

2023 年第九届全国应用型
-----人才综合技能大赛-----

八足仿蟹机器人

设计说明书

目录

一. 绪论	1
1.1 选题的目的和意义	1
1.2 生物分析	1
二. 骨架的零件工程	2
2.1 Solidworks 建模	2
2.2 模型展示与原理解析	3
三. 整体的功能设计	4
3.1 功能说明	4
3.2 元件的选择	5
四. 运动与结构计算	7
4.1 钳子相关计算	7
4.2 运动计算	7
五. 控制部分	8
5.1 控制逻辑	8
5.2 控制界面	8

一. 绪论

1.1 选题的目的和意义

以仿生为核心研究，做出以机械结构优势达到一些普通小车不能或不方便行走的地方前进，在一些人力不方便勘探的地方（例如化工工厂，或者一些粉尘或者有害物质遍布的工作环境）进行实地勘探，采集数据等工作，达到节省人力，保护人的目的。

在生活中，最常见的移动机器多数采用轮子滚动进行移动，而在一些特殊环境（例如沙尘遍布，或者一些腐蚀液体遍布的地面空间）使用轮子进行移动易将沙尘和腐蚀液卷入机械体内，对机械结构有损害，严重时致机械结构短路损坏等，这种环境下以轮子进行移动显然不是一个优良的选择。移动都成困难，机器正常工作就更难实现，若是采用建立轨道，并让机器在上面进行移动，虽可避免损坏机器，但成本花费就会提升很多。

故我们设计了一种由机械腿传动，仿生螃蟹运动的机器，大大降低了机器的制造成本，使机器在这些恶劣环境下也能正常工作。用仿生螃蟹腿代替了传统轮子滚动的移动方式，减少了粉尘或者液体进入机械体内从而影响机械本身工作的可能性。

1.2 生物分析

- （1） 分析螃蟹生理结构和运动机理，为仿生螃蟹结构设计提供数据基础，分析各个关节的最大驱动扭矩，为关节驱动器的选择做准备。
- （2） 分析螃蟹在各种情况下的运动速度，对螃蟹行走步态进行分析，为电动机的选择提供可依靠的数据。
- （3） 仿生多足蟹整体设计

通过对螃蟹的外貌观察与资料的查找，我们得到以下结论:螃蟹有 8 条腿，也就是它们的 8 条步足。这 8 条腿平均分布在它们身体的两侧，一边 4 条。由于步足的构造比较特殊，关节只能上下活动，所以这些步足只能帮助螃蟹左右活动，而不能帮助它们前后活动。除了步足之外，螃蟹还有 2 条螯足，主要作用是挖洞、防御、进攻等。由于主要作用不是活动，所以不能将这 2 条螯足算作它们的腿，螃蟹蟹螯的钳子部分一半固定，一半活动。对腿部机构进行了自由度数的选择和布置，主、

被动动关节的确定，设计曲柄摇杆机构作为腿部运动结构，同时设计了由 3 个舵机组成的蟹。

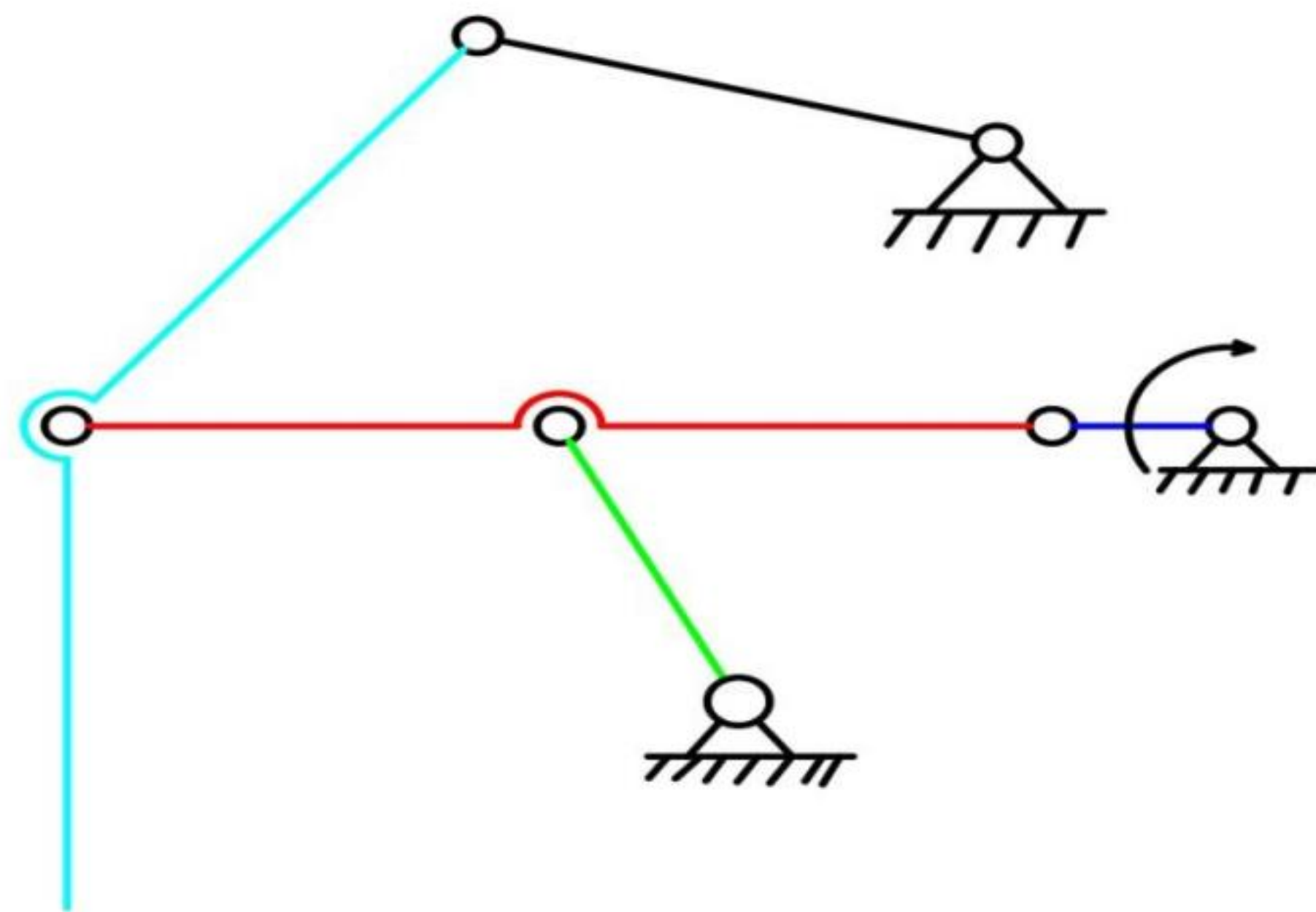
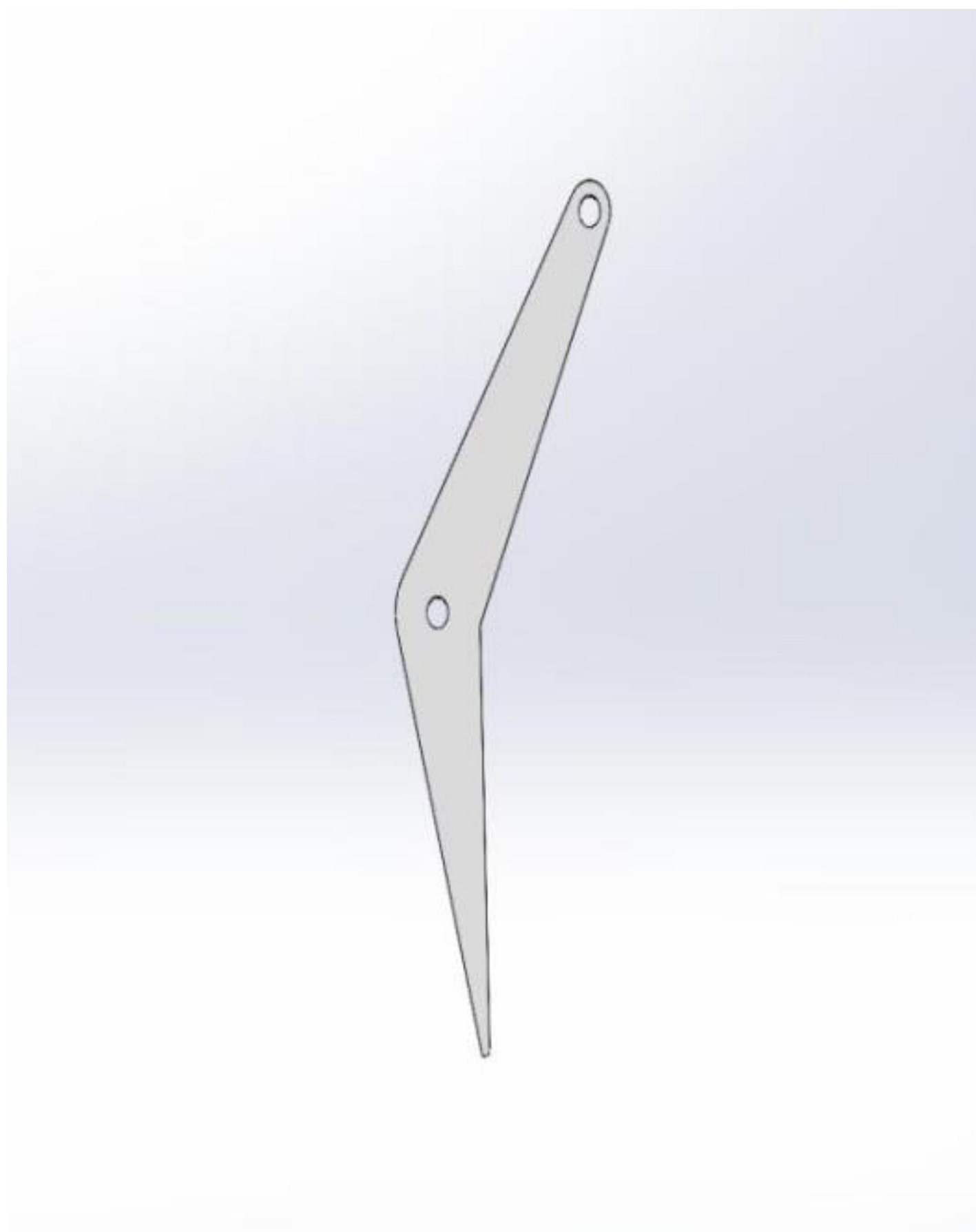


图 1 腿部的基本结构图

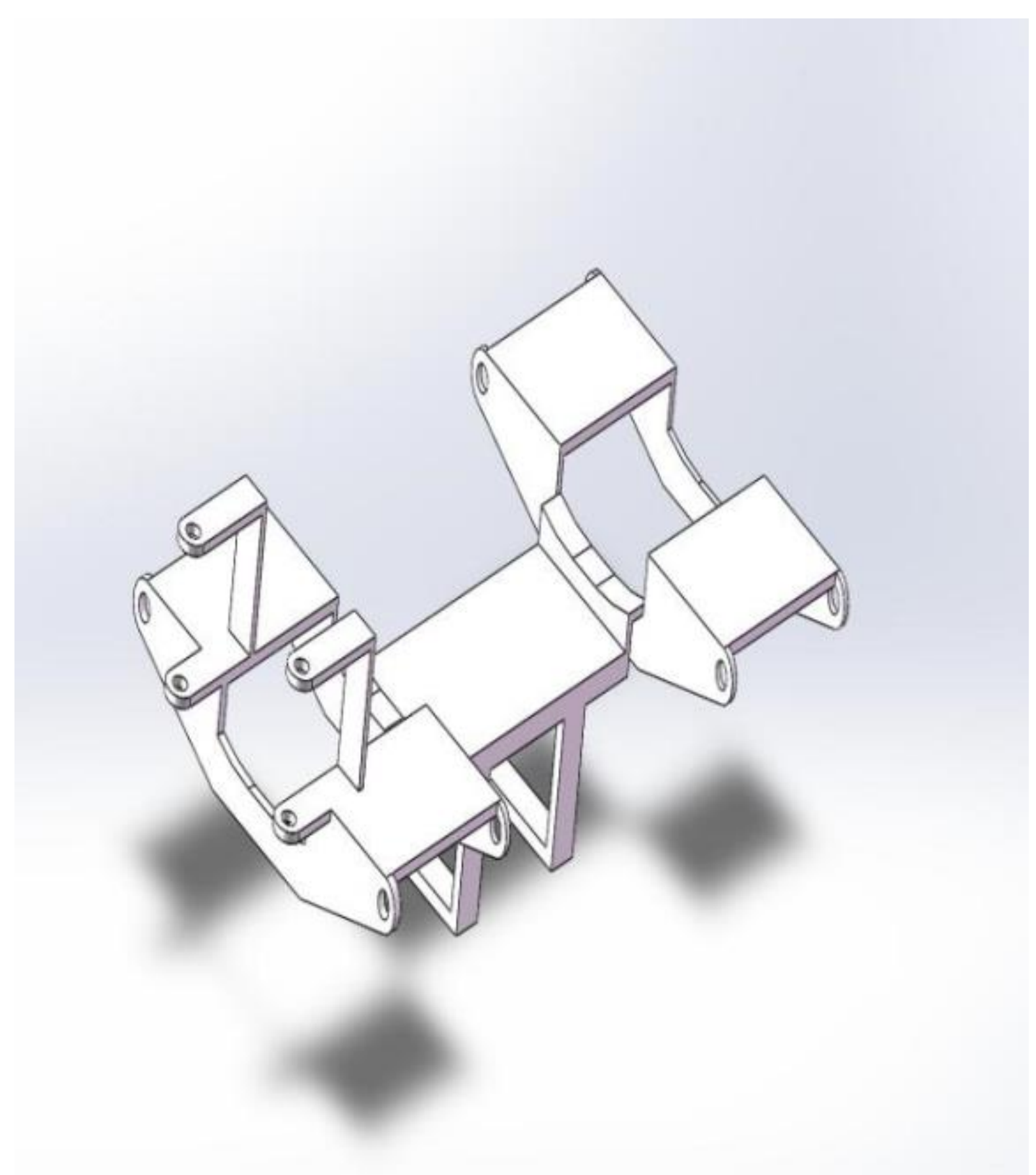
二. 骨架的零件工程

使用 SolidWorks、三维软件绘制零件，使用 SolidWorks 三维软件把这些零件进行装配，装配成整个三维模型。

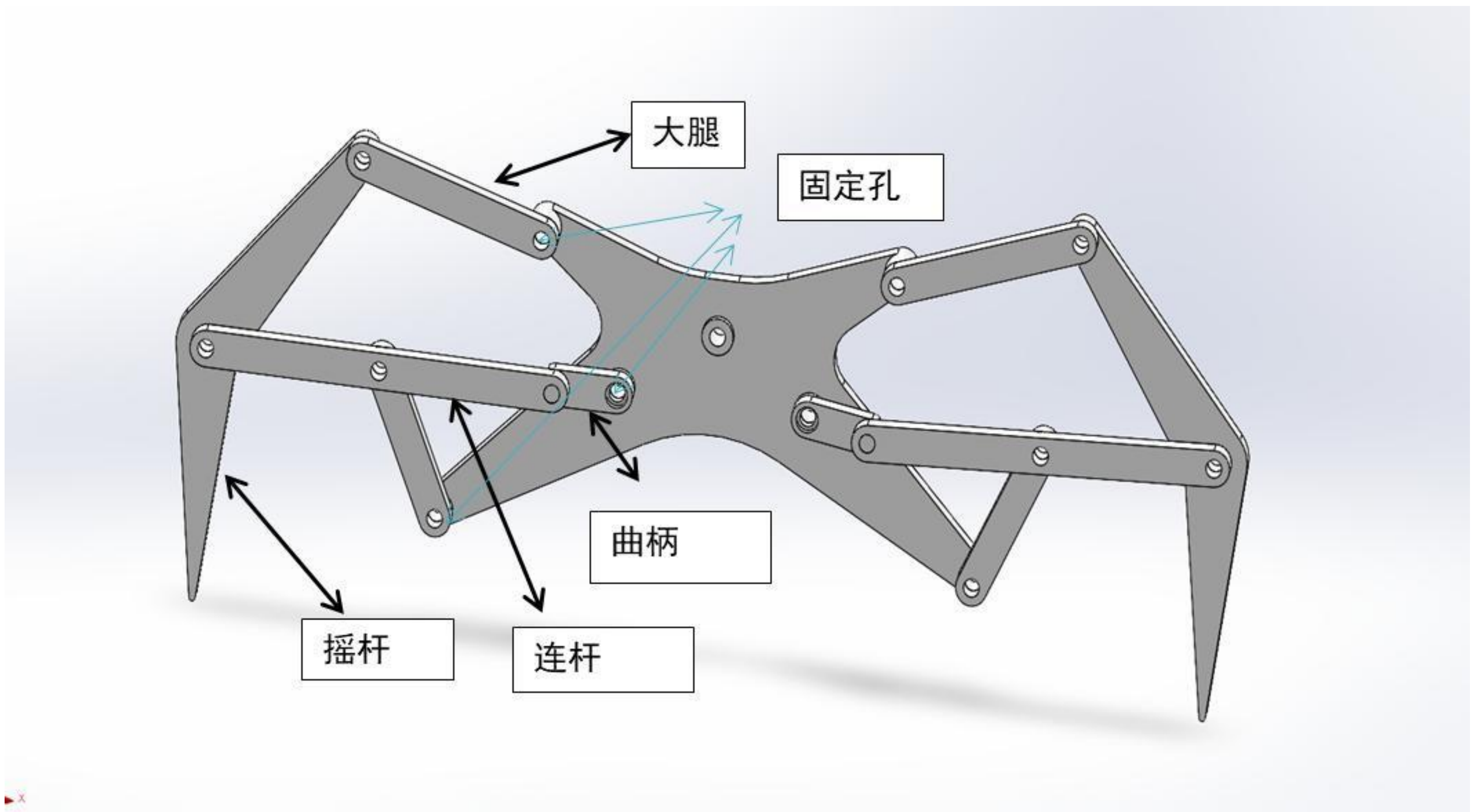
2.1 Solidworks 建模



a. 摇杆模型



b. 身体模型



c.其中一排腿的整体结构（整个机器含有 4 个此机构）图 2Solidworks
建模图

2.2 模型展示与原理解析

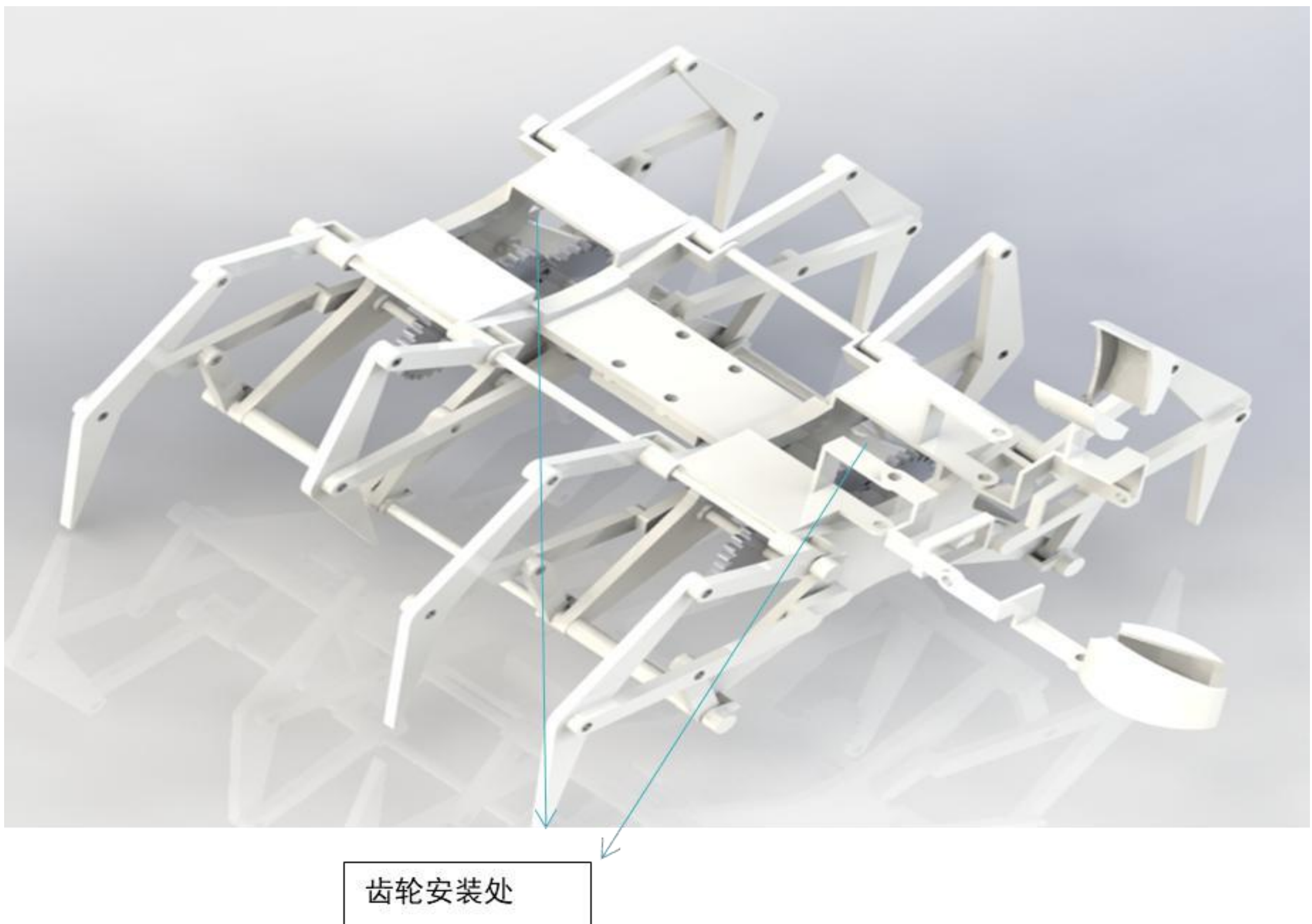


图 3 齿轮安装位置示意图

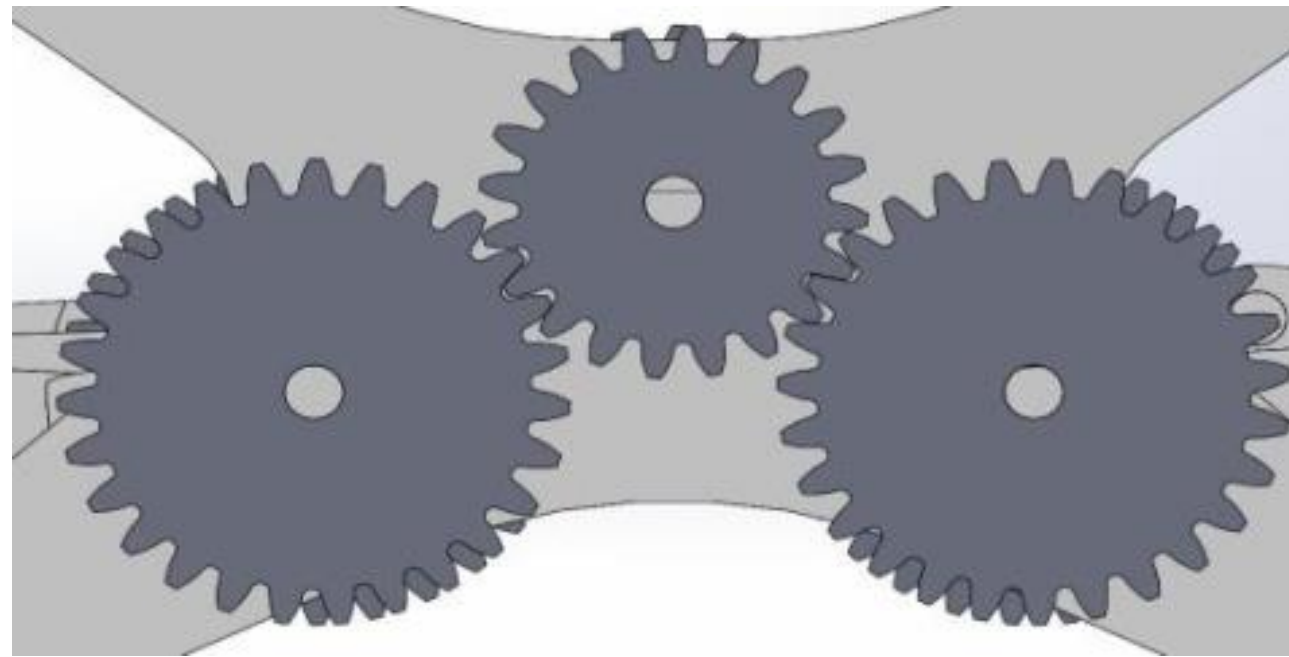


图 4 齿轮布置方式原理解析如

下：

1. 中间的为主动轮与电机直接相连，电机转动，主动轮转动从动轮随之转动，从动轮与腿部得曲柄相连，二者做同轴转动。曲柄转动带动摇杆，摇杆转动，机器移动。
(腿部原理可见图 2 (c))
2. 每三个齿轮为一个齿轮组，每一个齿轮组由一个电机驱动，每一个齿轮组带动两排腿运动。

三. 整体的功能设计

3.1 功能说明

3.1.1 基本功能：

- (1) 较为精确的夹取样本，并且可放回收集箱中（样本采集）。
- (2) 一电机正传，另一个电机反转可实现转弯。
- (3) 电机正转反转（机器的往复运动）。

3.1.2 钳子的结构：

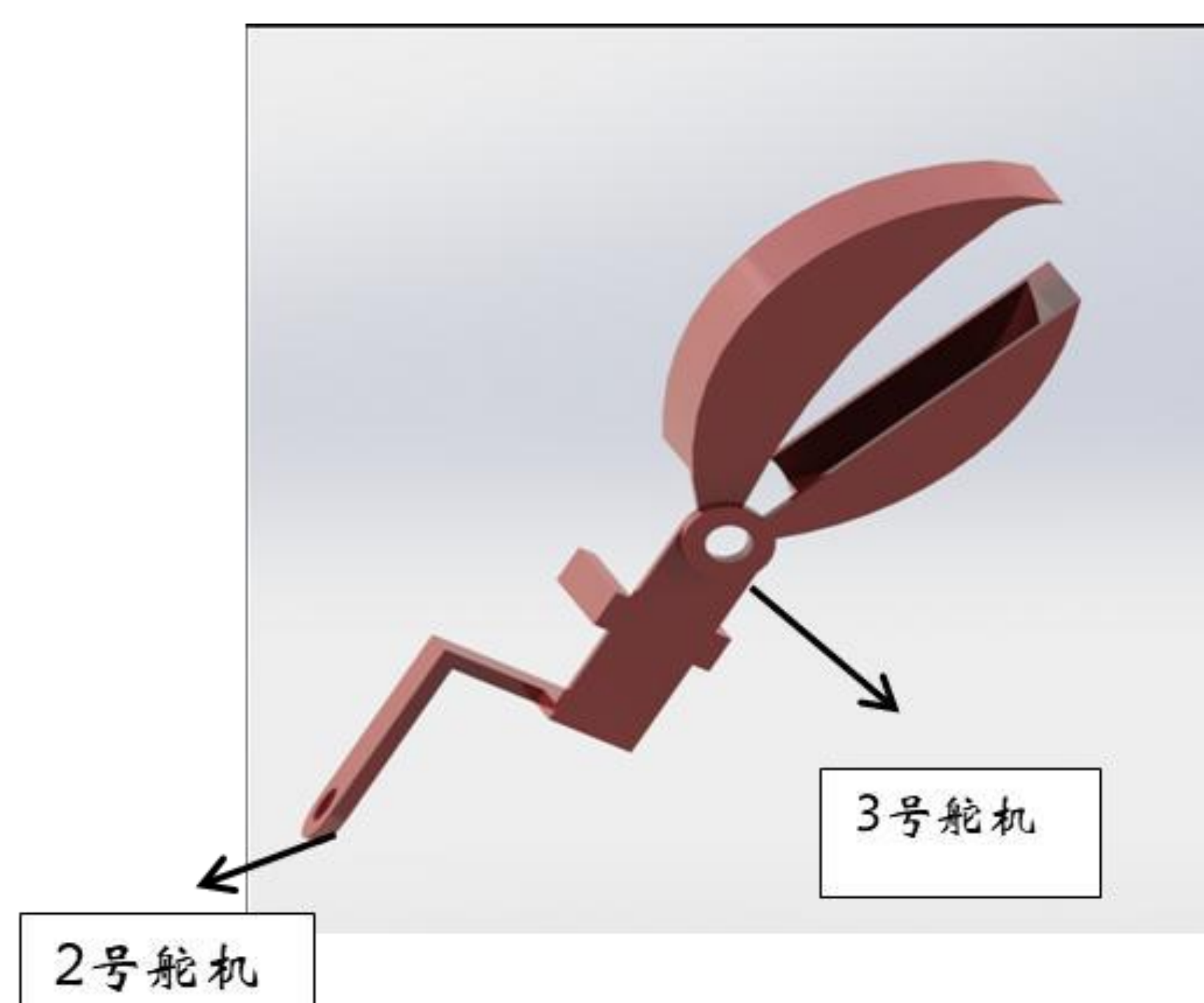


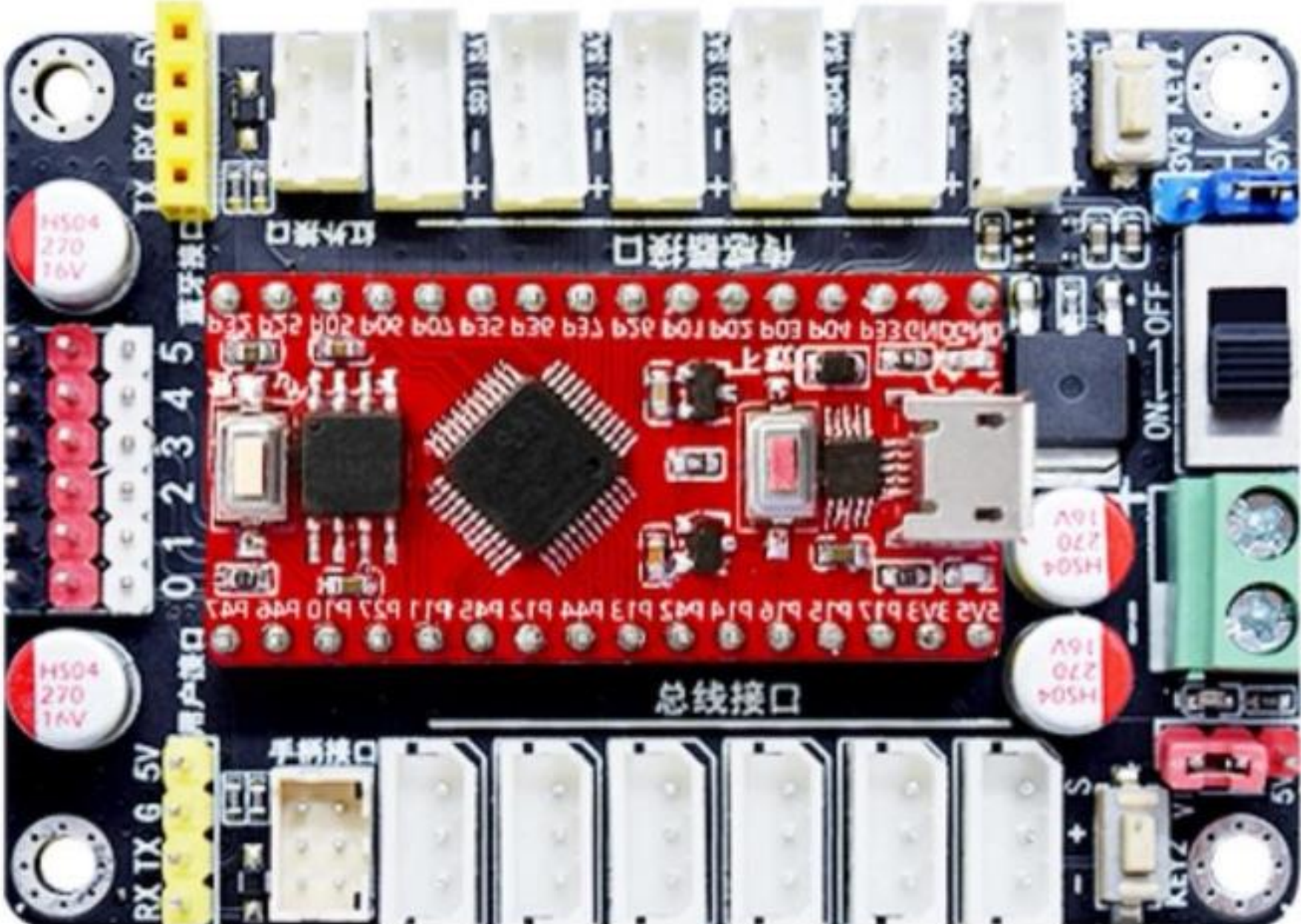
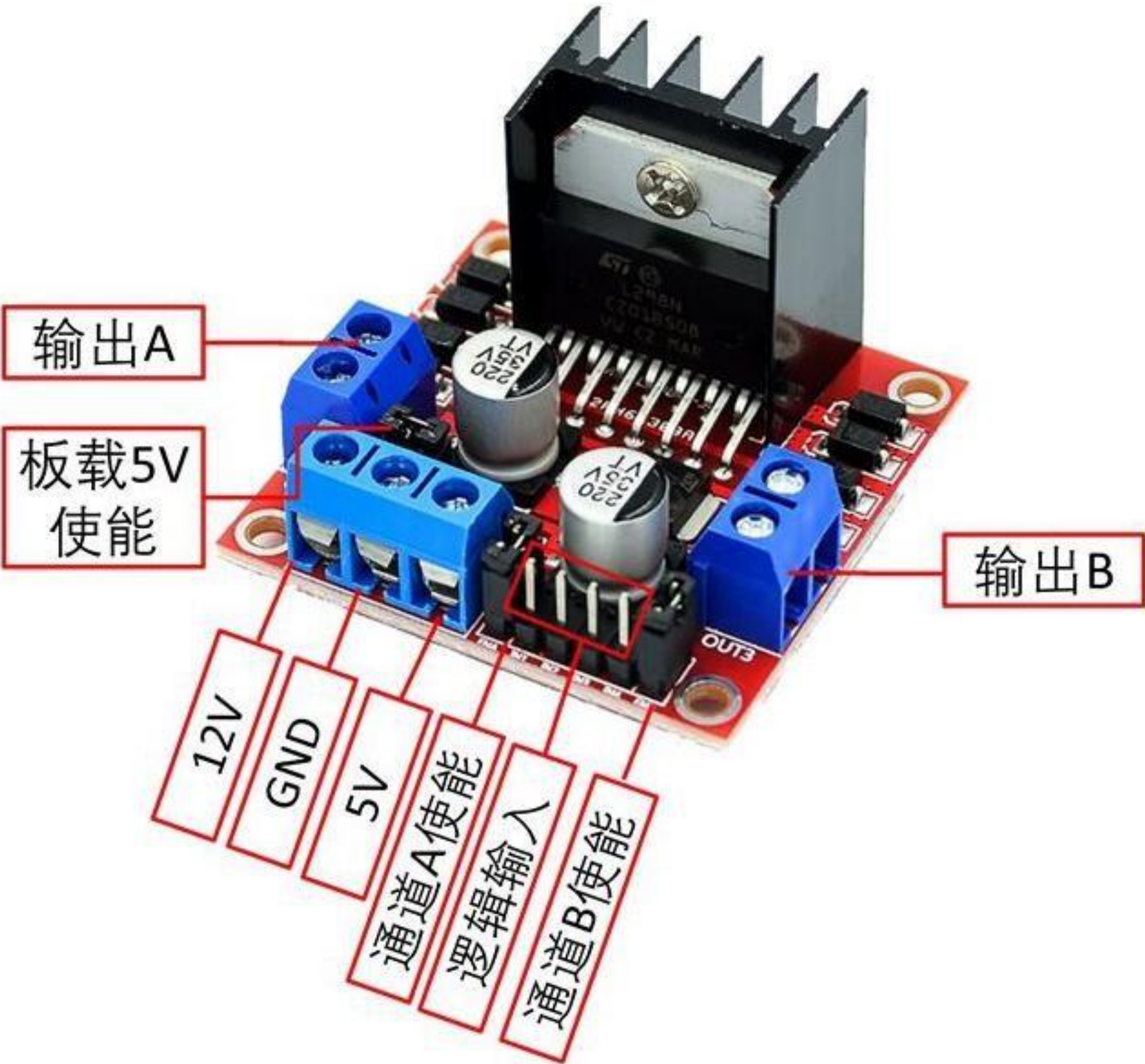
图 5 钳子的结构示意图


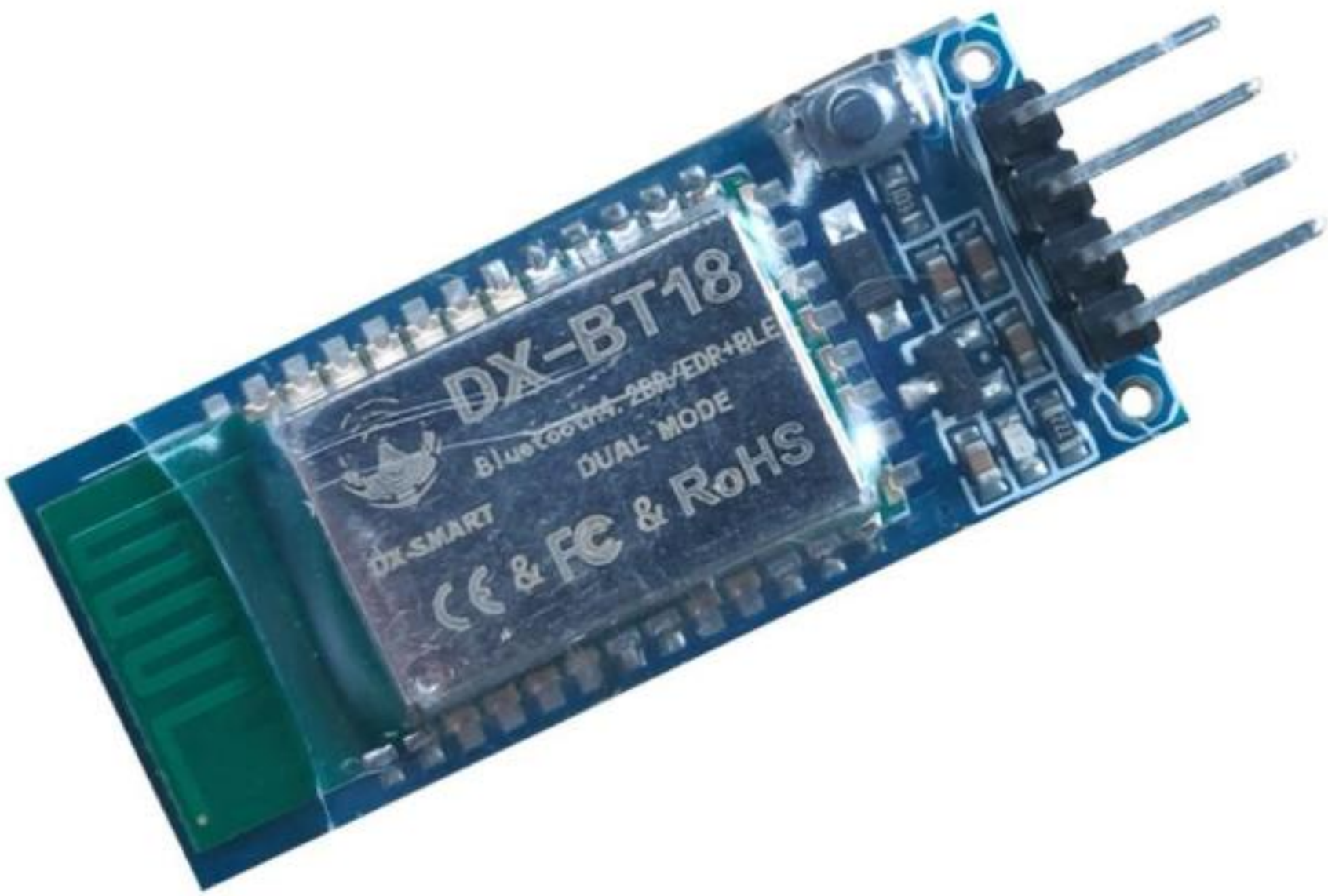
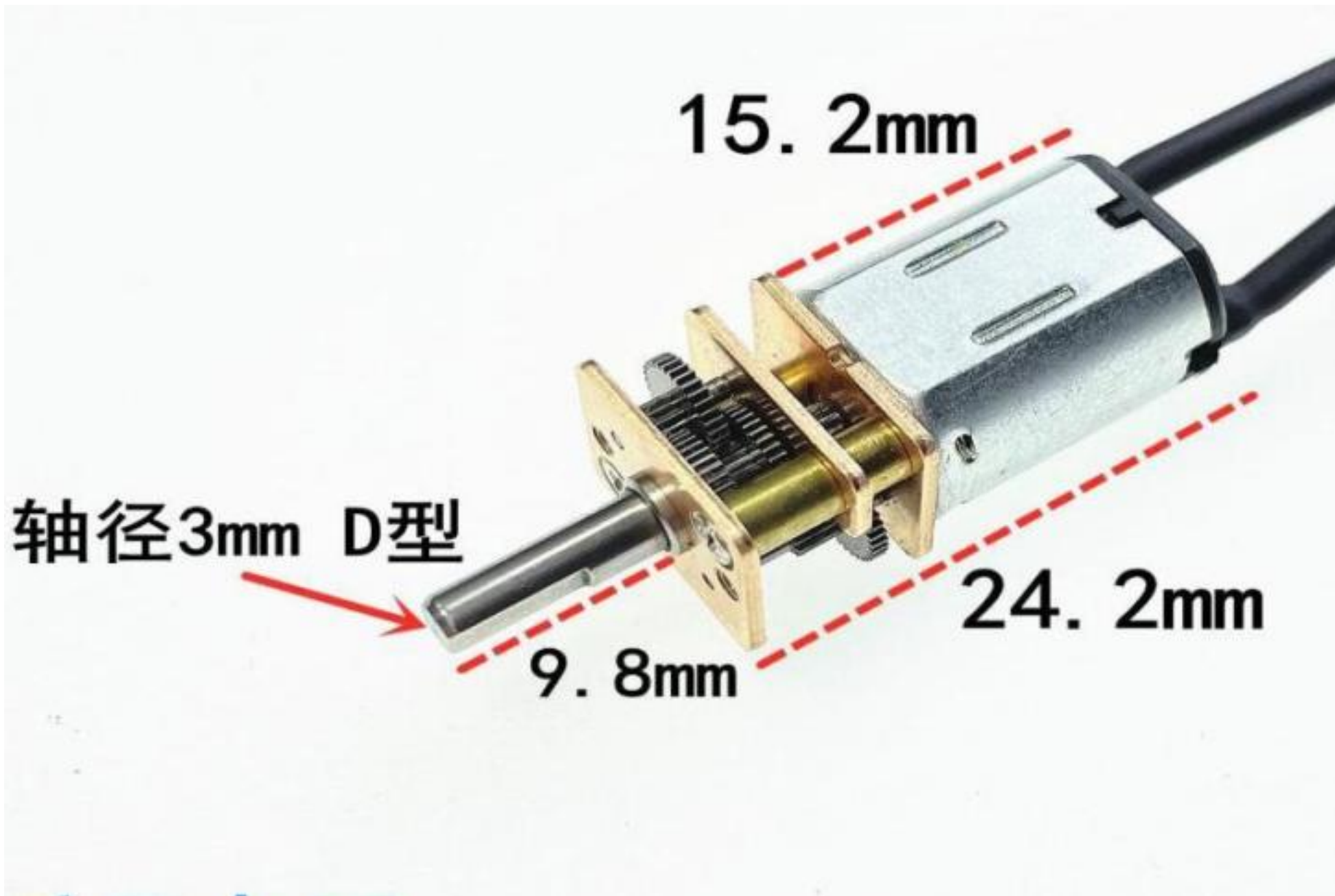
在钳子的结构中，由 3 个舵机配合使用使其拥有 3 自由度。1 号舵机与身体相连，2 号舵机放在螯中间作为关节，3 号舵机控制钳子的开合。其中：

- 1 号舵机（与身体直接相联的部分）：0~180 度；
- 2 号舵机：0~360 度；
- 3 号舵机：0~180 度。

3.2 元件的选择

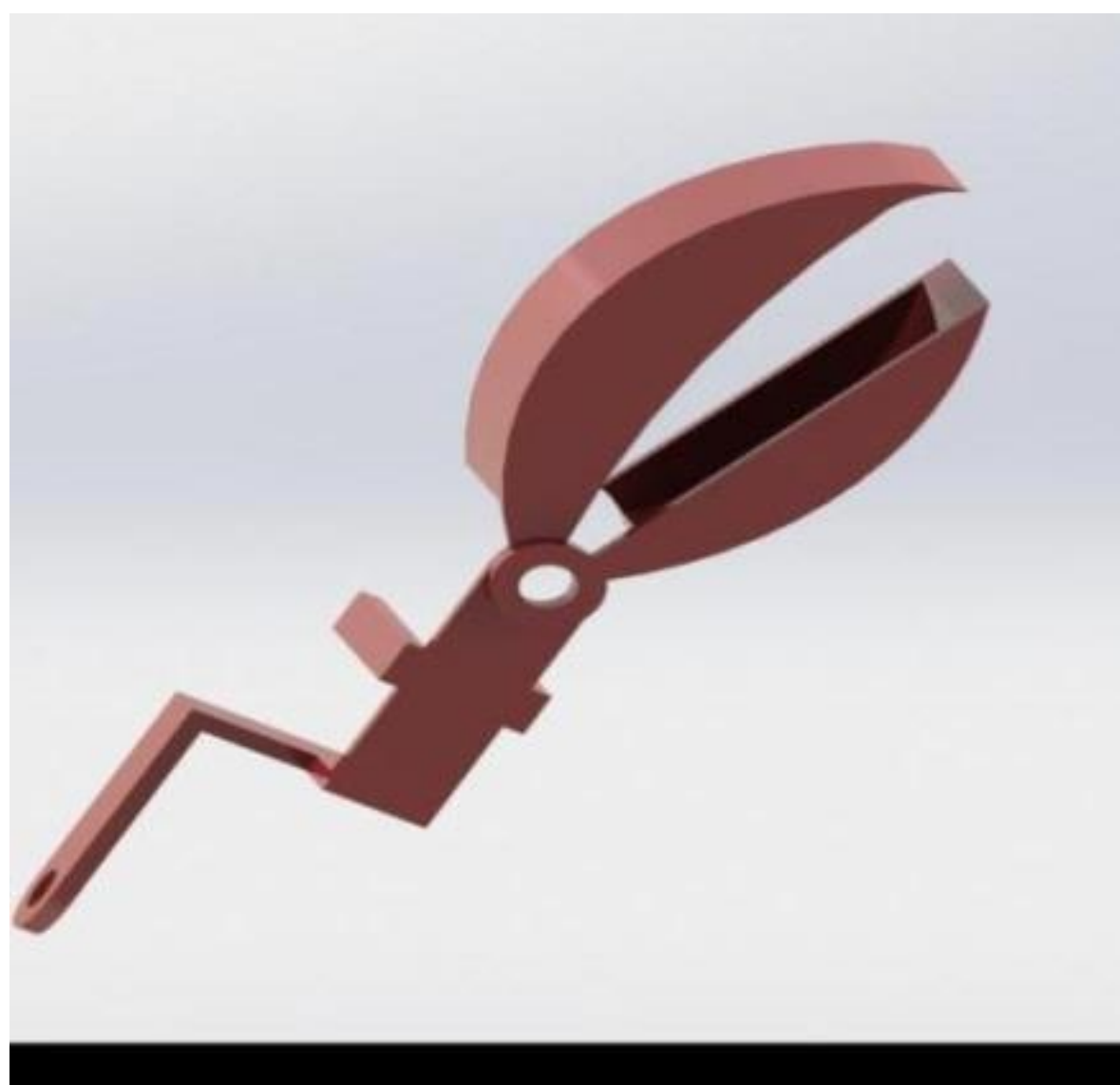
表 1 元件的选择

元件	元件名称及选型
	51 控制模块
	直流电机驱动模块

	<div data-bbox="1381 359 1806 816"><p>MG90S 舵机</p><p>基本参数：</p><p>重量：9g</p><p>堵转扭矩：1.3kg/cm 空载转速：0.16/60° 工作电压：3~6v</p></div>
	<div data-bbox="1430 949 1755 1038"><p>蓝牙模块</p></div>
 <p>15.2mm</p> <p>轴径3mm D型</p> <p>9.8mm</p> <p>24.2mm</p>	<div data-bbox="1325 1647 1860 2211"><p>9V 直流减速电机</p><p>基本参数：轴径：3mm</p><p>输出轴长度：9.8mm</p><p>适用电压：≤9V</p><p>转速：20~120 转/min</p></div>

四. 运动与结构计算

4.1 钳子的相关计算



钳子的长度约为 50mm，舵机的扭矩为 1.3g/cm. 经换

算得扭矩为 12.7N/cm.

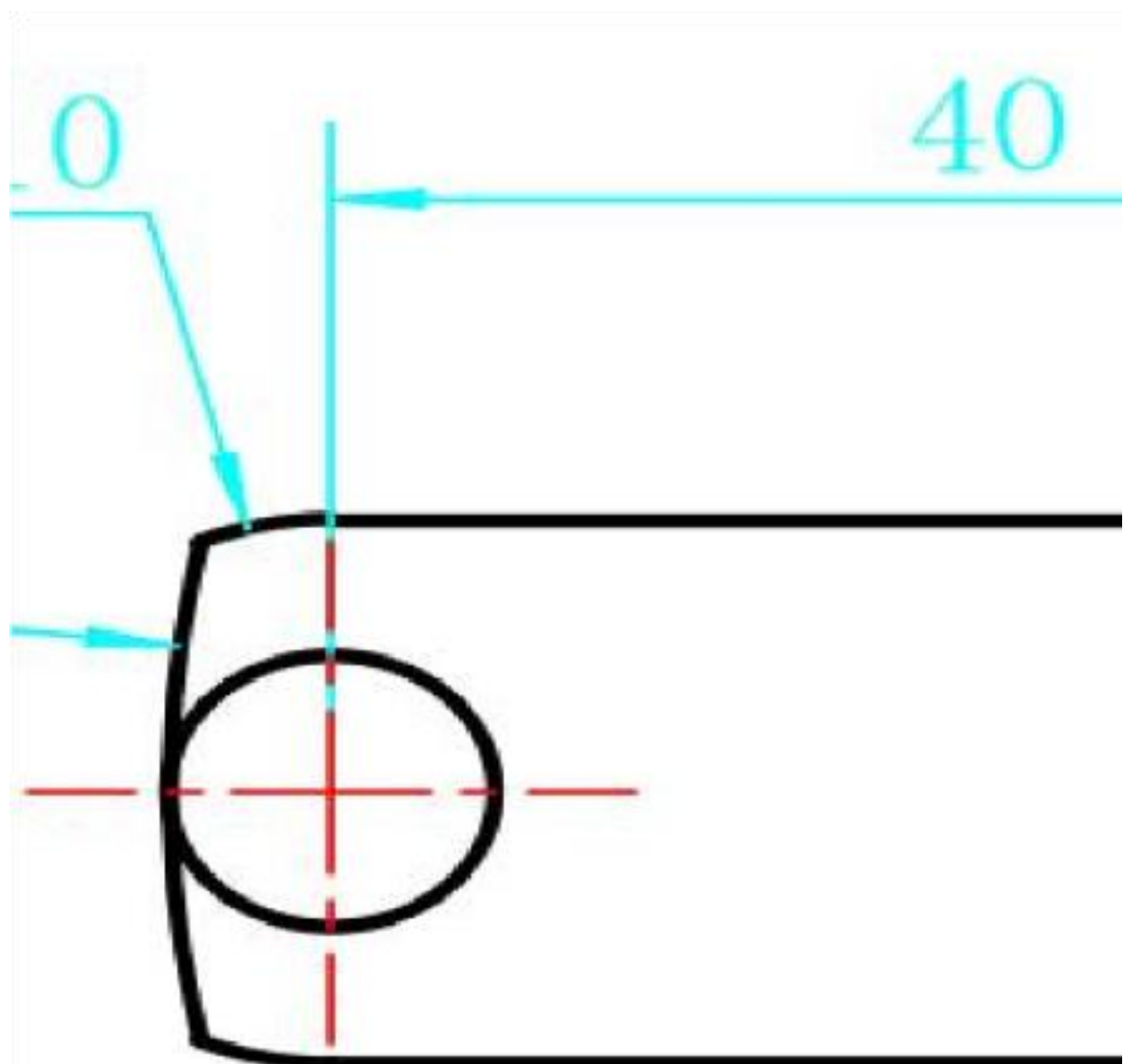
由公式得 $F = M / L$ 得钳子末端输出的力为 2.54N

$$\frac{M}{Ld(L)} = \frac{M}{L} \frac{1}{d(L)} = \frac{M(\ln 5 - \ln 0.5)}{0.5} = 27.94$$

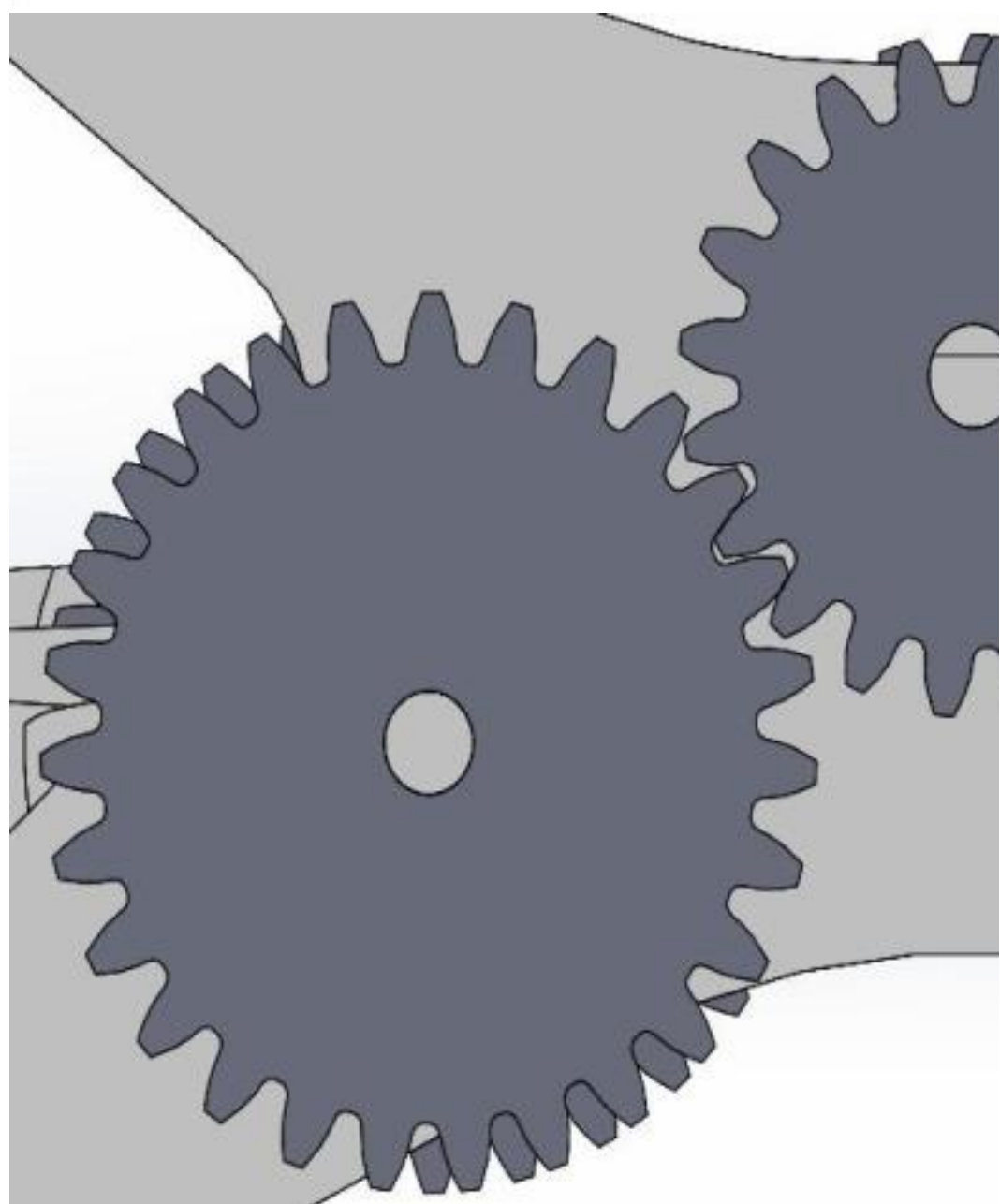
整条钳子可输出的力为 27.94N

$\omega = 375$ 钳子的空载转速为 375°/s

4.2 运动计算



曲柄的长度为
4cm



小齿轮参数

模数: $m=1.5$ 齿

数: $z=20$

大齿轮参数

模数: $m=1.5$ 齿

数: $z=27$

已知电动机转速为 20~120 转/分，则在理想状态下主动轮的转速为 20~120 转/分，根据公式 $n_1/n_2 = z_2/z_1$ ，得从动轮转速为：

$$n_2 = z_1 / z_2 * n_1 = 14.8 \sim 88.9 r/min$$

从动轮每转一圈机器就前进 4cm，则理论上(不考虑零件间的摩擦，机械效率为 100%) 每分钟可前进 59cm~352cm。

$$V(\text{min})=L/t=0.0098\text{m/s}$$

$$V(\text{max})=L/t=0.058\text{m/s}$$

五. 控制部分

5.1 控制逻辑

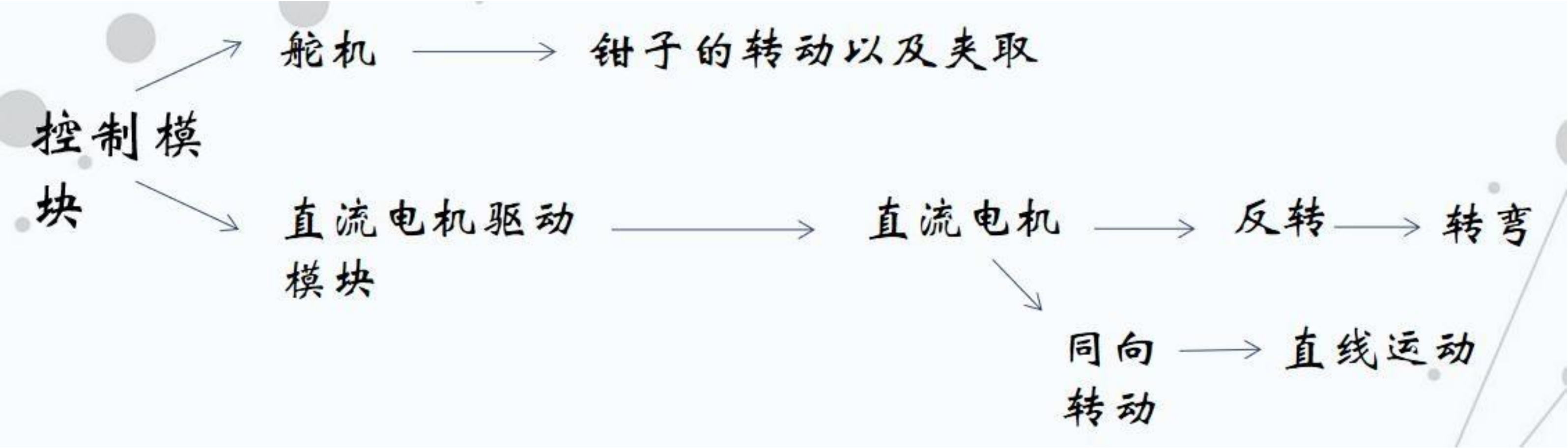
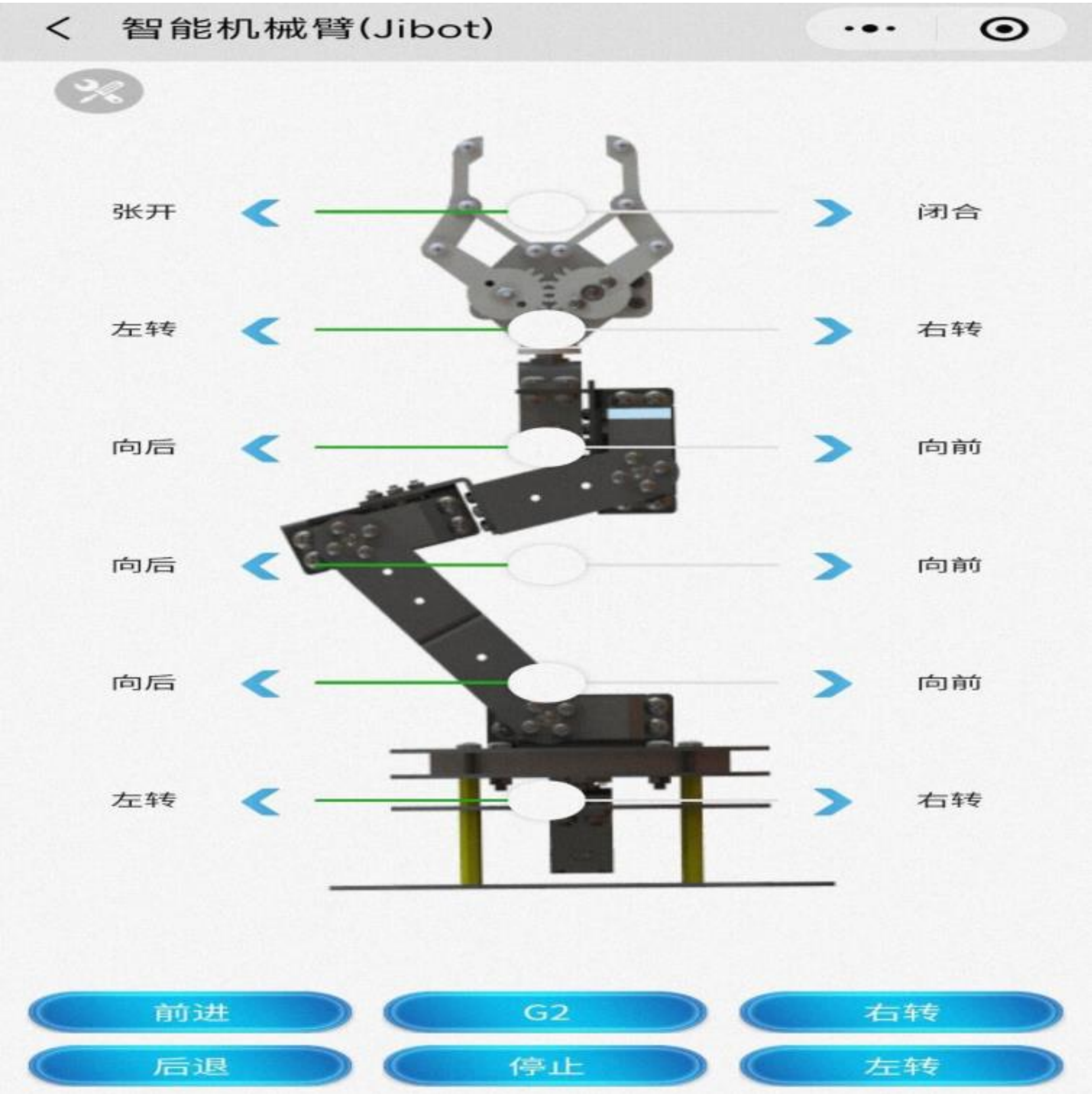


图 6 控制逻辑图

5.2 控制界面



当蓝牙模块接入 51 控制板后，可利用手机 APP 对其进行控制。

5.3 控制程序

```
#ifndef _ANGLE_CONTROL_H_
#define _ANGLE_CONTROL_H_
```

```

#include "derivative.h"
#define Pi 3.14
#define ZERO_ERROR 30
#define ZERO_ERROR_Z 1
#define dt_2 0.002
#define SPEED_TIME 0.01
#define SP_TIME 10
//结构体定义
typedef struct PID1
{ volatile float P; volatile float
    I; volatile float D;
    volatile float D_gro;
    volatile float P_out;
    volatile float D_out;
    volatile float OUT;
    volatile float OUT_old;
}PID;
/*****外部变量声明*****/
extern volatile float g_now_speed; extern
volatile int16 g_right_ftm_cumulative; extern
volatile int16 g_left_ftm_cumulative; extern
u8 g_SpeedControl_priod; extern u8
dir_Control_priod; extern u8 go_car; extern
PID Angle; extern PID speed; extern u8
g_really_speed_expect; extern u8
g_speed_expect; extern short g_Accel[3];
extern short g_Gyro[3]; extern volatile float
g_gyro_angle; extern volatile float
g_acc_angle; extern float car_angle;
extern volatile float g_gyro_z; extern
volatile float g_gyro_y; extern volatile
float g_real_speed_use; extern float
g_speed_expect_k; extern volatile
float g_speed_error; extern volatile
float g_distant; extern volatile int
g_long_way; extern float g_angle_P;
extern float g_angle_D;

```

```

void ACC_angle_calculate(void); void
Gyro_angle_calculate(void); void
Angle_control(void); void
Get_speed(void); void
Cal_SpeedError(void); void
SpeedControl_Out(void); void
speed_control(void); void
get_error(void); void dir_judge(void);
void dir_control(void); void
dir_Out(void); void Output(PID
*Angle); void
Angular_Velocity_Control(void);
#endif
#ifndef _AUXILIARY_FUNCTION_H_
#define _AUXILIARY_FUNCTION_H_
#include "derivative.h" extern
int g_disp_power; extern float
g_fstart_go_angle; void
Start_Go(void); void
Stop_Car(void); int abs_m(int
c); void Uart_send_data1(u16
data); void Uart_send_data(int32
data); void Basic_parameters(u8
ch); void Judge_key(void); void
Judge_key1(void); void
Judge_power(void); void
Ultrasonic_ranging(void);
#endif

#ifndef _CASCADE_CONTROL_
#define _CASCADE_CONTROL_
#include "derivative.h" extern float dis_R; extern
float dis_exp_R; extern float g_change_angle;
extern float g_R_priod; float Get_new_R(void);
void Get_dir_R(void); void Dir_c_control(void);
void Dir_c_control_angle_speed(void); float
Linear_velocity_filter(float Linear_velocity); void
Exp_R_Out(void); #endif
#include "management.h"

```

```

#include <math.h>
#define DIR_TIME 2
/*****方向控制函数*****/

volatile float g_dir_error=0; volatile float
g_dir_error_last=0; volatile float g_turn_speed=0;
u8 g_dir_flag=0; PID dir={0}; volatile float
g_dir_output=0; u8 dir_Control_priod=0; float
g_ec=0; float
g_ring_max=0,g_ring_right=0,g_ring_left=0; u16
g_ring_count=0;
PID Ring={0};
//显示变量
float dis_ec=0;
//最大限幅
float g_error_max=40;
float g_ec_max=20; float
g_dir_out_max=8000; float
g_dir_integral=0;
//调节参数
float g_dir_P=0.0;
float g_dir_D=0.0;
float g_dir_D_gro=0.0;
float ring_error=45;
float ring_ec=0; float
v_p=0; float g_z_ac=0;
/*****all_output*****/
/*****方向环控制程序*****/

void dir_judge(void) {
    //static u8 dir_flag_last=0;
    static u16 leave_value=30;
    if(adc_value[0]>adc_value[3])
    { g_dir_flag=0;//左拐是 0
    } else
    if(adc_value[0]<=adc_value[3])
    { g_dir_flag=1;//右拐是 1
    }
    //丢线判断

```

```

        if(adc_value[0]<leave_value&&adc_value[3]<leave_value&&adc_value[1]<leave_value&&adc
_value[2]<leave_value)
        { go_car=0;
        }

//    dir_flag_last=g_dir_flag;
//    printf("%d\n",g_dir_flag);
} void
get_error(void)
{
    float gain=3000;//开根差比和放大系数 float
    ADC_0=0,ADC_1=0,ADC_2=0,ADC_3=0;
    ADC_0=adc_value[0];
    ADC_1=adc_value[1];
    ADC_2=adc_value[2];
    ADC_3=adc_value[3];
    //adc_value[3]=adc_value[3];
    g_dir_error_last=g_dir_error; g_dir_error=gain*(sqrt(adc_value[0])-
sqrt(adc_value[3]))/(adc_value[0]+adc_value[3]);
    //均值滤波
    g_dir_error=Error_filter(g_dir_error);
    ADC_1 = ADC_1<5?5:ADC_1;
    All_flag.ERROR=500.0f*(ADC_1-ADC_2)/(ADC_0);//测试偏差
} void
Get_turn_speed()
{ volatile float turn_speed=0;
    MPU6050ReadGyro(g_Gyro);
    g_gyro_z=g_Gyro[2]-ZERO_ERROR_Z;
    turn_speed=g_gyro_z/16.40;
    g_turn_speed=-turn_speed;
} void
dir_control(void)
{ volatile static float st_dir_out[4]={0};
    dir.OUT_old=dir.OUT;
    Get_Membership(g_dir_error,g_ec); dir.P=g_dir_P+Fuzzy_Control()*20;
    dir.D=g_dir_D; dir.D_gro=g_dir_D_gro; st_dir_out[3]=st_dir_out[2];
    st_dir_out[2]=st_dir_out[1]; st_dir_out[1]=st_dir_out[0];
    st_dir_out[0]=dir.P*g_dir_error+dir.D*g_ec+dir.D_gro*g_turn_speed;

```

```

    dir.OUT=st_dir_out[0];/*0.4+st_dir_out[1]*0.3+st_dir_out[2]*0.2+st_dir_out[3]*0.1
    ;
    //方向输出限幅
    dir.OUT=dir.OUT>g_dir_out_max?g_dir_out_max:dir.OUT; dir.OUT=dir.OUT<-
    g_dir_out_max?-g_dir_out_max:dir.OUT;
} void
dir_Out(void)
{ g_dir_output = (dir.OUT - dir.OUT_old)*(dir_Control_priod+1)/DIR_TIME + dir.OUT_old;
} void
Final_Out(void)
{ volatile float left_out=0;

    volatile float right_out=0;
    float just_go=1;
    //调试方向环
    if(All_flag.dir_debug)
    { g_dir_output=0;
    }
    if(g_dir_output>=0)
    { left_out =Angle.OUT+(2-just_go)*g_dir_output;
      right_out=Angle.OUT-just_go*g_dir_output;
    } else
    if(g_dir_output<0)
    { left_out =Angle.OUT+just_go*g_dir_output;
      right_out=Angle.OUT-(2-just_go)*g_dir_output;
    }
    if(!go_car)
    { left_out=0;
      right_out=0
      ;
    }
    if(left_out>0)
    { //左轮反转

        FTM_PWM_set_duty(FTM_2, CHANNEL5, 0);
        FTM_PWM_set_duty(FTM_2, CHANNEL0, left_out);
    }
    else

```

```

    { //左轮正转
        left_out=-left_out;
        FTM_PWM_set_duty(FTM_2, CHANNEL5, left_out);
        FTM_PWM_set_duty(FTM_2, CHANNEL0, 0);
    }
    if(right_out>0)
    { //右轮反转
        FTM_PWM_set_duty(FTM_2, CHANNEL3, right_out);
        FTM_PWM_set_duty(FTM_2, CHANNEL2, 0);
    }
    else
    { //右轮正转
        right_out=-right_out;
        FTM_PWM_set_duty(FTM_2, CHANNEL3, 0);
        FTM_PWM_set_duty(FTM_2, CHANNEL2, right_out);
    }
}

* @brief 获取速度增益
* @param    none
* @return    none
* @time

float Get_Gain()
{ float speed_gain=0.0f;
    speed_gain=g_real_speed_use/g_really_speed_expect;
    return speed_gain;
}

```