实训项目 6 输送链码垛

1、在 ROBOTSTUDIO 中进行机器人及周边设备的合理布局。
2、在 ROBOTSTUDIO 中的基本工具的使用
3、Smart 组件的基本使用
4、机器人 IO 信号的设定与链接
5、机器人轨迹的创建
6、仿真的调试。

机器人工作站的布局

所有的部件已包含在打包文件中。双击打开打包文件后，请按照以下的图 1 中所示进行布局。

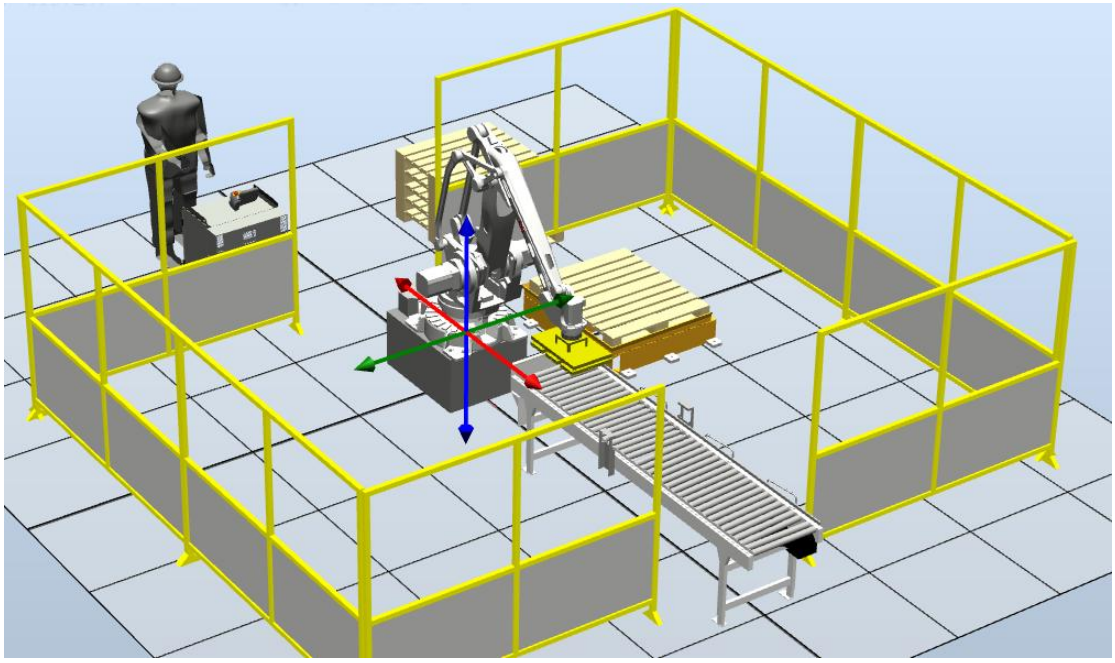


图 1 机器人工作站的布局示意

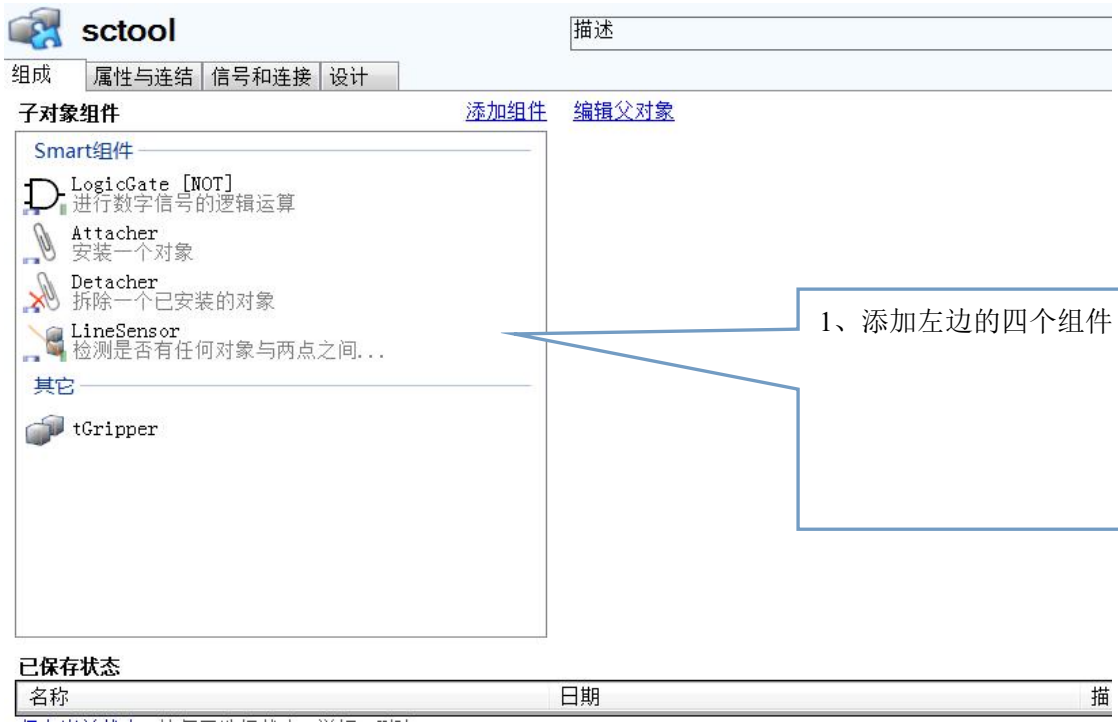
要注意的问题：

1、机器人与周边的部件的位置要合理，周边部件应在机器人的工作范围的中间位置为佳。

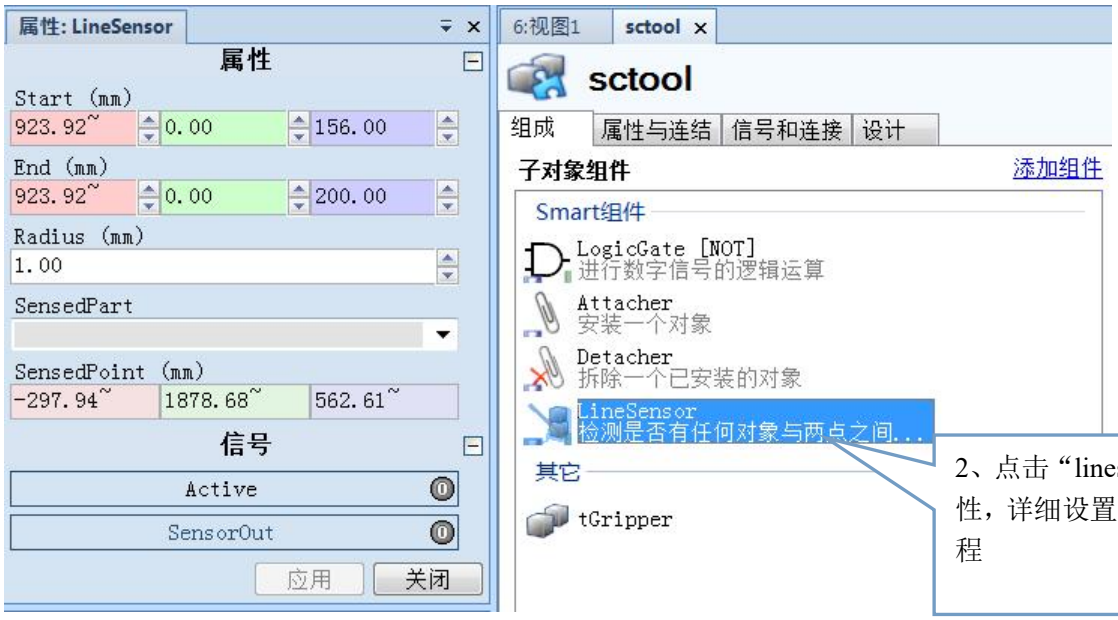
2、可以对机器人的操作，以确认机器人可以到达要取、放的最远端是可以顺利到达的，否则以后再调整就会很麻烦了。

Smart 组件的基本使用

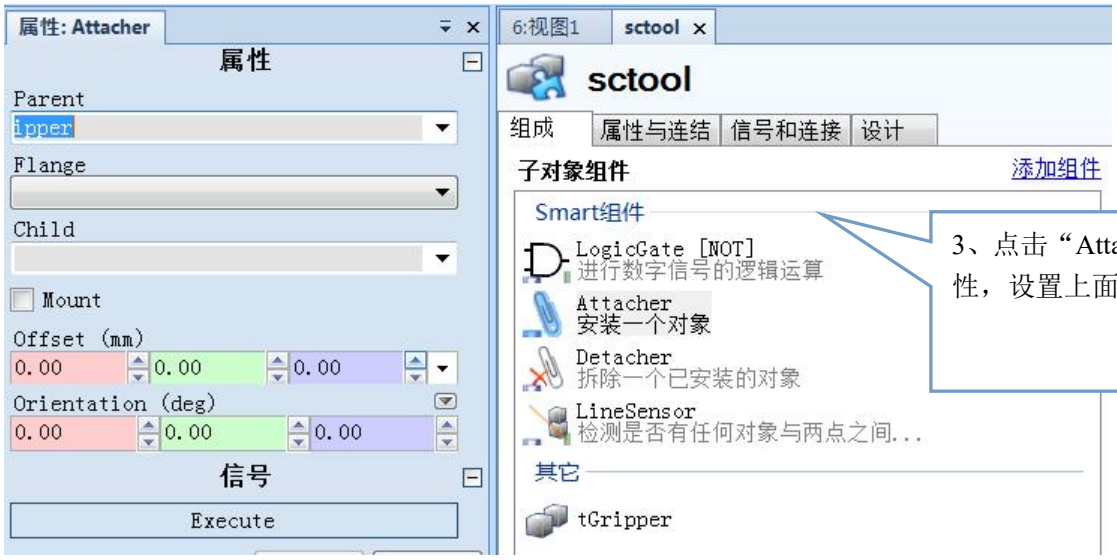
smart 组件创建操作：建模---smart 组件---添加组件，建立如下表组件，详情看视频。



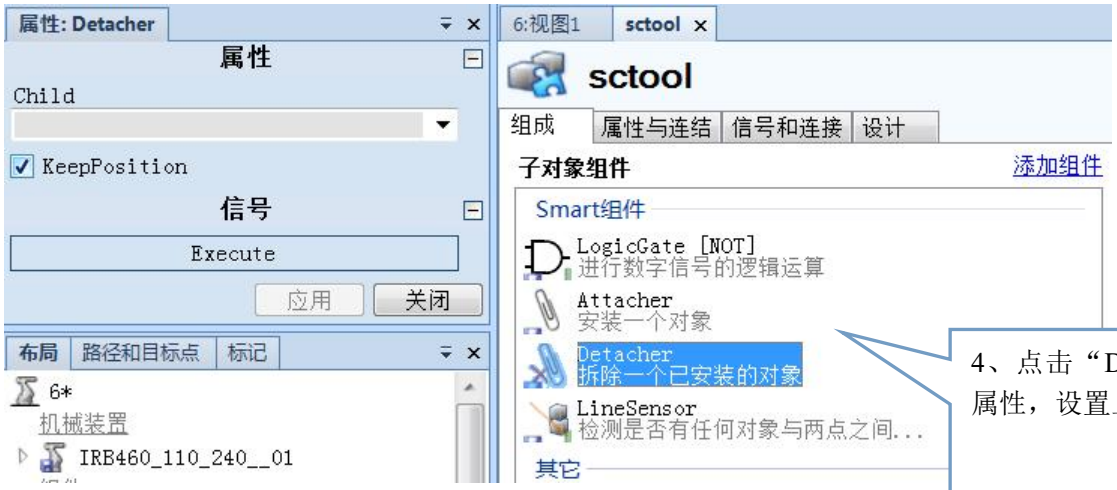
1、添加左边的四个组件



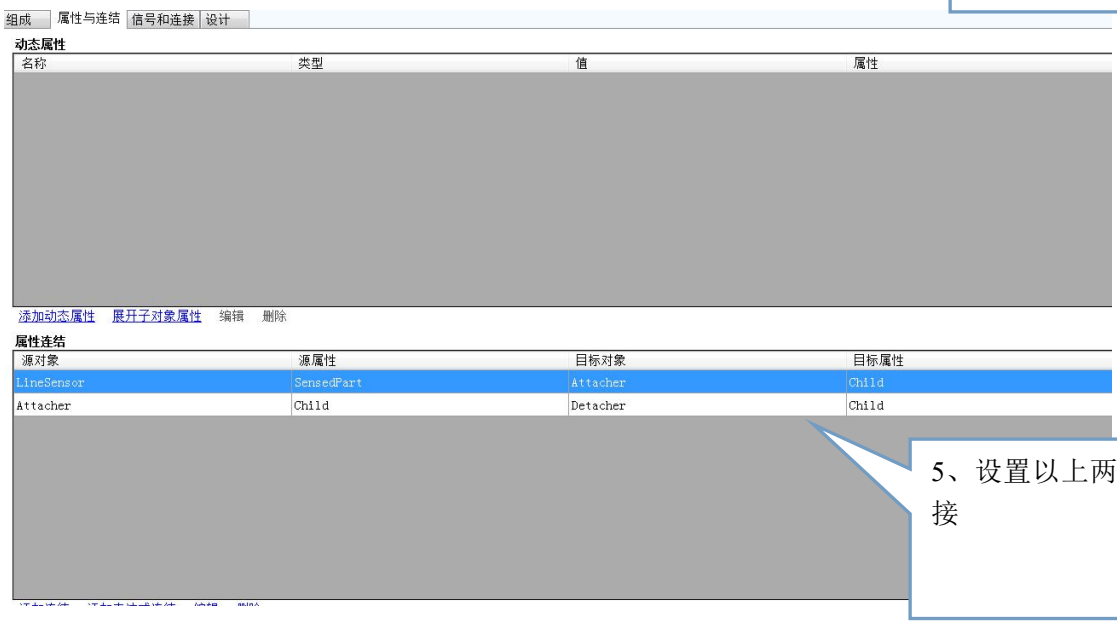
2、点击“linesenor”设置属性，详细设置方法看视频教程



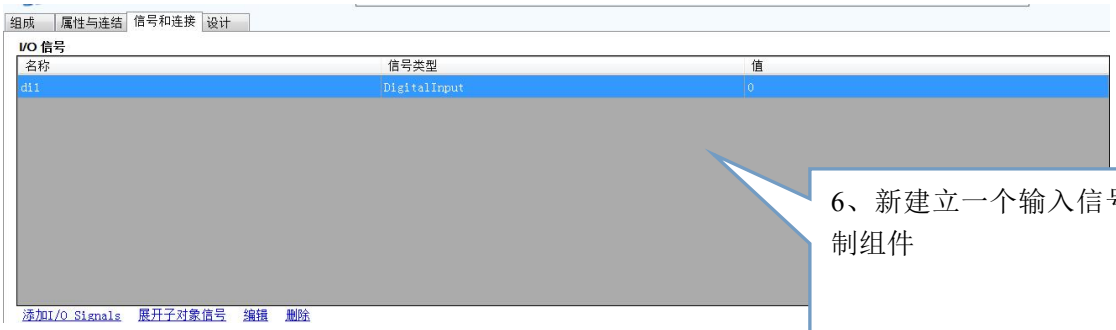
3、点击“Attacher”设置属性，设置上面参数



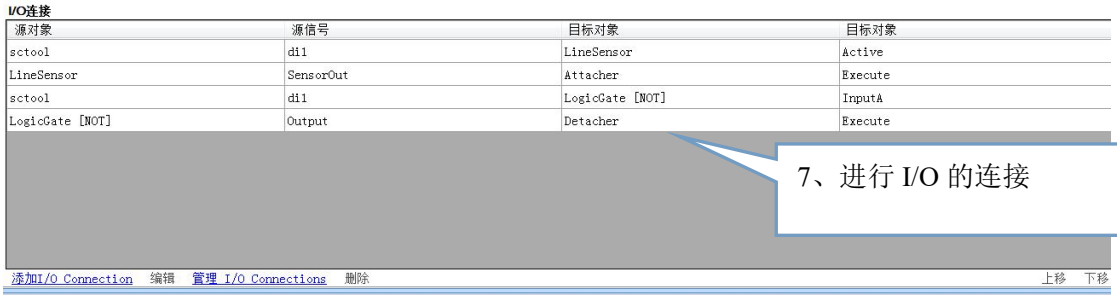
4、点击“Dettacher”设置属性，设置上面参数



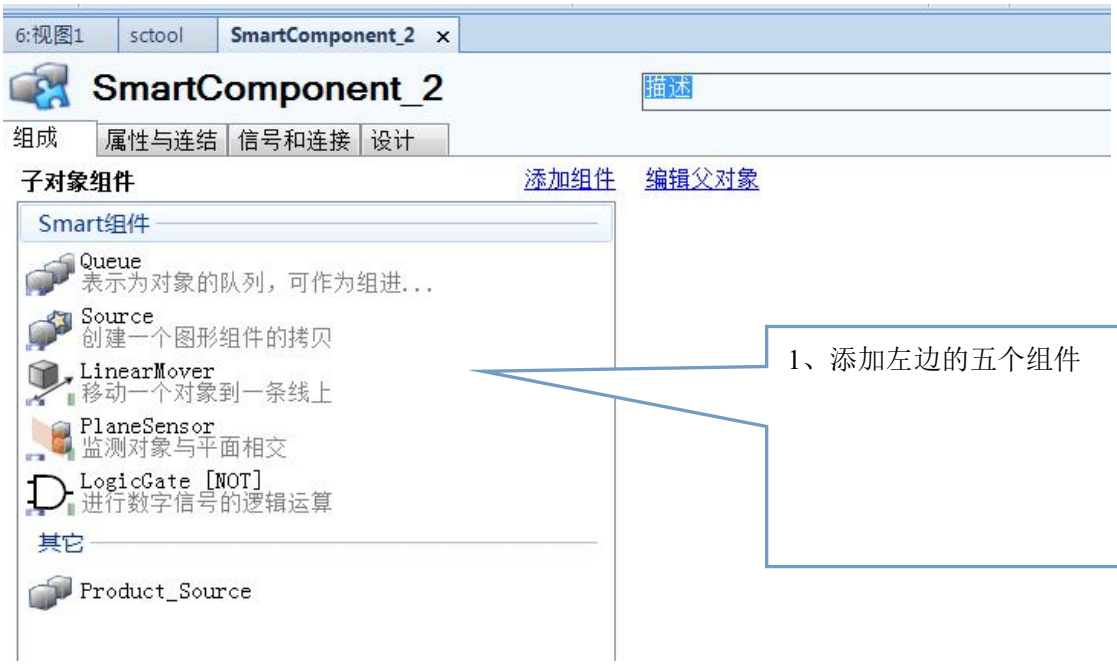
5、设置以上两个属性与连接



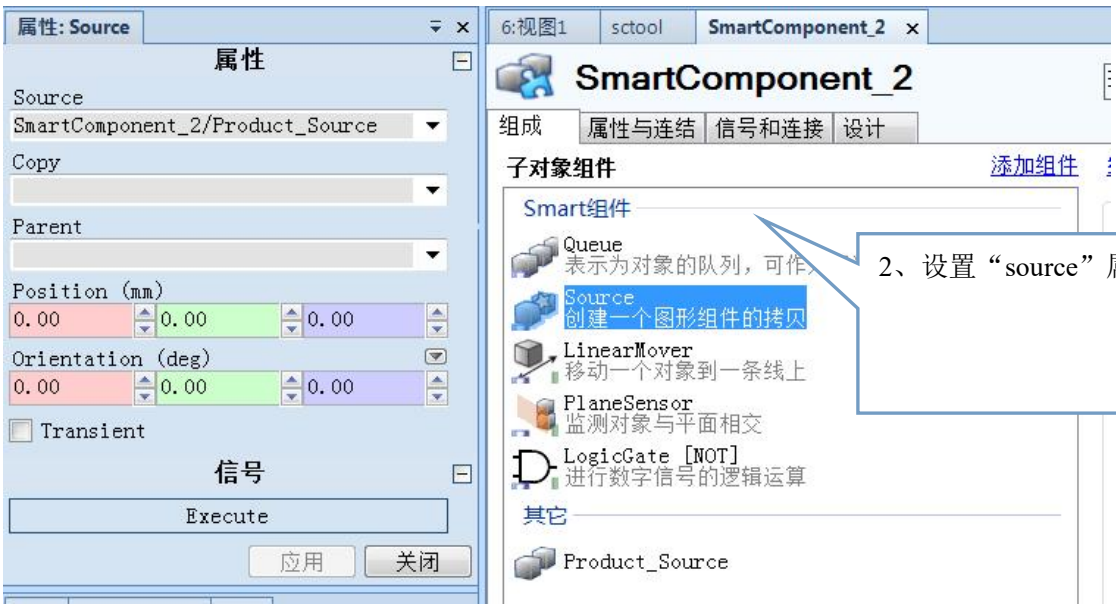
6、新建立一个输入信号控制组件

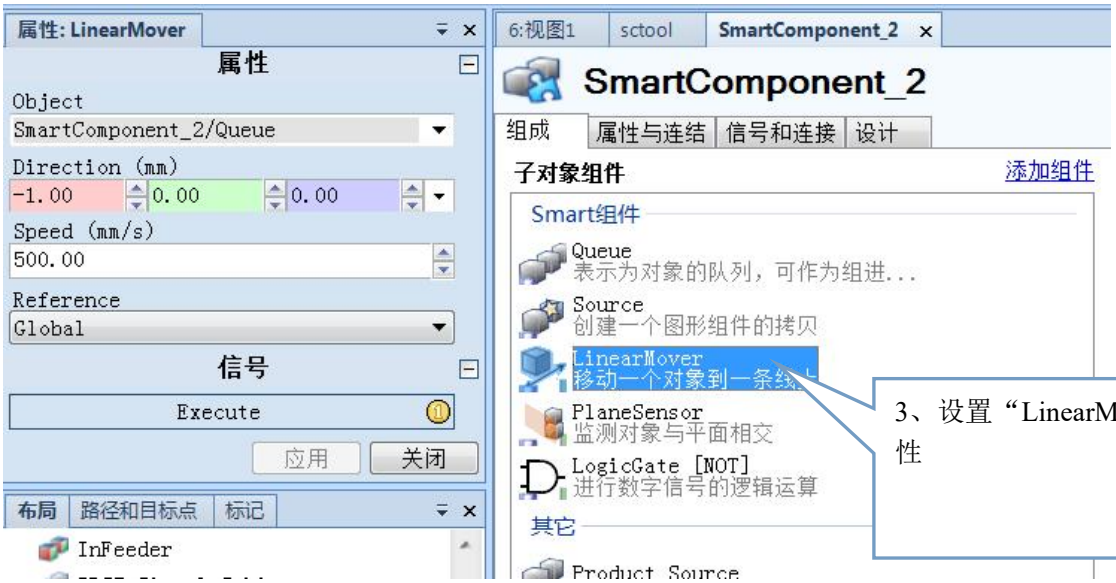


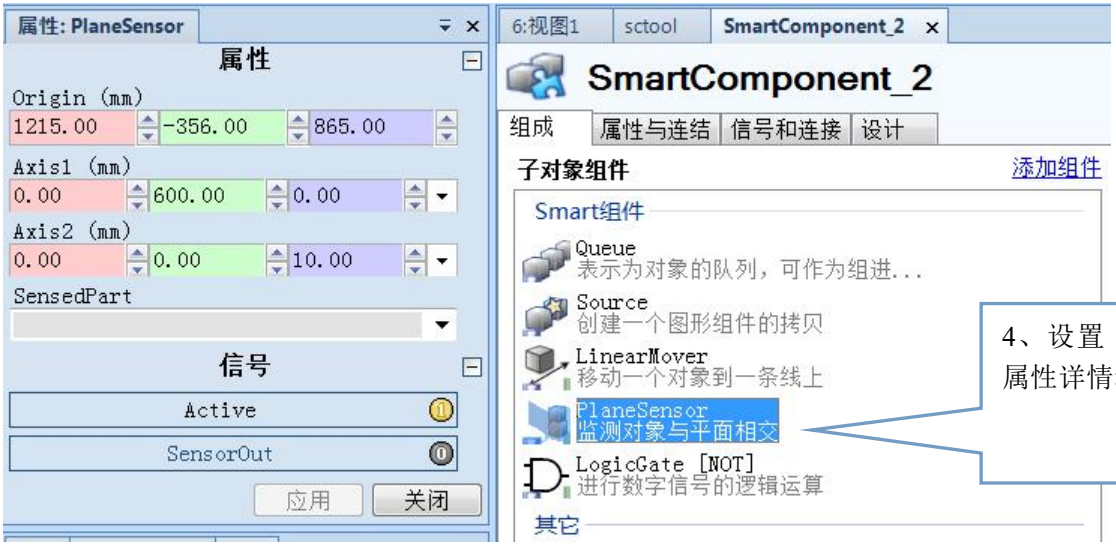
7、进行 I/O 的连接

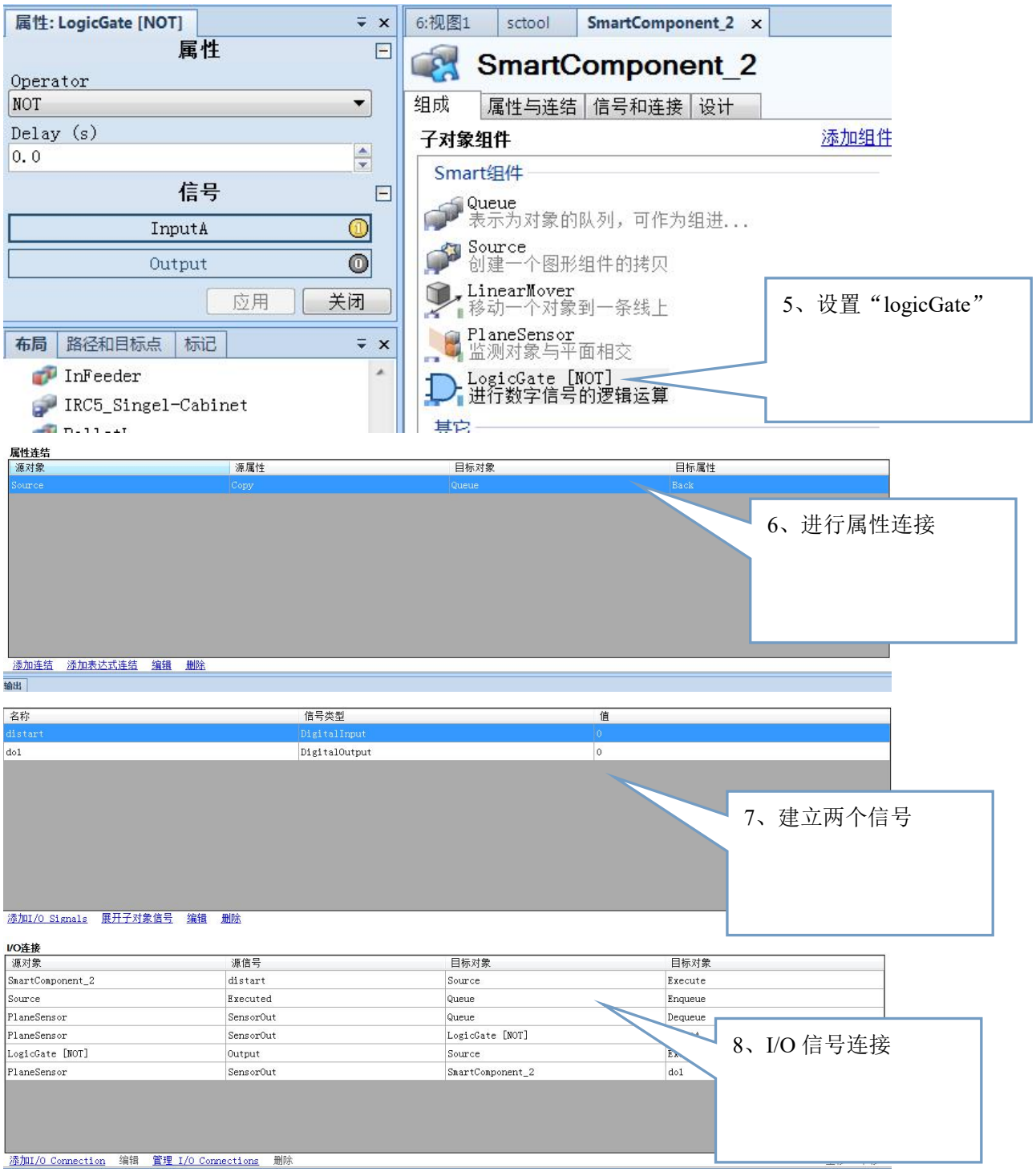


1、添加左边的五个组件


属性: Source
属性
Source
SmartComponent_2/Product_Source
Copy
Parent
Position (mm)
0.00
0.00
0.00
Orientation (deg)
0.00
0.00
0.00
Transient
信号
Execute
应用
关闭
6:视图1
sctool
SmartComponent_2
SmartComponent_2
组成
属性与连结
信号和连接
设计
子对象组件
添加组件
Smart组件
Queue
表示为对象的队列，可作
Source
创建一个图形组件的拷贝
LinearMover
移动一个对象到一条线上
PlaneSensor
监测对象与平面相交
LogicGate [NOT]
进行数字信号的逻辑运算
其它
Product_Source
2、设置“source”属性


属性: LinearMover
属性
Object
SmartComponent_2/Queue
Direction (mm)
-1.00
0.00
0.00
Speed (mm/s)
500.00
Reference
Global
信号
Execute
应用
关闭
布局
路径和目标点
标记
InFeeder
6:视图1
sctool
SmartComponent_2
SmartComponent_2
组成
属性与连结
信号和连接
设计
子对象组件
添加组件
Smart组件
Queue
表示为对象的队列，可作为组进...
Source
创建一个图形组件的拷贝
LinearMover
移动一个对象到一条线上
PlaneSensor
监测对象与平面相交
LogicGate [NOT]
进行数字信号的逻辑运算
其它
Product_Source
3、设置“LinearMover”属性

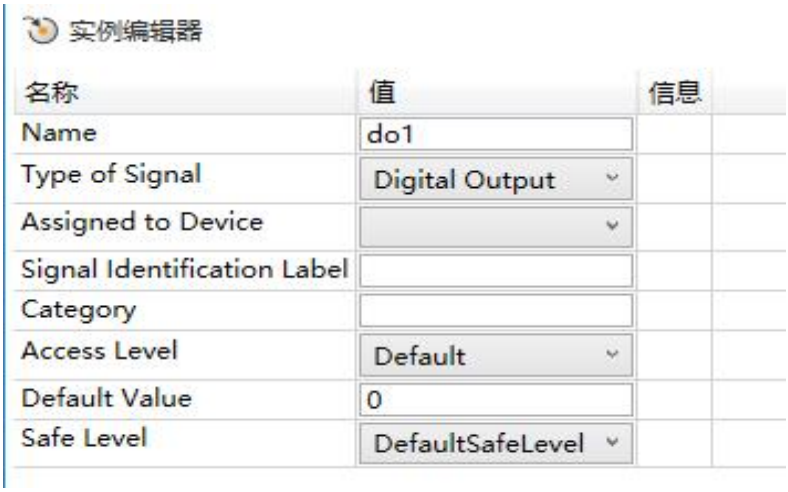

属性: PlaneSensor
属性
Origin (mm)
1215.00
-356.00
865.00
Axis1 (mm)
0.00
600.00
0.00
Axis2 (mm)
0.00
0.00
10.00
SensedPart
信号
Active
SensorOut
应用
关闭
6:视图1
sctool
SmartComponent_2
SmartComponent_2
组成
属性与连结
信号和连接
设计
子对象组件
添加组件
Smart组件
Queue
表示为对象的队列，可作为组进...
Source
创建一个图形组件的拷贝
LinearMover
移动一个对象到一条线上
PlaneSensor
监测对象与平面相交
LogicGate [NOT]
进行数字信号的逻辑运算
其它
4、设置“planeSenor”属性详情看视频



机器人 IO 的设定

为了实现真夹具动作的夹/放的动作控制，为了至少需要设定一个虚拟的数字输出信号，这个信号只用于虚拟仿真的作用，并没有与实际的总线或 IO 板进行关联。

数字输出信号的设定菜单操作为：控制器---配置编辑器---IO SYSTEM---SIGNAL。然后将信号设定为以下的表 1 的参数：

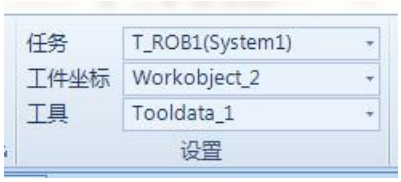


机器人轨迹的创建

机器人的动作是从左侧的码垛盘存放处搬运到右边的的方形的码垛盘处。

具体的操作方法如下：

1、设置正确的工件坐标与工具，如下图所示：



2、根据具体的情况，设定正确的机器人运动指令的参数，如下图所示：

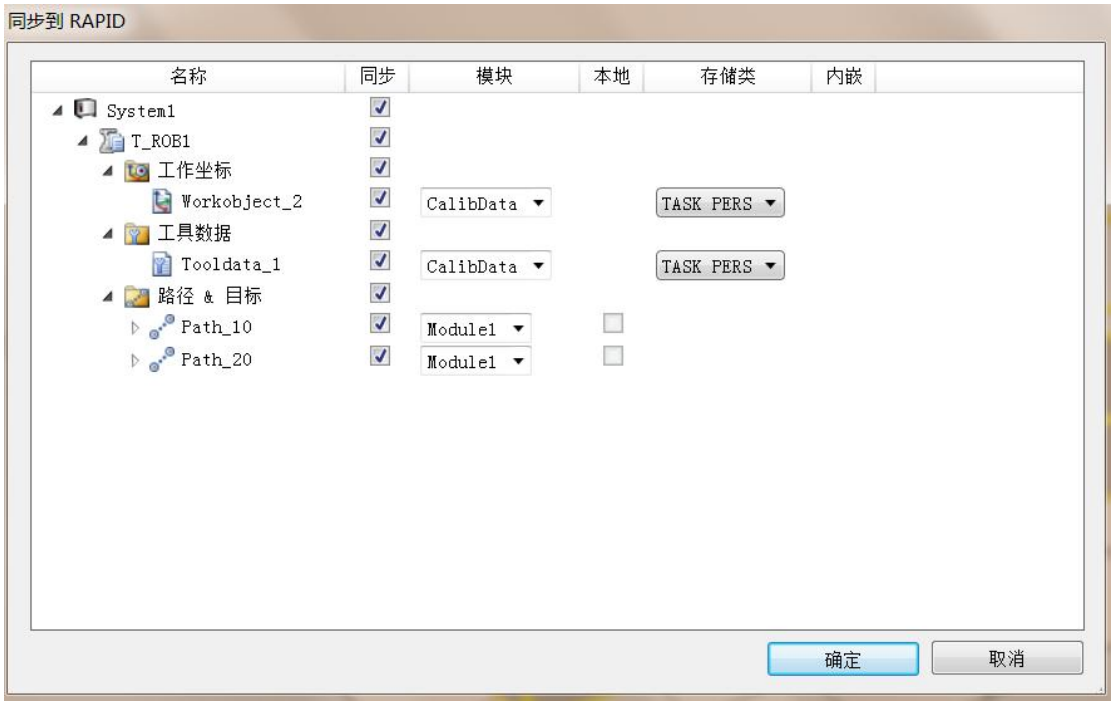
3、根据动作的要求通过示教指令的方法，创建对应的轨迹程序，程序样板如下图所示：

```
MODULE Module1
CONST robtarget
phome:=[[2125.775,375,1260.000193412],[0,0,1,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,
9E9]];
CONST robtarget
pick12:=[[2106.435808777,354.734330409,857.782503715],[0,0,1,0],[-1,0,-1,0],[9E9,
9E9,9E9,9E9,9E9,9E9]];
CONST robtarget
pick:=[[2106.434497406,354.734348298,523.967107723],[0,0,1,0],[-1,0,-1,0],[9E9,9
E9,9E9,9E9,9E9,9E9]];
CONST robtarget
pick13:=[[1360.001770503,1627.450977251,857.781278091],[0,0,1,0],[0,0,0,0],[9E9,
9E9,9E9,9E9,9E9,9E9]];
CONST robtarget
pl1:=[[276.898159785,2244.831490724,96.768295443],[0,0,1,0],[1,0,1,0],[9E9,9E9,9
E9,9E9,9E9,9E9]];
```

```
    CONST robtarget
Target_10:=[[276.897645854,2244.834285225,429.563668441],[0,0,1,0],[1,0,1,0],[9
E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget
Target_20:=[[276.895523852,1686.627880885,395.231541888],[0,0,1,0],[1,0,1,0],[9
E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget
pl2:=[[276.896517039,1686.624092663,96.768037259],[0,0,1,0],[1,0,1,0],[9E9,9E9,9
E9,9E9,9E9,9E9]];
    CONST robtarget
Target_30:=[[881.021938231,2243.945752607,369.207529123],[0,0,1,0],[0,0,0,0],[9
E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget
pl3:=[[881.021349878,2243.941527394,96.768175471],[0,0,1,0],[0,0,0,0],[9E9,9E9,9
E9,9E9,9E9,9E9]];
    CONST robtarget
Target_40:=[[881.022529491,1686.630605361,376.803148395],[0,0,1,0],[0,0,0,0],[9
E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget
pl4:=[[881.021746601,1686.626659644,96.767977503],[0,0,1,0],[0,0,0,0],[9E9,9E9,9
E9,9E9,9E9,9E9]];
    PROC Path_10()
        MoveJ phome,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        WaitTime 100;
    ENDPROC
    PROC Path_20()
        Set do1s;
        Reset do1s;
        MoveJ phome,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        WaitDI dipos,1;
        MoveJ pick,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Set do2;
        WaitTime 0.3;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick13,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ Target_10,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pl1,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Reset do2;
        WaitTime 0.3;
        MoveJ Target_10,v1000,z100,Tooldata_1\WObj:=Workobject_2;
       MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
       WaitDI dipos,1;
       MoveJ pick,v1000,fine,Tooldata_1\WObj:=Workobject_2;
```

```
        Set do2;
        WaitTime 0.3;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick13,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ Target_20,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveL pl2,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Reset do2;
        WaitTime 0.3;
        MoveJ Target_20,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        WaitDI dipos,1;
        MoveJ pick,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Set do2;
        WaitTime 0.3;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick13,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ Target_30,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveL pl3,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Reset do2;
        WaitTime 0.3;
        MoveJ Target_30,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        WaitDI dipos,1;
        MoveJ pick,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Set do2;
        WaitTime 0.3;
        MoveJ pick12,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ pick13,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveJ Target_40,v1000,z100,Tooldata_1\WObj:=Workobject_2;
        MoveL pl4,v1000,fine,Tooldata_1\WObj:=Workobject_2;
        Reset do2;
        WaitTime 0.3;
        MoveJ Target_40,v1000,z100,Tooldata_1\WObj:=Workobject_2;
    ENDPROC
ENDMODULE
```

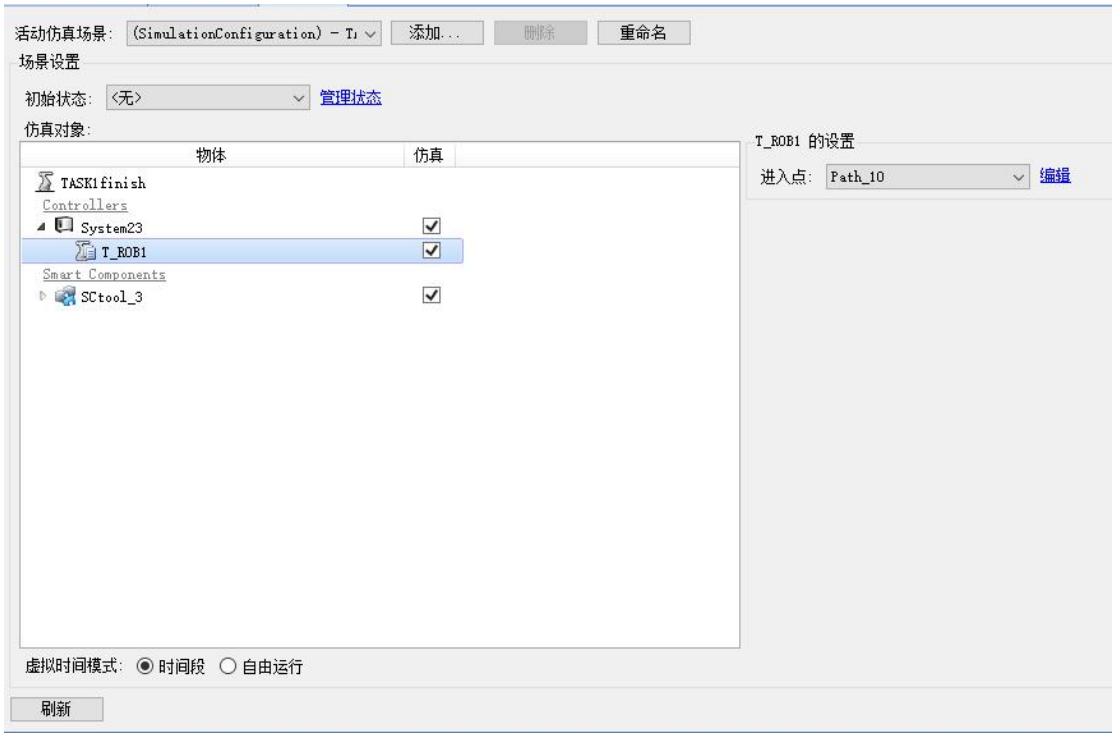
将写好的程序同步到RAPID，菜单操作：基本---同步---同步到RAPID，如下图所示：



仿真的调试

在完成了设置与编程以后，先保存初始状态，接着下来就是要验证一下仿真动画的结果了，具体的操作如下：

1、设定要运行的 RAPID 子程序，在本项目中是 PATH20，菜单操作如下：仿真---仿真设定---指定 PATH2 0，如下图所示：



2、点击仿真菜单中的“播放”就可以看到动画效果了。动画结束后，点击“重置”，恢复到原来的状态。