移动端温度控制设计

年级专业：16 级软件工程

方向：嵌入式软件与系统

组员：宋晓彤 16340192

苏依晴 16340196

熊秭鉴 16340259

叶梓豪 16340277

张金亮 16340288

完成日期：2018/12/7

【设计思路】

通过使用单片机上的热阻温度检测模块对温度进行检测，然后通过单片机数码管显示；内置变量模拟温度设定，可通过手机端/单片机按钮模块进行调整，温度达到设定目标或倒计时走完后单片机进行提示；手机端与单片机通过 wifi 模块进行连接；外接液晶屏显示日期等信息。

【实验功能概述】

1. 使用单片机上的热感电阻进行室温检测
2. 扫描单片机键盘输入，通过单片机通信设定温度
3. 手机端使用 wifi 模块连接单片机，远程设定温度
4. 液晶屏显示日期等信息
5. 数码管切换显示【当前温度，设定温度】以及倒计时信息
6. 当倒计时结束/温度达到设定温度时进行 LED 灯闪烁提示

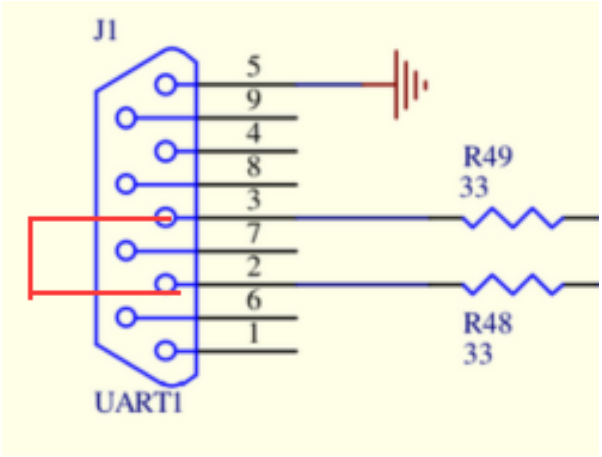
【主要知识点应用及拓展】

1. 单片机通信
2. RTC 时钟
3. 键盘扫描
4. 温度模块、wifi 模块、液晶屏模块、LED 灯模块应用

一、物理逻辑

【串口连接】

由于使用是单台单片机进行，因此需要将单片机上的 UART 中的发送串口（TX，图中 3）和接收串口（RX,图中 2）连接起来

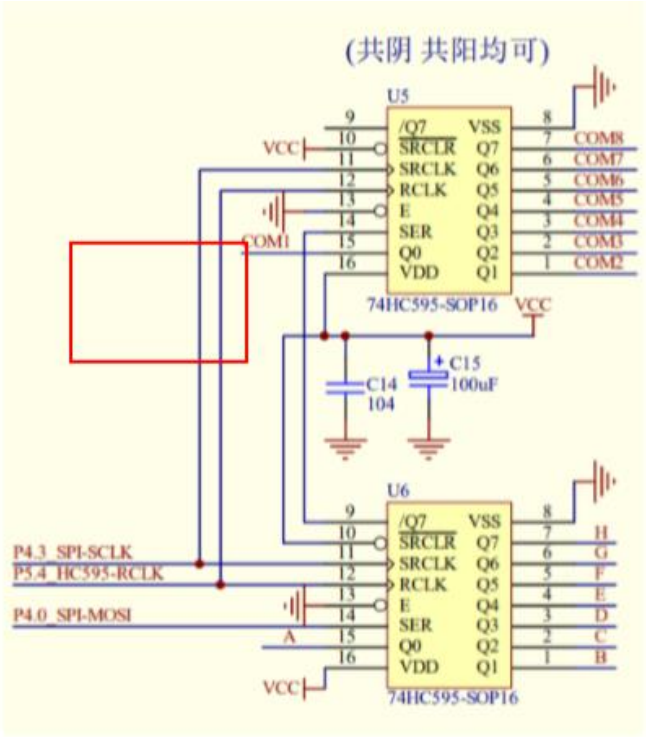


【将串口数据通过 USB 输出】

将实验板的 P16 与 P30、P17 与 P31 分别连接，使得串口 1 收发的数据可以转换到烧录使用的 USB 口，使得通信可以使用电脑端的串口助手进行观察。

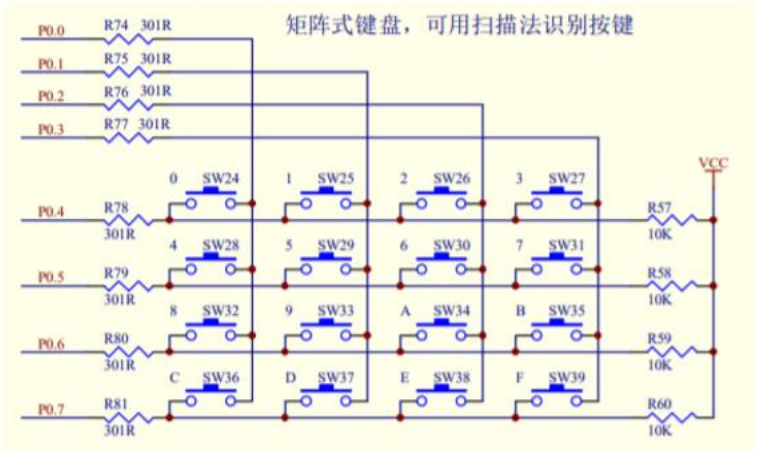
【数码管显示部分】

数码管显示时分秒，使用按键可以调节时和分，连接如图：



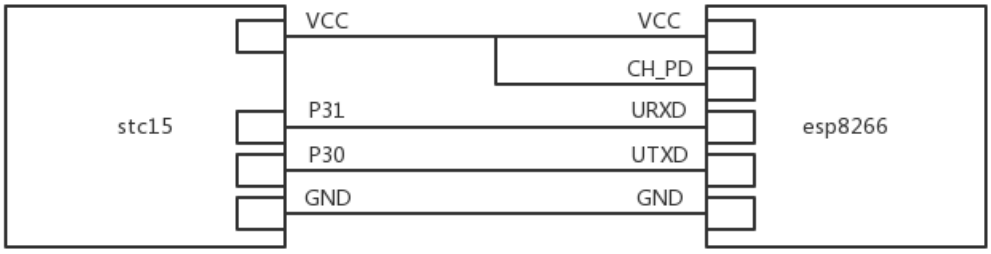
【按键部分】

按键部分使用扫描法识别按键的按下，并根据按键来进行对应的操作



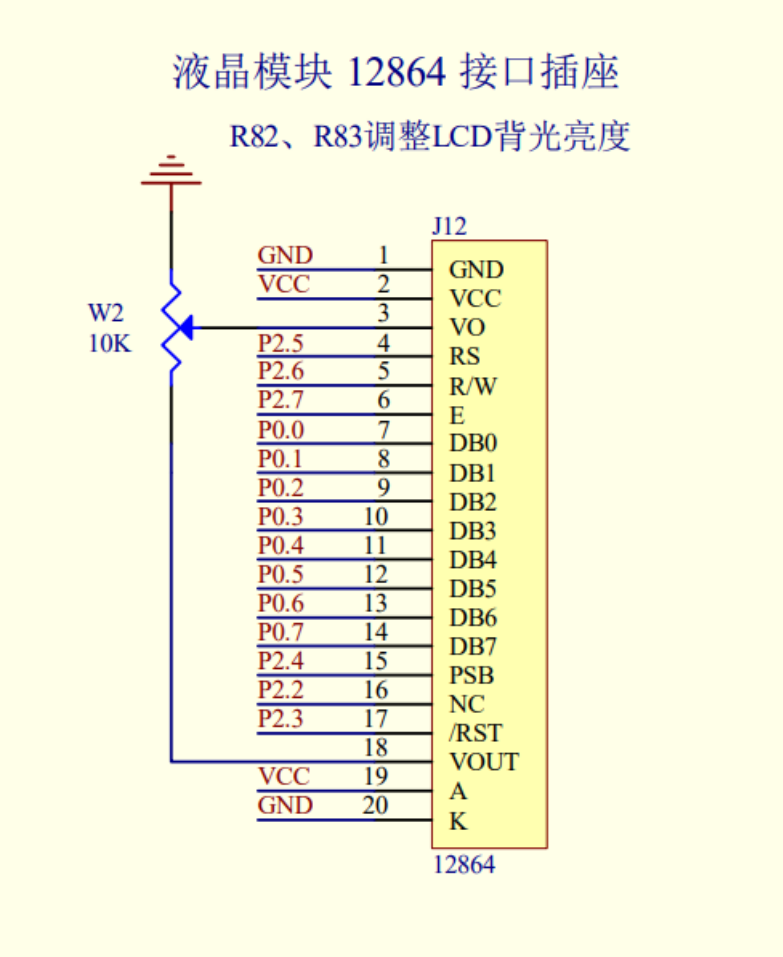
【wifi 模块部分】

电路图（本来 esp8266 的 VCC 和 CH_PD 应该接 3.3V 的，不过没在我们板上找到 5V 转 3.3V 的，就先直接插 VCC。）



【液晶屏模块部分】

这是参考资料中给出的电路示意图，我们的 0.96 寸 OLED 显示屏需要和这个进行连接，然后我们需要在 Keil 上实现代码对显示屏进行显示控制。



我们所购买的 OLED 显示屏是 7 线的，上面分别是 GND，VCC，D0，D1，RES，DC，CS

GND 和 VCC 分别连接对应的接口就行，D0 对应时钟，D1 对应数据，RES 复位，剩下的 DC 是数据命令选通端。

【手机端部分】

前端：使用 html+css+js 实现前端界面的搭建。

通讯：使用 websoket 与 Wi-Fi 模块进行通讯。

二、软件实现及原理分析

【功能原理】

单片机启动时先初始化串口、按键等数据，然后进入循环：

1.检测是否收到新的设定温度数据/新的检测温度并进行处理。

2.按照周期刷新数码管的显示

3.按照周期扫描行列键盘，判断是否有按键操作；如果有键按下，则通过串口发送按键的代码

4.接收方接收到按键码放入缓存 RX1_Buffer，回到步骤 1

【使用方法】

单片机端：

按下所有按键都会通过串口发送键位的号码，但只有 0~3 会被使用

按下单片机行列键盘的 0：倒计时设定的小时+1

按下单片机行列键盘的 1：倒计时设定的小时-1

按下单片机行列键盘的 2：倒计时设定的分钟+1

按下单片机行列键盘的 3：倒计时设定的分钟-1

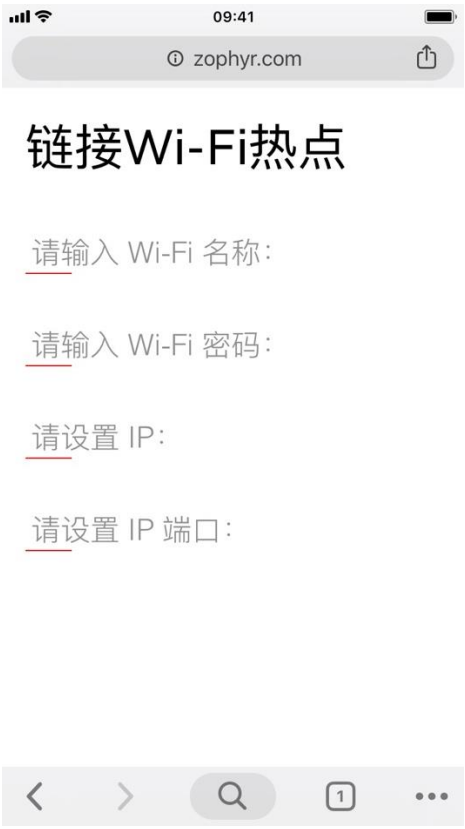
按下单片机行列键盘的 4：将倒计时重置

按下单片机行列键盘的 7：进入倒计时设定功能，此时数码管切换显示停止，时钟倒计时停止

按下单片机行列键盘的 11：完成倒计时设定，数码管重启切换显示，倒计时开始

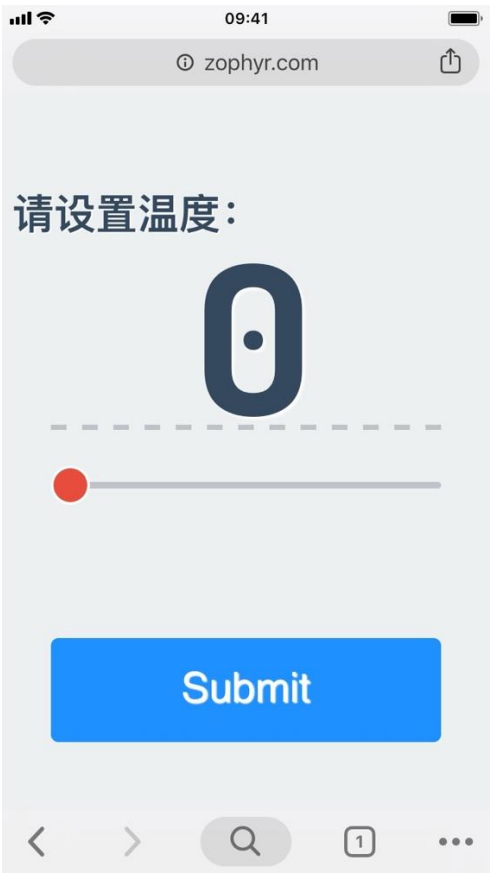
手机端：

1. 打开网页

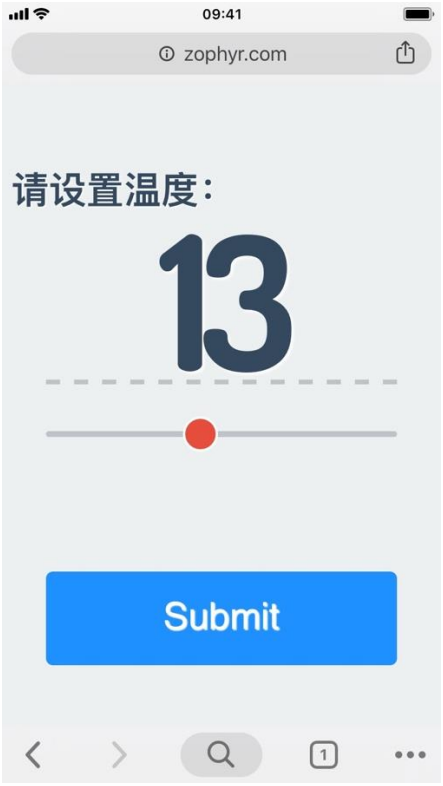


2. 输入相应的 Wi-Fi 名称、密码、设置访问 IP、IP 监听端口

3. 点击链接，进入温控界面。



4. 拖动滑块，调整设置的温度。



5. 点击‘submit’提交设置的温度。

【wifi 端】

设置接入点名称为 esp8266，密码为 0123456789，通道号为 11，加密模式 3（WPA2_PSK）

```
AT+CWMODE=2

OK
AT+CWSAP="esp8266","0123456789",11,3

OK
```

模块的 ip 为 192.168.4.1

```
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"86:f3:eb:31:b3:78"
```

实现效果:

【具体功能实现】

1.通信功能（单片机串口）实现

```
//----------------按键扫描模块------------------//
//----------------通信模块------------------------//
if((TX1_Cnt != RX1_Cnt) && (!B_TX1_Busy))  //接受到通信
    {
        //RX1_Buffer 接受数据缓冲
        //收到数据，并将数据中的键码赋给变量remoteKeyCode
        remoteKeyCode = RX1_Buffer[TX1_Cnt];
        if(++TX1_Cnt >= UART1_BUF_LENGTH)   TX1_Cnt = 0;  //UART1-BU
    }

//-----------发送方-----------//
if(KeyCode != 0) //keyCode不等于零，即有按键被按下，需要发送一个指令出去
{
  send_buffer[0] = KeyCode;
  send_buffer[1] = 0;
  PrintString1(send_buffer);
  KeyCode = 0;
}

//------------接收方----------//
if(remoteKeyCode != 0) //接收方接受到有键按下
```

2. 键盘扫描模块（使用 XY 查找 4x4 键的方法）

```
void    IO_KeyScan(void)    //50ms call
```

3.LED 显示模块（设定温度）

```
//------------------显示部分--------------//

++SleepDelay;
if(if_change && SleepDelay % 2 == 0){
//if(SleepDelay % 2 == 0){
  //设定温度
  LED8[2] = Target_tem % 10;
  LED8[1] = Target_tem % 100 / 10;
  LED8[0] = Target_tem / 100;
```

4.实际温度检测并显示

```
j = Get_ADC10bitResult(3);  //参数0~7,查询方式做一次ADC, 返回值就
j += Get_ADC10bitResult(3);
j += Get_ADC10bitResult(3);
j += Get_ADC10bitResult(3);

if(j < 1024*4)
{
  j = get_temperature(j); //计算温度值

  if(j >= 400)   F0 = 0, j -= 400;    //温度 >= 0度
  else       F0 = 1, j  = 400 - j; //温度 <  0度
  LED8[4] = j / 1000;    //显示温度值
  LED8[5] = (j % 1000) / 100;
  LED8[6] = (j % 100) / 10 + DIS_DOT;
  LED8[7] = j % 10;
  if(LED8[4] == 0)  LED8[4] = DIS_BLACK;
  if(F0)  LED8[4] = DIS_;    //显示-
}
else  //错误
{
  for(i=0; i<8; i++)  LED8[i] = 0;
}
```

5.倒计时计算

```
if(++msecond >= 1000) //1秒到
{
  //倒计时
  if(if_change){
    if(hour == 0 && min == 0 && sec == 0){
      run_light();
    }
    else if(sec > 0){
      sec--;
    }
    else{
      sec = 59;
      if(min > 0){
        min--;
      }
      else{
        hour--;
        min = 59;
      }
    }
  }
```

6. 前端与 Wi-Fi 模块通信

与 Wi-Fi 模块通信，本质上是 TCP 链接的建立。我们可以使用 websocket 来非常简单的实现。

建立 socket 连接：

```
var ws = new WebSocket("192.168.4.1:8080");
```

当与服务器通信时，log 出提示并且发送温度设置：

```
ws.onopen = function (evt) {
    console.log("Connection open ...");
    ws.send(theTemp);
};
```

回调函数，现实接受的信息：

```
ws.onmessage = function (evt) {
    console.log("Received Message: " + evt.data);
    ws.close();
};
```

结束通行，say goodbye：

```
ws.onclose = function (evt) {
    console.log("Connection closed.");
};
```

7.WiFi 模块实现

开启多连接模式和服务器模式，端口号 8080

```
460  void connect_init()
461  {
462    char *e = "AT+CIPMUX=1\r\n";
463    char *f = "AT+CIPSERVER=1,8080\r\n";
464
465    while(*e != '\0') {
466      SBUF = *e;
467      while(!TI);
468      TI = 0;
469      e++;
470    }
471    delay_ms(10);
472    while(*f != '\0') {
473      SBUF = *f;
474      while(!TI);
475      TI = 0;
476      f++;
477    }
478  }
```

串口配置，配置内容如注释

```
480  void UartConfiguration()
481 {
482   TMOD = 0x20;  //timer1 mode2
483   TH1 = 0xfd;   //timer1 init
484   TL1 = 0xfd;   //timer1 init
485   PCON = 0x00;  //baud no multi
486   SCON = 0x50;  //model receive open
487   ES = 1;       //ttl interrupt ok
488   TR1 = 1;      //timer1 run
489 }
```

中断接受，接受内容格式如图：

```
+IPD,0,7:gettemp
```

有用的信息只有冒号后的东西，如果冒号后是 gettemp，就传送温度过去。

main.c

```
507  void Uart() interrupt 4
508 {
509   sizeof(j);
510   char *send = "AT+CIPSEND=0,4\r\n";
511   char res;
512   res = SBUF;
513   RI = 0;
514   if (res==':'||ptr>0) {
515     if (res == '\n') {
516       if (*receive == ":gettemp") {
517         while(*send != '\0') {
518           SBUF = *send;
519           while(!TI);
520           TI = 0;
521           send++;
522         }
523         delay_ms(10);
524         sendInt(j);
525       }
526       ptr = 0;
527       memset(receive,0,50);
528     } else {
529       receive[ptr] = res;
530       ptr++;
531     }
532   }
533 }
```

8.液晶屏模块（单独实现）

代码实现：

主函数首先需要调用多个初始化函数，对晶振、波特率还有我们的显示屏进行初始化。然后进行不停的循环，在循环中我们根据复位信号的变化对显示屏进行相应处理。buff 是一个数组，用于接受缓冲字节。然后下面的 LCD_P6x8Str()是一个显示 6*8 一组标准 ASCII 字符串

显示的坐标（x,y），y 为页范围 0～7.

```
/*--主函数--*/
void main(void)
{
    int temp;
    char p[2];
    TimInit();
    Ds18b20Init();
    LCD_Init();
    while(1)                            /*无限循环*/
    {
      if(flag_REC==1)                   //
       {
                flag_REC=0;
                if(buff[0]=='O'&&buff[1]=='N')  //第一个字节为O，第二个字节为N，第三个字节为控制码
                switch(buff[2])
                 {
                        case open :                         // 开启
                        flagrun=1;
                        break;

                        case stop:                      // 停止
                        flagrun=0;
                        break;
                 }
       }
         if(flagrun)
             run();
         else
             stoprun();
        temp=Ds18b20ReadTemp();
        p[0]=temp/10;
        p[1]=temp%10;
        LCD_P6x8Str(54,0,p);
    }
}
```

除了主函数我还实现了头文件 lcd 和 temp 以及 i2c。首先介绍头文件 temp。

下面是 temp.h 的定义，定义使用的 IO 口以及几个全局函数。主要是和温度模块进行交互，能够读取温度和写入温度

```h
#ifndef __TEMP_H_
#define __TEMP_H_

#include "reg52.h"
//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

//--定义使用的IO口--//
sbit DSPORT=P3^7;

//--声明全局函数--//
void Delay1ms(uint y);
uchar Ds18b20Init();
void Ds18b20WriteByte(uchar dat);
uchar Ds18b20ReadByte();
void  Ds18b20ChangTemp();
void  Ds18b20ReadTempCom();
int Ds18b20ReadTemp();

#endif
```

```c
/*******************************************************************************
* 函 数 名         : Ds18b20Init
* 函数功能         : 初始化
* 输    入         : 无
* 输    出         : 初始化成功返回1，失败返回0
*******************************************************************************/

uchar Ds18b20Init()
{
    uchar i;
    DSPORT = 0;           //将总线拉低480us~960us
    i = 70;
    while(i--);//延时642us
    DSPORT = 1;           //然后拉高总线，如果DS18B20做出反应会将在15us~60us后总线拉低
    i = 0;
    while(DSPORT)    //等待DS18B20拉低总线
    {
        Delay1ms(1);
        i++;
        if(i>5)//等待>5MS
        {
            return 0;//初始化失败
        }

    }
    return 1;//初始化成功
}
```

```
/*******************************************************************************
* 函 数 名         : Ds18b20WriteByte
* 函数功能         : 向18B20写入一个字节
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void Ds18b20WriteByte(uchar dat)
{
    uint i, j;

    for(j=0; j<8; j++)
    {
        DSPORT = 0;            //每写入一位数据之前先把总线拉低1us
        i++;
        DSPORT = dat & 0x01;  //然后写入一个数据，从最低位开始
        i=6;
        while(i--); //延时68us，持续时间最少60us
        DSPORT = 1; //然后释放总线，至少1us给总线恢复时间才能接着写入第二个数值
        dat >>= 1;
    }
}

}
/*******************************************************************************
* 函 数 名         : Ds18b20ReadTempCom
* 函数功能         : 发送读取温度命令
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void  Ds18b20ReadTempCom()
{

    Ds18b20Init();
    Delay1ms(1);
    Ds18b20WriteByte(0xcc);  //跳过ROM操作命令
    Ds18b20WriteByte(0xbe);  //发送读取温度命令
}

}
/*******************************************************************************
* 函 数 名         : Ds18b20ReadTemp
* 函数功能         : 读取温度
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

int Ds18b20ReadTemp()
{
    int temp = 0;
    uchar tmh, tml;
    Ds18b20ChangTemp();             //先写入转换命令
    Ds18b20ReadTempCom();           //然后等待转换完后发送读取温度命令
    tml = Ds18b20ReadByte();        //读取温度值共16位，先读低字节
    tmh = Ds18b20ReadByte();        //再读高字节
    temp = tmh;
    temp <<= 8;
    temp |= tml;
    return temp;
}
```

然后是头文件 lcd，下面给出该头文件的定义以及比较重要的两个函

数，它们分别是写数据以及写命令

```
#ifndef __LCD_H_
#define __LCD_H_



void DelayMS(unsigned int msec);
void LCD_DLY_ms(unsigned int ms);
void LCD_WrDat(unsigned char dat);
void LCD_WrCmd(unsigned char cmd);
void LCD_Set_Pos(unsigned char x, unsigned char y);
void LCD_Fill(unsigned char bmp_dat);
void LCD_CLS(void);
void LCD_Init(void);
//void LCD_P8x16Str(unsigned char x, unsigned char y,unsigned char ch[]);
void LCD_P16x16Ch(unsigned char x, unsigned char y, unsigned char N);
void LCD_P6x8Str(unsigned char x, unsigned char y,unsigned char ch[]);
#endif

```

一个数据是 8 位的，我们需要对每一位都单独进行操作。对相应的信号需要进行处理

```
/*********************LCD写数据*************************************/
void LCD_WrDat(unsigned char dat)
{
    unsigned char i=8, temp=0;
    LCD_DC=1;
    for(i=0;i<8;i++) //发送一个八位数据
    {
        LCD_SCL=0;

        temp = dat&0x80;
        if (temp == 0)
        {
            LCD_SDA = 0;
        }
        else
        {
            LCD_SDA = 1;
        }
        LCD_SCL=1;
        dat<<=1;
    }
}
/*********************LCD写命令*************************************/
void LCD_WrCmd(unsigned char cmd)
{
    unsigned char i=8, temp=0;
    LCD_DC=0;
    for(i=0;i<8;i++) //发送一个八位数据
    {
        LCD_SCL=0;

        temp = cmd&0x80;
        if (temp == 0)
        {
            LCD_SDA = 0;
        }
        else
        {
            LCD_SDA = 1;
        }
        LCD_SCL=1;
```

最后是 i2c 部分，这个部分是 OLED 的核心，需要以下六个函数：

I2C.H | i2c.c

```
#ifndef __I2C_H_
#define __I2C_H_

#include<reg52.H>

sbit SCL=P1^2;
sbit SDA=P1^3;

void I2cStart();
void I2cStop();
unsigned char I2cSendByte(unsigned char dat);
unsigned char I2cReadByte();
void At24c02Write(unsigned char addr,unsigned char dat);
unsigned char At24c02Read(unsigned char addr);

```

I2cStart 和 I2cStop 分别是起始函数和中止函数。I2cStart 函数的功能是在 SCL 时钟信号在高电平期间 SDA 信号产生一个下降沿；I2cStop 函数的功能是在 SCL 时钟信号高电平期间 SDA 信号产生一个上升沿。这两个函数比较简洁，在这里不过多叙述。

这个是读取字节的函数，我们每次接受八个字节，每接收一个都进行一定程度的延时，这是为了保证整个过程能够按照既定顺序不会发生碰撞。

```
/*****************************************************************************************
* 函数名         : I2cReadByte()
* 函数功能       : 使用I2c读取一个字节
* 输入           : 无
* 输出           : dat
* 备注           : 接收完一个字节SCL=0,SDA=1.
*****************************************************************************************/

unsigned char I2cReadByte()
{
    unsigned char a=0,dat=0;
    SDA=1;          //起始和发送一个字节之后SCL都是0
    Delay10us();
    for(a=0;a<8;a++)//接收8个字节
    {
        SCL=1;
        Delay10us();
        dat<<=1;
        dat|=SDA;
        Delay10us();
        SCL=0;
        Delay10us();
    }
    return dat;
}
```

三、实验分析及思考

1.这次试验参考了之前做过的通信和 RTC 时钟实验，以及 stc 样例，最后决定做一个具有现实应用意义的东西，也符合老师在课堂上提到的物联网的思维。

2.由于我们之前并没有对单片机以及相应的模块进行学习或研究，因此在基础的配置上（包括液晶屏的点亮、wifi 模块的通信、手机端与单片机连接等部分）花费了不少的时间，也从中学到了许多关于嵌入式软件和硬件的知识。

3.由于对工作量的预估不明确，因此我们一开始想实现的功能其实是更多而且更广了，然而由于时间的限制和技术的不足（流下了没技术的眼泪.jpg），因此最后需求一再缩小，这也提示我们作为产品一定要从实际需求出发，在版本的迭代中更新功能，而不是一开始就提出大量无法实现的需求。

四、附录

1.实验分工

整个大作业由小组全体成员共同合作完成

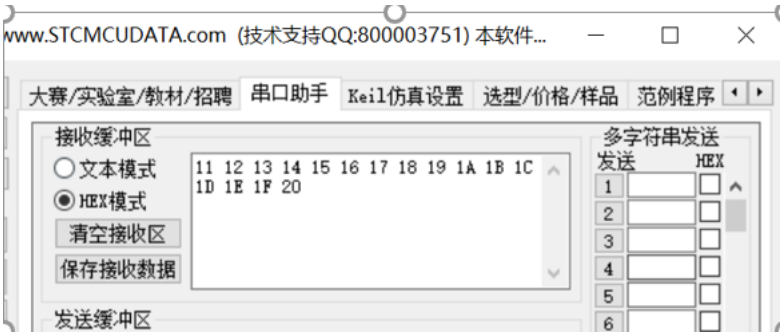
2.源码及实验现象

（1）板上的源码见附件文件夹【源码】

（2）手机端的源码见 github：

https://github.com/Xiongzj5/embedded-design

（3）将串口输出放到 USB 串口后，可以通过串口助手观察到数据



（4）具体实现使用视频方式进行展示，百度云网址如下:

https://pan.baidu.com/s/16gblrz4GJRTUhPmPRxQY_Q

3.最初的产品说明文档：https://shimo.im/docs/NSSFMfxFUy8waVrc