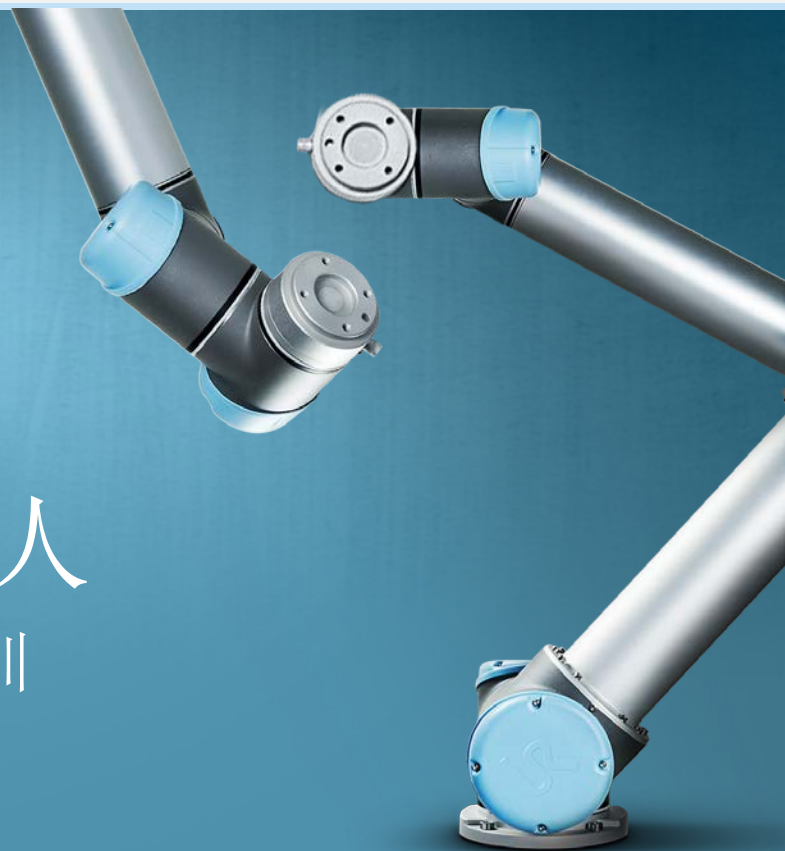




UNIVERSAL ROBOTS

# 优傲机器人

## 高级课程培训





# 欢迎到高级课程培训

- 实用的动手经验
- USB盘和模拟软件
- 培训结束后测试



## 培训老师介绍:

姓名:

职务:



1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

10

故障处理流程



1 脚本代码

2 变量

3 特征

4 高级TCP使用

5 Modbus 服务器

6 FTP 服务器

7 Dashboard 服务器

8 客户端界面

9 Socket 通讯

10 故障处理流程



## 什么是脚本代码?

- 脚本代码
  - 优越开发的高水平脚本语言
  - 能替代Polyscope GUI建立程序结构
  - 类似于Python 脚本语言
  - 《脚本手册》包含所有脚本代码的定义
  - Polyscope 程序在执行前可以转化为脚本代码



**UNIVERSAL  
ROBOTS**

The URScript Programming Language

Version 3.2  
September 28, 2015



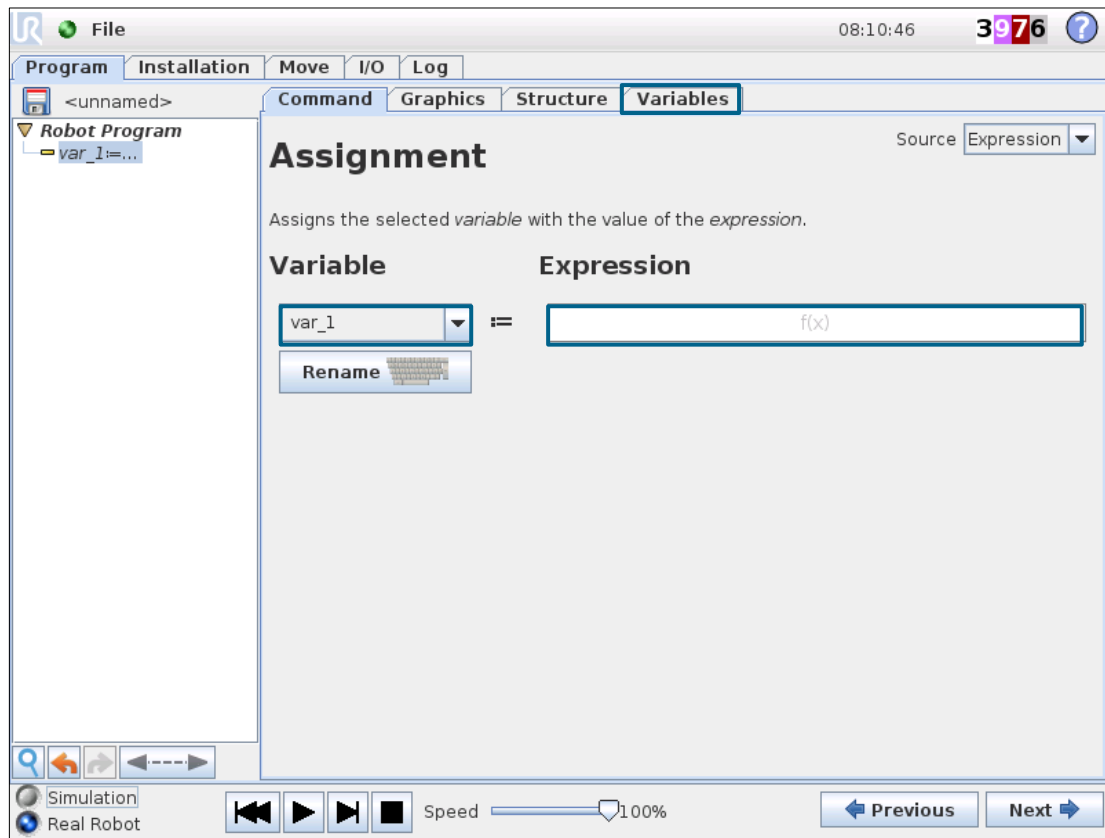
## 如何用脚本代码

- 用脚本代码有几个方法
  - 赋值
  - 脚本代码——线性
  - 脚本代码——文件
  - 调用功能
  - 客户端界面



## 赋值

- 执行脚本指令
- 给一个变量返回值
- 允许返回值在变量窗口可见







## 表达式编辑器

- 常用脚本指令列表
- 保存全部指令
- 用力（force）指令做一个简单程序

Robot Program

MoveL

Waypoint\_1



IF force() &lt; 30

Waypoint\_2

Expression Editor Interface:

- Top bar: Keyboard icon, Green text input field, "<<" button.
- Operator Grid:
  - Logical: and, or, xor, not
  - Comparison: <, >, =, !=, >=, <=
  - Boolean: True (HI), False (LO)
- Dropdowns: <Input>, <Output>, <Variable>, <Pose>, <Function>
- Numeric Keypad: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, \*, /, (, ), =, <, >, <<, >>, and a green checkmark button.
- Bottom: Shift button, Large empty text field, Red X button.

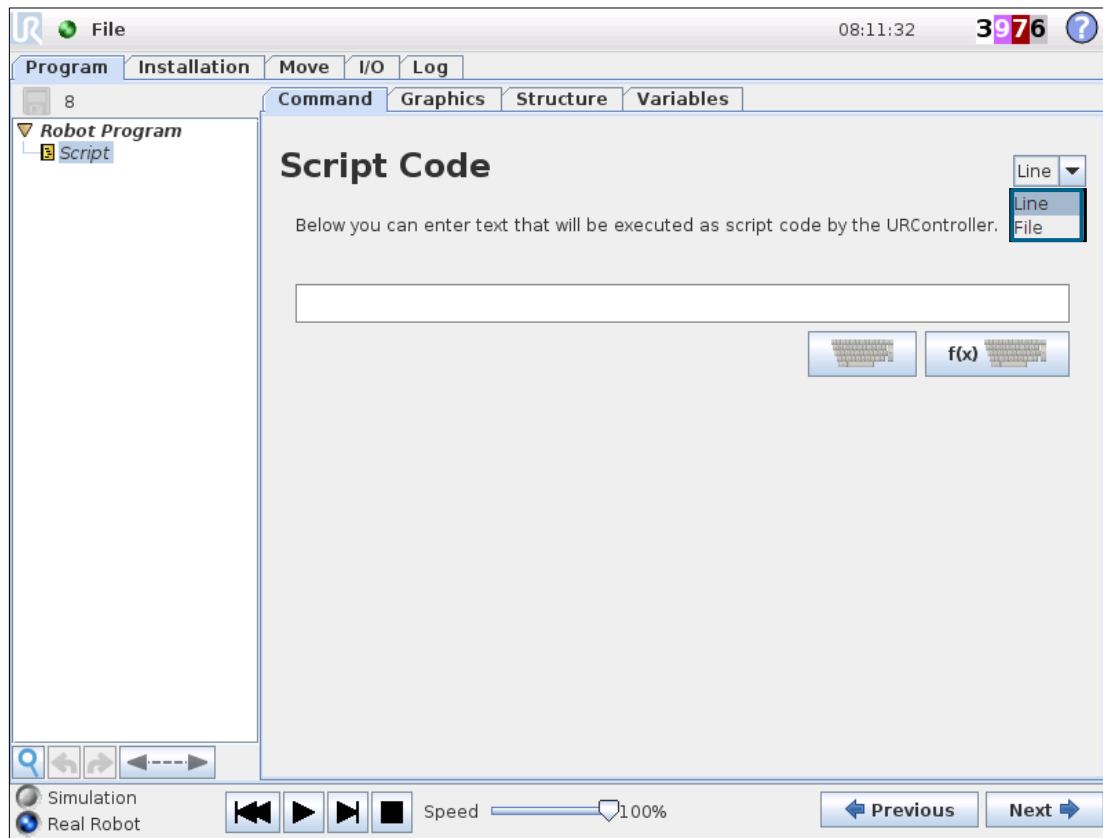


## 脚本代码——线性

- 执行单独线性的脚本代码
  - 在脚本中定义的变量不显示在变量窗口
  - 在脚本中定义的路点不显示在图形窗口

Robot Program

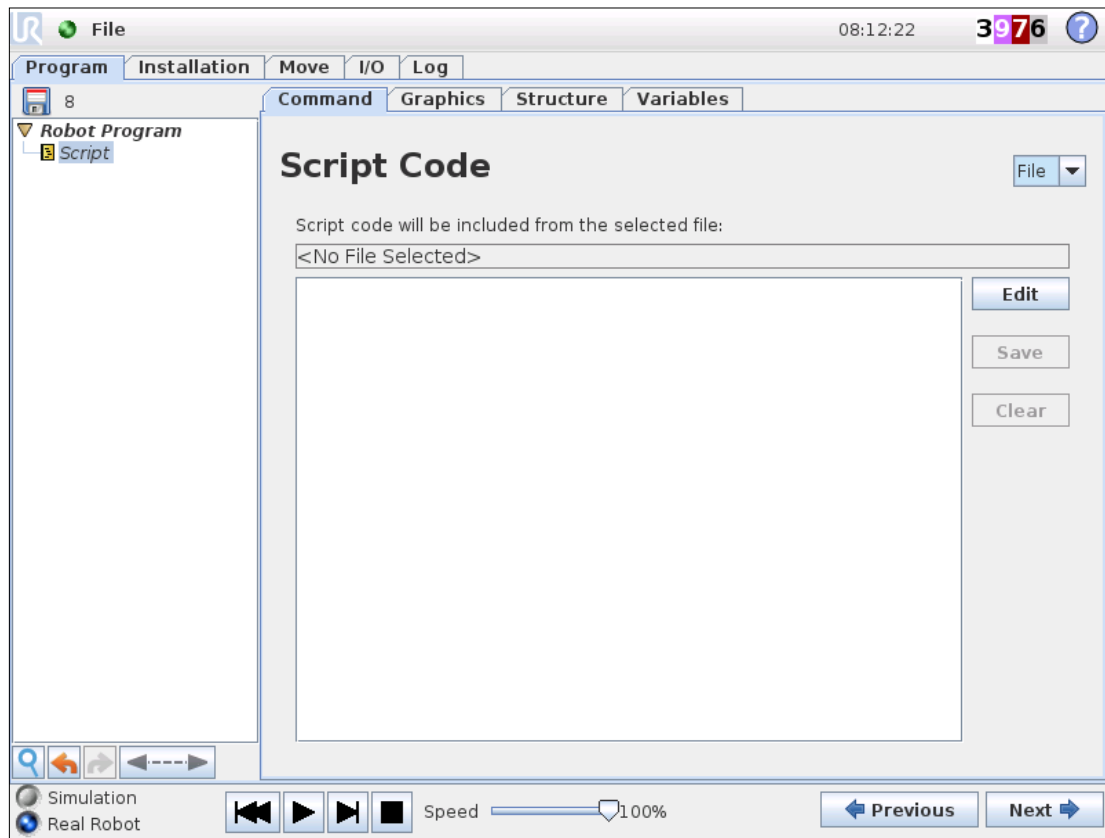
```
movel(p[0,-0.4,0.3,0,3.14,0])  
sleep(1)  
movel(p[0.2,-0.4,0.3,0,3.14,0])
```





## 脚本代码——文件

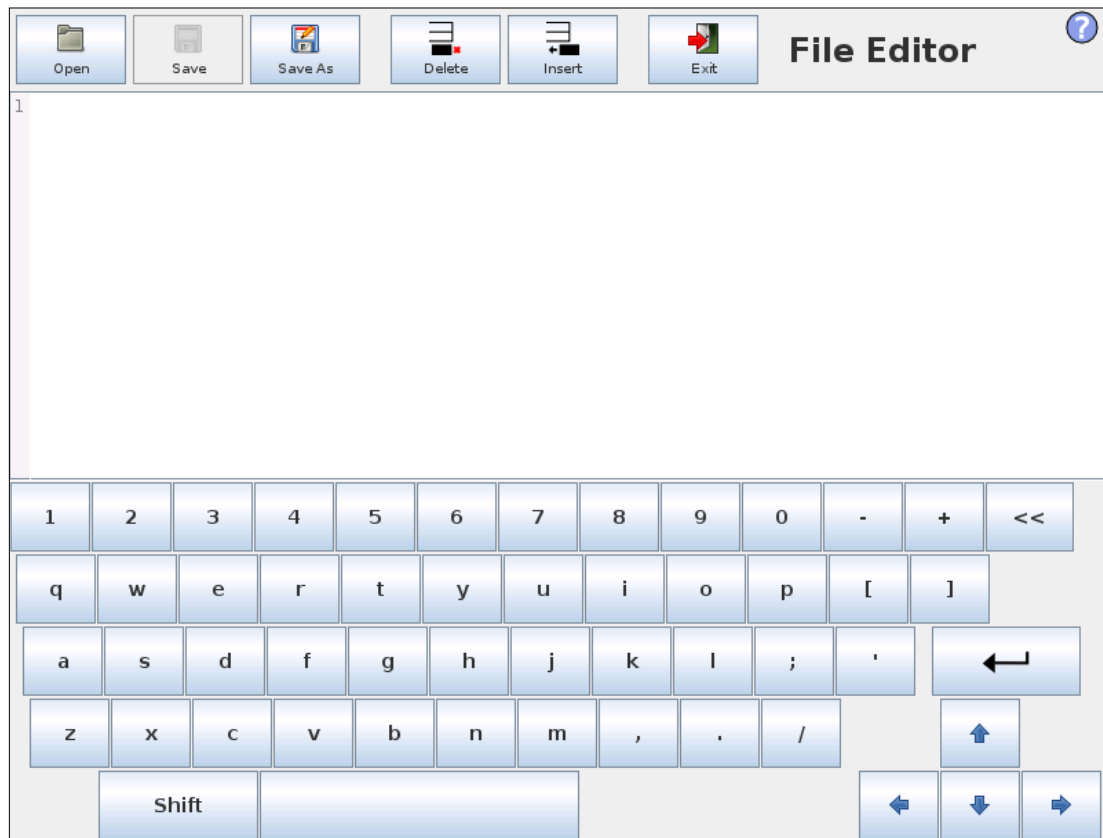
- 导入并运行现存的脚本文件
- 执行多行脚本程序
- 通过以太网的文件传输在今后的培训中包含
- 选择“编辑”按钮





## 脚本代码——文件

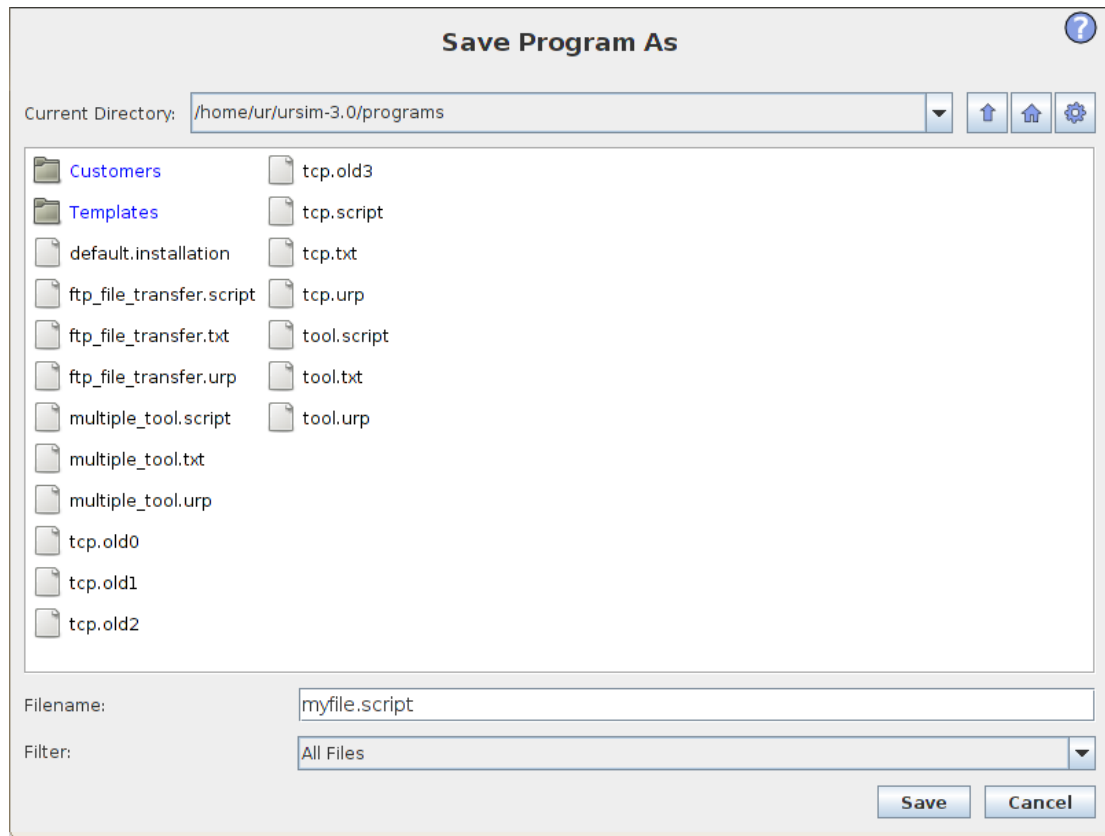
- 选择新文件





## 脚本代码——文件

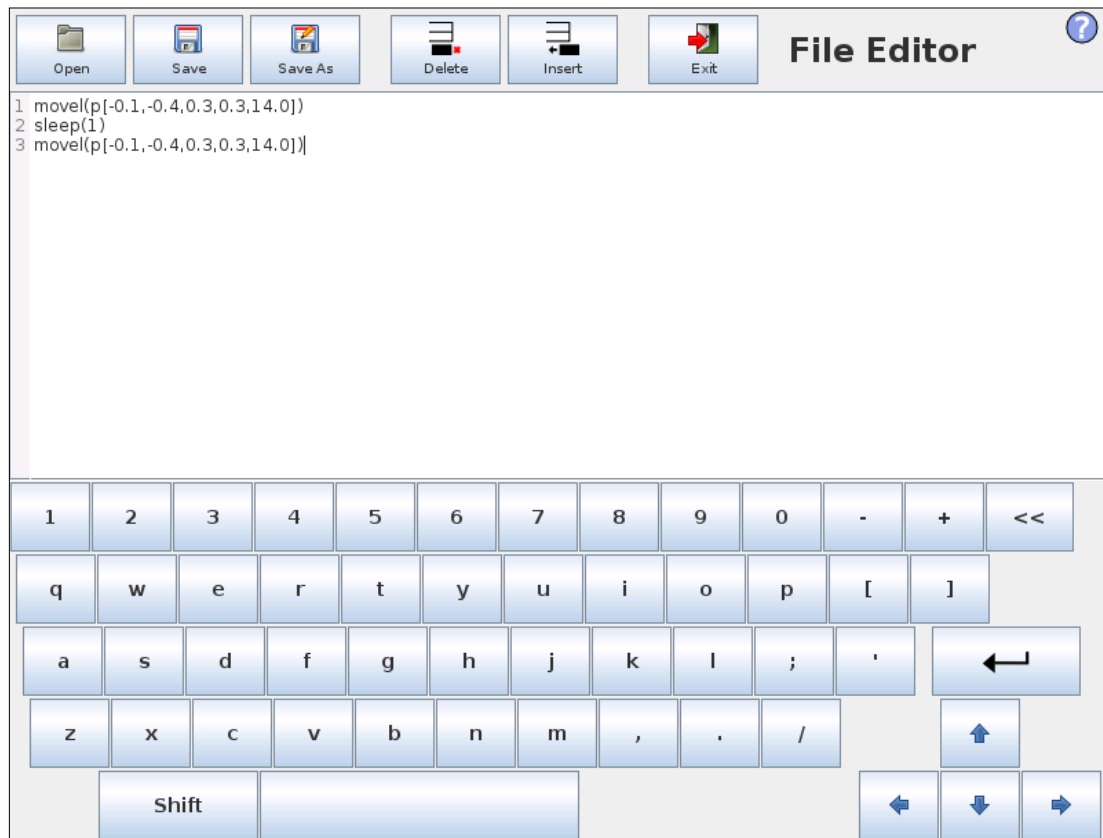
- 在文件类型下拉菜单中选择所有文件
- 写一个名字并用 .script 作为扩展名
- 保存文件





## 脚本代码——文件

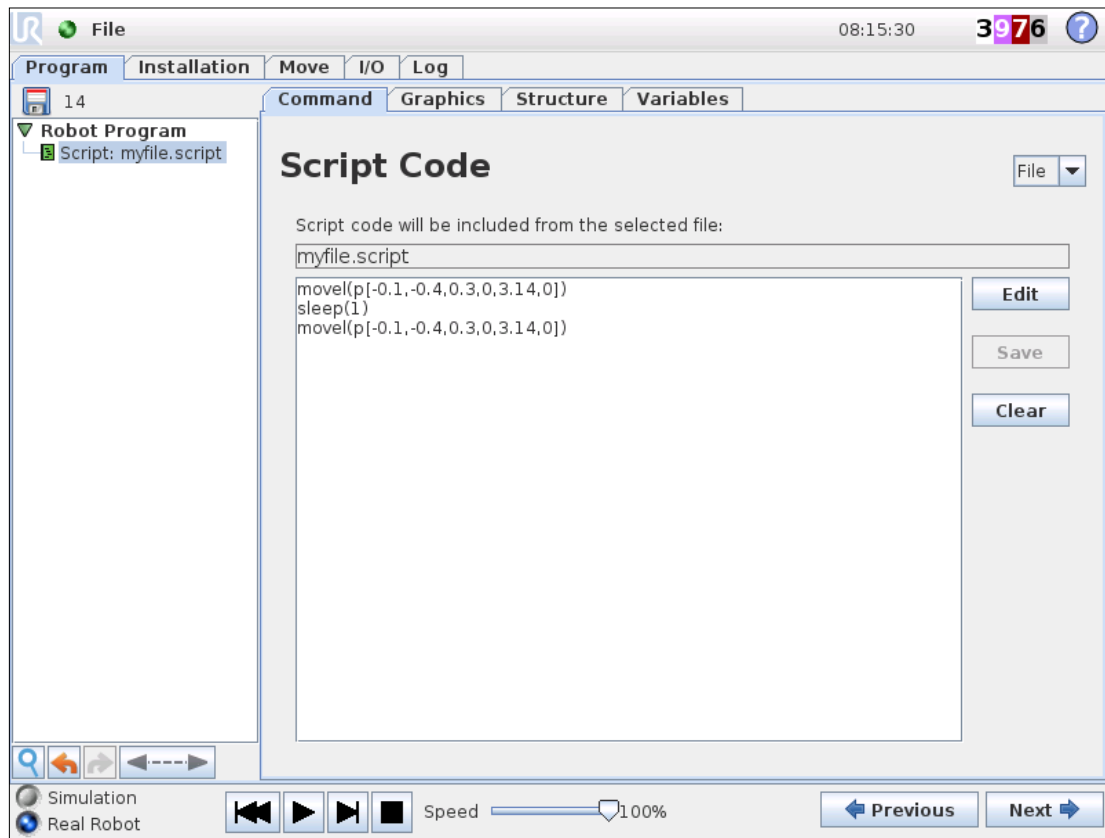
- 插入3行
- 每一行写入脚本代码
- 保存和退出





## 脚本代码——文件

- 运行程序





## 脚本代码——函数

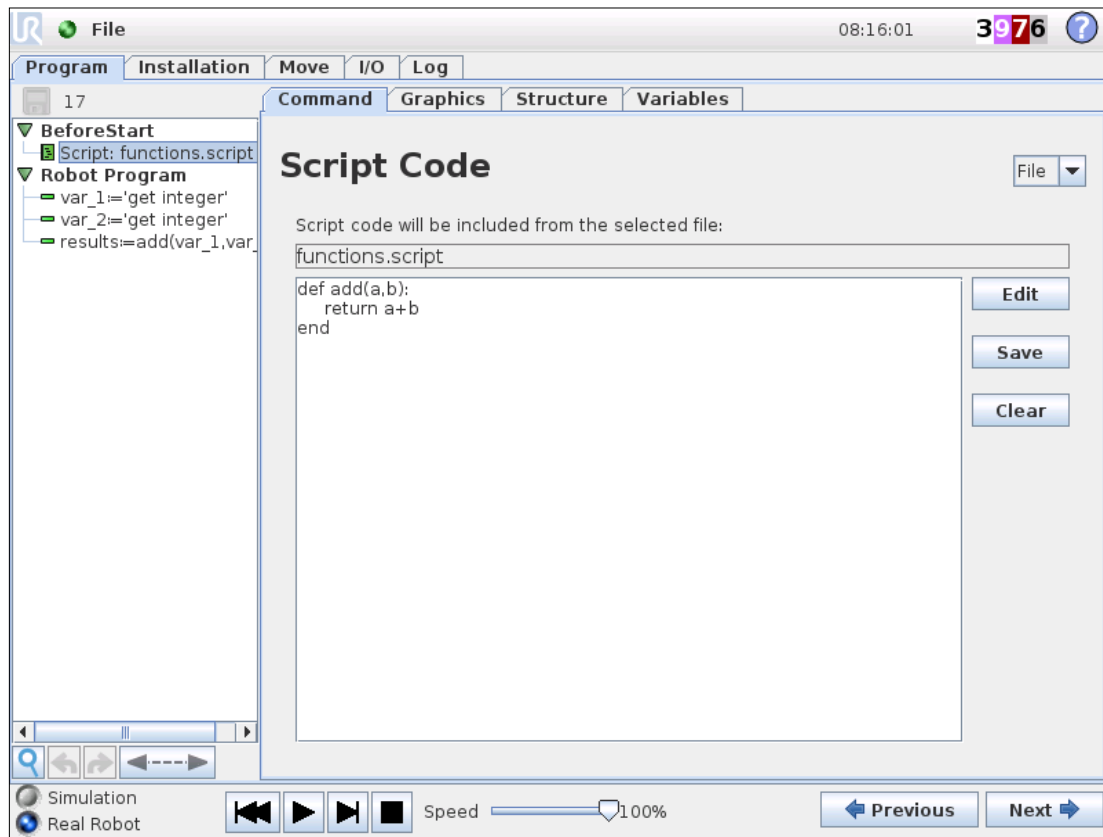
- 在脚本中函数是这样表示
  - `def name()`
    - `script`
    - `...`
  - `end`
- 函数能带参数和返回值
  - `def name(argument)`
    - `...`
    - `...`
    - `return`
  - `End`
- 在程序中同一个函数可以调用多次





## 脚本代码——函数

- 把两个数相加并返回结果的函数
  - 用函数创建一个脚本文件
    - 给函数一个名字
    - 定义两个参数
    - 插入返回值
- 由操作者输入两个变量
- 用赋值指令表明变量和函数





## 客户端界面

- 脚本代码可以由外部设备通过以太网直接发送给机器人控制箱
- 在机器人中自动运行3个客户端界面
- 后面解释客户端界面部分内容



## 培训练习

- 写一个简单的脚本代码文件
  - 定义一个函数
  - 接收一个参数
  - 执行接收值
  - 返回运行结果
- 在程序中加载这个文件
- 从赋值指令调用函数
- 执行程序
- 查看结果



1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

10

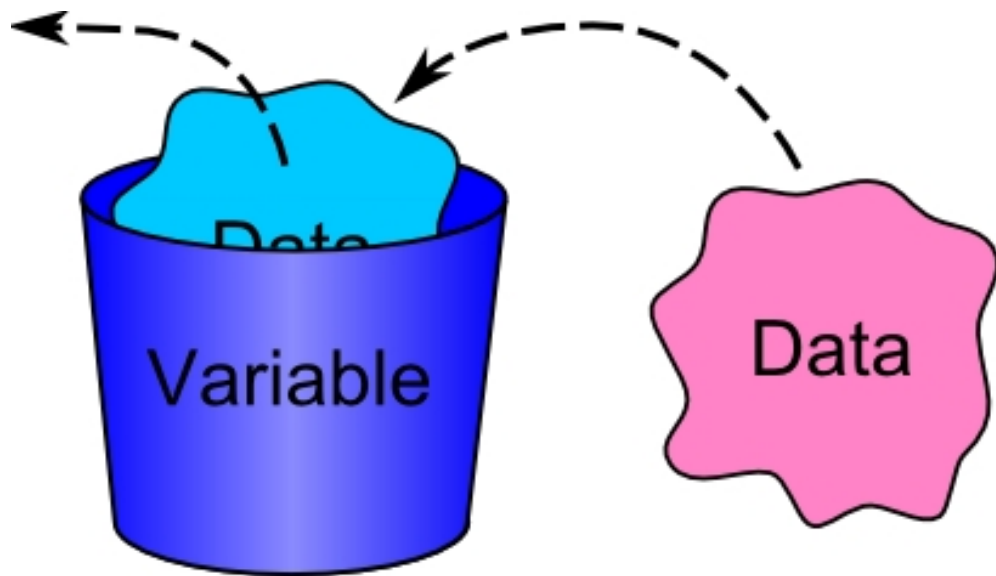
故障处理流程



## 什么是变量

- 变量是一个存储位置（容器）
  - 容器内的内容可以改变
- 变量可读写
  - 值能被覆盖
  - 值能被读
- 变量可以和其他变量或信号状态相比较

基本培训总结





## 变量类型

## 基本培训总结

变量类型	值
布尔变量 ( <code>boolean</code> )	true/false
整数变量 ( <code>integer</code> )	整数 (32 bit)
浮点变量 ( <code>floating point</code> )	实数 (带小数点)
字符串 ( <code>string</code> )	文本 (ASCII 码)
位姿变量 ( <code>pose</code> )	位姿变量p[x,y,z,rx,ry,rz]
数组 ( <code>list</code> )	变量排列组合



## 变量类型

## 基础培训总结

范围	位置
局部	在程序中
全局	在安装设置中

- 局部变量
  - 在程序中定义
  - 在同一个程序中可接近
  - 关机后数值被清除
- 全局变量
  - 在安装设置中定义
  - 用同样安装设置文件的所有程序中可接近
  - 数值保存在CF卡的文件中

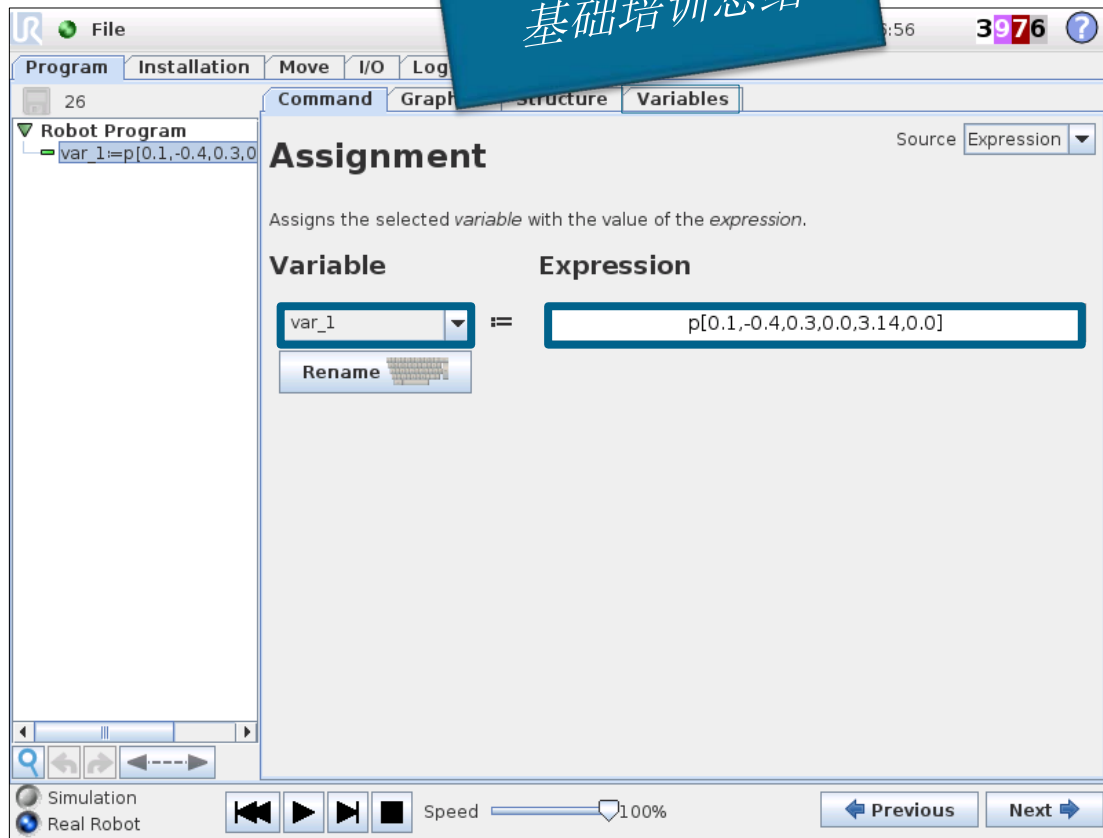


## 如何在程序中插入一个变量

- 赋值指令
  - 定义变量名字
  - 声明变量类型
  - 给变量赋值

Robot Program  
var\_1 = True  
Wait 0.5  
var\_1 = False  
Wait 0.5

基础培训总结







## 位姿变量

- 定义

- 位姿变量是在笛卡尔坐标系下对路点的位置和方向的矢量描述
- 包含：
  - 位置矢量 (x,y,z)
  - 选择矢量 (rx,ry,rz)

`pose_var` = `p[ x , y , z , rx , ry , rz ]`

变量名字

TCP 位置

旋转矢量

- 单位

- x,y,z = 米
- rx,ry,rz = 弧度



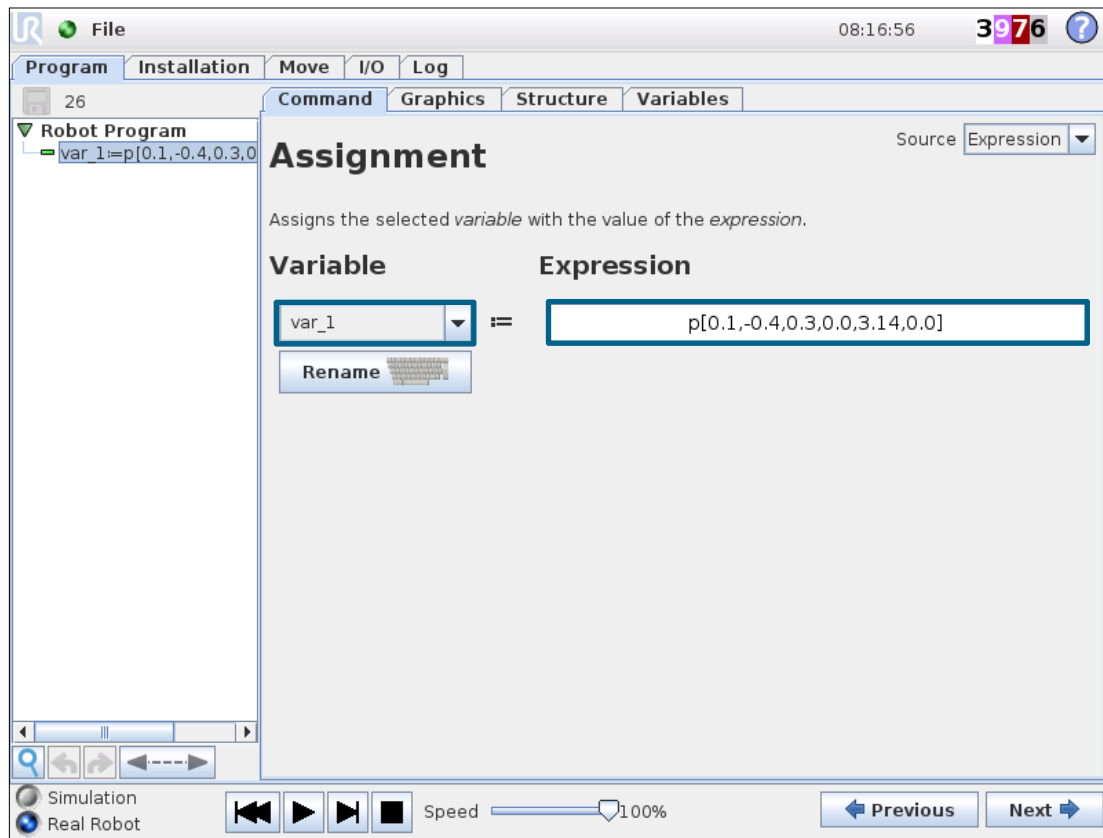
## 使用位姿变量

- 手动输入位姿坐标

- 插入赋值指令
- 定义变量名字
- 用位姿输入表达式

例如：

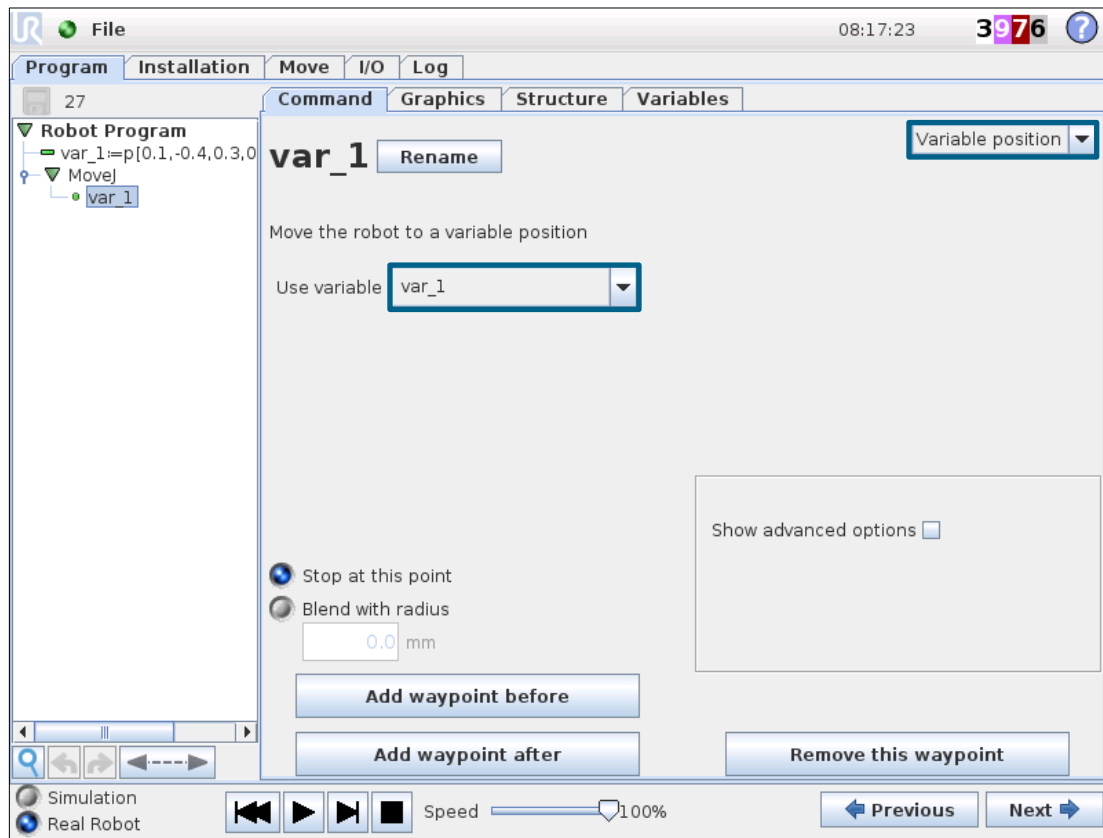
- x = 100 mm
- y = -400 mm
- z = 300 mm
- rx = 0
- ry = 3.14
- rz = 0





## 使用位姿变量

- 运动到位置点
  - 插入移动指令
  - 选择可变路点
  - 使用变量



- 注意：如果初始点为可变或相对路点，将跳过用“自动”回到初始点的操作



## 用于位姿变量的脚本函数

- 一些脚本函数对位姿变量是有效的：

脚本代码	功能
<code>get_actual_tcp_pose()</code>	返回当前得到的TCP位置坐标
<code>get_actual_tcp_speed()</code>	返回当前TCP的速度值
<code>get_inverse_kin()</code>	反向运动方程
<code>get_target_tcp_pose()</code>	返回当前目标TCP位置坐标
<code>get_target_tcp_speed()</code>	返回当前目标TCP位置速度
<code>interpolate_pose(p_from, p_to, alpha)</code>	工具位置和方向的线性插补
<code>pose_add(p_1, p_2)</code>	p_1和p_2的位置相加
<code>pose_dist(p_from, p_to)</code>	返回两点之间的距离
<code>pose_inv(p_from)</code>	得到位置点的相反数
<code>pose_sub(p_to, p_from)</code>	位置点相减
<code>pose_trans(p_from, p_to)</code>	位置点转化



### 例子: `get_actual_tcp_pose()`

- 如何读当前位置点并且运动到安全位置
  - 添加 **BeforeStart** 序列
  - 读当前位置点, 保存在变量中
  - 在另一个变量中保存Z—方向值
  - 定义新的位姿变量
    - 除Z值外, 其他值取当前点相同
    - 设Z值到 400 mm.
  - 运动到 *safe\_pos* 可变路点
    - 减少速度

#### BeforeStartSequence

```
cp = get_actual_tcp_pose()
z = cp[2]
safe_pos = p[cp[0], cp[1], z+0.4, cp[3], cp[4], cp[5]]
Wait 1
MoveL
    safe_pos
Robot Program
Halt
```

#### 位姿变量指定位置

p[x,y,z,rx,ry,rz]	索引号
x	[0]
y	[1]
z	[2]
rx	[3]
ry	[4]
rz	[5]



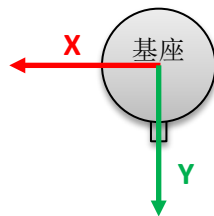
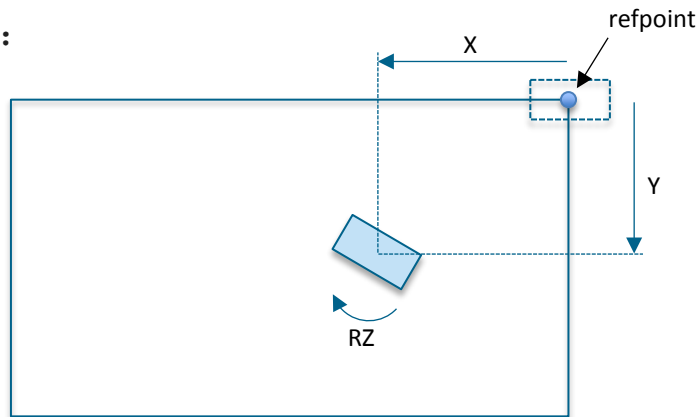
## 例子: `pose_add()`

- 如何运动机器人从参考点到可变的抓取点
  - 模拟来自视觉系统的数据
  - 例如: 目标点 (**object**) 是对参考点 (**refpoint**) 的补偿:
    - $x = 250 \text{ mm}$
    - $y = 120 \text{ mm}$
    - $rz = 31 \text{ degrees}$
  - 定义包含补偿值的位姿变量
  - 用 `pose_add()` 增加变量到参考点 (**refpoint**)
    - `pose_add()` 用基座坐标系作为参考

### Robot Program

```
MoveJ
  refpoint
Wait DI[0] = True
Calc-folder
  x = 250
  y = 120
  rz = 31
  offset = p[(x/1000),(y/1000),0,0,0,d2r(rz)]
  pick_pos = pose_add(refp, offset)
MoveL
  pick_pos
```

### 基座坐标系作为参照坐标系

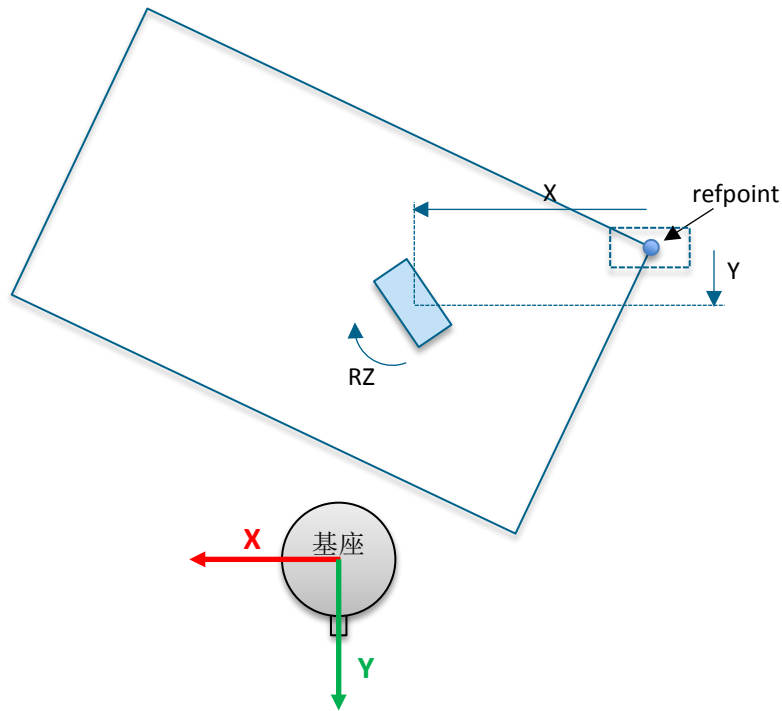




### 例子: `pose_trans()`

- 在真实应用中，一般不太可能视觉坐标系与基座坐标系一致
- 如果我们用 `pose_add()` 脚本函数，将很难到正确的目标点
- 我们需要转换X和Y到目标坐标系
- 这样就引入 `pose_trans()` 脚本函数

### 基座坐标系作为参考坐标系





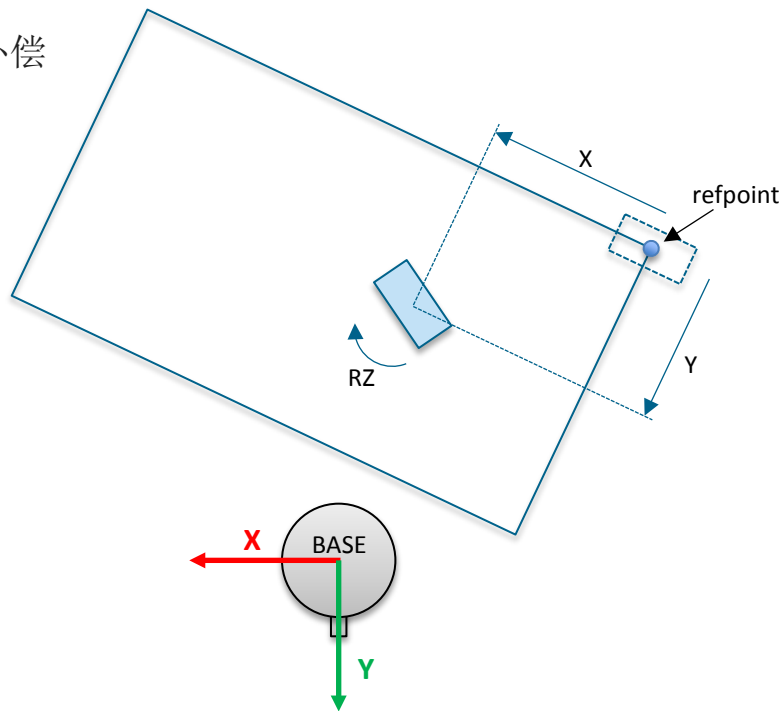
## 例子: `pose_trans()`

- 如何运动机器人从参考点到可变的抓取点
  - 来自视觉系统的模拟数据
  - 例如: 目标点 (object) 是对参考点 (refpoint) 的补偿
    - $x = 250 \text{ mm}$
    - $y = 120 \text{ mm}$
    - $rz = 31 \text{ degrees}$
  - 定义包含补偿值的位姿变量
  - 用 `pose_trans()` 增加变量到参考点 (refpoint)
    - `pose_trans()` 用第一个点作为参考

### Robot Program

```
MoveJ
  refpoint
Wait DI[0] = True
Calc-folder
  x = 250
  y = 120
  rz = 31
  offset = p[(x/1000),(y/1000),0,0,0,d2r(rz)]
  pick_pos = pose_trans(refp, offset)
MoveL
  pick_pos
```

## Refpoint as reference

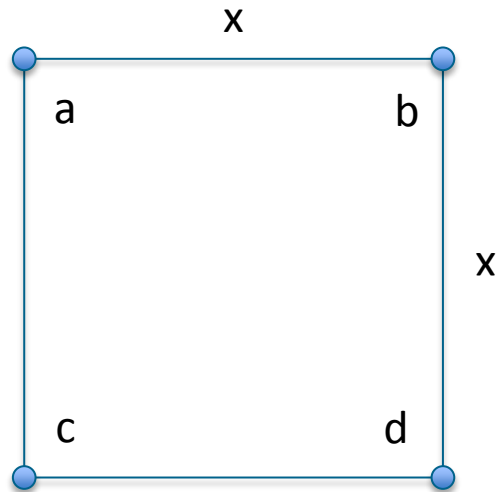






## 培训练习 1

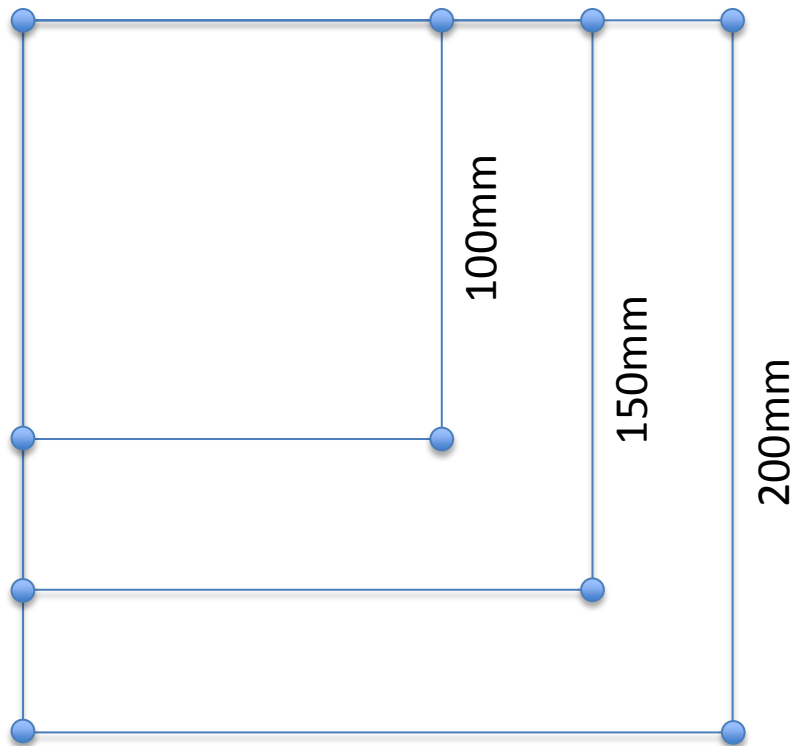
- 从机器人当前位置开始，画一个边长100mm的正方形
- 使用：
  - `get_actual_tcp_pose()`
  - `pose_trans()`
  - `MoveL`
- 编一个画正方形的程序
- 用变量 **x** 定义边长





## 培训练习 2

- 用练习1的程序，用同样的变量
- 画3个正方形
- 每个正方形边长增加50mm





1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

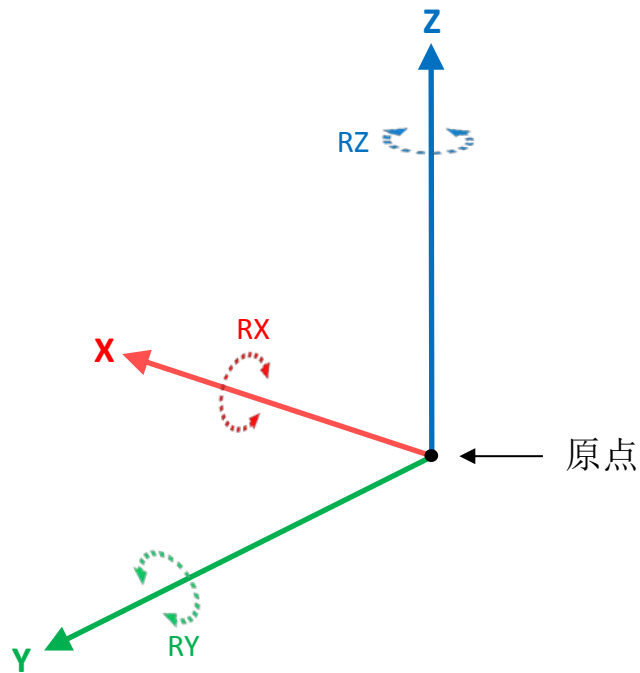
10

故障处理流程



## 什么是特征？

- 特征指的是笛卡尔坐标系
  - 坐标系是以某个固定点为原点定义的坐标平面和轴的方向

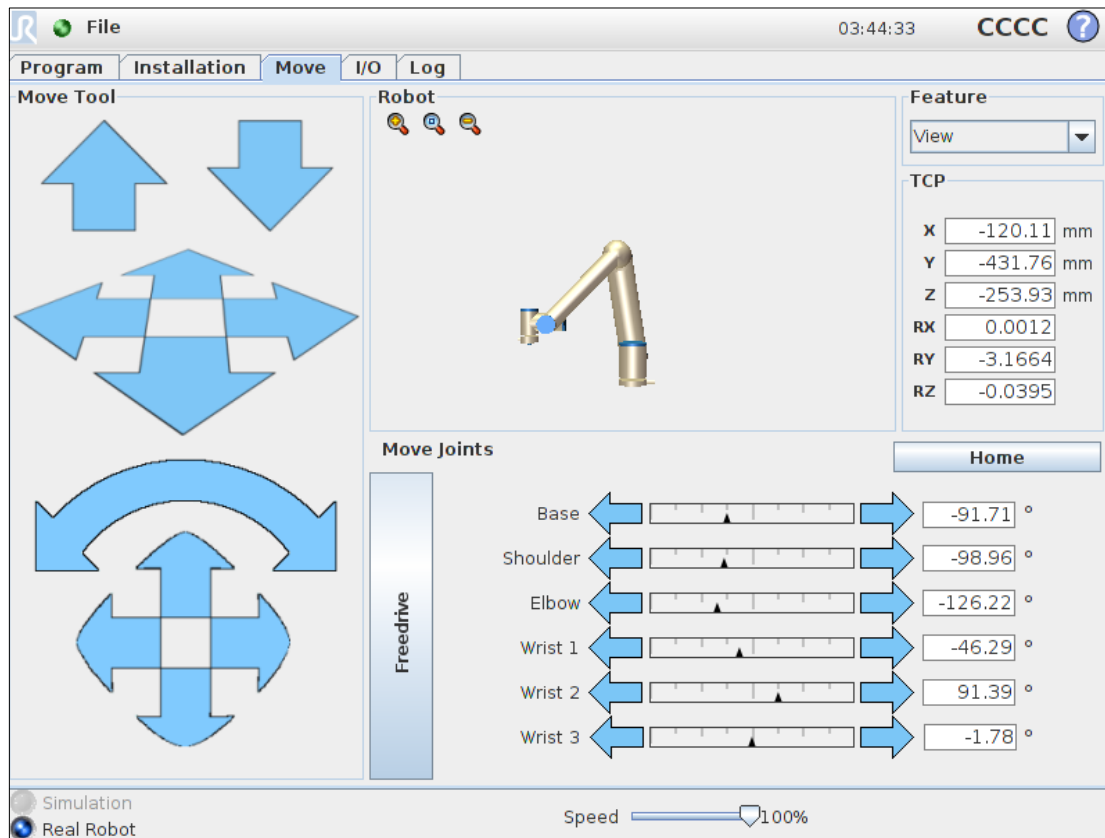


- 特征作为位姿变量保存



## 什么是特征?

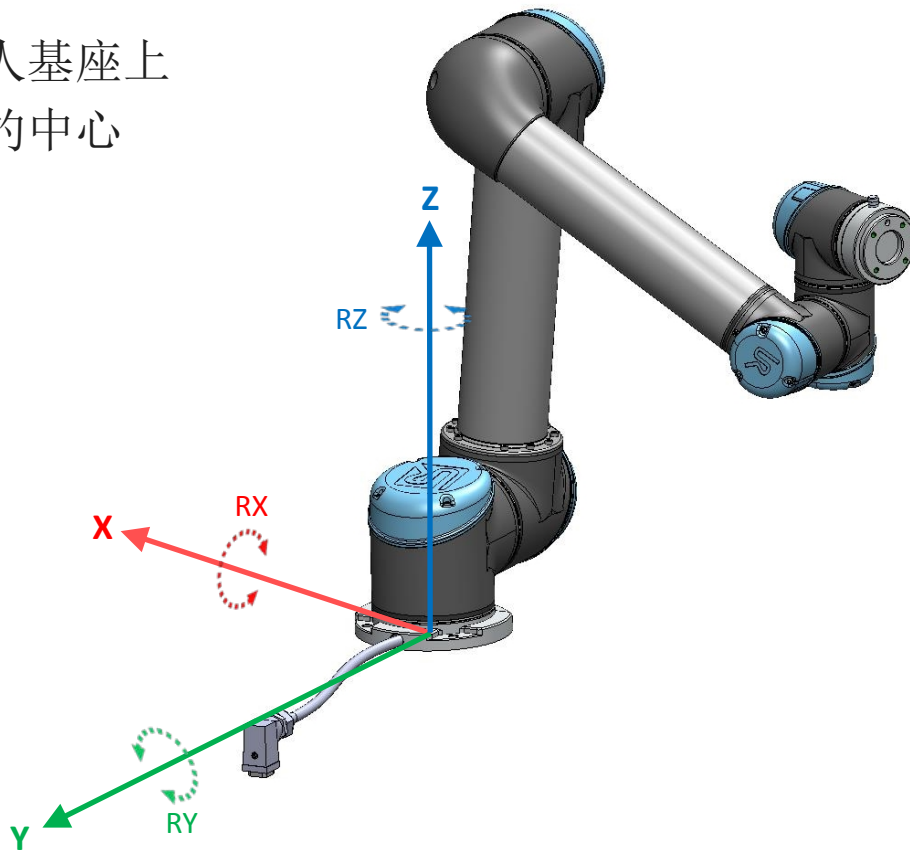
- 基于某个特征操作机器人
  - 基座
  - 工具
- 工具位置
  - X, Y, Z = TCP位置
    - 单位: mm / inch
  - Rx, Ry, Rz = TCP方向
    - 单位: 弧度
- 注意: 方向是旋转矢量, 矢量长度是旋转角的度数





## 基座特征

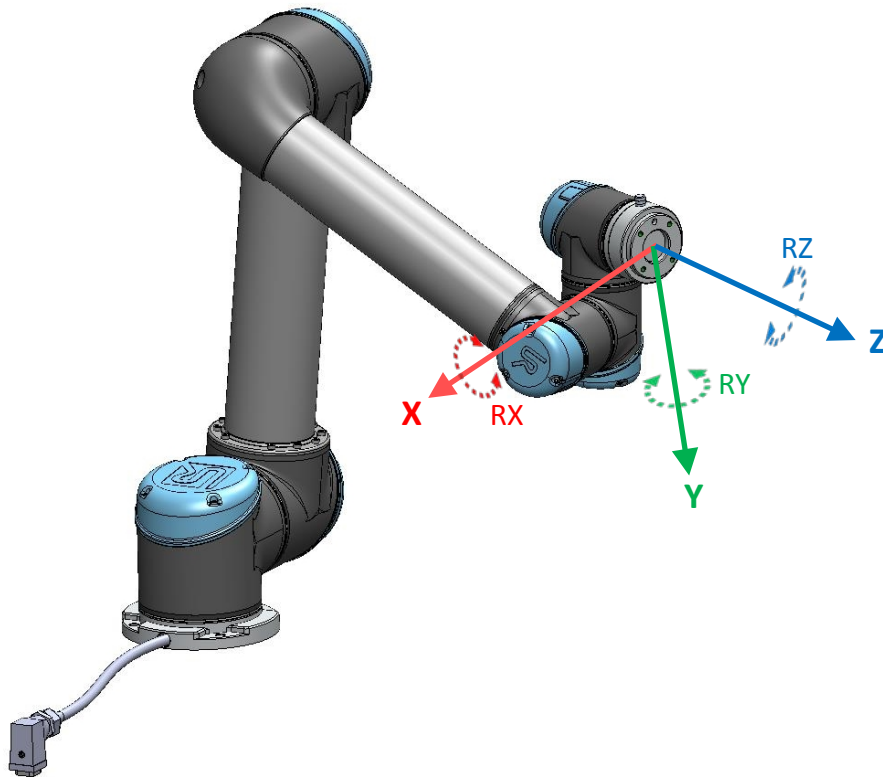
- 定义在机器人基座上
- 原点是基座的中心





## 工具特征

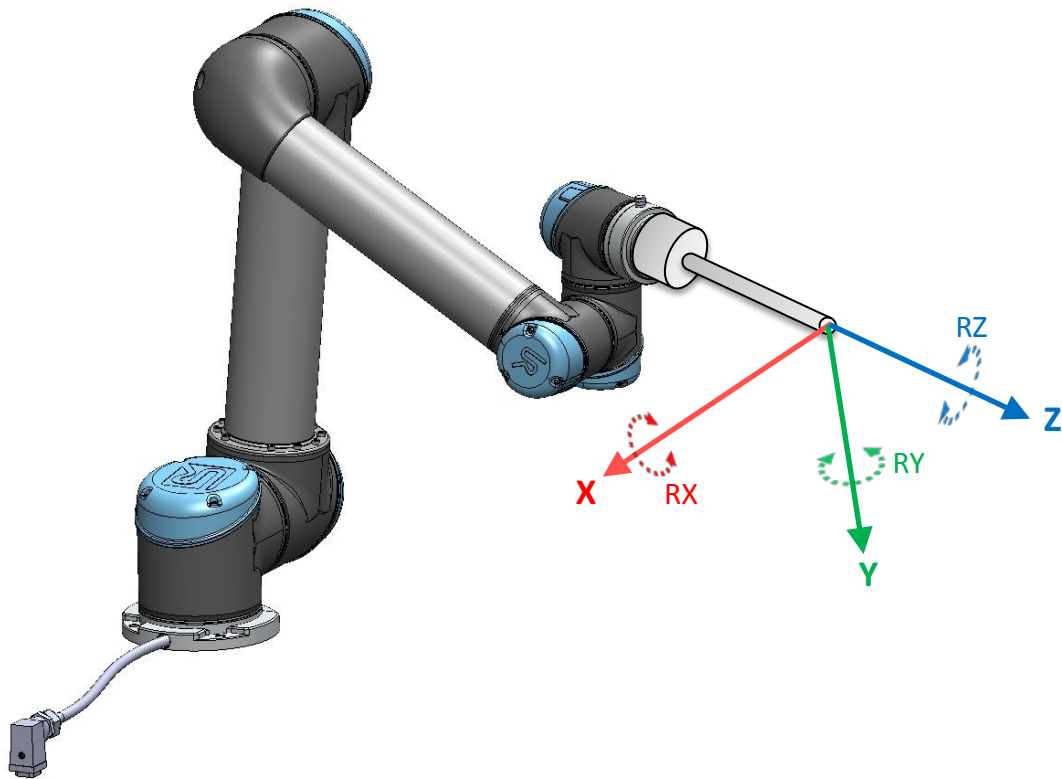
- 原点在工具法兰盘中心





## 工具特征

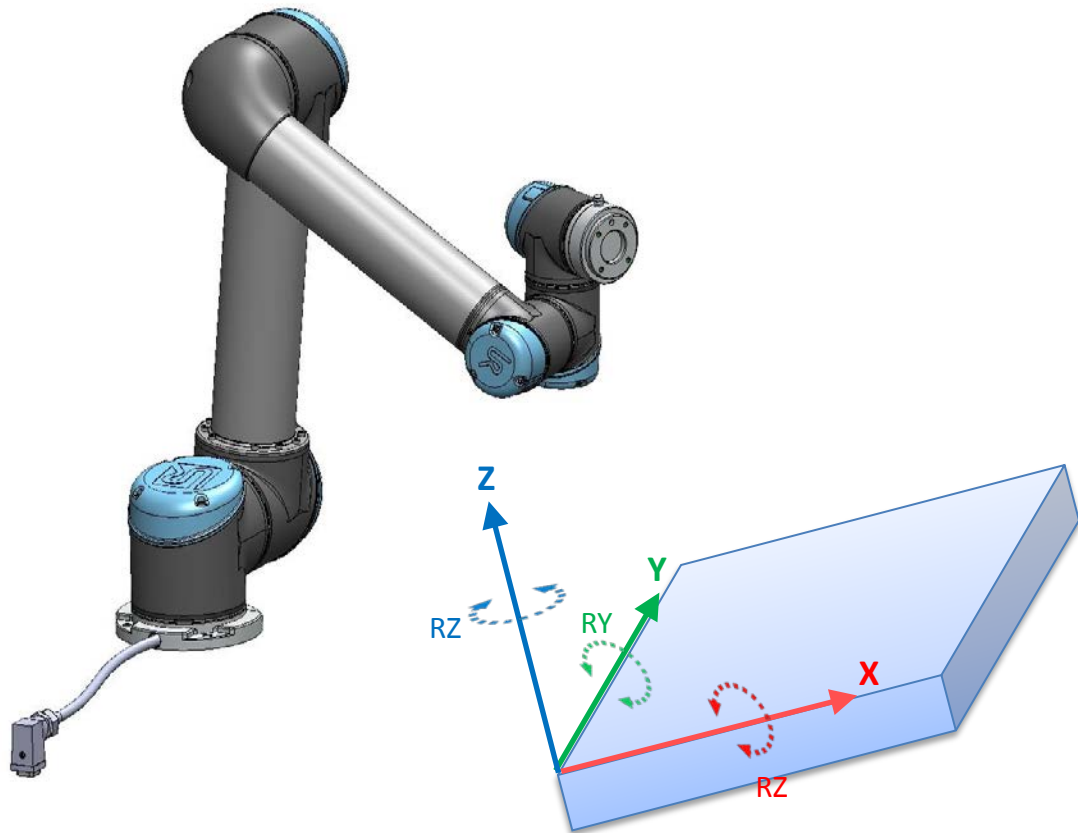
- 当TCP被定义，工具坐标系原点为TCP中心





## 用户定义特征

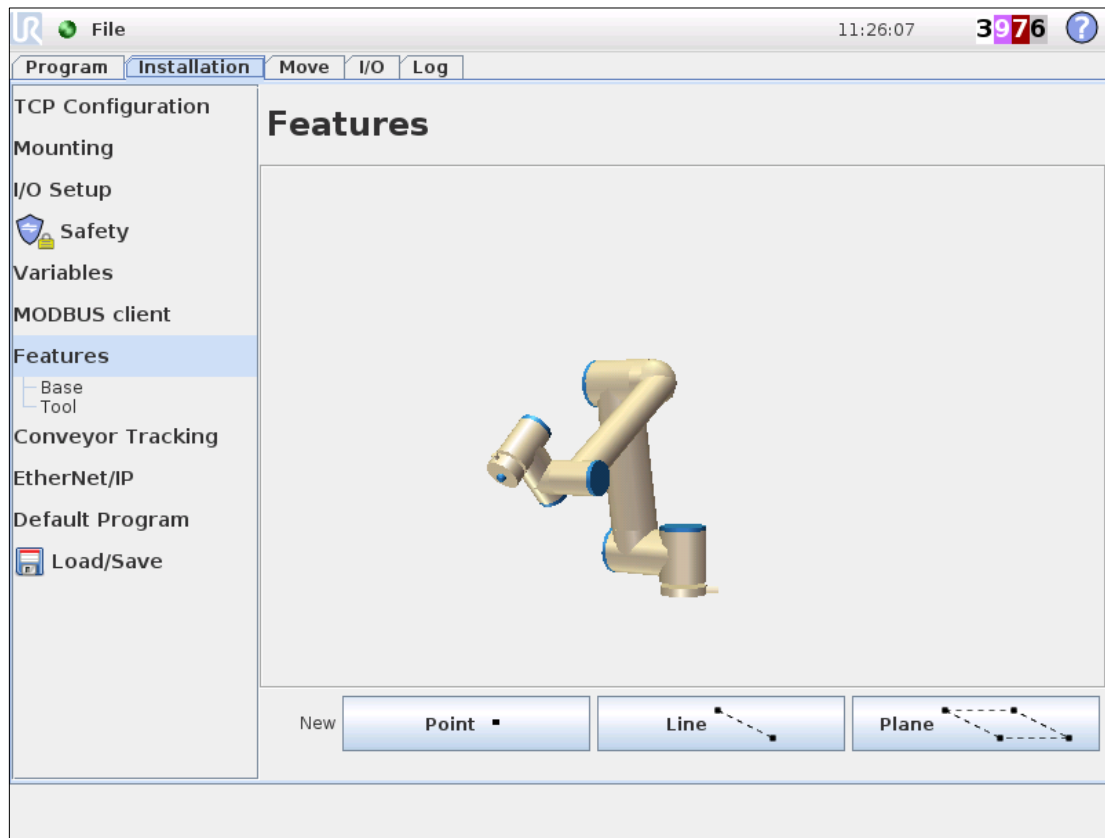
- 由操作者定义的坐标系





## 建立特征

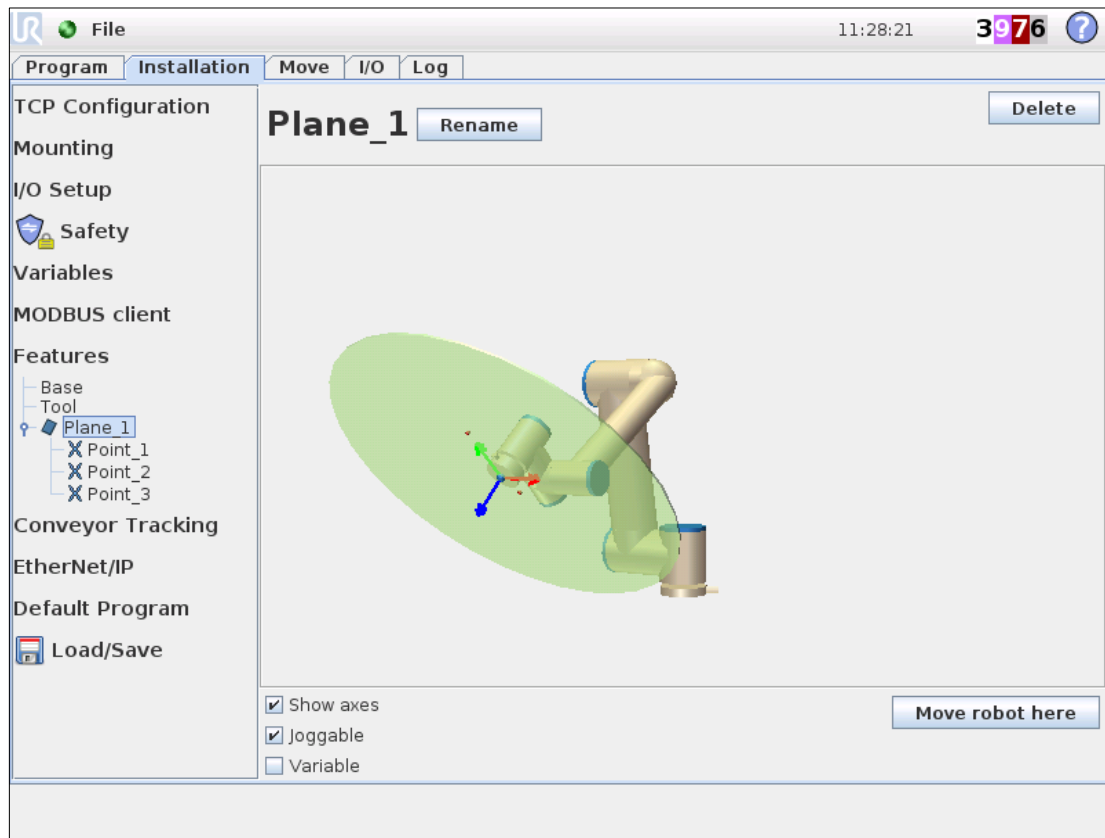
- 特征
  - 在 PolyScope 中特征可以定义为一个平面
  - 可以建立多个特征
  - 设置特征为：
    - 点
    - 线
    - 平面
- 加一个新特征
  - 平面





## 建立特征

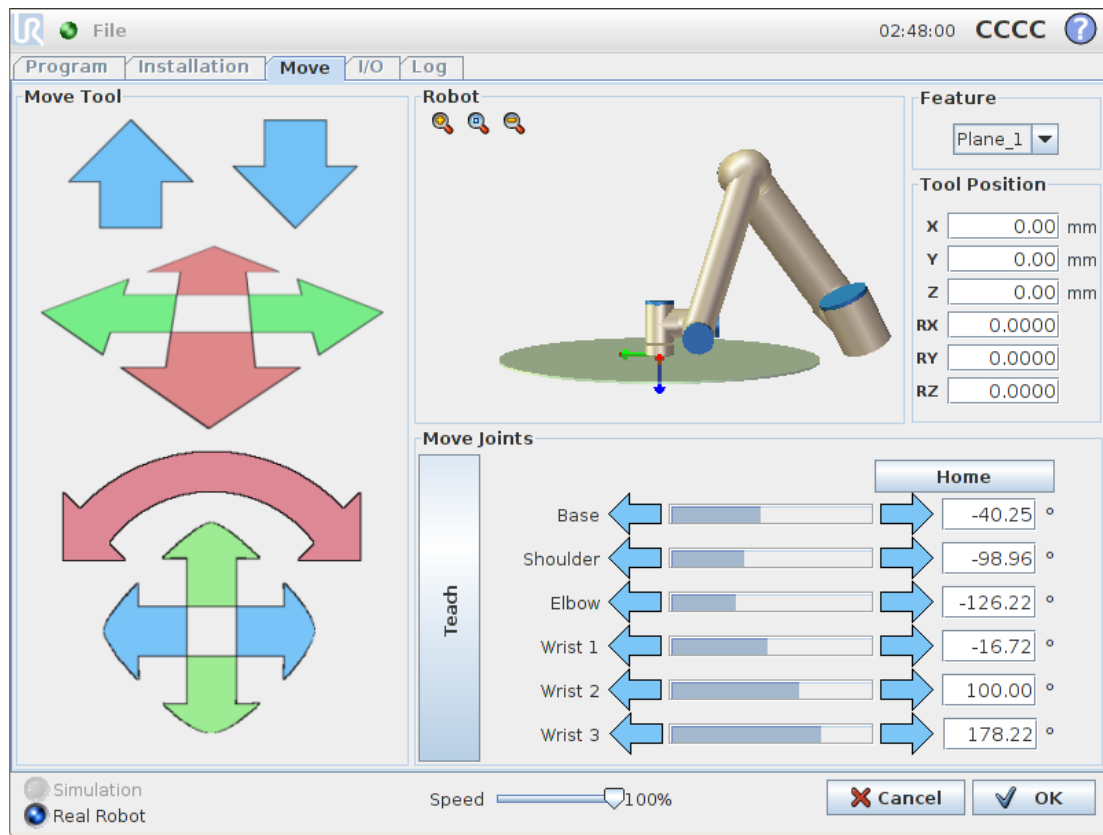
- 属性
  - 显示坐标轴（show axes）
    - 显示颜色
  - 可唤醒（joggable）
    - 在移动窗口激活特征
  - 变量（Variable）
    - 为特征创建一个备份
- 移动机器人到此处
  - TCP位置垂直于特征





## 相对特征移动机器人

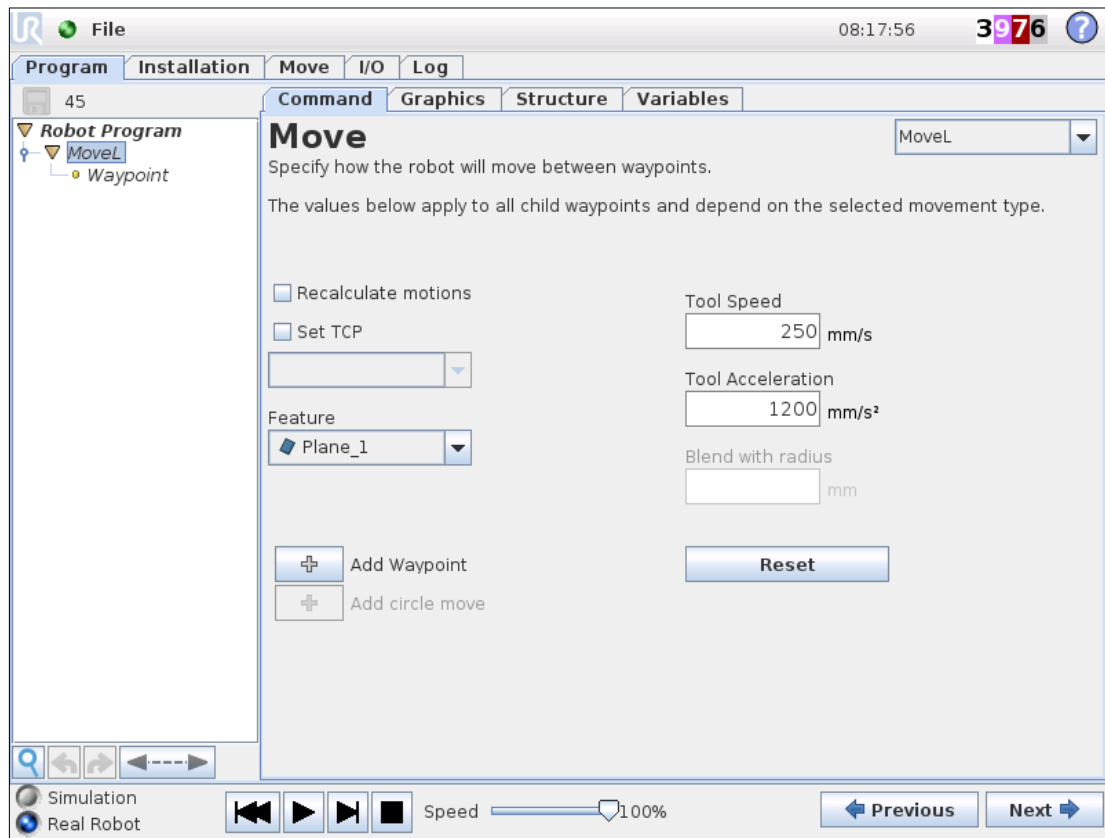
- 选择特征
  - 用箭头键移动机器人
  - 表达式编辑器





## 相对特征移动机器人

- 移动指令
  - 特征适用于
    - MoveL
    - MoveP
    - MoveC
- 共享参数
  - 选择用户定义特征





## 用特征工作

- 创建一个程序
  - 插入 MoveJ
    - 示教 home 位置
  - 插入 MoveL
    - 选择 **Plane\_1** 作为特征
    - 示教4个路点
  - 运行程序
  - 重新示教一个特征
    - 检查特征作为变量
    - 检查路点被转移了

### Robot Program

MoveJ

home

MoveL

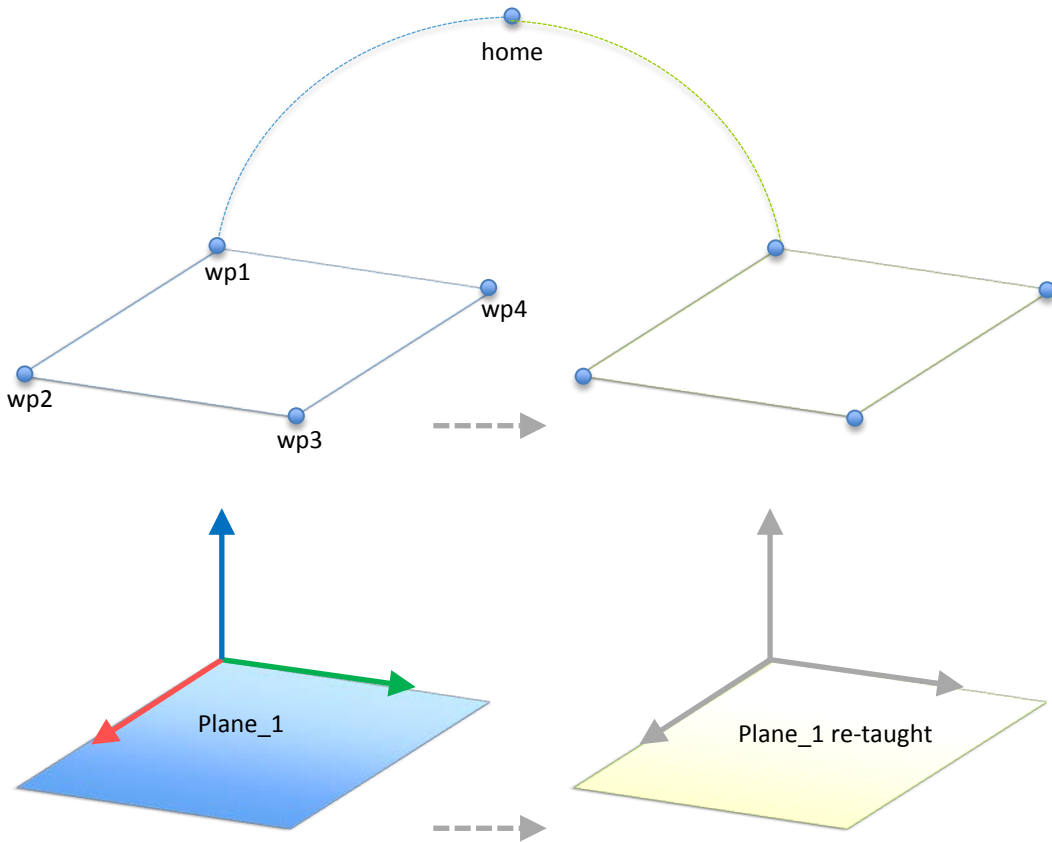
Waypoint\_1

Waypoint\_2

Waypoint\_3

Waypoint\_4

Waypoint\_1





## 用特征工作

- 使用一个简单程序
  - 用脚本代码补偿特征
    - `pose_add()`
    - `pose_trans()`
  - 如何做
    - 在Y方向上定义补偿值
    - 在位姿变量中用补偿值
    - 用位姿变量到补偿特征

### Robot Program

```
MoveJ
```

```
home
```

```
y = 0.05
```

```
offset = p[0,y,0,0,0]
```

```
Plane_1_var = pose_trans(Plane_1_var,offset)
```

```
MoveL
```

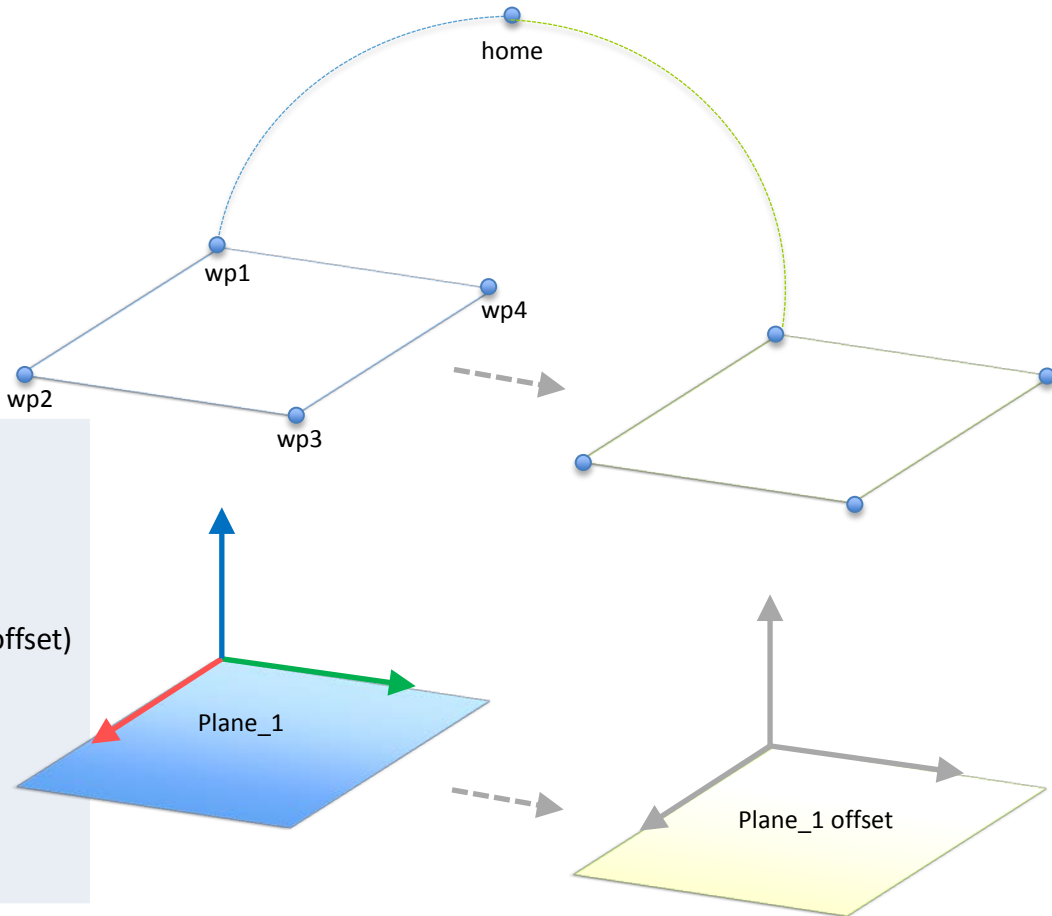
```
Waypoint_1
```

```
Waypoint_2
```

```
Waypoint_3
```

```
Waypoint_4
```

```
Waypoint_1
```





## 用特征工作

- 使用一个简单程序
  - 创建两个特征
    - Plane\_1
    - Plane\_2
  - 用两个输入信号切换特征
    - If DI[0] is high, 用Plane\_1
    - If DI[1] is high, 用Plane\_2

### Robot Program

MoveJ

home

IF DI[0] = True

Plane\_1\_var = Plane\_1

ELSEIF DI[1] = True

Plane\_1\_var = Plane\_2

MoveL

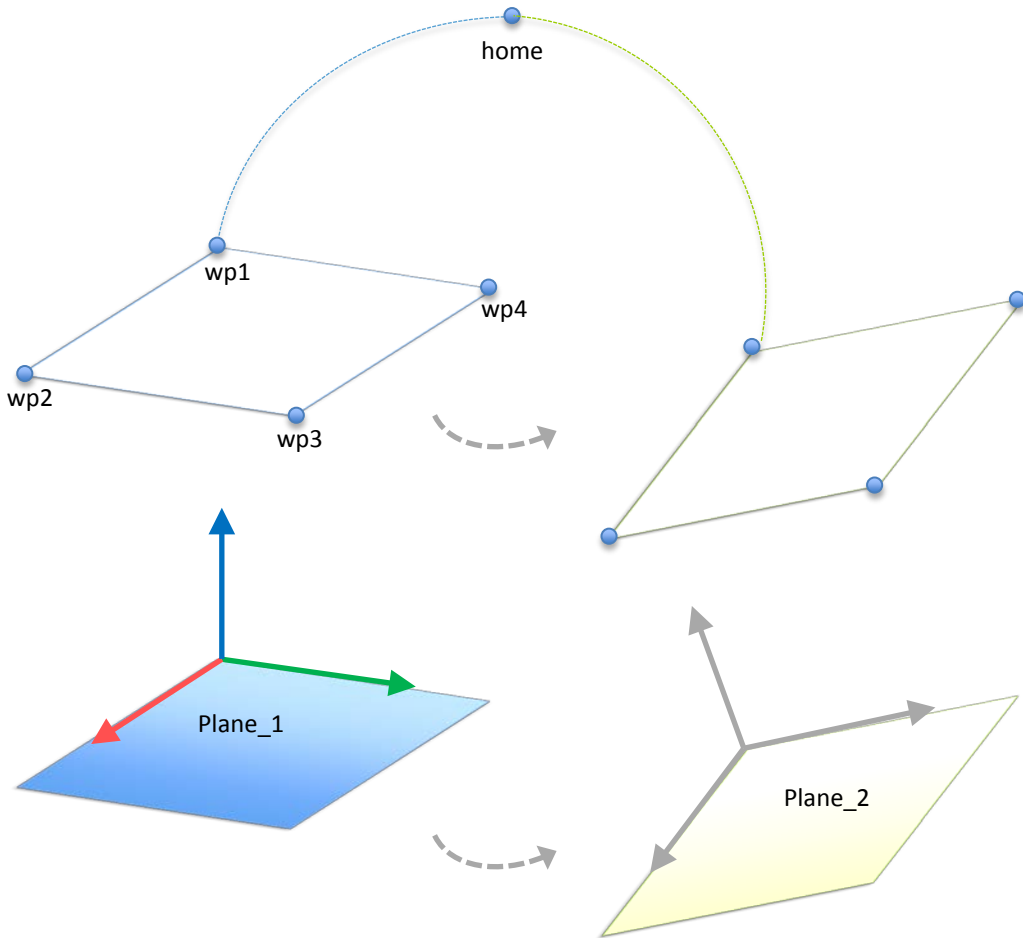
Waypoint\_1

Waypoint\_2

Waypoint\_3

Waypoint\_4

Waypoint\_1

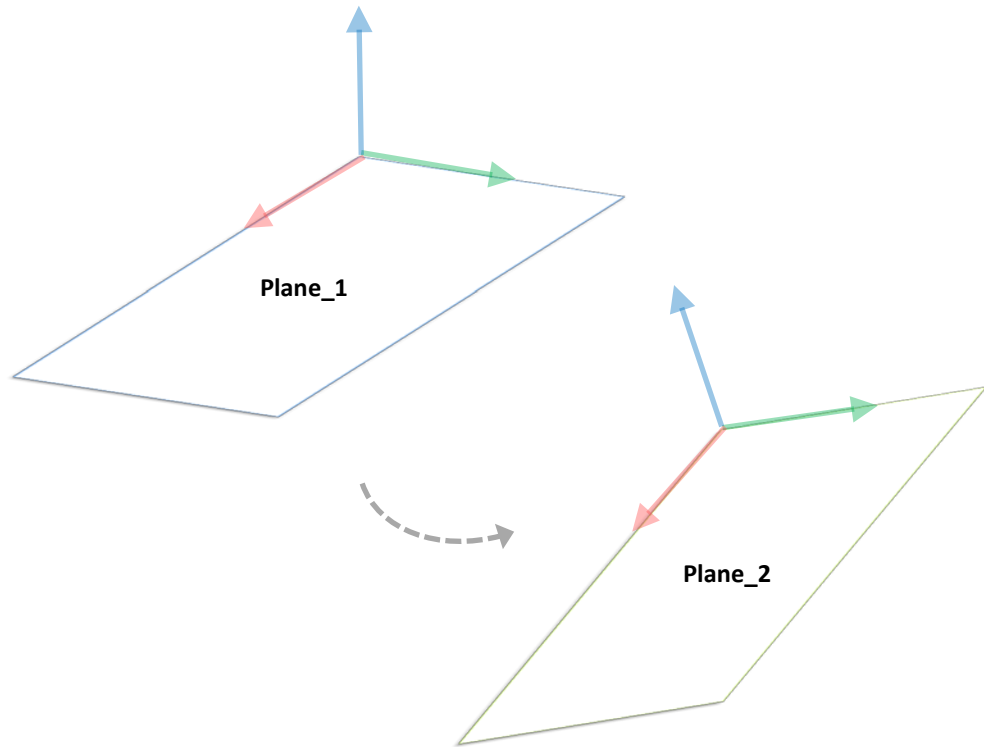






## 培训练习 1

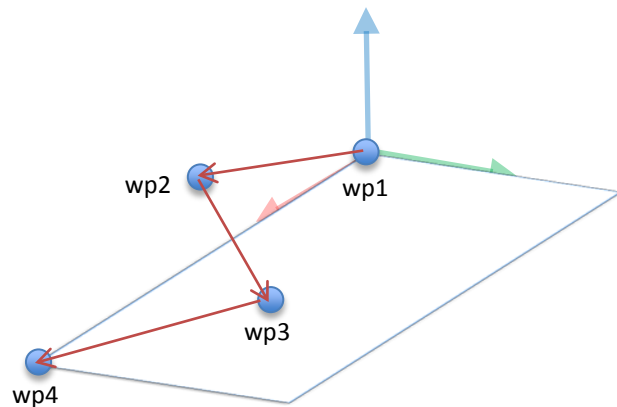
- 建立两个特征平面
  - Plane\_1
  - Plane\_2
- 把他们设为变量
- 保存在安装设置文件中





## 培训练习 2

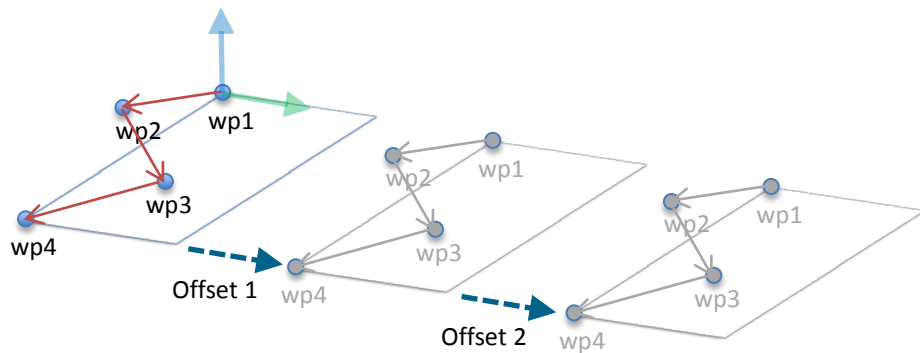
- 创建一个程序
  - 加一个 MoveL 指令
  - 在 MoveL 指令中设置特征为 Plane\_1
  - 用 MoveL 示教4个路点
  - 设置第一个路点为特征 Plane\_1 的原点





### 培训练习 3

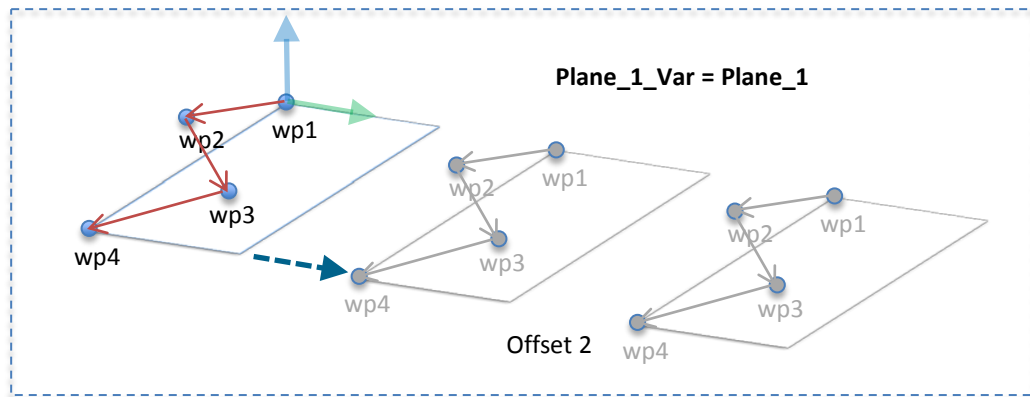
- 一旦完成 MoveL 程序
  - 在y方向上为 plane\_1 补偿100mm
  - 重复 MoveL
  - 在同一个方向上补偿另外100 mm
  - 重复 MoveL



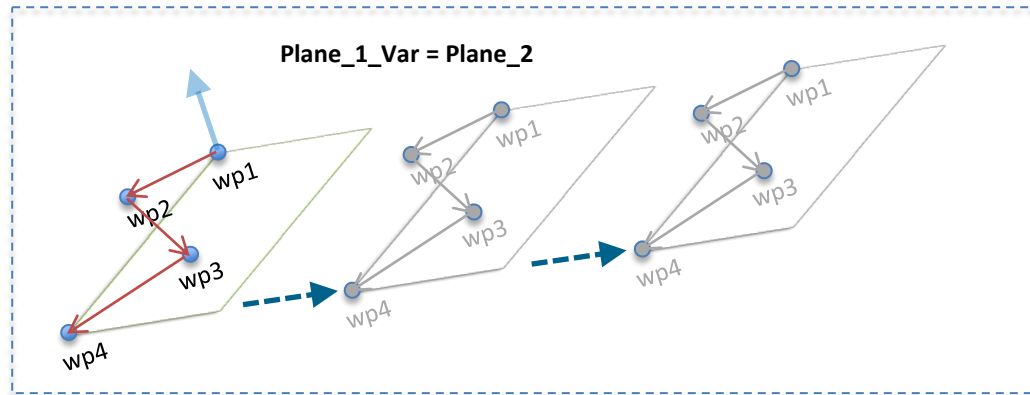


## 培训练习 4

- 完成第三个 MoveL 程序后
  - 设置 plane\_1\_var 为第二个特征 (plane\_2)
  - 在 plane\_2 特征中执行运动



Switch to Feature 2





1 脚本代码

2 变量

3 特征

4 TCP高级使用

5 Modbus 服务器

6 FTP 服务器

7 Dashboard 服务器

8 客户端界面

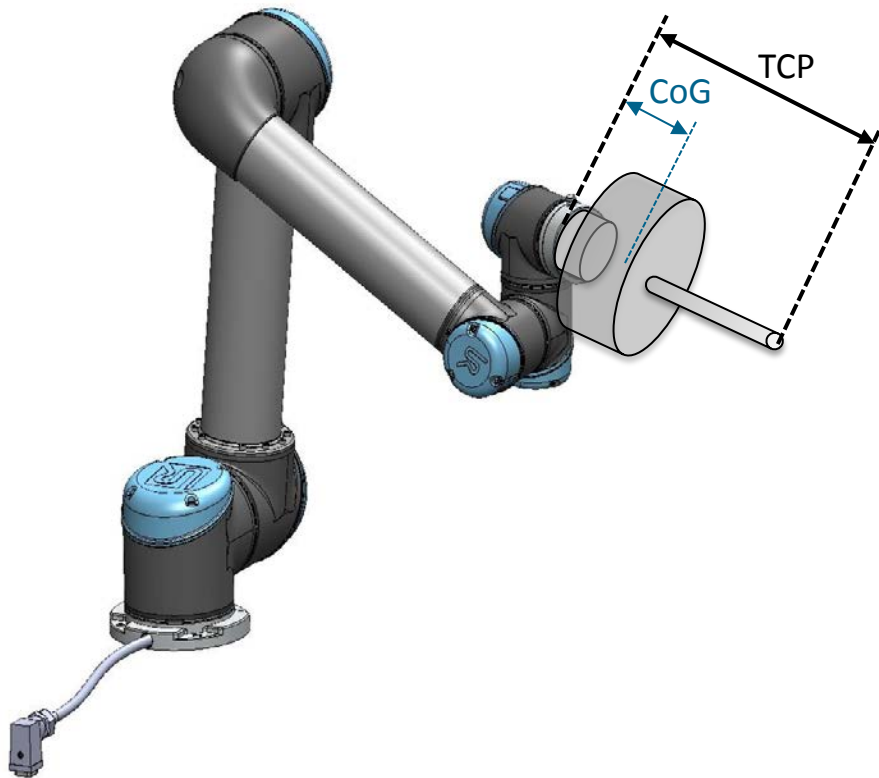
9 Socket 通讯

10 故障处理流程



## 重心 Centre of Gravity (CoG)

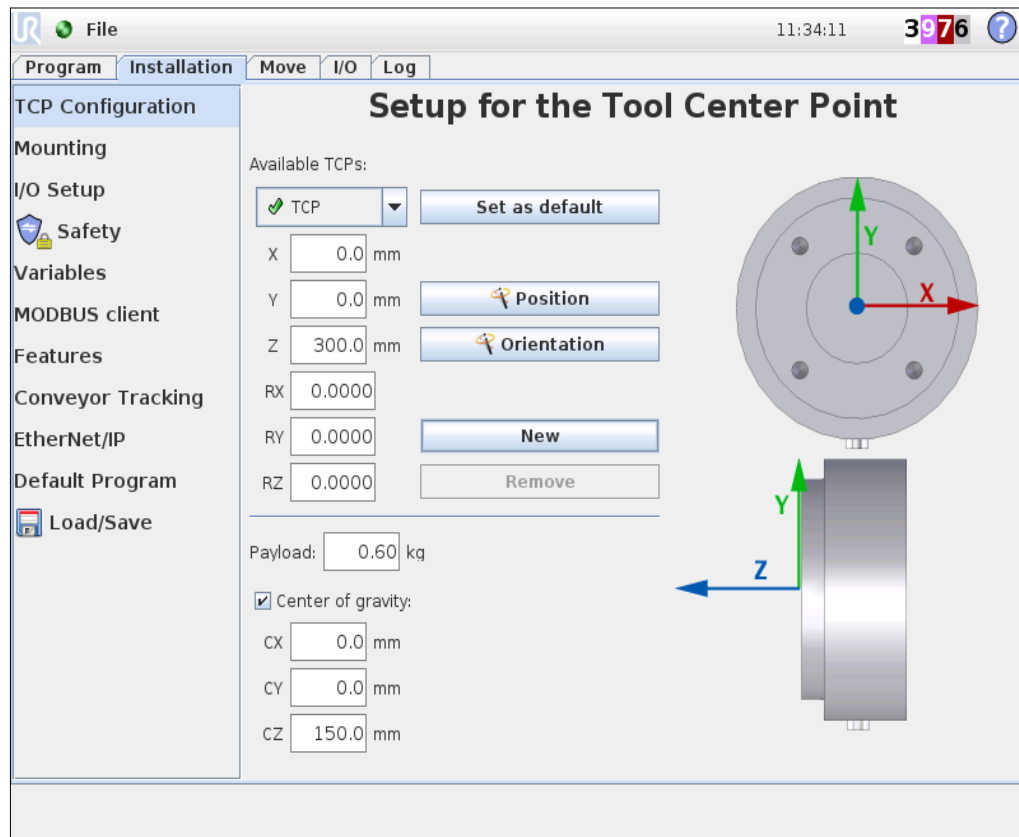
- 定义
  - 重心是TCP负载平衡的位置点
- 脚本代码
  - `set_payload(m, CoG)`
    - $m$  = 重量, kg
    - $CoG$  = 重心( $CoGx, CoGy, CoGz$ ), 米
- 注意
  - 如果重心没有定义, 软件将把TCP做为重心





## 重心 (CoG)

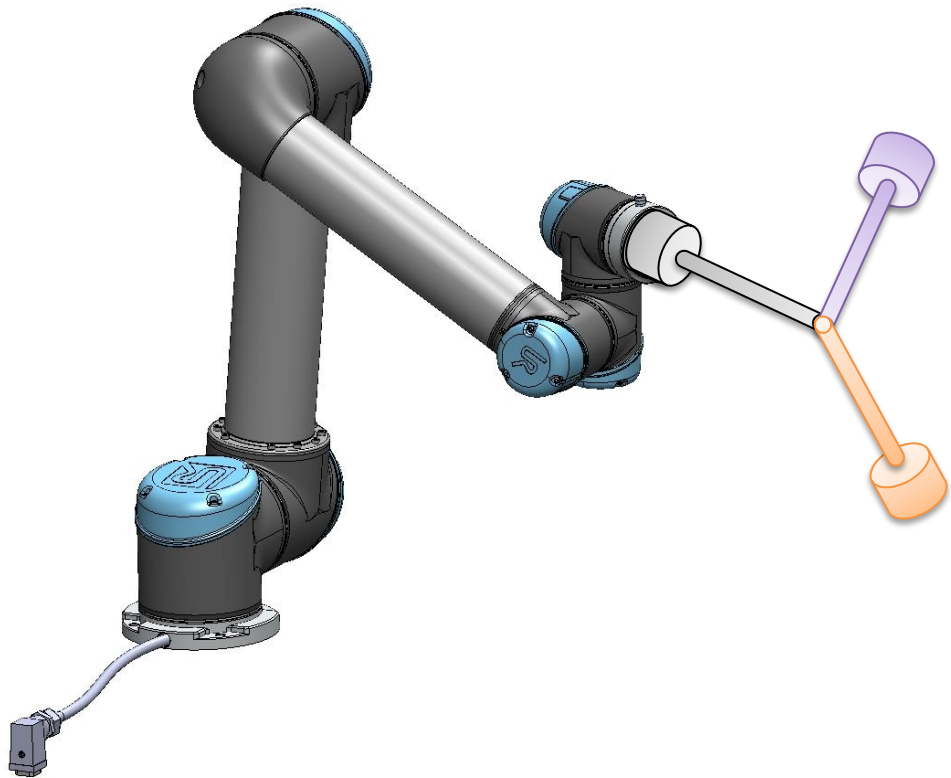
- 图形化用户界面TCP载荷设置
  - 很容易定义重心 CoG
  - 见下一页
- 如果重心在执行时改变, 仍然需要脚本代码设置





## 示教TCP位置

- 可以通过运动TCP尖端从3个角度对准同一个固定位置定义
- 这些位置示教后，机器人将计算出TCP  $x, y, z$  值
- 增加第四点改善精度

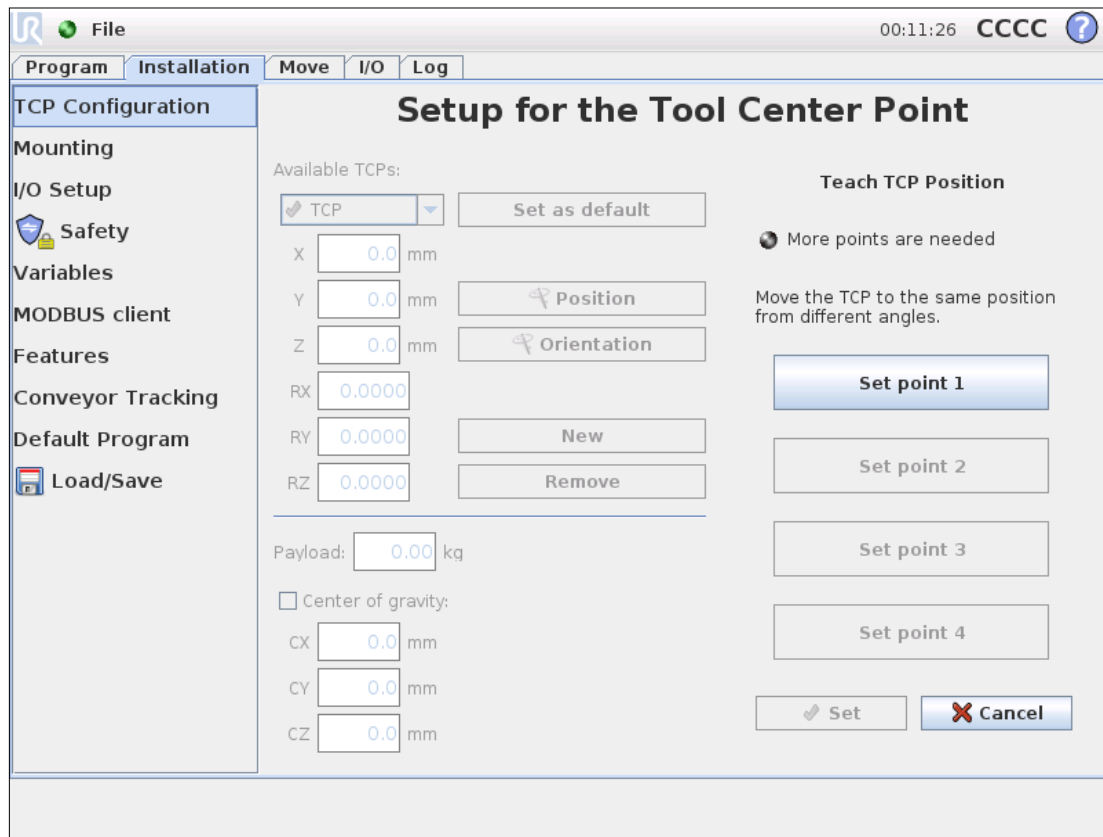






## 示教TCP位置

- 点“位置”按键示教TCP位置
- 设置3或4点
- 类似LED指示灯将显示TCP计算结果和每一个点的质量





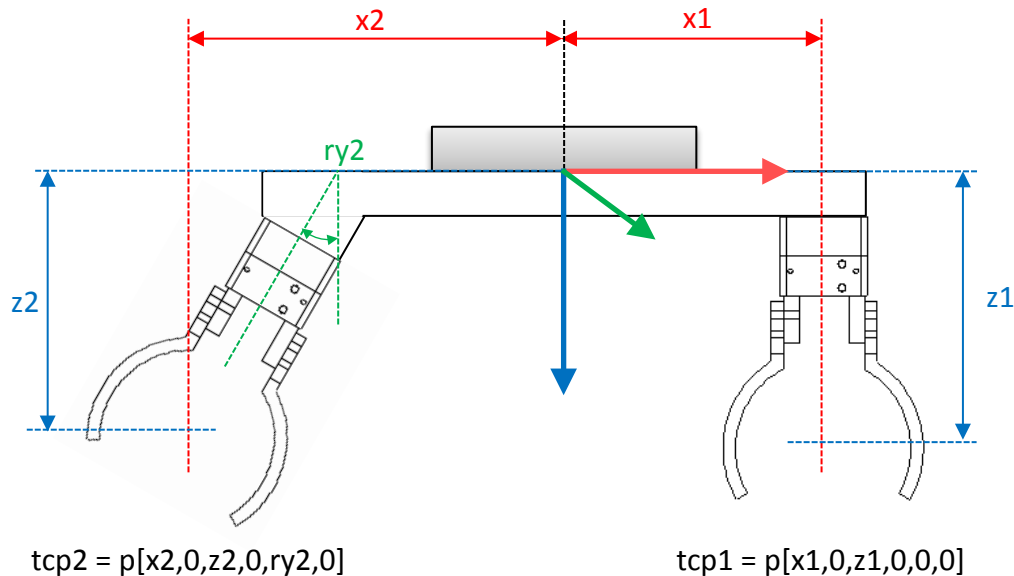
## 多个 TCP

- 设置 TCP
  - 在安装设置中只能设置一个TCP
  - 脚本代码能设置多个TCP

- 脚本代码
  - `set_tcp(pose)`
  - `pose = p[x,y,z,rx,ry,rz]`

- 安装设置
  - 设置基座坐标为变量

- 注意：
  - 只有MoveL 和 MoveP 受`set_tcp()`影响





## 多个 TCP

- 设置 tcp1

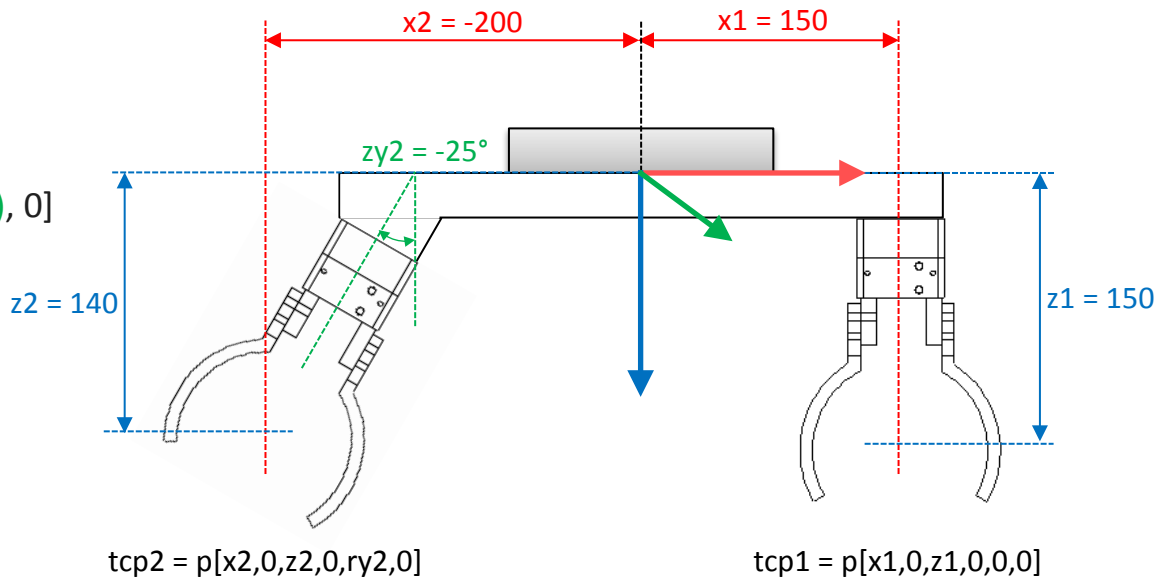
- $x1 = 150 \text{ mm.}$
- $z1 = 150 \text{ mm.}$

- $\text{tcp1} = p[0.15, 0, 0.15, 0, 0, 0]$

- 设置 tcp2

- $x2 = -200 \text{ mm.}$
- $z2 = 140 \text{ mm.}$
- $ry2 = -25^\circ$

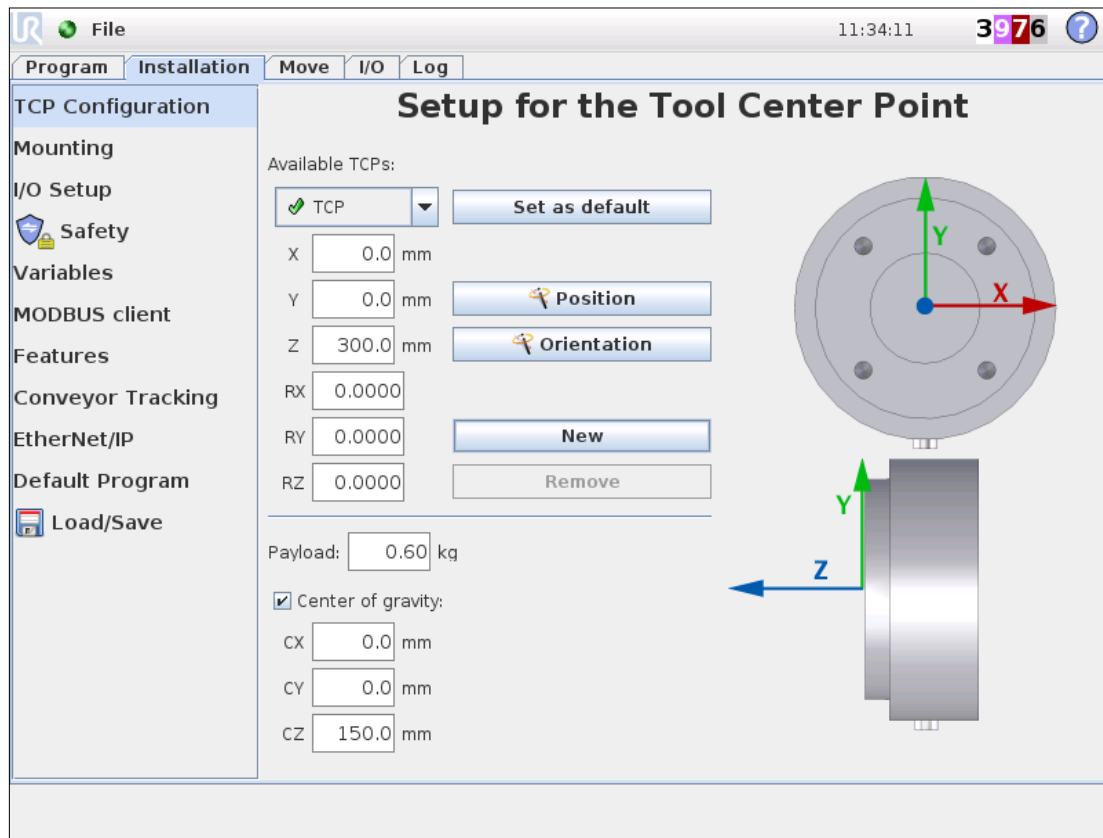
- $\text{tcp2} = p[-0.2, 0, 0.14, 0, d2r(-25), 0]$





## 设置 TCP

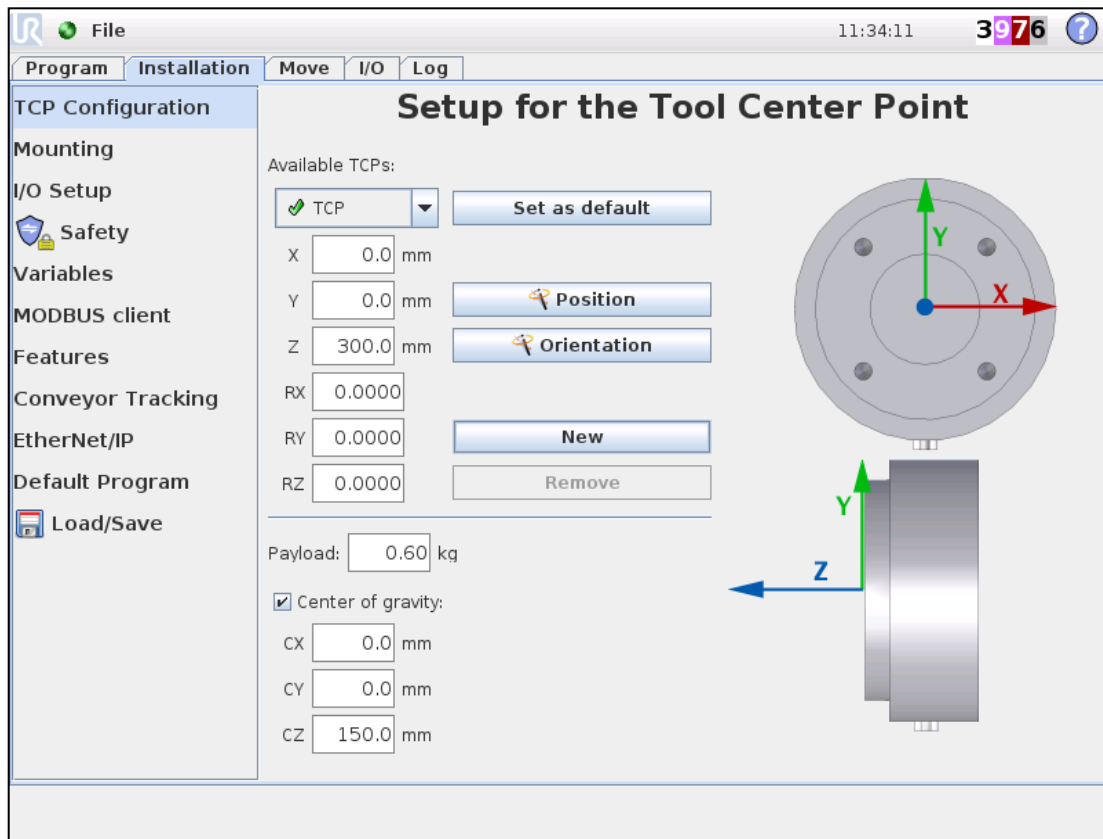
- 设置 TCP\_1 作为默认 TCP
- TCP\_1
  - x1 = 150 mm.
  - z1 = 150 mm.





## 设置 TCP

- 点“新建”增加第2个TCP
- TCP\_2
  - $x2 = -200 \text{ mm}$ .
  - $z2 = 140 \text{ mm}$ .
  - $ry2 = -25^\circ$
- 选择角度必须用弧度定义
  - $25 \times \pi / 180 = 0.4363$





## 执行程序

- 程序例子
  - 在安装设置窗口中把基座特征设为变量
  - 定义整数变量 *tcp* 用于更换不同的TCP
  - 在 MoveL 指令中设置特征为基座特征

BeforeStartSequence

```
tcp = 1
```

```
tcp1 = p[0.15,0,0.15,0,0,0]
```

```
tcp2 = p[-0.2,0,0.14,0,d2r(-25),0]
```

```
MoveJ
```

```
home
```

Robot Program

```
IF tcp = 1
```

```
set_tcp(tcp1)
```

```
tcp = 2
```

```
ELSEIF tcp = 2
```

```
set_tcp(tcp2)
```

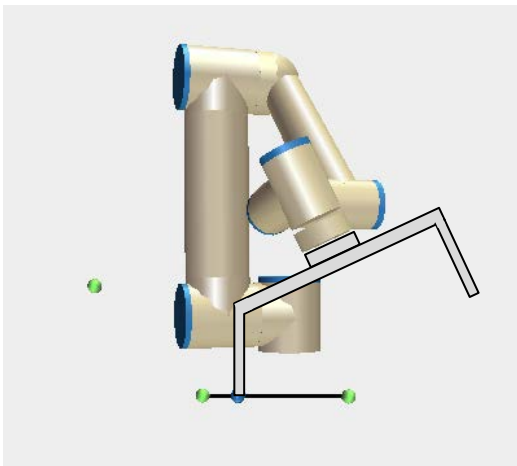
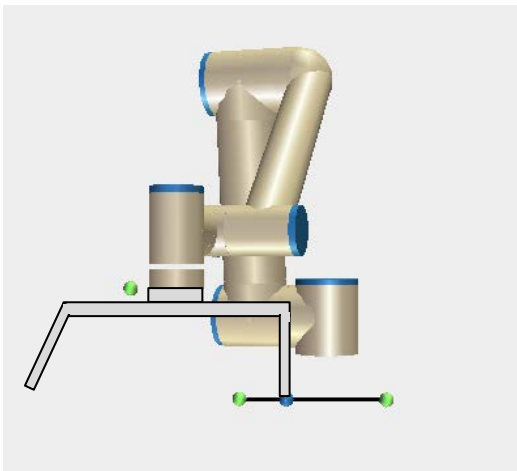
```
tcp = 1
```

```
MoveL
```

```
Waypoint_1
```

```
Waypoint_2
```

```
Waypoint_1
```





## 培训练习 1

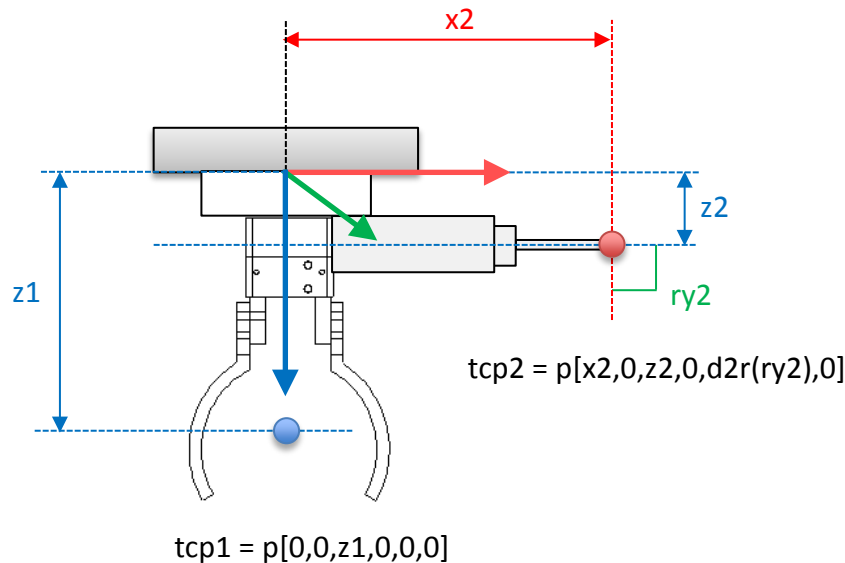
- 在安装设置中模拟定义两个TCP，一个是夹爪，一个是螺丝刀

- 夹爪 (tcp1)

- Z 方向偏移200mm

- 螺丝刀 (tcp2)

- X 方向偏移200mm
- Z 方向偏移50mm
- RY 偏移 90°



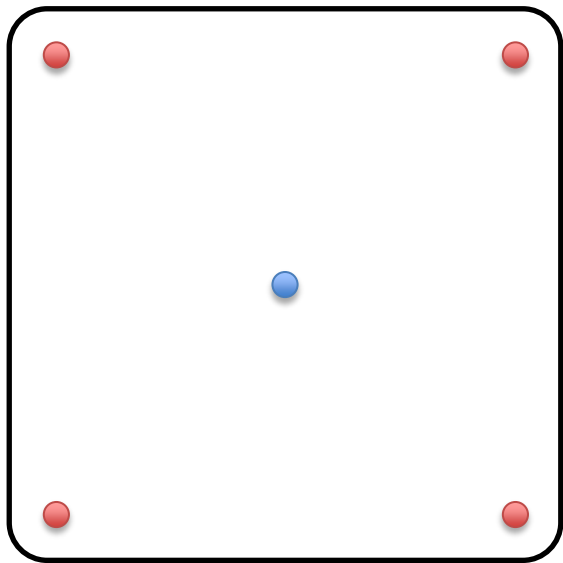
- 注意:

- set\_tcp() 只影响 MoveL, MoveP 和 MoveC



## 培训练习 2

- 模拟在方形空间用夹爪拾取放置，在用螺丝刀在4个位置锁螺丝
- 设置 TCP 为夹爪 (tcp1)
  - 运动到你自已选择的拾取位置
  - 运动到你自已选择的放置位置
- 设置 TCP 为螺丝刀 (tcp2)
  - 用 `pose_trans()` 在 放置点周边  $x=\pm 7.5\text{cm}$  and  $y=\pm 7.5\text{cm}$  的位置定义4个点
- 写一个子程序
  - 在工具坐标系下用MoveL 到相对路点（Z方向加50mm）
  - 在主程序中调用相对路点







1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

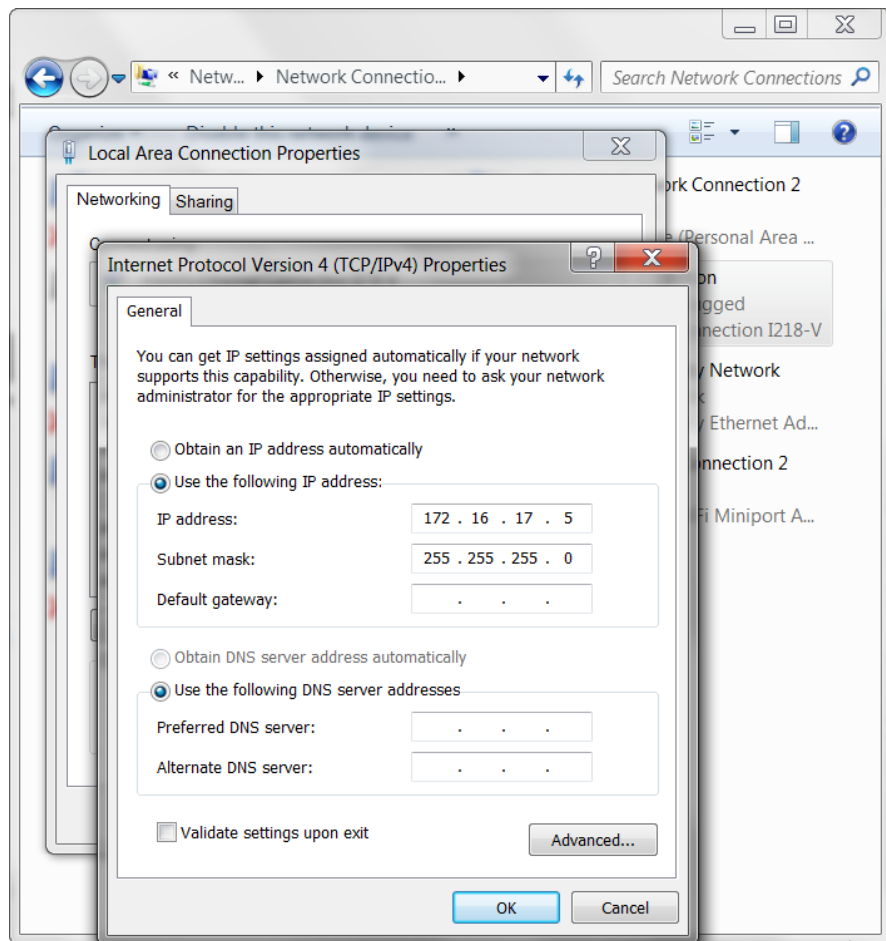
10

故障处理流程



## PC 网络设置

- 在任何 TCP/IP 通讯之前，必须设置PC和机器人网络
- 在网络设备属性界面中设置：
  - IP 地址
  - 子网掩码
- PC和机器人IP地址应该除最后一个数，其他3个数相同，在这个案例中为：
  - 192.16.17.x
- 二者子网掩码都是：
  - 255.255.255.0





## 机器人网络设置

- 进入“设置机器人”，选择设置机器人网络
  - IP 地址
  - 子网掩码
- IP 地址与PC除最后一个数，其他相同
  - 192.16.17.x
- 子网掩码都是
  - 255.255.255.0

### Setup Robot

Initialize Robot

Language and Units

Update Robot

Set Password

Calibrate Screen

Setup Network

Set Time

Back

#### Setup Network

Select your network method

☐ DHCP

☒ Static Address

☐ Disabled network

##### Network detailed settings:

IP address:	172.16.17.8	
Subnet mask:	255.255.255.0	
Default gateway:	0.0.0.0	
Preferred DNS server:	0.0.0.0	
Alternative DNS server:	0.0.0.0	

Apply

Update



## 什么是 TCP modbus?

- Modbus TCP
  - 基于以太网通讯协议
- 通讯协议
  - 协议设备间通讯的通用语言A
  - 可以在两个设备间传输数据

基础培训总结



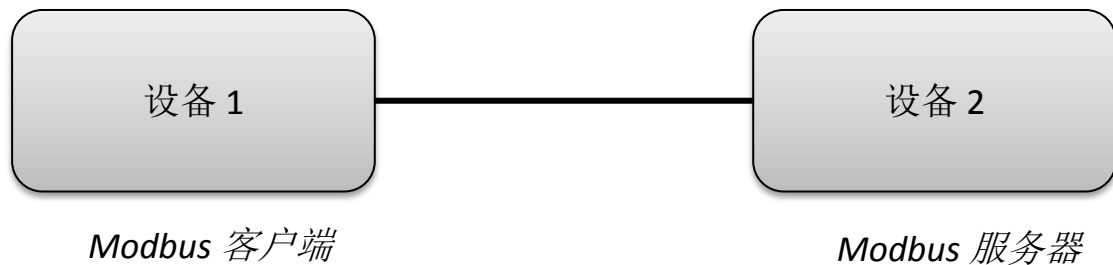
- 客户端/服务器的关系



## 客户端 / 服务器

- 服务器
  - 一个设备可以作为服务器
  - 监听来自客户端的请求
- 客户端
  - 其他设备作为客户端
  - 发请求信息给服务器

## 基础培训总结



- 每一个设备必须有唯一的IP地址



## 数据类型

- 可用于Modbus TCP 的数据类型

基本培训总结

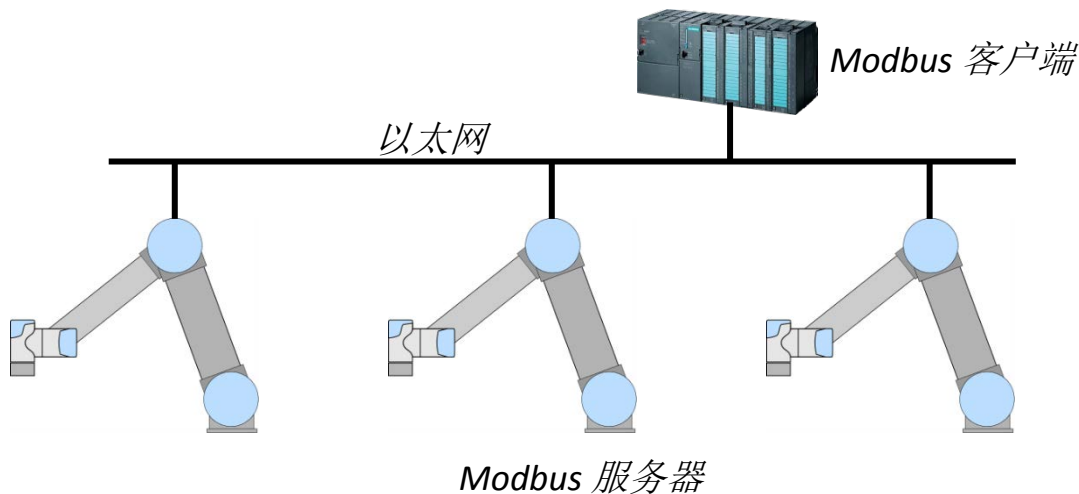
数据类型	值	地址范围
数字输入 (Digital inputs)	On/Off	技术文件来源于设备 供应商
数字输出 (Digital outputs)	On/Off	
寄存器输入 (Register inputs)	16 bit	
寄存器输出 (Register outputs)	16 bit	

- 地址范围
  - 每一个数字信号和寄存器有唯一的地址
  - 地址在供应商提供的文件中定义



## UR Modbus 服务器

- 基本培训
  - 包含如何用机器人作为客户端并连接到服务器
- 高级培训
  - 将包含优傲控制箱作为服务器并且接收客户端连接





## Modbus 服务器特征

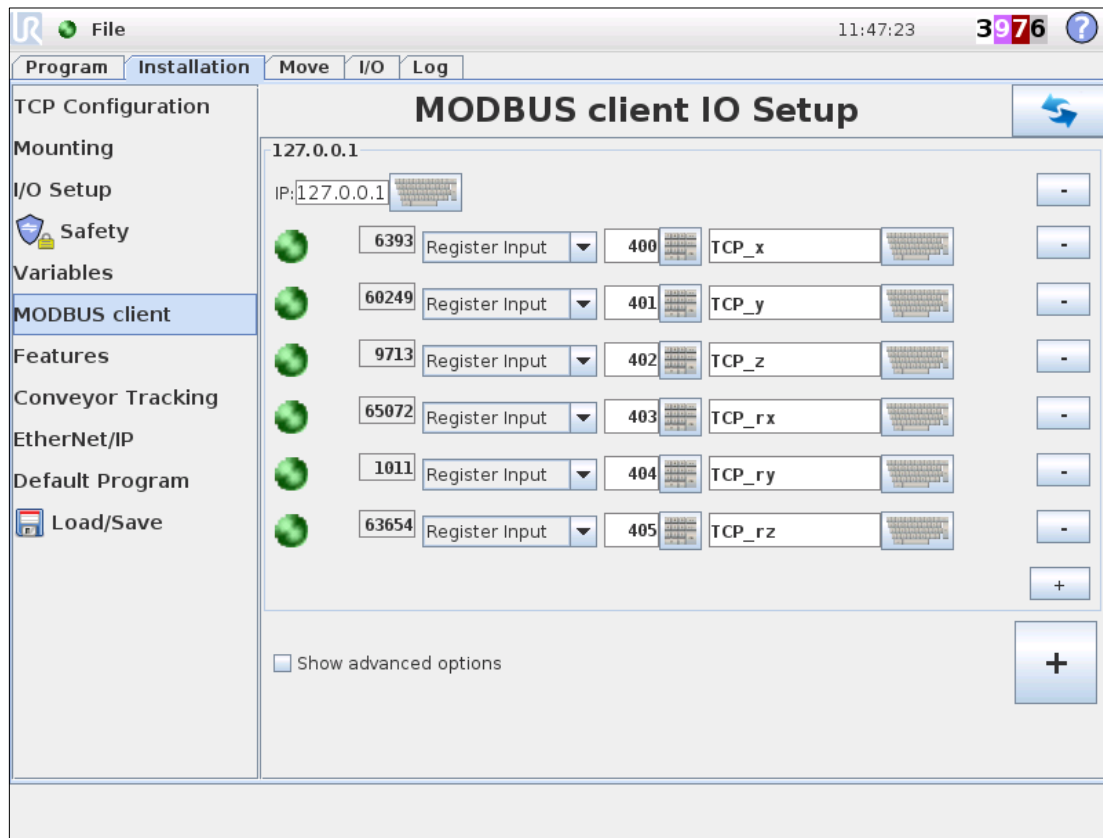
- Modbus 服务器总是运行在优傲控制箱
  - IP 地址： 在机器人设置菜单中设置
  - 端口号： 502 （不需要设置）
- 特征：
  - Modbus 服务器传输机器人状态信息，包括：
    - 关节角度/速度/电流/温度
    - TCP 位置/速度/偏移值
    - 程序状态—急停/示教模式等
  - 可以远程读写I/O 信号
    - I/O 状态传输
    - 远程设备可以控制输出
  - 通用寄存器
    - 128 个寄存器
- 寄存器地址详细列表在技术支持网站可找到





## 机器人自己服务器通道

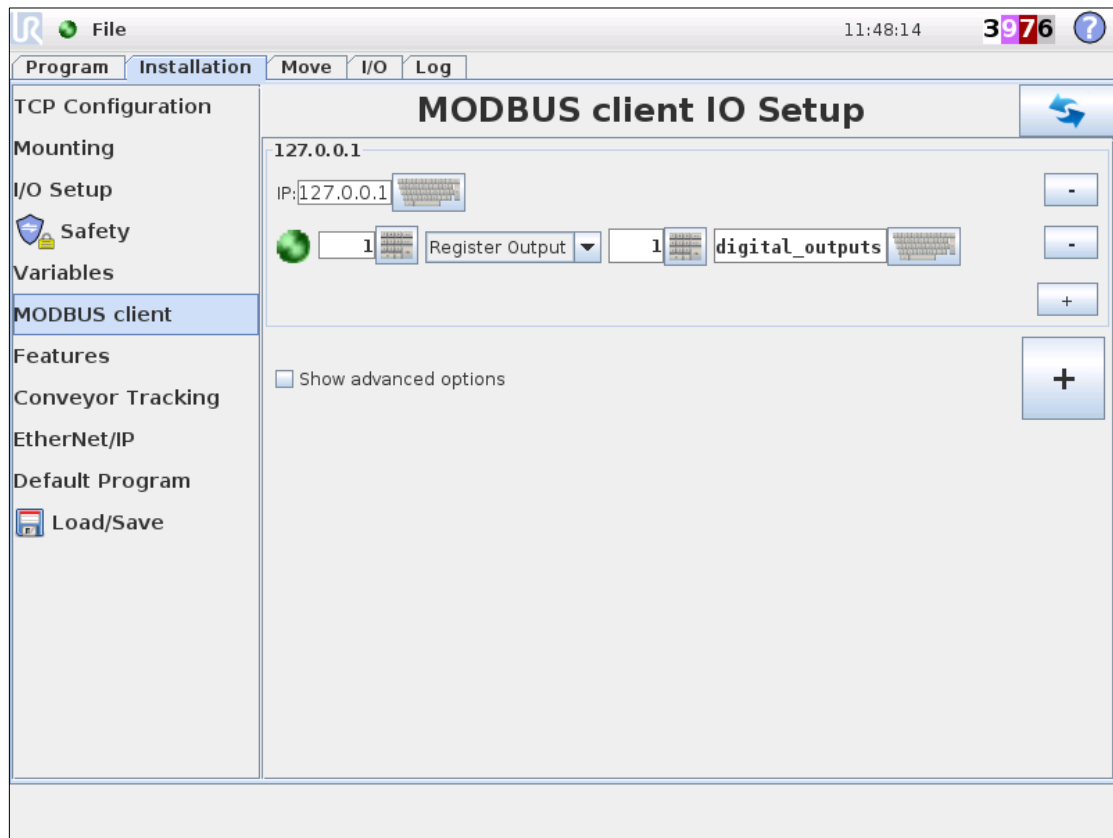
- 机器人自己服务器
  - IP 地址: 127.0.0.1
- 读TCP位置
  - 寄存器地址:
    - 400 TCP-x , 0.1mm
    - 401 TCP-y , 0.1mm
    - 402 TCP-z , 0.1mm
    - 403 TCP-rx , 千分之一弧度
    - 404 TCP-ry , 千分之一弧度
    - 405 TCP-rz , 千分之一弧度
  - 在基座坐标系
    - 263 是按示教按钮





## I/O 寄存器通道

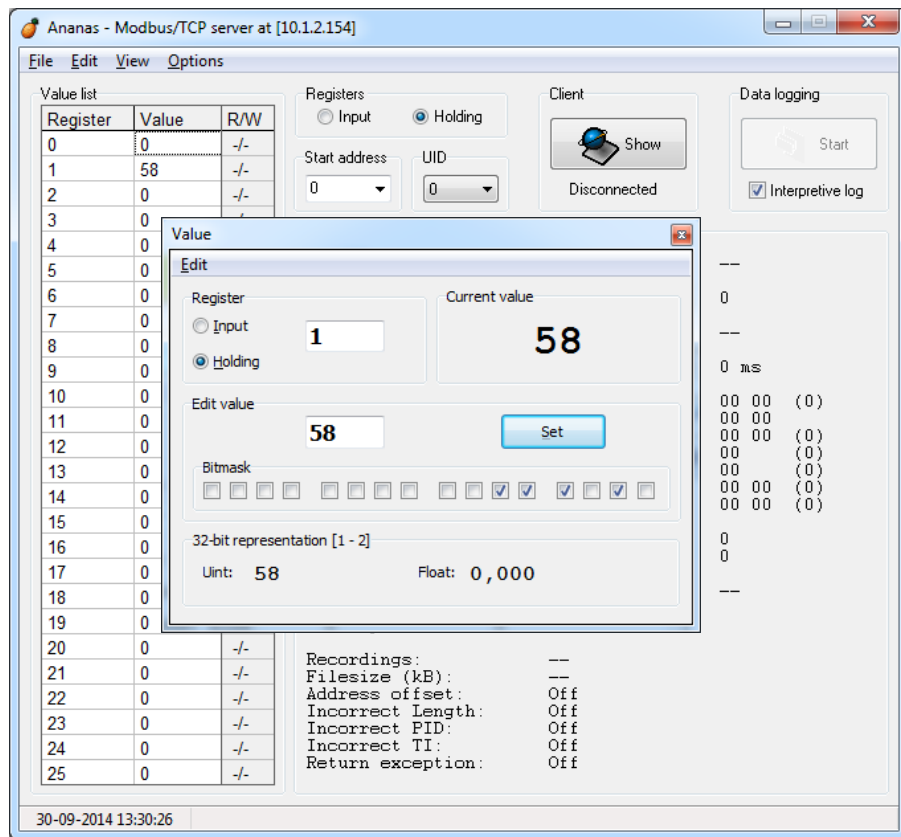
- 设置寄存器为数字输出信号
  - 寄存器地址 = 1
- 特征
  - 允许多通道I/O信号运行，节省时间
- 格式：
  - 16 bit 整数寄存器
  - [BBBBBBBTTxxxxxx]  
*B=盒子, T=工具, x=未定义*





## Modbus 应用

- Modbus服务器潜在的应用包括：
  - 从远程设备控制机器人I/O
- 如何测试
  - 安装 Ananas
    - 测试Modbus功能的免费软件
  - 设置 Ananas 作为客户端
  - 通过在控制箱检查/不检查bit和编辑整数值转换数字I/O





## 培训练习 1

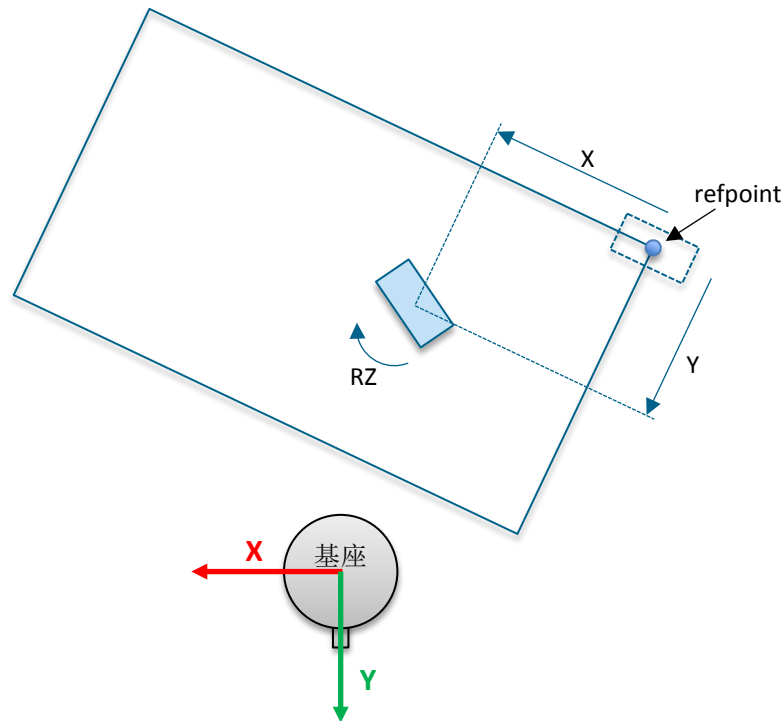
- 写一个程序
  - 读数字输出寄存器
  - 转换到二进制代码
  - 触发通道3输出信号
  - 转换回整数值
  - 写回到寄存器
- 运行程序并且观察输出改变



## 培训练习 2

- 写一个程序，接收来自于Modbus服务器的模拟视觉坐标系
  - 从Modbus服务器通用寄存器索取拾取位置变量
  - 运动到这个位置，基于 pose\_trans()
- 寄存器地址：
  - 128: 新数据有效 (0=no, 255=yes)
  - 129: x
  - 130: y
  - 131: rz

*Refpoint 为参考*





1 脚本代码

2 变量

3 特征

4 高级TCP使用

5 Modbus 服务器

6 FTP 服务器

7 Dashboard 服务器

8 客户端界面

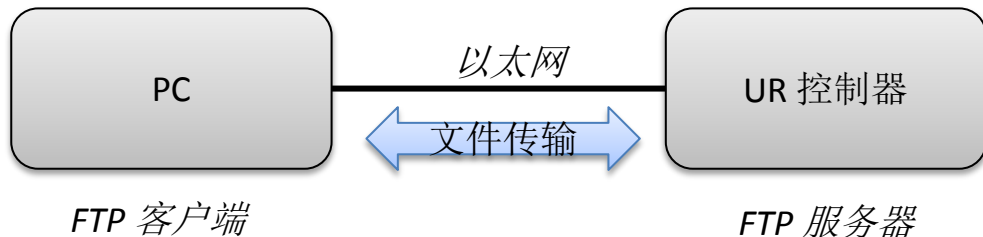
9 Socket 通讯

10 故障处理流程



## 什么是FTP服务器？

- FTP 服务器
  - 客户端/服务器进行文件传输的协议
  - 基于以太网通讯
- 目的
  - 用于很方便的传输程序
  - 很容易的备份
- 在优傲机器人控制器上的服务器
  - 服务器总是运行在优傲机器人控制器上
  - 优傲机器人控制器不需要设置





## FTP 客户端

### • FileZilla

- 免费FTP客户端软件
- [filezilla-project.org](http://filezilla-project.org)
- 通过 via ssh 或 sftp 连接

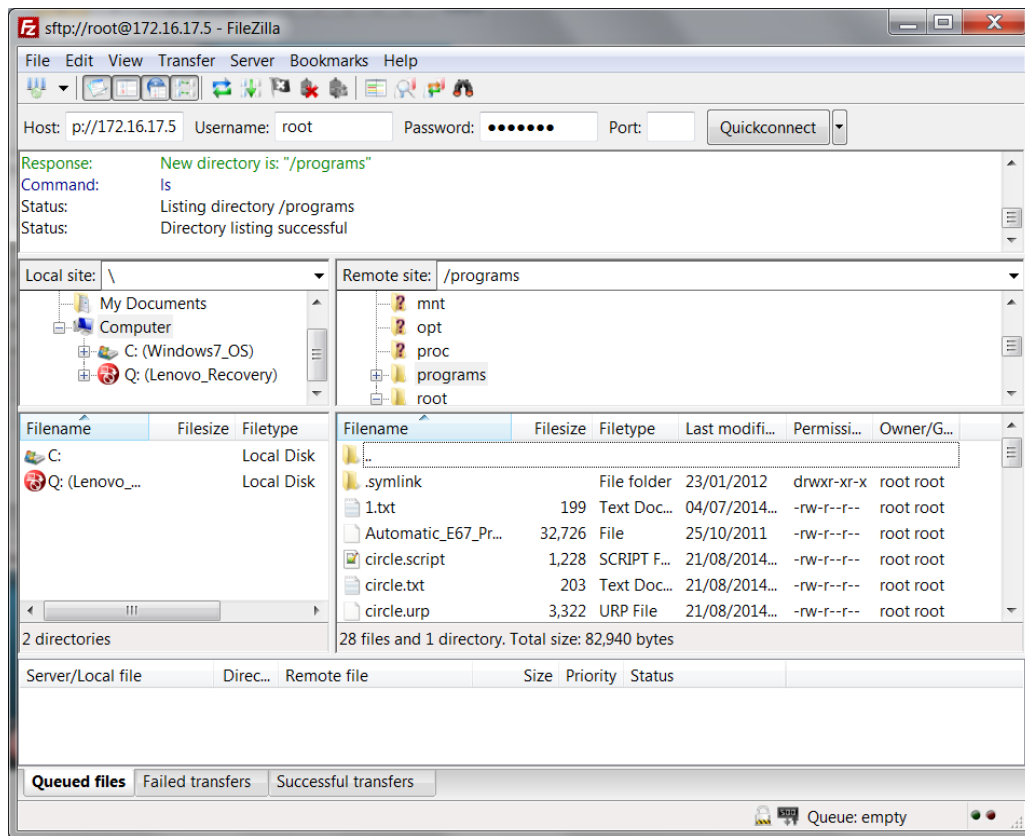
### • 如何连接

#### • 设置 FileZilla

- Host: 机器人IP地址
- User: root
- Password: easybot
- Port: 22

### • 如何传输

- 在PC和控制器/*programs* 文件夹之间拖拽文件
- 警告:  
控制箱文件系统没有写保护,  
使用错误将导致系统故障

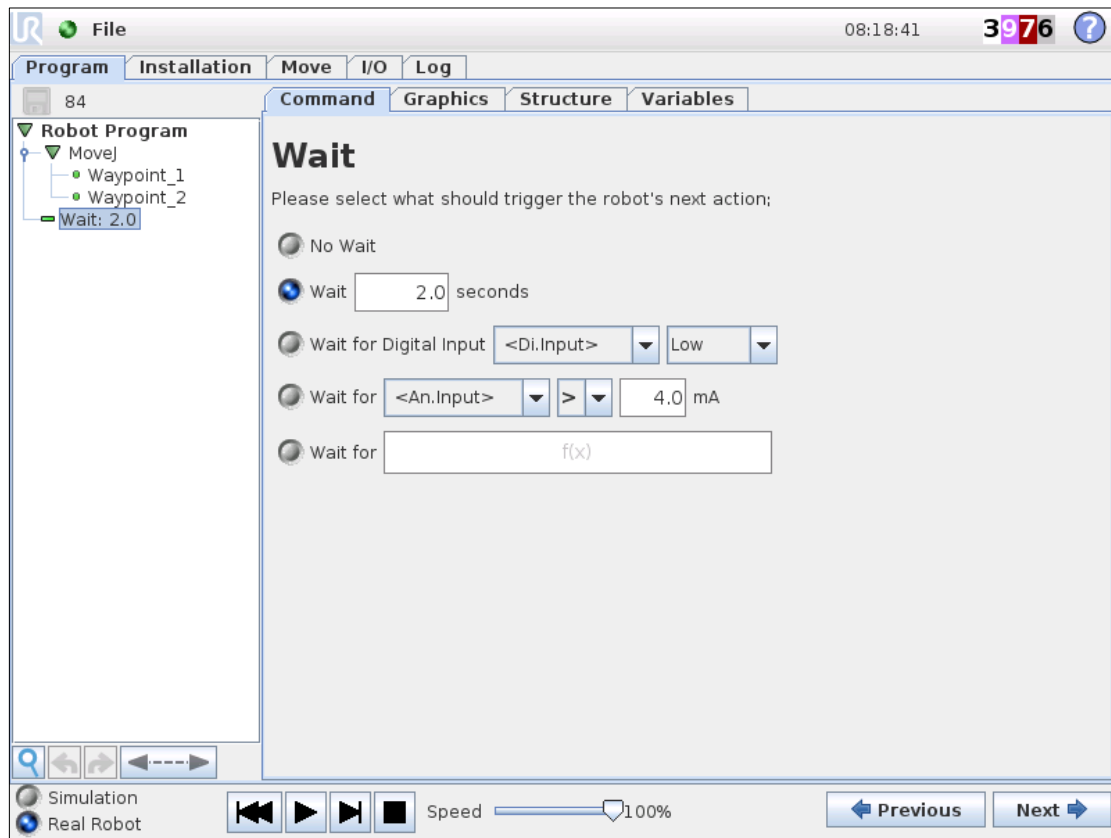






## 培训练习

- 把在 *programs* 文件夹的所有文件做备份
- 在电脑上用 URSim 写一个简单程序
- 把写好的程序拷贝到 *programs* 文件夹中
- 在机器人上运行这个程序





1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

10

程序问题



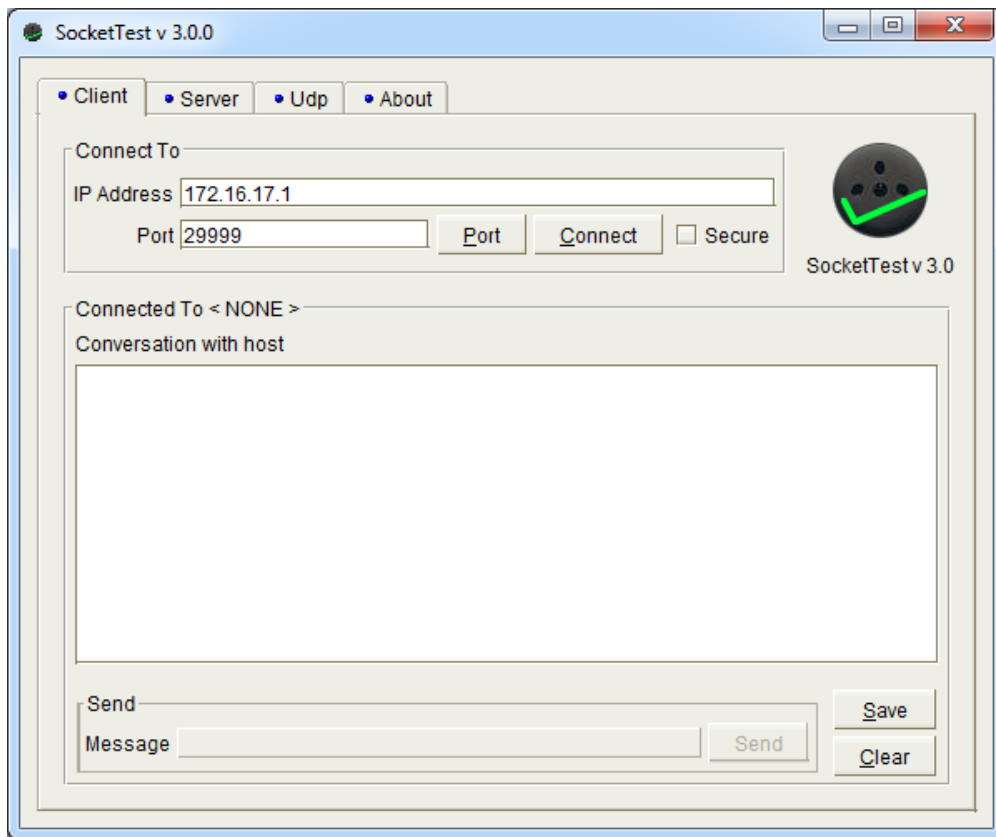
## 什么是 Dashboard 服务器？

- Dashboard 服务器
  - 允许通过端口号29999远程控制机器人
  - 控制机器人操作，如导入程序、启动程序等
  - 得到机器人状态反馈（如机器人正在运行、停止等）
  - 显示/关闭弹出窗口
  - 设置用户权限水平（如GUI界面某些部分禁止访问）
  - 在技术支持网站可找到所有指令列表



## SocketTest

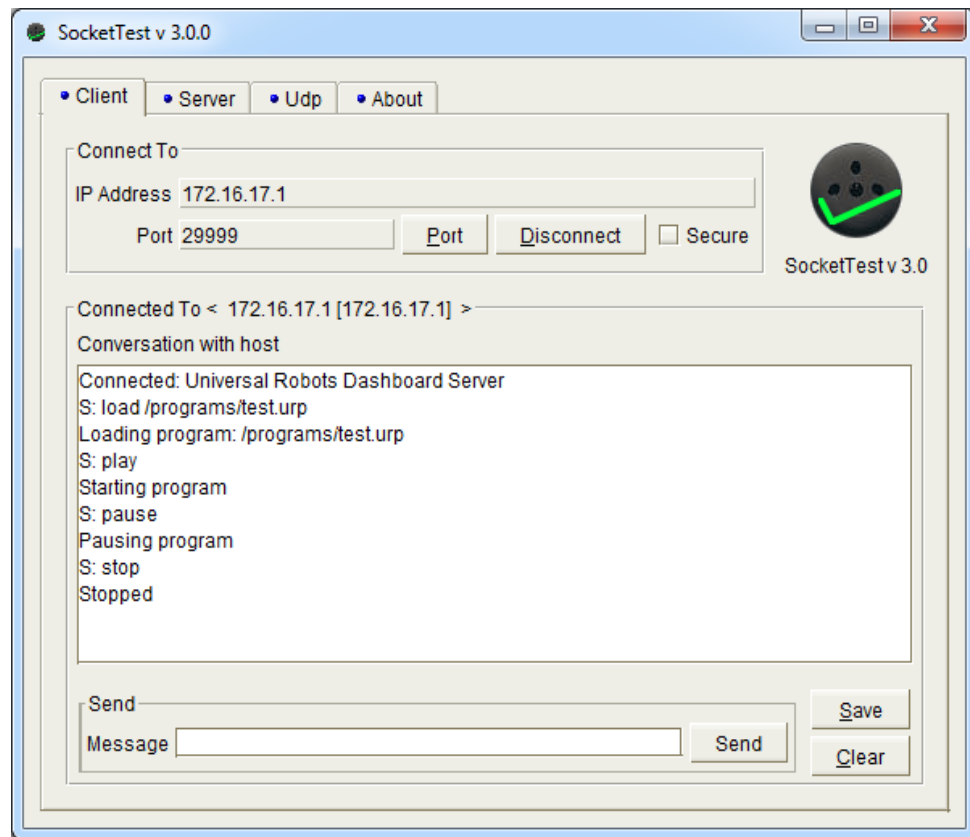
- SocketTest
  - 测试TCP/IP socket 通讯的免费软件
  - [sockettest.sourceforge.net](http://sockettest.sourceforge.net)
- 如何连接
  - 在机器人和PC上设好IP地址
  - 打开 SocketTest
    - 选择客户端（Client）窗口
    - 输入机器人IP地址
    - 输入Dashboard 服务器端口号 (29999)
    - 点连接（Connect）





## Dashboard 远程操作

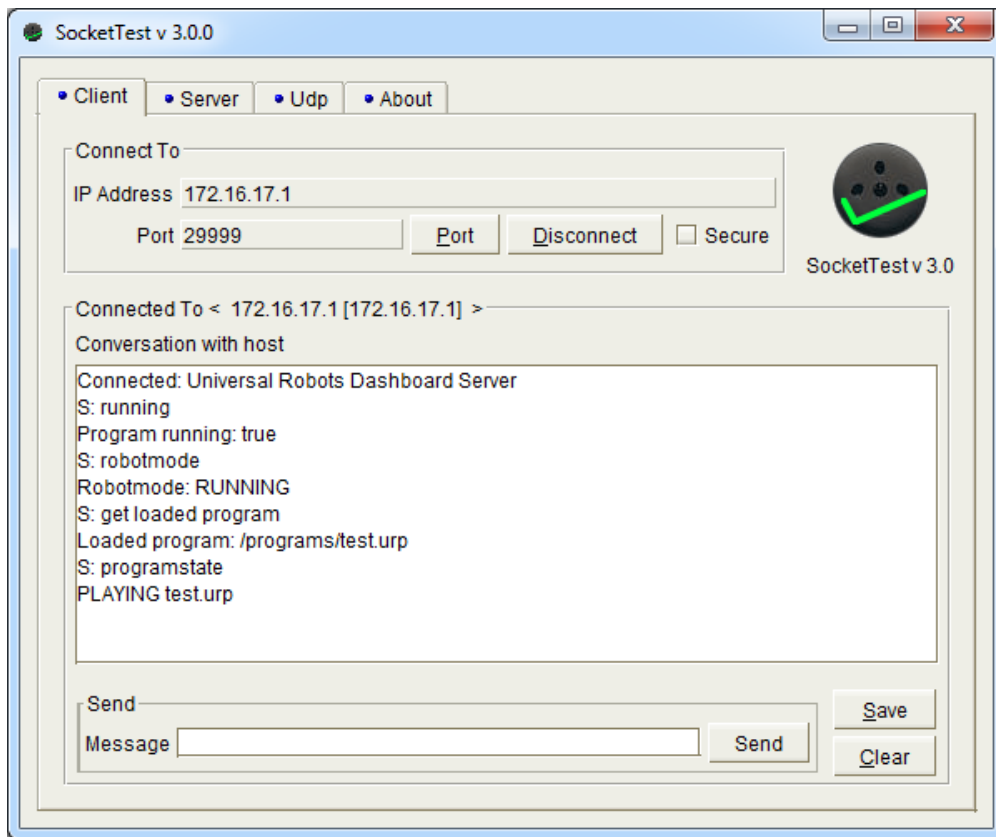
- 远程控制机器人
  - 导入程序load (*<program.urp>*)
  - 启动程序play (*starts loaded program*)
  - 停止程序stop (*stops running program*)
  - 暂停程序pause (*pauses the running program*)
  - 关闭连接quit (*closes connection*)
  - 关机shutdown (*shuts down and turns off the robot and controller*)





## Dashboard 反馈

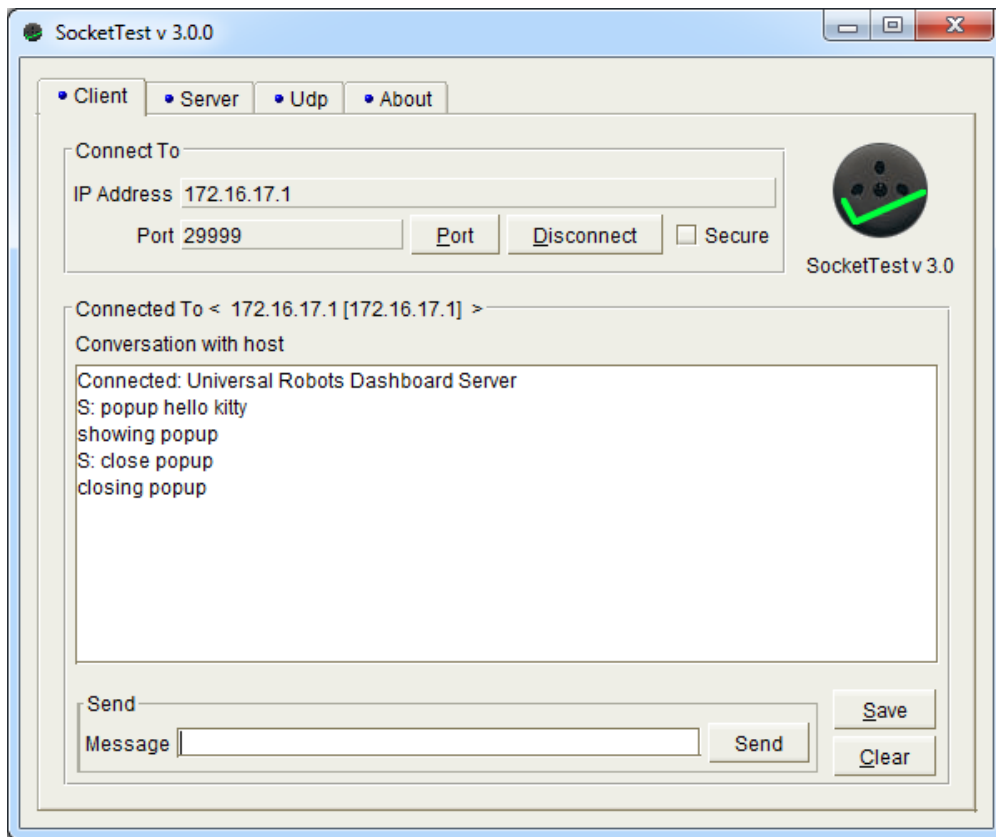
- 请求机器人状态
  - 执行状态请求running (*execution state enquiry*)
  - 机器人模式请求robotmode (*robot mode enquiry*)
  - 得到加载的程序Get loaded program (*which program is loaded*)
  - 返回程序保存状态 isProgramSaved (*Returns the save state of the active program*)
  - 返回程序状态 programState (*Returns the state of the active program, or STOPPED if no program is loaded*)





## Dashboard 弹出窗口

- 在 GUI 中打开或者关闭弹出窗口
  - 打开一个弹出窗口popup  
<popup-text> (open a popup)
  - 关闭一个弹出窗口close popup  
(closes the popup)





## Dashboard 用户角色

- 在欢迎屏幕控制可选项
  - 设置用户角色 `setUserRole <role>`

用户角色 (<role>)	描述
编程者 (programmer)	“设置机器人”按钮无效，专家模式“Expert Mode”可用（需要正确的密码）
操作者 (operator)	仅仅“运行程序”和“关闭机器人”有效，专家模式“Expert Mode”不能被激活
无限制 (none)	所有按钮有效，专家模式“Expert Mode”有效（需要正确的密码）
锁定 (locked)	所有按钮无效，专家模式“Expert Mode”不能被激活





## 培训练习

- 返回欢迎屏幕
- 连接到 dashboard 服务器
- 通过FTP加载一个程序到机器人
- 运行这个程序



1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

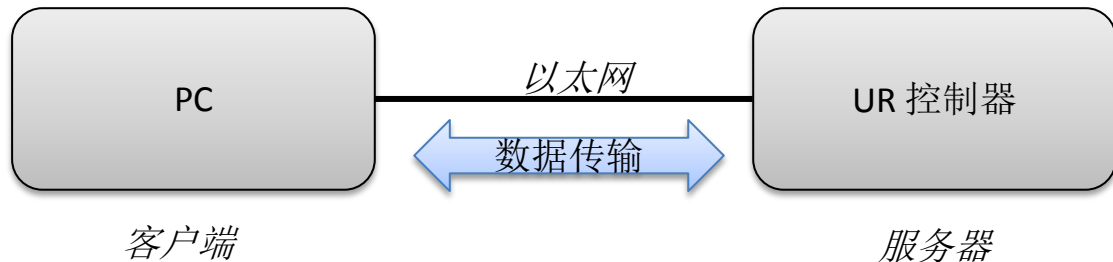
10

故障处理流程



## 什么是客户端界面？

- 什么是客户端界面？
  - 服务器仅仅运行在机器人控制器
- 功能性
  - 服务器连续发出机器人状态的数据流
  - 客户端能发出机器人脚本（URScript）指令给服务器
    - （服务器总是等待输入的指令）





## 有效的界面

- 有效的界面

	初级	第二级	实时
传输（Transmit）	机器人状态和附加信息	机器人状态信息	机器人状态信息
接收（Receive）	脚本指令	脚本指令	脚本指令
端口号（Port no.）	30001	30002	30003

- 实时客户端界面比初级/第二级更快



## 传输数据流

- 数据流
  - 传输数据流意味着包含大量的数据
  - 主要用于GUI和控制器之间通讯
  - 需要写一个程序从接收的数据中提取请求的数据
  - 技术支持网站有更多信息
  - 接收数据需要许多工作，并不是最好的选项



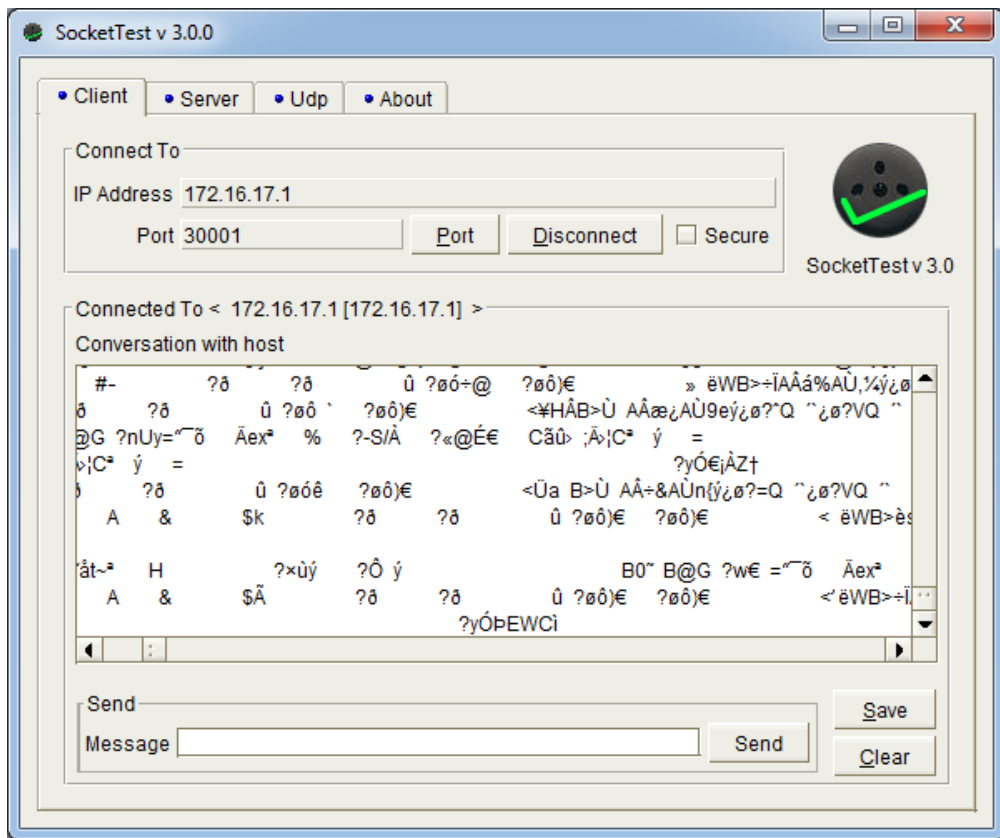
## 发送指令

- 发送到客户端界面
  - 从远程上位机用客户端界面发送脚本指令更有用
  - 在接收时各个指令按顺序执行
  - 发一个函数并且稍后执行它
  - 不通过GUI，能够用脚本写一个程序并且执行它



## 发送指令

- 打开 SocketTest
  - 选择客户端 (Client)
  - 设置端口号 port no. 30001
  - 连接
- 发送一个简单指令，例如：
  - popup()
  - set\_digital\_out()
  - move()
  - speedj()



- 参考“脚本手册”，有对脚本代码语法的详细解释



## 培训练习

- 通过 SocketTest 发送指令到机器人客户端执行以下操作（参考脚本手册，注意正确的语法）：
- 用下列指令在基座坐标系下移动机器人TCP：
  - Z 速度（velocity） = 0.3 m/s
  - X, Y, RX, RY, RZ 所有速度 = 0 m/s.
  - 延时 Timeout = 1 second
  - 加速度（Acceleration） = 0.5m/s<sup>2</sup>
- 用下列指令在工具坐标系下移动机器人TCP：
  - $Z = Z + 0.1 \text{ m}$  （当前TCP Z值增加 0.1 m ）
  - 在工具空间直线运动
  - 速度（Velocity） = 0.2 m/s
  - 加速度（Acceleration） = 0.5m/s<sup>2</sup>
  - 提示：在运动指令中用 `get_actual_tcp_pose()` 和 `pose_trans()`





1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

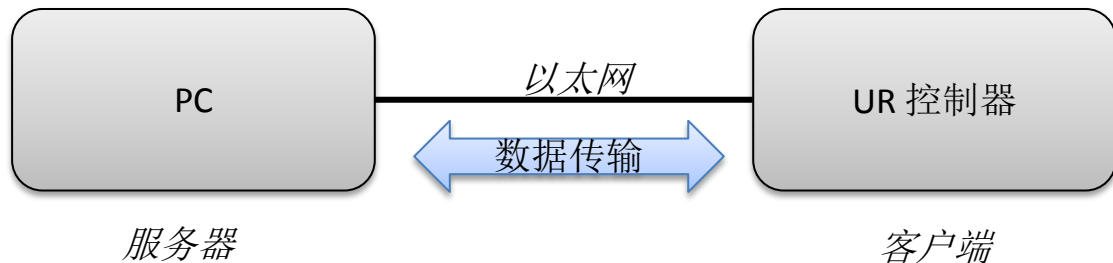
10

故障处理流程



## 什么是 Sockets?

- Socket 通讯
  - 简单的 TCP/IP socket 通讯对机器人和其他设备传输数据很有用
  - 机器人=客户端, 其他设备=服务器
- 功能性
  - 服务器总是监听来自客户端的通讯请求
  - 开/关 sockets 指令
  - 发送/接收不同数据类型指令

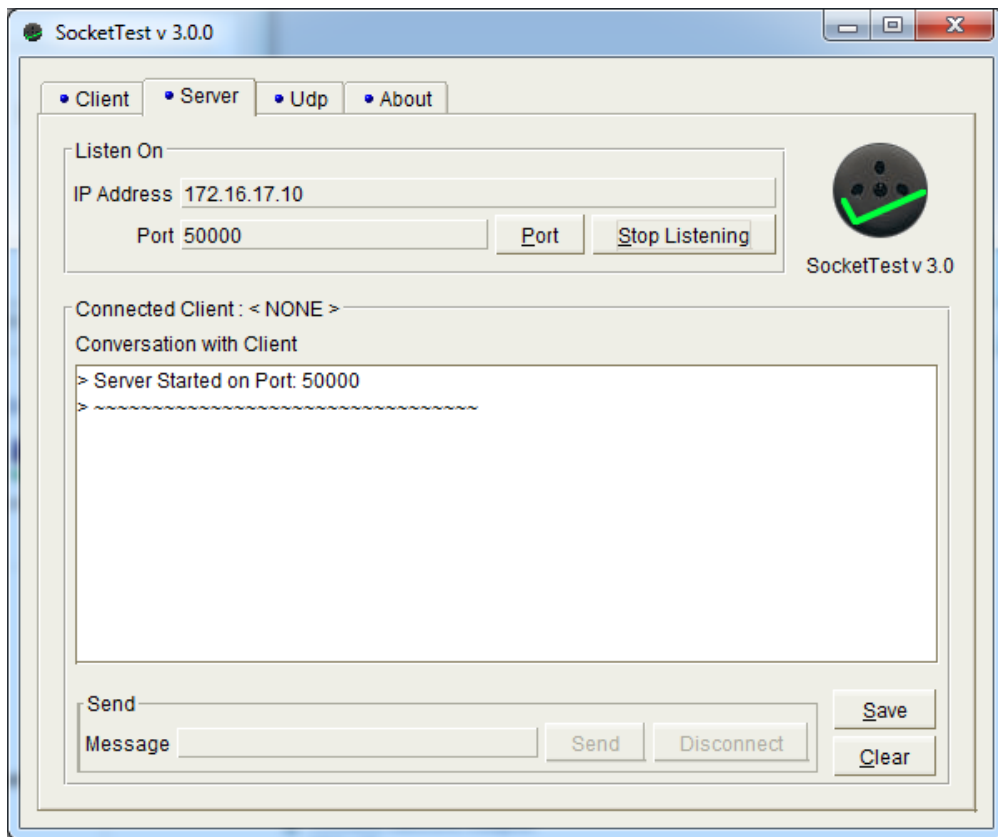




## 启动服务器

- 打开 SocketTest

- 选择 Server
- 定义IP地址
- 定义端口号
- 启动监听



- IP 地址与PC地址相同



## 打开一个 Socket

- `socket_open(address, port, socket_name)`
  - 地址 (address)
  - 端口 (port)
  - Socket名字 (socket\_name)
  - 如果程序只打开一个socket, socket名字不需要
- 返回值
  - 如果socket打开成功, 返回TRUE
  - 如果失败, 返回FALSE

**socket\_open(address, port, socket\_name=' socket\_0' )**

Open ethernet communication

Attempts to open a socket connection, times out after 2 seconds.

**Parameters**

address: Server address (string)

port: Port number (int)

socket\_name: Name of socket (string)

**Return Value**

False if failed, True if connection successfully established

Robot Program

```
socket_open("172.16.17.10", 50000)
```



## 发送一个字符串

- `socket_send_string(str, socket_name)`
  - 字符串 (`str`)
  - 名字 (`socket_name0`)
- 返回值
  - 无

Robot Program

```
socket_send_string("test")
```

**`socket_send_string(str, socket_name=' socket_0')`**

Sends a string to the server

Sends the string <str> through the socket in ASCII coding. Expects no response.

**Parameters**

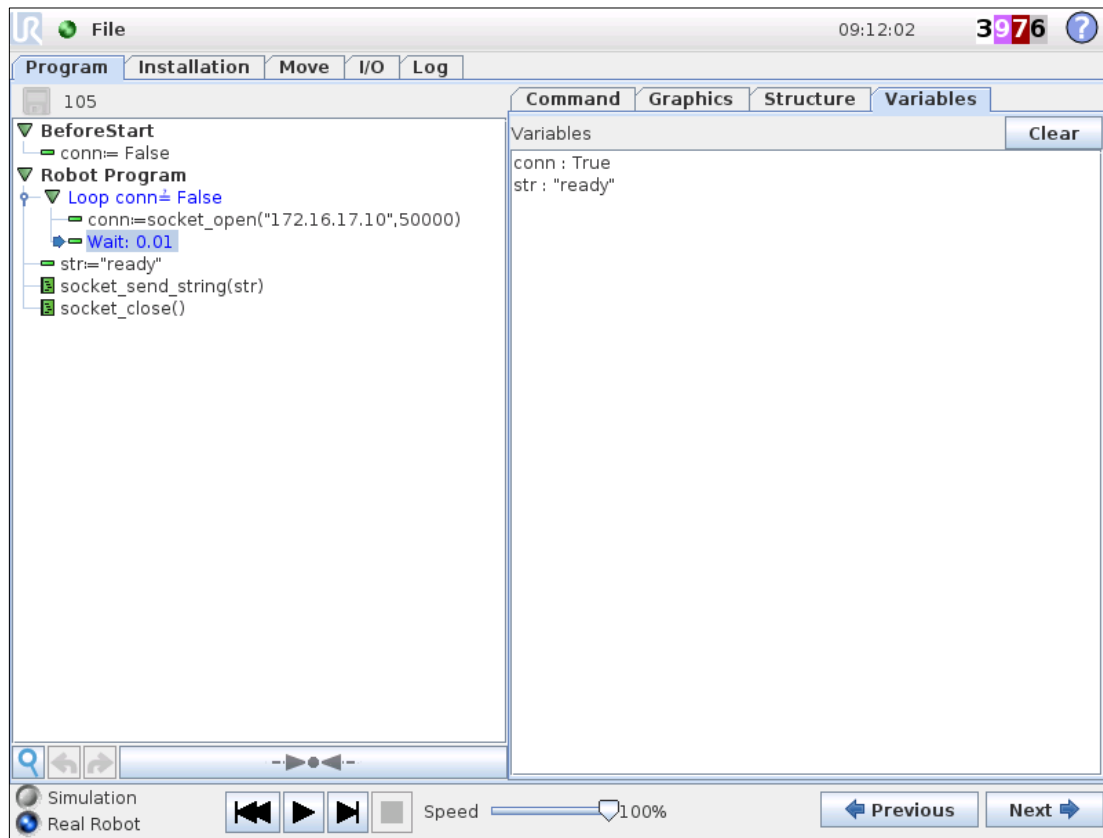
`str`: The string to send (ascii)

`socket_name`: Name of socket (string)



## 发送字符串例子

- 开启服务器
- 发送字符串到服务器
  - 打开 socket
  - 等待 socket 打开
  - 发送字符串
  - 关闭 socket
- 验证服务器的值





## 接收一个响应

- `Socket_read_ascii_float (n, socket_name)`
  - 数字 (number)
  - 名字 (socket\_name)
- 返回值
  - 浮点列表+1个整数
  - (在列表中第一个数是整数, 并且定义列表中有有效浮点数的数量)

Robot Program

```
socket_read_ascii_float(3)
```

**`socket_read_ascii_float(number, socket_name='socket_0')`**

Reads a number of ascii formatted floats from the TCP/IP connected. A maximum of 30 values can be read in one command.

```
>>> list_of_four_floats = socket_read_ascii_float(4)
```

The format of the numbers should be in parantheses, and seperated by ",". An example list of four numbers could look like "( 1.414 , 3.14159, 1.616, 0.0)".

The returned list contains the total numbers read, and then each number in succession. For example a read\_ascii\_float on the example above would return (4, 1.414, 3.14159, 1.616, 0.0).

A failed read or timeout after 2 seconds will return the list with 0 as first element and then "Not a number (nan)" in the following elements (ex. (0, nan., nan, nan) for a read of three numbers).

### Parameters

`number:` The number of variables to read (int)  
`socket_name:` Name of socket (string)

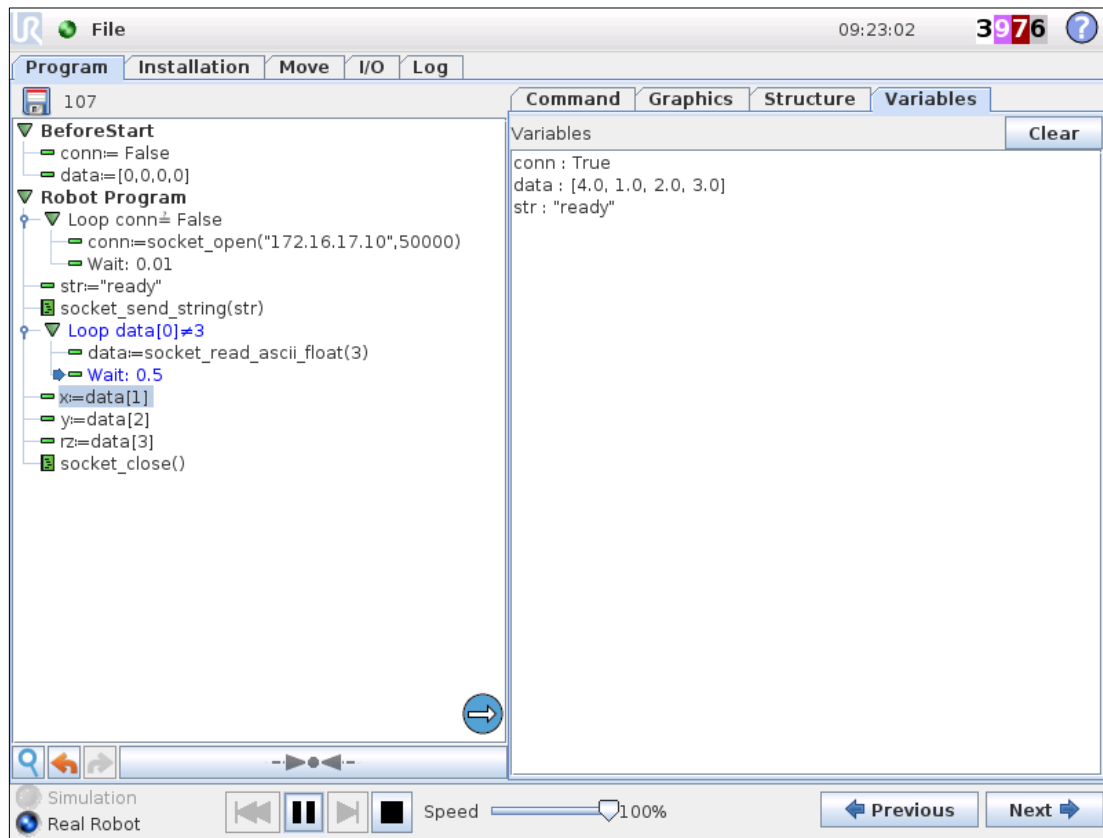
### Return Value

A list of numbers read (list of floats, length=number+1)



## 例子：接收一个响应

- 继续
  - 打开 socket
  - 等待socket被打开
  - 发送字符串
  - 给服务器发送请求
  - 读三个浮点数
  - 等待三个浮点数收到
  - 在变量中保存浮点数 \*
    - x
    - y
    - rz
  - 关闭socket



\* 服务器可以是视觉相机





## 数据格式

- 有效数据格式
  - 字符串 (String)
  - 浮点 (Float)
  - 整数 (Integer)
  - 比特 (Byte)
  - 列表 (List)
- 全部socket指令参考“脚本手册”
  - 用 `set_var()` 和 `get_var()` 允许变量名字和整数值在单个信息中传输



## 基于以太网TCP/IP通讯总结

通讯协议	端口号	软件	客户端	服务器	功能
Modbus	502	无	UR	I/O模块	扩展I/O
Modbus	502	Ananas	PC	UR	读UR参数, I/O状态
FTP	22	FileZilla	PC	UR	远程传输文件
Dashboard	29999	SocketTest	PC	UR	远程控制UR/读UR状态
客户端界面	30001 30002 30003	SocketTest	PC	UR	数据流（解析）
Socket		SocketTest	UR	PC/相机	字符串/数组



## 培训练习 1

- 写一个程序：
  - 和 SocketTest 服务器建立连接
  - 通过 socket 接收两个浮点
  - 把他们加在一起
  - 把结果返回给服务器
  - 等待另一对浮点
- 例子：
  - 服务器发送: (5,2)
  - 程序返回: 7

Received = (x, y)

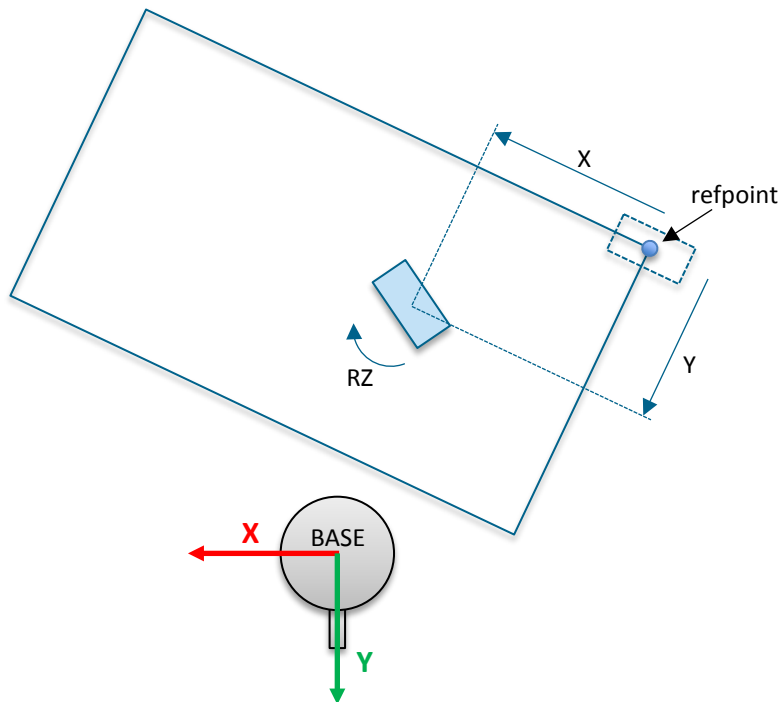
Added = x+y

Return: Added



## 培训练习 2

- 写一个程序通过socket模拟机器视觉坐标系，并且移动到接收的数据的位置
  - 通过socket发送信息请求数值，然后接收3个浮点数值
  - 根据变量部分练习的pose\_trans() 运算结果运动到新的位置
- 数据格式
  - 发送字符串 “ready” 给服务器
  - 读3个 ascii 码浮点数:
  - x
  - y
  - rz





1

脚本代码

2

变量

3

特征

4

高级TCP使用

5

Modbus 服务器

6

FTP 服务器

7

Dashboard 服务器

8

客户端界面

9

Socket 通讯

10

故障处理流程



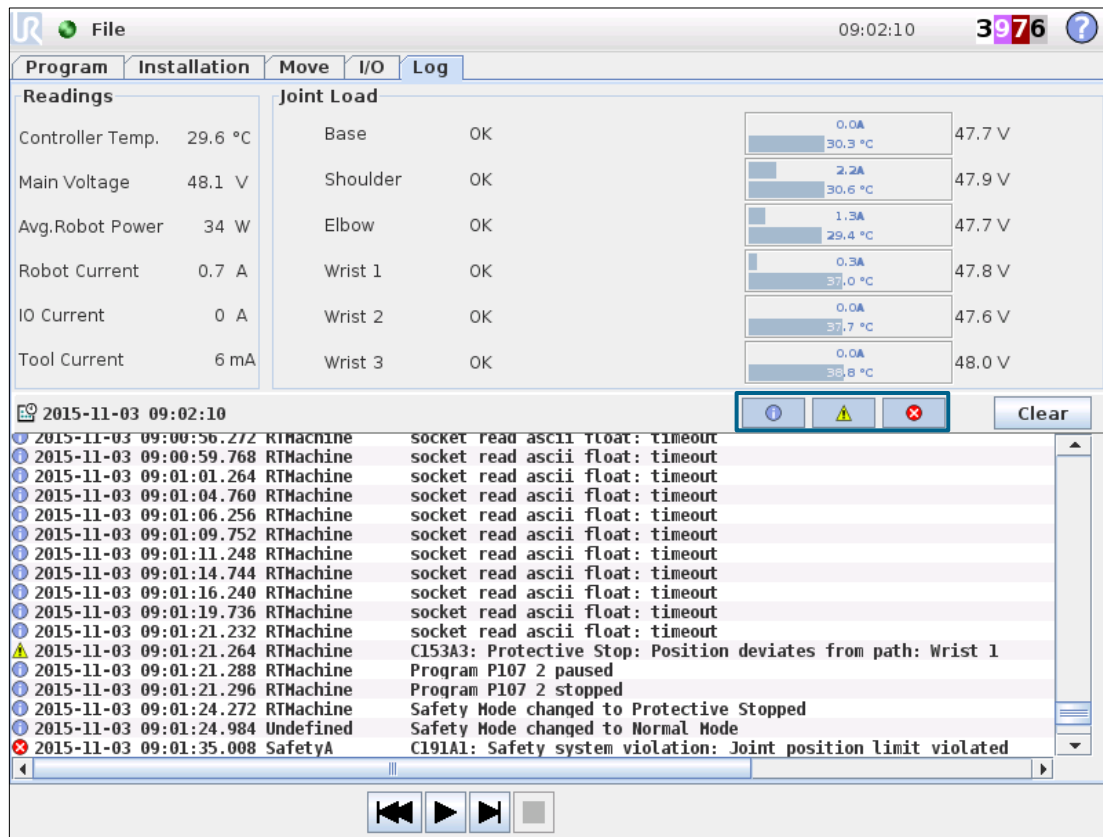
## 日志文件分析

### 日志历史

- 包含
  - 信息
  - 警告
  - 错误
- 可以用显示/隐藏分别显示不同类型日志
- 可以通过魔法文件或FTP备份日志文件

### 日志分析

- 日志信息的含义是什么？
- 如何改善程序结构？

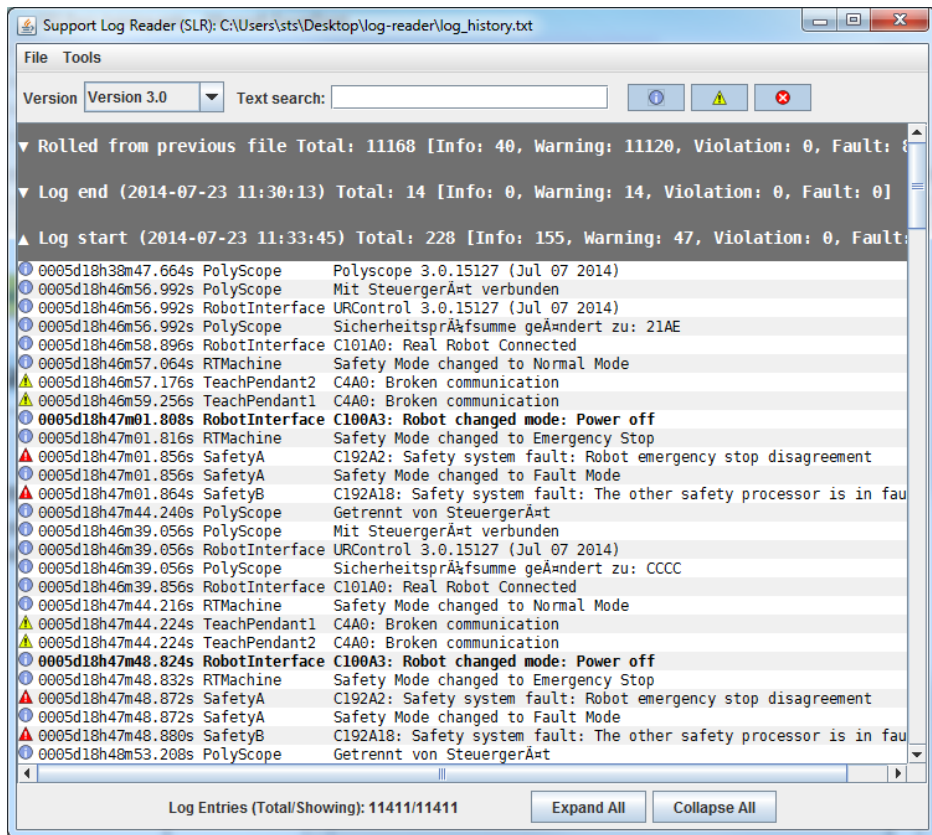




## 日志阅读软件（Support Log Reader）

- 日志阅读软件（SLR）

- 读日志文件
- 语言转换
- 输出到 csv-file
- 过滤搜索
- 支持
  - CB3 文件格式
  - CB2 文件格式



- 错误代码和解决方案参考“服务手册”



## 监测力过高

- 错误代码



- C114A0: 监测力过高警告

- 导致原因

- 警告在程序中加速度过高或负载设置不正确
  - 单独只有这个信息不影响机器人运行，但可能导致：
    - URControl C100A4: 机器人状态改变：安全停止
    - URControl C113A0: 力限制保护停止
  - 这些将停止程序运行



- 如何修复

- 检查TCP负载设置
  - 减少程序速度和加速度





## CPU处理器负载过高

- 错误代码



- C116A216: 处理器负载过高警告

- 导致

- CPU 处理器过载警告



- 单独只有这个信息不影响机器人运行，但可能导致

- URControl C100A4: 机器人模式改变，安全停止
    - R 0002d02h47m51.616s SafetySys C19A0: 主机没有数据传输给关节
    - R 0002d02h47m51.624s SafetySys C29A1: 检测到PC到机器人之间以太网数据包丢失

- 这些将停止机器人程序

- 如何修复

- 减少if 的数量，如可能用if ... else if ... else 替代
  - 增加等待或 sync() 指令，解放CPU时间
  - 确保没有大量的MODBUS通道被高速刷新



## 检测到无限循环

- 错误代码
  - 运行时间错误：在程序中存在无限循环
- 导致
  - 如果程序运行不断循环且没有等待时间
  - 程序将停止
- 如何修复
  - 在任何循环中加一个短的等待时间
  - 用 `sync()` 脚本也可以



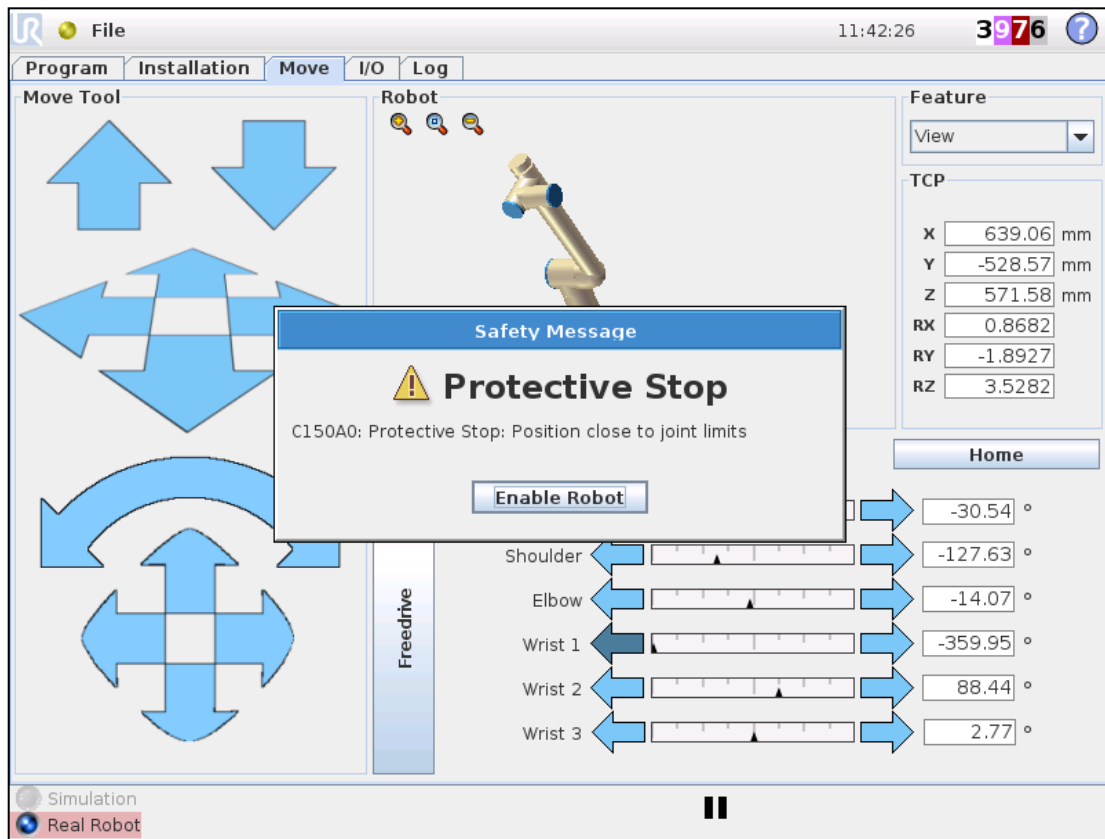
## 奇异点（Singularities）

- 错误代码
  - C154A20: 保护性停止, 在奇异点位置
- 导致
  - 当在笛卡尔坐标系控制机器人移动时（move1/movep），可能会出现奇异点问题
  - 当以下情况会发生奇异点：
    - 多轴参与同一个运动（不必要的轴多余的轴运动）
    - 在笛卡尔空间机器人移动时，机器人在某些位置关节移动速率非常高
- 如何解决
  - 在靠近奇异点时，使用MoveJ指令
  - 如果需要在笛卡尔空间运动，调整路点位置姿态
  - 修改应用布局，调整路点位置
  - 修改工装夹具，改变工具角度



## 关节限制

- 错误代码
  - C150A20: 保护性停止  
接近关节限制位置
- 导致
  - 表明机器人某一个关节到它的极限位置
- 如何修复
  - 打开移动窗口
  - 检查关节运动部分
  - 如果某一个关节在360度极限位置, 返回范围的中心
  - 重新示教MoveJ路点





UNIVERSAL ROBOTS

优傲机器人贸易（上海）有限公司

上海恒基688广场2008室

南京西路688号

上海市静安区

20060 上海

电话: +86 21 61326299

邮编: 200041

邮件: [support.china@universal-robots.com](mailto:support.china@universal-robots.com)

[www.universal-robots.com](http://www.universal-robots.com)

