浙江大学实验报告

专业：生物医学工程
姓名：
学号：
日期：2025.5.20
地点：东 1B-416

课程名称：微机原理及其应用　指导老师：陈星　成绩：
实验名称：AD 采样程序设计调试　实验类型：微机实验　同组学生姓名：

一、实验目的和要求

1. 实验目的

本实验旨在深入理解 A/D（模数转换）芯片的工作原理及其控制方法。通 过将温度传感器（NTC）采集到的温度值显示在单片机的数码管上，学生将能够掌握 A/D 采样的基本流程。同时，实验还将增强学生对数据上传控制的理解，培养他们的编程能力和调试技能。

2. 实验要求

1). 以实验六的程序为基础，将 A/D 芯片采集到的温度值显示在单片机的数码管上

2). 利用 XPT2046 的 A/D 进行电压采样，并将采样数据转换成对应的电压值和温度值后显示在数码管上。

3). 设置两个按键，一个为开始键，用于启动数据上传；另一个为结束键，用于停止数据上传。

4). 串口的波特率设置为 4800。

5). 考虑到采样数据可能含有噪声，对采样的数据进行滤波以提高数据准确性。

6). 数据缓存区不得设置在外部数据存储器。

7). 对编写的程序先在 proteus 上仿真验证，由于 proteus 里没有 XPT2046 的芯片，只需仿真数码管显示和数据上传控制功能（温度和电压可以给固定值）。后续用 keil 软件编译程序生成 HEX 文件，下载到开发板进行实机验证调试。

二、实验原理

1. SPI 通讯方式

SPI 系统总线一般有 4 条线：串行时钟线（SCLK），主机输入/从机输出 线（MISO），主机输出/从机输入（MOSI）和低电平有效的片选线$\overline{CS}$，亦有数据手册将片选线叫为$\overline{SS}$

单片机使用 SPI 接口一般有两种方式，一种是单片机自身带有 SPI 的硬件接口，那使用者只需要配置 SPI 的工作方式，比如时钟极性和时钟相位以及时钟的频率等，便可通过 SPI 发送数据，这种硬件 SPI 的方式通信速度很快；另一种是单片机自身不带 SPI 硬件接口，可采用软件控制 I/O 来模拟 SPI 的操作，包括串行时钟，数据输入输出，这种软件模拟 SPI 的方式速度较慢，但是驱动移植会很方便。本次实验用到的 89C52 没有硬件 SPI 接口，所以采用第二种软件模拟 SPI 的方式。

2. A/D 转换器

能够将连续的模拟量转换为离散的数字量的器件称为模/数转换器，简称 A/D 转换器或 ADC。计算机能够识别并处理的都是数字量，然而在一些实际的测控系统中所发生的各种物理参数常常是一些模拟量（如压

力，温度等）。对于这些物理参数首先用传感器将其转换为电信号，在经过 ADC 转成数字信号，传送给计算机处理。

3. 中断工作方式原理

中断是指计算机在执行程序的过程中，当出现某些事件（如外部设备的数据到达）时，计算机停止当前正在执行的程序，转而去执行处理该事件的程序（中断服务程序），处理完毕后再返回原来被中断的程序处继续执行。

在本实验的按键检测中，采用中断方式进行按键检测，提高 CPU 的工作效率，减少无谓的轮询操作。

4. 按键消抖工作原理

为了克服按键触点机械抖动所致的检测误判，必须采取消抖措施，本次实验不在使用实验二和实验三的方法进行消抖，而是采用按键触发中断后延时一段时间（一般为 10ms）再去检测按键的状态的方法，如果还是按下的状态则记为有效按压，并进行后续处理，如果不是按下的状态，则认为是误触发中断，系统不做处理。

5. NTC 热敏电阻

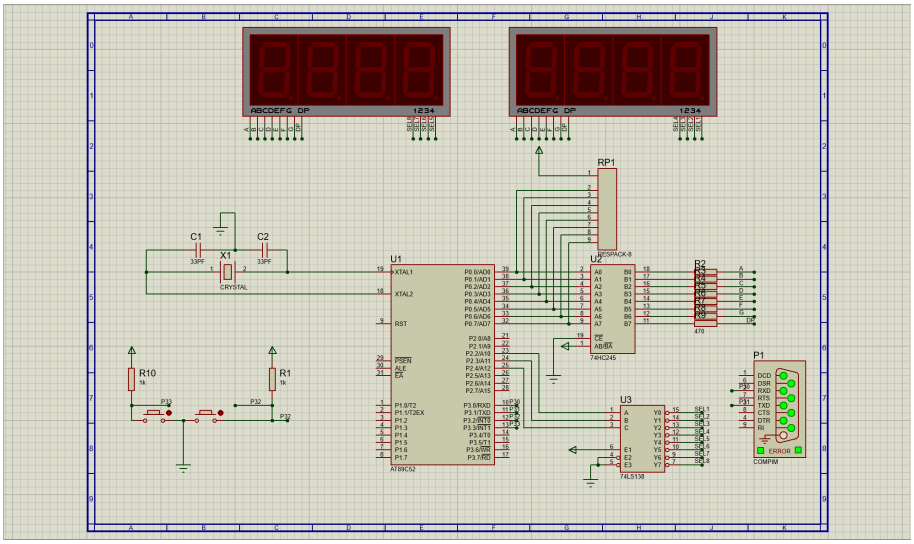
NTC 热敏电阻是指具有负温度系数的热敏电阻。是使用单一高纯度材料、具有接近理论密度结构的高性能陶瓷。因此，在实现小型化的同时，还具有电阻值、温度特性波动小、对各种温度变化响应快的特点，可进行高灵敏度、高精度的检测。

三、实验设备及器材

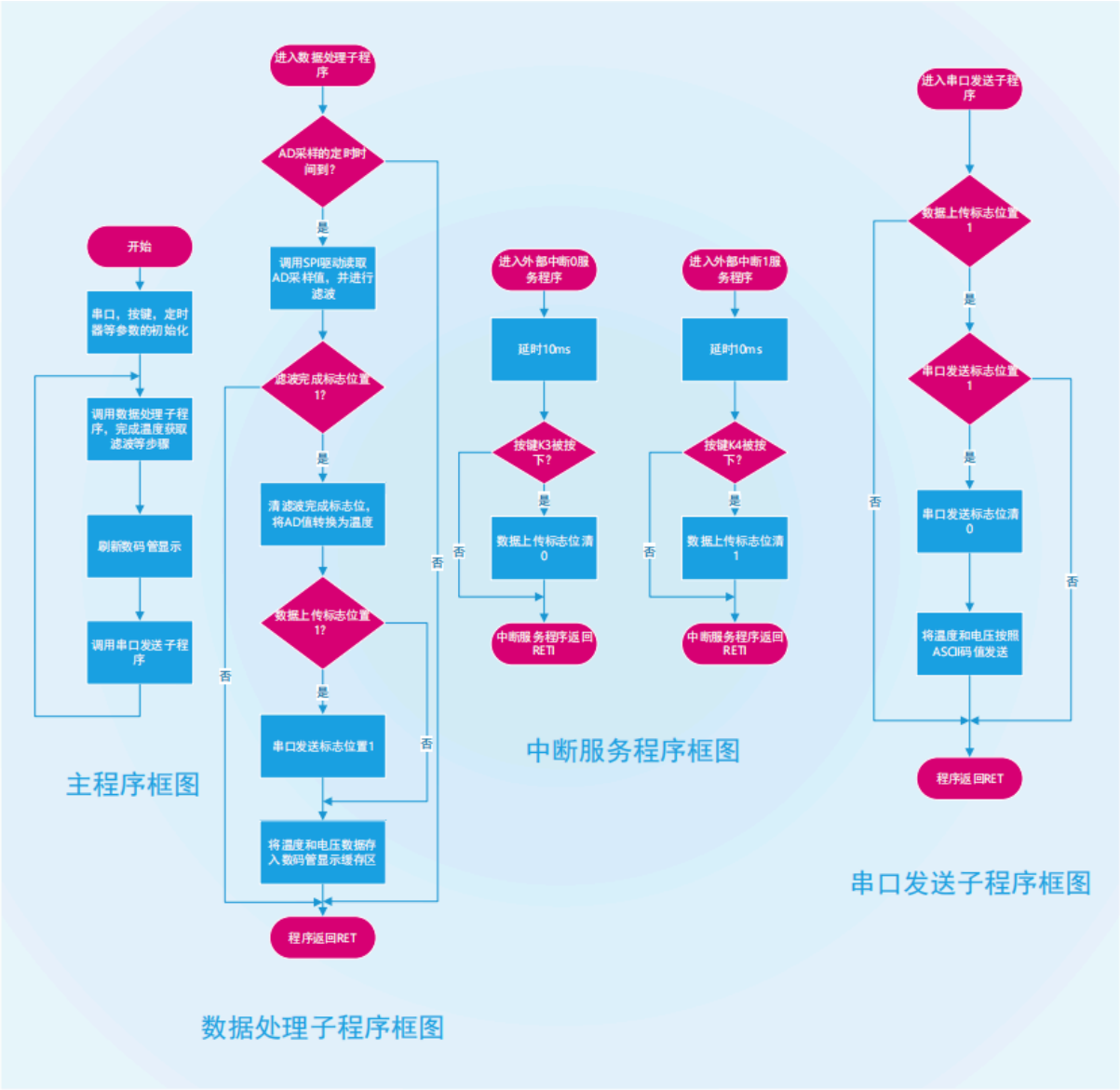
计算机，开发板：HC6800 - ES V2.0 开发板，USB 转串口线（开发板自带）

四、实验步骤和方法

1. 硬件电路设计



2. 软件流程图



3. 程序代码

main.c 文件

```
#include <reg52.h>
#include <intrins.h>
#include <math.h>
#include <XPT2046.h>

// 定义引脚
sbit K3 = P3^2;  // 开始键
sbit K4 = P3^3;  // 结束键
sbit LSA = P2^2;
sbit LSB = P2^3;
sbit LSC = P2^4;
static bit flag, send_flag;


typedef unsigned int u16;    //对数据类型进行声明定义
typedef unsigned char u8;

// 数码管段码表
u16 count = 0;
u16 display_temp,display_vin;
```

```
uchar code smgduan[16] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x77,
0x7c, 0x39, 0x5e, 0x79, 0x71};
uchar disp[8];  // 数码管显示缓冲区


/*******************************************************************************
* 函 数 名         : delay
* 函数功能        : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
  while(i--);
}
void keypros() {
    if(K3==0)  //检测按键K3是否按下
    {
        delay(1000);  //消除抖动，一般大约10ms
        flag=1;  //发送标志为1
        while(!K3);
    }
    if(K4==0)  //检测按键K4是否按下
    {
        delay(1000);  //消除抖动，一般大约10ms
        flag=0;  //发送标志为0
        while(!K4);
    }
}
/*************************************************************************
***********
* 函 数 名 : Timer0Interrupt(void) interrupt 1
* 函数功能 : 定时器0 的中断函数
* 输 入 : 无
* 输 出 : 无
*************************************************************************
**********/
void Timer0Interrupt(void) interrupt 1 {
    TH0 = 0x3C;         // 重载初值
    TL0 = 0xB0;
    if(++count >= 10) { // 50ms×10=0.5秒
        count = 0;
        send_flag = 1;  // 触发发送
    }
}
/*************************************************************************
***********
* 函 数 名 : Board_Get_NTC_R(unsigned int ad)
* 函数功能 : NTC 电阻值转换函数
* 输 入 : AD 值
* 输 出 : 电阻值
**********/
float Board_Get_NTC_R(unsigned int ad) {
    float R = 0;
    float vin;
    vin = ad * 5.0 / 4096;
    R = (5 - vin) / (vin / 10000);
    return R;
}
/*************************************************************************
***********
* 函 数 名 : BoardGetNTCTemp(unsigned int ad)
* 函数功能 : NTC 温度转换函数
* 输 入 : AD 值
* 输 出 : 温度值
*************************************************************************
**********/
float BoardGetNTCTemp(unsigned int ad) {
    float Rt = 0;
    float Rp = 10000;
    float T2 = 273.15 + 25;
    float Bx = 3950;
```

```
    float Ka = 273.15;
    float temp = 0;
    Rt = Board_Get_NTC_R(ad);
    temp = Rt / Rp;
    temp = log(temp); // ln(Rt/Rp)
    temp /= Bx; // ln(Rt/Rp)/B
    temp += (1 / T2);
    temp = 1 / temp;
    temp -= Ka;
    return temp;
}

/*******************************************************************************
* 函数名          :datapros()
* 函数功能      :数据处理函数
* 输入            : 无
* 输出            : 无
*******************************************************************************/
void datapros()
{
  u16 ad;
  float temp,vin,R;
  static u8 i;
  if(i==50)
  {
    i=0;
    ad= Read_AD_Data(0xD4);
    temp = BoardGetNTCTemp(ad);//   AIN1 热敏电阻
    display_temp =(u16)(temp*100);
    R=  Board_Get_NTC_R(ad);
    vin = 5.0*R/(R+10000);
    display_vin=(u16)(vin*1000);
  }
  i++;
  disp[0]=smgduan[display_temp/1000];//千位
  disp[1]=smgduan[display_temp%1000/100]|0x80;//百位
  disp[2]=smgduan[display_temp%1000%100/10];//个位
  disp[3]=smgduan[display_temp%1000%100%10];
  disp[4]=smgduan[display_vin/ 1000] | 0x80; // 千位显示小数点
  disp[5]=smgduan[display_vin % 1000 / 100]; // 百位
  disp[6]=smgduan[display_vin % 1000 % 100 / 10]; // 个位
  disp[7]=smgduan[display_vin % 1000 % 100 % 10];
}


/*******************************************************************
***********
* 函数名 :DigDisplay()
* 函数功能 :数码管显示函数
*输入 :无
*输出 :无
*******************************************************************
**********/
void DigDisplay() { // 数码管显示函数
    static unsigned char i = 0;
    for(i=0; i<8; i++) {
        switch(i) { // 位选,选择点亮的数码管
            case(0):
                LSA=0; LSB=0; LSC=0; break; // 显示第0 位
            case(1):
                LSA=1; LSB=0; LSC=0; break; // 显示第1 位
            case(2):
                LSA=0; LSB=1; LSC=0; break; // 显示第2 位
            case(3):
                LSA=1; LSB=1; LSC=0; break; // 显示第3 位
            case(4):
                LSA=0; LSB=0; LSC=1; break; // 显示第4 位
            case(5):
                LSA=1; LSB=0; LSC=1; break; // 显示第5 位
            case(6):
```

```
                LSA=0; LSB=1; LSC=1; break; // 显示第6 位
            case(7):
                LSA=1; LSB=1; LSC=1; break; // 显示第7 位
        }
        P0 = disp[7-i]; // 发送数据
        delay(100); // 间隔一段时间扫描
        P0 = 0x00; // 消隐
    }
}

/*******************************************************************************
* 函 数 名         : main
* 函数功能     : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
  SCON=0X50;  //设置为工作方式1
  TMOD=0X21;  //设置计数器工作方式2
  PCON=0X80;  //波特率加倍
  TH1=0XF3;   //计数器初始值设置，注意波特率是4800的
  TL1=0XF3;
  TH0=0x3C;         // 50ms定时 @11.0592MHz
  TL0=0xB0;
  ET0=1;            // 允许定时器0中断
  TR0=1;            // 启动定时器0
  ES=0;        //打开接收中断
  EA=1;        //打开总中断
  IT0=1;  // 外部中断0下降沿触发
  IT1=1;  // 外部中断1下降沿触发
  TR1=1;       //打开计数器
  while(1)
  {
    datapros();  //数据处理函数
    DigDisplay();//数码管显示函数
    keypros();
    if (flag==1 && send_flag==1)
    {
                //将数据送至数据缓冲寄存器
        SBUF= display_temp /1000 + 0x30 ;//将千位送至数据缓冲寄存器
        while(!TI);  //等待发送数据完成
        TI=0;         //清除发送完成标志位
        SBUF= display_temp %1000/100 + 0x30;
        while(!TI);
        TI = 0;
        SBUF= 0x2E;
        while(!TI);
        TI = 0;
        SBUF= display_temp %1000%100/10 + 0x30;
        while(!TI);
        TI = 0;
        SBUF= display_temp %1000%100%10 + 0x30;
        while(!TI);
        TI = 0;
        SBUF=0x20; //发送空格
        while(!TI);
        TI = 0;
        SBUF= display_vin /1000 + 0x30 ;
        while(!TI);
        TI=0;
        SBUF= 0x2E;
        while(!TI);
        TI = 0;
        SBUF= display_vin %1000/100 + 0x30;
        while(!TI);
        TI = 0;
        SBUF= display_vin %1000%100/10 + 0x30;
        while(!TI);
        TI = 0;
```

```
231          SBUF= display_vin %1000%100%10 + 0x30;
232          while(!TI);
233          TI = 0;
234          send_flag=0;
235       }
236     }
237  }
```

XPT2046.c

```
1  #include"XPT2046.h"
2  /*****************************************************************************
3  *函数名：TSPI_Start
4  *输  入：无
5  *输  出：无
6  *功  能：初始化触摸SPI
7  *****************************************************************************/
8
9  void SPI_Start(void)
10 {
11   CLK = 0;
12   CS  = 1;
13   DIN = 1;
14   CLK = 1;
15   CS  = 0;
16 }
17 /*****************************************************************************
18 *函数名：SPI_Write
19 *输  入：dat：写入数据
20 *输  出：无
21 *功  能：使用SPI写入数据
22 *****************************************************************************/
23
24 void SPI_Write(uchar dat)
25 {
26   uchar i;
27   CLK = 0;
28   for(i=0; i<8; i++)
29   {
30     DIN = dat >> 7;    //放置最高位
31     dat <<= 1;
32     CLK = 0;       //上升沿放置数据
33
34     CLK = 1;
35
36   }
37 }
38 /*****************************************************************************
39 *函数名：SPI_Read
40 *输  入：无
41 *输  出：dat：读取 到的数据
42 *功  能：使用SPI读取数据
43 *****************************************************************************/
44
45 uint SPI_Read(void)
46 {
47   uint i, dat=0;
48   CLK = 0;
49   for(i=0; i<12; i++)    //接收12位数据
50   {
51     dat <<= 1;
52
53     CLK = 1;
54     CLK = 0;
55
56     dat |= DOUT;
57
58   }
59   return dat;
60 }
61
```

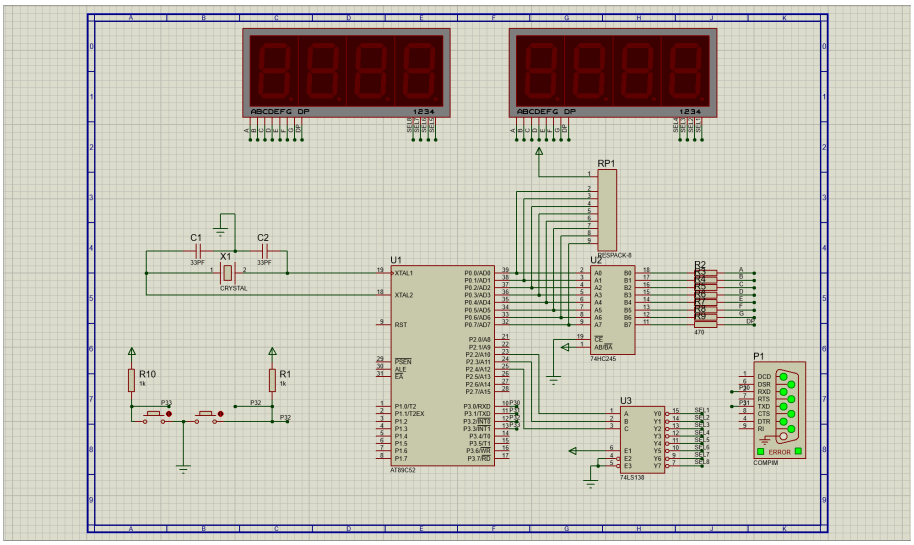
```c
/*********************************************************************************
*函数名：Read_AD_Data
*输　入：cmd：读取的X或者Y
*输　出：endValue：最终信号处理后返回的值
*功　能：读取触摸数据
*********************************************************************************/
uint Read_AD_Data(uchar cmd)
{
  uchar i;
  uint AD_Value;
  CLK = 0;
  CS  = 0;
  SPI_Write(cmd);
  for(i=6; i>0; i--);    //延时等待转换结果
  CLK = 1;     //发送一个时钟周期，清除BUSY
  _nop_();
  _nop_();
  CLK = 0;
  _nop_();
  _nop_();
  AD_Value=SPI_Read();
  CS = 1;
  return AD_Value;
}
```

XPT2046.h 头文件

```c
#ifndef   __XPT2046_H_
#define   __XPT2046_H_

//---包含头文件---//
#include<reg52.h>
#include<intrins.h>

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

#ifndef ulong
#define ulong  unsigned long
#endif

//---定义使用的IO口---//
sbit DOUT = P3^7;   //输出
sbit CLK  = P3^6;   //时钟
sbit DIN  = P3^4;   //输入
sbit CS   = P3^5;   //片选

uint Read_AD_Data(uchar cmd);
uint SPI_Read(void);
void SPI_Write(uchar dat);

#endif
```

五、实验结果与分析

1. 硬件电路设计

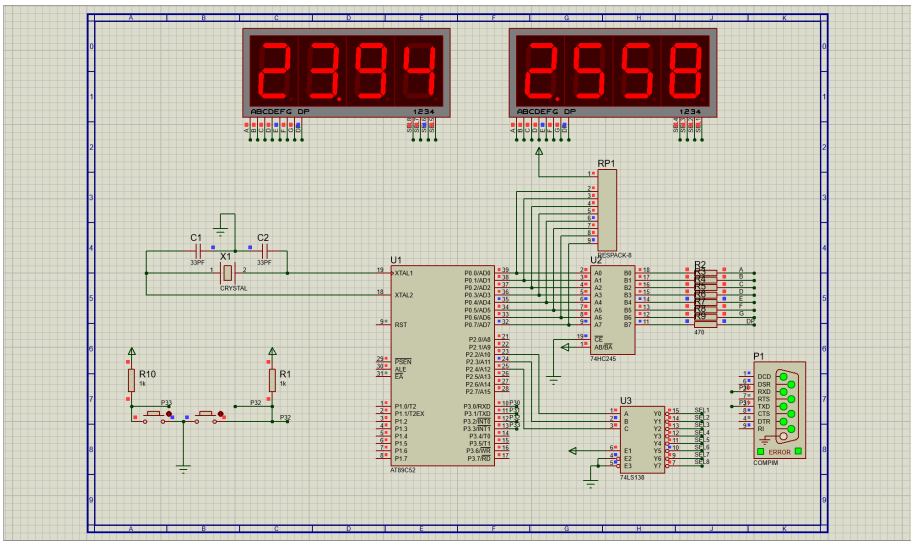


2. 软件编写经历

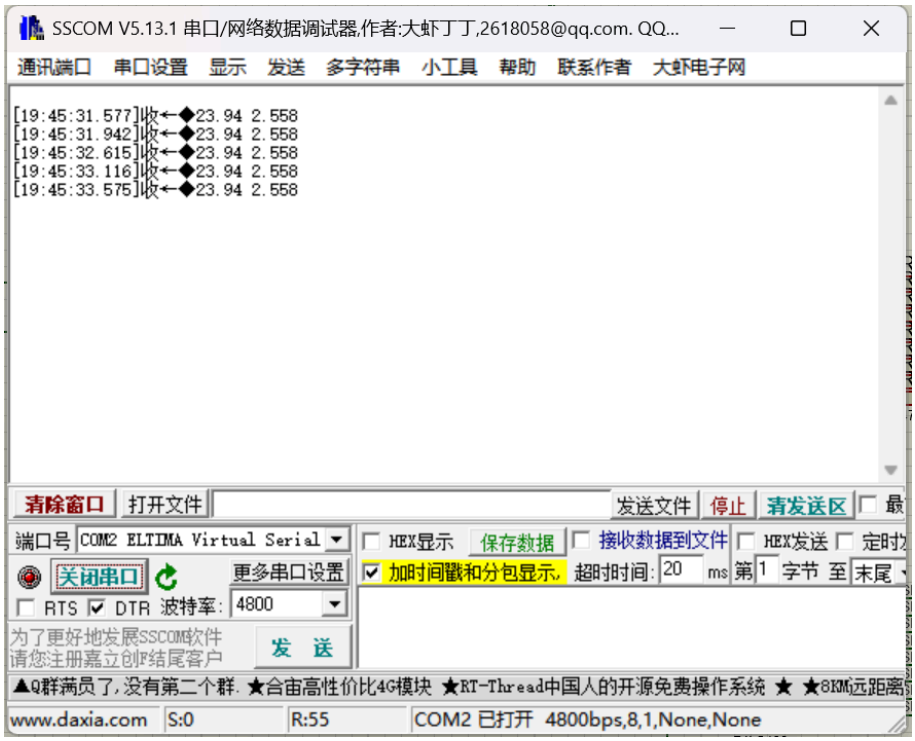
本次编写的代码主要是结合了例程给出的代码和单片机光盘程序中给出的相关示例代码综合完成的。其中对于按键的处理响应没有通过例程的中断来进行，而是直接采用了标志位的方式。一开始有关串口的发送我也没有采用定时器中断的方式来进行，导致串口端每秒接受到的数据量非常大，我认为这是不易处理的，所以最后采用定时器中断的方式，每隔 0.5 秒置位 send_flag 变量，达到每隔 0.5 秒向串口传输数据的效果。代码也有许多优化的空间，滤波的功能没有实现。

3. 软件仿真运行结果

给定 ad 值为 2000，仿真结果如下：

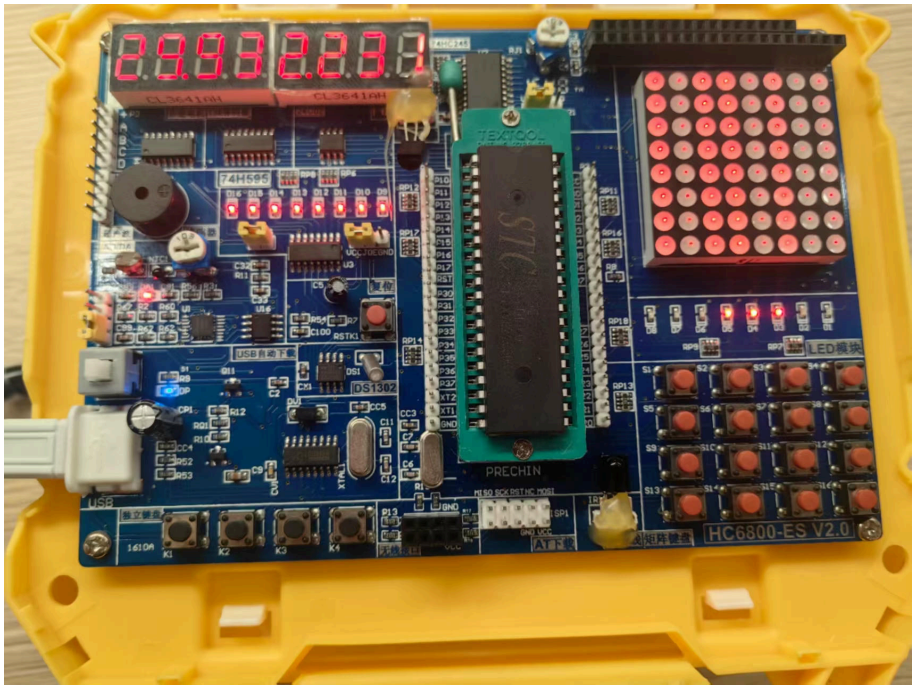


串口端接收结果：

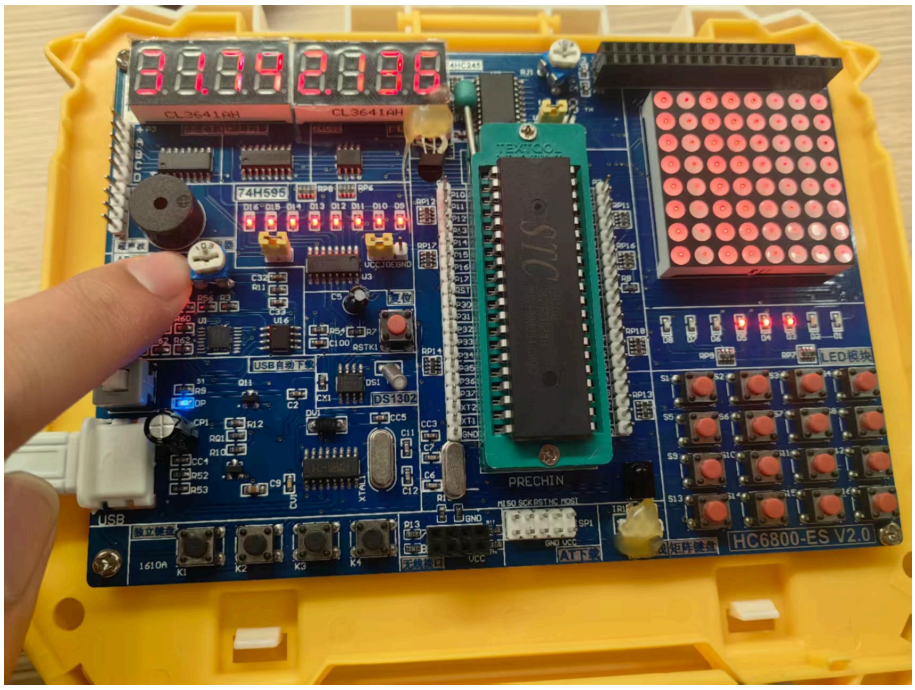


4. 开发板实际运行结果

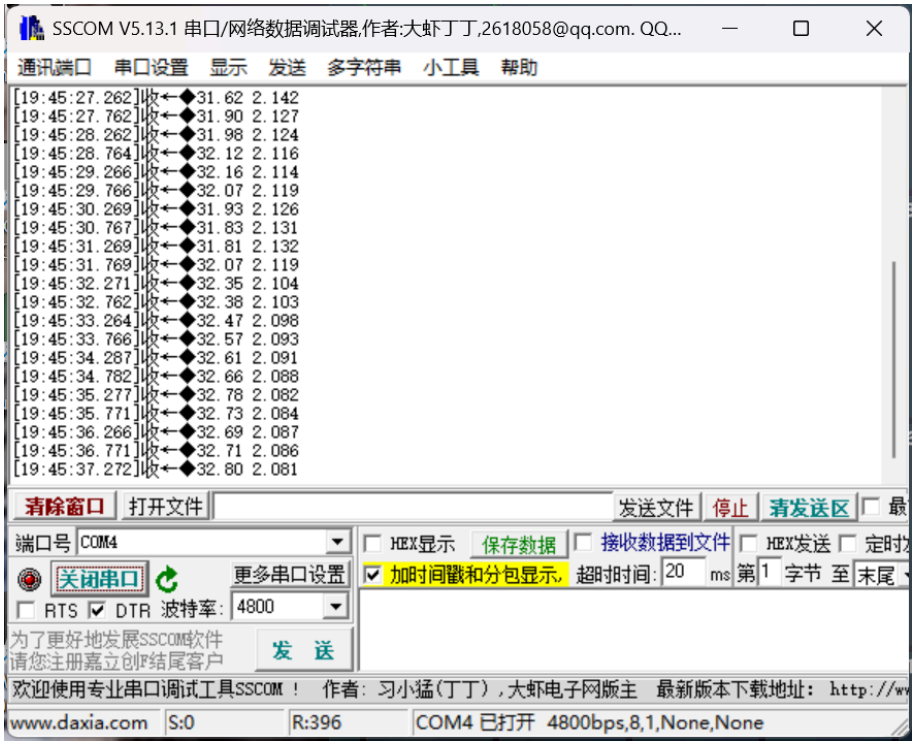
开始时显示的温度和电压值：



用手触摸后，温度上升:



串口端接收结果：



六、扩展思考题

改为中断式发送的的相关代码如下

```
#define UART_BUF_SIZE 32  // 发送缓冲区大小

u8 uart_tx_buf[UART_BUF_SIZE]; // 发送缓冲区
bit uart_sending = 0;              // 发送状态标志
u8 uart_tx_index = 0;          // 当前发送位置
```

```
u8 uart_tx_len = 0;             // 待发送数据长度

void UART_SendString(uchar *str, uchar len) {
    if(uart_sending) return;  // 正在发送则退出

    // 复制数据到发送缓冲区
    for(u8 i=0; i<len; i++) {
        uart_tx_buf[i] = str[i];
    }
    uart_tx_len = len;
    uart_tx_index = 0;
    uart_sending = 1;         // 标记发送状态

    SBUF = uart_tx_buf[0];    // 启动首次发送
}
void UART_ISR() interrupt 4 {
    if(TI) {                  // 发送中断
        TI = 0;               // 必须清除标志
        uart_tx_index++;

        if(uart_tx_index < uart_tx_len) {
            SBUF = uart_tx_buf[uart_tx_index]; // 发送下一字节
        } else {
            uart_sending = 0; // 发送完成
        }
    }

    if(RI) {                  // 接收中断（本实验未用）
        RI = 0;               // 清除标志
    }
}
```

七、讨论与心得

本次实验在上一次实验的基础上，继续巩固了 51 单片机的数码管的动态显示。同时回顾了串口通讯、中断响应的相关知识，完成了本次的 AD 采样程序，实现了对热敏电阻温度值和电压值的显示，对于单片机结合 XPT2046 芯片实现 AD 转换功能的实现有了更加深刻的理解。