Linrob URCap Manual - ENG

Inhalt

Author: Karim Zanaty

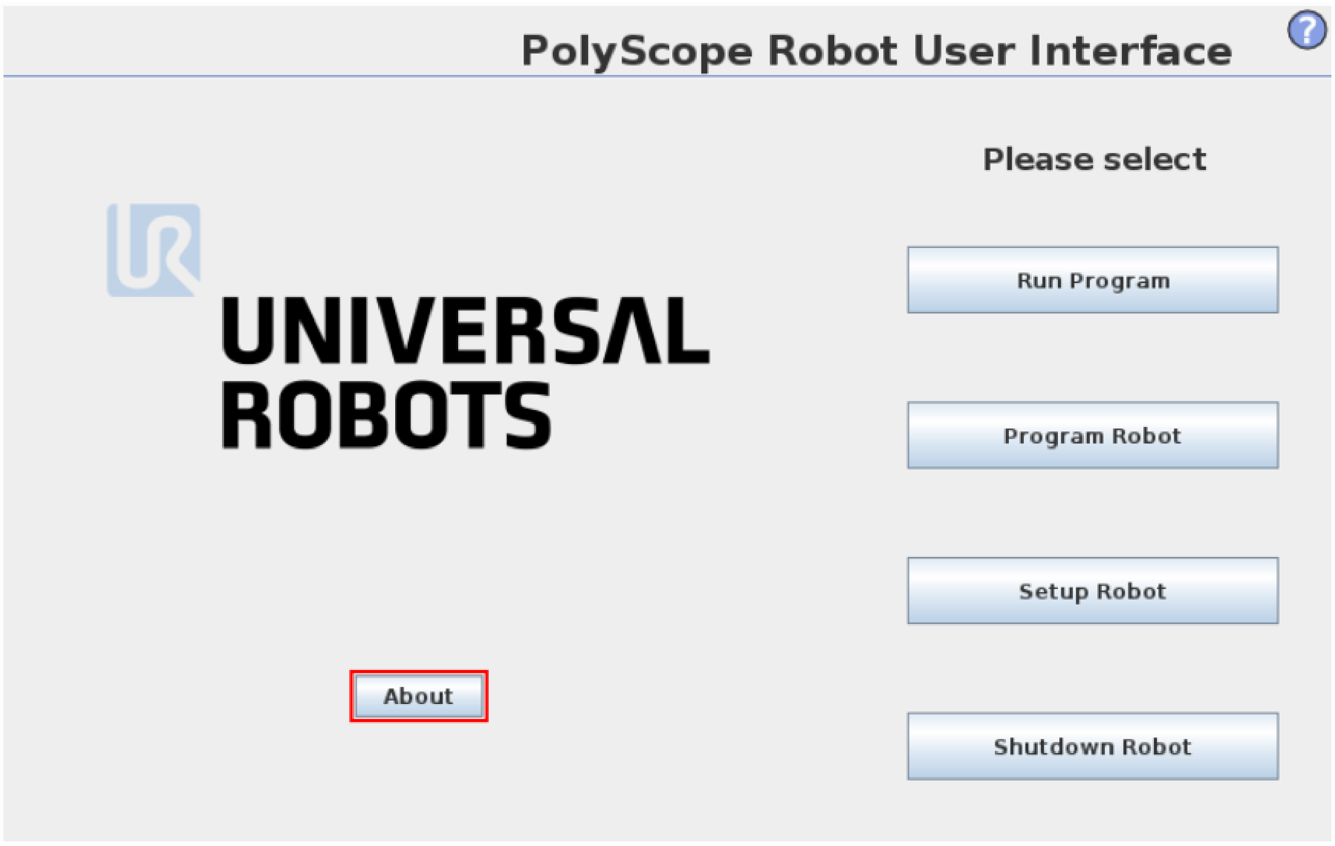
Version: v1.4.4_dev_b

1. General information

The linrob URCap is working with *Polyscope 5.0.x* and higher.

The linrob URCap does not work with the latest, *Polyscope X*, yet.

To check the Polyscope version currently installed in your robot, click the About button on the home screen.



2. Connect linrob with UR

2.1. Get the latest version

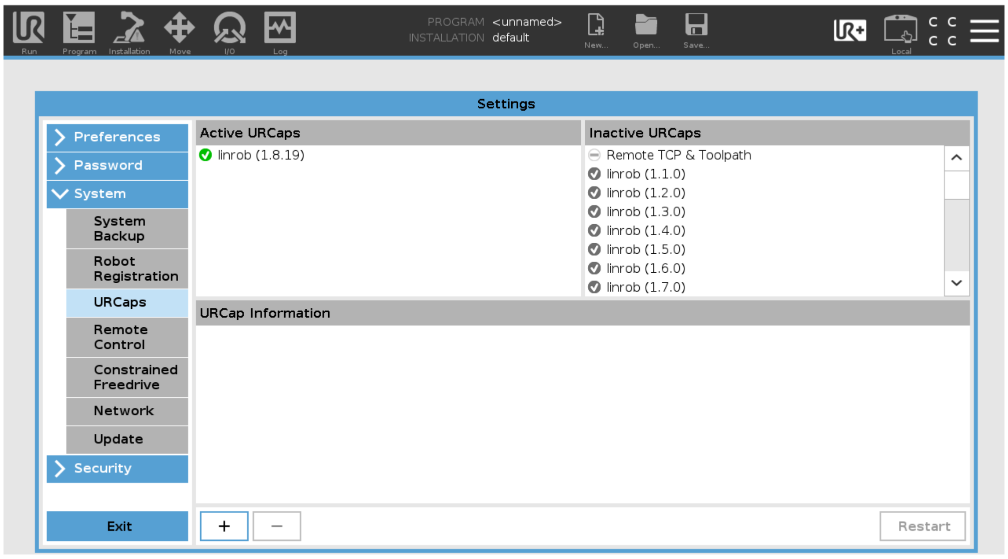
Visit https://linrob.io/produkte/ur and get the latest up-to-date manual. Download the corresponding *linrob-x.x.x.urcap* and save it on an USB stick.

2.2. Install software

After you Get the latest version, insert the USB stick into the teach pendant of the robot.

Tap the *triple bar* icon in the upper right corner and select the *System* menu on the left.

Tap the URCaps submenu and then tap the "+" sign at the bottom of the screen. Open *linrob-x.x.x.urcap* and use the *Restart* button to activate the URCap.

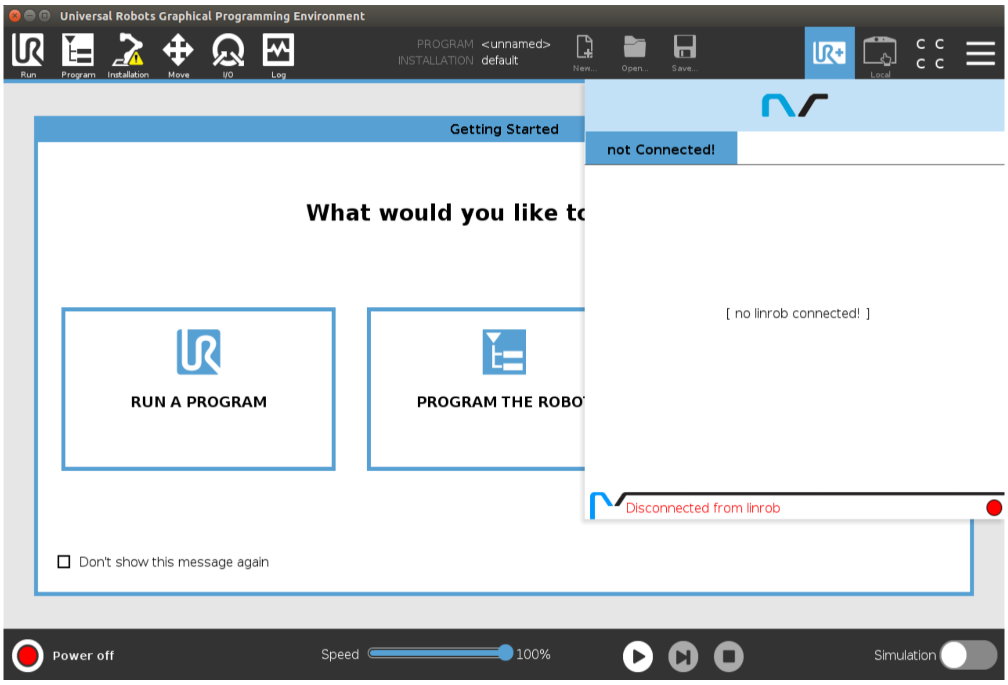


Active URCaps	Inactive URCaps
linrob (1.8.19)	Remote TCP & Toolpath
	linrob (1.1.0)
	linrob (1.2.0)
	linrob (1.3.0)
	linrob (1.4.0)
	linrob (1.5.0)
	linrob (1.6.0)
	linrob (1.7.0)

URCap Information

2.3. Activate linrob

If no linrob device is connected to the UR, the following screen will indicate that.

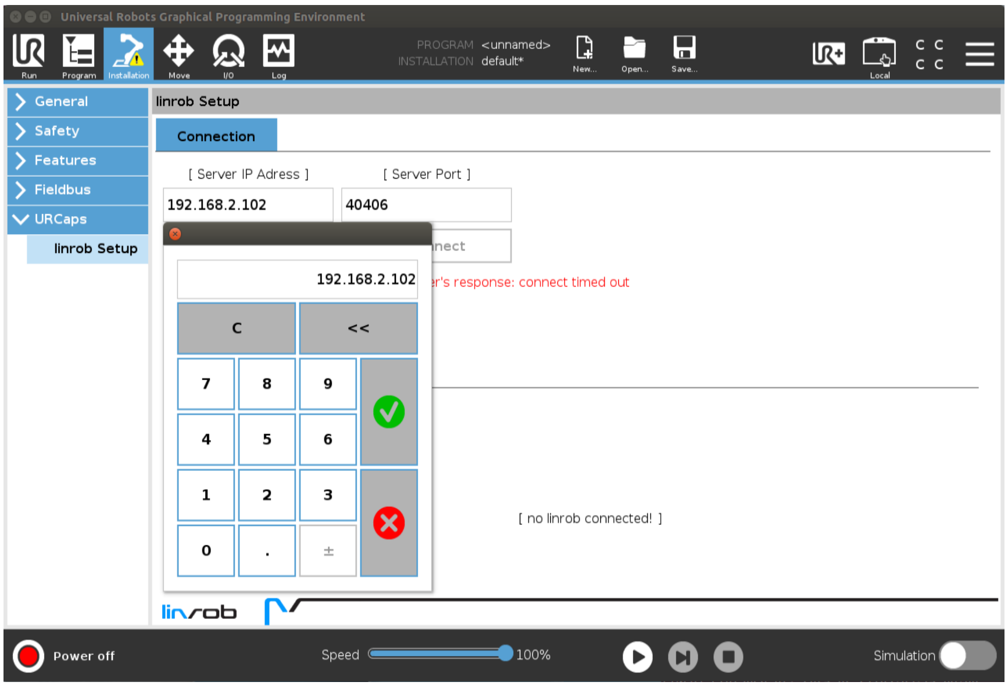


[no linrob connected!]

Disconnected from linrob

Therefore, tap the *Installation* button at the top of the left side. Then, tap the *URCaps* button in the navigation panel on the left side. Select “*linrob Setup*” in the dropdown list.

A tab will show for setting the server address and server port.



linrob Setup

[Server IP Adress] 192.168.2.102

[Server Port] 40406

192.168.2.102

r's response: connect timed out

[no linrob connected!]

Linrob port is always : 40400

The IP address is usually 192.168.1.1

Congratulations, now you are connected to the Linrob.

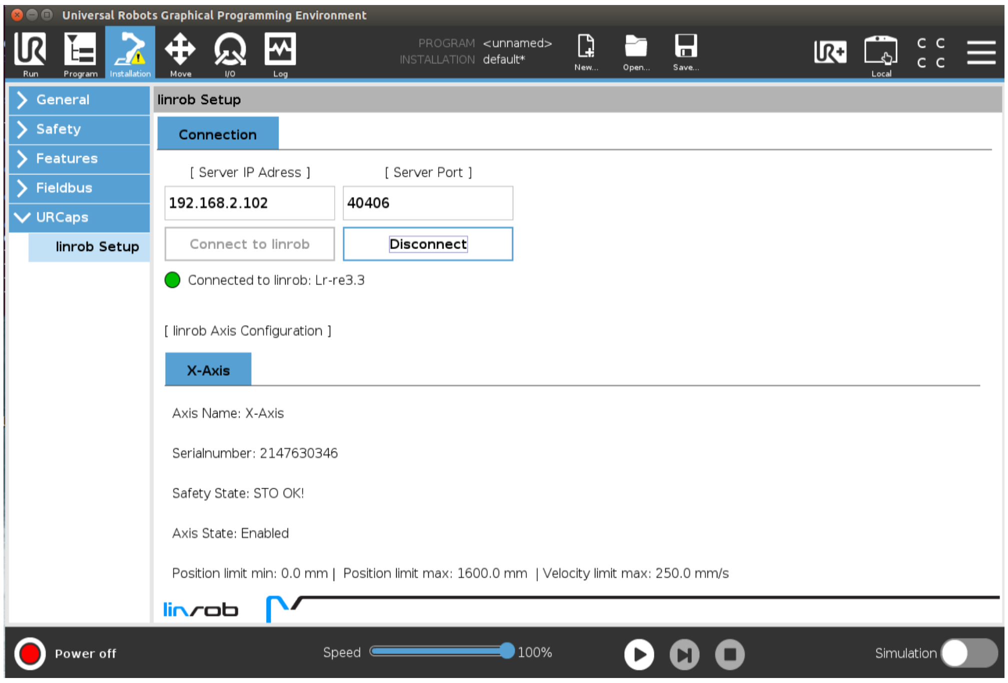
.

3. The linrob URCap interface

linrob axes integrate seamlessly with Universal Robots by means of a URCap plugin. This plugin exposes a set of linrob specific command blocks to control and operate the connected axes. This section documents the interface of the URCap plugin.

3.1. Interface

The interface, as shown in the image below, provides a user-friendly environment for managing your Linrob setup.



PROGRAM <unnamed>
INSTALLATION default*

linrob Setup

Connection

[Server IP Adress] 192.168.2.102
[Server Port] 40406

Connect to linrob | Disconnect

● Connected to linrob: Lr-re3.3

[linrob Axis Configuration]

X-Axis

Axis Name: X-Axis

Serialnumber: 2147630346

Safety State: STO OK!

Axis State: Enabled

Position limit min: 0.0 mm | Position limit max: 1600.0 mm | Velocity limit max: 250.0 mm/s

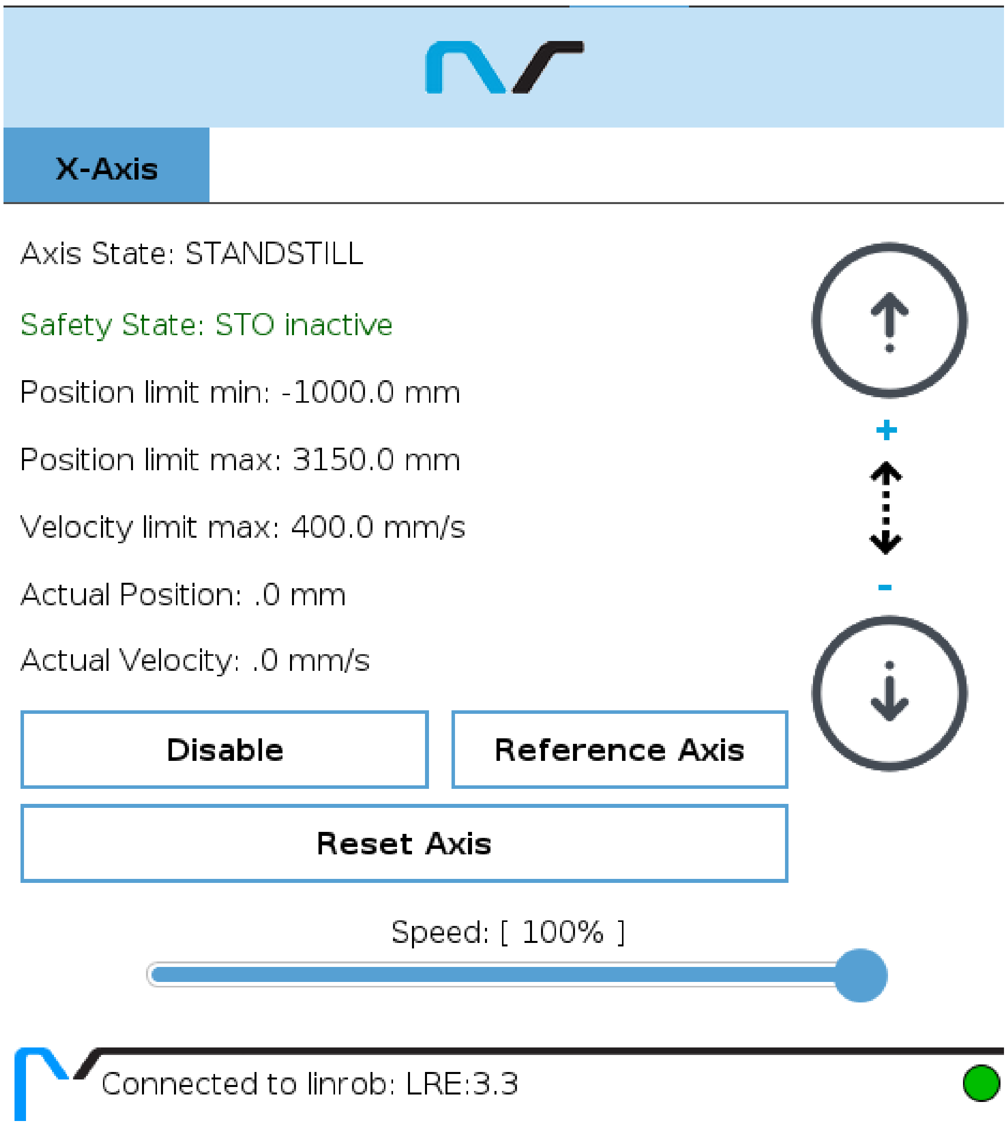
The Linrob Setup interface is divided into several sections:

Connection

This section displays the current connection status. Once connected, it shows the server IP address and port number used for the connection, as well as buttons to either connect to or disconnect from the Linrob.

Linrob Axis Configuration

Below the connection status, you'll find the `Linrob Axis Configuration` section. It displays the axis' current data, safety, and operational states, including position and velocity limits.



Axis State: STANDSTILL

Safety State: STO inactive

Position limit min: -1000.0 mm

Position limit max: 3150.0 mm

Velocity limit max: 400.0 mm/s

Actual Position: .0 mm

Actual Velocity: .0 mm/s

Speed: [100%]

Connected to linrob: LRE:3.3

The "UR+" toolbar, displayed in the top right corner, is a quick access toolbar, provides jog controls, an enable/disable toggle, referencing and reset, and displays current axis data for efficient operation.

The Speed Slider, located beneath the reset button, is exclusively applicable during jogging.

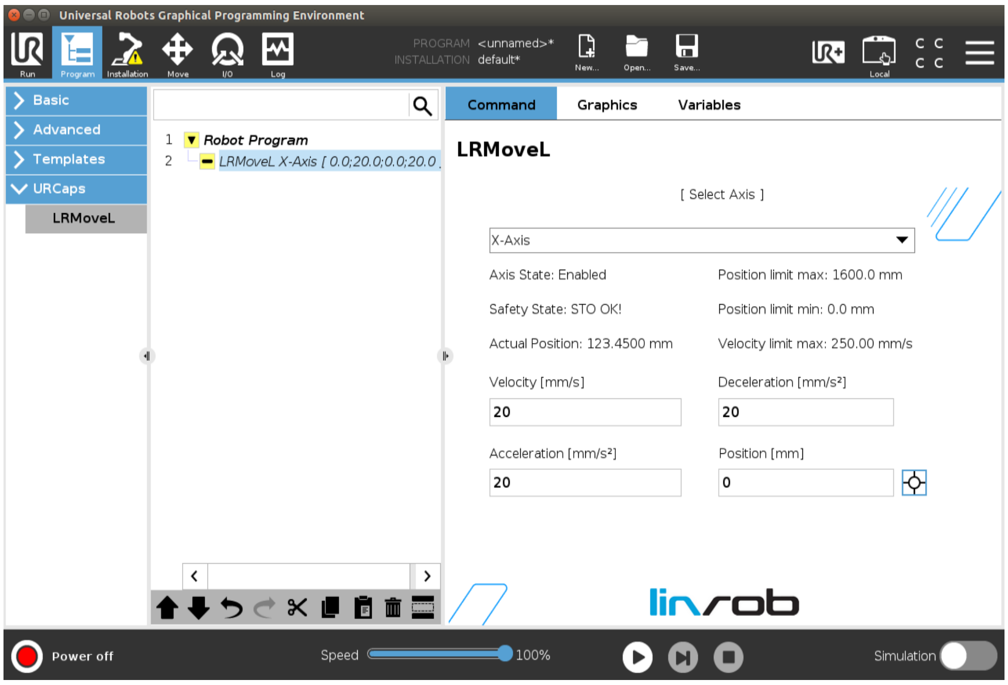
The maximum velocity is 250 mm/s.

In this version, modification of the scale factor is not available to users. To request changes, please contact us.

3.2. Use Nodes to program the linrob

Tap the Program button on the left side.

- Tap the URCaps button.
- The only available command is LRMoveL. Insert the command as often as you need to open a script for each desired movement.



1 Robot Program
2 LRMoveL X-Axis [0.0;20.0;0.0;20.0

LRMoveL

[Select Axis]

X-Axis

Axis State: Enabled
Position limit max: 1600.0 mm
Safety State: STO OK!
Position limit min: 0.0 mm
Actual Position: 123.4500 mm
Velocity limit max: 250.00 mm/s

Velocity [mm/s]: 20
Deceleration [mm/s²]: 20
Acceleration [mm/s²]: 20
Position [mm]: 0

linrob

You have to name each script after one axis.

The box next to the script is yellow if no position and/or speed is registered.

If every position/speed is valid, the box color turns into white.

Now you can move the linrob to your desired position once you start the program.

```
The permissible ranges differ for each application and depend
on the limits that are either defined based on test results or customer
requirements. The units of the parameters are specified in:
Acceleration: mm/s²
Deceleration: mm/s²
Velocity: mm/s
Position: mm
```

3.3. Commands

The following button commands are fixed implementation as described below:

3.3.1. Enable and Disable

The Enable and Disable button works depending on the state of the axis. Only in disabled states the axis will be enabled. Only in enabled states the axis will be disabled.

The functions Disable and Enable are called in the background.

3.3.2. Reference

An unreferenced axis must be referenced. Otherwise, the axis will not accept any sent motion commands.

The axis will move with a fixed slow velocity into the negative direction. This happens until a rising edge signal from the inductive sensor is detected, setting the reached position to 0. Subsequently, the axis will move to the position 100.00 mm.

Please do not interact with the axis during this process.

3.3.3. Reset

The reset must be set in two scenarios:

- Axis is in state "OUTDATED"
- Axis is in state "ERRORSTOP"

Being "outdated" indicates, that the axis is in a configuration mode and might faced issues during the start up of the system. A simple reset might fix the issue. However, a restart of the complete system might be necessary.

An error stop occurs when the drive faces issues that interfere with the usual operations. These can be in range of >invalid motion commands due to limits< up to >emergency stops< triggered by an operator.

3.3.4. Jog

The two arrows up and down on the right side allow tha manual movement of the axis. The target velocity (250 mm/s * slide override) as well as the other dynamic parameters are fixed and cannot be changed.

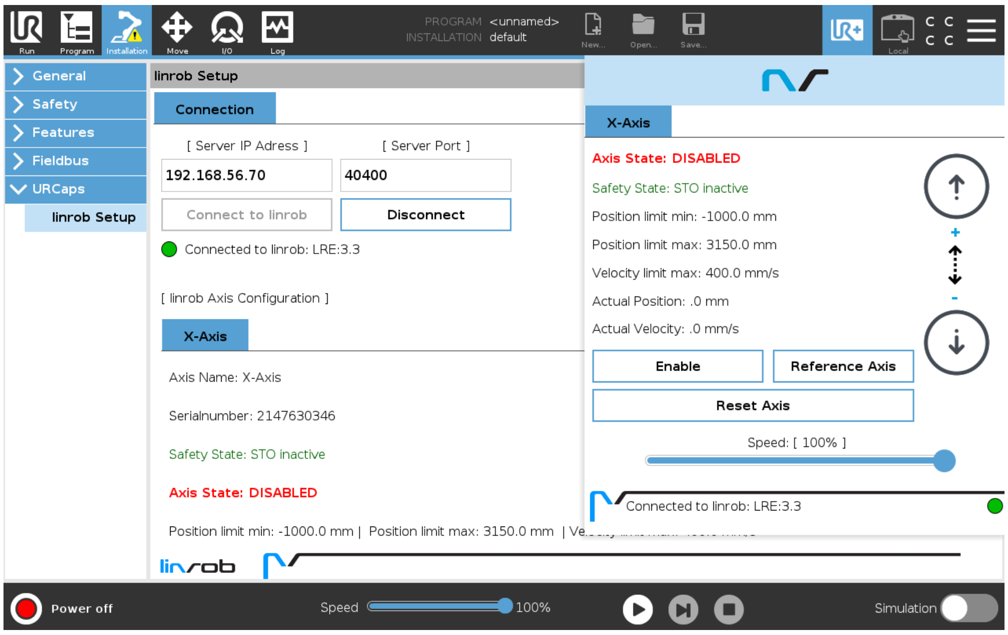
Only an enabled axis in state "STANDSTILL" can and should be moved that way.

The following script commands can be used inside the programming interface and affect the drive actively.

3.3.5. Disable

```
lr_disable_servo(axis)
```

Once successfully executed, the disabled state is clearly indicated within the user interface.



linrob Setup

[Server IP Adress] 192.168.56.70
[Server Port] 40400

Connected to linrob: LRE:3.3

[linrob Axis Configuration]

Axis Name: X-Axis

Serialnumber: 2147630346

Safety State: STO inactive

Axis State: DISABLED

Position limit min: -1000.0 mm | Position limit max: 3150.0 mm | Ve

Axis State: DISABLED

Safety State: STO inactive

Position limit min: -1000.0 mm

Position limit max: 3150.0 mm

Velocity limit max: 400.0 mm/s

Actual Position: .0 mm

Actual Velocity: .0 mm/s

Speed: [100%]

Connected to linrob: LRE:3.3

This function disables the drive and applies the break, if available. The attached axes are not operational and in a safe state.

```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.3.6. Enable

```
lr_enable_servo(axis)
```

This function enabled the drive and brings it under regulation. The brake, if available, is released.

```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.3.7. Move

```
lr_move_l(axis, position, velocity, acceleration, deceleration)
```

This function moves the axis absolute to a defined position.

```
    axis            (int)::
        Axis to select from available connected. Not in use!
    position        (float)::
        Target position in mm.
    velocity        (float)::
        Target velocity in mm/s.
    acceleration    (float)::
        Target acceleration in mm/s².
    deceleration    (float)::
        Target deceleration in mm/s².
```

3.4. Helper functions

As opposed to commands, that don´t have a return value; the Linrob URcap also exposes a number of helper functions that return useful information. They typically are used as the expression of a conditional, such as an `if` statement.

3.4.1. Axis in motion

```
lr_axis_in_motion(axis)
```

Returns, if the specified axis is currently in motion as a string boolean.

```
axis            (int)::
    Axis to select from available connected (e.g. "X-Axis"). Not in use!
```

3.4.2. Wait for Position

```
lr_wait_for_position(axis)
```

Returns, if the last commanded target position was reached by the specified axis as a string boolean

```
axis            (int)::
    Axis to select from available connected (e.g. "X-Axis"). Not in use!
```

3.4.3. Get STO state

```
lr_get_actual_sto_state(axis)
```

*Returns the current state of the STO (**Safe Torque Off**) signal of the specified drive. Usually, all drives should return the same signal. Value is a string boolean*

```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.4.4. Get state

```
lr_get_actual_state(axis)
```

Returns the current state of the specified axis. The state can be one of :

OUTDATED_RESET
: Axis must be reset, drive is in configuration state.

OUTDATED
: Axis must be reset, drive is in configuration state. (Bug state)

UNREF_DISABLED
: Axis is disabled and not references. Please enable.

UNREFERENCED
: Axis is enabled but not referenced. Please reference.

DISABLED
: Axis is disabled. It cannot perform any motion.

STANDSTILL
: Axis is enabled and ready to move.

STANDSTILL_PENDING
: Axis will be enabled, process not finished.

ABORTING
: Current motion will be stopped. Deceleration phase.

DISCRETE_MOTION
: Axis is in motion.

ERRORSTOP
: Error occured and effected drive. Axis must be reset.

```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.4.5. Get position

```
lr_get_actual_position(axis)
```

Returns the actual position in mm.

```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.4.6. Get velocity

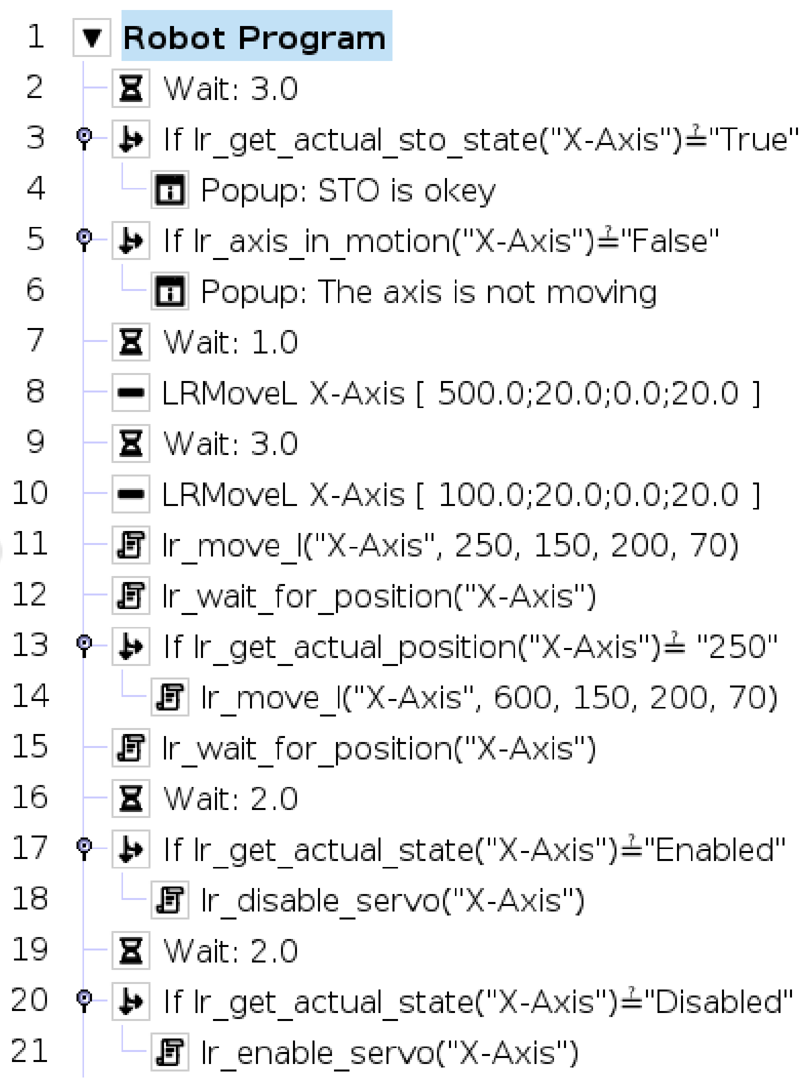
```
lr_get_actual_velocity(axis)
```

Returns the actual velocity in mm/s.

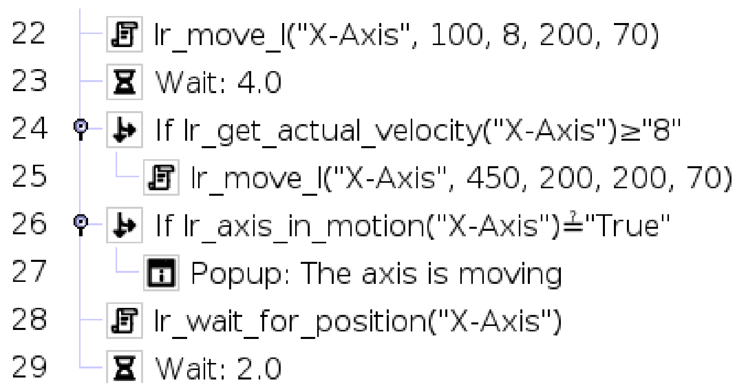
```
axis            (int)::
    Axis to select from available connected. Not in use!
```

3.5. Example program

Let’s examine our program example in more detail. As highlighted earlier, axis control can be executed in two ways: one is integrated within the installation interface, and the other is activated via Nodes in the program nodes interface. By examining both methods in this example, we aim to provide a clear and thorough understanding of the system’s functionality. So, let’s proceed.



```
 1  Robot Program
 2    Wait: 3.0
 3    If lr_get_actual_sto_state("X-Axis")≟"True"
 4      Popup: STO is okey
 5    If lr_axis_in_motion("X-Axis")≟"False"
 6      Popup: The axis is not moving
 7    Wait: 1.0
 8    LRMoveL X-Axis [ 500.0;20.0;0.0;20.0 ]
 9    Wait: 3.0
10    LRMoveL X-Axis [ 100.0;20.0;0.0;20.0 ]
11    lr_move_l("X-Axis", 250, 150, 200, 70)
12    lr_wait_for_position("X-Axis")
13    If lr_get_actual_position("X-Axis")≟ "250"
14      lr_move_l("X-Axis", 600, 150, 200, 70)
15    lr_wait_for_position("X-Axis")
16    Wait: 2.0
17    If lr_get_actual_state("X-Axis")≟"Enabled"
18      lr_disable_servo("X-Axis")
19    Wait: 2.0
20    If lr_get_actual_state("X-Axis")≟"Disabled"
21      lr_enable_servo("X-Axis")
```



```
22    lr_move_l("X-Axis", 100, 8, 200, 70)
23    Wait: 4.0
24    If lr_get_actual_velocity("X-Axis")≥"8"
25      lr_move_l("X-Axis", 450, 200, 200, 70)
26    If lr_axis_in_motion("X-Axis")≟"True"
27      Popup: The axis is moving
28    lr_wait_for_position("X-Axis")
29    Wait: 2.0
```

1. **Line 2:** The program begins with a three-second wait (`Wait: 3.0`).
2. **Lines 3-4:** It checks the actual STO state of the "X-Axis." If `True`, a popup appears stating "STO is okey" (`If lr_get_actual_sto_state("X-Axis")=="True"` followed by `Popup: STO is okey`).
3. **Line 5-6:** The program checks if the "X-Axis" is in motion. If `False`, a popup states "The axis is not

moving" (`If lr_axis_in_motion("X-Axis")=="False"` followed by `Popup: The axis is not moving`).

4. **Line 7:** There's a one-second wait (`Wait: 1.0`).
5. **Line 8:** An LRMoveL command moves the X-Axis to a position of 500 (`LRMoveL X-Axis [ 500.0;20.0;0.0;20.0 ]`).
6. **Line 9:** This is followed by a three-second wait (`Wait: 3.0`).
7. **Line 10:** Another LRMoveL command moves the X-Axis to a position of 100 (`LRMoveL X-Axis [ 100.0;20.0;0.0;20.0 ]`).
8. **Line 11-12:** The "X-Axis" is commanded to move to a position of 250, and the program waits for it to reach this position (`lr_move_l("X-Axis", 250, 150, 200, 70)` followed by `lr_wait_for_position("X-Axis")`).
9. **Line 13-14:** The actual position is checked, and if it's 250, another move command is issued (`If lr_get_actual_position("X-Axis")=="250"` followed by `lr_move_l("X-Axis", 600, 150, 200, 70)`).
10. **Line 15-18:** After waiting for the position to be reached and a subsequent two-second wait, the program checks if the "X-Axis" is enabled. If so, it disables the servo (`If lr_get_actual_state("X-Axis")=="Enabled"` followed by `lr_disable_servo("X-Axis")`).
11. **Line 19-20:** The state is checked again; if the axis is disabled, the servo is enabled (`If lr_get_actual_state("X-Axis")=="Disabled"` followed by `lr_enable_servo("X-Axis")`).
12. **Line 22-23:** The program moves the "X-Axis" to a new position with a slower velocity and waits for four seconds (`lr_move_l("X-Axis", 100, 8, 200, 70)` followed by `Wait: 4.0`).
13. **Line 24-25:** The actual velocity is checked. If it is 8, a command is issued to move the axis (`If lr_get_actual_velocity("X-Axis")=="8"` followed by `lr_move_l("X-Axis", 450, 200, 200, 70)`).
14. **Line 26-27:** It checks if the "X-Axis" is in motion. If `True`, a popup indicates "The axis is moving" (`If lr_axis_in_motion("X-Axis")=="True"` followed by `Popup: The axis is moving`).
15. **Line 28:** The program waits for the "X-Axis" to reach the commanded position (`lr_wait_for_position("X-Axis")`).
16. **Line 29:** The sequence ends with a final two-second wait (`Wait: 2.0`).

```
This example program showcases the use of Linrob URCap commands for
moving an axis, enabling/disabling the servo, checking the state of the
axis, and ensuring that movements are executed as expected through
conditional checks and popup notifications for user awareness.
```

```
Within the installation interface, executing the LRMoveL command is a
self-contained action: the command autonomously monitors its progress
and completes once the specified position is achieved. On the other
hand, in the program node, the lr_move_l command requires an explicit
confirmation of position attainment. This is done by using the
lr_wait_for_position command, which acts as a sentinel, ensuring that
the program pauses until the movement to the intended position is
```

```
confirmed.
```

4. Combined Motions

In this section, a method for executing combined movements of the robot and our Linrob axis is detailed.

Under this single-threaded approach, combined movement is executed sequentially. Begin by initiating the movement of the Linrob axis using the `lr_move_l` command, ensuring all required parameters are accurately set. Following the completion of this step, you can then select either the `moveL` or `moveJ` command, depending on your requirements, to execute the robot's movement.

5. Handling State Transitions

In this section, we delve into the behavioral dynamics of Linrob in response to user interactions, specifically the pressing of play, stop, or pause buttons within the Linrob URcap interface on the teach pendant.

Initial State: Stopped

Upon the installation of the Linrob URcap and the creation of your custom program, the initial state observed is 'Stopped'. This state is by design, necessitating the user to initiate the program by pressing the play button.

Transitioning States: Play, Pause, and Stop

Play to Pause: When the user transitions the program from play to pause, Linrob enters a temporary pause state. After this transition, the axis will stop its current movement right after the transition.

Pause to Resume: Upon pressing the resume button, Linrob transitions out of the pause state and resuming the application from the point of pause. If the axis interrupted its movement, the re-triggered movement will be executed but will not wait until the target position is reached! (See "Known issues")

Pressing Stop: Activating the stop button initiates a complete halt of the application. The axis executes the last command completely. To resume operations after a stop, restarting from the beginning of or any other point within the program is necessary.

6. Safety

In our Linrob URCap setup, we have integrated the UR controller’s configurable outputs (CO1 and CO0) with our system’s Safe Torque Off (STO) pins. This integration creates a unified emergency stop system.

To resume normal operation after an emergency stop has been engaged, it is necessary to release the emergency stop button.

Activating the emergency stop button on the teach pendant instantly triggers a global safety response. This action not only halts all operations of the robot but also ceases any movement of the Linrob axis.

After the emergency stop has been activated, particularly during axis jogging via the UR+ toolbar, it is essential to properly reset the system. Recovery involves re-enabling the axis.

If the emergency stop is activated while the main robot program is running through the Program interface, the entire program will be halted immediately. Recovery may involve re-enabling of the axis.

Should there be a need to adjust operational parameters beyond the stated ranges, it is advisable to contact our support team for assistance.

7. Known issues

Every software comes with a certain amount of know issues. Although we are working continuously on our product and its features, we are aware of some smaller issues. We encourage every customer to report unknown issues to us!

In the current version, in the case of the axis being moved using the toolbar (jogging) and an e-stop being triggered in the same time, the pressed button stays highlighted in the UI even when no longer pressed.

The label of the "Enable/Disable" button is correct most of the time. However, after startup it copies the actual state of the axis and displays it. After a reset, the axis usually enters the state "DISABLED" but the button label does not follow.

The label does not affect the functionality.

Our linCap comes in updated versions regularly, especially at the current state. Alongside the linCap comes the intermediate application handling the UR requests and convert them to executable commands for the linrob controller.

We are working on an intuitive, easy to follow compatibility check. During this process we encourage you to get in contact as soon as you face any incompatibilities.

When the axis is paused (movement interrupted) and the program is resumed, the last command will be sent again. However, it does wait until the target position is reached and jumps right into the following command.