嵌入式平台 QR 码译码程序的移植方法①

杨柏松, 高美凤

(江南大学 物联网工程学院, 无锡 214122)

摘 要: 给出一种嵌入式平台QR码译码程序的移植方法. 选用UP-NETARM2410-S嵌入式平台作为硬件开发平台, 首先介绍了系统的硬件平台的组成, 然后, 对使用 Qt-Creator 进行 QR 码译码程序的开发的具体流程进行详细的介绍, 测试阶段, 利用 qvfb 虚拟屏, 对程序在开发平台的运行情况进行模拟. 最后, 将程序向嵌入式平台移植. 测试结果表明, 译码程序能够在嵌入式平台正常运行, 并能进行QR码的译码. 所提出的移植方法, 对于不同平台QR码译码程序移植有一定的借鉴意义.

关键词: UP-NETARM2410-S 嵌入式平台; QR 码译码程序; Qt-Creator; qvfb 虚拟屏; 移植

Transplantation Method of QR Code Decoding Program Based on Embedded Platforms

YANG Bai-Song, GAO Mei-Feng

(School of IoT Engineering, Jiangnan University, Wuxi 214122, China)

Abstract: A transplantation method of QR code decoding program based on embedded platform was presented. The UP-NETARM2410-S was selected as the hardware development platform. Firstly, the system hardware composition was given. Then, the whole development process of QR code decoding program using Qt-Creator was introduced in detail. In the test phase, in order to simulate the running state of QR code decoding, the qvfb visual screen was used. Finally, the program was transplanted to true embedded platform. Test results showed that the decoding program can run normally on the embedded platform and correctly decode QR code information. The proposed transplantation method may have certain reference significance for different platforms QR code decoding.

Key words: UP-NETARM2410-S embedded platform; QR code decoding program; Qt-Creator; qvfb visual screen; transplant

1 引言

二维条码较常见的一种的快速响应码(QR 码), 与其它类型的二维条码相比, 它具有信息容量大, 可靠性高, 识别速度快和全方位识读等优点. QR 码的译码通常是在 PC 机上完成.

近年来, 随着计算机和信息网络技术的发展, 嵌入式系统以其小巧、灵活、实用性强等优点, 得到了越来越广泛的应用. 图形用户接口(GUI)作为人机交互的重要环节, 大量应用于嵌入式系统中. Qt 是 Trolltech 公司研发的可用于 Linux 操作系统的多平台的 C++图形用户界面程序框架, 它支持多个 GUI 平台交互开发, 被广泛的用于各种嵌入式设备开发中. 由于嵌入式平台开发环境和 PC 机开发环境的差异, 在 PC 机开发的译码程序无法直接复制到嵌入式系统中运行, 针对这一问题, 本文阐述了一种基于 Qt/Embedded 开发的 QR 码译码程序并移植到以 ARM 为核心的嵌入式系统的技术方法.

2 硬件环境构建

采用博创公司的 UP-NETARM2410-S 嵌入式平台作为硬件开发平台, 该平台的核心是三星公司的基于 ARM920T 内核的 S3C2410 处理器芯片, 处理器最高主频可达 203MHz,能够很好的满足大多数的高速处理应用需求. 同时开发平台还集成了多种硬件资源, 包

① 收稿时间:2015-06-27;收到修改稿时间:2015-08-17

括 64MBSDRAM, 64MBNandFlash, USB 控制器, LCD 控制器等[1]. 这些硬件资源特别适合嵌入式条码识别系统的开发.

摄像头采用的是与硬件开发平台配套的 Webeye2000 USB 摄像头, LCD 显示屏采用的是三星 8 英寸 TFT 带触摸液晶屏, 主要用于采集图像的显示和识别结果的显示. 系统硬件结构框图如图 1 所示.

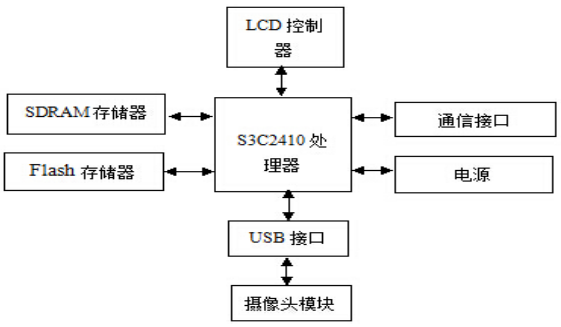

图 1 硬件结构框图

3 软件环境构建

考虑到嵌入式开发板上资源有限, 直接在开发板上开发和调试应用软件几乎不可能, 因此, 在进行嵌入式应用程序开发时, 一般的做法是构建交叉编译环境[2], 即在高性能 PC 机的集成开发环境上完成应用程序的编译和调试工作, 生成可以在开发板上运行的目标文件(.bin 文件), 最后再将该目标文件下载到目标板(开发板)上运行. 交叉开发环境模型[8]如图 2 所示.

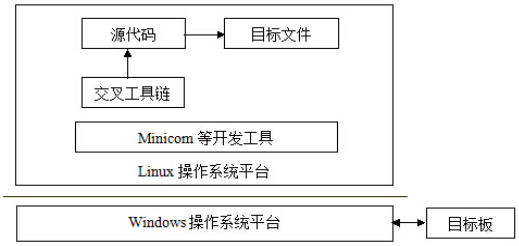

图 2 交叉开发环境模型

综合考虑 Windows 操作系统平台和 Linux 操作系统平台的优点, 软件环境构建步骤如下: 首先在 PC 机上安装虚拟机 VMware8.0, 然后在虚拟机中安装 Ubuntu12.04 操作系统, 并安装交叉编译器. 交叉编译器选择的版本是 arm-linux-gcc 3.4.1.

3.1 Bootloader 的移植

Bootloader 是在目标板上电后最开始执行的一段小代码, 它的作用是对应用程序的引导加载. 在常用的引导程序中, 三星公司的 Vivi 的结构与 Linux 的内核结构较为相似, 而且具有命令操作简单等优点, 因此本文选择 Vivi 作为引导程序, 用于引导 Linux 启动. 由于 Vivi 支持 ARM920T 内核, 对 Vivi 进行移植操作, 只需进行一些简单的宏定义和硬件配置修改即可[8].

(1)宏定义修改. 宏定义修改主要是针对 Makefile 文件, 作以下三个地方的修改:

①将 LINUX_INCLUDE_DIR 定义为交叉编译器的 include 文件夹路径. 修改如下:

LINUX_INCLUDE_DIR=/usr/local/arm/3.4.1/include/

②将 CROSS_COMPILE 定义为交叉编译器工具所在路径. 修改如下:

CROSS_COMPILE = /usr/local/arm/3.4.1/bin/arm-linux-

③将 ARM_GCC_LIBS 定义为交叉编译工具执行所依赖的库文件. 修改如下:

ARM_GCC_LIBS=/usr/local/arm/3.4.1/lib/gcc/arm-linux/3.4.1

(2)硬件配置修改. 一般情况下, Nand Flash 分为四个区域: Vivi、param、kernel 和 root, 其中 Vivi 用于存放编译好的 Vivi 二进制文件, param 为传递到 Linux 内核的参数, kernel 用于存放 Linux 内核映像文件, root 表示根文件系统. 由于根文件系统是只读文件系统, 不能用于存放用户编写的应用程序, 需要对硬件配置进行修改.

硬件配置修改主要针对 smdk.c 文件, 找到该文件中的 mtd_partition_t default_mtd_partitions[] 数组, 在该数组中, 添加分区 jffs2, 用于存放用户编写的应用程序. 添加的分区信息如下:

```
name:        "jffs2",
offset:      0x00430000,
size:        0x02af8000,
flag:        MF_JFFS2
```

在 PC 机虚拟机软件的 Ubuntu12.04 终端中输入以下指令:

```
root@ubuntu:/home/linux/vivi# make menuconfig
```

进入 Vivi 配置界面, 如图 3 所示. 在配置完后执行以下指令:

```
root@ubuntu:/home/linux/vivi# make
```

编译生成 Vivi 二进制文件，以备烧录到目标板.

图 3　Vivi 配置界面

3.2 Linux 内核重新配置与模块驱动加载

系统使用到了 USB 摄像头模块和 LCD 液晶显示屏模块, Linux 内核默认支持 LCD 液晶显示屏，并加载了相应的驱动. 因此，只需在内核中配置[3]对 USB 摄像头模块的支持，并加载该模块驱动.

在 PC 机虚拟机软件的 Ubuntu12.04 的终端中执行以下指令:

root@ubuntu:/arm2410s/kernel-2410s#make menuconfig

进入 Linux 内核配置界面，配置界面如图 4 所示.

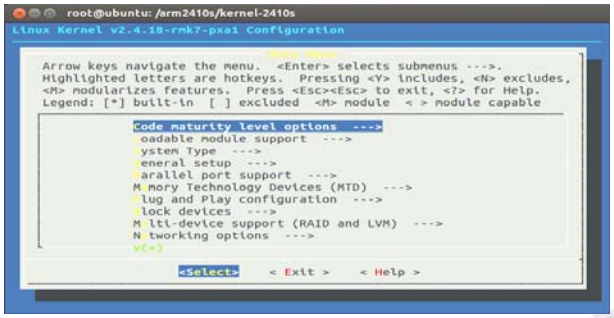

图 4　Linux 内核配置界面

选中配置菜单 USB Multimedia devices 选项中的 USB OV511 摄像头驱动支持，如图 5 所示，将 USB 摄像头模块连接到开发平台的 USB 接口，并使用如下两条命令: insmod ov511.o, insmod videodev.o 便可完成摄像头模块的加载.

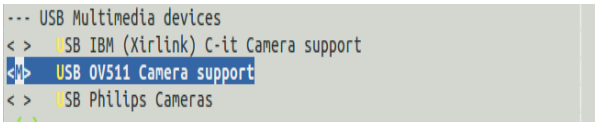

图 5　USB Multimedia devices 选项

完成 Vivi 的移植、Linux 内核配置和模块驱动加载后，将 Vivi、内核和根文件系统的映像文件烧写进开发板，上电开发板，在 Vivi 命令行中敲击 boot 命令，在 PC 机的超级终端显示如图 6 所示的启动信息，嵌入式 Linux 系统启动成功.

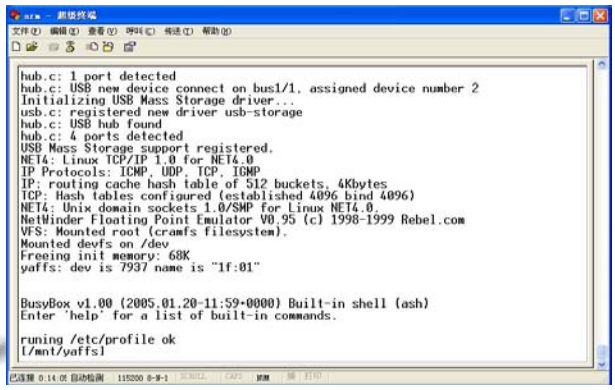

图 6　Linux 启动界面

4　QR码译码程序开发

4.1 Qt 编译环境构建

开发的译码程序最终要运行在嵌入式开发平台的 LCD 显示屏上, Qt 作为常用的一种跨平台软件开发工具，具有类功能齐全，完全面向对象等优点，因此选择 Qt 作为界面开发工具[6]，进行嵌入式平台上图形界面的开发.

QR 码译码程序开发前，在 Ubuntu12.04 下安装三个版本的 Qt 开发环境[5]:

(1)X11 版本 Qt 开发环境: 用于生成应用程序开发所需要的工具 designer, uic, qvfb. 其中, qvfb 是一个虚拟 Framebuffer 调试程序，它可以将 Qt/Embedded 应用程序的运行结果显示在 PC 机界面上.

(2)X86 版本 Qt 开发环境: 由于 qvfb 只能执行 X86 框架的应用程序，因此，X86 版本 Qt 主要用于生成关于 X86 的库和其他文件.

(3)Embedded 版本 Qt 开发环境: 用于生成应用程序在 Qt/Embedded 环境运行时所需要的库文件，以及对源程序进行交叉编译，生成在 ARM 平台上运行的可执行文件.

4.2 译码界面设计

QR 码译码界面设计通过在 PC 机 Ubuntu12.04 中安装 qtcreator-2.5.2，使用 Qt-Creator 进行译码程序界面的设计[6]. 设计的译码界面布局如图 7 所示.

设计的界面主要包括四个区域:

(1)采集图像显示区: 该区域位于界面的右半部分，

主要用于显示采集得到的包含 QR 码的图片.

(2)译码信息显示区(DecodeContent): 该区域位于界面的左半部分, 主要用于显示 QR 码译码得到的信息.

(3)采集图像按钮(Capture): 该按钮主要用于进行图像的采集.

(4)打开采集图片文件按钮(Open File): 该按钮主要用于对已保存的图片进行加载.

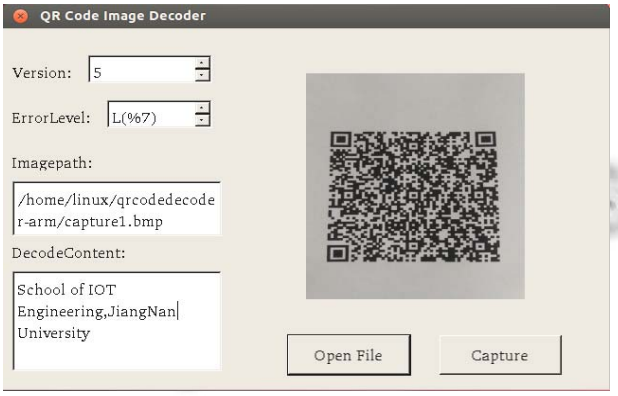

图 7 QR 码译码界面

4.3 译码程序编写

采用编写 QR 码译码类的方法实现 QR 码的译码, 设计的类主要有两个: QRcodeImage 类和 ContentDecoder 类.

(1) QRcodeImage 类的编写

QRcodeImage 类主要用于完成采集图像的操作, 该类包含的子函数以及每个子函数的功能说明, 详细见表 1.

表 1 QRcodeImage 类包含函数列表

函数名	功能说明
GetImageData()	获取采集图像信息
ImagePreprocess()	图像预处理操作
GetSamplingGrid()	建立采样网格
GetQRCodeMatrix()	由网格线建立信息矩阵

①子函数 GetImageData()主要用于获取采集到的图像的信息. 在编写该子函数时, 用到一些基本的图形分析元素, 如点, 线, 轴, 区域网格, 模块, 位置探测图形和矫正与定位图以及图像矩阵化所需要的采样网格等. 将每一个图像分析元素编写成一个以.h 结尾的头文件, 供函数 GetImageData()调用, 获取采集图像信息.

②子函数 ImagePreprocess()主要用于对采集得到的 QR 码图像进行预处理操作. 在编写该子函数时, 考虑到摄像头采集到的 QR 码图像存在的特殊情况, 将一些预处理算法添加到该函数中, 以提高译码速率. 预处理算法的部分代码如下:

```
//图像灰度化处理
 int k,r,g,b;
 for (y = 0; y < nHeight; y++) {
 for (x = 0; x < nWidth; x++) {
        for(k=0;k<3;k++)/{
          if(k==0)
       r=*(pBmpBuf+y*lineByte+x*3+k);
          if(k==1)
       g=*(pBmpBuf+y*lineByte+x*3+k);
          if(k==2)
          b=*(pBmpBuf+y*lineByte+x*3+k);}
     int m = (r * 30 + g * 59 + b * 11) / 100;//采用加权的方法进行灰度化
       intImage[y][x] = m;}
   }
```

在得到灰度化之后的图像后, 再进行二值化处理, 二值化处理部分关键代码如下:

```
for(i=0;i<height;i++){
 for(j=0;j<width;j++)
   { Histogram[intImage [i]]++;//求出图像的直方图数组
    N++;//求出图像总的象素个数
    }
}   double p[256];
    for(i=0;i<256;i++)
    {p[i]=Histogram[i]/(N*1.0);//求出各灰度级所占的比例}
    //一维最大熵二值化方法
 for(i=0;i<=t;i++)
    {if(p[i]>0) H1+=(-1.0)*p[i]*log10(p[i]);}
    N1=0;
for(i=0;i<=t;i++){
        N1+=Histogram[i];}
       if((N1==0)||(N1==N))
       {   value[t]=0;
           continue;
       }
```

```
PA=N1/(N*1.0);
Tmp1=H1/PA;
Tmp2=(H-H1)/(1-PA);
Tmp1+=Tmp2;
Tmp2=log10(PA*(1-PA));
value[t]=Tmp1+Tmp2;
int T=0;
for(i=0;i<256;i++)  {if(value[i]>value[T]) T=i;}
```

最终得到的 T 即是所需要的阈值，使用该阈值可完成对灰度化后的 QR 码图像进行二值化处理.

③子函数 GetSamplingGrid()和 GetQRCodeMatrix()主要用于对经过预处理后的图像建立采样网格，并根据建立的网格线构建信息矩阵，供下一步译码使用.

(2) ContentDecoder 类的编写

ContentDecoder 类主要完成对建立的信息矩阵的译码并进行译码信息的输出. 该类的核心函数是 DecodeData()，也是整个译码过程[4]的关键函数.

DecodeData()函数输入参数有三个：图像尺寸，版本号和图像信息矩阵. 通过输入参数传入图像的信息，首先，对图像进行去掩模运算，去掩模得到的信息矩阵，标记非数据位置，然后，通过调用读取数据码字子函数 GetCodeWord()、码字纠错函数 CorrectDataBlock()以及解析数据码字子函数 ParseDataCodeWord()，最终得到译码信息，并将译码信息显示在界面译码显示区域[7]. DecodeData()函数实现流程图如图 8 所示.

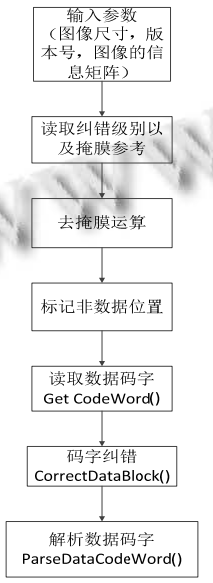

图 8　函数 DecodeData()程序流程图

5　实验测试

5.1 虚拟屏程序测试

完成 QR 码译码程序设计，下一步的工作是对程序进行交叉编译环境测试. 在虚拟机 Ubuntu12.04 操作系统下，使用虚拟 frambuffer 的应用程序 qvfb，模拟译码程序在 LCD 屏上运行的情况.

在进行测试时，首先在 Ubuntu12.04 操作系统打开一个终端，在该终端执行指令：qvfb -width 800 -height 600& 启动虚拟屏;

使用 X86 版本的 QT 库编译译码程序，生成可执行文件，再打开一个终端，在该终端下执行指令:

root@ubuntu:/qrcodedecoder-arm#./qrcodedecoder-arm –qws –qvfb

指令末端的–qws –qvfb 两项参数，用于指定使用虚拟屏方式，并且采用交叉编译进行调试.

在 qvfb 虚拟屏上运行所编写的应用程序，选用采集得到的 QR 码进行译码算法的测试，译码结果如图 9 所示.

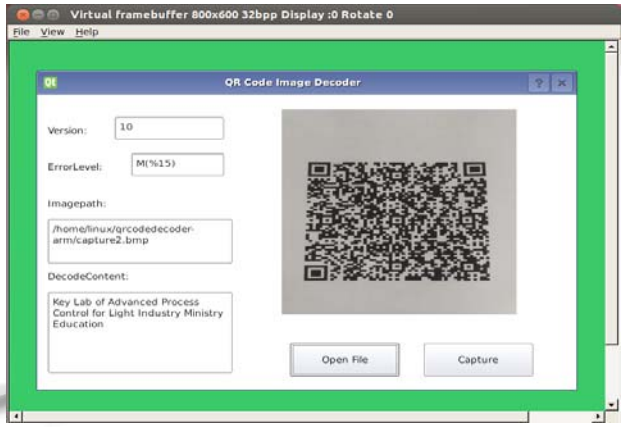

图 9　QR 码译码效果图

测试结果表明，译码程序能够进行正确译码，证明译码算法有效. 下一步的测试工作在嵌入式平台下进行.

5.2 嵌入式平台程序移植

完成虚拟屏程序的测试后，将译码程序向嵌入式平台移植，移植工作主要包括以下六个部分:

(1) 将编译出的 Qt/E 库文件拷贝到嵌入式开发平台根文件系统的 lib 目录下;

(2) 添加字库，由于 QT 应用程序运行需要依赖字体文件，将编译出的 Qt/E 库文件 fonts 目录下的字体文件拷贝到根文件系统的字体目录下;

(3) 安装 Qt 插件, 在根文件系统下建立 plugins 目录, 将 jpeg、gif 等支持图片格式的库插件拷贝到根文件系统的插件目录下;

(4) 安装应用程序, 使用交叉编译工具对调试成功的源程序进行编译, 生成运行在开发平台的可执行文件, 将该可执行文件拷贝到根文件系统 mnt/yaffs 目录下;

(5) 编写 Qt 启动脚本 qtconfig, 编辑根文件系统 etc/init.d/rcS 下启动脚本 qtconfig, 在最后加入启动语句./qtconfig&, 便可实现程序的开机自启动.

(6)制作根文件系统镜像, 执行./mkcramfs root root.cramfs, 将文件系统镜像通过串口下载到 Nand Flash 中;

启动系统, 译码应用程序自动运行, 在嵌入式平台的 LCD 屏上看到程序运行效果如图 10 所示, 从运行结果可知, 译码程序已经在嵌入式平台正常运行.

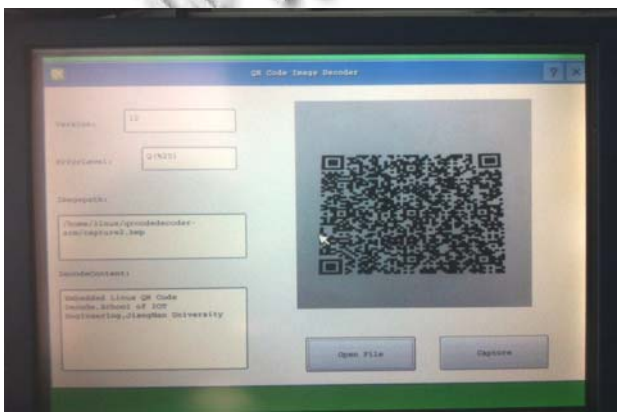

图 10 译码程序嵌入式平台运行效果图

6 结语

本文给出了一套完整的在 PC 机进行仿真调试, 以文件系统形式实现自启动的 QR 码译码程序的开发方法, 介绍了利用 Qt 集成开发平台进行程序设计的步骤和过程, 给出了具体的实现代码, 首先, 在 qvfb 虚拟屏完成对 QR 码译码程序的测试工作, 测试译码程序的译码效果.

最后, 将译码程序移植到嵌入式平台运行, 观察译码程序在目标板上的运行效果, 验证了译码程序的有效性, 对于译码程序向不同嵌入式平台移植有一定的借鉴意义.

参考文献

1 北京博创兴业科技有限公司.UP-NETARM 2410-S 开发平台硬件说明书.北京,北京博创兴业科技有限公司,2008.
2 刘永林,梁莹,王诗琴等.基于 Linux 的嵌入式交叉编译环境的建立及实现.电脑开发与应用,2011,24(7):68–70.
3 荀艳丽.Linux 内核在 S3C2410 上移植的研究.现代电子技术,2012,35(12):13–15.
4 中华人民共和国国家技术监督局.GB/T18284-2000.快速响应矩阵码.北京:中国标准出版社,2000.
5 田磊.嵌入式 Linux 系统中基于 QT 库的应用程序设计.实验室研究与探索,2014, 33(5): 84–86.
6 康维新.嵌入式 Linux 系统开发与应用.北京:机械工业出版社,2011.4,211–222
7 龙泉.基于 Arm_Linux 的 QRCode 识别系统的研究和实现[博士学位论文].成都:电子科技大学,2011
8 赵健雄.从 ARM9 到 Linux 系统设计与开发直通车.北京:电子工业出社,2014.