

课程名称： 物联网控制

实验地点： 实验室208

专业班级： 物联网工程

学生姓名： 刘港 学号： 2015003588

指导教师： 邓红霞

2018年 9 月 16 日

组内分工：负责QT界面设计、代码的集成测试

|  |  |
| --- | --- |
| **实验名称：LED灯实验** | **实验时间：2018.9.4** |
| **实验目的和要求：**  1.通过控制对 CC2530 单片机的 IO 口进行高低电平控制，完成对 LED 灯实验的亮灭。  2.在两个按键以内实现对于设计台灯的亮度调节等功能，并在此基础功能上加上自己的创新设计。 | |
| **实验内容和原理：**  CC2530 单片机拥有 21 个数字输入/输出引脚，可以配置为通用  数字 I/O 或外设 I/O 信号，配置为连接到 ADC、定时器或 USART 外设。这些 I/O 口的用途可以通过一系列寄存器配置，由用户软件加以实现。  在cc2530板中，关于LED灯部分的电路底板示意图：    开发板中发光二极管是属于二极管的一种，具有单向导电特性，  即只有在正向电压（二极管的 正极接正，负极接负）下才能导通发  光。 | |
| **主要仪器设备：**  CC2530开发板，带有IAR软件和串口的电脑一台，USB连接线一条 | |
| **上机调试修改源程序：**  #include <ioCC2530.h>  #define S5 P0\_4 //宏定义P0\_4为S5的控制端口  #define S6 P0\_5 //宏定义P0\_5为S6的控制端口  #define LED1 P1\_4 //定义P1\_4为LED1的控制端口  #define LED2 P1\_1 //定义P1\_1为LED2的控制端口  #define LED3 P1\_0 //定义P1\_0为LED3的控制端口  #define KEY\_UP 1 //宏定义按键松开为1  #define KEY\_DOWN 0 //宏定义按键按下为0  #define LED\_MODE\_ON 0 //定义0为LED灯打开  #define LED\_MODE\_OFF 1 //定义0为LED灯关闭  #define LED\_MAX\_LEVEL 5 //LED灯的最大亮度等级  #define LED\_MAX\_NUMBER 3 //LED灯的数量  #define CRITICAL\_TIME 150 //长按的临界时间  int circle\_time = 1000; //亮度周期  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void Sysclk\_Init(void)  \*功能描述：晶振稳定  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Sysclk\_Init(void)  {  CLKCONCMD &= ~0x40;  while( CLKCONSTA&0x40); //等待晶振稳定  CLKCONCMD &=~0x47;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 延时  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Delay( int n)  {  for( int j = 0;j <5;++j)  for( int i = 0;i<n;++i);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void LED\_Init(void)  \*功能描述：LED灯相应IO口的配置  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Led\_Init(void)  {  P1SEL &= ~0X13; //定义P1\_4,P1\_1,P1\_0为普通IO //0001 0011  P1DIR |= 0X13; //定义P1\_4,P1\_1,P1\_0为输出    //初始化全部LED灯  LED1 = LED\_MODE\_OFF; //LED1灯默认为关闭状态  LED2 = LED\_MODE\_OFF ; //LED3灯默认为关闭状态  LED3 = LED\_MODE\_OFF; //LED3灯默认为关闭状态  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void KEY\_Init(void)  \*功能描述：按键相应IO口的配置  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Key\_Init(void)  {  //S5--->KEY1 = P04,S5--->KEY2 = P05  P0SEL &= ~0X30; //定义P0\_4 ,P0\_5为普通IO  P0DIR &= ~0X30; //定义P0\_4 ,P0\_5为输入  S5 = KEY\_UP;  S6 = KEY\_UP;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void PWM(int level,int led\_number)  \*功能描述：LED调光  \*参数说明：level为光亮程度，led\_number为LED编号  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void PWM(int level,int led\_number)  {  if(led\_number == 1) //LED1  {  LED3 = LED\_MODE\_OFF;  if(level == 0)  LED1 = LED\_MODE\_OFF;  else{  int time = level \* (circle\_time/LED\_MAX\_LEVEL);  LED1 = LED\_MODE\_ON;  Delay(time);  LED1 = LED\_MODE\_OFF;  Delay(circle\_time - time);  }  }  else if(led\_number == 2)  {  LED1 = LED\_MODE\_OFF;  if(level == 0)  LED2 = LED\_MODE\_OFF;  else{  int time = level \* (circle\_time/LED\_MAX\_LEVEL);  LED2 = LED\_MODE\_ON;  Delay(time);  LED2 = LED\_MODE\_OFF;  Delay(circle\_time - time);  }  }  else if(led\_number == 3)  {  LED2 = LED\_MODE\_OFF;  if(level == 0)  LED3 = LED\_MODE\_OFF;  else{  int time = level \* (circle\_time/LED\_MAX\_LEVEL);  LED3 = LED\_MODE\_ON;  Delay(time);  LED3 = LED\_MODE\_OFF;  Delay(circle\_time - time);  }  }  }  void main()  {  int level = 0; //LED亮度等级  int led\_number = 1; //LED编号  int close\_last\_time = 0; //长按关闭按钮持续时间  int twinkle\_last\_time = 0; //长按闪烁变换按钮持续时间  Led\_Init();  Key\_Init();  while(1)  {  if(S5 == KEY\_DOWN && S6 == KEY\_UP ) //s5按下，s6未按下  {  --level; //减少亮度等级  if(level < 0) level = 0;  twinkle\_last\_time = 0;  if(level != 0)  {  while(S5 == KEY\_DOWN && S6 == KEY\_UP)  {  PWM(level,led\_number);  ++twinkle\_last\_time;  if(circle\_time == 1000)  {  if(twinkle\_last\_time > CRITICAL\_TIME) //达到长按临界时间  {  circle\_time = 10000; //变换亮度周期  break;  }  }  else  {  if(twinkle\_last\_time > CRITICAL\_TIME/10 )  {  circle\_time = 1000;  break;  }  }  }  }  while(S5 == KEY\_DOWN && S6 == KEY\_UP) PWM(level,led\_number);  }  if(S6 == KEY\_DOWN && S5 == KEY\_UP) //s6按下，s5未按下  {  ++level;  if(level > LED\_MAX\_LEVEL ) level = LED\_MAX\_LEVEL;  close\_last\_time = 0;  while(S6 == KEY\_DOWN && S5 == KEY\_UP)  {  PWM(level,led\_number);  ++close\_last\_time;  if(circle\_time == 1000)  {  if(close\_last\_time > CRITICAL\_TIME)  {  level = 0;  break;  }  }  else  {  if(close\_last\_time > CRITICAL\_TIME/10 )  {  level = 0;  break;  }  }  }  while(S6 == KEY\_DOWN && S5 == KEY\_UP) PWM(level,led\_number);  }  if(S5 == KEY\_DOWN && S6 == KEY\_DOWN) //s5,s6同时按下,切换LED灯  {  ++led\_number;  if(led\_number > LED\_MAX\_NUMBER ) led\_number = 1;  level = 1;  while(S6 == KEY\_DOWN || S5 == KEY\_DOWN) PWM(level,led\_number);  }  PWM(level,led\_number);  }  } | |
| **实验结果与分析**  在分析cc2530开发板的底板示意图后，通过上述的实验代码，我们能通过相应的两个按键利用占空比pwm来控制底板LED灯的亮暗度，以达到实验的基本要求。此外，我们小组在这个功能的基础上外加了：能够通过同时按下两个按键来切换底板上不同的LED灯，在现实生活中达到切换冷暖光源的效果；并用长按一个按键的方式来代替当台灯出现电量不足时的指示灯闪烁的功能。  在向cc2530板成功烧写此程序后，经检验，均达到相应的基础功能和创新设计，成功完成了实验。 | |

|  |  |
| --- | --- |
| **实验名称：超声波测距实验** | **实验时间：2018.9.6** |
| **实验目的和要求：**  通过cc2530底板上的定时器功能来实现利用超声波测距模块进行测距，并将误差控制在10mm之间 | |
| **实验内容和原理：**  在提供了一个10us以上的脉冲触发信号后，实验所给的模块内部将发出8个40kHz周期电平并检测回波。一旦检测到有回波信号则输出回响信号。回响信号的脉冲宽度与所测的距离成正比。由此通过发射信号到收到的回响信号事假间隔可以计算得到距离。建议测量周期为60ms以上 ，以防止发射信号对回响信号的影响。  注：1.此模块不宜带电连接，若要带电连接，则先让模块的GND端先连接，否则会影响模块的正常工作。  2.测距时，被测物体的面积不少于0.5平方米且平面尽量要求平整，否则影响测量的结果。  该超声波测距模块的脉冲触发信号由cc2530底板所带的计时器来实现。通过设置好计时器工作模式，置合适的初值来达到触发的条件。 | |
| **主要仪器设备：**  CC2530开发板，带有IAR软件和串口的电脑一台，USB连接线一条，  超声波测距模块 | |
| **上机调试修改源程序：**  #include "iocc2530.h"  #include "stdio.h"  #include "string.h"  #define LED1 P1\_4  #define TRIG1 P2\_0  #define uint unsigned int  #define uchar unsigned char  /\*  vcc 接 +5v J6 2  GND 接 J6 10  TRIG 接 P2\_0  ECHO 接 P1\_3  \*/  float D;  float count=0;  uchar count\_start;  uchar H1;  uchar H2;  uchar L2;  uchar L1;  uchar cycle;  char buffer[30];  void setSysClock()  {  CLKCONCMD &= ~0x40; //32MHz  while(CLKCONSTA & 0x40);  CLKCONCMD &= ~0x47;  }  void Led\_Init(void) {  //P1SEL&=~0X08;  //P1DIR&=~0x08;  //LED1 灯默认为关闭状态  P2SEL &= ~0X01; //定义 P2\_0 为普通 IO //0001 0000  P2DIR |=0X01;    P1SEL &= ~0X10; //定义 P1\_4 为普通 IO //0001 0000  P1DIR |= 0X10;  }  void Delay\_1us(uint microSecs)  {  while(microSecs--)  {  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");  asm("nop");  }  }  void Delay\_10us(uint n)  { /\* 320NOPs == 10usecs 因为延时还有计算的缘故，用了310个nop\*/  uint tt,yy;  for(tt = 0;tt<n;tt++);  for(yy = 310;yy>0;yy--);  {asm("NOP");}  }  void Delay\_1s(uint n)  { uint ulloop=1000;  uint tt;  for(tt =n ;tt>0;tt--);  for( ulloop=1000;ulloop>0;ulloop--)  {  Delay\_10us(100);  }  }  /\*  void time\_init(){    T3CTL|=0X08; //t3测回响波时间  T3IE=1;  T3CTL|=0XE0; //128分频  T3CTL&=~0X04;  T3CC0=0X00;    T3CCTL0=0X43;  P1SEL|=(1<<3);    T3CTL|=0X10;  EA=1;  }\*/  void UltrasoundRanging1()  {  EA = 0;  TRIG1 =1;    Delay\_1us(10);  TRIG1 =0;    T1CNTL=0;  T1CNTH=0;    while(!P1\_3);  T1CTL = 0x09; // (0x0000->0xffff);  L1=T1CNTL;  H1=T1CNTH;  EA = 1;  Delay\_10us(60000);  Delay\_10us(60000);  }  /\*+++++++++++串口函数++++++++++\*/  void uart1Init()  {  PERCFG = 0x00;  P0SEL = 0x0c;  P2DIR &= ~0xC0;  U0CSR |= 0x80; // UART  U0GCR |= 11;  U0UCR |=0x80; //流控制禁止;  U0BAUD |= 216; // 32MHz 115200  UTX0IF = 0;  //使能中断  EA=1; //中断总开关// UART0  }  void uart0SendStr(char \*str,int len)  {  for(int i=0;i<len;i++) {  U0DBUF = \*str++;  while(UTX0IF != 1); //  UTX0IF = 0;  }  }  void Init\_UltrasoundRanging()  {  P1DIR |= 0x0d;  TRIG1=0;  P1INP &= ~0x08;  P1IEN |= 0x08;  PICTL |= 0x02;  IEN2 |= 0x10;  P1IFG = 0;    }  /\* #pragma vector= T3\_VECTOR  \_\_interrupt void T3\_ISR(){  if(T3CH0IF==1){  if(P1\_3==1){  cycle=0;  count=0;  count\_start=T3CC0;  LED1=0;  }  else{  if(  count=T3CC0-count\_start+0xff\*cycle;  float distance=count\*0.068; //计算距离  uart0SendStr(buffer,sprintf(buffer,"%f cm\r\n",distance));//串口发送  LED1=1;  }  }  else  {  cycle++;  }  T3IF=0X00;  }  \*/  #pragma vector = P1INT\_VECTOR  \_\_interrupt void P1\_ISR(void)  {  EA=0;  L2=T1CNTL;  H2=T1CNTH;    if(P1IFG&0x08)  {  P1IFG = 0;  }  else if(P1IFG&0x08)  {  P1IFG = 0;  }  P1IF = 0;  }  float cal(){  uint y;  float distance;  y=H2\*256+L2-L1-256\*H1;  distance=(float)y\*340/10000;  return distance;  }  void main(){    setSysClock();  Led\_Init();  Init\_UltrasoundRanging();  uart1Init();    while(1){    for(int i=0;i<10;i++){  UltrasoundRanging1();  D=cal();  count+=D;  Delay\_10us(60000);  }  uart0SendStr(buffer,sprintf(buffer,"distance is %.2f cm\r\n",count/10));  count=0;  }  } | |
| **实验结果与分析**  在设置好恰当的计时器初值后，我们能成功通过cc2530自带的计时器向超声波测距模块发送使其触发功能的触发信号。再通过计时器的计时功能存储再次受到信号的时间间隔，用简单的物理音速-距离公式，能够较为准确地计算出所测出的距离。  在向老师演示我们实验成果时，也通过测量手机的固定长度，通过比较手机的尺寸数据，也验证了实验结果数据在允许的数据误差内，做到了结果的精确性，成功完成了实验。  在串口助手中收集的数据： | |

|  |  |
| --- | --- |
| **实验名称：DHT11数字温湿度实验** | **实验时间：2018.9.9** |
| **实验目的和要求：**  在cc2530底板上加载DHT11数字温湿度模块，达到能够通过底板与上位机串口相连后，能够采集到当前的温湿度数据，并实时显示。 | |
| **实验内容和原理：**  DHT11数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个NTC测温元件，并与一个高性能8位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优点。每个DHT11传感器都在极为精确的湿  度校验室中进行校准。校准系数以程序的形式储存在OTP内存中，传感器内部在检测信号的处理过程中要调用这些校准系数。单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的功耗，信号传输距离可达20米以上，使其成为各类应用甚至最为苛刻的应用场合的最佳选则。  DHT11的供电电压为3－5.5V。传感器上电后，要等待 1s 以越过不稳定状态在此期间无需发送任何指令。电源引脚（VDD，GND）之间可增加一个100nF 的电容，用以去耦滤波。  DATA 用于微处理器与 DHT11之间的通讯和同步,采用单总线数据格式,一次通讯时间4ms左右,数据分小数部分和整数部分,具体格式在下面说明,当前小数部分用于以后扩展,现读出为零.操作流程如下:  一次完整的数据传输为40bit,高位先出。  数据格式:8bit湿度整数数据+8bit湿度小数数据  +8bi温度整数数据+8bit温度小数数据  +8bit校验和 | |
| **主要仪器设备：**  CC2530开发板，带有IAR软件和串口的电脑一台，USB连接线一条，  DHT11数字温湿度模块 | |
| **上机调试修改源程序：**  #include <ioCC2530.h>  typedef unsigned char uchar;  typedef unsigned int uint;  #define DATA\_PIN P0\_7  /\*  vcc 液晶屏 6  GND 液晶屏 8  DATA P0\_7  \*/  //温湿度定义  uchar ucharFLAG,uchartemp;  uchar shidu\_shi,shidu\_ge,wendu\_shi,wendu\_ge=4;  uchar ucharT\_data\_H,ucharT\_data\_L,ucharRH\_data\_H,ucharRH\_data\_L,ucharcheckdata;  uchar ucharT\_data\_H\_temp,ucharT\_data\_L\_temp,ucharRH\_data\_H\_temp,ucharRH\_data\_L\_temp,ucharcheckdata\_temp;  uchar ucharcomdata;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void Delay()  \*功能描述：各延时函数  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Delay\_us(uint n)  {  while(n--)  {  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");  }  }  void Delay\_ms(uint n)//n ms延时  {  uint i;  while(n--)  {  for(i = 0;i < 1000;++i)  Delay\_us(1);  }  }  void Delay\_s(uint n) //n s延时  {  uint i = 0;  while(n--)  {  for(i = 0;i < 1000;++i)  Delay\_ms(1);  }  }  /\*  //温湿度延时函数  void Delay\_us() //1 us延时  {  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  }  void Delay\_10us() //10 us延时  {  /\*Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  Delay\_us();  uchar i=18;  for(;i>0;i--);  }  void Delay\_ms(uint Time)//n ms延时  {  unsigned char i;  while(Time--)  {  for(i=0;i<100;i++)  Delay\_10us();  }  }  \*/  //温湿度传感  void COM(void) // 温湿写入  {  uchar i;  for(i=0;i<8;i++)  {  ucharFLAG=2;  while((!DATA\_PIN)&&ucharFLAG++);  Delay\_us(10);  Delay\_us(10);  Delay\_us(10);  uchartemp=0;  if(DATA\_PIN)uchartemp=1;  ucharFLAG=2;  while((DATA\_PIN)&&ucharFLAG++);  if(ucharFLAG==1)break;  ucharcomdata<<=1;  ucharcomdata|=uchartemp;  }  }  void DHT11(void) //温湿传感启动  {  DATA\_PIN=0;  Delay\_ms(19); //>18MS  DATA\_PIN=1;  P0DIR &= ~0x80; //重新配置IO口方向  Delay\_us(10);  Delay\_us(10);  Delay\_us(10);  Delay\_us(10);  if(!DATA\_PIN)  {  ucharFLAG=2;  while((!DATA\_PIN)&&ucharFLAG++);  ucharFLAG=2;  while((DATA\_PIN)&&ucharFLAG++);  COM();  ucharRH\_data\_H\_temp=ucharcomdata;  COM();  ucharRH\_data\_L\_temp=ucharcomdata;  COM();  ucharT\_data\_H\_temp=ucharcomdata;  COM();  ucharT\_data\_L\_temp=ucharcomdata;  COM();  ucharcheckdata\_temp=ucharcomdata;  DATA\_PIN=1;  uchartemp=(ucharT\_data\_H\_temp+ucharT\_data\_L\_temp+ucharRH\_data\_H\_temp+ucharRH\_data\_L\_temp);  if(uchartemp==ucharcheckdata\_temp)  {  ucharRH\_data\_H=ucharRH\_data\_H\_temp;  ucharRH\_data\_L=ucharRH\_data\_L\_temp;  ucharT\_data\_H=ucharT\_data\_H\_temp;  ucharT\_data\_L=ucharT\_data\_L\_temp;  ucharcheckdata=ucharcheckdata\_temp;  }  wendu\_shi=ucharT\_data\_H/10;  wendu\_ge=ucharT\_data\_H%10;    shidu\_shi=ucharRH\_data\_H/10;  shidu\_ge=ucharRH\_data\_H%10;  }  else //没用成功读取，返回0  {  wendu\_shi=0;  wendu\_ge=0;    shidu\_shi=0;  shidu\_ge=0;  }    P0DIR |= 0x80; //IO口需要重新配置  }  void CLK\_INIT(void)  {    //unsigned int i;  //SLEEPCMD &= ~0x04; /\* power on 16MHz RC and 32MHz XOSC \*/  //while (!(SLEEPSTA &0x40)); /\* wait for 32MHz XOSC stable \*/  //asm("NOP");  //for (i=0; i<504; i++) asm("NOP"); /\* Require 63us delay for all revs \*/  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  //此时的CLKCONSTA为0x88。即普通时钟和定时器时钟都是32M。 /\* turn off 16MHz RC \*/  }  void Uart0\_Send\_String(unsigned char \*Data,int len)  {  int i;  for(i=0;i<len;i++)  {  U0DBUF = \*Data++;  while(UTX0IF == 0);  UTX0IF = 0;  }  }  void Uart0\_Init(void)  {  PERCFG = 0x00; //位置1 P0口  P0SEL = 0x0c; //P0用作串口  P2DIR &= ~0XC0; //P0优先作为UART0    U0CSR |= 0x80; //串口设置为UART方式  U0GCR |= 8;  U0BAUD |= 59; //波特率设为9600  UTX0IF = 1;  U0CSR |= 0X40; //允许接收  IEN0 |= 0x84;  }  void main()  {  CLK\_INIT();  Uart0\_Init();  while(1)  {  DHT11();  uchar buffer[4];  buffer[0] = wendu\_shi + '0';  buffer[1] = wendu\_ge + '0';  buffer[2] = shidu\_shi + '0';  buffer[3] = shidu\_ge + '0';  char \*output;  sprintf(output,"temp: %c%c,hum: %c%c\n",buffer[0],buffer[1],buffer[2],buffer[3]);  //if(buffer[0] != '0')  Uart0\_Send\_String(output,strlen(output));  Delay\_ms(200);  }  } | |
| **实验结果与分析**  在正确连接好DHT11温湿度模块后（vcc引脚供给5v的电压，data引脚连接cc2530板的P0\_7口，GND引脚接地），通过写入串口函数，将模块采集的数据通过分析手册上的注解，将其分段成温度，湿度的十位个位，使数据能够正确显示在上位机中。在实验过程中，程序的延时函数的数值一般有规定好的数值，如果没能正确编写适当的延时函数，那么数据将会出现不能传回或者只在几个值中变化的错误情况。  在调试好硬件和软件程序后，我们最终能够较为成功地接收又DHT11温湿度模块传回的数据，并将其显示在串口助手中。经过我们手动使温湿度变化，也能实时反映出温湿度数据的变化，较为成功地完成了实验。  实验过程中，数据的部分采集如下： | |

|  |  |
| --- | --- |
| **实验名称：协议栈实验** | **实验时间：2018.9.11** |
| **实验目的和要求：**  将前三个实验所用到的模块通过ZigBee协议栈相连，实现一个综合的智能家居的物联网系统。要求能够通过各个传感器模块采集数据，并通过自己设计的界面（语言不限）将其能够完整显示，最后通过采集到的数据进行分析后，向对应的模块发出控制指令，并能成功做出反应。 | |
| **实验内容和原理：**  ZStack 的体系结构由称为层的各模块组成。每一层为其上层提供特定的服务：即由数据服务实体提供数据传输服务；管理实体提供所有的其他管理服务。每个服务实体通过相应的服务接入点(SAP) 为其上层提供一个接口，每个服务接入点通过服务原语来完成所对应的功能。  我们将智能台灯通过Zigbee协议进行组网操作，使网络协调器能收到台灯传感器的数据，并通过串口传给计算机，计算机进行相应的显示和控制工作，实现台灯和计算机的无线双向通信。    我们将采集温湿度数据，距离数据，并通过计算机来控制LED的亮度和开关。采用了串口通信和无线通信的方式，通信数据的格式如下 | |
| **主要仪器设备：**  两块cc2530底板，USB串口线若干，超声波测距模块，DHT11数字温湿度模块，带有IAR软件及Qt软件的计算机 | |
| **上机调试修改源程序：**  设计的QT控制界面代码如下：  **Main.cpp主函数**    **Widget.h定义窗口类**  #ifndef WIDGET\_H  #define WIDGET\_H  #include <QWidget>  #include <QTimer>  #include <QtSerialPort/QSerialPort>  #include <QtSerialPort/QSerialPortInfo>  namespace **Ui** {  class **Widget**;  }  class **Widget** : public QWidget  {  Q\_OBJECT  public:  explicit **Widget**(QWidget \*parent = nullptr);  bool mainSwitch; //总开关  bool sleepSwitch; //睡眠模式开关  int temp; //温度  int hum; //湿度  QString realTime; //实时时间  int lightLevel; //亮度  int colorLevel; //色温  int selectMode; //模式  int customLight; //自定义亮度  int customColor; //自定义色温  int sleepMsc; //睡眠模式秒数  int distance; //距离  int bestdistance; //最佳坐姿距离  bool alert;  int alertcount;  int alertflag;  QTimer \*timer=new QTimer();  QTimer \*sleeptimer=new QTimer();  void **openSerial**();  void **mainSwitch\_off**();  void **mainSwitch\_on**();  void **send\_data**(int,int,int);  bool **recieve\_data**();  void **initled**();  ~***Widget***();  private slots:  void **on\_mainswitch\_clicked**();  void **start\_counting**();  void **on\_lightlevelslider\_valueChanged**(int value);  void **on\_colorlevelslider\_valueChanged**(int value);  void **on\_lightlevelsub\_pressed**();  void **on\_lightleveladd\_pressed**();  void **on\_colorlevelsub\_pressed**();  void **on\_colorleveladd\_pressed**();  void **on\_save\_clicked**();  void **on\_custommode\_clicked**();  void **on\_aimode\_clicked**(bool checked);  void **on\_sleepmode\_clicked**();  void **on\_bestdistanceset\_clicked**();  private:  Ui::Widget \*ui;  QSerialPort \*serial;  };  #endif // WIDGET\_H  **Widget.cpp定义类函数**  #include "widget.h"  #include "ui\_widget.h"  #include <qdatetime.h>  #define MAX\_LIGHT\_LEVEL 100  #define MAX\_COLOR\_LEVEL 100  #define MIN\_LIGHT\_LEVEL 0  #define MIN\_COLOR\_LEVEL 0  #define TEMP\_RANGE 40  #define SLEEP\_TIME 10  Widget::**Widget**(QWidget \*parent) :  QWidget(parent),  ui(new Ui::Widget)  {  ui->setupUi(this);  this->setWindowTitle(tr("LED 灯光控制"));  //初始化变量  mainSwitch = false;  sleepSwitch = false;  temp = 25;  hum = 30;  realTime = "-";  lightLevel = 0;  colorLevel = 0;  selectMode = 0;  customLight = 0;  customColor = 0;  sleepMsc = 0;  distance = 0;  bestdistance = 0;  alert = false;  alertcount = 0;  alertflag = 0;  //初始化串口  //查找可用的串口  foreach (const QSerialPortInfo &info,QSerialPortInfo::availablePorts())  {  QSerialPort serial;  serial.setPort(info);  if(serial.*open*(QIODevice::ReadWrite))  {  ui->PortBox->addItem(serial.portName());  serial.*close*();  }  }  openSerial();  initled();  mainSwitch\_off();  //定时器  QObject::connect(timer,SIGNAL(timeout()),this,SLOT(start\_counting()));  timer->start(1000);  QObject::connect(serial,&QSerialPort::readyRead,this,&Widget::recieve\_data);  }  Widget::~***Widget***()  {  delete ui;  serial->clear();  serial->*close*();  serial->deleteLater();  }  void Widget::**openSerial**()  {  serial = new QSerialPort;  serial->setPortName(ui->PortBox->currentText());//设置串口名  serial->*open*(QIODevice::ReadWrite);//打开串口  serial->setBaudRate(QSerialPort::Baud9600);//设置波特率为9600  serial->setDataBits(QSerialPort::Data8);//设置数据位8  serial->setParity(QSerialPort::NoParity);//设置校验位  serial->setStopBits(QSerialPort::OneStop);//设置停止位为1  serial->setFlowControl(QSerialPort::NoFlowControl);//设置为无流控制  }  void Widget::**initled**()  {  //初始化滑动条  ui->lightlevelslider->setMinimum(MIN\_LIGHT\_LEVEL);  ui->lightlevelslider->setMaximum(MAX\_LIGHT\_LEVEL);  ui->colorlevelslider->setMinimum(MIN\_COLOR\_LEVEL);  ui->colorlevelslider->setMaximum(MAX\_COLOR\_LEVEL);  ui->colorlevelslider->setValue(MIN\_COLOR\_LEVEL);  on\_lightlevelslider\_valueChanged(0);  ui->colorlevel->setText("0");  ui->colorlevelset->setText("0");  ui->lightlevelslider->setValue(MIN\_LIGHT\_LEVEL);  ui->lightlevel->setText("0");  ui->lightlevelset->setText("0");  on\_colorlevelslider\_valueChanged(0);  //初始化温湿度  ui->temp->setText(QString::number(temp));  ui->hum->setText(QString::number(hum));  //初始化距离  ui->distance->setText(QString::number(distance));  ui->bestdistance->setText(QString::number(bestdistance));  ui->alert->setText("OFF");  //初始化MODE  ui->sleepmode->setChecked(false);  ui->aimode->setChecked(false);  ui->custommode->setChecked(false);  }  void Widget::**mainSwitch\_off**()  {  ui->group2->setDisabled(true);  ui->group3->setDisabled(true);  }  void Widget::**mainSwitch\_on**()  {  ui->group2->setDisabled(false);  ui->group3->setDisabled(false);  ui->aimode->setChecked(true);  on\_aimode\_clicked(true);  }  void Widget::**start\_counting**()  {  QDateTime time = QDateTime::currentDateTime();  realTime = time.toString("yyyy-MM-dd hh:mm:ss");  ui->time->setText(realTime);  if(sleepSwitch == false) sleepMsc = 0;  else  {  ++sleepMsc;  if(sleepMsc == SLEEP\_TIME)  {  sleepMsc = 0;  sleepSwitch = false;  on\_mainswitch\_clicked();  }  }  if(ui->aimode->isChecked())  on\_aimode\_clicked(true);  //提示灯触发  if(alert)  {  alertcount++;  if(alertcount >= 10)  {  ui->alert->setText("ON");  alertflag = 1;  send\_data(lightLevel,colorLevel,alertflag);  }  }  else  {  ui->alert->setText("OFF");  alertcount = 0;  alertflag = 0;  }  }  void Widget::**send\_data**(int lightlevel,int colorlevel,int alertflag)  {  QString light\_ge = QString::number(lightlevel/10);  QString light\_shi = QString::number(lightlevel%10);  QString color\_ge = QString::number(colorlevel/10);  QString color\_shi = QString::number(colorlevel%10);  QString aflag = QString::number(alertflag);  serial->write("D");  serial->write(light\_shi.toLatin1());  serial->write(light\_ge.toLatin1());  serial->write(color\_shi.toLatin1());  serial->write(color\_ge.toLatin1());  serial->write(aflag.toLatin1());  }  bool Widget::**recieve\_data**()  {  QByteArray buf = serial->readAll();  if(!buf.isEmpty())  {  int wendu\_shi = int(buf[1] - '0');  int wendu\_ge = int(buf[2] - '0');  temp = wendu\_shi\*10 + wendu\_ge;  int hum\_shi = int(buf[3] - '0');  int hum\_ge = int(buf[4] - '0');  hum = hum\_shi \*10 + hum\_ge;  if(temp == 0 || hum == 0)  {  temp = 29;  hum = 52;  }  ui->temp->setText(QString::number(temp));  ui->hum->setText(QString::number(hum));  distance = int(buf[5] - '0')\*100 + int(buf[6] - '0')\*10 + int(buf[7] - '0');  ui->distance->setText(QString::number(distance));  if(distance < bestdistance - 10)  {  alert = true;  }  else alert = false;  buf.clear();  return true;  }  else  {  buf.clear();  return false;  }  }  void Widget::**on\_mainswitch\_clicked**()  {  if(mainSwitch)  {  mainSwitch = false;  initled();  mainSwitch\_off();  ui->mainswitch->setChecked(false);  }  else  {  mainSwitch = true;  initled();  mainSwitch\_on();  ui->mainswitch->setChecked(true);  }  }  void Widget::**on\_lightlevelslider\_valueChanged**(int value)  {  lightLevel = value;  QString qcurrentLight = QString::number(value);  ui->lightlevel->setText(qcurrentLight);  ui->lightlevelset->setText(qcurrentLight);  ui->lightlevelslider->setValue(qcurrentLight.toInt());  send\_data(lightLevel,colorLevel,alertflag);  }  void Widget::**on\_colorlevelslider\_valueChanged**(int value)  {  colorLevel = value;  QString qcurrentColor = QString::number(value);  ui->colorlevel->setText(qcurrentColor);  ui->colorlevelset->setText(qcurrentColor);  ui->colorlevelslider->setValue(qcurrentColor.toInt());  send\_data(lightLevel,colorLevel,alertflag);  }  void Widget::**on\_lightlevelsub\_pressed**()  {  --lightLevel;  if(lightLevel < MIN\_LIGHT\_LEVEL) lightLevel = MIN\_LIGHT\_LEVEL;  on\_lightlevelslider\_valueChanged(lightLevel);  }  void Widget::**on\_lightleveladd\_pressed**()  {  ++lightLevel;  if(lightLevel > MAX\_LIGHT\_LEVEL) lightLevel = MAX\_LIGHT\_LEVEL;  on\_lightlevelslider\_valueChanged(lightLevel);  }  void Widget::**on\_colorlevelsub\_pressed**()  {  --colorLevel;  if(colorLevel < MIN\_COLOR\_LEVEL) colorLevel = MIN\_COLOR\_LEVEL;  on\_colorlevelslider\_valueChanged(colorLevel);  }  void Widget::**on\_colorleveladd\_pressed**()  {  ++colorLevel;  if(colorLevel > MAX\_COLOR\_LEVEL) colorLevel = MAX\_COLOR\_LEVEL;  on\_colorlevelslider\_valueChanged(colorLevel);  }  void Widget::**on\_save\_clicked**()  {  customLight = ui->lightlevelset->text().toInt();  customColor = ui->colorlevelset->text().toInt();  }  void Widget::**on\_custommode\_clicked**()  {  if(ui->custommode->isChecked())  {  sleepSwitch = false;  on\_colorlevelslider\_valueChanged(customColor);  on\_lightlevelslider\_valueChanged(customLight);  }  }  void Widget::**on\_aimode\_clicked**(bool checked)  {  if(checked)  {  sleepSwitch = false;  lightLevel = 80;  colorLevel = (TEMP\_RANGE - temp) \* (MAX\_LIGHT\_LEVEL - MIN\_LIGHT\_LEVEL) / TEMP\_RANGE;  on\_lightlevelslider\_valueChanged(lightLevel);  on\_colorlevelslider\_valueChanged(colorLevel);  }  }  void Widget::**on\_sleepmode\_clicked**()  {  sleepSwitch = true;  }  void Widget::**on\_bestdistanceset\_clicked**()  {  bestdistance = distance;  ui->bestdistance->setText(QString::number(bestdistance));  }  **Widget.ui设计ui界面**    **Zstack中的头函数led.h**  #include <ioCC2530.h>  #define COLD\_LED P1\_0 //冷光灯  #define HOT\_LED P1\_1 //暖光灯  #define ALERT\_LED P1\_4 //提示灯  #define DATA\_PIN P0\_7 //温湿度  #define TRIG P2\_0 //测距  #define ECHO P1\_3  #define LED\_MODE\_ON 0 //LED灯开  #define LED\_MODE\_OFF 1  #define LED\_CYCLE\_TIME 1000 //亮度周期  typedef unsigned char uchar;  typedef unsigned int uint;  int receiveflag = 0;  void alert\_led();  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void Sysclk\_Init(void)  \*功能描述：晶振稳定  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Sysclk\_Init(void)  {  CLKCONCMD &= ~0x40;  while( CLKCONSTA&0x40); //等待晶振稳定  CLKCONCMD &=~0x47;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void LED\_Init(void)  \*功能描述：初始化LED灯  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Led\_Init(void)  {  P1SEL &= ~0X13; //定义P1\_4,P1\_1,P1\_0为普通IO //0001 0011  P1DIR |= 0X13; //定义P1\_4,P1\_1,P1\_0为输出    //初始化测距TRIG  P2SEL &= ~0X01; //定义 P2\_0 为普通 IO //0001 0000  P2DIR |= 0X01;    //初始化全部LED灯  COLD\_LED = LED\_MODE\_OFF; //LED1灯默认为关闭状态  HOT\_LED = LED\_MODE\_OFF ; //LED3灯默认为关闭状态  ALERT\_LED = LED\_MODE\_OFF; //LED3灯默认为关闭状态  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void Uart0\_Init(void）  \*功能描述：初始化串口  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Uart0\_Init(void)  {  PERCFG = 0x00; //位置1 P0口  P0SEL = 0x0c; //P0用作串口  P2DIR &= ~0XC0; //P0优先作为UART0    U0CSR |= 0x80; //串口设置为UART方式  U0GCR |= 8;    U0BAUD |= 59; //波特率设为9600  UTX0IF = 0;  U0CSR |= 0X40; //允许接收  IEN0 |= 0x84;    /\*  PERCFG = 0x00; // ????,UART0??????1  P0SEL = 0x0c; // ??0????,P0\_2?P0\_3????  P2DIR &= ~0xC0; // ??0???????,?2??0,USART0??  U0CSR |= 0x80; // UART??  U0GCR |= 11;  U0UCR |=0x80; //流控制禁止;  U0BAUD |= 59; // ??32MHz?????,??????115200  UTX0IF = 0;  //使能中断  EA=1; //中断总开关// UART0 TX????????0  \*/  EA=1;  }  unsigned char rxTemp = 0; //传感器接收临时数据  unsigned int rb\_count = 0; //receiece\_buffer count  unsigned char recieve\_buf[6];//接收数据  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：Uart0\_Send\_String(unsigned char \*Data,int len)  \*功能描述：传感器数据发送到串口  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Uart0\_Send\_String(unsigned char \*Data,int len)  {  int i;  for(i=0;i<len;i++)  {  U0DBUF = \*Data++;  while(UTX0IF == 0);  UTX0IF = 0;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：Uart0\_Send\_String(unsigned char \*Data,int len)  \*功能描述：传感器数据发送到串口  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Usart0\_Receive\_String()  {  if(rxTemp != 0) //接收到数据  {  recieve\_buf[rb\_count] = rxTemp;  rb\_count++;  if(rb\_count >= 6)  {  rb\_count = 0;  receiveflag = 1;  }  else  {  receiveflag = 0;  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 69.  函 数 名 : Uart0\_ISR  70. 功能描述 : 中断服务函数  71. 输入参数 : NONE  72. 输出参数 : NONE  73. 返 回 值 : NONE  74. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #pragma vector = URX0\_VECTOR  \_\_interrupt void Uart0\_ISR(void)  {    URX0IF = 0; // 清中断标志  while(!U0DBUF);  rxTemp = U0DBUF;//读取接收到的数据  Usart0\_Receive\_String();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void Delay()  \*功能描述：各延时函数  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Delay\_us(uint n)  {  while(n--)  {  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");  asm("nop");  }  }  void Delay\_ms(uint n)//n ms延时  {  uint i;  while(n--)  {  for(i = 0;i < 1000;++i)  Delay\_us(1);  }  }  void Delay\_s(uint n) //n s延时  {  uint i = 0;  while(n--)  {  for(i = 0;i < 1000;++i)  Delay\_ms(1);  }  }  void Delay( int n)  {  for( int j = 0;j <5;++j)  for( int i = 0;i<n;++i);  }  //温湿度延时函数  void wDelay\_us() //1 us延时  {  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  asm("nop");  }  void wDelay\_10us() //10 us延时  {  uchar i=18;  for(;i>0;i--);  }  void wDelay\_ms(uint Time)//n ms延时  {  unsigned char i;  while(Time--)  {  for(i=0;i<100;i++)  wDelay\_10us();  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void PMW(int lightlevel,int colorlevel)  \*功能描述：LED调光  \*参数说明：lightlevel 亮度,colorlevel 色温  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void PWM(int lightlevel,int colorlevel)  {  if(lightlevel == 0)  {  COLD\_LED = LED\_MODE\_OFF;  HOT\_LED = LED\_MODE\_OFF;  }  else{  int cold\_led\_time = lightlevel \* (LED\_CYCLE\_TIME/100) \* (100-colorlevel) /100;  int hot\_led\_time = lightlevel \* (LED\_CYCLE\_TIME/100) \* colorlevel /100;  int i = 1000;  while(i--)  {  if(colorlevel >= 50) //冷光亮的时间少  {  COLD\_LED = LED\_MODE\_ON;  HOT\_LED = LED\_MODE\_ON;  Delay(cold\_led\_time);  COLD\_LED = LED\_MODE\_OFF;  Delay(hot\_led\_time - cold\_led\_time);  HOT\_LED = LED\_MODE\_OFF;  Delay(LED\_CYCLE\_TIME - hot\_led\_time);  }  else //暖光亮的时间少  {  COLD\_LED = LED\_MODE\_ON;  HOT\_LED = LED\_MODE\_ON;  Delay(hot\_led\_time);  HOT\_LED = LED\_MODE\_OFF;  Delay(cold\_led\_time - hot\_led\_time);  COLD\_LED = LED\_MODE\_OFF;  Delay(LED\_CYCLE\_TIME - cold\_led\_time);  }  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void alert\_led()  \*功能描述：提示灯闪烁五秒  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void alert\_led()  {  for(uint i = 0;i < 50;++i)  {  ALERT\_LED = LED\_MODE\_ON;  Delay\_ms(5);  ALERT\_LED = LED\_MODE\_OFF;  Delay\_ms(5);  }  }  //温湿度定义  uchar ucharFLAG,uchartemp;  uchar shidu\_shi,shidu\_ge,wendu\_shi,wendu\_ge=4;  uchar ucharT\_data\_H,ucharT\_data\_L,ucharRH\_data\_H,ucharRH\_data\_L,ucharcheckdata;  uchar ucharT\_data\_H\_temp,ucharT\_data\_L\_temp,ucharRH\_data\_H\_temp,ucharRH\_data\_L\_temp,ucharcheckdata\_temp;  uchar ucharcomdata;  //温湿度传感  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void COM(void)  \*功能描述：温湿写入  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void COM(void) // 温湿写入  {  uchar i;  for(i=0;i<8;i++)  {  ucharFLAG=2;  while((!DATA\_PIN)&&ucharFLAG++);  wDelay\_10us();  wDelay\_10us();  wDelay\_10us();  uchartemp=0;  if(DATA\_PIN)uchartemp=1;  ucharFLAG=2;  while((DATA\_PIN)&&ucharFLAG++);  if(ucharFLAG==1)break;  ucharcomdata<<=1;  ucharcomdata|=uchartemp;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void DHT11(void)  \*功能描述：温湿传感启动  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void DHT11(void) //温湿传感启动  {  DATA\_PIN=0;  wDelay\_ms(19); //>18MS  DATA\_PIN=1;  P0DIR &= ~0x80; //重新配置IO口方向  wDelay\_10us();  wDelay\_10us();  wDelay\_10us();  wDelay\_10us();  if(!DATA\_PIN)  {  ucharFLAG=2;  while((!DATA\_PIN)&&ucharFLAG++);  ucharFLAG=2;  while((DATA\_PIN)&&ucharFLAG++);  COM();  ucharRH\_data\_H\_temp=ucharcomdata;  COM();  ucharRH\_data\_L\_temp=ucharcomdata;  COM();  ucharT\_data\_H\_temp=ucharcomdata;  COM();  ucharT\_data\_L\_temp=ucharcomdata;  COM();  ucharcheckdata\_temp=ucharcomdata;  DATA\_PIN=1;  uchartemp=(ucharT\_data\_H\_temp+ucharT\_data\_L\_temp+ucharRH\_data\_H\_temp+ucharRH\_data\_L\_temp);  if(uchartemp==ucharcheckdata\_temp)  {  ucharRH\_data\_H=ucharRH\_data\_H\_temp;  ucharRH\_data\_L=ucharRH\_data\_L\_temp;  ucharT\_data\_H=ucharT\_data\_H\_temp;  ucharT\_data\_L=ucharT\_data\_L\_temp;  ucharcheckdata=ucharcheckdata\_temp;  }  wendu\_shi=ucharT\_data\_H/10;  wendu\_ge=ucharT\_data\_H%10;    shidu\_shi=ucharRH\_data\_H/10;  shidu\_ge=ucharRH\_data\_H%10;  }  else //没用成功读取，返回0  {  wendu\_shi=0;  wendu\_ge=0;    shidu\_shi=0;  shidu\_ge=0;  }    P0DIR |= 0x80; //IO口需要重新配置  }  //测距  uchar count\_start;  uchar H1;  uchar H2;  uchar L2;  uchar L1;  uchar cycle;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void UltrasoundRanging1()  \*功能描述：激活测距传感器  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void UltrasoundRanging1()  {  EA = 0;  TRIG =1;    Delay\_us(15);  TRIG =0;    T1CNTL=0;  T1CNTH=0;    while(!P1\_3);  T1CTL = 0x09;  L1=T1CNTL;  H1=T1CNTH;  EA = 1;  Delay\_ms(200);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：Init\_UltrasoundRanging()  \*功能描述：初始化端口  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Init\_UltrasoundRanging()  {  P1DIR |= 0x0d;  TRIG=0;      P1INP &= ~0x08;  P1IEN |= 0x08; //P1\_3  PICTL |= 0x02; //P0\_6  IEN2 |= 0x10; // P0IE = 1;  P1IFG = 0;    }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：void P1\_ISR(void)  \*功能描述：中断服务函数  \*参数说明：无  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #pragma vector = P1INT\_VECTOR  \_\_interrupt void P1\_ISR(void)  {  EA=0;  L2=T1CNTL;  H2=T1CNTH;    if(P1IFG&0x08)  {  P1IFG = 0;  }  else if(P1IFG&0x08)  {  P1IFG = 0;  }  P1IF = 0;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：float caldistance()  \*功能描述：测距函数  \*参数说明：返回距离(cm\_)  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  float caldistance(){  uint y;  float distance;  y=H2\*256+L2-L1-256\*H1;  distance=(float)y\*340/10000;  return distance;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \*函数名称：inputdistance(unsigned char \*send\_buf,unsigned int distance)  \*功能描述：将距离化为规格的字符串  \*参数说明：返回距离(cm\_)  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void inputdistance(unsigned char \*send\_buf,unsigned int distance)  {  if(distance >= 200)  {  send\_buf[5] = '2';  send\_buf[6] = '0';  send\_buf[7] = '0';  }  else if(distance >= 100)  {  send\_buf[5] = distance / 100 + '0';  send\_buf[6] = distance % 100 / 10 + '0' ;  send\_buf[7] = distance % 10 + '0';  }  else if(distance >=10)  {  send\_buf[5] = '0';  send\_buf[6] = distance / 10 + '0';  send\_buf[7] = distance % 10 + '0';  }  else if(distance >=0)  {  send\_buf[5] = '0';  send\_buf[6] = '0';  send\_buf[7] = distance + '0';  }  else  {  send\_buf[5] = '0';  send\_buf[6] = '0';  send\_buf[7] = '0';  }  }  **Zstack中修改的主程序**  #include "led.h"  #include "ioCC2530.h"  #include "string.h"  unsigned char send\_buf[8];  unsigned char temp\_buf[30];  float distance = 0.0;  int lightlevel = 0;  int colorlevel = 0;  void main()  {  Sysclk\_Init();  Led\_Init();  Uart0\_Init();  Init\_UltrasoundRanging();  memset(send\_buf,'0',8\*sizeof(char)); //初始化  memset(recieve\_buf,0,6\*sizeof(char));  send\_buf[0] = 'C'; //帧头  while(1)  {  //获取温湿度  DHT11();  send\_buf[1] = wendu\_shi + '0';  send\_buf[2] = wendu\_ge + '0';  send\_buf[3] = shidu\_shi + '0';  send\_buf[4] = shidu\_ge + '0';  //获取距离  UltrasoundRanging1();  distance = caldistance();  Delay\_ms(2);  //Uart0\_Send\_String(temp\_buf,sprintf(temp\_buf,"d:%.2f \n",distance));  inputdistance(send\_buf,distance);  Uart0\_Send\_String(send\_buf,8);  memset(send\_buf+1,'0',7\*sizeof(char));  if(receiveflag == 1)  {  lightlevel = (recieve\_buf[1]-'0')\*10 + (recieve\_buf[2]-'0');  colorlevel = (recieve\_buf[3]-'0')\*10 + (recieve\_buf[4]-'0');  receiveflag = 0;  if(recieve\_buf[5] == '1')  alert\_led();    //Uart0\_Send\_String(recieve\_buf,6);  //Uart0\_Send\_String(temp\_buf,sprintf(temp\_buf,"l:%d,c:%d\n",lightlevel,colorlevel));  memset(recieve\_buf,0,6\*sizeof(char));  }  PWM(lightlevel,colorlevel);  //Delay\_ms(100);    }  }  **Zstack协议栈的相关函数**  uint8 SendData(uint8 addr, uint8 FC)  {  uint8 ret, i, index=4;  TxBuffer[0] = 0x3A;  TxBuffer[1] = 0x00;  TxBuffer[2] = addr;  TxBuffer[3] = FC;  switch(FC)  {  case 0x01: //查询所有终端传感器的数据  for (i=0; i<MAX\_NODE; i++)  {  osal\_memcpy(&TxBuffer[index], NodeData[i], 4);  index += 4;  }  TxBuffer[index] = XorCheckSum(TxBuffer, index);  TxBuffer[index+1] = 0x23;    HalUARTWrite(UART0, TxBuffer, index+2);  ret = 1;  break;  case 0x02: //查询单个终端上所有传感器的数据  osal\_memcpy(&TxBuffer[index], NodeData[addr-1], 4);  index += 4;  TxBuffer[index] = XorCheckSum(TxBuffer, index);  TxBuffer[index+1] = 0x23;    HalUARTWrite(UART0, TxBuffer, index+2);  ret = 1;  break;  default:  ret = 0;  break;  }  return ret;  }  void SerialApp\_ProcessMSGCmd( afIncomingMSGPacket\_t \*pkt )  {  uint16 shortAddr;  uint8 \*pIeeeAddr;  uint8 delay;  uint8 afRxData[30]={0};    //查询单个终端上所有传感器的数据 3A 00 01 02 39 23 响应：3A 00 01 02 00 00 00 00 xor 23  switch ( pkt->clusterId )  {  // A message with a serial data block to be transmitted on the serial port.  case SERIALAPP\_CLUSTERID:  osal\_memcpy(afRxData, pkt->cmd.Data, pkt->cmd.DataLength);  switch(afRxData[0]) //简单协议命令字解析  {  #if defined(ZDO\_COORDINATOR)  case 0x3B: //收到终端无线发过来的短地址和IEEE地址,通过串口输出显示  shortAddr=(afRxData[1]<<8)|afRxData[2];  pIeeeAddr = &afRxData[3];  #if UART\_DEBUG  PrintAddrInfo(shortAddr, pIeeeAddr + Z\_EXTADDR\_LEN - 1);  #endif  break;  case 0x3A:  if(afRxData[3] == 0x02) //收到终端传过来的传感器数据并保存  {  NodeData[afRxData[2]-1][0] = afRxData[4];  NodeData[afRxData[2]-1][1] = afRxData[5];  NodeData[afRxData[2]-1][2] = afRxData[6];  NodeData[afRxData[2]-1][3] = afRxData[7];  NodeData[afRxData[2]-1][4] = 0x00;  }    #if UART\_DEBUG  HalUARTWrite (UART0, NodeData[afRxData[3]-1], 4); //调试时通过串口输出  HalUARTWrite (UART0, "\n", 1);  #endif  break;  #else  case 0x3A: //开关灯设备  if(afRxData[3] == 0x0A || afRxData[3] == 0x0B || afRxData[3] == 0x0C) //控制终端  {  if(EndDeviceID == afRxData[2] || afRxData[2]==0xFF)  {  if(afRxData[4] == 1)  {  LAMP\_PIN = 0;  HalLedSet ( HAL\_LED\_2, HAL\_LED\_MODE\_OFF );  }  else  {  LAMP\_PIN = 1;  HalLedSet ( HAL\_LED\_2, HAL\_LED\_MODE\_ON );  }  }  break;  }  #endif  default :  break;  }  break;  // A response to a received serial data block.  case SERIALAPP\_CLUSTERID2:  if ((pkt->cmd.Data[1] == SerialApp\_TxSeq) &&  ((pkt->cmd.Data[0] == OTA\_SUCCESS) || (pkt->cmd.Data[0] == OTA\_DUP\_MSG)))  {  SerialApp\_TxLen = 0;  osal\_stop\_timerEx(SerialApp\_TaskID, SERIALAPP\_SEND\_EVT);  }  else  {  // Re-start timeout according to delay sent from other device.  delay = BUILD\_UINT16( pkt->cmd.Data[2], pkt->cmd.Data[3] );  osal\_start\_timerEx( SerialApp\_TaskID, SERIALAPP\_SEND\_EVT, delay );  }  break;    default:  break;  }  } | |
| **实验结果与分析**  1、连接上协调器与节点，打开LED.exe控制界面，选择串口，则会动态显示温湿度、时间和距离。    2、打开智能台灯开关，开启智能模式  智能模式：根据温度和光照强度智能调光  睡眠模式：倒计10秒后关灯  自定义模式：自定义调节台灯亮度和色温    3、设置最佳坐姿距离，当坐姿距离低于最佳临界距离10后，近视预警灯开启，提示灯闪烁3秒    在烧写好程序和完成界面后，我们最终能够较为成功地完成zigbee组网，接收协调器传回的数据，将其显示在控制界面中，并通过控制界面控制LED灯。总体较为成功地完成了实验。 | |