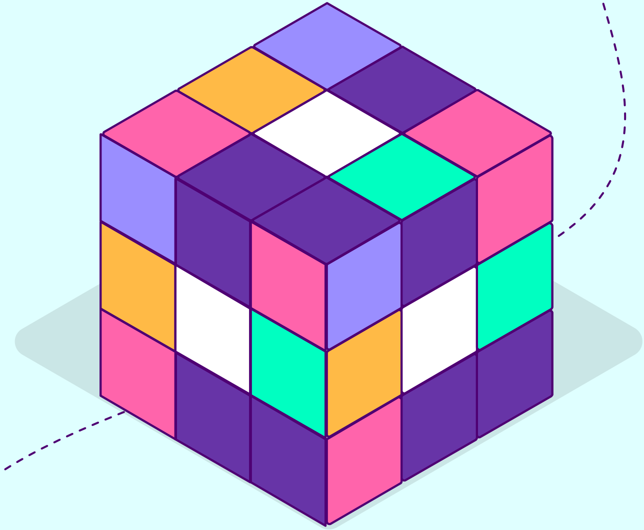
Visual Programming in JupyterLab with Blockly



PyData

DenisaCG

DenisaCheciu

Denisa Checiu

Robotics Software Developer

Bachelor student @Jacobs University Bremen

Passionate about Machine Learning, A.I. & Computer Vision

QuantStack
Scientific Computing

TABLE OF CONTENTS

01

Visual Programming

Why and when do we need it?

02

JupyterLab-Blockly

Extension for block-based programming

03

Base for other extensions

Diving deeper into how you can build on top of it

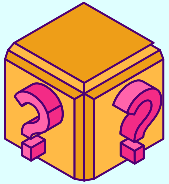
04

Robotics Applications

Live demo for JupyterLab-Niryo & JupyterLab-Lego-Boost

Visual Programming

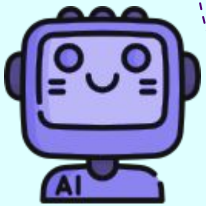
Programming without having to deal with specific syntax



Learn fundamental programming concepts faster

Important part of early computer science education

Provide a smoother ramp for learners in the Jupyter ecosystem

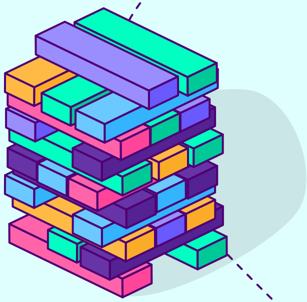


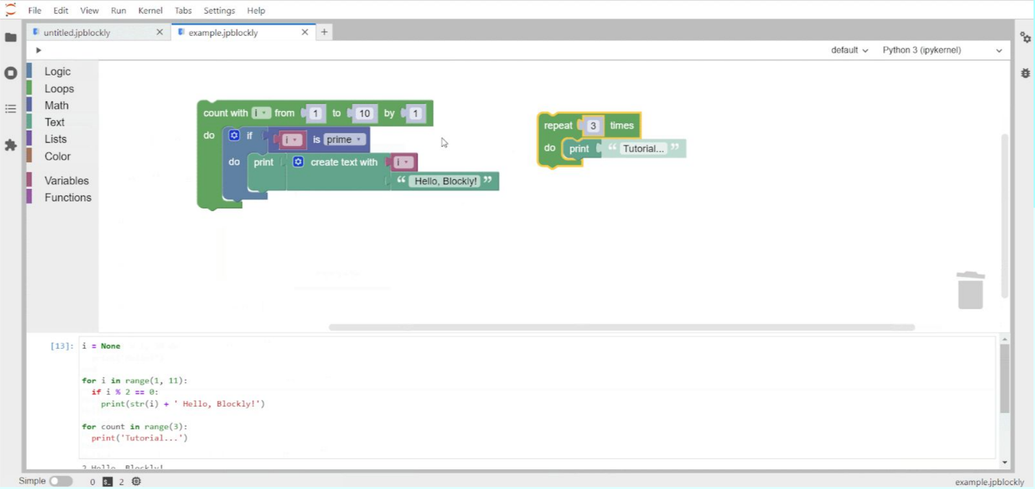
Great tool for robotics prototyping

JupyterLab-Blockly

Blockly

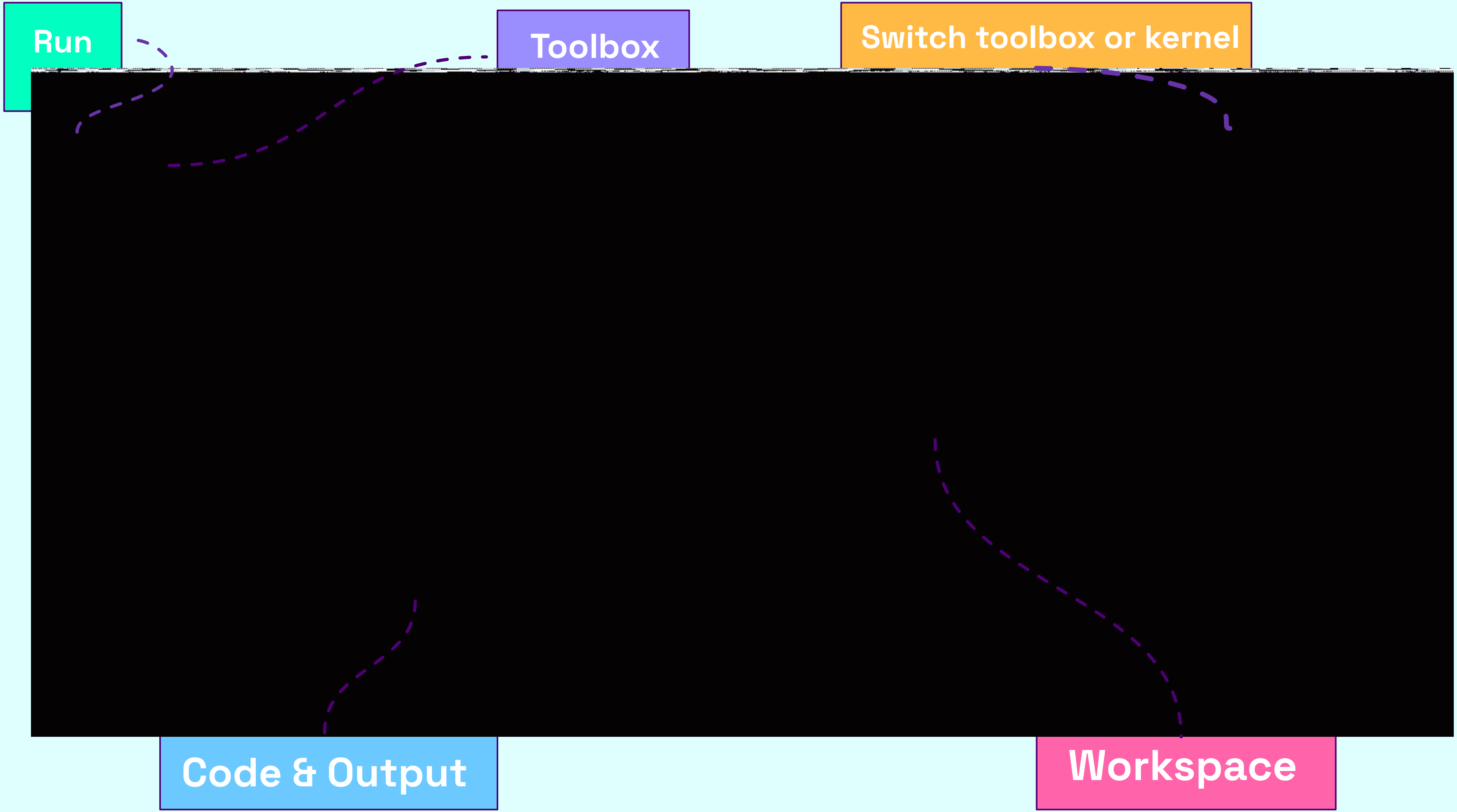
An open source library designed by Google to make coding easier and more accessible through block-based visual programming.





```
[13]: i = None

      for i in range(1, 11):
        if i % 2 == 0:
          print(str(i) + ' Hello, Blockly!')

      for count in range(3):
        print('Tutorial...')

      2 Hello, Blockly!
```

A JupyterLab extension that uses **interlocking graphical blocks** to represent coding concepts, while giving the user full creative freedom, it removes all language specific syntax requirements.



Run

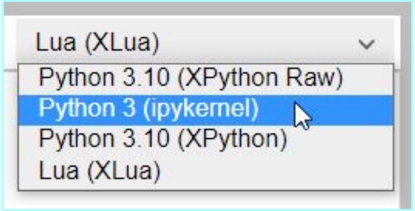
Toolbox

Switch toolbox or kernel

Code & Output

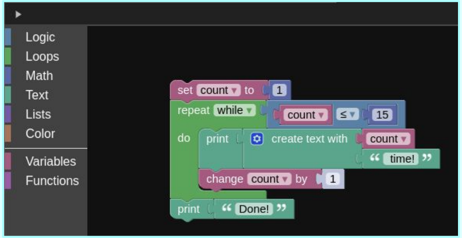
Workspace

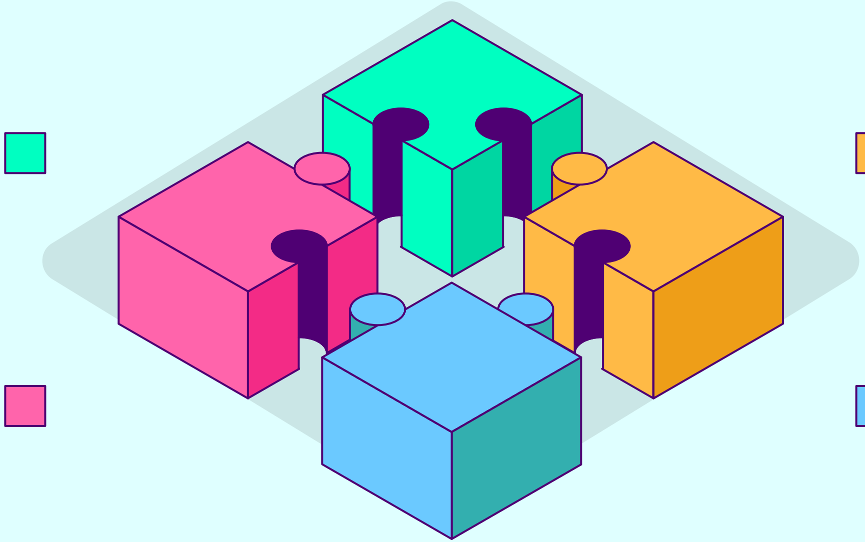
Smooth Jupyter Integration



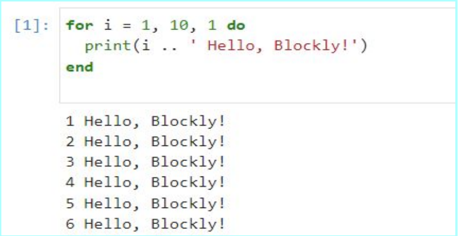
Use **Jupyter kernels** to execute the generated code 🟩

Modify its colors based on your individual **theme** (dark, light or personalized) 🟥





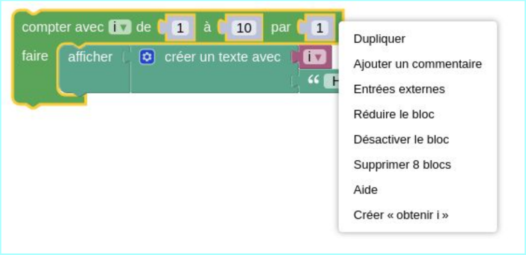
```
mamba install -c conda-forge jupyterlab-blockly
```



```
[1]: for i = 1, 10, 1 do
       print(i .. ' Hello, Blockly!')
     end

     1 Hello, Blockly!
     2 Hello, Blockly!
     3 Hello, Blockly!
     4 Hello, Blockly!
     5 Hello, Blockly!
     6 Hello, Blockly!
```

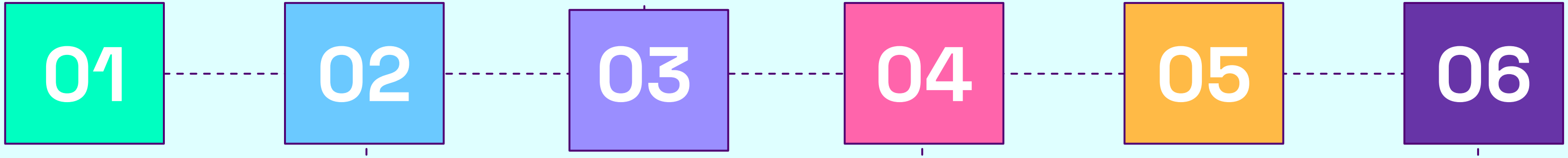
🟧 Reuse the JupyterLab **code cell** component to display the generated code

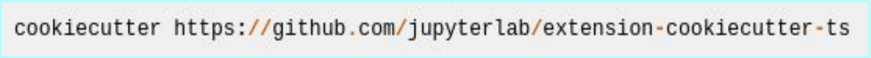
🟦 Support **translations and localization** (blocks and workspace)



jupyterlab-blockly.readthedocs.io

STEP BY STEP how to build on top of JupyterLab-Blockly

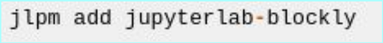




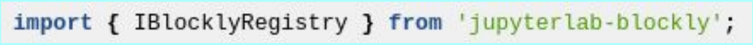
```
cookiecutter https://github.com/jupyterlab/extension-cookiecutter-ts
```

Create JL extension from template

Import JupyterLab-Blockly



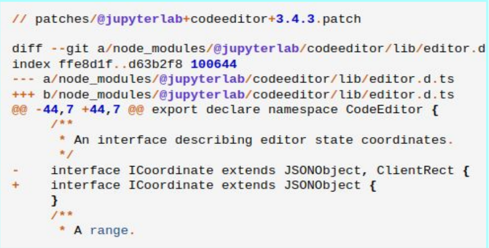
```
jlpm add jupyterlab-blockly
```



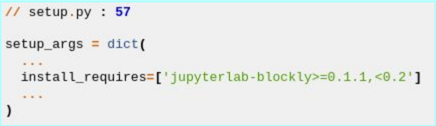
```
import { IBlocklyRegistry } from 'jupyterlab-blockly';
```

Register new blocks, toolboxes, generators

Include patches



```
// patches/@jupyterlab+codeeditor+3.4.3.patch

diff --git a/node_modules/@jupyterlab/codeeditor/lib/editor.d
index ffe8d1f..d63b2f8 100644
--- a/node_modules/@jupyterlab/codeeditor/lib/editor.d.ts
+++ b/node_modules/@jupyterlab/codeeditor/lib/editor.d.ts
@@ -44,7 +44,7 @@ export declare namespace CodeEditor {
     /**
      * An interface describing editor state coordinates.
      */
-    interface ICoordinate extends JSONObject, ClientRect {
+    interface ICoordinate extends JSONObject {
     }
     /**
      * A range.
```



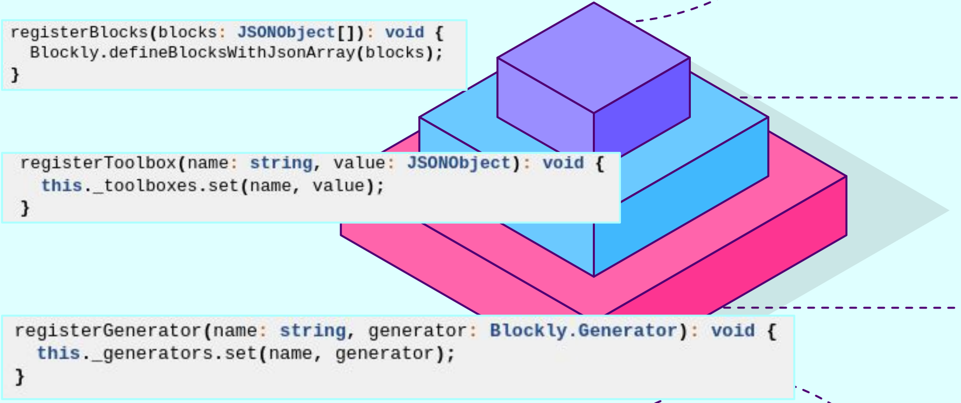
```
// setup.py : 57

setup_args = dict(
  ...
  install_requires=['jupyterlab-blockly>=0.1.1,<0.2']
  ...
)
```

Additional configurations

Extra! Use conda to package the extension for easier installation

Importing the IBlocklyRegistry

The class that the JupyterLab-Blockly extension exposes to other plugins, it allows other plugins to register new Toolboxes, Blocks and Generators that users can use in the Blockly editor.



```
registerBlocks(blocks: JSONObject[]): void {
  Blockly.defineBlocksWithJsonArray(blocks);
}
```

```
registerToolbox(name: string, value: JSONObject): void {
  this._toolboxes.set(name, value);
}
```

```
registerGenerator(name: string, generator: Blockly.Generator): void {
  this._generators.set(name, generator);
}
```

01

Register Blocks

Using their JSON definition and generator code in all supported programming languages

02

Register Toolboxes

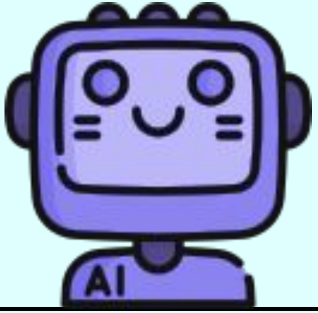
Once registered, the toolbox will appear automatically in your Blockly editor

03

Register Generators

Once registered you can easily switch to it in the Blockly editor

NIRYO
HUMAN - MOTION - ROBOT

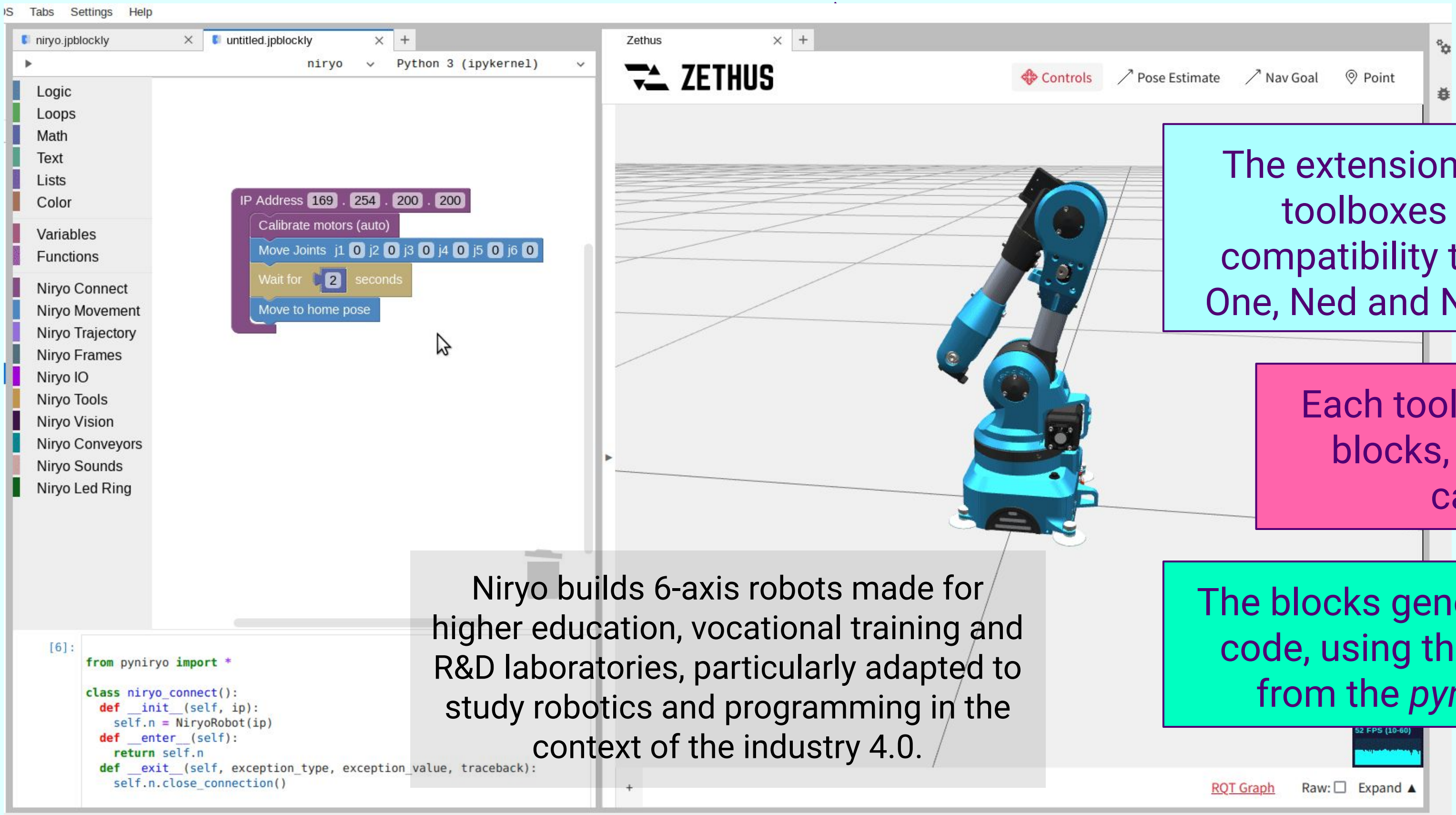


LEGO BOOST
BUILD CODE PLAY

Now that you saw how to build on top of JupyterLab-Blockly, let's take a look at the robotics applications we created!

JupyterLab-Niryo-One

NIRYO
HUMAN - MOTION - ROBOT



[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()
```

The extension offers two toolboxes and full compatibility to the Niryo One, Ned and Ned2 robots.

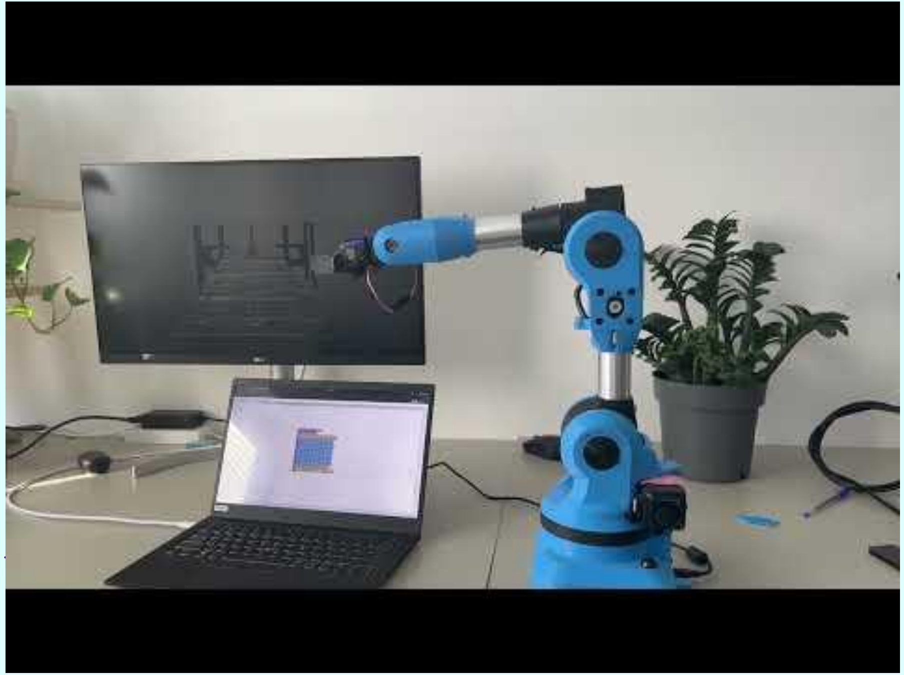
Each toolbox contains 130 blocks, organized in 10 categories.

The blocks generate Python code, using the functions from the *pyniryo* API.

Niryo builds 6-axis robots made for higher education, vocational training and R&D laboratories, particularly adapted to study robotics and programming in the context of the industry 4.0.

Thank you for Zethus configuration to Isabel Paredes

JupyterLab-Niryo-One

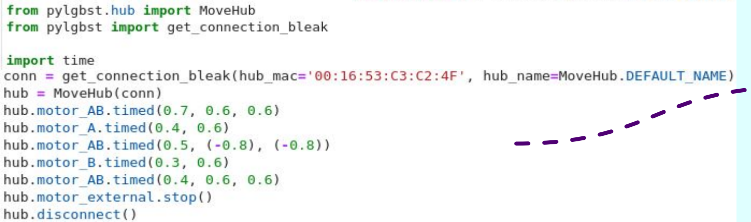


JupyterLab-Lego-Boost

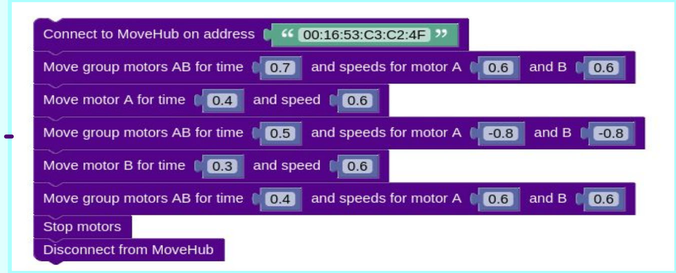
Live Demo!

Say hi to Vernie!

- Communicate with the MoveHub (a bluetooth hardware piece)
- Pass commands through Bluetooth Low Energy (BLE) wireless protocol
- Use blocks which generate Python code with the use of the *pylgbst* library



```python
from pylgbst.hub import MoveHub
from pylgbst import get_connection_bleak

import time
conn = get_connection_bleak(hub_mac='00:16:53:C3:C2:4F', hub_name=MoveHub.DEFAULT_NAME)
hub = MoveHub(conn)
hub.motor_AB.timed(0.7, 0.6, 0.6)
hub.motor_A.timed(0.4, 0.6)
hub.motor_AB.timed(0.5, (-0.8), (-0.8))
hub.motor_B.timed(0.3, 0.6)
hub.motor_AB.timed(0.4, 0.6, 0.6)
hub.motor_external.stop()
hub.disconnect()
```

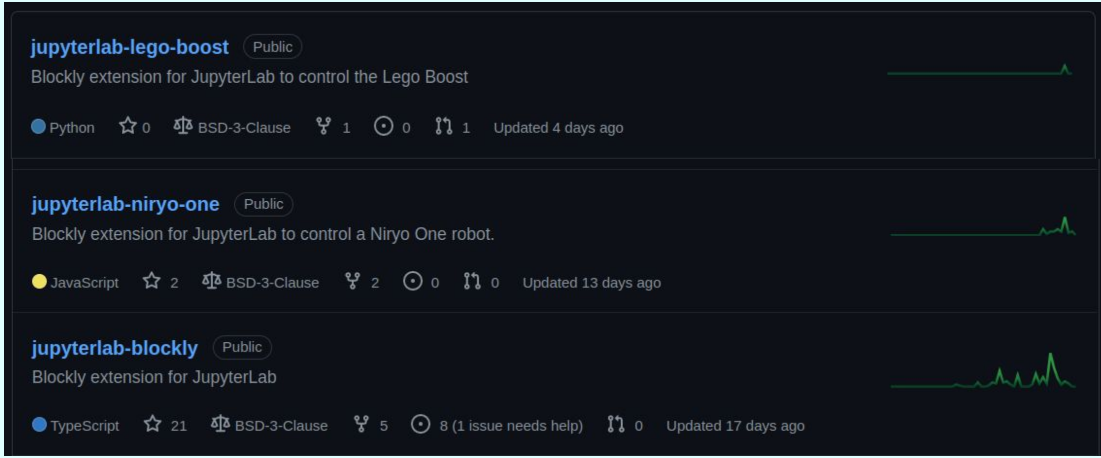


Connect to MoveHub on address “ 00:16:53:C3:C2:4F ”
Move group motors AB for time 0.7 and speeds for motor A 0.6 and B 0.6
Move motor A for time 0.4 and speed 0.6
Move group motors AB for time 0.5 and speeds for motor A -0.8 and B -0.8
Move motor B for time 0.3 and speed 0.6
Move group motors AB for time 0.4 and speeds for motor A 0.6 and B 0.6
Stop motors
Disconnect from MoveHub

All source code can be found on the **QuantStack Github** page - feel free to give it a try and let us know what you think!

A special thank you to
Carlos Herrero!

hbcarlos
carlosherrerob



jupyterlab-lego-boost Public
Blockly extension for JupyterLab to control the Lego Boost
● Python ☆ 0 BSD-3-Clause 1 0 1 Updated 4 days ago

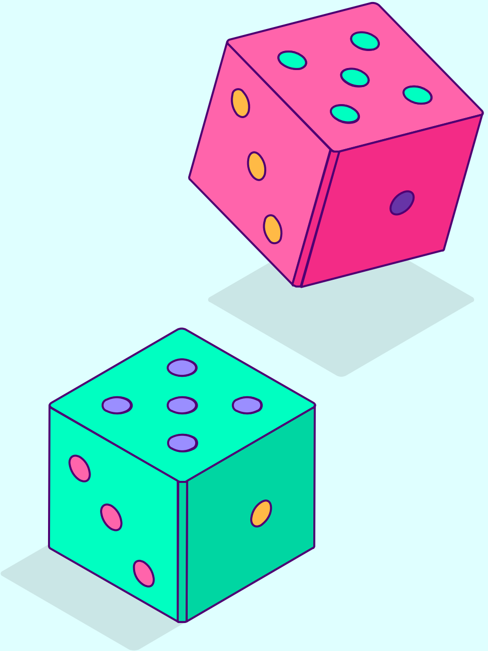
jupyterlab-niryo-one Public
Blockly extension for JupyterLab to control a Niryo One robot.
● JavaScript ☆ 2 BSD-3-Clause 2 0 0 Updated 13 days ago

jupyterlab-blockly Public
Blockly extension for JupyterLab
● TypeScript ☆ 21 BSD-3-Clause 5 8 (1 issue needs help) 0 Updated 17 days ago

THANK YOU!

QuantStack
Scientific Computing

PyData



Presentation template from Slidesgo.