



Python Productivity for Zynq



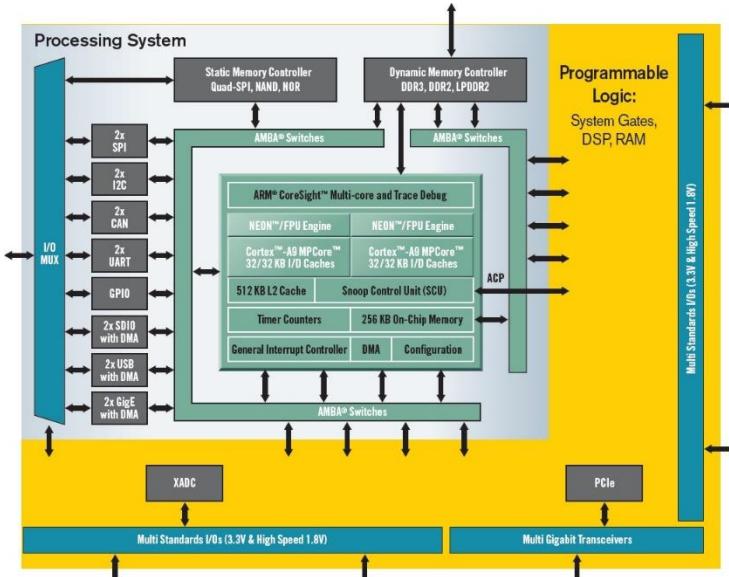
Outline

- > Zynq & Zynq Ultrascale+
- > PYNQ Framework
- > Technologies
- > Community

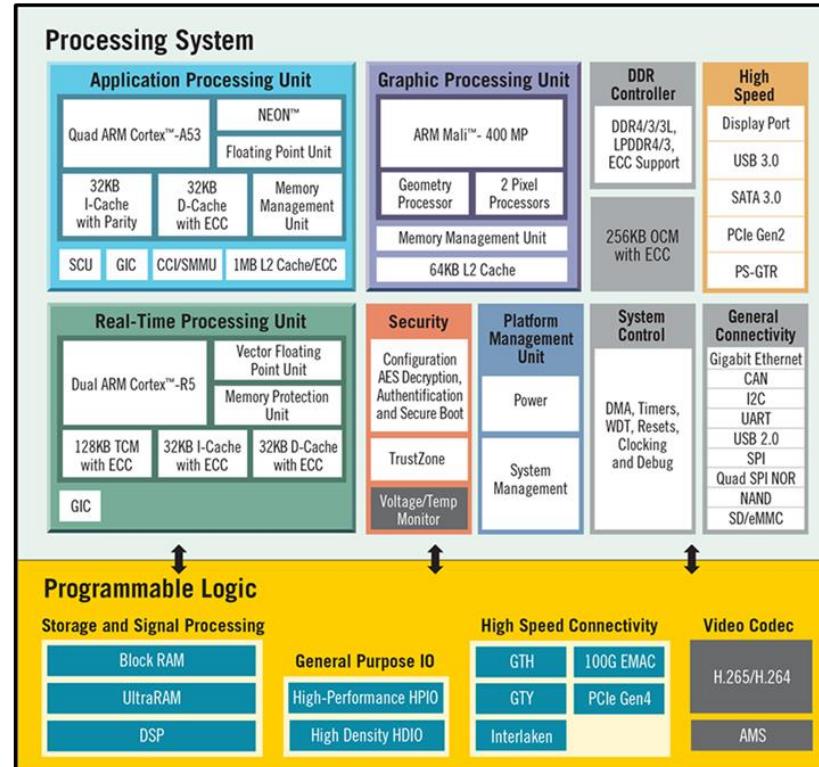
ZYNQ and ZYNQ UltraSCALE+

Best-in-class, All Programmable SoCs

ZYNQ 7000



ZYNQ UltraSCALE+



FPGAs and tightly-integrated CPUs enable entirely new opportunities

Zynq applications

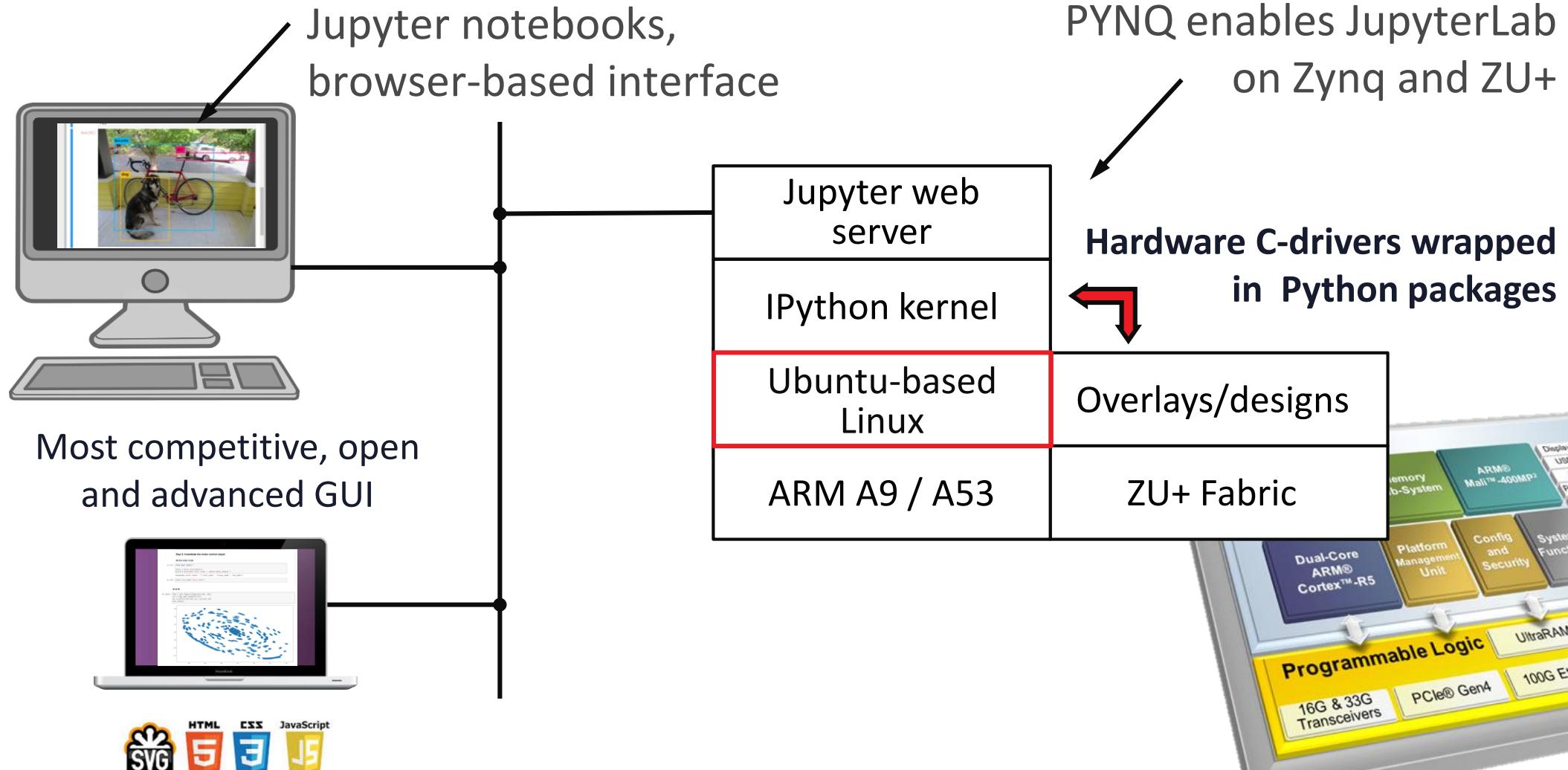


- > Make Zynq so easy-to-use that programmers can access the benefits of FPGAs without learning advanced digital design skills



PYNQ framework





Ubuntu-based Linux versus embedded Linux

Ubuntu-based Linux

➤ **Optimized for developer productivity**



- > All the Linux libraries and drivers you expect
 - > Pre-built SD card image
 - > Ubuntu/Debian ecosystem & community
- >>145,000,000 Google hits

3 orders of magnitude difference

Embedded Linux



➤ **Optimized for deployment efficiency**

- > Selective Linux libraries and drivers
 - > Commonly delivered in flash memory on board
 - > PetaLinux ecosystem:
- >> 143,000 Google hits

PYNQ's Ubuntu-based Linux

PYNQ uses Ubuntu's:

- Root file system (RFS)
- Package manager (*apt-get*)
- Repositories

PYNQ bundles :

- Development tools
 - Cross-compilers
- Latest Python packages



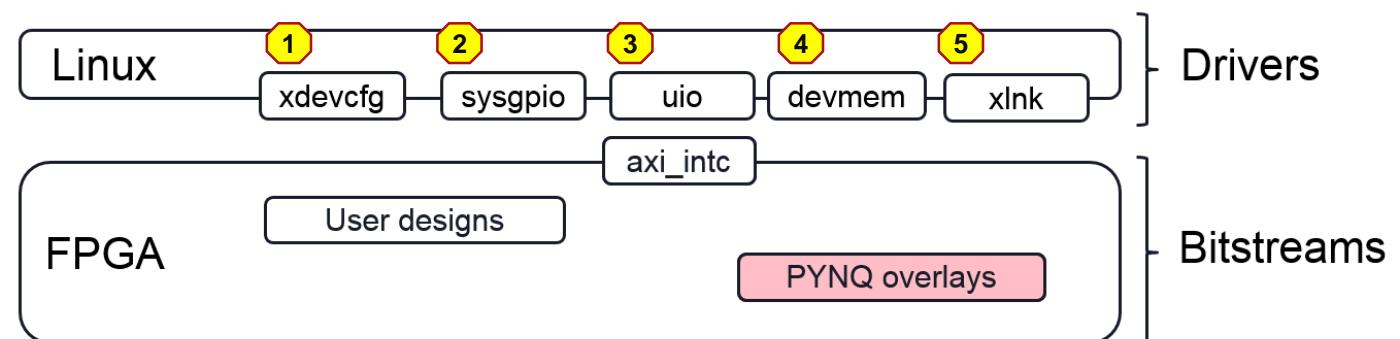
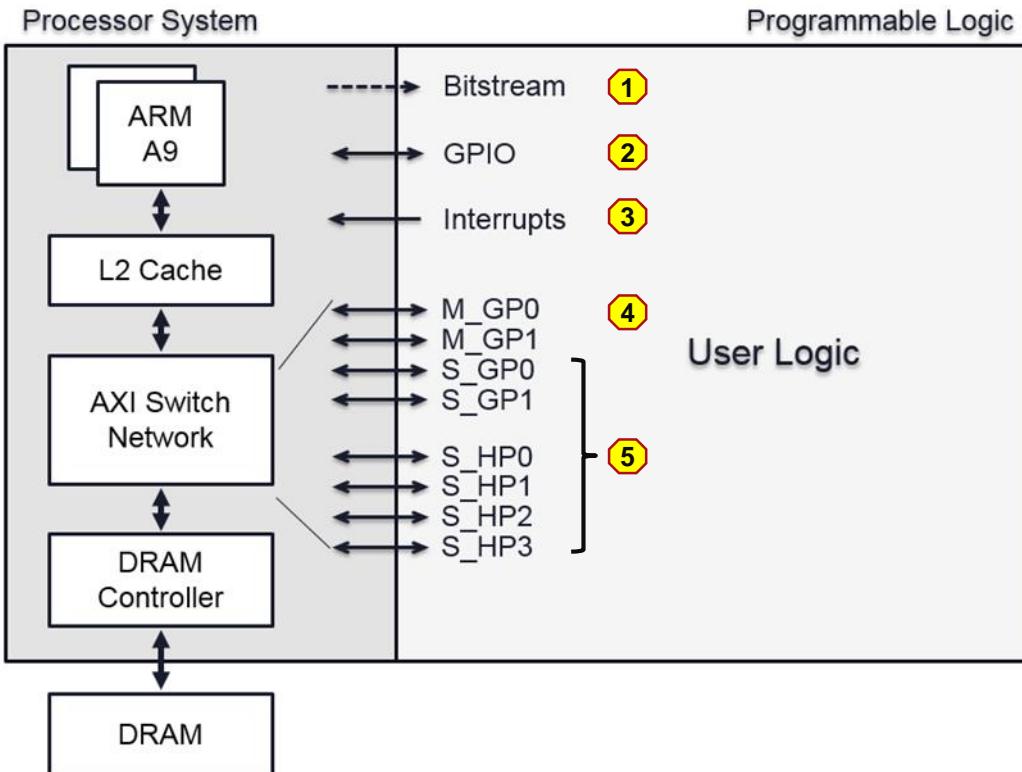
PYNQ uses the PetaLinux build flow and board support package:

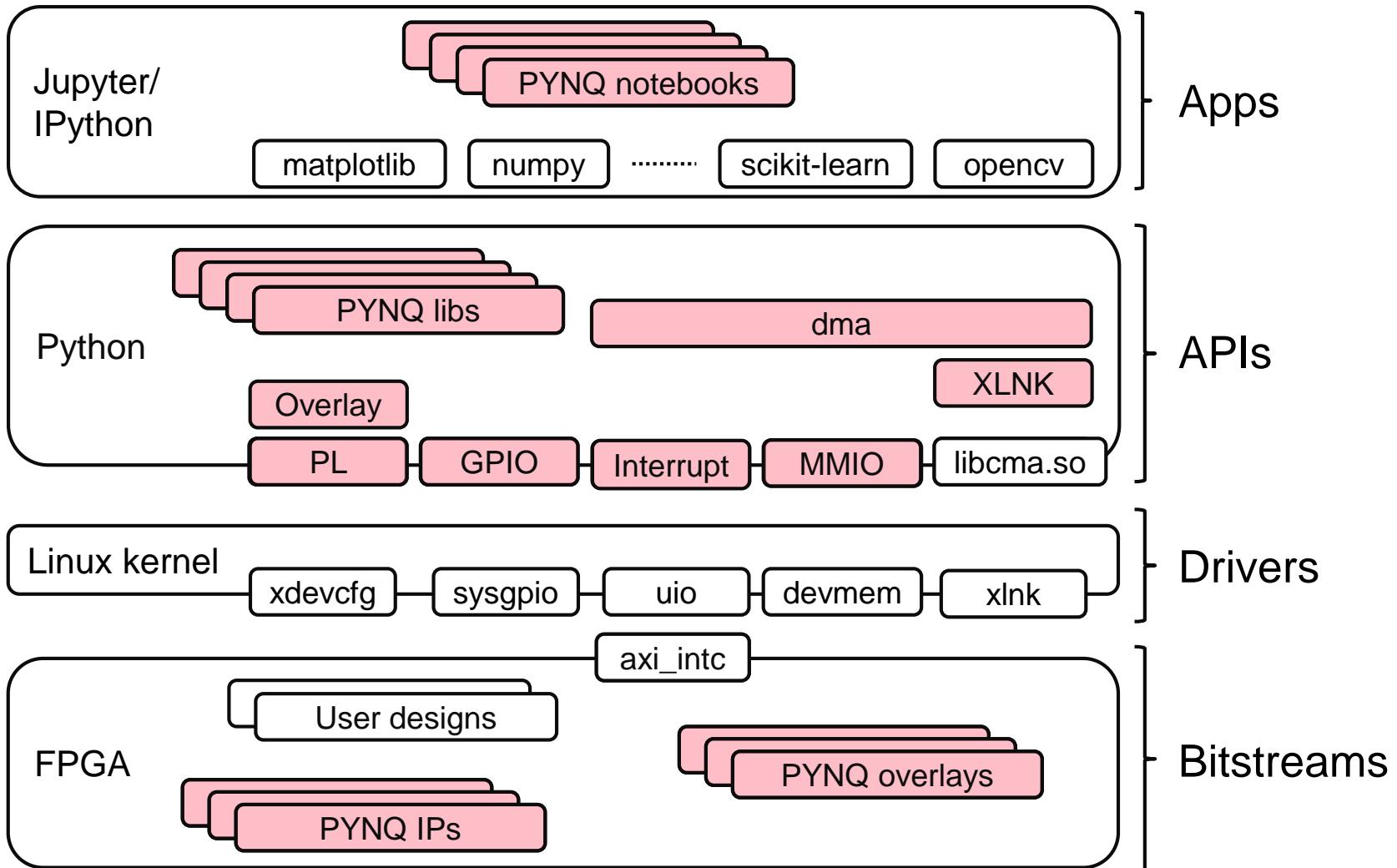
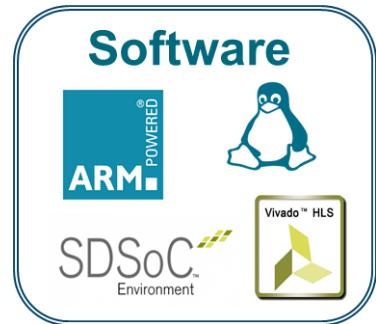
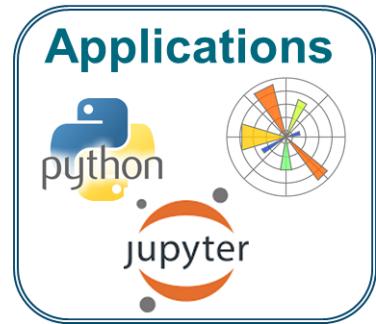
- Access to all Xilinx kernel patches
- Works with any Xilinx supported board
- Configured with additional drivers for PS-PL interfaces

PYNQ provides Linux Drivers for PS-PL Interfaces ...

wrapped in Python Libraries

Zynq





Vivado Design Metadata available from Python

- > **PYNQ passes the Vivado metadata file to the target platform**
 - >> Initially Vivado TCL file
 - >> Now moving to newer DSA file (which replaces the TCL file)
- > **It then parses the Vivado metadata file to:**
 - >> Set Zynq clock frequencies automatically
 - >> Assign memory-map attributes for every IP
 - >> Assign default MMIO drivers to IP, when no drivers are specified
- > **Creates a Python dictionary for the IP in the bitstream from the metadata file**
 - >> Enables bitstream metadata to be queried and modified in Python at runtime

Hybrid Packages

- > New *hybrid packages* are created by extending Python packages with additional files:
 - » Design Bitstream
 - » Design metadata file
 - » C drivers
 - » Jupyter notebooks
- > Hybrid packages enable software-style packaging and distribution of designs
- > Use the Python package installer, PIP to install a hybrid package just like any regular Python (software only) package
 - » Delivers package's files to target board
- > Uses Python standard setup.py script for installation

Software-style Packaging & Distribution of Designs

Enabled by new *hybrid packages*

The figure displays four GitHub repository pages for Xilinx projects, each featuring a Jupyter notebook:

- Xilinx / QNN-MO-PYNQ**: A notebook titled "3. Open image to be classified" showing code to load and preprocess an image of a beagle dog.
- Xilinx / IoT-SPYN**: A notebook titled "SPYN - III phase AC motor control" demonstrating the control of a 3-phase AC motor using an EDPS board.
- Xilinx / PYNQ-DL**: A notebook titled "Resizing an image" illustrating how to run a kernel for resizing an image on the FPGA.
- Xilinx / PYNQ-ComputerVision**: A notebook titled "OpenCV Overlay: Filter2D and Dilate" showing the use of OpenCV overlays on a PYNQ-Z1 board.

Each screenshot shows the GitHub interface with code snippets, output cells, and associated images or plots.

Download a design from GitHub with a single Python command:

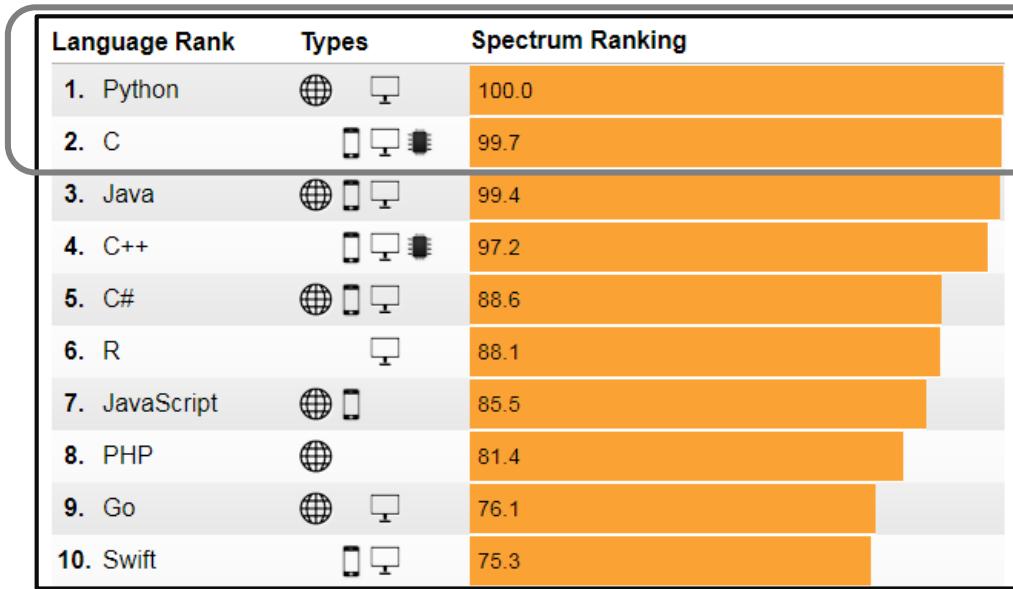
```
pip3 install git+https://github.com/Xilinx/pynqDL.git
```

PYNQ enabling technologies

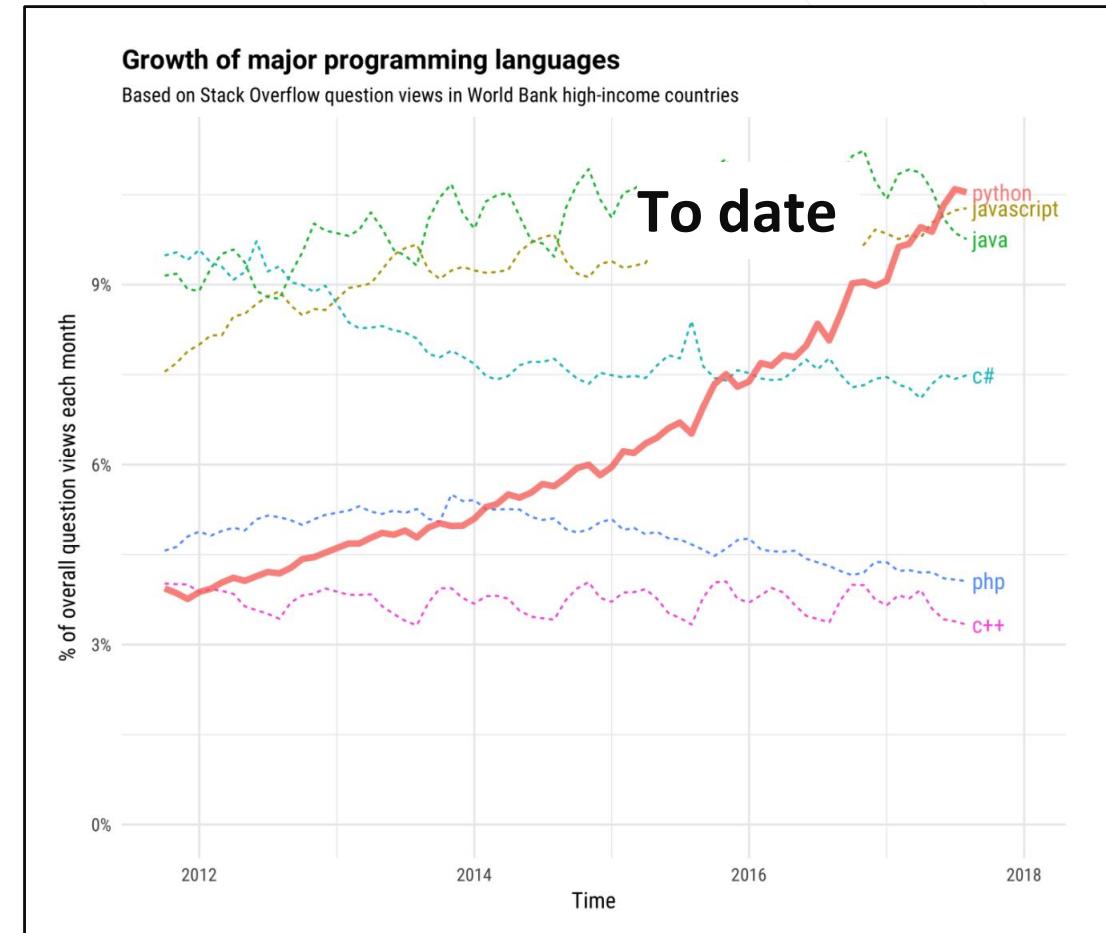


Python is increasingly the language of choice

Top Programming Languages,
IEEE Spectrum, July'17



<https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>



<https://stackoverflow.blog/2017/09/06/incredible-growth-python/>

Python is the fastest growing language: driven by data science, AI, ML and academia

Ecosystem advantage: there's a Python library for that...

135,734 projects 944,172 releases 1,257,368 files 273,938 users



The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages. Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

C₀Python is written in C ... and most popular C/C++ frameworks have Python libraries

Jupyter Notebooks ... the engine of data science

The screenshot shows a Jupyter Notebook interface with several code cells and their outputs.

Code Cells:

- In [6]:** Python code for executing the last convolutional layer in Python. It includes timing and memory usage calculations.
- In [7]:** Python code for drawing detection boxes using Darknet. It involves loading a network, processing probabilities, and drawing bounding boxes on an image.
- In [8]:** Python code for showing the result, which displays an image of a dog sitting next to a bicycle with a car in the background. Bounding boxes are drawn around the objects, labeled as "bicycle", "car", and "dog".



Open source browser-based, executable documents

Live code, text, multimedia, graphics, equations, widgets ...

1.7 million notebooks on GitHub

Taught to 1,000+ Berkeley data science students

JupyterLab: web-based IDE incl. Notebooks

This screenshot shows the JupyterLab interface. On the left is a file browser with a tree view of files and folders. In the center is a terminal window showing Python code and its execution results. The right side shows a Python 3 notebook cell containing code and plots.

```
?      -> Introduction and overview of IPython's features.
?quickref -> Quick reference.
help     -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: %matplotlib inline
from numpy.random import beta
import matplotlib.pyplot as plt
plt.style.use('bmh')

def plot_beta_hist(a, b):
    plt.hist(beta(a, b, size=10000), histtype="stepfilled",
            bins=25, alpha=0.8, normed=True)
    return

plot_beta_hist(10, 10)
plot_beta_hist(4, 12)
plot_beta_hist(50, 12)
plot_beta_hist(6, 55)

In [2]: %run ~/Downloads/mri_with_eeg.py
loading eeg /Users/fperez/usr/conda/lib/python3.5/site-packages/matplotlib/mpl-data/sample_data/eeg.dat

```

The terminal output shows:

```
1  [██████████] 18.1% Tasks: 305 total, 1 running
2  [██████████] 5.0% Load average: 2.29 2.07 2.09
3  [██████████] 15.6% Uptime: 4 days, 21:59:11
4  [██████████] 5.0%
5  [██████████] Mem: 19987/81920B
6  [██████████] Swap: 2487/38720B
```

The notebook cell output shows:

```
In [1]:
```

```
1 #!/usr/bin/env python
2 """
3 This now uses the imshow command instead of pcolor which *is* much
4 faster.
5 """
6 from __future__ import division, print_function
7
8 import numpy as np
9
10 from matplotlib import *
11 from matplotlib.collections import LineCollection
12 import matplotlib.cbook as cbook
13 import matplotlib.cm as cm
14 # I use it to break up the different regions of code visually
15
16 if 1: # load the data
17     # data are 256x256 16 bit integers
18     dfile = cbook.get_sample_data('s1045 ima.gz')
19     im = np.fromstring(dfile.read(), np.uint16).astype(float)
20     im.shape = 256, 256
21
22 if 1: # plot the MRI in pcolor
23     subplot(221)
24     imshow(im, cmap=cm.gray)
25     axis('off')
26
27 if 1: # plot the histogram of MRI intensity
28     subplot(222)
29     im = np.ravel(im)
30     im = im[im.nonzero()] # ignore the background
31     im = np.log2(im/15) # normalize
32     hist(im, 100)
33     xticks([-1, -0.5, 0, 0.5, 1])
34     yticks([1])
35     xlabel('Intensity')
36     ylabel('MRI Density')
37
38 if 1: # plot the EEG
39     # load the data
40
41     numSamples, numRows = 800, 4
42     eegfile = cbook.get_sample_data('eeg.dat', asfileobj=False)
43     print('loading eeg %s' % eegfile)
44     data = np.fromstring(open(eegfile, 'rb').read(), float)
45     data.shape = numSamples, numRows
46     t = np.arange(numSamples, dtype=float)/numSamples
47     ticklabel = np.arange(10)
48     ax = subplot(212)
49     xlim(0, 10)
50     xticks(np.arange(10))
51     dmin = data.min()
52     dmax = data.max()
53     dr = (dmax - dmin)*0.7 # Crowd them a bit.
54     y0 = dmin
55     y1 = (numRows - 1) * dr + dmax
56     ylim(y0, y1)
57
58     segs = []
59     for i in range(numRows):
60         segs.append((t, data[i]))
```

Jupyter Notebook is now one of many plug-ins within the JupyterLab integrated development environment

This screenshot shows the JupyterLab interface. On the left is a file browser with a tree view of files and folders. In the center is a terminal window showing Python code and its execution results. The right side shows a Python 3 notebook cell containing code and plots.

```
A simple polar plot
An example taken from the matplotlib gallery.

In [1]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

N = 20
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
ax = plt.subplot(111, projection='polar')
bars = ax.bar(theta, radii, width=width, bottom=0.0)
for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.jet(r / 10.))
    bar.set_alpha(0.5)

In [2]: %matplotlib inline
from numpy.random import beta
import matplotlib.pyplot as plt
plt.style.use('bmh')

def plot_beta_hist(a, b):
    plt.hist(beta(a, b, size=10000), histtype="stepfilled",
            bins=25, alpha=0.8, normed=True)
    return

plot_beta_hist(10, 10)
plot_beta_hist(4, 12)
plot_beta_hist(50, 12)
plot_beta_hist(6, 55)
```

The terminal output shows:

```
1  [██████████] 18.1% Tasks: 305 total, 1 running
2  [██████████] 5.0% Load average: 2.29 2.07 2.09
3  [██████████] 15.6% Uptime: 4 days, 21:59:11
4  [██████████] 5.0%
5  [██████████] Mem: 19987/81920B
6  [██████████] Swap: 2487/38720B
```

JupyterLab - an open-source, extensible IDE in a browser

PYNQ enabled boards



PYNQ-enabled boards

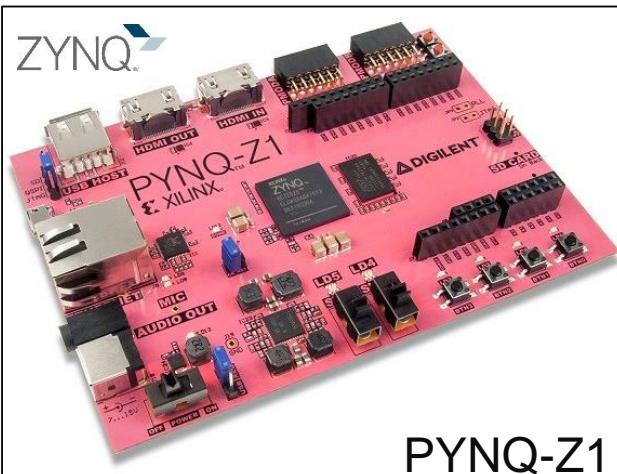


> Python productivity for Zynq

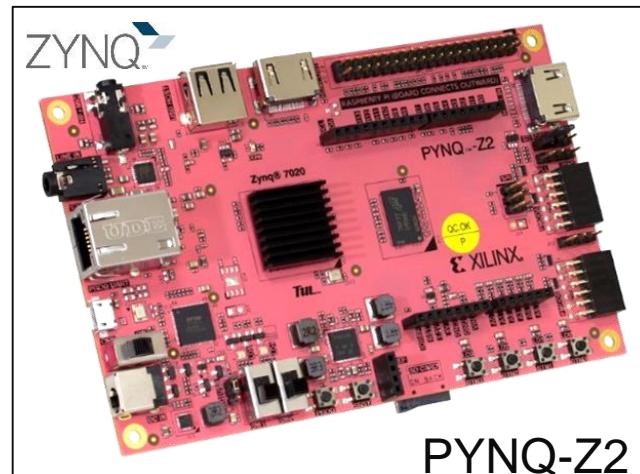
- >> Open source
- >> Build image for other Zynq boards

> Downloadable SD card image

- >> Zynq 7000
 - PYNQ-Z1 (Digilent)
 - PYNQ-Z2 (TUL)
- >> Zynq Ultrascale+
 - ZCU104 (Xilinx)



PYNQ-Z1

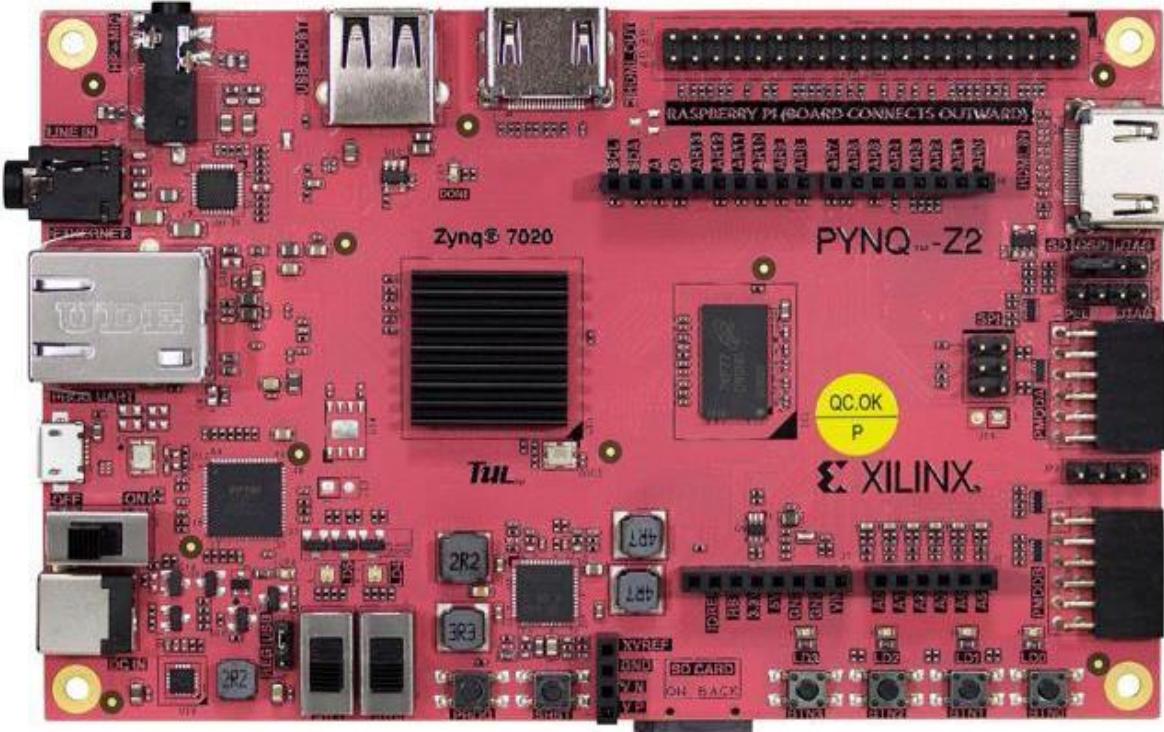


PYNQ-Z2



ZCU104

New PYNQ-Z2 Board available now



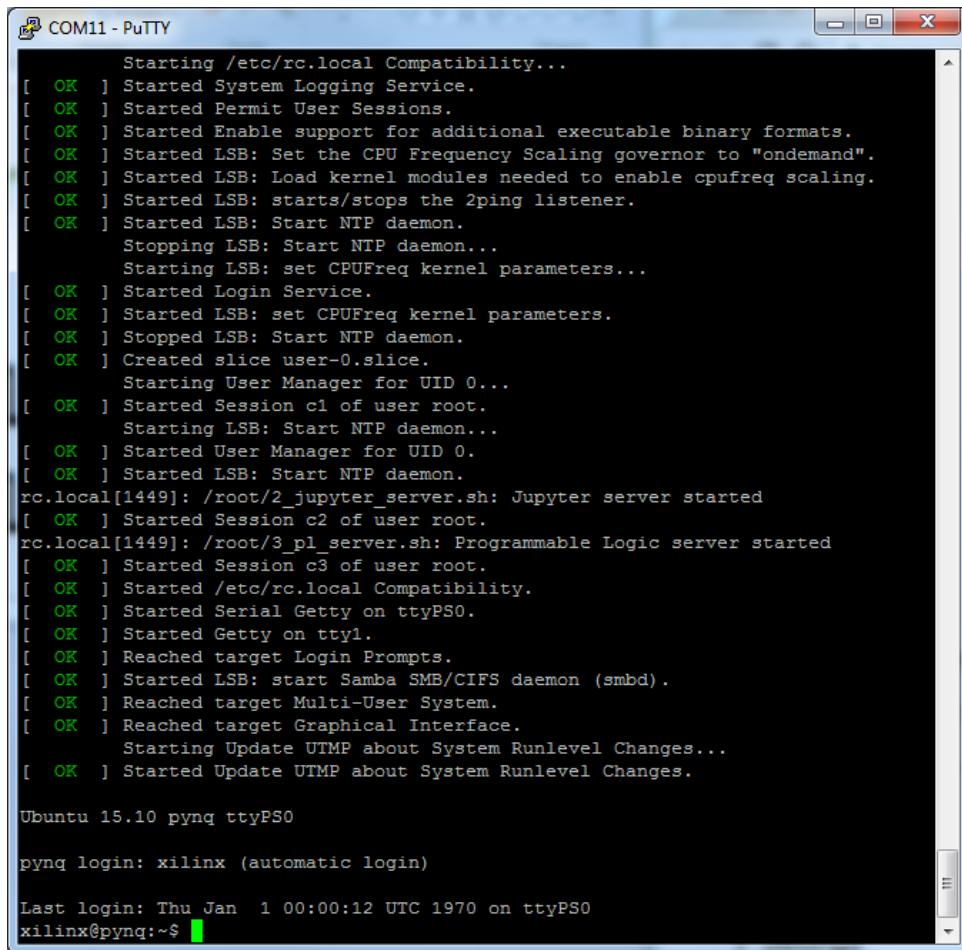
\$119/€119 or equivalent

- New PYNQ reference platform
- New stereo audio with on-board codec
- New Raspberry Pi connector
- Open source design
- Manufactured in Taiwan by TUL
- Distributed globally by Premier Farnell
- Also Newegg in US
- Academic discounts & donations available

Benefits of PYNQ



Start using PYNQ out-of-the-box



```
Starting /etc/rc.local Compatibility...
[ OK ] Started System Logging Service.
[ OK ] Started Permit User Sessions.
[ OK ] Started Enable support for additional executable binary formats.
[ OK ] Started LSB: Set the CPU Frequency Scaling governor to "ondemand".
[ OK ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
[ OK ] Started LSB: starts/stops the 2ping listener.
[ OK ] Started LSB: Start NTP daemon.
Stopping LSB: Start NTP daemon...
Starting LSB: set CPUFreq kernel parameters...
[ OK ] Started Login Service.
[ OK ] Started LSB: set CPUFreq kernel parameters.
[ OK ] Stopped LSB: Start NTP daemon.
[ OK ] Created slice user-0.slice.
Starting User Manager for UID 0...
[ OK ] Started Session c1 of user root.
Starting LSB: Start NTP daemon...
[ OK ] Started User Manager for UID 0.
[ OK ] Started LSB: Start NTP daemon.
rc.local[1449]: /root/2_jupyter_server.sh: Jupyter server started
[ OK ] Started Session c2 of user root.
rc.local[1449]: /root/3_pl_server.sh: Programmable Logic server started
[ OK ] Started Session c3 of user root.
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started Serial Getty on ttyPS0.
[ OK ] Started Getty on tty1.
[ OK ] Reached target Login Prompts.
[ OK ] Started LSB: start Samba SMB/CIFS daemon (smbd).
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Ubuntu 15.10 pynq ttyPS0

pynq login: xilinx (automatic login)

Last login: Thu Jan  1 00:00:12 UTC 1970 on ttyPS0
xilinx@pynq:~$
```

- > **PYNQ delivered as downloadable SD card image**
 - >> Linux preconfigured
- > **Additional packages and drivers pre-installed**
 - >> USB peripheral drivers: webcams, wifi modules ...
- > **PYNQ is for Zynq**
 - >> PYNQ image is portable to other Zynq boards

Start using Zynq out of the box ✓

Desktop Linux

> Network/Internet access

- >> “apt-get” to install packages from Ubuntu universe
- >> Samba(Network drive)
- >> Web services

> Git directly on board

> Compilers and other development tools

- >> Gcc., MicroBlaze, RISC-V

> Python packages

- >> “pip install”
- >> PYNQ Community examples

A selection of projects from the PYNQ community is shown below. Note that some examples are built on different versions of the PYNQ image.

Binary Neural Network
Xilinx Labs, NTNU,
University Sydney



SPynq:
NTUA Greece



Processing noisy filters
PYNQ Japan user group



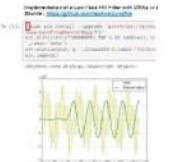
Soft GPU for ZC706
Ruhr University Bochum



Video Processing
VectorBlox



FIR filter example
CU Boulder



DMA and stream
Tutorial



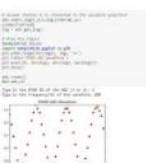
CNN Example
Imperial College London



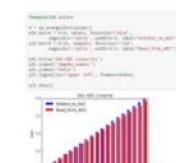
Example Notebooks

A selection of notebook examples are shown below that are included in the PYNQ image. The notebooks contain live code, and generated output from the code can be saved in the notebook. Notebooks can be viewed as webpages, or opened on a Pynq enabled board where the code cells in a notebook can be executed.

ADC waveforms



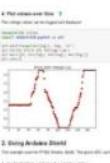
DAC ADC example



Downloading overlays



Grove ADC



Simplify downloading bitstreams to PL

- > **PYNQ 'Overlay' class**
 - >> Simplifies downloading bitstream
 - >> two lines of code
 - >> No Xilinx tools required
- > **Maintain many bitstreams on the SD card**
 - >> E.g. multiple different demos
- > **Can execute Python in browser, or from command line**

```
from pynq import Overlay  
  
ol = Overlay('gray.bit')
```

Simply and fast way to configure Programmable Logic ✓

Simplify IP debug and prototyping

> Debug of IP typically uses C/C++

> SDK tools used to:

- » Compile test application
- » Download application to board
- » Step through code

> PYNQ MMIO class allows peek/poke of IP registers from Python

- » Python executes directly on the board
- » No offline compilation, download loop
- » No SDK tools

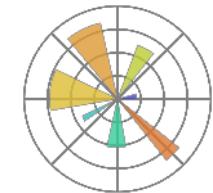
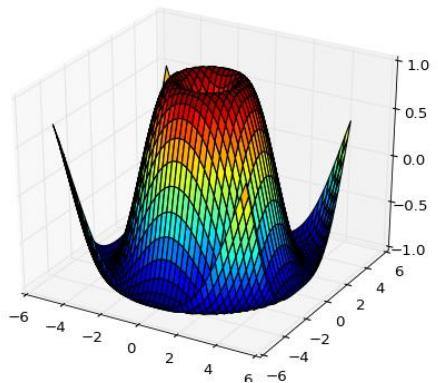
C code – compile, debug from host

```
/**************************************************************************/  
* This function does a selftest on the IIC device and XIic driver as an  
* example.  
* @param DeviceId is the XPAR_<IIC_instance>_DEVICE_ID value from  
* xparameters.h.  
*****/  
int IicSelfTestExample(u16 DeviceId)  
{  
    Status = XIic_CfgInitialize(&Iic, ConfigPtr, ConfigPtr->BaseAddress);  
    if (Status != XST_SUCCESS) {  
        return XST_FAILURE;  
    }  
  
    /* Perform a self-test to ensure that the hardware was built */  
    Status = XIic_SelfTest(&Iic);  
    if (Status != XST_SUCCESS) {  
        return XST_FAILURE;  
    }  
}
```

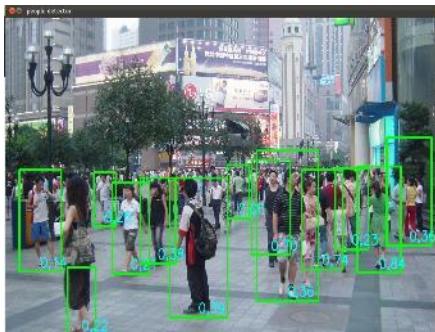
Python executes directly on target

```
from pynq import MMIO  
  
# Map registers to MMIO instance  
my_ip = MMIO(my_ip_addr, LENGTH)  
  
# Write 0x1 to start IP  
my_ip.write(CONTROL_REGISTER, 0x1)  
# Check status register  
my_ip.read(STATUS_REGISTER)
```

Python packages for data analysis and visualisation



Matplotlib



> **Take advantage of Python for data analysis and processing**

- >> NumPy
 - Scientific computing package for Python
- >> Matplotlib
 - Python 2D plotting library
- >> Pandas
 - Data analysis tools for Python
- >> OpenCV
 - Computer Vision and machine learning software

Optimized open-source software libraries ✓

Example

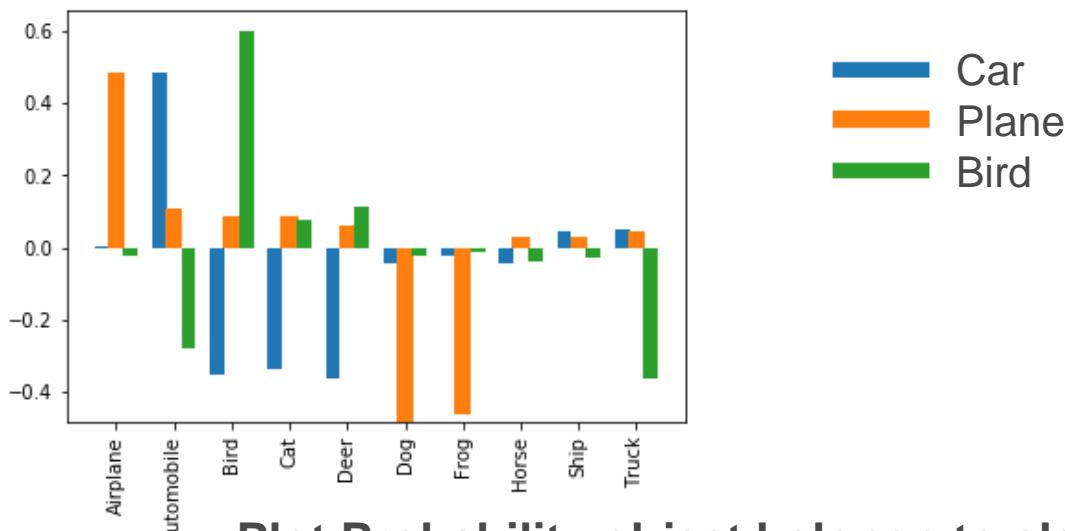
6. Detailed Classification Information

In addition to highest ranked class, it is possible to get the rank of a couple of images of a car, an airplane, and a bird and place them in a plot.

```
In [8]: from IPython.display import display
```

```
%matplotlib inline
import matplotlib.pyplot as plt

x_pos = np.arange(len(car_class))
fig, ax = plt.subplots()
ax.bar(x_pos - 0.25, (car_class/255)-1, 0.25)
ax.bar(x_pos, (air_class/255)-1, 0.3)
ax.bar(x_pos + 0.25, (bird_class/255)-1, 0.25)
ax.set_xticklabels(classifier.bnn.classes, rotation='vertical')
ax.set_xticks(x_pos)
ax.set
plt.show()
```



Probability object belongs to class

oseconds

images per second

69 163 244 249 244 267 268]

oseconds

images per second

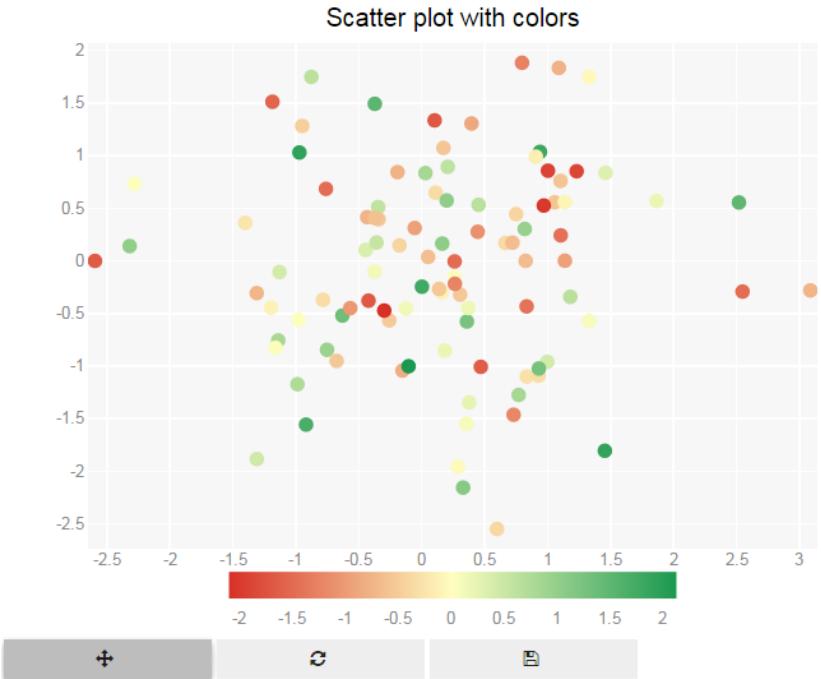
277 277 271 132 137 262 263 266]

oseconds

images per second

274 284 249 252 245 248 163]

Take advantage of interactive widgets

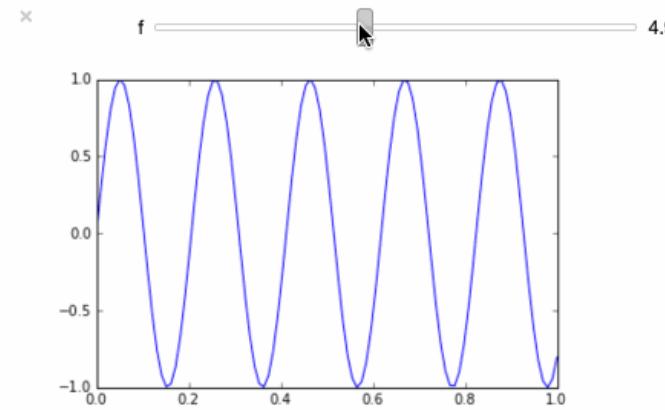


<http://jupyter.org/widgets.html>

```
In [22]: from IPython.html.widgets import *
t = arange(0.0, 1.0, 0.01)

def pltsin(f):
    plt.plot(x,sin(2*pi*t*f))
    plt.show()

interact(pltsin, f=(1,10,0.1))
```



<https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/>

```
In [7]: p2 = IntProgress(max=56)
p2.value += 10
p2.description = 'Running'
display(p2)
```

ThingA ThingA (modified)
ThingB ThingB
ThingC ThingC (modified)
ThingD ThingD

show

group_by

values

stereo fullscreen

0

Add intuitive graphical interfaces ✓

Why PYNQ is a Game-changer!

- > **PYNQ makes Zynq/ZynqU+ accessible to non-traditional customers**
- > **PYNQ delivers open source benefits**
 - >> Huge ecosystem
 - >> Extensive knowledge base
 - >> Amazing community support
- > **PYNQ enables highly-productive ...**
 - >> Prototyping
 - >> Debug
 - >> Verification
 - >> Evaluation
- > **PYNQ powers awesome demonstrators**
- > **PYNQ documentation flows are amazing**
 - >> Capture your own work
 - >> Capture work you want to re-use
- > **PYNQ designs can be ... just like software**
 - >> Packaged, published and distributed
- > **“PYNQ makes FPGAs FUN again!”, J. Gray, Xilinx Power User***

*<http://fpga.org/wp-content/uploads/2017/09/GRVI-Phalanx-on-Pynq-FCCM2017-Pynq-Workshop.pdf>

Community

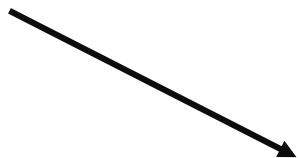




Home Get Started PYNQ-Z1 Bo

Community Projects

Selection of projects
and notebooks



A selection of projects from the PYNQ community is shown below. Note that some examples are built on different versions of the PYNQ image.

Binary Neural Network

Xilinx Labs, NTNU,
University Sydney

[BINonPynq](#)
The project shows how to build a binary neural network using the Xilinx Zynq SoC. It includes a Jupyter notebook and a hardware design.



SPyng:

NTUA Greece

[SPyng initiation](#)
In this project we use Python to initialize the Zynq SoC after booting up. It also shows how to read and write to memory.



Processing noisy filters

PYNQ Japan user group

[ではじめる機械学習](#)
Gardine Maruya
This notebook shows how to implement a simple linear regression model using Python and the PYNQ Zynq SoC.

Soft GPU for ZC706

Ruhr University Bochum

[Soft GPU for ZC706](#)
This notebook shows how to implement a soft GPU for the ZC706 SoC.

Video Processing

Vectorblox

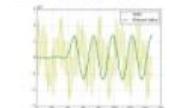
[VectorBlox Video Processing](#)
In this module, several filters can be applied to YUV input images. These filters include motion blur, motion blur with threshold value, motion blur with threshold value and motion blur with threshold value.



FIR filter example

CU Boulder

[Implementation of a Linear Phase FIR Filter with DMA on Zynq](#)
Detailed description of the FIR filter implementation with DMA on Zynq.



DMA and stream

Tutorial

[DMA to streamified interface example](#)
Detailed description of the DMA and streamified interface example.



CNN Example

Imperial College London

[FPGA Deep learning CNN example](#)
Detailed description of the CNN example.

Example Notebooks

A selection of notebook examples are shown below that are included in the PYNQ image. The notebooks contain live code, and generated output from the code can be saved in the notebook. Notebooks can be viewed as webpages, or opened on a Pynq enabled board where the code cells in a notebook can be executed.

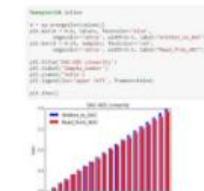
ADC waveforms

[ADC waveforms](#)
Detailed description of the ADC waveforms example.



DAC ADC example

[DAC ADC example](#)
Detailed description of the DAC ADC example.



Downloading overlays

[Downloading Overlays](#)
Detailed description of how to download an FPGA overlay and update it.

1. Installing an overlay
To update an overlay, you must first present the file to the Zynq SoC. This involves file download via a serial port connection. The example file contains the details of the overlay.

2. Applying the overlay
After the file has been presented to the Zynq SoC, the overlay needs to be applied. This is done via a command to the Zynq SoC.

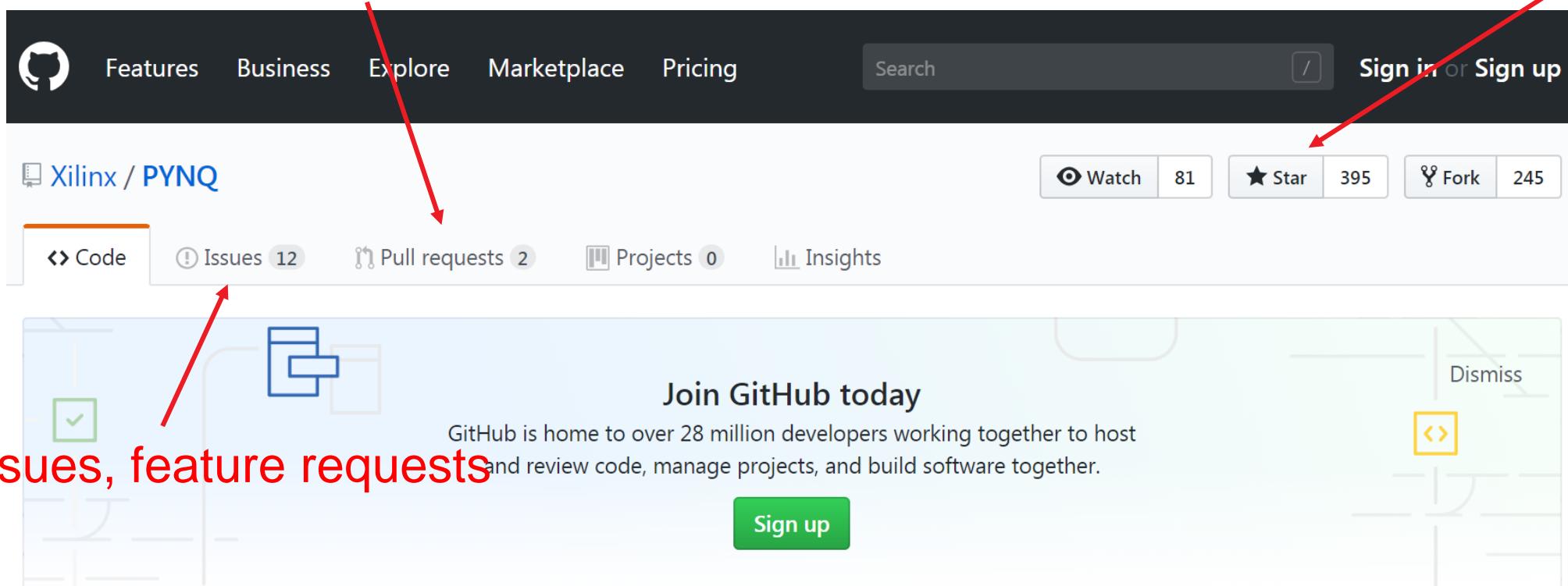
Grove ADC

[Grove ADC](#)
Detailed description of the Grove ADC example.



All Feedback helps

Contribute



Python Productivity for ZYNQ <http://www.pynq.io/>

pynq

Summary



- > PYNQ is Python productivity for Zynq
- > Everything runs on Zynq, access via a browser
- > Support for Zynq Ultrascale+
- > Overlays are hardware libraries and enable software developers to use Zynq
- > Provides a rapid prototyping framework for hardware developers



pynq.io

pynq.readthedocs.org

github.com/Xilinx/PYNQ

tul.com.tw/ProductsPYNQ-Z2.html

pynq.io/support

Adaptable. Intelligent.

