江 蘇 大 學

JIANGSU UNIVERSITY

“计算机网络”实验报告

学院名称: 计算机科学与通信工程学院

专业班级:

学生姓名: cooding-boy

学生学号:

实验名称: 基于 UDP 方式实现对象间通信

完成时间: 2021.12.4

2021 年 12 月

目录

一、项目名称

基于 UDP 方式实现对象间通信

二、项目介绍

一般一个典型的物联网系统包括感控层（传感器），网络层和应用层组成，而网络层主要用于实现感控对象与应用层的服务对象之间的通信。本次作业就以 TCP/IP 协议栈中传输层协议的应用开发为目标，以 UDP 方式实现一种感控对象与服务对象之间的通信机制，其体系结构如图所示。其中感控对象为一个虚拟路灯对象，在实现过程中用随机数模拟其温度、湿度和环境照度等感知数据，灯作为被控对象，可以通过服务器对其进行打开、关闭控制，且用不同颜色表示其开关状态。每个虚拟路灯都将有一个标识，以示区别。而服务对象可以同时与若干个虚拟路灯对象通信，每个虚拟路灯会定期向服务对象发送其当前状态，服务对象可以对任一个虚拟路灯进行开关控制。

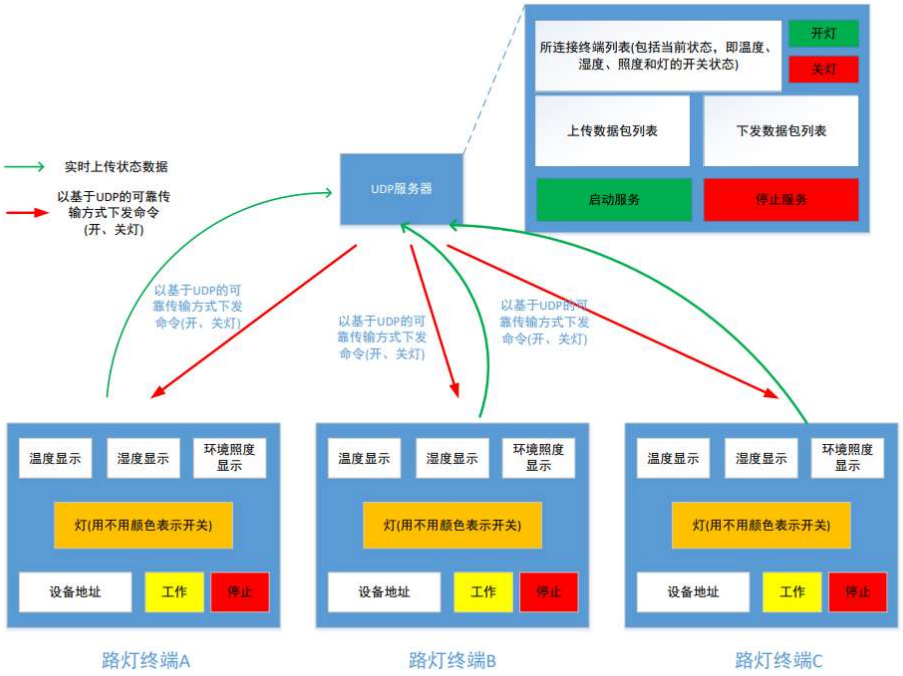

图 1

二、项目目标

- 利用已有的计算机网络相关知识，设计合适的数据包格式
- 掌握 UDP 协议，使用套接字实现基于 UDP 协议实现设备间通信
- 熟练运用所学编程语言，设计美观、友好的图形化界面

三、设计与实现

3.1 报文设计

我在 Java 的环境下创建了一个类来封装协议，这样只需创建一个对象即可实现对数据

包的封装与解析。数据包的具体定义及相关参数含义如下（图 2，表 1）：

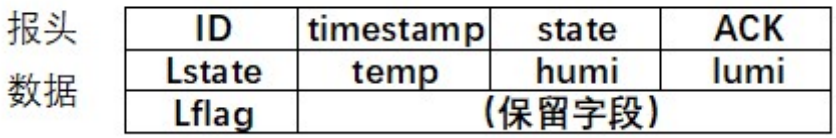

图 2

表 1

报文字段	报文含义
ID	路灯编号
timestamp	时间戳
state	终端状态
ACK	应答标记
Lstate	灯光状态（路灯数据）
temp	温度（路灯数据）
humi	湿度（路灯数据）
lumi	照度（路灯数据）
Lflag	控制信号(服务器数据)

可以看到，路灯和服务器端共享数据包的报头部分，但在数据部分略有差异。此外，我为协议留下了一段保留字段，用于以后使用。

3.2 程序设计

出于报告篇幅限制及突出本课程学习的内容的原因，本小节将主要介绍数据封装、数据传输、数据解析的实现，其余代码可在附录中查看。

3.2.1 数据封装

```
    while(isAlive) {//路灯在线才发送
            SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");
            //设置日期格式
            rt = new RanNum();
            timestamp = (String)df.format(new Date());
            temp = Integer.toString(rt.getRanTemp());     //获取环境温度
            humi = Integer.toString(rt.getRanHumi());     //获取环境湿度
            lumi = Integer.toString(rt.getRanLumi());     //获取环境照度
            message.setTimeStamp(timestamp);
            message.setTemp(temp);
            message.setHumi(humi);
            message.setLumi(lumi);
         try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
        }
    }
```

3.2.2 数据传输

路灯发送数据:

```
        ByteArrayOutputStream bout=new ByteArrayOutputStream();
        ObjectOutputStream oout;       //通过对象流实现
        try {
            oout = new ObjectOutputStream(bout);
            oout.writeObject(message);

            oout.flush();
            byte[] sendBuff=bout.toByteArray();       //转化为字节数组
            DatagramPacket datagram = new DatagramPacket(sendBuff,
sendBuff.length, InetAddress.getByName("localhost"), 2000);
            ds.send(datagram);
            Thread.sleep(1000);        //1s发送一次数据
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();

        }
```

路灯接收数据:

```
        try {
            ds.receive(dp);
           ByteArrayInputStream bint=new ByteArrayInputStream(dp.getData());
            ObjectInputStream oint=new ObjectInputStream(bint);
            message=(Message)oint.readObject();       //反序列化，恢复对象
            //对服务器发来的数据进行操作
            LightControl(message.getLflag());
        } catch (IOException e) {
                e.printStackTrace();
        } catch (ClassNotFoundException e) {
                e.printStackTrace();

        }
```

服务器端的收发代码与其类似，只是在端口号和解析方式上略有区别，具体代码如下:

服务器发送数据:

```
    ByteArrayOutputStream bout=new ByteArrayOutputStream();
    ObjectOutputStream oout;
    try {
        oout = new ObjectOutputStream(bout);
        oout.writeObject(message);
```

```
            oout.flush();
            byte[] sendBuff=bout.toByteArray();       //转化为字节数组
            DatagramPacket datagram = new DatagramPacket(sendBuff,
sendBuff.length, InetAddress.getByName("localhost"), LampPort);
            ds.send(datagram);
        } catch (IOException e) {
            e.printStackTrace();

        }
```

服务器接收数据:

```
    while(true) {   //持续接收数据
        try {
            ds.receive(dp);
           ByteArrayInputStream bint=new ByteArrayInputStream(dp.getData());
                ObjectInputStream oint=new ObjectInputStream(bint);
                message=(Message)oint.readObject();       //反序列化，恢复对象

                DataRefresh(message);       //处理接受到的数据包

            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }

    }
```

3.2.3 数据解析

路灯解析数据:

```
    if(message.getLflag()) {
        System.out.println("灯亮");//测试用，后期删
        view.textD.setBackground(Color.green);
        message.setLstate(true);
        l.setLstate(true);
        flag=true;
    }
    else {
        System.out.println("灯灭");//测试用，后期删
        view.textD.setBackground(Color.red);
        message.setLstate(false);
        l.setLstate(false);
        flag=true;
    }
```

```
    if(flag) {
        message.setACK(1);
        l.setACK(1);
        l.Return(l,message);

    }
```

服务器解析数据:

```
    view.textA.setText(message.getTemp());
    view.textB.setText(message.getHumi());
    view.textC.setText(message.getLumi());
    view.textD.setText(getlstate(message));
    if(message.getTemp()!=null) {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeSta
mp()+":温度: "+message.getTemp()+", 湿度: "+message.getHumi()+", 照度:
"+message.getLumi()+",ACK: "+message.getACK()+"\n");
    }
    else {
    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeSta
mp()+",ACK: "+message.getACK()+"\n");
    }
    view.showArea1.setCaretPosition(view.showArea1.getText().length());
```

3.3 UI 设计

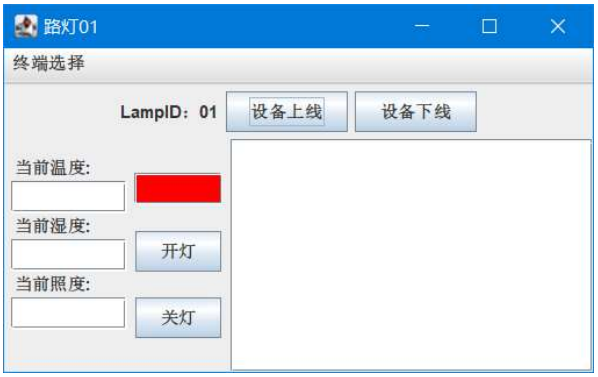

图 3

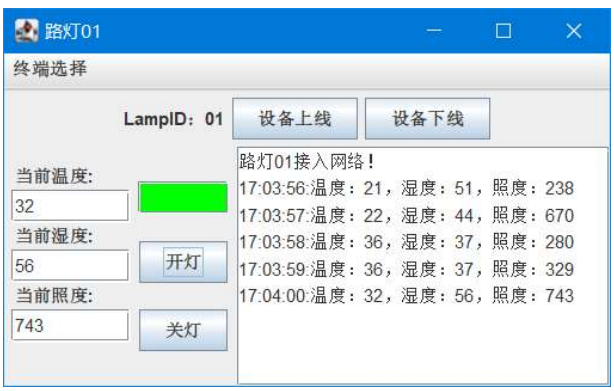

图 4

```
l1 = new Lamp();
handleWindowLamp = new HandleWindowLamp();
textA = new JTextField(5);
textB = new JTextField(5);
textC = new JTextField(5);
textD = new JTextField(0);
showArea1 = new JTextArea();
controlButton1 = new JButton("设备上线");
controlButton2 = new JButton("设备下线");
controlButton3 = new JButton("开灯");
controlButton4 = new JButton("关灯");
p1North = new JPanel();
p1m = new JPanel();

p1North.add(new JLabel("LampID：01"));
p1North.add(controlButton1);
p1North.add(controlButton2);

p1West =Box.createVerticalBox();
p1West.add(new JLabel("当前温度:"));
p1West.add(textA);
p1West.add(new JLabel("当前湿度:"));
p1West.add(textB);
p1West.add(new JLabel("当前照度:"));
p1West.add(textC);

p1East =Box.createVerticalBox();
p1East.add(new JLabel(" "));
p1East.add(textD);
textD.setBackground(Color.red);
p1East.add(new JLabel(" "));
p1East.add(controlButton3);
p1East.add(new JLabel(" "));
p1East.add(controlButton4);

p1m.add(p1West);
p1m.add(p1East);

p1 = new JPanel();
p1.setLayout(new BorderLayout());
p1.add(p1North,BorderLayout.NORTH);
p1.add(new JScrollPane(showArea1),BorderLayout.CENTER);
p1.add(p1m,BorderLayout.WEST);
controlButton1.addActionListener(handleWindowLamp);
controlButton2.addActionListener(handleWindowLamp);
controlButton3.addActionListener(handleWindowLamp);
controlButton4.addActionListener(handleWindowLamp);
```

图 5


服务器
LampID:01 当前温度(℃): 当前湿度(%): 当前照度(lx): 灯光状态:
LampID:02 当前温度(℃): 当前湿度(%): 当前照度(lx): 灯光状态:
LampID:03 当前温度(℃): 当前湿度(%): 当前照度(lx): 灯光状态:
服务器上线
服务器下线
Lamp01:
Lamp02:
Lamp03:
开灯
关灯
开灯
关灯
开灯
关灯

图 6


服务器
LampID:01 当前温度(℃): 21 当前湿度(%): 47 当前照度(lx): 297 灯光状态: on
LampID:02 当前温度(℃): 30 当前湿度(%): 32 当前照度(lx): 588 灯光状态: off
LampID:03 当前温度(℃): 30 当前湿度(%): 57 当前照度(lx): 626 灯光状态: off
服务器上线
服务器下线
ID:02,time:17:05:47:温度：31，湿度：30，照度：440,ACK：0
ID:01,time:17:05:47:温度：20，湿度：32，照度：507,ACK：0
ID:03,time:17:05:48:温度：34，湿度：31，照度：699,ACK：0
ID:02,time:17:05:48:温度：32，湿度：31，照度：576,ACK：0
ID:01,time:17:05:48:温度：35，湿度：39，照度：225,ACK：0
ID:03,time:17:05:49:温度：30，湿度：57，照度：626,ACK：0
ID:02,time:17:05:49:温度：30，湿度：32，照度：588,ACK：0
ID:01,time:17:05:49:温度：21，湿度：47，照度：297,ACK：0
Lamp01:
Lamp02:
Lamp03:
开灯
关灯
开灯
关灯
开灯
关灯

图 7

```
handleWindowServer = new HandleWindowServer();
textA = new JTextField(5);
textB = new JTextField(5);
textC = new JTextField(5);
textD = new JTextField(5);
textE = new JTextField(5);
textF = new JTextField(5);
textG = new JTextField(5);
textH = new JTextField(5);
textI = new JTextField(5);
textJ = new JTextField(5);
textK = new JTextField(5);
textL = new JTextField(5);

showArea1 = new JTextArea();
controlButton1 = new JButton("服务器上线");
controlButton2 = new JButton("服务器下线");
controlButton3 = new JButton("开灯");
controlButton4 = new JButton("关灯");
controlButton5 = new JButton("开灯");
controlButton6 = new JButton("关灯");
controlButton7 = new JButton("开灯");
controlButton8 = new JButton("关灯");

mNorth = new JPanel();
mSouth= new JPanel();
p1m = new JPanel();
textline1 = new JPanel();
textline2 = new JPanel();
textline3 = new JPanel();

textline1.add(new JLabel("LampID:01"));
textline1.add(new JLabel("当前温度(℃):"));      textline1.add(textA);
textline1.add(new JLabel("当前湿度(%):"));      textline1.add(textB);
textline1.add(new JLabel("当前照度(lx):"));     textline1.add(textC);
textline1.add(new JLabel("灯光状态："));        textline1.add(textD);
textline2.add(new JLabel("LampID:02"));
textline2.add(new JLabel("当前温度(℃):"));      textline2.add(textE);
textline2.add(new JLabel("当前湿度(%):"));      textline2.add(textF);
textline2.add(new JLabel("当前照度(lx):"));     textline2.add(textG);
textline2.add(new JLabel("灯光状态："));        textline2.add(textH);
textline3.add(new JLabel("LampID:03"));
textline3.add(new JLabel("当前温度(℃):"));      textline3.add(textI);
textline3.add(new JLabel("当前湿度(%):"));      textline3.add(textJ);
textline3.add(new JLabel("当前照度(lx):"));     textline3.add(textK);
textline3.add(new JLabel("灯光状态："));        textline3.add(textL);
mN1 = Box.createVerticalBox();
mN1.add(textline1);     mN1.add(textline2);     mN1.add(textline3);
mN2 = Box.createVerticalBox();
mN2.add(controlButton1);        mN2.add(controlButton2);
mNorth.add(mN1);
mNorth.add(mN2);

mS1 = Box.createVerticalBox();
mS1.add(new JLabel("Lamp01:"));
mS1.add(controlButton3);        mS1.add(controlButton4);
mS2 = Box.createVerticalBox();
mS2.add(new JLabel("Lamp02:"));
mS2.add(controlButton5);        mS2.add(controlButton6);
mS3 = Box.createVerticalBox();
mS3.add(new JLabel("Lamp03:"));
mS3.add(controlButton7);        mS3.add(controlButton8);
mSouth.add(mS1);        mSouth.add(mS2);        mSouth.add(mS3);

menu = new JPanel();
menu.setLayout(new BorderLayout());
menu.add(mNorth,BorderLayout.NORTH);
menu.add(new JScrollPane(showArea1),BorderLayout.CENTER);
menu.add(mSouth,BorderLayout.SOUTH);
```

图 8

四、测试结果

4.1 测试环境

实验环境为 8GB 内存 AMD R5 5500U，操作系统是 Microsoft windows 10 64bit。本项目通过 Java 实现，运行在 Eclipse IDE 上，版本号为 2021-09。

4.2 测试截图

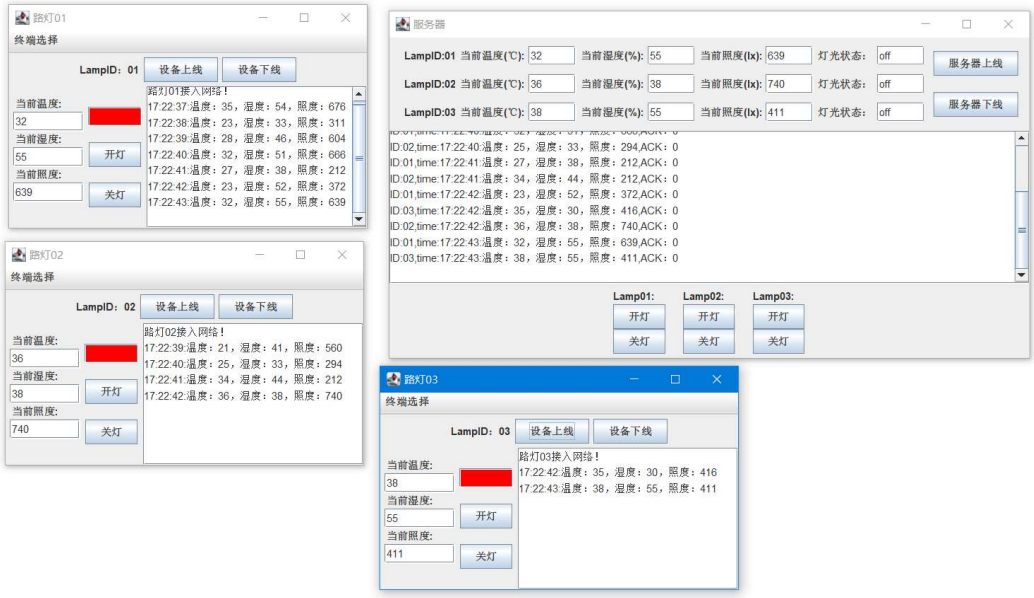

图 9

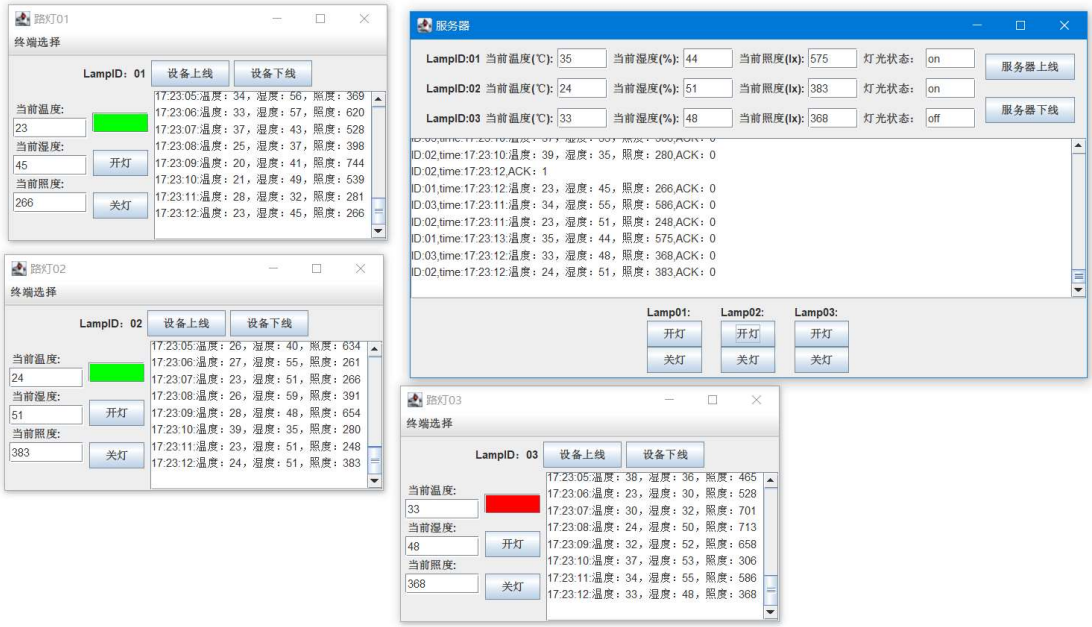

图 10

图 11

4.3 测试视频

项目的完整源代码及测试视频可通过链接，在 GitHub 中获取。

https://github.com/cooding-boy/Smart-streetlamp

五、总结与展望

通过本次实验，我更深刻地认识了计算机网络中基于 UDP 通信的相关概念，并通过 Java 语言对方案做出实现，也取得了可观的实验结果。在这次作业中，我通过自学与向他人请教相结合的方式，掌握了网络编程、序列化、多线程、图形化界面等相关技术，工程能力大大提高。

当然，完成作业的过程中也遇到了一些问题，如：出于课程安排的原因，完成本作业的时候 Java 课程尚未介绍到相关内容。这使得前期准备时消耗了我的大量时间。同时，因为 Java 是本学期新学习的语言，我对面向对象的理解尚浅，在写代码的时候也常遇到代码执行不合预期的情况。好在经过不懈努力，这些困难均已克服。

由于时间紧任务重，加之自学的多线程可能在理解上存在偏差，目前代码中仍存在一些无效的冗余的代码。同时，在路灯执行下线操作时还有一些 bug。对于这些问题我将在后续逐渐完善。

最后，感谢 Java 课程陈老师、研 2004 裴学长在面向对象和多线程方面的指导，也感谢软件 2001 马同学、软件 2003 沈同学在使用 Java 制作图形化界面方面的点拨。他们对我完成本次作业给予了莫大的帮助，我对此深表感激。

附录

源文件：ExampleS.java

```
package myserver;
import java.net.SocketException;
public class ExampleS {
        public static void main(String args[]) throws SocketException{
          WindowServer win = new WindowServer();
```

```
                win.setTitle("服务器");
                win.setBounds(600,100,740,400);
            }
        }
```

源文件：HandleWindowServer.java

```
package myserver;

import java.awt.event.*;
import java.net.SocketException;
import javax.swing.*;
public class HandleWindowServer implements ActionListener {//控制器
    WindowServer view;
    Message message=null;
    Server server = new Server();
    JTextField textD= new JTextField(5);
    public void setView(WindowServer view) {
        this.view = view;
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == view.controlButton1) {
            try {
                server.Online();
                view.showArea1.append("服务器接入网络！\n");
                server.setIsAlive(true);
                server.setView(view);
                server.Receive();
            } catch (SocketException e1) {
                e1.printStackTrace();
            }
        }
        if(e.getSource() == view.controlButton2) {
            server.Offline();
            server.setIsAlive(false);

            view.textA.setText(" ");
            view.textB.setText(" ");
            view.textC.setText(" ");
            view.textD.setText(" ");
            view.textE.setText(" ");
            view.textF.setText(" ");
            view.textG.setText(" ");
            view.textH.setText(" ");
            view.textI.setText(" ");
            view.textJ.setText(" ");
```

```
                view.textK.setText(" ");
                view.textL.setText(" ");
                view.showArea1.append("服务器已离线……\n");
            }
            if(e.getSource() == view.controlButton3) {
                message = new Message();
                message.setLflag(true);
                server.Send(message,2001);
            }
            if(e.getSource() == view.controlButton4) {
                message.setLflag(false);
                server.Send(message,2001);
            }
            if(e.getSource() == view.controlButton5) {
                message = new Message();
                message.setLflag(true);
                server.Send(message,2002);
            }
            if(e.getSource() == view.controlButton6) {
                message.setLflag(false);
                server.Send(message,2002);
            }
            if(e.getSource() == view.controlButton7) {
                message = new Message();
                message.setLflag(true);
                server.Send(message,2003);
            }
            if(e.getSource() == view.controlButton8) {
                message.setLflag(false);
                server.Send(message,2003);
            }
    }
}
```

源文件：Message.java

```
package myserver;
import java.io.Serializable;
public class Message implements Serializable{
    private   String ID;            //路灯 ID
    private String timestamp;   //时间戳
    private Boolean Lflag;      //控制信号
    private Boolean Lstate;     //灯光状态
    private Boolean state;      //终端状态
    private String temp;        //温度
    private String humi;        //湿度
```

```
private String lumi;            //照度
private int ACK = 0;            //
public Message(String ID, String temp) {
    this.ID = ID;
    this.temp = temp;
}
public Message() {
    // TODO Auto-generated constructor stub
}
public String getID() {
    return ID;
}
public void setID(String ID) {
    this.ID = ID;
}
public String getTemp() {
    return temp;
}
public void setTemp(String temp) {
    this.temp = temp;
}
public String getHumi() {
    return humi;
}
public void setHumi(String humi) {
    this.humi = humi;
}
public String getLumi() {
    return lumi;
}
public void setLumi(String lumi) {
    this.lumi = lumi;
}
public String getTimeStamp() {
    return timestamp;
}
public void setTimeStamp(String timestack) {
    this.timestamp = timestack;
}
public Boolean getLflag() {
    return Lflag;
}
public void setLflag(Boolean lflag) {
    Lflag = lflag;
```

```
	}
	public Boolean getState() {
		return state;
	}
	public void setState(Boolean state) {
		this.state = state;
	}
	public Boolean getLstate() {
		return Lstate;
	}
	public void setLstate(Boolean lstate) {
		Lstate = lstate;
	}
	public int getACK() {
		return ACK;
	}
	public void setACK(int aCK) {
		ACK = aCK;
	}
}
```

源文件：Server.java

```
package myserver;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
public class Server {
	int HostPort;
	DatagramSocket ds = null;
	DatagramPacket dp = null;
	WindowServer view;
	Message message=null;
	ServerSend ss = null;
	ServerReceive sr= null;
	Boolean isAlive = true;

	public void setIsAlive(Boolean isAlive) {
		this.isAlive = isAlive;
	}
	public Server() {
		// TODO Auto-generated constructor stub
	}

	public void setView(WindowServer view) {
		this.view = view;
```

```
    }
    public void setMessage(Message message) {
        this.message = message;
    }
    public void Online() throws SocketException {
        HostPort = 2000;
        ds = new DatagramSocket(HostPort);
        byte[] bytes = new byte[512];
        dp = new DatagramPacket(bytes, bytes.length);
        System.out.println("服务器接入网络！");
    }
    public void Offline() {
        ds = null;
        dp = null;
        HostPort = 0;
        sr.setIsAlive(false);
    }
    public void Send(Message message, int LampPort) {
        ss = new ServerSend(ds, message, LampPort);
        ss.start();
    }
    public void Receive() {
        sr = new ServerReceive(dp,ds);
        sr.setView(view);
        sr.start();
    }
}
```

源文件：ServerReceive.java

```
package myserver;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class ServerReceive extends Thread{
    DatagramSocket ds = null;
    DatagramPacket dp = null;
    Message message = new Message();
    Boolean isAlive = true;
    WindowServer view;
    public void setView(WindowServer view) {
        this.view = view;
    }
    public void setIsAlive(Boolean isAlive) {
```

```
        this.isAlive = isAlive;
    }
    public ServerReceive(DatagramPacket dp, DatagramSocket ds) {
        this.dp = dp;
        this.ds = ds;
    }
    @Override
    public void run() {
        while(true) {   //持续接收数据
            try {
                ds.receive(dp);
                ByteArrayInputStream bint=new ByteArrayInputStream(dp.getData());
                ObjectInputStream oint=new ObjectInputStream(bint);
                message=(Message)oint.readObject();          //反序列化，恢复对象

                System.out.println("ID:"+message.getID()+"     ,     温     度     :
"+message.getTemp()+"   ,   湿   度   :   "+message.getHumi()+"   ,   照   度   :
"+message.getLumi()+message.getACK()+message.getLstate());
                DataRefresh(message);      //处理接受到的数据包
            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
    }
    private String getlstate(Message message) {
        if(message.getLstate()==true)
            return "on";
        else
            return "off";
    }
    private void DataRefresh(Message message) {
        if(message.getID().equals("01")) {
            if(message.getTemp()!=null||message.getACK()==1) {
                if(message.getACK()==1||message.getState()==true) {
                    view.textA.setText(message.getTemp());
                    view.textB.setText(message.getHumi());
                    view.textC.setText(message.getLumi());
                    view.textD.setText(getlstate(message));
                    if(message.getTemp()!=null) {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+":
温   度   :   "+message.getTemp()+"   ,   湿   度   :   "+message.getHumi()+"   ,   照   度   :
```

```
"+message.getLumi()+",ACK:  "+message.getACK()+"\n");
                        }
                        else {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+",A
CK:  "+message.getACK()+"\n");
                        }

                        view.showArea1.setCaretPosition(view.showArea1.getText().length());
        //自动滚动
                    }
                    else {
                        view.textA.setText(" ");
                        view.textB.setText(" ");
                        view.textC.setText(" ");
                        view.textD.setText(" ");
                        view.showArea1.append("路灯"+message.getID()+"已离线……\n");
                    }
                }
            }
            if(message.getID().equals("02")) {
                if(message.getTemp()!=null||message.getACK()==1) {
                    if(message.getACK()==1||message.getState()==true) {
                        view.textE.setText(message.getTemp());
                        view.textF.setText(message.getHumi());
                        view.textG.setText(message.getLumi());
                        view.textH.setText(getlstate(message));
                        if(message.getTemp()!=null) {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+":
温 度 :  "+message.getTemp()+" ,  湿 度 :  "+message.getHumi()+" ,  照 度 :
"+message.getLumi()+",ACK:  "+message.getACK()+"\n");
                        }
                        else {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+",A
CK:  "+message.getACK()+"\n");
                        }
                        view.showArea1.setCaretPosition(view.showArea1.getText().length());
        //自动滚动
                    }
                    else {
                        view.textE.setText(" ");
                        view.textF.setText(" ");
```

```
                    view.textG.setText(" ");
                    view.textH.setText(" ");
                    view.showArea1.append("路灯"+message.getID()+"已离线……\n");
                }
            }
        }
        if(message.getID().equals("03")) {
            if(message.getTemp()!=null||message.getACK()==1) {
                if(message.getACK()==1||message.getState()==true) {
                    view.textI.setText(message.getTemp());
                    view.textJ.setText(message.getHumi());
                    view.textK.setText(message.getLumi());
                    view.textL.setText(getlstate(message));
                    if(message.getTemp()!=null) {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+":
温 度 : "+message.getTemp()+" , 湿 度 : "+message.getHumi()+" , 照 度 :
"+message.getLumi()+",ACK: "+message.getACK()+"\n");
                    }
                    else {

    view.showArea1.append("ID:"+message.getID()+",time:"+message.getTimeStamp()+",A
CK: "+message.getACK()+"\n");
                    }
                    view.showArea1.setCaretPosition(view.showArea1.getText().length());
        //自动滚动
                }
                else {
                    view.textI.setText(" ");
                    view.textJ.setText(" ");
                    view.textK.setText(" ");
                    view.textL.setText(" ");
                    view.showArea1.append("路灯"+message.getID()+"已离线……\n");
                }
            }
        }
    }
}
```

源文件：ServerSend.java

```
package myserver;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
```

```
import java.net.DatagramSocket;
import java.net.InetAddress;
public class ServerSend extends Thread{
	DatagramSocket ds = null;
	Message message;
	int LampPort;
	public ServerSend(DatagramSocket ds,Message message,int LampPort) {
		this.ds = ds;
		this.message = message;
		this.LampPort = LampPort;
	}
	@Override
	public void run() {
		ByteArrayOutputStream bout=new ByteArrayOutputStream();
		ObjectOutputStream oout;
		try {
			oout = new ObjectOutputStream(bout);
			oout.writeObject(message);
			 oout.flush();
			 byte[] sendBuff=bout.toByteArray();		//转化为字节数组
			 DatagramPacket datagram = new DatagramPacket(sendBuff, sendBuff.length,
InetAddress.getByName("localhost"), LampPort);
			 ds.send(datagram);
		} catch (IOException e) {
			e.printStackTrace();
		}
		try {
			Thread.sleep(1000);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}
}
```

源文件：WindowServer.java

```
package myserver;
import java.awt.*;
import javax.swing.*;
public class WindowServer extends JFrame {//视图
	Server server;
	HandleWindowServer handleWindowServer;
	JTextField textA,textB,textC,textD,textE,textF,textG,textH,textI,textJ,textK,textL;
	JTextArea showArea1,showArea2,showArea3;
	JButton
controlButton1,controlButton2,controlButton3,controlButton4,controlButton5,controlButton
```

```
6,controlButton7,controlButton8;
    Container con;
    JPanel menu,mNorth,mCenter,mSouth, p1m,textline1,textline2,textline3;
    Box mN1,mN2,mS1,mS2,mS3;
    CardLayout card = new CardLayout();
    WindowServer() {
        init();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  void init() {
        handleWindowServer = new HandleWindowServer();
        textA = new JTextField(5);
        textB = new JTextField(5);
        textC = new JTextField(5);
        textD = new JTextField(5);
        textE = new JTextField(5);
        textF = new JTextField(5);
        textG = new JTextField(5);
        textH = new JTextField(5);
        textI = new JTextField(5);
        textJ = new JTextField(5);
        textK = new JTextField(5);
        textL = new JTextField(5);

        showArea1 = new JTextArea();
        controlButton1 = new JButton("服务器上线");
        controlButton2 = new JButton("服务器下线");
        controlButton3 = new JButton("开灯");
        controlButton4 = new JButton("关灯");
        controlButton5 = new JButton("开灯");
        controlButton6 = new JButton("关灯");
        controlButton7 = new JButton("开灯");
        controlButton8 = new JButton("关灯");

        mNorth = new JPanel();
        mSouth= new JPanel();
        p1m = new JPanel();
        textline1 = new JPanel();
        textline2 = new JPanel();
        textline3 = new JPanel();

        textline1.add(new JLabel("LampID:01"));
        textline1.add(new JLabel("当前温度(℃):"));
```

```
textline1.add(textA);
textline1.add(new JLabel("当前湿度(%):"));
textline1.add(textB);
textline1.add(new JLabel("当前照度(lx):"));
textline1.add(textC);
textline1.add(new JLabel("灯光状态："));
textline1.add(textD);
textline2.add(new JLabel("LampID:02"));
textline2.add(new JLabel("当前温度(℃):"));
textline2.add(textE);
textline2.add(new JLabel("当前湿度(%):"));
textline2.add(textF);
textline2.add(new JLabel("当前照度(lx):"));
textline2.add(textG);
textline2.add(new JLabel("灯光状态："));
textline2.add(textH);
textline3.add(new JLabel("LampID:03"));
textline3.add(new JLabel("当前温度(℃):"));
textline3.add(textI);
textline3.add(new JLabel("当前湿度(%):"));
textline3.add(textJ);
textline3.add(new JLabel("当前照度(lx):"));
textline3.add(textK);
textline3.add(new JLabel("灯光状态："));
textline3.add(textL);
mN1 = Box.createVerticalBox();
mN1.add(textline1);
mN1.add(textline2);
mN1.add(textline3);
mN2 = Box.createVerticalBox();
mN2.add(controlButton1);
mN2.add((new JLabel(" ")));
mN2.add(controlButton2);
mNorth.add(mN1);
mNorth.add(mN2);

mS1 = Box.createVerticalBox();
mS1.add(new JLabel("Lamp01:"));
mS1.add(controlButton3);
mS1.add(controlButton4);
mS2 = Box.createVerticalBox();
mS2.add(new JLabel("Lamp02:"));
mS2.add(controlButton5);
mS2.add(controlButton6);
```

```
            mS3 = Box.createVerticalBox();
            mS3.add(new JLabel("Lamp03:"));
            mS3.add(controlButton7);
            mS3.add(controlButton8);
            mSouth.add(mS1);
            mSouth.add((new JLabel("      ")));
            mSouth.add(mS2);
            mSouth.add((new JLabel("      ")));
            mSouth.add(mS3);

            menu = new JPanel();
            menu.setLayout(new BorderLayout());
            menu.add(mNorth,BorderLayout.NORTH);
            menu.add(new JScrollPane(showArea1),BorderLayout.CENTER);
            menu.add(mSouth,BorderLayout.SOUTH);
//          监视器
            controlButton1.addActionListener(handleWindowServer);
            controlButton2.addActionListener(handleWindowServer);
            controlButton3.addActionListener(handleWindowServer);
            controlButton4.addActionListener(handleWindowServer);
            controlButton5.addActionListener(handleWindowServer);
            controlButton6.addActionListener(handleWindowServer);
            controlButton7.addActionListener(handleWindowServer);
            controlButton8.addActionListener(handleWindowServer);
            con = this.getContentPane();
            con.setLayout(card);
            con.add("first",menu);
            card.show(con,"first");
            handleWindowServer.setView(this);//将视图对象（自己）转给控制器
    }
}
```

源文件：EchoItem.java

```
package mylamp;

import myserver.Message;

public class EchoItem extends Thread{
      WindowLamp view;
      Message message=null;
      Boolean isAlive;
      public EchoItem(WindowLamp view, Message message) {
            this.view = view;
            this.message = message;
      }
```

```
    public void setIsAlive(Boolean isAlive) {
        this.isAlive = isAlive;
    }
    @Override
    public void run() {
        while(isAlive) {
            if(message.getTemp()!=null) {
                view.textA.setText(message.getTemp());
                view.textB.setText(message.getHumi());
                view.textC.setText(message.getLumi());
                view.showArea1.append(message.getTimeStamp()+":  温  度  :
"+message.getTemp()+", 湿度: "+message.getHumi()+", 照度: "+message.getLumi()+"\n");
                view.showArea1.setCaretPosition(view.showArea1.getText().length());
    //自动滚动
            }

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

源文件：ExampleL.java

```
package mylamp;

import java.net.SocketException;

public class ExampleL {
        public static void main(String args[]) throws SocketException{
          WindowLamp win = new WindowLamp();
            win.setTitle("路灯 01");
            win.setBounds(100,100,420,260);
        }
    }
```

源文件：HandleWindowLamp.java

```
package mylamp;

import java.awt.*;
import java.awt.event.*;
import java.net.SocketException;

import javax.swing.*;
```

```
import myserver.Message;
public class HandleWindowLamp implements ActionListener {//控制器
    WindowLamp view;
    Message message=null;
    EchoItem ei = null;
    JTextField textD= new JTextField(5);
    Lamp l1= new Lamp("01",2001);
  public void setView(WindowLamp view) {
        this.view = view;
  }
  public void actionPerformed(ActionEvent e) {
        if(e.getSource() == view.controlButton1) {
            try {
                l1.Online();
                view.showArea1.append("路灯"+l1.LampID+"接入网络！\n");
                message=l1.GetItem();

                l1.setIsAlive(true);
                message.setState(true);
                l1.setLstate(false);        //设置灯光初状态为 off

                l1.setView(view);   //将控制传给 l1
                l1.Receive(l1);
                l1.Send(l1,message);

                ei = new EchoItem(view, message);
                ei.setIsAlive(true);
                ei.start();
            } catch (SocketException e1) {
                e1.printStackTrace();
            }
        }
        if(e.getSource() == view.controlButton2) {
            message.setState(false);
            l1.Send(l1,message);
            try {
                ei.sleep(1000);
            } catch (InterruptedException e1) {
                e1.printStackTrace();
            }
            l1.Offline();
            ei.setIsAlive(false);
            l1.setIsAlive(false);
```

```
                view.textA.setText(" ");
                view.textB.setText(" ");
                view.textC.setText(" ");
                view.showArea1.append("路灯"+l1.LampID+"已离线……\n");

            }
            if(e.getSource() == view.controlButton3) {
                view.textD.setBackground(Color.green);
                l1.setLstate(true);
                message.setLstate(true);

            }
            if(e.getSource() == view.controlButton4) {
                view.textD.setBackground(Color.red);
                l1.setLstate(false);
                message.setLstate(false);
             }
            else if(e.getSource() == view.item1) {
                view.card.show(view.con,"first"); //显示 con 中代号为"first"的组件
            }
            else if(e.getSource() == view.item2) {
                view.card.show(view.con,"second");
            }
            else if(e.getSource() == view.item3) {
                view.card.show(view.con,"third");
            }
    }
}
```

源文件：Lamp.java

```
package mylamp;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

import myserver.Message;

public class Lamp {
    String LampID;
    int LampPort;
    private int ACK;
    static int temp,humi,lumi;
    DatagramSocket ds = null;
    DatagramPacket dp = null;
```

```
WindowLamp view;
Message message=null;
LampSend ls = null;
LampReceive lr = null;
LampGetItem lg = null;
LampReturn r = null;
Boolean isAlive;
static Boolean Lstate;   //灯光状态

public void setIsAlive(Boolean isAlive) {
	this.isAlive = isAlive;
}
public Boolean getIsAlive() {
	return isAlive;
}
public Lamp() {
	// TODO Auto-generated constructor stub
}
public Lamp(String LampID, int LampPort) {
	// TODO Auto-generated constructor stub
	this.LampID=LampID;
	this.LampPort=LampPort;
}
public Boolean getLstate() {
	return Lstate;
}
public void setLstate(Boolean lstate) {
	Lstate = lstate;
}
public void setView(WindowLamp view) {
	this.view = view;
}
public void setMessage(Message message) {
	this.message = message;
}
public void Online() throws SocketException {
	ds = new DatagramSocket(LampPort);
	byte[] bytes = new byte[512];
	dp = new DatagramPacket(bytes, bytes.length);
	System.out.println("路灯"+LampID+"接入网络！");
}
public void Offline() {
	ds = null;
	dp = null;
```

```
        LampPort = 0;
        lg.setIsAlive(false);
        ls.setIsAlive(false);
        lr.setIsAlive(false);
        isAlive = false;
    }
    public void Send(Lamp l, Message message) {
        message.setID(LampID);
        message.setLstate(Lstate);
        ls = new LampSend(l,message,ds);
        ls.setIsAlive(getIsAlive());
        ls.setACK(lr.GetACK());
        ls.start();

    }
    public void Receive(Lamp l) {
        Message message = new Message();
        lr = new LampReceive(this, dp,ds);
        lr.setView(view);

        lr.setIsAlive(getIsAlive());
        lr.start();
        message.setACK(lr.GetACK());

        message = lr.getMessage();
    }
    public Message GetItem() {
        Message message = new Message();
        lg = new LampGetItem();
        lg.start();
        message = lg.getMessage();

        return message;
    }

    public Message Return(Lamp l, Message message) {
        message.setID(LampID);
        message.setLstate(Lstate);
        r = new LampReturn(this, message,ds);
        r.setIsAlive(getIsAlive());
        r.setACK(lr.GetACK());
        r.start();

        return message;
```

```
    }
    public int getACK() {
        return ACK;
    }
    public void setACK(int aCK) {
        ACK = aCK;
    }

}
```

源文件：LampGetItem.java

```
package mylamp;

import java.text.SimpleDateFormat;
import java.util.Date;

import myserver.Message;

public class LampGetItem extends Thread{

    String temp,humi,lumi,timestamp;
    RanNum rt = null;
    Message message = new Message();
    Boolean isAlive = true;

    public void setIsAlive(Boolean isAlive) {
        this.isAlive = isAlive;
    }
    @Override
    public void run() {
        while(isAlive) {
            SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");//设置日期格式
            rt = new RanNum();
            timestamp = (String)df.format(new Date());
            temp = Integer.toString(rt.getRanTemp());       //获取环境温度
            humi = Integer.toString(rt.getRanHumi());       //获取环境湿度
            lumi = Integer.toString(rt.getRanLumi());       //获取环境照度
            getMessage().setTimeStamp(timestamp);
            getMessage().setTemp(temp);
            getMessage().setHumi(humi);
            getMessage().setLumi(lumi);
          try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
            }
        }
        }
        public Message getMessage() {
            return message;
        }

}
```
源文件：LampReceive.java
```
package mylamp;

import java.awt.Color;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

import myserver.Message;

public class LampReceive extends Thread{
    Lamp l;
    LampReturn r;
    int ACK;
    DatagramSocket ds = null;
    DatagramPacket dp = null;
    Message message = new Message();
    Boolean isAlive = true;
    WindowLamp view;
    public void setView(WindowLamp view) {
        this.view = view;
    }
    public void setIsAlive(Boolean isAlive) {
        this.isAlive = isAlive;
    }
    public LampReceive(Lamp l,DatagramPacket dp, DatagramSocket ds) {
        this.l =l;
        this.dp = dp;
        this.ds = ds;
    }
    public int GetACK() {
        return ACK;
    }
    @Override
```

```
public void run() {
	// TODO Auto-generated method stub
	while(isAlive) {
		ACK=0;
		l.setACK(0);
		try {
			ds.receive(dp);
			ByteArrayInputStream bint=new ByteArrayInputStream(dp.getData());
			ObjectInputStream oint=new ObjectInputStream(bint);
			message=(Message)oint.readObject();		//反序列化，恢复对象

			//对服务器发来的数据进行操作
			LightControl(message.getLflag());

		} catch (IOException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

}
private void LightControl(Boolean isOn) {
	boolean flag =false;
	if(isOn) {
		view.textD.setBackground(Color.green);
		message.setLstate(true);
		l.setLstate(true);

		flag=true;
	}
	else {
		view.textD.setBackground(Color.red);
		message.setLstate(false);
		l.setLstate(false);

		flag=true;
	}
	if(flag) {
		message.setACK(1);
		ACK = 1;
		l.setACK(1);
		l.Return(l,message);
	}
```

```
            else {
                message.setACK(0);
                ACK = 0;
                l.setACK(0);
            }
        }
        public Message getMessage() {
            return message;
        }
}
```

源文件：LampReturn.java

```
package mylamp;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.text.SimpleDateFormat;
import java.util.Date;
import myserver.Message;
public class LampReturn extends Thread{
        Lamp l;
        int ACK;
        int HostPort = 2000;
        private String ID ;
        Message message = null;
        DatagramSocket ds = null;
        Boolean isAlive = true;

        public void setIsAlive(Boolean isAlive) {
            this.isAlive = isAlive;
        }
        public LampReturn(Lamp l,Message message, DatagramSocket ds) {
            this.l =l;
            this.message = message ;
            this.ds = ds;
        }
        @Override
        public void run() {
            l.setLstate(message.getLstate());
            SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");//设置日期格式
            message.setTimeStamp((String)df.format(new Date()));
            ByteArrayOutputStream bout=new ByteArrayOutputStream();
```

```
            ObjectOutputStream oout;
            try {
                  oout = new ObjectOutputStream(bout);
                  oout.writeObject(message);

                   oout.flush();
                   byte[] sendBuff=bout.toByteArray();          //转化为字节数组
                   DatagramPacket datagram = new DatagramPacket(sendBuff, sendBuff.length,
InetAddress.getByName("localhost"), 2000);
                   ds.send(datagram);
                   Thread.sleep(1000);
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }
      public void setACK(int aCK) {
            ACK = aCK;
      }
}
```

源文件：LampSend.java

```
package mylamp;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

import myserver.Message;

public class LampSend extends Thread{
      Lamp l;
      int ACK;
      int HostPort = 2000;
      private String ID ;
      Message message = null;
      DatagramSocket ds = null;
      Boolean isAlive = true;

      public void setIsAlive(Boolean isAlive) {
            this.isAlive = isAlive;
```

```
    }
    public LampSend(Lamp l,Message message, DatagramSocket ds) {
        this.l =l;
        this.message = message ;
        this.ds = ds;
    }
    @Override
    public void run() {
        while(isAlive) {      //isAlive 判断是否在线
            message.setLstate(l.getLstate());
            ByteArrayOutputStream bout=new ByteArrayOutputStream();
            ObjectOutputStream oout;        //通过对象流实现
            try {
                oout = new ObjectOutputStream(bout);
                oout.writeObject(message);

                 oout.flush();
                 byte[] sendBuff=bout.toByteArray();          //转化为字节数组
                 DatagramPacket datagram = new DatagramPacket(sendBuff,
sendBuff.length, InetAddress.getByName("localhost"), 2000);
                 ds.send(datagram);
                 Thread.sleep(1000);                //1s 发送一次数据
            } catch (IOException e) {
                e.printStackTrace();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public void setACK(int aCK) {
        ACK = aCK;
    }
}
```

源文件：RanNum.java

```
package mylamp;

public class RanNum {
    public int getRanTemp() {
        int max=40,min=20;
        int ran = (int) (Math.random()*(max-min)+min);
        return ran;
    }
    public int getRanHumi() {
        int max=60,min=30;
```

```
            int ran = (int) (Math.random()*(max-min)+min);
            return ran;
        }
        public int getRanLumi() {
            int max=750,min=200;
            int ran = (int) (Math.random()*(max-min)+min);
            return ran;
        }
}
```

源文件：WindowLamp.java

```
package mylamp;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class WindowLamp extends JFrame {//视图
    JMenuBar menubar;
    JMenu menu;
    JMenuItem item1,item2,item3;
    Lamp l1;
    HandleWindowLamp handleWindowLamp;
    JTextField textA,textB,textC,textD,textE,textF,textG,textH,textI,textJ;
    JTextArea showArea1,showArea2,showArea3;
    JButton controlButton1,controlButton2,controlButton3,controlButton4;
    Container con;
    JPanel p1,p2,p3,p1North,p2North,p3North, p1m;
    Box p1West,p1East,p2West,p2East,p3West,p3East;
    CardLayout card = new CardLayout();
    //BorderLayout border =new BorderLayout();
    WindowLamp() {
        init();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    void init() {
        l1 = new Lamp();
        handleWindowLamp = new HandleWindowLamp();
        textA = new JTextField(5);
        textB = new JTextField(5);
        textC = new JTextField(5);
        textD = new JTextField(0);
        showArea1 = new JTextArea();
        controlButton1 = new JButton("设备上线");
        controlButton2 = new JButton("设备下线");
        controlButton3 = new JButton("开灯");
```

```
controlButton4 = new JButton("关灯");
p1North = new JPanel();
p1m = new JPanel();

p1North.add(new JLabel("LampID：01"));
p1North.add(controlButton1);
p1North.add(controlButton2);

p1West =Box.createVerticalBox();
p1West.add(new JLabel("当前温度:"));
p1West.add(textA);
p1West.add(new JLabel("当前湿度:"));
p1West.add(textB);
p1West.add(new JLabel("当前照度:"));
p1West.add(textC);

p1East =Box.createVerticalBox();
p1East.add(new JLabel(" "));
p1East.add(textD);
textD.setBackground(Color.red);
p1East.add(new JLabel(" "));
p1East.add(controlButton3);
p1East.add(new JLabel(" "));
p1East.add(controlButton4);

p1m.add(p1West);
p1m.add(p1East);

p1 = new JPanel();
p1.setLayout(new BorderLayout());
p1.add(p1North,BorderLayout.NORTH);
p1.add(new JScrollPane(showArea1),BorderLayout.CENTER);
p1.add(p1m,BorderLayout.WEST);
controlButton1.addActionListener(handleWindowLamp);
controlButton2.addActionListener(handleWindowLamp);
controlButton3.addActionListener(handleWindowLamp);
controlButton4.addActionListener(handleWindowLamp);

menubar = new JMenuBar();
menu = new JMenu("终端选择");
item1 = new JMenuItem("路灯 01");
item1.addActionListener(handleWindowLamp);
item2 = new JMenuItem("路灯 02");
item2.addActionListener(handleWindowLamp);
```

```
            item3 = new JMenuItem("路灯 03");
            item3.addActionListener(handleWindowLamp);

            menu.add(item1);
            menu.addSeparator();
            menu.add(item2);
            menu.addSeparator();
            menu.add(item3);
            menubar.add(menu);
            setJMenuBar(menubar);
            con = this.getContentPane();
            con.setLayout(card);
            con.add("first",p1);
            //con.add("second",p2);
            //con.add("third",p3);

            card.show(con,"first");
            handleWindowLamp.setView(this);//将视图对象（自己）转给控制器
    }
}
```