BLDC 方波速度控制

实验报告

学院 机电工程与自动化

姓名 吕家昊

学号 210320111

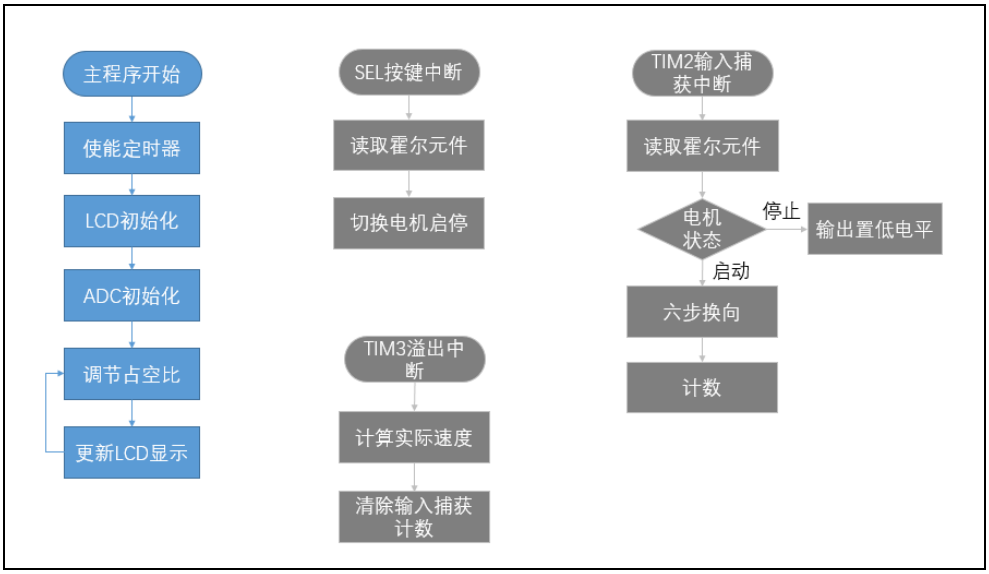
日期 2024.4.29

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。



2) 对所画流程图进行说明，尽量详细的描述程序设计。

主程序：

使能定时器包括TIM8的刹车中断、TIM2的输入捕获中断与TIM3的配置，其中TIM3为周期10ms的定时器，通过对一个周期内霍尔元件触发中断进行计数，得到电机实际运行速度。

LCD初始化包括清屏、显示各变量提示（如”ADC_VAL”）等流程。

ADC初始化为配置对应采样通道，以及设置DMA为循环模式，实现连续采样。

在while(1)中读取电位器电压，将0~4095的ADC值转换为0~80%的占空比，此后将设置的占空比与电位器电压、实际转速等一系列数据通过LCD进行显示。

SEL 按键中断：

当检测到对应引脚低电平（按键按下）时，切换电机启停状态。为确保电机正常启动，读取当前霍尔元件电平。

TIM2 输入捕获中断：

分别读取 PA15, PB3, PB10 电平即霍尔元件状态，根据六步换向表对 TIM8 的三 PWM 输出通道与负端 GPIO 输出进行控制。若电机此时设为停止状态，则将所有输出设为低电平（或关闭 PWM 输出）。

TIM3 溢出中断：

由于每次进入 TIM2 输入捕获中断时均会进行计数，因此根据一个周期内的计数，可以得到电机转过的角度，进而得到转速。

TIM3 中断：

程序会记录最后 3 次计算得到的转速，分别为 x[0]，x[1]，x[2]。设此时滤波后转速为 y[0]，使用二阶巴特沃斯低通滤波器（采样频率 fs=100Hz），得 y[0]= 1.1394y[1] - 0.4982y[2] + 0.0447x[0] + 0.0894x[1] + 0.0447x[2]。

此后通过串口发送数据，为提高数据传输准确性，数据包添加了 2 位包头。

1.2 功能代码

在此处粘贴自己编写的代码，并写好注释。

```
// 此部分代码在循环前执行
// 使能 TIM8 刹车中断
__HAL_TIM_ENABLE_IT(&htim8, TIM_IT_BREAK);
// 使能 TIM2 输入捕获
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_3);
// 读取当前霍尔元件状态
HAL_TIM_IC_CaptureCallback(&htim2);
HAL_Delay(4);
HAL_TIM_IC_CaptureCallback(&htim2);
// 使能 TIM3，进行测速
HAL_TIM_Base_Start_IT(&htim3);
TIM3->ARR = 10000 - 1;  // 设置测速周期 10ms
// LCD 初始化
STM32_LCD_Init();
LCD_SetTextColor(-1);
LCD_SetBackColor(0);
/* LCD 显示参数名与小数点代码略 */

// 测量母线与电位器电压
HAL_ADC_Start_DMA(&hadc1, volt, 1);
HAL_ADC_Start_DMA(&hadc3, adc_value, 1);

while (1)
 {
   /* 根据电位器调节占空比(0~80%)，请自行实现*/

   // LCD 显示更新
   // 对于 ADC 采样值,通过乘上 10 的指数将小数转换为整数,再利用小数点配合显示
```

```
    LCD_ShowNum(0, 160, adc_value);
    LCD_ShowNum(25, 160, (uint16_t)(adc_value / 4096.0 * 0.8 * 10000));
    LCD_ShowNum(50, 160, speed);
    // 母线电压，/7.32*1127.32 表示电阻分压
    LCD_ShowNum(75, 160, (uint16_t)(volt * 3.3 / 4096 / 7.32 * 1127.32 * 1000));
    HAL_Delay(2);
  }

// SEL 按键中断，控制电机起停
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_10)
  {
    // HAL_Delay(10);
    if (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10))
      motor_on = !motor_on;
      // 对霍尔元件进行读取，实现自启动
      HAL_TIM_IC_CaptureCallback(&htim2);
  }
}

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  // 读取霍尔元件电平，并对中断触发计数
  hall[0] = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_15);
  hall[1] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3);
  hall[2] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_10);
  ic_cnt++;

  if(motor_on){
      /* 六步换向，请自行实现 */
    }
}

void HAL_TIMEx_BreakCallback(TIM_HandleTypeDef *htim)
{
  // 刹车
  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(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
```

```
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);
    __HAL_TIM_DISABLE_IT(&htim8, TIM_IT_BREAK);
    motor_on = 0;
    state = 8;
  }
}

// 每 10ms 进入一次 TIM3 中断，利用一个测量周期内触发 TIM2 捕获中断的次数进行转速测量
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  if (htim->Instance == TIM3){
    // 1 round->6p IC_callbacks, v = (ic_cnt/6p)/100ms, set p=4
    speed = (uint16_t)(100.0 * ic_cnt / 6 / 4 / 3);
    ic_cnt = 0;
  }
}
```

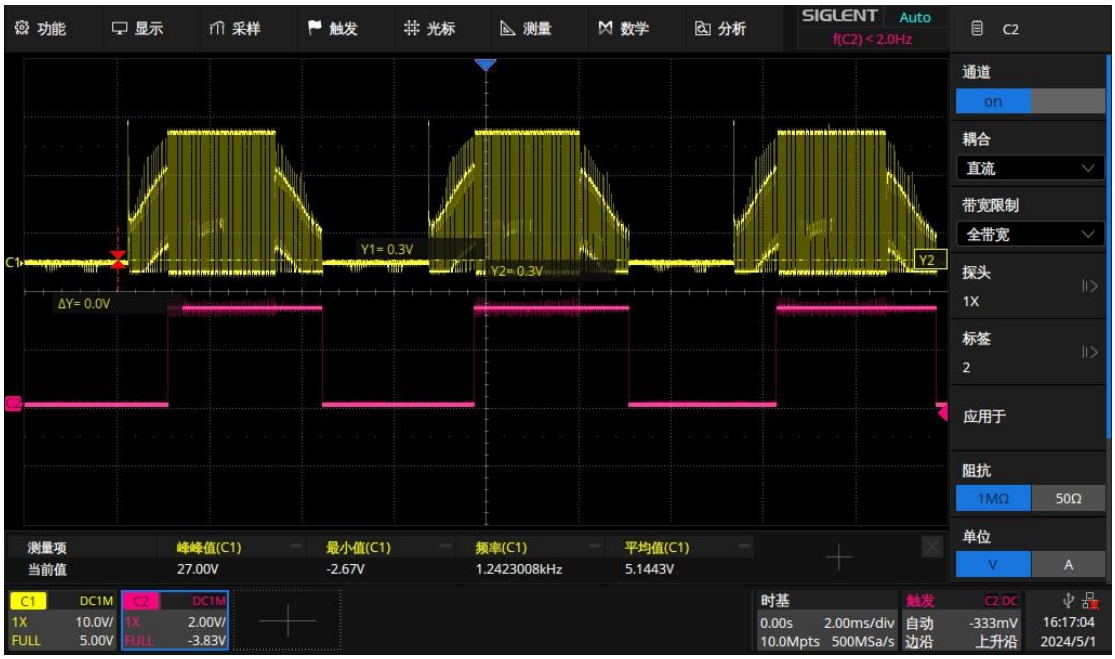
1.3 波形测量

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，并思考 PWM 载波频率对电机转动的影响。

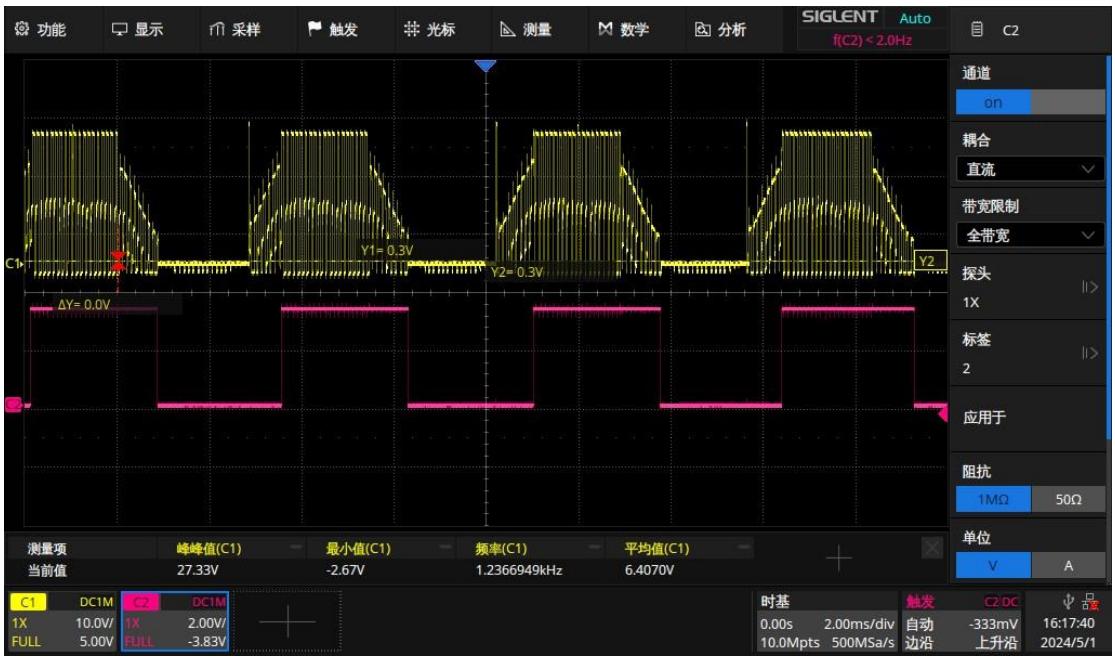
1) 调整PWM载波频率，测量所得波形如下：

黄色波形为 U 相反电动势，红色波形为霍尔元件 A 电压值。PWM 载波占空比均为 50%。

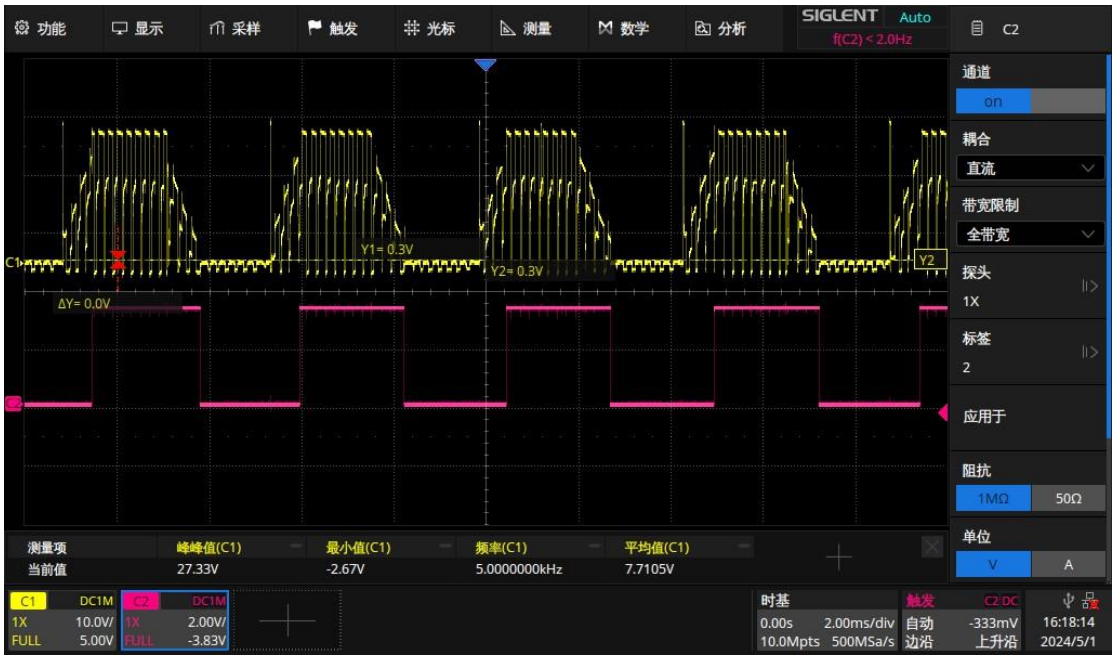
载波频率 f1=20kHz



载波频率 f2=10kHz



载波频率 f3=5kHz



2) 说说载波频率对电机转动影响。

当载波频率增加时，电机内部充放电频率更高（每一个梯形波内部反电动势的升降更密）。相同占空比下，载波频率增加时，电机转速下降。

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。

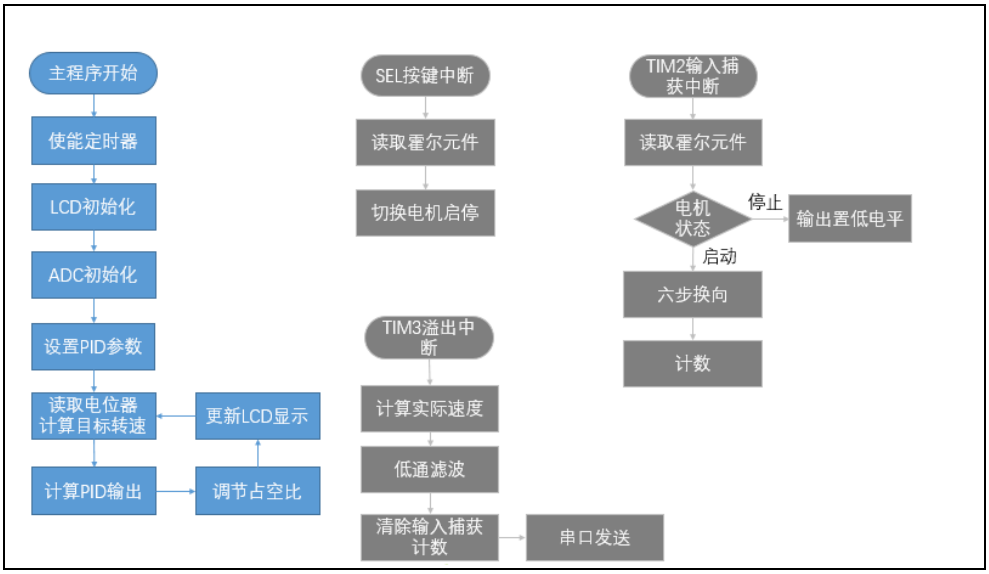
1. 完成六步换向代码编写后，按下 SEL，电机无法启动。
使用 Watch 窗口观察代码执行情况，发现代码不断 TIM8 的 Break 中断，最终发现 Cube 中 BRK Polarity 设置错误，修改为 Low 后正常运行。

2. 电机运动过程中，换向有时只出现 3 个状态，当占空比高时可能触发 Break 中断，但电机表现正常。
开始六步换向代码写在主程序中，之后将此部分代码移动到 TIM2 的输入捕获中断，代码正常运行。
猜测问题在于 main()的 while(1)中执行周期为 2ms，这段时间内电机可能触发多次中断，导致换向跳步。

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。



2) 对所画流程图进行说明，尽量详细的描述程序设计。

代码中#ifdef CLOSE_LOOP为闭环控制部分代码，通过#define CLOSE_LOOP进行开环与闭环的切换，便与调试。

以下对闭环控制中修改的部分进行描述。

主程序：

创建 PID 结构体类型，用于存放 PID 控制器的参数，以及误差、累积误差等变量，在 while(1)之前进行设置。

while(1)中，通过 ADC3 读取电位器电压值，将电压转换为 0~50rps 转速。此后利用反馈转速与目标转速的误差，计算出 PID 控制器输出，单位为占空比（0~80%）。

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。

```
TIM8->ARR = 8400 - 1;  // 168MHz / 8400 = 20kHz
 // break
 __HAL_TIM_ENABLE_IT(&htim8, TIM_IT_BREAK);
 // enable IC
 HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
 HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);
 HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_3);
 // Hall
 HAL_TIM_IC_CaptureCallback(&htim2);
 HAL_Delay(4);
 HAL_TIM_IC_CaptureCallback(&htim2);
 // speed measure
 HAL_TIM_Base_Start_IT(&htim3);
 TIM3->ARR = 5000 - 1;  // 5ms
```

```
// LCD Init
STM32_LCD_Init();
LCD_SetTextColor(-1);
LCD_SetBackColor(0);
/* LCD 显示参数名与小数点代码略 */

#ifdef CLOSE_LOOP
  // PID
  PID pid;
  /* PID 参数配置略 */
#endif

// 测量母线与电位器电压
HAL_ADC_Start_DMA(&hadc1, volt, 1);
HAL_ADC_Start_DMA(&hadc3, adc_value, 1);

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  #ifdef CLOSE_LOOP
    ref_speed = adc_value / 4096.0 * 50;
    // pid
    if(motor_on){
    /* PID 计算，请自行实现 */
    } else {
      pid.err_sum = 0;
      pid.output = 0;
    }
      // set duty cycle
      TIM8->CCR1 = (uint16_t)(TIM8->ARR * pid.output);
      TIM8->CCR2 = (uint16_t)(TIM8->ARR * pid.output);
      TIM8->CCR3 = (uint16_t)(TIM8->ARR * pid.output);
  #else
    // Duty cycle(0~80%)
    TIM8->CCR1 = (uint16_t)(TIM8->ARR * (adc_value / 4096.0 * 0.8));
    TIM8->CCR2 = (uint16_t)(TIM8->ARR * (adc_value / 4096.0 * 0.8));
    TIM8->CCR3 = (uint16_t)(TIM8->ARR * (adc_value / 4096.0 * 0.8));
```

```
    #endif

    // LCD update
    LCD_ShowNum(0, 160, adc_value);
    #ifdef CLOSE_LOOP
      // ref
      LCD_ShowNum(25, 160, (uint16_t)(ref_speed * 1000));
    #else
      LCD_ShowNum(25, 160, (uint16_t)(adc_value / 4096.0 * 0.8 * 10000));
    #endif
    LCD_ShowNum(50, 160, (uint16_t)(speed * 1000));
    LCD_ShowNum(75, 160, (uint16_t)(volt * 3.3 / 4096 / 7.32 * 1127.32 * 1000));
    HAL_Delay(2);
       }

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_10)
  {
    // HAL_Delay(10);
    if (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10))
      motor_on = !motor_on;
        HAL_TIM_IC_CaptureCallback(&htim2);
  }
}

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  hall[0] = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_15);
  hall[1] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_3);
  hall[2] = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_10);
  ic_cnt++;

  if(motor_on){
      /* 六步换向，请自行实现 */
    }
}

void HAL_TIMEx_BreakCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM8)
  {
    HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
```

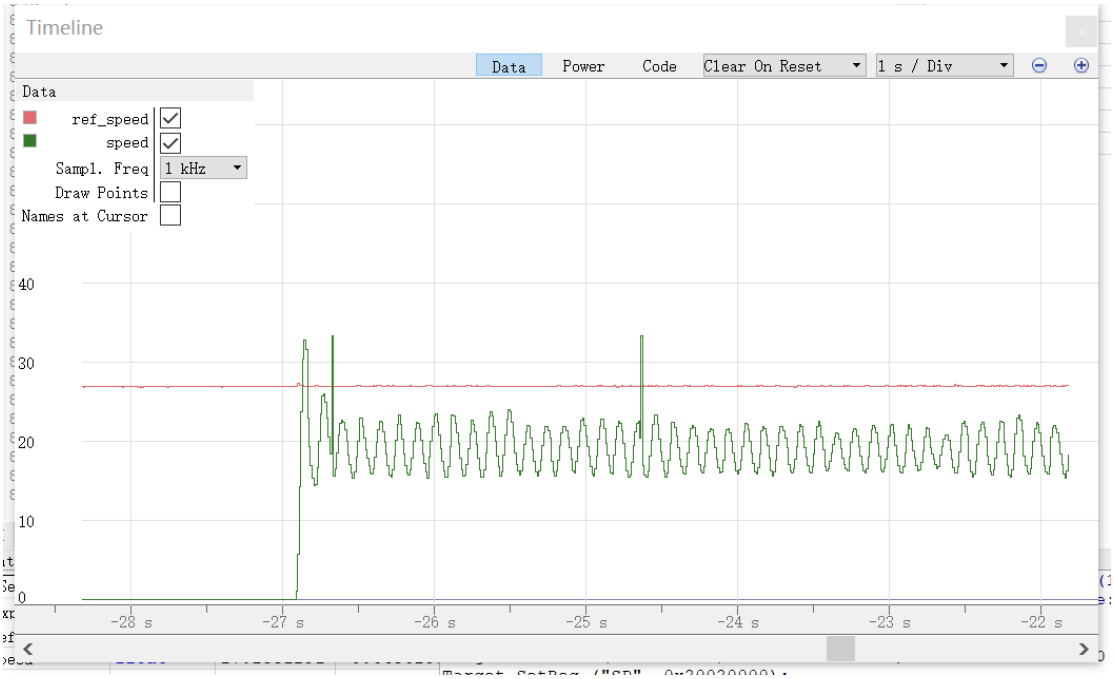
```
    HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_3);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);
    __HAL_TIM_DISABLE_IT(&htim8, TIM_IT_BREAK);
    motor_on = 0;
    state = 8;
  }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  if (htim->Instance == TIM3){
    // 1 round->6p IC_callbacks, v = (ic_cnt/6p)/5ms, set p=4
    // 采样周期建议适当增大
    // 不清楚为什么要/3，但加了之后结果正确(?)
    speed = 200.0 * ic_cnt / 6 / 4 / 3;

    /* 对反馈速度 2 阶低通滤波，代码略 */

    #ifdef CLOSE_LOOP
      uint8_t txbuffer[10];
      txbuffer[0] = 0x55;
      txbuffer[1] = 0x03;
      memcpy(txbuffer + 2, &speed, 4);
      memcpy(txbuffer + 6, &ref_speed, 4);
      HAL_UART_Transmit(&huart1, txbuffer, 10, 100);
    #endif

    ic_cnt = 0;
  }
   }
```

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。

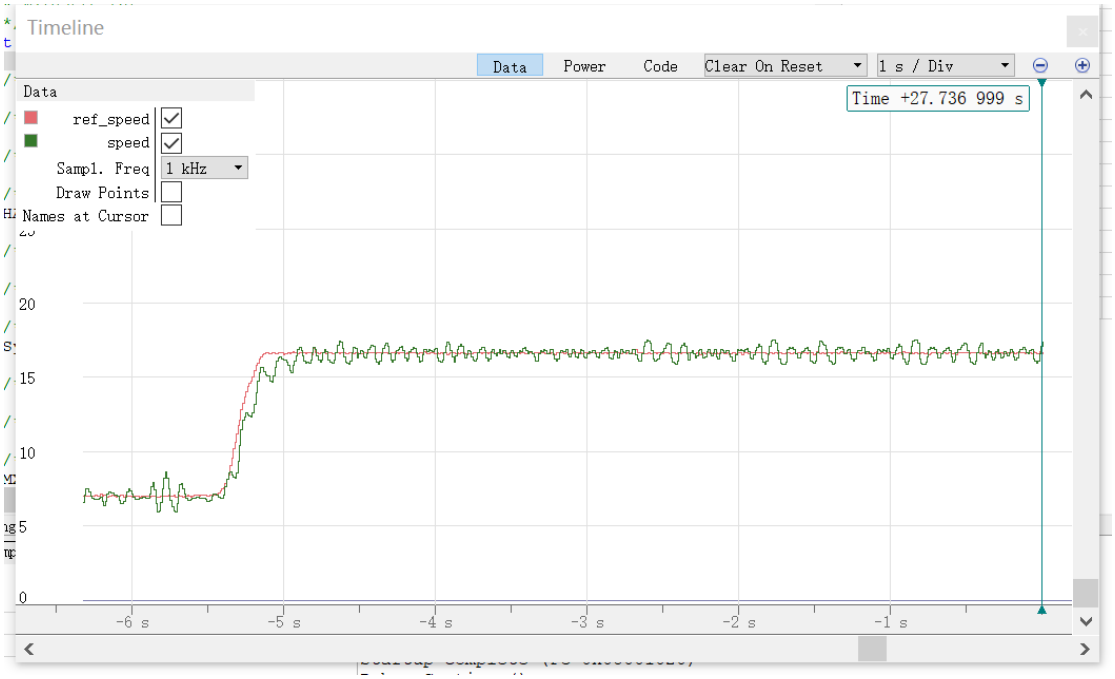
1) 调整PID参数与采样周期Ts，测量所得波形如下：

KP=0.05, KI=0 , KD=0, Ts=0.005;



KP=0.03, KI=0.0025, KD=0.000625, Ts=0.005;

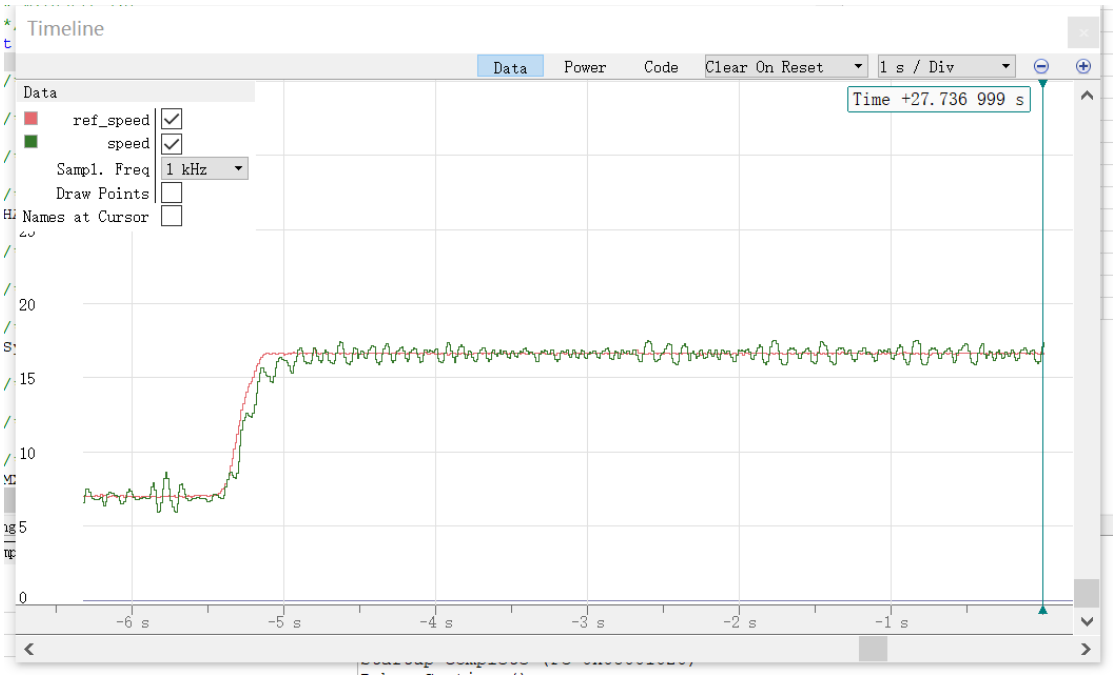
（参数略）



尽量详细的记录调参的过程！

2) 最终结果

(参数略)



3) 简述参数调试过程

首先对Kp进行调试：

由于输出单位为占空比（output<1），因此初选KP=0.001，逐渐增加KP

的数量级，发现当Kp=0.05时系统近似等幅振荡。

因此，取KP=0.6*0.05=0.03，并根据采样频率选取KI=0.0025, KD=0.000625。

考虑适当提高响应速度，微调KP，最终参数略。

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。

可以观察到，反馈速度仍有较大波动。一方面，可通过增加采样时间（即TIM3 周期），以减小噪声对速度反馈值的影响，但采样时间过长会影响系统性能与稳定性；另一方面，可对反馈速度进行滤波。

通过测试，最终选取速度测量环节采样时间 Ts=5ms，并对速度测量值进行二阶低通滤波。