

**课 程 设 计**

**课 程：物联网工程开发与应用实例**

**题 目：基于Zigbee的智能温湿度检测系统-终端设计**

**学生姓名：黄鑫达**

**学 号：15115061008**

**二级学院：信息科学与工程学院**

**专 业：物联网工程**

**班 级：2015级01班**

**指导教师姓名及职称：刘晓樑 讲师**

**起止时间：2018年 9 月—— 2019年 1月**

（教务处制）

**基于Zigbee的智能温湿度检测系统**

**摘 要**： 随着信息技术的发展，在各行各业信息化已经越来越广泛，尤其在农业方面，过去的靠天吃饭已经不再为人们所用。通过信息化，许多农作物可以通过传感器来实时检测其状态，本次系统所做的就是利用温湿度传感器检测出农作物附近的温湿度环境，从而采取有效的措施去对农作物进行种植。

同时，该系统还能在家庭环境中使用，通过传感器将家里的温湿度传到网络中，实时检测出家里的状况。本系统利用Zigbee进行通信，通过嵌入式系统来实施检测，利用QT生成检测界面，让人们更方便的看到实时数据。

**关键字**： ZigBee；局域网；QT

**Zigbee-based intelligent temperature and humidty detection system**

**Abstract:** With the development of information technology, information technology in all walks of life has been more and more widely, especially in agriculture, the weather is no longer used by the information for people, many of the crops can be real time detection by the sensor. The system does is near the temperature and humidity environment temperature and humidity sensors to detect the crops, so as take effective measures to grow crops at the same time.

The system also can be used in the family environment, through the sensor, the temperature and humidity of the home to the network, the real time detect the status of the home in this system, Zigbee is used for communication, embedded system is used for detection, and QT is used to generate detection interface, so that people can see real time data more conveniently.

**Keywords:** Zigbee ;Local Area Network; QT

**目录**

[1温湿度监测系统的设计相关技术 1](#_Toc497056372)

[1.1 ZigBee传输技术 1](#_Toc497056373)

[1.2 Z-Stack协议栈 2](#_Toc497056374)

[1.3 IAR开发环境 3](#_Toc497056375)

[1.4终端及协调器 4](#_Toc497056376)

[2 系统总体设计方案 4](#_Toc497056378)

[2.1功能结构分析 4](#_Toc497056379)

[2.2技术流程 4](#_Toc497056380)

[2.3 各模块功能 5](#_Toc497056381)

[2.3.1采集模块 5](#_Toc497056382)

[2.3.2数据传输模块 5](#_Toc497056383)

[2.3.3协调器模块 8](#_Toc497056385)

2.3.4终端控制显示模块………………………………….…………………….…8

2.4 数据库设计……………………………………………………………………….….9

[3 系统硬件设计与实现 1](#_Toc497056388)0

[3.1 硬件部分 1](#_Toc497056389)0

[3.1.1 CC2530原理图 1](#_Toc497056390)0

[3.1.2 传感器原理图 1](#_Toc497056390)1

[3.1.3 数据采集模块 12](#_Toc497056393)

[3.2 终端代码设计 12](#_Toc497056399)

[4系统测试 2](#_Toc497056400)0

[4.1 硬件测试 2](#_Toc497056401)0

[4.2 软件测试 21](#_Toc497056402)

[5 总结及展望 22](#_Toc497056403)

[5.1 总结 22](#_Toc497056404)

[5.2 展望 22](#_Toc497056405)

[参考文献 22](#_Toc497056407)

**基于Zigbee的智能温湿度检测系统-终端设计**

**1 温湿度检测的设计相关技术**

## ZigBee传输技术

本文采用ZigBee技术，ZigBee技术是一种新兴的短距离、低复杂度、低功耗、低数据速率、低成本的无线网络技术，因其具有低功耗、低成本、可靠性高、灵活的拓扑结构，被广泛应用于各类智能系统中。通过ZigBee技术，可以实时检测空气温湿度的变化，可应用于各个场所中。

红外技术是比较早出现的无线技术，虽然有成本低、简单易用的特点，但是，有物体阻挡时，便显得无可奈何了，且仅可以直线传播，所以，不符合本设计需求；WiFi是应用的最多、最流行无线通信技术，传输最远范围是100-300米，它的最高速率将近300Mbps，功耗在10-15mA之间；Bluetooth的有效覆盖半径仅有2-30米，功耗小于WiFi，但大于ZigBee；ZigBee有效覆盖半径在50-300米之间，功耗5mA，还可以自组网，节点数为216个。显然不管是在功耗还是成本，或是传输距离方面ZigBee都是最适合本设计的。

经科学地分析和比对，本系统选择了ZigBee技术作为本系统的无线传输技术支持。另外ZigBee通信需要至少需要两个CC2530模块，如图2-1所示，左边的节点是用于与传感器相连接的，右边的协调器是用于接收节点数据或者发送指令的。在本系统中，用了1个节点（节点是指连接传感器的主控模块），和一个协调器模块来搭建无线网和完成数据的汇总。



图1-1 ZigBee组网套件

## 

## 1.2 Z-Stack协议栈

## Z-Stack是用于CC2530组件的小型的操作系统，它是一个ZigBee模块化的协议栈，能用于于多个平台开发。它是一种小型的操作系统，它里面集成了很多函数方法，比如、、、等，可以十分容易的初始化操作系统，配置硬件，还可以十分方便的使用定时、串口、点灯等功能，可以大大的提高程序员编程效率。它主要进行了CC2530模块硬件的初始化，如WatchDog的使能、中断的初始化、显示屏的初始化等。Z-Stack协议栈架构如图1-2，分为物理层、MAC、NWK、APL层面。其工作流程如图1-3所示：

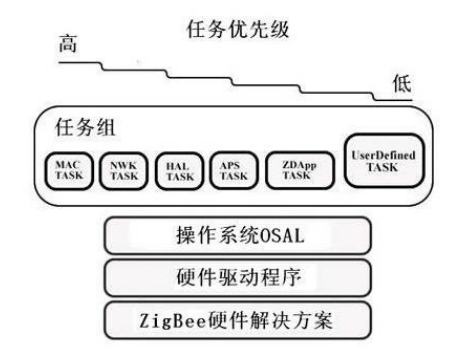


图1-2 Z-Stack软件架构

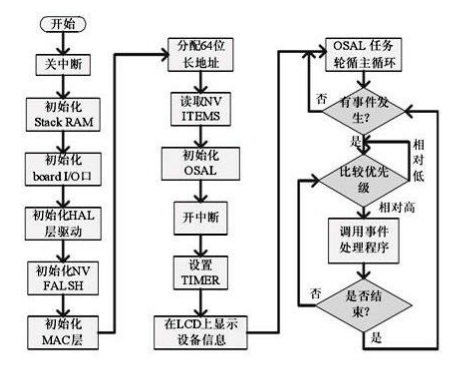


图1-3 Z-Stack软件流程图

## 1.3 IAR开发环境

## 硬件开发用的是Windows环境下的IAR8.10版本，IAR可以用于CC2530的开发，它是优秀的C编译器，它能够支持非常多半导体企业的芯片，例如ARM、CC2540、CC2530等芯片。它能够打开Z-Stack协议栈，在Z-Stack协议栈的配合下，可以很方便的开发出ZigBee+传感器框架的各种产品，开发软件界面如图1-4所示。

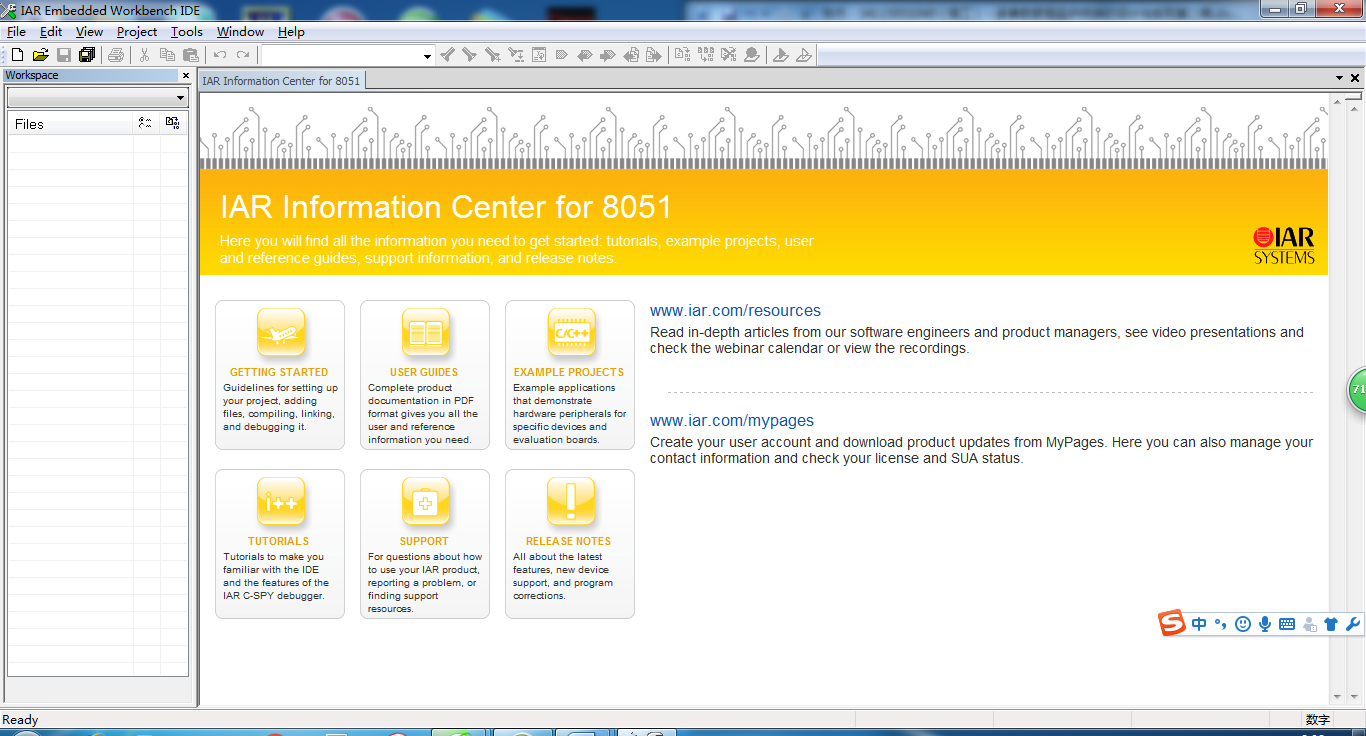


图1-4 IAR集成开发软件界面

## 1.4终端及协调器

## 被经常用作终端机的主要设备分别是三种手机、电脑及嵌入式设备。首先手机不能直接和ZigBee通信，需要通过复杂的转换，且难以保证24小时不间断监护，所以本文不推荐使用手机作为终端；通过在电脑生成QT界面，直接与协调器接收数据，比较简单，因此选择用电脑作为数据显示的载体。

协调器作为终端接收的模块，是将终端接收后的数据进行一系列的计算后，通过串口传输到电脑上进行显示。网络协调器的主要功能是协调建立网络，其他功能还包括：传输网络信标、管理网络节点以及存储网络节点信息，并且提供关联节点之间的路由信息；此外，网络协调器要存储一些基本信息，如节点数据设备、数据转发表以及设备关联表等。

# **2系统总体设计方案**

## 2.1功能结构分析

本设计利用温湿度传感器采集农作物、家庭周围环境的温湿度利用CC2530搭建的ZigBee无线传感网络进行数据的实时传输，所有数据统一聚合到温湿度终端上做解析、处理、存储，数据再经过局域网转发至QT界面上，再由数据库来存储。

其中，数据库表格包含用户登录报表，环境温湿度报表等。

## 2.2技术流程

使用温湿度传感器结合CC2530模块（温湿度模块）搭建传感器节点模组，来采集周围环境各种信号，再通过cc2530模块搭建zigbee无线网络，将温湿度信号无线转发至协调器端，协调器再利用串口通信的方式有线给电脑，最终在电脑上进行数据分析处理，在嵌入式设备搭建的QT图形界面显示实时提供温湿度信息等。

## 2.3 各模块功能

### 2.3.1采集模块

系统采集模块能采集周围环境的温湿度。

DHT11温湿度传感器是一款有已校准数字信号输出的温湿度传感器。精准湿度+-5%RH，温度+-2℃，量程湿度20-90%RH，温度0-50℃。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温湿度检验室中进行校准，校准系数以程序的形式存在OTP内存中，传感器内部在检测型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，使其成为最为苛刻的应用场合的最佳选择。产品为3针单排引脚封装。

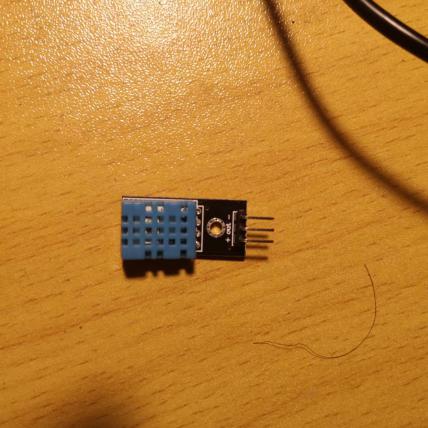


图2.1 DHT11温湿度传感器

### 2.3.2数据传输模块

数据传输部分有一个无线和一个有线模块。无线模块：ZigBee模块； ZigBee模块是来完成温湿度检测数据的无线通信的。即建立一个ZigBee网，把所有温湿度数据进行收集，无线传送到协调器上，实现实时检测数据。

UART驱动和USB串口接收，本设计有一处使用了UART传输协议和USB串口传输，分别是传感器与CC2530的通信和协调器与PC机进行数据传输。它在本设计的作用主要是串联各种数据，使得数据到达PC端控制显示温湿度数据，然后做处理。

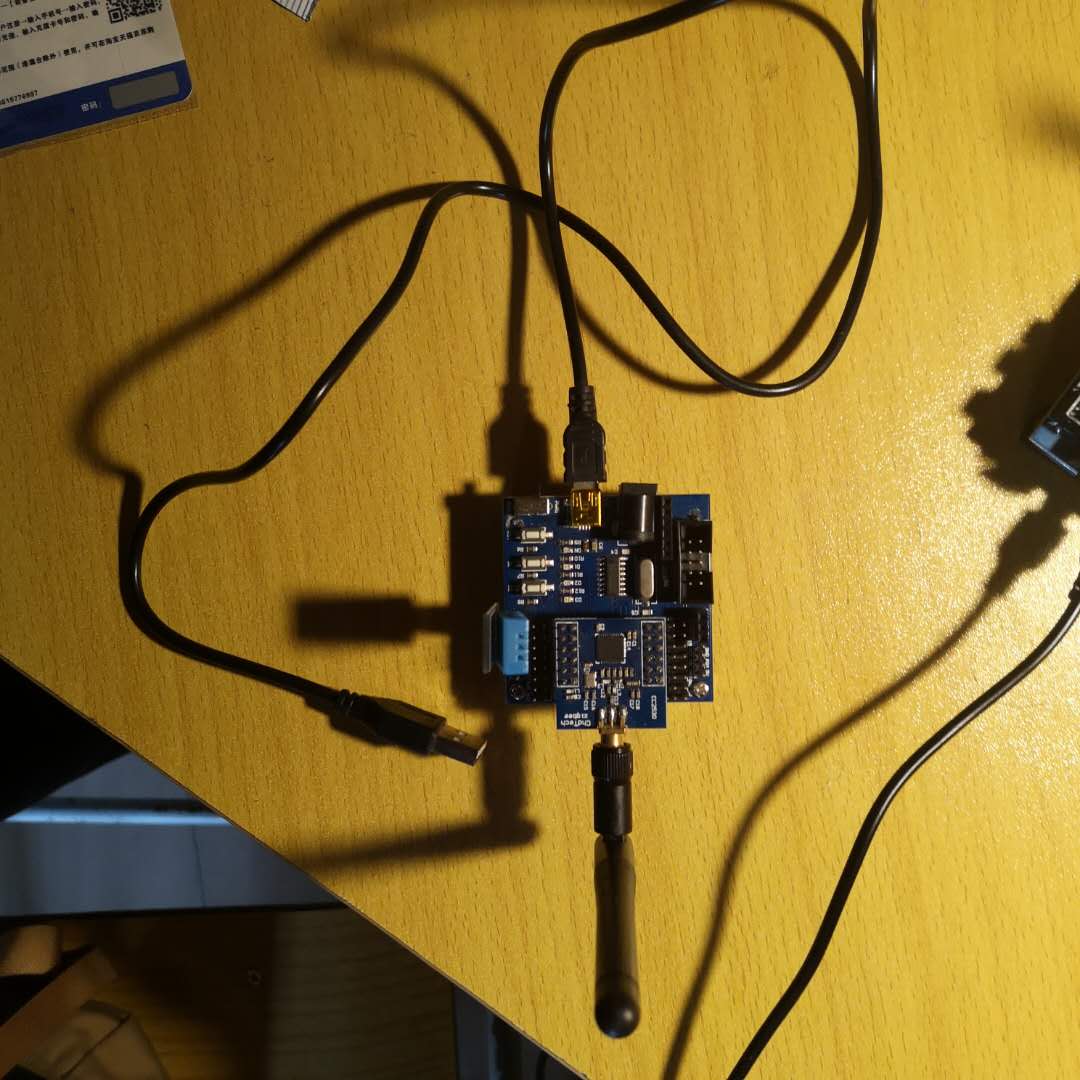


图2-2 终端模块

### 传感器采集主要代码代码如下：

void COM(void); // 温湿写入

void DHT11(void) ; //温湿传感启动

#define uchar unsigned char

#define uint unsigned int

//温湿度定义

uchar shidu\_shi,shidu\_ge,wendu\_shi,wendu\_ge;

uchar ucharFLAG,uchartemp;

//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\_10us();

Delay\_10us();

Delay\_10us();

uchartemp=0;

if(DATA\_PIN)uchartemp=1;//如果超过30us还没置0，则数据为1，置0则数据为0

ucharFLAG=2;

while((DATA\_PIN)&&ucharFLAG++); //等待置0

if(ucharFLAG==1)break;

ucharcomdata<<=1;//ucharcomdata左移一位赋值给ucharcomdata

ucharcomdata|=uchartemp;//两句合起来就是给最高位赋值

}

}

void DHT11(void) //温湿传感启动

{

DATA\_PIN=0;//置低电平

Delay\_ms(19); //>18MS

DATA\_PIN=1; //置高电平，开始信号发送结束

P0DIR &= ~0x80; //重新配置IO口方向，输入

Delay\_10us();

Delay\_10us();

Delay\_10us();

Delay\_10us(); //高电平，延时40us等待

if(!DATA\_PIN) //如果电平变低，则证明检测到开始信号

{

ucharFLAG=2;

while((!DATA\_PIN)&&ucharFLAG++);//低电平，等待响应结束

ucharFLAG=2;

while((DATA\_PIN)&&ucharFLAG++); //高电平，准备输出

COM();

ucharRH\_data\_H\_temp=ucharcomdata;//湿度高八位

COM();

ucharRH\_data\_L\_temp=ucharcomdata;//湿度低八位

COM();

ucharT\_data\_H\_temp=ucharcomdata;//温度高八位

COM();

ucharT\_data\_L\_temp=ucharcomdata;//温度低八位

COM();

ucharcheckdata\_temp=ucharcomdata;//校验八位

DATA\_PIN=1; //置高，空闲状态

### 2.3.3协调器模块

协调器模块是通过Zigebee网络与终端模块进行通信，通过Zigbee协议将传感器的数据进行接收，然后在协调器通过函数处理后通过串口连接发送到PC端，具体如图所示



图2-3 协调器模块

### 2.3.4 终端控制显示模块

终端控制显示模块是由Qt来编写，通过虚拟机直接在PC端显示实时温湿度数据，其内容如图2-2所示。

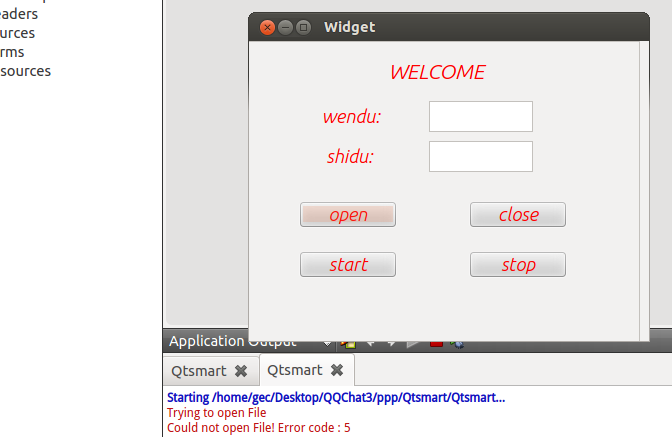


图2-4 温湿度显示界面

## 2.4 数据库设计

本设计中，用户登录信息和用户监测的温湿度数据通过IP连接然后将数据发送到数据库中

其中用户登录信息表设计如图3-5，由5个字段组成。分别是usr\_id(登录名)、usr\_pwd（用户密码）。



图2-5 数据库登陆界面



图2-6 数据库数据界面

## **3 系统硬件的设计****与实现**

## 3.1 硬件部分

### 3.1.1 CC2530原理图

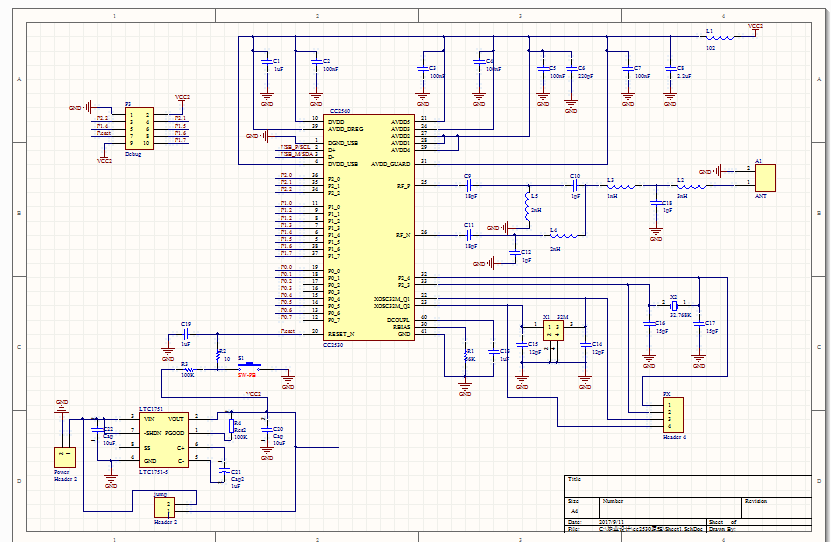


图3-1 CC2530核心板

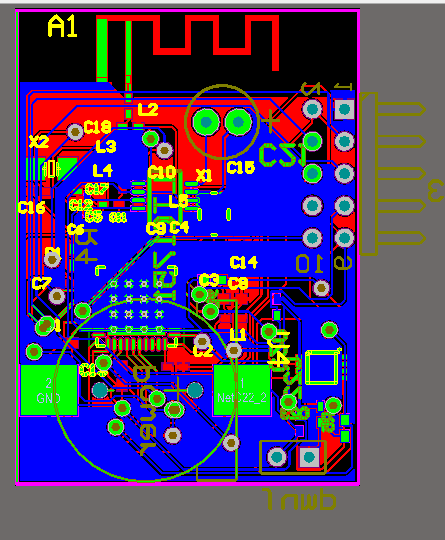


图3-2 CC2530核心板 PCB图

如图3-1 CC2530原理图，它是CC2530最小系统，其中包含了稳压模块的设计、时钟、复位、debug与射频电路。根据CC2530数据手册可知，CC2530的宽电源电压范围是2V-3.6V，本设计可以使用电源为3V的cr2032电池或5v的USB电源，分别为稳定输出电源选择了LTC1751-3.3稳压模块将电源稳定在3.3V和74LVC245将5v电源转为3.3v。其中LTC1751-3.3电压输入范围是2v-4.4v，输出电压典型值为3.3V，74LVC245最高输入电压可达5V，输出约2.7v-3.6v满足CC2530供电需求。图3-2为cc2530 PCB版图。

3.1.2 传感器原理图

本系统采用的是DHT11温湿度传感器，下图为传感器的类型，通过一个湿度元件和温度元件进行测量，测出了十六位的二进制的数据，通过算法将其变成十进制的数字，接着通过Zigbee发送到协调器，再由协调器通过USB串口发送到PC端

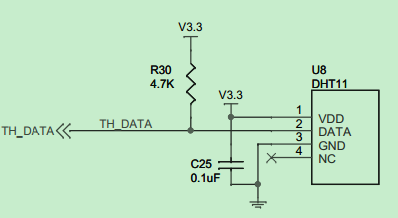


图3-3 DHT11连接示意图

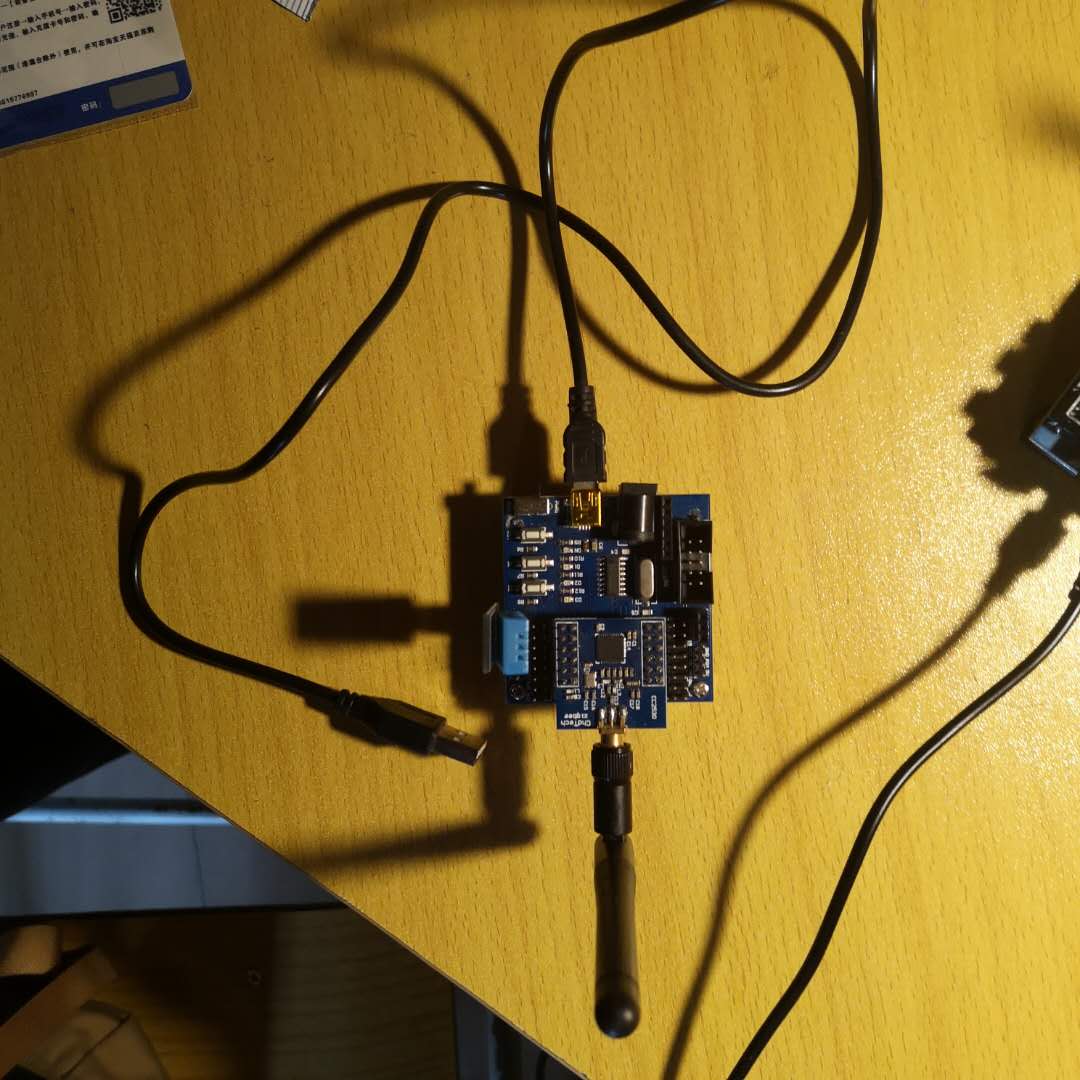


图3-4 DHT11实物连接图

### 3.1.3 数据采集模块

采集模块开发，即参数在CC2530上的处理，温湿度使用串口进行通信的，，使用串口有个好处就是z-stack协议栈定义好了回调函数接口，一旦节点接收到串口就绪，就会自动调用rxCB()，因此只需要在回调函数执行时调用发送函数即可。同理，协调器发送指令时，收到节点发来的射频数据时直接写入到串口发送至PC端。

## 3.2 终端代码设计

本人在组内是负责该系统的终端部分，其他内容简略带过，以下是终端的代码：

#include <stdio.h>

#include "OSAL.h"

#include "AF.h"

#include "ZDApp.h"

#include "ZDObject.h"

#include "ZDProfile.h"

#include <string.h>

#include <ioCC2530.h>

//#include "GenericApp.h"

#include "DebugTrace.h"

#include "DH.h"

#if !defined( WIN32 )

#include "OnBoard.h"

#endif

/\* HAL \*/

#include "hal\_lcd.h"

#include "hal\_led.h"

#include "hal\_key.h"

#include "hal\_uart.h"

#define SEND\_DATA\_EVENT 0x01

#define DATA\_PIN P0\_7

void Delay\_us(void); //1 us延时

void Delay\_10us(void); //10 us延时

void COM(void); // 温湿写入

void DHT11(void) ; //温湿传感启动

#define uchar unsigned char

#define uint unsigned int

//温湿度定义

uchar shidu\_shi,shidu\_ge,wendu\_shi,wendu\_ge;

uchar ucharFLAG,uchartemp;

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;

//--------------------------------

//-----湿度读取子程序 ------------

//--------------------------------

//----以下变量均为全局变量--------

//----温度高8位== U8T\_data\_H------

//----温度低8位== U8T\_data\_L------

//----湿度高8位== U8RH\_data\_H-----

//----湿度低8位== U8RH\_data\_L-----

//----校验 8位 == U8checkdata-----

//----调用相关子程序如下----------

//---- Delay();, Delay\_10us();,COM();

//--------------------------------

//延时函数

void Delay\_us() //1 us延时

{

MicroWait(1);

}

void Delay\_10us() //10 us延时

{

MicroWait(10);

}

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\_10us();

Delay\_10us();

Delay\_10us();

uchartemp=0;

if(DATA\_PIN)uchartemp=1;//如果超过30us还没置0，则数据为1，置0则数据为0

ucharFLAG=2;

while((DATA\_PIN)&&ucharFLAG++); //等待置0

if(ucharFLAG==1)break;

ucharcomdata<<=1;//ucharcomdata左移一位赋值给ucharcomdata

ucharcomdata|=uchartemp;//两句合起来就是给最高位赋值

}

}

void DHT11(void) //温湿传感启动

{

DATA\_PIN=0;//置低电平

Delay\_ms(19); //>18MS

DATA\_PIN=1; //置高电平，开始信号发送结束

P0DIR &= ~0x80; //重新配置IO口方向，输入

Delay\_10us();

Delay\_10us();

Delay\_10us();

Delay\_10us(); //高电平，延时40us等待

if(!DATA\_PIN) //如果电平变低，则证明检测到开始信号

{

ucharFLAG=2;

while((!DATA\_PIN)&&ucharFLAG++);//低电平，等待响应结束

ucharFLAG=2;

while((DATA\_PIN)&&ucharFLAG++); //高电平，准备输出

COM();

ucharRH\_data\_H\_temp=ucharcomdata;//湿度高八位

COM();

ucharRH\_data\_L\_temp=ucharcomdata;//湿度低八位

COM();

ucharT\_data\_H\_temp=ucharcomdata;//温度高八位

COM();

ucharT\_data\_L\_temp=ucharcomdata;//温度低八位

COM();

ucharcheckdata\_temp=ucharcomdata;//校验八位

DATA\_PIN=1; //置高，空闲状态

uchartemp=(ucharT\_data\_H\_temp+ucharT\_data\_L\_temp+ucharRH\_data\_H\_temp+ucharRH\_data\_L\_temp);

if(uchartemp==ucharcheckdata\_temp)//校验

{

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

}

// This list should be filled with Application specific Cluster IDs.

const cId\_t GenericApp\_ClusterList[GENERICAPP\_MAX\_CLUSTERS] =

{

GENERICAPP\_CLUSTERID

};

const SimpleDescriptionFormat\_t GenericApp\_SimpleDesc =

{

GENERICAPP\_ENDPOINT, // int Endpoint;

GENERICAPP\_PROFID, // uint16 AppProfId[2];

GENERICAPP\_DEVICEID, // uint16 AppDeviceId[2];

GENERICAPP\_DEVICE\_VERSION, // int AppDevVer:4;

GENERICAPP\_FLAGS, // int AppFlags:4;

0,

(cId\_t \*)NULL, // 改 byte \*pAppInClusterList;

GENERICAPP\_MAX\_CLUSTERS, // byte AppNumInClusters;//输入簇ID的个数

(cId\_t \*)GenericApp\_ClusterList, // byte \*pAppInClusterList;

};

// This is the Endpoint/Interface description. It is defined here, but

// filled-in in GenericApp\_Init(). Another way to go would be to fill

// in the structure here and make it a "const" (in code space). The

// way it's defined in this sample app it is define in RAM.

endPointDesc\_t GenericApp\_epDesc;

byte GenericApp\_TaskID;

byte GenericApp\_TransID; // This is the unique message ID (counter)

devStates\_t GenericApp\_NwkState;//用于保存节点状态信息

void GenericApp\_MessageMSGCB( afIncomingMSGPacket\_t \*pckt );//消息处理函数

void GenericApp\_SendTheMessage( void );

void GenericApp\_Init( byte task\_id )

{

P1\_0=0;

P1\_1=0;

GenericApp\_TaskID = task\_id;

GenericApp\_NwkState = DEV\_INIT;//节点状态初始化为DEV\_INIT，表示未连接到zigbee上

GenericApp\_TransID = 0;

GenericApp\_epDesc.endPoint = GENERICAPP\_ENDPOINT;

GenericApp\_epDesc.task\_id = &GenericApp\_TaskID;

GenericApp\_epDesc.simpleDesc

= (SimpleDescriptionFormat\_t \*)&GenericApp\_SimpleDesc;

GenericApp\_epDesc.latencyReq = noLatencyReqs;

}

UINT16 GenericApp\_ProcessEvent( byte task\_id, UINT16 events )

{

afIncomingMSGPacket\_t \*MSGpkt;

if ( events & SYS\_EVENT\_MSG )

{

MSGpkt = (afIncomingMSGPacket\_t \*)osal\_msg\_receive( GenericApp\_TaskID );

//读取节点的设备类型

while ( MSGpkt )

{

switch ( MSGpkt->hdr.event )

{

case ZDO\_STATE\_CHANGE:

case AF\_INCOMING\_MSG\_CMD:

GenericApp\_MessageMSGCB(MSGpkt);

GenericApp\_NwkState = (devStates\_t)(MSGpkt->hdr.status);

if (GenericApp\_NwkState == DEV\_END\_DEVICE )

{

// Start sending "the" message in a regular interval.

GenericApp\_SendTheMessage();

osal\_set\_event(GenericApp\_TaskID,SEND\_DATA\_EVENT);

}

break;

default:

break;

}

// Release the memory

osal\_msg\_deallocate( (uint8 \*)MSGpkt );

// Next

MSGpkt = (afIncomingMSGPacket\_t \*)osal\_msg\_receive( GenericApp\_TaskID );

}

// return unprocessed events

return (events ^ SYS\_EVENT\_MSG);

}

if(events & SEND\_DATA\_EVENT)

{

///HalLedBlink(HAL\_LED\_1,0,50,500);

GenericApp\_SendTheMessage();

//HalLedBlink(HAL\_LED\_1,0,50,500);

//check();

osal\_start\_timerEx(GenericApp\_TaskID,SEND\_DATA\_EVENT,1500);//定时1200ms

return(events^SEND\_DATA\_EVENT);

}

//Discard unknown events

return 0;

}

uchar wen[6];

void GenericApp\_SendTheMessage( void )

{

afAddrType\_t my\_DstAddr;

my\_DstAddr.addrMode=(afAddrMode\_t)Addr16Bit;

my\_DstAddr.endPoint=GENERICAPP\_ENDPOINT;

my\_DstAddr.addr.shortAddr=0x0000;

DHT11();

wen[0]=wendu\_shi+0x30;

wen[1]=wendu\_ge%10+0x30;

wen[2]=':';

wen[3]= shidu\_shi+0x30;

wen[4]=shidu\_ge%10+0x30;

AF\_DataRequest( &my\_DstAddr, &GenericApp\_epDesc,

GENERICAPP\_CLUSTERID,

osal\_strlen(wen)+1,

wen,

//theMessageData,

&GenericApp\_TransID,

AF\_DISCV\_ROUTE, AF\_DEFAULT\_RADIUS);

//HalLedBlink(HAL\_LED\_1,0,50,500);

//HalLedBlink(HAL\_LED\_1,0,90,500);

}

void GenericApp\_MessageMSGCB( afIncomingMSGPacket\_t \*pkt )//消息处理

{

HalLedBlink(HAL\_LED\_1,0,50,500);

unsigned char buffer[2]={'0'};

switch ( pkt->clusterId )

{

case GENERICAPP\_CLUSTERID:

// "the" message

osal\_memcpy(buffer,pkt->cmd.Data,1);//将数据拷贝到缓冲区

if(buffer[0]=='2'){

//HalLedBlink(HAL\_LED\_1,0,50,500);

P1\_0=1;

P1\_1=0;

//HalLedBlink(HAL\_LED\_1,0,50,500);

//osal\_memcpy(theMessageData,"OFF",20);//将数据拷贝到缓冲区

}

else if(buffer[0]=='3'){

P1\_0=0;

P1\_1=0;

HalLedBlink(HAL\_LED\_1,0,50,500);

}

//HalUARTWrite(0,buffer,20);

break;

}

}

# **4系统测试**

## 4.1 硬件测试

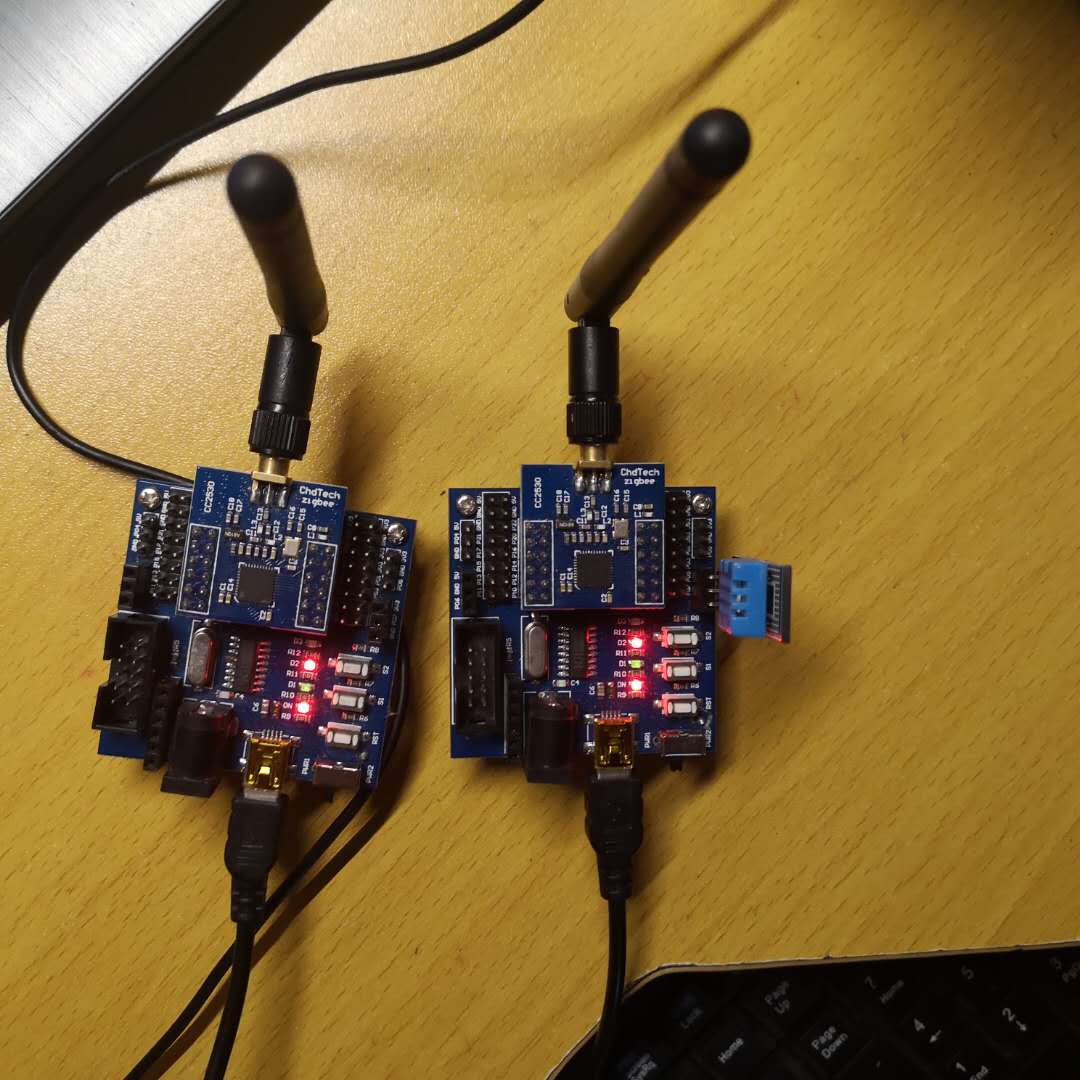


图4-1 协调器与终端进行通信

通过上图我们可以看到硬件模块成功调试并进行了通信，通信后所产生的温湿度数据通过串口发送到PC端

## 4.2 软件测试

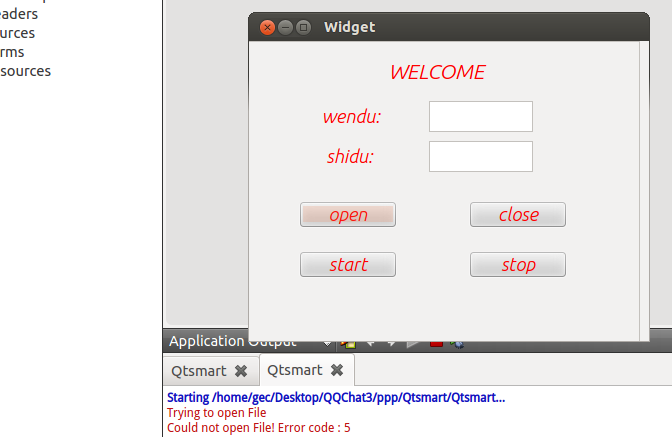


图4-2 接收数据界面



图4-3 数据库界面

经过多次试验测试，观察数据库的数据，可知系统基本符合周围环境的温度和湿度，检测结果基本准确，数据库表单等内容功能可用。

# **5 总结及展望**

## 5.1 总结

本文主要从设计需求、系统设计方案、技术方案以及系统实现，三个主要方面分析了基于Zigbee的温湿度检测系统的设计和实现，经测试得出以下总结：

1. 做项目时要系统地规划、认真的分析、要有明确的体系框架；
2. 作品的前期方案比技术更为重要，方案决定了一个设计的优劣，技术只是作为支撑；
3. 需要注意代码编写的标准性来增强软件的可读性，等；

经测试，系统框架完整，功能完善，扩展性强，功能可用。

## 5.2 展望

由于设备条件的限制，在测试过程中，检测的数据与周围环境的温湿度可能在精度、准确度上与真实结果有较大的误差，所以我希望本设计今后能进行更严谨的更准确的测试，并对其中的不足进行改进。

另外，在系统中有待改进的地方如下：

1. 使用自己设计的传感器模块代替购买的模块；
2. 增强系统功能；
3. 增加多个传感器进行检测；

**参考文献：**

**[1]唐瑜梅.基于Zigbee的温湿度数据采集与组网设计[J].电脑迷,2018(08):254-255.**

**[2]杨怀德.基于ZigBee的无线温湿度采集系统设计[J].电脑知识与技术,2016,12(14):229-231.**

**[3]邓呈轩. 基于ZigBee的远程无线温湿度测控系统的研究[D].兰州交通大学,2016.**