

# GoPynq!

## Python Productivity for Zynq





**What is Pynq and who is it for?**



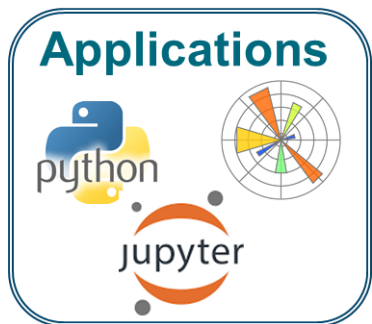
# What is Pynq and who is it for?

- > PYNQ is an open-source framework from Xilinx® that makes it easy to design embedded systems with Xilinx Zynq® Systems on Chips (SoCs).
- > Using the Python language and libraries, designers can exploit the benefits of programmable logic and microprocessors in Zynq to build more capable and exciting embedded systems.

PYNQ is intended to be used by a wide range of designers and developers including:

- > Software developers who want to take advantage of the capabilities of Zynq and programmable hardware without having to use ASIC-style design tools to design hardware.
- > System architects who want an easy software interface and framework for rapid prototyping and development of their Zynq design.
- > Hardware designers who want their designs to be used by the widest possible audience.

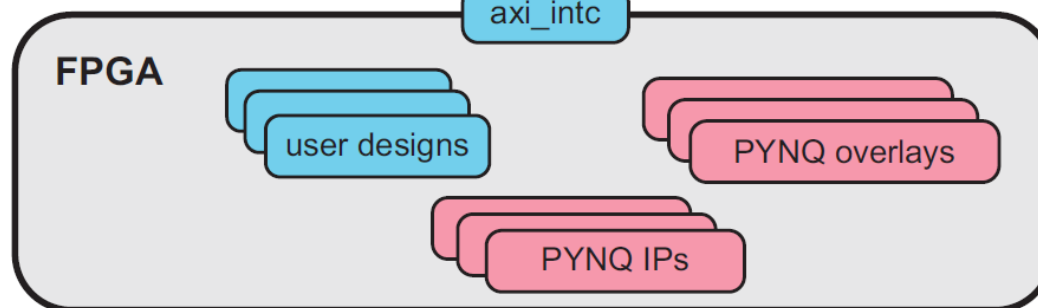
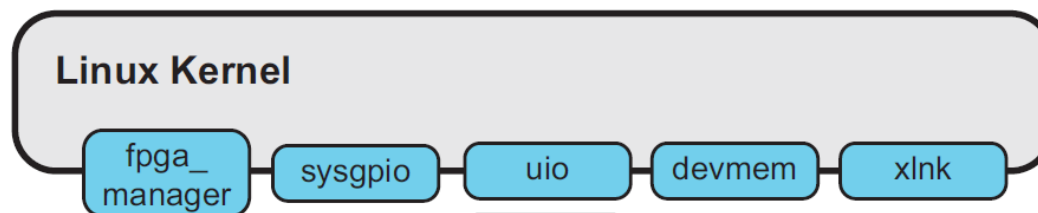
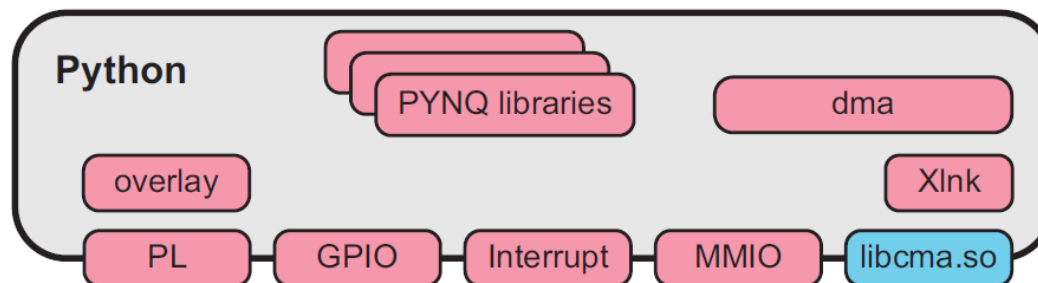
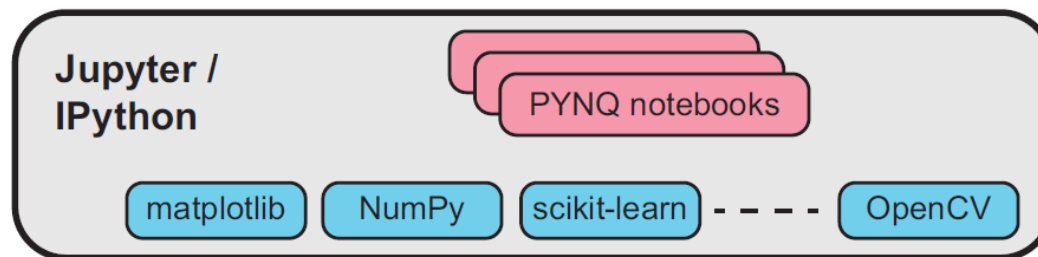
# PYNQ™ is a Framework



Applications

Software

Hardware



Apps

APIs

Drivers

Bitstreams

PYNQ™

# Technologies used by Pynq

- > Jupyter Notebook is a browser based interactive computing environment.



- > A PYNQ enabled Zynq board can be easily programmed in Jupyter Notebook using Python.



- > Using Python, developers can use hardware libraries and overlays on the programmable logic.



- > The PYNQ image is a bootable Linux image and includes the **pynq** Python package, and other open-source packages.



# Jupyter Notebook Interface



# Jupyter Notebooks ... the engine of data science

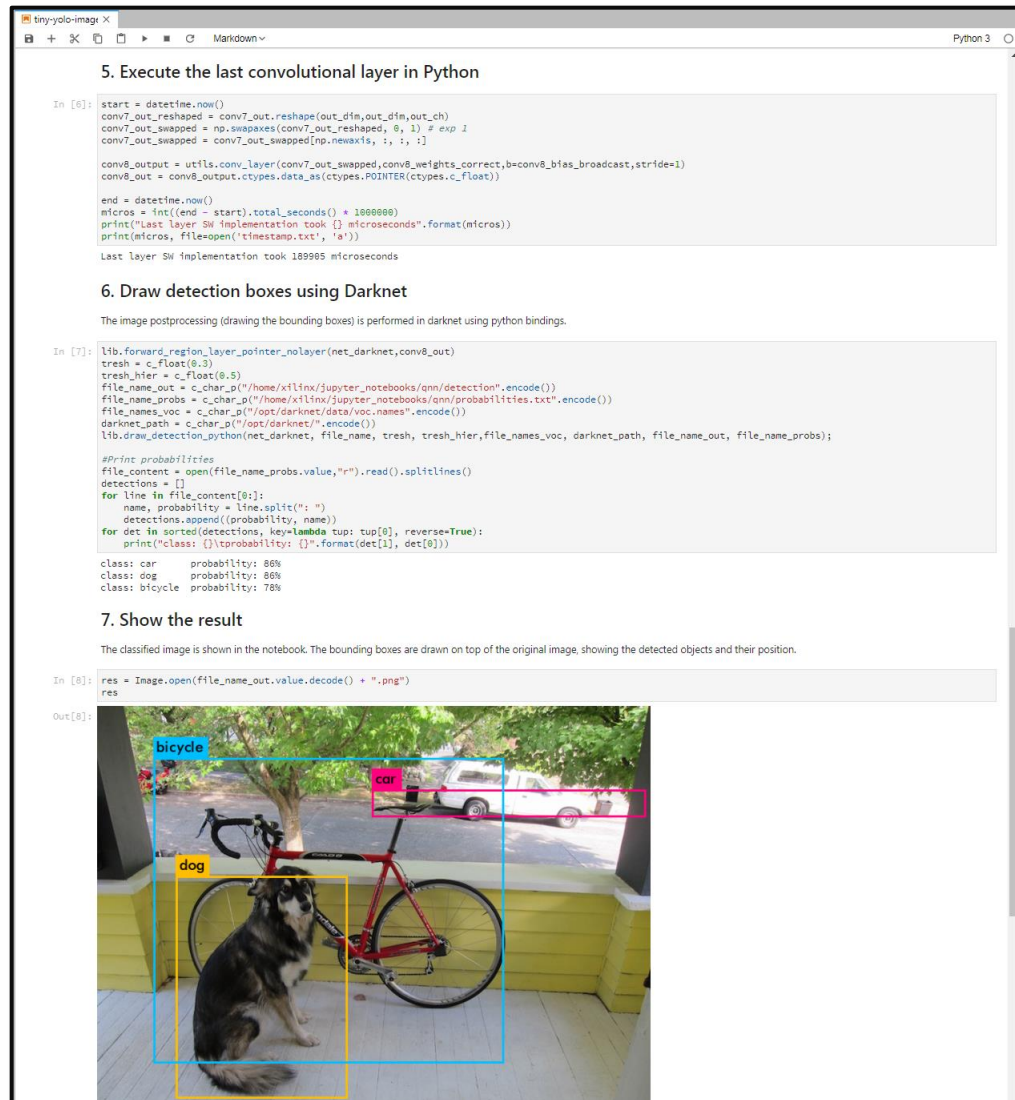


Open source browser-based, executable documents

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

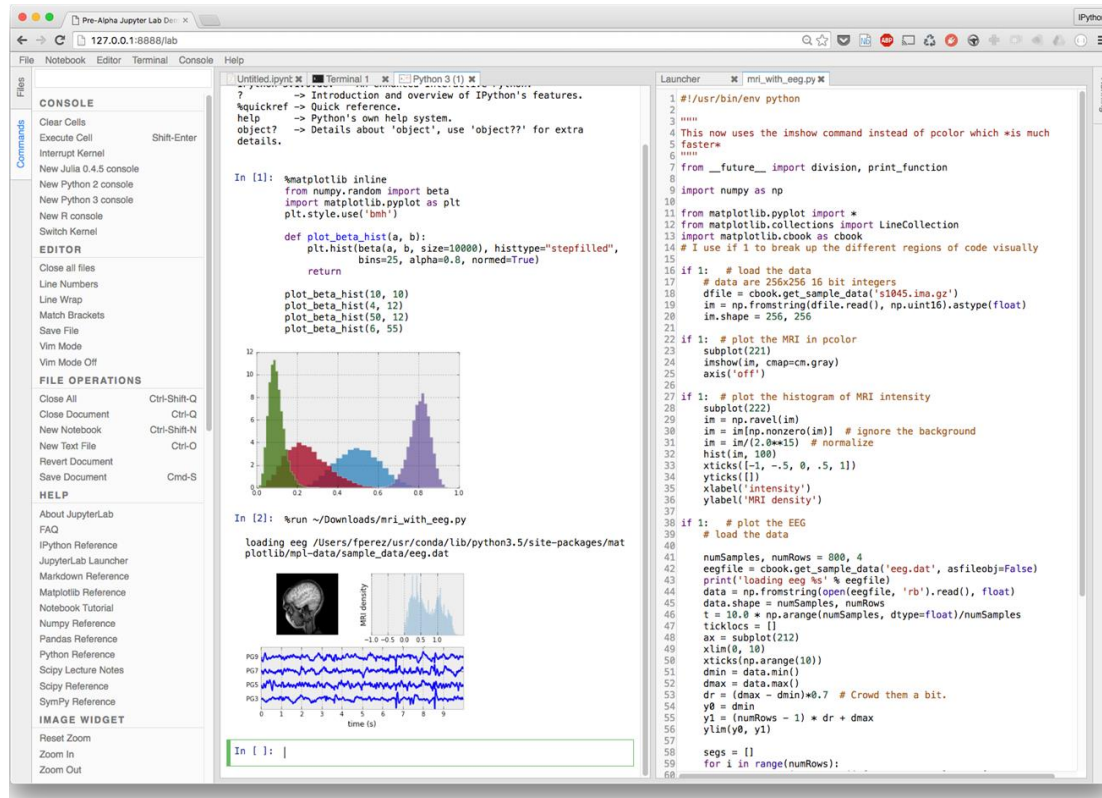
1.7 million notebooks on GitHub

Taught to 1,000+ Berkeley data science students

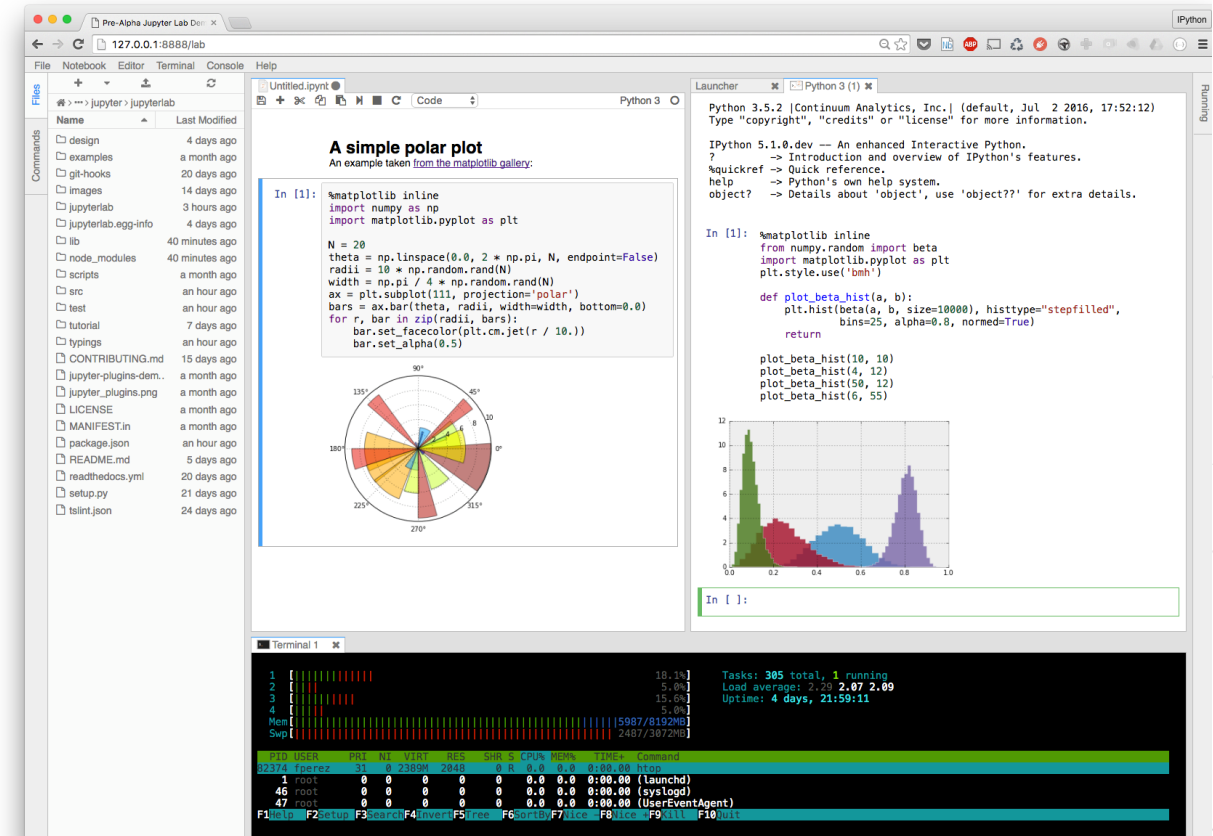


# JupyterLab: web-based IDE incl. Notebooks

Add this slide



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



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



# 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)

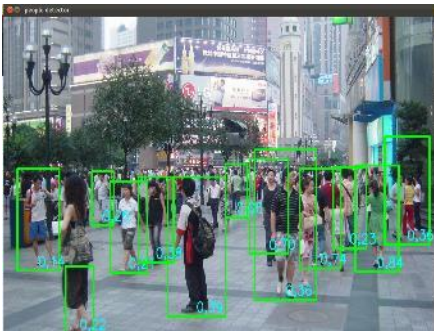
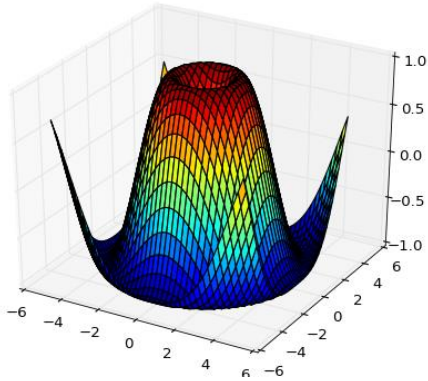
- > Overlays are generic FPGA designs that target multiple users with new design abstractions and tools
  - Post-bitstream programmable via software APIs
  - Typically optimized for given domains
  - Encourages the use of open source tools & fast compilation
  - Enables productivity from re-using pre-optimized designs
  - Exposes benefits of FPGAs to new users
- > Active research area with many papers



## PYNQ Packages



# Python packages for data analysis and visualisation



> 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 image displays four screenshots of GitHub repositories from Xilinx, illustrating the software-style packaging and distribution of designs using hybrid packages.

- QNN-MO-PYNQ:** Shows a notebook titled "3. Open image to be classified" with Python code for image processing and a resulting image of a dog.
- SPYN:** Shows a notebook titled "SPYN - III phase AC motor control" with objectives, a step-by-step guide, and a scatter plot.
- PYNQ-DL:** Shows a notebook titled "Resizing an image" with a hardware block diagram illustrating the data flow between PS (ARM), PL (Resize IP), and DRAM, connected via AXI Interconnect and DMA.
- PYNQ-ComputerVision:** Shows a notebook titled "OpenCV Overlay: Filter2D and Dilate" with code for image processing and a resulting image of a dog.

Download a design from GitHub with a single Python command:

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

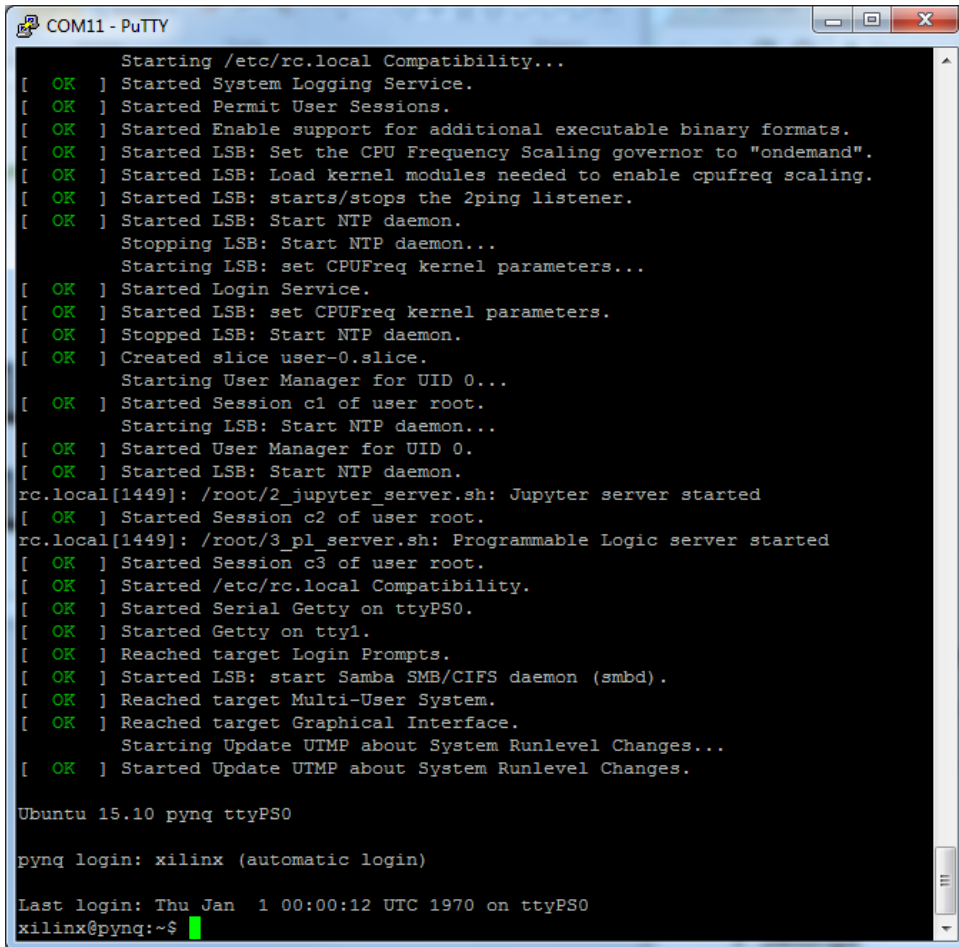


## Benefits of PYNQ





# Start using PYNQ out-of-the-box



```
COM11 - PuTTY
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 on [www.pynq.io](http://www.pynq.io)

## PYNQ community projects

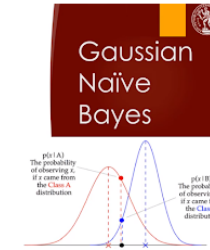
**Ultra96 Facial Recognition Deadbolt Using PYNQ**  
Julian Bartolone



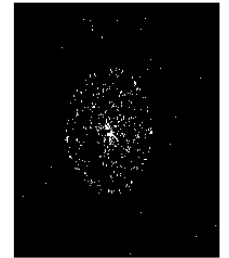
**PYNQ Controlled NeoPixel LED Cube**  
Adam Taylor



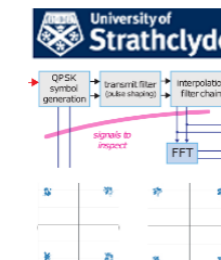
**Gaussian Naive Bayes**  
Giorgos Tzanos, NTUA



**N-Particle Gravity Simulation on Ultra96**  
Rajeev Patwari, Nathalie Chan King Choy



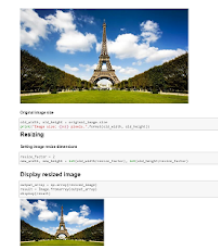
**PYNQ RFSoc**  
University Strathclyde  
QPSK demo on ZCU111



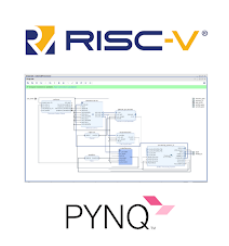
**PYNQ-PRIO**  
BYU  
Partial reconfiguration Input/Output



**PYNQ Hello World**  
Hardware accelerated image resizer example



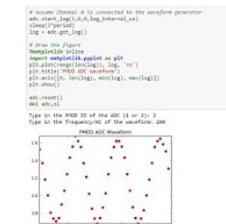
**RISC-V on PYNQ**  
UCSD



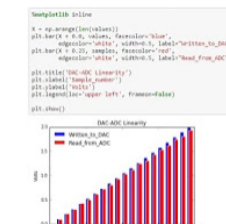
## Example Notebooks

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



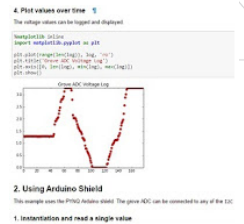
### DAC ADC example



### Downloading overlays

**Downloading Overlays**  
This notebook demonstrates how to download an FPGA overlay and execute it on the PYNQ board. The first step is to download the overlay from the PYNQ community. The second step is to load the overlay onto the PYNQ board. The third step is to execute the overlay. The fourth step is to view the results of the execution.

### Grove ADC



# Simplified 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!



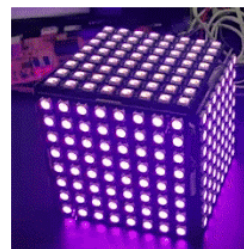


## Community Projects PYNQ community projects

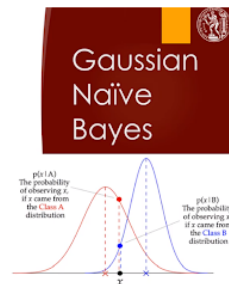
**Ultra96 Facial Recognition Deadbolt Using PYNQ**  
Julian Bartolone



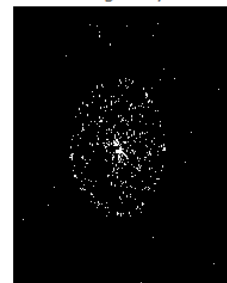
**PYNQ Controlled NeoPixel LED Cube**  
Adam Taylor



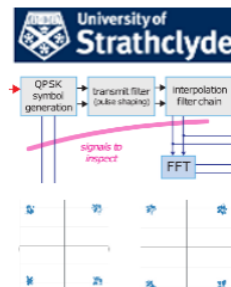
**Gaussian Naive Bayes**  
Georgios Tzanos, NTUA



**N-Particle Gravity Simulation on Ultra96**  
Rajeev Patwari, Nathalie Chan King Choy



**PYNQ RFSoc**  
University Strathclyde  
QPSK demo on ZCU111



**PYNQ-PRIO**  
BYU  
Partial reconfiguration Input/Output

byucl / PYNQ-PRIO

Code Issues Pull requests

gakint Update setup.py

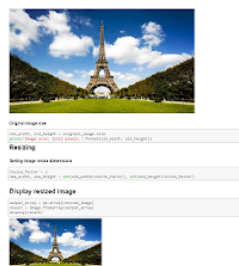
```
overlay.set_partial_region('pr_0')
overlay.download(PARTIAL_BITSTREAM_PATH)

overlay.set_partial_region('pr_3')
overlay.download(PARTIAL_BITSTREAM_PATH)

overlay.set_partial_region('pr_2')
overlay.download(PARTIAL_BITSTREAM_PATH)

overlay.set_partial_region('pr_3')
overlay.download(PARTIAL_BITSTREAM_PATH)
```

**PYNQ Hello World**  
Hardware accelerated image resizer example



**RISC-V on PYNQ**  
UCSD



# All Feedback helps

Contribute

If you like it, star it!

The screenshot shows the GitHub repository page for Xilinx / PYNQ. The repository has 81 Watchers, 395 Stars, and 245 Forks. The navigation bar includes links for Features, Business, Explore, Marketplace, and Pricing, along with a search bar and Sign in / Sign up buttons. The repository tabs show Code, Issues (12), Pull requests (2), Projects (0), and Insights. A red arrow points from the word 'Contribute' to the 'Issues' tab. Another red arrow points from the text 'If you like it, star it!' to the 'Star' button. A third red arrow points from the text 'Issues, feature requests' to the 'Issues' tab. A green 'Sign up' button is visible in the 'Join GitHub today' banner.

Xilinx / PYNQ

Watch 81 Star 395 Fork 245

Code Issues 12 Pull requests 2 Projects 0 Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

Issues, feature requests

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

pynq

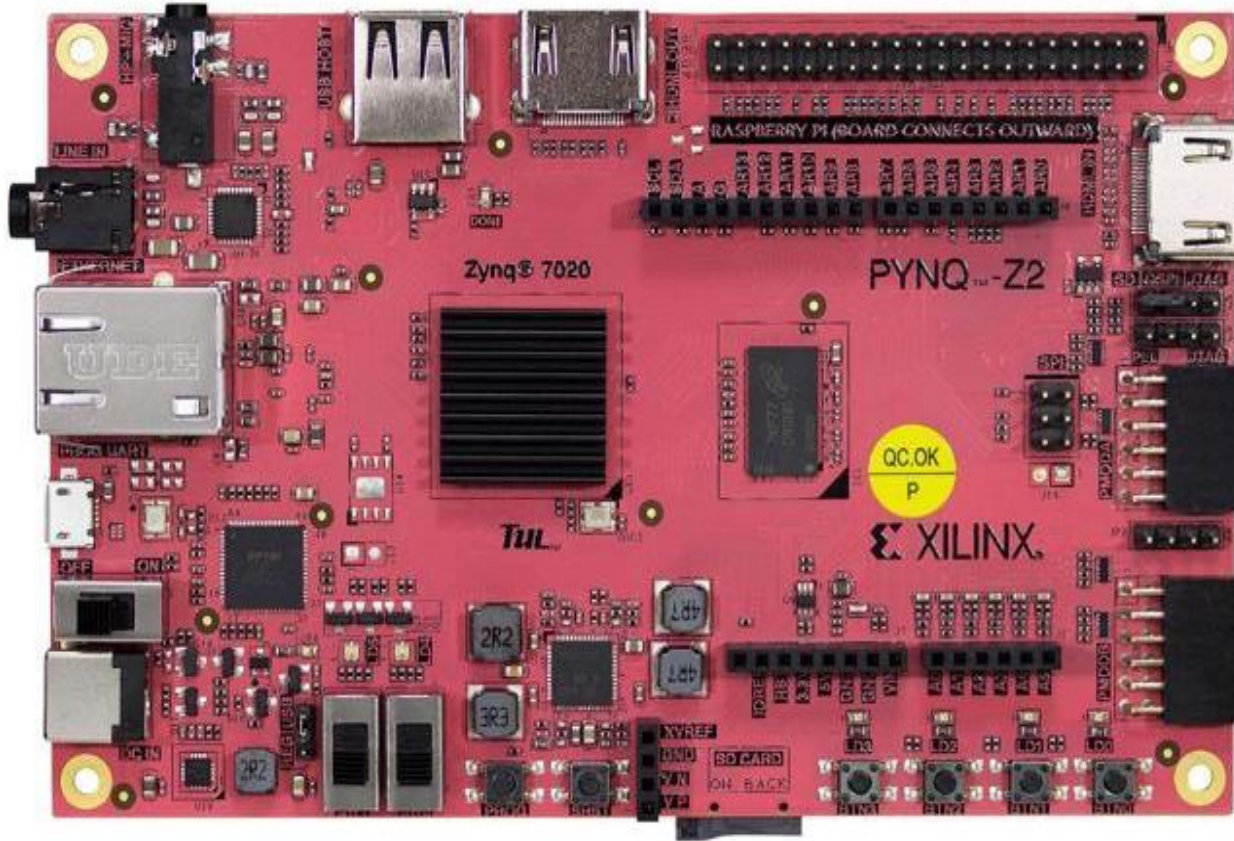


## Boards for the competition





# PYNQ-Z2 Board



- > Pynq – Z2 board supports Pynq and numerous other I/O abilities

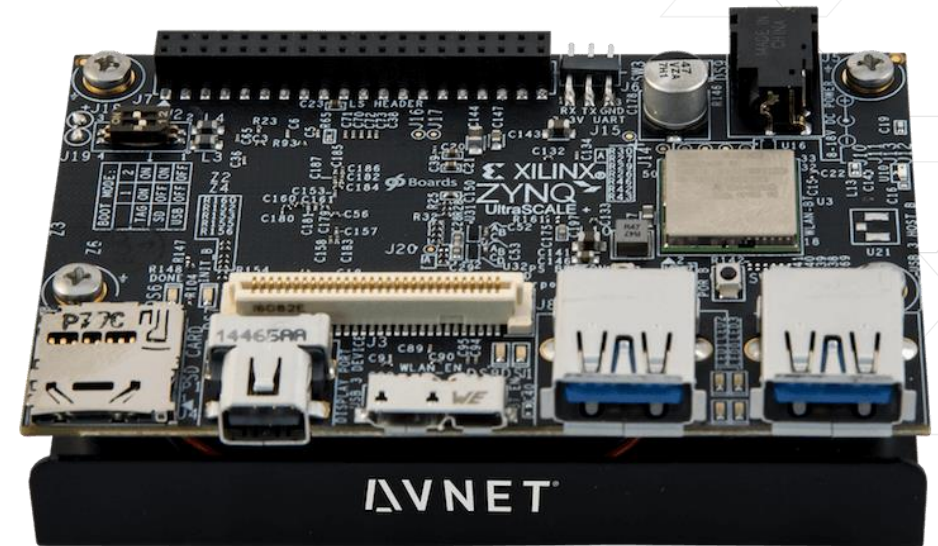
	PYNQ-Z2
Device	Zynq Z7020
Memory	512MB DDR3
Storage	MicroSD
Video	HDMI In & Out
Audio	ADAU1761 codec with HP + Mic, Line in
Network	10/100/1000 Ethernet
Expansion	USB host (PS)
GPIO	1x Arduino Header
	2x Pmod*
	1x RaspberryPi header*
Other I/O	6x user LEDs
	4x Pushbuttons
	2x Dip switches
Dimensions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

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



# Ultra96

- > Ultra96 is an Arm-based, Xilinx Zynq UltraScale+™ MPSoC development board by Avnet based on the Linaro 96Boards Consumer Edition (CE) specification
- > 96Boards is a range of hardware specifications created by Linaro to make the latest ARM-based processors available to developers at a reasonable cost.
- > Supports Pynq : <https://ultra96-pynq.readthedocs.io/en/latest/index.html>



# Ultra96 and Mezzanine

- > Ultra96 can be easily paired with Mezzanine board which come with a wide variety of I/O capabilities.
- > Sensor Mezzanine board (in the picture to the right) can be stacked on the Ultra96 board and a multitude of peripherals can be interfaced through it.
- > LCD, touch sensor, sound sensor, rotatory angle sensor, light sensor, buzzer, temperature sensor are examples of a few sensors that can be used with the Sensor Mezzanine board.
- > Visit the github page for examples and details:  
[https://github.com/96boards/Sensor\\_Mezzanine\\_Getting\\_Started](https://github.com/96boards/Sensor_Mezzanine_Getting_Started)



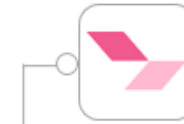
# Links and Documentation

- > Pynq can also be used on the Ultra96 board. Here is a related video from 96Boards:  
<https://www.youtube.com/watch?v=ptzrg9dPI3w>
- > This link provides more information about the Ultra96 board:  
<http://zedboard.org/product/ultra96>
- > The official github page for documentation is here:  
<https://github.com/96boards/documentation/tree/master/consumer/ultra96>
- > To know more about Ultra96, use 96boards website:  
<https://www.96boards.org/product/ultra96/>

# Summary



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



[pynq.io](https://pynq.io)



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



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



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



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

**Adaptable.**  
**Intelligent.**

