附录A 硬件电路图



附录B 程序

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

农业检测控制系统主程序

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

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "dht11.h"

#include "timer.h"

#include "usart3.h"

#include "oled.h"

#include "BH1750.h"

#include "gizwits\_product.h"

//名称替换

#define TScope currentDataPoint.valueTemp\_AlarmScope

#define HScope currentDataPoint.valueShidu\_AlarmScope

#define GQScope currentDataPoint.valueGQ\_AlarmScope

/\* 用户区当前设备状态结构体\*/

dataPoint\_t currentDataPoint;

//机智云协议初始化

void Gizwits\_Init(void)

{

TIM3\_Int\_Init(9,7199);//1MS系统定时

uart\_init(9600);//这里的波特率只能是9600

memset((uint8\_t\*)&currentDataPoint, 0, sizeof(dataPoint\_t));//设备状态结构体初始化

gizwitsInit();//缓冲区初始化

}

//数据采集、并在OLED显示数据、上报数据、判断是否触发警报

void userHandle(void)

{

static u16 GQ\_Data;

static u8 temp, hum;

if(timeuser > 1000)//每1秒上报一次数据

{

timeuser=0;//在.h文件里面已经声明过

//在oled和app上显示温湿度值

DHT11\_Read\_Data(&temp,&hum);//读取 DHT11 传感器

//x,y，第三位是数据位，长度，字体大小

OLED\_ShowNum(48,0,temp/10%10,1,16);

OLED\_ShowNum(56,0,temp%10,1,16);

OLED\_ShowNum(48,2,hum/10%10,1,16);

OLED\_ShowNum(56,2,hum%10,1,16);

//上传温湿度数据

currentDataPoint.valueTemp = temp ;

currentDataPoint.valueShidu = hum;

//在oled和app上显示光强数值

GQ\_Data = BH1750\_ReadContinuous1(); //读取GY-30传感器

OLED\_ShowNum(48,4,GQ\_Data/10000%10,1,16);

OLED\_ShowNum(56,4,GQ\_Data/1000%10,1,16);

OLED\_ShowNum(64,4,GQ\_Data/100%10,1,16);

OLED\_ShowNum(72,4,GQ\_Data/10%10,1,16);

OLED\_ShowNum(80,4,GQ\_Data%10,1,16);

currentDataPoint.valueGQ = GQ\_Data;//上传光强数值

if(0 == currentDataPoint.valueLED1)//判断led1发送数据

{

OLED\_ShowCHinese(48,6,13);

//判断温湿度以及光强是否超出给定值，若超出蜂鸣器响，led闪烁if((temp>=TScope||hum>=HScope||GQ\_Data>=GQScope)&&(TScope!=0&&HScope!=0&&GQScope!=0))

{

OLED\_ShowCHinese(0,6,17);

BEEP = 1;

LED0 = 0;

delay\_ms(500);

OLED\_ShowCHinese(0,6,18);

BEEP = 0;

LED0 = 1;

}

currentDataPoint.valueLED = currentDataPoint.valueBEEP = 0;

}

else

{

OLED\_ShowCHinese(48,6,11);

}

}

}

//主函数

int main(void)

{

delay\_init(); //延时函数初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设NVIC中断分组

DHT11\_Init();

OLED\_Init(); //初始化OLED

BH1750\_Init();

Gizwits\_Init(); //协议初始化

//OLED显示“温度”

OLED\_ShowCHinese(0,0,0); //x,y,第三位是汉字库的次序

OLED\_ShowCHinese(16,0,1);

OLED\_ShowCHinese(32,0,10);

OLED\_ShowCHinese(72,0,8);

//湿度

OLED\_ShowCHinese(0,2,2);

OLED\_ShowCHinese(16,2,3);

OLED\_ShowCHinese(32,2,10);

OLED\_ShowCHinese(72,2,9);

//光强

OLED\_ShowCHinese(0,4,4);

OLED\_ShowCHinese(16,4,6);

OLED\_ShowCHinese(32,4,10);

OLED\_ShowString(96,4,"lx",16);//这里lx是光强的单位

OLED\_ShowCHinese(48+16,6,14);

OLED\_ShowCHinese(48+32,6,15);

OLED\_ShowCHinese(48+48,6,16);

while(1)

{

userHandle();//用户采集函数

gizwitsHandle((dataPoint\_t \*)&currentDataPoint);//协议处理

}

}

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

DHT11程序——dht11.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "dht11.h"

#include "delay.h"

//复位DHT11

void DHT11\_Rst(void)

{

DHT11\_IO\_OUT(); //SET OUTPUT

DHT11\_DQ\_OUT=0; //拉低DQ

delay\_ms(20); //拉低至少18ms

DHT11\_DQ\_OUT=1; //DQ=1

delay\_us(30); //主机拉高20~40us

}

//等待DHT11的回应

u8 DHT11\_Check(void)

{

u8 reply=0;

DHT11\_IO\_IN();//SET INPUT

while (DHT11\_DQ\_IN&&reply<100)//DHT11会拉低40~80us

{

reply++;

delay\_us(1);

};

if(reply>=100)return 1;

else reply=0;

while (!DHT11\_DQ\_IN&&reply<100)//DHT11拉低后会再次拉高40~80us

{

reply++;

delay\_us(1);

};

if(reply>=100)return 1;

return 0;

}

//从DHT11读取一个位

u8 DHT11\_Read\_Bit(void)

{

u8 reply=0;

while(DHT11\_DQ\_IN&&reply<100)//等待变为低电平

{

reply++;

delay\_us(1);

}

reply=0;

while(!DHT11\_DQ\_IN&&reply<100)//等待变高电平

{

reply++;

delay\_us(1);

}

delay\_us(40);//等待40us

if(DHT11\_DQ\_IN)return 1;

else return 0;

}

//从DHT11读取一个字节

u8 DHT11\_Read\_Byte(void)

{

u8 i,dat;

dat=0;

for (i=0; i<8; i++)

{

dat<<=1;

dat|=DHT11\_Read\_Bit();

}

return dat;

}

//从DHT11读取一次数据

u8 DHT11\_Read\_Data(u8 \*temp,u8 \*humi)

{

u8 buf[5];

u8 i;

DHT11\_Rst();

if(DHT11\_Check()==0)

{

for(i=0; i<5; i++) //读取40位数据

{

buf[i]=DHT11\_Read\_Byte();

}

if((buf[0]+buf[1]+buf[2]+buf[3])==buf[4])

{

\*humi=buf[0];

\*temp=buf[2];

}

} else return 1;

return 0;

}

//初始化DHT11的IO口 DQ 同时检测DHT11的存在

u8 DHT11\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

//使能PB,PE端口时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

DHT11\_Rst();

return DHT11\_Check();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DHT11程序——dht11.h

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

#ifndef \_\_DHT11\_H

#define \_\_DHT11\_H

#include "sys.h"

//IO方向设置

//#define DHT11\_IO\_IN() {GPIOA->CRH&=0XFFFF0FFF;GPIOA->CRH|=8<<12;}

//#define DHT11\_IO\_OUT() {GPIOA->CRH&=0XFFFF0FFF;GPIOA->CRH|=3<<12;}

//这里是高八位设置，若是低八位改成L即可

#define DHT11\_IO\_IN() {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=8<<4;}

#define DHT11\_IO\_OUT() {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=3<<4;}

////IO操作函数

#define DHT11\_DQ\_OUT PBout(9) //数据端口 PB9

#define DHT11\_DQ\_IN PBin(9) //数据端口 PB9

u8 DHT11\_Init(void); //初始化DHT11

u8 DHT11\_Read\_Data(u8 \*temp,u8 \*humi);//读取温湿度

u8 DHT11\_Read\_Byte(void); //读出一个字节

u8 DHT11\_Read\_Bit(void); //读出一个位

u8 DHT11\_Check(void); //检测是否存在DHT11

void DHT11\_Rst(void); //复位DHT11

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GY-30程序——BH1750.c

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

#include "BH1750.h"

#include "delay.h"

#include "sys.h"

#include "stm32f10x.h"

#include "stdint.h"

u8 BUF[8];

u16 temp2=0;

static u32 lux=0;

/\*\*\*\*向BH1750单次写命令\*\*\*\*/

void Single\_Write\_BH1750(unsigned char Reg\_Address)

{

IIC\_Start1();

IIC\_Send\_Byte(0x46); //发送器件地址0100 0110 最后一位0，表示写

IIC\_Send\_Byte(Reg\_Address);

IIC\_Stop1();

}

/\*\*\*\*BH1750初始化\*\*\*\*/

void BH1750\_Init(void)

{

IIC\_Config();

Single\_Write\_BH1750(0x01);

}

//从BH1750单次读lx

void BH1750\_ReadOneTime(void)

{

IIC\_Start1();

IIC\_Send\_Byte(0x46); //发送器件地址0100 0110 最后一位0，表示写

IIC\_Wait\_Ack1();

IIC\_Send\_Byte(0x20);

IIC\_Wait\_Ack1();

IIC\_Stop1();

delay\_ms(200);

IIC\_Start1();

IIC\_Send\_Byte(0x47);

IIC\_Wait\_Ack1();

BUF[0]=IIC\_Read\_Byte(1);

BUF[1]=IIC\_Read\_Byte(0);

IIC\_Stop1();

}

//从BH1750连续读lx

u16 BH1750\_ReadContinuous1(void)

{

u16 temp=0,temp1=0;

IIC\_Start1();

IIC\_Send\_Byte(0x46); //发送器件地址0100 0110 最后一位0，表示写

IIC\_Wait\_Ack1();

IIC\_Send\_Byte(0x10);

IIC\_Wait\_Ack1();

IIC\_Stop1();

delay\_ms(200);

IIC\_Start1();

IIC\_Send\_Byte(0x47);

IIC\_Wait\_Ack1();

temp=IIC\_Read\_Byte(1);

temp1=IIC\_Read\_Byte(0);

IIC\_Stop1();

temp2=temp1+(temp<<8);

lux=temp2;

return lux;

}

void Conversion(void)

{

temp2=BUF[0];

temp2=(temp2<<8)+BUF[1];

lux=(float)temp2/1.2; //lux是float型

}

void IIC\_Config(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE);

GPIO\_InitStructure.GPIO\_Pin=GPIO\_Pin\_10|GPIO\_Pin\_11;

GPIO\_InitStructure.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_Init(GPIOB,&GPIO\_InitStructure);

GPIO\_SetBits(GPIOB,GPIO\_Pin\_10|GPIO\_Pin\_11); //将总线拉高释放总线

}

//总线初始化

void IIC\_Start1(void)

{

SDA\_OUT() //设置SDA线为输出

//在开始数据传输前，先让SDA和SCL都拉高

IIC\_SDA=1; //发送起始条件的数据信号，释放总线

delay\_us(2);

IIC\_SCL=1;

delay\_us(5); //Tsu;STA：起始条件的建立时间大于4.7us。

IIC\_SDA=0; //SDA由高变为低表示开始信号

delay\_us(4); //起始条件的保持时间大于4us

IIC\_SCL=0; //准备发送或者接收数据

delay\_us(2);

}

//停止信号

void IIC\_Stop1(void)

{

SDA\_OUT() //设置SDA线为输出

IIC\_SDA=0; //发送停止信号的数据信号

delay\_us(2);

IIC\_SCL=1; //发送停止信号的时钟信号

delay\_us(5); //停止信号的建立时间大于4us

IIC\_SDA=1; //发送停止信号

delay\_us(4); //

}

//等待应答信号到来

u8 IIC\_Wait\_Ack1(void)

{

u8 ucErrorTime=0;

SDA\_IN();

IIC\_SDA=1;

delay\_us(2);

IIC\_SCL=1;

delay\_us(2);

while(READ\_SDA)

{

ucErrorTime++;

if(ucErrorTime>=250)

{

IIC\_Stop1();

return 1;

}

}

IIC\_SCL=0;

return 0;

}

//产生Ack应答

void IIC\_Ack(void)

{

IIC\_SCL=0;

SDA\_OUT();

IIC\_SDA=0; //主器件应答

delay\_us(2);

IIC\_SCL=1;

delay\_us(4); //SCL高电平周期大于4us

IIC\_SCL=0; //清时钟线，钳住IIC总线以便继续接收

delay\_us(2);

}

//不产生Ack应答

void IIC\_NAck(void)

{

IIC\_SCL=0;

SDA\_OUT();

IIC\_SDA=1; //主器件不应答

delay\_us(2);

IIC\_SCL=1;

delay\_us(4); //SCL高电平周期大于4us

IIC\_SCL=0; //清时钟线，钳住IIC总线以便继续接收

delay\_us(2);

}

//发送一个字节

void IIC\_Send\_Byte(unsigned char c) //要传送的数据长度为8位

{

u8 i;

SDA\_OUT();

for(i=0; i<8; i++)

{

if((c<<i)&0x80) //判断发送位

IIC\_SDA=1;

else IIC\_SDA=0;

delay\_us(2);

IIC\_SCL=1; //拉高SCL，通知被控器开始接收数据位

delay\_us(4); //保证SCL高电平周期大于4us

IIC\_SCL=0; //拉低SCL，允许SDA传输下一位数据。

delay\_us(2);

}

IIC\_SCL=0; //拉低SCL，为下次数据传输做好准备

delay\_us(2);

}

//接收一个字节

u8 IIC\_Read\_Byte(unsigned char ack)

{

unsigned char i,receive=0;

SDA\_IN();

for(i=0; i<8; i++)

{

IIC\_SCL=0; //置SCL为低，准备接收数据位

delay\_us(5); //时钟低电平周期大于4.7us

IIC\_SCL=1; //置SCL为高，使SDA上数据有效

delay\_us(2);

receive=receive<<1;

//读取SDA，把接收的数据位放入receve中

if(READ\_SDA) receive=receive+1;

delay\_us(2);

}

if(!ack) IIC\_NAck(); //发送NAck

else IIC\_Ack(); //发送Ack

return receive;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GY-30程序——BH1750.h

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

#ifndef \_\_BH1750\_H

#define \_\_BH1750\_H

#include "sys.h"

//IO操作函数

#define IIC\_SCL PBout(10) //SCL

#define IIC\_SDA PBout(11) //SDA

#define READ\_SDA PBin(11) //输入SDA

//IO方向设置，设置输入还是输出，半双工就这样

#define SDA\_IN() {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=8<<12;}

#define SDA\_OUT() {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=3<<12;}

void IIC\_Config(void);

void IIC\_Init(void) ;

void IIC\_Start1(void); //起始信号

void IIC\_Stop1(void); //停止信号

u8 IIC\_Wait\_Ack1(void);

void IIC\_Ack(void);

void IIC\_NAck(void);

void IIC\_Send\_Byte(unsigned char c); //要传送的数据长度为8位

u8 IIC\_Read\_Byte(unsigned char ack);

void Single\_Write\_BH1750(unsigned char Reg\_Address);

void BH1750\_Init(void);

void BH1750\_ReadOneTime(void);

u16 BH1750\_ReadContinuous1(void);

void BH1750\_ReadContinuous2(void);

void BH1750\_Multiple\_Read(void);

void Conversion(void);

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OLED程序——oled.c

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

#include "oled.h"

#include "stdlib.h"

#include "codetab.h"

#include "delay.h"

//IIC Start

void IIC\_Start()

{

OLED\_SCLK\_Set();

OLED\_SDIN\_Set();

OLED\_SDIN\_Clr();

OLED\_SCLK\_Clr();//注意一下

}

//IIC Stop

void IIC\_Stop()

{

OLED\_SCLK\_Set() ;

OLED\_SDIN\_Clr();

OLED\_SDIN\_Set();

}

void IIC\_Wait\_Ack()

{

OLED\_SCLK\_Set() ;

OLED\_SCLK\_Clr();

}

// IIC Write byte

void Write\_IIC\_Byte(unsigned char IIC\_Byte)

{

unsigned char i;

unsigned char m,da;

da=IIC\_Byte;

OLED\_SCLK\_Clr();

for(i=0; i<8; i++)

{

m=da;

m=m&0x80;

if(m==0x80)

{

OLED\_SDIN\_Set();

}

else OLED\_SDIN\_Clr();

da=da<<1;

OLED\_SCLK\_Set();

OLED\_SCLK\_Clr();

}

}

// IIC Write Command

void Write\_IIC\_Command(unsigned char IIC\_Command)

{

IIC\_Start();

Write\_IIC\_Byte(0x78); //Slave address,SA0=0

IIC\_Wait\_Ack();

Write\_IIC\_Byte(0x00); //write command

IIC\_Wait\_Ack();

Write\_IIC\_Byte(IIC\_Command);

IIC\_Wait\_Ack();

IIC\_Stop();

}

// IIC Write Data

void Write\_IIC\_Data(unsigned char IIC\_Data)

{

IIC\_Start();

Write\_IIC\_Byte(0x78); //D/C#=0; R/W#=0

IIC\_Wait\_Ack();

Write\_IIC\_Byte(0x40); //write data

IIC\_Wait\_Ack();

Write\_IIC\_Byte(IIC\_Data);

IIC\_Wait\_Ack();

IIC\_Stop();

}

void OLED\_WR\_Byte(unsigned dat,unsigned cmd)

{

if(cmd)

{

Write\_IIC\_Data(dat);

}

else

{

Write\_IIC\_Command(dat);

}

}

//坐标设置

void OLED\_Set\_Pos(unsigned char x, unsigned char y)

{

OLED\_WR\_Byte(0xb0+y,OLED\_CMD);

OLED\_WR\_Byte(((x&0xf0)>>4)|0x10,OLED\_CMD);

OLED\_WR\_Byte((x&0x0f),OLED\_CMD);

}

//清屏函数,清完屏,整个屏幕是黑色的

void OLED\_Clear(void)

{

u8 i,n;

for(i=0; i<8; i++)

{

OLED\_WR\_Byte (0xb0+i,OLED\_CMD); //设置页地址（0~7）

OLED\_WR\_Byte (0x00,OLED\_CMD); //设置显示位置—列低地址

OLED\_WR\_Byte (0x10,OLED\_CMD); //设置显示位置—列高地址

for(n=0; n<128; n++)OLED\_WR\_Byte(0,OLED\_DATA);

} //更新显示

}

//在指定位置显示一个字符,包括部分字符

//x:0~127

//y:0~8

//mode:0,反白显示;1,正常显示

//size:选择字体 16/12

void OLED\_ShowChar(u8 x,u8 y,u8 chr,u8 Char\_Size)

{

unsigned char c=0,i=0;

c=chr-' ';//得到偏移后的值

if(x>128-1) {

x=0;

y=y+2;

}

if(Char\_Size ==16)

{

OLED\_Set\_Pos(x,y);

for(i=0; i<8; i++)

OLED\_WR\_Byte(F8X16[c\*16+i],OLED\_DATA);

OLED\_Set\_Pos(x,y+1);

for(i=0; i<8; i++)

OLED\_WR\_Byte(F8X16[c\*16+i+8],OLED\_DATA);

}

else {

OLED\_Set\_Pos(x,y);

for(i=0; i<6; i++)

OLED\_WR\_Byte(F6x8[c][i],OLED\_DATA);

}

}

//m^n函数

u32 oled\_pow(u8 m,u8 n)

{

u32 result=1;

while(n--)

result\*=m;

return result;

}

//显示2个数字

void OLED\_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size2)

{

u8 t,temp;

u8 enshow=0;

for(t=0; t<len; t++)

{

temp=(num/oled\_pow(10,len-t-1))%10;

if(enshow==0&&t<(len-1))

{

if(temp==0)

{

OLED\_ShowChar(x+(size2/2)\*t,y,' ',size2);

continue;

} else enshow=1;

}

OLED\_ShowChar(x+(size2/2)\*t,y,temp+'0',size2);

}

}

//显示一个字符号串

void OLED\_ShowString(u8 x,u8 y,u8 \*chr,u8 Char\_Size)

{

unsigned char j=0;

while (chr[j]!='\0')

{ OLED\_ShowChar(x,y,chr[j],Char\_Size);

x+=8;

if(x>120) {

x=0;

y+=2;

}

j++;

}

}

//显示汉字

void OLED\_ShowCHinese(u8 x,u8 y,u8 no)

{

u8 t,adder=0;

OLED\_Set\_Pos(x,y);

for(t=0; t<16; t++)

{

OLED\_WR\_Byte(Hzk[2\*no][t],OLED\_DATA);

adder+=1;

}

OLED\_Set\_Pos(x,y+1);

for(t=0; t<16; t++)

{

OLED\_WR\_Byte(Hzk[2\*no+1][t],OLED\_DATA);

adder+=1;

}

}

//初始化SSD1306

void OLED\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructer;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

GPIO\_InitStructer.GPIO\_Pin=GPIO\_Pin\_6|GPIO\_Pin\_7; //6--SCL 7--SDA

GPIO\_InitStructer.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_InitStructer.GPIO\_Mode=GPIO\_Mode\_Out\_PP;

GPIO\_Init(GPIOB, &GPIO\_InitStructer);

delay\_ms(200);

OLED\_WR\_Byte(0xAE,OLED\_CMD);//--display off

OLED\_WR\_Byte(0x00,OLED\_CMD);//---set low column address

OLED\_WR\_Byte(0x10,OLED\_CMD);//---set high column address

OLED\_WR\_Byte(0x40,OLED\_CMD);//--set start line address

OLED\_WR\_Byte(0xB0,OLED\_CMD);//--set page address

OLED\_WR\_Byte(0x81,OLED\_CMD); // contract control

OLED\_WR\_Byte(0xFF,OLED\_CMD);//--128

OLED\_WR\_Byte(0xA1,OLED\_CMD);//set segment remap

OLED\_WR\_Byte(0xA6,OLED\_CMD);//--normal / reverse

OLED\_WR\_Byte(0xA8,OLED\_CMD);//--set multiplex ratio(1 to 64)

OLED\_WR\_Byte(0x3F,OLED\_CMD);//--1/32 duty

OLED\_WR\_Byte(0xC8,OLED\_CMD);//Com scan direction

OLED\_WR\_Byte(0xD3,OLED\_CMD);//-set display offset

OLED\_WR\_Byte(0x00,OLED\_CMD);//

OLED\_WR\_Byte(0xD5,OLED\_CMD);//set osc division

OLED\_WR\_Byte(0x80,OLED\_CMD);//

OLED\_WR\_Byte(0xD8,OLED\_CMD);//set area color mode off

OLED\_WR\_Byte(0x05,OLED\_CMD);//

OLED\_WR\_Byte(0xD9,OLED\_CMD);//Set Pre-Charge Period

OLED\_WR\_Byte(0xF1,OLED\_CMD);//

OLED\_WR\_Byte(0xDA,OLED\_CMD);//set com pin configuartion

OLED\_WR\_Byte(0x12,OLED\_CMD);//

OLED\_WR\_Byte(0xDB,OLED\_CMD);//set Vcomh

OLED\_WR\_Byte(0x30,OLED\_CMD);//

OLED\_WR\_Byte(0x8D,OLED\_CMD);//set charge pump enable

OLED\_WR\_Byte(0x14,OLED\_CMD);//

OLED\_WR\_Byte(0xAF,OLED\_CMD);//--turn on oled panel

OLED\_Clear();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OLED程序——oled.h

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

#ifndef \_\_OLED\_H

#define \_\_OLED\_H

#include "sys.h"

#include "stdlib.h"

//-----------------OLED IIC端口定义----------------

#define OLED\_SCLK\_Clr() GPIO\_ResetBits(GPIOB,GPIO\_Pin\_6 )//SCL

#define OLED\_SCLK\_Set() GPIO\_SetBits(GPIOB,GPIO\_Pin\_6)

#define OLED\_SDIN\_Clr() GPIO\_ResetBits(GPIOB,GPIO\_Pin\_7)//SDA

#define OLED\_SDIN\_Set() GPIO\_SetBits(GPIOB,GPIO\_Pin\_7)

#define OLED\_CMD 0 //写命令

#define OLED\_DATA 1 //写数据

//OLED控制用函数

void OLED\_WR\_Byte(unsigned dat,unsigned cmd);

void OLED\_Display\_On(void);

void OLED\_Display\_Off(void);

void OLED\_Init(void);

void OLED\_Clear(void);

void OLED\_DrawPoint(u8 x,u8 y,u8 t);

void OLED\_Fill(u8 x1,u8 y1,u8 x2,u8 y2,u8 dot);

void OLED\_ShowChar(u8 x,u8 y,u8 chr,u8 Char\_Size);

void OLED\_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size);

void OLED\_ShowString(u8 x,u8 y, u8 \*p,u8 Char\_Size);

void OLED\_Set\_Pos(unsigned char x, unsigned char y);

void OLED\_ShowCHinese(u8 x,u8 y,u8 no);

void OLED\_DrawBMP(unsigned char x0, unsigned char y0,unsigned char x1, unsigned char y1,unsigned char BMP[]);

void Delay\_50ms(unsigned int Del\_50ms);

void Delay\_1ms(unsigned int Del\_1ms);

void fill\_picture(unsigned char fill\_Data);

void Picture(u8 i);

void IIC\_Start(void);

void IIC\_Stop(void);

void Write\_IIC\_Command(unsigned char IIC\_Command);

void Write\_IIC\_Data(unsigned char IIC\_Data);

void Write\_IIC\_Byte(unsigned char IIC\_Byte);

void IIC\_Wait\_Ack(void);

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ESP8266程序——gizwits\_product.c

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

#include "stdio.h"

#include "string.h"

#include "gizwits\_product.h"

#include "common.h"

#include "usart3.h"

#include "led.h"

#include "beep.h"

#include "usart.h"

//定义变量

static uint32\_t timerMsCount;

uint8\_t aRxBuffer;

uint16\_t timeuser = 0;

extern dataPoint\_t currentDataPoint;

//在这里控制蜂鸣器等

int8\_t gizwitsEventProcess(eventInfo\_t \*info, uint8\_t \*gizdata, uint32\_t len)

{

uint8\_t i = 0;

dataPoint\_t \*dataPointPtr = (dataPoint\_t \*)gizdata;

moduleStatusInfo\_t \*wifiData = (moduleStatusInfo\_t \*)gizdata;

protocolTime\_t \*ptime = (protocolTime\_t \*)gizdata;

#if MODULE\_TYPE

gprsInfo\_t \*gprsInfoData = (gprsInfo\_t \*)gizdata;

#else

moduleInfo\_t \*ptModuleInfo = (moduleInfo\_t \*)gizdata;

#endif

if((NULL == info) || (NULL == gizdata))

{

return -1;

}

for(i=0; i<info->num; i++)

{

switch(info->event[i])

{

case EVENT\_LED:

currentDataPoint.valueLED = dataPointPtr->valueLED;

GIZWITS\_LOG("Evt: LED%d\n", currentDataPoint.valueLED);

if(0x01 == currentDataPoint.valueLED)

{

if(1 == currentDataPoint.valueLED1)

LED0 = 0;

}

else

{

if(1 == currentDataPoint.valueLED1)

LED0 = 1;

}

break;

case EVENT\_BEEP:

currentDataPoint.valueBEEP = dataPointPtr->valueBEEP;

GIZWITS\_LOG("Evt: BEEP %d \n", currentDataPoint.valueBEEP);

if(0x01 == currentDataPoint.valueBEEP)

{

if(1 == currentDataPoint.valueLED1)

BEEP = 1;

}

else

{

if(1 == currentDataPoint.valueLED1)

BEEP = 0;

}

break;

case EVENT\_LED1:

currentDataPoint.valueLED1 = dataPointPtr->valueLED1;

GIZWITS\_LOG("Evt: LED1 %d \n", currentDataPoint.valueLED1);

if(0x01 == currentDataPoint.valueLED1)

{

LED2 = 0;

}

else

{

LED2 = 1;

}

break;

case EVENT\_Temp\_AlarmScope:

currentDataPoint.valueTemp\_AlarmScope = dataPointPtr->valueTemp\_AlarmScope;

GIZWITS\_LOG("Evt:EVENT\_Temp\_AlarmScope %d\n",currentDataPoint.valueTemp\_AlarmScope);

break;

case EVENT\_Shidu\_AlarmScope:

currentDataPoint.valueShidu\_AlarmScope = dataPointPtr->valueShidu\_AlarmScope;

GIZWITS\_LOG("Evt:EVENT\_Shidu\_AlarmScope %d\n",currentDataPoint.valueShidu\_AlarmScope);

break;

case EVENT\_GQ\_AlarmScope:

currentDataPoint.valueGQ\_AlarmScope = dataPointPtr->valueGQ\_AlarmScope;

GIZWITS\_LOG("Evt:EVENT\_GQ\_AlarmScope %d\n",currentDataPoint.valueGQ\_AlarmScope);

break;

case WIFI\_SOFTAP:

break;

case WIFI\_AIRLINK:

break;

case WIFI\_STATION:

break;

case WIFI\_CON\_ROUTER:

break;

case WIFI\_DISCON\_ROUTER:

break;

case WIFI\_CON\_M2M:

break;

case WIFI\_DISCON\_M2M:

break;

case WIFI\_RSSI:

GIZWITS\_LOG("RSSI %d\n", wifiData->rssi);

break;

case TRANSPARENT\_DATA:

GIZWITS\_LOG("TRANSPARENT\_DATA \n");

break;

case WIFI\_NTP:

GIZWITS\_LOG("WIFI\_NTP : [%d-%d-%d %02d:%02d:%02d][%d] \n",ptime->year,ptime->month,ptime->day,ptime->hour,ptime->minute,ptime->second,ptime->ntp);

break;

case MODULE\_INFO:

GIZWITS\_LOG("MODULE INFO ...\n");

#if MODULE\_TYPE

GIZWITS\_LOG("GPRS MODULE ...\n");

#else

GIZWITS\_LOG("WIF MODULE ...\n");

GIZWITS\_LOG("Type : [%d] \n",ptModuleInfo->moduleType);

#endif

break;

default:

break;

}

}

return 0;

}

void userInit(void)

{

memset((uint8\_t\*)&currentDataPoint, 0, sizeof(dataPoint\_t));

}

void gizTimerMs(void)

{

timerMsCount++;

timeuser++;

}

uint32\_t gizGetTimerCount(void)

{

return timerMsCount;

}

void mcuRestart(void)

{

\_\_set\_FAULTMASK(1);

NVIC\_SystemReset();

}

//发送给云服务器

int32\_t uartWrite(uint8\_t \*buf, uint32\_t len)

{

uint32\_t i = 0;

if(NULL == buf)

{

return -1;

}

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

{

USART\_SendData(USART1,buf[i]);

//循环发送,直到发送完毕

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)==RESET);

if(i >=2 && buf[i] == 0xFF)

{

USART\_SendData(USART1,0x55);

//循环发送,直到发送完毕

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)==RESET);

}

}

#ifdef PROTOCOL\_DEBUG

GIZWITS\_LOG("MCU2WiFi[%4d:%4d]: ", gizGetTimerCount(), len);

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

{

GIZWITS\_LOG("%02x ", buf[i]);

if(i >=2 && buf[i] == 0xFF)

{

GIZWITS\_LOG("%02x ", 0x55);

}

}

GIZWITS\_LOG("\n");

#endif

return len;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ESP8266程序——gizwits\_product.h

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

#ifndef \_GIZWITS\_PRODUCT\_H

#define \_GIZWITS\_PRODUCT\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

#include <stdint.h>

#include <stm32f10x.h>

#include "gizwits\_protocol.h"

extern uint16\_t timeuser;

#define SOFTWARE\_VERSION "03030000"

#define HARDWARE\_VERSION "03010100"

#define MODULE\_TYPE 0 //0,WIFI ;1,GPRS

extern dataPoint\_t currentDataPoint;

void gizTimerMs(void);

void timerInit(void);

void uartInit(void);

void userInit(void);

void userHandle(void);

void mcuRestart(void);

uint32\_t gizGetTimerCount(void);

int32\_t uartWrite(uint8\_t \*buf, uint32\_t len);

int8\_t gizwitsEventProcess(eventInfo\_t \*info, uint8\_t \*data, uint32\_t len);

#ifdef \_\_cplusplus

}

#endif

#endif