华中科技大学电子与信息工程系
2011 年 TI 杯电子设计大赛项目总结报告

项目标题：温度测量与多路无线巡检装置

指导老师：汪小燕

成员名单：李逸飞　电信 0803　u200812905

李紫惠　电信 0803　u200812906

刘洋　电信 0803　u200812935

方宸　电信 0806　u200812855

组员签名：_____

_____

_____

_____

日期：2011 年 7 月 8 日

目录

1 实验任务

设计并制作一个温度测量与多路无线巡检装置，可测量传感器感知的接触表面温度并进行诸如巡检等功能扩展。

1.1 基本要求

A. 温度测量范围为 0℃～45℃，具有温度数码显示功能，分辨率为 0.1℃。
B. 具有输入控制功能，可由外界输入温度超限报警门限。
C. 具有温度超限报警功能，当温度超出指定温度，必须给出声或光提示信号。
D. 进行多路温度巡检，显示当前巡检传感器的温度测量值；可根据输入选择工作在巡检或指定传感器测量模式。
E. 能够实现多路温度巡检数据的无线传送以及接收无线控制指令进行温度测量并回送测量数据。

1.2 发挥部分

A. 能记录并实时显示温度调节过程的曲线，显示的误差绝对值小于 2℃。
B. 进行温度的自动调节控制，可调节范围为 5℃～35℃，最小设定分度为 1℃。温度控制范围可由外界输入，当温度达到某一设定值并稳定后，装置接触表面的温度波动范围控制在±5℃以内。要求温度调控达到稳定状态时，必须给出声或光提示信号。
C. 其他功能

2 设计目标

以 TI 公司的 msp430 为本地端和远端主控芯片，搭配美国 DALLAS 公司生产的 DS18B20 来测量远端四路温度，依靠挪威 Nordic 公司生产的 NRF905（活着 NRF24L01）来实现无线传输数据的功能。在远端，MSP430 收到控制数据将相应的路数的温度数值通过 NRF905 传递给本地端，本地端再将数据传递给 MSP430，在 LCD12864 屏幕上显示当前该路的温度数值并能显示温度波形。本地端用 8 个矩阵式键盘作为控制按键，能够设定阈值温度和选择需要测量的路数在液晶屏上显示出来。当温度超过阈值温度时，蜂鸣器报警。同时，温控系统将温度回复到阈值温度内，完成后会发出光信号。

3 团队成员

我们团队由 4 人组成，分别是：
李逸飞　电信 0803 班
李紫惠　电信 0803 班
刘　洋　电信 0803 班
方　宸　电信 0806 班
小组分工如下：
李逸飞：　负责无线收发和报警模块，原器件的采购，代码融合和文档的整理工作。
李紫惠：　主要负责 LCD 显示模块，PCB 布板和后期展示工作。
刘　洋:　主要负责键盘模块的设计，代码融合以及整机联调工作。
方　宸：　主要负责温度采集模块，protel 设计和硬件制作工作。

4 系统设计方案

4.1 原始方案

4.1.1 本地端原始方案

（1）主要器件：

MSP430F449------1
蜂鸣器----------1
4*4 矩阵键盘-----1
NRF24L01----------1
LCD12864----------1
稳压芯片----------1
电阻电容若干----1

（2）系统框图

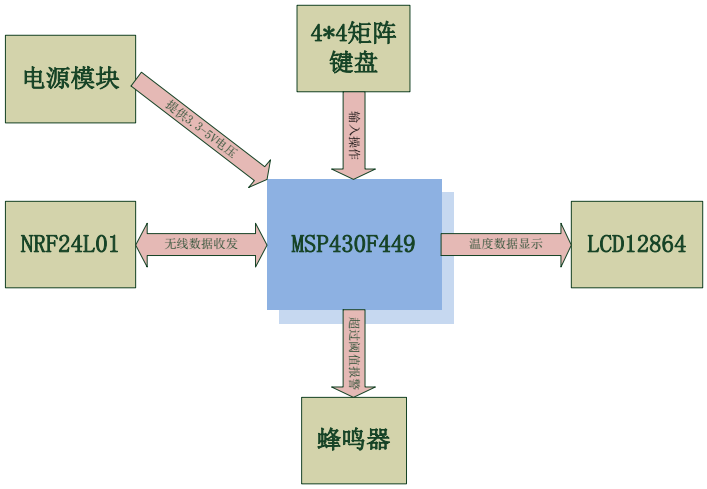


图 4.1

（3）实现原理

A. MSP430F449 作为总控制器。用来接收判断按键，接受来自 NRF24L01 的数据并将数据通过 NRF24L01 发送给远端，同时将温度数据传递给 LCD12864 让其显示。超过阈值电压后，通过 MSP430F449 的 GPIO 口控制蜂鸣器的报警。

B. 主程序中以判断键盘按下哪一个案件展开。当收到按键终端以后执行相应的程序。采用 4*4 矩阵键盘，我们使用其中的 8 个按键。其中 4 个按键用于调整阈值温度，两个用于调整上限阈值温度，两个用于调整下限阈值温度。另外四个按键用于切换路数。

C. LCD12864 能够显示温度阈值，当前路的温度已经温度波形曲线。

E. NRF24L01 采用双向收发模式，同时发送控制命令和接收远端温度数据。

F. MSP430F449 的特性如下

MSP430F449 是 TI 公司生产的一款超低功耗单片机，能够在 1.8 V～3.6 V 电压、1 MHz 的时钟条件下运行，具有 5 种节电模式，耗电流为 0.1 μA～400 μA，能够在数据检测时充分节电，满足电池供电下的节能要求；具有强大的处理能力(RISC 结构、8MHz 晶体、驱动指令周期 125 ns) 和丰富的片内外设(60 KB+256B Flash、2 KB RAM、12 位 A/D 转换,2 个串口/SPI 接口, 48 个 I/O 口等) 。

4.1.2 远端原始方案

（1）主要器件

STC12LE5408--------1

DS18B20----------4

NRF24L01----------1

电容电阻若干

（2）系统框图

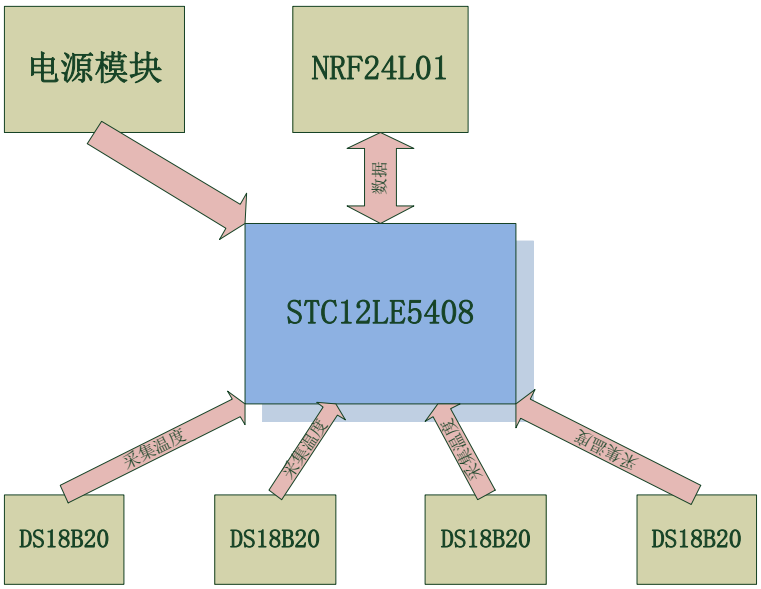


图 4.2

（3）实现原理

A. 用 51 单片机 STC12LE5408 来收集并 DS18B20 的数据并进行相应的转换。

B. 用 4 个 DS18B20 采集 4 路温度。有对应的控制字确定要采集哪一路温度。

C. 用 NRF24L01 完成对数据的无线收发功能

4.2 最终方案

4.2.1 本地端方案

（1）主要器件

MSP430F149 最小板------1

蜂鸣器----------1

4*4 矩阵键盘----------1

NRF905----------1

LCD12864----------1

电阻电容若干

（2）系统框图

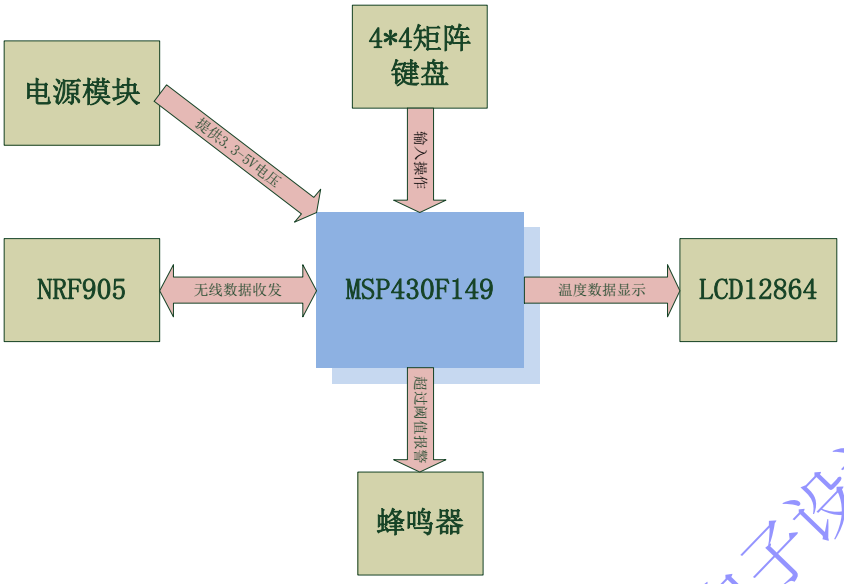


图 4.3

（3）实现原理：

A. MSP430F149 作为总控制器。用来接收判断按键，接受来自 NRF905 的数据并将数据通过 NRF905 发送给远端，同时将温度数据传递给 LCD12864 让其显示。超过阈值电压后，通过 MSP430F149 的 GPIO 口控制蜂鸣器的报警。

B. 主程序中以判断键盘按下哪一个案件展开。当收到按键终端以后执行相应的程序。采用 4*4 矩阵键盘，我们使用其中的 8 个按键。其中 4 个按键用于调整阈值温度，两个用于调整上限阈值温度，两个用于调整下限阈值温度。另外四个按键用于切换路数。

C. LCD12864 能够显示温度阈值，当前路的温度已经温度波形曲线。

D. NRF905 采用双向收发模式，同时发送控制命令和接收远端温度数据。

（4）方案改进解释

A. 最终我们采用 MSP430F149 的最小核心板，是因为由于实验室的限制，我们之前够得的是 TI 公司的芯片不能完成 PCB 布板的工作，且武汉市场上 MSP430F449 的最小系统板较少见，而是以 MSP430F149 的最小系统板为主导，故我们最终决定选用 MSP430F149 的最小系统板。

B. 原先计划采用 NRF24L01 作为无线收发模块的主芯片。但是由于在后期调试中，我们多够得的 NRF24L01 模块调试不成功，且经查资料，NRF905 的稳定性更好，所以我们最终才用了 NRF905 的无线收发模块。经调试后能够正确的接发数据。

4.2.2 远端最终方案

（1）主要器件

MSP430F149 最小板-----1

DS18B20----------4

NRF905----------1

电容电阻若干

（2）系统框图

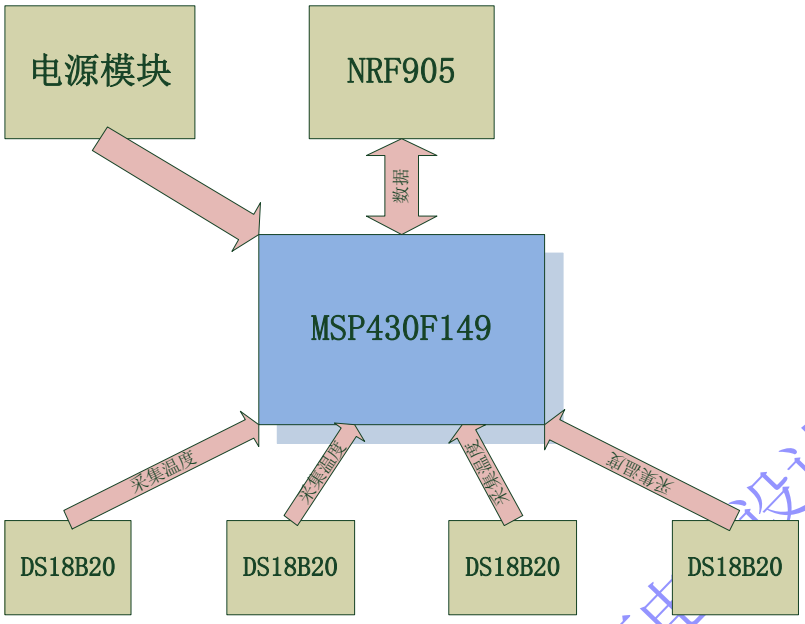


图 4.4

（3）实现原理

A. 用 MSP430F149 来收集并 DS18B20 的数据并进行相应的转换。

B. 用 4 个 DS18B20 采集 4 路温度。有对应的控制字确定要采集哪一路温度。DS18B20 与 MSP430 的 4 个 GPIO 口相连。

C. 用 NRF905 完成对数据的无线收发功能，实行双向通信。

（4）方案改进解释

原先计划采用 51 单片机进行远端的操控，是因为 51 单片机价格便宜，且在市面上容易获得。初期由于任务分配不合理，远端的温度采集和无线收发都让一个同学来做，但是在 51 实验箱上一直接收不到信号，故最终打算两端均用 MSP430F149 芯片，无线模块由一个同学完成，远端的温度采集模块由另一个同学完成设计。

5 硬件设计与实现

5.1 本地端

5.1.1 无线模块

（1）芯片介绍

nRF905 单片无线收发器是挪威 Nordic 公司推出的单片射频发射器芯片，工作电压为 1.9-3.6V，32 引脚 QFN 封装（5mm×5mm），工作于 433/868/915MHz3 个 ISM 频道（可以免费使用）。其工作在 433/868/915MHZ 的 ISM 频段，是一个完全集成的频率调制器，一个带解调器的接收器，一个功率放大器，一个晶体振荡器和一个调节器组成。

该器件有 4 种工作模式，即发送模式，接受模式，掉电模式和 standby 模式。工作在收发状态时即为工作在 ShockBurst 模式。该模式的特点是自动产生前导码和 CRC，客户以很容易通过 SPI 口进行编程配置。电流消耗很低，在发射功率为-10dbm 时发射电流为 11mA,接收电流为 12.5mA.进入 powerdown 模式以后可以很容易实现节电。

无线模块如图所示

图 5.1

（2）与 MCU 连接图

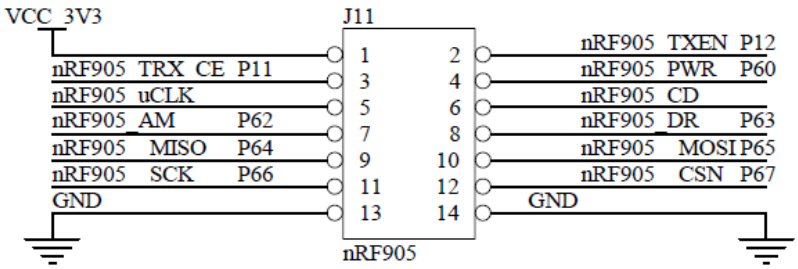


图 5.2

由图可知，我们并没有将 NRF905 的 SPI 口与 MSP430F149 的 SPI 口相连。这是为了在程序实现上更简单一些，直接通过通用 GPIO 口调试程序，通过软件方式控制 NRF905 会相对简单。

（3）硬件原理

NRf905 的使用非常简单。其工作发送或接收时工作在 ShockBurst 的接收（RX）和发送（TX）模式。

我们通过配置 NRF905 的 TRX_CE，PWR _UP 和 TX_EN 来设置工作模式。

其规则如下表所示：

PWR_UP	TRX_CE	TX_EN	工作模式
0	X	X	掉电和 SPI 编程
1	0	X	Standby 和 SPI 编程
1	1	0	ShockBurst RX
1	1	1	ShockBurst TX

图 5.3

典型的 ShockBurst TX 模式的配置方式如下：

A. 当应用 MCU 有遥控数据节点时，接收节点的地址和有效数据的地址通过 SPI 接口传送给 NRF905，应用协议或 MCU 设置接口速度。

B. MCU 设置 TRX_CE，TX_EN 为高来激活传输模式。

C. 无线系统自动上电，数据包打包完成并发送。

D. 若 AUTO_RETRAN 被设置为高，NRF905 将连续发送数据包，直到 TRX_CE 被设置为低。

当 TRX_CE 被设置为低，NRF905 结束数据传输并将自己设置成 standby 模式。

注：NRF905 在发送数据时，无论 TRX_CE 和 TX_EN 如何被设置，数据总能完整发送出去。

直到发送结束后，新的模式才会被激活。

A. 典型的 ShockBurst RX 模式的配置方式如下：
B. 设置 TRX_CE 为高，TX_EN 为低选择 ShockBurst RX 模式
C. 650us 以后 NRF905 检测空中信息。
D. 当 NRF905 发现和接收频率相同的载波时，载波检测 CD 被置高。
E. NRF905 接收到有效的地址时，地址匹配 AM 被置高。
F. NRF905 接收到有效的数据包时，数据准备就绪位（DR）置高。
G. MCU 将 TRX_CE 置低，进入 standby 模式。
H. MCU 以合适的速率通过 SPI 读出有效数据。
I. 数据读完后，NRF905 将 AM 和 DR 置低。
J. NRf905 再准备进入其他模式。
注意：在接受数据的时候如果 TRX_CE 或 TX_EN 状态改变，NRF905 就会改变工作模式，数据包丢失。

（4）重要时序

NRF905 在各个状态间的转换所需的时间如下图表所示。软件编程时需要严格按照以下的时序，否则会导致数据丢包严重，甚至器件不能正常工作。

nRF905 时序	最大值
PWR_DWN→ST_BY 模式	3ms
STBY→TX ShockBurst 模式	650us
STBY→RX ShockBurst 模式	650us
RX ShockBurst→TX ShockBurst 模式	550us
TX ShockBurst→RX ShockBurst 模式	550us

图 5.4

5.1.2 液晶显示模块

（1）模块介绍

液晶模块我们采用的是绘晶公司的 HJ12864ZW。

HJ12864ZW 中文字库型液晶显示模块可以显示字母、数字、中文字型及图形，具有绘图及文字画面混合显示功能。可显示 128（列）× 64（行）点阵。8×4 个（16×16 点阵）中文汉字。提供三种控制接口： 8 位微处理器接口、4 位微处理器接口、串行接口。该模块内置 2M-位元中文字型 ROM (CGROM) 总共提供 8192 个中文字型(16×16 点阵)，16K-位元半宽字型 ROM (HCGROM) 总共提供 126 个符号字型(16×8 点阵)，64×16-位元显示 RAM (DDRAM)，另外绘图显示画面提供一个 64×256 点的绘图区域（GDRAM），可以和文字画面混和显示。提供多功能指令：画面清除（Display clear）、光标归位（Return home）、显示打开/关闭（Display on/off）、光标显示/隐藏（Cursor on/off）、显示字符闪烁（Display character blink）、光标移位（Cursor shift）、显示移位（Display shift）、垂直画面卷动（Vertical line scroll）、反白显示（reverse display）。
其实物图如图 5.5 所示

图 5.5

（2）与 MCU 连接图

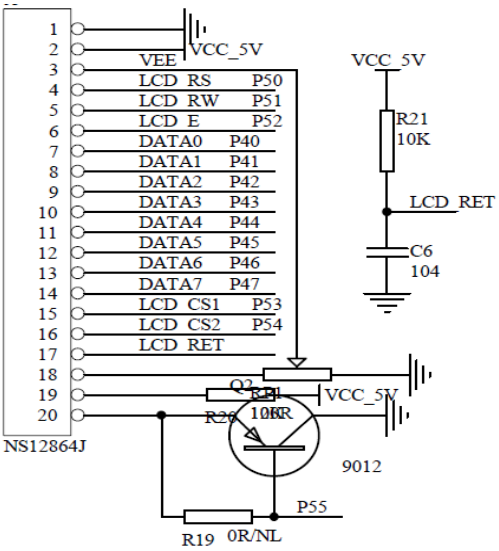


图 5.6

（3）硬件原理

模块引脚说明

引脚号	引脚名称	方向	功能说明
1	VSS	-	模块的电源地
2	VDD	-	模块的电源正端
3	V0	-	LCD 驱动电压输入端
4	RS(CS)	H/L	并行的指令/数据选择信号；串行的片选信号
5	R/W(SID)	H/L	并行的读写选择信号；串行的数据口
6	E(CLK)	H/L	并行的使能信号；串行的同步时钟
7	DB0	H/L	数据 0
8	DB1	H/L	数据 1
9	DB2	H/L	数据 2
10	DB3	H/L	数据 3
11	DB4	H/L	数据 4
12	DB5	H/L	数据 5
13	DB6	H/L	数据 6
14	DB7	H/L	数据 7

15	PSB	H/L	并/串行接口选择：H-并行；L-串行
16	NC		空脚
17	/RET	H/L	复位 低电平有效
18	NC		空脚
19	LED_A	-	背光源正极（LED+5V）
20	LED_K	-	背光源负极（LED-OV）

逻辑工作电压（VDD）：4.5~5.5V
电源地（GND）：0V
工作温度（Ta）：0~60℃

LCD12864 的连接方法分为串行连接方式和并行连接方式。
并行连接方式时序图如下所示：

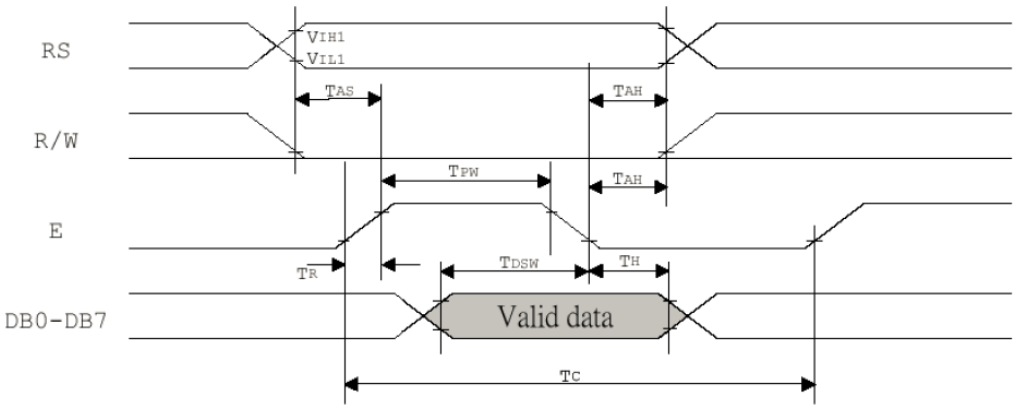


图 5.7 并行连接方式时序图（写）

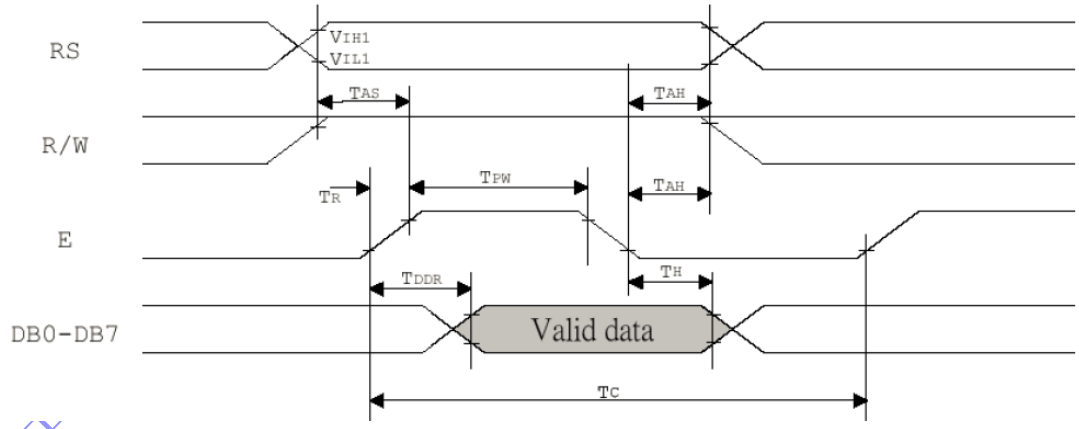


图 5.8 并行连接方式时序图（读）

串行数据传送共分三个字节完成：
第一字节：串口控制—格式 11111ABC
A 为数据传送方向控制：H 表示数据从 LCD 到 MCU，L 表示数据从 MCU 到 LCD
B 为数据类型选择：H 表示数据是显示数据，L 表示数据是控制指令
C 固定为 0
第二字节：(并行)8 位数据的高 4 位—格式 DDDD0000
第三字节：(并行)8 位数据的低 4 位—格式 0000DDDD
串行接口时序参数：(测试条件：T=25℃ VDD=4.5V)

在 LCD12864 中，英文显示需要 1 个字节，中文显示需要 2 个字节，故想要显示汉字前，要先取汉字字模。

5.1.3 矩阵键盘模块

（1）模块介绍

我们采用薄膜键盘。其由杜邦线引线，方便做外观设计。

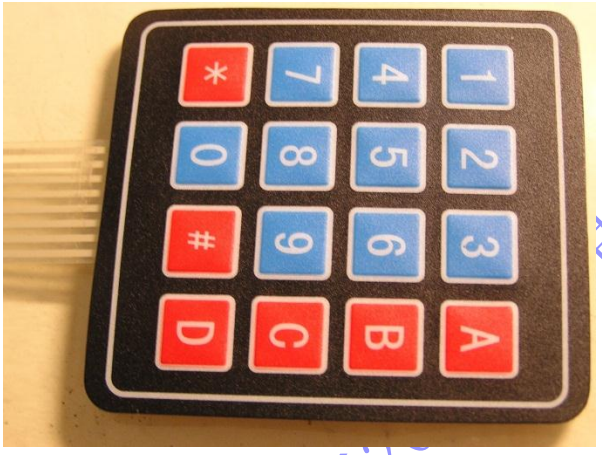


图 5.9

（2）与 MCU 连接图

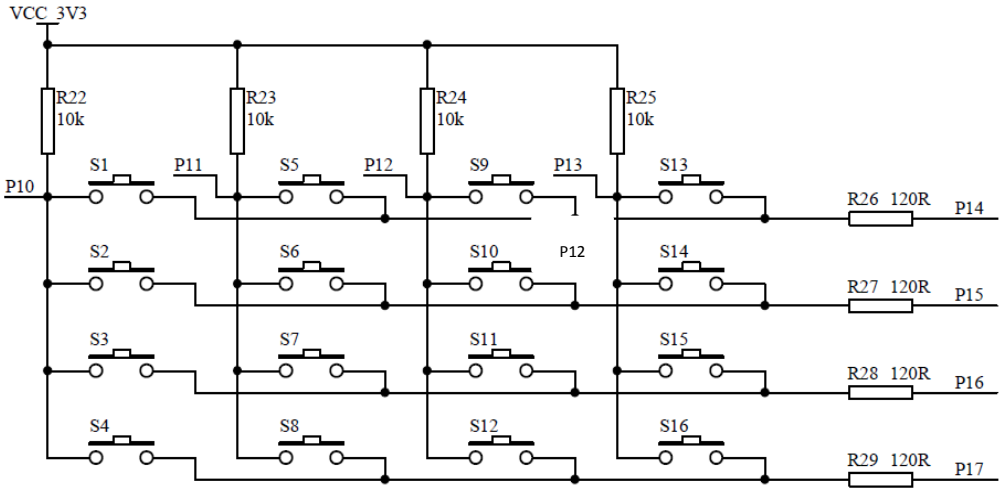


图 5.10

（3）硬件原理

矩阵式键盘使用于按键数量较多的场合，它由行线与列线组成，按键位于行、列的交叉点上，行、列线分别列接到按键开关的两端。行线通过上拉电阻接到+5V 上。无键按下时，行线处于低电平状态，而当有按键按下时，行线电平状态将由与此行线相连的列线电平一样为高电平。这是识别矩阵键盘按键是否被按下的关键所在。一个 4x4 的行列可以构成一个 16 按键的键盘。

本次以扫描法来识别按键。在扫描法中分两步处理按键，首先是判断有无键按下，让所有的列线置高电平，检查各行线电平是否有变化，如行线有一个为高，则有键按下。当判断有键按下时，使列线依次变低，其余各列为高电平，读行线，进而判断出具体哪个键被按下。

下表为 7 段共阴极段码表：

5.1.4 报警模块

（1）模块介绍

报警模块我们采用有源蜂鸣器。蜂鸣器使用较喇叭相对简单，无需控制输出脉冲的频率，只要一直输出高电平就会发出蜂鸣声。我们将蜂鸣器接 P3.0 口用 P3OUT 控制蜂鸣器的发声。通过对收到温度数据的软件处理判断是否超出了阈值温度。

实物图如下所示：

图 5.11

（2）与 MCU 连接图

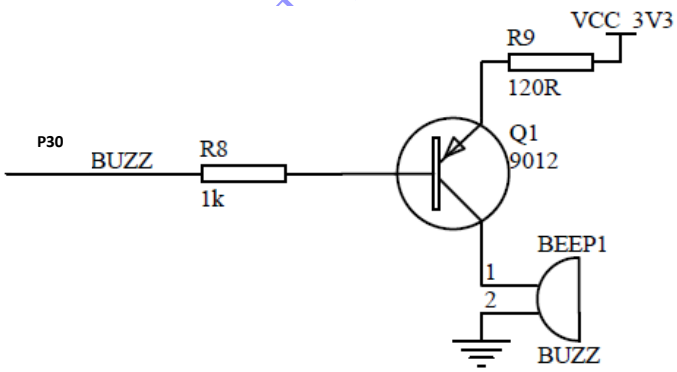


图 5.12

5.2 远端

5.2.1 整机电路图

电路图说明

5.2.2 无线模块　远端的无线模块与本地端的无线模块相同，故请参考 5.1.2。

5.2.3 温度采集模块

(1)模块介绍

DS18B20 是美国 DELLA 公司生产的可组网数字温度传感器芯片，具有耐磨耐磨，体积小，使用方便，封装形式多样等特点，适用于各种狭小空间设备数字测温和控制领域。DS18B20 与微处理器连接时仅需要一条口线即可实现微处理器与 DS18B20 的双向通讯。在使用中不需要任何外围元件。可用数据线供电，电压范围：+3.0~ +5.5 V。测温范围为-55 ~+125 ℃。固有测温分辨率为 0.5 ℃。通过编程可实现 9~12 位的数字读数方式。用户可自设定非易失性的报警上下限值。

实物图如下图所示

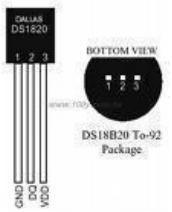


图 5.13

（2）与 MCU 连接图

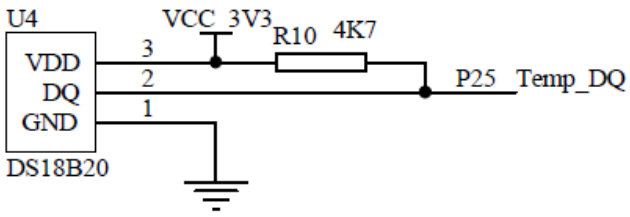


图 5.14

（3）硬件原理

DS18B20 的初始化时序如下图所示：

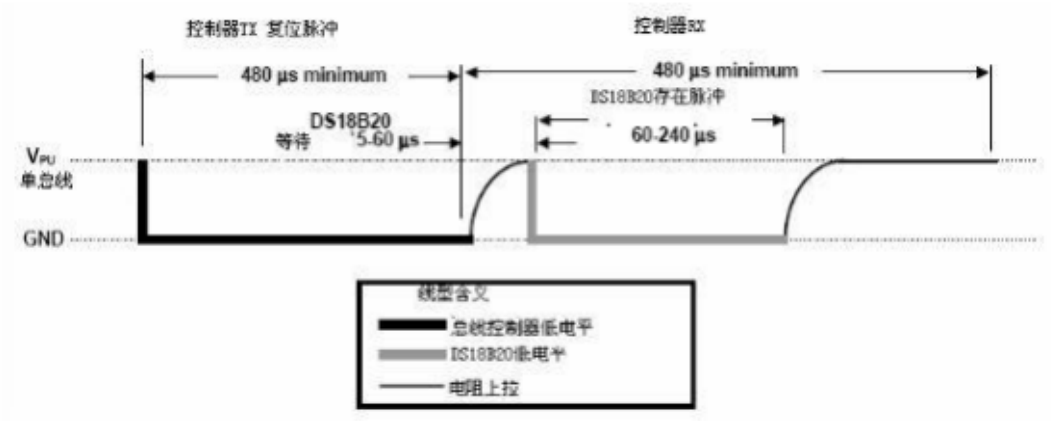


图 5.15

DS18B20 的读写时序如下图所示：

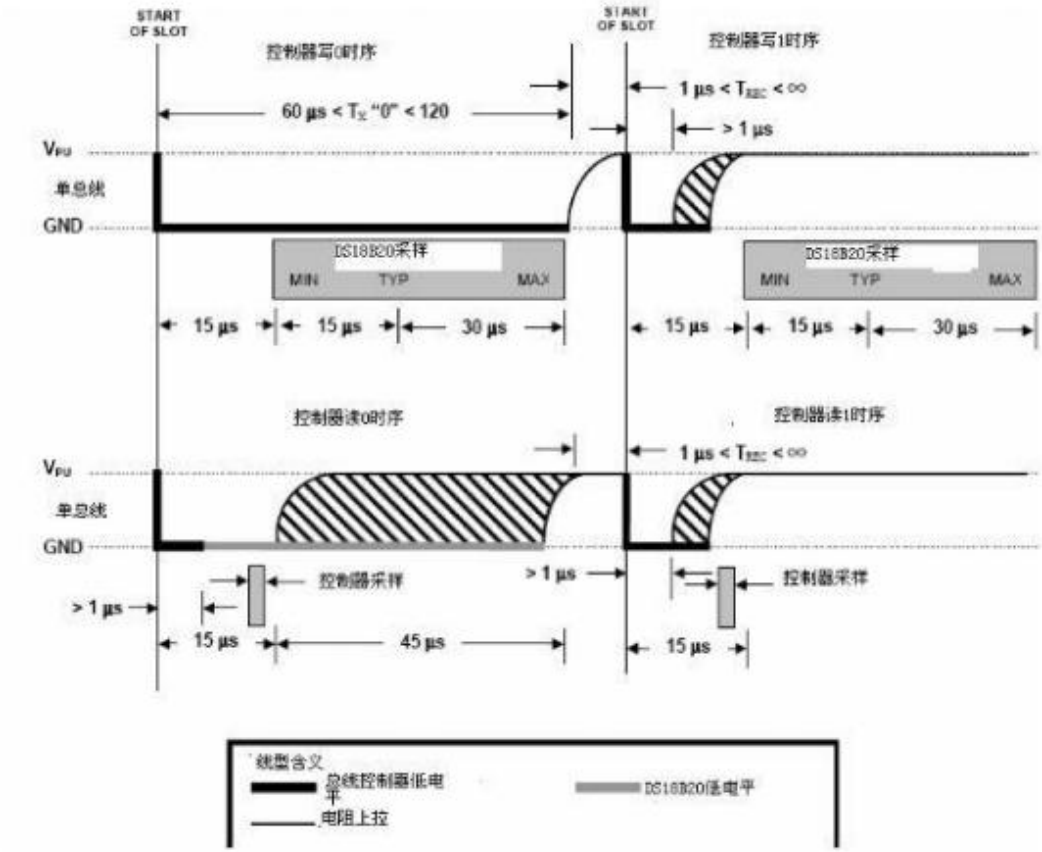


图 5.16

当 DS18B20 接收到温度转换命令后，开始启动转换。转换完成后的温度值就以 16 位带符号扩展的二进制补码形式存储在高速暂存存储器的第 1，2 字节。单片机可通过单线接口读到该数据，读取时低位在前，高位在后，数据格式以 0.062 5 ℃/LSB 形式表示。温度值格式如图 3.4 所示。

2^1	2^1	2^1	2^1	2^0	2^{-1}	2^{-1}	2^{-1}

MSB　　LSB

S	S	S	S	S	S	S	S

MSB LSB

这是 12 位转化后得到的 12 位数据，存储在 18B20 的两个 8 比特的 RAM 中，二进制中的前面 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于 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为 1，测到的数值需要取反加 1 再乘于 0.0625 即可得到实际温度。图中，S 表示位。对应的温度计算：当符号位 S=0 时，表示测得的温度植为正值，直接将二进制位转换为十进制；当 S=1 时，表示测得的温度植为负值，先将补码变换为原码，再计算十进制值。例如+125℃的数字输出为 07D0H，+25.0625℃的数字输出为 0191H，—25.0625℃的数字输出为 FF6FH，—55℃的数字输出为 FC90H。DS18B20 温度传感器主要用于对温度进行测量，数据可用 16 位符号扩展的二进制补码读数形式提供，并以 0.0625℃／LSB 形式表示。

6.软件设计与实现

6.1 本地端

6.1.1 整机系统软件流程图

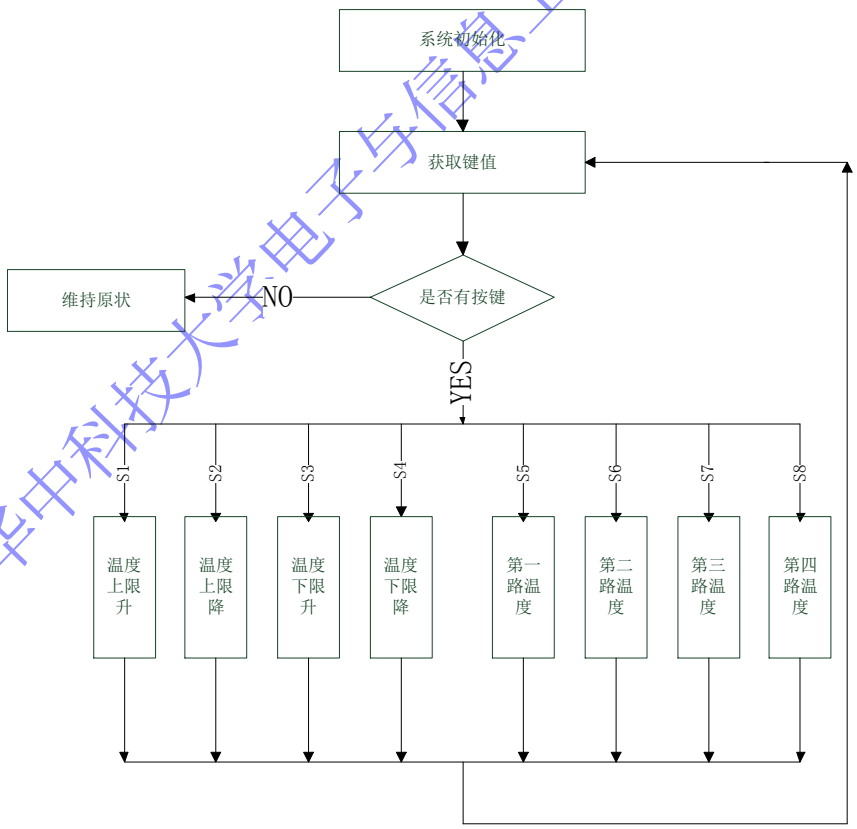


图 6.1

6.1.2 无线模块软件设计

（1）设计思想

本地端采用单工模式，只接受远端发送来的数据。远端发送来的数据为 char 型，8 个字节，，每收到一个数据我们就将其存在 RxBuf[8]中，格式如下（以发送 31.2℃为例）：

“3”	“1”	“.”	“2”	X	x	0x01	0x00

该 8 字节的前 6 个字节用来存储温度信息，第 5，6 两个字节为摄氏度的符号的编码。第七个字节用来指示这是第几路信号：

0x01----------第一路
0x02----------第二路
0x04----------第三路
0x08----------第四路

本地端根据案件来选择要接受第几路信号，即直接判断 RxBuf[6]是否为该路对应的标识.若接受正确，就将该数据送给液晶显示。

（2）程序流程图

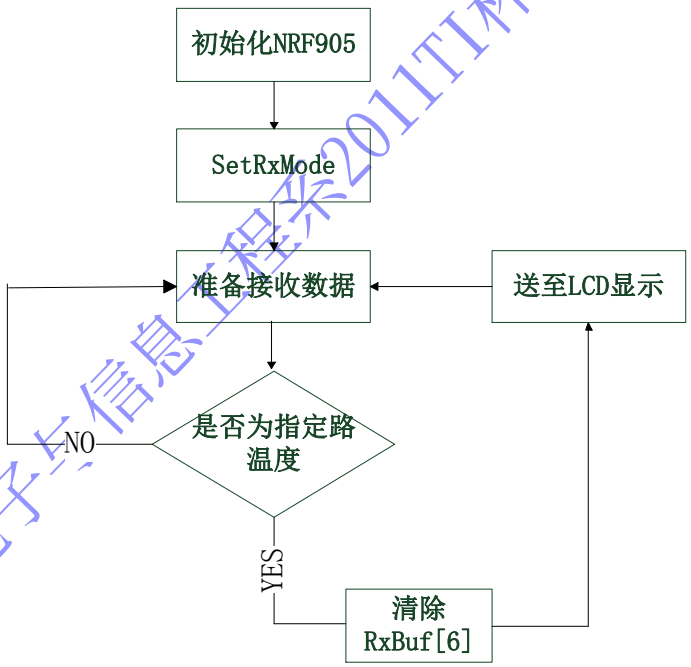


图 6.2

（3）重要代码

```
//NRF905 配置
extern unsigned char RFConf[11]=
{
   0x00, -------(1)                          //配置命令
   0x6c,--------(2)                          //CH_NO,配置频段在 430MHZ
   0x0c,--------(3)                          //输出功率为 10db,不重发，节电为
//正常模式
   0x44, --------(4)                         //地址宽度设置，为 4 字节
   0x08,0x08,------(5)                       //接收发送有效数据长度为 4 字节
   0xCC,0xCC,0xCC,0xCC,-------(6)            //接收地址
```

0x58,-------(7)　　//CRC 充许，8 位 CRC 校验，外部时钟//信号不使能，16M 晶振

}

解释:

(3)

0	1	2	3	4	5	6	7

其中第 5，6 位表示输出功率，00 代表-10dbm，01 代表-2dbm,10 代表 6dbm,11 代表 10dbm。

(4)

0	1	2	3	4	5	6	7

其中 1，2，3 位表示发送地址宽度 001 表示 1 字节，100 表示 4 字节

5，6，7 位表示接受地址宽度，001 表示 1 字节，100 表示 4 字节

(5)

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

低 8 位表示接收有效数据宽度

高 8 位表示发送有效数据宽度

0x01--------1 字节

0x02--------2 字节

0x04--------4 字节

0x08--------8 字节

0x10--------16 字节

0x20--------32 字节

(7)

0	1	2	3	4	5	6	7

第 0 位表示 CRC_模式，0 表示 8 位 CRC 校验，1 表示 16 位 CRC 校验

第 1 位表示 CRC 校验允许，0 表示不允许

第 2，3，4 位表示晶体振荡器频率

000-----4MHZ

001-----8MHZ

010-----12MHZ

011-----16MHZ

100-----20MHZ

第 5 位表示输出时钟使能，0 为没有外部时钟

第 6，7 位表示输出时钟频率

00------4MHZ

01------2MHZ

10------1MHZ

11------500KHZ

//接收端代码：

```
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;//关闭看门狗
    Portinit();      //初始化端口
    Init_CLK();      //初始化时钟
```

```
    Lcd_Init();      //初始化 LCD
    nRF905Init();      //初始化 NRF905
    Config905();      //启用 NRF905，准备配置
    RD_Config905();   //配置 NRF905 状态寄存器
    float temp;            //定义 temp 型，将 char 型数据转换成 temp 型与阈值比较
    while(1)
    {
        RX()       //接受数据
        if(RxBuf[6]==0x04)   //我们假设要接受第 3 路信号
        {
            temp=get_temp_float(RxBuf);//将 char 型温度转换成 float 型
            if((temp>=temp_high)||(temp<=temp_low))   与温度阈值相比较
            {
               P2OUT&=~BIT6;
             }
            if((temp<temp_high)&&(temp>temp_low))
             {
             P2OUT|=BIT6;
            }
           printfbuf[0]=RxBuf[0];   //将接收到的数据存入 printfbuf 中交给 LCD 显示
           printfbuf[1]=RxBuf[1];
           printfbuf[2]=RxBuf[2];
           printfbuf[3]=RxBuf[3];
           printfbuf[4]=RxBuf[4];
           printfbuf[5]=RxBuf[5];
           printfbuf[6]=0x00;
           printfbuf[7]=0x00;
           W_Menu1();     //LCD 显示
        }

        Delay_Ms(50);
    }
}
```

6.1.3 液晶显示模块。

（1）设计思想

我们最终只打算让液晶屏显示汉字和数字。液晶屏要能显示出阈值温度和当前路数的温度，当按下调整阈值温度的按键，显示出阈值温度的变化，当按下路数按键，则能够显示指定路数的温度和设定的阈值电压。当本地端接收到新的温度数值时才会刷新屏幕，否则温度屏幕不会变化。

（2）程序流程

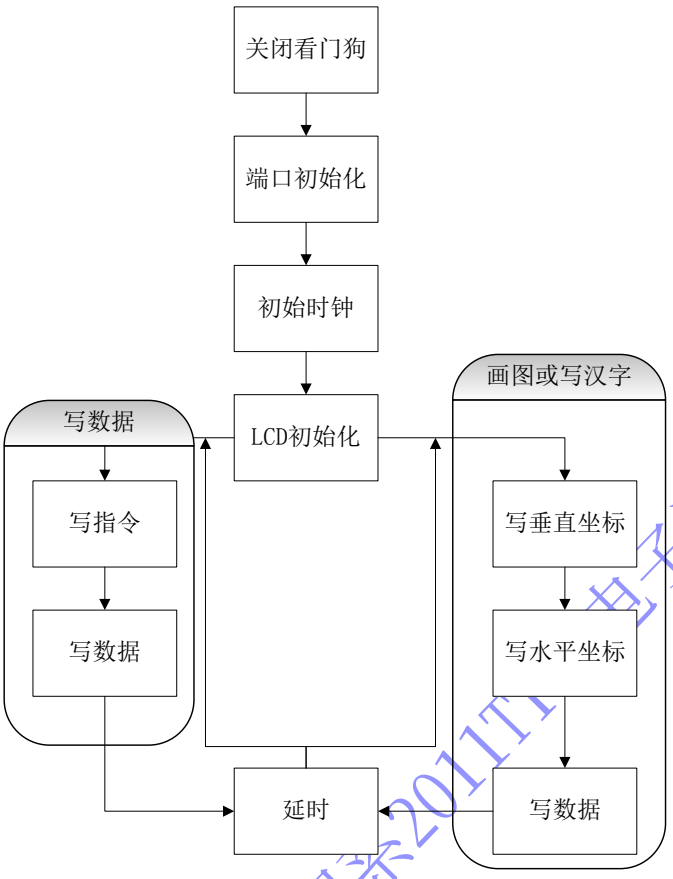


图 6.3

（3）重要代码

```
/*****写指令代码*****/
void Lcd_WriteCmd(uchar cmdcode)
{
    CLR_LCD_RS;
    CLR_LCD_RW;
    SET_LCD_EN;
    WaitBusy();
    LCM_Data_Out = cmdcode;
    CLR_LCD_EN;
}
```

先将 RS 置 0，表示写指令；再将 RW 置 0，表示 MPU 写入 LCD；再将 EN 使能置 1，表示将开始传数据，WaitBusy 的作用是延时一小段时间，等待 LCD 空闲；然后再将控制代码写入所分配的管脚，最后将 EN 置 0

```
/*****写数据代码*****/
void Lcd_WriteData(uchar dispdata)
{
    SET_LCD_RS;
    CLR_LCD_RW;
    SET_LCD_EN;
    WaitBusy();
```

```
    LCM_Data_Out = dispdata;
    CLR_LCD_EN;
}
```

与写指令的区别就是先将 RS 置 1，表示写数据；再将 RW 置 0，写数据

```
/*****写汉字代码*****/

void hanzi_Disp(uchar x,uchar y,uchar    *s)
{               //x、y 为汉字坐标
    Lcd_WriteCmd(addr_tab[8*x+y]);   //写地址
    while(*s>0)
    {
        Lcd_WriteData(*s);      //写数据
        s++;
    }
}
```

在 lcd12864 头文件当中设定汉字表后，屏幕可以显示 8*4 个汉字，一排八个，一共四行

```
/*****画图像代码*****/
void pic_Disp(uchar *pic)       //显示 Gdram 内容（显示图片）
{
  uchar x,y,i;
  for(i=0;i<9;i=i+8)
  for(y=0;y<32;y++)
  {
    for(x=0;x<8;x++)
    {
      Lcd_WriteCmd(0x36);   //扩充指令，开绘图显示
      Lcd_WriteCmd(0x80+y);   //行地址
      Lcd_WriteCmd(0x80+x+i);   //列地址
      Lcd_WriteData(*pic++);   //写数据  D15－D8
      Lcd_WriteData(*pic++);   //写数据  D7－D0
      Lcd_WriteCmd(0x30);
    }
  }
}
```

先控制扩展指令打开绘图功能

打点原理：将整个屏幕分为上半屏和下半屏，有 128*32*2 个点（像素），根据确定他们横纵坐标来控制某个点的亮灭。

```
/*****显示温度*****/
sprintf(Buff,"%.1f℃",setuppertemp);
hanzi_Disp(1,0,"阈值上限：");
hanzi_Disp(1,5,(unsigned char*)Buff);
```

先将阈值上限放入 char 型的 Buff 中，再以 c 语言的显示形式显在 LCD 上

6.1.4 键盘模块

（1）设计思想

我们采用的键盘为 4X4 的行列式键盘，总共有 16 个按键，足够我们使用。由于按键足够，我们采用了其中 8 个按键来实现功能切换。其中，S1 为温度上限增加，S2 为温度上限减小，S3 为下线温度增加，S4 为下线温度减小，S5 为第一路温度，S6 为第二路温度，S7 为第三路温度，S8 为第四路温度。

（2）程序流程

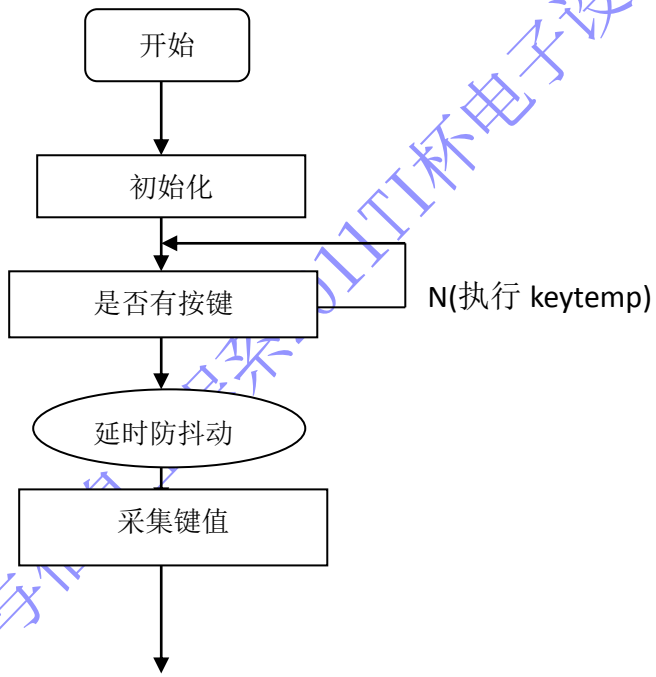


图 6.4

（3）重要代码

//按键判断程序

原理，当按键按下的时候，对应 PxIN 端口的值置 0，否则为 1（默认为 1），由此与 0x0F 相与，可根据相与之后的值来判断时候有按键按下。

程序代码如下：

```
unsigned char _keyCheck(void)
{
  unsigned char x=0x00;
  P1SEL=0x00;
  P1DIR |=BIT7+BIT6+BIT5+BIT4;
  P1OUT =0x00;
  P1OUT &= ~0xF0;
  x=P1IN&0x0F;
  return(x);
```

```
}

//按键扫描程序
   按键扫描程序代码如下：
   unsigned char _keyScan(void)
{
   //P1DIR =0xf0;
      P1OUT = 0xef;//扫描第一列
      _NOP();
      _NOP();
      //P1DIR &= 0x0f;//读列
      if((P1IN & 0x0f)==0x0e)return '1';
      if((P1IN & 0x0f)==0x0d)return '5';
      if((P1IN & 0x0f)==0x0b)return '9';
      if((P1IN & 0x0f)==0x07)return 'D';

      //P1DIR =0x0f;
      P1OUT = 0xdf;//扫描第二行
      _NOP();
      _NOP();
      //P1DIR &= 0x0f;//读列
      if((P1IN & 0x0f)==0x0e)return '2';
      if((P1IN & 0x0f)==0x0d)return '6';
      if((P1IN & 0x0f)==0x0b)return 'A';
      if((P1IN & 0x0f)==0x07)return 'E';

      //P1DIR =0xff;
      P1OUT = 0xbf;//扫描第三行
      _NOP();
      _NOP();
      //P1DIR &= 0x0f;//读列
      if((P1IN & 0x0f)==0x0e)return '3';
      if((P1IN & 0x0f)==0x0d)return '7';
      if((P1IN & 0x0f)==0x0b)return 'B';
      if((P1IN & 0x0f)==0x07)return 'F';

      //P1DIR =0xff;
      P1OUT = 0x7f;//扫描第四行
      _NOP();
      _NOP();
      //P1DIR &= 0x0f;//读列
      if((P1IN & 0x0f)==0x0e)return '4';
      if((P1IN & 0x0f)==0x0d)return '8';
      if((P1IN & 0x0f)==0x0b)return 'C';
```

```
    if((P1IN & 0x0f)==0x07)return '0';
    return 0;
}
```

6.1.5 报警模块

（1）设计思想

因为本地接收到的数据是以 char 形式存储的，即 char RxBuf[8]。为了便于比较，我们首先要将这 8 个字节变成 float 型数据。

RxBuf 的存储方式如下表所示，以 31.2℃为例

“3”	“1”	“.”	“2”	X	x	0x01	0x00
0	1	2	3	4	5	6	7

我们只用对第 0，1，3 位进行转换即可。由于字符形式存储的都是 ASCII 码，故我们只需要将每个字符对应的 ASCII 码值减去 48 就可以了。

具体公式如下

temp1=(RxBuf[0]-48)*10.0+(RxBuf[1]-48)*1.0+(RxBuf[3]-48)*0.1;

B.数据转换完成后，只需将其与温度阈值 temp_high 与 temp_low 比较即可，超出温度阈值后，将 P3OUT 置高，蜂鸣器开始报警。

（2）程序流程图

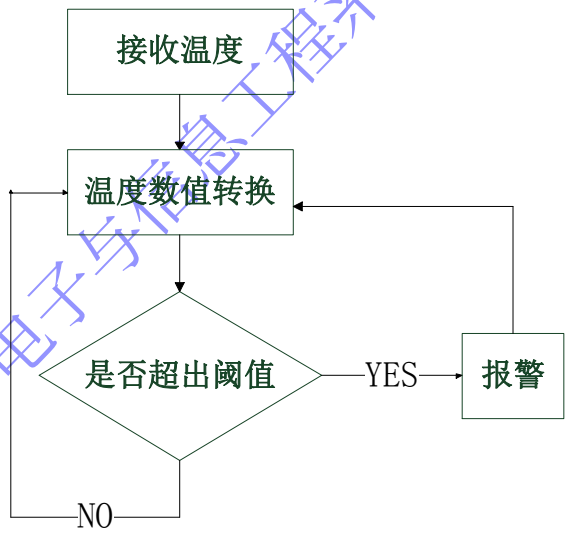


图 6.5

（3）重要代码

```
  //温度数值转换函数
      float get_temp_float(char *RxBuf)
{
   float temp1;
   temp1=(RxBuf[0]-48)*10.0+(RxBuf[1]-48)*1.0+(RxBuf[3]-48)*0.1;//   按照公式来转换
   return temp1;
}
//判断是否超出阈值
temp=get_temp_float(RxBuf);
```

```
if((temp>=temp_high)||(temp<=temp_low))//若超出阈值温度，就报警
{
   P3OUT&=~BIT0;
 }
if((temp<temp_high)&&(temp>temp_low)) //若回到阈值以内，不报警
 {
 P3OUT|=BIT0;
}
```

6.2 远端

6.2.1 无线收发模块

（1）设计思想

在远端，NRF905 工作在单工发送模式轮流发送 4 路的温度数据，在其发送的温度数据中还包含了该路的标志，用于本地端判断。

（2）程序流程图

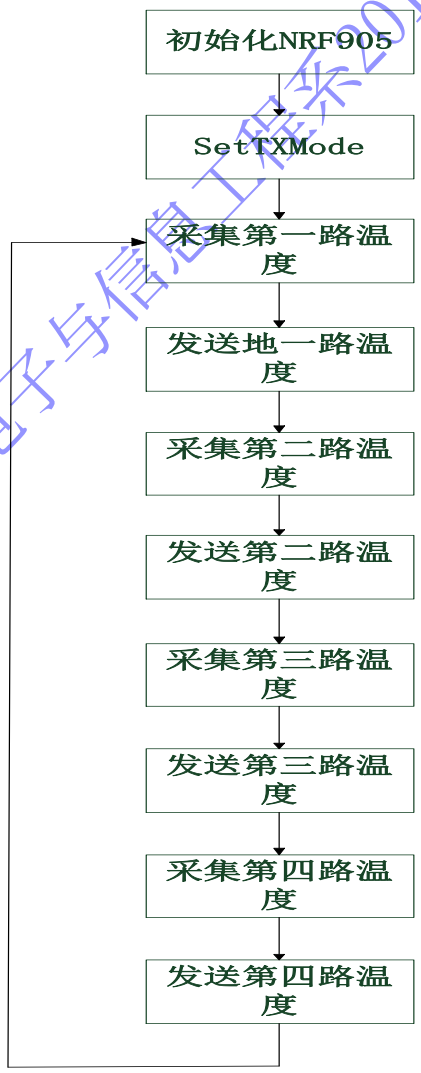


图 6.6

（3）重要代码

```
void main(void)
{
 //     unsigned char e=0;
      // 停止 WDT
      WDTCTL = WDTPW + WDTHOLD;
      Portinit();
      Init_CLK();
      nRF905Init();
      Config905();
      RD_Config905();

      while(1)
      {
            float temp;
            SetTxMode();

            temp = ReadTemperature1();//接收第一路温度
            sprintf(TxBuf,"%.1f℃",temp);//温度数值以 char 型存储
            TxBuf[6]=0x01;//加入路数标识
            TxPacket(TxBuf);//发送数据

            temp = ReadTemperature2();//第二路
            sprintf(TxBuf,"%.1f℃",temp);
            TxBuf[6]=0x02;



            temp = ReadTemperature3();//第三路
            sprintf(TxBuf,"%.1f℃",temp);
            TxBuf[6]=0x04;
            TxPacket(TxBuf);


            temp = ReadTemperature4();//第四路
            sprintf(TxBuf,"%.1f℃",temp);
            TxBuf[6]=0x08;
            TxPacket(TxBuf);
      }
}
```

6.2.2 温度采集模块

（1）设计思想

在远端我们用 4 个 DS18B20 来测量 4 路的温度，4 个 DS18B20 分别与 MSP430F149 的 4 个 P2 端口相连，每一次 MCU 都循环采样这 4 路数据并发送出去，由本地端选择要接收哪一路数据。

（2）程序流程图

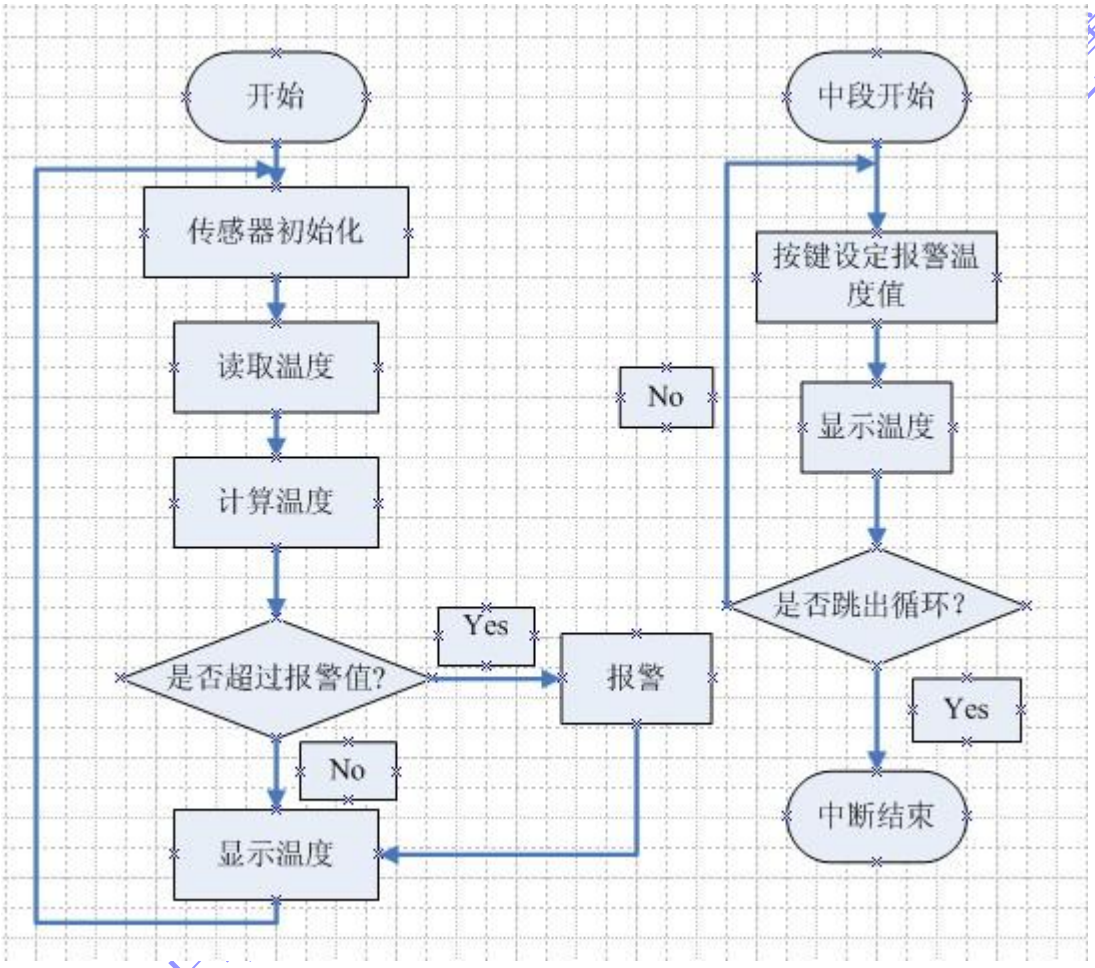


图 6. 7

（3）重要代码解释

根据 DS18B20 的初始化时序可以得到第一路 DS18B20 的初始化函数：

```
//初始化
void Init_DS18B20(void)
{
  SET_TEMP_DQ;          //DQ= 1 复位
  Delay_DS18B20(8);     //稍做延时
  CLR_TEMP_DQ;          //DQ = 0 单片机将 DQ 拉低
  Delay_DS18B20(80);    //精确延时，大于 480us
  SET_TEMP_DQ;          //DQ= 1 拉高总线
  Delay_DS18B20(14);
  P2DIR &= ~(BIT5);
  P2DIR |= BIT5;
```

```
  Delay_DS18B20(20);
}
```

根据 DS18B20 的读写时序可以得到 DS18B20 的读写函数：

```
//读 DS18B20
unsigned char ReadOneChar(void)
{
  unsigned char i=0;
  unsigned char dat = 0;
  for (i=8;i>0;i--)
  {

    CLR_TEMP_DQ;    // DQ = 0 给脉冲信号
    dat>>=1;
    SET_TEMP_DQ;    // DQ = 1 给脉冲信号
    //设置为输入
    P2DIR &= ~(BIT5);

    //if(DQ)
    //dat|=0x80;
    if(TEMP_DQ_In)
    {
       dat|=0x80;
    }
    //设置为输出
    P2DIR |= BIT5;
    Delay_DS18B20(4);
  }
  return(dat);
}

//写 DS18B20
void WriteOneChar(unsigned char dat)
{
  unsigned char i=0;
  for (i=8; i>0; i--)
  {
    CLR_TEMP_DQ;          // DQ = 0 给脉冲信号
    //DQ = dat&0x01;
    if(dat&0x01)
    {
       SET_TEMP_DQ;       // DQ = 1 给脉冲信号
    }
    else
    {
```

```
      CLR_TEMP_DQ;       // DQ = 0 给脉冲信号
    }
    Delay_DS18B20(5);
    SET_TEMP_DQ;         // DQ = 0 给脉冲信号
    dat>>=1;
  }
}
```

利用以上函数可以得到当前 DS18B20 的温度值。

```
//读温度
float ReadTemperature(void)
{
  unsigned char a=0;
  unsigned char b=0;
  unsigned int data = 0;
  float t=0;
  float tt=0;
  Init_DS18B20();
  WriteOneChar(0xCC);   //跳过读序号列号的操作
  WriteOneChar(0x44);   //启动温度转换
  Init_DS18B20();
  WriteOneChar(0xCC);   //跳过读序号列号的操作
  WriteOneChar(0xBE);   //读取温度寄存器
  a = ReadOneChar();    //读低 8 位
  b = ReadOneChar();    //读高 8 位
  data=b;
  data<<=8;
  t=data|a;
  tt=t*0.0625;
  t= tt*10+0.5;     //放大 10 倍输出并四舍五入
  return(t/10);
}

//获取温度
float GetTemp_18B20(void)
{
  float temper=1;
        Init_18B20();
        Delay_18B20(2000);
        Skip_18B20();
        Delay_18B20(2000);
        Convert_18B20();
        Delay_18B20(60000);
        Delay_18B20(60000);
```

```
        Init_18B20();
        Delay_18B20(2000);
        Skip_18B20();
Delay_18B20(2000);
        Read_D18B20();
        Delay_18B20(2000);
      temper = ReadTemp_18B20();
        Delay_18B20(5000);
        return temper;                          //返回读取的温度值
}
```

说明：

由 DS18B20 读取出来的温度的一共是 12 位，低四位是小数部分，高八位是整数部分，因此，将读取的温度值 data 乘以 1/16，即可得到正确的温度值（存放在 tt 内）。由于要求温度精度是 0.1 摄氏度，因此再令 temp=（tt*10+0.5）/10 即可得到所要求的精度。

7 模块测试与调试

调试思路：

我们将要两个模块之间通信的数据设置成全局变量，这样每个人负责的模块先单独调通以后，再将所有的全局变量融合在一起，这些全局变量就是联系各个模块的枢纽。比如说，LCD 需要显示阈值电压和当前温度数值，那么阈值电压就是按键模块与 LCD 模块的桥梁，而温度数值就是 LCD 模块与本地无限模块的连接点。

7.1 LCD 显示程序

我们首先让 LCD 能够在屏幕上正确显示阈值温度，当前温度。显示格式如下所示：

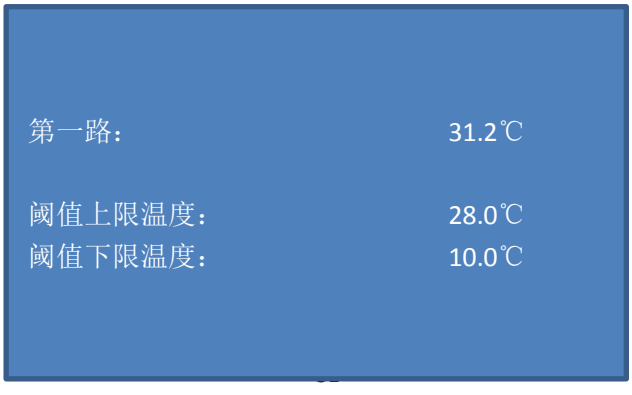


图 7.1

我们很顺利的在初期就能让 LCD 按照要求正确显示，实例图如下所示：

7.2 无线模块

无线模块的调试分为三部分。

(1) 做好相应的配置工作以后，我们让远端给本地端发送一个 8 个字节的 uchar 字符串（选择 8 个字节的原因是 LCD 模块用 8 个字节显示），并且字符串的第一个字节的数据为 0x01，本地端收到数据并存在 RxBuf 中以后，判断接收到的数据的第一个字节是否为 0x01，是的话就会让发光二极管亮。第一次调试便成功，并且接受频率很快。

(2) 本地端将接收到的数据送至 LCD 显示。将 LCD 的显示缓存变为 RxBuf 以后，能够在图 7.1 中第一行的“第一路”后面正确显示接收到的数据。受到 LCD 刷频的影响，接收频率略有降低但是仍然较快。

(3) 让远端循环发送 4 个不同的数据，本地端在主程序中设置想要接收哪一路数据（由前可知，当发送多路数据时我们会在数据后面加上标识符）并将其送往 LCD 显示。这个时候接收的频率明显降低，较为理想的情况大约变为 1 秒一次。通过设置 NRF905 接收数据的延迟，可以调整接收的频率。但目前的测试结果仍是 1 秒一次为上限。

7.3 键盘模块

编写好简单的按键扫描程序以后，我们让 LCD 屏幕显示不同按键对应的内容，一次成功。

后期，我们再将本地端所有模块嵌入到按键模块中，这将在随后提到。

7. 4 报警模块

(1) 先设置 3 个 float 型的全局变量，分别为 Temp_H(阈值温度高),Temp_L（阈值温度低）和 temp1(当前温度)，用 get_temp_float 函数将 uchar 型数据变为 float 型存入 temp1 中，然后再判断其与阈值温度的大小。

(2) 让远端发送设定的温度数据给本地端，收到数据后将数据转换为 float 型存入 temp1 中，再与阈值温度比较，超限则报警。

7.5 温度采集模块

我们直接将温度采集模块采集到的温度数值（float）用 sprintf 函数转变为 uchar 型发送给本地端，再在本地端显示出来。在无线模块正常运作的情况下，单路调试成功。
当在远端循环采集 4 路温度并发送的时候，本地端接收频率明显降低，通过调试本地端接收函数中延迟大小可以将接收频率控制在 1.5 秒一次。

7.6 整机调试

整机调试的原则是以键盘为中心，一步步将模块按键对应的处理程序中。
调试流程如下：

A. 将 8 个按键对应要实现的功能在 LCD 上显示出来。比如说，其中 4 个按键是用来调节阈值温度的，那么当按这 4 个键的时候，LCD 就显示出阈值温度的增加或降低。另外 4 个按键用来选择路数，每一路要显示的图形即为图 7.1 所示

B. 将无线模块加入进来。远端不采集温度，只是给本地端发送同一个数据，LCD 要能显示由无线模块接收到的数据。

C. 远端仍然不采集温度，让远端循环发送四路固定的数据，让 LCD 显示。

D. 在本地端加入报警模块,每收到一个数据就判断是否超过阈值温度。

E. 远端启用温度采集模块，将温度数据通过无线传输给本地端，先测试 1 路数据，再将 4 路数据循环发送。

F. 由于当远端发送多路数据会导致本地端接收数据频率降低，我们还需要在此基础上调整本地端接收函数的延迟，来提升整体的性能，是接收频率尽可能降低。

G. 在实验箱上整体测试成功后，将程序整体移植到我们自己制作的电路板上，再进行整机的调试。

在测试的时候我们保留了一个本地端和发送端的测试程序。这个程序实现起来非常简单，能够让我们确定当程序不能正确执行时到底是硬件的问题还是软件设计的问题。同时，在测试的时候我们每个阶段成功以后都会将程序做一个备份，这样当进行到后续步骤出错时便于我们及时重新开始。

7.7 遇到的困难

在联调过程中主要的困难是无线模块和整机联调。

在无线模块调试过程中，我们的实现方案有了一次彻底的变化。由全双工模式变为了单工模式。这是由于在全双工模式，我们虽然在远端接收到了控制信息（即指明需要哪一路信号的标志，为一个字节，分别用 0x01,0x02,0x04,0x08 表示），但是这个时候数据的发送就出现问题，而且这个问题是我们不能理解和解决的（具体问题描述请见个人的报告）。同时，工作在全双工模式下，由于没有 PCB 布板，用导线连接无线模块会导致信号干扰非常大，接收信号的频率非常低。故最终我们由双工改为单攻，由本地端来选择需要接收哪一路数据。

在整机联调的过程中，我们发现 LCD 显示函数（updata（））在单独调试的时候没有出

现任何问题，但是嵌入到键盘模块中以后就出现无法保留当前键值的情况。我们 debug 进入 updata（）以后，该显示程序执行完毕后 IAR 即显示 target execution stopped。但是如果我们把 updata() 函数的内容直接复制到主函数当中，即能正确执行。为什么会出现这个问题我们还没有找到原因。

8 最终成果

最终我们能够采集 3 路的温度，并在本地端 LCD 上显示出当前温度和阈值温度。

在空旷地无限传输范围为 200M 左右，在实验室内，在东 303 教室能够收到东 307 拐角处的数据。

成果展示如下图所示：

9.硬件课程设计总结

因为不想为大学生活留下遗憾，所以我们四个人选择了挑战自己，参加电信系的“TI 杯电子设计大赛”。我们不是为了拿名次而参赛，我们的初衷很简单，实现自己的价值，证明自己的能力，让大学前 3 年学到的东西能够真正发挥用途。

现在看来，我们的初衷实现了，虽然路途很艰辛。

我们四个人没有电工电子基地的经验，有的只是模电数电的实验课的经历。这一次课设让我们体会到基础知识的重要性。输出电压不够了，要接三极管放大；信号有干扰了，要注意走线的合理性和使用电容滤除噪声；程序不能正确执行了，用 debug 一步步执行，检查寄存器的值和本地值是否正确；硬件不行，用万用表测量各个管脚对应的点评是否符合要求。

在课设前期，我们像无头苍蝇一样乱撞，花了些冤枉钱买了贴片器件。结果是在因为时间不够无法做 PCB 布板，这些芯片业没有派上用场。这两个星期内我们跑广埠屯的次数比一个学期去的还多，但是也因此认识了很多热心的老板，协助我们解决了很多问题。

两周的时间内，我们的心情经历过大起大落。程序调不出来，我们焦虑，彷徨；终于出成果了，我们欢呼，雀跃，一起相约下馆子奖励一下自己。我们不仅学到了知识，更在一起努力的过程中结下了深厚的友谊。

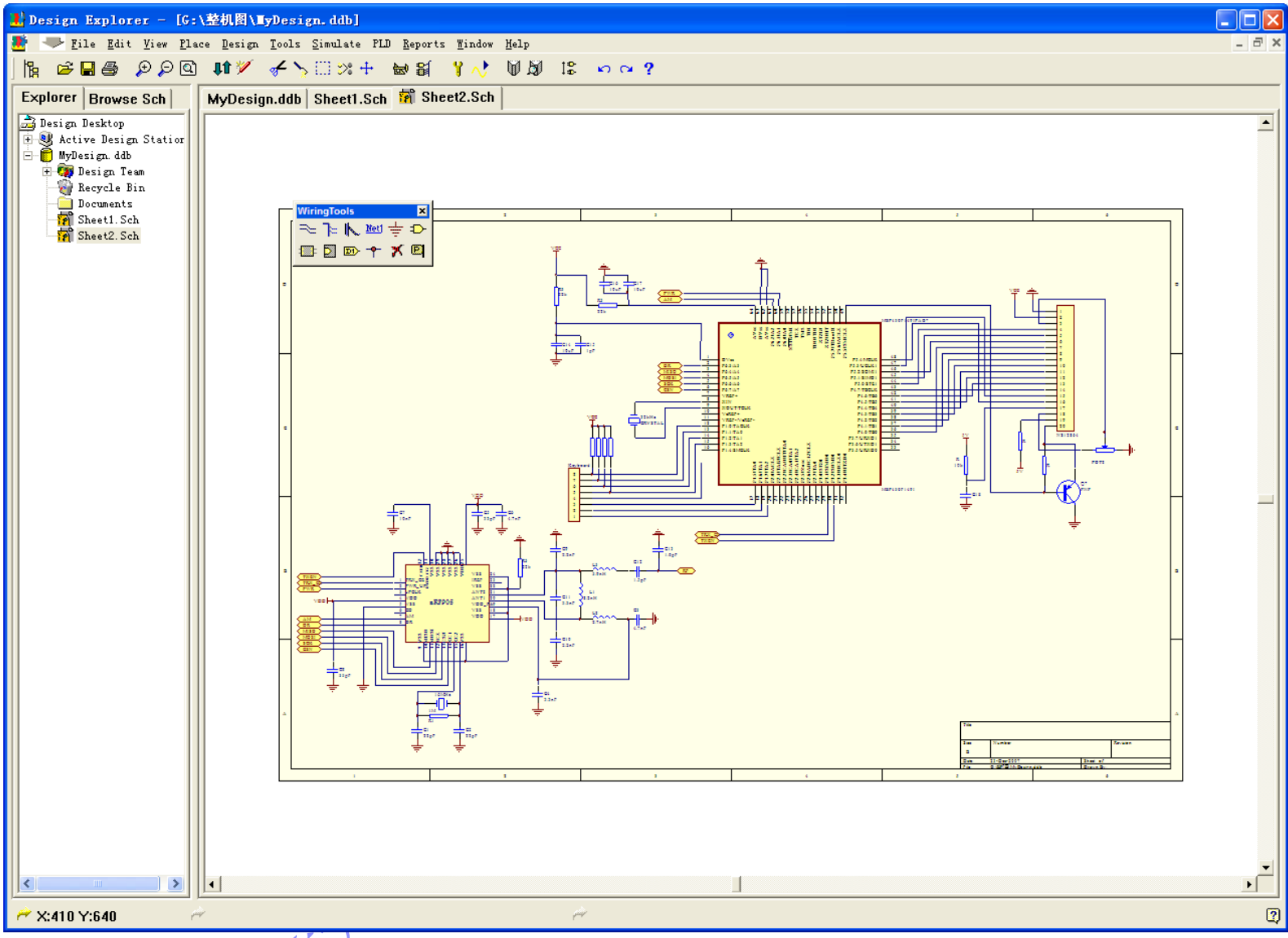
虽然过程很艰辛，但是能够为自己喜欢的事情废寝忘食，这也是一种幸福！

10 参考文献

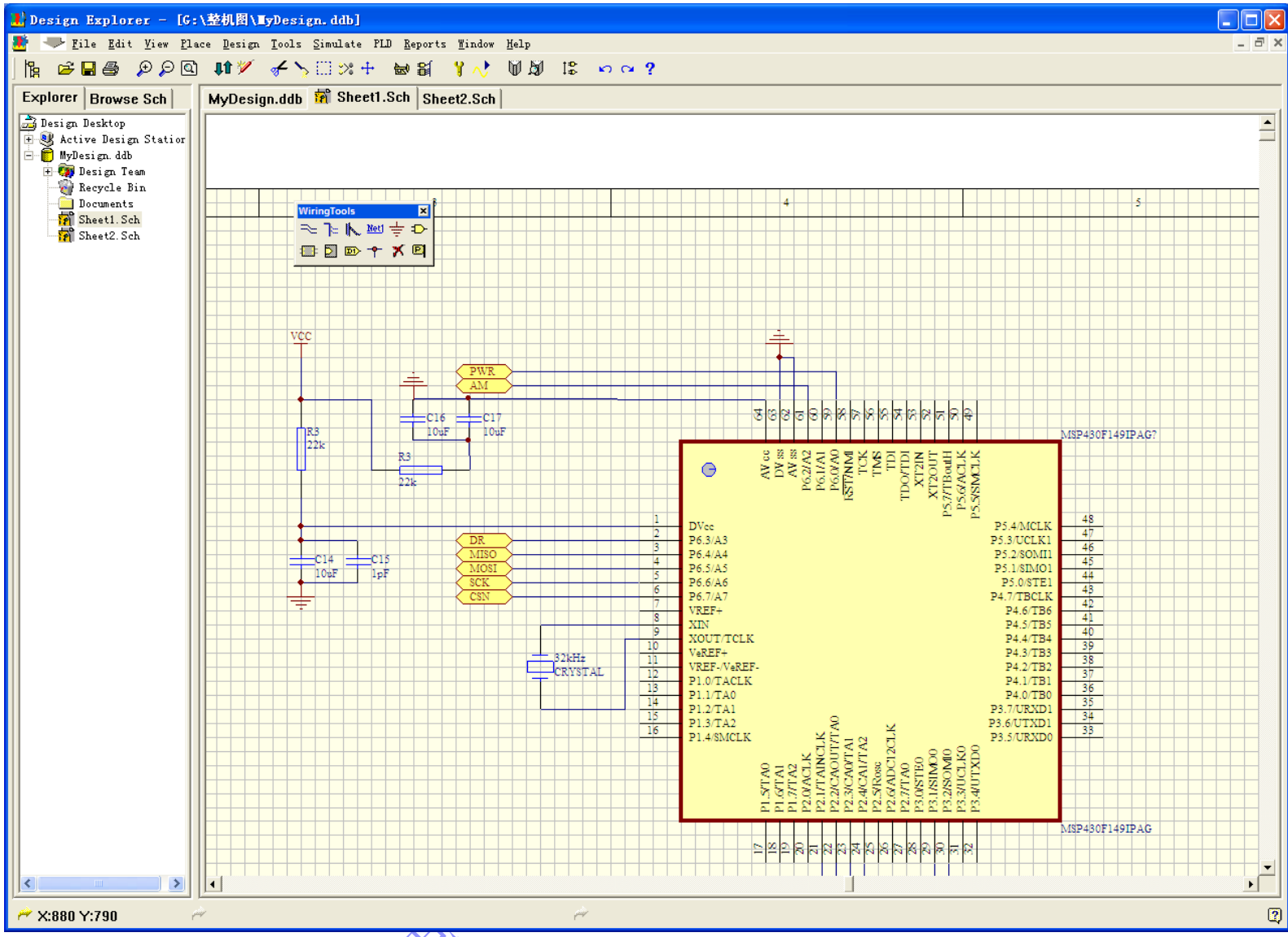
《MSP430 单片机自学笔记》	**张福才**	**北京航空航天大学出版社**
《MSP430 单片机原理与应用实例》	**洪利，章扬等**	**北京航空航天大学出版社**

附录（一）　整机电路图

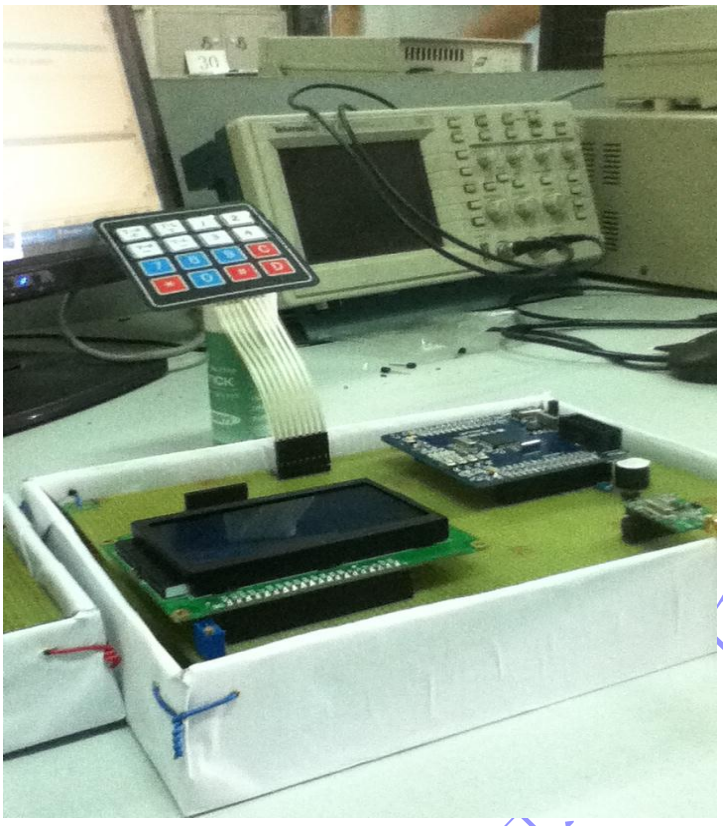
本地端



远方测试端



附录（二） 设计成品图



附录（三）所用器件

MSP430F149 核心板----------2
蜂鸣器----------1
NRF905----------2
4*4 矩阵键盘----------1
DS18B20----------4
万用板----------2