jupyterlab-urdf

Release 0.2.0

Isabel Paredes

Oct 11, 2022

CONTENTS:

With this JupyterLab extension, you can easily create and modify URDF files from the comfort of your web browser.

CHAPTER

ONE

TRY IT WITH JUPYTERLITE!

1.1 Installation

1.1.1 Requirements

- JupyterLab >= 3.0

For running the examples and interfacing with ROS, the following packages are also required:

- `ros-noetic-urdf-tutorial`
- `ros-noetic-turtlebot3-description`

These packages can be installed with `conda` from the RoboStack channel.

1.1.2 Install

To install the extension, execute:

1. `conda install jupyterlab-urdf -c conda-forge` or
2. `pip install jupyterlab-urdf` or
3. `npm i jupyterlab_urdf`

1.1.3 Uninstall

To remove the extension, execute:

1. `conda remove jupyterlab-urdf` or
2. `pip uninstall jupyterlab-urdf` or
3. `npm uninstall jupyterlab_urdf`

1.1.4 Development install

Note: You will need NodeJS to build the extension package.

The `jlpm` command is JupyterLab's pinned version of yarn that is installed with JupyterLab. You may use `yarn` or `npm` in lieu of `jlpm` below.

```
# Clone the repo to your local environment
# Change directory to the jupyterlab_urdf directory
# Install package in development mode
pip install -e .
# Link your development version of the extension with JupyterLab
jupyter labextension develop . --overwrite
# Rebuild extension Typescript source after making changes
jlpm build
```

You can watch the source directory and run JupyterLab at the same time in different terminals to watch for changes in the extension's source and automatically rebuild the extension.

```
# Watch the source directory in one terminal, automatically rebuilding when needed
jlpm watch
# Run JupyterLab in another terminal
jupyter lab
```

With the watch command running, every saved change will immediately be built locally and available in your running JupyterLab. Refresh JupyterLab to load the change in your browser (you may need to wait several seconds for the extension to be rebuilt).

By default, the `jlpm build` command generates the source maps for this extension to make it easier to debug using the browser dev tools. To also generate source maps for the JupyterLab core extensions, you can run the following command:

```
jupyter lab build --minimize=False
```

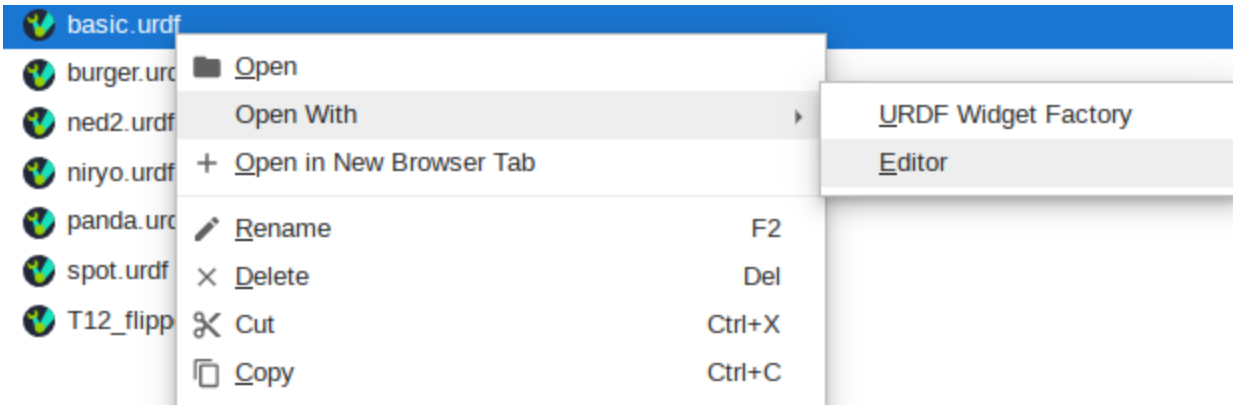
1.1.5 Development uninstall

```
pip uninstall jupyterlab_urdf
```

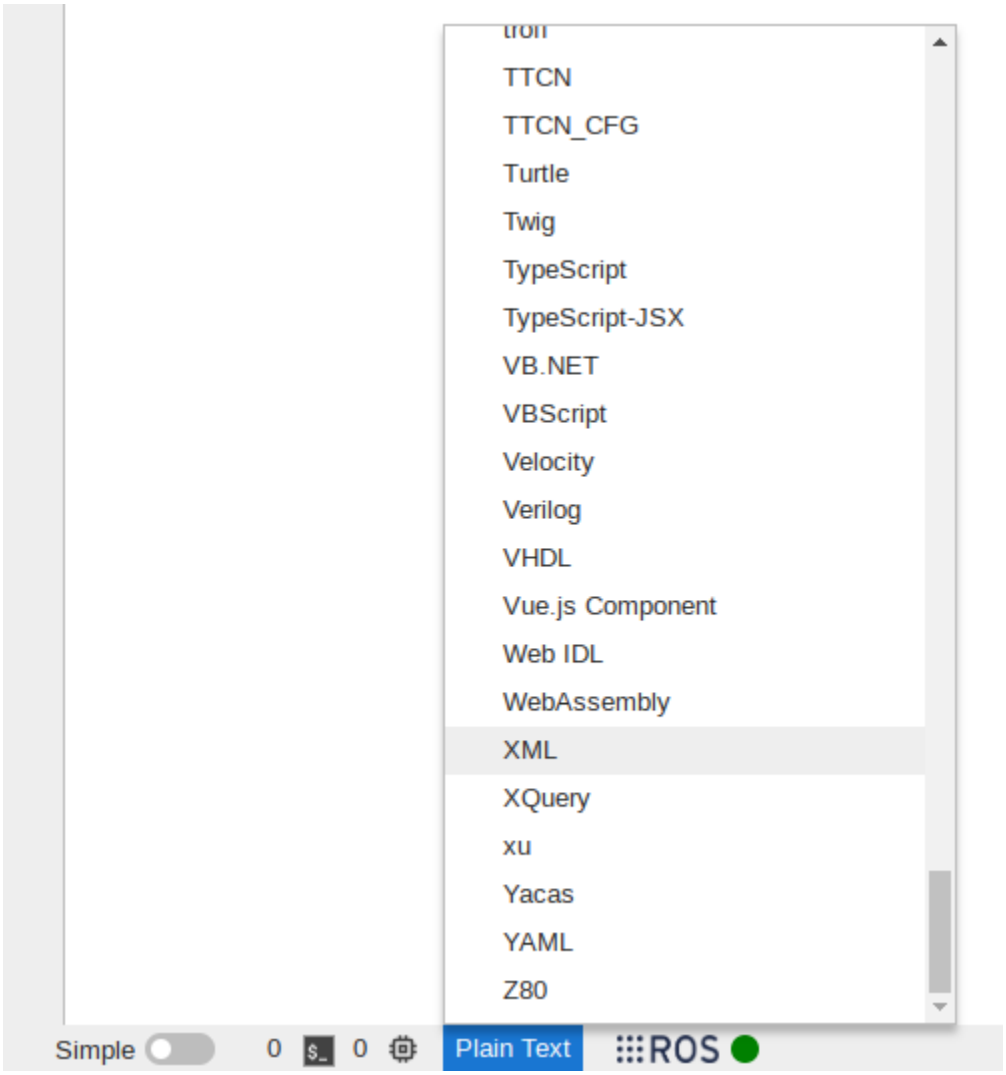
In development mode, you will also need to remove the symlink created by `jupyter labextension develop` command. To find its location, you can run `jupyter labextension list` to figure out where the `labextensions` folder is located. Then you can remove the symlink named `jupyterlab_urdf` within that folder.

1.2 URDF Editor

To open the text editor for any URDF file, simply right click on the file and select "Open With" > "Editor".



Once the file is open, syntax highlighting can be enabled by selecting "XML" from the language menu in the bottom left corner.



1.3 URDF Viewer

The simplest way to open the viewer is by double clicking on any already existing URDF file from the File Browser. New files can also be created from the top menu or from the launcher.

With the URDF editor open, files can be modified directly and the changes will be immediately reflected in the viewer.

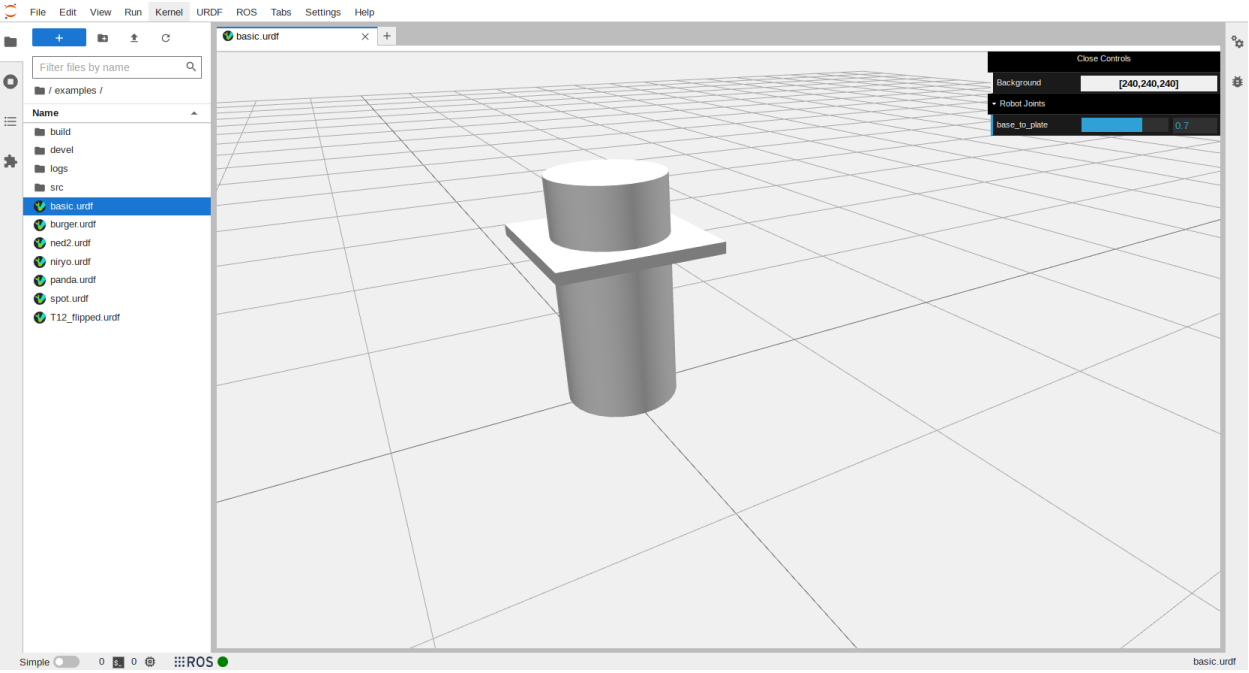
Note: if the URDF file contains too many errors or incomplete lines, the viewer will not be able to display the robot and thus the viewer will have to be reloaded.

The viewer provides a control panel to manually modify the joint positions. The changes are only reflected in the viewer and are not saved to the URDF file.

Only revolute and prismatic joints will be included in the panel, fixed joints are automatically ignored. If the joints have upper and lower limits, those will become the limits of the joint slider, otherwise a default range will appear.

1.4 Examples

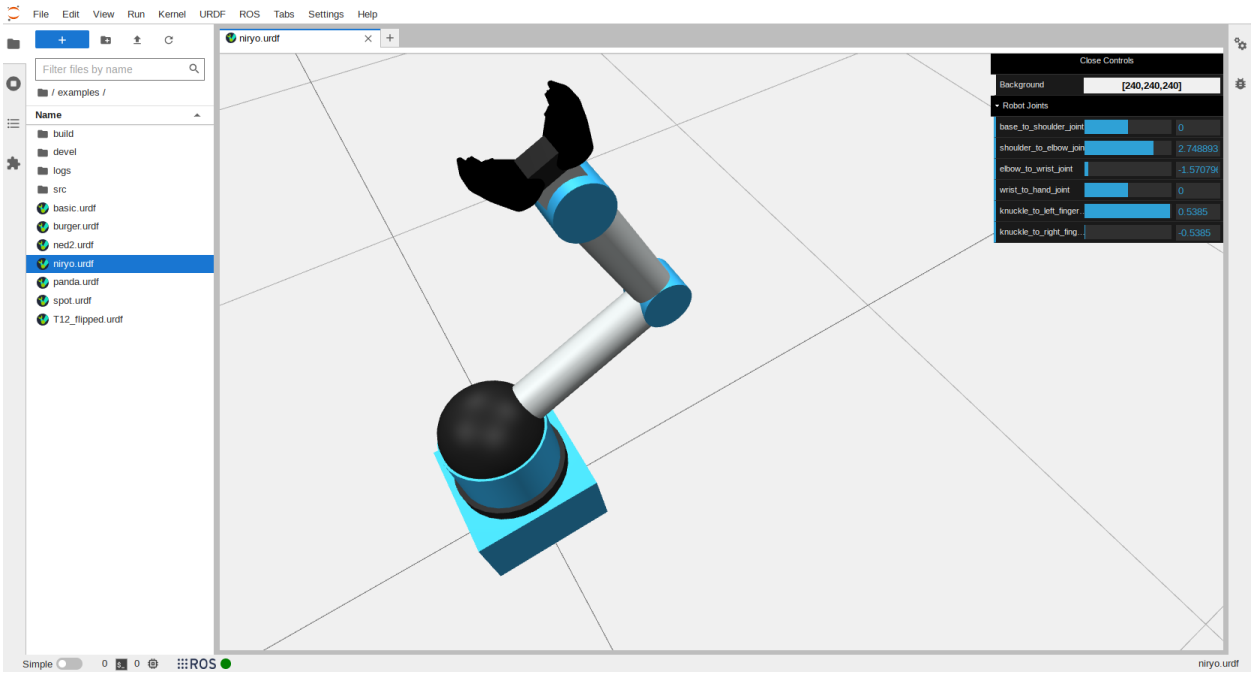
Without any configuration, the basic robot example can be displayed in the viewer. This robot is simply two geometric components with a single prismatic joint.

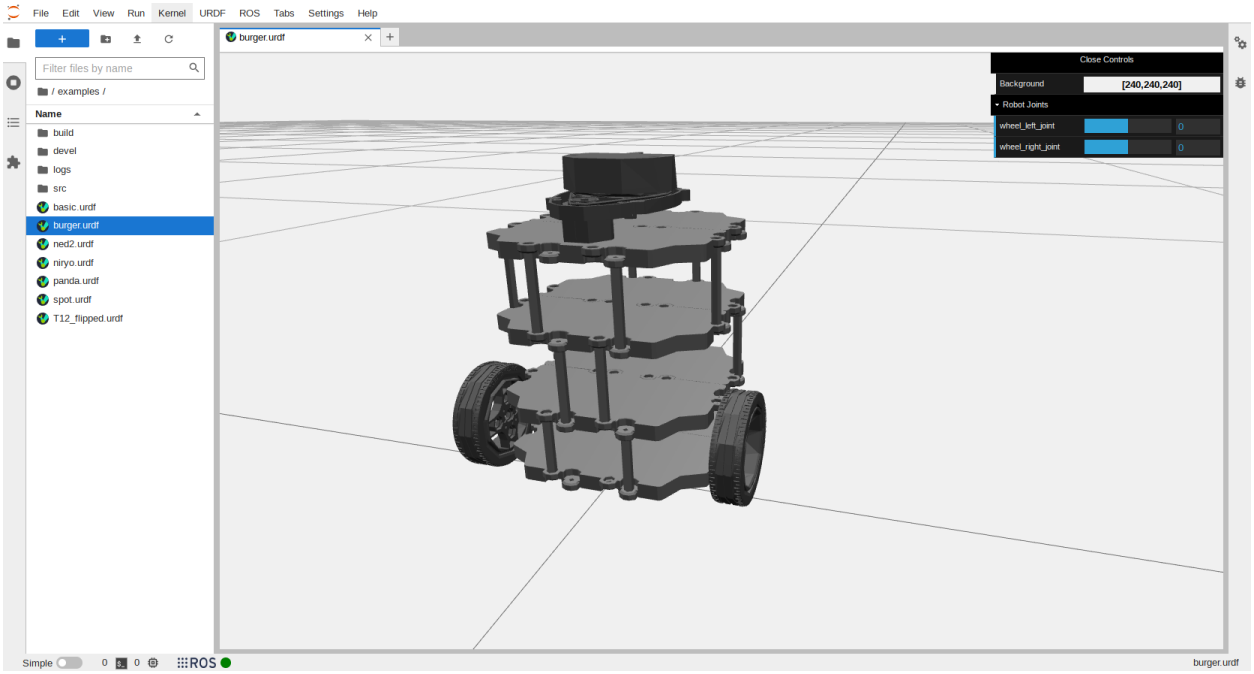


To display the basic Niryo and the Burger robot, a ROS environment needs to be configured accordingly in order for the viewer to locate the necessary mesh files, see Tutorial for Configuring ROS Environment.

The following packages are required:

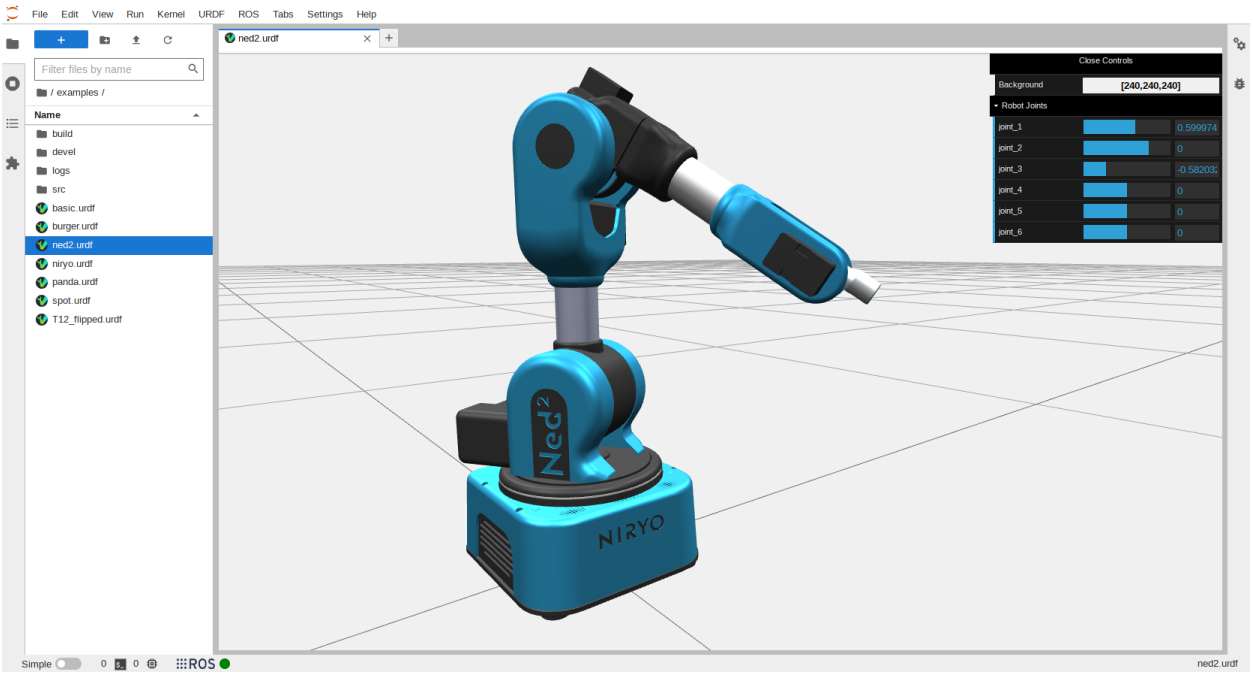
- `ros-noetic-urdf-tutorial`
- `ros-noetic-turtlebot3-description`

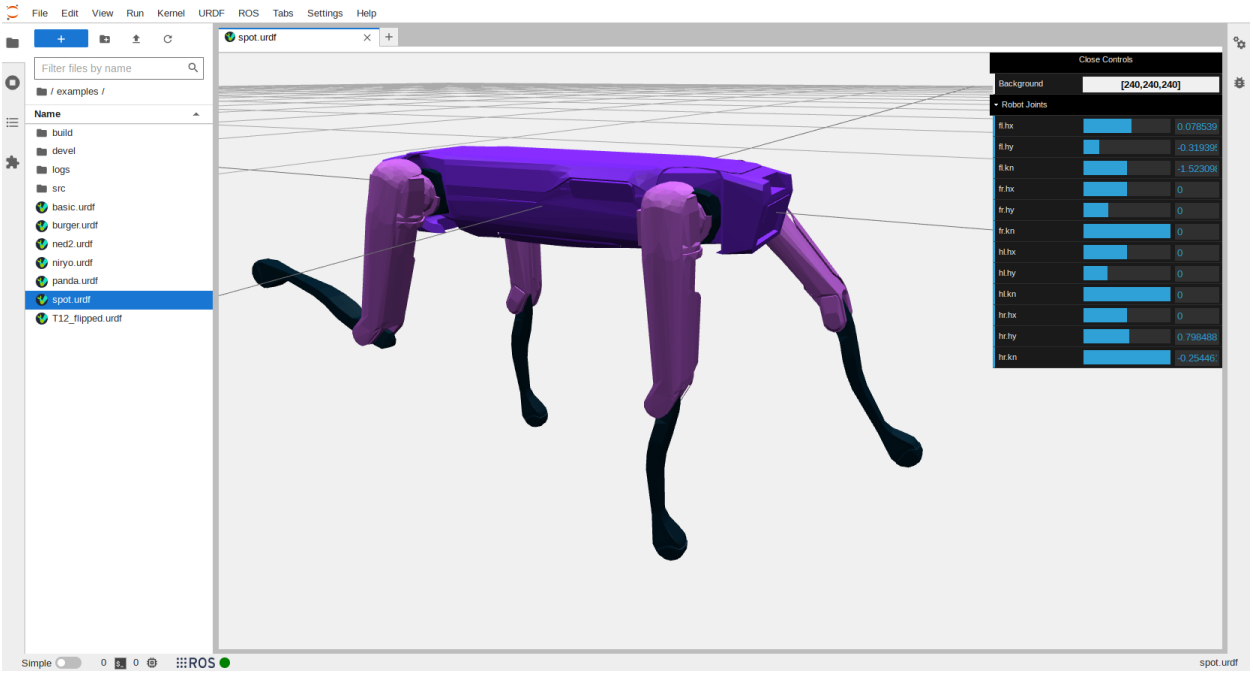


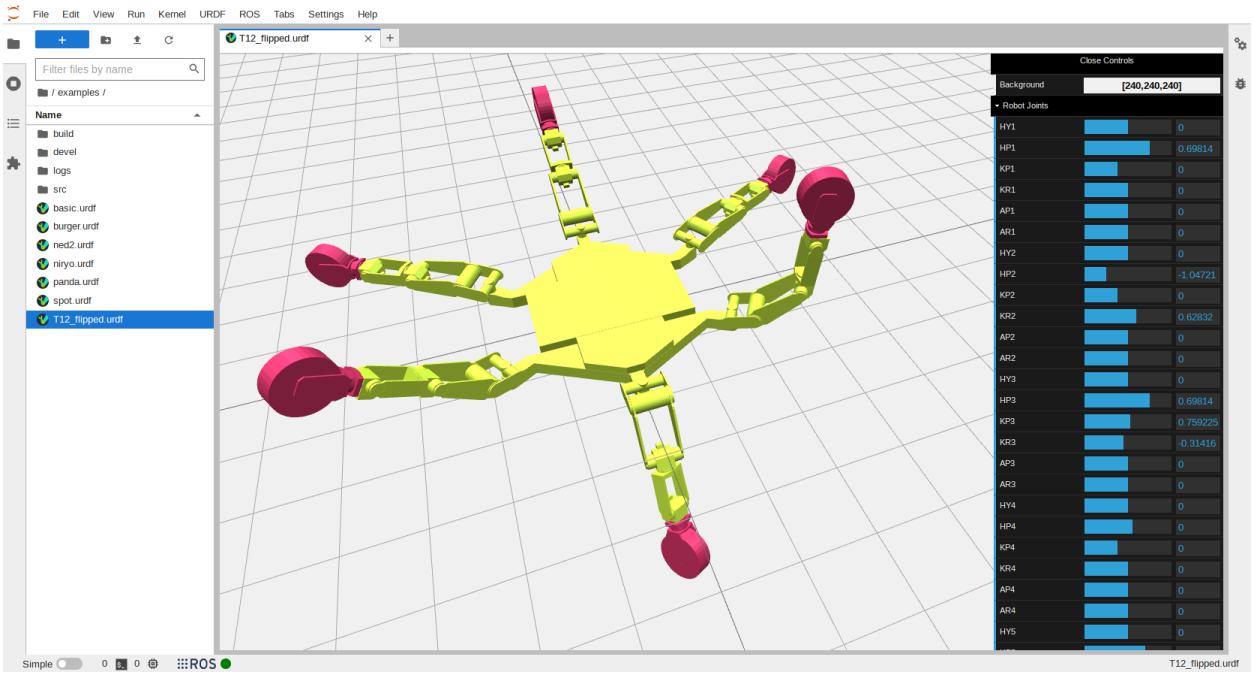


Similarly, Ned, Spot and the T12 robot use robot description ROS packages to store their respective mesh files. These packages will need to be installed locally and the ROS environment configured correctly (`source <workspace>/devel/setup.bash`) so that these packages can be accessed.

- `ned_description`
- `spot_description`
- `t12_description`


File Edit View Run Kernel URDF ROS Tabs Settings Help
Filter files by name
/ examples /
Name
build
devel
logs
src
basic.urdf
burger.urdf
ned2.urdf
niryo.urdf
panda.urdf
spot.urdf
T12_flipped.urdf
ned2.urdf
Close Controls
Background
[240,240,240]
Robot Joints
joint_1
joint_2
joint_3
joint_4
joint_5
joint_6
Simple
ROS
ned2.urdf


File Edit View Run Kernel URDF ROS Tabs Settings Help
Filter files by name
/ examples /
Name
build
devel
logs
src
basic.urdf
burger.urdf
ned2.urdf
niryo.urdf
panda.urdf
spot.urdf
T12_flipped.urdf
spot.urdf
Close Controls
Background
[240,240,240]
Robot Joints
Simple
ROS
spot.urdf


File Edit View Run Kernel URDF ROS Tabs Settings Help
Filter files by name
/ examples /
Name
build
devel
logs
src
basic.urdf
burger.urdf
ned2.urdf
niryo.urdf
panda.urdf
spot.urdf
T12_flipped.urdf
T12_flipped.urdf
Close Controls
Background
[240,240,240]
Robot Joints
HY1 0
HP1 0.69814
KP1 0
KR1 0
AP1 0
AR1 0
HY2 0
HP2 -1.04721
KP2 0
KR2 0.62832
AP2 0
AR2 0
HY3 0
HP3 0.69814
KP3 0.759225
KR3 -0.31416
AP3 0
AR3 0
HY4 0
HP4 0
KP4 0
KR4 0
AP4 0
AR4 0
HY5 0
Simple
ROS
T12_flipped.urdf

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search