全爱人形机器人使用说明文档

驱动板部分

版本号	V1.0
创建时间	2020-7-20

第一部分：机器人系统的组成

机器人由很多部件组成，全爱机器人系统的组成可分为四大部分：视觉检测装置、控制板、舵机驱动板和执行机构。系统框图和各部分之间的关系如图 1.1 所示。

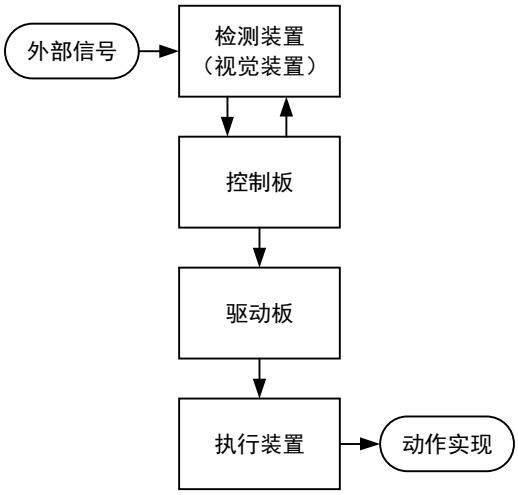


图 1 . 1　机器人系统框图

除上图中的系统组成部分外，一个完整的机器人还需要一套成型的结构件来支撑。本章将对机器人的各组成部分进行说明。

1.1 结构件

结构件是组成机器人实体的各种构件，组成全爱机器人的结构件包括足部结构件、U 型结构件、膝盖结构件、前后胸结构件、手部结构件、头部结构件及其他亚克力盖板。

1.2 视觉检测装置

视觉检测装置相当于机器人的“眼睛”，其作用是检测机器人的工作状态、运动情况及周围环境，并根据不同的需求将采集到的图像反馈给控制系统，从而让执行机构完成相应的动作或进行相应的调整。

全爱机器人采用 4K 摄像头作为机器人的视觉检测装置，可以提供更高的分辨率。

1.3 控制系统

控制系统即全爱机器人胸前的控制板，其作用是接收视觉检测装置的信号并进行相应的处理，再根据不同的处理结果给驱动装置或其他外接装置下发命令，从而完成对机器人的控制工作。

控制系统有两种方式：一种是集中式控制，即机器人的全部控制工作均由一台微型计算

机来完成；另一种是分级式控制，即采用多台微型计算机来完成机器人的控制工作。

1.4 驱动装置

驱动装置是驱使执行装置（舵机）运动的装置。驱动装置接收到控制系统发出的指令信号后，借助动力元器件驱使机器人进行运动。

1.5 执行装置

执行机构是机器人的本体。其转动副常被称为“关节”，机器人“关节”的个数就是机器人的自由度数。如全爱机器人共有 17 个舵机可以运动，也就是拥有 17 个“关节”，称为“17 自由度机器人”。

第二部分：机器人的组装

在第一部分中介绍了机器人的系统组成，本章将对机器人的结构零部件及安装做详细介绍。一台完整的机器人如图 2.1 所示。

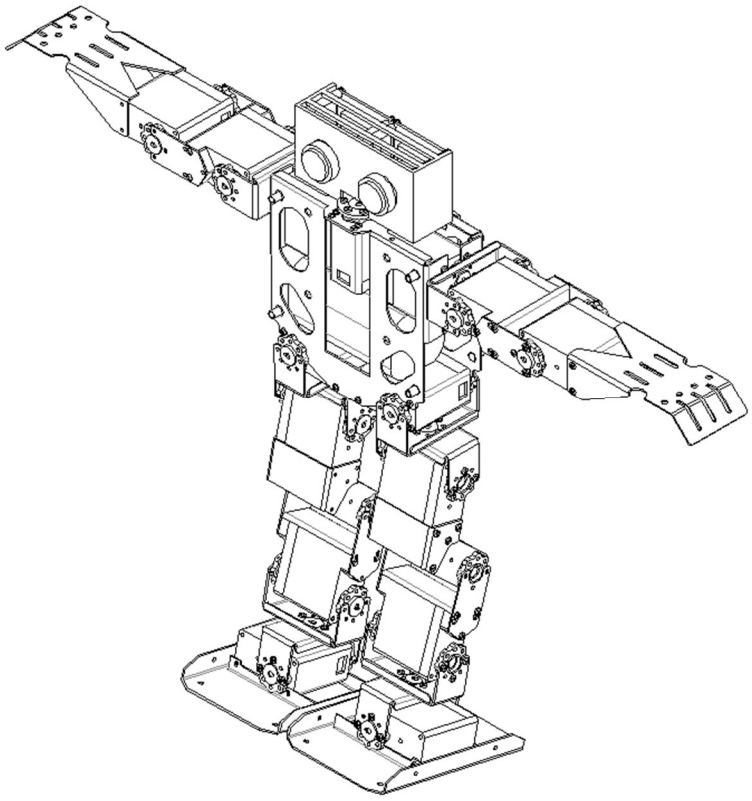
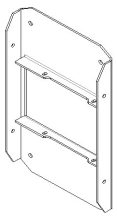
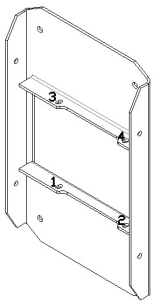


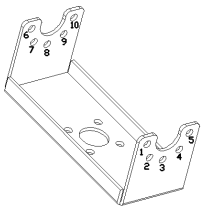
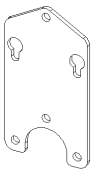
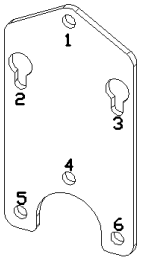
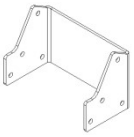
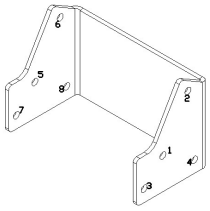
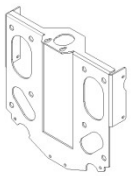
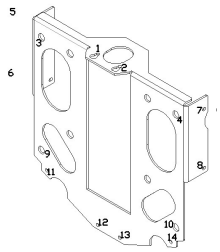
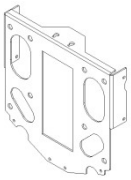
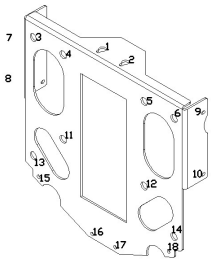
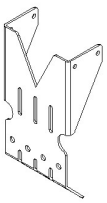
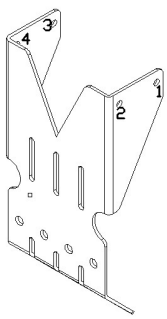
图 2.1　　机器人整机图示

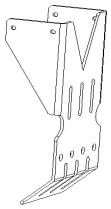
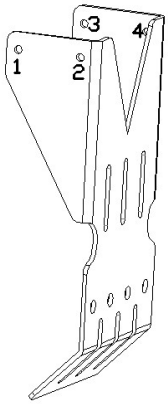
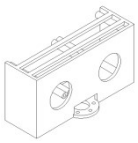
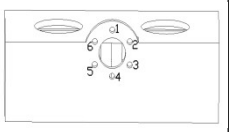
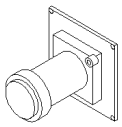
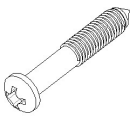
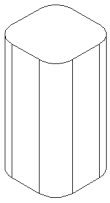
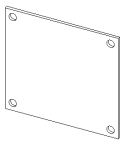
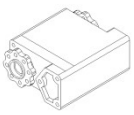
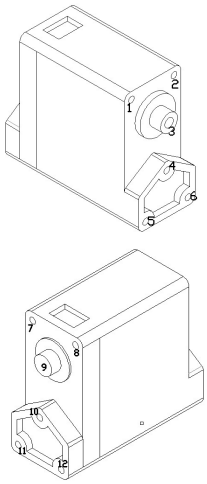
2.1 机器人的结构零部件清单

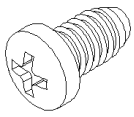
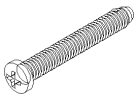
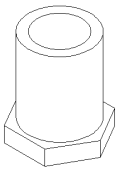
机器人结构零部件清单如表 1-1 所示。

表 1-1　机器人结构零部件

序号	图例	孔定义	名称	单机用量	尺寸 (mm)	材质	备注/用途
1			足部结构件	2	厚度 1mm	硬铝合金 6061T6	

2			U 形 支架	10	厚度 1mm	硬铝合金 6061T6	用于关节处连接
3			关节 连片	8	厚度 1mm	硬铝合金 6061T6	小腿及手臂处连接
4			膝部 结构件	2	厚度 1mm	硬铝合金 6061T6	大腿处连接
5			前胸 结构件	1	厚度 1mm	硬铝合金 6061T6	电池组将置于其中
6			后胸 结构件	1	厚度 1mm	硬铝合金 6061T6	后胸与主板相连
7			左手 结构件	1	厚度 1mm	硬铝合金 6061T6	

8			右手 结构件	1	厚度 1mm	硬铝合金 6061T6	
9			头部 结构件	1	厚度 1mm	硬铝合金 6061T6	
10			摄像头	2	38*38*3 3		
11			螺丝	8	M2.3*16	45	用于固定摄像头
12			电池组	1	66*36*3 6		
13			主板	1	60*55		
14			舵机	17	40*20*3 7	外壳为塑料	动力件

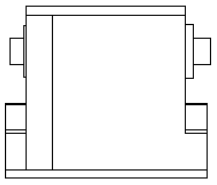
15			主舵盘	21			舵机组件
16			副舵盘	17			舵机组件
17			螺丝	21		45	用于连接固定主舵盘和舵机，固定控制器
18			螺丝	17		45	用于固定连接副舵盘和支架
19			螺丝	121	M2*4	45	用于连接固定副舵盘和支架
20			螺丝	52	M2*16	45	用于连接固定舵机和支架
21			螺丝	16	M2*20	45	用于连接固定舵机和支架
			压铆螺母柱	12	M3*6		用于固定主板
22			舵机连接线	11			腿部从下往上第四个及以上舵机和主板连接线

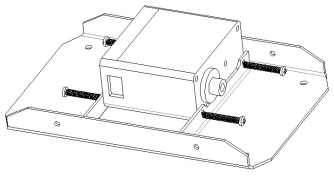
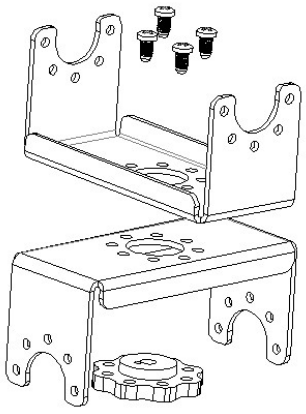
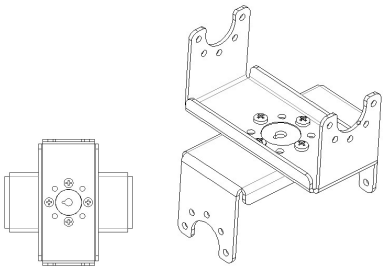
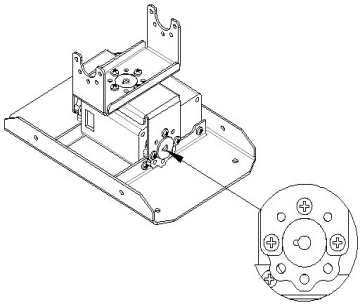
23			舵机 连接线	2			腿部从下往上第三个舵机和主板连接线
24			舵机 连接线	2			腿部从下往上第二个舵机和主板连接线
25			舵机 连接线	2			腿部从下往上第一个舵机和主板连接线

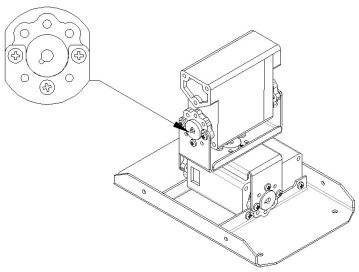
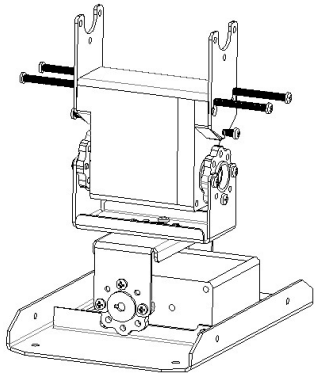
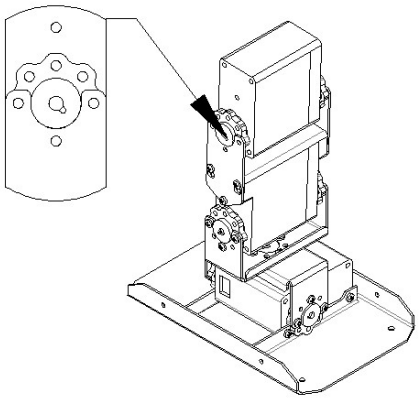
部件组装

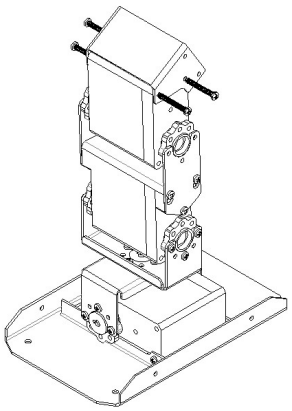
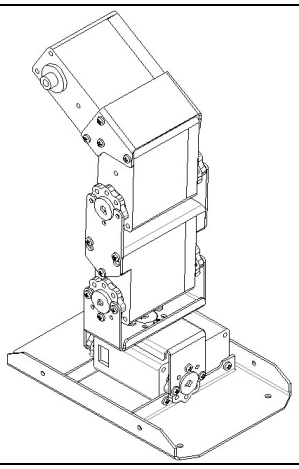
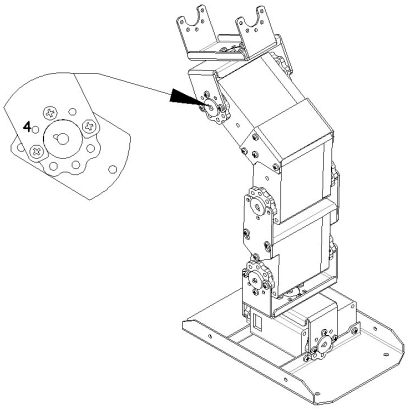
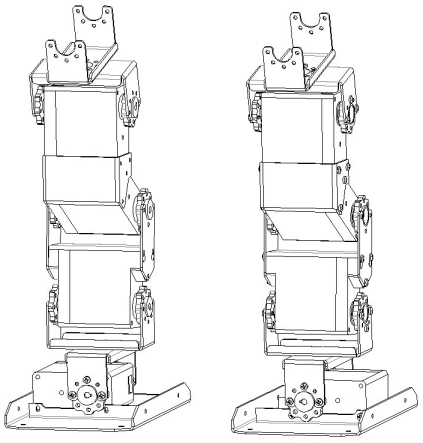
本装配工艺适合单人或多人流水线配合操作执行，其部件装配件的数量是按照单台机器人必备部件而设定。如果是流水线作业，可将装配的安装步骤改为流水线的工序。每个安装工序也可按照内容需要分解为多个子序列过程。本机器人的安装步骤分为四个部分：腿部、身体、手臂、头部。

左腿部关节与舵机的装配

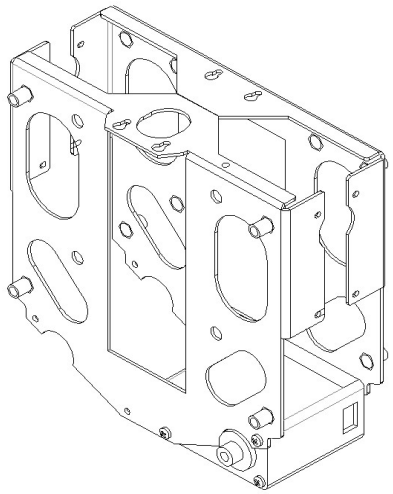
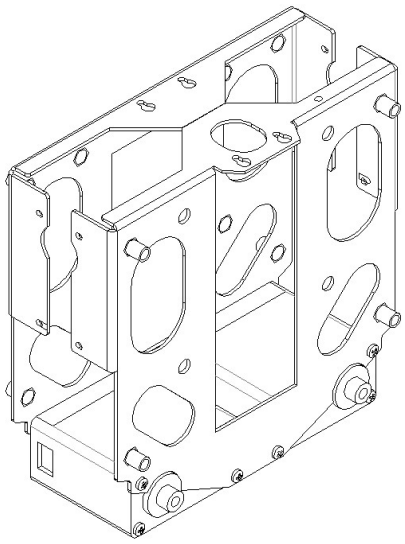
步骤	装配内容	图示
步骤 1	① 本机器人上共有十七个舵机，将所有舵机的主、副舵盘拆下，并保留所拆下的主、副舵盘上的螺丝。	

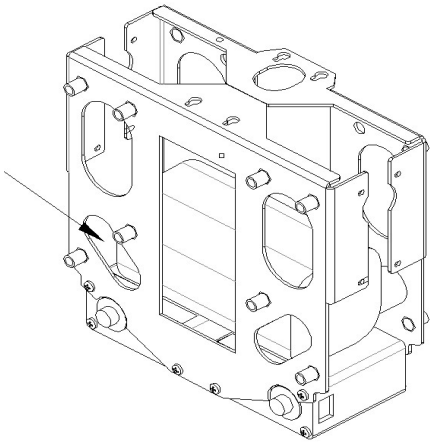
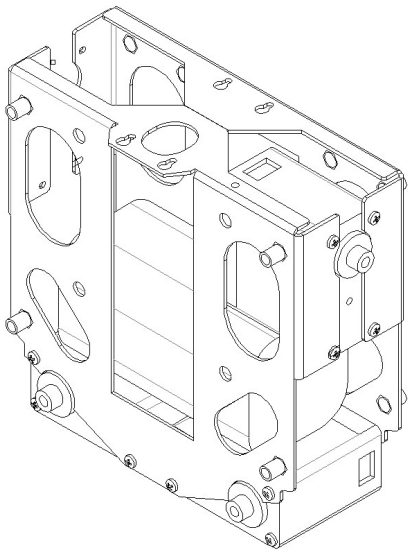
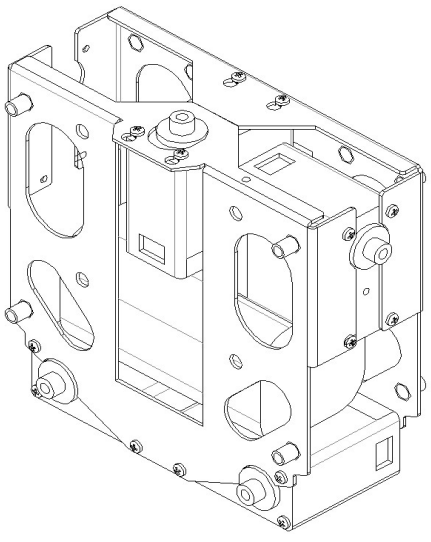
步骤 2	① 首先将舵机 3、5、6、9、11、12 孔的螺丝拧出，并拆除底部的外壳。再将 1 号孔的螺丝拧出，将该螺丝拧入 6 号口;将 8 号孔的螺丝拧出，将该螺丝拧入 11 号口。 ② 将舵机的 1 号孔对准脚掌的 1 号孔；舵机的 5 号孔对准脚掌的 2 号孔；舵机的 8 号孔对准脚掌的 3 号孔；舵机的 11 号孔对准脚掌的 4 号孔。全部对准后拧入四颗 M2*16 的自攻螺丝。	
步骤 3	① 将两个 U 形支架以 90 度垂直的方向相接触。 ② 用一个八角螺旋圆盘插入两 U 形支架中。	
步骤 4	① 拧入四颗 M2*4 的自攻螺丝将其紧固，形成装配体，如图所示。	
步骤 5	① 将步骤 2 中的舵机装上主、副舵盘。 ② 将步骤 2 和步骤 4 的装配体结合。	

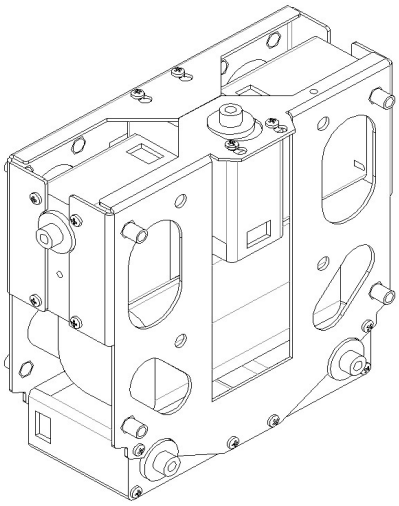
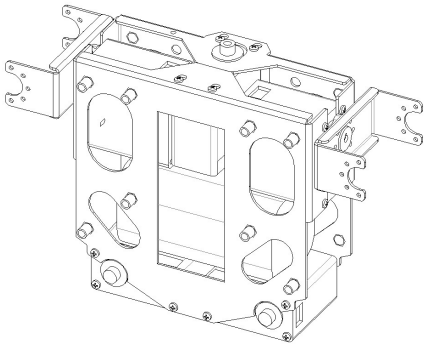
	➢ 注意，主舵盘缺口应对齐 U 形支架的 5 号孔。	
步骤 6	① 将舵机换上特制主、副舵盘。 ② 舵机与步骤 5 的装配体结合。 ③ 用六颗 M2*4 的自攻螺丝固定连接 U 形支架和舵盘，如图所示。 ➢ 注意，主舵盘缺口应对准 U 形支架 2 号孔。	
步骤 7	① 将舵机的 5、6、11、12 孔的螺丝拧下，保留底部外壳。用四颗 M2*20 的螺丝将尖形支架与舵机相连接。 ② 用两颗 M2*4 的螺丝拧入两个尖形支架 1 号孔，与舵机的 4、10 号孔相连。	
步骤 8	① 将舵机换上特制舵盘。 ② 拆去 5、6、11、12 孔螺丝，并拆除底部外壳。 ③ 将舵机与步骤 7 装配体结合。 ➢ 注意，主舵盘缺口应指向尖形支架的 4 与 5 号孔的中间，如图所示。	

步骤 9	① 将固定连接块的 5 号孔对准舵机的 6 号孔，将舵机固定连接块的 6 号孔对准舵机的 5 号孔。 ② 用四颗 M2*16 的自攻螺丝拧入固定块的 1、2、5、6 号孔。	
步骤 10	① 将舵机 5、6、11、12 号孔螺丝拆除，并拆除底部外壳。 ② 将固定连接块的 7 号孔对准舵机的 5 号孔，将固定连接块的 8 号孔对准舵机的 6 号孔。 ③ 用四颗 M2*16 的自攻螺丝将其固定，如图所示。	
步骤 11	① 将舵机换上特制主、副舵盘。 ② 再装一套步骤 4 的装配体。 ③ 用六颗 M2*4 的自攻螺丝将两套装配体连接，如图所示。 ➢ 注意，主舵盘缺口应指向 U 形支架 4 号孔。	
步骤 12	右腿与左腿沿竖直方向对称	

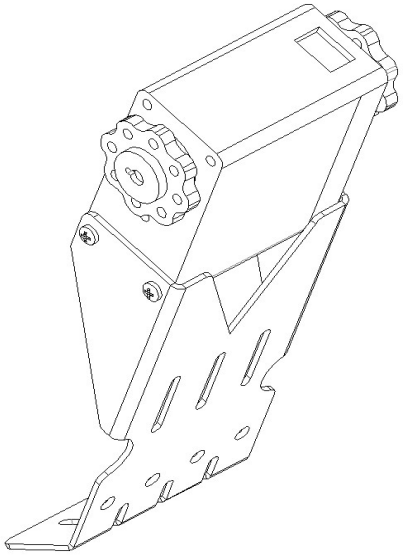
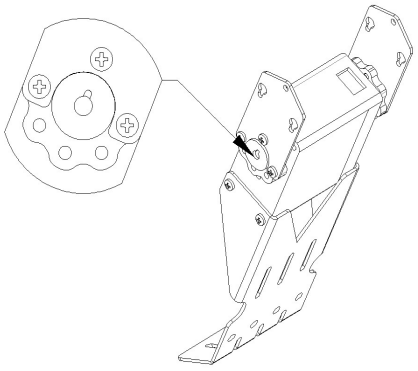
身体的装配

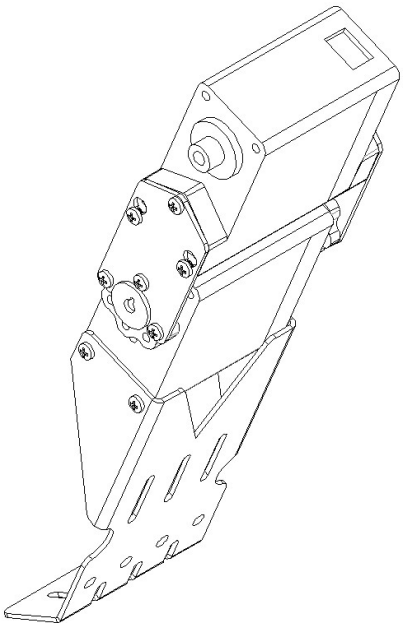
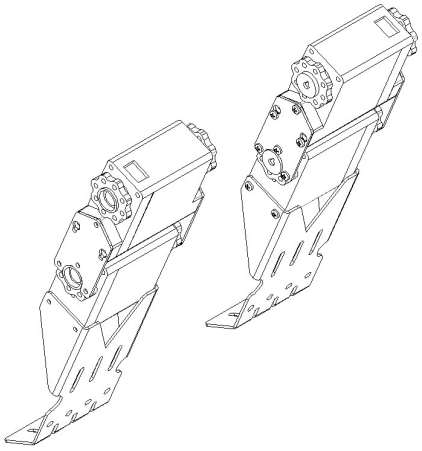
步骤	装配内容	图示
步骤 1	① 将舵机的 5、6、11、12 号孔的螺丝拧出，并拆除底部外壳。将舵机的 1 号孔螺丝拧出，然后把该螺丝拧入 5 号孔；将舵机的 8 号孔螺丝拧出，然后将该螺丝拧入 12 号孔。 ② 将舵机 1 号孔对准前胸 14 号孔，舵机 6 号孔对准前胸 13 号孔；舵机 8 号孔对准后胸 15 号孔，将舵机 11 号孔对准后胸 16 号孔，用四颗 M2*16 的自攻螺丝将其固定。	
步骤 2	① 将舵机的 5、6、11、12 号孔的螺丝拧出，并拆除底部外壳。将舵机的 2 号孔螺丝拧出，然后把该螺丝拧入 6 号孔；将舵机的 7 号孔螺丝拧出，然后将该螺丝拧入舵机 11 号孔。 ② 将舵机 2 号孔对准前胸 11 号孔，舵机 5 号孔对准前胸 12 号孔；舵机 7 号孔对准后胸 18 号孔，将舵机 12 号孔对准后胸 17 号孔，用四颗 M2*16 的自攻螺丝将其固定。	

步骤 3	① 将电池装入，电源线从后胸图示处穿出。	
步骤 4	① 将舵机 5、6、11、12 号孔螺丝拧出，并拆去底部外壳。然后将舵机 1、2 号孔螺丝拧出，再将拧出的螺丝分别拧入舵机 11、12 号孔。 ② 将舵机 1、5、2、6 号孔分别对准前胸 7、8 号孔和后胸 7、8 号孔。 ③ 用四颗 M2*16 自攻螺丝将舵机固定。	
步骤 5	① 将舵机 5、6、11、12 号孔螺丝拧出，并拆去底部外壳。然后将舵机 1、2 号孔螺丝拧出，再将拧出的螺丝分别拧入 11、12 号孔。 ② 将舵机 1、2、5、6 号孔分别对准前胸 2、1 号孔和后胸 1、2 号孔。 ③ 用四颗 M2*16 自攻螺丝将舵机固定。	

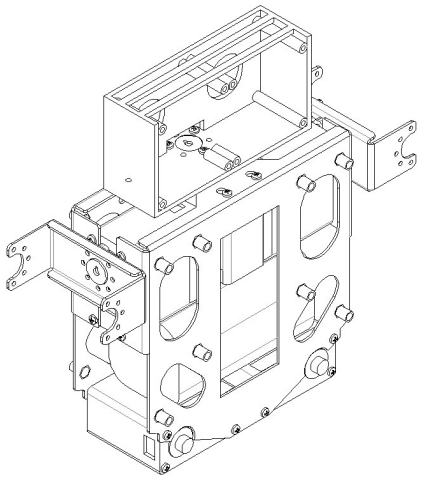
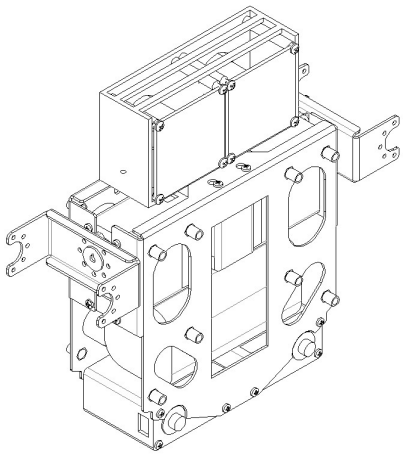
步骤 6	① 将舵机 5、6、11、12 号孔螺丝拧出，并拆去底部外壳。然后将舵机 1、2 号孔螺丝拧出，再将拧出的螺丝分别拧入 11、12 号孔。 ② 将舵机的 1、5、2、6 号孔分别对准后胸 9、10 号孔和前胸 5、6 号孔。 ③ 四颗 M2*16 自攻螺丝将舵机固定。	
步骤 7	① 将左上方和右上方的舵机装上主舵盘。 ② 用八颗 M2*4 的自攻螺丝将两个主舵盘分别与两个 U 形支架连接固定，支架方向和身体竖直方向垂直。如图所示。	

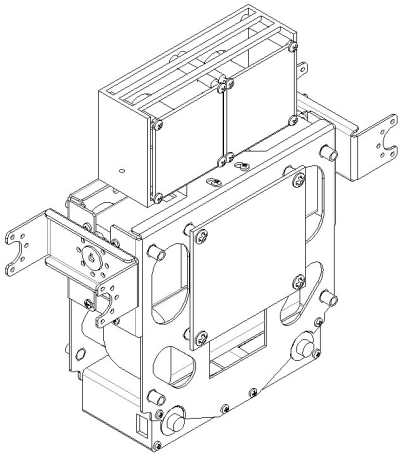
左手臂的装配

步骤	装配内容	图示
步骤 1	① 将舵机 5、6、11、12 号孔螺丝拧出，同时拆除底部外壳，并将四个孔依次对准手掌的 3、4、2、1 号孔，用四颗 M2*16 自攻螺丝固定。 ② 将舵机装上主、副舵盘。	
步骤 2	① 将舵机和两个尖形支架用六颗 M2*4 的自攻螺丝连接固定。 ➢ 注意，主舵盘缺口应对准尖形支架 4 号孔。	

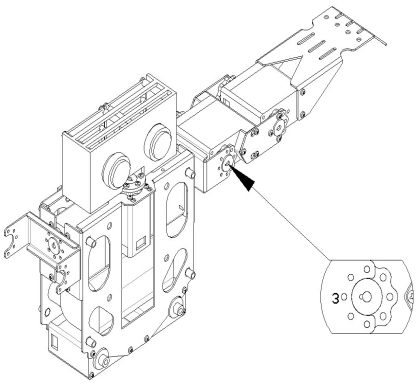
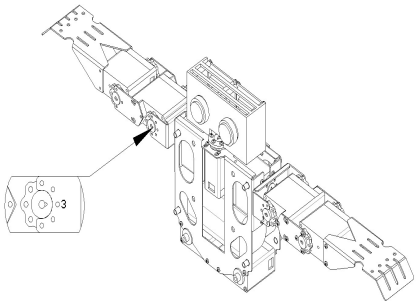
步骤 3	① 将舵机的 5、6、11、12 号孔螺丝拆除，保留底部外壳。 ② 将舵机 5、6、11、12 依次对准两个尖形支架的 2、3 号孔，用四颗 M2*20 的自攻螺丝将其固定。 ③ 用两颗 M2*4 的螺丝分别拧入两个尖形支架的 1 号孔。	
步骤 4	① 将舵机装上主、副舵盘。 ② 右手与左手沿竖直方向对称。	

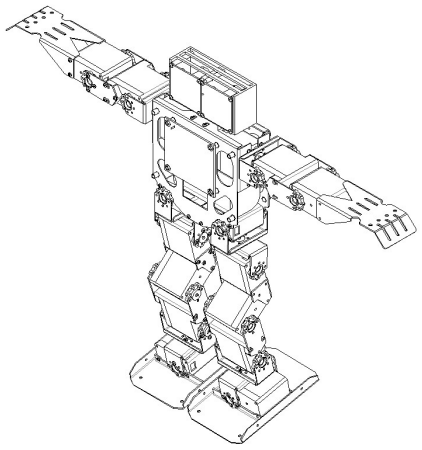
头部的装配

步骤	装配内容	图示
步骤 1	① 将控制头部转向的舵机装上主舵盘。 ② 用四颗 M2*4 的自攻螺丝将头部外壳和舵盘连接。	
步骤 2	① 用八颗 M2.3*16 的半牙自攻螺丝将两个摄像头固定。	

步骤 3	① 用四颗 M3*5 的自攻螺丝将主板固定。	

总装

步骤	装配内容	图示
步骤 1	① 将左手最上方舵机装上主、副舵盘。 ② 用六颗 M2*4 的自攻螺丝将左手和身体连接固定。 ➢ 注意，主舵盘缺口应对准 U 形支架的 3 号孔。	
步骤 2	① 将右手最上方舵机装上主、副舵盘。 ② 用六颗 M2*4 的自攻螺丝将右手和身体连接固定。 ➢ 注意，主舵盘缺口应对准 U 形支架的 3 号孔。	

步骤 3	① 将身体左下、右下角的舵机装上主、副舵盘。 ② 用十二颗 M2*4 的自攻螺丝将双腿与身体连接。	

第三部分：驱动板硬件电路简介

DR24 舵机驱动板是一款基于 STM32F103 单片机设计的 24 路舵机驱动板，其主要作用是作为舵机驱动板来驱动全爱科技的人形机器人，也可作为其他产品如：机器人、机械臂等的舵机驱动板。

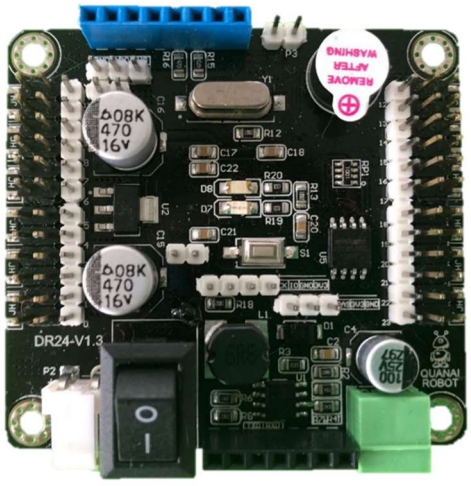


图 3.1　　DR24 舵机驱动板

3.1 驱动板参数介绍

驱动板参数：

外形尺寸：60mm×55mm
安装孔距：53mm×48mm
安装孔大小：3.175mm(直径)
重量：26.8g
最高高度：1.9cm
输入电压范围：6~12V
输入电压推荐范围：7~9V
升压电路输出电压：12V
升压电路输出最大电流：10A

3.2 驱动板预留接口定义

驱动板当前版本为 V1.3，其预留接口定义如图 3.2 所示。

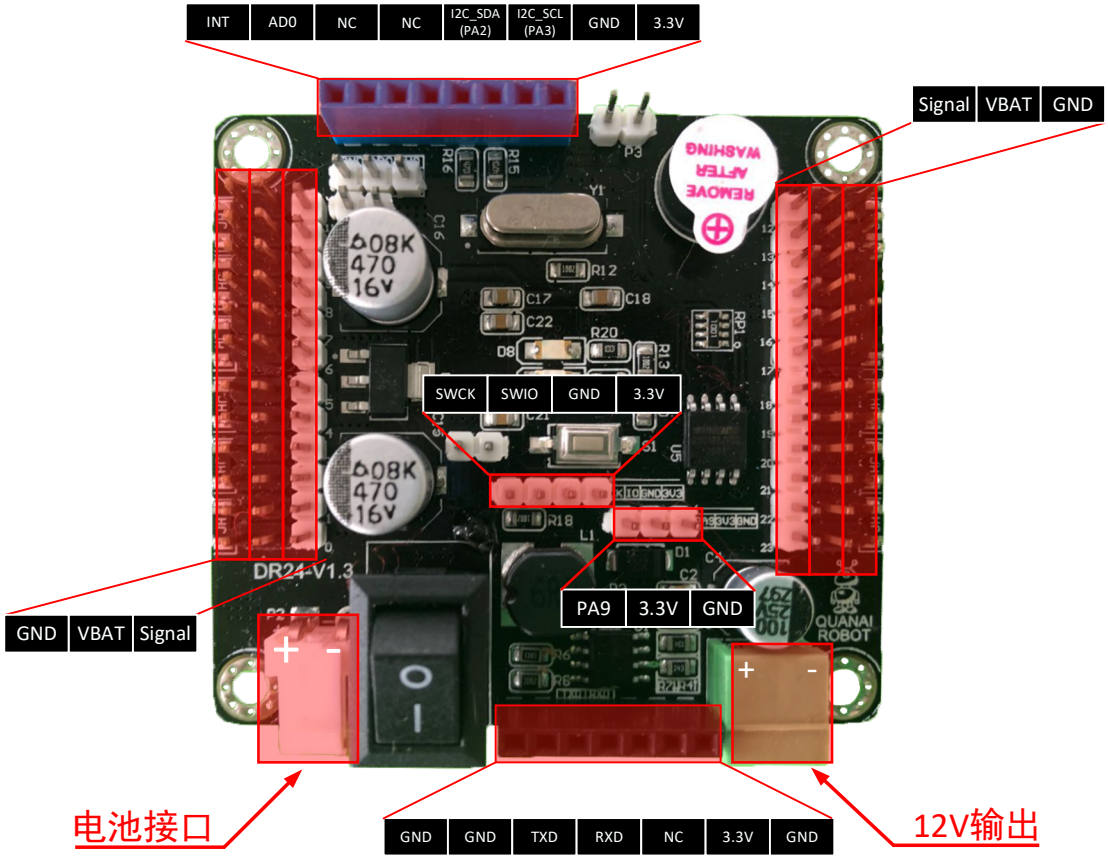


图 3.2　　DR24-V1.3 驱动板接口示意图

上图中两侧的排针即为舵机接口。左右两侧最里面的一排为信号线接口；中间一排为电源线接口；最外面的一排为地线接口。

左下侧的电池接口为驱动板的供电接口；右下侧的 12V 输出接口用来为 QA200BOX 供电。

驱动板上方的蓝色 7Pin 排母可用来连接全爱科技提供的 MPU6050 模块；驱动板下方的黑色 7Pin 排母可用来连接全爱科技提供的 LoRa 模块。

此外，驱动板还预留了 PA9 端口，以及一组 swd 接口用来调试。

第四部分：舵机的工作原理

4.1 舵机简介

舵机（英文名 Servo）：是一种伺服电机。它由直流电机、减速齿轮组、传感器和控制电路组成的一套自动控制系统。通过发送信号，指定输出轴旋转角度。一般而言舵机都有最大旋转角度，与普通直流电机的区别主要在于：直流电机是一圈圈转动的，舵机只能在一定角度内转动，不能一圈圈转（数字舵机可以在舵机模式和电机模式中切换，没有这个问题）。普通直流电机无法反馈转动的角度信息，而舵机可以。

舵机的用途与电机也不相同，普通直流电机一般是整圈转动做动力用，舵机是控制某物体转动一定角度。比如作为机器人的各个关节控制机器人的运动状态。

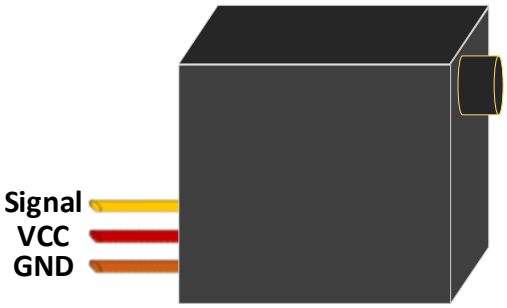


图 4.1　　舵机示意图

4.2 舵机的工作原理

舵机的伺服系统由可变宽度的脉冲（PWM）来进行控制，控制线是用来传送脉冲的。脉冲的参数有最小值，最大值，和频率。一般而言，舵机的基准信号都是周期为 20ms，宽度为 1.5ms。这个基准信号定义的位置为中间位置。舵机有最大转动角度，中间位置的定义就是从这个位置到最大角度与最小角度的量完全一样。最重要的一点是，不同舵机的最大转动角度可能不相同，但是其中间位置的脉冲宽度是一定的，那就是 1.5ms。

全爱机器人使用的舵机关于 PWM 控制说明如图 4.2 所示。其从 0°到 180°对应 0.5ms 到 2.5ms（500μs -2500μs）。

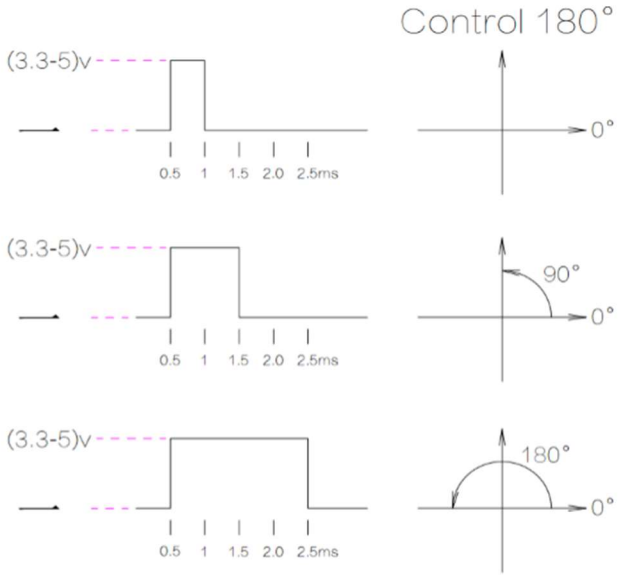


图 4.2　PWM 控制舵机示意图

例如：舵机在出厂时一般都设定在中位，即图 4.2 中 90°的位置。此时脉冲宽度为 1.5ms，我们想要将舵机旋转至 180°，只需要将脉冲宽度调节为 2.5ms 即可。这一过程的具体实现方法将在后面为机器人设计动作时做详细说明。

第五部分：机器人的校准

刚组装好的机器人或多或少都会存在误差，这种误差的产生原因包括安装误差、舵机本身误差、结构件形状误差、定位孔误差等，如果没有对机器人的误差进行校准，那么在后续的开发过程中可能会出现很多问题，如果手臂部位出现误差会导致实际运行的动作与设计动作有差异；如果腿部出现误差，则会导致机器人在运行时摔倒；如果头部出现误差，则会导致机器人的视野偏移。

机器人的误差是无法避免的，而且误差导致的问题也有很多，轻则导致动作不协调，重则导致舵机堵转、机器人摔倒。在设计过程中，我们可以通过各种方式来尽量将误差降到最低。

5.1 机器人校准方法

在调试过程中，根据机器人误差产生的不同原因有不同的校准方法，大致分为以下几种：

调整/更换结构件

适用场景：结构件出现明显形变或结构件上的定位孔出现明显偏差。

更换舵机

适用场景：舵机损坏。

软件调节

适用场景：在结构件无明显形变、舵机无损坏的情况下，做校准动作时关节存在肉眼可见但不严重的误差。

注意：软件调节误差只适用于舵机本身的偏差调节，如存在严重误差，则需要对产生误

差的部分重新安装。

重新安装

在结构件无明显形变、舵机无损坏的情况下，做校准动作时存在肉眼可见的严重误差。

5.2 软件调节误差方法

在上文提及的机器人校准方法中，“调整/更换结构件”、“更换舵机”及“重新安装”可以理解成对机器人组装的重复操作，在此不做详细说明。这里主要介绍一下软件调节舵机误差的方法。

舵机、舵盘存在的误差都可以用软件的方法进行调节，其具体方法如下：

方法 1：在 IAR 中直接调节

打开项目资料文件夹→“Code”→“DR24_Code”→“EWARM”→“Project.eww”，在左侧的“workspace”窗口中点开“UserCode”文件夹前面的“ + ”，找到“SteeringEngine.c”，如图 5.1 所示。

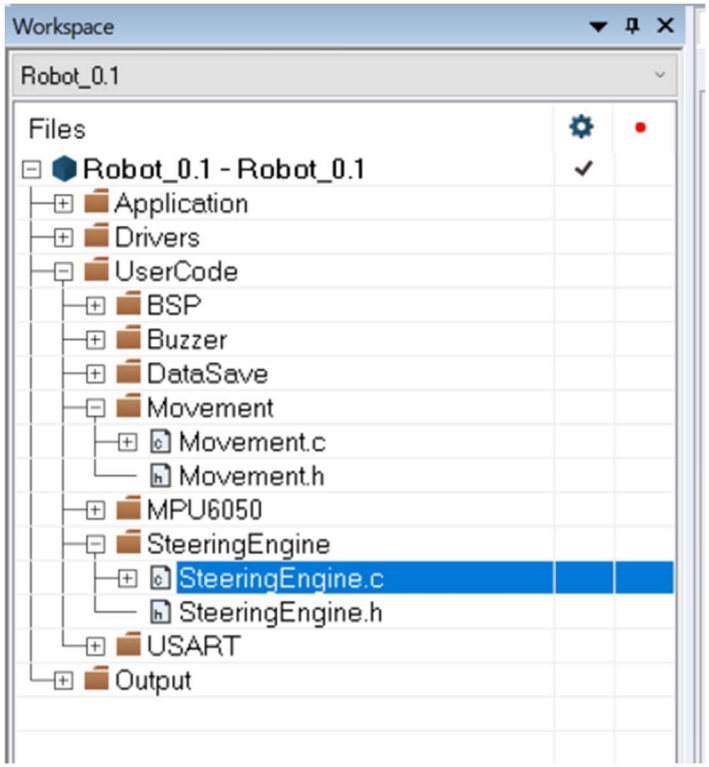
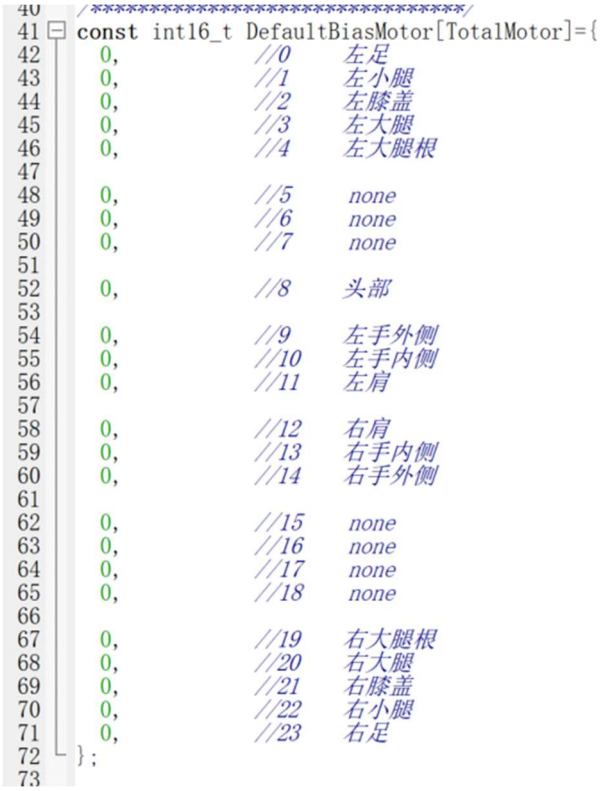


图 5.1　　SteeringEngine.c 文件位置

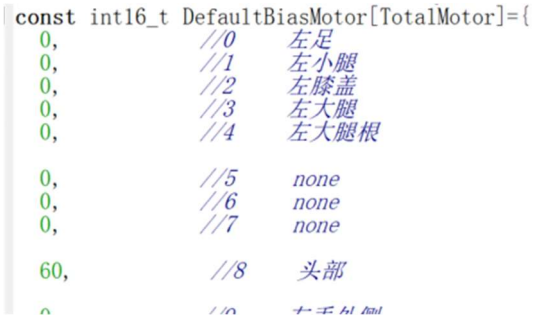
双击“SteeringEngine.c”，在代码窗口中的第 41 行，找到“const int16_t DefaultBiasMotor [TotalMotor]，如图 5.2 所示。



```
/*********************************/
const int16_t DefaultBiasMotor[TotalMotor]={
  0,          //0     左足
  0,          //1     左小腿
  0,          //2     左膝盖
  0,          //3     左大腿
  0,          //4     左大腿根

  0,          //5     none
  0,          //6     none
  0,          //7     none

  0,          //8     头部

  0,          //9     左手外侧
  0,          //10    左手内侧
  0,          //11    左肩

  0,          //12    右肩
  0,          //13    右手内侧
  0,          //14    右手外侧

  0,          //15    none
  0,          //16    none
  0,          //17    none
  0,          //18    none

  0,          //19    右大腿根
  0,          //20    右大腿
  0,          //21    右膝盖
  0,          //22    右小腿
  0,          //23    右足
};

```

图 5.2　　舵机误差修改位置

在上一章中提到舵机的工作原理是 PWM。这里的误差调节所修改的参数也是脉冲宽度。比如头部的舵机存在误差，可将头部的参数做适当修改，如图 5.3 所示。



```
const int16_t DefaultBiasMotor[TotalMotor]={
  0,          //0     左足
  0,          //1     左小腿
  0,          //2     左膝盖
  0,          //3     左大腿
  0,          //4     左大腿根

  0,          //5     none
  0,          //6     none
  0,          //7     none

  60,          //8     头部
```

图 5.3　　头部舵机修改示例

图 5.3 中将头部的参数修改为 60，也就是大概做了 5.5°的误差校准。

需要注意的是：在校准误差的同时也修改了舵机的初始值。比如头部的舵机初始化状态下位置为 90°，此时舵机向左和向右的运动范围均为 90°，若是因为舵盘的误差导致需要做误差校准，在做了 5.5 度的误差校准后，机器人头部舵机左右运动范围就会发生变化。某一方向的运动范围会从 90°变为 84.5°。所以若存在较大的误差，建议使用其他方法进行调整，以免影响舵机的运动范围。该参数修改范围推荐为±150。

修改完成后，点击上方的“Download and Debug”，将程序烧录到驱动板中后，点击“Run”即完成舵机误差调整。

方法 2：使用 QARob-Nano 模块调节

使用 USB 为 QA-Nano 模块供电，打开项目资料文件夹→“Code”→“ErrorAdjust”，使用

Arduino IDE 开发环境打开“ErrorAdjust.ino”文件，将程序切换到“Adjust”窗口，如图 5.4 所示。

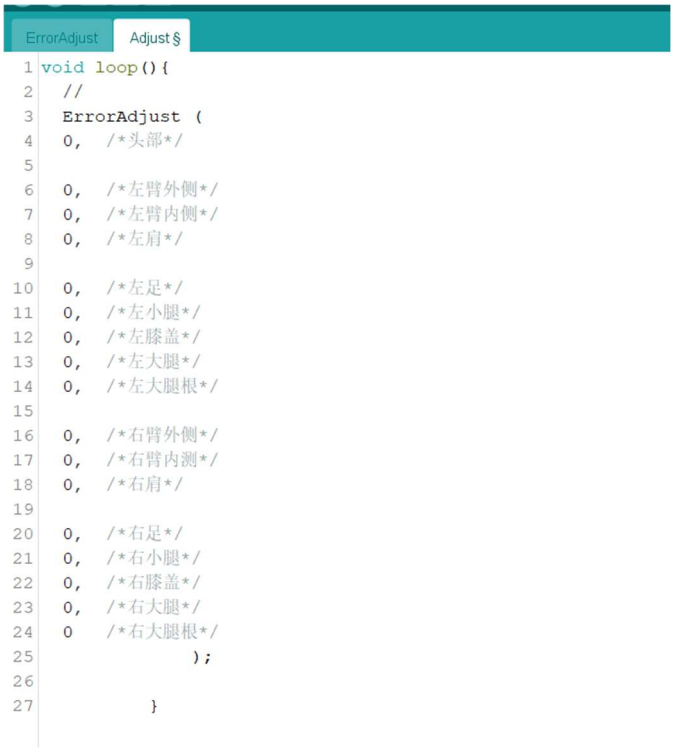


图 5.4　　Arduino IDE 软件中调节机器人误差

图 5.4 所示的参数同样是 PWM 的高电平时间，修改完成后点击“上传”按钮即可。

第六部分：机器人的动作设计

机器人的动作设计实际上是对舵机的转动角度进行修改。在这一过程中需要注意机器人的重心及动作的连贯性（包括动作的关键帧和每一个动作的完成时间）。本章将对机器人的动作设计做出详细说明。

6.1 机器人动作的衔接

“动作衔接”是一个电影术语，指主体动作具有连贯性的剪接方法。其目的是使上一个镜头与下一个镜头的转换连接具有连续性而无跳跃感。在设计动作的过程中，想要让自己设计的动作无跳跃感，就要找到动作的关键帧。例如图 6.1 所示的一个简单的抬手动作，我们只需要设计一组动作即可完成，中间的运动过程不需设计，机器人会自动补全。

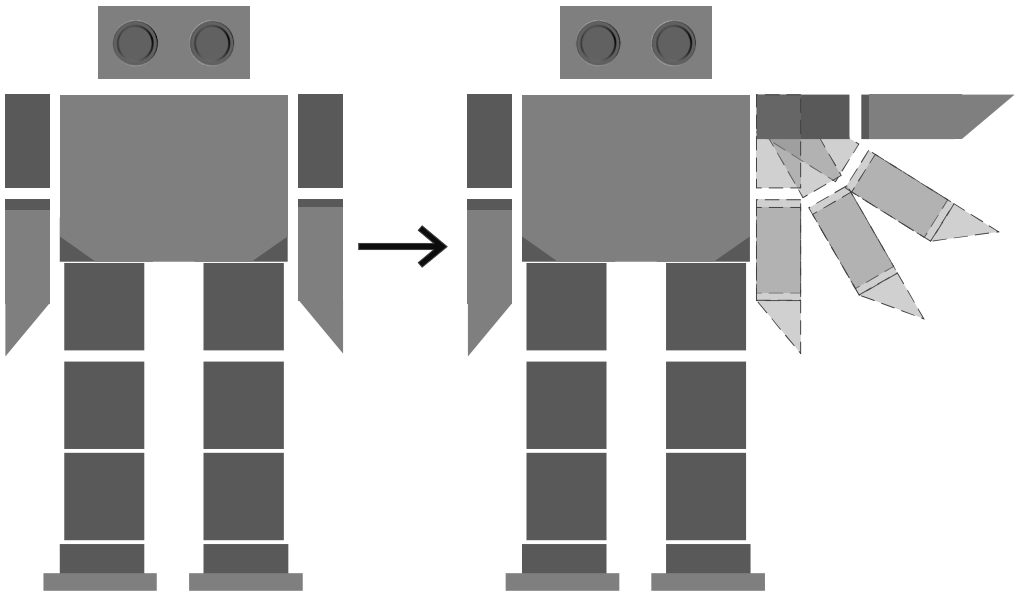


图 6.1 机器人抬手动作的关键帧

6.2 机器人动作的时间规划

机器人的运动时间规划同样是一个非常重要的环节，尤其是在为机器人设计舞蹈动作的时候，如果没有规划好每一个动作完成的时间，就会使机器人跟不上节奏，甚至会导致机器人运动不协调，出现摔倒的情况。

6.3 机器人动作设计方法

打开项目资料文件夹→“Code”→“DR24_Code”→“EWARM”→“Project.eww”，在左侧的“workspace”窗口中点开“UserCode”文件夹前面的“+”，找到“Movement.c”，如图 6.2 所示。

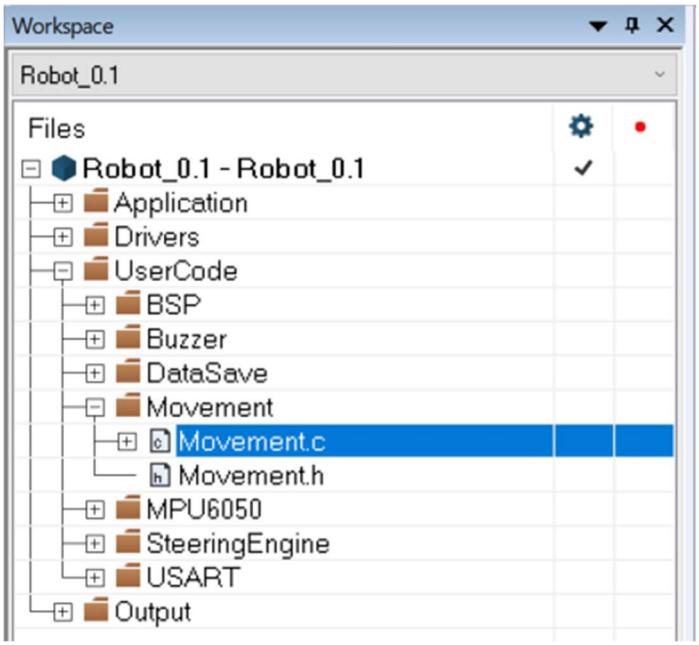


图 6.2　　机器人动作设计文件位置

在这个 c 文件中，我们可以看到许多“const ACTION_GROUP xx[ ]”这样的数组。这些就是已经设计好的动作组文件。如图 6.3 所示。

```
const ACTION_GROUP MovementReady[1]={500, {1500,1800,1000,1500,1500,0,0,0,1500,1500,1500,1500,
                                           1500,1500,1500,0,1500,0,0,1500,1500,2000,1200,1500}};
```

图 6.3　　机器人动作组设计

图 6.3 所示的“MovementReady”是机器人的校准动作，后面的”[1]”表示该动作组中只有一组动作，第一个大括号里面的“500”即为完成该组动作所需时间，单位是毫秒。

在第四部分中，我们有介绍过舵机角度的改变方法是改变脉冲宽度，后面的大括号里面有 24 个参数，第 0 项参数到第 23 项参数对应的即是机器人驱动板后面从 0 到 23 的 24 路舵机接口。其中的每一项参数即为该位号舵机的脉冲宽度，单位是微秒。

例如：我们现在想将 0 号舵机从 90°修改为 180°，只需要将大括号中的第一项参数“1500”改为“2500”即可（0°→180°对应 500-2500）。

为方便动作组的设计，可在”Movement.c”中找到”RobTest”数组，如图 6.4 所示，该数组是方便单一动作组设计的。

```
const ACTION_GROUP RobTest[1]={
                                  1000,    //t  完成时间
                                 {
                                 /* 1500-> */  1300,    // H 编号:  0  左脚1,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往外
                                 /* 1800-> */  1050,    // G 编号:  1  左脚2,        初始1800  范围:  500 ~ 2500 → 0°~ 180°越大越往后
                                 /* 1000-> */  2000,    // F 编号:  2  左脚3,        初始1000  范围:  500 ~ 2500 → 0°~ 180°越大越往后
                                 /* 1500-> */  1400,    // E 编号:  3  左脚4,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往前
                                 /* 1500-> */  1300,    // D 编号:  4  左脚5,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往里
0 ,0 ,0 ,
                                 /* 1500-> */  1500,    // M 编号:  8  头部,         初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往右

                                 /* 1500-> */  1500,    // A 编号:  9  左臂最外侧,   初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往上
                                 /* 1500-> */  2500,    // B 编号: 10  左臂中间,     初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往上
                                 /* 1500-> */   500,    // C 编号: 11  左臂内,       初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往前

                                 /* 1500-> */  2500,    // C 编号: 12  右臂内,       初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往后
                                 /* 1500-> */   500,    // B 编号: 13  右臂中间,     初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往下
                                 /* 1500-> */  1500,    // A 编号: 14  右臂最外侧,   初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往下
0 , 0 , 0 , 0 ,
                                 /* 1500-> */  1700,    // D 编号: 19  右脚5,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往外
                                 /* 1500-> */  1600,    // E 编号: 20  右脚4,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往后
                                 /* 2000-> */  1000,    // F 编号: 21  右脚3,        初始2000  范围:  500 ~ 2500 → 0°~ 180°越大越往前
                                 /* 1200-> */  1950,    // G 编号: 22  右脚2,        初始1200  范围:  500 ~ 2500 → 0°~ 180°越大越往前
                                 /* 1500-> */  1700     // H 编号: 23  右脚1,        初始1500  范围:  500 ~ 2500 → 0°~ 180°越大越往里
                                 };
```

图 6.4　　单一动作组调试

在设计好动作后，可以新建一个数组，名字自己定义，例如设计了一个关键帧个数为 30 的动作取名为“TestMove”。方法为在 Movement.c 文件中新建一个数组，“const ACTION_GROUP TestMove[30]”,然后将动作组写入该数组即可，具体注意事项可参照 Movement.c 文件中的其他动作组格式。

在设计好新的动作组后并写入后，还需要进行以下步骤的操作。此处以上文中所举的“TestMove”为例。

步骤 1：在图 6.5 所示的位置加入设计好的动作组名称。

```
const ACTION_GROUP* AllActionGroup[TOTAL_ACTION_GROUP+1]=
  {
    /*0*/   StandingUp,                 //undefined 站立
    /*1*/   MovementReady,              //undefined 准备动作
    /*2*/   MovementStop,               //undefined 停止动作
    /*3*/   RobCorrect,                 //undefined 机器人校准
    /*4*/   Squat,                      //DOL 下蹲
    /*5*/   GoForward,                  //DOA 前进
    /*6*/   GoForwardSlow,              //DOF 前进（慢）
    /*7*/   GoBack,                     //DOB 后退
    /*8*/   GoBackSlow,                 //DOG 后退（慢）
    /*9*/   TurnLeft,                   //DOC 左转（原地）
    /*10*/  TurnLeft_2,                 //undefined 左转（移动）
    /*11*/  TurnRight,                  //DOD 右转
    /*12*/  MoveLeft,                   //DOH 左移
    /*13*/  MoveRight,                  //DOI 右移
    /*14*/  SquatGo,                    //undefined 下蹲前进
    /*15*/  Punch,                      //undefined 打拳动作
    /*16*/  FallForward_Standing,       //undefined 前倒爬起
    /*17*/  FallBack_Standing,          //undefined 后倒爬起
    /*18*/  WolfDisico,                 //undefined 舞蹈
    /*19*/  TrafficRight,               //DOK 交通指挥（右）
    /*20*/  TrafficLeft,                //DOJ 交通指挥（左）
    /*21*/  TrafficStop,                //undefined 交通指挥（停止）
    /*22*/  TrafficParkingArea,         //undefined 交通指挥（停车场）
    /*23*/  TurnLeftSmallAngle,         //DOM 左转（小角度）
    /*24*/  TurnRightSmallAngle,        //DON 右转（小角度）
    /*25*/  RobTest,                    //undefined 动作测试
    /*26*/  OneTest,                    //undefined 动作调试
    /*27*/  GroupTest,                  //undefined 动作组测试
            TestMove,
            UserDefined
```

图 6.5　　步骤 1

步骤 2：在“ActionSpeedControl”中添加一个参数“100”，如图 6.6 所示。

```
uint16_t ActionSpeedControl[TOTAL_ACTION_GROUP+1]={100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
                                                   100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
                                                   100, 100, 100, 100, 100, 100, 100, 100, 100, 100};
```

图 6.6　　步骤 2

步骤 3：在* AllActionGroup 的数组最后一位“0”前面插入动作组关键帧个数，如图 6.7 所示。

```
const uint16_t AllActionIndex[TOTAL_ACTION_GROUP+1]={1, 1, 1, 1, 1, 4, 4,
                                                     4, 4, 4, 5, 4, 3, 3, 6, 2, 18,
                                                     7, 42, 1, 1, 1, 8, 4, 4, 1, 1, 4,30, 0};
```

图 6.7　　步骤 3

步骤 4：在左侧的“WorkSpace”中找到“UserCode”→“Movent”→“Movement.h”文件，找到“#define TOTAL_ACTION_GROUP”，如图 6.8 所示。

```
#define TOTAL_ACTION_GROUP                    29
```

图 6.8　　步骤 4

此处的“29”即为当前的动作组个数，由于我们设计了一个新的动作组，就要在这里加 1，即此处改为“30”。后续增加或删除动作组，此处都应作出相应的修改。

机器人驱动板上面有一个轻触开关，可用来脱机工作，也可用来做动作测试，其使用方法如下（以“TestMove”为例）。

步骤 1：在图示位置确定“TestMove”动作序号（注意：数组是从 0 开始的），如图 6.9 所示。此处“TestMove”为 28 号。

```
const ACTION_GROUP* AllActionGroup[TOTAL_ACTION_GROUP+1]=
  {
    /*0*/   StandingUp,               //undefined 站立
    /*1*/   MovementReady,            //undefined 准备动作
    /*2*/   MovementStop,             //undefined 停止动作
    /*3*/   RobCorrect,               //undefined 机器人校准
    /*4*/   Squat,                    //DOL 下蹲
    /*5*/   GoForward,                //DOA 前进
    /*6*/   GoForwardSlow,            //DOF 前进（慢）
    /*7*/   GoBack,                   //DOB 后退
    /*8*/   GoBackSlow,               //DOG 后退（慢）
    /*9*/   TurnLeft,                 //DOC 左转（原地）
    /*10*/  TurnLeft_2,               //undefined 左转（移动）
    /*11*/  TurnRight,                //DOD 右转
    /*12*/  MoveLeft,                 //DOH 左移
    /*13*/  MoveRight,                //DOI 右移
    /*14*/  SquatGo,                  //undefined 下蹲前进
    /*15*/  Punch,                    //undefined 打拳动作
    /*16*/  FallForward_Standing,     //undefined 前倒爬起
    /*17*/  FallBack_Standing,        //undefined 后倒爬起
    /*18*/  WolfDisico,               //undefined 舞蹈
    /*19*/  TrafficRight,             //DOK 交通指挥（右）
    /*20*/  TrafficLeft,              //DOJ 交通指挥（左）
    /*21*/  TrafficStop,              //undefined 交通指挥（停止）
    /*22*/  TrafficParkingArea,       //undefined 交通指挥（停车场）
    /*23*/  TurnLeftSmallAngle,       //DOM 左转（小角度）
    /*24*/  TurnRightSmallAngle,      //DON 右转（小角度）
    /*25*/  RobTest,                  //undefined 动作测试
    /*26*/  OneTest,                  //undefined 动作调试
    /*27*/  GroupTest,                //undefined 动作组测试
          TestMove,
          UserDefined
```

图 6.9　　确认动作组编号

步骤 2：在程序中找到“uint8_t NumDoAction = ?”，如图 6.10 所示，将“? ”处的数字改为“TestMove”动作组的编号即：“uint8_t NumDoAction = 28”。

```
int stepAction=0;
uint8_t NumDoAction=11;                    // 所运行动作的编号
```

图 6.10　修改按键运行的动作组编号

全部修改完成后，点击“Make”,如果没有报错，将机器人与电脑连接，点击“Download and Debug”,将程序烧写到机器人驱动板中即完成了动作的设计。

第七部分：驱动板与其他设备通信

我们设计好机器人的动作之后，动作组会保存在驱动板的 flash 中，想要让其他控制板下发指令给驱动板，就需要控制板与驱动板进行通信。

全爱机器人通过摄像头采集数据，经过 QA200BOX 对数据进行处理后，将指令通过串口下发给 DR24 舵机驱动板，再由驱动板控制舵机完成相应的动作。如图 7.1 所示。

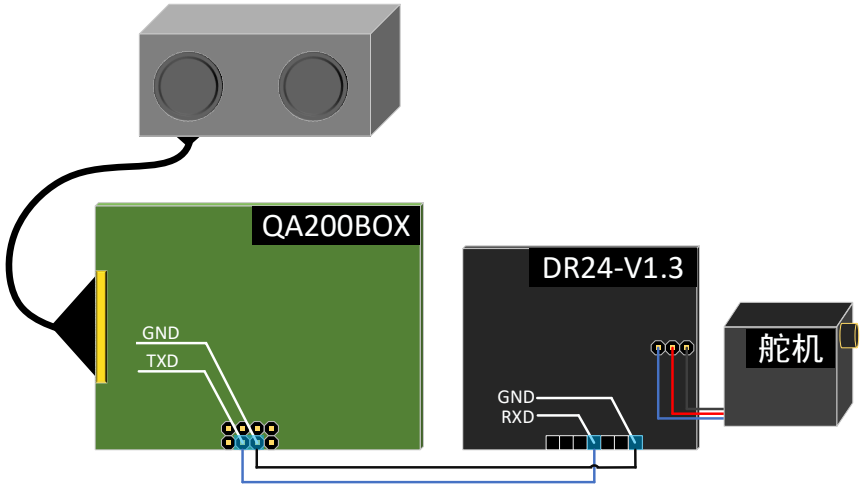


图 7.1　全爱机器人系统简图

驱动板串口通信波特率为 115200，在程序中，我们可以将自己写好的动作组做成相应的“命令”，驱动板接收到该命令，就会执行相应的动作。

我们可以通过 USB 转串口工具将机器人与电脑连接，再通过串口助手进行调试。

7.1 串口指令的设置

例如我们在”Movement.c”中新建了一个有 30 个关键帧的动作组“const ACTION_GROUP TestMove[30]”，在图 7.2 所示的位置中确定动作组的序号，此处“TestMove”动作组序号为 28 号。

```
const ACTION_GROUP* AllActionGroup[TOTAL_ACTION_GROUP+1]=
{
  /*0*/   StandingUp,                 //undefined 站立
  /*1*/   MovementReady,              //undefined 准备动作
  /*2*/   MovementStop,               //undefined 停止动作
  /*3*/   RobCorrect,                 //undefined 机器人校准
  /*4*/   Squat,                      //DOL 下蹲
  /*5*/   GoForward,                  //DOA 前进
  /*6*/   GoForwardSlow,              //DOF 前进（慢）
  /*7*/   GoBack,                     //DOB 后退
  /*8*/   GoBackSlow,                 //DOG 后退（慢）
  /*9*/   TurnLeft,                   //DOC 左转（原地）
  /*10*/  TurnLeft_2,                 //undefined 左转（移动）
  /*11*/  TurnRight,                  //DOD 右转
  /*12*/  MoveLeft,                   //DOH 左移
  /*13*/  MoveRight,                  //DOI 右移
  /*14*/  SquatGo,                    //undefined 下蹲前进
  /*15*/  Punch,                      //undefined 打拳动作
  /*16*/  FallForward_Standing,       //undefined 前倒爬起
  /*17*/  FallBack_Standing,          //undefined 后倒爬起
  /*18*/  WolfDisico,                 //undefined 舞蹈
  /*19*/  TrafficRight,               //DOK 交通指挥（右）
  /*20*/  TrafficLeft,                //DOJ 交通指挥（左）
  /*21*/  TrafficStop,                //undefined 交通指挥（停止）
  /*22*/  TrafficParkingArea,         //undefined 交通指挥（停车场）
  /*23*/  TurnLeftSmallAngle,         //DOM 左转（小角度）
  /*24*/  TurnRightSmallAngle,        //DON 右转（小角度）
  /*25*/  RobTest,                    //undefined 动作测试
  /*26*/  OneTest,                    //undefined 动作调试
  /*27*/  GroupTest,                  //undefined 动作组测试
          TestMove,
          UserDefined
```

图 7.2　　确定动作组的序号

确定了动作组序号之后，在左侧的“workspace”中“UserCode”文件夹下的”UART”文件夹中找到 uart3.c 文件。如图 7.3 所示。

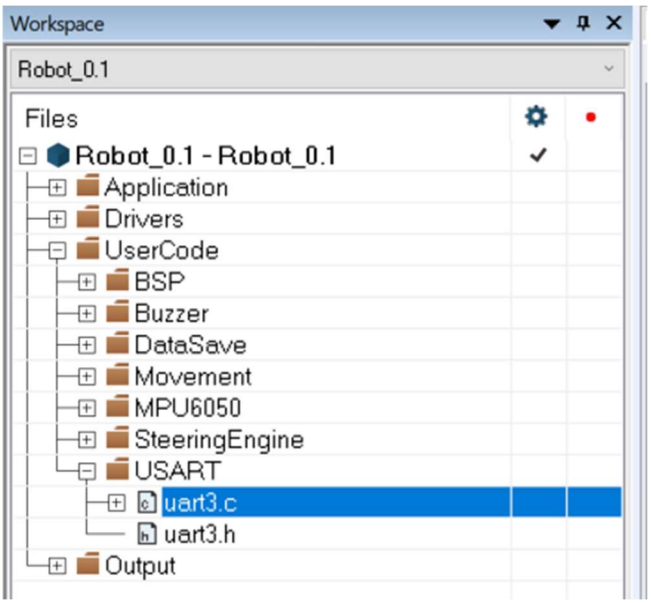


图 7.3　　uart3.c 文件位置

找到该 c 文件中的 switch 语句，如图 7.4 所示。

```
void BLE_Cmd_Respond(UART_HandleTypeDef *huart)
{
  switch(Uart3_RxBuf[2])
  {
  case 'A':case'a': //前进
    if(StateMove==STOP)
    {
      StateMove = START;
      stepAction = 0;
      NumDoAction = 5;
      TimesDoAction = 0;
      TimesDoActionCache=TimesDoAction;
    }
    break;
```

图 7.4　　switch 语句位置

在上图的 switch 语句中加入新的 case’值’，例如我们希望 28 号动作组的值为”C”,在

switch 语句中增加如下代码即可：

```
case 'C':
    if(StateMove == STOP)
    {
        StateMove =START;
        stepAction = 0;
        NumDoAction = 28;
        TimesDoAction = 0;
        TimesDoActionCache = TimesDoAction;
    }
    break;
```

完成添加后，将程序烧录到驱动板，打开“项目资料文件夹”→“串口助手”，运行里面的应用程序。通过 Jlink 自带的串口工具或 usb 转串口工具将机器人的串口与电脑相连。选择端口号，波特率设置为 115200，数据位设置为 8，奇偶校验位设置为 None;停止位设置为 1，如图 7.5 所示。

图 7.5 串口设置

设置完成后点击“打开串口”，在发送栏输入“DOC”，点击“发送”，机器人即执行相应的动作。