

GO PYNQ!

Get introduced to Xilinx FPGAs and Fun



Why PYNQ?

Ease of use of Python with Programmable technologies



Introducing to:

- > PYNQ and ZYNQ
- > PYNQ Framework
- > PYNQ-Z2 Board
- > Jupyter Notebook Interface
- > PYNQ Overlay
- > PYNQ Packages
- > Benefits of PYNQ
- > Join the Community!

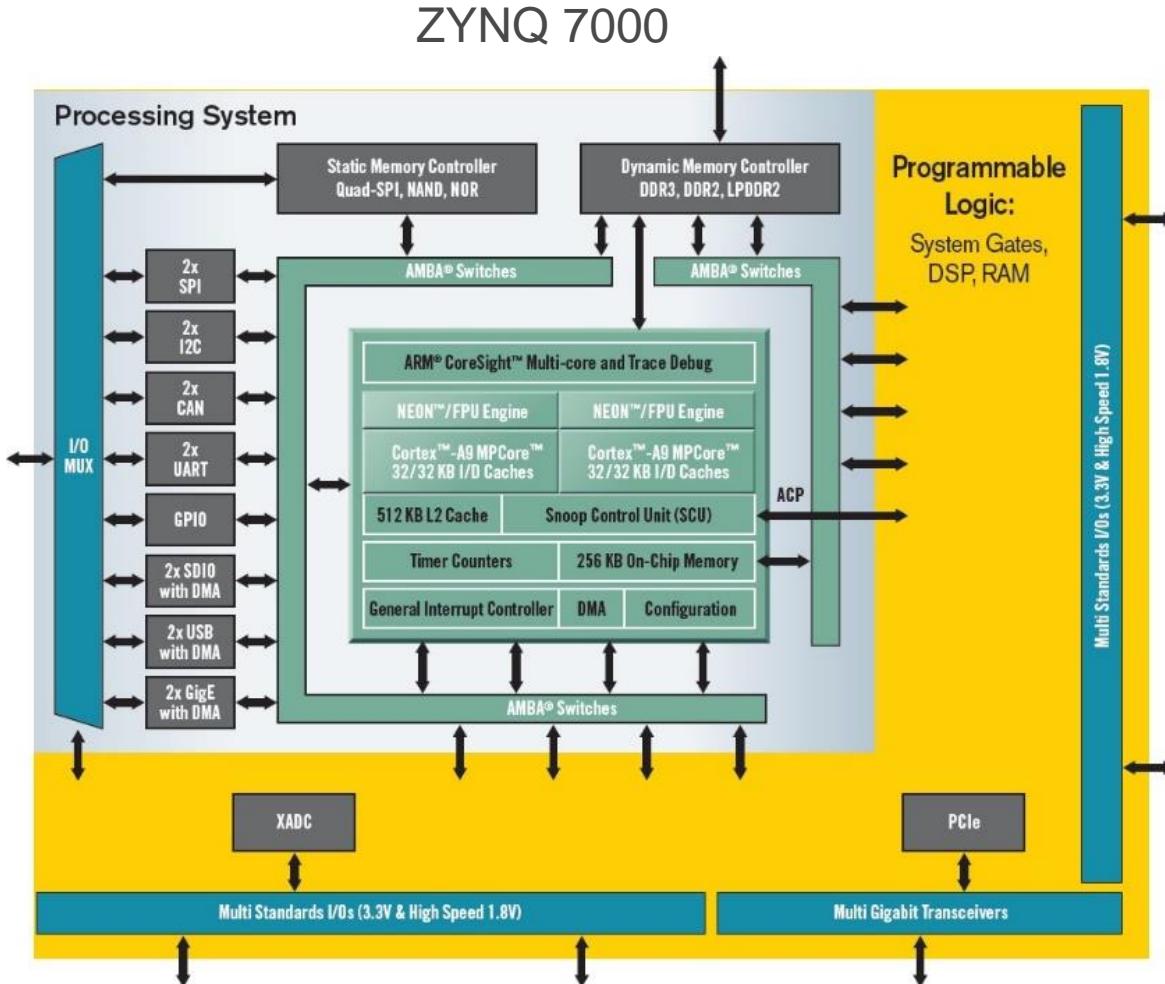




Python Productivity for Zynq



Best-in-class, All Programmable SoCs



- The device is based on a dual-core ARM[®] Cortex[®]-A9 processor (referred to as the *Processing System* or **PS**), integrated with FPGA fabric (referred to as *Programmable Logic* or **PL**).
- The *PS* subsystem includes a number of dedicated peripherals (memory controllers, USB, Uart, IIC, SPI etc) and can be extended with additional hardware IP in a *PL Overlay*.

Zynq applications

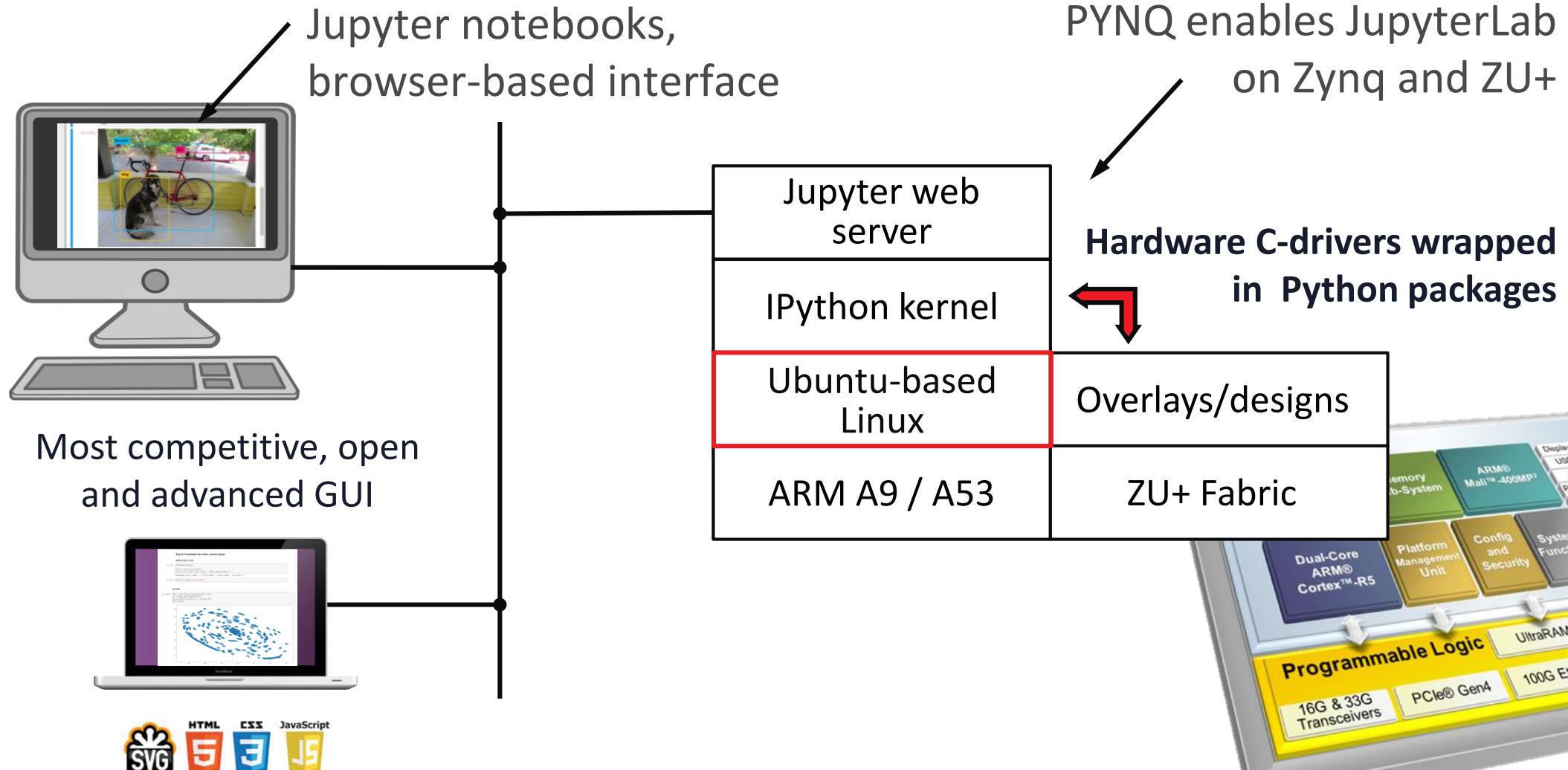


FPGAs and tightly-integrated CPUs enable entirely new opportunities



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



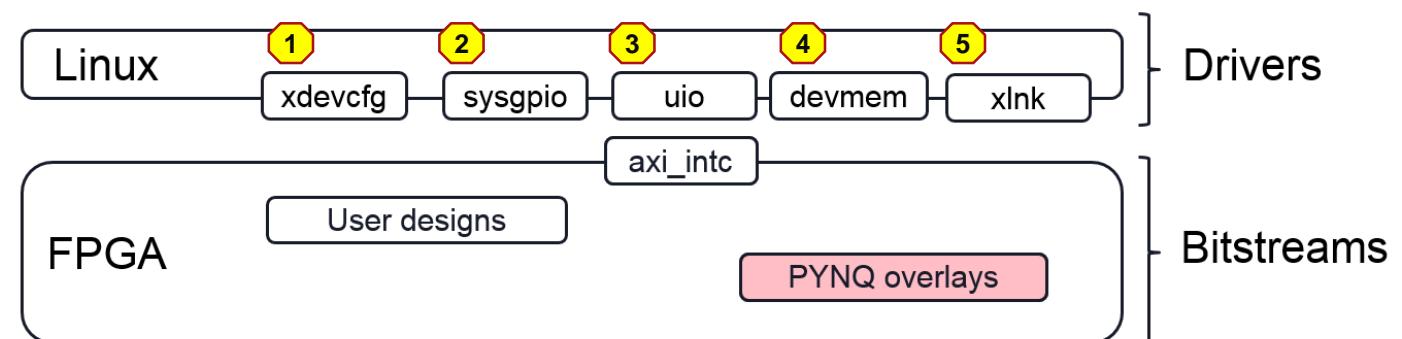
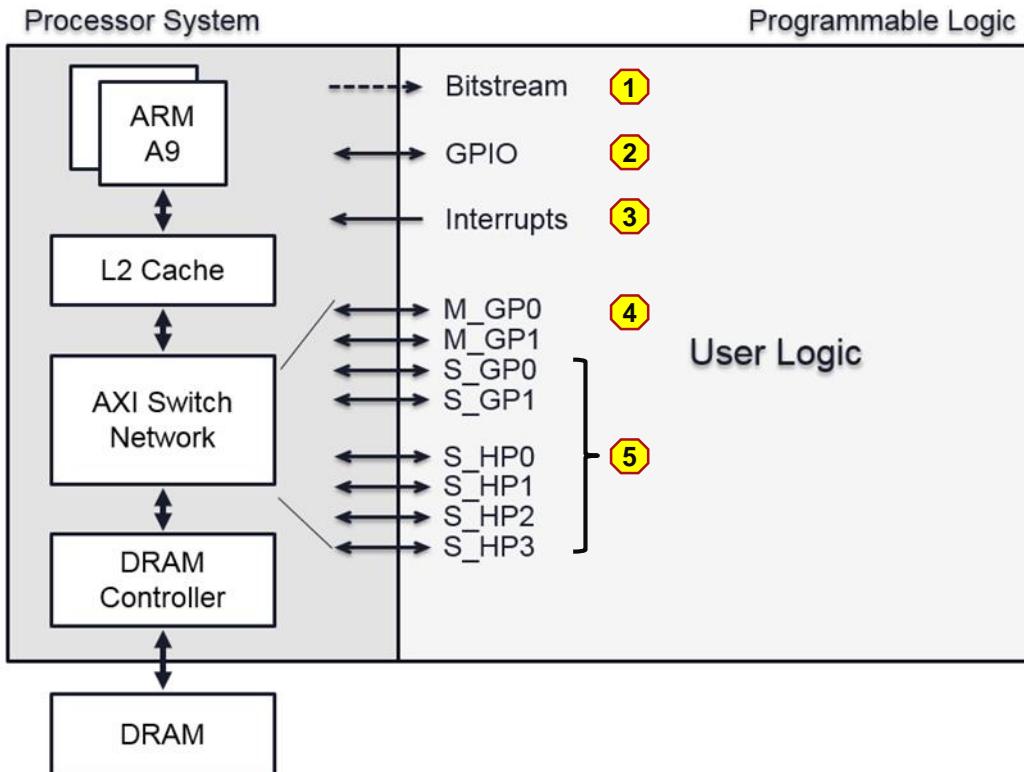
PYNQ uses the PetaLinux build flow and board support package:

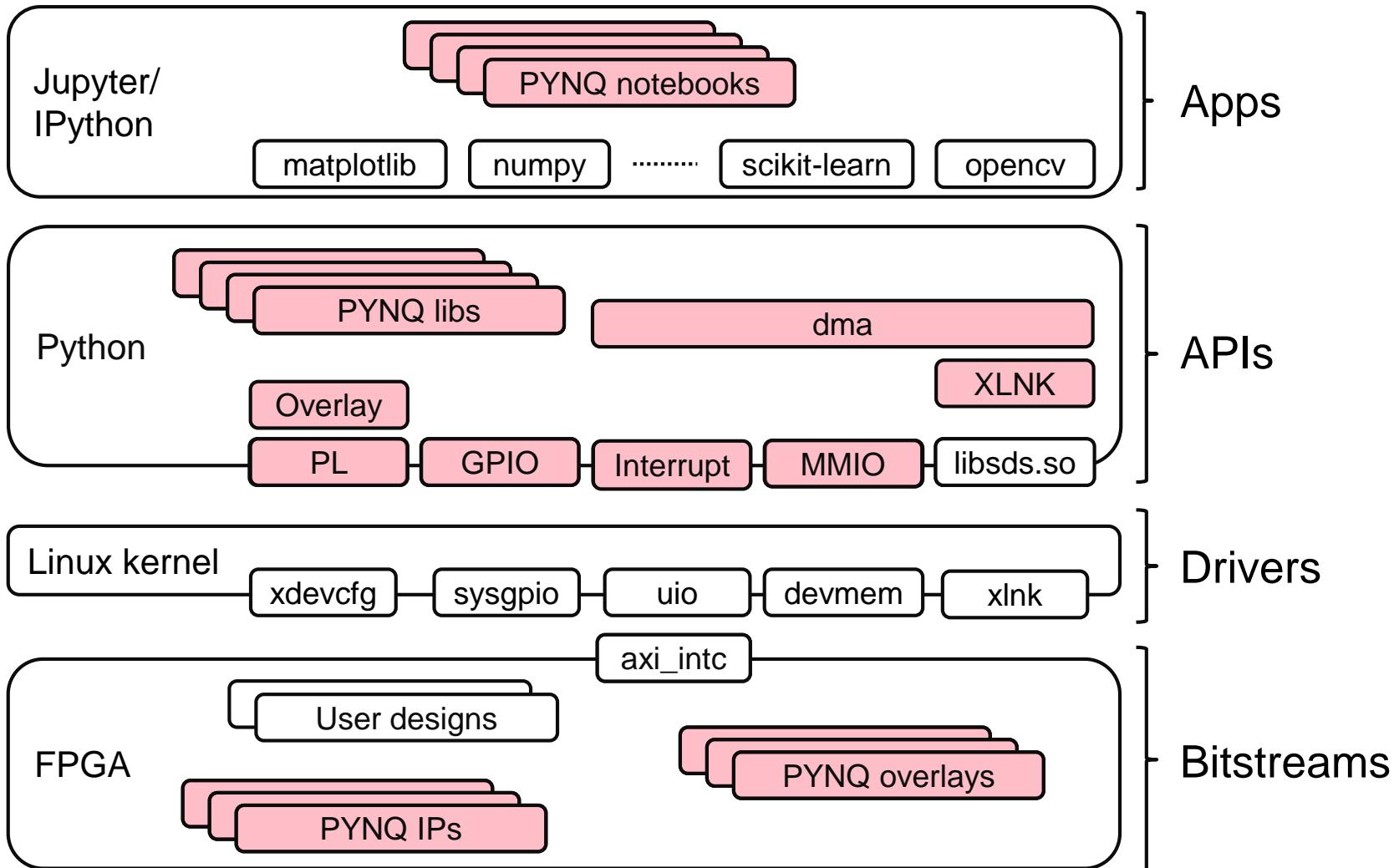
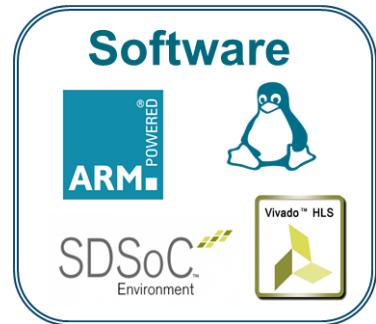
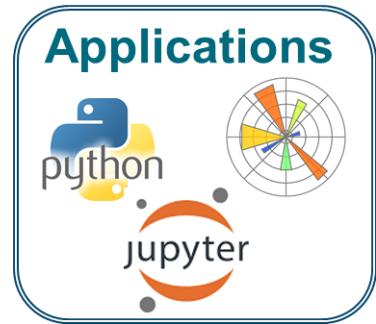
- Access to all Xilinx kernel patches
- Works with any Xilinx supported board
- Configured with additional drivers for PS-PL interfaces

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

wrapped in Python Libraries

Zynq



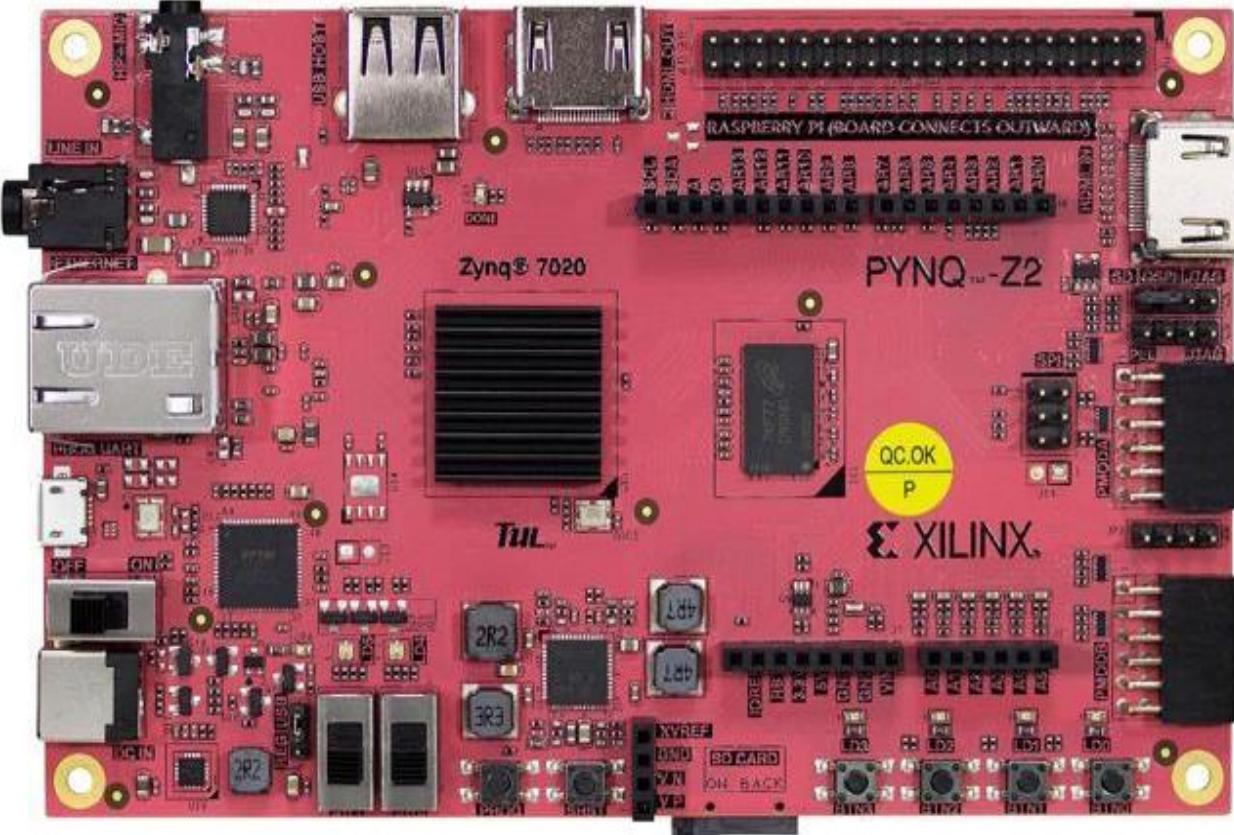




PYNQ-Z2 Board



PYNQ-Z2 Board



	PYNQ-Z2
Device	Zynq Z7020
Memory	512MB DDR3
Storage	MicroSD
Video	HDMI In & Out
Audio	ADAU1761 codec with HP + Mic, Line in
Network	10/100/1000 Ethernet
Expansion	USB host (PS)
GPIO	1x Arduino Header 2x Pmod* 1x RaspberryPi header*
Other I/O	6x user LEDs 4x Pushbuttons 2x Dip switches
Dimensions	3.44" x 5.51" (87mm x 140mm)
Webpage	TUL PYNQ-Z2 webpage

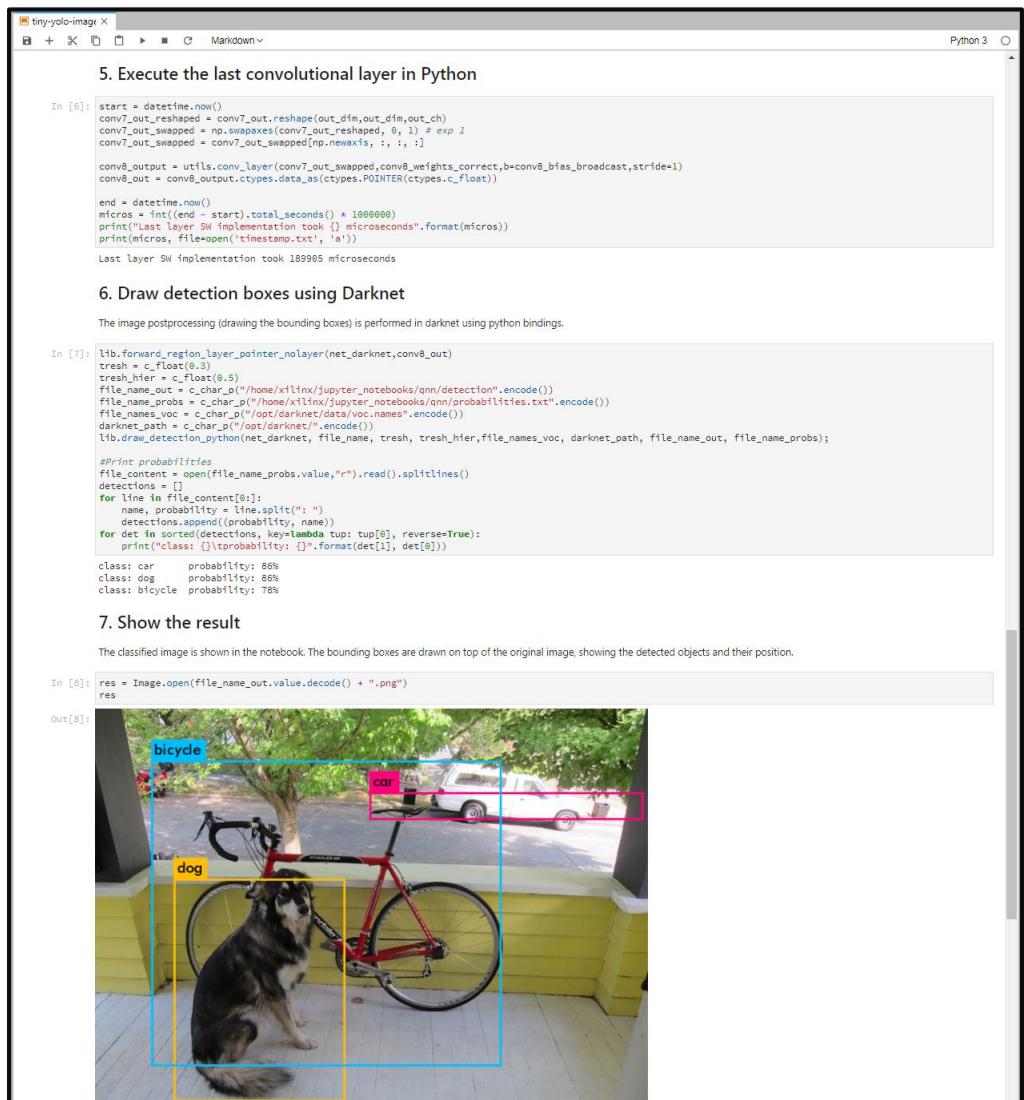
*PYNQ-Z2 RaspberryPi header shares 8 pins with 1 Pmod



Jupyter Notebook Interface



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);

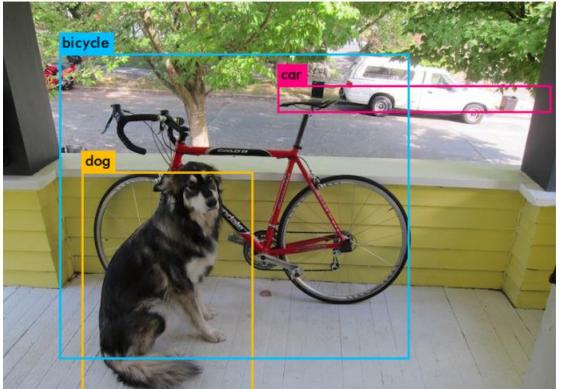
#print(probabilities
file_content = open(file_name_probs.value.decode(),"r").readlines()
detections = []
for line in file_content[0:]:
    name, probability = line.split(": ")
    detections.append((probability, name))
for det in sorted(detections, key=lambda tup: tup[0], reverse=True):
    print("class: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

Jupyter Notebook

```
In [1]: from pynq import PL, Overlay  
from pynq.iop import PMODB, Pmod_OLED  
  
In [2]: ol = Overlay("base.bit")  
ol.download() # programs the Zynq FPGA  
  
In [3]: oled = Pmod_OLED(PMODB)  
  
In [4]: oled.write("1 2 3 4 5 6")
```

6 lines of user code ... thanks to Python, FPGA overlays,
abstraction & re-use

PYNQ Overlays



FPGA overlays – hardware libraries (kernels)

> Overlays are generic FPGA designs that target multiple users with new design abstractions and tools

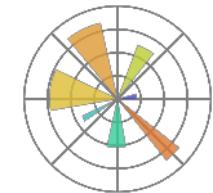
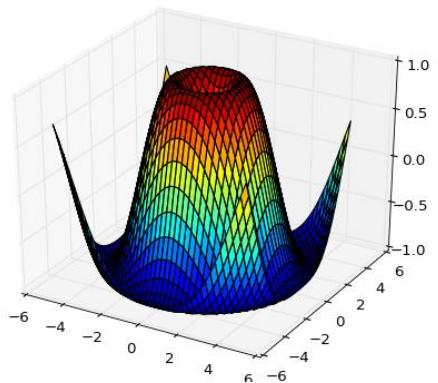
- Post-bitstream programmable via software APIs
- Typically optimized for given domains
- Encourages the use of open source tools & fast compilation
- Enables productivity from re-using pre-optimized designs
- Exposes benefits of FPGAs to new users

> Active research area with many papers

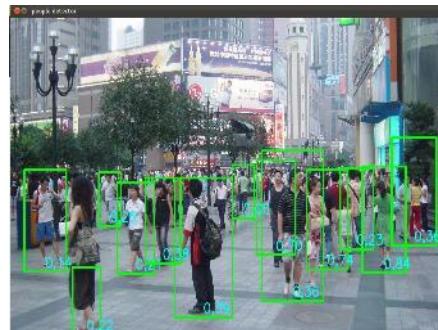
PYNQ Packages



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 ✓

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 / Bot-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.

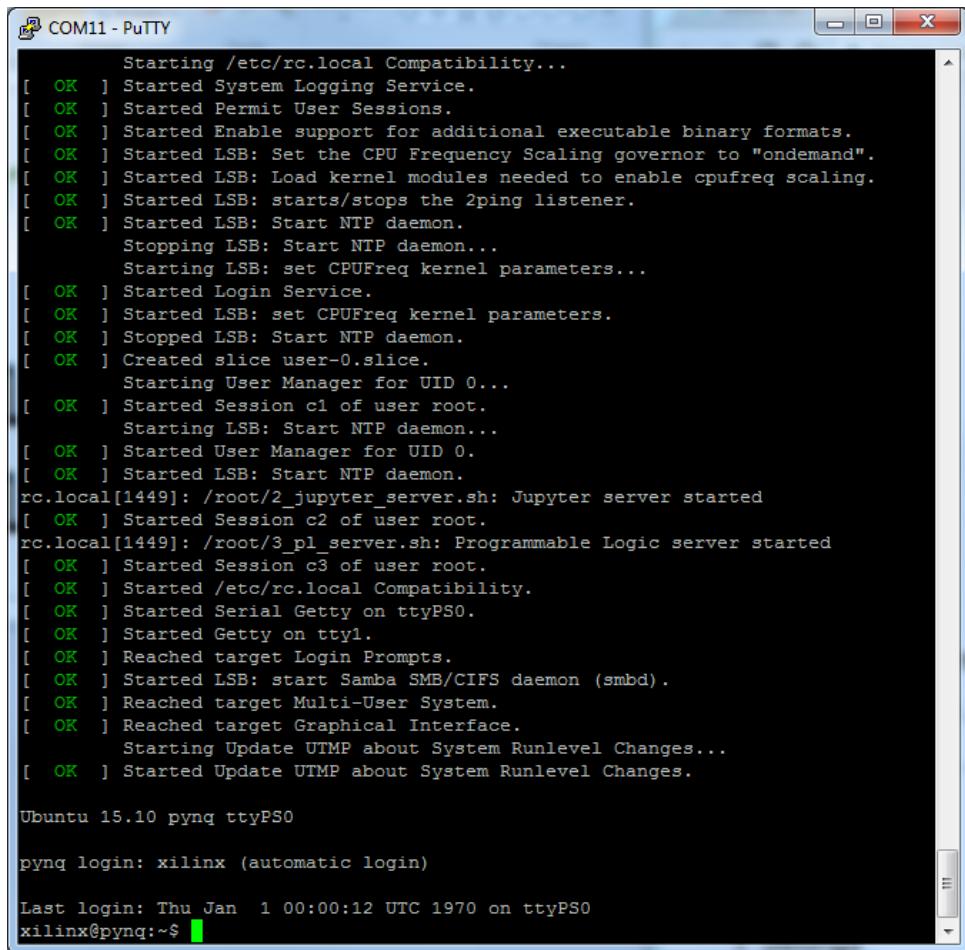
Download a design from GitHub with a single Python command:

```
pip3.6 install git+https://github.com/Xilinx/PYNQ-DL.git
```

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.

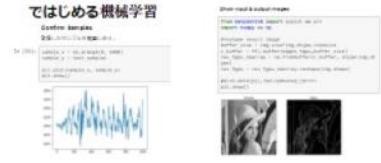
Binary Neural Network
Xilinx Labs, NTNU,
University Sydney



SPynq:
NTUA Greece



Processing noisy filters
PYNQ Japan user group



Soft GPU for ZC706
Ruhr University Bochum



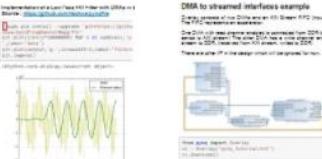
Video Processing
VectorBlox



FIR filter example
CU Boulder



DMA and stream
Tutorial



CNN Example
Imperial College London



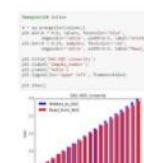
Example Notebooks

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 ✓

Join the 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](#)

This example shows how to use Python to implement the Binary Neural Network architecture on a Zynq SoC using the PYNQ image.



SPyng:

NTUA Greece

[SPyng: Python interface](#)

In this project we propose Python to interface the Zynq SoC after running PyPynq and run code from the host PC.

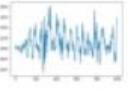


Processing noisy filters

PYNQ Japan user group

[ではじめる機械学習](#)

この例題は、Pythonで機械学習を実装するための基礎知識を学ぶためのものです。主な機能は、データの読み込みと表示、学習アルゴリズムの実装、モデルの評価などです。



Soft GPU for ZC706

Ruhr University Bochum

[Soft GPU for ZC706](#)

This example shows how to use Python to implement a soft GPU for the ZC706 Evaluation Board.



Video Processing

Vectorblobx

[VectorBlox Video Processing](#)

In this module, several filters can be applied to YUV input images. These filters are: median, average, blur, crop, rotate, flip, and threshold. All of these filters are implemented in 16000 pixel width and 1600 pixel height.



FIR filter example

CU Boulder

[Implementation of a Linear Phase FIR Filter with DMA via IP](#)

The FIR filter example demonstrates how to implement a linear phase FIR filter using DMA. The filter has a cutoff frequency of 0.4*pi rad/sample. The filter is implemented using a direct form II structure.



DMA and stream

Tutorial

[DMA to streamified interface example](#)

Starting off with the DMA and an AXI Stream IP Core on a Zynq SoC, this example shows how to implement a streamified interface between the two cores. The DMA core receives data from a camera and the AXI Stream IP core sends data to a monitor.



CNN Example

Imperial College London

[FPGA Deep learning CNN example](#)

This example shows how to implement a CNN on a Zynq SoC using the Keras API. The network consists of three layers: a convolutional layer, a max pooling layer, and a fully connected layer.



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](#)

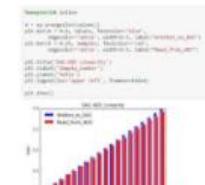
This example shows how to generate ADC waveforms on the Zynq SoC.



DAC ADC example

[DAC ADC example](#)

This example shows how to generate DAC waveforms and read them back using the ADC.



Downloading overlays

[Downloading Overlays](#)

This example shows how to download an FPGA overlay and use it.

1. Installing an overlay
To install an overlay, you need to have the overlay source code present on the host drive. This example file does not need to be split if it is present in one single package. It is recommended to use a zip file for the overlay source code.

2. Bringing the overlay to the Zynq SoC
In the notebook, we can use Jupyter to push the overlay to the Zynq SoC. A kernel will automatically be created for the Zynq SoC. To bring the overlay to the Zynq SoC, we can use the command:

Grove ADC

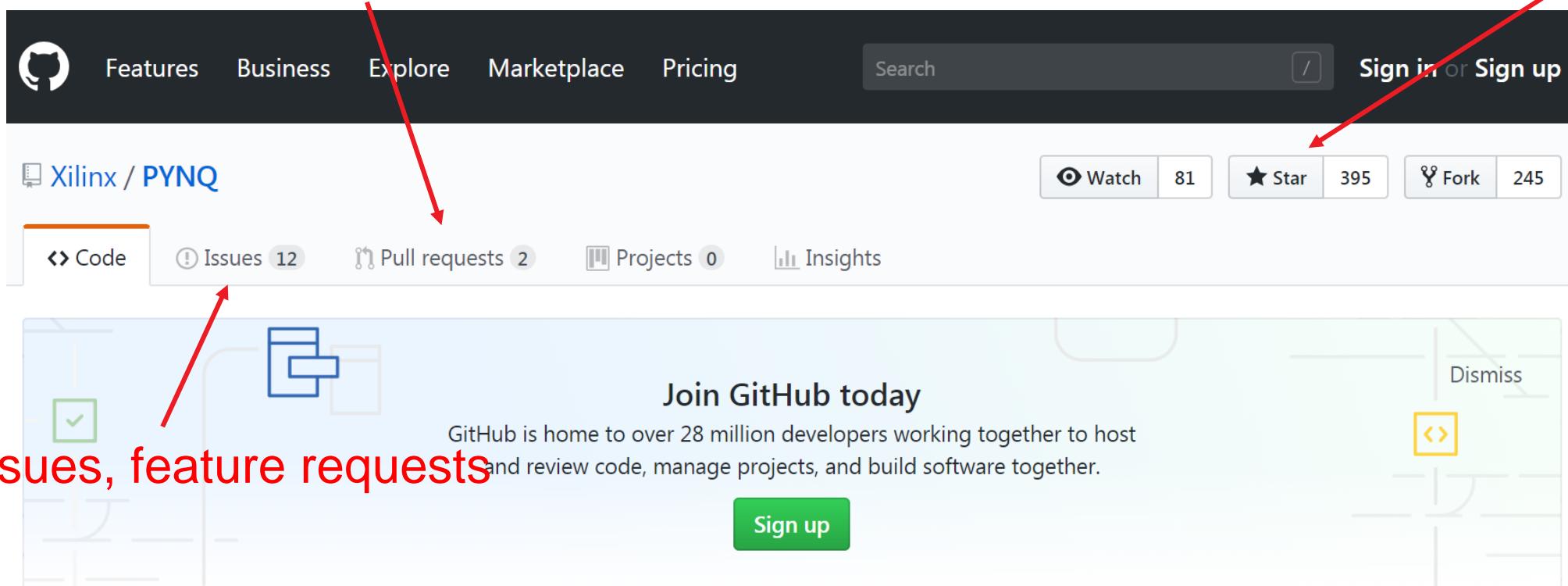
[Grove ADC](#)

This example shows how to read a single value from a Grove ADC.



All Feedback helps

Contribute



Issues, feature requests

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

pynq

>> 30

XILINX INTERNAL

XILINX

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.**

