

## Tutorials on robot arm servo

This actual combat course mainly uses the application layer function of the servo to realize, the concrete function use please refer to "the servo library function Chinese manual".

For better and more convenient testing of the servo, it is recommended that you buy a separate development board and a servo for testing, and of course, you can also test with the servo of the robotic arm, if you want to use the servo on the arm to test. It is recommended that you start using the servo with ID 5, that is, the servo attached to the end mechanism (the servo ID on the end mechanism is 6), as the servo can rotate 360 degrees without being easily damaged.

**Note:** Please use power cord to power up the arm and servo, and then connect the USB wire. Prevent the arm from needing too much current to burn your computer's USB port.

## preparatory work:

The preparation is to put the function library and examples of the servo under the library of arduino, please refer to the OPERATION INSTRUCTION.

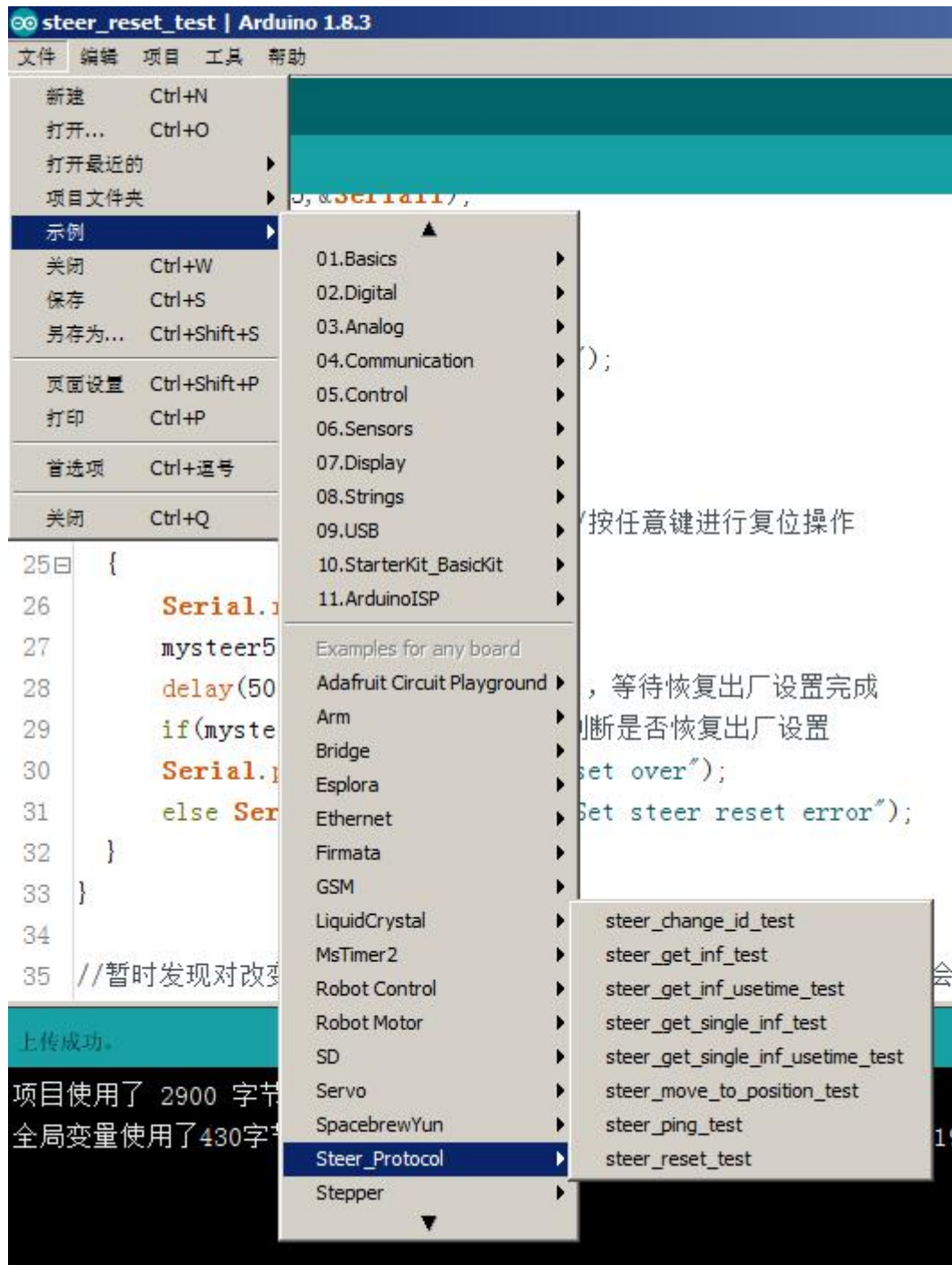


图 0.6

## Chapter 1. SERVO response test (ping)

With a simple servo function, we take you together to respond to our first servo and open the door for us to learn the robotic arm. Now we start the first step into the robotic arm gate, the servo response test.

### Brief introduction of Test response process

The servo response test is to test whether the servo is communicating properly (which is generally used to determine whether the servo is normal), the so-called servo response test, which is a set of data that we send to the servo. This set of data includes the ID of the servo, then the corresponding ID servo will return a set of data to us. By verifying the set of data, we can determine whether the servo is responding properly. In this process, our library functions have been encapsulated for us, and we only need to call the library functions to determine whether our servo is communicating properly or not.

### Programming on Arduino

Because our prerequisite is to initialize a servo object, we look at the default constructor of Steer in the same way to initialize a Steer object.

```
14 #include<Steer.h>
15
16 Steer mystter1(1,&Serial1);
17 //构造函数输入参数：第一个是ID，第二个是与舵机通信的串口
18
19 void setup() {
20     Serial.begin(115200);
21     Serial.println("Steer_Ping_Test");
22 }
23
24 void loop() {
25     boolean flag = mystter1.Steer_Ping();
26     Serial.print("flag = "); Serial.println(flag);
27     delay(1000);
28 }
```

图 1.3.1

We first initialize a Steer steering object, and in order to see the response result, we initialize the serial port Serial to display the result.

You can see that the Steer\_Ping () method is called directly in loop to test the response. Note: refer to the steer\_ping\_test.ino in the routine file for the specific routine code

Important note: as for the encapsulation of servo functions like this, we use the ID attribute to be private because, in general, we don't change the servo ID value, so, When we initialize an object with a servo class, we don't have to worry about its ID-we know that the object we create corresponds to the real physical servo, and we can devote ourselves to the servo control. And the downside is that the flexibility of the ID operation is not that good, and we'll see in the next chapter that we're going to modify the ID.。

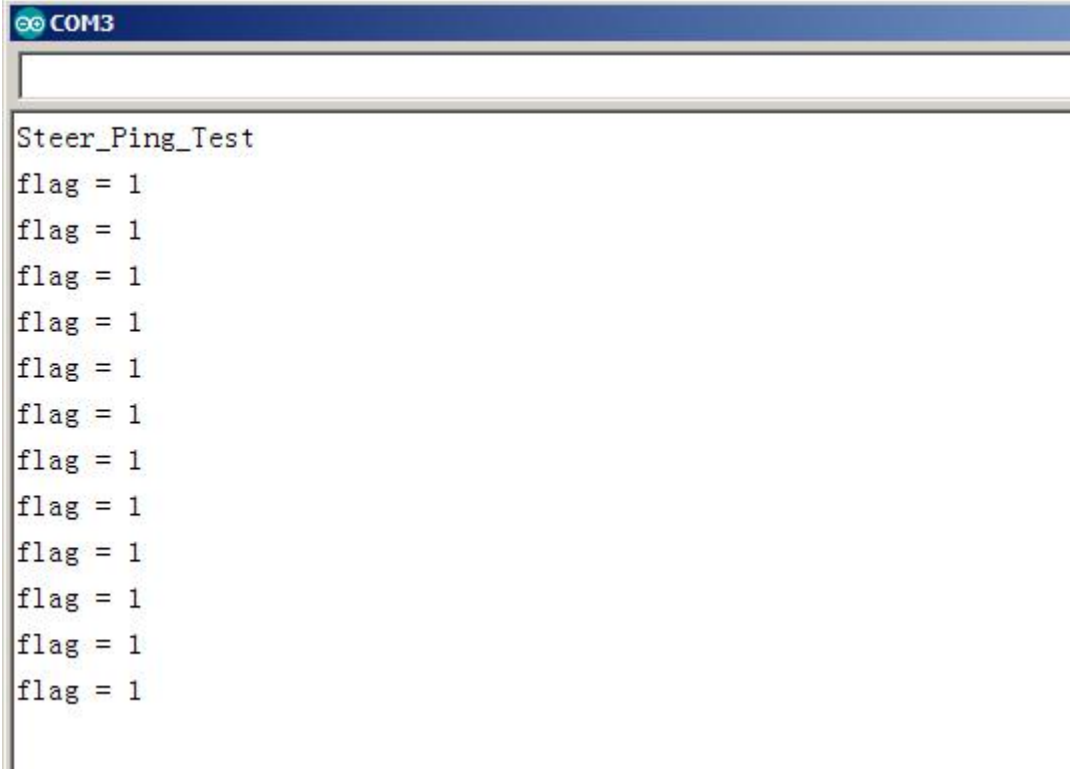
## Download verification

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



图 1.4.0

After downloading the program, open the serial display, we can see the following results:



```
COM3  
  
Steer_Ping_Test  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1  
flag = 1
```

图 1.4.1

Every second to visit the servo, the servo to give a response.

Flag = 1, indicating a successful rudder response

## The second chapter changes the servo ID experiment (**change\_ID**)

### 2.1 Brief introduction of changing servo ID

Changing the servo ID is changing the original servo ID to the ID we want, by calling Change\_Steer\_ID function to change the ID。

Note: after changing the servo ID, we can still use this object to control the servo because the id in the private variable of the object is changed at the same time。

### 2.2 Introduction of changing servo ID function

Through the lookup manual, we see the following functions:

Set_Steer_Min_Angle_Limit	设置舵机的最小角限制, 建议设置在 4096, 即 360° 之内
Set_Steer_Torque_On	设置舵机舵机扭矩开关为开状态, 即舵机获得扭矩
Set_Steer_Torque_Off	设置舵机舵机扭矩开关为关状态, 即舵机失去扭矩
Set_Steer_position_runtime	舵机运行函数, 配置该函数使舵机产生运动
Change_Steer_ID	改变舵机的 ID
Set_Steer_Reset	舵机恢复出厂设置
Get_Steer_All_Inf	得到舵机的所有当前信息

4 / 17

图 2.2.0

Referring to the first chapter, we refer to the following function prototypes:

```
void Change_Steer_ID( byte new_id);
```

For this function, there's nothing to say about it. Let's just input the new ID of the servo, and its ID will change, so let's take a look at what's inside the function.。

```
void Steer::Change_Steer_ID( byte new_id){ byte
    new_id_ = new_id;
    write(id, 0x05, &new_id_, sizeof(new_id_)); id =
    new_id;
}
```

The ID variable is a private variable of Steer. In fact, when we do any operation to the servo, that is, when we call almost all of the application layer functions, we call ID. We do not accidentally change the ID, so we set the ID as a private property, and when initializing Steer, the constructor performs the assignment of the ID. operation。

Again, changing the ID of the servo can still call the servo with this object, because there is a statement in the function (id = newid;), look at the last line of the function just given, that is, when we change the ID of the servo, We also changed the value of the private variable id inside the servo object。

How to use this function is to initialize a servo object, and then use that object to call the function, such as:

```
Steer str1(1,&Serial1);          //We initialize a servo ID :1
```

```
Str. Change_Steer_ID(2); //We changed its servo ID to 2
```

Note that when we typically initialize the general servo ID, we initialize the number behind its str, for example; The servo ID is 1, and our object is named str1, so we generally see the number behind the servo object, and we know the id of the servo (because we don't change the servo ID when we use it normally, so we name it this way. This lesson is to change servo IDs, so we can name them according to our own ideas.



## 2.3 Arduino routine file writing

```
14 #include<Steer.h>
15
16 Steer mystter1(1,&Serial1);    //初始化一个舵机ID为1的对象
17
18 void setup() {
19     Serial.begin(115200);        //初始化与电脑通信串口（波特率115200）
20     Serial.println("steer_change_id_test");
21     Serial.println("Please enter in the following format: new_ID");
22     Serial.println("such as: 3");
23 }
24
25 void loop() {
26     while(Serial.available()){    //等待输入一个新的ID
27         byte new_id = Serial.parseInt();    //接收新ID
28         mystter1.Change_Steer_ID(new_id);    //改变舵机的ID为新ID
29         delay(500);                //等待改变，必须至少等待500ms（重要）
30         if(mystter1.Steer_Ping())    //检验是否改变ID成功
31             Serial.println("Change over. Please input another.");
32
33         else Serial.println("error: failed to change id");
34     }
35 }
```

图 2.3.0

## Download verification

Find this function in the example, as shown in the following figure:



图 2.4.0

Validation results:

The new ID number is entered according to the prompt. I input 16 as shown below.。

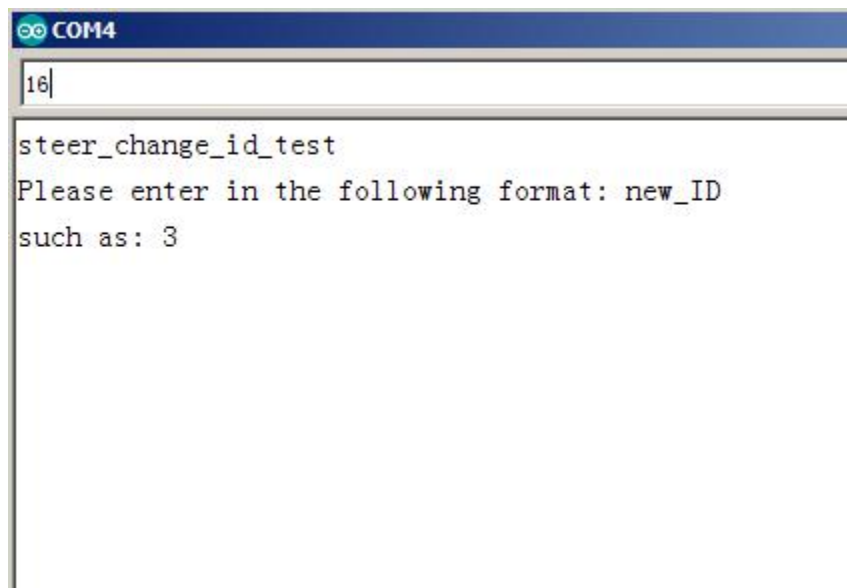


图 2.4.1

Press enter to get the following results to indicate success under the change ID。

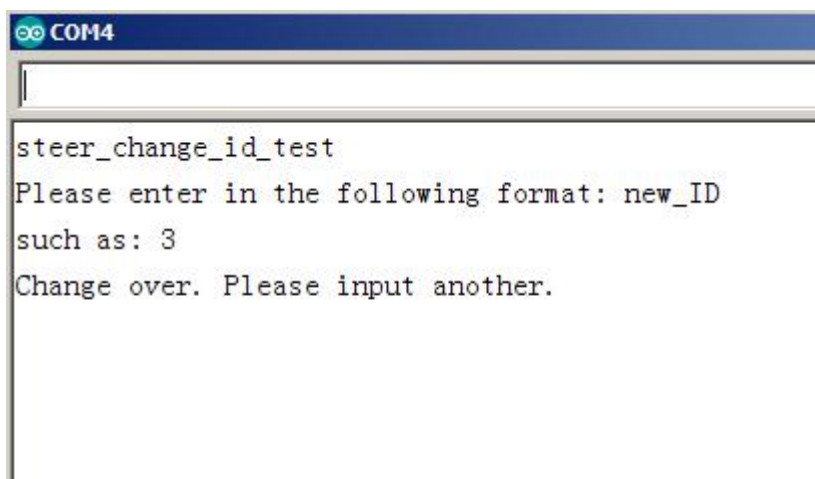


图 2.4.2

We can keep typing, keep changing.。

## servo position control experiment

### (Set\_Steer\_position\_runtime)

In this chapter, we're going to take control of the servo, and we just need to adjust the function above, and you can easily control the servo, so let's get started.

#### 3.1 Brief introduction of Control servo

Control servo:

Control what? How to control the rotation position of the servo? Direct input location information

Is there a problem? Yes, then enter the position, how long does it run to that position, can you control it? Of course, so if we want to control the servo, we need to input two information to the servo, that is, the position and time of the servo.。

#### 3.2 Introduction of Control steering Machine function

So, through the introduction, we know that we need to control a servo that should have at least location and time information, so we look up the Chinese manual of the servo, and we see the function, and we refer to the first chapter, and I'm going to give the prototype of the function directly.。

```
void Set_Steer_position_runtime(word pos, word runtime);
```

There are really only two input parameters. The first parameter POS is the location information, while the parameter runtime is the running time. Let's introduce the two parameters in detail.。

pos: Position parameter, value range (0~4096) , Corresponding angle range (0~360°)

runtime: Run time, value range (0~30000) , unit ms

Method of use: we initialize a servo object, directly call the function, input the corresponding position and time information, we can control the servo to the corresponding position. As follows:

```
Steer str5(5, &Serial1); //Initialize an ID 5 servo
str5. Set_Steer_position_runtime(2047, 2000); //Run to the middle of two seconds
```

### 3.3 Arduino routine file writing

Open routine **steer\_move\_to\_position\_test**, We see the following procedure:

```

14 #include<Steer.h>
15
16 Steer mysteer5(5,&Serial1);          //初始化一个ID为5的舵机对象
17
18 void setup() {
19     Serial.begin(115200);
20     Serial.println("Please input in the following format :");
21     Serial.println("position(word)  runtime(word)");
22     Serial.println("such as : 2334 4301");
23 }
24 int i = 0;                          //记录次数
25 void loop() {
26     while(Serial.available()){      //等待接收数据
27         word pos = word(Serial.parseInt()); //读取位置数据
28         word tim = word(Serial.parseInt()); //读取时间数据
29         Serial.println(i++);         //输出次数
30         Serial.print("position = "); Serial.println(pos);
31         Serial.print("time = "); Serial.println(tim);
32         Serial.println();
33         mysteer5.Set_Steer_position_runtime(pos, tim); //控制舵机旋转
34     }
35 }

```

图 3.3.0

### Download verification

Find an example of the program below, click Open and download it to the chip:



图 3.4.0

After burning program, open the serial display and enter the relevant values according to the format of the prompt, as shown in the following figure:

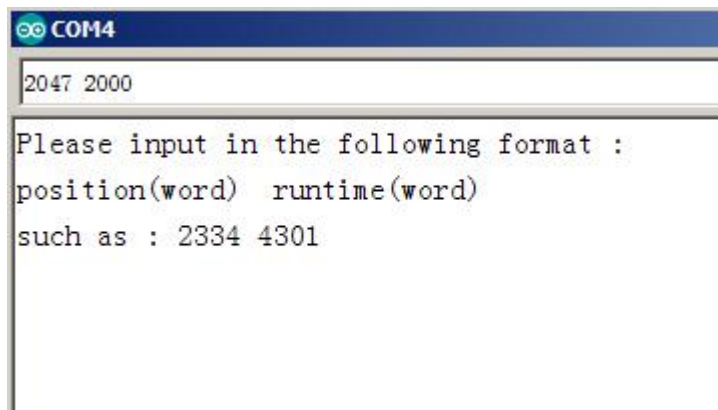


图 3.4.1

Press enter and hear that the servo with ID 5 starts spinning

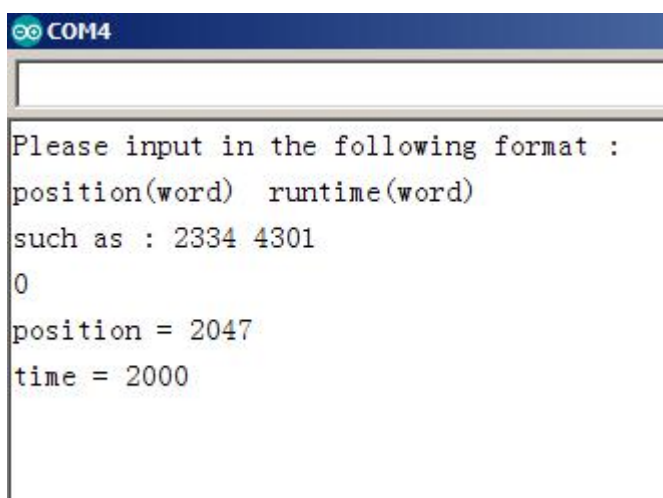


图 3.4.2

If you don't see your servo rotating, there are generally three possibilities: first, the servo is already in this position, so the servo won't turn. Second, the servo ID is wrong, you wrong servo ID number, we suggest that you use the first chapter of the routine test to know. Third, the servo is broken, which is rare, but it is also possible。



## The experiment of restoring the installation of the servo to the factory

( **Set\_Steer\_Reset** )



A brief introduction of the experiment on the recovery of the servo from the factory

What is the use of restoring the factory setting? Restoration of factory settings, as the name implies, is to restore all the internal registers of the servo to factory settings, that's all.

Old question, how to restore the servo to factory setting? The old answer, we encapsulate, you can call directly.

**Note: when the factory setup is restored, the servo ID is set to 0X01.**

Introduction to the installation function of the recovery of the servo from the factory

In fact, this function is very simple, and we're going to give the prototype of the function directly.

The prototype of the function is as follows:

**void Set\_Steer\_Reset();**

> Using the method, initialize a servo object and call the method directly with that object. As follows:

```
Steer str5(5, &Serial1);    //Initialize a servo object with ID 5  
Str5. Set_Steer_Reset();    //Restore the servo to factory setting
```

Again, the servo's ID is set to 0X01 after the servo is restored.。

## 5.1 Arduino routine file writing

We open the `steer_reset_test` program in the servo application layer routine, and we see the following program:

```
14 #include<Steer.h>
15 Steer mysteer5(5,&Serial1);
16
17 void setup() {
18     Serial.begin(115200);
19     Serial.println("steer_reset_test");
20 }
21
22 void loop() {
23     while(Serial.available())    //按任意键进行复位操作
24     {
25         Serial.read();
26         mysteer5.Set_Steer_Reset();
27         delay(500);    //至少延时500ms，等待恢复出厂设置完成
28         if(mysteer5.Steer_Ping()) //判断是否恢复出厂设置
29             Serial.println("Set steer reset over");
30         else Serial.println("Error: Set steer reset error");
31     }
32 }
```

图 5.3.0

Download verification

Find and open the example as shown in the following figure:

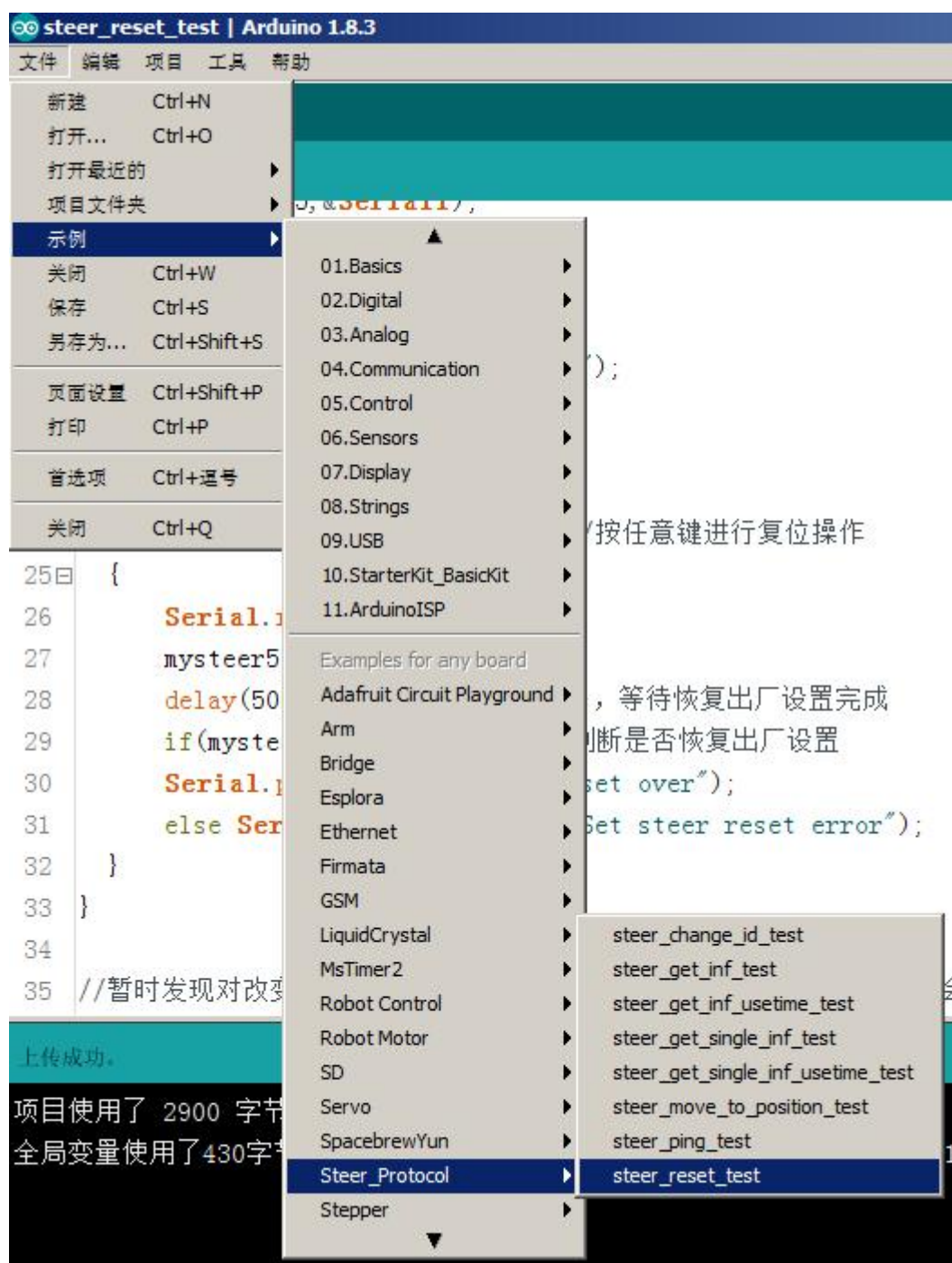


图 5.4.0

Burn the program to the chip and open the serial display, as shown in the following figure:

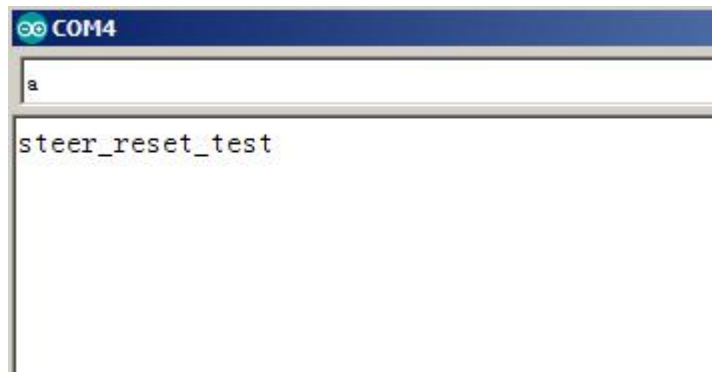


图 5.4.1

We enter any value and press enter button to start restoring factory settings, as shown in the following figure, indicating successful recovery of factory settings:

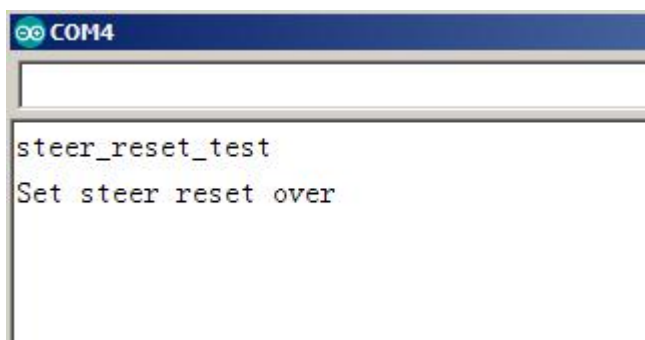


图 5.4.2

Note: ID after restoring factory settings is set to 0X01。