PID 控制

智能车程序处理对于数据的处理可简化成这样，其中单片机就是智能车的核心。

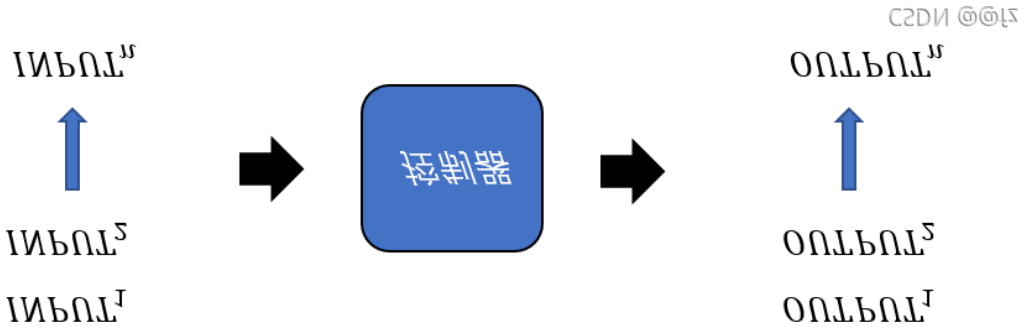
单片机主要完成:

·将传感器采集的数据进行处理

·自动控制算法

·输出控制信号

控制算法对小车的性能起了决定性的作用。



PID 控制，作为一种控制方法被广泛运用在小车控制中。在大多时候你只要会用 pid、会调 pid。

1.什么是 PID

PID 的全称为比例积分微分控制，P 为比例，I 为积分，D 为微分。PID 往往都是应用于惰性系统，所谓惰性系统，就是变化较慢且无法精确控制和调节的对象。最经典的控制对象就为温度的控制。PID 控制分增量式和位置式两种。

比例调节（P）作用：

是按比例反应系统的偏差， 系统一旦出现了偏差， 比例调节立即产

生调节作用用以减少偏差。 比例作用大， 可以加快调节， 减少误差， 但是过大的比例，使系统的稳定性下降， 甚至造成系统的不稳定。

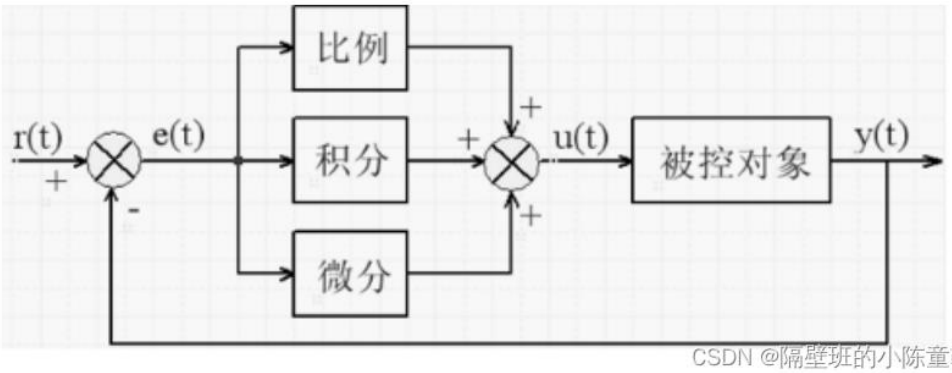
积分调节（I） 作用：

是使系统消除稳态误差， 提高无差度。 因为有误差， 积分调节就进行， 直至无差， 积分调节停止， 积分调节输出一常值。 积分作用的强弱取决与积分时间常数 Ti， Ti 越小， 积分作用就越强。 反之 Ti 大则积分作用弱， 加入积分调节可使系统稳定性
下降， 动态响应变慢。 积分作用常与另两种调节规律结合， 组成 PI 调节器或 PID 调节器。

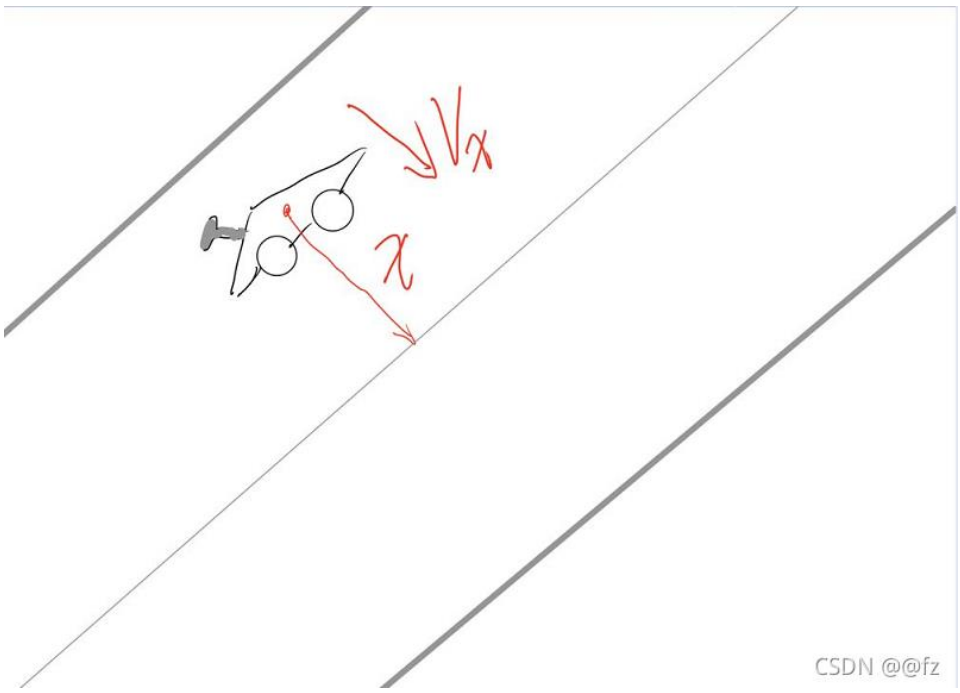
微分调节（D） 作用：

微分作用反映系统偏差信号的变化率， 具有预见性， 能预见偏差变化的趋势， 因此能产生超前的控制作用， 在偏差还没有形成之前，已被微分调节作用消除。 因此， 可以改善系统的动态性能。 在微分时间选择合适情况下， 可以减少超调， 减少调节时间。 微分作用对噪声干扰有放大作用，因此过强的加微分调节， 对系统抗干扰不利。 此外， 微分反应的是变化率， 而当输入没有变化时， 微分作用输出为零。 微分作用不能单独使用， 需要与另外两种调节规律相结合， 组成 PD 或 PID 控制器。

P 可看作对当前的控制，I 看作对过去的控制，D 看作对未来的控制。



2、位置式 PID



我们以小车巡线来介绍位置式 pid。这里我们需要实现让小车一直沿着中线前进。

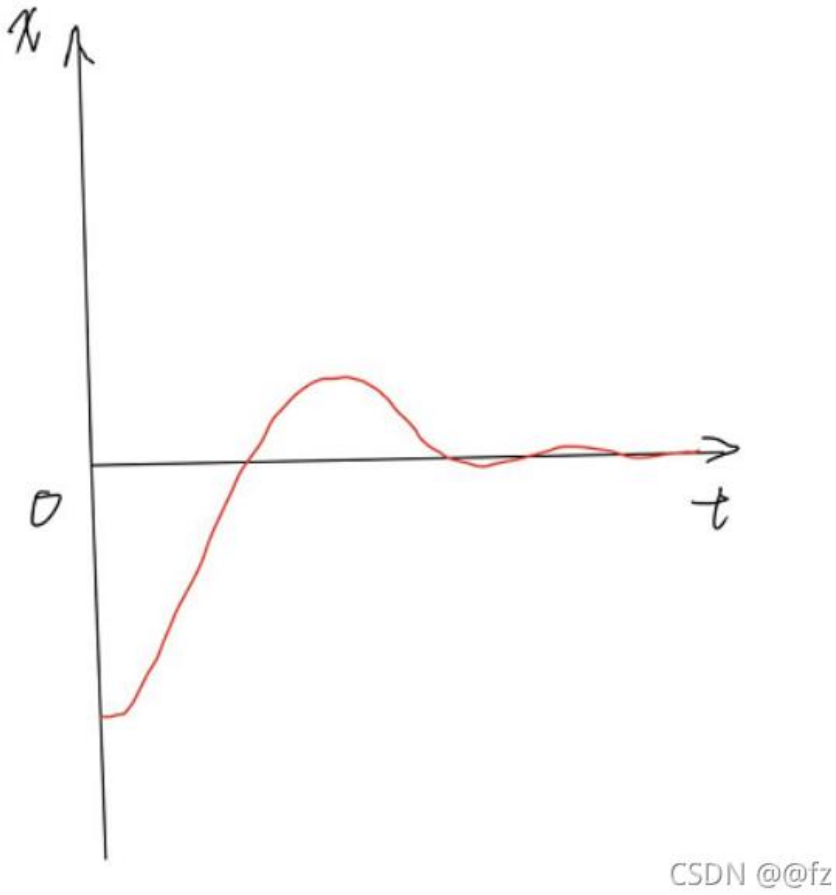
为了实现高速巡线，在巡线过程中我们需要让车快速、稳定地回到中线。

设小车方向盘的转角为Θ，通过控制Θ就可以实现小车的转向。

我们已知小车偏离中线的距离 x，当我们仅仅只考虑这一个参数的时候，我们就可以用比例（Proportion）控制。也就是离中线远方向就打的急，离中线近方向打的缓。

$$\theta = K_P x$$

以时间 t 为横坐标，以偏差 x 为纵坐标作图：

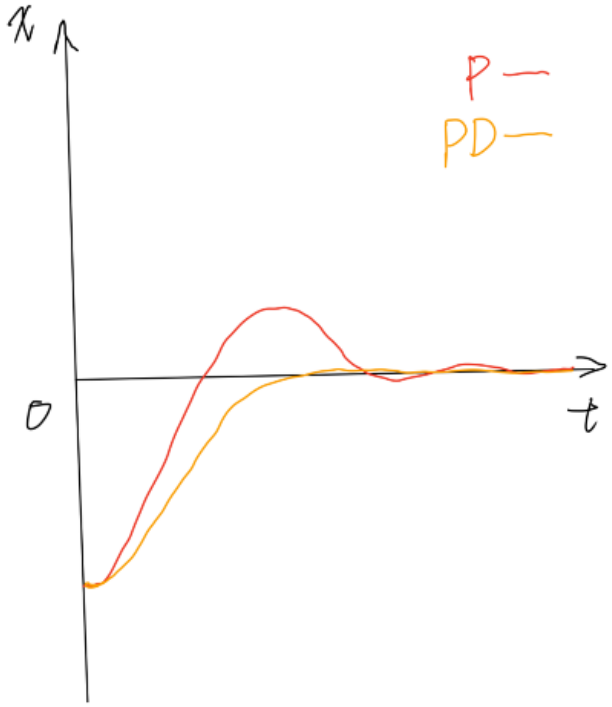


图像中的曲线斜率代表小车横向移动的速度 Vx。这里可以看到，当小车快要接近中线的时候，Vx 没有明显的下降，而是出现的一个控制滞后的现象。这其实是因为没有考虑小车的质量，也就是质量产生的惯性使他有一个过冲现象。

为了消除这个现象我们可以引入小车的速度 Vx（即 x 的微分 Differentiation），在小车横移速度比较快的时候对 P 产生的控制量 KpX 起一个牵制的，可理解为阻尼。相当于在车横移速度比较快的时候轻点一下刹车，去消除过冲现象，踩刹车的轻重与速度成正比。

$$\theta = \mathrm{K_P x} - \mathrm{K_d}\frac{\mathrm{dx}}{\mathrm{dt}}$$

其实 PD 控制就是我们通常用来控制巡线的方法。



舵机控制一般选用位置式 PID ， 因为舵机本次的控制量输出值与上次控制量输出值关系不大，只与过去状态有关，舵机需要的是快速转到某个角度。而位置式 PID 正好不需要对控制量进行记忆，而是直接对偏差值进行计算得出期望的控制量。

部分参考 C 语言代码:

```
//PID 结构体
typedef struct
{
   float Kp;                          //比例系数 Proportional
   float Ki;                          //积分系数 Integral
   float Kd;                          //微分系数 Derivative

   float Ek;                          //当前误差
   float Ek1;                         //前一次误差 e(k-1)
   float Ek2;                         //再前一次误差 e(k-2)
   float LocSum;                      //累计积分位置
}PID_TypeDef;

//pwm=Kp*e(k)+Ki*∑e(k)+Kd[e（k）-e(k-1)]
/**********************************************
函数名称 ： PID_Loc
功    能 ： 位置式 PID
参    数 ： SetValue ------ 设置值(期望值)
```

```
                ActualValue ---  实际值(反馈值)
                PID ----------- PID 数据结构
返 回 值 :   PWM -------- PWM 输出
**********************************************/

float PID_Loc(float SetValue, float ActualValue, PID_TypeDef *PID)
{
    float PWM;                                          //PWM 输出

    PID->Ek = SetValue - ActualValue;               //计算当前误差
    PID->LocSum += PID->Ek;                             //累计误差

     PWM = PID->Kp * PID->Ek + (PID->Ki * PID->LocSum) + PID->Kd
     * (PID->Ek1 - PID->Ek);                        //位置式 PID 控制

    PID->Ek1 = PID->Ek;                                 //保存上一次偏差

    if(PWM > Servo_max)
    {
       PWM =   Servo_max;
    }
    else if(PWM < Servo_min)
    {
       PWM =   Servo_min;
    }                                                   //限幅

    return PWM;
}
```

3 增量式 PID

所谓的增量，就是本次控制量和上次控制量的差值。增量式 PID 是一种对控制量的增量进行 PID 控制的一种控制算法。

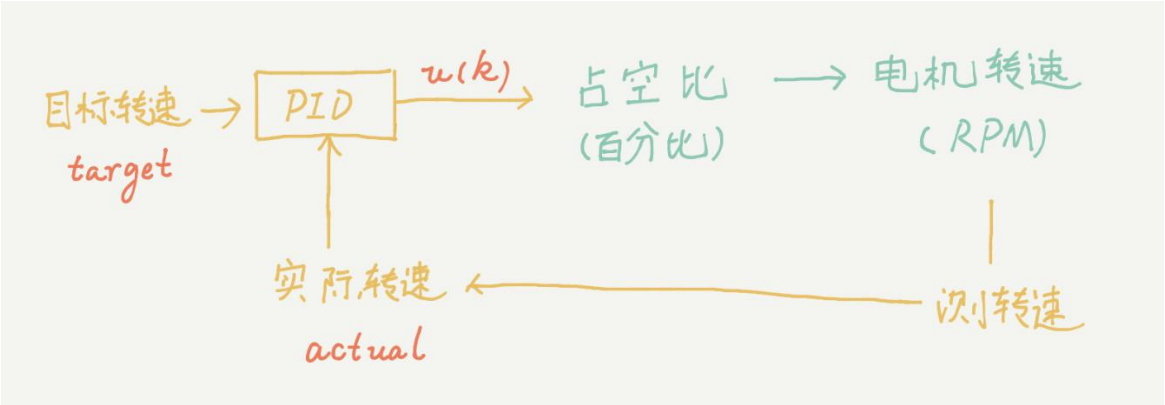
公式：

$$\Delta u[n] = K_p\{e[n] - e[n-1]\} + K_i e[n] + K_d\{e[n] - 2e[n-1] + e[n-2]\}$$

（说明：Kp->P,Ki->I,Kd->D,e数组->error数组,
e[n]->本次差值，e[n-1]->上次差值，e[n-2]->上上次差值）

电机一般选用增量式 PID，当执行机构需要的控制量是增量，而不是位置量的绝对数值时，可以使用增量式 PID 控制算法进行控制。

给定占空比 -> 测转速 -> 比较实际转速和目标转速 -> 重新调整占空比，这样的过程其实就是一个闭环控制，我们发现这个过程形成了一个回环：每次调整的占空比大小都是基于上一次结果得到的。相比开环控制，闭环控制多了信息反馈环节（测电机转速），我们根据反馈信息再做出进一步调整，接着获得调整后的反馈信息，再基于更新过的反馈信息进行新一轮的调控。



部分参考 C 语言代码：

```
//PID 结构体
typedef struct
{
   float Kp;                          //比例系数 Proportional
   float Ki;                          //积分系数 Integral
   float Kd;                          //微分系数 Derivative

   float Ek;                          //当前误差
   float Ek1;                         //前一次误差 e(k-1)
   float Ek2;                         //再前一次误差 e(k-2)
   float LocSum;                      //累计积分位置
}PID_TypeDef;

//pwm+=Kp[e（k）-e(k-1)]+Ki*e(k)+Kd[e(k)-2e(k-1)+e(k-2)]
/**********************************************
函数名称 ： PID_Inc
功　　能 ： 增量式 PID
参　　数 ： SetValue ------ 设置值(期望值)
            ActualValue --- 实际值(反馈值)
            PID ---------- PID 数据结构
返 回 值 ： PWM -------- 本次 PID 增量(+/-)
**********************************************/
float PID_Inc(float SetValue, float ActualValue, PID_TypeDef *PID)
```

```
{
    float PWM;                                          //增量

    PID->Ek = SetValue - ActualValue;                   //计算当前偏差
    PID->LocSum += PID->Ek;                             //累计误差
    PWM = PID->Kp * (PID->Ek - PID->Ek1) + PID.Ki * EK +
    PID.Kd * (PID.Ek - 2 * PID.Ek1 + PID.Ek2);
                                                        //增量式 PID 控制器
    PID->Ek2 = PID->Ek1;
    PID->Ek1 = PID->Ek;

    if(PWM > motor_Max)
    {
      PWM = motor_Max;
    }
    else if(PWM < motor_Min)
    {
      PWM = motor_Min;
    }                                                   //限幅

    return PWM;
}
```