

# Python Productivity on ZYNQ & SPYN

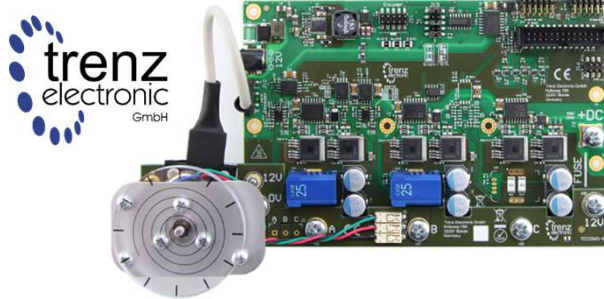
Kiran Vishal Thanjavur Bhaaskar  
Industrial IoT Solutions Architect  
Industrial, Vision, Healthcare & Sciences  
June 2019, Japan



# Recap: Electric Drives Demonstration Platform

- > Design Methodology Predicated on Open Source & Ease of Use
- > EDDP takes complete advantages of **Xilinx Zynq SoCs**
- > Platform offers two different flows to build motor control solutions
  - >> Xilinx SDSoC Design Flow
  - >> Xilinx Vivado HLS Design Flow

## EDDP KIT



1

2

3

Three simple steps  
to get started!  
[EDDP Demo Video](#)

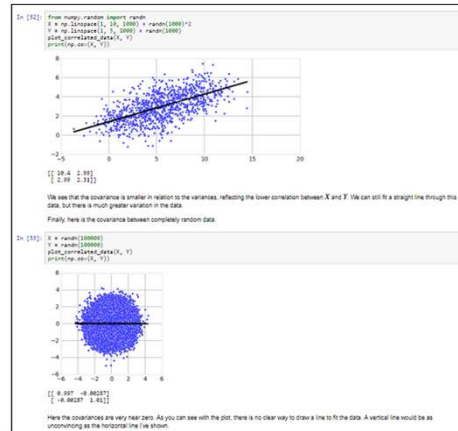
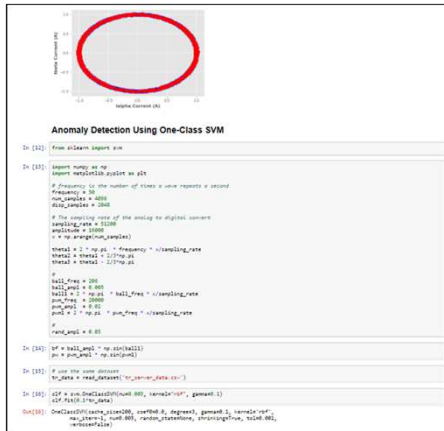


# Recap: Intelligent Motor Control



## > Python productivity for Extreme Edge Analytics – Cloud Compute at the Edge

>> Anomaly Detection for Predictive Maintenance via Numpy and Scikit-learn, many other capabilities













## > PYNQ

>> Visit [pynq.io](http://pynq.io) for more information on the PYNQ Framework and the SPYN Motor Control Design

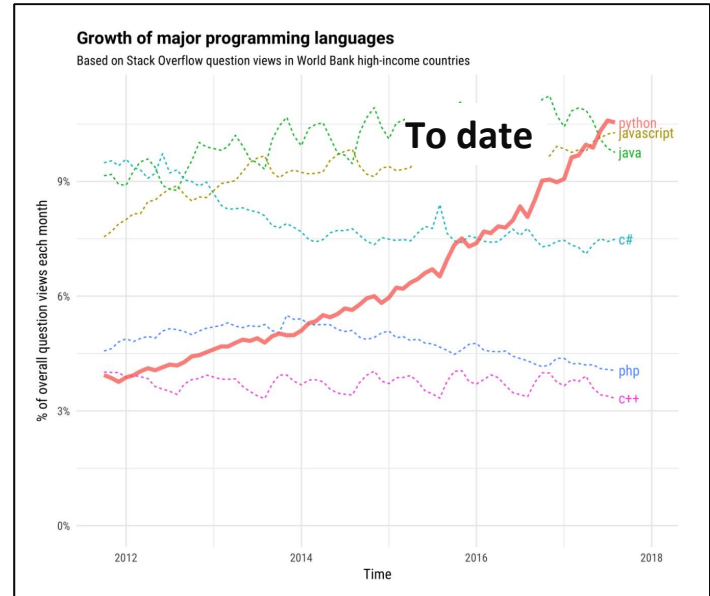
# Python is increasingly the Language of Choice

## Top Programming Languages, IEEE Spectrum, July'18

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		98.4
3. C		98.2
4. Java		97.5
5. C#		89.8
6. PHP		85.4
7. R		83.3
8. JavaScript		82.8
9. Go		76.7
10. Assembly		74.5

Python is listed as an  
embedded language  
for the first time

<https://spectrum.ieee.org/at-work/innovation/the-2018-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

# PYNQ: Python Productivity for Zynq



# PYNQ™ Python Productivity on Zynq



Domain Experts  
Software Engineers

New users are not always hardware designers,  
or embedded systems designers

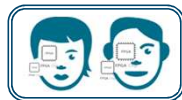


Embedded software  
Engineers

PYNQ™

*Enables more people to program Xilinx  
processing platforms, more productively*

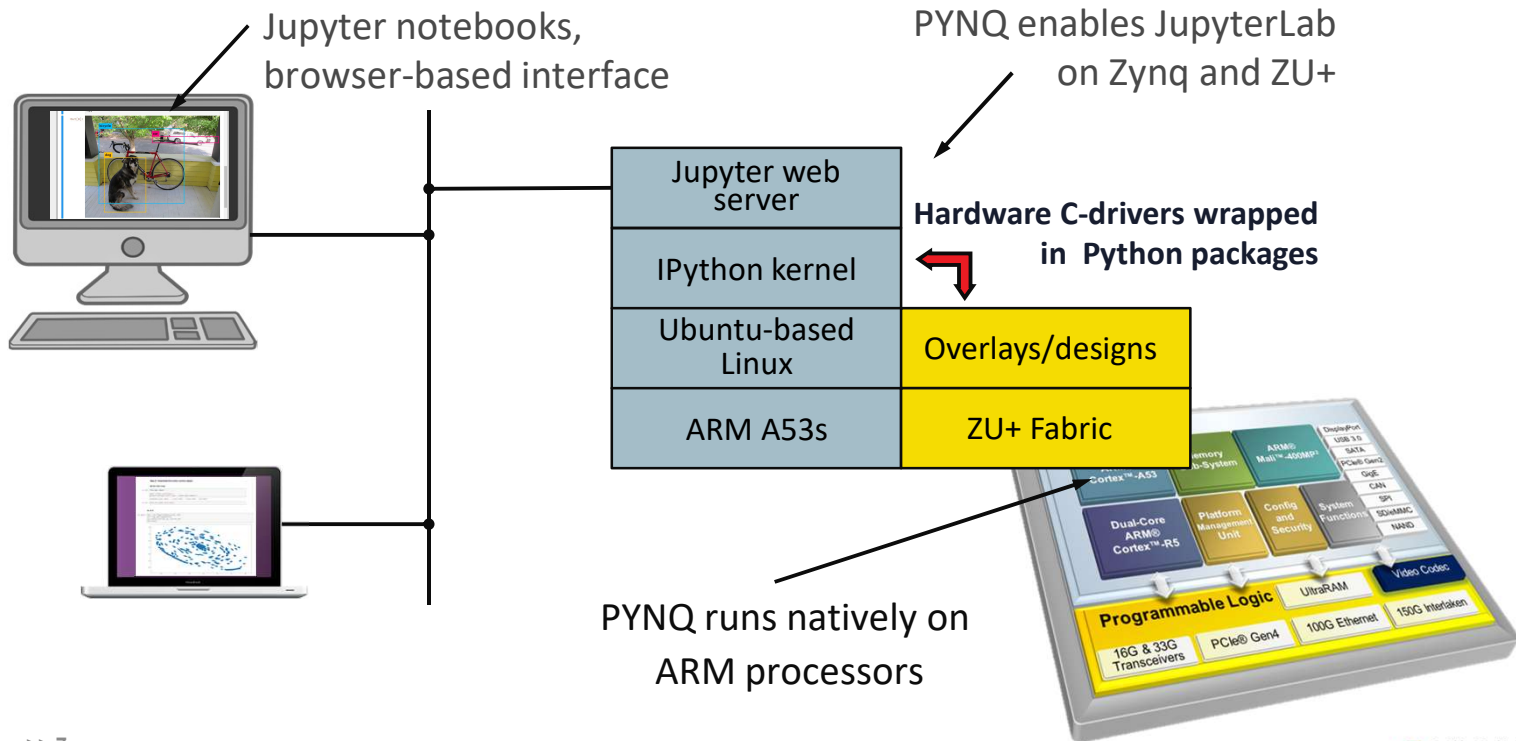
**AND**



Hardware  
Engineers

*Offers more rapid development for h/w designers  
and embedded s/w engineers*

# Python productivity for Zynq



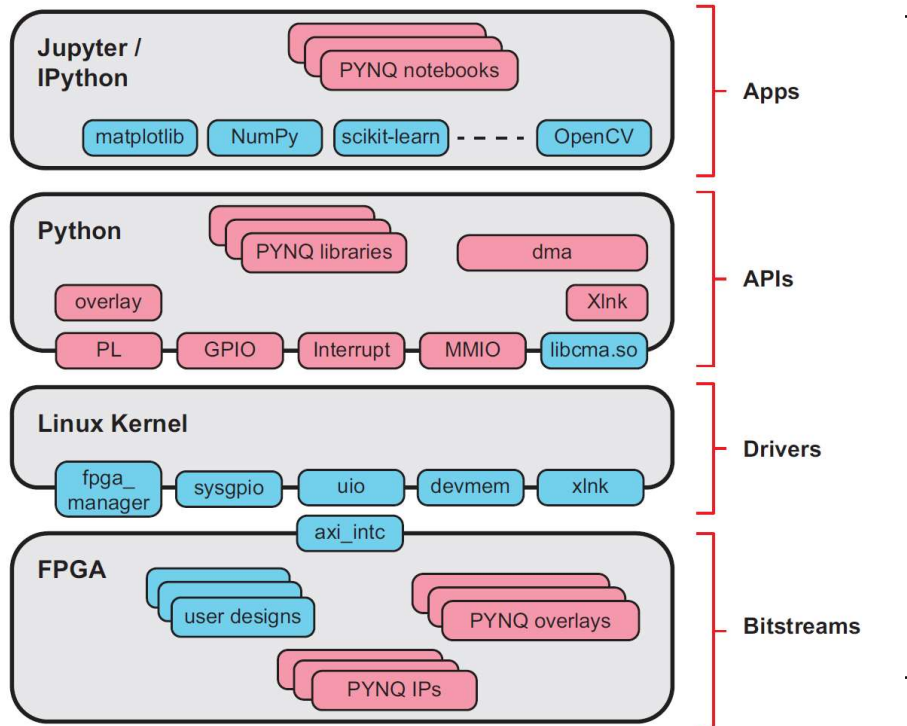
# PYNQ™ is an open source Framework



Applications

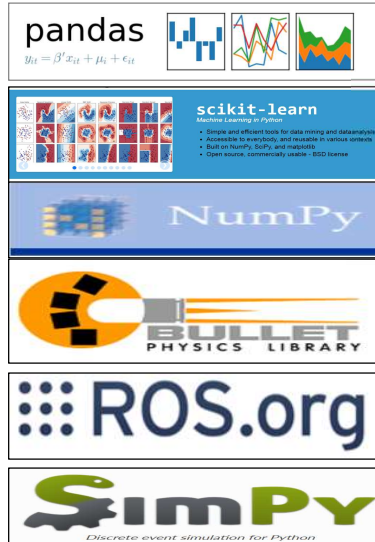
Software

Hardware



PYNQ™

# Python Powered: Machine Learning & Analytics Libraries



- > SPYN enabled by PYNQ can support all of these libraries
- > Many of the libraries are extensively used in:
  - >> Data manipulation
  - >> Machine Learning
  - >> Analytics
- > Many more libraries available!!

# Software-style packaging & distribution of designs

Enabled by new *hybrid packages*

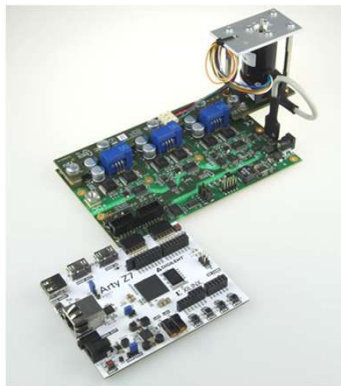
The image displays four screenshots of the Xilinx PYNQ IDE interface, showcasing different hybrid packages available for download and execution on a PYNQ board.

- SPYN - III phase AC motor control:** This package is designed for controlling a 3-phase AC motor using the ESP8266 (Chao Electric Drive Power Stage (EDPS) Board). It includes a search bar, project details, and a code snippet for downloading the design.
- PYNQ-DL:** This package is for deep learning applications. It features a search bar, project details, and a code snippet for downloading the design.
- PYNQ-ComputerVision:** This package is for computer vision applications. It includes a search bar, project details, and a code snippet for downloading the design.
- PYNQ-OpenCV:** This package is for OpenCV applications. It features a search bar, project details, and a code snippet for downloading the design.

Download a design from GitHub with a single Python command:

```
pip install git+https://github.com/Xilinx/pynq-helloworld.git
```

# SPYN: Bridging two worlds for an Intelligent Drive



**EDDP**

Electric Drives Demo Platform

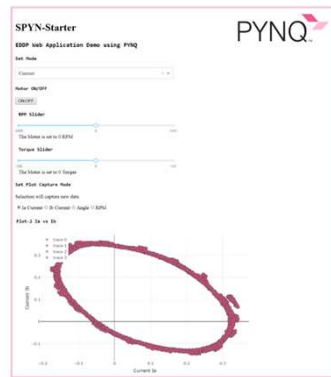
+



**PYNQ**

Python Productivity for Zynq

=



**SPYN**

Extreme Edge Analytics  
for Motor Control

- > **SPYN takes advantage of both EDDP Kit & PYNQ framework**
- > **EDDP kit can also be used to test, modify & build SPYN project at no additional charge**
- > **The solution enables python powered machine learning & edge analytics for motor control**
- > **Python libraries are leveraged to provide UI for control, data manipulation, analytics & visualization**

# PYNQ harnesses the power of Programmable Logic

*Two most common use models for PYNQ—SPYN Showcases Both*

## > Control & Query Programmable Logic IP

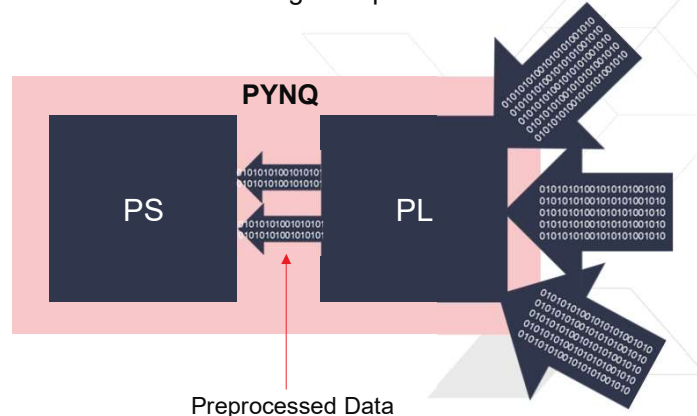
Steering wheel, pedals, gauges: PYNQ Framework



High Performance Engine:  
Programmable Logic

## > Division of labor between PS & PL

- PL to rapidly preprocess raw, streaming IO data from sensors and passes to PS
- Processor leverages extensive libraries to execute wide range of operations on data



# Python Powered: IIoT-SPYN Design Enhanced Capabilities (1 of 3)

```
motor_status = [(motor._read_controlreg(i + ANGLE.offset)) for i in
                 range(0, 16, 4)]
high_sp, low_sp = bytesplit(motor_status[1])
high_id, low_id = bytesplit(motor_status[2])
high_iq, low_iq = bytesplit(motor_status[3])

print(f'Angle in degrees : {motor_status[0] * 0.36}')
print(f'Angle in steps per thousand: {(motor_status[0])}')
print(f'Id : {np.int16(low_id) * 0.00039} Amp')
print(f'Iq : {np.int16(low_iq) * 0.00039} Amp')
print(f'Speed in RPM : {-(np.int16(low_sp))}')
```

```
Angle in degrees : 104.39999999999999
Angle in steps per thousand: 290
Id : -0.062009999999999996 Amp
Iq : 0.31785 Amp
Speed in RPM : 983
```

```
import time
motor.set_mode('rpm_mode')
for i in range(2):
    motor.set_rpm(1000)
    time.sleep(1)
    motor.set_rpm(0)
    time.sleep(2)
    motor.set_rpm(-50)
    time.sleep(2)
    motor.set_rpm(0)
    time.sleep(2)
motor.stop()
```

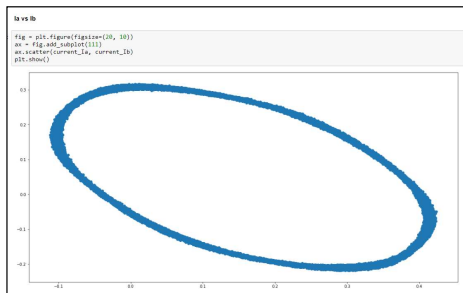
## Monitoring

Monitor the status parameters of the motor remotely anytime

## Control:

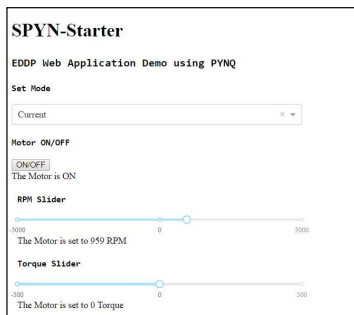
Simple python code routine to control motor

# Python Powered: IIoT-SPYN Design Enhanced Capabilities (2 of 3)



## Visualization:

Leverage matplotlib library to draw plots to visualize motor data



## Custom Interactive UI:

Custom UI to control motor & generate plots using dash by plotly

# Python Powered: IIoT-SPYN Design Enhanced Capabilities (2 of 3)

```
import pandas as pd

data = {'Ia': current_Ia,
        'Ib': current_Ib,
        'angle': cap_list[1],
        'rpm': cap_list[2]}

df = pd.DataFrame(data, columns = ['Ia', 'Ib', 'angle', 'rpm'])
df
```

	Ia	Ib	angle	rpm
0	0.09204	0.09009	846	3541
1	0.09048	0.08853	845	3541
2	0.08619	0.09126	845	3527
3	0.08424	0.09321	845	3527
4	0.08307	0.09945	844	3527
5	0.08502	0.09789	844	3527
6	0.08697	0.09711	843	3527
7	0.09126	0.09594	843	3527
8	0.09360	0.09204	843	3527
9	0.08892	0.09282	842	3527
10	0.08463	0.09204	842	3527

## Analytics & ML Ready Data Format:

Acquired Motor Data is stored  
as pandas data frames  
providing training data for ML &  
Analytics

## Installable & Upgradable packaging :

```
root@kv_pynq:/home/xilinx# sudo pip3.6 install --upgrade git+https://github.com/Xilinx/IIoT-SPYN.git
Collecting git+https://github.com/Xilinx/IIoT-SPYN.git
  Cloning https://github.com/Xilinx/IIoT-SPYN.git to /tmp/pip-ge20fs95-build
Installing collected packages: spyn
  Found existing installation: spyn 1.0
    Uninstalling spyn-1.0:
      Successfully uninstalled spyn-1.0
  Running setup.py install for spyn ... done
Successfully installed spyn-1.0
root@kv_pynq:/home/xilinx#
```

IIoT-SPYN is made into a  
installable package &  
enables remote upgradability

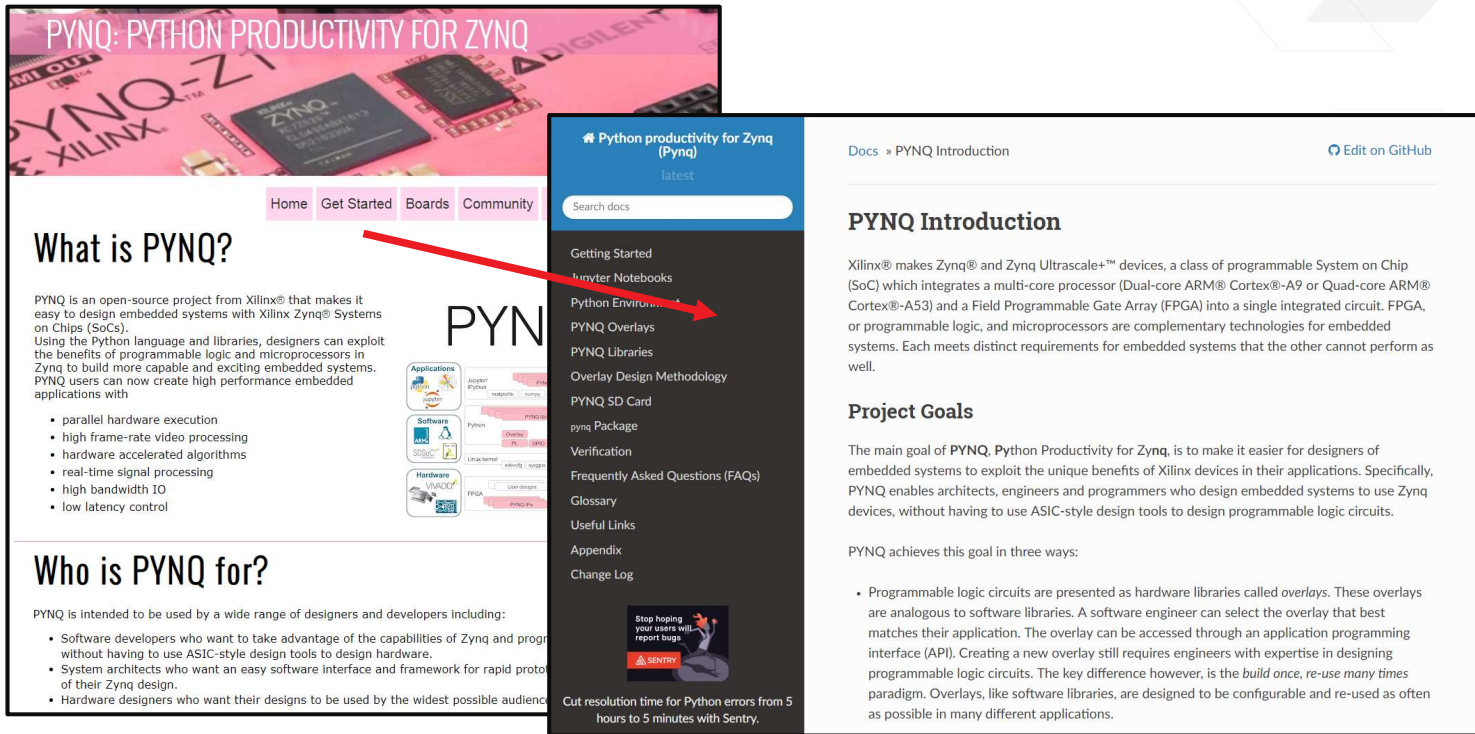


# Next Steps

## Getting started with PYNQ



# Find out more at [www.pynq.io](http://www.pynq.io)



**PYNQ: PYTHON PRODUCTIVITY FOR ZYNQ**

Home Get Started Boards Community

## What is PYNQ?

PYNQ is an open-source project 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 users can now create high performance embedded applications with

- parallel hardware execution
- high frame-rate video processing
- hardware accelerated algorithms
- real-time signal processing
- high bandwidth IO
- low latency control

## Who is PYNQ for?

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

**Python productivity for Zynq (Pynq)**  
latest

Search docs

- Getting Started
- Jupyter Notebooks
- Python Environment
- PYNQ Overlays
- PYNQ Libraries
- Overlay Design Methodology
- PYNQ SD Card
- PYNQ Package
- Verification
- Frequently Asked Questions (FAQs)
- Glossary
- Useful Links
- Appendix
- Change Log

Stop hoping your users will report bugs  
SENTRY

Cut resolution time for Python errors from 5 hours to 5 minutes with Sentry.

Docs » PYNQ Introduction [Edit on GitHub](#)

## PYNQ Introduction

Xilinx® makes Zynq® and Zynq Ultrascale+™ devices, a class of programmable System on Chip (SoC) which integrates a multi-core processor (Dual-core ARM® Cortex®-A9 or Quad-core ARM® Cortex®-A53) and a Field Programmable Gate Array (FPGA) into a single integrated circuit. FPGA, or programmable logic, and microprocessors are complementary technologies for embedded systems. Each meets distinct requirements for embedded systems that the other cannot perform as well.

## Project Goals

The main goal of PYNQ, Python Productivity for Zynq, is to make it easier for designers of embedded systems to exploit the unique benefits of Xilinx devices in their applications. Specifically, PYNQ enables architects, engineers and programmers who design embedded systems to use Zynq devices, without having to use ASIC-style design tools to design programmable logic circuits.

PYNQ achieves this goal in three ways:

- Programmable logic circuits are presented as hardware libraries called *overlays*. These overlays are analogous to software libraries. A software engineer can select the overlay that best matches their application. The overlay can be accessed through an application programming interface (API). Creating a new overlay still requires engineers with expertise in designing programmable logic circuits. The key difference however, is the *build once, re-use many times* paradigm. Overlays, like software libraries, are designed to be configurable and re-used as often as possible in many different applications.

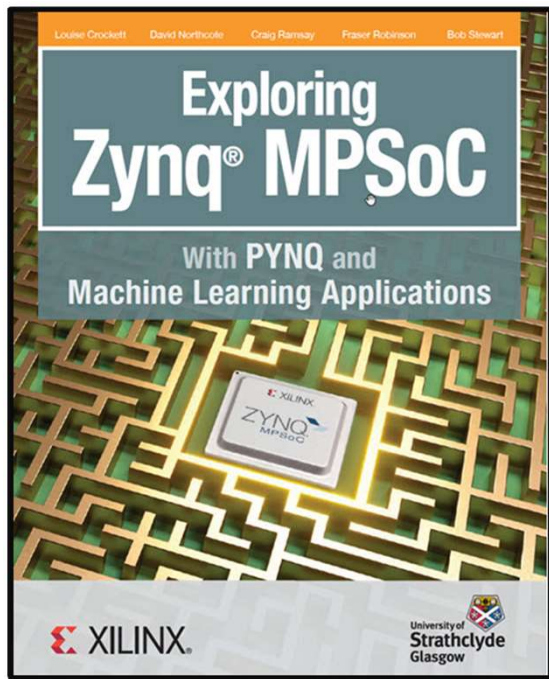
# PYNQ Community

[www.pynq.io/community.html](http://www.pynq.io/community.html)

The collage displays various PYNQ projects:

- PYNQ RFSoc**: University Strathclyde, QPSK demo on ZCU111. Includes a block diagram of the QPSK system.
- PYNQ-PRIO**: BYU, Partial reconfiguration Input/Output. Shows a code editor with Verilog code for partial reconfiguration.
- PYNQ Hello World**: Hardware accelerated image resizer example. Features a photo of the Eiffel Tower before and after resizing.
- RISC-V on PYNQ**: UCSO. Shows the RISC-V logo and a screenshot of the PYNQ interface.
- Extended Kalman filter**: University Sydney. Includes a complex block diagram of the filter implementation.
- spoonNN**: ETH Zurich, FPGA-based neural network inference project. Shows a neural network diagram and a table of inference results.
- iSmart DNN**: FPGA-based neural network inference for DAC 2018 contest. Includes a screenshot of the competition results page.
- TGIIF**: FPGA-based neural network inference for DAC 2018 contest. Shows a screenshot of the competition results page.
- cv2PYNQ**: FAU, accelerated OpenCV image filtering library. Includes a screenshot of a car in a video frame with filtering applied.
- Video processing**: KU Leuven, Hardware accelerated video processing. Shows a sequence of frames from a video being processed.
- ZipML-PYNQ**: ETH Zurich, Hardware accelerated compression. Includes the ZipML logo and a diagram of the compression pipeline.
- PYNQ bot**: IT Tofflight, Control of robotic car from PYNQ. Shows a small robotic car on a track.

# Exploring Zynq MPSoC with PYNQ & ML Applications



**PDF version available free of charge**

<https://www.zynq-mpsoc-book.com/>

# SPYN GitHub

## IIoT-SPYN

**IIoT-EDDP:** Industrial IoT Electric Drive Demonstration Platform is an open-source project that provides all necessary software and hardware components for development and evaluation of motor control applications.

**PYNQ:** Python on Zynq is an open-source project from Xilinx that makes it easy to design embedded systems with Zynq All Programmable Systems on Chips (APSoCs). 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.

**IIoT-SPYN:** Industrial IoT SPYN is an open source project that leverages IIoT-EDDP and PYNQ. Using IIoT-SPYN users can control, monitor, capture data, visualize and analyze Industrial grade motors.

IIoT-SPYN is intended to work with the EDDP kit. Here is the link to purchase the kit: [EDDP Kit](#)

## Quick Start for Arty-Z7-10

Step 1: Download the [Arty-Z7-10 PYNQ image](#)

Step 2: Write the image file to a SD card

Step 3: Use the following command in a terminal to install IIoT-SPYN

```
$ sudo pip3 install --upgrade git+https://github.com/Xilinx/IIoT-SPYN.git
$ sudo reboot now
```

After the setup, new Jupyter notebooks will be added under the spyn folder, ready to try out, no additional steps are needed.

## Quick Start for Pynq-Z1 / Arty-Z7-20

Step 1: Download the [PYNQ image](#)

Step 2: Write the image file to a SD card

Step 3: Use the following command in a terminal to install IIoT-SPYN

```
$ sudo pip3 install --upgrade git+https://github.com/Xilinx/IIoT-SPYN.git
$ sudo reboot now
```

After the setup, new Jupyter notebooks will be added under the spyn folder, ready to try out, no additional steps are needed.

<https://github.com/Xilinx/IIoT-SPYN>

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

