10 种简单的数字滤波算法(C 语言源程序)

假定从 8 位 AD 中读取数据（如果是更高位的 AD 可定义数据类型为 int）,子程序为 get_ad();

1、限副滤波

```
/*  A 值可根据实际情况调整
    value 为有效值，new_value 为当前采样值
    滤波程序返回有效的实际值  */
#define A 10
char value;
char filter()
{
  char  new_value;
  new_value = get_ad();
  if ( ( new_value - value > A ) || ( value - new_value > A )
    return value;
  return new_value;
}
```

2、中位值滤波法

```
/*  N 值可根据实际情况调整
    排序采用冒泡法*/
#define N  11
char filter()
{
  char value_buf[N];
  char count,i,j,temp;
  for ( count=0;count<N;count++)
  {
    value_buf[count] = get_ad();
    delay();
  }
  for (j=0;j<N-1;j++)
  {
    for (i=0;i<N-j;i++)
    {
      if ( value_buf[i]>value_buf[i+1] )
      {
        temp = value_buf[i];
        value_buf[i] = value_buf[i+1];
         value_buf[i+1] = temp;
      }
    }
  }
```

```
  return value_buf[(N-1)/2];
}
```

3、算术平均滤波法

```
/*
*/
#define N 12
char filter()
{
  int  sum = 0;
  for ( count=0;count<N;count++)
  {
    sum + = get_ad();
    delay();
  }
  return (char)(sum/N);
}
```

4、递推平均滤波法（又称滑动平均滤波法）

```
/*
*/
#define N 12
char value_buf[N];
char i=0;
char filter()
{
  char count;
  int  sum=0;
  value_buf[i++] = get_ad();
  if ( i == N )   i = 0;
  for ( count=0;count<N,count++)
    sum = value_buf[count];
  return (char)(sum/N);
}
```

5、中位值平均滤波法（又称防脉冲干扰平均滤波法）

```
/*
*/
#define N 12
char filter()
{
  char count,i,j;
  char value_buf[N];
  int  sum=0;
  for  (count=0;count<N;count++)
```

```
  {
    value_buf[count] = get_ad();
    delay();
  }
  for (j=0;j<N-1;j++)
  {
    for (i=0;i<N-j;i++)
    {
      if ( value_buf[i]>value_buf[i+1] )
      {
        temp = value_buf[i];
        value_buf[i] = value_buf[i+1];
         value_buf[i+1] = temp;
      }
    }
  }
  for(count=1;count<N-1;count++)
    sum += value[count];
  return (char)(sum/(N-2));
}
```

6、限幅平均滤波法

```
/*
*/
```

略 参考子程序 1、3

7、一阶滞后滤波法

```
/* 为加快程序处理速度假定基数为 100，a=0~100 */
#define a 50
char value;
char filter()
{
  char   new_value;
  new_value = get_ad();
  return (100-a)*value + a*new_value;
}
```

8、加权递推平均滤波法

```
/* coe 数组为加权系数表，存在程序存储区。*/
#define N 12
char code coe[N] = {1,2,3,4,5,6,7,8,9,10,11,12};
char code sum_coe = 1+2+3+4+5+6+7+8+9+10+11+12;
char filter()
{
  char count;
```

```
    char value_buf[N];
    int   sum=0;
    for (count=0,count<N;count++)
    {
      value_buf[count] = get_ad();
      delay();
    }
    for (count=0,count<N;count++)
      sum += value_buf[count]*coe[count];
    return (char)(sum/sum_coe);
}
```

9、消抖滤波法

```
#define N 12
char filter()
{
    char count=0;
    char new_value;
    new_value = get_ad();
    while (value !=new_value);
    {
      count++;
      if (count>=N)    return new_value;
       delay();
      new_value = get_ad();
    }
    return value;
}
```

10、限幅消抖滤波法

```
/*
*/
```

一个非常适合单片机的滤波算法

一个非常适合单片机的滤波算法（内附匠人分析）

2011-01-08 09:17

匠人按：今天在论坛里看到一位网友发了个滤波算法。开始以为是一种新算法，后来仔细分析，发现原来是一阶滤波（低通滤波）的变形。原文及匠人的分析附录如下：

----------以下为原文----------

连接：http://bbs.21ic.com/icview-170880-1-1.html

单片机大多资源小，算法占用的资源越小越好，现在介绍就是一个占用很小资源的算法，这个算法是本人在进行扫描仪设计，实现灰度转二值时实现动态阈值，当时为了跟踪灰度等级的变化，需要一个灰度积分跟踪电路，开始使用一个电容积分电路，用灰度信号对电容充电，放电时以该电容电压的比例进行，实现对输入信号的跟踪，但用电容的电路设计比较复杂。过后发现这种比例放电的思想用软件实现非常简单，且具有积分、微分的作用。

具体公式如下：

SUM=SUM-SUM/n+S

其中：S 为采样值，SUM 为保存值，n 是放电比例、最好选 2 的幂次数，单片机移位即可，不需要做除法，跟随后得到的值为 SUM/n，SUM 注意不溢出，预留的容量为采样数最大值的 n 倍，初始化时如果是跟踪一段时间后使用，可以是任何值，否则可以用采样值乘 n 初始化。使用值为 SUM/n（下文中 SA），实现 SUM/n 对 S 的跟踪。还有一个关键是计算周期 T，即多长时间进行一次。

一、积分作用：

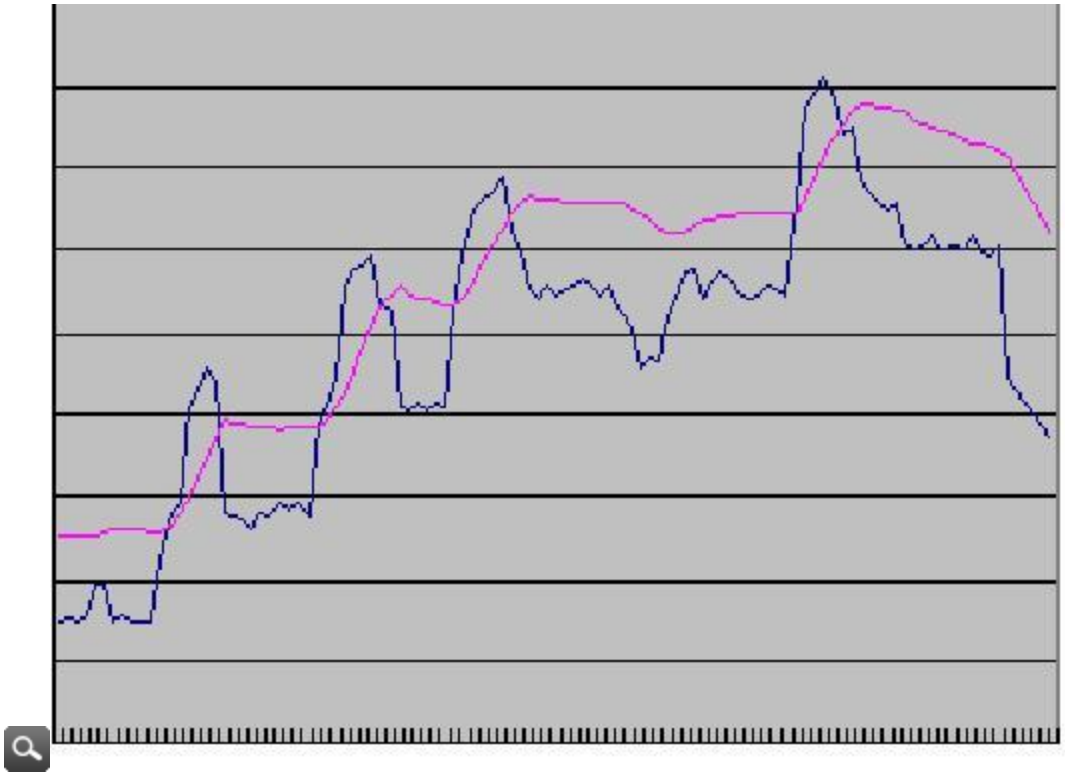
1．平滑滤波（滑动平均滤波）

由公式中可以看出，每次采样、计算后，当前采样的影响对 SUM/n 只有 1/n,而且采到的值随次数的增加影响越来越小直至没有，相关性逐渐减弱，而且是连续相关。如果计算周期与采样周期相同，使用计算后的值对干扰有 n 倍的抑制，即积分的平滑滤波作用，如 1ms 采样一次，同时运算一次，

则使用值 SA=SUM/n 为抑制干扰的结果，且同样是 1ms 给出一个结果，使用两个变量实现平滑滤波，并且是即时使用的，与采样几次平均的平滑不同。

2．动态阈值

在很多应用中需要动态阈值，比如触摸按键的键阈值门限，血压计的心率检出，前面提到的灰度转二值黑白图像等（灰度转二值因为扫描速度 2.5Mbyte/S,不能使用软件运算，但可以使用可编程逻辑实现）。动态阈值是对信号积分后得到的低频变化再与基本门限相加在触摸按键中增加动态阈值可以提高其适应性和可靠性。关键是根据按键反应时间和按键间隔确定按键积分参数，跟踪速度，n、T 越大跟踪的越慢，积分效果越好。



3．锁相作用：把上边的积分运算，用于对时间上周期的信号，例如根据过零触发信号锁定交流电源周期，使用两次 T 时间不同，其它相同的运算，由于 T 不同，跟踪速度不同，当两次运算的结果相等时可以确认为锁定，这时得到的是准确的电源周期值，而相位偏差也很小。

二、微分作用：
公式中的 SA 趋近采样值 S，如果 S 是线性的，SA 的值是可控滞后于 S，那么运算的间隔时间 T 不同，得到的跟踪曲线的滞后特性不同，这种滞后特性的差和间隔时间就是微分特性，表示曲线的变化规律。如电热水壶，温度的变化相当于采样时间是还相当慢的，局部可以作为线性变化来处理。下边以设计电热水壶的过程来说明微分作用。
电热水壶出口一直使用蒸汽开关这种需要交专利费的方式。不使用蒸汽开关检测压力只能使用热敏器件检测温度。
温度检测的环境要求：
1. 海拔高度不同的地区水开的温度不同。
2. 热敏器件的误差较大，必须克服，否则可生产性不足。
3. 环境温度不同，电源电压不同，装水量不同。
由要求 1、2 决定检测温度不能判别水开与否，需要检测温度的变化率，但温度变化率的判别又和要求 3 相关，下边曲线图为热水器的加热曲线。蓝线为即时温度，橙色为一次运算后的曲线。

图中加热过程中间添加了冷水，曲线有一段下降，过后的加热过程两个曲线有个

差异滞后，同一个时间的两个曲线差表示了加热效率的变化，其中最大的加热效率体现了环境温度不同，电源电压不同，装水量不同的综合效果。由于滞后的时间可以通过计算周期 T 来调整，知道滞后时间又有相减的差，这就是微分效应，加热过程整个就是效率的变化过程。我们可以通过 1 秒钟计算一次，2 秒钟计算一次，加上原始数据得到三个曲线，效率的变化一目了然。

第一次的水开检测使用效率的方法，同时也会得到水开时的温度检测值，微分特性本身是可以预知变化趋势的，如果 1 秒钟计算一次，用当前检测值减去这次计算的结果，这个差与当前值相加，就可以做为当前 1 秒后的结果，也就是预知 1 秒后加热的检测值，结合第一次得到的水开温度检测值，以后的水开检测就有两个判断条件。

----------以下为匠人的分析----------

拨开迷雾看真相，作者的这个算法，本质上，就是一阶滤波（低通滤波）。

引用作者原来的公式

SUM=SUM-SUM/n+S

首先点破一下，等号前面的 SUM 代表的是本次运算结果，而等号后面的 SUM 代表的是上次运算结果。

且看匠人如何推导：

设：

SUM=A

SUM/n=B=本次滤波结果

1/n=a （一阶滤波系数）

S=本次新采样值

则：

A=nB

B=A/n

另外：

A、B 代表本次值

A’、B’ 代表上次值

作者原公式逐步推导：

原始：SUM=SUM-SUM/n+S

第 1 步：A=A’ – A’/n +S

第 2 步：nB=nB’ – B’ +S

第 3 步：B= (nB’ – B’ +S)/n

第 4 步：B=B’ –B’/n +S/n

第 5 步：B=（1- 1/n）B’ + （1/n）*S

第 6 步：B= （1-a）B’+ a *S

推导到最后一步，是不是很眼熟啦？

呵呵，这就是经典的一阶滤波（低通滤波）的标准公式了。

基于 C8051F310 的高灵敏车辆检测算法

2009-10-27　　嵌入式在线　　收藏 | 打印

1 引言

弯路转弯处经常出现一段盲区，司机看不到弯路对面是否有车辆通过，因而引发大量的交通事故，因此，消除盲区造成的交通事故显得尤为重要。为此，设计了基于 C8051F310 的山路转弯预警系统。该系统当检测到弯路对面有车时可及时通过交通警示灯提前警示司机注意避让。因此，准确判断是否有车辆经过是该系统设计的关键。

2 系统设计

2．1 系统设计方案

该系统设计的主要目的是警示司机在行驶时注意安全，预防事故。在山路转弯处两边分别放置该系统，每边系统控制一警示灯。当一方系统检测到车辆时，通过 RF 通讯发送至对方系统，对方系统接收到信号后，控制警示灯闪烁以提示司机。

图 1 为系统设计组成框图。其中，车辆检测传感器采用正弦波振荡电路检测车辆，在检测电路中，输出信号频率由 C8051F310 采集得到，然后通过一阶滤波算法处理，滤除掉因环境因素等产生的频率干扰，并进一步计算验证 C8051F310 的采集精度。

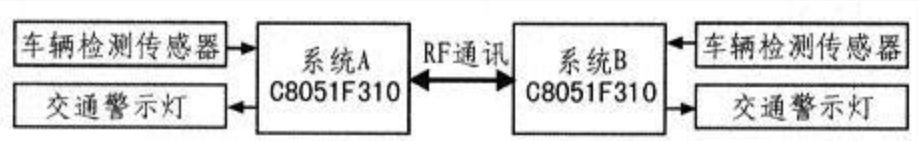


图 1　系统设计组成框图

2．2 系统的硬件电路设计

图 2 为系统主要硬件电路。车辆检测传感器的输入信号为 U(t)，该正弦信号通过比较器变为方波信号后，再输入到单片机 C8051F310，然后单片机通过计数器采集信号频率。

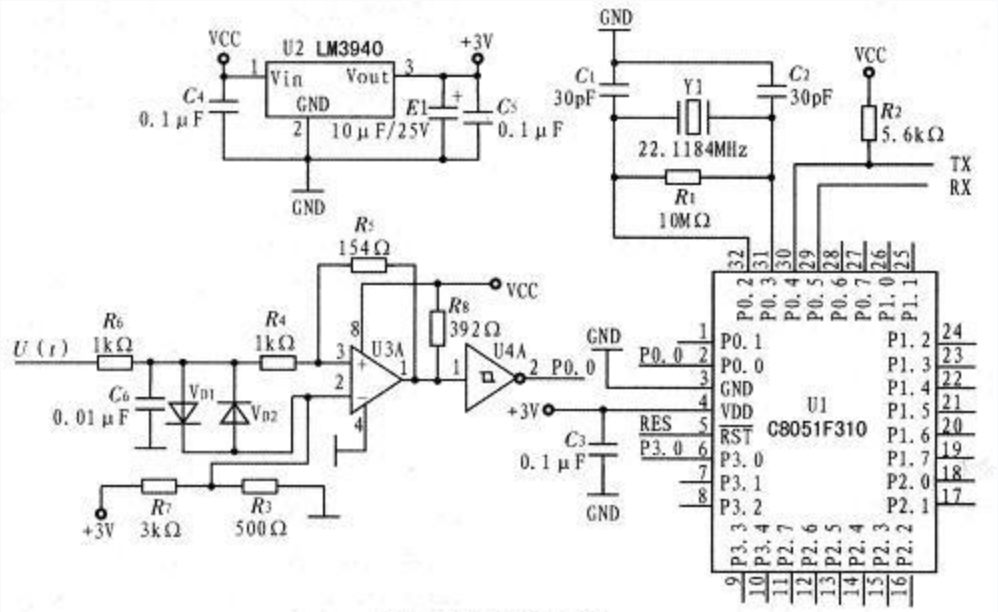


图 2　系统硬件电路

2．3 一阶滤波算法

一阶滤波，即一阶惯性滤波。一阶低通滤波算法公式为：

$$Y(n)=\alpha X(n)+(1-\alpha)Y(n-1) \qquad (1)$$

式中，α 为滤波系数，X(n)为本次采样值，Y(n-1)为上次滤波输出值，Y(n)为本次滤波输出值。

一阶低通滤波法采用本次采样值与上次滤波输出值进行加权，得到有效滤波值，使得输出对输入有反馈作用。滤波系数为 0～1；该系数决定新采样值在本次滤波结果中所占的权重。一阶滤波系数可以是固定的，也可以按一定程序算法自动计算。但一阶滤波算法无法完全兼顾灵敏度和平稳度。只能寻找一个平衡点，在该系统设计可接受的灵敏度范围内选取尽可能好的平稳度。即当数据快速变化时，滤波结果能及时跟进(灵敏度优先)；而当数据趋于稳定，在固定点上下振荡时，滤波结果趋于平稳(平稳度优先)。

2．4 车辆检测电路

图 3 为正弦波振荡电路。该电路用于车辆检测电路传感器，能够感应出金属物体的存在。采用涡流传感方式，将埋入地下的探测线圈直接接入正弦波振荡电路。

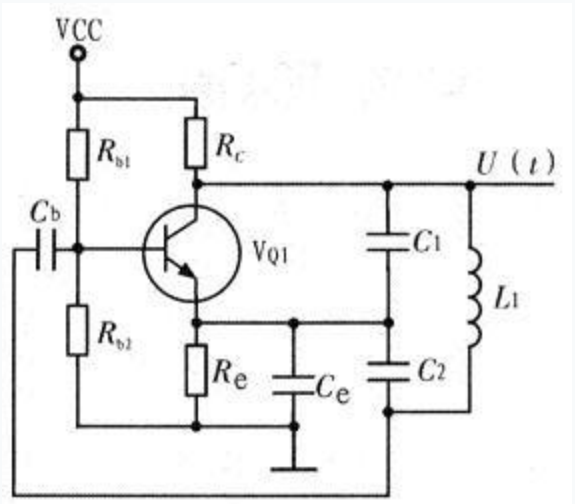


图 3　正弦波振荡电路

在未检测到车辆等金属物体时，振荡电路输出信号 u0(t)的频率基本变化不大，但该值不是一直保持不变，而会在一定范围内漂移。当检测到车辆等金属物体时，U0(t)的频率 f0 会突变为 f。频率差△f=f-f0，其中△f 的范围经大量的实验得出一般为几百赫兹到几千赫兹。图 3 中电路的振荡频率为：

$$f \approx \frac{1}{2\pi\sqrt{L\dfrac{C_1C_2}{C_1+C_2}}} \qquad (2)$$

式中，f 与电路中 L、C1、C2 有关。

当电感 L 数值变化时，f 也会相应改变。同样当电容容值发生变化时，f 也随之变化。一般情况下，电容值随环境温度变化而变化，因此振荡频率 f 也随温度变化而变化。

2．5 检测电路频率算法

因为检测电路中信号频率随时在改变，这为检测机动车辆等金属物体带来一定困难，尤其在环境温度急剧变化时，信号自身频率值会大幅变化。高温时在室外环境所采集的振荡电路数据分析得出：f 值在 1 h 内随温度变化几百赫兹，测量期间没有金属物体靠近。因此该设计采用基准动态改变方法。具体计算方法如下：设定 fz 为基准频率;fc 为参与计算和判断的采集频率；f 为实际采集频率。m 和 n 为滤波因子。系统没有上电时,fz 初始化值为 0；上电后，把第 1 次采集到的频率 f 作为 fz 的初始值，随后定时更换 fz 值。

先把实际采集到的频率 f 按式(2)进行一阶滤波处理，然后计算 fc 的值：

$$f_c(n)=f\frac{n}{m}+f_c(n-1)\frac{m-n}{m} \quad (3)$$

再计算基准频率 f_z：

$$f_z(n)=f_c\frac{n}{m}+f_z(n-1)\frac{m-n}{m} \quad (4)$$

式(3)、式(4)中的滤波因子 m，n 通过试验获得。当 f 值快速变化时滤波结果及时跟进且数据变化越快，灵敏度越高。在检测车辆时需定时更换 fz，根据室外温度变化更换时间。

当有金属物体经过线圈时，采集的频率值为 f，而这时基准频率为 fz。判断车辆的算法是南式(3)得出 fc，再由式(5)得：

$$\Delta f=f_c-f_z \quad (5)$$

通过判断△f 是否在同定范围中得出车辆经过情况。该范同是通过大量实验得出的。具体 CPU 算法流程见图 4。

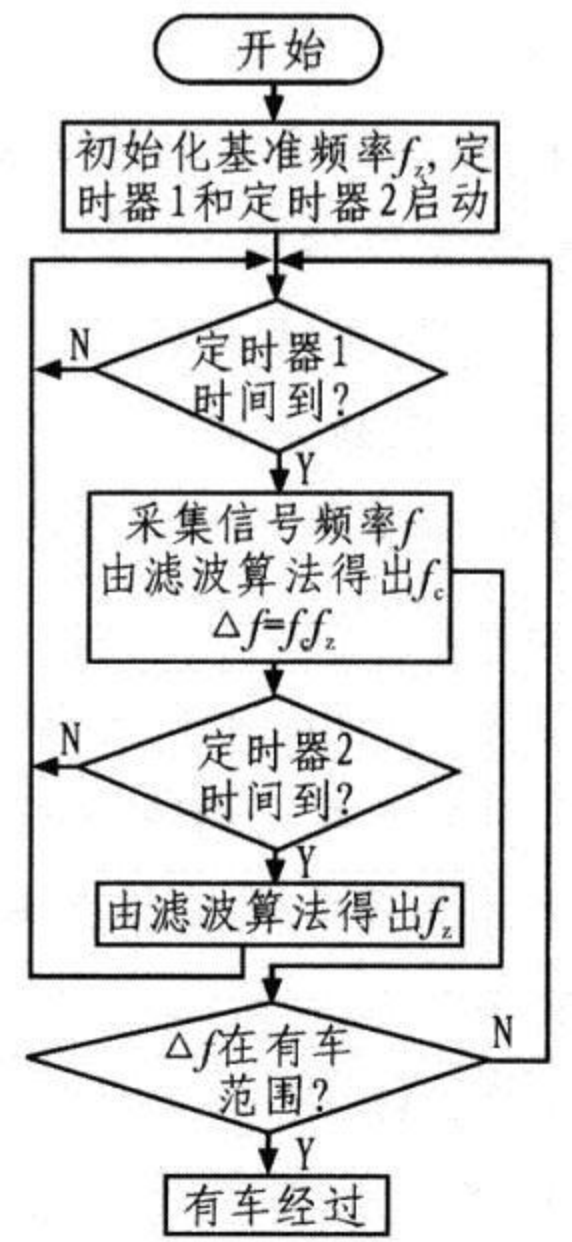


图 4　CPU 算法流程

3 实验及结果分析

该算法是经过大量验证试验而得出的。在室内，试验采用 45 cm×45 cm 的线圈(匝数 n=12)。模拟机动车辆为一辆长 1．2 m，宽 0．8 m 的铁皮小推车。当小车经过线圈时，采集频率 fc 与基准频率 fz 的差值约 400 Hz，CPU 能够准确判断出有小车经过。在室外试验则是把线圈埋入公路地面下，当汽车经过线圈时，采集频率 fc 与基准频率 fz 差值约 400～2 000 Hz。这个差值随着车辆型号不同和车辆底盘的高低而改变，而 CPU 也能准确判断

出有机动车辆经过。通过室内和室外大量试验，及时调整了滤波算法中的滤波因子，提高了检测灵敏度，使其能够满足不同车辆的需求。

4 结束语

采用滤波算法并实时更新振荡电路的基准频率可减少电路频率变化对于车辆检测的干扰。这里采用 c8051F310 设计的山路转弯预防警示系统现已安装在盘山路上，该系统能够准确检测出车辆，并发送警示信息。同时设计中充分考虑到环境因素和维护的不便，设计有上位机监控系统。因此，该系统结构简单，性能可靠，价格低廉，已引起交通部门的广泛关注。

防脉冲干扰移动平均值法数字滤波器的 C 语言算法

在许多的数据采集系统中，现场的强电设备较多，不可避免地会产生尖脉冲干扰,这种干扰一般持续时间短，峰值大，对这样的数据进行数字滤波处理时，仅仅采用算术平均或移动平均滤波时，尽管对脉冲干扰进行了 1/n 的处理，但，其剩余值仍然较大。这种场合最好的策略是：将被认为是受干扰的信号数据去掉，这就是防脉冲干扰平均值滤波法的原理。

防脉冲干扰平均值滤波法的算法是：对连续的 n 个数据进行排序，去掉其中最大和最小的 2 个数据，将剩余数据示平均值。

在一般 8051 单片机的应用中为了加快数据处理速度，n 可以取值 6。而对于具有较快速度的处理器，则 n 值可以适当取大一些。但最好是 n=2^k+2， k 为整数,因为这样在求平均值 average=SUM/(n-2)=SUM/2^k 时，可以写成 average=SUM>>k，用移位的方法，可以加快处理速度。

上述算法显然还存在一个不足之处，就是每采集一个数据就要进行一次排序，这样会大量占用系统宝贵的时间。这可以通过存储当前数据中的最大值和最小值来改进。具体做法是：

系统中用两个变量来存储当前 n 个数据的最大值和最小值在这个数组中的偏移量(也就是数组下标，存储数组下标而直接不存储数据本身是因为：在一般的系统中，n 不会超无符号短整形的表示范围，因此用一个 char 形变量就可以存储了，而如果直接存储数据本身，则许多情况下要用 int 形变量，甚至更长的类型)。这样只要在当前输入的数据将要覆盖的数据正好是当前的最大值或最小值时才在下个数组中查找最大值或最小值,而其他情况下则只要将输入的数据与最大值和最小值比较就可以修改下最大值和最小值了，而且不用进行数据排序。

这个算法很简单，下面是对应的 C 语言代码实现，可以很方便的应用的具体的 51 单片机，或其他处理器上，只须做少量的修改。

```
#i nclude"stdio.h"
#define  dtype unsigned  int // 采集数据的数据类型
#define  uint8 char

#define  LEN  6  //移动算术平均的个数+2=SHIFT<<2+2
#define  SHIFT  2  //2^SHIFT

uint8  pdata;   //移动指针
uint8  pmax,pmin;   //记录数据表中最大值和最小值的位置,
     //在一般的数据采集系统中,数据的长度>=8,
     //因此用指针记录而不是直接记录最大值和最小值
dtype  datas[LEN];

dtype  szlb(dtype  _data)
{
```

```
/*****************************/
/* 在调用此子程序前必须对 */
/* pdata,datas[]数组, */
/* pmax,pmin 进行初始化 */
/*****************************/

 uint8 i;
 dtype average=0; //清零，用来计算平均值
 pdata=(pdata+1)%LEN; //指针下标在 0 到 LEN-1 上滑动
 datas[pdata]=_data; //采样所得数据存入数据表中
 for(i=0;i<LEN;i++)
        average+=datas[i]; //求所有数据总和

/*******去除被认为是脉冲的数据******/
 if(_data>datas[pmax])
     pmax=pdata;  //得到最大值的指针
 else if(_data<datas[pmin])
     pmin=pdata;  //得到最小值的指针
 if(pdata==pmax)  //如果当前输入值将存入当前最大值的位置时
 {    //由以上方法将不可行，必须从其他位置中查找极值
     for(i=0;i<LEN;i++)
         if(datas[i]>datas[pmax])
         pmax=i;
 }
 else if(pdata==pmin)//如果当前输入值将存入当前最大值的位置时
 {    //由以上方法将不可行，必须从其他位置中查找极值
    for(i=0;i<LEN;i++)
         if(datas[i]<datas[pmin])
         pmin=i;
 }
 average=average-datas[pmax]-datas[pmin];//减去脉冲

 return (average>>SHIFT);    //求算术平均值
}

/******以下是在 VC＋＋6.0 环境下运行的测试程序**/

/***通过手动输入来模拟数据采集过程****/

void main()
{
 uint8 i;
 dtype _data;
 pdata=0;
 pmax=0;
```

```
 pmin=0;
 for(i=0;i<LEN;i++)
   datas[i]=0;
 printf("数据：                    最大      最小\n");
 while(1)
 {
      scanf("%u",&_data);
      szlb(_data);
      for(i=0;i<LEN;i++)
      printf("%-3u   ",datas[i]);
      printf("     %-3u      %-3u",datas[pmax],datas[pmin]);
      printf("\n");
 }
}


uchar  ReadADCVal()
{
//   static  uchar  num=0;
   static  uchar  cVin[7]={0,0,0,0,0,0,0};//220,220,220,220,220,220,220};//设输入电压缓冲区
   //uchar  i,j,v,t;
   uchar  i,j,v,t;
   if(bAdcVOk)
   {   iVtmp=iAdcVal;          //从 ADC 中读出数据
       iVtmp/=(5000/220);
      if(iVtmp>255)   v=255;
      else  v=(char)iVtmp;

      for(i=0;i<7;i++)
      {   if(v<=cVin[i])
            break;
      }
      if(i<=3)  //小于中值
      {   for(j=i;j<7;j++)
        {   t=cVin[j];
            cVin[j]=v;
            v=t;
        }
      }
      else  //大于中值
      {   for(j=1;j<i;j++)
        {   cVin[j-1]=cVin[j];
            cVin[j]=v;
        }
```

```
        }
        bAdcVOk=0;
    }
    iVtmp=cVin[2];
    iVtmp+=cVin[3];
    iVtmp+=cVin[4];
    iVtmp/=3;

    return   (char)iVtmp;//
}
```

对于单片机采集回来的信号，往往存在干扰信号，必须去掉，我们可以通过软件抗干扰的滤波方法来处理，常用的方法主要有以下 10 种，在这里和大家分享下：

1、限幅滤波法（又称程序判断滤波法）

A、方法：

根据经验判断，确定两次采样允许的最大偏差值（设为 A）

每次检测到新值时判断：

如果本次值与上次值之差<=A,则本次值有效

如果本次值与上次值之差>A,则本次值无效,放弃本次值,用上次值代替本次值

B、优点：

能有效克服因偶然因素引起的脉冲干扰

C、缺点

无法抑制那种周期性的干扰

平滑度差

2、中位值滤波法

A、方法：

连续采样 N 次（N 取奇数）

把 N 次采样值按大小排列

取中间值为本次有效值

B、优点：

能有效克服因偶然因素引起的波动干扰

对温度、液位的变化缓慢的被测参数有良好的滤波效果

C、缺点：

对流量、速度等快速变化的参数不宜

3、算术平均滤波法

A、方法：

连续取 N 个采样值进行算术平均运算

N 值较大时：信号平滑度较高，但灵敏度较低

N 值较小时：信号平滑度较低，但灵敏度较高

N 值的选取：一般流量，N=12；压力：N=4

B、优点：

适用于对一般具有随机干扰的信号进行滤波

这样信号的特点是有一个平均值，信号在某一数值范围附近上下波动

C、缺点：

对于测量速度较慢或要求数据计算速度较快的实时控制不适用

比较浪费 RAM

4、递推平均滤波法（又称滑动平均滤波法）

A、方法：

把连续取 N 个采样值看成一个队列

队列的长度固定为 N

每次采样到一个新数据放入队尾,并扔掉原来队首的一次数据.(先进先出原则)

把队列中的 N 个数据进行算术平均运算,就可获得新的滤波结果

N 值的选取：流量，N=12；压力：N=4；液面，N=4~12；温度，N=1~4

B、优点：

对周期性干扰有良好的抑制作用，平滑度高

适用于高频振荡的系统

C、缺点：

灵敏度低

对偶然出现的脉冲性干扰的抑制作用较差

不易消除由于脉冲干扰所引起的采样值偏差

不适用于脉冲干扰比较严重的场合

比较浪费 RAM

5、中位值平均滤波法（又称防脉冲干扰平均滤波法）

A、方法：

相当于“中位值滤波法”+“算术平均滤波法”

连续采样 N 个数据，去掉一个最大值和一个最小值

然后计算 N-2 个数据的算术平均值

N 值的选取：3~14

B、优点：

融合了两种滤波法的优点

对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差

C、缺点：

测量速度较慢，和算术平均滤波法一样

比较浪费 RAM

6、限幅平均滤波法

A、方法：

相当于“限幅滤波法”+“递推平均滤波法”

每次采样到的新数据先进行限幅处理，

再送入队列进行递推平均滤波处理

B、优点：

融合了两种滤波法的优点

对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差

C、缺点：

比较浪费 RAM

7、一阶滞后滤波法

A、方法：

取 a=0~1

本次滤波结果=（1-a）*本次采样值+a*上次滤波结果

B、优点：

对周期性干扰具有良好的抑制作用

适用于波动频率较高的场合

C、缺点：

相位滞后，灵敏度低

滞后程度取决于 a 值大小

不能消除滤波频率高于采样频率的 1/2 的干扰信号

8、加权递推平均滤波法

A、方法：

是对递推平均滤波法的改进，即不同时刻的数据加以不同的权

通常是，越接近现时刻的数据，权取得越大。

给予新采样值的权系数越大，则灵敏度越高，但信号平滑度越低

B、优点：

适用于有较大纯滞后时间常数的对象

和采样周期较短的系统

C、缺点：

对于纯滞后时间常数较小，采样周期较长，变化缓慢的信号

不能迅速反应系统当前所受干扰的严重程度，滤波效果差

9、消抖滤波法

A、方法：

设置一个滤波计数器

将每次采样值与当前有效值比较：

如果采样值＝当前有效值，则计数器清零

如果采样值<>当前有效值，则计数器+1，并判断计数器是否>=上限 N(溢出)

如果计数器溢出,则将本次值替换当前有效值,并清计数器

B、优点：

对于变化缓慢的被测参数有较好的滤波效果,

可避免在临界值附近控制器的反复开/关跳动或显示器上数值抖动

C、缺点：

对于快速变化的参数不宜

如果在计数器溢出的那一次采样到的值恰好是干扰值,则会将干扰值当作有效值导

入系统

10、限幅消抖滤波法

A、方法：

相当于“限幅滤波法”+“消抖滤波法”

先限幅,后消抖

B、优点：

继承了“限幅”和“消抖”的优点

改进了“消抖滤波法”中的某些缺陷,避免将干扰值导入系统

C、缺点：

对于快速变化的参数不宜