UR3e □□

industrial collaborative robot arm

- UR Offline Simulator for UR3e
  - Running UR offline simulator in VirtualBox
  - UR offline simulator in WSL2 (Ubuntu22.4 Yammy □□) and Docker
  - MoveJ, MoveL and MoveP
- Controlling the robot via MQTT and Python □□
  - Reading joint states with an MQTT client from a Mosquitto MQTT broker □□

UR Offline Simulator for UR3e

Running UR offline simulator in VirtualBox

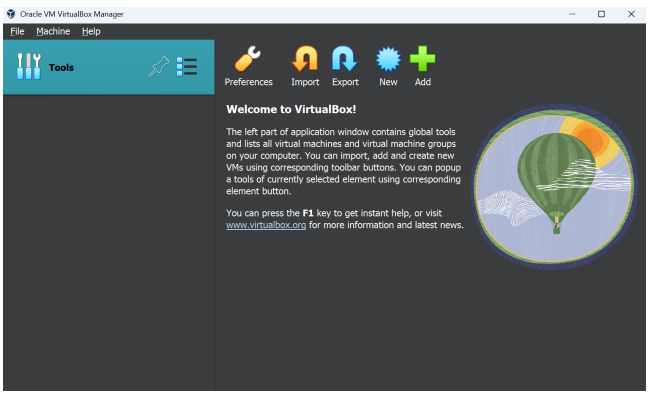
Download UR Sim Lubuntu Image

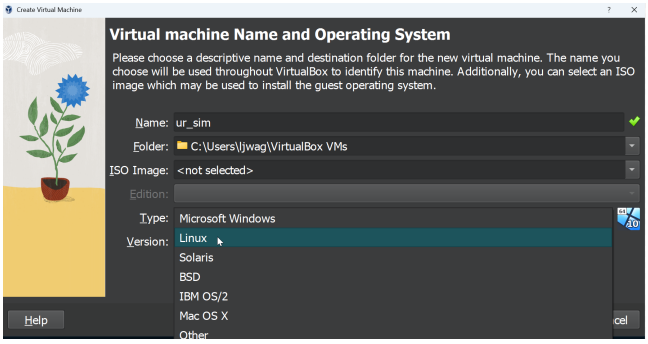
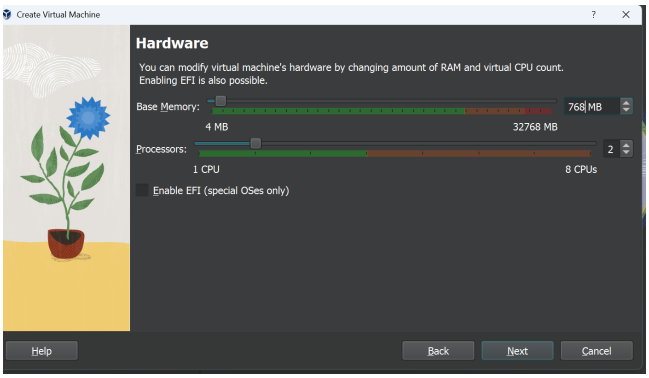
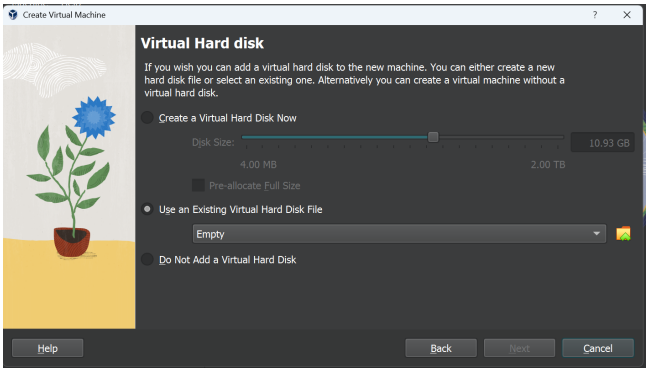
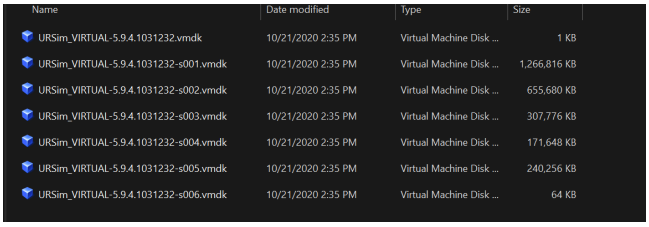
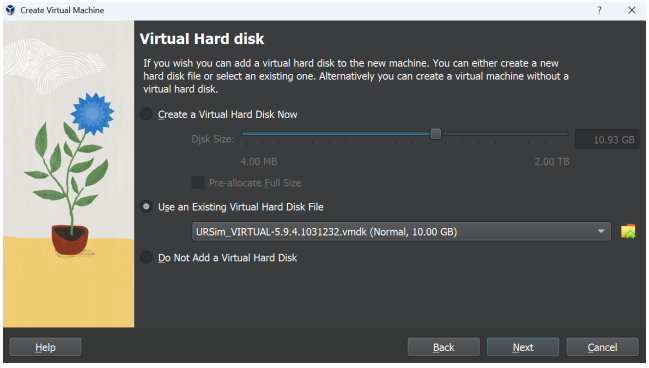
1. Download the **URSim system image**
2. **unzip it! if you get an error, download 7zip under WINDOWs**
3. > **for MAC users please download the already unzipped files from Sciebo**

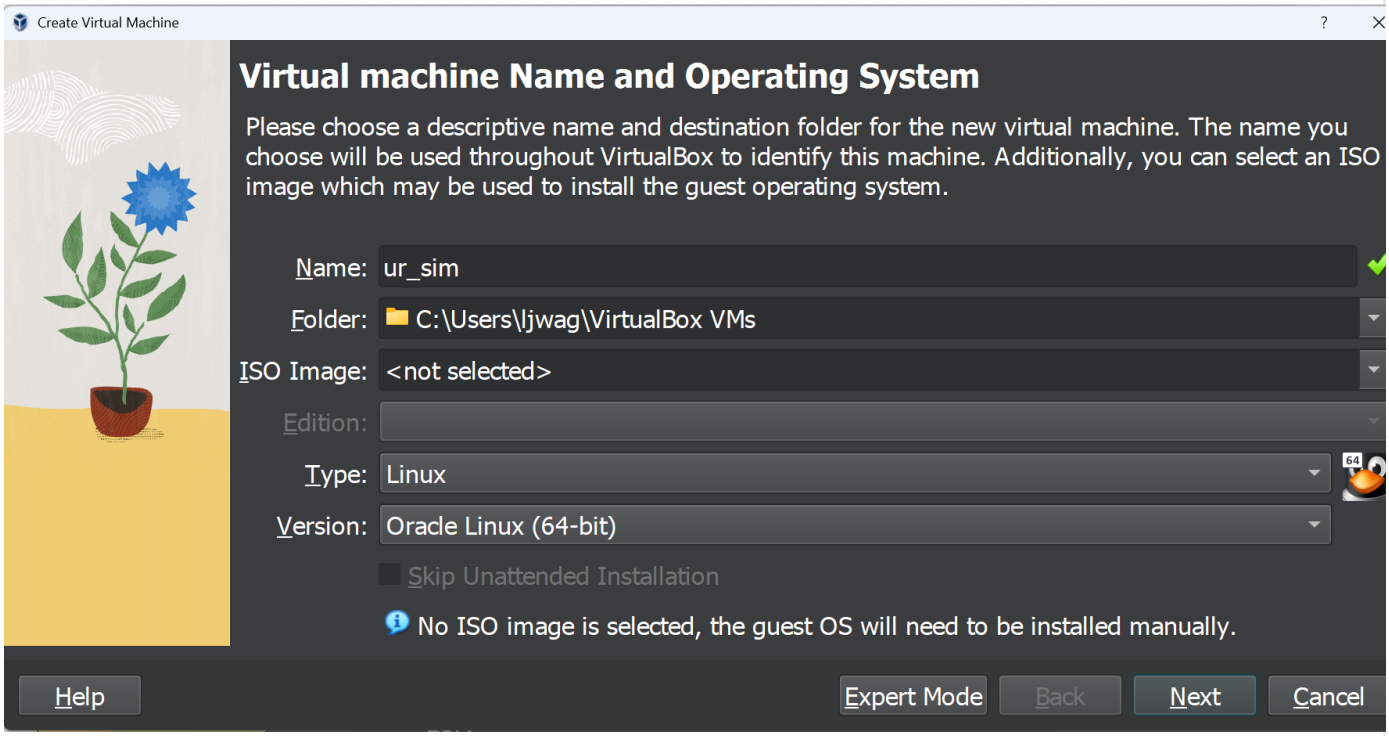
Install virtualBox

1. Visit VirtualBox **download page**, choose your operating system and download virtual box
2. Click the downloaded executable
3. click 'next' in the setup wizard until it's installed

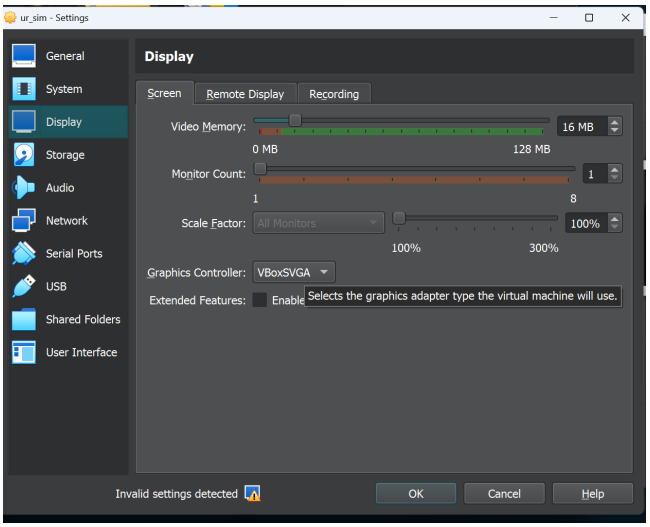
Setup the virtual machine inside virtualBox

open VirtualBox	
click on 'New' to open the dialogue to configure a new virtual machine	

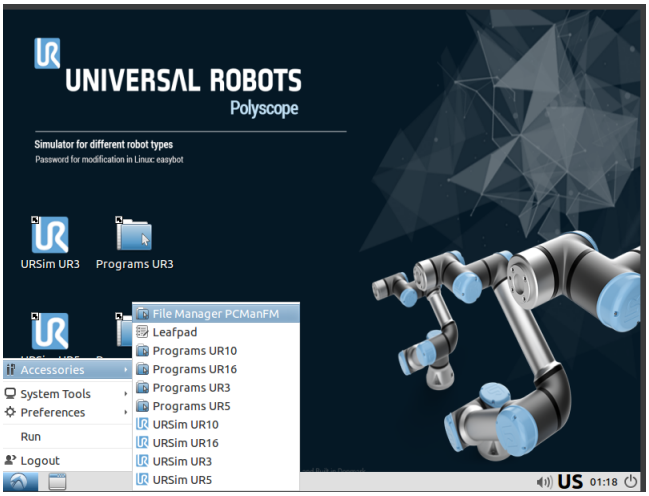
use the following settings: give a name to your virtual machine don't select an ISO image choose 64bit Linux as the 'Version'	
3. Select Memory size of 768 MB and press 'Next'	
click on 'Use an Existing Virtual Hard Disk File'	
from the folder containing the extracted files, choose the first one:	
click 'next'	



configure Lubuntu UR Sim to start Polyscope in the right resolution

1. go to the ⚙settings and open the dialogue 2. in the settings go to display ◫ and change the Graphics Controller to VBoxSVGA	
open VirtualBox	

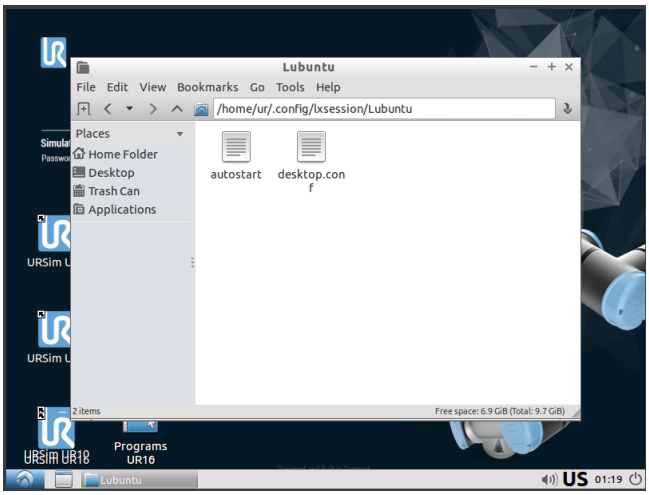
In the left corner click the icon
choose File Manager



inside File Manager navigate to

/home/ur/.config/lxsession/Lubuntu

double click on the file named 'autostart'

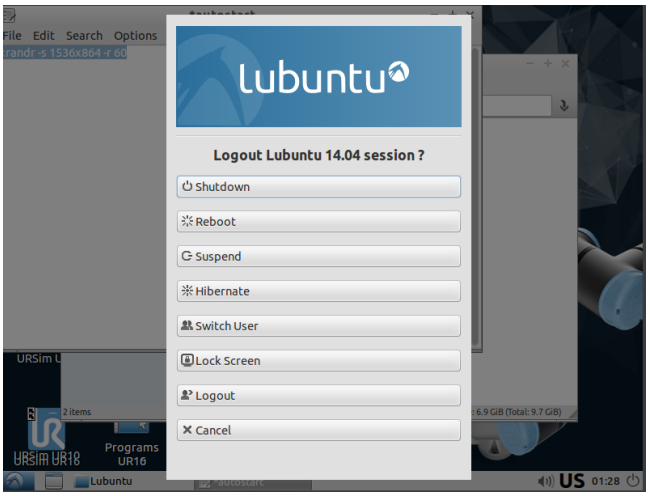


add the following line:

```
xrandr -s 1536x864 -r 60
```

click file --> save or ctr+s

Image not found or type unknown

click on the icon on the right bottom corner click on 'reboot'	

UR offline simulator in WSL2 (Ubuntu22.4 Yammy □□) and Docker

Install WSL 2 and Ubuntu 22.04

1. **Enable WSL 2**: Follow Microsoft's official documentation to install and set up **WSL2.**
2. **Install Ubuntu 22.04**: Once WSL 2 is installed and enabled, install Ubuntu 22.04 from the Microsoft Store.

Install Docker Desktop

1. Open a Ubuntu 22.04.2 LTS terminal via Windows Terminal
2. Install Docker according to these instructions
3. Pull the newest URSim docker image - instructions here.

Run UR Sim in Docker

1.

```
sudo docker run --rm -it universalrobots/ursim_e-series
```

2. Access the ursim-interface **through the URL in the output**

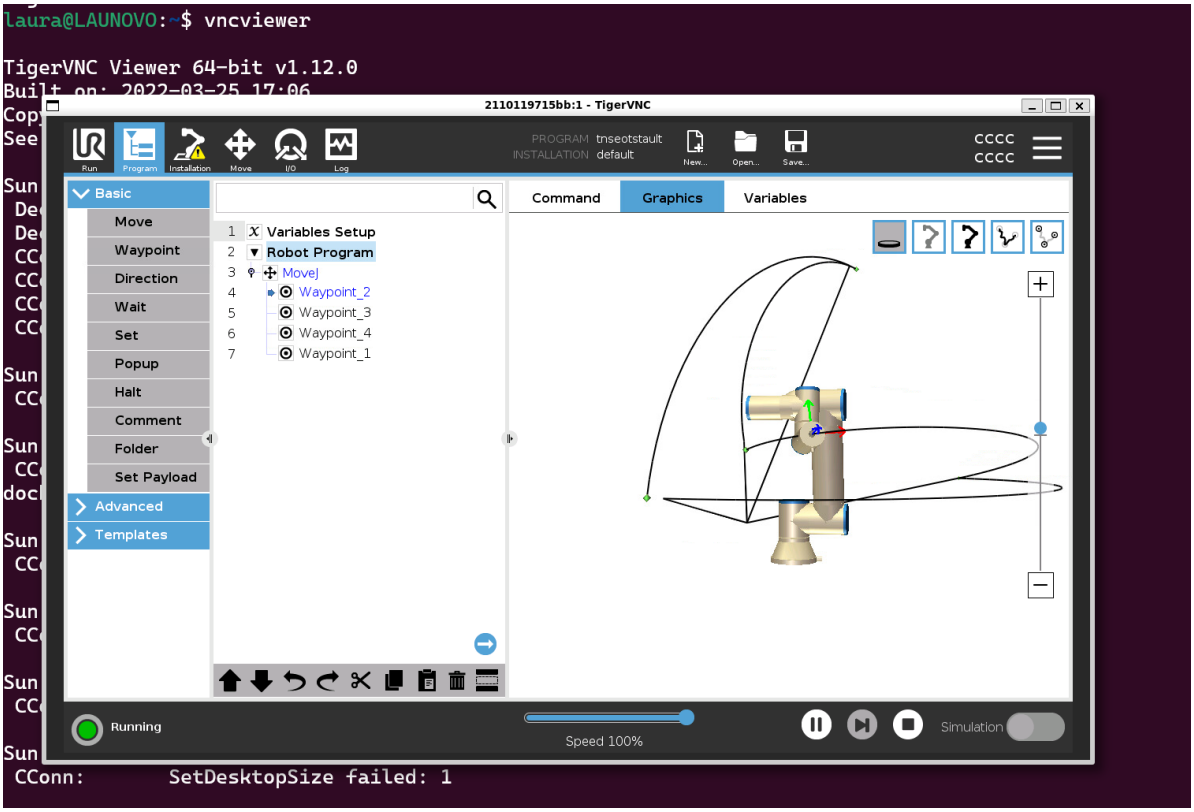
optional: install tiger vnc

```
sudo apt install tigervnc-viewer
```

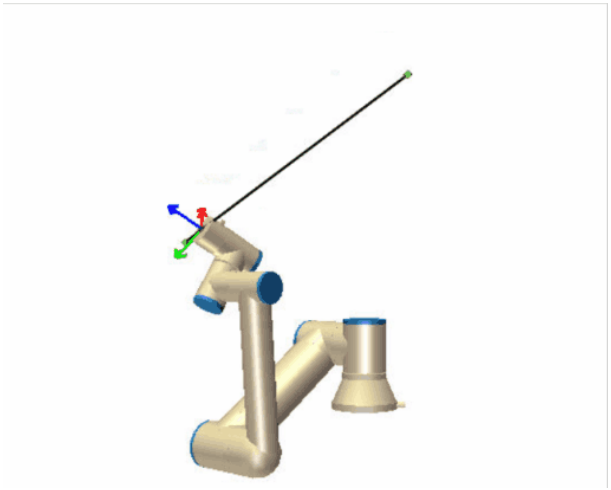
starting vnc viewer:

```
vncviewer
```

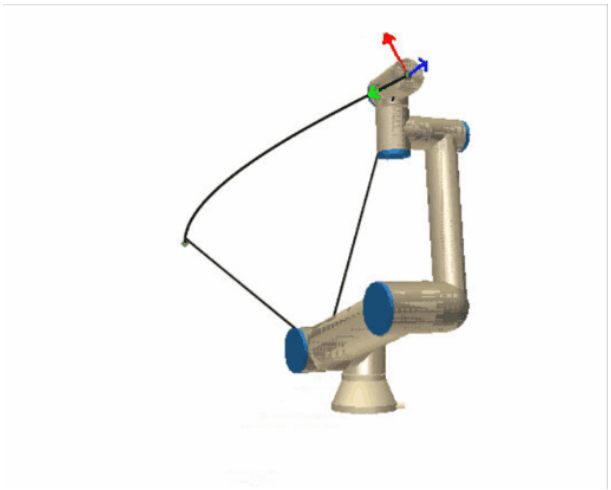
> Best is to open the chromium-browser also in WSL2 and copy the ip or using a vinc client like TigerVNC



MoveJ, MoveL and MoveP

- 

  moveL (Move Linear):
  - "L" stands for "Linear."
  - With the "moveL" command, the robot moves in a straight line from its current position to the target position.
  - Unlike "moveJ," where the focus is on the joints, "moveL" focuses on the path that the robot's end-effector (like its hand) takes. It moves linearly, meaning in a straight line.
  - Back to your arm analogy, using "moveL" is like moving your hand from point A to point B in a straight line, which might require more complex, coordinated movements from your shoulder, elbow, and wrist.



moveJ (Move Joint):
- "J" stands for "Joint."
- In a "moveJ" command, the robot moves from its current position to the target position by rotating its joints.
- The primary focus here is on the joints' movements: each joint reaches its target angle to achieve the desired position. This might not be the straightest path, as the robot prioritizes the joints' ease of motion.
- Imagine your arm is the robot arm. If you were to use "moveJ," you'd move each joint (shoulder, elbow, wrist) individually to place your hand in the desired position, regardless of the path your hand takes through the air.

	moveP (Move Process): ◦ "P" stands for "Process." ◦ The "moveP" command is a bit more complex. It's used for a more controlled movement, where the robot moves **steadily** ideal for a process task, like gluing, welding, or painting. ◦ In the arm analogy, "moveP" would be like drawing a smooth, continuous line with a marker. Your hand (the end-effector) needs to move at a **consistent speed** and path to create the best line, even if that means your arm's joints (the robot's joints) need to adjust in complex ways.

Controlling the robot via MQTT and Python □□

Reading joint states with an MQTT client from a Mosquitto MQTT broker □□

Install an MQTT Client

1. Install a mosquitto □□ client for example **MQTTX**

Connect to the WIFI

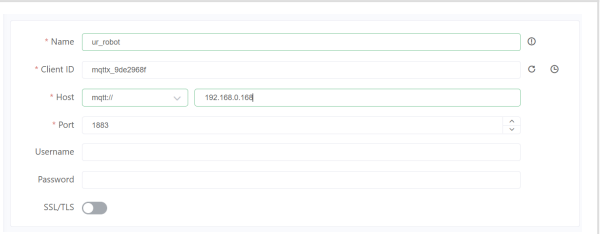
Connect to the same WI-FI the MQTT-Broker is in.

> In our case the WI-FI's name is '**ROOM_240**'

> In our case the device running the broker is an RaspberryPi □□ with the IP **192.168.0.168**

Subscribe to a topic via your MQTT client

1. Setup a new connection by clicking □ :

Setup a new connection to a "broker" in our case this is the Raspberry Pi □□ connected to the Robot. Provide a name and leave the username and password fields blank.	

Click on "connect" and you'll see the joint positions coming in and constantly updated!