****

****

Beijing Institute of Petrochemical Technology

**短距离无线通信实践个人总结报告**

|  |  |
| --- | --- |
| **学院** | **信息工程学院** |
| **专业** | **通信工程** |
| **项目名** | **基于Zigbee工具的地下酒窖环境监测系统项目** |
| **班级** | **通192** |
| **项目** | **石冠伟** |
| **学号** | **2019311359** |
| **指导教师** | **田小平** |

2021年9月10日

目录

[目录 3](#_Toc82157600)

[1基于Zigbee工具的地下酒窖环境监测系统项目的实践环节 1](#_Toc82157601)

[1.1 项目目标 1](#_Toc82157602)

[1.2 项目方案 1](#_Toc82157603)

[1.3 项目协议设计 1](#_Toc82157604)

[1.4 项目实现效果 3](#_Toc82157605)

[1.5 本人在项目中的贡献及收获 4](#_Toc82157606)

[2 实验九实践环节 6](#_Toc82157607)

[2.1 考核要求 6](#_Toc82157608)

[2.2 比较实验设计 6](#_Toc82157609)

[2.3 原始实验效果 11](#_Toc82157610)

[2.4 修改参数后的实验效果 11](#_Toc82157611)

[2.5 本人在实验九实践环节中的收获 11](#_Toc82157612)

[3 实验八实践环节 12](#_Toc82157613)

[3.1 考核要求 12](#_Toc82157614)

[3.2 比较实验设计 12](#_Toc82157615)

[3.3 原始实验效果 19](#_Toc82157616)

[3.4 修改参数后的实验效果 19](#_Toc82157617)

[3.5 本人在实验八实践环节中的收获 19](#_Toc82157618)

[4 实验三实践环节 20](#_Toc82157619)

[4.1 考核要求 20](#_Toc82157620)

[4.2 比较实验设计 20](#_Toc82157621)

[4.3 原始实验效果 29](#_Toc82157622)

[4.4 修改参数后的实验效果 29](#_Toc82157623)

[4.5 本人在实验三实践环节中的收获 29](#_Toc82157624)

1基于Zigbee工具的地下酒窖环境监测系统项目的实践环节

# 1.1 项目目标

设计一个地下酒窖监视系统，来判断地下酒窖是否有人经过，来检测地下酒窖里的温度和湿度，看是否适合酒的发酵；为了在地下酒窖工作人员的身心健康，该地下酒窖监视系统能够检测地下酒窖的酒精气体浓度，达到来判断酒精气体浓度是否超标，因为超标有爆炸的危险，这严重危害地下酒窖的身心健康。

# 1.2 项目方案

项目方案：我们组的实验是想设计一个地下酒窖检测器，来实现对地下酒窖的检测，从而达到维护好地下酒窖的目的。我们方案是用三个传感器来实现对地下酒窖的检测功能，其中包括气体传感器、红外传感器和温湿度传感器。

我们先做外观，当该程序运行的时候，先把我们组的各个成员的信息显现出来，然后我们先把温湿度传感器跑一遍，没错误的话我们就在温湿度传感器的基础上来增加红外传感器和气体传感器，最后就是实践，检测当触发条件发生时，传感器是否能正常运行以及在该板子上是否能显示出相应的效果。

# 1.3 项目协议设计

1.ZigBee 协议栈采用分层的思想，好处在于上层实现的功能对于下层来说是不知道，上层可以调用下层提供的函数来实现某些功能。

2.各层之间的数据传递是通过服务接入点（Service Access Point 简称：SAP）来实现的。

3.服务接入点主要两种类型：一种是数据传输服务接入点，另一种是管理的服务 接入点。

4.建立工程时，也表现了这个分层思想，便于代码的编写和管理。

Zigbee协议是一种通信标准，通信双方需要按照这一标准进行正常的数据发射和接收。协议栈是协议的具体实现形式，通俗讲协议栈就是协议和用户之间的一个接口，此次实验通过使用协议栈来使用这个协议，进而实现无线数据收发。

如图1所示：Zigbee协议分为两部分，IEEE 802.15.4定义了PHY（物理层）和MAC（介质访问层）技术规范；Zigbee联盟定义了NWK（网络层）、APS（应用程序支持层）、APL（应用层）技术规范。Zigbee协议栈就是将各个层定义的协议都集合在一起，以函数的形式实现，并给用户提供API（应用层），用户可以直接调用。

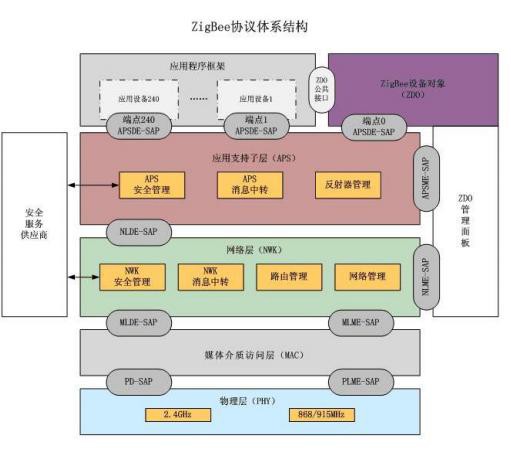


图1 ZigBee无线网络协议层的架构图

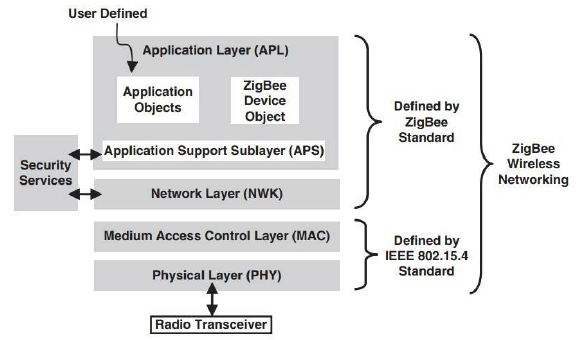
Zigbee协议栈是协议的实现，可以理解为代码，库函数，供上层应用调用，协议较底下的层与应用是相互独立的。

此次实验使用Zigbee协议栈的步骤为：

① 组网：调用协议栈组网函数、加入网络函数，实现网络的建立和节点的加入

② 发送：发送节点调用协议栈的发送函数，实现数据无线发送

③ 接收：接收节点调用协议栈的无线接收函数，实现无线数据接收



# 1.4 项目实现效果

我们做的地下酒窖监视系统主要实现三个功能。

第一个功能：判断地下酒窖是否有人经过，这个功能可以起到防盗窃的作用，当然还有其他许许多多的作用，比如可以起到监测的作用，当在一定危险时期，我们可以不用保安守着这里，我们可以用这一个功能来判断那个时期是否有人进入，从而起到预防的作用。效果：我们运用的主要是红外传感器，当有人经过这个传感器，我们这个地下酒窖监视系统上的灯光就会亮。

第二个功能：判断地下酒窖里的温湿度是否适宜；我们这个功能主要实现的是创造一个合适的温度和湿度来让地下酒窖里的酒更好地发酵。效果是我们的地下酒窖监视器上会有一个屏幕来实时显示地下酒窖里的湿度和温度，并且每时每刻都会刷新你，所以我们可以看到实时温度，因此效果就是我们的地下酒窖监视器上会显示地下酒窖的湿度和温度。

第三个功能是检测地下酒窖里酒精的浓度；由于酒精是易燃易爆气体，所以需要对他进行检测，我们运用气体传感器来对酒精浓度进行检测。效果是当地下酒窖中的酒精浓度超标时，我们的地下酒窖检测器上的灯会亮，当然我们可以设置成一亮一亮的，也可以改成其他形式。

# 1.5 本人在项目中的贡献及收获

项目中的贡献：我在本组中担任的主要是协助组长、熟悉代码和查询资料的作用，在外面此次的项目中，我们组的组长和组员都各司其职，我的任务主要是熟悉代码、查询资料。

收获：大三啦！开学第一周是无线通信短距离实践，一开始的时候，我是有点懵的，因为通过这个名字我不知道这个实践做的是什么，后来老师在云班课里发了一些关于这个实践的一些相关资料，我的心才落下地，原来我们做的是这样子的。

实践第一天，老师让我们先安装好开发环境，我在安装开发环境的时候，其实自己是走了一些歪路的，安装开发环境，有时候我没有以管理员身份运行，所以这就导致后面出现了许许多多的一些问题，后面又重装了一次。第二天是做一些基础实验的调试，这天的内容还好，重要的是我们应该认真观看老师给的教程，那上面的实验步骤与现象写得很清楚，其实按着上面的教程来完成就没什么大的问题，我现在印象深刻的是老师问了我一个关于如何打开这个函数的问题，我一开始还去上面来找这个函数的程序，后面经过组员的提醒，才知道原来是可以通过右键来打开这个函数的程序的。

第三天是每一个人抽三个实验来完成，来改这三个实验中的参数，这是有困难的，不过困难是用来克服的。第四天和第五天是做自己组的实验项目和报告，最后这三天是十分具有挑战性的。任务越具有挑战性，那么完成这个任务之后就越具有收获。在我们组的全体努力下，我们共同完成了这个项目，我十分感谢老师、组长和组员们对我的帮助。

我的收获是：

1、其实许多事情看着简单，但当你动起手来的时候就不是这么简单了；在我做基础实验的时候，看着教程上的步骤很简单，并且代码也不是特别多，但当自己放到板子上运行的时候，还是会出现许许多多的问题；

2、完成一件事要究其原理，而不是追求其表面，表面是可以改变的，而一旦触及到真正的东西时，表面自然对于你来说就只有一种变化了。在完成自己抽的三个实验的时候，里面其实许许多多的东西，只要你看清代码之后也就变得不那么困难了。

2 实验九实践环节

# 2.1 考核要求

基础实验9联机模式下的单步进入

# 2.2 比较实验设计

该考核要求是让我们能够掌握单步进入如何来操作，该实验设计是在基础实验九上面，在电脑上连接好板子来实现单步执行的操作，首先我们应该要找到单步执行的按钮在哪，然后单步进入的按钮在哪，当单步执行到一些预定义函数的时候我们可以按下单步进入按钮来实现联机模式下的单步进入。

步骤：先连接ZB2530并且将代码下载到其中，点击主函数，按单步执行按钮，当在一个之前定义的函数中，我们可以点击单步进入按钮，从而进入到这个函数中去，并且可以看到他的运行。

原代码：

#include <ioCC2530.h>

#include <string.h>

#define uint unsigned int

#define uchar unsigned char

//定义控制灯的端口

#define LED1 P1\_0

#define LED2 P1\_1

#define LED3 P0\_4

//函数声明

void Delay(uint);

void initUARTSEND(void);

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

char Txdata[25]="HELLO! zigbee!\r\n";

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

延时函数

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

void Delay(uint n)

{

uint i;

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

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

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

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

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

}

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

串口初始化函数

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

void initUARTSEND(void)

{

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

PERCFG = 0x00; //位置1 P0口

P0SEL = 0x0c; //P0\_2,P0\_3用作串口

P2DIR &= ~0XC0; //P0优先作为UART0

U0CSR |= 0x80; //UART方式

U0GCR |= 8; //U0GCR.BAUD\_E

U0BAUD |= 59; //波特率设为9600 UxBAUD.BAUD\_M

UTX0IF = 0; //UART0 TX中断标志初始置位0

}

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

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 = 0;

}

}

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

主函数

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

void main(void)

{

uchar i;

P1DIR = 0x03; //P1控制LED

P0DIR |= 0x10; //P04定义为输出

initUARTSEND();

LED1 = 1; //关LED

LED3 = 1; //关LED

LED2 = 0;

UartTX\_Send\_String(Txdata,25);

for(i=0;i<30;i++)Txdata[i]=' ';

strcpy(Txdata,"HELLO! zigbee!\r\n"); //将UART0 TX test赋给Txdata;

while(1)

{

UartTX\_Send\_String(Txdata,strlen(Txdata)); //串口发送数据

Delay(50000); //延时

LED2=!LED2; //D2灯在闪，标志发送状态

Delay(50000);

Delay(50000);

}

}

# 2.3 原始实验效果

当按单步执行按钮的时候，程序会一直运行。

# 2.4 修改参数后的实验效果

这是修改参数后的实验效果，我们在这个地方按下了单步进入按钮，然后我们可以很清晰地看到这个函数的代码，并且我们可以看到程序在这个函数里面的代码是如何运行的。

# 2.5 本人在实验九实践环节中的收获

通过这次实验，我学会了单步执行，我学会了能够让程序在板子上单步执行，并且能够在单步执行的时候进入到该单步执行的程序。这对我们以后的程序设计是非常有帮助的，我们可以通过单步执行看我的程序哪里出了错误，也可以用单步进入看我的程序运行到了哪里。

3 实验八实践环节

# 3.1 考核要求

基础实验八采用32分频并且显示效果不变。

# 3.2 比较实验设计

原来的实验中使用的是128分频，我们这要改成32分频并且显示的效果不变。这里我的思路是：首先，我要在源程序中找到关于分频的代码，然后将该代码中分频的部分改成32分频。由于他的这里的分频改变，所以他的LED的显示效果也会改变，其中对于LED改变的是他间隔亮的时间，我们这里要计算改变分频之后，如何把时间改成128分频的，这里就要通过计算，最后通过计算我们算出的结果是9800左右，这样的话就可以把128分频改成32分频并且显示效果不变。

步骤：先找到该分频部分的代码，然后将分频数改成自己需要设定的数值。然后再计算闪0.5s所用的N为多少，再除以255就可以得到一个数值，将count循环中的245改成9800即可。

原代码：

#include <ioCC2530.h>

#include "lcd.h"

typedef unsigned char uchar;

typedef unsigned int uint;

#define LED1 P1\_0 // P1.0口控制LED1

uint count; //用于定时器计数

void InitLed(void)

{

P1DIR |= 0x01; //P1.0定义为输出

LED1 = 1; //使LED1灯上电默认为熄灭

}

void InitT3()

{

T3CTL |= 0x08 ; //开溢出中断

T3IE = 1; //开总中断和T3中断

T3CTL |= 0X50; //128分频,128/16000000\*N=0.5S,N=62500

T3CTL &= ~0x03; //自动重装 00－>0xff 62500/255=245(次)

T3CTL |= 0x10; //启动

EA = 1; //开总中断

}

void dispLED1State()

{

if(LED1>0)

{

//显示"D1:灭"

LCD\_P8x16Str(16, 5, "D1:");

LCD\_P16x16Ch(40, 5, 15);

}

else

{

//显示"D1:亮"

LCD\_P8x16Str(16, 5, "D1:");

LCD\_P16x16Ch(40, 5, 14);

}

}

//定时器T3中断处理函数

#pragma vector = T3\_VECTOR

\_\_interrupt void T3\_ISR(void)

{

IRCON = 0x00; //清中断标志, 也可由硬件自动完成

if(count++ > 245) //245次中断后LED取反，闪烁一轮（约为0.5 秒时间）

{ //经过示波器测量确保精确

count = 0; //计数清零

LED1 = ~LED1; //改变LED1的状态

dispLED1State();

}

}

void main(void)

{

InitLed(); //设置LED灯相应的IO口

InitT3(); //设置T3相应的寄存器

LCD\_Init();//LCD初始化

LCD\_CLS();//清屏

LCD\_welcome();

dispLED1State();

while(1)

{};

}

修改后的代码：

#include <ioCC2530.h>

#include "lcd.h"

typedef unsigned char uchar;

typedef unsigned int uint;

#define LED1 P1\_0 // P1.0口控制LED1

uint count; //用于定时器计数

void InitLed(void)

{

P1DIR |= 0x01; //P1.0定义为输出

LED1 = 1; //使LED1灯上电默认为熄灭

}

void InitT3()

{

T3CTL |= 0x08 ; //开溢出中断

T3IE = 1; //开总中断和T3中断

T3CTL |= 0X50; //128分频,128/16000000\*N=0.5S,N=62500

T3CTL &= ~0x03; //自动重装 00－>0xff 62500/255=245(次)

T3CTL |= 0x10; //启动

EA = 1; //开总中断

}

void dispLED1State()

{

if(LED1>0)

{

//显示"D1:灭"

LCD\_P8x16Str(16, 5, "D1:");

LCD\_P16x16Ch(40, 5, 15);

}

else

{

//显示"D1:亮"

LCD\_P8x16Str(16, 5, "D1:");

LCD\_P16x16Ch(40, 5, 14);

}

}

//定时器T3中断处理函数

#pragma vector = T3\_VECTOR

\_\_interrupt void T3\_ISR(void)

{

IRCON = 0x00; //清中断标志, 也可由硬件自动完成

if(count++ > 9800) //245次中断后LED取反，闪烁一轮（约为0.5 秒时间）

{ //经过示波器测量确保精确

count = 0; //计数清零

LED1 = ~LED1; //改变LED1的状态

dispLED1State();

}

}

void main(void)

{

InitLed(); //设置LED灯相应的IO口

InitT3(); //设置T3相应的寄存器

LCD\_Init();//LCD初始化

LCD\_CLS();//清屏

LCD\_welcome();

dispLED1State();

while(1)

{};

}

# 3.3 原始实验效果

当程序上是128分频的时候，其中的LED会每间隔0.5s闪一次，并且液晶显示屏上会显示灯的亮灭。

# 3.4 修改参数后的实验效果

将程序中的代码修改成32分频后，如果没改时间，就闪的特别快。

将其中的count的上限改了之后，灯也每间隔0.5s亮一次。

# 3.5 本人在实验九实践环节中的收获

在本次实验中，我的收获是我对ZigBee的分频有了研究，并且自己知道如何改变分频，分频对屏幕的显示是十分重要的，如果分频太低，这就会导致液晶屏幕上显示不出来什么东西，所以我对分频有了一定的掌握，这对我后面的编程设计有很大的帮助。

4 实验三实践环节

# 4.1 考核要求

基础实验三中去掉对LED2的控制

# 4.2 比较实验设计

原来的程序设计是对LED2有控制的，这里的考核要求是能够去掉对LED2的控制，这里去掉LED2的控制的思路是在程序上找到关于影响LED2的代码，由于是要去掉LED2的控制，所以我可以先试着将关于LEDD2相关的代码删去，然后看测试运行结果。

实验步骤：先找到控制LED2灯的代码，再逐渐调试，将LED2的代码删去，再逐渐调试，知道ZB2530上的灯LED2不再说S1的控制即可。

原代码：

#include <ioCC2530.h>

#define uint unsigned int

#define uchar unsigned char

//定义控制灯的端口

#define LED1 P1\_0 //定义LED1为P10口控制

#define LED2 P1\_1 //定义LED2为P11口控制

#define LED3 P0\_4 //定义LED3为P04口控制

#define KEY1 P0\_1 //定义按键S1为P01口控制

//函数声明

void Delay(uint); //延时函数声明

void Initial(void); //初始化函数声明

void InitKey(void); //初始化按键函数声明

uchar KeyScan(void); //按键扫描函数声明

uchar Keyvalue = 0 ; //定义变量记录按键动作

uint KeyTouchtimes = 0 ; //定义变量记录按键次数

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

//延时

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

void Delay(uint n)

{

uint i;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}

void InitKeyINT(void)

{

P0INP |= 0x02; //上拉

P0IEN |= 0X02; //P01设置为中断方式

PICTL |= 0X01; //下降沿触发

EA = 1;

IEN1 |= 0X20; // P0设置为中断方式;

P0IFG |= 0x00; //初始化中断标志位

}

void InitIO(void)

{

P1DIR |= 0x03; //P10、P11定义为输出

P0DIR |= 0x10; //P04定义为输出

LED1 = 1;

LED2 = 1;

LED3 = 1; //LED灯初始化为灭

}

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

//中断处理函数

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

#pragma vector = P0INT\_VECTOR

\_\_interrupt void P0\_ISR(void)

{

if(P0IFG>0) //按键中断

{

P0IFG = 0;

Delay(100);

if(P0IFG==0) //按键中断

{

Delay(100);

KeyTouchtimes = KeyTouchtimes+1; //每次中断发生时记录按键次数加1

}

}

P0IF = 0; //清中断标志

}

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

//主函数

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

void main(void)

{

InitIO();

InitKeyINT(); //调用初始化函数

while(1)

{

if(KeyTouchtimes == 1) //按键设置为LED3,LED2,LED1倒序流水闪烁

{

LED3 = !LED3;

Delay(30000);

LED2 = !LED2;

Delay(30000);

LED1 = !LED1;

Delay(30000);

KeyTouchtimes = 0;

}

}

}

修改后的代码：

#include <ioCC2530.h>

#define uint unsigned int

#define uchar unsigned char

//定义控制灯的端口

#define LED1 P1\_0 //定义LED1为P10口控制

#define LED3 P0\_4 //定义LED3为P04口控制

#define KEY1 P0\_1 //定义按键S1为P01口控制

//函数声明

void Delay(uint); //延时函数声明

void Initial(void); //初始化函数声明

void InitKey(void); //初始化按键函数声明

uchar KeyScan(void); //按键扫描函数声明

uchar Keyvalue = 0 ; //定义变量记录按键动作

uint KeyTouchtimes = 0 ; //定义变量记录按键次数

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

//延时

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

void Delay(uint n)

{

uint i;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}

void InitKeyINT(void)

{

P0INP |= 0x02; //上拉

P0IEN |= 0X02; //P01设置为中断方式

PICTL |= 0X01; //下降沿触发

EA = 1;

IEN1 |= 0X20; // P0设置为中断方式;

P0IFG |= 0x00; //初始化中断标志位

}

void InitIO(void)

{

P1DIR |= 0x03; //P10、P11定义为输出

P0DIR |= 0x10; //P04定义为输出

LED1 = 1;

LED3 = 1; //LED灯初始化为灭

}

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

//中断处理函数

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

#pragma vector = P0INT\_VECTOR

\_\_interrupt void P0\_ISR(void)

{

if(P0IFG>0) //按键中断

{

P0IFG = 0;

Delay(100);

if(P0IFG==0) //按键中断

{

Delay(100);

KeyTouchtimes = KeyTouchtimes+1; //每次中断发生时记录按键次数加1

}

}

P0IF = 0; //清中断标志

}

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

//主函数

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

void main(void)

{

InitIO();

InitKeyINT(); //调用初始化函数

while(1)

{

if(KeyTouchtimes == 1) //按键设置为LED3,LED2,LED1倒序流水闪烁

{

LED3 = !LED3;

Delay(30000);

LED1 = !LED1;

Delay(30000);

KeyTouchtimes = 0;

}

}

}

# 4.3 原始实验效果

原始的现象是按下S1按键后，LED1和LED2和LED3都会亮，再按下S1按键之后，LED1和LED2和LED3会再次熄灭。

# 4.4 修改参数后的实验效果

这是修改后的现象，我们可以看到当我们按下S1的时候，LED2不会再像LED1和LED3那样受S1的控制。出现的现象是按下S1按键后，LED1和LED3都会亮，再按下S1按键之后，LED1和LED3会再次熄灭，而其中LED2的现象是一直熄灭，并没有收到S1按键的控制。

# 4.5 本人在实验三实践环节中的收获

本次实验的要求是去掉对LED2的控制，想要达到这个目的，我们需要把代码读懂，并且找到关于控制LED2的部分，在寻找代码中控制LED2的过程中，我提高自己读代码的能力，如果是长代码的话，可以先看他的主函数，主函数里面有一些自定义函数我们可以到前面去寻找并且读懂他的含义。

第二个收获是知道做成什么事情应该要慢慢地来，一朝一夕是完不成的，在完成某项任务的时候，我们应该先慢慢地调试，其中肯定会出现许许多多的错误，但先不用着急，慢慢地调试慢慢地改正边行了。

**5 建议**

在这次无线短距离通信中实践中，老师先带我们安装开发环境和让我们完成基础实验来打下基础，然后再个人做自己抽到的三个实验，根据相关要求来修改实验中的某些参数，在这个过程中，我们运用ZigBee的能力也得到了很大的提升，最后再做自己小组的项目，我觉得这个实践过程的教学是层次分明的，难度由低到高的，课程上的安排是十分合理的。

我的建议是在本次实验中的ZB2530的设备有点少了，我觉得俩个人一套ZB2530最好，感觉四个人一套ZB2530设备太少了，因为在调式代码的过程中是一直需要ZB2530的。

最后感谢老师和组长和组员们对我的帮助。