华南理工大学 广州学院 Guangzhou College of South China University of Technology

《嵌入式课程项目实践》说明书

基于 51 单片机的温度采集及 LED 灯控制系统

院　　系：计算机工程学院

专业班级：计算机科学与技术 3 班

学　　号：201610089208

姓　　名：高文涛

指导教师：阳韬

一级指标	二级指标	分值	评分及成绩 得分
作品完成度（占 50%）	1. 工作量达标，完成基本功能完整，独立完成	30	
	2. 完成扩展功能，特色鲜明，有设计亮点	10	
	3. 功能测试设计合理性，逻辑条理性	10	
文档撰写（30%）	1. 文字描述规范，思路描述清晰	10	
	2. 功能测试结果完整	10	
	3. 结果分析合理	10	
答辩成绩（占 20%）	1. 能够对设计进行合理说明	10	
	2. 能较好的回答答辩所提问题，解释合理清晰	10	
合计（百分制）			
总 评	□优　□良　□中　□及格　□不及格		签名

一、概述

1.开发工具：

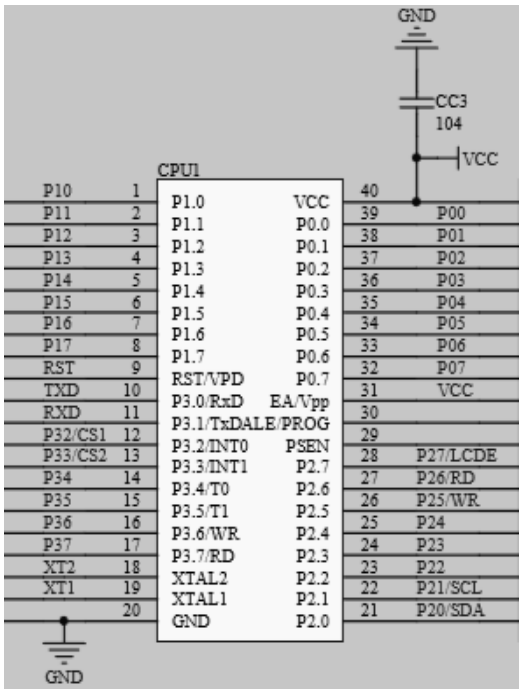
硬件：STC52 单片机开发板、ESP8266、DS18B20、LCD1602 ；
软件：Keil4、Android Studio、串口调试软件。

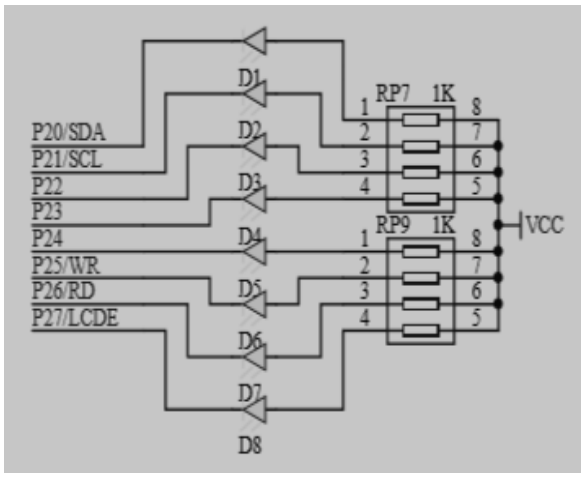
2.功能简述：

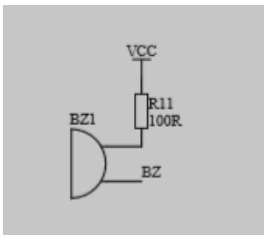
单片机采集 DS18B20 温度传感器的信号，将信号转化为对应的温度信息，并在 LCD1602 液晶屏上显示，同时使用 ESP8266 模块，使得手机端接入该模块发射的 WIFI，在手机 APP 上实现对温度的实时采集以及对 LED 灯的控制。

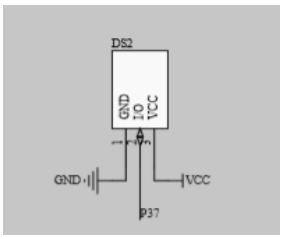
二、硬件设计与使用说明

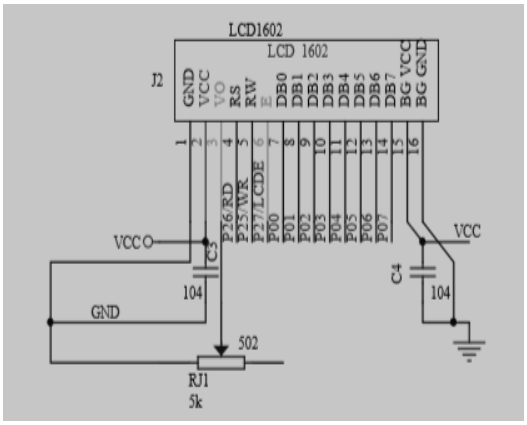
1、STC89C52 与模块的接线图：

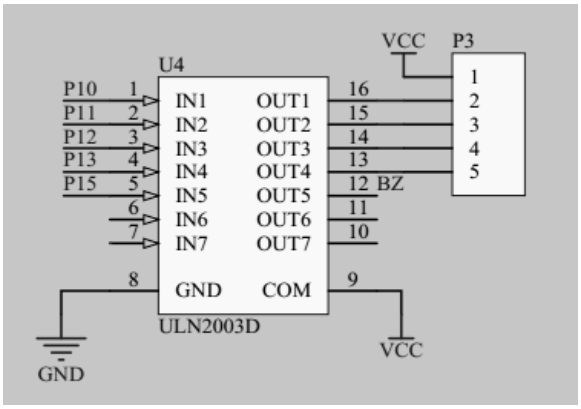












2、DS18B20 模块：

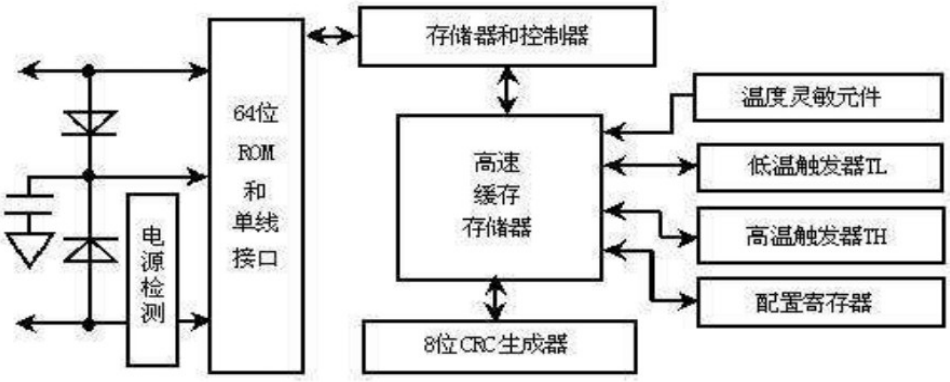
2.1 简介：

DS18B20 数字温度传感器接线方便，封装后可应用于多种场合，如管道式，螺纹式，磁铁吸附式，不锈钢封装式。主要根据应用场合的不同而改变其外观。

2.2 特点：

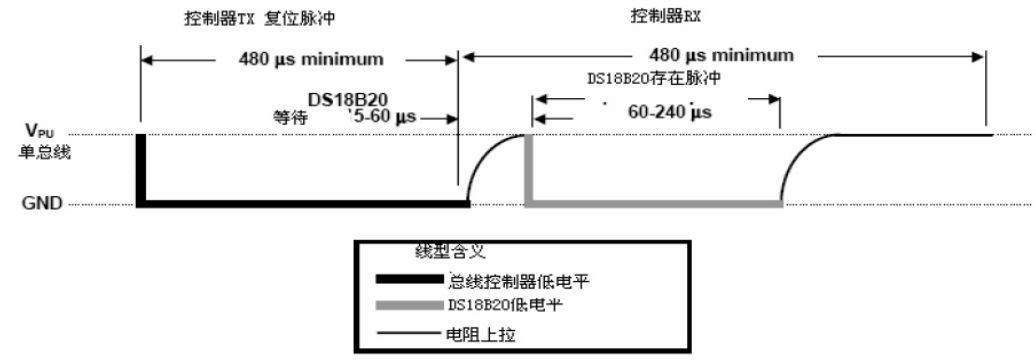
1. 独特的单线接口方式，DS18B20 在与微处理器连接时仅需要一条线即可实现微处理器与 DS18B20 的双向通讯；
2. DS18B20 在使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内；
3. 可编程的分辨率为 9～12 位，对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃，可实现高精度测温，在上电状态下默认的精度 12 位；
4. 在 9 位分辨率时最多在 93.75ms 内把温度转换为数字，12 位分辨率时最多在 750ms 内把温度值转换为数字，速度更快；
5. 温范围－55℃～+125℃。

2.3 DS18B20 内部结构：

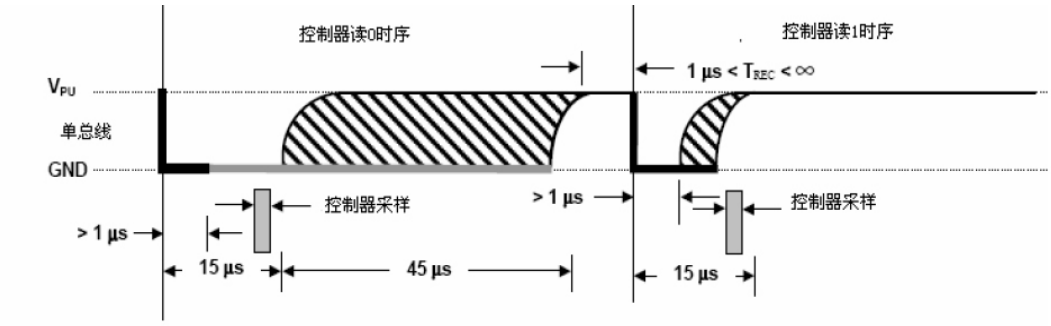


2.4 DS18B20 的初始化时序

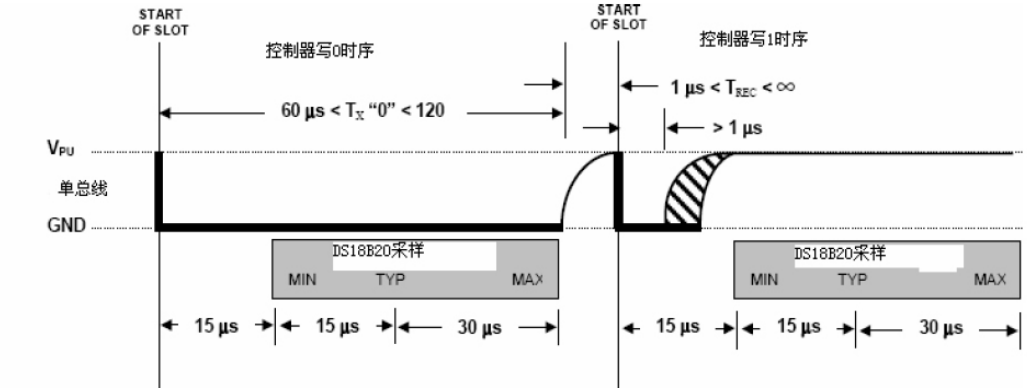
初始化时序 图 13



2.5 DS18B20 的读时序



2.6 DS18B20 的写时序



2.7 使用到的 DS18B20 指令表

ROM 指令表

指令	约定代码	功能
跳过 ROM	CCH	忽略 64 位 ROM 地址，直接向 DS18B20 发温度变换命令，适用于单片机工作

RAM 指令表

指令	约定代码	功能
温度转换	44H	启动 DS18B20 进行温度转换，12 位转换时长为 750ms，结果存放于内部 9 字节的 RAM 中
读暂存器	BEH	读内部 RAM 中 9 字节的内容

注：由于上电状态下默认的精度 12 位，12 位转化后得到的 12 位数据，存储在 DS18B20 的两个 8 位的 RAM 中，高字节的前 5 位是符号位，如果测得的温度大于 0，这 5 位为‘0’，只要将测到的数值乘以 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为‘1’，测到的数值需要先减 1 再取反再乘以 0.0625 即可得到实际温度。

3、LCD1602 模块：

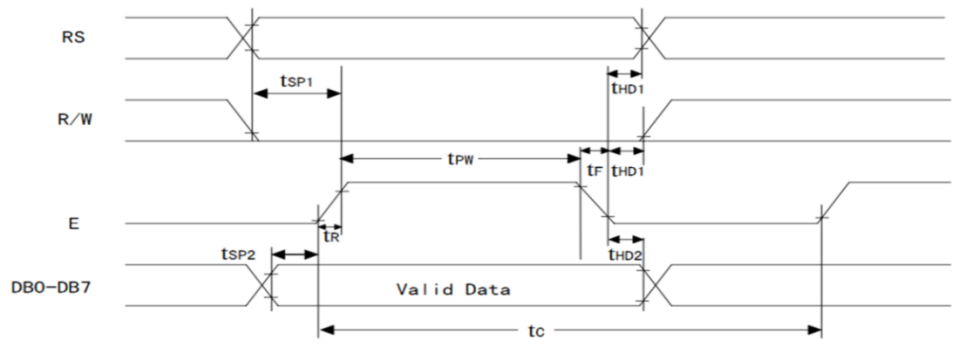
3.1 简介：

1602 液晶也叫 1602 字符型液晶，它是一种专门用来显示字母、数字、符号的点阵型液晶模块。它是由若干个 5x7 或 5x10 的点阵字符位组成，每个点阵字符位都可以用显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用。

3.2LCD1602 结构图及引脚功能

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	Data I/O
2	VDD	电源正极	10	D3	Data I/O
3	VL	液晶显示偏压信号	11	D4	Data I/O
4	RS	数据/命令选择端（H/L）	12	D5	Data I/O
5	R/W	读/写选择端（H/L）	13	D6	Data I/O
6	E	使能信号	14	D7	Data I/O
7	D0	Data I/O	15	BLA	背光源正极
8	D1	Data I/O	16	BLK	背光源负极

3.3 LCD1602 写入时序图



注：R/S=0：选择发送命令；R/W=1：选择发送数据；R/W=0：写入。

3.4 使用到的 LCD1602 指令表

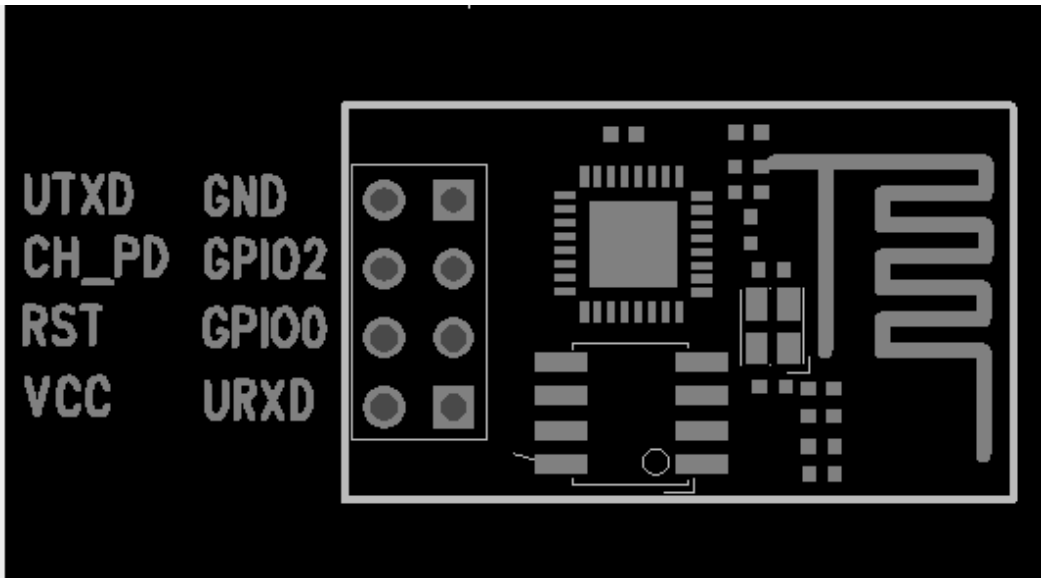
指令	代码	功能
功能设定	38H	告诉模块数据线为 8 位，显示 2 行，5×7 点阵/每字符
显示开关控制	0CH	开启显示功能，不显示光标
进入模式设定	06H	写入新数据后显示屏不移动，光标右移
清屏	01H	清除液晶屏内容
显示设定	80H	告诉模块在哪里显示字符，80H 为第一行第一个

4、ESP8266 模块：

4.1 简介：

ESP8266 集成了 32 位 Tensilica 处理器、标准数字外设接口、天线开关、射频 balun、功率放大器、低噪放大器、过滤器和电源管理模块等，仅需很少的外围电路，可将所占 PCB 空间降低。

4.2 ESP8266 结构图



4.3 使用说明

从网上购买的 ESP8266 模块商家已经刷入固件了，所以只需利用 USB-TTL 模块与电脑连接，在串口调试助手发送 AT 指令对 ESP8266 模块进行调试。之后就可以接线到开发板上进行使用了。

接线说明：ESP8266 模块的 VCC,GND,CH_PD,UTXD,URXD 口分别接 USB 转 TTL 模块的 3.3v,地,3.3v,RXD,TXD

AT 指令	功能
AT	返回 OK，则证明模块可以正常使用
AT+CWMODE=2	设置模式，1：STA 模式；2：AP 模式；3：STA+AP 模式

AT+CWSAP=”TEST”, ”1234567890”,11,3	配置 AP 参数：1.设置 WIFI 账户；2.设置 WIFI 密码；3.设置通道号；4.加密模式
AT+RST	重启模块，上条指令设置后重启生效
AT+CIPMUX=1	开启多路连接模式　　（断电后需重新配置）
AT+CIPSERVER=1,8080	开启服务器，端口号为 8080（断电后需重新配置）
AT+CIPSTART=0,"TCP", "192.168.4.2",5000	建立 TCP 连接，客户端 IP 为 192.168.4.2，端口为 5000
AT+CIPSEND=0, ()	确定发送数据的长度

4.4 ESP8266 模块对数据的接受和发送

1. 手机成功连接 ESP8266 模块的 WIFI，打开手机网络调试助手，把协议类型设置为"TCP Client",IP 地址设置为用 AT+CIFSR 指令查询到的 ESP8266 的 IP,端口号为设置好的 8080。此时手机发送 1，电脑串口调试助手收到的信息为：+IPD,0,3:1。
2. 从电脑串口调试助手通过 ESP8266 模块发送信息到手机端，需要用到两条 AT 指令对 ESP8266 进行设置，之后发送信息。

5、蜂鸣器：

开发板上采用无源蜂鸣器与 ULN2003 组成电路。

无源蜂鸣器：利用电磁感应现象，为音圈接入交变电流后形成的电磁铁与永磁铁相吸或相斥而推动振膜发声，接入直流电只能持续推动振膜而无法产生声音，只能在接通或断开时产生声音，调节驱动的 PWM 频率还有占空比，来改变无源蜂鸣器的鸣叫声音

ULN2003：是大电流驱动阵列,多用于单片机、智能仪表、PLC、数字量输出卡等控制电路中。可直接驱动蜂鸣器、继电器等负载

三、单片机程序设计

1、DS18B20 模块部分程序代码与解析

引脚定义：sbit　DSPORT=P3^7;

1.1 DS18B20 初始化函数，根据时序图对相应引脚进行电平变化：

```
void Ds18b20Init()
{
    u8 i;
    DSPORT = 0;         //总线拉低并延时 480us~960us
    i = 70;
    while(i--);
    DSPORT = 1;         //拉高总线并等待 DS18B20 拉低总线
    Delay1(5);          //延时 5ms 即可初始化成功
}
```

1.2 向 DS18B20 写入一个 8 位的数据，根据时序图对引脚电平的变化，使得其进入接受数据的模式：

```
void Ds18b20Write(u8 dat)
{
    u16 i, j;
    for(j=0; j<8; j++)
    {   DSPORT = 0;       //总共写入 8 位数据，每次写入前先拉低总线
        i++;
        DSPORT = dat & 0x01;   //从最低位开始写入
        i=6;
        while(i--);            //每次写完一位延时 60us
        DSPORT = 1;            //然后释放总线
        dat >>= 1;
    }
 }
```

1.3 向 DS18B20 读取 8 位的数据，操作引脚电平使得其返回数据：

```
u8 Ds18b20ReadByte()
{
    u8 byte, b;
    u16 i, j;
    for(j=8; j>0; j--)
    {
        DSPORT = 0;     //总共读取 8 位数据，每次读取前先拉低总线
        i++;
        DSPORT = 1;     //然后拉高总线
        i++;i++;        //延时 6us 等待数据稳定
        b = DSPORT;     //读取数据，最低位开始
        byte = (byte >> 1) | (b << 7);//将读取的一位数据并存放
        i = 4;          //延时 48us 再读取下一位数
        while(i--);
    }
    return byte;
 }
```

1.4 由 DS18B20 的使用说明可知，

如需获取温度值需要发送四条指令（CCH、44H 和 CCH、BEH），即跳过 ROM 操作命令、温度转化命令和跳过 ROM 操作命令、读取温度命令，利用函数 Ds18b20Write（）实现。

DS18B20 设置完毕后即可读取一个 16 位温度值，利用 Ds18b20ReadByte() 每次读取一个字节的数据，进行两次操作，最后再将 16 位的二进制数转换成十

进制数。

2、LCD1602 模块部分程序代码与解析

引脚的定义：

```
#define LCD1602_DATAPINS P0
sbit LCD1602_E=P2^7;            //LCD1602 使能引脚
sbit LCD1602_RW=P2^5;           //LCD1602 读写选择端引脚
sbit LCD1602_RS=P2^6;           //LCD1602 数据命令选择端引脚
```

2.1.LCD1602 初始化函数，根据使用说明，写入数据使其开启：

```
void LcdInit()
{
    LcdWriteCom(0x38);          //开显示
    LcdWriteCom(0x0c);          //开显示不显示光标
    LcdWriteCom(0x06);          //写一个指针加一
    LcdWriteCom(0x01);          //清屏
    LcdWriteCom(0x80);          //设置数据指针起点
}
```

2.2. 控制电平变化并向 LCD1602 写入一个字节的命令的函数：

```
void LcdWriteCom(u8 com)
{
    LCD1602_E = 0;              //使能
    LCD1602_RS = 0;             //选择发送命令
    LCD1602_RW = 0;             //选择写入
    LCD1602_DATAPINS = com;     //放入命令
    Lcd1602_Delay1ms(1);        //延时等待数据稳定¨
    LCD1602_E = 1;              //写入时序
    Lcd1602_Delay1ms(5);        //延时等待
    LCD1602_E = 0;
}
```

2.3 向 LCD1602 写入一个字节的数据

```
void LcdWriteData(u8 dat)
{
    LCD1602_E = 0;              //使能清零
    LCD1602_RS = 1;             //选择输入数据
    LCD1602_RW = 0;             //选择写入
    LCD1602_DATAPINS = dat;     //写入数据
    Lcd1602_Delay1ms(1);        //延时
    LCD1602_E = 1;              //写入时序
    Lcd1602_Delay1ms(5);        //延时等待
    LCD1602_E = 0;
}
```

2.4 使用 LCD1602 模块只需要根据时序图对其写入控制命令，再写入数据就可以在液晶屏上显示数据。

3、ESP8266 模块部分程序代码与解析

ESP8266 的输入输出引脚分别接在单片机的 TXD 和 RXD 进行串口通信，所以需要用到中断处理，则要对中断控制寄存器（TCON)、定时/计数器工作模式寄存器（TMOD)、串口控制寄存器（SCON）进行设置，同时设置 4800 的波特率。

3.1 对寄存器进行初始化的函数

```
void UartConfiguration()
{
    TMOD=0x20;       //使用定时器计数器 T1 的方式 2
    TH1=0xF3;        //对 T1 的初值进行设置，产生 4800 波特率
    TL1=0xF3;
    PCON=0x80;       //波特率加倍
    SCON=0x50;       //设置为工作方式 1
    EA=1;            //打开总中断
    ES=1;            //打开接收中断
    TR1=1;           //打开计数器
}
```

3.2 由 ESP8266 的使用说明可知，对 ESP8266 初始化时需要发送两条指令，而向 ESP8266 发送数据则需要两条指令和数据字符串

```
u8 *mux="AT+CIPMUX=1\r\n";
u8 *server="AT+CIPSERVER=1,8080\r\n";

u8 *start="AT+CIPSTART=0,\"TCP\",\"192.168.4.2\",5000\r\n";
u8 *len="AT+CIPSEND=0,6\r\n";
void SendData(u8 *dat)
{
    while(*dat != '\0')
    {
       SBUF = *dat;        //一位数据放入数据缓存寄存器 SBUF
       while(!TI);         //等待数据发送完成，硬件自动置一
       TI = 0;             //TI: 发送中断标志位
       dat++;
    }  delay1ms(300);
}
```

4、蜂鸣器和 LED 灯控制程序代码

引脚的定义：

```
#define led P2
sbit beep=P1^5;
sbit led1=P2^4;
sbit led2=P2^3;
sbit led3=P2^2;
sbit led4=P2^1;
sbit led5=P2^0;
```

4.1 流水灯的函数，使用头文件 intrins.h 左移函数_crol_()右移函数_cror_()。

```
void flow(u8 i)
{
     u8 j;
     if(i==1)
     {
         led=0xfe;
         delayus(30000);          //大约延时 450ms
         for(j=0;j<7;j++)         //将 led 左移一位
         {
             led=_crol_(led,1);
             delayus(30000);      //大约延时 450ms
         }
         for(j=0;j<7;j++)         //将 led 右移一位
         {
             led=_cror_(led,1);
             delayus(30000);      //大约延时 450ms
          }
          led=0xff;
     } }
```

4.2 由于开发板采用的是无源蜂鸣器，需要不停地输入高低电平才能发声，所以该函数执行的时间大约为 1 秒，这样不会影响到其他程序的执行。

```
void voice(u8 i)
{
     u16 j=10000;
     if(i==1)                     //判断是否执行
     {   while(j--)               //执行大约一秒后退出
         {
           beep=~beep;            //不停地变换高低电平
           delayus(10);
        }
     }}
```

5、main 函数程序代码

```
void main()
{
   ESP8266_init();                  //ESP8266 模块的初始化
   while(1)
  {
      LcdInit();                    //LCD1602 模块的初始化
      datapros(Ds18b20ReadTemp());//将温度数值转换成字符串
      for(i=0;i<15;i++)
      {                          //每次向 LCD1602 写入一个字符
        LcdWriteData(DisplayData[i]);
      }
      Delay(1000);          //每隔一秒采集数据并显示
      SendToPhone(btn1)     //当 btn1=1，向手机端发送温度值
      voice(btn2);          //当 btn2=1，执行蜂鸣器响起
      light(btn3);          //当 btn3=1，led 灯全部点亮
      flow(btn4);           //当 btn4=1，实现 led 流水灯
 }
```

6、中断函数

由 ESP8266 的使用说明可知，手机端发送数据的格式为：+TPD,1,1：(),中断函数每次接受一位数据，当接收到 10 位数据后进行判断，确定执行哪个函数。

```
 void Uart() interrupt 4
 {
   if(RI == 1)
   {
      RI = 0;     receiveTable[j]=SBUF;
      if(receiveTable[0]=='+')    //判断数据格式是否正确
          j++;
      else        j=0;
      if(j==10)                    //10 位数据接受完毕，
      {
          j=0;
          switch(receiveTable[9])
          {
            case '1':             //btn1=1 时向手机端发送温度数值
                btn1=exchange(btn1);
                break;
            case '2':             //btn2=1 时使得蜂鸣器响起
                btn2=exchange(btn2);
                break;
             ·····················}}}}
```

四、APP 程序设计

1、创建一个用于向 ESP8266 发送数据的 Send 类

```
//这里定义连接 ESP8266 的 IP 和端口号的变量
  private static final String IP = "192.168.4.1";
  private static final int PORT = 8080;

//定义 socket 类型的变量用于客户端和服务器的通信，PrintStream 的变量用来发送数据到服务端
  private Socket client = null;
  private PrintStream out = null;

//重写 doInBackground（），编写异常处理机制，用来连接服务器并向服务器发送数据
  protected Void doInBackground(String... params) {
       String str = params[0];
       try {
              //通过 IP 地址和端口实例化 Socket，请求连接服务器
               client = new Socket(IP, PORT);
              client.setSoTimeout(5000);
              // 获取 Socket 的输出流，用来发送数据到服务端
              out = new PrintStream(client.getOutputStream());
              out.print(str);
              out.flush();
              if (client == null) {
                  return null;
              } else {
                  out.close();
                  client.close();
              }
          } catch (IOException e) {
              e.printStackTrace();
       }
       return null;
   }
```

2、创建一个设置手机端服务器端口号以及接受服务器端数据

PhoneServer 类

```
//创建一个 ServerSocket 类型的变量，用于开启手机端端口
//创建一个 DataInputStream 类型的变量，用来接受数据
    private ServerSocket server;
    private DataInputStream in;
```

```
    private byte[] receice;
//主线程无法进行时间比较繁长的任务,所以需要子线程进行处理,
    public void setHandler(Handler handler)
    {
        this.handler = handler;
    }

//编写 run (), 利用异常处理机制对客户端发送的信息进行处理, 存放在线程
//的 message 中, 以便在主程序中调用
```

```
    public void run() {
       try {
             //5000 是手机端开启的服务器的端口号, ESP8266 进行 TCP 连接
              时使用的端口, 而 IP 也是通过指令查询的联入设备的 IP
             server = new ServerSocket(5000);
              while (true) {
                 Socket client = server.accept();
                 in = new DataInputStream(client.getInputStream());
                 receice = new byte[50];
                 in.read(receice);
                 in.close();

                 Message message = new Message();
                 message.what = 1;
                 message.obj = new String(receice);
                 handler.sendMessage(message);
              }
         } catch (IOException e) {
             e.printStackTrace();
        }
        try {
             server.close();
        } catch (IOException e) {
             e.printStackTrace();
        }
    }
```

3、在 MainActivity 中设置按钮监听事件及对接受到信息进行显示

```
//定义控件的变量
    private TextView tv_content;
    private Switch btn1,btn2, btn3, btn4;
    private CheckBox btn5, btn6, btn7, btn8, btn9;
    private String str;
```

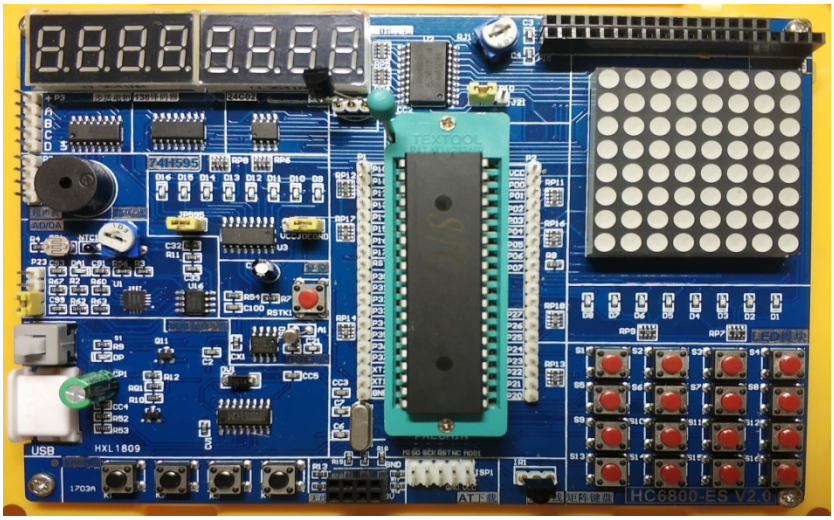
```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        InitView();                                    //按钮初始化
        lisetenOnClick();                              //设置按钮监听事件
        PhoneServer mobileServer = new PhoneServer();//开启服务器
        mobileServer.setHandler(handler);              //创建线程
        new Thread(mobileServer).start();              //打开线程
    }
```

//编写控件的监听事件，打开或者关闭都将发送信息给服务器端，如下是温度开
//关控件，当控件关闭时，发送数据到服务器端，并设置 TestView 的值为空。

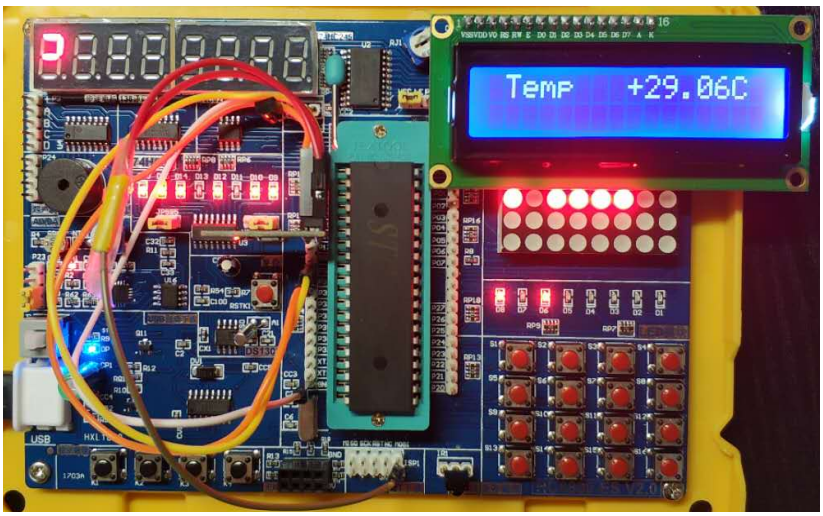
```
  public void lisetenOnClick()
  {
        btn1.setOnCheckedChangeListener(newCompoundButton.OnCheckedChangeListener() {
              public void onCheckedChanged(CompoundButton buttonView,
                  boolean isChecked) {
                  if (isChecked) {
                      str = "1";new Send().execute(str);
                  }
                  else{
                      str = "1";new
                        Send().execute(str);tv_content.setText(" ");
                   }
               }
           } );
//创建子线程，当服务器端发送数据，对其进行接受并在 TestView 控件上显示
//温度值
    Handler handler = new Handler() {
          public void handleMessage(Message msg) {
             ////////////////////温度数值的显示////////////
              switch (msg.what) {
                  case 1:
                      tv_content.setText(""+msg.obj);
              }
          }
      }
```

五、功能调试

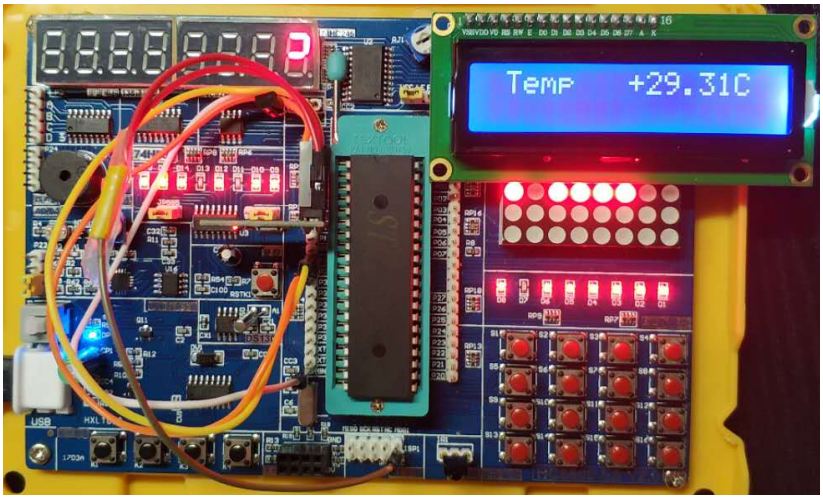
1、开发板实物图



2、开发板连接模块并打开电源



3、用手机端打开 LED 灯



4. 手机端 APP 界面与打开温度采集按钮

5. 第三、四以及 LED 灯模式控制按钮是互斥事件，是因为程序只控制 8 个 LED 灯，若三类控制开关同时打开并不能很好的呈现功能效果，所以三个按钮中打开一个按钮其他两个进行锁定，待打开按钮关闭，另外两个按钮才能解锁。

六、收获及体会

51 单片机开发板我是在大三第一学期才开始购买并学习的，到这学期开学的时候我学完该开发板上的所有模块了，所以此次项目实践我选择一个人一组，不进行组队，一方面是想对自己的学习成果进行检验，另一方面是可以按照自己的想法进行功能的拓展，制作得能许发复杂点。

因为之前没有接触过 WIFI 模块，不知道具体使用方法，在网上查找了很多资料，摸索了好久才发现使用起来非常简单，只需写入几条 AT 指令即可正常使用了。由于之前已经完整地学完该开发板上的教程，经过了几天的努力，终于实现了用手机端 APP 对开发板的温度采集、蜂鸣器以及 LED 等的控制。回顾起来其实 51 单片机的开发很简单，了解每个模块的时序图、内部结构或者使用说明，最后将多个模块组合起来就可以实现。同时发现我对嵌入式方面知识的学习还不够多，需在今后的时间中，更深入地学习，才能有所收获。

七、参考文献

《单片机原理与接口技术》 清华大学出版社

《Android 移动平台应用开发高级教程》 清华大学出版社

《C 程序设计》 清华大学出版社

《51 单片机通过 ESP8266 模块与手机进行通讯》 CSDN 博客

《使用 ESP8266 简单实现和 APP 通讯》 简书