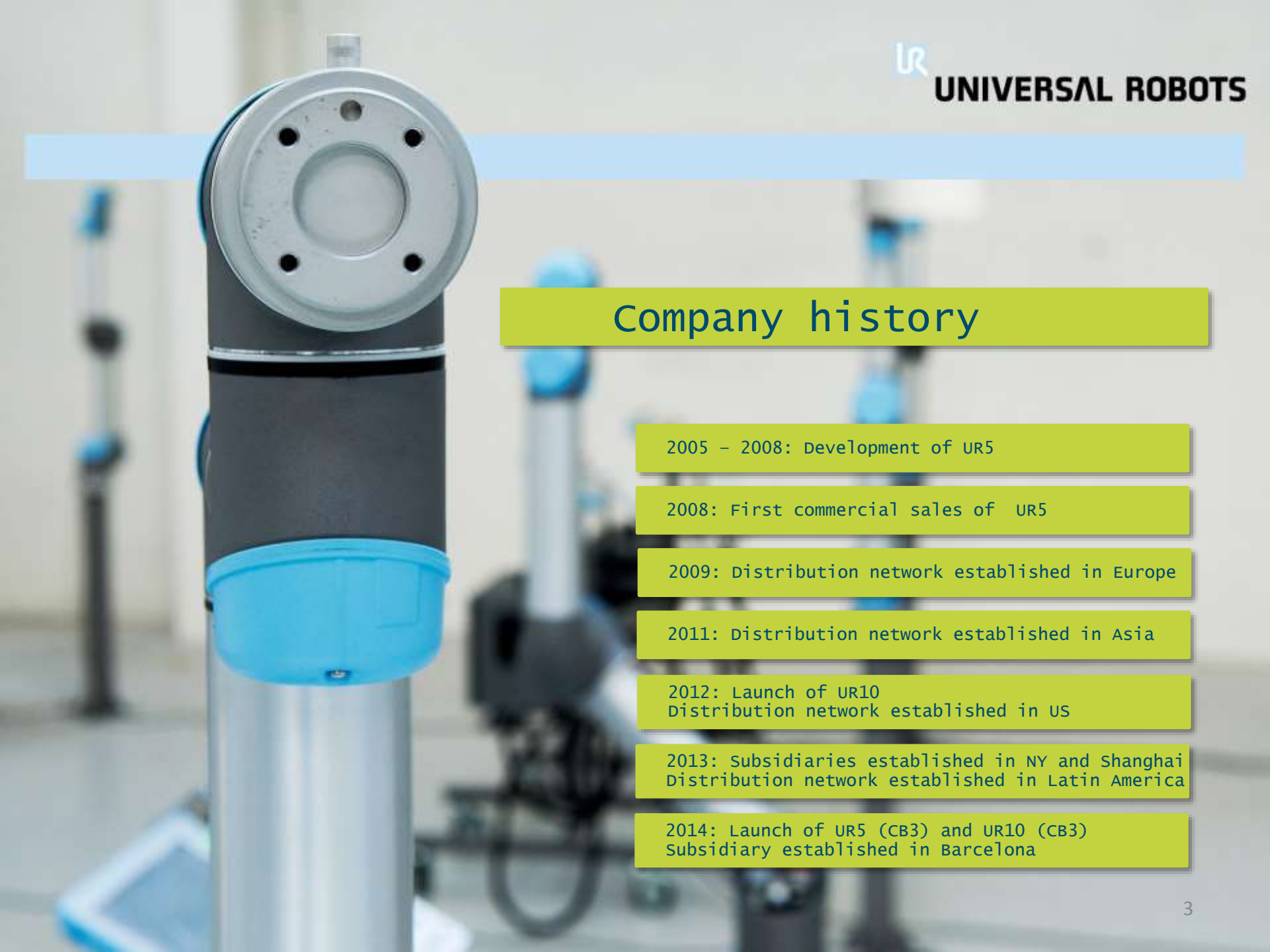

# Basic Training


# Company history



UNIVERSAL ROBOTS

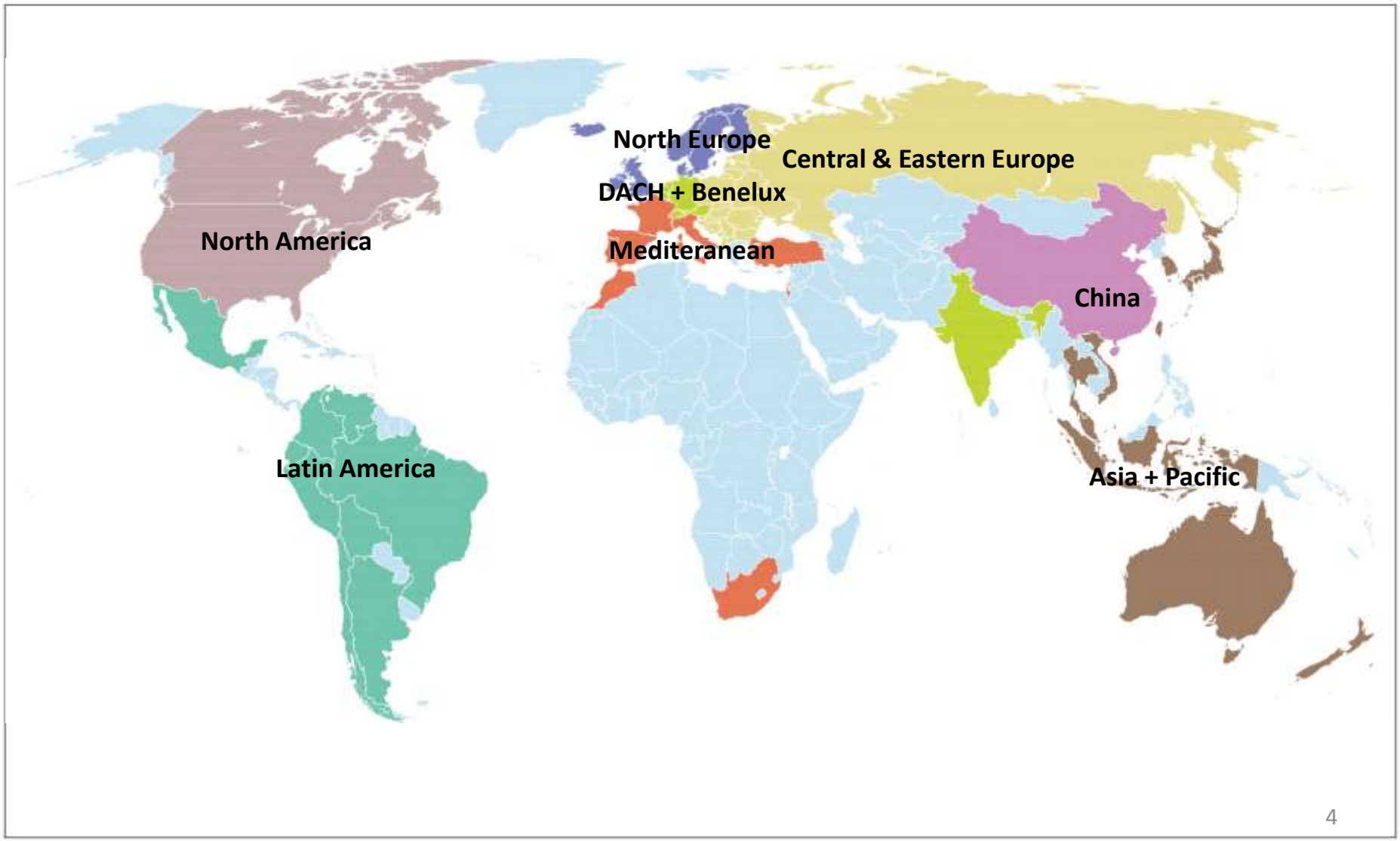
# Company history

- 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

# Company history

UNIVERSAL ROBOTS

## Global distribution 2014 regions




# Welcome to

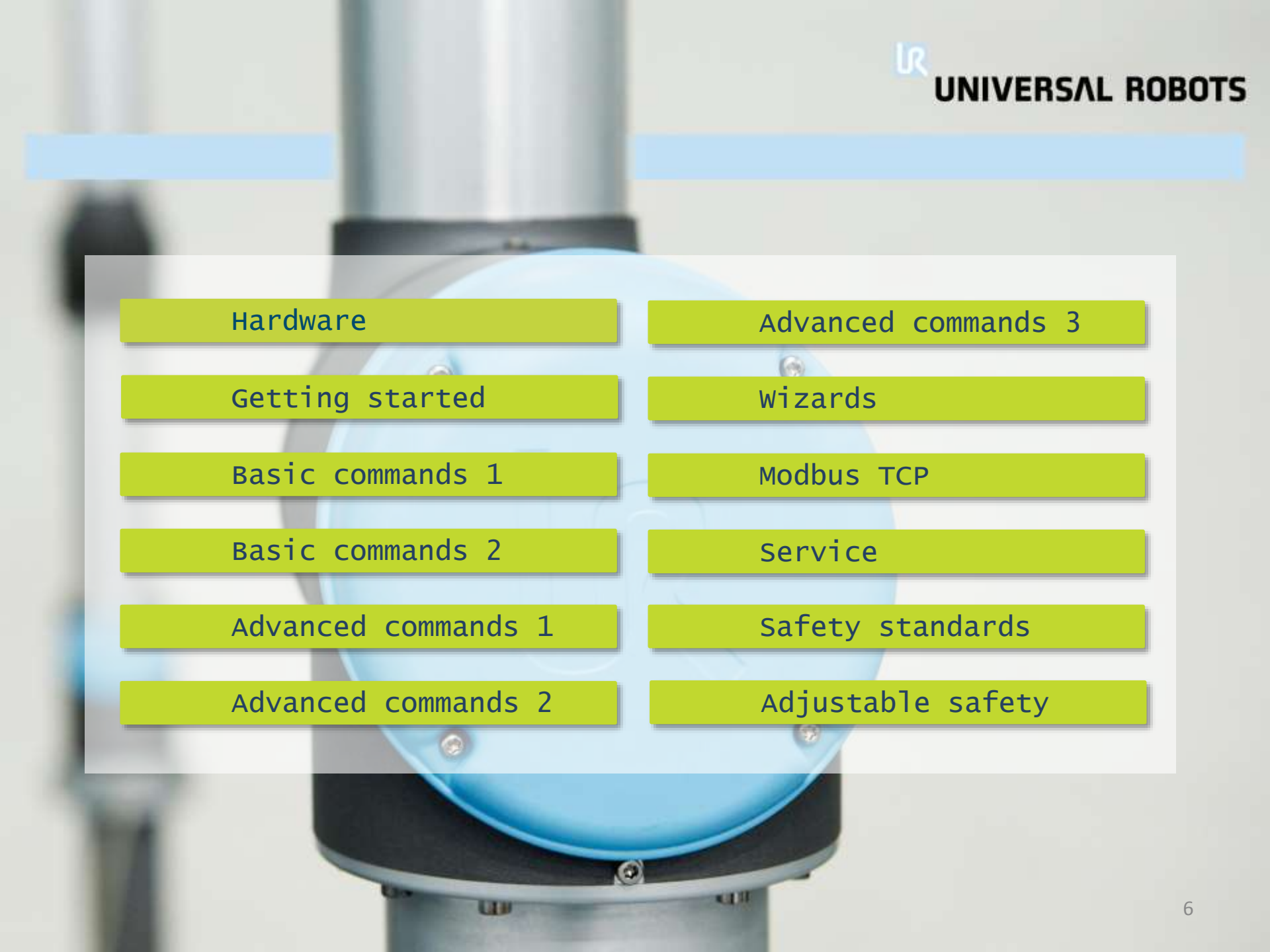
## Universal Robots Basic Training

» A practical hands-on experience

» Sample programs saved during training

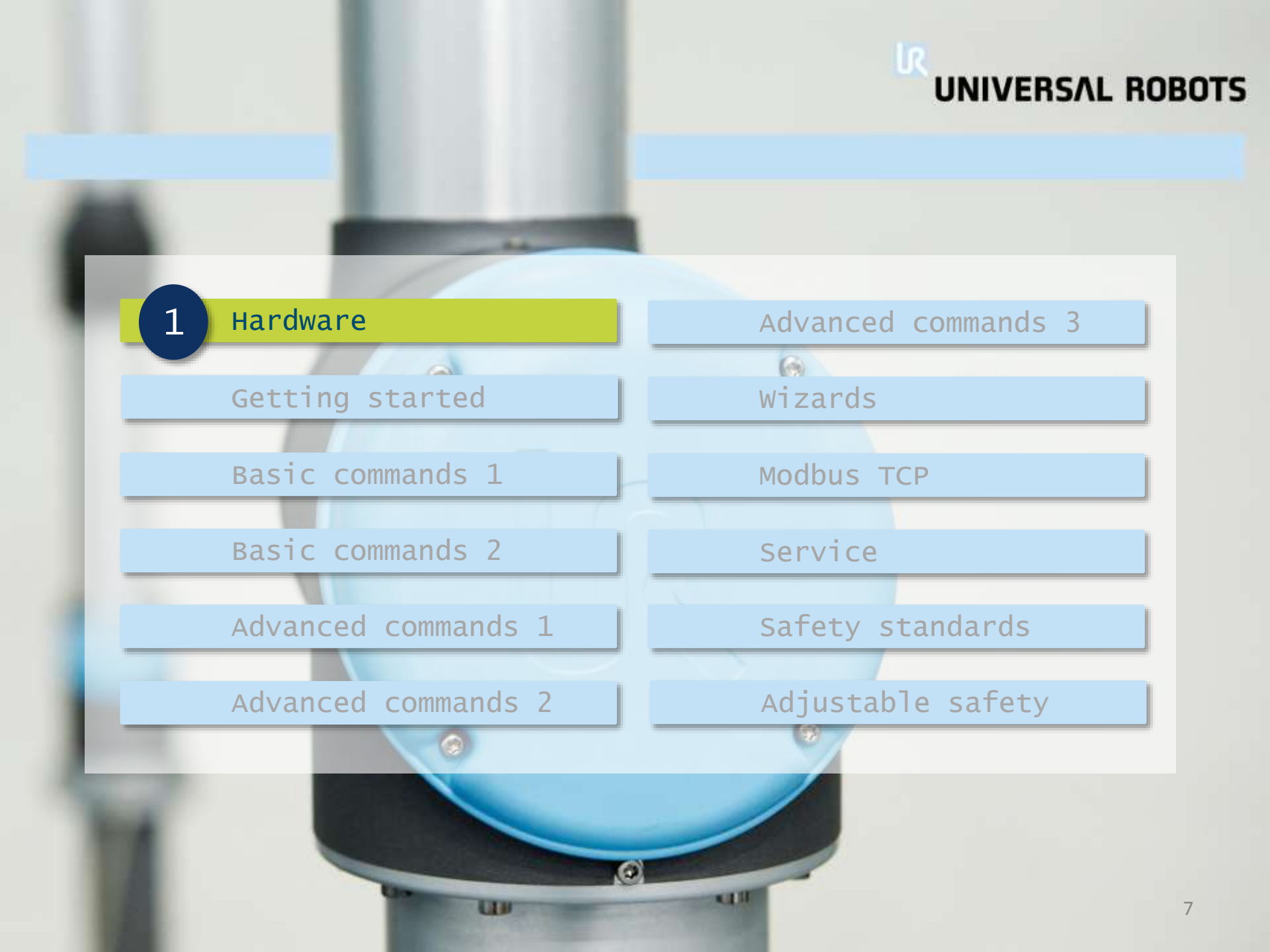
» Take notes

» 30 min. examination at end of training



UNIVERSAL ROBOTS

- 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



UNIVERSAL ROBOTS

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

UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

## Robot arm

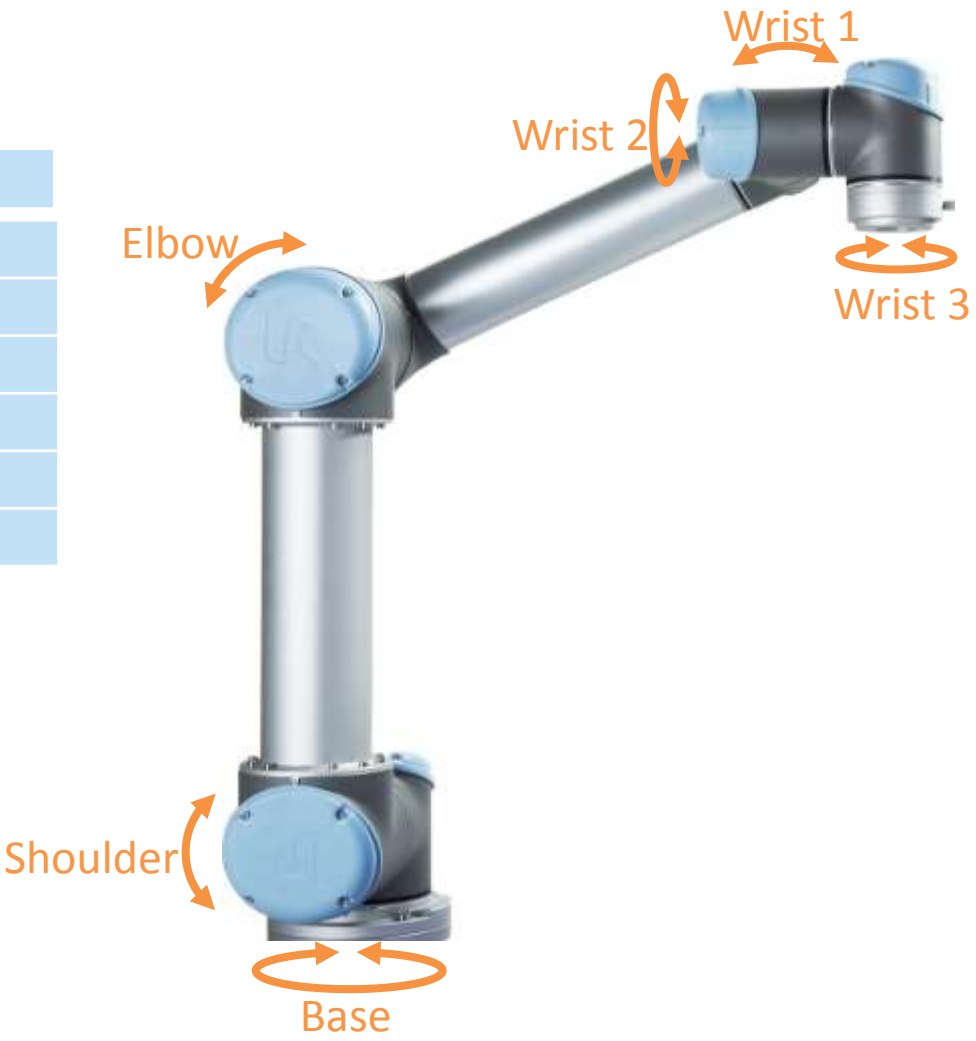
- Manipulator design
  - 6 axes
  - Articulated robot (*closely resembles human arm*)
  - Modular design
  - +/- 360° freedom
  - 3-phase AC servo motors
- Available types
  - UR5
  - UR10

UNIVERSAL ROBOTS

## 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 |



UNIVERSAL ROBOTS

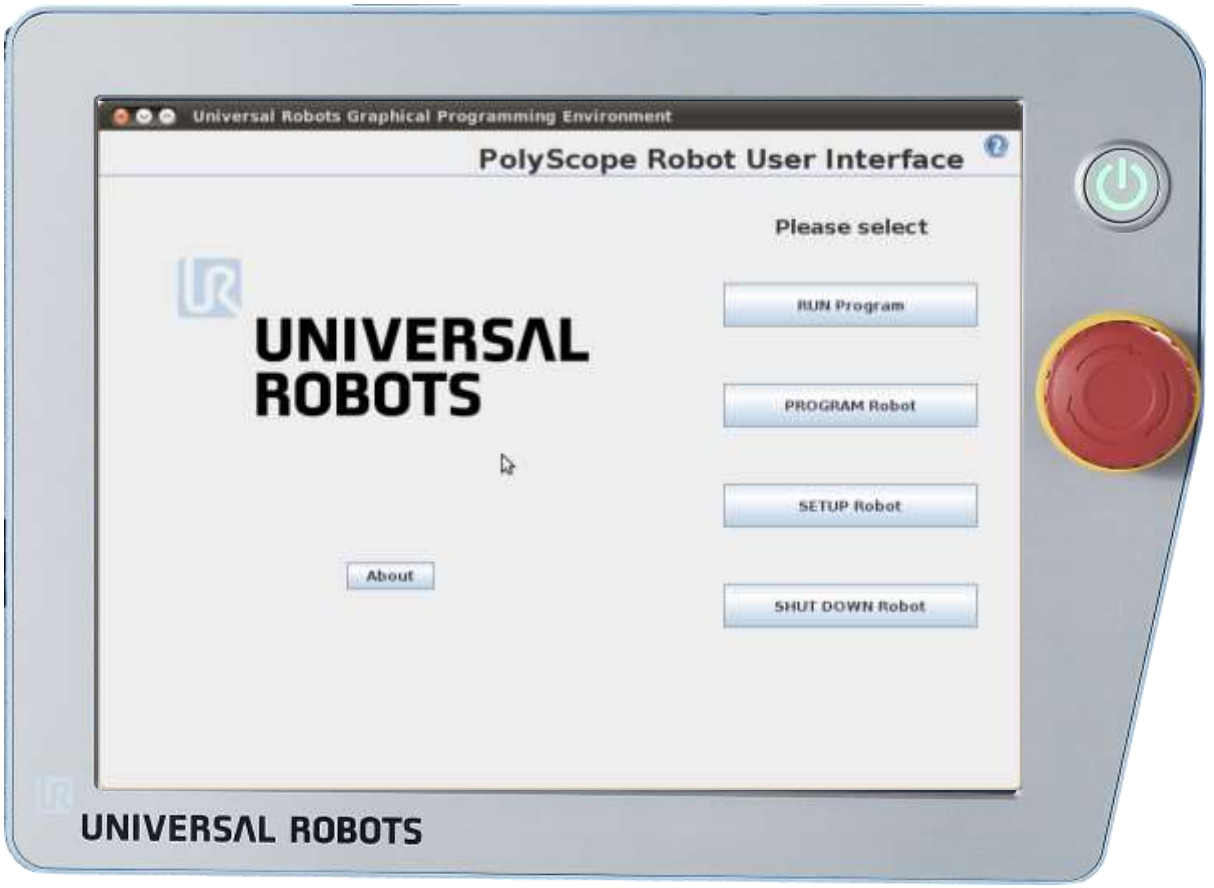
## Control box

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

UNIVERSAL ROBOTS

## Teach pendant

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



UNIVERSAL ROBOTS

# Specifications

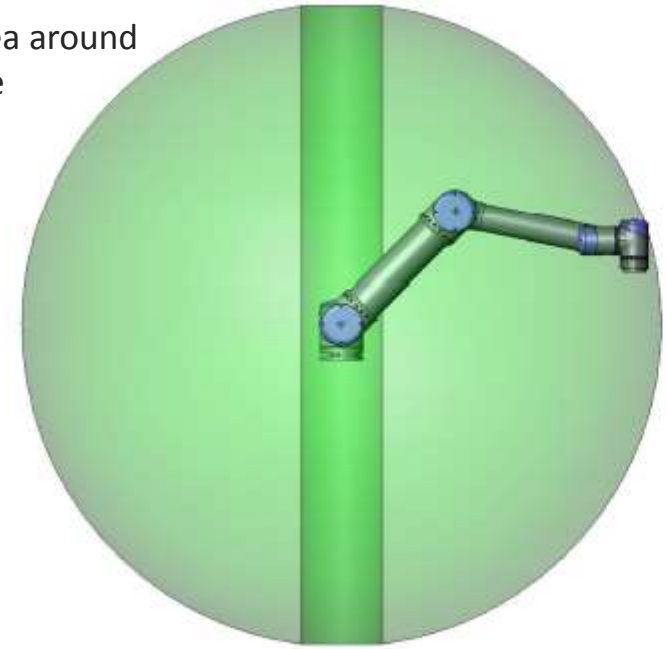
| | UR5 | UR10 |
|---|---|---|
| *Payload* | 5 kg. | 10 kg. |
| *Reach* | 850 mm | 1300 mm |
| *Joint ranges* | +/-360° | +/-360° |
| *Repeatability* | +/-0.1 mm | +/-0.1 mm |
| *Joint max. Speed* | 180°/sec | 120°/sec and 180°/sec |
| *Tool max. speed* | 1000 mm/sec | 1000 mm/sec |
| *Weight* | 18.4 kg | 28.9 kg |
| *IP rating* | IP54 | IP54 |
| *Temp. range* | 0-50°C | 0-50°C |
| *Power supply* | 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)*

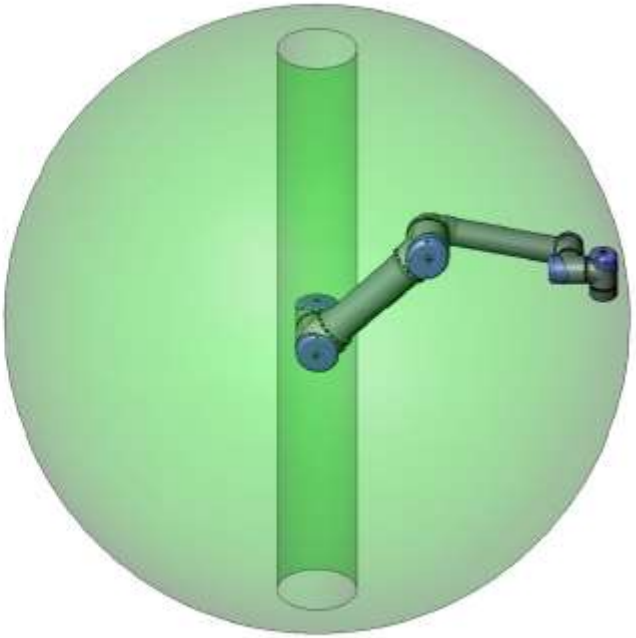
UNIVERSAL ROBOTS

## Robot workspace

- UR5 workspace
  - Approximate sphere of Ø170 cm
- Limitation
  - Cylindrical area around center of Base



Front view

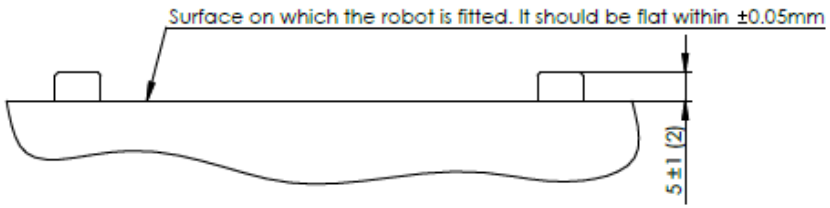


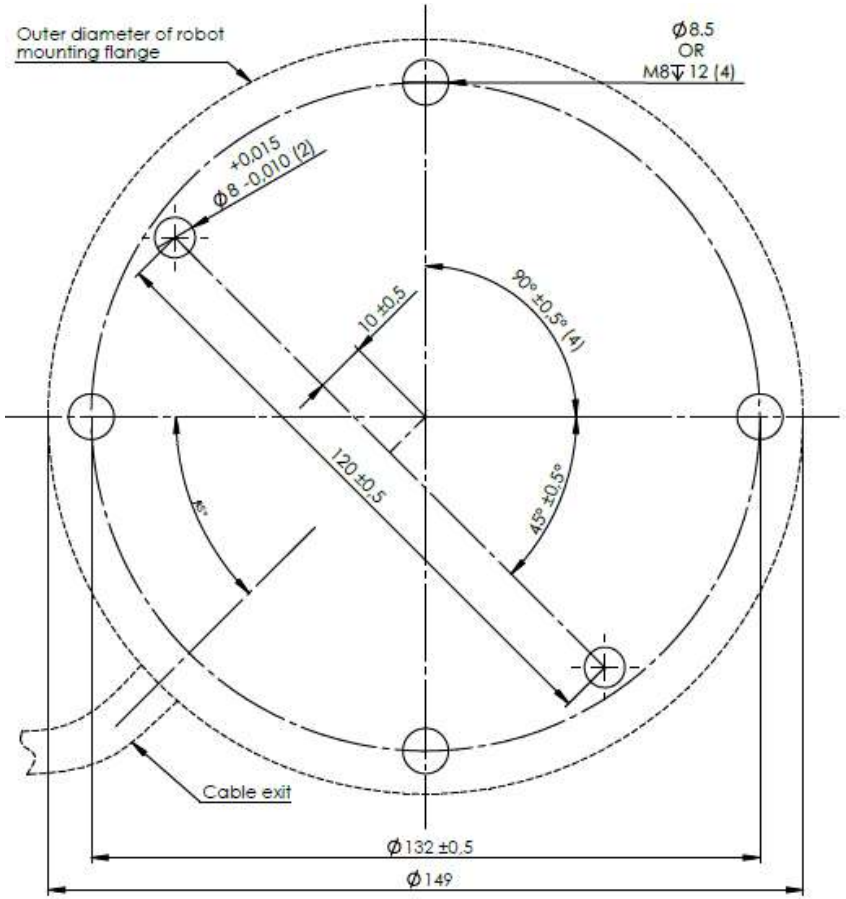
Tilted view

UNIVERSAL ROBOTS

# Mounting the base

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

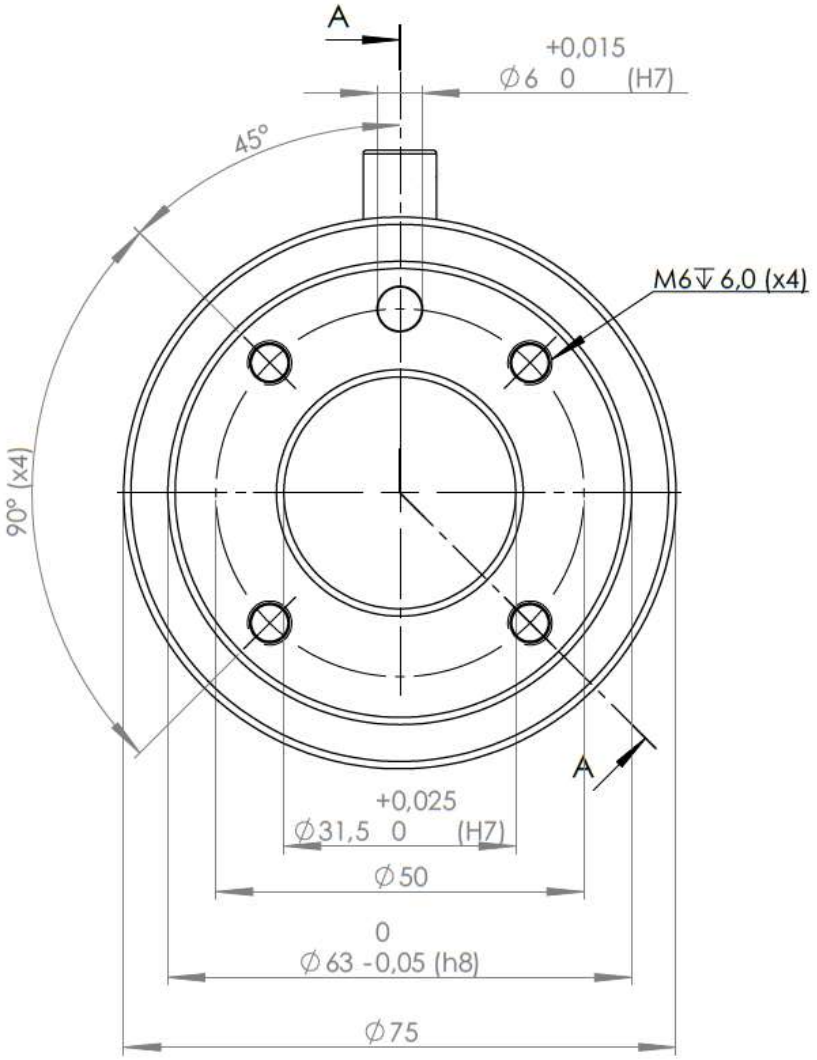


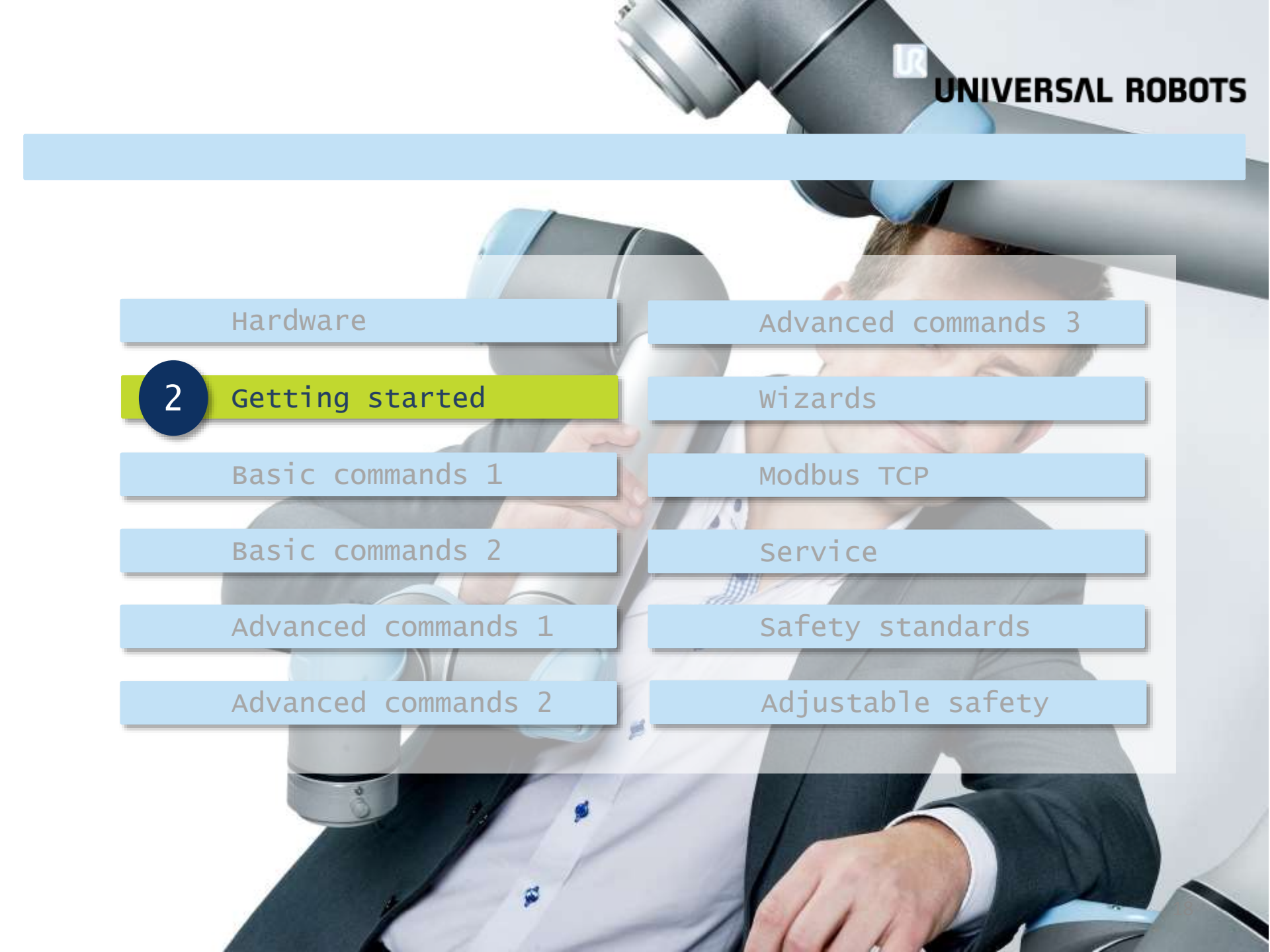



UNIVERSAL ROBOTS

## Mounting the tool

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





UNIVERSAL ROBOTS

Hardware

2 Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

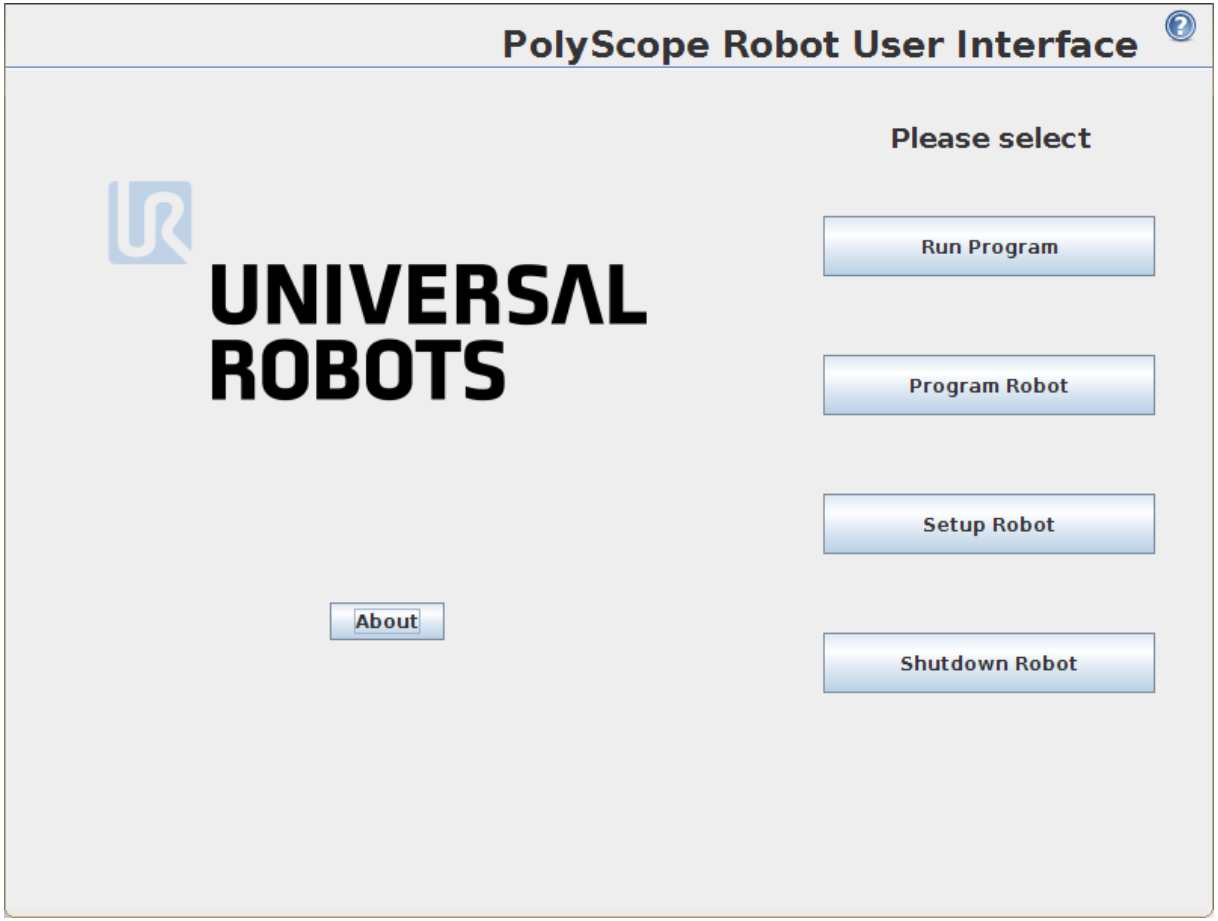
Safety standards

Adjustable safety

UNIVERSAL ROBOTS

## Introduction to PolyScope

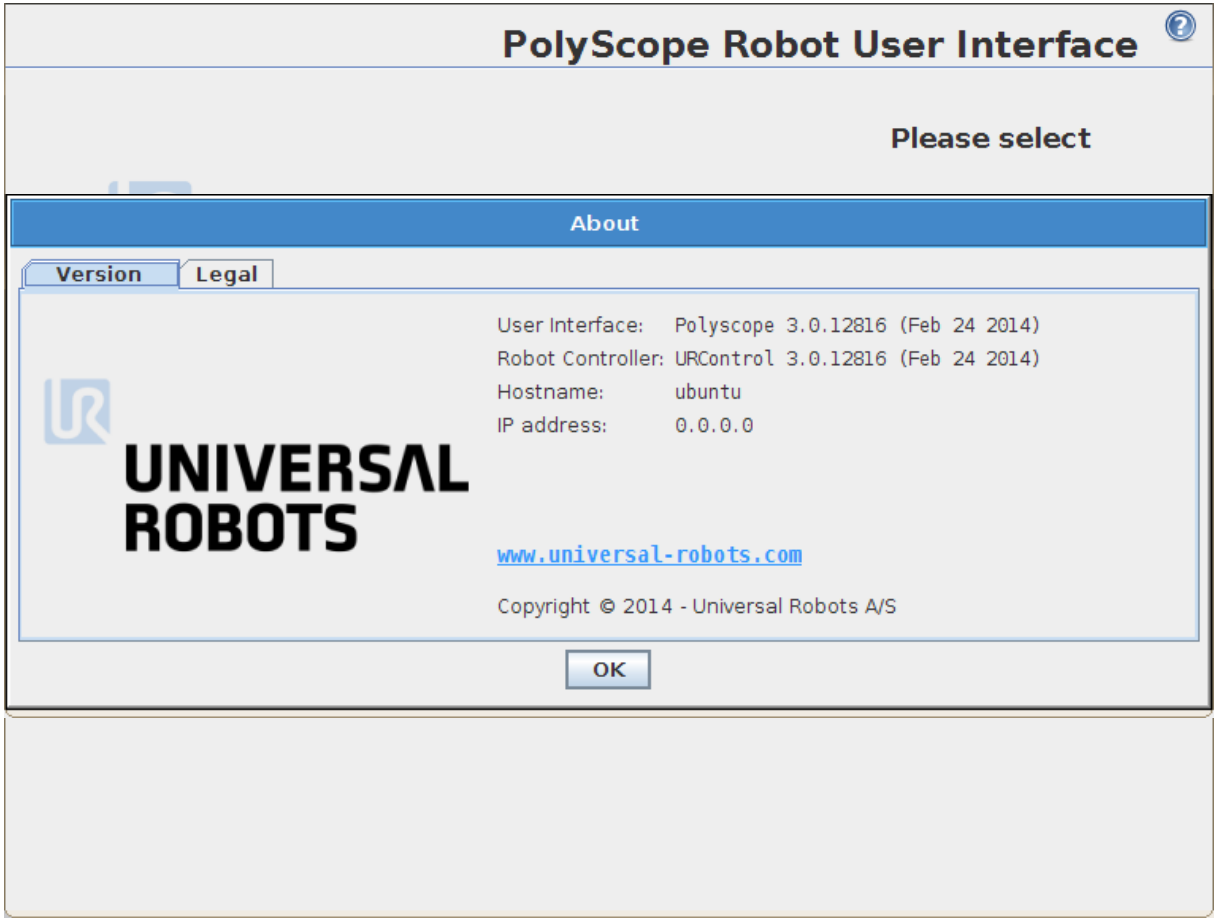
- PolyScope
  - Developed by UR
  - Free updates
- Operating system
  - Debian Linux



UNIVERSAL ROBOTS

## About

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



UNIVERSAL ROBOTS

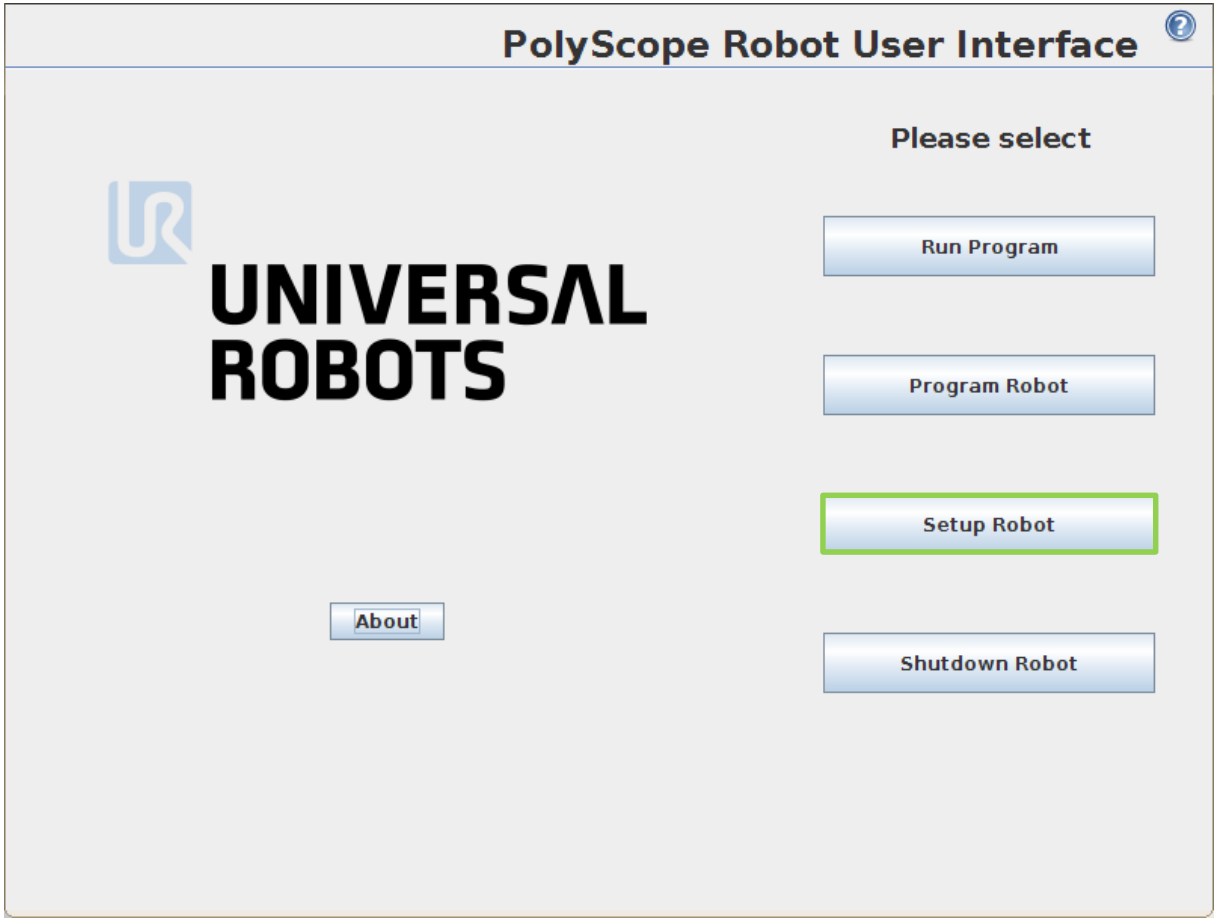
# Online manual

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

UNIVERSAL ROBOTS

## Setup robot

- Setup robot
  - Adjust software settings



UNIVERSAL ROBOTS

## Setup robot

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

UNIVERSAL ROBOTS

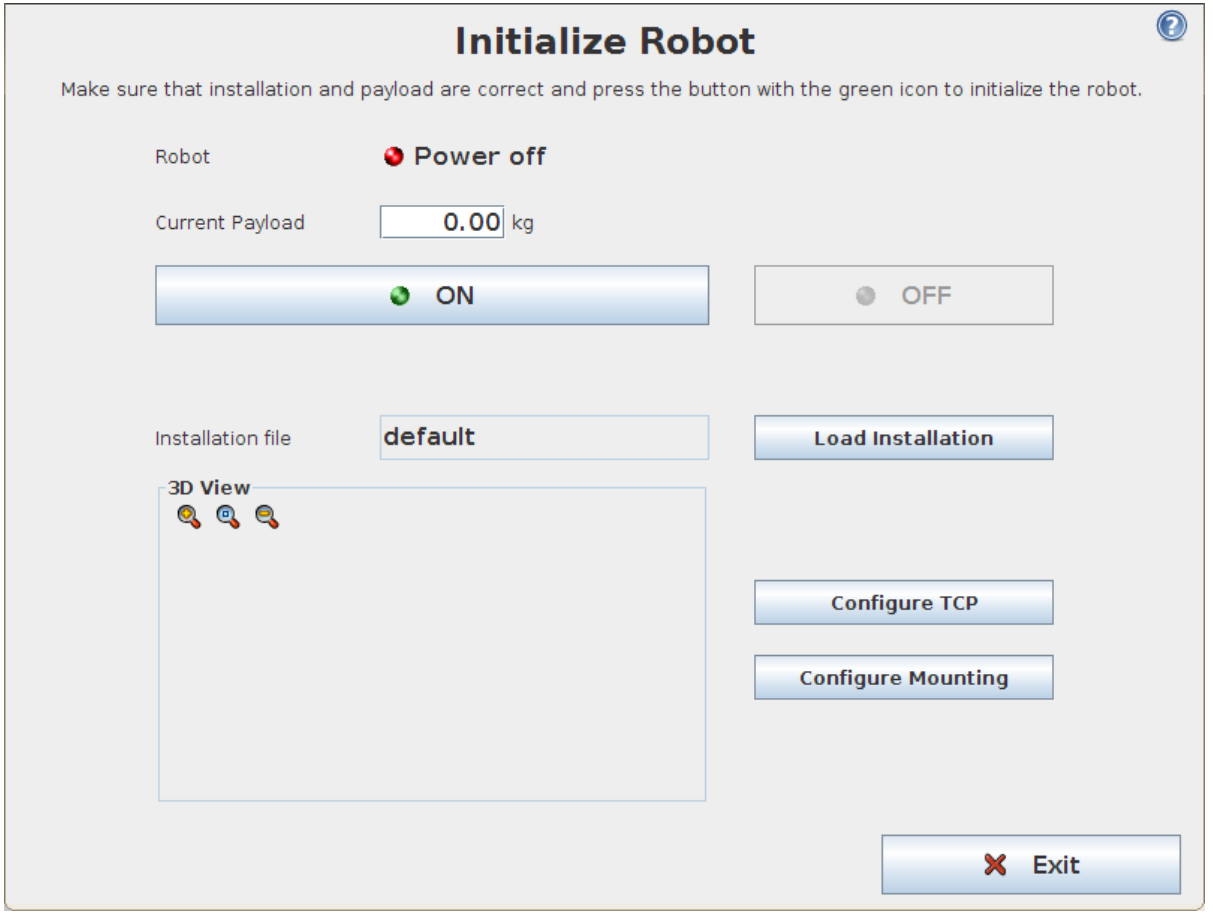
# Initialize robot

- 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

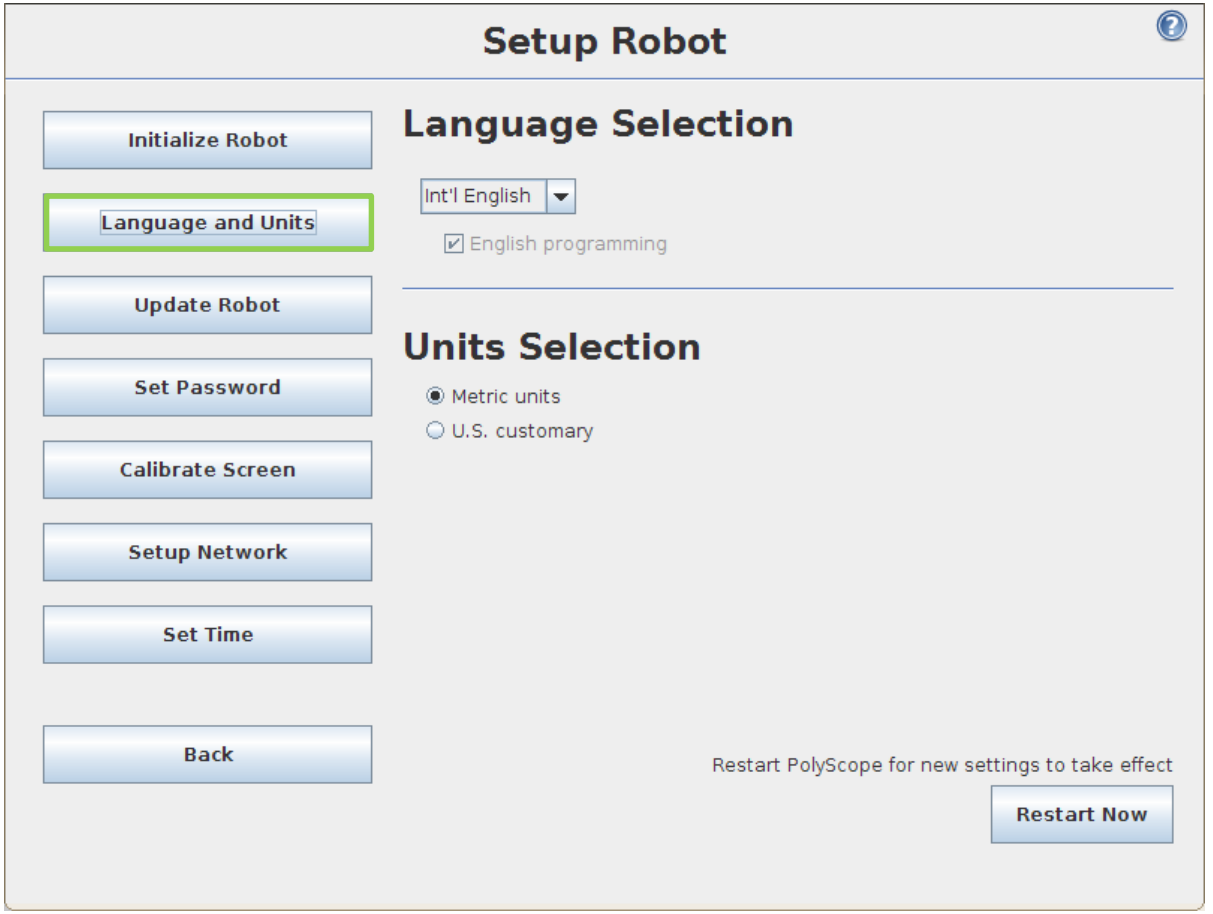
- Backdrive mode
  - When close to collision, use Backdrive mode



UNIVERSAL ROBOTS

# Language and units

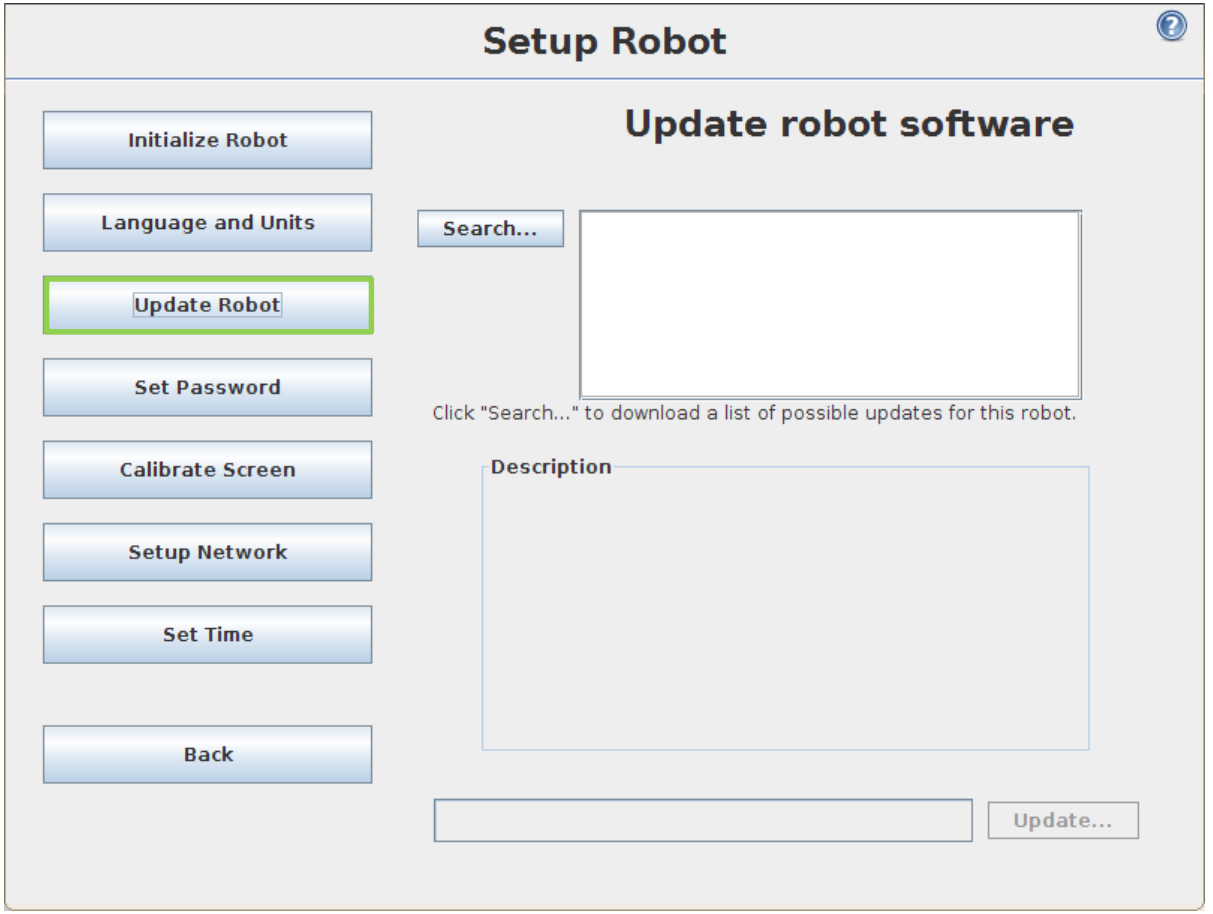
- Languages
  - 20 languages
  - English programming
    - Keeps command names in English language
- Units
  - Metric
  - U.S.



UNIVERSAL ROBOTS

# Update robot

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

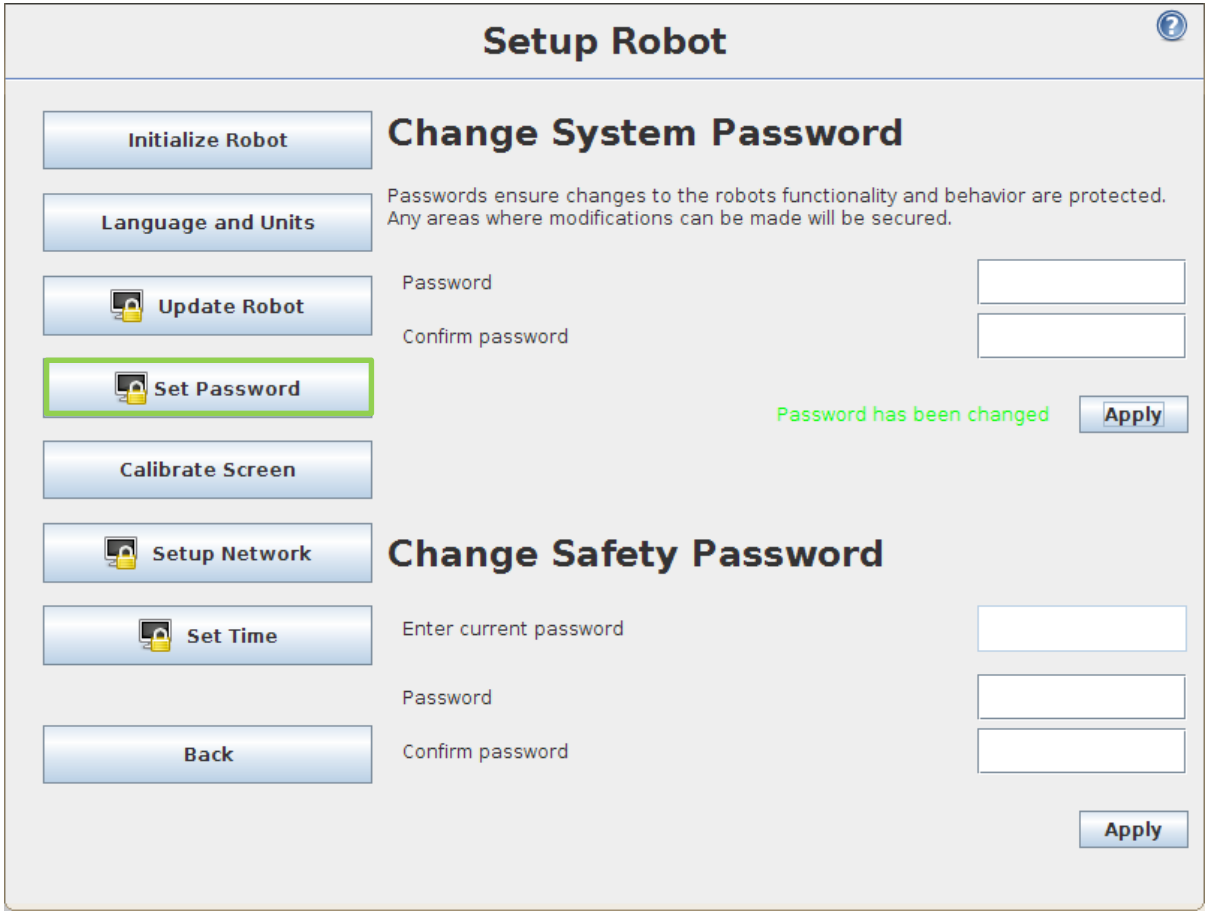



- Procedure to update will be covered later in training session

UNIVERSAL ROBOTS

# Set password

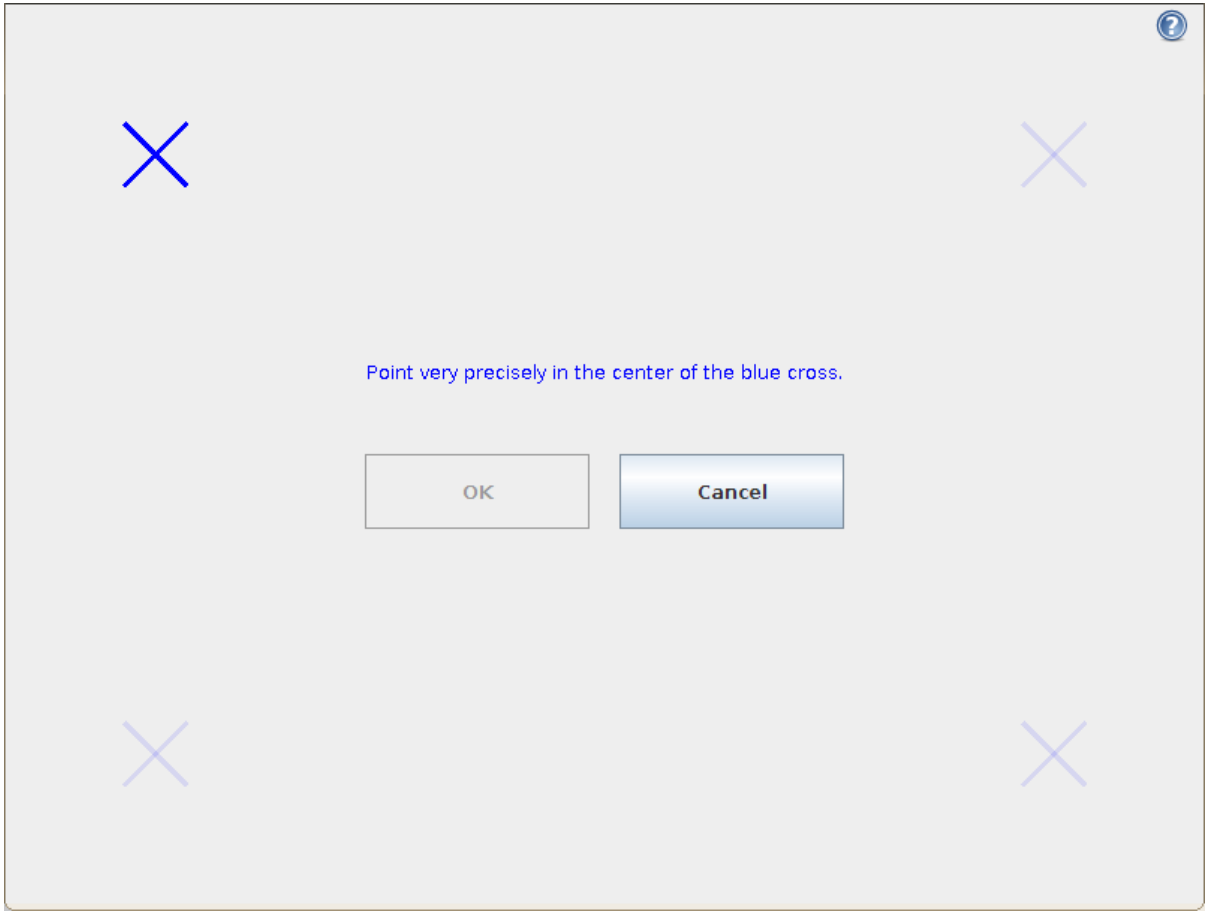
- System password
  - Limit access to parts of software
- Safety password
  - Required for modifying safety settings



UNIVERSAL ROBOTS

## Calibrate screen

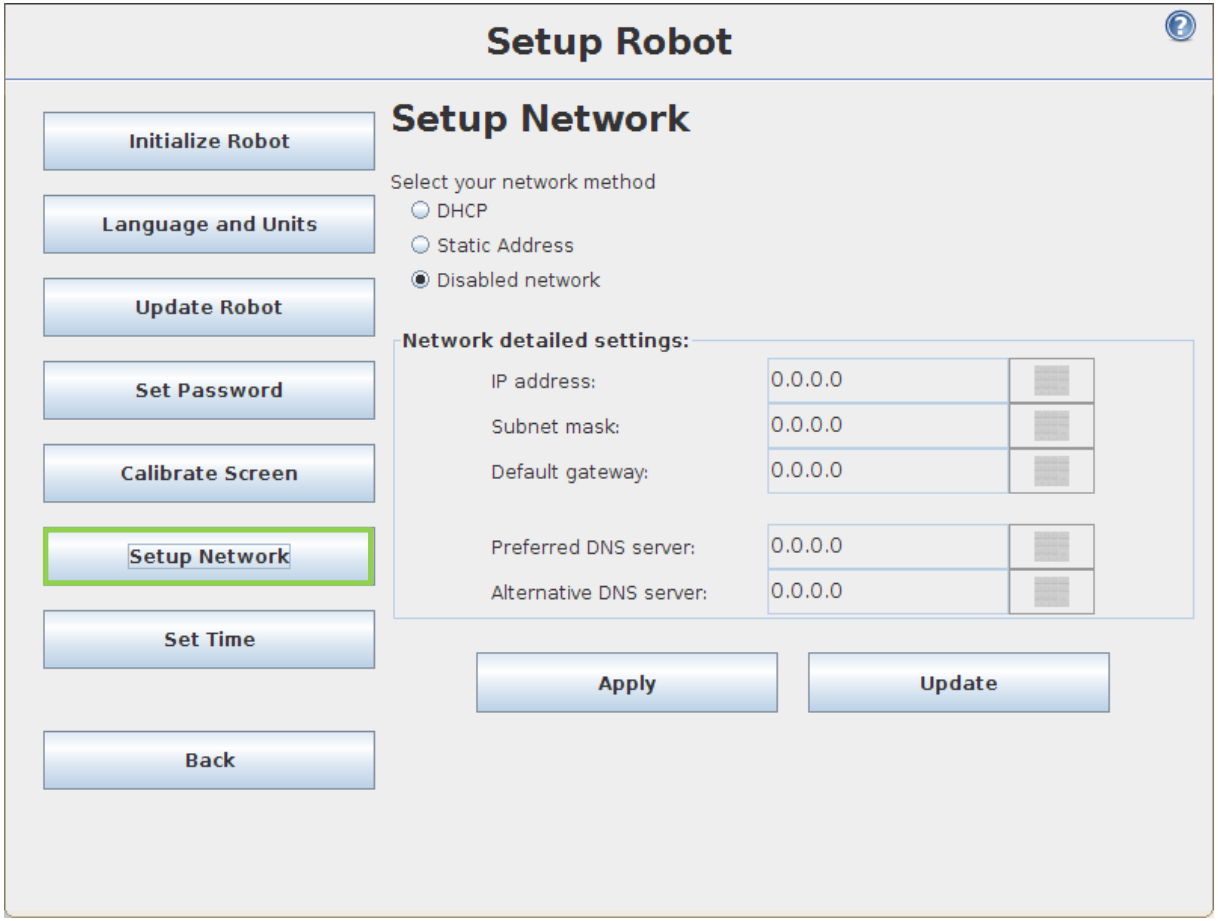
- Calibration of touch screen
  - Mark four corners to calibrate



UNIVERSAL ROBOTS

# Setup network

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

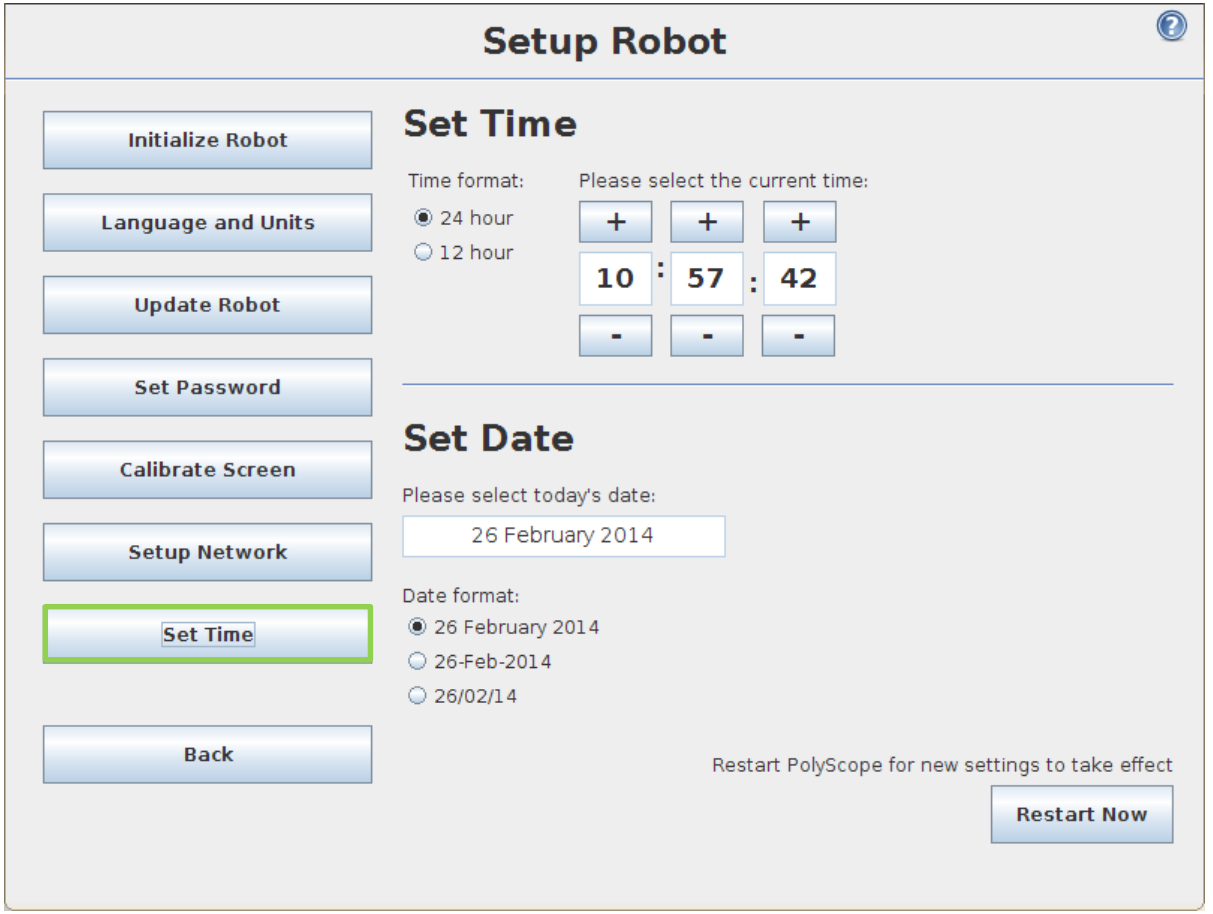



- Live demo will be covered later in training session

UNIVERSAL ROBOTS

## Set time

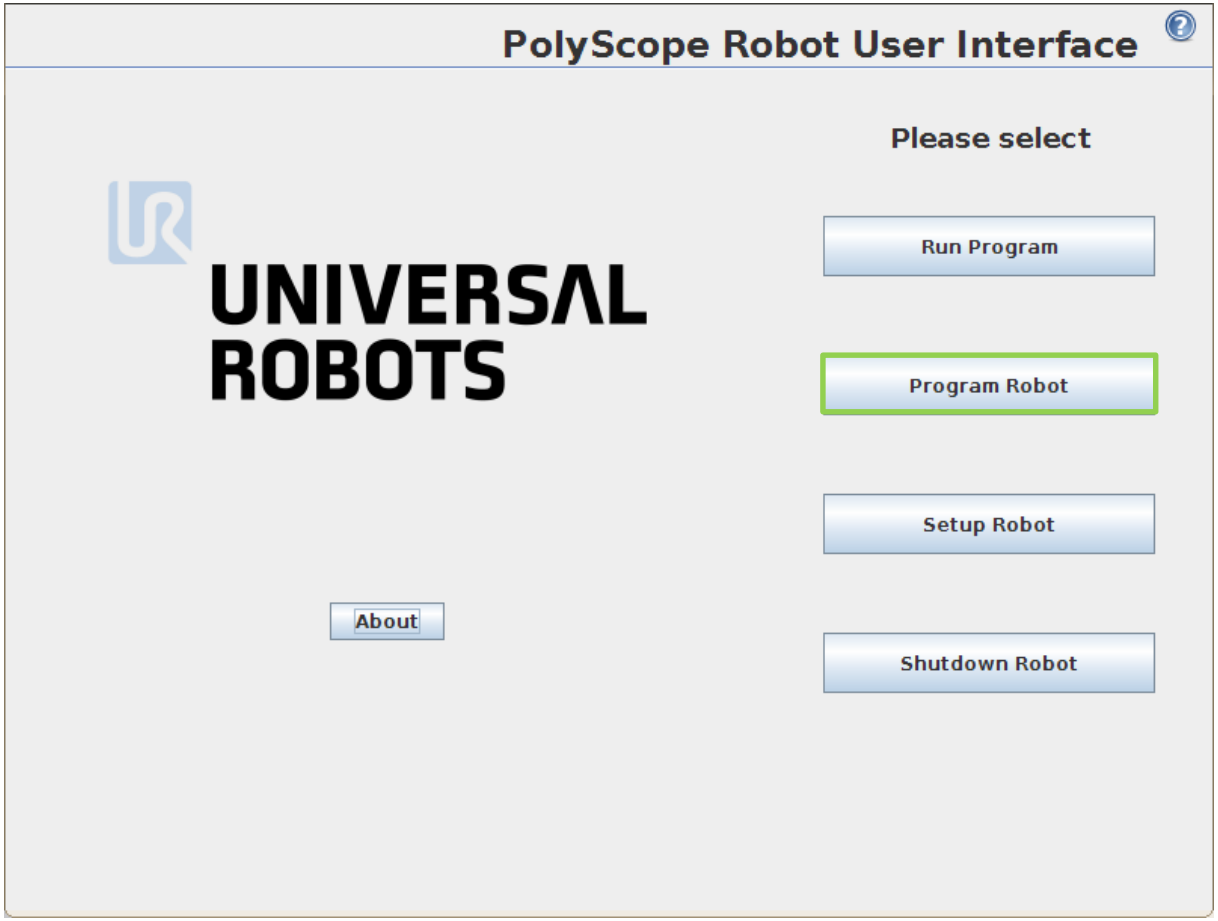
- Time
  - Time format
    - 24 hour
    - 12 hour
- Date
  - Date format



UNIVERSAL ROBOTS

## Program robot

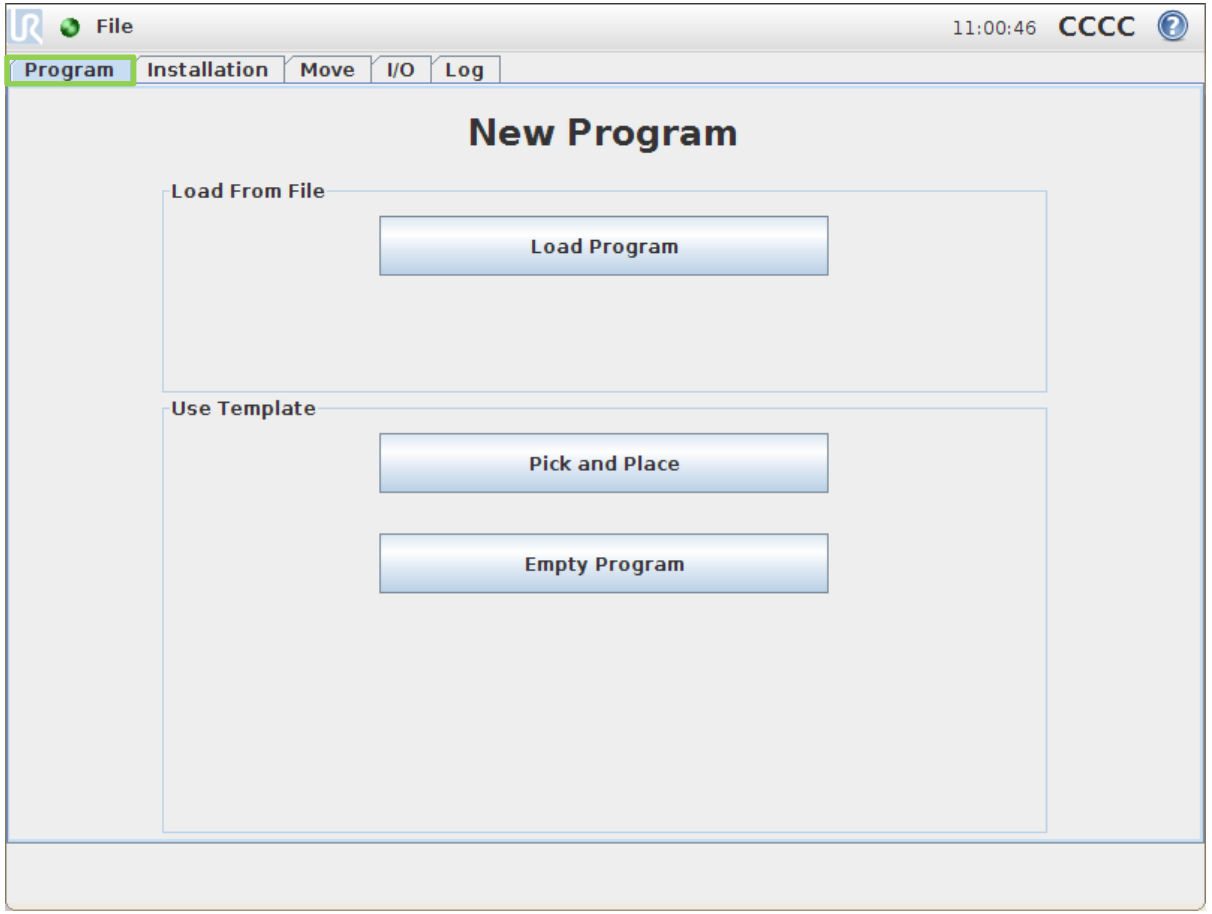
- Program robot
  - Overview of tabs



UNIVERSAL ROBOTS

# Program tab

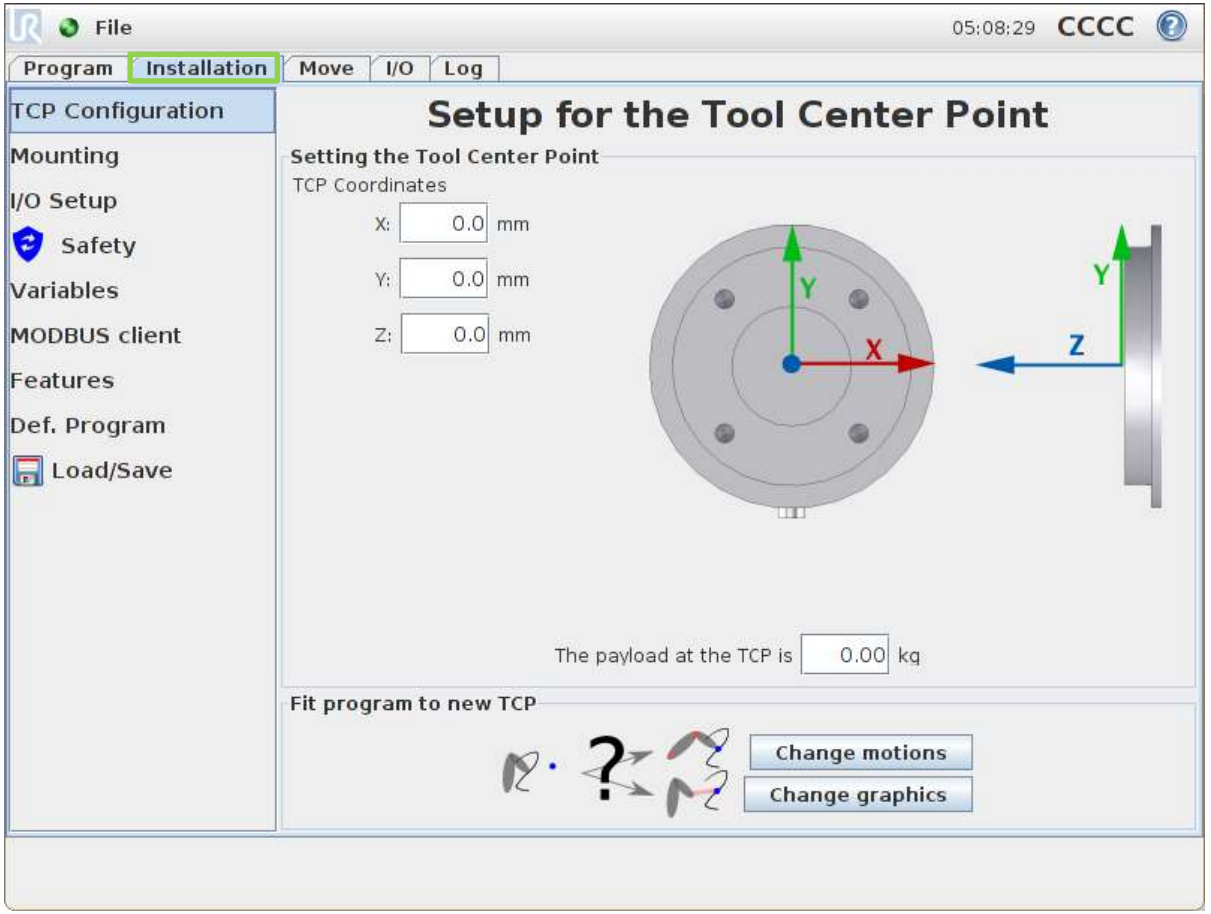
- Program
  - Load existing program
  - Create new program



UNIVERSAL ROBOTS

# Installation tab

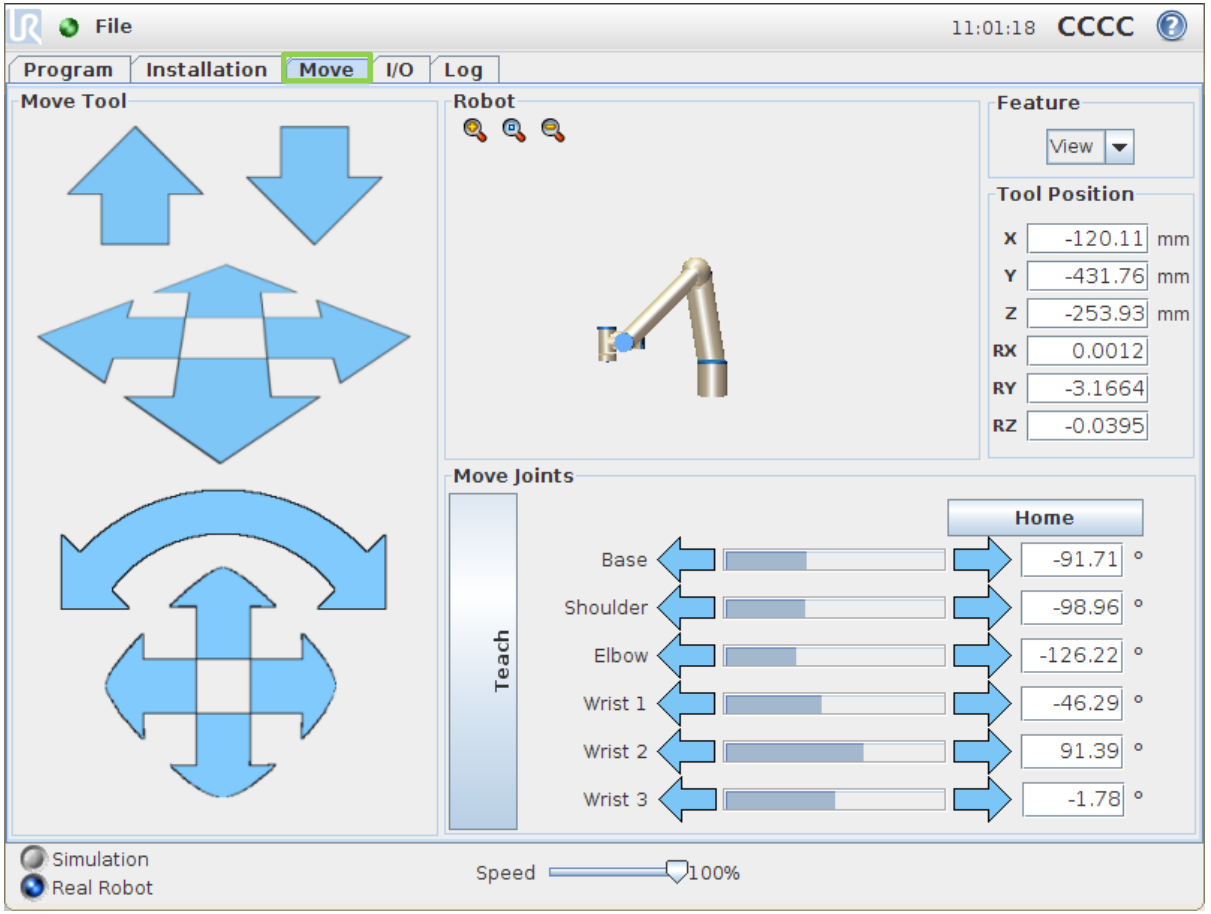
- Setup
  - Environment settings
  - Safety settings



UNIVERSAL ROBOTS

## Move tab

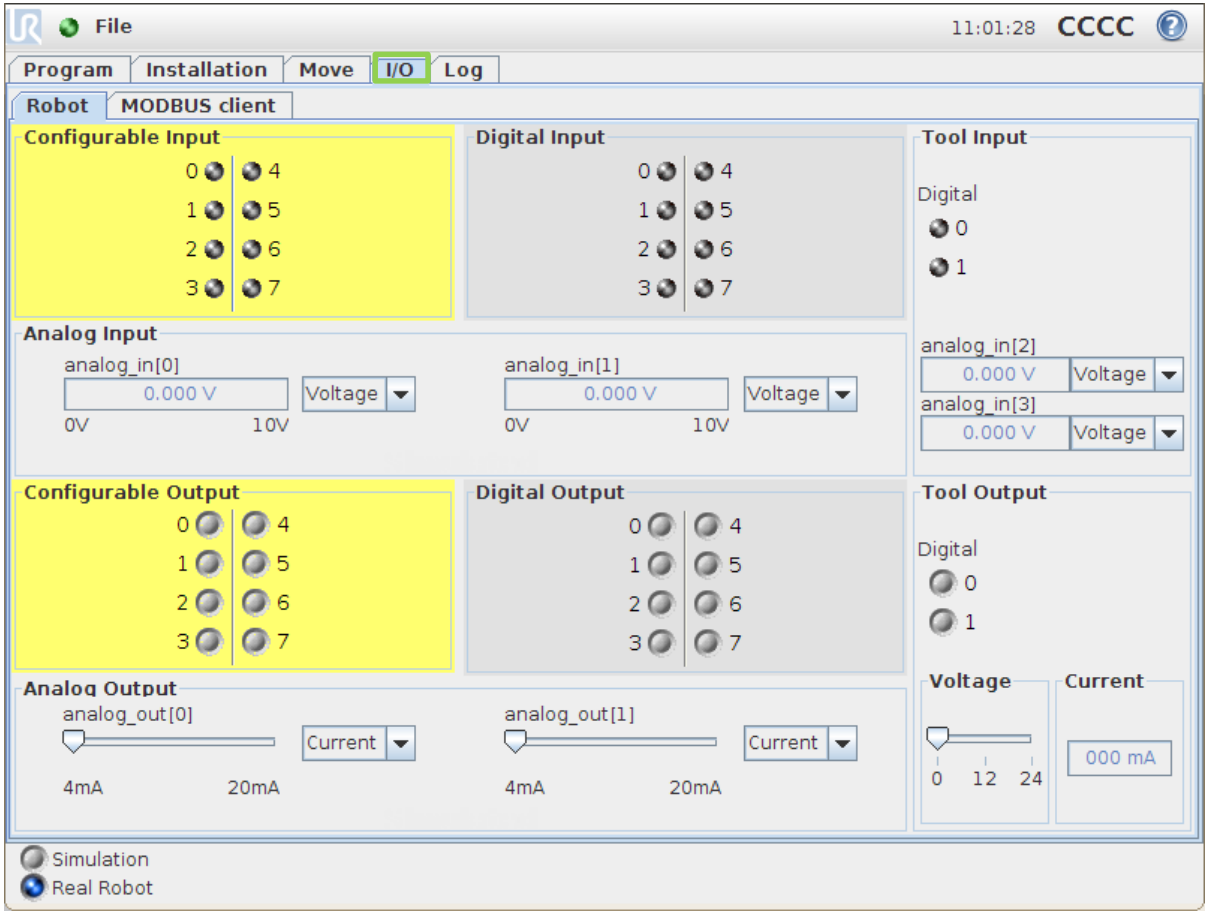
- Move
  - Jog robot manually
  - Actual position displayed



UNIVERSAL ROBOTS

# I/O tab

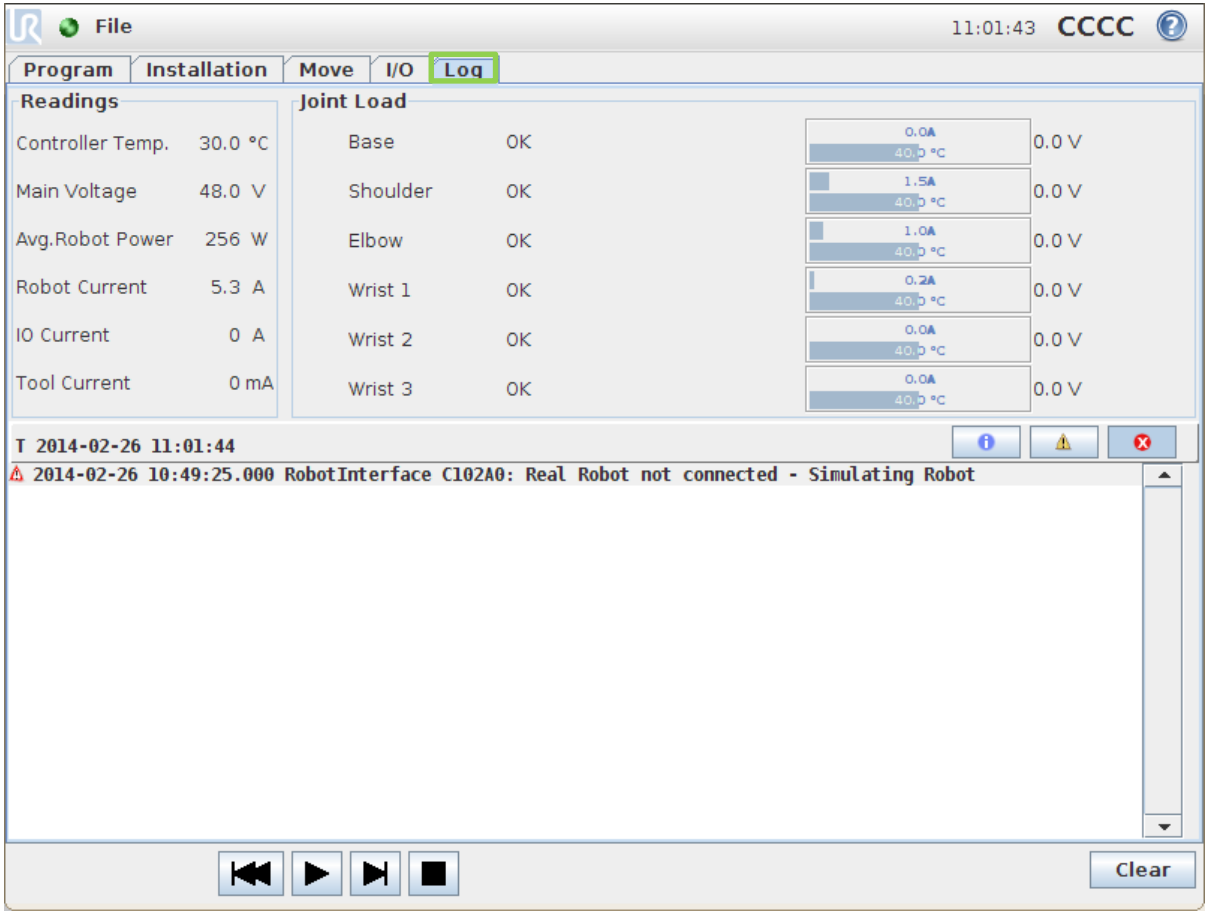
- I/O
  - Monitoring signals
  - Activate signal
  - Setup analog signals



UNIVERSAL ROBOTS

# Log tab

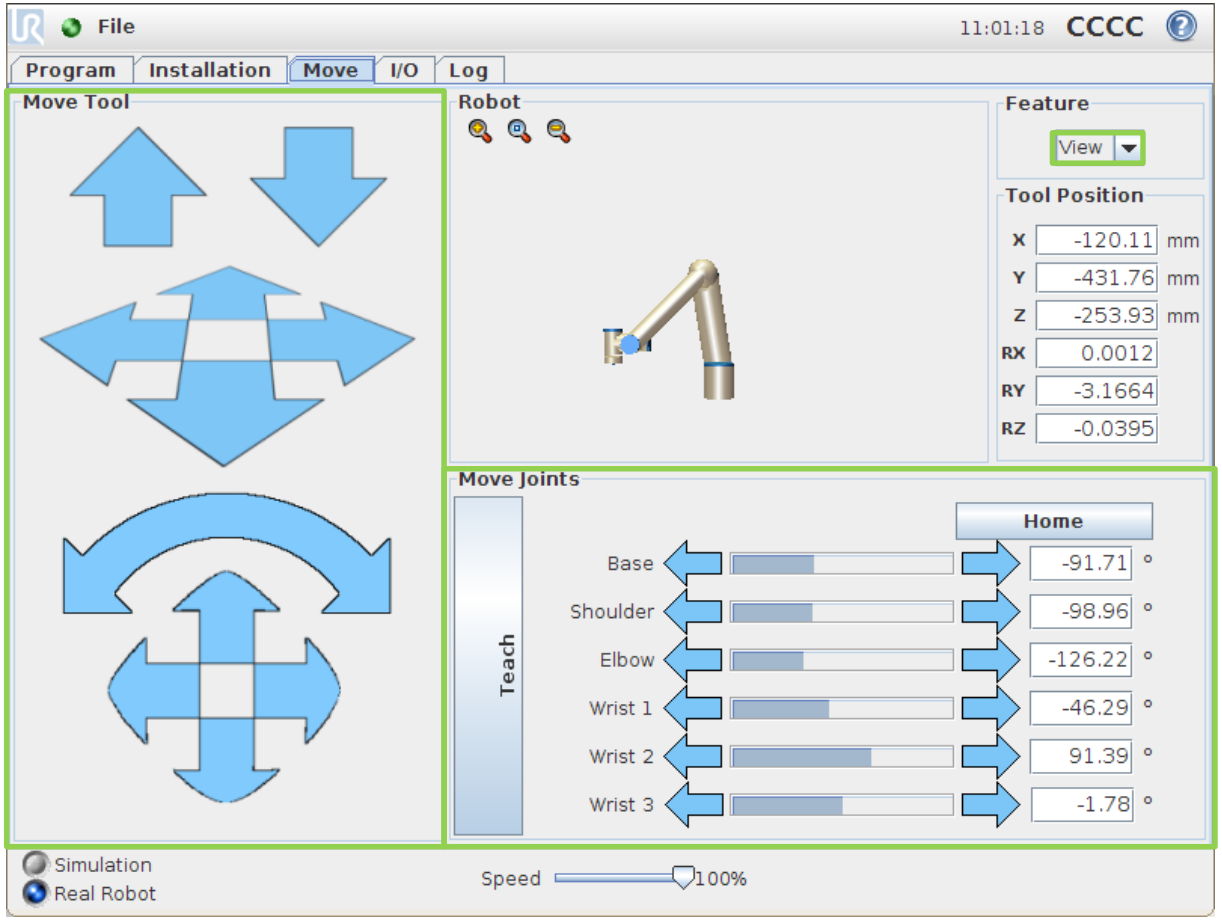
- Status
  - Control box
  - Joints
- Log history
  - Info messages
  - Warnings
  - Errors



UNIVERSAL ROBOTS

# Moving the robot

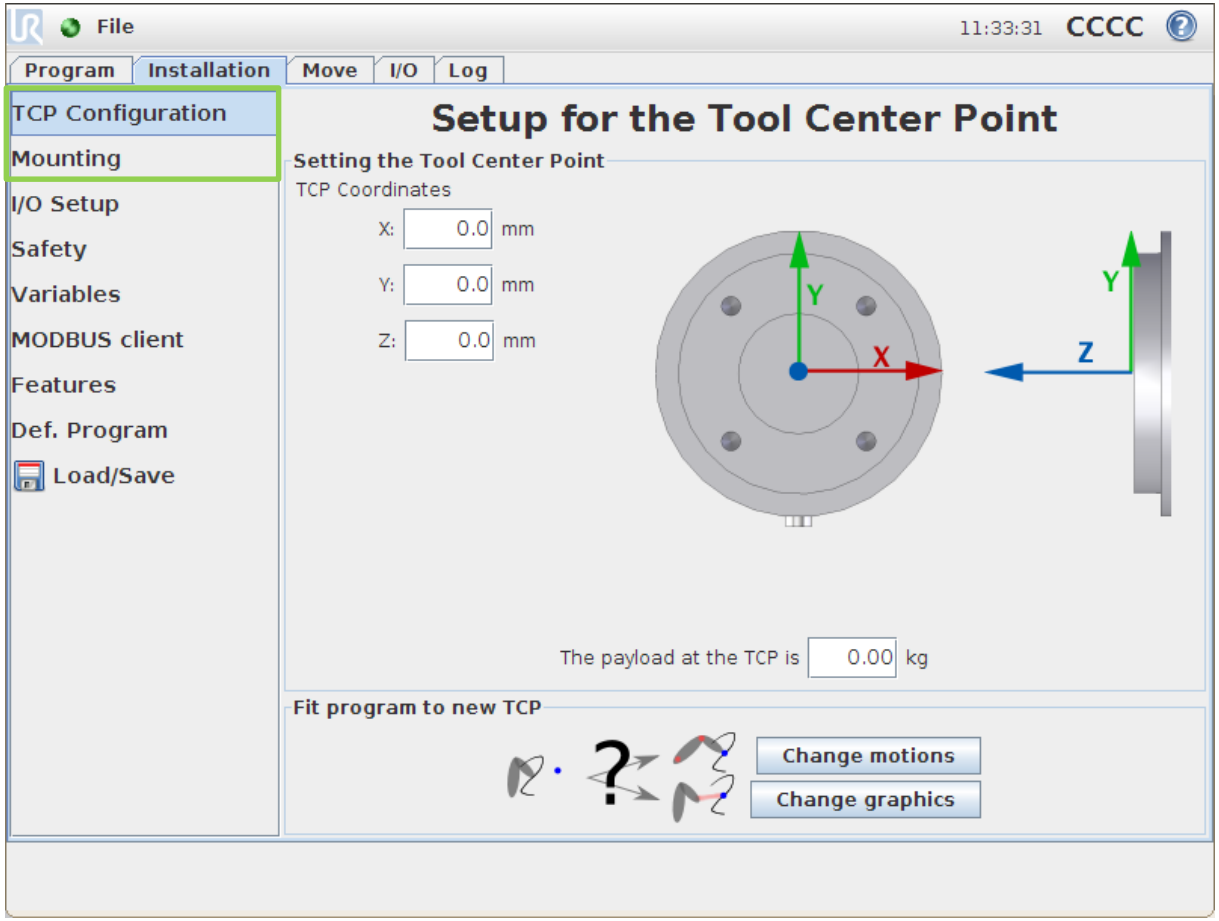
- Jogging
  - One joint individually
    - Tool orientation changes
    - Min/max limits
    - Angular value displayed
  - In linear space
    - Tool orientation is fixed when jogging in XYZ
    - Relative to selected Feature
      - View
      - Base
      - Tool
  - By teach function
    - Teach button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



UNIVERSAL ROBOTS

# Initial setup

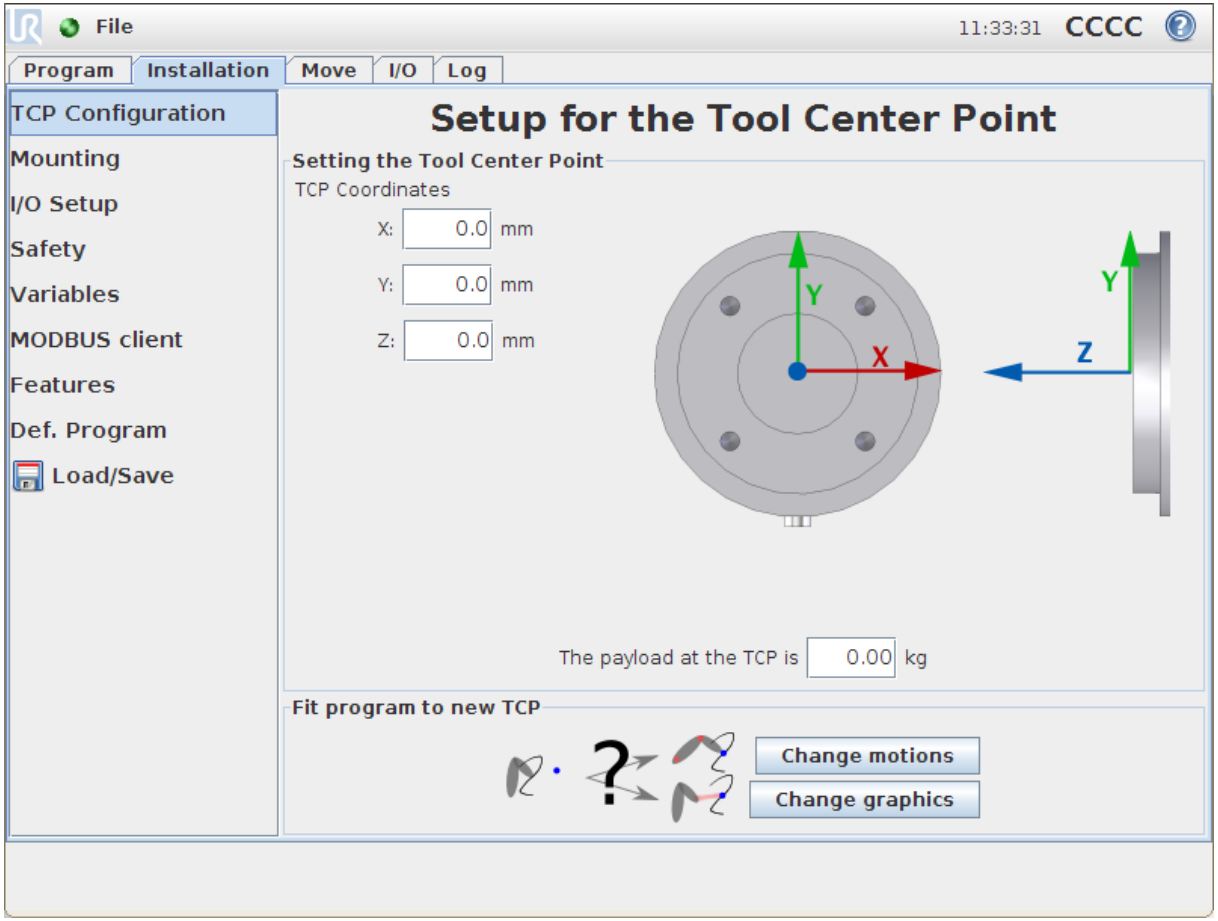
- Initial setup
  - TCP configuration
  - Mounting



UNIVERSAL ROBOTS

# TCP configuration

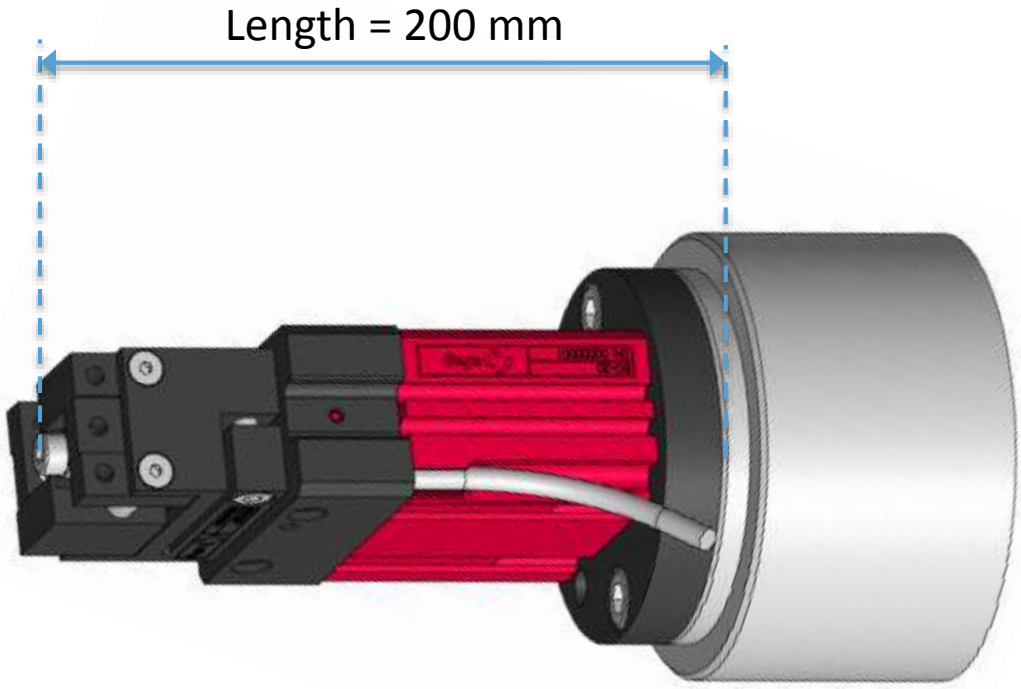
- Tool Center Point (TCP)
  - Linear distance from center of tool flange to tip of tool
  - Set XYZ coordinates according to illustration
- Payload
  - Weight of tool



UNIVERSAL ROBOTS

# Payload calculation

- Example:
  - how to calculate max. allowed payload

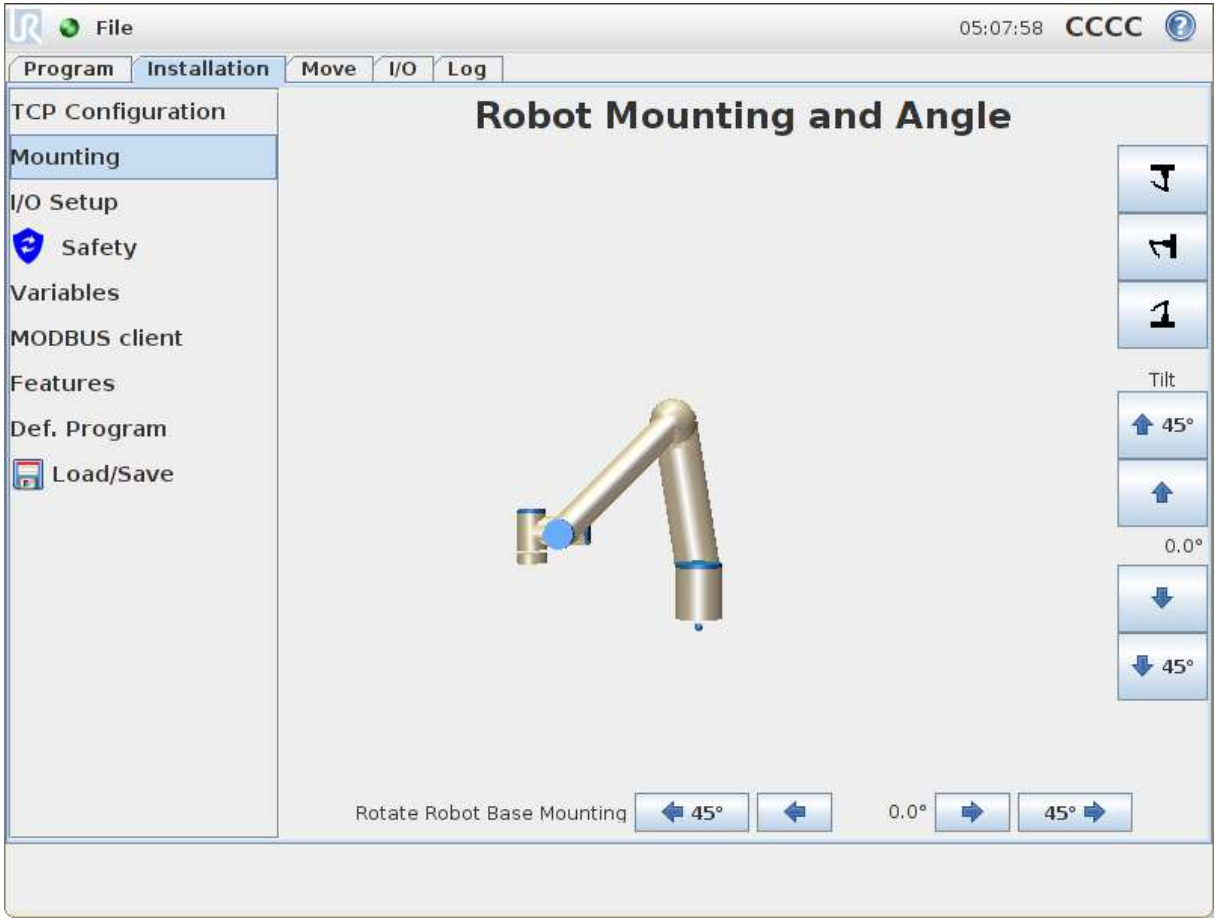



$$Payload = \frac{4.5}{(0.9 + length)} = \frac{4.5}{(0.9 + 0.2)} = 4.09\ kg$$

UNIVERSAL ROBOTS

## Mounting

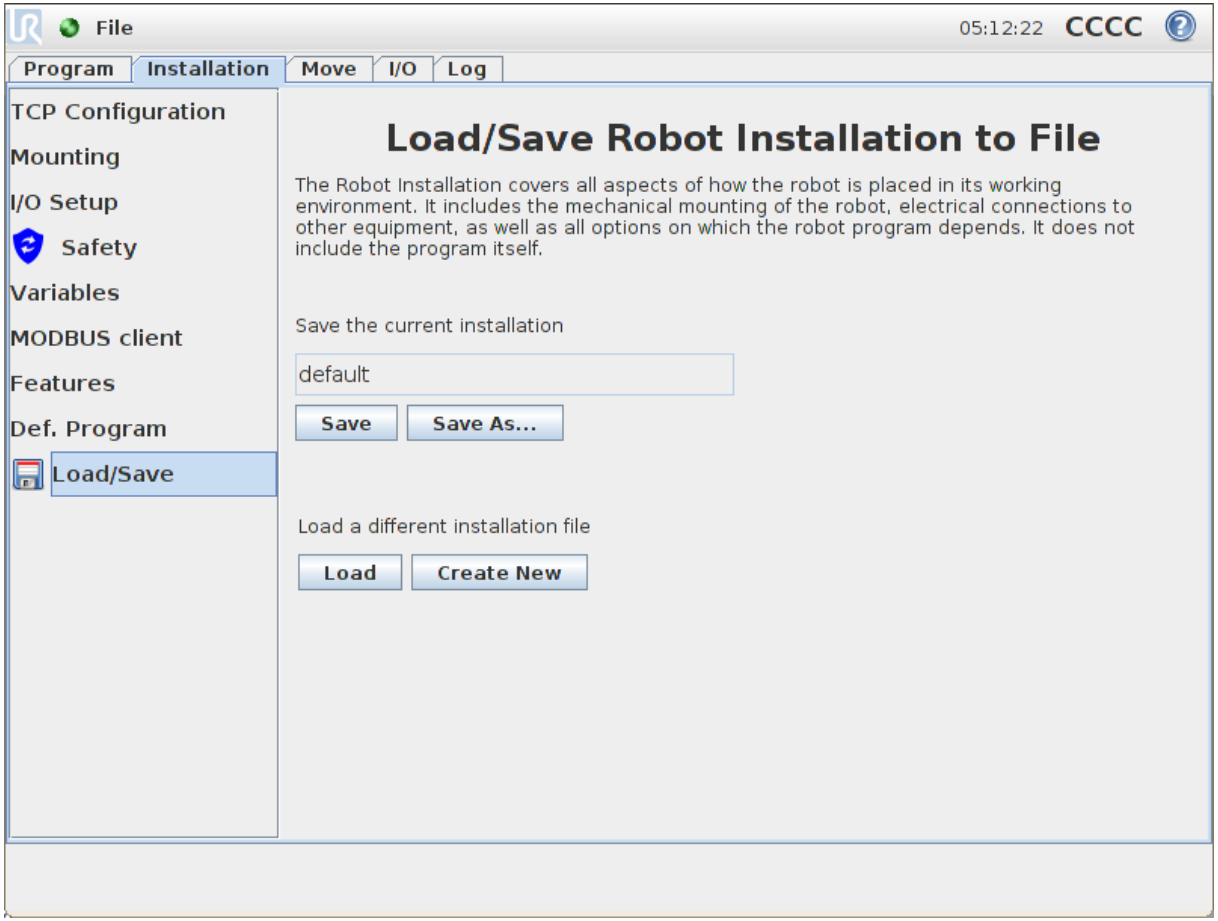
- Setup how robot base is mounted

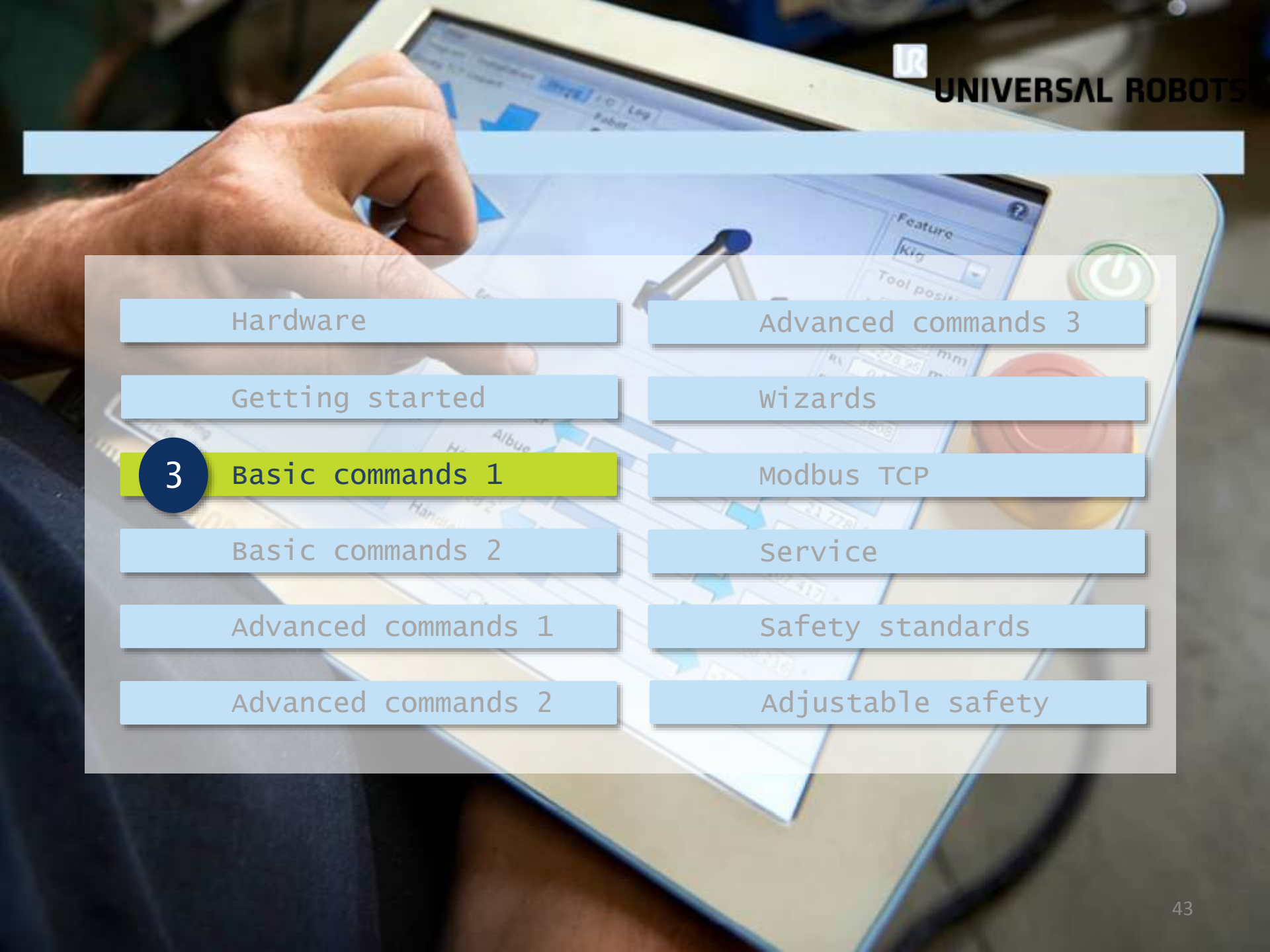


UNIVERSAL ROBOTS

# 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





UNIVERSAL ROBOTS

Hardware

Getting started

3 Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

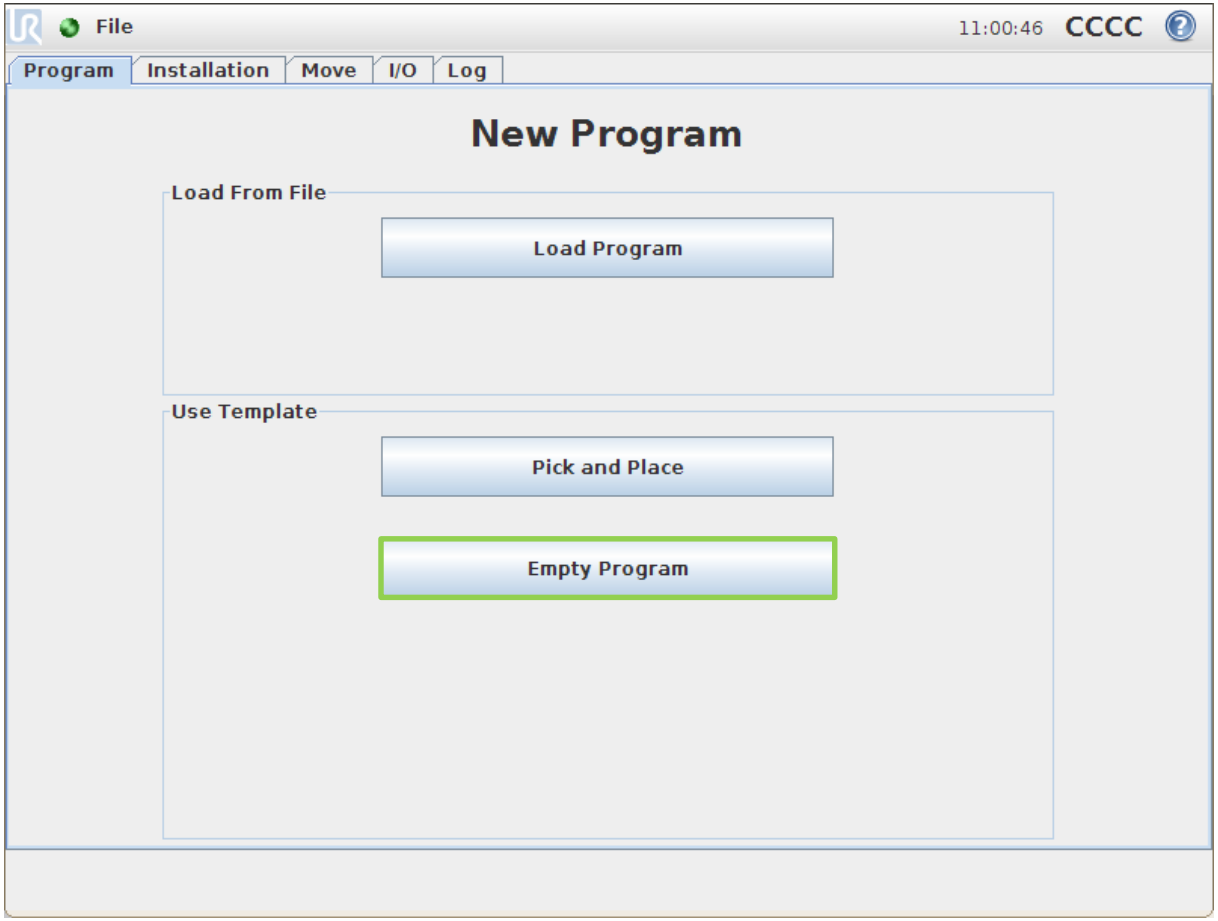
Safety standards

Adjustable safety

UNIVERSAL ROBOTS

## Create new program

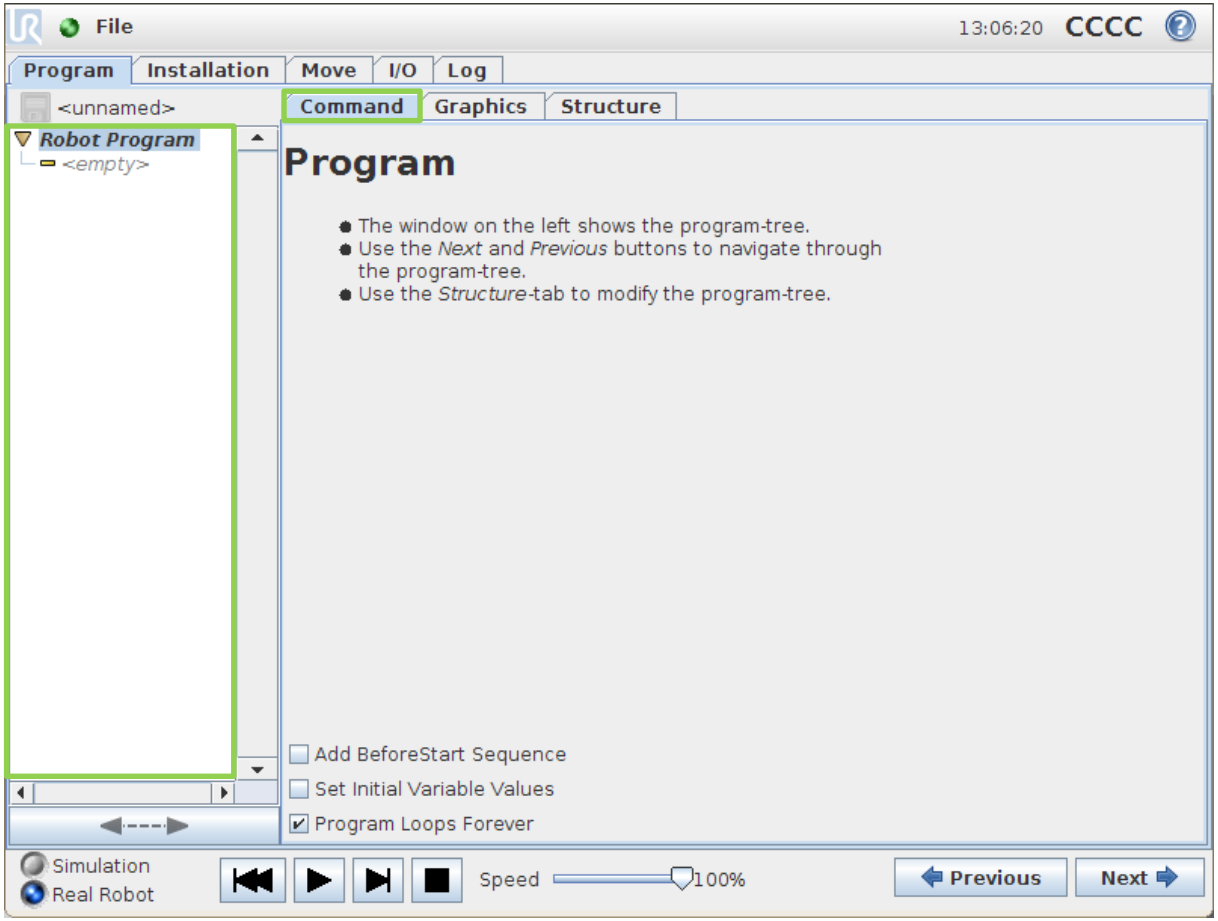
- Use template
  - Empty Program



UNIVERSAL ROBOTS

## Command tab

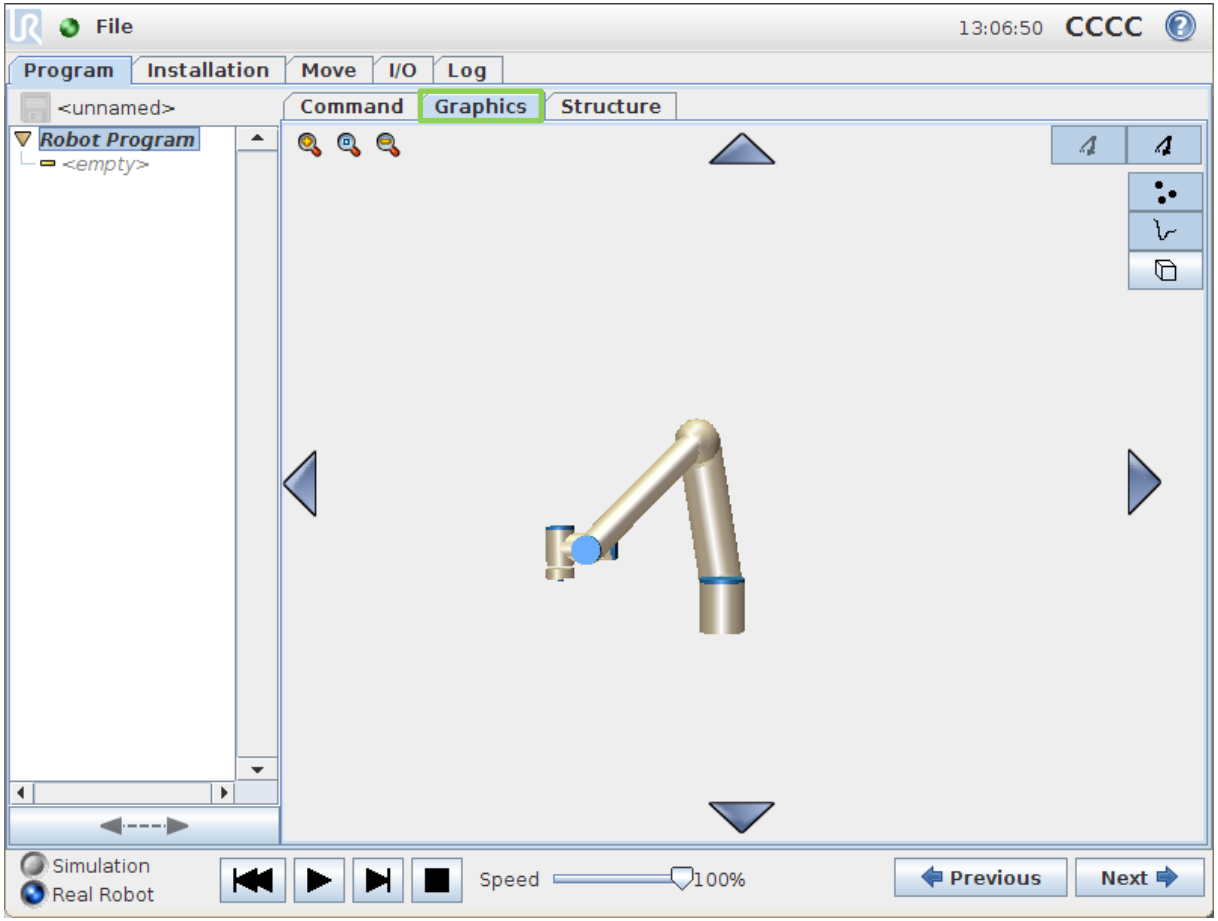
- Program tree
  - Structure of program
  - Contains all commands
  - Executed sequentially
- Command tab
  - Edit of selected command
  - Set general settings for program



UNIVERSAL ROBOTS

## Graphics tab

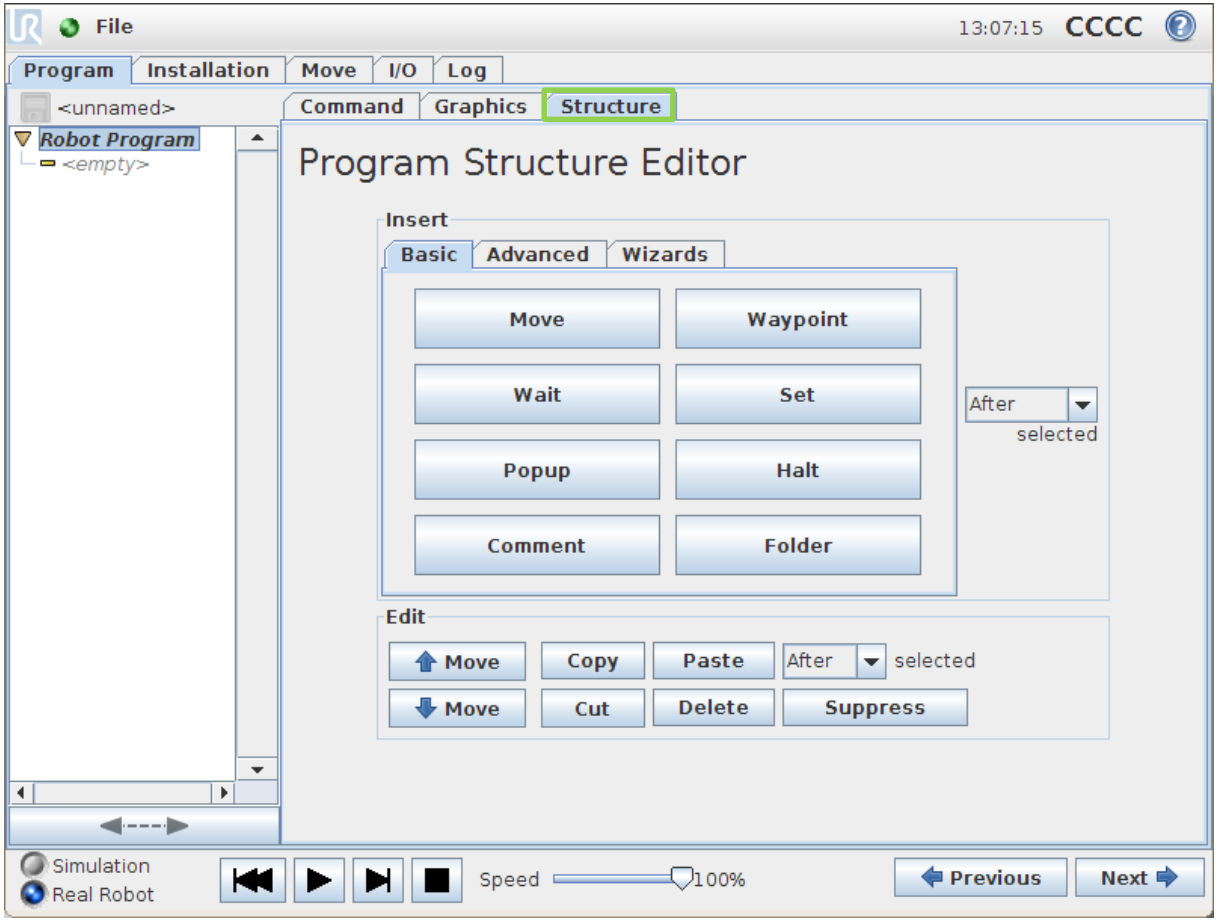
- Graphics tab
  - Actual position
  - Taught positions
  - Trajectories



UNIVERSAL ROBOTS

## Structure tab

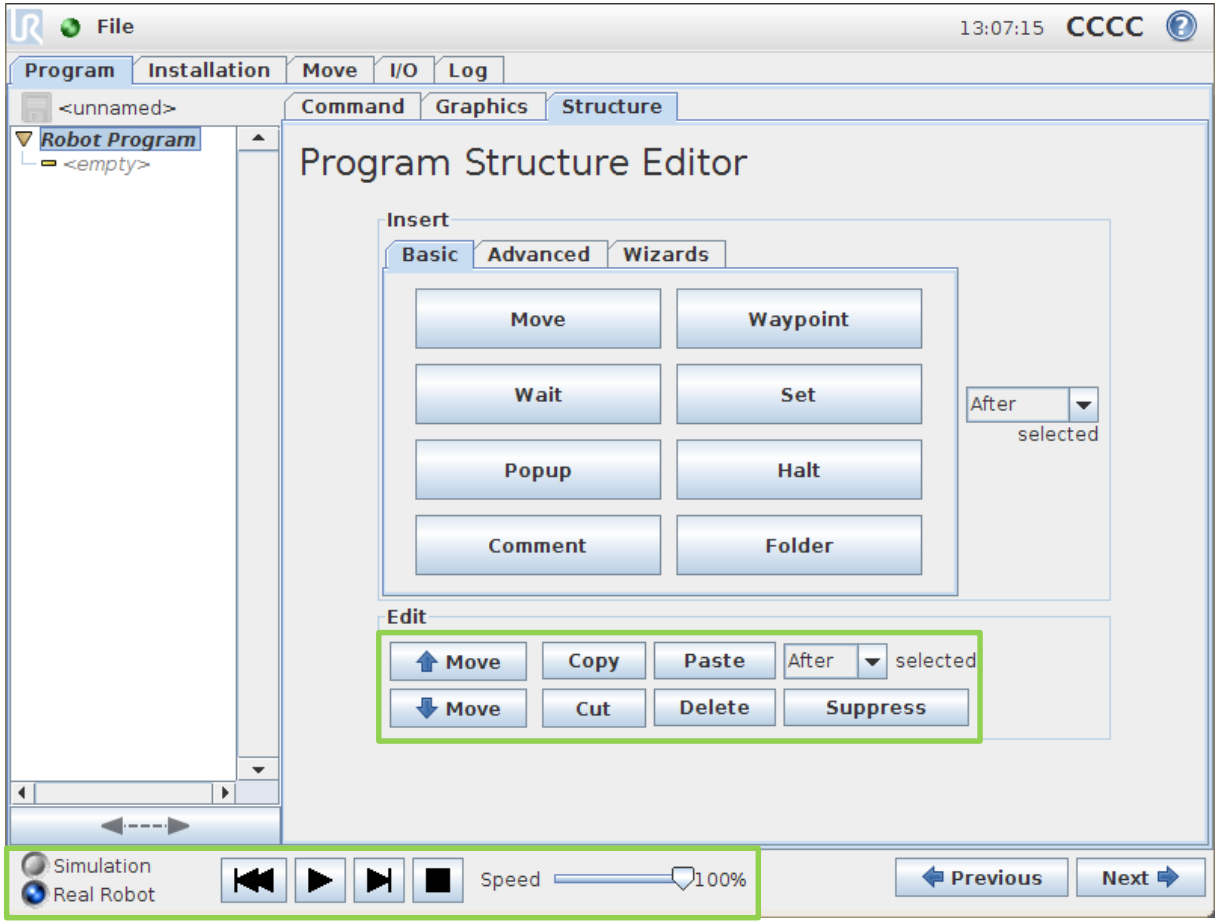
- Structure tab
  - Contains all commands
  - Split into
    - Basic
    - Advanced
    - Wizards



UNIVERSAL ROBOTS

# Editing area / Dashboard

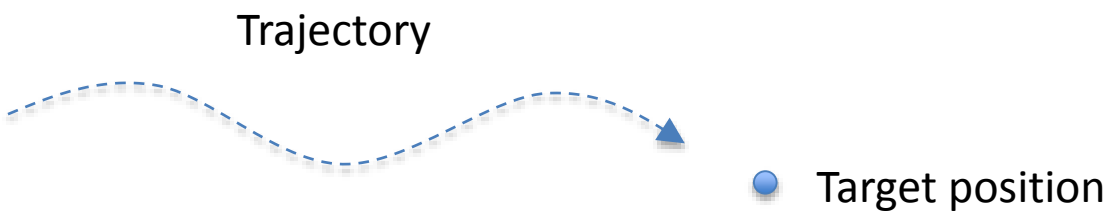
- Editing area
  - Edit and organize program tree
  - Editing tools
  - Suppress function
- Dashboard
  - Control program execution
  - Simulation / Real Robot



UNIVERSAL ROBOTS

## Insert motion

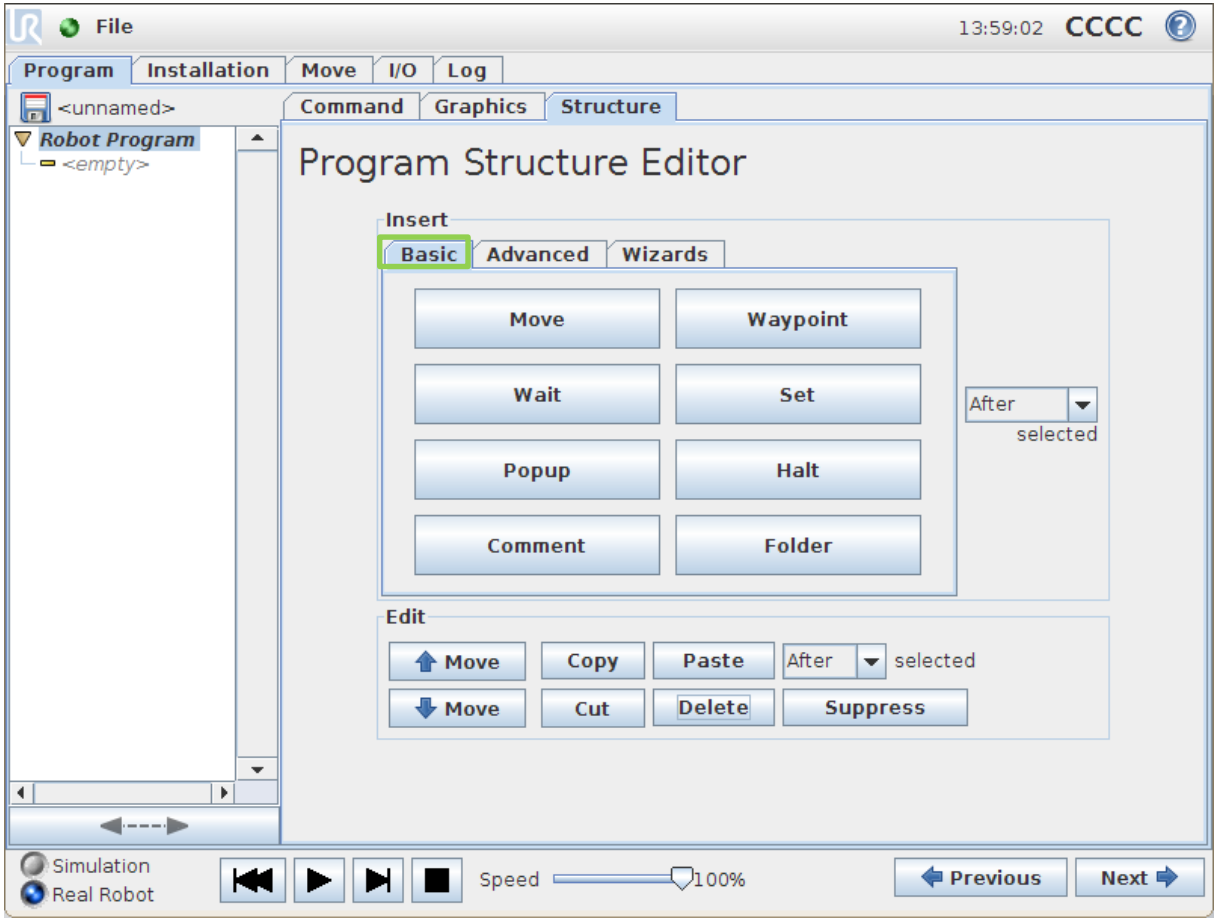
- **Waypoint** specifies the target position
- **Move** specifies the trajectory robot will follow on it's way to target position



UNIVERSAL ROBOTS

## Basic commands

- Basic commands
  - Perform simple operations

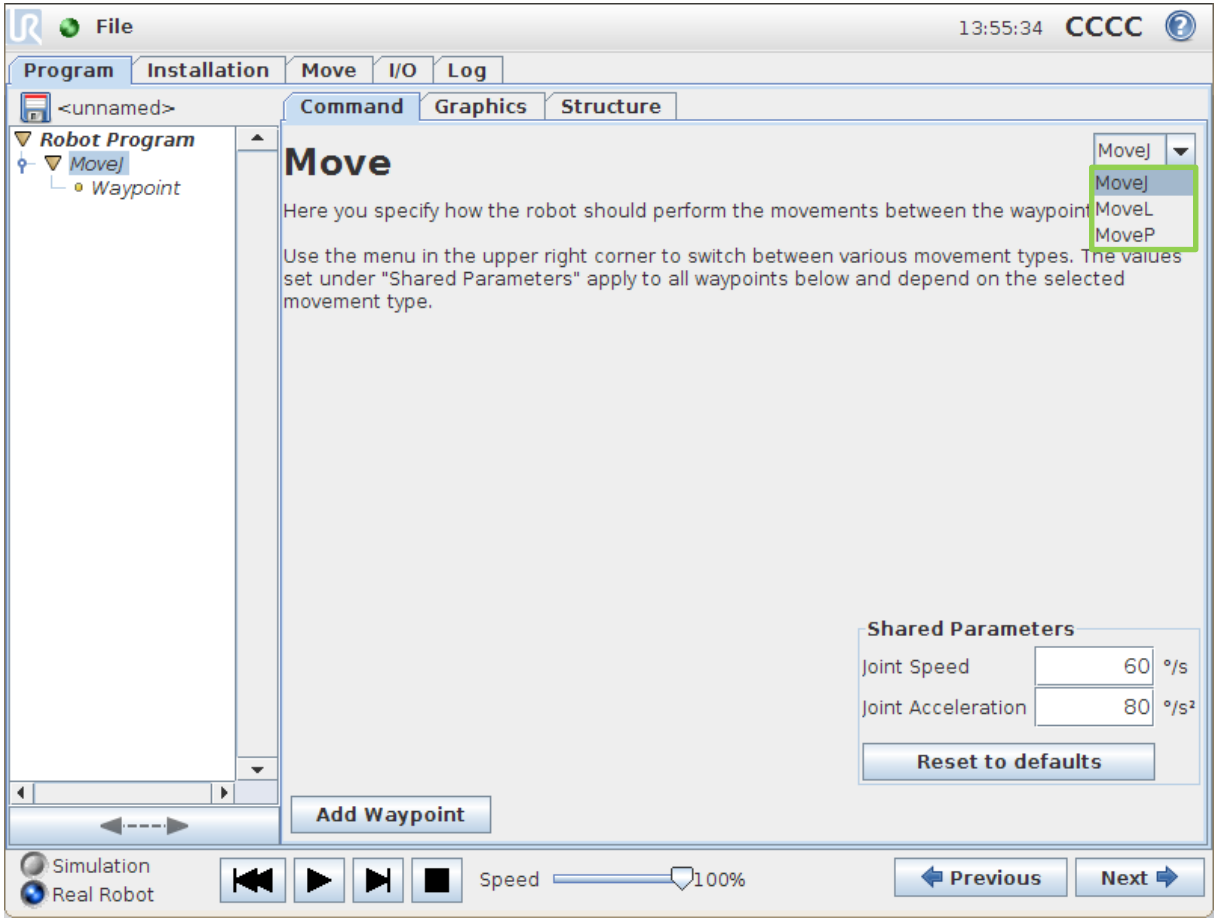



- Insert Move command

UNIVERSAL ROBOTS

## Move command

- Movement types
  - MoveJ *(default)*
  - MoveL
  - MoveP
    - MoveC
- Shared parameters
  - Joint speed
    - Default: 60 °/s
  - Joint acceleration
    - Default: 80 °/s²



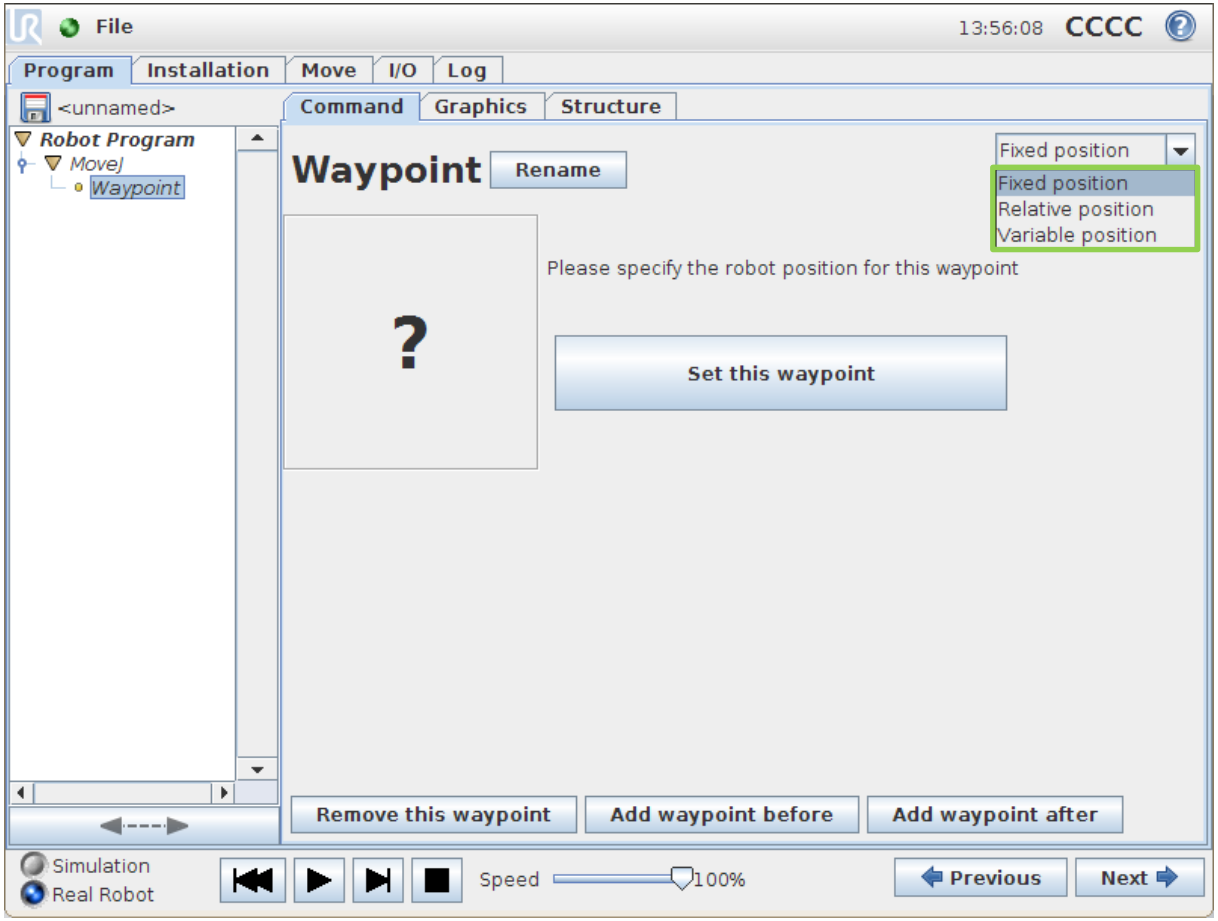

- Waypoint automatically added when inserting Move command

UNIVERSAL ROBOTS

## Waypoint

- Waypoint types
  - Fixed *(default)*
  - Relative
  - Variable

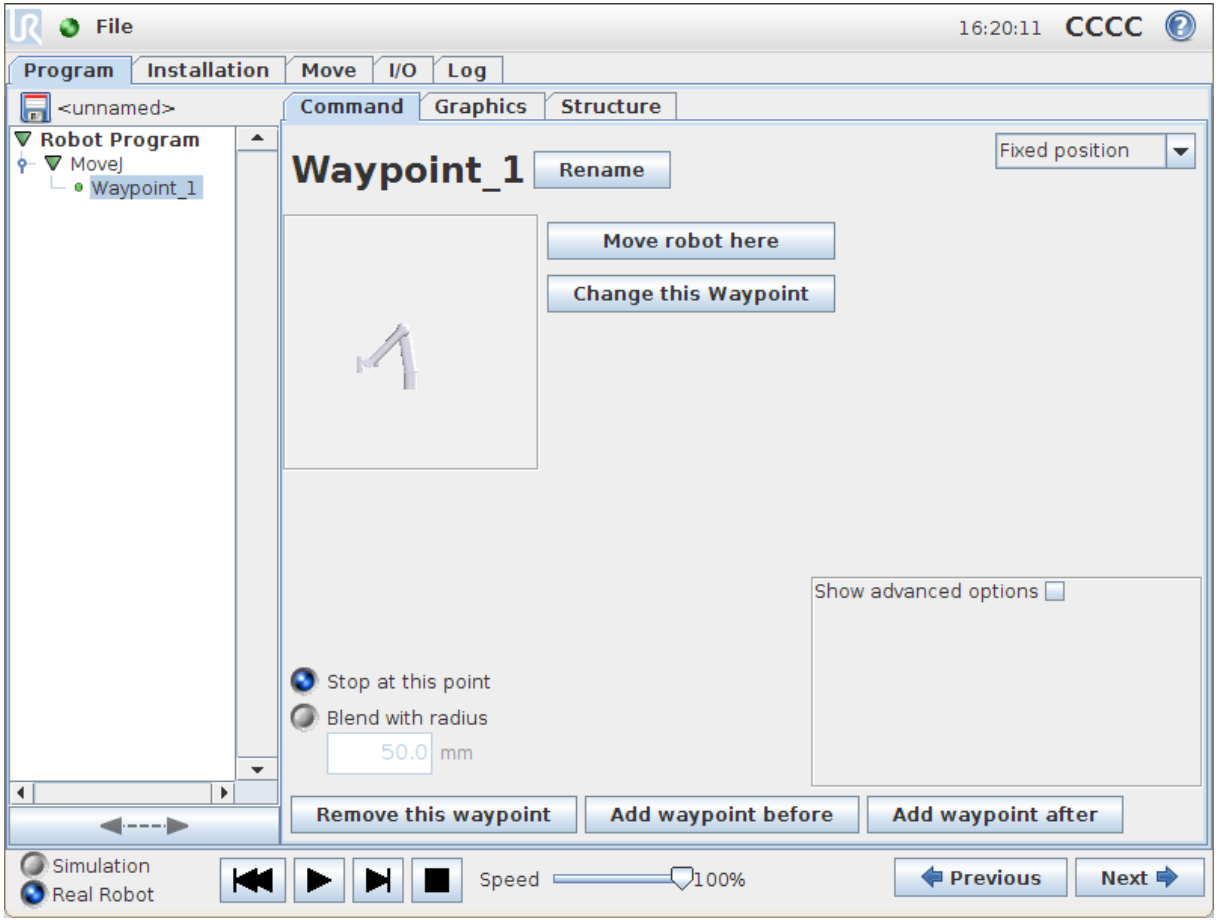
- Teach Waypoint
  - Press *Set this Waypoint*
  - Teach position and press OK for saving



UNIVERSAL ROBOTS

## Waypoint

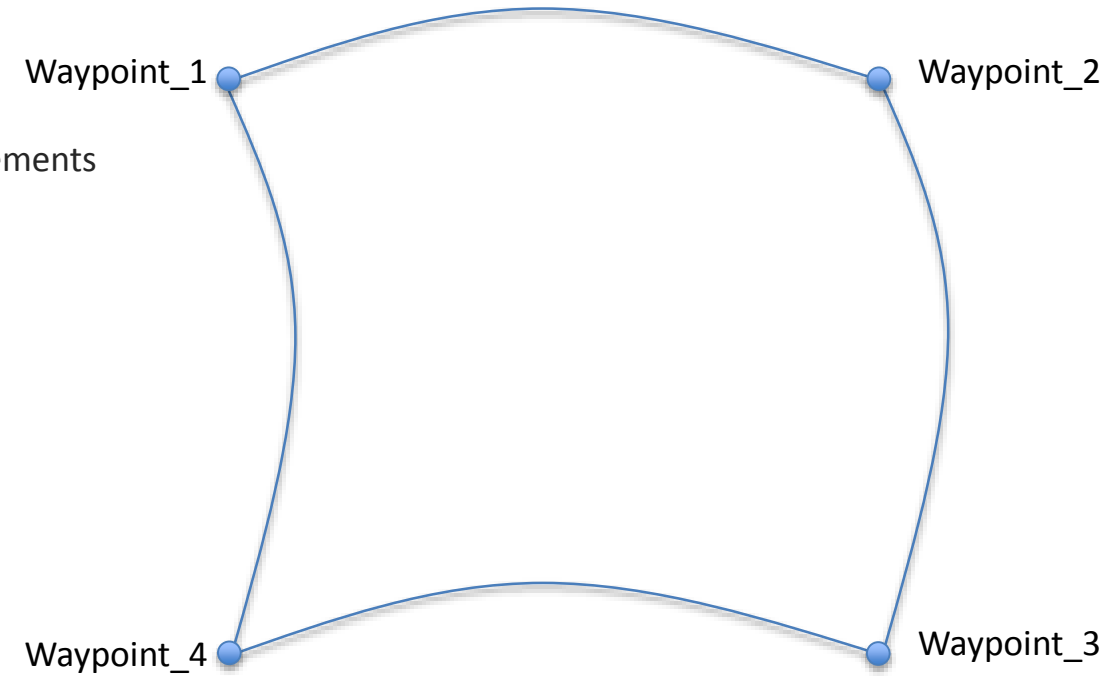
- Options
  - Rename Waypoint
    - Improves readability
  - Move robot here
    - Moves to position
  - Change this Waypoint
    - Modify position



UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

## MoveL

- Linear movement
  - Interpolation on
  - Linear trajectory for TCP

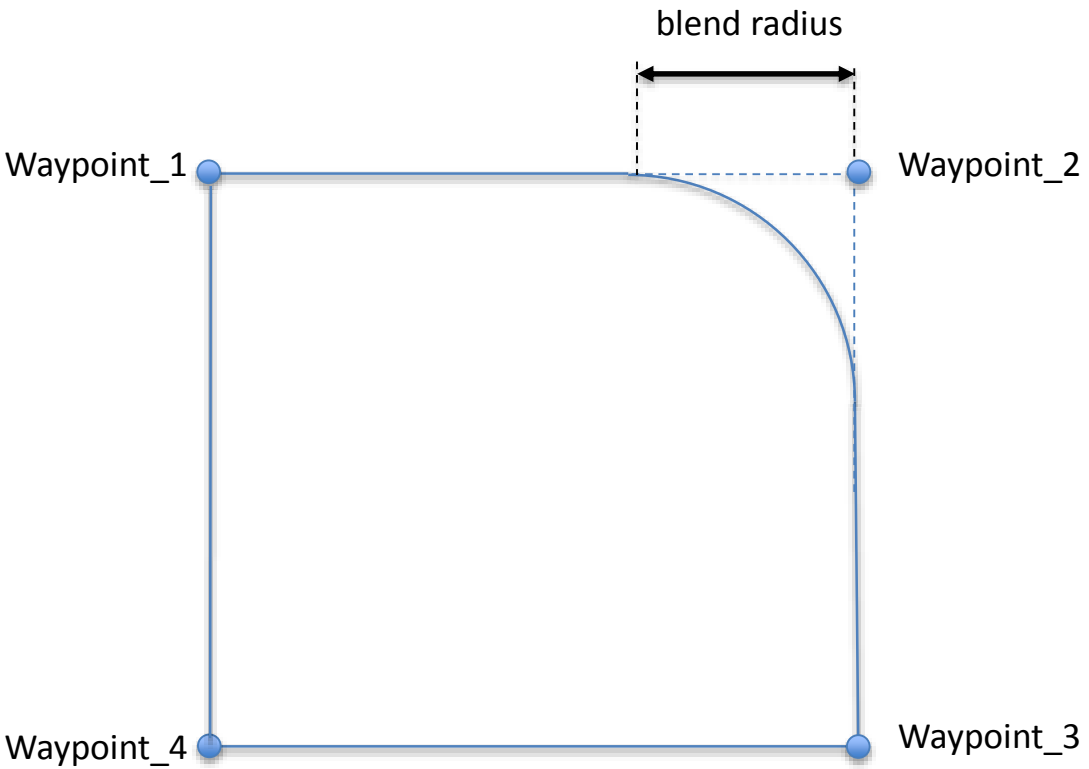

Robot Program
  MoveL
    Waypoint_1
    Waypoint_2
    Waypoint_3
    Waypoint_4

- Save sample program as movel.urp

UNIVERSAL ROBOTS

# MoveL with blend

- Blend radius
  - Continuous movement
  - No stop in Waypoint



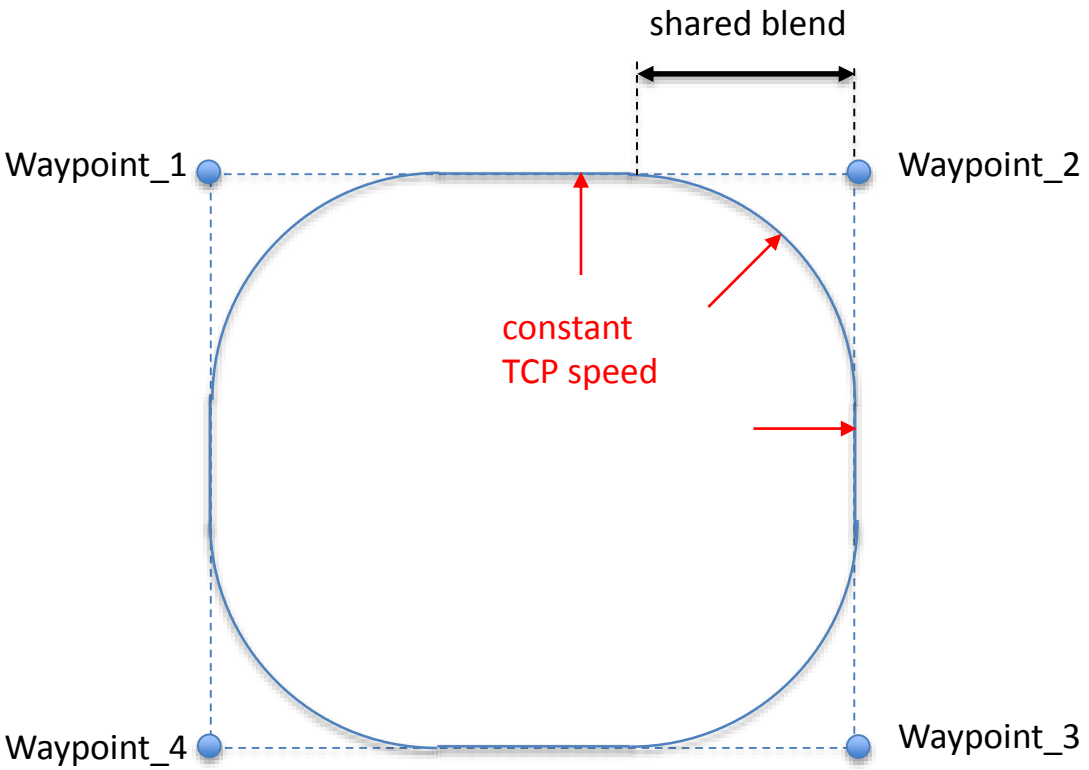

Robot Program
  MoveL
    Waypoint_1
    Waypoint_2 (r=25)
    Waypoint_3
    Waypoint_4

- Save sample program as move_with_blend.urp

UNIVERSAL ROBOTS

## MoveP

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



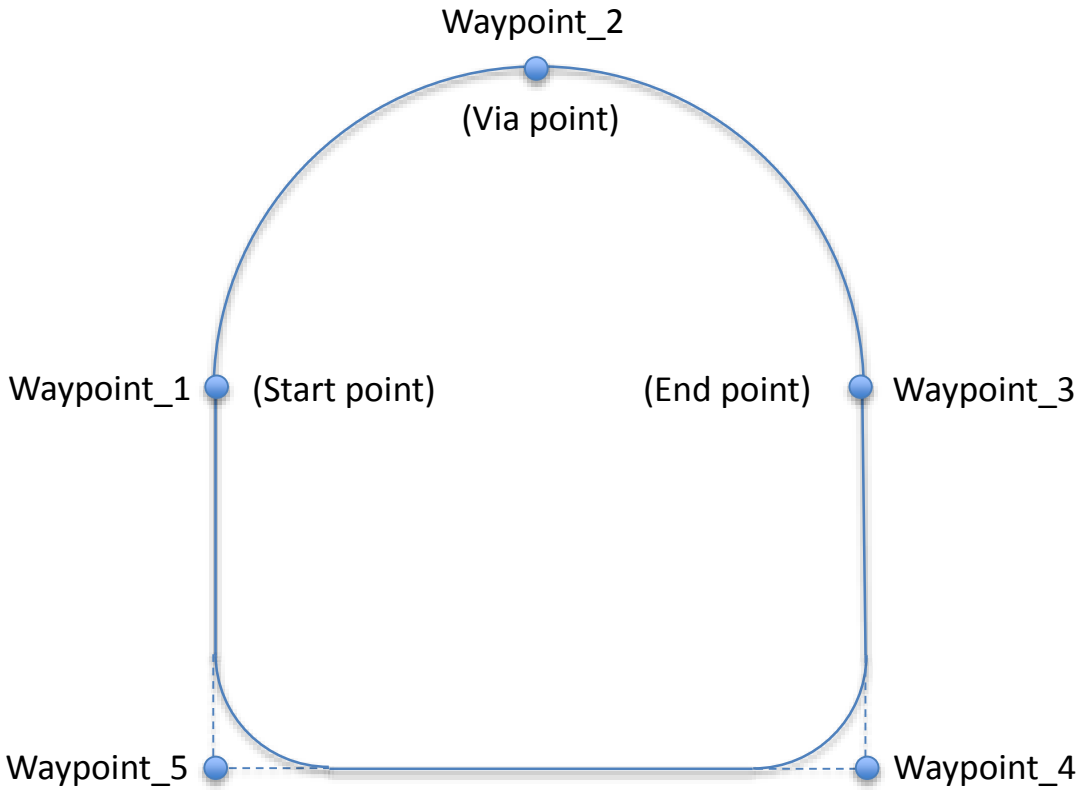

```
Robot Program
    MoveP
        Waypoint_1
        Waypoint_2
        Waypoint_3
        Waypoint_4
```

- Save sample program as movep.urp

UNIVERSAL ROBOTS

## MoveC

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



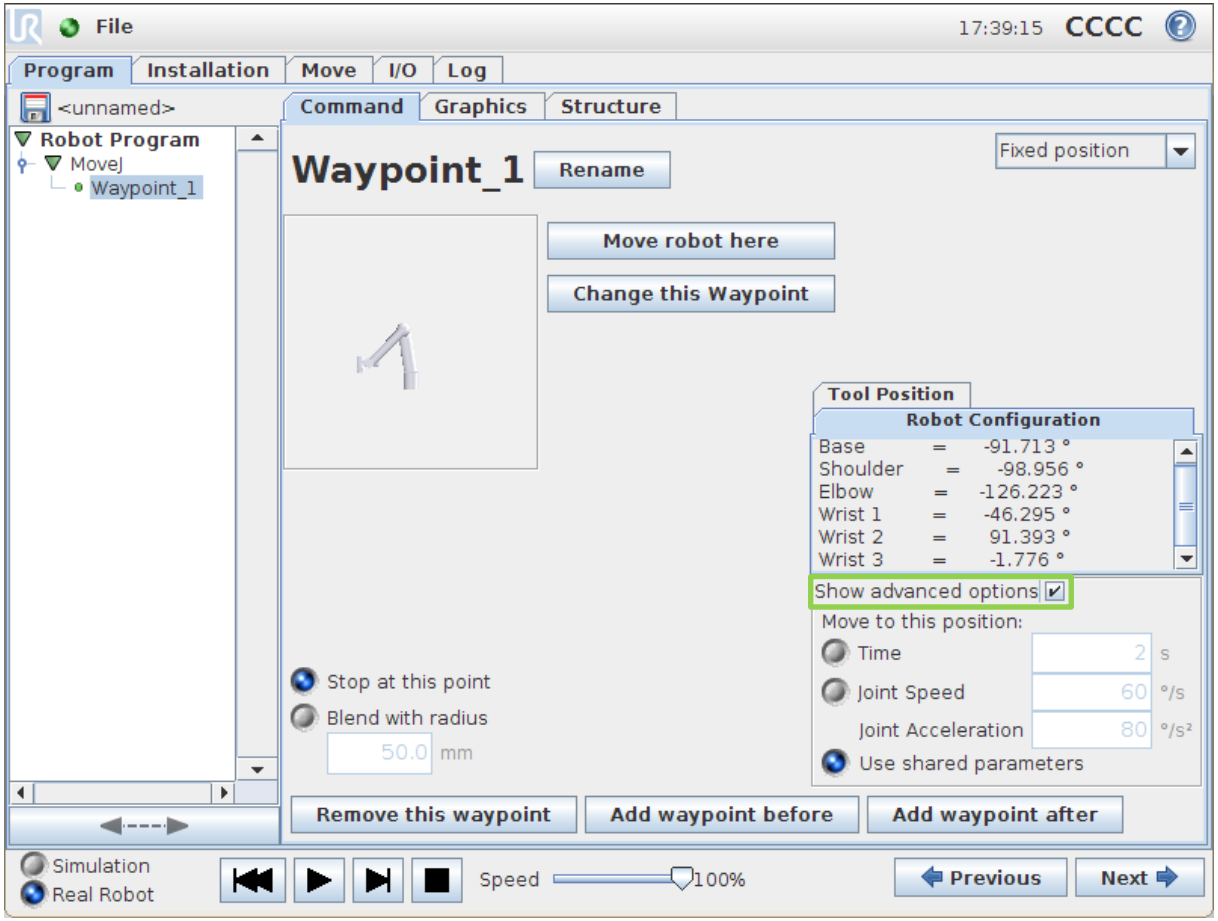

```
Robot Program
    MoveP
        Waypoint_1
        CircleMove
                Waypoint_2
                Waypoint_3
        Waypoint_4
        Waypoint_5
```

- Save sample program as movec.urp

UNIVERSAL ROBOTS

## Advanced options

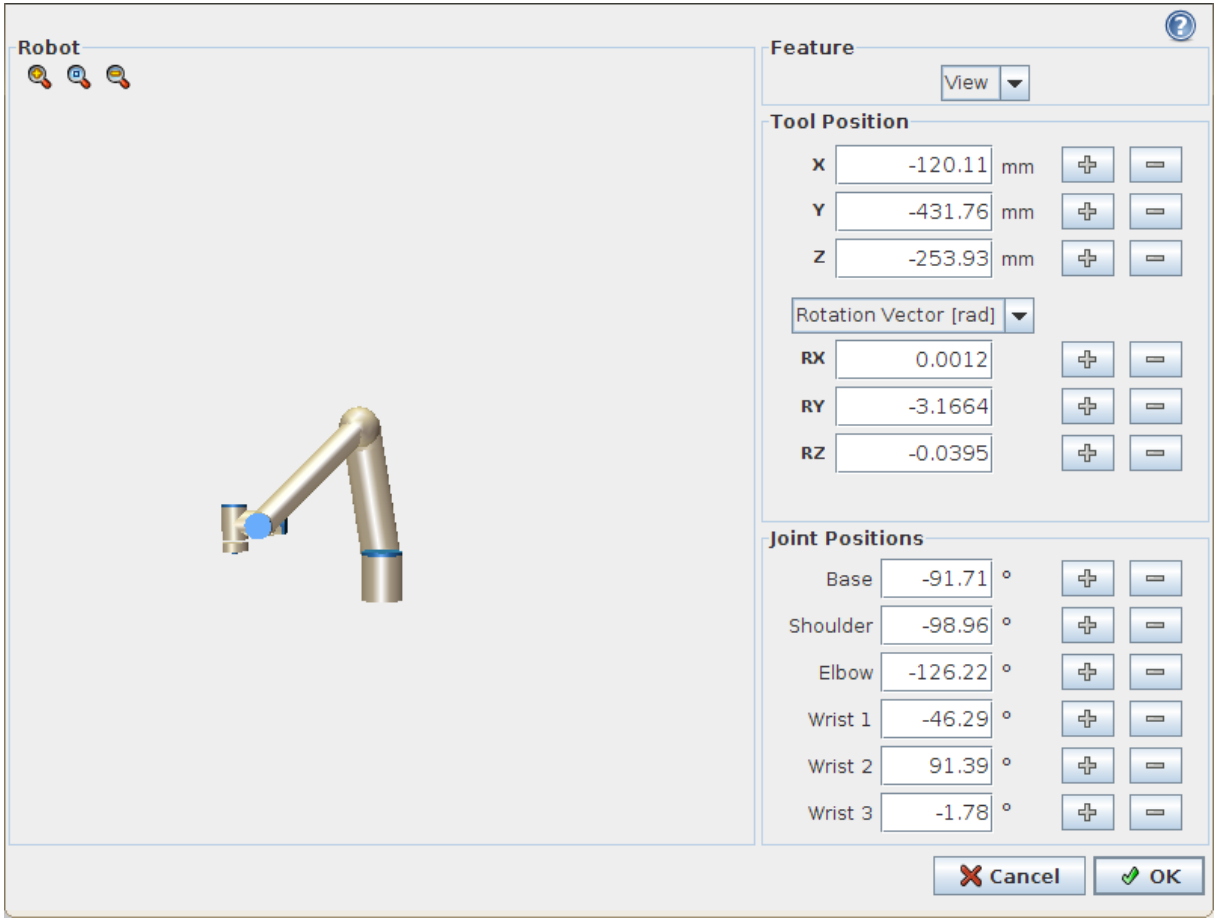
- Advanced options
  - Set individual
    - Speed
    - acceleration
  - Waypoint position
    - Joint angles
    - Tool position



UNIVERSAL ROBOTS

## Pose editor

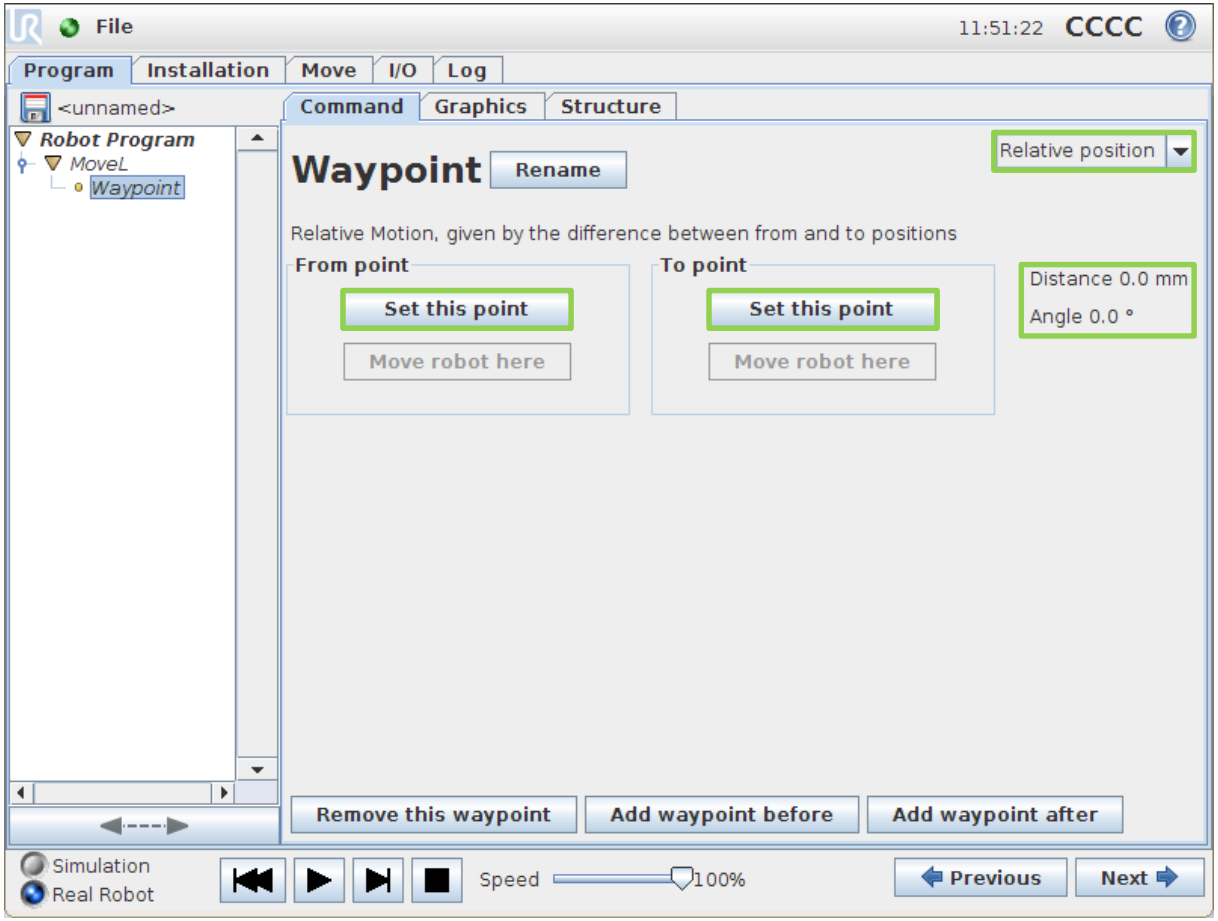
- Modify
  - Absolute value
  - Add/subtract to value
- Joint positions
  - Angular value in *degrees*
- Tool position
  - Cartesian value in *mm.*
  - Define rotation unit



UNIVERSAL ROBOTS

## Relative waypoint

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



UNIVERSAL ROBOTS

# Relative waypoint

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

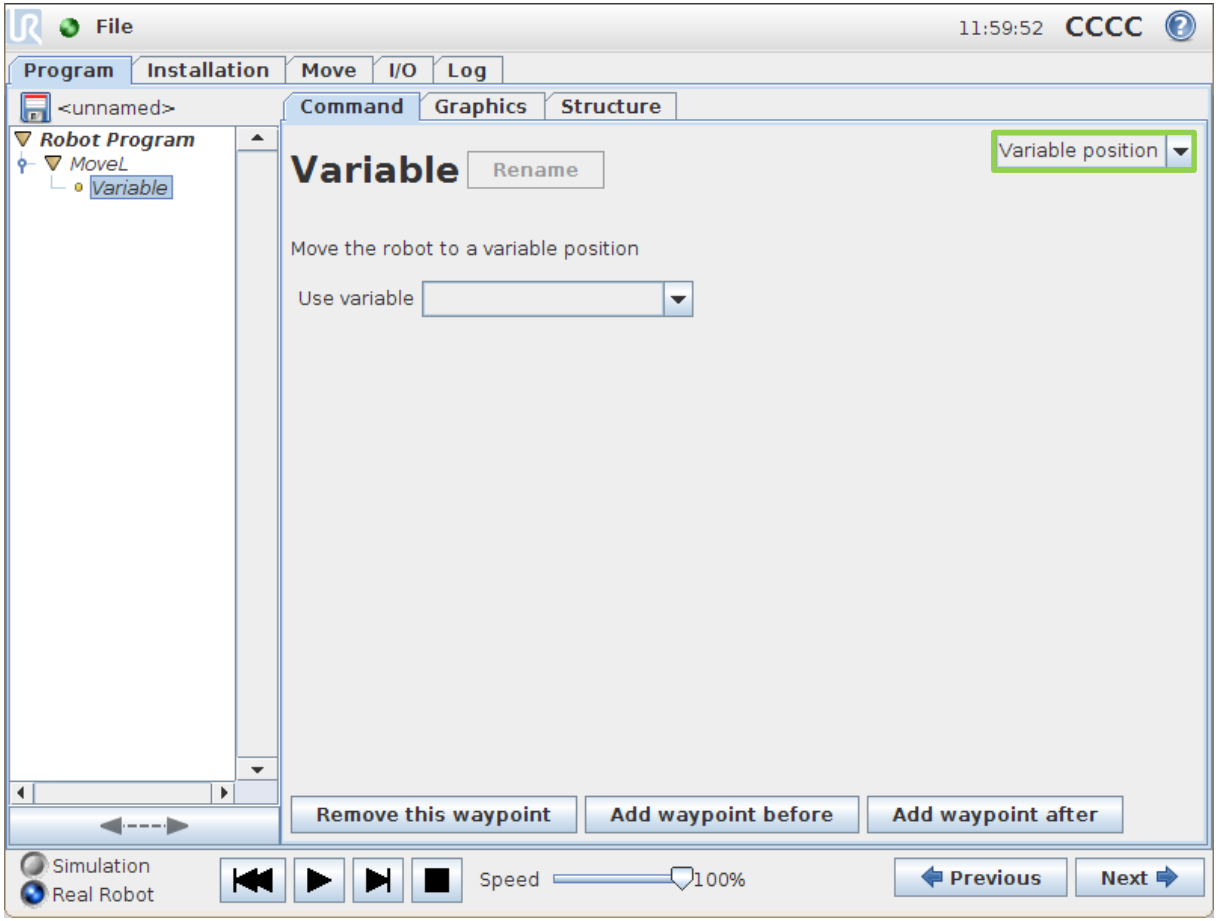

Robot Program
  MoveL
    Waypoint_1
    Waypoint_2 *rel. pos*
    Waypoint_3 *rel. pos*
    Waypoint_4 *rel. pos*

- Save sample program as movel_with_relative_waypoint.urp

UNIVERSAL ROBOTS

# Variable waypoint

- Variable position
  - Position that can be calculated / offset

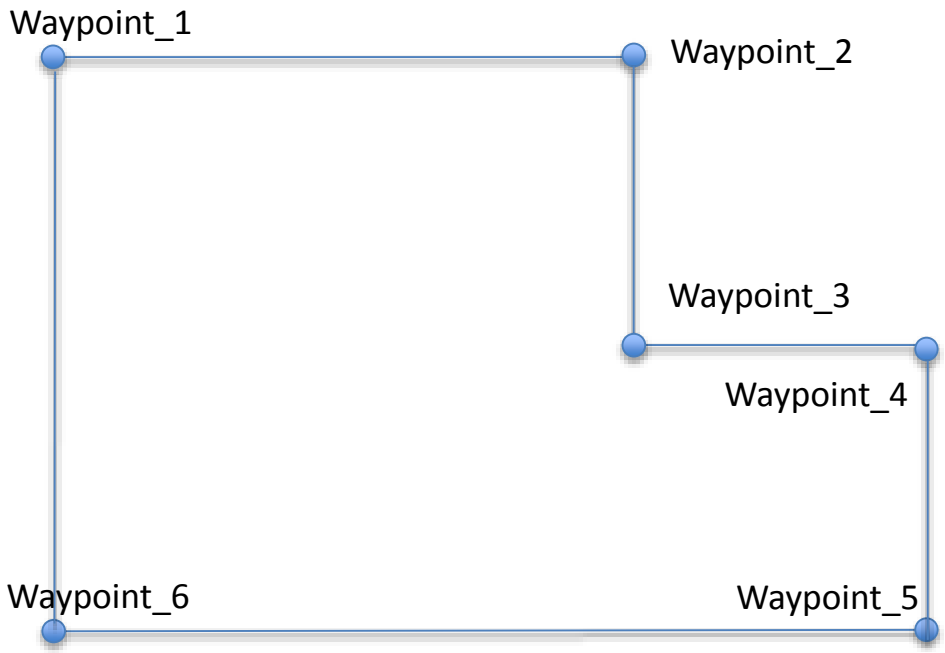



- Definition of and how to use variable Waypoints is part of Advanced Training

UNIVERSAL ROBOTS

## Lab exercise part 1

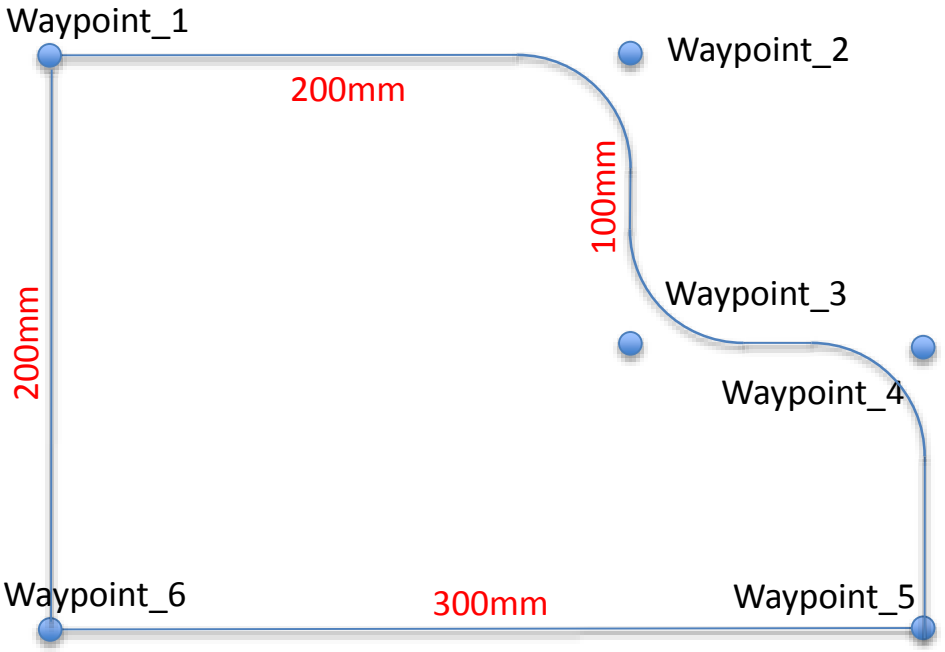
- Create a MoveL
  - Create 6 waypoints in positions like those in the diagram.
  - Use the move tab to move the robot tool into appropriate positions
  - This shape can be defined anywhere in the robot workspace



UNIVERSAL ROBOTS

## Lab exercise part 2

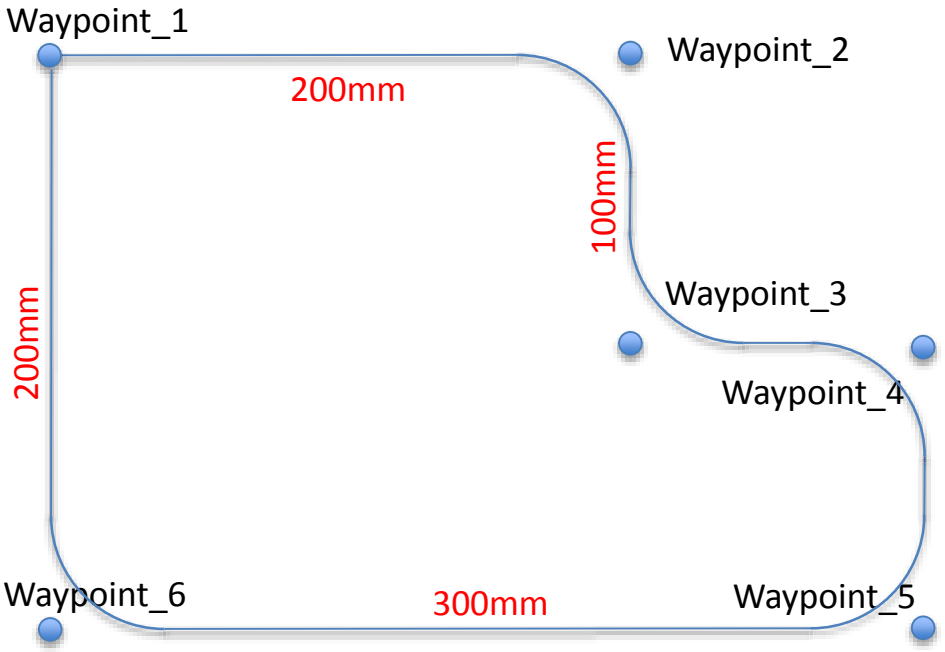
- Adjust the waypoints
  - Adjust the waypoints according to the dimensions on the diagram using the pose editor.
  - Add a 50mm radius to waypoints 2, 3 and 4.



UNIVERSAL ROBOTS

## Lab exercise part 2

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



UNIVERSAL ROBOTS

- 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

UNIVERSAL ROBOTS

## next

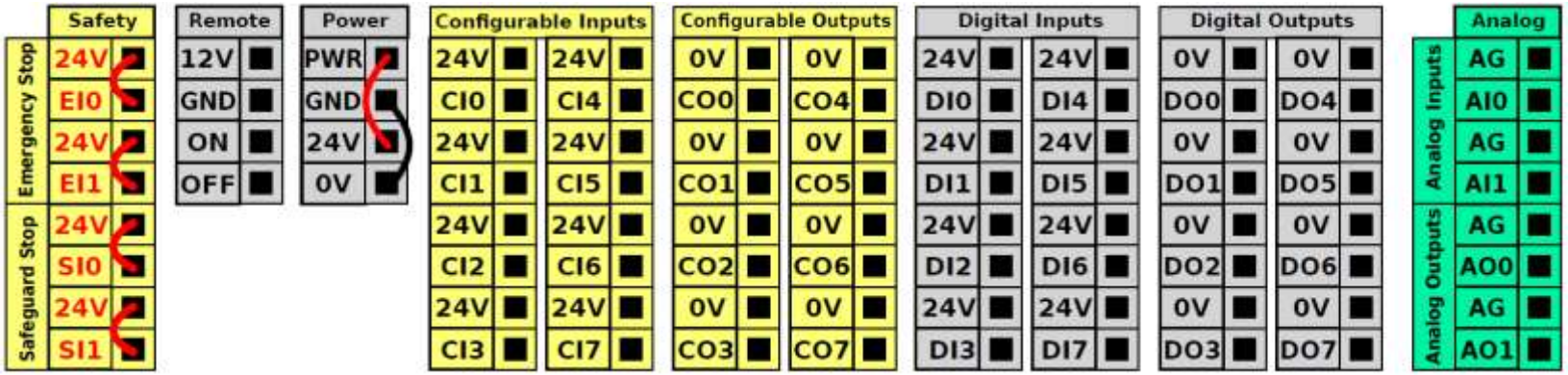
- Signal handling
  - Interaction with external devices

UNIVERSAL ROBOTS

## 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

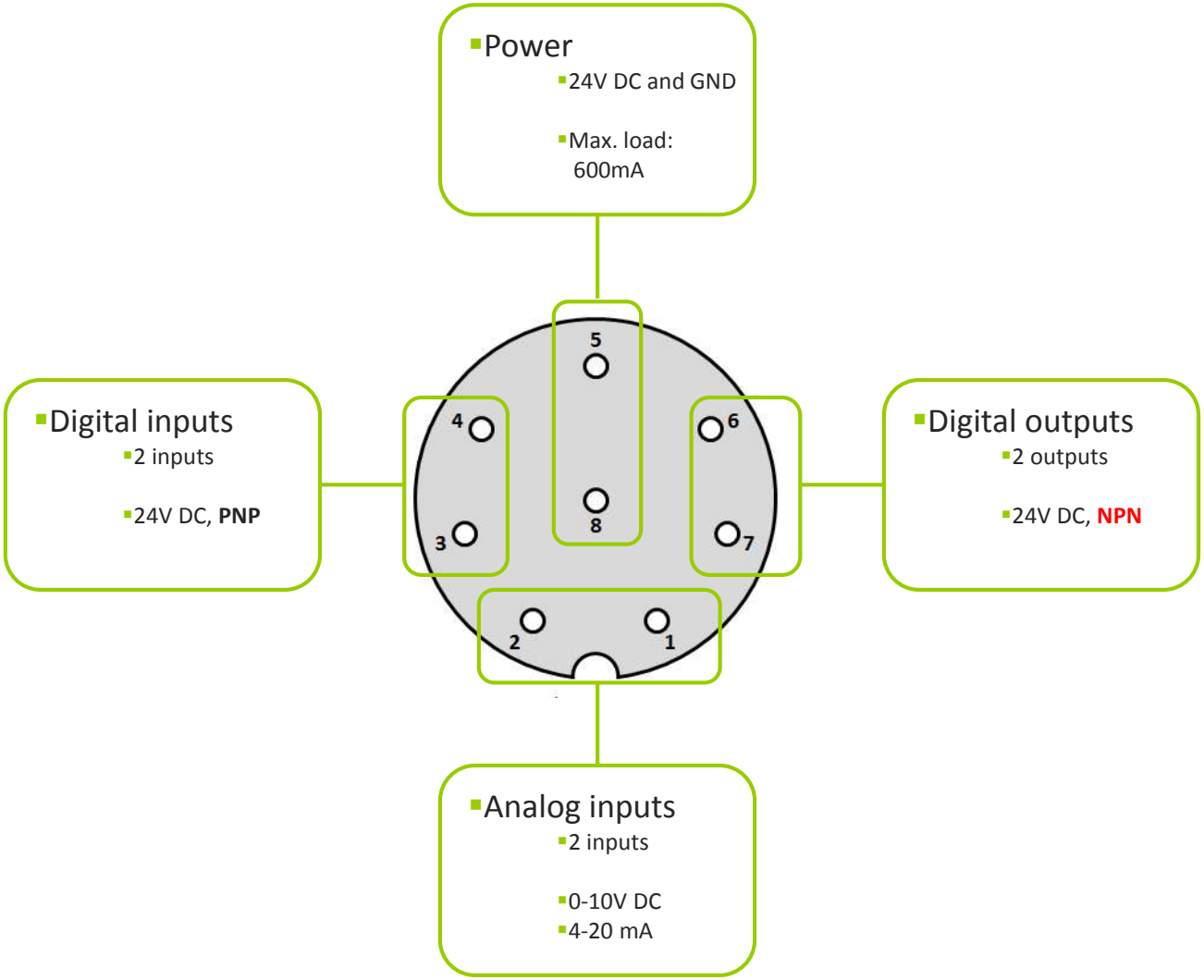



- *Consult User Manual, chapter 4. Electrical Interface*

UNIVERSAL ROBOTS

## User interface

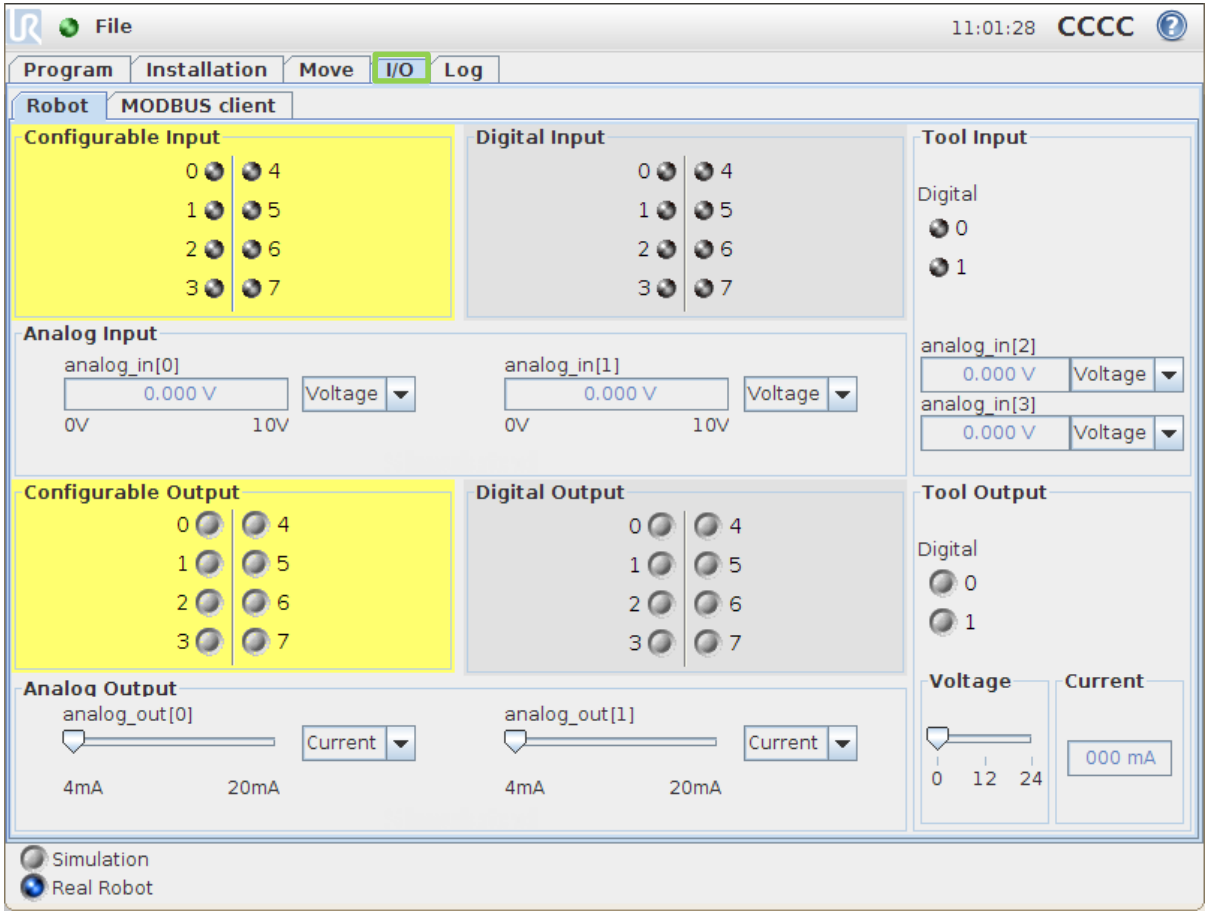
- Tool connector
  - pin 1 AI[2]
  - pin 2 AI[3]
  - pin 3 TI[0]
  - pin 4 TI[1]
  - pin 5 24V DC
  - pin 6 TO[0]
  - pin 7 TO[1]
  - pin 8 GND



UNIVERSAL ROBOTS

# Signal handling

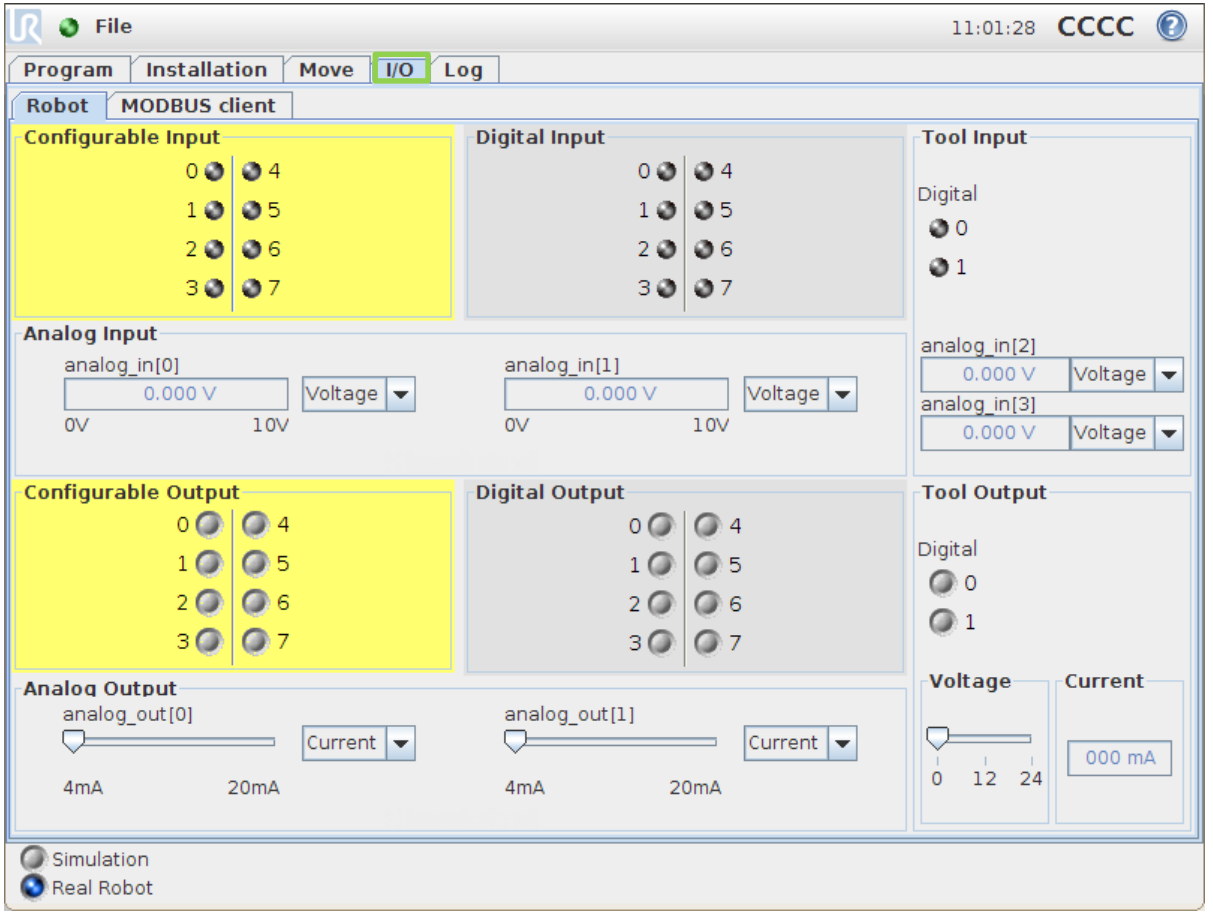
- Purpose
  - Interaction with external devices
  - Hardwired to
    - Control box
    - Tool connector
  - Configuration
    - Control box
      - 16 DI
      - 16 DO
      - 2 AI
      - 2 AO
    - Tool connector
      - 2 DI
      - 2DO
      - 2AI



UNIVERSAL ROBOTS

## Signal handling

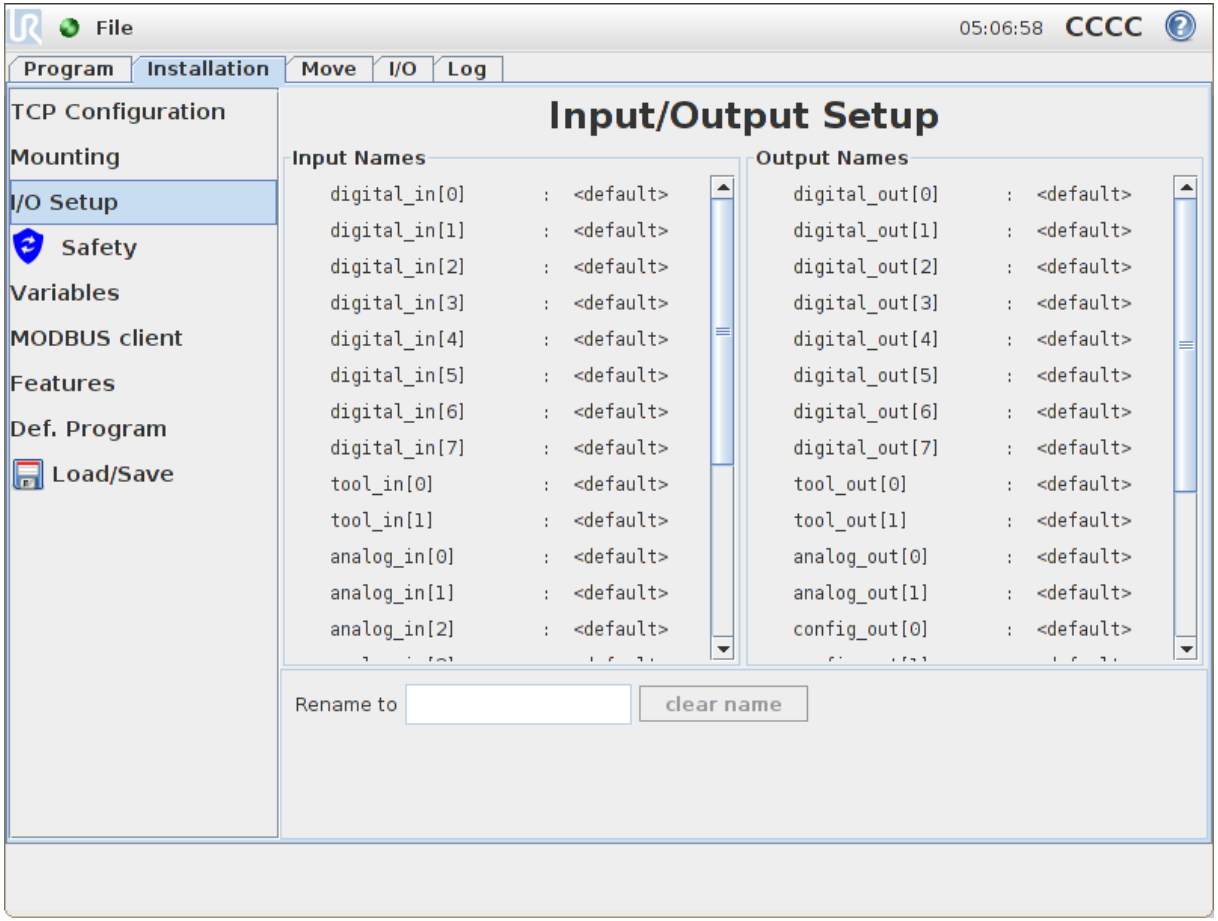
- Digital signals
  - Voltage
    - 24V DC
    - GND
  - State
    - OFF (Low)
    - ON (High)
- Analog signals
  - Range
    - Current 4-20 mA
    - Voltage 0-10 V



UNIVERSAL ROBOTS

# I/O setup

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

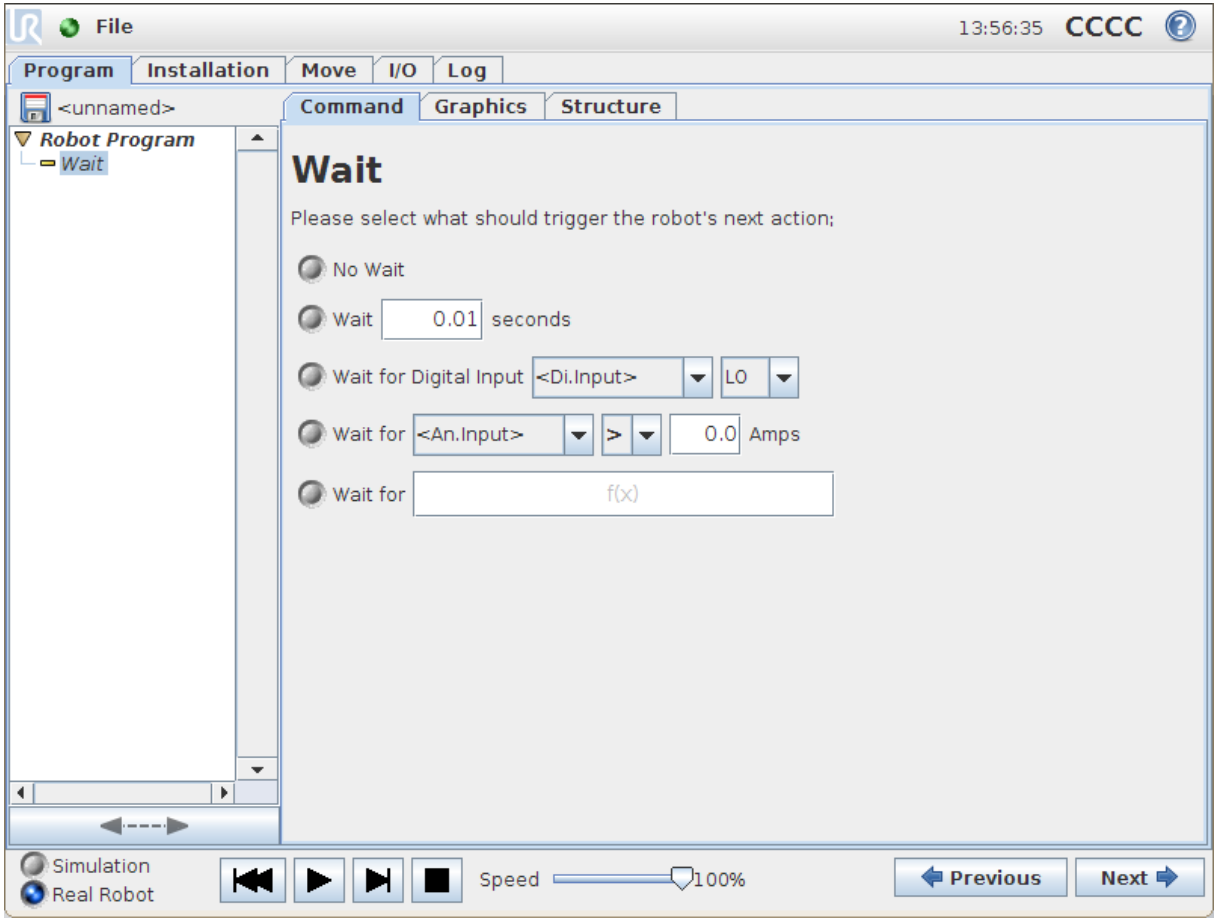



- Program and I/O tab is affected when renaming

UNIVERSAL ROBOTS

# Wait command

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



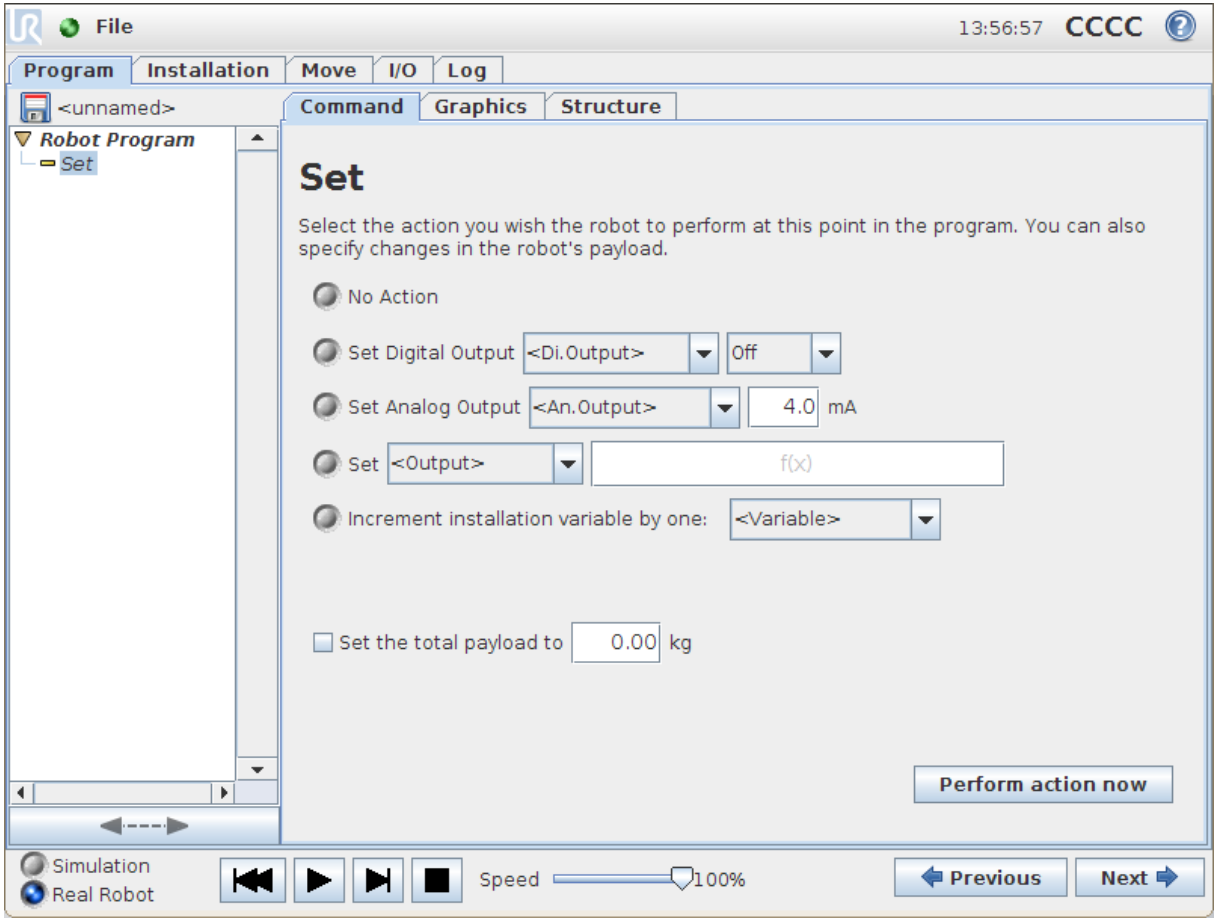

```
Robot Program
    MoveL
        Waypoint_1
        Waypoint_2
        Wait 1.0
        Waypoint_3
        Waypoint_4
        Wait DI[0] = True
```

- Save sample program as wait.urp

UNIVERSAL ROBOTS

# Set command

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



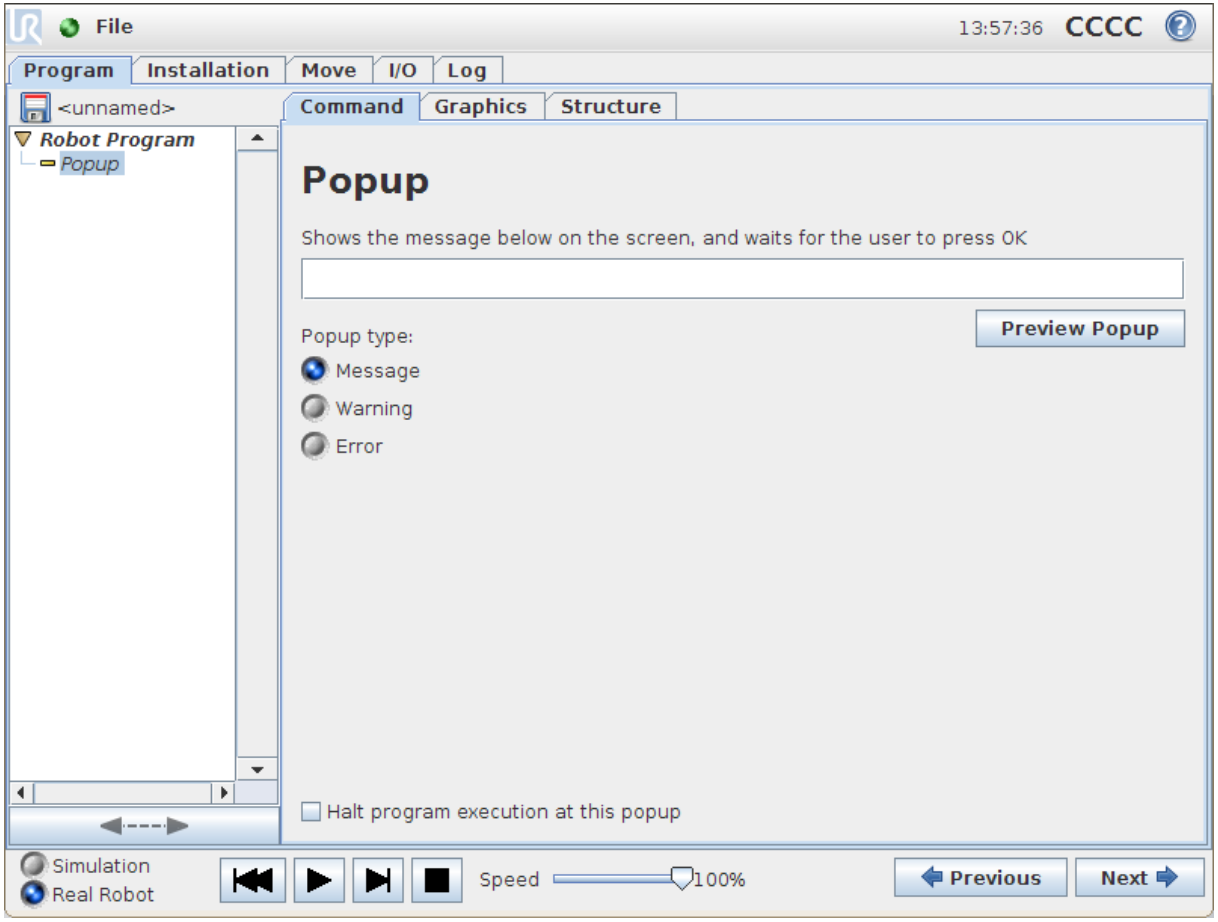

```
Robot Program
    MoveL
        Waypoint_1
        Set DO[0] = True
        Waypoint_2
        Waypoint_3
        Waypoint_4
        Set DO[0] =  False
```

- Save sample program as set.urp

UNIVERSAL ROBOTS

# Popup command

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



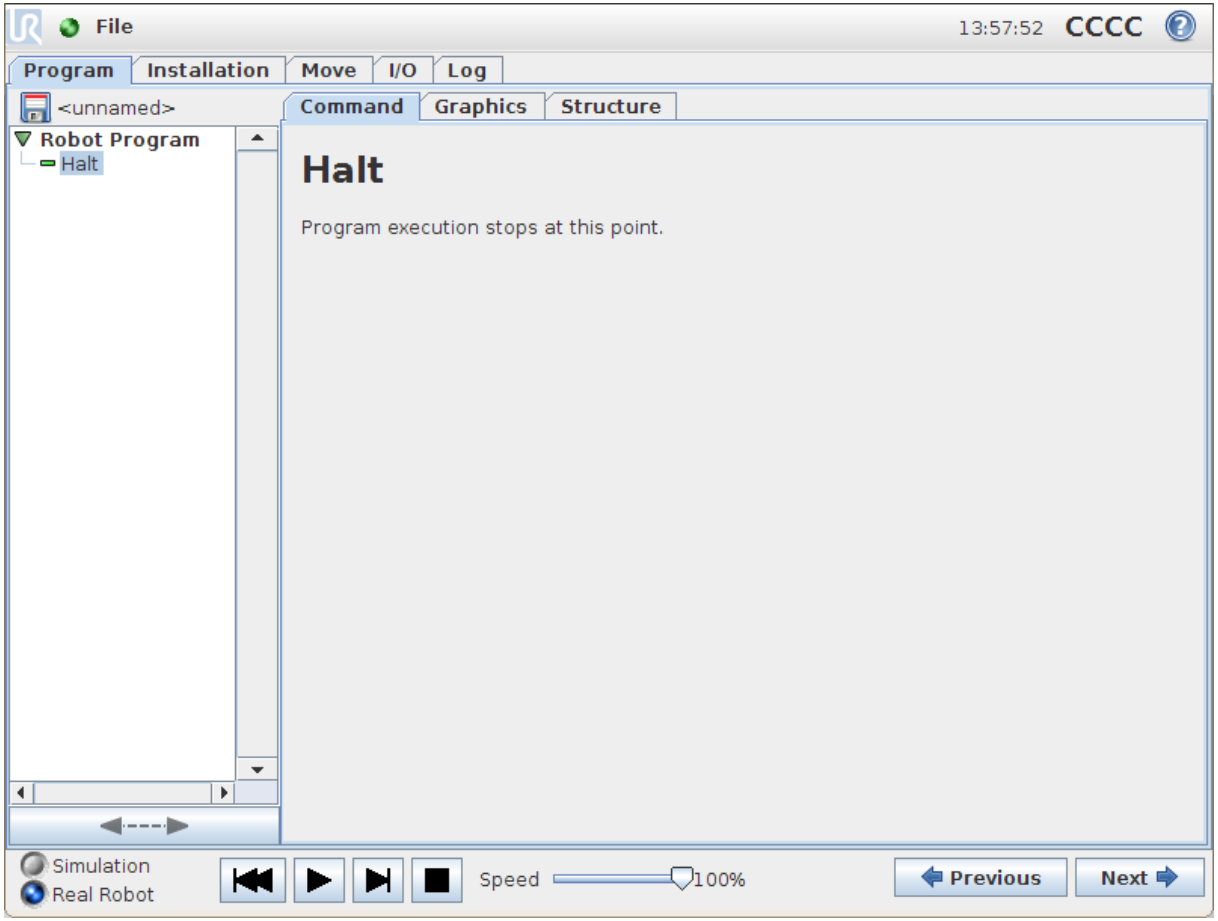

Robot Program
- MoveL
  - Waypoint_1
  - Waypoint_2
  - Wait 1.0
  - Waypoint_3
  - Waypoint_4
  - Popup

- Save sample program as popup.urp

UNIVERSAL ROBOTS

## Stop command

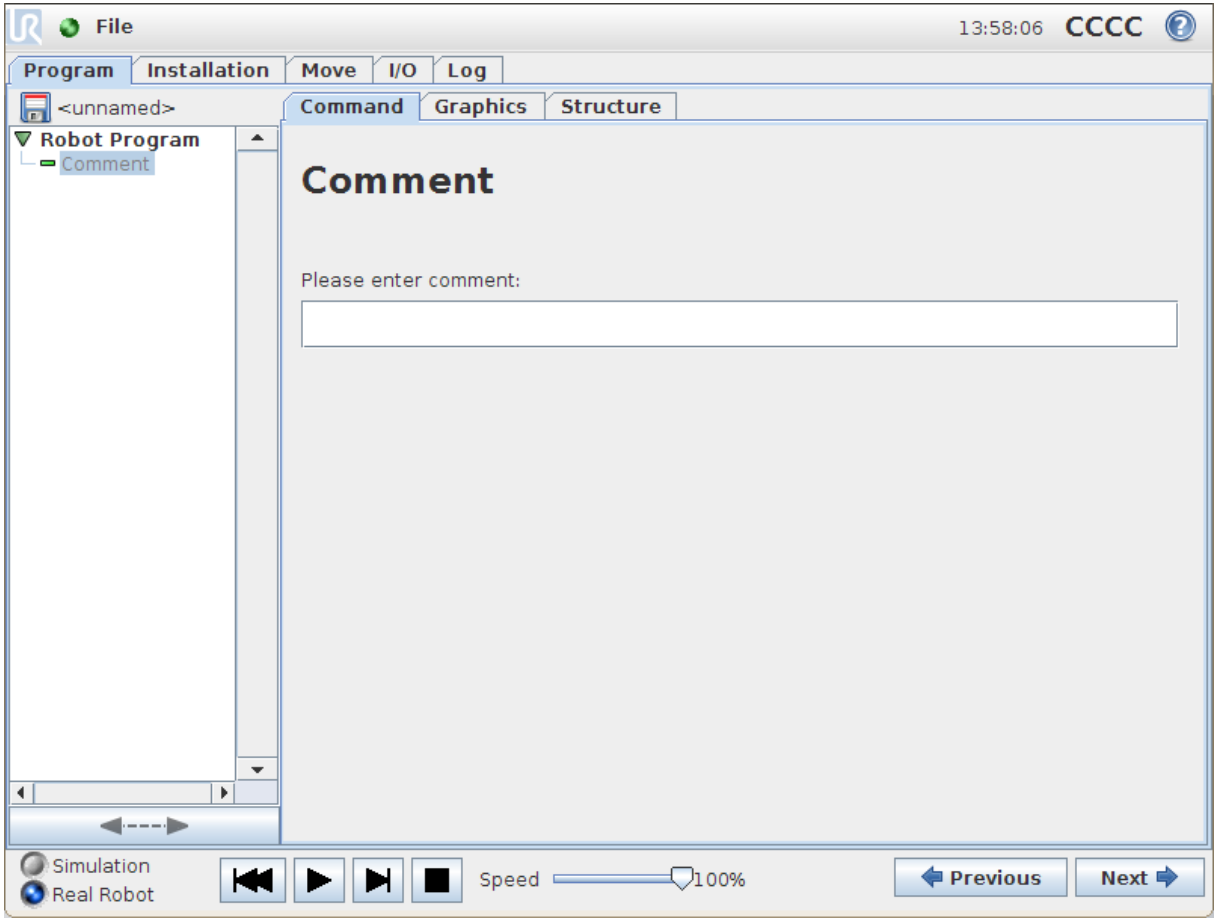
- Halt
  - Stops program execution



UNIVERSAL ROBOTS

## Comment command

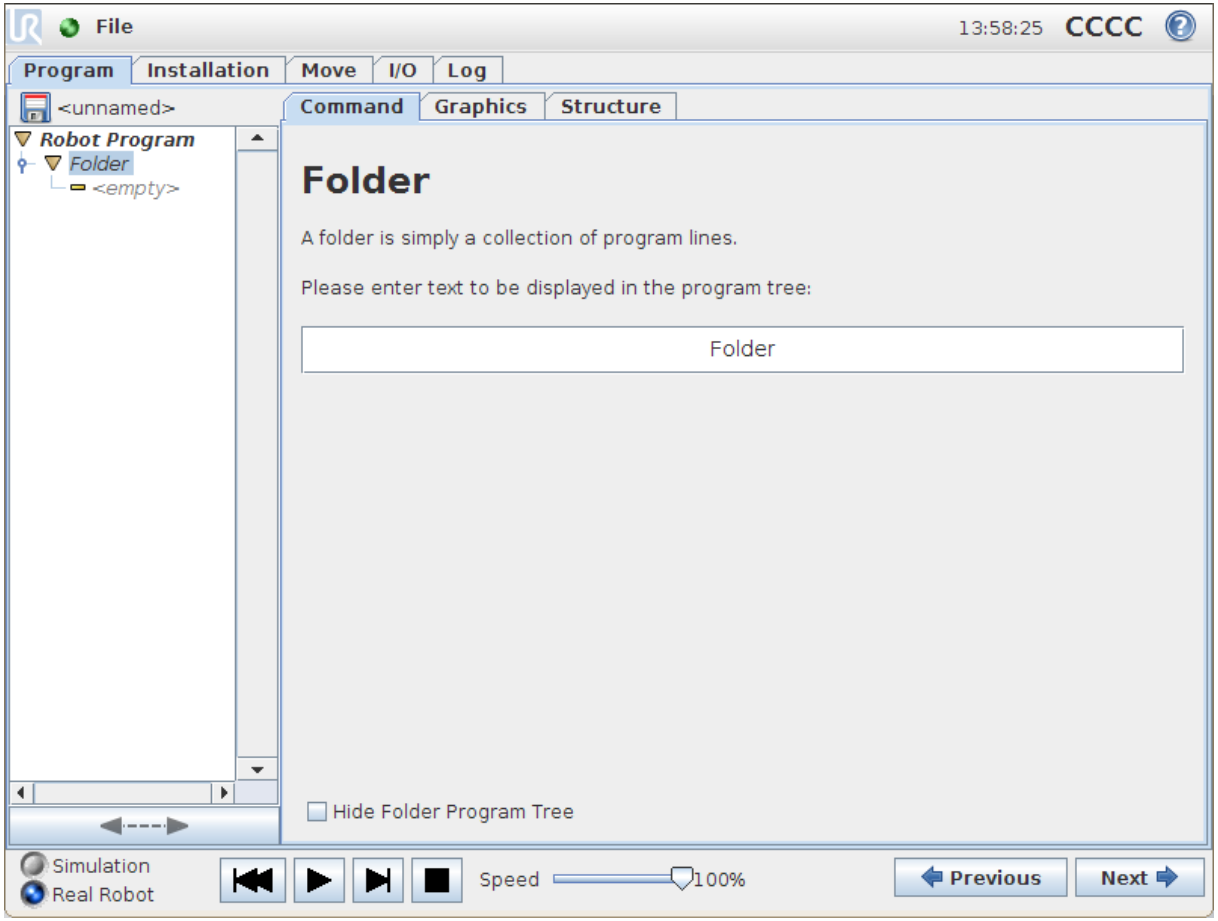
- Comments
  - Add text to program
  - Program execution *not* affected



UNIVERSAL ROBOTS

# Folder command

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

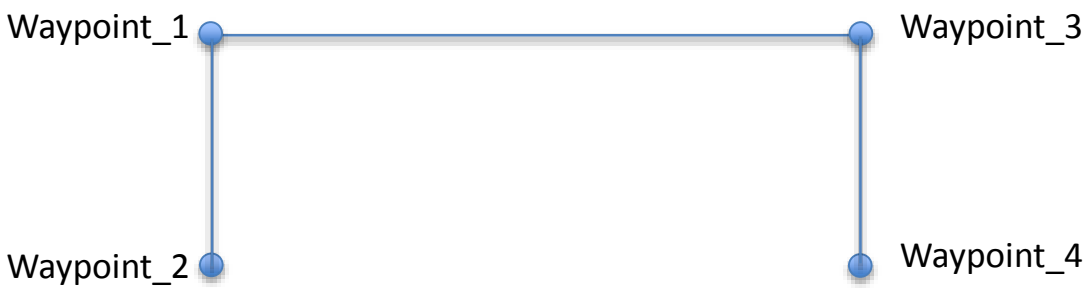


UNIVERSAL ROBOTS

## 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
```

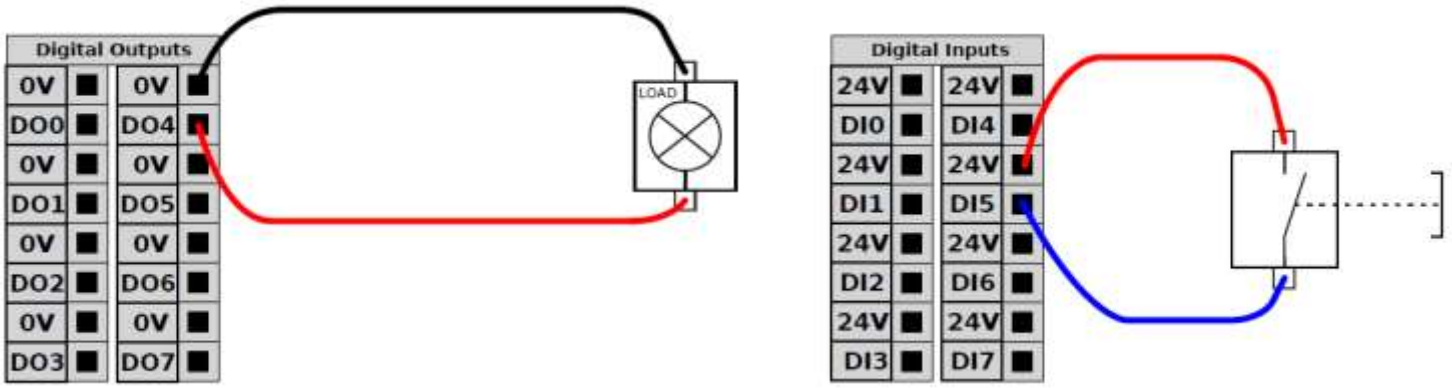



- Save sample program as pick_and_place.urp

UNIVERSAL ROBOTS

# Lab exercise part 1

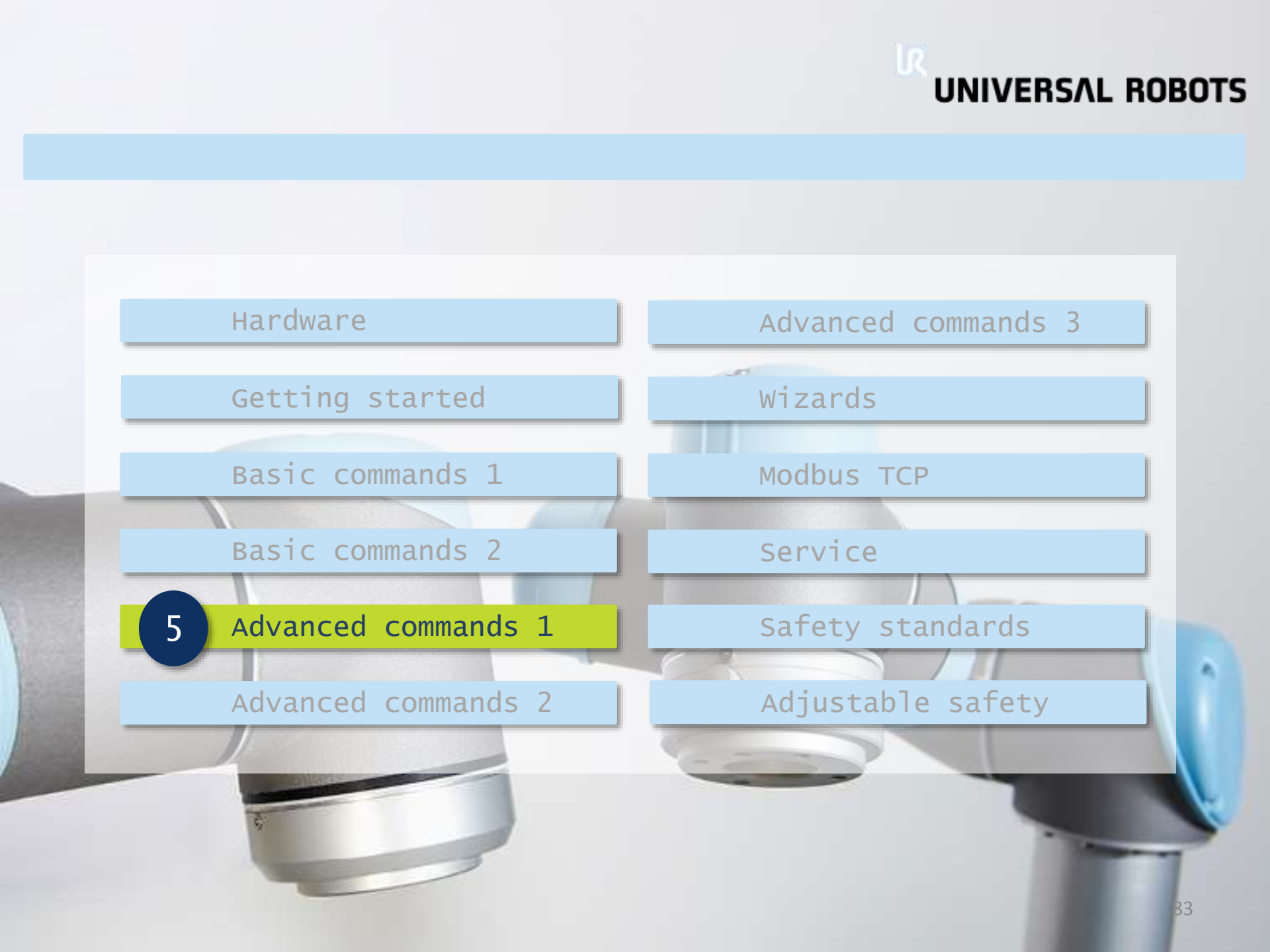
- Create a simple signal handling program
- Setup:
  - Rename digital_out[4] to ”Lamp” and digital_in[5] to ”Button”
  - Connect the provided Lamp to Digital Output 4 in the control box as shown below.
  - Connect the provided button to Digital Input 5 in the control box as shown below.



UNIVERSAL ROBOTS

## Lab exercise part 2

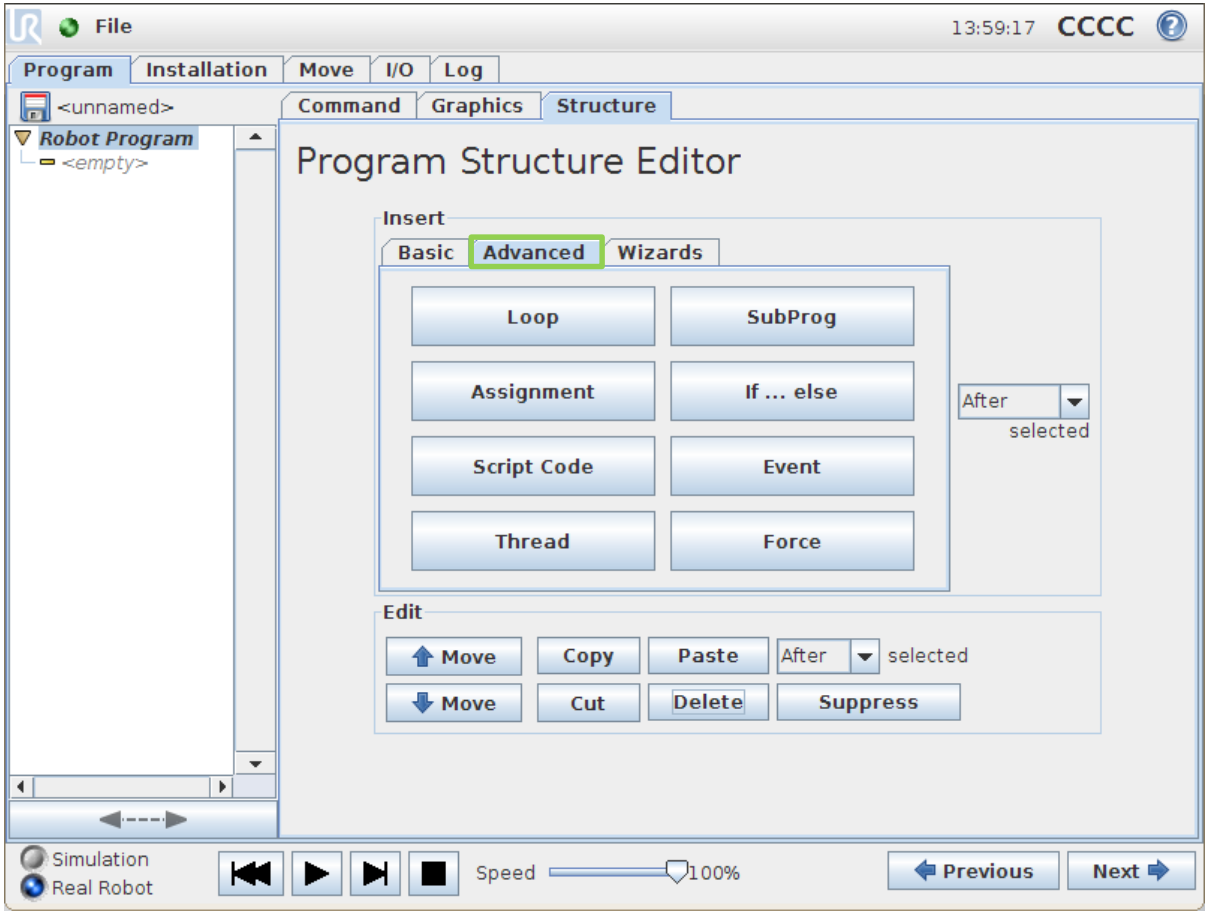
- Program – Button_Folder
  - Create a folder called Button_Folder
  - Wait for the button to be pushed before continuing
  - Add a comment at the start of the folder explaining what this code does.
- Program – Lamp_Folder
  - Create a folder called Lamp_Folder
  - Turn on the lamp, wait for 5 seconds 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.



UNIVERSAL ROBOTS

# Advanced commands tab

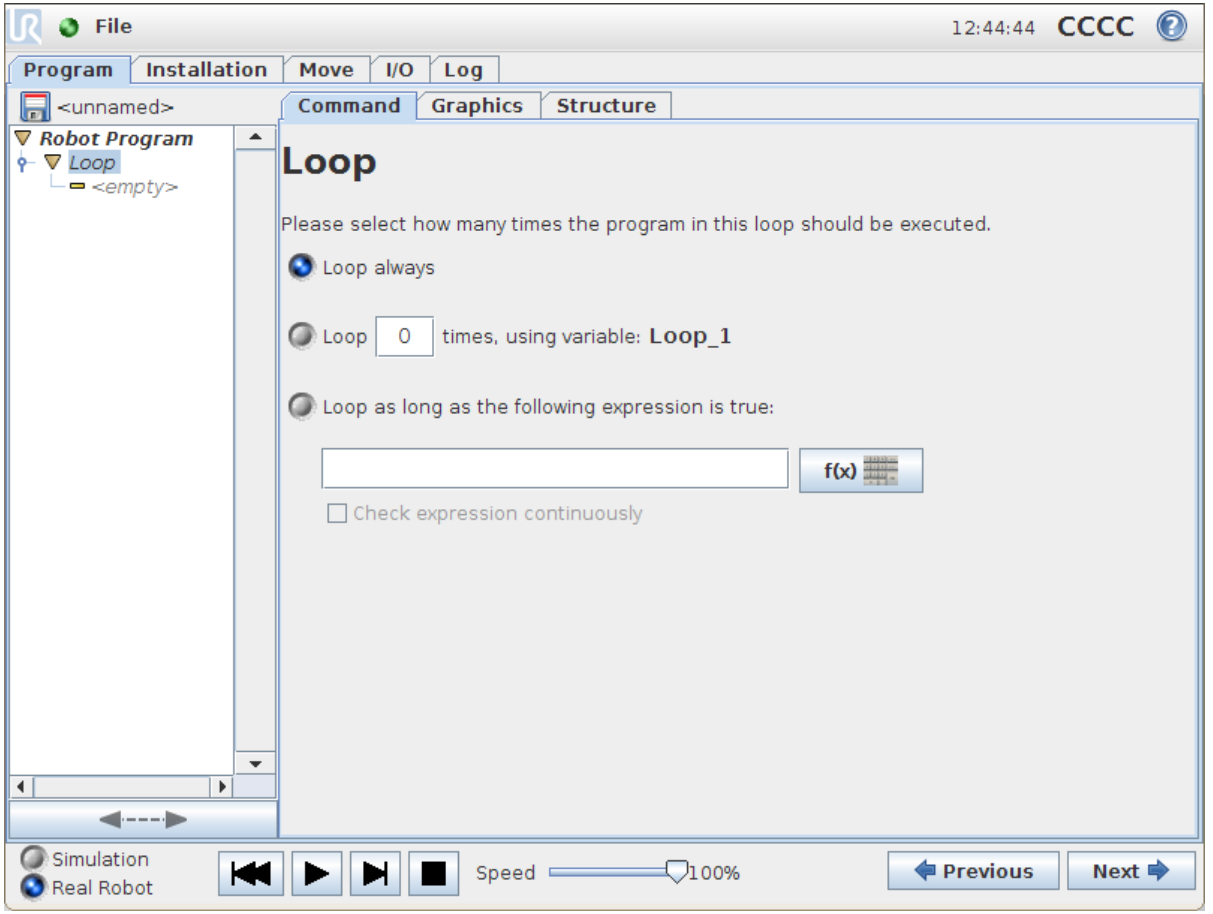
- Advanced commands
  - Perform advanced operations



UNIVERSAL ROBOTS

# Loop command

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



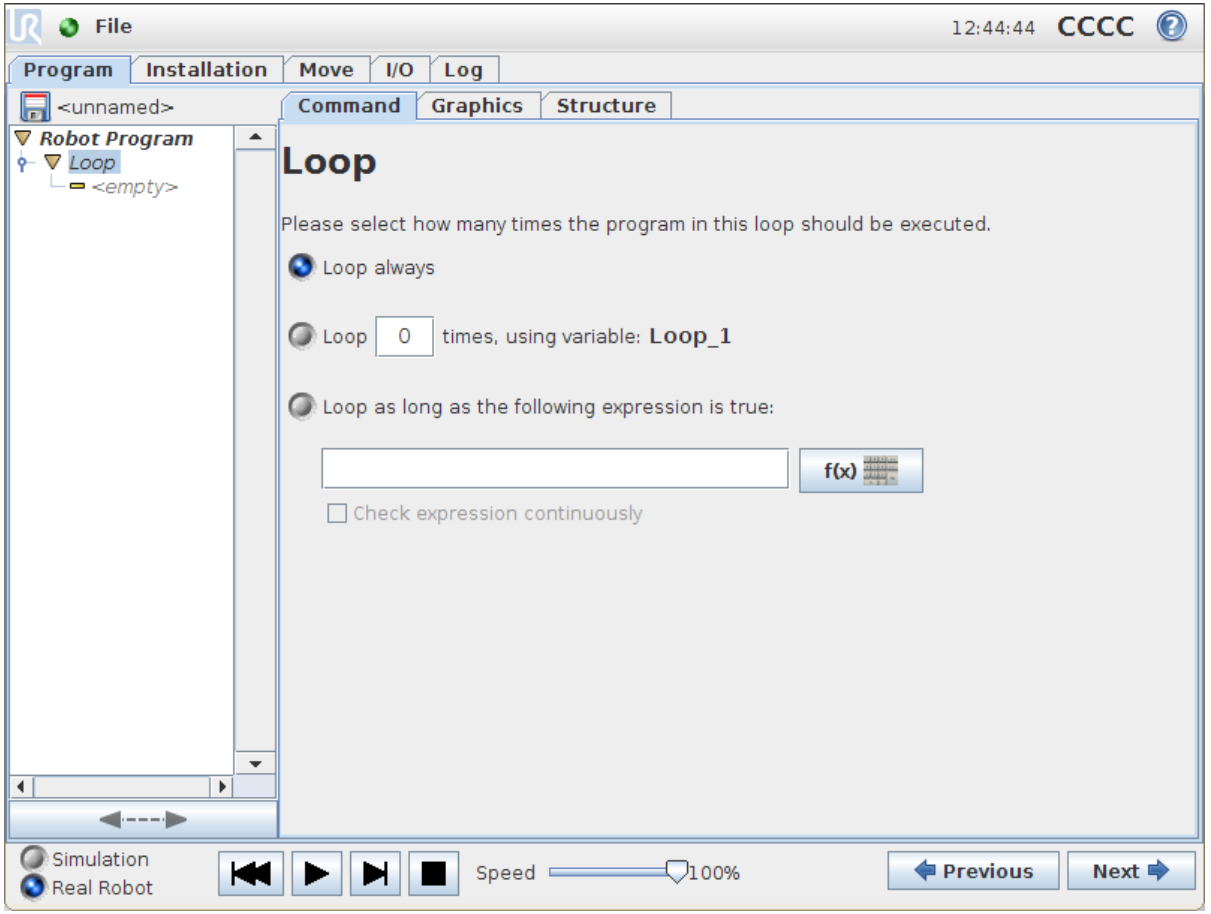

```
Robot Program
    MoveJ
        Waypoint_5
    Loop 3 times
        MoveL
        Waypoint_1
        Waypoint_2
        Waypoint_3
        Waypoint_4
    MoveJ
        Waypoint_5
```

- Save sample program as loop.urp

UNIVERSAL ROBOTS

# Loop command

- Interrupt
  - Check expression continuously



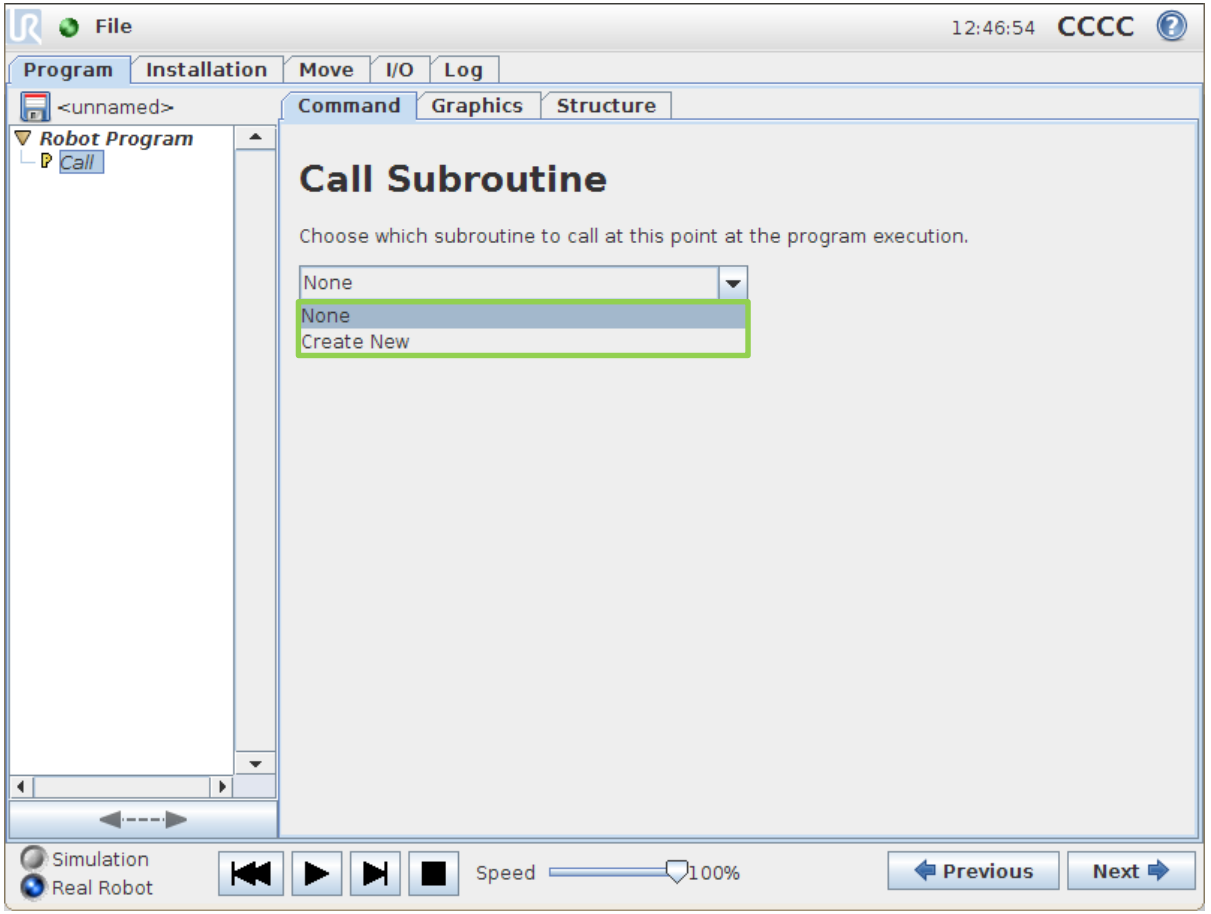

```
Robot Program
    MoveJ
        Waypoint_5
    While DI[0] = True
        MoveL
        Waypoint_1
        Waypoint_2
        Waypoint_3
        Waypoint_4
    MoveJ
        Waypoint_5
```

- Save sample program as loop_interrupt.urp

UNIVERSAL ROBOTS

## SubProg command

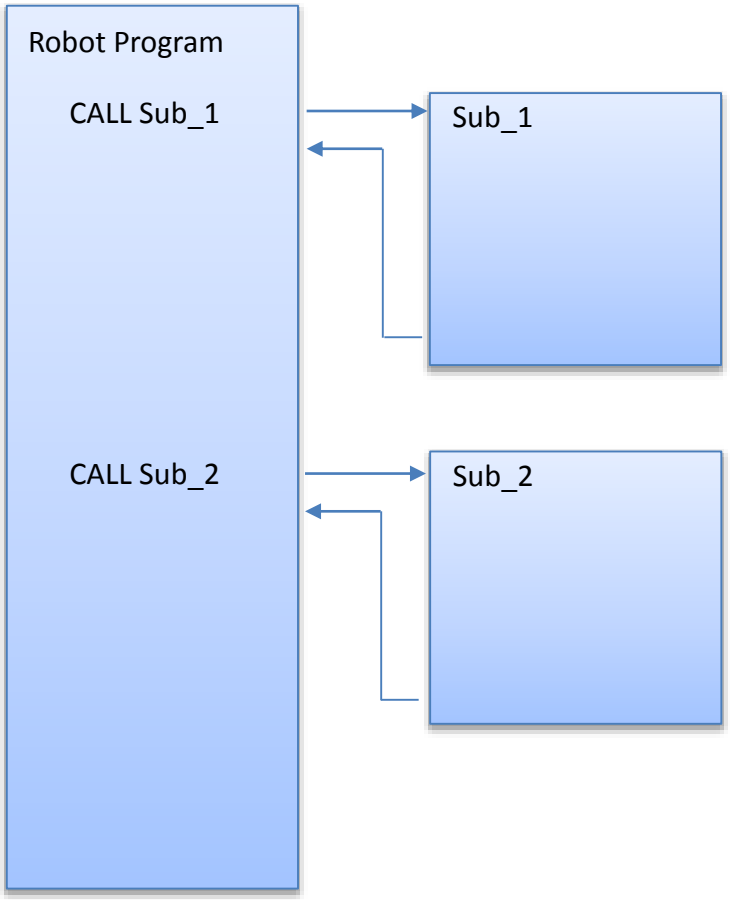
- Subprograms
  - Organize program
  - Split program into subprograms
  - Re-use subprogram in multiple programs



UNIVERSAL ROBOTS

## SubProg command

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

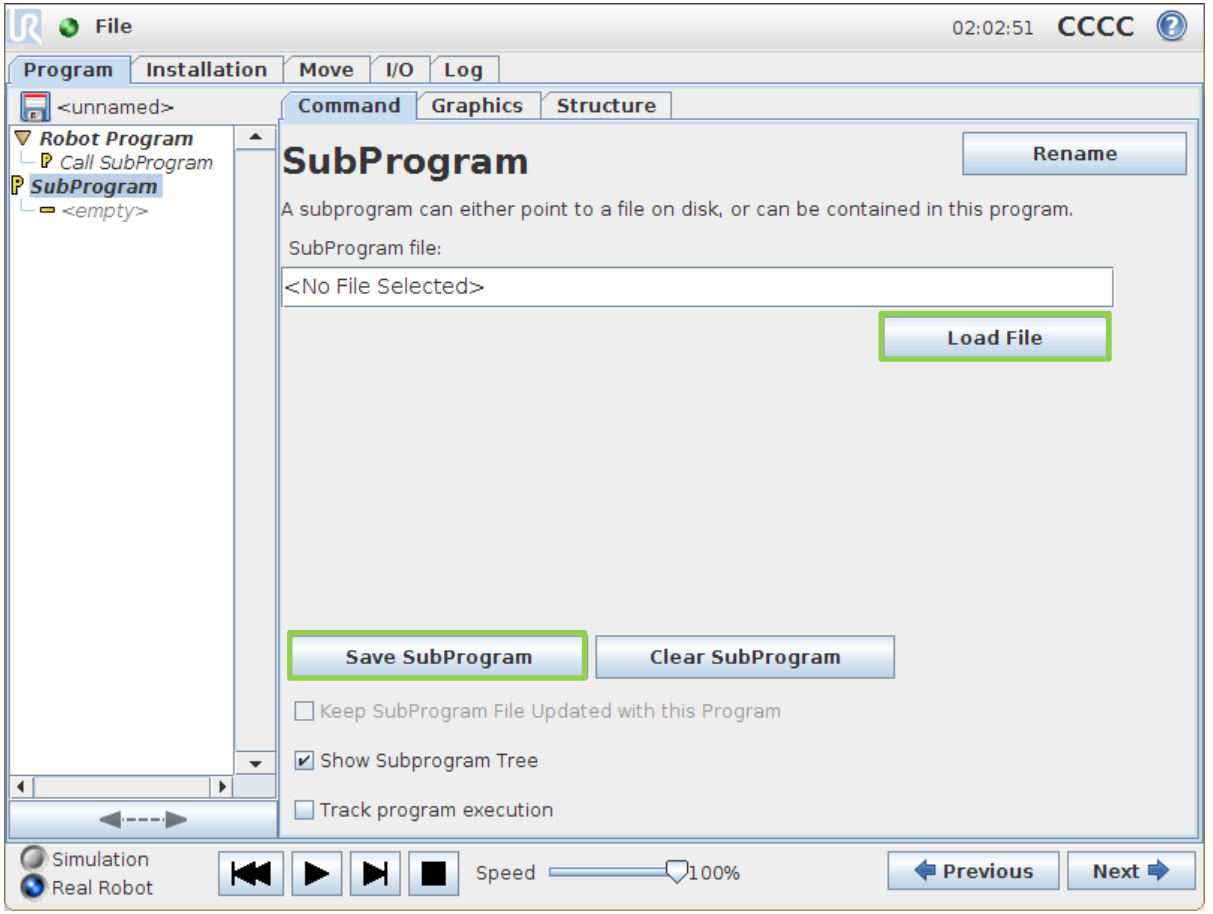



- Note: Calling subprogram from subprogram is not supported

UNIVERSAL ROBOTS

# 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
```

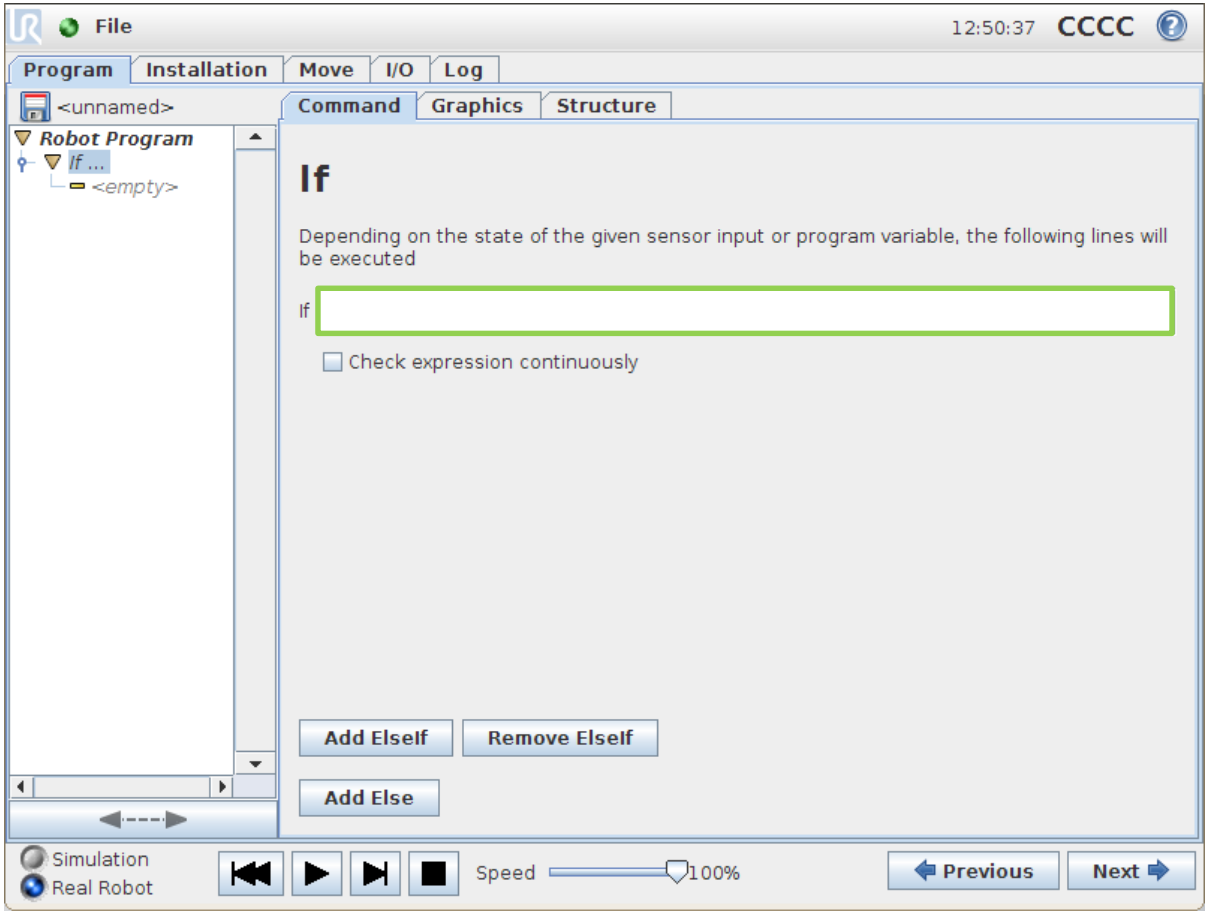
- Save sample program as call_sub.urp

UNIVERSAL ROBOTS

# 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
```



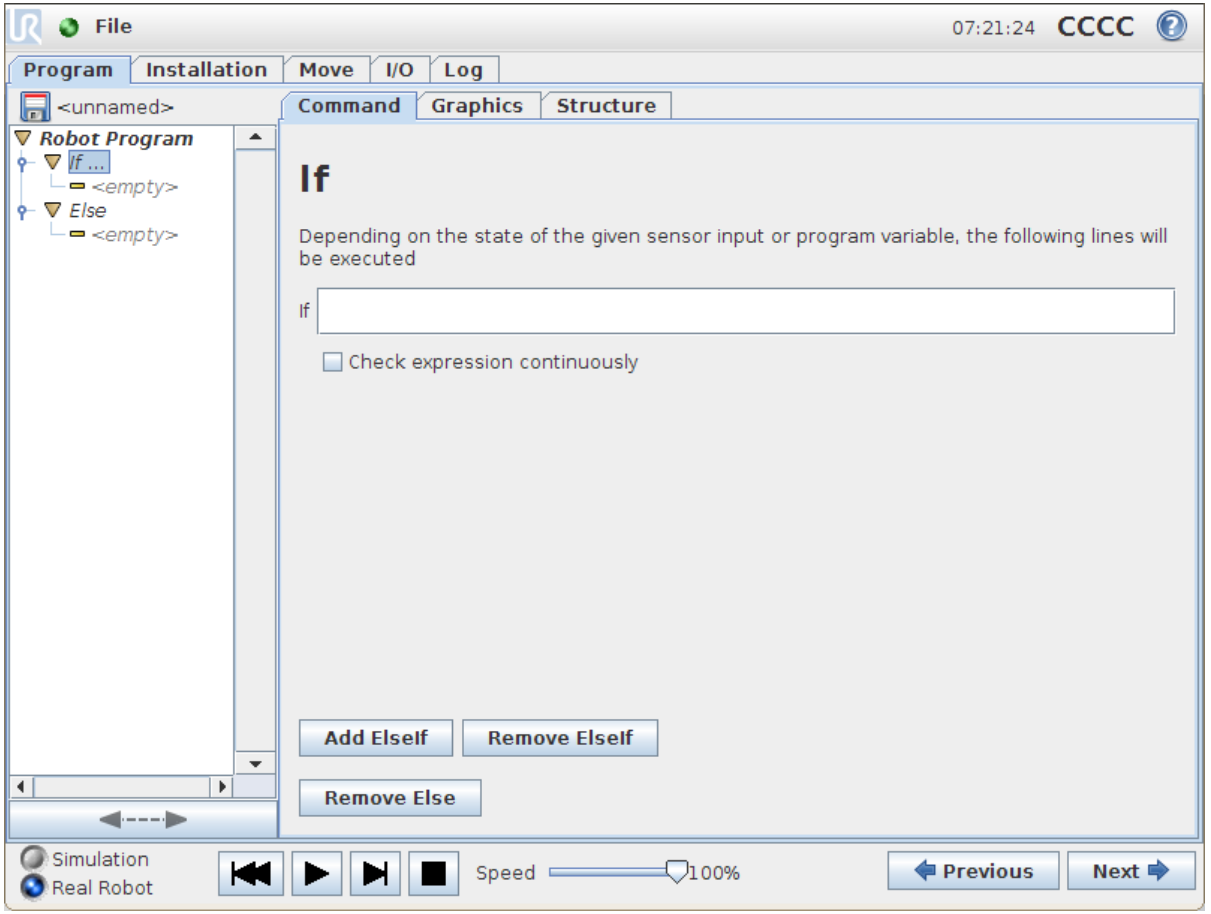

- Save sample program as if.urp

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

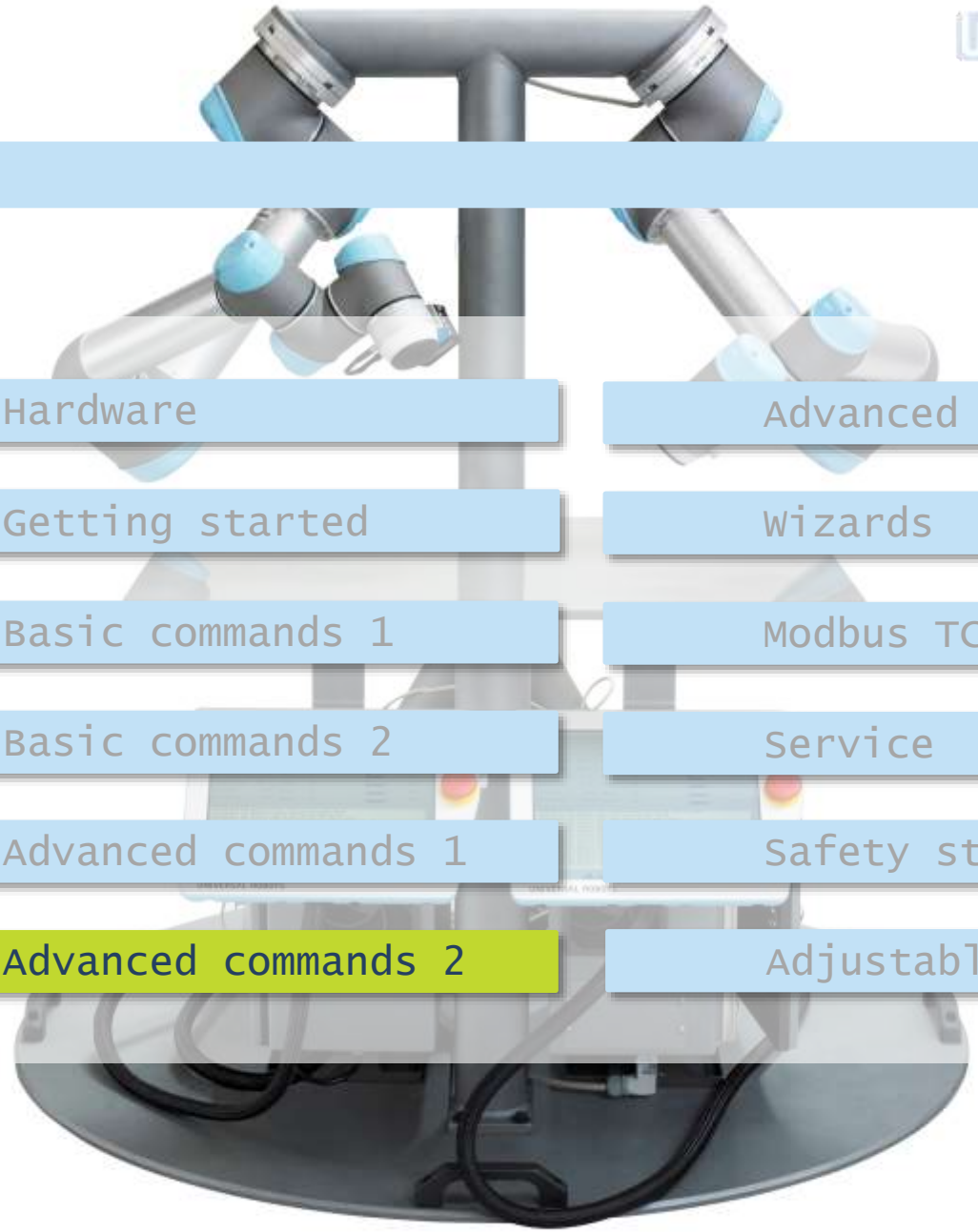
## Lab exercise part 1

- 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

UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

6 Advanced commands 2

Advanced commands 3

Wizards

Modbus TCP

Service

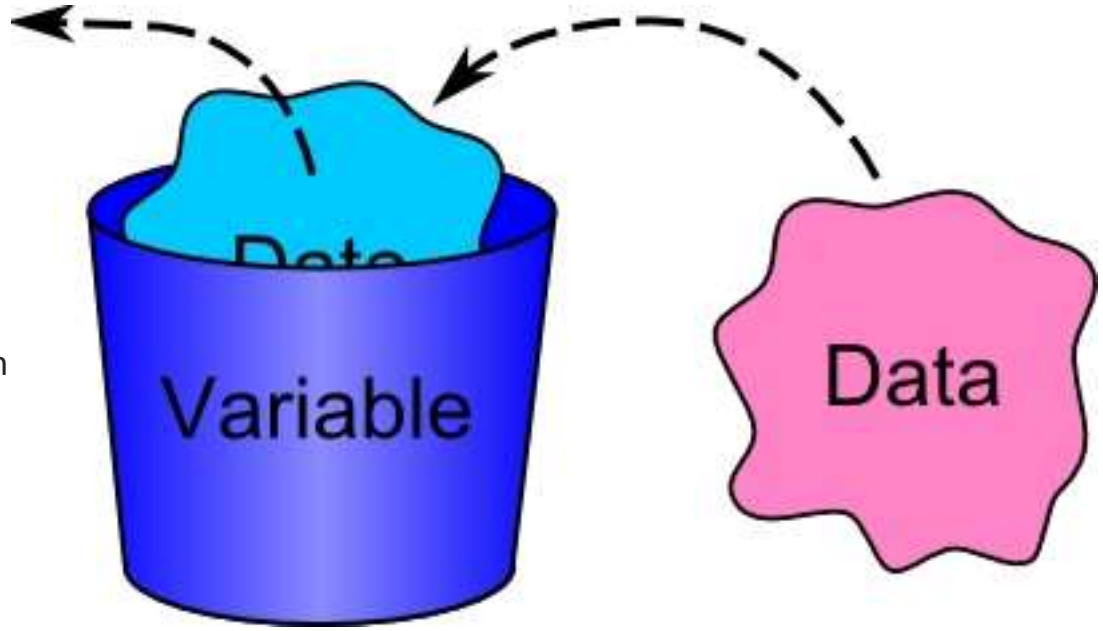
Safety standards

Adjustable safety

UNIVERSAL ROBOTS

# 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



UNIVERSAL ROBOTS

## 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 |

UNIVERSAL ROBOTS

# Variable scope

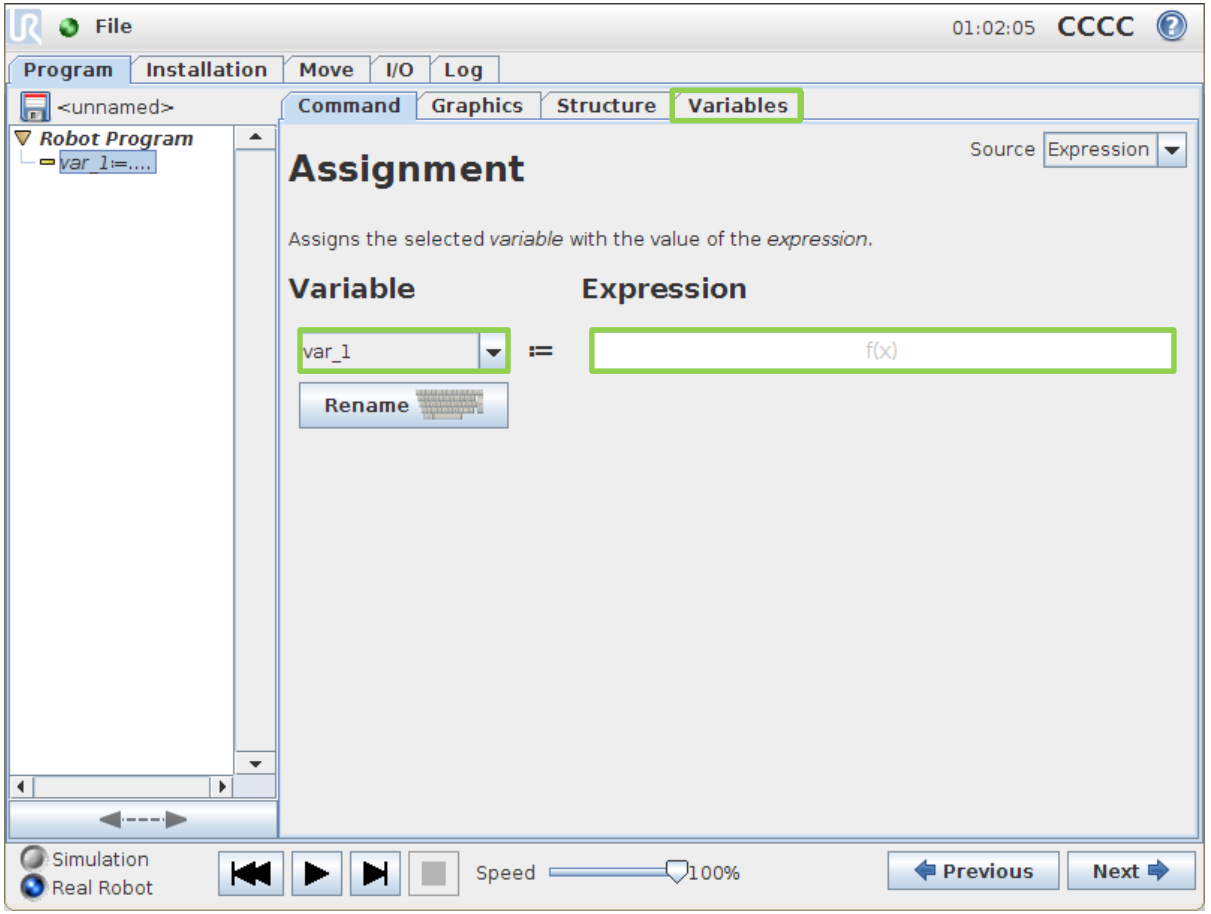
| scope | location |
|---|---|
| local | program |
| global | installation |

- 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

UNIVERSAL ROBOTS

# Assignment command

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



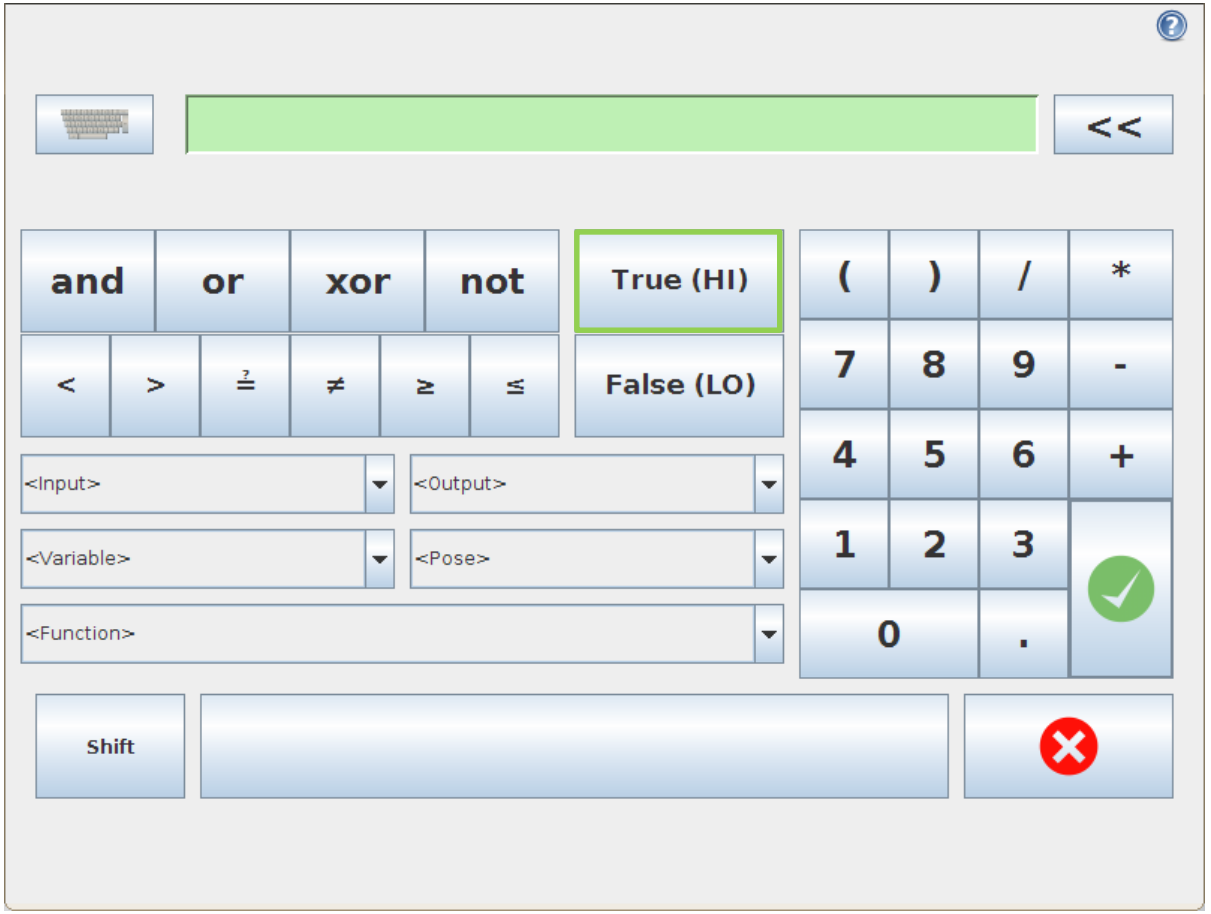

```
Robot Program
    var_1 = True
    Wait 0.5
    var_1 = False
    Wait 0.5
```

- Save sample program as var_bool.urp

UNIVERSAL ROBOTS

## Expression editor

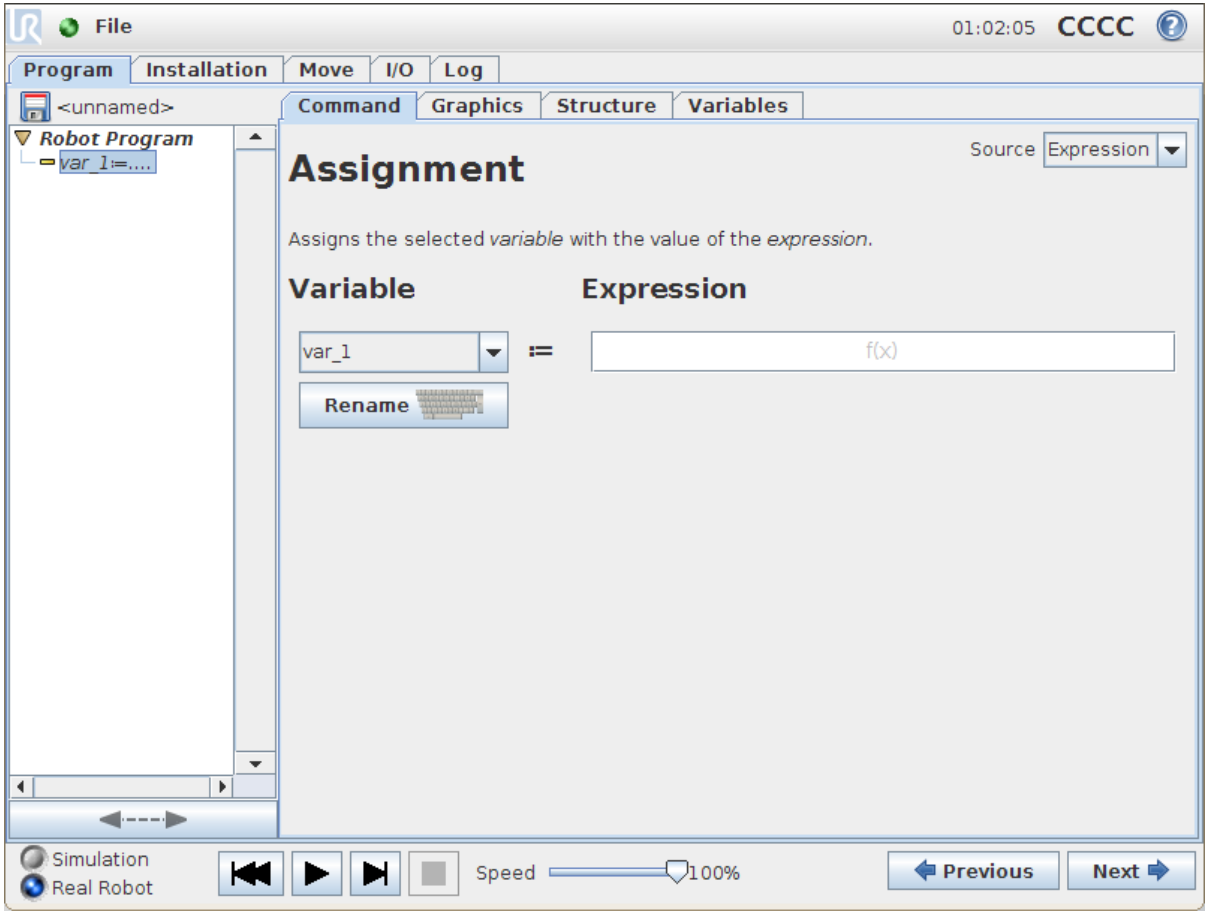
- Options
  - Numerical values
  - Inputs
  - Outputs
  - Variables
  - Poses
  - Script codes
  - Logic operators
  - Keyboard



UNIVERSAL ROBOTS

# Use variable as counter

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



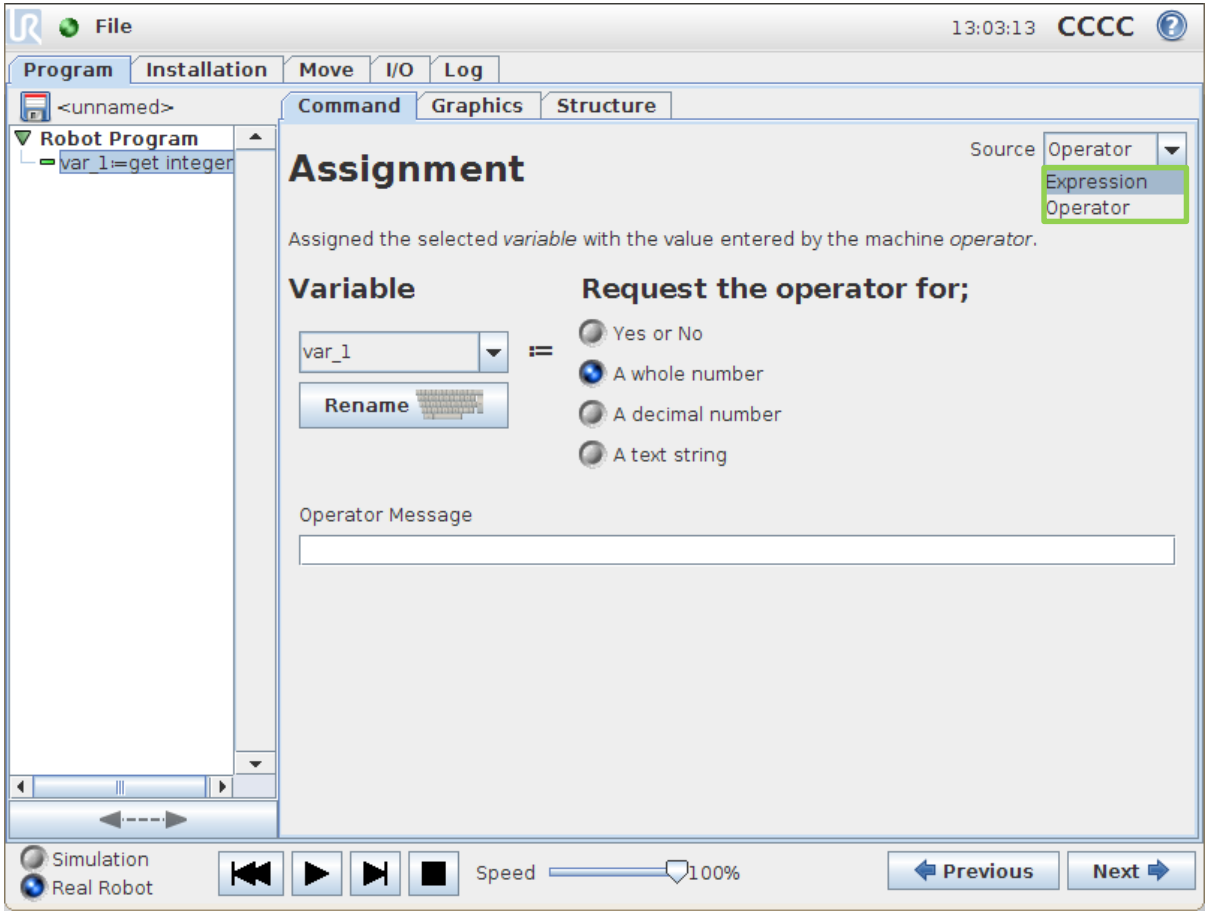

```
Robot Program
    var_1 = 0
    While var_1 < 5
        Pick_part
        Place_part
        var_1 = var_1 + 1
```

- Save sample program as var_counter.urp

UNIVERSAL ROBOTS

# Operator input

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



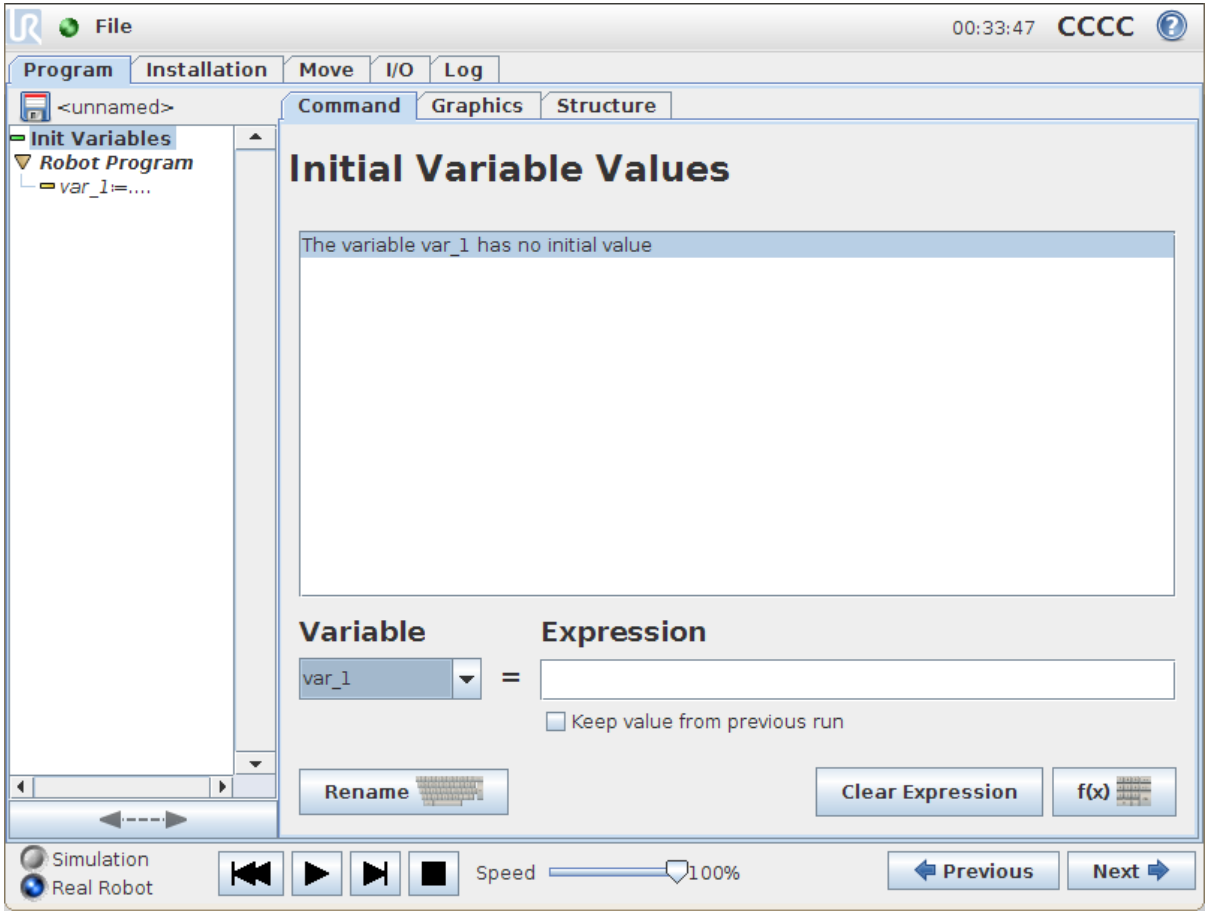

Robot Program
var_1 = 0
var_2 = *operator input*
While var_1 < var_2
*Pick_part*
*Place_part*
var_1 = var_1 + 1

- Save sample program as var_operator_input.urp

UNIVERSAL ROBOTS

# Init variables

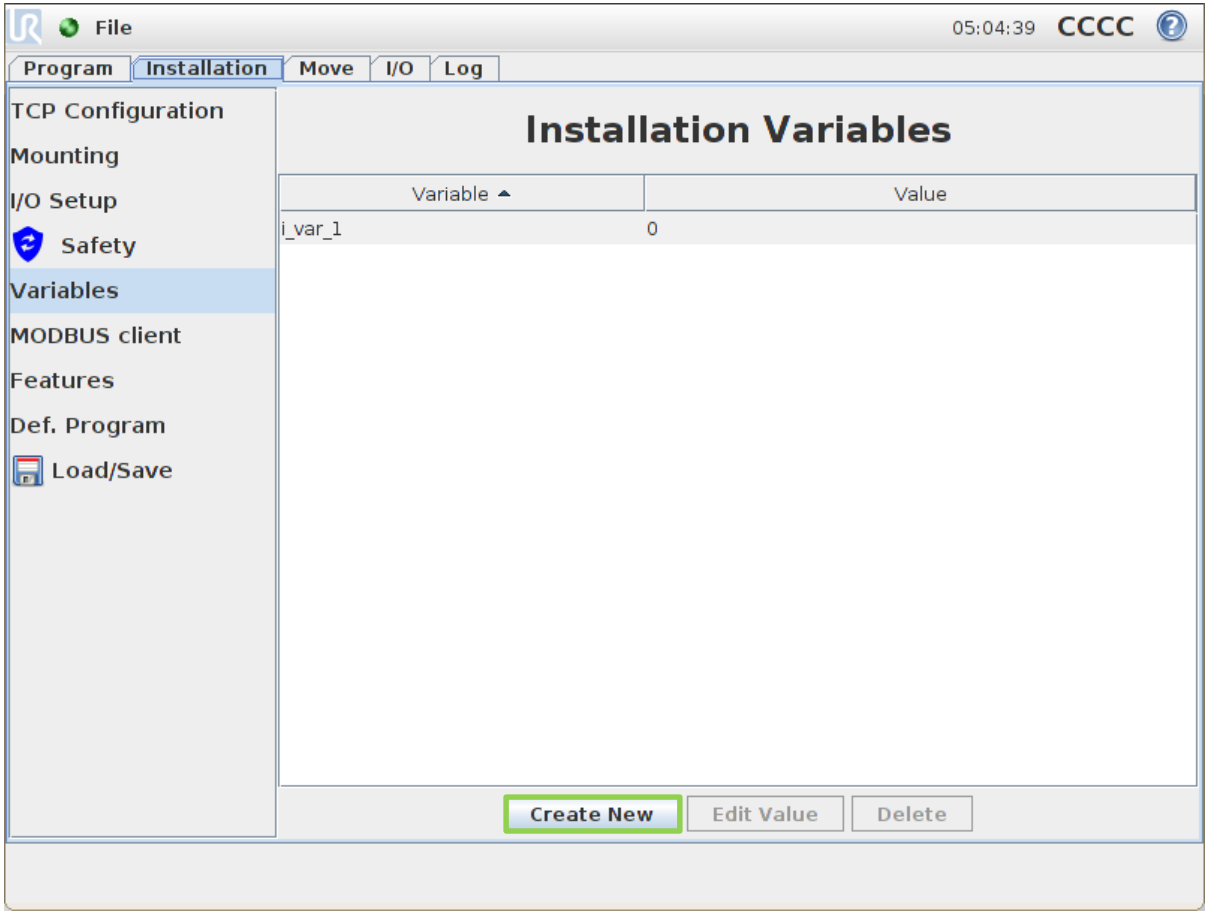
- Inititialize variables
  - Local variables listed
  - Preset to fixed value



UNIVERSAL ROBOTS

# Installation variables

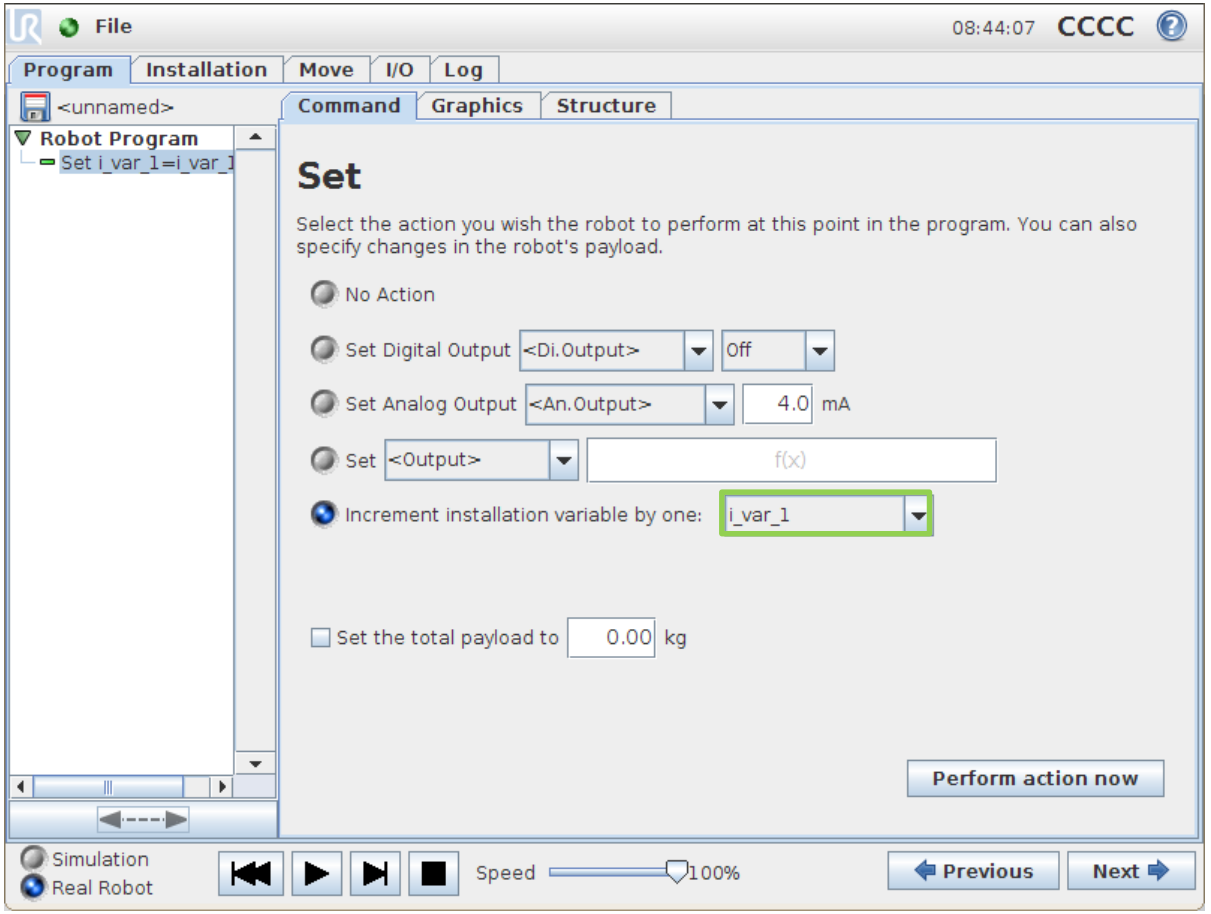
- Features
  - Listed in Installation tab
  - Stored in separate file
  - Keeps value after reboot
- Functionality
  - Global variable
  - Accessible from all programs
  - Same functionality as local variable



UNIVERSAL ROBOTS

# Use installation variable as counter

- Set command
  - Increment installation variable by one



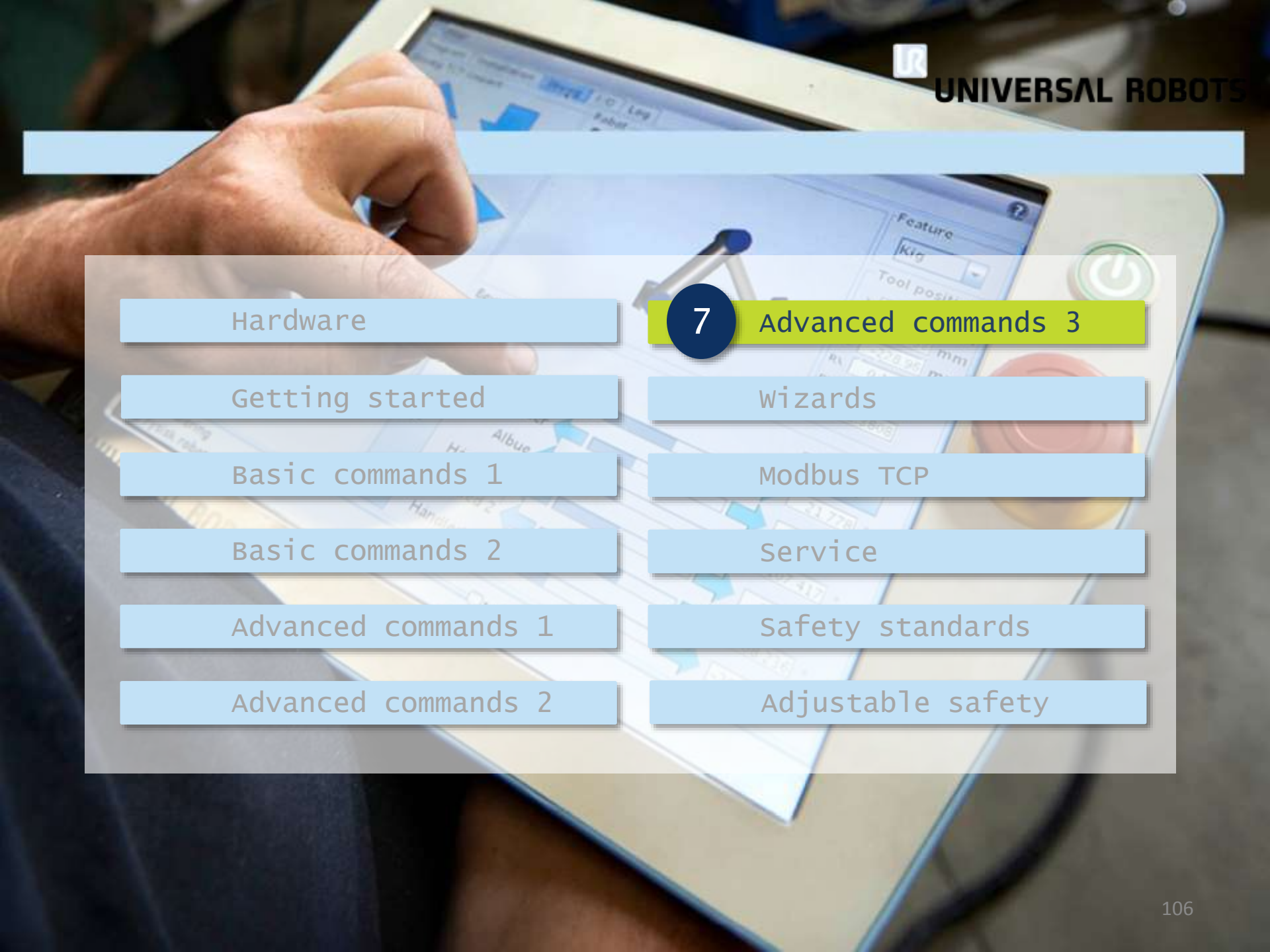

```
Robot Program
    i_var_1 = 0
    var_2 = operator input
    While i_var_1 < var_2
        Pick_part
        Place_part
        Set i_var_1 = i_var_1 + 1
```

- Save sample program as inst_var_operator_input.urp

UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

Hardware

7 Advanced commands 3

Getting started

Wizards

Basic commands 1

Modbus TCP

Basic commands 2

Service

Advanced commands 1

Safety standards

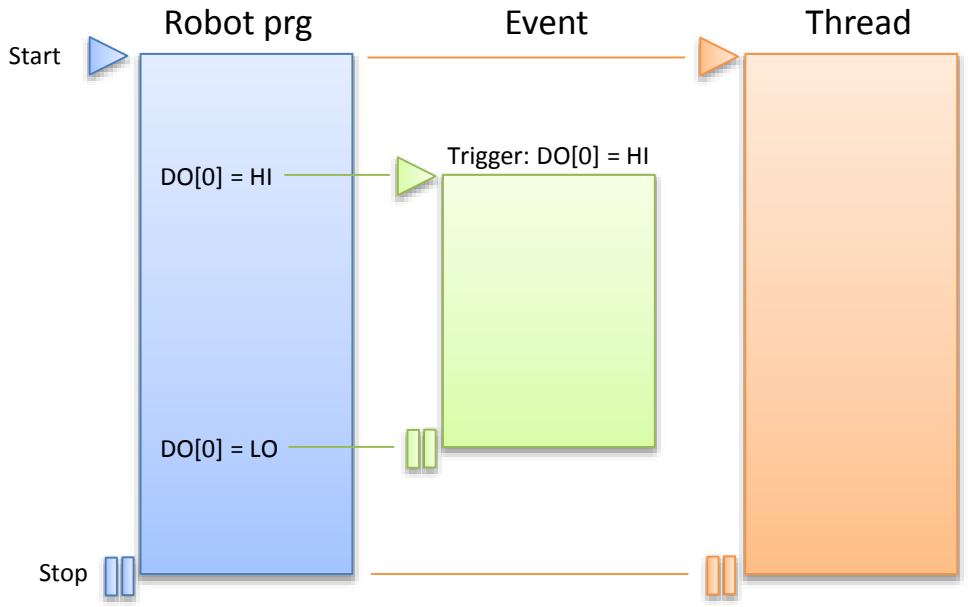
Advanced commands 2

Adjustable safety

UNIVERSAL ROBOTS

## Threads/events

- Thread
  - Parallel process
  - Continuously running
- Event
  - Parallel process
  - Triggered by a condition

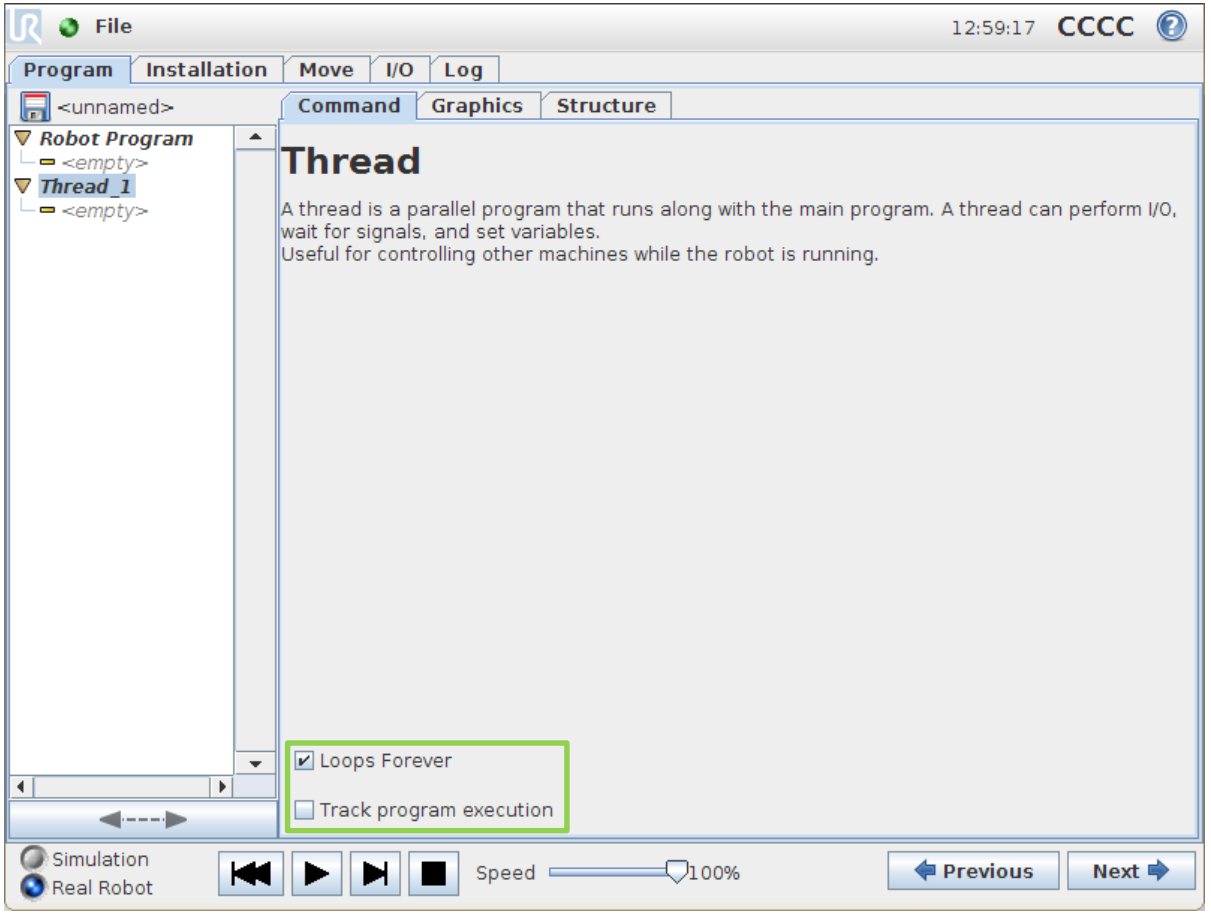



- Purpose
  - Useful for controlling I/O communication with other machines, do complex calculations etc.

UNIVERSAL ROBOTS

# Thread

- Settings
  - Loops forever
  - Track program execution




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

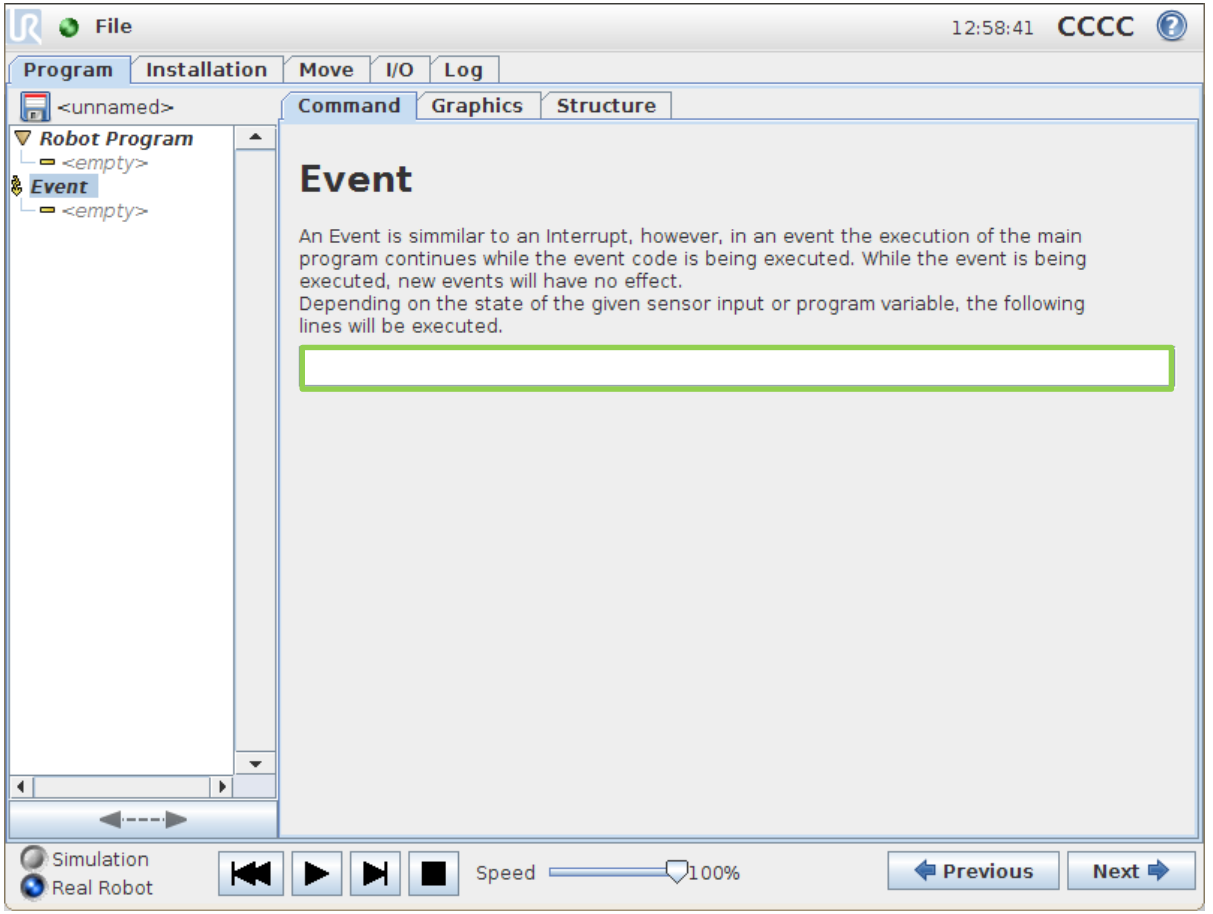
```
Robot Program
    MoveJ
        Waypoint_1
        Waypoint_2
Thread_1
    IF DI[0] = True
        HALT
    WAIT 0.01
```

- Save sample program as thread.urp

UNIVERSAL ROBOTS

## Event

- Trigger
  - Define condition to run Event



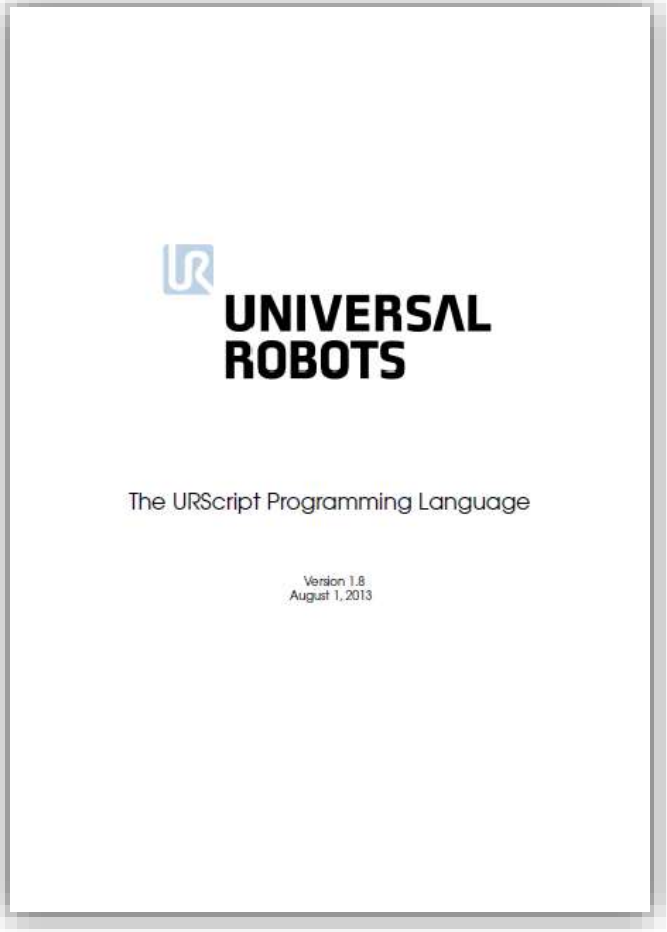

```
Robot Program
    MoveJ
        Waypoint_1
        Waypoint_2
Event DI[0] = True
    HALT
```

- Save sample program as event.urp

UNIVERSAL ROBOTS

# 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

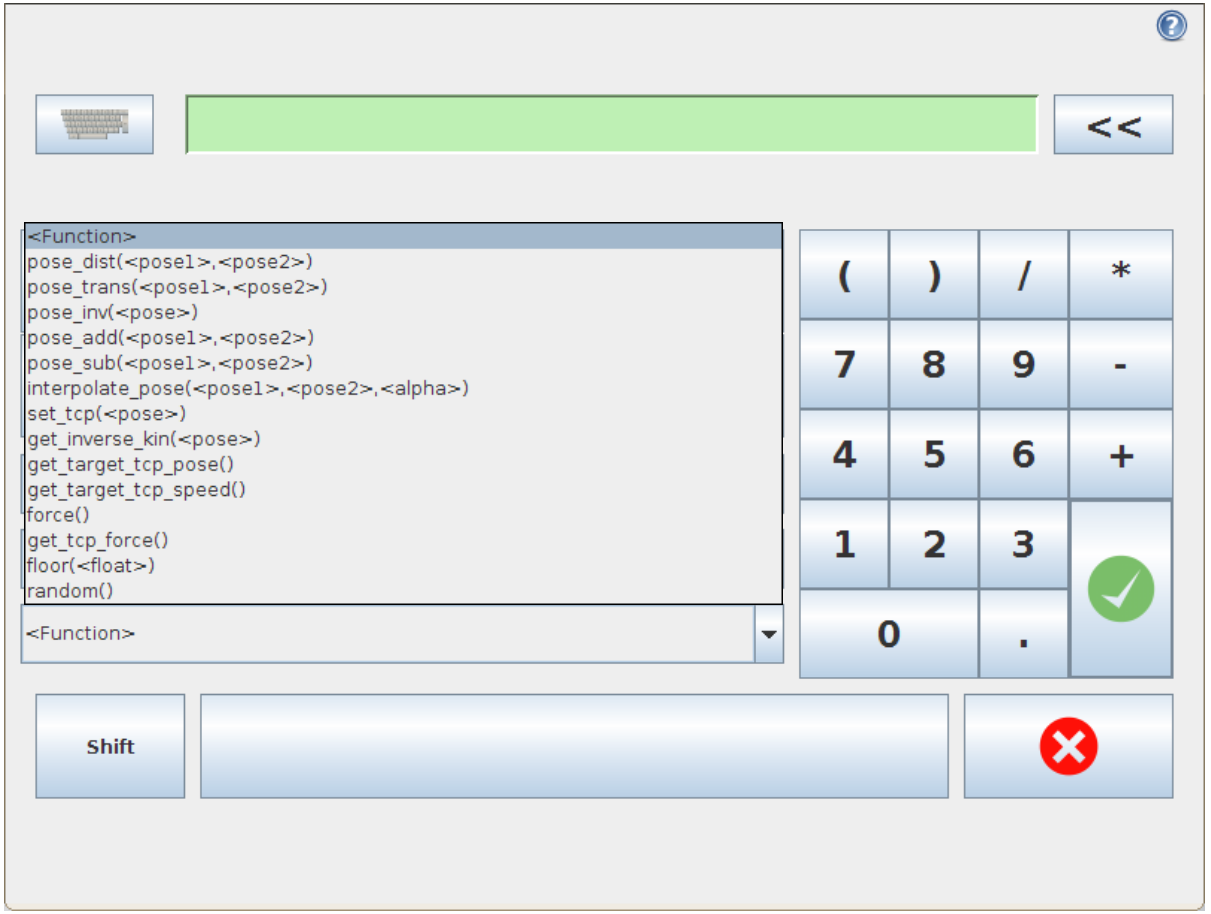


UNIVERSAL ROBOTS

# 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
```



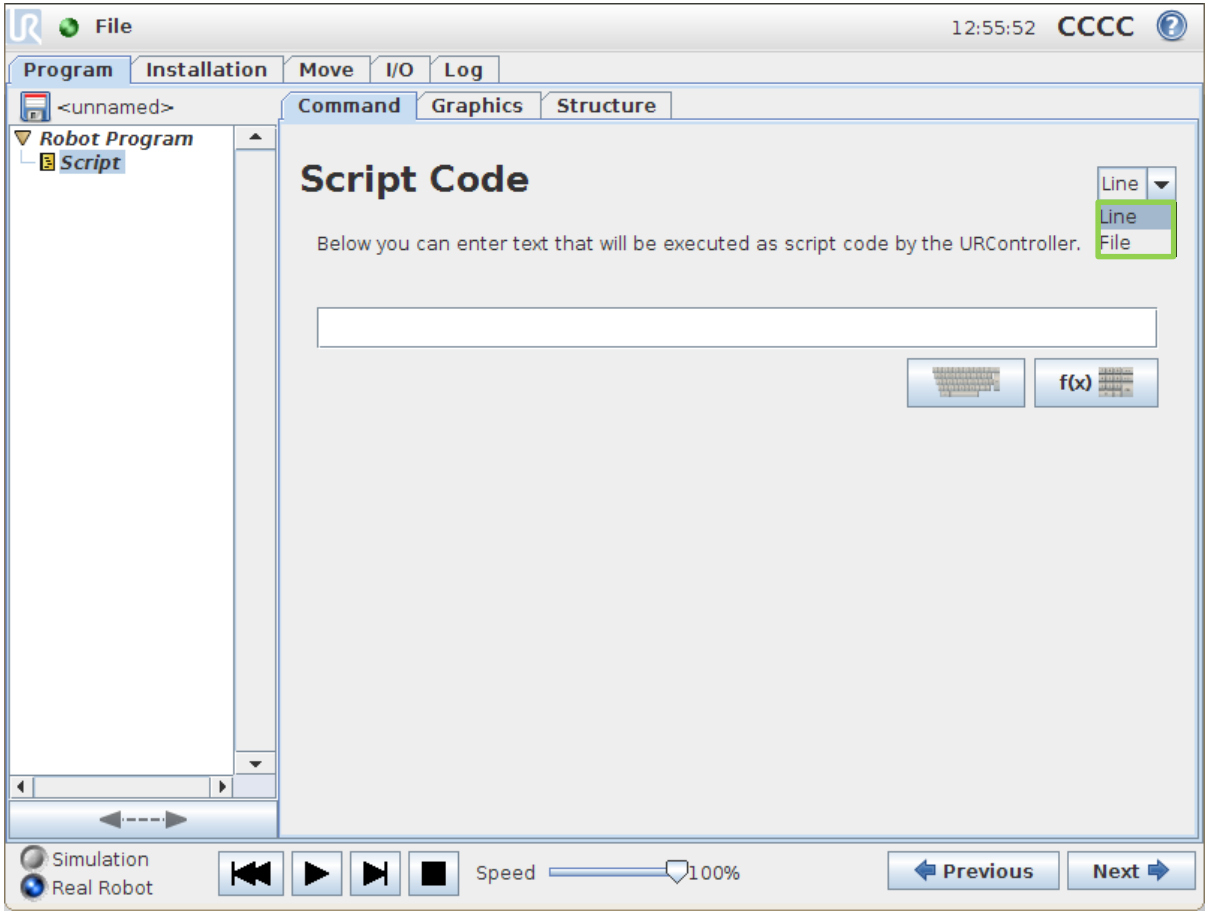

- Save sample program as force_feedback.urp

UNIVERSAL ROBOTS

# Script command

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

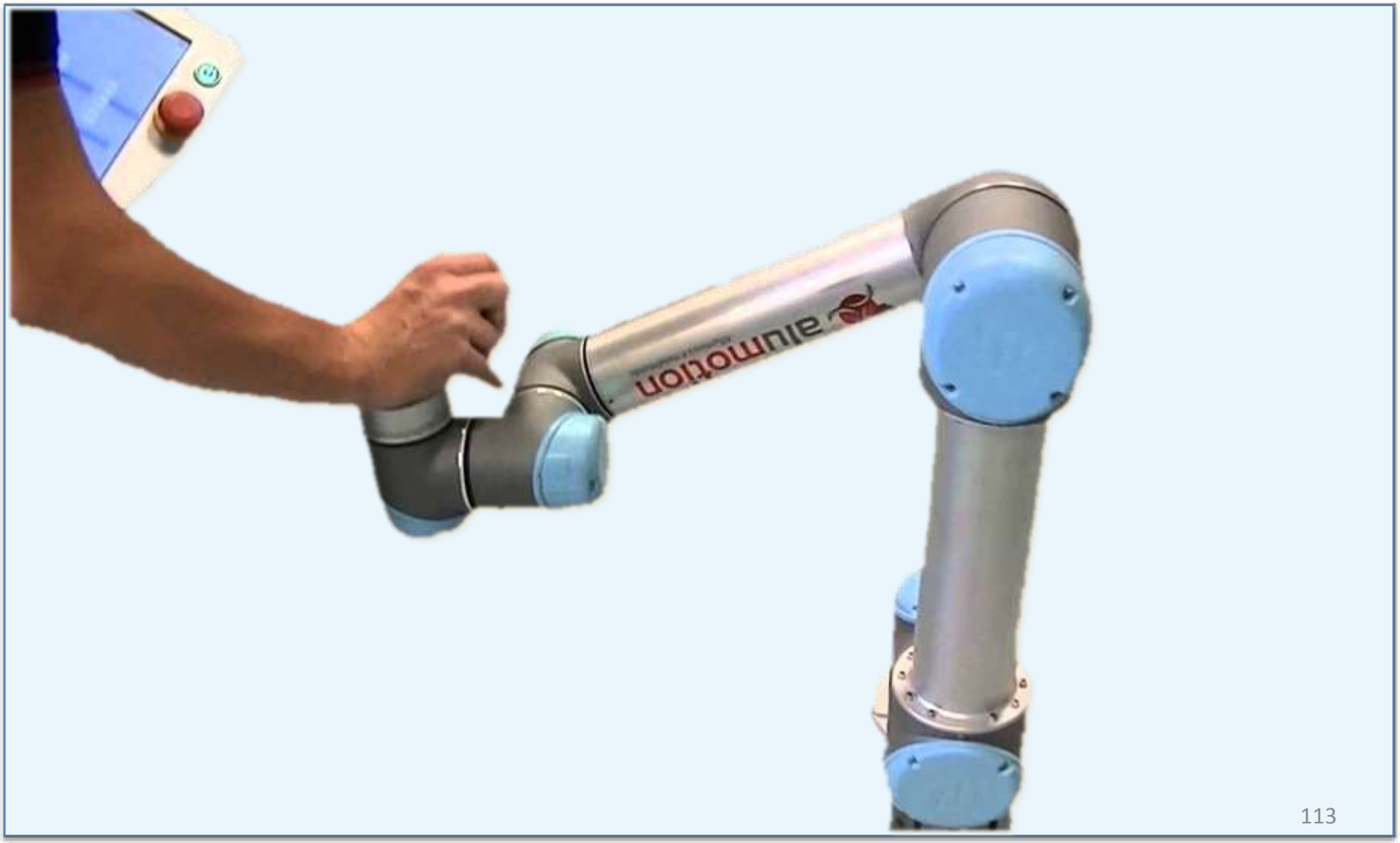
```
Robot Program
    set_digital_out(0,True)
    WAIT 0.5
    set_digital_out(0,False)
    WAIT 0.5
```




- Save sample program as script_line.urp

UNIVERSAL ROBOTS

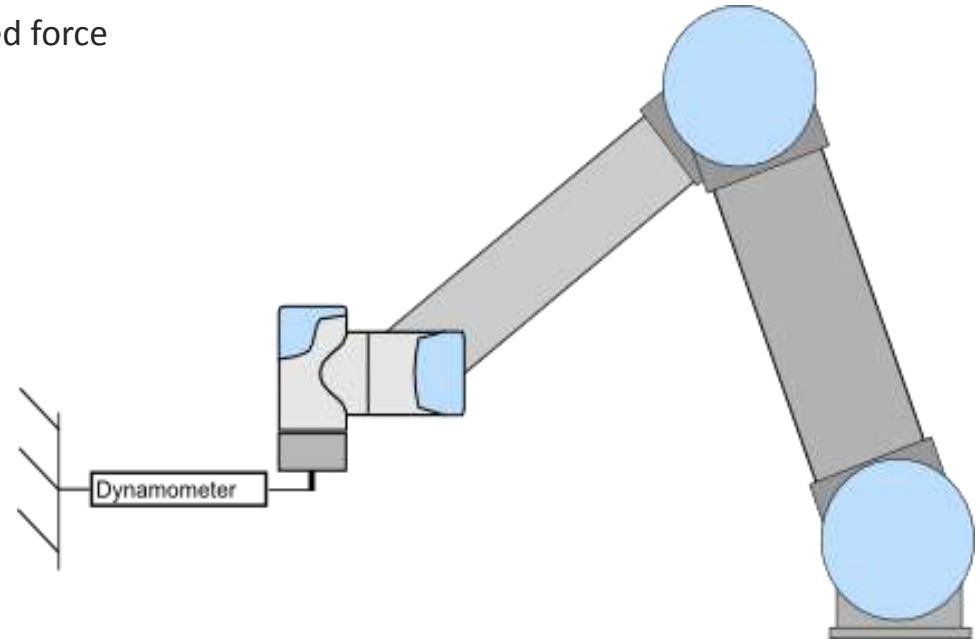
## Force Control



UNIVERSAL ROBOTS

## Force command

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

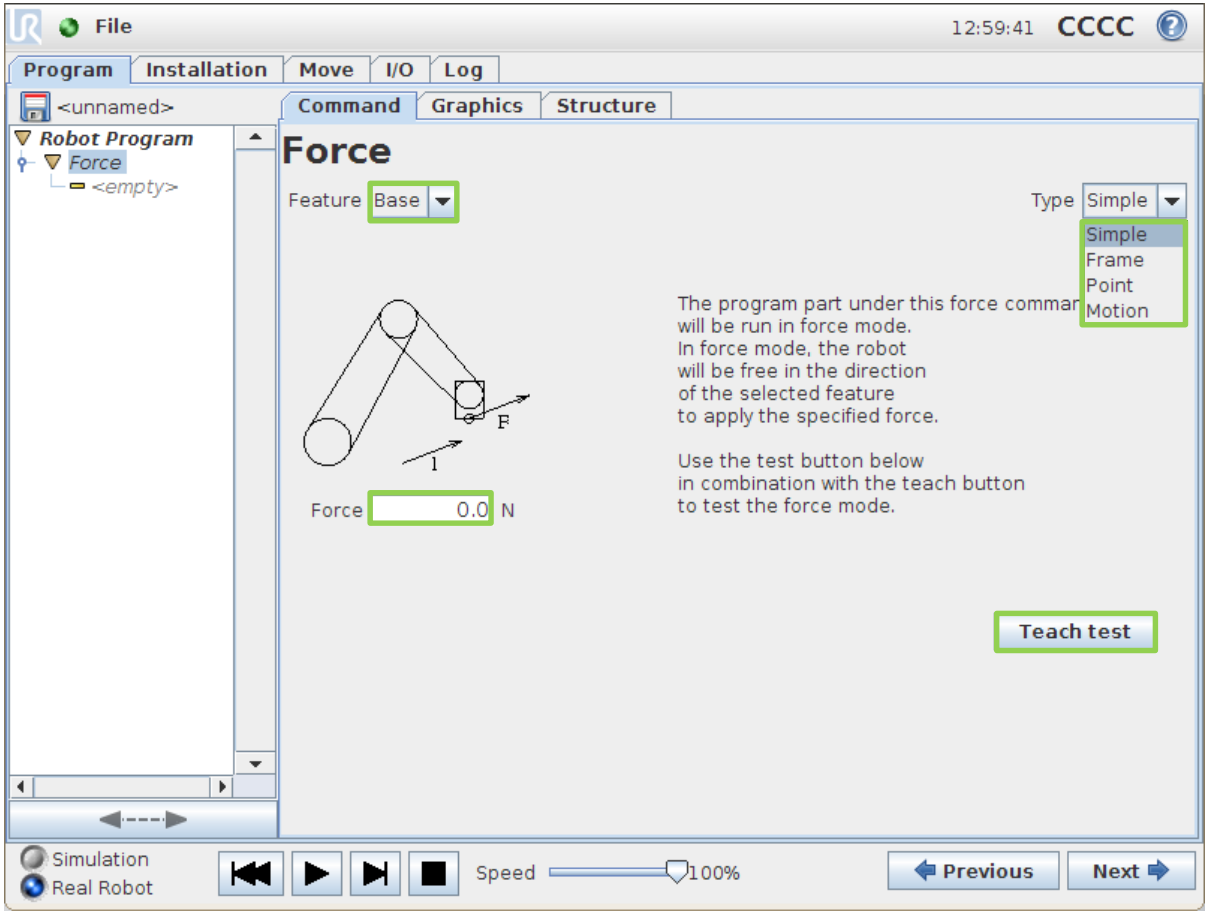



- Specifications
  - Force precision ± 10 N
  - Torque precision ± 5 Nm
  - Position precision ± 5 mm
  - Orientation precision ± 0.5 °

UNIVERSAL ROBOTS

# How to use force

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

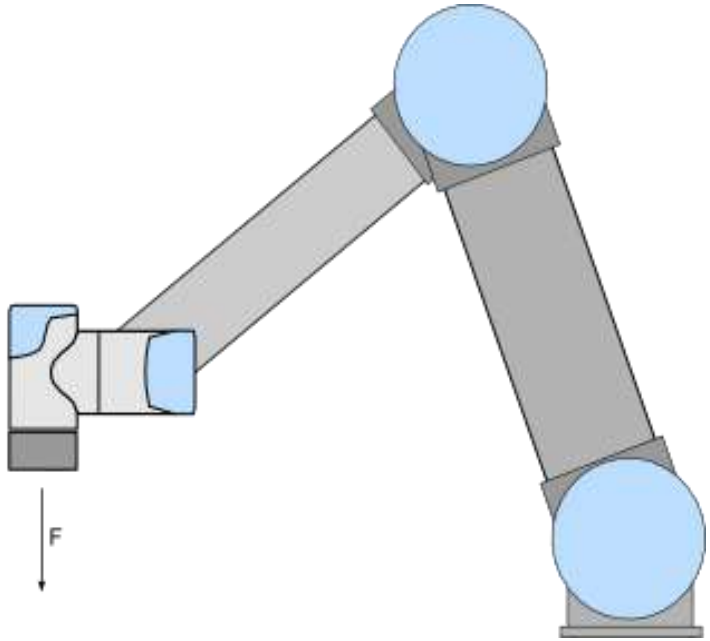


UNIVERSAL ROBOTS

## Force type: Simple

- Features
  - One axis on compliant mode
  - Force direction in Z-axis

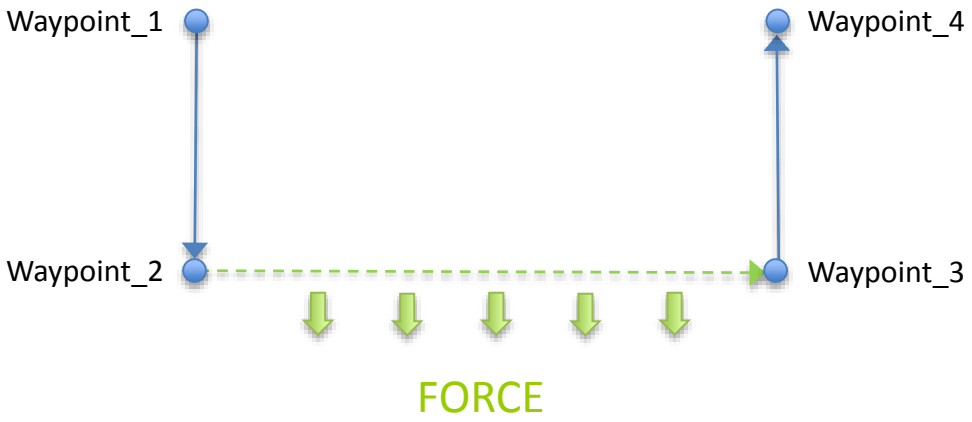
- Test
  - Add Force to new program
  - Set Type: Simple
  - Set Feature: BASE
  - Set Force: 30 N
    - Teach test
  - Set Force: -30 N
    - Teach test
  - Set Feature: TOOL
    - Teach test



UNIVERSAL ROBOTS

# Force type: Simple

- Sample program



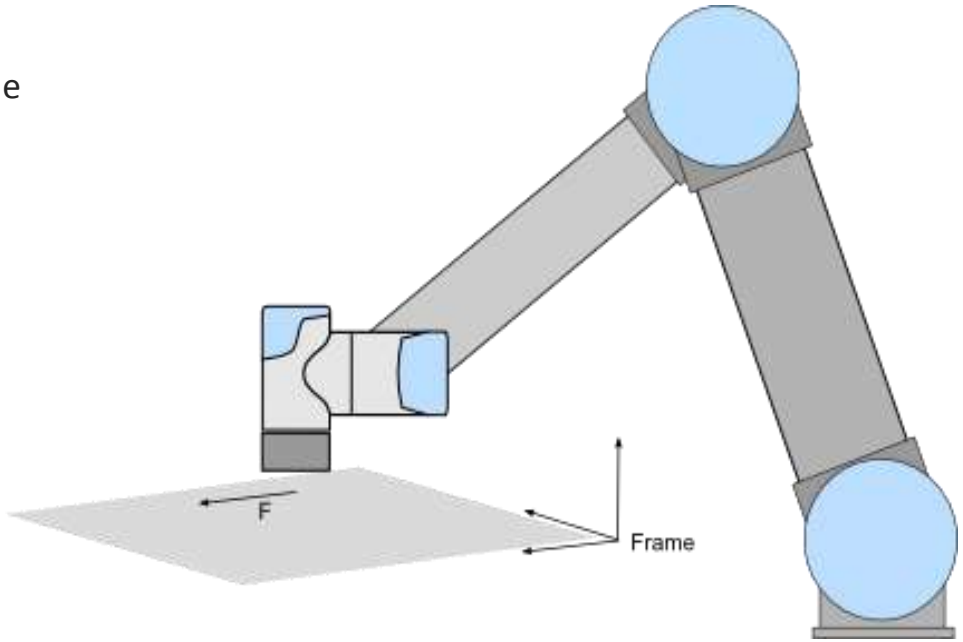

```
Robot Program
    MoveL
        Waypoint_1
        Waypoint_2
        Force
            Waypoint_2
            Waypoint_3
         Waypoint__4
```

- Save sample program as force_simple.urp

UNIVERSAL ROBOTS

## Force type: Frame

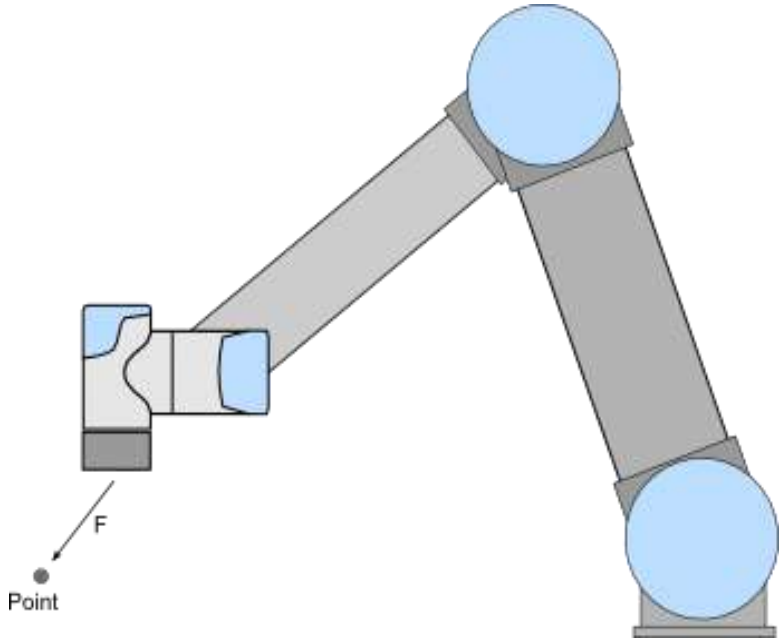
- Features
  - Multiple axes in compliant mode
  - Force level individual for each axis
  - Define speed for axes in compliant mode
  - Base, Tool, user defined frames



UNIVERSAL ROBOTS

## Force type: Point

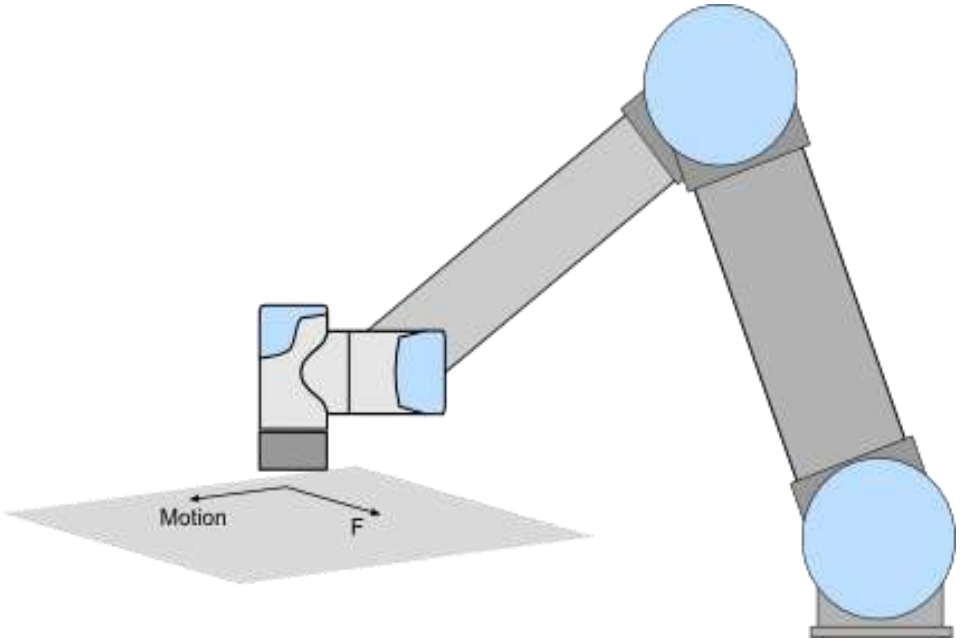
- Features
  - Specify Feature point
  - Y-axis of task frame points towards Feature point
  - Task frame changes during runtime



UNIVERSAL ROBOTS

## Force type: Motion

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




  - X-a*xis not compliant*
  - *Teach mode not applicable*

UNIVERSAL ROBOTS

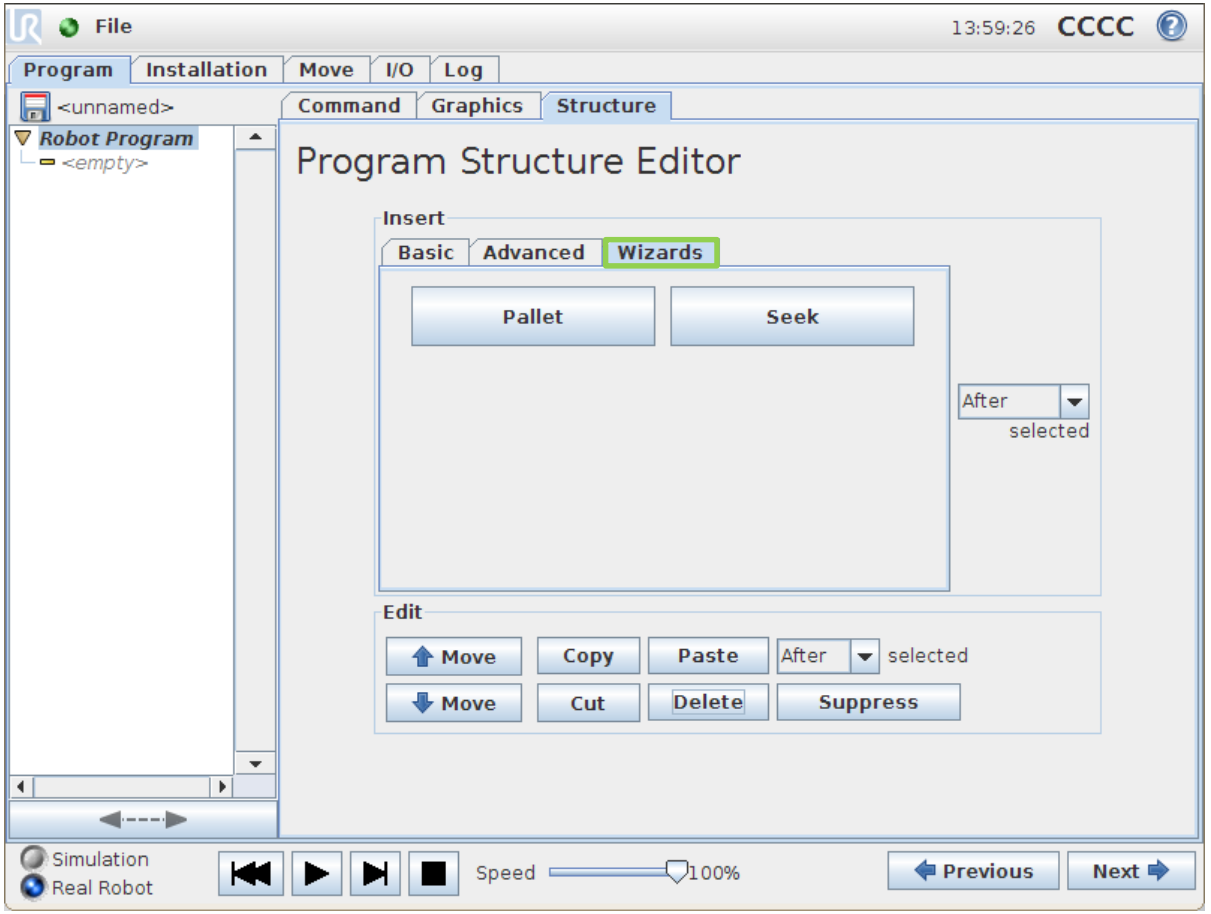
## Lab exercise

- 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.

UNIVERSAL ROBOTS

# Wizards

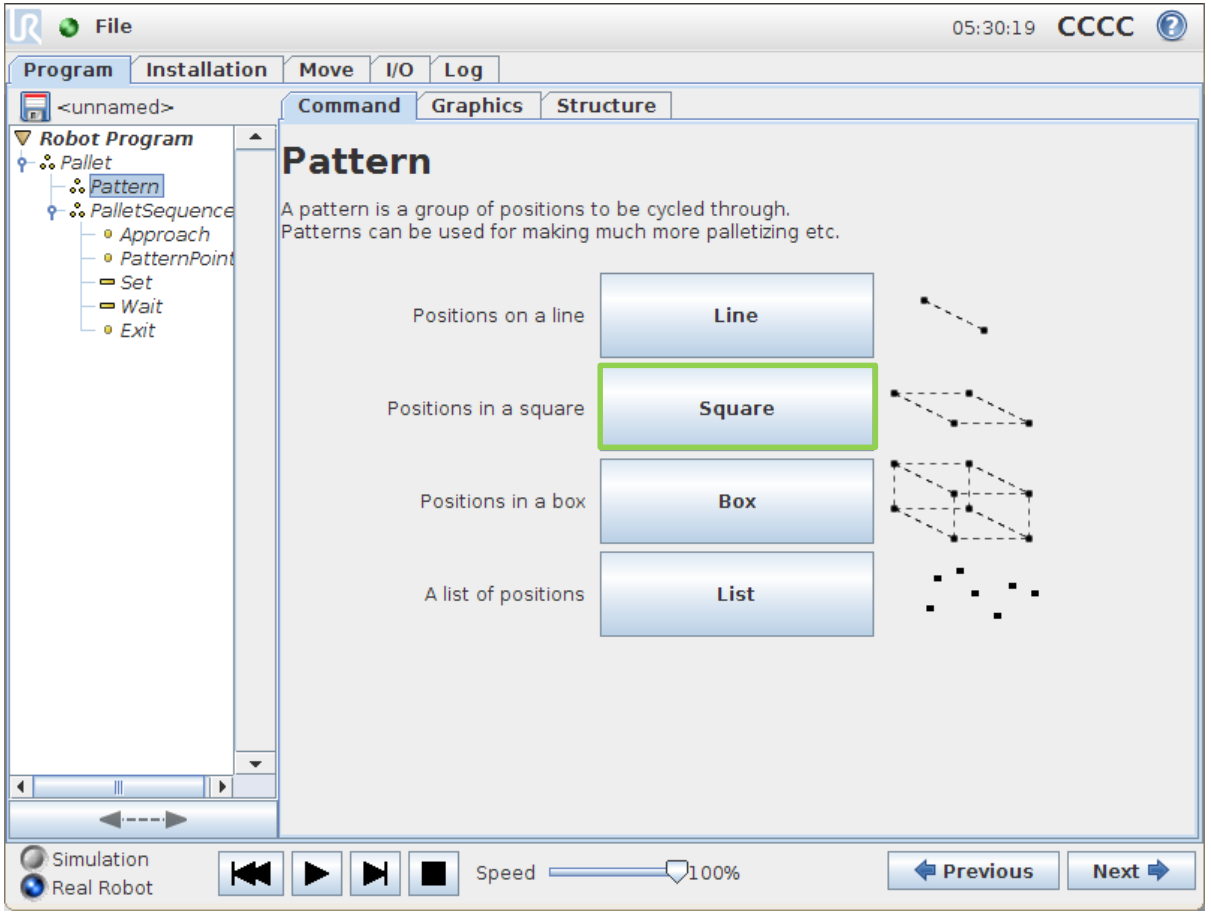
- Pallet
  - Palletizing/depalletizing
  - Patterns
- Seek
  - Search function
- Euromap
  - Injection molding
  - Optional



UNIVERSAL ROBOTS

# Pallet wizard

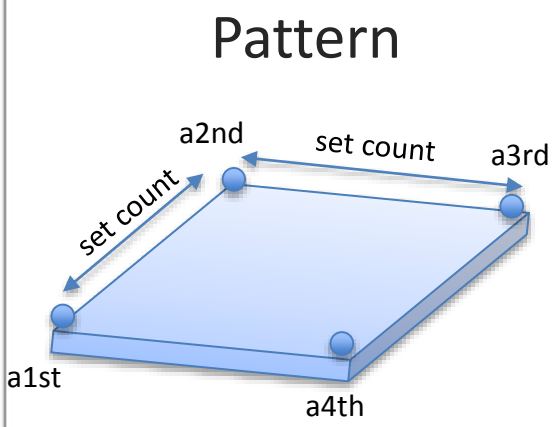
- Pattern
  - Determine palletizing pattern
  - Patterns
    - Line
    - Square
    - Box
    - List
- PalletSequence
  - What to do at each position in pattern

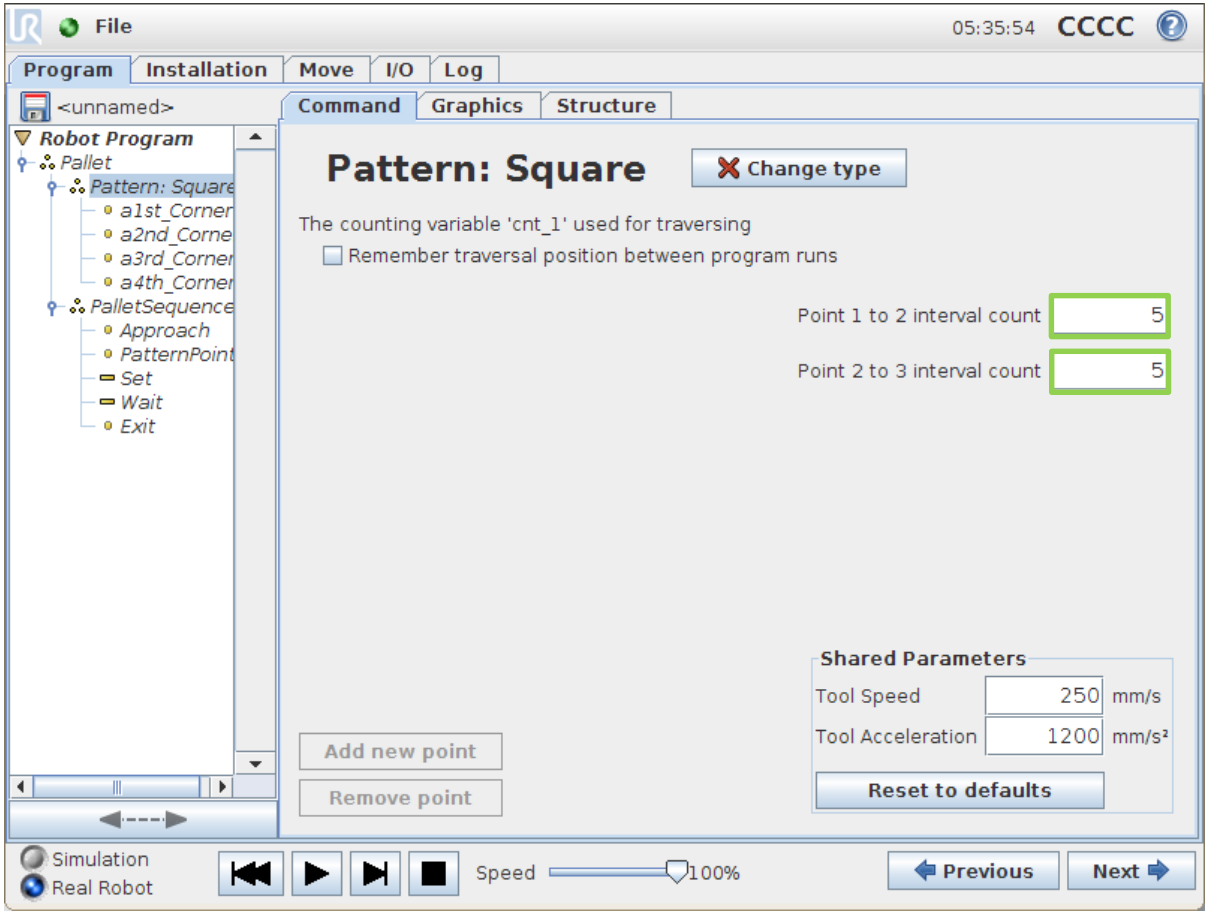


UNIVERSAL ROBOTS

# Pattern: Square

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



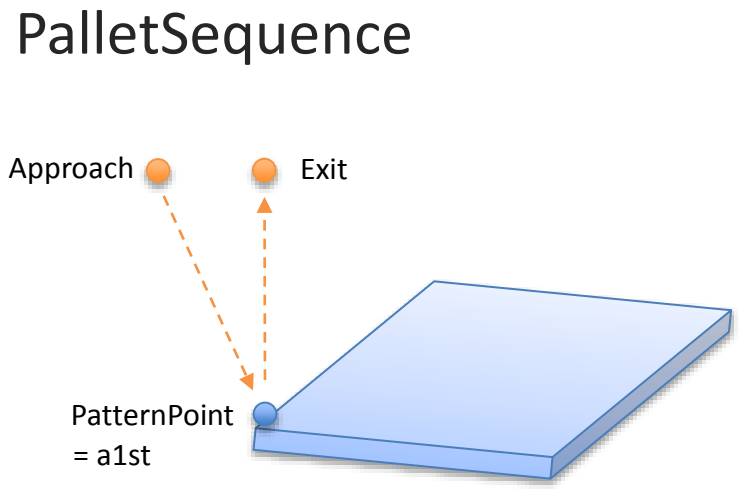



UNIVERSAL ROBOTS

# 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
```

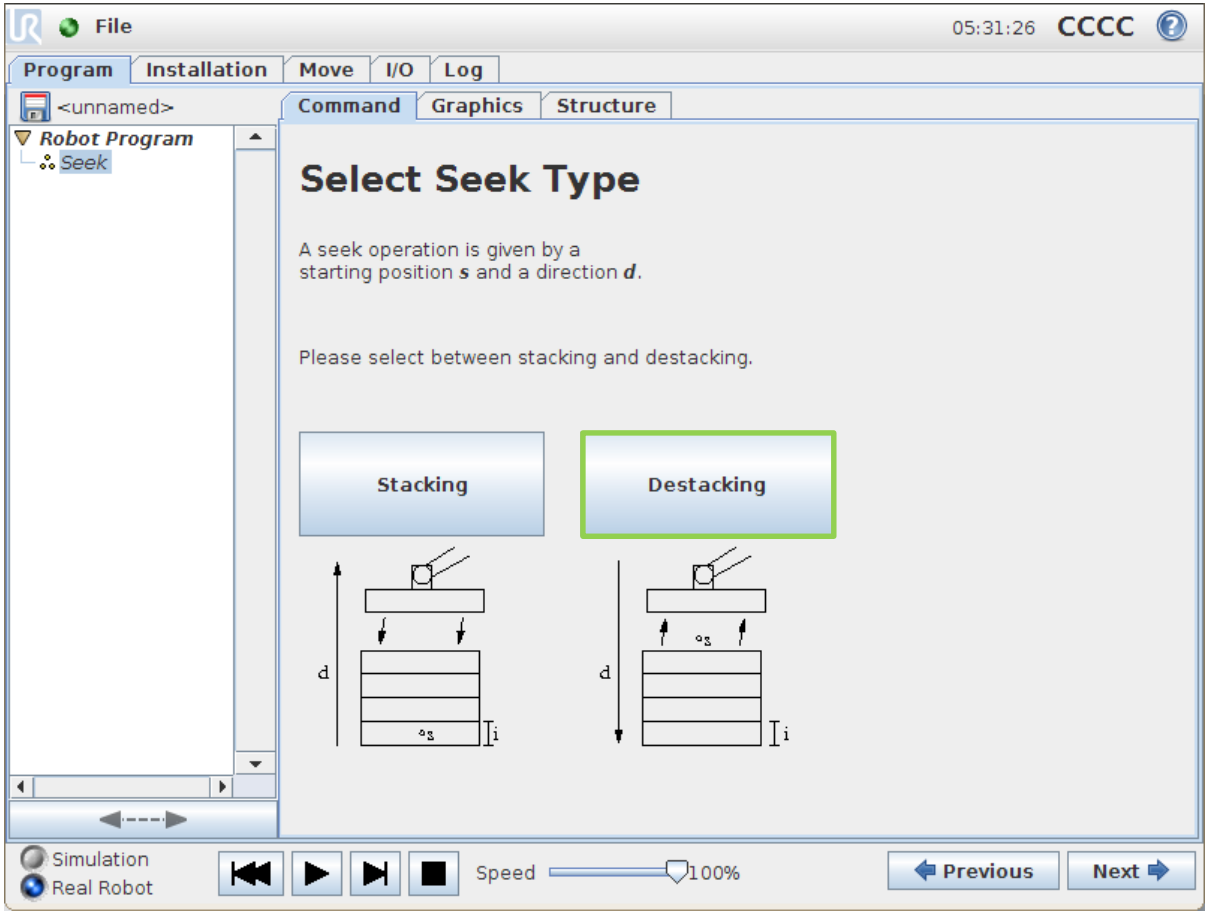



- Save sample program as pallet.urp

UNIVERSAL ROBOTS

# Seek wizard

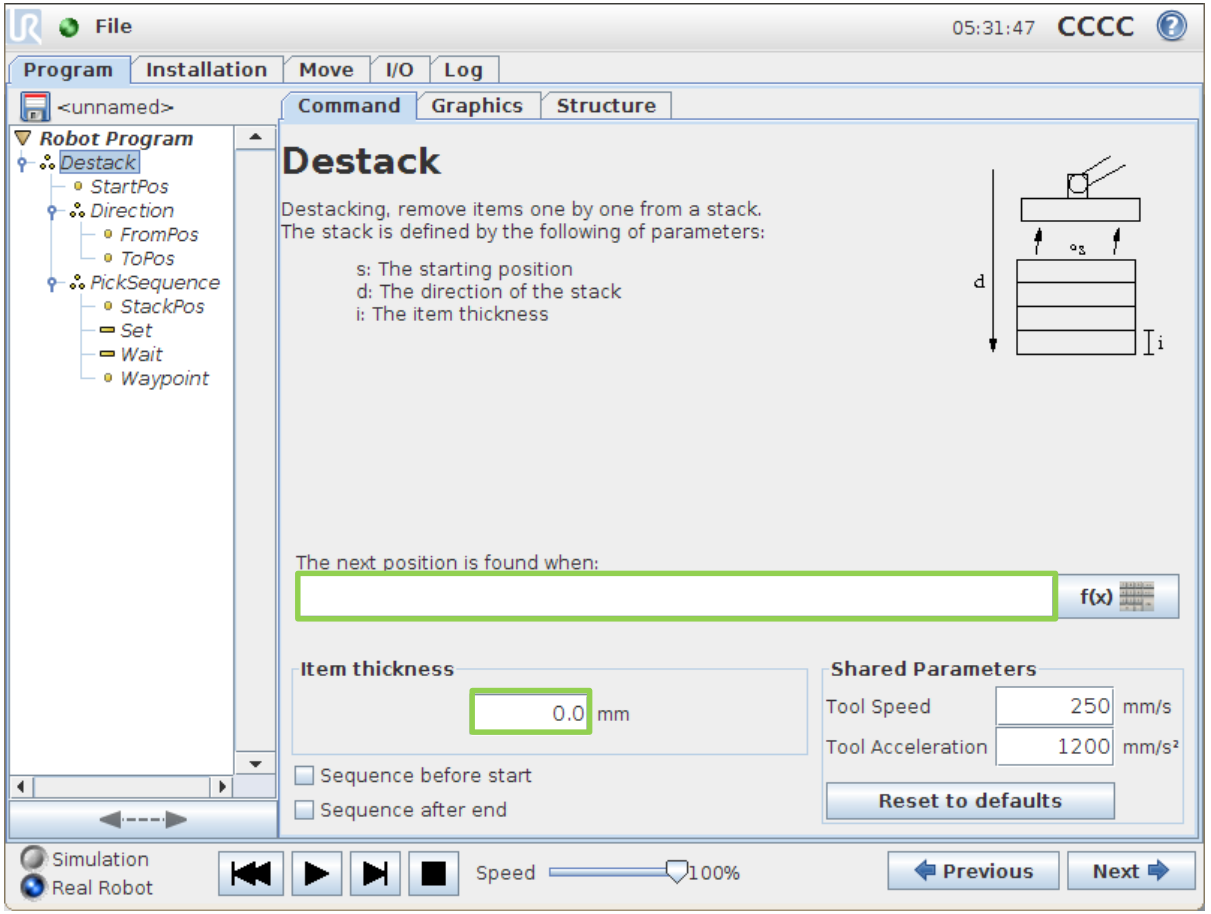
- Stacking
  - Add items to a stack
- Destacking
  - Remove items from stack



UNIVERSAL ROBOTS

# Destacking

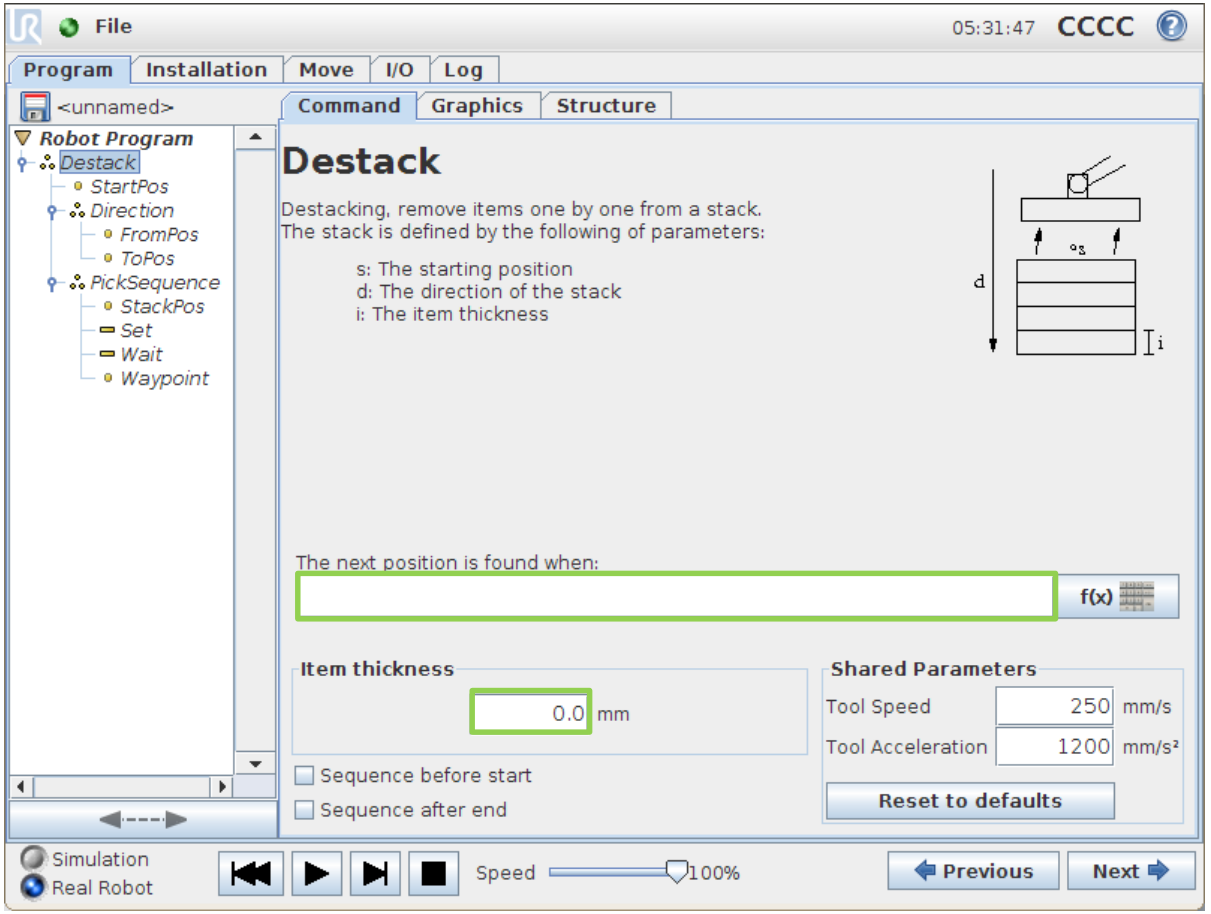
- Sample
  - Add Seek to program
  - Select Destack
- Destack
  - Set StartPos
    - From where to start seek operation
  - Set Direction of stack
    - Can be any linear direction
    - FromPos
    - ToPos
  - Set item thickness
    - Same thickness for all items
  - Set sensor input
    - To determine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



UNIVERSAL ROBOTS

# Destacking

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



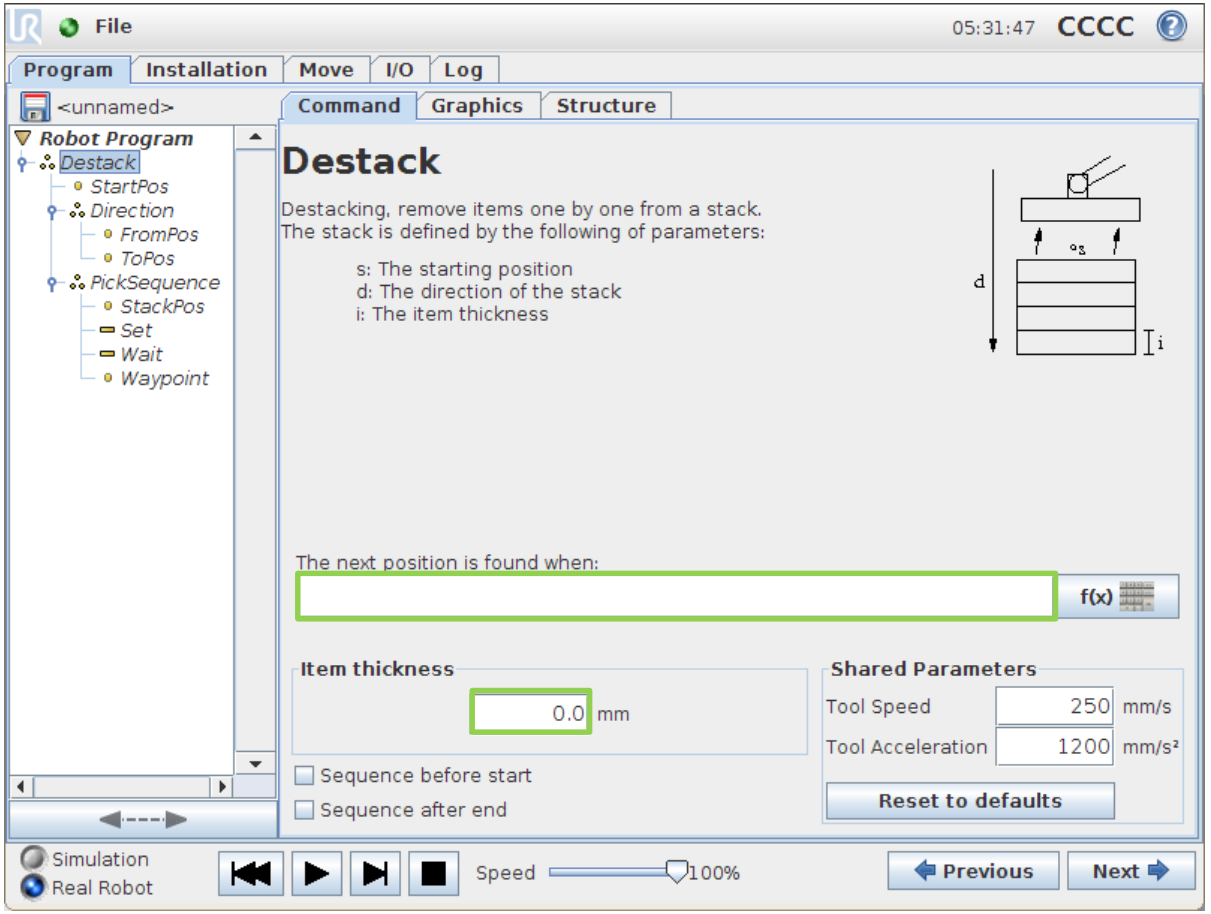

- rule of thumb is to teach StackPos as same point as StartPos

UNIVERSAL ROBOTS

# 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




- Save sample program as seek_destack.urp

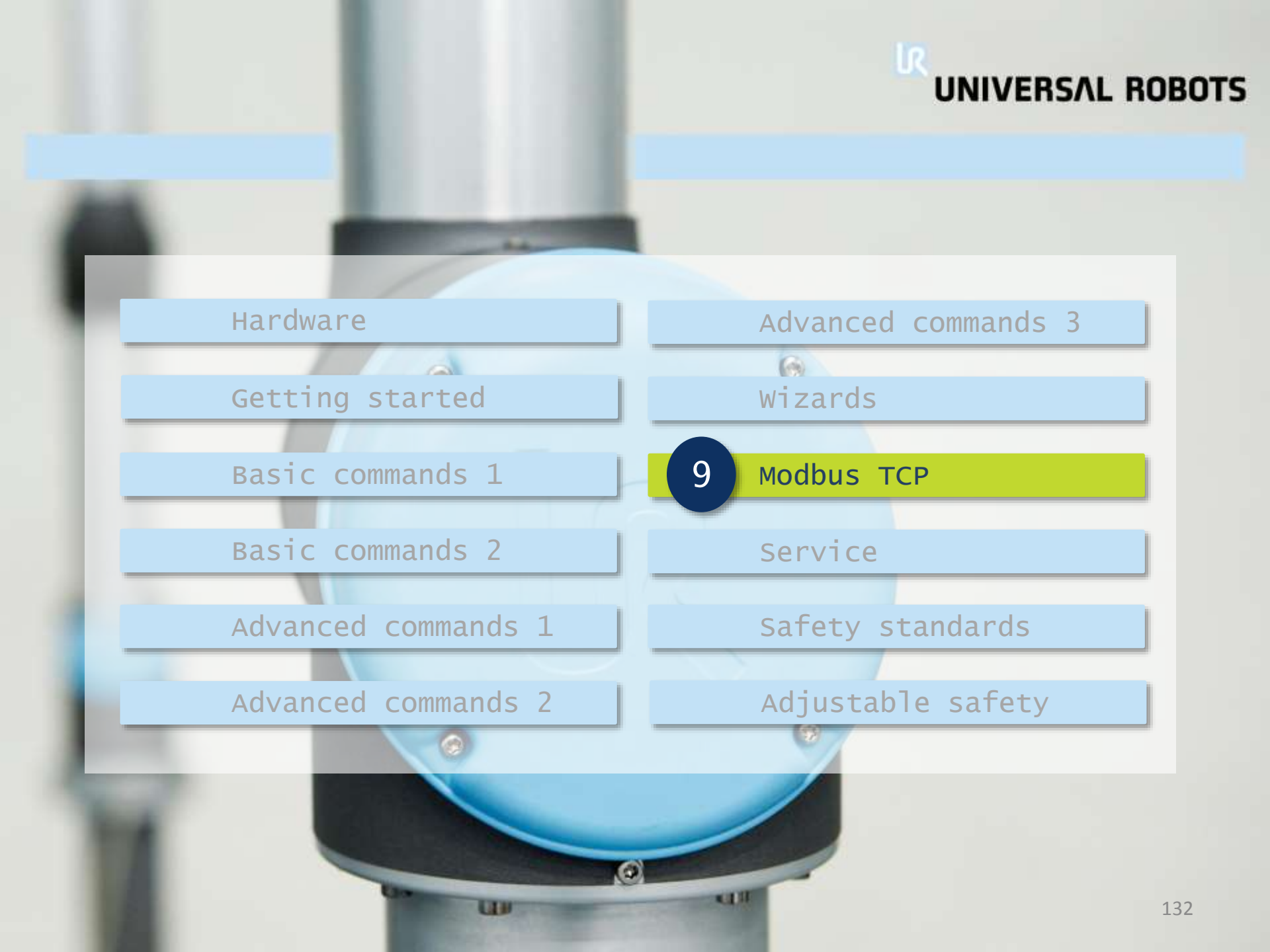
UNIVERSAL ROBOTS

## 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 |



UNIVERSAL ROBOTS

Hardware

Getting started

Basic commands 1

Basic commands 2

Advanced commands 1

Advanced commands 2

Advanced commands 3

Wizards

9 Modbus TCP

Service

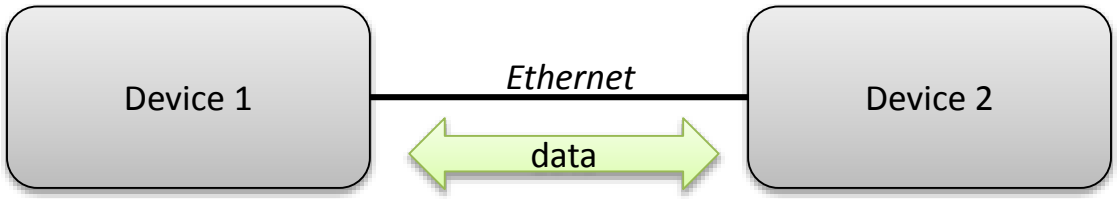
Safety standards

Adjustable safety

UNIVERSAL ROBOTS

## 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

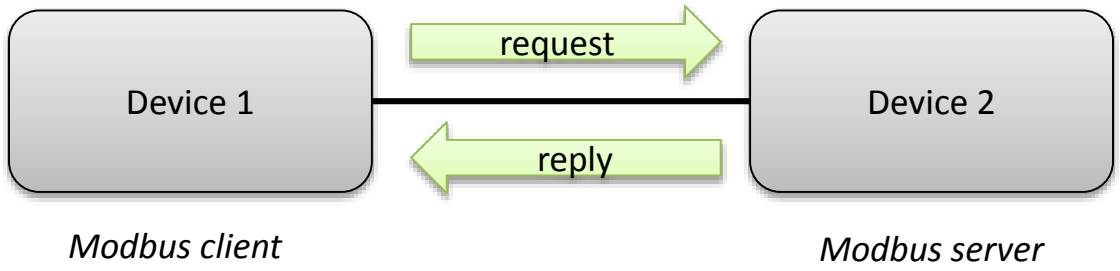



- Client / Server relationship

UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

## Data types

- Available data types for Modbus TCP

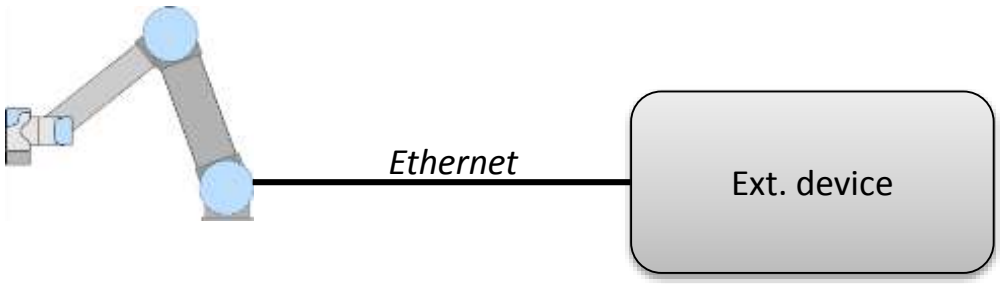
| data type | value | address range |
|---|---|---|
| Digital inputs | On/Off | *Consult documentation provided by vendor of Server device* |
| Digital outputs | On/Off | |
| Register inputs | 16 bit | |
| Register outputs | 16 bit | |

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

UNIVERSAL ROBOTS

## Modbus TCP

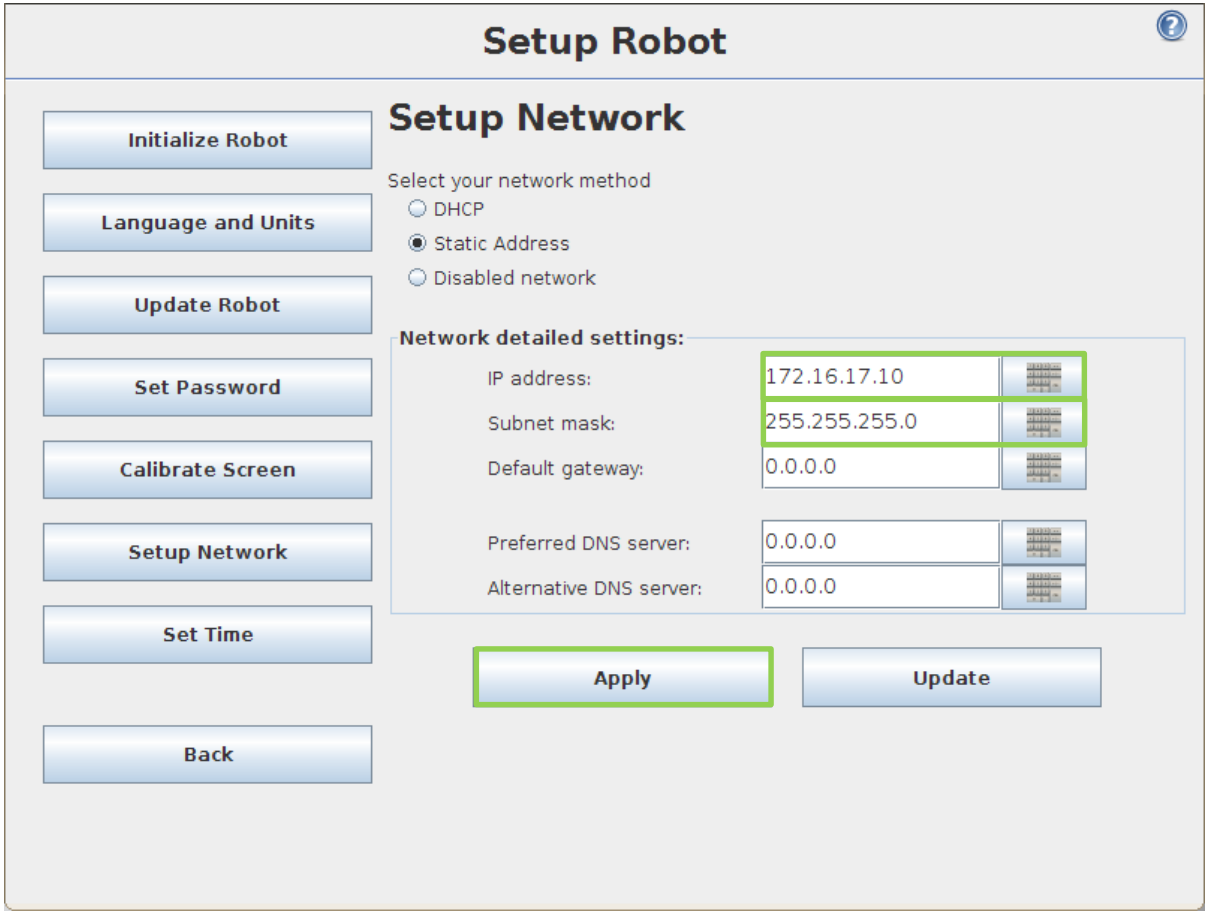
- Sample
  - Use robot as Client and connect to a Server



UNIVERSAL ROBOTS

# Setup network

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




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

UNIVERSAL ROBOTS

## Server

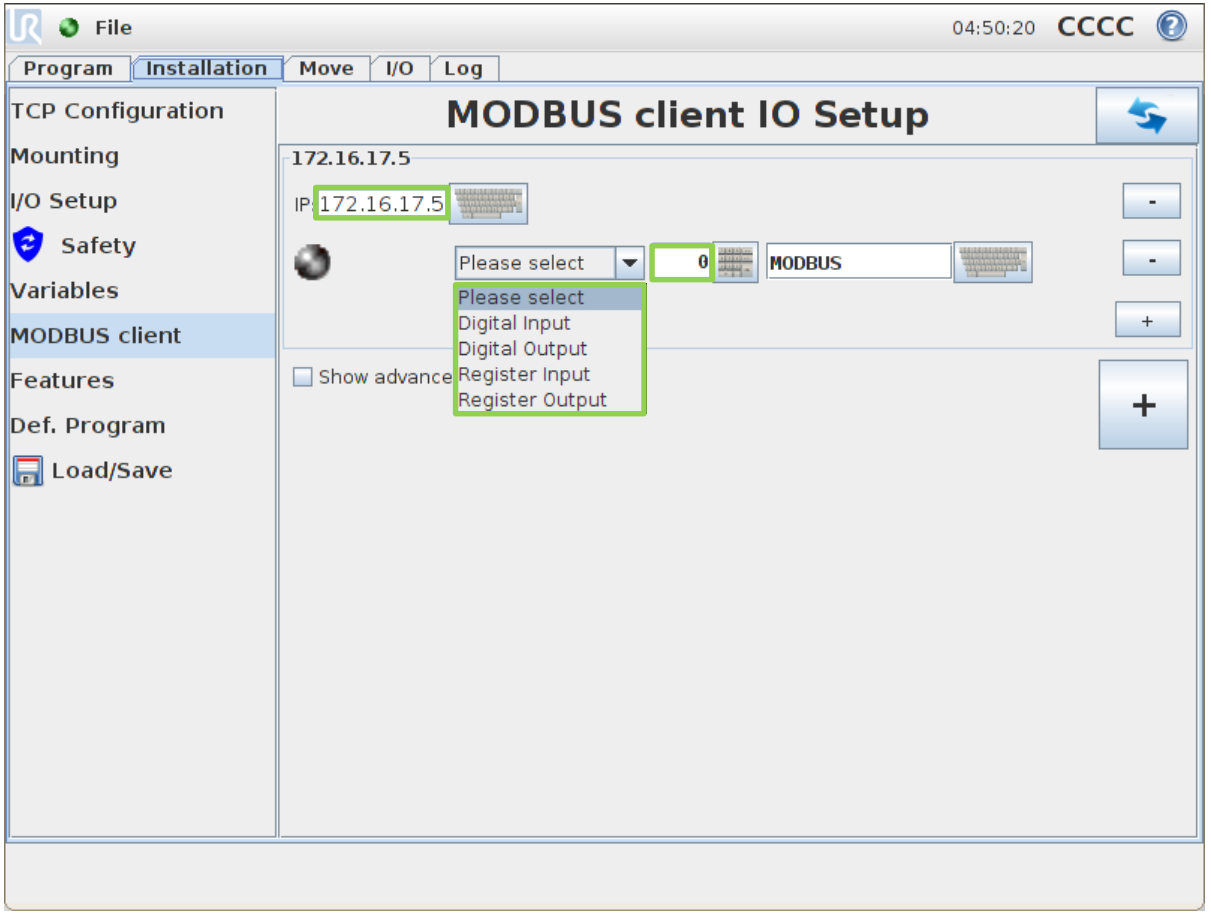
- Phoenix Contact ILB ETH 24 DI16 DIO16 2TX
  - 16 digital outputs
  - 16 digital inputs
  - 2-port switch

| | Modbus Register Table (16-Bit Words) | Modbus Input Discretes Table (Bits) | Modbus Coil Table | Access | Function |
|---|---|---|---|---|---|
| Process data | 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 |
| Diagnostics | 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) |
| Special register | 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 |

UNIVERSAL ROBOTS

## 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

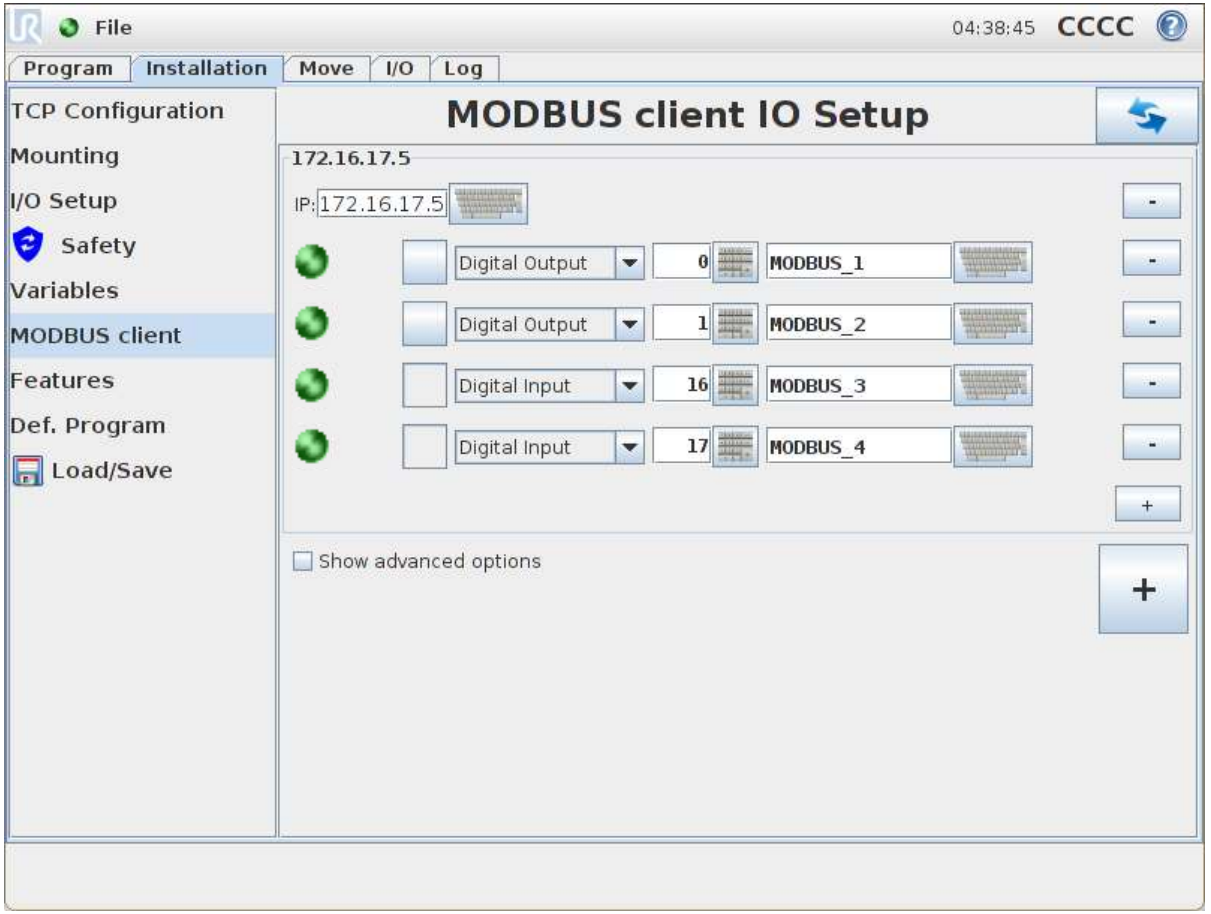


UNIVERSAL ROBOTS

## Setup server

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

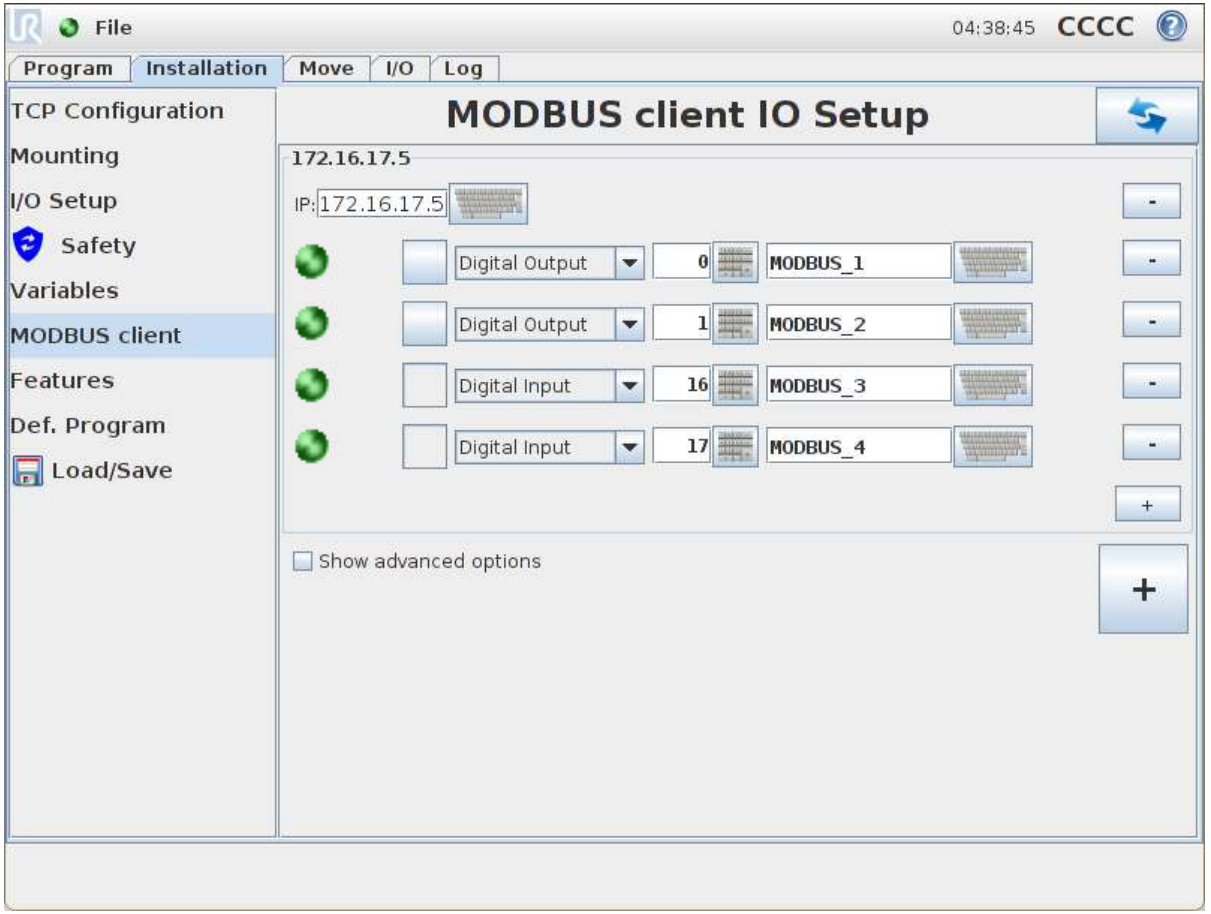
| | Status |
|---|---|
| | Connection ok |
| | Update frequency warning |
| | No connection |
| E4 | Exception code |



UNIVERSAL ROBOTS

# Use Modbus signal in program

- Use of signals
  - Same functionality as normal digital signals



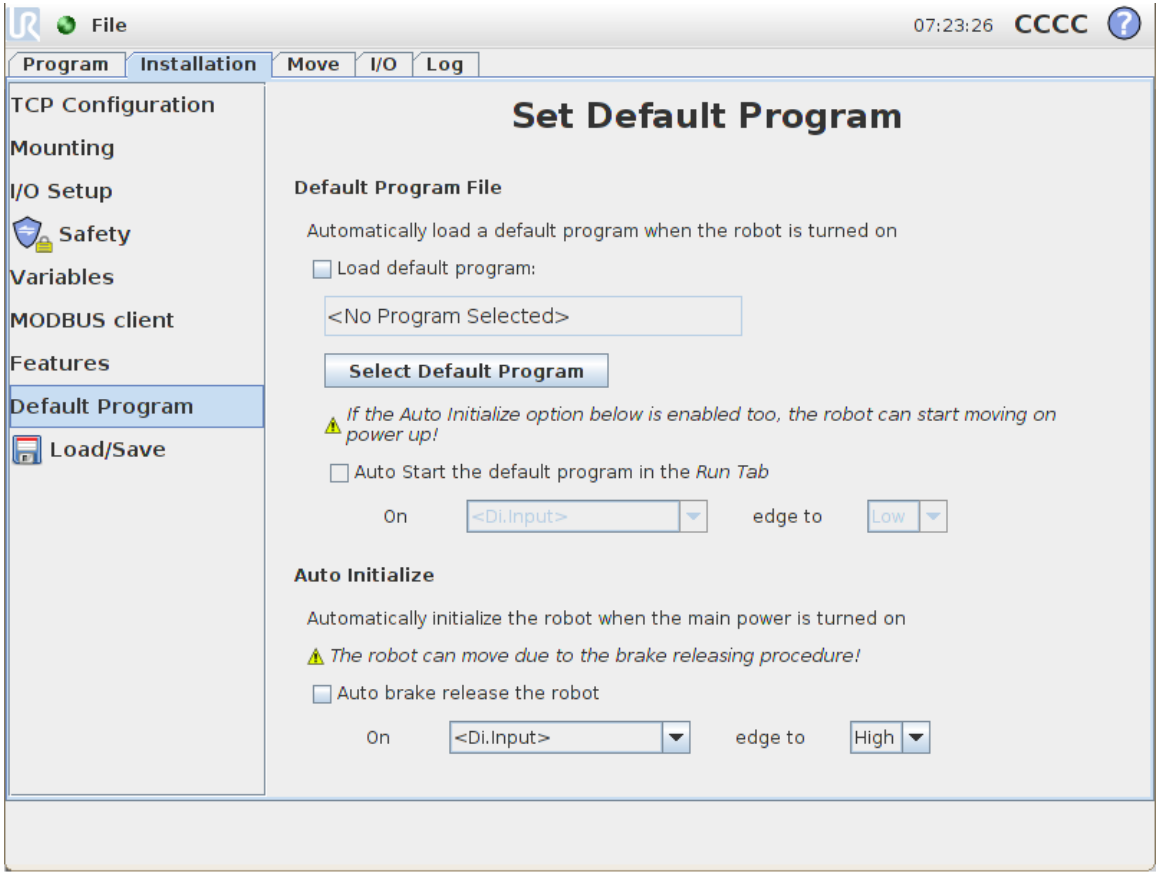

Robot Program
MoveL
Waypoint_1
Set MODBUS_1 = True
Waypoint_2
Waypoint_3
Waypoint_4
Set MODBUS_1 = False

- Save sample program as modbus.urp

UNIVERSAL ROBOTS

# 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



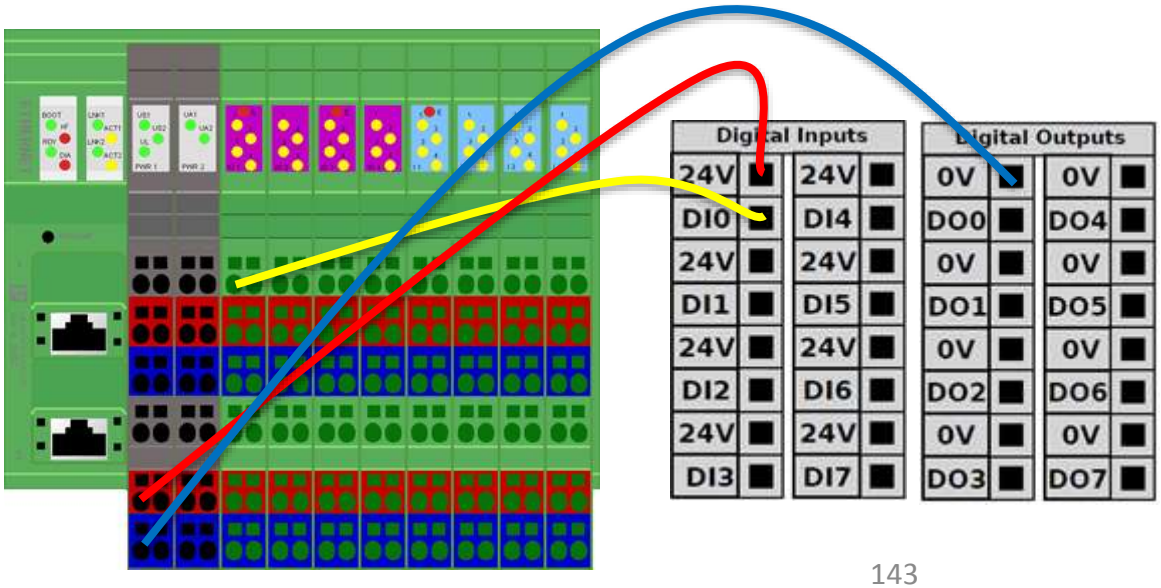

- Note: robot will *always* load default.installation at startup

UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

## 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.

UNIVERSAL ROBOTS

## Downloads and tips

support.universal-robots.com

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

UNIVERSAL ROBOTS

# Magic files

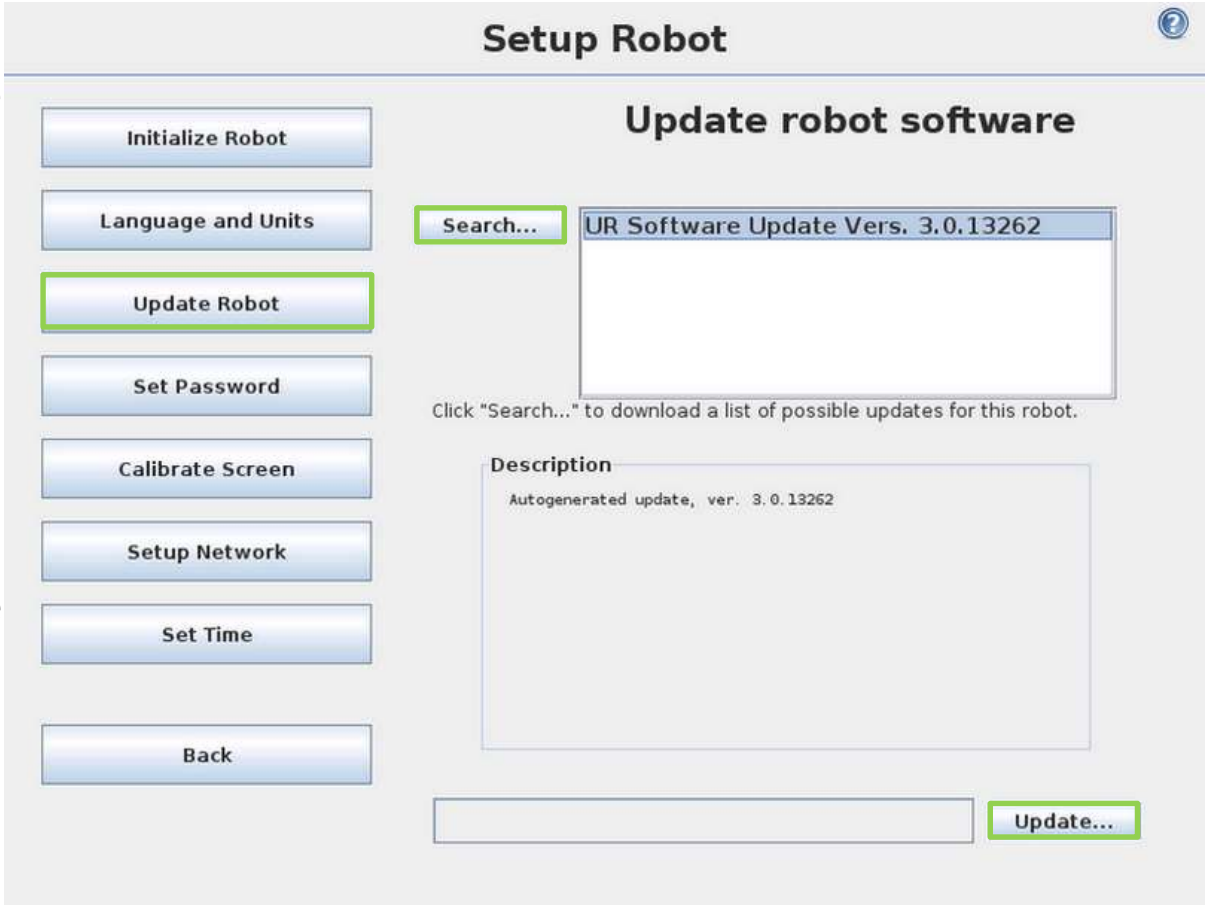
- Easy backup
  - Backup programs
  - Backup log history
  - Backup configuration files
- Others
  - Upload programs
  - Screenshot of GUI

- 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

UNIVERSAL ROBOTS

# 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

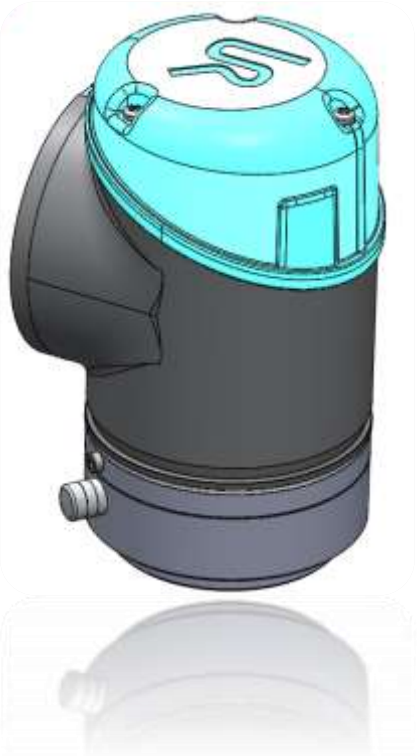



- IMPORTANT: ALWAYS MAKE FULL BACKUP BEFORE UPDATING SOFTWARE!

UNIVERSAL ROBOTS

# Firmware update

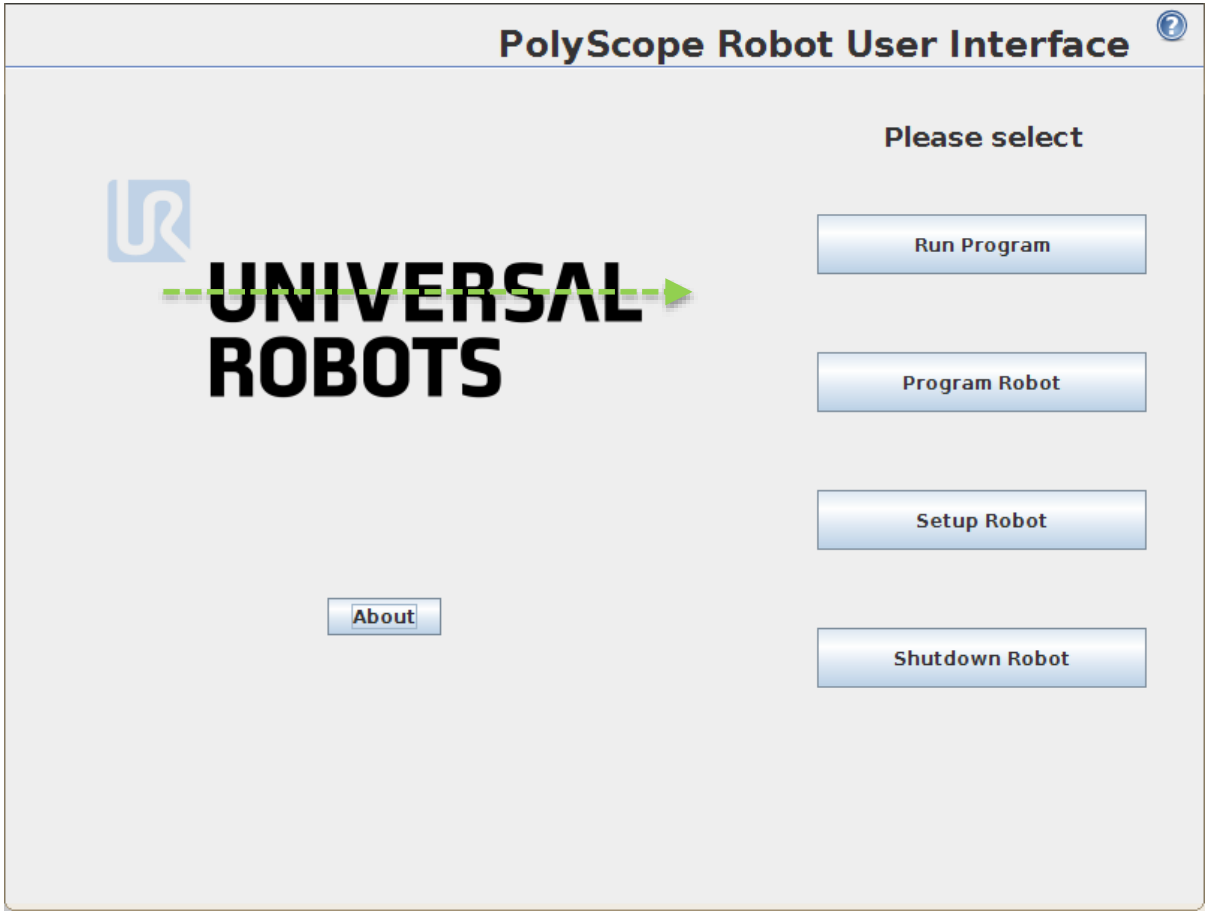
- Firmware is software located in each joint
  - Firmware controls the joint
  - Can be updated if neccessary
- Software update *is required* prior to updating firmware
  - During software update, the firmware is automatically copied to flash card
  - No additional download required



UNIVERSAL ROBOTS

# Firmware update

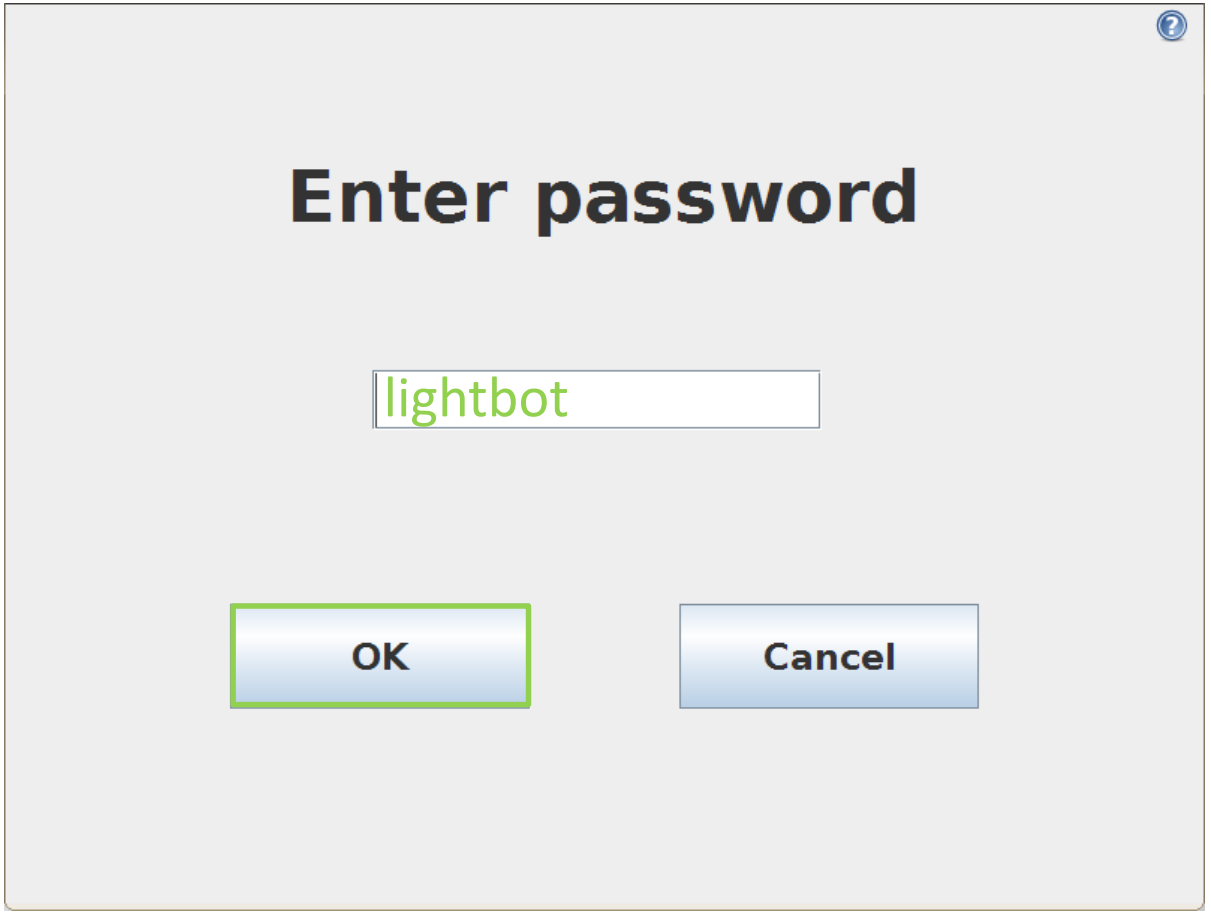
- Update procedure
  - Drag finger across UNIVERSAL logo on Welcome screen



UNIVERSAL ROBOTS

# Firmware update

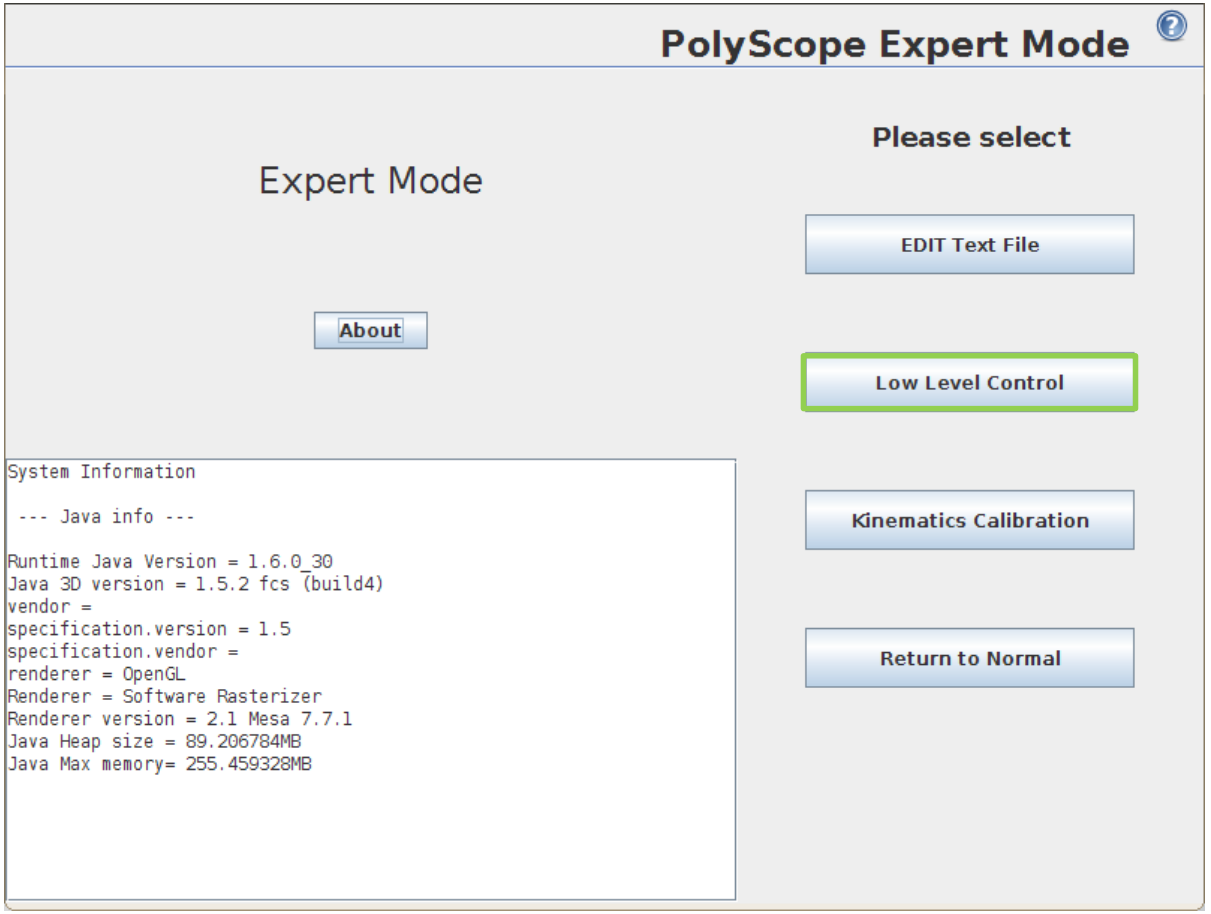
- Update procedure
  - Enter password *lightbot*
  - Press OK to access Expert Mode



UNIVERSAL ROBOTS

# Firmware update

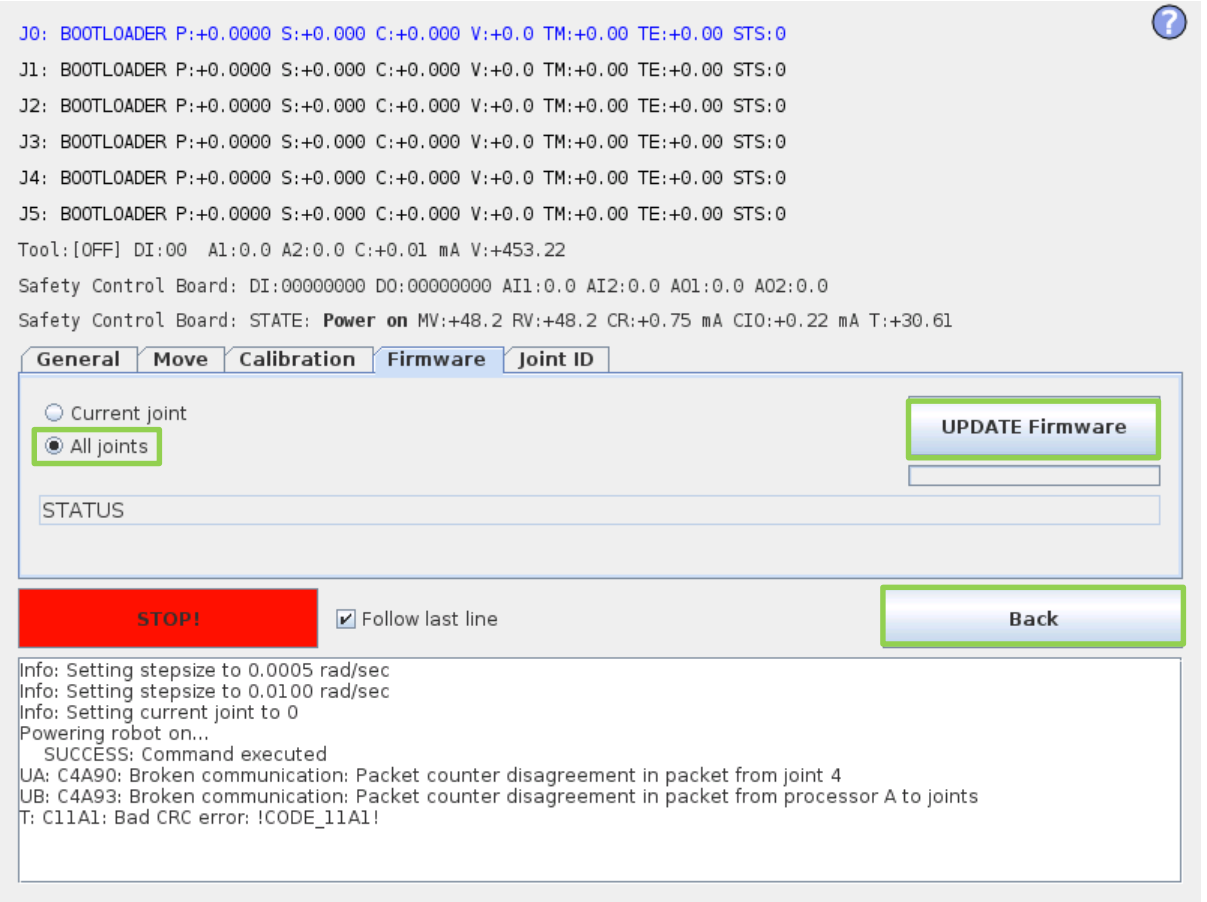
- Update procedure
  - Press ”Low Level Control”



UNIVERSAL ROBOTS

# 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”



UNIVERSAL ROBOTS

# Offline simulator

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

*Virtual machine*

UNIVERSAL ROBOTS

next

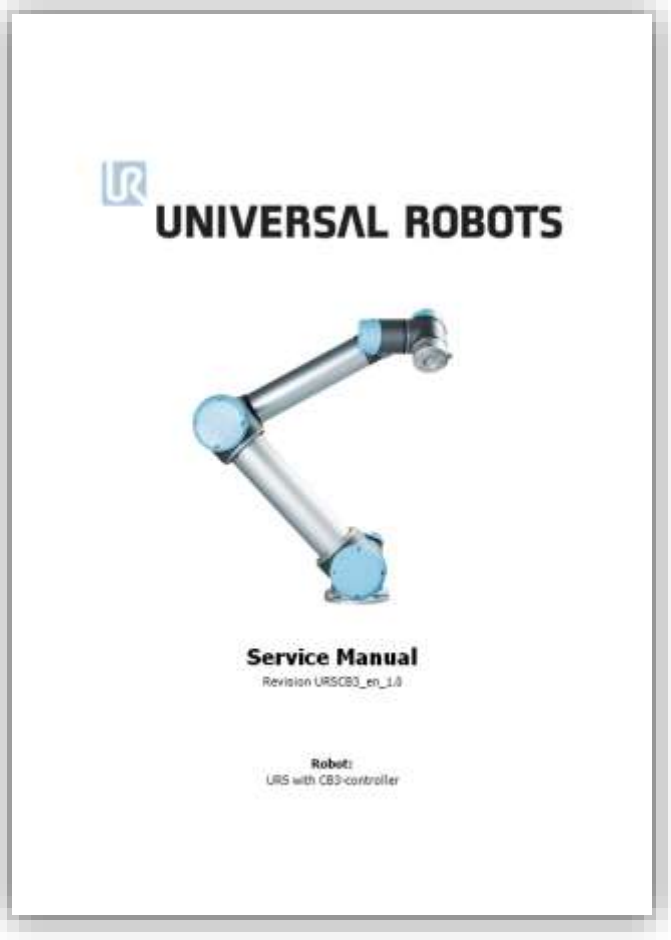
# Maintenance

UNIVERSAL ROBOTS

# 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

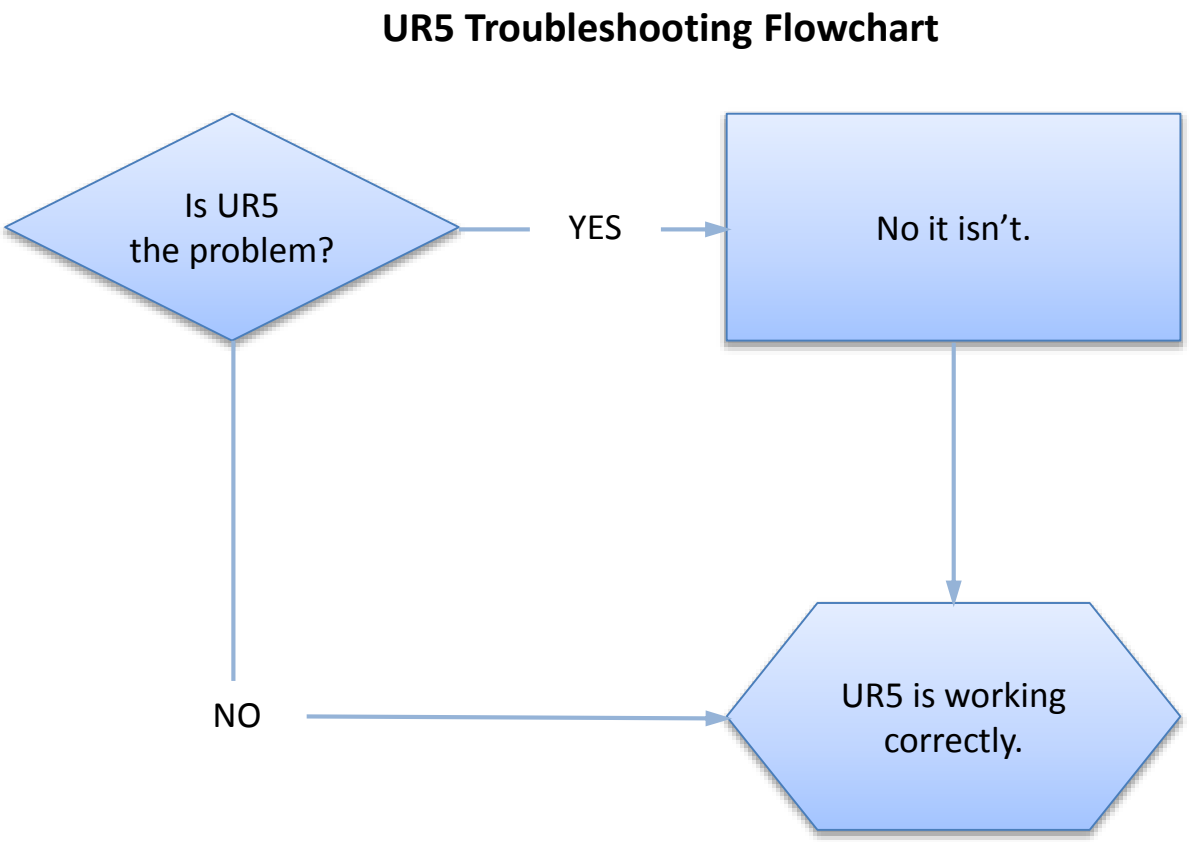
- Preventive maintenance
  - Download Service Manual from support site
  - Walk through *Chapter 2. Preventive Maintenance*



UNIVERSAL ROBOTS

# Service manual

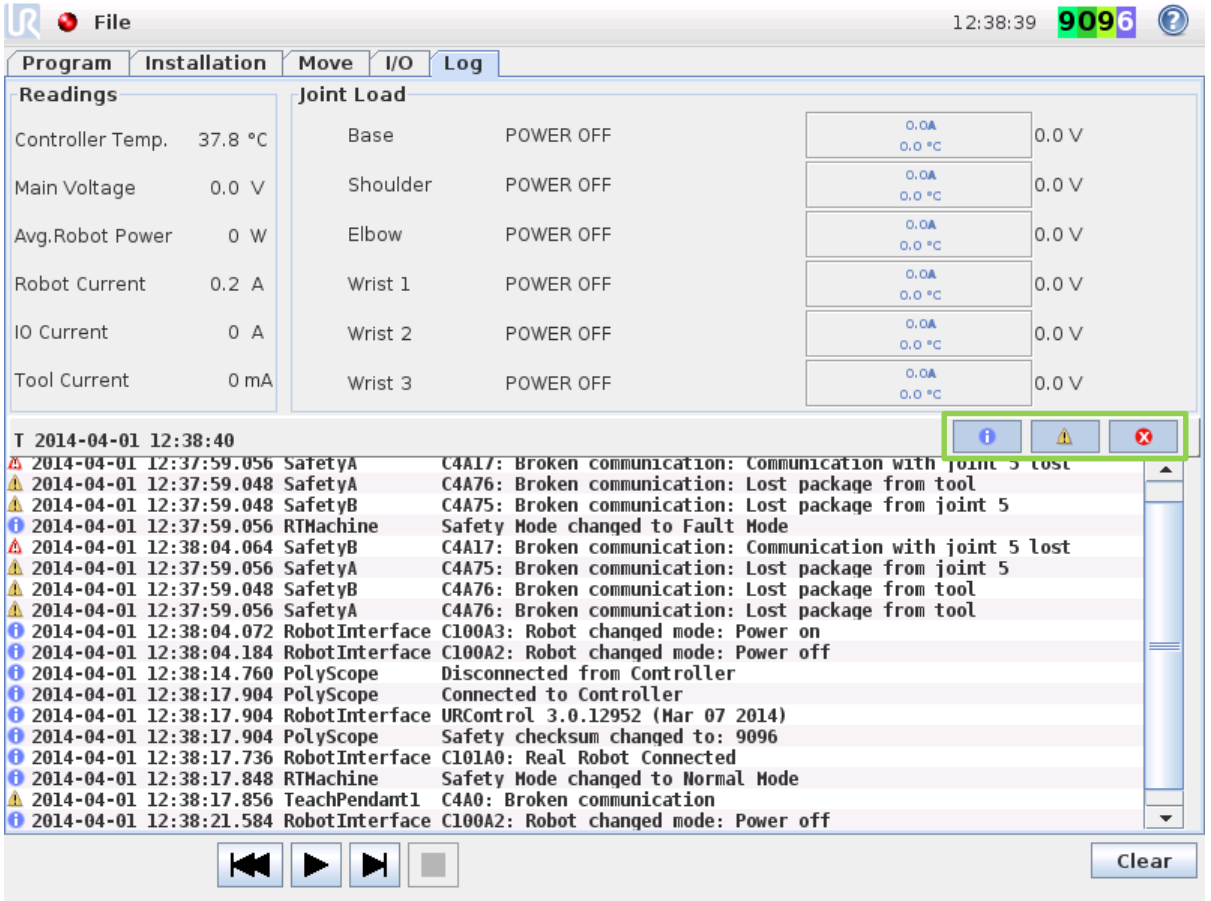
- Troubleshooting
  - Log history
  - Joint replacement
  - Joint calibration
  - Change joint ID
  - Warranty claim



UNIVERSAL ROBOTS

# Log history

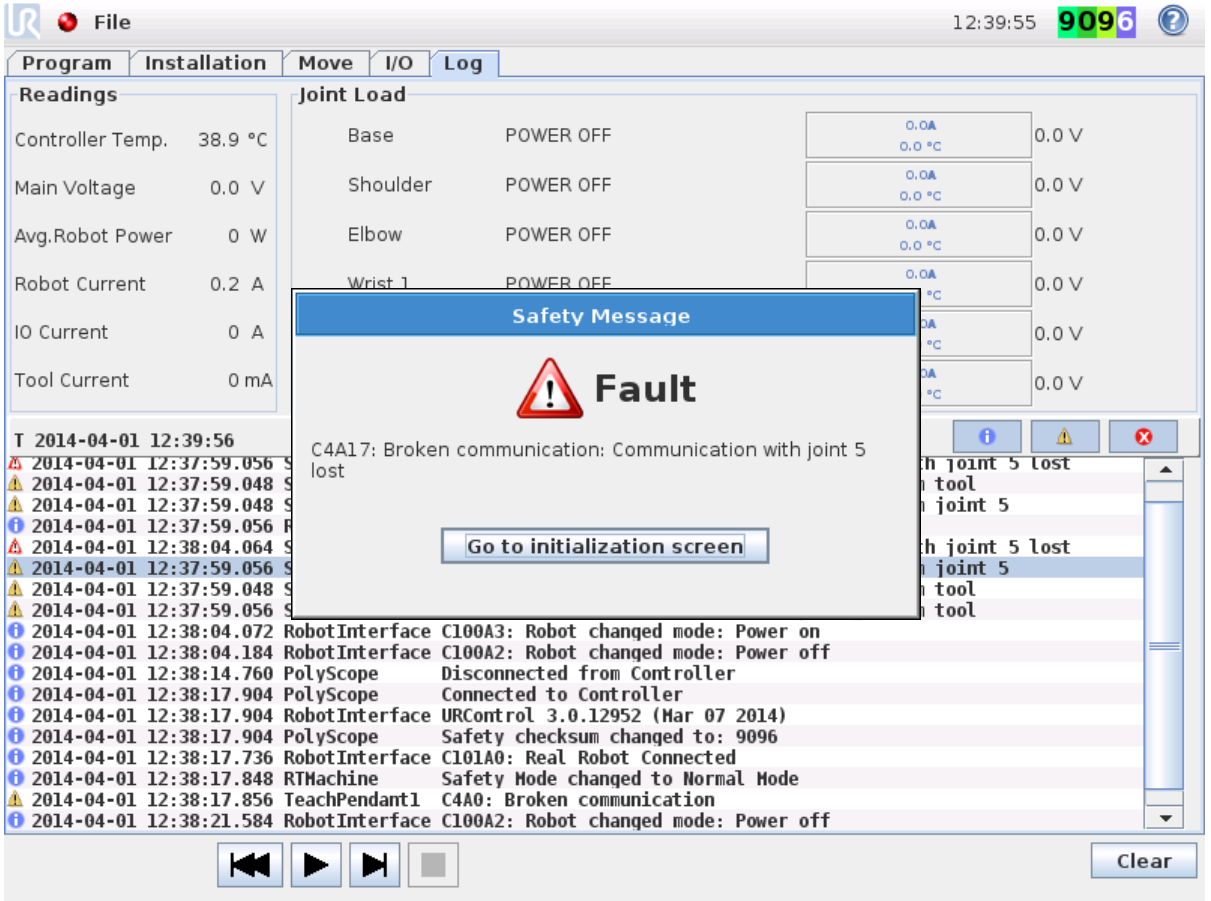
- Readings
  - Control box health
- Joint load
  - Joint status
- Log history
  - Displays valuable information about robot health
  - Show/hide
    - Information
    - Warnings
    - Errors



UNIVERSAL ROBOTS

# Log history

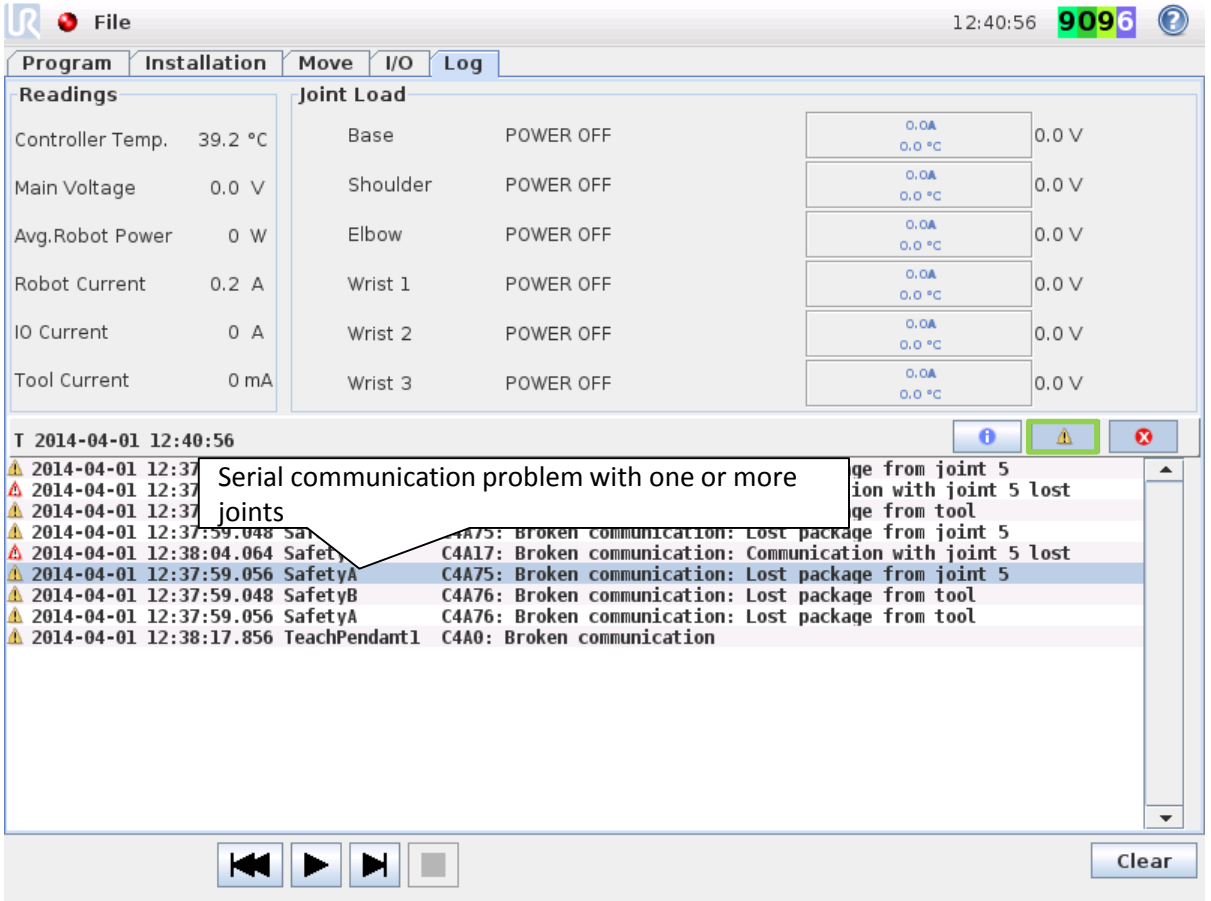
- Demonstrate error
  - Remove blue lid from Wrist 3
  - Disconnect green comm. cable from prev. joint
  - Verify that robot detects error and displays popup



UNIVERSAL ROBOTS

# Log history

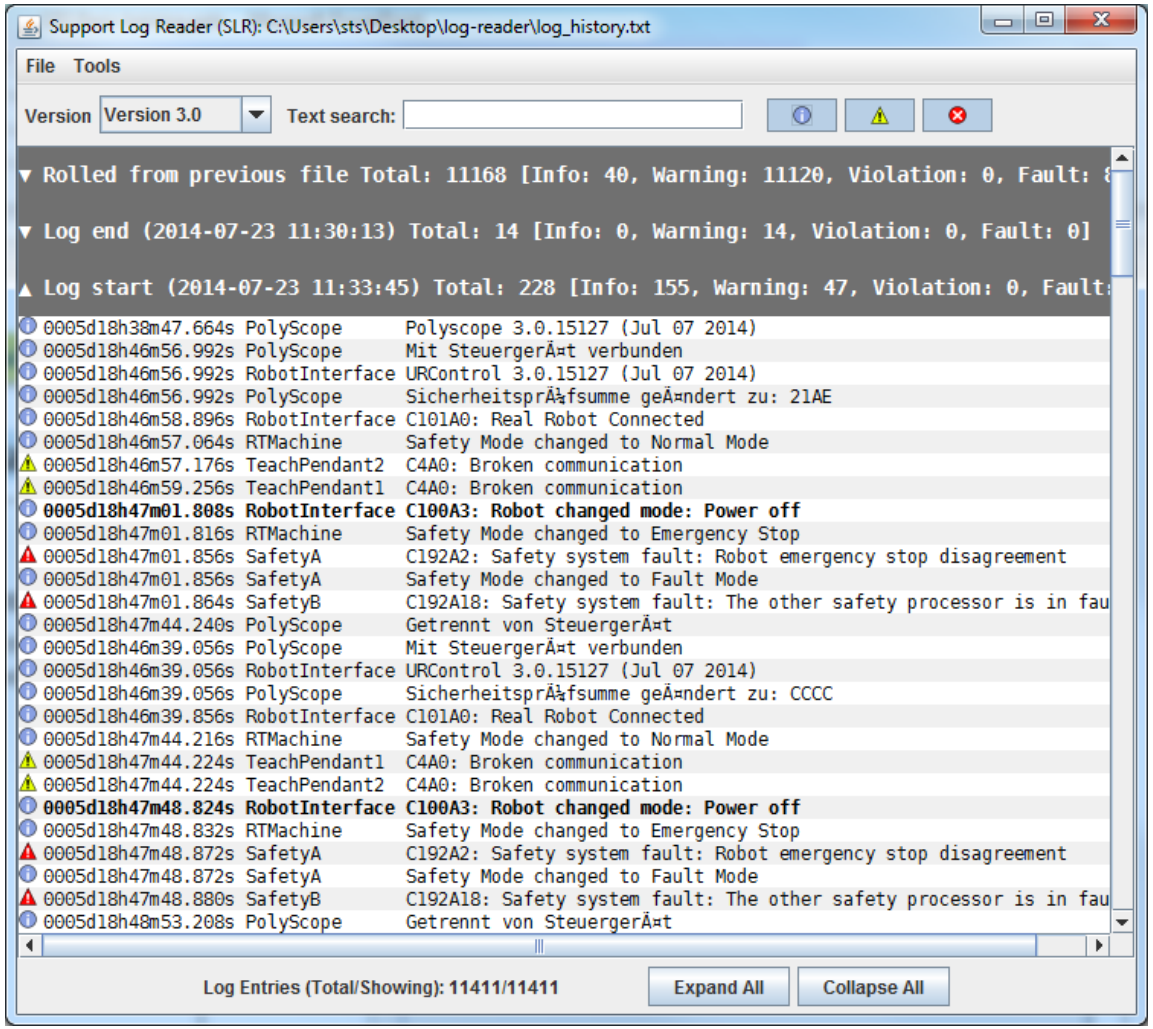
- Demonstrate error
  - Check log history
  - Show/hide Information entries
  - Highlight line by tapping once on it
- Log history
  - Saved in textfile named log_history.txt
  - Use Magic file to backup



UNIVERSAL ROBOTS

# 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



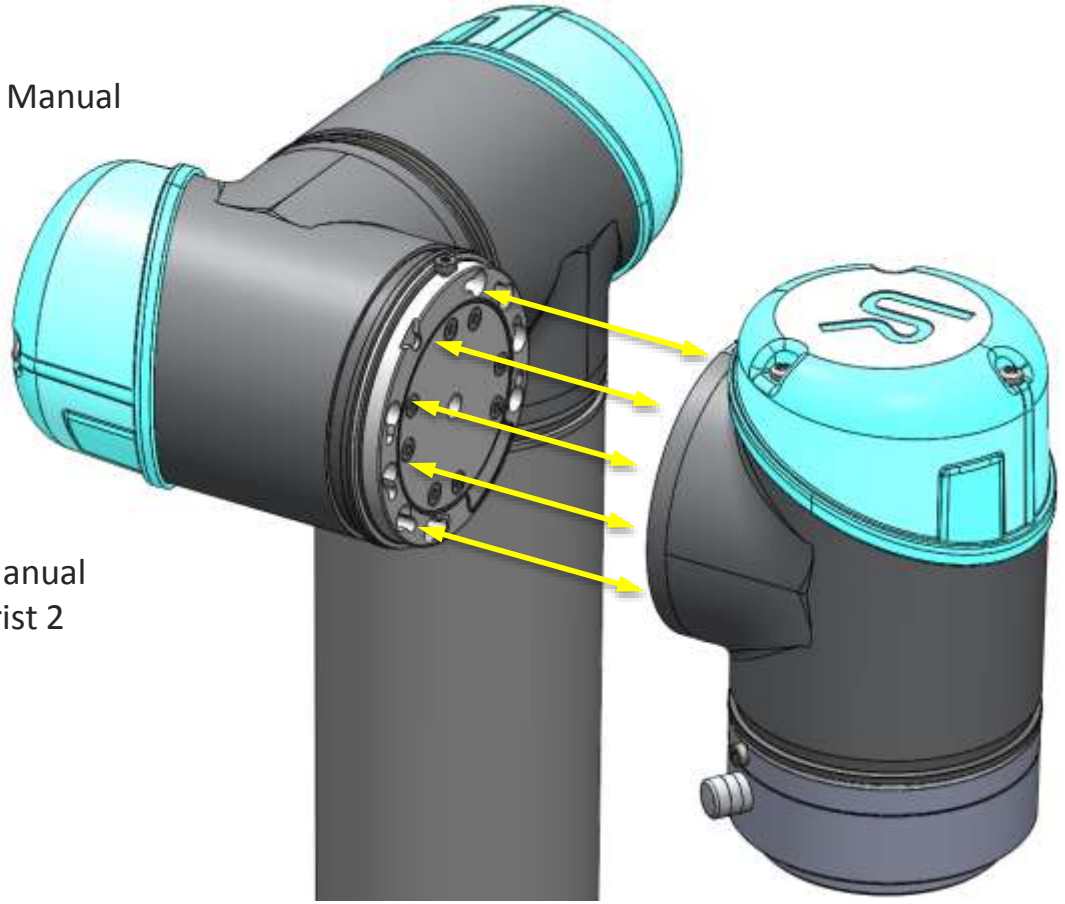
UNIVERSAL ROBOTS

# Joint replacement

UNIVERSAL ROBOTS

# 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



UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# Change joint ID

UNIVERSAL ROBOTS

# 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 |

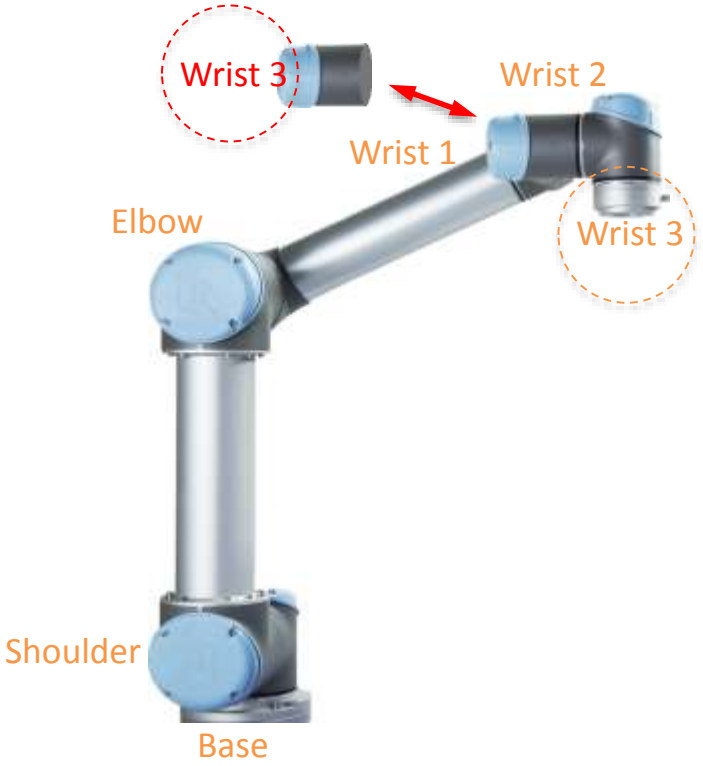

- It is *not* possible to have two joints with same ID no. on same robot
  - » ID conflict will occur, causing malfunction of robot

UNIVERSAL ROBOTS

# 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

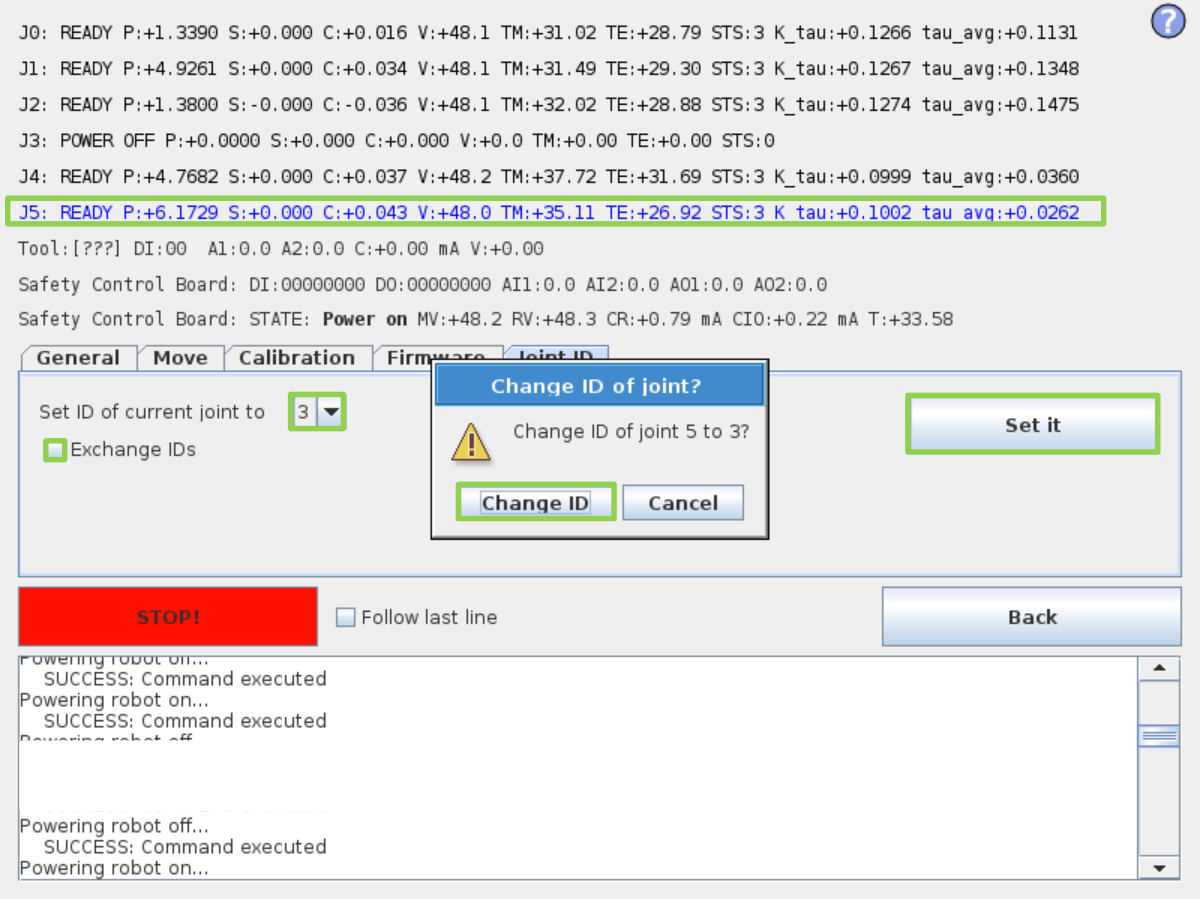
- How to solve
  - Disconnect comm. connector for joint with correct ID no.
  - » Wrist 3 is then not accessible
  - Enter Low Level Control
  - » Change ID no. for replacement joint (J5) to J3



UNIVERSAL ROBOTS

# 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




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

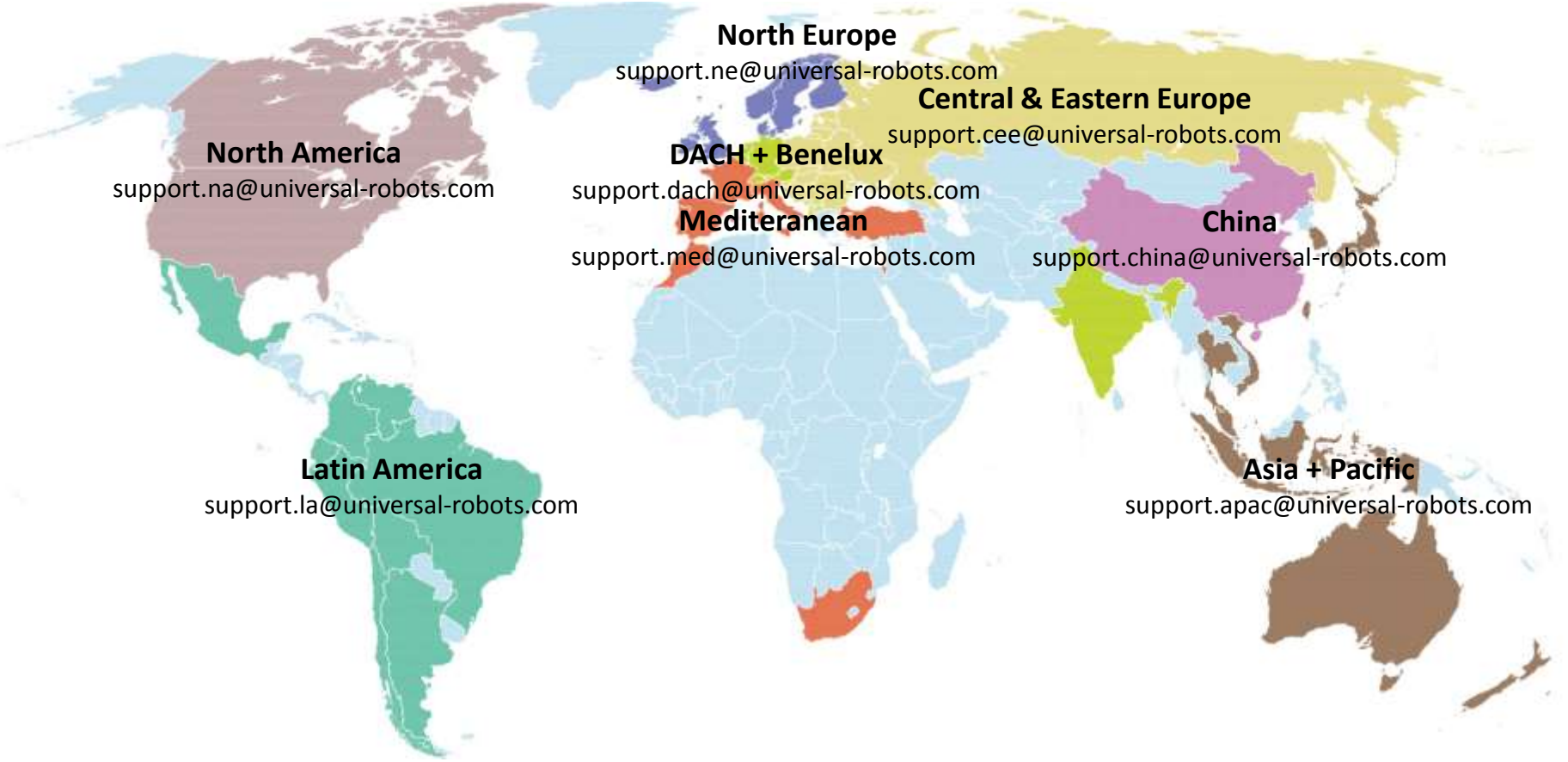
UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

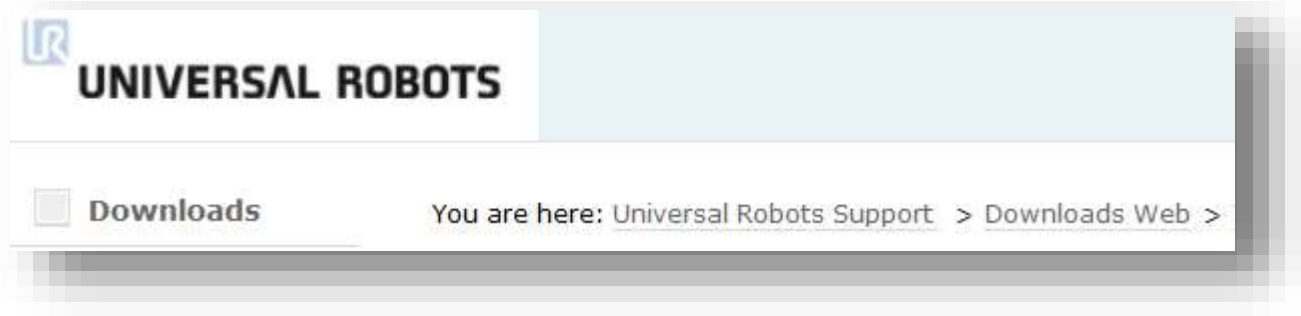
# Regional support

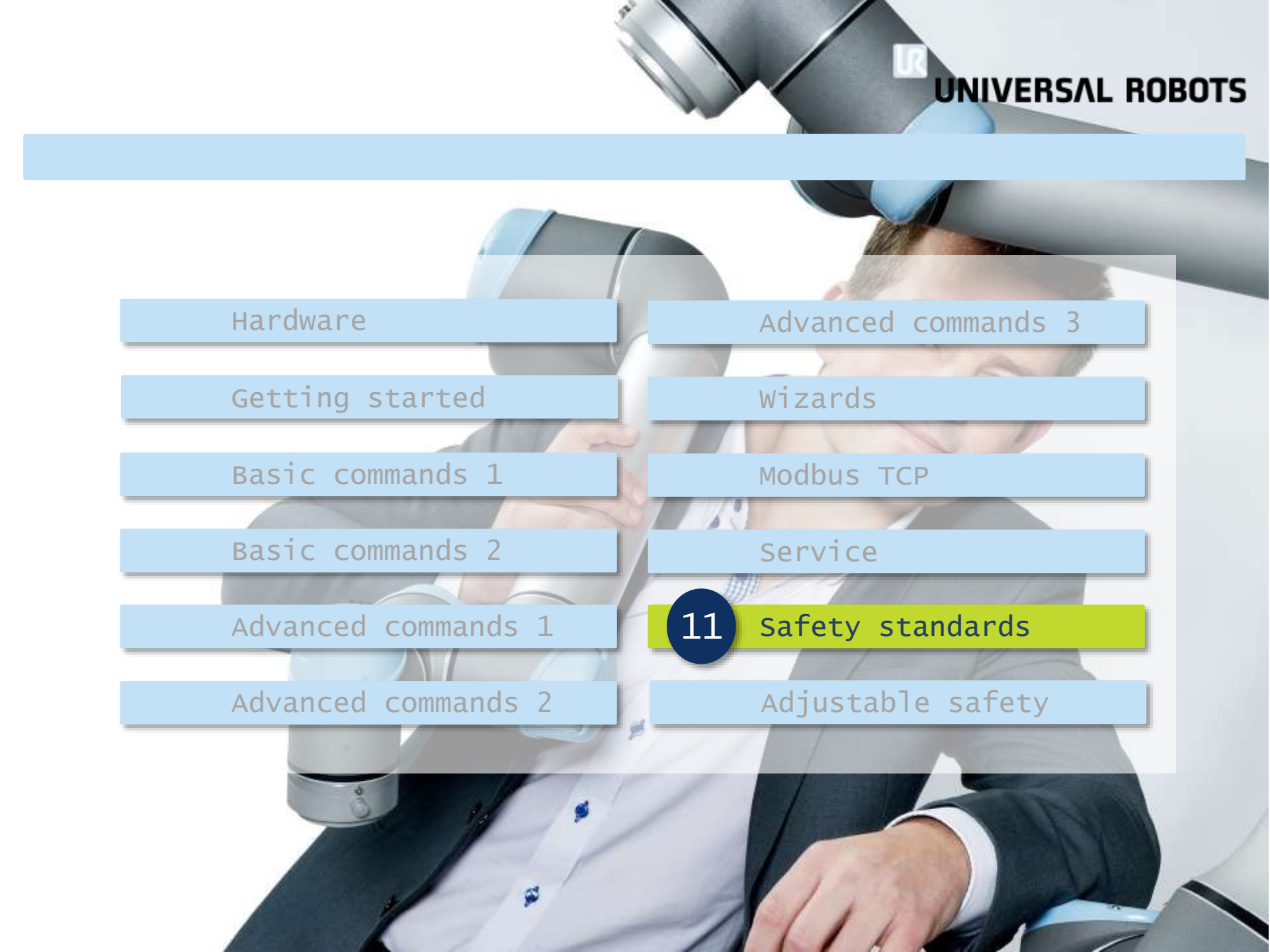


UNIVERSAL ROBOTS

## 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





UNIVERSAL ROBOTS

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

UNIVERSAL ROBOTS

## ...and now

UNIVERSAL ROBOTS

## 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 | |

UNIVERSAL ROBOTS

## 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

UNIVERSAL ROBOTS

## 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**

UNIVERSAL ROBOTS

## 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)**


TÜV NORD

ZERTIFIKAT
CERTIFICATE

Hiermit wird bescheinigt, dass die Firma / This is to certify, that the company

Universal Robots A/S
Energivej 25
5260 Odense S
Denmark

berechtigt ist, das unten genannte Produkt mit dem abgebildeten Zeichen zu kennzeichnen.
is authorized to provide the product described below with the mark as illustrated.

Geprüft nach / Tested in accordance with: EN ISO 13849:2008 PL d, EN ISO 10218-1:2011, Clause 5.4.3

Beschreibung des Produktes (Details s. Anlage 1) / Description of the product (Details see Annex 1): Universal Robots Safety System for UR 10 and UR 5 robots

Typenbezeichnung / Type description: URSafety 3.0

Bemerkung / Remark: See annex 2

Gültigkeit / Validity: von / from 2014-05-27, bis / until 2019-05-26

Registrier-Nr. / Registered-No. 44 207 14097601
Prüfbericht Nr. / Test Report No. 3514 2383

Essen, 2014-05-27

Zertifizierungsstelle der TÜV NORD CERT GmbH
Certification body of TÜV NORD CERT GmbH

TÜV NORD CERT GmbH Langemarckstraße 20 45141 Essen www.tuev-nord-cert.de technology@tuev-nord.de

TÜV NORD CERT GmbH Safety Approved

DAkkS Deutsche Akkreditierungsstelle D-ZE-12037-01-00

UNIVERSAL ROBOTS

## 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**

UNIVERSAL ROBOTS

## 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

# 11 Safety standards

UNIVERSAL ROBOTS

## 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**

UNIVERSAL ROBOTS

## Summary

- MANDATORY

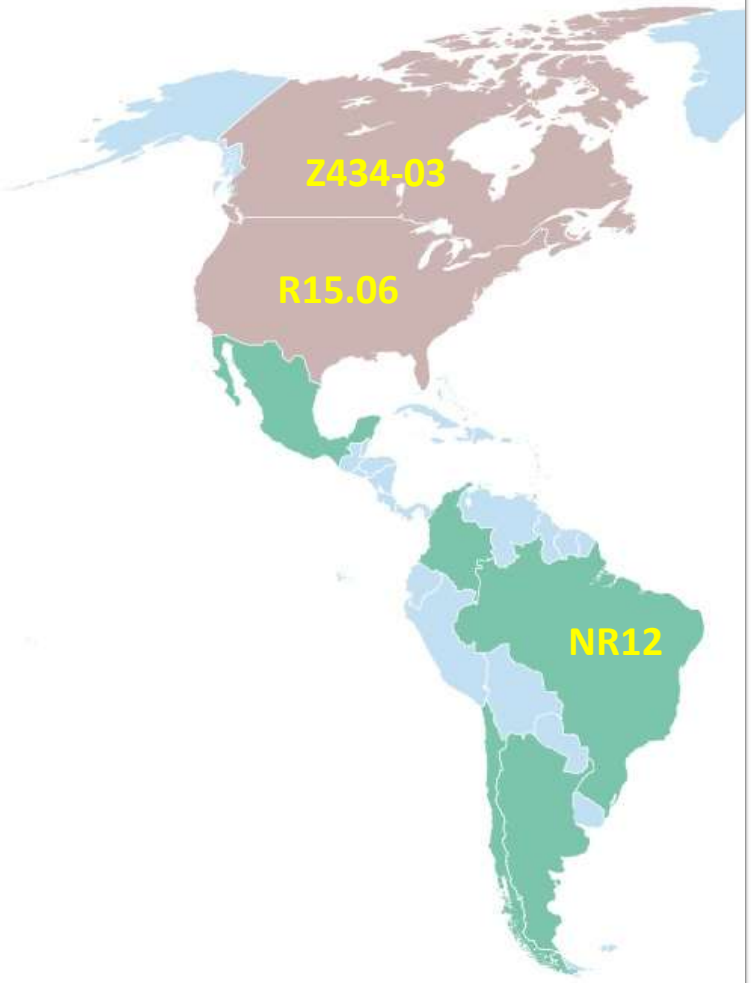
**Integrator *must* perform risk assessment**


- NOT MANDATORY
  - Compliance with the standards
  - *It is recommended to comply with the standards!*
  - In case of failure:
    - System complies with the standards » burden of proof lies with the prosecutor
    - System does *not* comply with the standards » burden of proof lies with the integrator

UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

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

UNIVERSAL ROBOTS

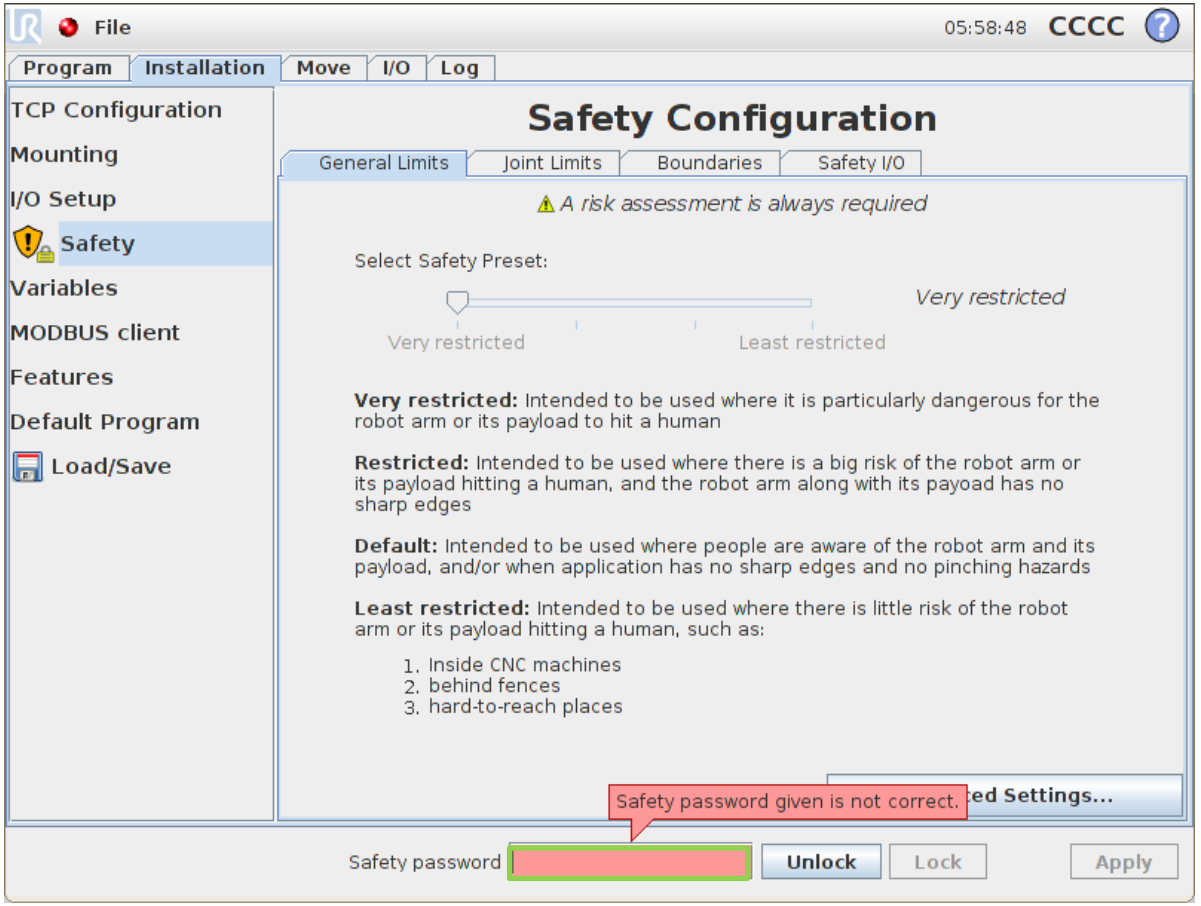
## Features

- 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

UNIVERSAL ROBOTS

# Safety password

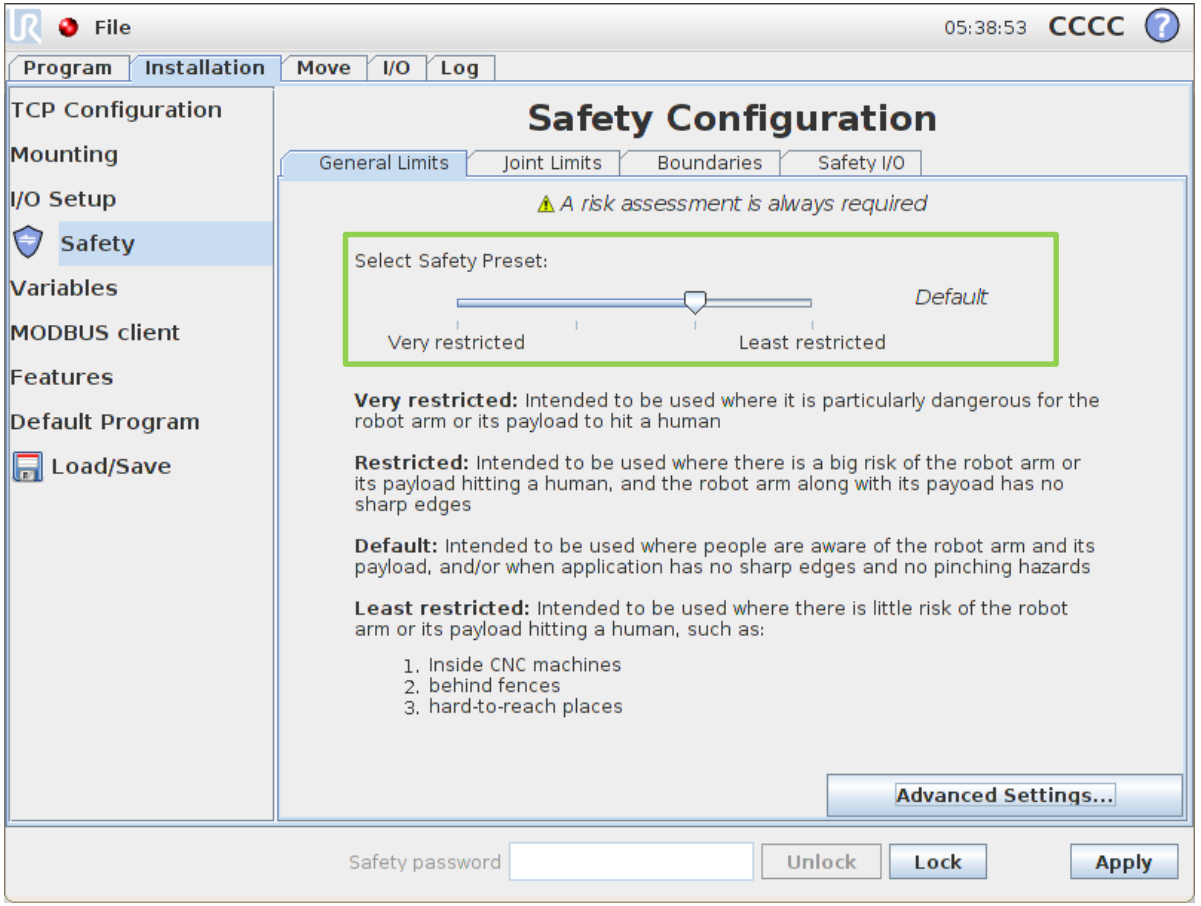
- Safety configuration is password protected
  - Lock
    - Protect safety configuration
  - Unlock
    - Enable modification of safety configuration
- Set safety password
  - Go to Setup robot
  - Select Set Password
  - Enter Password
  - Press Apply



UNIVERSAL ROBOTS

## Basic settings

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



UNIVERSAL ROBOTS

## Basic settings

- Default settings

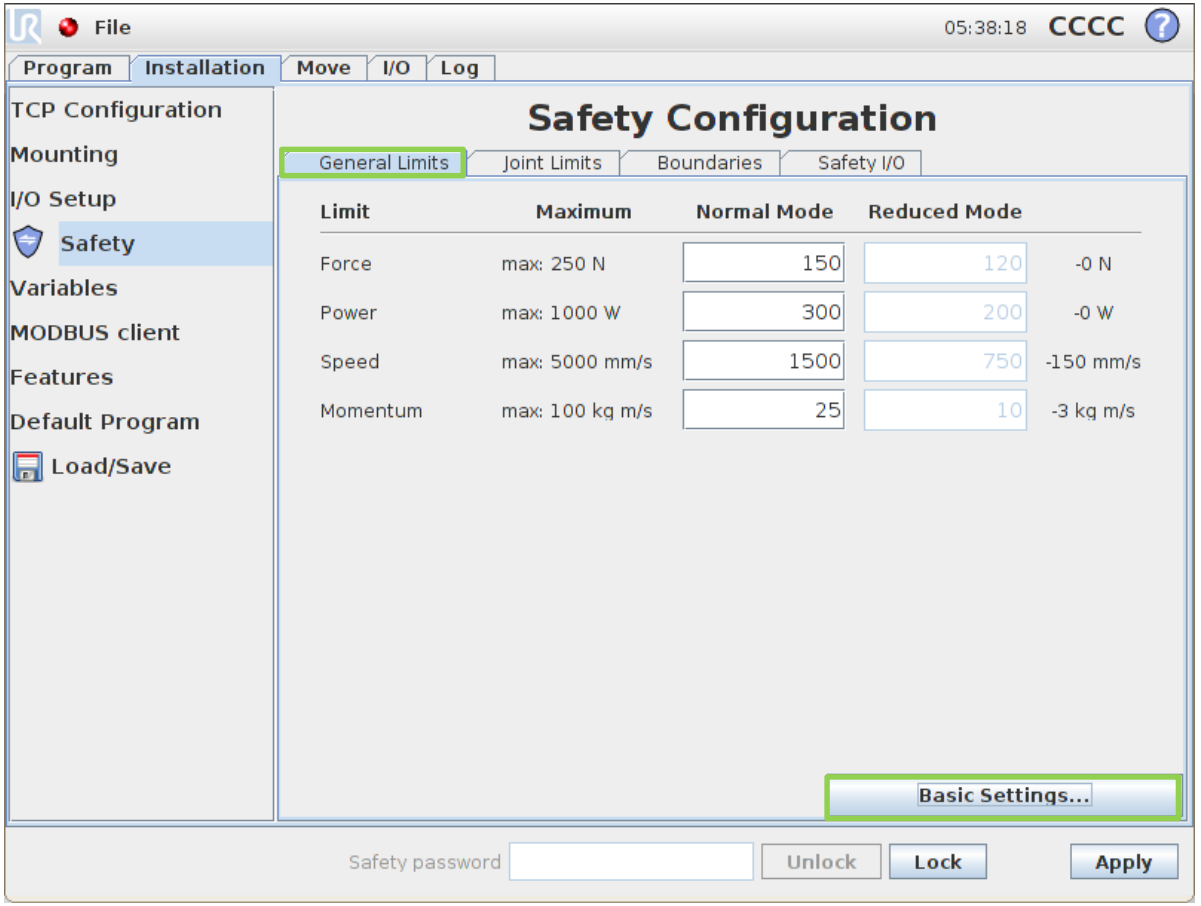
| Mode | Very restricted | Restricted | Default | Least restricted |
|---|---|---|---|---|
| Force (N) | 100 | 120 | 150 | 250 |
| Power (W) | 80 | 200 | 300 | 1000 |
| Speed (mm/s) | 250 | 750 | 1500 | 5000 |
| Momentum (kg·m/s) | 5 | 10 | 25 | 100 |

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

UNIVERSAL ROBOTS

## Advanced settings

- Customize settings
  - Force
  - Power
  - Speed
  - Momentum
- Modes
  - Normal Mode
  - Reduced Mode
    - requires setup of safety input



UNIVERSAL ROBOTS

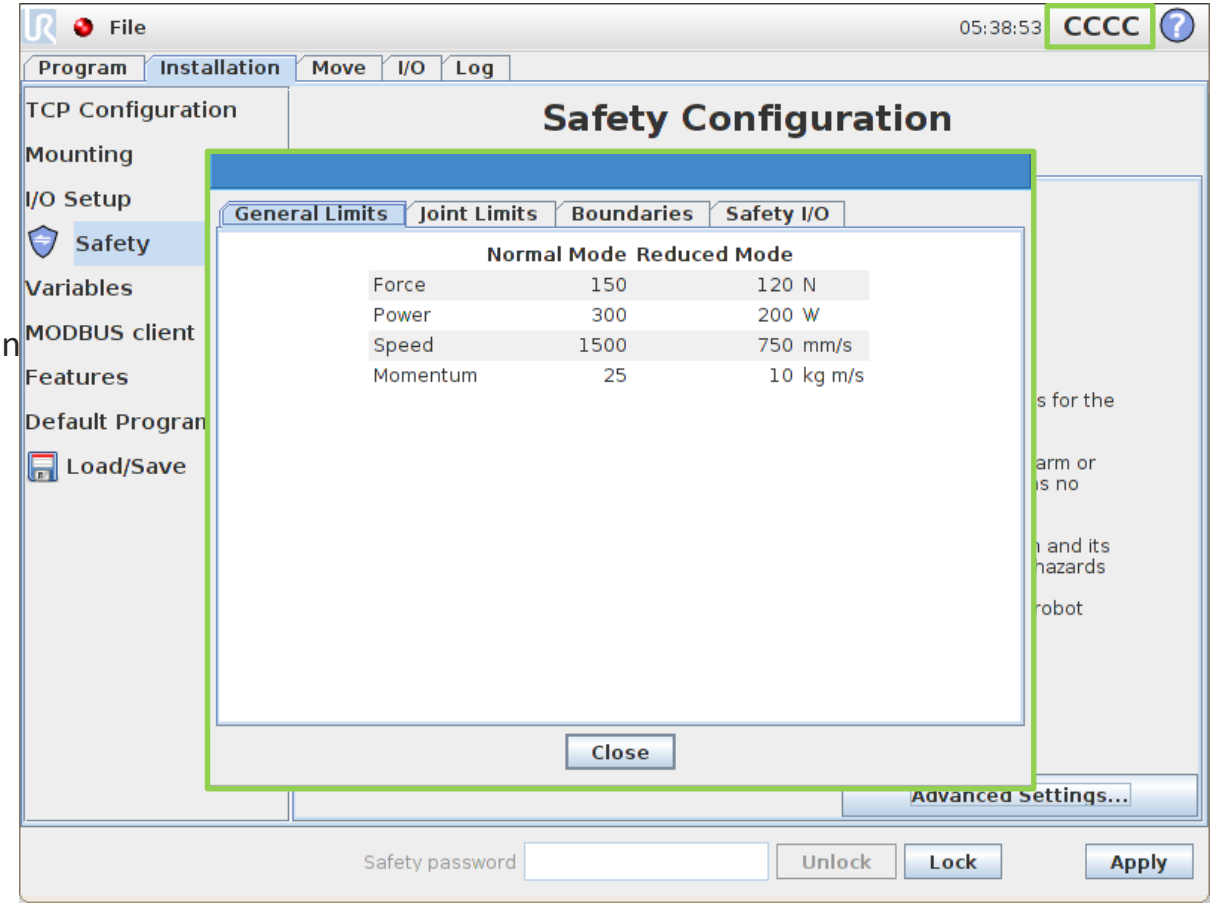
## Safety modes

- 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

UNIVERSAL ROBOTS

## Safety checksum

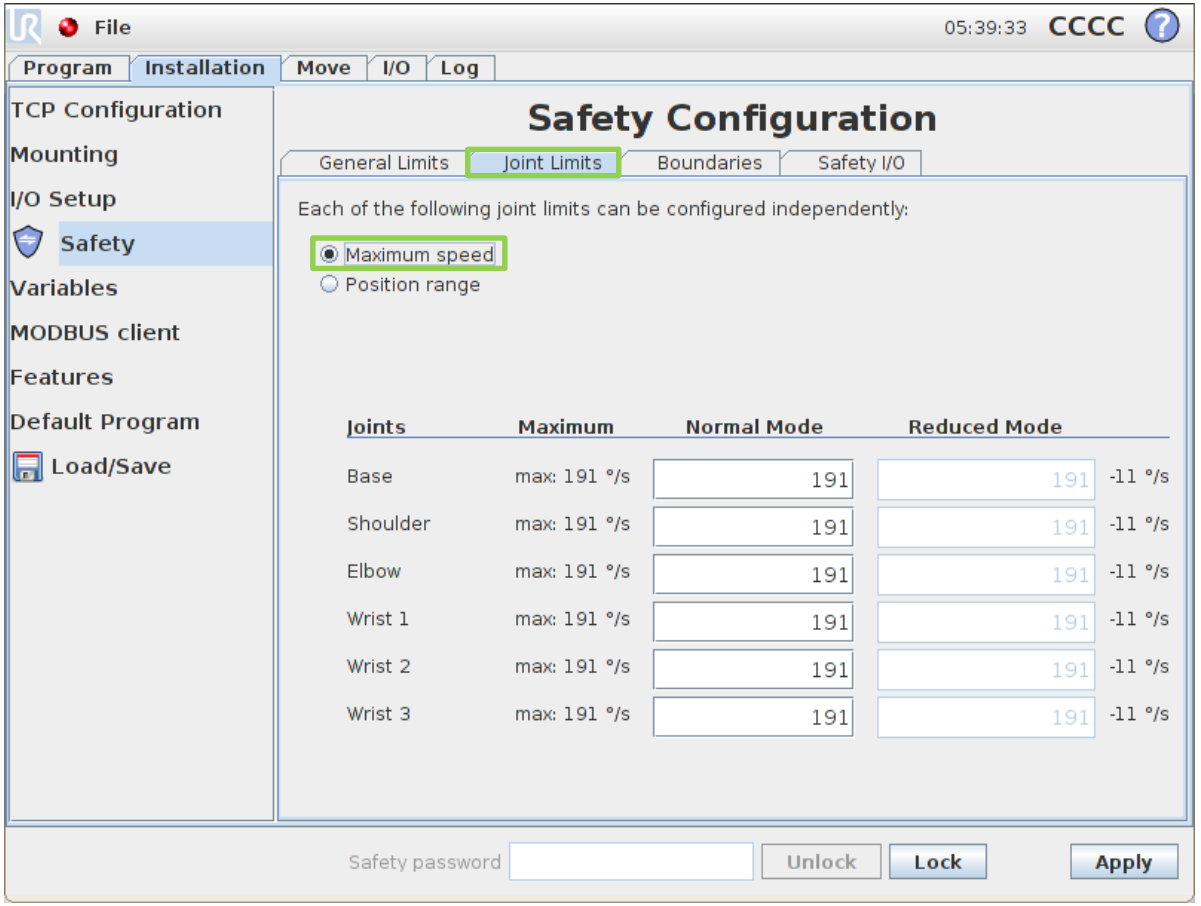
- Checksum
  - Visual indication of safety configuration
  - Indicated as colors and numbers
  - Checksum will change when safety configuration is altered



UNIVERSAL ROBOTS

## Joint limits

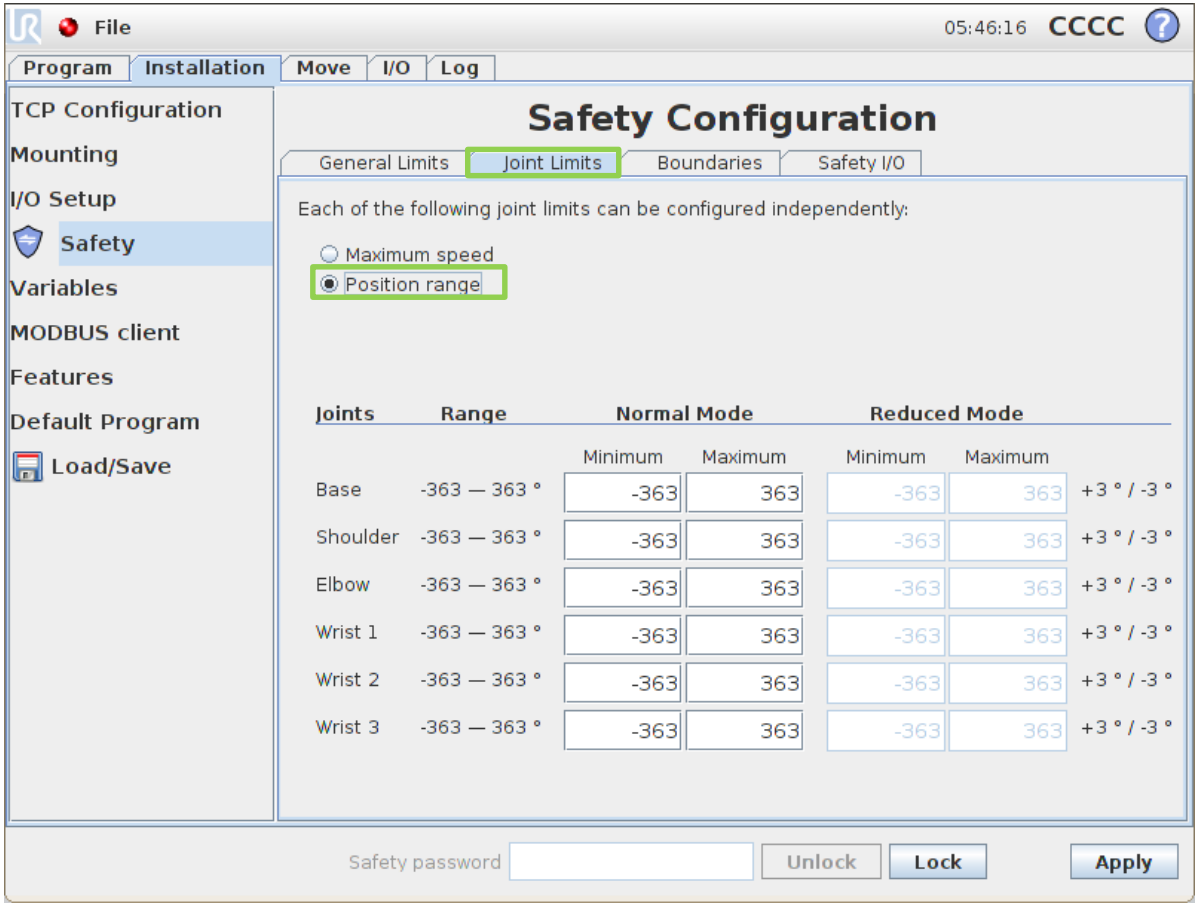
- Maximum speed
  - Set max. speed for each joint
- Modes
  - Normal Mode
  - Reduced Mode
    - requires setup of safety input



UNIVERSAL ROBOTS

## Joint limits

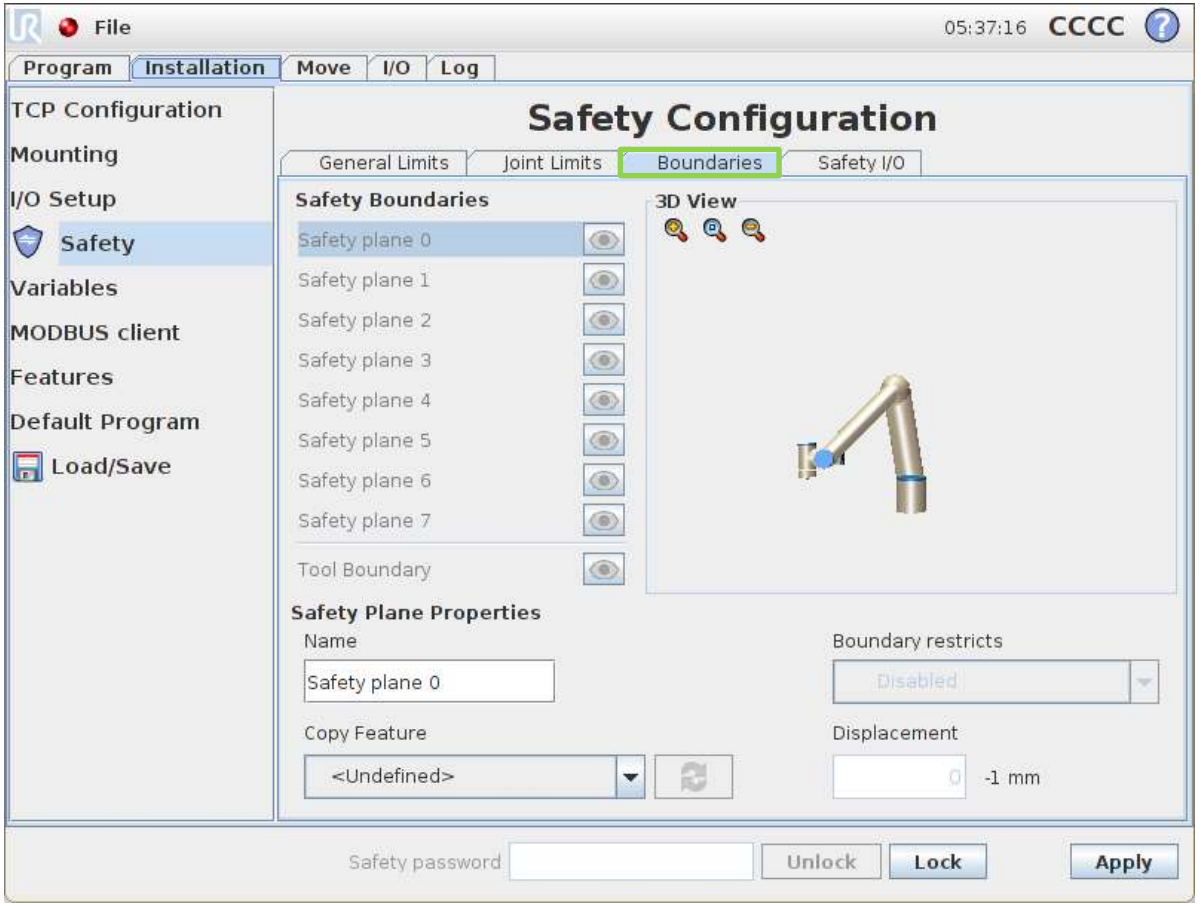
- Position range
  - Set min. and max. range for each joint
- Modes
  - Normal Mode
  - Reduced Mode
    - requires setup of safety input



UNIVERSAL ROBOTS

## 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

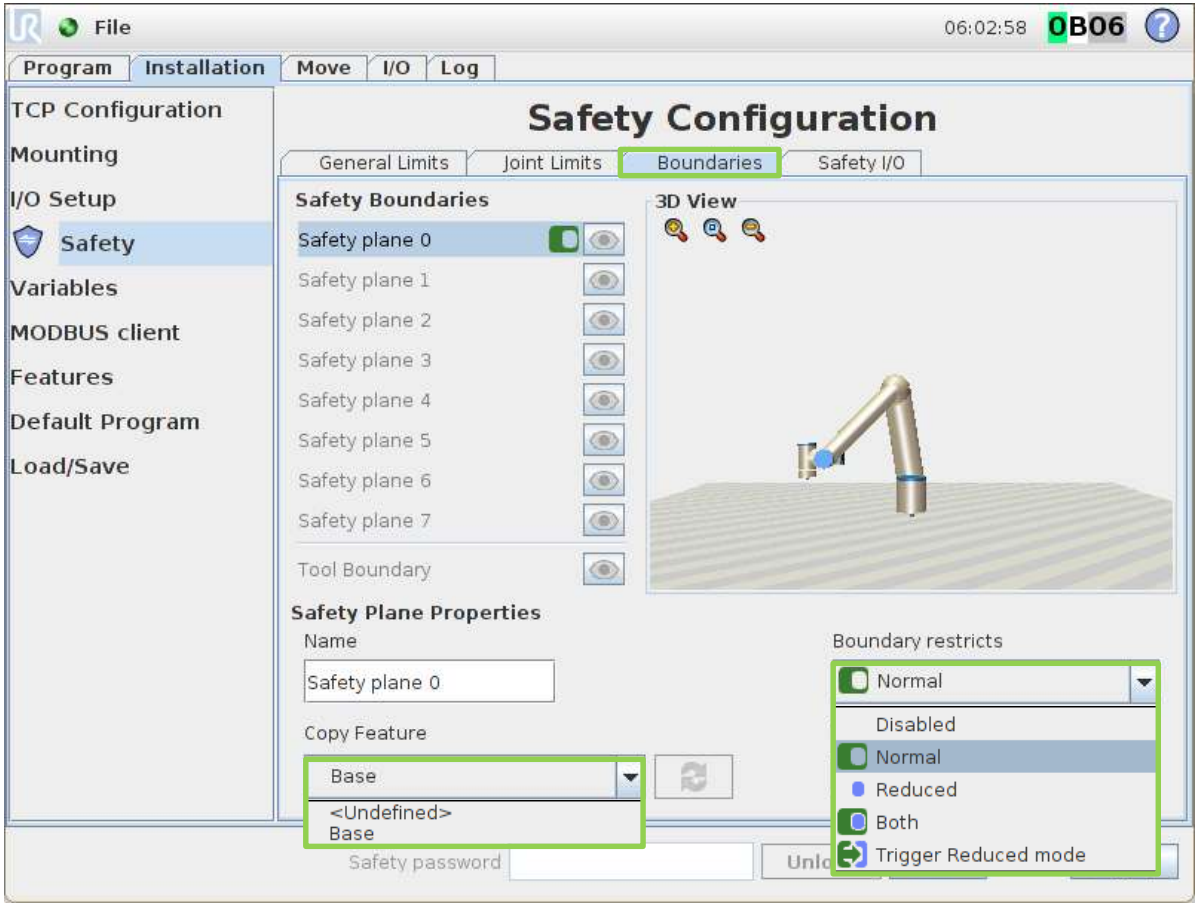



- IMPORTANT: Safety boundary defines a limit only for robot TCP, not overall limit for robot arm

UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

## Safety Modes

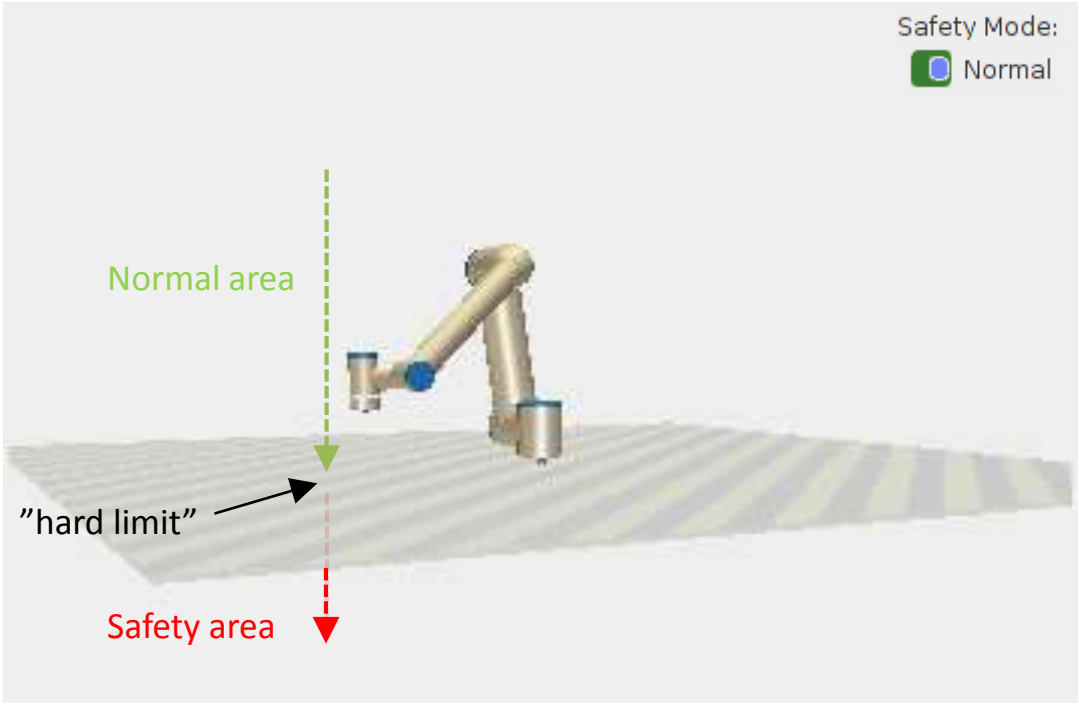
- Safety Modes behaviour

| 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 |

UNIVERSAL ROBOTS

## 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

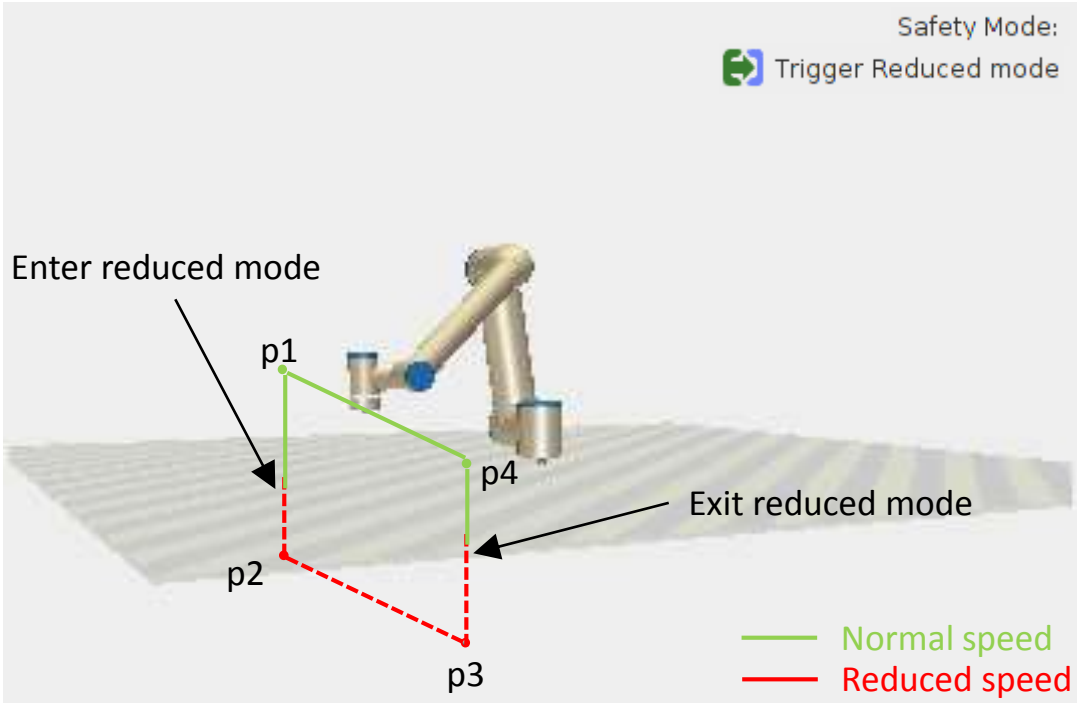



  - Behaviour in Run mode
    - Program execution will be aborted, safety violation will popup

UNIVERSAL ROBOTS

# 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*



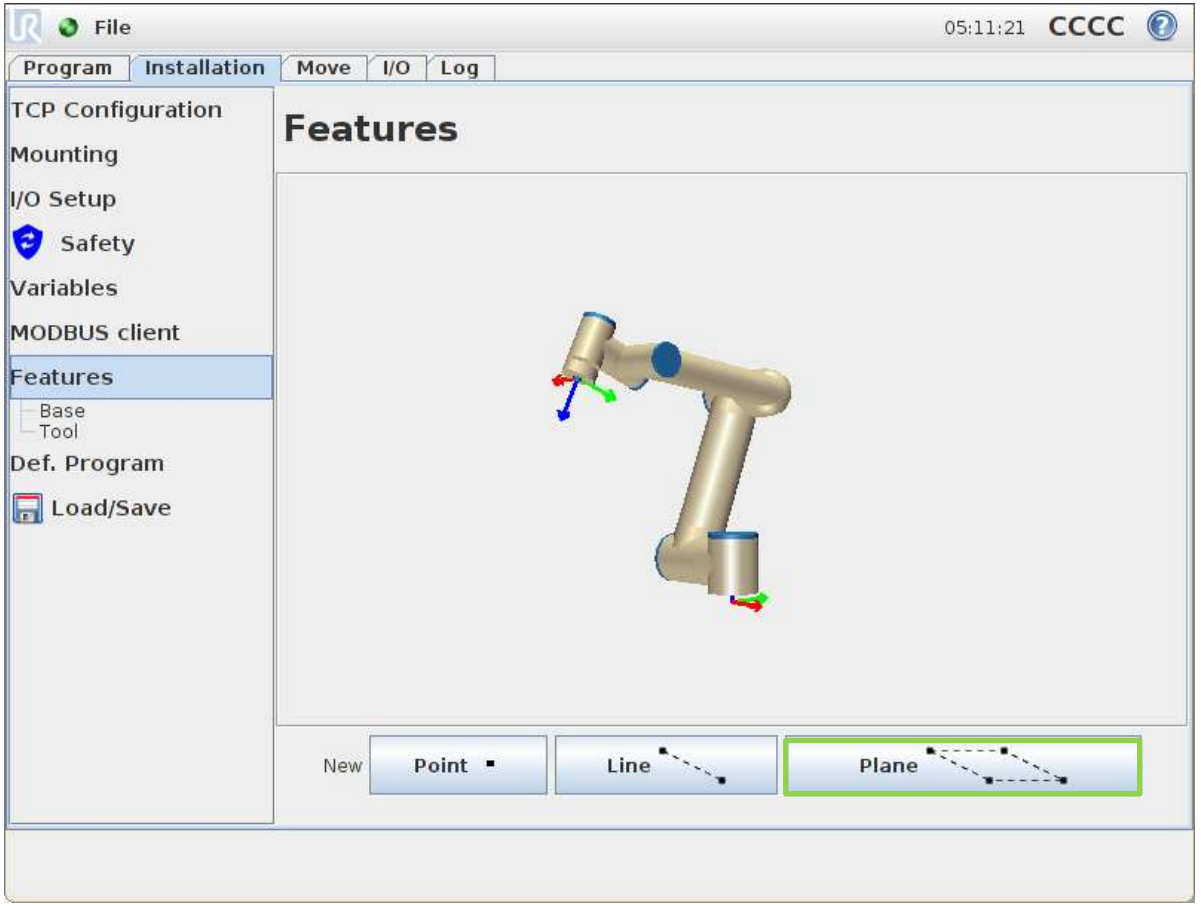

Robot Program
MoveL
Waypoint_1
Waypoint_2
Waypoint_3
Waypoint_4

- Save sample program as trigger_reduced_mode.urp

UNIVERSAL ROBOTS

## User defined safety plane

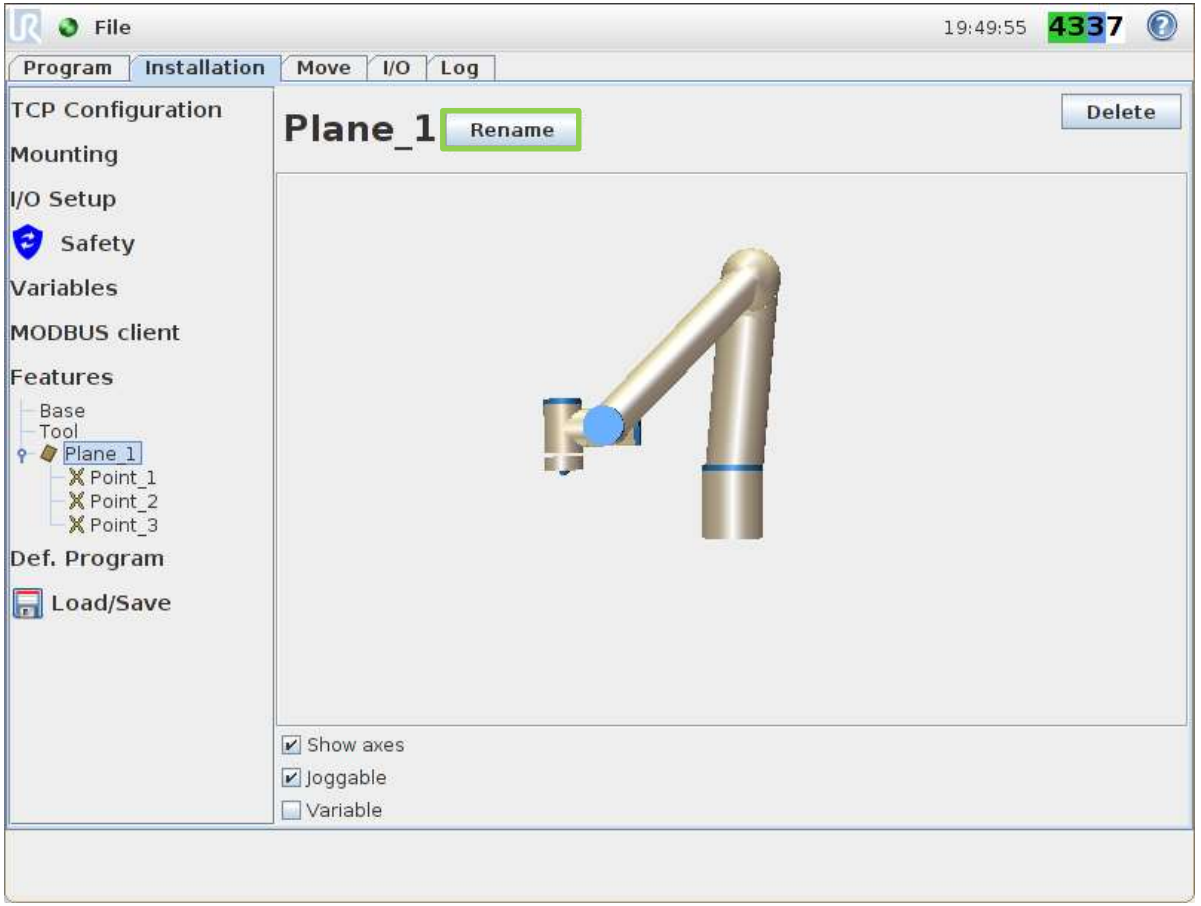
- Features
  - In PolyScope a Plane is defined as a Feature
  - Multiple Features can be set
  - Set Feature as
    - Point
    - Line
    - Plane
- Add Feature
  - Plane



UNIVERSAL ROBOTS

## User defined safety plane

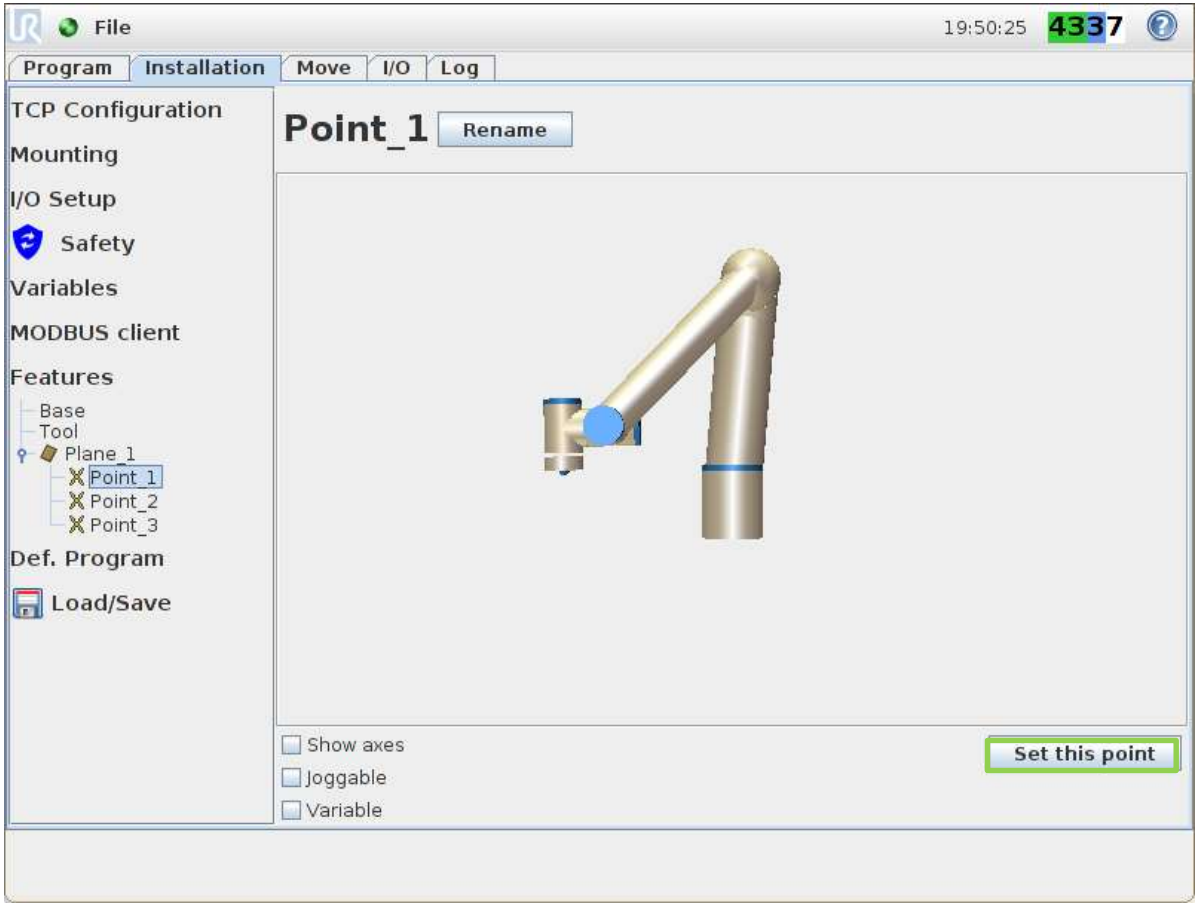
- Options
  - Rename Feature
  - Delete Feature
- Parameters
  - Show axes
  - Joggable
  - Variable



UNIVERSAL ROBOTS

## User defined safety plane

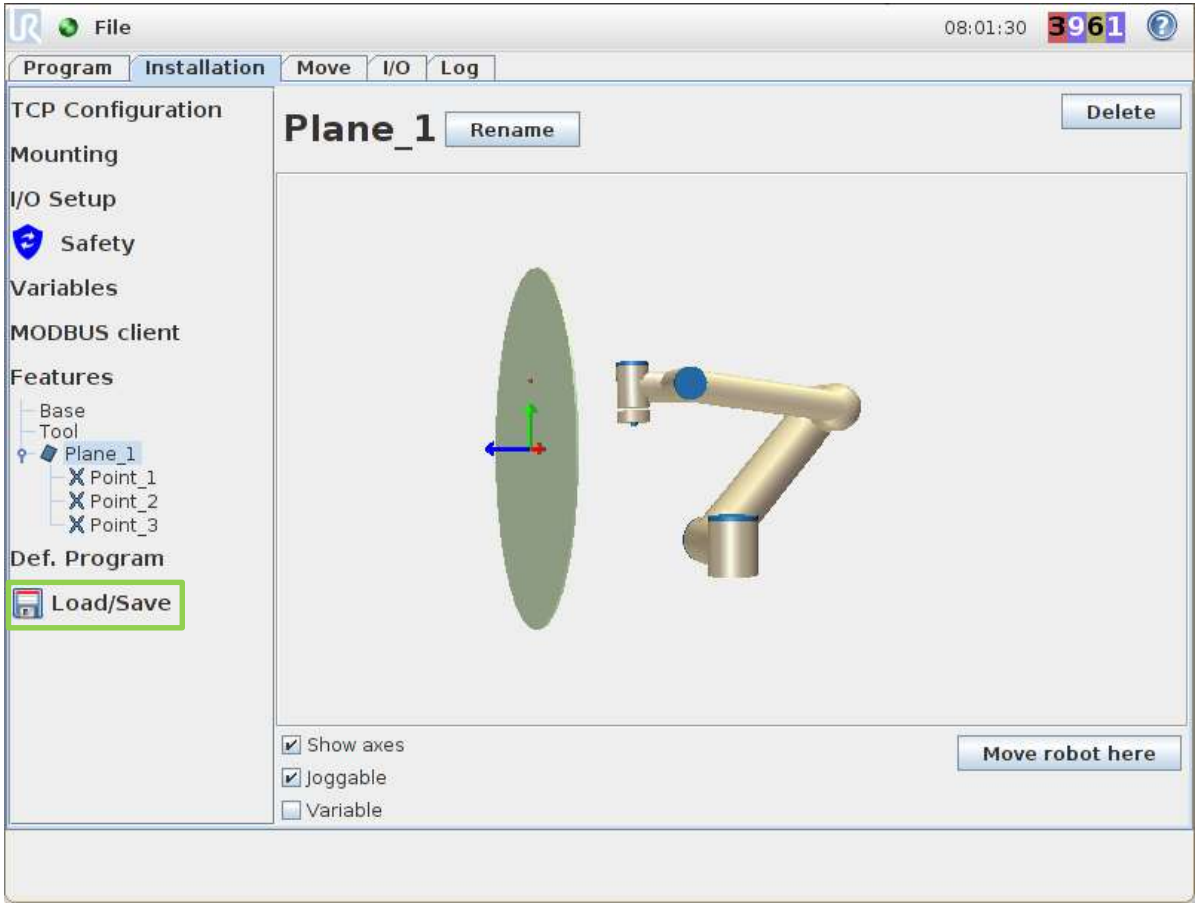
- Setup Plane
  - Plane consists of three fixed waypoints
- Teach vertical plane
  - Point_1 = origin
  - Point_2 = Y-direction
  - Point_3 = X-direction



UNIVERSAL ROBOTS

## User defined safety plane

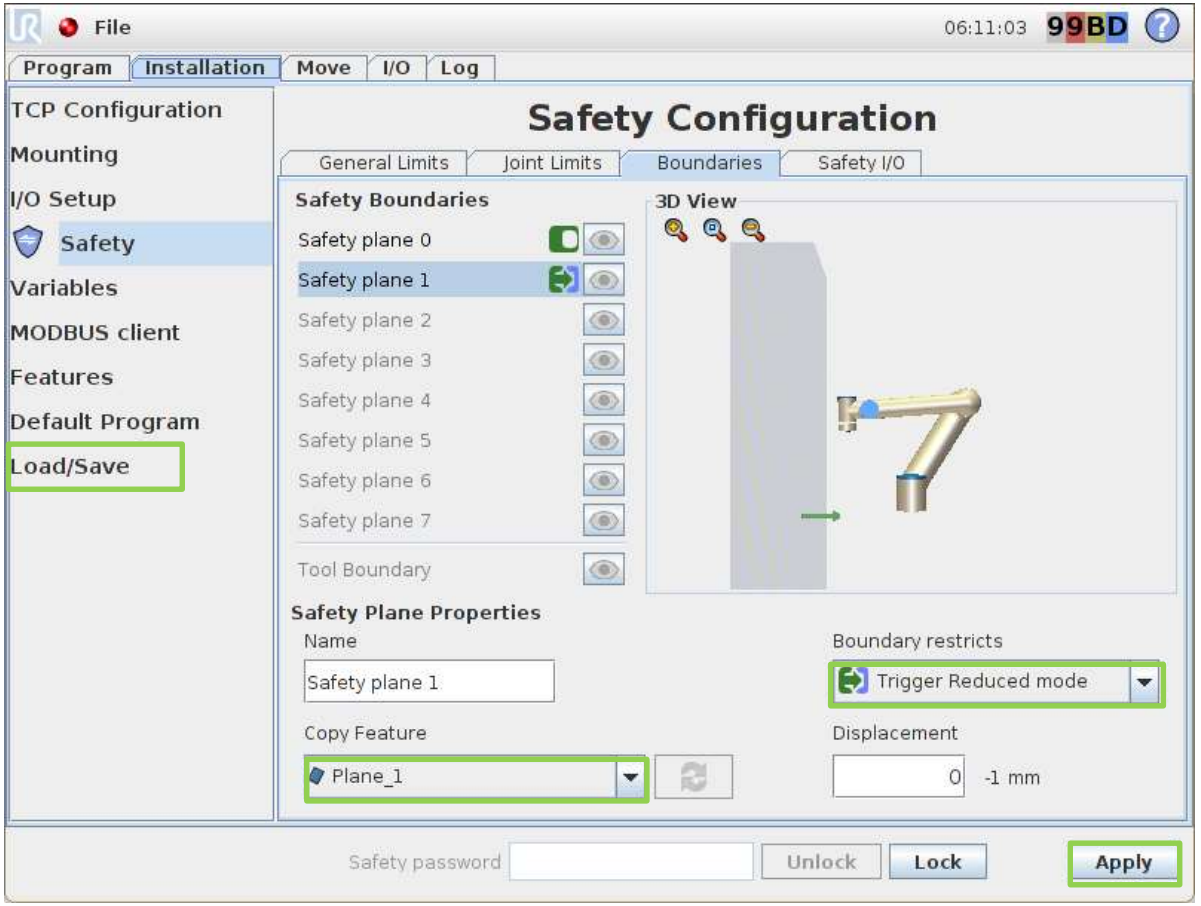
- Graphical illustration of taught Plane
  - Save Installation



UNIVERSAL ROBOTS

## User defined safety plane

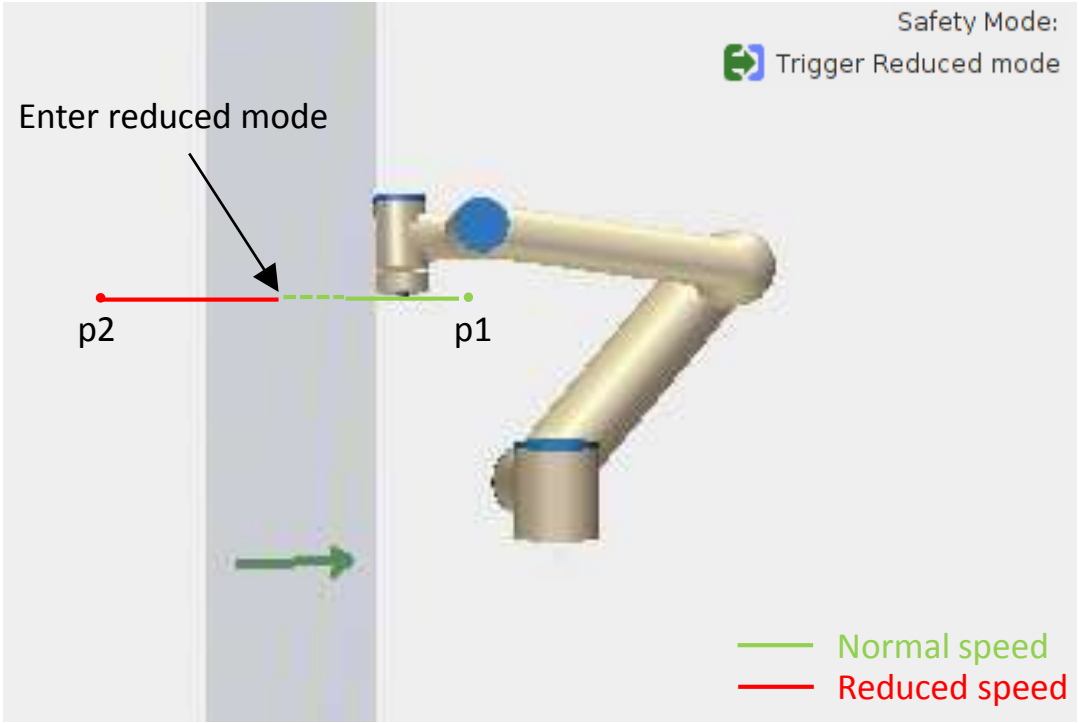
- Setup new safety plane
  - Set Feature plane to: *Plane_1*
  - Set Safety Mode to: *Trigger Reduced Mode*
  - Apply settings
  - Save as *safety.installation*



UNIVERSAL ROBOTS

# 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



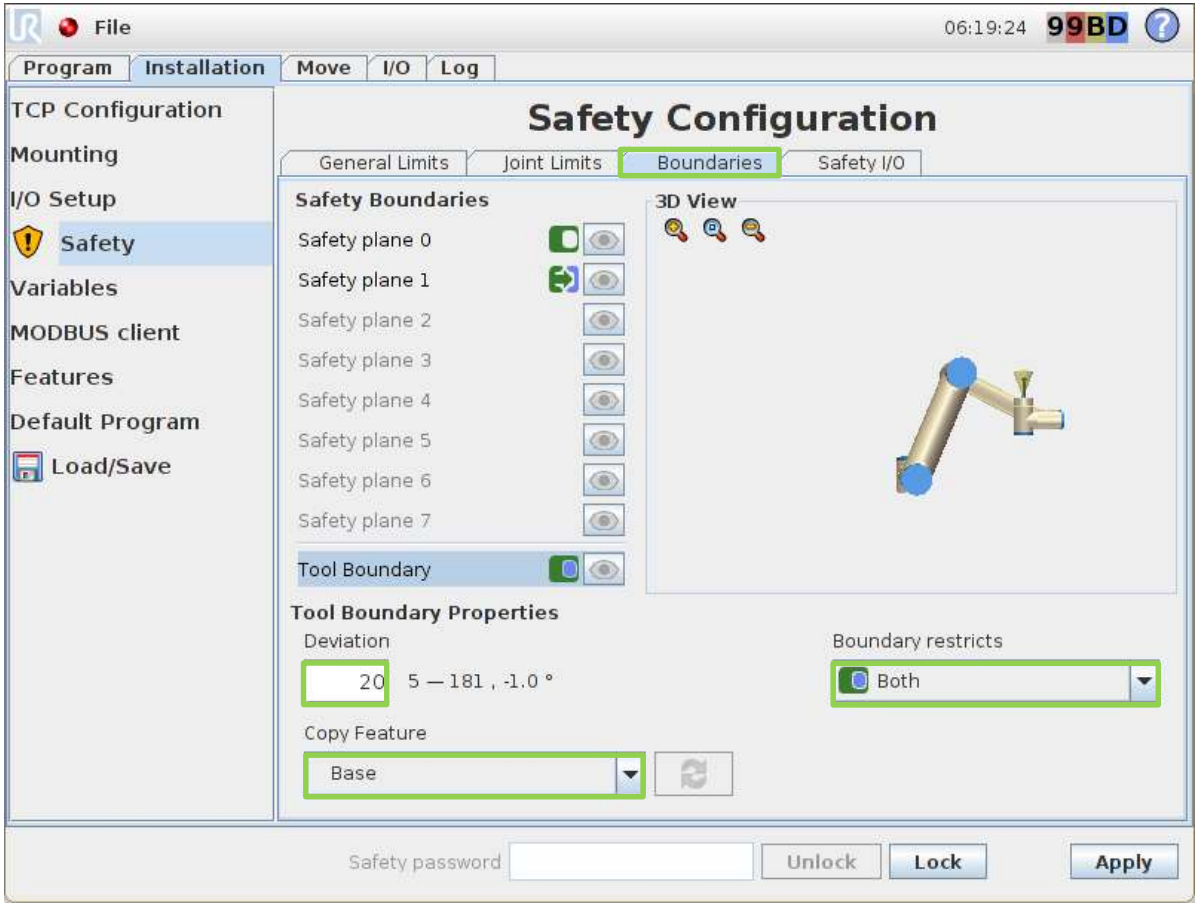

Robot Program
MoveL
Waypoint_1
Waypoint_2

- Save sample program as trigger_reduced_mode_feature.urp

UNIVERSAL ROBOTS

## Tool boundary

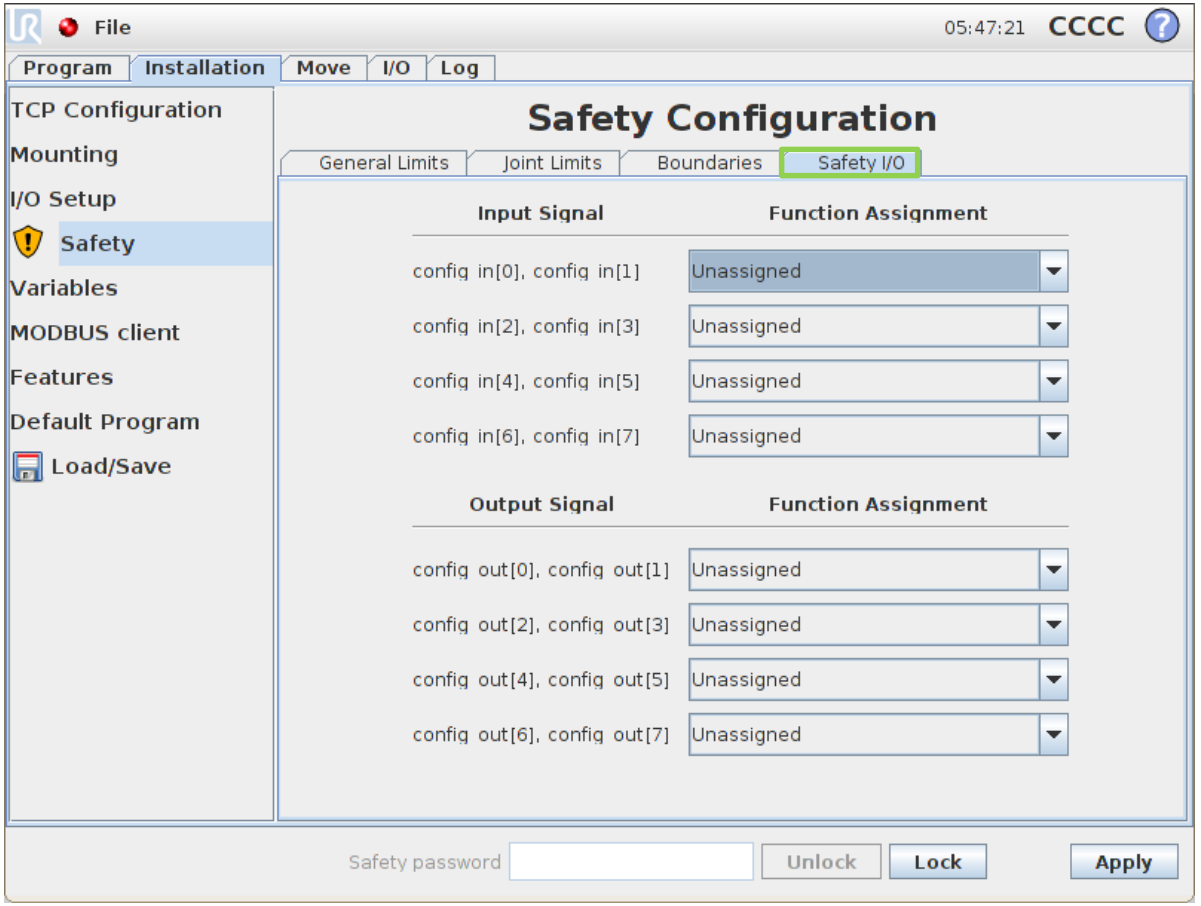
- Restrict angular deviation of TCP
  - Set max. TCP deviation in respect to selected Feature
- Test
  - Set Feature plane: *Base*
  - Set Deviation: *25°*
  - Set Safety Mode to: *Normal & Reduced Mode*
  - Apply
  - Test in Teach mode



UNIVERSAL ROBOTS

## Safety I/O

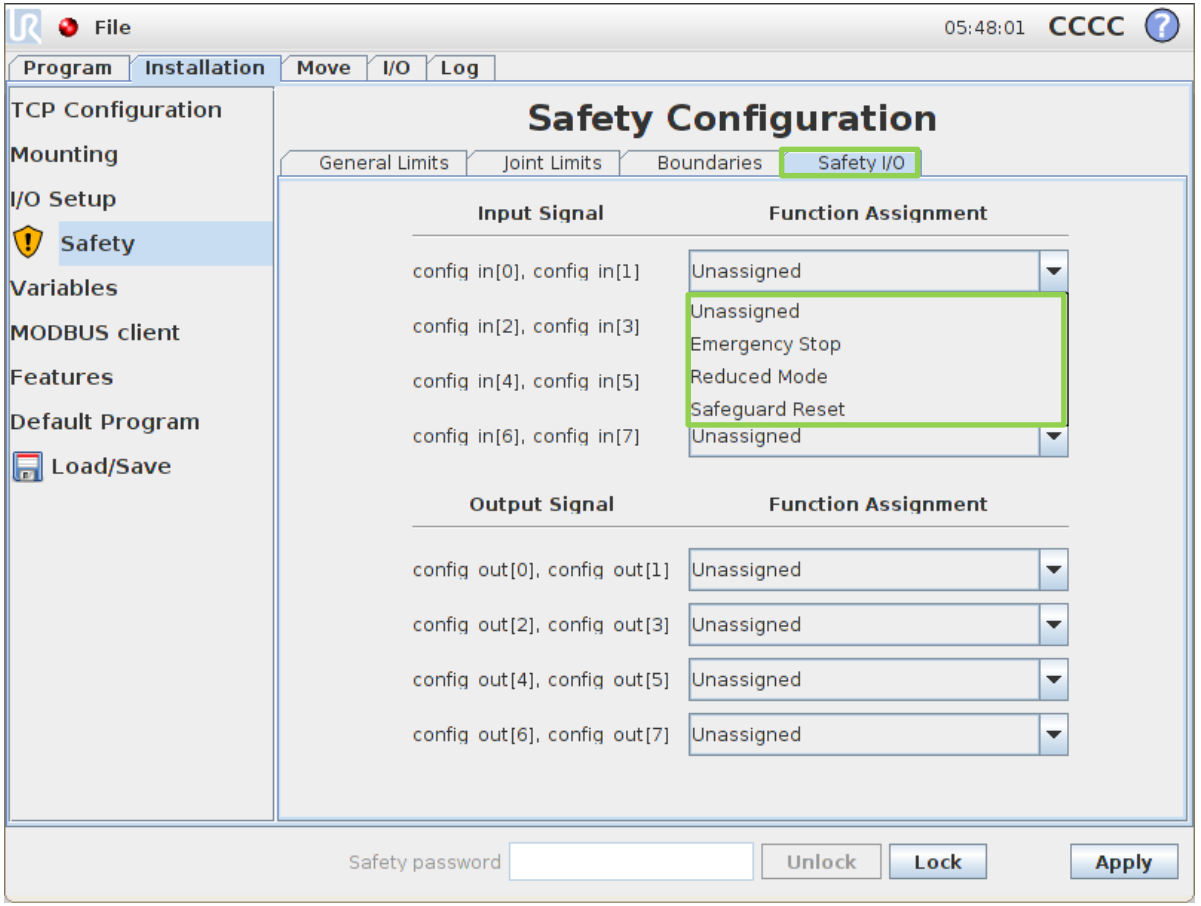
- Safety functions
  - Safety functions can be assigned to the Configurable I/O's
  - All functions are redundant
    - Two signals for each function
    - Category 3, PLd
  - Configurable I/O's
    - Digital inputs
    - Digital outputs



UNIVERSAL ROBOTS

# Safety inputs

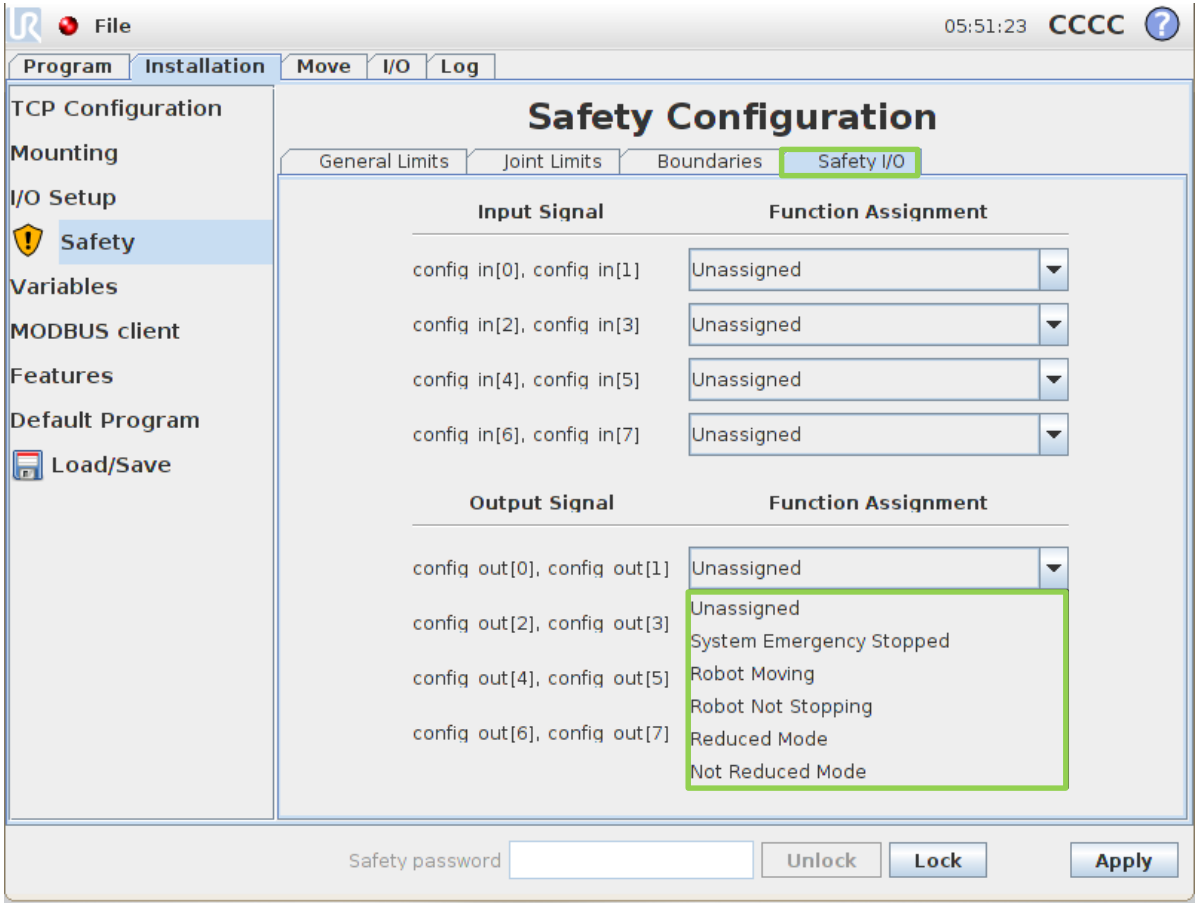
- Safety inputs
  - Emergency stop
    - For connecting external EMG-button or safety PLC
  - Reduced Mode
    - Low signal: robot operates in normal mode
    - High signal: robot operates in reduced mode
  - Safeguard Reset
    - If Safeguard is hardwired to Safety Control Board, the safeguard can be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



UNIVERSAL ROBOTS

## 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



UNIVERSAL ROBOTS

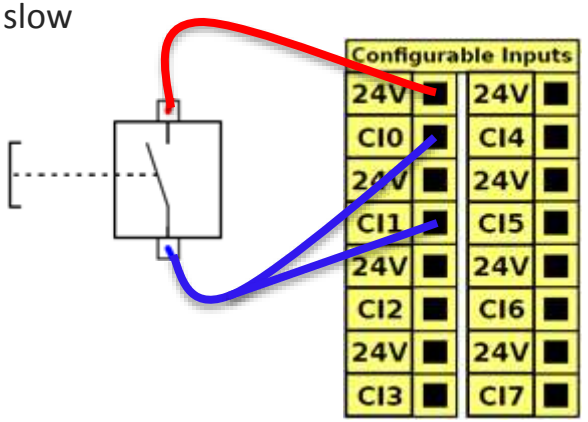
## Lab exercise part 1

- 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=400mm 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

UNIVERSAL ROBOTS

## 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.



UNIVERSAL ROBOTS

## 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!

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# 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

UNIVERSAL ROBOTS

# 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