MYIR PYNQ

User Manual

Version V1.0

2019/08 /16

Contents

Introduction to PYNQ

PYNQ is called" Python Productivity for Zynq", that is, on the basis of the original Zynq architecture, it adds support for python. PYNQ hopes to effectively reduce the development threshold of the Zynq embedded system with the help of the python language itself, easy to use and easy to learn, extensive and comprehensive extension libraries, and high community active contribution. PYNQ completely encapsulates the bottom-level interaction logic between the ARM processor and the FPGA device. The top-level package uses python, you only need to import the corresponding module name to import the corresponding hardware module to interact with the bottom-up data or provide hardware acceleration for the system. . For PYNQ developers, a Linux system is running on ARM, FPGA is abstracted into a number of accelerated IP, developers can complete the dynamic loading of bitstream through a simple python script, and transfer the data stream to accelerated IP integration through DMA Output.

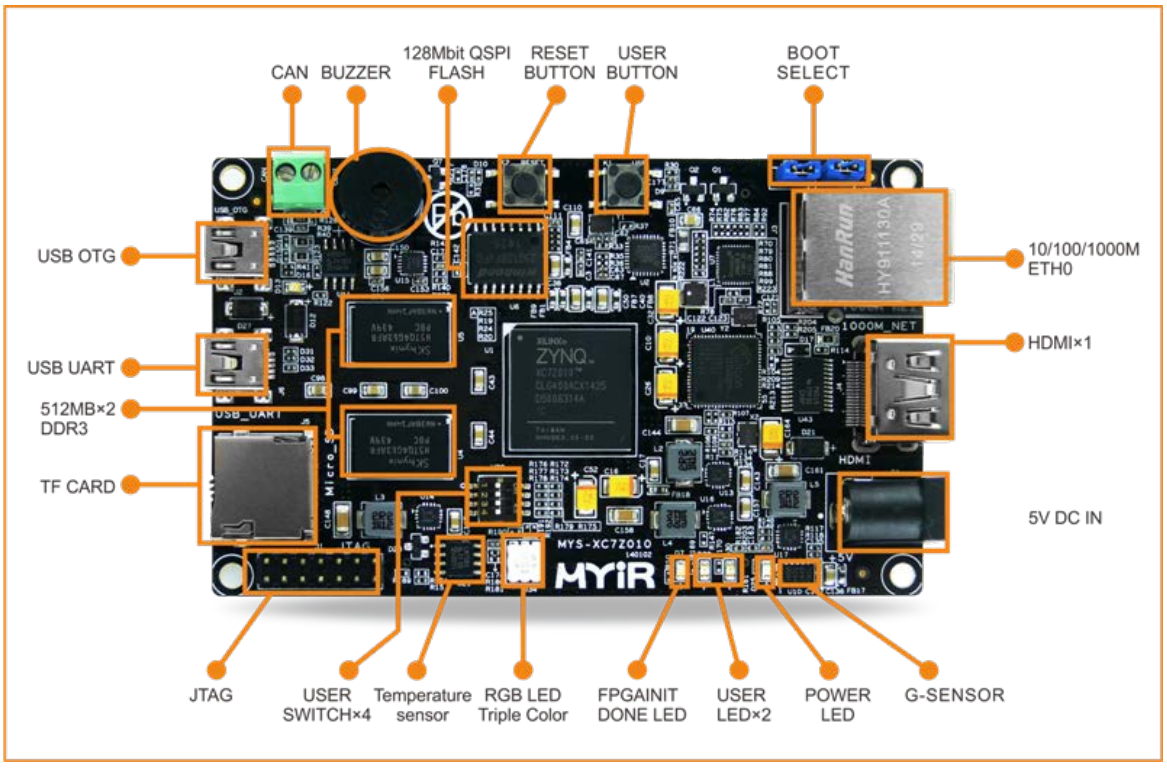
MYIR-PYNQ is a project based on Xilinx's PYNQ project of MYIR Technology transplanted to Zturn board. Software running on ARM A9 CPU includes:

- Web server for Jupyter Notebooks design environment
- IPython kernel and packages
- Ubuntu 18.04
- Basic hardware library and API of FPGA

Official resources of PYNQ:

Official website:http://www.pynq.io

Project address: https://github.com/Xilinx/PYNQ



1.Hardware and software preparation

1.1 Hardware introduction

Z-Turn Board is an embedded development board with Xilinx Zynq-7z010/7z020 as the main processor introduced by MYIR Technology. Z-Turn Board adopts Xilinx's latest Zynq-7000 All Programmable SoC platform based on 28nm process flow, tightly integrating ARM processor and FPGA architecture. This product has dual-core ARM Cortex-A9 MPCore's high performance and low power consumption characteristics, and can better meet various industrial needs in the design.

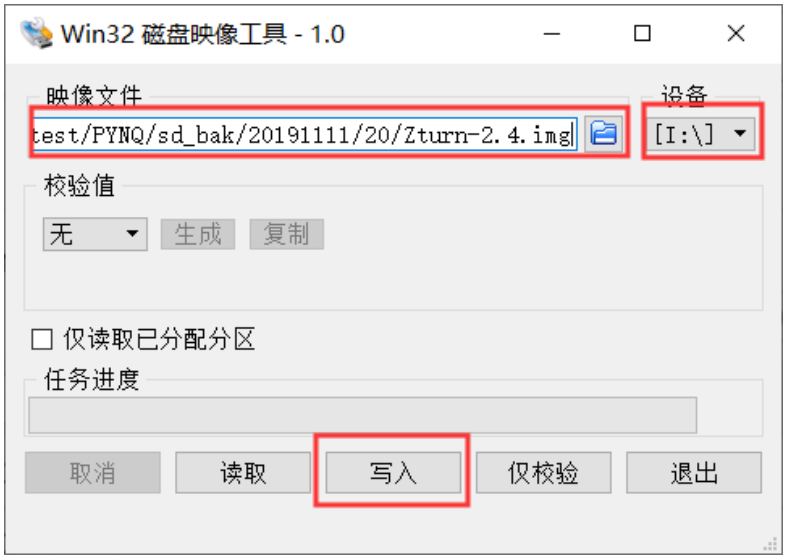
1.2 Software deployment

The CD data provided by Zturn Board does not support PYNQ. You need to burn the PYNQ image to the SD card by yourself. The burning method is as follows:

a. Prepare an SD card no less than 8GB, unzip Zturn-2.4.zip to get the Zturn-2.4.img image file

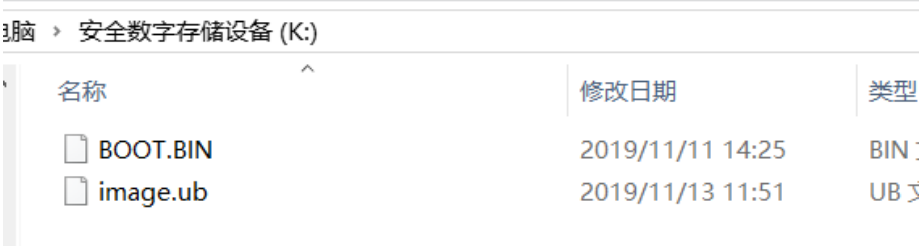
b. Insert the blank SD card into the computer and burn the image file:

1. Windows System:Install the “Win32DiskImager” tool which provided in the CD_ROM, open it with administrator rights,select the inserted SD card,then select the PYNQ image to be burned, and click “Write”



2. Linux or MacOS：Use the “dd” command,you can refer to the official tutorial ImageWriter.

c. After the writing is completed, the SD card will be divided into two partitions. The first partition is the Fat partition and the boot file (Boot.bin Image.ub) is placed. The second partition is the EXT4 partition and the PYNQ root file system is placed.

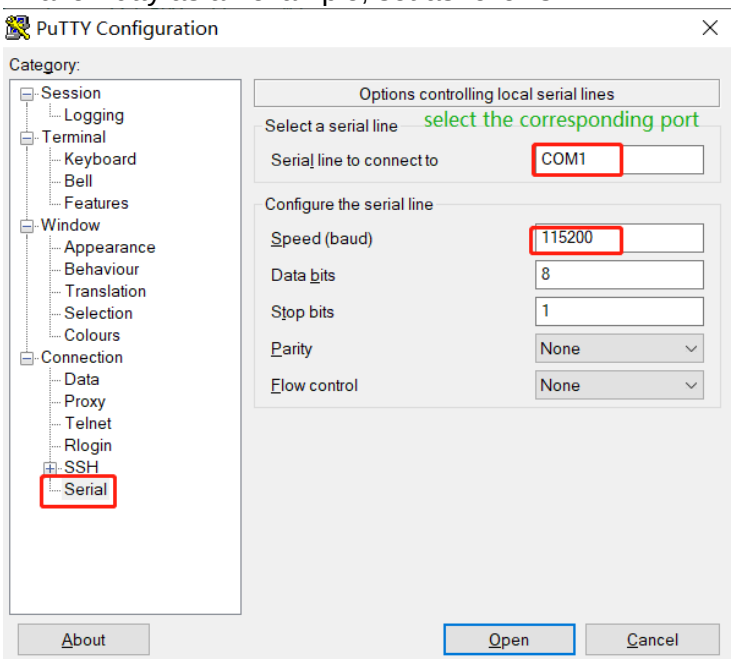


1.3 Start PYNQ

a. Insert the SD card with the PYNQ image into the development board, and set it to SD card startup mode according to the “LINUX development manual”in the CD-ROM to start the

development board.

b. Use the MicroUSB cable to connect the USB UART (J6) of the development board to the computer, and use a terminal tool (such as Putty) to open the COM port corresponding to the development board. Take Putty as an example, set as follows:



c. Connect the Ethernet port of the development board to the router or directly to the PC,and a static IP of 192.168.2.99 will be automatically added after the startup is completed

d. Detailed description of the development board Ethernet connection:

- You can connect the Ethernet port of the development board to the following devices:

  1. Connect to a router on the same network as the computer
  2. Connect directly to the computer's Ethernet port

  In order to facilitate the later update or installation of software packages on the development board, it is recommended to connect the development board to a network that can normally access the Internet.

- Connect to router

  If the development board is connected to a router with DHCP function, the board will automatically obtain an IP address:

  1. Connect the development board to the router,
  2. Visit via browser:http://pynq:9090

- Directly connected to the computer

  1. Add an IP of the 192.168.2.x network segment on the computer (the method of setting a static IP in multiple systems can refer to the official instructions of PYNQ: assign-your-computer-a-static-ip)
  2. Connect the development board to the computer
  3. Access http://192.168.2.99:9090

e. Install a browser that supports Jupyter on your computer

Tips： The latest stable version of the browser can support Jupyter Notebook:

Chrome
Safari
Firefox

1.4 Install a package or Python library

Please ensure that the development board can access the Internet normally

a. Install packages via apt

The image of PYNQ is based on Ubuntu 18.04, you can install the software package through the apt tool:

```
xilinx@pynq:~$ sudo apt-get install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
vim is already the newest version (2:8.0.1453-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

b. Install Python library via pip

Python3 has been installed in the PYNQ image, and third-party libraries can be installed through the PIP tool:

```
xilinx@pynq:~$ pip3 install bs4
Collecting bs4
  Downloading https://files.pythonhosted.org/packages/10/ed/7e8b97591f6f456174139ec089c769f89a94a1a4025fe967691de971f314/bs4-0.0.1.tar.gz
Collecting beautifulsoup4 (from bs4)
  Downloading https://files.pythonhosted.org/packages/3b/c8/a55eb6ea11cd7e5ac4bacdf92bac4693b90d3ba79268be16527555e186f0/beautifulsoup4-4.8.1-py3-none-any.whl (101kB)
    100% |████████████████████████████████| 102kB 162kB/s
Collecting soupsieve>=1.2 (from beautifulsoup4->bs4)
  Downloading https://files.pythonhosted.org/packages/81/94/03c0f04471fc245d08d0a99f7946ac228ca98da4fa75796c507f61e688c2/soupsieve-1.9.5-py2.py3-none-any.whl
Building wheels for collected packages: bs4
  Running setup.py bdist_wheel for bs4 ... done
  Stored in directory: /home/xilinx/.cache/pip/wheels/a0/b0/b2/4f80b9456b87abedbc0bf2d52235414c3467d8889be38dd472
Successfully built bs4
Installing collected packages: soupsieve, beautifulsoup4, bs4
Successfully installed beautifulsoup4-4.8.1 bs4-0.0.1 soupsieve-1.9.5
```

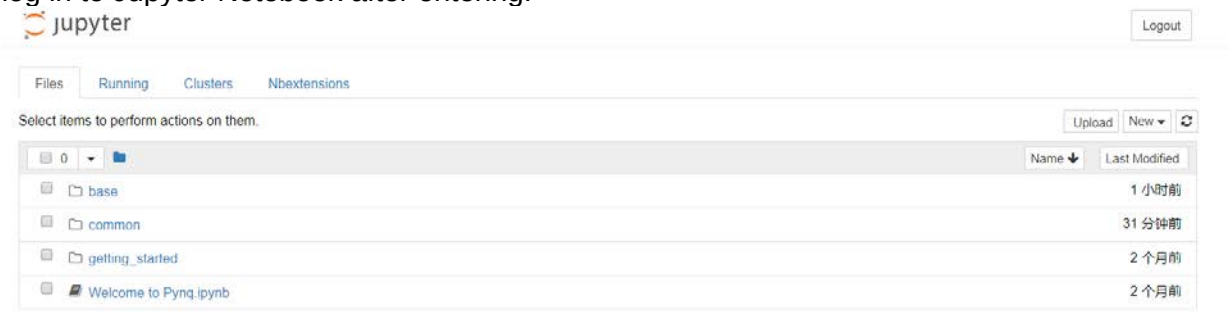
2. Use of Jupyter Notebook

Jupyter Notebook (formerly IPython Notebook) is a web-based interactive computing environment for creating Jupyter Notebook documents. The term Notebook can easily refer to many different entities, mainly Jupyter web applications, Jupyter Python web servers or Jupyter document formats (depending on the context). A Jupyter Notebook document is a JSON document that follows a versioned pattern and contains an ordered list of input/output cells. These cells can contain code, text, math, graphs, and rich media, and usually end with ".ipynb" extension.

After installing the zturn PYNQ image file provided by MYIR, you can use the Python language to call the relevant hardware library in the Jupyter Notebook and the Overlay to easily interact with the hardware platform.

1.Connect to Jupyter

If the development board is connected to the router via a network cable, the development board will be automatically assigned an IP address. Enter: http://pynq:9090/ in the address bar of the browser to access the login page of PYNQ Jupyter. Assign IP, you can log in through the static IP assigned by default: http://192.168.2.99:9090/, the password is "Xilinx", you can log in to Jupyter Notebook after entering.



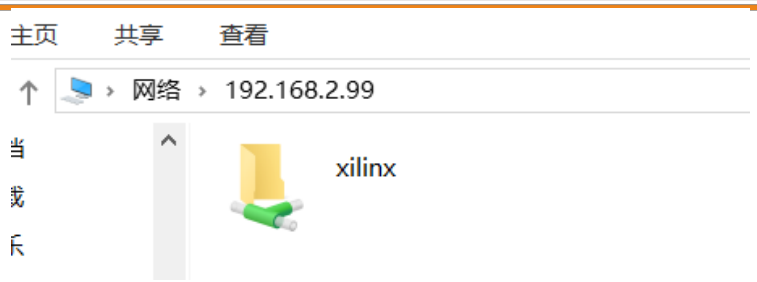
The instructions for using Jupyter can be found in the getting_started folder of the Jupyter homepage. It is recommended that users be familiar with the basic operations of Jupyter.

We have integrated and classified many sample documents in the released images:

- common: Examples of non-targeted overlay projects
- base/board: Examples of Myir PYNQ base overlay
- base/video: Examples of HDMI display and USB camera

2. Samba file sharing

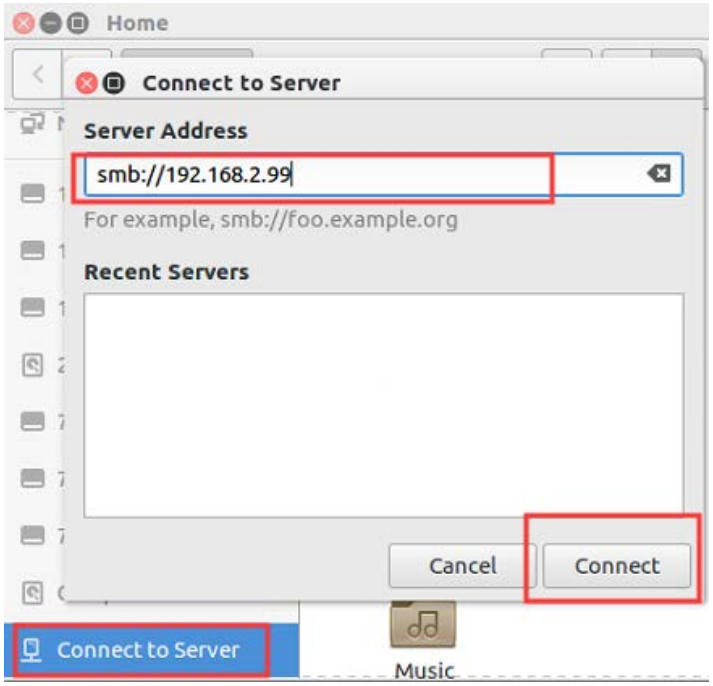
a. Windows System: Enter: \\192.168.2.99 in the file resource manager, the user name and password are both ” xilinx”,



After successful login, you can access the files on the development board.

.cache	2019/11/12 14:08	文件夹	
.gnupg	2019/11/12 14:08	文件夹	
jupyter_notebooks	2019/11/13 9:38	文件夹	
pynq	2019/11/14 11:17	文件夹	
.bash_logout	2019/11/12 14:07	Bash Logout 源...	1 KB
.bashrc	2019/11/12 14:07	Bash RC 源文件	4 KB
.profile	2019/11/12 14:07	Profile 源文件	1 KB
.sudo_as_admin_successful	2019/11/13 9:57	SUDO_AS_ADMI...	0 KB
REVISION	2019/11/12 20:05	文件	1 KB

b. Linux: Click "connect to server", enter ” smb://192.168.2.99”, and then click "connect", the username and password are both ” xilinx”.



3. Samples

In the PYNQ project, the official provides a lot of demos, For example, the more basic LED and button control, use of USB camera,use of WIFI, HDMI video output ,etc. There are also many open source routines in the PYNQ forum:

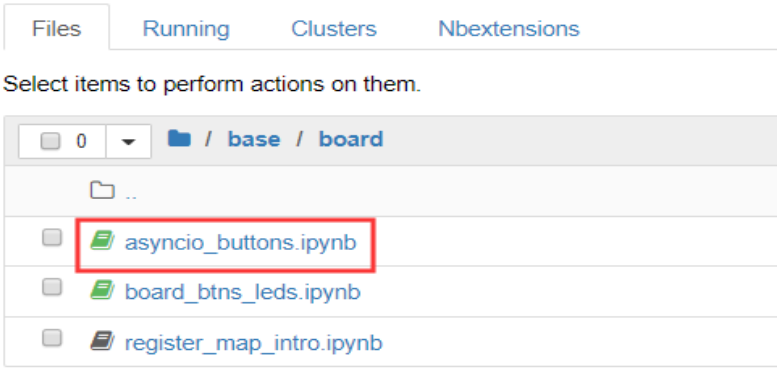
http://www.pynq.io/community.html.

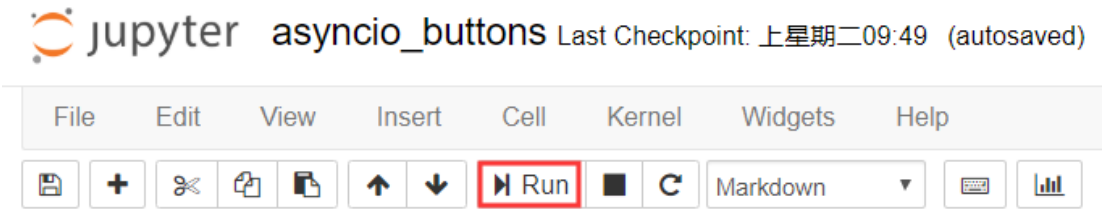
This chapter will select two examples to explain.

All samples in Jupyter Notebook have detailed notes，The relevant introduction of Library in PYNQ can be found in the official documents of PYNQ:
https://pynq.readthedocs.io/en/latest/pynq_libraries.html

3.1 LED and button test

Use a browser to visit http://192.168.2.99:9090 to log in to Jupyter Notebook, After successful login, enter base/board/asuncio_buttons.ipynq. Click the run icon in the toolbar or select Cell->Run to run the code.





code show as below:

Import the supporting files of the development board

```
from pynq import PL
from pynq.overlays.base import BaseOverlay

base = BaseOverlay("base.bit")
```

Define the task of LED blinking

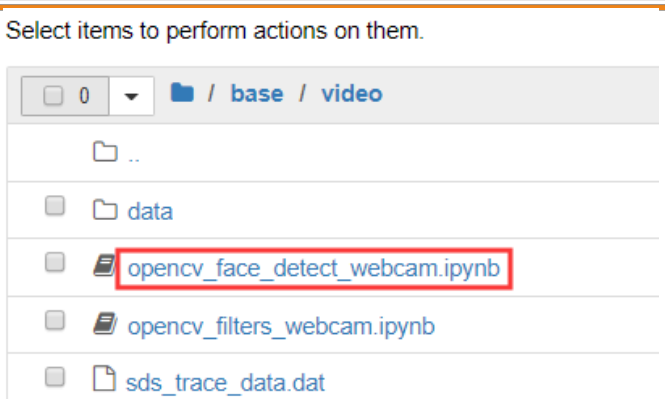
```
import asyncio

@asyncio.coroutine
def flash_led(num):
    while True:
        yield from base.buttons[num].wait_for_value_asyn
c(1)
        while base.buttons[num].read():
```

```
            base.leds[num].toggle()
            yield from asyncio.sleep(0.1)
        base.leds[num].off()
# Create task
tasks = [asyncio.ensure_future(flash_led(i)) for i in ra
nge(4)]
# Detect CPU usage
import psutil

@asyncio.coroutine
def print_cpu_usage():
    # Calculate the CPU utilisation by the amount of idl
e time
    # each CPU has had in three second intervals
    last_idle = [c.idle for c in psutil.cpu_times(percpu
=True)]
    while True:
        yield from asyncio.sleep(3)
        next_idle = [c.idle for c in psutil.cpu_times(pe
rcpu=True)]
        usage = [(1-(c2-
c1)/3) * 100 for c1,c2 in zip(last_idle, next_idle)]
        print("CPU Usage: {0:3.2f}%, {1:3.2f}%".format(*
usage))
        last_idle = next_idle

# Start task
asyncio.get_event_loop().run_forever()
# Clear task
[t.cancel() for t in tasks]
```

Note: The 4 buttons in this example correspond to the U20 dial switch on the development board

3.2 USB camera face detection

Connect the USB camera to the development board through the Micro USB OTG adapter cable, connect the HDMI interface of the development board to the HDMI interface of the display, and open base/video/opencv_face_detect_webcam.ipynb. This sample is based on the OpenCV face detection demo. Grab the data from the USB camera, process it by OpenCV related libraries, and finally output it to the screen through HDMI.



The code is divided into code blocks, and each piece of code has related comments. Click the run icon on the toolbar or select Cell->Run to run the following code blocks in sequence. The code block in operation will be marked as [*], as shown in the figure. At this time, you should wait for the end of the operation and you cannot proceed to the next step. After the operation is over, [*] will become a number, indicating that this is the first few pieces of code to be executed, in order of record.

code show as below:

```
from pynq.overlays.base import BaseOverlay
from pynq.lib.video import *
base = BaseOverlay("base.bit")
```

Initialize HDMI output and USB camera

```
# monitor configuration: 640*480 @ 60Hz
Mode = VideoMode(640,480,24)
hdmi_out = base.video.hdmi_out
hdmi_out.configure(Mode,PIXEL_BGR)
hdmi_out.start()
# monitor (output) frame buffer size
frame_out_w = 1920
frame_out_h = 1080
# camera (input) configuration
frame_in_w = 640
frame_in_h = 480
```

Initialize camera by OpenCV

```
# initialize camera from OpenCV
import cv2

videoIn = cv2.VideoCapture(0)
videoIn.set(cv2.CAP_PROP_FRAME_WIDTH, frame_in_w)
videoIn.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_in_h)
```

```
print("Capture device is open: " + str(videoIn.isOpened(
)))
```

Output one frame of camera data through HDMI

```
# Capture webcam image
import numpy as np

ret, frame_vga = videoIn.read()

# Display webcam image via HDMI Out
if (ret):
    outframe = hdmi_out.newframe()
    outframe[0:480,0:640,:] = frame_vga[0:480,0:640,:]
    hdmi_out.writeframe(outframe)
else:
    raise RuntimeError("Failed to read from camera.")
```

Display image through matplotlib in Jupyter

```
# Output webcam image as JPEG
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
plt.imshow(frame_vga[:,:,[2,1,0]])
```


```
plt.show()
```

Use the already trained model in OpenCV to recognize faces

```
import cv2

np_frame = frame_vga
```

```
face_cascade = cv2.CascadeClassifier(
    '/home/xilinx/jupyter_notebooks/base/video/data/'
    'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(
    '/home/xilinx/jupyter_notebooks/base/video/data/'
    'haarcascade_eye.xml')

gray = cv2.cvtColor(np_frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:
    cv2.rectangle(np_frame,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = np_frame[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0
,255,0),2)
```

Output the final result via HDMI

```
# Output OpenCV results via HDMI
outframe[0:480,0:640,:] = frame_vga[0:480,0:640,:]
hdmi_out.writeframe(outframe)
```

Display the final result in Jupyter

```
# Output OpenCV results via matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
plt.imshow(np_frame[:,:,[2,1,0]])
plt.show()
```

Release camera and HDMI resources

```
videoIn.release()
hdmi_out.stop()
del hdmi_out
```

Appendix 1 Warranty & Technical Support Services

MYIR Tech Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR's products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (**support@myirtech.com**), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:

1. Technical support service

a) MYIR offers technical support for the hardware and software materials which have provided to customers;
b) To help customers compile and run the source code we offer;
c) To help customers solve problems occurred during operations if users follow the user manual documents;
d) To judge whether the failure exists;
e) To provide free software upgrading service.

However, the following situations are not included in the scope of our free technical support service:

a) Hardware or software problems occurred during customers' own development;
b) Problems occurred when customers compile or run the OS which is tailored by themselves;
c) Problems occurred during customers' own applications development;
d) Problems occurred during the modification of MYIR's software source code.

2. After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

a) The warranty period is expired;
b) The customer cannot provide proof-of-purchase or the product has no serial number;
c) The customer has not followed the instruction of the manual which has caused the damage the product;
d) Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads the defects of appearance/function;
e) Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which have caused the damage of the products or defects of appearance;
f) Due to unauthorized weld or dismantle parts or repair the products which has caused the damage of the products or defects of appearance;
g) Due to unauthorized installation of the software, system or incorrect configuration or

computer virus which has caused the damage of products.

Warm tips:

1) MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.
2) Please do not use finger nails or hard sharp object to touch the surface of the LCD.
3) MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.
4) Do not clean the surface of the screen with chemicals.
5) Please read through the product user manual before you using MYIR's products.
6) For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR's support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

3. Maintenance period and charges

a) MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.
b) For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.
4. Shipping cost
During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.
5. Products Life Cycle
MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR's products, like serial port, USB, Ethernet, LCD, etc.
2. MYIR provides the services of OS porting, BSP drivers' development, API software development, etc.
3. MYIR provides other products supporting services like power adapter, LCD panel, etc.
4. ODM/OEM services.

MYIR Tech Limited

Room 04, 6th Floor, Building No.2, Fada Road, Yunli Smart Park, Bantian, Longgang District, Shenzhen, Guangdong, China 518129

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com