轻舟机器人舵机介绍及控制

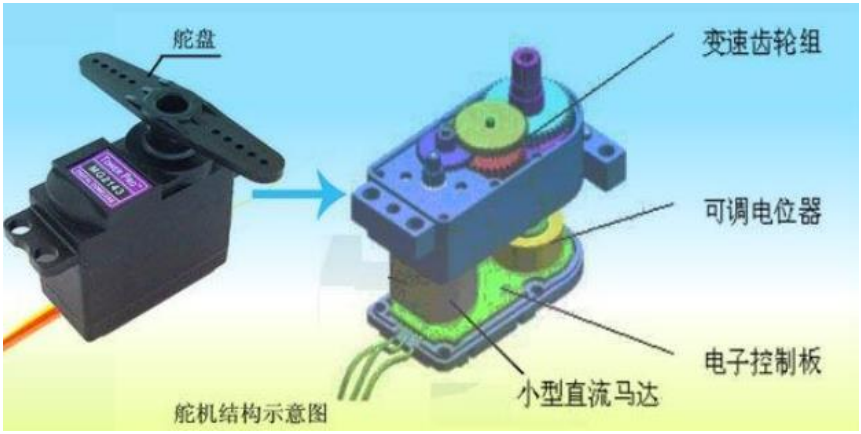
AI 航 团队

舵机是一种位置（角度）伺服的驱动器，适用于那些需要角度不断变化并可以保持的控制系统。使用 stm32 控制机器时，经常要用到舵机，如使某个部位转到特定的角度，或者在行进过程中的方向控制，由于其可以通过程序连续控制转角，被广泛应用在智能小车以实现转向以及机器人各类关键运动中。

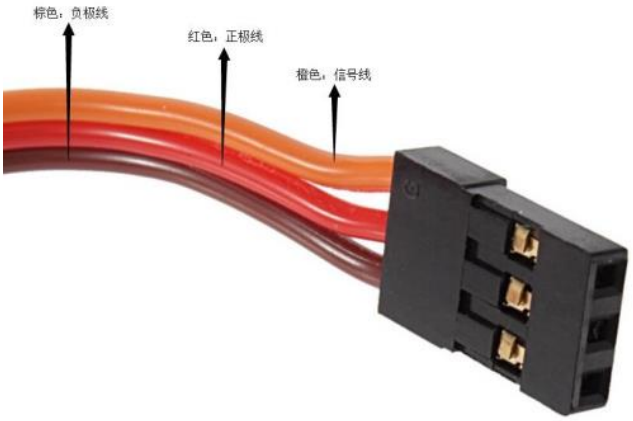
在设计智能小车中，舵机是小车的转向控制机构，具有体积小、力矩大、外部机械设计简单、稳定性高等特点，轻舟机器人正是使用舵机来完成转向功能的。

1.舵机原理

舵机一般由变速齿轮箱，电位器，电路板与直流电机组成。电机的高速、短周期运动由齿轮箱转换为慢速、长周期的运动，最终到最外端的齿轮，齿轮的转动带动电位器转动，电位器的电位与信号线进行比较，从而实现转动到特定角度的功能。



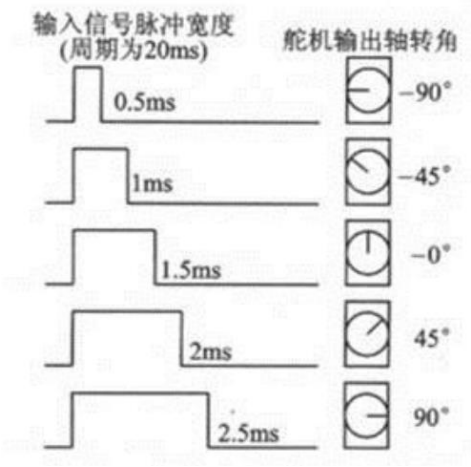
舵机的输入线有三根，分别是电源线、地线和控制信号线，一般来说中间线（红色）为电源线，棕色或黑色的线为地线，另外一根线为控制信号线（橘黄色或白色）。



舵机自带的控制电路接受来自信号线的控制信号，控制电机转动，电机带动一系列齿轮组，减速后传送至输出舵盘，舵机的输出轴和位置反馈电位计是相连的，舵盘转动的同时，带动位置反馈电位计，电位计将输出一个电压信号到控制板，进行反馈，然后控制电路板根据所在位置决定电机转动的方向和速度，从而达到目标停止。

2. 舵机的控制

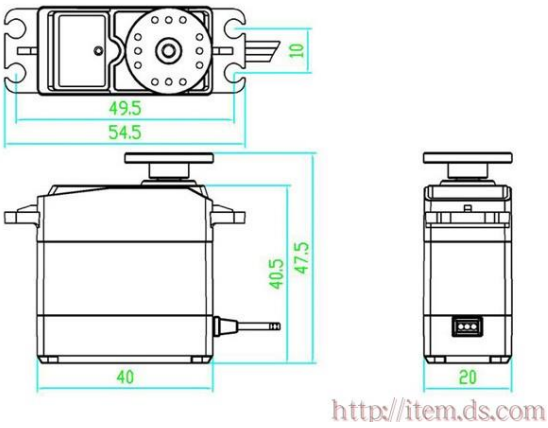
舵机的控制由一个脉冲宽度调制信号(PWM）来实现，该信号由 stm32 发出。通常来说，舵机的控制信号周期为20ms的脉宽调制信号，其中脉冲宽度从0.5-2.5对应舵盘位置的0-180度，呈线性变化，也就是说给它提供一定的脉宽，它的输出轴就会保持在一定对应角度上，无论外界力矩怎么改变，直到给它提供另外宽度的脉冲信号，它才会改变输出角度到新的位置上。



3. 轻舟机器人舵机参数介绍

轻舟机器人前轮转向所用的电机为 DSServo 达盛舵机，型号为 DS3230，适用于航模、车模、船模及机器人的小型舵机，额定扭矩 3N.m，转动角度 270 度。

具体参数及尺寸如下：



产品介绍	DS3230
齿轮	不锈钢材质
脉冲宽度	500-2500us
信号死区dead	2 us
工作频率Frequnce	50－330hz
电机motor	铁芯电机
电压working Voltage	4.8v-7.2v
速度5v Speed	0.16sec/60°
速度6v Speed	0.14sec/60°
速度6.8v Speed	0.12sec/60°
扭矩5v Torque	27 kg.cm
扭矩6v Torque	30 kg.cm
扭矩6.8v Torque	32 kg.cm
尺寸 Size	40 * 20 *40.5 mm
重量 Weight	65 g
线长 Wire length	JR 300mm

4. 轻舟机器人舵机控制程序

```
//Motor.h 文件，声明电机相关寄存器及初始化函数
#ifndef __MOTOR_H
#define __MOTOR_H
#include <sys.h>

#define PWMA1     TIM8->CCR2   //左后轮驱动电机 PWM
#define PWMA2     TIM8->CCR1   //左后轮驱动电机 PWM

#define SERVO     TIM1->CCR1    //舵机引脚

#define PWMB1     TIM8->CCR4   //右后轮驱动电机 PWM
#define PWMB2     TIM8->CCR3   //右后轮驱动电机 PWM

void MiniBalance_PWM_Init(u16 arr,u16 psc); //声明后轮驱动电机初始化函数
void Servo_PWM_Init(u16 arr,u16 psc);          //声明舵机初始化函数
#endif
```

```
//motor.c 文件，舵机初始化函数
/**************************************************************************
函数功能：舵机 PWM 以及定时中断初始化
入口参数：入口参数：arr：自动重装值   psc：时钟预分频数
返回  值：无
**************************************************************************/
void Servo_PWM_Init(u16 arr,u16 psc)
{
	GPIO_InitTypeDef GPIO_InitStructure;
	TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
	TIM_OCInitTypeDef  TIM_OCInitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA , ENABLE);  //使能 GPIO 外设时钟使能

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;               // TIM1_CH1
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  //复用推挽输出
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	TIM_TimeBaseStructure.TIM_Period = arr; //设置下一个更新事件装入活动的自动重装载寄存器周期值
	TIM_TimeBaseStructure.TIM_Prescaler =psc; //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(TIM1, &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; //选择定时器模式:TIM 脉冲宽度调制模式 1
	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(TIM1, &TIM_OCInitStructure);    //根据 TIM_OCInitStruct 中指定的参数初始化外设 TIMx

	TIM_CtrlPWMOutputs(TIM1,ENABLE);                    //MOE 主输出使能
	IM_OC1PreloadConfig(TIM1, TIM_OCPreload_Enable);   //CH4 预装载使能
	TIM_ARRPreloadConfig(TIM1, ENABLE);                 //使能 TIMx 在 ARR 上的预装载寄存器
	TIM_Cmd(TIM1, ENABLE);                              //使能 TIM
}


//control.c 文件，包含控制算法
/**************************************************************************
函数功能：小车运动数学模型
入口参数：速度和转角
返回  值：无
**************************************************************************/
void Kinematic_Analysis(float velocity,float angle)
{
		Servo=SERVO_INIT-angle*K;          //舵机转向   angle*
		if(Servo > 2100){                  //  限幅最大 2100
			Servo = 2100;
			angle = (double)(Servo - SERVO_INIT)/K;
		}
		else if(Servo < 1000){                    //限幅最小 1000
			Servo = 1000;
			angle = (double)(Servo - SERVO_INIT)/K;
		}
		Tand = tan(angle/57.3);;//(int)tan(angle);
		Target_Left=-velocity*(1-T*Tand/2/L);
		Target_Right=velocity*(1+T*Tand/2/L);      //后轮差速
		Servo=SERVO_INIT-angle*K;                  //舵机转向
}
```