题目：LCD 显示的数字电子秤设计

班级： 华中科技大学控制系测控 0902 班

成员： 余良驹 学号： U200914537
黄小虎 学号： U200914531
杨 玉 学号： U200914534

指导老师 张林 罗杰

日期：2012-5-26

题目：LCD 显示的数字电子称

摘要

该设计是以 ARM Cortex-M3 内核的 STM32F103RBT6 的开发板为控制核心，实现电子称的基本控制功能。在设计系统时采用模块化的设计方法，分别设计各个单元功能模块。系统的硬件部分主要包括最小系统、数据采集、人机交互以及系统电源四个部分。最小系统主要为 STM32F103RBT6 处理器。数据采集部分主要由压力传感器、前级信号滤波处理以及 A/D 转换三个部分组成，信号的放大及 A/D 转换主要是基于 ADI 公司生产的 24 位高精度 A/D 转换器 AD7799。人机交互界面利用 2.4 寸的 TFT 的彩色触摸显示屏实现，触摸显示屏采用基于 uC/OS-II 和 uC/GUI 的界面设计，有功能齐全的菜单块、数字小键盘以及称重、单价及总价的显示区域。

关键词：压力传感器 AD7799 STM32F103RBT6 处理器 基于 uC/OS-II 和 uC/GUI 的界面设计

TOPIC ： Digital Electronic Scale with LCD Display

Abstract

Our design is based on the ARM Cortex-M3 core of STM32F103RBT6 development board said that the basic control functions for the control of the core electronic. Modular design approaches to design a system were designed for each unit function modules. The hardware part of the system including the smallest systems, data acquisition, human-computer interaction and system power four parts. The minimum system consists of STM32F103RBT6 processor. The data acquisition part is composed of three parts namely pressure sensors, the main pre-signal filtering by the pressure sensors, as well as A / D converter. Signal amplification and A / D converter is based mainly on the production of ADI's 24 high-precision A / D Converter AD7799.The interactive interface is based on a 2.4-inch TFT color touch screen. The touch-screen display with GUI design, full-featured menu, numeric keypad, as well as weighing, unit price and total price of the display area

Keyword: Pressure sensor、AD7799、STM32F103RBT6 processor、LCD display based on uC/OS-II 和 uC/GUI

目　录

一、系统设计目标

1.1 设计内容与要求

设计设计一个高精度的数字电子秤，要求用 LCD 显示。其组成框图如下：

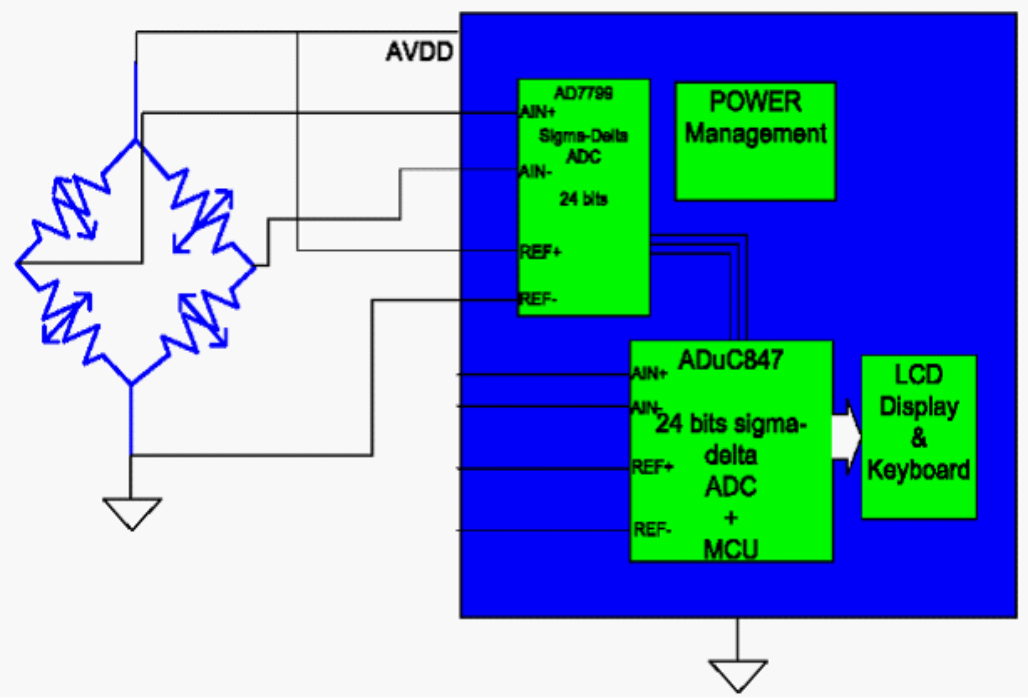


图 1 LCD 显示的数字电子称组成框图

1.2 基本设计目标

- 额定的最大称重量：3.0 kg；
- 精度：能分辨 1g 物体的重量；
- LCD 能够分别显示物体的重量和 ADC 芯片转换后的数字代码；

1.3 发挥与拓展

- 提高精度到 0.5g；
- 设置单价
- 计算并显示总价
- 总价清零
- 去皮功能

二、系统方案设计与论证

2.1 设计方案一

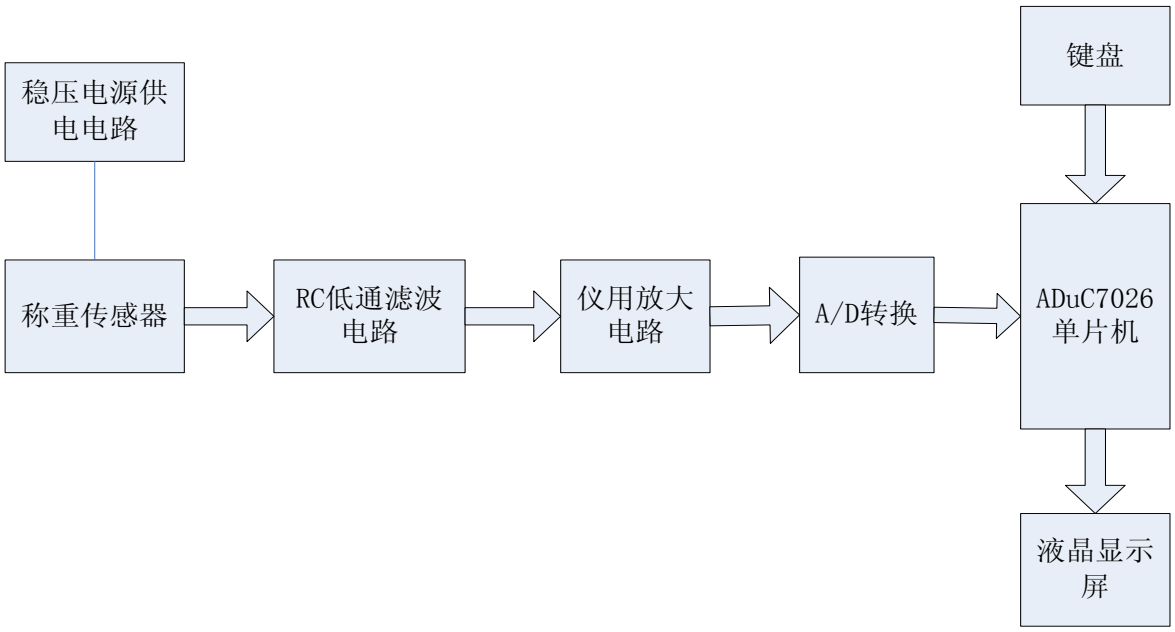


图 2 方案一框图

方案一论述：

电源部分：采用价格便宜、应用广泛地 7815 和 7915 稳压芯片，为仪用放大电路提供+15V 和-15V 的双电源供电，称重传感器可采用+15V 供电

称重传感器：采用上海华宝传感器厂生产、型号为 96J、规格为 20K、等级为 002 的压力传感器。

RC 低通滤波电路：

$U(s)=\frac{1}{1+sRC}$,截止频率为$f_s=\frac{1}{2\pi RC}$,通过改变R和C的值来调整截止频率。由于本设计中采用的是直流放大，因而R和C的值适当取得大一些。

仪用放大电路：

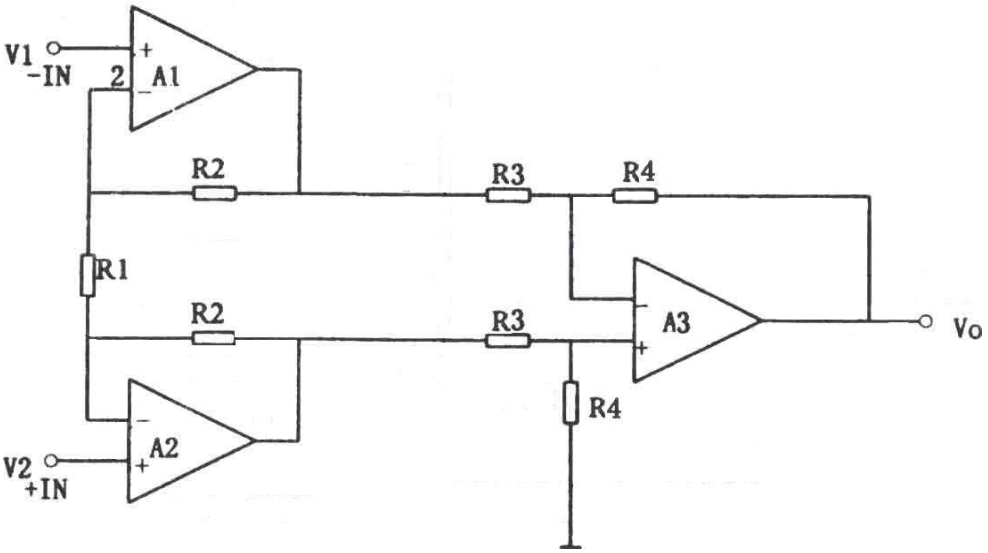


图 3 仪用放大电路图

以上仪用放大电路的增益为：$A_V = \frac{R_4}{R_3} \cdot \frac{R_1 + 2R_2}{R_1}$，一般保持$R_3 = R_{4,} R_3 = R_4$，通过匹配$R_1$和$R_2$达到所需的放大倍数。

由于称重传感器输出的是差压信号，因而采用一般的单端输入的集成运算放大电路不能满足需求，因而需要采用双端输入的仪用放大电路，这种比较成熟的放大电路可以通过匹配电阻值达到所需的放大倍数。

A/D 转换：采用 ADI 公司的 24 位的高精度 AD 实现。

单片机：采用 ADI 的 ADuC7026

液晶显示屏：采用市面上比较常见的 128*64 的 12864 点阵液晶，已经有比较完善的驱动程序

HS12864-15 系列液晶硬件特性：

①64x16 位字符显示 RAM(DDRAM 最多 16 字符 x4 行,LCD 显示范围 16x2 行)；

②2M 位中文字型 ROM，总共提供 8192 个中文字型；

③16K 位半宽字型 ROM,总共提供 128 个西文字型。

本设计中，需要在液晶上显示 A/D 转换值、称重值、用户设置的单价以及总价，因而采用 16 字符 x4 行的 12864 点阵液晶可以满足需求。另外 12864 点阵液晶在市面上已经比较广泛地被运用，有比较完善的驱动程序可作为学习的模板。

键盘：采用电子称集成薄膜键盘，自带 0~9 数字小键盘，并有单价重置、置零、清除以及去皮等功能按键，是专为电子称设计的一种按键，能够满足本设计中的功能需求。

2．2 设计方案二

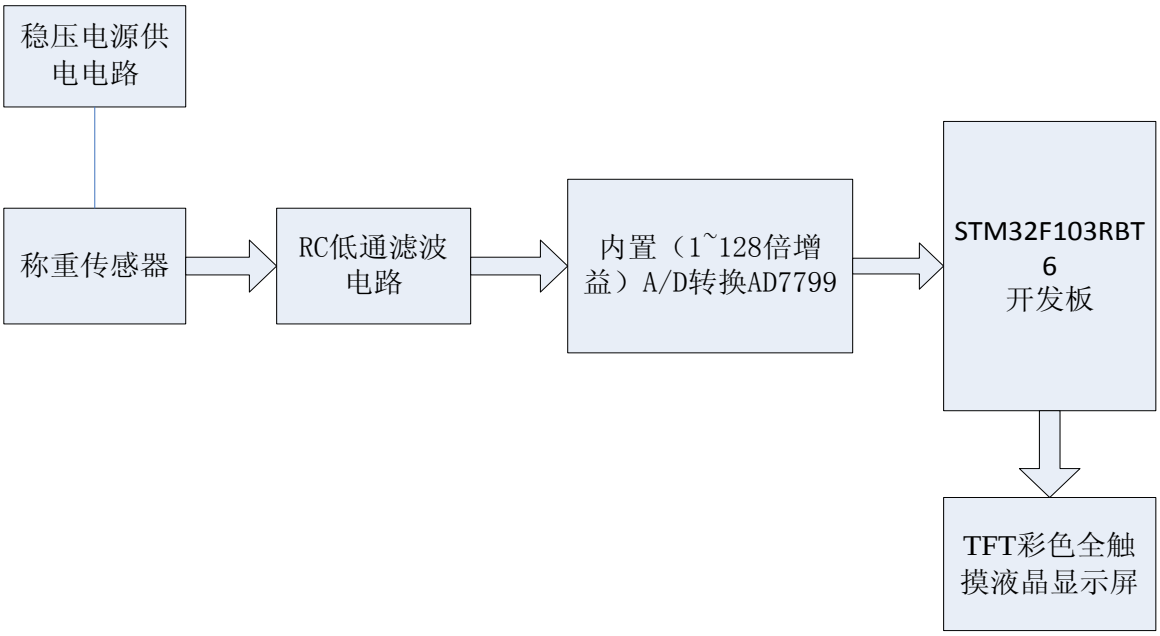


图 4 方案二框图

方案二论述：

方案二中电源部分、称重传感器部分以及 RC 低通滤波电路同方案一，区别为以下几个方面：

① 采用不设计外部信号放大电路而采用 AD7799 内置的增益可调（1~128 倍）的放大电路。

②采用比较高级的全触摸彩色显示屏，基于 GUI 的设计省去了外部接入的键盘，一来减小了外部接口，使系统的稳定性变好，二来采用全触屏设计操作方便可以达到友好的交互界面。

2.3 方案比较及选择

方案一和方案二比较：

方案一采用的是比较传统、比较保守的设计，由于采用外部设计放大电路因而可以根据需要设计参数达到所需的放大倍数。另外这种设计方案降低了对 A/D 的要求，选用通用的 AD 即可。但不可回避的是这种分散性的设计在性能方面可能远

远不及 AD7799 内部集成的增益可调放大器，为了使电路具有较好的性能，需要做一些额外的抗噪声、抑制零漂等设计。但是考虑到称重传感器输出的信号微弱，AD7799 内置的最高增益为 128 倍的集成仪用放大电路可能不能满足需求，因而外部的仪用放大电路仍然不能摒弃。方案二是在我们经过深思熟虑之后讨论出的比较大胆、比较创新的设计。其亮点在于采用了全触摸彩色的显示屏，基于 GUI 的图形化设计省去了外接键盘的麻烦并且彩色的显示界面可以带给用户更好的视觉享受。这种方案是基于全触摸式电子产品不断普及的考虑，按键式的输入不断被市场淘汰，而取而代之的是触摸的显示屏。

另外，考虑到重力传感器的输出信号较小，AD7799 内置的增益可调放大器可能达不到所需的放大倍数，我们最后做了一个折中的方案选择，得到了如下的设计方案：

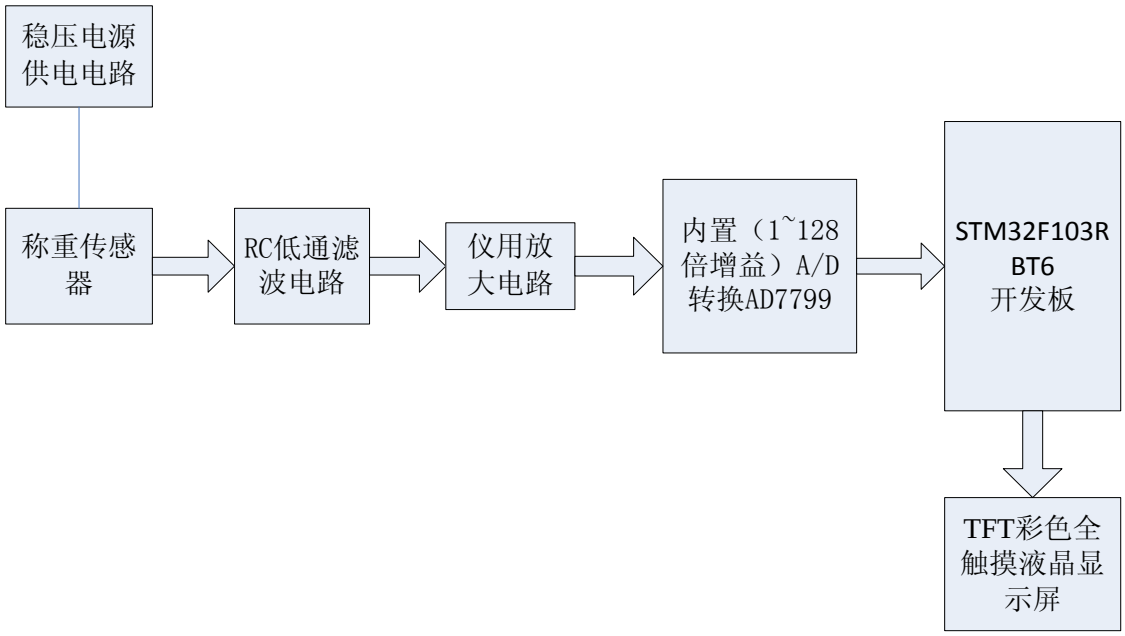


图 5 最终确定的设计方案

最终确定的方案可以说是结合了方案一和方案二的优点，外加的放大电路解决了 AD7799 内置的增益可调放大的倍数可能不足的问题，同时也充分利用了 AD7799 这样一款专用于电子称设计的 A/D 转换芯片的优点，采用软硬件相结合的同时放大方式可以使设计留有充足的余量。当然，以上只是经过初步的分析敲定的方案，最终的设计方案可能会在此基础上做出适当的修正。

三、理论分析计算

3.1 A/D 转换芯片位数：

近似计算过程如下：

指标要求额定的最大称重为 3.0kg，考虑到留有一定余量从而避免电子称由于超过最大称重范围而损坏，设计最大称重为 5kg，则设计 5kg 对应 AD 的参考电压 REF+（假定设计 REF-接地）。因而若要达到 1g 的分辨率，则 A/D 的位数 n 应满足 $\frac{1}{2^n} \le \frac{1}{4000} \Rightarrow n \ge 12$，即若要达到 1g 的精度，A/D 的位数至少应为 12 位。若要继续提高精度到 0.5g，则依据同样的计算方法 A/D 的位数至少要 13 位。

3.2 外部放大电路所需增益

外部放大电路增益需根据所提供的称重传感器输出的信号大小以及所选的 A/D 参考电压来确定。在本设计中初步选择的 A/D 的参考电压为 1.25v。利用面包板搭建增益可调的放大电路测得当采用 Av=130 倍的放大电路时，150 的称重输出电压信号为 21.5mV。考虑到选用 1.25V 的参考电压，设计最大称重为 5kg，则 150g 的称重对应的输出信号应达 $\frac{150}{5000}\times 1.25=0.075V=37.5mV$，则应该修改放大倍数到 $\frac{37.5}{21.5}\times 130 \approx 227$。实际设计电路中考虑到参数匹配的方便性，可在这附近取值。

3.3 RC 低通滤波电路电阻、电容参数

RC低通滤波电路的截止频率为$f_s = \frac{1}{2\pi RC}$,取$R = 1k\Omega\ C = \mu F$

截止频率为159. 2HZ。能够满足直流放大的要求。

四、原理说明

4.1 24 位高精度 A/D 转换 AD7799

ADI 公司的 24 位高精度、低噪声 A/D 装换芯片 AD7799 是一款专用于电子磅秤、压力测量、应变传感器、血气分析、工业过程控制仪器仪表、便携式仪表等领域的专用芯片。

4.1.1 特性参数如下：

●RMS 噪声（有效值噪声）：
在 4.13HZ 转换率下紧为 27nV(AD7799).
在 16.7HZ 转换率下为 65Nv.
● 低功 耗 ，典型为 300uA
● 内置 1~128 倍增益的低噪声可编程仪表放大器;
● 内置时钟振荡器，省去了外接晶振;
● 低非线性度:0.0015%;
● 内设自校准电路;
● 带有 SPI 数据接口，可以方便地与 DPS 或者 MCU 连接;
● 50 H z 和 60Hz 同步陷波，消除 50Hz 和 60Hz 工频干扰;
● 可配置 3 个差分输人通道;
● 21 位 有效分辨率(PGA=1)，19BIST (PGA=128
工作电源范围 2.7V----5.5V，-40 摄氏度 至 105 摄氏度的温度范围，独立供电接口；
●16 脚的 TSSOP 封装。

各管脚及功能

管脚号	助记符	管脚功能及描述
1	SCLK	串行时钟输入,这是数据传输和 ADC 的串行时钟输入。该时钟的施密特触发输入，使界面适合光电隔离的应用。
2	CS	片选输入，对 AD7799 的操作在 CS 低电平时有效
3	AIN3+/P1	模拟输入/数字输出。另外该引脚还可以作为一般用途上的输出参照位
4	AIN3-/P2	同上
5	AIN1+	模拟输入
6	AIN1-	模拟输入
7	AIN2+	模拟输入
8	AIN2-	模拟输入
9	REFIN+	同相参考输入。外部参考电压可应用在这两脚之间，参考值是 2.5V，但对于不同的应用可以在 0.1V---AVDD 之间
10	REFIN-	反向参考输入。参考值在 GND 到 AVDD-0.1V 之间
11	PSW	
12	GND	
13	AVDD	电源 2.7V-5.25V
14	DVDD	数字电源
15	DOUT/RDY	串口数据输出端/数据准备输出。低电平表明数据转换

		完成。DOUT/RDY 的下降沿可作为一个中断处理，表明有效的数据已就绪。随着外部时钟的变化，可以使用这个脚来读取数据。当 CS 低时，数据/控制字信息在 SCLK 的下降沿和有效的上升沿上，
16	DIN	串行数据输入到 ADC 的输入移位寄存器。

4.1.2 AD7799 内部结构及工作原理

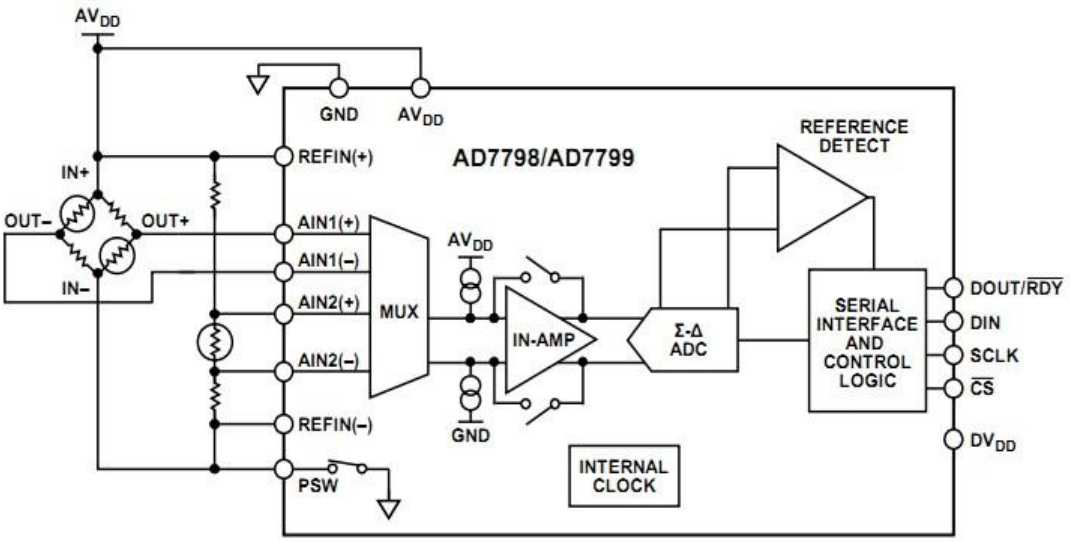


图 6 AD7799 内部结构图

图 7 所示为 AD7799 的内部结构。AD779 内部主要由模拟多路开关(MUX)、输入缓冲器(BUF)、可编程增益放大器(PGA)、一体化△-Σ调制器、可编程数字滤波器、9 个状态/控制寄存器、串行 SPI 接口以及时钟发生器等组成。

其中，输入多路选择器(MUX)主要用来提供 3 路模拟输人差分组合。而输人缓冲器(BUF)用于在信号通路中隔离开关电容器阵列与外部电路。当没有使用输人缓冲器时，AD779 的输人阻抗为 5M 欧，当使用 AD779 内部缓冲器时，其输人电压的波动减小，输人电流增大。其内部输人缓冲器是通过内部 CON 寄存器的 BUF 位控制的。

AD779 内部的可编程增益放大器(PGA)的放大倍数可以通过 CON 寄存器的 G0 - G2 位设定为 1 到 128 倍，增益步长为 2。

通过数字滤波器可提高 ADC 的转换精度和分辨率。数字滤波有一定的建立时间。AD7799 建立时间 由模式寄存器(MR)的 FS0 - FS3 位设置的，建立时间越短时滤波精度越低，而建立时间越长时滤波精度越高。

AD7799 采用外部参考电压，外部参考电压是差动输人，输人范围为 0 - AVDD。

AD7799 采用四线制(时钟信号线 SCLK、数据输入线 DIN、数据输出线 DOUT 以及片选线/CS)SPI 通讯方式。SPI 的最大通信时钟可达 5MHZ。AD7799 只能工作在 SPI 通讯的从模式下，可通过各种主控制器(如单片机等)给它发送同步传送命

令。在 SPI 传送过程中，数据被同步地发送和接收，SCLK 和 DIN、 DOUT 同步移动。图 2 所示是 SPI 通讯时序关系。

AD779 采用 9 个寄存器来直接控制 AD7799 的工作过程，而且其中有 7 个寄存器可以被直接读写。实际上，这些寄存器可以用来配置 AD7799 的所有参数，比如数据格式、通道选择、增益设置等，整个器件工作过程的建立可通过对 7 个独立的寄存器的设置来完成。 通过写模式寄存器可以使 AD7799 工作在连续转换模式、单次转换模式、空闲模式、省电模式、片内零度校准（输入端内部接 AGND） 、片内满度校准（输入端内部接 VREF） 、系统零度校准、系统满度校准、转换速度。当设计采用 50HZ 的转换速度时，其 RMS 可介于 21 BITS（33.3HZ）和 20.5 BITS（62.5HZ）之间。通过写配置寄存器还可以选择测量通道、该通道是双极性还是单极性、该通道是否开通 100NA 电流源、该通道的放大倍数、是否开通参考电源检测功能。零度校准寄存器、满度校准寄存器是 24 位可读写寄存器，可以通过写模式寄存器启动片内零度校准，系统零度校准和片内满度校准，再启动满度校准填充数值，也可以把存在 ROM 中的人工校准数据写入。先启动零度校准，再启动满度校准可以提高 AD 转换的线形度。 可以通过写模式寄存器启动 AD 进行单次转换或者连续转换，转换结果存在数据寄存器中，转换完毕 DOUT/RDY 变为低电平，而读完 24 位数据 DOUT/RDY 变为高电平。如果采用连续转换方式就不用每次读完数据都重新启动 AD。如果采用单次转换方式每次读完数据都必须重新写模式寄存器启动 AD，而且转换速度是设定速度的 1/3，这是因为 AD 内部的数字滤波器需要消耗 2 倍的转换时间的启动时间。要想从连续转换方式退出，可以在依次转换结束向模式寄存器写入单次转换命令，也可以通过向 DIN 输出 32 位高电平，复位整个 AD 转换器。另外 AD7799 还提供 64K 片内晶振（不需要外接晶振） 、2 路数字量输出、模拟电源电压测量、参考电源检测等、100NA 传感器监测恒流源等功能。

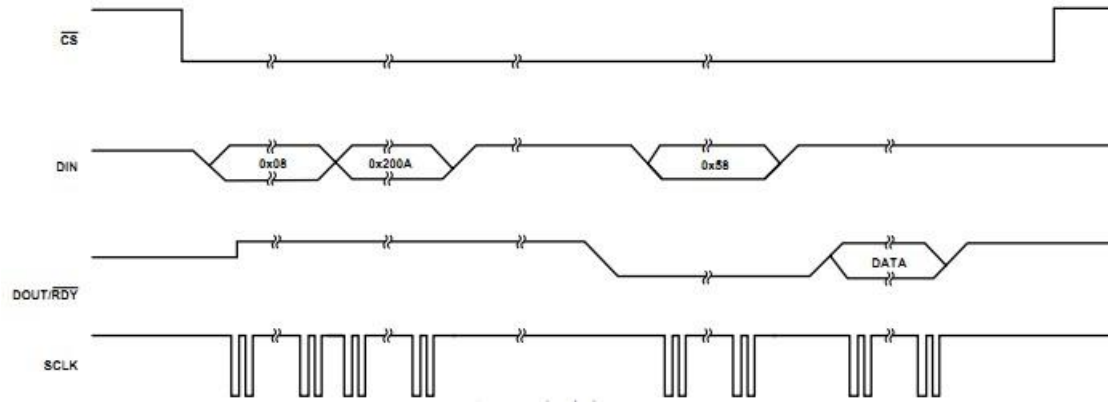


图 7 SPI 时序图

4.2 STM32F103RBT6 的开发板

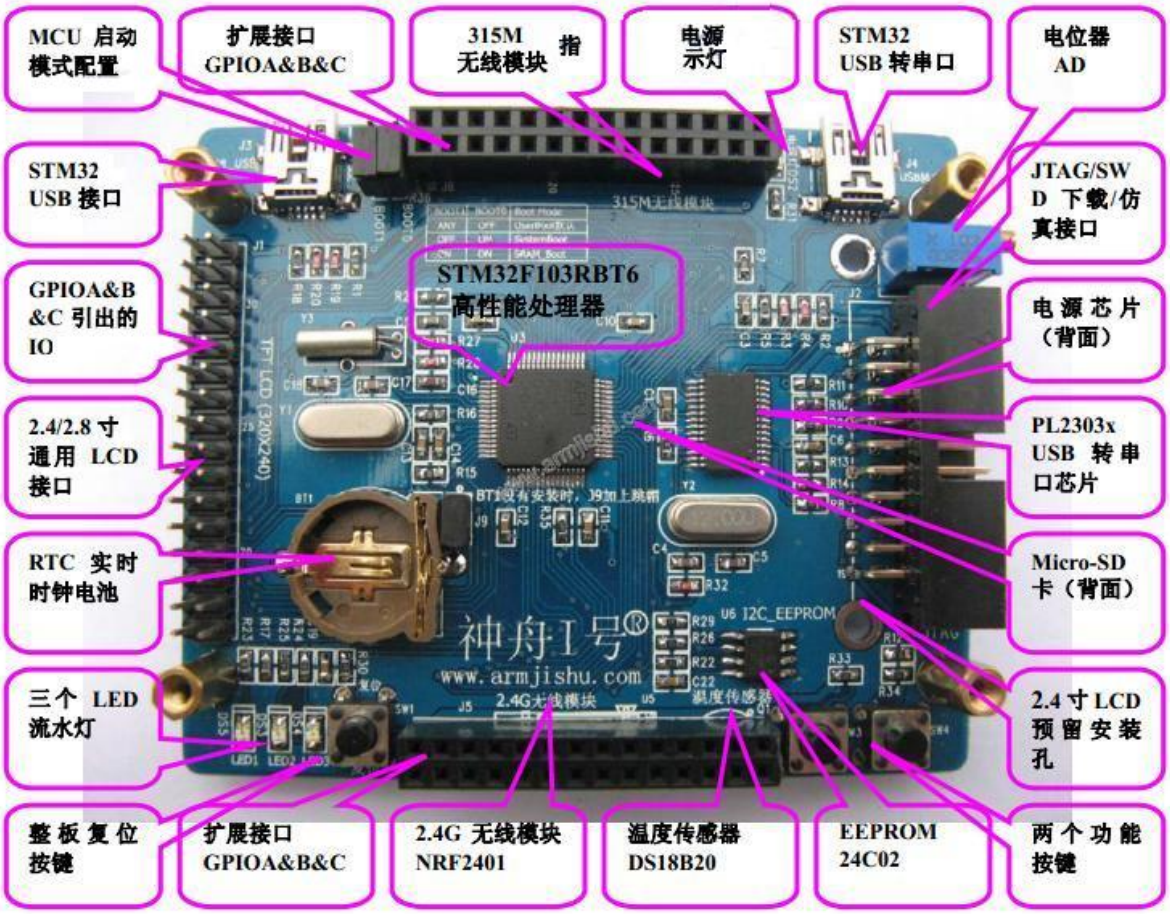


图 8 神州 I 号 STM32 开发板及各功能模块

神舟 I 号开发板板载资源如下：

- ■STM32F103RBT6,ARM Cortex-M3 内核，ARM Cortex-M3 内核，主频 72Mhz，内部含有 128K 字节的 FLASH 和 20K 字节的 SRAM
- ■ 1 个 USB _MINI 全速接口（包括+5V 电源的输入）
- ■ 1 个标准的 2.4 / 2.8 寸 TFT LCD 接口，支持触摸屏，分辨率 320X240, 26 万色（包括 1 个 SD 卡接口）
- ■ 1 个 Micro SD 卡座
- ■ 1 个 2.4G 无线通信模块接口
- ■ 1 个 315M 无线通信模块接口
- ■ 1 个温度传感器接口
- ■ 1 个 IIC 接口的 EEPROM 芯片，24C02，容量 2K 比特
- ■ 1 个 USB 转串口接口，PL2303x 芯片（包括+5V 电源的输入）
- ■ 1 个复位按钮，控制整板硬件复位
- ■ 2 个用户功能按钮
- ■ 1 个电源指示灯（绿色）
- ■ 3 个用户状态指示灯（LED1~LED3：绿色）

■ 2 个启动模式选择配置接口
■ 1 个 RTC 后备电池座，并带电池
■ 1 个标准的 JTAG/SWD 仿真调试下载接口
■ 1 路电位器（可调电阻）模拟输入，可以做模数转换实验
■ 支持从 JLINK 取 5V 电源或 3.3V 电源
■ 除晶振占用的 IO 管脚外，其余大部分 IO 口全部引出到扩展双排插针

以下为神州 I 号开发板的各个功能模块：

1、STM32F103RBT6 处理器

开发板使用了 STMF103 系列中的高性能、高配置的 Cortex-M3 内核 32 位处理器 STM32F103RBT6，72M 主频，LQFP64 封装，片内 FLASH 容量为 128K,片内 SRAM 容量达 20K。神舟 I 号开发板选用的是外设资源和管脚资源较为丰富的 64 脚 LQFP 封装的 STM32F103RBT6 芯片，具有丰富的硬件资源。

2、2.4/2.8 寸 LCD 接口

该接口是一个目前比较通用的 LCD 液晶触摸屏接口，一个 32 芯 LCD 接口引出了 LCD 控制器和触摸屏的全部信号，它的线序兼容市面上在售的主流触摸屏模块，比如 ARMJISHU.COM 推出的液晶模块、红牛开发板使用的液晶模块、STMSKY 开发板使用的液晶模块等。320 x 240 的显示分辨率 64 万色可以逼真的显示图片、文字和菜单等，配合触摸功能实现灵活的控制，我们提供已经调 试成功的 LCD 液晶屏和触摸屏的示例代码。该接口是一个目前比较通用的 LCD 接口，可以用来连接很多市面上在售的液晶模块，比如 ARMJISHU.COM 推出液晶模块。

3、 USB 接口

标准的 USB SLAVE 接口，其他主设备，如 PC 电脑可通过此接口与神舟 I 号通讯，另外，该接口也可以作为神舟 I 号的电源输入，由 PC 或其他 USB 主设备提供电源。

4、扩展接口

神舟 I 号将所有的 GPIO 的使用标准双排插针引出，方便大家的实验和测试，调试其他模块或功能扩展。

5、 USB 接口与电源

神舟 I 号 STM32 开发板支持的供电方式主要有三种，分别是：

a) USB 接口供电，最大 500mA
b) USB 转串口接口供电
c) JLINK V8 供电，包括 5V 和 3.3V

板上的电源转换芯片将 USB 接口或是 USB 转串口接口输入的 5V 电源转换成 3.3V 的电源，作为处理器和相关外围电路的工作电源。

6、 液晶显示模块

神舟 I 号开发板载有目前比较通用 2.4/2.8 寸液晶触摸显示模块接口。其设计原理图如图 9 所示：

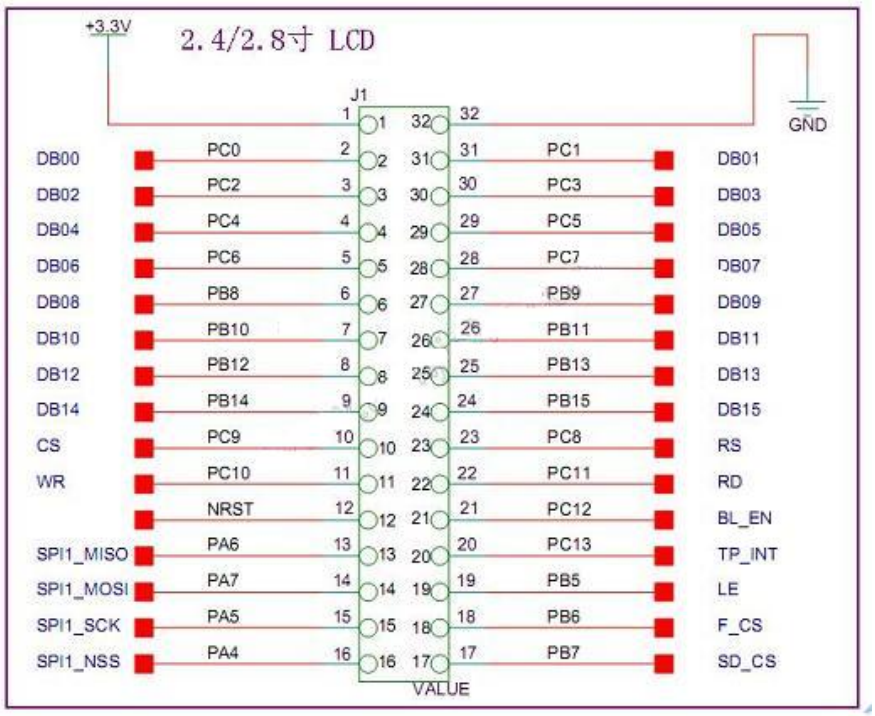


图 9 液晶显示模块原理图

神舟 I 号通过 FSMC 总线对屏进行访问操作，此外，2.4/2.8 寸 LCD 屏模块上还集成了 SPI FLASH 和 SPI 接口的 SD 卡座（神舟 I 号板载了一个 Micro SD 卡座）。

4.3 液晶屏，触摸屏的驱动

TFT 就是“Thin Film Transistor”的简称，一般代指薄膜液晶显示器，而实际上指的是薄膜晶体管（矩阵）—可以“主动的”对屏幕上的各个独立的像素进行控制。对与图像产生的基本原理为：显示屏由许多可以发出任意颜色的光线的像素组成，主要控制各个像素显示相应的颜色就可以达到目的。在 TFT LCD 中一般会采用背光技术，为了能精确的控制每一个像素的颜色和亮度就需要在每一个像素之后安装一个类似百叶窗的开关，当“百叶窗”打开时光线就可以透射过来，而“百叶窗”关上之后，光线就无法透射。

对于神舟 I 号开发板上配带的 TFT LCD 屏，采用 ILI9320 控制器，LCD 屏为 320x240 分辨率，16 位真彩显示。下面就 ILI9320 控制器进行简单的介绍。

ILI9320 控制器是一款带有 262144 种颜色的单芯片 SOC 驱动的晶体管显示器，320x240 的分辨率，包含 720 路源极驱动以及 320 路的栅极驱动，自带有显存，容量为 172800 字节。关于 ILI9320 的具体控制命令在此不作介绍。

在 TFT 彩屏显示的基础上，加上 TFT LCD 屏的触摸功能，将触摸采样到的数据在 LCD 屏上进行显示，主要借助 SPI 总线实现对触摸芯片 ADS7843 的控制。

触摸屏逐渐取代键盘成为通信常用的人机交互工具，手机支持触摸功能、PDA 手持设备等等的运用。

触摸屏一般分为电阻、电容、表面声波、红外线扫描和矢量压力传感器等，其中使用最多的是四线或无线电阻触摸屏。四线电阻触摸屏是由两个透明电阻膜构成的，在它的水平和垂直

电阻网上施加电压，就可以通过 A/D 转换面板在触摸点测量出电压，从而对应出坐标值。

神舟一号的触摸屏附在 LCD 屏的表面上，与 LCD 屏相配合使用，主要使用的触摸芯片是 ADS7843，业界上与 ADS7843 芯片兼容的触摸芯片还有 ADS7846、AK4182 等，驱动原理基本上一致。

ADS7843 是一款 4 线式触摸屏控制器，内含 12 位分辨率，125KHz 转换速率，逐渐逼近型的 A/D 转换器。

下面我们将通过 ADS7843 芯片讲解触摸原理。ADS7843 内部有一个由多个模拟开关组成的供电测量电路网络和 12 位的 A/D 转换器。其可以根据处理器（stm32f103RTB6 通过 SPI 总线）发来的不同测试命令导通不同的模拟开关，以便向工作面电极对提供电压，并把相应测量电极上的触点坐标位置所对应的电压模拟量引入到 A/D 转换器。在触摸点 X、Y 坐标的测试过程中，测试电压与测量点的等效电路图 10 所示：（P 为测量点）

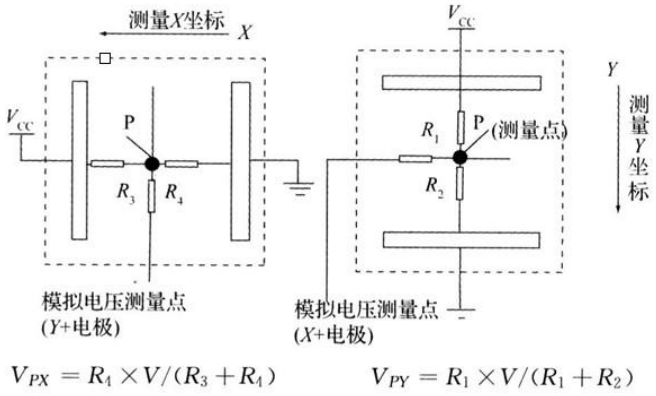


图 10 测量电压与测量点关系等效电路

当触摸屏受到点击或是挤压时，ADS7843 通过中断请求通知处理器（stm32f103RTB6）有触摸发生。当没有触摸时，ADS7843 内部的 MOSFET 开关状态正常，则中断输出引脚通过外加的上拉电阻输出为高，当有触摸时，MOSFET 开关状态改变，则中断输出引脚连接到地，输出为低，从而向处理器发出中断请求。

4.4 基于 uC/OS-II 操作系统 uC/GUI 的界面设计

主要包括 UC/OS-II 介绍、UCGUI 介绍、界面实现的各个功能介绍及实现原理。

4.4.1 uC/OS-II 介绍：

uC/OS-II(Micro Control Operation System Two)是一个可以基于 ROM 运行的、可裁减的、抢占式、实时多任务内核，具有高度可移植性，特别适合于微处理器和控制器，是和很多商业操作系统性能相当的实时操作系统(RTOS)。为了提供最好的移植性能，uC/OS-II 最大程度上使用 ANSI C 语言进行开发，并且已经移植到近 40 多种处理器体系上，涵盖了从 8 位到 64 位各种 CPU(包括 DSP)。 uC/OS-II 可以简单的视为一个多任务调度器，在这个任务调度器之上完善并添加了和多任务操作系统相关的系统服务，如信号量、邮箱等。其主要特点有公开源代码，代

码结构清晰、明了，注释详尽，组织有条理，可移植性好，可裁剪，可固化。内核属于抢占式，最多可以管理 60 个任务。从 1992 年开始，由于高度可靠性、鲁棒性和安全性，uC/OS-II 已经广泛使用在从照相机到航空电子产品的各种应用中。

uC/OS-II 可以大致分成核心、任务处理、时间处理、任务同步与通信，CPU 的移植等 5 个部分。 1) 核心部分(OSCore.c) 是操作系统的处理核心，包括操作系统初始化、操作系统运行、中断进出的前导、时钟节拍、任务调度、事件处理等多部分。能够维持系统基本工作的部分都在这里。 2) 任务处理部分(OSTask.c) 任务处理部分中的内容都是与任务的操作密切相关的。包括任务的建立、删除、挂起、恢复等等。因为 uC/OS-II 是以任务为基本单位调度的，所以这部分内容也相当重要。 3) 时钟部分(OSTime.c) uC/OS-II 中的最小时钟单位是 timetick（时钟节拍）。任务延时等操作是在这里完成的。 4) 任务同步和通信部分 为事件处理部分，包括信号量、邮箱、消息队列、事件标志等部分；主要用于任务间的互相联系和对临界资源的访问。 5) 与 CPU 的接口部分 是指 uC/OS-II 针对所使用的 CPU 的移植部分。由于 uC/OS-II 是一个通用性的操作系统，所以对于关键问题上的实现，还是需要根据具体 CPU 的具体内容和要求作相应的移植。这部分内容由于牵涉到 SP 等系统指针，所以通常用汇编语言编写。主要包括中断级任务切换的底层实现、任务级任务切换的底层实现、时钟节拍的产生和处理、中断的相关处理部分等内容。

4.4.2 uC/GUI 简介

uC/GUI 是 Micrium 公司研发的通用的嵌入式用户图像界面软件。他给任何使用图像 LCD 的应用程式提供单独于处理器和 LCD 控制器之外的有效的图形用户接口。能够应用于单一任务环境，也能够应用于多任务环境中。uC/GUI 能够应用于任何 LCD 控制器和 CPU 的任何尺寸的物理显示或模拟显示中。

uC/GUI 的特点如下：适用于任何 8 位/16 位/32 位 CPU，可允许于支持 ANSI C 的任何编译器 适用于任何控制器驱动任何 LCD（单色，灰度，或彩色） 通过配置宏，可支持任何接口 可配置显示尺寸 可在 LCD 的任何一点上显示字符和画位图 对于显示尺寸和速度提供优化进程，编译时间依赖于采用的优化进程 支持虚拟显示，虚拟显示的尺寸比实际显示大。

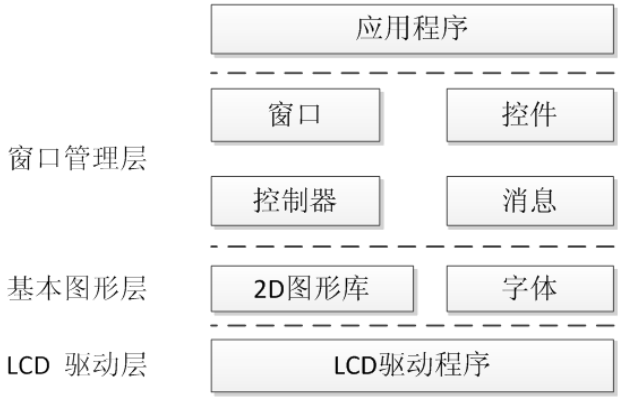


图 11 uC/GUI 的分层模结构

uC/GUI 的模块结构如图 11 所示，可以分为三层：LCD 驱动层、基本图形层和窗口管理层。

（1）LCD 驱动层

该层是 uC/GUI 的软硬件接口层，主要包含在 LCD.h 和 LCDConfig.h 中，包含了操作 LCD 硬件需要的所有宏定义和函数。该层和平台使用的硬件密切相关。

对于不同的 LCD 控制器，该层的内容可能有很大差别，总体来说该层包含的模块如下：

- 硬件设置：LCD 的尺寸、色深、寄存器名称、读写命令队列等。
- 颜色转化：uC/GUI 支持不同的色深，如 555、565、323、444 等。在这里提供了把各种不同色深图像转化为 24 位色的函数，并且用户可以自己设置调色板。
- 抗锯齿模块：在 LCDAA.c 中实现，通过对颜色进行混合达到抗锯齿效果。
- 驱动接口：包括打开/关闭 LCD、画点、画线、区域填充、BMP 位图显示等函数。

（2）基本图形层

该层提供了文本显示、数值显示、2D 图形显示的支持，构成了 uC/GUI 的基本核心层。该层函数高效精简，为资源有限的嵌入式系统提供基本服务。

用户可以调整该层函数来改变背景颜色、前景颜色；也可以改变当前字体，使用正常模式或者透明模式显示。

uC/GUI 支持 8 位 ASCII 码和 16 位 Unicode 码的字符，支持等宽和比例字体。用户可以根据需要加入任意个数的字体。

（3）窗口管理层

该层为高级用户提供窗口显示和管理，需要更多内存空间，时独立的可选层。系统初始化时自动创建一个覆盖整个 LCD 的桌面窗口，其他窗口都是桌面的子窗口。系统为每个窗口维护一个句柄，对句柄的操作就是对相应窗口的操作。窗口控制器用来实现窗口的控制管理，如窗口区域的剪切、消息的发送和处理。在控件方面，其支持编辑框、按钮、进度条、下拉列表等常用的控件。

五、单元电路设计与功能实现说明

5.1 仪用放大电路设计

由于称重传感器输出的信号很微弱，无法直接利用数字示波器测量出来，因而仪用放大电路增益的确定是一个不断尝试和摸索的过程，我首先确定的增益为 100 倍，利用面包板根据计算好的参数搭建电路，测量传感器输出的信号，发现放大倍数不够，后修改参数使增益达到 210 倍。

集成运算放大器的选型主要从以下几个方面考虑：

①输入噪声

②失调电压

③零漂（温漂）

几种常用的集成运算放大器参数对比

芯片型号	uA741（单运放）	OP07（单运放）	NE5532(双运放)
输入失调电压	1.0mV	250uV	0.5mV
输入失调电流	20nA	8nA	10nA
输入偏置电流	80nA	9nA	200nA
失调漂移		0.5μV/℃	
说明	通用型	低噪声	低噪声

选型：从输入噪声、失调电压以及零漂等三个方面考虑，集成运放 OP07 无疑是最佳的选择。OP07 集成运放是一种低噪声，非斩波稳零的双极性运算放大器集成电路。由于 OP07 具有非常低的输入失调电压（对于 OP07A 最大为 25μV），所以 OP07 在很多应用场合不需要额外的调零措施。OP07 同时具有输入偏置电流低（OP07A 为±2nA）和开环增益高（对于 OP07A 为 300V/mV）的特点，这种低失调、高开环增益的特性使得 OP07 特别适用于高增益的测量设备和放 大传感器的微弱信号等方面。

采用 OP07 集成运放以及电阻、电容等分立性元件组成如下的仪用放大电路放大倍数为 210 倍。

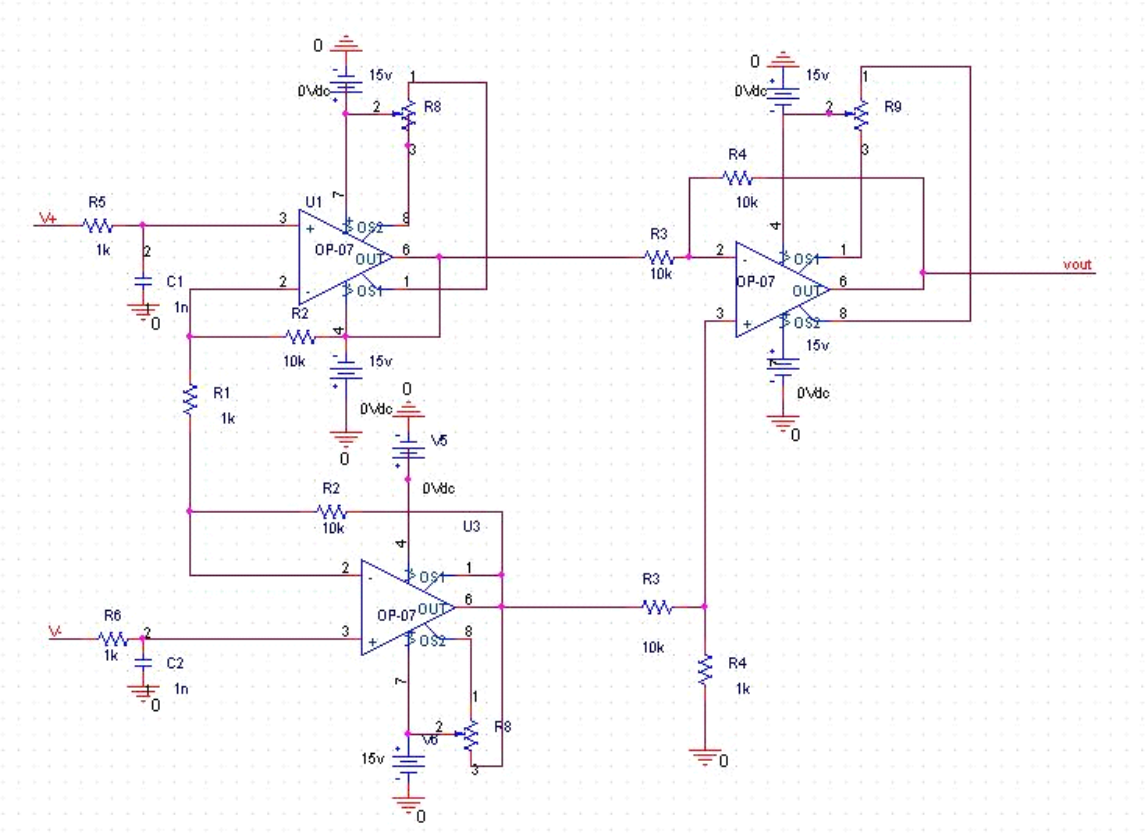


图 12 仪用放大电路

5.2 AD 模块

5.2.1 AD 硬件连线

将 AD 的四位 SPI 总线的片选线/CS、时钟信号线 SCLK、数据输入线 DIN 以及数据输出线 DOUT 分别接到单片机的 PB1、PB0、PA1、PA0 引脚，程序中宏定义如图 13 所示：

```
#define AD_cs    GPIO_Pin_1                    //PB1  作为 AD_cs 连接脚
#define AD_sclk    GPIO_Pin_0                  //PB0 作为 AD_sclk 连接脚
#define AD_DIN    GPIO_Pin_1                   //PA1 作为 AD_DIN 连接脚
#define AD_DOUT    GPIO_Pin_0                  //PA0 作为 AD_DOUT 连接脚
```

其中 REF 为 0.3V：因为依照原设计将 REF 设置为 3.3V 时，AD 转换后数据为 16 位，但是由于电源影响，数据后 12 位一直在跳动，因此不准确的位数相对较多，使得换算后得到的重量不稳定，误差大概在 5g 左右。将 REF 设置为 0.3V 后，AD 转换后数据为 21 位，由于电源影响，数据后 12 位仍然一直在跳动，但此时不准确的位数相对较少，使得换算后得到的重量只有最后一位在跳动，误差减小到 0.5g 左右。

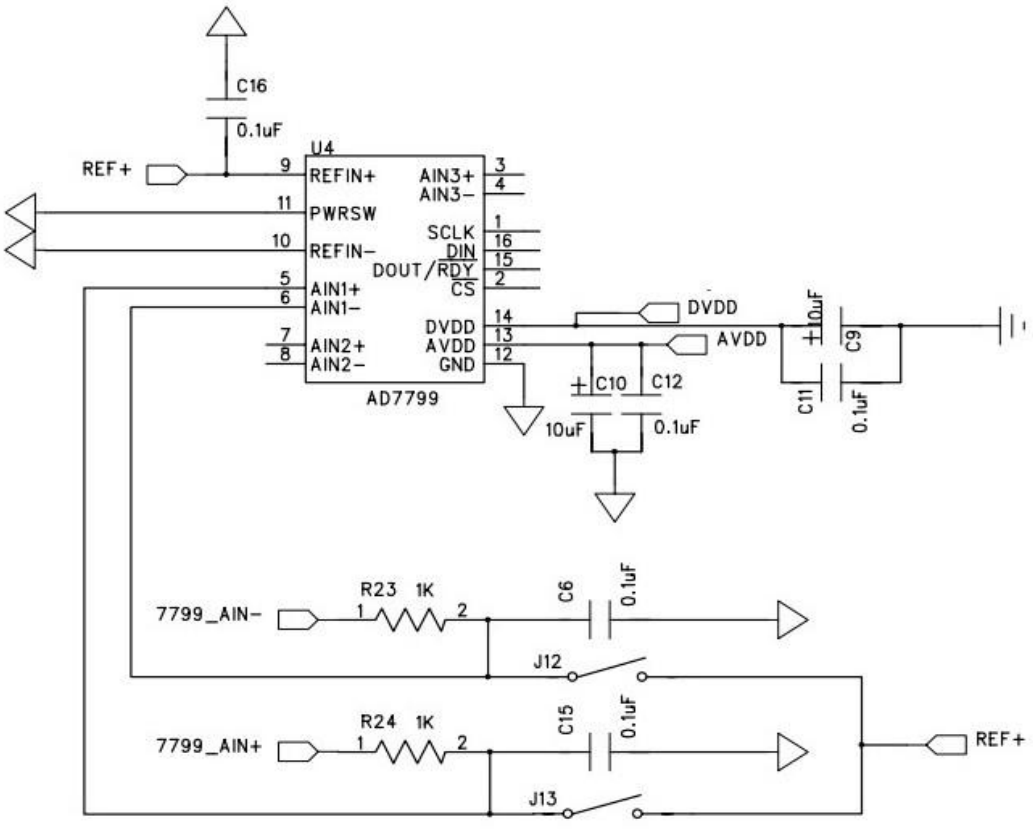


图 13　AD7799 接线图

5.2.2 AD7799 驱动程序设计

AD7799 驱动程序设计主要包括 SPI 协议介绍、基于 GPIO 的 SPI 读写实现（从 51 移植过来的 AD 驱动程序，所以没有使用硬件）、AD 读写原理（寄存器介绍），数据换算成重量以及数字滤波。

AD 驱动程序：共有四个主要函数：

write_AD 函数用来写 AD 的寄存器，该函数的执行过程如下：每次通过 SPI 总线操作，写入一位数据，然后通过移位操作，共移 8 次，读入一个字节的数据。

```
//写 AD7799 寄存器程序
void write_AD(unsigned char DINdata)
{
    unsigned int i;       // i 为计数变量，共需移位 8 次
    delay(10);
     GPIO_ResetBits(GPIOB,AD_cs );      // AD 使能位有效
     delay(10);
    for(i=0;i<8;i++)   //循环 8 次（char 为 8 位）
    {
        delay(10);
        GPIO_ResetBits(GPIOB,AD_sclk);
        delay(10);
        if(DINdata&0x80)
        GPIO_SetBits(GPIOA,AD_DIN);
        else                //转化
        GPIO_ResetBits(GPIOA,AD_DIN);
        DINdata=DINdata<<1;
        GPIO_SetBits(GPIOB,AD_sclk);
    }
    delay(10);
    GPIO_SetBits(GPIOB,AD_cs );    //AD 使能位无效
    delay(10);
}
```

read_AD 函数用来读 AD 的寄存器，该函数的执行过程如下：每次通过 SPI 总线操作，读出一位数据，然后通过移位操作，共移 8 次，读出一个字节的数据。

```
//读 AD7799 寄存器程序
unsigned char read_AD()
{
    unsigned char DOUTdata;//读出数据定义
    unsigned int i;
    DOUTdata=0;              //读出数据赋初值为 0
    GPIO_SetBits(GPIOB,AD_sclk);
    delay(10);
    GPIO_ResetBits(GPIOB,AD_cs );//AD 使能位有效
```

```
    delay(10);
    for(i=0;i<7;i++)
    {
        delay(10);
        GPIO_ResetBits(GPIOB,AD_sclk);  //转化
        delay(10);
        if(GPIO_ReadInputDataBit(GPIOA, AD_DOUT))
        DOUTdata=DOUTdata|0x01;
        DOUTdata=DOUTdata<<1;
        GPIO_SetBits(GPIOB,AD_sclk);
    }
    delay(10);
    GPIO_ResetBits(GPIOB,AD_sclk);
    delay(10);
    if(GPIO_ReadInputDataBit(GPIOA, AD_DOUT))
        DOUTdata=DOUTdata|0x01;
    GPIO_SetBits(GPIOB,AD_sclk);
    delay(10);
    GPIO_SetBits(GPIOB,AD_cs );      //AD 使能无效
    return(DOUTdata);   //返回 DOUTdata
}
```

initial_AD 函数用来初始化 AD，该函数的执行过程如下：与流程图中的操作相同，先将 0x10 写入通信寄存器，选择下次写操作为写配置寄存器；然后将 0x1710 写入配置寄存器，配置 AD 状态如下：禁止恒流源输出、AD 单极性、放大倍数 128 倍、使能内部缓冲器、选择输入通道 1；再次先将 0x08 写入通信寄存器，选择下次写操作为写模式寄存器；最后将 0x100F 写入模式寄存器，设置 AD 为单次转换模式，并且将转换速率设置为最低的 4.17Hz，以得到最长的建立时间 ，从而使得滤波精度越高。

```
void initial_AD()
{
    write_AD(0x10);  //写通信寄存器
    write_AD(0x17);  //写配置寄存器
    write_AD(0x10);
    write_AD(0x08);  //写通信寄存器
    write_AD(0x10);  //写模式寄存器
    write_AD(0x0F);
}
```

talk_AD 函数用来读取转换后的 AD 数据，该函数的执行过程如下：当 AD 单次转换成功后，其转换结果放在数据寄存器中，且 RDY 变为低，则通过判断 AD 的 RDY 位是否有效，决定何时开始读取数据。当 RDY 有效时，将 0x58 读入通信寄存器中，选择下次写操作为写数据寄存器；然后读取 AD 数据寄存器中的 24 位数据，并且将其转换为长整型的数据作为函数的返回值。

```
long talk_AD()
{
```

```
    //unsigned char ErrNUM=0;
    unsigned char ready;
    ready=read_statereg(); //读状态寄存器 RDY 位
    while(ready==0x00)
    {
        ready=read_statereg(); //如果 RDY 不为低，则一直读取 RDY 位，直至其变为低电平
    }
    write_AD(0x58);  //写通信寄存器
    ad_data[0]=read_AD(); //读数据寄存器
    ad_data[1]=read_AD();
    ad_data[2]=read_AD();
    return (ad_data[0] * 65536 + ad_data[1] * 256 + ad_data[2]);
}
```

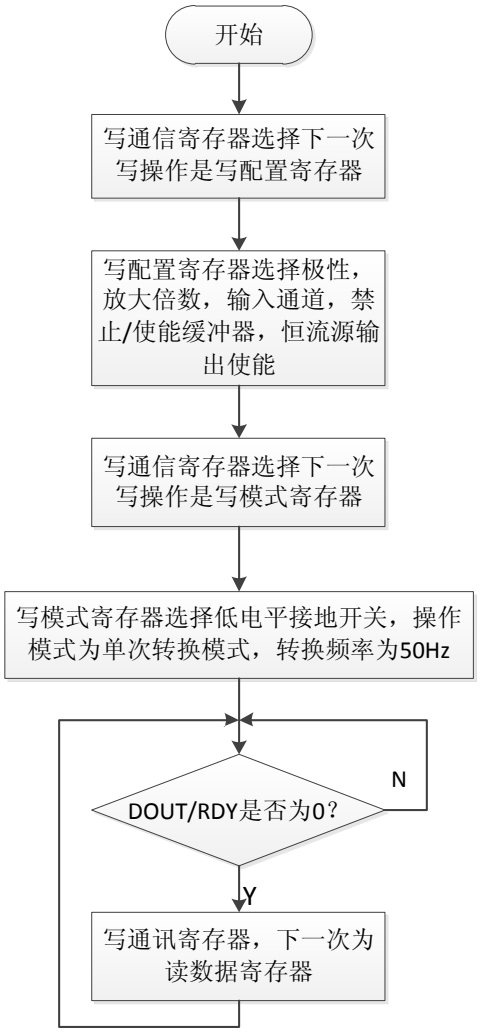


图 14 AD 程序流程图

5.3 uC/ GUI 界面模块实现

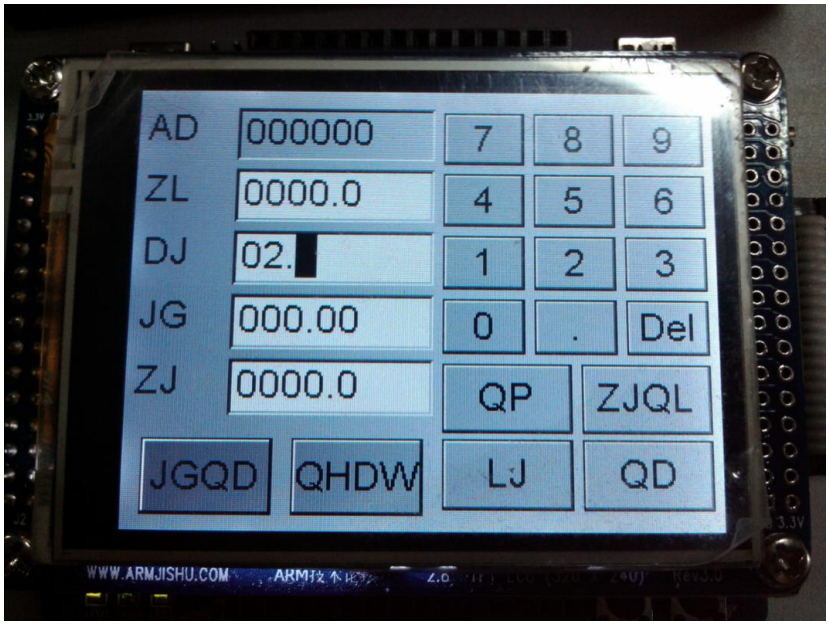


图 15 基于 GUI 程序设计的界面

程序的界面如图 15 所示，整个界面由如下几部分组成：

（1）左上角的文本框用于显示结果。AD 显示 AD 转换的 16 进制结果，ZL 显示当前物品的重量，DJ 用于输入当前物品的单价，JG 显示当前物品的总价，ZJ 显示所有物品的总价。

（2）右上角的数字按钮用于编辑价格。

（3） 其他的按钮为功能键。QP 用于实现去皮功能，ZJQL 用于将所有物品的总价清零，JGQD 用于实现价格清单的功能，QHDW 用于实现切换挡位，LJ 用于累加总价（即将当前物品的价格累加到所有物品的总价中），QD 为确定按钮。

电子称的使用过程如下：

（1）按 ZJQL 按钮将所有物品的总价清零。

（2）将物品的外包装放到秤上，按 QP 按钮将外包装的重量去掉。

（3）将物品放到秤上，并在 DJ 输入框输入物品的单价。

（4）从 ZL 显示框读取物品的重量，从 JG 显示框读取物品的价格。

（5）按 LJ 按钮将当前物品的价格累加到所有物品的总价上。

（6）如果要称的物品不止一件，重复 2-5 步，最终在 ZJ 上读取所有物品的总价。

界面中主要使用了三种 uC/GUI 的控件：按钮为 BUTTON 控件，文本输入和显示框为 EDIT 控件，还有文本输入和显示框前面的标签为 TEXT 控件。除此之外整个背景使用 uC/GUI 的 DIALOG 实现的，而如上所诉的控件以此对话框为父窗口。

uC/GUI 中对话框的新建通过调用 GUI_CreateDialogBox 函数实现，该函数的原型如下：

```
WM_HWIN
GUI_CreateDialogBox(const GUI_WIDGET_CREATE_INFO* paWidget,
                    int NumWidgets,
```

```
                                    WM_CALLBACK* cb,
                                    WM_HWIN hParent,
                                       int x0, int y0)
```

其中 paWidget 为 DIALOG 包含的子控件的描述的数组（下文介绍），NumWidgets 为子控件的数量，cb 为该对话框的回调函数（下文介绍），hParent 为该对话框父窗口的句柄，x0 和 y0 为对话框左上角的坐标。函数返回新建的对话框的句柄。需要说明的是对话框创建的同时，对话框的所有子控件也自动创建了。

在上述函数中 paWidget 描述了对话框所包含的所有子控件的描述，其类型为 GUI_WIDGET_CREATE_INFO 的数组，该类型的具体定义如下所示：

```
typedef struct    GUI_WIDGET_CREATE_INFO_struct GUI_WIDGET_CREATE_INFO;
struct GUI_WIDGET_CREATE_INFO_struct {
   GUI_WIDGET_CREATE_FUNC* pfCreateIndirect; /*  子控件的创建函数  */
   const char* pName;                                  /*  子控件的名称  */
   I16 Id;                                             /*  子控件的 ID 号  */
   I16 x0, y0, xSize, ySize;                           /*  子控件的位置和大小  */
   U16 Flags;                                          /*  子控件相关的标志  */
   I32 Para;                                           /*  子控件相关的参数  */
};
```

在本程序中该数组的定义如下（由于篇幅关系，尽列出部分）：

```
static const GUI_WIDGET_CREATE_INFO _aDialogNumPad[] = {
{ WINDOW_CreateIndirect,   0,      0,     0,     0, 320, 240},
{  TEXT_CreateIndirect,    "AD",    GUI_ID_TEXT0,    5,    10,    50,    30,
TEXT_CF_LEFT },
{  EDIT_CreateIndirect,        0,    GUI_ID_EDIT0,   55,   10,   110,   30,   0,
12},
{  TEXT_CreateIndirect,    "ZL",    GUI_ID_TEXT1,    5,    45,    50,    30,
TEXT_CF_LEFT },
...
{ BUTTON_CreateIndirect,   "7", GUI_ID_USER + 7,   keystartx+5,   keystarty+5,
45,   30},
...
{  BUTTON_CreateIndirect,    "QHDW",   GUI_ID_USER + 17,   90,   192,   70,
40}};
```

对话框的回调函数会在如下几种情况下被调用：当子控件发生变化时（比如按钮被用户按下、编辑框被编辑等），uC/GUI 会调用对话框的回调函数，因此我们在回调函数中实现了程序的主要功能。在对话框刚初始化时也会调用回调函数，我们在此时刻对子控件进行必要的初始化。

重量值的定时刷新是通过 uC/GUI 的定时器实现的，定时器的创建通过 GUI_TIMER_Create 函数实现，该函数的定义如下所示：

```
GUI_TIMER_HANDLE
GUI_TIMER_Create(GUI_TIMER_CALLBACK* cb, int Time, U32 Context, int Flags)
```

其中 cb 为定时器的回调函数，本程序通过该回调函数读取 AD 的值并刷新重量、价格的显示值。Time 为定时器的定时时间，Context 为传给回调函数的参数，Flags 为定时器的相应标志。

定时器创建以后通过调用 GUI_TIMER_Restart 函数启动定时器。

在本程序中定时器的创建和启动是在对话框初始化时在对话框的回调函数中启动的。

5.4 提高精度的算法设计

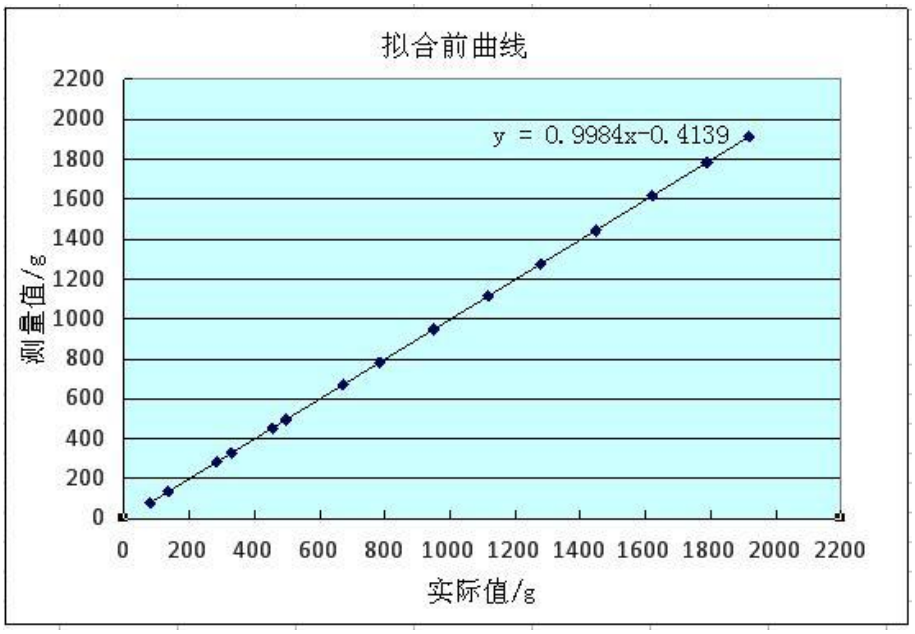
衡量电子称的一个很重要的指标就是精度，为了提高这一指标我们在软件设计中采用了拟合的算法。考虑到称重传感器具有一定的非线性性，我们采用了分段拟合的算法。现将算法的实现过程描述如下：

首先利用较高精度的电子称测得一系列物体的称重并做好记录作为相应物体的实际称重值，同时利用所涉计的电子称对该一序列物体进行称重做好记录作为物体称重的测量值，实现拟合算法的数据表格及拟合直线如下：

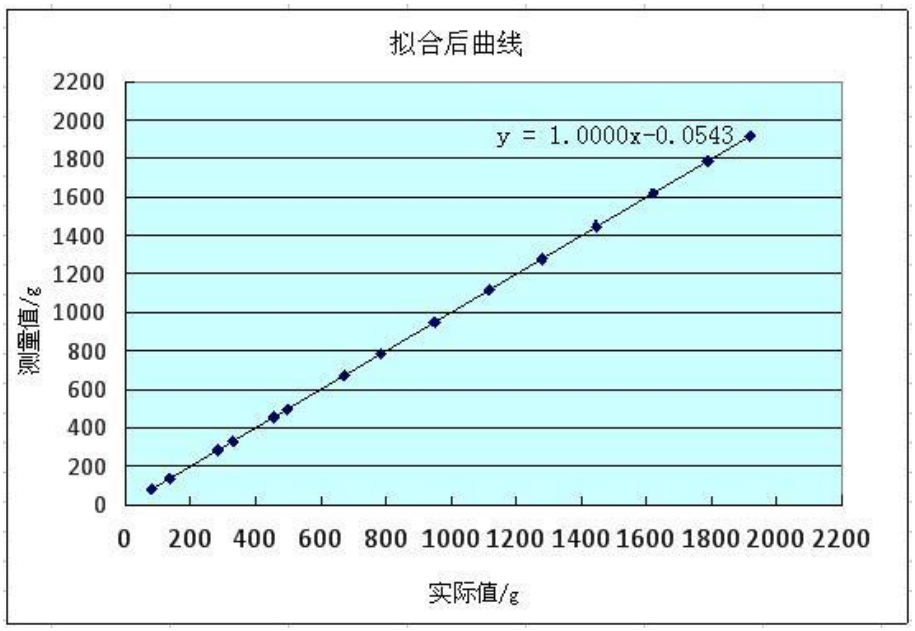
A	B	C	D
序号	实际值/g	拟合前测得值/g	拟合后测得值/g
1	79.7	79.1	79.6
2	135.8	135.1	135.6
3	284.1	283.2	283.9
4	329.5	328.2	329.7
5	454.6	453.1	454.9
6	497.4	496	497
7	671.3	670.1	671.5
8	784.1	782.3	784.1
9	949.2	947.9	949
10	1117	1115.1	1116.8
11	1278.7	1276.8	1278.9
12	1446.5	1443.7	1446
13	1620.5	1617.3	1620.8
14	1788.4	1784.5	1788.2
15	1917.5	1913.7	1917.5

拟合前后测量数据表

利用表一可得修正前后的拟合直线如下：



修正前的拟合直线

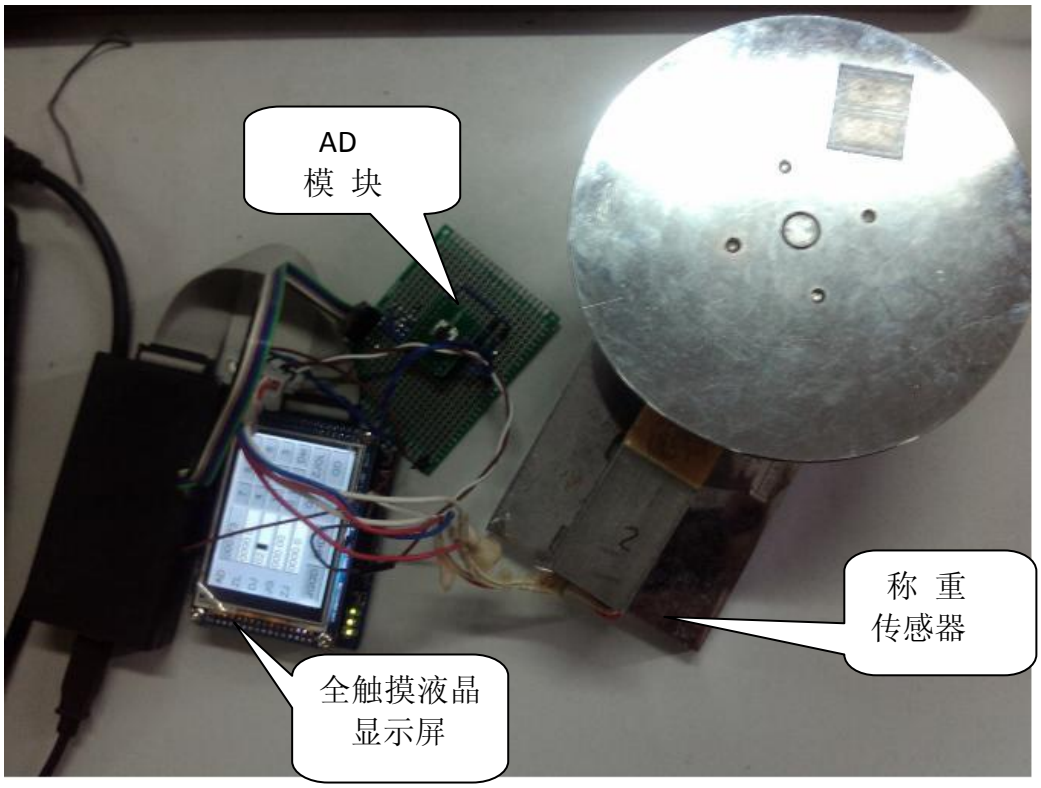


修正后的拟合直线

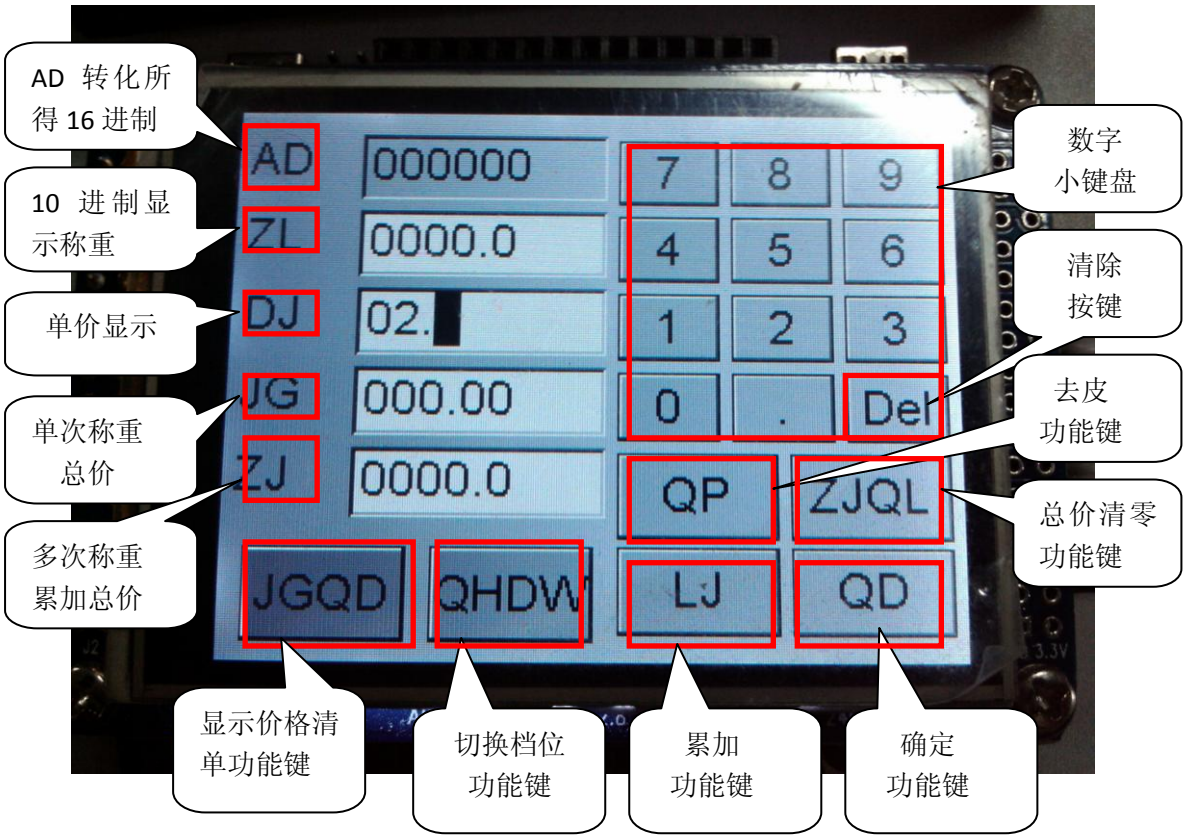
观察并比较修正前后的拟合直线值，通过这种软件的拟合修正法来提高电子称的精度是很有效的。因而其它称重范围段也可采用相同的修正方法。

六、实物图

6.1 整机实物图



6.2 基于全触摸的交互界面



七、测试方案与测试结果

7.1 测试方案

1、确定测试内容

基于 LCD 显示的数字电子称待测试内容主要为以下几个方面：

- 精度测试
- 量程测试
- 功能键测试

2、测试方案

①、精度测试方案

利用市面上精度达 0.01g,量程达 5kg 的电子称进行测试，利用该高精度电子称对 10g-5kg 范围内的物体进行称重，并记录下每一次的称重，将其作为物体的实际称重，同时利用设计的电子称测量每一个物体进行称重并记录下称重，将其作为测量称重，将一系列物体的测量称重值与实际称重值作对比，并进行一定的数据处理即可获知所设计的电子称的精度。

②、量程测试方案

量程测试主要是为了测得所设计的传感器的最大称重值，实际的电子称的最大称重值往往大于所标定的额定称重值，这主要是为了避免用户使用过程中超载损坏，所以测量并标定最大称重是产品化设计必不可少的一步。 我们确定的测试方案是利用市面上额定称重为 5kg 的电子称进行校准，得出所设计的电子

称的最大称重值。

③、功能键测试方案

功能键的测试则比较容易，通过观察、利用计算器以及相应的分析即可完成。

7.2 测试仪器

- 一台精度为 0.01g、额定称重为 5kg 的数字电子称
- 学生用计算器

7.3 测试结果

说明：

测试地点—华中科技大学启明学院实验室 608

测试环境—环境温度 24 摄氏度

1、测试记录表一

A	B	C	D
序号	实际值/g	测量值/g	误差/g
1	15.09	15.6	0.51
2	18.39	18.5	0.11
3	28.08	28.6	0.52
4	42.48	42.7	0.22
5	47.59	47.4	-0.11
6	61.83	61.7	-0.13
7	93.18	92.8	-0.38
8	145.61	145	-0.61
9	177.57	177.2	-0.37
10	211.12	211.6	0.48
11	261.45	262	0.55
12	312.42	312.6	0.18
13	395.07	395.8	0.73
14	498.69	498.3	-0.39
15	528.53	528	-0.53

测量数据表一

2、测试记录表一直线拟合

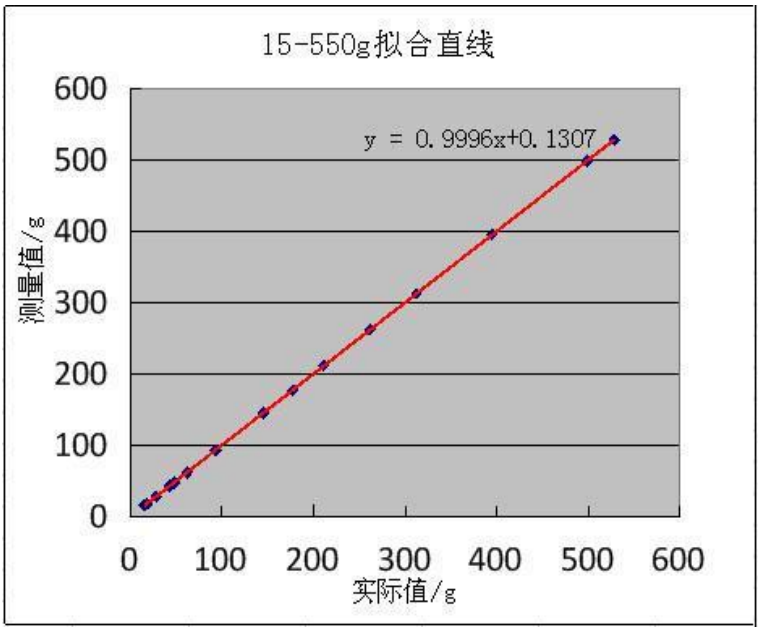


图 16 表一直线拟合

3、测试记录表二

E	F	G	H
序号	实际值/g	测量值/g	误差/g
16	556.42	556.7	0.28
17	590.69	590.3	-0.39
18	607.57	606.8	-0.77
19	684.89	684.1	-0.79
20	710.23	709.3	-0.93
21	781.06	781	-0.06
22	830.07	830.8	0.73
23	903.27	903.9	0.63
24	943.07	942.9	-0.17
25	1014.5	1014.1	-0.4
26	1087.84	1087.8	0.4
27	1161.33	1160.9	-0.43
28	1257.46	1256.9	-0.56
29	1336.58	1336.1	-0.48
30	1416.77	1416.9	0.13

4、测试记录表二直线拟合

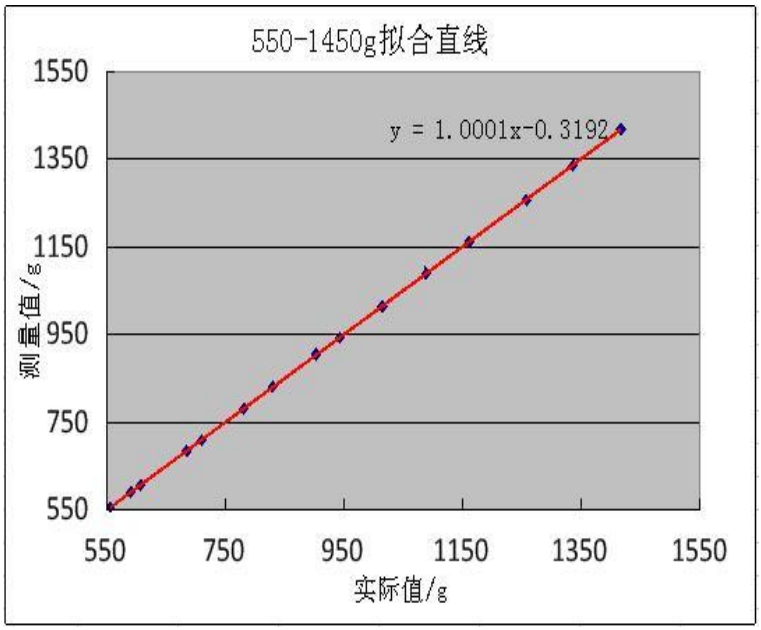


图 17 表二直线拟合

5、测试记录表三

I	J	K	L
序号	实际值/g	测量值/g	误差/g
31	1531.6	1530.7	-0.9
32	1604.72	1604	-0.72
33	1740.44	1739.7	-0.74
34	1791.13	1791	-0.13
35	1811.86	1811.8	0.06
36	1964	1963.9	-0.1
37	2111.88	2111.6	-0.28
38	2206	2205.5	-0.5
39	2295.31	2295.4	0.09
40	2388.17	2387.8	-0.37
41	2481.24	2481.6	0.36
42	2529.19	2528.5	-0.69
43	2641.66	2641.7	0.04
44	2749.22	2748.4	-0.82
45	2843.26	2842.8	-0.46

5、测试记录表三直线拟合

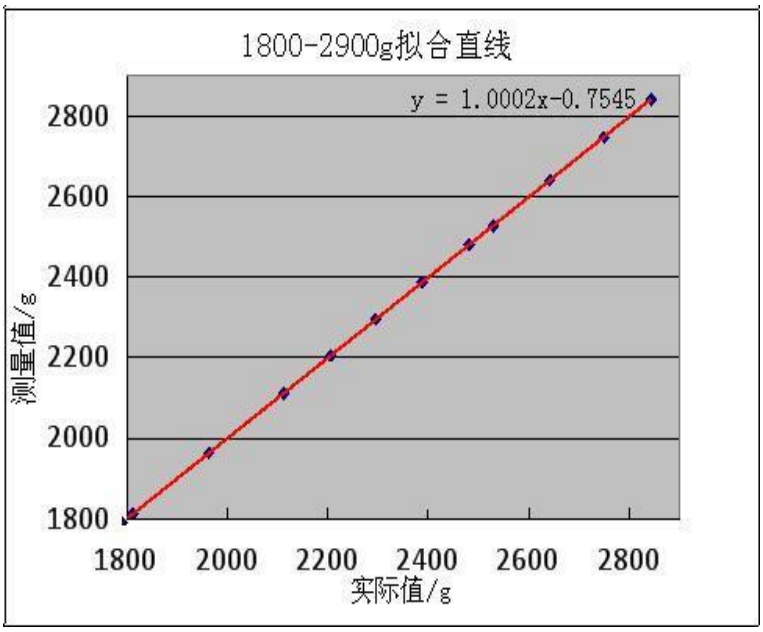


图 18 表三直线拟合

7、测试记录表四

序号	实际值/g	测量值/g	误差/g
46	2916.76	2916.2	-0.56
47	3010.84	3010.4	-0.44
48	3118.45	3118.8	0.35
49	3287.17	3287	-0.17
50	3428.19	3427.6	-0.59
51	3524.34	3523.9	-0.44
52	3642.05	3641.7	-0.35
53	3698.41	3698	-0.41
54	3858.77	3858.1	-0.67
55	3957.93	3957.3	-0.63
56	4025.36	4024.9	-0.46
57	4240.21	4239.4	-0.81
58	4352.87	4352	-0.87
59	4638.18	4637.8	-0.38
60	5096.41	5096.2	-0.21

8、测试记录表四拟合直线

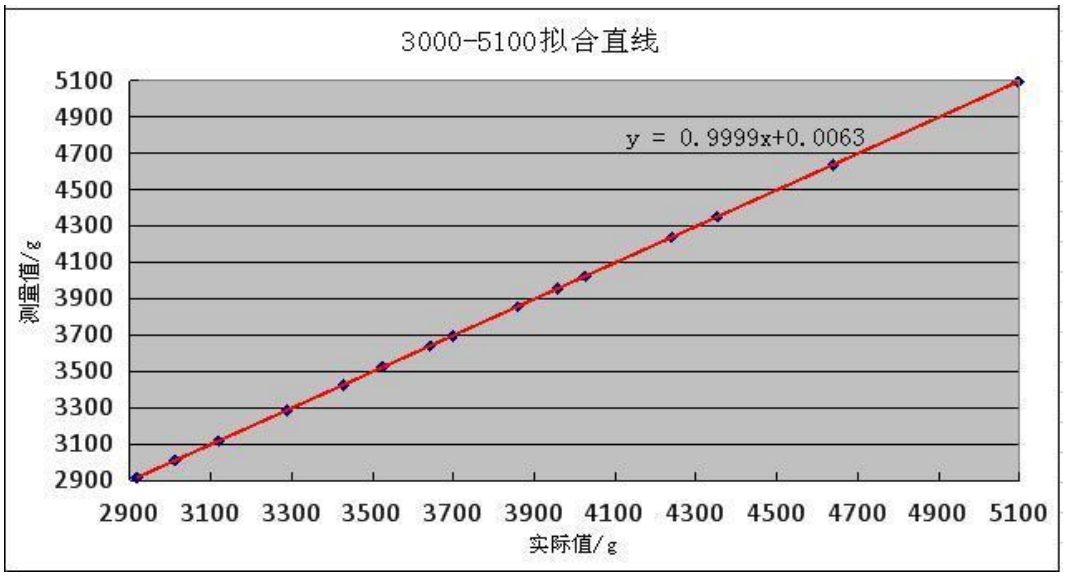


图 19 表四直线拟合

7.4 测试结果分析

我们将此次测试结果与现场答辩时所测得的结果比较发现，此次测试所得精度比答辩现场所测得精度要低一些，事后我们做了仔细的分析，总结出这可能是由于环境温度发生了变化的原因。我们所采用的称重传感器是应变式传感器，在一定的程度上受温度的影响。但是这个问题是可以解决的，由于温漂的影响是一个近似恒定的值，因而只要每次启动时利用软件做一下补偿即可。由于在测试之前我们疏忽了这一问题所以需要每次根据环境温度手动修改补偿，这正是我们的设计存在的一点不足。

根据测试结果知该数字电子称的精度达到了 0.5~1g,具有很宽的称重范围，最大称重值超过了 5kg.

八、元器件清单及造价估算

8.1 元器件清单

器件型号	数量	单价
AD7799	1	120
STM32F103RBT6 开发板	1	100
R -10k	4	
R-100k	2	
R-1k	1	
104 瓷片电容	6	

集成运放 OP07	4	
通用板	1	

8.2 造价估算

总共花费约为：100+120+20=240 元，其中 20 元为通用板、焊锡等的花费。

九、收获、体会与建议

余良驹：参加 ADI 比赛后，我感觉学习颇多，受益匪浅！

在大二暑假参加电赛时，我在组内担任队长，主要负责硬件部分。这一次参加 ADI 比赛，班级组队做 LCD 显示电子称时，我就想要做一些软件方面的工作，让自己更能掌握单片机的使用，学到更多的东西。这一次我主要负责组内的软件，一开始老师规定要统一使用 ADI 的开发板，可是因为开发板的数量有限，我们组并没有领取到，所以我另挑选了一块 ARM 开发板——神州一号 STM32 开发板，因为其板上有一块已经连接的 LCD，使用方便。

这次比赛，我主要完成了 AD7799 的驱动、基于 uC/OS-II 和 uC/GUI 的界面设计、菜单块的功能函数等。我们最初的规划是在前几周时把主体功能做出来，感觉一开始干劲十足，想趁热打铁把事情做好。于是在第 8 周之前，我一直泡在实验室里，首先我在老师给的 uC847 上调 AD7799 的驱动程序，第 3 周时将 AD7799 调通了。之后我就开始学习使用神舟一号 STM32 开发板，大概花了 2~3 周时间学习的开发板上的例程（如 I/O 口的使用、USART 串口驱动等等），然后就在原来的 uC/OS-II 和 uC/GUI 的例程基础上进行学习，开始完成自己的代码设计，第 7 周时，我完成了基于 uC/OS-II 和 uC/GUI 的主菜单模块以及一部分的功能函数。后来，因为使用了 ARM 开发板，我将 AD7799 的驱动程序进行了修改（主要是 I/O 口的驱动方面），在 AD7799 的硬件基础上想将其调试成功，然而，因为 13 周的到来，就将这件事情耽搁了。在 14 周时，队友开始帮忙调试 AD7799 且在原基础上加入一些使得 AD7799 更容易达到稳定等的辅助函数，效果十分显著。并且在 15 周时，队友与我完善了当初未写完的菜单部分的功能函数（去皮等），同时，在提高精度方面想了许多办法，最终共同想出了降低参考电压的方法，将精度提高到了 0.5g。

这一次课程设计，的的确确让我收获良多，会成为一次让我珍惜的宝贵经历！

黄小虎：ADI 比赛基本告一段落，在这个过程中学到了很多课本上有或没有的知识，当然收获中更为重要的是一次体验，一次经历！

本来抱着的是参与就好的心态，所以刚开始没怎么在意，只是一个人在一边看着 SD 卡相关的资料，后来时间越来越少了。感觉也越来越紧张了，因此就认认真真的把这次比赛当做一件重要的事来完成，而且想把它做好，尽量做完美。因此最后一段比较长得时间里花了很多时间在这件事上面，当然付出总有回报，在学到了很多知识的同时看到了自己的不足之处或有待改进的方面。

刚开始时，大家讨论想做个 SD 卡的存储的功能，后来这个任务就分配给我了，当然我也没闲着。我找了很多这方面的资料，想着时间足够，就尽量去把 SD 卡的机理和应用彻底搞清楚，因为自己并不喜欢就把网上的代码拷过来修改的作风。从 SD 卡的结构，到 SPI 通信，到文件，我都看了一遍，不过说实话，

有点难懂，不过有个大概的认识。后来想着不怎么需要 SD 卡的存储功能，也就没怎么去研究这方面的资料。不过这也许成为了一点不足的地方，现在想起来应该先找代码来实现，如果有时间的话再去看那些机理结构方面比较复杂的功能。

后来，将自己的任务调整到 ARM 的程序上面来，主要是用 ARM 来驱动 AD7799 进行 AD 转换。同组同学写了一个比较完整的 AD 驱动，然后我去调试、检验以及加入一些使得 AD 更容易达到稳定等的辅助函数。开始拿到 AD 后先是检查焊接是否有误，后面是将 AD 程序烧进 ARM 后给输入电压进行实际的测试，可能是第一次 AD 引脚全部外接用 I/O 口来控制，在程序上犯下了很多很是低级的错误，不过还好，最后在学长的帮助下解决了这些问题。通过这次试验，对 ARM 的开发有了一个基本的认识，从而为以后的学习扫清了些许障碍！

杨　玉：已经是大三下学期的我其实对于一次工程性的项目是期待已久。一方面我希望用机会用实践来检验自己所学的知识；另一方面我也希望能够在实践中证明自己。正所谓“学以致用”，我始终相信知识只有在运用中才能得到巩固和创新。

因而当我听到老师说有这样一个参赛机会时，我感到欣喜若狂，心里感叹着终于到了大展拳脚的时候了。可是事实似乎并没有我想像中那般美好，在此次电子技术课程设计中我主要负责硬件电路的设计，由于只有电子线路测试课程中的一点设计电路的基础，极度缺乏经验的我在拿到分配的任务时感到极度的迷茫，好在有张林老师的指导，才让我慢慢找到了入手点，迈出了万里长城的第一步。由于称重传感器输出的是微弱信号，这让我利用放大电路以及数字示波器等测出传感器的输出信号的大致范围就花了不少时间。其后的电路设计以及焊接电路遇到的困难之多自是不用多说。好在有同学和老师的帮助让我不断摸索出了解决办法。

对于我而言此次参赛经历带给我的远远不只是知识上的收获。首次有机会担当组长的角色，一方面让我感到庆幸，庆幸有这么一个难得的机会去锻炼自己；另一方面也让我感到压力很大。三人一起的战斗和个人的单打独斗迥然不同，如何协调进度，如何促进组内的交流沟通都是我需要考虑的问题。我觉得自己做得比较好的一点是定期会组织大家聚到一起，一来可以向组内其他成员汇报自己的进度，二来也可以将自己遇到的问题提出来作一下交流。当然由于是自己首次担当领导者的角色加上缺乏经验，无疑我有很多做得不足的地方，但让我感到庆幸的是我在这些不足之中收获了成长总结了不少经验教训，学会了怎样与别人团结合作。

当然，在此不得不提的是我要感谢我的队友、我的搭档们给予我的支持和帮助。对于我而言，结果不是最重要的，那些彻夜未眠一起在启明学院实验室奋斗的日子见证了我们的付出，见证了我们的友情也见证了我们的团结合作，而这些经历将永远成为我最宝贵的财富。

十、致谢

首先要感谢 ADI 公司的大力赞助，为我们提供了这样一个锻炼和挑战自我的平台，无疑这对于我们所有参赛者而言都是一次全新的蜕变，我们有了知识和能力上的双重收获。

其次要感谢张林老师和罗杰老师的辛勤指导以及同学的帮助，由于是初次参

加项目设计，在做整个课程设计的过程中我们遇到了很多困难，而老师和同学担当了我们坚实的后盾，这让我们感觉到任何时候我们都不是在孤军奋战而是有一群人站在我们身后默默地支持和帮助者我们，这让我们收获了不少感动。

最后不得感谢的是各位队友毫无保留的付出，无疑这是一次三人一起的战斗，因此团结合作、互帮互助是共同完成任务的先决条件。

十一、参考文献

【1】 模拟电子技术基础（第五版） 主编 康华光 副主编 陈大钦 张林 高等教育出版社

【2】传感器原理及应用 （第三版） 王化祥 张淑英 编著 天津大学出版社

【3】单片微型计算机原理与应用（第二版）胡乾斌 李光斌 李玲 喻红编著 华中科技大学出版社

【4】新编 MCS-51 单片机应用设计（第二版）张毅刚 彭喜元等编著 哈尔滨工业大学出版社

【5】神州 I 号 STM32F103 开发板用户手册 V1.0 发布

十二、源程序代码

12.1 AD7799_0.c

```
/*
文件描述：对 AD7799 的相关操作
*/

#include "stm32f10x_gpio.h"

#define AD_cs    GPIO_Pin_1                    //PB1  作为 AD_cs 连接脚
#define AD_sclk    GPIO_Pin_0                  //PB0 作为 AD_sclk 连接脚
#define AD_DIN    GPIO_Pin_1                   //PA1 作为 AD_DIN 连接脚
#define AD_DOUT    GPIO_Pin_0                  //PA0 作为 AD_DOUT 连接脚

void ADC_GPIO_Configuration(void)      //ADC 配置函数
{
   GPIO_InitTypeDef GPIO_InitStructure;
   GPIO_InitStructure.GPIO_Pin = AD_cs|AD_sclk|AD_DIN;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //输出模式
   GPIO_InitStructure.GPIO_Speed= GPIO_Speed_50MHz;
   GPIO_Init(GPIOB, &GPIO_InitStructure);       //GPIOB 组
   GPIO_Init(GPIOA, &GPIO_InitStructure);       //GPIOA 组
```

```
    //PA0 作为 AD_DOUT 连接脚
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;        //管脚 0
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;   //输入模式
    GPIO_Init(GPIOA, &GPIO_InitStructure);        //GPIOA 组
}

extern unsigned char ad_data[3];

void delay(unsigned char t)
{
      unsigned char i;
      for(i=0;i<t;i++);
}

//写 AD7799 寄存器程序
void write_AD(unsigned char DINdata)
{
      unsigned int i;        // i 为计数变量，共需移位 8 次
      delay(10);
                        GPIO_ResetBits(GPIOB,AD_cs );      // AD 使能位有效
                        delay(10);
      for(i=0;i<8;i++)    //循环 8 次（char 为 8 位）
      {
            delay(10);
            GPIO_ResetBits(GPIOB,AD_sclk);
            delay(10);
            if(DINdata&0x80)
            GPIO_SetBits(GPIOA,AD_DIN);
            else                    //转化
            GPIO_ResetBits(GPIOA,AD_DIN);
            DINdata=DINdata<<1;
            GPIO_SetBits(GPIOB,AD_sclk);
      }
      delay(10);
      GPIO_SetBits(GPIOB,AD_cs );     //AD 使能位无效
      delay(10);
}

//读 AD7799 寄存器程序
unsigned char read_AD()
{
      unsigned char DOUTdata;//读出数据定义
      unsigned int i;
      DOUTdata=0;                  //读出数据赋初值为 0
```

```
    GPIO_SetBits(GPIOB,AD_sclk);
    delay(10);
    GPIO_ResetBits(GPIOB,AD_cs );//AD 使能位有效
    delay(10);
    for(i=0;i<7;i++)
    {
        delay(10);
        GPIO_ResetBits(GPIOB,AD_sclk);  //转化
        delay(10);
        if(GPIO_ReadInputDataBit(GPIOA, AD_DOUT))
        DOUTdata=DOUTdata|0x01;
        DOUTdata=DOUTdata<<1;
        GPIO_SetBits(GPIOB,AD_sclk);
    }
    delay(10);
    GPIO_ResetBits(GPIOB,AD_sclk);
    delay(10);
    if(GPIO_ReadInputDataBit(GPIOA, AD_DOUT))
        DOUTdata=DOUTdata|0x01;
    GPIO_SetBits(GPIOB,AD_sclk);
    delay(10);
    GPIO_SetBits(GPIOB,AD_cs );      //AD 使能无效
    return(DOUTdata);   //返回 DOUTdata
}

/*读状态寄存器函数*/
unsigned char read_statereg()
{
    unsigned char state;
    unsigned char i;
    write_AD(0x40);      //写通信寄存器
    state=read_AD();//读状态寄存器
    if((~state)&0x80)//判断 RDY 位是否为 1
        i=1;
    else
        i=0;
    return(i);
}

/*AD7799 初始化函数*/
void initial_AD()
{
    write_AD(0x10);  //写通信寄存器
```

```
    write_AD(0x17);   //写配置寄存器
    write_AD(0x10);

    write_AD(0x08);   //写通信寄存器

    write_AD(0x10);   //写模式寄存器
    write_AD(0x0F);
}

long talk_AD()
{
    //unsigned char ErrNUM=0;
    unsigned char ready;
    ready=read_statereg(); //读状态寄存器 RDY 位
    while(ready==0x00)
    {
        ready=read_statereg();   // 如 果  RDY  不 为 低 ， 则 一 直 读
取 RDY 位，直至其变为低电平
    }
    write_AD(0x58);   //写通信寄存器

    ad_data[0]=read_AD(); //读数据寄存器
    ad_data[1]=read_AD();
    ad_data[2]=read_AD();

    return (ad_data[0] * 65536 + ad_data[1] * 256 + ad_data[2]);
}


/*
**********************************************************************
**********************************
*
```

12.2 uC/GUI

```
---------------------------------------------------------------------
File              : WIDGET_NumPad.c
Purpose           : Shows how to use a numpad as input device on a touch screen
--------------------------END-OF-HEADER-----------------------------
*/

#include "DIALOG.h"
```

```
#include <math.h>

#define BUFF_SIZE (128)
#define  PTH    0x4000
//#define WTH      0.1
#define WTH        0.5
long buf[BUFF_SIZE];
long pre_data=0;
double pre_weight=0;
//float peel_weight=0;
long peel_ad=0;


long talk_AD(void);
long ad_filter(long data);
long avr_ad_data();
double weight_balance(double weight);
//long avr_AD(void);

/*********************************************************************
*
*
*            Static data
*
**********************************************************************
*
*/

static GUI_TIMER_HANDLE timer=0;

/* Bitmap data for arrow keys */
static GUI_CONST_STORAGE GUI_COLOR _aColorsArrow[] = {
  0xFFFFFF, 0x000000
};

static GUI_CONST_STORAGE GUI_LOGPALETTE _PalArrow = {
  2,      /* number of entries */
  1,      /* No transparency */
  &_aColorsArrow[0]
};

static GUI_CONST_STORAGE unsigned char _acArrowRight[] = {
  ____XX__, ________,
  ____XXXX, ________,
```

```
  XXXXXXXX, XX______,
  ____XXXX, _________,
  ____XX__, _________
};

static GUI_CONST_STORAGE unsigned char _acArrowLeft[] = {
  ____XX__, _________,
  __XXXX__, _________,
  XXXXXXXX, XX______,
  __XXXX__, _________,
  ____XX__, _________
};

static GUI_CONST_STORAGE GUI_BITMAP _bmArrowRight = {
  10,                 /* XSize */
  5,                  /* YSize */
  2,                  /* BytesPerLine */
  1,                  /* BitsPerPixel */
  _acArrowRight, /* Pointer to picture data (indices) */
  &_PalArrow          /* Pointer to palette */
};

static GUI_CONST_STORAGE GUI_BITMAP _bmArrowLeft = {
  10,                 /* XSize */
  5,                  /* YSize */
  2,                  /* BytesPerLine */
  1,                  /* BitsPerPixel */
  _acArrowLeft,    /* Pointer to picture data (indices) */
  &_PalArrow          /* Pointer to palette */
};

#define keystartx 165
#define keystarty 7
/* Dialog resource of numpad */
static const GUI_WIDGET_CREATE_INFO _aDialogNumPad[] = {
/*    Function                    Text          Id                      Px      Py      Dx
Dy */
  { WINDOW_CreateIndirect,      0,              0,                      0,      0, 320, 240},
  {  TEXT_CreateIndirect,         "AD",         GUI_ID_TEXT0,           5,    10,    50,    30,
TEXT_CF_LEFT    },
  {  EDIT_CreateIndirect,         0,            GUI_ID_EDIT0,           55,    10,  110,    30,
0, 12},
  {  TEXT_CreateIndirect,         "ZL",         GUI_ID_TEXT1,           5,    45,    50,    30,
TEXT_CF_LEFT },
```

```
  { EDIT_CreateIndirect,        0,          GUI_ID_EDIT1,        55,   45,  110,   30,
0, 12},
  { TEXT_CreateIndirect,        "DJ",       GUI_ID_TEXT2,         5,   80,   50,   30,
TEXT_CF_LEFT   },
  { EDIT_CreateIndirect,        0,          GUI_ID_EDIT2,        55,   80,  110,   30,
0, 12},
  { TEXT_CreateIndirect,        "JG",       GUI_ID_TEXT3,         5,  115,   50,   30,
TEXT_CF_LEFT },
  { EDIT_CreateIndirect,        0,          GUI_ID_EDIT3,        55,  115, 110,   30,
0, 12},
  { TEXT_CreateIndirect,        "ZJ",       GUI_ID_TEXT4,         5,  150,   50,   30,
TEXT_CF_LEFT   },
  { EDIT_CreateIndirect,        0,          GUI_ID_EDIT4,        55,  150, 110,   30,
0, 12},

  { BUTTON_CreateIndirect,      "7",          GUI_ID_USER  +   7,     keystartx+5,
keystarty+5,   45,   30},
  { BUTTON_CreateIndirect,      "8",          GUI_ID_USER  +   8,    keystartx+55,
keystarty+5,   45,   30},
  { BUTTON_CreateIndirect,      "9",          GUI_ID_USER  +   9,   keystartx+105,
keystarty+5,   45,   30},
  { BUTTON_CreateIndirect,      "4",          GUI_ID_USER  +   4,     keystartx+5,
keystarty+40,  45,   30},
  { BUTTON_CreateIndirect,      "5",          GUI_ID_USER  +   5,    keystartx+55,
keystarty+40,  45,   30},
  { BUTTON_CreateIndirect,      "6",          GUI_ID_USER  +   6,   keystartx+105,
keystarty+40,  45,   30},
  { BUTTON_CreateIndirect,      "1",          GUI_ID_USER  +   1,     keystartx+5,
keystarty+75,  45,   30},
  { BUTTON_CreateIndirect,      "2",          GUI_ID_USER  +   2,    keystartx+55,
keystarty+75,  45,   30},
  { BUTTON_CreateIndirect,      "3",          GUI_ID_USER  +   3,   keystartx+105,
keystarty+75,  45,   30},
  { BUTTON_CreateIndirect,      "0",          GUI_ID_USER  +   0,     keystartx+5,
keystarty+110, 45,   30},
  { BUTTON_CreateIndirect,      ".",          GUI_ID_USER  +  10,    keystartx+55,
keystarty+110, 45,   30},
  { BUTTON_CreateIndirect,      "Del",        GUI_ID_USER  +  11,   keystartx+105,
keystarty+110, 45,   30},
  { BUTTON_CreateIndirect,      "QP",         GUI_ID_USER  +  12,     keystartx+5,
keystarty+145, 70,   37},
  { BUTTON_CreateIndirect,      "ZJQL",       GUI_ID_USER  +  13,    keystartx+80,
keystarty+145, 70,   37},
```

```
  { BUTTON_CreateIndirect,      "LJ",          GUI_ID_USER + 14,    keystartx+5,
keystarty+185,   70,   37},
  { BUTTON_CreateIndirect,      "QD",          GUI_ID_USER + 15,    keystartx+80,
keystarty+185,   70,   37},
  { BUTTON_CreateIndirect,      "JGQD",      GUI_ID_USER + 16,    10,  192,    70,
40},
  { BUTTON_CreateIndirect,      "QHDW",      GUI_ID_USER + 17,    90,  192,    70,
40},
};

/* Dialog resource of user dialog */
static const GUI_WIDGET_CREATE_INFO _aDialogUser[] = {
/*    Function                    Text          Id                        Px     Py     Dx
Dy */
  { FRAMEWIN_CreateIndirect, "Dialog", 0,                    5,    5, 140,   120,
FRAMEWIN_CF_MOVEABLE},
  { EDIT_CreateIndirect,        0,            GUI_ID_EDIT0,        10,   10,  110,   40,
0, 12},
  { EDIT_CreateIndirect,        0,            GUI_ID_EDIT1,        10,   60,  110,   40,
0, 12},
};

/* Title of sample */
static char _aTitle[] = {"WIDGET_NumPad"};

/* Explanation of sample */
static char * _apExplain[] = {
  {"This sample shows how to use a numpad as input"},
  {"device. This can be usefull if no keyboard"},
  {"is available and the user should edit numeric"},
  {"values or text on a touch screen."},
};


/*********************************************************************
*
*
*           Static code
*
**********************************************************************
*
*/
static void _cbTimer( GUI_TIMER_MESSAGE* pTM )
{
```

```
    long ad;
    static long old_ad;
    WM_HWIN hDlg, hItem;
    static float single;
    static float weight;
    static float price;
    hDlg = (WM_HWIN)(pTM->Context);
    hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT0);
    ad = talk_AD();
     ad = (old_ad*9+ad)/10;
    old_ad=ad;
    ad = ad-0x1f2100;
    ad = ad-peel_ad;
     ad=ad_filter(ad);
    EDIT_SetValue(hItem,ad);
    hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT1);
          //weight=(ad*3.3/0xFFFFFF)*25000-peel_weight;
     weight=(ad*3.0)/0xFFFFFF*2610.9;
    if(weight<=100)
        weight=0.9977*weight-0.2932;
    else
        weight=1.0016*weight+0.538;

   // weight = DigFil(weight,0,0);
    weight=weight_balance(weight);
    EDIT_SetFloatValue(hItem,weight);

    //weight = EDIT_GetFloatValue(hItem);

    hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT2);
    price = EDIT_GetFloatValue(hItem);
    hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT3);
    single = weight*price/1000;
    EDIT_SetFloatValue(hItem,single);
    GUI_TIMER_Restart(timer);
}

/*********************************************************************
*
*
*           _cbDialogNumPad
*
* Purpose:
*     Callback function of the numpad.
```

```
*/
static void _cbDialogNumPad(WM_MESSAGE * pMsg) {
   GUI_RECT r;
   int i, NCode, Id, Pressed = 0;
   float single,All;
   long ADdata;
   WM_HWIN hDlg, hItem;
   hDlg = pMsg->hWin;
   switch (pMsg->MsgId) {
   case WM_INIT_DIALOG:
      for (i = 0; i < GUI_COUNTOF(_aDialogNumPad) - 1; i++) {
         hItem = WM_GetDialogItem(hDlg, GUI_ID_USER + i);
         BUTTON_SetFocussable(hItem,  0);                                    /* Set all
buttons non focussable */
      }
      hItem = WM_GetDialogItem(hDlg, GUI_ID_TEXT0);
      TEXT_SetFont(hItem,&GUI_Font24_ASCII);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_TEXT1);
      TEXT_SetFont(hItem,&GUI_Font24_ASCII);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_TEXT2);
      TEXT_SetFont(hItem,&GUI_Font24_ASCII);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_TEXT3);
      TEXT_SetFont(hItem,&GUI_Font24_ASCII);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_TEXT4);
      TEXT_SetFont(hItem,&GUI_Font24_ASCII);

      hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT0);
      WM_DisableWindow(hItem);
      EDIT_SetMaxLen(hItem,6);
      EDIT_SetHexMode(hItem, 0, 0, 0xFFFFFF);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT1);
      EDIT_SetMaxLen(hItem,6);
      EDIT_SetFloatMode(hItem, 0, 0, 6000, 1, 0);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT2);
      EDIT_SetMaxLen(hItem,4);
      EDIT_SetFloatMode(hItem, 2, 0, 100, 1, 0);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT3);
      EDIT_SetMaxLen(hItem,6);
      EDIT_SetFloatMode(hItem, 0, 0, 1000, 2, 0);
      hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT4);
      EDIT_SetMaxLen(hItem,6);
      EDIT_SetFloatMode(hItem, 0, 0, 6000, 1, 0);

      timer = GUI_TIMER_Create(_cbTimer,0,(U32)hDlg,0);
```

```
      GUI_TIMER_SetPeriod(timer,1);
      GUI_TIMER_Restart(timer);

       break;
    case WM_NOTIFY_PARENT:
       Id        = WM_GetId(pMsg->hWinSrc);          /* Id of widget */
       NCode = pMsg->Data.v;                         /* Notification code */
       switch (NCode) {
       case WM_NOTIFICATION_CLICKED:
          Pressed = 1;
         break;
       case WM_NOTIFICATION_RELEASED:
          if ((Id >= GUI_ID_USER) && (Id <(GUI_ID_USER + 11))) {
                int Key;
                char acBuffer[10];
                BUTTON_GetText(pMsg->hWinSrc,  acBuffer,  sizeof(acBuffer));  /*  Get
the text of the button */
                Key = acBuffer[0];
                GUI_SendKeyMsg(Key,  1);                                        /*
Send a key message to the focussed window */
           }
          else if(Id == GUI_ID_USER + 11)
          {
                 int Key;
                 Key = GUI_KEY_BACKSPACE;
                 GUI_SendKeyMsg(Key, 1);
          }
          else if(Id == GUI_ID_USER + 12)
          {
                 hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT0);
                 peel_ad = EDIT_GetValue(hItem);
          }
          else if(Id == GUI_ID_USER + 13)
          {
                 hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT4);
                 EDIT_SetFloatValue(hItem,0);
          }
          else if(Id == GUI_ID_USER + 14)
          {
                 hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT4);
                 All=EDIT_GetFloatValue(hItem);
                 hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT3);
                 single=EDIT_GetFloatValue(hItem);
                 hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT4);
```

```
                All=single+All;
                EDIT_SetFloatValue(hItem,All);

          }
           break;
       }
    default:
       WM_DefaultProc(pMsg);
    }
}

/*********************************************************************
*
*
*           _cbDialogUser
*
* Purpose:
*     Callback function of the user dialog.
*/
static void _cbDialogUser(WM_MESSAGE * pMsg) {
   int i, NCode, Id;
   WM_HWIN hDlg, hItem;
   hDlg = pMsg->hWin;
   switch (pMsg->MsgId) {
   case WM_INIT_DIALOG:
      for (i = 0; i < 2; i++) {
         hItem = WM_GetDialogItem(hDlg, GUI_ID_EDIT0 + i);   /* Get the handle of
the edit widget */
         EDIT_SetText(hItem, "123456");                      /* Set text */
      }
      break;
   case WM_NOTIFY_PARENT:
      Id       = WM_GetId(pMsg->hWinSrc);          /* Id of widget */
      NCode = pMsg->Data.v;                        /* Notification code */
      switch (NCode) {
         case WM_NOTIFICATION_RELEASED:          /* React only if released */
            if (Id == GUI_ID_OK) {                /* OK Button */
               GUI_EndDialog(hDlg, 0);
            }
            if (Id == GUI_ID_CANCEL) {            /* Cancel Button */
               GUI_EndDialog(hDlg, 1);
            }
            break;
      }
```

```
      break;
    default:
      WM_DefaultProc(pMsg);
    }
}

/*********************************************************************
*
*
*           _cbDesktop
*
* Purpose:
*     This routine handles the drawing of the desktop window.
*/
static void _cbDesktop(WM_MESSAGE * pMsg) {
  int i;
  switch (pMsg->MsgId) {
  case WM_PAINT:
    GUI_SetBkColor(GUI_RED);
    GUI_Clear();
    GUI_SetFont(&GUI_Font24_ASCII);
    GUI_DispStringHCenterAt(_aTitle, 160, 5);
    GUI_DispNextLine();
    GUI_SetFont(GUI_DEFAULT_FONT);
    GUI_DispNextLine();
    for (i = 0; i < GUI_COUNTOF(_apExplain); i++) {
      GUI_DispStringHCenterAt(_apExplain[i], 160, GUI_GetDispPosY());
      GUI_DispNextLine();
    }
    break;
  }
}

/*********************************************************************
*
*
*           Exported code
*
**********************************************************************
*
*/
/*********************************************************************
*
*
```

```
*            MainTask
*/

void MainTask(void) {
   WM_HWIN hNumPad;
   GUI_Init();
   BUTTON_SetDefaultFont(&GUI_Font24_ASCII);
   EDIT_SetDefaultFont(&GUI_Font24_ASCII);
// WM_SetCallback(WM_HBKWIN, _cbDesktop);
   hNumPad = GUI_CreateDialogBox(_aDialogNumPad,
                                 GUI_COUNTOF(_aDialogNumPad),
                                 _cbDialogNumPad,  WM_HBKWIN,  0,  0);  /*
Create the numpad dialog */
   WM_SetStayOnTop(hNumPad, 1);

   while (1) {
//       GUI_ExecDialogBox(_aDialogUser,
//                         GUI_COUNTOF(_aDialogUser),
//                         _cbDialogUser,  WM_HBKWIN,  0,  0);                  /*
Execute the user dialog */
      GUI_ExecCreatedDialog (hNumPad);
       GUI_Delay(100);
   }
}


long avr_AD(void)
{
      int i;
      long avr;
      for(i=0;i<BUFF_SIZE;i++)
      {
            avr=avr+talk_AD();
      }
      avr=avr/BUFF_SIZE;
}

long ad_filter(long data)
{
        long avr_data;
         char j;

        avr_data=avr_ad_data();
        if(abs(data-avr_data)<=PTH)
```

```
        {
           for(j=0;j<BUFF_SIZE-1;j++)
              buf[j+1]=buf[j];
           buf[0]=data;
        }
        if(abs(data-avr_data)>PTH)
        {
             if(abs(pre_data-avr_data)>=PTH)
           {
              for(j=0;j<BUFF_SIZE;j++)
                   buf[j]=data;
           }
        }
        pre_data=data;
              avr_data=avr_ad_data();
        return avr_data;
}

long avr_ad_data()
{
       char i;
      long avr_data,sum=0;
       for(i=0;i<BUFF_SIZE;i++)
            sum=sum+buf[i];
      avr_data=sum/BUFF_SIZE;
      return avr_data;
}
double weight_balance(double weight)
{
     double iweight;
     if(fabs(pre_weight-weight)>WTH)
         iweight=weight;
     else
          iweight=pre_weight;
     pre_weight=iweight;
     return iweight;
}
```

12.3 APP.c

```
/*
```

```
*********************************************************************************************
*                                                EXAMPLE CODE
*
*                                (c) Copyright 2003-2006; Micrium, Inc.; Weston, FL
*
*                  All rights reserved.  Protected by international copyright laws.
*                  Knowledge of the source code may NOT be used to develop a similar product.
*                  Please help us continue to provide the Embedded community with the finest
*                  software available.  Your honesty is greatly appreciated.
*********************************************************************************************
*/

/*
*********************************************************************************************
*                                            INCLUDE FILES
*********************************************************************************************
*/

#include <includes.h>


/* ----------------- APPLICATION GLOBALS ------------------ */
static OS_STK AppTaskStartStk[APP_TASK_START_STK_SIZE];
static OS_STK AppTaskUserIFStk[APP_TASK_USER_IF_STK_SIZE];
static OS_STK AppTaskKbdStk[APP_TASK_KBD_STK_SIZE];

//static    OS_EVENT            *AppUserIFMbox;

int ExTick;

/*
*********************************************************************************************
*                                      LOCAL FUNCTION PROTOTYPES
*********************************************************************************************
*/
```

```
static void AppTaskCreate(void);
static void AppTaskStart(void *p_arg);

static void AppTaskUserIF(void *p_arg);
static void AppTaskKbd(void *p_arg);

/*
*********************************************************************************
*********************************
*                                                                  main()
*
* Description : This is the standard entry point for C code.  It is assumed that your code will call
*                       main() once you have performed all necessary initialization.
*
* Arguments     : none
*
* Returns       : none
*********************************************************************************
*********************************
*/

void MainTask(void);

int  main(void)
{
    INT8U  err;
    /* Set the Vector Table base location at 0x08000000 */
    NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);

    BSP_IntDisAll(); /* Disable all interrupts until we are ready to accept them */

    OSInit();           /* Initialize "uC/OS-II, The Real-Time Kernel" */

    /* Create the start task */
    OSTaskCreateExt(AppTaskStart,
                    (void *)0,
                    (OS_STK *)&AppTaskStartStk[APP_TASK_START_STK_SIZE-1],
                    APP_TASK_START_PRIO,
                    APP_TASK_START_PRIO,
                    (OS_STK *)&AppTaskStartStk[0],
                    APP_TASK_START_STK_SIZE,
                    (void *)0,
```

```
                        OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 13)
    OSTaskNameSet(APP_TASK_START_PRIO, "Start Task", &err);
#endif

    OSStart();                                                  /* Start
multitasking (i.e. give control to uC/OS-II)            */
}

/*
*********************************************************************
**********************************
*                                                 STARTUP TASK
*
* Description : This is an example of a startup task.    As mentioned in the book's text,
you MUST
*                    initialize the ticker only once multitasking has started.
*
* Arguments   : p_arg   is the argument passed to 'AppTaskStart()' by
'OSTaskCreate()'.
*
* Returns     : none
*
* Notes       : 1) The first line of code is used to prevent a compiler warning
because 'p_arg' is not
*                    used.   The compiler should not generate any code for this
statement.
*********************************************************************
**********************************
*/

static void AppTaskStart (void *p_arg)
{
    (void)p_arg;

    BSP_Init();                                                 /* Initialize
BSP functions                                       */
#if (OS_TASK_STAT_EN > 0)
    OSStatInit();                                               /* Determine
CPU capacity                                        */
#endif
```

```
//      AppUserIFMbox  =  OSMboxCreate((void  *)0);                    /*  Create
MBOX for communication between Kbd and UserIF         */
      AppTaskCreate();                                                 /*  Create
application tasks                                                     */

       while(DEF_TRUE)
      {
         /* Task body, always written as an infinite loop. */
           OSTaskSuspend(OS_PRIO_SELF);
         /* ExTick=TPReadY();
           GUI_DispDecAt(ExTick,20,40,4);
         */

        }
}
/*
*********************************************************************
***********************************
*                                              CREATE APPLICATION TASKS
*
* Description:    This function creates the application tasks.
*
* Arguments    :    none
*
* Returns      :    none
*********************************************************************
***********************************
*/

static    void    AppTaskCreate(void)
{
   INT8U    err;

   OSTaskCreateExt(AppTaskUserIF,(void                                *)0,(OS_STK
*)&AppTaskUserIFStk[APP_TASK_USER_IF_STK_SIZE-1],APP_TASK_USER_IF_PRIO,AP
P_TASK_USER_IF_PRIO,(OS_STK *)&AppTaskUserIFStk[0],
                           APP_TASK_USER_IF_STK_SIZE,
                           (void *)0,
                           OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);
#if (OS_TASK_NAME_SIZE > 8)
   OSTaskNameSet(APP_TASK_USER_IF_PRIO, "User I/F", &err);
#endif
```

```
   OSTaskCreateExt(AppTaskKbd,(void                                    *)0,(OS_STK
*)&AppTaskKbdStk[APP_TASK_KBD_STK_SIZE-1],APP_TASK_KBD_PRIO,APP_TASK_K
BD_PRIO,(OS_STK *)&AppTaskKbdStk[0],
                           APP_TASK_KBD_STK_SIZE,
                           (void *)0,
                           OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 8)
   OSTaskNameSet(APP_TASK_KBD_PRIO, "Keyboard", &err);
#endif
}
/*
*********************************************************************
**********************************
*                                                  USER INTERFACE TASK
*
* Description : This task updates the LCD screen based on messages passed to it by
AppTaskKbd().
*
* Arguments     : p_arg     is the argument passed to 'AppStartUserIF()' by
'OSTaskCreate()'.
*
* Returns       : none
*********************************************************************
**********************************
*/

static   void   AppTaskUserIF (void *p_arg)
{

  (void)p_arg;
   // GUI_Init();
   while(DEF_TRUE)
   {
        MainTask();
   }
}


/*
*********************************************************************
**********************************
*                                          KEYBOARD RESPONSE TASK
*
```

```
* Description : This task monitors the state of the push buttons and passes messages
to AppTaskUserIF()
*
* Arguments    : p_arg    is the argument passed to 'AppStartKbd()' by
'OSTaskCreate()'.
*
* Returns      : none
*********************************************************************
**********************************
*/

static  void  AppTaskKbd (void *p_arg)
{
    u8 tick=0;
  (void)p_arg;
    while(DEF_TRUE)
    {
        tick++;
          OSTimeDlyHMSM(0,0,0,10);
         GUI_TOUCH_Exec();

    }
}
```