# 附录

[附录 I](#_Toc422213157)

[附录1系统原理图 1](#_Toc422213158)

[附录2系统PCB和实物 2](#_Toc422213159)

[附录3手机端应用程序界面 3](#_Toc422213160)

[附录4专利证书 4](#_Toc422213161)

[附录5期刊论文 5](#_Toc422213162)

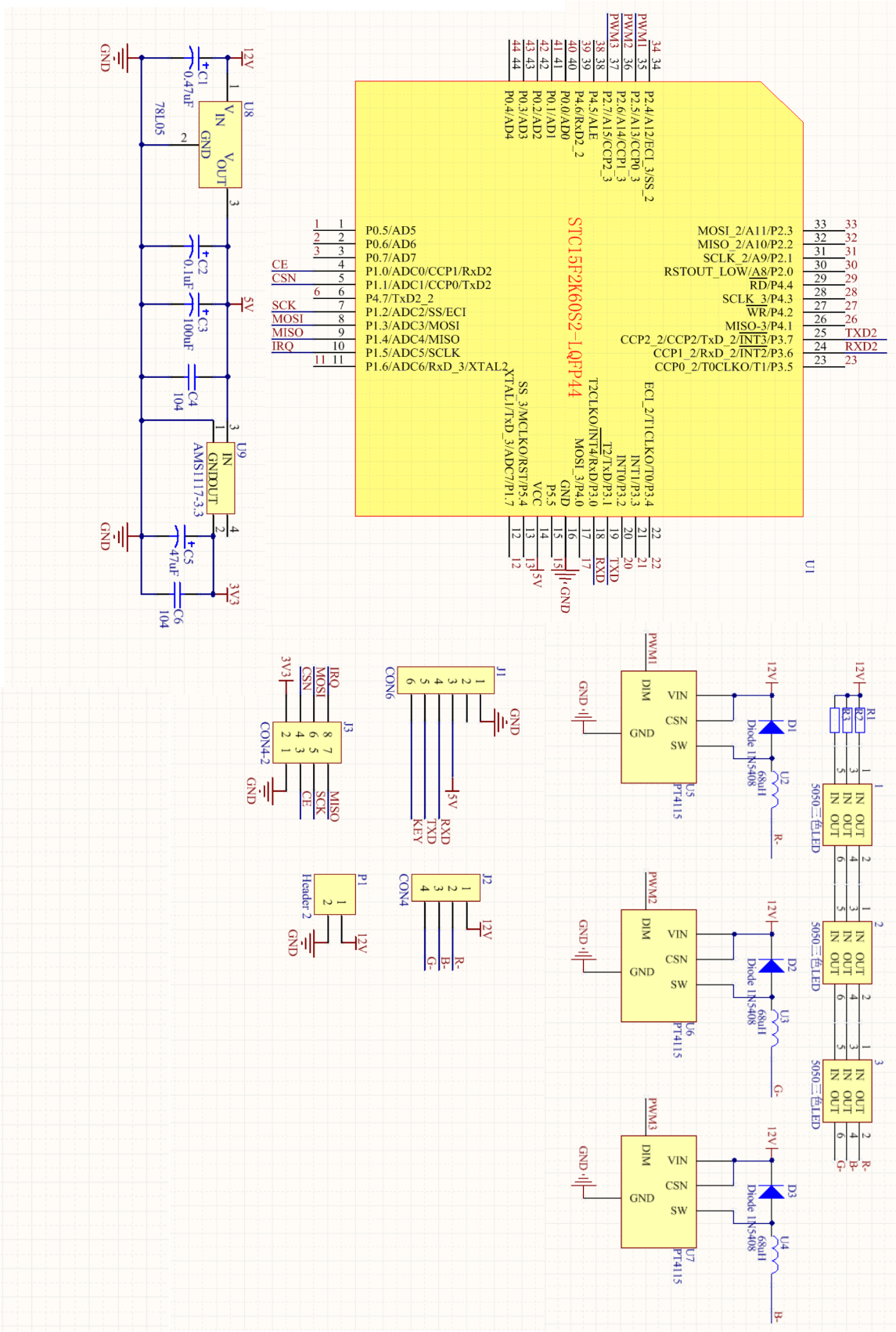
[附录6获奖证书 10](#_Toc422213163)

[附录7系统手机端程序 11](#_Toc422213164)

[附录8系统LED控制器程序 22](#_Toc422213165)

## 附录1系统原理图

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统原理图



## 附录2系统PCB和实物

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统PCB图

|  |  |
| --- | --- |
| 2014-10-08_221621 | 2014-10-08_221359 |

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统实物图

|  |
| --- |
| D:\用户目录\Documents\Tencent Files\1522525981\Image\C2C\BA33B479B394C6621F9A230F46E66926.jpg |

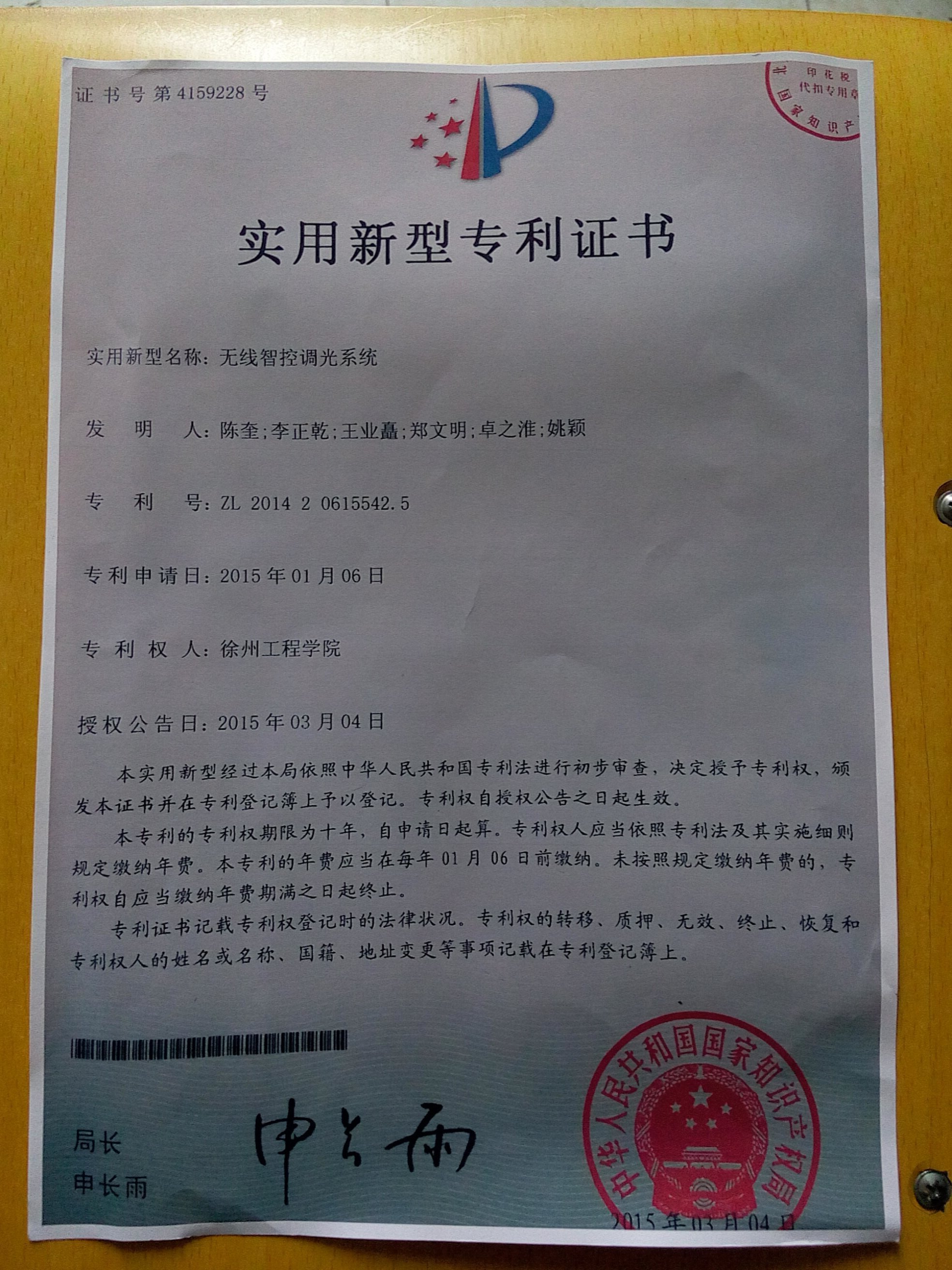
## 附录3手机端应用程序界面

基于手机蓝牙控制的 i-Light智能家居彩灯和控制系统手机应用程序界面

|  |  |
| --- | --- |
| 手机APP主界面 | 灯光状态调节界面 |
| S50520-014524  音乐频谱 | 七彩渐变 |

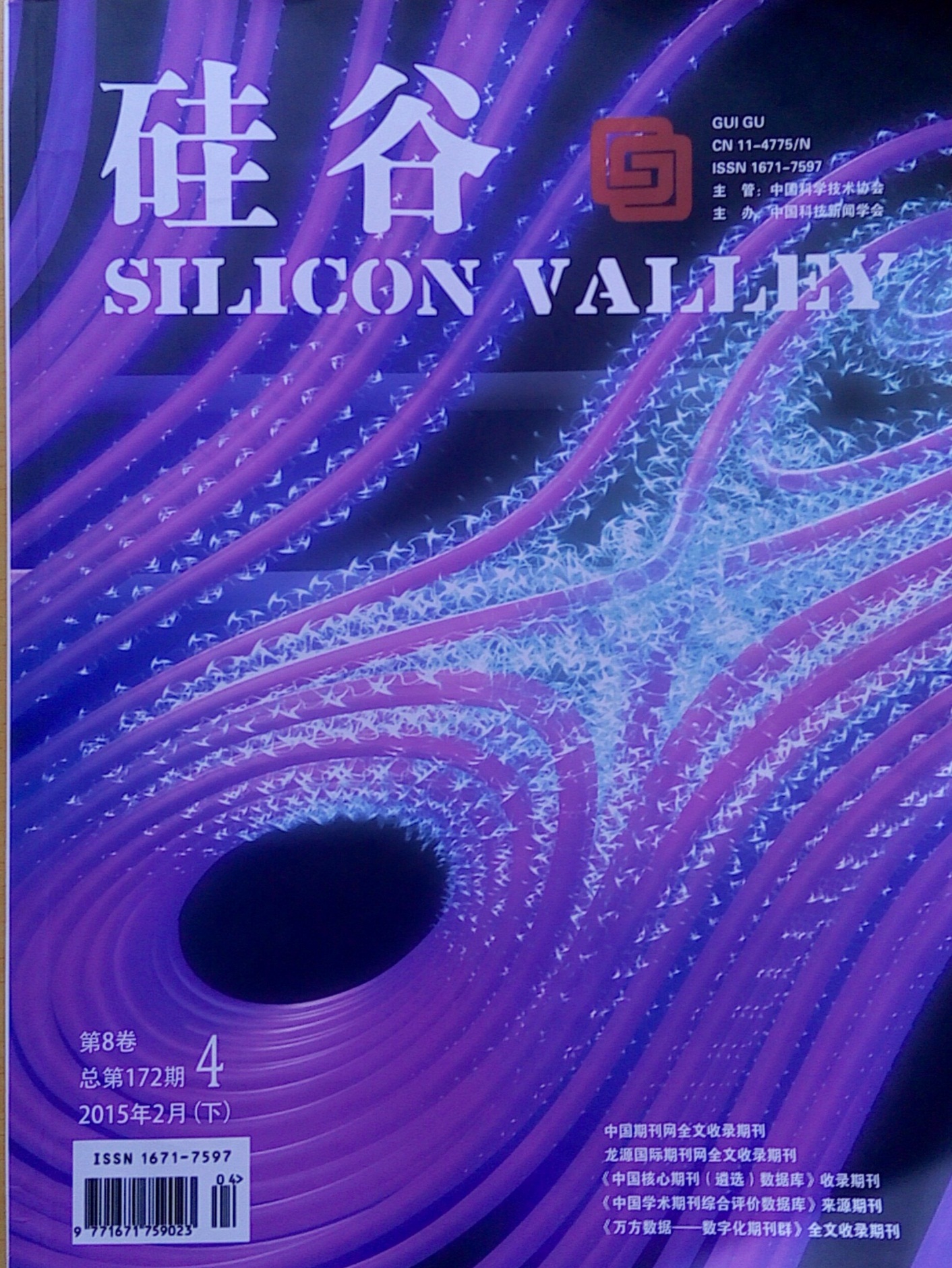
## 附录4专利证书

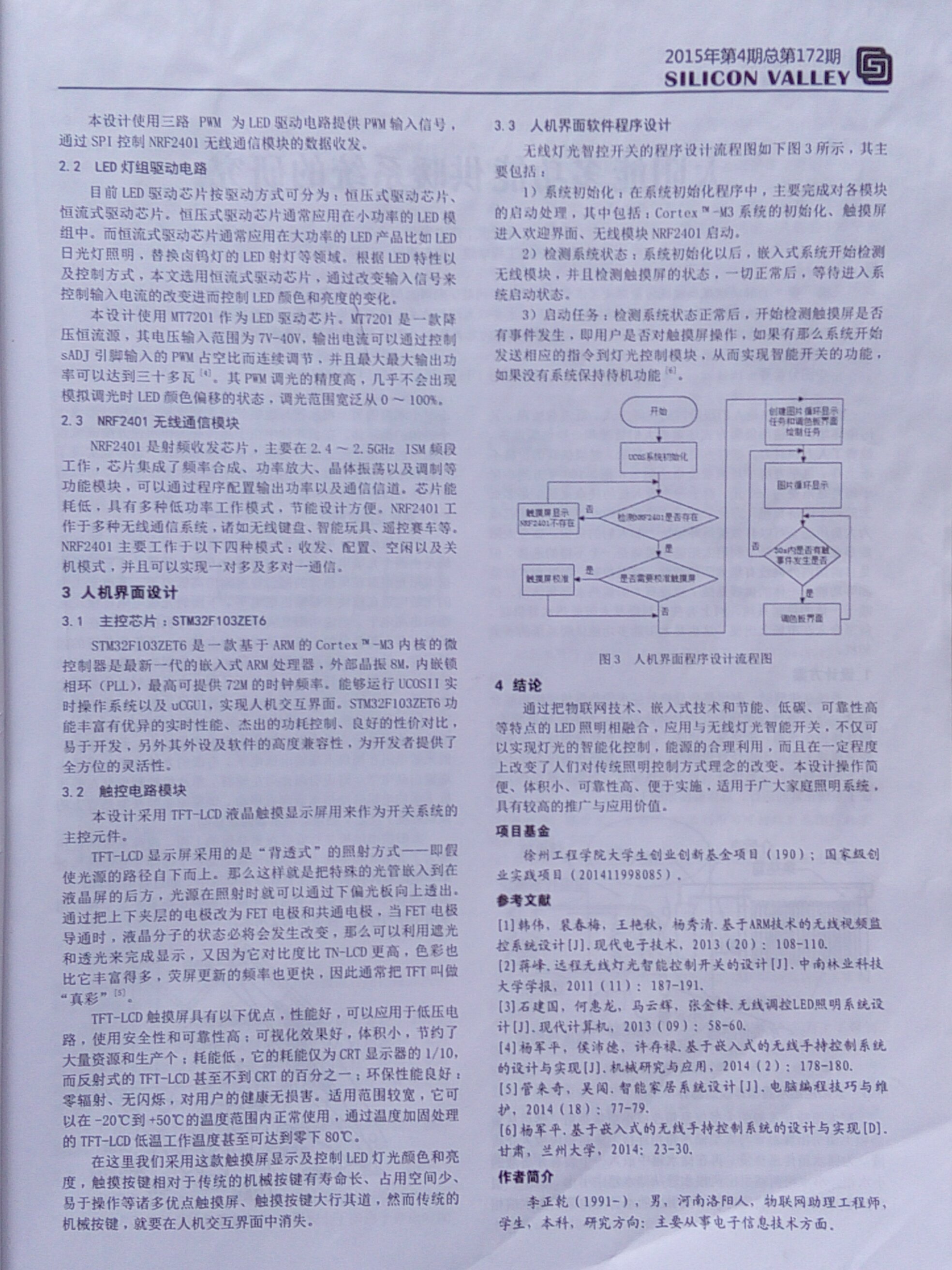
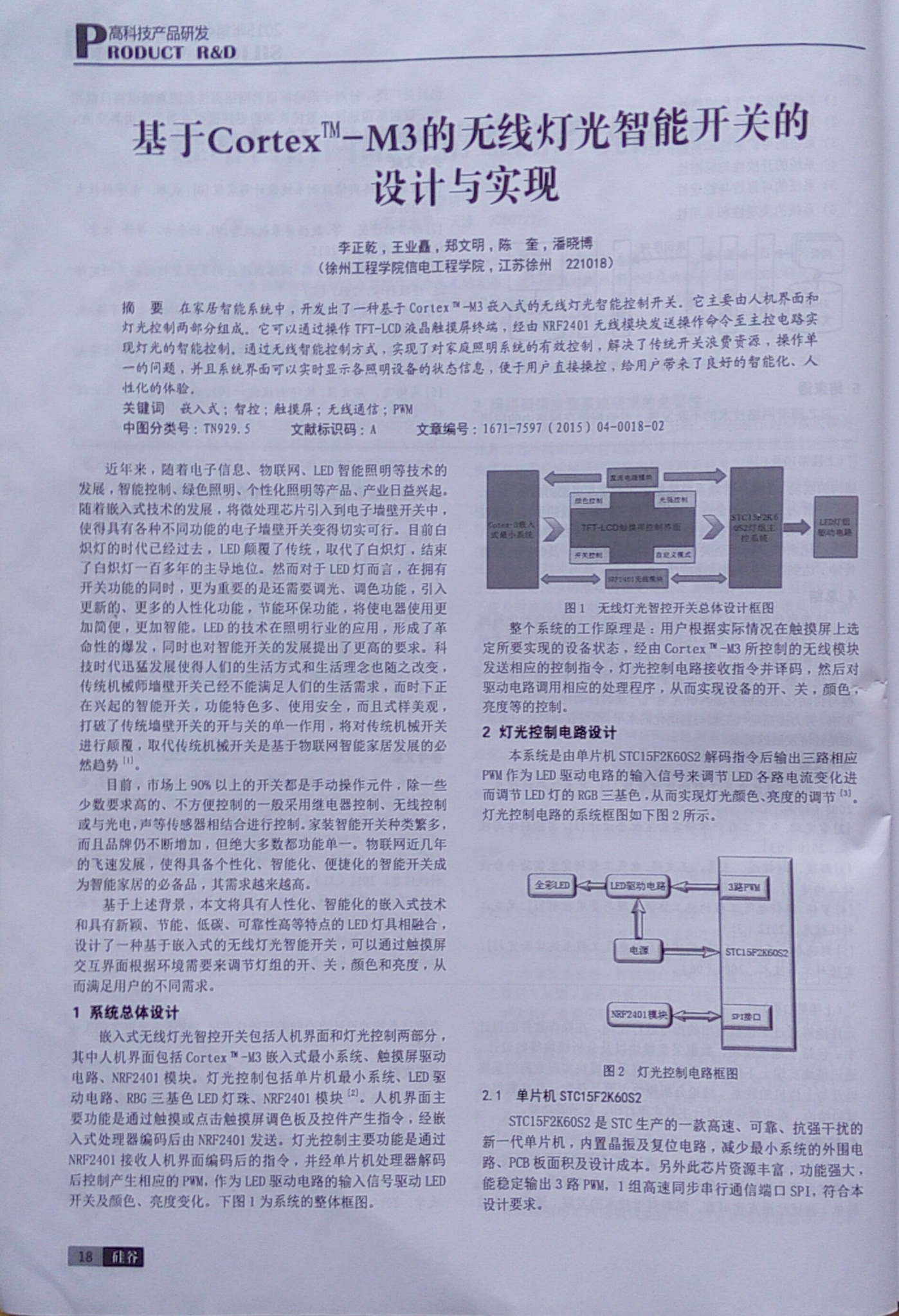
基于手机蓝牙控制的i-Light智能家居彩灯和控制系统的实用新型专利



## 附录5期刊论文

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统的期刊论文





## 附录6获奖证书

基于手机蓝牙控制的-Light智能家居彩灯和控制系统获奖证书



## 附录7系统手机端程序

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统的手机端程序

package com.qian.bluetooth;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.UUID;

import com.qian.bluetooth.sysinfo.SysInfoUtil;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences.Editor;

import android.util.Log;

import android.widget.Toast;

public class BluetoothActivity extends Activity {

private static final int OPEN\_BLUETOOTH\_REQUESTCODE = 2;

private static final int DISCOVERY\_BLUETOOTH\_REQUESTCODE = 1;

protected static boolean isConnected = false;

protected String bluetoothMAC;

private static String TAG = "BaseActivity.java--";

protected static BluetoothSocket socket = null;

private static InputStream bluetooth\_inStream = null;

private static OutputStream bluetooth\_outStream = null;

//打开蓝牙

protected void openBluetooth() {

Intent intent = new Intent();

intent.setAction(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);

startActivityForResult(intent, OPEN\_BLUETOOTH\_REQUESTCODE);

}

// 判断是否已连接蓝牙设备

public boolean isConnected() {

return isConnected;

}

protected void initMAC() {

bluetoothMAC = getStrData("BluetoothMAC");

}

private String getStrData(String key) {

return getSharedPreferences(new SysInfoUtil(this).getPackageName(),

Context.MODE\_PRIVATE).getString(key, null);

}

private void saveMAC(String key, String macAddress) {

Editor editor = getSharedPreferences(

new SysInfoUtil(this).getPackageName(), Context.MODE\_PRIVATE)

.edit();

editor.putString(key, macAddress);

editor.commit();

}

//搜索蓝牙

protected void discoveryBluetoothMAC() {

Intent intent = new Intent();

intent.setClass(this, ListActivityMAC.class);

startActivityForResult(intent, DISCOVERY\_BLUETOOTH\_REQUESTCODE);

}

//蓝牙连接

protected boolean createBluetoothConnection() {

BluetoothDevice device;

if (isConnected) {

try {

bluetooth\_inStream.close();

bluetooth\_outStream.close();

socket.close();

} catch (IOException e) {

bluetooth\_inStream = null;

bluetooth\_outStream = null;

socket = null;

isConnected = false;

}

}

try {

device = BluetoothAdapter.getDefaultAdapter().getRemoteDevice(

bluetoothMAC);

socket = device.createRfcommSocketToServiceRecord(UUID

.fromString("00001101-0000-1000-8000-00805F9B34FB"));

socket.connect();

// 保存MAC

saveMAC("BluetoothMAC", bluetoothMAC);

bluetooth\_inStream = socket.getInputStream();

bluetooth\_outStream = socket.getOutputStream();

Log.e(TAG, "BluetoothSocket获取输入、输出流成功!");

isConnected = true;

return true;

} catch (Exception e) {

bluetoothMAC = null;

saveMAC("BluetoothMAC", bluetoothMAC);

Log.e(TAG, "BluetoothSocket获取输入、输出流失败!");

isConnected = false;

return false;

}

}

//字节流发送数据

protected int sendDate(byte[] data) {

if (isConnected) {

try {

bluetooth\_outStream.write(data);

return 1;

} catch (IOException e) {

terminateConnection();

return -1;

}

} else

return 0;

}

// 中断与外部蓝牙设备的连接

private void terminateConnection() {

if (isConnected) {

isConnected = false;

try {

bluetooth\_inStream.close();

bluetooth\_outStream.close();

socket.close();

} catch (IOException e) {

bluetooth\_inStream = null;

bluetooth\_outStream = null;

socket = null;

}

}

Log.e(TAG, "发送数据异常时中断与外部蓝牙设备的连接");

}

protected void showInfo(String info) {

Toast.makeText(this, info, Toast.LENGTH\_SHORT).show();

}

}

package com.qian.palette;

import java.io.IOException;

import com.qian.bluetooth.BluetoothActivity;

import com.qian.bluetooth.BluetoothCtrl;

import com.qian.palette.R;

import android.app.AlertDialog;

import android.app.AlertDialog.Builder;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.AsyncTask;

import android.os.Bundle;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.Window;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.CompoundButton;

import android.widget.ImageButton;

import android.widget.Switch;

import android.widget.CompoundButton.OnCheckedChangeListener;

public class MainActivity extends BluetoothActivity implements OnClickListener {

private Switch allLights;

private Button[] btns = new Button[8];

private int[] ids = new int[] { R.id.room1, R.id.room2, R.id.room3,

R.id.room4, R.id.room5, R.id.room6, R.id.duanBtn, R.id.connBtn };

private BluetoothAdapter bTA;

private AlertDialog dialog = null;

private byte[] data;

private String TAG = "RoomActivity--.java";

/\*

\* AsyncTask:创建 BluetoothSocket对象,并使其初始化和管理与外部蓝牙设备连接.

\*/

class MyTask extends AsyncTask<String[], String, Integer> {

@Override

protected Integer doInBackground(String[]... params) {

Log.e(TAG + "AsyncTask", "doInBackground() ");

if (createBluetoothConnection())

return 1;

else

return -1;

}

@Override

protected void onPostExecute(Integer result) {

Log.e(TAG + "AsyncTask", " onPostExecute(Integer integer) ="

+ result.intValue());

if (result.intValue() == 1)

showInfo(getString(R.string.msg\_connect\_ok));

else

showInfo(getString(R.string.msg\_connect\_fail));

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

getWindow().requestFeature(Window.FEATURE\_NO\_TITLE);

setContentView(R.layout.activity\_main);

init();

initMAC();

}

private void init() {

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

btns[i] = (Button) findViewById(ids[i]);

btns[i].setOnClickListener(this);

}

allLights = (Switch) findViewById(R.id.all\_lights);

allLights.setOnCheckedChangeListener(new SwitchListener());

bTA = BluetoothAdapter.getDefaultAdapter();

if (!bTA.isEnabled()) {

bTA.enable();

}

}

class SwitchListener implements OnCheckedChangeListener {

@Override

public void onCheckedChanged(CompoundButton buttonView,

boolean isChecked) {

// 控制灯的开关.

if (isChecked) {

sendRGB("255x255y255z\*");

// doSend(1);

} else {

sendRGB("0x0y0z\*");

// doSend(2);

}

}

}

void sendRGB(String dStr) {

int flag;

try {

flag = sendDate(dStr.getBytes());

if (flag == -1)

showInfo(getString(R.string.msg\_\_SendBytesErr));

else if (flag == 0)

showInfo(getString(R.string.msg\_please\_connect));

} catch (Exception e) {

showInfo("数据格式异常!");

}

}

// 处理灯的开关,单命令控制.(接收方接收到的是:ASCII值.)

public void doSend(int f) {

Log.e(TAG + "126-----", "doSend():");

data = new byte[1];

// 将十进制数强制转化为 byte.

if (f == 1) {

data[0] = (byte) (0xff & Integer.decode("0x4f"));

} else if (f == 2) {

data[0] = (byte) (0xff & Integer.decode("0x43"));

}

int flag = sendDate(data);

if (flag == -1)

showInfo(getString(R.string.msg\_\_SendBytesErr));

else if (flag == 0)

showInfo(getString(R.string.msg\_please\_connect));

}

@Override

public void onClick(View v) {

Intent intent = new Intent();

intent.setClass(this, GuestRoom.class);

switch (v.getId()) {

case R.id.room1:

this.startActivity(new Intent(this, PaletteActivity1.class));

break;

case R.id.room2:

this.startActivity(new Intent(this, PaletteActivity2.class));

break;

case R.id.room3:

this.startActivity(new Intent(this, PaletteActivity3.class));

break;

case R.id.room4:

this.startActivity(new Intent(this, PaletteActivity4.class));

break;

case R.id.room5:

this.startActivity(new Intent(this, PaletteActivity5.class));

break;

case R.id.room6:

intent.putExtra("room", "room4");

this.startActivity(intent);

break;

case R.id.duanBtn:

Log.e(TAG, "断开按钮");

if (socket != null) {

try {

socket.close();

isConnected = false;

showInfo("断开成功!");

} catch (IOException e) {

showInfo("断开失败!");

}

}

break;

case R.id.connBtn:

connectDevices();

break;

}

}

// 连接蓝牙设备.

private void connectDevices() {

// 连接设备前,判断蓝牙是否开启.

if (!bTA.isEnabled()) {

openBluetooth();

return;

}

if (!isConnected()) {

if (bluetoothMAC == null) {

// 搜索蓝牙设备

discoveryBluetoothMAC();

return;

}

// bluetoothMAC != null,连接

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle(getString(R.string.menu\_main\_Connection));

builder.setMessage(getString(R.string.msg\_connect\_history));

builder.setPositiveButton(getString(R.string.btn\_connect),

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

new MyTask().execute(new String[0]);

}

});

builder.setNegativeButton(getString(R.string.btn\_reSearch),

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

discoveryBluetoothMAC();

}

});

builder.create().show();

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

Log.e(TAG + "", "onActivityResult(int,int,Intent)");

Log.e("onActivityResult(int,int,Intent)", "Intent data=" + data);

if (requestCode == 2 && resultCode == RESULT\_CANCELED) {// 请求打开蓝牙

finish();

} else if (requestCode == 1 && resultCode == -1) {

BluetoothDevice device = (BluetoothDevice) data

.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

if (device.getBondState() == BluetoothDevice.BOND\_NONE) {

try {

showInfo(getString(R.string.msg\_actDiscovery\_Bluetooth\_Bond\_none));

BluetoothCtrl.createBond(device);

} catch (Exception e) {

showInfo(getString(R.string.msg\_actDiscovery\_Bluetooth\_Bond\_fail));

}

} else {

bluetoothMAC = device.getAddress();

Log.e(TAG, "获取MAC=" + bluetoothMAC);

// 执行异步任务 asynchronous

Log.e(TAG, "获取MAC=" + new String[0]);

new MyTask().execute(new String[0]);

}

}

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.about) {

showDialogInfo(R.layout.about\_text);

} else if (id == R.id.help) {

showDialogInfo(R.layout.help\_text);

}

return true;

}

private void showDialogInfo(int id) {

AlertDialog.Builder dialogBuilder = new Builder(this);

LayoutInflater inflater = LayoutInflater.from(this);

final View view = inflater.inflate(id, null);

ImageButton imgBtn = (ImageButton) view.findViewById(R.id.backBtnId);

imgBtn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

dialog.dismiss();

}

});

dialogBuilder.setView(view);

dialog = dialogBuilder.create();

dialog.show();

}

@Override

protected void onDestroy() {

super.onDestroy();

Log.e("onDestroy()--", "关闭蓝牙");

if (bTA.isEnabled()) {

bTA.disable();

}

}

}

package com.qian.palette;

import com.qian.bluetooth.BluetoothActivity;

import com.qian.bluetooth.ConstantUtil;

import com.qian.palette.R;

import com.qian.palette.PaletteView.onColorChangedListener;

import android.graphics.Color;

import android.os.Bundle;

import android.util.Log;

import android.view.Window;

import android.widget.CompoundButton;

import android.widget.CompoundButton.OnCheckedChangeListener;

import android.widget.ImageView;

import android.widget.ToggleButton;

public class PaletteActivity1 extends BluetoothActivity implements

onColorChangedListener {

private ToggleButton switchOff;

private ImageView img\_state;

private byte[] data;

private static String initColor = "#000000";

// 控制是否可发送RGB.

private boolean isOpened = false;

// 接收并传递开关的状态值(开->true.)

private boolean flag\_light = false;

private String TAG = "PaletteActivity.java---------------";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// 设置布局

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

setContentView(R.layout.palette\_layout);

init();

}

private void init() {

switchOff = (ToggleButton) findViewById(R.id.switchOnOff);

switchOff.setOnCheckedChangeListener(new SwitchListener());

img\_state = (ImageView) findViewById(R.id.light\_state);

PaletteView palette = (PaletteView) this.findViewById(R.id.paletteView);

palette.setOnColorChangedListener(this);

}

@Override

public void setOnColorChanged(int color) {

// 提取RGB值.

convertToRGBHEX(color);

Log.e(TAG + "----------", "rgb:" + convertToRGBHEX(color));

// 将RGB发送.(当点击和滑动事件同时发生时,只在滑动情况下发送数据.)

if (isOpened && ConstantUtil.touchEventCtrl) {

Log.e("发送一次数据", "发送一次数据");

if (!initColor.equals(convertToRGBHEX(color)))

sendRGB(convertToRGB(color));

}

}

private void sendRGB(String dStr) {

int flag;

try {

flag = sendDate(dStr.getBytes());

if (flag == -1)

showInfo(getString(R.string.msg\_\_SendBytesErr));

else if (flag == 0)

showInfo(getString(R.string.msg\_please\_connect));

else {

isOpened = flag\_light;

if (isOpened)

switchOff.setTextOn("开灯");

else

switchOff.setTextOff("关灯");

img\_state.setImageResource(isOpened ? R.drawable.on

: R.drawable.off);

}

} catch (Exception e) {

showInfo("数据格式异常!");

}

}

// 将颜色值 int 转换为 对应的 hex

public static String convertToRGB(int color) {

String red = Integer.toHexString(Color.red(color));

String green = Integer.toHexString(Color.green(color));

String blue = Integer.toHexString(Color.blue(color));

if (red.length() == 1) {

red = "0" + red;

}

if (green.length() == 1) {

green = "0" + green;

}

if (blue.length() == 1) {

blue = "0" + blue;

}

String str\_red = null;

String str\_green = null;

String str\_blue = null;

try {

str\_red = String.valueOf(Integer.decode("#" + red));

str\_green = String.valueOf(Integer.decode("#" + green));

str\_blue = String.valueOf(Integer.decode("#" + blue));

} catch (Exception e) {

}

return str\_red + "x" + str\_green + "y" + str\_blue + "z"+"!";

}

public static String convertToRGBHEX(int color) {

String red = Integer.toHexString(Color.red(color));

String green = Integer.toHexString(Color.green(color));

String blue = Integer.toHexString(Color.blue(color));

if (red.length() == 1) {

red = "0" + red;

}

if (green.length() == 1) {

green = "0" + green;

}

if (blue.length() == 1) {

blue = "0" + blue;

}

return "#" + red + green + blue;

}

class SwitchListener implements OnCheckedChangeListener {

@Override

public void onCheckedChanged(CompoundButton buttonView,

boolean isChecked) {

// 控制灯的开关.

if (isChecked) {

flag\_light = isChecked;

sendRGB("255x255y255z!");

//doSend("0x42");//B

} else {

flag\_light = isChecked;

sendRGB("0x0y0z!");

//doSend("0x62");//b

}

}

}

// 处理灯的开关,单命令控制.(接收方接收到的是:ASCII值.)

public void doSend(String hexStr) {

Log.e(TAG + "126-----", "doSend():");

data = new byte[1];

// 将十进制数强制转化为 byte.

data[0] = (byte) (0xff & Integer.decode(hexStr));

int flag = sendDate(data);

if (flag == -1)

showInfo(getString(R.string.msg\_\_SendBytesErr));

else if (flag == 0)

showInfo(getString(R.string.msg\_please\_connect));

else {

isOpened = flag\_light;

if (isOpened)

switchOff.setTextOn("开灯");

else

switchOff.setTextOff("关灯");

img\_state.setImageResource(isOpened ? R.drawable.on

: R.drawable.off);

}

Log.e("是否灯已经打开...", "isOpened=" + isOpened);

}

}

## 附录8系统LED控制器程序

基于手机蓝牙控制的i-Light智能家居彩灯和控制系统的LED控制器程序

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "reg52.h"

#include "Delay.h"

#include "2401.h"

#include "stdlib.h"

#include "string.h"

#include "uart.h"

unsigned char pwm0=0x40,pwm1=0x40,pwm2=0x40,flag0=1;

unsigned char charx,chary,charz ,str[20]="",strx[20]="";

uchar test[]="";uchar RevTempDate0[20]="";

char \* mid(char \*dst,char \*src, int n,int m);

void System\_init(void);

unsigned char ReceiveDate();

void main()

{ System\_init();//系统初始化

while(1){};

}

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

/\*函数原型: void System\_init(void)

/\*函数功能:系统初始化

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

void System\_init(void)

{ CCON = 0;//初始化PCA控制寄存器//PCA定时器停止//清除CF标志//清除模块中断标志

CL = 0; CH = 0;//复位PCA寄存器

CMOD = 0x0A; //设置PCA时钟源 //时钟源输入：sysclk/4

PCA\_PWM0 = 0x00; //PCA模块0工作于8位PWM

CCAP0H = CCAP0L = 0x96;//PWM0的占空比为87.5% ((100H-20H)/100H)

CCAPM0 = 0x02;//PCA模块0为8位PWM模式 //脉宽调节模式

PCA\_PWM1 = 0x00; // PCA模块0工作于8位PWM

CCAP1H = CCAP1L = 0x96;//PWM1的占空比为87.5% ((100H-20H)/100H)

CCAPM1 = 0x02;//PCA模块0为8位PWM模式 //脉宽调节模式

PCA\_PWM2 = 0x00;// PCA模块0工作于8位PWM

CCAP2H = CCAP2L = 0x96;//PWM2的占空比为87.5% ((100H-20H)/100H)

CCAPM2 = 0x02; //PCA模块0为8位PWM模式 //脉宽调节模式

CR = 1; //禁止PCA定时器溢出中断

CCAP0H = CCAP0L = pwm0; //产生PWM0

CCAP1H = CCAP1L = pwm1; //产生PWM1

CCAP2H = CCAP2L = pwm2; //产生PWM2

Uart\_Init();//串口初始化

NRF24L01Int();//NRF2401初始化

}

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

/\*函数原型:unsigned char ReceiveDate(void)

/\*函数功能:NRF2401接收数据

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

unsigned char ReceiveDate(void)

{ uchar i;//接收通道号

sta=NRFReadReg(R\_REGISTER+STATUS);//发送数据后读取状态寄存器

if(sta&0x40) // 判断是否接收到数据 MASK\_RX\_DR

{ flag0=1;CE=0;//待机

NRFReadRxDate(R\_RX\_PAYLOAD,RevTempDate0,RX\_DATA\_WITDH); // 从RXFIFO读取数据通道0

for(i=0;i<strlen(RevTempDate0);i++)//清空RevTempDate0缓存

RevTempDate0[i]='\0';}

else flag0=0;return flag0;}

char \* mid(char \*dst,char \*src, int n,int m) /\*n为长度，m为位置\*/

{ char \*p = src; char \*q = dst; int len = strlen(src);

if(n>len) n = len-m;if(m<0) m=0;if(m>len) return NULL;

p += m;while(n--) \*(q++) = \*(p++);\*(q)='\0';

return dst;}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*串口中断服务子程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*函数原型: void Uart\_Receive() interrupt 4

/\*函数功能:执行串口中断程序

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

void Uart\_Receive() interrupt 4

{ unsigned char temp,k,count,charx,chary,charz ,str[20]="",strx[20]="";//定义变量和缓存

if(RI==1){

RI = 0;//清空接收中断标志位

temp = SBUF;test[count++]=temp;//接收数据test[count]='\0';//添加结束符

switch(temp){

case '!'://数据处理后直接产生3路PWM输入LED控制器驱动电路

strcpy(strx,test);charx=strchr(strx,'x')-strx+1;//截取PWM0字符型数据

chary=strchr(strx,'y')-strx+1;//截取PWM1字符型数据

charz=strchr(strx,'z')-strx+1;//截取PWM2字符型0数据

pwm0=atoi(mid(str,strx,charx-1,0));// PWM0字符串转为整数

pwm1=atoi(mid(str,strx,chary-charx-1,charx)); // PWM1字符串转为整数

pwm2=atoi(mid(str,strx,charz-chary-1,chary)); // PWM2字符串转为整数

CCAP0H = CCAP0L = pwm0; //产生PWM0

CCAP1H = CCAP1L = pwm1; //产生PWM1

CCAP2H = CCAP2L = pwm2; //产生PWM2

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '#'://通过NRF2401模块发送接收的数据到第二个LED控制器

NRFSetTxMode(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '$'://通过NRF2401模块发送接收的数据到第三个LED控制器

NRFSetTxMode1(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '%'://通过NRF2401模块发送接收的数据到第四个LED控制器

NRFSetTxMode2(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '^'://通过NRF2401模块发送接收的数据到第五个LED控制器

NRFSetTxMode3(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '&'://通过NRF2401模块发送接收的数据到第六个LED控制器

NRFSetTxMode4(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

case '\*'://通过NRF2401模块发送接收的数据到第七个LED控制器

NRFSetTxMode5(test);//设置NRF2401为发送模式

while(CheckACK()); //等待发送完毕

for(k=0;k<strlen(strx);k++)strx[k]='\0';//清空strx缓存

for(k=0;k<strlen(test);k++)test[k]=0;//清空test缓存

count=0;//接收字节数清零break;跳出switch语句

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NRF24L01驱动程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include"2401.h"

#include"delay.h"

#define uint unsigned int

#define uchar unsigned char

uchar code RxAddr0[]={0x34,0x43,0x10,0x10,0x01};//编号0接收地址这个地址和发送方地址一样!

uchar code RxAddr1[]={0xc2,0xc2,0xc2,0xc2,0xc1};//编号1

uchar code RxAddr2[]={0xc3};//编号2

uchar code RxAddr3[]={0xc4};//编号3

uchar code RxAddr4[]={0xc5};//编号4

uchar code RxAddr5[]={0xc6};//编号5

uchar code TxAddr[]={0xc1,0xb2,0xb3,0xb4,0x01};//发送地址

uchar code TxAddr1[]={0xb0,0xb2,0xb3,0xb4,0x01};//编号1

uchar code TxAddr2[]={0xb1,0xb2,0xb3,0xb4,0x01};//编号2

uchar code TxAddr3[]={0xb2,0xb2,0xb3,0xb4,0x01};//编号3

uchar code TxAddr4[]={0xb3,0xb2,0xb3,0xb4,0x01};//编号4

uchar code TxAddr5[]={0xb4,0xb2,0xb3,0xb4,0x01};//编号5

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*状态标志\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar bdata sta; //状态标志

sbit RX\_DR=sta^6;

sbit TX\_DS=sta^5;

sbit MAX\_RT=sta^4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI时序函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar NRFSPI(uchar date)

{

uchar i;

for(i=0;i<8;i++) // 循环8次

{ if(date&0x80)MOSI=1;

else MOSI=0; // byte最高位输出到MOSI

date<<=1; // 低一位移位到最高位

SCLK=1;

if(MISO) // 拉高SCK，nRF24L01从MOSI读入1位数据，同时从MISO输出1位数据

date|=0x01; // 读MISO到byte最低位

SCLK=0; // SCK置低

}

return(date); // 返回读出的一字节

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NRF24L01初始化函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void NRF24L01Int()

{ Delay(2);//让系统什么都不干

CE=0; //待机模式1

CSN=1;

SCLK=0;

IRQ=1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI读寄存器一字节函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar NRFReadReg(uchar RegAddr)

{ uchar BackDate;

CSN=0;//启动时序

NRFSPI(RegAddr);//写寄存器地址

BackDate=NRFSPI(0x00);//写入读寄存器指令

CSN=1;

return(BackDate); //返回状态

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI写寄存器一字节函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar NRFWriteReg(uchar RegAddr,uchar date)

{

uchar BackDate;

CSN=0;//启动时序

BackDate=NRFSPI(RegAddr);//写入地址

NRFSPI(date);//写入值

CSN=1;

return(BackDate);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI读取RXFIFO寄存器的值\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar NRFReadRxDate(uchar RegAddr,uchar \*RxDate,uchar DateLen)

{ //寄存器地址//读取数据存放变量//读取数据长度//用于接收

uchar BackDate,i;

CSN=0;//启动时序

BackDate=NRFSPI(RegAddr);//写入要读取的寄存器地址

for(i=0;i<DateLen;i++)RxDate[i]=NRFSPI(0);//读取数据

CSN=1;

return(BackDate);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI写入TXFIFO寄存器的值\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar NRFWriteTxDate(uchar RegAddr,uchar \*TxDate,uchar DateLen)

{ //寄存器地址//写入数据存放变量//读取数据长度//用于发送

uchar BackDate,i;

CSN=0;

BackDate=NRFSPI(RegAddr);//写入要写入寄存器的地址

for(i=0;i<DateLen;i++)NRFSPI(\*TxDate++);//写入数据

CSN=1;

return(BackDate);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*NRF设置为发送模式并发送数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void NRFSetTxMode(uchar \*TxDate)

{//发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P0,TxAddr,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x03f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x03f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*NRF设置为发送模式并发送数据\*\*\*\*\*\*\*\*\*\*\*\*\*/

void NRFSetTxMode1(uchar \*TxDate)

{//发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr1,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P1,TxAddr1,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

void NRFSetTxMode2(uchar \*TxDate)

{//发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr2,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P2,TxAddr2,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

void NRFSetTxMode3(uchar \*TxDate)

{ //发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr3,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P3,TxAddr3,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

void NRFSetTxMode4(uchar \*TxDate)

{ //发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr4,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P4,TxAddr4,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

void NRFSetTxMode5(uchar \*TxDate)

{ //发送模式

CE=0;

NRFWriteTxDate(W\_REGISTER+TX\_ADDR,TxAddr5,TX\_ADDR\_WITDH);//写寄存器指令+接收地址使能指令+接收地址+地址宽度

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P5,TxAddr5,TX\_ADDR\_WITDH);//为了应答接收设备，接收通道0地址和发送地址相同

NRFWriteTxDate(W\_TX\_PAYLOAD,TxDate,TX\_DATA\_WITDH);//写入数据

/\*\*\*\*\*\*下面有关寄存器配置\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能接收通道0自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道0

NRFWriteReg(W\_REGISTER+SETUP\_RETR,0x0a); // 自动重发延时等待250us+86us，自动重发10次

NRFWriteReg(W\_REGISTER+RF\_CH,0x40); // 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0e); // CRC使能，16位CRC校验，上电

CE=1;

Delay(5);//保持10us秒以上

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*NRF设置为接收模式并接收数据\*\*\*\*\*\*\*\*\*\*\*\*\*/

//接收模式

void NRFSetRXMode()

{ CE=0;

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P0,RxAddr0,TX\_ADDR\_WITDH); // 接收设备接收通道0使用和发送设备相同的发送地址

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P1,RxAddr1,TX\_ADDR\_WITDH); // 接收设备接收通道1使用和发送设备相同的发送地址

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P2,RxAddr2,1); // 接收设备接收通道0使用和发送设备相同的发送地址

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P3,RxAddr3,1); // 接收设备接收通道1使用和发送设备相同的发送地址

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P4,RxAddr4,1); // 接收设备接收通道0使用和发送设备相同的发送地址

NRFWriteTxDate(W\_REGISTER+RX\_ADDR\_P5,RxAddr5,1); // 接收设备接收通道1使用和发送设备相同的发送地址

NRFWriteReg(W\_REGISTER+EN\_AA,0x3f); // 使能数据通道5自动应答

NRFWriteReg(W\_REGISTER+EN\_RXADDR,0x3f); // 使能接收通道5

NRFWriteReg(W\_REGISTER+RX\_PW\_P0,TX\_DATA\_WITDH); // 接收通道0选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RX\_PW\_P1,TX\_DATA\_WITDH); // 接收通道1选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RX\_PW\_P2,TX\_DATA\_WITDH); // 接收通道0选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RX\_PW\_P3,TX\_DATA\_WITDH); // 接收通道1选择和发送通道相同有效数据宽度 NRFWriteReg(W\_REGISTER+RX\_PW\_P0,TX\_DATA\_WITDH); // 接收通道0选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RX\_PW\_P4,TX\_DATA\_WITDH); // 接收通道1选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RX\_PW\_P5,TX\_DATA\_WITDH); // 接收通道0选择和发送通道相同有效数据宽度

NRFWriteReg(W\_REGISTER+RF\_CH,0x40);// 选择射频通道0x40

NRFWriteReg(W\_REGISTER+RF\_SETUP,0x07); // 数据传输率1Mbps，发射功率0dBm，低噪声放大器增益

NRFWriteReg(W\_REGISTER+CONFIG,0x0f); // CRC使能，16位CRC校验，上电，接收模式

CE = 1; Delay(5);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*检测应答信号\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar CheckACK()

{ //用于发射

sta=NRFReadReg(R\_REGISTER+STATUS); // 返回状态寄存器

if(TX\_DS||MAX\_RT) //发送完毕中断

{ NRFWriteReg(W\_REGISTER+STATUS,0xff); // 清除TX\_DS或MAX\_RT中断标志

CSN=0; NRFSPI(FLUSH\_TX); CSN=1;

return(0);

}

else return(1);

}

/\*\*\*\*\*\*\*\*\*\*判断是否接收收到数据，接到就从RX取出\*\*\*\*\*\*\*\*\*\*\*\*/

//用于接收模式

uchar NRFRevDate(uchar \*RevDate)

{ uchar RevFlags=0;

sta=NRFReadReg(R\_REGISTER+STATUS);//发送数据后读取状态寄存器

if(sta&0x4) // 判断是否接收到数据

{ CE=0; //SPI使能

NRFReadRxDate(R\_RX\_PAYLOAD,RevDate,RX\_DATA\_WITDH);// 从RXFIFO读取数据

RevFlags=1; //读取数据完成标志

}

NRFWriteReg(W\_REGISTER+STATUS,0xff); //接收到数据后RX\_DR,TX\_DS,MAX\_PT都置高为1，通过写1来清楚中断标

CSN=0; NRFSPI(FLUSH\_RX); CSN=1;

return(RevFlags);

}