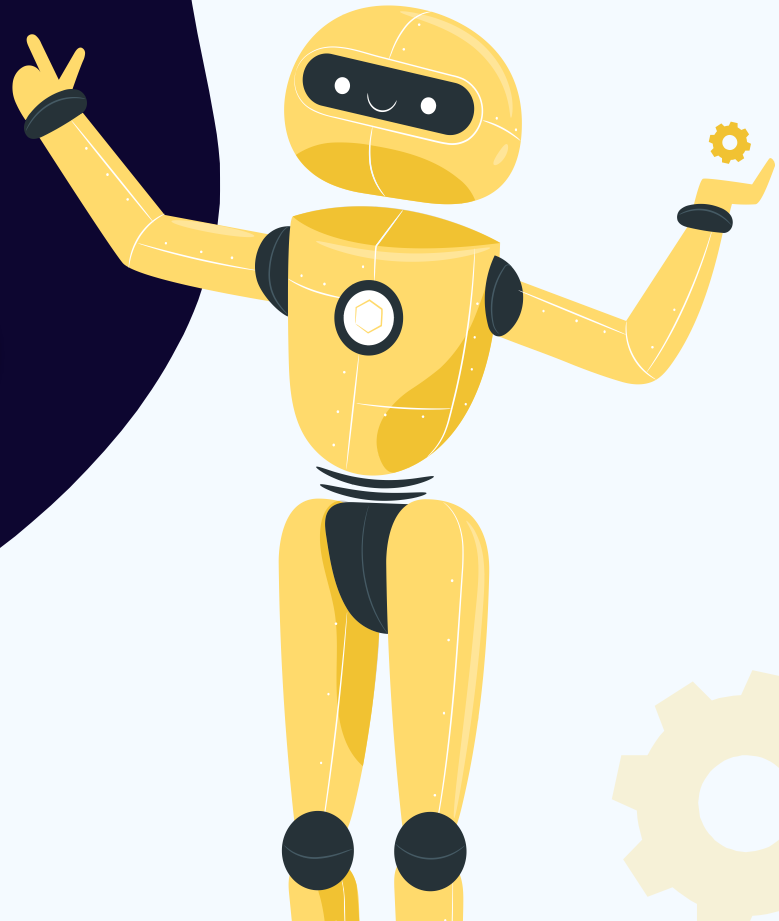
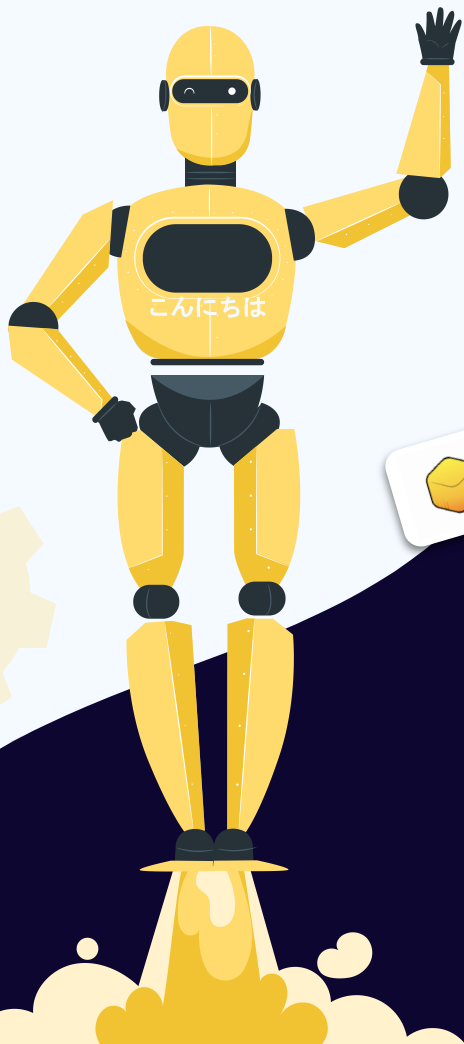


Teaching ROS with JupyterLab

Wolf Vollprecht
Isabel Paredes





QuantStack

Scientific Computing



WOLF
VOLLPRECHT



ISABEL
PAREDES



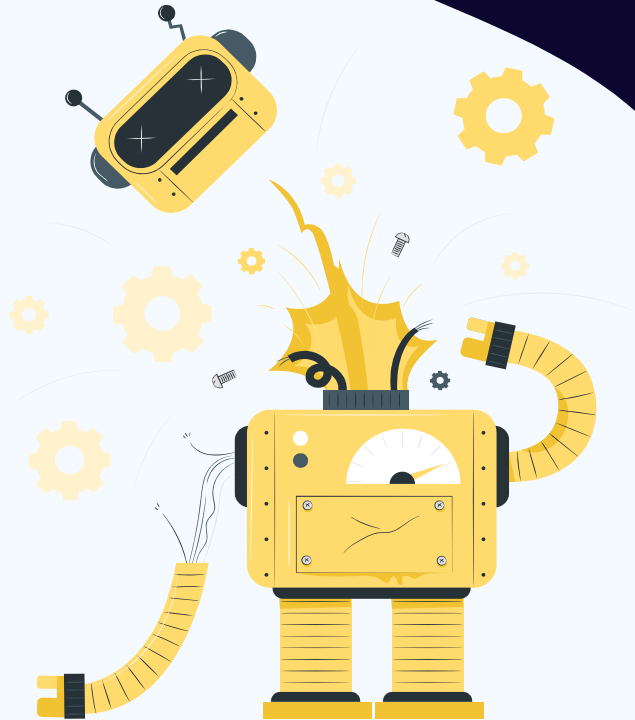
CARLOS
HERRERO



DENISA
CHECIU



PROBLEM



Students

- Hardware requirements
- Background knowledge

Instructors

- Shareable workspaces
- Reproducibility



SOLUTION

RoboStack

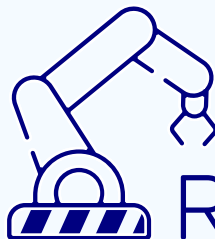
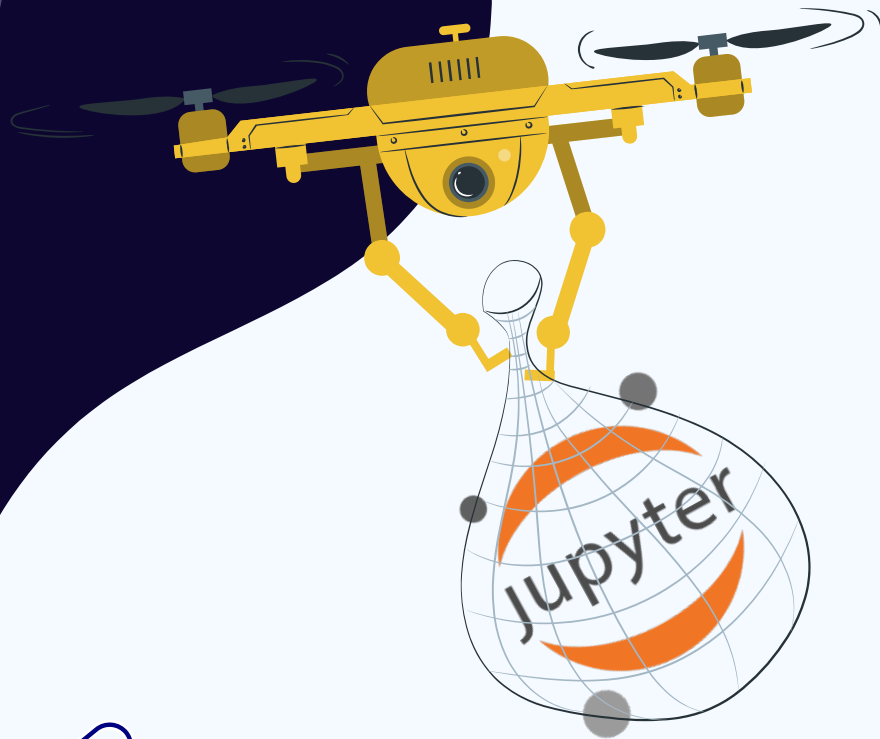
- Cross-platform packages for ROS 1 and 2

JupyterLab Extensions

- JupyROS
- JupyterLab-ROS
- JupyterLab-Blockly
- JupyterLab-URDF

JupyterHub

- Docker



RoboStack

OPEN SOURCE ROBOTICS SOFTWARE

GETTING STARTED



INSTALLATION

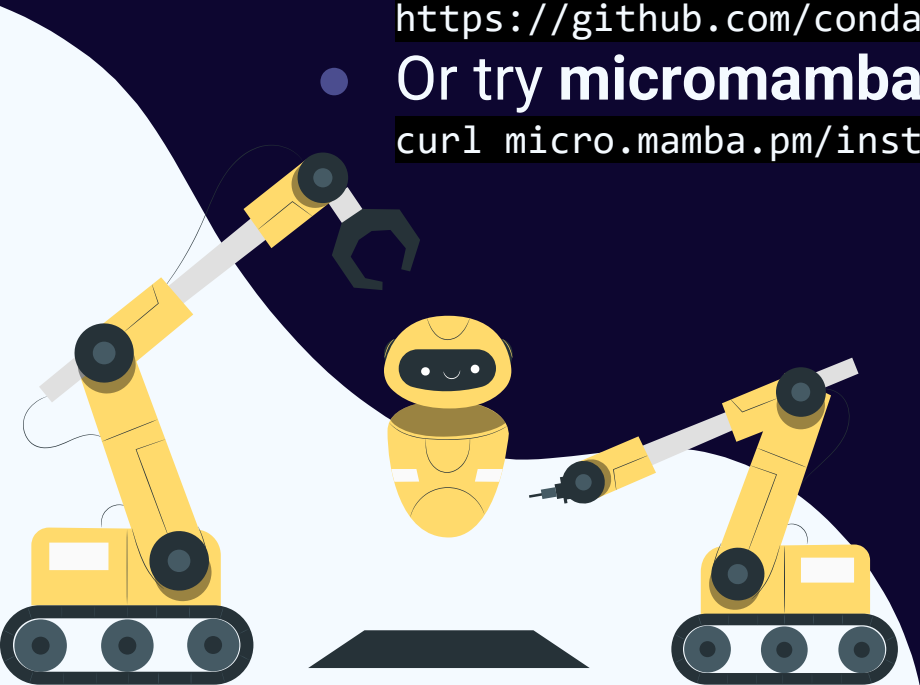
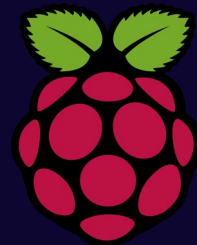
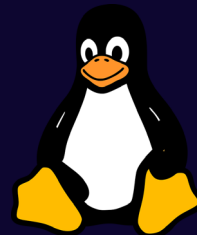
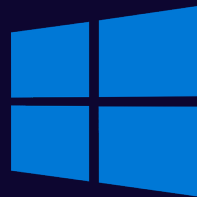
Get the mamba package manager

- Install **mambaforge** to get started

```
https://github.com/conda-forge/miniforge#mambaforge
```

- Or try **micromamba**

```
curl micro.mamba.pm/install.sh | bash
```



ROS1 ENVIRONMENT

```
$ mamba create -n ros1_env ros-noetic-desktop -c robostack
```

```
...
```

```
$ mamba activate ros1_env
```

Install extensions:

```
$ mamba install jupyros -c conda-forge  
                jupyterlab-ros -c robostack  
                jupyterlab-blockly -c conda-forge  
                jupyterlab-urdf -c conda-forge
```

ROS2 ENVIRONMENT

```
$ mamba create -n ros2_env ros-humble-desktop -c robostack-humble  
...  
$ mamba activate ros2_env
```

Install extensions:

```
$ mamba install jupyros -c conda-forge
```

The image features the JupyterLab logo in a bold, white, sans-serif font, centered on a solid yellow background. The logo is composed of the word "JUPYTER" followed by "LAB" in a slightly smaller font size. The letters are closely spaced, with some overlapping. Surrounding the central text are several gear icons of varying sizes, also in white. There are five gears in total: one in the top left, one in the top right, one in the middle right, one in the bottom left, and a larger one in the bottom right. A white, wavy, abstract shape is located in the bottom left corner, partially overlapping the yellow background.

JUPYTERLAB

JUPYROS WIDGETS

```
import rospy
import jupyter
from std_msgs.msg import String
from geometry_msgs.msg import Pose
```

```
rospy.init_node('subpub_node')
```

```
jupyter.subscribe('/pose_stream', Pose, lambda msg: print(msg))
```

Stop

position:

x: 1.0

y: 2.0

z: 3.0

orientation:

x: 9.0

y: 8.0

z: 7.0

w: 0.0

```
jupyter.publish('/pose_stream', Pose)
```

position

x: 1

y: 2

z: 3

orientation

x: 9

y: 8

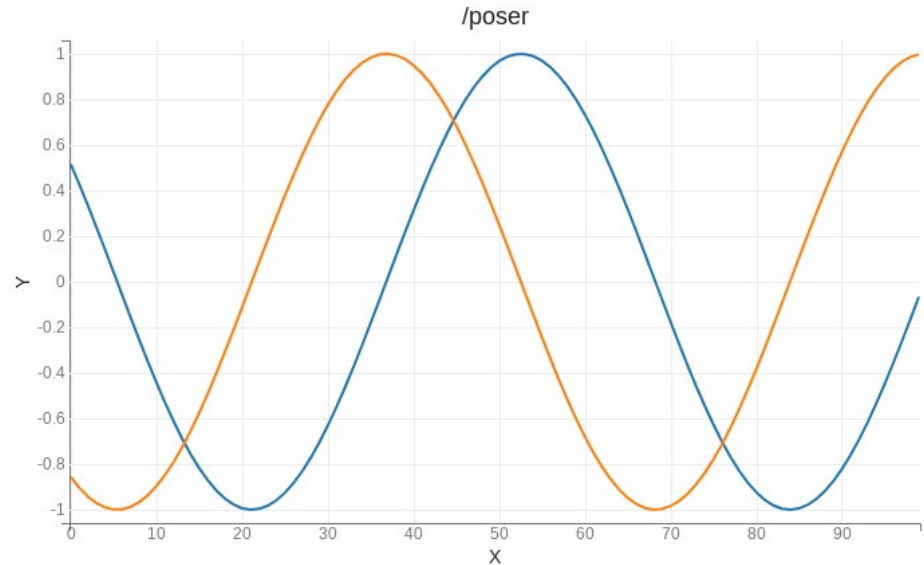
z: 7

w: 0

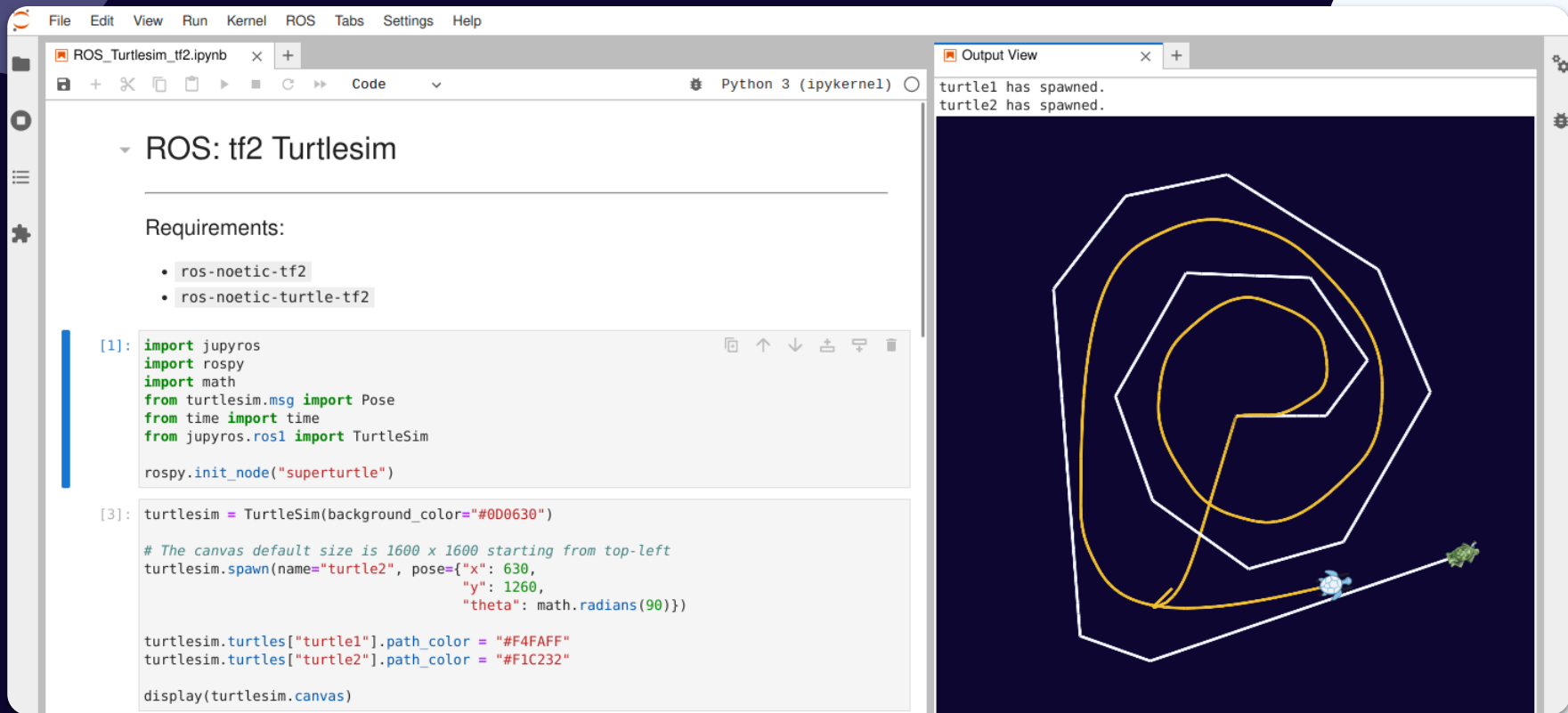
Send Message

☐ Latch Message

```
jr.live_plot('/poser/:x:y', Vector3)
```



JUPYROS TURTLES



The screenshot displays a Jupyter Notebook environment with a dark theme. The notebook is titled "ROS_Turtlesim_tf2.ipynb" and is running on a "Python 3 (ipykernel)" environment. The code is organized into cells, with the first cell containing imports and node initialization, and the second cell containing the simulation setup and display commands.

ROS: tf2 TurtleSim

Requirements:

- `ros-noetic-tf2`
- `ros-noetic-turtle-tf2`

```
[1]: import jupyteros
import rospy
import math
from turtlesim.msg import Pose
from time import time
from jupyteros.ros1 import TurtleSim

rospy.init_node("superturtle")

[3]: turtlesim = TurtleSim(background_color="#0D0630")

# The canvas default size is 1600 x 1600 starting from top-left
turtlesim.spawn(name="turtle2", pose={"x": 630,
                                     "y": 1260,
                                     "theta": math.radians(90)})

turtlesim.turtles["turtle1"].path_color = "#F4FAFF"
turtlesim.turtles["turtle2"].path_color = "#F1C232"

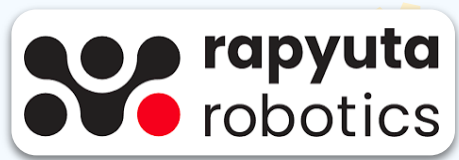
display(turtlesim.canvas)
```

The Output View on the right shows the following messages:

```
turtle1 has spawned.
turtle2 has spawned.
```

The visualization shows a dark blue canvas with a white spiral path. Two turtles are visible: a light blue turtle (turtle1) and a green turtle (turtle2). The green turtle is currently at the end of the spiral path, which is colored yellow.

JUPYTER LAB ROS



File Edit View Run Kernel URDF ROS Tabs Settings Help

Filter files by name

Name

- launch
- joint_state_publisher.py
- ROS_3D_Grid.ipynb
- ROS_3D_Laser_Scan.ipynb
- ROS_3D_Robot_Model.ipynb
- ROS_3D_TEB_Markers.ipynb
- ROS_Actions_Clients.ipynb
- ROS_Joint_States.ipynb
- ROS_Live_Plotting.ipynb
- ROS_Publisher_Subscriber.ipynb
- ROS_Services_Clients.ipynb
- ROS_Turtlesim_tf2.ipynb
- ROS_Turtlesim.ipynb
- Ros2 Pub_Sub_Communication....
- zethus.launch

ZETHUS

Controls Pose Estimate Nav Goal Point

Connected.

[Edit Configuration](#)

Background Color:

Grid size: 30

Fixed frame: world

[Add Visualization](#)

RobotModel

- Links
- Joints

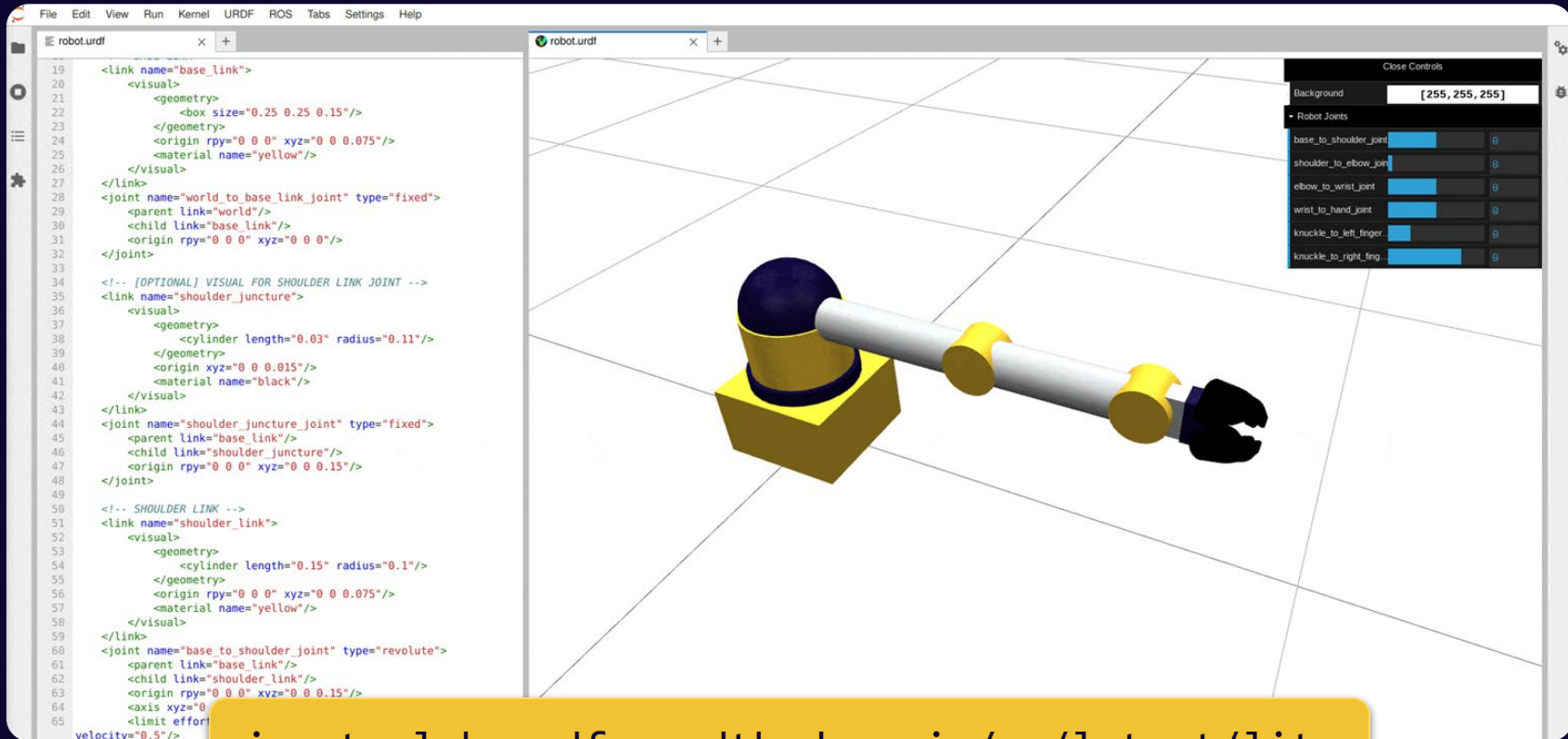
Delete Hide

48 FPS (0-60)

[RQT Graph](#) Raw: ☐ Expand

A 3D rendering of a white robotic arm with black joints, positioned on a white base. The arm is shown in a 3D environment with a grid floor and a light gray background. The interface includes a file explorer on the left, a configuration panel on the top left, and a toolbar on the top right.

JUPYTERLAB URDF



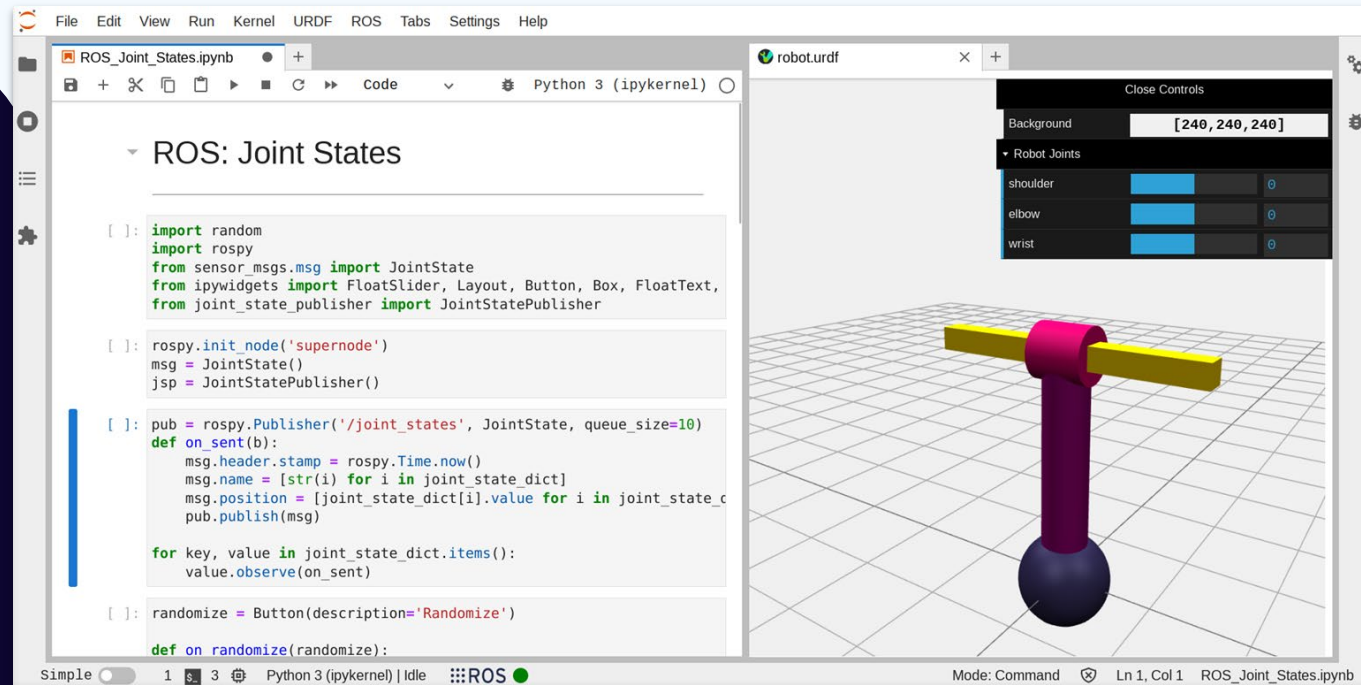
jupyterlab-urdf.readthedocs.io/en/latest/lite

JUPYTER LAB BLOCKLY

The image displays the JupyterLab Blockly interface, which is used for developing and controlling a robotic arm simulation. The interface is divided into several sections:

- File Explorer (Left):** Shows a file tree with folders like `build`, `devel`, `dynamixel_sdk`, `logs`, `mcp_can_rpi`, `src`, `LICENSE`, `niryo.jpblockly`, `niryo.launch`, `README.md`, and `untitled.jpblockly`.
- Blockly (Center):** A visual programming environment where blocks are used to create logic. The current blocks include:
 - IP Address: 169 . 254 . 200 . 200
 - Calibrate motors (auto)
 - Move Joints: j1 0 j2 0 j3 0 j4 0 j5 0 j6 0
 - Wait for: 2 seconds
 - Move to home pose
- Code Editor (Bottom):** Contains Python code for the `niryo` class:

```
[6]: from pyniryo import *  
  
class niryo_connect():  
    def __init__(self, ip):  
        self.n = NiryoRobot(ip)  
    def enter(self):  
        return self.n  
    def exit(self, exception_type, exception_value, traceback):  
        self.n.close_connection()
```
- Zethus (Right):** A 3D simulation environment showing a blue robotic arm. The interface includes controls for `Controls`, `Pose Estimate`, `Nav Goal`, and `Point`. A status bar at the bottom right shows `52 FPS (10-60)` and `RQT Graph`.



The image displays a JupyterLab environment with two main panes. The left pane shows a Jupyter Notebook titled 'ROS: Joint States' with the following Python code:

```
[ ]: import random
import rospy
from sensor_msgs.msg import JointState
from ipywidgets import FloatSlider, Layout, Button, Box, FloatText,
from joint_state_publisher import JointStatePublisher

[ ]: rospy.init_node('supernode')
msg = JointState()
jsp = JointStatePublisher()

[ ]: pub = rospy.Publisher('/joint_states', JointState, queue_size=10)
def on_sent(b):
    msg.header.stamp = rospy.Time.now()
    msg.name = [str(i) for i in joint_state_dict]
    msg.position = [joint_state_dict[i].value for i in joint_state_dict]
    pub.publish(msg)

for key, value in joint_state_dict.items():
    value.observe(on_sent)

[ ]: randomize = Button(description='Randomize')

def on_randomize(randomize):
```

The right pane shows a 3D visualization of a robot arm in a grid environment. The robot arm has a yellow horizontal bar, a purple vertical bar, and a dark blue spherical base. A control panel on the right side of the 3D view is titled 'Close Controls' and includes a 'Background' color picker set to [240, 240, 240] and a 'Robot Joints' section with sliders for 'shoulder', 'elbow', and 'wrist', each set to 0.

The bottom status bar of the JupyterLab interface shows 'Simple' mode, 'Python 3 (ipykernel) | Idle', and 'ROS' status. The bottom right corner of the interface indicates 'Mode: Command', 'Ln 1, Col 1', and the file name 'ROS_Joint_States.ipynb'.

JUPYTERLAB

JUPYTERHUB:ROS IN THE CLOUD

- Already widely adopted by universities
- Launch Jupyter Notebooks & software environments in the cloud / cluster
- SSO with uni user accounts
- ROS pre-installed and hassle free experience
- Completely free & open source

Pilot project with
RWTH Aachen to teach ROS



OUTLOOK

How to help

- Missing a package? Open a PR
- Help us maintain all of ROS in RoboStack

Future

- WebAssembly ROS2 in JupyterLite
- Feature parity of Jupyter extensions between ROS 1 and ROS 2
- Integrate webviz / Foxglove into JupyterLab



THANKS!

Questions?

gitter.im/RoboStack/Lobby



@RoboStack



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Stories