舵机实训手册

睿尔曼智能科技（北京）有限公司

2024 年 4 月

目录

前言

本文介绍了复合机器人舵机模块的上位机使用与 ROS 发布话题控制舵机运动，舵机模块是复合机器人重要的组成部分。上位机软件的使用与 ROS（Robot Operating System）的发布话题控制，为舵机模块提供了强大的控制能力。上位机软件是控制复合机器人舵机模块的重要工具。通过上位机，用户可以直观地设置舵机的运动参数，如角度、速度等。用户只需在上位机界面上输入相应的参数，即可实现对舵机的精确控制。同时，上位机软件还提供了实时监控功能，用户可以随时查看舵机的运动状态，确保机器人的正常运行。

ROS 也为复合机器人舵机模块的控制提供了强大的支持。ROS 作为一种开源的机器人操作系统，为机器人开发者提供了丰富的工具和库。通过 ROS 的发布话题机制，开发者可以将舵机的控制指令发布到 ROS 网络中，机器人接收到控制指令后，会驱动舵机进行相应的运动，从而实现对舵机的控制。这种方式不仅提高了控制的灵活性，还使得多个舵机模块能够协同工作，实现更复杂的机器人动作。

一、产品介绍

1.1 产品参数

1.1.1 主控板

舵机主控板采用幻尔的串行总线舵机控制板，主控板有以下优点：

1、CPU 采用高性能的 STM32 单片机，高精度控制舵机运动，速度可调。

2、支持手动编程，可无需上位机设置即可快速给舵机设置角度，实现快速编程。

3、接线简单，控制板只有正负极两路电源接口。

4、低压报警功能，电压低于 5.6V 后，蜂鸣器会发出“嘀嘀”的声音。

5、开关内置，使用方便。

6、支持在线调试，无需安装驱动。

7、支持串口通信，控制器可跟别的单片机进行连接，通过别的单片机可向控制板发送指令。

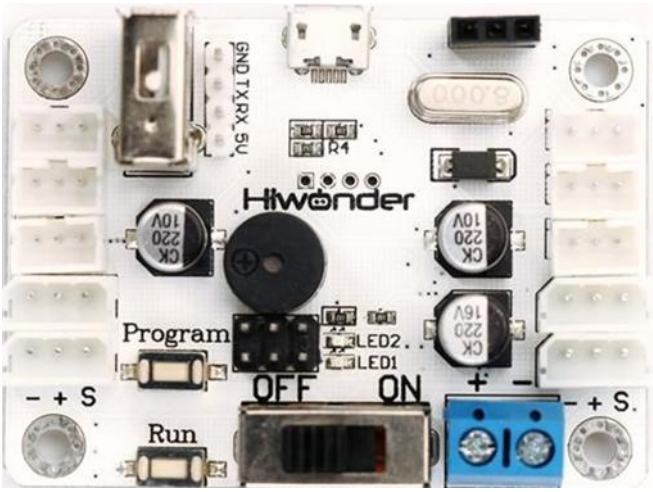

图 1 串行总线舵机主控板

主控板配置参数如下表：

名称	串行总线舵机控制器	
尺寸参数	外形面积尺寸(长*宽)	58mm*42mm
	安装孔距尺寸(长*宽)	49mm*34mm
	重量	21g
供电电压	锂电池	6V-12V

	适配器	
控制方式	上位机	支持
	手柄	支持
	蓝牙	支持
	脱机	支持
编程方式	上位机	支持
	手动按键	支持
通讯方式	串行通讯	参照协议
	波特率	9600
其他参数	内存大小	16MB
	低压报警(5.6 以下)	支持
	开关内置	支持
	外接单片机	支持
	外接 MP3/蓝牙模块	支持
	电脑免驱	支持
	舵机角度回读	支持

表 1 主控板配置参数

1.1.2 总线舵机

本模块采用的是串行总线舵机，实物如下图所示。相比于 PWM 舵机，总线舵机具有更高的控制精度和更快的响应速度。总线舵机采用串口的形式发送指令，

使舵机按照既定的速度目标位置执行工作。总线舵机可以方便地串联在一起，通过一个控制板就可以实现对多个舵机的统一控制。就是说一个舵机串一个舵机，最后接到控制板。每个舵机有分配 ID，类似身份标识，只有接收到对应的 ID 号和指令，才会做出执行。如发送的是 ID：1 的指令，标识为 ID:1 的舵机才会响应并做出执行指令。这种设计不仅简化了线路连接，还提高了系统的可扩展性。

总线舵机是具备闭环反馈功能的，其内置的控制板聚集了电压、速度、温度、位置、电流、负载的传感器。这些数据可以实时反馈给控制板做监测，当扭力超过设定的百分比或者输入的电压超过了多少 V 或者温度超过多少度或者电流超过多少 A 时，舵机将采取设定的方式停止运行或者卸力等待。所有的参数均可在上位机做出相应的设定。除此之外，舵机还可以对运行的角度，波特率，工作模式（如电机模式，连续旋转的）进行设置。这些设置选项使得总线舵机能够适应各种不同的应用场景和需求。

图 2 总线舵机

1.2 总线舵机通信协议

舵机通信协议参考附件

1.3 复合产品介绍

在单臂复合机器人、双臂复合机器人、双臂升降机器人头部安装了舵机，通过 USB 接口与主控模块实现通讯。

图 3 单臂复合机器人

图 4 双臂复合机器人

图 5 双臂升降机器人

二、总线舵机

2.1 上位机介绍与使用

通用总线舵机上位软件 Bus Servo Control，软件界面简洁明了、操作简单、设置便捷等诸多优点，是一款优秀的工具软件。

图 6 软件图标

2.1.1 上位机功能介绍

Bus Servo Control 上位软件一共拥有四个操作界面，分别是通用模式界面、六足模式界面、人形模式界面、舵机调试工具界面。通用模式界面主要是对通用舵机动作进行编辑和下载；六足模式界面是六足机器人动作进行编辑和下载；人形模式是应用于人形机器人动作编辑和下载；舵机调试界面主要是单个舵机参数设置与动作组进行下载。上位机主要是用来设置舵机的 ID 和舵机转动的角度范围。一般机器人出厂时，都设置好了舵机的 ID 和舵机转动角度范围，非必要时，用户不要轻易更改。

通用模式界面布局：

图 7 通用模式界面

舵机调试界面布局：

图 8 舵机调试工具界面

在舵机参数设置区，可以设置舵机 ID 号、角度范围、供电电压范围、温度范围、控制板 LED 的开关及 LED 报警。在右侧能够实时显示当前电压、当前温度、当前位置。当然，也可以导入预设定的舵机动作等。

2.2 硬件连接

(1)将电源对接线插入控制器的电源接口（红线插入+，黑线插入—），使用螺丝刀将接口扭紧(供电电压 6V-12V)。

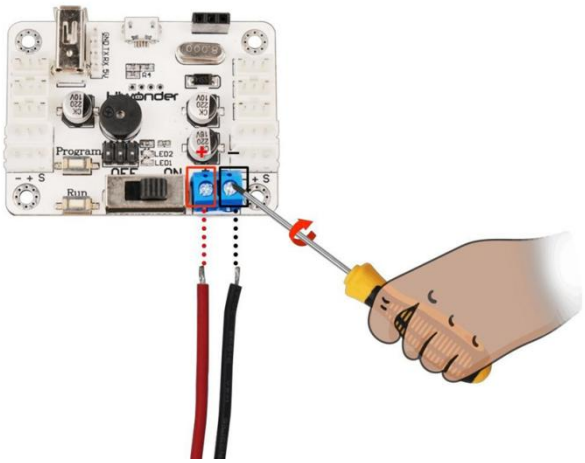

图 9 控制板与电源接线图

(2)将锂电池(或者可调直流电压源)黑色端口直接插入电池对接线。

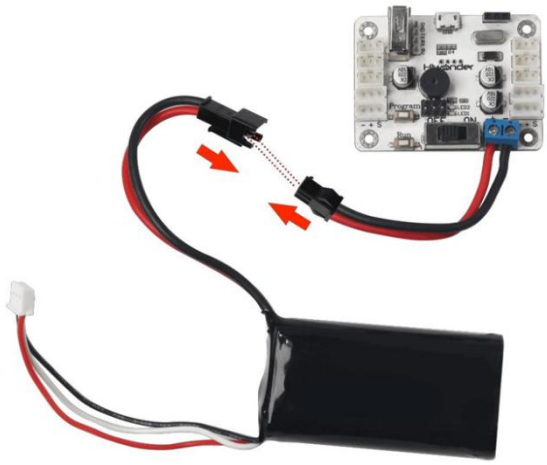

图 10 控制板与电源接线图

(3)如需断开锂电池，可轻轻按下对接线的凸槽，将锂电池取出。（切勿直接硬拔）

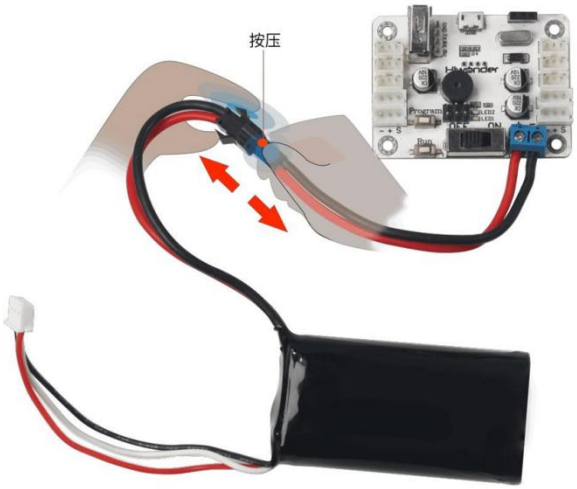

图 11 电源接线示意图

(4)给舵机设置 ID 编号时，我们需要将要设置的舵机单独连接至串行总线控制板进行设置（设置 ID 时，控制板只能连接 1 个舵机），将充满电的锂电池通过连接线连接至舵机控制板。再将控制板通过 USB 线连接至电脑上，开关由“OFF”推动到“ON”，打开控制板即可。

连接图如下：

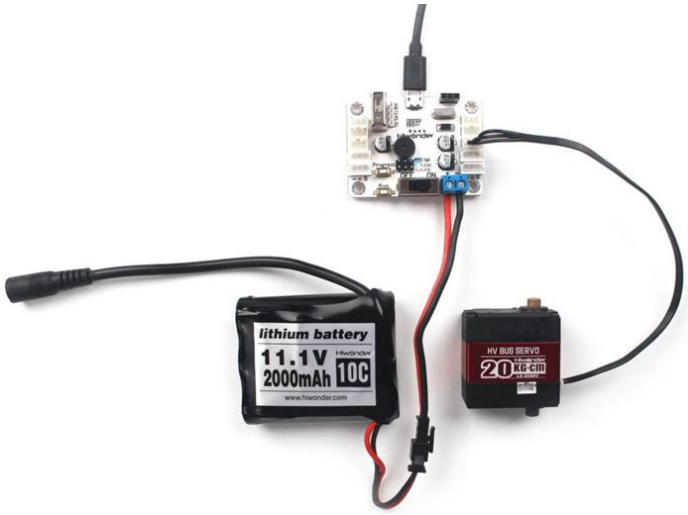

图 12 舵机整体接线

若舵机连接成功，上位机页面如下图：

图 13 上位机页面

2.3 舵机 ID 更改

一般复合产品舵机模块都是使用 2 个舵机进行配合的。因此我们需要先给逐一给舵机设置 ID 编号，才能保证动作组能顺利运行。

1)连接好舵机之后，打开上位机软件，在菜单模式选则处选择舵机调试工具选项卡进入到舵机调试界面。进入界面是会弹出一个提示窗口，选择“确定”即可。

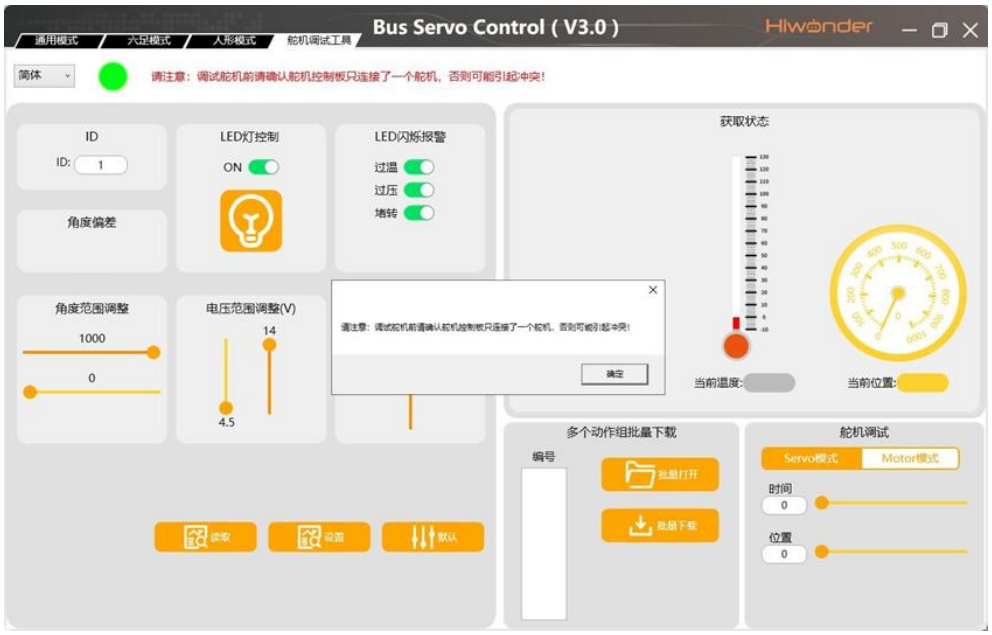

图 14 舵机设置页面

2)我们在舵机参数设置区，将我们需要设置的 ID 号填入 ID 框，点击下方的设置按钮即可。我们分别设置两个舵机的 ID 为 1 和 2。更改完舵机 ID 点击“设置”，即可完成设置舵机 ID。

3)如果我们想查看当前舵机的 ID 号，那么可以通过按钮进行读取。读取成功后会在 ID 框显示对应的舵机 ID，并在状态区和参数设置区将舵机的其它参数反馈在软件界面中。

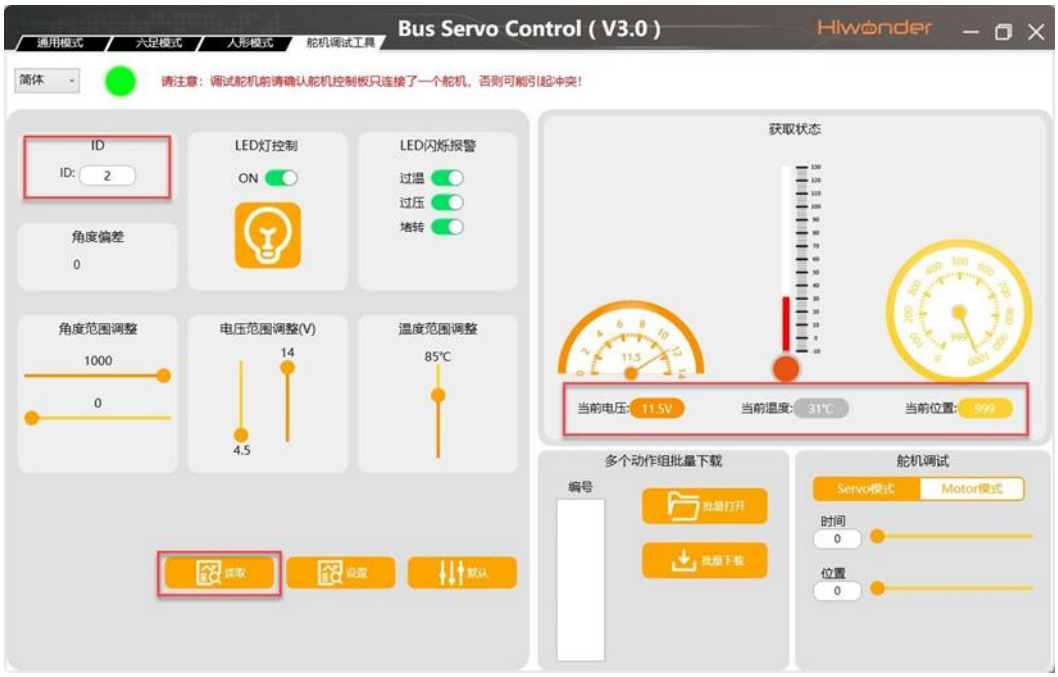

图 15 读取舵机信息

2.4 舵机转动角度设置

舵机上位机软件可设置舵机的运动范围为 0 到 1000，舵机角度范围是 240°，240/1000 = 0.24，例如，舵机转动到 120°，对应换算：120/0.24 = 500

所以 ID 为 1 舵机角度范围为 96° 至 144°，将 ID 为 1 舵机的角度范围设置为 400 至 600，点击“设置”即可。

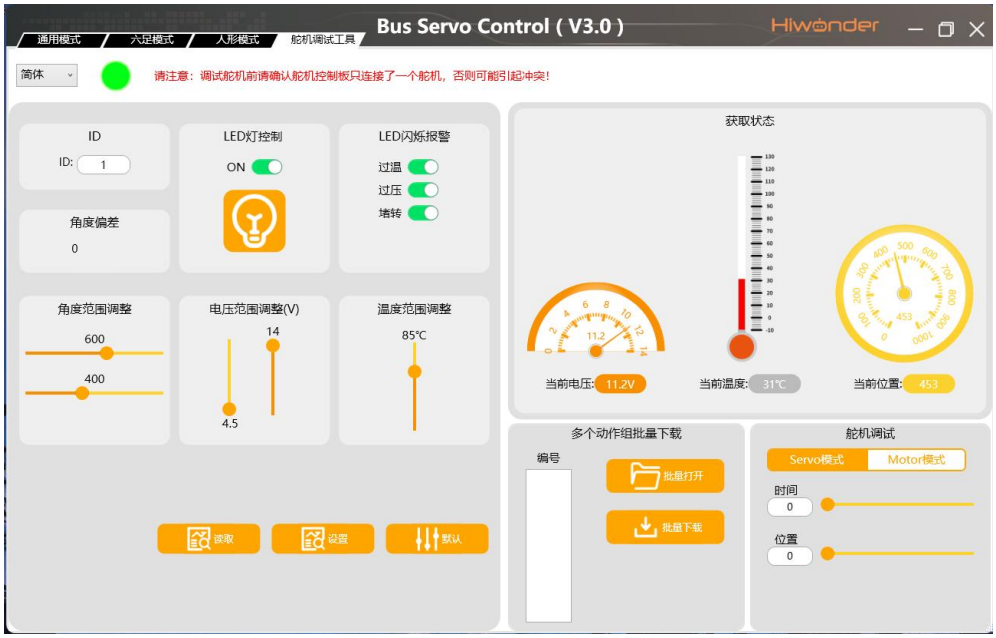

图 16 舵机 1 角度范围设置

ID 为 2 舵机角度范围为 48° 至 192°，将 ID 为 2 舵机的角度范围设置为 200 至 800，点击“设置”即可。

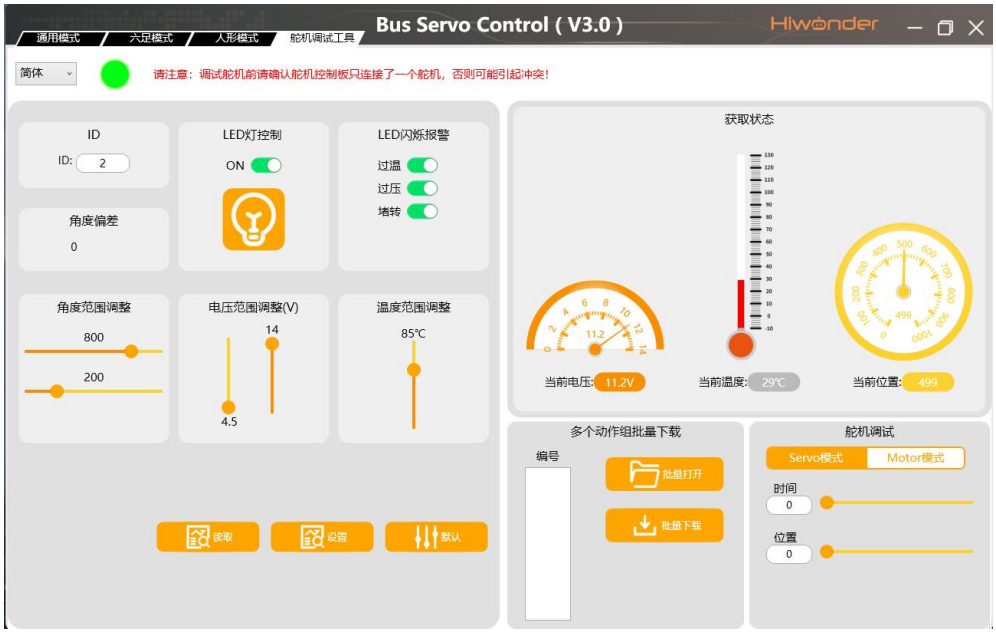

图 17 舵机 2 角度范围设置

在复合机器人头部舵机安装如图所示，上方舵机 ID 为 1，下方舵机 ID 为 2

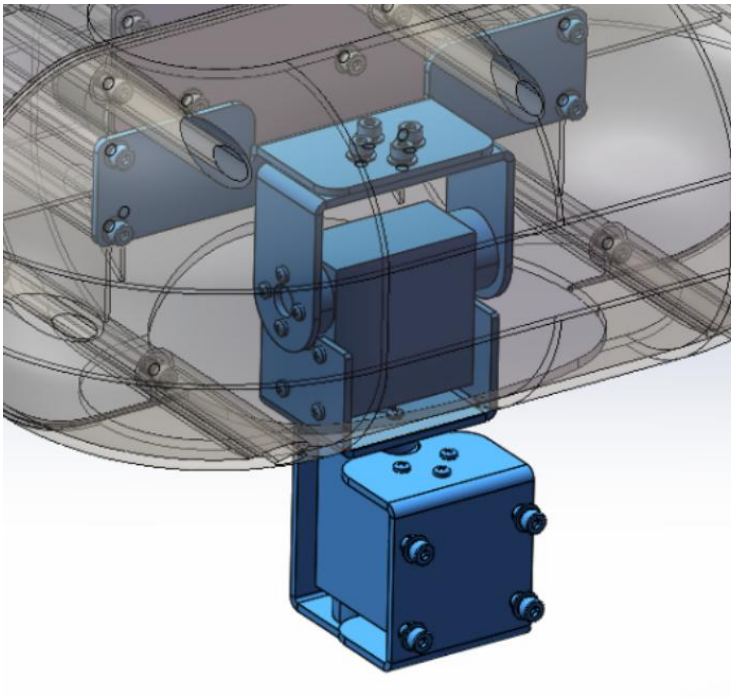

图 18 复合机器人舵机安装示意图

2.5 上位机控制舵机

设置好舵机 ID 和角度范围后，将舵机 1 和 2 串联起来，并进入通用界面。由于设置好了舵机 1 的角度范围在 400° - 600°，舵机 2 的角度范围在 200° - 800°，拖动舵机 1 和 2 的滑块，舵机 1 和 2 在预定的角度范围转动。

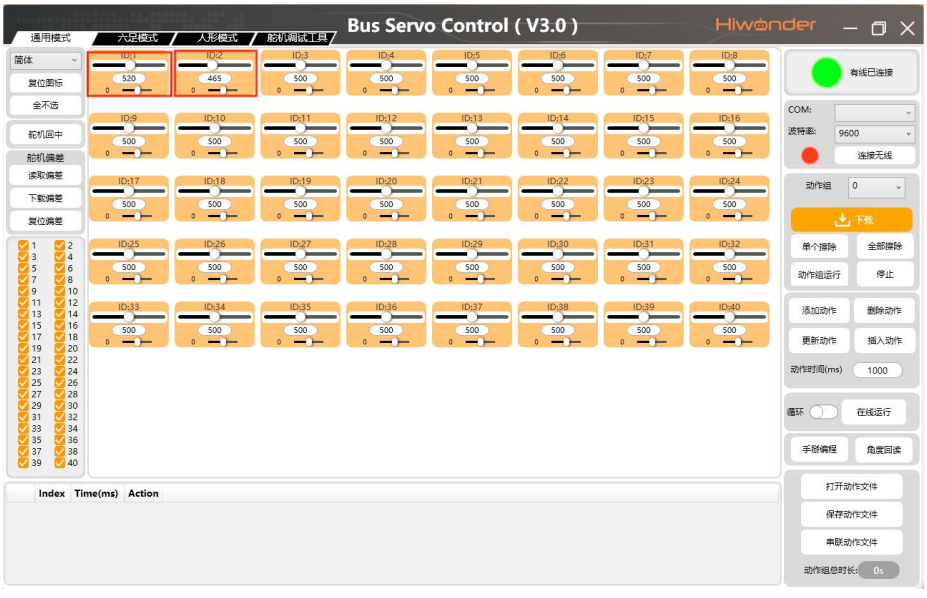

图 19 舵机通用模式

三、ROS 控制舵机

3.1 通过 ROS 的 Message 控制舵机运动

通过 ROS 控制舵机代码如下：

```
#include <ros/ros.h>
#include <serial/serial.h>
#include <std_msgs/String.h>
#include <std_msgs/Empty.h>
#include <dual_arm_msgs/Servo_Move.h>
#include <dual_arm_msgs/Servo_GetAngle.h>

serial::Serial ros_ser;
#define   sBUFFERSIZE   10//send buffer size  串口发送缓存长度
unsigned char s_buffer[sBUFFERSIZE];//发送缓存
//舵机转动控制回调函数
void callback_servoMove(const dual_arm_msgs::Servo_Move& msg){
      //   cmd_to_serial(msg);
      //   cmd_servo_move(msg.servo_id,msg.angle);
      memset(s_buffer,0,sizeof(s_buffer));
       s_buffer[0]=0x55;
}
```

```
//读取舵机角度位置值回调函数
void callback_servoGetAngle(const dual_arm_msgs::Servo_GetAngle& msg){
      //    cmd_to_serial(msg);
      //    cmd_servo_move(msg.servo_id,msg.angle);
      memset(s_buffer,0,sizeof(s_buffer));
       s_buffer[0]=0x55;
       s_buffer[1]=0x55;
       s_buffer[2]=0x04;
       s_buffer[3]=0x15;
       s_buffer[4]=0x01;
       s_buffer[5]=msg.servo_id;    //舵机 ID
      //    data_to_serial(s_buffer, 10);
      ros_ser.write(s_buffer, 6);
      ROS_INFO_STREAM("Get Servo Angle");
 }
int main (int argc, char** argv){
       ros::init(argc, argv, "my_serial_node");
       ros::NodeHandle n;
       //订阅控制舵机转动主题
       ros::Subscriber    sub_servo_move    =    n.subscribe("/servo_control/move",    1000,
callback_servoMove);
       //订阅获取舵机角度位置的主题
       ros::Subscriber sub_servo_getAngle = n.subscribe("/servo_control/get_angle", 1000,
callback_servoGetAngle);
       //发布主题:舵机状态
        ros::Publisher    pub_servo_state    =    n.advertise<std_msgs::String>("/servo_state",
1000);
       try
       {
            ros_ser.setPort("/dev/duoji");
            ros_ser.setBaudrate(9600);
            serial::Timeout to = serial::Timeout::simpleTimeout(1000);
            ros_ser.setTimeout(to);
            ros_ser.open();
       }
       catch (serial::IOException& e)
       {
            ROS_ERROR_STREAM("Unable to open port ");
            return -1;
       }
       if(ros_ser.isOpen()){
            ROS_INFO_STREAM("Serial Port opened");
       }else{
            return -1;
```

```
    }
    ros::Rate loop_rate(10);
    while(ros::ok()){
        ros::spinOnce();

      //获取缓冲区内的字节数
      size_t n = ros_ser.available();
      if(n!=0)
      {
          uint8_t buffer[1024];
          //读出数据
          n = ros_ser.read(buffer, n);
          ROS_INFO_STREAM("Reading from serial port");
          for(int i=0; i<n; i++)
          {
              //16 进制的方式打印到屏幕
              std::cout << std::hex << (buffer[i] & 0xff) << " ";
          }
          std::cout << std::endl;
      }
      //   if(ros_ser.available()){
      //       ROS_INFO_STREAM("Reading from serial port");
      //       std_msgs::String serial_data;
      //       //获取串口数据
      //       serial_data.data = ros_ser.read(ros_ser.available());
      //       ROS_INFO_STREAM("Read: " << serial_data.data);
      //       //将串口数据发布到主题 sensor
      //      //    pub_servo_state.publish(serial_data);
      //   }
        loop_rate.sleep();
    }
    ros_ser.close();
}
```

编写好代码以后，首先打开一个终端，开启 roscore，再开启一个终端，运行 rosrun servo_control servo_controller

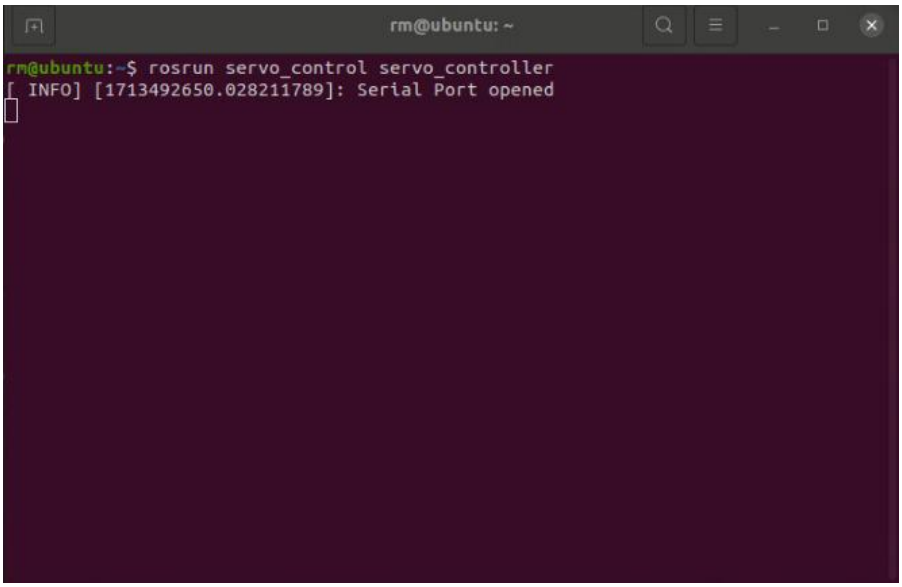

图 20 rosrun 指令

此时，我们可以通过 rostopic list 查询到话题，可以查看到 get_angle 和 move 话题。

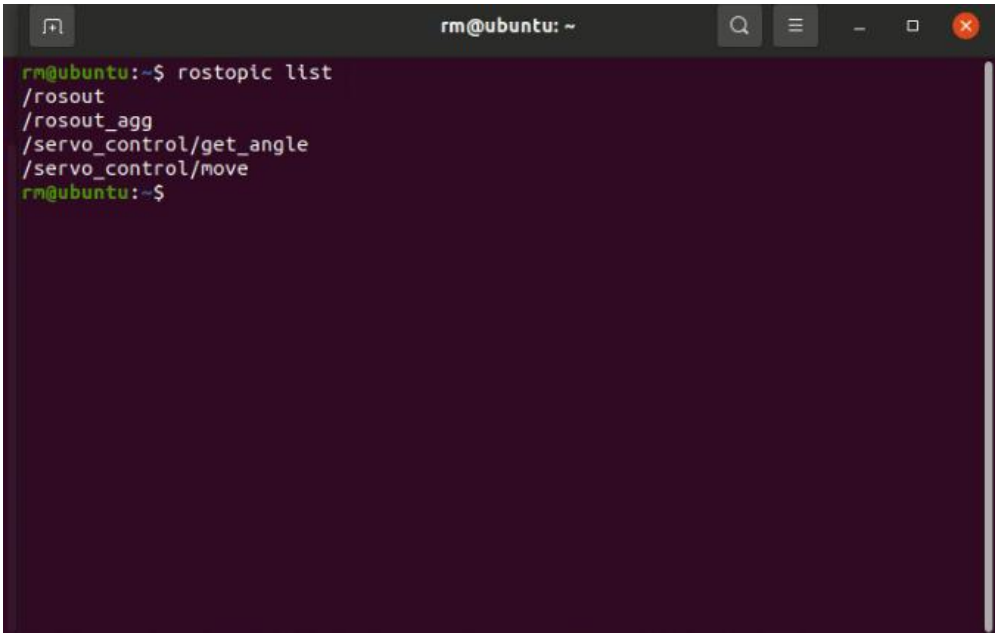

图 21 rostopic list 指令

然后，我们可以发布 get_angle 话题获取舵机 1 的当前角度。开启一个终端，运行 rostopic pub -1 /servo_control/get_angle dual_arm_msgs/Servo_GetAngle “servo_id: 1”

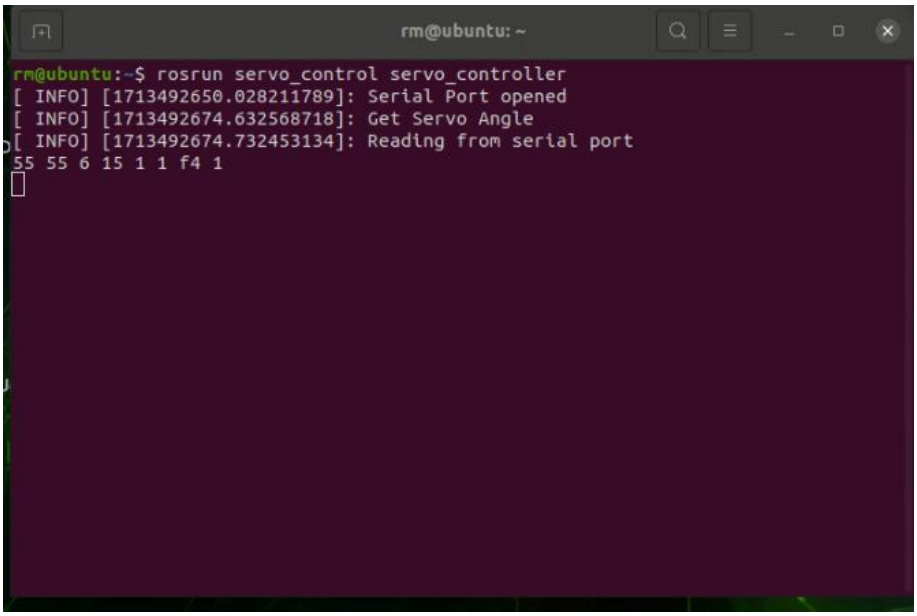

图 22 查询舵机 1 当前角度

输入指令 rostopic pub -1 /servo_control/get_angle dual_arm_msgs/Servo_GetAngle “servo_id: 2”，可查询到舵机 2 的当前角度。

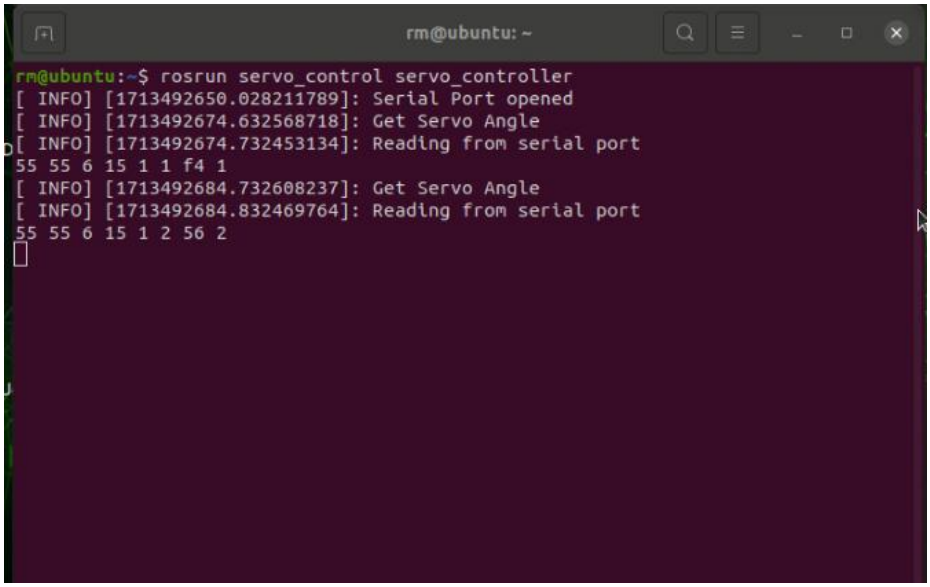

图 23 查询舵机 2 当前角度

输入指令 rostopic pub -1 /servo_control/move dual_arm_msgs/Servo_Move “servo_id: 1”，angle 输入 500，舵机 1 便可转动到 500。舵机 2 同理。

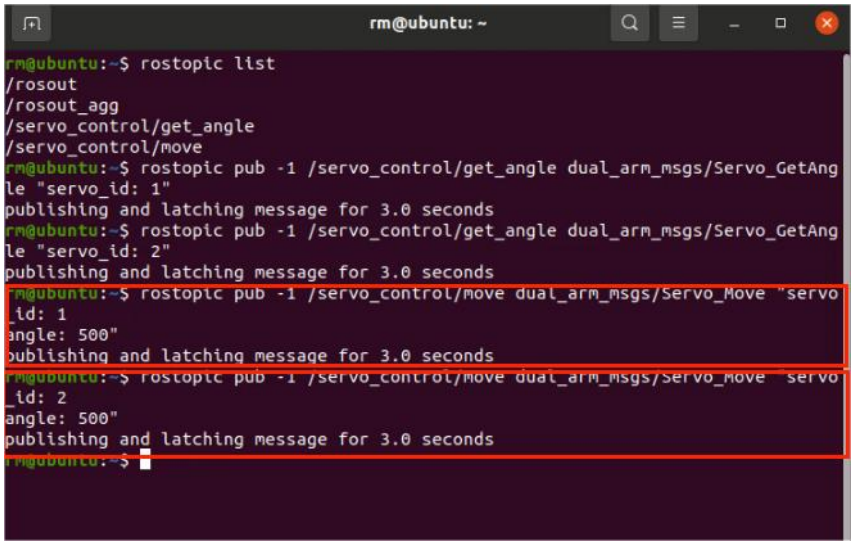

图 24 发布舵机运动话题

运行结果如下：

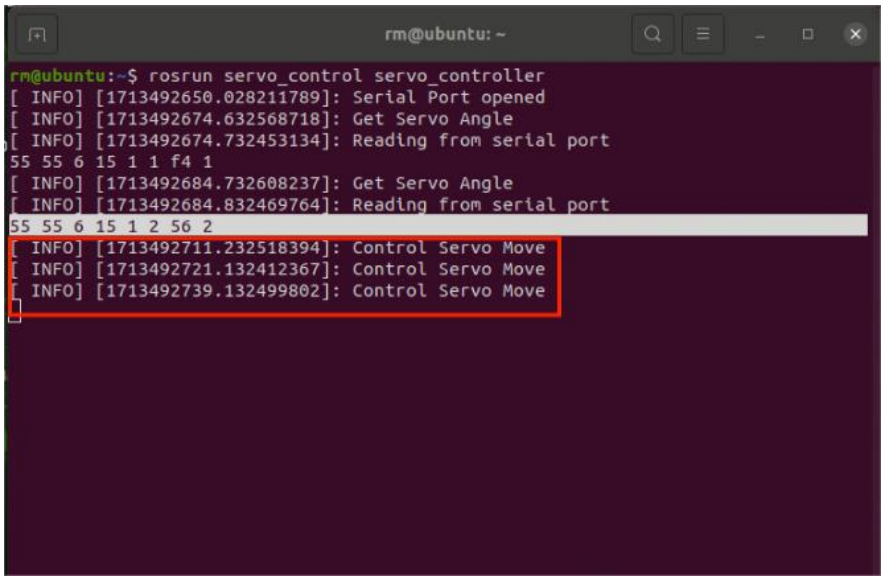

图 25 运行结果