轮 趣 科 技

TB6612 电机驱动

用户手册

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
v1.0	2022/11/2	第一次发布
V1.1	2023/06/07	内容勘误
V2.0	2023/06/29	增加了 D24A 模块内容
V2.1	2024/08/27	增加 PWM 的频率范围说明

网址：www.wheeltec.net

序言

我们将通过这篇教程与大家一起学习直流电机的原理和控制、减速器的作用，并介绍一款直流电机驱动芯片 TB6612FNG。

其中 D103A 为精简版模块，除了 TB6612 芯片的外围必须旁路电容，其余接口仅作引出。D153B 为带稳压版双路驱动，板载稳压电源能同时驱动两路电机。D24A 为带稳压版四路驱动，板载稳压电源能同时驱动四路电机。

目录

1. 直流电机入门基础知识

1.1 直流电机原理

下面是分析直流电机的物理模型图。其中，固定部分有磁铁，这里称作主磁极；固定部分还有电刷。转动部分有环形铁心和绕在环形铁心上的绕组。(其中 2 个小圆圈是为了方便表示该位置上的导体电势或电流的方向而设置的)

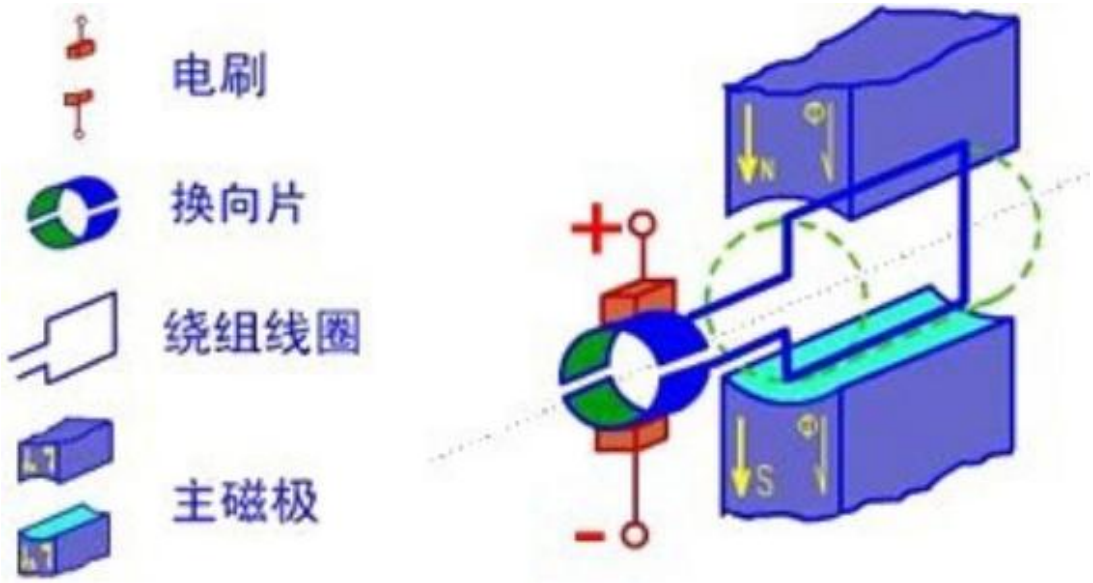

图 1-1-1　直流电机的物理模型图

它的固定部分（定子）上，装设了一对直流励磁的静止的主磁极 N 和 S，在旋转部分（转子）上装设电枢铁心。在电枢铁心上放置了两根导体连成的电枢线圈，线圈的首端和末端分别连到两个圆弧形的铜片上，此铜片称为换向片。换向片之间互相绝缘，由换向片构成的整体称为换向器。换向器固定在转轴上，换向片与转轴之间亦互相绝缘。在换向片上放置着一对固定不动的电刷B1 和 B2，当电枢旋转时，电枢线圈通过换向片和电刷与外电路接通。

在电刷上施加直流电压 U,电枢线圈中的电流流向为：N 极下的有效边中的电流总是一个方向，而 S 极下的有效边中的电流总是另一个方向。这样两个有效边所受的洛伦兹力的方向一致（可以根据左手法则判定），电枢开始转动。

具体就是上图中的+和-分别接到电池的正极和负极，电机即可转动；如果是把上图中的+和-分别接到电池的负极和正极，则电机会反方向转动。电机的转速可以理解为和外接的电压是正相关的（实际是由电枢电流决定）。

总结，如果我们可以调节施加在电机上面的直流电压大小，即可实现直流电

机调速，变施加电机上面直流电压的极性，即可实现电机换向。

1.2 减速器

一般直流电机的转速都是一分钟几千上万转的，所以一般需要安装减速器。减速器是一种相对精密的机械零件，使用它的目的是降低转速，增加转矩。减速后的直流电机力矩增大、可控性更强。按照传动级数不同可分为单级和多级减速器；按照传动类型可分为齿轮减速器、蜗杆减速器和行星齿轮减速器。

图 1-2-1 减速器类型

齿轮减速箱体积较小，传递扭矩大，但是有一定的回程间隙。蜗轮蜗杆减机的主要特点是具有反向自锁功能，可以有较大的减速比，但是一般体积较大，传动效率不高，精度不高。行星减速机其优点是结构比较紧凑，回程间隙小、精度较高，使用寿命很长，额定输出扭矩可以做的很大，但价格略贵。

以下是一款搭配多级齿轮减速箱的电机。

图 1-2-2 多级齿轮减速箱的电机

1.3 电机实物接线图解

具体到我们的电机，我们可以看看电机后面的图解。

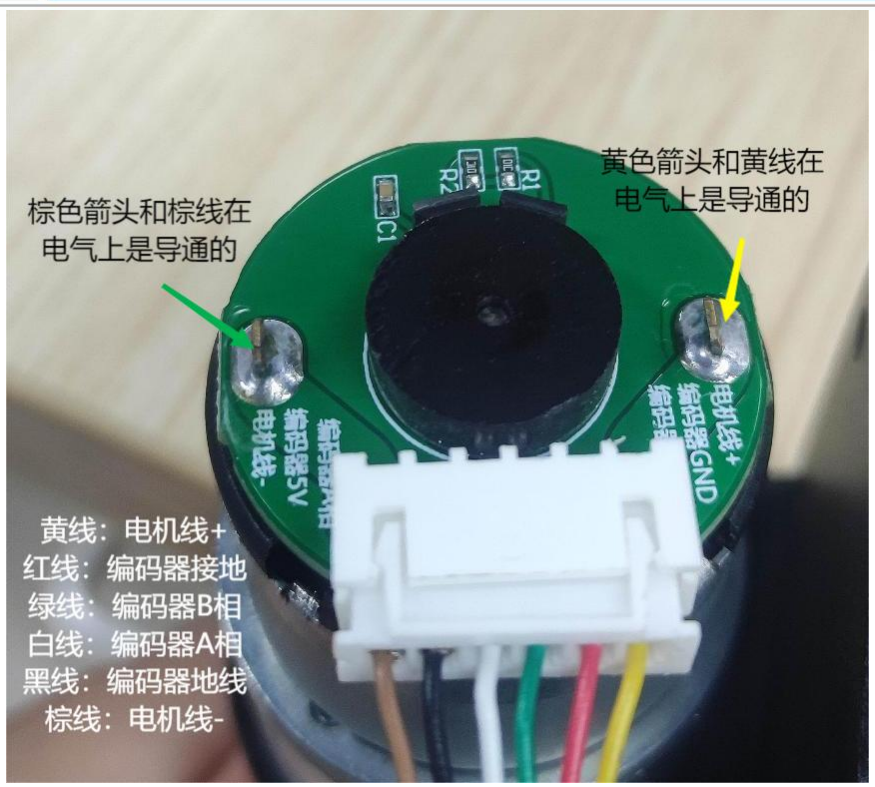

图 1-3-1　电机接线的详细图解

上面介绍了一大堆说直流电机只引出两个线，怎么这个电机有 6 个线，而且还有两个大焊点呢？其实，根据上面的图解也知道，那两个焊点分别和黄线和棕线是连接在一起的。也就是说只有 6 个线，而 6P 排线中，中间的四根线（红绿白黑）是编码器的线，只是用于测速，和直流电机本身没有联系。我们在实现开环控制的时候无需使用。

综上所述，我们只需控制施加在黄线和棕色线两端的直流电压大小和极性即可实现调试和换向。

2. TB6612 模块介绍

(1) TB6612 是东芝半导体公司生产的的一款直流电机驱动器件，它具有大电流的 MOSFET-H 桥结构，双通道电路输出，可同时驱动两个电机。如果我们需要驱动两个电机，只需要一块 TB6612 芯片即可。

(2) 相比 L298N 的热耗性和外围二极管续流电路，它无需外加散热片，外围电路简单，只需要外接电源滤波电容就可以直接驱动电机，利于减小系统尺寸。

(3) 对于 PWM 信号输入频率范围，高达 100kHz 的频率更是足以满足我们大部分的需求了。

(4) TB6612 的主要参数：

最大输入电压：VM = 15V

最大输出电流：Iout=1.2A（平均）或=3.2A（峰值）

具有正反转、短路刹车、停机功能模式

内置过热保护和低压检测电路

2.1 D103A 模块说明

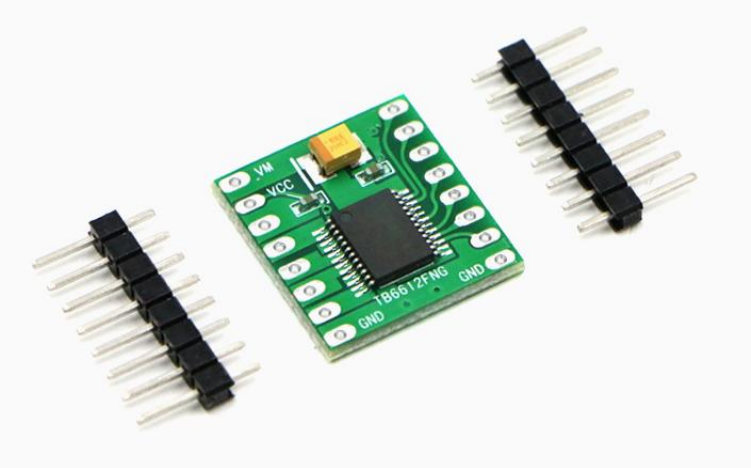

图 2-1-1 D103A 实物图

D103A 模块主要由 TB6612FNG 芯片和三个电容组成，D103A 模块直接引出 TB6612 芯片的引脚，除此之外还增加了几个必要的旁路电容，增强用户使用稳定性。

2.2 D153B 模块说明

这个模块除了 1.1 上面所概述的功能，还集成了以下的功能：

（1）新增了 5V 稳压电路，支持 5V、5A 的输出。共有 3 个这样的引脚可以对外供电。

（2）增加电压测量电路，通过串联一个 10kΩ和一个 1kΩ的电阻，对输入电源进行 1/11 的分压后，可以通过 ADC 采集并计算得到电源的电压进行监控。

（3）引出电机标准的 6PIN 接口，可以通过用排线连接，AB 相编码器的信号接到单独的引脚输出。

（4）新增了电源输出电路，电源输入接口做了并联输出，可以多个模块级联使用。

（5）含有电源开关，可以进行开启。

（6）具有过温保护功能,关断温度为 65℃。

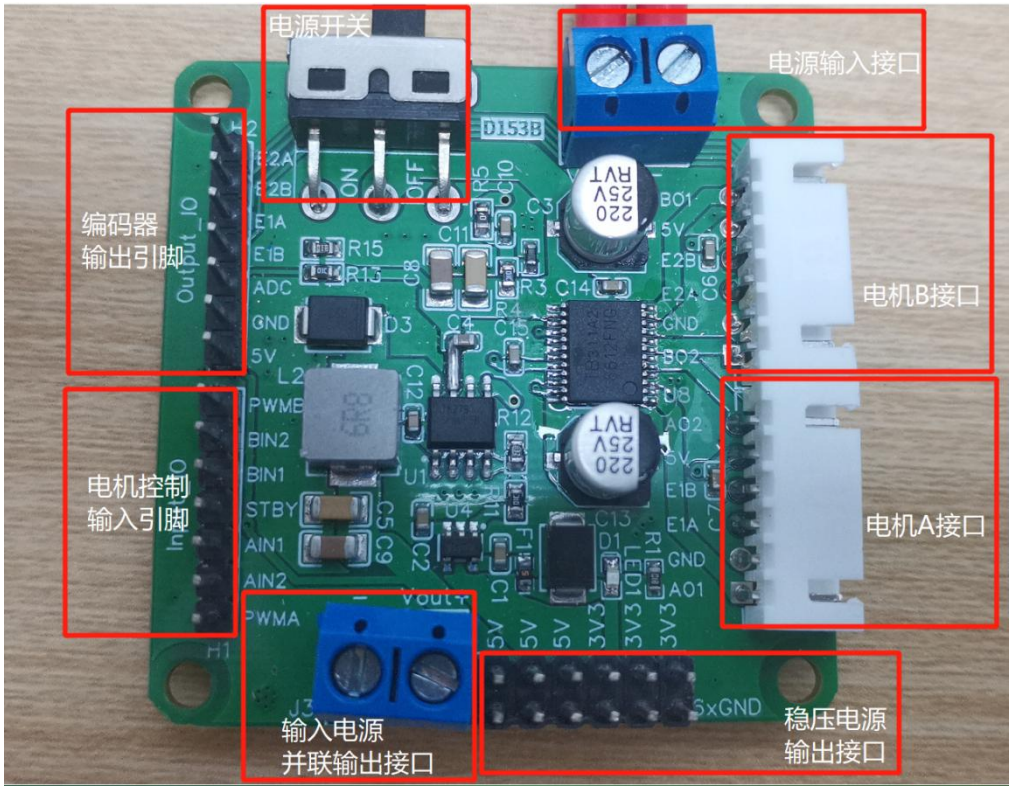

图 2-2-1　TB6612 带稳压模块详细图

2.3 D24A 模块说明

该模块具有四路驱动，可以同时驱动四个直流电机。除了 1.1 上面所概述的功能，还集成了以下的功能：

（1）新增了 5V 稳压电路，支持 5V、5A 的输出。共有 4 个这样的引脚可以对外供电。

（2）增加电压测量电路，通过串联一个 10kΩ和一个 1kΩ的电阻，对输入电源进行 1/11 的分压后，可以通过 ADC 采集并计算得到电源的电压进行监控。

（3）引出电机标准的 6PIN 接口，可以通过用排线连接，AB 相编码器的信号接到单独的引脚输出。

（4）含有电源开关，可以进行开启。

（5）具有过温保护功能,关断温度为 65℃。

图 2-3-1　D24A 模块详细图

3. 原理图说明

3.1 TB6612 芯片原理图介绍与控制说明

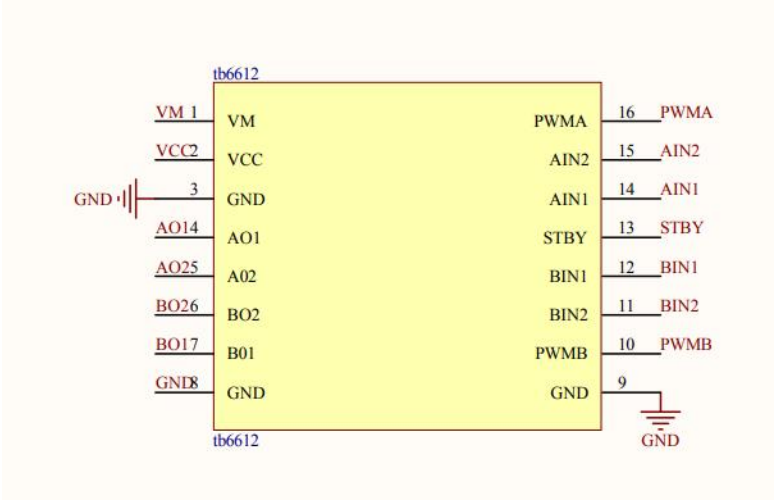

图 3-1-1　TB6612 芯片原理图

表 3-1-1　TB6612 引脚表

引脚	功能
VM	电机驱动电压输入（4.5V-13.5V）
VCC	逻辑电平输入端（2.7V-5.5V）
GND	接地
AO1	A 电机输出端 1
AO2	A 电机输出端 2
BO1	B 电机输出端 1
BO2	B 电机输出端 2
PWMA	A 控制信号输入
AIN1	A 电机输入端 1
AIN2	A 电机输入端 2
PWMB	B 控制信号输入
BIN1	B 电机输入端 1
BIN2	B 电机输入端 2
STBY	正常工作\待机状态控制端

在使用的过程中，VM 需要比较大的电压，如 12V，而且 VCC 还需要接 5V。VM 是用来给电机供电的，VCC 是给芯片供电的，切记勿要接错，否则将会烧坏芯片。在控制电机时，AO1、AO2、BO1、BO2 连接电机的两个引脚，PWMA、PWMB 输入 PWM 信号（一般为 10kHz 的 PWM 信号），AIN1、AIN2、BIN1、BIN2 用来控制电机的运动方向。STBY 端是一个使能信号端，当 STBY=1 时，正常工作，输入 PWM 信号，电机可正常运行；当 STBY=0 时，电机驱动处于待机状态，输入信号，电机不会运行。该芯片中的两个 GND 引脚，需一个接电源地，一个接单片机地。

注：PWM 频率范围 0-100Khz(推荐使用 10Khz)

表 3-1-2 TB6612 真值表

PWM	1	1	1	1	0
IN1	0	1	0	1	x
IN2	0	0	1	1	x
功能	制动停止	正转	反转	制动停止	停止

想要电机转动，必须要有 PWM 输入，才有 O1、O2 输出，只接 IN1、IN2 是不会产生 O1、O2 信号。

如果手上没有单片机，我们可以直接接电源的引脚来进行测试：

AIN1 接 3.3V~5V、AIN2 接 GND、PWMA 接 3.3V~5V。这样就相当于电机满占空比正转。

AIN2 接 3.3V~5V，AIN1 接 GND、PWMA 接 3.3V~5V。这样就相当于电机满占空比反转。

这是控制电机的正反向运动，如果想要控制电机的转速，需要通过 PWM 来控制。

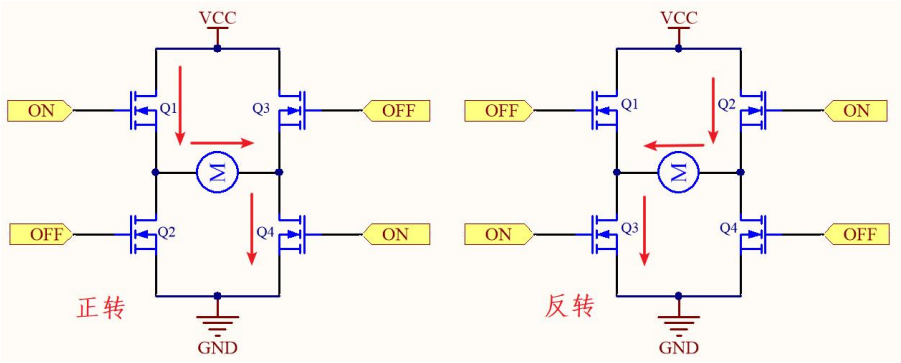

图 3-1-2 正反转原理说明图

要想让电机可以转动，那就必须使对角的 mos 管导通，如图 2 所示，如果想

让电机正转，那就使 Q1 和 Q4 导通，如果想让电机反转，那就让 Q2 和 Q3 导通。但是要注意不能让同一侧的 mos 管同时导通，因为这时电路上没有任何负载，电流会变得非常大，会烧坏电路。

3.2 D103A 模块原理图介绍

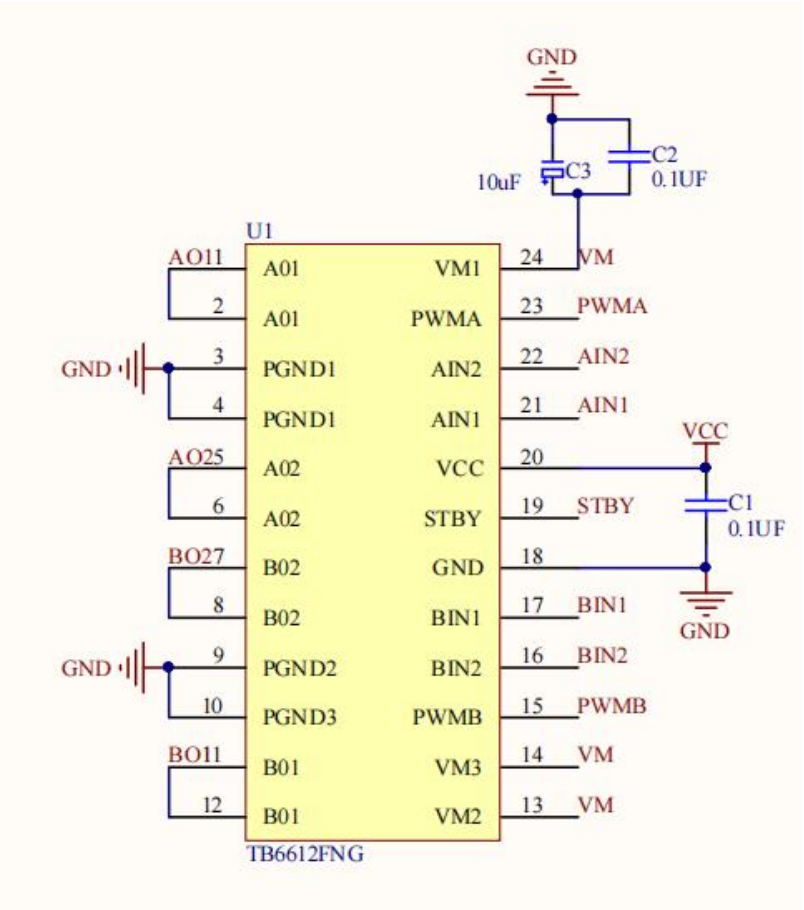

图 3-2-1　D103A 模块原理图

D103A 模块和 TB6612 的基本一样，只不过 D103B 在 VM1 上并接两个电容。一个为 0.1μF 无极性电容和 10μF 电解电容，这个搭配方式我们可以在电路中经常看到，因为供电时的实际电源并不稳定，夹杂高频和低频干扰。10μF 电容对与滤波干扰有较好的作用，但对于高频干扰，电容呈现感性，阻抗很大，无法有效滤波，所以再并连一个 0.1μF 的电容滤除高频分量。

在 VCC 和 GND 处连接了电容，起到了一个储能作用，避免电路有时候耗电大，有时小的问题。

3.3 D153B 原理图介绍

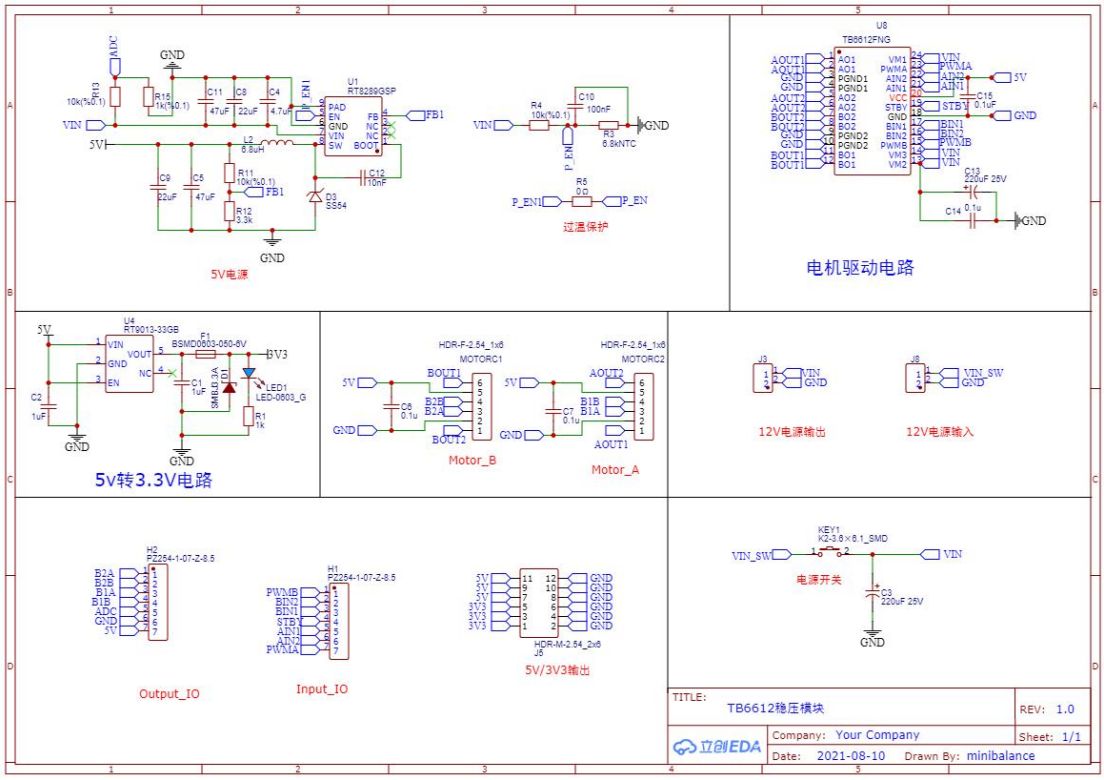

图 3-3-1　D153B 原理图

D153B 模块多了稳压电路、电机接口、12V 电源输入输出、5V 电源和 3.3V 电源的并联输出和电源开关。这就可以直接和一个单片机的最小模块直接使用，可以采集 ADC，对模块的电源进行监控；有了电机接口，可以直接插上电机线，对电机的编码器数据读取，还只需要接一个外部电源，再由 D153B 的 5V 电源输出为单片机供电，比较简便。

3.4 D24A 原理图介绍

D24A 模块也具有稳压电路、电机接口、12V 电源输入、5V 电源和 3.3V 电源和电源开关。这也可以直接和一个单片机的最小模块直接使用，可以采集 ADC，对模块的电源进行监控；有了电机接口，可以直接插上电机线，对电机的编码器数据读取，还只需要接一个外部电源，再由 D24A 的 5V 电源输出为单片机供电，比较简便。

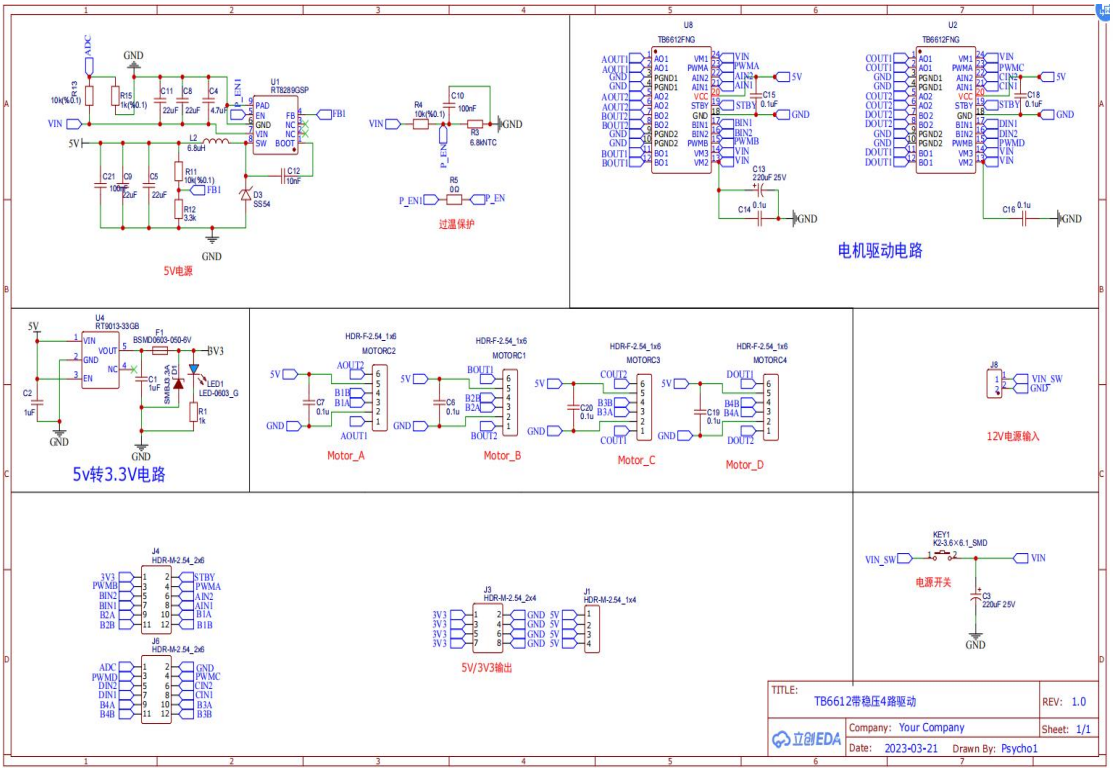

图 3-3-2　D24A 原理图

4. 接线说明

提供的接线说明都是基于本公司产品的例程，我们提供了 STM32F103C8T6 核心板和 ArduinoUNO 的例程各一份。别的产品需要自己摸索一下，但也只需改动引脚，只要是和本例程功能相同的引脚即可。但是 D24A 只提供 STM32F103RCT6 核心板例程一份（**注意：想要同时实现 4 个电机的闭环控制，那使用的单片机至少拥有 5 个定时器的引脚。在 D24A 例程中，使用了定时器 5 来输出 PWM，定时 2、3、4、8 作为编码器**）。

4.1 D103A 模块接线说明

① D103A 模块和 STM32F103C8T6 核心板接线如下表所示：

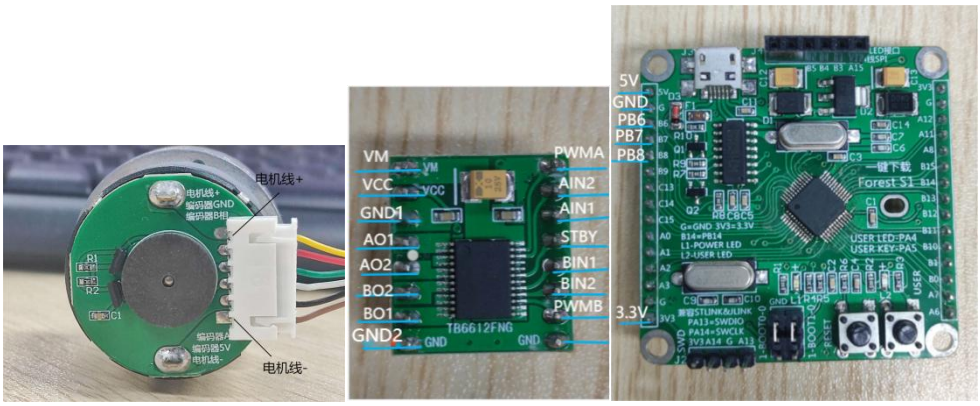

图 4-1-1　各模块引脚图

表 4-1-1　D103A 和 stm32F103C8T6 接线表

STM32F103C8T6	D103A 模块	电机	
	VM		5V~15V 外接电源+
5V	VCC		
GND	GND(两个都要接)		
	AO1	电机电源+	
	AO2	电机电源-	
PB6	PWMA		
PB8	AIN1		
PB7	AIN2		
3.3V	STBY		
	GND2		外接地

② 4.1.2 D103A 和 Arduino UNO 接线如下表所示：

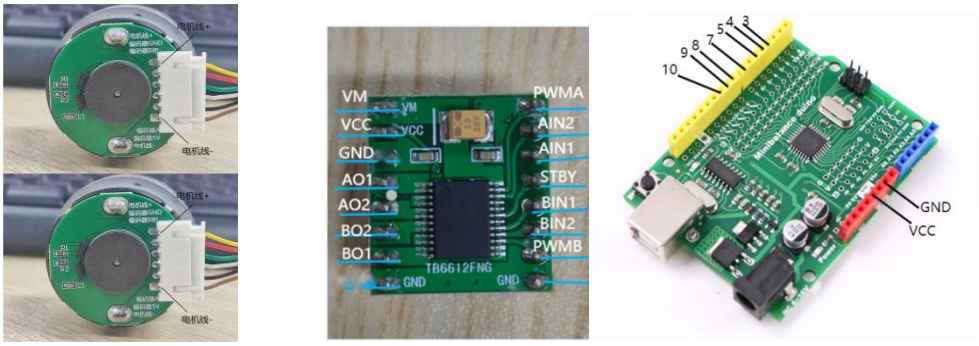

图 4-1-2　各模块引脚图

表 4-1-2　D103A 与 Arduino 连线表

电机	G103A 模块	Arduino	
	VM		5V~15V 外接电源
	VCC	VCC	
	GND	GND	
电机线+	AO1		
电机线-	AO2		
电机线+	BO2		
电机线-	BO1		
	PWMA	3	
	AIN2	4	
	AIN1	5	
	STBY	7	
	BIN1	8	
	BIN2	9	
	PWMB	10	

4.2 D153B 模块接线说明

① D153B 带稳压模块与 STM32F103C8T6 接下如下图所示：

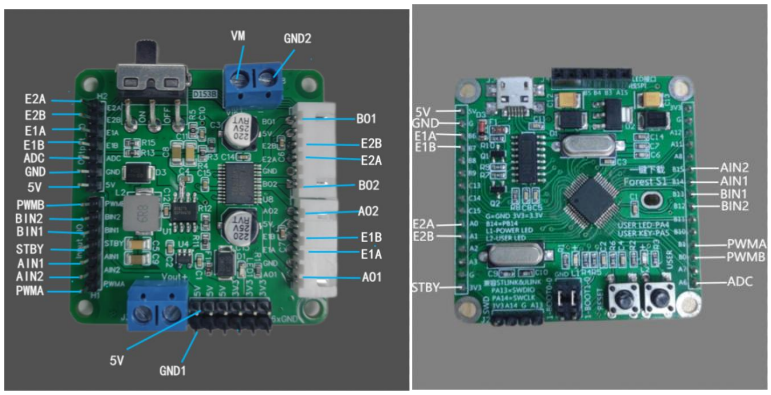

图 4-2-1 各模块引脚图

表 4-2-1 D153B 模块与 STM32F103C8T6 接线表

电机(用 6pin 接线连接)	D153B	STM32F103C8T6	
	ADC	PA6	
	PWMB	PB0	
	BIN2	PB12	
	BIN1	PB13	
	STBY	3.3V	
	AIN1	PB14	
	AIN2	PB15	
	PWMA	PB1	
	VM		5V~15V 外接电源
	GND2		外接地
	5V	5V	
	GND1	GND	
编码器 B 相	E2B	PA1	
编码器 A 相	E2A	PA0	
编码器 B 相	E1B	PB7	
编码器 A 相	E1A	PB6	
电机线-	A01		
电机线+	A02		
电机线+	B01		
电机线-	B02		

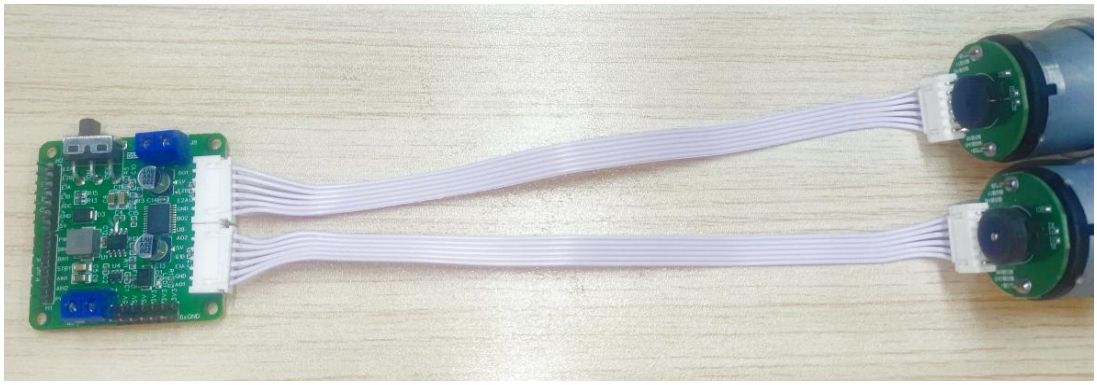

图 4-2-1　电机和 D153B 接线示意图

② D153B 带稳压模块与 Arduino UNO 接线如下表所示：

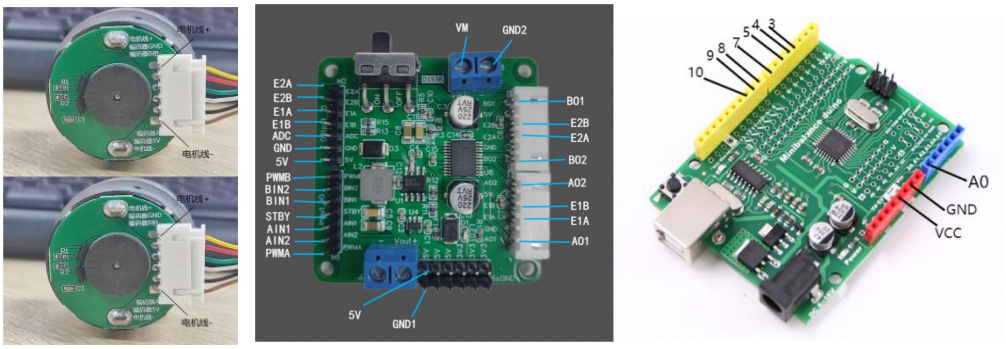

图 4-2-2　各模块引脚图

表 4-2-1　D153B 模块与 Arduino 连线表

电机	D153B	Arduino	
	ADC	A0	
	PWMB	10	
	BIN2	9	
	BIN1	8	
	STBY	7	
	AIN1	5	
	AIN2	4	
	PWMA	3	
	5V	VCC	
	GND1	GND	
电机线-	BO1		
电机线+	BO2		
电机线+	AOI		
电机线-	AO2		
	VM		5V~15V 外接电源
	GND2		外接地

4.3 D24A 模块接线说明

表 4-3-1　D24A 模块与 STM32F103RCT6 接线表

电机(用 6pin 接线连接)	D24A 模块	STM32F103RCT6	
	ADC	A5	
	PWMA	PA0	
	PWMB	PA1	
	PWMC	PA2	
	PWMD	PA3	
	AIN1	PC13	
	AIN2	PC14	
	BIN1	PB12	
	BIN2	PB13	
	CIN1	PB0	
	CIN2	PB1	
	DIN1	PC1	
	DIN2	PC2	
编码器 A 相	E1A	PC6	
编码器 B 相	E1B	PC7	
编码器 A 相	E2A	PA15	
编码器 B 相	E2B	PB3	
编码器 A 相	E3A	PA6	
编码器 B 相	E3B	PA7	
编码器 A 相	E4A	PB6	
编码器 B 相	E4B	PB7	
	STBY	3.3V	跳线帽连接
	VIN+		5V~15V 外接电源
	-		外接地
	5V	5V	
	GND	GND	

5. 代码例程

5.1 D103A 的 STM32F103C8T6 的例程代码

```
void Gpio_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;                    //定义结构体GPIO_InitStructure

  RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB, ENABLE); // 使能PB端口时钟
  GPIO_InitStructure.GPIO_Pin =   GPIO_Pin_7| GPIO_Pin_8;      //PB7 PB8
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //推挽，增大电流输出能力
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //IO口速度
  GPIO_Init(GPIOB, &GPIO_InitStructure);                  //GBIOB初始化

}
```

图 5-1-1 电机接口初始化

电机接口的初始化，也就是对 AIN1、AIN2 这两个接口初始化，使它输出高低电平，从而控制电机正反转动。

```
void    pwm_int(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;                //定义结构体GPIO_InitStructure
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;       //定义结构TIM_TimeBaseStructure
    TIM_OCInitTypeDef  TIM_OCInitStructure;               //定义结构TIM_OCInitStructure

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE);  //使能定时器4时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//使能PB端口时钟

    GPIO_InitStructure.GPIO_Pin =   GPIO_Pin_6;          //PB6
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;     //IO口速度
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;        //复用模式输出
    GPIO_Init(GPIOB, &GPIO_InitStructure);              //GBIOB初始化

  TIM_TimeBaseStructure.TIM_Period = 7199;    //设置在下一个更新事件装入活动的自动重装载寄存器周期的值
  TIM_TimeBaseStructure.TIM_Prescaler = 0; //设置用来作为TIMx时钟频率除数的预分频值  不分频
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;  //设置时钟分割:TDTS = Tck_tim
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //TIM向上计数模式
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);//根据TIM_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位

  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;   //PWMTIM脉冲宽度调制模式2
  TIM_OCInitStructure.TIM_Pulse  = 0;                 //输入脉冲捕获值
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;       //设置TIM输出比较极性为高
  TIM_OCInitStructure.TIM_OutputState= TIM_OutputState_Enable;   //比较输出使能
  TIM_OC1Init(TIM4, &TIM_OCInitStructure);  //根据TIM_OCInitStructure中指定的参数初始化外设TIM4

  TIM_CtrlPWMOutputs(TIM4,ENABLE);                    //主输出使能

  TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);       //使能预装载寄存器

  TIM_ARRPreloadConfig(TIM4, ENABLE);      //使能自动装载允许位

  TIM_Cmd(TIM4, ENABLE);   //启动定时器TIM4

}
```

图 5-1-2　电机 PWM 初始化

图 5-1-2 是对 PWM 外设的初始化，用的是定时器 4。

```
void moto(int mode)
{
    if(mode==1)
    {
     GPIO_SetBits(GPIOB, GPIO_Pin_7);     // 高电平
     GPIO_ResetBits(GPIOB, GPIO_Pin_8);   // 低电平

     TIM_SetCompare1(TIM4,3000);    //设置TIM4通道1的占空比  3000/7200
    }
     if(mode==0)
    {
     GPIO_SetBits(GPIOB, GPIO_Pin_8);     // 高电平
     GPIO_ResetBits(GPIOB, GPIO_Pin_7);   // 低电平

     TIM_SetCompare1(TIM4,4000);    //设置TIM4通道1的占空比  4000/7200
     }

}
```

图 5-1-3　控制电机转向控制

这里的转向控制的代码和我们上面的真值表是一样的，必须一个高电平一个低电平并且需要 PWM 的输入，电机才能转动起来。图 5.1-3 中的 AIN1 为 0 时，AIN2 为 1 时为反转，AIN1 为 1、AIN2 为 0 时为正转，PWM 是调速的，如下图 5-1-4 所示，我们设置的 PWM 占空比为 7200，当占空比为 7200 时，转速为满转。

```
void    pwm_int(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;                  //定义结构体GPIO_InitStructure
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;       //定义结构TIM_TimeBaseStructure
    TIM_OCInitTypeDef  TIM_OCInitStructure;               //定义结构TIM_OCInitStructure

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE);  //使能定时器4时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//使能PB端口时钟

    GPIO_InitStructure.GPIO_Pin =   GPIO_Pin_6;           //PB6
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;     //IO口速度
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;       //复用模式输出
    GPIO_Init(GPIOB, &GPIO_InitStructure);               //GBIOB初始化

  TIM_TimeBaseStructure.TIM_Period = 7199;     //设置在下一个更新事件装入活动的自动重装载寄存器周期的值
  TIM_TimeBaseStructure.TIM_Prescaler = 0; //设置用来作为TIMx时钟频率除数的预分频值   不分频
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;   //设置时钟分割:TDTS = Tck_tim
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //TIM向上计数模式
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);//根据TIM_TimeBaseInitStruct中指定的参数初始化TIMx的
```

图 5-1-4　PWM 的占空比设置

5.2 D153B 与 STM32F103C8T6 的例程代码

```
void Gpio_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;            //定义结构体GPIO_InitStructure

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); // 使能PB端口时钟
  GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_12| GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;    //PB12 PB813 PB14 PB15
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;       //推挽, 增大电流输出能力
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //IO口速度
    GPIO_Init(GPIOB, &GPIO_InitStructure);          //GBIOB初始化

}
```

图 5-2-1　D153B 电机接口初始化

D153B 是初始化了 AIN1、AIN2、BIN1、BIN2。

```
void PWM_Int(u16 arr,u16 psc)
{
    GPIO_InitTypeDef GPIO_InitStructure;                //定义结构体GPIO_InitStructure
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;      //定义结构体TIM_TimeBaseStructure
    TIM_OCInitTypeDef TIM_OCInitStructure;              //定义结构体TIM_OCInitStructure

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);//使能PB端口时钟
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3,ENABLE);//使能定时器3

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;          //复用模式输出
    GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1; //PB0 、PB1
    GPIO_InitStructure.GPIO_Speed= GPIO_Speed_50MHz;         //IO口速度
    GPIO_Init(GPIOB,&GPIO_InitStructure);                    //GPIO初始化

    TIM_TimeBaseStructure.TIM_Period = arr;                 //设置下一个更新活动的自动重装载寄存器的值
    TIM_TimeBaseStructure.TIM_Prescaler = psc;              //预分配值
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;            //时钟分割
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //向上计数
    TIM_TimeBaseInit(TIM3,&TIM_TimeBaseStructure);

    TIM_OCInitStructure.TIM_OCMode= TIM_OCMode_PWM1;              //PWM脉冲宽度调制1
    TIM_OCInitStructure.TIM_Pulse = 0;                            //设置待装入捕获比较寄存器的脉冲值
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;     //设置TIM输出极性为高
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;//比较输出使能
    TIM_OC3Init(TIM3,&TIM_OCInitStructure);
    TIM_OC4Init(TIM3,&TIM_OCInitStructure);

    TIM_CtrlPWMOutputs(TIM3,ENABLE);//主输出使能

    TIM_OC3PreloadConfig(TIM3,TIM_OCPreload_Enable);
    TIM_OC4PreloadConfig(TIM3,TIM_OCPreload_Enable);//使能预装载寄存器

    TIM_ARRPreloadConfig(TIM3,ENABLE);              //使能自动装载允许位
    TIM_Cmd(TIM3,ENABLE);//启动定时器3


}
```

图 5-2-2　D153B 电机 PWM 的初始化

D153B 的 PWM 使用的是定时器 3。

```
void Encoder_Init_Tim2(void)
{
   TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);//使能定时器4的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能PB端口时钟

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;  //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOA, &GPIO_InitStructure);                          //根据设定参数初始化GPIOB

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; //选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;///TIM向上计数
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM2, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;     //滤波10
  TIM_ICInit(TIM2, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM2, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM2,0);
  TIM_Cmd(TIM2, ENABLE);
}
```

图 5-2-3 编码器 TIM2 的初始化

```
void Encoder_Init_Tim4(void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);//使能定时器4的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//使能PB端口时钟

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;  //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOB, &GPIO_InitStructure);                //根据设定参数初始化GPIOB

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;//选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;////TIM向上计数
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM4, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;
  TIM_ICInit(TIM4, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM4, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM4,0);
  TIM_Cmd(TIM4, ENABLE);
}
```

图 5-2-4　编码器 TIM4 的初始化

因为D153B使用的是电机接线来连接电机，所以我们的例程也包括了编码器，实现一个闭环的操作。并且在主函数处串口打印出我们的编码器值和电源电压的值。

```
/**************************************************************************
函数功能：电机的闭环控制
入口参数：左右电机的编码器值
返回值  ：电机的PWM
**************************************************************************/
int Velocity_A(int TargetVelocity, int CurrentVelocity)
{
    int Bias;  //定义相关变量
        static int ControlVelocity, Last_bias; //静态变量，函数调用结束后其值依然存在

        Bias=TargetVelocity-CurrentVelocity; //求速度偏差

        ControlVelocity+=Velcity_Kp*(Bias-Last_bias)+Velcity_Ki*Bias;  //增量式PI控制器
                                                                       //Velcity_Kp*(Bias-Last_bias) 作用为限制加速度
                                                                         //Velcity_Ki*Bias  速度控制值由Bias不断积分得到 偏差越大加速度越大
        Last_bias=Bias;
        if(ControlVelocity>7200) ControlVelocity=7200;
        else if(ControlVelocity<-7200) ControlVelocity=-7200;
        return ControlVelocity; //返回速度控制值
}
```

```
int TargetVelocity=500;
int main(void)
  {
```

修改此值可以修改速度

图 5-2-5　电机的闭环控制

如图 5-2-5 所示，这是一个电机的闭环控制代码，利用的是 PID 控制电机的 PWM 保持在我们想要的一个值。PID 就是比例、微分和积分控制。先计算出编码器当下的值，利用电机当下的值来和我们设定的值来进行 PID 控制，从而实现电机一直保持在我们想要的速度。用户可以自行 DIY 出自己想要的效果，这里只是对编码器值的一个反馈，没有计算出速度与编码器的关系，所以无法进行速度的控制。例如：速度=编码器读数（利用中断进行一个周期读数）*读取频率*/倍频数/减速比/编码器精度*轮子的周长。

5.3 D24A 与 STM32F103RCT6 的例程代码

```
void Gpio_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;            //定义结构体GPIO_InitStructure

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOC, ENABLE); // 使能PB端口时钟
    GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_12| GPIO_Pin_13;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //推挽，增大电流输出能力
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //IO口速度
    GPIO_Init(GPIOB, &GPIO_InitStructure);          //GBIOB初始化

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_13|GPIO_Pin_14;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //推挽，增大电流输出能力
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //IO口速度
    GPIO_Init(GPIOC, &GPIO_InitStructure);          //GBIOB初始化

}
```

图 5-3-1　D24A 电机接口初始化

电机接口的初始化，也就是对 AIN1、AIN2、BIN1、BIN2、CIN1、CIN2、DIN1、DIN2 这两个接口初始化，使它输出高低电平，从而控制 4 个电机正反转动。

```
void PWM_Int(u16 arr,u16 psc)
{
    GPIO_InitTypeDef GPIO_InitStructure;                //定义结构体GPIO_InitStructure
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;      //定义结构体TIM_TimeBaseStructure
    TIM_OCInitTypeDef TIM_OCInitStructure;              //定义结构体TIM_OCInitStructure

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);//使能PA端口时钟
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5,ENABLE);//使能定时器3

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Speed= GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    TIM_TimeBaseStructure.TIM_Period = arr;                 //设置下一个更新活动的自动重装载寄存器的值
    TIM_TimeBaseStructure.TIM_Prescaler = psc;              //预分配值
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;            //时钟分割
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //向上计数
    TIM_TimeBaseInit(TIM5,&TIM_TimeBaseStructure);

    TIM_OCInitStructure.TIM_OCMode= TIM_OCMode_PWM1;            //PWM脉冲宽度调制1
    TIM_OCInitStructure.TIM_Pulse = 0;                          //设置待装入捕获比较寄存器的脉冲值
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;   //设置TIM输出极性为高
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;//比较输出使能
    TIM_OC1Init(TIM5,&TIM_OCInitStructure);
    TIM_OC2Init(TIM5,&TIM_OCInitStructure);
    TIM_OC3Init(TIM5,&TIM_OCInitStructure);
    TIM_OC4Init(TIM5,&TIM_OCInitStructure);


    TIM_OC1PreloadConfig(TIM5,TIM_OCPreload_Enable);
    TIM_OC2PreloadConfig(TIM5,TIM_OCPreload_Enable);
    TIM_OC3PreloadConfig(TIM5,TIM_OCPreload_Enable);
    TIM_OC4PreloadConfig(TIM5,TIM_OCPreload_Enable);//使能预装载寄存器

    TIM_ARRPreloadConfig(TIM5,ENABLE);              //使能自动装载允许位
    TIM_Cmd(TIM5,ENABLE);//启动定时器5
}
```

图 5-3-2　D24A 模块电机 PWM 的初始化

这里的 PWM 使用的是定时器 5 的 4 个引脚，设置为 PWM 值为 7200 时满转。

```
void Encoder_Init_Tim8(void)
{
   TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM8, ENABLE);//使能定时器4的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);//使能PB端口时钟


  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;  //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOC, &GPIO_InitStructure);                //根据设定参数初始化GPIOA

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;//选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;////TIM向上计数
  TIM_TimeBaseInit(TIM8, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM8, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;    //滤波10
  TIM_ICInit(TIM8, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM8, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM8, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM8,0);
  TIM_Cmd(TIM8, ENABLE);
}
```

图 5-3-3　编码器 TIM8 的初始化

```
void Encoder_Init_Tim2(void)
{
   TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);//使能定时器4的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO, ENABLE);//使能PB端口时钟

  GPIO_PinRemapConfig(GPIO_FullRemap_TIM2,ENABLE);
  GPIO_PinRemapConfig(GPIO_Remap_SWJ_Disable,ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15;    //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOA, &GPIO_InitStructure);                //根据设定参数初始化GPIOB

  GPIO_InitStructure.GPIO_Pin =GPIO_Pin_3;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOB,&GPIO_InitStructure);

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;//选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;////TIM向上计数
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM2, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;    //滤波10
  TIM_ICInit(TIM2, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM2, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM2,0);
  TIM_Cmd(TIM2, ENABLE);
}
```

图 5-3-4　编码器 TIM2 的初始化

这里的 TIM2 是使用重定义，将 PA15 和 PB3 分别重定义为定时器 2 的通道 1 和通道 2。

```
void Encoder_Init_Tim3(void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);//使能定时器3的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能PB端口时钟

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;  //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOA, &GPIO_InitStructure);                //根据设定参数初始化GPIOB

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;//选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;////TIM向上计数
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM3, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;    //滤波10
  TIM_ICInit(TIM3, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM3, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM3,0);
  TIM_Cmd(TIM3, ENABLE);
}
```

图 5-3-5 编码器 TIM3 的初始化

```
void Encoder_Init_Tim4(void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_ICInitTypeDef TIM_ICInitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);//使能定时器4的时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//使能PB端口时钟

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;  //端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
  GPIO_Init(GPIOB, &GPIO_InitStructure);                //根据设定参数初始化GPIOB

  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // 预分频器
  TIM_TimeBaseStructure.TIM_Period = 65535; //设定计数器自动重装值
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;//选择时钟分频：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;////TIM向上计数
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_EncoderInterfaceConfig(TIM4, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//使用编码器模式3
  TIM_ICStructInit(&TIM_ICInitStructure);
  TIM_ICInitStructure.TIM_ICFilter = 10;
  TIM_ICInit(TIM4, &TIM_ICInitStructure);
  TIM_ClearFlag(TIM4, TIM_FLAG_Update);//清除TIM的更新标志位
  TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);
  //Reset counter
  TIM_SetCounter(TIM4,0);
  TIM_Cmd(TIM4, ENABLE);
}
```

图 5-3-6 编码器 TIM4 的初始化

因为 D24A 使用的也是电机接线来连接电机，所以我们的例程也包括了编码器，实现一个闭环的操作。并且在主函数处串口打印出我们的编码器值和电源电压的值。

```
/**************************************************************************
函数功能：电机的闭环控制
入口参数：左右电机的编码器值
返回值  ：电机的PWM
**************************************************************************/
int Velocity_A(int TargetVelocity, int CurrentVelocity)
{
    int Bias;  //定义相关变量
        static int ControlVelocity, Last_bias; //静态变量，函数调用结束后其值依然存在

        Bias=TargetVelocity-CurrentVelocity; //求速度偏差

        ControlVelocity+=Velcity_Kp*(Bias-Last_bias)+Velcity_Ki*Bias;  //增量式PI控制器
                                                                       //Velcity_Kp*(Bias-Last_bias) 作用为限制加速度
                                                                       //Velcity_Ki*Bias  速度控制值由Bias不断积分得到 偏差越大加速度越大
        Last_bias=Bias;
        if(ControlVelocity>7200) ControlVelocity=7200;
        else if(ControlVelocity<-7200) ControlVelocity=-7200;
        return ControlVelocity; //返回速度控制值
}
```

```
int TargetVelocity=500;
int main(void)
 {
```

修改此值可以修改速度

图 5-3-7 电机的闭环控制

如图 5-3-7 所示，这一个电机的闭环控制代码，和 D153B 一致，利用的也是 PID 控制电机的 PWM 保持在我们想要的一个值。这里只是对编码器值的一个反馈，没有计算出速度与编码器的关系，所以无法进行速度的控制。例如：速度=编码器读数（利用中断进行一个周期读数）*读取频率*/倍频数/减速比/编码器精度*轮子的周长。