物联网综合实践报告



题目：窗户自动控制系统

学院：计算机科学学院

专业年级：物联网工程2013级

指导老师：杨明根、邹正伟、周瑞东

小组成员：王艺洁、周星宇、岳林、陈开勇、汪国兴

2016-7月

选题背景

随着人们生活水平的提高，人们对其住宅的要求也越来越高。人们的生活环境随着窗户的诞生得到了很大程度上的改变，从其在现实生活中的应用来看，它可以用作房屋采光通风的洞口，也是家庭装饰的一部分，人类可以通过它来感受大自然的美和吸收大自然新鲜清爽的空气。经研究表明，如果室内的通风条件不好，室内的空气流通不畅，不但会影响人类的心情，更容易滋生病毒，危害人类的健康，可见窗户的诞生给人类的生活带来了极大的方便与益处。随着科技及电子技术的发展，光控及湿度控制自能窗户系统设计给人们的生活带来了极大的方便。为了保证日常生活环境的愉悦，关键的问题就是室内的通风。但通常的方法就是通过人工控制，但这种方法不但费时费力，同时在很多时候人们没法及时高效率的完成这些看是简单的工作。热门不仅希望拥有舒适、温馨的住所，而且对其安全性、智能性等方面也提出了更高的要求。现在的自动控制系统已经为我们的生活提供了太多的便利。考虑到我校的化工特色,各种化工实验室普遍存在做实验会有各种类型的污染出现,因此保持通风换气显得尤为重要,除了实验室必须具备通风换气设备外,通过窗户通风换气也是必不可少的。窗户的开关关系到室内环境的清新，与居住者的健康有很大的关系。且有的同学离开实验室时忘记关窗户碰到刮风下雨天会给实验室带来损失，因此设计一个自动开关窗系统很有必要。

自动窗控制系统的工作原理

自动窗的控制由驱动部分和传感器两部分组成，每栋大楼的窗户之多有成百上千个，在每个窗户上添加一个温度，湿度传感器，用于检测外界的天气状况，并在每个窗户的下方安装上直流电机与齿轮齿条机构，并将检测到的天气的温度，湿度状况信息传送到PC上，通过PC向直流电机发送信号，电机上电后转动，带动电机上的齿轮转动，再通过齿轮与窗户下的齿条啮合从而带动窗户的关闭与开启，保证室内良好的通风和防潮。

自动窗的发展现状

随着国民经济的不断增长与生活水平的不断提高，我国对建筑节能的需求和认识在逐步提高，并已从无认识无法规的阶段逐步进入了高度重视与分布明确实施的法规化阶段，1996年颁布的《民用建筑节能设计标准》要求建筑节能指标提高50%，1997年颁布的《国家化学建材推广应用“九五”计划和2010年规划纲要》规定新建节能建筑和既有住宅节能改造工程必须使用节能窗，优先使用塑料窗，建设部、国家经贸委、质量技监局、建材局文件，建住房[1999]295号《关于在住宅建设中淘汰落后产品的通知》中明确指出“积极开发生产、推广应用木塑复合窗、钢塑复合共挤节能保温窗及隔热保温型铝合金窗等新型节能窗”。以及其它建筑节能法规的颁布实施，表明建筑节能的更进一步发展，已经成为我国经济技术发展中的一项重大举措。在我国的建筑门窗产品市场上，铝门窗产品占的比例最大，为55%;其次是塑料门窗，占的比例为35%;钢门窗产品有6%的份额;其它材料的产品占了剩余的4%。我国金属门窗拥有巨大的市场和广阔的发展空间。《建设情况通报》[建设部办公厅2000年2月18日第四期]有数字显示：从1996年到1998年，每年分别建成城乡住宅12.22亿、12.75亿平方米，我们在2001年—2010年间，按全国新建住宅及旧房改造每年需要12亿平米的建筑面积，建筑门窗量按20%计算，每年需要的门窗总量约为2.4亿平米，铝合金门窗量约9000万-1.3亿平米，塑料门窗量约3600万-7200万平米，其他为铝木复合窗，木窗，钢窗等。隐形纱窗约占门窗总量的4∕1左右，约为6000万平米，可见需求之大。在近年的几年来，欧洲、北美、日、韩地区，住宅窗体产品启闭方式已开始向自动化方向发展。如德国的诺托、意大利的图兰朵、韩国的LG等，均有自动化窗体生产。在中国，自动窗正在逐步为人们所重视，不仅各种进口的自动窗和自动开窗器出现在中国的市场上，而且以天韵自动窗为代表的具备完全的自主知识产权，技术先进，功能完善的国产自动窗逐步成为国内市场的主体，越来越多的自动窗体被应用到各种建筑当中。由于自动窗是刚刚发展起来的新产品，还没有形成成熟、稳定的行业，尚未形成统一的行业标准和规范，因此目前市场上的自动窗产品在各个方面的差异非常大，多数产品还不能算是成熟、完善的自动窗。

自动窗控制系统领域的发展趋势

进入21世纪以来，特别在我国加入WTO后，国内各类产品均面临巨大挑战，各行各业特别是我国的传统产业因生产技术较为落后为适应时代的潮流而必须要不断的改进技术引进先进的电子设备和控制设备。例如室内空气质量是影响人们日常生活的重要因素，但传统的人们对室内的通风及阴雨天对窗户的控制主要依靠人工来完成，这种控制方法需要耗费大量的劳动力，对于高楼大厦的窗户来说不但多而且高，对工人来说具有一定的危险性，同时遇到阴天下雨时，这么多窗户很多时候不能够及时的关闭窗户，就情况来讲，传统的方法已不能满足现代的生活要求，随着科学技术的不断发展，人们逐渐意识到要摆脱原始的生活方式，采用现代科学技术来应对日新月异的高科技术，调整自己的生活方式。自动窗的设计不但解决了人工开关窗费时费力的问题，还可根据外界的湿度状况来控制开关窗，可以有效地解决室内因湿度过高而发霉发潮的问题。

设计方案选择与论证

自动窗控制系统有许多的特点：总体上可以从价格和结构两方面来讲，自动窗不但结构简单，更是价格低廉，一般情况下，自动窗是以直流电机作为动力的，和其它类型的自动窗和这种相比结构都比较简单，价格也相对较低。靠温湿度传感器件动作，本设计的一个主要优点就是可靠性较高，因在+5V电压下便可以工作，对于那些位置较高、不方便手动启闭的外窗来说，实用性更好。但是，因监测系统会受到外界各种因素的干扰会产生较大的偏差；又由于检测环境的复杂、以及各种干扰的影响，检测系统的稳定性和可靠性均会有不同程度的下降。

概要设计

窗户自动控制开关系统采用先进的ZigBee无线技术而设计。系统硬件由PC、温湿度传感器、协调器、直流电机组成。首先根据窗外的温湿度检测后将数据通过无线传输的方式发送给协调器，然后协调器再通过串口发送给pc，pc接收到数据后进行实时的检测，然后将数据存入数据库中，并且在pc上的上位机可以控制发送给直流电机的数据，然后将数据通过串口发送给直流电机，然后实现他的自动控制。

湿度传感器的选择

506-97型温湿度传感器芯片为原装进口传感器，有着卓越的稳定性、一致性，采用RS485接口,可实现多点同时监测,组网并远传，现场数字显示等功能，RS485接口方式使用了工业领域广泛应用的Modbus数据通讯协议进行数据通讯，可与各类控制设备、显示仪表和计算机等直接连接。

技术参数

|  |  |
| --- | --- |
| **参数** | |
|  | 技术指标 |
| 测量范围 | 湿度:0%RH～-100%RH 温度: 0℃～60℃ |
| 测量精度20%RH—80%RH@25℃ | 湿度:±3.0%RH  温度:±0.4℃ |
| 接口方式 | RS485 |
| 通讯距离 | 小于1200m |
| 通讯协议 | 标准Modbus |
| 响应时间 | 15s |
| 可编地址 | 1-255 |
| 波特率 | 1200-38400 |
| 供电电压 | 12V DC-24V DC （吸顶安装9-12VDC） |
| 工作温度 | -10℃～80℃ |
| 功耗 | 小于4W |

### 传感器的参数设置

本温湿度传感器初次使用时必须进行初始化设置，以完成设备与上位机通讯的参数设置。可设置参数包括：波特率设置（1200bps~38400bps），通讯地址设置（001->255）。具体设置方法如下：

***注****：M：功能键、确认键，C：退出键，▲：上调键，▼下调键。*

1）进入设置

长按M键，液晶显示出密码输入菜单，按▲键和▼键来调节，输入默认密码“506”进入设置（此密码不可更改）。

2）地址设置

长按M键，液晶出现设置菜单，第一页为地址设置，地址设置范围为1~255。可以通过▲键和▼键来调节地址码，调节完毕后再次按下M键，确认设置，按C键退出设置状态。

3）波特率设置

长按M键，液晶出现设置菜单，再次按M键切换到第2页波特率设置界面，波特率设置范围为1200~38400，可以通过▲键和▼键来调节波特率，调节完毕后第三次按下M键，确认设置，按C键退出设置状态。

4）奇偶校验设置

长按M键，液晶出现设置菜单，再次按M键切换到第3页奇偶校验设置界面，设置顺序为：0：无校验；1：奇校验；2：偶校验。可以通过▲键和▼键来调节，调节完毕后第三次按下M键，确认设置，按C键退出设置状态。

### 硬件电路连线图

传感器以标准RS485为通讯接口，通讯协议采用工业领域广泛应用的Modbus通讯协议，多台传感器备通过485总线串行连接，每台设备外接电源单独供电，总线端点通过数字信号转换器同计算机串口连接，具体组网方式和接线端子连接方式参见图5、图6。

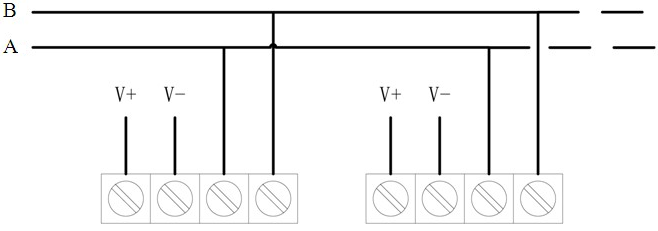


图1.1.1 引线连接示意图

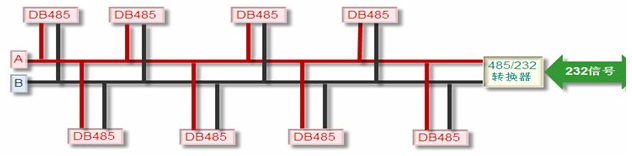


图 1.1.2 组网示意图

### 数据采集

本传感器的数据采集方法：利用串口向该传感器发送读取数据命令，传感器收到命令后会返回温度、湿度的值。命令的报文格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **发送内容** | **字节数** | **发送数据** | **备注** |
| 从机地址 | 1 | XXH | 从机地址 |
| 功能码 | 1 | 03H | 读取寄存器 |
| 起始寄存器地址 | 2 | 0000H | 该寄存器中保存设备的父类型和子类型 |
| 读取寄存器数量 | 2 | 0003H | 读取3个字（6个字节） |
| CRC校验 | 2 | XXXXH | 前面所有数据的CRC码 |

从机返回的报文格式：

|  |  |  |  |
| --- | --- | --- | --- |
| **发送内容** | **字节数** | **发送数据** | **备注** |
| 从机地址 | 1 | XXH | 从机地址 |
| 功能码 | 1 | 03H | 读取寄存器应答 |
| 返回字节长度 | 1 | 06H | 返回6个字节 |
| 返回数据 | 6 | XXYYZZH | XX为湿度数据，YY为温度数据，ZZ为露点数据 |
| CRC校验 | 2 | XXXXH | 前面所有数据的CRC码 |

通过其发送过来的数据，在PC上进行分析它的数据帧格式得到如下：

FE（帧头）

17（长度）

69 80（功能码1，2）

B7 51 00 00 7B

41 30 3D 32 38 2E 38 39 2C (a0=xx.xx,){温度}

41 31 3D 35 35 2E 30 37 7D (a1=xx.xx}){湿度}

3F（效验码）

主控方案的选择

采用单片机主控方案。采用集成的单片机主控，通过光照传感器和雨水传感器，将结果传送到单片机控制的主控器，数据通过显示器显示。如下图所示

协调器

PC

湿度传感器

串口

无线传输

无线传输

直流电机

硬件设计

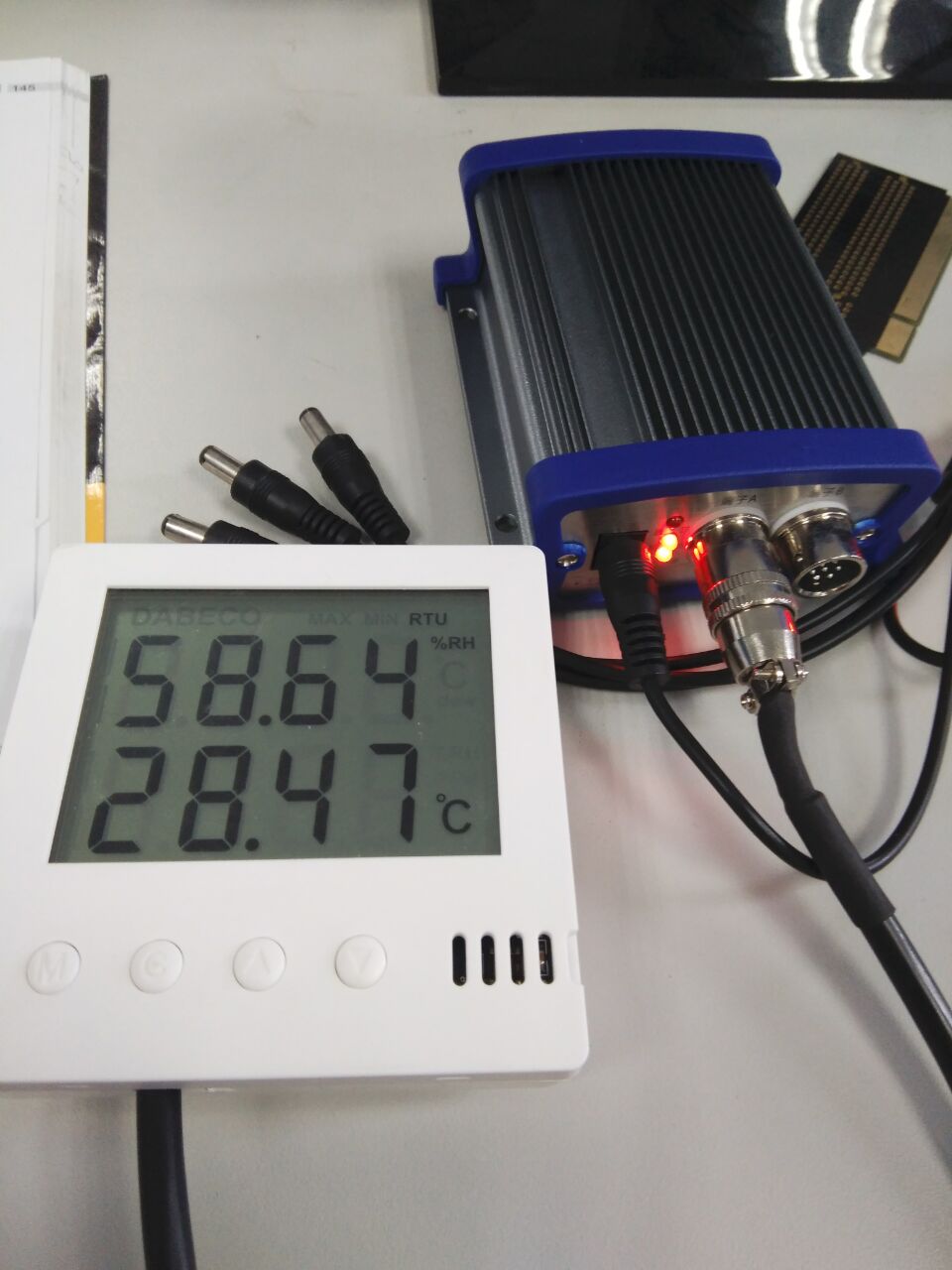
1.**协调器**:协调器用来无线接收传感器发送过来的数据，并且将受到的数据通过串口的方式发送给PC端。

2.**温，湿度传感器**:感知外部温湿度通过无限的方式发送，五秒发送来达到实时监控站。

3.**直流电机** : 直流电机用来控制窗户的打开于关闭。

4.PC：主机用于接收和处理协调器传送过来的数据。





软件设计

1．通过java设计一个温度的实时监控窗口

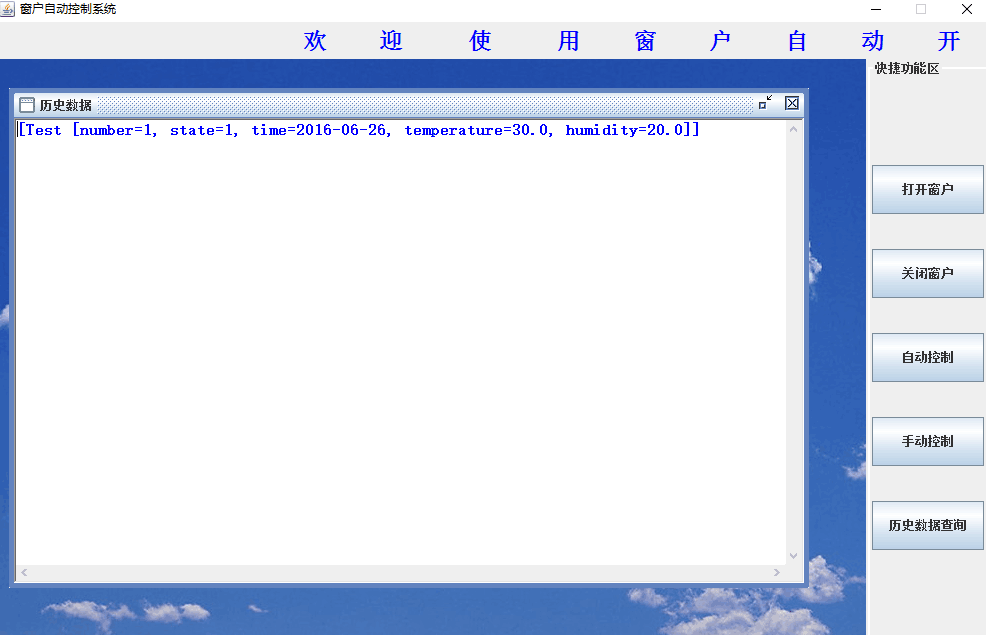
2．再设置一个自动控制按钮和手动控制按钮

3．设置打开和关闭窗户开关



数据库设计

1. 存储数据。（窗口编号，开关状态，时间，温度，湿度）
2. 查看历史数据。



一、软件设计的代码：

1. 串口接收与发送数据的代码：

/\*

\* SerialBean

SerialBean是本类库与其他应用程序的接口。该类库中定义了SerialBean的构造方法以及初始化串口，从串口读取数据，往串口写入数据以及关闭串口的函数。具体介绍如下：

public SerialBean(int PortID)

本函数构造一个指向特定串口的SerialBean，该串口由参数PortID所指定。PortID = 1 表示COM1，PortID = 2 表示COM2，由此类推。

public int Initialize()

本函数初始化所指定的串口并返回初始化结果。如果初始化成功返回1，否则返回-1。初始化的结果是该串口被SerialBean独占性使用，其参数被设置为9600, N, 8, 1。如果串口被成功初始化，则打开一个进程读取从串口传入的数据并将其保存在缓冲区中。

public String ReadPort(int Length)

本函数从串口(缓冲区)中读取指定长度的一个字符串。参数Length指定所返回字符串的长度。

public void WritePort(String Msg)

本函数向串口发送一个字符串。参数Msg是需要发送的字符串。

public void ClosePort()

本函数停止串口检测进程并关闭串口。

另注配置：

将javacomm20-win32 .zip下载的文件解压缩后，在\javacomm20-win32\commapi目录下有必需的三个文件：

comm.jar，javax.comm. properties和win32comm.dll。

将文件comm.jar拷贝到%JAVA\_HOME%\jre\lib\ext;

文件 javax.comm. properties拷贝到%JAVA\_HOME%\jre\lib;

文件win32comm.dll拷贝到%JAVA\_HOME%\bin。

注意%JAVA\_HOME%是jdk的路径，而非jre。

\*/

package serial;

import java.io.\*;

import java.util.\*;

import javax.comm.\*;

public class SerialBean {

static String PortName;

CommPortIdentifier portId;

SerialPort serialPort;

static OutputStream out;

static InputStream in;

SerialBuffer SB;

ReadSerial RT;

/\*\*

\*

\* Constructor

\*

\* @param PortID the ID of the serial to be used. 1 for COM1,

\* 2 for COM2, etc.

\*

\*/

public SerialBean(int PortID)

{

PortName = "COM" + PortID;

System.out.println(PortName);

}

/\*\*

\*

\* This function initialize the serial port for communication. It startss a

\* thread which consistently monitors the serial port. Any signal capturred

\* from the serial port is stored into a buffer area.

\*

\*/

public int Initialize()

{

int InitSuccess = 1;

int InitFail = -1;

try

{

portId = CommPortIdentifier.getPortIdentifier(PortName);

try

{

serialPort = (SerialPort)

portId.open("Serial\_Communication", 2000);

} catch (PortInUseException e)

{

System.out.println("1try");

return InitFail;

}

//Use InputStream in to read from the serial port, and OutputStream

//out to write to the serial port.

try

{

in = serialPort.getInputStream();

out = serialPort.getOutputStream();

} catch (IOException e)

{

System.out.println("2try");

return InitFail;

}

//Initialize the communication parameters to 9600, 8, 1, none.

try

{

serialPort.setSerialPortParams(38400,

SerialPort.DATABITS\_8,

SerialPort.STOPBITS\_1,

SerialPort.PARITY\_NONE);

} catch (UnsupportedCommOperationException e)

{

System.out.println("3try");

return InitFail;

}

} catch (NoSuchPortException e)

{

System.out.println("4try");

return InitFail;

}

// when successfully open the serial port, create a new serial buffer,

// then create a thread that consistently accepts incoming signals from

// the serial port. Incoming signals are stored in the serial buffer.

SB = new SerialBuffer();

RT = new ReadSerial(SB, in);

RT.start();

// return success information

return InitSuccess;

}

/\*\*

\*

\* This function returns a string with a certain length from the incomin

\* messages.

\*

\* @param Length The length of the string to be returned.

\*

\*/

public String ReadPort(int Length)

{

String Msg;

Msg = SB.GetMsg(Length);

return Msg;

}

/\*\*

\*

\* This function sends a message through the serial port.

\*

\* @param Msg The string to be sent.

\*

\*/

public void WritePort(String Msg)

{

int c;

try

{

for (int i = 0; i < Msg.length(); i++)

out.write(Msg.charAt(i));

} catch (IOException e) {}

}

/\*\*

\*

\* This function closes the serial port in use.

\*

\*/

public void ClosePort()

{

RT.stop();

serialPort.close();

}

}

/\*

\* ReadSerial

ReadSerial是一个进程，它不断的从指定的串口读取数据并将其存放到缓冲区中。

public ReadSerial(SerialBuffer SB, InputStreamPort)

本函数构造一个ReadSerial进程，参数SB指定存放传入数据的缓冲区，参数Port指定从串口所接收的数据流。

public void run()

ReadSerial进程的主函数，它不断的从指定的串口读取数据并将其存放到缓冲区中。

\*/

package serial;

import java.io.\*;

public class ReadSerial extends Thread {

private SerialBuffer ComBuffer;

private InputStream ComPort;

/\*\*

\*

\* Constructor

\*

\* @param SB The buffer to save the incoming messages.

\* @param Port The InputStream from the specific serial port.

\*

\*/

public ReadSerial(SerialBuffer SB, InputStream Port)

{

ComBuffer = SB;

ComPort = Port;

}

public void run()

{

int c;

try

{

while (true)

{

c = ComPort.read();

ComBuffer.PutChar(c);

}

} catch (IOException e) {}

}

}

/\*

\* SerialBuffer

SerialBuffer是本类库中所定义的串口缓冲区，它定义了往该缓冲区中写入数据和从该缓冲区中读取数据所需要的函数。

public synchronized String GetMsg(int Length)

本函数从串口(缓冲区)中读取指定长度的一个字符串。参数Length指定所返回字符串的长度。

public synchronized void PutChar(int c)

本函数望串口缓冲区中写入一个字符，参数c 是需要写入的字符。

在往缓冲区写入数据或者是从缓冲区读取数据的时候，必须保证数据的同步，因此GetMsg和PutChar函数均被声明为synchronized并在具体实现中采取

措施实现的数据的同步。

\*/

package serial;

public class SerialBuffer {

private String Content = "";

private String CurrentMsg, TempContent;

private boolean available = false;

private int LengthNeeded = 1;

/\*\*

\*

\* This function returns a string with a certain length from the incomin

\* messages.

\*

\* @param Length The length of the string to be returned.

\*

\*/

public synchronized String GetMsg(int Length)

{

LengthNeeded = Length;

notifyAll();

if (LengthNeeded > Content.length())

{

available = false;

while (available == false)

{

try

{

wait();

} catch (InterruptedException e) { }

}

}

CurrentMsg = Content.substring(0, LengthNeeded);

TempContent = Content.substring(LengthNeeded);

Content = TempContent;

LengthNeeded = 1;

notifyAll();

return CurrentMsg;

}

/\*\*

\*

\* This function stores a character captured from the serial port to the

\* buffer area.

\*

\* @param t The char value of the character to be stored.

\*

\*/

public synchronized void PutChar(int c)

{

Character d = new Character((char) c);

Content = Content.concat(d.toString());

if (LengthNeeded < Content.length())

{

available = true;

}

notifyAll();

}

}

1. 界面窗口代码：

package win.data;

import java.util.Date;

public class WindowsData {

private int number;// 窗口编号

private int state;// 1开，0关

private Date time;// 时间

private float temperature;// 温度

private float humidity;// 湿度

public WindowsData(int number, int state, Date time, float temperature,

float humidity) {

// super();

this.number = number;

this.state = state;

this.time = time;

this.temperature = temperature;

this.humidity = humidity;

}

public WindowsData() {

// super();

}

public int getNumber() {

return number;

}

public void setNumber(int number) {

this.number = number;

}

public int getState() {

return state;

}

public void setState(int state) {

this.state = state;

}

public Date getTime() {

return time;

}

public void setTime(Date time) {

this.time = time;

}

public float getTemperature() {

return temperature;

}

public void setTemperature(float temperature) {

this.temperature = temperature;

}

public float getHumidity() {

return humidity;

}

public void setHumidity(float humidity) {

this.humidity = humidity;

}

@Override

public String toString() {

return "Test [number=" + number + ", state=" + state + ", time=" + time

+ ", temperature=" + temperature + ", humidity=" + humidity

+ "]";

}

}

1. 连接数据库、初始化 代码：

package win.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBUtil {

private static final String URL="jdbc:mysql://127.0.0.1:3306/windowsdata?useUnicode=true&amp;characterEncoding=utf-8";

private static final String USER="root";

private static final String PASSWORD="123456";

private static Connection conn=null;

static {

try {

//1.加载驱动程序

Class.forName("com.mysql.jdbc.Driver");

//2.获得数据库的连接

conn=DriverManager.getConnection(URL, USER, PASSWORD);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static Connection getConnection(){

return conn;

}

}

1. 向数据库中写入数据：

package win.db;

import java.sql.Connection;

import java.sql.Date;

import java.sql.PreparedStatement;

import win.data.WindowsData;

public class DuAndXue {

public static void add(WindowsData t) throws Exception{

Connection conn=DBUtil.getConnection();

String sql="" +

"insert into test " +

"values(" +

"?,?,?,?,?)";

PreparedStatement ptmt=conn.prepareStatement(sql);

ptmt.setInt(1, t.getNumber());

ptmt.setInt(2, t.getState());

ptmt.setDate(3, new Date(t.getTime().getTime()));

ptmt.setFloat(4, t.getTemperature());

ptmt.setFloat(5, t.getHumidity());

//System.out.println("链接失败");

ptmt.execute();

}

}

1. Main（）函数：

package View;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.EventQueue;

import java.awt.Font;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Date;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JDesktopPane;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import serial.SerialBean;

import win.data.WindowsData;

import win.db.DuAndXue;

public class MainView extends JFrame {

public static void main(String[] args) throws Exception {

new MainView();

}

/\*\*

\*

\*/

private int flag = 1;// 默认自动控制

private static final long serialVersionUID = 1L;

private JPanel main\_panel = null;// 主面板

private JPanel wel\_panel = null;// 欢迎面板

private JPanel btn\_panel = null;// 按钮面板

private JPanel windw\_01 = null;

private JPanel windw\_02 = null;

private JPanel windw\_03 = null;

private JPanel windw = null;

private JDesktopPane funcDesktop = null;// 桌面面板

private JTextField con\_state = null; // 控制方式

private JTextField text\_number = null;

private JTextField text\_state = null;

private JTextField text\_temperature = null;

private JTextField text\_humidity = null;

private JTextField con\_state2 = null; // 控制方式

private JTextField text\_number2 = null;

private JTextField text\_state2 = null;

private JTextField text\_temperature2 = null;

private JTextField text\_humidity2 = null;

private JTextField con\_state3 = null; // 控制方式

private JTextField text\_number3 = null;

private JTextField text\_state3 = null;

private JTextField text\_temperature3 = null;

private JTextField text\_humidity3 = null;

private JButton open\_window = null;// 开启窗户

private JButton close\_window = null;// 关闭窗户

private JButton btn\_self = null;// 手动

private JButton btn\_hand = null;// 自动

private JLabel lb\_con\_state = null;// 控制方式标签

private JLabel lb\_number = null;

private JLabel lb\_state = null;

private JLabel lb\_temperature = null;

private JLabel lb\_humidity = null;

private JLabel lb\_con\_state2 = null;// 控制方式标签

private JLabel lb\_number2 = null;

private JLabel lb\_state2 = null;

private JLabel lb\_temperature2 = null;

private JLabel lb\_humidity2 = null;

private JLabel lb\_con\_state3 = null;// 控制方式标签

private JLabel lb\_number3 = null;

private JLabel lb\_state3 = null;

private JLabel lb\_temperature3 = null;

private JLabel lb\_humidity3 = null;

private JLabel deskLabel = null;// 图片标签

private JLabel lb\_welcom = null;// 欢迎标题

public MainView() throws Exception {

WindowsData winData = new WindowsData();

init();

WindowsControl(winData);

}

public void init() {

main\_panel = new JPanel(new BorderLayout());

btn\_panel = new JPanel(new GridLayout(7, 1, 0, 35));

windw\_01 = new JPanel(new GridLayout(10, 1, 0, 10));

windw\_02 = new JPanel(new GridLayout(10, 1, 0, 10));

windw\_03 = new JPanel(new GridLayout(10, 1, 0, 10));

windw = new JPanel(new GridLayout(1, 3, 10, 0));

open\_window = new JButton("打开窗户");

close\_window = new JButton("关闭窗户");

btn\_self = new JButton("自动控制");

btn\_hand = new JButton("手动控制");

///////////

lb\_con\_state = new JLabel("控制方式");

lb\_number = new JLabel("窗口编号1");

lb\_state = new JLabel("窗口状态");

lb\_temperature = new JLabel("空气温度");

lb\_humidity = new JLabel("空气湿度%");

con\_state = new JTextField();

text\_number = new JTextField();

text\_state = new JTextField();

text\_temperature = new JTextField();

text\_humidity = new JTextField();

//设置文本框不可写

con\_state.setEditable(false);

con\_state.setBackground(Color.white);

text\_number.setEditable(false);

text\_number.setBackground(Color.white);

text\_state.setEditable(false);

text\_state.setBackground(Color.white);

text\_temperature.setEditable(false);

text\_temperature.setBackground(Color.white);

text\_humidity.setEditable(false);

text\_humidity.setBackground(Color.white);

///////////////

lb\_con\_state2 = new JLabel("控制方式");

lb\_number2 = new JLabel("窗口编号2");

lb\_state2 = new JLabel("窗口状态");

lb\_temperature2 = new JLabel("空气温度");

lb\_humidity2 = new JLabel("空气湿度%");

con\_state2 = new JTextField();

text\_number2 = new JTextField();

text\_state2 = new JTextField();

text\_temperature2 = new JTextField();

text\_humidity2 = new JTextField();

//设置文本框不可写

con\_state2.setEditable(false);

con\_state2.setBackground(Color.white);

text\_number2.setEditable(false);

text\_number2.setBackground(Color.white);

text\_state2.setEditable(false);

text\_state2.setBackground(Color.white);

text\_temperature2.setEditable(false);

text\_temperature2.setBackground(Color.white);

text\_humidity2.setEditable(false);

text\_humidity2.setBackground(Color.white);

///////////////

lb\_con\_state3 = new JLabel("控制方式");

lb\_number3 = new JLabel("窗口编号3");

lb\_state3 = new JLabel("窗口状态");

lb\_temperature3 = new JLabel("空气温度");

lb\_humidity3 = new JLabel("空气湿度%");

con\_state3 = new JTextField();

text\_number3 = new JTextField();

text\_state3 = new JTextField();

text\_temperature3 = new JTextField();

text\_humidity3 = new JTextField();

//设置文本框不可写

con\_state3.setEditable(false);

con\_state3.setBackground(Color.white);

text\_number3.setEditable(false);

text\_number3.setBackground(Color.white);

text\_state3.setEditable(false);

text\_state3.setBackground(Color.white);

text\_temperature3.setEditable(false);

text\_temperature3.setBackground(Color.white);

text\_humidity3.setEditable(false);

text\_humidity3.setBackground(Color.white);

// 初始化显示面板

con\_state.setText("自动");

windw\_01.add(lb\_con\_state);

windw\_01.add(con\_state);

windw\_01.add(lb\_number);

windw\_01.add(text\_number);

windw\_01.add(lb\_state);

windw\_01.add(text\_state);

windw\_01.add(lb\_temperature);

windw\_01.add(text\_temperature);

windw\_01.add(lb\_humidity);

windw\_01.add(text\_humidity);

windw\_02.add(lb\_con\_state2);

windw\_02.add(con\_state2);

windw\_02.add(lb\_number2);

windw\_02.add(text\_number2);

windw\_02.add(lb\_state2);

windw\_02.add(text\_state2);

windw\_02.add(lb\_temperature2);

windw\_02.add(text\_temperature2);

windw\_02.add(lb\_humidity2);

windw\_02.add(text\_humidity2);

windw\_03.add(lb\_con\_state3);

windw\_03.add(con\_state3);

windw\_03.add(lb\_number3);

windw\_03.add(text\_number3);

windw\_03.add(lb\_state3);

windw\_03.add(text\_state3);

windw\_03.add(lb\_temperature3);

windw\_03.add(text\_temperature3);

windw\_03.add(lb\_humidity3);

windw\_03.add(text\_humidity3);

windw.add(windw\_01);

windw.add(windw\_02);

windw.add(windw\_03);

windw.setOpaque(false);

windw\_01.setOpaque(false);

windw\_02.setOpaque(false);

windw\_03.setOpaque(false);

btn\_panel.add(new JLabel());// 用来填充的标签控件

btn\_panel.add(open\_window);

btn\_panel.add(close\_window);

btn\_panel.add(btn\_self);

btn\_panel.add(btn\_hand);

btn\_panel.add(new JLabel());// 用来填充的标签控件

// 设置面板边框

btn\_panel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createRaisedBevelBorder(), "快捷功能区"));

windw.setBorder(BorderFactory.createTitledBorder(BorderFactory.createRaisedBevelBorder(), "数据区"));

// 初始化欢迎面板

wel\_panel = new JPanel();

lb\_welcom = new JLabel("欢 迎 使 用 窗 户 自 动 开 关 系 统");

lb\_welcom.setFont(new Font("宋体", Font.BOLD, 23));

lb\_welcom.setForeground(Color.BLUE);

wel\_panel.add(lb\_welcom);

// 初始化桌面面板

funcDesktop = new JDesktopPane();

funcDesktop.setBorder(BorderFactory.createTitledBorder(BorderFactory.createRaisedBevelBorder(), "数据显示区"));

ImageIcon image = new ImageIcon("src/back.jpg");

deskLabel = new JLabel(image);

deskLabel.setBounds(0, 0, image.getIconWidth(), image.getIconHeight());

// deskLabel.add(windw);

funcDesktop.add(deskLabel, new Integer(Integer.MIN\_VALUE));

// 设置文本框字体

con\_state.setFont(new Font("宋体", Font.BOLD, 23));

con\_state.setForeground(Color.BLUE);

text\_number.setForeground(Color.BLUE);

text\_number.setFont(new Font("宋体", Font.BOLD, 23));

text\_number.setForeground(Color.BLUE);

text\_state.setFont(new Font("宋体", Font.BOLD, 23));

text\_state.setForeground(Color.BLUE);

text\_temperature.setFont(new Font("宋体", Font.BOLD, 23));

text\_temperature.setForeground(Color.BLUE);

text\_humidity.setFont(new Font("宋体", Font.BOLD, 23));

text\_humidity.setForeground(Color.BLUE);

main\_panel.add(btn\_panel, BorderLayout.EAST);

main\_panel.add(wel\_panel, BorderLayout.NORTH);

// main\_panel.add(funcDesktop,BorderLayout.CENTER);

main\_panel.add(windw, BorderLayout.CENTER);

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

new Thread(new DynaminThread()).start();

}

});

this.setTitle("窗户自动控制系统");

this.getContentPane().add(main\_panel);

this.setSize(1000, 650);

this.setVisible(true);

this.setResizable(false);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setLocationRelativeTo(null);

}

private void WindowsControl(WindowsData winData) throws Exception {

SerialBean SB = new SerialBean(1);

SB.Initialize();

System.out.println("初始化：" + SB.Initialize());

winData.setState(1);// 默认开窗

while (true) {

if (flag == 1) {

String Msg;

String Temperature;

String Humidity;

float temp;

float hum;

Msg = SB.ReadPort(28);

Date now = new Date();

Temperature = Msg.substring(12, 17);

Humidity = Msg.substring(21, 26);

temp = Float.parseFloat(Temperature);

hum = Float.parseFloat(Humidity);

winData.setNumber(1);

winData.setTime(now);

winData.setTemperature(temp);

winData.setHumidity(hum);

DuAndXue.add(winData);

System.out.println(winData.toString());

if (winData.getState() == 1) {

text\_state.setText("开");

} else {

text\_state.setText("关");

}

text\_number.setText(winData.getNumber() + "");

text\_temperature.setText(winData.getTemperature() + "");

text\_humidity.setText(winData.getHumidity() + "");

if ((winData.getHumidity() >= 95.25) && winData.getState() == 1) {

SB.WritePort(0 + "");// 关窗

winData.setState(0);

text\_state.setText("关");

// System.out.println(winData.getState());

}

if ((winData.getHumidity() < 95.25) && winData.getState() == 0) {

SB.WritePort(1 + "");// 开窗

winData.setState(1);

text\_state.setText("开");

// System.out.println(winData.getState());

}

registerListener(winData);

}

if (flag == 0) {

String Msg;

String Temperature;

String Humidity;

float temp;

float hum;

Msg = SB.ReadPort(28);

Date now = new Date();

Temperature = Msg.substring(12, 17);

Humidity = Msg.substring(21, 26);

temp = Float.parseFloat(Temperature);

hum = Float.parseFloat(Humidity);

winData.setNumber(1);

winData.setTime(now);

winData.setTemperature(temp);

winData.setHumidity(hum);

System.out.println(winData.toString());

if (winData.getState() == 1) {

text\_state.setText("开");

} else {

text\_state.setText("关");

}

text\_number.setText(winData.getNumber() + "");

text\_temperature.setText(winData.getTemperature() + "");

text\_humidity.setText(winData.getHumidity() + "");

}

}

}

private void registerListener(WindowsData windata) throws Exception {

SerialBean SB = new SerialBean(1);

open\_window.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

windata.setState(1);

SB.WritePort(1 + "");

text\_state.setText("开");

}

});

close\_window.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

windata.setState(0);

SB.WritePort(0 + "");

text\_state.setText("关");

}

});

btn\_self.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

flag = 1; // 自动

con\_state.setText("自动");

}

});

btn\_hand.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

flag = 0; // 手动

con\_state.setText("手动");

}

});

}

// 线程类，让欢迎字移动

private class DynaminThread implements Runnable {

@Override

public void run() {

while (true) {

for (int i = 1000; i > -750; i--) {

try {

Thread.sleep(10);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

lb\_welcom.setLocation(i, 5);

}

}

}

}

}

二、硬件设计的代码：

1．协调器的代码：

#include "ZComDef.h"

#include "OSAL.h"

#include "sapi.h"

#include "hal\_key.h"

#include "hal\_led.h"

#include "DebugTrace.h"

#include "SimpleApp.h"

#include "hal\_flash.h"

#if defined( MT\_TASK )

#include "osal\_nv.h"

#endif

#include "mt\_app.h"

#include "mt\_uart.h"

#include "mt.h"

#include "rtg.h"

#include "mac\_radio\_defs.h"

#include "AddrMgr.h"

#include "nwk\_util.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define USE\_SYS\_FIND\_DEVICE 0

#define NODE\_NAME "000"

#define NODE\_CATEGORY 1

#define OSAL\_NV\_PAGE\_BEG HAL\_NV\_PAGE\_BEG

#define OSAL\_NV\_PAGE\_END (OSAL\_NV\_PAGE\_BEG + HAL\_NV\_PAGE\_CNT - 1)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* CONSTANTS

\*/

// Application States

#define APP\_INIT 0

#define APP\_START 1

// Application osal event identifiers

#define \_START\_EVT 0x0010

#define \_\_REPORT\_EVT 0x0020

// Same definitions as in SimpleSensor.c

#define TEMP\_REPORT 0x01

#define BATTERY\_REPORT 0x02

#define REPORT\_DELAY 30

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TYPEDEFS

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LOCAL VARIABLES

\*/

static uint8 myAppState = APP\_INIT;

static uint8 myStartRetryDelay = 10;

#if defined( MT\_TASK )

extern uint8 aExtendedAddress[8];

#endif

static void my\_report\_proc(void);

static void process\_set\_command\_call(int (\*fun)(char \*ptag, char \*pval, char \*pout));

static void process\_package(char \*pkg, int len);

static int process\_command\_callback(char \*ptag, char \*pval, char \*pout);

void zb\_HanderMsg(osal\_event\_hdr\_t \*pMsg);

static void processCommand(uint16 cmd, byte \*dat, uint8 len);

void my\_FindDevice(uint8 searchType, uint8 \*searchKey);

int my\_FindDeviceProc( uint16 source, uint16 command, uint16 len, uint8 \*pData);

static char\* read\_al(char \*buf, int len);

static char\* read\_nb(char \*buf, int len);

static uint16 panid;

static uint8 logicalType;

static uint16 \_tm\_cnt, \_tm\_delay;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* GLOBAL VARIABLES

\*/

// Inputs and Outputs for Switch device

#define NUM\_OUT\_CMD\_COLLECTOR 2

#define NUM\_IN\_CMD\_COLLECTOR 3

// List of output and input commands for Switch device

const cId\_t zb\_InCmdList[NUM\_IN\_CMD\_COLLECTOR] =

{

ID\_CMD\_READ\_RES,

ID\_CMD\_WRITE\_RES,

ID\_CMD\_REPORT,

};

const cId\_t zb\_OutCmdList[NUM\_OUT\_CMD\_COLLECTOR] =

{

ID\_CMD\_READ\_REQ,

ID\_CMD\_WRITE\_REQ,

};

// Define SimpleDescriptor for Switch device

const SimpleDescriptionFormat\_t zb\_SimpleDesc =

{

MY\_ENDPOINT\_ID, // Endpoint

MY\_PROFILE\_ID, // Profile ID

DEV\_ID\_COLLECTOR, // Device ID

DEVICE\_VERSION\_COLLECTOR, // Device Version

0, // Reserved

NUM\_IN\_CMD\_COLLECTOR, // Number of Input Commands

(cId\_t \*) zb\_InCmdList, // Input Command List

NUM\_OUT\_CMD\_COLLECTOR, // Number of Output Commands

(cId\_t \*) zb\_OutCmdList // Output Command List

};

#define K4 P0\_1

#define K5 P0\_4

static void key\_init(void);

static void key\_init(void)

{

/\* P0.4 按键检测\*/

P0SEL &= ~0x12; //通用IO

P0DIR &= ~0x12; //作输入

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_HandleOsalEvent

\*

\* @brief The zb\_HandleOsalEvent function is called by the operating

\* system when a task event is set

\*

\* @param event - Bitmask containing the events that have been set

\*

\* @return none

\*/

void zb\_HandleOsalEvent( uint16 event )

{

uint8 startOptions;

if (event & ZB\_ENTRY\_EVENT)

{

key\_init();

if ( K5 == 0 || K4 == 0) {

/\*

extern void zgInitItems( uint8 setDefault );

uint32 ch;

osal\_nv\_read( ZCD\_NV\_PANID, 0, sizeof( panid ), &panid );

osal\_nv\_read( ZCD\_NV\_CHANLIST, 0, sizeof( ch ), &ch );

zgInitItems(TRUE);

uint8 pg;

for ( pg = OSAL\_NV\_PAGE\_BEG; pg <= OSAL\_NV\_PAGE\_END; pg++ )

{

HalFlashErase(pg);

}

osal\_nv\_write( ZCD\_NV\_PANID, 0, sizeof( panid ), &panid );

osal\_nv\_write( ZCD\_NV\_CHANLIST, 0, sizeof( ch ), &ch );

\*/

uint8 startOptions = ZCD\_STARTOPT\_DEFAULT\_NETWORK\_STATE;

zb\_WriteConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions ); //标记网络状态发生改变

while (K5 == 0 || K4 == 0) {

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_ON );

HalLedSet( HAL\_LED\_2, HAL\_LED\_MODE\_OFF );

//HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_BLINK );

}

zb\_SystemReset();

}

zb\_ReadConfiguration( ZCD\_NV\_LOGICAL\_TYPE, sizeof(uint8), &logicalType );

if ( logicalType != ZG\_DEVICETYPE\_COORDINATOR )

{

logicalType = ZG\_DEVICETYPE\_COORDINATOR;

zb\_WriteConfiguration(ZCD\_NV\_LOGICAL\_TYPE, sizeof(uint8), &logicalType);

}

zb\_ReadConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions );

if ((startOptions & ZCD\_STARTOPT\_AUTO\_START) == 0) {

zb\_StartRequest();

}

osal\_nv\_read( ZCD\_NV\_PANID, 0, sizeof( panid ), &panid );

HalLedSet( HAL\_LED\_2, HAL\_LED\_MODE\_FLASH );

process\_set\_command\_call(process\_command\_callback);

}

if ( event & \_START\_EVT )

{

zb\_StartRequest();

}

if ( event & \_\_REPORT\_EVT) {

if (\_tm\_cnt > 0) {

my\_report\_proc();

osal\_start\_timerEx( sapi\_TaskID, \_\_REPORT\_EVT, \_tm\_delay \* 1000);

\_tm\_cnt --;

}

}

}

void zb\_HanderMsg(osal\_event\_hdr\_t \*msg)

{

mtSysAppMsg\_t \*pMsg = (mtSysAppMsg\_t\*)msg;

uint16 dAddr;

uint16 cmd;

uint16 addr = NLME\_GetShortAddr();

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_OFF );

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_BLINK );

if (pMsg->hdr.event == MT\_SYS\_APP\_MSG) {

//if (pMsg->appDataLen < 4) return;

dAddr = pMsg->appData[0]<<8 | pMsg->appData[1];

cmd = pMsg->appData[2]<<8 | pMsg->appData[3];

if (dAddr != 0) {

zb\_SendDataRequest(dAddr, cmd, pMsg->appDataLen-4, pMsg->appData+4, 0, AF\_ACK\_REQUEST, AF\_DEFAULT\_RADIUS );

}

if (dAddr == 0 || dAddr == 0xffff) {

processCommand(cmd, pMsg->appData+4, pMsg->appDataLen-4);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_HandleKeys

\*

\* @brief Handles all key events for this device.

\*

\* @param shift - true if in shift/alt.

\* @param keys - bit field for key events. Valid entries:

\* EVAL\_SW4

\* EVAL\_SW3

\* EVAL\_SW2

\* EVAL\_SW1

\*

\* @return none

\*/

void zb\_HandleKeys( uint8 shift, uint8 keys )

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_StartConfirm

\*

\* @brief The zb\_StartConfirm callback is called by the ZigBee stack

\* after a start request operation completes

\*

\* @param status - The status of the start operation. Status of

\* ZB\_SUCCESS indicates the start operation completed

\* successfully. Else the status is an error code.

\*

\* @return none

\*/

void zb\_StartConfirm( uint8 status )

{

// If the device sucessfully started, change state to running

if ( status == ZB\_SUCCESS )

{

uint8 startOptions = ZCD\_STARTOPT\_AUTO\_START;

myAppState = APP\_START;

HalLedSet( HAL\_LED\_2, HAL\_LED\_MODE\_ON );

//zb\_AllowBind(0xff);

zb\_WriteConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions );

//osal\_start\_timerEx( sapi\_TaskID, \_\_REPORT\_EVT, (osal\_rand()%REPORT\_DELAY) \* 1000);

}

else

{

// Try again later with a delay

osal\_start\_timerEx( sapi\_TaskID, \_START\_EVT, myStartRetryDelay \* 1000);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_SendDataConfirm

\*

\* @brief The zb\_SendDataConfirm callback function is called by the

\* ZigBee after a send data operation completes

\*

\* @param handle - The handle identifying the data transmission.

\* status - The status of the operation.

\*

\* @return none

\*/

void zb\_SendDataConfirm( uint8 handle, uint8 status )

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_BindConfirm

\*

\* @brief The zb\_BindConfirm callback is called by the ZigBee stack

\* after a bind operation completes.

\*

\* @param commandId - The command ID of the binding being confirmed.

\* status - The status of the bind operation.

\*

\* @return none

\*/

void zb\_BindConfirm( uint16 commandId, uint8 status )

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_AllowBindConfirm

\*

\* @brief Indicates when another device attempted to bind to this device

\*

\* @param

\*

\* @return none

\*/

void zb\_AllowBindConfirm( uint16 source )

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_FindDeviceConfirm

\*

\* @brief The zb\_FindDeviceConfirm callback function is called by the

\* ZigBee stack when a find device operation completes.

\*

\* @param searchType - The type of search that was performed.

\* searchKey - Value that the search was executed on.

\* result - The result of the search.

\*

\* @return none

\*/

void zb\_FindDeviceConfirm( uint8 searchType, uint8 \*searchKey, uint8 \*result )

{

byte res[Z\_EXTADDR\_LEN+2];

if (ZB\_IEEE\_SEARCH == searchType) { //通过mac地址寻找对应的节点

osal\_memcpy(res, searchKey, Z\_EXTADDR\_LEN);

res[Z\_EXTADDR\_LEN] = result[1];

res[Z\_EXTADDR\_LEN+1] = result[0];

MT\_ReverseBytes( res, Z\_EXTADDR\_LEN );

zb\_ReceiveDataIndication(0, 0x0101, 8+2, res);

}

if (ZB\_NWKA\_SEARCH == searchType) { //通过网络地址寻找对应的节点

res[0] = searchKey[1];

res[1] = searchKey[0];

osal\_memcpy(res+2, result, Z\_EXTADDR\_LEN);

MT\_ReverseBytes( res+2, Z\_EXTADDR\_LEN ); //mac地址反转

zb\_ReceiveDataIndication(0, 0x0102, 8+2, res);//网络地址在前，mac地址在后发送给网关

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn zb\_ReceiveDataIndication

\*

\* @brief The zb\_ReceiveDataIndication callback function is called

\* asynchronously by the ZigBee stack to notify the application

\* when data is received from a peer device.

\*

\* @param source - The short address of the peer device that sent the data

\* command - The commandId associated with the data

\* len - The number of bytes in the pData parameter

\* pData - The data sent by the peer device

\*

\* @return none

\*/

void zb\_ReceiveDataIndication( uint16 source, uint16 command, uint16 len, uint8 \*pData )

{

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_OFF );

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_BLINK );

mtOSALSerialData\_t\* msg = (mtOSALSerialData\_t\*)osal\_msg\_allocate(sizeof(mtOSALSerialData\_t)+len+4);

if (msg) {

msg->hdr.event = MT\_SYS\_APP\_RSP\_MSG;

msg->hdr.status = len+4;

msg->msg = (byte\*)(msg+1);

msg->msg[0] = (source>>8)&0xff;

msg->msg[1] = source&0xff;

msg->msg[2] = (command>>8)&0xff;

msg->msg[3] = command&0xff;

osal\_memcpy(msg->msg+4, pData, len);

osal\_msg\_send( MT\_TaskID, (uint8 \*)msg );

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn myApp\_ReadTemperature

\*

\* @brief Reports temperature sensor reading

\*

\* @param

\*

\* @return

\*/

static int paramWrite(uint16 pid, byte \*dat)

{

int len = 0;

switch (pid) {

default:

break;

}

return len;

}

static int paramRead(uint16 pid, byte \*dat)

{

int len = 0;

switch (pid) {

case 0x0001:

dat[0] = 0x12; dat[1] = 0x09;

len = 2;

break;

case 0x0002:

dat[0] = 0x11; dat[1] = 0x44;

len = 2;

break;

case 0x0003:

dat[0] = 0x00; dat[1] = 0x01;

len = 2;

break;

case 0x0004:

dat[0] = dat[1] = dat[2] = dat[3] = dat[4] = dat[5] = 1;

len = 6;

break;

case 0x0005:

dat[0] = DEV\_ID\_COLLECTOR;

len = 1;

break;

/\* ----------- 网络参数 ------------------- \*/

case 0x0014: //mac地址

/\*osal\_nv\_read( ZCD\_NV\_EXTADDR, 0, Z\_EXTADDR\_LEN, pBuf ); rm by liren \*/

ZMacGetReq( ZMacExtAddr, dat ); // add by liren

// Outgoing extended address needs to be reversed

MT\_ReverseBytes( dat, Z\_EXTADDR\_LEN );

len = Z\_EXTADDR\_LEN;

break;

case 0x0015:

{

uint8 assocCnt = 0;

uint16 \*assocList;

int i;

#if defined(RTR\_NWK) && !defined( NONWK )

assocList = AssocMakeList( &assocCnt );

#else

assocCnt = 0;

assocList = NULL;

#endif

dat[0] = assocCnt;

for (i=0; i<assocCnt&&i<16; i++) {

dat[1+2\*i] = HI\_UINT16(assocList[i]);

dat[1+2\*i+1] = LO\_UINT16(assocList[i]);

}

len = 1 + 2 \* assocCnt;

break;

}

}

/\* ------------------------------------ \*/

return len;

}

#include "ZDApp.h"

static void processCommand(uint16 cmd, byte \*pData, uint8 len)

{

int i;

uint16 pid;

byte dat[64];

byte rlen = 1;

int ret;

switch (cmd) {

case 0x0000:

process\_package(pData, len);

break;

case 0x0101: //通过mac地址寻找对应的节点

{

uint8 \*pExtAddr = pData;

MT\_ReverseBytes( pExtAddr, Z\_EXTADDR\_LEN );

ZMacGetReq( ZMacExtAddr, dat ); //获取当前节点的mac地址

#if USE\_SYS\_FIND\_DEVICE

zb\_FindDeviceRequest(ZB\_IEEE\_SEARCH, pExtAddr);

#else

if (TRUE == osal\_memcmp(pExtAddr, dat, Z\_EXTADDR\_LEN) || //如果mac地址匹配

TRUE == osal\_memcmp(pData, "\x00\x00\x00\x00\x00\x00\x00\x00", Z\_EXTADDR\_LEN))

{

ret = 0;

zb\_FindDeviceConfirm(ZB\_IEEE\_SEARCH, pExtAddr, (unsigned char \*)&ret);

} else {

my\_FindDevice(ZB\_IEEE\_SEARCH, pExtAddr);

}

#endif

}

break;

case 0x0102: //通过网络地址寻找对应的节点

{

uint16 shortAddr = (pData[0]<<8) | pData[1];

uint16 sa = NLME\_GetShortAddr(); //获取当前节点的网络地址

if (shortAddr == sa) { //如果网络地址匹配

ZMacGetReq( ZMacExtAddr, dat ); //获取当前节点的mac地址

zb\_FindDeviceConfirm(ZB\_NWKA\_SEARCH, (unsigned char \*)&sa, dat);

} else {

#if USE\_SYS\_FIND\_DEVICE

ZDP\_IEEEAddrReq( shortAddr, ZDP\_ADDR\_REQTYPE\_SINGLE, 0, 0 );

#else

my\_FindDevice(ZB\_NWKA\_SEARCH, (uint8\*)pData);

#endif

}

}

break;

case ID\_CMD\_WRITE\_REQ:

for (i=0; i<len; i+=2) {

pid = pData[i]<<8 | pData[i+1];

ret = paramWrite(pid, &pData[i+2]);

if (ret <= 0) {

dat[0] = 1;

zb\_ReceiveDataIndication( 0, ID\_CMD\_WRITE\_RES, 1, dat );

return;

}

i += ret;

}

dat[0] = 0;

zb\_ReceiveDataIndication( 0, ID\_CMD\_WRITE\_RES, 1, dat);

break;

case ID\_CMD\_READ\_REQ:

for (i=0; i<len; i+=2) {

pid = pData[i]<<8 | pData[i+1];

dat[rlen++] = pData[i];

dat[rlen++] = pData[i+1];

ret = paramRead(pid, dat+rlen);

if (ret <= 0) {

dat[0] = 1;

zb\_ReceiveDataIndication( 0, ID\_CMD\_READ\_RES, 1, dat );

return;

}

rlen += ret;

}

dat[0] = 0;

zb\_ReceiveDataIndication( 0, ID\_CMD\_READ\_RES, rlen, dat );

break;

}

}

////////////////////////////////////////////////////////////////////////////////

static int (\*process\_command\_call)(char \*ptag, char \*pval, char \*pout);

static void process\_set\_command\_call(int (\*fun)(char \*ptag, char \*pval, char \*pout))

{

process\_command\_call = fun;

}

static char wbuf[256];

static void process\_package(char \*pkg, int len)

{

char \*p;

char \*ptag = NULL;

char \*pval = NULL;

char \*pwbuf = wbuf+1;

if (pkg[0] != '{' || pkg[len-1] != '}') return;

pkg[len-1] = 0;

p = pkg+1;

do {

ptag = p;

p = strchr(p, '=');

if (p != NULL) {

\*p++ = 0;

pval = p;

p = strchr(p, ',');

if (p != NULL) \*p++ = 0;

if (process\_command\_call != NULL) {

int ret;

ret = process\_command\_call(ptag, pval, pwbuf);

if (ret > 0) {

pwbuf += ret;

\*pwbuf++ = ',';

}

}

}

} while (p != NULL);

if (pwbuf - wbuf > 1) {

wbuf[0] = '{';

pwbuf[0] = 0;

pwbuf[-1] = '}';

uint16 cmd = 0;

zb\_ReceiveDataIndication( 0, cmd, pwbuf-wbuf, wbuf );

}

}

int process\_command\_callback(char \*ptag, char \*pval, char \*pout)

{

int val;

int ret = 0;

val = atoi(pval);

if (0 == strcmp("DEFNWS", ptag)) {

if (val == 1) {

uint8 startOptions = ZCD\_STARTOPT\_DEFAULT\_NETWORK\_STATE;

zb\_WriteConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions ); //标记网络状态发生改变

}

}else

if (0 == strcmp("ECHO", ptag)) {

ret = sprintf(pout, "ECHO=%s",pval);

} else

if (0 == strcmp("PANID", ptag)) {

if (0 == strcmp("?", pval)) {

uint16 tmp16;

osal\_nv\_read( ZCD\_NV\_PANID, 0, sizeof( tmp16 ), &tmp16 );

ret = sprintf(pout, "PANID=%u", tmp16);

} else {

uint8 startOptions = ZCD\_STARTOPT\_DEFAULT\_NETWORK\_STATE;

uint16 tmp16;

osal\_nv\_read( ZCD\_NV\_PANID, 0, sizeof( tmp16 ), &tmp16 );

if (tmp16 != val) {

osal\_nv\_write(ZCD\_NV\_PANID, 0, osal\_nv\_item\_len( ZCD\_NV\_PANID ), &val);

zb\_WriteConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions ); //标记网络状态发生改变

}

}

}

if (0 == strcmp("CHANNEL", ptag)) {

static uint32 chs[] = {0x00000800, 0x00001000, 0x00002000, 0x00004000, 0x00008000,

0x00010000, 0x00020000, 0x00040000,0x00080000,0x00100000,0x00200000,

0x00400000,0x00800000,0x01000000,0x02000000,0x04000000};

if (0 == strcmp("?", pval)) {

uint32 tmp32;

uint8 i;

osal\_nv\_read( ZCD\_NV\_CHANLIST, 0, sizeof( tmp32 ), &tmp32 );

for (i=0; i<16; i++) {

if (tmp32 == chs[i]) break;

}

i += 11;

ret = sprintf(pout, "CHANNEL=%u", i);

} else {

uint32 tmp32, t32;

uint8 startOptions = ZCD\_STARTOPT\_DEFAULT\_NETWORK\_STATE;

tmp32 = val - 11;

osal\_nv\_read( ZCD\_NV\_CHANLIST, 0, sizeof( tmp32 ), &t32 );

if (tmp32 < 16) {

if (t32 != chs[tmp32]) {

osal\_nv\_write(ZCD\_NV\_CHANLIST, 0, osal\_nv\_item\_len( ZCD\_NV\_CHANLIST ), &chs[tmp32]);

zb\_WriteConfiguration( ZCD\_NV\_STARTUP\_OPTION, sizeof(uint8), &startOptions ); //标记网络状态发生改变

}

}

}

}

/\*if (0 == strcmp("RSSI", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "RSSI=%d", rssi);

}

}\*/

if (0 == strcmp("TYPE", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "TYPE=%d%d%s", NODE\_CATEGORY, logicalType, NODE\_NAME);

}

}

if (0 == strcmp("NODE\_CATEGORY", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "NODE\_CATEGORY=%d", NODE\_CATEGORY);

}

}

if (0 == strcmp("NODE\_TYPE", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "NODE\_TYPE=%d", logicalType);

}else{

logicalType = val;

zb\_WriteConfiguration(ZCD\_NV\_LOGICAL\_TYPE, sizeof(uint8), &logicalType);

}

}

if (0 == strcmp("NODE\_NAME", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "NODE\_NAME=%s", NODE\_NAME);

}

}

if (0 == strcmp("RTG", ptag)) {

if (0 == strcmp("?", pval)) {

int i;

char \*p;

sprintf(pout, "RTG=");

p = &pout[4];

for (i=0; i<MAX\_RTG\_ENTRIES; i++) {

rtgEntry\_t \*prtg;

prtg = &rtgTable[i];

if (prtg->status == RT\_ACTIVE) {

char mac[Z\_EXTADDR\_LEN];

if (AddrMgrExtAddrLookup(prtg->nextHopAddress, mac) == TRUE) {

MT\_ReverseBytes( mac, Z\_EXTADDR\_LEN );

sprintf(p, "%02X%02X", mac[6], mac[7]);

p = p + strlen(p);

}

}

}

if (strlen(pout) == 4) {

sprintf(p, "NULL");

}

ret = strlen(pout);

}

} // RTG

if (0 == strcmp("AL", ptag)) {

if (0 == strcmp("?", pval)) {

char \*p;

sprintf(pout, "AL=");

p = read\_al(pout+3, -1);

if (strlen(p) == 0) {

sprintf(pout+3, "NULL");

}

ret = strlen(pout);

}

}// AL

if (0 == strcmp("NB", ptag)) {

if (0 == strcmp("?", pval)) {

sprintf(pout, "NB=");

read\_nb(pout+strlen(pout), -1);

if (strlen(pout) == 3) {

sprintf(pout+3, "NULL");

}

ret = strlen(pout);

}

}//NB

if (0 == strcmp("AN", ptag)) {

if (0 == strcmp("?", pval)) {

sprintf(pout, "AN=");

read\_al(pout+strlen(pout), -1);

read\_nb(pout+strlen(pout), -1);

if (strlen(pout) == 3) {

sprintf(wbuf+3, "NULL");

}

ret = strlen(pout);

}

} //AN

if (0 == strcmp("TPN", ptag)) {

/\* 参数格式 x/y 表示在y分钟内上报x次数据

\* x = 0 停止上报,

\* 限制每分钟最大上报6次,最少上报1次

\*/

char \*s = strchr(pval, '/');

if (s != NULL) {

int v1, v2;

\*s = 0;

v1 = atoi(pval);

v2 = atoi(s+1);

if (v1 > 0 && v2 > 0) {

\_tm\_delay = v2\*60/v1;

if (\_tm\_delay >= 10 && \_tm\_delay <= 65) {

if (\_tm\_cnt == 0) {

// start timer

osal\_start\_timerEx( sapi\_TaskID, \_\_REPORT\_EVT, (osal\_rand()%\_tm\_delay) \* 1000);

}

\_tm\_cnt = v1;

}

}

}

} //TMAN

return ret;

}

static char\* read\_nb(char \*buf, int len)

{

int i;

char \*p;

buf[0] = 0;

p = buf;

for (i=0; i<MAX\_NEIGHBOR\_ENTRIES; i++) {

neighborEntry\_t \*pnb = &neighborTable[i];

if (pnb->panId == panid

&& memcmp(pnb->neighborExtAddr,"\x00\x00\x00\x00\x00\x00\x00\x00", 8)!=0

&& pnb->age <= NWK\_ROUTE\_AGE\_LIMIT) {

sprintf(p, "%02X%02X", pnb->neighborExtAddr[1], pnb->neighborExtAddr[0]);

p = p + strlen(p);

}

}

return buf;

}

static char\* read\_al(char \*buf, int len)

{

uint8 assocCnt = 0, i;

uint16 \*assocList;

char mac[8];

char \*p = buf;

p[0] = 0;

for (i=0; i<NWK\_MAX\_DEVICES; i++) {

associated\_devices\_t \*pa = &AssociatedDevList[i];

if (pa->nodeRelation == CHILD\_FFD\_RX\_IDLE || pa->nodeRelation == CHILD\_FFD) {

if (pa->age > NWK\_ROUTE\_AGE\_LIMIT) {

if (TRUE == AddrMgrExtAddrLookup(pa->shortAddr, mac)) {

AssocRemove(mac);

}

}

}

}

assocList = AssocMakeList( &assocCnt );

for (i=0; i<assocCnt; i++) {

if (TRUE == AddrMgrExtAddrLookup(assocList[i], mac)) {

MT\_ReverseBytes( mac, Z\_EXTADDR\_LEN );

sprintf(p, "%02X%02X", mac[6], mac[7]);

p = p + strlen(p);

}

}

return buf;

}

static void my\_report\_proc(void)

{

sprintf(wbuf, "{PN=");

//read\_al(wbuf+strlen(wbuf), -1);

read\_nb(wbuf+strlen(wbuf), -1);

if (strlen(wbuf) == 4) {

sprintf(wbuf+4, "NULL");

}

sprintf(wbuf+strlen(wbuf), ",TYPE=%d%d%s}", NODE\_CATEGORY, logicalType, NODE\_NAME);

zb\_ReceiveDataIndication(0/\*source\*/, 0/\*cmd\*/, strlen(wbuf), wbuf);

}

1. 温湿度传感器代码：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 变量A0表示温度，变量A1表示湿度，V0表示主动上报时间间隔

\* D0(Bit0/Bit1)分别表示温度/湿度的主动上报使能

\* 默认值：A0=0.0,A1=0.0,V0=30,D0=3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include "sapi.h"

#include "osal\_nv.h"

#include "addrmgr.h"

#include "mt.h"

#include "hal\_led.h"

#include "hal\_adc.h"

#include "hal\_uart.h"

#include "OnBoard.h"

#include "humi\_temp\_sensor.h"

#include "UserApp.h"

static uint8 D0 = 3; // 默认打开主动上报功能

static float A0 = 0.0; // A0存储温度值

static float A1 = 0.0; // A1存储湿度值

static uint16 V0 = 1; // V0设置为上报时间间隔，默认为30s

static uint16 myReportInterval = 1; // 上报时间间隔，seconds

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：sensor\_init()

\* 功能：传感器硬件初始化

\* 参数：无

\* 返回：无

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sensor\_init(void)

{

// 初始化传感器代码

hts\_io\_init(); // 初始化传感器代码

// 启动定时器，触发事件：MY\_REPORT\_EVT

osal\_start\_timerEx(sapi\_TaskID, MY\_REPORT\_EVT, (uint16)((osal\_rand()%10) \* 1000));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：updateV0()

\* 功能：更新V0的值

\* 参数：\*val -- 待更新的变量

\* 返回：V0 -- 返回更新后的V0值

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint16 updateV0(char \*val)

{

// 将字符串变量val解析转换为整型变量赋值

myReportInterval = atoi(val);

V0 = myReportInterval;

return V0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：updateA0()

\* 功能：更新A0的值

\* 参数：无

\* 返回：A0 -- 返回更新后的A0值

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float updateA0(void)

{

A0 = get\_hts\_temp(); // 获取温度值

return A0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：updateA1()

\* 功能：更新A1的值

\* 参数：无

\* 返回：A1 -- 返回更新后的A1值

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float updateA1(void)

{

A1 = get\_hts\_humi(); // 获取湿度值

return A1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：sensor\_update()

\* 功能：处理主动上报的数据

\* 参数：无

\* 返回：无

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sensor\_update(void)

{

uint16 cmd = 0;

uint8 pData[128];

uint8 \*p = pData + 1;

int len;

// 根据D0的位状态判定需要主动上报的数值

if ((D0 & 0x01) == 0x01){ // 若温度上报允许，则pData的数据包中添加温度数据

updateA0();

len = sprintf((char\*)p, "A0=%.2f", A0);

p += len;

\*p++ = ',';

}

if ((D0 & 0x02) == 0x02){ // 若湿度上报允许，则pData的数据包中添加湿度数据

updateA1();

len = sprintf((char\*)p, "A1=%.2f", A1);

p += len;

\*p++ = ',';

}

// 将需要上传的数据进行打包操作，并通过zb\_SendDataRequest()发送到协调器

if (p - pData > 1) {

pData[0] = '{';

p[0] = 0;

p[-1] = '}';

zb\_SendDataRequest( 0, cmd, p-pData, pData, 0, AF\_ACK\_REQUEST, AF\_DEFAULT\_RADIUS );

HalLedSet( HAL\_LED\_1, HAL\_LED\_MODE\_BLINK ); // 通信LED闪烁一次

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：usr\_process\_command\_call()

\* 功能：解析收到的控制命令

\* 参数：\*ptag -- 控制命令名称

\* \*pval -- 控制命令参数

\* \*pout -- 控制响应数据，将数据返回给上级调用，通过zb\_SendDataRequest{}发送给协调器

\* 返回：ret -- pout字符串长度

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int usr\_process\_command\_call(char \*ptag, char \*pval, char \*pout)

{

int val;

int ret = 0;

// 将字符串变量pval解析转换为整型变量赋值

val = atoi(pval);

// 控制命令解析

if (0 == strcmp("CD0", ptag)) {

D0 &= ~val;

}

if (0 == strcmp("OD0", ptag)) {

D0 |= val;

}

if (0 == strcmp("D0", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "D0=%u", D0);

}

}

if (0 == strcmp("A0", ptag)) {

if (0 == strcmp("?", pval)) {

updateA0(); // 更新温度数值

ret = sprintf(pout, "A0=%.2f", A0);

}

}

if (0 == strcmp("A1", ptag)) {

if (0 == strcmp("?", pval)) {

updateA1(); // 更新湿度数值

ret = sprintf(pout, "A1=%.2f", A1);

}

}

if (0 == strcmp("V0", ptag)) {

if (0 == strcmp("?", pval)) {

ret = sprintf(pout, "V0=%u", V0);

}else{

updateV0(pval);

}

}

return ret;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称：MyEventProcess()

\* 功能：自定义事件处理

\* 参数：event -- 事件编号

\* 返回：无

\* 修改：

\* 注释：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MyEventProcess( uint16 event )

{

if (event & MY\_REPORT\_EVT)

{

static uint16 count = 0;

// 每1秒获取一次温湿度值

hts\_update();

// 每1秒上报一次温湿度值

if ((count % myReportInterval == 0) && (count != 0))

{

sensor\_update();

count = 0;

}

count++;

// 启动定时器，触发事件：MY\_REPORT\_EVT

osal\_start\_timerEx(sapi\_TaskID, MY\_REPORT\_EVT, (uint16)(1 \* 1000));

}

}