



# Python Productivity for Zynq



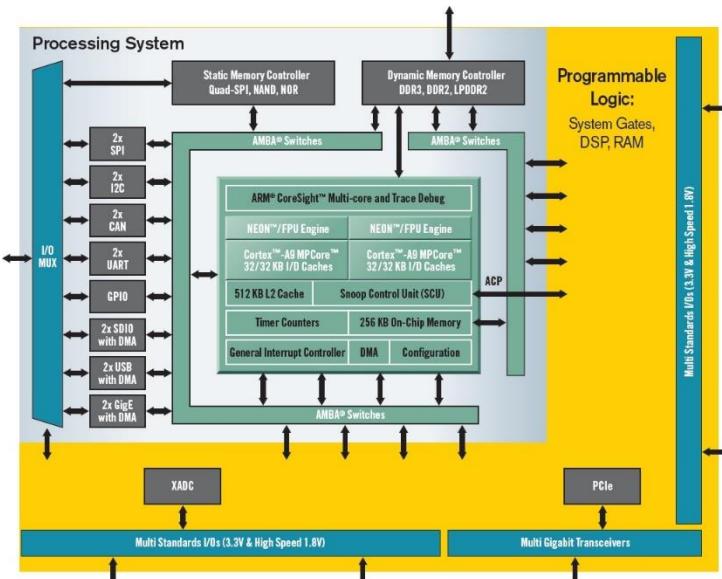
- > Zynq & Zynq Ultrascale+
- > PYNQ Framework
- > Technologies
- > Community



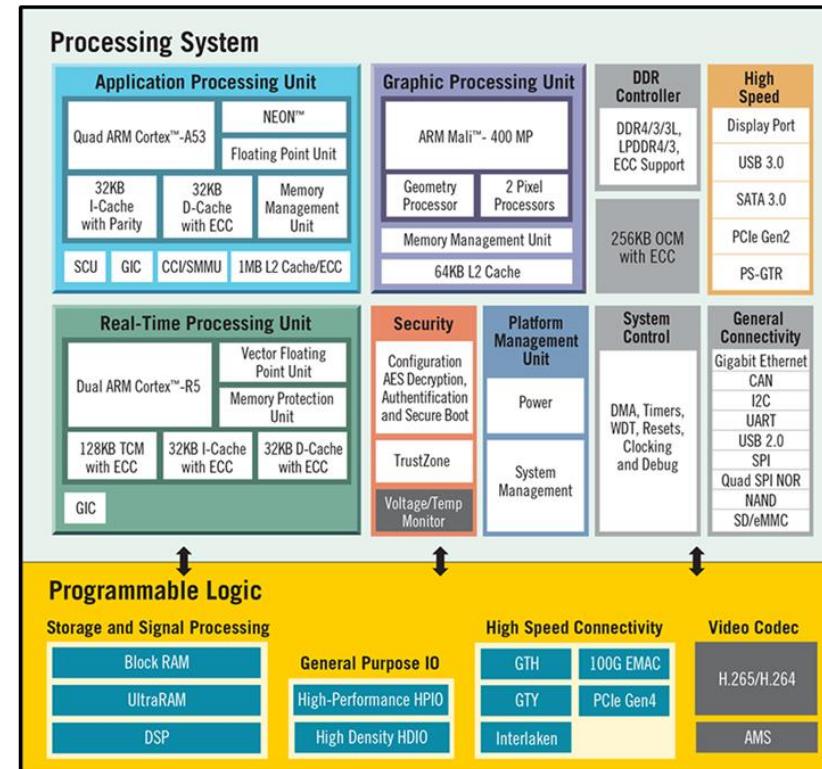
# ZYNQ and ZYNQ UltraSCALE+

Best-in-class, All Programmable SoCs

ZYNQ 7000



ZYNQ UltraSCALE+



FPGAs and tightly-integrated CPUs enable entirely new opportunities

# Zynq applications

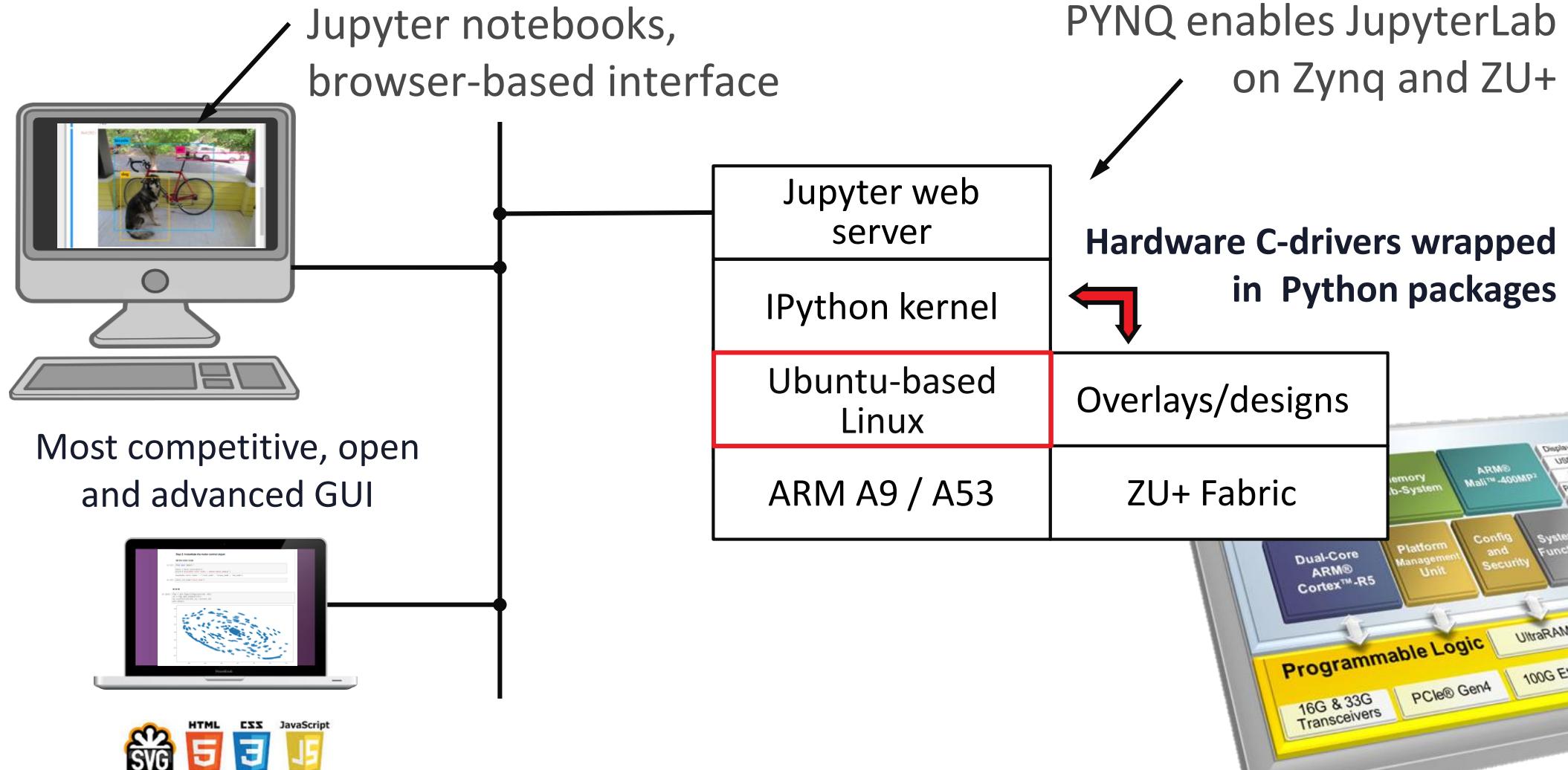


- > Make Zynq so easy-to-use that programmers can access the benefits of FPGAs without learning advanced digital design skills



## 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



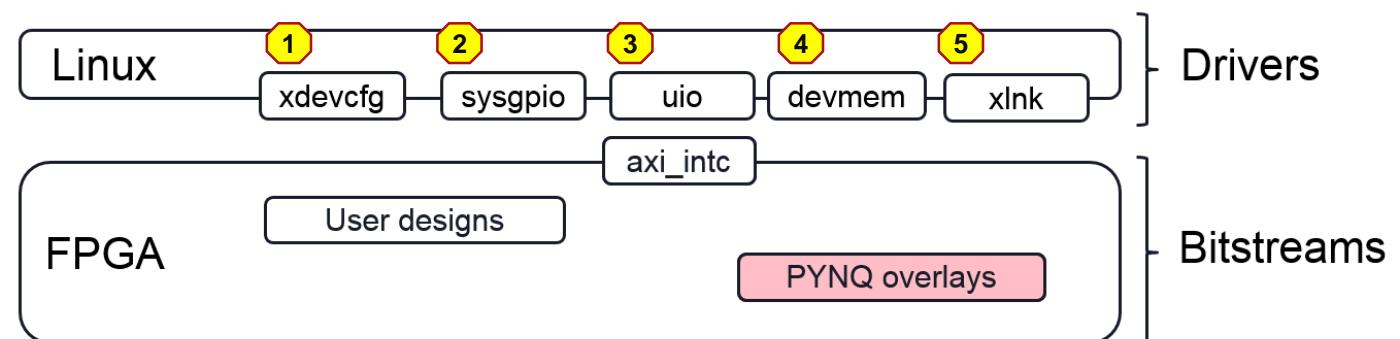
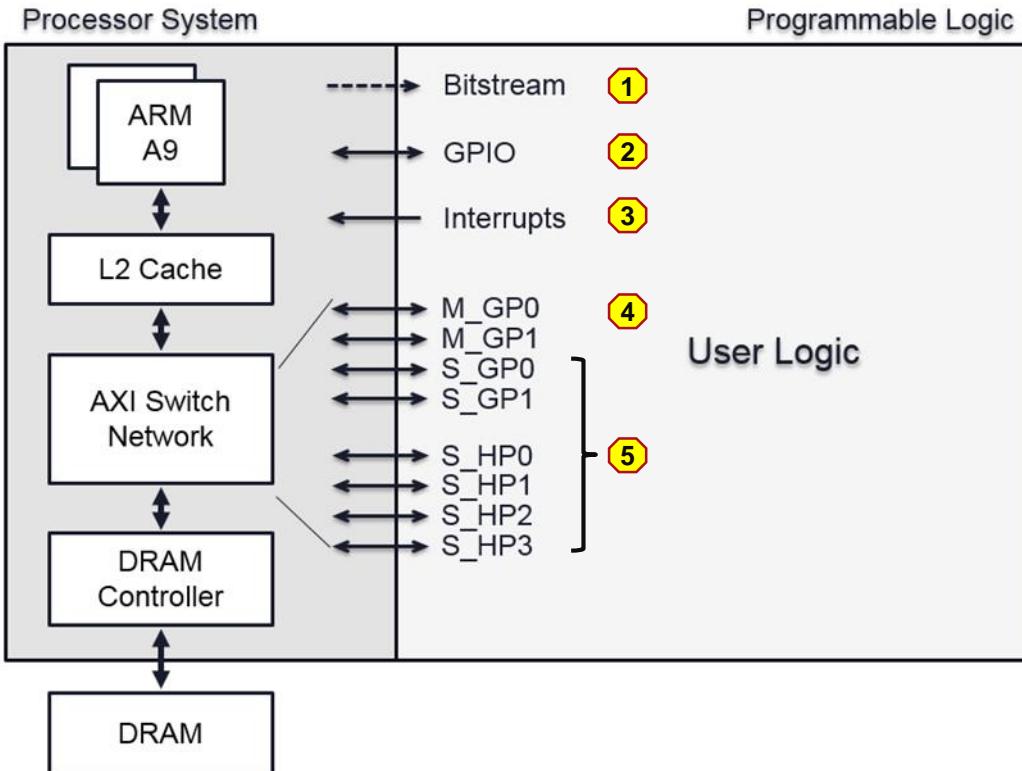
PYNQ uses the PetaLinux build flow and board support package:

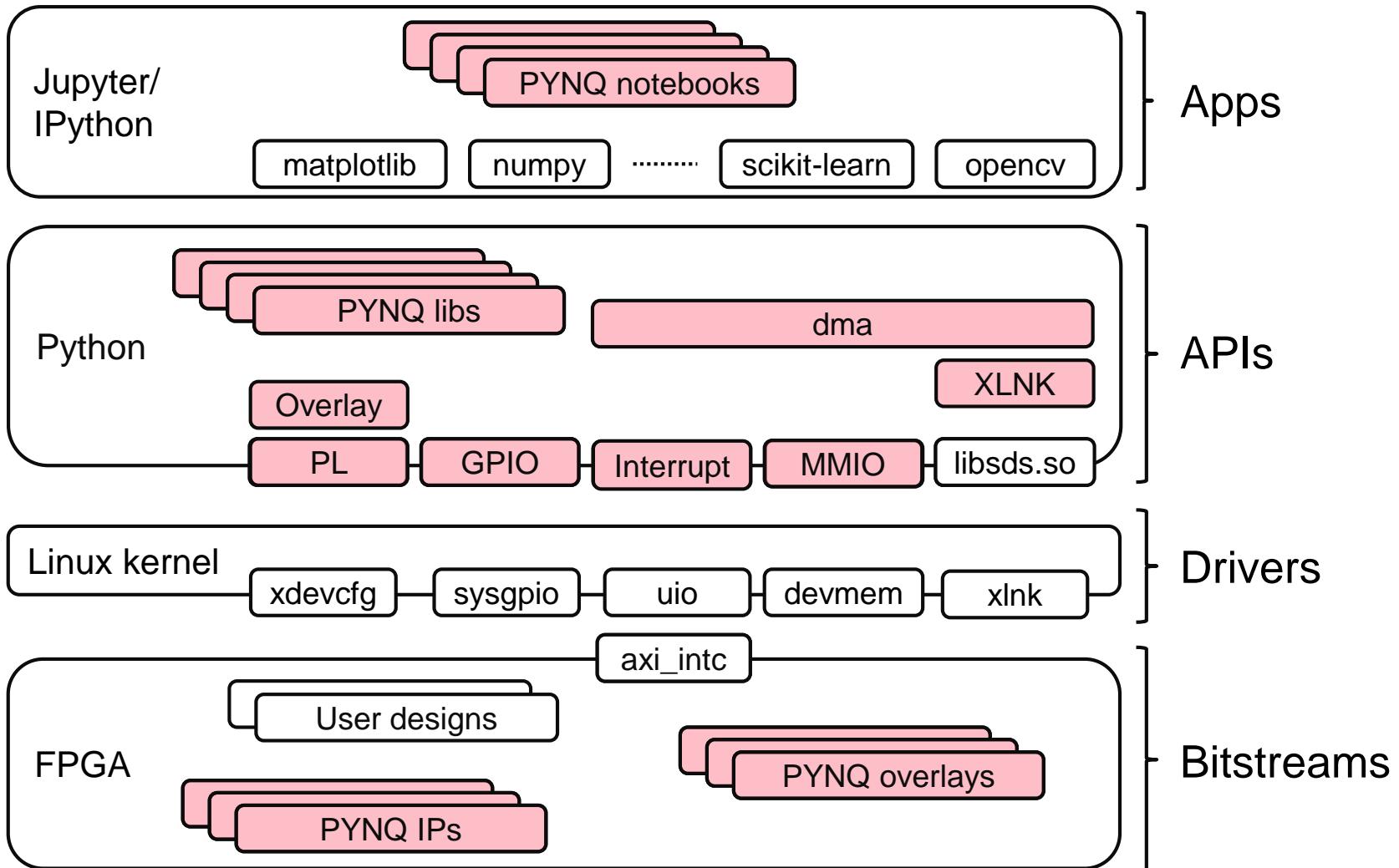
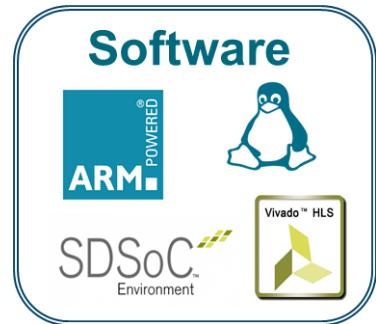
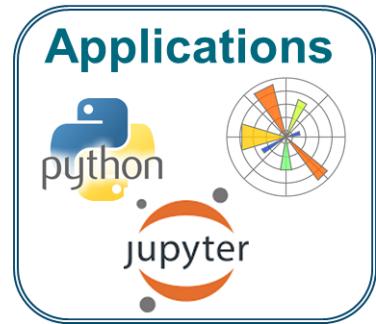
- Access to all Xilinx kernel patches
- Works with any Xilinx supported board
- Configured with additional drivers for PS-PL interfaces

# 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:

- Xilinx / QNN-MO-PYNQ**: A notebook titled "dorefanet-imagenet-samples.ipynb" showing code for image classification and a Beagleboard photo.
- Xilinx / IoT-SPYN**: A notebook titled "spyn.ipynb" demonstrating AC motor control.
- Xilinx / PYNQ-DL**: A notebook titled "resize.ipynb" illustrating image resizing.
- Xilinx / PYNQ-ComputerVision**: A notebook titled "filter2d\_and\_dilate.ipynb" showing OpenCV overlay functionality.

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

Download a design from GitHub with a single Python command:

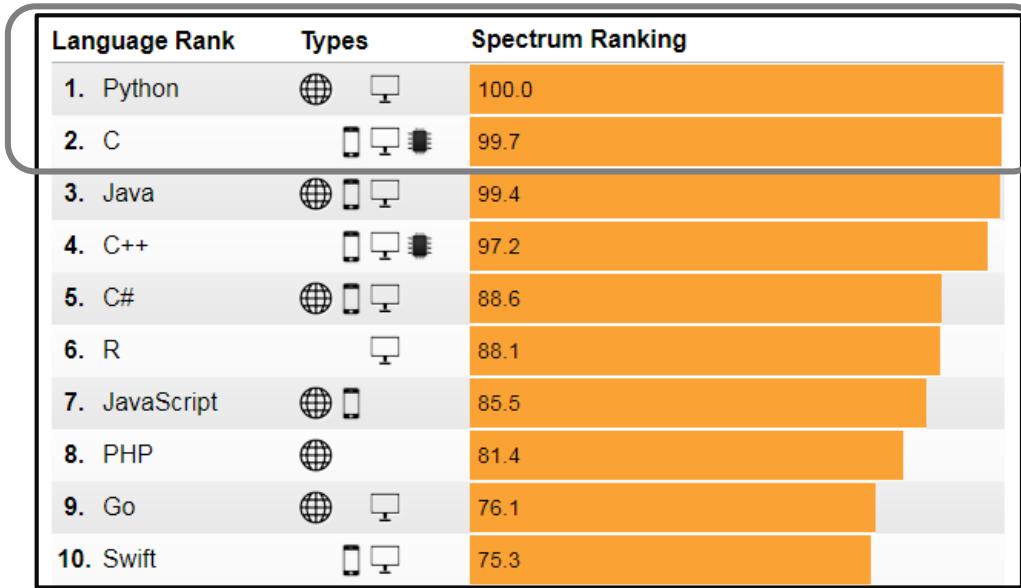
```
pip3.6 install git+https://github.com/Xilinx/pynqDL.git
```

# PYNQ enabling technologies

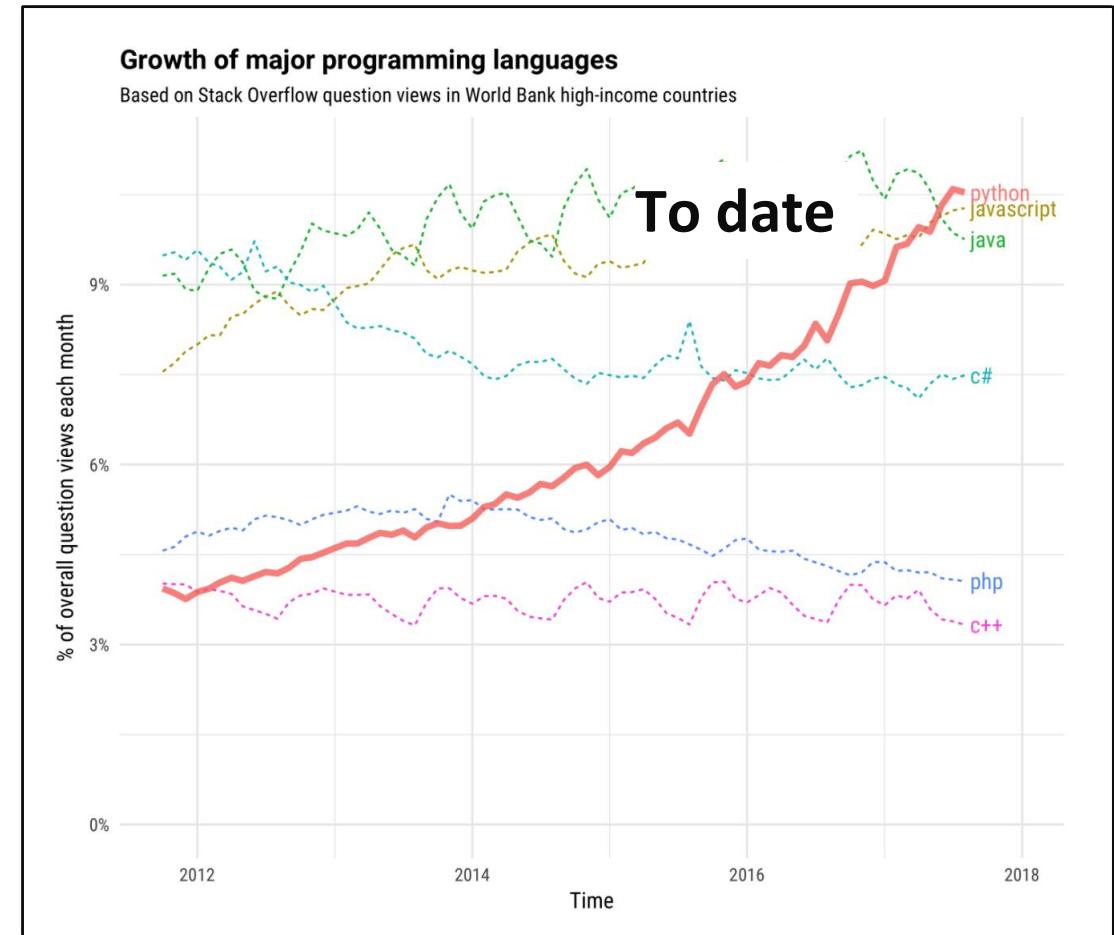


# Python is increasingly the language of choice

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...

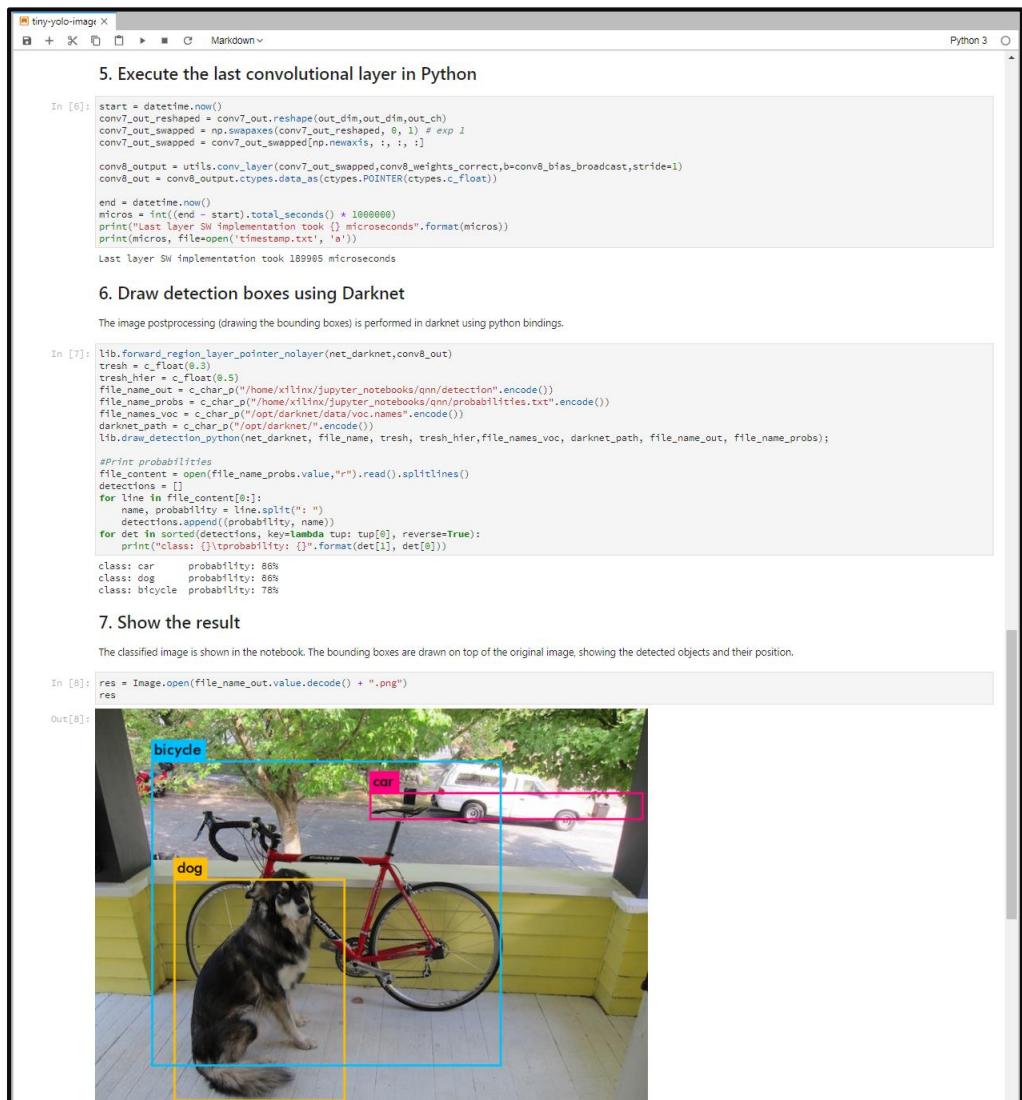
135,734 projects    944,172 releases    1,257,368 files    273,938 users



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.

C<sub>0</sub>Python 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.

**In [6]:**

```
5. Execute the last convolutional layer in Python
In [6]: start = datetime.now()
conv7_out_reshaped = conv7_out.reshape(out_dim,out_dim,out_ch)
conv7_out_swapped = np.swapaxes(conv7_out_reshaped, 0, 1) # exp 1
conv7_out_swapped = conv7_out_swapped[np.newaxis, :, :, :]
conv8_output = utils.conv_layer(conv7_out_swapped,conv8_weights_correct,b=conv8_bias_broadcast,stride=1)
conv8_out = conv8_output.ctypes.data_as(ctypes.POINTER(ctypes.c_float))

end = datetime.now()
micros = int((end - start).total_seconds() * 1000000)
print("Last layer SW implementation took {} microseconds".format(micros))
print(micros, file=open('timestamp.txt', 'a'))

Last layer SW implementation took 189965 microseconds
```

**In [7]:**

```
6. Draw detection boxes using Darknet
The image postprocessing (drawing the bounding boxes) is performed in darknet using python bindings.

In [7]: lib.forward_region_layer_pointer_nolayer(net_darknet,conv8_out)
tresh = c_float(0.3)
tresh_hier = c_float(0.8)
file_name_out = c_char_p("/home/xilinx/jupyter_notebooks/qnn/detection".encode())
file_name_probs = c_char_p("/home/xilinx/jupyter_notebooks/qnn/probabilities.txt".encode())
file_name_voc = c_char_p("/opt/darknet/data/voc.names".encode())
darknet_path = c_char_p("/opt/darknet".encode())
lib.draw_detection_python(net_darknet, file_name, tresh, tresh_hier,file_names_voc, darknet_path, file_name_out, file_name_probs);

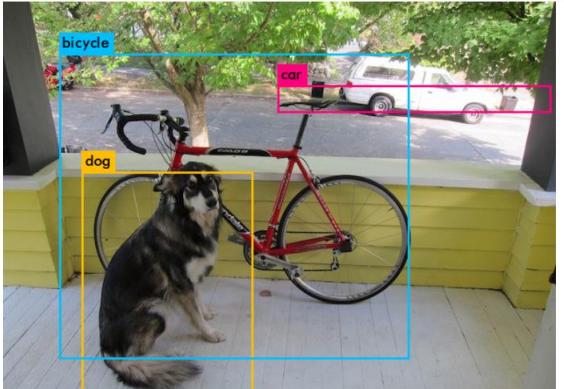
#print(probabilities
file_content = open(file_name_probs.value.decode(),"r").readlines()
detections = []
for line in file_content[0:]:
    name, probability = line.split(": ")
    detections.append((probability, name))
for det in sorted(detections, key=lambda tup: tup[0], reverse=True):
    print("class: {} \tprobability: {}".format(det[1], det[0]))
```

**In [8]:**

```
7. Show the result
The classified image is shown in the notebook. The bounding boxes are drawn on top of the original image, showing the detected objects and their position.

In [8]: res = Image.open(file_name_out.value.decode()) + ".png"
res
```

**Out[8]:**



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 a search bar. In the center is a terminal window showing Python code and its execution results, including a histogram plot. To the right is a Python 3 notebook cell containing code for plotting MRI data. Below the terminal and notebook are two small image 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

```

```
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
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=cmap.magma_r)
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 = im[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 a search bar. In the center is a Python 3 notebook cell containing code for plotting polar data, resulting in a circular plot. To the right is a terminal window showing system statistics and a process list.

```
Python 3.5.2 (Continuum Analytics, Inc.) (default, Jul 2 2016, 17:52:12)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0.dev -- An enhanced Interactive Python.
? --> Quick reference.
help --> Python's own help system.
object? --> Details about 'object', use 'object??' for extra details.

In [1]: %matplotlib inline
import numpy as np
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)

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)
```

```
1 [1] Tasks: 305 total, 1 running
5.00% Load average: 2.29 2.07 2.09
15.6% Uptime: 4 days, 21:59:11
5.00%
Mem: 19987/319296
Swap: 2487/387296
```

| PID   | USER   | PRI | NI | VIRT  | RES  | SHR | CPU | MEM% | TIME% | Command          |
|-------|--------|-----|----|-------|------|-----|-----|------|-------|------------------|
| 82374 | fperez | 31  | 0  | 2389M | 2048 | 0   | 0.0 | 0.00 | 0.00  | httpd            |
| 46    | root   | 0   | 0  | 0     | 0    | 0   | 0.0 | 0.00 | 0.00  | (launchd)        |
| 47    | root   | 0   | 0  | 0     | 0    | 0   | 0.0 | 0.00 | 0.00  | (syslogd)        |
| 48    | root   | 0   | 0  | 0     | 0    | 0   | 0.0 | 0.00 | 0.00  | (UserEventAgent) |

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

# PYNQ enabled boards



# PYNQ-enabled boards

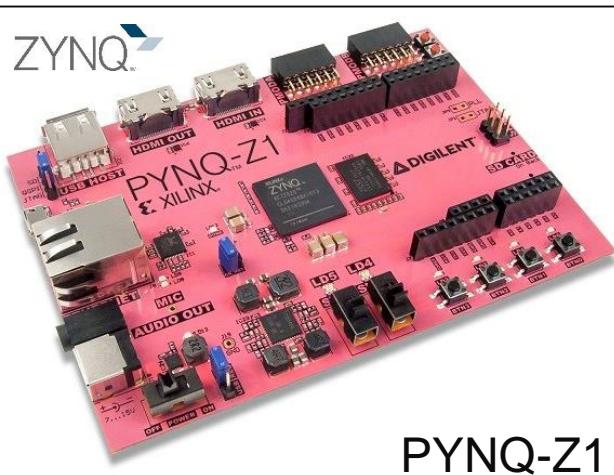


## > Python productivity for Zynq

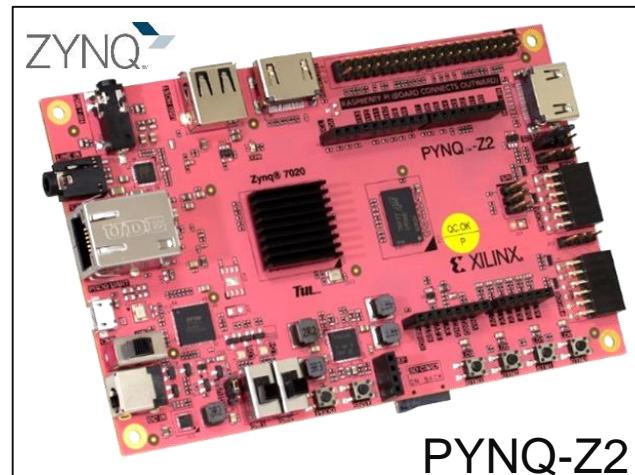
- >> Open source
- >> Build image for other Zynq boards

## > Downloadable SD card image

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



PYNQ-Z1

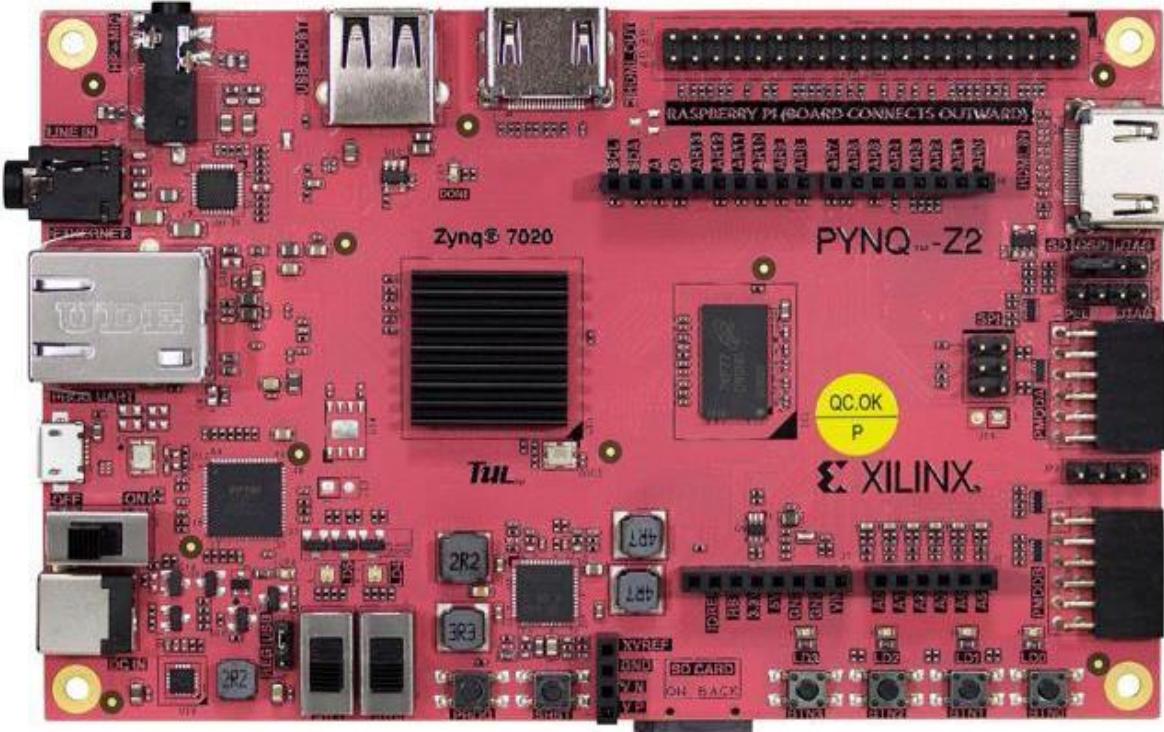


PYNQ-Z2



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
- Z2 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



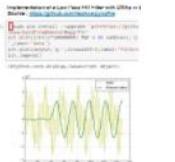
**Soft GPU for ZC706**  
Ruhr University Bochum



**Video Processing**  
VectorBlox



**FIR filter example**  
CU Boulder



**DMA and stream**  
Tutorial



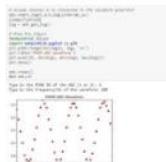
**CNN Example**  
Imperial College London



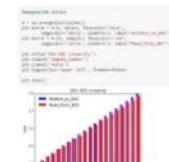
## Example Notebooks

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

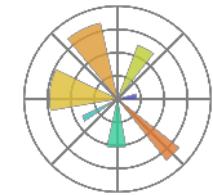
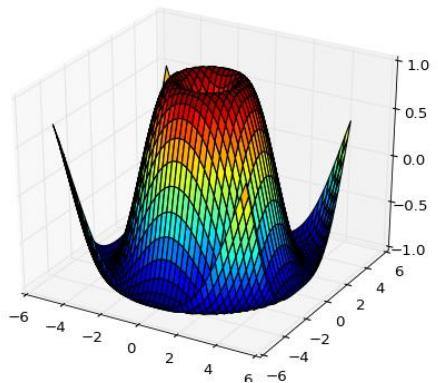
```
from pynq import MMIO

# Map registers to MMIO instance
my_ip = MMIO(my_ip_addr, LENGTH)

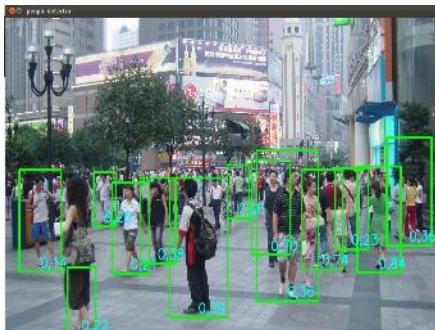
# Write 0x1 to start IP
my_ip.write(CONTROL_REGISTER, 0x1)
# Check status register
my_ip.read STATUS_REGISTER)
```

Rapid testing and prototyping ✓

# 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.

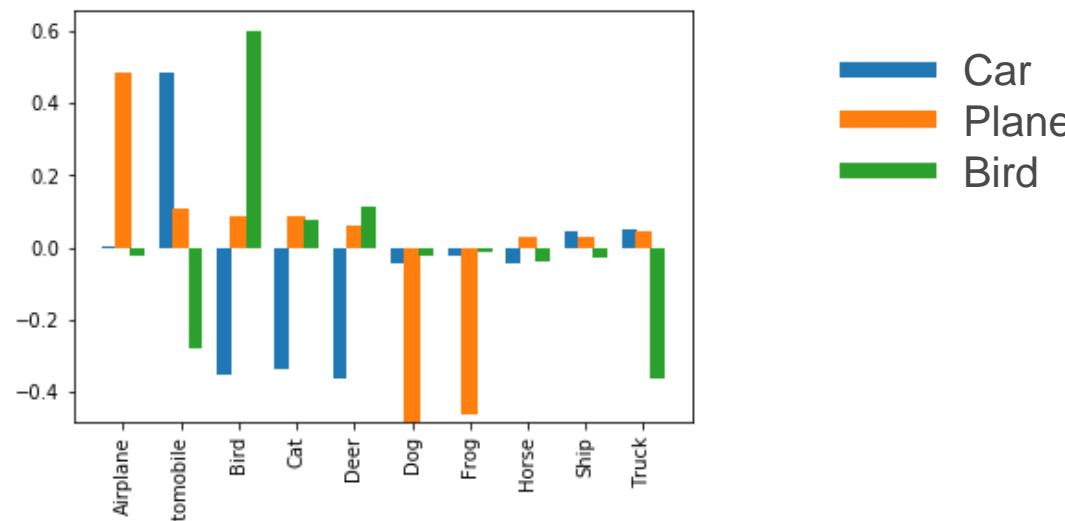


### Probability object belongs to class

```
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()
```



oseconds

images per second

69 163 244 249 244 267 268]

oseconds

images per second

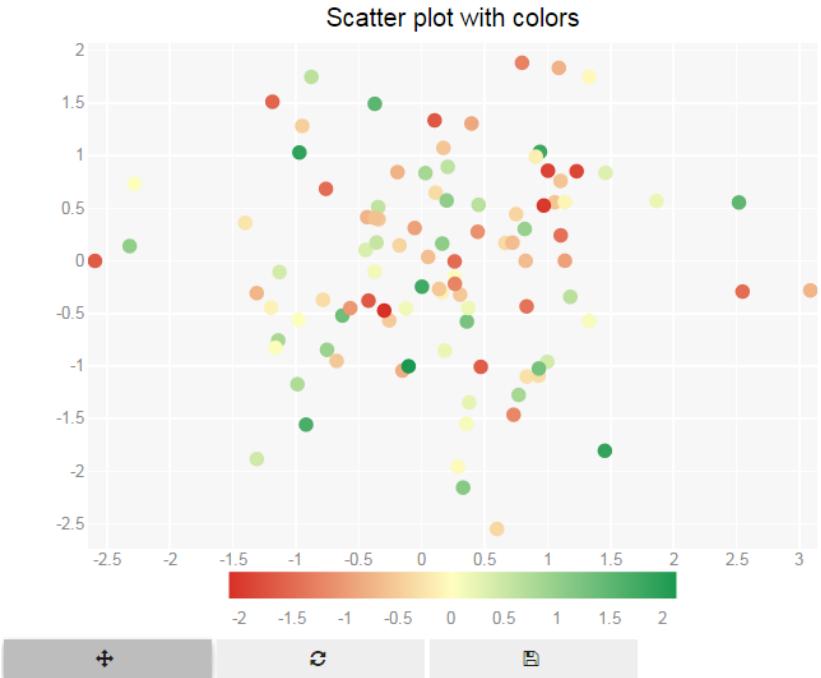
277 277 271 132 137 262 263 266]

oseconds

images per second

274 284 249 252 245 248 163]

# Take advantage of interactive widgets

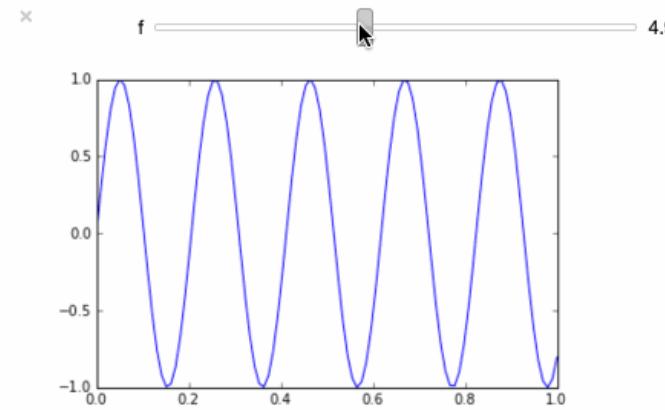


<http://jupyter.org/widgets.html>

```
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)
```

Running

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

show Annotator themes ▾  
group\_by How often do you use Jupyter Notebook? ▾  
values Percentages ▾

stereo   
fullscreen

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**

# 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 with a Xilinx Zynq SoC. It uses the Zynq's soft floating-point library to implement the binary neural network. The code is written in Python and runs on the Zynq's soft processor.



### SPyng:

NTUA Greece

[SPyng](#)  
In this project we propose Pyng to implement the BrainCo software application on the Zynq SoC. The application is a brain-computer interface (BCI) that allows users to control a cursor using their thoughts. The code is written in Python and runs on the Zynq's soft processor.



### Processing noisy filters

PYNQ Japan user group

[processing\\_noisy\\_filters](#)  
This project shows how to process noisy filters using PYNQ. It includes a Python notebook that demonstrates how to filter a noisy signal using a Butterworth filter. The code is written in Python and runs on the Zynq's soft processor.



### Soft GPU for ZC706

Ruhr University Bochum

[soft\\_gpu\\_zc706](#)  
This project shows how to implement a soft GPU on a ZC706 board. It includes a Python notebook that demonstrates how to run a simple CUDA kernel on the soft GPU. The code is written in Python and runs on the Zynq's soft processor.



### Video Processing

Vectorblobx

[VideoProcessing](#)  
This project shows how to implement video processing on a Zynq SoC using VectorBlox. It includes a Python notebook that demonstrates how to process a video frame using a VectorBlox block. The code is written in Python and runs on the Zynq's soft processor.



### FIR filter example

CU Boulder

[FIRfilterexample](#)  
This project shows how to implement a Linear Phase FIR filter with DMA on a Zynq SoC. It includes a Python notebook that demonstrates how to filter a signal using a FIR filter. The code is written in Python and runs on the Zynq's soft processor.



### DMA and stream

Tutorial

[DMAandstream](#)  
This tutorial shows how to use DMA and streams to transfer data between memory and the Zynq SoC. It includes a Python notebook that demonstrates how to use DMA and streams to transfer data. The code is written in Python and runs on the Zynq's soft processor.



### CNN Example

Imperial College London

[CNNExample](#)  
This project shows how to implement a convolutional neural network (CNN) on a Zynq SoC. It includes a Python notebook that demonstrates how to train a CNN on a dataset. The code is written in Python and runs on the Zynq's soft processor.



## 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.

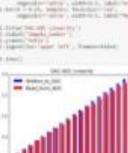
### ADC waveforms

[ADCwaveforms](#)  
This notebook shows how to generate ADC waveforms on a Zynq SoC. It includes a Python notebook that demonstrates how to generate ADC waveforms. The code is written in Python and runs on the Zynq's soft processor.



### DAC ADC example

[DACADCexample](#)  
This notebook shows how to generate DAC and ADC signals on a Zynq SoC. It includes a Python notebook that demonstrates how to generate DAC and ADC signals. The code is written in Python and runs on the Zynq's soft processor.



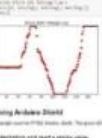
### Downloading overlays

[DownloadOverlays](#)  
This notebook shows how to download an FPGA overlay and update an overlay. It includes a Python notebook that demonstrates how to download an overlay and update it. The code is written in Python and runs on the Zynq's soft processor.



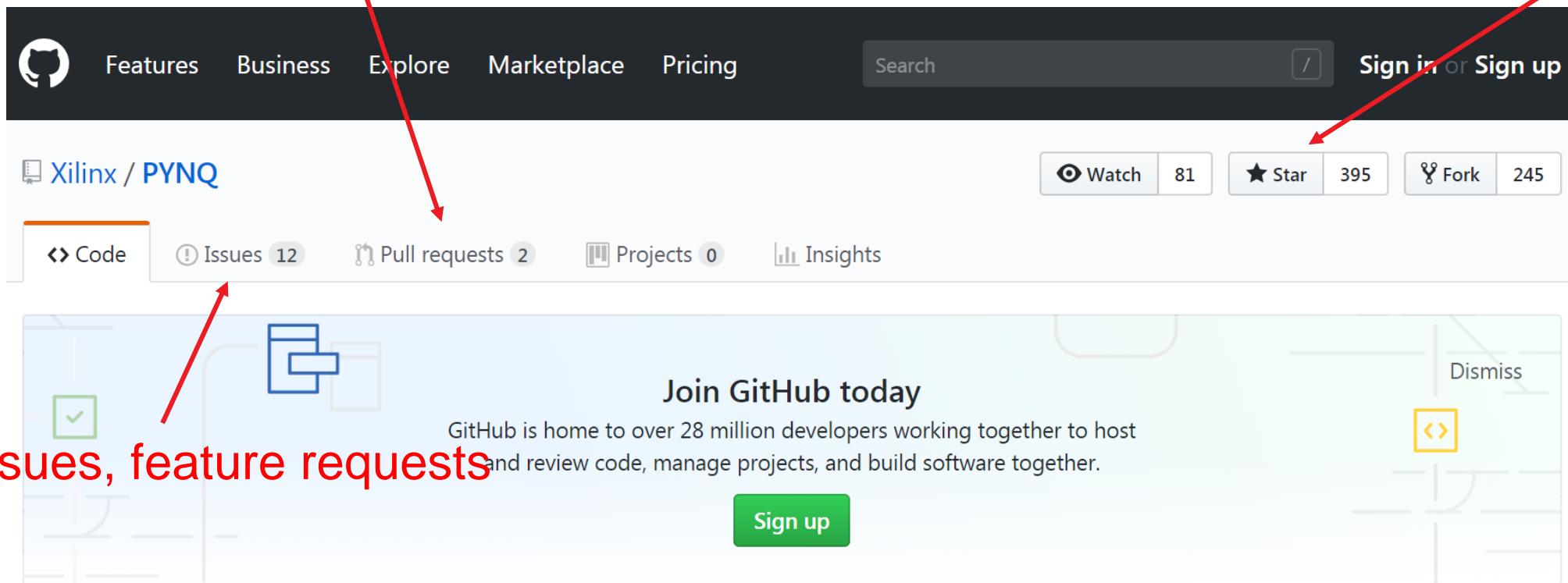
### Grove ADC

[GroveADC](#)  
This notebook shows how to use a Grove ADC module with a Zynq SoC. It includes a Python notebook that demonstrates how to use a Grove ADC module. The code is written in Python and runs on the Zynq's soft processor.



# All Feedback helps

Contribute



Issues, feature requests

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](http://pynq.io)

[pynq.readthedocs.org](http://pynq.readthedocs.org)

[github.com/Xilinx/PYNQ](http://github.com/Xilinx/PYNQ)

[tul.com.tw/ProductsPYNQ-Z2.html](http://tul.com.tw/ProductsPYNQ-Z2.html)

[pynq.io/support](http://pynq.io/support)

# Adaptable. Intelligent.

