四川农业大学

本科期末课程报告

（物联网工程2014级）

|  |  |
| --- | --- |
| 题 目： | 基于Zigbee的温度控制系统 |
| 学 院： | 信息工程学院 |
| 专 业： | 物联网工程 |
| 学生姓名： | 杜德平 学号：20140405  高文静 学号：201400401  邢忠洁 学号：20140358  代 淼 学号：20140373 |
| 导 师： | 王曼韬 职称：副教授 |
| 完成日期： | 2018年 06月 25日 |

目录

[摘要： 3](#_Toc486198116)

[关键词： 3](#_Toc486198117)

[Abstract: 3](#_Toc486198118)

[Key words: 4](#_Toc486198119)

[1. 项目介绍 4](#_Toc486198120)

[1.1 项目背景与意义 4](#_Toc486198121)

[1.2 项目系统方案 4](#_Toc486198122)

[2. 项目总体构架 5](#_Toc486198123)

[3. 终端模块 5](#_Toc486198124)

[3.1 功能 5](#_Toc486198125)

[3.2 具体描述 5](#_Toc486198126)

[4. 协调器模块 6](#_Toc486198127)

[4.1 功能 6](#_Toc486198128)

[4.2 具体描述 6](#_Toc486198129)

[5. PC模块 6](#_Toc486198130)

[5.1 功能 6](#_Toc486198131)

[5.2 具体描述 7](#_Toc486198132)

[5.3 模块的实现 7](#_Toc486198133)

[6. 服务器模块 8](#_Toc486198134)

[6.1 功能 8](#_Toc486198135)

[6.2 具体描述 8](#_Toc486198136)

[6.3 模块的实现 8](#_Toc486198137)

[7. 浏览器模块 9](#_Toc486198138)

[7.1 功能 9](#_Toc486198139)

[7.2 具体描述 9](#_Toc486198140)

[8. 数据协议 9](#_Toc486198141)

[8. 总结与展望 9](#_Toc486198141)

[10. 部分代码与结果运行图 10](#_Toc486198143)

[10.1 部分代码 10](#_Toc486198144)

[10.1.1 协调器代码 10](#_Toc486198145)

[10.1.2 终端代码 13](#_Toc486198146)

[10.2 部分运行结果图 18](#_Toc486198147)

[致谢 21](#_Toc486198148)

基于zigbee的温度控制系统

物联网工程专业 杜德平 邢中杰 高文静 代淼

指导教师：王曼韬

# 摘要：

近年来，随着计算机技术、网络技术与无线通信技术的高速发展和广泛应用，无线通讯网络成为一个在国际上备受关注的前沿热点的研究领域。目前各种无线技术百花争艳，竞争非常的激烈，无线生活即将到来。

本设计主要利用zigbee网络监控各种生产车间各个地点的温度值，然后通过zigbee网络传递到监控室，上位机能够实时的显示各个监控点对应的温度，并且可以设置对个不同监控点的报警温度，温度超过报警值时上位机可以视觉报警，在现场监控点可以声音报警，并且可以随意增加监控点，随意移动监控点，传输数据安全稳定可靠，采用电池供电，操作非常方便，避免了繁琐的拉线等操作。最后通过温度采集实验对系统进行了测试，结果表明，系统运行稳，效果良好。

# 关键词：

zigbee 无线网络 温度控制

Temperature control system based on ZigBee

# Abstract:

In recent years, with the computer technology, network technology and wireless communications technology rapid development and extensive application of wireless communication network become a focus in the world of cutting-edge hot area of research. Nowadays, all kinds of wireless technology are competitive, the competition is very fierce, the wireless life is coming.

This design mainly use zigbee network monitoring production workshop temperature of each location, and then to the control room by zigbee network, PC can display real-time each corresponding to the temperature of the point to point, and can be set to a different point of alarm temperature, temperature more than alarm value when the PC can be visually alarm, can point to point voice alarm at the scene, and can increase, point to point move monitory point, stable and reliable data transmission security, used batteries, convenient operation, to avoid the complicated operations such as arrows. Finally, the system is tested by temperature acquisition experiment. The results show that the system is stable and effective.

# Key words:

Zigbee wireless network temperature control

# 1. 项目介绍

## 1.1 项目背景与意义

随着计算机软硬件技术、网络技术和工业综合自动化系统整合水平的不断发展，对监控数据传输的实时性、数据接口的开放性以及数据链接的安全性的要求越来越高，有线控制网络的局限性也越来越突出。将设备数据采集之后再通过无线ZigBee网络进行传输是ZigBee技术在工业现场环境中的一种应用，这种新兴的技术必将给工厂现代化注入新的活力。ZigBee

技术填补了低成本、低功耗和低速率无线通信市场的空白，其使用的便捷性是该技术成功的关键，它适用于短距离小范围的基于无线通信的控制领域，必将在工业自动化等领域得到广泛的应用。

## **1.2 项目系统方案**

本系统采用网状结构，通过传感器节点采集温度，并发送这些数据到中心收集节点进行处理。当采集节点的距离太远是我们必须增加路由节点的传递采集节点要发给中心收集节点，在采集节点与中心收集节点之间合适的位置放置几个路由器将大大提高整个网络的稳定传输，为了简单期间，只有一个中心节点收集这些信息，处理后通过串口送到计算机，这些信息可以在串口调试工具或超级终端上看到。

# 2. 项目总体构架

本设计将温度传感器采集来的数据通过zigbee协议传递到上位机显示，并能够在现场实时显示采集的温度，人机交互采用按键控制，LED灯闪烁，（当温度超过设定值，led等亮，LED等相当于风扇，温度高于设定值，开始工作）系统框图如下图。

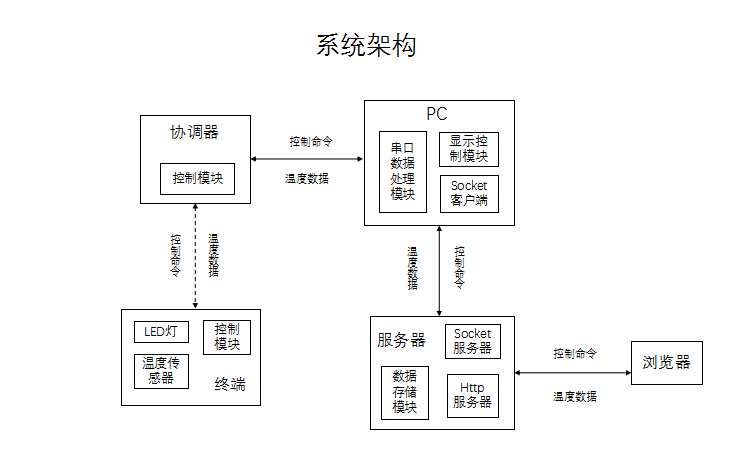


图1

# 3. 终端模块

## 3.1 功能

* 加入网络；
* 采集温度数据;
* 发送温度数据；
* 接收控制命令;
* 控制LED状态；
* 发送节点状态。

## 3.2 具体描述

* 加入网络；
* 5 s采集一次温度数据，通过网络发送到协调器；
* 从网络总接收协调器转发来的控制命令，并控制LED灯状态；

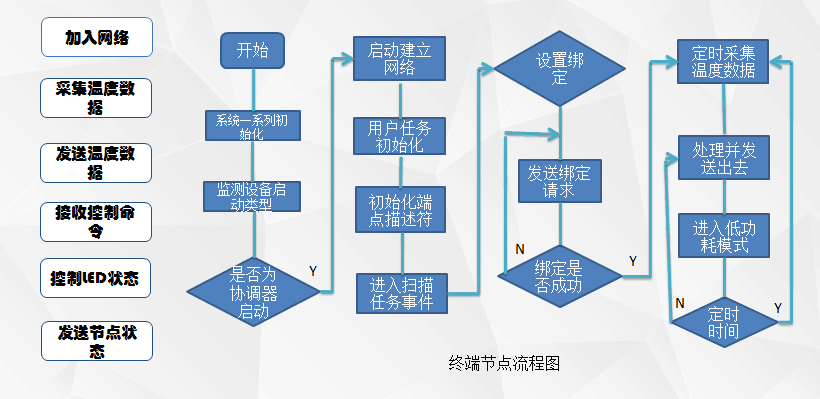


图2

# 4. 协调器模块

## 4.1 功能

* 创建网络；
* 转发温度信息；
* 转发控制命令；
* 转发和发送节点状态。

## 4.2 具体描述

* 创建网络；
* 从网络中接收温度数据，并通过串口发送给PC；
* 从串口接收控制命令，并通过网络发送给终端；

# 5. PC模块

## 5.1 功能

* 转发温度数据；
* 转发控制命令；
* 显示当前温度；
* 发送控制命令。

## 5.2 具体描述

* 连接Socket服务器；
* 从串口接收温度数据，并通过Socket客户端发送到Socket服务器；
* 将从Socket客户端接收到的控制命令通过串口发送到协调器；
* 显示从串口接收到的温度信息；
* 通过串口向协调器发送控制LED灯状态的命令，向服务器发送控制后的LED状态；

## 5.3 模块的实现

有串口数据处理模块、Socket客户端、显示模块、控制模块；集成在一个Winform窗体应用程序中；所有模块共用串口变量currentSerial和Socket变量currentSocket

（1）串口数据处理模块

使用C#的SerialPort类操作串口，发送控制命令直接调用Send函数，读取温度数据通过订阅SerialDataReceivedEventHandler事件使用Read函数读取接收到温度数据之后，通过Socket变量将温度数据发送到Socket服务器；

（2）Socket客户端

使用C#的Socket类实现网络通信，发送数据直接使用Send函数，接收服务器的数据需要开启一个线程循环使用Receive读取控制风扇命令；接收到控制命令之后通过串口变量将命令发送到串口，并改变控制模块的控制按钮状态；

（3）显示模块

串口数据处理模块读取到数据时候，调用本模块提供的接口将温度信息显示在程序首页上；

（4）控制模块

点击相应控制按钮(开风扇/关风扇)之后，通过串口变量将控制风扇的命令发送到串口，通过Socket变量将控制后的风扇状态发送到Socket服务器；

# 6. 服务器模块

## 6.1 功能

* 接收温度信息；
* 存储温度信息；
* 接收各设备状态信息；
* 存储各设备状态；
* 转发控制命令；
* 提供温度数据。

## 6.2 具体描述

* 开启Socket服务器，监听客户端的连接，接收客户端发送的温度数据、各设备的状态信息；
* 将温度数据存入xml文件(数据文件按小时切割并存放在对应的按日期切割的文件夹中)；
* web应用程序响应浏览器请求，读取当前温度和各设备状态并返回给浏览器；
* 接收浏览器的控制命令，发送到Socket客户端，并修改文件中LED状态。

## 6.3 模块的实现

有Http服务器、Socket服务器、温度存储模块，模块之间共用Socket变量currentSocket、温度变量temp和风扇状态变量fan\_state；

（1）Http服务器

使用C#的HttpListener类实现一个简单的Http服务器；使用GetContext函数接收客户端的连接，并创造一个Http上下文，使用http上下文的Request属性获取请求信息，通过http上下文的Response属性设置响应信息，然后通过Response属性的OutputStream属性的Write方法返回响应内容。

返回客户端请求的温度数据和风扇数据；接收开关风扇请求，调用Socket变量将风扇控制命令发送到Socket客户端；并更新变量fan\_state。

（2）Socket服务器

使用Socket类开启一个监听的Socket，监听到客户端的连接之后会关闭当前的Socket连接并创建一个新的通信Socket开启多线程去接收客户端发送的温度数据和风扇状态信息（3）温度存储

以对象的方式访问xml文件，每个5秒钟将不为空的温度temp变量写入xml中；xml文件以小时分割存放在以日志分割的温度记录文件夹中。

# 7. 浏览器模块

## 7.1 功能

* 显示当前温度；
* 显示各设备状态；
* 发送控制命令。

## 7.2 具体描述

* 使用Ajax向web服务器请求当前温度、设备状态；
* 显示当前温度、设备状态；
* 通过Ajax发送LED控制命令。

# 8. 数据协议

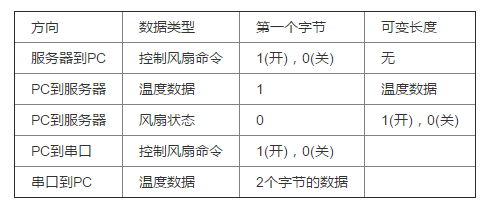


图3

# 9. 总结与展望

通过本次课程设计，大家都学到了很多知识，但是还是有很多遗憾，反向控制灯并没有完成；而且整个过程没有做到节约资源；后期将定时发送数据改为当温度变化超过设定的值时才发送数据，做到节约无线传感网络资源，但是，这样做之后，就要设定每个设备操作多少时间没有发送或转发数据就要发送指定的信息标示自己状态正常；但是这样设计之后整个系统的实现难度将成倍增加；系统还可以增加的模块还很多，也还有很多模块需要我们去完善；在学习中一步一步前行，就可以实现未来。

# 10. 部分代码与结果运行图

## 10.1 部分代码

### 10.1.1 协调器代码

#include <iocc2530.h>

#include <stdio.h>

//#define RX

#define LED1 P1\_0

#define LED2 P1\_1

static char buf[128];

static int len=0;

unsigned char i;

void Delay(unsigned int n)

{

unsigned int i,tt;

for(i=0;i<5;i++)

for(tt=0;tt<n;tt++)

;

}

void rf\_init()

{

//硬件CRC以及AUTO\_ACK使能

FRMCTRL0 |= (0x20 | 0x40);

//设置TX抗混叠过滤器以获得合适的带宽

TXFILTCFG = 0x09;

//调整AGC目标值

AGCCTRL1 = 0x15;

//获得最佳的EVM

FSCAL1 = 0x00;

// RXPKTDONE 中断位使能

RFIRQM0 |= (1<<6);

// RF 中断使能

IEN2 |= (1<<0);

//开中断

EA = 1;

//信道选择，选择11信道

FREQCTRL = 0x0b;

//目标地址过滤期间使用的短地址

SHORT\_ADDR0 = 0x05;

SHORT\_ADDR1 = 0x00;

//目标地址过滤期间使用的PANID

PAN\_ID0 = 0x22;

PAN\_ID1 = 0x00;

//清除RXFIFO缓冲区并复位解调器

RFST = 0xed;

//为RX使能并校准频率合成器

RFST = 0xe3;

//禁止帧过滤

FRMFILT0 &= ~(1<<0);

}

void tx()

{

unsigned char i;

unsigned char tx[13]={'a','b','c','d','e','f','g','h','i','j','k','\r','\n'};

//为RX使能并校准频率合成器

RFST = 0xe3;

// TX\_ACTIVE | SFD

while (FSMSTAT1 & ((1<<1) | (1<<5)));

//禁止RXPKTDONE中断

RFIRQM0 &= ~(1<<6);

//禁止RF中断

IEN2 &= ~(1<<0);

// 清除TXFIFO缓存

RFST = 0xee;

// 清除 TXDONE 中断

RFIRQF1 = ~(1<<1);

// 发送的第一个字节是传输的帧长度

RFD = 13+2;//tx中包含的13个字符+2;填充缓冲区填充过程需要增加2字节，CRC校验自动填充

//将mac的内容写到RFD中

for(i=0;i<13;i++)

{

RFD = tx[i];

}

// 打开RX中断

RFIRQM0 |= (1<<6);

//打开RF中断

IEN2 |= (1<<0);

//校准后使能TX

RFST = 0xe9;

//等待传输结束

while (!(RFIRQF1 & (1<<1)));

//清除 TXDONE状态

RFIRQF1 = ~(1<<1);

//LED1灯状态改变

LED1=~LED1;

//延时

Delay(20000);

}

void initLed(void)

{

//P1为普通 I/O 口

P1SEL &= ~(1<<0);

//P1.0 P1.1设置为输出

P1DIR |= 0x03;

//关闭LED1

LED1=1;

//关闭LED2

LED2=1;

}

### 10.1.2 终端代码

//发送温度数据

#include <ioCC2530.h>

#define uint unsigned int

//定义控制LED灯的端口

#define LED1 P1\_0 //定义LED1为P10口控制

#define LED2 P1\_1 //定义LED2为P11口控制

float avgtemp;

char tempvalue[10];

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

//延时函数

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

void Delay(uint n)

{

uint i,t;

for(i = 0;i<5;i++)

for(t = 0;t<n;t++);

}

//初始化时钟

void initClock(void)

{

/\*相关寄存器可以在 cc2530数据手册中查找\*/

CLKCONCMD &= ~0x40; //设置系统时钟源为 32MHZ晶振

while(CLKCONSTA & 0x40); //等待晶振稳定

CLKCONCMD &= ~0x47; //设置系统主时钟频率为 32MHZ X0XXX000

SLEEPCMD |= 0x04;

}

//初始化Led

void InitLed(void)

{

P1DIR |= 0x03; //P1\_0、P1\_1定义为输出

LED1 = 1; //LED1灯熄灭

LED2 = 1; //LED2灯熄灭

}

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

串口发送字符串函数

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

void UartTX\_Send\_String(char \*Data,int len)

{

int j;

for(j=0;j<len;j++)

{

U0DBUF = \*Data++;

while(UTX0IF == 0) //UTX0IF等于1（发生中断），则一直循环

;

UTX0IF = 0;

}

}

void InitUart()

{

CLKCONCMD &= ~0x40;//设置系统时钟源为32MHZ晶振

while(CLKCONSTA & 0x40 // 打开RX中断

//打开RF中断

//校准后使能TX

//等待传输结束

//清除 TXDONE状态

//LED1灯状态改变

//延时

//P1为普通 I/O 口

//P1.0 P1.1设置为输出

//关闭LED1

//关闭LED2

//发送温度数据

//定义控制LED灯的端口

//定义LED1为P10口控制

//定义LED2为P11口控制

//延时函数

//初始化时钟

/\*相关寄存器可以在 cc2530数据手册中查找\*/

//设置系统时钟源为

32MHZ晶振

//等待晶振稳定

//设置系统主时钟频率为 32MHZ

X0XXX000

//初始化Led

//P1\_0、P1\_1定义为输出

//LED1灯熄灭

//LED2灯熄灭

串口发送字符串函数

//UTX0IF等于1（发生中断），则一直循环

//设置系统时钟源为32MHZ晶振

;

);//等待晶振稳定

CLKCONCMD &= ~0x47;//设置系统主时钟频率为32MHZ

PERCFG = 0x00; //位置1 P0口

P0SEL = 0x3c; //P0用作串口

P2DIR &= ~0XC0;//P0用作串口

U0CSR |= 0x80; //UART方式

U0GCR |= 11; //baud\_e = 11;

U0BAUD |= 216; //波特率设为115200

UTX0IF = 1;

IEN0 |= 0x80;//开总中断

IEN2 |= 0x04;//发送中断使能

IEN0 |= 0x04;//开接收中断使能

URX0IE = 1;

U0CSR |= 0X40; //允许接收

}

//测量片内温度传感器温度

float getTemperature(void)

{

uint value;

ADCCON3 = 0x3e; //选择1.25V为参考电压；14位分辨率；对片内温度传感器采样

ADCCON1 |= 0x30;//ADC启动方式选择为ADCCON1.ST=1事件

ADCCON1 |= 0x40;//ADC启动转换

while(!(ADCCON1 & 0x80))//等待AD转换完成

;

value = ADCL >>4;/\*ADCL寄存器低4位无效，ADC最高只能达到12位有效位。网络上很多代码这里都是右移两位，那是不对的\*/

value |= (((uint)ADCH)<<4);//连接ADCH和ADCL，并赋值给value

return (value - 1367.5)/4.5;

//return ((value) >> 4) - 315; //根据AD值，计算出实际的温度

//return value\*0.06229-311.43;

//return (value-1367.5)/4.5-4; //上面三个方法不知道用那个 高人指点一下

//return value\*0.06229-303.3-4;/\*温度的计算公式为：温度=（（测量电压-某一电压）/温度系数）-温度的误差值\*/

}

void rf\_init()

{

//硬件CRC以及AUTO\_ACK使能

FRMCTRL0 |= (0x20 | 0x40);

//设置TX抗混叠过滤器以获得合适的带宽

TXFILTCFG = 0x09;

//调整AGC目标值

AGCCTRL1 = 0x15;

//获得最佳的EVM

FSCAL1 = 0x00;

// RXPKTDONE 中断位使能

RFIRQM0 |= (1<<6);

// RF 中断使能

IEN2 |= (1<<0);

//开中断

EA = 1;

//信道选择，选择11信道

FREQCTRL = 0x0b;

//目标地址过滤期间使用的短地址

SHORT\_ADDR0 = 0x05;

SHORT\_ADDR1 = 0x00;

//目标地址过滤期间使用的PANID

PAN\_ID0 = 0x22;

PAN\_ID1 = 0x00;

//清除RXFIFO缓冲区并复位解调器

RFST = 0xed;

//为RX使能并校准频率合成器

RFST = 0xe3;

//禁止帧过滤

FRMFILT0 &= ~(1<<0);

}

void tx(char \*tx,int len)

{

unsigned char i;

//为RX使能并校准频率合成器

RFST = 0xe3;

// TX\_ACTIVE | SFD

while (FSMSTAT1 & ((1<<1) | (1<<5)));

//禁止RXPKTDONE中断

RFIRQM0 &= ~(1<<6);

//禁止RF中断

IEN2 &= ~(1<<0);

// 清除TXFIFO缓存

RFST = 0xee;

// 清除 TXDONE 中断

RFIRQF1 = ~(1<<1);

// 发送的第一个字节是传输的帧长度

RFD = len+2;//tx中包含的13个字符+2;填充缓冲区填充过程需要增加2字节，CRC校验自动填充

//将mac的内容写到RFD中

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

{

RFD = tx[i];

}

// 打开RX中断

RFIRQM0 |= (1<<6);

//打开RF中断

IEN2 |= (1<<0);

//校准后使能TX

RFST = 0xe9;

//等待传输结束

while (!(RFIRQF1 & (1<<1)));

//清除 TXDONE状态

RFIRQF1 = ~(1<<1);

//LED1灯状态改变

//延时

Delay(20000);

}

void main(void)

{

initClock();

InitUart();

InitLed();

ATEST = 0x01;//开启温度传感器

TR0 = 0X01; //将温度传感器与ADC连接起来

while(1)

{

avgtemp = getTemperature();//取得温度数

tempvalue[0] = (unsigned char)(avgtemp)/10 + 48; //十位

tempvalue[1] = (unsigned char)(avgtemp)%10 + 48; //个位

UartTX\_Send\_String(tempvalue,2);

tx(tempvalue,2);

Delay(50000); //延时

if((tempvalue[0]\*10+tempvalue[1])>24)

{

LED1=0; //标志发送状态

Delay(50000);

Delay(50000);

}

else

LED1=1;

}

}

## 10.2 部分运行结果图

（1）Socket服务器和Http服务器

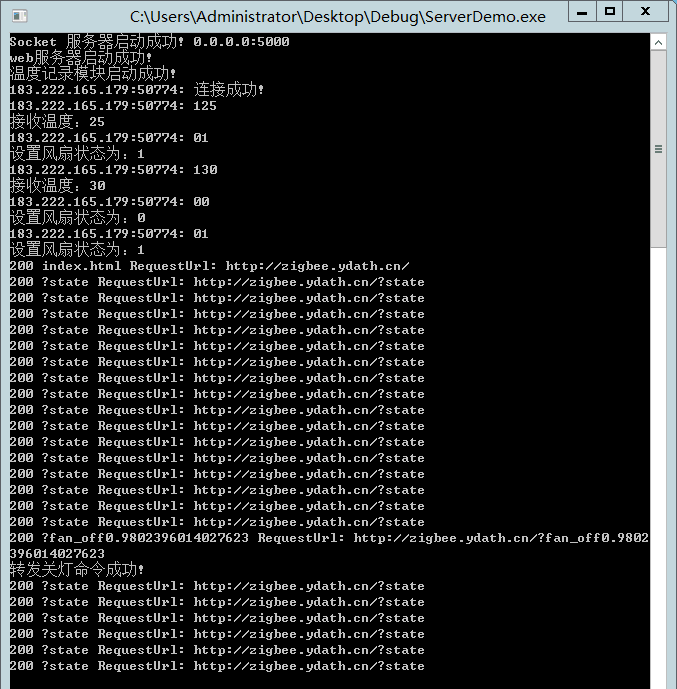


图4

（2）上位机、串口处理模块

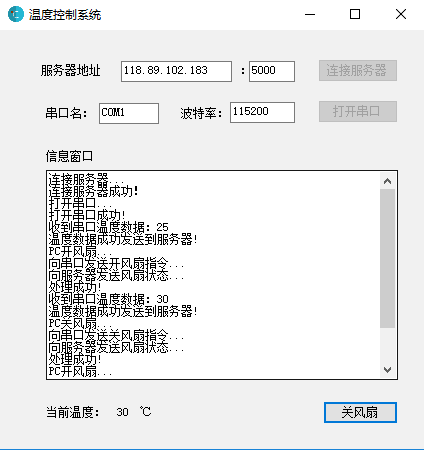


图5

（3）电脑浏览器访问http服务器

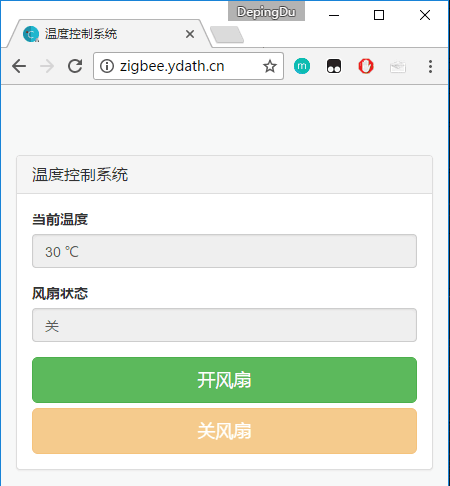


图6

（4）手机浏览器访问http服务器

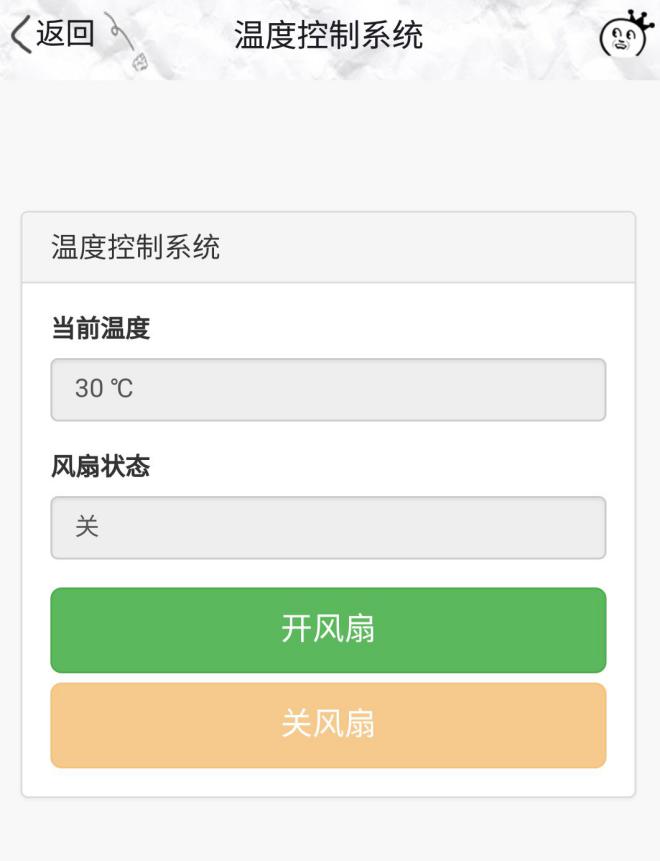


图7

（5）温度存储模块

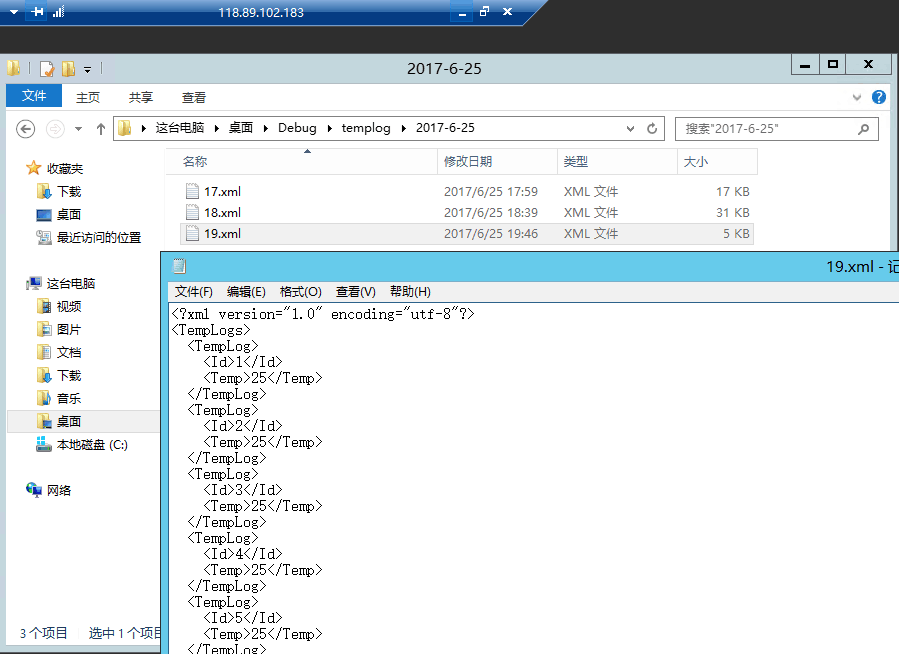


图8

# 致谢

三年时光如白驹过隙，大学余下的时间所剩无几；我们在梧桐的绿荫下走进教室，又踏着桂花的芬芳离开；在这周而复始的日子里，身旁的良师益友陪伴着我们前行；图书馆的安静和机房键盘的吵闹告诉我们：学无止境！

在这里感谢王老师长情的陪伴，这三年您用渊博的知识，丰富的教学经验，兢兢业业的治学精神教给了我们太多太多。这不是您传授给我们的第一门课，也不是最后一门，但是借此机会我们还是要给您说一声感谢。