**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 吴俊达

学号 210320621

日期 2024.5.6

目录

[一、BLDC方波速度开环控制实验 1](#_Toc6903)

[1.1 程序流程图 1](#_Toc28353)

[1.2 功能代码 3](#_Toc30515)

[1.3 波形测量 13](#_Toc12064)

[1.4 实验总结 14](#_Toc15747)

[二、BLDC方波速度闭环控制实验 15](#_Toc12808)

[2.1 程序流程图 15](#_Toc28378)

[2.2 功能代码 17](#_Toc15600)

[2.3 PID参数调试 17](#_Toc2554)

[2.4 实验总结 19](#_Toc28203)

# 

# 一、BLDC方波速度开环控制实验

|  |  |
| --- | --- |
| 1. LCD 初始化及显示 | 1. TIM3（测速定时器） |
| 1. ADC & DMA | 1. TIM2: HALL（中断读取霍尔状态） |
| 1. 换向（Change Direction） | 1. TIM8: BREAK（刹车中断） |
| 1. 启停（START & STOP，状态机） | 1. 增量式 PID 控制器计算 |

## 1.1 程序流程图

代码里有编号，是指各项子任务：

1. 画出程序流程图。



图1：主函数、定时器溢出中断与捕获中断逻辑



图2：霍尔传感器读取、换向逻辑、刹车中断回调、切换状态（采用按键中断回调）、停机函数逻辑



1. 对所画流程图进行说明，尽量详细的描述程序设计。

|  |
| --- |
| 已经在流程图中说明得很清楚，此处从略。 |

## 1.2 功能代码

我的开环和闭环代码是写在一起的，利用变量Closed\_loop 来区分，此变量置0时为开环，置1时为闭环。此处系统生成的代码已经删除。

代码里有编号，是指各项子任务：

|  |  |
| --- | --- |
| 1. LCD 初始化及显示 | 1. TIM3（测速定时器） |
| 1. ADC & DMA | 1. TIM2: HALL（中断读取霍尔状态） |
| 1. 换向（Change Direction） | 1. TIM8: BREAK（刹车中断） |
| 1. 启停（START & STOP，状态机） | 1. 增量式 PID 控制器计算 |

具体代码如下：

/\* USER CODE BEGIN Includes \*/

#include "stdio.h"

#include "string.h"

#include "stm32\_2.8\_lcd.h" // 引入必要的头文件

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN PV \*/

const uint8\_t Closed\_loop = 0; // 开环闭环标志，1表示闭环

uint32\_t ADC\_VOLTAGE\_32[32] = {0}; // 电位器电压接收buffer

uint32\_t BUS\_VOLTAGE\_32[32] = {0}; // 母线电压接收buffer

uint32\_t Duty\_Cycle = 0; // 占空比，最大值为TIM8最大计数值

float ADC\_VOLTAGE, BUS\_VOLTAGE, Duty\_Cycle\_Display; // 用于转换与显示的中间变量

uint8\_t HALL\_STATUS = 0; // 霍尔传感器状态

uint8\_t START\_STATUS = 0; // 电机状态，0/停机1，1/正转，2/停机2，3/反转

uint16\_t TIME\_GAP = 0; // 缓转/停转标志

uint8\_t TIME\_GAP\_CNT = 0; // 用于计算速度

uint8\_t TIME\_TURN\_CNT = 0; // 用于计算速度

// PID

float Speed = 0;

float targetSpeed = 0;

// 速度环PID参数

#define PERIOD 10 // 控制周期

float u1, ek11, ek12; // 增量式PID式中三项，分别用于记录上一控制时刻PWM、上一控制时刻偏差和上上控制时刻偏差

float velocity\_kp= 0.0003; // 比例系数

float velocity\_ti= 0.7; // 积分时间

float velocity\_td= 0; // 微分时间

uint16\_t PWM = 0; // 调速PWM值

/\* USER CODE END PV \*/

/\* USER CODE BEGIN PFP \*/

void ChangeDirection(void); // 换向

void StopMotor(void); // 停机

float pidController(void); // PID计算

/\* USER CODE END PFP \*/

int main(void)

{

/\* USER CODE BEGIN 2 \*/

// 1. LCD Initialize

STM32\_LCD\_Init();

LCD\_Clear(BackColor);

LCD\_SetTextColor(Red);

LCD\_DisplayStringLine(1,"PF9 VOLTAGE");

LCD\_DisplayStringLine(2,"BUS VOLTAGE");

if(Closed\_loop) LCD\_DisplayStringLine(3,"PWM Value");

else LCD\_DisplayStringLine(3,"Duty Cycle");

LCD\_DisplayStringLine(4,"SPEED(\*.1rps)");

LCD\_DisplayStringLine(5,"Target SPEED");

// 3. ADC & DMA

HAL\_NVIC\_DisableIRQ(DMA2\_Stream1\_IRQn); // 禁用DMA中断

HAL\_NVIC\_DisableIRQ(DMA2\_Stream0\_IRQn); // 禁用DMA中断

HAL\_ADC\_Start(&hadc1);

HAL\_ADC\_Start(&hadc3);

HAL\_ADC\_Start\_DMA(&hadc1, &BUS\_VOLTAGE\_32[0], 1); // Continuous Conversion mode, BUS VOLTAGE

HAL\_ADC\_Start\_DMA(&hadc3, &ADC\_VOLTAGE\_32[0], 1); // Continuous Conversion mode, Potentialmeter VOLTAGE

// 4. TIM2: HALL

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);

// Duty\_Cycle = 2000; // 定占空比，调试时使用

// 6. TIM8: BREAK

\_\_HAL\_TIM\_ENABLE\_IT(&htim8, TIM\_IT\_BREAK); // 使能刹车

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

ADC\_VOLTAGE = ADC\_VOLTAGE\_32[0] \* 33 / 40.96; // 电位器电压(\*0.01V)，最大值为330，即对应3.3V

BUS\_VOLTAGE = BUS\_VOLTAGE\_32[0]; // 母线电压(\*0.1V)，显示数值约二百，也就是二十多伏

// BUS\_VOLTAGE = BUS\_VOLTAGE\_32[0] \* 3.3\* 11.2732 /7.32 / 40.96;

if(Closed\_loop == 0) Duty\_Cycle\_Display = ADC\_VOLTAGE\_32[0] / 4.096; // 开环时的占空比显示，最大值为1000

// 1. LCD Display

LCD\_Draw\_NUM(24,120, (uint16\_t) ADC\_VOLTAGE); // 显示ADC采集电压

LCD\_Draw\_NUM(48,120, (uint16\_t) BUS\_VOLTAGE); // 显示母线电压

if(Closed\_loop) LCD\_Draw\_NUM(72,120, (uint16\_t) PWM); // 闭环 PWM显示，最大值8399

else LCD\_Draw\_NUM(72,120, (uint16\_t) Duty\_Cycle\_Display); // 开环 占空比显示

LCD\_Draw\_NUM(96,120, (uint16\_t) Speed); // 速度显示，单位\*0.1rps，例:显示330即为33rps

if(Closed\_loop) LCD\_Draw\_NUM(120,120, (uint16\_t)targetSpeed); // 目标速度显示，单位说明同上

LCD\_Draw\_NUM(144,120, (uint16\_t)TIME\_GAP); // 停转标志，每进入3次转动中断重置1次，若长期不重置则进入下面逻辑

if(Closed\_loop) printf("%f,%f\r\n",Speed,targetSpeed); // 闭环时，发送速度与目标速度数据

if(TIME\_GAP > 2000 && TIME\_GAP < 10000){ // 停转标志较大，尝试重启

ChangeDirection();

HAL\_Delay(4);

ChangeDirection();

}

else if(TIME\_GAP > 10000){ // STOP

HAL\_TIM\_Base\_Stop\_IT(&htim3);

TIME\_GAP = 0;

START\_STATUS = 2;

StopMotor();

}

if(START\_STATUS == 1){

LCD\_DisplayStringLine(7,"FORWARD"); // 显示正转

}

else if(START\_STATUS == 3){

LCD\_DisplayStringLine(7,"REVERSE"); // 显示反转

}

else LCD\_DisplayStringLine(7,"STOP "); // 显示停止

HAL\_Delay(200); // 每秒钟约更新5次

}

/\* USER CODE END 3 \*/

}

// 中间系统生成的时钟初始化、外部中断初始化代码略去

/\* USER CODE BEGIN 4 \*/

// 2. TIM3 & 8. PID Controller

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim){

UNUSED(htim);

if(htim->Instance == TIM3){

++TIME\_GAP; // 停转标志增加

++TIME\_GAP\_CNT;

// 定时, 计量进中断次数来求速度

if((START\_STATUS == 1 || START\_STATUS == 3) && TIME\_GAP\_CNT == 50){

Speed = 10000 / 24 / TIME\_GAP\_CNT \* TIME\_TURN\_CNT; // 用于显示与PID计算的速度，单位(\*0.1rps)，例如380代表38rps

//else Speed = 0;

TIME\_TURN\_CNT = 0;

TIME\_GAP\_CNT = 0;

}

}

if(htim->Instance == TIM4 && Closed\_loop){

targetSpeed = ADC\_VOLTAGE\_32[0] / 6; // 目标速度(\*0.1rps)，最大值4096/6

PWM = (uint16\_t) pidController(); // PID控制器

}

}

// 4. TIM2: HALL

void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim){

UNUSED(htim);

if(htim->Instance == TIM2){

++TIME\_TURN\_CNT;

if(TIME\_TURN\_CNT >= 3) TIME\_GAP = 0; // 进入hall中断3次，停转标志置0（正在旋转）

HALL\_STATUS = 0;

if(HAL\_GPIO\_ReadPin(GPIOA,GPIO\_PIN\_15)==GPIO\_PIN\_SET) //A

HALL\_STATUS |= 0x01;

if(HAL\_GPIO\_ReadPin(GPIOB,GPIO\_PIN\_3)==GPIO\_PIN\_SET) //B

HALL\_STATUS |= 0x04;

if(HAL\_GPIO\_ReadPin(GPIOB,GPIO\_PIN\_10)==GPIO\_PIN\_SET) //C

HALL\_STATUS |= 0x08;

ChangeDirection();

}

}

// 5. HALL Change Direction

void ChangeDirection(){

if(Closed\_loop) Duty\_Cycle = PWM; // Close loop

else Duty\_Cycle = ADC\_VOLTAGE\_32[0]\*2; // Open loop，最大值为4096\*乘数

// PWM计数器最大值8400时，设定乘数为2；PWM计数器最大值为4200时，设定乘数为1

if(START\_STATUS == 1){

if((HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_1, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if(!(HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_1, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);//C-

}

else if(!(HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_2, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);//C-

}

else if(!(HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_2, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_SET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if((HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_3, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_SET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if((HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_3, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

}

if(START\_STATUS == 3){

if((HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_2, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_SET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if(!(HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_3, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_SET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if(!(HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && (HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_3, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if(!(HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_1, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

else if((HALL\_STATUS & 0x08) && (HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_1, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);//C-

}

else if((HALL\_STATUS & 0x08) && !(HALL\_STATUS & 0x04) && !(HALL\_STATUS & 0x01) ){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Start(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

\_\_HAL\_TIM\_SetCompare(&htim8, TIM\_CHANNEL\_2, Duty\_Cycle);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);//C-

}

}

}

void StopMotor(void){

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_1);//A+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_2);//B+

HAL\_TIM\_PWM\_Stop(&htim8, TIM\_CHANNEL\_3);//C+

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);//A-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);//B-

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);//C-

}

// 6. TIM8: BREAK

void HAL\_TIMEx\_BreakCallback(TIM\_HandleTypeDef \*htim){

if(htim->Instance==TIM8){

StopMotor();

\_\_HAL\_TIM\_DISABLE\_IT(&htim8, TIM\_IT\_BREAK);

}

}

// 7. START AND STOP

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin){

if(GPIO\_Pin== GPIO\_PIN\_10){

for(uint32\_t i = 0; i<500000; ++i); // 软件消抖

// HAL\_Delay(10);

if(HAL\_GPIO\_ReadPin(GPIOG,GPIO\_PIN\_10)==GPIO\_PIN\_RESET){

if(START\_STATUS == 0){ // 停转1，准备正转

// 2. TIM3

HAL\_TIM\_Base\_Start\_IT(&htim3);

// 8. PWM

HAL\_TIM\_Base\_Start\_IT(&htim4);

// START

START\_STATUS = 1;

HAL\_TIM\_IC\_CaptureCallback(&htim2);

// HAL\_Delay(4);

HAL\_TIM\_IC\_CaptureCallback(&htim2);

}

else if(START\_STATUS == 1){ // 正转时按下按钮，进入停车2

HAL\_TIM\_Base\_Stop\_IT(&htim3);

HAL\_TIM\_Base\_Stop\_IT(&htim4);

TIME\_GAP = 0;

StopMotor();

START\_STATUS = 2;

}

else if(START\_STATUS == 2){ // 停车2按下按钮，准备反转

// 2. TIM3

HAL\_TIM\_Base\_Start\_IT(&htim3);

// 8. PWM

HAL\_TIM\_Base\_Start\_IT(&htim4);

// START

START\_STATUS = 3;

HAL\_TIM\_IC\_CaptureCallback(&htim2);

// HAL\_Delay(4);

HAL\_TIM\_IC\_CaptureCallback(&htim2);

}

else { // 反转按下按钮，进入停车1

HAL\_TIM\_Base\_Stop\_IT(&htim3);

HAL\_TIM\_Base\_Stop\_IT(&htim4);

TIME\_GAP = 0;

StopMotor();

START\_STATUS = 0;

}

}

}

}

// 8. Incremental PID Controller

float pidController()

{

float q0 = velocity\_kp \* (1 + PERIOD / velocity\_ti + velocity\_td / PERIOD);

float q1 = -velocity\_kp \* (1 + 2 \* velocity\_td / PERIOD);

float q2 = velocity\_kp \* velocity\_td / PERIOD;

float ek10;

ek10 = targetSpeed - Speed;

u1 = u1 + q0 \* ek10 + q1 \* ek11 + q2 \* ek12;

if (u1 > 8399)

u1 = 8399;

if (u1 <= 0)

u1 = 0;

ek12 = ek11;

ek11 = ek10;

return u1;

}

/\* USER CODE END 4 \*/

**在stm32f4xx\_hal\_msp.c中的代码：**

/\* USER CODE BEGIN Includes \*/

#include "stdio.h"

extern UART\_HandleTypeDef huart1;

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN 0 \*/

int fputc(int ch, FILE \*f){

HAL\_UART\_Transmit(&huart1, (uint8\_t \* )&ch, 1, 0xffff);

return ch;

}

int fgetc(FILE \*f){

uint8\_t ch = 0;

HAL\_UART\_Receive(&huart1, (uint8\_t \* )&ch, 1, 0xffff);

return ch;

}

/\* USER CODE END 0 \*/

## 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：

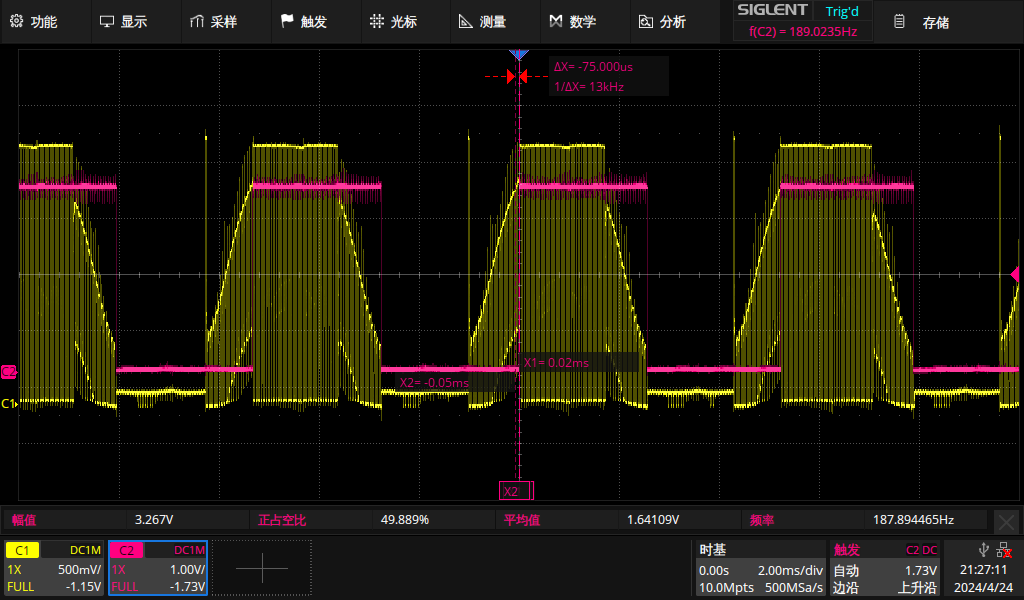
载波频率*f*1＝10kHz



载波频率*f*2＝2kHz



载波频率*f*3＝15kHz



1. 说说载波频率对电机转动影响。

载波频率过高：由于电机是感性元件，其电流变化存在过渡过程，因此若控制频率过高，则其电流尚未到达稳态时，控制模态就已改变，可能使电机转矩变小，甚至不能启动。

载波频率过低：电机控制不平顺（有段落感），且有啸叫声。

## 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. 卡死在DMA读取数据里（接收数据正确，但电机控制信号未发出），后来加入禁用DMA全局中断的两行代码，即

HAL\_NVIC\_DisableIRQ(DMA2\_Stream1\_IRQn); // 禁用DMA中断

HAL\_NVIC\_DisableIRQ(DMA2\_Stream0\_IRQn); // 禁用DMA中断

方解决问题。

2. 在按钮的外部中断中调用HAL\_Delay(4);，程序卡死，后改用软件中断

for(uint32\_t i = 0; i<500000; ++i); // 软件消抖

实现，并将代码的优化层级从-O3改为-O1防止该语句被优化。

# 

# 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。（橙色表示和开环时**不同之处**）



图1：主函数、定时器溢出中断与捕获中断逻辑





图2：霍尔传感器读取、换向逻辑、刹车中断回调、切换状态（采用按键中断回调）、停机函数、增量式PID控制函数逻辑

1. 对所画流程图进行说明，尽量详细的描述程序设计。

|  |
| --- |
| 已经在流程图中说明得很清楚，此处从略。 |

## 2.2 功能代码

我的开环和闭环代码是写在一起的，利用变量Closed\_loop 来区分，此变量置0时为开环，置1时为闭环。故代码详见上面的1.2节，此处不再重新书写。

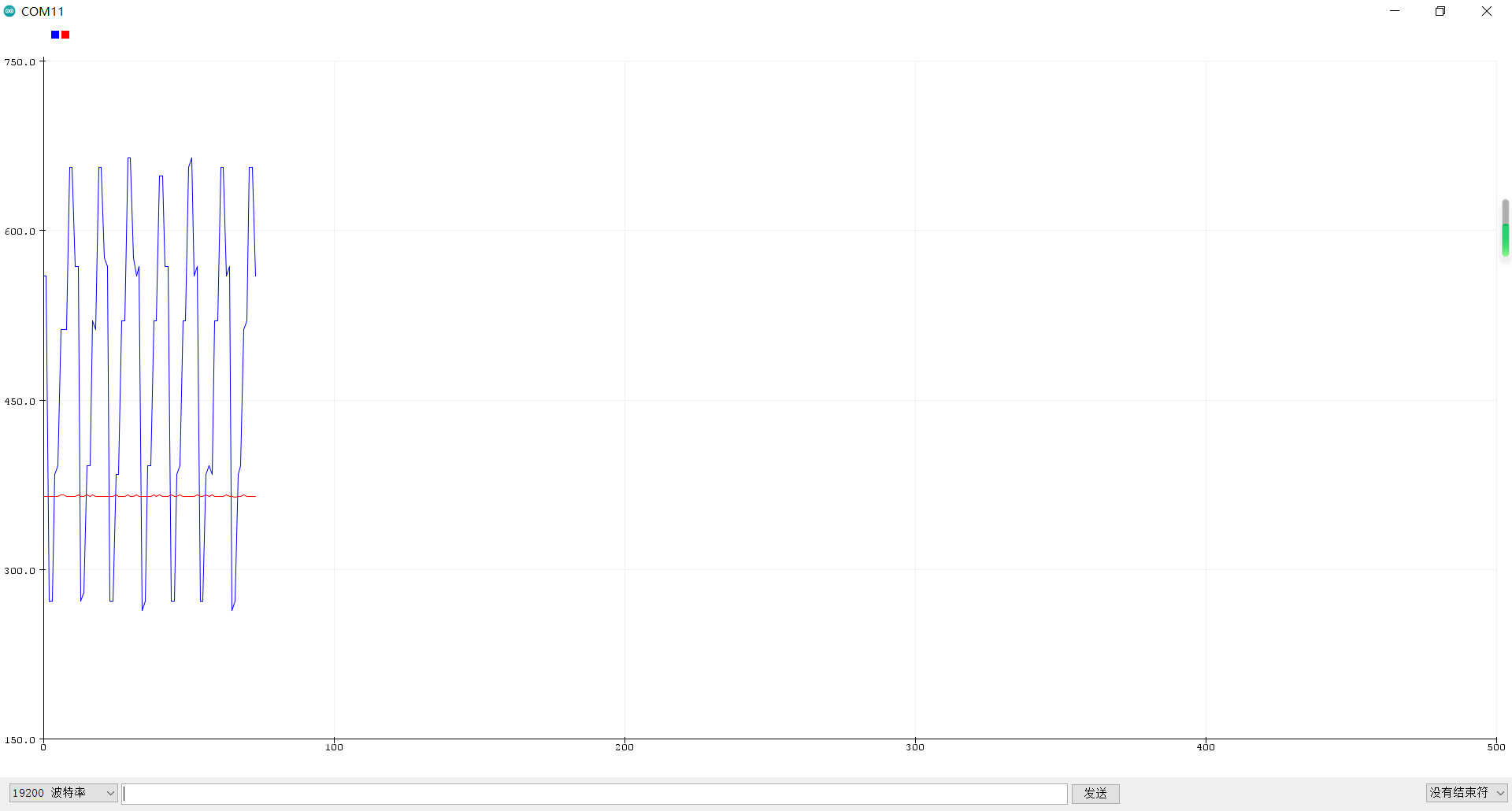
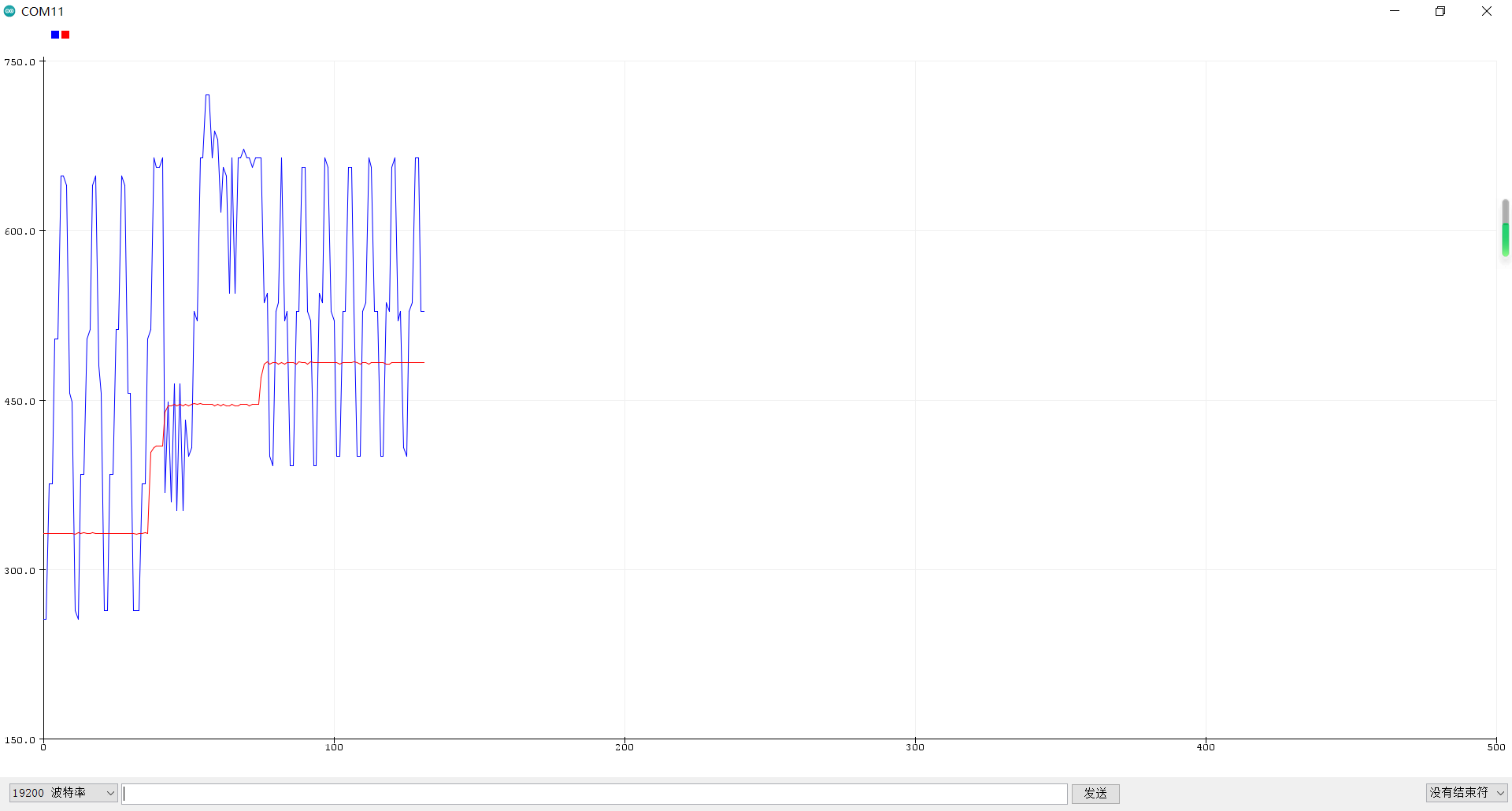
## 2.3 PID参数调试

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：

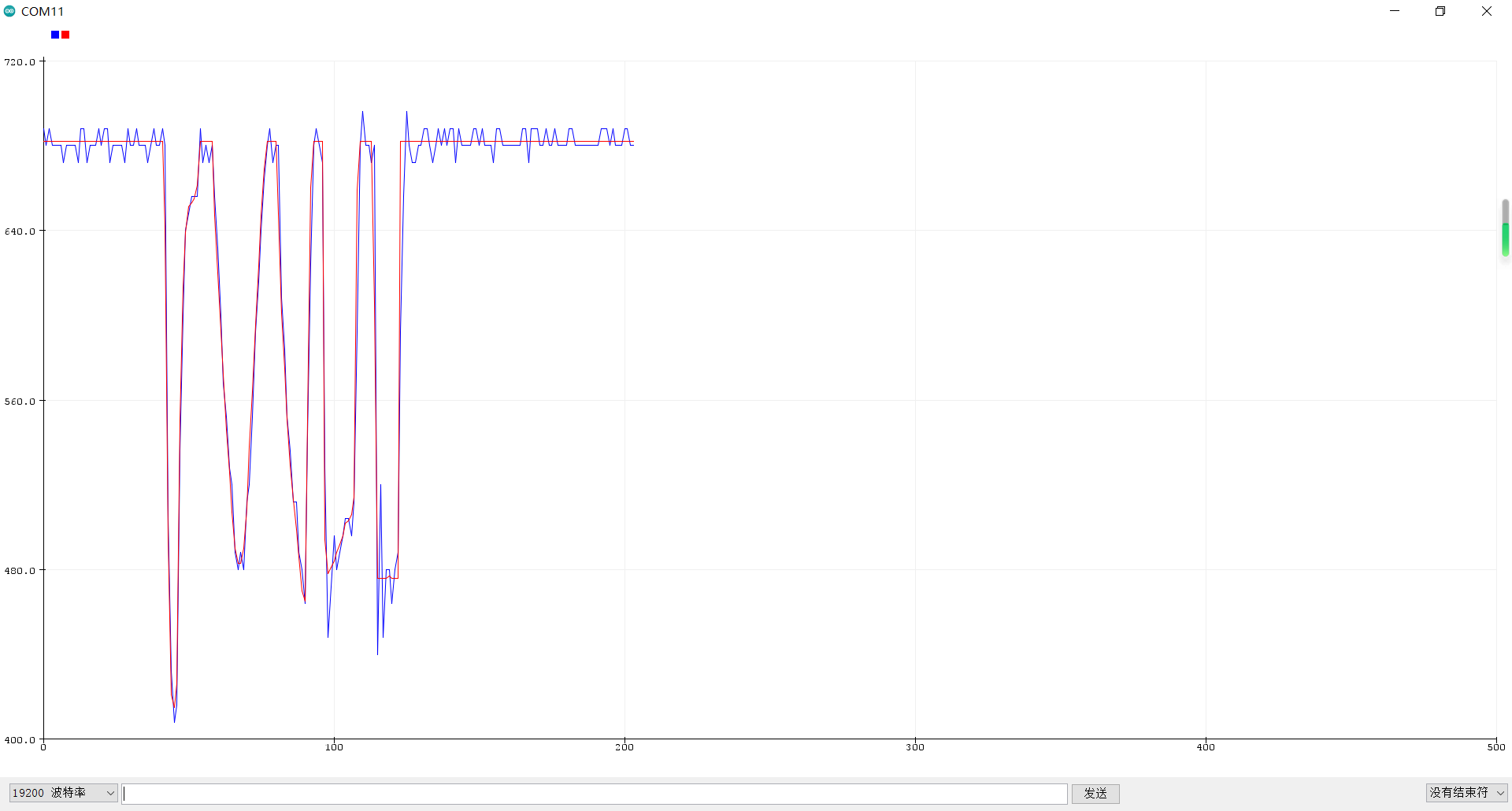
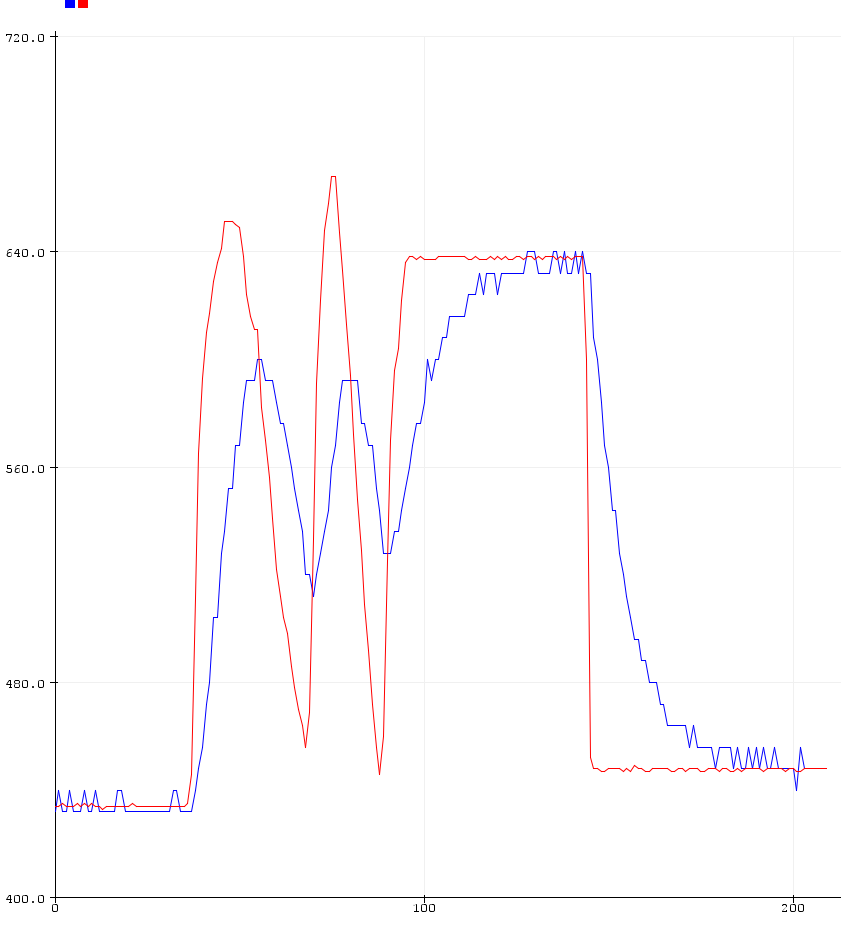
使用的是增量式PID，参数有：*Kp*,*T­­i*,*T­­d*。

波形中：红线：目标速度（×0.1rps）；蓝线：实测转速（×0.1rps）

左图：*Kp*=0.01, *T­­i*=0.1, *T­­d*=0, *T­­s*=10ms; 右图：*Kp*=0.005, *T­­i*=0.1, *T­­d*=0, *T­­s*=10ms;

先将积分时间*T­­i*取为0.1左右（使积分作用尽量小），*T­­d*取为0，然后将*Kp*调整为0.01左右，发现出现严重的振荡现象。然后逐渐减小*Kp*，发现振荡逐渐减小。最后，*Kp*取为0.0002左右效果较好，如下图。

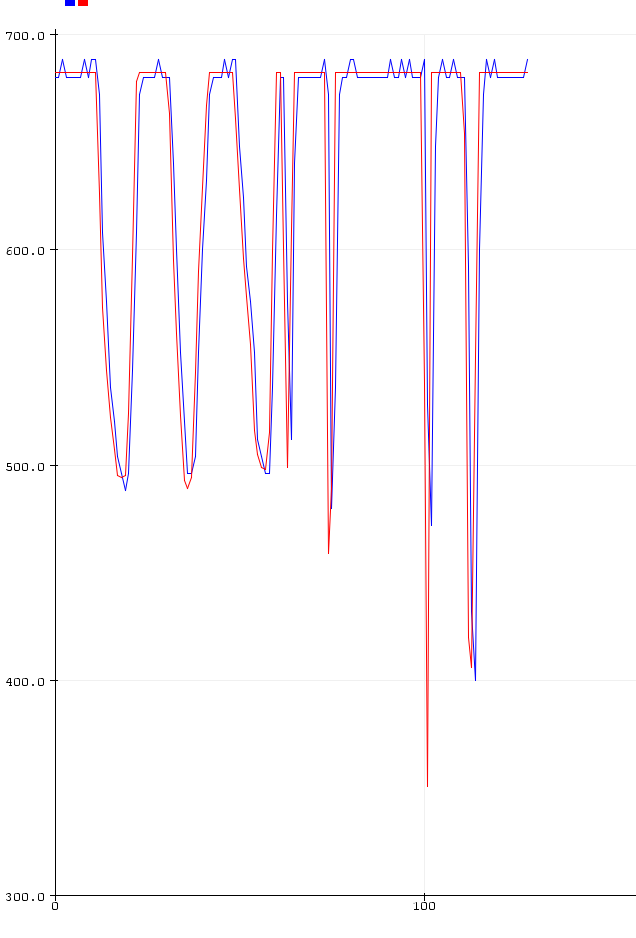
左图：*Kp*=0.0002, *T­­i*=0.1, *T­­d*=0, *T­­s*=10ms; 右图：*Kp*=0.00005, *T­­i*=1, *T­­d*=0, *T­­s*=10ms

但是，在目标速度骤减时，速度降低过快，可能导致电机直接停机。于是，尝试了较大的积分时间，系统快速性降低，但缓解了电机因减速过快而直接停机的问题，电机重新启动的性能也较好。于是适当增大比例系数以改善快速性。我还尝试了积分时间过大且增益过小的情形，发现系统的动态响应性能差；如上右图。

1. 最终结果

*Kp*=0.0003, *T­­i*=0.5, *T­­d*=0, *T­­s*=10ms，跟踪效果如下图

波形中：红线：目标速度（×0.1rps）；蓝线：实测转速（×0.1rps）



1. 简述参数调试过程

见上，与图形一同说明了。

## 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。

1. 每更改一次参数就烧录一次，调参效率低。

解决方法：打开debug中的watch视窗，输入三个参数变量名，即可实时更改其值，最后在代码中更改即可。