



# UNIVERSAL ROBOTS

## Basic Training



**Company history**

**UNIVERSAL ROBOTS**



## Company history

2005 – 2008: Development of UR5

2008: First commercial sales of UR5

2009: Distribution network established in Europe

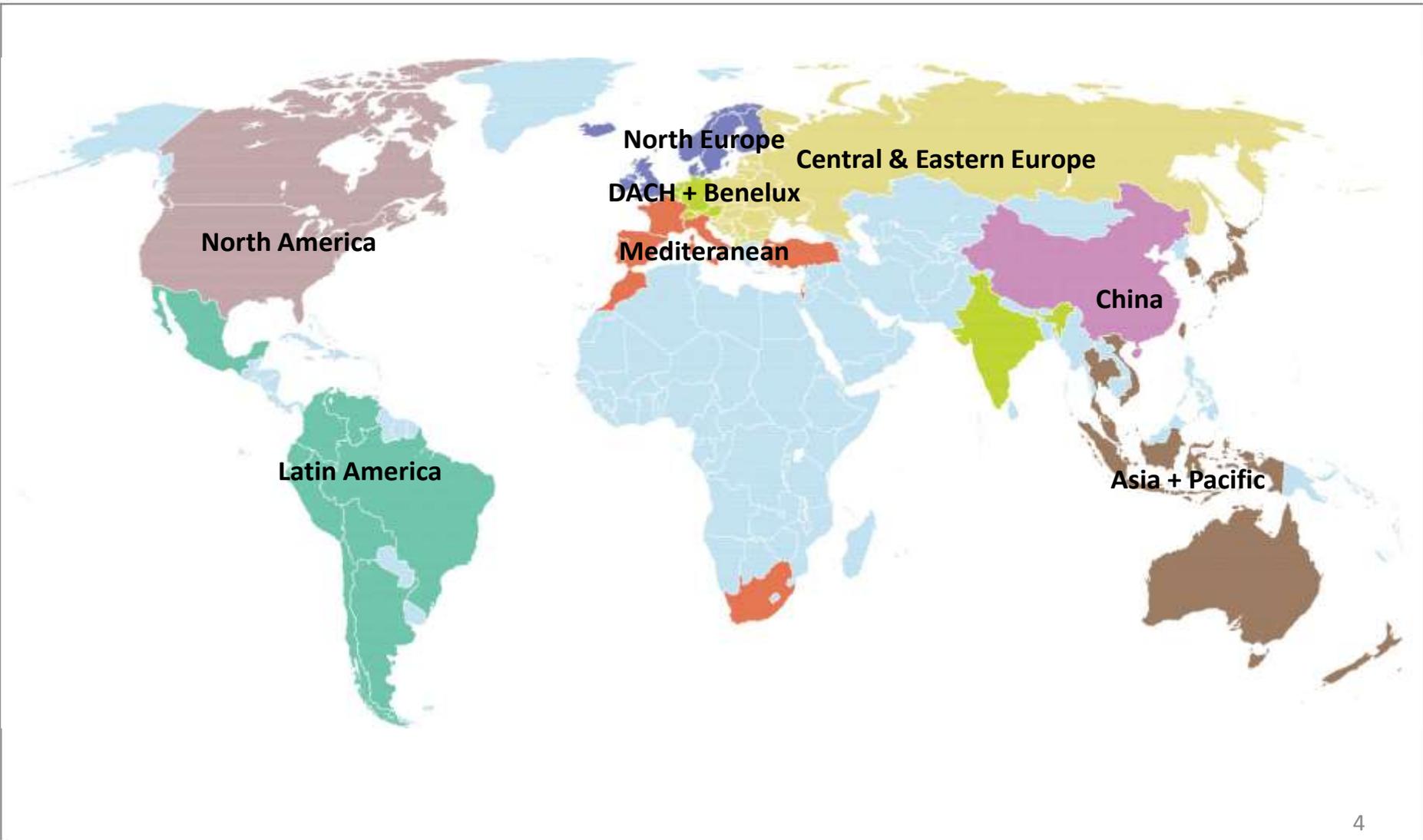
2011: Distribution network established in Asia

2012: Launch of UR10  
Distribution network established in US

2013: Subsidiaries established in NY and Shanghai  
Distribution network established in Latin America

2014: Launch of UR5 (CB3) and UR10 (CB3)  
Subsidiary established in Barcelona

## Global distribution 2014 regions



welcome to

Universal Robots **Basic Training**

- » A practical hands-on experience
- » Sample programs saved during training
- » Take notes
- » 30 min. examination at end of training

Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

wizards

Modbus TCP

Service

safety standards

Adjustable safety

1

## Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

Safety standards

Adjustable safety

## what's in the box?

- Box 1
  - Robot arm
- Box 2
  - Control box
  - Mains cable
  - Mounting brackets
  - Tool cable
  - UR laserpen
  - Manuals
  - Production test certificate



- Robot arm and control box comes in ESD-approved bags

# 1 Hardware

## What's in the box?

- Box 1
  - Robot arm
- Box 2
  - Control box
  - Mains cable
  - Mounting brackets
  - Tool cable
  - UR laserpen
  - Manuals
  - Production test certificate



## Robot arm

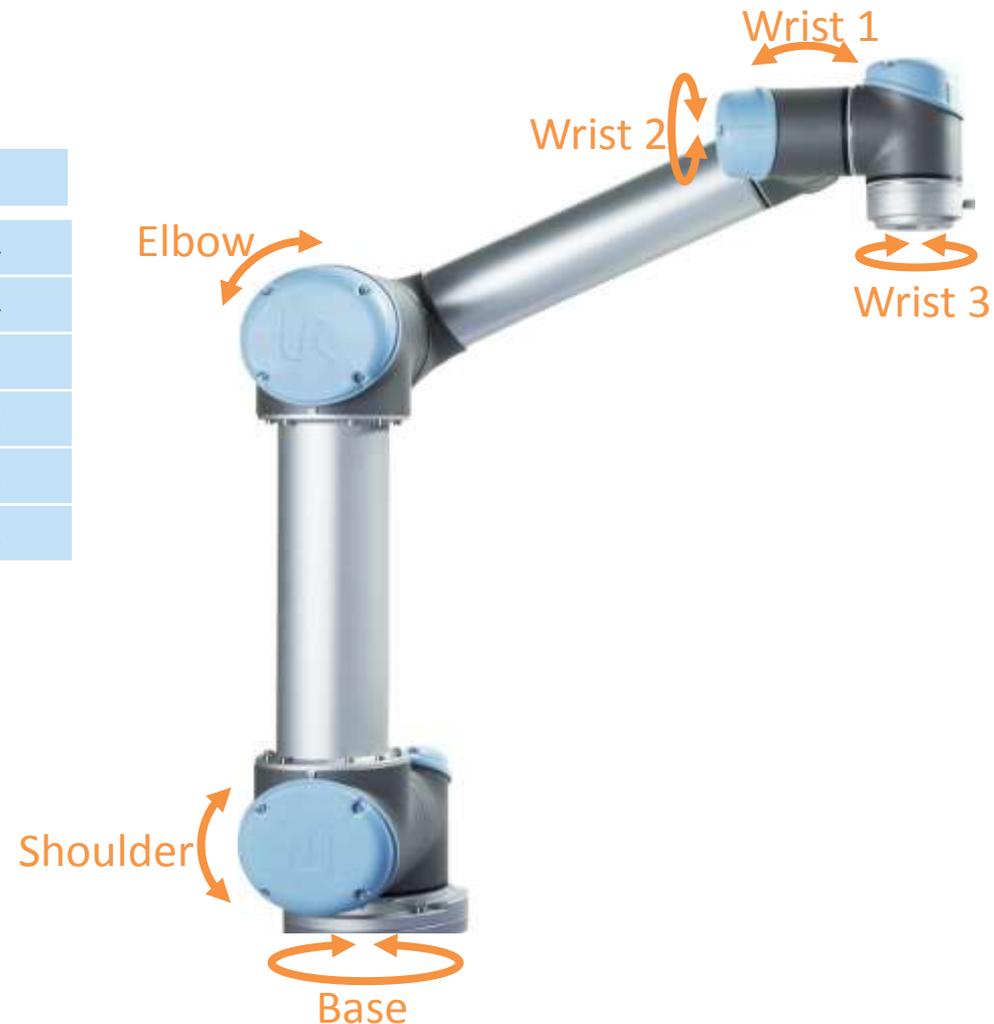
- Manipulator design
  - 6 axes
  - Articulated robot (*closely resembles human arm*)
  - Modular design
  - +/- 360° freedom
  - 3-phase AC servo motors
- Available types
  - UR5
  - UR10



## Robot arm

- Joint designation and sizes

	UR5	UR10
Base	Size 3	Size 4
Shoulder	Size 3	Size 4
Elbow	Size 3	Size 3
Wrist 1	Size 1	Size 2
Wrist 2	Size 1	Size 2
Wrist 3	Size 1	Size 2



# 1 Hardware

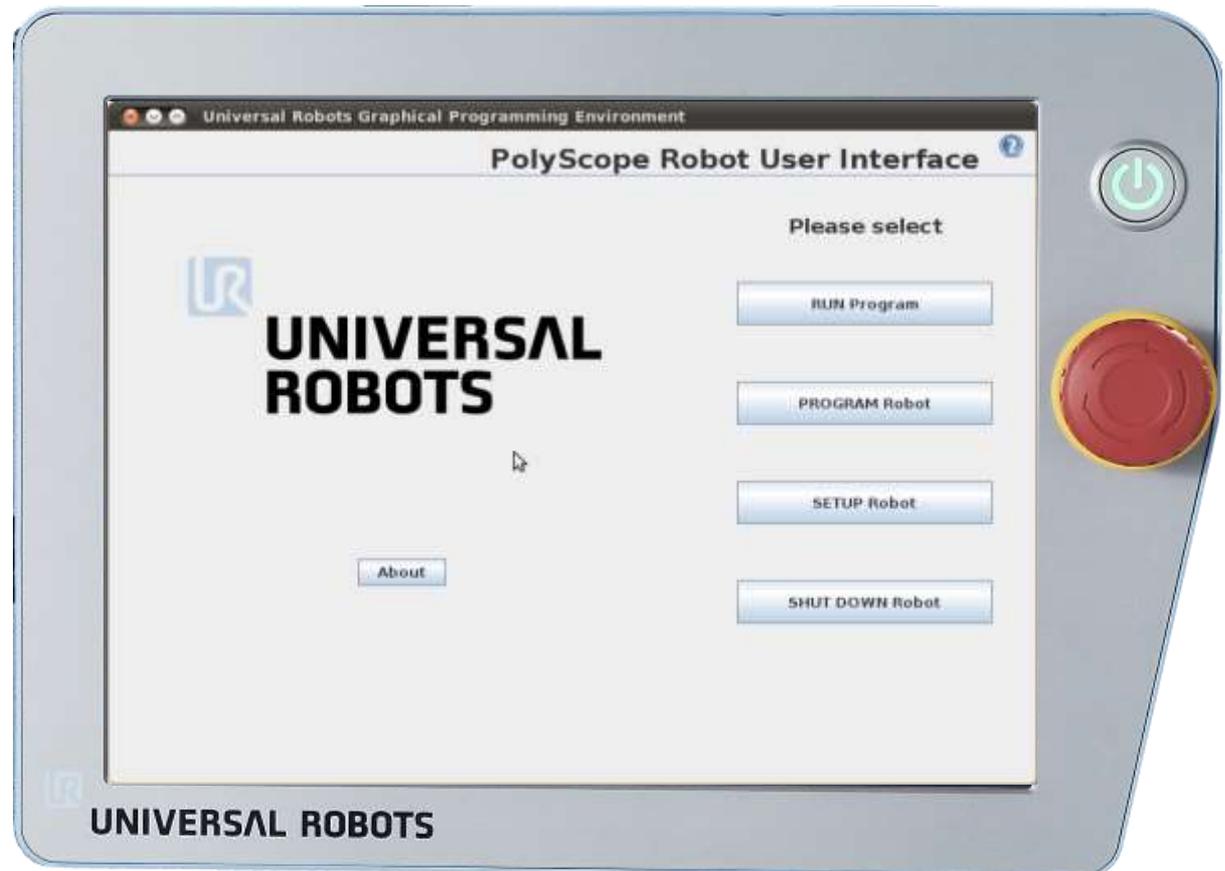
## Control box

- Contains
  - Flashcard with software
  - Power to robot arm
  - Safety system
  - Communication to peripheral devices
- Connectors
  - Power 220/110 Vac
  - Ethernet
  - USB
  - Robot arm



## Teach pendant

- Touch sensitive monitor
- TP includes
  - Power button
  - Emergency button
  - Teach button
  - USB connector



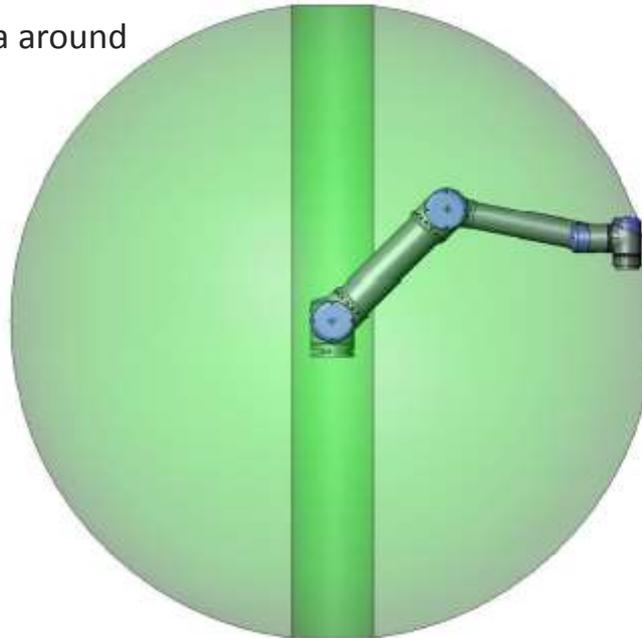
## Specifications

	UR5	UR10
<i>Payload</i>	5 kg.	10 kg.
<i>Reach</i>	850 mm	1300 mm
<i>Joint ranges</i>	+/-360°	+/-360°
<i>Repeatability</i>	+/-0.1 mm	+/-0.1 mm
<i>Joint max. Speed</i>	180°/sec	120°/sec and 180°/sec
<i>Tool max. speed</i>	1000 mm/sec	1000 mm/sec
<i>Weight</i>	18.4 kg	28.9 kg
<i>IP rating</i>	IP54	IP54
<i>Temp. range</i>	0-50°C	0-50°C
<i>Power supply</i>	100-240V AC, 50-60 Hz	100-240V AC, 50-60Hz

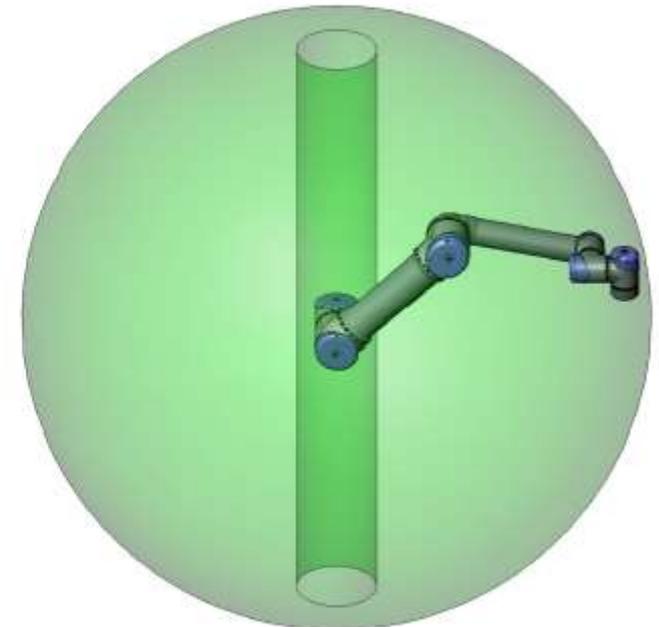
- Additional specs can be found on [www.universal-robots.com](http://www.universal-robots.com)

## Robot workspace

- UR5 workspace
  - Approximate sphere of  $\varnothing 170$  cm
- Limitation
  - Cylindrical area around center of Base



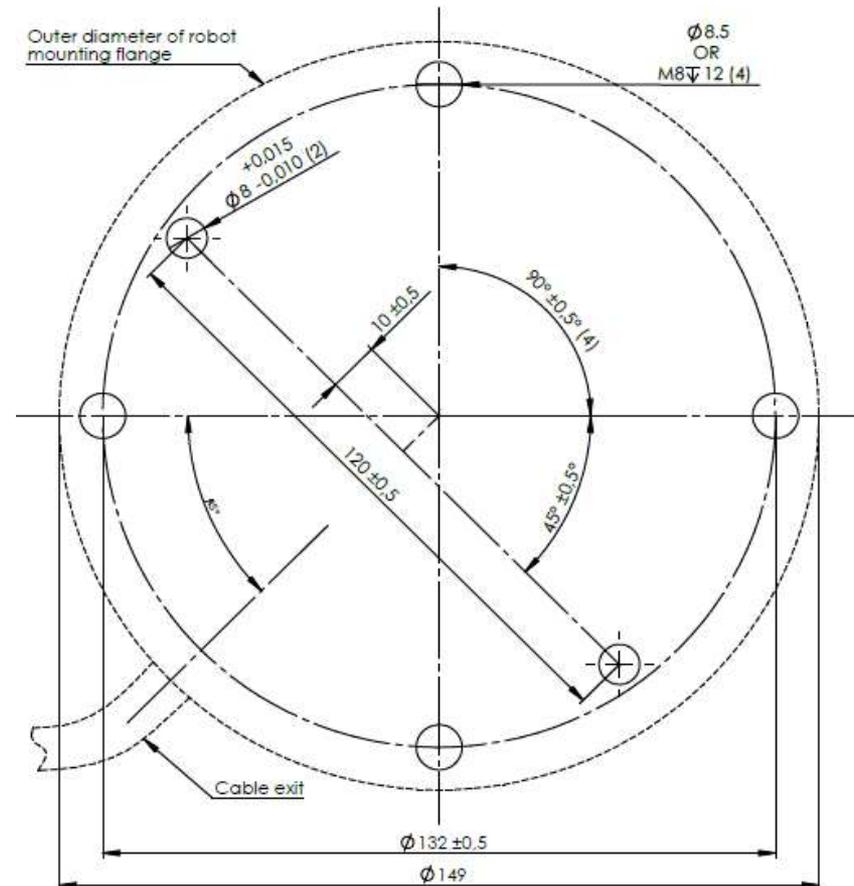
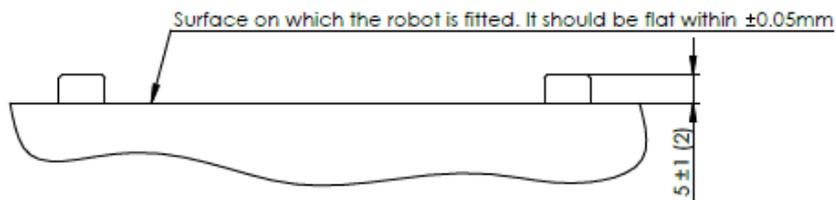
Front view



Tilted view

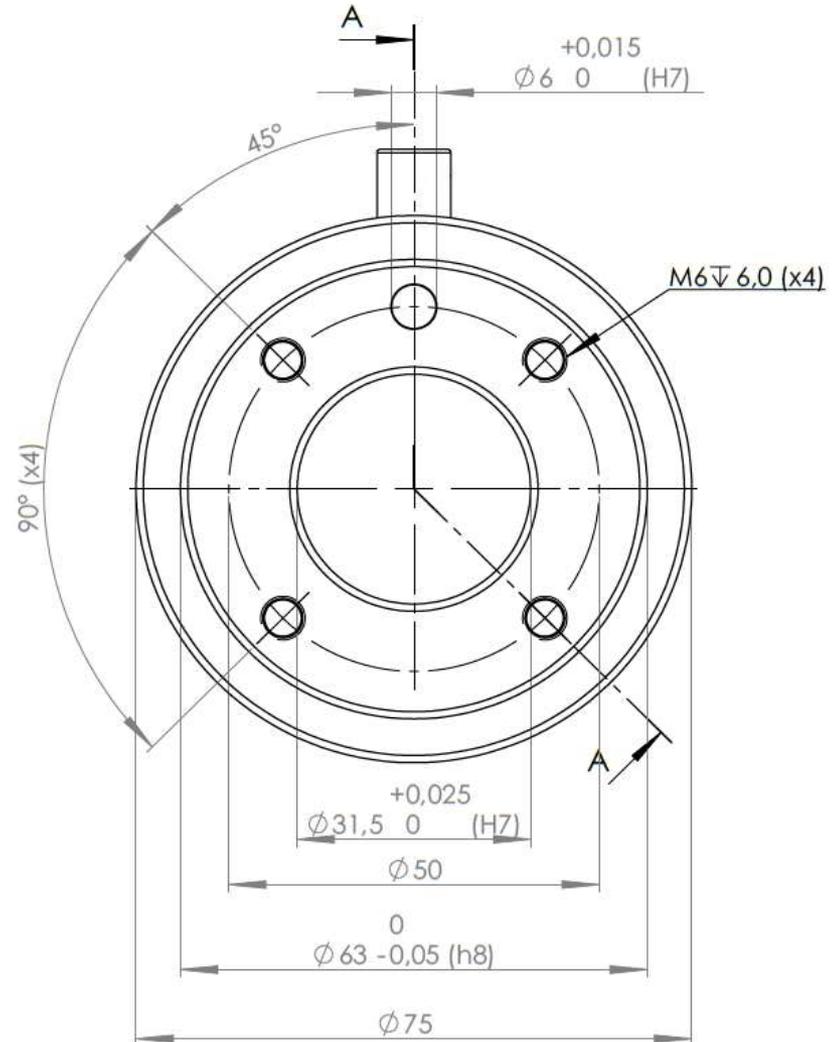
## Mounting the base

- Requirements
  - Solid surface
  - Footprint 149 mm
  - 4 pc. M8 bolts



## Mounting the tool

- Mounting standard
  - ISO 9409-1-50-4-M6
- Tool connector
  - 8 pin connector
  - Lumberg RKMV-8-354



Hardware

2

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

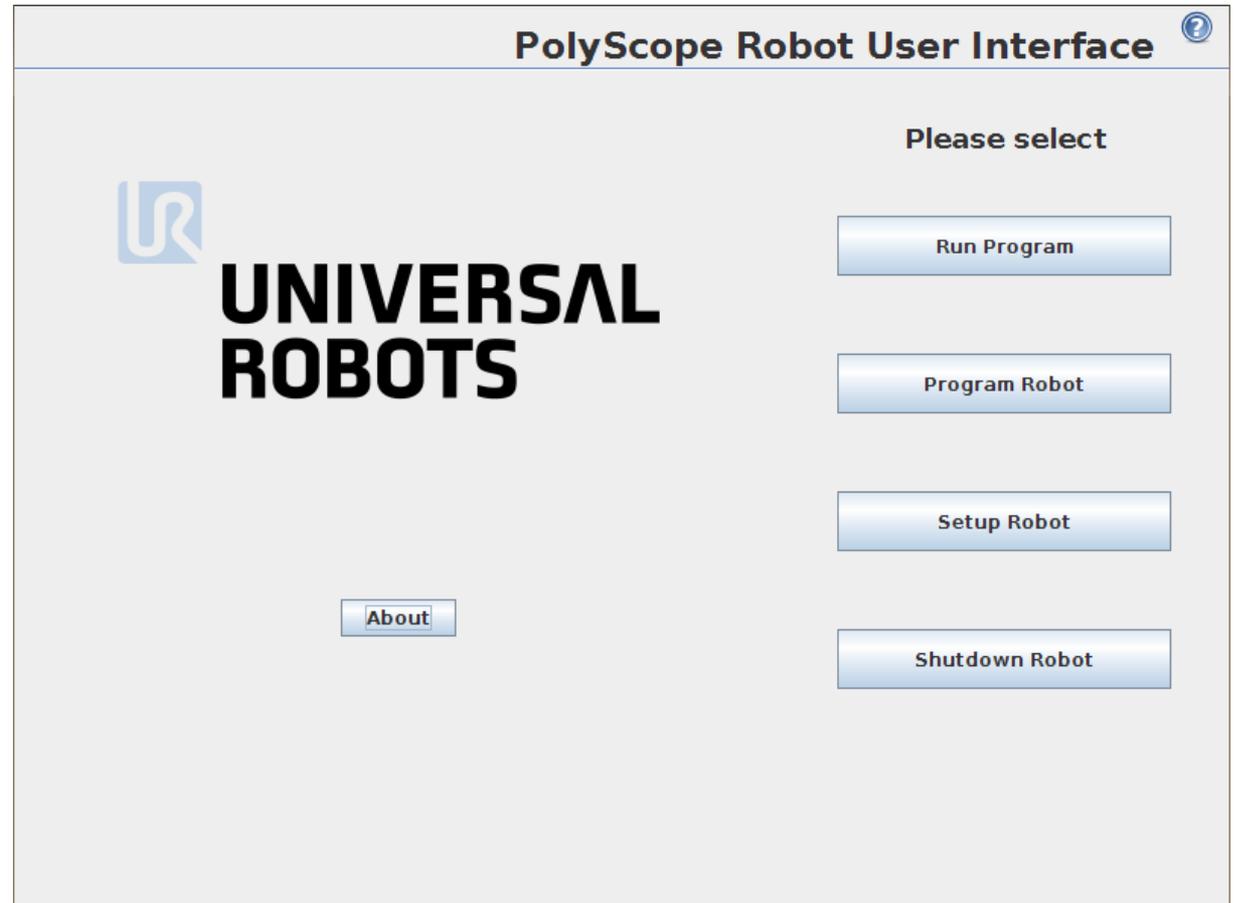
Service

Safety standards

Adjustable safety

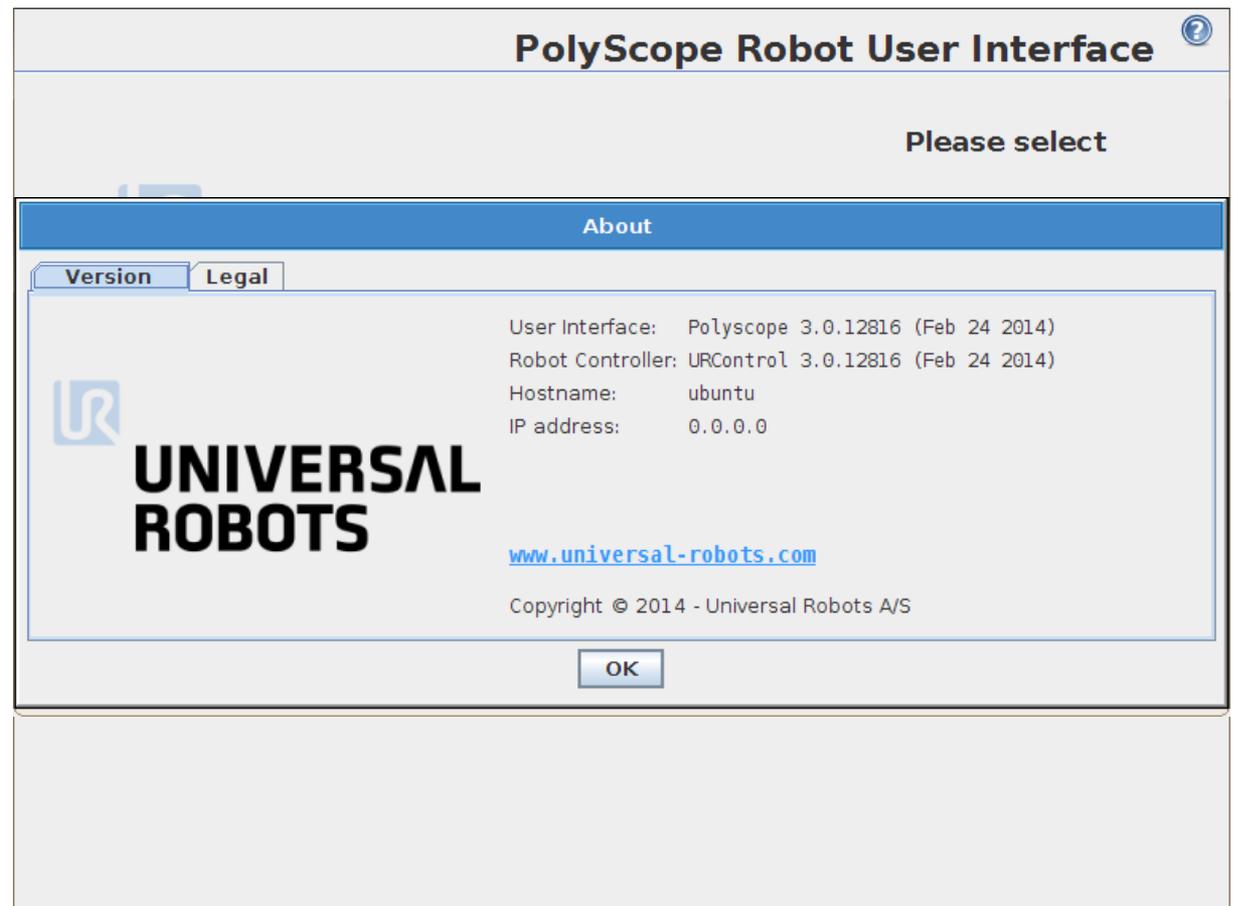
### Introduction to PolyScope

- PolyScope
  - Developed by UR
  - Free updates
- Operating system
  - Debian Linux



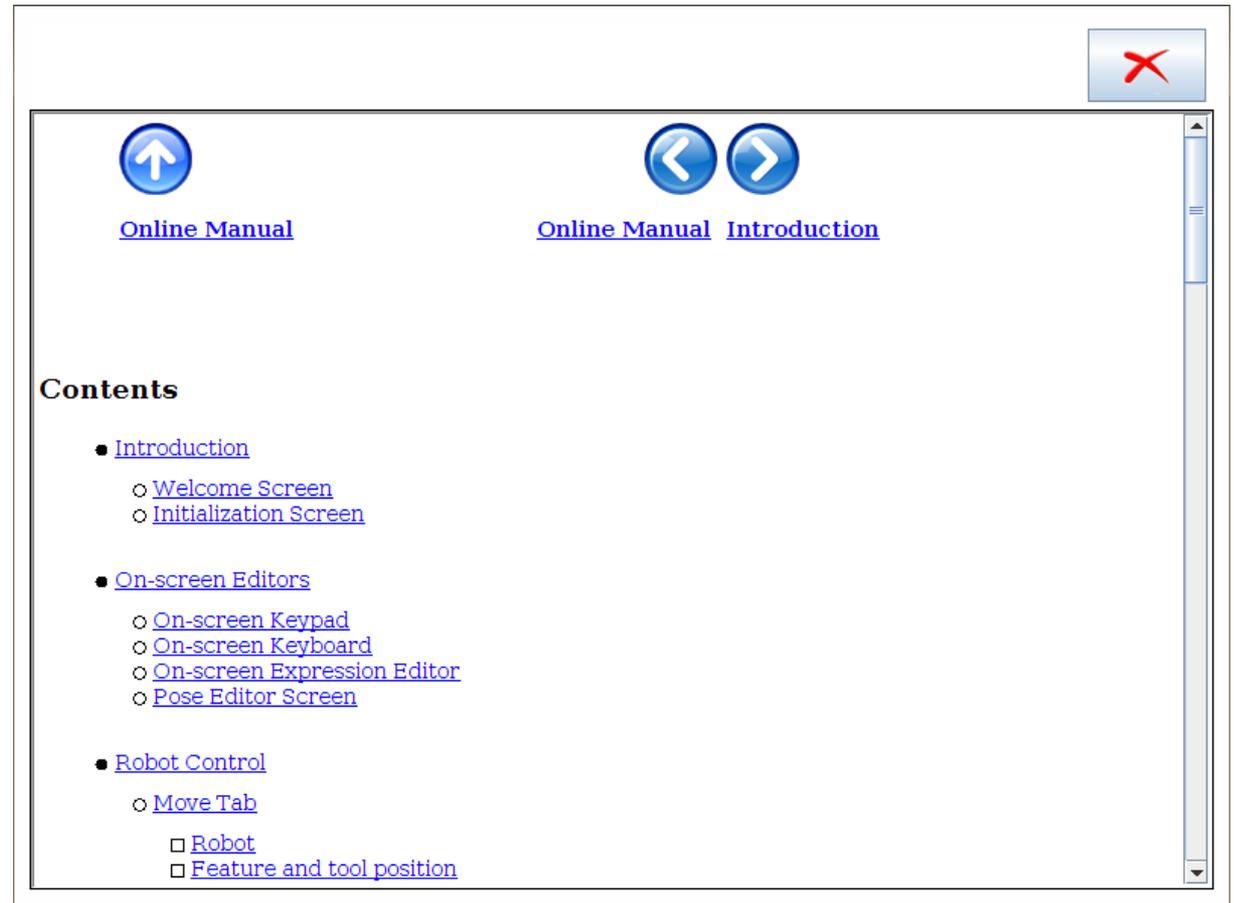
### About

- Information
  - Serial number
  - Software version
  - IP-address



### online manual

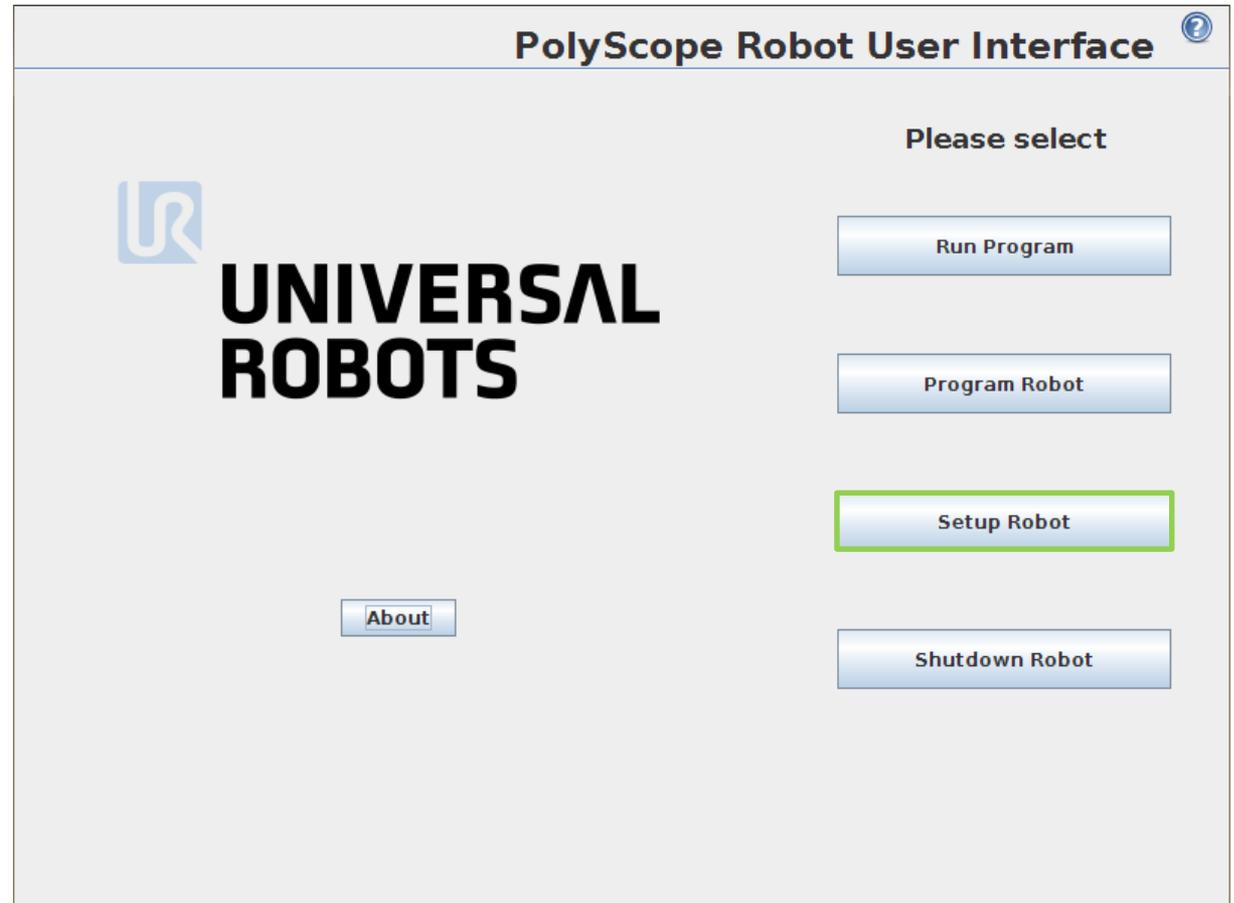
- Features
  - Displayed in selected language
  - "Light" version of software manual



## 2 Getting started

### Setup robot

- Setup robot
  - Adjust software settings



### Setup robot

- Software settings
  - Initialize Robot
  - Language and Units
  - Update Robot
  - Set Password
  - Calibrate Screen
  - Setup Network
  - Set Time



## 2 Getting started

### Initialize robot

#### Robot states

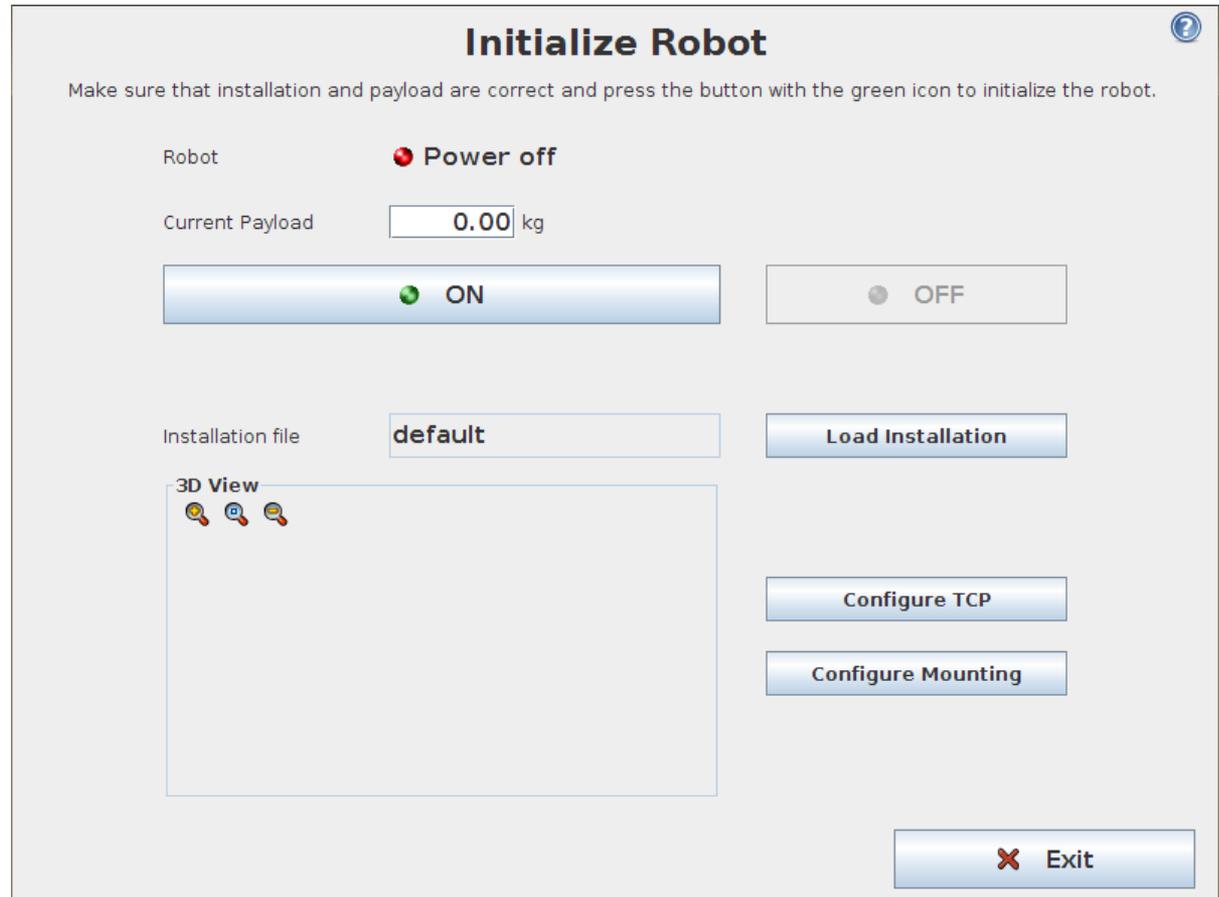
State	Power	Brakes
 Power off	OFF	Engaged
 Idle	ON	Engaged
 Normal	ON	Released

#### Initialize robot

- Check payload setting
- Click ON: enables power
- Click ON: releases brakes

#### Backdrive mode

- When close to collision, use Backdrive mode



**Initialize Robot**

Make sure that installation and payload are correct and press the button with the green icon to initialize the robot.

Robot  Power off

Current Payload  kg

 ON  OFF

Installation file  **Load Installation**

3D View   

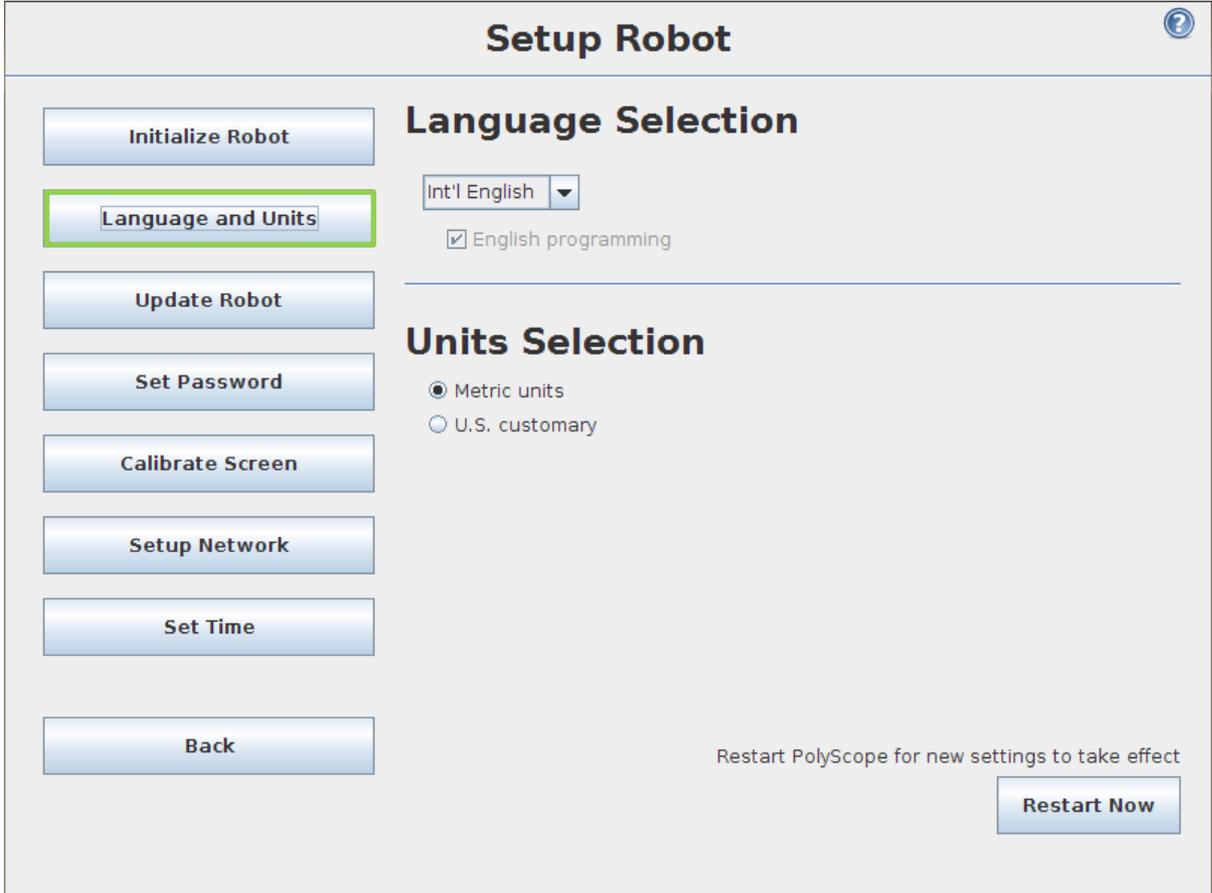
**Configure TCP**

**Configure Mounting**

 **Exit**

### Language and units

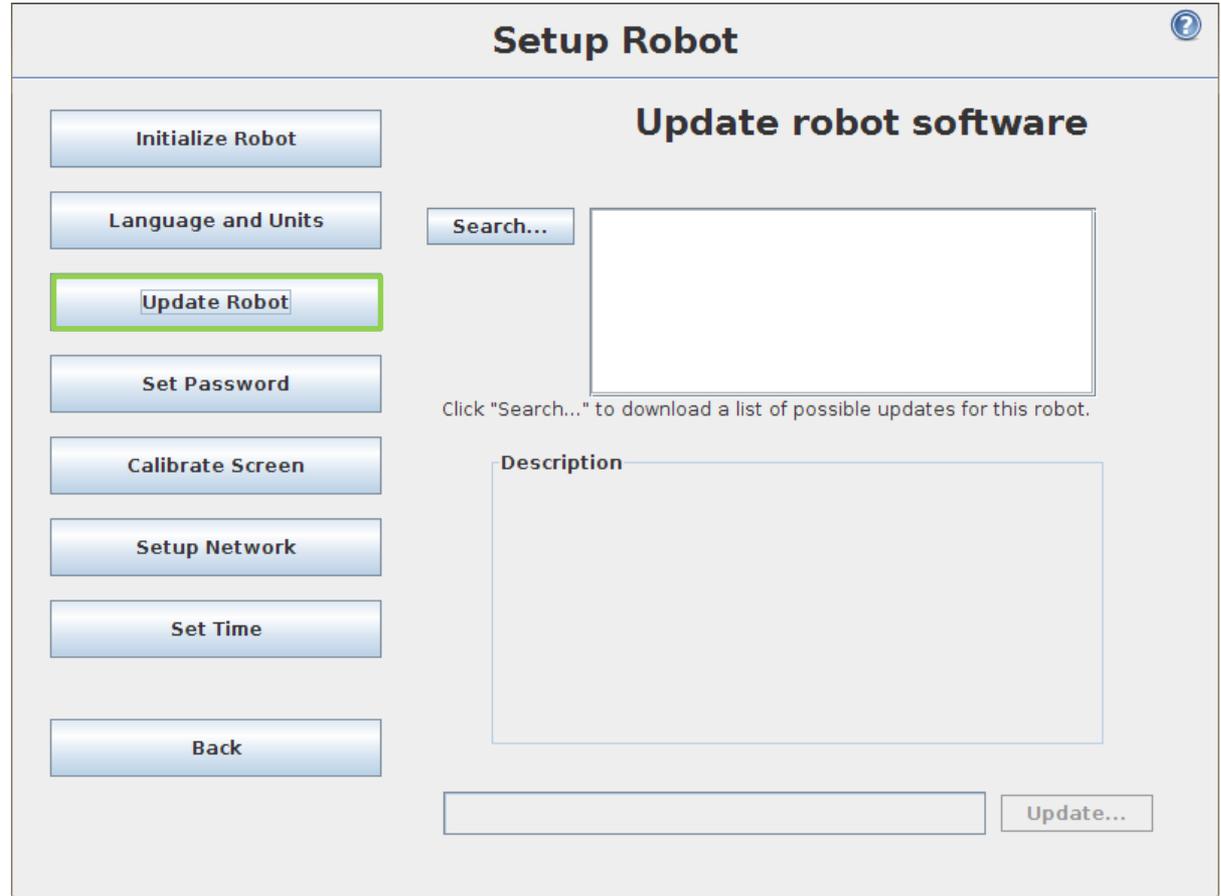
- Languages
  - 20 languages
  - English programming
    - Keeps command names in English language
- Units
  - Metric
  - U.S.



The screenshot shows the 'Setup Robot' interface. On the left is a vertical menu with buttons: 'Initialize Robot', 'Language and Units' (highlighted with a green border), 'Update Robot', 'Set Password', 'Calibrate Screen', 'Setup Network', 'Set Time', and 'Back'. The main area is titled 'Setup Robot' and contains two sections: 'Language Selection' and 'Units Selection'. In 'Language Selection', there is a dropdown menu set to 'Int'l English' and a checked checkbox for 'English programming'. In 'Units Selection', there are two radio buttons: 'Metric units' (selected) and 'U.S. customary'. At the bottom right, there is a note: 'Restart PolyScope for new settings to take effect' and a 'Restart Now' button.

### Update robot

- Robot software
  - Free updates
  - Download from UR support site

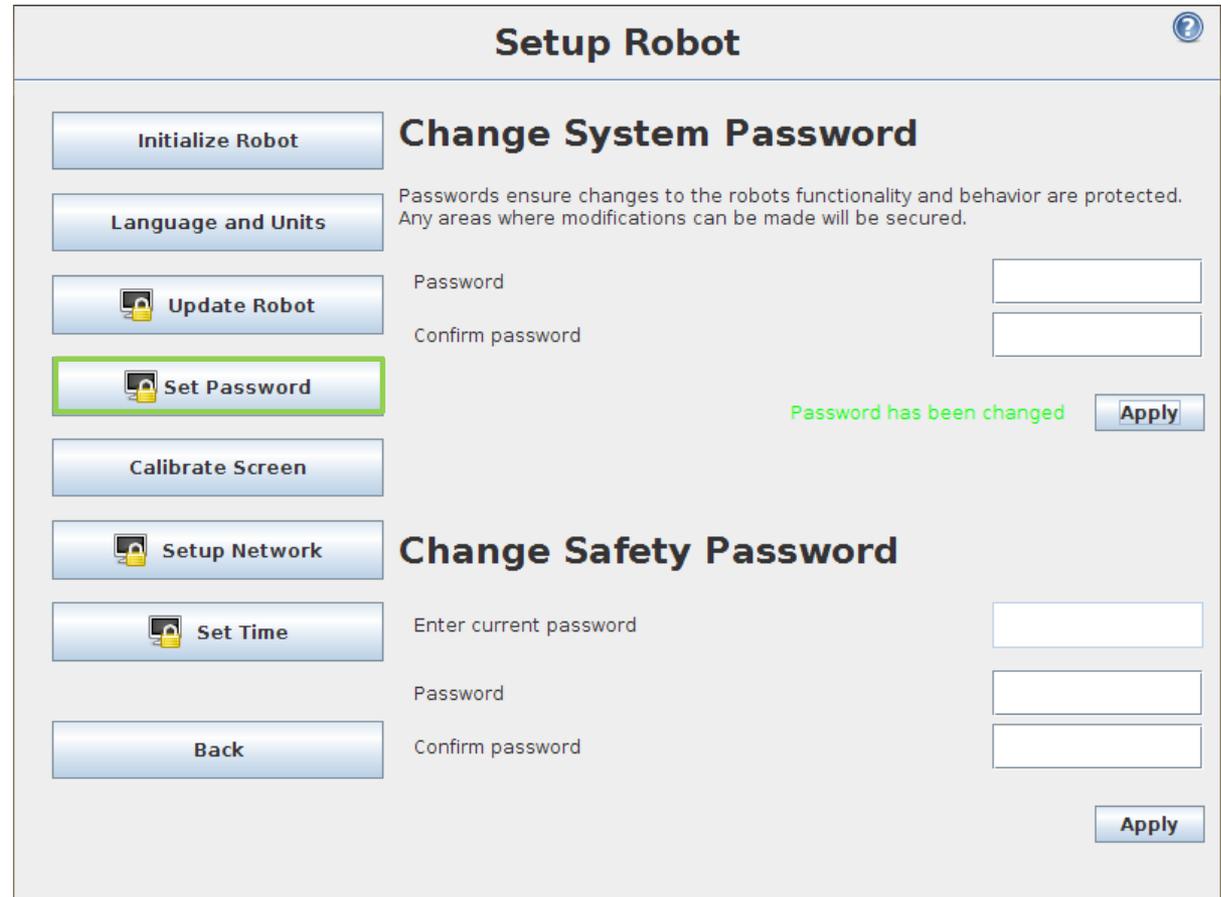


The screenshot shows the 'Setup Robot' interface. On the left is a vertical menu with buttons: 'Initialize Robot', 'Language and Units', 'Update Robot' (highlighted with a green border), 'Set Password', 'Calibrate Screen', 'Setup Network', 'Set Time', and 'Back'. The main area is titled 'Update robot software' and contains a 'Search...' button, a large empty text box, and a 'Description' label above another large empty text box. Below these is a 'Back' button and an 'Update...' button. A small help icon is in the top right corner.

- Procedure to update will be covered later in training session

## Set password

- System password
  - Limit access to parts of software
- Safety password
  - Required for modifying safety settings



**Setup Robot** ?

**Change System Password**

Initialize Robot

Language and Units

Update Robot

**Set Password**

Calibrate Screen

Setup Network

Set Time

Back

Passwords ensure changes to the robots functionality and behavior are protected. Any areas where modifications can be made will be secured.

Password

Confirm password

Apply

Password has been changed

**Change Safety Password**

Enter current password

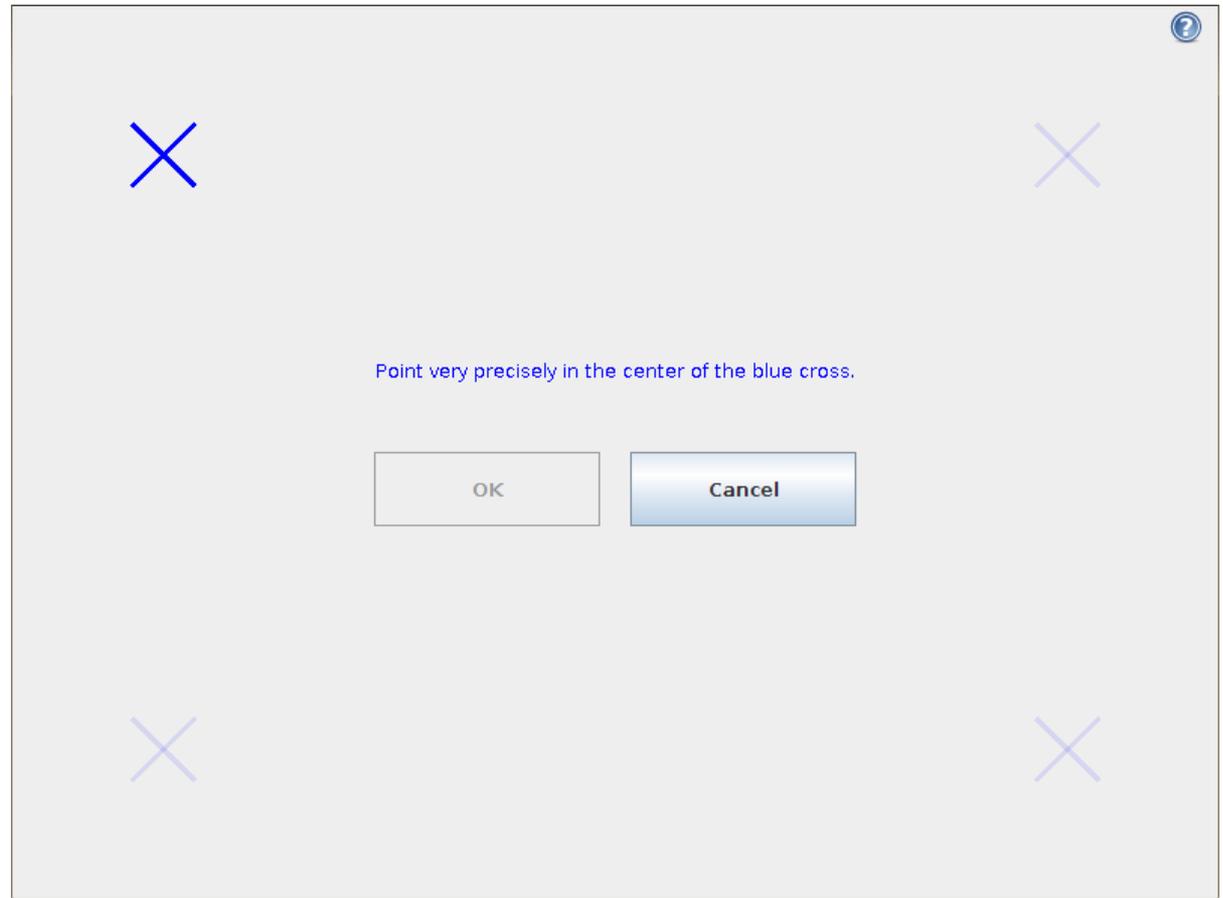
Password

Confirm password

Apply

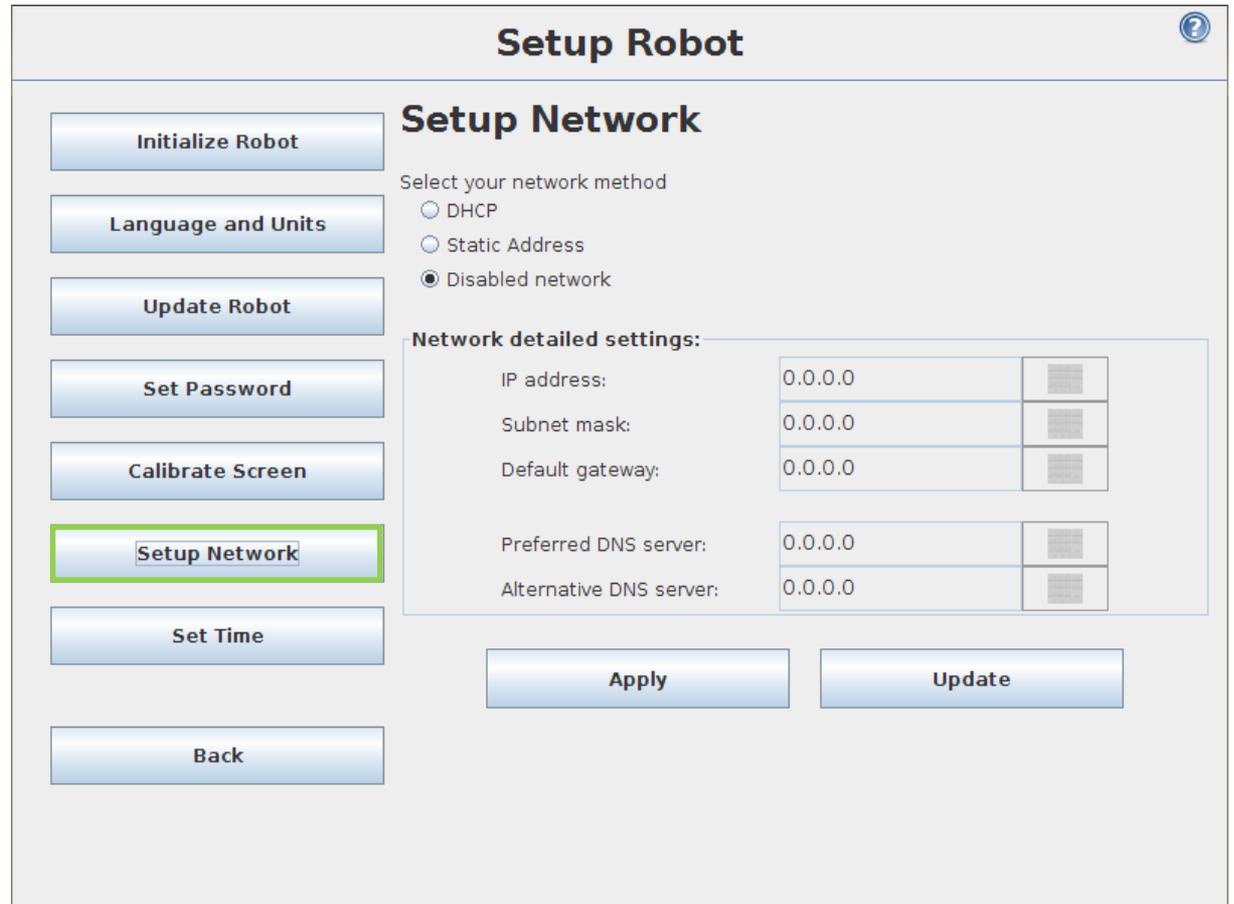
### calibrate screen

- Calibration of touch screen
  - Mark four corners to calibrate



### Setup network

- Network configuration
  - IP-address of robot can be set in this menu



**Setup Robot** ?

**Setup Network**

Select your network method

- DHCP
- Static Address
- Disabled network

**Network detailed settings:**

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

**Apply** **Update**

**Initialize Robot**

**Language and Units**

**Update Robot**

**Set Password**

**Calibrate Screen**

**Setup Network**

**Set Time**

**Back**

- Live demo will be covered later in training session

## Set time

- Time
  - Time format
    - 24 hour
    - 12 hour
- Date
  - Date format

### Setup Robot

**Initialize Robot**

**Language and Units**

**Update Robot**

**Set Password**

**Calibrate Screen**

**Setup Network**

**Set Time**

**Back**

#### Set Time

Time format: Please select the current time:

24 hour     12 hour

+	+	+		
10	:	57	:	42
-	-	-		

---

#### Set Date

Please select today's date:

26 February 2014

Date format:

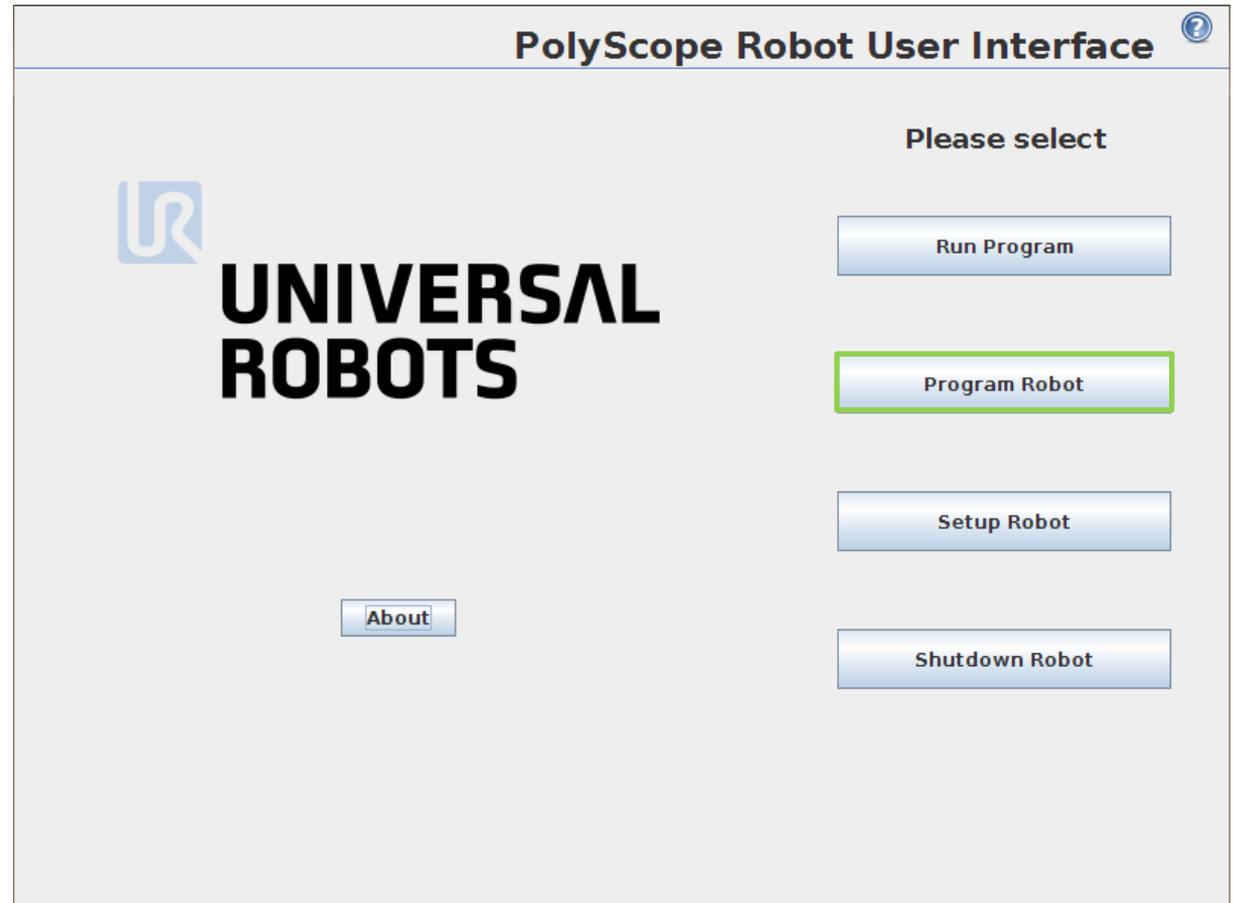
26 February 2014     26-Feb-2014     26/02/14

Restart PolyScope for new settings to take effect

**Restart Now**

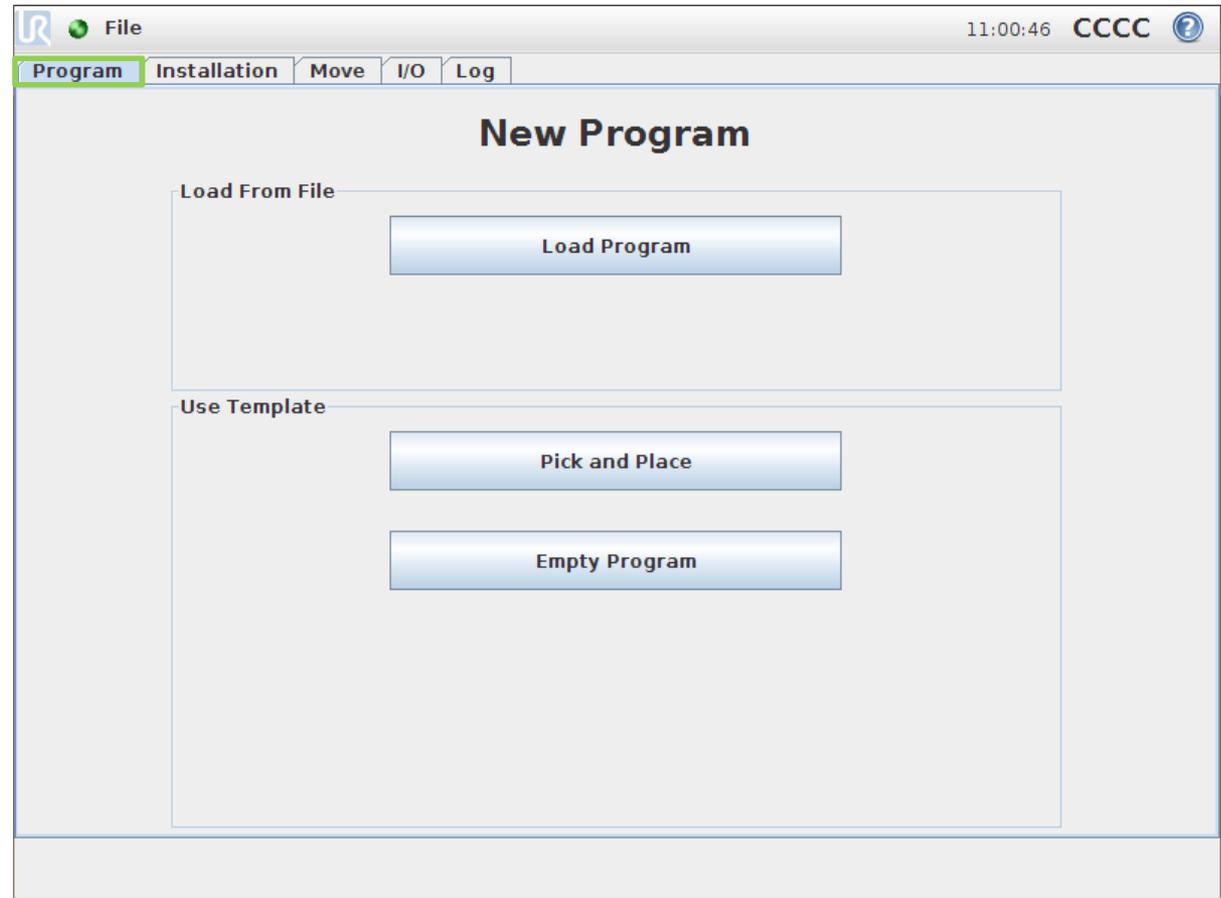
### Program robot

- Program robot
  - Overview of tabs



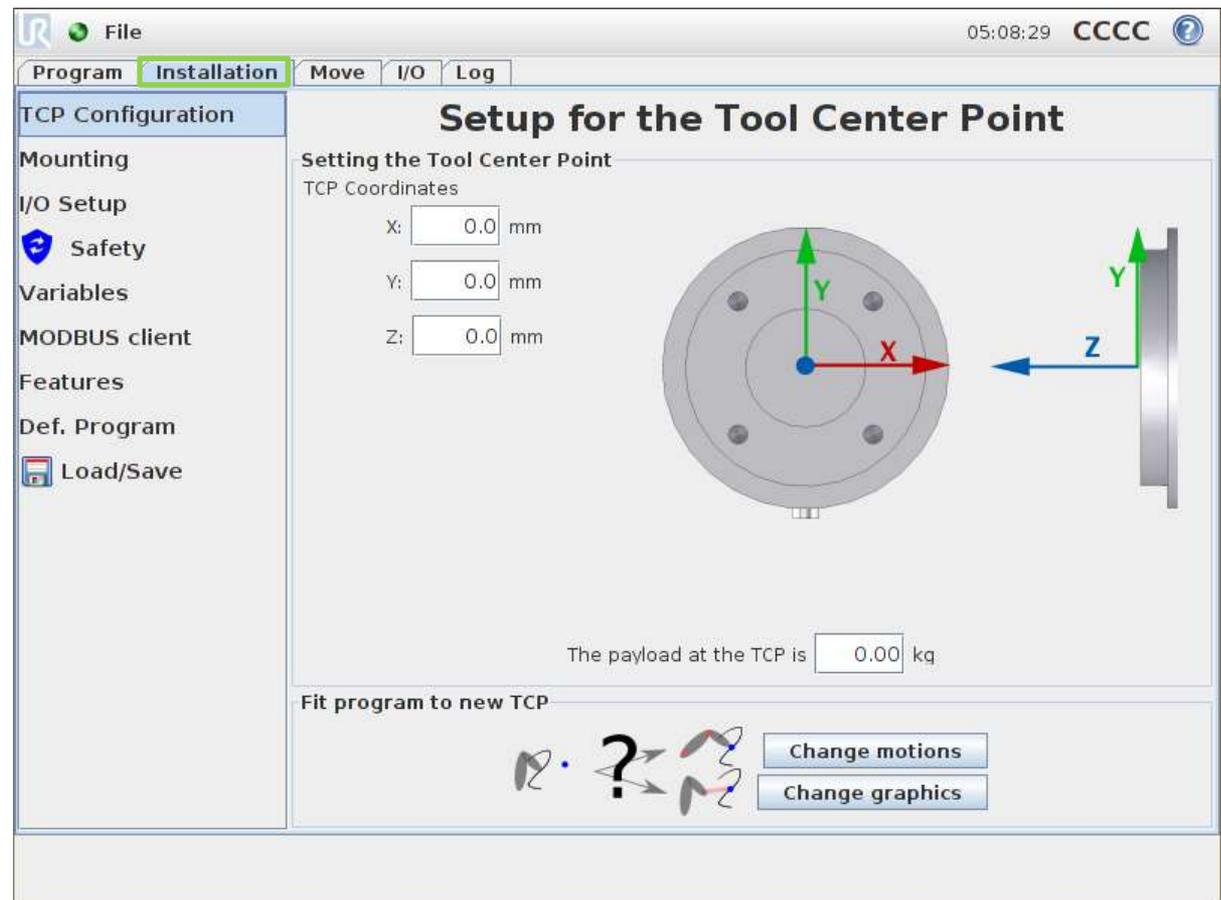
### Program tab

- Program
  - Load existing program
  - Create new program



## Installation tab

- Setup
  - Environment settings
  - Safety settings



## 2 Getting started

### Move tab

- Move
  - Jog robot manually
  - Actual position displayed

The screenshot displays the 'Move' tab in the Universal Robots software interface. The window title is 'File' and the time is '11:01:18'. The 'Move' tab is selected, showing manual jogging controls. The 'Move Tool' section contains three sets of blue arrows for manual movement: a vertical pair, a horizontal pair, and a circular rotation arrow. The 'Robot' section shows a 3D model of the robot arm. The 'Feature' section has a 'View' dropdown. The 'Tool Position' section displays the following values:

Axis	Value	Unit
X	-120.11	mm
Y	-431.76	mm
Z	-253.93	mm
RX	0.0012	
RY	-3.1664	
RZ	-0.0395	

The 'Move Joints' section shows a 'Teach' vertical bar and a 'Home' button. The joint angles are as follows:

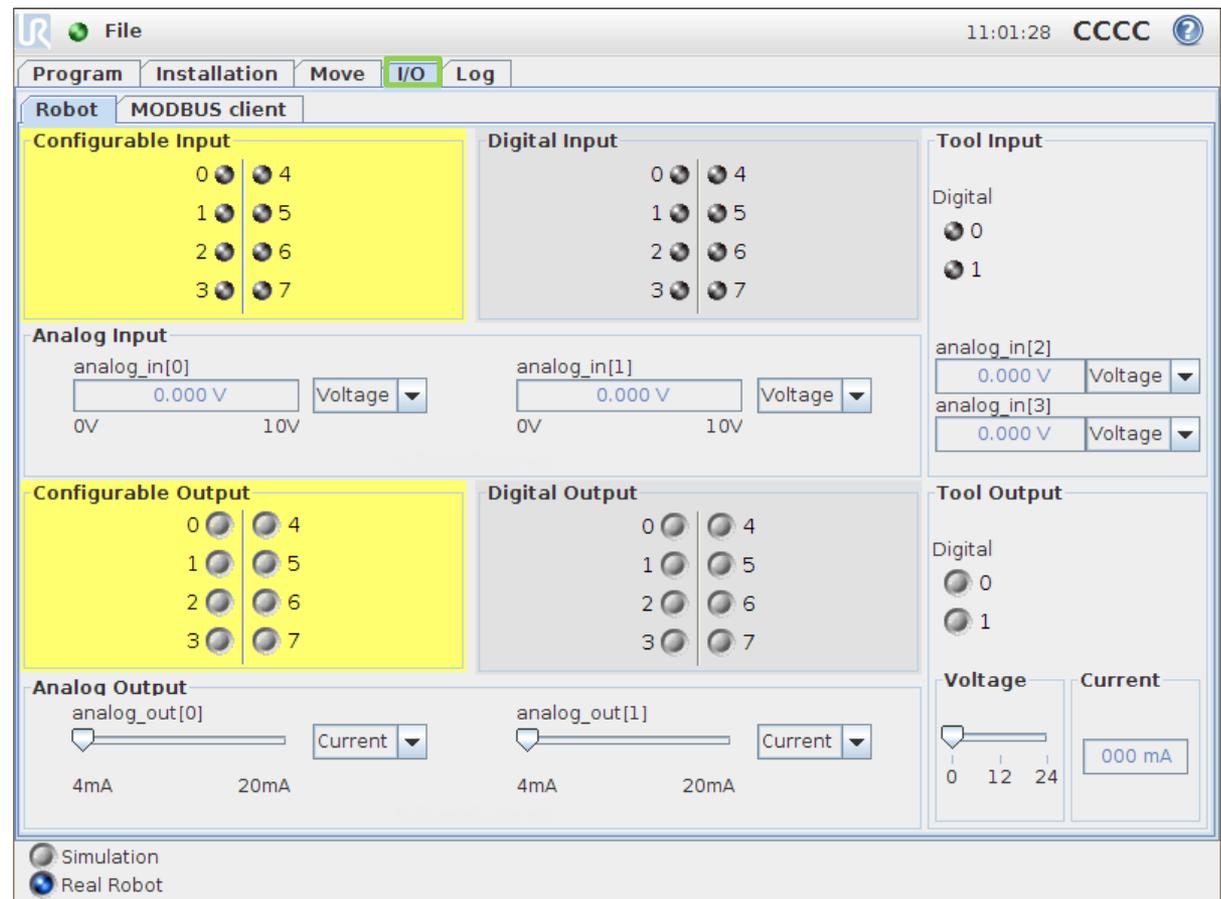
Joint	Angle
Base	-91.71 °
Shoulder	-98.96 °
Elbow	-126.22 °
Wrist 1	-46.29 °
Wrist 2	91.39 °
Wrist 3	-1.78 °

At the bottom, there are radio buttons for 'Simulation' and 'Real Robot', and a 'Speed' slider set to 100%.

## 2 Getting started

### I/O tab

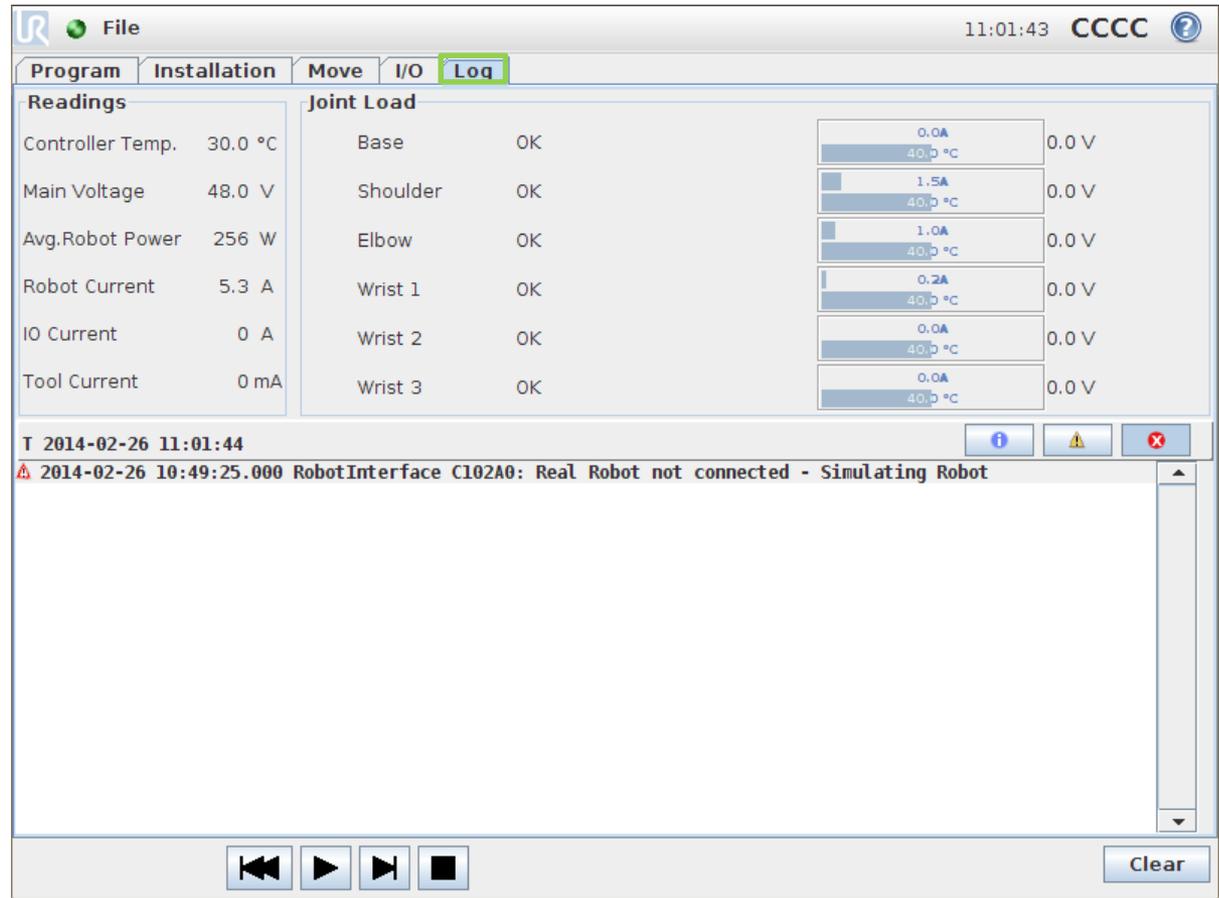
- I/O
  - Monitoring signals
  - Activate signal
  - Setup analog signals



## 2 Getting started

### Log tab

- Status
  - Control box
  - Joints
- Log history
  - Info messages
  - Warnings
  - Errors



File 11:01:43 CCCC

Program Installation Move I/O **Log**

**Readings**

Controller Temp.	30.0 °C
Main Voltage	48.0 V
Avg. Robot Power	256 W
Robot Current	5.3 A
IO Current	0 A
Tool Current	0 mA

**Joint Load**

Base	OK	0.0A	0.0 V
Shoulder	OK	1.5A	0.0 V
Elbow	OK	1.0A	0.0 V
Wrist 1	OK	0.2A	0.0 V
Wrist 2	OK	0.0A	0.0 V
Wrist 3	OK	0.0A	0.0 V

T 2014-02-26 11:01:44

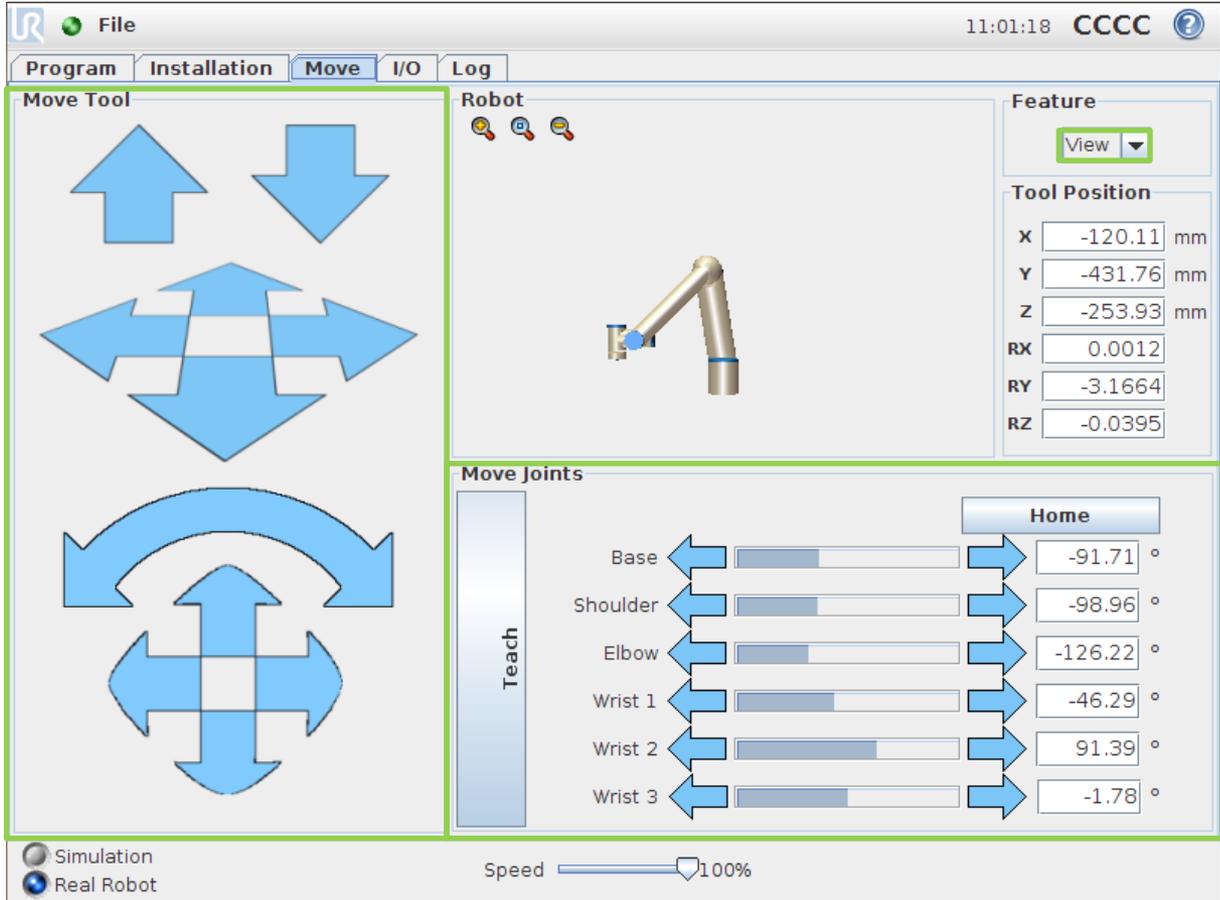
2014-02-26 10:49:25.000 RobotInterface C102A0: Real Robot not connected - Simulating Robot

Clear

## Moving the robot

### Jogging

- One joint individually
  - Tool orientation changes
  - Min/max limits
  - Angular value displayed
- In linear space
  - Tool orientation is fixed when jogging in XYZ
  - Relative to selected Feature
    - View
    - Base
    - Tool
- By teach function
  - Teach button on TP
  - Teach button in UI



The screenshot displays the Universal Robots software interface. The top menu bar includes 'File', 'Program', 'Installation', 'Move', 'I/O', and 'Log'. The 'Move' tab is active. The interface is divided into several sections:

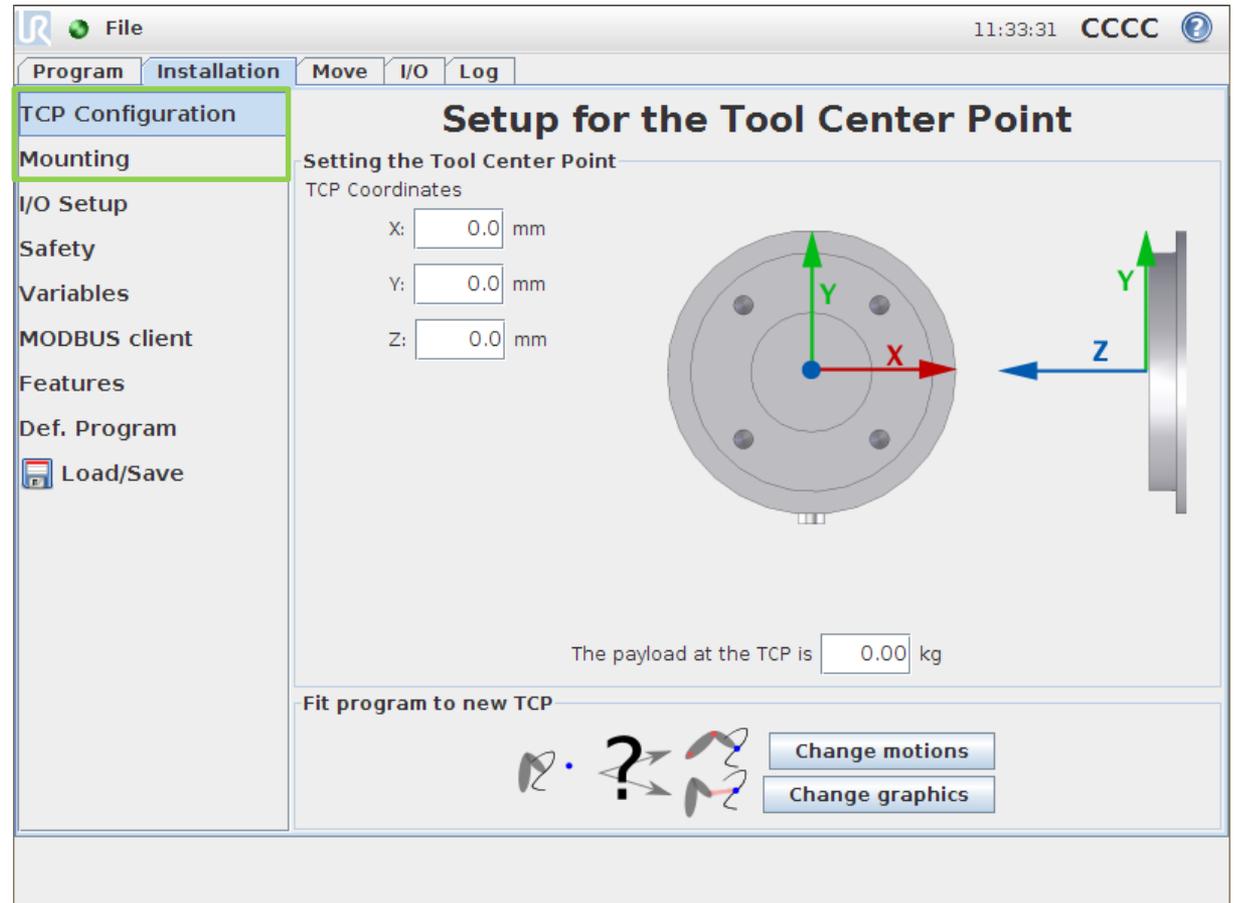
- Move Tool:** Contains three sets of blue arrows: two vertical arrows (up and down), a set of four horizontal arrows (up, down, left, right), and a set of four curved arrows (up, down, left, right) surrounding a central cross.
- Robot:** Shows a 3D model of a robotic arm.
- Feature:** Includes a 'View' dropdown menu.
- Tool Position:** Displays numerical values for X, Y, Z, RX, RY, and RZ in millimeters and degrees.
- Move Joints:** Shows a 'Teach' button and a 'Home' button. Below them are six rows of joint controls, each with a slider and a numerical value in degrees: Base (-91.71), Shoulder (-98.96), Elbow (-126.22), Wrist 1 (-46.29), Wrist 2 (91.39), and Wrist 3 (-1.78).
- Simulation/Real Robot:** A toggle at the bottom left, currently set to 'Real Robot'.
- Speed:** A slider at the bottom right, currently set to 100%.

Joint	Value	Unit
Base	-91.71	°
Shoulder	-98.96	°
Elbow	-126.22	°
Wrist 1	-46.29	°
Wrist 2	91.39	°
Wrist 3	-1.78	°

Axis	Value	Unit
X	-120.11	mm
Y	-431.76	mm
Z	-253.93	mm
RX	0.0012	
RY	-3.1664	
RZ	-0.0395	

## Initial setup

- Initial setup
  - TCP configuration
  - Mounting



File 11:33:31 CCCC ?

Program Installation Move I/O Log

**Setup for the Tool Center Point**

Setting the Tool Center Point

TCP Coordinates

X:  mm

Y:  mm

Z:  mm

The payload at the TCP is  kg

Fit program to new TCP

Change motions

Change graphics

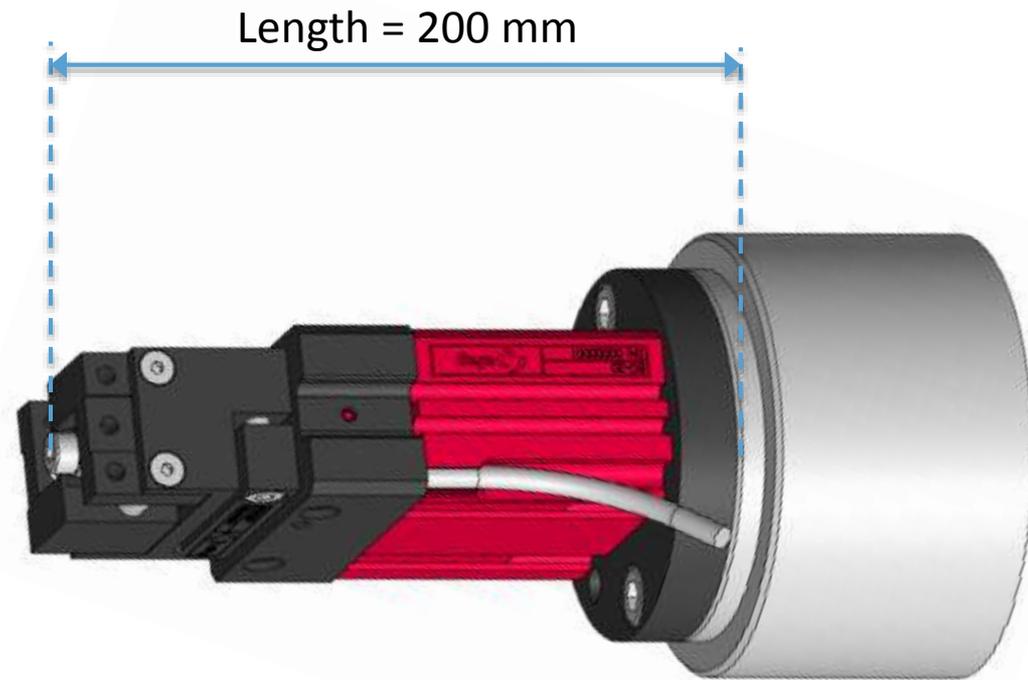
### TCP configuration

- Tool Center Point (TCP)
  - Linear distance from center of tool flange to tip of tool
  - Set XYZ coordinates according to illustration
- Payload
  - Weight of tool

The screenshot shows the 'Setup for the Tool Center Point' window in the Universal Robots software. The window has a menu bar with 'File', 'Program', 'Installation', 'Move', 'I/O', and 'Log'. The 'Installation' tab is active, and the 'TCP Configuration' sub-tab is selected. The main area is titled 'Setting the Tool Center Point' and contains 'TCP Coordinates' with input fields for X, Y, and Z, all set to 0.0 mm. To the right of the input fields is a diagram of a tool flange with a coordinate system: X (red arrow pointing right), Y (green arrow pointing up), and Z (blue arrow pointing left). Below the diagram, there is a field for 'The payload at the TCP is' set to 0.00 kg. At the bottom, there are icons for 'Fit program to new TCP' and two buttons: 'Change motions' and 'Change graphics'.

### Payload calculation

- Example:
  - how to calculate max. allowed payload

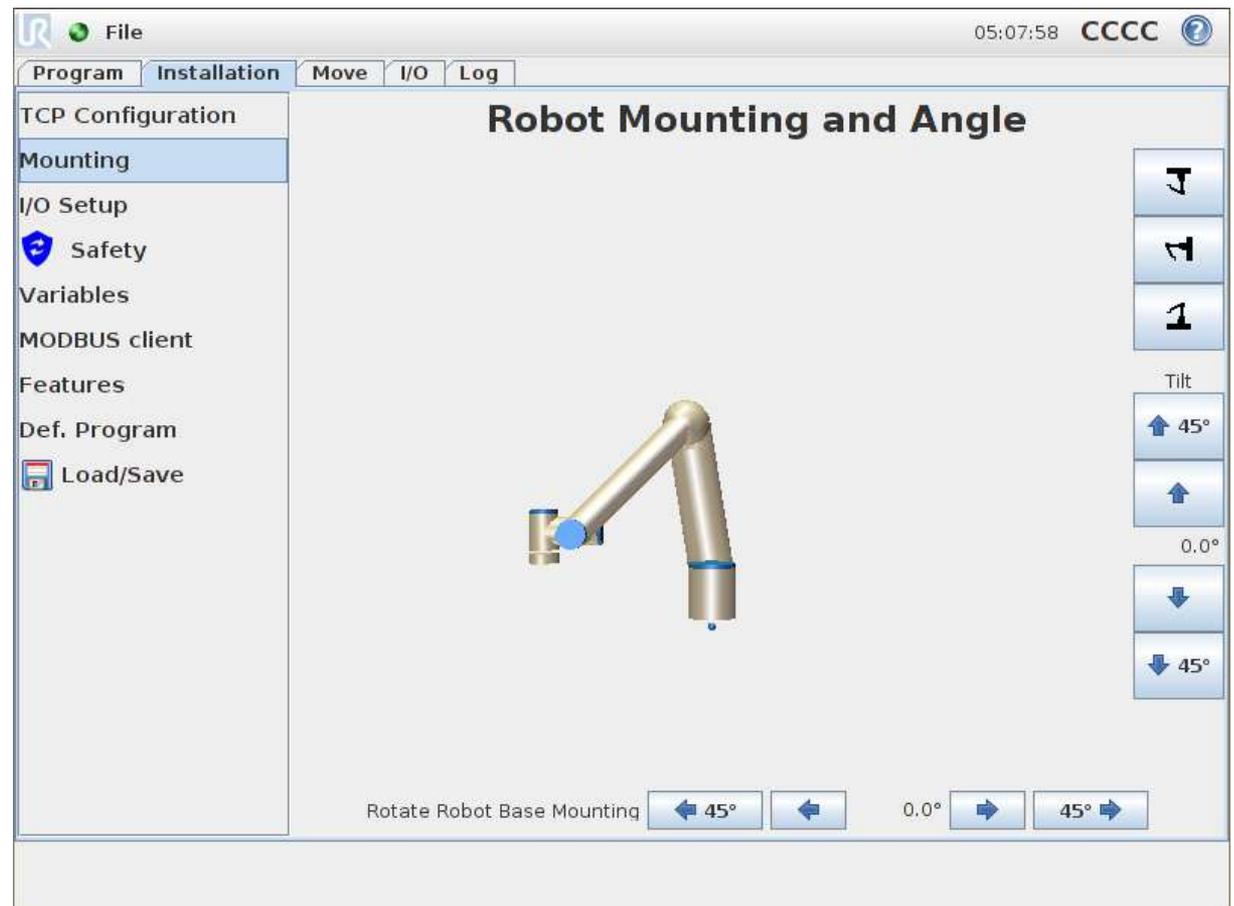


$$\text{Payload} = \frac{4.5}{(0.9 + \text{length})} = \frac{4.5}{(0.9 + 0.2)} = 4.09 \text{ kg}$$

## 2 Getting started

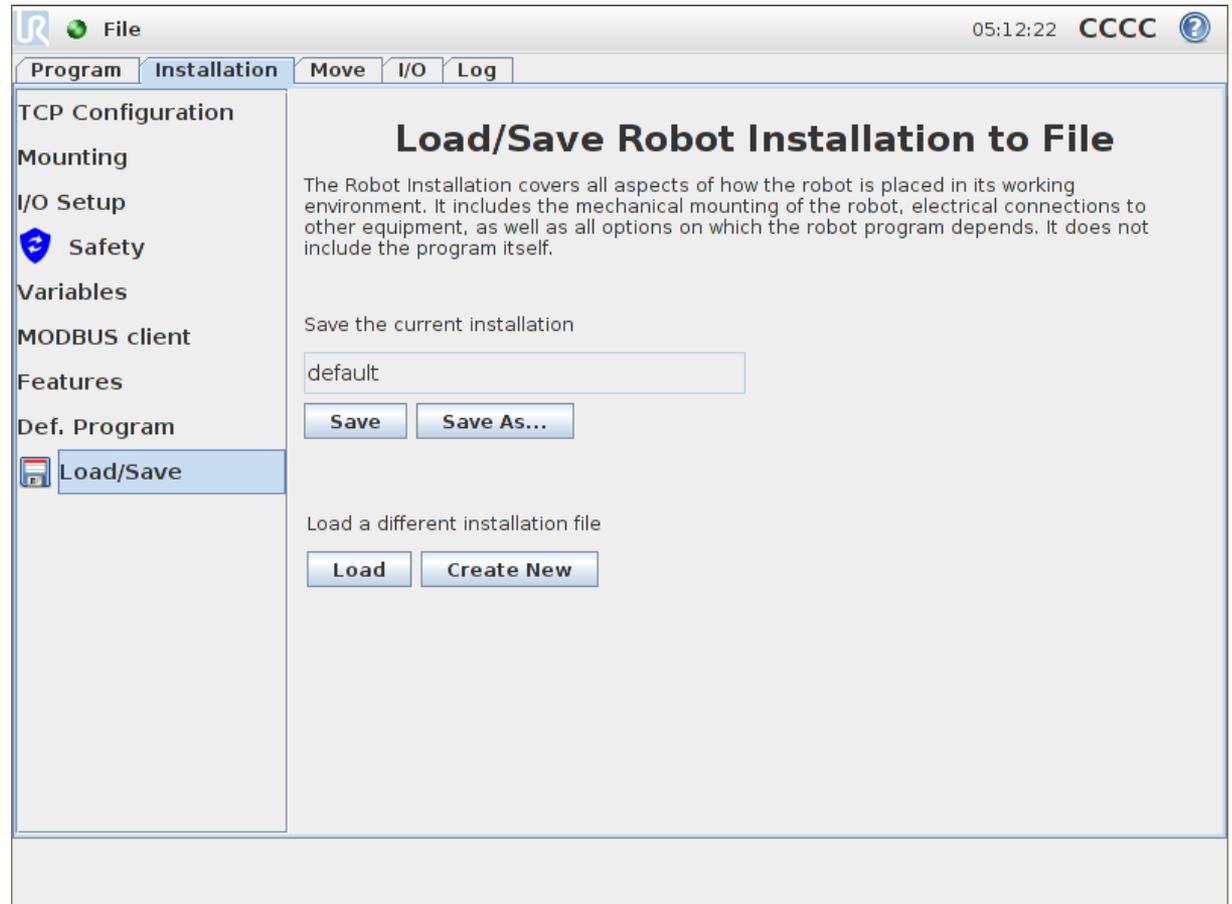
### Mounting

- Setup how robot base is mounted



### Load/save

- Save Installation
  - All settings in Installation is saved in file
- Load Installation
  - Loads a saved Installation file
- Default name
  - default.installation
  - Loaded when booting robot





Hardware

Advanced commands 3

Getting started

Wizards

**3** Basic commands 1

Modbus TCP

Basic commands 2

Service

Advanced commands 1

Safety standards

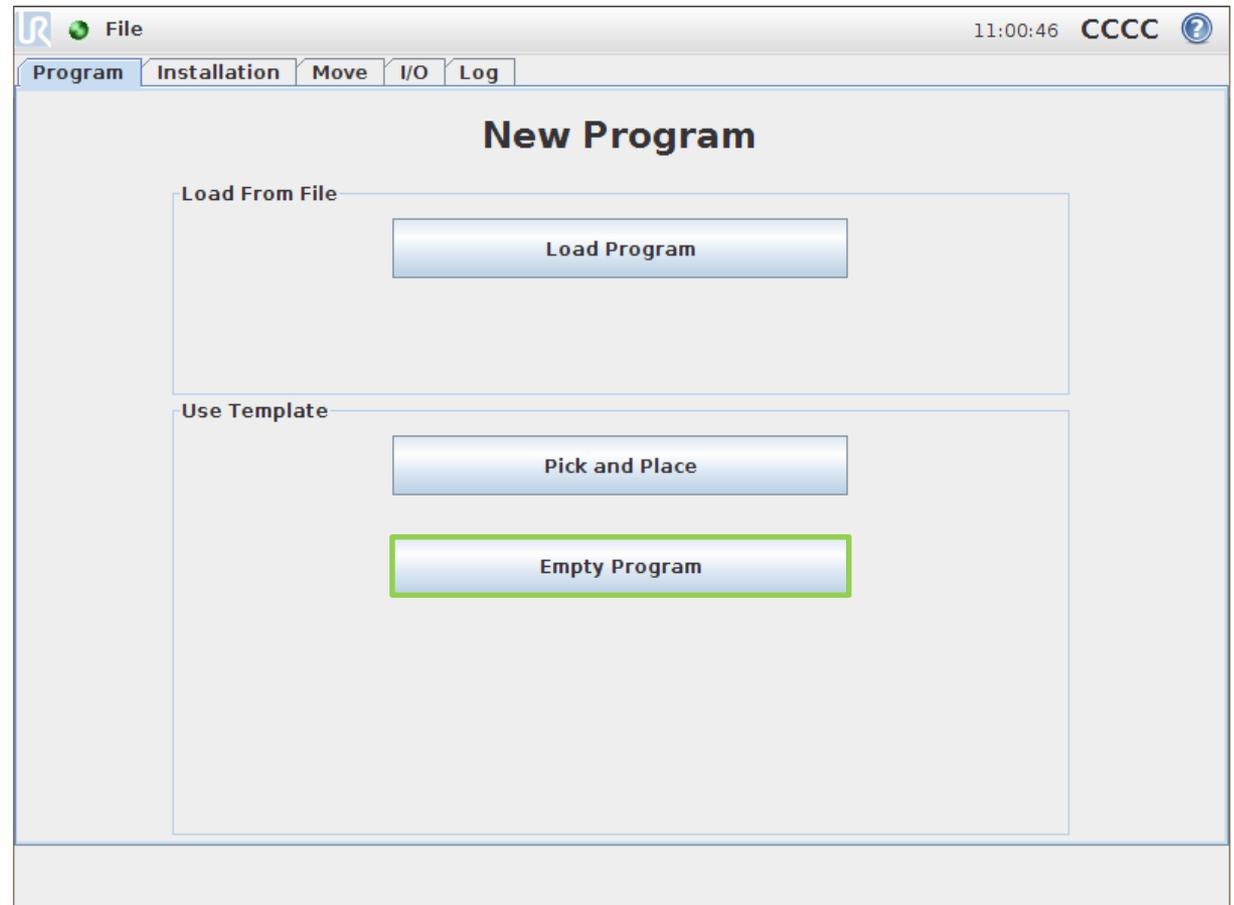
Advanced commands 2

Adjustable safety

# 3 Basic commands 1

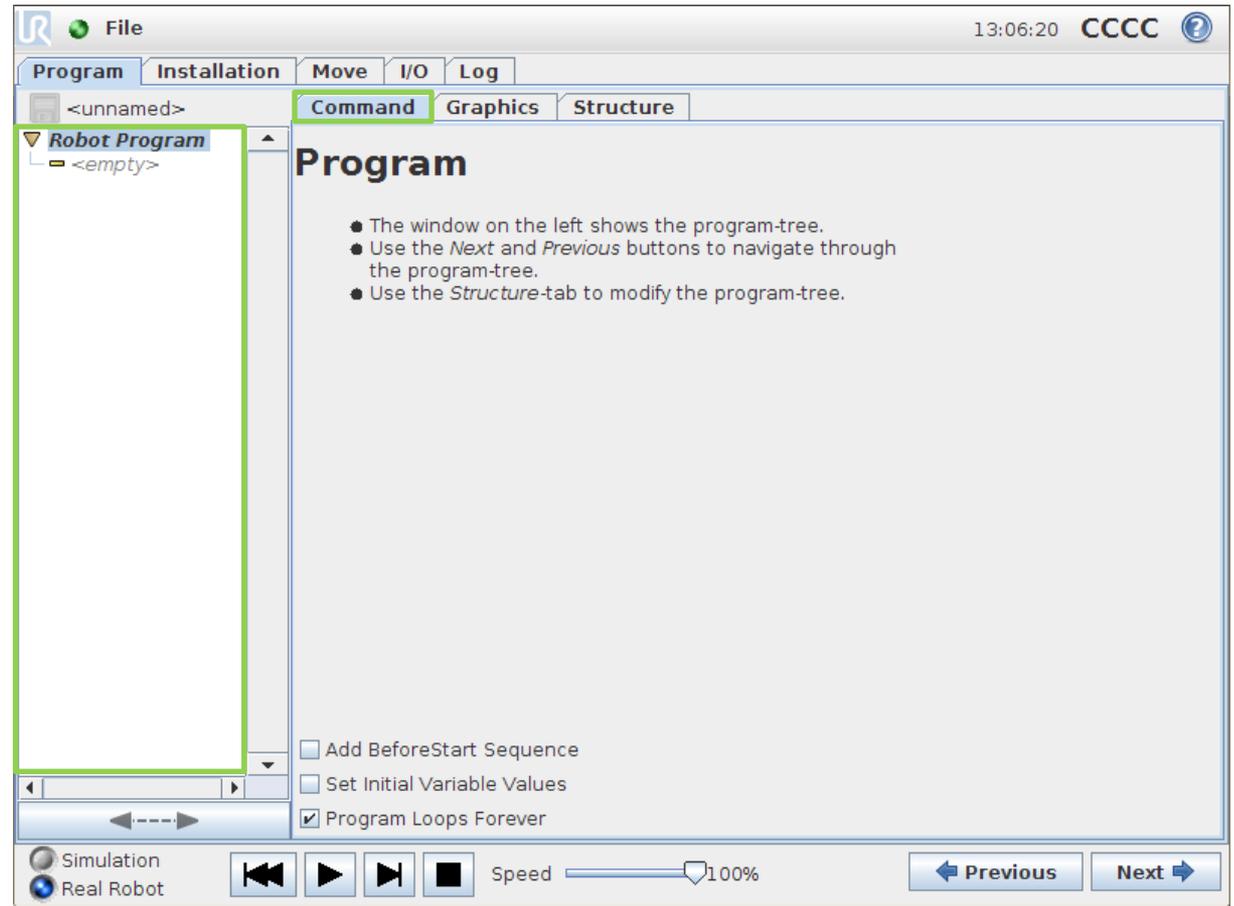
## Create new program

- Use template
  - Empty Program



## Command tab

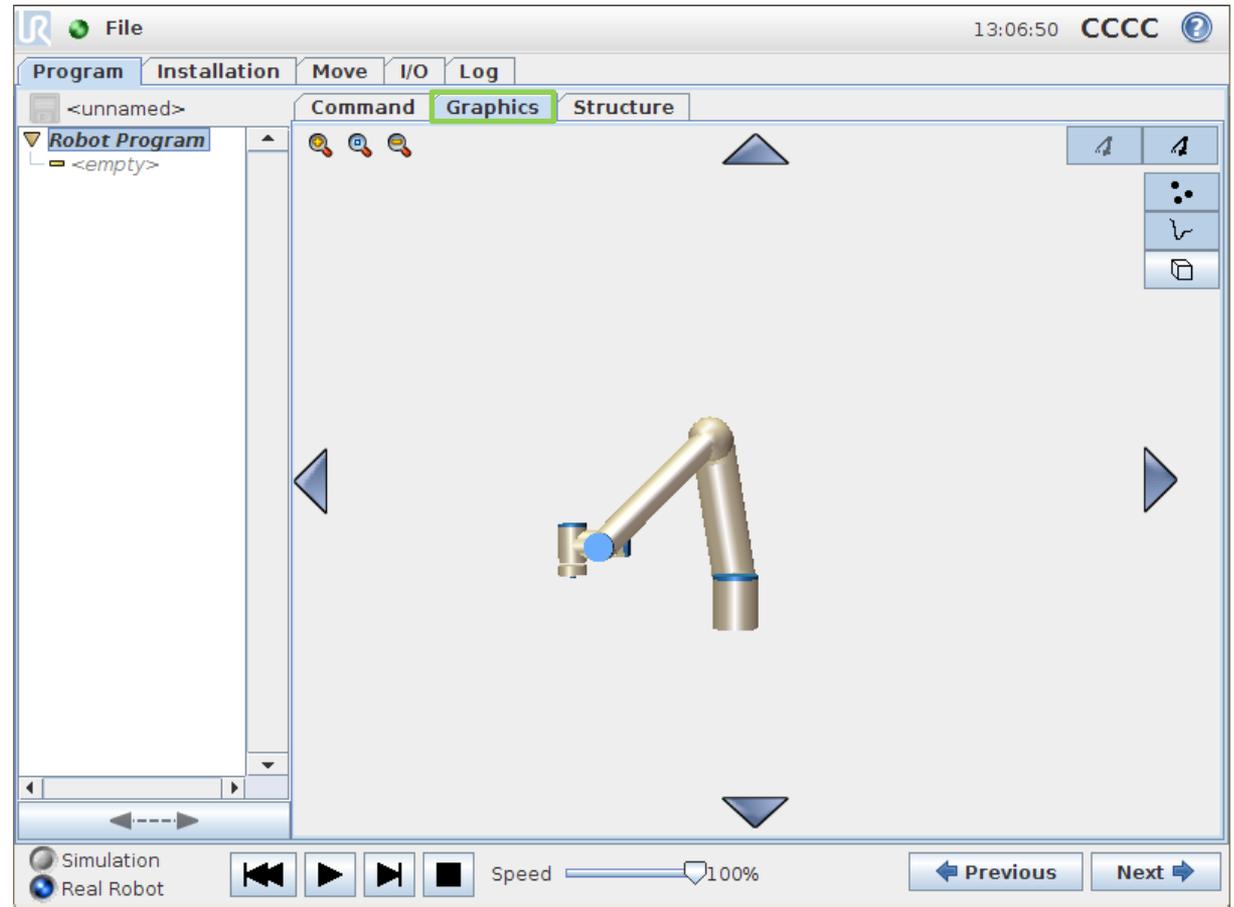
- Program tree
  - Structure of program
  - Contains all commands
  - Executed sequentially
- Command tab
  - Edit of selected command
  - Set general settings for program



# 3 Basic commands 1

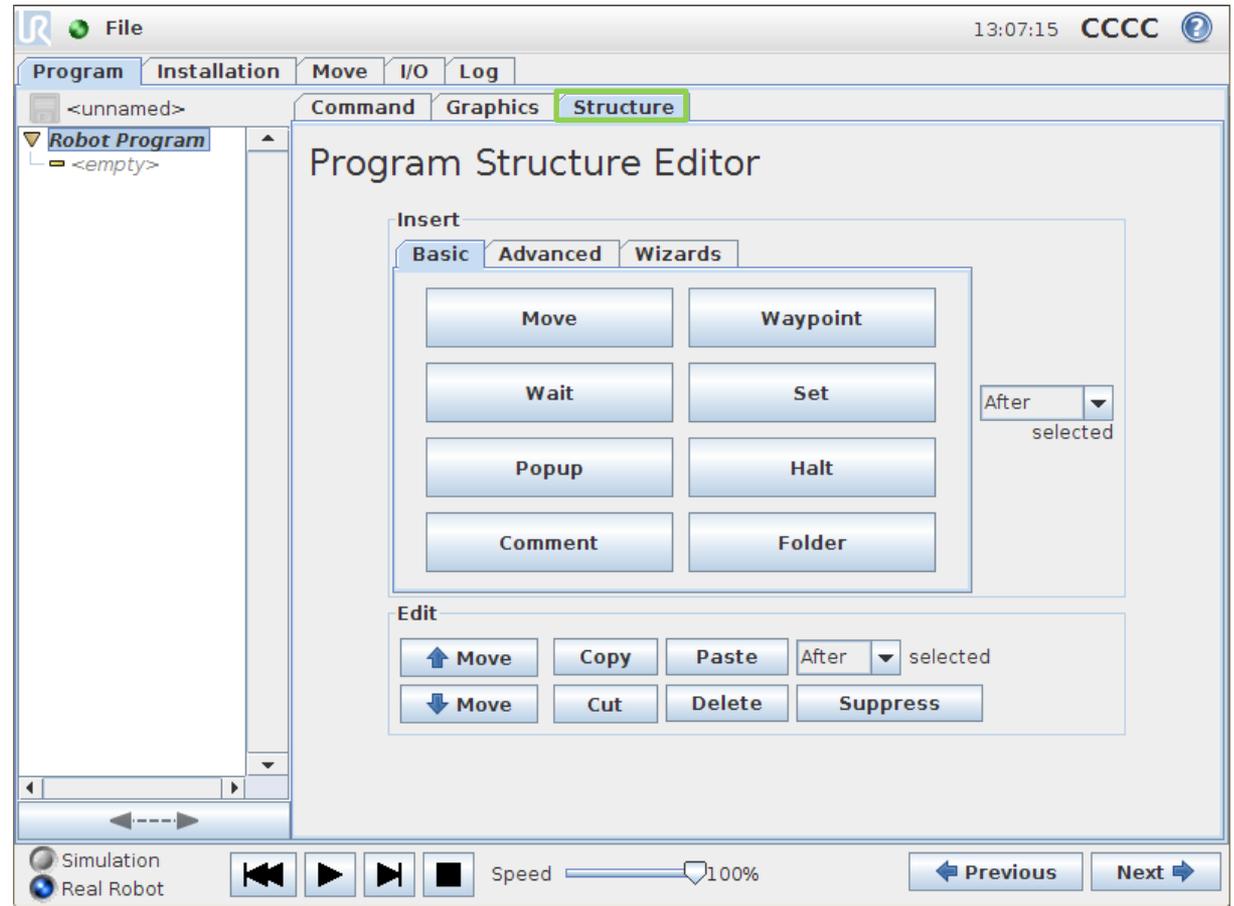
## Graphics tab

- Graphics tab
  - Actual position
  - Taught positions
  - Trajectories



## Structure tab

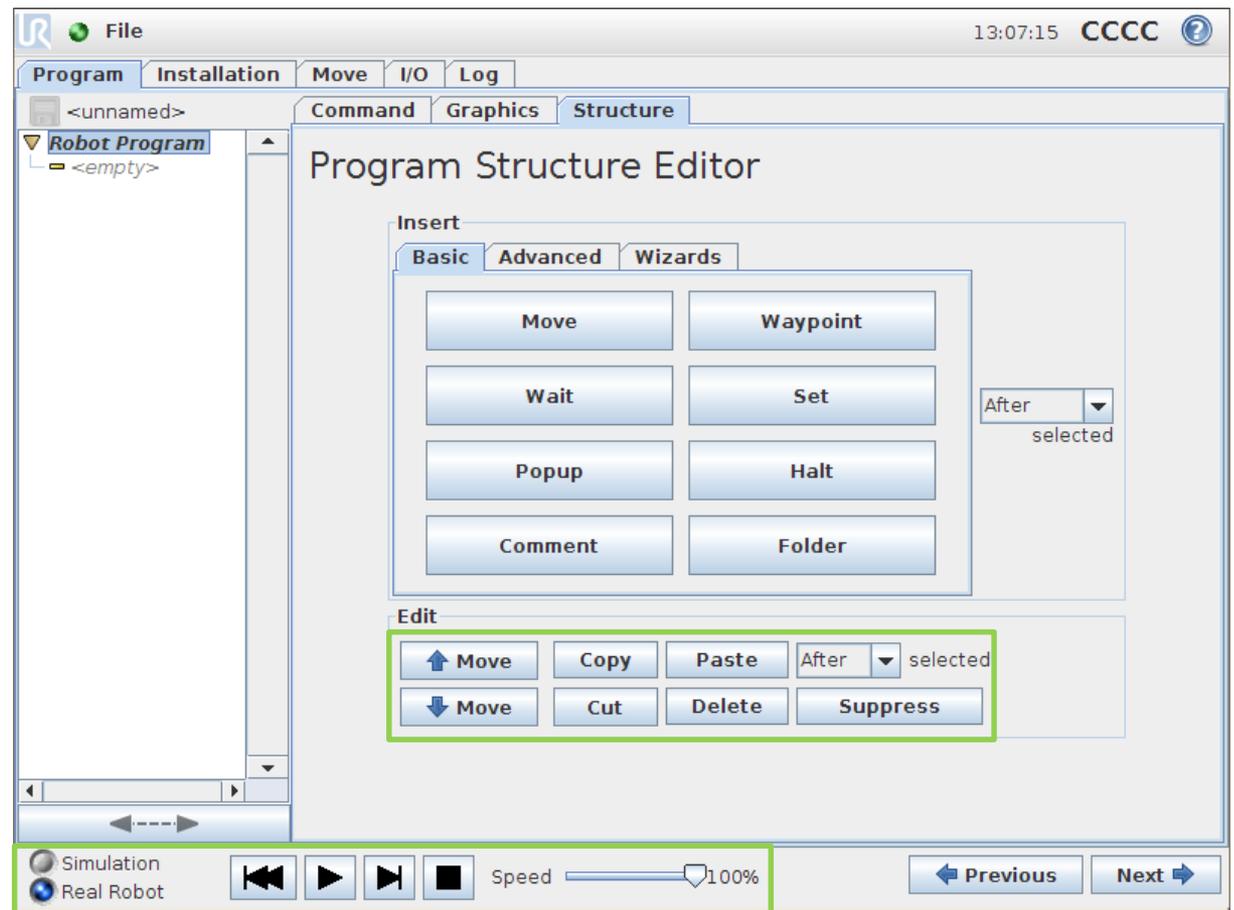
- Structure tab
  - Contains all commands
  - Split into
    - Basic
    - Advanced
    - Wizards



# 3 Basic commands 1

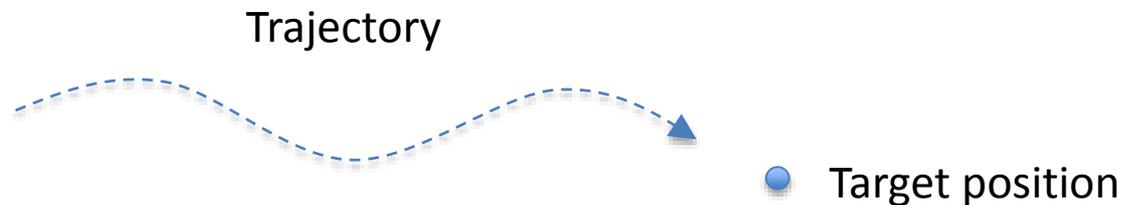
## Editing area / Dashboard

- Editing area
  - Edit and organize program tree
  - Editing tools
  - Suppress function
- Dashboard
  - Control program execution
  - Simulation / Real Robot



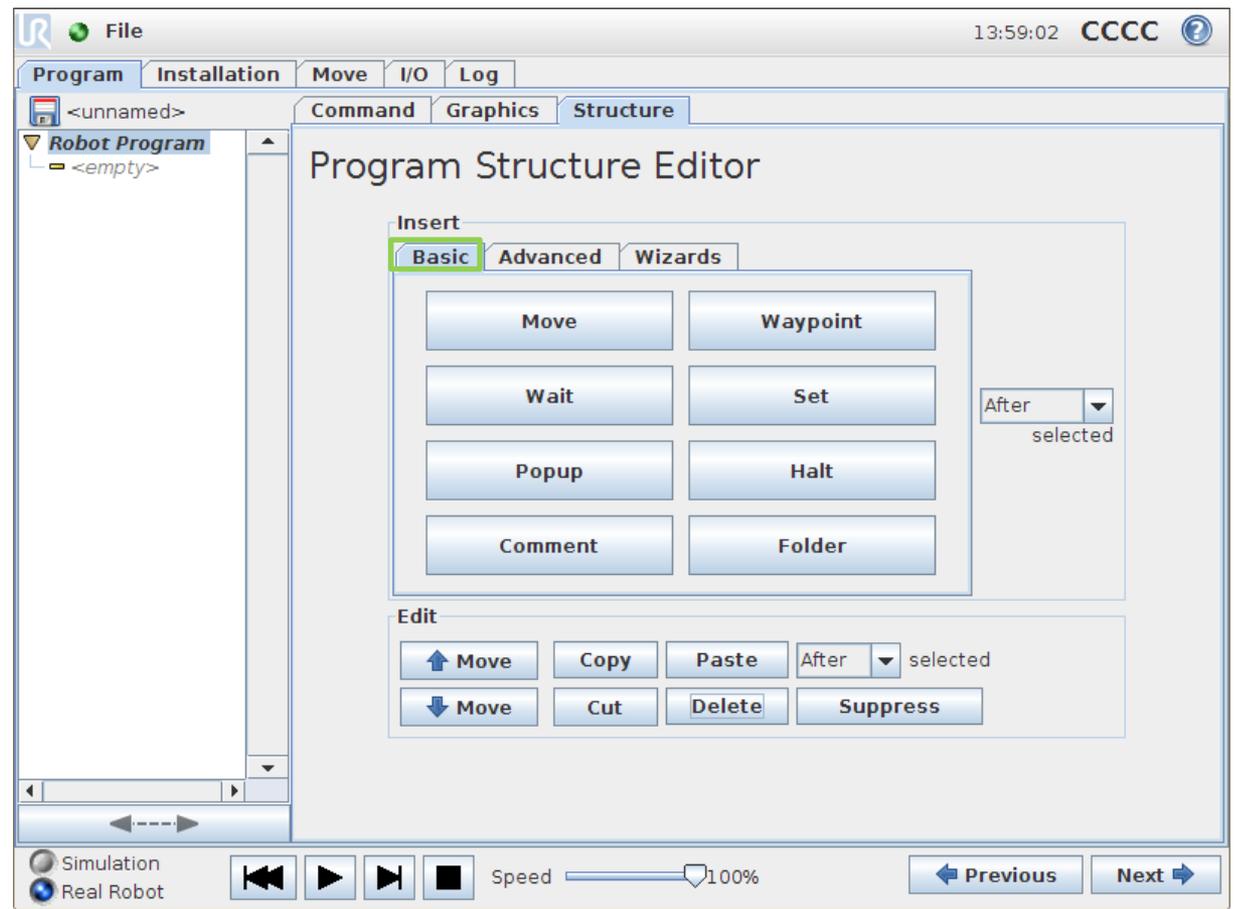
## Insert motion

- **Waypoint** specifies the target position
- **Move** specifies the trajectory robot will follow on it's way to target position



## Basic commands

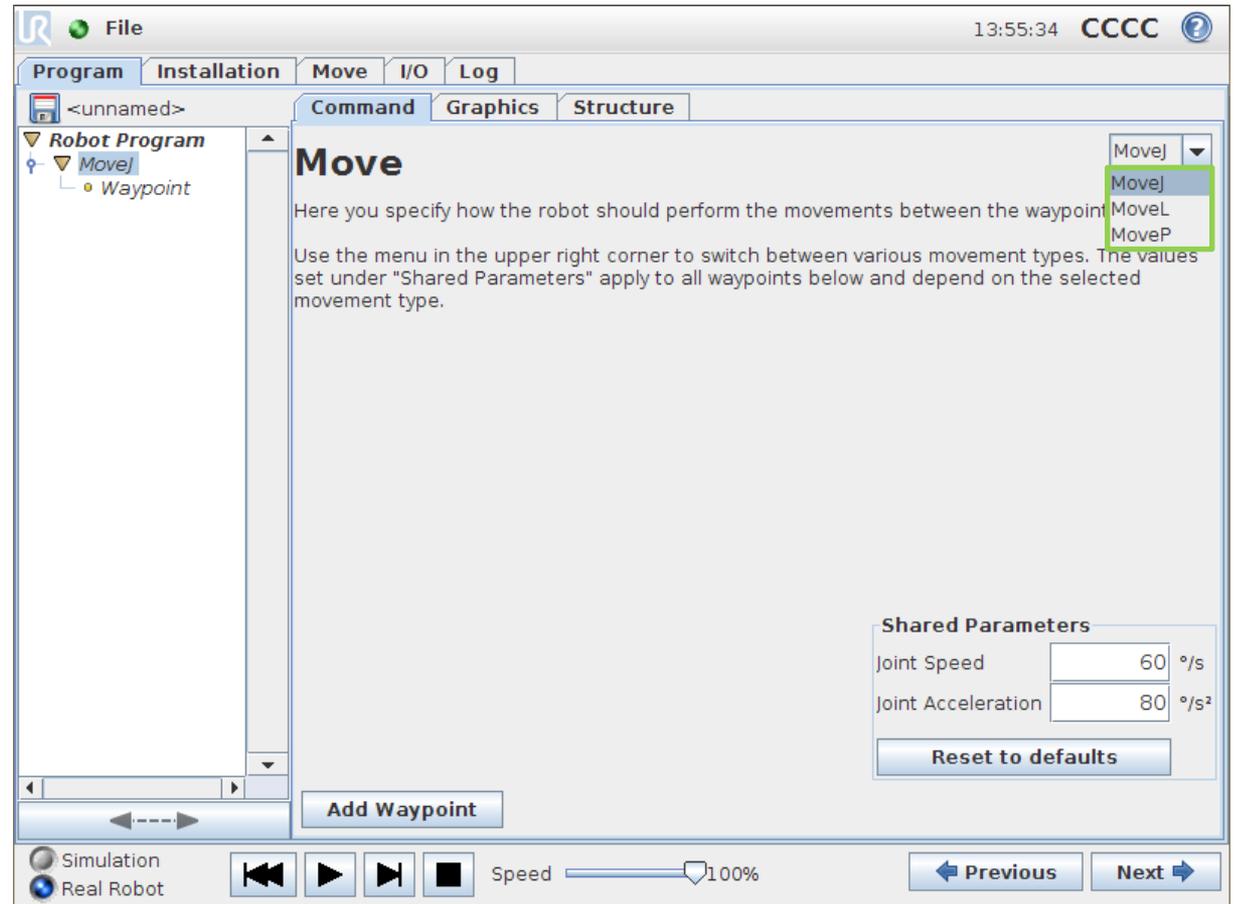
- Basic commands
  - Perform simple operations



- Insert Move command

## Move command

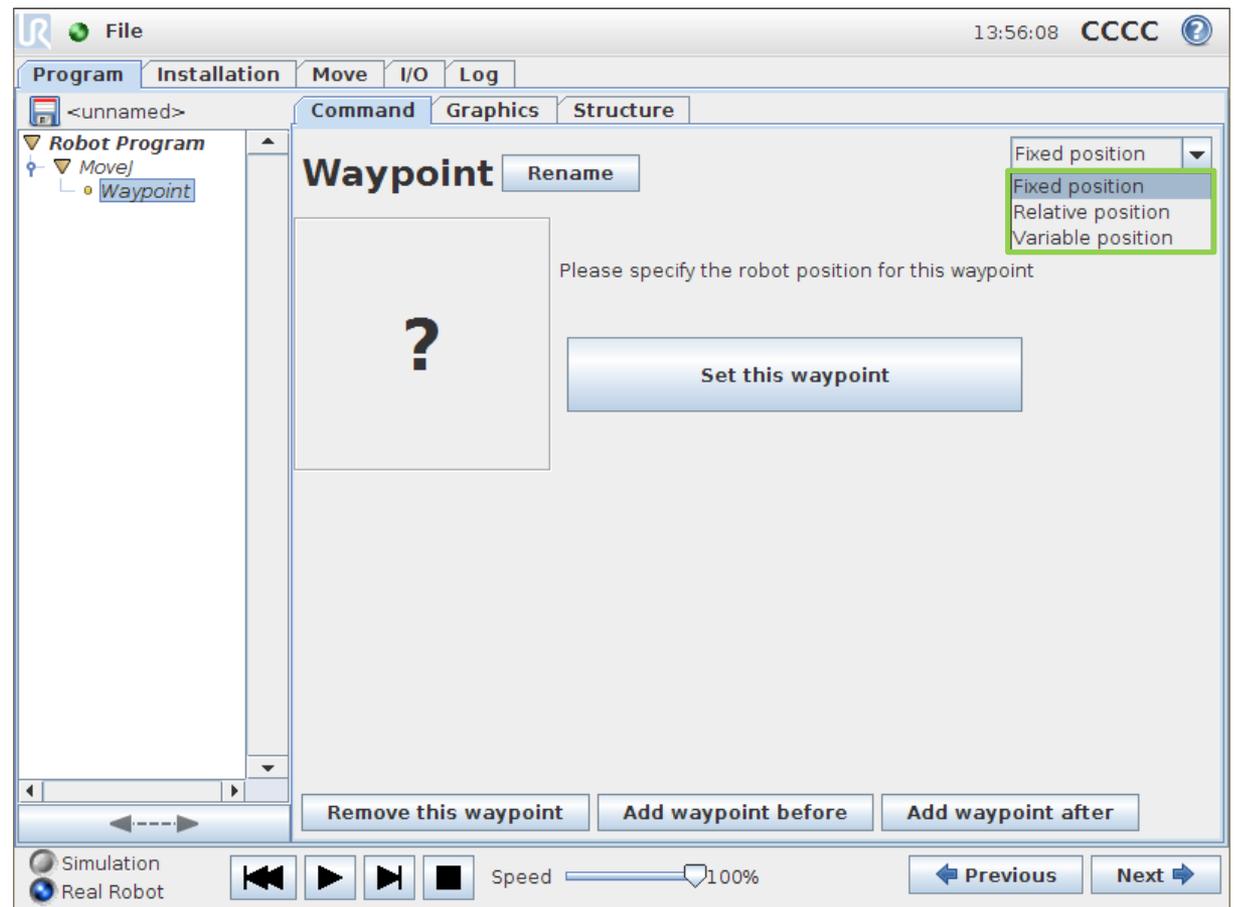
- Movement types
  - MoveJ (*default*)
  - MoveL
  - MoveP
    - MoveC
- Shared parameters
  - Joint speed
    - Default: 60 °/s
  - Joint acceleration
    - Default: 80 °/s<sup>2</sup>



- Waypoint automatically added when inserting Move command

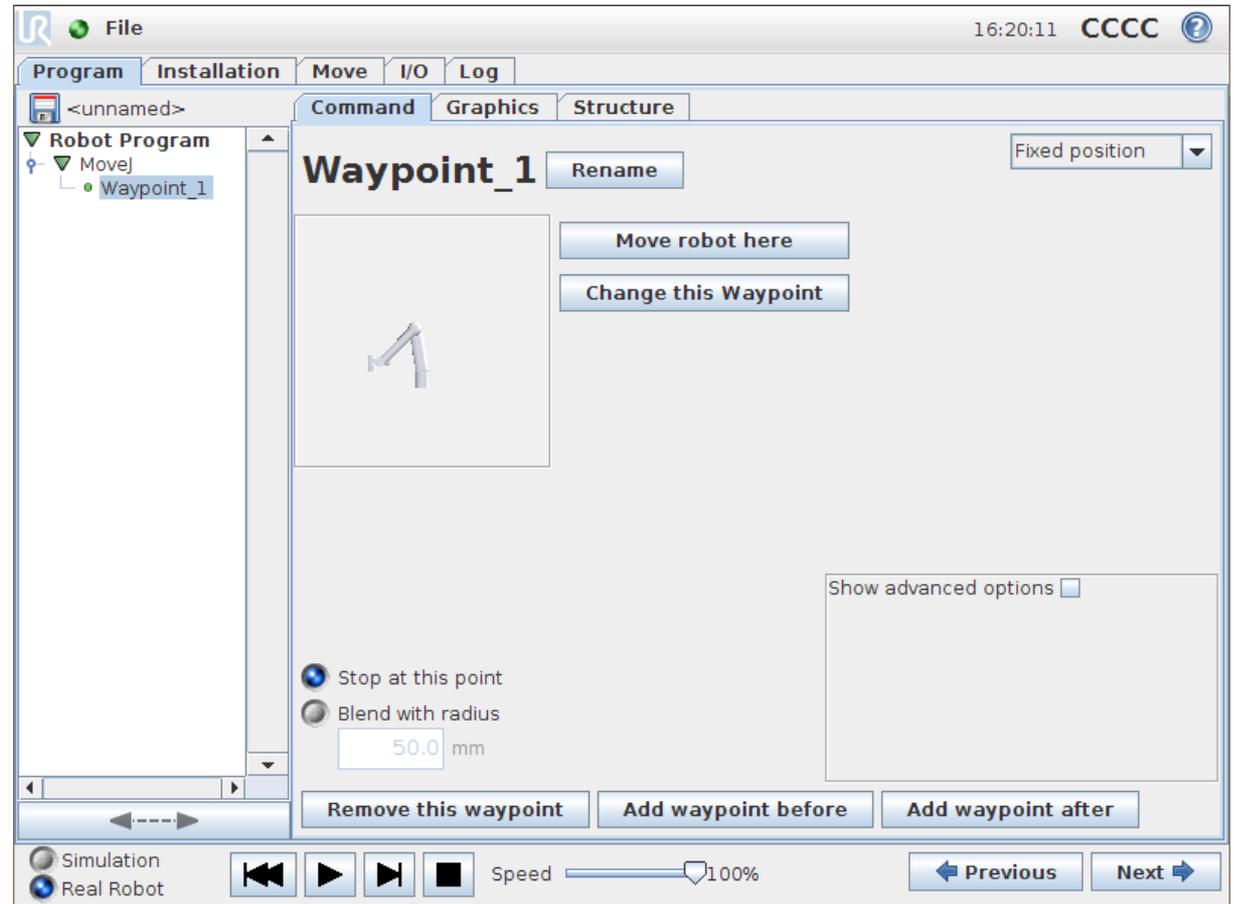
## Waypoint

- Waypoint types
  - Fixed (*default*)
  - Relative
  - Variable
- Teach Waypoint
  - Press *Set this Waypoint*
  - Teach position and press OK for saving



## Waypoint

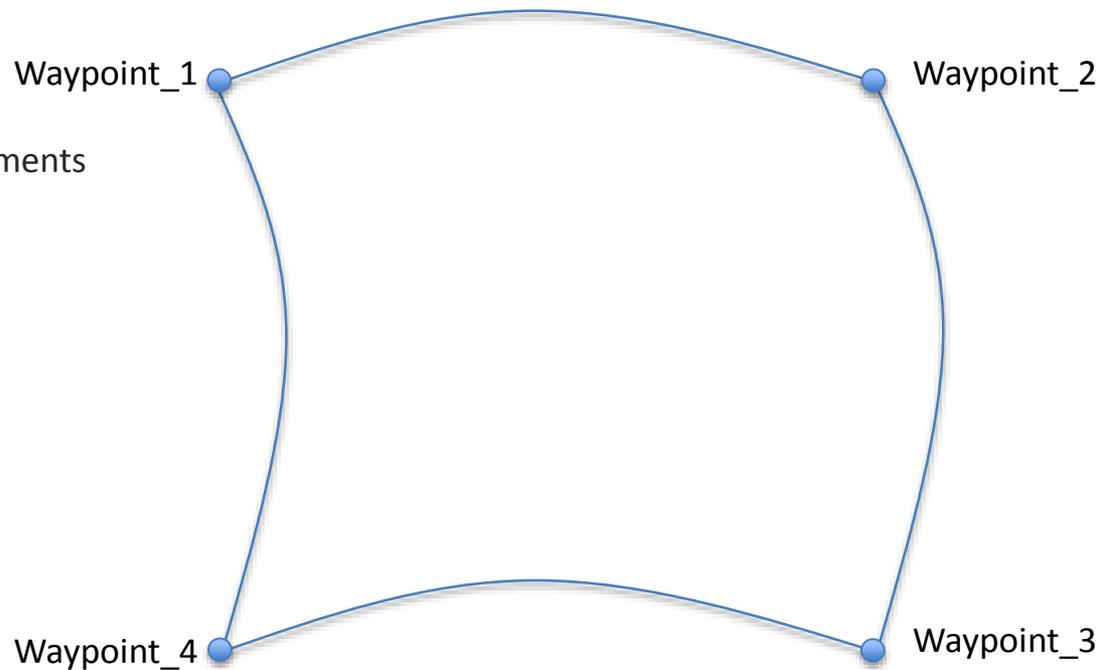
- Options
  - Rename Waypoint
    - Improves readability
  - Move robot here
    - Moves to position
  - Change this Waypoint
    - Modify position



## MoveJ

- Joint movement
  - No interpolation
  - Fastest move type
  - Useful in "free" space movements

```
Robot Program
MoveJ
Waypoint_1
Waypoint_2
Waypoint_3
Waypoint_4
```



- Save sample program as movej.urp

## MoveL

- Linear movement
  - Interpolation on
  - Linear trajectory for TCP

```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2
  Waypoint_3
  Waypoint_4
```

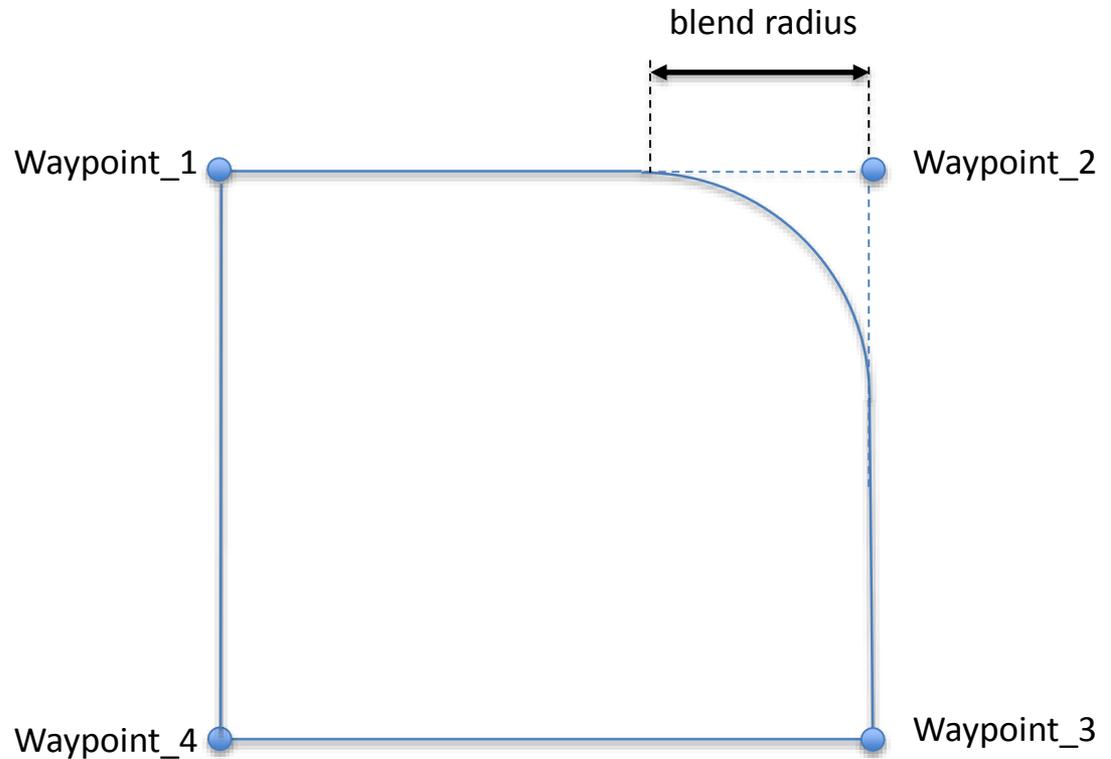


- Save sample program as movel.urp

## MoveL with blend

- Blend radius
  - Continuous movement
  - No stop in Waypoint

```
Robot Program
MoveL
Waypoint_1
Waypoint_2 (r=25)
Waypoint_3
Waypoint_4
```

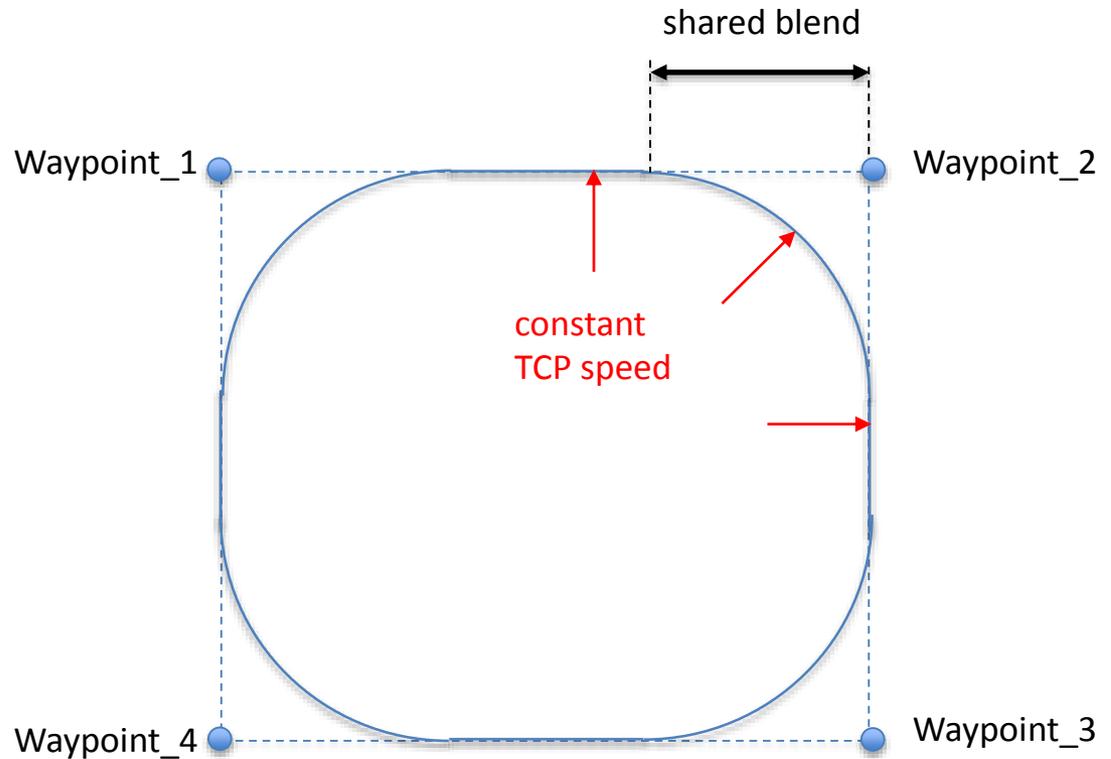


- Save sample program as move\_with\_blend.urp

## MoveP

- Process Movement
  - Process applications
  - Linear movement
  - Constant TCP speed
  - Shared blend radius

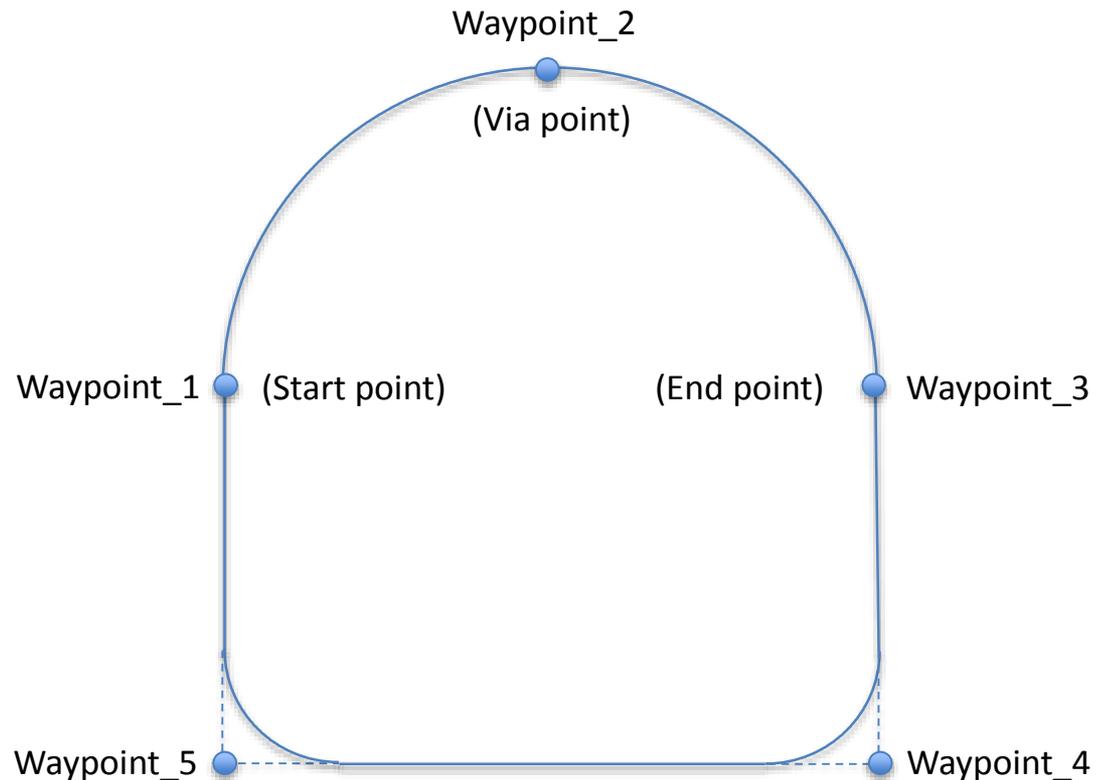
```
Robot Program
MoveP
Waypoint_1
Waypoint_2
Waypoint_3
Waypoint_4
```



- Save sample program as movep.urp

## MoveC

- Circular movement
  - Circular trajectory for TCP
  - Three Waypoints required
    - Start point
    - Via point
    - End point
  - Radius auto calculated

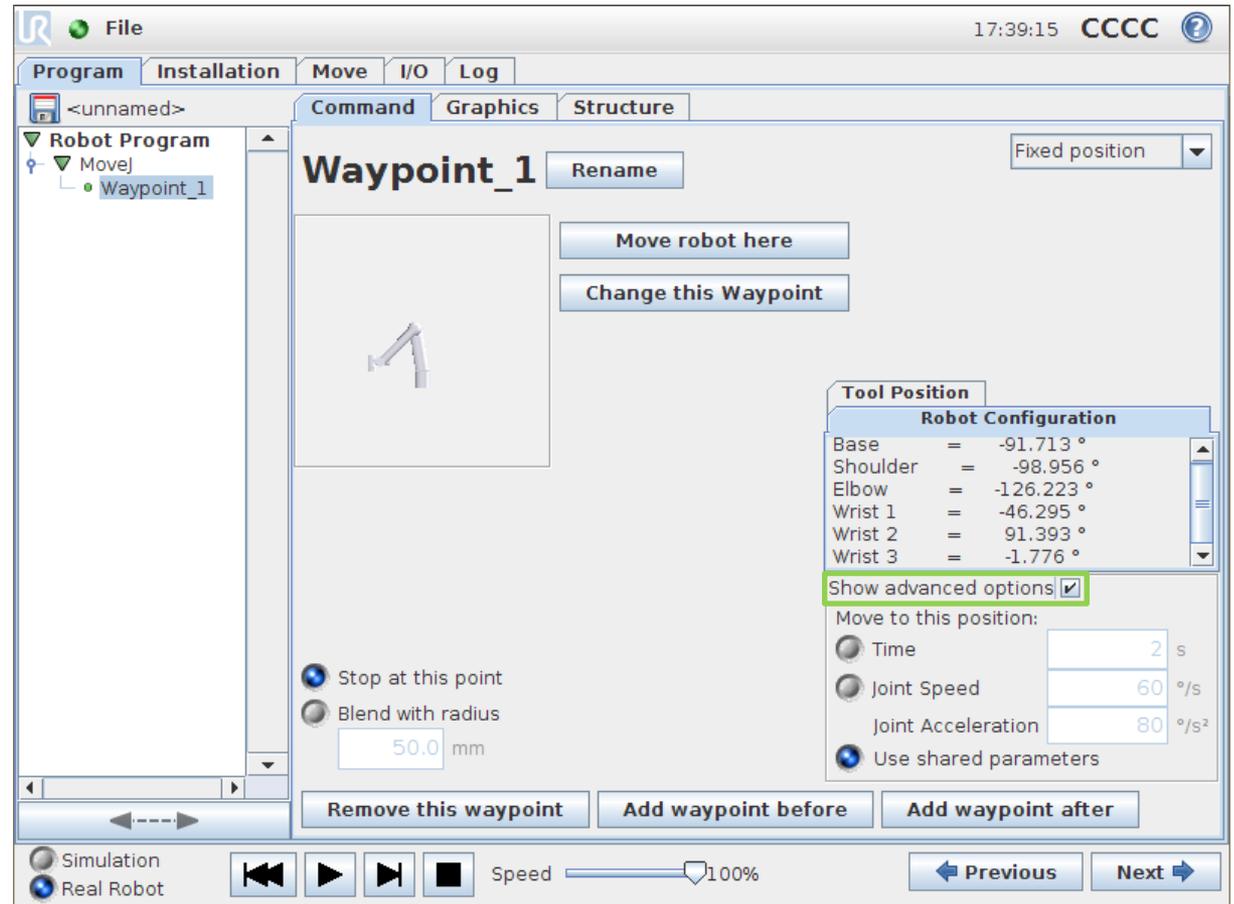


```
Robot Program
MoveP
  Waypoint_1
CircleMove
  Waypoint_2
  Waypoint_3
Waypoint_4
Waypoint_5
```

- Save sample program as movec.urp

## Advanced options

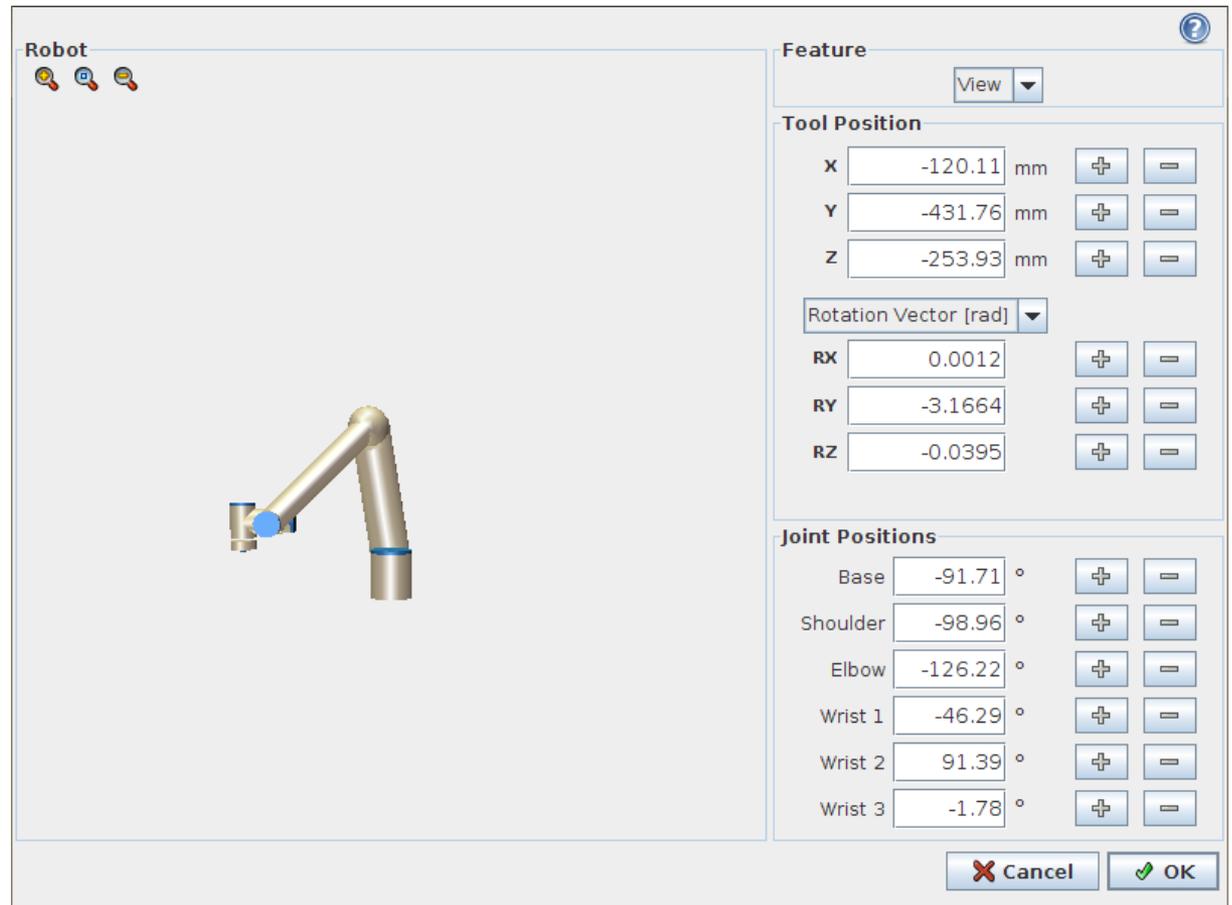
- Advanced options
  - Set individual
    - Speed
    - acceleration
  - Waypoint position
    - Joint angles
    - Tool position



# 3 Basic commands 1

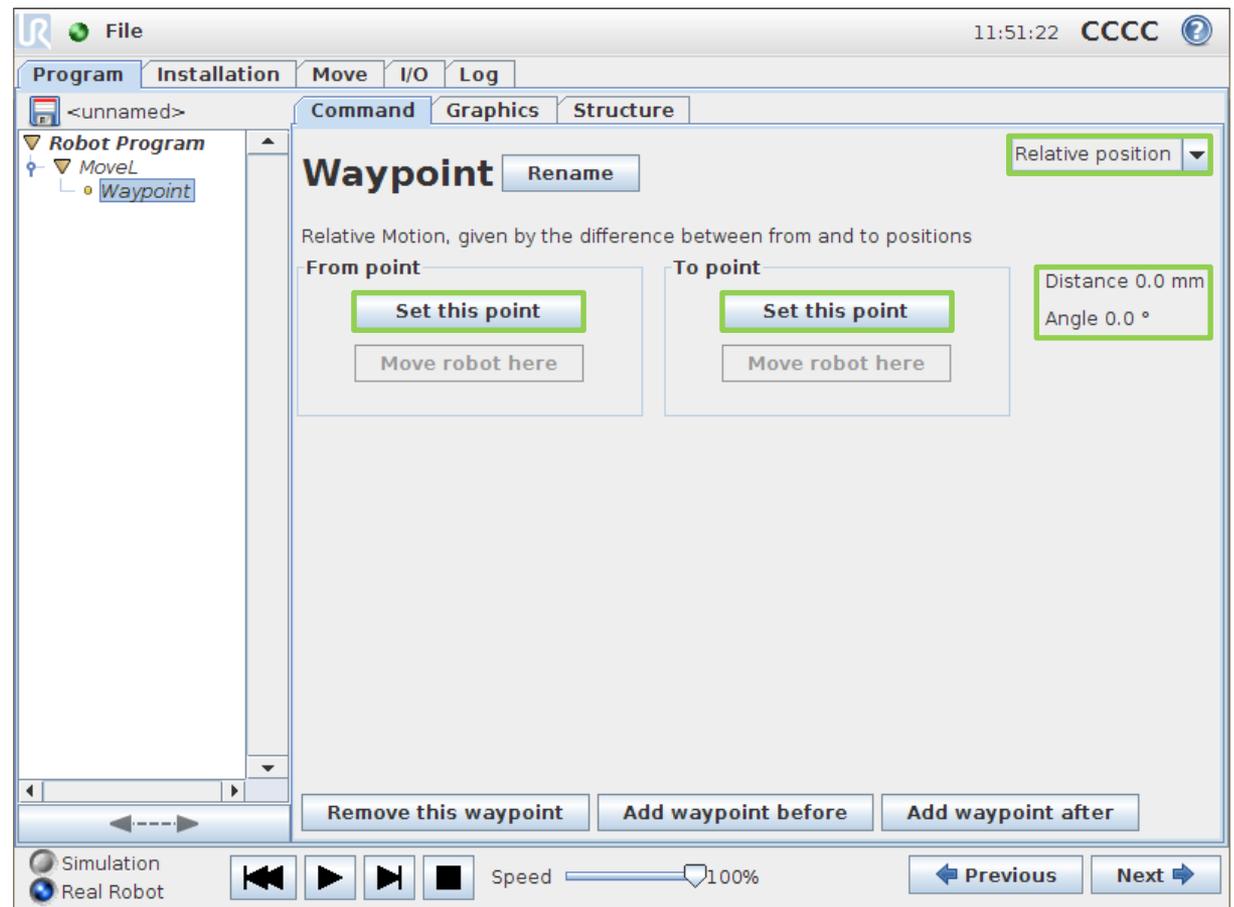
## Pose editor

- Modify
  - Absolute value
  - Add/subtract to value
- Joint positions
  - Angular value in *degrees*
- Tool position
  - Cartesian value in *mm*.
  - Define rotation unit



## Relative waypoint

- Relative movement
  - Linear movement
  - Relative to prior position in program
  - Distance and angle displayed



## Relative waypoint

- Define Relative Waypoint
  - Defined by two Waypoints
  - Can be defined anywhere in robot workspace

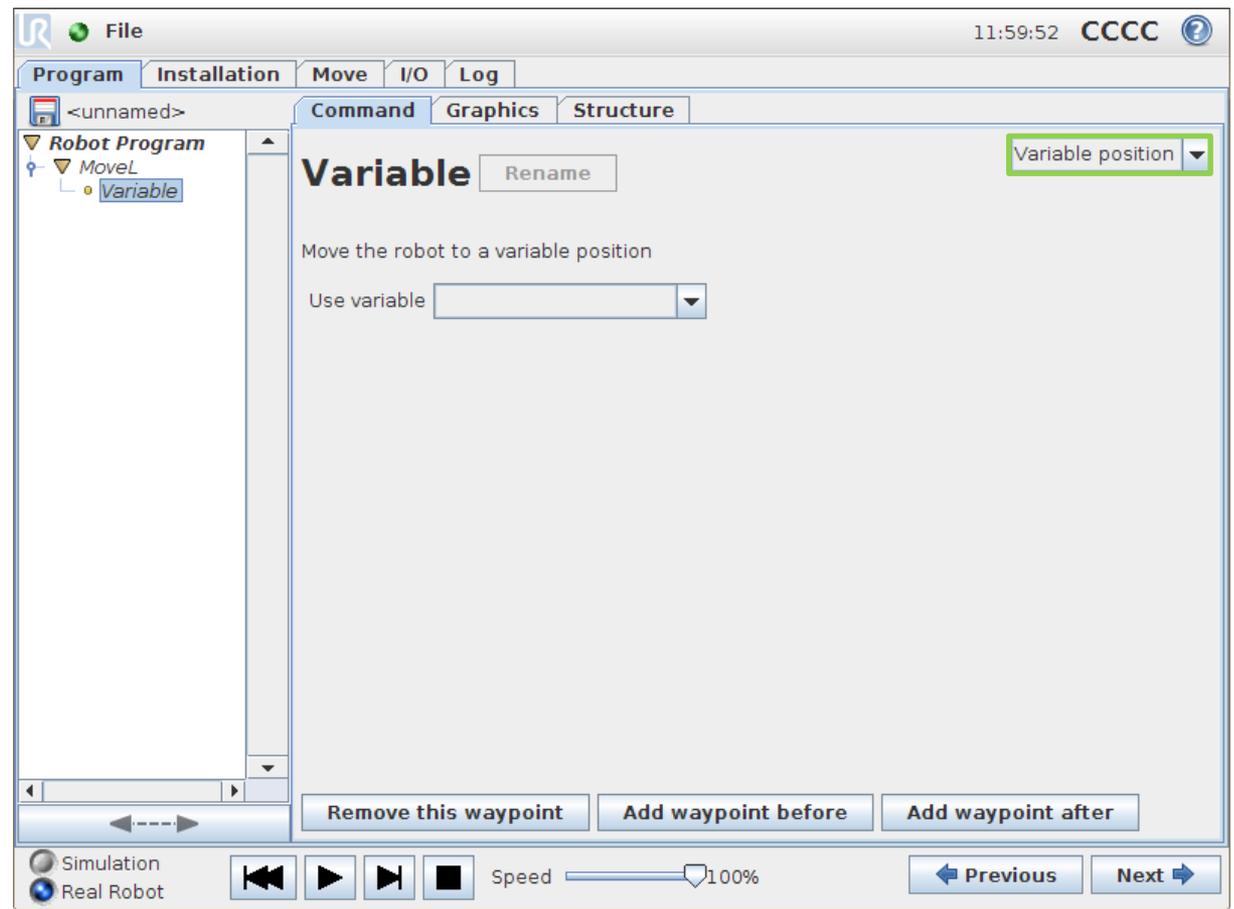
```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2 rel. pos
  Waypoint_3 rel. pos
  Waypoint_4 rel. pos
```



- Save sample program as `moveL_with_relative_waypoint.urp`

## Variable waypoint

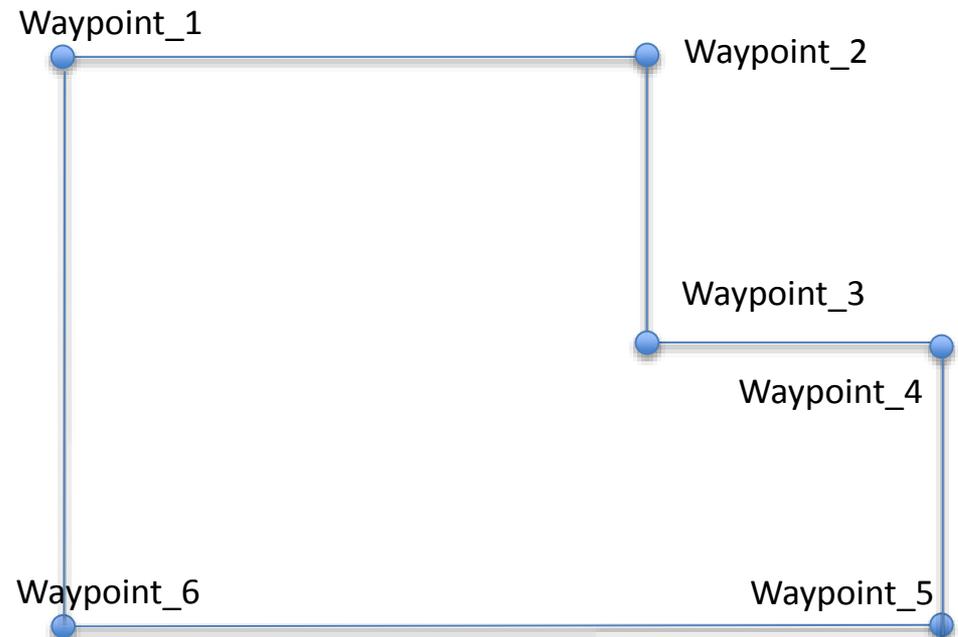
- Variable position
  - Position that can be calculated / offset



- Definition of and how to use variable Waypoints is part of Advanced Training

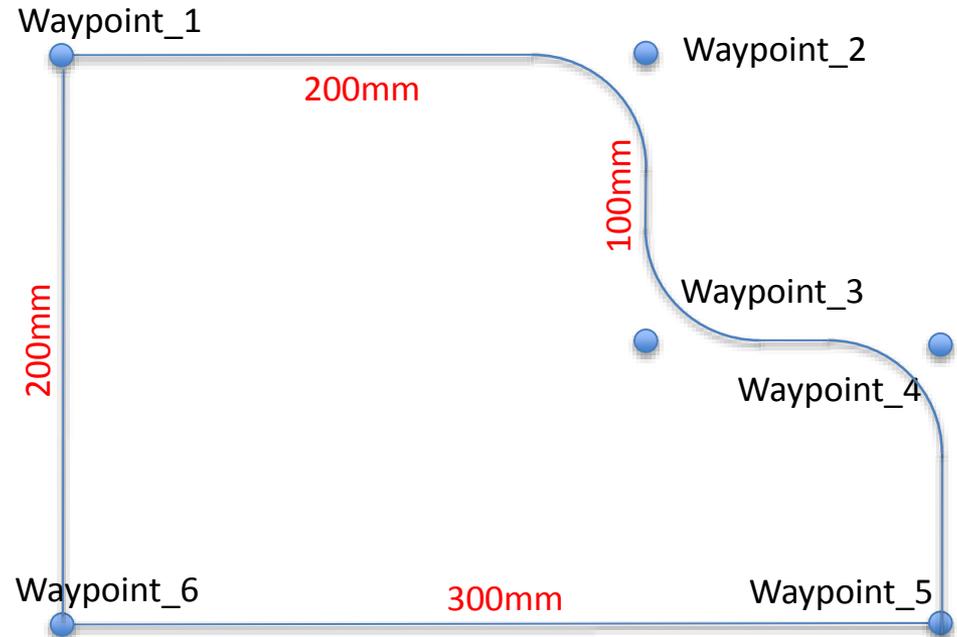
## Lab exercise part 1

- Create a MoveL
  - Create 6 waypoints in positions like those in the diagram.
  - Use the move tab to move the robot tool into appropriate positions
  - This shape can be defined anywhere in the robot workspace



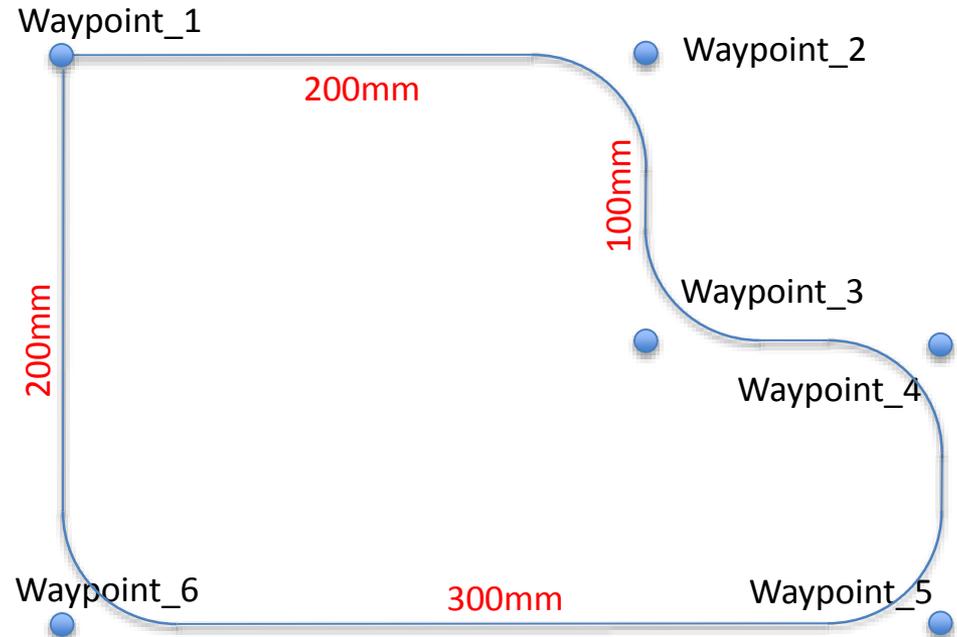
## Lab exercise part 2

- Adjust the waypoints
  - Adjust the waypoints according to the dimensions on the diagram using the pose editor.
  - Add a 50mm radius to waypoints 2, 3 and 4.



## Lab exercise part 2

- Convert to a MoveP
  - Change from a MoveL to a moveP
  - Set a 50mm radius at all points



Hardware

Getting started

Basic commands 1

4 Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

Safety standards

Adjustable safety

### next

- Signal handling
  - Interaction with external devices



## User interface

- Safety Control Board (SCB)

- Safety**

- Emergency stop
- Safeguard stop

- Remote**

- Power on/off

- Power**

- 2A internal PSU
- External PSU

- Config. signals**

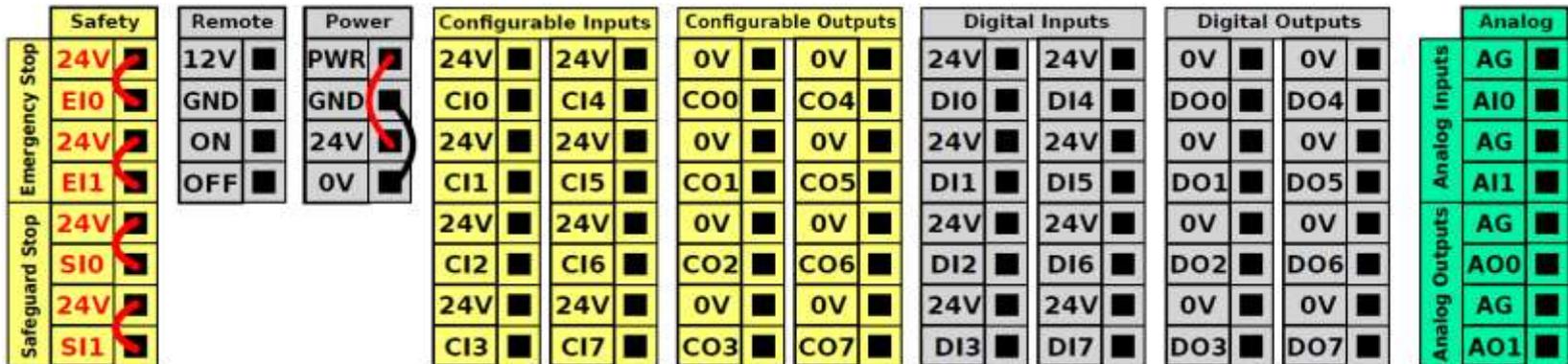
- 8 inputs
- 8 outputs
- 24V DC, PNP

- Digital signals**

- 8 inputs
- 8 outputs
- 24V DC, PNP

- Analog signals**

- 2 inputs
- 2 outputs
- 0-10V DC
- 4-20 mA

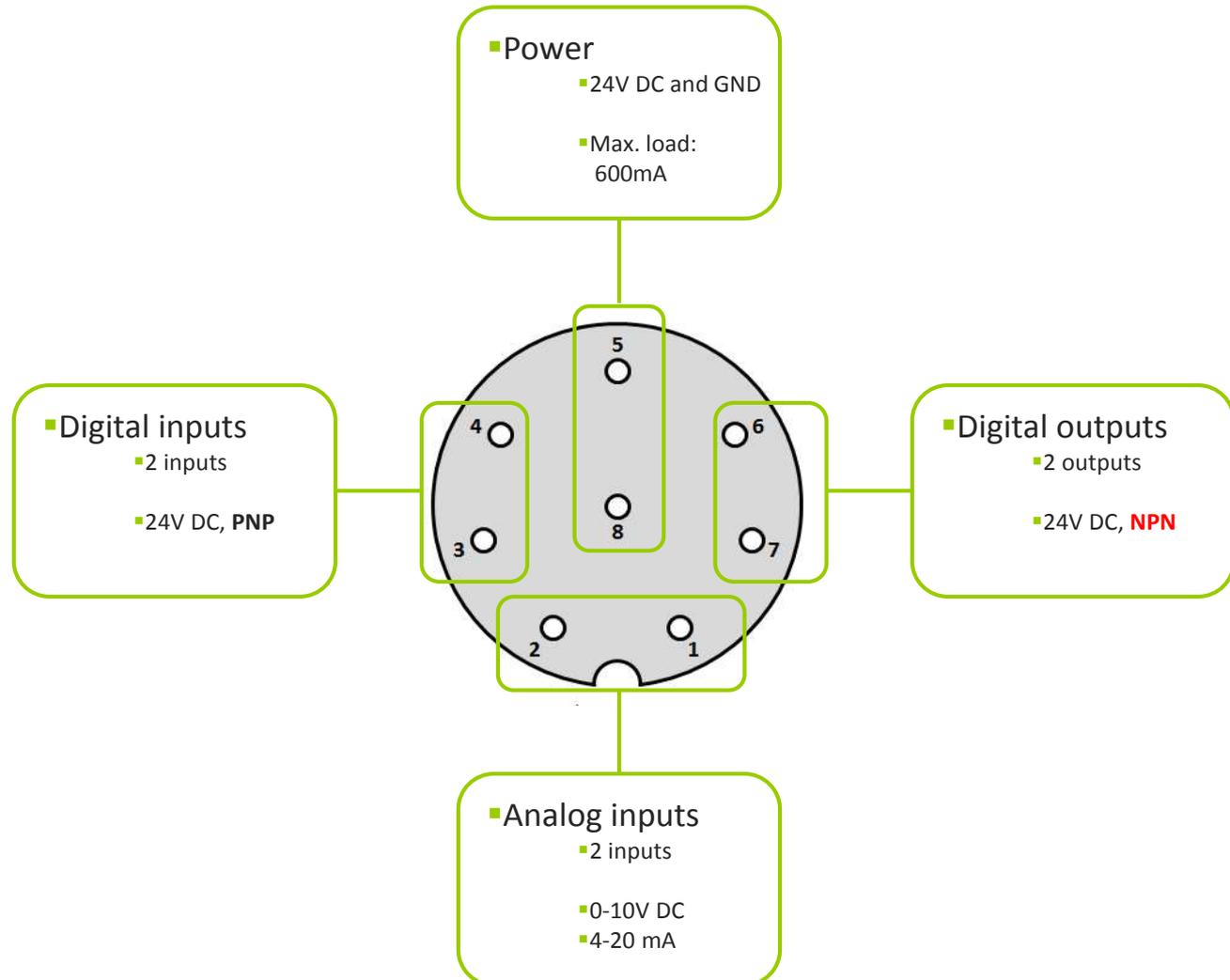


Consult User Manual, chapter 4. Electrical Interface

## User interface

### ■ Tool connector

- pin 1 AI[2]
- pin 2 AI[3]
- pin 3 TI[0]
- pin 4 TI[1]
- pin 5 24V DC
- pin 6 TO[0]
- pin 7 TO[1]
- pin 8 GND



## Signal handling

### Purpose

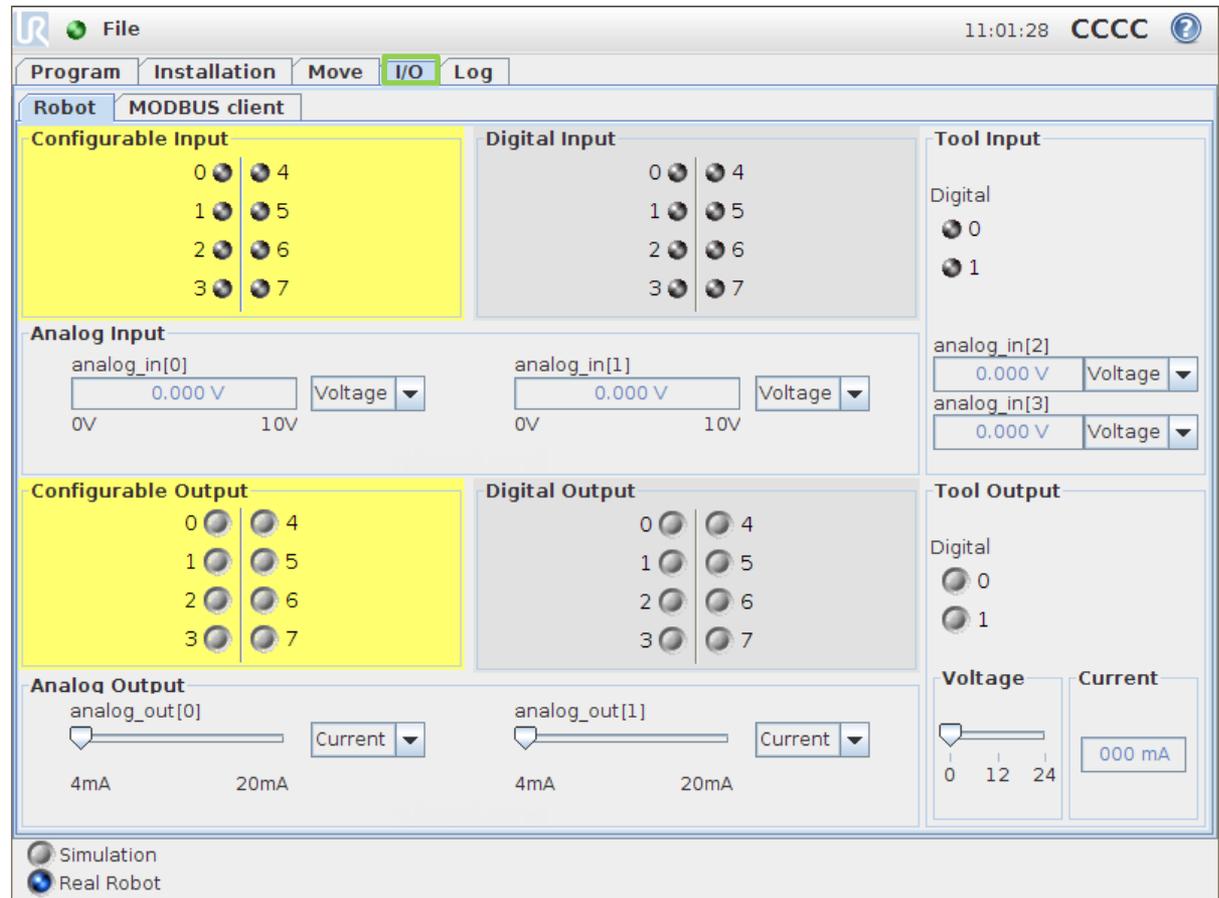
- Interaction with external devices

### Hardwired to

- Control box
- Tool connector

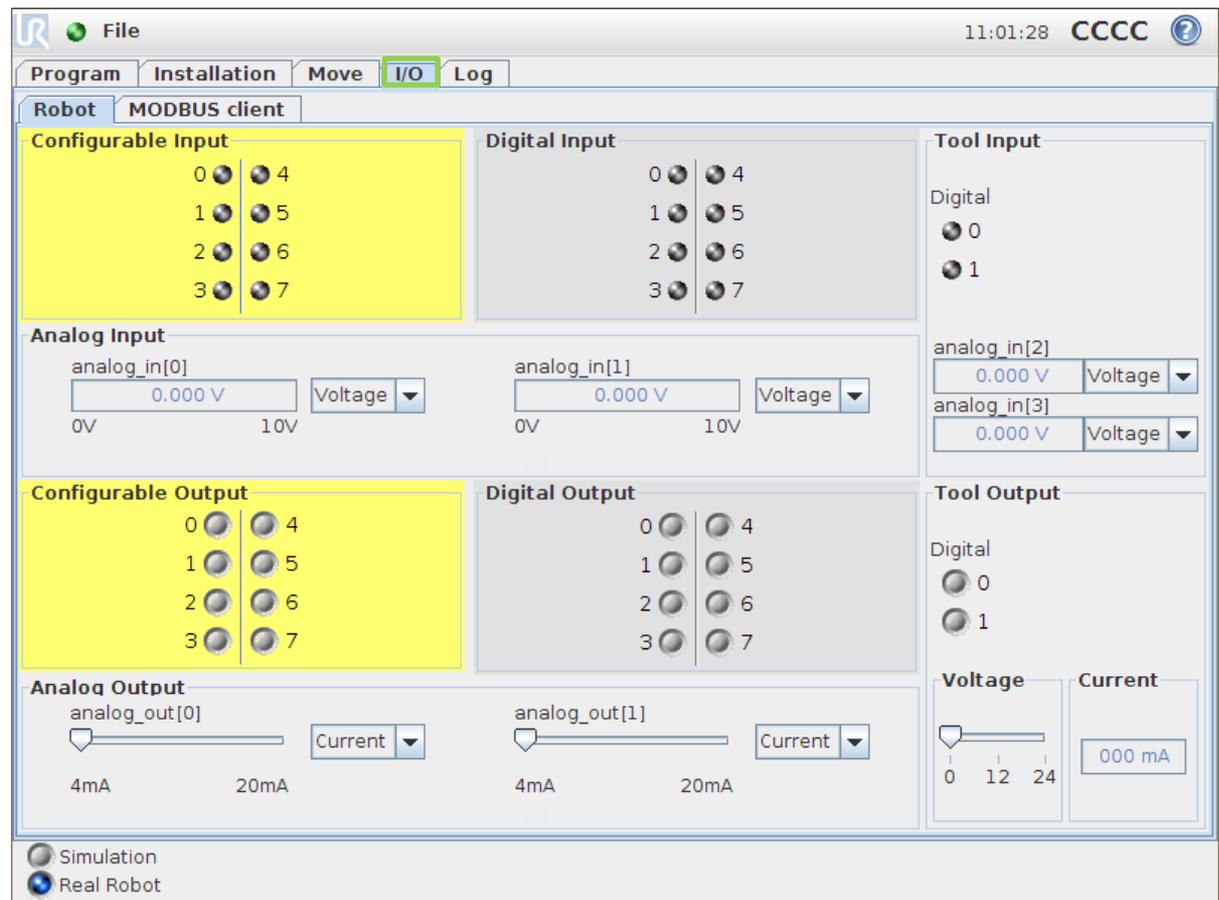
### Configuration

- Control box
  - 16 DI
  - 16 DO
  - 2 AI
  - 2 AO
- Tool connector
  - 2 DI
  - 2DO
  - 2AI



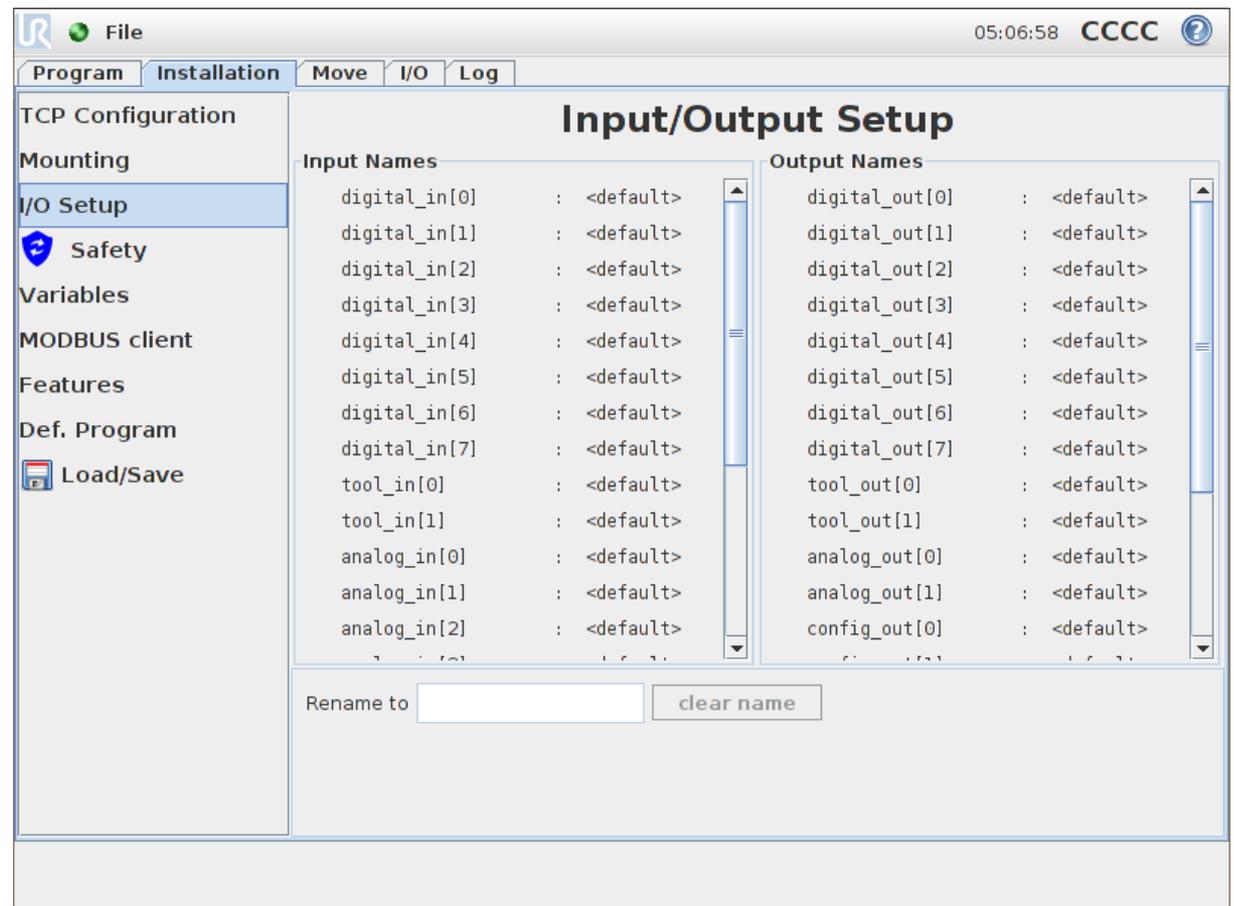
## Signal handling

- Digital signals
  - Voltage
    - 24V DC
    - GND
  - State
    - OFF (Low)
    - ON (High)
  
- Analog signals
  - Range
    - Current 4-20 mA
    - Voltage 0-10 V



## I/O setup

- Input
  - Rename signal
- Output
  - Rename signal
  - I/O tab control
  - Set signal state when stopped

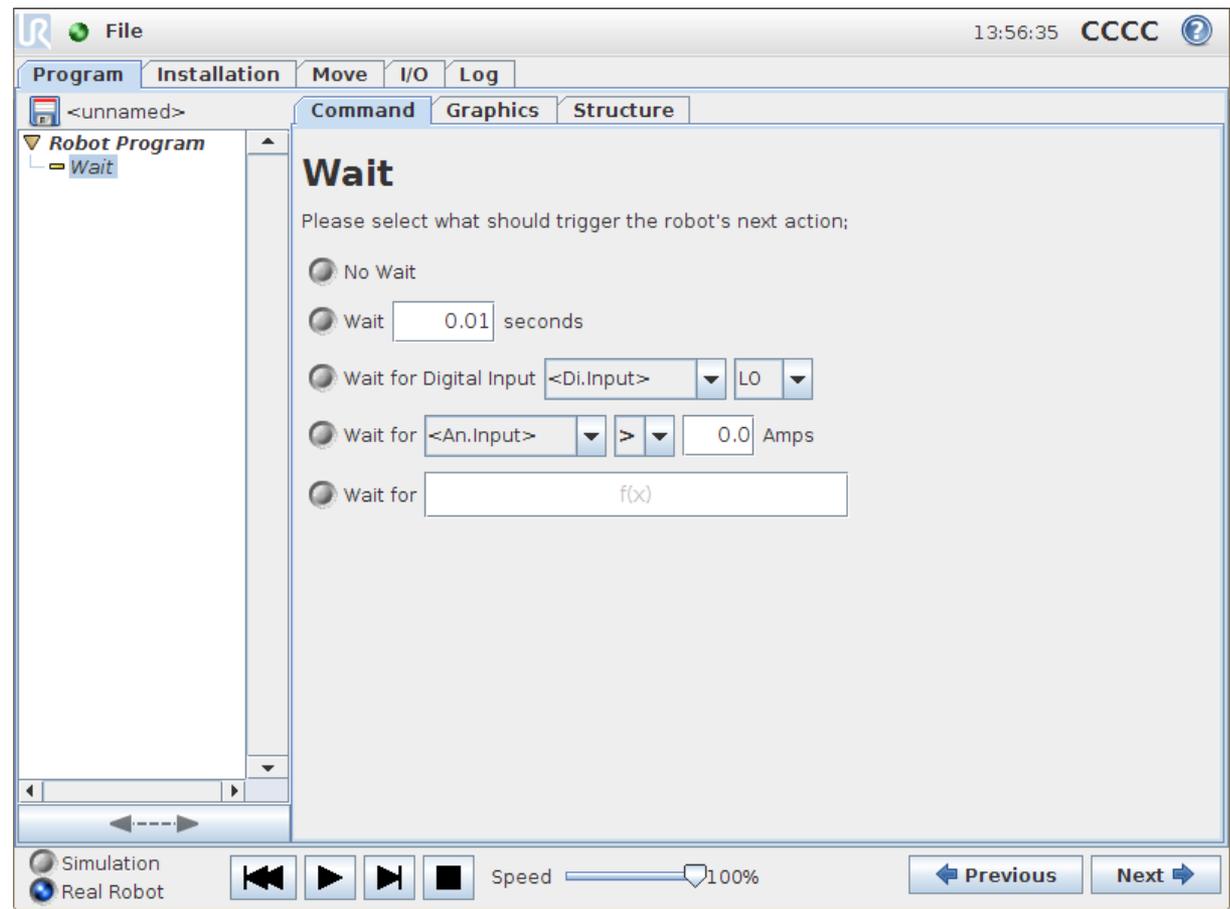


- Program and I/O tab is affected when renaming

## Wait command

- Wait for condition
  - No action
  - Wait an amount of time
  - Wait for digital input
  - Wait for analog input
  - Wait for *<expression>*

```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2
  Wait 1.0
  Waypoint_3
  Waypoint_4
  Wait DI[0] = True
```



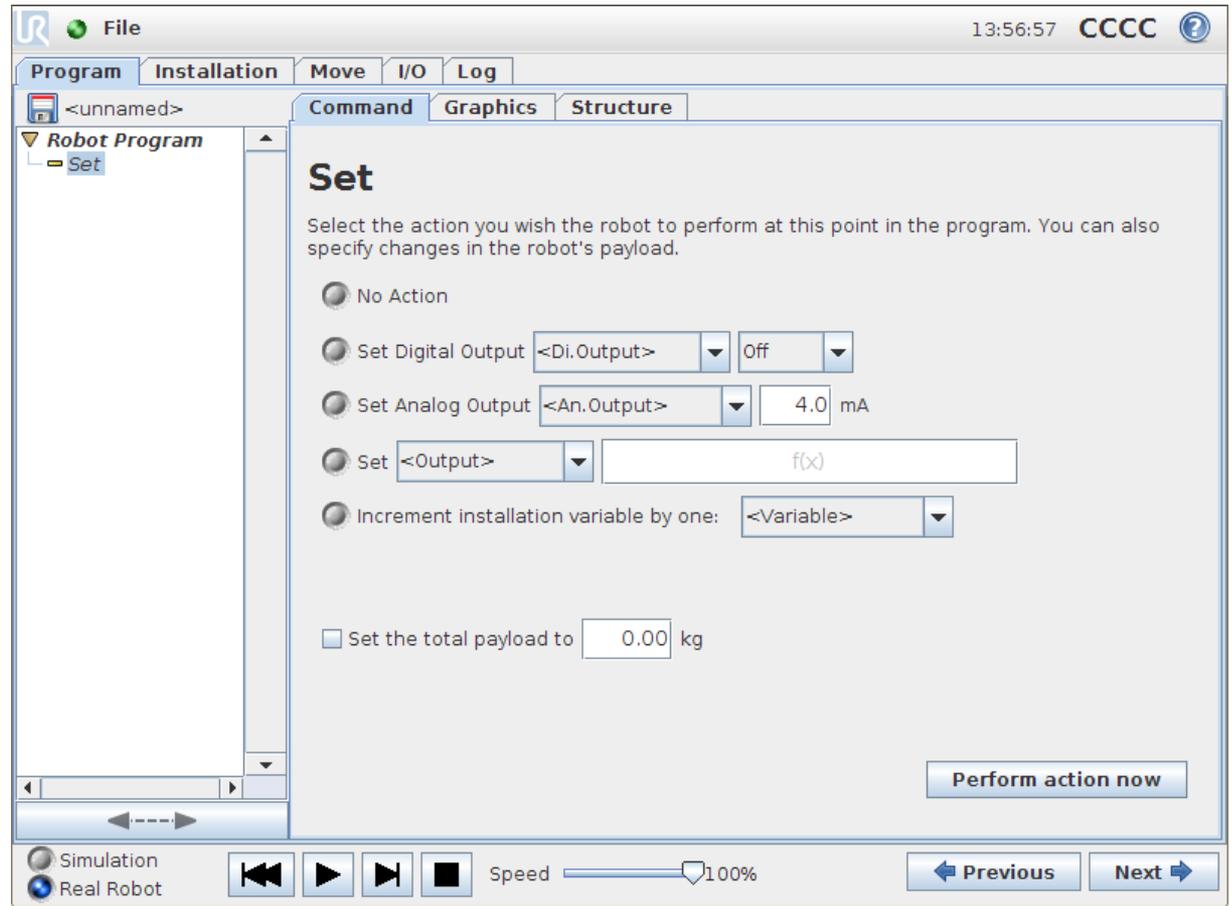
- Save sample program as wait.urp

## Set command

- Perform action
  - No action
  - Set digital Output
  - Set analog Output
  - Set *<expression>*
  - Increment variable

```

Robot Program
MoveL
  Waypoint_1
  Set DO[0] = True
  Waypoint_2
  Waypoint_3
  Waypoint_4
  Set DO[0] = False
    
```

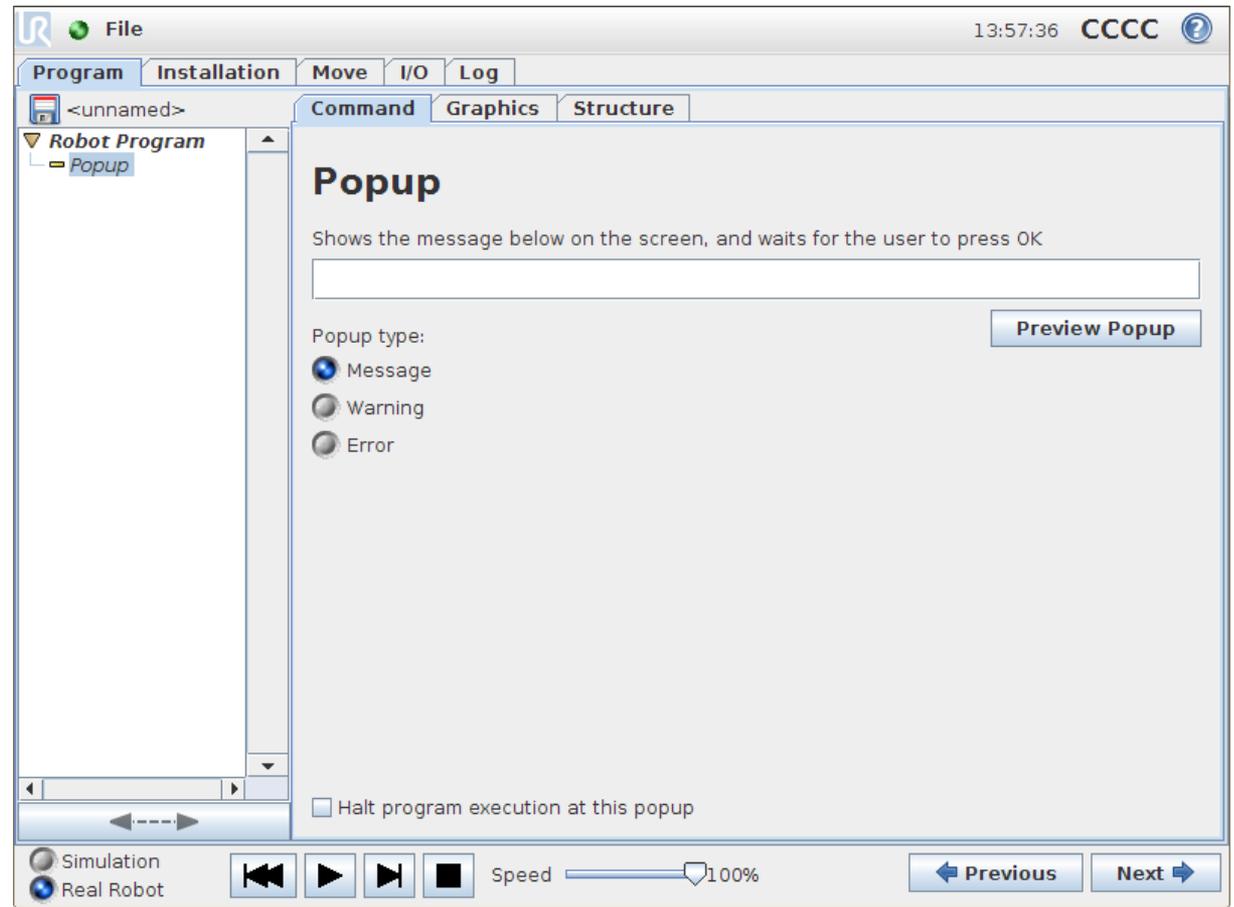


- Save sample program as set.urp

## Popup command

- Wait for operator
  - Pauses program
  - Define popup message
  - Popup types
    - Message
    - Warning
    - Error

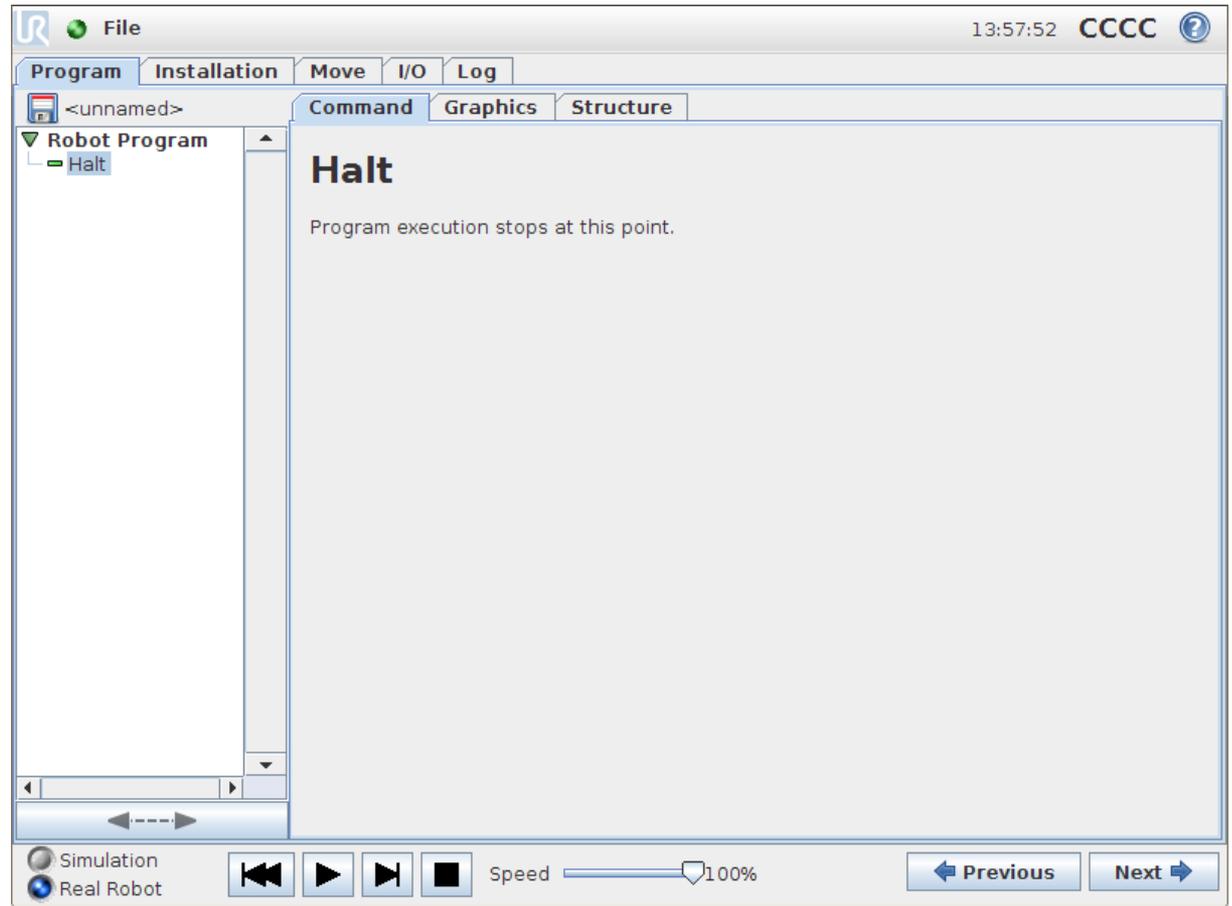
```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2
  Wait 1.0
  Waypoint_3
  Waypoint_4
  Popup
```



- Save sample program as popup.urp

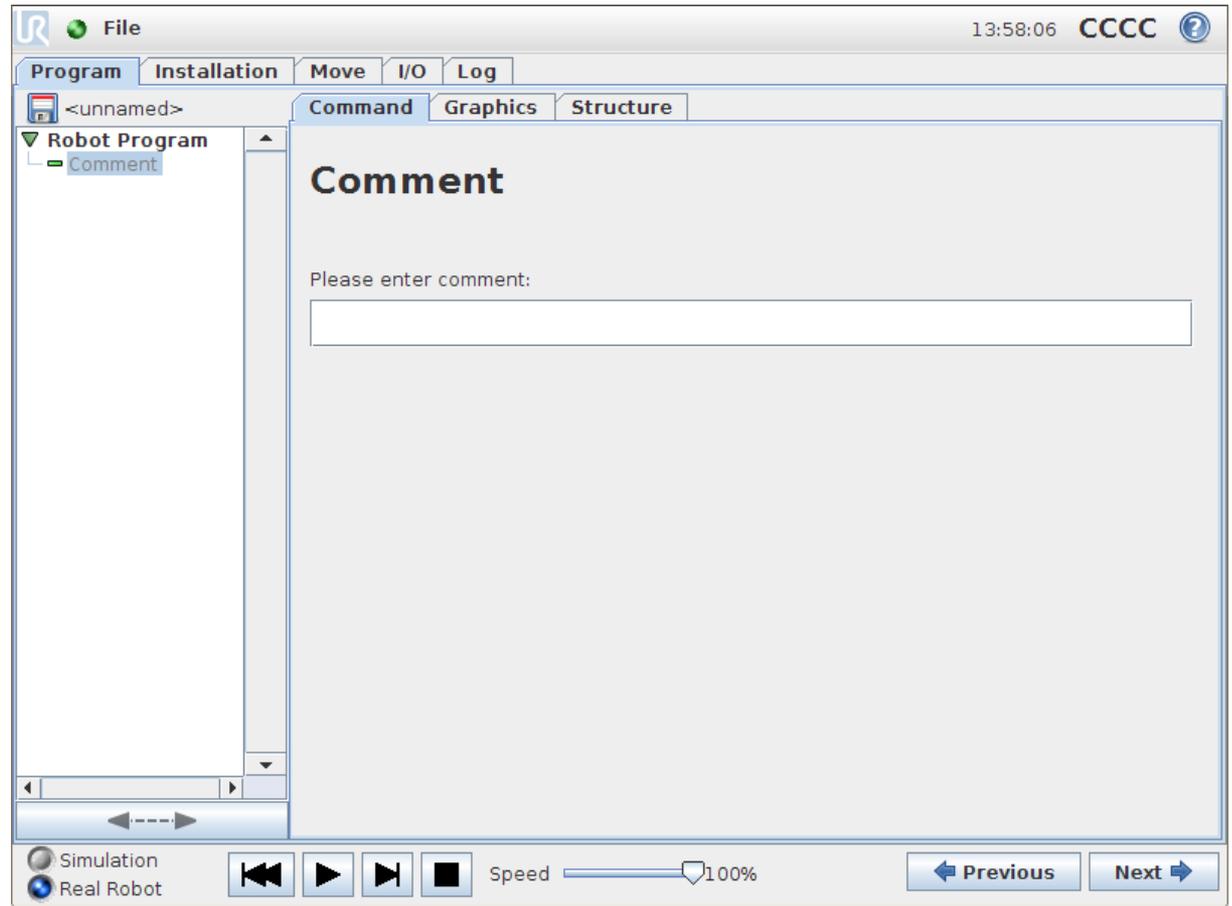
## Stop command

- Halt
  - Stops program execution



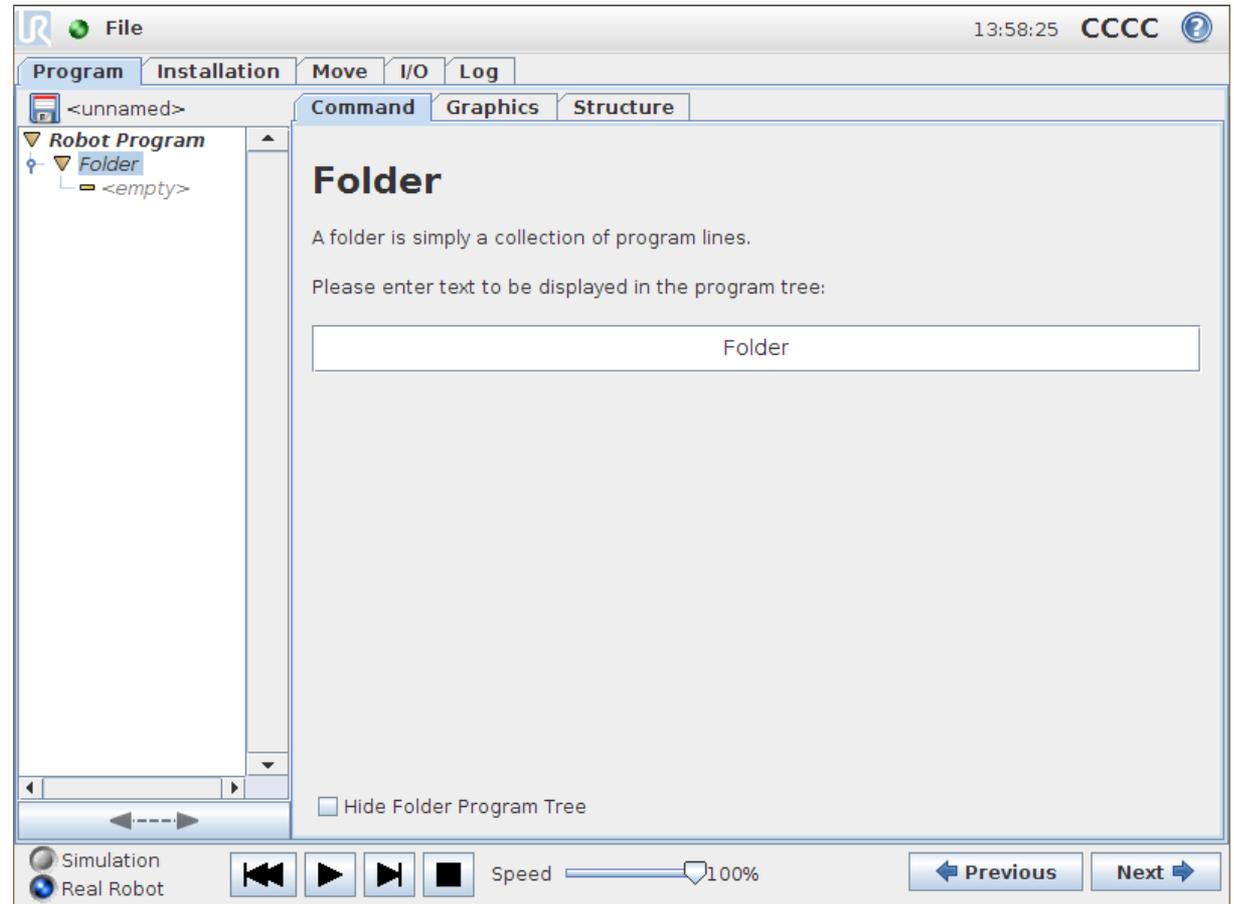
## Comment command

- Comments
  - Add text to program
  - Program execution *not* affected



## Folder command

- Folders
  - Organizing program
  - Label part of program
- Program execution *not* affected



## Folder command

- Sample pick'n'place program
  - Two folders
    - Pick\_part
    - Place\_part

### Robot Program

*Pick\_part*

MoveL

Waypoint\_1

Waypoint\_2

Set DO[0] = On

Wait 0.5

Waypoint\_1

*Place\_part*

MoveL

Waypoint\_3

Waypoint\_4

Set DO[0] = Off

Wait 0.5

Waypoint\_3

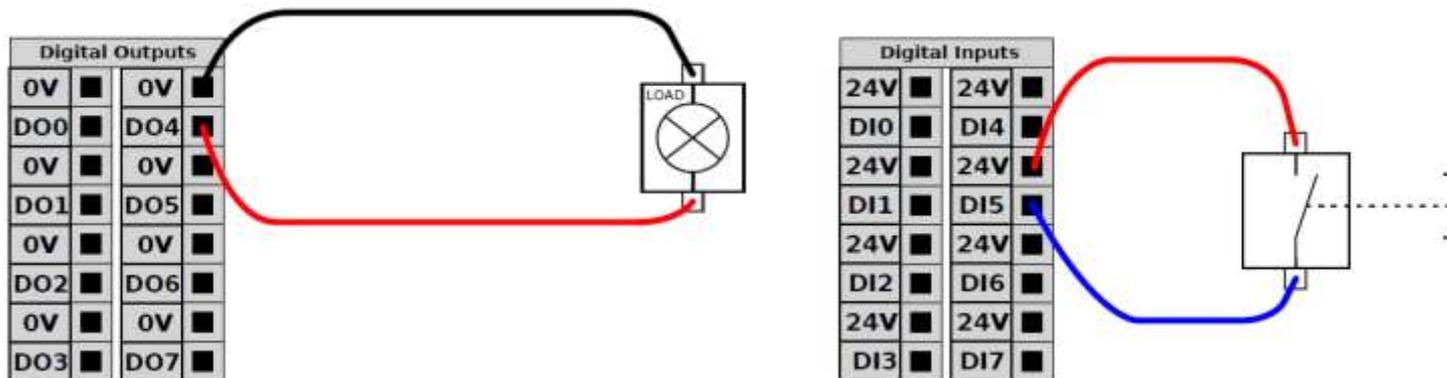


- Save sample program as pick\_and\_place.urp

## 4 Basic commands 2

### Lab exercise part 1

- Create a simple signal handling program
- Setup:
  - Rename `digital_out[4]` to "Lamp" and `digital_in[5]` to "Button"
  - Connect the provided Lamp to Digital Output 4 in the control box as shown below.
  - Connect the provided button to Digital Input 5 in the control box as shown below.



### Lab exercise part 2

#### ■ Program – Button\_Folder

- Create a folder called Button\_Folder
- Wait for the button to be pushed before continuing
- Add a comment at the start of the folder explaining what this code does.

#### ■ Program – Lamp\_Folder

- Create a folder called Lamp\_Folder
- Turn on the lamp, wait for 5 seconds then turn it off again.
- Create a popup that informs the user the program has completed, and halt the program at this popup.
- Add a comment at the start of the folder explaining what this code does.

Hardware

Getting started

Basic commands 1

Basic commands 2

**5** Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

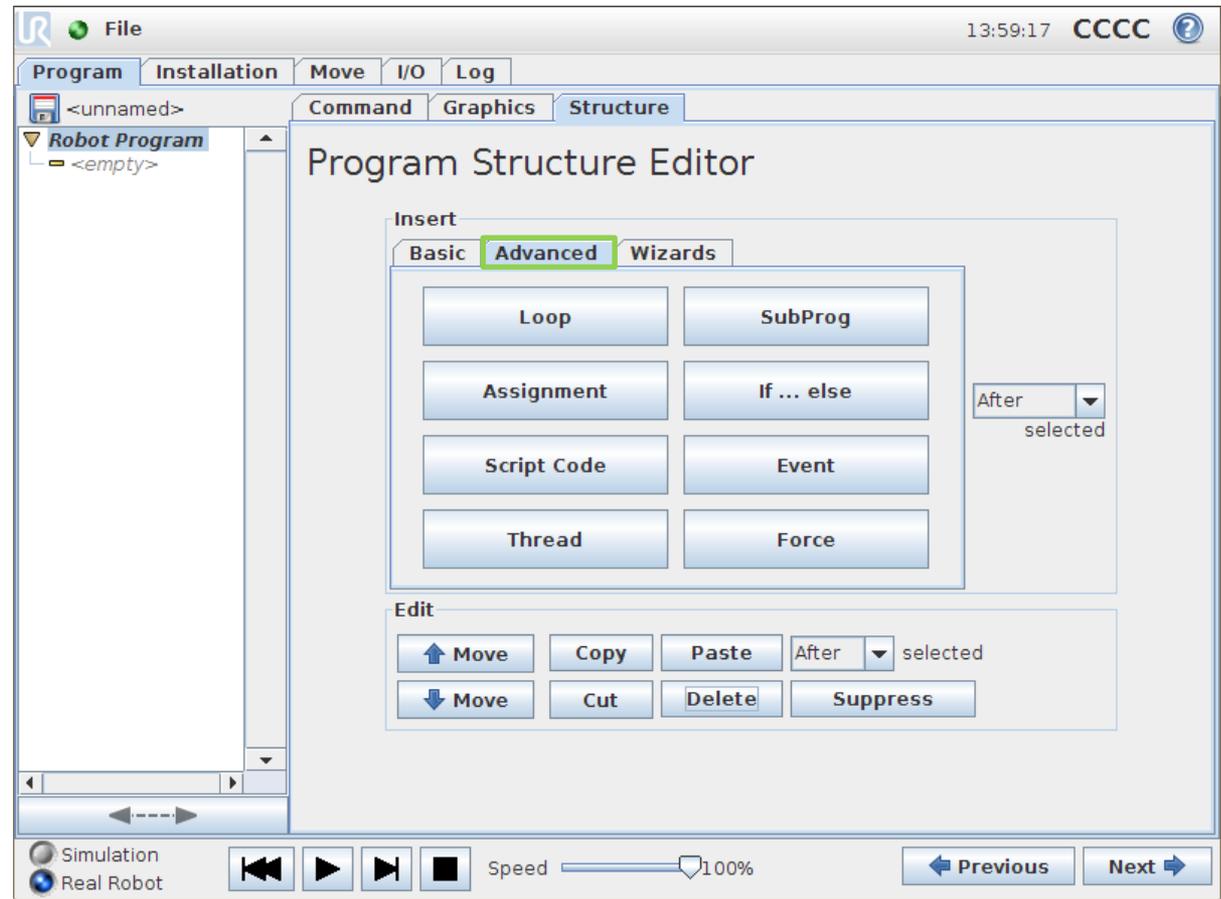
Safety standards

Adjustable safety

# 5 Advanced commands 1

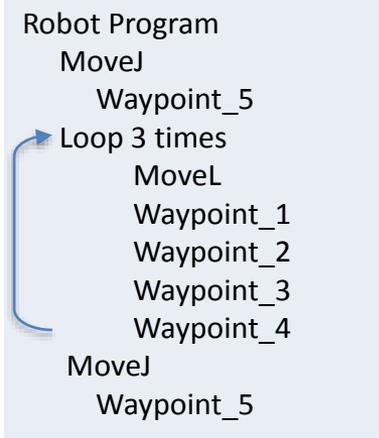
## Advanced commands tab

- Advanced commands
  - Perform advanced operations



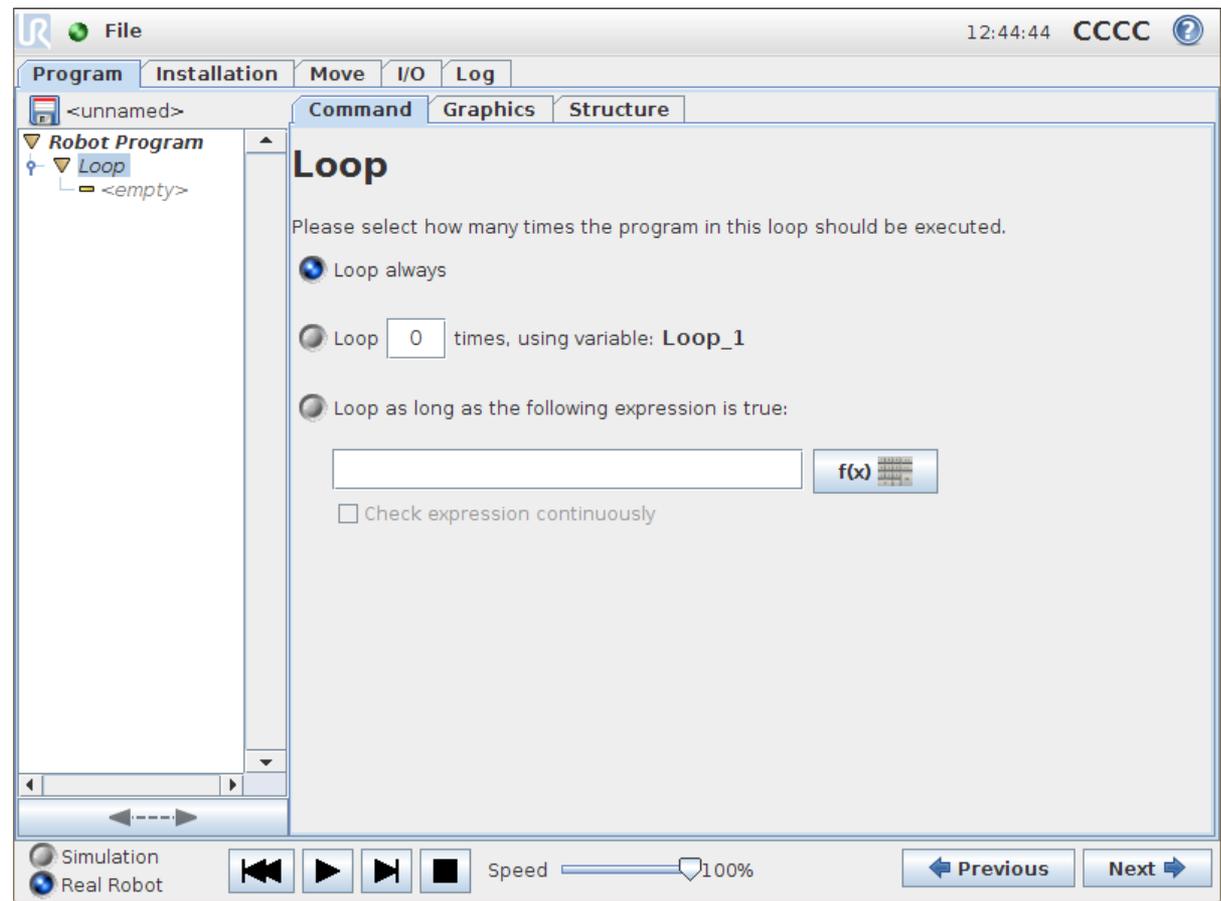
## Loop command

- Definition
  - Loop underlying commands no. of times
  - Loop types
    - Loop always
    - Loop n times
    - Loop <expression>



Robot Program  
MoveJ  
Waypoint\_5  
Loop 3 times  
MoveL  
Waypoint\_1  
Waypoint\_2  
Waypoint\_3  
Waypoint\_4  
MoveJ  
Waypoint\_5

The diagram shows a sequence of commands in a robot program. A blue arrow points from the 'Loop 3 times' command to the 'Waypoint\_1' through 'Waypoint\_4' block, indicating that these four commands are repeated three times.



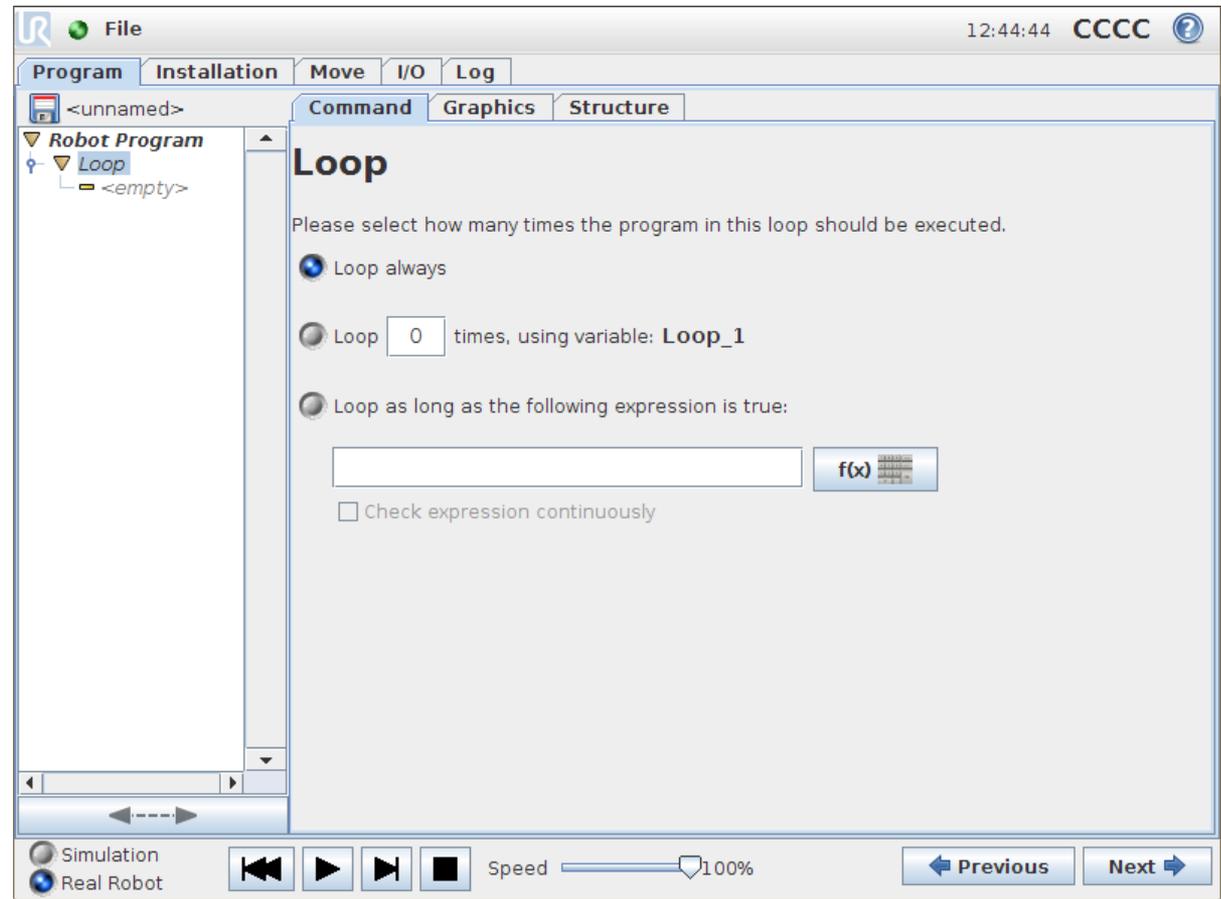
- Save sample program as loop.urp

# 5 Advanced commands 1

## Loop command

- Interrupt
  - Check expression continuously

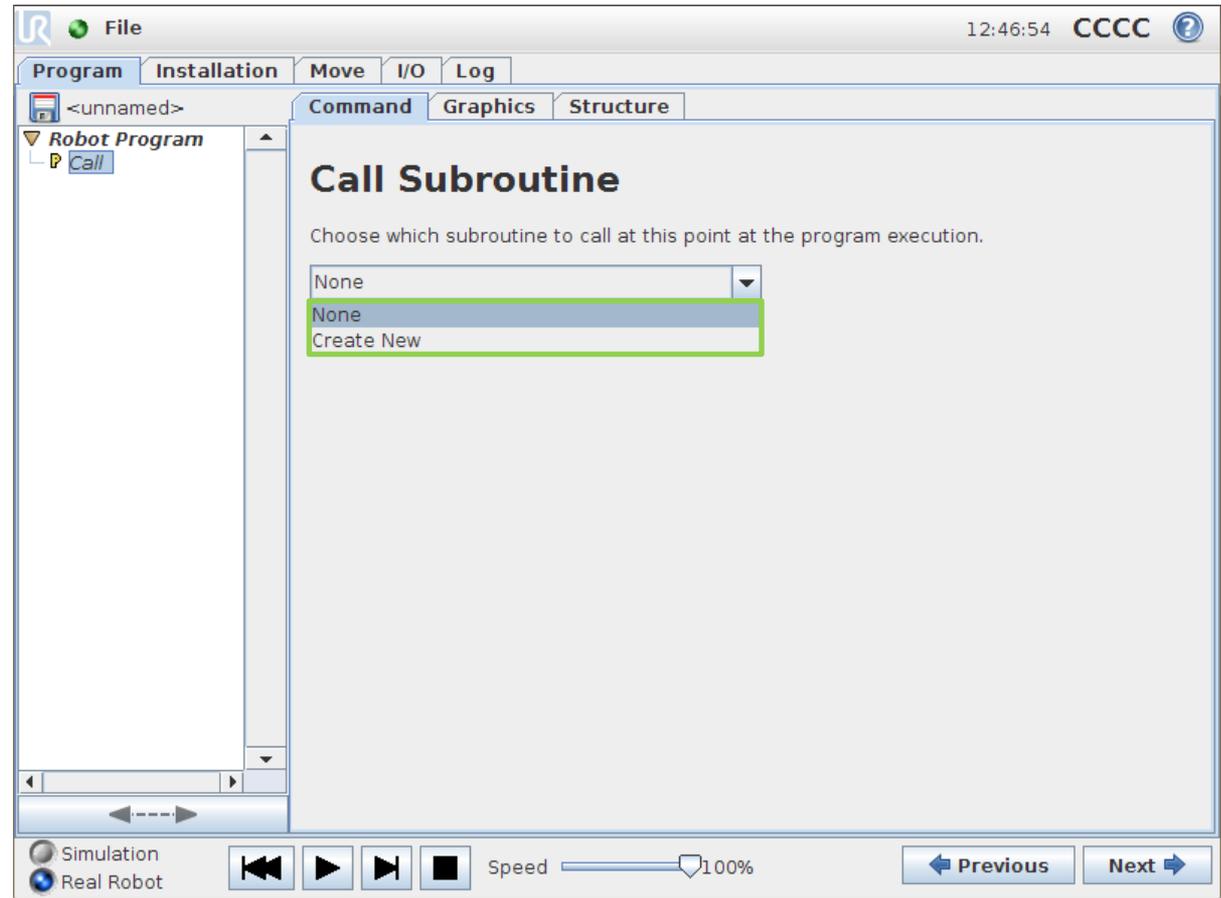
```
Robot Program
MoveJ
  Waypoint_5
While DI[0] = True
  MoveL
  Waypoint_1
  Waypoint_2
  Waypoint_3
  Waypoint_4
MoveJ
  Waypoint_5
```



- Save sample program as loop\_interrupt.urp

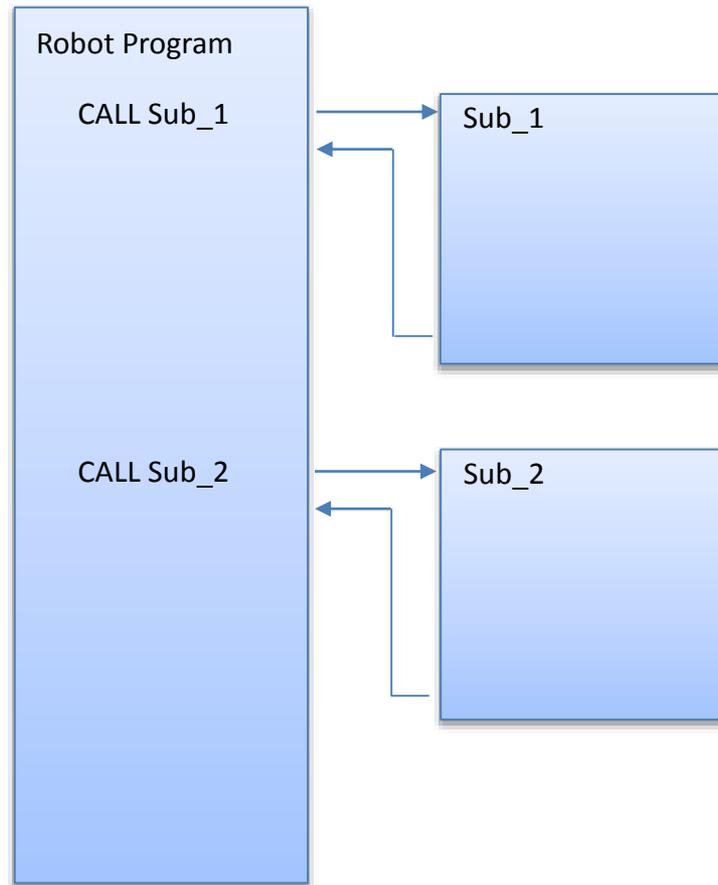
## SubProg command

- Subprograms
  - Organize program
  - Split program into subprograms
  - Re-use subprogram in multiple programs



## SubProg command

- Calling a subprogram
  - All commands in subprogram executed
  - Then return to "main" program



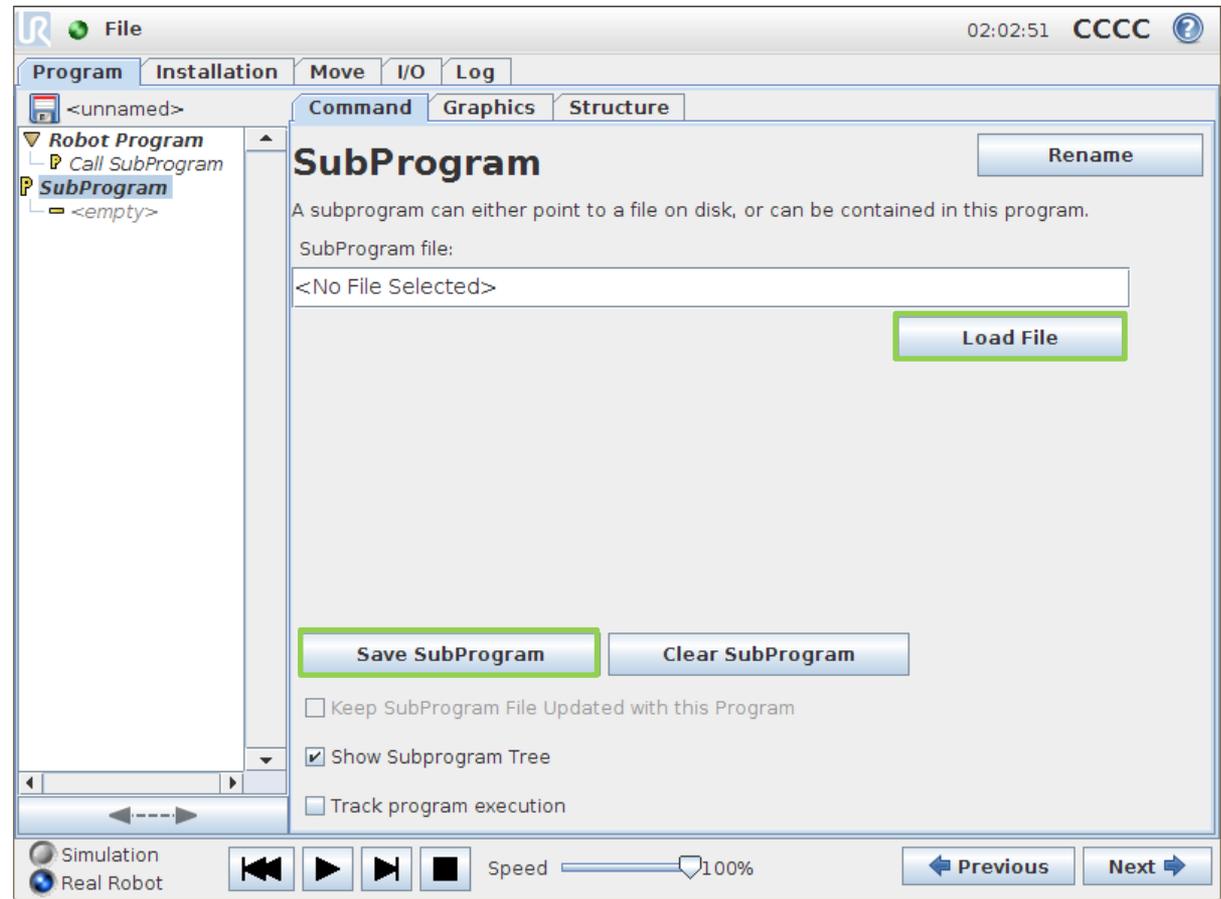
- Note: Calling subprogram from subprogram is not supported

# 5 Advanced commands 1

## SubProg command

- Call subprogram
  - Load existing file
  - Create program in empty program
    - Save as file
    - Part of main program

```
Robot Program
  MoveJ
    Waypoint_5
  Loop 3 times
  MoveL
    Waypoint_1
    Waypoint_2
    Waypoint_3
    Waypoint_4
  Call SubP_movec
  MoveJ
    Waypoint_5
```



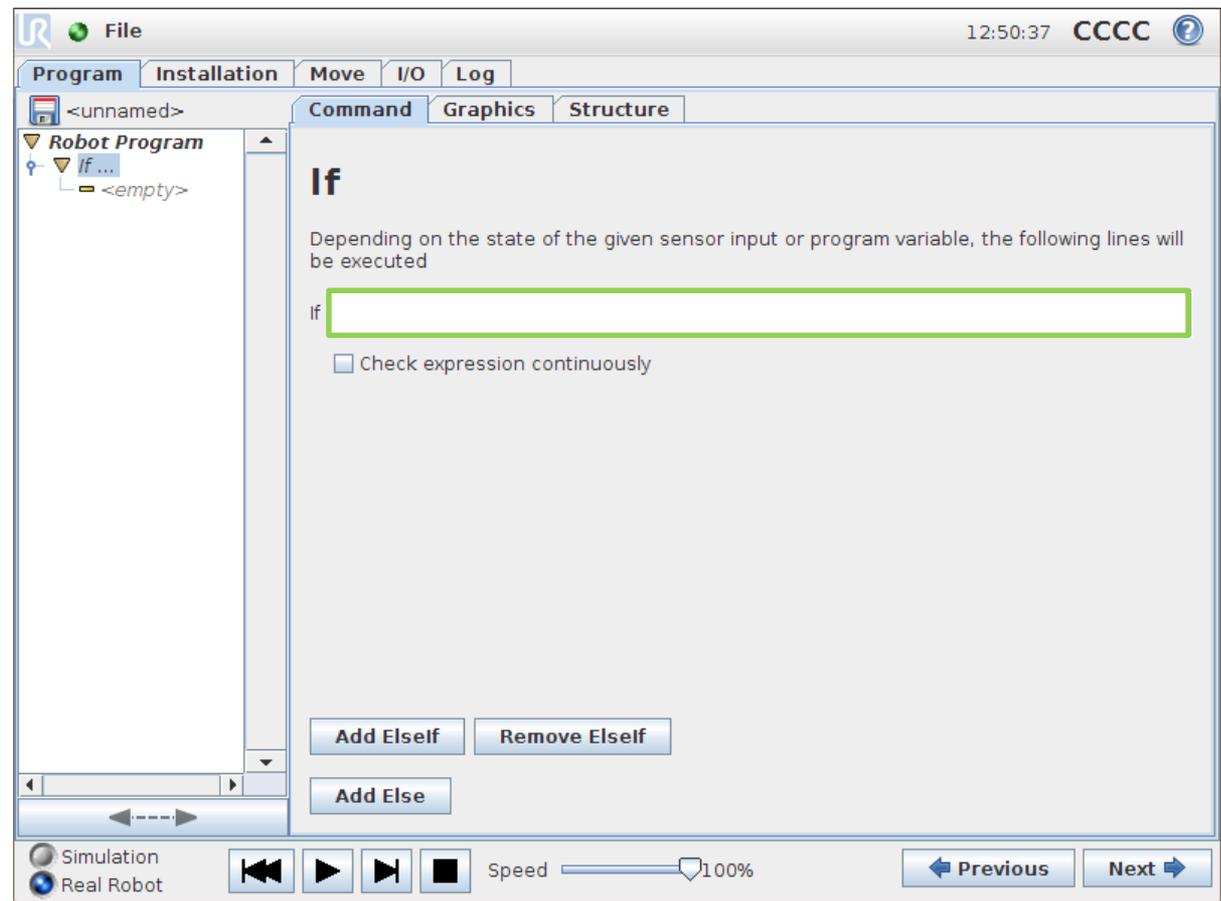
- Save sample program as call\_sub.urp

# 5 Advanced commands 1

## If ... else command

- If
  - Examines a condition:
    - State of sensor
    - Value of variable
    - Combination of various states
  - If condition = True
    - Execute underlying commands

```
Robot Program
MoveL
  Waypoint_1
  IF DI[0] = True
    Waypoint_2
  Waypoint_3
```



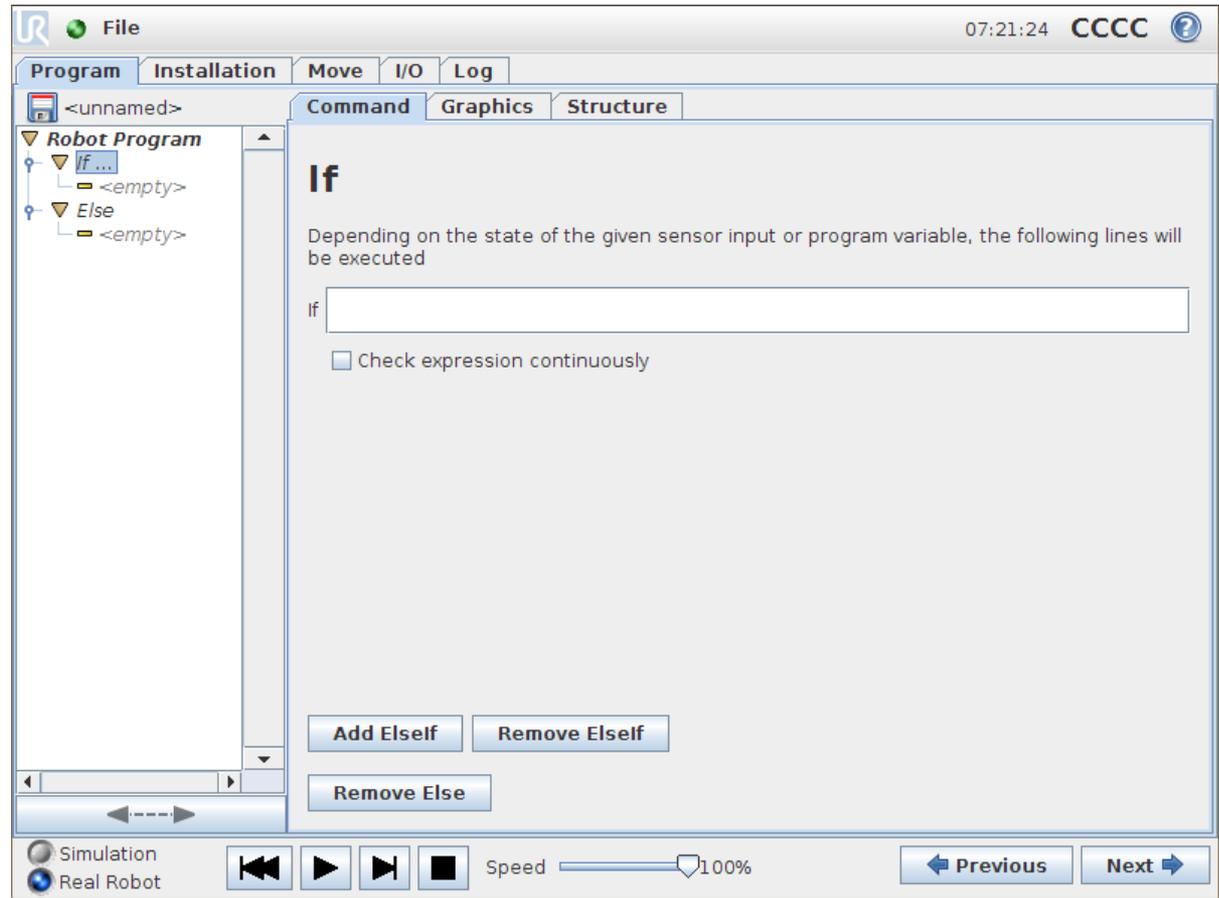
- Save sample program as if.urp

# 5 Advanced commands 1

## If ... else command

- ... elseif
  - Examines new condition
- ... else
  - Define what to do when no condition is not true

```
Robot Program
MoveL
  Waypoint_1
  IF DI[0] = True
    Waypoint_2
  Else
    Waypoint_3
```



- Save sample program as if\_else.urp

### Lab exercise part 1

- Create a program using sub programs, loops and if statements
- This will use the installation from Basic Commands 2 Exercise
- First create a simple program to flash the lamp (This will be used as a sub program)
- Lamp\_Flash:
  - Create a program that flashes the lamp 5 times using a loop
  - On for 0.5 seconds then off for 0.5 seconds (x5)
  - Save this program as Lamp\_Flash

### Lab exercise part 2

- Create another empty program
  - Create 3 waypoint in positions of your choice
  - Move to waypoint\_1
  - Wait for the button to be pushed
  - When button pushed call Lamp\_Flash sub program.
  - Move to waypoint\_2
  - Wait 1 second for potential button push
  - If button pushed move to waypoint\_1
  - Else move to waypoint\_3
  - Move between point 1 and 2 repeatedly, stop immediately upon button push



Hardware

Advanced commands 3

Getting started

Wizards

Basic commands 1

Modbus TCP

Basic commands 2

Service

Advanced commands 1

Safety standards

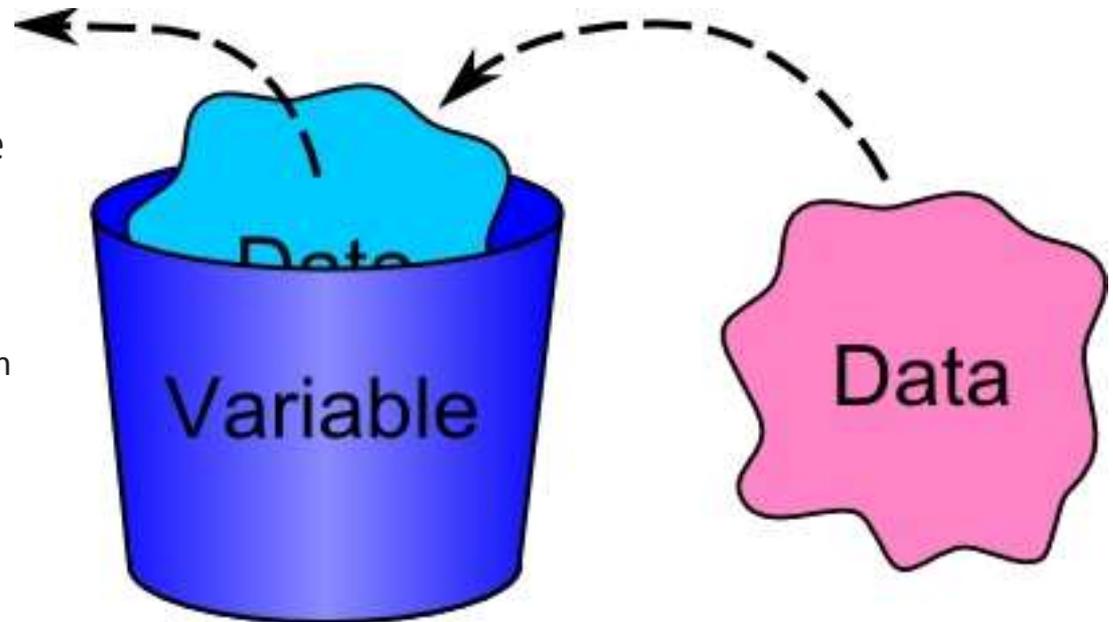
6

Advanced commands 2

Adjustable safety

### What is a variable?

- Variable is a storage location (container)
  - Content of container can change
- Variables are read and write
  - Value can be overwritten
  - Value can be read
- Variable can be compared with other variable or sensor state



### Variable types

var type	value
boolean	true/false
integer	16 bit, whole number
floating point	real number (decimal)
string	ASCII characters (text)
pose	position variable p[x,y,z,rx,ry,rz]
list	array of variables

### Variable scope

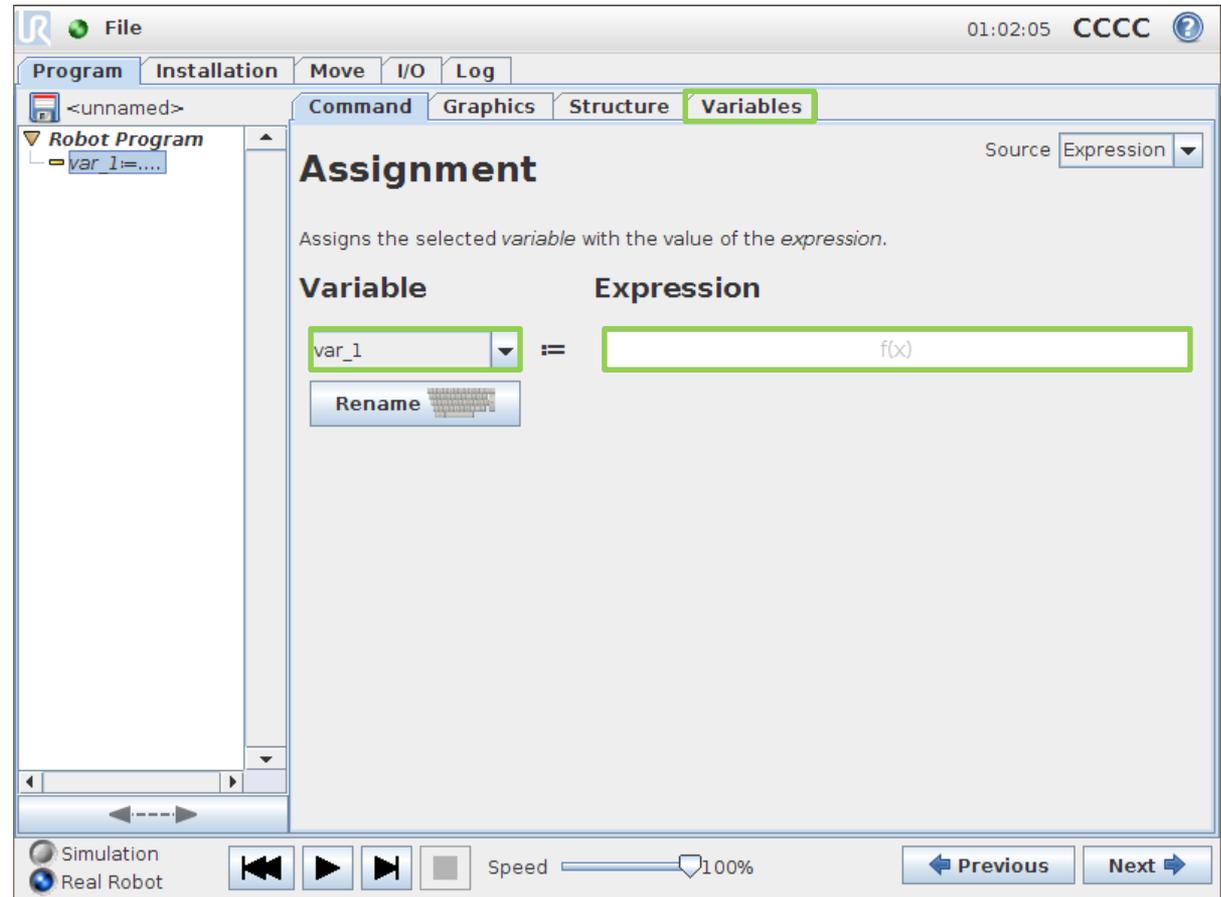
scope	location
local	program
global	installation

- Local variables
  - Declared in program
  - Accessible from same program
  - Value cleared at power down
- Global variables
  - Declared in Installation
  - Accessible from programs using same Installation
  - Value stored in file on disk

## Assignment command

- Options
  - Define variable name
  - Declare variable type
  - Assign value to variable

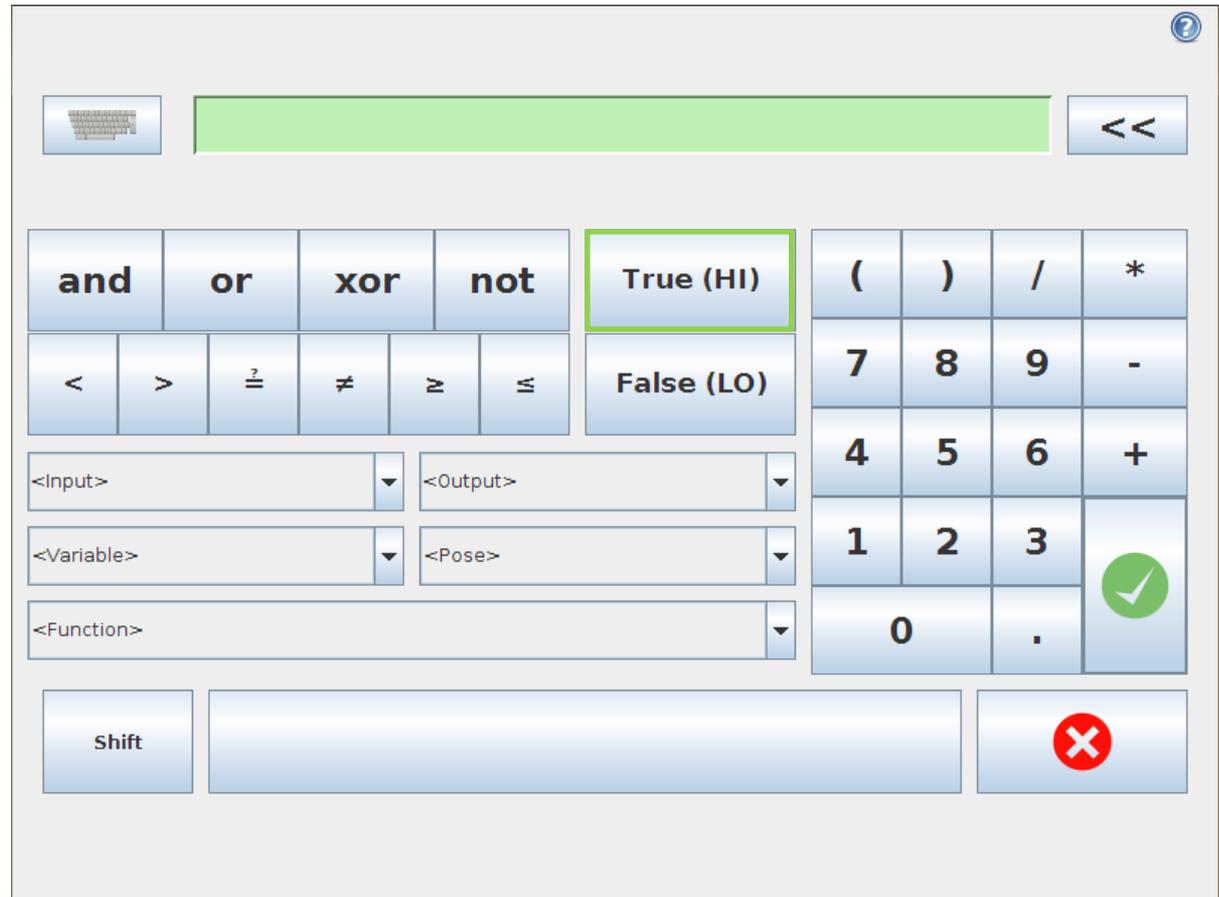
```
Robot Program  
var_1 = True  
Wait 0.5  
var_1 = False  
Wait 0.5
```



- Save sample program as var\_bool.urp

## Expression editor

- Options
  - Numerical values
  - Inputs
  - Outputs
  - Variables
  - Poses
  - Script codes
  - Logic operators
  - Keyboard

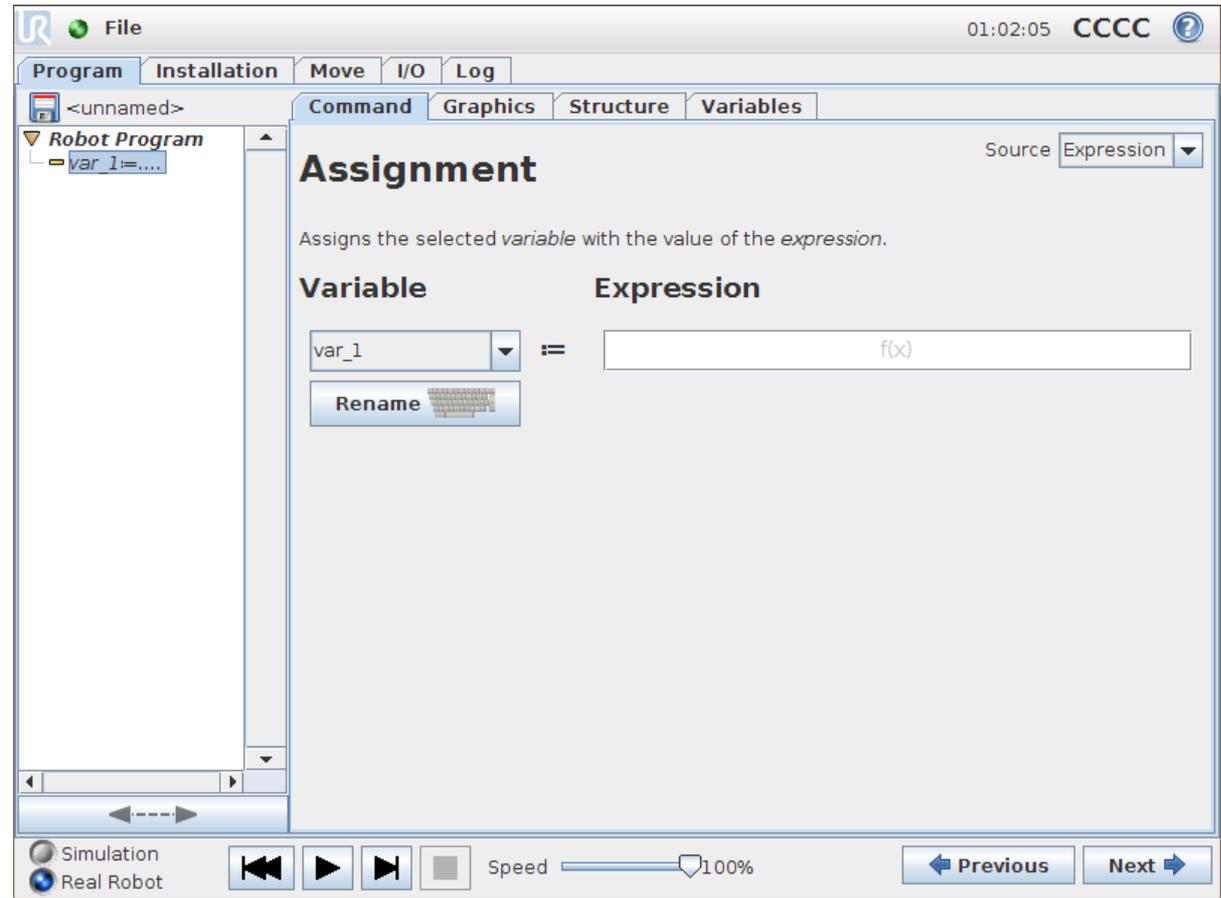


The screenshot shows the Expression Editor interface. At the top, there is a keyboard icon and a long green text input field with a double-left arrow button on the right. Below this is a grid of buttons for logic operators: **and**, **or**, **xor**, **not**, **True (HI)** (highlighted with a green border), and **False (LO)**. To the right of these is a numeric keypad with buttons for parentheses, division, multiplication, numbers 7-9, subtraction, numbers 4-6, addition, numbers 1-3, a checkmark, 0, and a decimal point. Below the operator grid are four dropdown menus: **<Input>**, **<Output>**, **<Variable>**, and **<Pose>**. At the bottom, there is a **shift** button, a large empty text input field, and a red **X** button.

## Use variable as counter

- Counter
  - Use integer variable
  - Increment variable in loop
  - Compare variable with number

```
Robot Program  
var_1 = 0  
While var_1 < 5  
  Pick_part  
  Place_part  
  var_1 = var_1 + 1
```

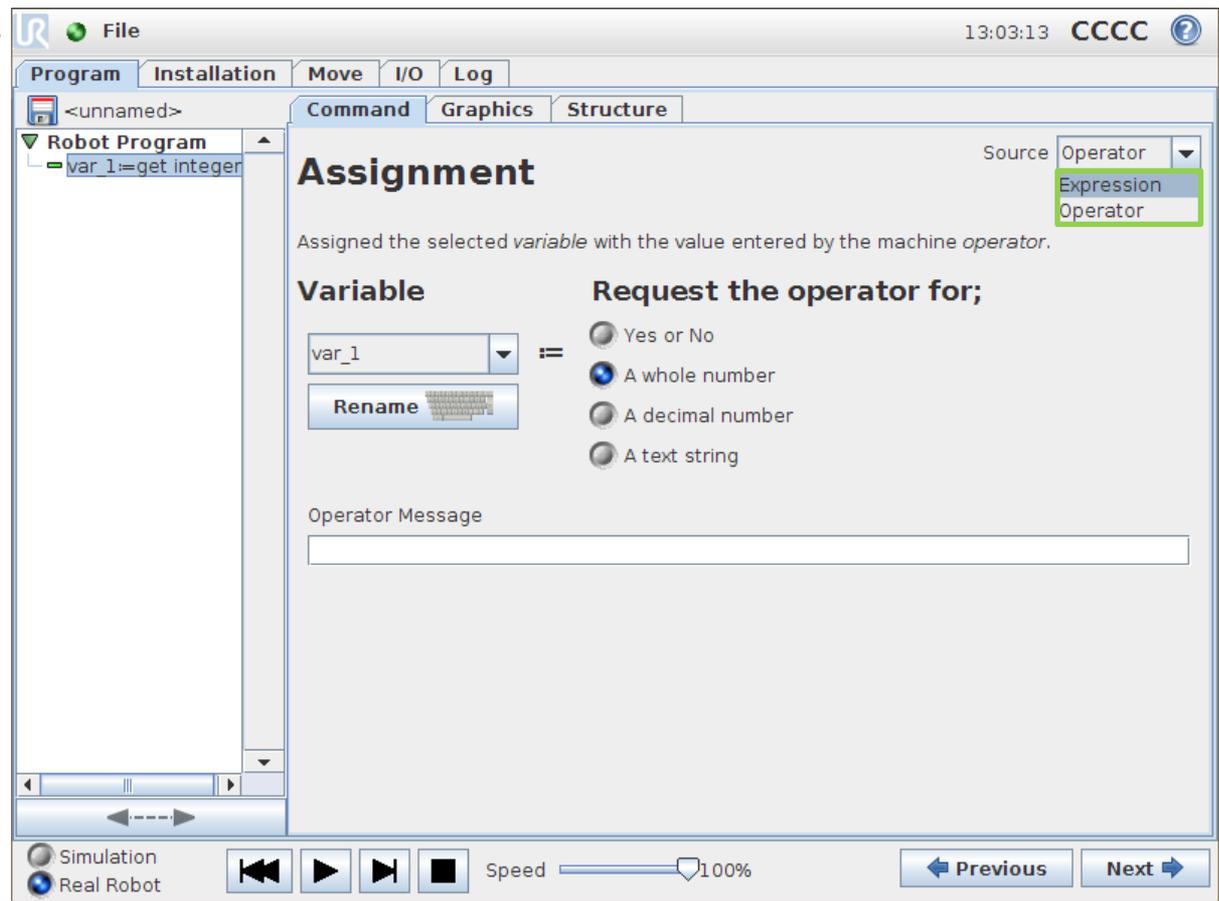


- Save sample program as var\_counter.urp

## Operator input

- Assignment by operator
  - Select Source
  - Define variable type
  - Add message

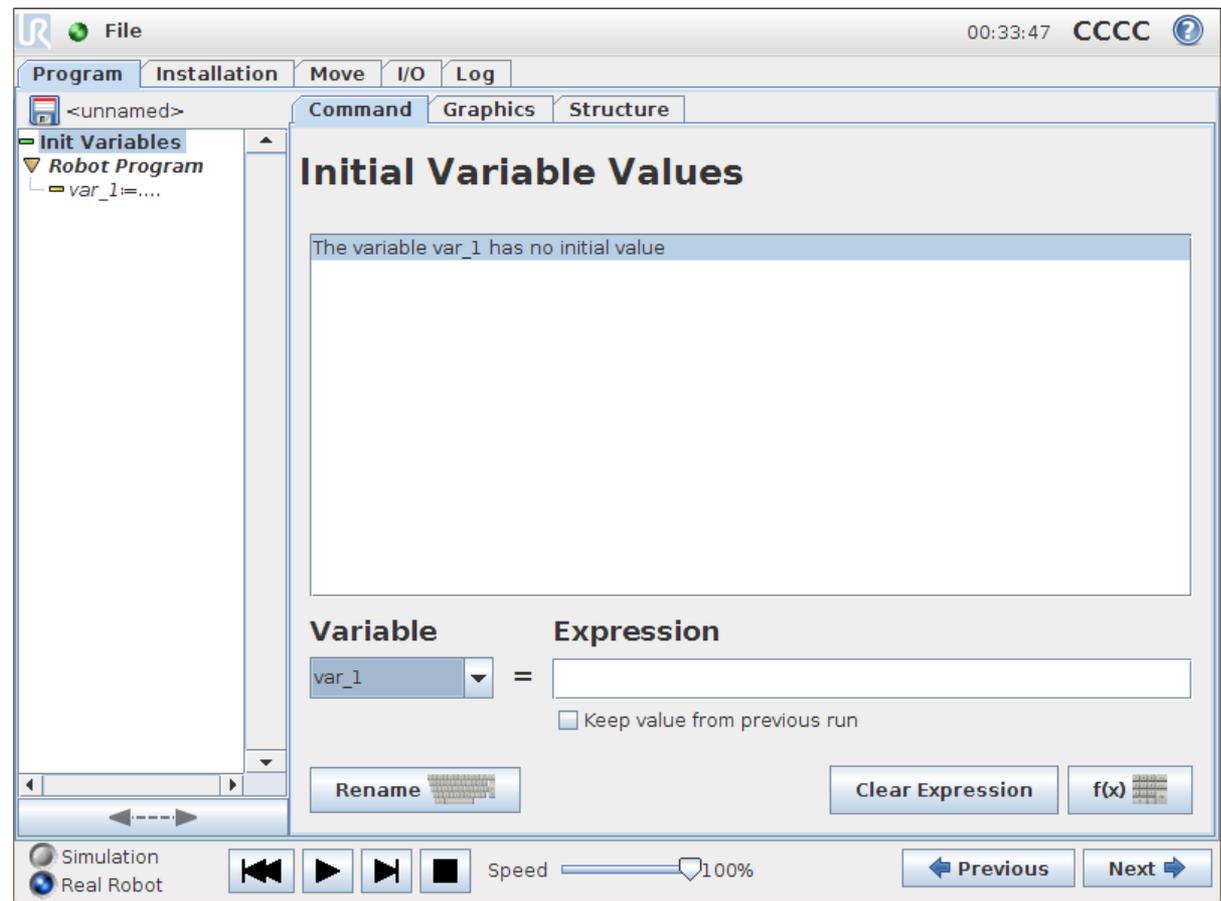
```
Robot Program
var_1 = 0
var_2 = operator input
While var_1 < var_2
  Pick_part
  Place_part
  var_1 = var_1 + 1
```



- Save sample program as var\_operator\_input.urp

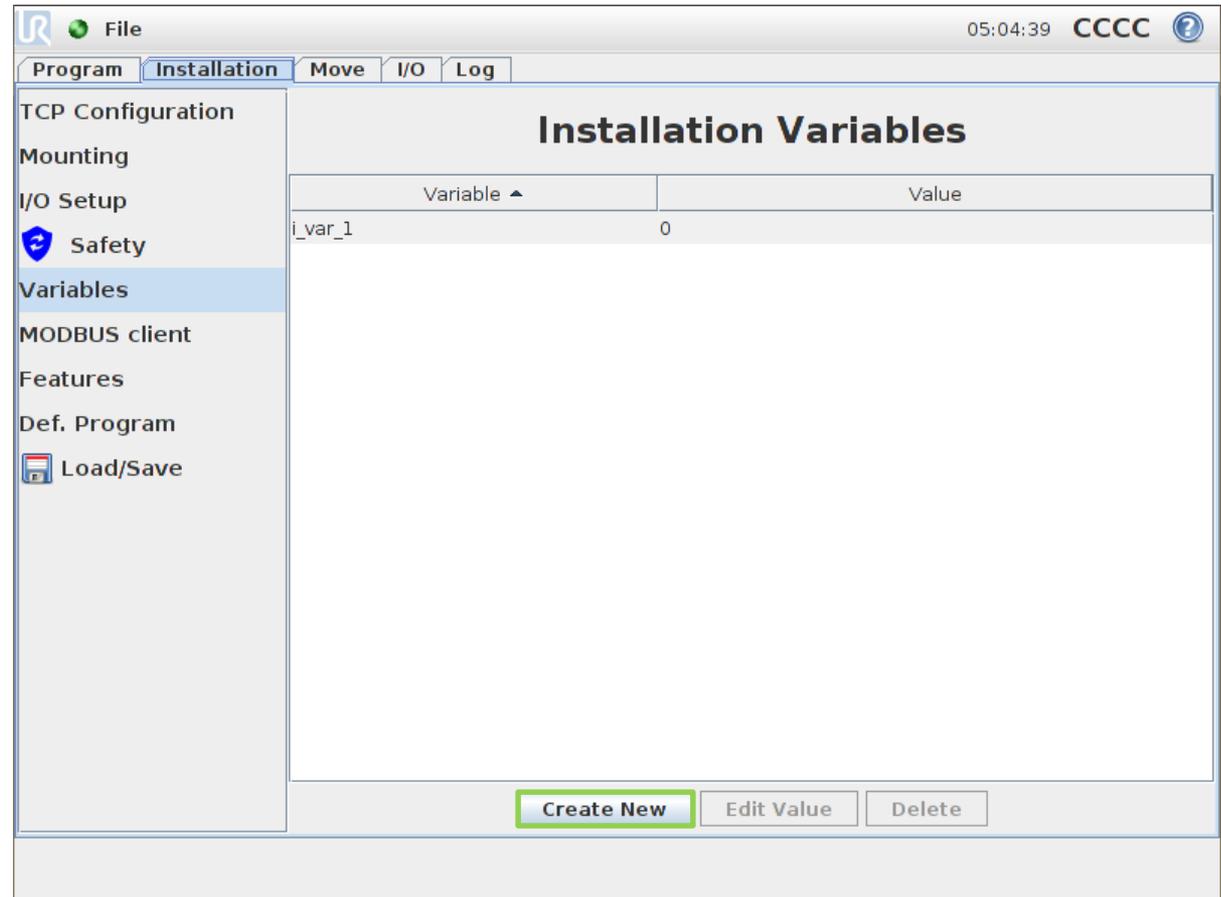
## Init variables

- Initialize variables
  - Local variables listed
  - Preset to fixed value



## Installation variables

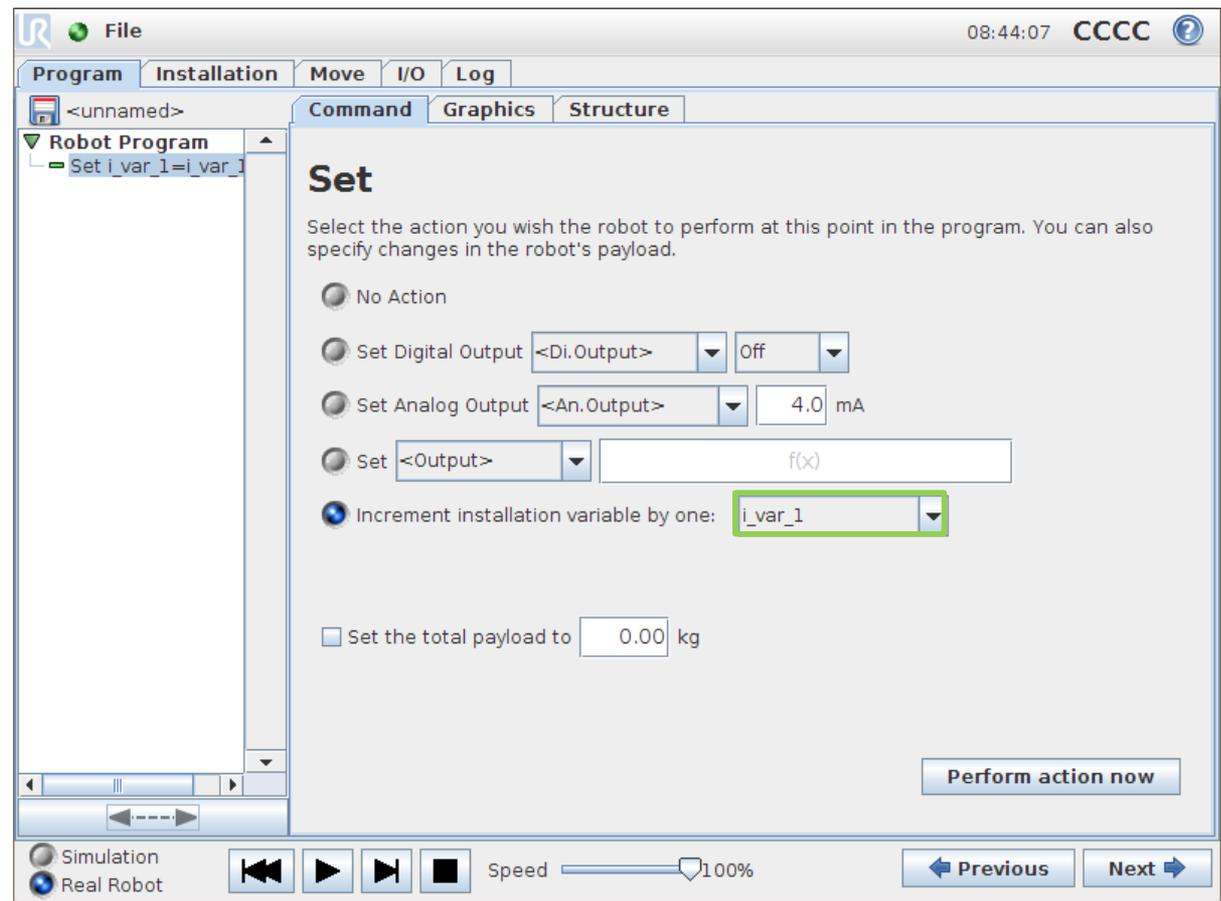
- Features
  - Listed in Installation tab
  - Stored in separate file
  - Keeps value after reboot
- Functionality
  - Global variable
  - Accessible from all programs
  - Same functionality as local variable



## Use installation variable as counter

- Set command
  - Increment installation variable by one

```
Robot Program
i_var_1 = 0
var_2 = operator input
While i_var_1 < var_2
  Pick_part
  Place_part
  Set i_var_1 = i_var_1 + 1
```



- Save sample program as inst\_var\_operator\_input.urp

### Lab exercise

- Create a program using installation variables to count program cycles
  - Create Installation variable called count and set it to 0
  - Create a simple program that moves between waypoint\_1 and waypoint\_2
  - After each move increment count variable.
  - When count reaches 10 clean tool (move to waypoint\_3)
  - When count variable reaches 20, show popup message "Change Feeder Tray".
  - Only continue when the count variable has been manually reset.
- Start/stop program and confirm count variable persists
- Restart robot and confirm count is the same



Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

7

Advanced commands 3

Wizards

Modbus TCP

Service

Safety standards

Adjustable safety

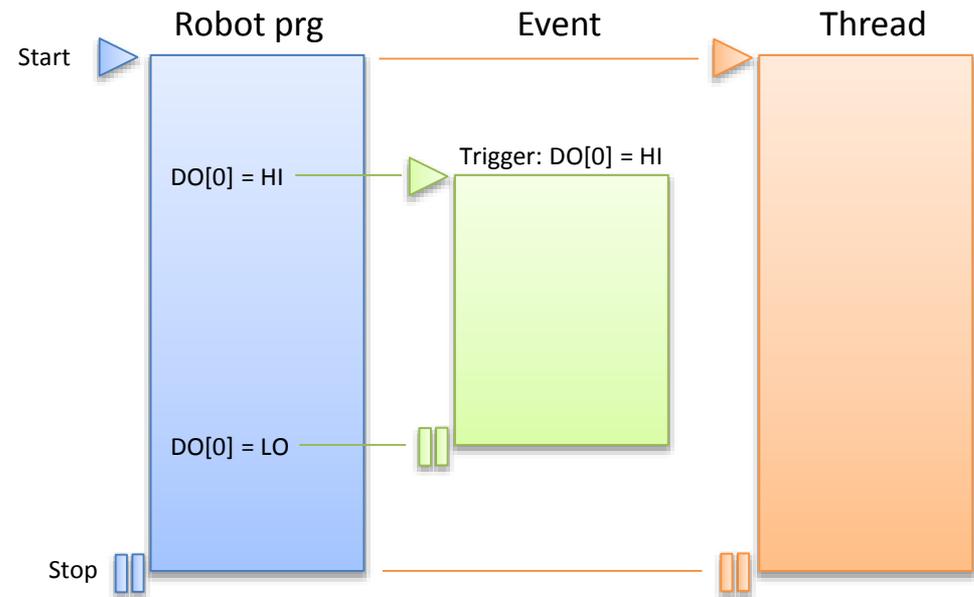
## Threads/events

### ■ Thread

- Parallel process
- Continuously running

### ■ Event

- Parallel process
- Triggered by a condition



### ■ Purpose

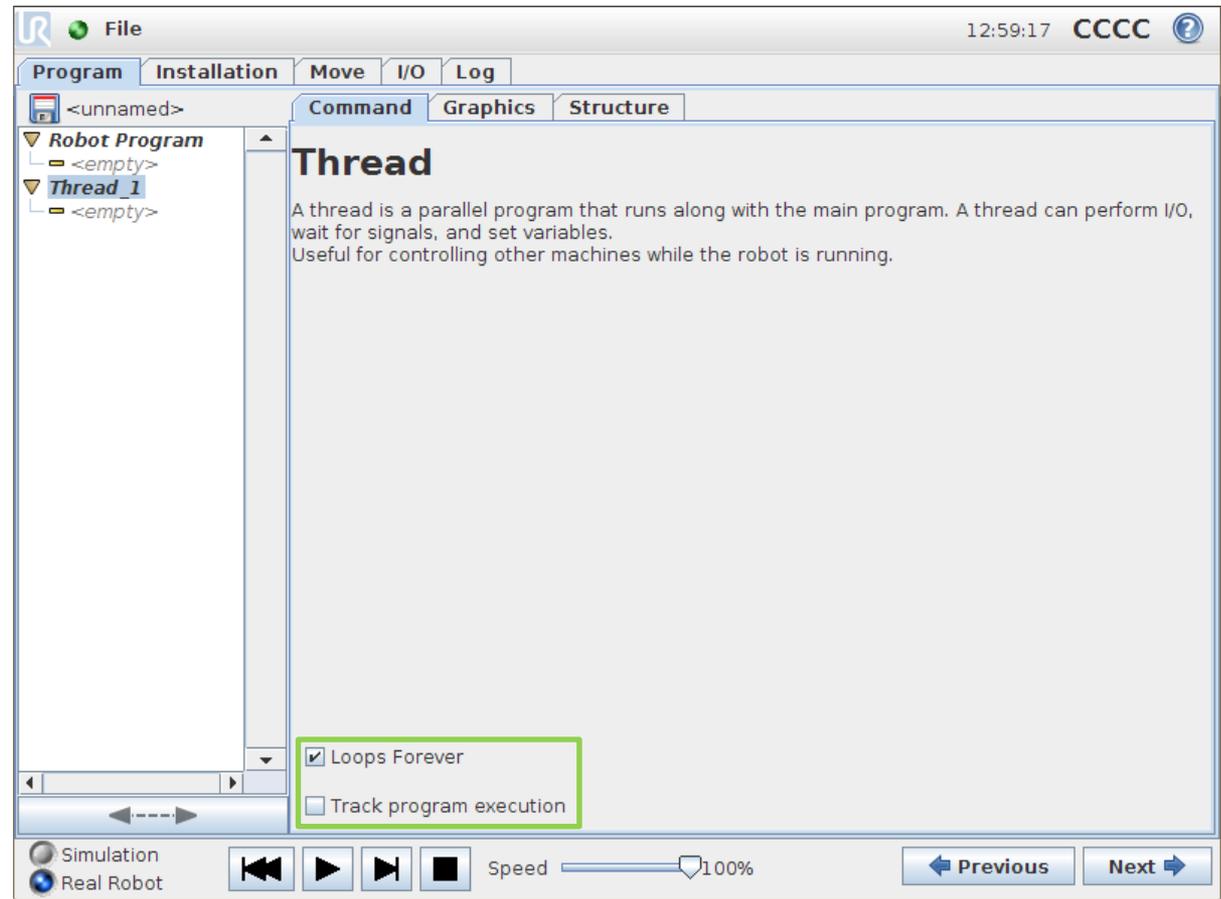
- Useful for controlling I/O communication with other machines, do complex calculations etc.

## Thread

- Settings
  - Loops forever
  - Track program execution

```
Robot Program
MoveJ
  Waypoint_1
  Waypoint_2
IF DI[0] = True
  HALT
```

```
Robot Program
MoveJ
  Waypoint_1
  Waypoint_2
Thread_1
IF DI[0] = True
  HALT
  WAIT 0.01
```

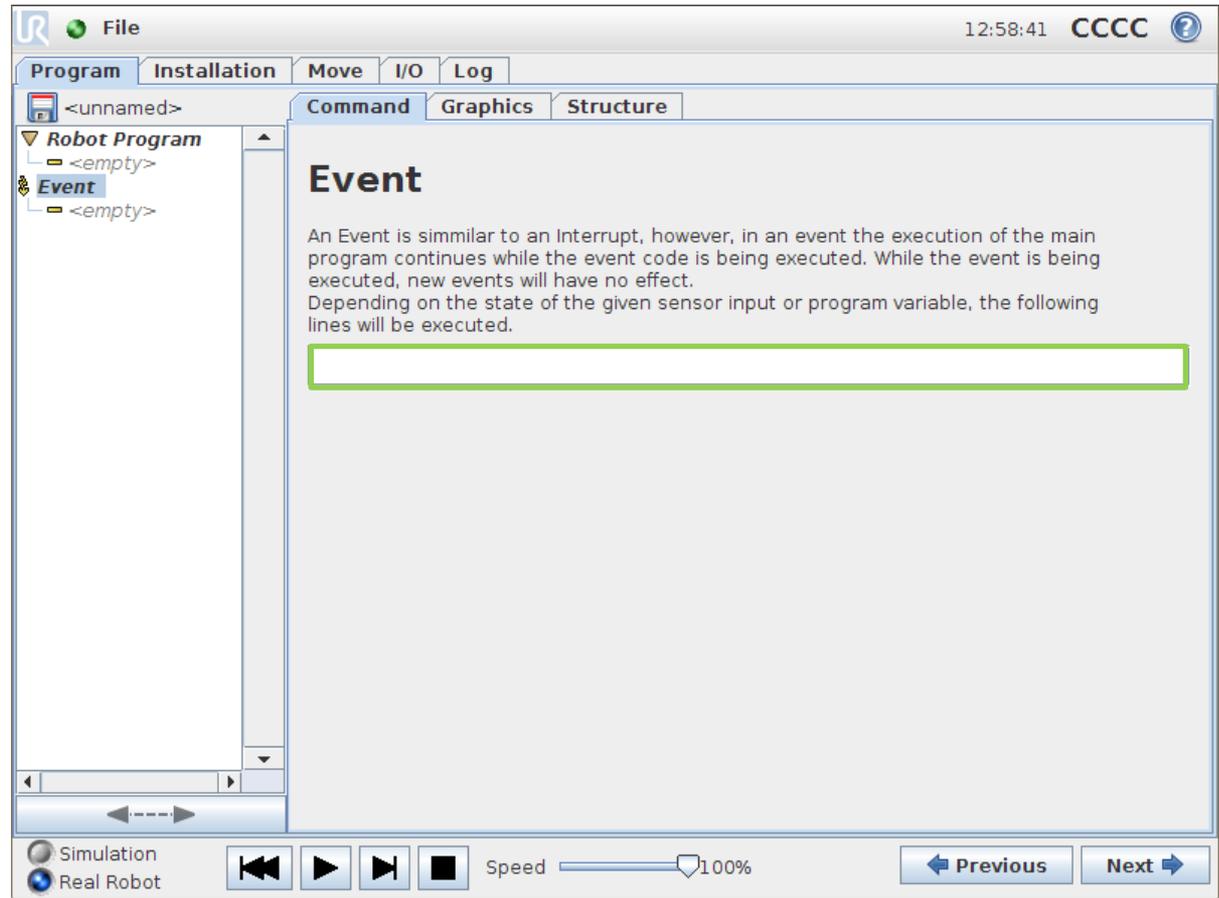


- Save sample program as thread.urp

## Event

- Trigger
  - Define condition to run Event

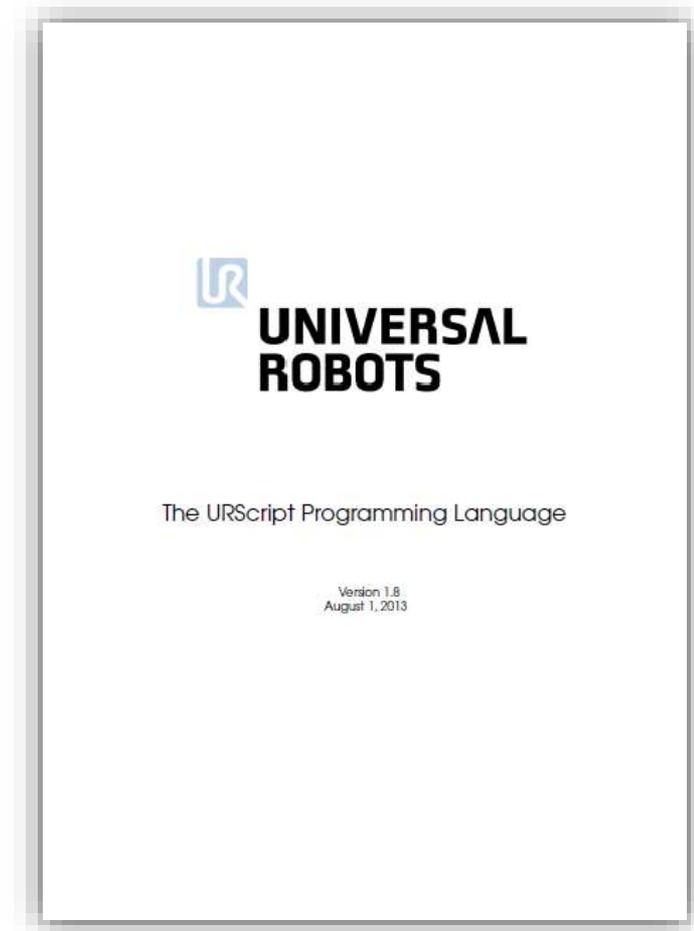
```
Robot Program
MoveJ
  Waypoint_1
  Waypoint_2
Event DI[0] = True
HALT
```



- Save sample program as event.urp

### What is script?

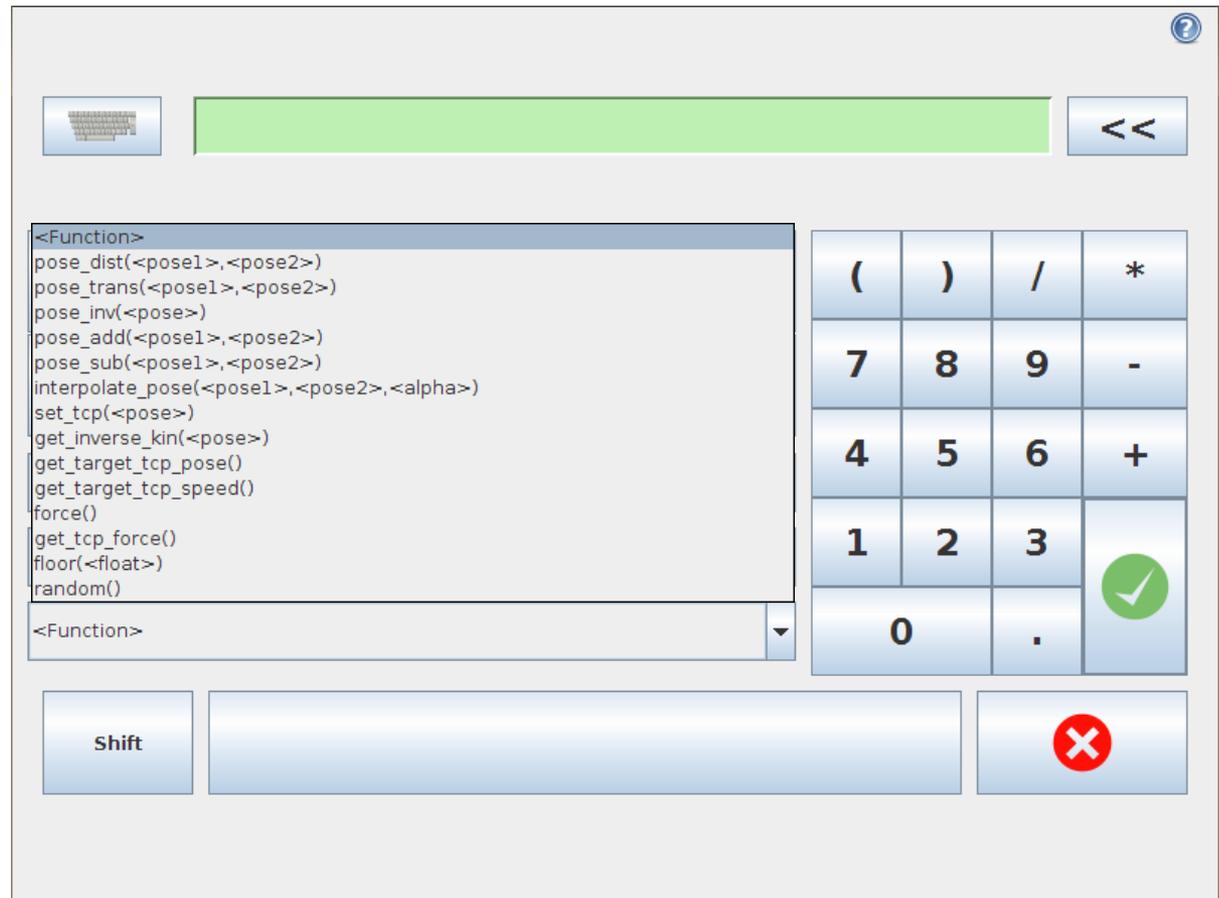
- This is an introduction to URScript
  - Further scripting will be covered in Advanced Training
- URScript
  - High level script language developed by UR
  - Similarities to Python script language
  - Script manual contains definitions of all available script codes



## How to use script

- Expression editor
  - Most common script codes listed
  - Sample program with *force()*
  - Return value: TCP force

```
Robot Program
MoveL
  Waypoint_1
  IF force() < 30
    Waypoint_2
```



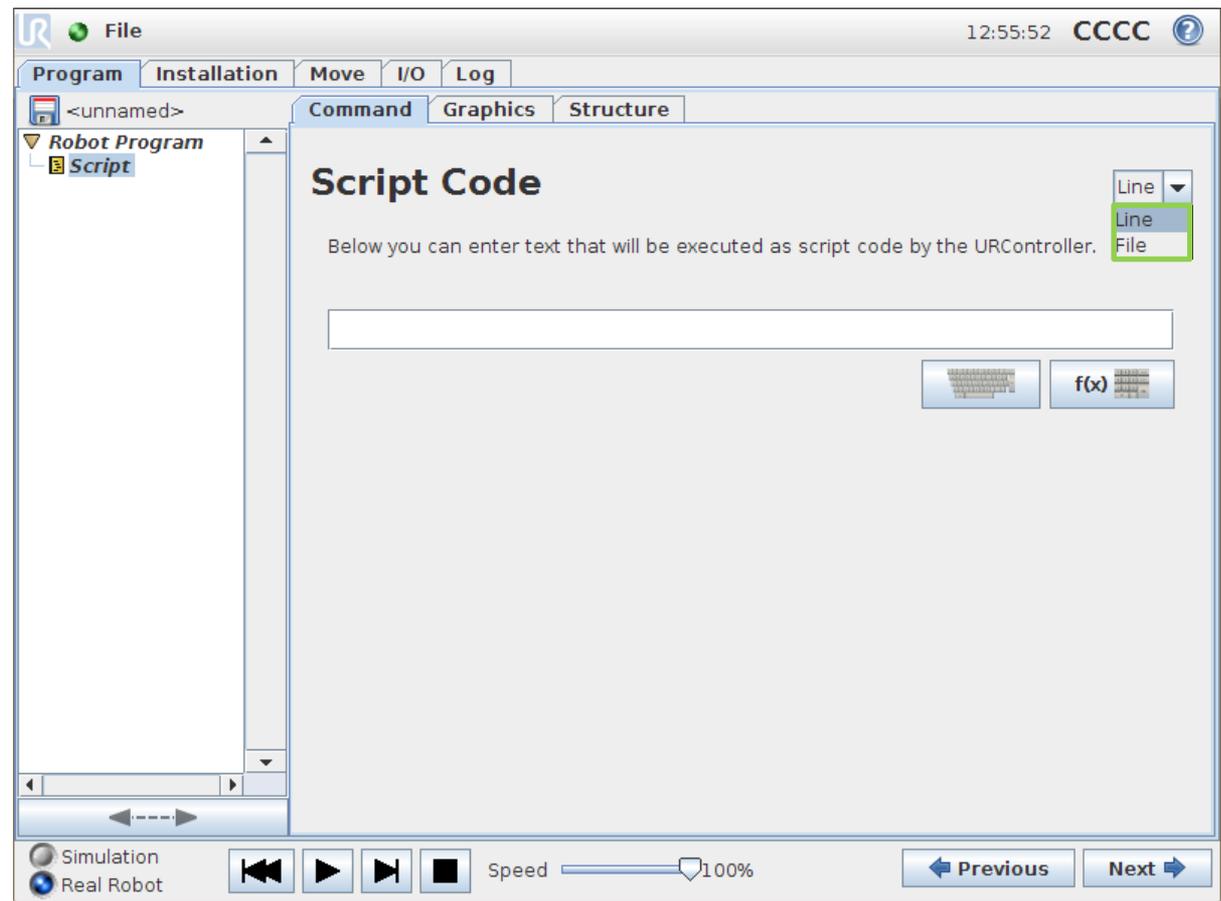
The screenshot displays the script editor interface. At the top, there is a keyboard icon and a green text input field. Below this is a list of functions: <Function>, pose\_dist(<pose1>, <pose2>), pose\_trans(<pose1>, <pose2>), pose\_inv(<pose>), pose\_add(<pose1>, <pose2>), pose\_sub(<pose1>, <pose2>), interpolate\_pose(<pose1>, <pose2>, <alpha>), set\_tcp(<pose>), get\_inverse\_kin(<pose>), get\_target\_tcp\_pose(), get\_target\_tcp\_speed(), force(), get\_tcp\_force(), floor(<float>), and random(). To the right of the function list is a numeric keypad with buttons for (, ), /, \*, 7, 8, 9, -, 4, 5, 6, +, 1, 2, 3, 0, ., and a green checkmark button. At the bottom, there is a Shift button, a large empty text input field, and a red X button.

- Save sample program as force\_feedback.urp

## Script command

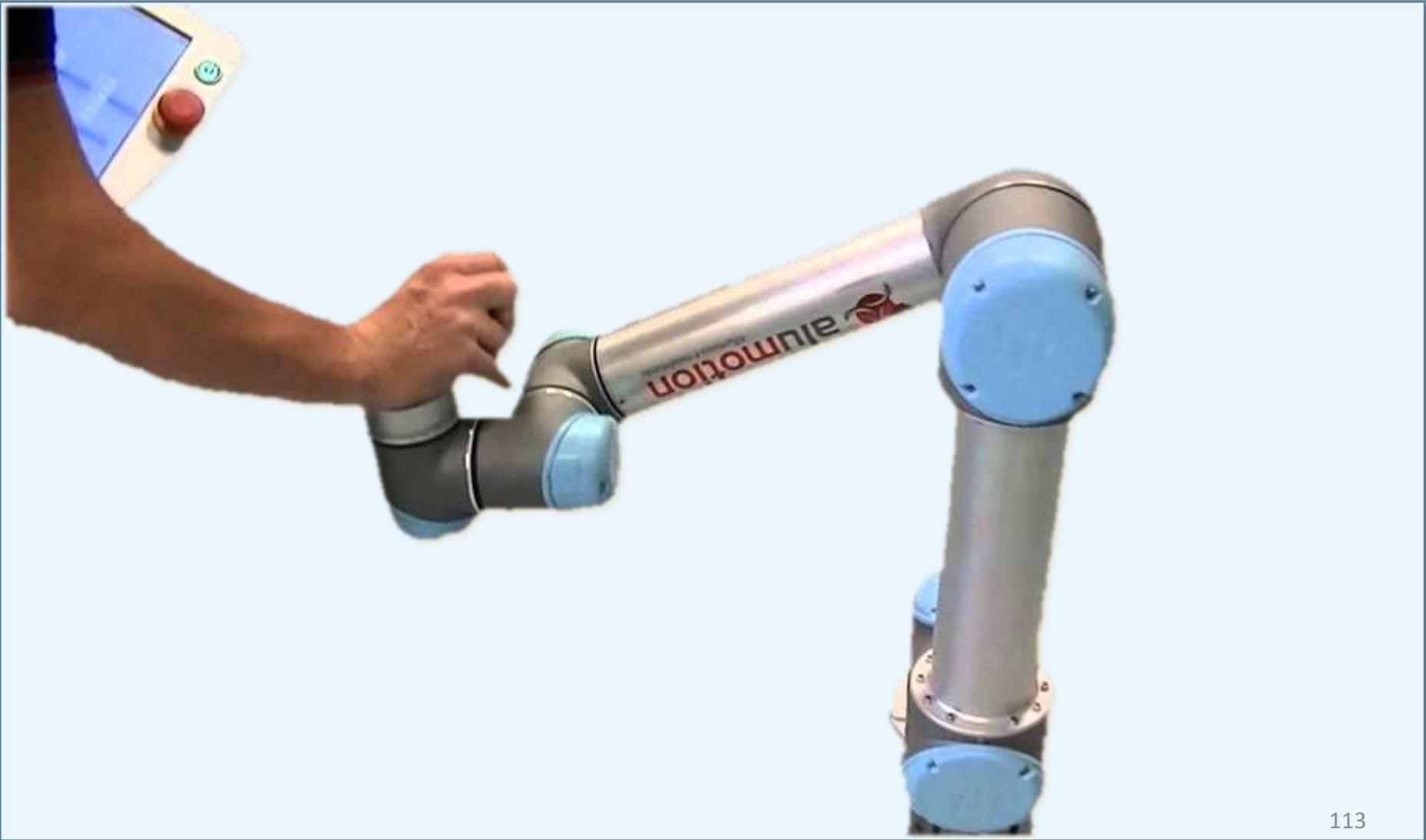
- Line
  - Insert one script command
- File
  - Call file containing multiple script codes

```
Robot Program
set_digital_out(0,True)
WAIT 0.5
set_digital_out(0,False)
WAIT 0.5
```



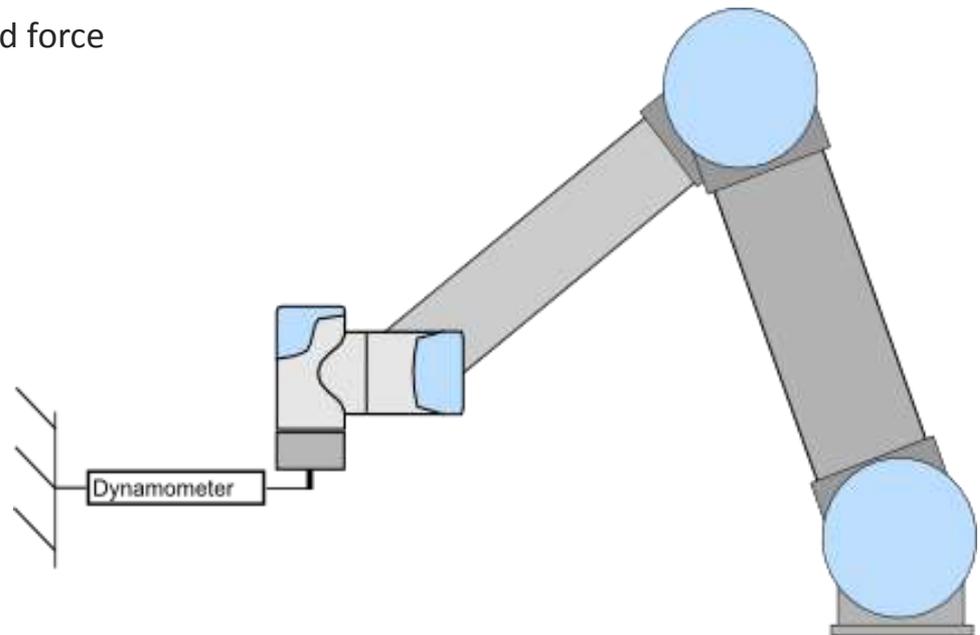
- Save sample program as script\_line.urp

## Force Control



## Force command

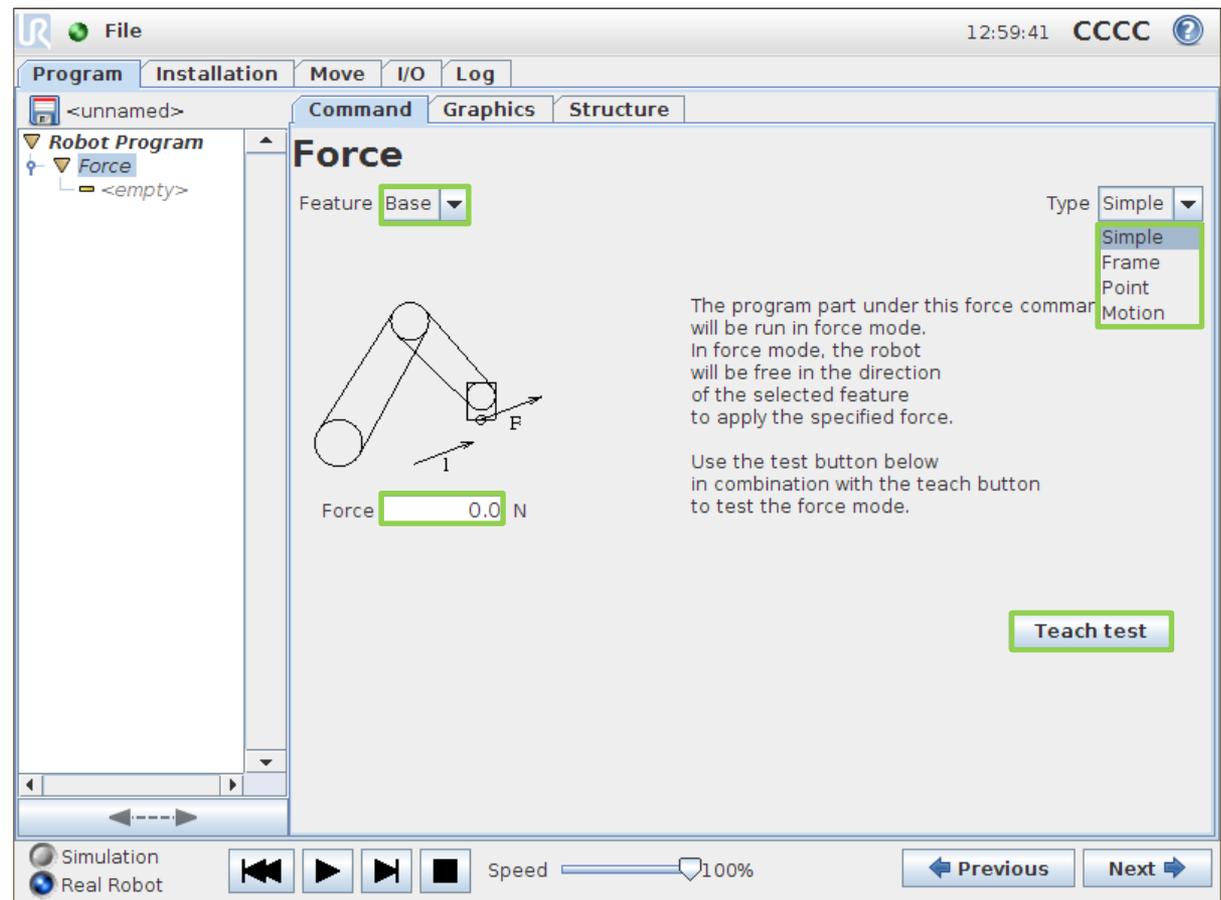
- Features
  - Compliance with environment
  - Adjusting of position to achieve defined force



- Specifications
  - Force precision  $\pm 10 \text{ N}$
  - Torque precision  $\pm 5 \text{ Nm}$
  - Position precision  $\pm 5 \text{ mm}$
  - Orientation precision  $\pm 0.5^\circ$

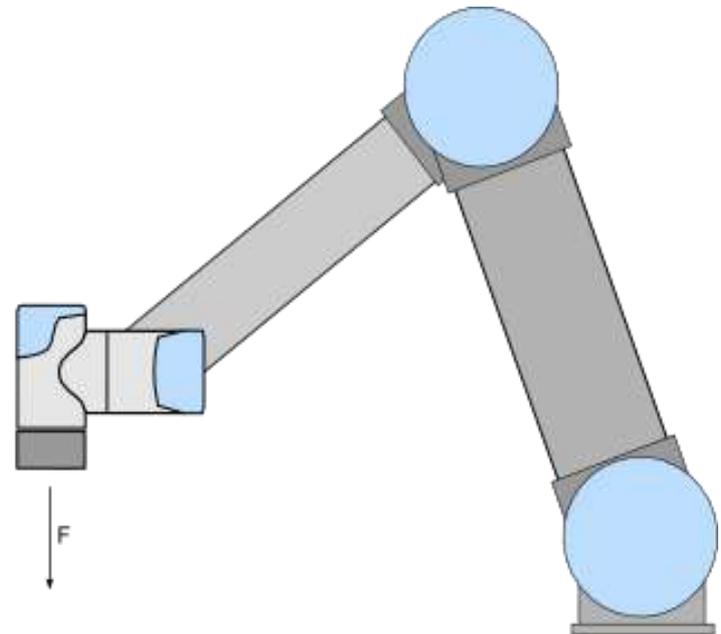
## How to use force

- Settings
  - Force type
  - Force level
  - Direction of force
- Easy testing
  - Teach test



## Force type: Simple

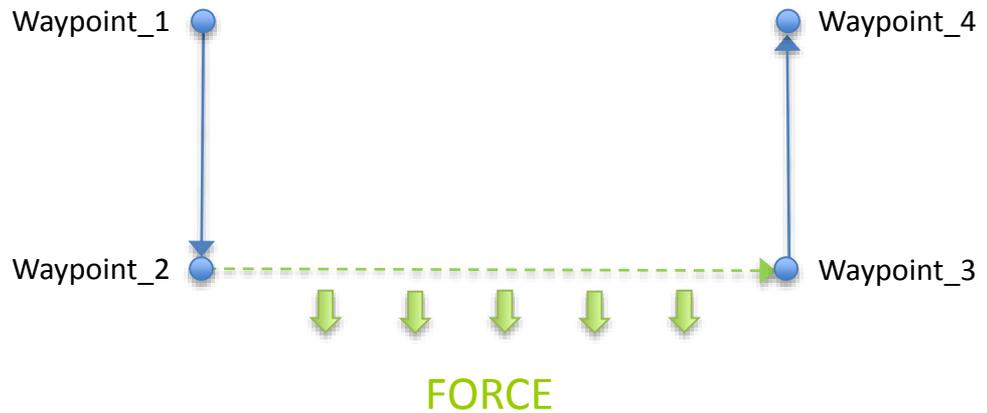
- Features
  - One axis on compliant mode
  - Force direction in Z-axis
  
- Test
  - Add Force to new program
  - Set Type: Simple
  - Set Feature: BASE
  - Set Force: 30 N
    - Teach test
  - Set Force: -30 N
    - Teach test
  - Set Feature: TOOL
    - Teach test



## Force type: Simple

- Sample program

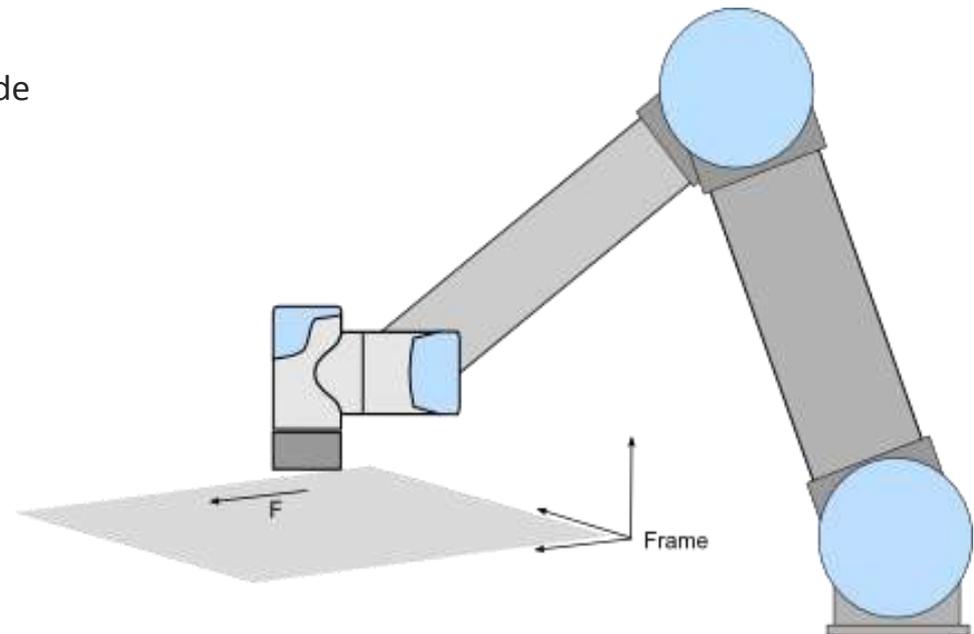
```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2
Force
  Waypoint_2
  Waypoint_3
  Waypoint_4
```



- Save sample program as force\_simple.urp

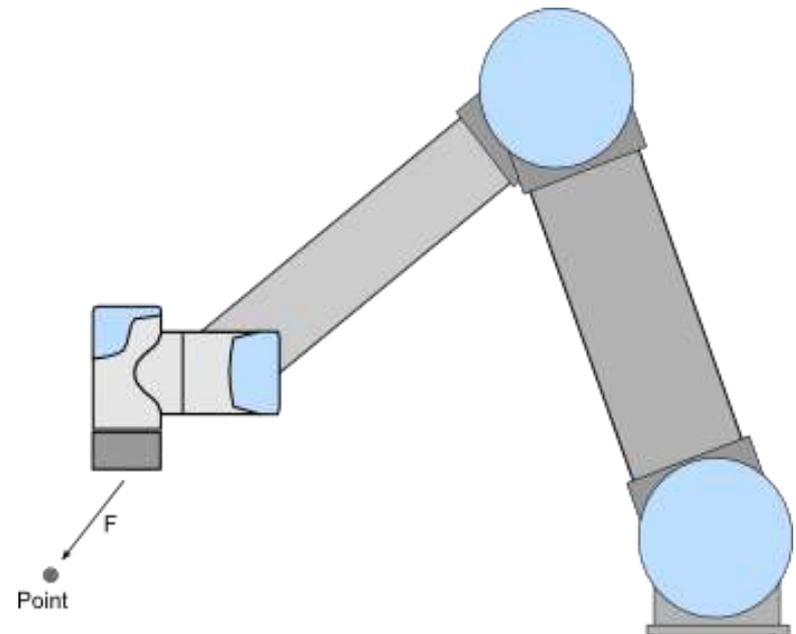
## Force type: Frame

- Features
  - Multiple axes in compliant mode
  - Force level individual for each axis
  - Define speed for axes in compliant mode
  - Base, Tool, user defined frames



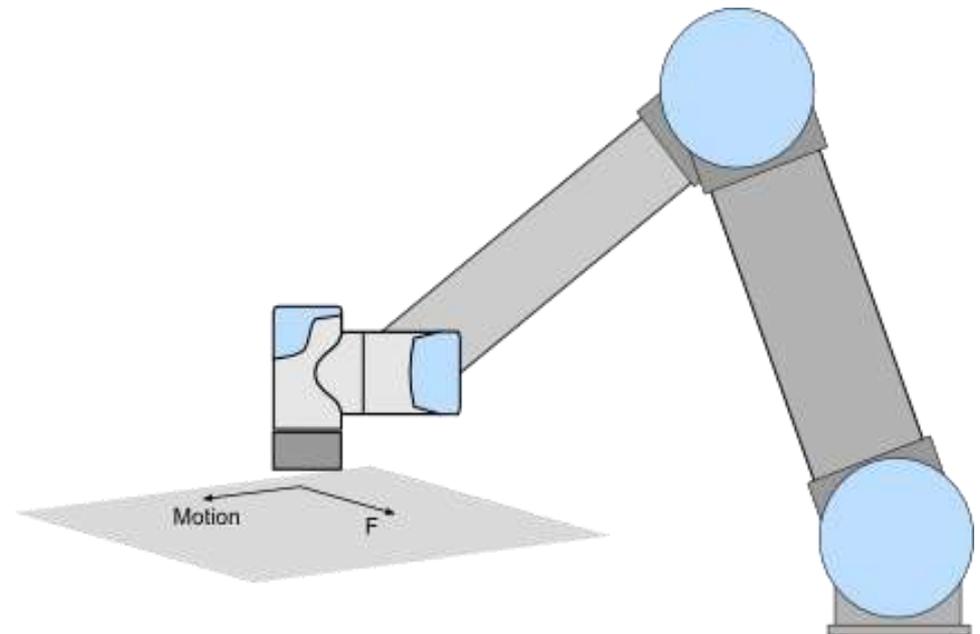
## Force type: Point

- Features
  - Specify Feature point
  - Y-axis of task frame points towards Feature point
  - Task frame changes during runtime



## Force type: Motion

- Features
  - TCP motion in X-axis of task frame
  - Task frame changes during runtime
  - Y-axis perpendicular to TCP motion



- *X-axis not compliant*
- *Teach mode not applicable*

### Lab exercise

- Create a Program using Threads and URScript
- Main Program:
  - Create a simple MoveL between waypoint\_1 and waypoint\_2
- Thread1:
  - Create a thread with an assignment that calls the force() URScript command and saves the result in a variable
  - Insert a 0.01 second wait after the command
  - Set the thread to loop forever
- Run the program and open the variables tab.
- You can now monitor how pushing on the tool plate affects the force value.

Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

8

Wizards

Modbus TCP

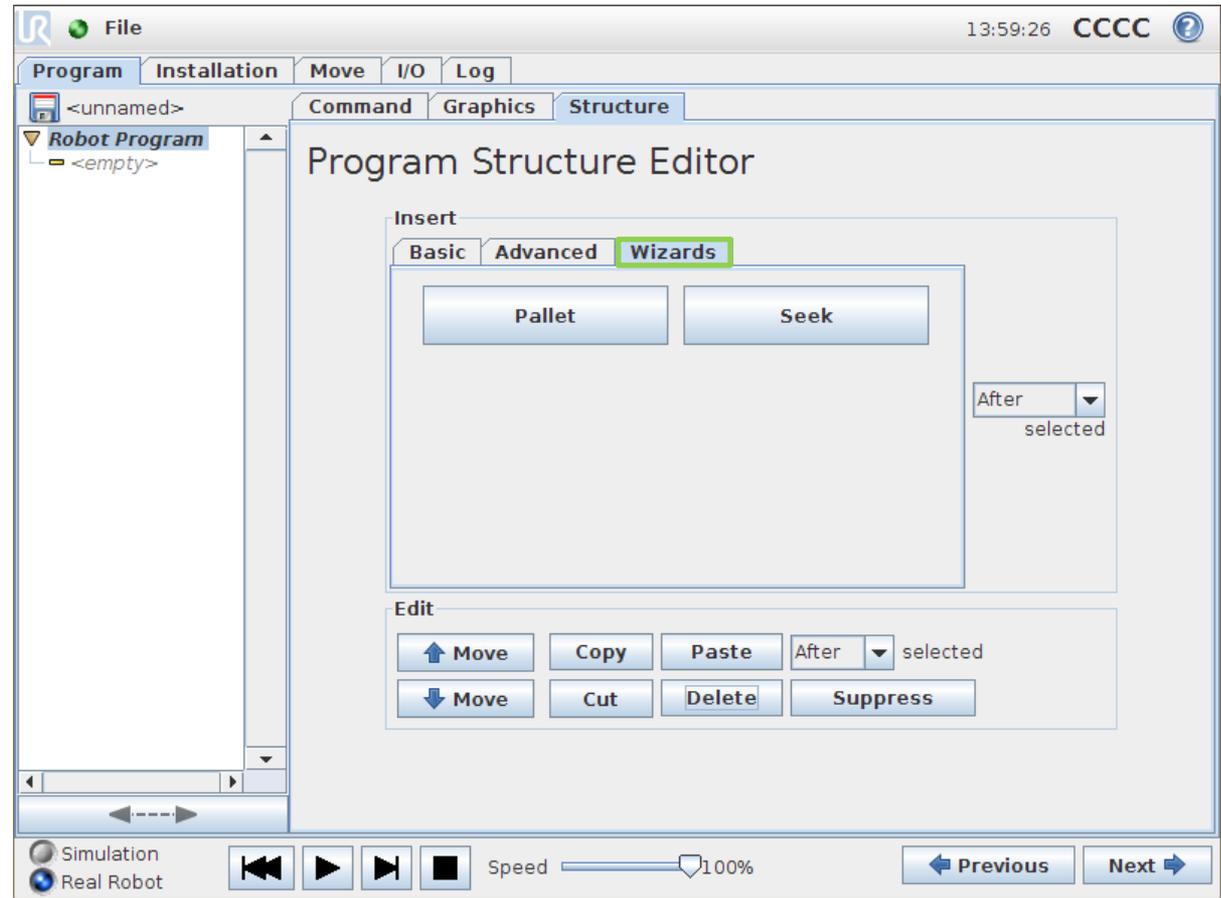
Service

Safety standards

Adjustable safety

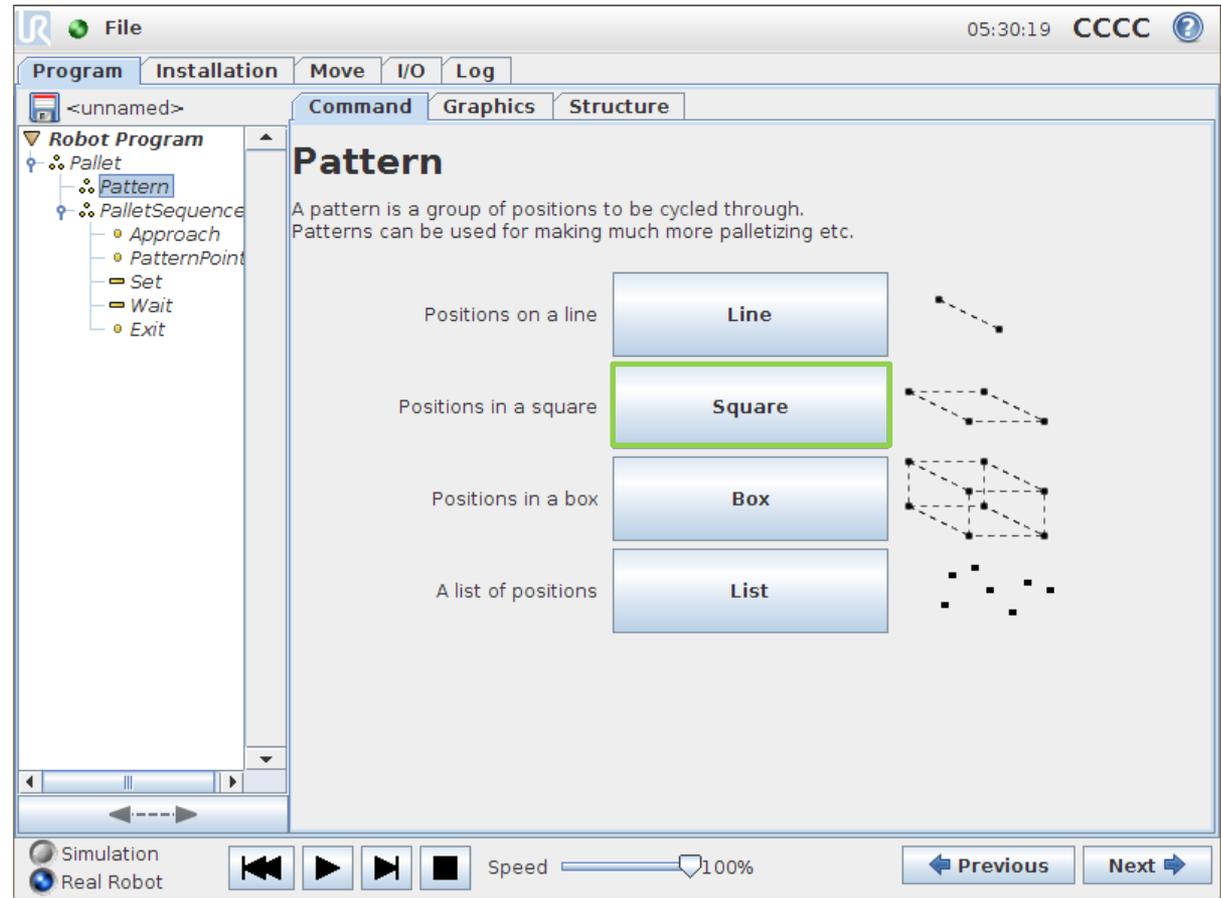
## Wizards

- Pallet
  - Palletizing/depalletizing
  - Patterns
- Seek
  - Search function
- Euromap
  - Injection molding
  - Optional



## Pallet wizard

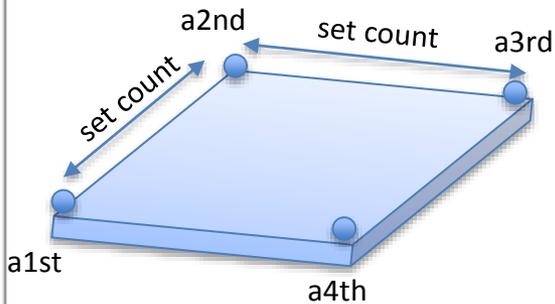
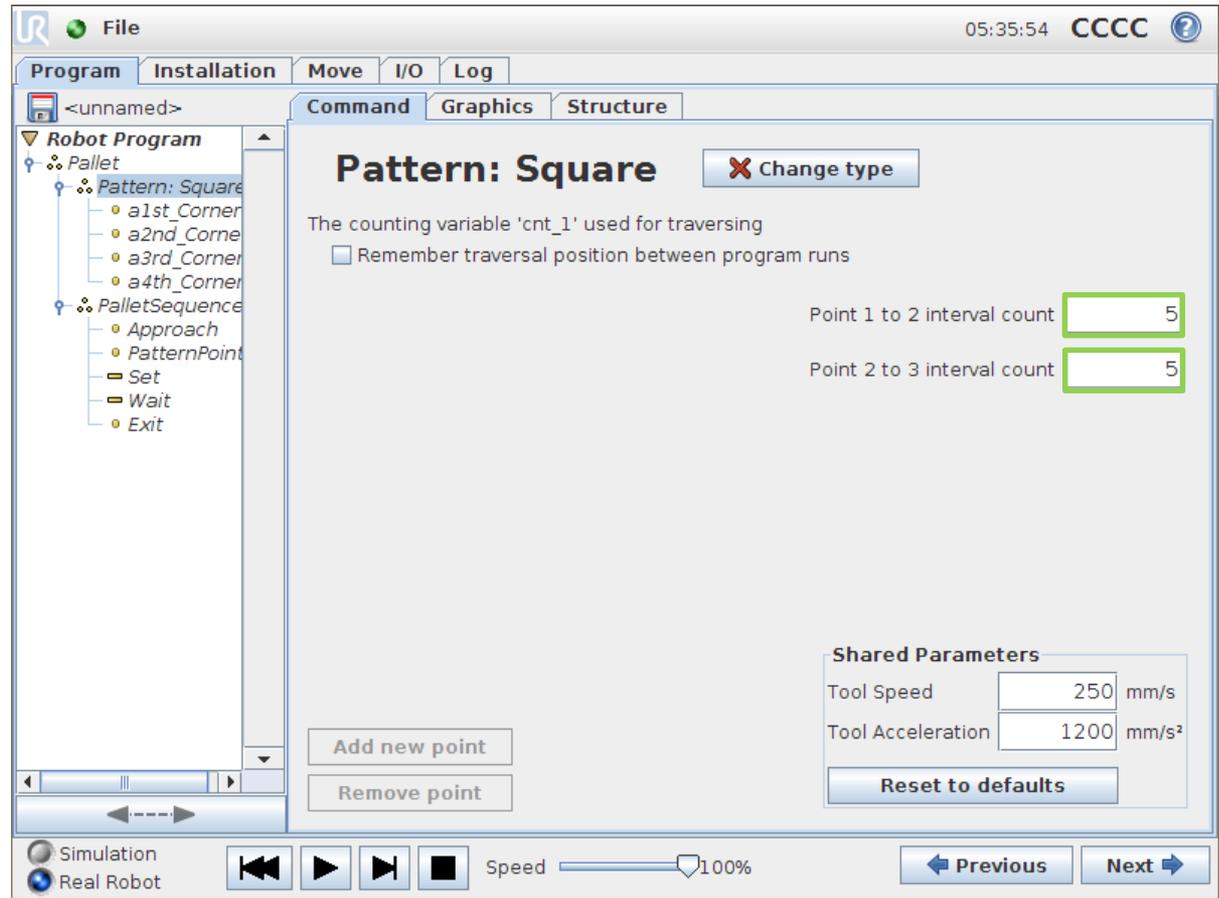
- Pattern
  - Determine palletizing pattern
  - Patterns
    - Line
    - Square
    - Box
    - List
- PalletSequence
  - What to do at each position in pattern



## Pattern: Square

- Pattern
  - Set pattern: Square
  - Set objects between
    - Point 1 to 2
    - Point 2 to 3
  - Teach the 4 corners

### Pattern

**Pattern: Square** ✖ Change type

The counting variable 'cnt\_1' used for traversing

Remember traversal position between program runs

Point 1 to 2 interval count

Point 2 to 3 interval count

**Shared Parameters**

Tool Speed  mm/s

Tool Acceleration  mm/s<sup>2</sup>

Reset to defaults

Add new point Remove point

Simulation  Real Robot  Speed  100%

⏪ Previous Next ⏩

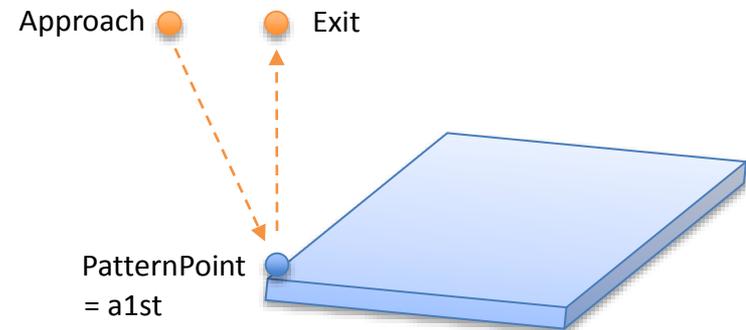
## PalletSequence

- PalletSequence
  - Teach points
    - PatternPoint
    - Approach
    - Exit
  - Define actions
  - Rule of thumb: teach PatternPoint as some points as a1st\_Corner

```

Robot Program
Pallet
  Pattern: Square
  a1st_Corner
  a2nd_Corner
  a3rd_Corner
  a4th_Corner
  PalletSequence
  Approach_1
  PatternPoint_1
  Set DO[0] =True
  Wait 0.5
  Exit_1
    
```

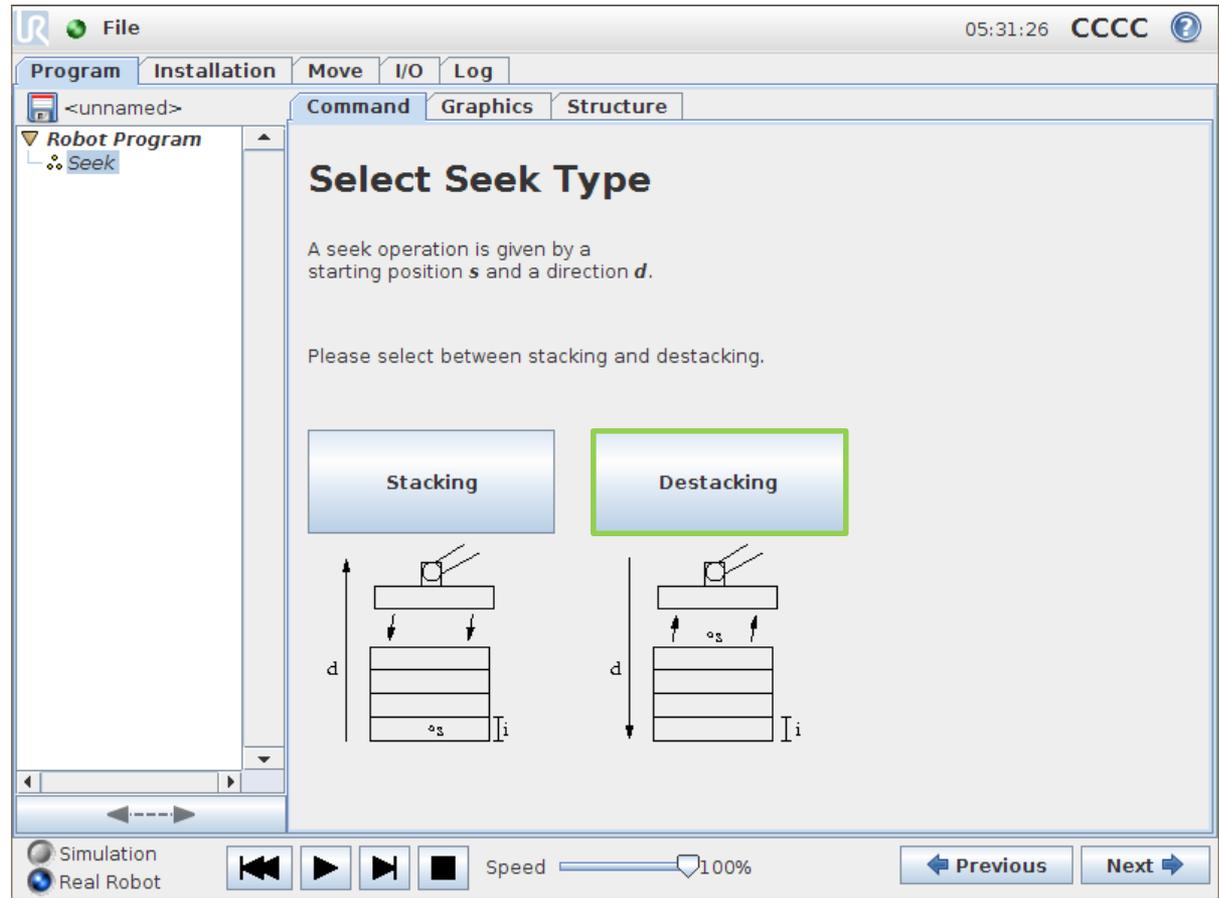
### PalletSequence



- Save sample program as pallet.urp

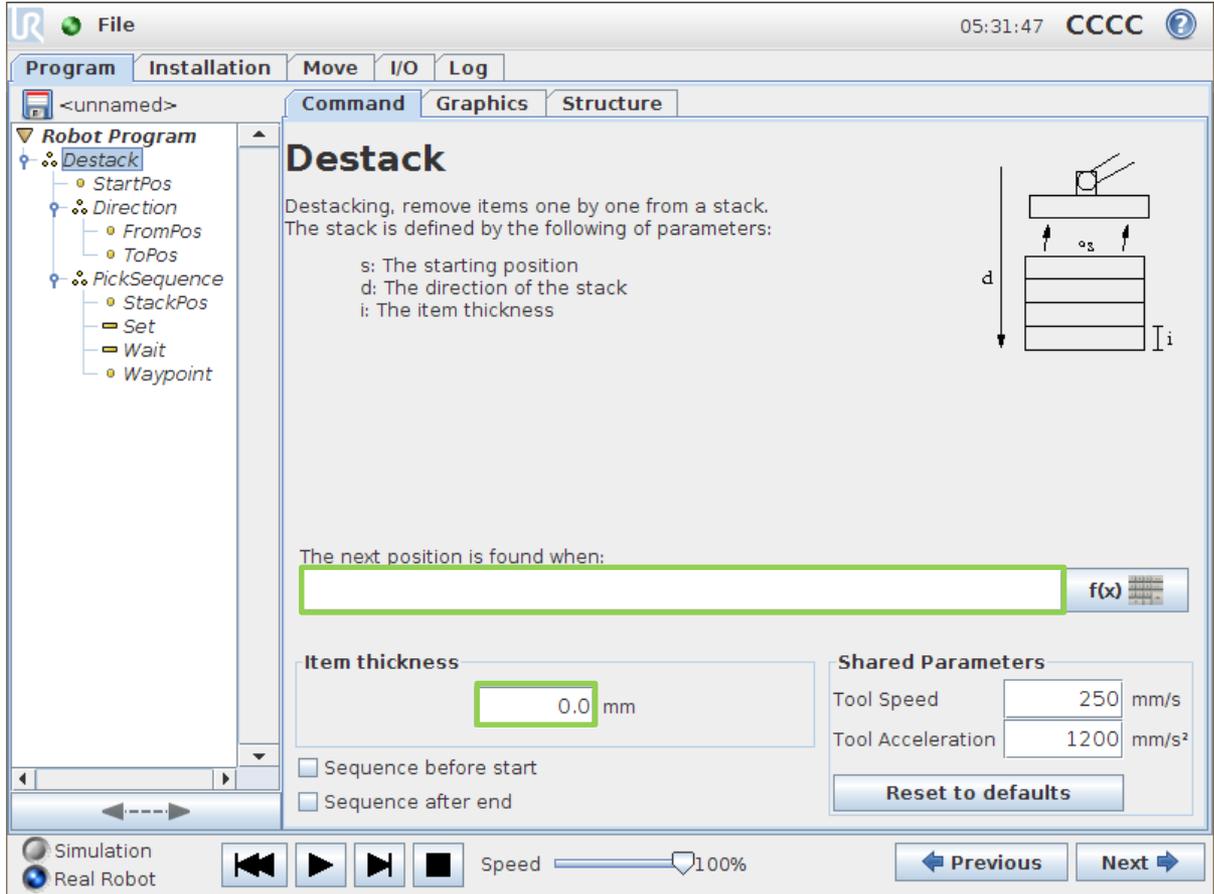
## Seek wizard

- Stacking
  - Add items to a stack
- Destacking
  - Remove items from stack



## Destacking

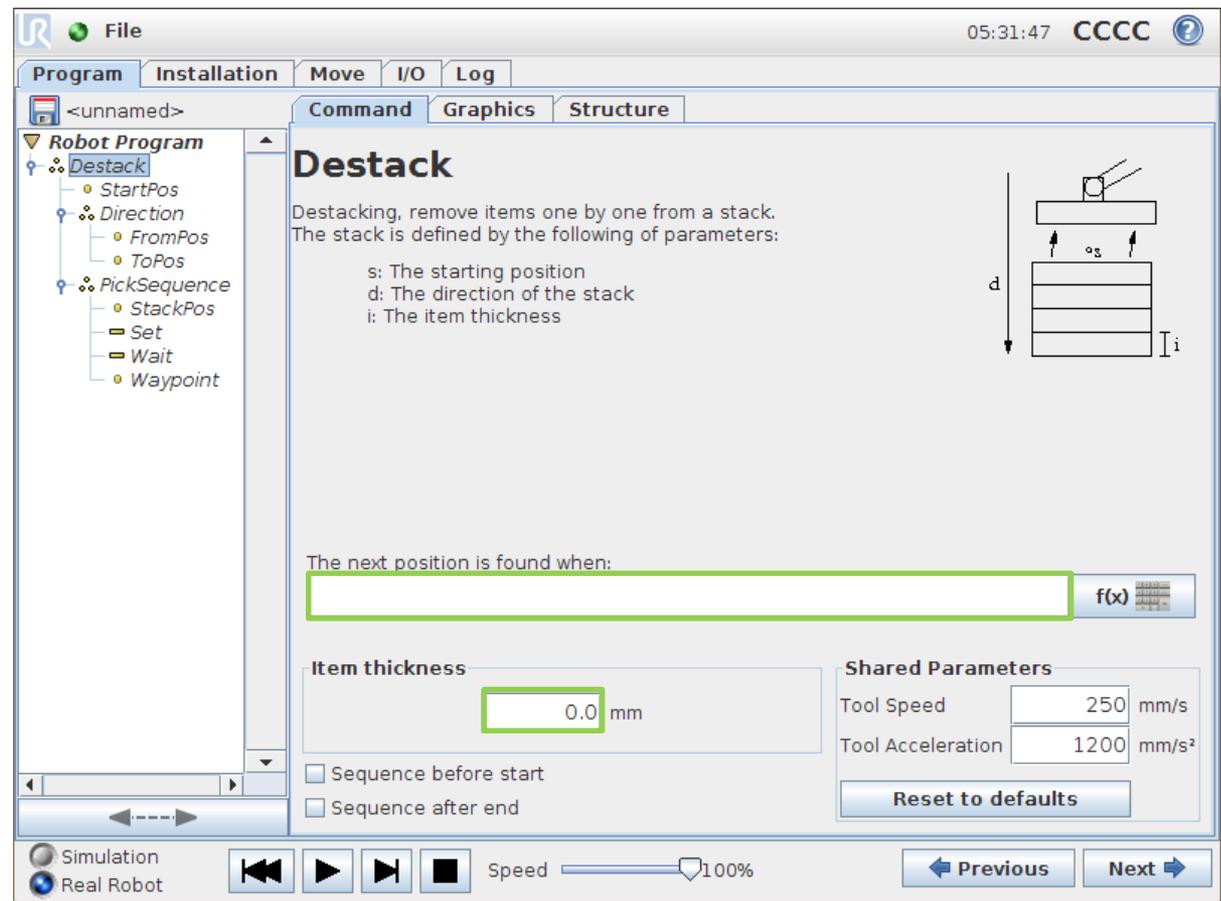
- Sample
  - Add Seek to program
  - Select Destack
  
- Destack
  - Set StartPos
    - From where to start seek operation
  - Set Direction of stack
    - Can be any linear direction
    - FromPos
    - ToPos
  - Set item thickness
    - Same thickness for all items
  - Set sensor input
    - To determine when item is found
    - Use `force() < 30`



The screenshot shows the 'Destack' wizard in the Universal Robots software. The interface includes a menu bar (File, Program, Installation, Move, I/O, Log), a toolbar (Command, Graphics, Structure), and a main workspace. On the left, a tree view shows the 'Robot Program' structure with 'Destack' selected. The main workspace displays the 'Destack' command configuration. A diagram on the right illustrates a stack of items with parameters: 's' for starting position, 'd' for direction, and 'i' for item thickness. The configuration area includes a text box for 'The next position is found when:' with a green border and a 'f(x)' button. Below this, the 'Item thickness' is set to '0.0 mm'. The 'Shared Parameters' section shows 'Tool Speed' at 250 mm/s and 'Tool Acceleration' at 1200 mm/s². There are checkboxes for 'Sequence before start' and 'Sequence after end', and a 'Reset to defaults' button. The bottom status bar shows 'Simulation' and 'Real Robot' modes, navigation buttons, and a speed slider set to 100%.

## Destacking

- PickSequence
  - StackPos
    - Defines where to pick item when detected
  - Waypoint
    - Defines how to exit after picking item
  - Define actions
    - Set
    - Wait



The screenshot shows the 'Destack' wizard in the Universal Robots software. The interface includes a menu bar (File, Program, Installation, Move, I/O, Log), a toolbar, and a main workspace. On the left, a tree view shows the 'Robot Program' structure with 'Destack' selected. The main workspace displays the 'Destack' wizard with the following elements:

- Command:** Destack
- Description:** Destacking, remove items one by one from a stack. The stack is defined by the following of parameters:
  - s: The starting position
  - d: The direction of the stack
  - i: The item thickness
- Diagram:** A schematic showing a stack of four rectangular items. A vertical arrow labeled 'd' indicates the direction of the stack. A horizontal arrow labeled 's' indicates the starting position. A vertical arrow labeled 'i' indicates the item thickness.
- Configuration:**
  - The next position is found when:** A text input field with a green border and a function button 'f(x)' to its right.
  - Item thickness:** A text input field with a green border containing '0.0 mm'.
  - Shared Parameters:**
    - Tool Speed: 250 mm/s
    - Tool Acceleration: 1200 mm/s<sup>2</sup>
    - Buttons: 'Reset to defaults'
  - Options:**
    - Sequence before start
    - Sequence after end
- Simulation/Real Robot:** Radio buttons at the bottom left.
- Speed:** A slider at the bottom center set to 100%.
- Navigation:** 'Previous' and 'Next' buttons at the bottom right.

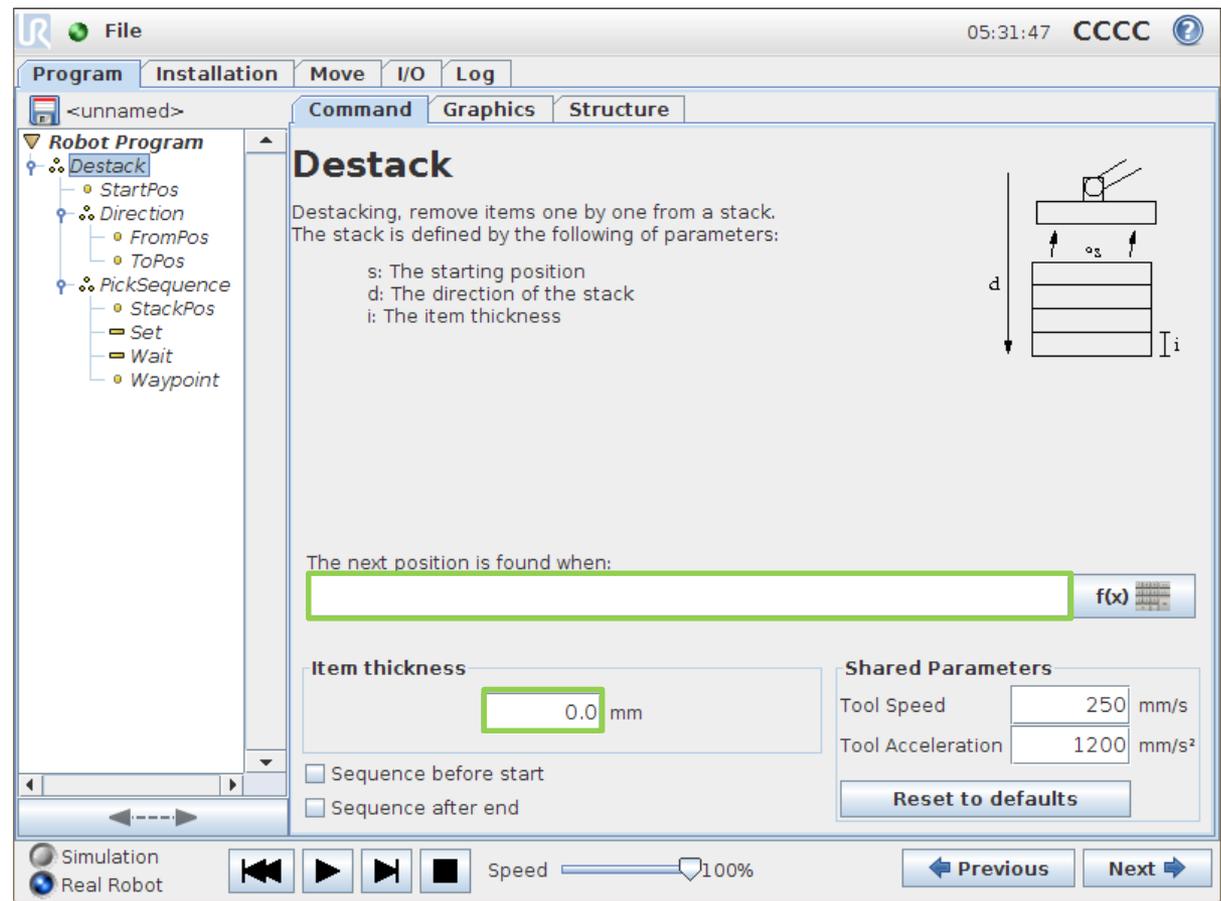
- rule of thumb is to teach StackPos as same point as StartPos

## Destacking

- Sample program
  - Add folder for placing destacked item

```

Robot Program
  Destack
    StartPos
    Direction
      FromPos
      ToPos
    PickSequence
      StackPos
      Set DO[0] = True
      Wait 0.5
      Waypoint_1
  Folder
    Waypoint_2
    Waypoint_3
    Set DO[0] = False
    Wait 0.5
    Waypoint_2
    
```



The screenshot shows the 'Destack' wizard in the Universal Robots software. The interface includes a menu bar (File, Program, Installation, Move, I/O, Log), a toolbar (Command, Graphics, Structure), and a main workspace. On the left, a tree view shows the 'Robot Program' structure with 'Destack' selected. The main workspace displays the 'Destack' command configuration. A diagram on the right illustrates a stack of items with parameters: 's' for starting position, 'd' for direction, and 'i' for item thickness. The configuration area includes a text box for 'The next position is found when:' with a green border, an 'Item thickness' input field set to '0.0 mm', and 'Shared Parameters' for Tool Speed (250 mm/s) and Tool Acceleration (1200 mm/s²). There are also checkboxes for 'Sequence before start' and 'Sequence after end', and a 'Reset to defaults' button. The bottom status bar shows 'Simulation' and 'Real Robot' modes, navigation buttons, and a speed slider set to 100%.

- Save sample program as seek\_destack.urp

## Lab exercise

- Use the Palletizing wizard to create a program that simulates placing a part on each of the locations on the diagram below. You will be provided with an A3 print out of this diagram to use in the demo.
- Palletizing program
  - Select the palletizing wizard template and create a new program
  - Define the number of positions in each dimension
  - Teach the 4 corner points of the Pallet
  - Set approach and exit waypoints 100mm directly above the PatternPoint
  - Set a feeder waypoint to collect a part from before moving to each position on the Pallet.

X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

wizards

**9** Modbus TCP

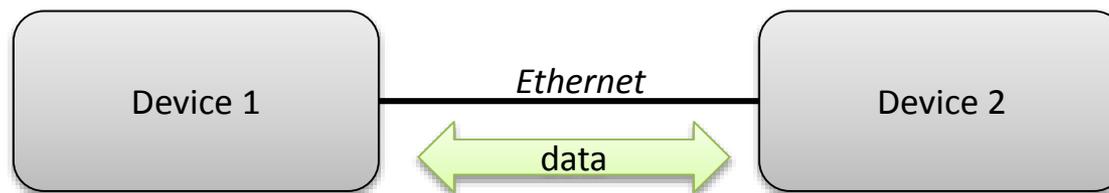
Service

Safety standards

Adjustable safety

## What is Modbus TCP?

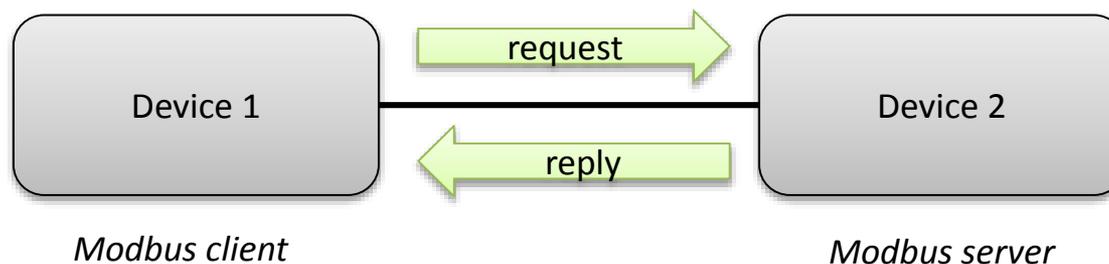
- Modbus TCP
  - Ethernet based communication protocol
- Communication protocol
  - A protocol is a common language with which two devices can communicate
  - Possible to transfer data between devices



- Client / Server relationship

## Client / Server

- Server
  - One device acts as Server
  - Listening on requests from Client
- Client
  - Other device(s) acts as Client
  - Sends requests to Server



- Each device must have a unique IP-address

## Data types

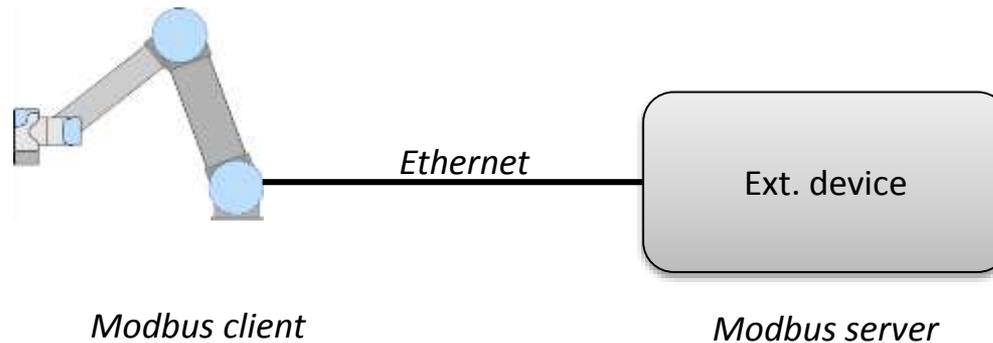
- Available data types for Modbus TCP

data type	value	address range
Digital inputs	On/Off	<i>Consult documentation provided by vendor of Server device</i>
Digital outputs	On/Off	
Register inputs	16 bit	
Register outputs	16 bit	

- Address range
  - Each digital signal and register have a unique address
  - Address is *always* specified in documentation provided by vendor

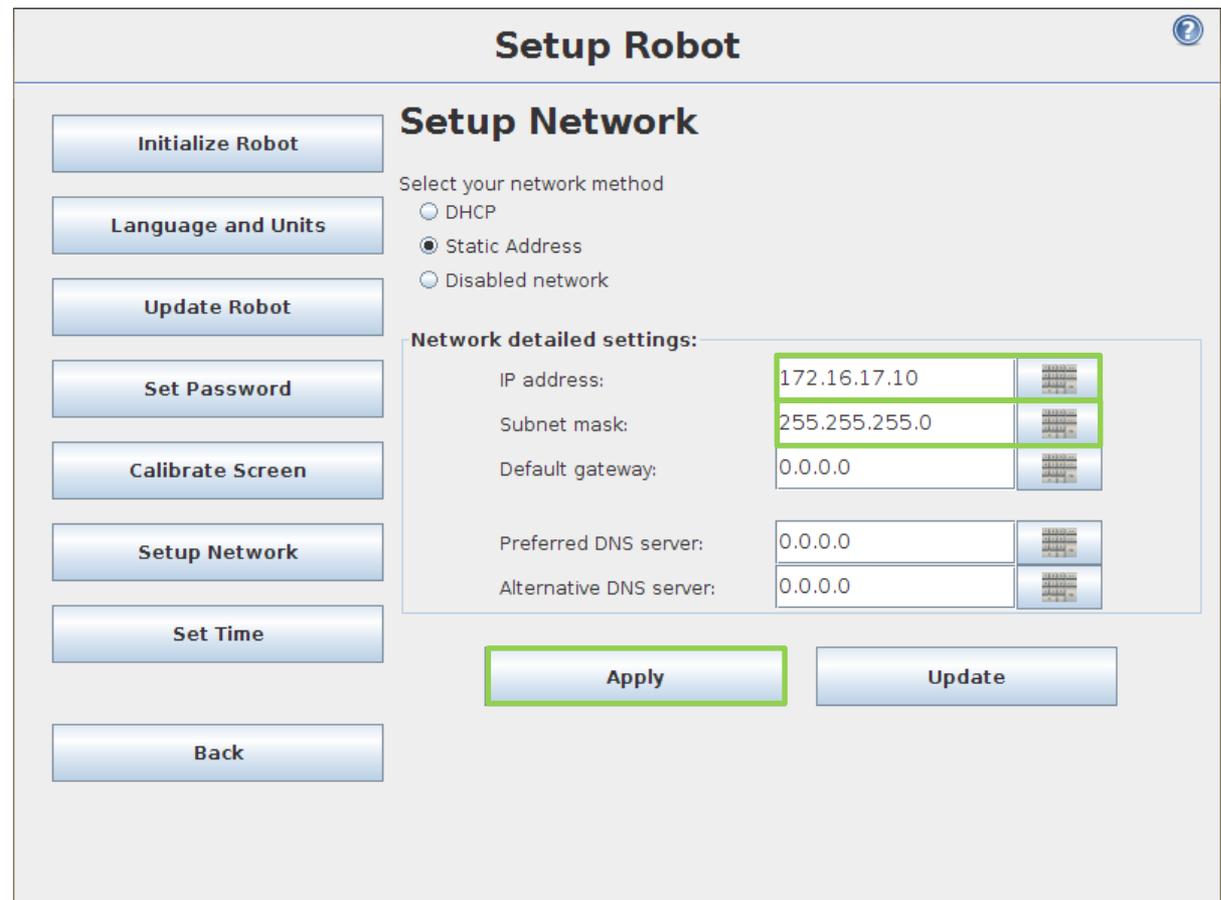
## Modbus TCP

- Sample
  - Use robot as Client and connect to a Server



## Setup network

- Settings
  - Select Static Address
  - Set IP-address of robot
  - Set Subnet mask
  - Apply to save configuration



**Setup Robot** ?

**Setup Network**

Select your network method

DHCP

Static Address

Disabled network

**Network detailed settings:**

IP address:	172.16.17.10	
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**

**Back**

- Tip: Use UPDATE button for "pinging" the other device

## Server

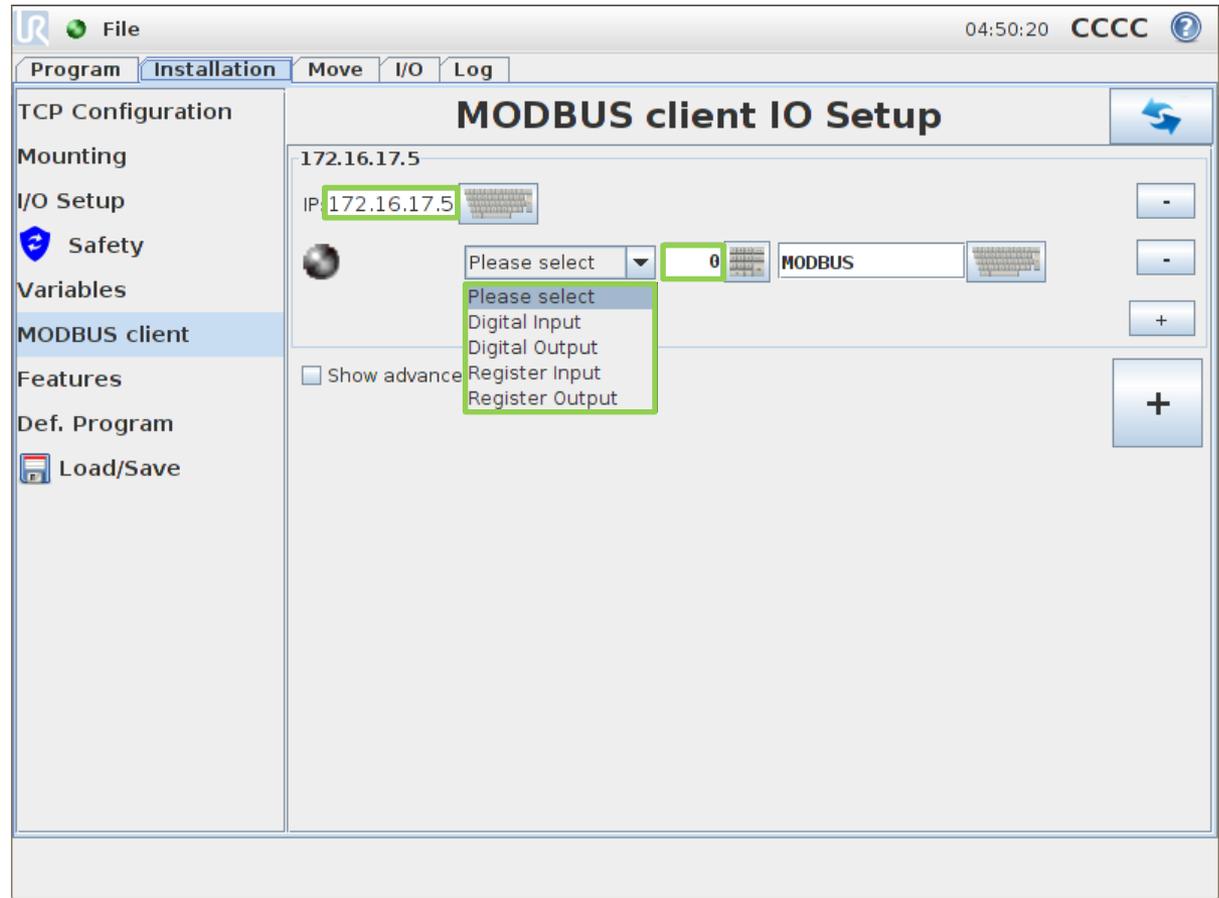
- Phoenix Contact ILB ETH 24 DI16 DIO16 2TX
  - 16 digital outputs
  - 16 digital inputs
  - 2-port switch



	Modbus Register Table (16-Bit Words)	Modbus Input Discretes Table (Bits)	Modbus Coil Table	Access	Function
<b>Process data</b>	0	0-15	-	Read only	Digital inputs (DIO)
	1	16-32	-	Read only	Digital inputs (DI)
	2	-	0-15	Read/write	Digital outputs
	3	-	-	Read only	Reserved
<b>Diagnostics</b>	4	-	-	Read only	Status register
	5	-	-	Read only	I/O diagnostic register
	6	-	-	Read only	NetFail reason
	7	-	-	Read only	IBS diagnostic register (for compatibility with FL IL 24 BK)
	8	-	-	Read only	IBS para register (for compatibility with FL IL 24 BK)
<b>Special register</b>	1280	-	-	Read/write	Modbus timeout connection monitoring
	2000	-	-	Read/write	Process data watchdog timeout
	2002	-	-	Read/write	Fault response mode
	2004	-	-	Read/write	NetFail test (same value as register 6)
	2006	-	-	Read/write	Command register

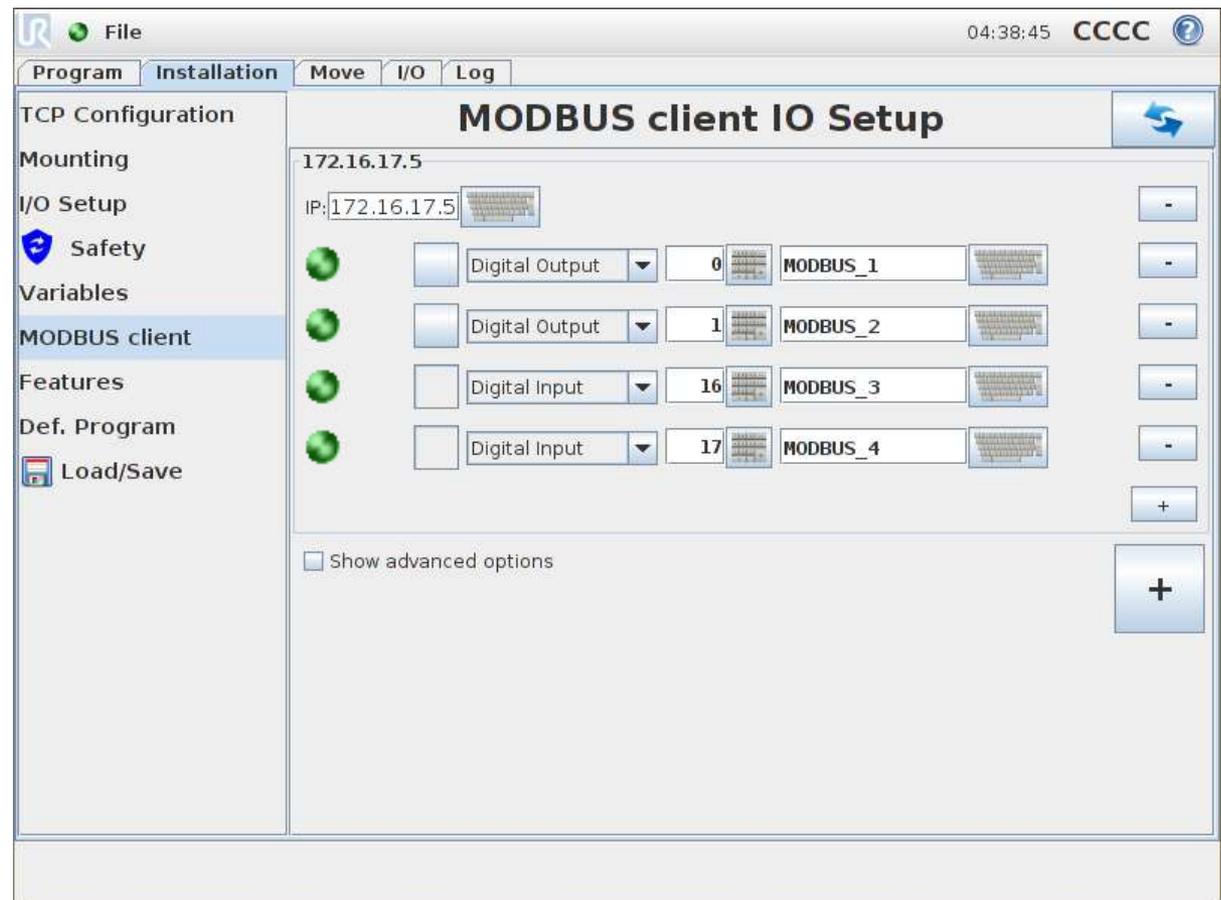
## Setup server

- Device setup
  - Add new device
  - Set IP-address of device (Server)
- Signal setup
  - Add new device
  - Add signal
  - Define data type
  - Set signal address
  - Define signal name



## Setup server

- Setup
  - Setup 2 digital inputs
  - Setup 2 digital outputs
  - Save Installation
  
- Monitoring
  - Signals can be monitored in I/O tab



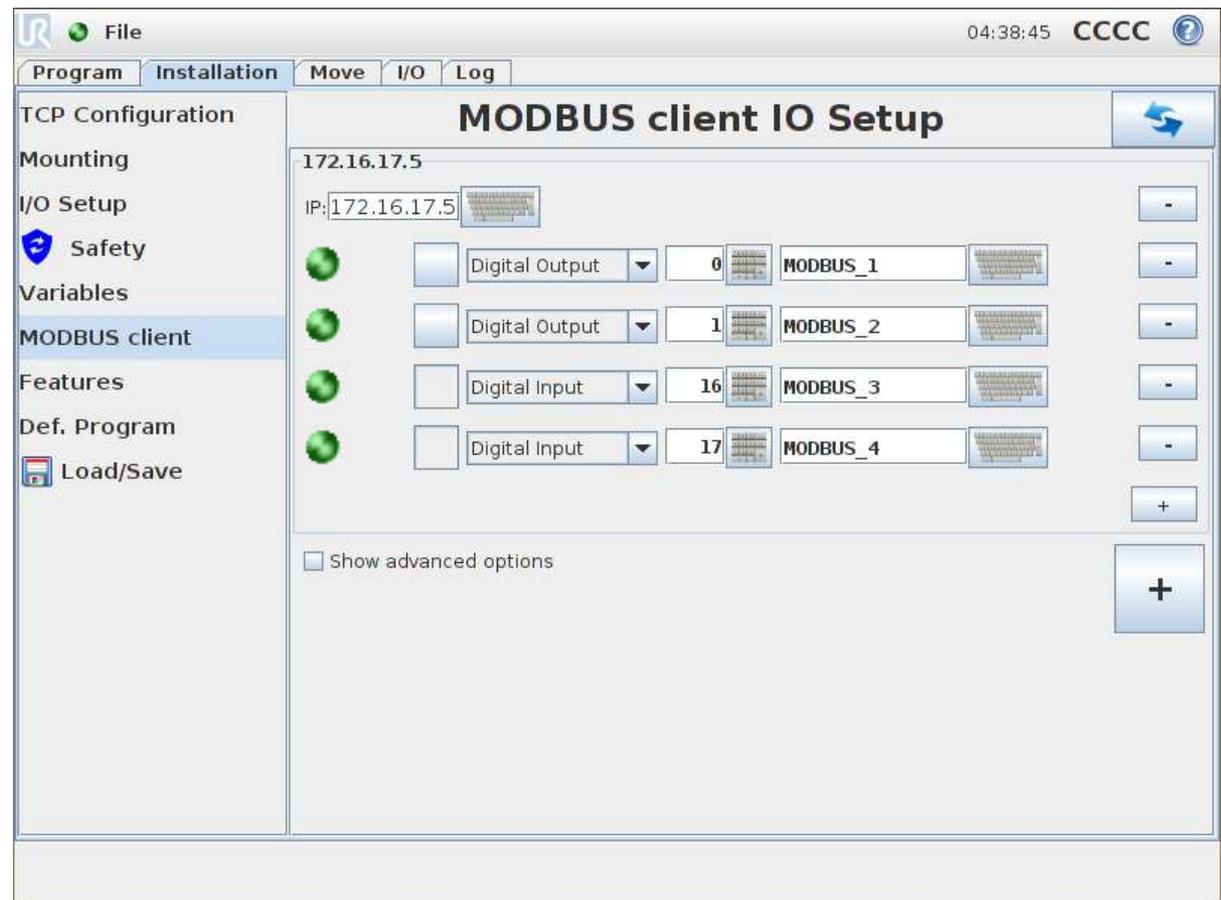
### Connectivity status

	Status
	Connection ok
	Update frequency warning
	No connection
<b>E4</b>	Exception code

## Use Modbus signal in program

- Use of signals
  - Same functionality as normal digital signals

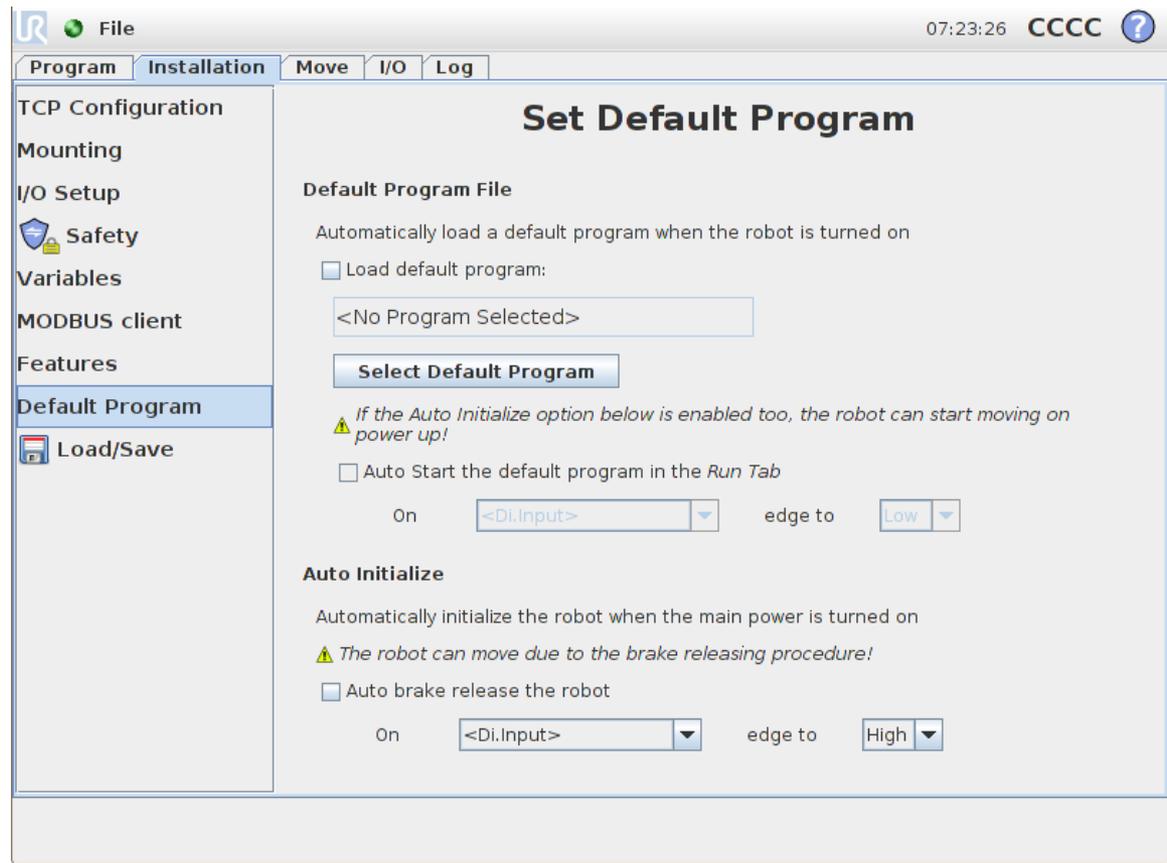
```
Robot Program
MoveL
  Waypoint_1
  Set MODBUS_1 = True
  Waypoint_2
  Waypoint_3
  Waypoint_4
  Set MODBUS_1 = False
```



- Save sample program as modbus.urp

## Default program

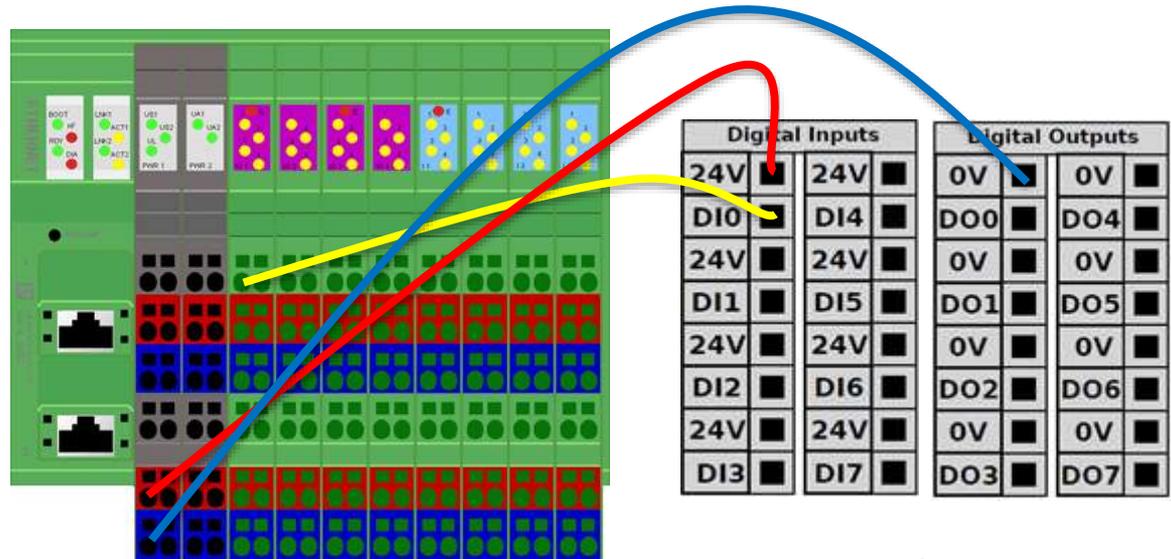
- Purpose
  - Set robot to auto start with no operator action needed
- How to
  - Select default program
  - Set digital input to auto start
  - Set digital input to auto initialize
  - Save Installation
  - Reboot



- Note: robot will *always* load default.installation at startup

## Lab exercise

- Set up a MODBUS connection and write a program to test it
- Connection Setup
  - Connect the MODBUS device power terminals to 0V and 24V supply in the controller
  - Connect pin 1.1 on block IO1 of your MODBUS device to DI0 in the controller
  - Setup network and ensure your Controller and MODBUS device are in the same subnet
  - Under the installation tab add the remote digital output
  - Toggle the output and watch the indicator light on the MODBUS device to verify the setup is correct



### Lab exercise 2

- Write a program to test the connection
  - Turn on remote output via MODBUS
  - Verify local input has gone high
  - Turn off remote output via MODBUS
  - Verify local input has gone low
  - Display a popup with the results of the test
- Set a Timeout
  - Now Use a thread to create a 5 second timeout on the test.



Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

wizards

Modbus TCP

10

**Service**

safety standards

Adjustable safety

## Downloads and tips

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

- Programming tips
- How to's
- Safety guide
- Download
  - Magic files
  - Software firmware
  - URSim
  - Log reader
  - CAD drawings
  - Manuals

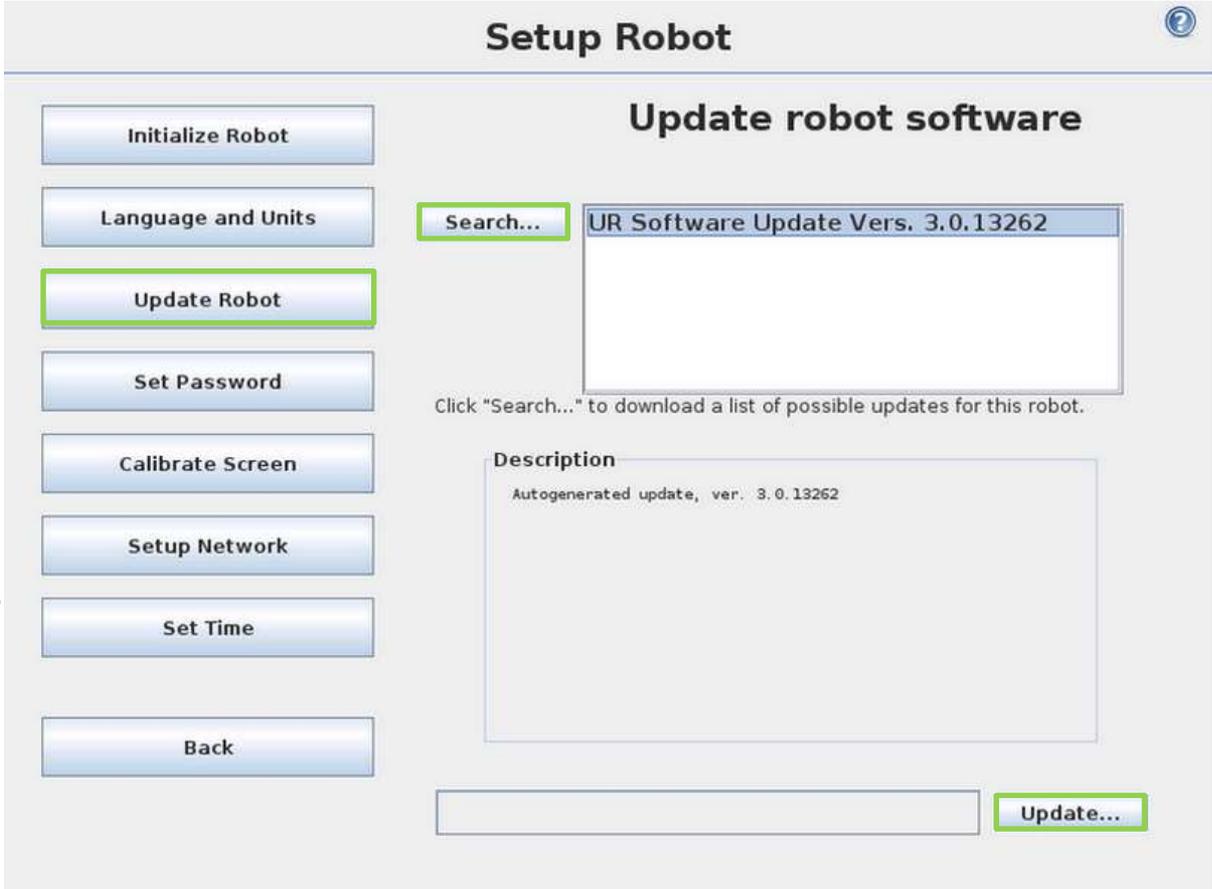
## Magic files

- Easy backup
  - Backup programs
  - Backup log history
  - Backup configuration files
- Others
  - Upload programs
  - Screenshot of GUI
- How to
  - Download Magic file from support site
  - Copy file to USB-stick
  - Insert USB-stick in TP » red **USB** warning appears
  - Await green **USB** sign on GUI » copy completed



## Software update

- How to
  - Download latest software from support site
  - Copy file to USB-stick
  - Insert USB-stick in TP
  - Go to Setup\Update
  - Press "Search" and select the update
  - Press "Update"
- NOTE: Robot will power down after update

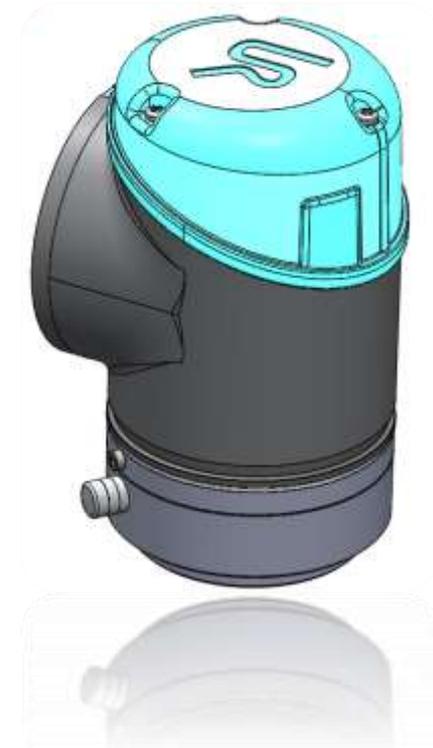


The screenshot shows the 'Setup Robot' interface. On the left, a vertical menu contains buttons for 'Initialize Robot', 'Language and Units', 'Update Robot' (highlighted with a green border), 'Set Password', 'Calibrate Screen', 'Setup Network', 'Set Time', and 'Back'. The main area is titled 'Update robot software' and features a 'Search...' button (highlighted with a green border) next to a text box containing 'UR Software Update Vers. 3.0.13262'. Below this, a 'Description' box shows 'Autogenerated update, ver. 3.0.13262'. At the bottom right, there is an 'Update...' button (highlighted with a green border) and an empty text input field. A note below the search box reads: 'Click "Search..." to download a list of possible updates for this robot.'

- IMPORTANT: ALWAYS MAKE FULL BACKUP BEFORE UPDATING SOFTWARE!**

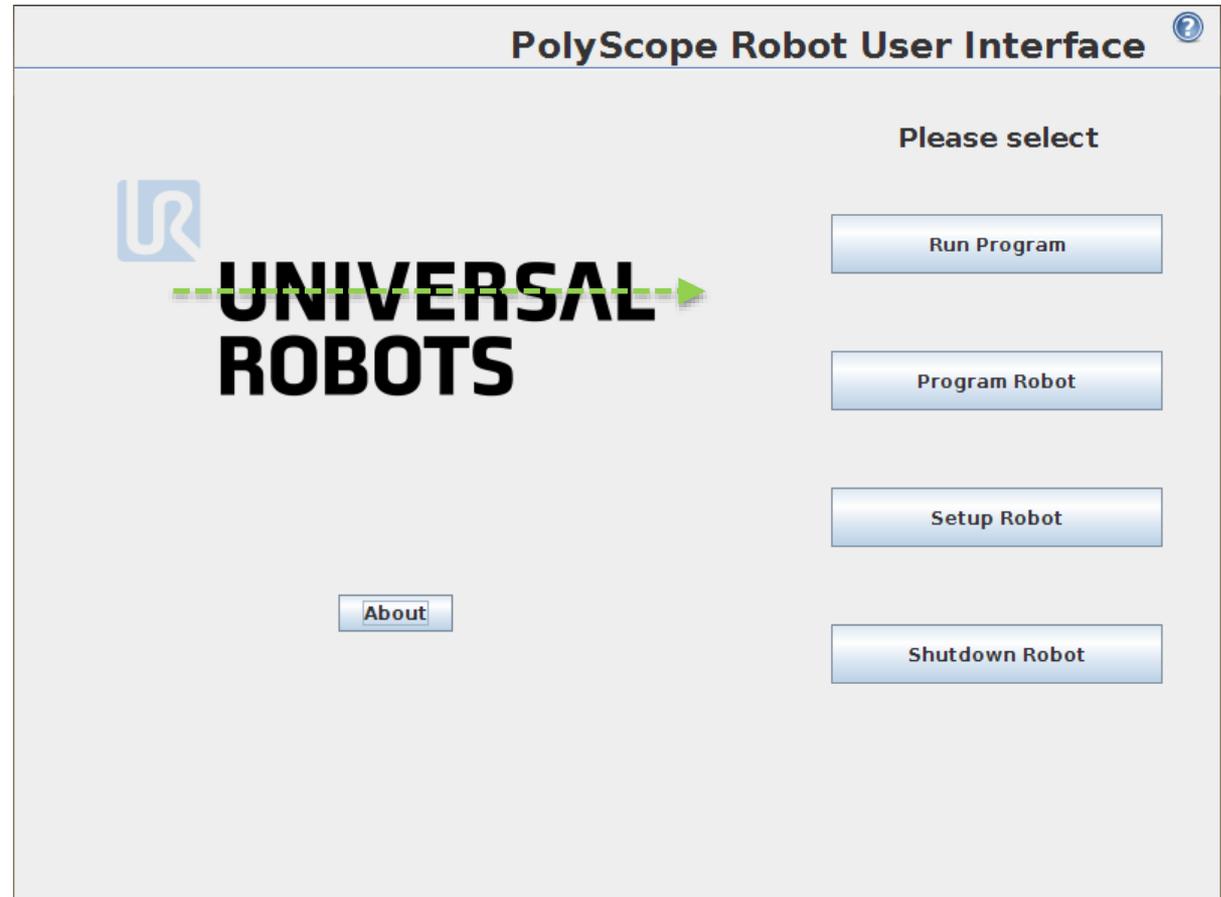
## Firmware update

- Firmware is software located in each joint
  - Firmware controls the joint
  - Can be updated if necessary
- Software update *is required* prior to updating firmware
  - During software update, the firmware is automatically copied to flash card
  - No additional download required



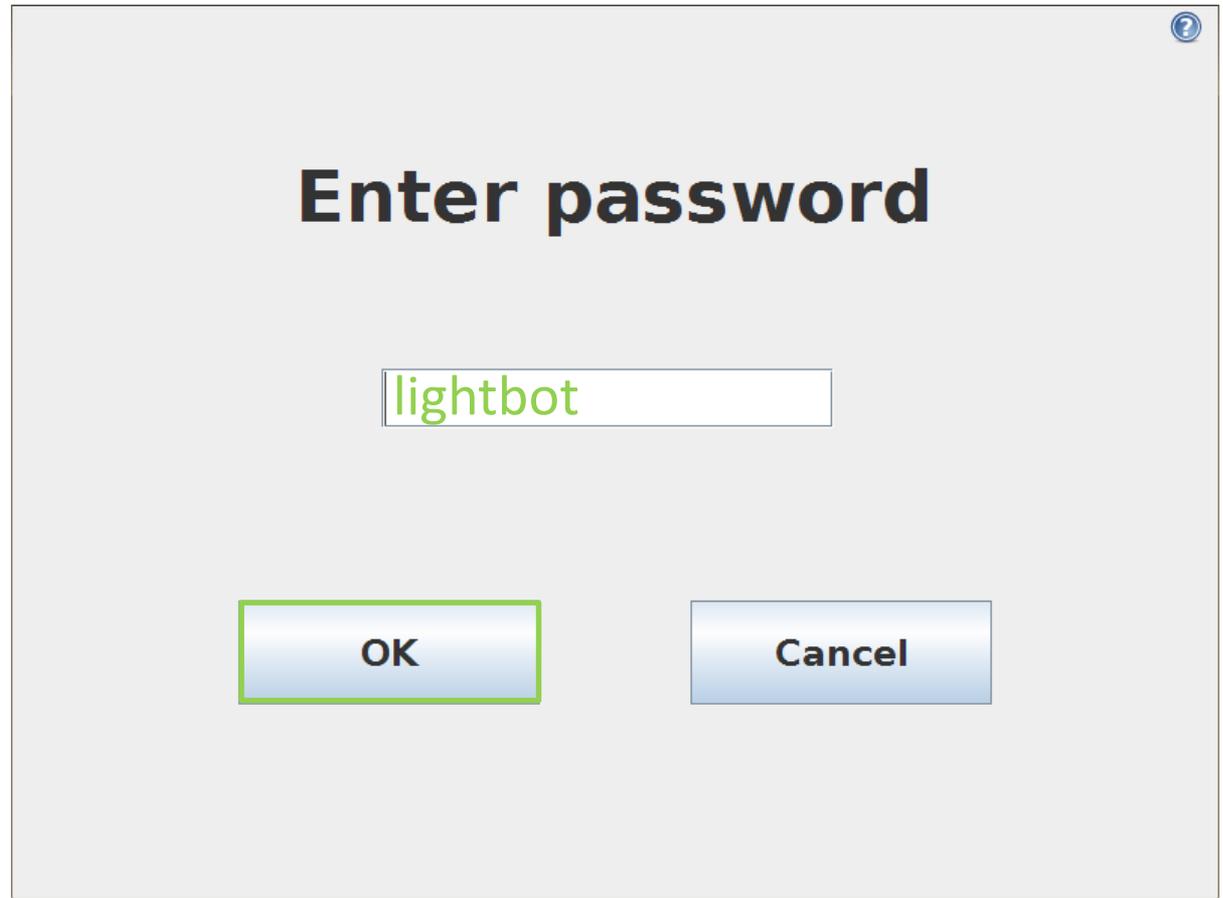
## Firmware update

- Update procedure
  - Drag finger across UNIVERSAL logo on Welcome screen



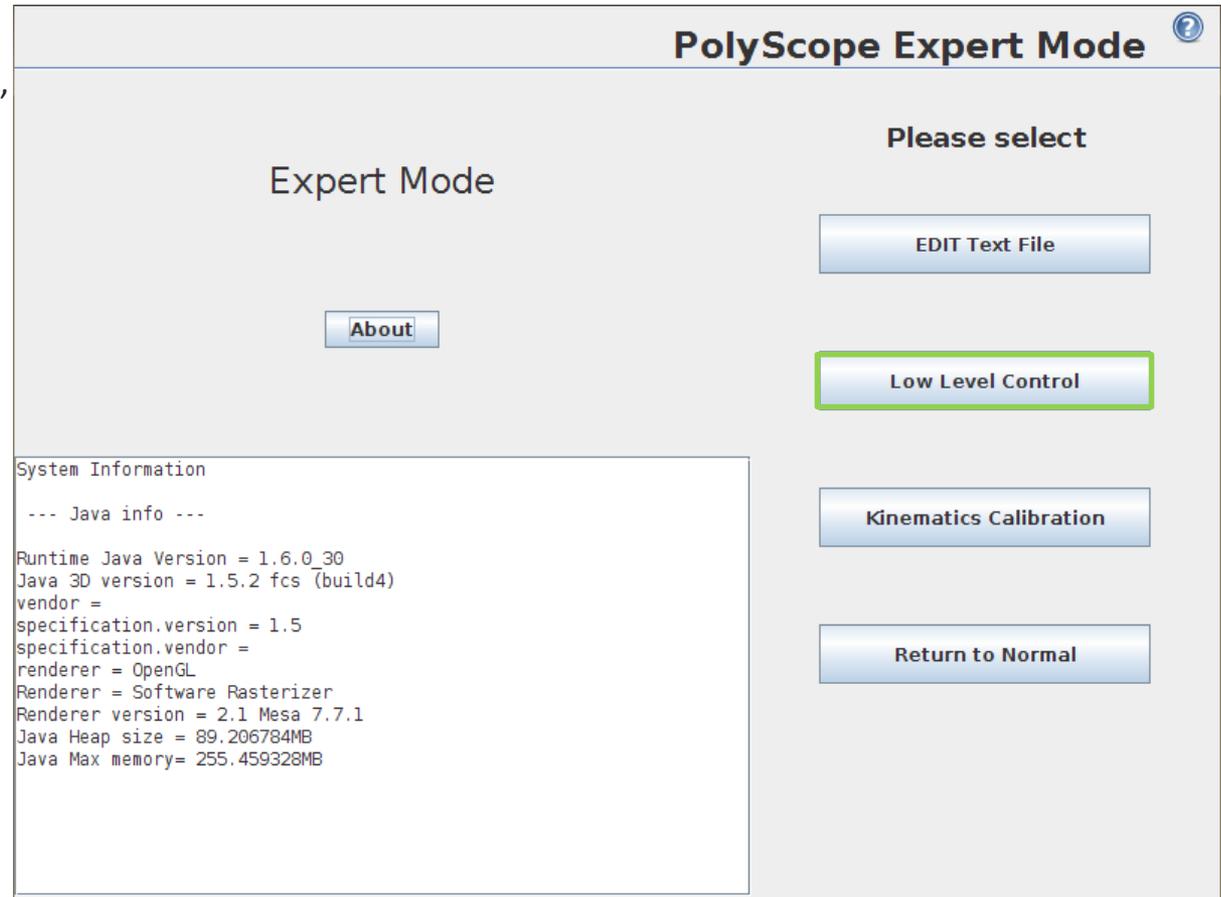
## Firmware update

- Update procedure
  - Enter password  
*lightbot*
  - Press OK to access  
Expert Mode



## Firmware update

- Update procedure
  - Press "Low Level Control"



The screenshot shows the PolyScope Expert Mode interface. At the top, it says "PolyScope Expert Mode" with a help icon. The main area is titled "Expert Mode" and contains an "About" button. On the right side, there is a "Please select" section with four buttons: "EDIT Text File", "Low Level Control" (highlighted with a green border), "Kinematics Calibration", and "Return to Normal". At the bottom left, there is a "System Information" window displaying the following text:

```
System Information
--- Java info ---
Runtime Java Version = 1.6.0_30
Java 3D version = 1.5.2 fcs (build4)
vendor =
specification.version = 1.5
specification.vendor =
renderer = OpenGL
Renderer = Software Rasterizer
Renderer version = 2.1 Mesa 7.7.1
Java Heap size = 89.206784MB
Java Max memory= 255.459328MB
```

## Firmware update

- Update procedure
  - General tab
    - Press "Turn power on"
    - Verify status on all joints: BOOTLOADER
  - Firmware tab
    - Select "All joints"
    - Press "UPDATE Firmware"
    - Await "Firmware update complete" on STATUS line
- Press "Back" and "Return to Normal"

```

J0: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
J1: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
J2: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
J3: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
J4: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
J5: BOOTLOADER P:+0.0000 S:+0.000 C:+0.000 V:+0.0 TM:+0.00 TE:+0.00 STS:0
Tool: [OFF] DI:00 A1:0.0 A2:0.0 C:+0.01 mA V:+453.22
Safety Control Board: DI:00000000 DO:00000000 AI1:0.0 AI2:0.0 A01:0.0 A02:0.0
Safety Control Board: STATE: Power on MV:+48.2 RV:+48.2 CR:+0.75 mA CIO:+0.22 mA T:+30.61
                    
```

?

General
Move
Calibration
Firmware
Joint ID

Current joint  
 All joints

UPDATE Firmware

STATUS

STOP!

Follow last line

Back

```

Info: Setting stepsize to 0.0005 rad/sec
Info: Setting stepsize to 0.0100 rad/sec
Info: Setting current joint to 0
Powering robot on...
SUCCESS: Command executed
UA: C4A90: Broken communication: Packet counter disagreement in packet from joint 4
UB: C4A93: Broken communication: Packet counter disagreement in packet from processor A to joints
T: C11A1: Bad CRC error: !CODE_11A1!
                    
```

## offline simulator

- URSim
  - Offline programming software
  - Only runs on Linux operating system
- Available for download from support site as:
  - Installation file
  - Virtual machine

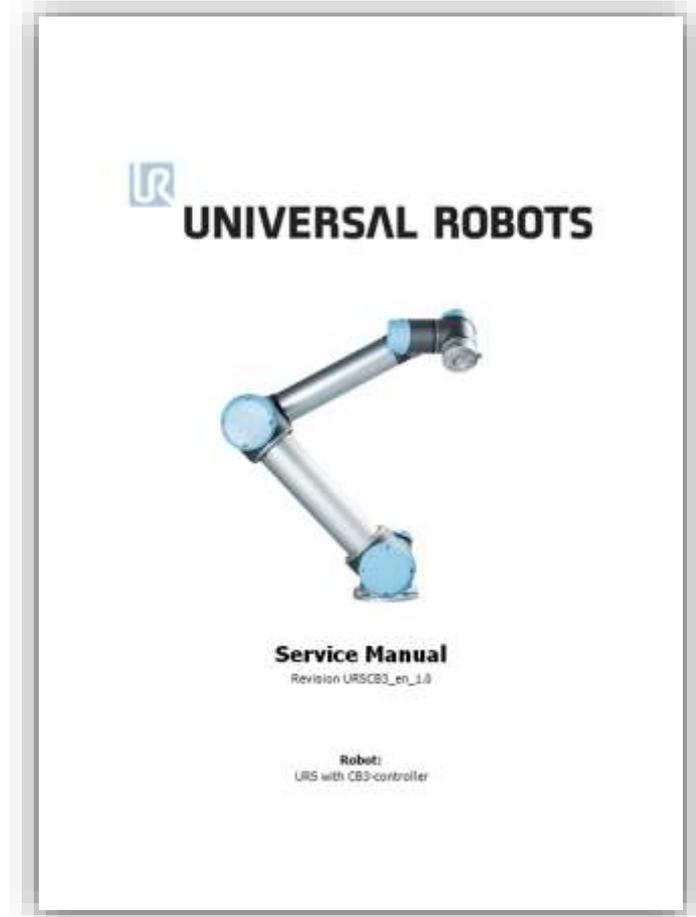


next

Maintenance

## Preventive maintenance

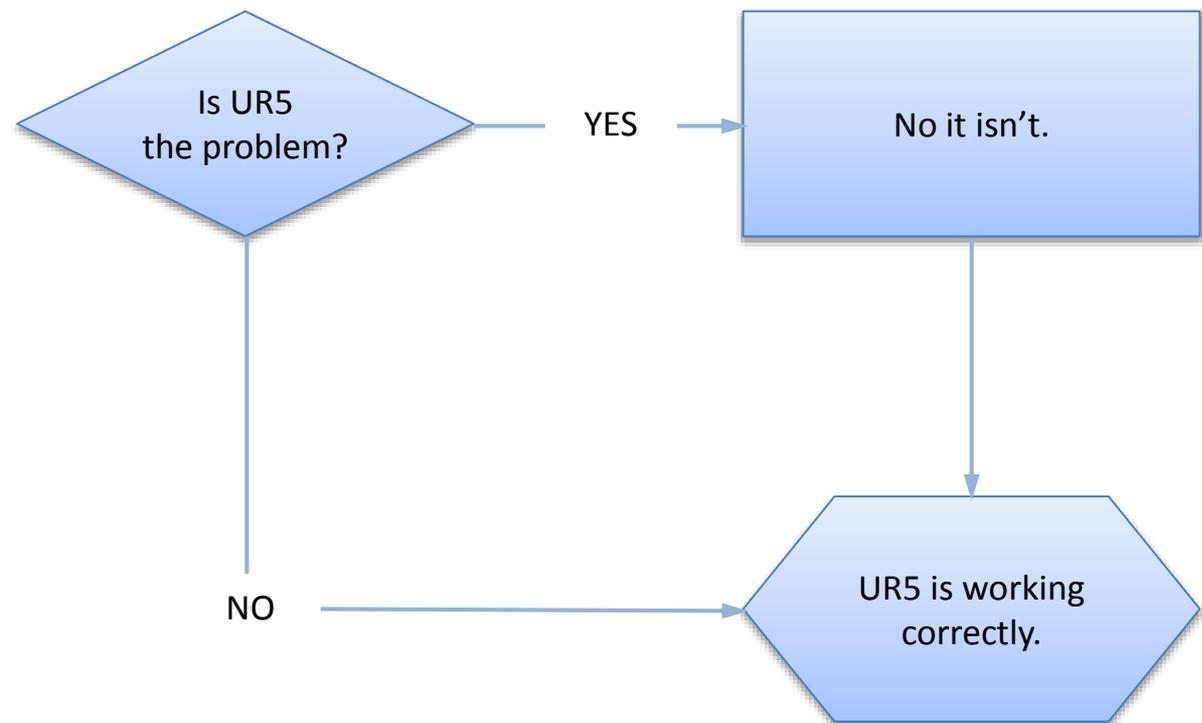
- Service manual content
  - Preventive maintenance
    - Robot arm
    - Controller box
  - Schematic drawings
  - Service and replacement of parts
  - Software
  - Troubleshooting
  - Spare parts
  - Packing of robot
  
- Preventive maintenance
  - Download Service Manual from support site
  - Walk through *Chapter 2. Preventive Maintenance*



## Service manual

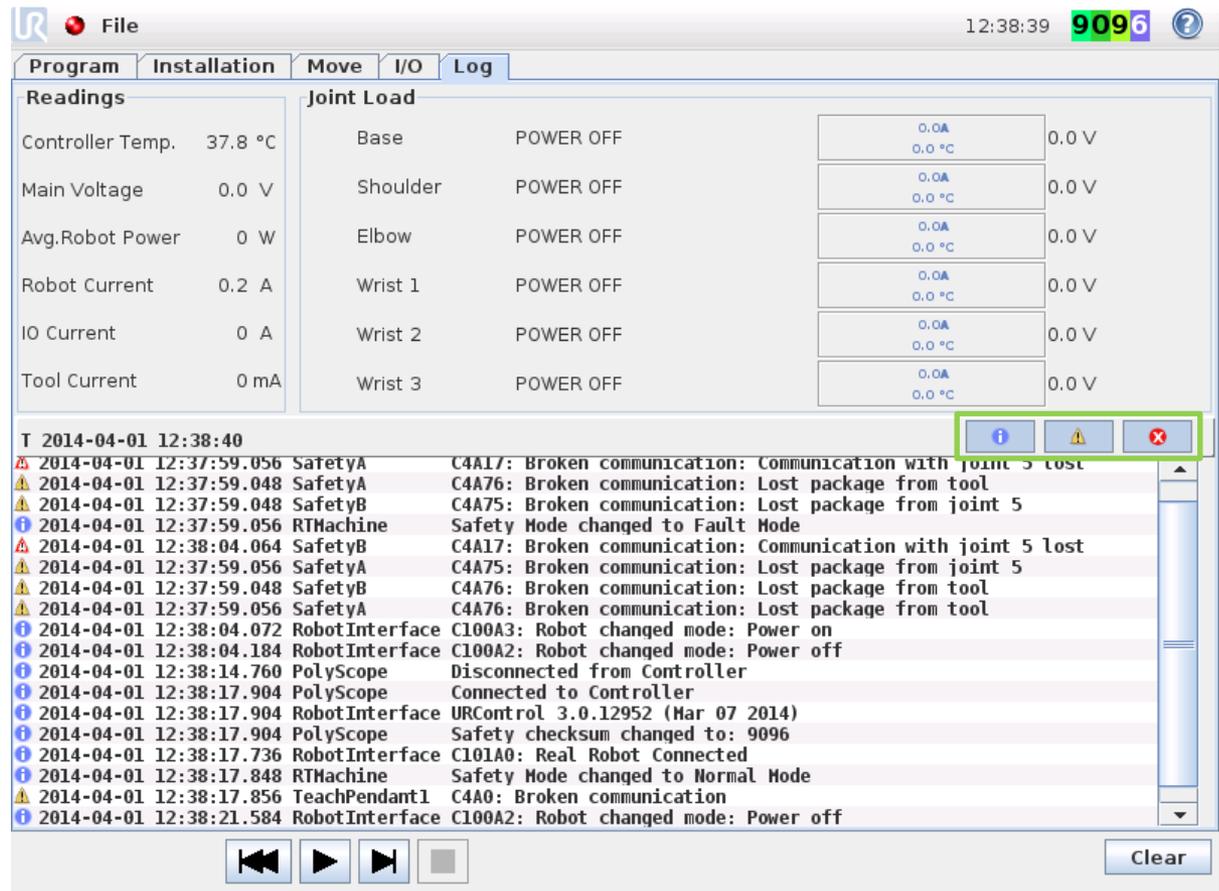
- Troubleshooting
  - Log history
  - Joint replacement
  - Joint calibration
  - Change joint ID
  - Warranty claim

### UR5 Troubleshooting Flowchart



## Log history

- Readings
  - Control box health
  
- Joint load
  - Joint status
  
- Log history
  - Displays valuable information about robot health
  - Show/hide
    - Information
    - Warnings
    - Errors



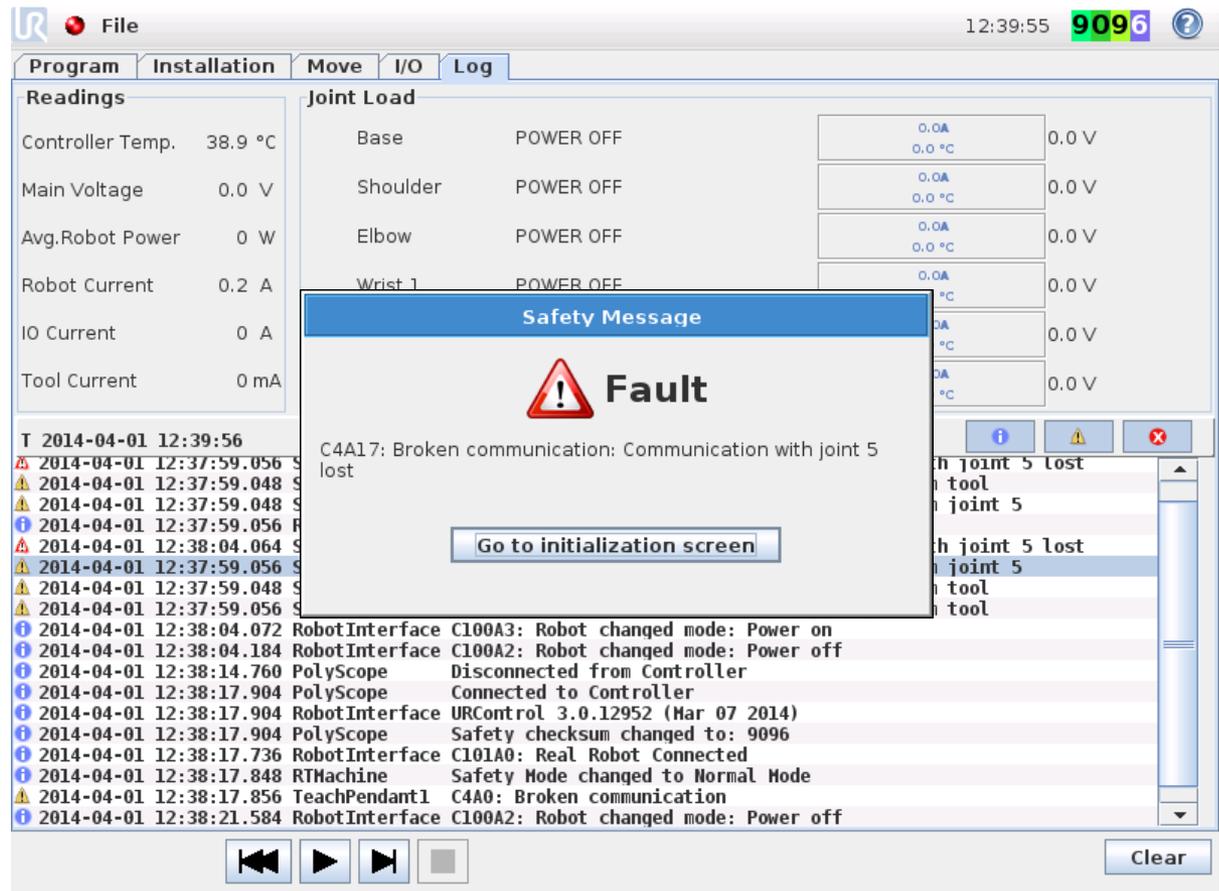
Program	Installation	Move	I/O	Log
<b>Readings</b>		<b>Joint Load</b>		
Controller Temp.	37.8 °C	Base	POWER OFF	0.0A 0.0 °C 0.0 V
Main Voltage	0.0 V	Shoulder	POWER OFF	0.0A 0.0 °C 0.0 V
Avg.Robot Power	0 W	Elbow	POWER OFF	0.0A 0.0 °C 0.0 V
Robot Current	0.2 A	Wrist 1	POWER OFF	0.0A 0.0 °C 0.0 V
IO Current	0 A	Wrist 2	POWER OFF	0.0A 0.0 °C 0.0 V
Tool Current	0 mA	Wrist 3	POWER OFF	0.0A 0.0 °C 0.0 V

Time	Component	Message
T 2014-04-01 12:38:40		
2014-04-01 12:37:59.056	SafetyA	C4A17: Broken communication: Communication with joint 5 lost
2014-04-01 12:37:59.048	SafetyA	C4A76: Broken communication: Lost package from tool
2014-04-01 12:37:59.048	SafetyB	C4A75: Broken communication: Lost package from joint 5
2014-04-01 12:37:59.056	RTMachine	Safety Mode changed to Fault Mode
2014-04-01 12:38:04.064	SafetyB	C4A17: Broken communication: Communication with joint 5 lost
2014-04-01 12:37:59.056	SafetyA	C4A75: Broken communication: Lost package from joint 5
2014-04-01 12:37:59.048	SafetyB	C4A76: Broken communication: Lost package from tool
2014-04-01 12:37:59.056	SafetyA	C4A76: Broken communication: Lost package from tool
2014-04-01 12:38:04.072	RobotInterface	C100A3: Robot changed mode: Power on
2014-04-01 12:38:04.184	RobotInterface	C100A2: Robot changed mode: Power off
2014-04-01 12:38:14.760	PolyScope	Disconnected from Controller
2014-04-01 12:38:17.904	PolyScope	Connected to Controller
2014-04-01 12:38:17.904	RobotInterface	URControl 3.0.12952 (Mar 07 2014)
2014-04-01 12:38:17.904	PolyScope	Safety checksum changed to: 9096
2014-04-01 12:38:17.736	RobotInterface	C101A0: Real Robot Connected
2014-04-01 12:38:17.848	RTMachine	Safety Mode changed to Normal Mode
2014-04-01 12:38:17.856	TeachPendant1	C4A0: Broken communication
2014-04-01 12:38:21.584	RobotInterface	C100A2: Robot changed mode: Power off

## Log history

- Demonstrate error
  - Remove blue lid from Wrist 3
  - Disconnect green comm. cable from prev. joint
  - Verify that robot detects error and displays popup



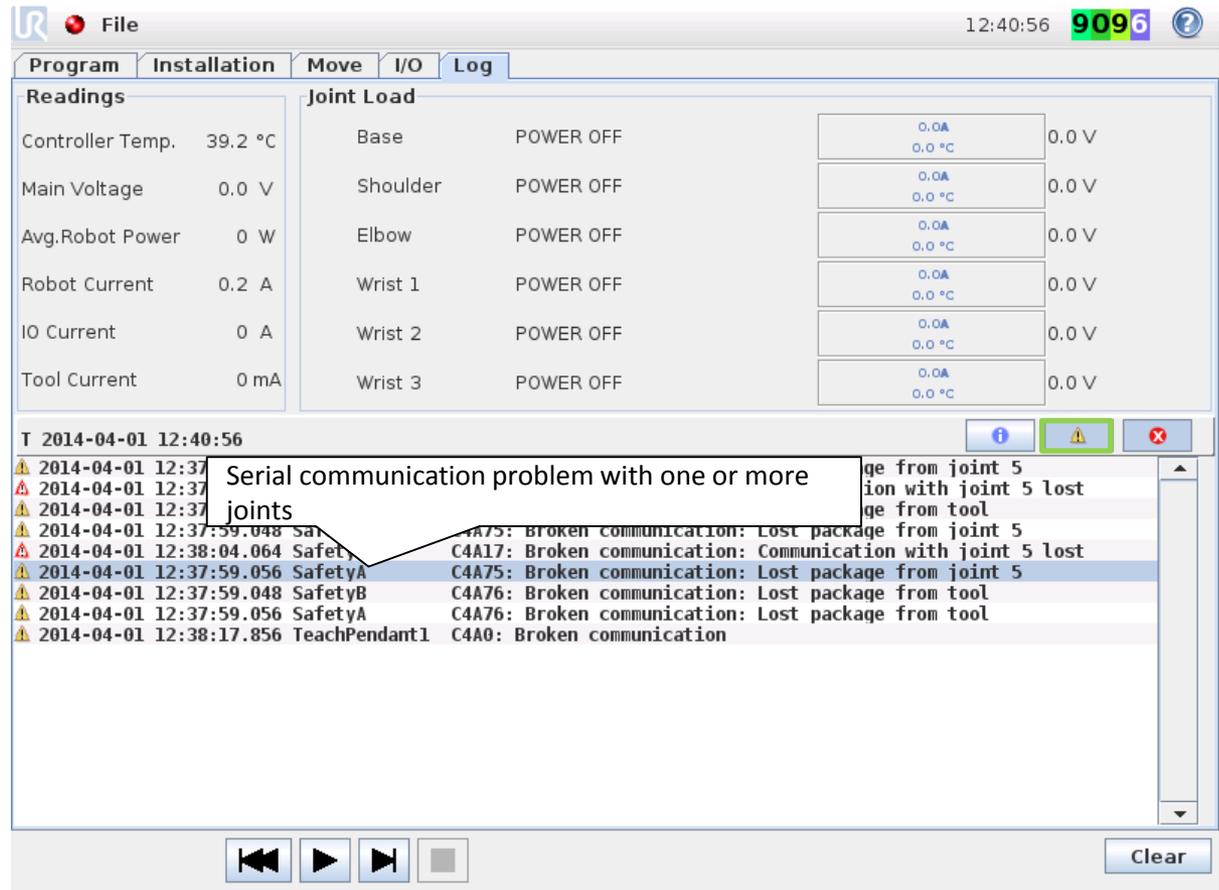
The screenshot shows the Universal Robots control interface. A 'Safety Message' popup is displayed in the center, indicating a 'Fault' with the message 'C4A17: Broken communication: Communication with joint 5 lost'. Below the message is a 'Go to initialization screen' button. In the background, the 'Log' window is open, showing a list of system events. The log entries include:

- T 2014-04-01 12:39:56
- 2014-04-01 12:37:59.056 S
- 2014-04-01 12:37:59.048 S
- 2014-04-01 12:37:59.048 S
- 2014-04-01 12:37:59.056 P
- 2014-04-01 12:38:04.064 S
- 2014-04-01 12:37:59.056 S
- 2014-04-01 12:37:59.048 S
- 2014-04-01 12:37:59.056 S
- 2014-04-01 12:38:04.072 RobotInterface C100A3: Robot changed mode: Power on
- 2014-04-01 12:38:04.184 RobotInterface C100A2: Robot changed mode: Power off
- 2014-04-01 12:38:14.760 PolyScope Disconnected from Controller
- 2014-04-01 12:38:17.904 PolyScope Connected to Controller
- 2014-04-01 12:38:17.904 RobotInterface URControl 3.0.12952 (Mar 07 2014)
- 2014-04-01 12:38:17.904 PolyScope Safety checksum changed to: 9096
- 2014-04-01 12:38:17.736 RobotInterface C101A0: Real Robot Connected
- 2014-04-01 12:38:17.848 RTMachine Safety Mode changed to Normal Mode
- 2014-04-01 12:38:17.856 TeachPendant1 C4A0: Broken communication
- 2014-04-01 12:38:21.584 RobotInterface C100A2: Robot changed mode: Power off

The interface also shows 'Readings' on the left (Controller Temp. 38.9 °C, Main Voltage 0.0 V, Avg. Robot Power 0 W, Robot Current 0.2 A, IO Current 0 A, Tool Current 0 mA) and 'Joint Load' on the right (Base, Shoulder, Elbow, Wrist 1, all POWER OFF).

## Log history

- Demonstrate error
  - Check log history
  - Show/hide Information entries
  - Highlight line by tapping once on it
  
- Log history
  - Saved in textfile named log\_history.txt
  - Use Magic file to backup

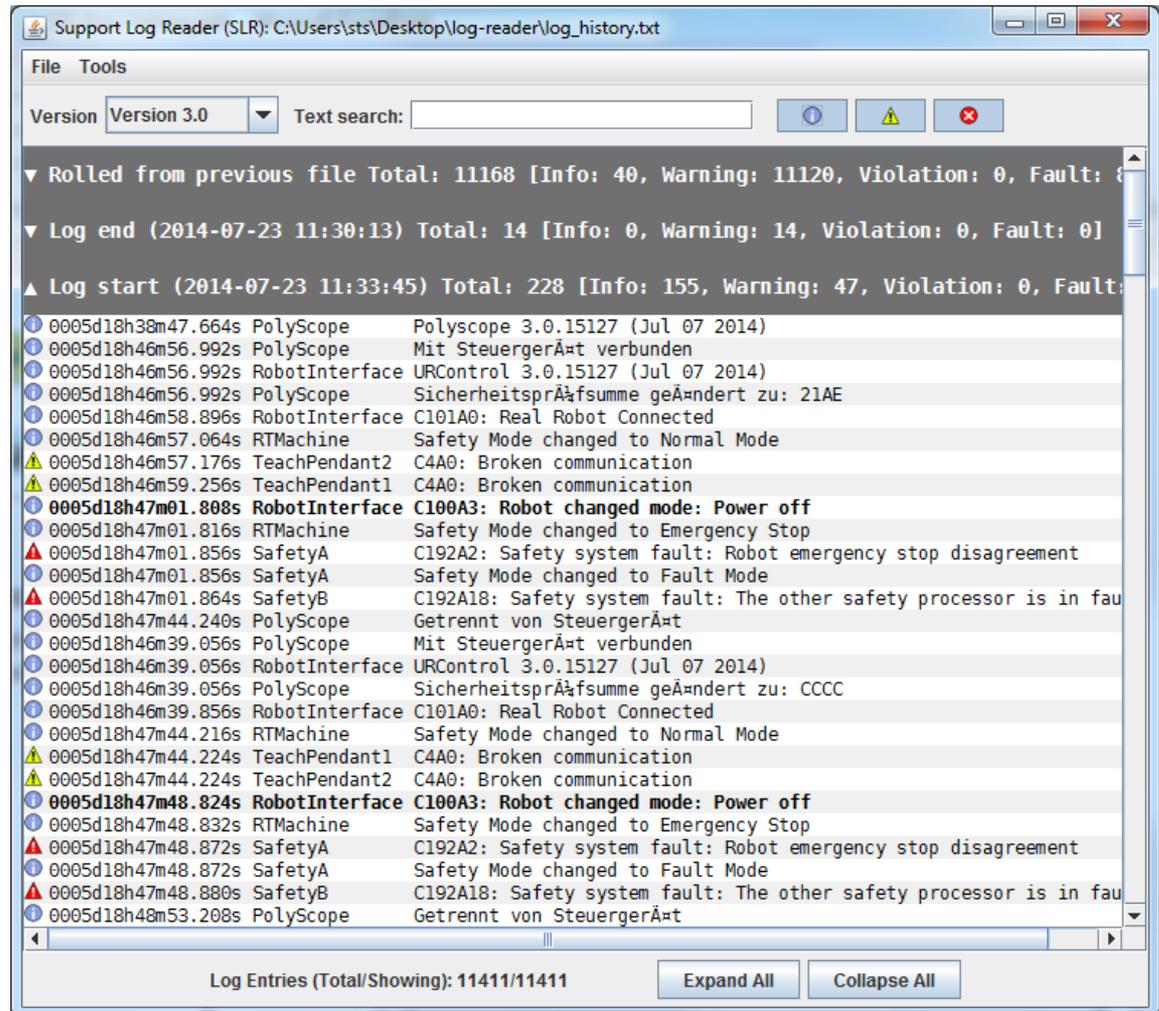


The screenshot shows the 'Log' tab in the software interface. At the top, there are tabs for 'Program', 'Installation', 'Move', 'I/O', and 'Log'. Below these are two sections: 'Readings' and 'Joint Load'. The 'Readings' section shows various sensor values like Controller Temp (39.2 °C), Main Voltage (0.0 V), Avg. Robot Power (0 W), Robot Current (0.2 A), IO Current (0 A), and Tool Current (0 mA). The 'Joint Load' section shows a table of joint statuses, all currently 'POWER OFF'. Below these sections is a log window titled 'T 2014-04-01 12:40:56'. It contains a list of error messages with timestamps and descriptions. A tooltip is visible over the first error message.

Timestamp	Source	Message
2014-04-01 12:37:59.048	SafetyA	Serial communication problem with one or more joints
2014-04-01 12:37:59.048	SafetyB	C4A75: Broken communication: Lost package from joint 5
2014-04-01 12:37:59.056	SafetyA	C4A17: Broken communication: Communication with joint 5 lost
2014-04-01 12:37:59.056	SafetyA	C4A75: Broken communication: Lost package from joint 5
2014-04-01 12:37:59.048	SafetyB	C4A76: Broken communication: Lost package from tool
2014-04-01 12:37:59.056	SafetyA	C4A76: Broken communication: Lost package from tool
2014-04-01 12:38:17.856	TeachPendant1	C4A0: Broken communication

## Support Log Reader

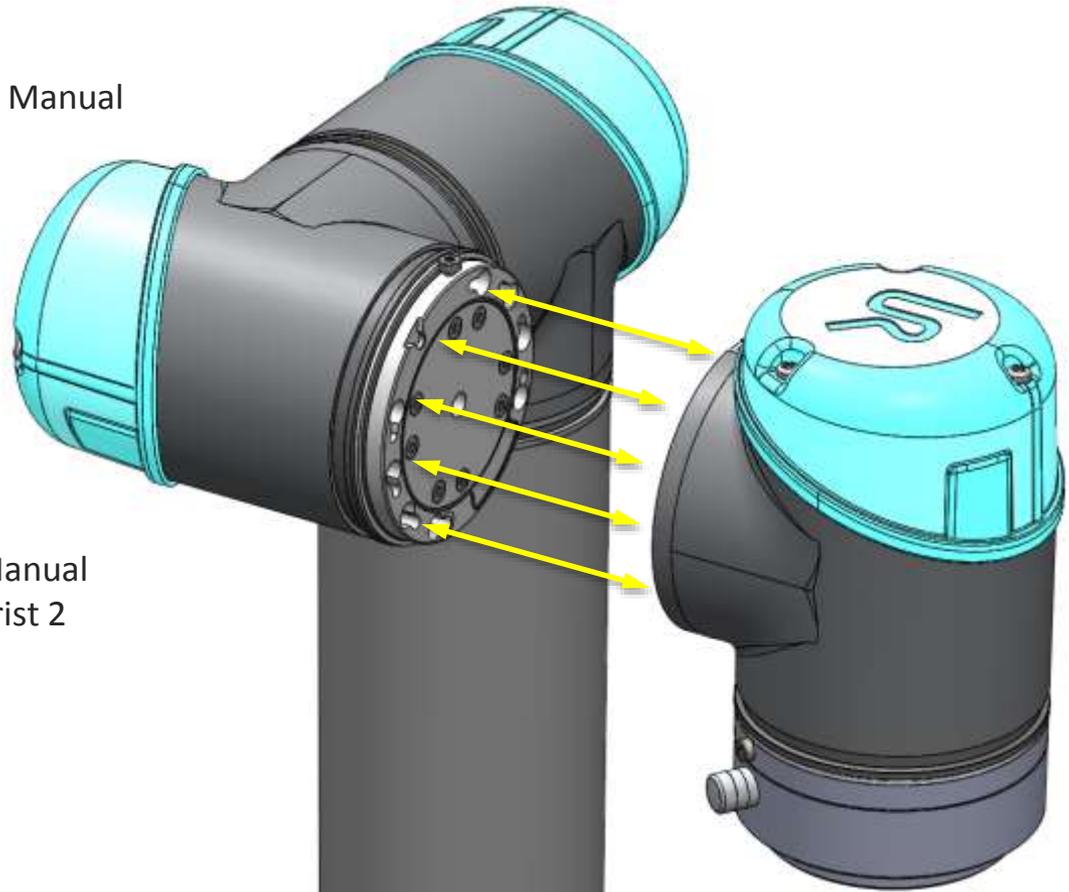
- Support Log reader
  - Read log files
  - Convert language
  - Convert to csv-file
  - Filter search
- Supports
  - CB3 file format
  - CB2 file format
- CB3 file format
  - Language converted to English
- CB2 file format
  - Language will be kept in original language



# Joint replacement

## Joint replacement

- Replacing a joint
  - Instructions available in Service Manual
- Replacing Wrist 3
  - Tools needed
    - 5.5 mm open ended spanner
    - Torx screwdriver T10
    - Allen key torx T10
    - Torque wrench 5.5 mm
    - Regular small screwdriver
  - Follow instructions in Service Manual for dismantling Wrist 3 from Wrist 2
  - Mount Wrist 3 back on



## Joint calibration

- Calibration
  - Each joint has a zero-position
  - Zero position can be set in software
- Calibrating Wrist 3
  - Follow instructions in Service Manual for performing a calibration of a joint

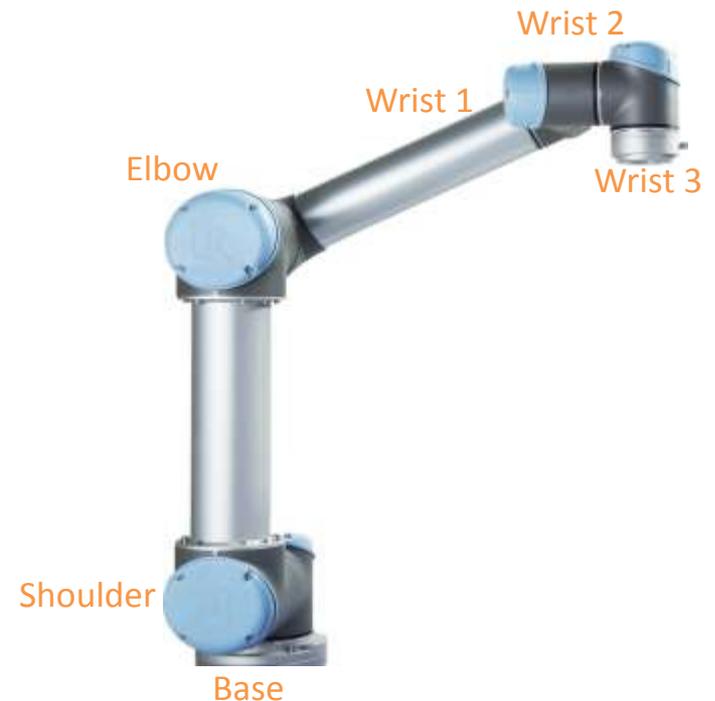


# Change joint ID

## Change joint ID

- Each joint has a unique ID no.

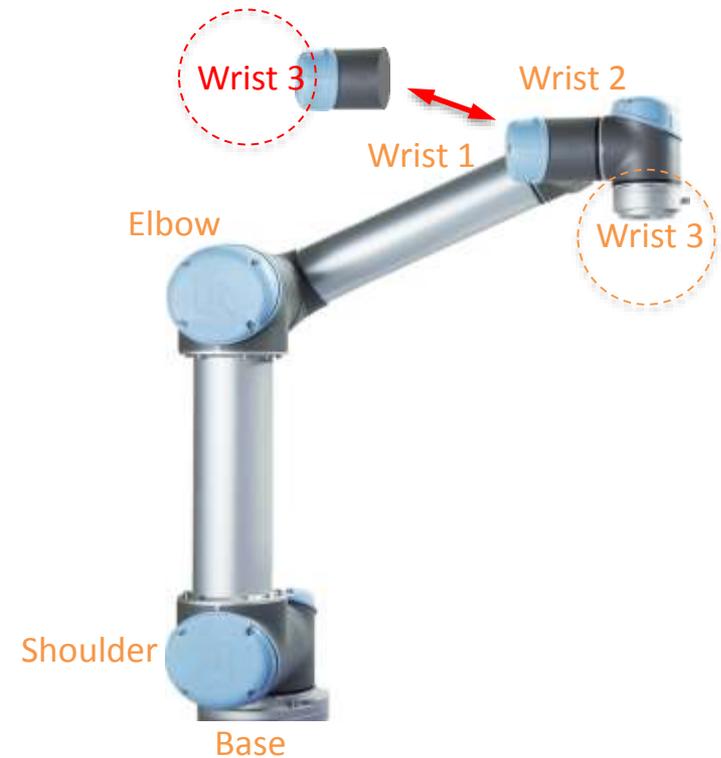
ID	joint
J0	Base
J1	Shoulder
J2	Elbow
J3	Wrist 1
J4	Wrist 2
J5	Wrist 3



- It is *not* possible to have two joints with same ID no. on same robot
  - » ID conflict will occur, causing malfunction of robot

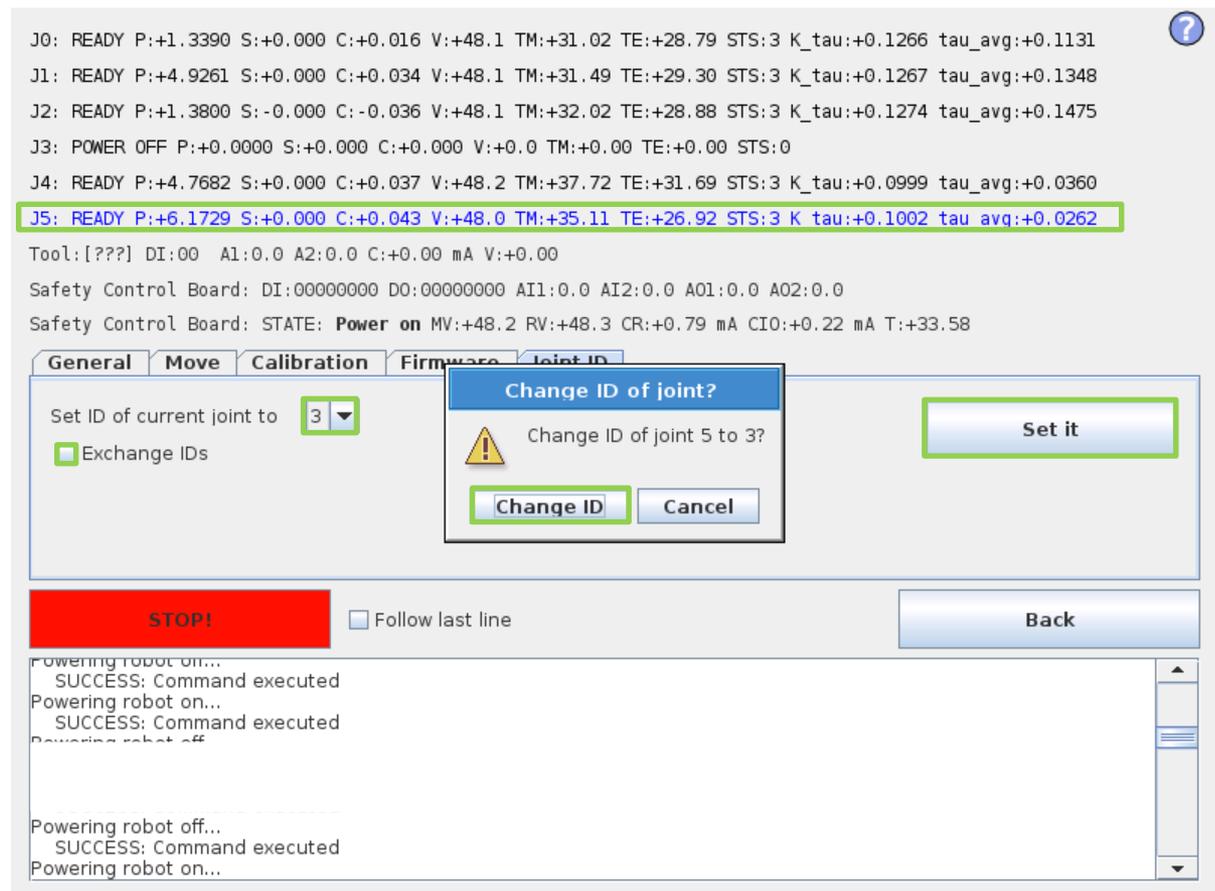
## Change joint ID

- Example
  - Wrist 1 (J3) needs to be replaced
  - Spare joint is a Wrist 3 (J5)
    - » Conflict will occur as robot has two joints with ID J5
- How to solve
  - Disconnect comm. connector for joint with correct ID no.
    - » Wrist 3 is then not accessible
  - Enter Low Level Control
    - » Change ID no. for replacement joint (J5) to J3



## Change joint ID

- Joint ID
  - Enter Low Level Control
  - Go to General tab
    - Click "Power on"
    - Click "Go to Idle"
  - Go to Joint ID tab
  - Select J5 (the one to be changed)
  - **IMPORTANT:** uncheck "Exchange IDs" box
  - In dropdown box, select ID no. 3 and press SET IT



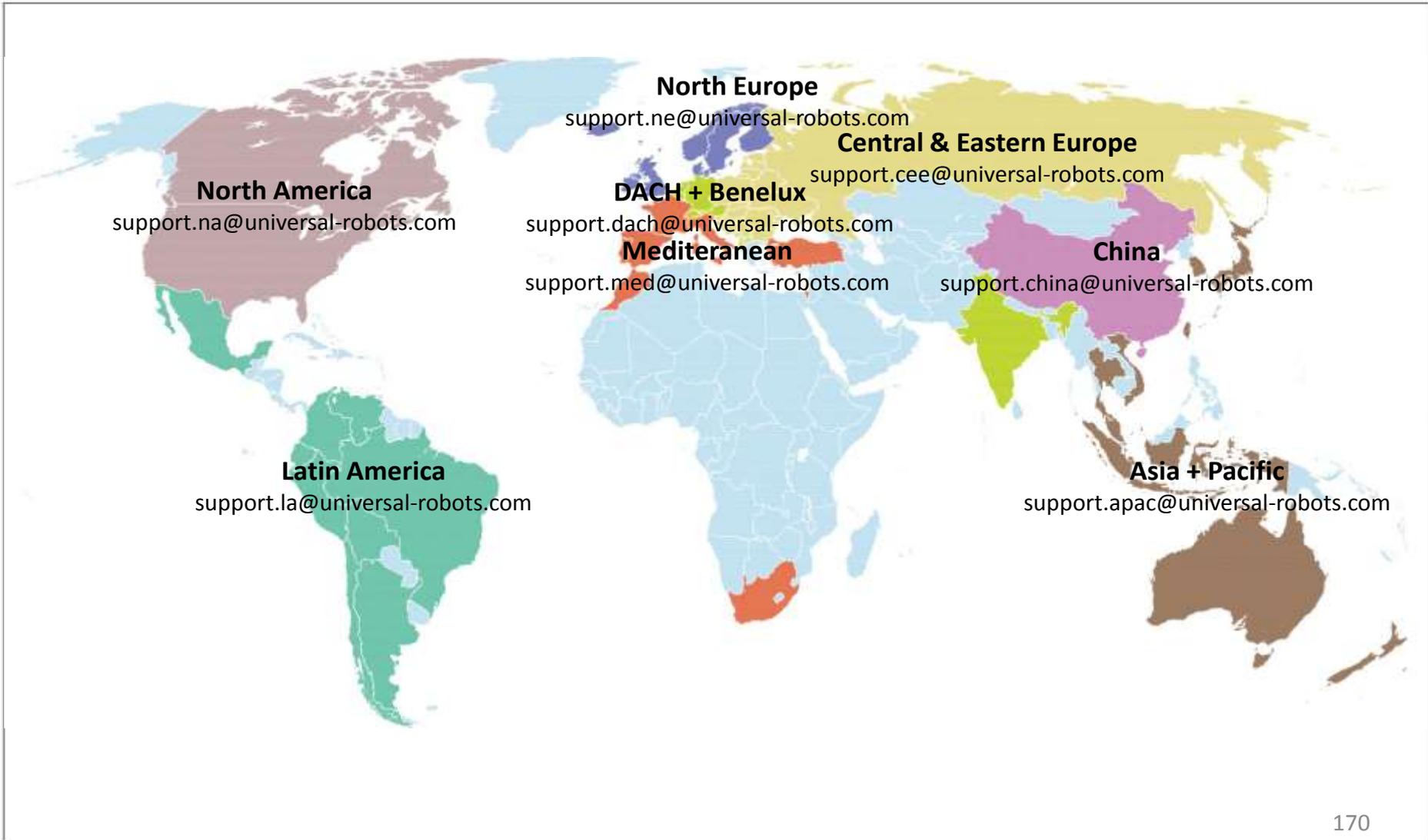
The screenshot displays the 'Joint ID' tab in the software interface. At the top, a list of joint statuses is shown, with J5 highlighted in green: J5: READY P:+6.1729 S:+0.000 C:+0.043 V:+48.0 TM:+35.11 TE:+26.92 STS:3 K\_tau:+0.1002 tau\_avg:+0.0262. Below this, the 'General' tab is active, showing 'Set ID of current joint to' with a dropdown menu set to '3'. The 'Exchange IDs' checkbox is unchecked. A dialog box titled 'Change ID of joint?' is open, asking 'Change ID of joint 5 to 3?' with 'Change ID' and 'Cancel' buttons. A 'Set it' button is also visible. At the bottom, a red 'STOP!' button and a 'Back' button are present. A log window at the very bottom shows messages like 'Powering robot on... SUCCESS: Command executed'.

- Full guide on how to change joint ID is found in Service Manual

## Warranty claim

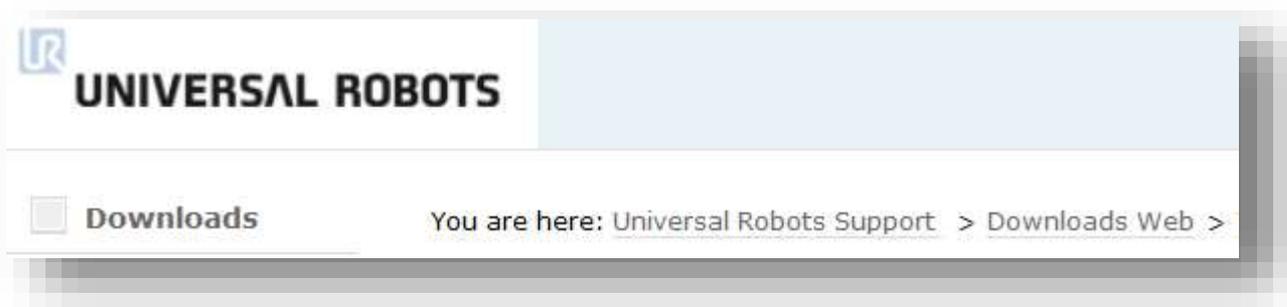
- Email regional UR support office
  
- Required information to UR
  - Robot s/n
  - Software version
  - Detailed error description
  - Attach log file
  - » Upon receiving information, regional support office will open RMA-file and ship sparepart

## Regional support



## Lab exercise

- Log onto the support site – If you do not know your login details tell us now!
- Download the log history magic file and log file reader
- Copy the log file from the robot using the magic file
- View the log file using the log file reader



Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

11

safety standards

Adjustable safety

# 11 Safety standards

...and now



## Complied international standards

- A collaborative robot system should comply with the requirements of the following international standards

Standard	Describes	Responsible
ISO 13849-1	Safety related parts of control system	Manufacturer
ISO 10218-1	Safety requirements for industrial robots	
ISO 10218-2	Safety requirements for integration of robots	Integrator
TS 15066	Collaborative robots technical specifications	
ISO 12100	Guidance for performing risk assessment	

## ISO 13849-1: 2008

- Standard describes
    - Safety related parts of control system
  - Purpose
    - Provide guidance of principles of design for the manufacturer of robot
  - Contains
    - Definitions of Safety Categories and Performance Levels (PL)
-  **UR5 and UR10 classifies as Performance Level d (PLd)**
- PLd is the second highest reliability classification, meaning that the safety function is extremely reliable

## ISO 10218-1: 2011

- Standard describes
  - Safety requirements for industrial robots
- Purpose
  - Provide guidance of principles of design for the manufacturer of robot
- 10218-1 is designed for traditional industrial robots
  - ISO 10218-1 Section 5.10 says: “Robots designed for collaborative operation shall provide a visual indication when the robot is in collaborative operation and shall comply with one or more of the requirements in 5.10.2 to 5.10.5
    - 5.10.2 Safety rated monitored stop
    - 5.10.3 Hand guiding
    - 5.10.4 Speed and Separation mode
    - 5.10.5 Power and force limiting by inherent design and control



**UR5 and UR10 comply with 5.10.5, as power and force limiting function is always active**

# 11 safety standards

## Safety system certified by TÜV

- Tested in accordance with
  - ISO 13849-1: 2008
  - ISO 10218-1: 2011

 All safety functions rated as Performance Level d (PLd)



## Your responsibility as integrator

- Risk assessment is mandatory
  - Recommended to comply with the following standards

Standard	Describes	Responsible
ISO 13849-1	Safety related parts of control system	Manufacturer
ISO 10218-1	Safety requirements for industrial robots	
ISO 10218-2	Safety requirements for integration of robots	Integrator
TS 15066	Collaborative robots technical specifications	
ISO 12100	Guidance for performing risk assessment	



**Identify risks and reduce them to appropriate level**

## ISO 10218-2: 2011

- Standard describes
  - Safety requirements for integration of robots
- Purpose
  - Provide guidance for integrators of industrial robot
  - ▶ **Consider the design of the installation where the robot is used**
- Considerations
  - Definitions of workspace, restricted space, collaborativ space
  - Location of controls and E-stops
  - Design of end effector
  - Movement and speeds of robot
  - Position of operator

## TS 15066 (*draft*)

- Standard describes
    - Collaborative robots technical specification
  - *Draft-guide only*
    - Currently in development
    - Scheduled for release 2015
  - Contains
    - Detailed set of guidelines for integrators when deploying collaborative robots
- Force related limits for collaborative robots**

## Summary

- MANDATORY

▶ Integrator **must** perform risk assessment

- NOT MANDATORY

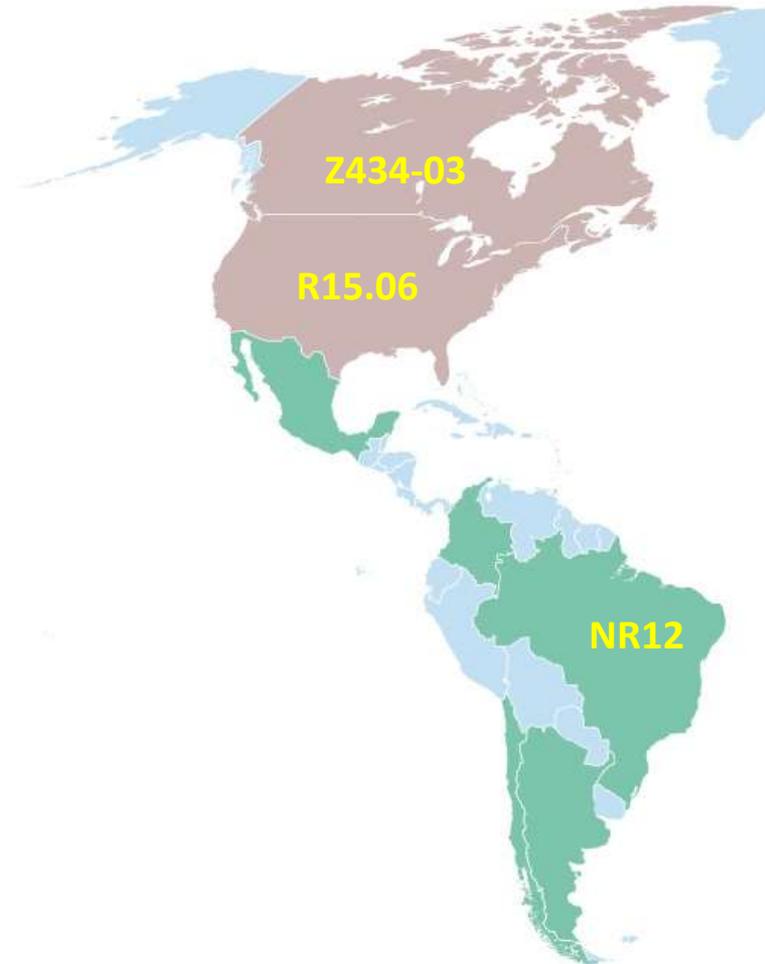
- Compliance with the standards
- It is recommended to comply with the standards!*
- In case of failure:
  - System complies with the standards
  - System does **not** comply with the standards

- » burden of proof lies with the prosecutor
- » burden of proof lies with the integrator



## Regional differences

- USA
  - ANSI/RIA R15.06: 2012
    - Harmonized with international ISO standards
    - ISO 10218-1 and ISO 10218-2 are combined into one document
- Canada
  - CAN/CSA-Z434-03: 2013
    - Harmonized with international ISO standards
    - Consists of ISO 10218-1 and ISO 10218-2 with regional deviations
- Brazil
  - NR 12
    - Standard is *not* harmonized with international ISO standards



Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

Safety standards

12

Adjustable safety

### Features

- Configurable safety settings
  - Advanced and patented safety system
  - Redundant safety
  - Password protected
- Purpose
  - Safety can be adjusted to the individual application
  - For ensuring no harm is made to personnel and peripheral equipment
- Risk assessment
  - Always perform risk assessment when installing a robot in an application
  - The configurable safety settings eases the risk assessment

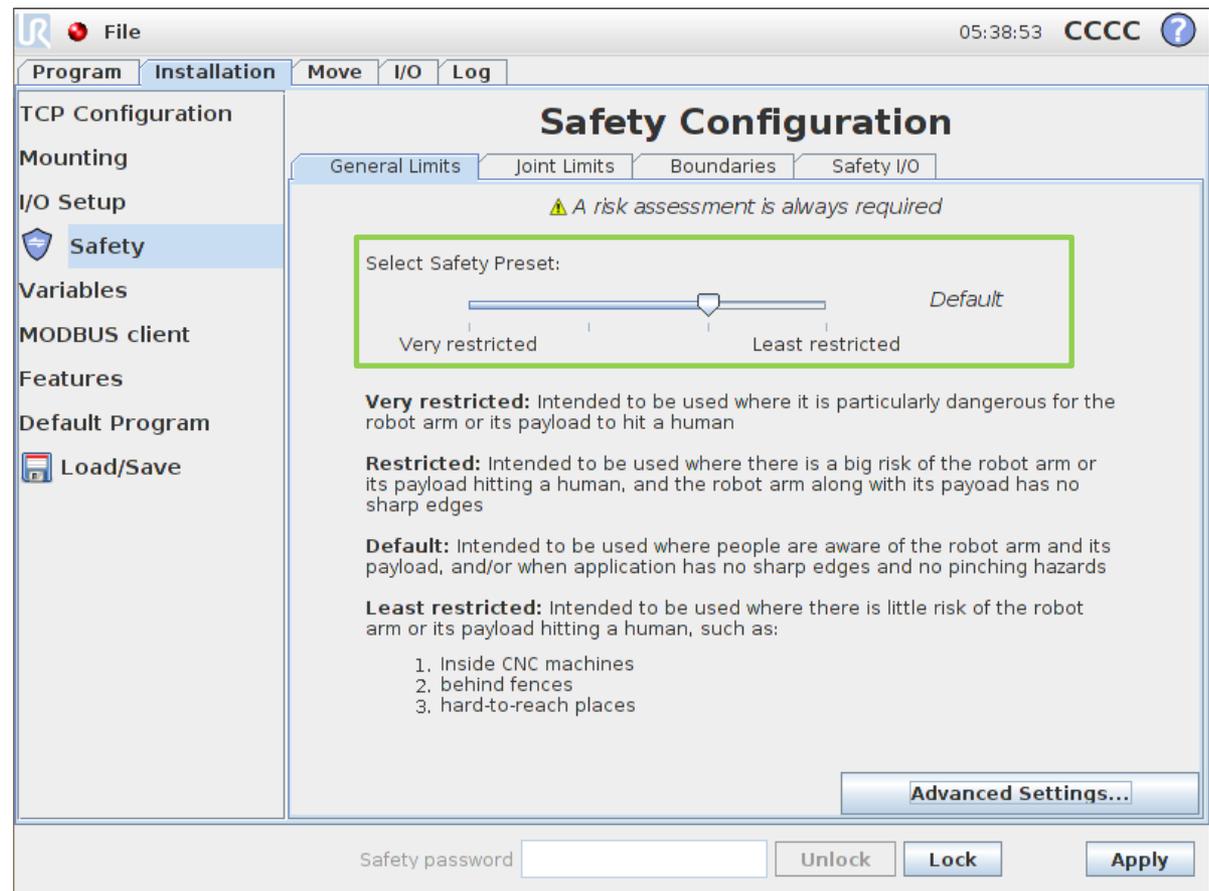


## Safety password

- Safety configuration is password protected
  - Lock
    - Protect safety configuration
  - Unlock
    - Enable modification of safety configuration
- Set safety password
  - Go to Setup robot
  - Select Set Password
  - Enter Password
  - Press Apply

## Basic settings

- Safety setting
  - Very restricted
  - Restricted
  - Default
  - Least restricted
- Safety configuration
  -  Synchronized
  -  Altered
  -  Invalid configuration
- Save configuration
  - Press "Apply"
  - Confirm settings



File 05:38:53 CCCC ?

Program Installation Move I/O Log

TCP Configuration  
Mounting  
I/O Setup  
Safety  
Variables  
MODBUS client  
Features  
Default Program  
Load/Save

### Safety Configuration

General Limits Joint Limits Boundaries Safety I/O

⚠ A risk assessment is always required

Select Safety Preset:

Very restricted | | | | | Least restricted Default

**Very restricted:** Intended to be used where it is particularly dangerous for the robot arm or its payload to hit a human

**Restricted:** Intended to be used where there is a big risk of the robot arm or its payload hitting a human, and the robot arm along with its payload has no sharp edges

**Default:** Intended to be used where people are aware of the robot arm and its payload, and/or when application has no sharp edges and no pinching hazards

**Least restricted:** Intended to be used where there is little risk of the robot arm or its payload hitting a human, such as:

1. Inside CNC machines
2. behind fences
3. hard-to-reach places

Advanced Settings...

Safety password [ ] [Unlock] [Lock] [Apply]

## Basic settings

- Default settings

Mode	Very restricted	Restricted	Default	Least restricted
Force (N)	100	120	150	250
Power (W)	80	200	300	1000
Speed (mm/s)	250	750	1500	5000
Momentum (kg·m/s)	5	10	25	100

- Limits are theoretical maximum values, if exceed robot will security stop

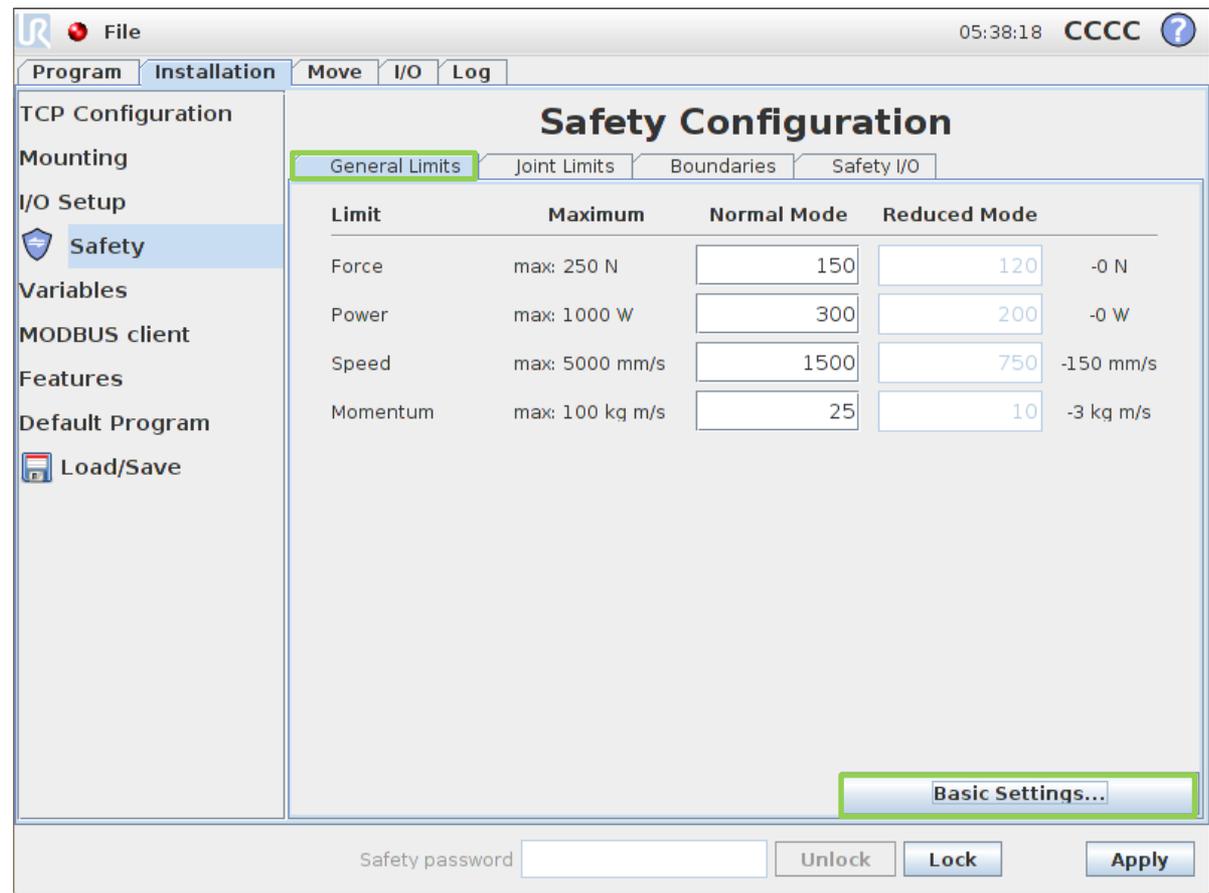
## Advanced settings

- Customize settings

- Force
- Power
- Speed
- Momentum

- Modes

- Normal Mode
- Reduced Mode
  - requires setup of safety input



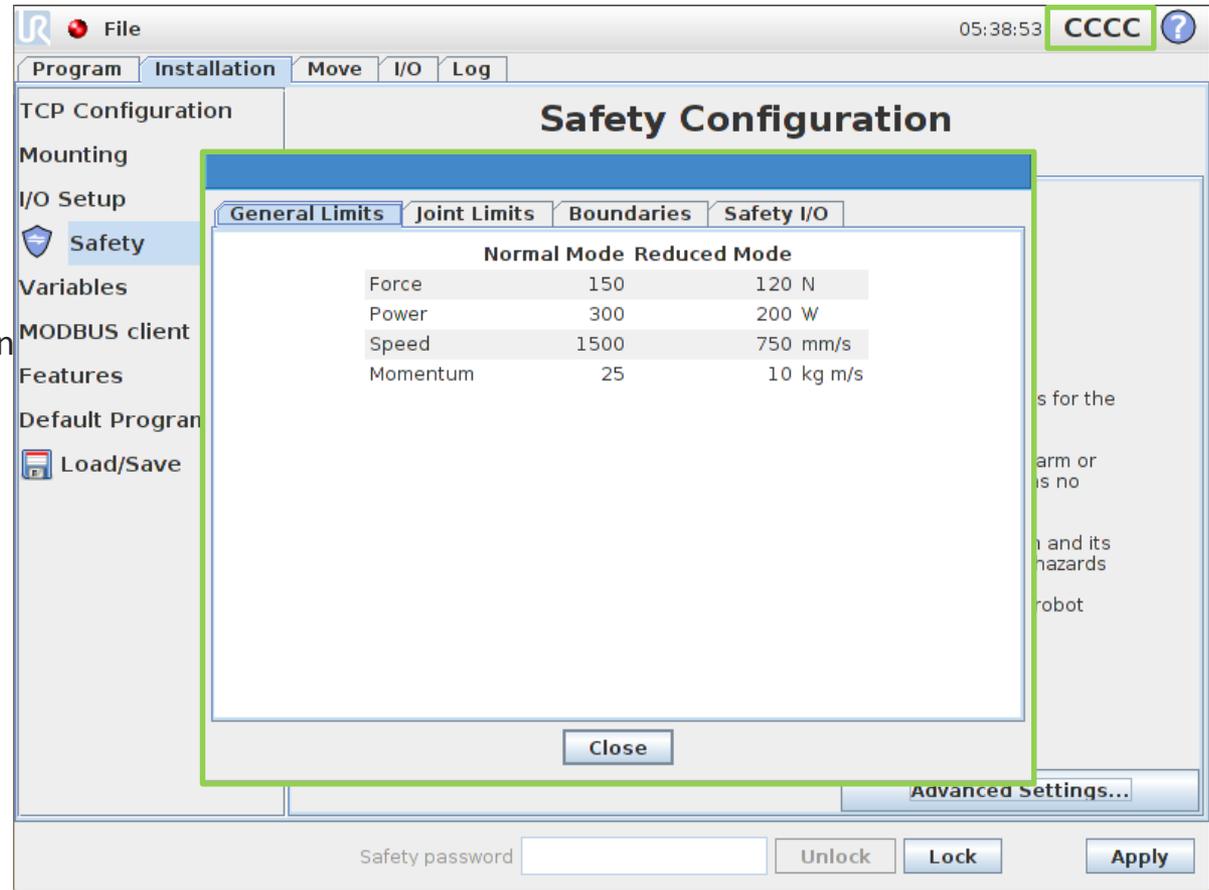
Limit	Maximum	Normal Mode	Reduced Mode	
Force	max: 250 N	<input type="text" value="150"/>	<input type="text" value="120"/>	-0 N
Power	max: 1000 W	<input type="text" value="300"/>	<input type="text" value="200"/>	-0 W
Speed	max: 5000 mm/s	<input type="text" value="1500"/>	<input type="text" value="750"/>	-150 mm/s
Momentum	max: 100 kg m/s	<input type="text" value="25"/>	<input type="text" value="10"/>	-3 kg m/s

### Safety modes

- Normal mode
  - Safety mode active by default
- Reduced mode
  - Active when robot TCP is positioned beyond *Trigger Reduced Mode Plane*
  - Active when using configurable input to trigger *Reduced Mode*
- *In case of violation:*
  - Recovery mode
    - Active when robot arm is in violation of one of the other modes
    - This mode allows robot arm to be manually adjusted until all violations has been solved
    - Not possible to run program when this mode is active
    - Joint position limits are disabled in Recovery Mode

## safety checksum

- Checksum
  - Visual indication of safety configuration
  - Indicated as colors and numbers
  - Checksum will change when safety configuration is altered



The screenshot shows the 'Safety Configuration' dialog box in the Universal Robots software. The dialog box is titled 'Safety Configuration' and has tabs for 'General Limits', 'Joint Limits', 'Boundaries', and 'Safety I/O'. The 'General Limits' tab is selected, showing a table of limits for Normal Mode and Reduced Mode. The table has the following data:

	Normal Mode	Reduced Mode
Force	150	120 N
Power	300	200 W
Speed	1500	750 mm/s
Momentum	25	10 kg m/s

At the bottom of the dialog box, there is a 'Close' button. Below the dialog box, there is a 'Safety password' field, 'Unlock' and 'Lock' buttons, and an 'Apply' button. The background window shows the 'Safety' configuration page with a 'CCCC' status indicator in the top right corner.

## Joint limits

- Maximum speed
  - Set max. speed for each joint
  
- Modes
  - Normal Mode
  - Reduced Mode
    - requires setup of safety input

File 05:39:33 CCCC ?

Program Installation Move I/O Log

TCP Configuration  
Mounting  
I/O Setup  
Safety  
Variables  
MODBUS client  
Features  
Default Program  
Load/Save

### Safety Configuration

General Limits **Joint Limits** Boundaries Safety I/O

Each of the following joint limits can be configured independently:

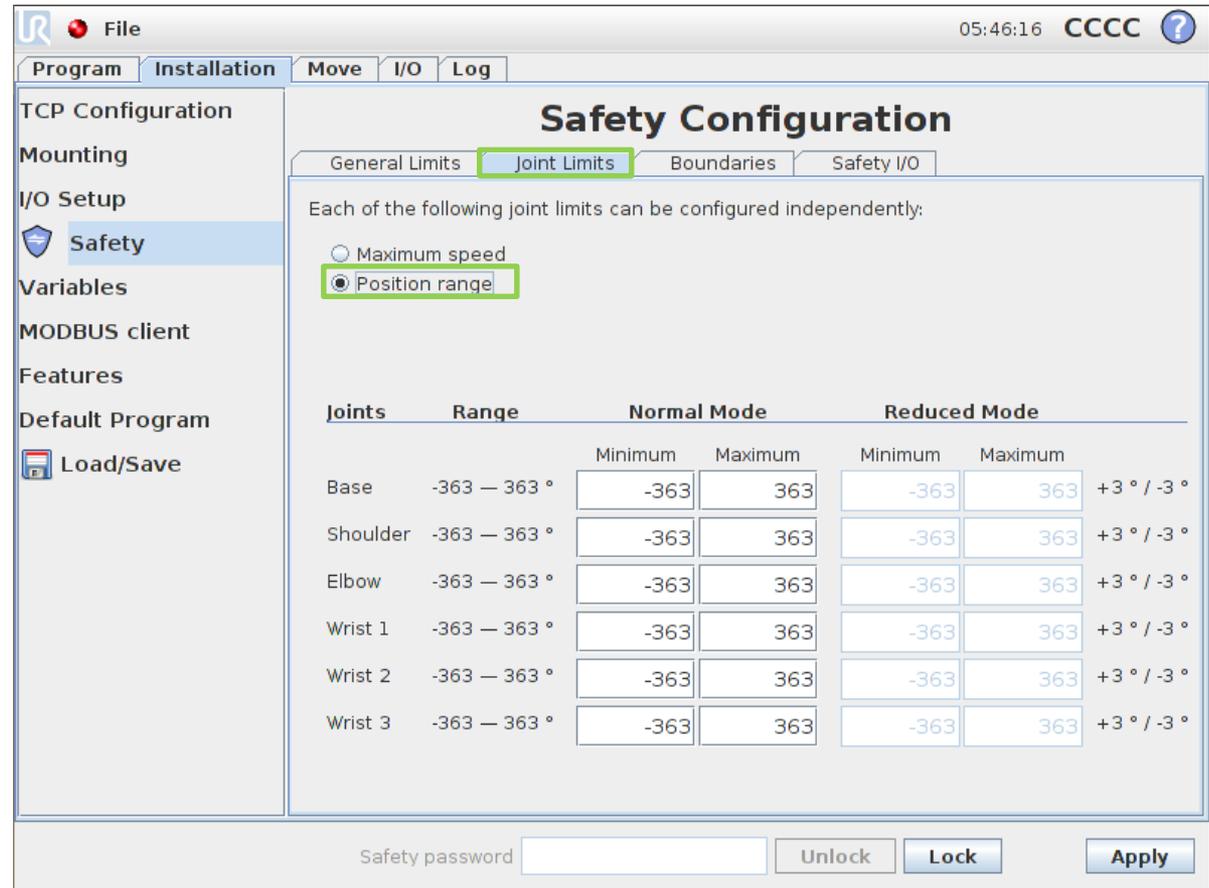
Maximum speed  
 Position range

Joints	Maximum	Normal Mode	Reduced Mode	
Base	max: 191 °/s	191	191	-11 °/s
Shoulder	max: 191 °/s	191	191	-11 °/s
Elbow	max: 191 °/s	191	191	-11 °/s
Wrist 1	max: 191 °/s	191	191	-11 °/s
Wrist 2	max: 191 °/s	191	191	-11 °/s
Wrist 3	max: 191 °/s	191	191	-11 °/s

Safety password

## Joint limits

- Position range
  - Set min. and max. range for each joint
  
- Modes
  - Normal Mode
  - Reduced Mode
    - requires setup of safety input



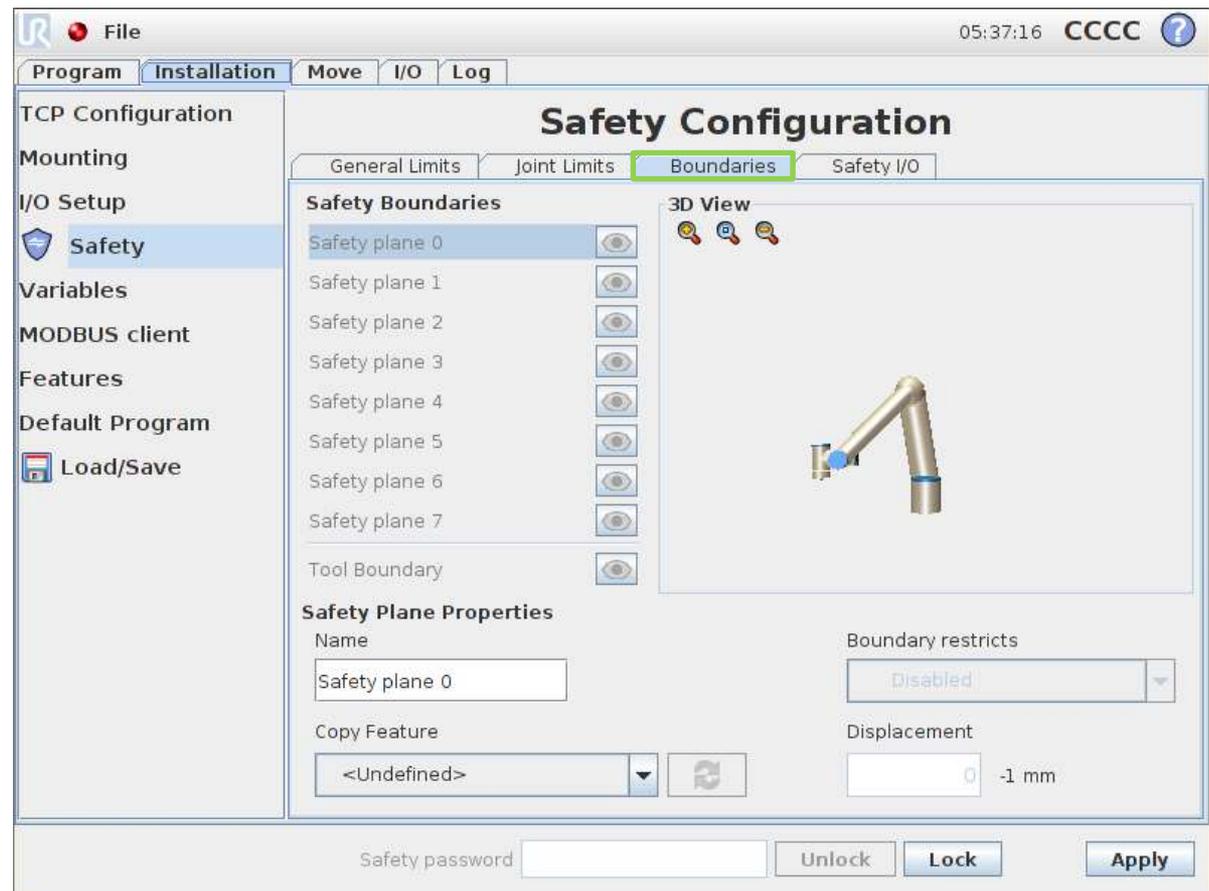
The screenshot shows the 'Safety Configuration' window with the 'Joint Limits' tab active. The 'Position range' radio button is selected. The table below shows the configuration for each joint.

Joints	Range	Normal Mode		Reduced Mode		
		Minimum	Maximum	Minimum	Maximum	
Base	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °
Shoulder	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °
Elbow	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °
Wrist 1	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °
Wrist 2	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °
Wrist 3	-363 — 363 °	-363	363	-363	363	+3 ° / -3 °

At the bottom of the window, there is a 'Safety password' field, 'Unlock' and 'Lock' buttons, and an 'Apply' button.

## Boundaries

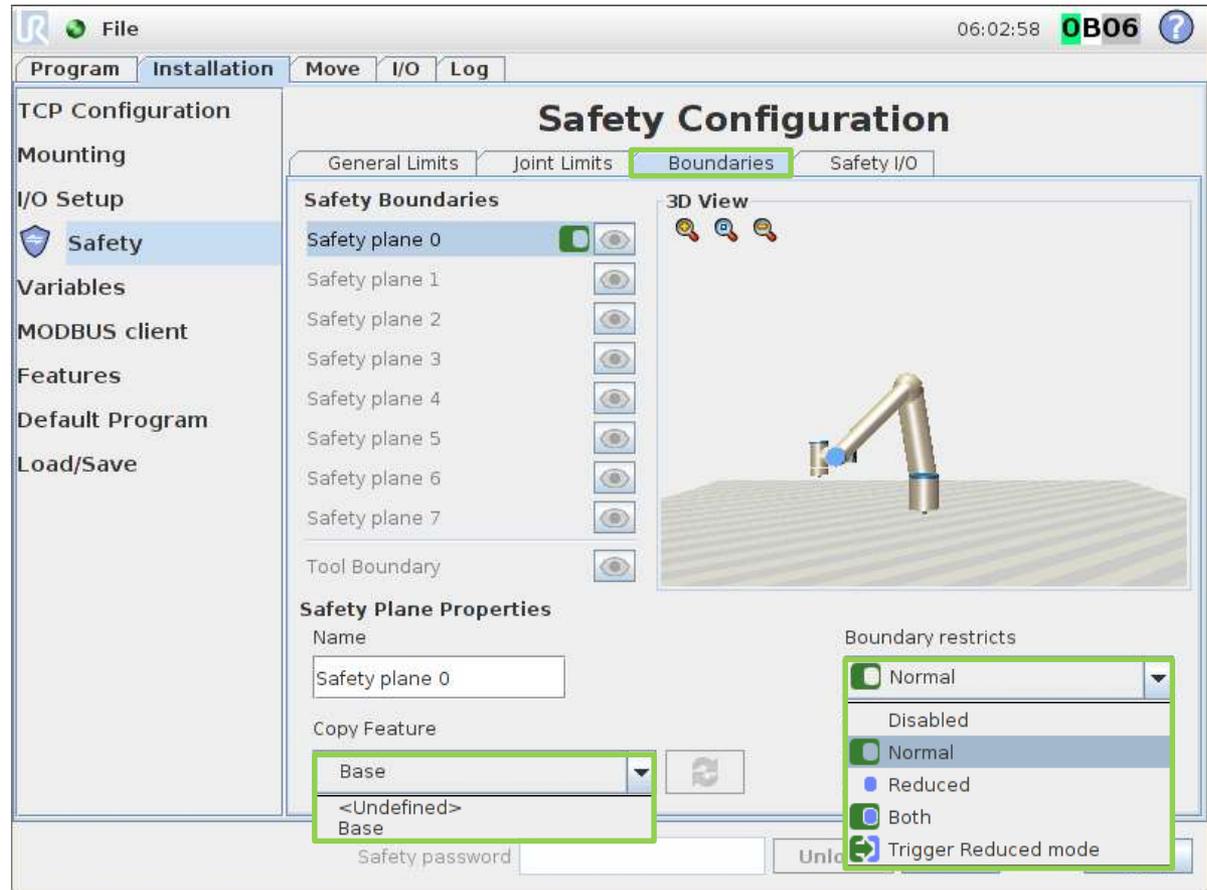
- Safety plane
  - Restrict allowed workspace
  - 8 planes can be defined
  - Active in both Teach and Run mode
  
- Plane can trigger *Reduced Mode* when TCP is entering plane



- **IMPORTANT:** Safety boundary defines a limit only for robot TCP, not overall limit for robot arm

## Boundaries

- Setup plane
  - Set Feature plane
    - Defines which plane to use
  - Set Safety Mode
    - Defines when safety plane is active
  - Displacement
    - Offsets the plane
  - Apply
    - Activates the configuration changes



## Safety Modes

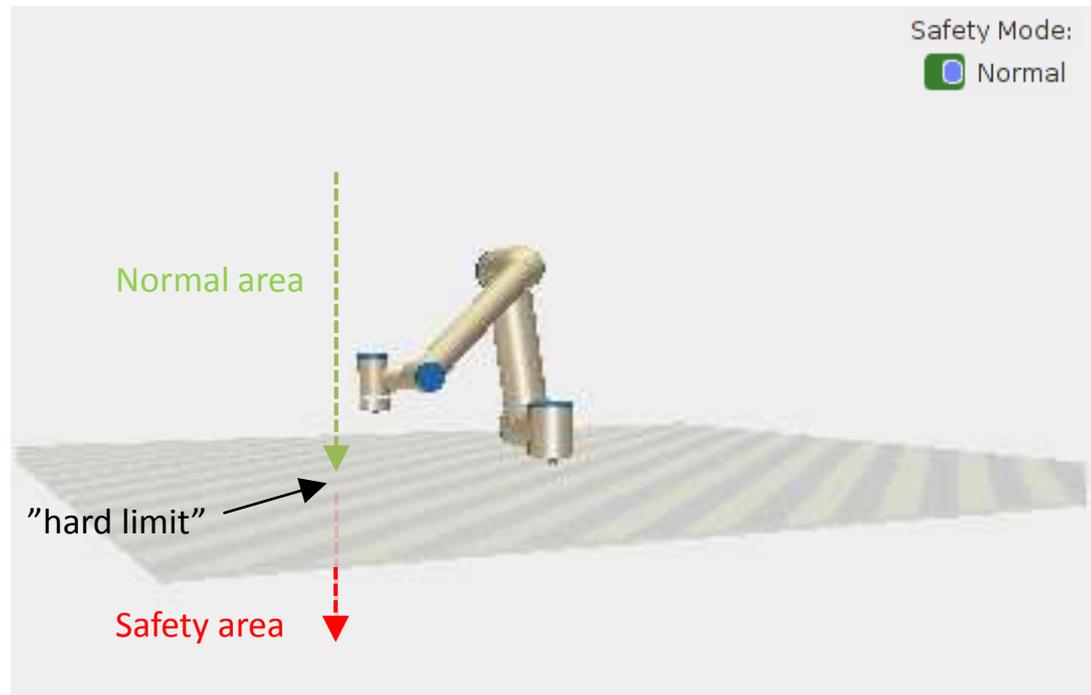
- Safety Modes behaviour

Safety Mode	Behaviour
Disabled	inactive
 Normal	acts as "hard limit" when in normal mode
 Reduced	acts as "hard limit" only if robot is in reduced mode
 Both	acts as "hard limit" at all times
 Trigger Reduced Mode	robot switches to reduced mode when TCP is entering plane

## Behaviour of safety boundary

### ■ Test

- Set Feature plane = *Base*
- Set Safety Mode = *Normal*
- Apply settings
- Test safety in teach mode
  - Move robot from Normal area towards Safety area



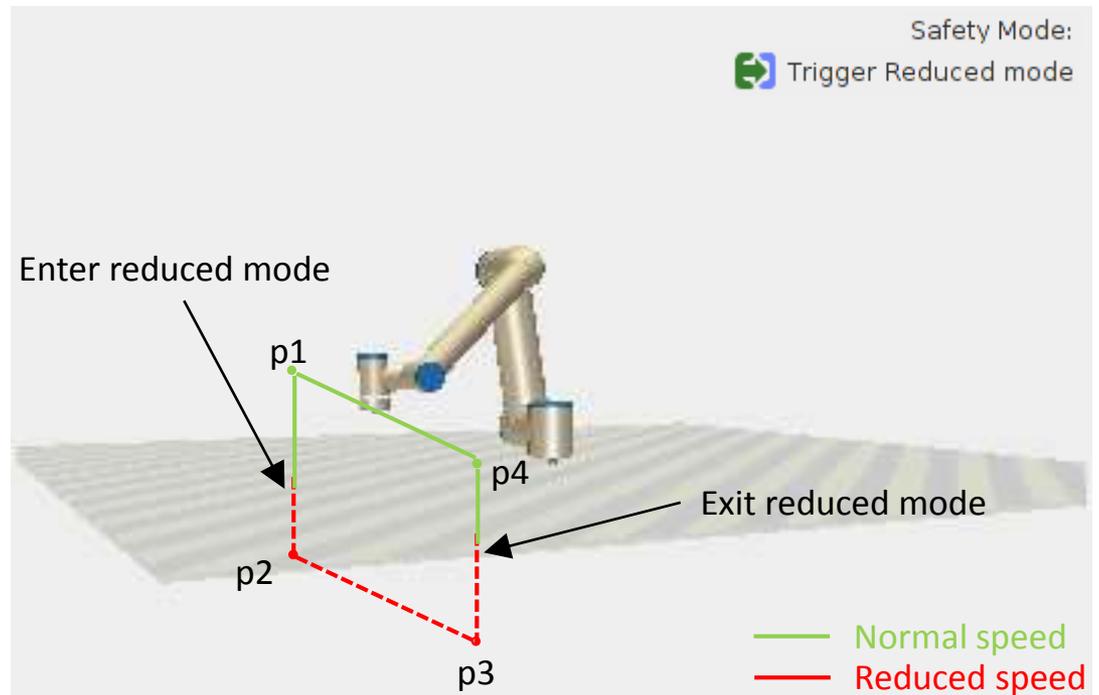
- Behaviour in Run mode
  - Program execution will be aborted, safety violation will popup

## Trigger reduced mode

### ■ Test

- Change Safety Mode to: *Trigger reduced mode*
- Set max. speed for reduced mode to: *50 mm/s*
- Apply settings
- Save as *safety.installation*

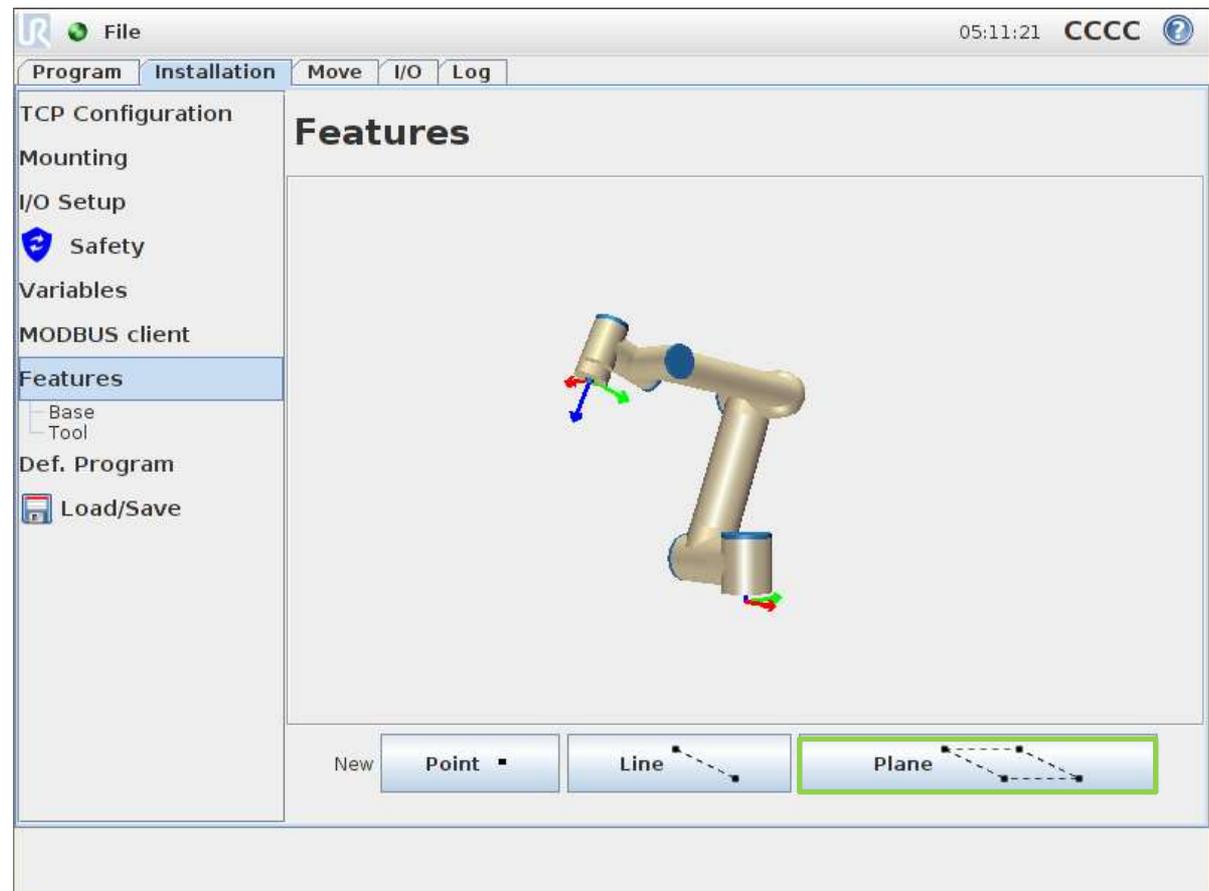
```
Robot Program
MoveL
  Waypoint_1
  Waypoint_2
  Waypoint_3
  Waypoint_4
```



- Save sample program as `trigger_reduced_mode.urp`

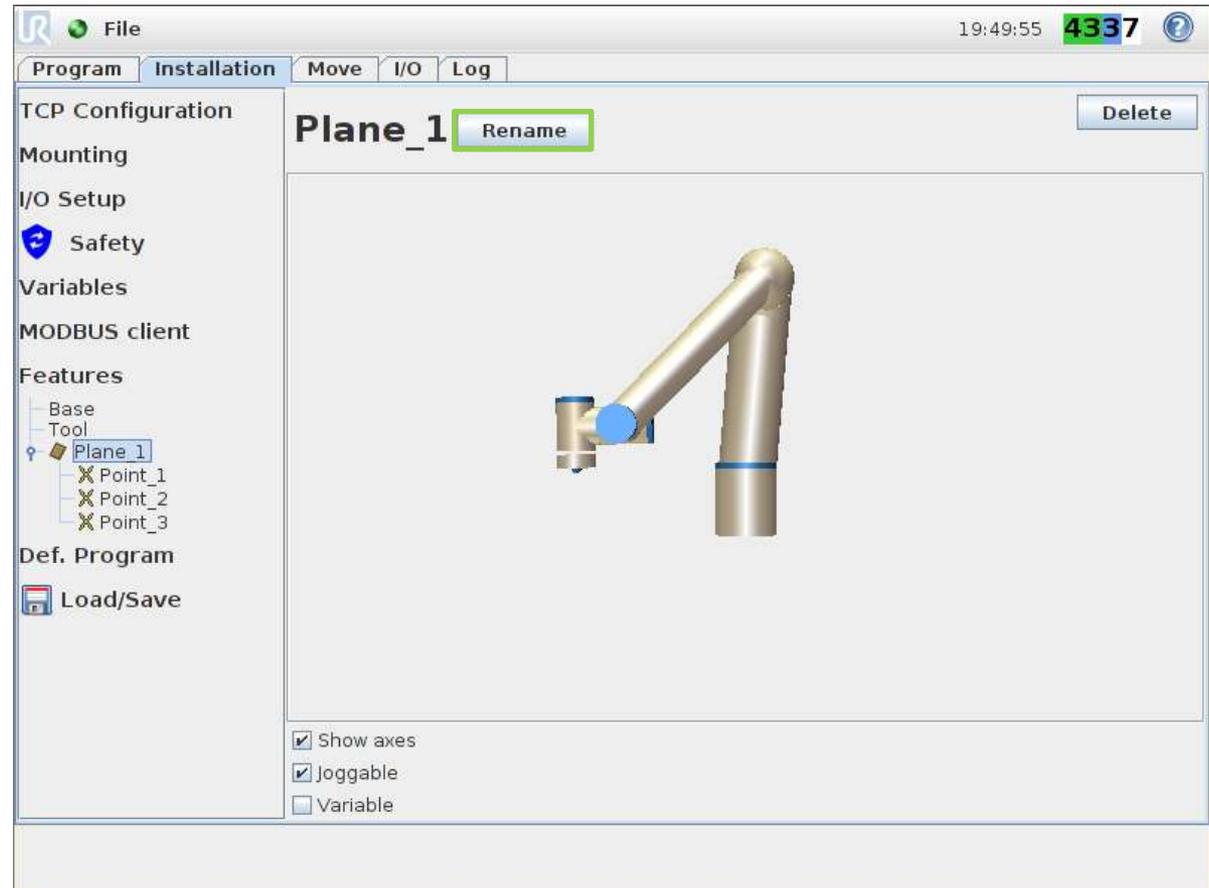
## User defined safety plane

- Features
  - In PolyScope a Plane is defined as a Feature
  - Multiple Features can be set
  - Set Feature as
    - Point
    - Line
    - Plane
- Add Feature
  - Plane



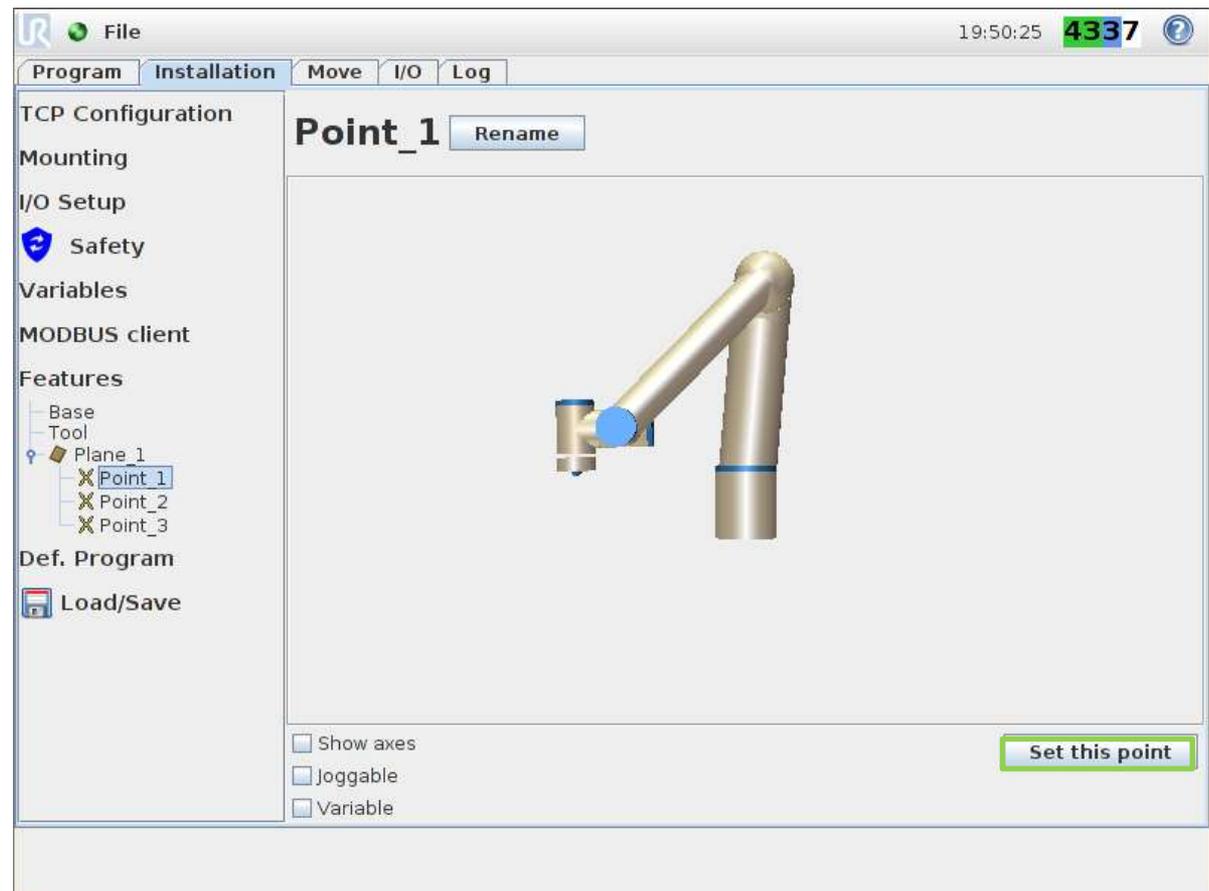
## User defined safety plane

- Options
  - Rename Feature
  - Delete Feature
- Parameters
  - Show axes
  - Joggable
  - Variable



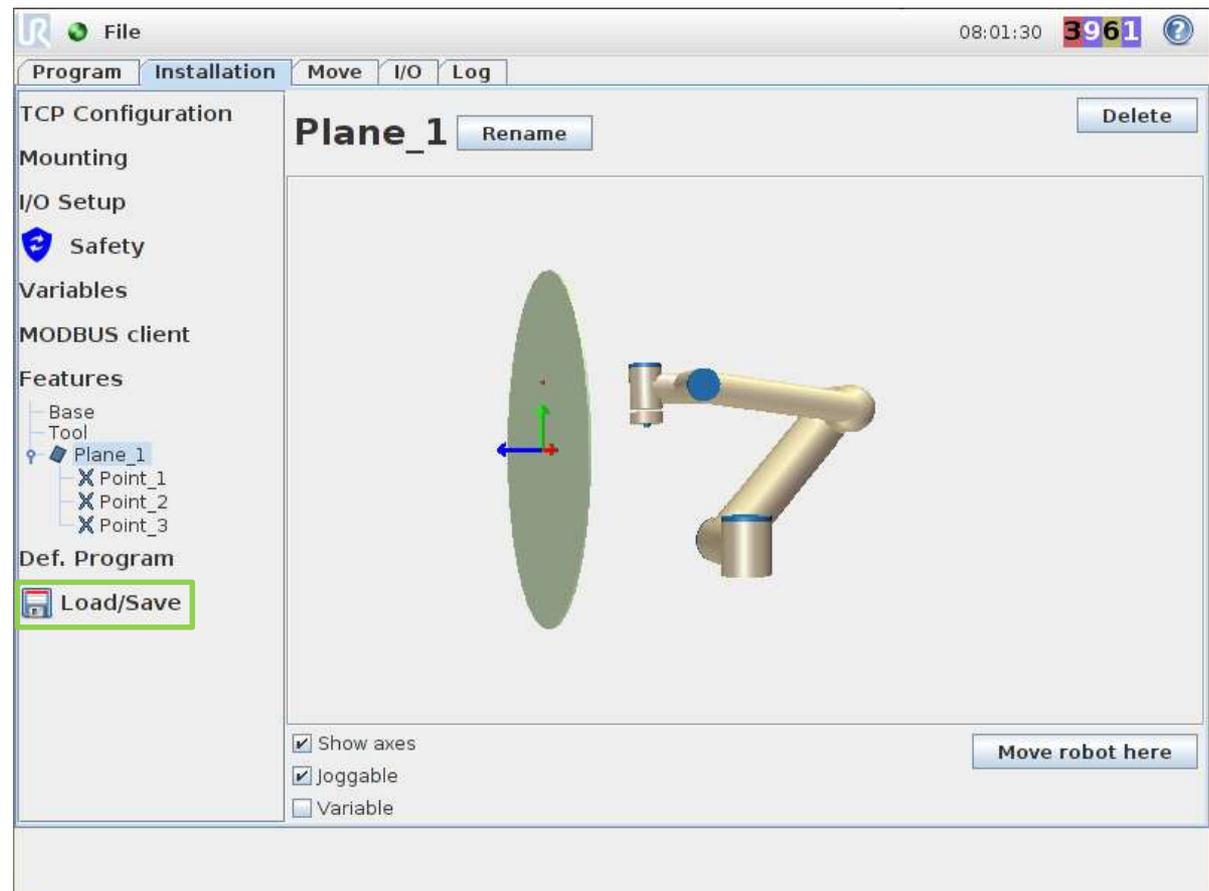
## User defined safety plane

- Setup Plane
  - Plane consists of three fixed waypoints
- Teach vertical plane
  - Point\_1 = origin
  - Point\_2 = Y-direction
  - Point\_3 = X-direction



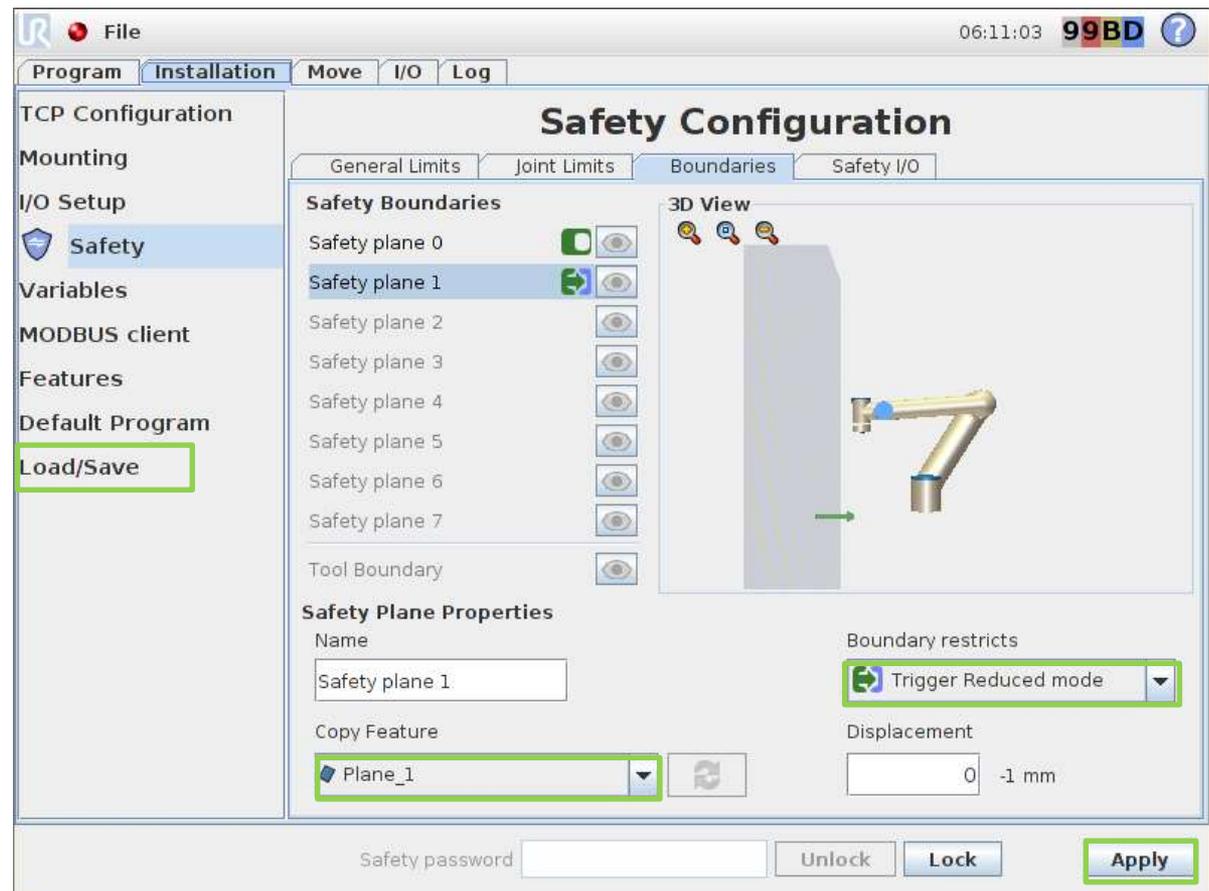
## User defined safety plane

- Graphical illustration of taught Plane
  - Save Installation



## User defined safety plane

- Setup new safety plane
  - Set Feature plane to: *Plane\_1*
  - Set Safety Mode to: *Trigger Reduced Mode*
  - Apply settings
  - Save as *safety.installation*



## Behaviour of user defined safety plane

### ■ Test

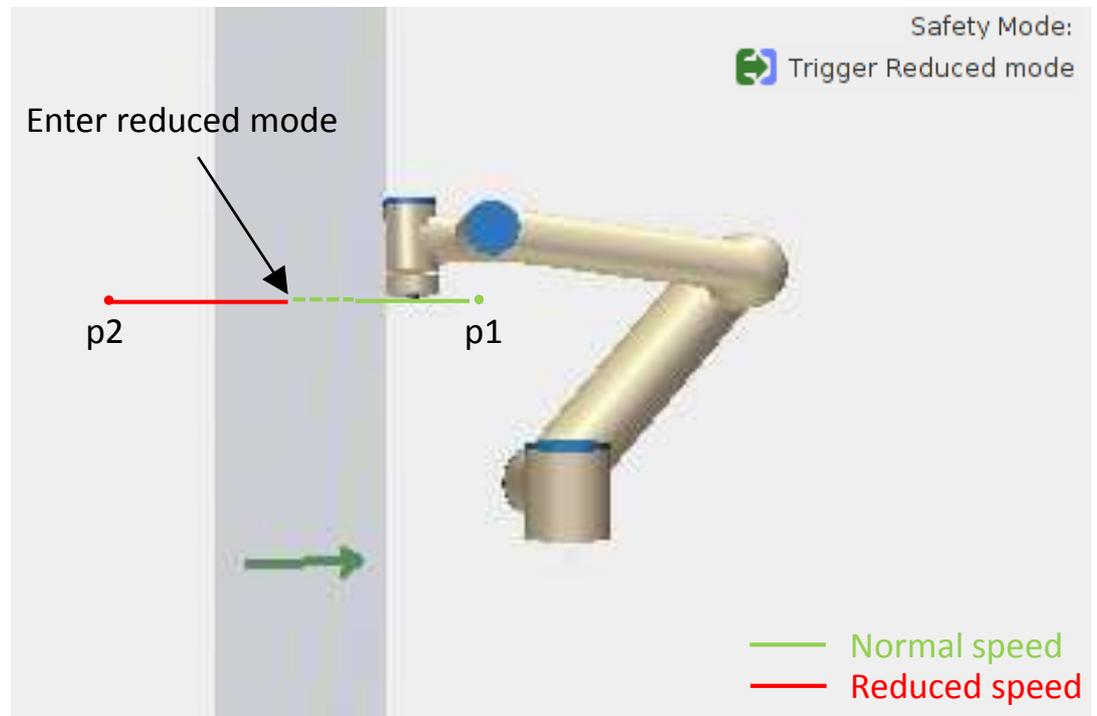
#### ■ Teach mode

- Move robot from Normal area towards Safety area

#### ■ Run mode

- Run sample program and verify TCP speed changes

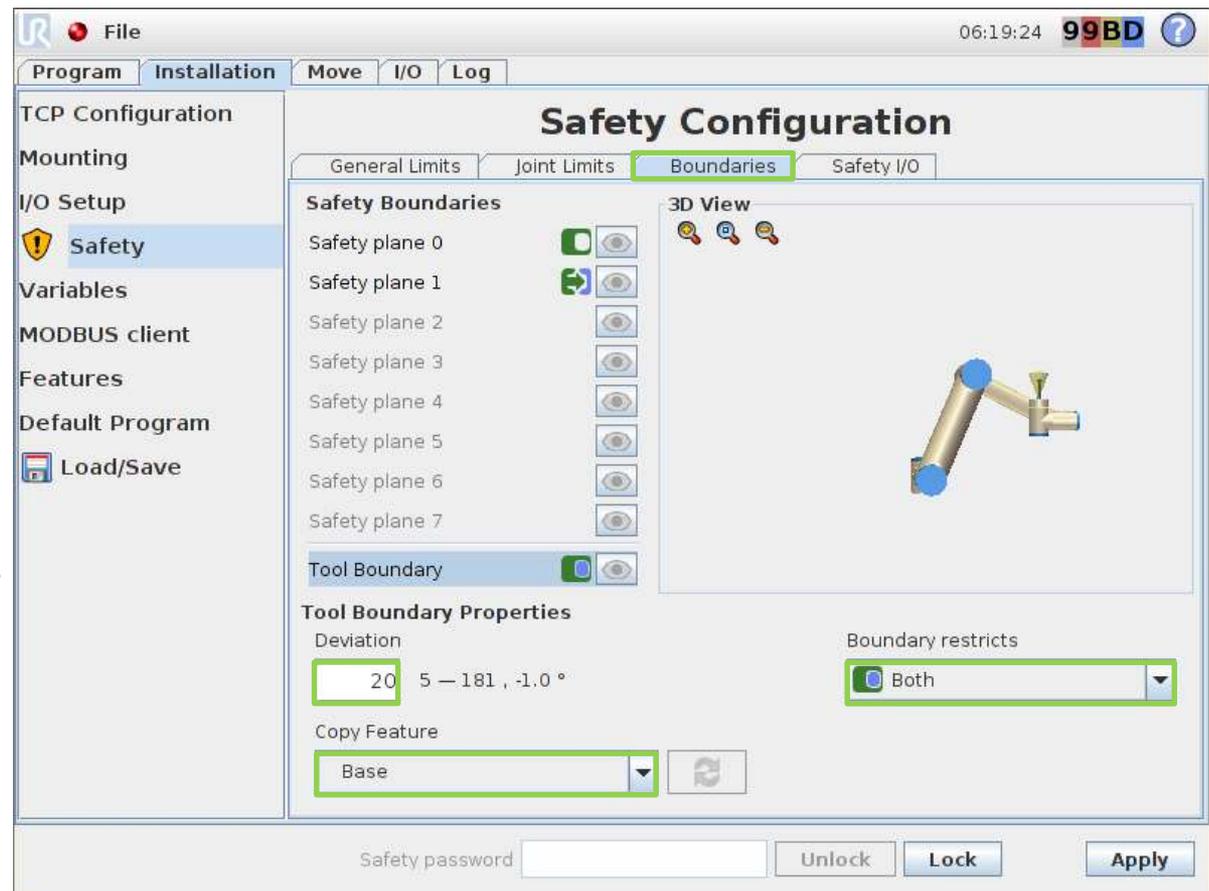
```
Robot Program  
MoveL  
  Waypoint_1  
  Waypoint_2
```



- Save sample program as trigger\_reduced\_mode\_feature.urp

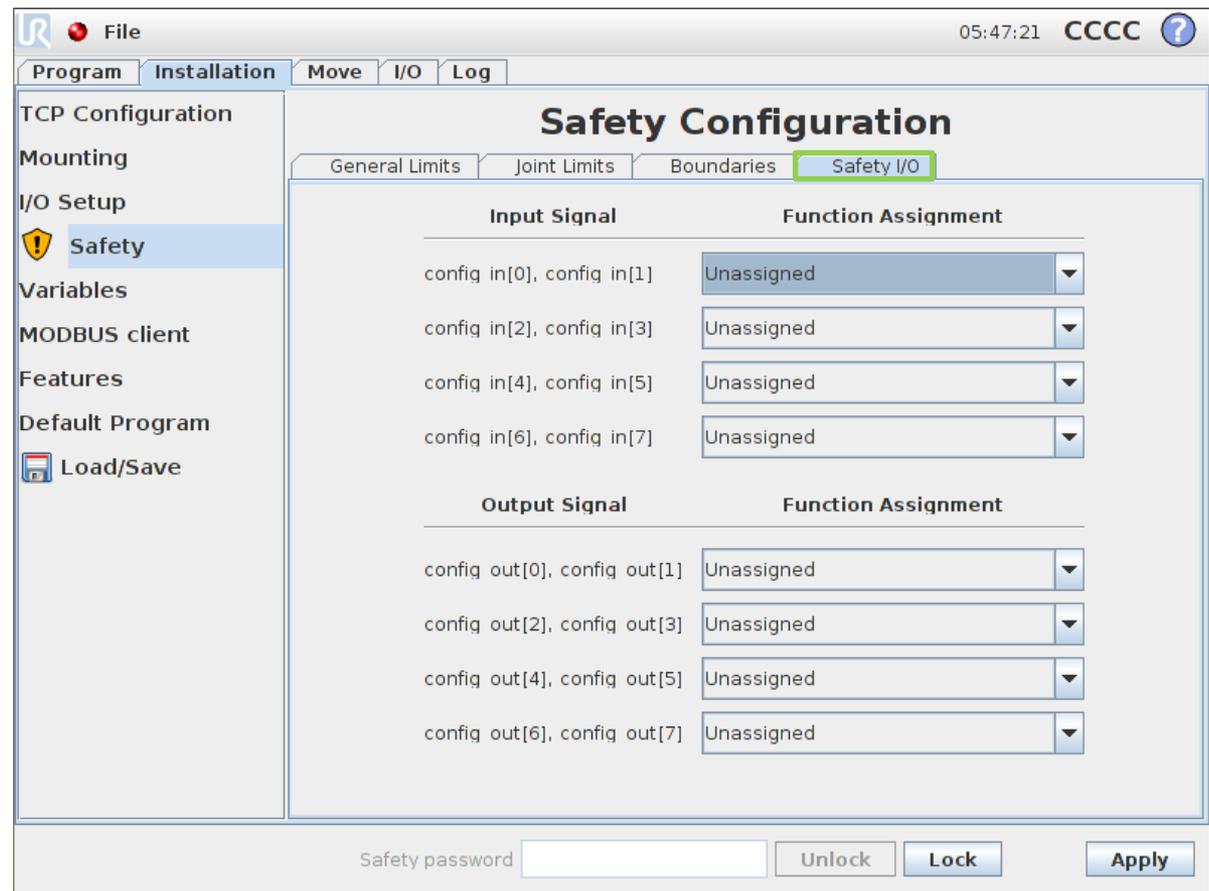
## Tool boundary

- Restrict angular deviation of TCP
  - Set max. TCP deviation in respect to selected Feature
- Test
  - Set Feature plane: *Base*
  - Set Deviation:  $25^\circ$
  - Set Safety Mode to: *Normal & Reduced Mode*
  - Apply
  - Test in Teach mode



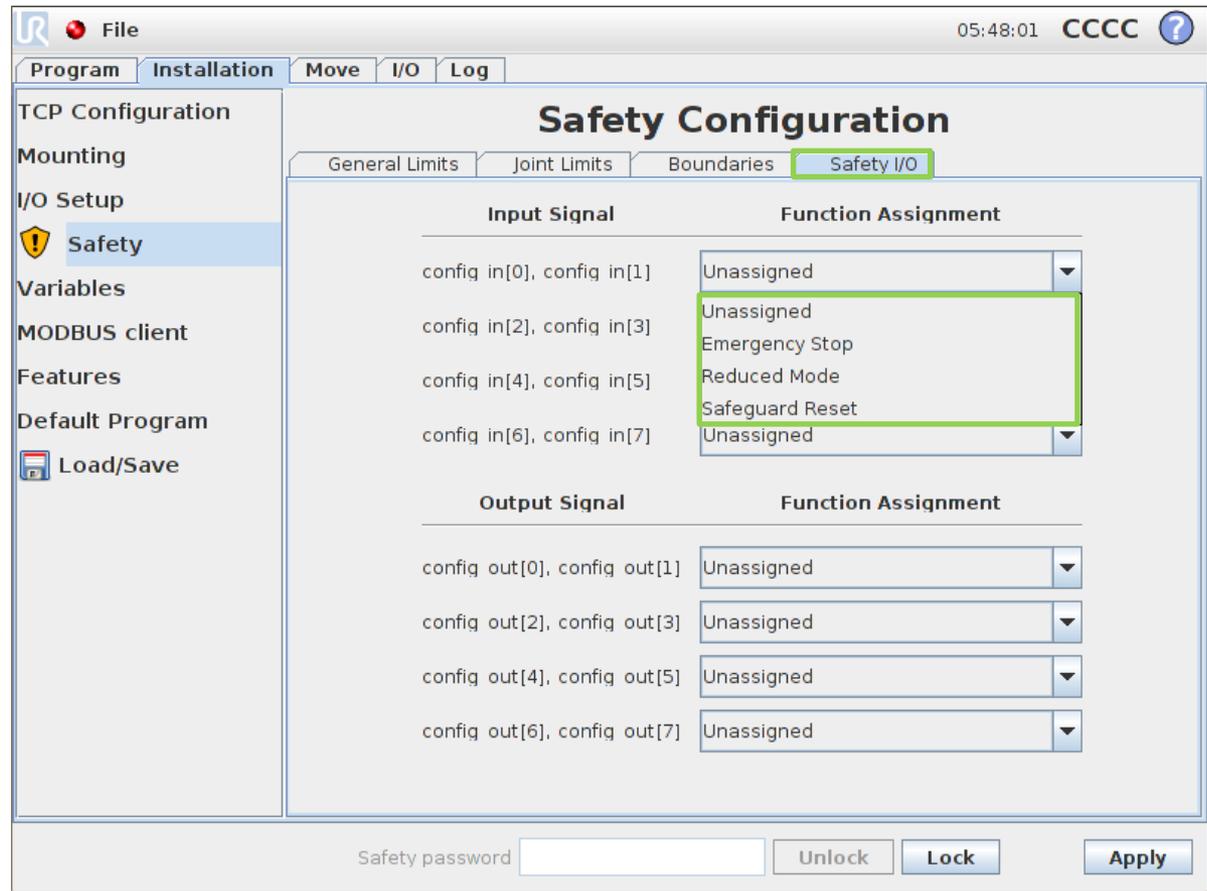
## Safety I/O

- Safety functions
  - Safety functions can be assigned to the Configurable I/O's
- All functions are redundant
  - Two signals for each function
  - Category 3, PLd
- Configurable I/O's
  - Digital inputs
  - Digital outputs



## Safety inputs

- Safety inputs
  - Emergency stop
    - For connecting external EMG-button or safety PLC
  - Reduced Mode
    - Low signal: robot operates in normal mode
    - High signal: robot operates in reduced mode
  - Safeguard Reset
    - If Safeguard is hardwired to Safety Control Board, the safeguard can be reset with this signal



UR File 05:48:01 CCCC ?

Program Installation Move I/O Log

TCP Configuration

Mounting

I/O Setup

! Safety

Variables

MODBUS client

Features

Default Program

Load/Save

### Safety Configuration

General Limits Joint Limits Boundaries **Safety I/O**

Input Signal	Function Assignment
config in[0], config in[1]	Unassigned
config in[2], config in[3]	Unassigned Emergency Stop Reduced Mode Safeguard Reset
config in[4], config in[5]	Unassigned
config in[6], config in[7]	Unassigned

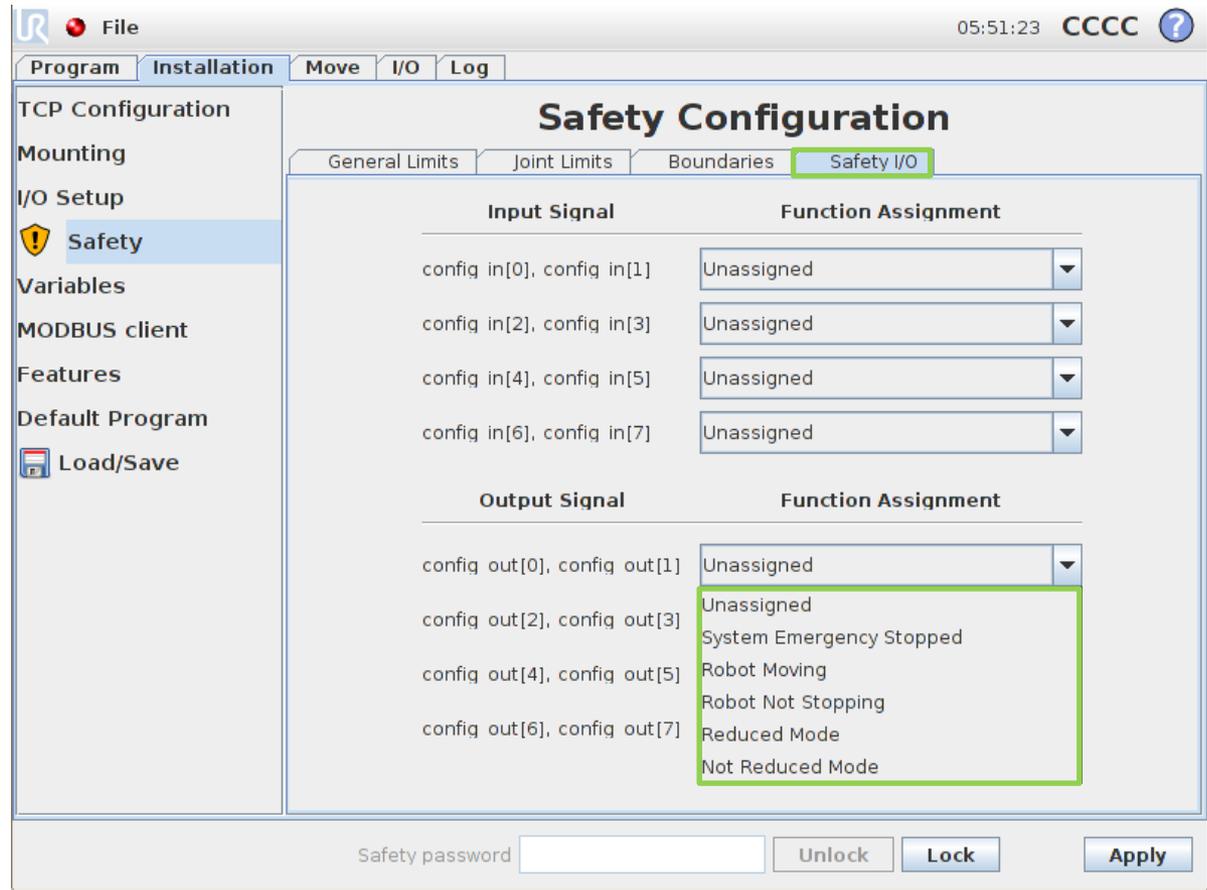
  

Output Signal	Function Assignment
config out[0], config out[1]	Unassigned
config out[2], config out[3]	Unassigned
config out[4], config out[5]	Unassigned
config out[6], config out[7]	Unassigned

Safety password

## Safety outputs

- Safety outputs
  - System Emergency Stop
    - HI: normal mode
    - LO: emergency stopped
  - Robot Moving
    - HI: robot not moving
    - LO: robot moving
  - Robot Not Stopping
    - HI: robot requested to stop signal is high until robot stops
    - LO: no stop request
  - Reduced Mode
    - HI: normal mode
    - LO: reduced mode
  - Not Reduced Mode
    - Inverse state of reduced mode



The screenshot shows the 'Safety Configuration' window with the 'Safety I/O' tab active. The interface is divided into 'Input Signal' and 'Output Signal' sections, each with a 'Function Assignment' dropdown menu. The 'Output Signal' dropdown for 'config out[2], config out[3]' is expanded, showing the following options: Unassigned, System Emergency Stopped, Robot Moving, Robot Not Stopping, Reduced Mode, and Not Reduced Mode. The 'Safety password' field and 'Unlock', 'Lock', and 'Apply' buttons are visible at the bottom.

Input Signal	Function Assignment
config in[0], config in[1]	Unassigned
config in[2], config in[3]	Unassigned
config in[4], config in[5]	Unassigned
config in[6], config in[7]	Unassigned

Output Signal	Function Assignment
config out[0], config out[1]	Unassigned
config out[2], config out[3]	Unassigned System Emergency Stopped Robot Moving Robot Not Stopping Reduced Mode Not Reduced Mode
config out[4], config out[5]	Unassigned
config out[6], config out[7]	Unassigned

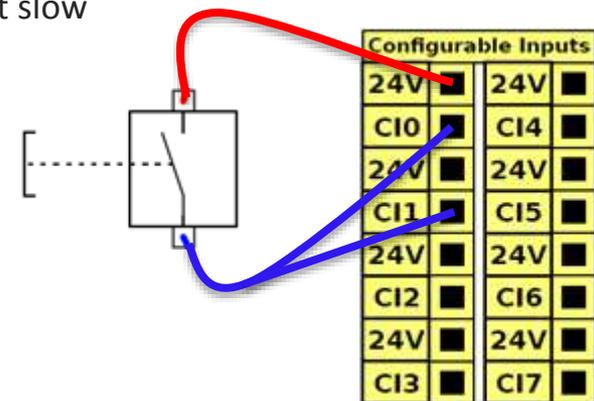
### Lab exercise part 1

- Create a program using safety zones and safety inputs
- Setup Safety Features:
  - Select Restricted mode under the General Limits tab.
  - Create a horizontal safety plane at the base (copy base feature) with a displacement of -50mm and use the Both restriction profile, so the robot will not crash into the table it is mounted on.
  - Create a vertical safety plane at  $x=400\text{mm}$  and select the Trigger Reduced Mode profile, as if this part of the workspace is shared with a human.
- Apply the changes and move the robot around by hand in teach mode to feel the effect of the safety features

## 12 Adjustable safety

### Lab exercise part 2

- Setup a button to trigger reduced mode
  - Connect the button we used previously to both CI0 and CI1
  - Select Reduced Mode for config\_in[0] and [1] under the Safety I/O tab
  - Write a simple program with 2+ waypoints and run it.
  - Press the button to enter reduced mode and watch the robot slow down.



## Examination

- You have 30 minutes to answer the questions.
- Questions can have multiple answers, so please review all the choices carefully before answering.
- Indicate the right answer by ticking or circling the correct choices and filling in the blanks clearly where required.
- Feel free to use any material available to you and avoid discussing with other examinees.

- Good luck!



## Examination results

- 1. What settings on the “Installation” screen need to be verified before running any program on the robot?
  - a.  MODBUS client setup
  - b.  TCP, payload and Mounting settings
  - c.  Safety Settings for CB3
  - d.  All of the above
  - e.  None of the above
  
- 2. What is the Set command used for?
  - a.  Set Digital or Analog Outputs
  - b.  Set MODBUS register output values
  - c.  Set and reset Payload during pick-up and drop-off
  - d.  All of the above
  - e.  None of the above
  
- 3. Which options below enable instantaneous response to a Digital Input State change?
  - a.  Use the "Event" function
  - b.  Enable "Check-expression continuously" check-box under If-Else
  - c.  Enable "Check-expression continuously" check-box under Loop
  - d.  All of the above
  - e.  None of the above
  
- 4. How would you call another program from within your current program?
  - a.  Use the “Folders” option to invoke the program
  - b.  Use the “Sub-Program” option to call the program
  - c.  Use the “Script code” button
  - d.  All of the above
  - e.  None of the above

## Examination results

- 5. What is an installation variable?
  - a.  Global variable available across all programs
  - b.  Variable that retains value even on Power Down
  - c.  Installation variable?? There's no such thing
  - d.  (a) and (b)
  - e.  (c) or (d)
  
- 6. What is the possible cause for a "Force limit protective stop" alarm?
  - a.  The robot ran into an obstruction
  - b.  The robot has incorrect TCP, payload and mounting settings
  - c.  Too high acceleration settings
  - d.  All of the above
  - e.  None of the above
  
- 7. Why do we need to "Set" and "Reset" TCP payload on a UR robot?
  - a.  The motor tuning parameters are dynamically calculated based on payload
  - b.  Incorrect payload affects stability of robot
  - c.  Don't need to, Maxing out payload works fine
  - d.  (a) and (b)
  - e.  (c) and (d)
  
- 8. How would you create a variable that accepts and stores input from an operator?
  - a.  Use the "Assignment" button and change settings to "Operator"
  - b.  Use the initialize variables option
  - c.  Use the installation variables option
  - d.  All of the above
  - e.  None of the above

## Examination results

- 9. What is the possible cause for a "Torque limit violation" alarm?
  - a.  The robot has incorrect TCP, payload and mounting settings
  - b.  The robot ran into an obstruction
  - c.  Too high acceleration settings
  - d.  (a) or (c)
  - e.  This error does not exist
  
- 10. How do I create a variable that holds floating point values?
  - a.  Use assignments tab to create variable
  - b.  Rename variable name to "floating point"
  - c.  Create and Initialize variable with floating point value
  - d.  All variables are floating points by default
  - e.  None of the above
  
- 11. What is a pose variable and how is it represented?
  - Solution :- **It is a position saved as a variable containing six values, which can be adjusted programmatically.**  
**Represented like: p[x, y, z, rx, ry, rz]**
  
- 12. What is the difference between a MOVL and a MOVP ?
  - a.  There is no difference
  - b.  You can set a blend radius in a MOVP but not on MOVL
  - c.  MOVP maintains joint speed throughout the tool path, MOVL does not
  - d.  MOVP maintains TCP speed throughout the tool path, MOVL does not
  - e.  None of the above

## Examination results

- 13. How do you trace a curved path ?
  - a.  Break the curve into smaller curves and use Circle Move under the MOVP command
  - b.  Use the "movec" function after a "movep" function in the UR script language
  - c.  Not possible on this robot
  - d.  (a) or (b)
  - e.  (b) or (c)
  
- 14. What safety standards do UR comply with?
  - a.  ISO 10218 Sections 1 and 2
  - b.  ANSI/RIA R15.06-2012
  - c.  CAN/CSA Z434-2003(R2013)
  - d.  All of the above
  - e.  None of the above
  
- 15. What makes the UR robot collaborative?
  - a.  It can't smile so it's not collaborative
  - b.  It can be speed limited
  - c.  It is power and force limited
  - d.  Need Risk assessment to verify collaborative operation
  - e.  (b) (c) and (d)
  
- 16. How do you monitor a variable in parallel to the main program?
  - a.  Use an "Event" command
  - b.  Use a "Before Start Sequence"
  - c.  Use an "Assignment" command within a "Thread" function
  - d.  All of the above
  - e.  None of the above

## Examination results

- 17. A risk assessment is only required when a human will be working within the workspace of the robot.
  - a.  True
  - b.  False
  
- 18. The following features have been added in the CB3:
  - a.  Configurable safety settings
  - b.  New hubcaps
  - c.  True absolute encoders
  - d.  Mounting/Payload settings check monitor
  - e.  (a) (c) and (d)
  
- 19. In the CB3, safety boundaries can NOT be configured to which of the following?
  - a.  Using user-defined feature planes
  - b.  Using complex shapes such as ellipses and curved surfaces
  - c.  To offset a given plane
  - d.  To trigger reduced mode
  - e.  None of the above
  
- 20. You should zero calibrate your robot when:
  - a.  A joint is replaced
  - b.  An update to the joint firmware is made
  - c.  A new robot is shipped from the factory and installed
  - d.  When the robot reports "Force Limit Protective Stop"
  - e.  None of the above

## List of sample programs

- Sample programs
  - movej.urp
  - movel.urp
  - movel\_with\_blend.urp
  - movep.urp
  - movec.urp
  - movel\_with\_relative\_waypoint.urp
  - wait.urp
  - set.urp
  - popup.urp
  - pick\_and\_place.urp
  - loop.urp
  - loop\_interrupt.urp
  - call\_sub.urp
  - if.urp
  - if\_else.urp
  - var\_bool.urp
  - var\_counter.urp
  - var\_operator\_input.urp
  - inst\_var\_operator\_input.urp
  - thread.urp
  - event.urp
  - force\_feedback.urp
  - script\_line.urp
  - force\_simple.urp
  - pallet.urp
  - seek\_destack.urp
  - modbus.urp
  - trigger\_reduced\_mode.urp
  - trigger\_reduced\_mode\_feature.urp
  - default.installation
  - safety.installation