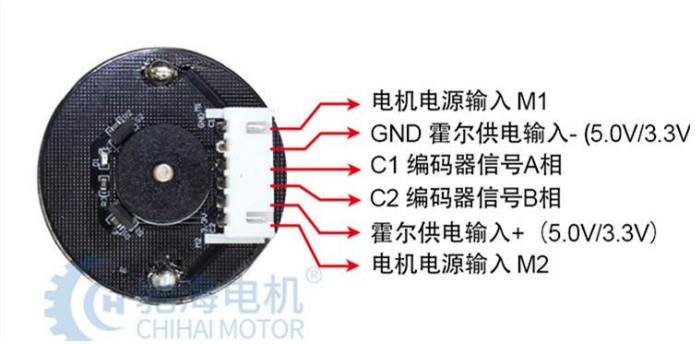
轻舟机器人驱动电机控制

AI 航 团队

上节讲了请求机器人的驱动电机选型及相关参数信息，本节将继续讲述如何使用该电机完成差速控制。直流有刷电机驱动方式比较简单，非常适合初学者入门训练。针对轻舟机器人使用的电机型号，属于直流减速电机，即齿轮减速电机，在普通的电机上配套齿轮减速箱，以便能够提供较低的转速和较大的力矩。减速电机一般在生产时已经由厂商进行集成组装好成套供货。

1. 直流减速电机

轻舟机器人采用的直流减速电机接口方式如下图，是一个带编码器的直流减速电机，减速比 1:30，编码器的作用是测速。一般包括六个接线端子，电机电源输入 M1 和电机电源输入 M2 是直流电机引脚，电机的旋转和速度调节只需这两个引脚即可。剩下中间的四个引脚是编码器，将在下一节进行详细介绍。



【外径尺寸】∮37mm长度尺寸依据参数不同长度不同

【轴长】21mmD长12mm

【轴径】直径6mmD字型轴

【电压】6-24V

【重量】290g左右不同减速比不同参数重量不同

【接线规格】XH2.54-6PIN端子连接头

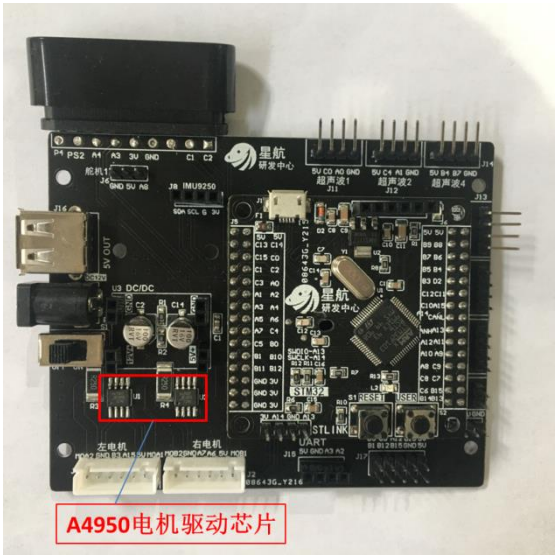
【编码器规格】AB双相编码器11线基本信号电压3.3V或5.0V

【基础脉冲数】绿色板**17 PPR** 黑色板 **11PPR** 已收到实物为准；

电机电源输入 M1 和 M2 接 12V 电压，我们使用 stm32 作为控制器，需要电机驱动模块实现电机驱动输出，为电机提供核定电压和大电流。

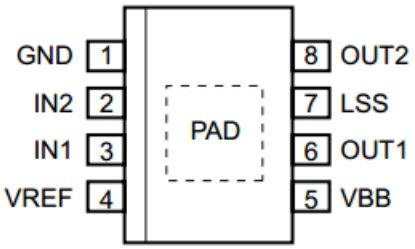
2. A4950 电机驱动模块介绍

轻舟机器人驱动板上集成了 A4950 电机驱动芯片，如下图所示：



A4950 设计用于直流电机的脉冲宽度调制（PWM）控制，其峰值输出电流为±3.5 A，最大工作电压为 40 V。提供输入端子，用于通过外部应用的 PWM 控制信号控制直流电机的速度和方向。提供内部同步整流控制电路，以降低脉宽调制操作期间的功耗。内部电路保护包括过电流保护、电机引线对地或电源短路、滞后热关机、VBB 欠压监测和交叉过电流保护，采用低剖面 8 针 SOICN 封装。

Pin-out Diagram



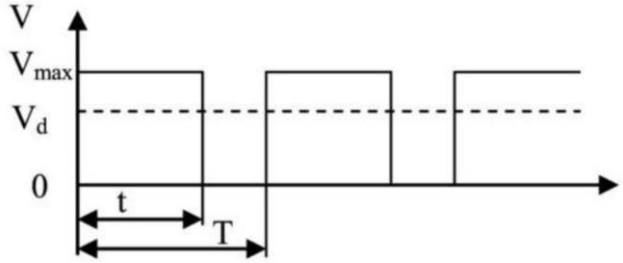
Terminal List Table

Number	Name	Function
1	GND	Ground
2	IN2	Logic input 2
3	IN1	Logic input 1
4	VREF	Analog input
5	VBB	Load supply voltage
6	OUT1	DMOS full bridge output 1
7	LSS	Power return – sense resistor connection
8	OUT2	DMOS full bridge output 2
–	PAD	Exposed pad for enhanced thermal dissipation

3. PWM 简介

PWM 是指将输出信号的基本周期 固定，通过调整基本周期内工作周期的大小来控制输

出功率的方法。在 PWM 驱动控制的系统中，按一个固定的频率来接通和断开电源，并根据需要改变一个周期内“接通”和“断开”时间的长短。因此，PWM 又被成为“开关驱动装置”。脉冲作用下，当电机通电时，速度增加；电机断电时，速度逐渐减少。只要按一定规律改变通断电时间，即可让电机转速得到控制。



设电机接通电源时，电机转速最大为 V_{max}

设占空比为：$D = \frac{t}{T} \times 100\%$

则电机的平均速度为：$V_d = V_{max} \times D$

式中：V_d表示电机的平均速度；$V_{\max}$表示电机全通电时的速度（最大）；D 是占空比。由公式可见，当改变占空比 D 时，就可以得到不同的电机平均速度，从而达到调速的目的。

占空比 D 表示了在一个周期 T 里边开关导通的时间 t 与周期的百分比。可见 D 的变化范围为：$0 \le D \le 1$.

在外部供电电源不变情况下（对应$V_{\max}$不变），输出电压的平均值（对应V_d大小）取决于占空比 D 的大小，改变 D 值也就可以改变输出电压的平均值，从而达到控制电机转速的目的，这就是 PWM 调速。

可能会有人问，这样信号总是通通断断的会不会造成电机抖动？实际上，我们给电机控制的 PWM 信号频率都是比较高的，而且电机本身就是感性部件，所以一般不会存在因为 PWM 信号导致的抖动问题。

占空比 D 的大小有 t 和 T 两个数值大小决定，所以一般有几种方法可以改变 D 的大小：定宽调频法（t 不变、T 改变）、定频调宽法（t 改变、T 不变）和调宽调频法（t 改变、T 改变）。在一般的微控制器中，经常使用定频调宽法来改变占空比大小。

4. stm32 代码实现

stm32 F103 的高级控制定时器 TIM1 和 TIM8 在基本定时器的基础上引入了外部引脚，可以输入捕获和输出比较功能。高级定时器能够完成输入捕获和输出比较的功能，输出比较包括翻转、强制为有效电平、PWM1 和 PWM2 等模式，其中 PWM 模式时最常用的。

脉冲宽度调制模式，具体来说就是利用微处理器的数字输出来对模拟电路进行控制的一种技术。在 stm32 中，由 TIMx_ARR 寄存器确定频率，由 TIMx_CCRx 确定占空比。PWM 模式分为模式 1 和模式 2，我们采用模式 1，即在向上计数时，TIMx_CNT<TIMx_CCR1 时通道 1 位有效电平，否则为无效电平；在向下计数时，TIMx_CNT>TIMx_CCR1 时通道 1 位无效电平，否则为有效电平。

头文件 motor.h，声明 PWMA1、PWMA1、PWB1、PWB2 对应为定时器 8 的比较寄存器，改变比较寄存器的值即可改变脉冲宽度。

```
#ifndef __MOTOR_H
#define __MOTOR_H
#include <sys.h>

#define PWMA1   TIM8->CCR2
#define PWMA2   TIM8->CCR1

#define PWMB1   TIM8->CCR4
#define PWMB2   TIM8->CCR3
void MiniBalance_PWM_Init(u16 arr,u16 psc);
endif
```

motor.c 文件主要完成定时器 8 的初始化，配置引脚和相关参数，主要包括定时器使能、外设时钟使能、配置引脚和引脚输出模式、周期值和分频值以及 PWM1 输出模式，具体如下：

```
#include "motor.h"

void MiniBalance_PWM_Init(u16 arr,u16 psc)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
    TIM_OCInitTypeDef    TIM_OCInitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM8, ENABLE);      //  使能 TIM8 的时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC , ENABLE);   //使能 GPIO 外设时钟使能

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8|GPIO_Pin_9; //配置输出引脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;                    //复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    TIM_TimeBaseStructure.TIM_Period = arr;      //设置自动重装载寄存器周期的值
    TIM_TimeBaseStructure.TIM_Prescaler =psc;   //设置用来作为 TIMx 时钟频率除数的预分频值
    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(TIM8, &TIM_TimeBaseStructure);   //初始化 TIMx 的时间基数单位

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;        //选择定时器模式:PWM1
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能
    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;   //输出极性:TIM 输出比较极性高
    TIM_OC1Init(TIM8, &TIM_OCInitStructure);   //根据 TIM_OCInitStruct 的参数初始化外设 TIMx
    TIM_OC2Init(TIM8, &TIM_OCInitStructure);
    TIM_OC3Init(TIM8, &TIM_OCInitStructure);
    TIM_OC4Init(TIM8, &TIM_OCInitStructure);

     TIM_CtrlPWMOutputs(TIM8,ENABLE);                        //MOE  主输出使能

    TIM_OC1PreloadConfig(TIM8, TIM_OCPreload_Enable);   //CH1 预装载使能
    TIM_OC2PreloadConfig(TIM8, TIM_OCPreload_Enable);   //CH2 预装载使能
    TIM_OC3PreloadConfig(TIM8, TIM_OCPreload_Enable);   //CH3 预装载使能
    TIM_OC4PreloadConfig(TIM8, TIM_OCPreload_Enable);   //CH4 预装载使能

    TIM_ARRPreloadConfig(TIM8, ENABLE); //使能 TIMx 在 ARR 上的预装载寄存器

    TIM_Cmd(TIM8, ENABLE);                  //使能 TIM
}
```

设置 PWM 比较寄存器值，通过改变通道 1 和通道 2 的值，当电机正转时 PWMA1 为脉冲高电平，PWMA2 脉宽为 7200-motor_a；电机反转时 PWMA1 为 7200+motor_a，PWMA2 为 7200。

```
/**************************************************************************
函数功能：赋值给 PWM 寄存器
入口参数：PWM
返回  值：无
**************************************************************************/
void Set_Pwm(int motor_a,int motor_b)
{
        if(motor_a>0)            PWMA1=7200,PWMA2=7200-motor_a;
          else                    PWMA2=7200,PWMA1=7200+motor_a;

        if(motor_b>0)            PWMB1=7200,PWMB2=7200-motor_b;
          else                    PWMB2=7200,PWMB1=7200+motor_b;
}
```

```
/**************************************************************************
函数功能：限制 PWM 赋值
入口参数：幅值
返回  值：无
**************************************************************************/
void Xianfu_Pwm(int amplitude)
{
    if(Motor_Left<-amplitude)  Motor_Left=-amplitude;     //限制最小值
    if(Motor_Left>amplitude)   Motor_Left=amplitude;      //限制最大值
    if(Motor_Right<-amplitude) Motor_Right=-amplitude;    //限制最小值
    if(Motor_Right>amplitude)  Motor_Right=amplitude;     //限制最大值
}
```

Motor_Left 和 Motor_Right 通常由控制程序计算得出，在轻舟机器人当中，控制板根据上位机 ROS 系统发来的控制速度和角度变量，经过数学模型后解析出左后轮、右后轮速度和舵机转向角度，根据左右后轮速度和编码器的反馈值进行偏差计算，最后根据偏差进行闭环运算计算得出控制量 Motor_Left 和 Motor_Right，即 PWM 脉冲宽度。

```
/**************************************************************************
函数功能：增量 PI 控制器
入口参数：编码器测量值，目标速度
返回  值：电机 PWM
根据增量式离散 PID 公式
pwm+=Kp[e（k）-e(k-1)]+Ki*e(k)+Kd[e(k)-2e(k-1)+e(k-2)]
e(k)代表本次偏差
e(k-1)代表上一次的偏差  以此类推
pwm 代表增量输出
在我们的速度控制闭环系统里面，只使用 PI 控制
pwm+=Kp[e（k）-e(k-1)]+Ki*e(k)
**************************************************************************/
int Incremental_PI_Left (int Encoder,int Target)
{
    static int Bias,Pwm,Last_bias;
    Bias=Encoder-Target;                  //计算偏差
    Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias;   //增量式 PI 控制器
    if(Pwm>7200)Pwm=7200;
    if(Pwm<-7200)Pwm=-7200;
    Last_bias=Bias;                       //保存上一次偏差
    return Pwm;                           //增量输出
}
int Incremental_PI_Right (int Encoder,int Target)
{
    static int Bias,Pwm,Last_bias;
    Bias=Encoder-Target;                  //计算偏差
    Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias;   //增量式 PI 控制器
    if(Pwm>7200)Pwm=7200;
    if(Pwm<-7200)Pwm=-7200;
    Last_bias=Bias;                       //保存上一次偏差
    return Pwm;                           //增量输出
}
```

```
//在控制代码中调用以上电机控制函数的语句如下

Kinematic_Analysis(Velocity,-Angle);  //小车运动学分析，得到 Target_Left、Target_Right、Servo
Motor_Left=Incremental_PI_Left(Encoder_Left,Target_Left);       //速度闭环控制计算左电机最终 PWM
Motor_Right=Incremental_PI_Right(Encoder_Right,Target_Right); //速度闭环控制计算右电机最终 PWM
Xianfu_Pwm(6900);                       //PWM 限幅
Set_Pwm(Motor_Left,Motor_Right,Servo);      //赋值给 PWM 寄存器  }
```