物联网工程实习报告
——数据采集部分

项目名称：　RTD-PT100 温度数据采集系统

学院：信息科学与技术学院

指导老师：吴宗玲

专业：物联网工程

姓名：谭梓琦

学号：2015112210

二〇一八年七月

摘要

本文采用 RTD –PT100 作为温度传感器，AT89C51 单片机作为下位机主控芯片，使用 PCF8591 进行 AD 转换，LCD 显示屏分时显示八个温度测量数据，并在上位机上实时显示 8 个温度值，以达到温度实时监控的目的。创建一对虚拟串口，下位机与上位机分别绑定一个串口，进行通信。下位机使用 Proteus 进行仿真，单片机控制程序使用 Keil uVersion4 进行调试，上位机使用 Visual Studio 2017 C#进行设计。

关键词：RTD-PT100　AT89C51　Proteus　温度测量　C#上位机　VS2017　虚拟串口

目录

1 Introduction

1.1 项目背景

为提高物联网工程学生独立解决复杂工程的能力，该实习项目应运而生，通过仿真一个简单的物联网上位机下位机通信系统，学生能够了解感知层的信息感知以及传输过程，从而提高对物联网工程的认知深度，更重要的是提高了动手能力和实践能力。

1.2 需求分析

用户需求分析：使用 8 个温度传感器感知温度，将温度信息传输到通用计算机上实时显示。

系统需求分析：采用 RTD –PT100 作为温度传感器，AT89C51 单片机作为下位机主控芯片，使用 PCF8591 进行 AD 转换，LCD 显示屏分时显示八个温度测量数据，并在上位机上实时显示 8 个温度值，以达到温度实时监控的目的。创建一对虚拟串口，下位机与上位机分别绑定一个串口，进行通信。下位机使用 Proteus 进行仿真，单片机控制程序使用 Keil uVersion4 进行调试，上位机使用 Visual Studio 2017 C#进行设计。

1.3 验收方案

1、进行下位机仿真电路和 Keil 单片机程序的编写，要求传感器测量节点不小于 4 通道，并能用数码管或 LED 液晶在 Proteus 上独立进行展示；

2、进行上位机软件的设计，要求能够通过虚拟串口接收下位机发送的数据，并进行数据曲线或图形的绘制；

3、要求能够通过虚拟串口和串口服务器软件接收服务器端（教师端）的命令，收到命令后启动或停止数据采集工作。

2 项目管理

2.1 参加学生情况

表 2-1 项目成员情况

姓名	学号	年级	专业班级	特长
谭梓琦	2015112210	2015 级	物联网工程 1 班	软硬件设计与调试 程序设计 报告设计
张红莉	2015112224	2015 级	物联网工程 1 班	界面设计 配对 Debug 文书写作

2.2 项目分工

本次开发，采用配对开发的形式，为提高程序员找出错误的概率，另一方也同时参与编程，大大提高了程序设计的效率，但是人力成本也相应提高。

表 2-2 项目分工

姓名	项目分工
谭梓琦	1、 搜集温度传感器，PCF8591，串口通信的相关文献； 2、 设计 Proteus 原理图； 3、 设计与调试 Keil 单片机程序； 4、 调试上位机与下位机的串口通信；
张红莉	1、 研究 VS2017C#上位机工程的编程框架与语法； 2、 设计 VS2017 的 UI； 3、 与项目成员配对 Debug 4、 文书写作

3 项目设计过程

3.1 总体方案

本方案包括两个部分，上位机设计与下位机设计。下位机使用 8 个 RTD-PT100 温度传感器获取温度信号，并利用模数转换模块将模拟信号其转化为计算机能够识别的数字信号，经单片机处理后发送至 LCD 进行温度显示，并根据与上位机约定的格式把温度信息发送到上位机显示。下位机使用 Proteus 仿真，上位机使用 Visual studio 2017 的 C#设计。

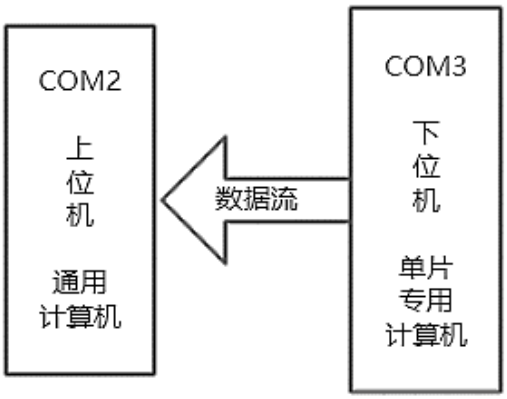


图 3-1 上位机与下位机

3.2 下位机设计

使用热电阻 RTD-PT100 温度传感器利用其感温效应，热电阻随环境温度的变化而变化，在电路图中，将电阻值的变化转换成电压的变化，再将电压值作为输入信号输入至 AD 转换器中进行模拟信号到数字信号的转换，其输出端接单片机，向单片机内写入源程序，将被测温度在显示器上显示出来。

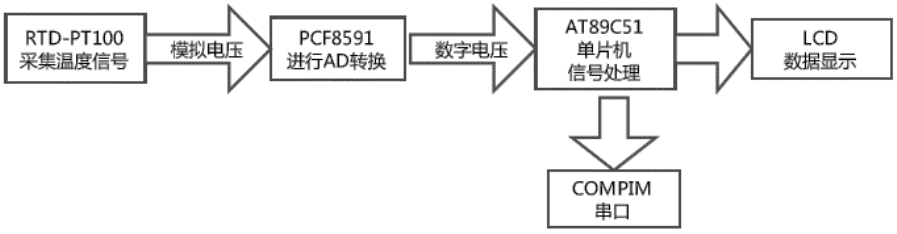


图 3-2 下位机设计框架

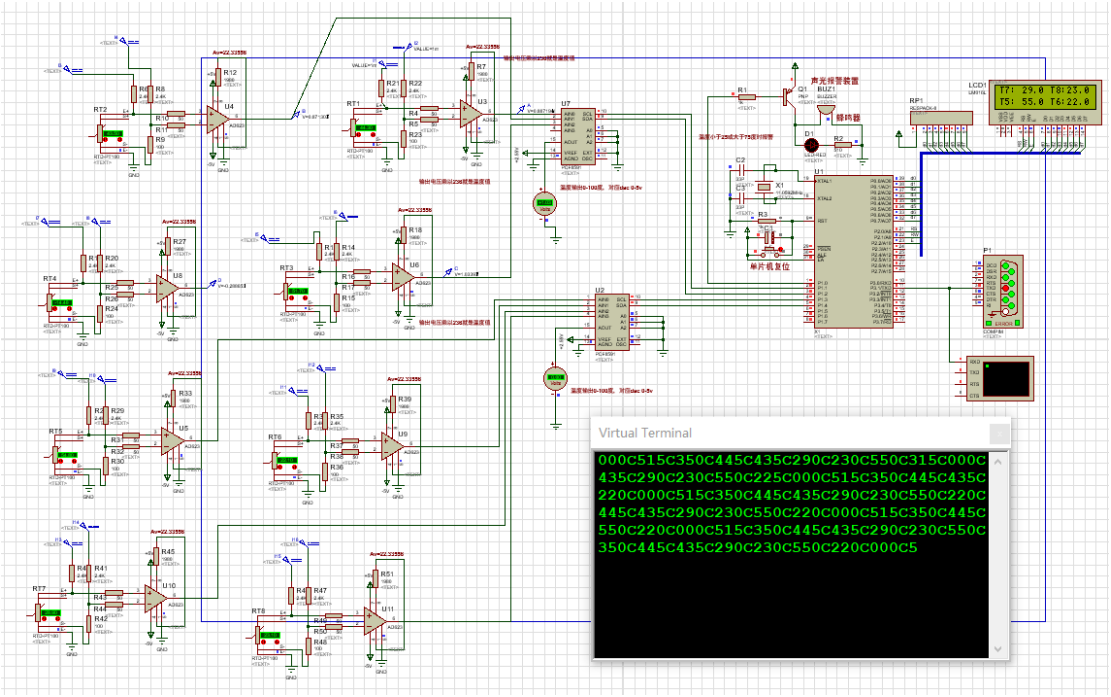


图 3-3 下位机原理图全貌

3.2.1 单片机 AT89C51

AT89C51 作为温度测试系统设计的核心器件。该器件是 INTEL 公司生产的 MCS—5I 系列单片机中的基础产品，采用了可靠的 CMOS 工艺制造技术。具有高性能的 8 位单片机，属于标准的 MCS-51 的 CMOS 产品。片内含 8Kbytes 的可反复擦写的只读程序存储器（PEROM）和 256bytes 的随机存取数据存储器（RAM），器件兼容标准的 MCS-51 指令统。片内置通用 8 位中央处理器（CPU）和 Flash 存储单元。结合了 HMOS 的高速和高密度技术及 CHMOS 的低功耗特征。

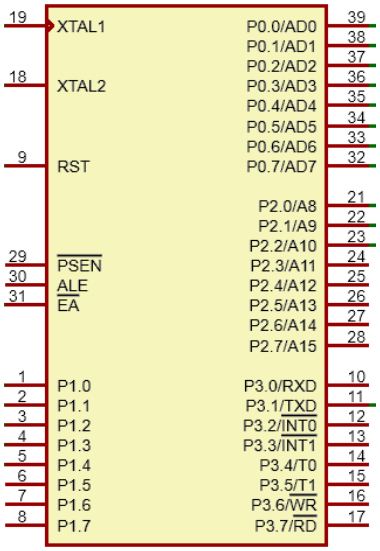


图 3-3 AT89C51 单片机封装

其具有如下性质:

（1）与 MCS-51 产品指令系统完全兼容
（2）4K 字节可重擦写 Flash 闪烁存储器。
（3）寿命：1000 写/擦循环。
（4）数据保留时间：10 年。
（5）全静态工作：0Hz-24Hz。
（6）三级程序存储器锁定。
（7）128*8 位内部 RAM。
（8）32 可编程 I/O 线。
（9）两个 16 位定时器/计数器。
（10）8 个中断源。
（11）可编程串行通道。
（12）低功耗的闲置和掉电模式。
（13）片内振荡器和时钟电路。

AT89C51 单片机提供以下标准功能：4k 字节 Flash，256 字节 RAM，32 位 I/O 口线，看门狗定时器，2 个数据指针，三个 16 位定时器/计数器，一个 6 向量 2 级中断结构，全双工串行口，片内晶振及时钟电路。另外，AT89C51 可降至 0Hz 静态逻辑操作，支持 2 种软件可选择节电模式。空闲模式下，CPU 停止工作，允许 RAM、定时器/计数器、串口、中断继续工作。掉电保护方式下，RAM 内容被保存，振荡器被冻结，单片机一切工作停止，直到下一个中断或硬件复位为止。

3.2.2 温度传感器 RTD-PT100

1、简介

Resistance Temperature Detector 电阻温度探测器，PT100 是铂热电阻。在 0℃时，它的阻值为 100 欧姆，在 100℃时它的阻值约为 138.5 欧姆。

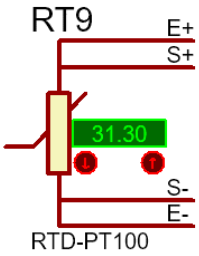


图 3-4 RTD-PT100 Proteus 器件

2、引线方式：

国标热电阻的引线主要有三种方式：

二线制：在热电阻的两端各连接一根导线来引出电阻信号的方式叫二线制：这种引线方法很简单，但由于连接导线必然存在引线电阻 r，r 大小与导线的材质和长度的因素有关，因此这种引线方式只适用于测量精度较低的场合。

三线制：在热电阻的根部的一端连接一根引线，另一端连接两根引线的方式称为三线制，这种方式通常与电桥配套使用，可以较好的消除引线电阻的影响，是工业过程控制中的最常用的引线电阻。

四线制：在热电阻的根部两端各连接两根导线的方式称为四线制，其中两根引线为热电阻提供恒定电流 I，把 R 转换成电压信号 U，再通过另两根引线把 U 引至二次仪表。可见这种引线方式可完全消除引线的电阻影响，主要用于高精度的温度检测。

3、本文采用的引线方式

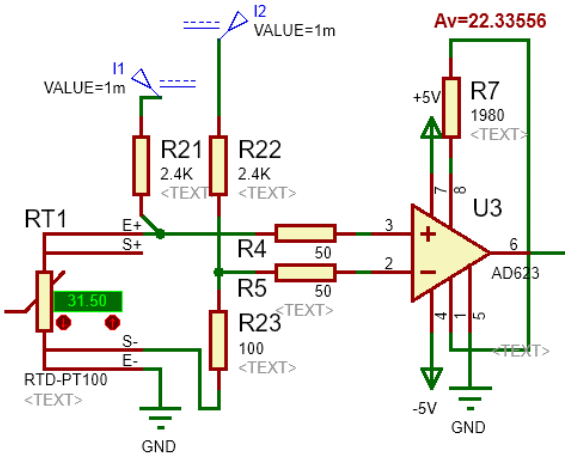


图 3-5 本文 RTD-PT100 引线方式

3.2.3 PCF8591

1、简介

8 位 AD 和 DA 转换器。

PCF8591 是一个单片集成、单独供电、低功耗、8-bit CMOS 数据获取器件。PCF8591 具有 4 个模拟输入、1 个模拟输出和 1 个串行 I²C 总线接口。PCF8591 的 3 个地址引脚 A0, A1 和 A2 可用于硬件地址编程，允许在同个 I2C 总线上接入 8 个 PCF8591 器件，而无需额外的硬件。在 PCF8591 器件上输入输出的地址、控制和数据信号都是通过双线双向 I2C 总线以串行的方式进行传输。

2、特性

单电源供电

工作电压：2.5~6V

待机电流低

I2C 总线串行输入/输出

通过 3 个硬件地址引脚编址

采样速率取决于 I2C 总线速度

4 个模拟输入可编程为单端或差分输入

自动增量通道选择

模拟电压范围：VSS~VDD

片上跟踪与保持电路

8 位逐次逼近式 AD 转换

带一个模拟输出的乘法

3、引脚

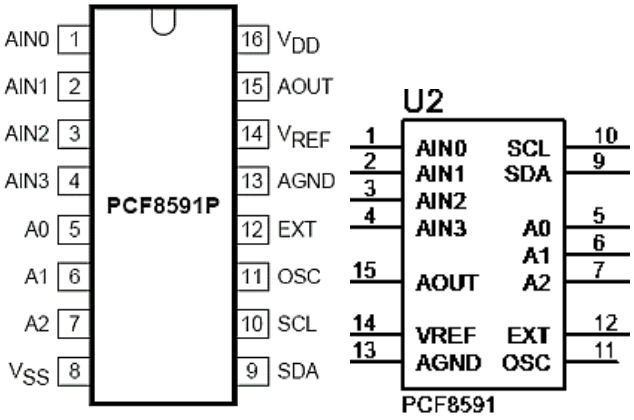


图 3-6 PCF8591 封装

表 3-1 PCF8591 引脚描述

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I^2C-bus data input/output
SCL	10	I^2C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

4、地址

I2C 总线系统中的每一篇 PCF8591 通过发送有效地址到该期间来激活。该地址包括固定部分和可编程部分。可编程部分必须根据引脚 A0，A1，A2 来设置。在 I2C 总线协议中地址必须是其实条件后作为第一个字节发送。地址字节的最后一位是用于设置以后数据传输方向的读/写位。

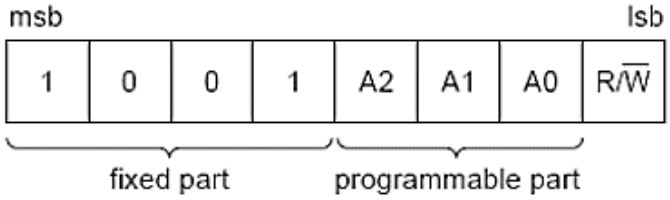


图 3-7 PCF8591 地址

5、控制字

发送到 PCF8591 的第二个字节将被存储在控制寄存器，用于控制器件功能。控制寄存器的高半字节用于允许模拟输出，和将模拟输入编程为单端或差分输入。低半字节选择一个由高半字节定义的模拟输入通道。如果自动增量（auto-increment）标志置 1，每次 A/D 转换后通道号将自动增加。

如果自动增量（auto-increment）模式是使用内部振荡器的应用中所需要的，那么控制字中模拟输出允许标志应置 1。这要求内部振荡器持续运行，因此要防止振荡器启动延时的转换错误结果。模拟输出允许标志可以在其他时候复位以减少静态功耗。

选择一个不存在的输入通道将导致分配最高可用的通道号。所以，如果自动增量（auto-increment）被置 1，下一个被选择的通道将总是通道 0。两个半字节的最高有效位（即 bit 7 和 bit 3）是留给未来的功能，必须设置为逻辑 0。控制寄存器的所有位在上电复位后被复位为逻辑 0。D/A 转换器和振荡器在节能时被禁止。模拟输出被切换到高阻态。

6、AD 转换

A/D 转换器采用逐次逼近转换技术。在 A/D 转换周期将临时使用片上 D/A 转换器和高增益比较器。 一个 A/D 转换周期总是开始于发送一个有效读模式地址给 PCF8591 之后。A/D 转换周期在应答时钟脉冲的后沿被触发，并在传输前一次转换结果时执行。

一旦一个转换周期被触发，所选通道的输入电压采样将保存到芯片并转换为对应的 8 位二进制码。

7、I2C 总线特性

I2C 总线是不同的 IC 或模块之间的双向两线通信。这两条线是串行数据线 SDA 和串行时钟线 SCL.这两条线必须通过上拉电路连接至正电源。数据传输只能在总线不忙时启动。

位传输：一个数据位在每一个时钟脉冲周期间传输。SDA 线上的数据必须在时钟脉冲的高电压期间保持稳定，这个期间数据线上的改变

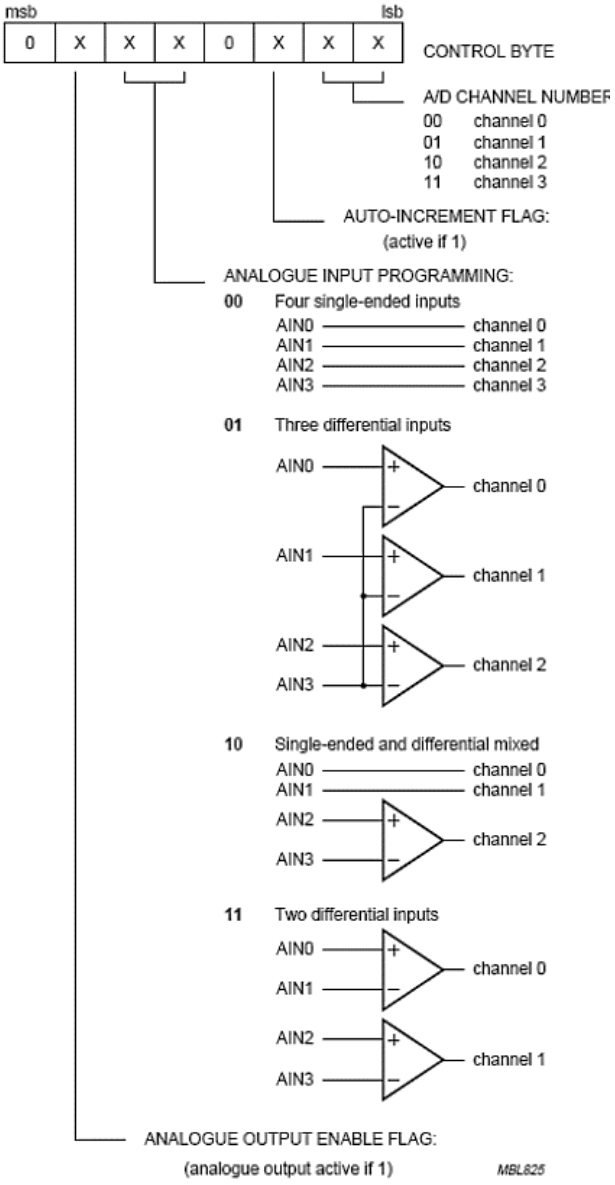


图 3-8 PCF8591 控制寄存器

原理图设计：

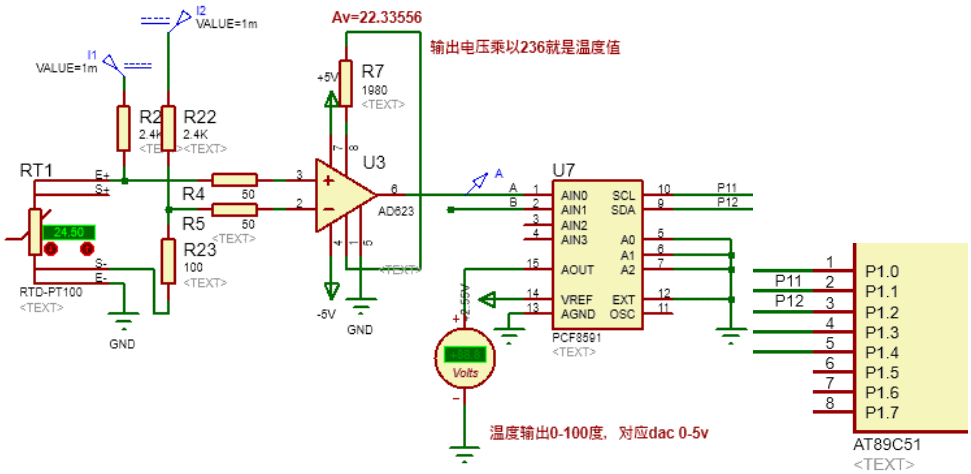


图 3-9 PCF8591 进行 AD 转换的原理图设计

3.2.4 LCD 显示屏

本系统显示部分采用 LCD 显示屏（Proteus 库中 LM016L）。

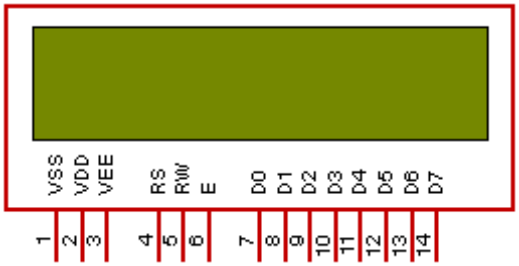


图 3-10 LCD 显示屏

工业字符型液晶，能够同时显示 16x02 即 32 个字符。(16 列 2 行)

1602 字符型 LCD 通常有 14 条引脚线或 16 条引脚线的 LCD，多出来的 2 条线是背光电源线。

VCC(15 脚)和地线 GND(16 脚)，其控制原理与 14 脚的 LCD 完全一样，其中：

表 3-3 LM016L 引脚功能描述

引脚	符号	功能说明
1	VSS	一般接地
2	VDD	接电源（+5V）
3	V0	液晶显示器对比度调整端，接正电源时对比度最弱，接地电源时对比度最高（对比度过高时会产生“鬼影”，使用时可以通过一个 10K 的电位器调整对比度）。
4	RS	RS 为寄存器选择，高电平 1 时选择数据寄存器、低电平 0 时选择指令寄存器。
5	R/W	R/W 为读写信号线，高电平(1)时进行读操作，低电平(0)时进行写操作。
6	E	E(或 EN)端为使能(enable)端， 写操作时，下降沿使能。 读操作时，E 高电平有效
7	DB0	低 4 位三态、 双向数据总线 0 位（最低位）
8	DB1	低 4 位三态、 双向数据总线 1 位
9	DB2	低 4 位三态、 双向数据总线 2 位
10	DB3	低 4 位三态、 双向数据总线 3 位
11	DB4	高 4 位三态、 双向数据总线 4 位
12	DB5	高 4 位三态、 双向数据总线 5 位
13	DB6	高 4 位三态、 双向数据总线 6 位
14	DB7	高 4 位三态、 双向数据总线 7 位（最高位）（也是 busy flag）
15	BLA	背光电源正极
16	BLK	背光 电源负极

表 3-4 寄存器选择控制表

RS	R/W	操作说明
0	0	写入指令寄存器（清除屏等）
0	1	读 busy flag（DB7），以及读取位址计数器（DB0~DB6）值
1	0	写入数据寄存器（显示各字型等）
1	1	从数据寄存器读取数据

注：关于 E=H 脉冲——开始时初始化 E 为 0，然后置 E 为 1，再清 0.

busy flag（DB7）：在此位为 1 时，LCD 忙，将无法再处理其他的指令要求。

原理图设计：

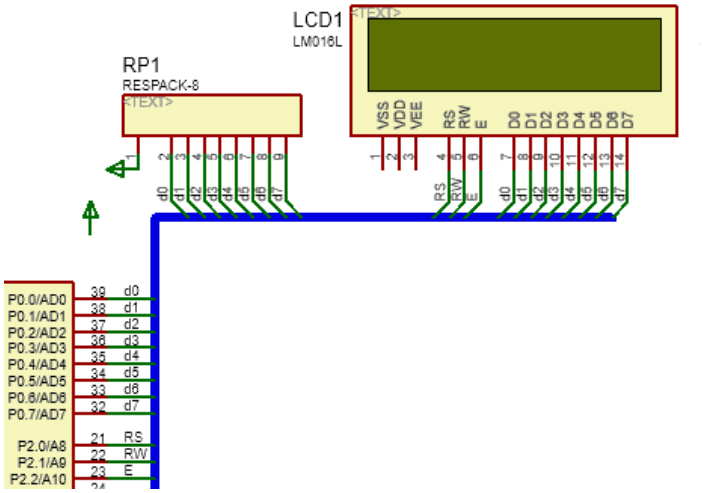


图 3-11 LCD 显示屏原理图

3.2.5 下位机串口调试

一、工具/原料

Virtual Serial Ports Driver 6.9（配置虚拟串口驱动工具）

串口调试助手 3.0

Proteus 8.0 Professional

Keil uVision4（生成单片机编译程序的编译器）

二、串口调试助手

步骤 1：运行虚拟串口程序，配置虚拟串口，这里我增加 COM2，COM3 两个虚拟串口。

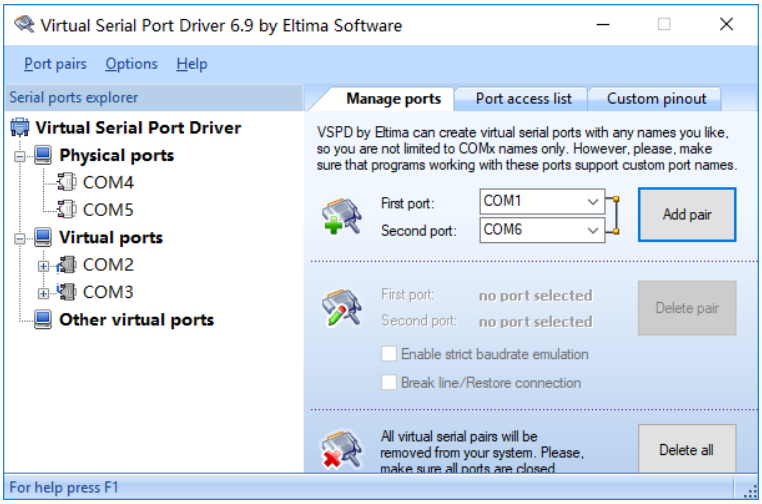


图 3-12 虚拟串口工具

在设备管理器中，也能感知到 COM2 和 COM3 的存在。

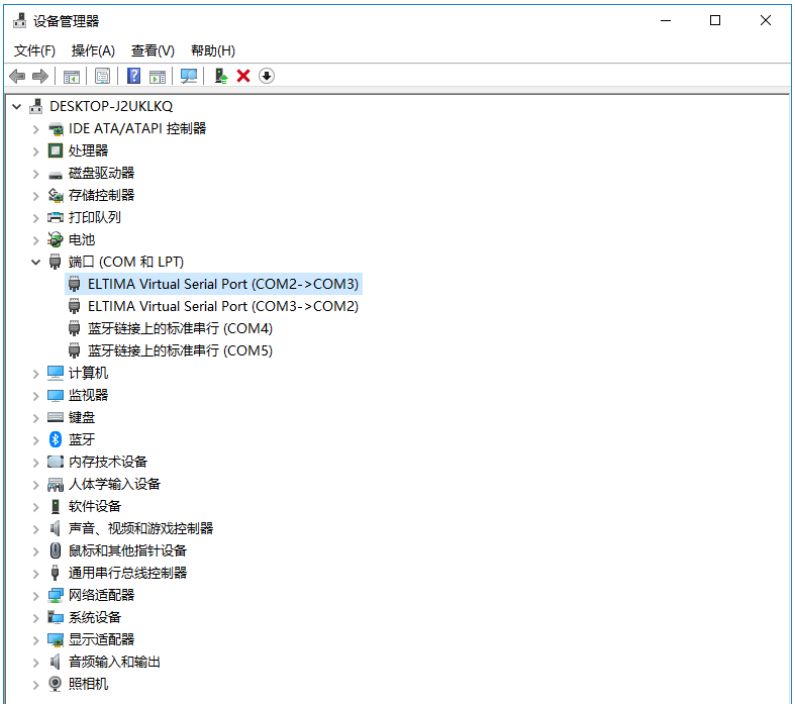


图 3-12 计算机设备管理器查看串口

步骤 2：运行两个串口调试助手实例，检查两者能否通信。

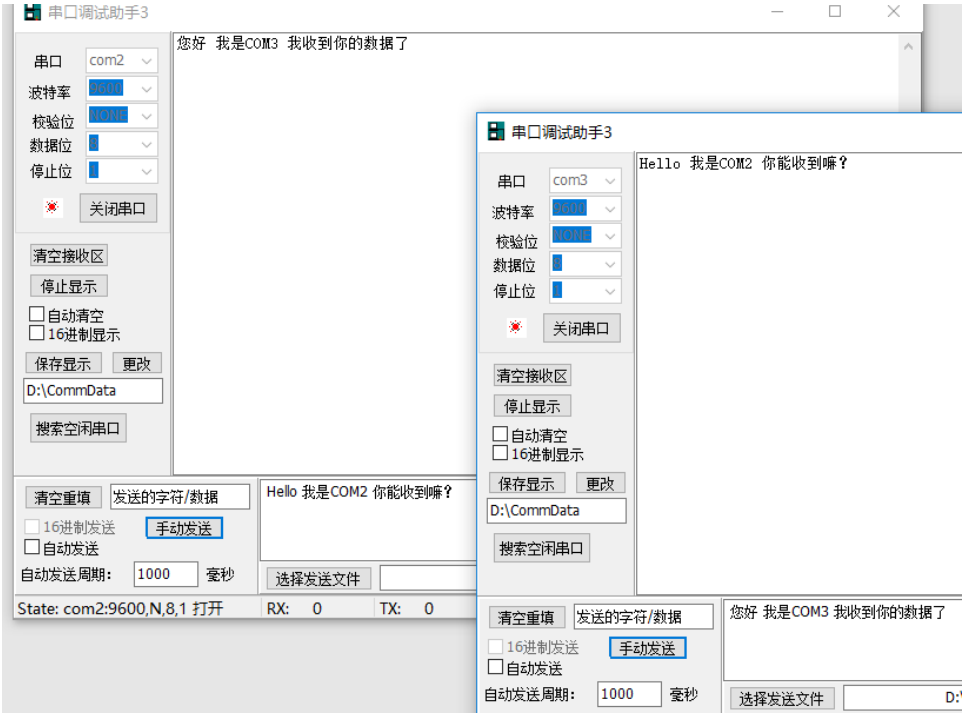


图 3-13 串口助手通信实例

三、下位机串口测试

步骤 1：打开 Proteus，按如下电路连接。

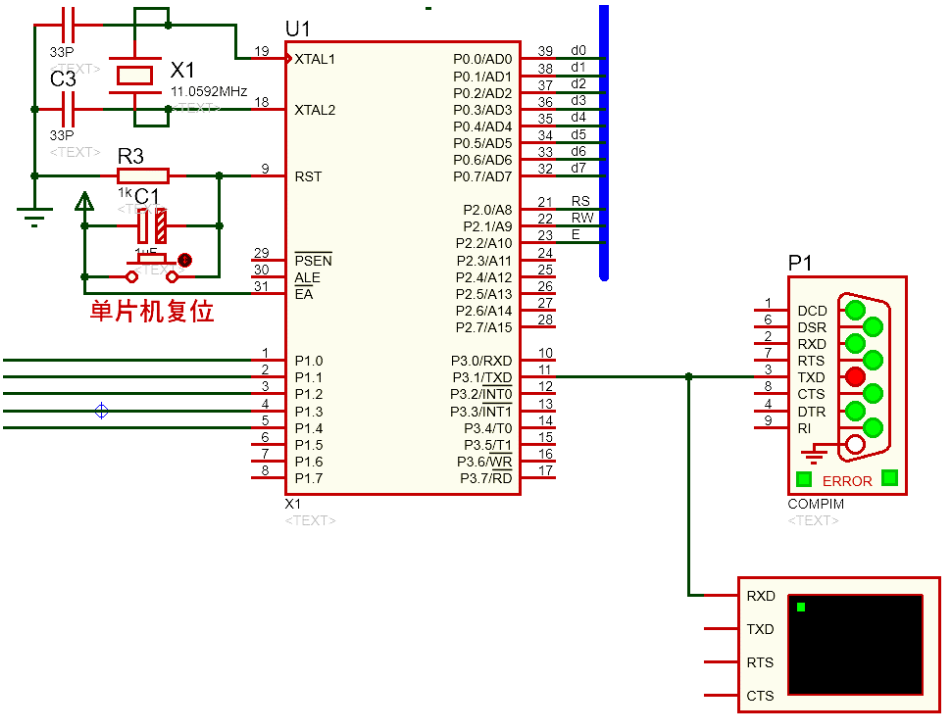


图 3-14 下位机串口通信原理图

步骤 2：设定外部晶振频率和单片机时钟频率，均设置为 11.0592MHz，方便产生 9600 波特率。Keil uVersion4 中的单片机频率也须设置为 11.0592MHz.

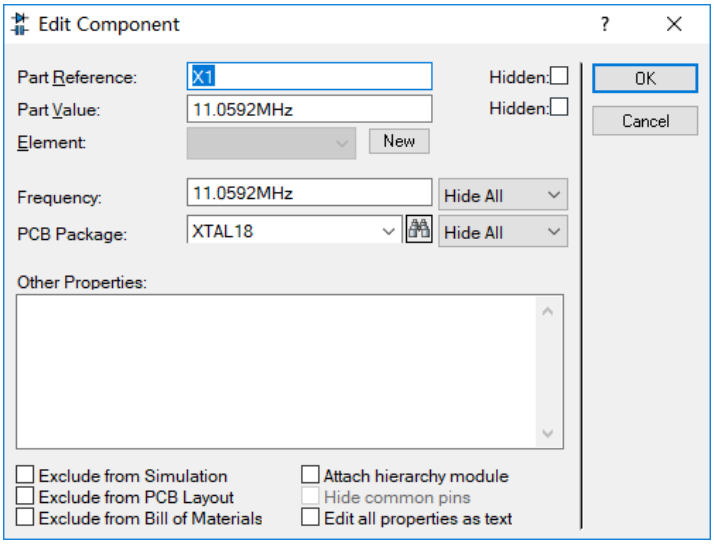


图 3-15 晶振频率设置

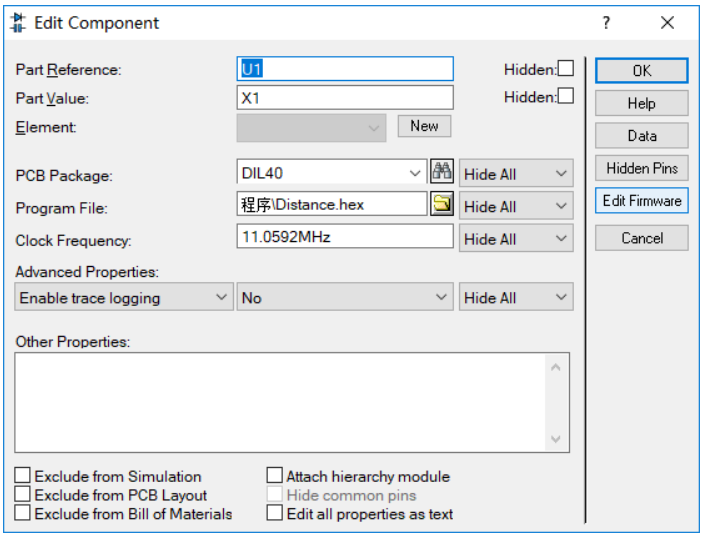


图 3-16 单片机时钟频率设置

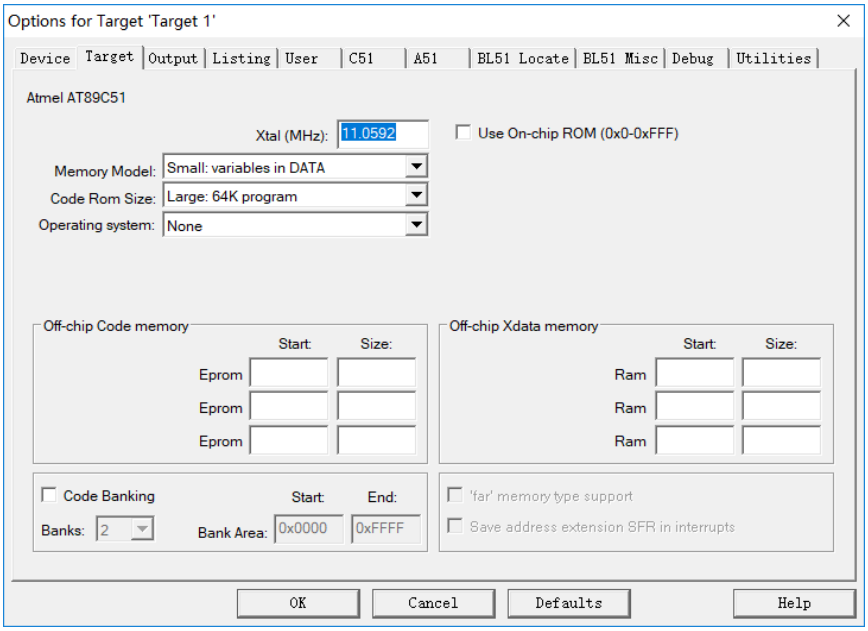


图 3-17 Keil uVersion4 单片机时钟频率设置

步骤 3：编写单片机串口通信程序

主函数部分核心代码：

```
    TMOD = 0x20;  //定时器 1 工作在 8 位自动重装模式，用于产生波特率
    TH1 = 0xFD;     //波特率 9600
    TL1 = 0xFD;
    SCON = 0x50;   //设定串行口工作方式
    PCON = 0x00;
    TR1 = 1;        //启动定时器 1
    IE = 0x0;     //禁止中断
    ES = 1;
    EA = 1;
```

发送单个字符：

```
void send_char( unsigned char ch ){
    SBUF = ch;
    while(!TI);
    TI = 0;
}
```

四、下位机与串口调试助手通信

将下位机串口设置为 COM3，串口助手串口设置为 COM2.

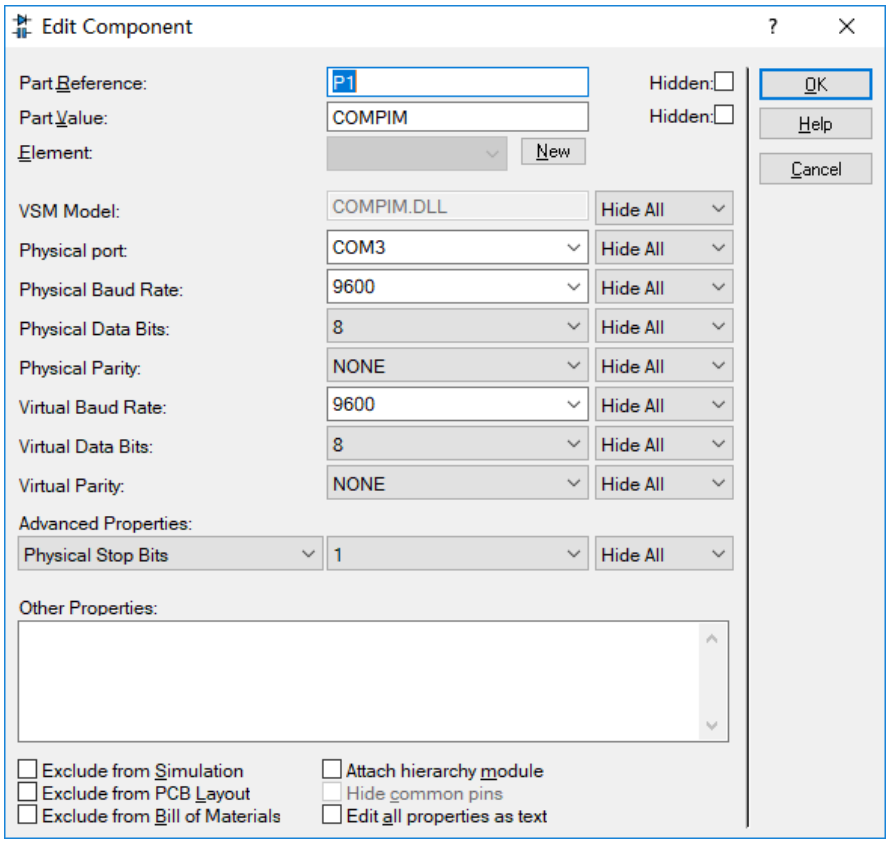


图 3-18 下位机串口参数设置

图 3-19 实现下位机与串口通信实例

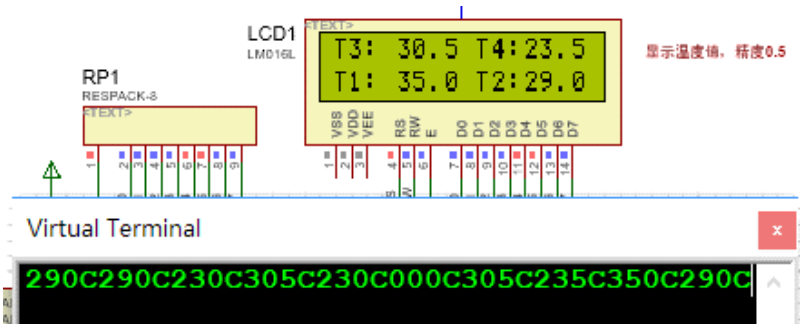
3.2.6 通信规约

为了方便上位机对接收到的字符串进行信息的提取，上位机与下位机双方约定：以字符串“000C”为起始标志，随后，每三个有效数字作为温度的有效值，每一个温度值以 C 结束，一共 8 个温度值。

例如：000C305C235C350C290C290C230C305C230C

表示：30.5 度，23.5 度，35.0 度，29.0 度，23.0 度，30.5 度，23.0 度

3.2.7 下位机运行演示



LCD 分时地显示 8 个温度数据

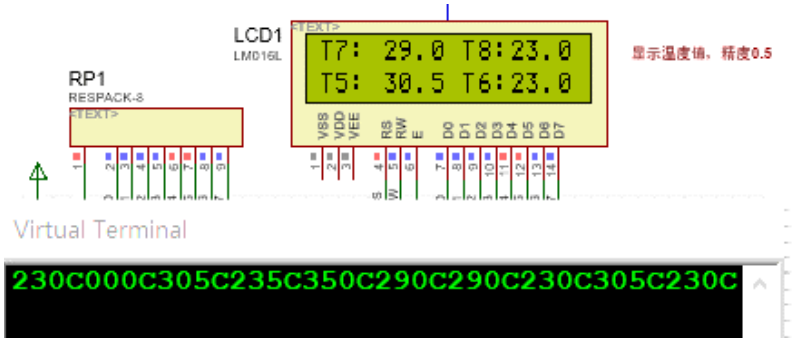


图 3-20 下位机演示

3.3 上位机

3.3.1 上位机与串口助手通信

使用类似方法，进行上位机与串口助手的通信，调试上位机是否能成功接收数据。将上位机端口设置为 COM2，波特率设置为 9600.

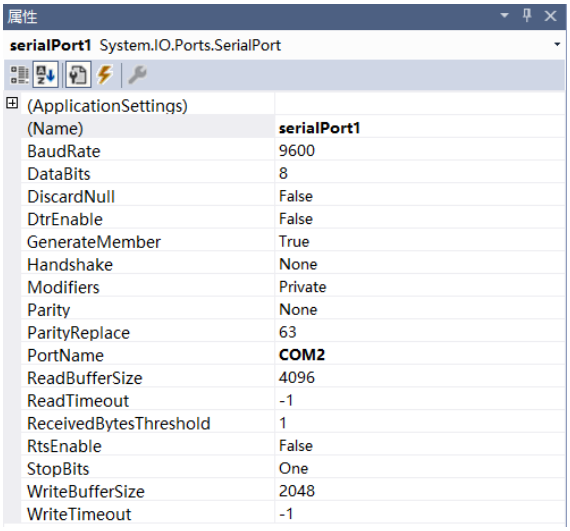


图 3-21 上位机串口参数设置

添加串口的事件监听器。

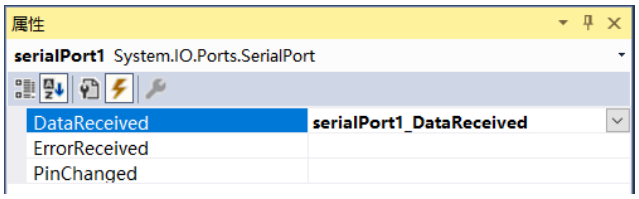


图 3-22 上位机串口参数设置

事件监听函数核心代码：

```
string str_data = null;
private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e) {
    if (serialPort1.BytesToRead >= 43)  //累计43个字符再进行处理
    {
        Byte[] Data = new Byte[serialPort1.BytesToRead];
        //创建字符串，大小为缓冲区已收到的数据大小
        serialPort1.Read(Data, 0, Data.Length);
        //提取缓冲区中所有数据
        string str = System.Text.Encoding.ASCII.GetString(Data);
        //转化为ASCII码
        str_data = str_data + str;
        serialPort1.DiscardInBuffer(); //清空缓冲区
    }
}
```

3.3.2 上位机与下位机实现通信

特别注意：若下位机发送太快，上位机可能会丢失数据，因此，没发一个字符，可以适当延时几毫秒。

3.3.3 上位机串口接收缓冲区字符串提取与处理

考虑到操作系统按块读写磁盘，为了避免系统抖动，充分利用局部性原理，提高上位机整体性能，我们等待上位机数据缓冲区所接受到的数据超过64个字符再进行字符串的接收和处理。我们一次性接收整个缓冲区的数据，并清空缓冲区。

在对字符串进行起始标志的判断，以及温度信息的提取时，要注意数组的越界问题。

核心代码：

```
int i;
for (i = 0; i <= str_data.Length - 1 - 3 && i + 40 < str_data.Length; i++)
                {
if (str_data[i] == '0' && str_data[i + 1] == '0' && str_data[i + 2] == '0' &&
str_data[i + 3] == 'C') //接收到起始符
                        {
                            //开始接受温度值
                            temp3 = " ";
                            temp3 += str_data[i + 4];
                            temp3 += str_data[i + 5];
                            temp3 += '.';
                            temp3 += str_data[i + 6];
                            temp3 += " Celsius degree";
                            textBox4.Text = temp3;  //更新温度值
                            textBox1.AppendText(temp3 + "\r\n");

                            temp4 = " ";
                            temp4 += str_data[i + 8];
                            temp4 += str_data[i + 9];
                            temp4 += '.';
                            temp4 += str_data[i + 10];
                            temp4 += " Celsius degree";
                            textBox5.Text = temp4;
                            textBox1.AppendText(temp4 + "\r\n");

                            temp1 = " ";
                            temp1 += str_data[i + 12];
                            temp1 += str_data[i + 13];
                            temp1 += '.';
                            temp1 += str_data[i + 14];
```

```
temp1 += " Celsius degree";
textBox2.Text = temp1;
textBox1.AppendText(temp1 + "\r\n");

temp2 = " ";
temp2 += str_data[i + 16];
temp2 += str_data[i + 17];
temp2 += '.';
temp2 += str_data[i + 18];
temp2 += " Celsius degree";
textBox3.Text = temp2;
textBox1.AppendText(temp2 + "\r\n");

temp7 = " ";
temp7 += str_data[i + 20];
temp7 += str_data[i + 21];
temp7 += '.';
temp7 += str_data[i + 22];
temp7 += " Celsius degree";
textBox8.Text = temp7;
textBox1.AppendText(temp7 +"\r\n");

temp8 = " ";
temp8 += str_data[i + 24];
temp8 += str_data[i + 25];
temp8 += '.';
temp8 += str_data[i + 26];
temp8 += " Celsius degree";
textBox9.Text = temp8;
textBox1.AppendText(temp8 + "\r\n");

temp5 = " ";
temp5 += str_data[i + 28];
temp5 += str_data[i + 29];
temp5 += '.';
temp5 += str_data[i + 30];
temp5 += " Celsius degree";
textBox6.Text = temp5;
textBox1.AppendText(temp5 + "\r\n");

temp6 = " ";
temp6 += str_data[i + 32];
temp6 += str_data[i + 33];
temp6 += '.';
```

```
                temp6 += str_data[i + 34];
                temp6 += " Celsius degree";
                textBox7.Text = temp6;
                textBox1.AppendText(temp6 + "\r\n");

                str_data = "";

                /*******************/
                   //此处省略更新柱状图和曲线图的代码
                /*****************/
                break;
            }
        }
```

3.3.4 上位机界面演示

图 3-23 上位机界面演示

进行数据传输时，要先打开上位机串口。

3.4 成果展示

运行 Protues 工程和上位机 C#工程，C#工程打开上位机串口，即可实现温度显示。

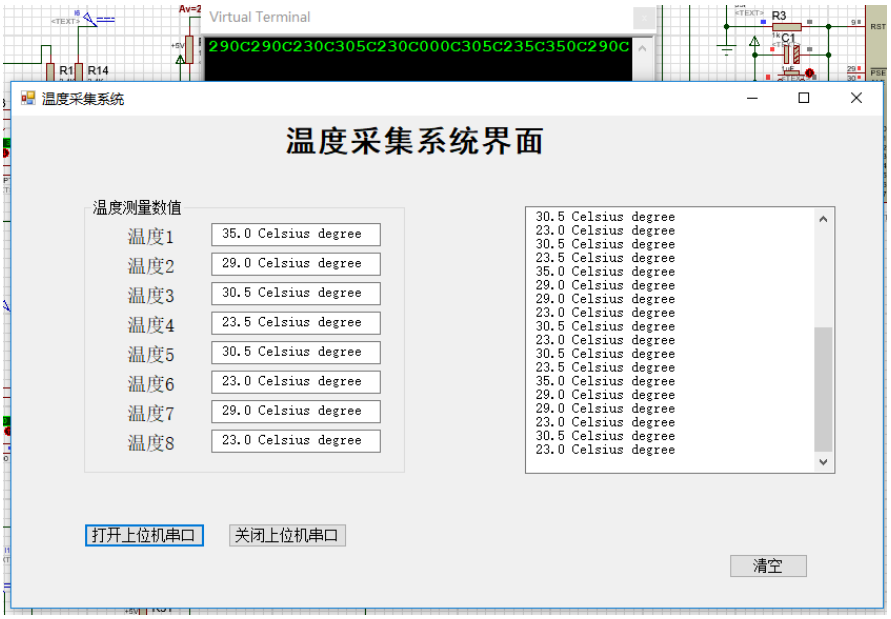


图 3-24 成果展示 1

改变某个热敏电阻的温度，如温度 6，产生相应的变化：

图 3-25 成果展示 2

3.5 上位机版本更新

加入了动态柱状图和折线图。

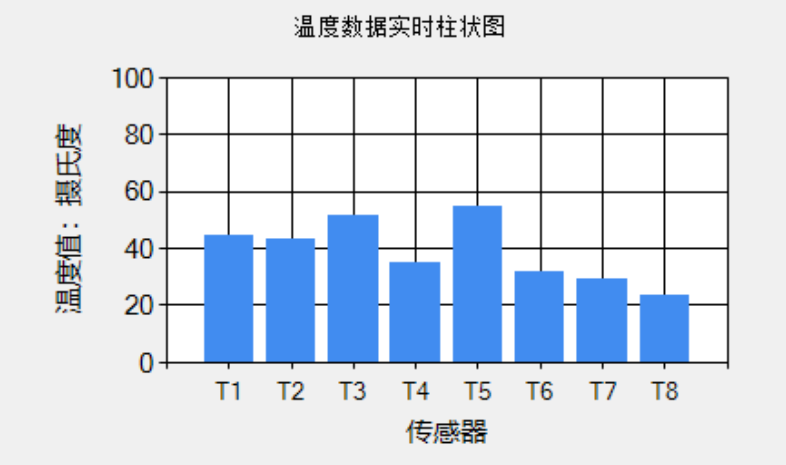


图 3-26 柱状图

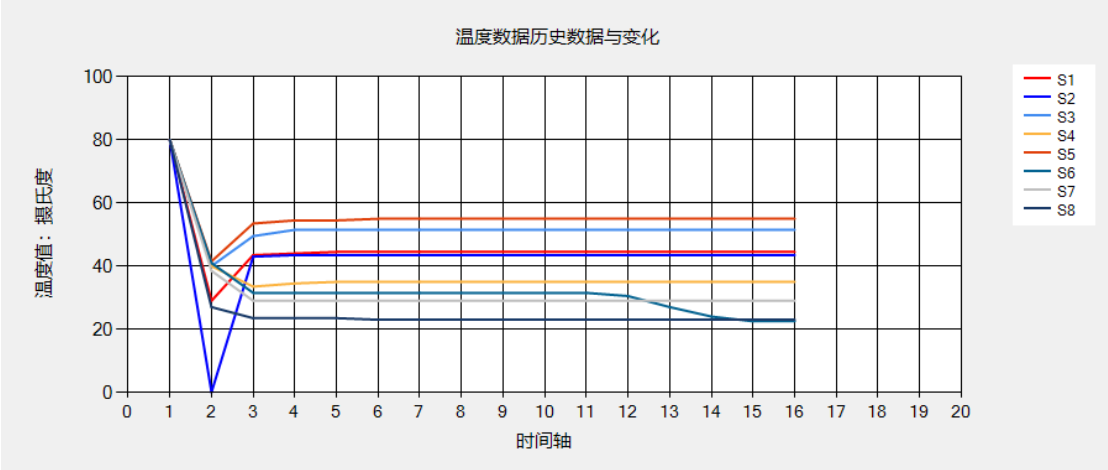


图 3-27 曲线图

调节 Proteus 中的传感器温度，折线图中 S6 会引起相应变化。

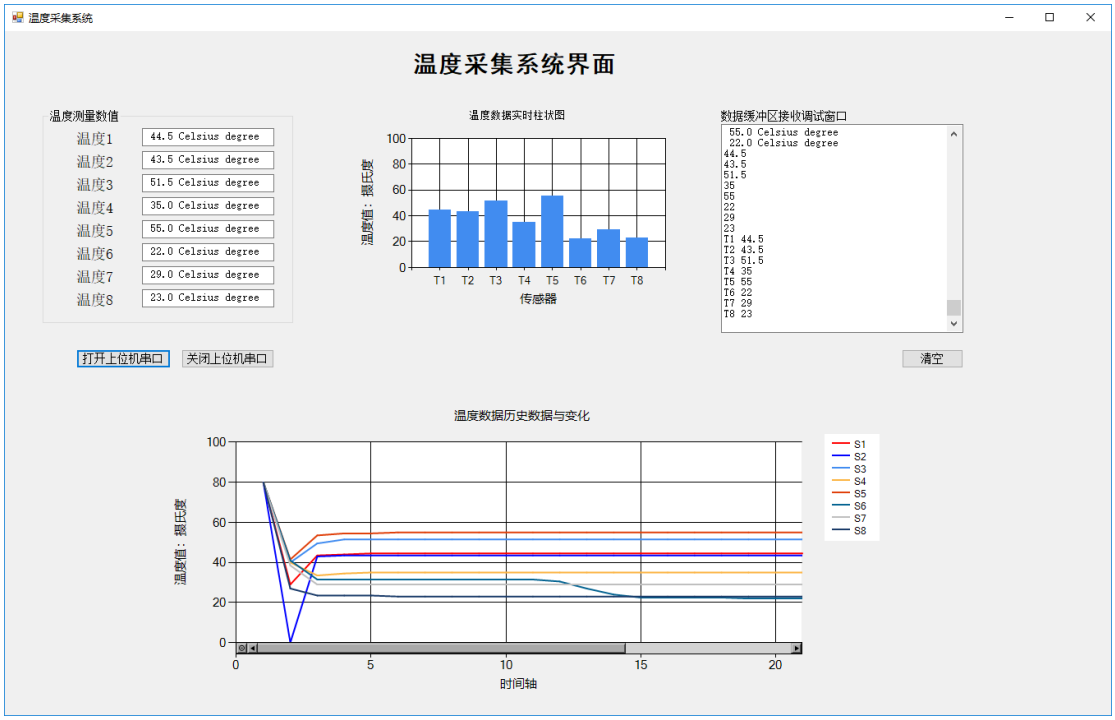


图 3-28 整体设计

主要代码：

```
chart1.Titles.Add("温度数据实时柱状图");
//chart1.Titles[0].ForeColor = Color.White;
chart1.Titles[0].Font = new Font("微软雅黑", 10f, FontStyle.Regular);
chart1Init();
//X坐标轴标题
chart1.ChartAreas[0].AxisX.Title = "传感器";
chart1.ChartAreas[0].AxisX.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);
//Y坐标轴标题
chart1.ChartAreas[0].AxisY.Title = "温度值：摄氏度";
chart1.ChartAreas[0].AxisY.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);
chart2.Titles.Add("温度数据历史数据与变化");
//chart1.Titles[0].ForeColor = Color.White;
chart2.Titles[0].Font = new Font("微软雅黑", 10f, FontStyle.Regular);

//X坐标轴标题
chart2.ChartAreas[0].AxisX.Title = "时间轴";
chart2.ChartAreas[0].AxisX.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);

//Y坐标轴标题
chart2.ChartAreas[0].AxisY.Title = "温度值：摄氏度";
```

```
    chart2.ChartAreas[0].AxisY.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);
    chart2.ChartAreas[0].AxisY.Maximum = 100;
    chart2.Series.Clear();
    mySeries1.ChartType = SeriesChartType.FastLine; //绘制折线图
    mySeries1.BorderWidth = 2;                      //线条粗细
    mySeries1.Color = System.Drawing.Color.Red;     //设置线条颜色
    chart2.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = true;  //滚动条在曲线图内
    chart2.ChartAreas[0].AxisX.ScrollBar.Enabled = true;  //显示滚动条
    chart2.ChartAreas[0].AxisX.ScaleView.Size = 20;        //显示当前最大点数

    chart2.Series.Add( mySeries1 );
    chart2.Series.Add( mySeries2 );
    chart2.Series.Add( mySeries3 );
    chart2.Series.Add( mySeries4 );
    chart2.Series.Add( mySeries5 );
    chart2.Series.Add( mySeries6 );
    chart2.Series.Add( mySeries7 );
    chart2.Series.Add( mySeries8 );
    chart1.Series.Clear();  //清楚所有柱子
    Series mySeries = new Series("温度"); //重新创建

    for( int i = 0; i <= 7; i++)
    {
        textBox1.AppendText(dataX[i]+" ");   //数据区输出查看
        textBox1.AppendText(dataY[i].ToString() + "\r\n"); //数据区输出查看
        mySeries.Points.AddXY(dataX[i], dataY[i].ToString()); //重新添加柱状图
    }

    chart1.Series.Add(mySeries);  //将系列加入chart
    chart1.Series.FindByName("温度").IsVisibleInLegend = false; //不显示图例
```

4 Future work and weakness

1) 该系统**实时性不强**，有几方面原因：首先，在下位机处理数据时，有适当的延时，芯片之间配合工作；此外，为了让 LCD 显示屏显示 8 个数据，采用了分时的方法，让发送的过程有了 2 秒的延时；最后，在上位机接收下位机的数据时，发送太快，有时候会丢失字符，因此，下位机每发一个字符，适当延时了 20 毫秒。
2) 该系统没有估算**硬件成本**。
3) 上位机的**软件设计功能可以更加丰富，界面可以更加友好**。比如：加入串口和波特率的选择。
4) 该系统**稳定性不强**，程序容易跑飞，在本文第 5 部分详述。

5 Difficulties and Proposals

一、遇到的困难

在收官阶段，遇到一个问题，在程序运行 2 分钟左右后，容易跑飞，出现以下 Bug——在 LCD 的 Dx 引脚检测到逻辑竞争，但是经过网上寻找，多方查找答案，还是没有结果。

后来经过单片机程序软件上的调试，把一大段密集的分支语句注释了，情况有所好转，可以连续运行半个多小时，但是依然会出现跑飞的情况。

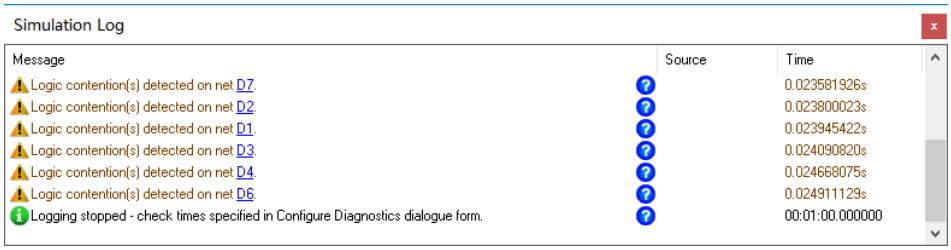


图 5-1 调试时出现的无法解决的错误

```
567         //这一段容易引发程序跑飞
568         /*if(temp1<50||temp1>150)BEEP=0;//温度超出时报警
569         else BEEP=1;
570         if(temp2<50||temp2>150)BEEP=0;//温度超出时报警
571         else BEEP=1;
572         if(temp3<50||temp3>150)BEEP=0;//温度超出时报警
573         else BEEP=1;
574         if(temp4<50||temp4>150)BEEP=0;//温度超出时报警
575         else BEEP=1;
576         if(temp5<50||temp5>150)BEEP=0;//温度超出时报警
577         else BEEP=1;
578         if(temp6<50||temp6>150)BEEP=0;//温度超出时报警
579         else BEEP=1;
580         if(temp7<50||temp7>150)BEEP=0;//温度超出时报警
581         else BEEP=1;
582         if(temp8<50||temp8>150)BEEP=0;//温度超出时报警
583         else BEEP=1;   */
```

图 5-2 调试时注释掉的 Keil 程序

二、建议

在进行界面设计的时候遇到诸多兼容性问题，win7、win8 系统能打开的 vs2005 工程，win10 不能打开或不能正常运行，win10 的 vs2017 或 vs2010 工程，vs2005 缺不能打开，影响了代码的前后兼容性，加大了开发难度，建议实验室使用新版的操作系统和 Visual Studio.

6 References

【1】 **Pt100 热电阻**，百度百科
https://baike.baidu.com/item/Pt100 热电阻/2572473?fr=aladdin

【2】 **PCF8591 中文数据手册**，百度文库
https://wenku.baidu.com/view/dc76007fdd3383c4bb4cd2e3.html

【3】 **PT100+51 单片机的温控系统程序+仿真图**，51 黑论
http://www.51hei.com/bbs/dpj-94135-1.html

【4】 **虚拟串口在 Proteus 中的使用**，百度经验
https://jingyan.baidu.com/article/5553fa82c615ba65a3393471.html

【5】 **C# 编写 Windows 窗体程序详解 C# 上位机实战开发指南**，51 黑论坛
http://www.51hei.com/bbs/dpj-117592-1.html

【6】 **C#之 Chart 篇**, CSDN 博客 https://blog.csdn.net/kang_xiong/article/details/53944487

【7】 **C# Chart 控件，chart、Series、ChartArea 曲线图绘制的重要属性**，新浪博客
http://blog.sina.com.cn/s/blog_621e24e20101cp64.html

【8】 **C#绘制动态折线图**，CSDN 博客
https://blog.csdn.net/mading0613/article/details/60145571

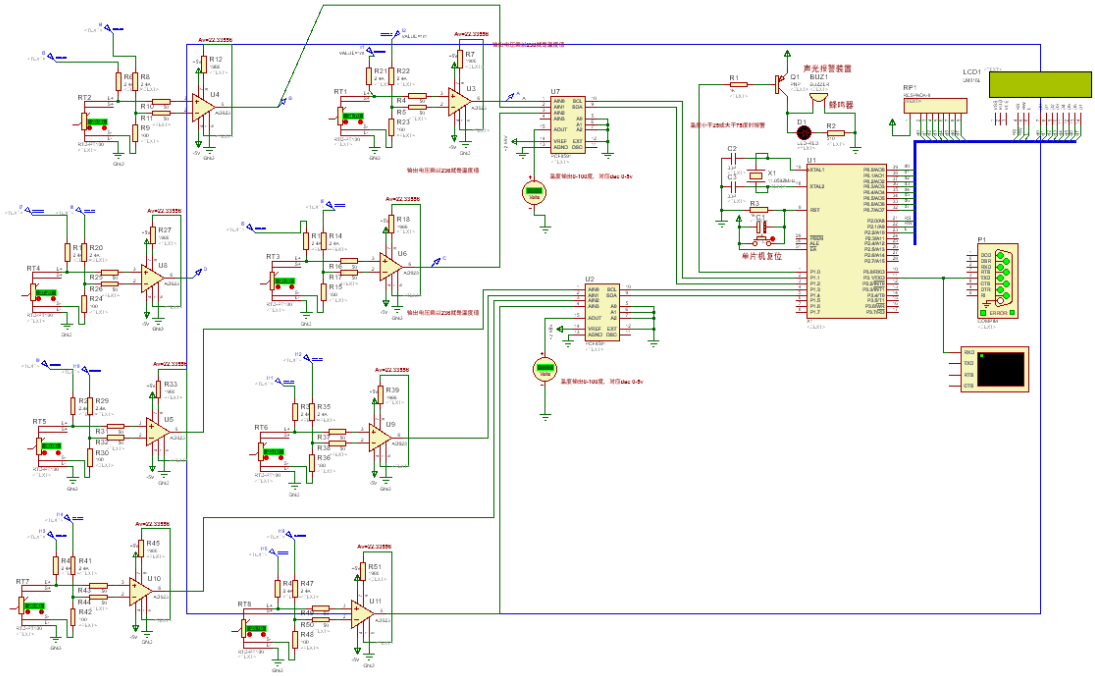
Applendix

附录清单

附录 1：下位机原理图设计；
附录 2：下位机程序设计；
附录 3：上位机工程与程序设计；

具体附录

附录 1：下位机原理图设计



附录 2：下位机程序设计

```
#include <reg52.h>
#include <intrins.h>
#define uint unsigned int
#define uchar unsigned char
#define NOP() _nop_()
#define Delay5us() {_nop_();_nop_();_nop_();_nop_();_nop_();}

#define LCD_LINE_ONE 1     //函数参数 表示要写 LCD 的第几行
#define LCD_LINE_TWO 2

sbit LCD_RS = P2^0 ;  // LCD 的三个控制引脚 RS，RW，EN
sbit LCD_RW = P2^1 ;
sbit LCD_EN = P2^2 ;

sbit SCL = P1^1;  // PCF8591 传感器 0 I2C bus data input/output
sbit SDA = P1^2;  // PCF8591 传感器 0 I2C bus clock input

sbit SCL1 = P1^3; // PCF8591 传感器 1 I2C bus data input/output
sbit SDA1 = P1^4;// PCF8591 传感器 1 I2C bus clock input

sbit BEEP = P1^0; //蜂鸣器控制引脚
uchar temp1 = 40; //暂存温度变量
uchar temp2 = 40;
uchar temp3 = 40;
uchar temp4 = 40;
uchar temp5 = 40;
uchar temp6 = 40;
uchar temp7 = 40;
uchar temp8 = 40;

uint Voltage[]={'0','0','0'};                    //AD 的 LCD 显示值
unsigned char LCD_Line_1[] = {"T3:                 "};//TEMP1-是温度值
unsigned char LCD_Line_2[] = {"T1:                 "};
unsigned char LCD_Line_3[] = {"T7:                 "};
unsigned char LCD_Line_4[] = {"T5:                 "};

extern void Convert_To_Voltage(uint val);          //函数声明 数字量转化为要显示的电压
uchar IIC_ERROR;                                   //IIC 错误标志

void send_char( unsigned char ch );
```

```
void Delay(uint ms)                    //较为精确的软件延时，参数的单位为毫秒
{
    uchar i;
    while(ms--)
    {
        for(i=0;i<120;i++);
    }
}

bit LCD_Busy_Check()              //检测 LCD 忙信号
{
    bit Result;
    LCD_RS = 0;
    LCD_RW = 1;
    LCD_EN = 1;
    Delay5us();
    Result = (bit)(P0&0x80);
    LCD_EN = 0;
    return Result;
}

void LCD_Write_Command(uchar cmd) //LCD 写命令
{
    while(LCD_Busy_Check());
    LCD_RS = 0;
    LCD_RW = 0;
    LCD_EN = 0;
    _nop_();
    _nop_();
    P0 = cmd;
    Delay5us();
    LCD_EN = 1;
    Delay5us();
    LCD_EN = 0;
}

void LCD_Write_Data(uchar dat) //LCD 写数据
{
    while(LCD_Busy_Check());
    LCD_RS = 1;
    LCD_RW = 0;
    LCD_EN = 0;
    P0 = dat;
    Delay5us();
```

```
    LCD_EN = 1;
    Delay5us();
    LCD_EN = 0;
}

void LCD_Set_Position(uchar pos)          //LCD 显示位置设置
{
    if(pos == LCD_LINE_ONE)
        LCD_Write_Command(0x80);          //写第一行
    else
        LCD_Write_Command(0xc0);          //写第二行
}

void LCD_Display_A_Line(uchar Line_Addr,uchar s[])//LCD 显示行
{
    uchar i;
    LCD_Set_Position(Line_Addr);
    for(i=0;i<16;i++)
    {
        LCD_Write_Data(s[i]);
    }
}

void LCD_Dispay(void)                             //LCD 实时显示
{
    Convert_To_Voltage(temp1*5);
                          //适当延时，防止上位机丢失数据
    send_char('0');
    Delay(40);
    send_char('0');
    Delay(40);
    send_char('0');
    Delay(40);
    send_char('C');
    Delay(40);

    LCD_Line_1[4]= Voltage[2];
    LCD_Line_1[5]= Voltage[1];
    LCD_Line_1[6]= '.';
    LCD_Line_1[7]= Voltage[0];                    //显示到 T3 位置去了

    //发送 T3
    send_char(Voltage[2]);
    Delay(40);
```

```
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
LCD_Display_A_Line(LCD_LINE_ONE,LCD_Line_1);//温度显示

Convert_To_Voltage(temp2*5);
LCD_Line_1[9]= 'T';
LCD_Line_1[10]= '4';
LCD_Line_1[11]= ':';
LCD_Line_1[12]= Voltage[2];
LCD_Line_1[13]= Voltage[1];
LCD_Line_1[14]= '.';
LCD_Line_1[15]= Voltage[0];

//发送 T4
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
                    //显示到 T4 位置去了
LCD_Display_A_Line(LCD_LINE_ONE,LCD_Line_1);

Convert_To_Voltage(temp3*5);
LCD_Line_2[4]= Voltage[2];
LCD_Line_2[5]= Voltage[1];
LCD_Line_2[6]= '.';
LCD_Line_2[7]= Voltage[0];                          //显示到 T1 位置去了

//发送 T1
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
```

```
LCD_Display_A_Line(LCD_LINE_TWO,LCD_Line_2);

Convert_To_Voltage(temp4*5);
LCD_Line_2[9]= 'T';
LCD_Line_2[10]= '2';
LCD_Line_2[11]= ':';
LCD_Line_2[12]= Voltage[2];
LCD_Line_2[13]= Voltage[1];
LCD_Line_2[14]= '.';
LCD_Line_2[15]= Voltage[0];

//发送 T2
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
                //T2 去了
LCD_Display_A_Line(LCD_LINE_TWO,LCD_Line_2);

 //分时显示到 LCD 显示屏上

Convert_To_Voltage(temp5*5);
LCD_Line_3[4]= Voltage[2];
LCD_Line_3[5]= Voltage[1];
LCD_Line_3[6]= '.';
LCD_Line_3[7]= Voltage[0];

//发送 T7
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
LCD_Display_A_Line(LCD_LINE_ONE,LCD_Line_3);    //T7 去了

Convert_To_Voltage(temp6*5);
```

```
LCD_Line_3[9]= 'T';
LCD_Line_3[10]= '8';
LCD_Line_3[11]= ':';
LCD_Line_3[12]= Voltage[2];
LCD_Line_3[13]= Voltage[1];
LCD_Line_3[14]= '.';
LCD_Line_3[15]= Voltage[0];

//发送 T8
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
LCD_Display_A_Line(LCD_LINE_ONE,LCD_Line_3);

Convert_To_Voltage(temp7*5);

LCD_Line_4[4]= Voltage[2];
LCD_Line_4[5]= Voltage[1];
LCD_Line_4[6]= '.';
LCD_Line_4[7]= Voltage[0];

//发送 T5
send_char(Voltage[2]);
Delay(40);
send_char(Voltage[1]);
Delay(40);
send_char(Voltage[0]);
Delay(40);
send_char('C');
Delay(40);
LCD_Display_A_Line(LCD_LINE_TWO,LCD_Line_4);

Convert_To_Voltage(temp8*5);
LCD_Line_4[9]= 'T';
LCD_Line_4[10]= '6';
LCD_Line_4[11]= ':';
LCD_Line_4[12]= Voltage[2];
LCD_Line_4[13]= Voltage[1];
LCD_Line_4[14]= '.';
```

```
    LCD_Line_4[15]= Voltage[0];

    //发送 T6
    send_char(Voltage[2]);
    Delay(40);
    send_char(Voltage[1]);
    Delay(40);
    send_char(Voltage[0]);
    Delay(40);
    send_char('C');
    Delay(40);

    LCD_Display_A_Line(LCD_LINE_TWO,LCD_Line_4);
}

void LCD_Initialise()                                   //LCD 初始化
{
    LCD_Write_Command(0x38);Delay(5);
    LCD_Write_Command(0x0c);Delay(5);
    LCD_Write_Command(0x06);Delay(5);
    LCD_Write_Command(0x01);Delay(5);

    LCD_Display_A_Line(1,LCD_Line_1);
    LCD_Display_A_Line(2,LCD_Line_2);
}

/*******************************PCF8591 部分***********************************/
void Convert_To_Voltage(uint val)//电压换算成温度或压力
{
    uint Tmp;
    Tmp = val;
    Voltage[2] = Tmp/100+'0';
    Tmp = Tmp%100*10;
    Voltage[1] = Tmp/100+'0';
    Tmp = Tmp%100*10;
    Voltage[0] = Tmp/100+'0';
}

/*****************IIC 初始化*****************/
void delay()
{;;}
```

```
void start() //PCF8591 开始传输的信号
{
    SDA=1;
    delay();
    SCL=1;
    delay();
    SDA=0;
    delay();
}

void stop()    //PCF8591LCD 结束传输的信号
{
    SDA=0;
    delay();
    SCL=1;
    delay();
    SDA=1;
    delay();
}

void respons()//应答信号
{
    uchar i;
    SCL=1;
    delay();
    while((SDA==1)&&(i<250))
    i++;
    SCL=0;
    delay();
}

void init()        //初始化 PCF8591 接口
{
    SDA=1;
    delay();
    SCL=1;
    delay();
}

uchar read_byte()//对 PCF8591 读一个字节数据
{
    uchar i,k;
    SCL=0;
    delay();
```

```
    SDA=1;
    delay();
    for(i=0;i<8;i++)
    {
        SCL=1;   //允许读取数据
        delay();
        k=(k<<1)|SDA;
        SCL=0;    //允许数据变化
        delay();
    }
    return k;
}

void write_byte(uchar date)   //对 PCF8591 写一个字节数据
{
    uchar i,temp1;
    temp1=date;
    for(i=0;i<8;i++)
    {
        temp1=temp1<<1; //  左移溢出时会影响 CY
        SCL=0;              //允许数据变化
        delay();
        SDA=CY;             //  利用 CY 控制写入什么数据
        delay();
        SCL=1;              //数据保持有效  SDA 写入芯片
        delay();
    }
    SCL=0;
    delay();
    SDA=1;
    delay();
}

void write_add(uchar control,uchar date)//输出电压
{
    start();
    write_byte(0x90);
    respons();
    write_byte(control);
    respons();
    write_byte(date);
    respons();
    stop();
```

```
}

uchar read_add(uchar control)//读取 AD 转换值
{
    uchar date;
    start();
    write_byte(0x90);    // 地址 1001 A2A1A0 R/W_
    respons();
    write_byte(control); //控制字
    respons();
    start();
    write_byte(0x90+1);    // 地址 1001 A2A1A0 R/W_
    respons();
    date=read_byte();
    stop();
    return date;
}

void start1()
{
    SDA1=1;
    delay();
    SCL1=1;
    delay();
    SDA1=0;
    delay();
}

void stop1()
{
    SDA1=0;
    delay();
    SCL1=1;
    delay();
    SDA1=1;
    delay();
}


void respons1()//应答信号
{
    uchar i;
    SCL1=1;
    delay();
```

```
    while((SDA1==1)&&(i<250))
    i++;
    SCL1=0;
    delay();
}

void init1()          //初始化 PCF8591 接口
{
    SDA1=1;
    delay();
    SCL1=1;
    delay();
}

uchar read_byte1()//对 PCF8591 读一个字节数据
{
    uchar i,k;
    SCL1=0;
    delay();
    SDA1=1;
    delay();
    for(i=0;i<8;i++)
    {
        SCL1=1; //允许读取数据
        delay();
        k=(k<<1)|SDA1;
        SCL1=0;   //允许数据变化
        delay();
    }
    return k;
}

void write_byte1(uchar date)   //对 PCF8591 写一个字节数据
{
    uchar i,temp1;
    temp1=date;
    for(i=0;i<8;i++)
    {
        temp1=temp1<<1; //  左移溢出时会影响 CY
        SCL1=0;                  //允许数据变化
        delay();
        SDA1=CY;                 //  利用 CY 控制写入什么数据
        delay();
        SCL1=1;              //数据保持有效  SDA 写入芯片
```

```
            delay();
        }
        SCL1=0;
        delay();
        SDA1=1;
        delay();
}

void write_add1(uchar control,uchar date)//输出电压
{
        start1();
        write_byte1(0x90);
        respons1();
        write_byte1(control);
        respons1();
        write_byte1(date);
        respons1();
        stop1();

}

uchar read_add1(uchar control)//读取 AD 转换值
{
        uchar date;
        start1();
        write_byte1(0x90);   //  地址  1001 A2A1A0 R/W_
        respons1();
        write_byte1(control); //控制字
        respons1();
        start1();
        write_byte1(0x90+1);     //  地址  1001 A2A1A0 R/W_
        respons1();
        date=read_byte1();
        stop1();
        return date;

}

void send_char( unsigned char ch ) //串口发送字符函数
{
        SBUF = ch;
        while(!TI);
        TI = 0;
}
```

```
void main()
{
    P1=0xf0;
    P2=0xff;
    BEEP=1;
    init();
    init1();                          //初始化 PCF8591 接口
    LCD_Initialise();                 //初始化液晶
    LCD_Dispay();                     //显示

    TMOD = 0x20;  //定时器 1 工作在 8 位自动重装模式，用于产生波特率
    TH1 = 0xFD;     //波特率 9600
    TL1 = 0xFD;
    SCON = 0x50;    //设定串行口工作方式
    PCON = 0x00;
    TR1 = 1;        //启动定时器 1
    IE = 0x0;     //禁止中断
    ES = 1;
    EA = 1;

    while(1)
    {
        //PCF8591  控制字
        //0x40 0100 0000 1 表示模拟输出允许，末两位 00 表示通道 0，1，2，3
        temp1=read_add(0x40);          //读取温度值
        temp2=read_add(0x41);
        temp3=read_add(0x42);
        temp4=read_add(0x43);

        temp5=read_add1(0x40);         //读取温度值
        temp6=read_add1(0x41);
        temp7=read_add1(0x42);
        temp8=read_add1(0x43);

        LCD_Dispay();                  //LCD 实时显示所有数据
        write_add(0x40,temp1);      //输出电压

        //这一段容易引发程序跑飞
        /*if(temp1<50||temp1>150)BEEP=0;//温度超出时报警
        else BEEP=1;
        if(temp2<50||temp2>150)BEEP=0;//温度超出时报警
        else BEEP=1;
        if(temp3<50||temp3>150)BEEP=0;//温度超出时报警
        else BEEP=1;
```

```
            if(temp4<50||temp4>150)BEEP=0;//温度超出时报警
            else BEEP=1;
            if(temp5<50||temp5>150)BEEP=0;//温度超出时报警
            else BEEP=1;
            if(temp6<50||temp6>150)BEEP=0;//温度超出时报警
            else BEEP=1;
            if(temp7<50||temp7>150)BEEP=0;//温度超出时报警
            else BEEP=1;
            if(temp8<50||temp8>150)BEEP=0;//温度超出时报警
            else BEEP=1;    */

    }
}
```

附录 3：上位机工程与程序设计

```
/*********************Form1.cs************************************/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Windows.Forms.DataVisualization.Charting;

namespace TempUIv2
{
    public partial class Form1 : Form
    {
        string temp1 = null; //新建变量，暂存温度值
        string temp2 = null;
        string temp3 = null;
        string temp4 = null;
        string temp5 = null;
        string temp6 = null;
        string temp7 = null;
        string temp8 = null;

        public Form1()
        {
            System.Windows.Forms.Control.CheckForIllegalCrossThreadCalls = false;
            // 开启跨线程访问控件
            InitializeComponent();

        }

        string[] dataX = new string[] { "T1", "T2", "T3", "T4", "T5", "T6", "T7",
"T8" };
        double[] dataY = new double[] { 80, 80, 80, 80, 80, 80, 80, 80 }; //初始化数据
        public void chart1Init()    //动态更新柱状图
        {
           chart1.Series.Clear();  //清楚所有柱子
           Series mySeries = new Series("温度"); //重新创建
```

```
            for( int i = 0; i <= 7; i++)
            {
                textBox1.AppendText(dataX[i]+" ");   //数据区输出查看
                textBox1.AppendText(dataY[i].ToString() + "\r\n"); //数据区输出查看
                mySeries.Points.AddXY(dataX[i], dataY[i].ToString()); //重新添加柱状图
            }

             chart1.Series.Add(mySeries);  //将系列加入chart
             chart1.Series.FindByName("温度").IsVisibleInLegend = false; //不显示图例

        }

        private void label1_Click_1(object sender, EventArgs e)
        {

        }

        int timeX = 1;      //时间点
        Series mySeries1 = new Series("S1");    //新建一个曲线
        Series mySeries2 = new Series("S2");
        Series mySeries3 = new Series("S3");
        Series mySeries4 = new Series("S4");
        Series mySeries5 = new Series("S5");
        Series mySeries6 = new Series("S6");
        Series mySeries7 = new Series("S7");
        Series mySeries8 = new Series("S8");
        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.AppendText("默认使用COM2，波特率9600\r\n");  //数据区输出查看

            chart1.Titles.Add("温度数据实时柱状图");
            //chart1.Titles[0].ForeColor = Color.White;
            chart1.Titles[0].Font = new Font("微软雅黑", 10f, FontStyle.Regular);
            chart1Init();
            //X坐标轴标题
            chart1.ChartAreas[0].AxisX.Title = "传感器";
            chart1.ChartAreas[0].AxisX.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);

            //Y坐标轴标题
            chart1.ChartAreas[0].AxisY.Title = "温度值：摄氏度";
            chart1.ChartAreas[0].AxisY.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);
```

```
            chart2.Titles.Add("温度数据历史数据与变化");
            //chart1.Titles[0].ForeColor = Color.White;
            chart2.Titles[0].Font = new Font("微软雅黑", 10f, FontStyle.Regular);

            //X坐标轴标题
            chart2.ChartAreas[0].AxisX.Title = "时间轴";
            chart2.ChartAreas[0].AxisX.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);

            //Y坐标轴标题
            chart2.ChartAreas[0].AxisY.Title = "温度值：摄氏度";
            chart2.ChartAreas[0].AxisY.TitleFont = new Font("微软雅黑", 10f,
FontStyle.Regular);

            chart2.ChartAreas[0].AxisY.Maximum = 100;

            chart2.Series.Clear();
            mySeries1.ChartType = SeriesChartType.FastLine; //绘制折线图
            mySeries1.BorderWidth = 2;                      //线条粗细
            mySeries1.Color = System.Drawing.Color.Red;     //设置线条颜色

            mySeries2.ChartType = SeriesChartType.FastLine;
            mySeries2.BorderWidth = 2;
            mySeries2.Color = System.Drawing.Color.Blue;

            mySeries3.ChartType = SeriesChartType.FastLine;
            mySeries3.BorderWidth = 2;

            mySeries4.ChartType = SeriesChartType.FastLine;
            mySeries4.BorderWidth = 2;

            mySeries5.ChartType = SeriesChartType.FastLine;
            mySeries5.BorderWidth = 2;

            mySeries6.ChartType = SeriesChartType.FastLine;
            mySeries6.BorderWidth = 2;

            mySeries7.ChartType = SeriesChartType.FastLine;
            mySeries7.BorderWidth = 2;

            mySeries8.ChartType = SeriesChartType.FastLine;
            mySeries8.BorderWidth = 2;
```

```
            chart2.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = true;  //滚动条在
曲线图内
            chart2.ChartAreas[0].AxisX.ScrollBar.Enabled = true;  //显示滚动条
            chart2.ChartAreas[0].AxisX.ScaleView.Size = 20;        //

            chart2.Series.Add( mySeries1 );
            chart2.Series.Add( mySeries2 );
            chart2.Series.Add( mySeries3 );
            chart2.Series.Add( mySeries4 );
            chart2.Series.Add( mySeries5 );
            chart2.Series.Add( mySeries6 );
            chart2.Series.Add( mySeries7 );
            chart2.Series.Add( mySeries8 );
        }

        string str_data = null;
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            if (serialPort1.BytesToRead >= 43)  //累计43个字符再进行处理
            {
                Byte[] Data = new Byte[serialPort1.BytesToRead];  //创建字符串，大小为
缓冲区已收到的数据大小
                serialPort1.Read(Data, 0, Data.Length);  //提取缓冲区中所有数据
                string str = System.Text.Encoding.ASCII.GetString(Data); //转化为ASCII
码
                str_data = str_data + str;
                serialPort1.DiscardInBuffer(); //清空缓冲区

                int i;
                for (i = 0; i <= str_data.Length - 1 - 3 && i + 40 < str_data.Length;
i++)
                {
                    if (str_data[i] == '0' && str_data[i + 1] == '0' && str_data[i + 2]
== '0' && str_data[i + 3] == 'C') //接收到起始符
                        {
                            //开始接受温度值
                            temp3 = " ";
                            temp3 += str_data[i + 4];
                            temp3 += str_data[i + 5];
                            temp3 += '.';
                            temp3 += str_data[i + 6];
                            temp3 += " Celsius degree";
                            textBox4.Text = temp3;  //更新温度值
```

```
textBox1.AppendText(temp3 + "\r\n");

temp4 = " ";
temp4 += str_data[i + 8];
temp4 += str_data[i + 9];
temp4 += '.';
temp4 += str_data[i + 10];
temp4 += " Celsius degree";
textBox5.Text = temp4;
textBox1.AppendText(temp4 + "\r\n");

temp1 = " ";
temp1 += str_data[i + 12];
temp1 += str_data[i + 13];
temp1 += '.';
temp1 += str_data[i + 14];
temp1 += " Celsius degree";
textBox2.Text = temp1;
textBox1.AppendText(temp1 + "\r\n");

temp2 = " ";
temp2 += str_data[i + 16];
temp2 += str_data[i + 17];
temp2 += '.';
temp2 += str_data[i + 18];
temp2 += " Celsius degree";
textBox3.Text = temp2;
textBox1.AppendText(temp2 + "\r\n");

temp7 = " ";
temp7 += str_data[i + 20];
temp7 += str_data[i + 21];
temp7 += '.';
temp7 += str_data[i + 22];
temp7 += " Celsius degree";
textBox8.Text = temp7;
textBox1.AppendText(temp7 +"\r\n");

temp8 = " ";
temp8 += str_data[i + 24];
temp8 += str_data[i + 25];
temp8 += '.';
temp8 += str_data[i + 26];
temp8 += " Celsius degree";
```

```
                    textBox9.Text = temp8;
                    textBox1.AppendText(temp8 + "\r\n");

                    temp5 = " ";
                    temp5 += str_data[i + 28];
                    temp5 += str_data[i + 29];
                    temp5 += '.';
                    temp5 += str_data[i + 30];
                    temp5 += " Celsius degree";
                    textBox6.Text = temp5;
                    textBox1.AppendText(temp5 + "\r\n");

                    temp6 = " ";
                    temp6 += str_data[i + 32];
                    temp6 += str_data[i + 33];
                    temp6 += '.';
                    temp6 += str_data[i + 34];
                    temp6 += " Celsius degree";
                    textBox7.Text = temp6;
                    textBox1.AppendText(temp6 + "\r\n");

                    str_data = "";

                    updateChart1Num();
                    chart1Init();
                    mySeries1.Points.AddXY(timeX, dataY[0]);
                    mySeries2.Points.AddXY(timeX, dataY[1]);
                    mySeries3.Points.AddXY(timeX, dataY[2]);
                    mySeries4.Points.AddXY(timeX, dataY[3]);
                    mySeries5.Points.AddXY(timeX, dataY[4]);
                    mySeries6.Points.AddXY(timeX, dataY[5]);
                    mySeries7.Points.AddXY(timeX, dataY[6]);
                    mySeries8.Points.AddXY(timeX, dataY[7]);
                    timeX++;

                    break;
                }
            }
        }

}

public void updateChart1Num()  //将接收到的字符转为double类型方便折线图处理
{
```

```
            try
            {
                //数据转换
                double data1 = double.Parse(temp1.Substring(1, 2)) +
double.Parse(temp1.Substring(4, 1)) / 10;
                double data2 = double.Parse(temp2.Substring(1, 2)) +
double.Parse(temp2.Substring(4, 1)) / 10;
                double data3 = double.Parse(temp3.Substring(1, 2)) +
double.Parse(temp3.Substring(4, 1)) / 10;
                double data4 = double.Parse(temp4.Substring(1, 2)) +
double.Parse(temp4.Substring(4, 1)) / 10;
                double data5 = double.Parse(temp5.Substring(1, 2)) +
double.Parse(temp5.Substring(4, 1)) / 10;
                double data6 = double.Parse(temp6.Substring(1, 2)) +
double.Parse(temp6.Substring(4, 1)) / 10;
                double data7 = double.Parse(temp7.Substring(1, 2)) +
double.Parse(temp7.Substring(4, 1)) / 10;
                double data8 = double.Parse(temp8.Substring(1, 2)) +
double.Parse(temp8.Substring(4, 1)) / 10;

                //存储数据到数组
                dataY[0] = data1;
                dataY[1] = data2;
                dataY[2] = data3;
                dataY[3] = data4;
                dataY[4] = data5;
                dataY[5] = data6;
                dataY[6] = data7;
                dataY[7] = data8;

                //数据区输出查看
                textBox1.AppendText(data1.ToString() + "\r\n");
                textBox1.AppendText(data2.ToString() + "\r\n");
                textBox1.AppendText(data3.ToString() + "\r\n");
                textBox1.AppendText(data4.ToString() + "\r\n");
                textBox1.AppendText(data5.ToString() + "\r\n");
                textBox1.AppendText(data6.ToString() + "\r\n");
                textBox1.AppendText(data7.ToString() + "\r\n");
                textBox1.AppendText(data8.ToString() + "\r\n");

            }catch( Exception ee)
            {
                //MessageBox.Show(ee.ToString());
            }
```

```
}

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if( !serialPort1.IsOpen )
        {
            serialPort1.Open();
            textBox1.AppendText("成功打开串口\r\n");
        }
        else
        {
            serialPort1.Close();
        }

    }
    catch( Exception ee )
    {
        MessageBox.Show(ee.ToString());
    }
}

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        if ( serialPort1.IsOpen )
        {
            serialPort1.Close();
            textBox1.AppendText("成功关闭串口\r\n");
        }
    }
    catch (Exception ee)
    {
        MessageBox.Show(ee.ToString());
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}
```

```
private void button3_Click(object sender, EventArgs e)
{
    textBox1.AppendText(str_data);

}

private void button4_Click(object sender, EventArgs e)
{
    textBox1.Clear();
}

private void label2_Click(object sender, EventArgs e)
{

}

private void label4_Click(object sender, EventArgs e)
{

}

private void groupBox1_Enter(object sender, EventArgs e)
{

}

private void label3_Click(object sender, EventArgs e)
{

}

private void toolStripLabel2_Click(object sender, EventArgs e)  //温度曲线
{

}

private void toolStripLabel1_Click(object sender, EventArgs e)  //温度数据
{

}

private void chart1_Click(object sender, EventArgs e)
{
```

```
        }

        private void chart2_Click(object sender, EventArgs e)
        {

        }
    }
}
```

/************Program.cs***/

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TempUIv2
{
    static class Program
    {
        /// <summary>
        /// 应用程序的主入口点。
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```