$7bot$ 机械臂库函数版

中文教程

Arm2.0 版本系列

张文超

机械臂设计思想

终于到了机械臂了，从舵机操作到机械臂的操作，实则机械臂的操作也就是舵机的操作，只不过思想需要转变，这时候我们将多个舵机封装到一个机械臂的整体中去了，学习机械臂的过程，简直就是 c++语言的一个缩影，从底层舵机的接口到上层机械臂的应用，我的开发思路也是按照 c++的特性仔细考虑接口的通用性设计的，希望在学习机械臂的过程中，可以带给你一些关于如何封装底层的感悟，我们的下一步是在视觉上，我也会继续给出之后的视觉处理教程，我也准备出一个关于上位机如何制作的教程，不出意外的话是用 qt 开发的（因为现在我正在用 qt 写上位机）。

谈一谈机械臂的开发过程，其实过程很简单，思路也很简单，首先阅读舵机的协议，写出舵机的寄存器版底层驱动，然后再写出一个应用层驱动，把底层的，裸露的寄存器操作严格封装好，在这个过程中要仔细考虑如何设计应用层的对外接口，这个是设计的核心，其实没有技术难点，只有封装的形式区别，充分考虑应用到机械臂上或者单独用的方便简洁性，再下一步就是对于机械臂的开发，同样，因为你这些函数是直接面向客户的，该怎么设计，思路同上。这就是机械臂的开发过程，截止我写这篇教程为止，暂时没有用到任何的算法，所以没有任何的难度，只是在调试中可能会浪费一些时间，感觉这个是必须的时间，因为程序员本质上就在做两件事，写 bug 和改 bug。

学习经验的过程总是令人愉悦的，而学习知识的过程总是枯燥乏味的。

废话不多说，进入我们的机械臂篇。

目录

第一章 机械臂质量检测

请开箱就进行机械臂质量检查，确保机械臂无质量问题。若有问题，请及时联系工作人员进行确认，以便维护您的权益。

1.1 准备工作

1.2 机械臂质量检查过程简介

1.3 与 7bot 工作人员沟通方式

1.1 准备工作

准备工作就是把机械臂的库文件添加到 arduino 的库里面，过程如舵机篇的教程一样，在这里不进行赘述。请参照舵机篇的准备工作。

1.2 舵机质量检测过程简介

舵机质量检查过程：

(1)如下图所示，找到该示例文件并。

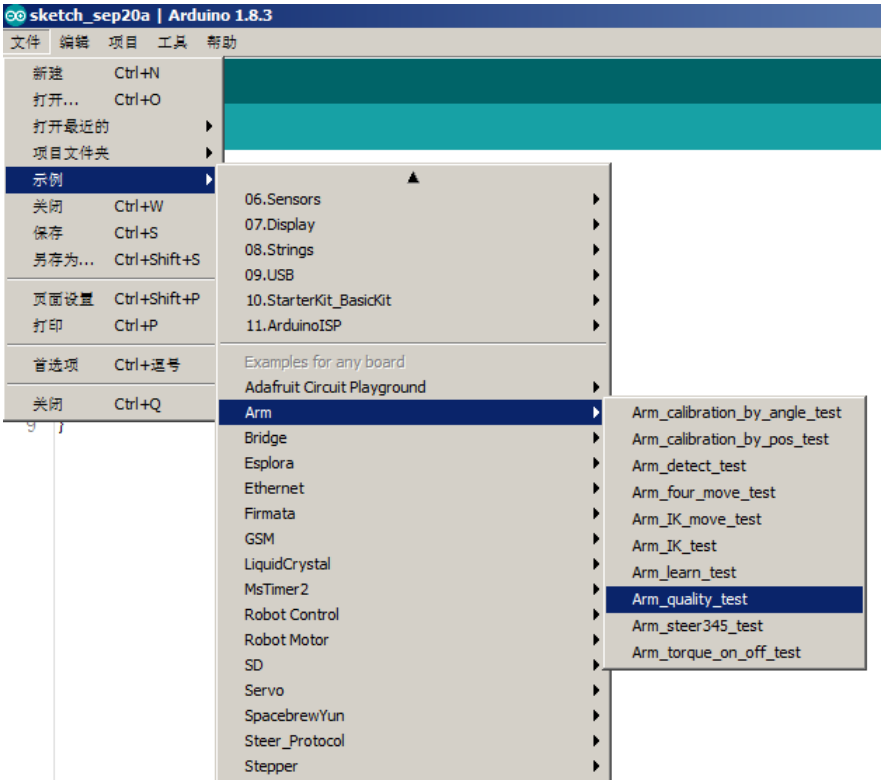

图 1.2.0

(1) 打开后进入到如下图所示的页面,然后点击下载。

注意：烧写完成后，机械臂后面的灯会闪烁。

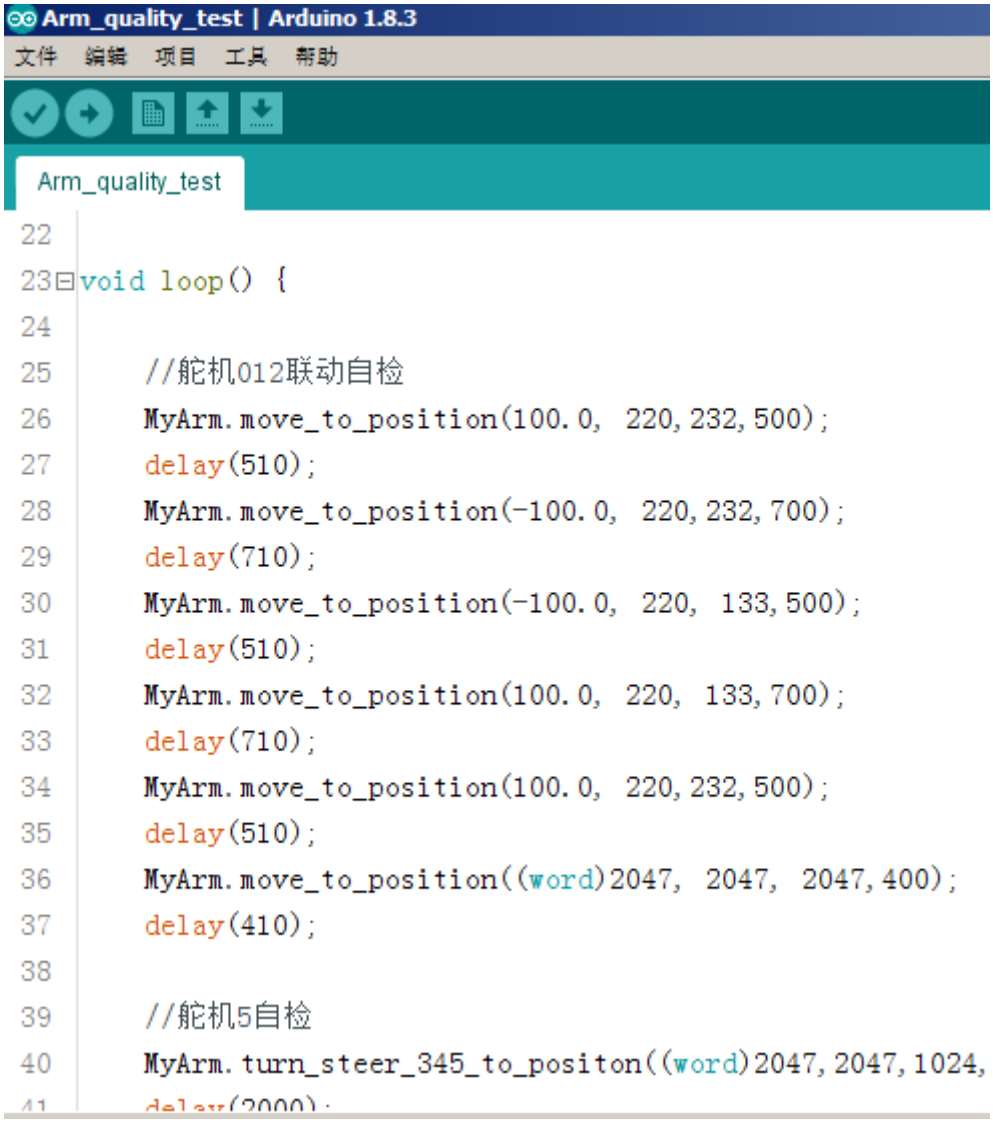

图 1.2.1

(3) 烧写完成后，会进行如下动作。

①首先，机械臂会自动进入初始化位置，各个舵机进入中位，机械臂成 90°直立，并且会等待一到三秒。

②其次，机械臂进入 ID 为 012 的三个舵机联动状态，这时候舵机末端执行画方操作。

③然后，机械臂再次进入初始化位置，这时候，最末端的舵机 5 开始自检过程：左右各旋转 90°后回到中位。

④之后，舵机 4 和 3 依次向两个方向各旋转 90°后，回到中位。

⑤重复 1 到 4 的过程

1.3 与 7bot 工作人员沟通方式

可以访问 7bot 官网，www.7bot.cc,在上面下载资料和提问，我们会尽快给您答复。当然，如果你想要得到即时恢复，请加入以下售后群：

7bot 售后 qq 群：6045566405

工作人员会尽力帮你解决开发上的问题，大家也可以在上面交流想法，学习和讨论。

第二章 机械臂舵机自检测试

机械臂舵机自检正常是机械臂能正常工作的前提，我们在每次机械臂的启动时都会进行自检。

注意：自检能够进行的前提：所有的舵机 id 必须从 0 开始，依次以 1 为单位递增才有效。

请注意与第一章区分我们这一章是进行舵机的自检，是在机械臂每次运行前都要检查舵机是否可以正常通信，并根据舵机应答数量的多少，在程序中会自动完成参数配置。

2.1 机械臂舵机自检过程简介

2.2 机械臂舵机自检函数介绍

2.3 Arduino 上编写程序

2.4 下载验证

2.1 机械臂舵机自检过程简介

机械臂自检，就是测试机械臂中的舵机是否都应答正常，本质上就是挨个询问舵机，看看到底有几个舵机进行了应答，以此来完成一些配置。

2.2 机械臂舵机自检函数简介

我们先进行《7bot 机械臂固件函数库中文手册》中机械臂的公有函数中去寻找，当当当，我仔细看了好几遍，啥，没有，又仔细看了看，确实没有。

然后急了，我去找到源文件 Arm.h 和 Arm.cpp 中，发现了如下图所示，这个函数确实有，但是它是私有函数，我们无法调用，但同时，我又发现，在公有函数的 begin 里面，也就是机械臂初始化里面有。

哈哈哈，基于此，我可以先调用 begin 初始化，再去打印公有变量里的 Steer_Num 变量，我就可以看到舵机自检是否正常，连我都佩服自己的机智。

```
22  class Arm{
23
24  private:
25      HardwareSerial *comSer;
26
27      int *pos_goal;
28
29      byte Steer_Detect();
30      void Para_Init();
31
```

图 2.2.0

2.3 Arduino 上编写程序

按照如上思路，我们写出如下的程序。

```
14  #include<Arm.h>
15
16  void setup() {
17
18    //机械臂初始化(这其中进行机械臂舵机自检)
19    MyArm.begin(USB_SER);
20
21    Serial.println("Arm_detect_test");
22    //机械臂位置初始化
23    MyArm.position_init();
24    delay(2000);
25  }
26
27  void loop() {
28    //每隔一秒打印一次机械臂自检过程中得到的舵机数量
29    Serial.println(MyArm.Steer_Num);
30    delay(1000);
31  }
```

图 2.3.0

可以看到程序就是之前思路的实现，接下来就进入到下载验证，请往下看。

2.4 下载验证

如下图所示找到该程序并打开：

图 1.4.0

下载程序后打开串口显示器，我们可以看到如下结果：

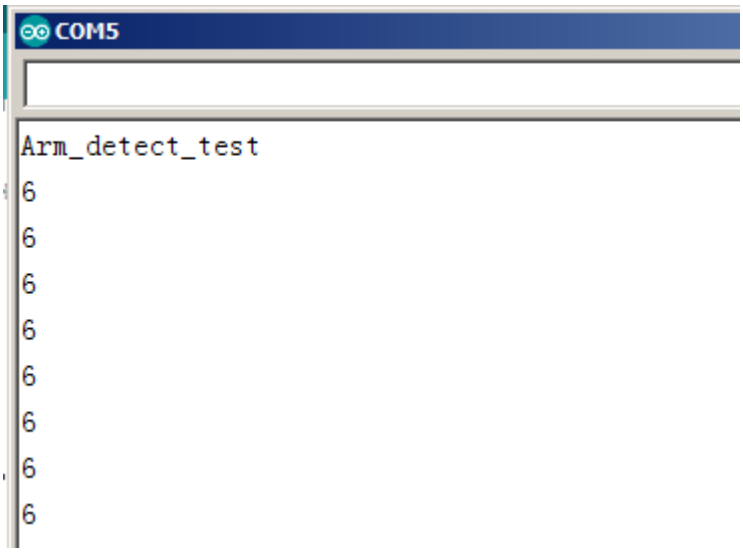

图 1.4.1

我数了数，我现在用的机械臂舵机数量正好是 6 个，验证通过。

第三章 机械臂偏置设置(pos)

机械臂初始化位置可能不是正好在中位，而是可能有一点偏差，这时候就需要我们手动进行设置偏置，使机械臂各个舵机到达中位。

那啥是中位呢？

中位就是机械臂的大臂和小臂处于垂直状态，各个舵机你可以观察到不是很正，这时候你只需要输入偏置调正即可（这个正你看到了就理解了）。

注意：标题上的 pos，强调，pos 是啥？在机械臂和舵机的很多地方都会出现这个参数，这个参数的含义，就是舵机的真正的磁编码值，也就是我们实际上写到舵机寄存器里的值就是它，它的值范围我们给出（0~4095），换成角度范围为（0~360°）。

3.1 机械臂偏置设置过程简介

3.2 机械臂偏置函数介绍

3.3 Arduino 上编写程序

3.4 下载验证

3.1 机械臂偏置过程简介

机械臂的偏置设置，其实很简单，我们的思路是这样的，设置一次偏置，以后一直可以用，所以，我们得把这个值存放在一个 ROM 里，事实上，我们也是这么做的，每次你给的偏置，我们都放在了 arduino 的 eeprom 里，这样就防止了数据丢失。然后你每次设置的时候，我们都会更新该值。另外，我们机械臂每次在动作的时候，我们都会加上该偏置（包括位置初始化）。

3.2 机械臂偏置函数介绍

我们查阅《7bot 机械臂固件函数库中文手册》，查阅 Arm 类中的公有函数列表如下图所示，在其中我们找到了如红色圈出的两个函数。

1.1 Arm 类中的公有函数列表

Table2. 给出了 Arm 的公有函数列表

函数名	描述
begin	机械臂配
position_init	机械臂位
inverse_movement	机械臂坐
move_to_position	机械臂运
Para_Init	机械臂参
Set_Arm_Torque_On	机械臂扭
Set_Arm_Torque_Off	机械臂扭
turn_steer_345_to_positon	第 3，第
Get_Offset	得到机械
offset_by_pos	通过机械
offset_by_angle	通过机械
Rad2Angle	弧度值转

图 3.2.0

今天我们主要介绍第一个函数，通过直接数据去设置偏置，我们到目录找到该函数，点下去找到它的详细介绍。如下图所示：

1.2.7 函数 offset_by_pos

Table10. 描述了函数 offset_by_pos

Table10.

函数名	offset_by_pos
函数原型	void offset_by_pos(byte id, short offset);
功能描述	通过机械臂的直接位置，设置机械臂偏置
输入参数 1	Id: 舵机的 ID 号
输入参数 2	Offset: 设置舵机的偏置，值的范围（-2046～+2046）
返回值	无
先决条件	无
被调用函数	无

例:

```
/**初始化通信串口为 USB_SER，并且给舵机 1 的偏置设为 200**/
MyArm.begin(USB_SER);
MyArm. offset_by_pos(1,200);
```

图 3.2.1

上图给出的很明白，我在这里再用通俗的话说一遍。

1）首先他给出了函数的原型，我们可以看到，该原型是有两个输入参数，没有返回值的。

2）再往下看，我们看到了两个输入参数的解释，第一个 id 是舵机的 ID 号，就是我们在设置机械臂的偏置时，实际上在设置舵机的偏置。

3）第二个参数是输入的偏置的值的范围。

4）再往下，就是他的一个使用例程。先初始化，再直接给出偏置。

3.3 Arduino 上编写程序

在 arduino 上编写，为了方便随时设置各个偏置，我们采用的方法是串口输入调试，请看代码：

```
#include<Arm.h>

void setup() {
  MyArm.begin(USB_SER);
  Serial.println("Arm_Calibration_By_Pos_Test");
  Serial.println("Please input in the following format
  Serial.println("id(byte)  offset(short)");
  Serial.println("such as : 5 100");
  MyArm.position_init();    //位置初始化
  delay(2200);              //等待位置初始化完成
}

void loop() {
  while(Serial.available())
  {
    byte id = Serial.parseInt();       //从串口得到ID
    short offset = Serial.parseInt(); //从串口得到偏置
    MyArm.offset_by_pos(id, offset);  //设置偏置到ROM
    MyArm.Get_Offset();               //从ROM得到偏置
    Serial.println("get offset");
    for(int i = 0; i < MyArm.Steer_Num; i++)
    {
      Serial.print("id = "); Serial.print(i); //打印从ROM得到的偏置
      Serial.print("  Offset = "); Serial.println(MyArm.offPos[i]);
    }
    MyArm.position_init();            //更新偏置就是更新初始化位置
  }
}
```

图 3.3.0

没有什么难点，有的话可能是本身对于 arduino 的函数不是很熟，关于我们应用这一块还是很简单的，有什么需求可以直接联系我们，这里不再赘述。直接进入到下载验证。

3.4 下载验证

如下图所示打开例程：

图 3.4.0

然后直接下载，下载完成后打开串口，如下图所示：

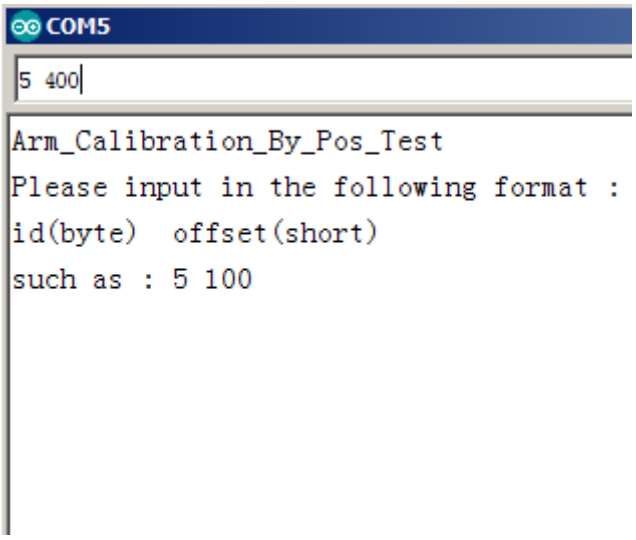

图 3.4.1

根据提示的格式输入，建议尽量用舵机 5 做实验，因为舵机 5 可以全方位旋转没有遮挡，不容易损坏。

注意：第二个参数不要输入过大，刚开始可以一点一点测试。值范围（ -2046~+2046 ）

点击 enter 后返回如下所示的值：

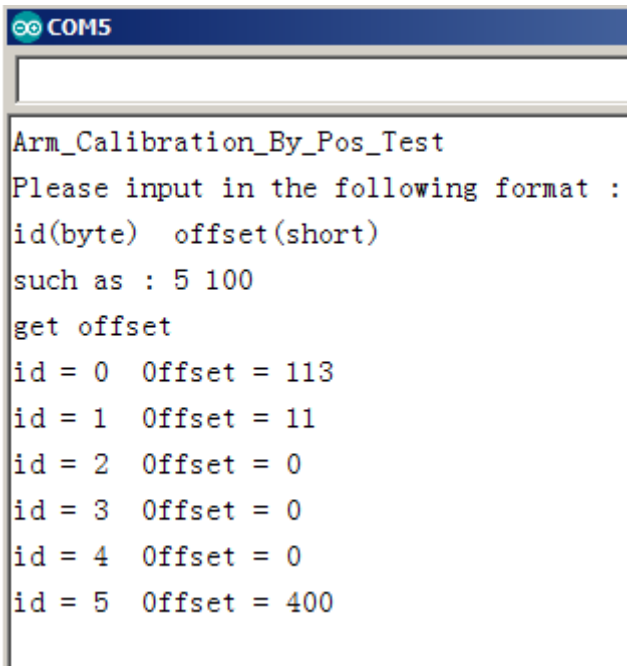

图 3.4.2

看到舵机 5 的偏置被改变，同时看到舵机 5 旋转。

到这里，关于舵机通过直接位置数据设置偏置的过程我们就学会了。同时，你可以去查阅通过角度去设置的方法，我们同样给出了例程，就是第二个输入参数变成了 double 型，范围也变成了（0~360°）。

请去尝试，以便你熟悉查阅文档的过程。

第四章　机械臂姿态逆解测试

机械臂姿态逆解就是运动学中的运动逆解。

4.1　机械臂姿态逆解过程简介

4.2　机械臂姿态逆解函数介绍

4.3　Arduino 上编写程序

4.4　下载验证

4.1 机械臂姿态逆解过程简介

机械臂姿态逆解，说的简单点，就是你给出机械臂末端的位置，机械臂通过反向运算，得出机械臂各个舵机的角度位置。

你观察下机械臂末端的位置与什哪些有关？很巧妙，只与前三个舵机的状态有关，所以对于机械臂逆解，最终出来的只是舵机 012 三个舵机的角度值。

4.2 机械臂姿态逆解函数介绍

我们查阅《7bot 机械臂固件函数库中文手册》，查阅 Arm 类中的公有函数列表如下图所示，在其中我们找到了如红色圈出的函数。

1.1 Arm 类中的公有函数列表

Table2. 给出了 Arm 的公有函数列表

函数名	描述
begin	机械臂配置和通信初始化
position_init	机械臂位置初始化
inverse_movement	机械臂坐标位置逆解函数
move_to_position	机械臂运动控制函数
Para_Init	机械臂参数初始化函数
Set_Arm_Torque_On	机械臂扭矩开启函数
Set_Arm_Torque_Off	机械臂扭矩关闭函数
turn_steer_345_to_positon	第 3，第 4，第 5 号舵机旋
Get_Offset	得到机械臂的偏置函数
offset_by_pos	通过机械臂的直接数据设
offset_by_angle	通过机械臂的角度设置机
Rad2Angle	弧度值转角度值函数
Pos2Rad	直接位置数据转换弧度值

图 4.2.0

我们到目录找到该函数，点下去找到它的详细介绍。如下图所示：

1.2.2 函数 inverse_movement（重载 1）

注意：inverse_movement 有 2 个重载，这是第 1 个

Table5. 描述了函数 inverse_movement

Table5.

函数名	inverse_movement
函数原型	void inverse_movement(double x_ , double y_, double z_)
功能描述	机械臂坐标位置逆解函数（由末端坐标逆解出前三个舵机的角度值）
输入参数 1	x_ ：机械臂末端的 x_坐标。
输入参数 2	Y_ ：机械臂末端的 y_坐标
输入参数 3	Z_ ：机械臂末端的 z_坐标
返回值	无
先决条件	无
被调用函数	atan; acos;

例:

```
/**初始化通信串口为 USB_SER，并且求解(120,120, 120)的角度值**/
MyArm.begin(USB_SER);
MyArm. inverse_movement (120,120, 120);
Serial.print(....)://略
```

图 4.2.1

上图给出的很明白，我在这里再用通俗的话说一遍。

就是你直接输入三个 double 型的坐标位置，就可以逆解出三个弧度，这三个弧度在哪呢？在公有变量里，我们去看一看。

Table0. 给出了 Arm 的公有变量列表

公有变量名(public)	描述
Steer_Num	机械臂中现有的舵机数量
offPos	机械臂的各个舵机偏差，得到舵机数量后，利用动态数组确定其大小
theta	机械臂的各个舵机弧度值，得到舵机数量后，利用动态数组确定其数组大小
steer	机械臂的各个舵机对象，得到舵机数量后，利用动态数组确定其数组大小

图 4.2.2

以手册为准，得到的是弧度值，对，计算的结果应该是弧度。存放在 theta 数组里。实验：确实是弧度值。

4.3 Arduino 上编写程序

```
#include<Arm.h>

Serial_arm sa; //初始化一个串口接收对象

void setup() {
  MyArm.begin(USB_SER);
  Serial.println("Inverse_Movement_Test");
  Serial.println("Please enter three double data:");
}

void loop() {

  while(Serial.available())
  {
    double x = sa.parsedouble(&Serial);//接收一个double型数据
    double y = sa.parsedouble(&Serial);
    double z = sa.parsedouble(&Serial);

    Serial.println("get xyz");
    MyArm.inverse_movement(x,y,z);//姿态逆解运算
    Serial.print("x = "); Serial.print(x);
    Serial.print("  y = "); Serial.print(y);
    Serial.print("  z = "); Serial.println(z);
    for(int i = 0; i < 3; i++)
    {
      //输出逆解运算结果(注意，在这里我们将角度值转化为了弧度值)
      Serial.print(MyArm.Rad2Angle(MyArm.theta[i]));
      Serial.print("  ");
    }
    Serial.println();
  }
}
```

图 4.3.0

注释的比较清楚，没什么难点，有什么需求可以直接联系我们，这里不再赘述。直接进入到下载验证。

4.4 下载验证

如下图所示打开例程：

图 4.4.0

然后直接下载，下载完成后打开串口，如下图所示：

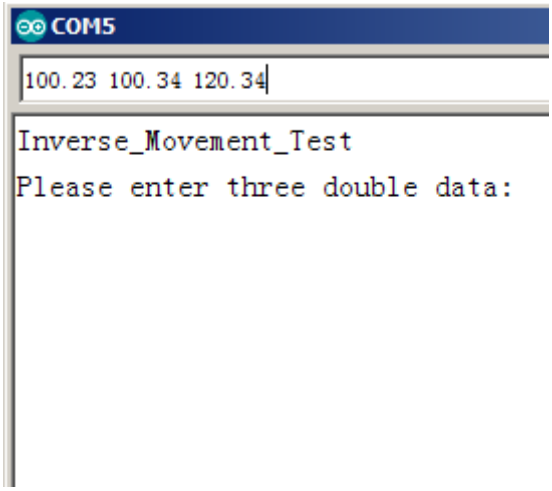

图 4.4.1

根据提示，输入三个合适的坐标值，double 类型。

点击 enter 后返回如下所示的值：

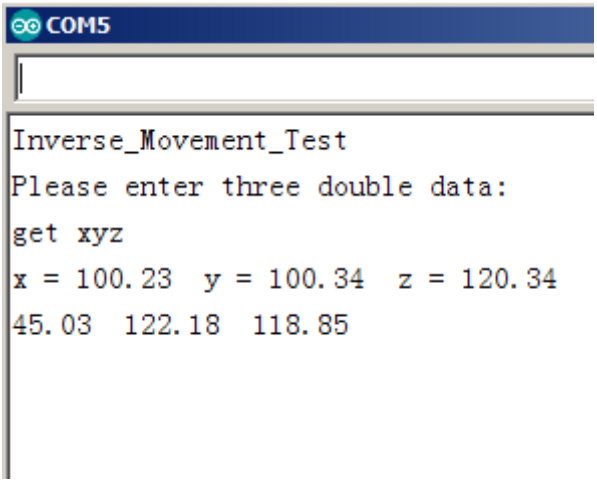

图 4.4.2

返回了三个值，分别是舵机 012 的三个角度值。