

7bot robot arm library function version

Arm2.0 version series

Robot arm design idea

Finally, the robot arm, from the servo operation to the mechanical arm operation, in fact, the mechanical arm operation is the servo operation, but the thought needs to change. At this time, we encapsulate multiple servo into the whole of one mechanical arm. The process of learning a robot arm is simply a microcosm of c language. From the interface of the bottom servo to the application of the upper robot arm, my development idea is also carefully considered the universal design of the interface according to the characteristics of c. Hopefully, in the process of learning the robotic arm, we can give you some insight into how to encapsulate the underlying layer. Our next step is to visually, and I will continue to give it. After the visual processing tutorial, I also prepared a tutorial on how to make the upper computer, not surprisingly, is developed in QT (because I am now using QT to write PC).

In fact, the process is very simple, and the train of thought is very simple. First, read the protocol of the servo, write out the bottom driver of the servo register version, and then write an application layer driver. The bare register operation is tightly encapsulated, and careful consideration should be given to how to design the application in the process

Layer external interface, this is the core of the design, in fact, there are no technical difficulties, only the form of packaging difference, full consideration of the application of mechanical arm or single use of the convenience of simplicity, and then the next step is to the mechanical arm

The development of the same, because your functions are directly customer-oriented, how to design, the same idea.

This is the robot arm development process, as of this tutorial, I have not used any algorithm, so there is no difficulty, just may waste some time in debugging, feel this is necessary

Time, because programmers are essentially doing two things, writing bug and changing bugs. com.

The process of learning experience is always pleasant, and the process of learning knowledge is always boring. Blah, blah.

Chapter 1 quality testing for robot arm

1.1 Preparations

The preparation is to add the robotic arm library file to the arduino library, the process is like the servo tutorial, here do not go into detail. Please refer to the preparation of the servo.

1.2 Brief introduction of quality testing process of servo

servo quality inspection process:

(1) as shown below, locate the sample file :

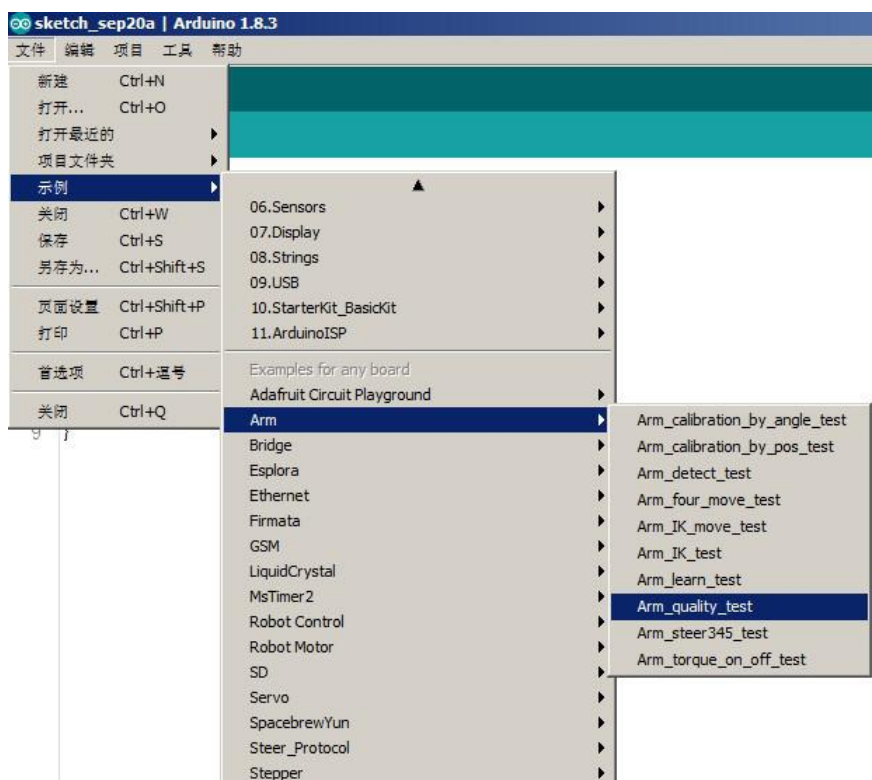


图 1.2.0

Open and go to the page shown below, then click download. Note: after writing is finished, the light behind the arm flashes.

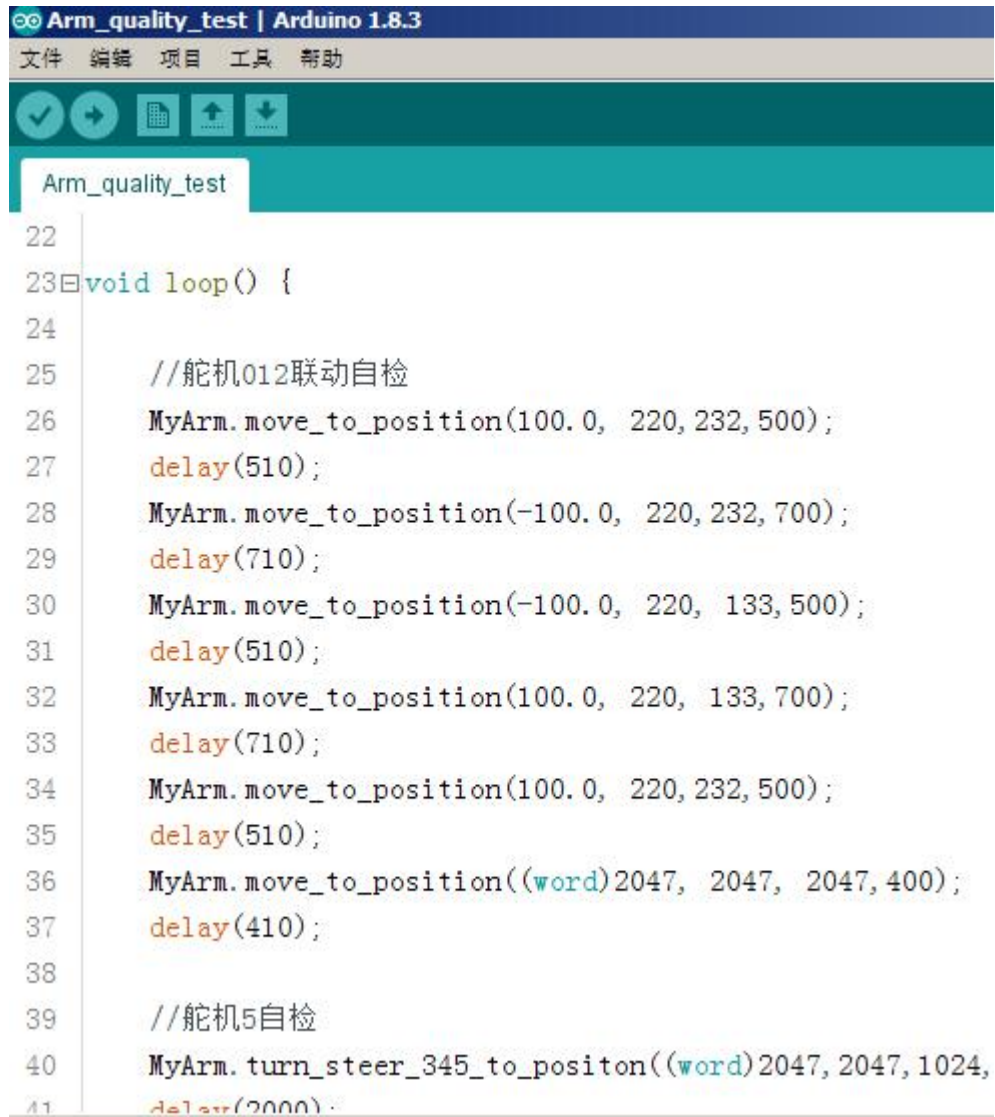


图 1.2.1

(3) When the burn is complete, the following actions will be performed.

- ① First, the arm automatically enters the initialization position, each servo enters the median position, and the arm is 90 °upright and waits for one to three seconds.
- ② Secondly, the robot arm enters the linkage state of three servo with ID 012, when the terminal of the servo performs the drawing operation.
- ③ Then, the robot arm enters the initialization position again, at this time, the terminal servo 5 begins the self-checking process: the left and right rotation is 90 ° and then returns to the median position.
- ④ After that, the servo 4 and 3 rotate 90 ° in each direction and return to the median position.
- ⑤ Repeat the process of 1 to 4

Chapter 2 Self-test of robot arm servo

The self-inspection of the robot arm servo is the prerequisite for the robot arm to work properly. We conduct self-inspection every time the arm starts.

Note: self-testing can be performed on the premise: all servo id must start from 0, in turn incremented by 1 units to be effective.

Please note that the first chapter of our chapter is to carry out self-inspection of the servo, which is to check that the servo can communicate normally before each operation of the robot arm, and to automatically complete the parameter configuration in the program according to the number of rudder responses.

- 2.1 A brief introduction to the self-testing process of the robot arm servo
- 2.2 Introduction of Self-Test function of robot arm servo
- 2.3 Programming on Arduino
- 2.4 Download verification

2.1 A brief introduction to the self-testing process of the robot arm servo

The robot arm self-test is to test whether the servo in the arm is responding normally, essentially asking the servo one by one to see how many servo have been answered to complete some configuration.

2.2 A brief introduction to the Self-Test function of the robot arm servo

Let's start by looking for the common functions of the robot arm in the firmware library of the 7bot robot arm. When I looked at it several times, no, no.

Then I went to find the source Arm.h and Arm.cpp, and I found that, as shown in the figure below, this function does exist, but it's a private function that we can't call, but at the same time, I also find that in the begin of the public function, That is, the mechanical arm initialization has.

Ha, based on this, I can first call begin initialization, then print the Steer_Num variable in the public variable, and I can see if the servo self-check is normal.

```
22  class Arm{
23
24  private:
25      HardwareSerial *comSer;
26
27      int *pos_goal;
28
29      byte Steer_Detect0;
30      void Para_Init0;
31
```

图 2.2.0

2.3 Programming on Arduino

According to the above thinking, we write the following procedure.

```
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

You can see that the program is the implementation of the previous ideas, and then into the download verification, please see below.

2.4 Download verification

Locate the program and open it as shown in the following figure:



图 1.4.0

After you download the program and open the serial port display, you can see the following results:

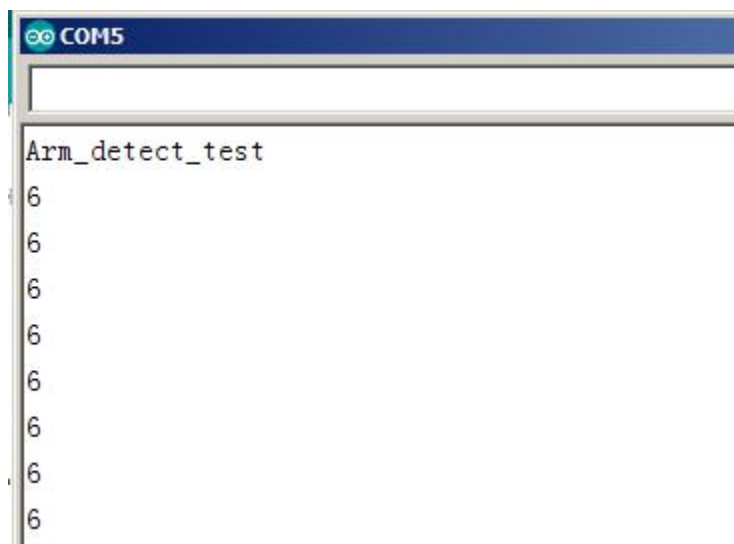


图 1.4.1

I counted, and the number of servos I'm using now is exactly six.

Chapter III offset arrangement of robot arm(pos)

The initializing position of the arm may not be in the middle position, but there may be a slight deviation, which requires us to set the offset manually to get the arm servo to the median position.

So what's the middle??

The median is that the arm of the robot arm is perpendicular to the forearm, and you can see that the servo is not very positive, and you just have to type in the offset alignment.

Note: poss on the title, what is it? This parameter appears in many parts of the robot arm and the servo, and what this parameter means is the true magnetic coding value of the servo, which is the value that we actually wrote in the servo register. The range of values we give is (0n4095), which is changed to an angle range (0-360 °).

- 3.1 Brief introduction to the process of setting offset of robot arm
- 3.2 Introduction to offset function of robot arm
- 3.3 Programming on Arduino
- 3.4 Download verification

3.1 A brief introduction to the biasing process of the robot arm

The biasing of the robot arm is actually very simple, and the idea is to set up a bias that can be used all the time, so we have to store this value in a ROM, and in fact, that's what we do. Every time you give us a bias, we put it in arduino's eeprom, which prevents data loss. Then we update the value every time you set it. In addition, each time we move the arm, we add the offset (including position initialization).

3.2 Introduction to offset function of robot arm

We refer to the Chinese manual of the 7bot robot arm firmware function library, and look at the list of public functions in the Arm class as shown in the figure below, where we find two functions such as the red circle.

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	机械臂生
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

Today we mainly introduce the first function, through the direct data to set the bias, we go to the directory to find the function, click to find it in detail. As shown below:

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

It's clear from the image above that I'm going to say it again here in plain words.

- 1) First of all, he gives the prototype of the function, and we can see that the prototype has two input parameters and no return value.
- 2) Looking next, we see the explanation of two input parameters. The first id is the ID number of the servo, that is, when we set the offset of the robot arm, we actually set the offset of the servo.
- 3) The second parameter is the range of the values of the input bias.
- 4) 再Down, is one of his usage routines. Initialize first, then give bias directly.

3.3 Programming on Arduino

In arduino, in order to set up each bias at any time, we use the method of serial port input debugging, please look at the code:

```

14 #include<Arm.h>
15
16 void setup() {
17     MyArm.begin(USB_SER);
18     Serial.println("Arm_Calibration_By_Pos_Test");
19     Serial.println("Please input in the following format");
20     Serial.println("id(byte)  offset(short)");
21     Serial.println("such as : 5 100");
22     MyArm.position_init();    //位置初始化
23     delay(2200);             //等待位置初始化完成
24 }
25
26 void loop() {
27     while(Serial.available())
28     {
29         byte id = Serial.parseInt();    //从串口得到ID
30         short offset = Serial.parseInt(); //从串口得到偏置
31         MyArm.offset_by_pos(id, offset); //设置偏置到ROM
32         MyArm.Get_Offset();             //从ROM得到偏置
33         Serial.println("get offset");
34         for(int i = 0; i < MyArm.Steer_Num; i++)
35         {
36             Serial.print("id = "); Serial.print(i); //打印从ROM得到的偏置
37             Serial.print("  Offset = "); Serial.println(MyArm.offPos[i]);
38         }
39         MyArm.position_init();           //更新偏置就是更新初始化位置
40     }
41 }

```

图 3.3.0

There is no difficulty, some may not be very familiar with the function of arduino, about our application of this piece is very simple, what requirements can be directly linked to us, we do not repeat here. Go directly to download validation.

3.4 Download verification

Open the routine as shown in the figure below:



图 3.4.0

Then download directly, and when the download is complete, open the serial port, as shown in the following figure:

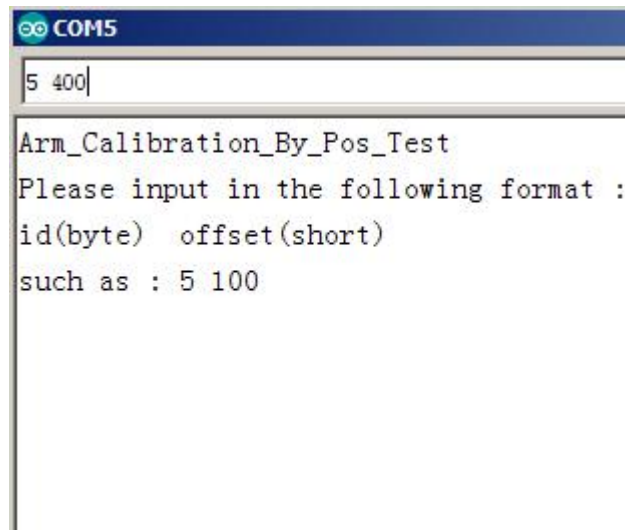
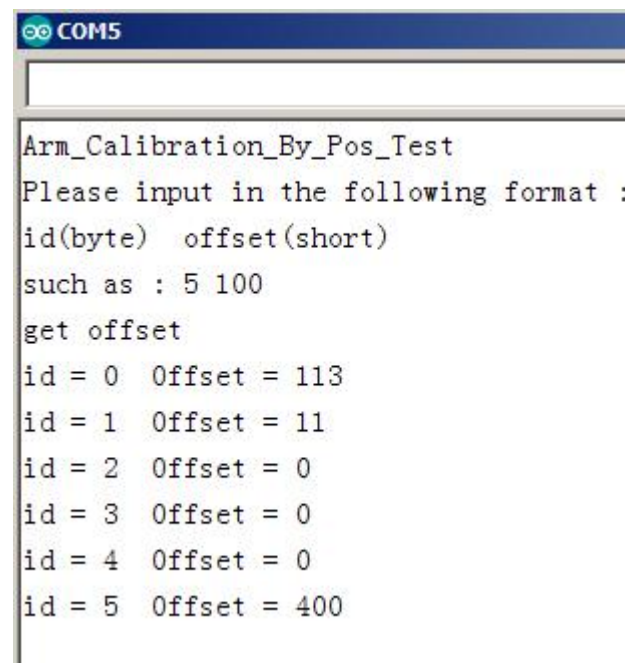


图 3.4.1

According to the format input of the prompt, it is recommended to use the servo 5 as far as possible to do the experiment, because the servo 5 can rotate in all directions without blocking, so it is not easy to damage.

Note: the second parameter should not be too large and can be tested a little at first. Value range (-2046 ~ 2046) and click enter to return the value shown below:

图 3.4.2



See that the offset of the servo 5 is changed and that the servo 5 rotates at the same time.

Here, we learn about the servo biasing through direct position data. At the same time, you can look at the way you set it by angle, and we also show the routine, where the second input parameter becomes

Double type, the range has also changed ($0 \sim 360^\circ$).

Please try so that you can familiarize yourself with the process of consulting the document。

Chapter 4 determination of attitude inverse solution of robot arm

The inverse solution of robot arm attitude is the inverse solution of Motion in Kinematics。

- 4.1 A brief introduction to the inverse solution process of robot arm attitude
- 4.2 An introduction to the inverse solution function of robot arm attitude
- 4.3 Programming on Arduino
- 4.4 Download verification

4.1 A brief introduction to the inverse solution process of robot arm attitude

The simple point is that you give the position of the end of the robot arm, and the robot arm works out the angular position of each servo by reverse operation.

What do you think about the position of the end of the robot arm? It is very ingenious, only related to the state of the first three steering machines, so the inverse solution to the robot arm is only the angle value of the steering engine 012 and three rudders.

4.2 An introduction to the inverse solution function of robot arm attitude

We refer to the Chinese manual of the 7bot robot arm firmware library and the list of public functions in the Arm class as shown in the figure below, where we find functions such as the red circle.

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

We go to the directory to find the function, and click to find the details of it. As shown below:

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

The picture above shows me clearly, and I'll say it again here in plain words.
That is, if you directly enter the three double coordinates, you can invert the three radians. Where are the three radians? In the public variable, let's take a look.

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

According to the manual, the radians are obtained. Yes, the results should be radians. Stored in the theta array. Experiment: it's really radians.

4.3 Programming on Arduino

```

14 #include<Arm.h>
15
16 Serial_arm sa; //初始化一个串口接收对象
17
18 void setup() {
19     MyArm.begin(USB_SER);
20     Serial.println("Inverse_Movement_Test");
21     Serial.println("Please enter three double data:");
22 }
23
24 void loop() {
25
26     while(Serial.available())
27     {
28         double x = sa.parsedouble(&Serial); //接收一个double型数据
29         double y = sa.parsedouble(&Serial);
30         double z = sa.parsedouble(&Serial);
31
32         Serial.println("get xyz");
33         MyArm.inverse_movement(x, y, z); //姿态逆解运算
34         Serial.print("x = "); Serial.print(x);
35         Serial.print(" y = "); Serial.print(y);
36         Serial.print(" z = "); Serial.println(z);
37         for(int i = 0; i < 3; i++)
38         {
39             //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)
40             Serial.print(MyArm.Rad2Angle(MyArm.theta[i]));
41             Serial.print(" ");
42         }
43         Serial.println();
44     }
45 }

```

图 4.3.0

The notes are clear, there are no difficulties, what needs can be directly linked to us, we do not repeat here. Go directly to download validation

4.4 Download verification

Open the routine as shown in the figure below:

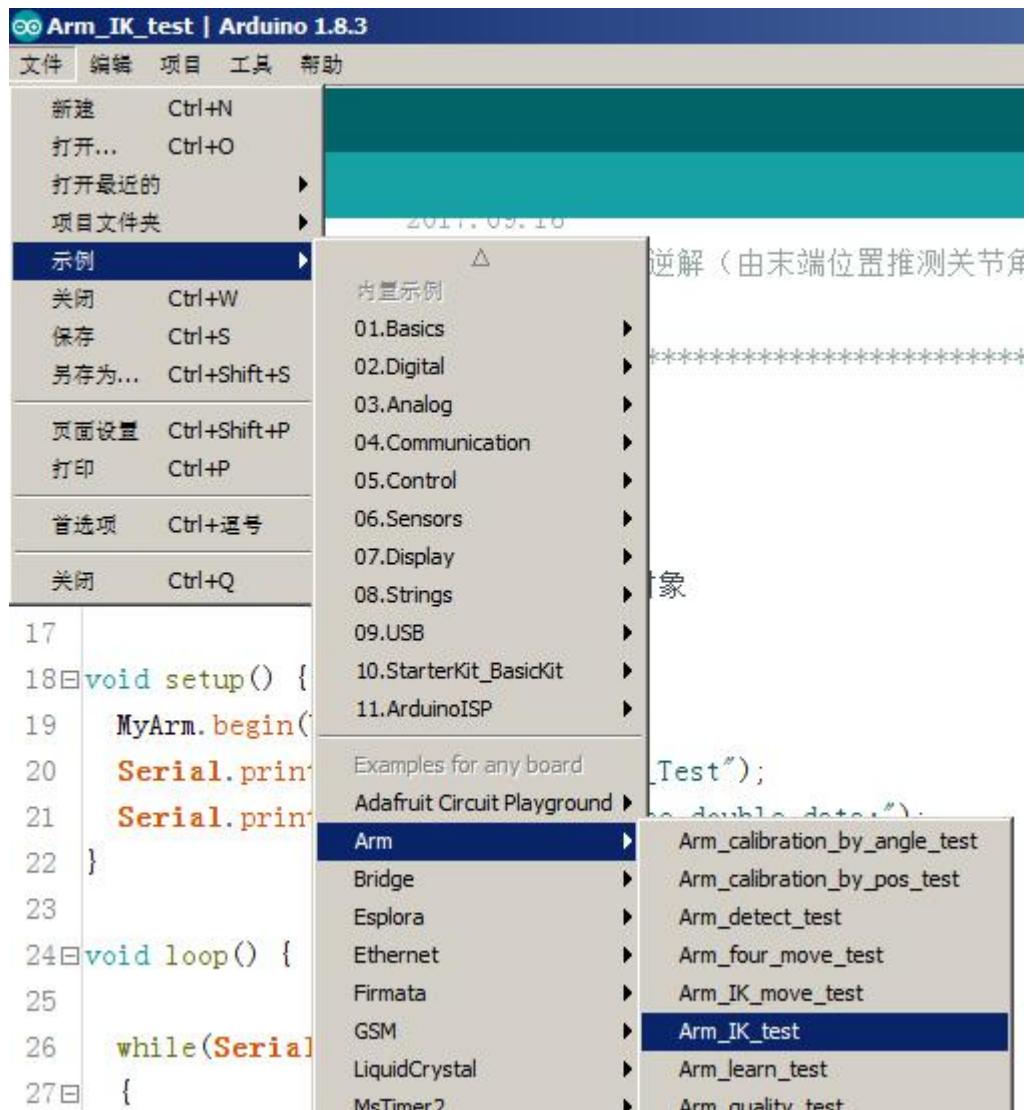


图 4.4.0

Then download directly, and when the download is complete, open the serial port, as shown in the following figure:

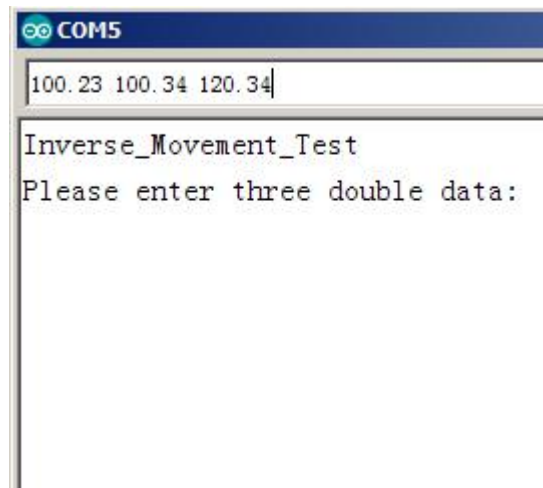


图 4.4.1

As prompted, enter three appropriate coordinate values for type double. Click enter to return the value shown below:

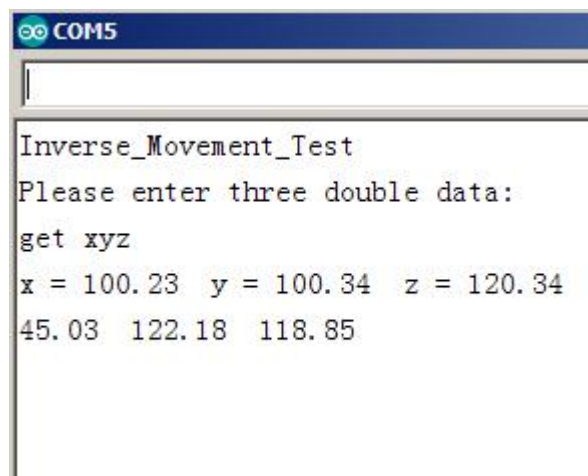


图 4.4.2

Returns three values, which are the three angle values of the servo 012。

第五章 Inverse kinematic test of robot arm

The inverse motion of the robot arm is actually the inverse solution and then the motion to the position of the robot arm.。

- 5.1 A brief introduction to the inverse kinematic process of the robot arm
- 5.2 An introduction to inverse Motion function of robot arm
- 5.3 Programming on Arduino
- 5.4 Download verification

5.1 A brief introduction to the inverse kinematic process of the robot arm

The inverse kinematics of the robot arm, literally, is the inverse solution and motion of the robot arm, and then he should include two processes at the same time, that is, inverse and motion, and it is really true. In fact, we need to give a coordinate and give the time the end of the robot arm will run to the time it needs to move to the coordinate. The coordinate.

5.2 An introduction to inverse Motion function of robot arm

We refer to the Chinese Manual of the 7bot robot arm firmware function Library, and look at the list of public functions in the Arm class as shown in the figure below, where we find functions such as the red circle.。

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
position_init	机械臂位置初始化
inverse_movement	机械臂坐标位置逆解
move_to_position	机械臂运动控制函数
Para_Init	机械臂参数初始化函
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 号角
Get_Offset	得到机械臂的偏置函
offset_by_pos	通过机械臂的直接数

图 5.2.0

We go to the directory to find the function, and click to find the details of it. As shown below:

1.2.4 函数 move_to_position (重载 1)

注意: move_to_position 有 3 个重载, 这是第 1 个

Table7. 描述了函数 move_to_position

Table7.

函数名	move_to_position
函数原型	void move_to_position(double x_ , double y_ , double z_ , word runtime);
功能描述	(3 个参数代表机械臂末端位置), 机械臂在 runtime 时间内运行到该坐标位置。
输入参数 1	X_: 机械臂末端的 x 轴坐标位置, 单位 mm
输入参数 2	Y_: 机械臂末端的 y 轴坐标位置, 单位 mm
输入参数 3	Z_: 机械臂末端的 z 轴坐标位置, 单位 mm
输入参数 4	runtime: 运行到目标位置的时间, 单位 ms
返回值	无
先决条件	无
被调用函数	inverse_movement

例:

```
/**初始化通信串口为 USB_SER, 并且在两秒内运动到(120,120, 120)坐标位置**/
MyArm.begin(USB_SER);
MyArm.move_to_position(120,120, 120, 2000);
```

图 5.2.1

In the figure above, you mainly look at the called function, which is the function that we used in the previous chapter. So we are looking at this function today, and it is not difficult for us to understand the internal working mechanism of this function. First of all, It passes the three position information xyz values to the called function (inverse_movement), which resolves the Radian parameters of each servo, and then converts it to the direct position information, to the servo, and to the time the servo runs to that position, so that, We control the coordinated operation of each servo, and the coordinated operation of each servo is the performance of the robot arm.

5.3 Programming on Arduino


```
14 #include<Arm.h>
15
16 Serial_arm sa; //初始化一个串口接收对象
17 void setup() {
18     MyArm.begin(USB_SER);
19     Serial.println("Arm_IK_move_Test");
20     Serial.println("Please enter three double and an int data:");
21     Serial.println("such as : 100.12 106.7 114.5 2000");
22     MyArm.position_init();
23     delay(2000);
24 }
25
26 void loop() {
27     while(Serial.available())
28     {
29         double x = sa.parsedouble(&Serial); //接收一个double型数据
30         double y = sa.parsedouble(&Serial);
31         double z = sa.parsedouble(&Serial);
32         int t = Serial.parseInt();
33
34         Serial.println("get xyzt");
35
36         Serial.print("x = "); Serial.print(x);
37         Serial.print(" y = "); Serial.print(y);
38         Serial.print(" z = "); Serial.print(z);
39         Serial.print(" t = "); Serial.println(t);
40
41         //在t的时间内，机械臂的末端位置运行到 (x, y, z) 位置处
42         MyArm.move_to_position((double)x, y, z, t);
43         for(int i = 0; i < 3; i++)
44         {
45             Serial.print(MyArm.Rad2Angle(MyArm.theta[i]));
46             Serial.print(" ");
47         }
48         Serial.println();
49     }
50 }
```

图 5.3.0

Code is not difficult, there are any requirements can be directly linked to us, here do not repeat. Go directly to download validation。

5.4 Download verification

Open the routine as shown in the figure below:

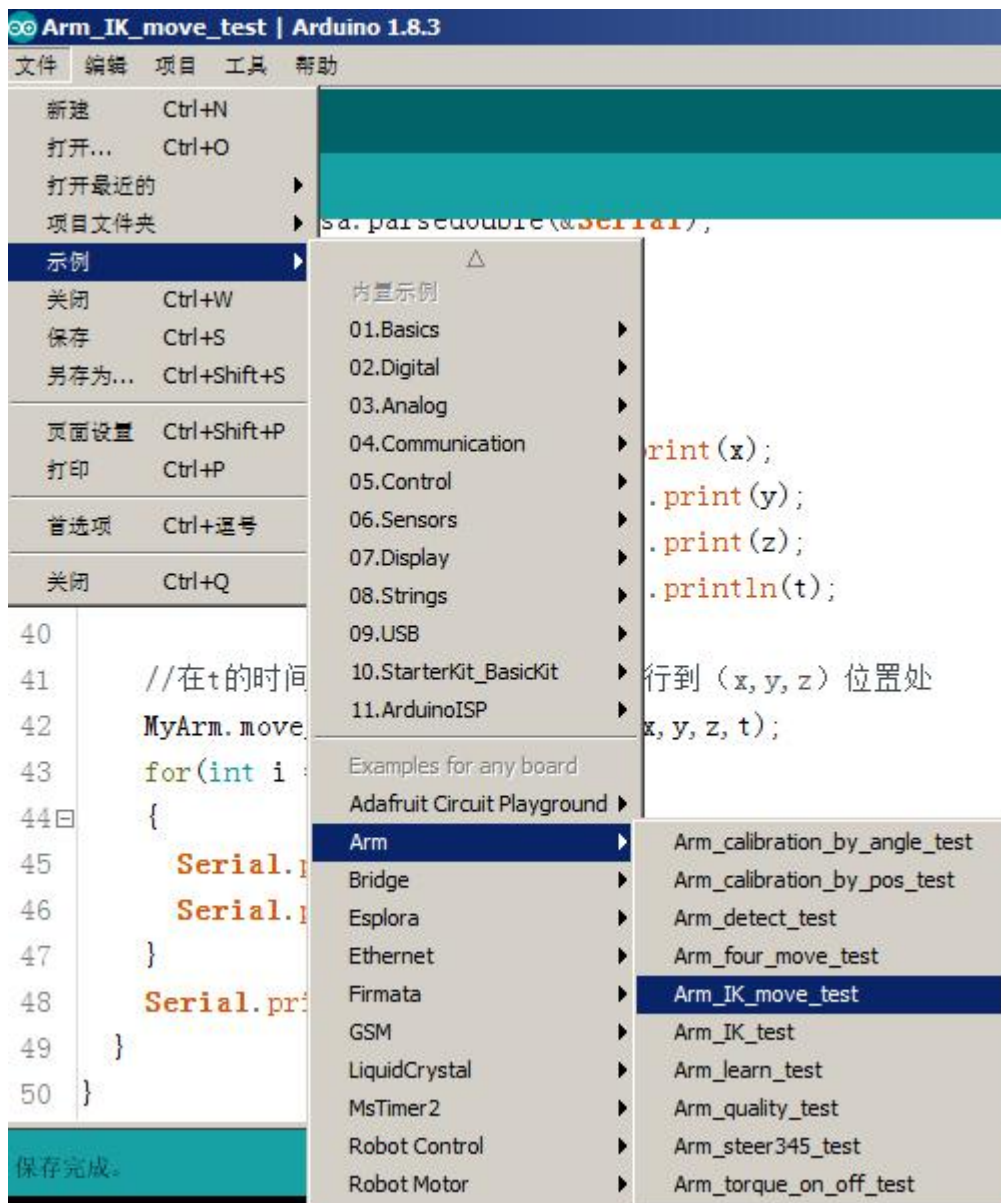


图 5.4.0

Then download directly, and when the download is complete, open the serial port, as shown in the following figure:

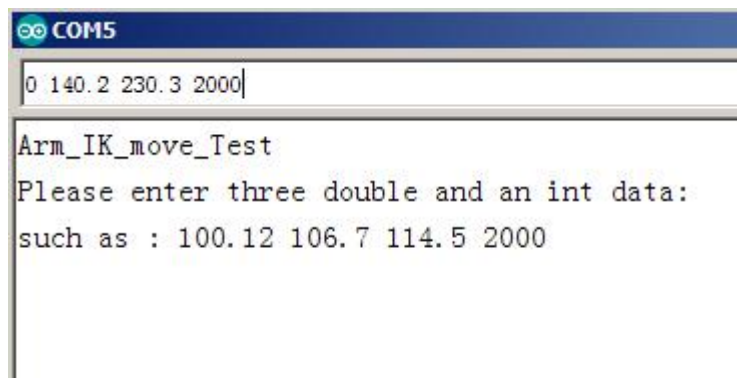
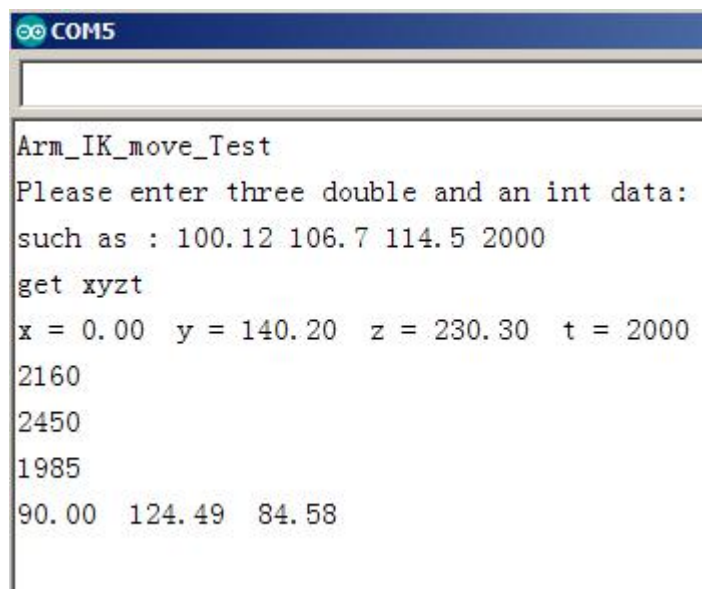


图 5.4.1

According to the prompt, enter three appropriate coordinate values for double type, and then enter the running time to that coordinate. Click enter to return the value shown below:

图 5.4.2



You print the value of the information you enter, then you have the direct data value that prints the inverse solution, and finally you print out the angle value of the servo 012 for the inverse solution.

At the same time, you'll see your robot arm move, and the action is in the right position.

Note: it is still the servo 012 linkage, the servo 345 is easy to control so we take it out alone. The next chapter is about the linkage of the servo 345.

Important: remember to be careful with the coordinates, because now we are not turning on the protection, the robot arm may be damaged.

In addition, the function has three overloads, please go yourself to learn the other two.

Chapter 6 robot arm end rotation test

The previous chapters were just using the inverse solution to deal with where the end was, but if the end had an end servo, it was like a human hand, and you had to rotate your wrist at the right angle when you took something. So we're going to talk about the control of rotation.

- 6.1 Brief introduction of end rotation Control of robot arm
- 6.2 An introduction to the end rotation function of the robot arm
- 6.3 Programming on Arduino
- 6.4 Download verification

6.1 Brief introduction of end rotation Control of robot arm

Let's count, without the end servos, the robot arm has six servo in total, and the first three servo control methods are described in the previous section, and the latter three servo controls, that is, the arm end rotation control.

6.2 An introduction to the end rotation function of the robot arm

We refer to the Chinese Manual of the 7bot robot arm firmware function Library, and look at the list of public functions in the Arm class as shown in the figure below, where we find functions such as the red circle.

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 号舵机旋转运动函数
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置机械臂偏置函数
offset_by_angle	通过机械臂的角度设置机械臂偏置函数
Rad2Angle	弧度值转角度值函数
Pos2Rad	直接位置数据转换弧度值函数

图 6.2.0

We go to the directory to find the function, and click to find the details of it. As shown below:

1.2.12 函数 turn_steer_345_to_positon(重载 2)

注意：该函数有两个重载，这是第 2 个

Table15. 描述了函数 turn_steer_345_to_positon

Table15.

函数名	turn_steer_345_to_positon
函数原型	Boolean turn_steer_345_to_positon(double angle3 , double angle4, double angle5 , word runtime);
功能描述	第 3, 第 4, 第 5 号舵机旋转运动函数 重要：舵机 345 的状态设置函数
输入参数 1	angle3: 舵机 3 的角度，范围（0~360） 注意：：如果开启保护，以保护中的位限为主
输入参数 2	angle4:舵机 4 的角度，范围（0~360） 注意：：如果开启保护，以保护中的位限为主
输入参数 3	angle5: 舵机 5 的角度，范围（0~360） 注意：：如果开启保护，以保护中的位限为主
输入参数 4	runtime: 机械臂由当前位置运行到指定位置所花的时间，单位是毫秒
返回值	无
先决条件	无
被调用函数	Set_Steer_position_runtime

例:

```
/**初始化通信串口为 USB_SER，并且使机械臂的 345 号舵机在两秒内运行到正中位置**/
MyArm.begin(USB_SER);
MyArm.Set Steer position runtime (90,90,90, 2000);
```

图 6.2.1

Notice that this is overloaded 2, that means that there is overloading 1, you can go to the Chinese manual to see, the parameter of overloading 1 is to receive direct data, is not the angle value, is the servo can directly write to register running data, And what we're using here is overloading 2, we're inputting angle information, and the type of data we're entering can be seen in the function prototype above, and it's double data.。

Important: the positive-angle information of the servo is 90 °, general input range (0 ° ~180 °)。

6.3 Programming on Arduino


```

14 #include<Arm.h>
15
16 Serial_arm sa; //初始化一个串口接收对象
17 void setup() {
18     MyArm.begin(USB_SER);
19     Serial.println("Arm_steer345_test");
20     Serial.println("Please enter three double and an int dat
21     Serial.println("such as: 70.9 80.9 120.12 3000");
22     MyArm.position_init(); //机械臂位置初始化
23     delay(2000);
24 }
25
26 void loop() {
27     while(Serial.available())
28     {
29         double st3 = sa.parsedouble(&Serial); //接收一个角度数据
30         double st4 = sa.parsedouble(&Serial);
31         double st5 = sa.parsedouble(&Serial);
32         word t = Serial.parseInt(); //得到运行时间参数
33
34         Serial.println("get st3 st4 st5 t");
35
36         Serial.print("st3 = "); Serial.print(st3);
37         Serial.print(" st4 = "); Serial.print(st4);
38         Serial.print(" st5 = "); Serial.print(st5);
39         Serial.print(" t = "); Serial.println(t);
40
41         //机械臂运动到输入的位置
42         MyArm.turn_steer_345_to_positon(st3, st4, st5, t);
43         Serial.println();
44     }
45 }

```

图 6.3.0

Code is not difficult, there are any requirements can be directly linked to us, here do not repeat. Go directly to download validation

6.4 Download verification

Open the routine as shown in the figure below:



图 6.4.0

Then download directly, and when the download is complete, open the serial port, as shown in the following figure:

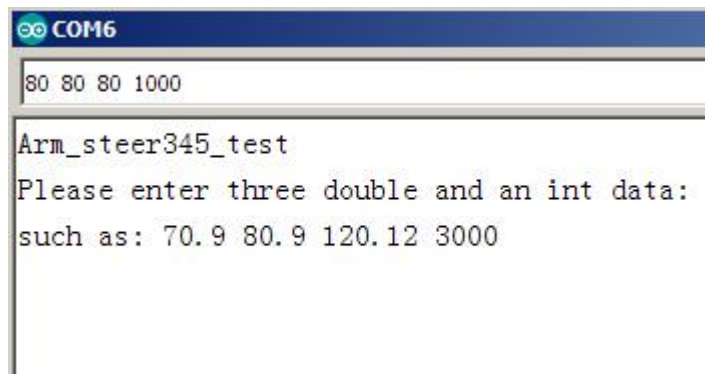
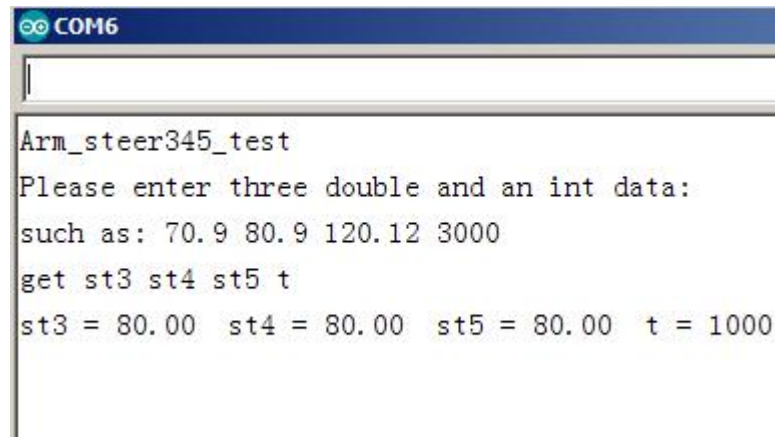


图 6.4.1

According to the prompt, enter three appropriate angle value double type, and then enter the running time to that angle. Click enter to return the value shown below:

图 6.4.2



```
COM6
Arm_steer345_test
Please enter three double and an int data:
such as: 70.9 80.9 120.12 3000
get st3 st4 st5 t
st3 = 80.00 st4 = 80.00 st5 = 80.00 t = 1000
```

You print the value of the information you enter, then you print the direct value of the inverse solution, and finally you print out the angle value of the servo 345 of the inverse solution.

At the same time, you will see your robot arm move, arrived exactly where the position.

Important: remember to be careful with the angle of rotation, because right now we're not turning on the protection, and the arm may be damaged.。

Chapter 7 Comprehensive measurement of robot arm motion

- 7.1 A brief introduction to the Synthetical Measurement of robot arm Motion
- 7.2 An introduction to the synthetic function of robot arm Motion
- 7.3 Programming on Arduino
- 7.4 Download verification

7.1 A brief introduction to the Synthetical Measurement of robot arm Motion

The first two chapters explain the location of the end and the rotation of the end.
Today, we are just making up a set of movements to see the effect of the six linkage.

7.2 An introduction to the end rotation function of the robot arm

Move_to_position / Chapter 5 functions, inverse motion, control of servo 012

Turn_steer_345_to_positon// control servo 345 rotation

7.3 Programming on Arduino

```

14 #include<Arm.h>
15
16 void setup() {
17     MyArm.begin(USB_SER);
18     Serial.println("Arm_four_move_test");
19     MyArm.position_init();
20     delay(3000);
21 }
22
23 void loop() {
24     //姿态1
25     MyArm.move_to_position(100.0, 200, 232, 500);
26     MyArm.turn_steel_345_to_positon((word)2300, 2600, 2700, 500);
27     delay(510);
28     //姿态2
29     MyArm.move_to_position(-100.0, 200, 232, 700);
30     MyArm.turn_steel_345_to_positon((word)1700, 1400, 1300, 700);
31     delay(710);
32     //姿态3
33     MyArm.move_to_position(-100.0, 200, 133, 500);
34     MyArm.turn_steel_345_to_positon((word)2500, 2400, 2700, 500);
35     delay(510);
36     //姿态4
37     MyArm.move_to_position(100.0, 200, 133, 700);
38     MyArm.turn_steel_345_to_positon((word)2047, 2047, 2047, 700);
39     delay(710);
40 }

```

图 7.3.0

With the code, say again those two that are the most common !

- 1) MyArm.move_to_position(-100.0, 200, 133, 500);

The end of the arm moves within the 500ms to the coordinate xyz = (-100,200,133). Note the direction of XY, which is indistinguishable or uncertain, and notes in the first chapter of the Chinese manual.。

- 2) MyArm.turn_steel_345_to_positon((word)2500,2400,2700,500);

robot arm in 500ms, end rotation (servo 345) to direct data position

(2500 / 2400 / 2700) , you can also use angle control。

7.4 Download verification

Open the routine as shown in the figure below:



图 7.4.0

After you open it and download it directly, you will see that the servo circulates together at four pose points.