实验五　重量测量

一、简述重量测量实验中，模拟量是怎样转换成数字量的？

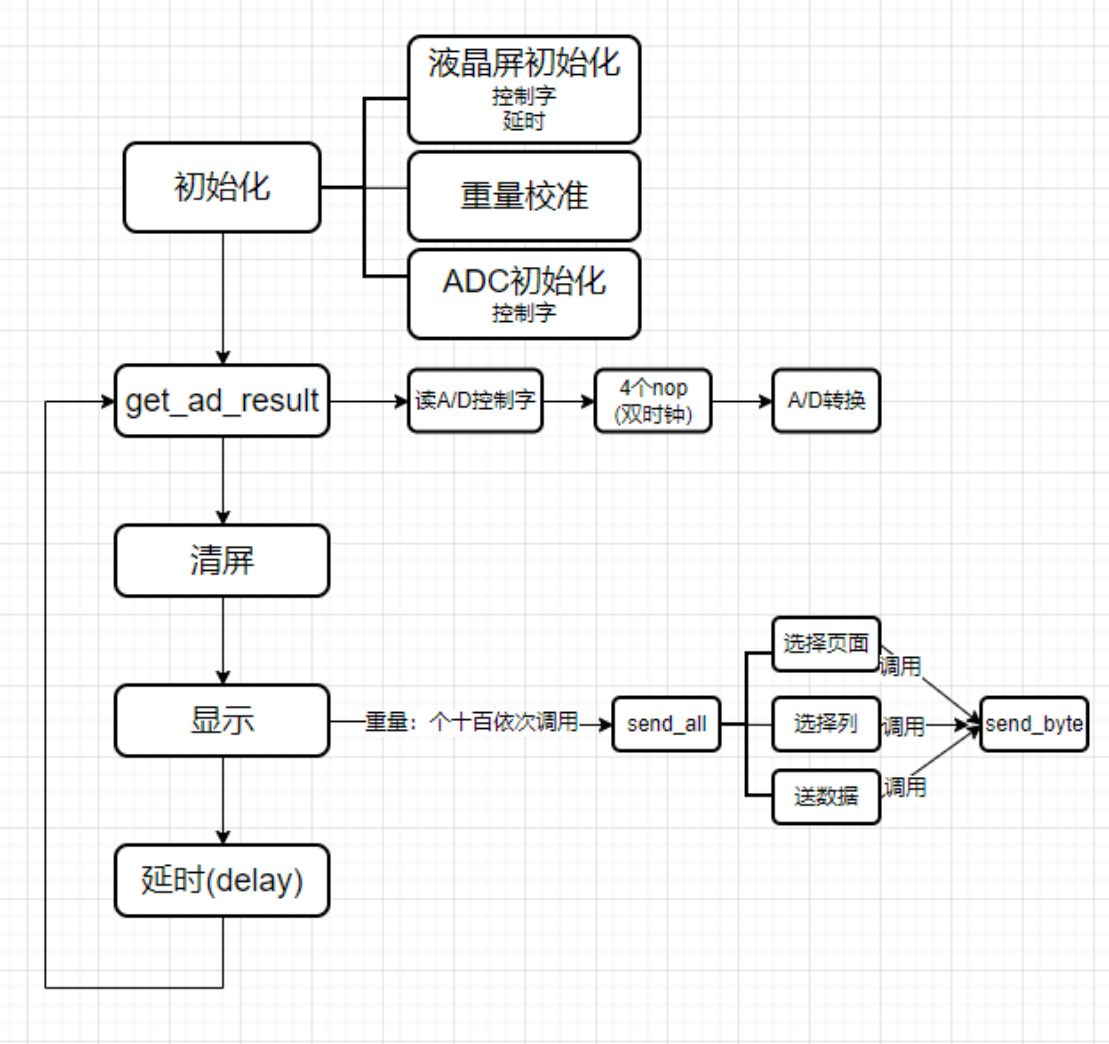
A/D 转化原理：

1）如何转化：重量传感器采用压敏电阻。利用压敏电阻采集应变,产生变化的阻值。利用放大电路将其转化为电压值，通过数模转换将电压值转化成 CPU 处理的数字信号。传感器根据编制的程序将数字信号转换为砝码重量显示输出。

2）工作方式：逐次逼近：对于我们得到的模拟量，首先取产生的 8 位数字量的一半，如果得到的模拟量在低位，则清空高位寄存器，以此类推，类似二分法。

二、流程图及程序

1、流程图



2、程序

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int

sbit CS1 = P1 ^ 7;  ///左半边
sbit CS2 = P1 ^ 6;  ///右半边
sbit E = P3 ^ 3;    ///使能信号
sbit RW = P3 ^ 4;   ///读写操作选择
sbit RS = P3 ^ 5;   ///寄存器选择(数据/指令)
sbit RES = P1 ^ 5;  ///复位 低电平有效
```

```
sbit BUSY = P2 ^ 7; //当前为运行状态

/**Declare SFR associated with the ADC */
sfr ADC_CONTR = 0xBC; ///ADC control register
sfr ADC_RES = 0xBD;   ///ADC hight 8-bit result register
sfr ADC_LOW2 = 0xBE;  ///ADC low 2-bit result register
sfr P1ASF = 0x9D;     ///P1 secondary function control register: P1 口模拟功能
控制寄存器
sfr AURX1 = 0xA2;     ///AURX1 中的 ADRJ 位用于转换结果寄存器的数据格式调整控制

/**Define ADC operation const for ADC_CONTR*/
#define ADC_POWER 0x80   ///ADC power control bit
#define ADC_FLAG 0x10    ///ADC complete flag
#define ADC_START 0x08   ///ADC start control bit
#define ADC_SPEEDLL 0x00 ///540 clocks
#define ADC_SPEEDL 0x20  ///360 clocks
#define ADC_SPEEDH 0x40  ///180 clocks
#define ADC_SPEEDHH 0x60 ///90 clocks

uchar ch = 0; ///ADC channel NO.0

uchar code zima[20][32] =
{
    0x00, 0x00, 0xC0, 0xE0, 0x30, 0x10, 0x08, 0x08, 0x08, 0x08, 0x08, 0x18, 0x30,
0xE0, 0xC0, 0x00,
    0x00, 0x00, 0x07, 0x0F, 0x18, 0x10, 0x20, 0x20, 0x20, 0x20, 0x20, 0x10, 0x18,
0x0F, 0x07, 0x00, ///*"0"*0/

    0x00, 0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0xF0, 0xF8, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x20, 0x20, 0x20, 0x20, 0x3F, 0x3F, 0x20, 0x20, 0x20, 0x20,
0x00, 0x00, 0x00, ///*"1"*1/

    0x00, 0x00, 0x60, 0x50, 0x10, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x98, 0xF0,
0x70, 0x00, 0x00,
    0x00, 0x00, 0x20, 0x30, 0x28, 0x28, 0x24, 0x24, 0x22, 0x22, 0x21, 0x20, 0x30,
0x18, 0x00, 0x00, ///*"2"*2/

    0x00, 0x00, 0x30, 0x30, 0x08, 0x08, 0x88, 0x88, 0x88, 0x88, 0x58, 0x70, 0x30,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x31, 0x11, 0x1F,
0x0E, 0x00, 0x00, ///*"3"*3/

    0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x10, 0xF0, 0xF8, 0xF8, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x04, 0x06, 0x05, 0x05, 0x04, 0x24, 0x24, 0x24, 0x3F, 0x3F, 0x3F, 0x24,
0x24, 0x24, 0x00, ///*"4"*4/

    0x00, 0x00, 0x00, 0xC0, 0x38, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x08,
0x08, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x29, 0x21, 0x20, 0x20, 0x20, 0x20, 0x20, 0x30, 0x11, 0x1F,
0x0E, 0x00, 0x00, ///*"5"*5/
```

```
    0x00, 0x00, 0x80, 0xE0, 0x30, 0x10, 0x98, 0x88, 0x88, 0x88, 0x88, 0x88, 0x98,
0x10, 0x00, 0x00,
    0x00, 0x00, 0x07, 0x0F, 0x19, 0x31, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x11,
0x1F, 0x0E, 0x00, ///*"6"*6/

    0x00, 0x00, 0x30, 0x18, 0x08, 0x08, 0x08, 0x08, 0x08, 0x88, 0x48, 0x28, 0x18,
0x08, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x3E, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, ///*"7"*7/

    0x00, 0x00, 0x70, 0x70, 0xD8, 0x88, 0x88, 0x08, 0x08, 0x08, 0x08, 0x98, 0x70,
0x70, 0x00, 0x00,
    0x00, 0x0C, 0x1E, 0x12, 0x21, 0x21, 0x20, 0x21, 0x21, 0x21, 0x23, 0x12, 0x1E,
0x0C, 0x00, 0x00, ///*"8"*8/

    0x00, 0xE0, 0xF0, 0x10, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x18, 0x10, 0xF0,
0xC0, 0x00, 0x00,
    0x00, 0x00, 0x11, 0x33, 0x22, 0x22, 0x22, 0x22, 0x22, 0x32, 0x11, 0x1D, 0x0F,
0x03, 0x00, 0x00, ///*"9"*9/

    0x08, 0x08, 0x0A, 0xEA, 0xAA, 0xAA, 0xAA, 0xFF, 0xA9, 0xA9, 0xA9, 0xE9, 0x08,
0x08, 0x08, 0x00,
    0x40, 0x40, 0x48, 0x4B, 0x4A, 0x4A, 0x4A, 0x7F, 0x4A, 0x4A, 0x4A, 0x4B, 0x48,
0x40, 0x40, 0x00, ///*"重"*10/

    0x40, 0x40, 0x40, 0xDF, 0x55, 0x55, 0x55, 0xD5, 0x55, 0x55, 0x55, 0xDF, 0x40,
0x40, 0x40, 0x00,
    0x40, 0x40, 0x40, 0x57, 0x55, 0x55, 0x55, 0x7F, 0x55, 0x55, 0x55, 0x57, 0x50,
0x40, 0x40, 0x00, ///*"量"*11/

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xC0, 0xC0, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, ///*":"*12/

    0x00, 0x04, 0x04, 0xE4, 0x24, 0x24, 0x24, 0x3F, 0x24, 0x24, 0x24, 0xE4, 0x04,
0x04, 0x00, 0x00,
    0x00, 0x00, 0x80, 0x43, 0x31, 0x0F, 0x01, 0x01, 0x01, 0x3F, 0x41, 0x43, 0x40,
0x40, 0x70, 0x00, ///*"克"*13/

};

void send_byte(uchar dat, uchar cs1, uchar cs2);
void send_all(uint page, uint lie, uint offset);
void delay(uint x);
void init_adc();
void init_yejing();
void calibrate();
int get_ad_result();
void clearscreen();

int cweight;
int weight;
```

```
void main()
{
    init_yejing();
    init_adc();
    calibrate(); ///校准

    while (1){
        weight = (get_ad_result() - cweight) / 2 - 50;
        weight += weight / 10; ///真实重量
        clearscreen();

        send_all(1, 1, 10);                ///重
        send_all(1, 2, 11);                ///量
        send_all(1, 3, 12);                ///:
        send_all(4, 3, weight / 100);       ///百
        send_all(4, 4, (weight / 10) % 10); ///十
        send_all(4, 5, weight % 10);        ///个
        send_all(4, 6, 13);                ///克

        delay(50000);
    }
}

void init_yejing()
{
    send_byte(192, 1, 1); ///设置起始行
    send_byte(63, 1, 1);  ///打开显示开关
}

void send_byte(uchar dat, uchar cs1, uchar cs2)
{
    P2 = 0xff;
    CS1 = cs1;
    CS2 = cs2;
    RS = 0;
    RW = 1;
    E = 1; //读状态字
    while (BUSY)
        ;

    ///送数据或控制字
    E = 0;
    RS = !(cs1 && cs2), RW = 0; //写指令代码
    P2 = dat;
    E = 1;
    delay(3);

    E = 0; //总线释放

    CS1 = CS2 = 0;
}
void send_all(uint page, uint lie, uint offset)
{
    uint i, j, k = 0;
```

```
    for (i = 0; i < 2; ++i)
    {
        send_byte(184 + i + page, 1, 1);                ///选择页面   184-页面地址设置
        send_byte(64 + lie * 16 - (lie > 3) * 64, 1, 1); ///选择列号
        for (j = 0; j < 16; ++j)
            send_byte(zima[offset][k++], lie < 4, lie >= 4); ///送数
    }
}

void init_adc()
{
    P1ASF = 1;     ///Set  P1.0 as analog input port
    AURX1 |= 0X04; ///AURX1 中的 ADRJ 位用于转换结果寄存器的数据格式调整控制

    ADC_RES = ADC_LOW2 = 0; ///Clear previous result

    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch; ///ch=0 ADC channel NO.0
    delay(4);                                           ///ADC power-on delay and Start A/D conversion
}

int get_ad_result()
{
    int ADC_result;
    ADC_RES = ADC_LOW2 = 0; ///Clear previous result
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_(); ///Must wait before inquiry
    while (!(ADC_CONTR & ADC_FLAG))
        ;                                           ///Wait complete flag
    ADC_result = (ADC_RES & 0x03) * 256 + ADC_LOW2; ///ADC_RES 中存高 2 位
    ADC_CONTR &= ~ADC_FLAG;                          ///Close ADC flag 位置 0
    return ADC_result;                               ///Return ADC result
}

void calibrate() //校正
{
    cweight = (get_ad_result() - 0) / 2;
}

void delay(uint x)
{
    while (x--)
        ;
}

void clearscreen()
{
```

```
    int i, j;
    for (i = 0; i < 8; ++i)
    {
        send_byte(184 + i, 1, 1); ///页
        send_byte(64, 1, 1);      ///列
        for (j = 0; j < 64; ++j)
        {
            send_byte(0x00, 0, 1);
            send_byte(0x00, 1, 0);
        }
    }
}
```

三、课后思考题

1. 调零的原理，软件调零和调零调零的区别。

答：

调零的原理使以测量仪器空载情况为基准，即真实值为零的时候，通过软件或硬件（机械）的方法调整读数，使读数也归零，从而减小误差。

软件调零是指在程序内部获得测量值后，通过一定的算法或公式来计算得到一个准确值再进行输出。

硬件调零是指在读取模拟量时更改电阻的阻值从而使测量值接近真实值。

二者的区别主要在于硬件调零的操作发生在 A/D 转换之前，因此程序读取的测量值就是真实值，而软件调零的操作发生在 A/D 转换之后，因此程序读取到测量值并不是真实值。

硬件调零通过调零电阻的精确度保障调零的精度，软件调零通过数据结构和算法来保证调零的精度。

在本实验中，硬件调零是调节三个变阻器。

2. 模/数和数/模的信号转换原理。

答：

模/数和数/模的信号转换原理：ADC 的转换原理根据的电路形式有所不同。

ADC 电路通常由两部分组成，它们是：采样、保持电路和量化、编码电路。其中量化、编码电路是最核心的部件，任何 AD 转换电路都必须包含这种电路。ADC 电路的形式很多，通常可以并为两类：

间接法：它是将采样保持的模拟信号先转 换成与模拟量成正比的时间或频率，然后再把它转换为数字量。这种通常是采用 时钟脉冲计数器，它又被称为计数器式。

它的工作特点是：工作速度低，转换精 度高，抗干扰能力强。

直接法：通过基准电压与采样保持信号进行比较，从而转换为数字量。

它的工作特点是：工作速度高，转换精度容易保证。

DA 转换器是由数码寄存器、模拟电子开关电路、解码电路、求和电路及基准电压及部分组成。数字量是以串行或并行方式输入并存储在数码寄存器中，寄存器 输出的每位数码驱动对应数位上的电子开关将电阻解码网络中获得的相应数位权值送入数位求和电路中，求和电路将各位权值相加就得到与数字量相应的模拟量。

3. I^2C 总线在信号通讯过程中的应用请写出校准电子秤的过程。

答：

I2C 总线在信号通讯过程中的应用：I2C 总线是 Philips 公司开发的一种双向两线多主机总线，利用两根信号线来实现设备之间的信息传递，一根为数据线 SDA，一根为时钟线 SCL。它能方便地实现芯片间的数据传输与控制。通过两线缓冲接口和内部控制与状态寄存器，可方便地完成多机间的非主从通信或主从通信。基于 I2C 总线的多机通信电路结构简单、程序编写方便，易于实现系统软硬件的模块化和标准化，被广泛用于系统内部微控制器和外部设备之间的串行通讯。

四、实验中遇到哪些问题，怎样解决的？有哪些收获？

对液晶显示器的控制方法不清，导致在显示时没有头绪。根据实验附录知道了在显示时首先要读控制字，然后依次送入页号，列号，数据，这些都由送入特定格式的数据实现。