BLDC 方波速度控制

实验报告

学院 机电工程与自动化学院

姓名 方尧

学号 190410102

日期 2022 年 6 月 14 日

目录

一、BLDC 方波速度开环控制实验

1.1 程序流程图

1) 画出程序流程图，包括主循环和自定义函数，可画多个。

图1：TIM2捕获霍尔信号执行换相

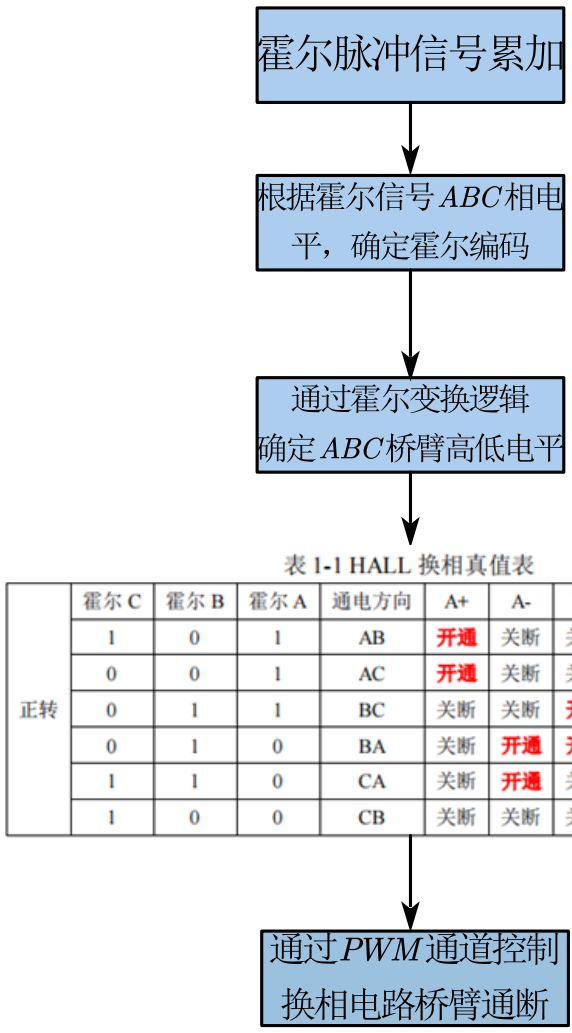
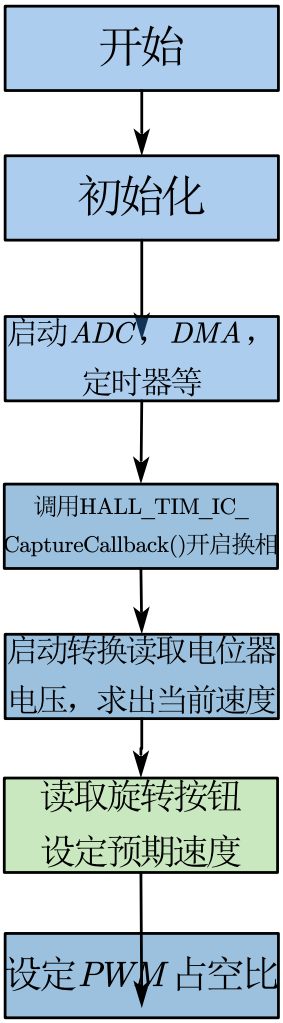


表 1-1 HALL 换相真值表

	霍尔 C	霍尔 B	霍尔 A	通电方向	A+	A-	B+	B-	C+	C-
正转	1	0	1	AB	开通	关断	关断	开通	关断	关断
	0	0	1	AC	开通	关断	关断	关断	关断	开通
	0	1	1	BC	关断	关断	开通	关断	关断	开通
	0	1	0	BA	关断	开通	开通	关断	关断	关断
	1	1	0	CA	关断	开通	关断	关断	开通	关断
	1	0	0	CB	关断	关断	关断	开通	开通	关断

图2：主程序



2) 对每个流程图进行阐释，尽量详细的描述程序设计。

首先初始化GPIO、DMA、ADC和定时器TIM2、TIM3、TIM8，启动ADC、DMA和TIM定时器，接着调用CaptureCallback()开启换向，启动电机运行。

软件启动AC转换读取电位器电压值，将电位器电压值转换成给定速度，设定相应通道的PWM占空比。

与此同时，TIM2捕获霍尔信号变化执行换向，对霍尔脉冲信号进行计数累加，读取霍尔信号A、B、C的电平，确定霍尔编码，通过IO口控制换相电路桥臂导通状态，通过PWM通道控制换相电路上桥臂导通状态。

1.2 功能代码

在此处粘贴自己编写的代码，并写好代码注释。

```
//霍尔信号
struct Hall
{
    unsigned int a;
    unsigned int b;
    unsigned int c;
};

//读取霍尔信号
void read_Hall()
{

    if (HAL_GPIO_ReadPin(HA_GPIO_Port, HA_Pin) == GPIO_PIN_SET)
        h.a = 1;
    else
        h.a = 0;
    if (HAL_GPIO_ReadPin(HB_GPIO_Port, HB_Pin) == GPIO_PIN_SET)
        h.b = 1;
    else
        h.b = 0;
    if (HAL_GPIO_ReadPin(HC_GPIO_Port, HC_Pin) == GPIO_PIN_SET)
        h.c = 1;
    else
        h.c = 0;
}
//霍尔换相逻辑
void drive_BLDC()
{
    __HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_1, ADC_Value_PWM);
    __HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_2, ADC_Value_PWM);
    __HAL_TIM_SetCompare(&htim8, TIM_CHANNEL_3, ADC_Value_PWM);
    if (h.a == 1 && h.b == 0 && h.c == 1)
    {
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_RESET);
    }
```

```
    if (h.a == 1 && h.b == 0 && h.c == 0)
    {
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_SET);
    }
    if (h.a == 1 && h.b == 1 && h.c == 0)
    {
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_SET);
    }
    if (h.a == 0 && h.b == 1 && h.c == 0)
    {
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_RESET);
    }
    if (h.a == 0 && h.b == 1 && h.c == 1)
    {
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_RESET);
    }
    if (h.a == 0 && h.b == 0 && h.c == 1)
    {
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_RESET);
```

```
    }
}
//紧急制动
void HAL_TIME8_BreakCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM8)
    {
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
        HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
        HAL_GPIO_WritePin(UL_GPIO_Port, UL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(VL_GPIO_Port, VL_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(WL_GPIO_Port, WL_Pin, GPIO_PIN_RESET);
        __HAL_TIM_DISABLE_IT(&htim8, TIM_IT_BREAK);
    }
}
//回调函数
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM2)
    {
        read_Hall();
        drive_BLDC();
        if(ttime++==100){
            Velocity=5560/tim3;
            tim3=0;
            ttime=0;
        }
    }
}
//TIM3 计数
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

    if (htim->Instance == TIM3)
    {
        tim3++;
    }
}
```

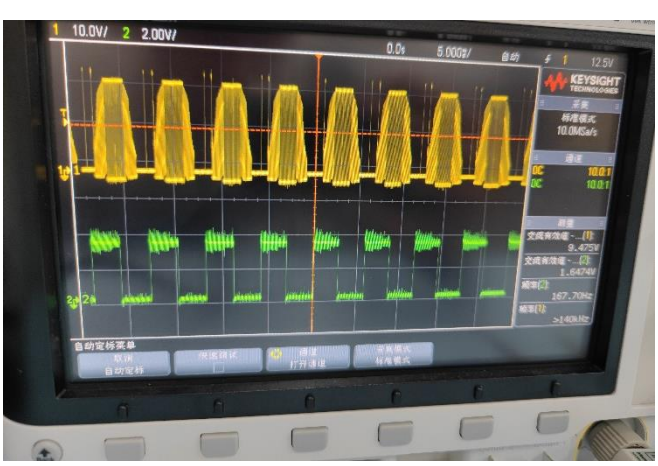
1.3 波形测量

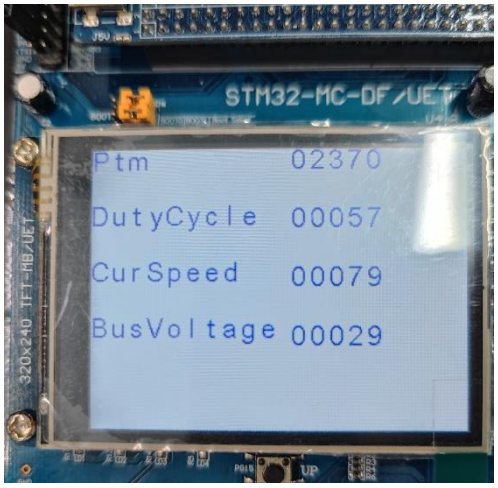
将不同 PWM 载波频率下的反电动势和 HALL 信号波形记录在实验报告中，并说说 PWM 载波频率对电机转动的影响。

1) 调整PWM载波频率，测量所得波形如下：

载波频率 f1=4200Hz

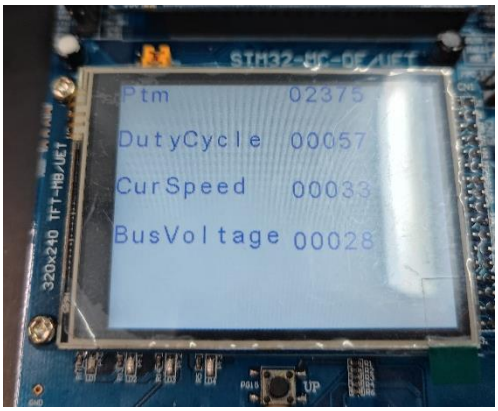
（波形记录 1）



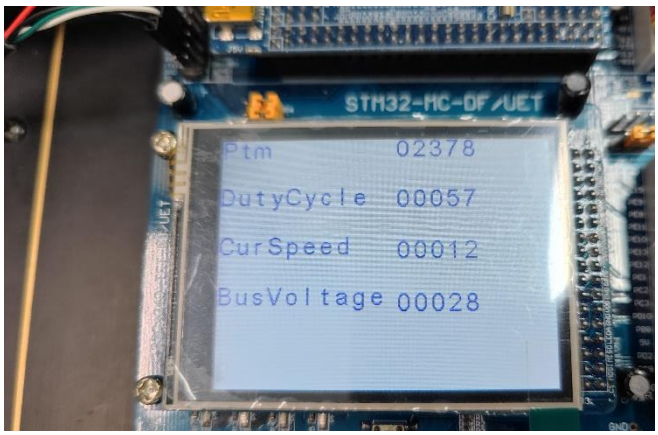


载波频率 f2=8400Hz

（波形记录 2）



载波频率 f3=16800Hz
（波形记录 3）



2) 简述载波频率对电机转动影响。

载波频率越高，一个周期内脉冲的个数就越多，电流波形的平滑性就越好，但是对其它设备的干扰也越大。载波频率越低或者设置的不好，电机就会发出难听的噪音。 通过调节载波频率可以实现系统的噪音最小，波形的平滑型最好，同时干扰也是最小的。

1.4 实验总结

阐述一下自己在开发过程中遇到的主要问题，及最终解决方法，至少三点。

1.CubeMX 未配置正确，导致无法获取电位器值，重新检查配置后发现错误。
2.霍尔换向表看错了，编写代码有问题，导致电机无法转起来，改正后功能正常。
3.LCD 显示母线电压部分出错，重新检查电路图后发现范围出错，重新编写代码后实现正常功能。

二、BLDC 方波速度闭环控制实验

2.1 程序流程图

1) 画出程序流程图，包括主循环和自定义函数，可画多个。

图1：TIM2捕获霍尔信号执行换相

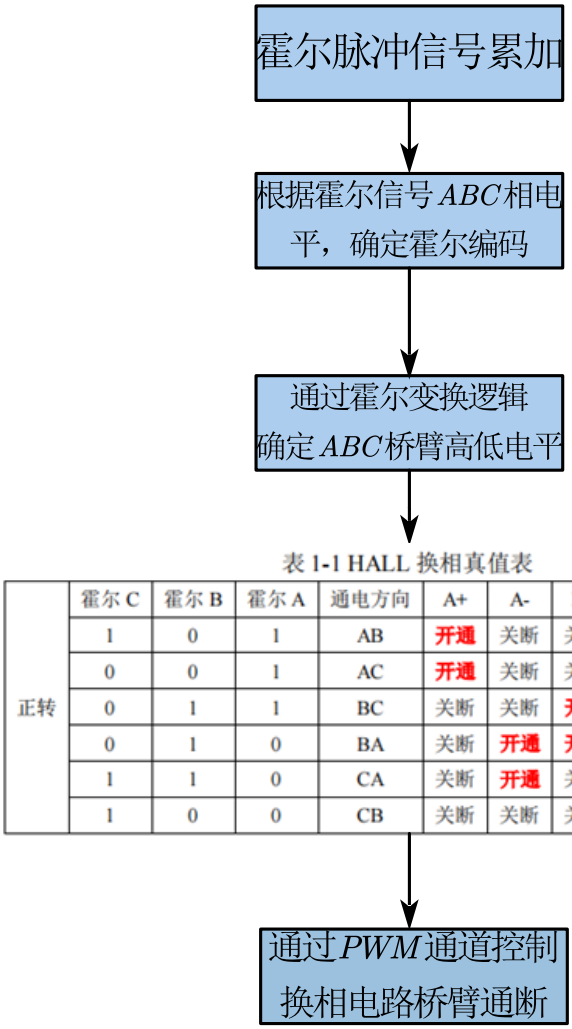
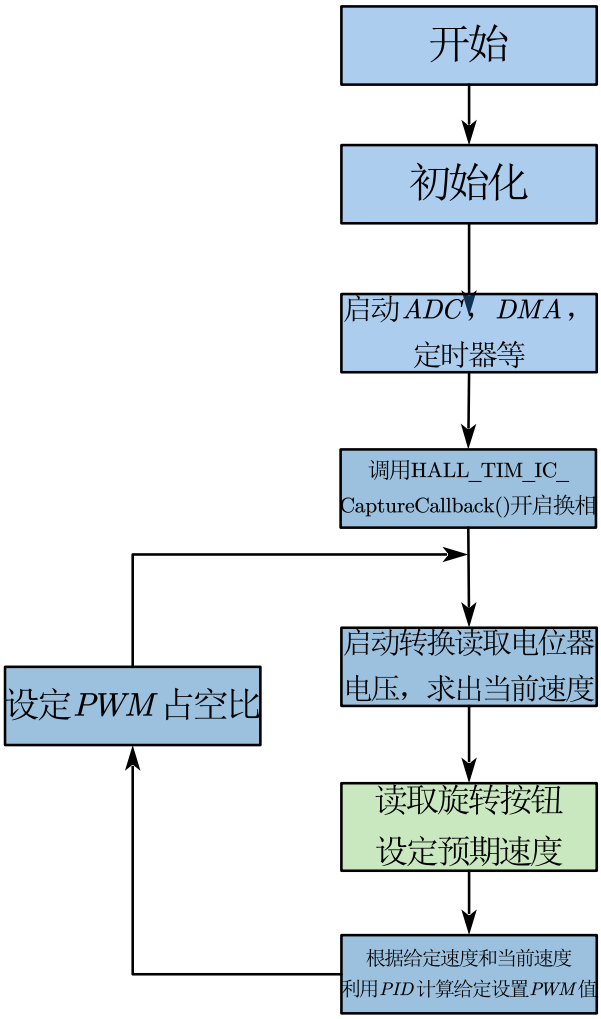


表 1-1 HALL 换相真值表

	霍尔 C	霍尔 B	霍尔 A	通电方向	A+	A-	B+	B-	C+	C-
正转	1	0	1	AB	**开通**	关断	关断	**开通**	关断	关断
	0	0	1	AC	**开通**	关断	关断	关断	关断	**开通**
	0	1	1	BC	关断	关断	**开通**	关断	关断	**开通**
	0	1	0	BA	关断	**开通**	**开通**	关断	关断	关断
	1	1	0	CA	关断	**开通**	关断	关断	**开通**	关断
	1	0	0	CB	关断	关断	关断	**开通**	**开通**	关断

图2：主程序



2) 对每个流程图进行解释，尽量详细的描述程序设计。

主要就是，相较于开环，多了一个测量反馈（读取当前速度和期望设定速度），用PID计算PWM调整值，作用在原PWM值上，形成闭环控制系统。

2.2 功能代码

在此处粘贴自己编写的代码，并写好代码注释。

```
//相比开环主要增加PID计算代码
//PID参数定义
double target=2800;
double speed=0;
double Kp=70;
double Ti=0.084;
double Td=0.001;
double T=50;
```

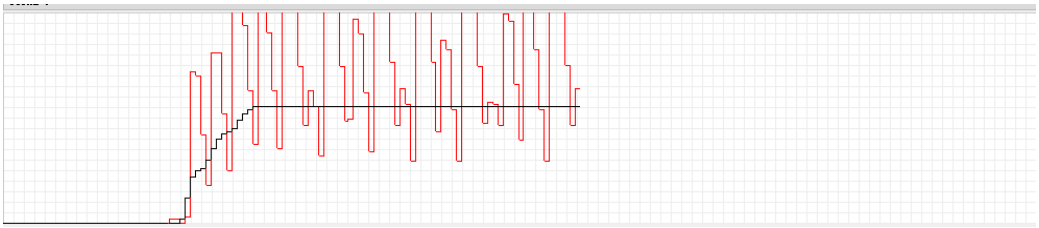
```
static double ek,ek_1,ek_2;
//PID
double calPID(void)
{
    double sT=T/1000;
    ek_2=ek_1;
    ek_1=ek;
    ek=target-speed;
    double delta;
    delta=Kp*(1+sT/Ti+Td/sT)*ek-Kp*(1+2*Td/sT)*ek_1+Kp*Td/sT*ek_2;
    //printf("%f\n",delta);
    return delta;
}
```

2.3 PID 参数调试

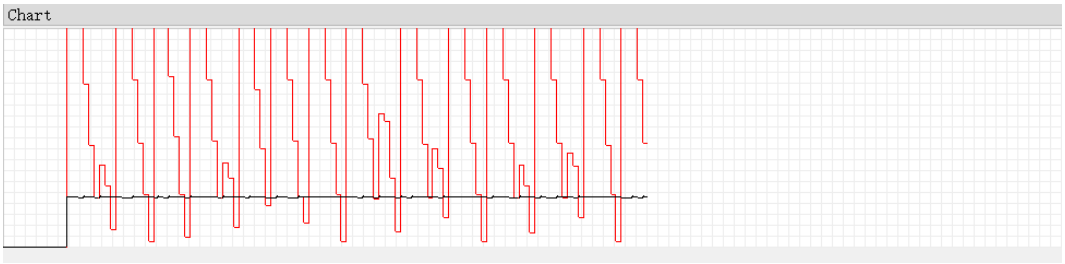
调节 PID 参数与采样周期 Ts，记录不同参数配置下，曲线的变化。选择最好的一组曲线对应的参数作为调参结果。

1) 调整PID参数与采样周期Ts，测量所得波形如下：

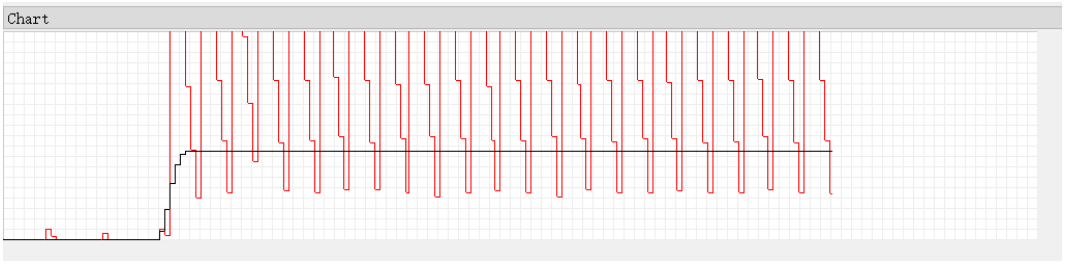
KP=10,TI=0.001,TD=0.001,Ts=50ms;



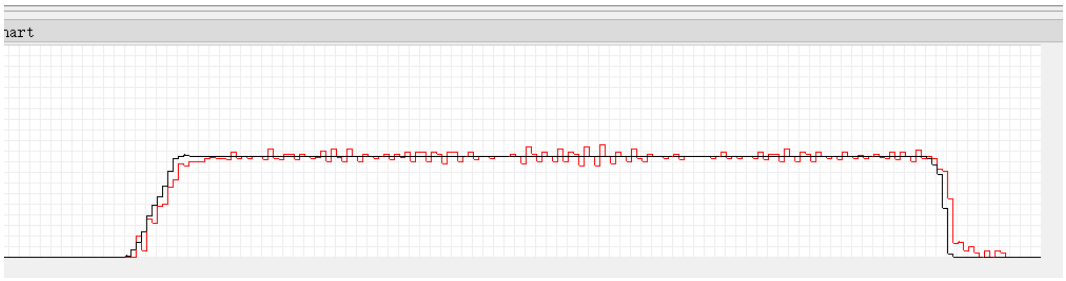
KP=80,TI=0.001,TD=0.001,Ts=50ms;



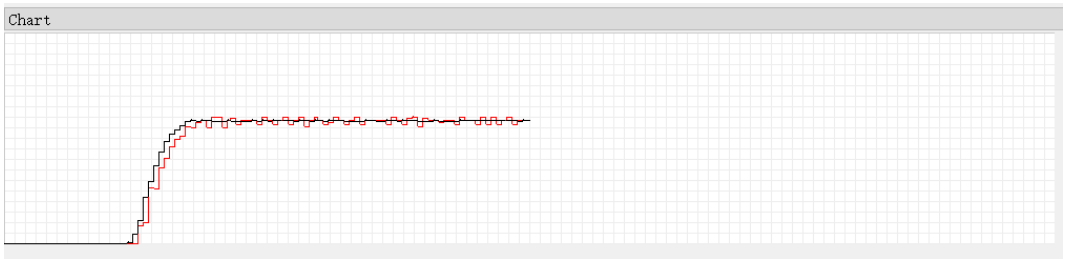
KP=70,TI=0.001,TD=0.001,Ts=50ms;



KP=70,TI=0.1,TD=0.001,Ts=50ms;



KP=70,TI=0.08,TD=0.001,Ts=50ms;



KP=70,TI=0.084,TD=0.001,Ts=50ms;

2) 最终结果

KP=70,TI=0.084,TD=0.001,Ts=50ms;

3) 简述参数调试过程

先调KP，KP调到大概等幅振荡，再调小至振荡消失（这里调至较小即结束）；

再调整TI，TI从大到小调整，直至响应良好；

这里不调整TD；

2.4 实验总结

阐述一下自己在开发过程中遇到的主要问题，及最终解决方法，至少三点。

1.波特率设置的不对，串口无法显示，改正后显示正确；
2.由于按键抖动，启停电机不灵敏，加入按键消抖后效果良好。
3.PID 转换成代码出现错误，仔细检查改正后解决了问题。