**1 Arduino**

**1.1 Arduino UNO特性**

 Arduino UNO是基于ATmega328P的单片机开发板。该开发板由14路数字输入/输出引脚（其中6路可以用作PWM输出）、 6路模拟输入、1个16MHz的石英晶体振荡器、一个USB接口、1个电源接头、1个ICSP数据头以及1个复位按钮组成。

UNO包含了单片机运行所需的所有要素，只需用USB连接线将其连接到计算机，或利用AC-DC适配器或电池供电后即可启动。

UNO的特色在于将ATmega16U2编程为一个USB-to-serial转换器，以便能简单、轻松和自由地安装驱动程序。

官方网站：https://www.arduino.cc/

**2 ESP8266**

ESP8266 是由Espressif Systems设计生产的一款高度集成的芯片，该芯片专门针对无线连接的需求而开发，是一个完整且自成系统的 WiFi 网络解决方案。它能够搭载软件应用，也能通过另一个应用处理器卸载所有的 WiFi 网络功能。

ESP8266 具备强大的片上处理和存储功能，这使其可通过 GPIO 口集成传感器及其他应用的特定设备，既缩短前期开发时间，也最大限度减少运行中系统资源的占用。ESP8266 高度片内集成，仅需极少的外部电路，而其包括前端模块在内的整个解决方案，可将设计中 PCB 所占的空间降到最低。

国内的安信可科技等基于该芯片生产了一系列的模块，并提供了基于开源SDK改造的firmware。

**产品规格**

* 802.11 b/g/n
* WiFi Direct (P2P)、soft-AP
* 集成 TCP/IP 协议栈
* 集成 TR 开关、 balun、LNA、 PA 和匹配网络
* 集成PLL、稳压器、DCXO 和电源管理单元
* 802.11b模式下 +19.5dBm 的输出功率
* 小于 10uA 的断电泄露电流
* 集成低功耗 32 位 CPU，可以兼作应用处理器
* SDIO 1.1/2.0、 SPI、UART 接口
* STBC、 1×1 MIMO、2×1 MIMO
* A-MPDU & A-MSDU 聚合 & 0.4ms 的保护间隔
* 2ms 之内唤醒并传递数据包
* 待机状态消耗功率少于 1.0mW (DTIM3)

**3 使用Python测试ESP8266**

程序来源 https://github.com/guyz/pyesp8266

esp8266test.py 是一个通过USB-TTL向ESP8266发送AT指令并接收响应的python程序。

esp8266server.py 这个python程序将ESP8266连接到一个无线AP或无线路由器，然后启动web程序，通过esp8266提供服务。模块工作在multiple connections模式，默认的固件在第二次执行AT+CIPCLOSE=0命令时会失去响应，必须对固件进行刷新。

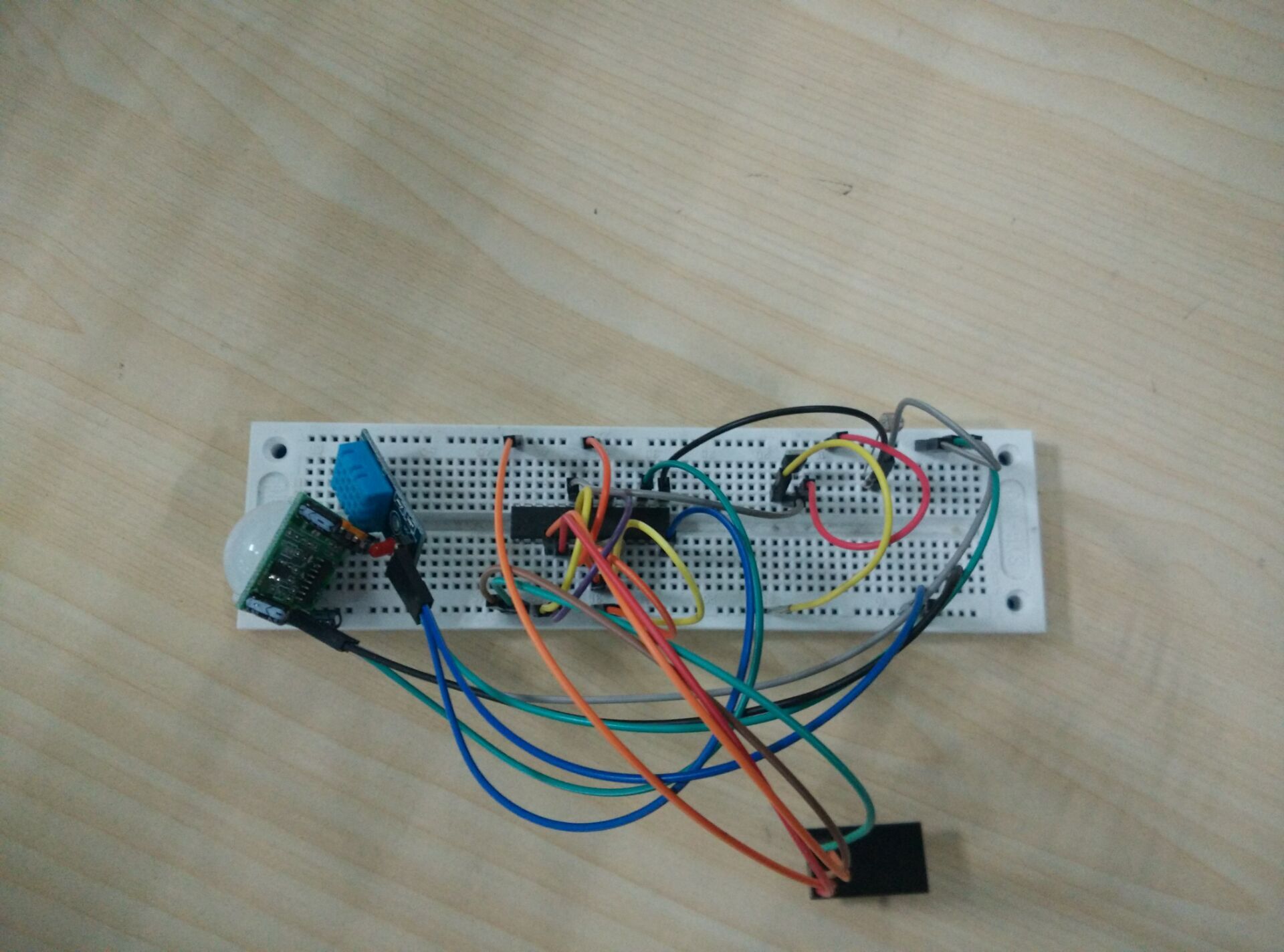
在测试之前，确保波特率和电气连接正确无误。

**4 Arduino连接ESP8266**

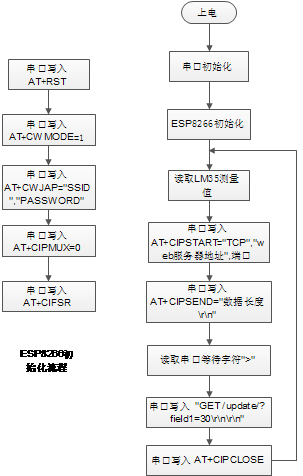
Arduino UNO只有一个内置串口，在使用USB连接线进行编程和串口监视时，串口就不够用了。因此更多时候，会使用SoftwareSerial连接ESP8266模块。在这种情况下，软串口的最高波特率为38400，一般设置为9600。

在连接之前，通过USB转TTL小板连接ESP8266，通过AT指令修改模块默认波特率。

**5 信息采集设备实现**

****

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过REST API将数据发送给PC上的Web服务器。



Web服务器采用Python Web框架Django开发，处理update操作的方法定义在views.py中，代码如下：

def update(request):

a=request.GET["field1"]

currTime=time.strftime("%Y-%m-%d %H:%M:%S",time.localtime(time.time()))

cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")

t=(float(a),currTime)

cx.execute("INSERT INTO temp(tmpr,ts) VALUES (?,?)",t)

cx.commit()

cx.close()

return HttpResponse("done")

在urls.py中，配置如下：

urlpatterns = [

url(r'^admin/', include(admin.site.urls)),

url(r'^view/', 'webtest.views.view', name='view'),

]

Django Web默认监听在127.0.0.1:8000，为了让其他设备能访问web，需要使用命令**python manage.py runserver 0.0.0.0:8000**修改。

Arduino程序如下：

#include <SoftwareSerial.h>

#include <stdlib.h>

#include <PCD8544.h>

#define DEBUG true

#define PIN\_CE 8 //Pin 3 on LCD

#define PIN\_RST 7 //Pin 4 on LCD

#define PIN\_DC 6 //Pin 5 on LCD

#define PIN\_DIN 5 //Pin 6 on LCD

#define PIN\_CLK 4 //Pin 7 on LCD

int ledPin = 13; // LED

int lm35Pin = 0; // LM35 analog input

static const byte LCD\_WIDTH = 84;

static const byte LCD\_HEIGHT = 48;

static PCD8544 lcd(4, 5, 6, 7, 8);

SoftwareSerial ser(2, 3); // RX, TX

void setup() {

pinMode(ledPin, OUTPUT);

Serial.begin(9600); // enable debug serial

ser.begin(9600); // enable software serial

// reset ESP8266

sendData("AT+RST\r\n",5000,DEBUG); // reset module

sendData("AT+CWMODE=1\r\n",2000,DEBUG); // configure as station

sendData("AT+CWJAP=\"SSID\",\"PASSWORD\"\r\n",15000,DEBUG);

sendData("AT+CIPMUX=0\r\n",2000,DEBUG); //single connection mode

res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address

displayIPAddr(res); //display IP Address

}

//把ESP8266的IP地址显示在LCD上

void displayIPAddr(String s) {

int p1 = s.indexOf("+CIFSR:STAIP,\"") + 14;

char ip[16];

int idx = 0;

while (s[p1 + idx] != '"') {

ip[idx] = s[p1 + idx];

idx++;

}

ip[idx] = 0;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(ip);

}

String sendData(String command, const int timeout, boolean debug)

{

String response = "";

ser.print(command); // send the read character to the esp8266

long int time = millis();

while( (time+timeout) > millis())

{

while(ser.available())

{

char c = ser.read(); // read the next character.

response+=c;

}

}

if(debug)

{

Serial.print(response);

lcd.clear();

lcd.setCursor(0, 0);

if (response.indexOf("OK") != -1)

lcd.print(command + ": OK");

else

lcd.print(command + ": Failure");

}

return response;

}

void loop() {

// read the value from LM35.read 10 values for averaging.

int val = 0;

for(int i = 0; i < 10; i++) {

val += analogRead(lm35Pin);

delay(500);

}

// 转换成温度数据

float temp = val\*50.0f/1023.0f;

// convert to string

char buf[16];

String strTemp = dtostrf(temp, 4, 1, buf);

Serial.println(strTemp);

// 和Web服务器建立连接

String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",8000\r\n";

sendData(cmd,5000,DEBUG);

// 准备 GET string

String getStr = "GET /update/?field1="+String(strTemp)+"\r\n\r\n";

//发送数据长度

cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";

ser.print(cmd);

if(ser.find(">")){

//发送数据

ser.print(getStr);

}

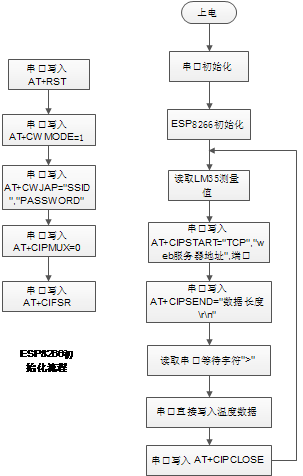
//关闭连接

sendData("AT+CIPCLOSE\r\n",3000,DEBUG);

delay(15000);

}

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过TCP将数据发送给PC上的TCP Server。



Arduino程序的差别主要在于发送数据部分，仅列出不同的部分：

void loop() {

// read the value from LM35.read 10 values for averaging.

int val = 0;

for(int i = 0; i < 10; i++) {

val += analogRead(lm35Pin);

delay(500);

}

// 转换成温度数据

float temp = val\*50.0f/1023.0f;

// convert to string

char buf[16];

String strTemp = dtostrf(temp, 4, 1, buf);

Serial.println(strTemp);

// 和Web服务器建立连接

String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",9999\r\n";

sendData(cmd,5000,DEBUG);

// 准备要发送的数据

String getStr = String(strTemp)+"\r\n\r\n";

//发送数据长度

cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";

ser.print(cmd);

if(ser.find(">")){

//发送数据

ser.print(getStr);

}

//关闭连接

sendData("AT+CIPCLOSE\r\n",3000,DEBUG);

delay(15000);

}

Python语言编写的TCP Server代码如下：

import SocketServer

import datetime

class MyTCPHandler(SocketServer.BaseRequestHandler):

def handle(self):

# self.request is the TCP socket connected to the client

self.data = self.request.recv(1024).strip()

print "ClientIP:", self.client\_address[0]

print "Rx: ", self.data

s = str(datetime.datetime.now())

print "Tx: ", s

self.request.sendall( s )

if \_\_name\_\_ == "\_\_main\_\_":

HOST, PORT = "192.168.199.174", 9999

# Create the server, binding to localhost on port 9999

server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)

print

print "Started: %s Port: %d" % ( HOST, PORT )

print "Server: ", server

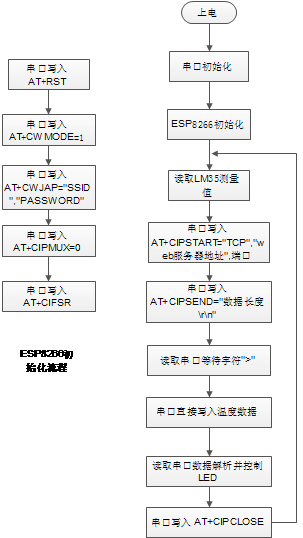
print

# Activate the server; this will keep running until you

# interrupt the program with Ctrl-C

server.serve\_forever()

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过TCP将数据发送给PC上的TCP Server，并解析服务器回传的数据控制LED的亮灭。



Arduino程序的loop部分

void loop() {

// read the value from LM35.read 10 values for averaging.

int val = 0;

for(int i = 0; i < 10; i++) {

val += analogRead(lm35Pin);

delay(500);

}

// 转换成温度数据

float temp = val\*50.0f/1023.0f;

// convert to string

char buf[16];

String strTemp = dtostrf(temp, 4, 1, buf);

Serial.println(strTemp);

// 和Web服务器建立连接

String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",9800\r\n";

sendData(cmd,5000,DEBUG);

// 准备要发送的数据

String getStr = String(strTemp)+"\r\n\r\n";

//发送数据长度

cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";

ser.print(cmd);

if(ser.find(">")){

String s = sendData(getStr, 1000, DEBUG);

if (s.indexOf("#LED:ON#") != -1)

toggleLed(1);

else if (s.indexOf("#LED:OFF#") != -1)

toggleLed(0);

else

Serial.println("bad command");

}

delay(15000);

}

Python程序如下： import SocketServer import sqlite3

class MyTCPHandler(SocketServer.BaseRequestHandler):

def handle(self):

# self.request is the TCP socket connected to the client

self.data = self.request.recv(1024).strip()

print "ClientIP:", self.client\_address[0]

print "Rx: ", self.data

cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")

cu = cx.cursor()

cu.execute("SELECT status FROM led")

row=cu.fetchone()

x=row[0]

cx.close()

s="#LED:OFF#"

if x==0:

s="#LED:OFF#"

print "Tx: ", s

else:

s="#LED:ON#"

print "Tx: ", s

self.request.sendall( s )

if \_\_name\_\_ == "\_\_main\_\_":

HOST, PORT = "192.168.199.174", 9800

# Create the server, binding to localhost on port 9800

server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)

print

print "Started: %s Port: %d" % ( HOST, PORT )

print "Server: ", server

print

# Activate the server; this will keep running until you

# interrupt the program with Ctrl-C

server.serve\_forever()

其中，LED的亮灭由数据库表led中的status决定。在Django应用中添加led控制器和一个led.html模板文件。在views.py中添加：

def led(request):

s=request.GET["led"]

cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")

cx.execute("UPDATE led set status="+s)

cx.commit()

cx.close()

return render(request,'led.html')

修改urls.py为：

urlpatterns = [

url(r'^admin/', include(admin.site.urls)),

url(r'^update/', 'webtest.views.update', name='update'),

url(r'^led/', 'webtest.views.led', name='led')

]

led.html内容如下：

<!DOCTYPE html>

{% load staticfiles %}

<html lang="en">

<head>

<meta charset="UTF-8">

<title></title>

<script src="{% static "js/jquery-1.9.1.min.js" %}"></script>

</head>

<body>

<button id="ledon" class="led">Turn On</button> <!-- button for pin 11 -->

<button id="ledoff" class="led">Turn Off</button> <!-- button for pin 12 -->

<script type="text/javascript">

$(document).ready(function(){

$(".led").click(function(){

var p = $(this).attr('id');

if(p=="ledon")

$.get("/led/?led=1"); // execute get request

else

$.get("/led/?led=0"); // execute get request

});

});

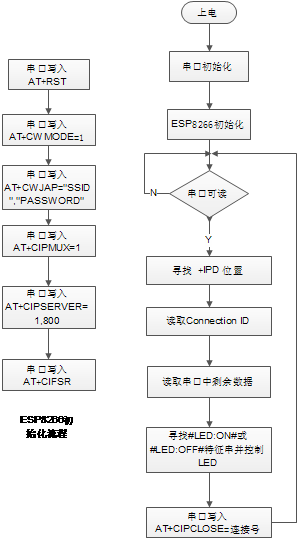
</script>

</body>

</html>

其中，Django中静态文件的存放和位置请参考文后参考资料。

ESP8266工作为workstation模式，Arduino作为服务器，PC机上的客户端程序通过TCP向Arduino发送命令控制LED。



ESP8266工作在Multiple connections模式，这种情况下必须得到连接客户端的id，在发送关闭连接的AT命令时格式为：AT+CIPCLOSE=id

Arduino程序在对ESP8266初始化时修改如下：

sendData("AT+RST\r\n", 5000, DEBUG); // reset module

sendData("AT+CWMODE=1\r\n", 2000, DEBUG); // configure as station

sendData("AT+CWJAP=\"HiWiFi\_Free\",\"openusing\"\r\n", 15000, DEBUG); //connect to AP

sendData("AT+CIPMUX=1\r\n", 1000, DEBUG); // configure for multiple connections

sendData("AT+CIPSERVER=1,800\r\n", 1000, DEBUG); // turn on server on port 800

res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address

displayIPAddr(res);

loop循环的内容如下：

void loop()

{

if (esp8266.available()) // check if the esp is sending a message

{

if (esp8266.find("+IPD,"))

{

delay(1000);

//获取connection id

int connectionId = esp8266.read() - 48;

String response = "";

while (esp8266.available())

{

char c = esp8266.read();

response += c;

}

if (response.indexOf("#LED:ON#") != -1)

digitalWrite(13, HIGH);

else if (response.indexOf("#LED:OFF#") != -1)

digitalWrite(13, LOW);

else

Serial.println("Bad command");

String closeCommand = "AT+CIPCLOSE=";

closeCommand += connectionId;

closeCommand += "\r\n";

sendData(closeCommand, 1000, DEBUG); // close connection

}

}

delay(1000);

}

PC端使用Python编写的TCP客户端程序如下：

import socket

HOST, PORT = "192.168.199.197", 800

# Create a socket (SOCK\_STREAM means a TCP socket)

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

data=raw\_input("command:")

# Connect to server and send data

sock.connect((HOST, PORT))

sock.sendall(data + "\n")

# Receive data from the server and shut down

received = sock.recv(1024)

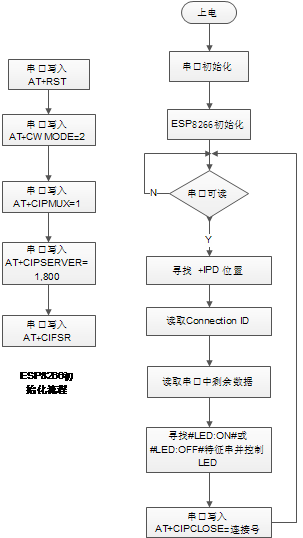
finally:

sock.close()

print "Sent: {}".format(data)

print "Received: {}".format(received)

ESP8266工作为AP模式，Arduino作为服务器，PC机上的客户端程序通过TCP向Arduino发送命令。



Arduino程序在对ESP8266初始化时修改如下：

sendData("AT+RST\r\n", 5000, DEBUG); // reset module

sendData("AT+CWMODE=2\r\n", 2000, DEBUG); // configure as station

sendData("AT+CIPMUX=1\r\n", 1000, DEBUG); // configure for multiple connections

sendData("AT+CIPSERVER=1,800\r\n", 1000, DEBUG); // turn on server on port 800

res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address

displayIPAddr(res);

PC必须连接到ESP8266建立的AP上获取IP地址，ESP8266默认的IP地址是192.168.4.1。将工作方式四的python程序中连接的服务器地址改为192.168.4.1即可进行测试。

当多个设备需要控制时，控制设备需要连接到不到的AP进行控制，这样是非常复杂的。因此AP模式不适合使用。

通过案例，可以了解到：

* 作为客户端工作时，不管是TCP还是Web客户端，均通过AT+CIPSTART指令建立连接，然后通过AT+CIPSEND向服务器发送数据，然后获取服务器返回的响应信息进行后续处理，最后通过AT+CIPCLOSE关闭连接。
* 作为服务器工作时，在特定端口进行监听之后，循环读取串行口数据并分析处理，最后通过AT+CIPCLOSE关闭连接。

**5 红外发射设备实现**

在分析设计阶段，详细分析了空调网络控制系统所需要的实现的功能，并确定了技术路线。

其中，空调控制器部分使用集成AVR芯片的Arduino Mega 2560开发板作为下位机，配合各种传感器等实现感知，使用红外编码（NEC协议）对进行空调进行控制，使用C语言进行编程。空调控制器通过WIFI模块和校园网络与上位机（服务器）进行通信。

服务器系统采用J2EE+Mysql的架构。根据系统功能进行了数据库设计。

图1为系统逻辑架构，其中包含了整个系统的构成部分及之间的逻辑关系。

图2为系统物理部署拓扑图，其中的节能控制器（即空调控制器）通过多个无线路由和服务器进行通信，因此在这种模式下，空调控制器的WIFI模块工作在TCP-Client模式，主动向Web服务器提交状态信息，Web服务器随后返回控制信息给空调控制器。

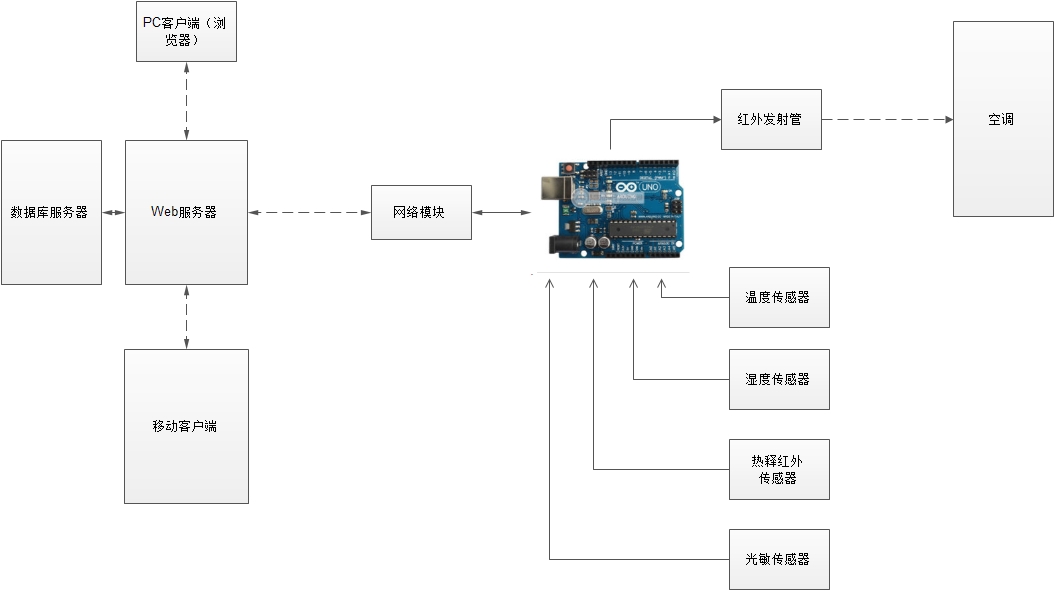


图1系统逻辑架构

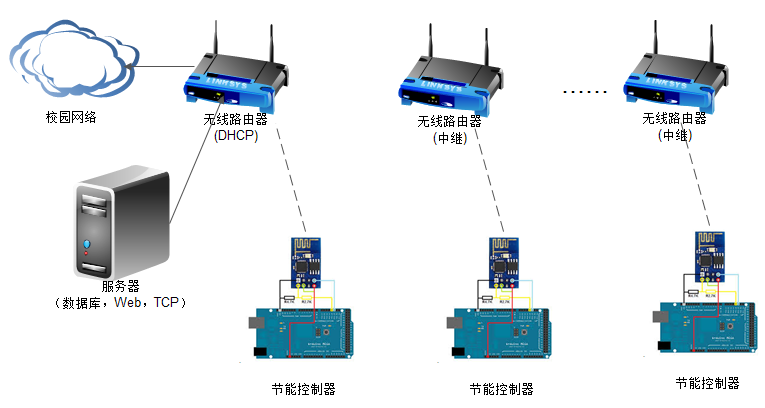


图2 系统物理部署拓扑图

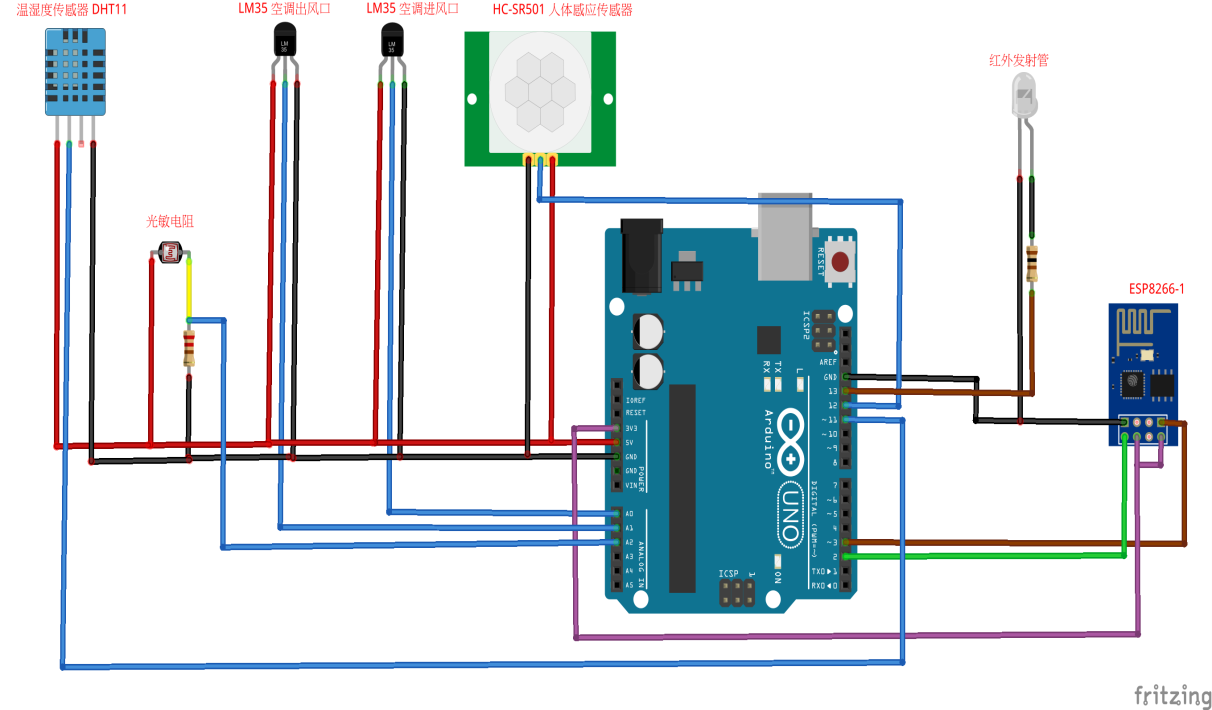


图3 控制器结构

图3为空调控制器的物理结构，由Arduino开发板、温湿度传感器、光敏传感器、热释红外传感器、温度传感器、红外发射管和WIFI模块组成。

**其中，两个温度传感器分别安装在空调出风口和空调侧面，通过采集到的温度数据的差值，可判断空调的开关状态和制冷制热状态。所有的状态信息（温度、湿度、光照度、有无人员等）全部提交到Web服务器进行处理。**

在系统设计中，还有一个重要的问题是通信协议的制定，由于本系统基于HTTP协议来实现，因此各种状态信息均**定时**以Query String的方式提交给Web服务器，同时Web服务器返回简单的控制信号。由于系统可同时控制多台空调，因此**每个空调控制器需要编号，并记录空调的品牌和型号。**

这种架构的优点是，下位机不需要红外编码生成逻辑，这些工作完全由计算能力强大的Web服务器完成，在Web数据库中可存储各个品牌空调的红外编码协议或具体编码，**服务器可根据下位机发送的空调编号来决定返回控制信号的协议和编码内容，因此适应性较强。**

**系统实现**

根据先前的技术储备和系统设计，分阶段实现：

3.1 硬件电路搭建

3.2 下位机C程序编写

3.3 上位机（Web服务器）程序编写

在此过程中，同时形成相关文档。

//添加流程图及必要文字说明



void setup(void)

{

Serial.begin(9600);

Serial.print("setup begin\r\n");

Serial.print("FW Version:");

Serial.println(wifi.getVersion().c\_str());

if (wifi.setOprToStationSoftAP()) {

Serial.print("to station + softap ok\r\n");

} else {

Serial.print("to station + softap err\r\n");

}

if (wifi.joinAP(SSID, PASSWORD)) {

Serial.print("Join AP success\r\n");

Serial.print("IP:");

Serial.println( wifi.getLocalIP().c\_str());

} else {

Serial.print("Join AP failure\r\n");

}

if (wifi.disableMUX()) {

Serial.print("single ok\r\n");

} else {

Serial.print("single err\r\n");

}

Serial.print("setup end\r\n");

}

void loop(void)

{

uint8\_t buffer[128] = {0};

if (wifi.createTCP(HOST\_NAME, HOST\_PORT)) {

Serial.print("create tcp ok\r\n");

//char \*hello = "Hello, this is client!";

// wifi.send((const uint8\_t\*)hello, strlen(hello));

while(true){

String str="";

unsigned int irSignal[140];

uint32\_t len = wifi.recv(buffer, sizeof(buffer), 10000);

if (len > 0) {

Serial.print("Received:[");

for(uint32\_t i = 0; i < len-1; i++) {

if((char)buffer[i]!=','){

Serial.print((char)buffer[i]);

str=str+((char)buffer[i]);

}else{

irSignal[i]=str.toInt();

str="";

}

}

Serial.print("]\r\n");

}

int khz = 38;

irsend.sendRaw(irSignal, sizeof(irSignal), khz);

}

} else {

Serial.print("create tcp err\r\n");

}

delay(5000);

}

硬件设备供电后，通过无线网络设备连入网络并连入socket服务器，向其发送其设备唯一识别码也就是硬件名称。服务器接收其硬件名称向数据库order表中查询有无对应设备的红外编码指令，查询到了即取出发回给硬件，硬件获取到红外指令后通过红外二级管发射，完成空调的调节控制。



**public** **class** M2MServer {

//用户集合

**private** **static** ArrayList<Socket> *list* = **new** ArrayList<Socket>();

**public** **static** **void** main(String[] args) {

//上线用户

Socket s = **null**;

//用户

String ip = **null**;

**try** {

//构建服务器对象

ServerSocket ss = **new** ServerSocket(1234);

//构建 用户集合

*list* = **new** ArrayList<Socket>();

System.*out*.println("服务器准备就绪 ...");

//循环监听

**while**(**true**){

//上线用户

s = ss.accept();

//上线的人都添加到 集合中

*list*.add(s);

//获取 Socket IP

ip = s.getInetAddress().getHostAddress();

System.*err*.println( ip + " 用户上线了 , 当前在线用户为: " + *list*.size() + "人 !" );

//构建 发送信息线程

M2MSend send = **new** M2MSend(s);

send.start();

}

} **catch** (IOException e) {

//用户下线

*list*.remove(s);

System.*err*.println(ip + " 已下线 , 当前在线人数为: " + *list*.size() + " 人 !");}

}

**public** **static** ArrayList<Socket> getList() {

**return** *list*;

}

**public** **static** **void** setList(ArrayList<Socket> list) {

M2MServer.*list* = list;

}

}

红外Socket服务器启动后，等待硬件连入，连入后创建线程，继续监听端口。线程接收硬件发送的设备识别码（设备名称），访问数据库order表查询有无红外指令记录，查到指令后取出发送给硬件设备并删除数据库中此条记录。

**public** **class** M2MSend **extends** Thread {

//用户集合

**private** ArrayList<Socket> list = M2MServer.*getList*();

//当前用户

**private** Socket s;

**public** M2MSend(Socket s){

**this**.s = s;

}

**public** **void** run(){

//获取该用户 IP

String ip = s.getInetAddress().getHostAddress();

**try** {

//读取用户信息

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(s.getInputStream()));

//不断的读取写出数据

**while**(**true**){

//接收数据

String info = **null**;

info=reader.readLine();

//如果读取信息不为空

//if((info=reader.readLine()) != null){

String sql="select comd from order where name ="+info+";";

String sql2="delete from order where name ="+info+";";

**try**{

Statement stmt = **null**;

Connection conn = **null**;

conn=DB.*getConn*();

stmt=DB.*createStmt*(conn);

ResultSet rs = stmt.executeQuery(sql);

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

info=rs.getString("comd");

}

stmt.execute(sql2);

}

**catch**(Exception e){

e.printStackTrace();

}

//获取对象的输出流

PrintWriter pw;

pw = **new** PrintWriter(s.getOutputStream());

//写入信息

pw.println(info);

pw.flush();

System.*out*.println(info);

Thread.*sleep*(5000);

//}

}

} **catch** (IOException | InterruptedException e1) {

//用户下线

list.remove(s);

System.*err*.println(ip + " 已下线 , 当前在线人数为: " + list.size() + " 人 !");

}

}

}



登入web端控制系统后，输入用户名密码登录进行管理。页面会显示当天天气情况，并且提供用户设备状况（是否开启）查询，未打开的设备可以控制打开。。如果环境状态采集器已打开连入网络，那么可以查询设备所处地环境状态。后台分析数据给出建议显示给出空调设定建议。如果空调控制设备打开连入网络，用户可以更改空调设置，发送保存至数据库order表。

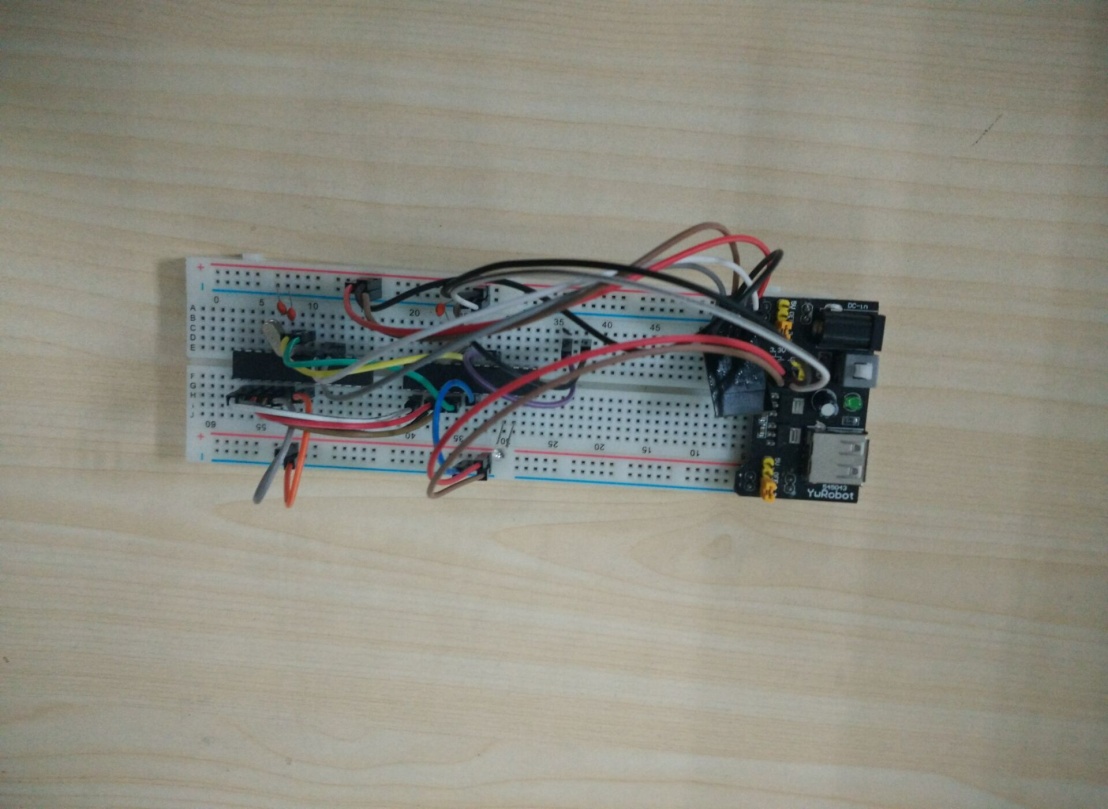


图4 空调控制器

在开发板上对系统进行测试成功之后，将程序烧写到AVR单片机芯片中，然后在面包板上搭建了最小系统，可显著降低成本。AVR最小系统仅需要一个单片机芯片、2个22pf的瓷片电容和1个12MHz的晶体振荡器。

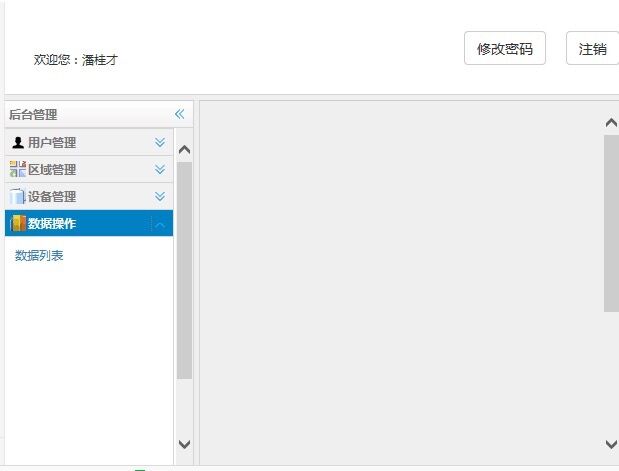


图5-图8 控制系统软件

控制系统软件接收控制器发送的状态信息，并根据预设的条件等生成具体的编码信息发送给下位机，由下位机完成空调的具体控制工作。

**参考文献**

* [Django基础教程](http://www.ziqiangxuetang.com/django/django-tutorial.html)
* [自造你的Arduino-UNO板](http://www.instructables.com/id/%E8%87%AA%E9%80%A0%E4%BD%A0%E7%9A%84Arduino-UNO%E6%9D%BF/)
* [Serial-to-WiFi Tutorial using ESP8266](http://fab.cba.mit.edu/classes/863.14/tutorials/Programming/serialwifi.html)
* [Arduino WiFi Control with ESP8266 Module](http://androidcontrol.blogspot.jp/2015/05/arduino-wifi-control-with-esp8266-module.html)
* [Easy ESP8266 WiFi Debugging with Python](http://www.instructables.com/id/Easy-ESP8266-WiFi-Debugging-with-Python/?ALLSTEPS)
* [ESP8266 Arduino LED Control (Control The Digital Pins Via WiFi, Send Data From Webpage to Arduino)](http://allaboutee.com/2015/01/02/esp8266-arduino-led-control-from-webpage/)
* [ESP8266 Serial WIFI Module](http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module)
* [diy-layout-creator](https://code.google.com/p/diy-layout-creator/)
* [Nokia 5110液晶显示模块的使用](http://yfrobot.com/thread-2412-1-1.html)
* [IPython Notebook: 交互计算新时代](http://mindonmind.github.io/2013/02/08/ipython-notebook-interactive-computing-new-era/)
* <http://ipython.org/index.html>
* [用Python进行SQLite数据库操作](http://www.cnblogs.com/yuxc/archive/2011/08/18/2143606.html)