

Hướng dẫn sử dụng bảng phát triển ALINX đen vàng

Phát triển ứng dụng Linux trên Ultrascale+

2020/05/18 V1.01



Tuyên bố bản quyền

Bản quyền © 2012-2019 Công ty TNHH Công nghệ điện tử Tín Nghĩa (Thượng Hải) -

Địa chỉ website:

[Http://www.alinx.com.cn](http://www.alinx.com.cn)

Diễn đàn kỹ thuật:

<http://www.heijin.org>

Cửa hàng chính hãng trên Tmall:

<https://alinx.tmall.com>

Cửa hàng chính hãng trên JD:

<http://alinx.jd.com>

Email:

avic@alinx.com.cn

Điện thoại:

021-67676997

Fax:

021-37737073ALINX

WeChat Official Account:



Lịch sử sửa đổi tài liệu:

Phiên bản	Thời gian	Mô tả
1.01	2020/5/18	Phiên bản ban đầu

Chúng tôi cam kết rằng hướng dẫn này không phải là một tài liệu cố định, không thay đổi. Chúng tôi sẽ liên tục sửa đổi và tối ưu hóa hướng dẫn dựa trên phản hồi từ diễn đàn và kinh nghiệm thực tiễn trong phát triển.

Mục lục

Tuyên bố bản quyền	2
Chương một Thiết lập môi trường làm việc tối giản	6
1.1 Giới thiệu về việc thiết lập môi trường hệ thống	6
1.2 Cài đặt hệ thống và các công cụ cơ bản	6
1.3 Cài đặt môi trường biên dịch	6
1.4 Cấu hình QtCreator	7
Chương hai Hello World có thể gỡ lỗi từ xa	12
2.1 Giới thiệu GDB	12
2.2 Giới thiệu QtCreator	12
2.3 Mục tiêu thí nghiệm	12
2.4 Dự án Qt Creator	13
2.5 Lấy địa chỉ IP của bo mạch	13
2.6 Cấu hình GDB kết nối với bo mạch trong QtCreator	13
2.7 Chạy Hello World	16
2.8 Cài đặt tham số vận hành	17
Chương ba Phát hiện biên trong OpenCV	20
3.1 Giới thiệu về OpenCV	20
3.2 Giới thiệu về phát hiện biên	20
3.3 Mục tiêu thí nghiệm	21
3.4 Sơ đồ quy trình phần mềm	21
3.5 Chuẩn bị chạy	21
3.6 Chạy chương trình	22
3.7 Phân tích mã	22
Chương 4 Phát hiện khuôn mặt với OpenCV+Qt	25
4.1 Bộ phân loại cascade của OpenCV	25
4.2 Biểu đồ điểm	26
4.3 Chọn đặc trưng	27
4.4 Giới thiệu về Qt	27
4.5 Mục tiêu thí nghiệm	28
4.6 Sơ đồ quy trình phần mềm	28
4.7 Chuẩn bị vận hành	28
4.8 Chạy chương trình	28
4.9 Phân tích mã	29
Chương năm Hiển thị camera của GStreamer	30
5.1 Giới thiệu về GStreamer	30
5.2 Các khái niệm cơ bản về GStreamer	31
5.3 Mục tiêu thí nghiệm	32

5.4 Công cụ thường dùng của Gstreamer	32
5.5 gst-launch-1.0 Hiển thị camera	33
5.6 Chương trình chạy mã	34
5.7 Phân tích mã	34
Chương sáu Hiển thị camera bằng Qt+DRM+Gstreamer	35
6.1 Giới thiệu về DRM	35
6.2 Giới thiệu thí nghiệm	36
6.3 Mục tiêu thí nghiệm	36
6.4 Chạy chương trình	36
6.5 Phân tích mã	37
Chương bảy Hiển thị camera bằng Qt+GPU	38
7.1 Giới thiệu về GPU Mali	38
Giới thiệu GPU MPSoc 7.2	38
Giới thiệu V4L2 7.3	39
7.4 Mục tiêu thí nghiệm	40
7.5 Chạy chương trình	40
7.6 Phân tích mã	41
Chương tám Thao tác với thanh ghi Linux	42
8.1 ánh xạ bộ nhớ linux	42
8.2 Giới thiệu thí nghiệm	42
8.3 Chạy chương trình	42
8.4 Phân tích mã	43

Chương một: Thiết lập môi trường làm việc tối giản

1.1 Giới thiệu về việc thiết lập môi trường hệ thống

Hướng dẫn này sử dụng QtCreator làm công cụ phát triển trên ubuntu để biên dịch chéo ứng dụng. Vì vậy, trên ubuntu, chúng ta cần cài đặt công cụ biên dịch chéo cũng như các thư viện cần thiết, điều này có thể được thực hiện thông qua gói cài đặt sdk.sh. Nguồn gốc của gói cài đặt này có thể tham khảo trong hướng dẫn của chương petalinux, ở đây không đề cập đến.

1.2 Cài đặt hệ thống và các công cụ cơ bản

- (1) Trong hệ thống win10, để cài đặt phần mềm máy ảo VMware, bạn có thể tham khảo tài liệu cài đặt 《vmware 安装.pdf》
- (2) Sử dụng ubuntu-18.04.2-desktop-amd64.iso để tạo hệ thống linux, bạn có thể tham khảo tài liệu cài đặt 《ubuntu 虚拟机创建.pdf》
- (3) cài đặt các thư viện cần thiết trên ubuntu

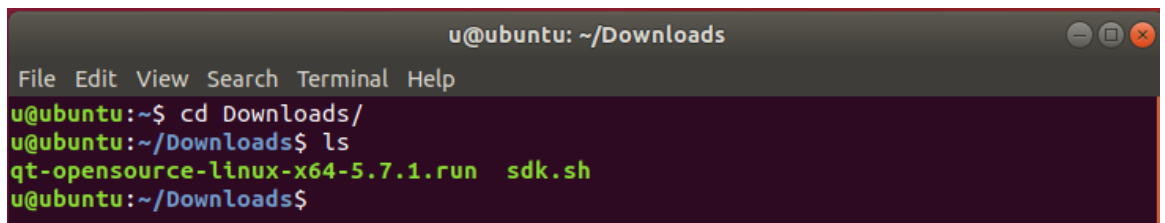
```
sudo apt-get install build-essential
```

Có thể tham khảo tài liệu cài đặt công cụ 《ubuntu thư viện cài đặt.pdf》

- (4) Hướng dẫn cài đặt QtCreator trên Ubuntu
có thể tham khảo trong tài liệu cài đặt 《linux 下 Qt 安装.pdf》

1.3 Cài đặt môi trường biên dịch

- (1) Sao chép sdk.sh vào hệ thống Ubuntu và mở terminal



```
u@ubuntu: ~/Downloads
File Edit View Search Terminal Help
u@ubuntu:~$ cd Downloads/
u@ubuntu:~/Downloads$ ls
qt-opensource-linux-x64-5.7.1.run  sdk.sh
u@ubuntu:~/Downloads$
```

- (2) Tạo thư mục tools trong thư mục gốc và thay đổi quyền truy cập

```
sudo mkdir /tools
sudo chmod 777 -R /tools
```

- (3) Chạy sdk.sh

```

u@ubuntu: ~/Downloads
File Edit View Search Terminal Help
u@ubuntu:~$ cd Downloads/
u@ubuntu:~/Downloads$ ls
qt-opensource-linux-x64-5.7.1.run  sdk.sh
u@ubuntu:~/Downloads$ ./sdk.sh
PetaLinux SDK installer version 2019.2
=====
Enter target directory for SDK (default: /opt/petalinux/2019.2):

```

- (4) Nhập thư mục cài đặt, ở đây thư mục cài đặt được thiết lập là "/tools/xilinx_sdk_tool", và nhấn Enter

```

u@ubuntu: ~/Downloads
File Edit View Search Terminal Help
u@ubuntu:~$ cd Downloads/
u@ubuntu:~/Downloads$ ls
qt-opensource-linux-x64-5.7.1.run  sdk.sh
u@ubuntu:~/Downloads$ ./sdk.sh
PetaLinux SDK installer version 2019.2
=====
Enter target directory for SDK (default: /opt/petalinux/2019.2): /tools/xilinx_s
dk_tool
You are about to install the SDK to "/tools/xilinx_sdk_tool". Proceed[Y/n]?

```

- (5) Nhập "Y" để bắt đầu cài đặt, cho đến khi kết thúc

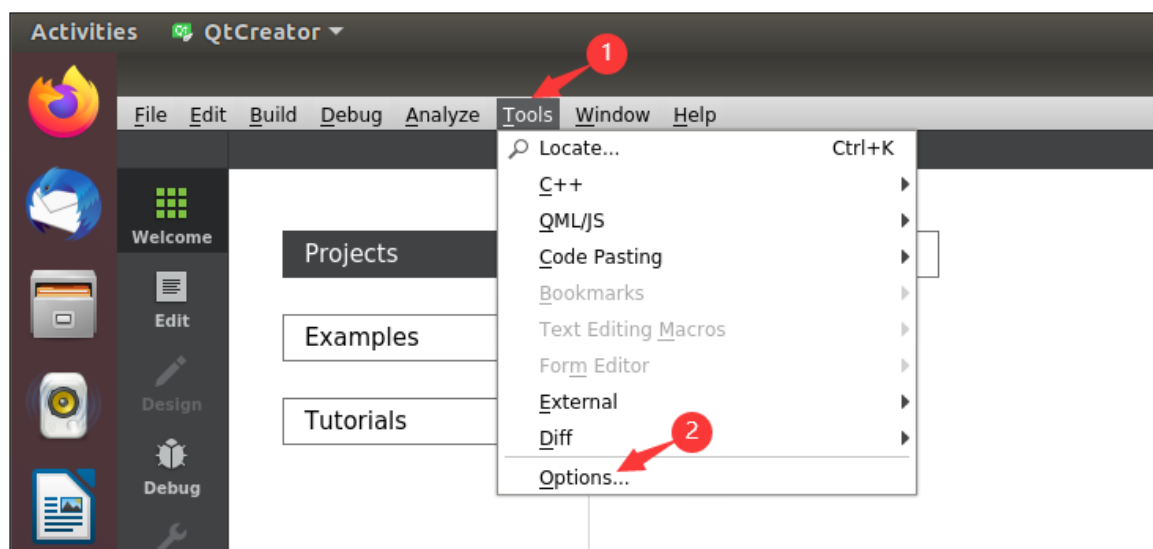
```

u@ubuntu: ~/Downloads
File Edit View Search Terminal Help
u@ubuntu:~$ cd Downloads/
u@ubuntu:~/Downloads$ ls
qt-opensource-linux-x64-5.7.1.run  sdk.sh
u@ubuntu:~/Downloads$ ./sdk.sh
PetaLinux SDK installer version 2019.2
=====
Enter target directory for SDK (default: /opt/petalinux/2019.2): /tools/xilinx_s
dk_tool
You are about to install the SDK to "/tools/xilinx_sdk_tool". Proceed[Y/n]? Y
Extracting SDK.....

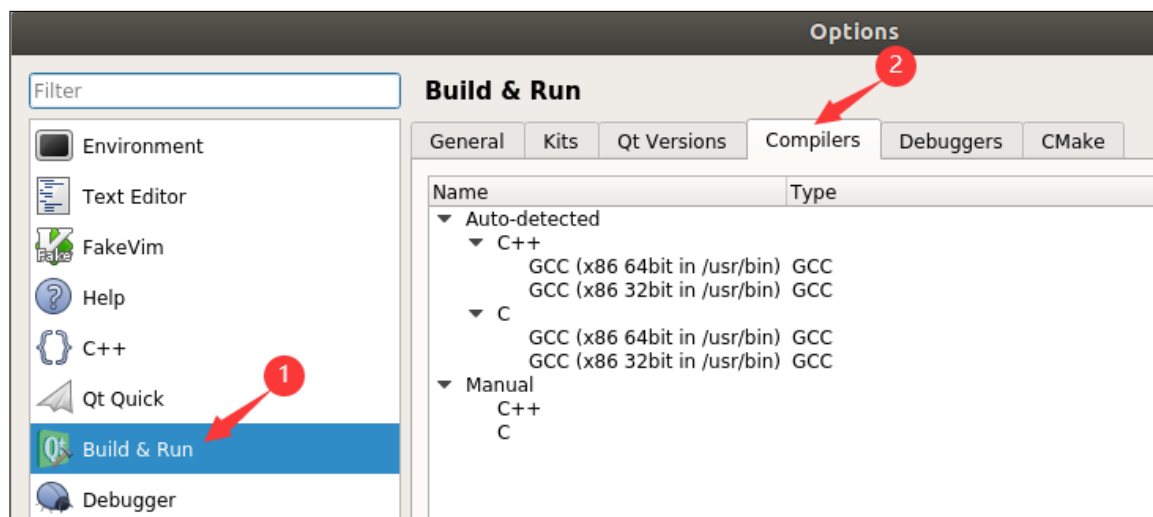
```

1.4 Cấu hình QtCreator

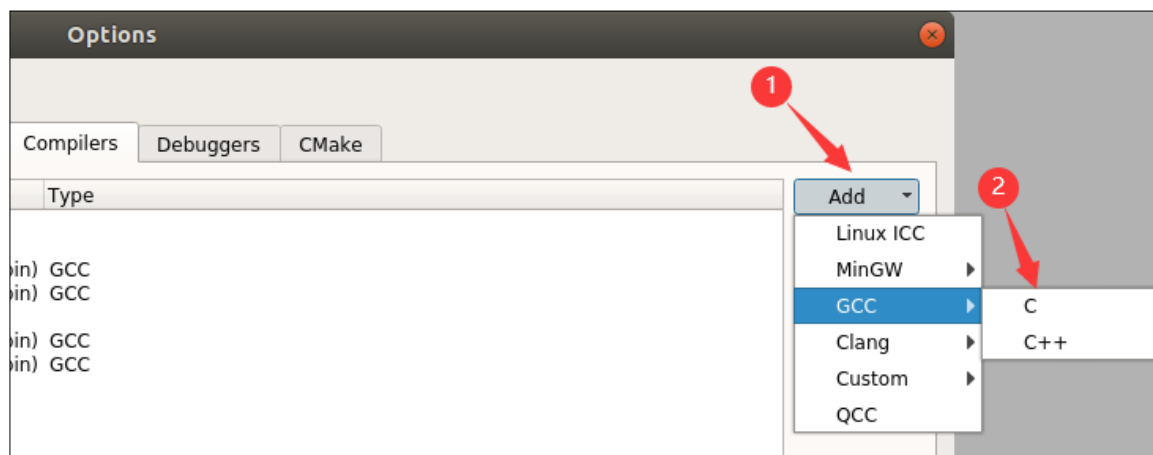
- (1) Mở mục cài đặt công cụ QtCreator



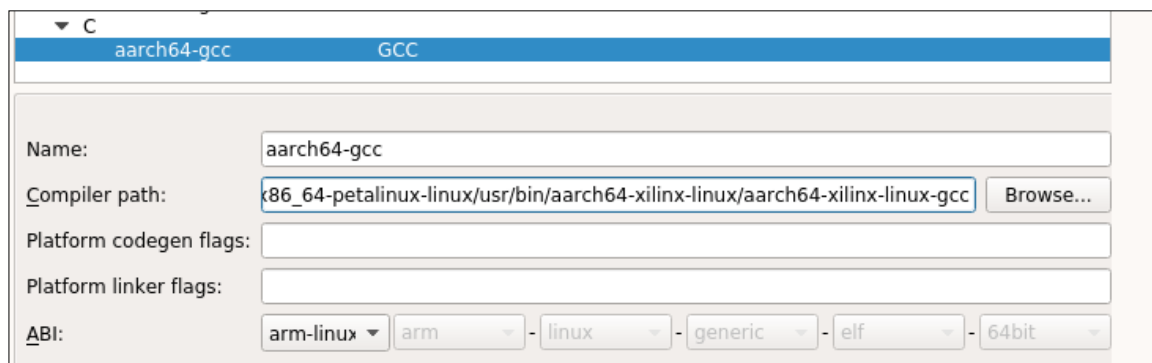
(2) Mở cài đặt công cụ biên dịch



(3) Thêm công cụ biên dịch C



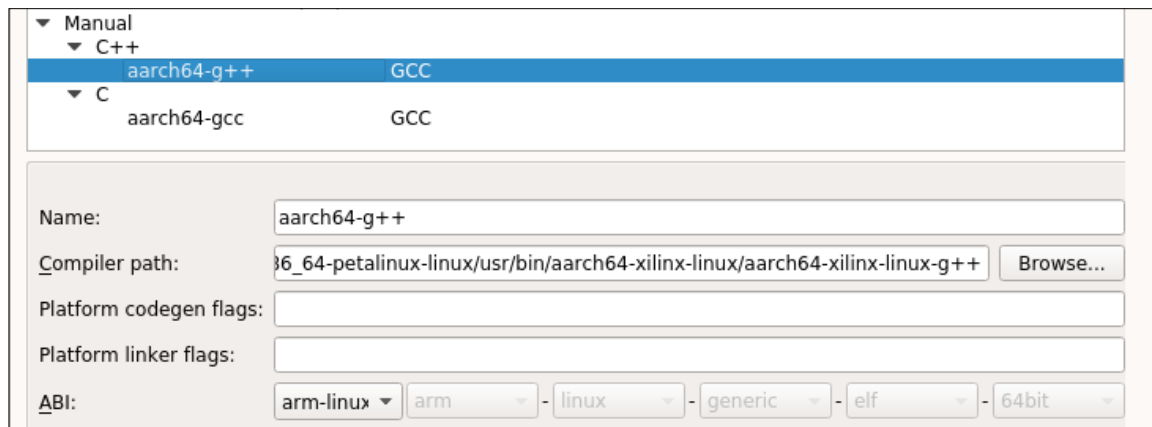
(4) Đặt tên và đường dẫn



Đường dẫn là

"/tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-gcc"

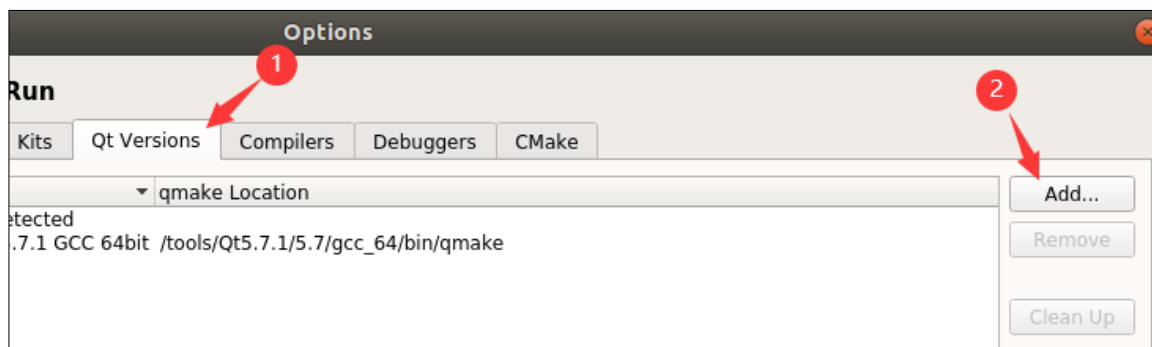
- (5) Cùng với các bước trên, thêm công cụ biên dịch C++



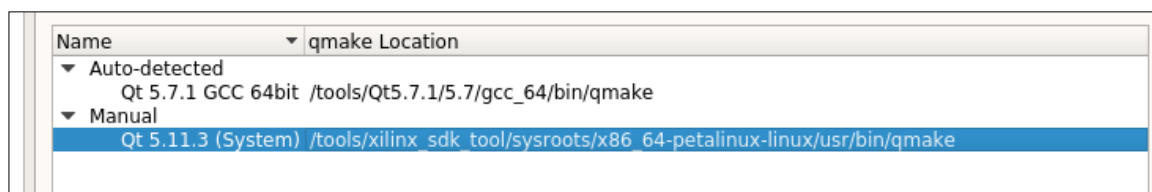
Đường dẫn là

"/tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-g++"

- (6) Nhấn nút "Apply" để hoàn tất cài đặt này
(7) Chuyển sang "Qt Versions", nhấn nút thêm



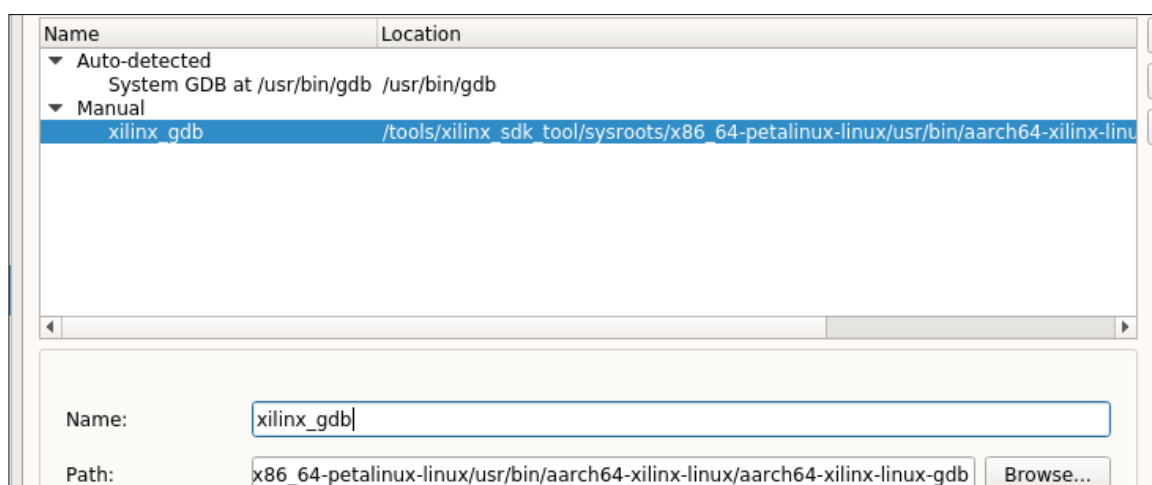
(8) Chọn qmake



Đường dẫn là `"/tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/qmake"`

(9) Nhấn nút "Apply" để hoàn tất cài đặt này

(10) Chuyển sang mục "Debuggers" và nhấn nút "Add"



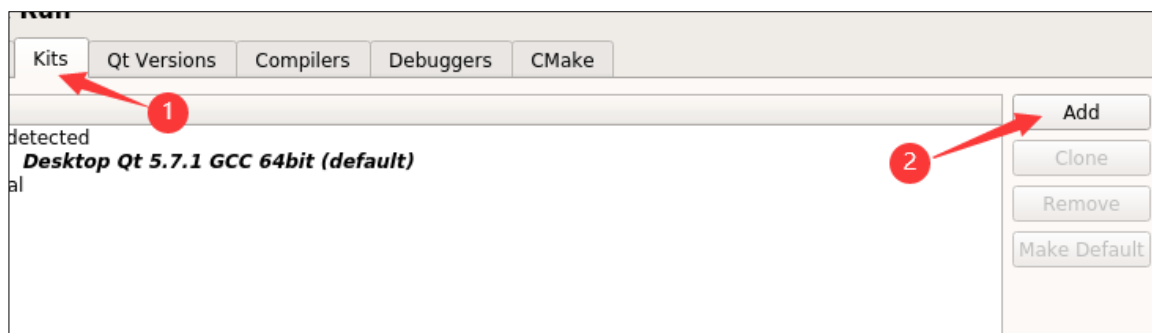
Thay đổi nội dung mục đã thêm như hình, trong đó đường dẫn

là

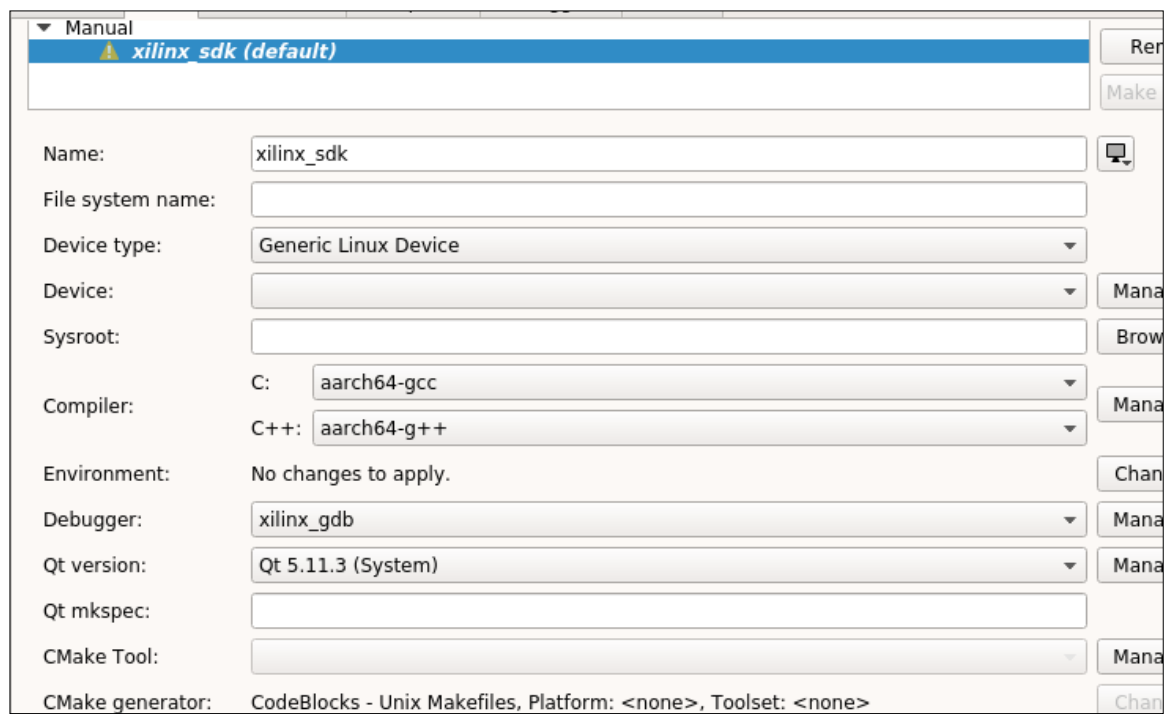
`"/tools/xilinx_sdk_tool/sysroots/x86_64-petalinux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-gdb"`

(11) Nhấn nút "Apply" để hoàn tất cài đặt này

(12) Chuyển sang mục "Kits" và nhấn nút "Add"



(13) Sửa đổi nội dung của các mục mới thêm như hình



The screenshot shows the configuration window for the 'xilinx_sdk (default)' project. The window has a sidebar on the left with a 'Manual' dropdown and a list of configuration items. The main area contains the following fields and options:

- Name:** xilinx_sdk
- File system name:** (empty)
- Device type:** Generic Linux Device
- Device:** (empty)
- Sysroot:** (empty)
- Compiler:** C: aarch64-gcc, C++: aarch64-g++
- Environment:** No changes to apply.
- Debugger:** xilinx_gdb
- Qt version:** Qt 5.11.3 (System)
- Qt mkspec:** (empty)
- CMake Tool:** (empty)
- CMake generator:** CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none>

On the right side of the window, there are several buttons: 'Ren', 'Make', 'Mana', 'Brow', 'Mana', 'Chan', 'Mana', 'Mana', and 'Chan'.

(14) Nhấn nút "OK" để hoàn tất cài đặt

Chương hai: Hello World có thể gỡ lỗi từ xa

2.1 Giới thiệu GDB

GDB, trình gỡ lỗi của Dự án GNU, cho phép bạn xem những gì đang xảy ra 'nội bộ' trong chương trình khi nó đang thực thi, hoặc những gì một chương trình đang làm khi bị sập. Các chức năng chính như sau:

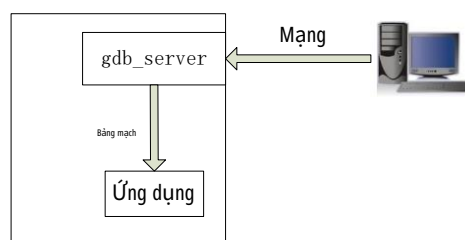
① Khởi động chương trình, chỉ định bất kỳ nội dung nào có thể ảnh hưởng đến hành vi của nó, chẳng hạn như biến môi trường, tham số chạy, v.

v. ② Dừng chương trình khi đạt được các điều kiện nhất định

③ Kiểm tra xem đã xảy ra điều gì khi chương trình dừng lại, chẳng hạn như trạng thái thanh ghi, giá trị biến, v.v. ④ Thay đổi thủ công

các giá trị khác nhau trong quá trình chạy của chương trình, điều này có thể giúp xác minh chức năng của chương trình một cách thuận tiện hơn.

Công cụ gỡ lỗi GNU trên Linux là gdb, gdbserver. Trong đó, gdb và gdbserver có thể thực hiện gỡ lỗi từ xa cho các ứng dụng trên bảng mục tiêu chạy dưới Linux. gdbserver là một ứng dụng rất nhỏ, chạy trên bảng mục tiêu, có thể giám sát quá trình chạy của tiến trình đang được gỡ lỗi và có thể giao tiếp qua mạng với gdb trên máy tính chủ. Các nhà phát triển có thể nhập lệnh vào gdb trên máy tính chủ để điều khiển quá trình trên bảng mục tiêu, xem nội dung bộ nhớ và thanh ghi.



2.2 Giới thiệu về QtCreator

Qt Creator là một môi trường phát triển tích hợp (IDE) đa nền tảng, cho phép các nhà phát triển tạo ứng dụng trên các nền tảng máy tính để bàn, di động và nhúng.

Thư viện Qt mà chúng ta thường nói đến là thư viện mà chúng ta gọi khi phát triển ứng dụng máy tính để bàn, nó sẽ được liên kết vào chương trình chạy của chúng ta. Còn Qt Creator là một công cụ, giúp chúng ta phát triển ứng dụng. Vì vậy, hai thứ này có thể coi là hoàn toàn khác nhau.

2.3 Mục tiêu thí nghiệm

① Nhận biết dự án Qt Creator ② Cấu

hình GDB trong Qt Creator để kết nối với bo mạch ③

Chuyển chương trình mục tiêu có thể chạy vào bo mạch ④

Truyền tham số cho chương trình chạy

2.4 Dự án Qt Creator

Tập dự án của Qt Creator là xxx.pro, trong tệp này "QT=" chỉ định các thành phần thư viện QT cần sử dụng, nếu để trống, có nghĩa là không gọi thư viện QT.

2.5 Lấy địa chỉ IP của bo mạch

- (1) Gắn thẻ phát triển của chúng tôi, lắp thẻ SD có hệ điều hành linux và cấp nguồn
- (2) Xem địa chỉ IP

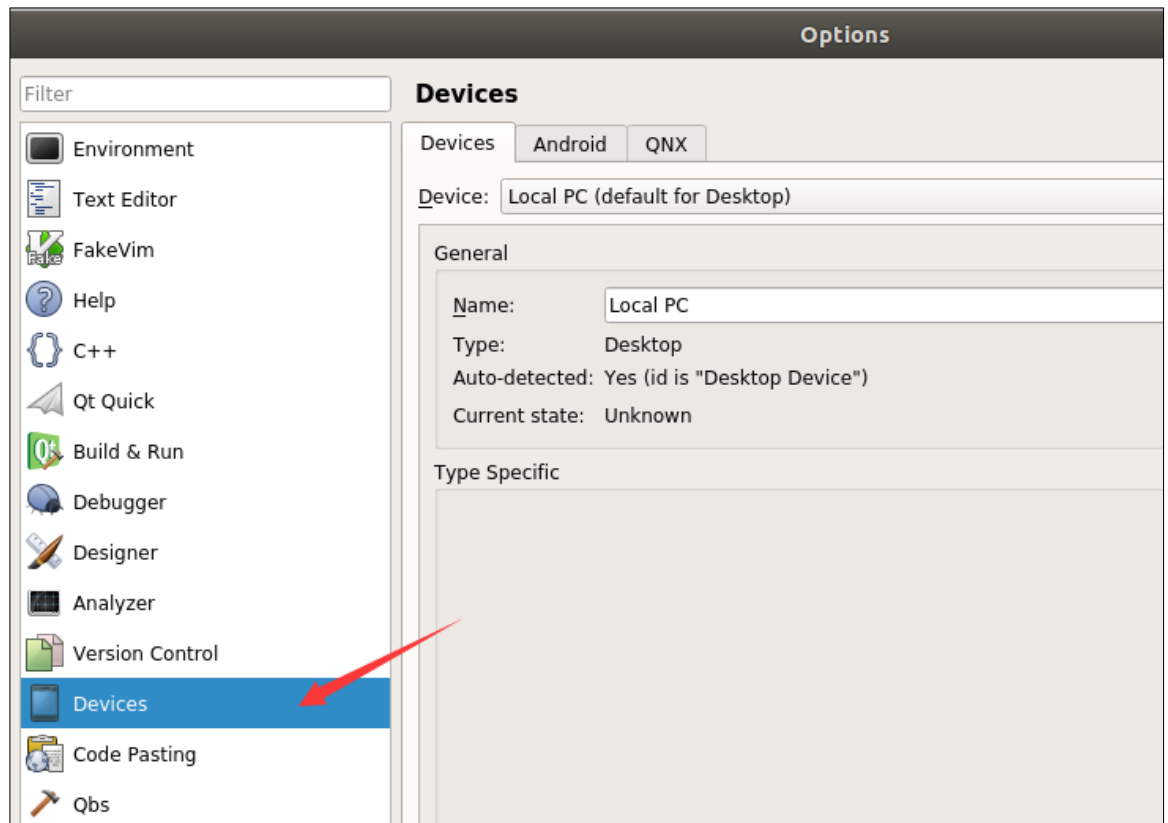
Có thể xem địa chỉ IP của card mạng bằng lệnh "ipconfig", hệ thống mặc định kích hoạt DHCP, nếu địa phương không hỗ trợ dịch vụ DHCP, kết quả truy vấn sẽ như hình dưới đây:

```
root@petalinux:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:01
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:29
```

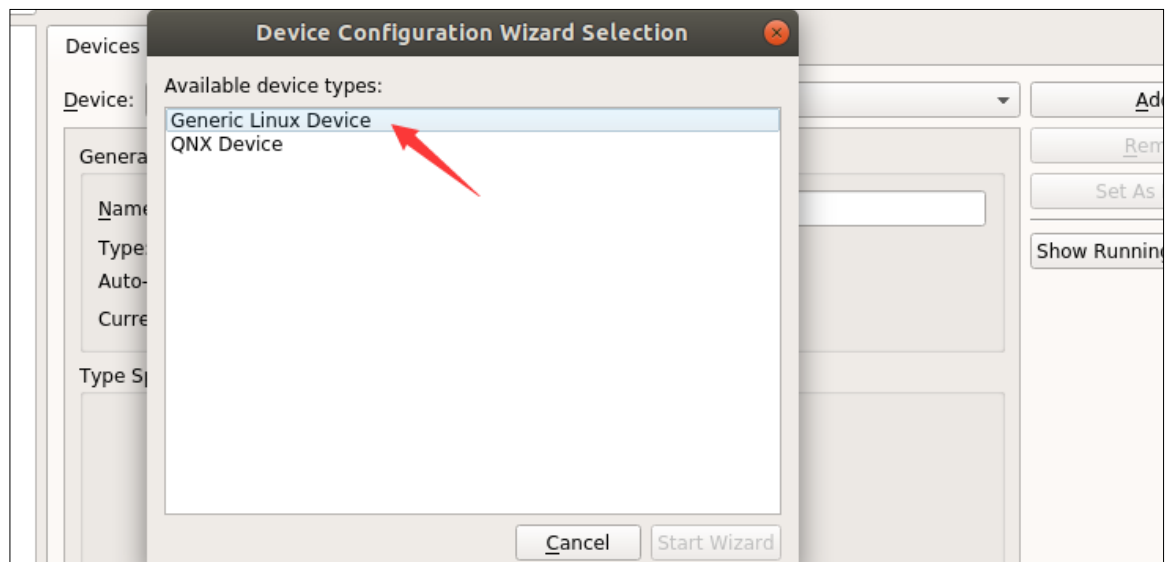
Lúc này cần phải phân bổ một địa chỉ IP một cách thủ công, có thể sử dụng lệnh "ipconfig eth0 192.168.1.45 netmask 255.255.255.0". Ở đây chúng ta thiết lập địa chỉ IP của card mạng là: 192.168.1.45

2.6 Cấu hình GDB kết nối với bo mạch trong QtCreator

- (1) Mở mục cài đặt công cụ của QtCreator



(2) Nhấn nút “Add...” và chọn “Generic Linux Device”

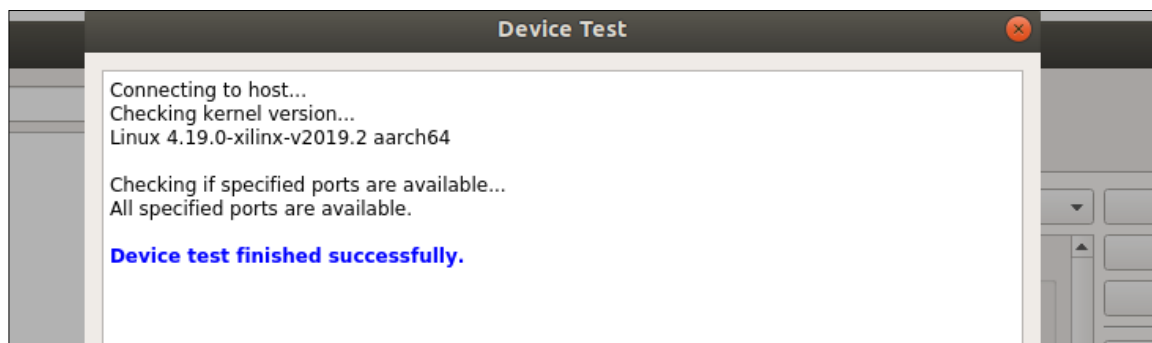


(3) Cài đặt tham số như sau



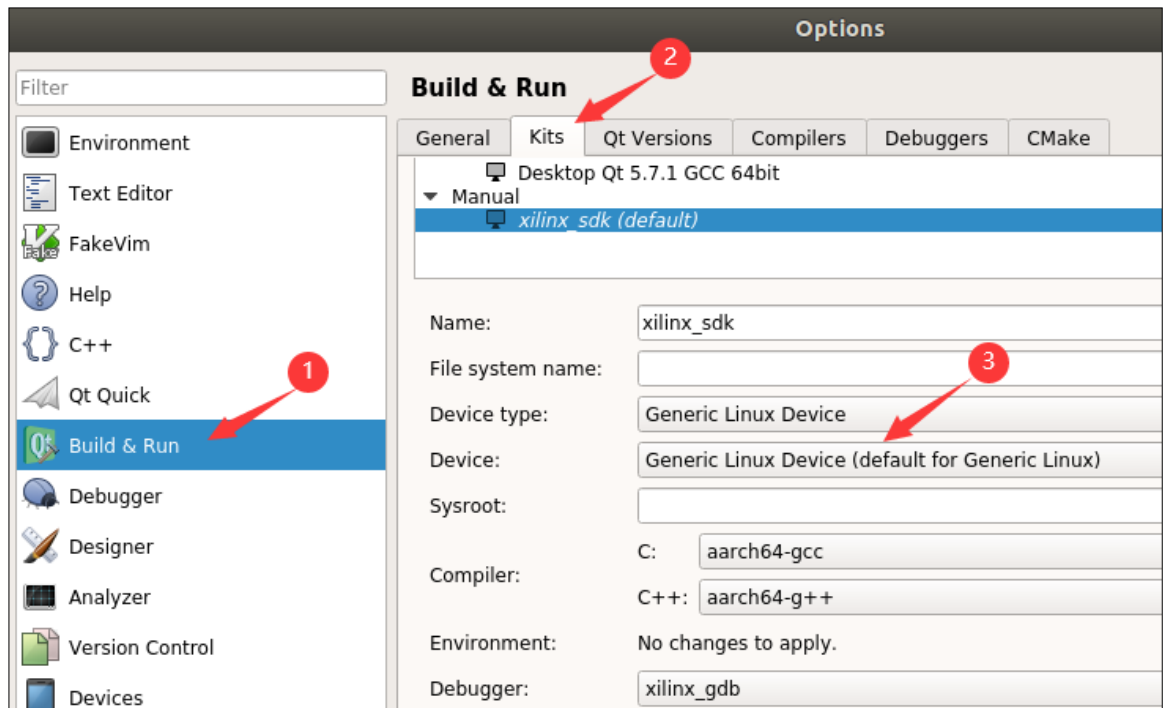
Mật khẩu mặc định là root, cần phải điền vào

- (4) Nhấn “Next” và “Finish”, giao diện thành công trong bài kiểm tra sau đó như sau



Nếu không thành công, cần kiểm tra kết nối mạng

- (5) Quay lại mục cài đặt “Device”, nhấn “Apply”, hoàn thành cài đặt này
(6) Chuyển đổi các mục cài đặt và cấu hình như sau



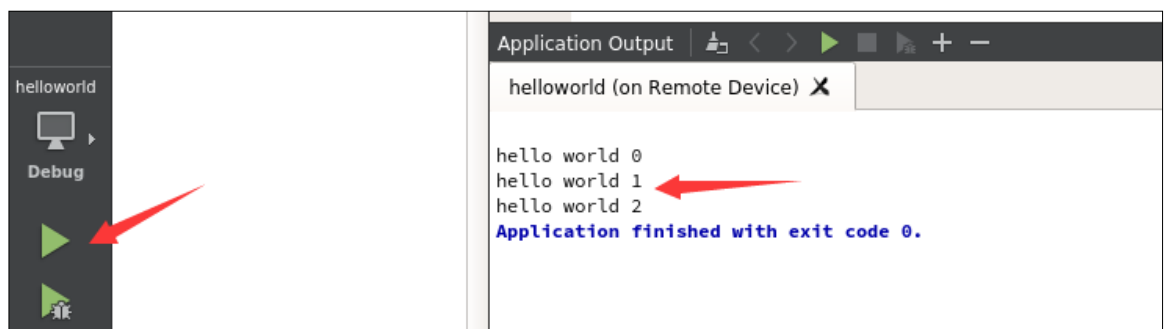
(7) Nhấn 'OK' để hoàn tất cài đặt

2.7 Chạy Hello World

- (1) Mở dự án helloworld
- (2) Biên dịch ứng dụng



- (3) Nhấn nút chạy trên QtCreator

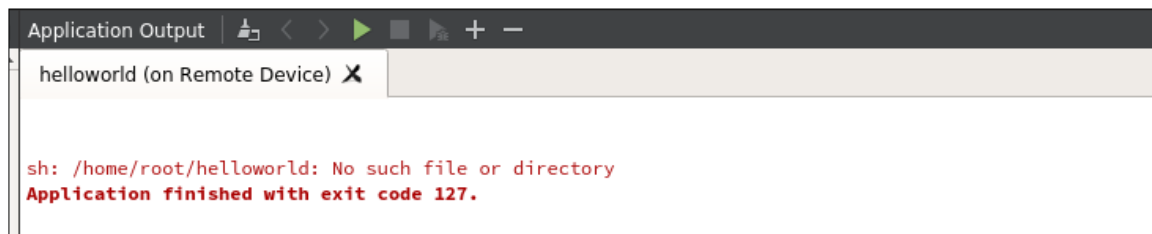


Chúng ta có thể thấy, thông tin in ra khi chạy ở bên phải, cho thấy chương trình đã chạy và thoát.

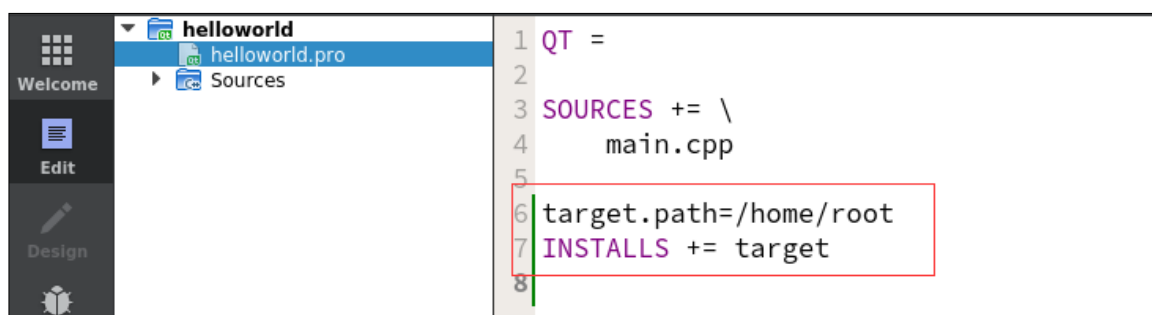
- ① Nếu xuất hiện lỗi như dưới đây, cần kiểm tra xem bo mạch có được cấp điện và vào hệ thống linux hay không, mạng có bình thường không.



- ② Nếu xuất hiện lỗi như dưới đây, cần xóa kết quả biên dịch và biên dịch lại.



- (4) Lúc này, chúng ta kiểm tra thư mục người dùng của bo mạch "/home/root", phát hiện có thêm một tệp helloworld.
(5) Chúng ta mở tệp pro của dự án

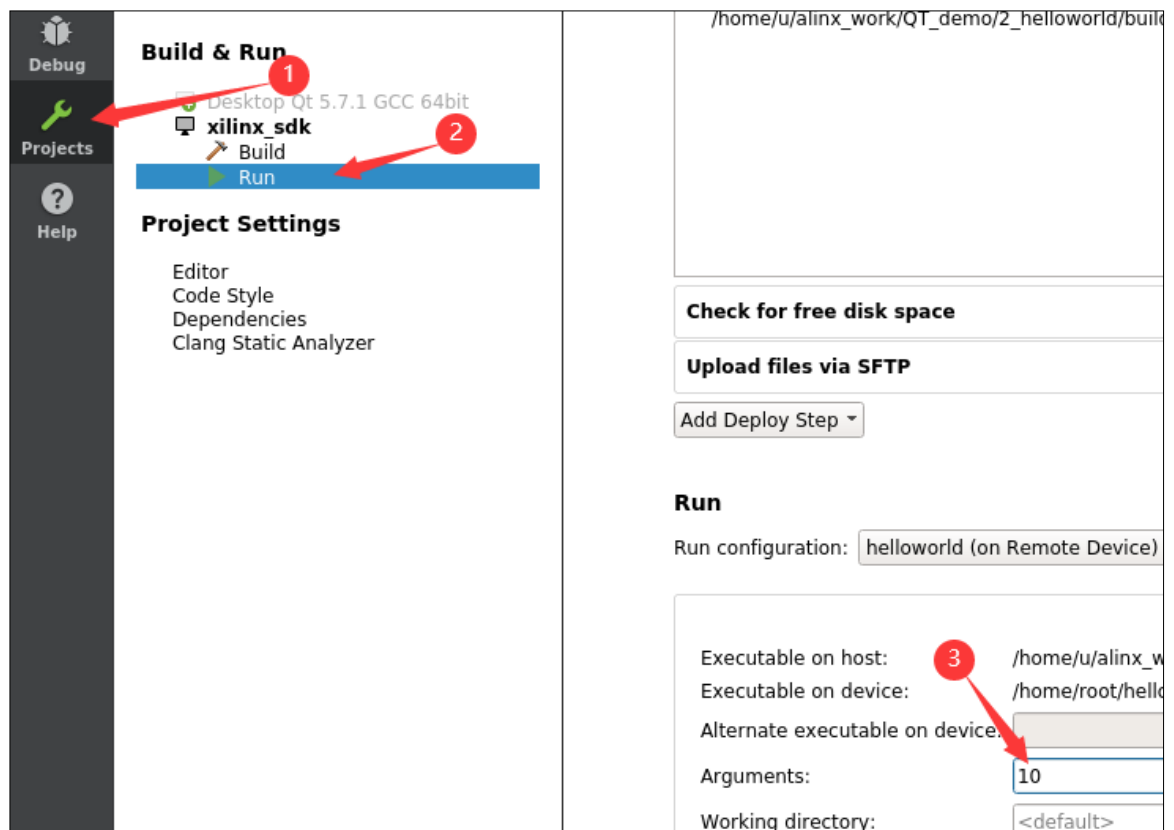


Khung màu đỏ, là thư mục mà chương trình tự động tải trong thời gian chạy, điều này giải thích tại sao lại xuất hiện tệp helloworld trong thư mục hệ thống của bo mạch phát triển "/home/root"

2.8 Cài đặt tham số chạy

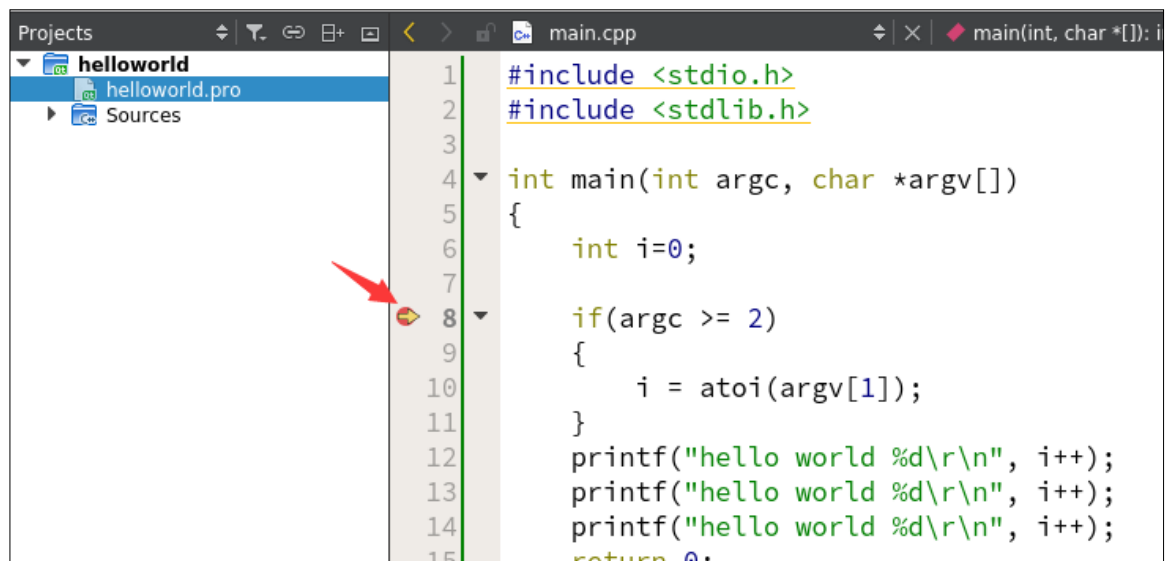
Đôi khi, khi chạy chương trình, chúng ta cần thêm một số tham số chạy ở phía sau, dưới đây chúng tôi sẽ giới thiệu cách thêm tham số chạy cho chương trình khi gỡ lỗi từ xa

- (1) Mở các mục cài đặt dưới đây

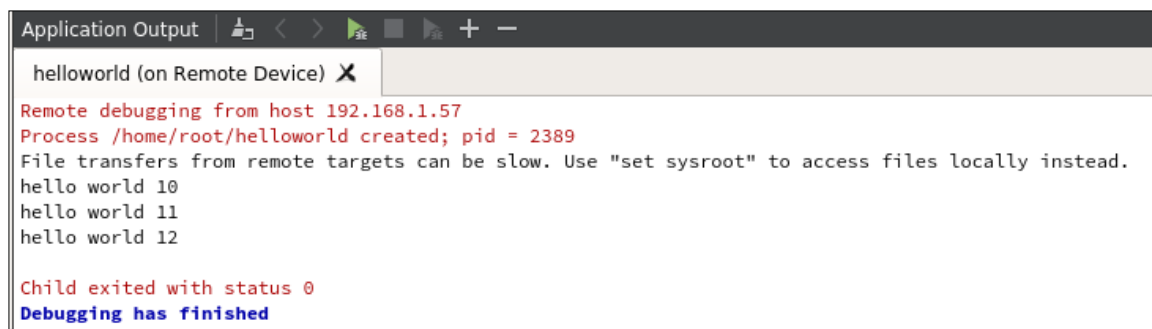


Tham số này được đặt là 10

- (2) Đặt một điểm dừng tại vị trí if trong hàm main
- (3) Nhấn nút debug hoặc phím tắt F5
- (4) Chúng ta thấy rằng chương trình đã dừng lại tại điểm dừng mà chúng ta đã thiết lập



- (5) Nhấn F5 để tiếp tục chạy, chúng ta thấy thông tin in ra như sau



```
Application Output | [Icons]
helloworld (on Remote Device) X
Remote debugging from host 192.168.1.57
Process /home/root/helloworld created; pid = 2389
File transfers from remote targets can be slow. Use "set sysroot" to access files locally instead.
hello world 10
hello world 11
hello world 12

Child exited with status 0
Debugging has finished
```

Chúng tôi thấy rằng, số bắt đầu sau "hello world" đã trở thành 10, cho thấy rằng tham số mà chúng tôi đã thiết lập đã có hiệu lực

Chương ba Phát hiện biên với OpenCV

3.1 Giới thiệu về OpenCV

OpenCV (Thư viện thị giác máy tính mã nguồn mở) là một thư viện phần mềm mã nguồn mở cho thị giác máy tính và học máy. OpenCV được xây dựng nhằm cung cấp một cơ sở hạ tầng chung cho các ứng dụng thị giác máy tính và tăng tốc độ áp dụng nhận thức máy móc trong các sản phẩm thương mại.

Là một sản phẩm được cấp phép theo BSD, OpenCV cho phép các doanh nghiệp dễ dàng sử dụng và sửa đổi mã nguồn.

Thư viện này có hơn 2500 thuật toán tối ưu hóa, bao gồm một bộ đầy đủ các thuật toán thị giác máy tính và học máy cổ điển cũng như hiện đại.

Các thuật toán này có thể được sử dụng để phát hiện và nhận diện khuôn mặt, nhận diện đối tượng, phân loại hành vi con người trong video, theo dõi chuyển động của camera, theo dõi các đối tượng chuyển động, trích xuất mô hình 3D của đối tượng, tạo ra đám mây điểm 3D từ camera stereo, ghép hình ảnh lại với nhau để tạo ra hình ảnh độ phân giải cao của toàn cảnh, tìm kiếm hình ảnh tương tự trong thư viện hình ảnh, loại bỏ hiện tượng mắt đỏ từ hình ảnh chụp bằng đèn flash, theo dõi chuyển động của mắt, nhận diện cảnh vật và xây dựng nhãn để phủ lên thực tế tăng cường, v.v. OpenCV có hơn 47000 cộng đồng người dùng, ước tính số lượt tải xuống vượt quá 1800 triệu. Nó được ứng dụng rộng rãi trong các công ty, nhóm nghiên cứu và cơ quan chính phủ.

Nhiều triển khai của reVision của Xilinx dựa trên giao diện của OpenCV, trong đó việc triển khai phần cứng là xfpencv. Điều này sẽ giúp chúng tôi thực hiện việc tăng tốc phần cứng (FPGA) cho các thuật toán tương ứng.

3.2 Giới thiệu về phát hiện biên

Phát hiện biên là một vấn đề cơ bản trong xử lý hình ảnh và thị giác máy tính, mục đích của phát hiện biên là xác định các điểm có sự thay đổi độ sáng rõ rệt trong hình ảnh số. Những thay đổi đáng chú ý trong thuộc tính hình ảnh thường phản ánh các sự kiện và biến đổi quan trọng của thuộc tính. Những điều này bao gồm sự không liên tục về độ sâu, sự không liên tục về hướng bề mặt, sự thay đổi thuộc tính vật chất và sự thay đổi ánh sáng trong cảnh. Phát hiện biên là một lĩnh vực nghiên cứu trong xử lý hình ảnh và thị giác máy tính, đặc biệt là trong việc trích xuất đặc trưng.

Thông tin biên của hình ảnh chủ yếu tập trung ở dải tần cao, thường nói đến việc làm sắc nét hình ảnh hoặc phát hiện biên, thực chất là lọc tần số cao. Chúng ta biết rằng phép toán vi phân là để tìm tỷ lệ thay đổi của tín hiệu, có tác dụng tăng cường thành phần tần số cao. Trong phép toán không gian, việc làm sắc nét hình ảnh là tính toán vi phân. Do tín hiệu hình ảnh số là rời rạc, phép toán vi phân trở thành việc tính toán sai khác hoặc độ dốc. Trong xử lý hình ảnh có nhiều toán tử phát hiện biên (độ dốc), thường dùng bao gồm sai khác bậc nhất thông thường, toán tử Robert (sai khác giao nhau), toán tử Sobel, v.v., dựa trên việc tìm kiếm cường độ độ dốc. Toán tử Laplace (sai khác bậc hai) dựa trên việc phát hiện điểm qua không. Bằng cách tính toán độ dốc, thiết lập ngưỡng, ta có được hình ảnh biên.

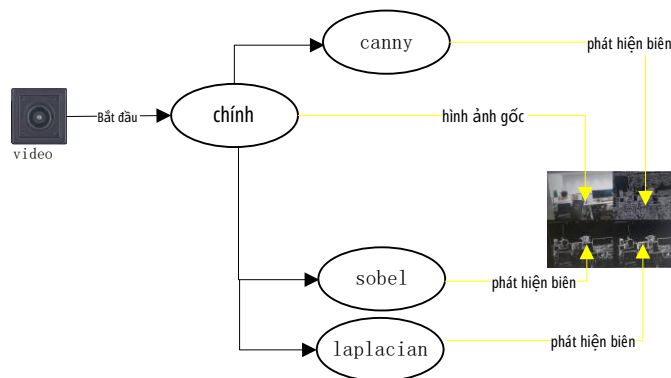
Các bước chính của phát hiện biên là:

- ① Lọc: Thuật toán phát hiện biên chủ yếu dựa trên đạo hàm bậc nhất và bậc hai của cường độ hình ảnh, nhưng đạo hàm rất nhạy cảm với tiếng ồn, do đó cần sử dụng bộ lọc để cải thiện hiệu suất của bộ phát hiện biên liên quan đến tiếng ồn
- ② Tăng cường: Cơ sở của việc tăng cường biên là xác định giá trị thay đổi cường độ của các điểm lân cận trong hình ảnh. Thuật toán tăng cường có thể làm nổi bật những điểm có giá trị cường độ lân cận thay đổi đáng kể
- ③ Phát hiện: Trong lân cận có nhiều điểm có giá trị gradient lớn, nhưng trong các ứng dụng cụ thể, những điểm này không phải là điểm biên cần tìm, cần phải lựa chọn

3.3 Mục tiêu thí nghiệm

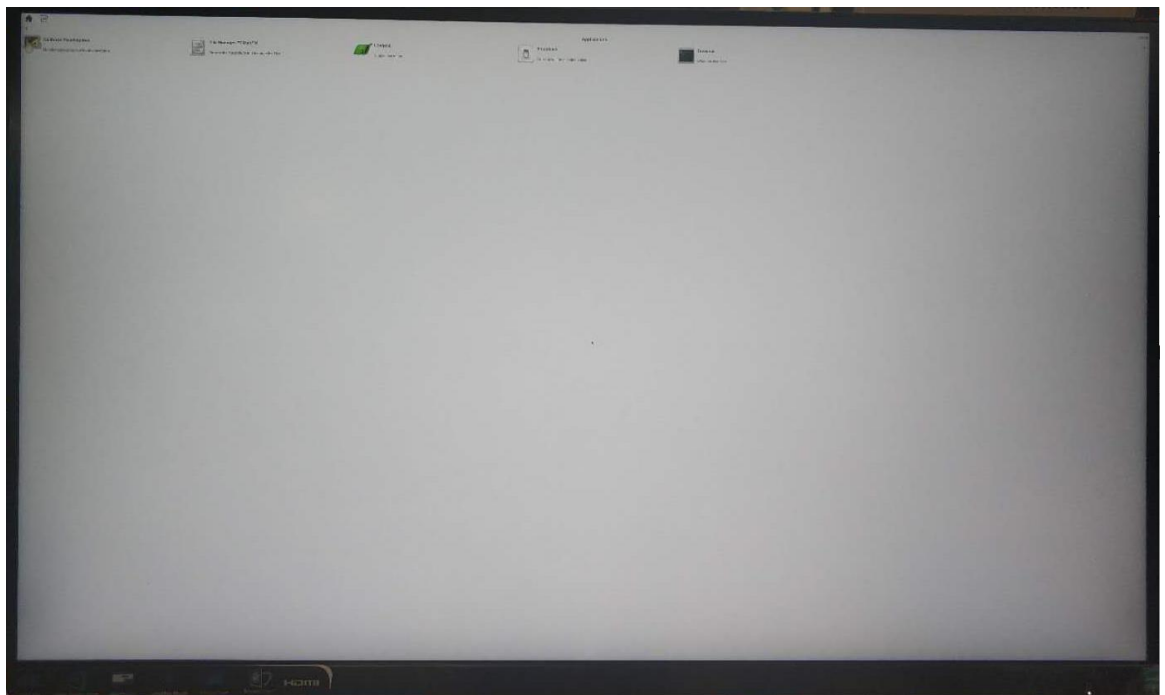
- ① Sử dụng opencv để bắt hình ảnh từ camera usb
- Phát hiện biên bằng các thuật toán canny, sobel, laplacian
- ③ Sử dụng opencv để hiển thị hình ảnh
- ④ Phản hồi thao tác đóng cửa sổ opencv

3.4 Sơ đồ quy trình phần mềm



Chuẩn bị chạy 3.5

- (1) Kết nối giao diện dp đến màn hình
- (2) Cắm camera usb vào
- (3) Khi bo mạch phát triển được cấp điện, sau khi hệ thống khởi động hoàn tất, màn hình dp có thể hiển thị đúng màn hình khởi động.



- (4) Tại terminal, có thể xem thấy thiết bị camera usb.

```
root@petalinux:~# ls /dev/video*
/dev/video0 /dev/video1
root@petalinux:~#
```

3.6 Chương trình đang chạy

- (1) Bạn có thể tham khảo ví dụ trong chương trước, mở dự án 3_opencv_edgedete, sử dụng QtCreator để chạy ứng dụng trực tiếp, hiệu ứng chạy như sau



3.7 Phân tích mã

- (1) Thiết lập biến môi trường

Nhập lệnh sau vào terminal, hiệu ứng tương đương với lệnh setenv trong mã.

```
xuất DISPLAY=:0.0
```

- (2) Khởi động desktop

```
/etc/init.d/xserver-nodm_xx bắt đầu
```

- (3) Điều chỉnh độ phân giải màn hình

Như trong thí nghiệm trên, sử dụng một màn hình 4K, vì vậy các biểu tượng trên màn hình khởi động rất nhỏ. Để có hiệu ứng hiển thị tốt hơn, trong ứng dụng, thông qua lệnh hệ thống, chúng ta sẽ thiết lập độ phân giải đầu ra hiển thị là 1080P. Lệnh như sau:

```
xrandr --output DP-1 --mode 1920x1080
```

- (4) Quản lý đầu ra màn hình

Hệ thống mặc định sử dụng quản lý nguồn đầu ra, tức là sau một khoảng thời gian, màn hình sẽ tắt. Lúc này, chúng ta có thể sử dụng lệnh,

Hủy bỏ chức năng này, lệnh như sau:

```
xset s 0
0xset dpms 0 0 0
```

(5) Thao tác đa luồng

Thí nghiệm này sử dụng ba thuật toán khác nhau để hoàn thành việc phát hiện biên, ba thuật toán được phân bổ cho ba nhiệm vụ luồng, có thể tận dụng tối đa bốn lõi A53 của ARM.

(6) Bắt hình từ camera USB

Độ phân giải của camera được thiết lập là 640x480, thiết bị camera mặc định được mở là: /dev/video0, vì việc tính toán thuật toán ở đây có chút chậm, hình ảnh hiển thị sẽ có một chút độ trễ.

(7) Phát hiện biên Canny

Thuật toán phát hiện biên đa cấp được John F. Canny phát triển vào năm 1986, được nhiều người coi là thuật toán phát hiện biên tối ưu. Ba tiêu chí chính để đánh giá phát hiện biên tối ưu là:

- Tỷ lệ sai sót thấp: Nhận diện càng nhiều biên thực tế càng tốt, đồng thời giảm thiểu tối đa các báo cáo sai do nhiễu.
- Định vị cao: Các biên được nhận diện phải gần gũi nhất với các biên thực tế trong hình ảnh.
- Phản ứng tối thiểu: Biên trong hình ảnh chỉ được nhận diện một lần.

cv : : hàm canny có các tham số như sau :

InputArray	hình ảnh,	hình ảnh đầu vào
OutputArray	edges	Output hình ảnh biên
double	ngưỡng1	Ngưỡng 1
double	Ngưỡng 2	Ngưỡng 2
số nguyên	kích thước kernel độ=3	Kích thước bộ lọc Sobel
bool	L2gradient=false	Có nên sử dụng cách tính toán chính xác hơn để tính toán độ dốc của hình ảnh

(8) phát hiện biên sobel

Đây là một phương pháp phát hiện biên khá phổ biến, bộ toán tử Sobel kết hợp giữa làm mịn Gaussian và đạo hàm vi phân, còn được gọi là toán tử đạo hàm bậc nhất, toán tử đạo hàm, thực hiện đạo hàm theo cả hai hướng ngang và dọc, thu được hình ảnh độ dốc theo hướng X và Y của hình ảnh. Nhược điểm: nhạy cảm, dễ bị ảnh hưởng, cần phải sử dụng làm mịn Gaussian (làm mịn) để giảm nhiễu, độ chính xác định vị biên không đủ cao.

Tham số của hàm cv::sobel như sau:

InputArray	src,	hình ảnh đầu vào
OutputArray	dst	Output hình ảnh biên
số nguyên	ddepth	Độ sâu hình ảnh đầu ra
số nguyên	dx	bậc của đạo hàm x.
số nguyên	dy	bậc của đạo hàm y
số nguyên	ksize = 3	Kích thước của kernel Sobel
double	scale = 1	Hệ số tỷ lệ tùy chọn cho giá trị đạo hàm được tính toán

double	delta = 0	Thêm giá trị gia tăng tùy chọn vào kết quả trước khi lưu trữ vào dst.
Int	borderType=BO RDER_DEFAULT	Phương pháp ngoại suy pixel

(9) Phát hiện biên Laplacian

Toán tử Laplace là một toán tử vi phân bậc hai trong không gian Euclid n chiều, phù hợp hơn khi chỉ quan tâm đến vị trí của biên mà không xem xét sự khác biệt độ xám của các pixel xung quanh. Toán tử Laplace phản ứng mạnh hơn với các pixel cô lập so với các biên hoặc đường, do đó chỉ thích hợp cho hình ảnh không có tiếng ồn. Trong trường hợp có tiếng ồn, cần phải thực hiện lọc thông thấp trước khi sử dụng toán tử Laplace để phát hiện biên. Vì vậy, các thuật toán phân đoạn thông thường thường kết hợp toán tử Laplace với toán tử làm mịn để tạo ra một mẫu mới. Toán tử Laplace cũng giống như toán tử Sobel, thuộc về các phép lọc làm sắc không gian.

cv : : Các tham số của hàm Laplace như sau :

InputArray	src,	hình ảnh đầu vào
OutputArray	dst	Output hình ảnh biên
số nguyên	ddepth	Độ sâu hình ảnh đầu ra
số nguyên	ksize = 1	Kích thước của hạt nhân để tính đạo hàm bậc hai
double	scale = 1	Hệ số tỷ lệ tùy chọn để tính Laplace
double	delta = 0	Thêm giá trị gia tăng tùy chọn vào kết quả trước khi lưu trữ vào dst.
Int	borderType=BO RDER_DEFAULT	Phương pháp ngoại suy pixel

Chương 4: Phát hiện khuôn mặt với OpenCV và Qt

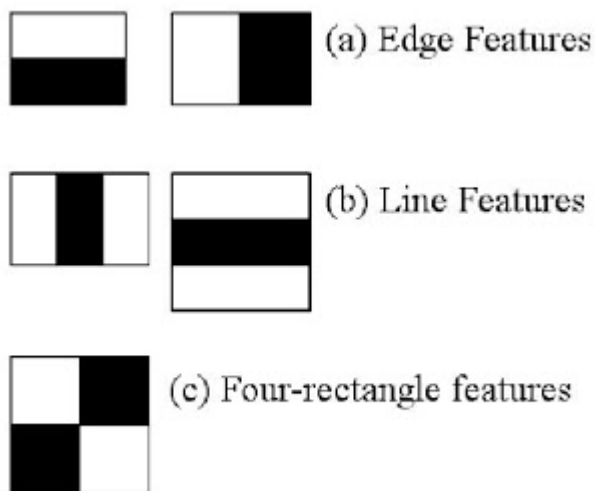
4.1 Bộ phân loại Cascade của OpenCV

Bộ phân loại là thiết bị dùng để xác định xem một sự vật có thuộc về một loại nào đó hay không. Bộ phân loại cascade: có thể hiểu là việc nối chuỗi N bộ phân loại đơn lẻ lại với nhau. Nếu một sự vật có thể thuộc về tất cả các bộ phân loại đã được nối lại này, thì kết quả cuối cùng sẽ được xác định là có, nếu có một bộ phân loại không phù hợp, thì sẽ được xác định là không. Ví dụ như khuôn mặt, nó có nhiều thuộc tính, chúng ta sẽ tạo một bộ phân loại cho mỗi thuộc tính, nếu một mô hình phù hợp với tất cả các thuộc tính mà chúng ta định nghĩa cho khuôn mặt, thì chúng ta coi mô hình đó là một khuôn mặt.

Vậy những thuộc tính này chỉ điều gì? Ví dụ, khuôn mặt cần có hai hàng lông mày, hai mắt, một cái mũi, một cái miệng, và một cái cằm hoặc đường viền hình chữ U.

Phát hiện đối tượng dựa trên bộ phân loại cascade với đặc trưng Haar là một phương pháp phát hiện đối tượng hiệu quả được Paul Viola và Michael Jones đề xuất trong bài báo năm 2001 "Phát hiện đối tượng nhanh chóng bằng cách sử dụng một chuỗi tăng cường của các đặc trưng đơn giản". Đây là một phương pháp dựa trên học máy, trong đó một hàm cascade được đào tạo từ nhiều hình ảnh tích cực và tiêu cực. Sau đó, nó được sử dụng để phát hiện đối tượng trong các hình ảnh khác.

Chúng ta sẽ thực hiện phát hiện khuôn mặt ở đây. Thuật toán này trước tiên cần một lượng lớn hình ảnh tích cực (hình ảnh khuôn mặt) và hình ảnh tiêu cực (hình ảnh không có khuôn mặt) để đào tạo bộ phân loại. Sau đó, chúng ta cần trích xuất các đặc trưng từ đó.



Các đặc trưng Haar được chia thành ba loại: đặc trưng cạnh, đặc trưng đường thẳng, đặc trưng trung tâm và đặc trưng đường chéo, kết hợp thành các mẫu đặc trưng. Trong mẫu đặc trưng có hai hình chữ nhật màu trắng và đen, và giá trị đặc trưng của mẫu được định nghĩa là tổng các pixel hình chữ nhật trắng trừ đi các pixel hình chữ nhật đen. Giá trị đặc trưng Haar phản ánh sự thay đổi độ xám của hình ảnh. Ví dụ: một số đặc trưng của khuôn mặt có thể được mô tả đơn giản bằng các đặc trưng hình chữ nhật, như: màu của mắt thường tối hơn má, hai bên sống mũi tối hơn sống mũi, và màu của miệng tối hơn xung quanh. Tuy nhiên, các đặc trưng hình chữ nhật chỉ nhạy cảm với một số cấu trúc hình học đơn giản, như cạnh và đoạn thẳng, vì vậy chỉ có thể mô tả các cấu trúc theo hướng cụ thể (ngang, dọc, chéo).

Bằng cách thay đổi kích thước và vị trí của mẫu đặc trưng, có thể liệt kê ra một lượng lớn các đặc trưng trong cửa sổ hình ảnh. Mẫu đặc trưng trong hình trên được gọi là "mẫu đặc trưng"; các đặc trưng thu được từ việc mở rộng (dịch chuyển và co giãn) mẫu đặc trưng trong cửa sổ hình ảnh được gọi là "đặc trưng hình chữ nhật"; giá trị của đặc trưng hình chữ nhật được gọi là "giá trị đặc trưng".

Đặc trưng hình chữ nhật có thể nằm ở bất kỳ vị trí nào trong hình ảnh, kích thước cũng có thể thay đổi tùy ý, vì vậy giá trị đặc trưng hình chữ nhật là hàm của ba yếu tố: loại mẫu, vị trí hình chữ nhật và kích thước hình chữ nhật.

Do sự thay đổi về loại, kích thước và vị trí, một cửa sổ phát hiện rất nhỏ có thể chứa rất nhiều đặc trưng hình chữ nhật, ví dụ: trong một cửa sổ phát hiện kích thước 24*24 pixel, số lượng đặc trưng hình chữ nhật có thể đạt tới 160.000. Như vậy, có hai vấn đề cần giải quyết: (1) Làm thế nào để tính toán nhanh chóng số lượng lớn các đặc trưng đó? (2) Những đặc trưng hình chữ nhật nào là hiệu quả nhất cho việc phân loại của bộ phân loại?

4.2 Biểu đồ tích phân

Để tính toán đặc trưng một cách nhanh chóng, ở đây đã giới thiệu biểu đồ tích phân, bất kể hình ảnh lớn như thế nào, nó sẽ giảm thiểu việc tính toán của pixel đã cho xuống chỉ liên quan đến bốn pixel. Biểu đồ tích phân là một thuật toán nhanh cho phép tính tổng pixel của tất cả các vùng trong hình ảnh chỉ bằng cách duyệt qua hình ảnh một lần, từ đó nâng cao hiệu quả tính toán giá trị đặc trưng của hình ảnh.

Ý tưởng chính của biểu đồ tích phân là lưu trữ tổng pixel của các vùng hình chữ nhật từ điểm bắt đầu đến từng điểm trong một mảng, khi cần tính tổng pixel của một vùng nào đó, có thể trực tiếp truy cập vào phần tử của mảng mà không cần tính toán lại tổng pixel của vùng đó, từ đó tăng tốc độ tính toán (điều này có một tên gọi tương ứng, gọi là thuật toán lập trình động). Biểu đồ tích phân có thể tính toán các đặc trưng khác nhau với cùng một thời gian (thời gian hằng số) ở nhiều quy mô khác nhau, do đó tăng tốc độ phát hiện một cách đáng kể.

Biểu đồ tích phân là một phương pháp biểu diễn ma trận có thể mô tả thông tin toàn cục. Cách xây dựng biểu đồ tích phân là giá trị tại vị trí (i,j) là tổng của tất cả các pixel theo hướng góc trái trên của hình ảnh gốc tại (i,j) :

$$ii(i,j) = \sum_{k \leq i, l \leq j} f(k,l)$$

Thuật toán xây dựng hình ảnh tích phân

- ① Sử dụng $s(i,j)$ để biểu thị tổng tích lũy theo hướng hàng, khởi tạo $s(i,-1)=0$;
- ② Sử dụng $ii(i,j)$ để biểu thị một hình ảnh tích phân, khởi tạo $ii(-1,i)=0$;
- ③ Quét hình ảnh theo từng hàng, tính toán đệ quy giá trị tổng tích lũy theo hàng $s(i,j)$ và giá trị hình ảnh tích phân $ii(i,j)$ của mỗi pixel (i,j) với $s(i,j)=s(i,j-1)+f(i,j)$

$$ii(i,j)=ii(i-1,j)+s(i,j)$$

- ④ Quét qua hình ảnh một lần, khi đến pixel ở góc dưới bên phải của hình ảnh, hình ảnh tích phân ii đã được xây dựng xong.

Sau khi xây dựng xong hình ảnh tích phân, tổng pixel của bất kỳ khu vực ma trận nào trong hình ảnh có thể được tính toán thông qua phép toán đơn giản như hình minh họa.

A	B	
C	D	

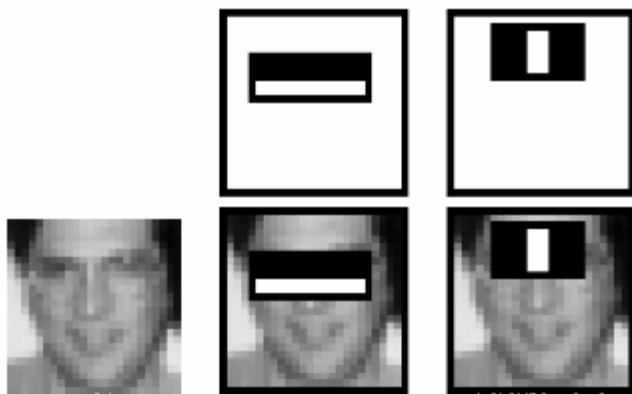
Gọi bốn đỉnh của D lần lượt là α , β , γ , δ , thì tổng pixel của D có thể được biểu diễn

$$D_{sum} = ii(\alpha) + ii(\beta) - (ii(\gamma) + ii(\delta));$$

Giá trị đặc trưng Haar-like chỉ đơn giản là sự chênh lệch giữa tổng các pixel của hai ma trận, và cũng có thể hoàn thành trong thời gian hằng số. Do đó, việc tính toán giá trị của các đặc trưng hình chữ nhật chỉ liên quan đến bảng tích phân của các điểm cuối của hình chữ nhật đặc trưng này, vì vậy bất kể sự thay đổi tỷ lệ của hình chữ nhật đặc trưng này là như thế nào, thời gian tiêu tốn để tính toán giá trị đặc trưng luôn là hằng số. Chỉ cần duyệt qua hình ảnh một lần, ta có thể tính toán được tất cả các giá trị đặc trưng của các cửa sổ con.

4.3 Chọn đặc trưng

Trong tất cả các đặc trưng mà chúng ta đã tính toán, hầu hết đều không liên quan. Ví dụ, hãy xem bức ảnh dưới đây. Hàng đầu tiên hiển thị hai đặc trưng rất tốt. Đặc trưng đầu tiên được chọn dường như tập trung vào khu vực mắt thường tối hơn khu vực mũi và má. Đặc trưng thứ hai được chọn phụ thuộc vào việc mắt tối hơn sống mũi. Nhưng cùng một cửa sổ áp dụng cho má hoặc bất kỳ nơi nào khác đều không có tác dụng. Vậy làm thế nào chúng ta có thể chọn được những đặc trưng tốt nhất từ hơn 160000 đặc trưng? Ở đây có thể thông qua Sử dụng thuật toán AdaBoost.



Chúng tôi sẽ áp dụng từng đặc trưng cho tất cả các hình ảnh huấn luyện. Đối với mỗi đặc trưng, nó sẽ tìm ra ngưỡng tốt nhất để phân loại khuôn mặt thành hai loại: tích cực và tiêu cực. Rõ ràng, sẽ có phân loại đúng hoặc sai. Chúng tôi chọn đặc trưng có tỷ lệ sai sót thấp nhất, điều này có nghĩa là chúng là những đặc trưng phân loại chính xác nhất cho hình ảnh khuôn mặt và không phải khuôn mặt. (Quá trình không đơn giản như vậy. Mỗi hình ảnh ban đầu có trọng số bằng nhau. Sau mỗi lần phân loại, trọng số của hình ảnh phân loại sai sẽ tăng lên. Sau đó thực hiện lại quá trình tương tự. Tính toán tỷ lệ sai sót mới. Còn có trọng số mới. Quá trình này sẽ tiếp tục cho đến khi đạt được độ chính xác hoặc tỷ lệ sai sót mong muốn hoặc tìm thấy số lượng đặc trưng cần thiết).

4.4 Giới thiệu về Qt

Qt là một thư viện phát triển C++ đa nền tảng, chủ yếu được sử dụng để phát triển các chương trình giao diện người dùng đồ họa (Graphical User Interface, GUI), tất nhiên cũng có thể phát triển các chương trình dòng lệnh (Command User Interface, CUI) không có giao diện. Qt cũng có các liên kết với các ngôn ngữ kịch bản như Python, Ruby, Perl, có nghĩa là có thể sử dụng ngôn ngữ kịch bản để phát triển các chương trình dựa trên Qt.

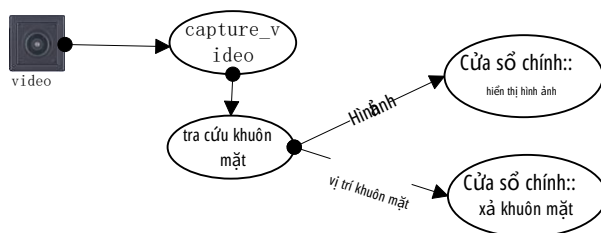
Qt hỗ trợ nhiều hệ điều hành khác nhau, chẳng hạn như hệ điều hành phổ thông Windows, Linux, Unix, hệ điều hành điện thoại thông minh Android, iOS, WinPhone, và các hệ thống nhúng như QNX, VxWorks, v.v. Mặc dù Qt thường được coi là một thư viện GUI để phát triển ứng dụng giao diện đồ họa, nhưng đó không phải là tất cả những gì Qt có; ngoài việc có thể vẽ những giao diện đẹp (bao gồm các điều khiển, bố cục, tương tác), Qt còn bao gồm nhiều chức năng khác, chẳng hạn như đa luồng, truy cập cơ sở dữ liệu, xử lý hình ảnh, xử lý âm thanh và video, giao tiếp mạng, thao tác tệp, v.v., tất cả đều đã được tích hợp trong Qt.

Mặc dù Qt cũng hỗ trợ các hệ điều hành di động, nhưng do Android đã có Java và Kotlin, iOS đã có Objective-C và Swift, nên thị phần của Qt trên di động gần như có thể bỏ qua.

4.5 Mục tiêu thí nghiệm

- ① Gọi thư viện Qt để hiển thị hình ảnh video và khung
- mặt② Sử dụng lớp CascadeClassifier của thư viện opencv

4.6 Sơ đồ quy trình phần mềm

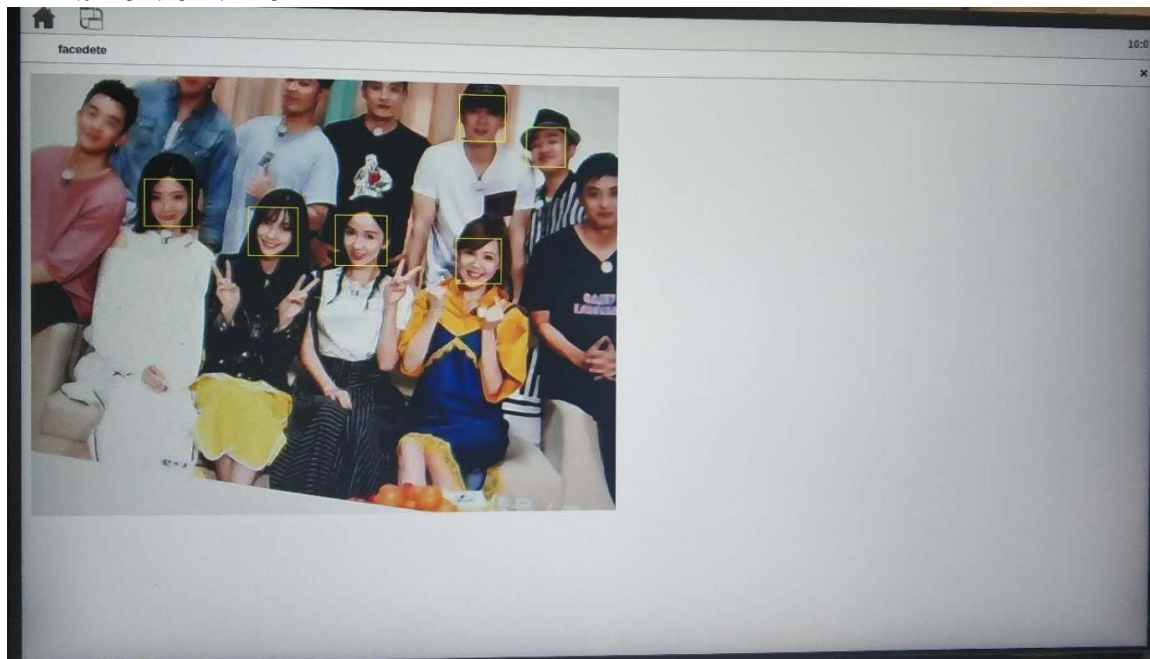


4.7 Chuẩn bị chạy

- (1) Chuẩn bị công việc theo chương trước
- (2) Sao chép thư mục haar_train vào thư mục /home/root trên bo mạch phát triển

4.8 Chạy chương trình

- (1) Chạy ứng dụng, hiệu ứng như sau



4.9 Phân tích mã

- (1) Việc phát hiện khuôn mặt chủ yếu diễn ra trong tệp `face_lookup.cpp` thông qua `face_cascade.detectMultiScale`, trong quá trình khởi tạo, gọi `face_cascade.load` để tải mô hình đã được đào tạo `haarcascade_frontalface_alt.xml`.
- (2) Trong `MainWindow::flushFace`, sẽ hiển thị khung khuôn mặt dựa trên kết quả phát hiện ở trên.
- (3) Do lý do huấn luyện mô hình, cần phải kiểm tra, khuôn mặt không được nghiêng hoặc ở bên cạnh.
- (4) `.pro` file
Vì gọi giao diện Q, trong tệp `facetedete.pro`, đã thêm các thành phần Qt core gui widgets sau "QT=".
- (5) Tín hiệu và khe cắm của Qt
Chương trình sẽ liên kết tín hiệu và khe cắm của `face_lookup` và `MainWindow`, để `MainWindow` xử lý hình ảnh cần hiển thị và đánh dấu vị trí của khuôn mặt.
- (6) Tương tự như ví dụ trên, hiệu ứng hiển thị khá giật lag.

Chương năm Hiện thị camera GStreamer

5.1 Giới thiệu về GStreamer

GStreamer là một khung ứng dụng để tạo ra các ứng dụng phát trực tuyến. Ý tưởng thiết kế cơ bản của nó đến từ ý tưởng về ống dẫn video của Trường Cao học Oregon, đồng thời cũng kế thừa ý tưởng thiết kế của DirectShow. Các đặc điểm của nó như sau:

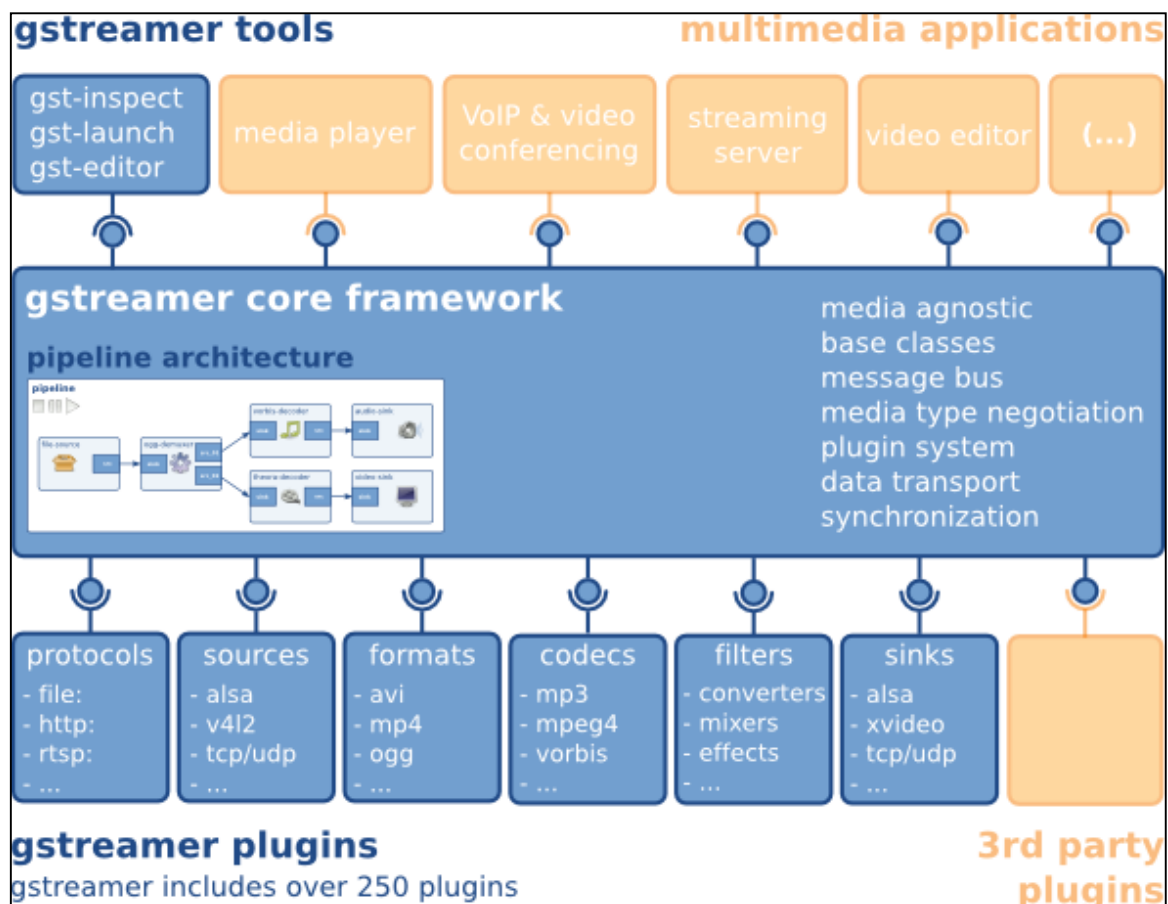
① Cấu trúc rõ ràng, chức năng mạnh mẽ②

Tư tưởng lập trình hướng đối tượng③ Tính

năng mở rộng linh hoạt④ Hiệu suất cao

⑤ Tách biệt thư viện lõi và plugin

Kiến trúc của GStreamer như sau, nó hỗ trợ các ứng dụng từ phát lại Ogg/Vorbis đơn giản, phát trực tuyến âm thanh/video đến xử lý âm thanh (trộn) và video (biên tập phi tuyến) phức tạp. Các ứng dụng có thể sử dụng một cách minh bạch các bộ mã hóa và công nghệ lọc tiên tiến.



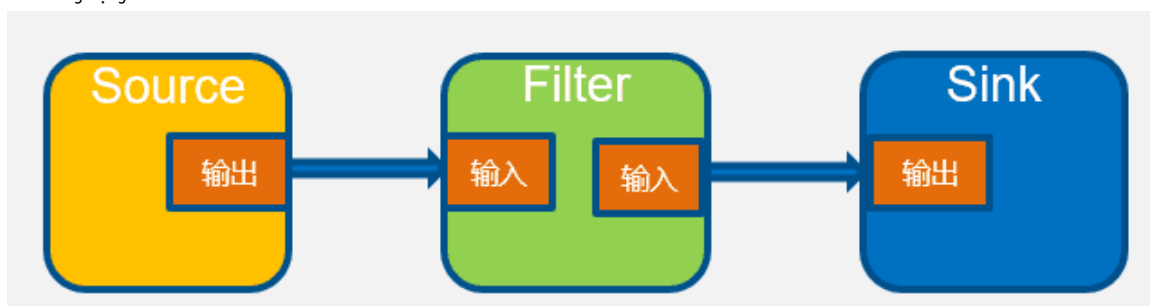
5.2 Các khái niệm cơ bản về GStreamer

(1) 元件 (element)

Elements là các cấu kiện cơ bản nhất tạo thành ống dẫn. Có thể kết nối nhiều 元件 (Elements) lại với nhau để tạo thành một ống dẫn (pipeline) nhằm hoàn thành một nhiệm vụ đặc biệt, chẳng hạn như phát media hoặc ghi hình. Elements được phân loại theo chức năng:

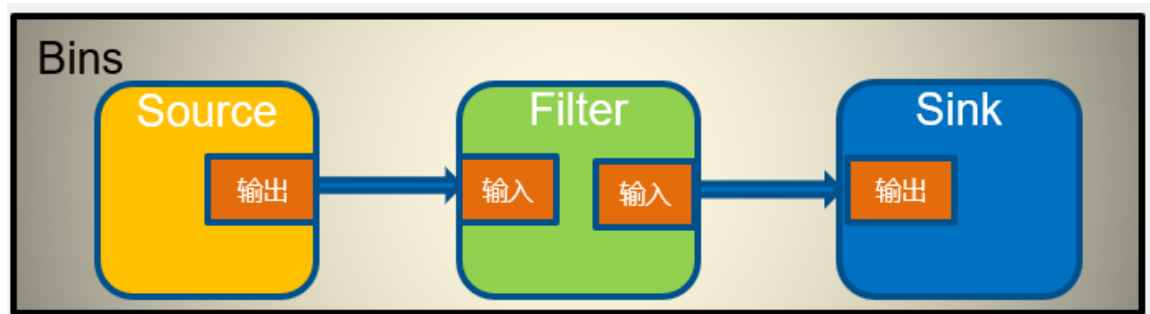
- ① Các yếu tố nguồn
- ② Các phần tử Sink
- ③ Các phần tử Lọc

Sơ đồ ứng dụng như sau:



(2) Thùng (bin) và Ống dẫn (Pipeline)

Hộp chứa (Bins) là một cái thùng có thể chứa các thành phần (element). Đường ống (pipelines) là một loại đặc biệt của hộp chứa (Bins), đường ống (pipelines) có thể thao tác với tất cả các thành phần (element) nằm bên trong nó. Như hình dưới đây



(3) Tấm đệm (Pad)

Tấm đệm (Pads) tương đương với giao diện của một thành phần, các thành phần (element) kết nối với nhau thông qua giao diện này, như vậy dữ liệu có thể được truyền tải giữa các thành phần này. Pads được phân loại:

- ① Loại cố định (always)
- ② Loại ngẫu nhiên (sometimes)
- ③ Loại theo yêu cầu (on request)

(4) Chức năng (Cap)

Caps mô tả định dạng dữ liệu có thể được truyền qua đệm hoặc định dạng dữ liệu hiện tại qua đệm. Có thể xem các loại phương tiện mà thành phần hỗ trợ bằng lệnh `gst-inspect-1.0`.

(5) Bus (Tuyến)

Mỗi đường ống mặc định bao gồm một tuyến, ứng dụng không cần phải tạo thêm tuyến. Ứng dụng thiết lập trên tuyến.

Một bộ xử lý tin nhắn. Khi vòng lặp chính đang chạy, bus sẽ kiểm tra bộ xử lý tin nhắn này xem có tin nhắn mới hay không, khi tin nhắn được thu thập, bus sẽ gọi hàm callback tương ứng để hoàn thành nhiệm vụ

(6) Bộ đệm (Buffer)

Bộ đệm chứa dòng dữ liệu trong ống dẫn được tạo ra. Thông thường, một thành phần nguồn sẽ tạo ra một bộ đệm mới, đồng thời thành phần đó cũng sẽ truyền dữ liệu của bộ đệm cho thành phần tiếp theo. Khi sử dụng cấu trúc GStreamer ở mức thấp để tạo một ống dẫn phương tiện, không cần phải tự xử lý bộ đệm, các thành phần sẽ tự động xử lý những bộ đệm này. Bộ đệm chủ yếu được cấu thành từ các thành phần sau:

① Con trỏ chỉ đến một vùng nhớ ② Kích

thước của vùng nhớ

③ Dấu thời gian của bộ đệm

④ Một bộ đếm tham chiếu, chỉ ra số lượng thành phần đang sử dụng bộ đệm. Khi không còn thành phần nào có thể tham chiếu, bộ đếm này sẽ được sử dụng để hủy bộ đệm

(7) Sự kiện (Events)

Các sự kiện bao gồm thông tin điều khiển trong ống dẫn, như tìm kiếm thông tin và tín hiệu kết thúc luồng.

5.3 Mục tiêu thí nghiệm

① Sử dụng công cụ gstreamer ②

Lập trình gstreamer

5.4 Công cụ thường dùng của Gstreamer

(1) gst-inspect-1.0

Không có tham số, nó sẽ liệt kê tất cả các element có sẵn, tức là tất cả các phần tử bạn có thể sử dụng, nếu có một tên tệp, nó sẽ mở tệp đó như một plugin của GStreamer, cố gắng mở nó và liệt kê tất cả các element bên trong, nếu có một element của GStreamer, nó sẽ liệt kê tất cả thông tin của element đó.

In danh sách plugin và thông tin của chúng, ví dụ:

In danh sách các plugin hỗ trợ in	gst-inspect-1.0 -a
In thông tin plugin filesrc	gst-inspect-1.0 filesrc

Mẫu Pad: Phần này sẽ liệt kê tất cả các loại Pad và các Caps của chúng. Qua đó, bạn có thể xác nhận xem có thể kết nối với một element nào đó hay không.

(2) gst-discoverer

Công cụ này có thể được sử dụng để xem các Caps có trong tệp, nó là một lớp bao bọc đối tượng GstDiscoverer. Chấp nhận một URI được nhập từ dòng lệnh và in ra tất cả thông tin. Điều này rất hữu ích khi xem cách mã hóa và tái sử dụng phương tiện, từ đó chúng ta có thể xác định nên đặt element nào vào pipeline.

(3) gst-launch-1.0

Công cụ này có thể tạo ra một pipeline, khởi tạo và sau đó chạy. Nó cho phép bạn nhanh chóng kiểm tra xem liệu nó có hoạt động hay không trước khi chính thức viết mã để thực hiện pipeline.



5.6 Chương trình chạy chương trình

Thông qua các lệnh, chúng ta có thể nhanh chóng xác minh nhiều phương án mà chúng ta đã hình dung, nhưng nếu muốn có nhiều giám sát trạng thái hơn hoặc kiểm soát của người dùng, thì điều này cần phải được thực hiện thông qua lập trình.

Ví dụ trong chương này là để thực hiện hiệu ứng của các lệnh trên thông qua chương trình.

- (1) Cần lưu ý rằng các vấn đề cần chú ý giống như ở trên, cần sửa đổi độ phân giải và định dạng mà camera USB hỗ trợ.
- (2) Chương trình thông qua việc thêm hàm callback sự kiện tin nhắn sink_message, theo dõi và xử lý các thay đổi trạng thái khác nhau.
- (3) Sau khi chạy chương trình, bạn có thể thấy hình ảnh hiển thị giống như hiệu ứng của lệnh chạy, lúc này nếu rút camera ra, thông tin mà chương trình in ra như sau và chương trình sẽ thoát khỏi việc chạy.

```
get error: Could not read from resource.
play error
playing out
Application finished with exit code 0.
```

5.7 Phân tích mã

(1) gst_parse_launch

Hàm này tạo ra một pipeline, các tham số của hàm như sau:

const gchar *	pipeline_description	Mô tả ống dẫn
GError **	lỗi	Trả về lỗi

Nội dung mô tả pipeline gần như giống hệt với nội dung đã được xác minh trước đó bằng gst-launch-1.0.

(2) gst_bus_add_watch

Thêm hàm callback cho bus, xử lý các thông điệp cần thiết.

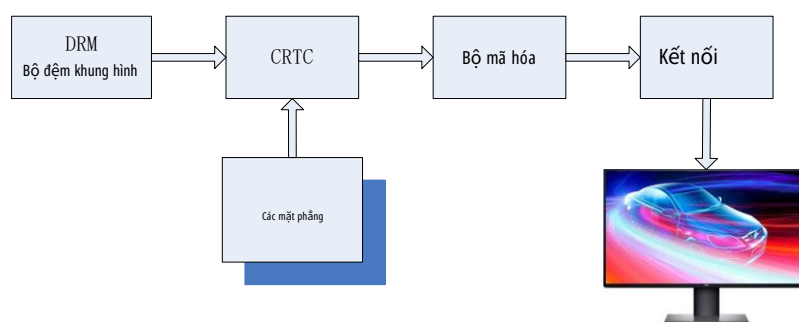
Chương sáu: Hiển thị camera bằng Qt+DRM+Gstreamer

6.1 Giới thiệu về DRM

DRM, viết tắt của Direct Rendering Manager, tức là Trình quản lý kết xuất trực tiếp. Nó được tạo ra để giải quyết vấn đề phối hợp sử dụng tài nguyên của Video Card giữa nhiều chương trình. Nó cung cấp cho không gian người dùng một tập hợp các API để quản lý đầu ra.

DRM hiện nay đã bao gồm nhiều chức năng mà trước đây được xử lý bởi các chương trình không gian người dùng, chẳng hạn như quản lý bộ đệm khung và thiết lập chế độ, đối tượng chia sẻ bộ nhớ và đồng bộ bộ nhớ. Một số mở rộng có tên gọi cụ thể, chẳng hạn như Trình quản lý thực thi đồ họa GEM hoặc Thiết lập chế độ nhân KMS, tất cả đều thuộc về hệ thống con DRM.

Hiển thị DRM chủ yếu liên quan đến 5 mô-đun sau



(1) Bộ đệm khung hình DRM

Theo quan điểm của các nhà phát triển, Bộ đệm khung hình là một vùng nhớ, việc ghi dữ liệu theo định dạng cụ thể vào vùng nhớ có nghĩa là xuất nội dung ra màn hình. Vì vậy, Bộ đệm khung hình giống như một bức tranh. Ví dụ, đối với Bộ đệm khung hình được khởi tạo với 16 bit màu, hai byte trong Bộ đệm khung hình đại diện cho một điểm trên màn hình, từ trên xuống dưới, từ trái sang phải, vị trí trên màn hình và địa chỉ trong bộ nhớ có mối quan hệ tuyến tính theo thứ tự.

(2) CRTC

Nhiệm vụ của CRTC là đọc hình ảnh cần hiển thị từ Framebuffer và xuất ra định dạng tương ứng cho Encoder.

Mặc dù CRTC theo nghĩa đen có nghĩa là bộ điều khiển ống tia âm cực, nhưng CRT đã bị loại bỏ trong các thiết bị hiển thị thông thường, CRTC trong DRI chủ yếu đảm nhận các vai trò sau:

- ① Cấu hình độ phân giải phù hợp với màn hình (kernel) và xuất ra thời gian tương ứng (logic phần cứng)
- ② Quét framebuffer để hiển thị lên một hoặc nhiều thiết bị hiển thị

(3) Planes

Với sự cập nhật không ngừng của công nghệ phần mềm, yêu cầu về hiệu suất phần cứng ngày càng cao, trong khi vẫn đảm bảo chức năng sử dụng bình thường, yêu cầu về tiêu thụ năng lượng cũng ngày càng khắt khe hơn. Ban đầu, GPU có thể xử lý tất cả các nhiệm vụ đồ họa, nhưng do tiêu thụ năng lượng quá cao khi hoạt động, các nhà thiết kế đã quyết định giao một phần nhiệm vụ đơn giản cho Bộ điều khiển hiển thị (Display Controller) xử lý (chẳng hạn như tổng hợp), trong khi để GPU tập trung vào việc vẽ (tức là kết xuất) nhiệm vụ chính, giảm bớt gánh nặng cho GPU, từ đó đạt được mục tiêu giảm tiêu thụ năng lượng và nâng cao hiệu suất. Do đó, Plane (đơn vị lớp phần cứng) đã ra đời. Planes chủ yếu là để chồng nhiều lớp lại với nhau để xuất ra, một số phần cứng hỗ trợ di chuyển, thu phóng, cắt, kênh alpha của các lớp.

(4) Bộ mã hóa (Encoder)

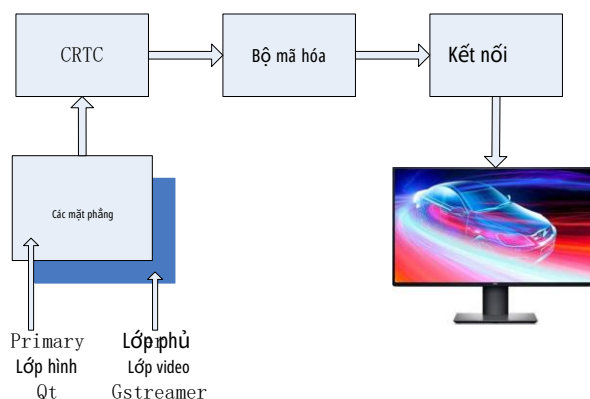
Mã hóa tín hiệu hình ảnh theo định dạng nhất định (như RGB, YUV, v.v.) thành tín hiệu mà kết nối cần xuất ra. Lấy ví dụ HDMI, đồng bộ khung/hàng/nội dung hiển thị đều được xuất qua bus nối tiếp dữ liệu TMDS, do đó, thời gian song song theo tiêu chuẩn HDMI được mã hóa thành thứ tự nối tiếp là nhiệm vụ của Bộ mã hóa.

(5) Kết nối

Kết nối thực chất là giao diện vật lý kết nối với màn hình, thường gặp có VGA/HDMI/DVI/DP, có thể đọc các tham số hỗ trợ của màn hình thông qua kết nối;

6.2 Giới thiệu thí nghiệm

Đầu ra DP của MPSoc hỗ trợ hai lớp, lớp trên (Primary) hỗ trợ kênh alpha. Trong chương này, sử dụng Qt để xuất nội dung tới lớp Primary, Gstreamer sẽ bắt dữ liệu video tới lớp dưới (Overlay). Cuối cùng, hai lớp này sẽ được phần cứng trộn lại và xuất ra. Sơ đồ khối công việc như sau:



6.3 Mục tiêu thí nghiệm

① Quản lý đầu ra DRM

② Quản lý lớp DRM ③

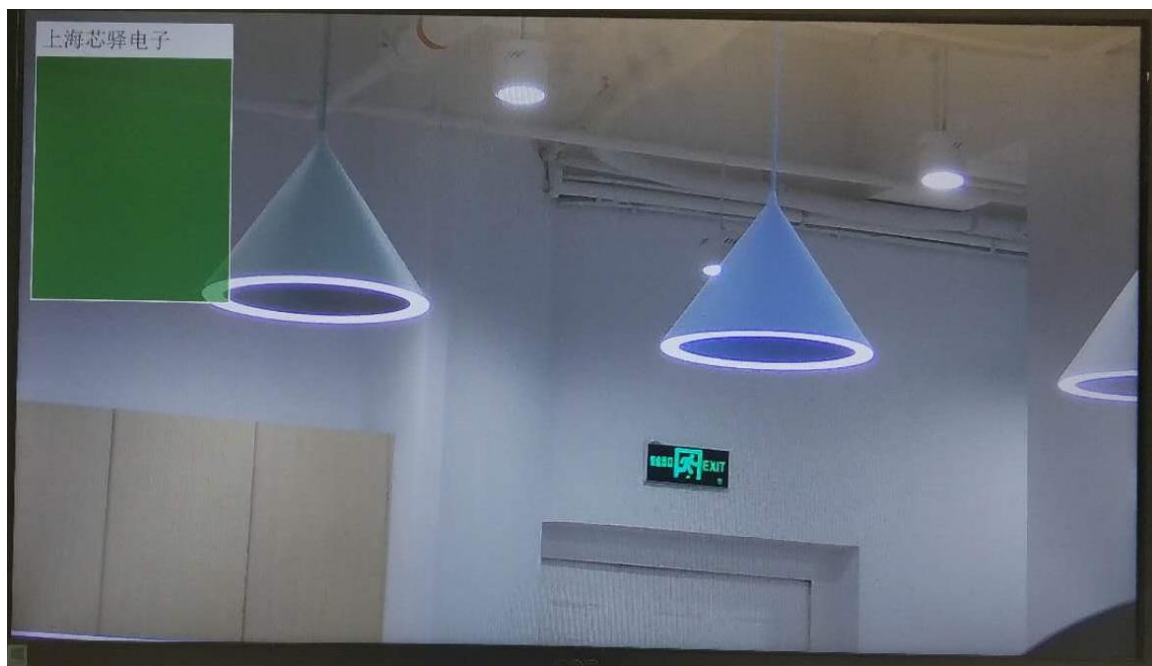
Hiển thị phân lớp Qt và GStreamer

6.4 Chạy chương trình

(1) Cần lưu ý, thí nghiệm này yêu cầu camera USB hỗ trợ độ phân giải 1080 (chỉ là luồng thô, như định dạng YUYV422 ở trên)

(2) Hiệu ứng chạy thí nghiệm

Có thể thấy trên video đã chồng lên một giao diện Qt



6.5 Phân tích mã

(1) Kiểm tra xem màn hình có được kết nối hay không

Trong hàm main, gọi `init_drm` để kiểm tra xem việc khởi tạo có thành công hay không, trả về -2, tức là không có định dạng hiển thị được hỗ trợ, cho thấy không có màn hình được kết nối.

(2) Thiết lập độ phân giải đầu ra của màn hình

Thông qua hàm `drmModeSetCrtc`, thiết lập độ phân giải và tần số làm tươi của đầu ra, ở đây không thiết lập buffer, vì vậy tham số thứ ba được đặt thành -1

(3) Thiết lập thuộc tính trong suốt của lớp đồ họa

Lớp đồ họa hỗ trợ thiết lập giá trị alpha toàn cục và đơn pixel. Ở đây sẽ thiết lập nó thành đơn pixel, tức là `g_alpha_en` được đặt thành 0

(4) Hiển thị Qt

Qt mặc định hiển thị trên lớp Primary

(5) Hiển thị GStreamer

Thông qua `drmModeGetPlaneResources` có thể lấy thông tin về plane, bao gồm cả số id của plane, ở đây đã lấy Lấy được ID của lớp Overlay là 35. Qua mã có thể thấy, thuộc tính của `kmssink` được thiết lập như sau: `kmssink sink-type=dp bus-id=fd4a0000.zynqmp-display fullscreen-overlay=false plane-id=35`

Chương bảy Hiện thị camera bằng Qt+GPU

7.1 Giới thiệu về GPU Mali

GPU viết tắt của Graphics Processing Unit, dịch sang tiếng Trung là “图形处理器”. Chúng ta thường gọi nó là card đồ họa. Có rất nhiều nhà sản xuất GPU, ba nhà sản xuất lớn nhất là Intel, NVIDIA, AMD, họ chủ yếu ứng dụng trong lĩnh vực PC.

CPU ARM chiếm 90% thị trường di động, nhưng GPU Mali của nó chỉ có thể được coi là một sự hiện diện nhỏ trong thị trường di động. GPU Mali được phát triển bởi công ty Falanx, một dự án tách ra từ Đại học Công nghệ Na Uy, và đã được ARM mua lại vào năm 2006, trở thành một bộ phận GPU của ARM, và vào năm 2007, Mali được phát hành như một phần của ARM với GPU mali-200.

Các kiến trúc của GPU Mali đã trải qua ba thế hệ như sau:

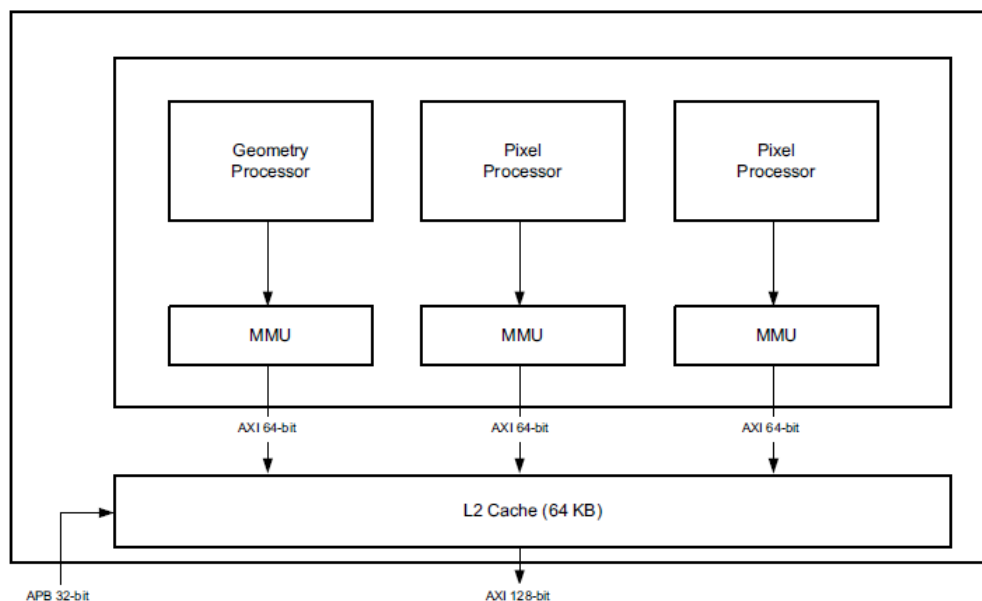
Thế hệ đầu tiên Utgard	Thế hệ thứ hai Midgard	Thế hệ thứ ba Bifrost
Mali-200/300/400/450/470	Dòng Mali-T600/T700/T800	Dòng Mali-G3/G5/G7

Phân loại theo hiệu suất:

Tiết kiệm năng lượng	Trung bình	Hiệu suất cao
Mali-400/450/470 Mali-G31	Mali-G51/G52 Mali-T820/T830 Mali-T720	Mali-G71/72 Mali-T860/T880 Mali-T760

Giới thiệu GPU MPSoc 7.2

GPU được sử dụng ở đây là một hệ thống tăng tốc 2D và 3D dựa trên phần cứng Mali™-400 MP2. Sơ đồ khối tài nguyên thành phần của nó như sau:



Các chức năng được hỗ trợ như sau:

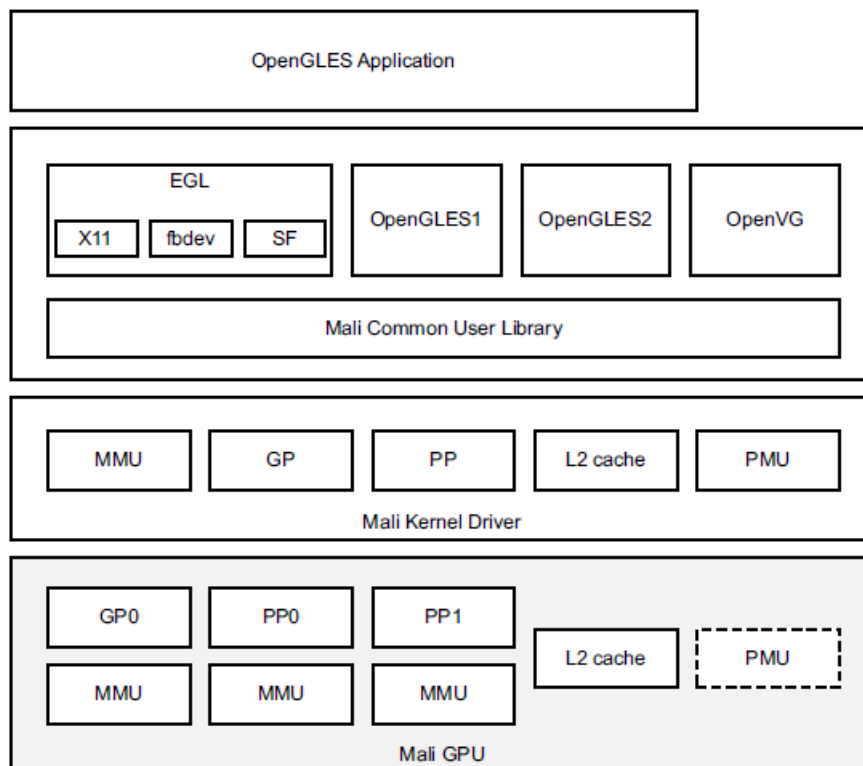
Hỗ trợ OpenGL ES 1.1 và 2.0

Hỗ trợ OpenVG 1.1

Hỗ trợ phép toán số thực 32 bit theo tiêu chuẩn

IEEE cho kích thước kết cấu lên đến 4096 x 4096 pixel.

Cấu trúc phần mềm hệ thống Linux như sau:



Tại tần số làm việc 400M, hiệu suất đỉnh mà nó có thể đạt được như sau:

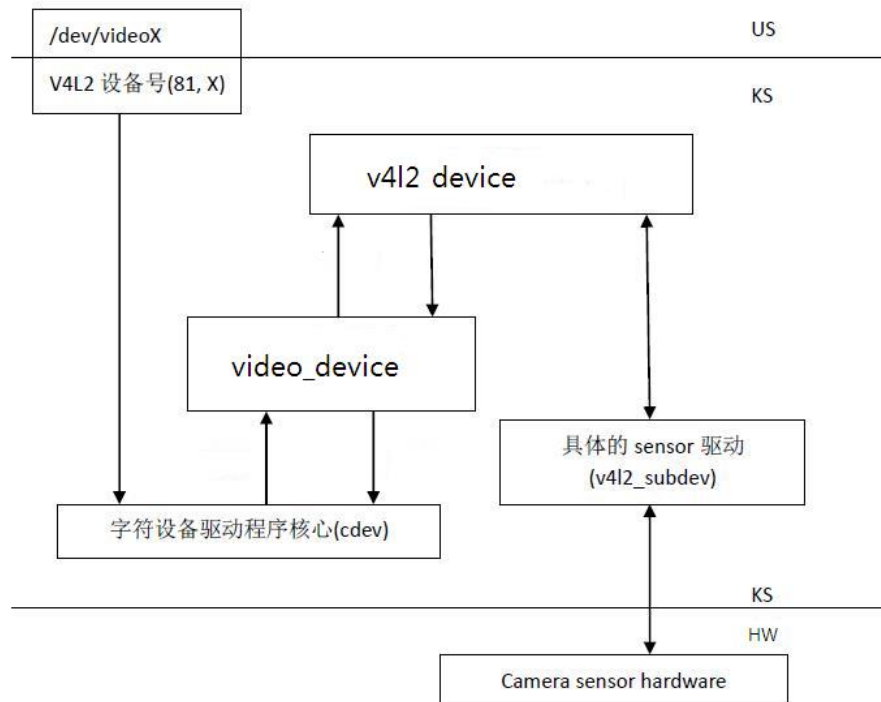
Tốc độ điền pixel: 800 Mpixel/giây,

tốc độ xử lý đỉnh: 40 Mvertex/giây

Giới thiệu V4L2 7.3

V4L2 có tên đầy đủ là Video For Linux Two, là giao diện thống nhất mà kernel cung cấp cho các ứng dụng truy cập vào các trình điều khiển âm thanh và video. Một ứng dụng cơ bản của nó là lấy dữ liệu video từ các thiết bị như camera. Tên thiết bị là /dev/video, số thiết bị chính 81, số thiết bị phụ 0-63.

Sơ đồ trong hệ thống Linux như sau:



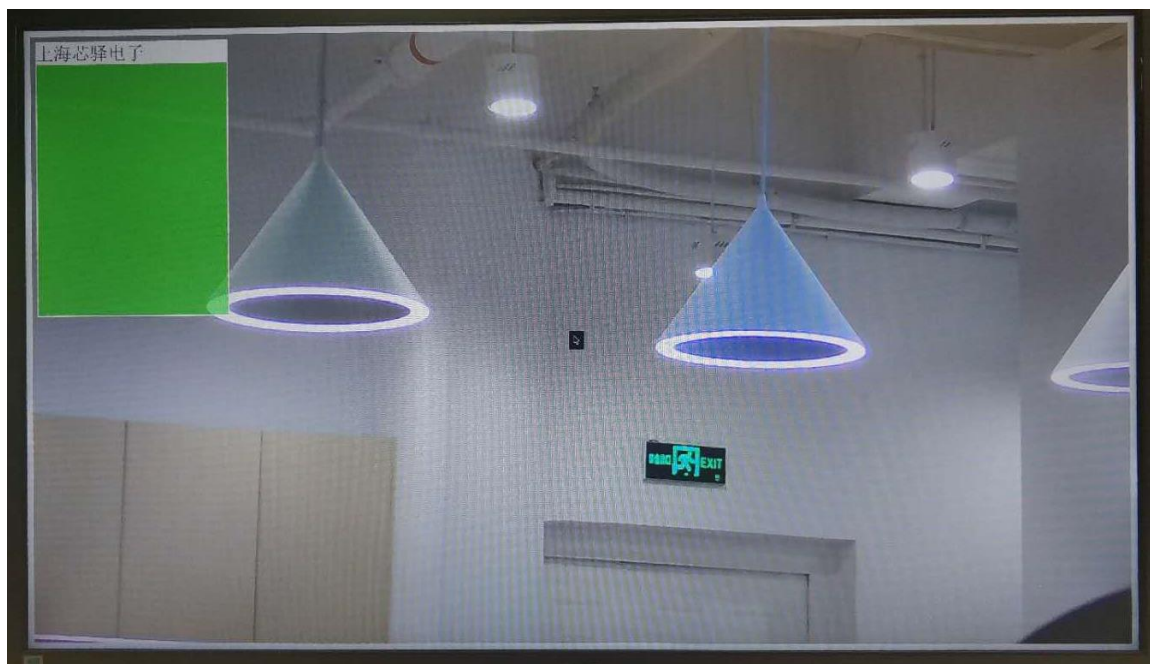
7.4 Mục tiêu thí nghiệm

① Thao tác OpenGL②

Thao tác V4L2

7.5 Chạy chương trình

Hiệu ứng chạy chương trình như sau, có thể thấy, so với hiệu ứng hiển thị ở chương trước, hình ảnh ở đây không được hiển thị toàn màn hình, vẫn để lại một vài pixel viền trắng.



7.6 Phân tích mã

Phần xử lý hiển thị chủ yếu được đóng gói trong lớp `uOpenGLYuv`, ở đây chúng tôi chỉ trình bày cách hiển thị hình ảnh định dạng YUYV422.

(1) Cách lấy một pixel trong chuỗi

fragmentStr của YUYV422:

Lấy giá trị Y từ kết cấu Y $yuv.x = texture2D(tex_y, textureOut).x$, lấy giá trị U từ kết cấu UV $yuv.y = texture2D(tex_uv, textureOut).y$, lấy giá trị V từ kết cấu UV $yuv.z = texture2D(tex_uv, textureOut).w$. Các phép toán khác là chuyển đổi yuv thành rgb.

(2) Cập nhật hình ảnh

Trong `MainWindow`, sau khi nhận được một khung hình, gọi `pOpenGLYuv->update()`, sau đó `uOpenGLYuv` gọi `paintGL`, tải hình ảnh dưới dạng kết cấu và hiển thị.

(3) Bắt hình từ camera

Việc bắt hình từ camera được thực hiện thông qua việc gọi giao diện `v4l2`.

Chương tám: thao tác thanh ghi trên Linux

8.1 ánh xạ bộ nhớ linux

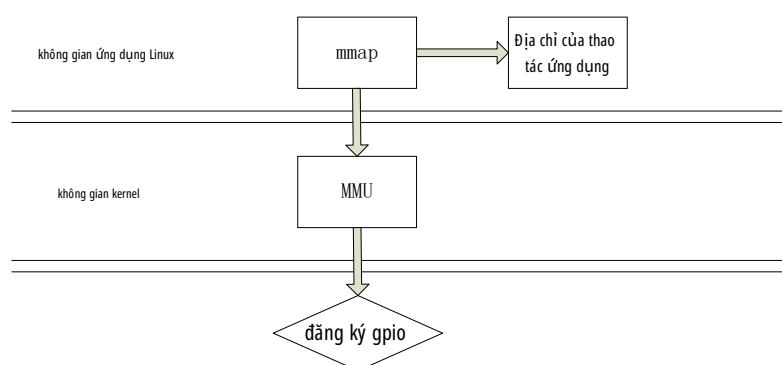
Bản đồ bộ nhớ, tức là ánh xạ một vùng bộ nhớ trong không gian người dùng vào không gian nhân, sau khi ánh xạ thành công, những thay đổi của người dùng đối với vùng bộ nhớ này có thể phản ánh trực tiếp vào không gian nhân, tương tự, những thay đổi của không gian nhân đối với vùng này cũng phản ánh trực tiếp vào không gian người dùng. Điều này rất hiệu quả cho việc truyền tải một lượng lớn dữ liệu giữa không gian nhân và không gian người dùng.

Ví dụ, khi đọc một tệp từ ổ cứng vào bộ nhớ, phải qua hệ thống tệp để thực hiện sao chép dữ liệu, và thao tác sao chép dữ liệu được thực hiện bởi hệ thống tệp và trình điều khiển phần cứng, lý thuyết mà nói, hiệu suất sao chép dữ liệu là như nhau. Tuy nhiên, việc truy cập tệp trên ổ cứng bằng phương pháp ánh xạ bộ nhớ có hiệu suất cao hơn so với các lệnh gọi hệ thống read và write, tại sao lại như vậy? Nguyên nhân là read() là một lệnh gọi hệ thống, trong đó có thực hiện sao chép dữ liệu, nó đầu tiên sao chép nội dung tệp từ ổ cứng vào một bộ đệm trong không gian nhân, sau đó lại sao chép những dữ liệu này vào không gian người dùng, trong quá trình này, thực tế đã hoàn thành hai lần sao chép dữ liệu; trong khi mmap() cũng là một lệnh gọi hệ thống, nhưng trong mmap() không có thực hiện sao chép dữ liệu, việc sao chép dữ liệu thực sự diễn ra trong quá trình xử lý ngắt thiếu trang, do mmap() ánh xạ trực tiếp tệp vào không gian người dùng, vì vậy hàm xử lý ngắt dựa trên mối quan hệ ánh xạ này, trực tiếp sao chép tệp từ ổ cứng vào không gian người dùng, chỉ thực hiện một lần sao chép dữ liệu. Do đó, hiệu suất của ánh xạ bộ nhớ cao hơn so với read/write.

8.2 Giới thiệu thí nghiệm

Đôi khi, một chức năng của thiết bị ngoại vi đã được xác minh thành công trong môi trường không có hệ điều hành, nếu thiết bị ngoại vi đó không sử dụng ngắt, chúng ta có thể thông qua việc ánh xạ thanh ghi trong linux để chuyển trực tiếp chương trình từ môi trường không có hệ điều hành sang sử dụng trong linux, như vậy, có thể tiết kiệm được việc phát triển driver.

Ví dụ này đã chuyển đổi ví dụ gpio trong vitis, điều khiển ps_led, thực hiện hành động nhấp nháy một lần mỗi giây. Các thao tác khác, quy trình cũng tương tự.



8.3 Chạy chương trình

Chạy ứng dụng register_opt, bạn sẽ thấy đèn led của PS nhấp nháy một lần mỗi giây

8.4 Phân tích mã

- (1) Về địa chỉ thanh ghi của gpio, ở đây đều được sao chép từ ví dụ gpio của vitisi, sau này tư duy phát triển của chúng ta cũng nên như vậy, trước tiên sử dụng chương trình không có hệ điều hành của vitisi để xác minh. Nhiều thiết bị ngoại vi và ip ở phía fpga, vitisi sẽ giúp chúng ta tạo ra các phương thức và địa chỉ thao tác, như vậy chúng ta không cần phải tìm mối quan hệ tương ứng.
- (2) Khi mở /dev/mem, sử dụng tùy chọn O_SYNC
Sau khi ghi dữ liệu ra ngoài, thường dữ liệu sẽ được ghi vào bộ đệm cache. O_SYNC sẽ đảm bảo rằng dữ liệu được ghi vào thiết bị ngoại vi trước khi trả về. Cần lưu ý rằng, O_SYNC ở đây chỉ ảnh hưởng đến thao tác ghi.
- (3) Gọi msync
Nếu cần ghi một lượng lớn dữ liệu vào thiết bị ngoại vi, việc gọi O_SYNC sẽ ảnh hưởng nghiêm trọng đến hiệu suất hệ thống. Trong trường hợp này, nếu chúng ta không sử dụng O_SYNC mà thay vào đó gọi msync sau khi đã ghi xong dữ liệu, điều này sẽ cải thiện hiệu suất ghi.
- (4) Vấn đề nhất quán trong thao tác đọc
Nếu cần đọc dữ liệu từ thiết bị ngoại vi, do sự tồn tại của bộ nhớ đệm, dữ liệu mà ứng dụng nhận được là dữ liệu trong bộ nhớ đệm, chứ không phải trạng thái mới nhất của thiết bị ngoại vi. Do đó, giá trị đọc được có thể là một giá trị sai.