****

****

**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** |
| **姓名** | **张晓燕** |
| **学号** | **2019311239** |
| **指导教师** | **田小平** |

2021年9月10日

# 目录

[目录 2](#_Toc29743)

[1 基于Zigbee工具的地下酒窖环境监测系统项目实践环节 1](#_Toc18618)

[1.1 项目目标 1](#_Toc1748)

[1.2 项目方案 1](#_Toc31811)

[1.3 项目协议设计 2](#_Toc9202)

[1.4 项目实现效果 2](#_Toc3936)

[1.5 本人在项目中的贡献及收获 3](#_Toc30291)

[2 实验三实践环节 4](#_Toc5049)

[2.1 考核要求 4](#_Toc6902)

[2.2 比较实验设计 4](#_Toc9310)

[2.3 原始实验效果 6](#_Toc12925)

[2.4 修改参数后的实验效果 6](#_Toc12814)

[2.5 本人在实验三实践环节中的收获 6](#_Toc31422)

[3 实验九实践环节 7](#_Toc15097)

[3.1 考核要求 7](#_Toc6677)

[3.2 比较实验设计 7](#_Toc4428)

[3.3 原始实验效果 9](#_Toc21046)

[3.4 修改参数后的实验效果 9](#_Toc12093)

[3.5 本人在实验九实践环节中的收获 9](#_Toc16005)

[4 实验十三实践环节 10](#_Toc19585)

[4.1 考核要求 10](#_Toc8773)

[4.2 比较实验设计 10](#_Toc18499)

[4.3 原始实验效果 17](#_Toc11544)

[4.4 修改参数后的实验效果 17](#_Toc21535)

[4.5 本人在实验十三实践环节中的收获 17](#_Toc4261)

[5 建议 19](#_Toc28592)

1 基于Zigbee工具的地下酒窖环境监测系统项目实践环节

# 1.1 项目目标

通过开发研究一套地下酒窖环境监测系统，利用气体传感器实时监测酒窖内二氧化碳和有害气体的浓度信息，当二氧化碳浓度达到上限值时系统自动发出灯光警示，从而警醒进入酒窖的人员，避免因酒窖内气体浓度过高引发中毒窒息的危险！

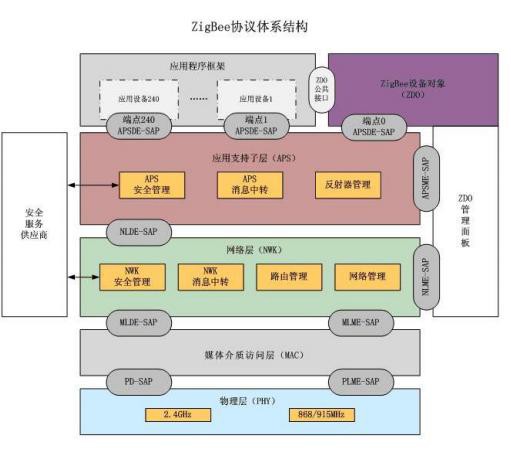
# 1.2 项目方案

本设计中主要研究对象为酒窖内的温湿度，气体浓度。通过温湿度传感器采集酒窖内的温湿度信号，通过将温湿度信号转换成字符串，供LCD显示，并将数据整合后方便发给协调器显示，获得的温湿度通过串口输出到电脑显示，最后输出到LCD显示。

另外在检测气体浓度时会用到气体传感器，用到IO引脚读取检测引脚电平，并输出显示，上传给协调器，接收数据，再利用周期性点播的定时器，间隔1秒定时采集、输出显示，将采集到的信息发送给协调器，协调器通过串口调试助手显示，并在协调器接受出上的LCD显示出来。本设计选用一片CC2530单片机作为系统的核心部件，实现控制与处理的功能，CC2530是用于2.4-GHz IEEE 802.15.4、ZigBee 和RF4CE 应用的一个真正的片上系统（SoC）解决方案。它能够以非常低的总的材料成本建立强大的网络节点。具有不同的运行模式，使得它尤其适应超低功耗要求的系统。运行模式之间的转换时间短进一步确保了低能源消耗。此次实验选用的是F256型号的CC2530，结合了德州仪器的业界领先的黄金单元ZigBee 协议栈（Z-Stack™），提供了一个强大和完整的ZigBee 解决方案。

另外此方案选用了DHT11温湿度传感器，DHT11是一款有已校准数字信号输出的温湿度传感器，其精度湿度±5%RH，温度±2℃，量程湿度5~95%RH，温度-20~+60℃、MQ-2光敏传感器，MQ-2型传感器具有良好的抗干扰性，重复性和长期的稳定性，初始稳定，响应时间短，长时间工作性能好。通过温湿度传感器实时监测并显示出酒窖内的温度湿度，从而知晓酒窖内酒的发酵情况。而气体传感器用于测出当前环境中各气体的含量，以防止单一或多种气体含量超标的现象出现。

# 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。

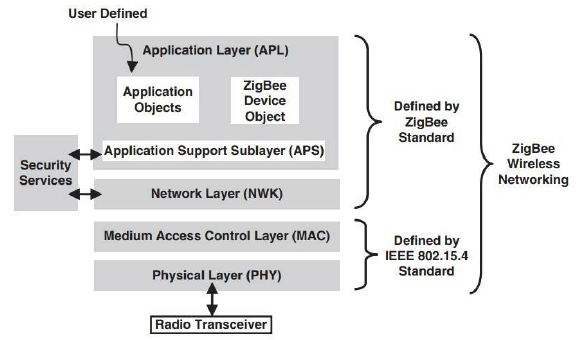


图1 ZigBee无线网络协议层的架构图

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

通过开发研究一套地下酒窖环境监测系统，利用气体传感器实时监测酒窖内二氧化碳和有害气体的浓度信息，当二氧化碳浓度达到上限值时系统自动发出灯光警示，从而警醒进入酒窖的人员，避免因酒窖内气体浓度过高引发中毒窒息的危险！

本设计中主要研究对象为酒窖内的温湿度，气体浓度。通过温湿度传感器采集酒窖内的温湿度信号，通过将温湿度信号转换成字符串，供LCD显示，并将数据整合后方便发给协调器显示，获得的温湿度通过串口输出到电脑显示，最后输出到LCD显示。

# 1.5 本人在项目中的贡献及收获

作为小组的项目成员，我在此次项目中主要负责的是进行资料的查询和项目汇报PPT的编写。及时与老师和同组成员讨论分析，进一步优化和细化我们的设计方案。

在实验开始初期，就我们队伍的项目内容以及项目的可行性进行了交流，最后基于实际问题决定开展基于Zigbee工具的地下酒窖环境监测系统这一项目。后续就本项目所需要的元器件展开讨论，对本项目进行了初步的构思和搭建。

在对装置进行其具体搭建的过程中，如何将不同传感器的代码进行整合是我们遇到的一个较大的问题，最终通过与老师不断沟通交流实现了该功能。在这个过程中，老师鼓励我们创新，积极引导我们敢于提出问题，让我们学会用创新的思维方式去观察、分析问题。小组成员自主探究、合作交流，在观察、实验、验证交流等过程中发现问题并及时解决问题。主动联系实际、应用拓展。实现解决问题的能力与知识、技能的同步提升。

2 实验三实践环节

# 2.1 考核要求

题目：外部中断控制流水灯

要求：增加按键中断5次时全亮

# 2.2 比较实验设计

通过改变if判断语句，将按键次数改为5.当按键中断5次时LED3,LED2,LED1倒序流水闪烁

**原始代码：**

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;

}

}

}

**修改代码：**

void main(void)

{

InitIO();

InitKeyINT(); //调用初始化函数

while(1)

{

**if(KeyTouchtimes == 5)** //按键设置为LED3,LED2,LED1倒序流水闪烁

{

LED3 = 0;

Delay(30000);

LED2 = 0;

Delay(30000);

LED1 = 0;

Delay(30000);

KeyTouchtimes = 0;

}

}

}

# 2.3 原始实验效果

当按键中断1次时LED3,LED2,LED1即可按倒序流水闪烁

# 2.4 修改参数后的实验效果

当按键中断5次时LED3,LED2,LED1倒序流水闪烁

# 2.5 本人在实验三实践环节中的收获

在与老师不断沟通和交流的过程中，老师鼓励我们创新，积极引导我们敢于提出问题，让我们学会用创新的思维方式去观察、分析问题。小组成员自主探究、合作交流，在观察、实验、验证交流等过程中发现问题并及时解决问题。主动联系实际、应用拓展。实现解决问题的能力与知识、技能的同步提升。学会利用整体的方法去思考和解决问题。进一步了解和学习了if语句的使用方法，能够大体读懂程序代码，根据具体题目的要求对已有的程序代码进行修改；可以通过改变延时函数来改变流水灯闪烁的频率；实现了通过按键控制流水灯的功能；可以独立自主的解决系统调试过程中出现的一些问题。

3 实验九实践环节

# 3.1 考核要求

题目：串口UART0发送字符串

要求：将波特率改为115200

# 3.2 比较实验设计

需要将代码中的波特率进行修改，并对串口软件中的波特率进行修改，从而实现程序的正常运行。

**原始代码：**

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

}

**修改代码：**

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 |= 11; //U0GCR.BAUD\_E**

**U0BAUD |= 216; //波特率设为9600 UxBAUD.BAUD\_M**

UTX0IF = 0; //UART0 TX中断标志初始置位0

}

# 3.3 原始实验效果

D2 串口发生变化，每发送一串字符闪一次

# 3.4 修改参数后的实验效果

修改系统波特率参数后，实验现象较改变前并不会发生变化

# 3.5 本人在实验九实践环节中的收获

能够根据本实验提供的相关资料，比对不同的参数对源代码的波特率进行修改。通过实践过程了解到需要先将代码下载到开发板中，再打开串口调试软件进行波特率的修改；除此之外还要将程序中的波特率和串口的波特率调整为统一参数这样才能保证程序运行过程中不会出现乱码现象；同时还要先将串口调试软件关闭后再将USB数据线拔出，否则会出现串口调试软件无法关闭的现象。

4 实验十三实践环节

# 4.1 考核要求

题目：睡眠定时器唤醒系统

要求：用不同的逻辑运算改变赋值

# 4.3 原始实验效果

开机后 LED1 闪 3 次后进入睡眠，睡眠 5 秒后，自动醒来，LED1 闪 3 次后再次进入睡眠。LED2 当睡眠醒来，继续跑，亮灭交替。

# 4.4 修改参数后的实验效果

用不同的逻辑运算改变赋值并不会影响实验效果，实验现象较之前并没有发生变化。

# 4.2 比较实验设计

**原代码：**

void InitLed(void)

{

**P1DIR |= 0x03; //P1.0定义为输出口**

LED1 = 1; //LED1灯上电默认为熄灭

}

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

\* 名 称: SysPowerMode()

\* 功 能: 设置系统工作模式

\* 入口参数: mode等于0为PM0 1为PM1 2为PM2 3为PM3

\* 出口参数: 无

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

void SysPowerMode(uchar mode)

{

if(mode < 4)

{

SLEEPCMD |= mode; //设置系统睡眠模式

PCON = 0x01; //进入睡眠模式 ,通过中断唤醒

}

else

PCON = 0x00; //通过中断唤醒系统

}

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

\* 名 称: ST\_ISR(void) 中断处理函数

\* 描 述: #pragma vector = 中断向量，紧接着是中断处理程序

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

#pragma vector = ST\_VECTOR

\_\_interrupt void ST\_ISR(void)

{

STIF = 0; //清标志位

SysPowerMode(4); //进入正常工作模式

}

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

\* 名 称: SysPowerMode()

\* 功 能: 初始化休眠定时器,设定后经过指定时间自行唤醒

\* 入口参数:

\* 出口参数: 无

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

void InitSleepTimer(void)

{

ST2 = 0X00;

ST1 = 0X0F;

ST0 = 0X0F;

EA = 1; //开中断

STIE = 1; //睡眠定时器中断使能 0： 中断禁止 1： 中断使能

STIF = 0; //睡眠定时器中断标志 0： 无中断未决 1： 中断未决

}

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

\* 名 称: Set\_ST\_Period()

\* 功 能: 设置睡眠时间

\* 入口参数: sec 睡眠时间

\* 出口参数: 无

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

void Set\_ST\_Period(uint sec)

{

ulong sleepTimer = 0;

**sleepTimer |= ST0;**

**sleepTimer |= (ulong)ST1 << 8;**

**sleepTimer |= (ulong)ST2 << 16;**

sleepTimer += ((ulong)sec \* (ulong)32768);

ST2 = (uchar)(sleepTimer >> 16);

ST1 = (uchar)(sleepTimer >> 8);

ST0 = (uchar) sleepTimer;

}

**修改代码：**

void InitLed(void)

{

**P1DIR =~ 0xFC; //P1.0定义为输出口**

LED1 = 1; //LED1灯上电默认为熄灭

}

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

\* 名 称: SysPowerMode()

\* 功 能: 设置系统工作模式

\* 入口参数: mode等于0为PM0 1为PM1 2为PM2 3为PM3

\* 出口参数: 无

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

void SysPowerMode(uchar mode)

{

if(mode < 4)

{

SLEEPCMD |= mode; //设置系统睡眠模式

PCON = 0x01; //进入睡眠模式 ,通过中断唤醒

}

else

PCON = 0x00; //通过中断唤醒系统

}

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

\* 名 称: ST\_ISR(void) 中断处理函数

\* 描 述: #pragma vector = 中断向量，紧接着是中断处理程序

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

#pragma vector = ST\_VECTOR

\_\_interrupt void ST\_ISR(void)

{

STIF = 0; //清标志位

SysPowerMode(4); //进入正常工作模式

}

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

\* 名 称: SysPowerMode()

\* 功 能: 初始化休眠定时器,设定后经过指定时间自行唤醒

\* 入口参数:

\* 出口参数: 无

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

void InitSleepTimer(void)

{

ST2 = 0X00;

ST1 = 0X0F;

ST0 = 0X0F;

EA = 1; //开中断

STIE = 1; //睡眠定时器中断使能 0： 中断禁止 1： 中断使能

STIF = 0; //睡眠定时器中断标志 0： 无中断未决 1： 中断未决

}

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

\* 名 称: Set\_ST\_Period()

\* 功 能: 设置睡眠时间

\* 入口参数: sec 睡眠时间

\* 出口参数: 无

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

void Set\_ST\_Period(uint sec)

{

ulong sleepTimer = 0;

**sleepTimer != ~ST0;**

**sleepTimer != ~(ulong)ST1 << 8;**

**sleepTimer != ~(ulong)ST2 << 16;**

sleepTimer += ((ulong)sec \* (ulong)32768);

ST2 = (uchar)(sleepTimer >> 16);

ST1 = (uchar)(sleepTimer >> 8);

ST0 = (uchar) sleepTimer;

}

# 4.5 本人在实验十三实践环节中的收获

根据已经学过的知识和最新掌握的，利用三种常用的逻辑运算符与、或、非实现对变量的赋值。除此之外还要考虑不同逻辑运算符的优先级，综合考虑多方面因素实现对程序的修改和准确运行！

5 建议

在保证可行性的情况下，尽可能让更多的同学参与进去，和大家一起沟通交流。在条件允许的情况下，尽量让大家能人手一套设备，锻炼和提高大家的动手协调能力。我们此次实验课程有些同学积极性不够，缺乏练习和实践，可能会出现各自能力参差不齐的情况，如果可以最好大家最后都可以人手实现一个自主项目。可以让各自小组长先跟随老师进行集训，再由小组长对自己的组员进行负责，如此一来可以节省时间也能帮老师分担一部分工作，除此之外先让小组长进行学习还可以提高效率，大家的注意力也会相对更加集中。

希望在教学安排中，老师增加理论知识环节的课时，这样有助于同学们更好的理解ZigBee开发板的基本原理，同时，期待此类实践课程能够更多一些，并且动员更多优秀的学长学姐加入到平时的实践指导中，这样可以保证每位同学都能让自己的问题得到详细的解决，同时也可以减小老师的负担，此类自主设计的实践课程的练习，对于工科生的培养有着深远的意义，即是对编程能力的锻炼，也是动手组装硬件能力的体现，这样软硬件相结合的课程正是我们通信工程专业学生所具备的，值得推广。