第三课 电机控制和 PID 参数整定

1. 直流电机的控制系统

本课程我们学习基于单片机的直流电机调速系统，重点介绍 PWM 调速的基本原理和 PID 反馈闭环调速系统。直流电机具有优良的调速特性，方便、调速范围广、过载能力大、能承受频繁的冲击负载。可实现频繁的无级快速启动、制动、反转。在 HiBot 上我们采用 12V 直流有刷电机，外接高精度的增量式 AB 相编码器。

在 slam 系统中必须建立机器人的轮组运动学模型，将运动学模型放到虚拟的 2D 环境中进行控制，通过轮子反馈的位置变化和速度信息间接转化为机器人中心在 2D 坐标系中的移动。关于机器人运动学模型的搭建，我们在后面课程中解释，本节课程我们深度关注电机的速度 PID 控制和 PID 的参数整定。下面的图示 1-1-0 清晰的描述了机器人电机的速度控制和反馈系统。

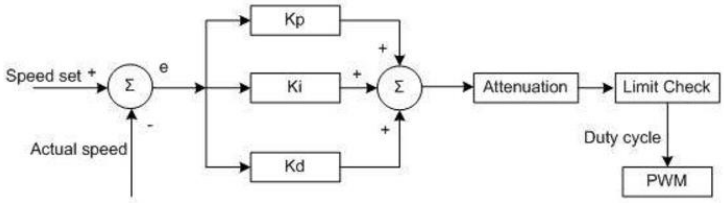


图 1-1-0 PID 调速系统

2. 电机 PWM 控制

在电机的控制领域中，不同的电机有不同的驱动方式和不同的反馈系统，但是最终的控制器模型和上图几乎接近。我们之前也了解了关于直流有刷电机的优缺点，同时清晰的明白 HiBot 的电机驱动器是由双路 PWM 调制控制的，那么如何在 STM32 上初始化 PWM 的输出配置呢？同时该如何配置 pwm 的输出频率和 PWM 的调制范围呢？
在机器人底盘代码中的 Huanyu_moto.c 文件中配置了关于电机 PWM 输出，同时也实现了两路直流电机的正反调节函数。

```
// Motor PWM output initialization, the frequency is 10.42Khz, PWM duty is 1-1000
HuanyuMoto PWM Init(1000, 10);
```

2.1. TIM1 PWM 配置

STM32 中一共有 11 个定时器，其中 TIM6、TIM7 是基本定时器；TIM2、TIM3、TIM4、TIM5 是通用定时器；TIM1 和 TIM8 是高级定时器，其中 TIM1 和 TIM8 是能够产生 3 对 PWM 互补输出，常用于三相电机的驱动，时钟由 APB2 的输出产生。TIM2-TIM5 是普通定时器，TIM6 和 TIM7 是基本定时器，其时钟由 APB1 输出产生。

若配置脉冲计数器 TIMx_CNT 为向上计数，而重载寄存器 TIMx_ARR 被配置为 N，即 TIMx_CNT 的当前计数值数值 X 在 TIMxCLK 时钟源的驱动下不断累加，当 TIMx_CNT 的数值 X 大于 N 时，会重置 TIMx_CNT 数值为 0 重新计数。而在 TIMxCNT 计数的同时，TIMxCNT 的计数值 X 会与比较寄存器 TIMx_CCR 预先存储了的数值 A 进行比较，当脉冲计数器 TIMx_CNT 的数值 X 小于比较寄存器 TIMx_CCR 的值 A 时，输出高电平（或低电平），相反地，当脉冲计数器的数值 X 大于或等于比较寄存器的值 A 时，输出低电平（或高电平）。如此循环，得到的输出脉冲周期就为重载寄存器 TIMx_ARR 存储的数值(N+1)乘以触发脉冲的时钟周期，其脉冲宽度则为比较寄存器 TIMx_CCR 的值 A 乘以触发脉冲的时钟周期，即输出 PWM 的占空比为 A/(N+1)。

```
TIM_TimeBaseStructure.TIM_Prescaler=pcs;
TIM_TimeBaseStructure.TIM_CounterMode=TIM_CounterMode_Up;
TIM_TimeBaseStructure.TIM_Period=arr;
TIM_TimeBaseStructure.TIM_ClockDivision=TIM_CKD_DIV2;
TIM_TimeBaseInit(TIM8,&TIM_TimeBaseStructure);

TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
TIM_OCInitStructure.TIM_OCIdleState = TIM_OCIdleState_Set;

TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Enable;
TIM_OCInitStructure.TIM_OCNPolarity = TIM_OCNPolarity_Low;
TIM_OCInitStructure.TIM_OCNIdleState = TIM_OCNIdleState_Reset;

TIM_BDTRInitStructure.TIM_OSSRState = TIM_OSSRState_Enable;
TIM_BDTRInitStructure.TIM_OSSIState = TIM_OSSIState_Enable;
TIM_BDTRInitStructure.TIM_LOCKLevel = TIM_LOCKLevel_1;
TIM_BDTRInitStructure.TIM_DeadTime = 0X94;
TIM_BDTRInitStructure.TIM_Break = TIM_Break_Disable;
TIM_BDTRInitStructure.TIM_BreakPolarity = TIM_BreakPolarity_Low;
TIM_BDTRInitStructure.TIM_AutomaticOutput = TIM_AutomaticOutput_Enable;
TIM_BDTRConfig(TIM8, &TIM_BDTRInitStructure);
```

图 2-1-0 定时器 1 的 pwm 输出配置

```
/*
 @ describetion: Pwm output function
 @ param: none
 @ return: none
 @ author: Xuewei Zhou
 @ date : 2019-4-17
 @ note:
 @ function void Huanyu_PWM_Output(signed short Motol_Left,signed short Moto_Right)
*/
 void Huanyu_PWM_Output(signed short Moto_Left,signed short Moto_Right)
{
    if(Moto_Left >= 0){
        TIM_SetCompare1(TIM8,0);
        TIM_SetCompare2(TIM8,fabs(Moto_Left));
    }
    else{
        TIM_SetCompare1(TIM8,fabs(Moto_Left));
        TIM_SetCompare2(TIM8,0);
    }
    if(Moto_Right >= 0){
        TIM_SetCompare3(TIM8,fabs(Moto_Right));
        TIM_SetCompare4(TIM8,0);
    }
    else{
        TIM_SetCompare3(TIM8,0);
        TIM_SetCompare4(TIM8,fabs(Moto_Right));

    }
}
```

图 2-1-0 电机的正反控制函数

3 PID 闭环控制实现

3.1 闭环控制

闭环控制系统（closed-loop control system）的特点是系统被控对象的输出（被控制量）会反送回来影响控制器的输出，形成一个或多个闭环。闭环控制系统有正反馈和负反馈，若反馈信号与系 统给定值信号相反，则称为负反馈（Negative Feedback），若极性相同，则称为正反馈，一般闭环控制系统均采用负反馈，又称负反馈控制系统。闭环控制系统的例子很多。比如人就是一个具有负反馈 的闭环控制系统，眼睛便是传感器，充当反馈，人体系统能通过不断的修正最后作出各种正确的动作。如果没有眼睛，就没有了反馈回路，也就成了一个开环控制系统。

3.2 阶跃响应

阶跃响应是指将一个阶跃输入（step function）加到系统上时，系统的输出。稳态误差是指系统的响应进入稳态后，系统的期望输出与实际输出之差。控制系统的性能可以用稳、准、快三个字 来描述。稳是指系统的稳定性（stability），一个系统要能正常工作，首先必须是稳定的，从阶跃响应上看应该是收敛的；准是指控制系统的准确性、控 制精度，通常用稳态误差来（Steady-state error）描述，它表示系统输出稳态值与期望值之差；快是指控制系统响应的快速性，通常用上升时间来定量描述。

3.3 参数整定

PID 控制器的参数整定是控制系统设计的核心内容。它是根据被 控过程的特性确定 PID 控制器的比例系数、积分时间和微分时间的大小。PID 控制器参数整定的方法很多，概括起来有两大类：一是理论计算整定法。它主要是 依据系统的数学模型，经过理论计算确定控制器参数。这种方法所得到的计算数据未必可以直接用，还必须通过工程实际进行调整和修改。二是工程整定方法，它主 要依赖工程经验，直接在控制系统的试验中进行，且方法简单、易于掌握，在工程实际中被广泛采用。一般工程整定发的口诀如下：

参数整定找最佳，从小到大顺序查。

先是比例后积分，最后再把微分加。

曲线振荡很频繁，比例度盘要放大。

曲线漂浮绕大湾，比例度盘往小扳。

曲线偏离回复慢，积分时间往下降。

曲线波动周期长，积分时间再加长。

曲线振荡频率快，先把微分降下来。

动差大来波动慢。微分时间应加长。

理想曲线两个波，前高后低 4 比 1。

一看二调多分析，调节质量不会低。

3.4 PID 控制函数的实现

首先明白我们的控制对象是电机的转速，使用 PID（比例、积分、微分）控制调节，实现函数如下，在调节过程中我们可以打开#define _Debug_LineShow_预编译选项，同时屏蔽 main 函数中的 Huanyu_SendTo_UbuntuPC();函数，然后就可以在上位机中实时查看调节的曲线效果。

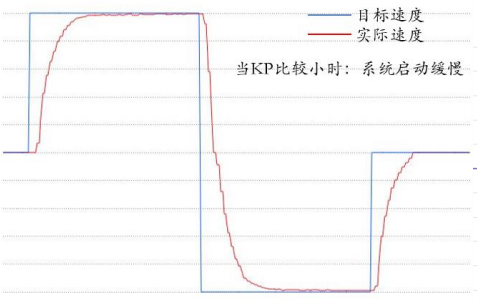
```
#define KP   225.9f
#define KI   165.0f
#define KD   110.0f


/*
 @ describetion: Moto speed PID control function
 @ param: float current_speed,float target_speed, unsigned char Moto_ID
 @ return: none
 @ author: Xuewei Zhou
 @ date : 2019-4-14
 @ function : void Huanyu_moto_Control_speed(float current_speed,float target_speed, unsigned char Moto_ID)
*/
//#define _Debug_LineShow_
void Huanyu_moto_Control_speed(float current_speed,float target_speed, unsigned char Moto_ID)
{
    float Error   = 0;
    float P_Error = 0;
    float I_Error = 0;
    float D_Error = 0;
    float add     = 0;

    if(Moto_ID == MOTO_LEFT)
    {
        Error = target_speed - current_speed;
        //Update the current one-two third-order error for the current proportional integral and differential calculation
        P_Error = Error;
        I_Error = Error - Left_moto.L_Error;
        D_Error = Error - 2*Left_moto.L_Error + Left_moto.LL_Error;

        //calculation current proportional integral and differential
        add = KP * P_Error + KI * I_Error + KD * D_Error;
        Left_moto.ESC_Output_PWM += add;

        Left_moto.LL_Error = Left_moto.L_Error;
        Left_moto.L_Error = Error;

        if(Left_moto.ESC_Output_PWM > ESC_output_PWM_LIMT)  Left_moto.ESC_Output_PWM = ESC_output_PWM_LIMT;
        else if(Left_moto.ESC_Output_PWM < -ESC_output_PWM_LIMT)    Left_moto.ESC_Output_PWM = -ESC_output_PWM_LIMT;
    }

    else if(Moto_ID == MOTO_RIGHT)
    {
        Error = target_speed - current_speed;
        //Update the current one-two third-order error for the current proportional integral and differential calculation
        P_Error = Error;
        I_Error = Error - Right_moto.L_Error;
        D_Error = Error - 2*Right_moto.L_Error + Right_moto.LL_Error;

        //calculation current proportional integral and differential
        add = KP * P_Error + KI * I_Error + KD * D_Error;
        Right_moto.ESC_Output_PWM += add;

        Right_moto.LL_Error = Right_moto.L_Error;
        Right_moto.L_Error = Error;

        if(Right_moto.ESC_Output_PWM > ESC_output_PWM_LIMT) Right_moto.ESC_Output_PWM = ESC_output_PWM_LIMT;
        else if(Right_moto.ESC_Output_PWM < -ESC_output_PWM_LIMT) Right_moto.ESC_Output_PWM = -ESC_output_PWM_LIMT;
```

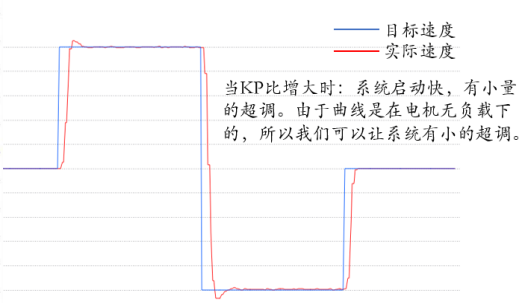
```
        #ifdef _Debug_LineShow_
        {
            send_data[0] = current_speed*1000;
            send_data[1] = target_speed*1000;
            send_data[2] = add;
            send_data[3] = Right_moto.ESC_Output_PWM;

            shanwai_send_data1((uint8_t*)&send_data,sizeof(send_data));
        }
        #endif
    }
    // moto control function
    Huanyu_PWM_Output(Left_moto.ESC_Output_PWM, Right_moto.ESC_Output_PWM);
}
```

在实际调节过程中，首先可以将 KI 和 KD 赋值为零，然后逐渐增大 KP。在调节过程中可以发现曲线有了很大的变化，电机的启停和正反切换也有了不同的效果，在负载平衡的时候，选定一组合适的参数，就可以让机器人的控制效果更加及时。在 HiBot 的 PID 调节中，我们没有加入电流的调节，也就是最大功率的启动和停止，因为 HiBot 本身的电机负载很小。有兴趣的同学可以去了解一下电机的电流、速度双闭环的实现。

在定时器中断服务函数中增加如下 3 行代码，可以在机器人上电时让电机开始运动，包含正向启动、反向切换、反向启动、停止等动作。

```
void TIM7_IRQHandler(void)
{
    if(TIM_GetITStatus(TIM7,TIM_IT_Update)==SET)
    {
        //program runing time ccount add
        (Safeware_Count == 42949672) ?(Safeware_Count=0) : (Safeware_Count++);  //时间基数常数

        if(Safeware_Count%10 == 0){Huanyu_BAT_Show(Source_Valtage);}    //电量计采样周期 1000ms

        //在pid参数整定时，打开电机运动
        if(Safeware_Count == 50)Kinematics_Positive(0.5, 0.0);      //正向启动
        if(Safeware_Count == 100)Kinematics_Positive(-0.5, 0.0);    //反向切换
        if(Safeware_Count == 150)Kinematics_Positive(0.0, 0.0);     //反向停止

        RUN_LED = ~RUN_LED;     //LED 指示周期 100ms

        Robot_Encoder_Get_CNT();    //编码器采样周期
    }
    TIM_ClearITPendingBit(TIM7,TIM_IT_Update);
}
```





4 总结

本节课程学习了如何让 STM32 输出特定频率的 PWM、电机驱动器的控制方式、PID 控制器的软件实现、PID 参数的整定等等，总之，我们需要让每一个环节都做到很细致，才能在 slam 系统中，让机器人的运动效果达到最好。