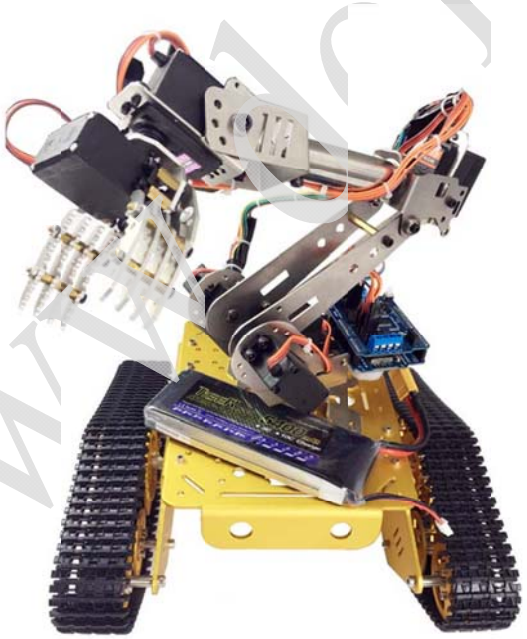
Doctors of Intelligence & Technology (DOIT)

移动 DoArm 机械臂使用手册(V1.0)



2016-4-22

目录

简介

本手册主要简单地介绍了 doit 公司 DoArm 机械臂的使用手册。在此平台上也可以进行二次开发，并完善相应的功能。谢谢！
购买地址：
https://item.taobao.com/item.htm?spm=a1z10.5-c.w4002-7420481794.16.SR7npF&id=530538902892

1 机械臂

工业六轴机械臂广泛应用到工厂自动化焊接、装配、喷涂和组装等场合，商业应用已经非常广泛。本章以 ABB 六轴机械臂“IRB4400”为原型等比例缩小的模型为基础，基于 ESPDuino 和 Doit 自主研发的 2 路电机 16 路舵机驱动板，构建一款可 WiFi 控制的机械臂，实现简单的抓取任务。DoArm 机械臂的特点如下：

（1）ABB IRB4400 机械臂等比例缩小模型机械臂，见下图所示；

（2）基于 ESP8266 SOC 实现 WiFi 控制；

（3）通过舵机驱动板实现六路舵机和一路抓手控制；

（3）内置 Telnet 调试界面，OTA 升级功能；

（4）内置机械臂逆运动学算法，可实现机械臂末端在笛卡尔坐标系或工具坐标系下的移动。

图 1 机械臂原型 ABB IRB4400

2 产品概述

DoArm 机械臂的功能包括如下几部分。

（1）标定和初始化

机械臂安装完成后，首先需要对舵机进行位置中位标定，确保各轴舵机在程序启动的时候达到期望的初始位置。同时要确定各轴的舵机的安全移动范围，保证各结构件以及舵机自身的安全。

（2）单轴控制

ESPduino 通过 I2C 接口实现控制驱动板上的 16 路舵机。针对机械臂上的六路舵机和抓手舵机，程序中通过指定驱动板上的通道号实现一一对应控制。

（3）多轴联动控制

根据 D-H 连杆矩阵的建立方法，建立机械臂的连杆矩阵获得其运动学方程，进而求解其逆运动学方程，实现机械臂末端的移动。

（4）Telenet 人机交互

利用 ESPDuino，建立 Telenet 服务器，实现简单的人机交互界面，完成机械臂的控制和状态显示。

DoArm 机械臂的系统结构如下所示。机械臂上安装 ESPDuino 和驱动板，通过 WiFi 与无线路由器相连，笔记本或者手机通过终端或 APP 实现机械臂控制。

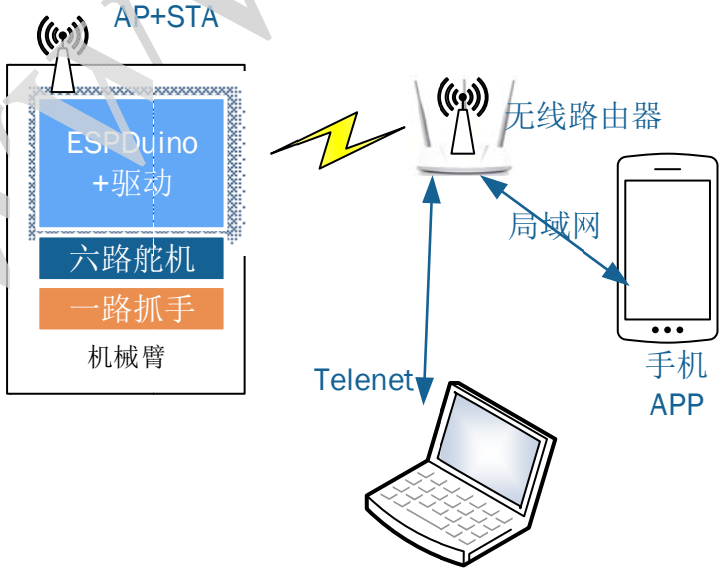


图 2 机械臂系统结构图

本产品所需的物料包括：板载 ESP8266 SOC 的 ESPDuino 开发板，Micro USB 线，2 路电

机 16 路舵机驱动板，以及机械臂零件一套、抓手一套。

3 机械臂安装

机械臂的安装可参考《机械臂安装手册》。在安装的时候，结构件和舵机需要同步安装。在安装舵机之前，需要注意舵机的中位位置。默认将舵机旋转到 90 度位置作为中位。然后再将舵机和机械齿轮的中位对准安装。具体安装文档请访问：

DoArm 安装手册：http://bbs.doit.am/forum.php?mod=forumdisplay&fid=78

坦克底盘的安装手册：http://bbs.doit.am/forum.php?mod=forumdisplay&[illegible]57

初步安装完成的照片如下图所示。

机械臂套件：

https://item.taobao.com/item.htm?spm=a1z10.5-c.w4002-7420481794.23.4KWRin&id=530485070938

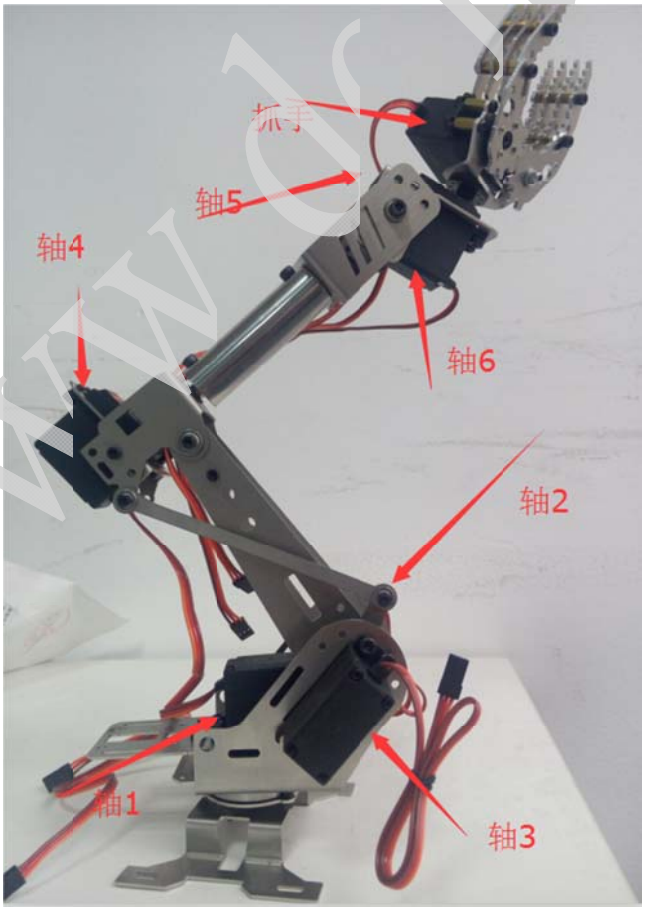
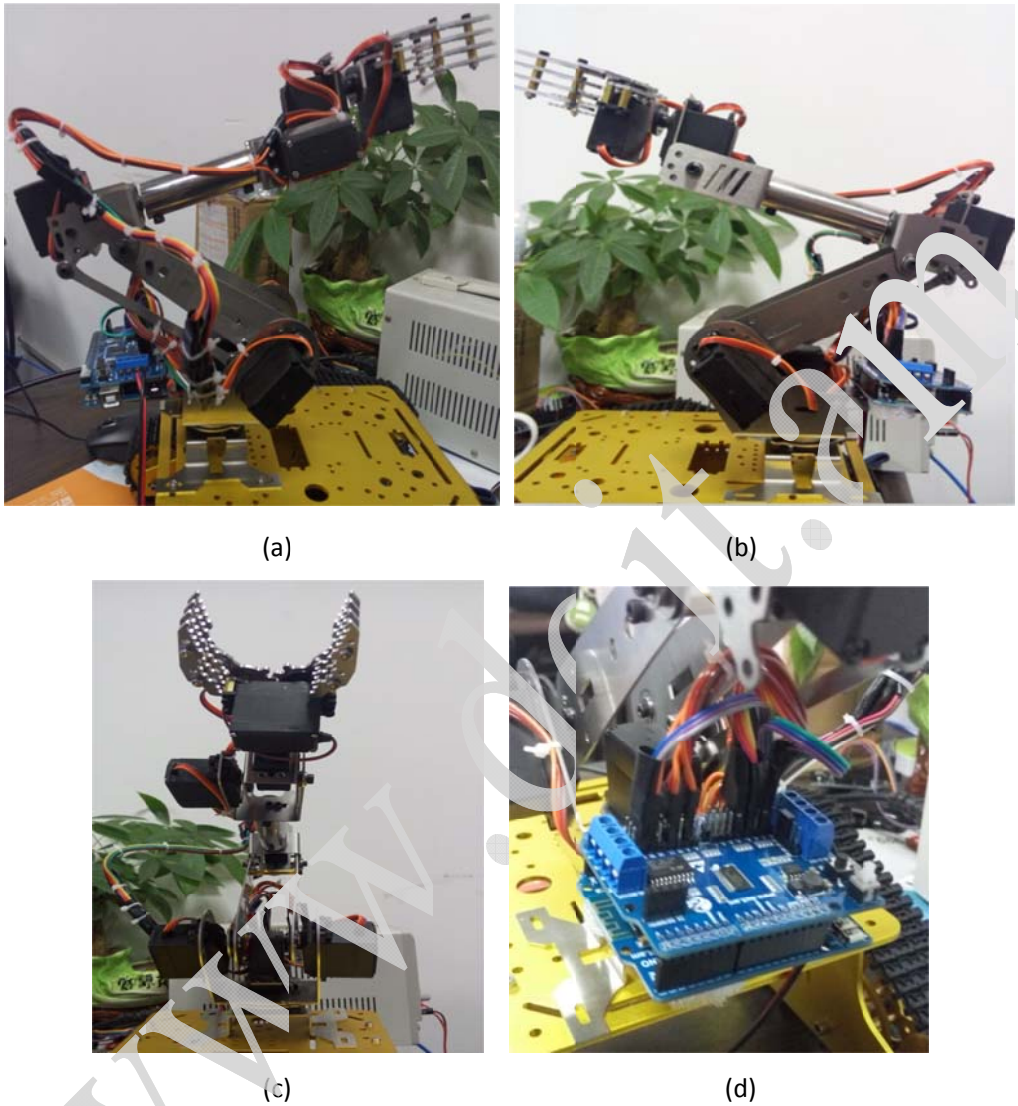


图 3 初装完成的机械臂

安装完成后需要将 ESPDuino 和驱动板安装到机械臂腰部，同时将各舵机的电源信号线接至驱动板上相应的位置。若舵机电源信号线不够长，可用三根杜邦线将其延长，插入驱动

板上合适的舵机通道。安装时，舵机通道可以随意指定，以后对每个舵机的控制可通过程序设定对应的通道号。

全部安装完成后的机械臂如下图所示。



(a) (b)

(c) (d)

(a)左侧视图 (b)右侧视图 (c)前视图 (d)控制板+驱动板

图 1.4 安装完成的机械臂

3 机械臂建模

机械臂使用六个舵机作为驱动元件。独立控制每一个舵机转到相应的角度很容易实现，但是如果要求六轴联动，实现机械臂末端执行器按照预定的位置移动，则必须对机械臂建立连杆坐标系，计算其连杆坐标系矩阵，建立运动学方程。若已知末端位姿，反过来求取各轴的转动角度值，则必须进行逆运动学求解。逆运动学求解不是一个简单的过程，超出了本书的范畴，感兴趣的读者可以参阅 2007 年高等教育出版社出版的《先进机器人控制》一书。

首先，定义机器人的初始位置如下图所示。定义基础坐标系的原点在轴 1 的正下方圆心处。基础坐标系的 X 轴在正前方，Z 轴竖直向上，根据右手定则确定 Y 轴方向。第 1~6 轴分别如 1.3 图所标示。

根据以上建立连杆坐标系。连杆坐标系各参数见下表。

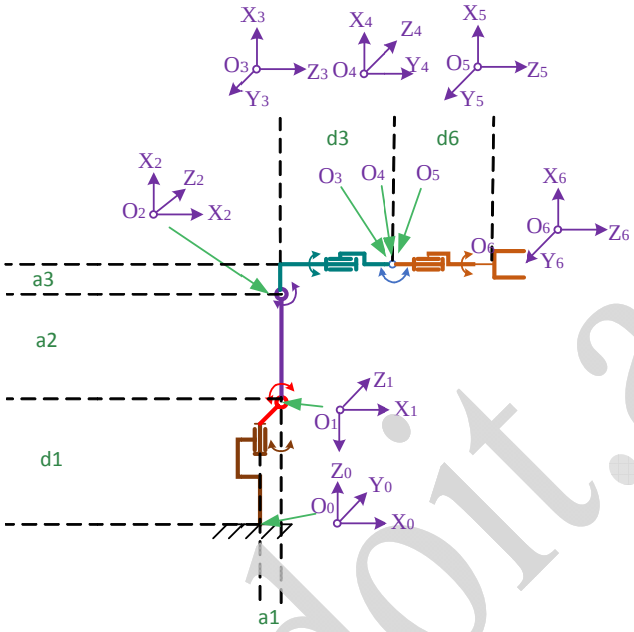


图 5 机械臂连杆坐标系

表 1.1 机械臂 D-H 参数表

连杆	关节角	扭转角	连杆长度(mm)	连杆偏移量(mm)
1	θ1	-90°	a1=45	d1=75
2	θ2	-90°	0°	a2=115	0
3	θ3	-90°	a3=20	d3=130
4	θ4	90°	0	0
5	θ5	-90°	0	0
6	θ6	0°	0	d6=50

4 机械臂控制

从 Github 代码仓库下载机械臂控制源码，编译后下载到 ESPDuino 并运行（源码下载地址参考本章最后一节）。运行 ESPDuino，配置其上网，这样就可以使用笔记本和 ESPDuino 在同一个局域网内进行调试。

4.1 人机界面

ESPDuino 机械臂内置了 Telnet 功能。通过 Telnet 实现人机交互，查看机械臂运行状态、当前姿态、当前设定速度等信息，同时可以设置单轴运动或者多轴运动方式，移动位置等。

ESPDuino 连接到无线路由器后，获得 IP 地址。在笔记本上使用 Telenet 工具（比如 SecureCRT）连接到该 IP。Telnet 的端口为 23，如下所示。

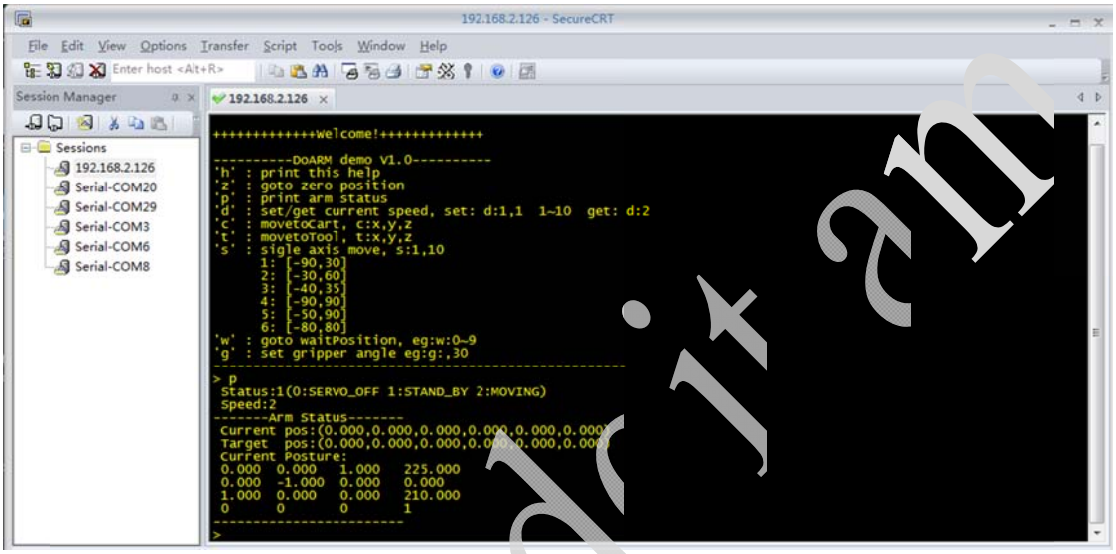


图 6 Telnet 界面

Telnet 界面下，通过输入相应的指令来控制机械臂。各指令含义如下：

“h”：打印帮助信息。

“z”：各轴回中位（即零点）。各轴回到的中位时确定的姿态是建立 D-H 参数矩阵时确定的初始姿态。回中位后，各轴当前位置、目标位置均为 0，当前姿态矩阵如下图所示。

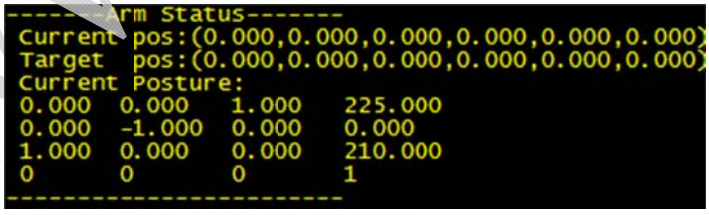


图 7 各轴回中位时的末端姿态

“p”：打印机械臂当前状态。打印的信息如上图所示，包括当前 status、设置速度、各轴当前角度、目标角度、末端当前姿态矩阵。

“d”：查询或者设置当前速度，速度的设置范围为 1~10，其中 1 表示最慢的速度，10 表示最高速度。

“c”：保持当前末端姿态不变，分别沿着笛卡尔坐标系（世界坐标系）的 X 轴、Y 轴和

Z 轴移动机械臂末端。移动的方式为增量，单位为毫米。命令格式为：“c:x,y,z”。

“t”：保持当前末端姿态不变，分别沿着末端工具坐标系的 X 轴、Y 轴和 Z 轴移动机械臂末端。移动的方式为增量，单位为毫米。命令格式为：“t:x,y,z”。

“s”：单轴移动。每个轴的初始位置为中位，此处的角度为 0。每个轴的最大移动范围为-90°~90°。移动角度的正负根据机械臂连杆坐标系建立时的右手定则确定。每个坐标系 Z 轴的旋转方向相同时为正，否则为负。根据各轴实际情况，通过实测确定各轴实际移动范围。

“w”：移动到指定的位置。位置是通过重新预先写入的。

“g”：控制抓手，参数为抓手的角度。例如 g:45 表示抓手舵机打开到 45 度。

与 Doit 智能控制三色灯项目类似，DoArm 机械臂源码提供 Web 配置和 OTA 升级功能。如下图所示。

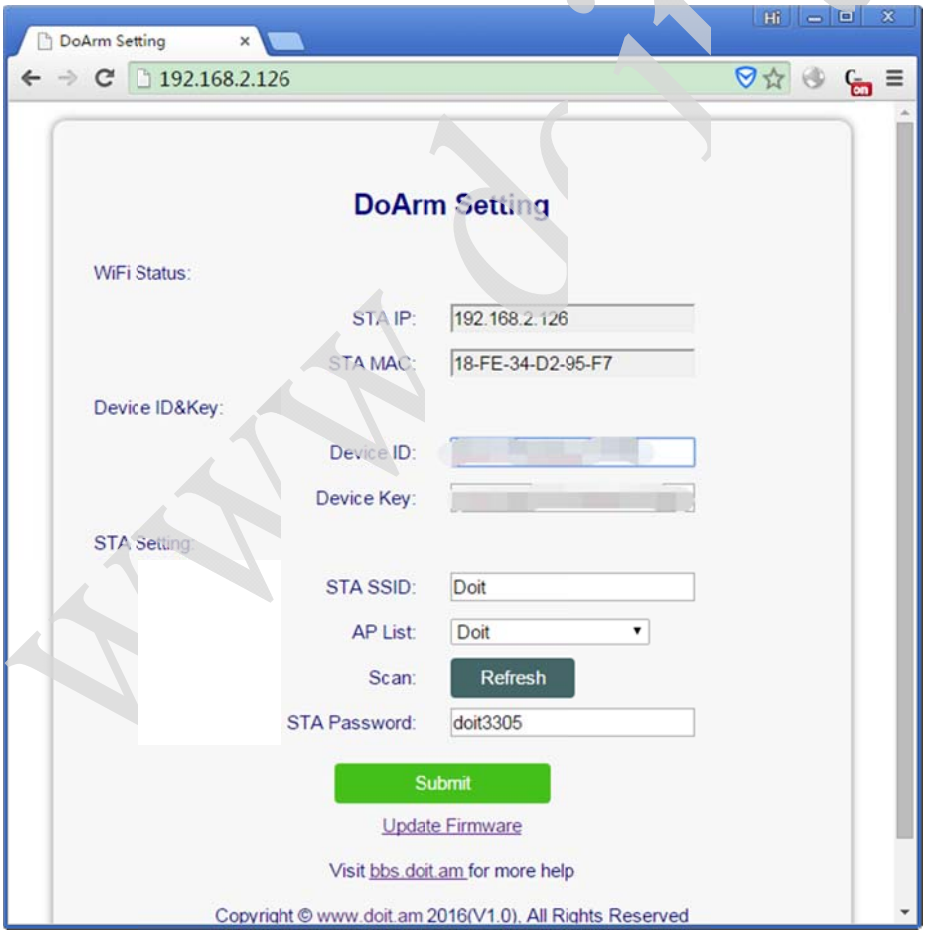


图 7 DoArm 的 Web 配置界面

4.2 标定和初始化

在完成机械臂接线后，由于机械臂各轴伺服舵机的电源信号线是随机插入到驱动板上的。首先需要将机械臂各轴与驱动板上驱动通道编号进行一一对应，然后通过程序确定各轴的中位偏移量，最后确定各轴的允许移动范围。

（1）确定机械臂各轴在驱动板上的通道编号

连接机械臂的舵机引线时，多条线捆绑在一起很难区分各轴对应的引线。因此随机将舵机电源信号线插到驱动板的舵机通道上通过程序来确定各轴对应的通道编号，上[illegible]图。

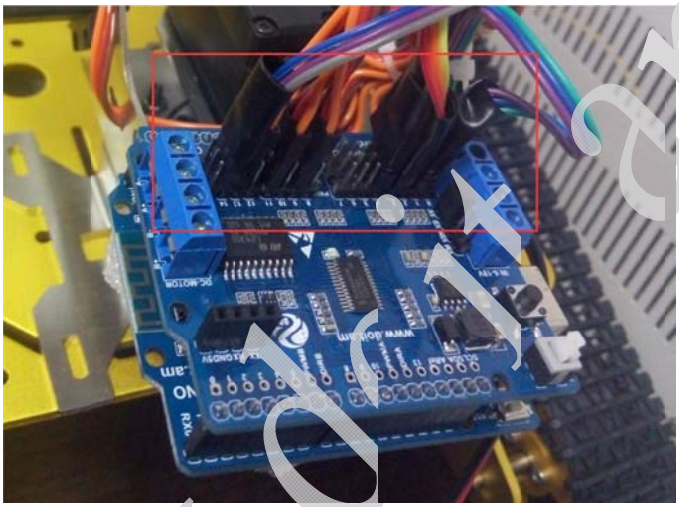


图 8 驱动板

在源代码中，将 initParseData()函数中首先将 bias[6]设置为{0,0,0,0,0,0}，各轴移动范围全部设置为-90°~90°。

```
//各轴舵机中位偏移
float bias[6] = {0, 0, 0, 0, 0, 0};
arm.setServoBias(bias, 6);
//各轴舵机移动范围
arm.setServoRange(-90, 90, -90, 90, -90, 90, -90, 90, -90, 90, -90, 90,);
```

将 arm.setChannel()设置为实际接入的通道。例如上图示例中，接入的通道分别有 1、3、4、9、11、13，则设置为：

```
arm.setChannel(1, 3, 4, 9, 11, 13);
```

编译代码并下载到 ESPduino 上，通过 Telnet 打开人机界面，使用“s:x,y”指令来确定各轴对应关系。例如：

“s:1,10”指令时，第 5 轴动。则通道“1”写入到轴 5 的位置

“s:2,10”指令时，第 6 轴动。则通道“3”写入到轴 6 的位置

“s:3,10”指令时，第 4 轴动。则通道“4”写入到轴 4 的位置

“s:4,10”指令时，第 2 轴动。则通道“9”写入到轴 2 的位置

“s:5,10”指令时，第 1 轴动。则通道“11”写入到轴 1 的位置

“s:6,10”指令时，第 3 轴动。则通道“13”写入到轴 3 的位置

最后得到正确的对应关系。

```
arm.setChannel(11, 9, 13, 4, 1, 3);
```

由于抓手是单独的，将其插入某个空闲通道即可。在程序中通过 arm. setGripperChannel() 设置。

```
//设置抓手通道号
arm.setGripperChannel(14);
```

（2）确定各轴中位偏移量

完成各轴与通道对应关系后，需要重新编译固件并下载到 ESPduino 上运行，利用 Telnet 人机界面确定各轴中位偏移量。

首先设置速度为最低值。命令：d:1,1

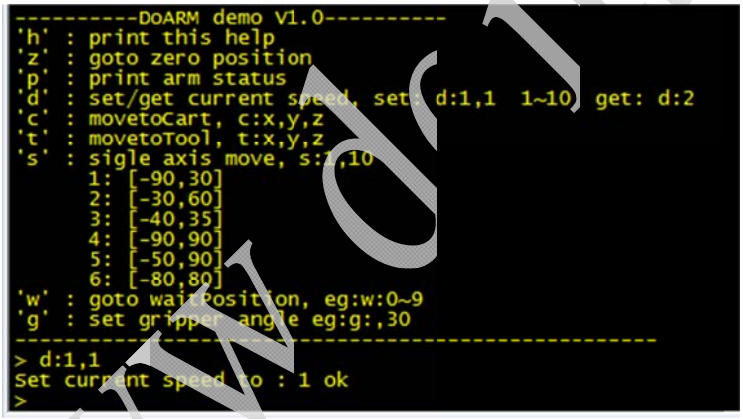


图 9 设置最低速度

然后使用“s:x,y”，单轴移动命令移动各轴，确定中位偏移。在初始状态下，中位偏移为0，此时观察各轴是否在中位，如果不在，则调整该轴使其到达中位。调整指令形如：“s:1,65”，表示第一轴要移动到 65 度才能使机械臂朝向与建模时对应的$\{O_0\}$坐标系相同。下面是本案例机械臂调整结果。

```
//各轴舵机中位偏移
float bias[6] = {65, 5, -8, 5, 0, 0};
arm.setServoBias(bias, 6);
```

将代码编译后下载运行，机械臂将会在上电后自动到中位，如下图所示。

图 10 初始状态下的机械臂

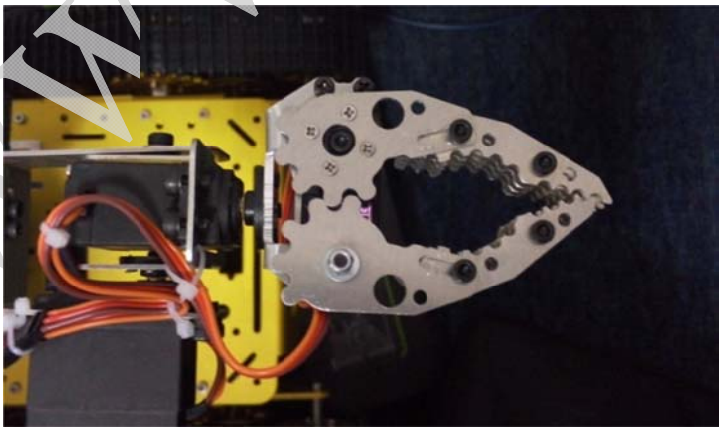
（3）确定各轴移动范围

确定各轴中位后，通过人机界面的“s:x,y”指令确定各轴移动范围。由于舵机限制，默认设置为-90°~90°。下面为本案例实际确定的各轴移动范围。在之后的单轴或者多轴控制中，若指令超过该移动范围，舵机将停止移动。

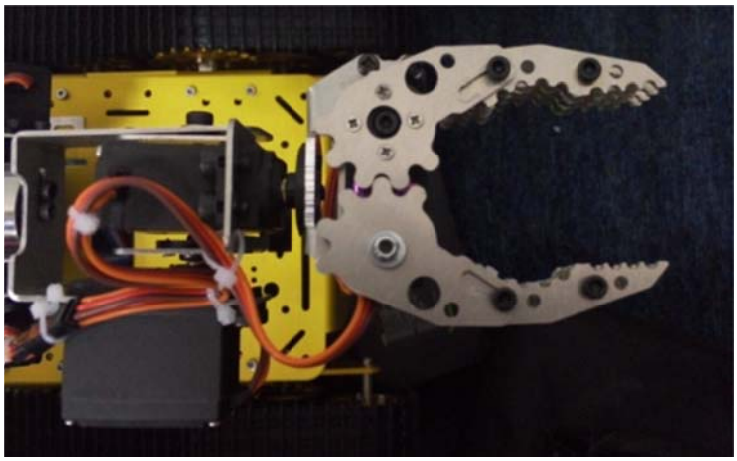
```
//各轴舵机移动范围
arm.setServoRange(-90, 30, -30, 60, -40, 35, -90, 90, -50, 90, -80, 80);
```

（4）抓手的控制

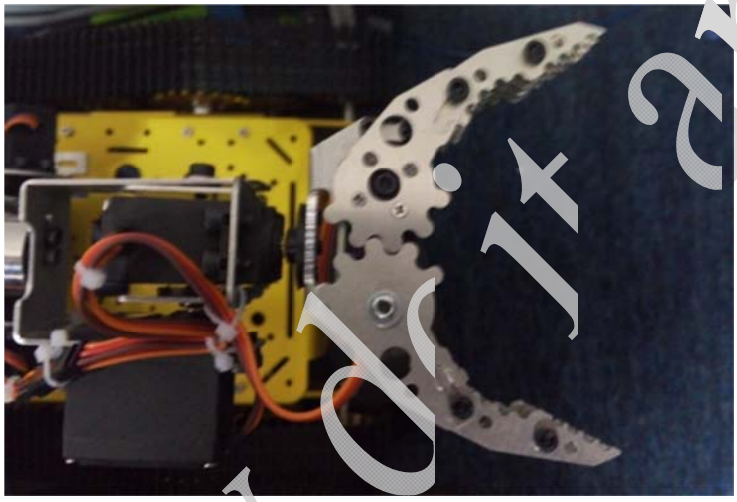
抓手是独立于机械臂六轴的，因此抓手单独控制。在人机界面上使用“g:x”指令来控制抓手。如下图所示。



（a）抓手 0°时的位置



（b）抓手 20°时的位置



（c）抓手 45°时的位置

图 11 抓手控制

4.3 单轴控制

在 Telnet 界面通过“s:x,y”指令来进行单轴控制，其中 x 表示指定运动的轴，范围为 1~6；y 表示移动的绝对角度，范围根据各轴允许移动范围确定，可以指定浮点型数据。每次运动完成后，程序会自动打印机械臂当前状态。

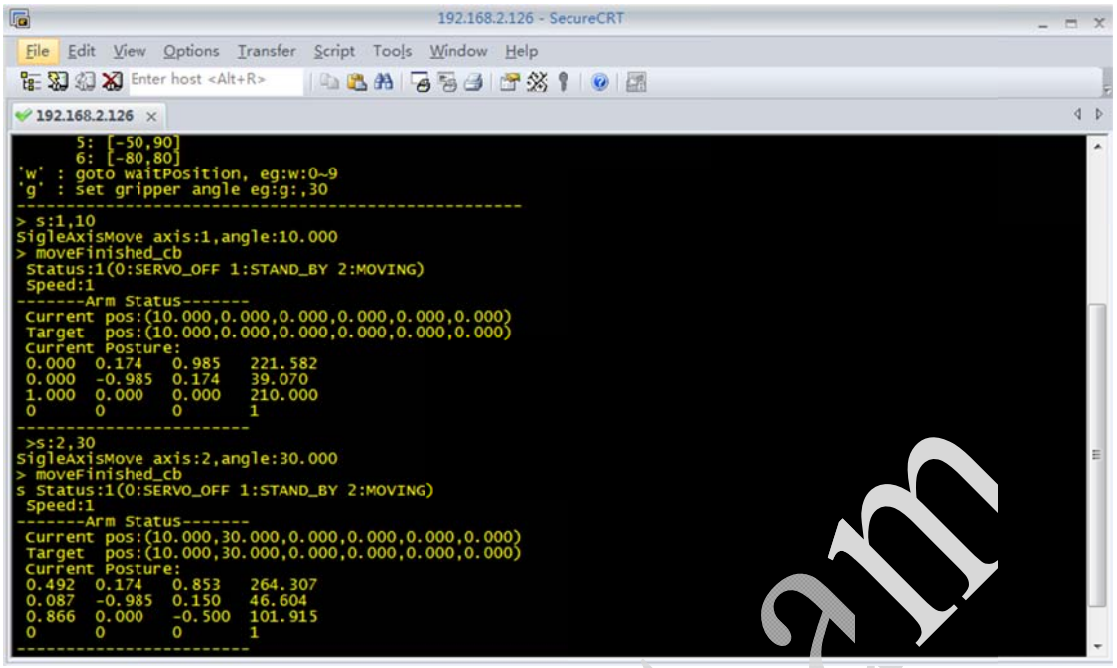


图 12 单轴运动后 Telnet 输出

4.4 联动控制

联动控制是指机械臂六个轴同时运动，实现联动。分三种模式：1，根据示教位置连续动作；2，机械臂末端在笛卡尔坐标系下移动；3，机械臂末端在工具坐标系下移动。

（1）根据示教位置连续动作

机械臂提供了 arm.setWaitPosition()和 arm.gotoWaitPosition()方法，分别用于设置示教位置，并移动到示教位置。源代码中提供了 9 个示教位置联动示例代码，可在人机界面上通过“w:x”来进行移动。其中“x”的取值为 0~8。特别的“w:9”是启动 w:0~w:8 共 9 个示教位置的连续移动。可在优酷上搜索“DoArm 机械臂”找到视频查看效果。

地址为：http://v.youku.com/v_show/id_XMTQ5ODk0NzEyMA==.html?from=y1.7-1.2

（2）机械臂末端在笛卡尔坐标系下移动

机械臂提供机械臂逆运动学算法函数 calcReverseKinematics()，计算机械臂末端在指定姿态和位置下各轴的旋转角度，实现逆运动学求解。特别的，Telnet 界面提供了机械臂在当前姿态下，沿着笛卡尔坐标系（建模时对应的{O_0}坐标系）的 X、Y、Z 轴方向移动的命令：c:x,y,z。其中 x,y,z 分别是机械臂末端沿着笛卡尔坐标系 X、Y、Z 轴移动增量，单位为毫米，浮点型数据。

例如下面的例子，将末端以初始姿态沿着笛卡尔坐标 Z 轴移动 20mm 指令为“c:0,0,20”，然后返回“c:0,0,-20”。将会观察到机械臂模块保持抓手姿态不变，竖直向上移动 20mm，之

后竖直向下移动 20mm，回到刚才的位置。在 Telnet 界面的输出如下。可以看到机械臂各轴初始位置均为 0.000。在接收到指令“c:0,0,20”后，计算得到的各轴位置分别是：（0.000,2.358,-11.361,0.000,9.003,0.000）。移动结束后，输入指令“c:0,0,-20”，计算得到各轴的位置分别是：（0.000, -0.000, 0.000, 0.000, -0.000, 0.000）。

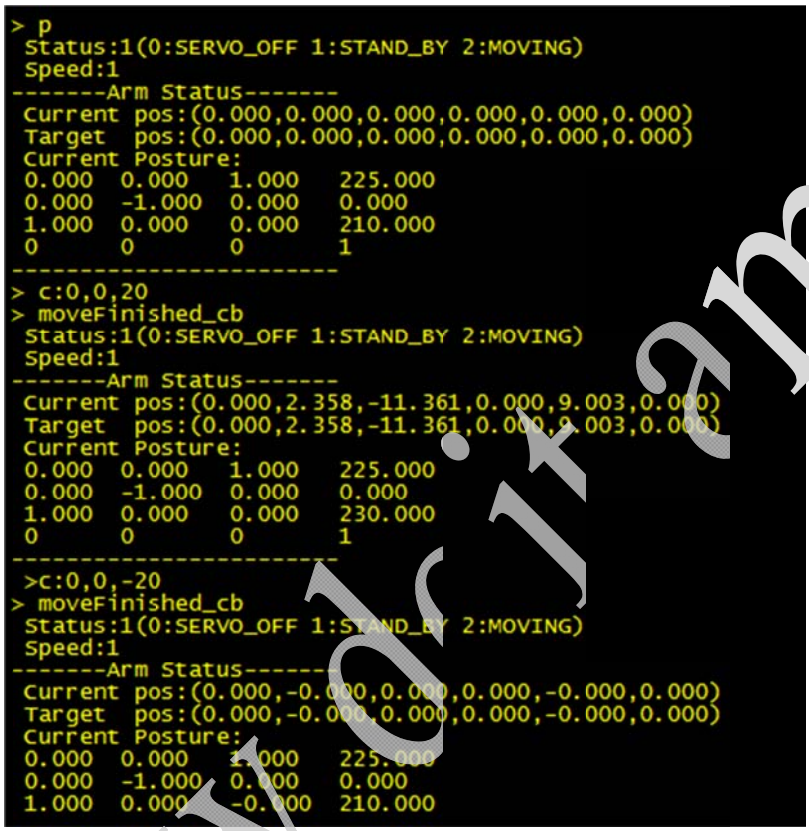


图 13 笛卡尔坐标系下运动

（3）机械臂末端在工具坐标系下移动

与机械臂末端在笛卡尔坐标系下运动类似，可以控制其末端在工具坐标系下的移动（建模时对应的{O_6}坐标系）。命令为 t:x,y,z。其中 x,y,z 分别是机械臂末端沿着工具坐标系 X、Y、Z 轴移动增量，单位为毫米，浮点型数据。

例如下面的例子，将末端以初始姿态沿着工具坐标 X 轴移动 20mm 指令为“t:20,0,0”，然后返回“t:20,0,0”。将会观察到机械臂模块保持抓手姿态不变，竖直向上移动 20mm，之后竖直向下移动 20mm，回到刚才的位置。效果与笛卡尔坐标系下运动一样，原因是坐标系{O_6}和坐标系{O_0}关于 X、Y、Z 轴定义不一样。在 Telnet 界面的输出如下。可以看到机械臂各轴初始位置均为 0.000，在接收到指令“t:20,0,0”后，计算得到的各轴位置分别是：（0.000,2.358,-11.361,0.000,9.003,0.000）。移动结束后，输入指令“t:-20,0,0”，计算得到各轴的位置分别是：（0.000, -0.000, 0.000, 0.000, -0.000, 0.000）。

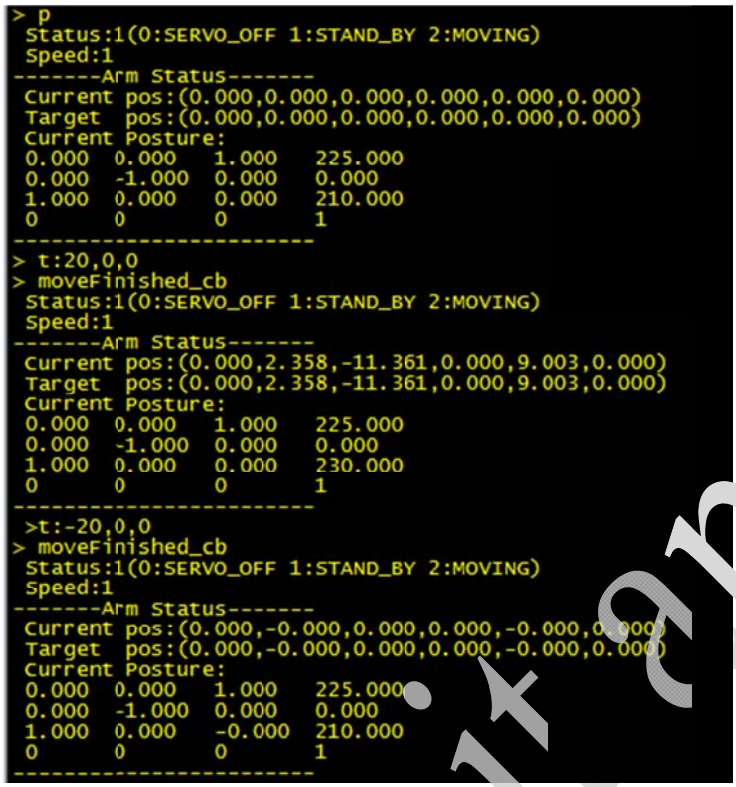


图 13 工具坐标系下运动

5 源码分析

本案例的代码已经由 Github 代码仓库托管，源码下载地址：

https://github.com/SmartArduino/DoitArm

源码文件共六个，分别是：“DoitArm_demo.ino”、“project.h”、“webPage.h”、“httpserver.ino”、“netTask.ino”、“parseData.ino”、“DoitArm.cpp”和“DoitArm.h”。

其中，“DoitArm_demo.ino”是主文件，完成初始化，配置逻辑以及主循环。

“project.h”是头文件，里面定义了 soft ap 的 ssid 和密码，版本号等。

“webPage.h”是以字符串形式存放的 webserver 的页面文件。

“httpserver.ino”中建立 http 服务器和 OTA 服务器，完成 webserver 以及 OTA 功能。

“netTask.ino”中建立 Telnet 服务器，监听 socket 链路数据并进行相应的处理。

“parseData.ino”：对机械臂进行初始化，处理 Telnet 客户端的数据，控制机械臂动作。

“DoitArm.cpp”：机械臂驱动库源码，包含驱动函数、初始化机械臂函数、获取机械臂状态函数、运动调用接口函数等。

“DoitArm.h”：机械臂驱动库头文件。

详细的内容可参考源代码，程序流程图如下。

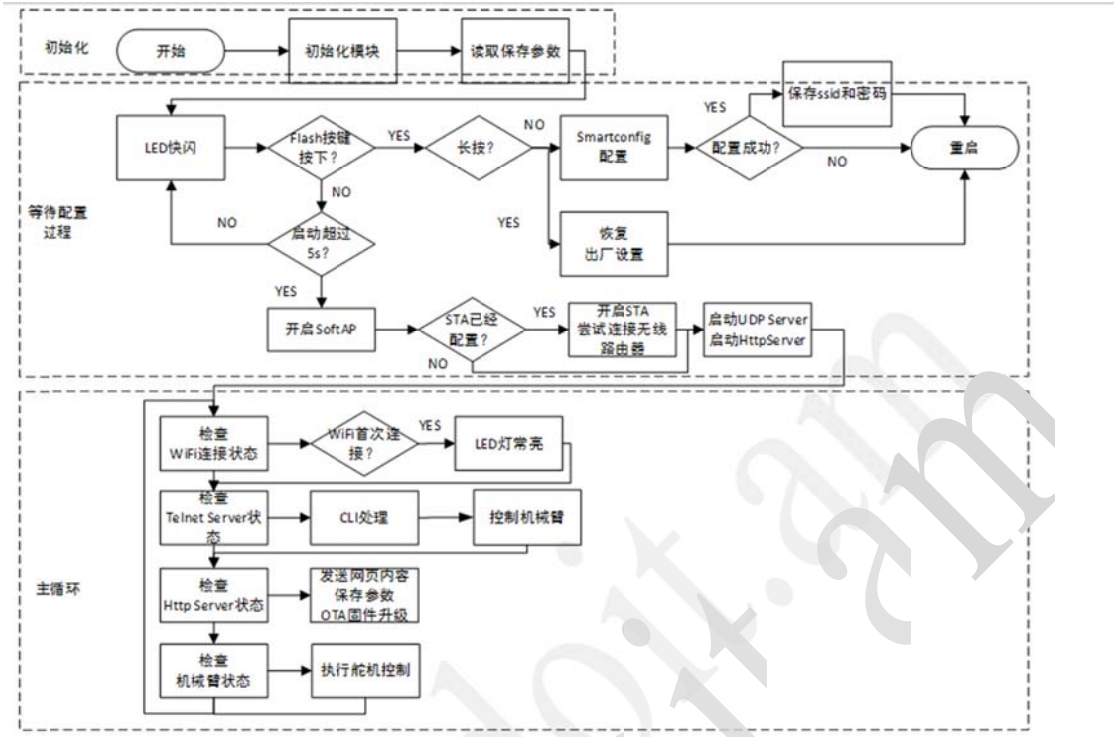


图 14 程序流程图

6. 支持与服务

购买地址：

https://item.taobao.com/item.htm?spm=686.1000925.0.0.Hbdw30&id=525385492770

技术支持 QQ 群：453053759