# 第一篇：调试串口TTYMSM0的使用

试用410c，苦于手边没有HDMI显示器，查看了一些资料了解到在板子的低速接口配备了4线的Uart0和2线的Uart1，由于输出debug信息是通过Uart1，本篇只论Uart1。

Uart1接口如下：



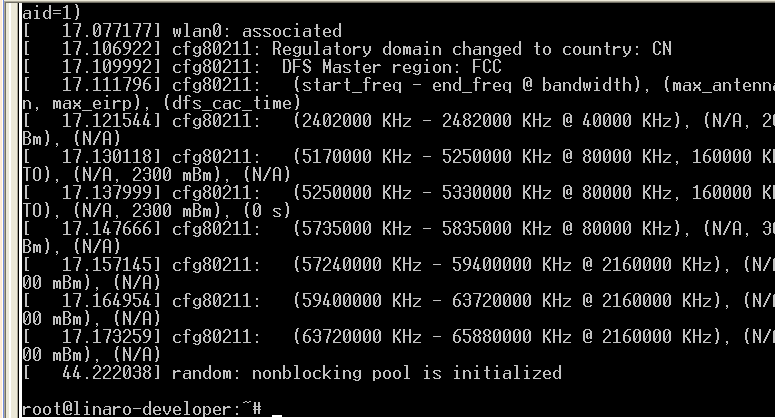
注意：



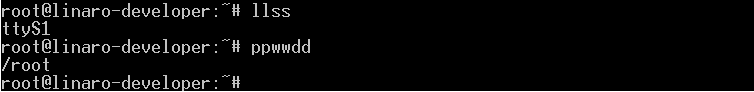
文档对410c的UART1电平指出是1.8V-TTL，需要使用1.8V的USB转TTL线才能正常输出调试信息。

按照上面接入了，如果使用Windows，需要安装PL2303驱动（一般1.8V线都是2303的）。

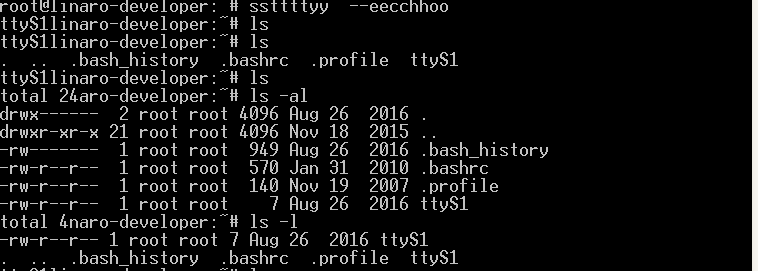
万事具备了，只需要开机、打开串口类软件（如XP的超级终端、PUTTY、XSHELL5、Minicom等）；



启动完毕，默认这个是开回显的，如下：



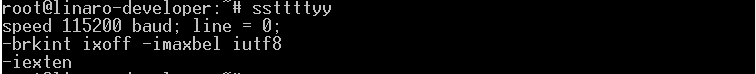
而关闭回显需要用到这个命令stty，先介绍它怎么关回显，使用stty –echo



关了回显其实效果也没多好，不带参数的命令也就不显示结果，不如用ssh，此处用串口是为了配置wifi，以便使用ssh或vnc，此处不表。

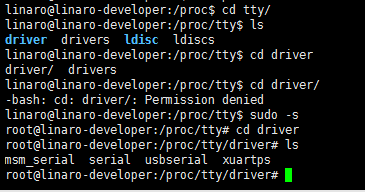
接下介绍一下stty的一些用途：

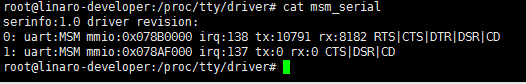
stty:



它提供了终端波特率信息等。还有stty size显示控制台大小和stty echo关闭回显，此处不表。

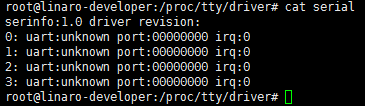
接下来我们看看高通对于Uart0，Uart1的节点设置，先进入/proc/tty/driver/目录下可以看到有几个文件



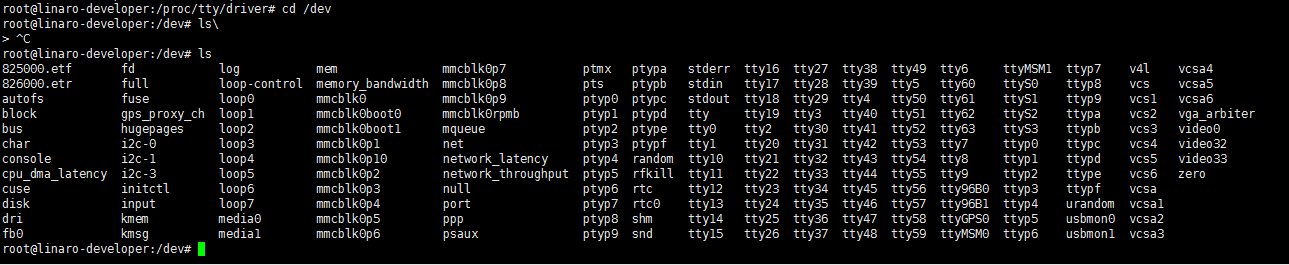
重点看第一个和第二个，第一个msm\_serial，我们用cat查看一下其内容：

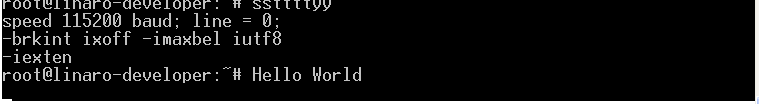
可以看到列出来了410c板载的2个UART信息，不过这两个编号与手册上并不一样，在下文利用shell发送数据就可以看出来。

我们在看看第二个，serial



可以看出来并没有对应什么实际串口，是linux的保留串口；接下来我们进入/dev/看一下



可以看到ttyMSM0，我们输入echo “Hello World” > /dev/ttyMSM0，再去串口端看看有什么输出

可以看到串口端成功接收并打印出来了信息，手册上指明Uart1，而此处却是MSM0，编号错误也被验证了。

# 第二篇：SSH连接

要想使用ssh，则必须要有网络接入，且在同一子网有一PC，410c自带wifi，在终端下有NetworkManager的两种命令行形式，纯命令式nmcli和文本GUI式nmtui，先介绍nmcli吧。

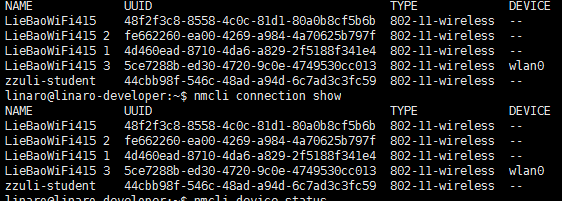
使用nmcli连接网络使用如下命令形式

nmcli dev wifi connect (SSID) password (password)

查看是否连接上了可以使用

nmcli connection show

可以看到连接的那一行最后有wlan0标注。

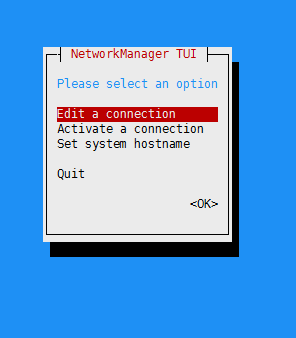


如果想了解更详细的连接信息，使用

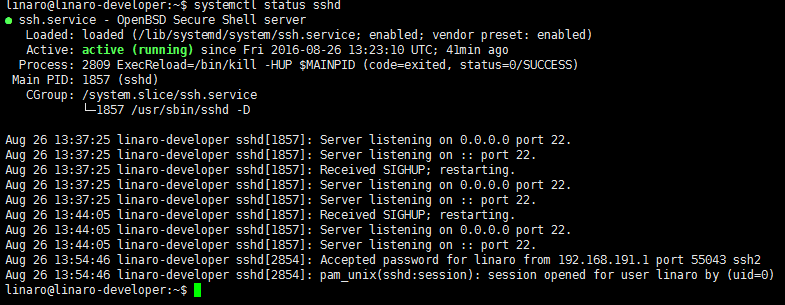
nmcli connection show (SSID)

使用nmcli device status可以查看配置详情。

接下来我们来介绍nmtui，nmtui要简单的多，直接输入命令nmtui



此处不是重点，接下来就开始真正主题，进行ssh连接，首先使用systemctl status sshd

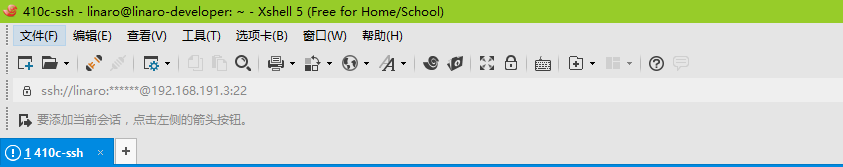


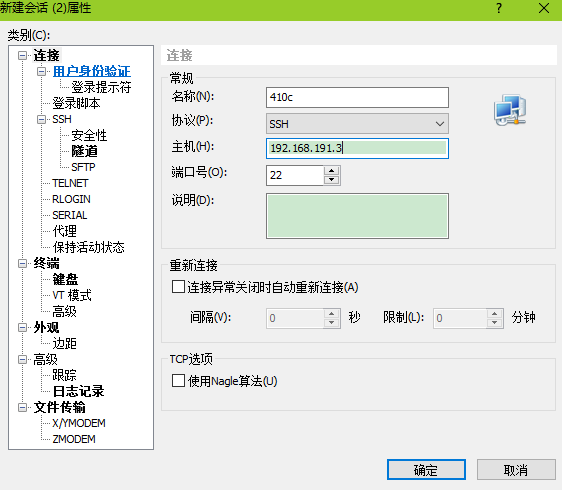
可以看到ssh服务是激活的

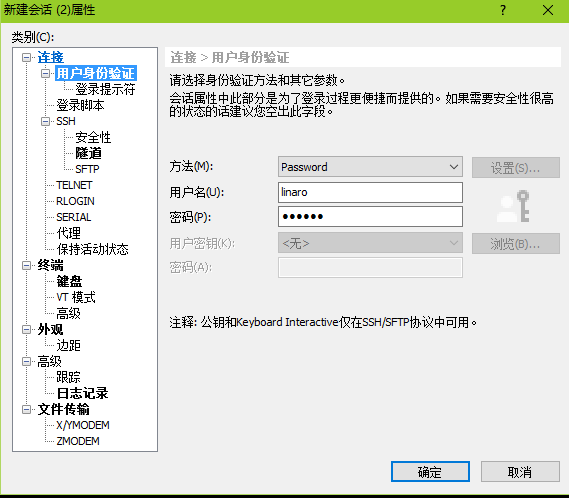
如果是这样，则需要安装ssh包，具体怎么安装，可以参考网上debian或ubuntu安装ssh的方法。

接着可以使用ssh连接，首先得获取到410c的IP

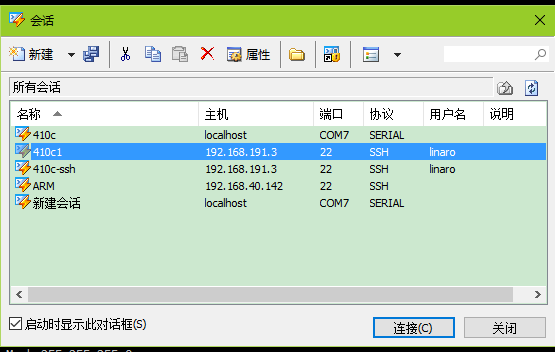


可以看到IP为192.168.191.3，接着打开处于同一局域网的PC，使用XShell5(其他都大致类似) 

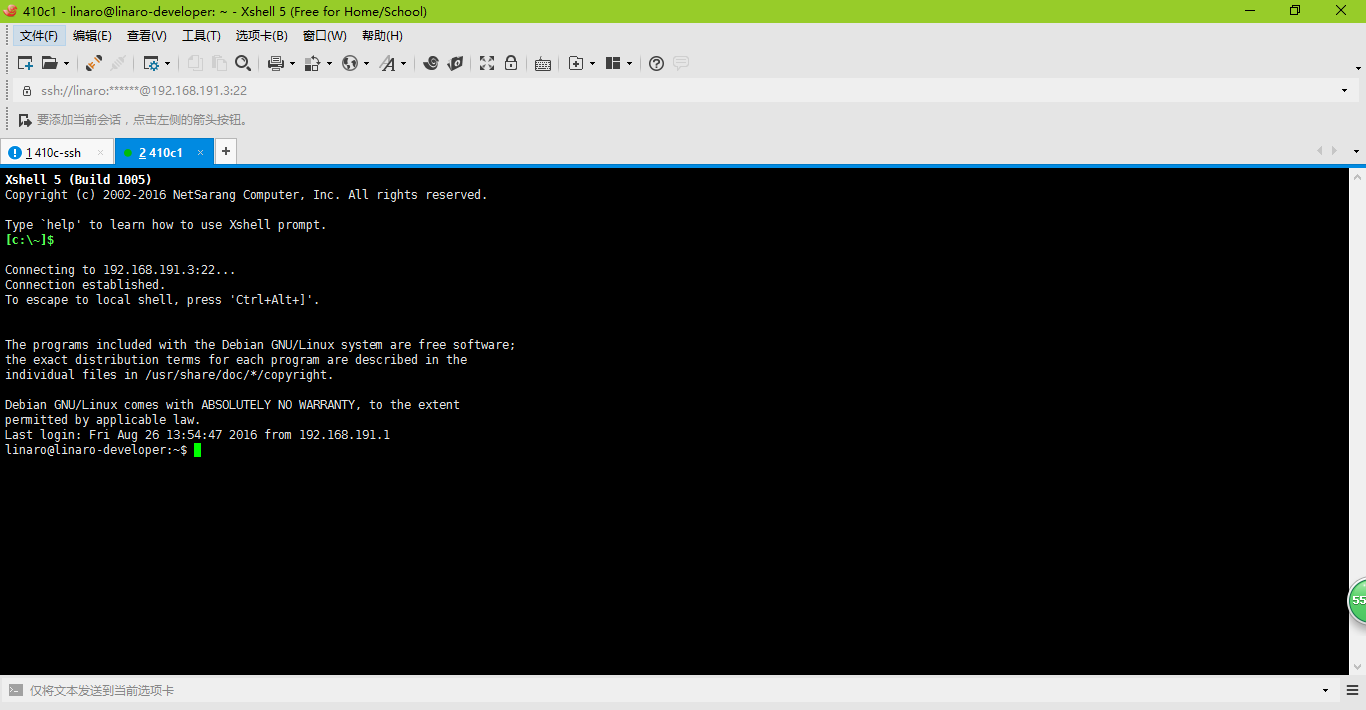
初次使用一般就点左端那个加号图标

按照自己实际配置修改，点击用户身份验证，用户名和密码都是linaro 

确定后就如下

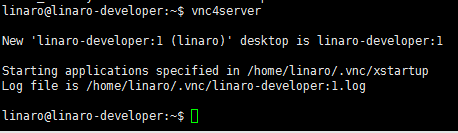


点击连接，如下就表示连接成功，就可以体验ssh的好处。



使用Vnc，要想显示桌面，则必须开启远程类桌面服务，如vnc此处简要说一下vnc，需要使用apt install vnc4server，其他的也可以，例如tigervnc，此处不表。

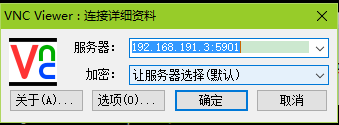
在ssh下输入vnc



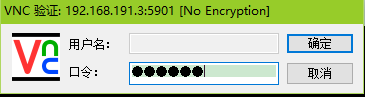
就表示vnc服务已经开启，设置vnc密码使用vnc4passwd:



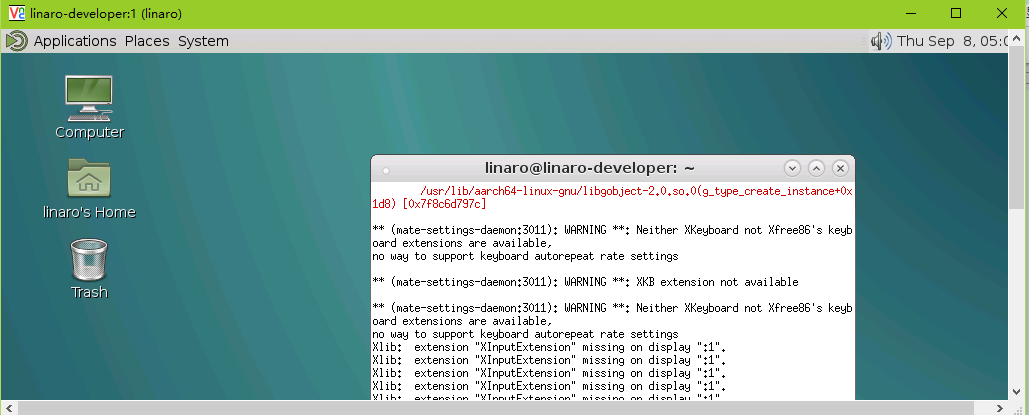
接在PC上运行VNC Viewer软件：



点确定



再确定



即可体验桌面之旅

# 第三篇：使用镜像源装OpenCV

之前使用vnc连接时，桌面显示的或是X-Window-Manager或是运行在图形化的Terminal，那么，若想使用已装好的Mate桌面，该怎么做呢？

可以这么做，直接在图形界面终端运行桌面的session；然而还有一种更好的办法，使用终端编辑home目录下的.vnc/xstartup，使用如下命令

vi ~/.vnc/xstartup

注释掉最后两行，添加mate-session &，文件内容修改后如下：

#!/bin/sh

# Uncomment the following two lines for normal desktop:

# unset SESSION\_MANAGER

# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup

[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources

xsetroot -solid grey

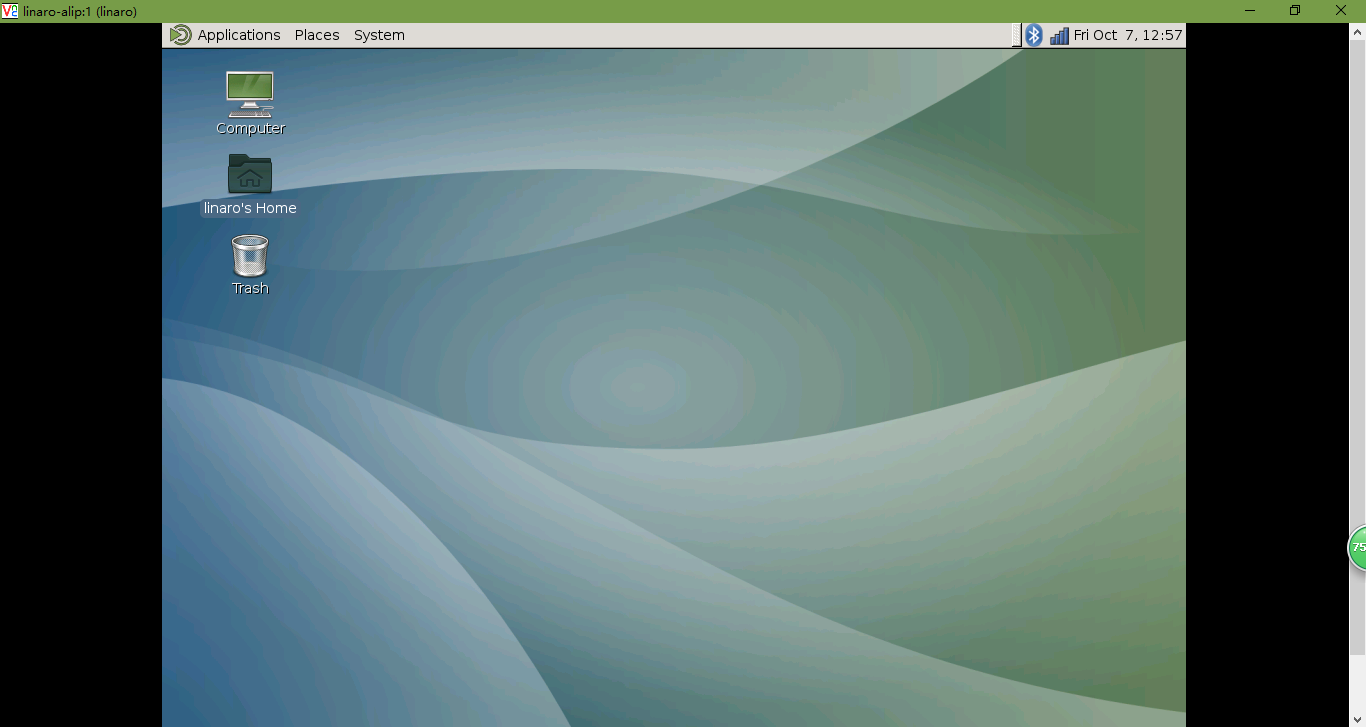
vncconfig -iconic &

#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &

#x-window-manager &

mate-session &

按照之前连接vnc，桌面显示如下：



接下来说说正题，安装opencv，有多种办法，但整体分为两大类，从镜像源里安装和自编译，自编译安装通用性好，尤其对于有一些生僻的架构，但，从镜像源里安装，是由一些人员编译完了，分类打包，好处是不需要编译就可以，与操作系统协调性好，但是缺点也比较多，例如，最新版往往不能及时提供，卸载麻烦等，总之有利有弊。

高通410c提供的alip版本内部已经提供了一些用于opencv程序运行的so库，但是为了开发，我们需要重新从源里索引一下，使用如下命令：

sudo apt update

源里提供的libopencv-dev，提供了opencv.pc，这是务必要装的，便于使用pkg-config；我从源里安装的opencv文件如下：

linaro@linaro-alip:~$ dpkg -l|grep opencv

ii libopencv-calib3d-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-calib3d

ii libopencv-calib3d2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Camera Calibration library

ii libopencv-contrib-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-contrib

ii libopencv-contrib2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision contrib library

ii libopencv-core-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-core

ii libopencv-core2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision core library

ii libopencv-dev 2.4.9.1+dfsg-1+deb8u1 arm64 development files for opencv

ii libopencv-features2d-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-features2d

ii libopencv-features2d2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Feature Detection and Descriptor Extraction library

ii libopencv-flann-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-flann

ii libopencv-flann2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Clustering and Search in Multi-Dimensional spaces library

ii libopencv-gpu-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-gpu2.4

ii libopencv-gpu2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision GPU library

ii libopencv-highgui-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-highgui

ii libopencv-highgui2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision High-level GUI and Media I/O library

ii libopencv-imgproc-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-imgproc

ii libopencv-imgproc2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Image Processing library

ii libopencv-legacy-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-legacy

ii libopencv-legacy2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision legacy library

ii libopencv-ml-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-ml

ii libopencv-ml2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Machine Learning library

ii libopencv-objdetect-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-objdetect

ii libopencv-objdetect2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Object Detection library

ii libopencv-ocl-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-ocl2.4

ii libopencv-ocl2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision OpenCL support library

ii libopencv-photo-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-photo2.4

ii libopencv-photo2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision computational photography library

ii libopencv-stitching-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-stitching2.4

ii libopencv-stitching2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision image stitching library

ii libopencv-superres-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-superres2.4

ii libopencv-superres2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Super Resolution library

ii libopencv-ts-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-ts2.4

ii libopencv-ts2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision ts library

ii libopencv-video-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-video

ii libopencv-video2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision Video analysis library

ii libopencv-videostab-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 development files for libopencv-videostab2.4

ii libopencv-videostab2.4:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 computer vision video stabilization library

ii libopencv2.4-java 2.4.9.1+dfsg-1+deb8u1 all Java bindings for the computer vision library

ii libopencv2.4-jni 2.4.9.1+dfsg-1+deb8u1 arm64 Java jni library for the computer vision library

ii opencv-data 2.4.9.1+dfsg-1+deb8u1 all development data for opencv

ii opencv-doc 2.4.9.1+dfsg-1+deb8u1 all OpenCV documentation and examples

linaro@linaro-alip:~$ dpkg -l|grep libcv

ii libcv-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 Translation package for libcv-dev

ii libcv2.4 2.4.9.1+dfsg-1+deb8u1 all computer vision library - libcv\* translation package

ii libcvaux-dev:arm64 2.4.9.1+dfsg-1+deb8u1 arm64 Translation package for libcvaux-dev

安装好了，我们再详细介绍pkg-config，此命令会查找环境变量PKG\_CONFIG\_PATH对应目录和/usr/lib/pkgconfig/下的.pc文件，其有三种参数，分别是list-all，cflags，libs

此下示例用法：

linaro@linaro-alip:~$ pkg-config --list-all |grep opencv

opencv OpenCV - Open Source Computer Vision Library

linaro@linaro-alip:~$ pkg-config --cflags opencv

-I/usr/include/opencv

linaro@linaro-alip:~$ pkg-config --libs opencv

/usr/lib/aarch64-linux-gnu/libopencv\_calib3d.so -lopencv\_calib3d /usr/lib/aarch64-linux-gnu/libopencv\_contrib.so -lopencv\_contrib /usr/lib/aarch64-linux-gnu/libopencv\_core.so -lopencv\_core /usr/lib/aarch64-linux-gnu/libopencv\_features2d.so -lopencv\_features2d /usr/lib/aarch64-linux-gnu/libopencv\_flann.so -lopencv\_flann /usr/lib/aarch64-linux-gnu/libopencv\_gpu.so -lopencv\_gpu /usr/lib/aarch64-linux-gnu/libopencv\_highgui.so -lopencv\_highgui /usr/lib/aarch64-linux-gnu/libopencv\_imgproc.so -lopencv\_imgproc /usr/lib/aarch64-linux-gnu/libopencv\_legacy.so -lopencv\_legacy /usr/lib/aarch64-linux-gnu/libopencv\_ml.so -lopencv\_ml /usr/lib/aarch64-linux-gnu/libopencv\_objdetect.so -lopencv\_objdetect /usr/lib/aarch64-linux-gnu/libopencv\_ocl.so -lopencv\_ocl /usr/lib/aarch64-linux-gnu/libopencv\_photo.so -lopencv\_photo /usr/lib/aarch64-linux-gnu/libopencv\_stitching.so -lopencv\_stitching /usr/lib/aarch64-linux-gnu/libopencv\_superres.so -lopencv\_superres /usr/lib/aarch64-linux-gnu/libopencv\_ts.so -lopencv\_ts /usr/lib/aarch64-linux-gnu/libopencv\_video.so -lopencv\_video /usr/lib/aarch64-linux-gnu/libopencv\_videostab.so -lopencv\_videostab

此刻我们使用一个显示简单百度Logo的Demo，代码如下：

#include <stdio.h>

#include <cv.h>

#include <highgui.h>

int main(void)

{

IplImage \*img = cvLoadImage("baidu.png", 1);

cvNamedWindow("Example", CV\_WINDOW\_AUTOSIZE);

cvShowImage("Example", img);

cvWaitKey(0);

cvReleaseImage(&img);

cvDestroyWindow("Example");

return 0;

}

命令行gcc编译有两种办法，一种使用-I、-l等指定共享库和头文件；另一种则是使用pkg-config，使用如下：

linaro@linaro-alip:~$ gcc -o main main.c `pkg-config --cflags --libs opencv`

/usr/bin/ld: /tmp/ccbQ85Ps.o: undefined reference to symbol 'lrint@@GLIBC\_2.17'

//lib/aarch64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command line

collect2: error: ld returned 1 exit status

会发现报错，此错是由于没添加math库，使用如下命令就可以：

linaro@linaro-alip:~$ gcc -o main main.c `pkg-config --cflags --libs opencv` -lm

linaro@linaro-alip:~$

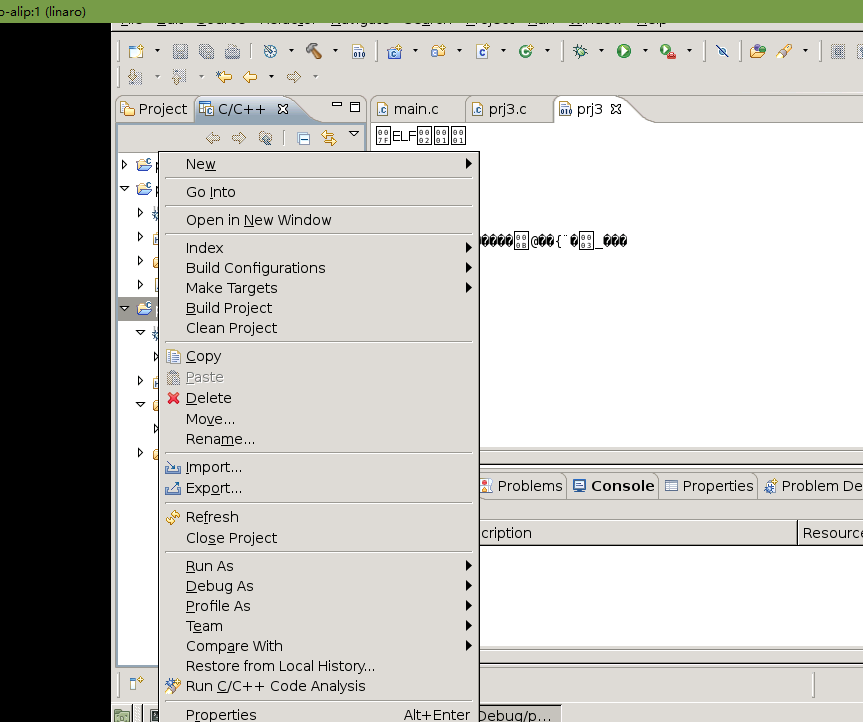
我们切换到vnc看看效果：

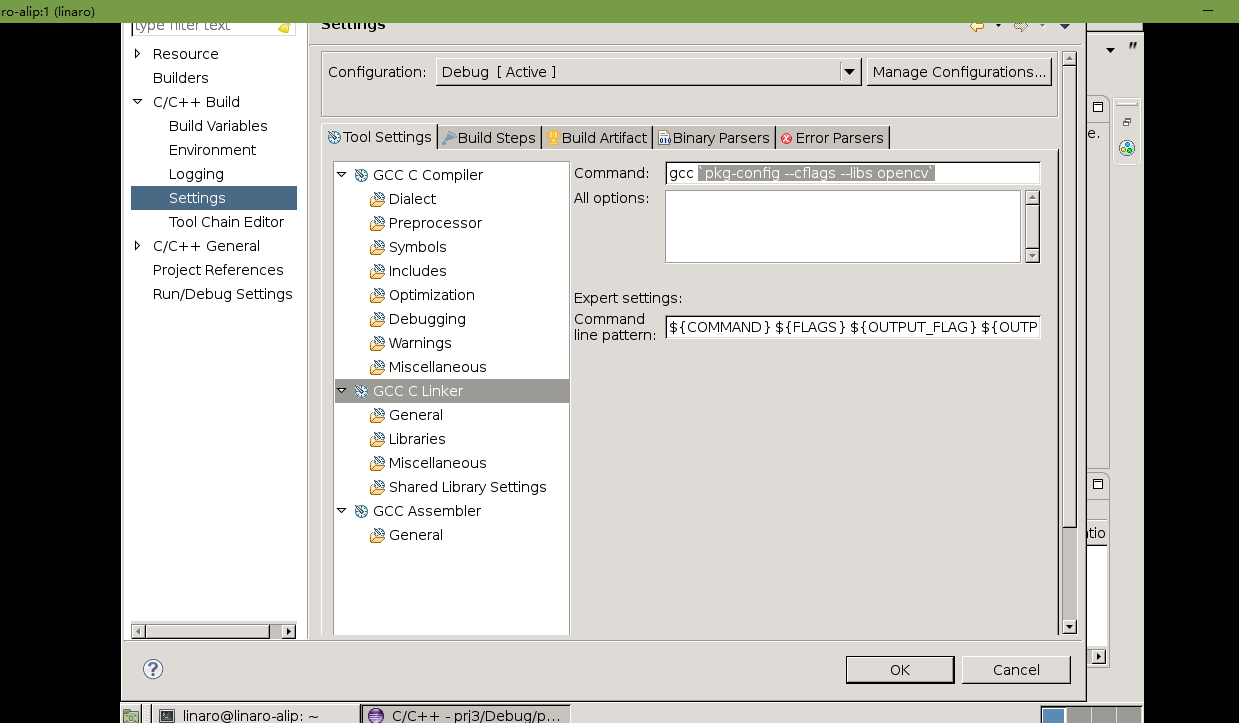


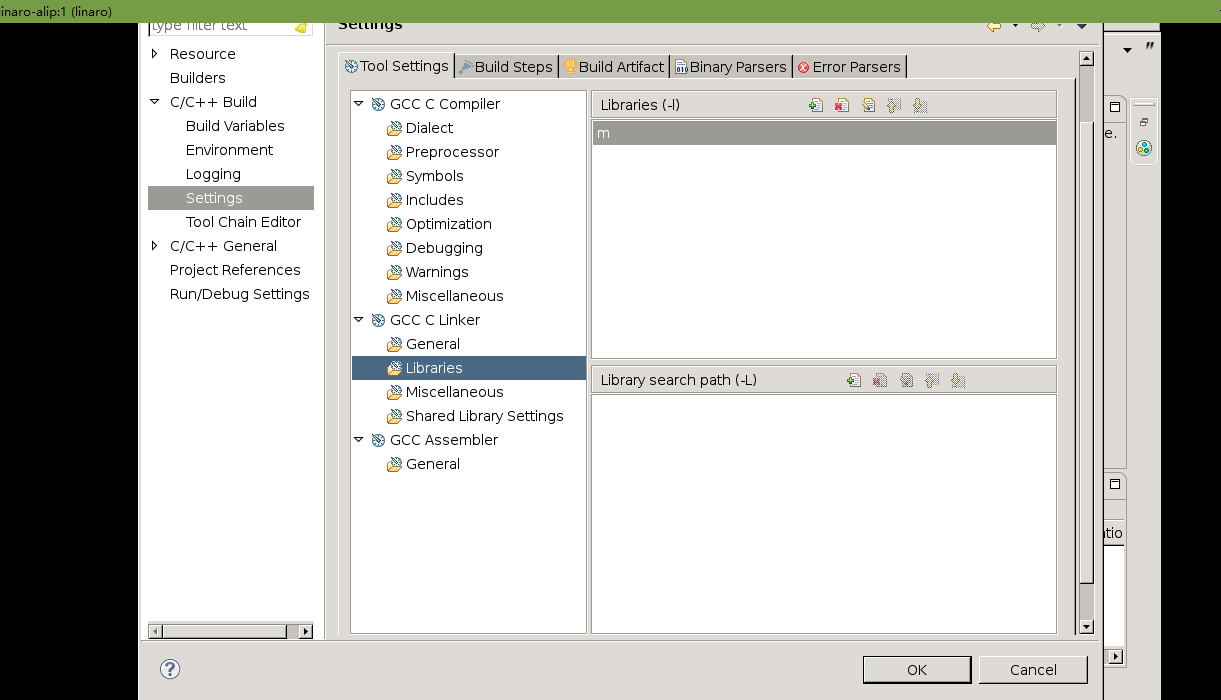
接下来我们使用eclipse，要使用eclipse，也有两种，源里和IBM提供源码编译，由于IBM只提供x86和x64的二进制， 使用如下命令就可以：

sudo apt install eclipse-cdt

就可以了，接着在eclipse使用opencv，要想使用，则需要配置相关的参数，选择项目的属性，不是Eclipse里的Window下的属性，选择C/C++ Build->Settings，直接按下图配置就行：







编译前需要保存，否则会报错。

# 项目结项：数字相册

## 综述

本次试用的项目实现了读取home分区下所有图片并按一定顺序将图片显示，切换图片间实现了左右切换、淡入淡出和百叶窗，通过2个按钮选择切换到上一张还是下一张，还对当前图像进行处理，实现增强对比度、亮度改变以及腐蚀膨胀，效果如下所示：



源码地址：https://github.com/agdhun/DigitalPhotoFrame

## 第一部分：路径读取与图片显示

我们通过QT与OpenCV结合，OpenCV实现图片切换和处理效果，而QT则负责读取文件路径和显示图片、控制程序启动与退出功能，如下图所示：

数字相册

OPENCV

图片切换效果处理

图片对比度、亮度等效果处理

QT

读取路径

显示图片

按键响应

对于路径读取，将目录转QDir类的一个实例，再定义QStringList的实例，通过迭代器把图片路径保存起来，实现代码如下：

QDir dir(path);

if(!dir.exists())

{

return;

}

QStringList filters;

filters<<QString("\*.jpeg")<<QString("\*.jpg")<<QString("\*.png")<<QString("\*.tiff")<<QString("\*.gif")<<QString("\*.bmp");

QDirIterator dir\_iterator(path,

filters,

QDir::Files | QDir::NoSymLinks,

QDirIterator::Subdirectories);

while(dir\_iterator.hasNext())

{

dir\_iterator.next();

QFileInfo file\_info = dir\_iterator.fileInfo();

QString file\_path = file\_info.absoluteFilePath();

string\_list.append(file\_path);

qDebug() << file\_path;

}

qDebug函数是打印输出信息，有如ARM的串口，Linux的Terminal等；

获得路径后我们就要把第一幅图片显示在QT的label上，我们需要先利用opencv的读取图片函数来将图片的信息读取保存到Mat结构中，虽然OpenCV也提供对图片显示的支持，但OpenCV对于界面也仅仅提供Window、Slider等基本功能，操作并不是很方便，而QT则是为显示而生，把OpenCV的图片数据显示到QT上，首先需要考虑OpenCV的数据格式与QT并不一样，OpenCV是以BGR排列，而QT是用GRB排列，需要用到OpenCV的图像数据转换函数，其原型如下：

CV\_EXPORTS\_W void cvtColor( InputArray src, OutputArray dst, int code, int dstCn=0 );

其需要提供至少三个参数，分别是：

Src：原图片数据；

Dst：目标图片数据；

Code：转换形式；

对于code，OpenCv提供了135种转换，常用于QT显示也就是CV\_BGR2GRB;

得到转换后的数据后，我们就需要将图片显示在QT程序上，使用QImage进行数据转换，再将其显示到QT的label控件上，其中需要注意的是，对于图像数据要考虑到其通道数，否则会出现图片扭曲现象，使用如下代码便可消除：

if(disp.channels() == 3)

{

cvtColor(disp, rgb, CV\_BGR2RGB);

img = QImage((const unsigned char\*)(rgb.data), rgb.cols, rgb.rows, rgb.cols\*rgb.channels(), QImage::Format\_RGB888);

}else

{

rgb = disp;

img = QImage((const unsigned char\*)(rgb.data), rgb.cols, rgb.rows, rgb.cols\*rgb.channels(), QImage::Format\_Indexed8);

}

接着通过label控件的setPixmap方法将图片显示。

## 第二部分：QT控件使用

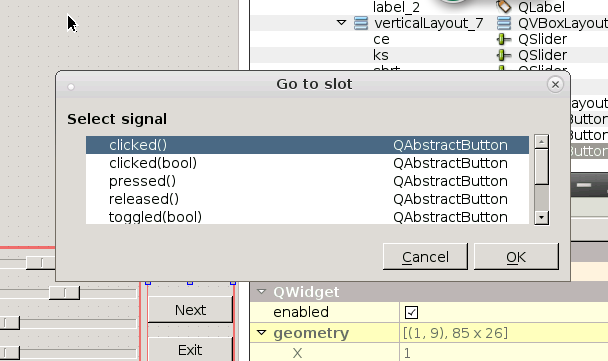
在QTcreator里可以使用设计师来设计ui文件，最后用c文件导入就可以，优点是简化了代码编写，所见即所得，当然也有缺点，多Ui间不能很好处理，与代码交互性并不是很好，控件方位修改不太灵活。

设计师提供了多种按钮，有单选按钮RadioButton、普通按钮PushButton等，如下所示：

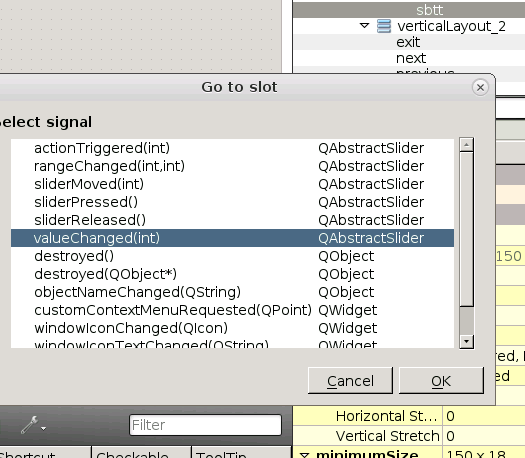


此例中，我们只需要用到Push Button和Radio Button，Push Button，即按下就会有信号，即clicked()信号，当然需要有相应的槽来接收，其实也可以不用槽信号接收，直接对此按钮进行信号重载，使用右键，Go to slot…，即可在相应的C文件产生私有的槽，此槽有缺点就是仅仅只用于该信号，这样就无需要在填充connect代码：





其他类似，我们还需要用到滑动条slider，同理可以用如上方法填充槽，好处是不用记住相应控件的系统槽函数，它们给你提供，缺点也有，当信号不是系统提供的，需要自己添加；



对于slider，使用如上方法，signal需要选择valueChanged(int)，而不能使用sliderMoved(int)或sliderPressed()或sliderReleased()，因为它们只能响应slider滑动或是点击的其中一种行为，并不能很好处理只要滑块移动就处理，当然也可以通过属性栏控制不能滑动或是单击，这样只需要处理其中一种情况，其他的用户操作无法实现滑块滑动。Label控件使用很简单，拖入即可，控制好范围即可。

## 第三部分：图片切换效果

此处我们使用OpenCV处理图片，从读取图片后，我们需要对图片进行裁剪，方可让图片在label上大小完美的显示，我们使用cv::resize函数，原型如下：

CV\_EXPORTS\_W void resize( InputArray src, OutputArray dst,

Size dsize, double fx=0, double fy=0,

int interpolation=INTER\_LINEAR );

提供输入图像的矩阵和输出图像矩阵变量，再提供输出矩阵大小就可以了，无特效处理很简单，一旦出现转换到下一页，我们直接让程序控制label显示第二幅图片，由于速度极快，相对于肉眼，就像连续过去；对于从左到右，我们需要让图片依次平移，当前图片像左平移，新图像从0列开始依次向左展开，并稍加延时，让肉眼能感知这种变化，也即成了效果：

同理，淡入淡出也是类似实现，我们使用cv::addWeighted函数，其原型如下：

CV\_EXPORTS\_W void addWeighted(InputArray src1, double alpha, InputArray src2,

double beta, double gamma, OutputArray dst, int dtype=-1);

也就是实现dst = alpha\*src1 + beta\*src2 + gamma这个公式，此处不详细探讨；

最后一个效果，百叶窗，大致思路是，将一副整图分割成n份，每一份都单独实现从左到右的效果。

## 第四部分：图片处理效果

对于图片处理效果，我实现了对比度调整、亮度调整、腐蚀膨胀调整，使用四个滑动条控制其中相应数据改变的触发，对比度调整使用之前的addWeighted函数，对比度也就是对每个像素同时放大或缩小N倍，而亮度则是加上在-255到+255范围的数值，腐蚀膨胀则是用erode和dilate函数。