其五个参量分别“赋值”
        ps.setString(2, password); // 给其五个参量分别“赋值”
        ps.executeUpdate(); // 参数准备后执行语句
        ps.close();getConnection().close();
    }
    Object[][] showData();{//数据库内容显示
        //为历史数据查询提供接口
        Object[][] rowData = new Object[30][5];
        String sql = "select * from MyDate_W";
        st = getConnection().createStatement();
        ResultSet rs = st.executeQuery(sql); // 用于返回结果
        int i = 0, j = 0;
        //存储信息
        return rowData;
    }
    add(double d,double e, int led, int y, int p, int g){;//添加
    //为信息更新提供接口，通过 ShowLED 处判断这条信息是否需要插入数据库
    String sql = "insert into MyDate_W (温度,湿度,灯状态,烟雾,光强反馈,
光照,操作日期)" + "values(?,?,?,?,?,?,?)";
        PreparedStatement ps = getConnection().prepareStatement(sql); //
用 preparedStatement 预处理来执行 sql 语句
        ps.setDouble(1, d); // 给其五个参量分别“赋值”
        ps.executeUpdate(); // 参数准备后执行语句
        ps.close();
    }
```

2.2.6 网络拓扑模块

实现：提取整个网络的拓扑结构，动态显示

模块伪代码

```
DrawMap(){
    //初始化显示界面
    //初始化 标记数组，与有向图数组
    for (int i = 0; i < vis.length; i++) vis[i] = -1;
        for (int i = 0; i < 256; i++)
            for (int j = 0; j < 256; j++)
                Mymap[i][j] = 0;
        show();//显示界面
}
Update(String s){
    //提取自身地址自身 MAC 地址，父节点地址父节点 MAC 地址，设备类型
    //devID,myIp,myMac,faIp,faMac
    //自身地址和设备类型进行映射为了方便输出。change[myIp]=devID;
```

```
    if(vis[myIp]==-1) else if(vis[myIP]==faIp) else
    //判断该数据，1、是新增节点 2、是已存在节点 （父节点改变/未改变）
    Show;//显示界面
}
Show(){
    以树形结构显示整个网络拓扑结构。
}
```

2.3 各模块调用关系

2.3.1 Enter 模块

表 2-1 函数解析

judge(String username, String password)	身份判定
rejudge(String username, String password)	身份注册判定
setBounds()	设置边界
setTitle()	设置标题
setBackGround();	设置背景
JOptionPane.showMessageDialog()	报错提示框
Client().launchFrame()	启动主界面

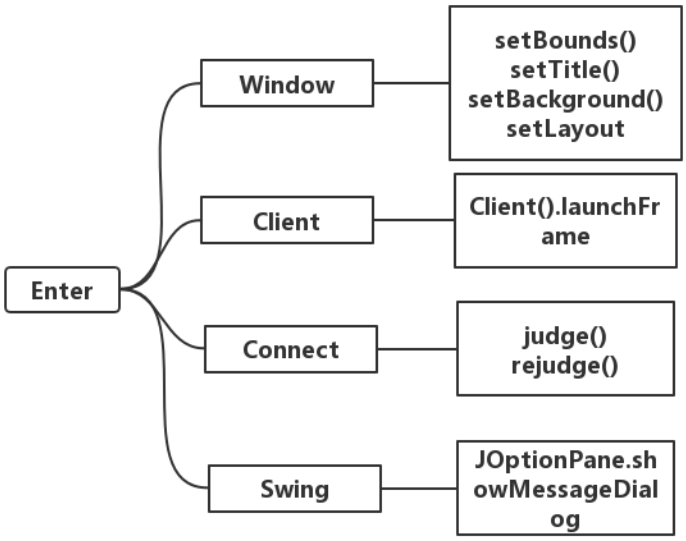

图 2-3 Enter 模块函数调用情况

2.3.2 Client 模块

表 2-2 函数解析

setVisble()	设置界面可视
addWindowListener()	添加监听
DataView().dataFrame()	启动监测界面
addKeyListener(KeyListener l)	添加键盘监听
setResizable()	设施边界是否可改变
setFont()	设置字体

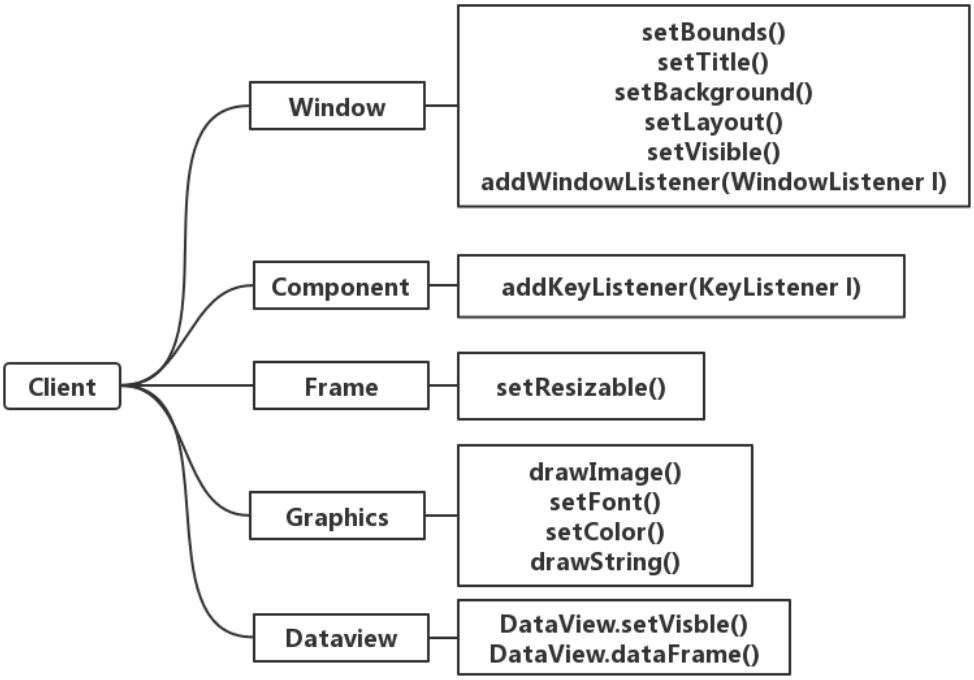

图 2-4 Client 模块函数调用情况

2.3.3 DataView 组网显示调用模块

表 2-3 函数解析

showLED.tosend()	调用 ShowLED 相应函数处理相对数据
PHOTO.tosend()	调用 PHOTO 相应函数处理相对数据
Smoke.tosend()	调用 Smoke 相应函数处理相对数据
Thread.sleep()	延时
DrawMap.update()	更新显示动态变化的组网结构

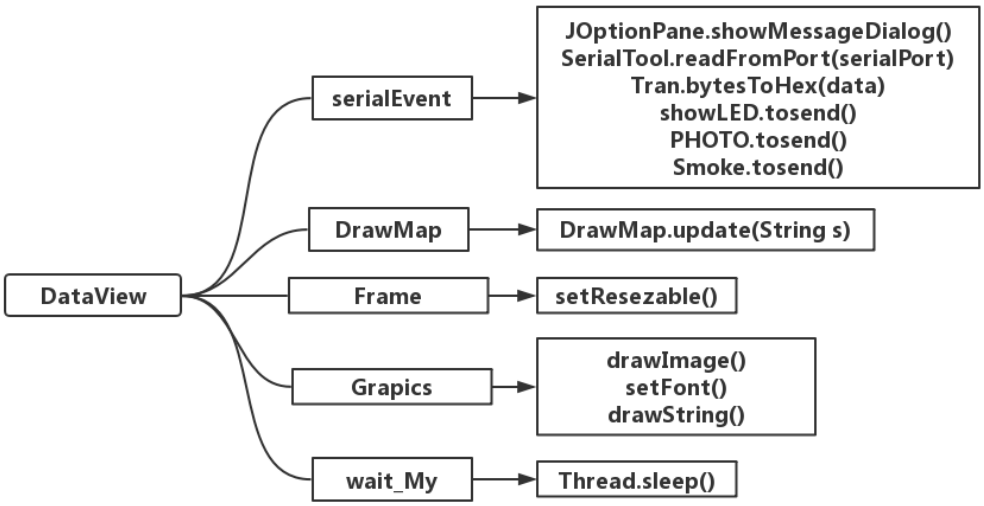

图 2-5 DataView 模块函数调用情况

2.3.4 ShowLED/PHOTO/Smoke 模块

表 2-4 函数解析

connect.add()	向数据库中插入数据
Connect.redDate()	从数据库中读取数据
SerialTool.sendToPort()	向设备发送数据

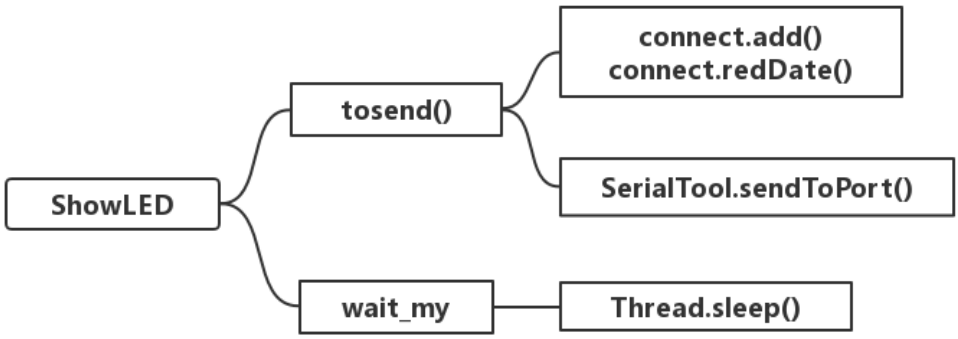

图 2-6 ShowLED 模块函数调用情况

2.3.5 History 模块

表 2-5 函数解析

showLED.tosend()	调用 ShowLED 相应函数处理相对数据
PHOTO.tosend()	调用 PHOTO 相应函数处理相对数据

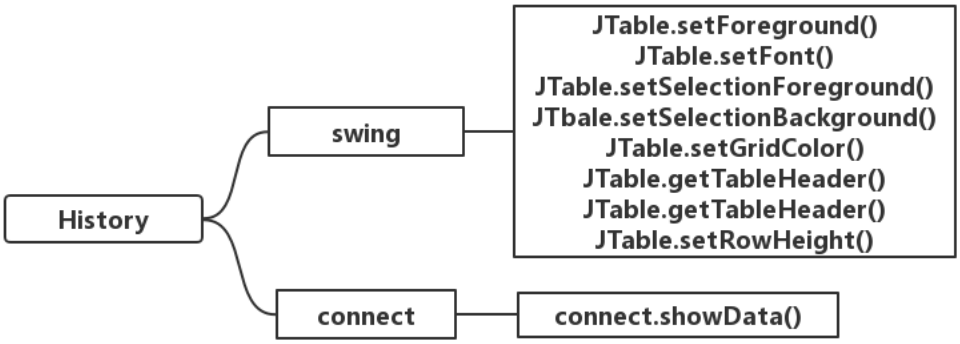

图 2-7 History 模块函数调用情况

2.3.6 Connect 模块

表 2-6 函数解析

DriverManager.getConnection()	a connection to the given database URL
Connection.createStatement()	Creates a Statement object for sendingSQL statements to the database
Statement.executeQuery()	Executes the given SQL statement
Connection.prepareStatement()	Creates a PreparedStatement object for sendingparameterized SQL statements to the database

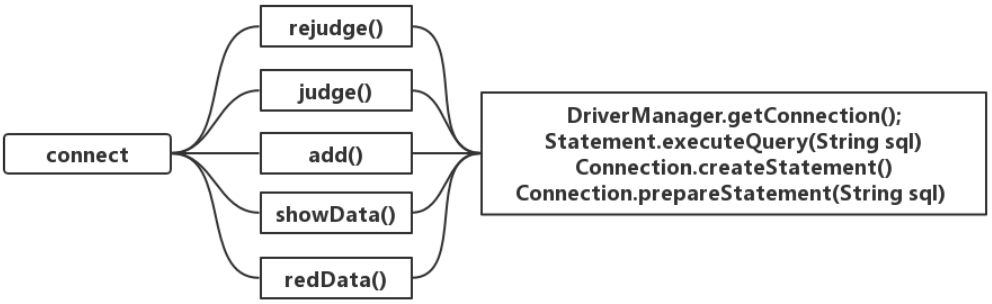

图 2-8 Connect 模块函数调用情况

2.3.7 SerialTool 模块

表 2-7 函数解析

findPort()	发现串口
openPort()	打开串口
SendToPort()	发送数据
readFromPort()	读取数据
closePort()	关闭串口
Tran.hexStringToByte	十六进制转为字节

Tran.byteaToHex	字节转为十六进制

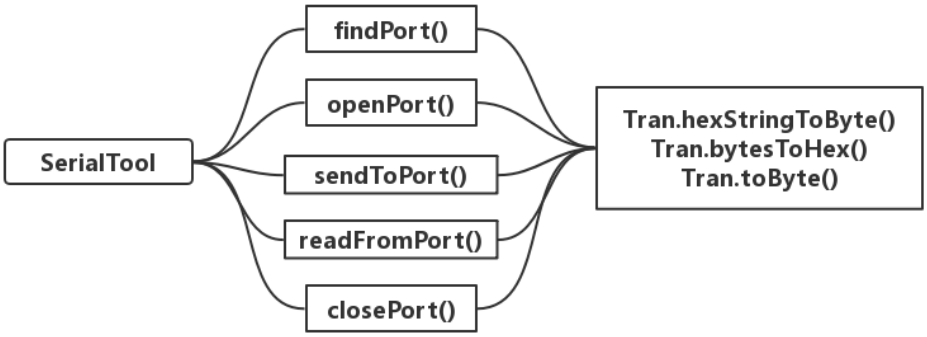

图 2-9 SerialTool 模块函数调用情况

2.4 网络拓扑分析

2.4.1 ETC-WSN 设备类型分析

在 ZigBee 无线传感器网络中有三种设备类型：协调器、路由器和终端节点，设备类型是由 ZigBee 协议栈不同的编译选项来选择的。

1）协调器(Coordinator)

协调器负责建立网络，系统上电后，协调器会自动选择一个信道，然后选择一个网络号，建立网络。协调器主要是在网络建立、网络配置方面起作用，一旦网络建立了，协调器就与路由器的功能是一致的。

2）路由器(Router)

在 ZigBee 网络中，路由器主要有三个功能：

•允许节点加入网络；

•进行数据的路由；

•辅助其孩子节点通信。

3）终端节点(End-device)

终端节点只需要加入已建立的网络即可，终端节点不具有网络维护功能。

2.4.2 分析指令格式：

1）指令格式

帧头 +节点编号+ 模块 ID + 传感器 ID + 命令（ParamH＋ParamL） + 帧尾 帧头：CC EE

节点编号：01-----FF ID:：01~FF

命令： 帧尾：FF

对于本次实验中使用的终端节点有：3 个

路由器节点：2 个

协调器：1 个

在不考虑节点时如何烧写下，通过对数据结构的分析：可在系统中拓扑出整个系统的组网系统。

对于协调器和终端节点：

打开协调器电源开关，然后打开终端节点电源开关，等终端节点连接上网络后，每隔 5s，会发现协调器的 LED 红灯闪烁两下，这说明协调器已经成功收到了终端节点发送的“ChinaSofti”数据。

对于协调器和路由节点：

协调器上电后建立网络，路由器自动加入网络，然后路由器调用上述 4 个函数获取本身的网络地址、MAC 地址、父节点网络地址和父节点 MAC 地址，然后通过串口将其输出到 PC 机。

2）PC 实际接收数据

对于 PC 来说我们收到的数据为 32 位 16 进制（以下以字节依旧是二进制的八位 16 进制的 2 位来说明）

PC 收到的第五位为 AA 则说明该数据为

通过判断模块 ID：00 网关，AF 为路由，03 温湿度，02 光敏，09 电机

通过命令中第六字节到第九字节：本身的网络地址、父节点网络地址

例如：如图 2-10

EECC 01 00 AA 00 00 00 00 00 00 00 00 00 00 FF(协调器)

EECC 01 AF AA 48 27 00 00 FF 17 00 00 00 1F FF(路由器 1 回复)

EECC 01 03 AA 88 28 48 27 FF 17 00 00 00 1F FF(温湿度回复)

其中路由器中 48 27 00 00 表示本身的网络地址 48 27、父节点网络地址 00 00

图 2-10 串口接收数据

则对于多个路由节点 可建立 如图 2-11 拓扑结构：

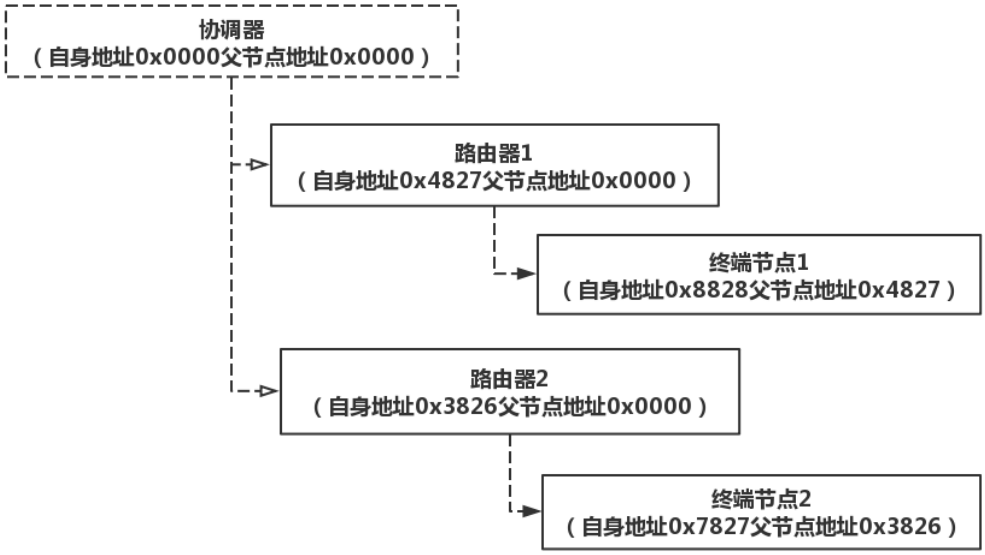

图 2-11 组网拓扑

2.4.3 实现组网拓扑显示与分析

1）广播处理

通过数据分析出自身地址与父节点地址转化为一个有向图，利用二维数组形式存储。由于设备不多，基本没有冲突，**采用地址前两位**的地址信息作为表示，那么地址信息范围为 0x00-0xff，转为十进制范围为 0-255，开一个 256*256 的二维数组可以存储，父节点到子节点标记为 1，表示有向边。

2）动态显示

将地址和设备关系进行映射，每当有新的节点加入或者是由节点的父节点地址改变，则刷新网络拓扑图。自身地址→设备 ID→设备名。

实际显示如图 2-12 某次组网架构

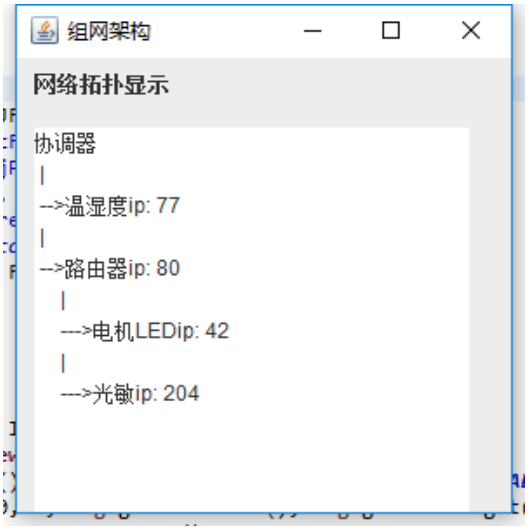

图 2-12 组网拓扑

三．设计功能的实现

3.1 系统功能概要设计

本基于物联网的智能农业大棚系统，主要包括对环境变量的采集和显示和控制设备，以及历史数据查询，身份验证，以及网络拓扑显示。通过选用合适的传感器，实现对环境温度、湿度、光照强度以及是否有人等数据进行监测，并根据光照强度控制 LED 灯的亮灭状态、根据温度变化控制电机正转反转，其具体系统框图如下：

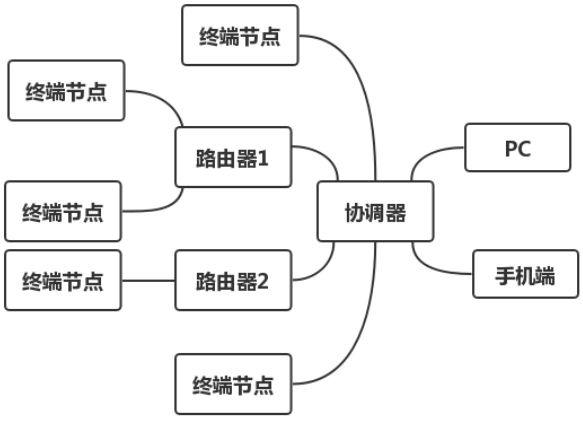

图 3-1 系统框图

智能农业系统是面向所有的农业大棚，为大棚管理人员提供对智能农业系统的各项功能。根据现在对旅游信息管理的需求，我们对智能农业硬件设备，植物生长环境，数据库信息存储等进行管理。主要包括环境信息管理，设备信息管理，系统管理，数据库设计。这四个部分互相协作共同实现智能农业系统管理的自动化、精细化和科学化。如下图系统总体结构图，表示了智能农业系统的主要的六大模块功能。

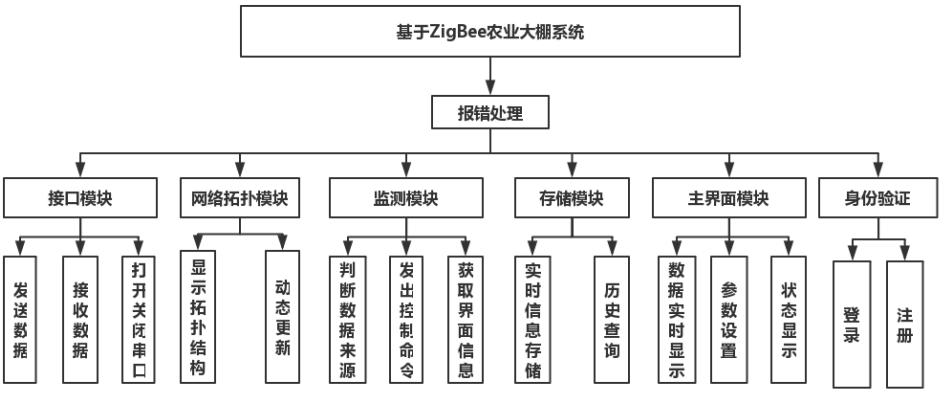

图 3-2 系统模块图

3.2 环境信息管理功能

环境信息管理功能是为系统管理员和业务人员提供查看植物生长环境信息、检索环境信息、以及对环境信息进行管理等，其功能结构图见图 。

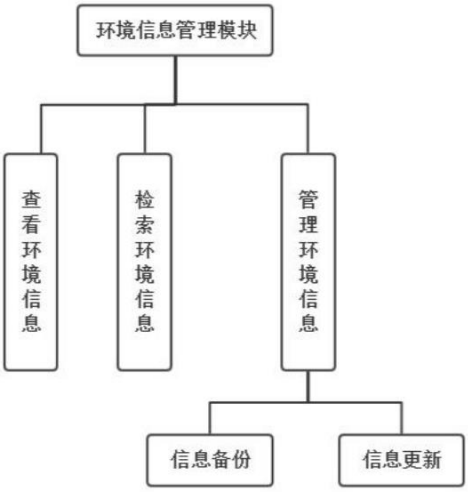

图 3-3 环境信息管理功能结构图

环境信息管理模块包括：查看环境信息模块、检索环境信息模块、环境信息备份模块和环境信息删除模块。具体功能如下： 查看环境信息：根据传感器节点收集的智能农业系统大棚内植物环境的各参 数值，实时地显示在环境信息界面，也可选择任一参数进行查看该采集的实时数据。

环境信息检索：当节点采集环境信息并将数据存入数据库内，系统管理员和业务人员可选择任意参数进行该数据采集的实时监测或者提取该参数的历史数据进行统计分析。

环境信息备份：对环境信息的数据进行备份存档，以便不良操作丢失。

环境信息删除：对环境信息进行删除。

环境信息管理模块中各子模块的实际意义是当系统管理员和业务人员在管理环境信息的时候，可以快速、便捷地将采集的环境信息添加到系统中，同时对信息进行维护和管理，并与数据库中数据相连接，这样不仅方便监测，还提高了

信息的准确性。系统管理员和业务人员还可以对环境信息进行查询，确保环境信息的准确性、及时性，从而保证系统的准确性和完整性。

3.3 设备信息管理功能

设备信息管理模块中各子模块的实际意义是当工作人员在管理设备信息的时候，可以快速、便捷地将设备信息添加的系统中，同时对信息进行维护和管理，当此设备更名或动迁时，也可以及时地对系统中的相应信息进行修改，提高信息的准确性。管理员还可以直接对设备信息进行查询，确保设备信息的准确性以及运行的可靠性，从而保证系统的准确性和完整性。信息的添加和修改与后台数据库进行动态的连接，其功能结构图如下图所示。

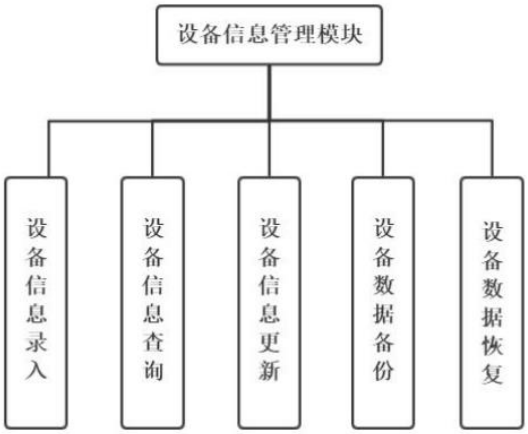

图 3-4　设备信息管理功能模块图

设备信息管理模块主要是对设备的信息进行管理和维护。

设备信息模块包括：设备信息查询模块、设备信息录入模块、设备信息更新模块和设备数据备份模块以及，具体功能如下：

设备信息查询：根据设备的名称进行相应的查询。也可以直接查询全部设备信息。

设备信息添加：添加新加入的设备信息。

设备信息更新：对于各设备相关信息的变更做出及时的修改。

设备数据备份：对于设备信息数据进行备份存档。

设备数据恢复：对于系统故障丢失数据可通过备份文档恢复设备数据。

3.4 系统管理功能

系统管理提供支撑包括环境监测管理系统在内的各功能模块正常运行的各项基础和通用功能，完成智能农业系统的系统管理、数据管理等一系列操作，其功能结构图如图所示。

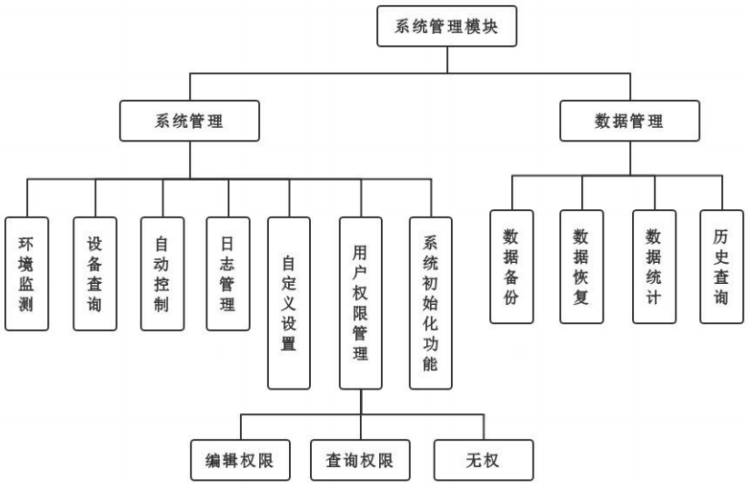

图 3-5　系统管理功能结构图

系统管理的主要功能有：环境监测、设备查询、自动控制，自定义设置、用户权限管理、历史查询、系统初始化，网络拓扑功能。

1）环境监测

环境监测功能，是智能农业系统的核心功能，主要将底层采集的数据，将数据经行处理，提供给管理人员。

2）设备查询

设备查询功能，提供农业大棚内控制设备的状态查询功能。实时提供当前系 统的设备运行的状态，为智能控制提供基础条件。

3、自动控制、

自动控制功能，设置环境数据的上下限，当目前环境实际值不在规定范围内，提供报警机制,同时自动开启相关设备进行改善环境参数。如,当大棚内温度过高，对植物生长不利，系统报警，同时开启风扇、水帘等相关设备进行温度缓解，若能控制温度恢复到预定值内则消除报警，否则系统持续报警，将任务反馈给管理员，交予处理。

3）用户权限管理

为了维护系统的安全，只有经过授权的用户才能进入智能农业系统，进行相 应的操作。

4）历史查询

历史查询功能是系统管理员以及业务人员可根据所需查询相关的环境或设备参数阶段性数据或者时间间隔内的数据，实现单一环境数据查询、可实现查询所有数据按所需次序排序显示，也可以查询设备阶段性的运行状态，更好地维护系统环境状况，提供良好的植物环境监测控制。

3.5 数据库设计

建表 Adp1 存储采集到的数据：

use DP create tableMyData_W（温度 float，湿度 float，灯状态 int(5)，烟雾 float (5)，操作日期 datetime，其他 char(20)，光照 int ）

温度	湿度	灯状态	烟雾	光强反馈	操作日期	其他	光照
31.59	35.36	23	0	1	2019-06-04 1...	NULL	386
31.44	35.5	12	0	1	2019-06-04 1...	NULL	0
30.82	36.43	12	0	1	2019-06-04 1...	NULL	0
29.29	57.74	13	0	1	2019-06-13 0...	NULL	178
28.86	57.09	13	0	1	2019-06-13 0...	NULL	176
29.35	55.38	13	0	1	2019-06-13 0...	NULL	333
29.82	55.88	13	0	1	2019-06-13 0...	NULL	127
30.96	36.95	23	0	1	2019-06-04 1...	NULL	393
30.94	36.54	12	0	1	2019-06-04 1...	NULL	393
29.59	54.45	13	0	1	2019-06-13 0...	NULL	179
29.97	53.69	13	0	1	2019-06-13 0...	NULL	179
29.82	54.32	13	0	1	2019-06-13 0...	NULL	0
29.89	55.26	13	0	1	2019-06-13 0...	NULL	335
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 3-6 数据存储

建表 Manage 存放用户名及密码：

```
use DP create table Manage(用户名 char(10) primary key,密码 char(15))
insert into people values('admin',123)
```

LAPTOP-O7I0V8DL.DP - dbo.people

	name	password
	admin	123

图 3-7 Manage 表

四. 实例测试及运行结果

4.1 测试结果

1）身份验证

图 4-1 登录界面

图 4-2 登录用户密码错误提示

图 4-3 注册失败提示

图 4-4 注册成果提示

2）监测界面

图 4-5 登陆成功界面

图 4-6 数据显示界面

3）历史查询

日期	温度	湿度	光敏	灯状态
2019-06-13	29.83	53.44	326	13
2019-06-13	29.82	54.32	0	13
2019-06-13	29.97	53.69	179	13
2019-06-13	29.59	54.45	179	13
2019-06-13	29.33	55.7	178	13
2019-06-04	30.94	36.54	393	12
2019-06-04	30.96	36.95	393	23
2019-06-13	29.82	55.88	127	13
2019-06-13	29.35	55.38	333	13
2019-06-13	28.86	57.09	176	13

图 4-7 历史数据查询界面

4）网络拓扑结构

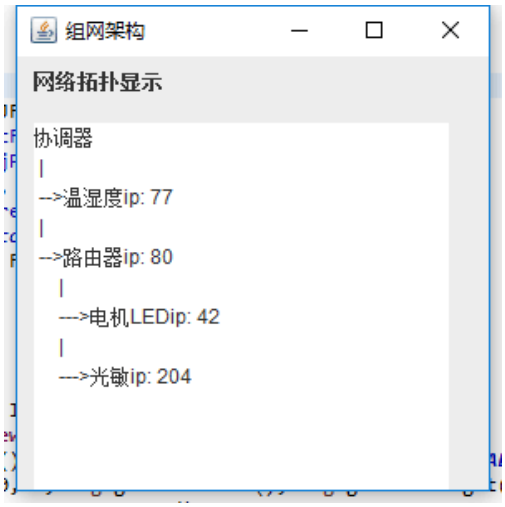

图 4-8 组网架构显示

4.2 测试分析

4.2.1 问题与解决以及回顾讨论与分析

1）新增的几个模块身份验证与历史查询都和数据库内容相关，新增了一个管理员表，此过程没有多少问题，主要还是 java swing 包的使用不太熟练。

2）操作不能实现，不能同时发送两条指令。当我们向关闭 LED3 时候下一条指令打开 LED1 就不能实现。传感器接收数据不是像电脑运行程序那样快，所以当在每两条传送数据中间加了一条 wait()操作，延时一秒发送，这个问题也就被解决了。

3)历史查询,默认的 ResultSet 对象不可更新,仅有一个向前移动的指针。因此，只能迭代它一次，并且只能按从第一行到最后一行的顺序进行。在用 jdbc 操作数据库时，使用 Statement stmt=Conn.createStatement 容易发生“只进结果集不支持请求的操作”的错误，这是因为，如果这个结果集就只能迭代一次，就不能 rs.XXXX 调用 ResultSet 下的某些方法了。解决办法，在调用对数据库增删改查代码之前，获取 Statement 的时候要加上以下语句，方可避免错误。st=getConnection().createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);getConnection().prepareStatement(sql,ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);

4）对于描绘组网架构时，开始困于没有很好的理解 ZigBee 协议，不懂他是如何判断出怎么接收到数据，最后和队友讨论了很久，最后明白了每个节点都会周期性的发送自身节点广播信息，该信息里面就包括了父节点地址和 MAC 地址，这样就可以构建出组网的结构了。

明白上面的内容之后，想用链表的形式存储，父节点地址下跟着子节点的地址，但是苦于队 Java 结构不熟悉，在 C++下只要只用 stl 库下的 vectoe<int> v 就可以解决二维链表。最后判断数据量并不是很大，直接用二维数组，有向图的

方式存储了整个网络拓扑结构，并随着改变动态更新。

4.2.2 算法的时空分析和改进设想；

对于每次匹配用户和密码，以及显示历史数据，读取数据内容时时间复杂度都是 O(n),n 为数据库中表的行数，且随着不断插入的数据而增大。对于数据库插入则是 O(1)的复杂度。单个数据的处理，更新的复杂度都为 O(1)。

经过改进在历史数据查询部分数据读取都不在是 O(n),而是 O(1)，而在组网显示部分复杂度为 O(n^2)，n=地址的大小 256。

改进对于数据库表操作不可避免，有两个方向可以降低。一为避免无用数据的增加，二为寻找更好的匹配算法，避免从头到尾的匹配这其实更多要涉及 Mysql 这里的知识。

改进对于组网显示部分，可以讲数据类型改变，以链表的形式，就可以大致降低 O(nlog(n))，还有基于本次网络并不会很复杂，我值进行了两次搜索，如果出现大型的存在很多路由节点和终端节点，这样的搜索方式可能会不准确。修改方式，以 dfs 的方式进行。

4.3 改进与体会

由于上次课设的内容就是智能农业系统，所以在本次物联网传输课设中系统还是沿袭上次的系统进行完善，增加了用户登录注册以及历史数据查询以及组网传输图。程序在编写时已经尽可能考虑了各个模块的耦合性，因此系统能够不断改进新增以适新的需求。编程主要还是涉及的是 ZigBee 协议栈的应用层，获取底层传来数据的信息，传入不同模块进行处理最后以最清晰的形式显示。可以改进的是在考虑后期组网传输时，对数据分析后，可以提供一个界面显示整个网络的组成形式。

组网中对拓扑结构的展现。这里需要掌握 ETC－WSN 传感器节点通信协议_V3.0，传感节点指令的格式，以及路由节点指令格式（帧头 +节点编号+ 模块 ID + 传感器 ID + 命令（ParamH + ParamL） + 帧尾）。通过接收协调器数据分析，解析出信息中每个节点返回的关于自身节点地址信息和父节点地址信息，利用有向图的形式存储信息，在通过过 GUI 显示。

五．实现提示

5.1 心得体会

最大的心体会就是要有一个想要学习和前进的心，有很多知识等我们挖掘，有很多没有做到的地方在等我们去实现。前期课设过程中就是原有基础上新增一些模块，没有深入去了解 zigbee 协议栈使用，在后期才去一点点了解。其实在最后既然 ZigBee 协议栈已经实现了 ZigBee 协议，那么用户就可以使用协议栈提供的 API 进行应用程序的开发，在开发过程中完全不必关心 ZigBee 协议的具体实现细节，只需关心一个核心的问题：应用程序数据从哪里来到哪里去，当我明

白了数据是怎们来的怎们去的，以及代表的都是什么含义，整个系统在组网架构的显示就解决出来了。

在学习的过程中,我们坚持下来,也就慢慢理解了，一点点去实现，一个模块一个模块的去攻破，前期知识的积累，和学习能力才是硬台阶。

无论是 Java 还是 C#，他们只是个工具，算法是核心，思想很关键，学习能力一定要强，一定要耐心。这个系统还有很多可以扩展的功能，现实世界有着复杂的信息和关系，只有正确理清这些，明白自己的目标是什么，才能正确的提取有用的数据，实现应有的功能。

最重要的一点是我们做项目时,我们要有团队的合作精神。只有这样我们才能很好的完成--个好的项目,才能完成一个项目。我们要一起起讨论问题，只有这样才能进步更快,才能学到更多的知识。最后想说“一分耕耘一分收获”。

5.2 附录：主要代码

5.2.1 发送与读取数据

往串口发送数据

```
    public static void sendToPort(SerialPort serialPort,String data) throws
Exception {
        System.out.println("data = " + data);
        byte[] writerBuffer = null;
        OutputStream out = null;
        try {
            writerBuffer = Tran.hexStringToByte(data);
            System.out.println(writerBuffer);
        } catch (NumberFormatException e) {
        throw new Exception("命令格式错误！");
        }
        try {
            out = serialPort.getOutputStream();
            out.write(writerBuffer); out.flush();

        } catch (IOException e) {
            throw new SendDataToSerialPortFailure();
        } finally {
            try {
                if (out != null) {
                    out.close(); out = null;
                }
            } catch (IOException e) {
                throw new SerialPortOutputStreamCloseFailure();
            }
        }
```

```
    }
从串口读取数据
        InputStream in = null;
        byte[] bytes = null;
        try {
            in = serialPort.getInputStream();
            int bufflenth = in.available();        //获取buffer里的数据长度
            while (bufflenth != 0) {
                bytes = new byte[bufflenth];    //初始化byte数组为buffer中数据的
长度
                in.read(bytes);
                bufflenth = in.available();
            }
        } catch (IOException e) {
            throw new ReadDataFromSerialPortFailure();
        } finally {
            try {
                if (in != null) {
                    in.close();in = null;
                }
            } catch(IOException e) {
                throw new SerialPortInputStreamCloseFailure();
            }
        }
        return bytes;
}
```

5.2.2 初始数据处理（格式转换）

我们一开始获得的数据显示出来都是乱码的,所以我们发送和接收数据都要将其转为能够识别的数据。

```
//字节转为字符串
    public static String bytesToHex(byte[] bytes) {
        StringBuffer sb = new StringBuffer();
        for(int i = 0; i < bytes.length; i++) {
            String hex = Integer.toHexString(bytes[i] & 0xFF);
            if(hex.length() < 2){
                sb.append(0);
            }
            sb.append(hex);
        }
        return sb.toString();
    }
    /*字符串转为字节*/
```

```
    public static byte[] hexStringToByte(String hex) {
           int len = (hex.length() / 2);
           byte[] result = new byte[len];
           char[] achar = hex.toCharArray();
           for (int i = 0; i < len; i++) {
            int pos = i * 2;
            result[i] = (byte) (toByte(achar[pos]) << 4 | toByte(achar[pos + 1]));
           }
           return result;
    }
    private static int toByte(char c) {
        byte b = (byte) "0123456789ABCDEF".indexOf(c);
        return b;
     }
```

5.2.3 数据处理

根据现能使用的传感器主要对温湿度光敏度进行计算显示，以其判断控制LED与电机

```
String flag = dataOriginal1.substring(0, 10);
                        if (flag.matches("eecc010301")) {
                    String temp = dataOriginal1.substring(10, 14);// 提取温度
                    String hump = dataOriginal1.substring(14, 18);// 提取湿度
                                    Integer tt = Integer.parseInt(temp, 16);
                                    Integer hh = Integer.parseInt(hump, 16);
                                    if (tt * 0.01 < 100 && hh * 0.01 < 100) {
                        tem.setText(1.0 * tt / 100 + " ℃");// 显示温度
                        hum.setText(1.0 * hh / 100 + " %");// 显示湿度
String showL = showled.tosend(serialPort, tt * 0.01, hh * 0.01, tu, td,
                                                        hu, hd, textphoto,
textSmoke, textpa);
                    if (showL != null) pa.setText(showL);
                    if(showL.matches("一切正常"))
                    pa.setBackground(Color.green);
                        else pa.setBackground(Color.red);
    }
else if (flag.matches("eecc010201"))// 关于光照的处理--暂时用不到---也可以处理
                                        {// 平均照度1000Lux以上。
                String Ph = dataOriginal1.substring(10, 14);
                Integer pp = Integer.parseInt(Ph, 16);
                                        if (pp > 100) {
                    Photo.setText(pp + "Lux");
                String showPH = PHO.tosend(serialPort, pp);
                if (showPH != null)          LED.setText(showPH);
```

```
                            if(showPH.matches("一切正常"))
                                    LED.setBackground(Color.black);
                            else    LED.setBackground(Color.yellow);
                }
                                                }
                                                // 对于烟雾的处理
                                                if (flag.matches("eecc010701")) {// 接收
的烟雾传感器的信息
                                                    startTime = endTime;
                                                    moke.setText("疑似烟雾，启动灭火");
                                                } else {
                                                    endTime =
System.currentTimeMillis();
                                                }

                                                if (Math.abs(endTime - startTime) >=
3000) {
                                                    moke.setText("一切正常");
                                                    smo.tosend(serialPort, textTemp,
textHum, 0);
}}}}
```

5.2.4 用户登录（GUI 界面）

```
    public void actionPerformed(ActionEvent e) {
        String user = jTextField.getText();
        String pass = jPasswordField.getText();
        if (e.getActionCommand().equals("确认")) {
            int flag = con.judge(user, pass);
            System.out.println(flag);
            if (flag == 1) {
                // 跳转
                this.setVisible(false);
                new Client().launchFrame();
                // System.out.println("OK");
            } else {
                JOptionPane.showMessageDialog(null, "可能是密码错误或者是没有该
管理员哦ヽ(ノ▽ヽ)ノ");
            }
        } else {
            // 注册
            int flag = con.rejudge(user, pass);
            if (flag == 1) {
                JOptionPane.showMessageDialog(null, "注册成功啦，放心登录把ヽ(-
```

```
▽-) ╭″);
            } else {
                JOptionPane.showMessageDialog(null, ″注册失败，已经存在该用户了
╮(╯▽╰)╭″);
            }
        }
    }

    public static void main(String[] args) {
        new Enter();
    }
```

5.2.5 数据存储（连接数据库）

```
public class Connect {
    // 定义数据库访问参数
    static String url = ″jdbc:sqlserver://localhost:1433;DatabaseName=MyJava″;
    static String user = ″sa″;
    static String password = ″wsymz44″;
    static String driverName = ″com.microsoft.sqlserver.jdbc.SQLServerDriver″;
    static Connection conn;
    static Statement st;
    // 1、加载驱动
    static {
        try {
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            System.out.println(″驱动加载失败″);
        }
    }
    // 2、创建连接对象
    public static Connection getConnection() throws SQLException {
        conn = DriverManager.getConnection(url, user, password);
        return conn;
    }
    public static int redDate() // 读取数据函数
    {
        try {
            String sql = ″select * from MyDate_W″; // 这是sql语句中的查询某个表，
注意后面的emp是表名！！！
            st = getConnection().createStatement();
            ResultSet rs = st.executeQuery(sql); // 用于返回结果

            int temp = 0;
```

```
            int humm = 0;
            int led = 0;
            int phone = 0;
            while (rs.next()) {
                temp = rs.getInt("温度");
                humm = rs.getInt("湿度");
                led = rs.getInt("灯状态");
                phone = rs.getInt("光照");
                // System.out.println(id + "\t\t" + job + "\t" + date);
            }
            rs.close();
            return led;

        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("数据库数据读取成功！" + "\n");
        }
        return 0;
    }

    public static int judge(String username, String password) //
    {
        try {
            String sql = "select * from Manage"; // 这是sql语句中的查询某个表，
注意后面的emp是表名！！！
            st = getConnection().createStatement();
            ResultSet rs = st.executeQuery(sql); // 用于返回结果
            int lenu = username.length();
            int lenp = password.length();
            String name = "";
            String pass = "";
            while (rs.next()) {
                name = rs.getString("用户名");
                pass = rs.getString("密码");
                if (lenu > name.length() || lenp > pass.length())
                    continue;
                if (username.equals(name.substring(0, lenu)) &&
password.equals(pass.substring(0, lenp))) {
                    rs.close();
                    return 1;
                }
```

```
                // System.out.println(username+" "+password+" "+name+" "+pass);
            }
            rs.close();
            return 0;
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("数据库数据读取成功！" + "\n");
        }
        return 0;
    }
    public static int rejudge(String username, String password) // 读取数据函数
    {
        try {
            String sql = "select * from Manage"; // 这是sql语句中的查询某个表，
注意后面的emp是表名！！！
            st = getConnection().createStatement();
            ResultSet rs = st.executeQuery(sql); // 用于返回结果
            int lenu = username.length();
            int lenp = password.length();
            String name = "";
            String pass = "";
            while (rs.next()) {
                name = rs.getString("用户名");
                pass = rs.getString("密码");
                if (username.equals(name.substring(0, lenu))) {
                    rs.close();
                    return 0;// 已经有该用户了
                }
                // System.out.println(username+" "+password+" "+name+" "+pass);
            }
            rs.close();
            String sql2 = "insert into Manage (用户名,密码)" + "values(?,?)";
            PreparedStatement ps = getConnection().prepareStatement(sql2); // 用
preparedStatement预处理来执行sql语句
            ps.setString(1, username); // 给其五个参量分别“赋值”
            ps.setString(2, password); // 给其五个参量分别“赋值”
            ps.executeUpdate(); // 参数准备后执行语句
            ps.close();
            getConnection().close();

            return 1;
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("数据库数据读取成功！" + "\n");
        }
        return 0;
    }
    public static Object[][] showData() // 读取数据函数
    {
        try {
            Object[][] rowData = new Object[30][5];
            String sql = "select * from MyDate_W"; // 这是sql语句中的查询某个表，
注意后面的emp是表名！！！
            st = getConnection().createStatement();
            ResultSet rs = st.executeQuery(sql); // 用于返回结果
            int i = 0, j = 0;
            while (rs.next()) {
                rowData[i][j++] = rs.getDate("操作日期");
                rowData[i][j++] = rs.getFloat("温度");
                rowData[i][j++] = rs.getFloat("湿度");
                rowData[i][j++] = rs.getInt("光照");
                rowData[i][j++] = rs.getInt("灯状态");
                i++;
                j = 0;
                i=i%30;
            }
            rs.close();
            return rowData;
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // System.out.println("数据库数据读取成功！"+"\n");
        }
        return null;
    }
    public void add(double d, double e, int led, int y, int p, int g) throws
ClassNotFoundException, SQLException {
            String sql = "insert into MyDate_W (温度,湿度,灯状态,烟雾,光强反馈,
光照,操作日期)" + "values(?,?,?,?,?,?,?)";
        PreparedStatement ps = getConnection().prepareStatement(sql); // 用
```

```
preparedStatement预处理来执行sql语句
        ps.setDouble(1, d); // 给其五个参量分别“赋值”
        ps.setDouble(2, e); // 给其五个参量分别“赋值”
        ps.setInt(3, led);
        ps.setDouble(4, y);// yaqiang
        ps.setDouble(5, p);
        ps.setDouble(6, g);
        Date date = new Date(); // 获取一个Date对象
        Timestamp time = new Timestamp(date.getTime()); // 讲日期时间转换为数据
库中的timestamp类型
        ps.setTimestamp(7, time);
        ps.executeUpdate(); // 参数准备后执行语句
        ps.close();
        getConnection().close();
    }
```

5.2.6 历史数据

```
    public  History() {
        JFrame jf = new JFrame("历史数据");
        //jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        JPanel panel = new JPanel();
        //JPanel panel = new JPanel();
        String[] columnNames = { "日期", "温度", "湿度", "光敏", "灯状态" };
        Object[][] rowData = new Object[30][5];
        rowData = con.showData();
        JTable table = new JTable(rowData, columnNames);
        // 设置表格内容颜色
        table.setForeground(Color.BLACK); // 字体颜色
        table.setFont(new Font(null, Font.PLAIN, 14)); // 字体样式
        table.setSelectionForeground(Color.DARK_GRAY); // 选中后字体颜色
        table.setSelectionBackground(Color.LIGHT_GRAY); // 选中后字体背景
        table.setGridColor(Color.GRAY); // 网格颜色
        // 设置表头
        table.getTableHeader().setFont(new Font(null, Font.BOLD, 14)); // 设置表
头名称字体样式
        table.getTableHeader().setForeground(Color.RED); // 设置表头名称字体颜
色
        table.getTableHeader().setResizingAllowed(false); // 设置不允许手动改变
列宽
        table.getTableHeader().setReorderingAllowed(false); // 设置不允许拖动重
新排序各列
        //设置行高
        table.setRowHeight(30);
```

```
        //第一列列宽设置为80
        table.getColumnModel().getColumn(0).setPreferredWidth(80);
        // 设置滚动面板视口大小（超过该大小的行数据，需要拖动滚动条才能看到）
        table.setPreferredScrollableViewportSize(new Dimension(400, 300));
        JScrollPane scrollPane = new JScrollPane(table);
        panel.add(scrollPane);
        // 设置 内容面板 到 窗口
        jf.setContentPane(panel);
        jf.pack();
        jf.setLocationRelativeTo(null);
        jf.setVisible(true);
    }
```

5.2.7 网络拓扑显示

```
public DrawMap() {
        JLabel label = new JLabel("网络拓扑\n 显示");
        label.setBounds(10, 5, 120, 20);// 设置文本的位置
        area.setBounds(10, 40, 250, 250);// 设置文本域的位置
        frame.setLayout(null);
        frame.add(label);
        frame.add(area);
        frame.setBounds(100,100,300,300);
        frame.setLocation(300, 200);
        frame.setVisible(true);
        IDname[0]="协调器";IDname[2]="光敏";IDname[3]="温湿度";
        IDname[4]="震动";IDname[Integer.parseInt("af", 16)]="路由器";
        IDname[9]="电机 LED";
        for (int i = 0; i < vis.length; i++) vis[i] = -1;
        for (int i = 0; i < 256; i++)
            for (int j = 0; j < 256; j++)
                Mymap[i][j] = 0;
        show();
    }
    public void update(String s) {
        int devID=Integer.parseInt(s.substring(6,8),16);
        int myIp = Integer.parseInt(s.substring(10, 12), 16);
        int myMac = Integer.parseInt(s.substring(12, 14), 16);
        int faIp = Integer.parseInt(s.substring(14, 16), 16);
        int faMac = Integer.parseInt(s.substring(16, 18), 16);
        change[myIp]=devID;
        System.out.println(myIp+"  "+devID);
        if (vis[myIp] == -1) {// 没有出现过添加
            vis[myIp] = faIp;
```

```
            Mymap[faIp][myIp] = 1;
            show();
        } else if (vis[myIp] == faIp) {// 出现过
            return;
        } else {// 出现 但是时别的节点的下
            Mymap[vis[myIp]][myIp] = 0;
            Mymap[faIp][myIp] = 1;
            vis[myIp] = faIp;
            show();
        }
    }

    public void show() {
        String tmp = " |\n";
        String tmp2 = "     --->";
        String tmp3 = " -->";
        String s = IDname[change[0]]+"   \n";
        for (int i = 1; i < 256; i++) {
            if (Mymap[0][i] == 1) {
                s += tmp + tmp3 + IDname[change[i]]+"ip: "+i+"\n";
                for (int j = 1; j < 256; j++) {
                    if (Mymap[i][j] == 1) {
                        s +="    "+ tmp + tmp2 + IDname[change[j]]+"ip: "+j+"\n";
                    }
                }
            }
        }
        area.setText(s);
    }
```