基于Android6.0的嵌入式红外图像远程采集系统

何宇清（1） 徐大为（2）

1. 天津大学电子信息工程学院300072
2. 北陆先端科学技术大学院大学

摘要：远程红外温度图像的采集，对于工业实际应用有重要价值。本文设计了基于Android6.0，采用FLIR ONE摄像头，红外热图像实时采集、传输解决方案,能够对测量物体温度进行准确、稳定、实时的监测，设计了远程控制系统。经过测试，系统运行稳定，成本低，运行效果良好。

关键词：红外图像采集；安卓；远程控制

Abstact：Remote infrared temperature image acquisition is important for industrial applications. This paper designs a system which cans remote measure object temperature, and it is accurate, stable. The system based on Android6.0. Infrared thermal image real-time acquisition uses FLIR ONE sensor. The system cans real-time monitoring and Real-time transmission. After testing, the system is stable, low cost, good operation.

Key words：infrared image acquisition；Android；remote control

前言

红外图像探测通过非接触探测红外能量（热量），并将其转换为电信号，进而在显示器上生成热图像和温度值，并可以对温度值进行计算。红外探测能够将探测到的热量精确量化，或测量，使科研人员不仅能够观察热图像，还能够对发热的故障区域进行准确识别和严格分析。红外探测成像已广泛应用于在过程监测以及机械设备状态监测等工业领域。 红外热成像不仪能够帮助我们看到实时热图像，而且还能通过红外视频及相关温度测量数据来进行过程控制及事前报警。

本文介绍了一种基于Android6.0的远程控制，红外图像采集，远程传输系统。在系统中android手机利用周围无线网络资源与远程计算机进行交互，实现相应功能。

1Android的系统架构

Android其本质就是在标准的Linux系统上增加了Java虚拟机Dalvik，并在Dalvik虚拟机上搭建了一个JAVA的应用程序框架，所有的应用程序都是 基于JAVA的应用程序框架之上。

Android操作系统自顶向下分为四个层次，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和linux核心层。其框架如图1所示。



图1 Android框架

1. 应用程序层。Android操作系统同一系列核心应用程序包一起发布，其核心应用程序包括E-mail客户端，短信，日历，网页浏览器等，目前所有的应用程序都是使用JAVA语言编写的，每一个应用程序由一个或者多个活动组成，活动必须以Activity类为超类，活动类似于操作系统上的进程，但是活动比操作系统的进程要更为灵活，与进程类似的是，活动在多种状态之间进行切换。利用JAVA的跨平台性质，基于Android框架开发的应用程序可以不用编译运行于任何一台安装有android系统的平台。
2. 应用程序框架层。开发者使用核心应用程序来调用Android框架提供的API，应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能，并且任何其它的应用程序都可以使用其所发布的功能（不过得遵循框架的安全性限制）。且该应用程序重用机制也使用户可以方便的替换程序组件。隐藏在每个应用后面的是一系列的服务和系统, 其中包括：

a.丰富而又可扩展的视图（Views），可以用来构建应用程序， 它包括列表（lists），网格（grids），文本框（text boxes），按钮（buttons）， 甚至可嵌入的web浏览器。

 b.内容提供器（Content Providers）使得应用程序可以访问另一个应用程序的数据（如联系人数据库）， 或者共享它们自己的数据。

 c.资源管理器（Resource Manager）提供非代码资源的访问，如本地字符串，图形，和布局文件（layout files）。

 d.通知管理器（Notification Manager）使得应用程序可以在状态栏中显示自定义的提示信息。

 e.活动管理器（Activity Manager）用来管理应用程序生命周期并提供常用的导航回退功能。

1. 系统运行库层。Android包含一些C/C++库，这些库能被Android系统中不同的组件使用。它们通过Android应用程序框架为开发者提供服务。主要包括基本的C库、以及多媒体库以支持各种多媒体格式、位图和矢量字体、2D和3D图形引擎、浏览器、数据库支持。另外这里还有一个硬件抽象层。其实Android并非所有的设备驱动都放在linux内核里面，有一部分实现在用户空间，这么做的主要原因是可以避开Linux所遵循的GPL协议，一般情况下如果要将Android移植到其他硬件去运行，只需要实现这部分代码即可。包括：显示器驱动，声音，相机，GPS，GSM等等。

对于Android 运行库，Android 包括了一个核心库，该核心库提供了JAVA编程语言核心库的大多数功能。每一个 Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。Dalvik被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik虚拟机执行（.dex）的Dalvik可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由JAVA编译器编译，然后通过SDK中的 "dx" 工具转化成.dex格式由虚拟机执行。 Dalvik虚拟机依赖于linux内核的一些功能，比如线程机制和底层内存管理机制。

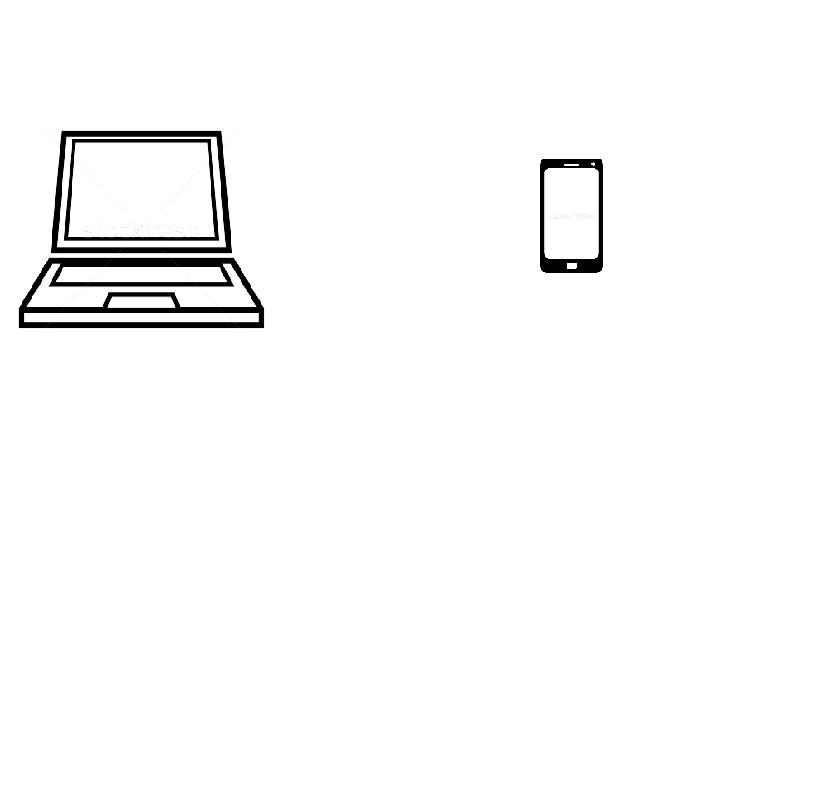
1. linux核心层。Android底层是一个基于Linux 2.6 内核开发的独立操作系统。主要是添加了一个名为Goldfish的虚拟CPU以及Android运行所需的特定驱动代码。该层用来提供底层服务，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 作为一个虚拟的中间层，该层位于硬件和软件层之间。

2系统的总体设计

基于智能手机的嵌入式热成像数据采集系统分为2部分，移动平台以及PC客户端。移动平台由智能手机和FLIR ONE热成像设备组成，作为系统的载体，进行热成像图像的获取，文件保存，和通过WLAN/数据线向PC客户端进行图像传输等工作任务。

PC客户端进行图像，环境信息的显示并对移动平台进行控制。系统总体框图如下图所示。本文将详细介绍基于Android的移动客户端开发

系统功能设计：热成像图像采集系统集成在智能手机上，因此，装有该系统的智能手机除了自身的移动通讯功能外, 还应具有热成像数据采集功能, 实现热成像数据采集涉及的数据获取、属性录入、数据预处理、数据显示、查询分析、数据存储及数据传输等。此外, 受智能手机的软硬件条件限制, 嵌入式热成像采集系统应尽可能精炼, 以适宜在资源紧缺的智能移动设备上使用。为此, 本系统主要设计了以下功能:热成像数据预览的动态显示、数据采集(即渲染模式的操作)、坐标转换、数据存储、数据的无线或有线传输。



**Mobile Device**

**Thermal photo Capture/Send**

**PC Client**

**Thermal photo Receiver**

Snap!

Wifi/Cable connected

**Thermal Image acquisition system**

3系统的具体实现

根据上述系统总体与功能设计, 本文以Android Studio为平台, 开展基于智能手机的嵌入式热成像数据采集系统的研发。在Android系统中进行开发, 需要事先配置系统访问权限。本系统配置的主要权限包括调用相关设备、连接互联网、及数据存储等。下面将对系统主要实现过程进行介绍。

主要函数：FLIR ONE设备每得到一帧热成像数据后，都要调用此委托方法，来完成显示屏幕中央的焦点温度，将像素包含的温度值影射到RGB值，保存当先热成像图片，与向PC客户端进行实时传输的图像流传输。

**public void** onFrameProcessed(**final** RenderedImage renderedImage){  
 **if** (renderedImage.imageType() == RenderedImage.ImageType.***ThermalRadiometricKelvinImage***){  
 **int**[] thermalPixels = renderedImage.thermalPixelValues();

*// 收集中央9个像素点温度的平均值作为图像焦点温度* **int** width = renderedImage.width();  
 **int** height = renderedImage.height();  
 **int** centerPixelIndex = width \* (height/2) + (width/2);  
 **int**[] centerPixelIndexes = **new int**[] {  
 centerPixelIndex, centerPixelIndex-1, centerPixelIndex+1,  
 centerPixelIndex - width,  
 centerPixelIndex - width - 1,  
 centerPixelIndex - width + 1,  
 centerPixelIndex + width,  
 centerPixelIndex + width - 1,  
 centerPixelIndex + width + 1  
 };  
  
 **double** averageTemp = 0;  
  
 **for** (**int** i = 0; i < centerPixelIndexes.**length**; i++){  
 *// FLIR ONE设备采集来的数据需要调用renderedImage.thermalPixelValues()方法*

*// 来得到无符号值* **int** pixelValue = (thermalPixels[centerPixelIndexes[i]]);  
 averageTemp += (((**double**)pixelValue) - averageTemp) / ((**double**) i + 1);  
 }  
 **double** averageC = (averageTemp / 100) - 273.15;  
 NumberFormat numberFormat = NumberFormat.*getInstance*();  
 numberFormat.setMaximumFractionDigits(2);  
 numberFormat.setMinimumFractionDigits(2);  
 **final** String spotMeterValue = numberFormat.format(averageC) + **"ºC"**;  
 runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 ((TextView)findViewById(R.id.***spotMeterValue***)).setText(spotMeterValue);  
 }  
 });

**if** (**frameProcessor**.getImageTypes().size() == 1){  
 *// 将0到100摄氏度映射到8位灰度图像* **byte**[] argbPixels = **new byte**[width \* height \* 4];  
 **final byte** aPixValue = (**byte**)255;  
 **for** (**int** p = 0; p < thermalPixels.**length**; p++) {  
 **int** destP = p \* 4;  
 **byte** pixValue = (**byte**)(Math.*min*(0xff,Math.*max*(0x00,(thermalPixels[p]-27315)\*(255.0/10000.0))));  
  
 argbPixels[destP + 3] = aPixValue;  
 argbPixels[destP] = argbPixels[destP + 1] = argbPixels[destP + 2] = pixValue;  
 }  
 **final** Bitmap demoBitmap = Bitmap.*createBitmap*(width, renderedImage.height(), Bitmap.Config.***ARGB\_8888***);  
  
 demoBitmap.copyPixelsFromBuffer(ByteBuffer.*wrap*(argbPixels));  
  
 updateThermalImageView(demoBitmap);  
 }  
 }**else** {  
 **if** (**thermalBitmap** == **null**){  
 **thermalBitmap** = renderedImage.getBitmap();  
 } **else** {  
 **try** {  
 renderedImage.copyToBitmap(**thermalBitmap**);  
 } **catch** (IllegalArgumentException e){  
 **thermalBitmap** = renderedImage.getBitmap();  
 }  
 }  
 updateThermalImageView(**thermalBitmap**);  
 }  
 *//当快门按下后，如果有需要，则按要求改变图像渲染模式*   
 **if**(**this**.**capImageType**>0){  
 **if**(**this**.**capImageType**==1){  
 **frameProcessor**.setImageTypes(EnumSet.*of*(RenderedImage.ImageType.***BlendedMSXRGBA8888Image***));  
 **this**.**capImageType**=0;  
 }**else if**(**this**.**capImageType**==2){  
 **frameProcessor**.setImageTypes(EnumSet.*of*(RenderedImage.ImageType.***VisibleAlignedRGBA8888Image***));  
 **this**.**capImageType**=0;  
 }**else if**(**this**.**capImageType**==3){  
 **frameProcessor**.setImageTypes(EnumSet.*of*(RenderedImage.ImageType.***ThermalRadiometricKelvinImage***));  
 **this**.**capImageType**=0;  
 }  
 }  
 */\*  
 当快门按下后，按需要的渲染模式保存为混合热成像图片，和像素温度数组文件  
 \*/* **if** (**this**.**imageCaptureRequested**) {  
 **imageCaptureRequested** = **false**;  
 **final** Context context = **this**;  
 **new** Thread(**new** Runnable() {  
 **public void** run() {  
 String path = Environment.*getExternalStoragePublicDirectory*(Environment.*DIRECTORY\_PICTURES*).toString();  
 SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy-MM-dd-HH-mm-ssZ"**, Locale.*getDefault*());  
 String formatedDate = sdf.format(**new** Date());  
 String fileNameT = **"D-"** + formatedDate + **".txt"**;  
 String fileNameP = **"T-"** + formatedDate + **".jpg"**;  
 String fileNameV = **"V-"** + formatedDate + **".jpg"**;  
 **try**{  
 **if** (renderedImage.imageType() == RenderedImage.ImageType.***BlendedMSXRGBA8888Image***) {  
 **lastSavedPath** = path + **"/"** + fileNameP;  
 renderedImage.getFrame().save(**new** File(**lastSavedPath**), RenderedImage.Palette.***Iron***, RenderedImage.ImageType.***BlendedMSXRGBA8888Image***);  
 }  
 **if** (renderedImage.imageType() == RenderedImage.ImageType.***VisibleAlignedRGBA8888Image***) {  
 **lastSavedPath** = path + **"/"** + fileNameV;  
*// renderedImage.getFrame().save(new File(lastSavedPath), , RenderedImage.ImageType.VisibleAlignedRGBA8888Image);* }  
 **if** (renderedImage.imageType() == RenderedImage.ImageType.***ThermalRadiometricKelvinImage***){  
 **int**[] thermalPixels = renderedImage.thermalPixelValues();  
 **short**[] shortPixels=renderedImage.thermalPixelData();  
 **int** height = renderedImage.height();  
 **int** width = renderedImage.width();  
 **lastSavedPath** = path+ **"/"** + fileNameT;  
 File myFile = **new** File(**lastSavedPath**);  
 myFile.createNewFile();  
 FileOutputStream fOut = **new** FileOutputStream(myFile);  
 PrintStream pStream = **new** PrintStream(fOut);  
 pStream.printf(**"%d %d\r\n"**,height,width);  
 pStream.printf(**"%d\r\n"**,shortPixels.**length**);  
 *// for (int p = 0; p < thermalPixels.length; p++){  
 // pStream.printf("%d ",thermalPixels[p]);  
 // }* **for** (**int** i = 0; i < shortPixels.**length**; i++){  
 pStream.printf(**"%d "**,(**int**)shortPixels[i]);  
 }  
 pStream.close();  
 fOut.close();  
 }  
  
 MediaScannerConnection.*scanFile*(context,  
 **new** String[]{path + **"/"** + fileNameP}, **null**,  
 **new** MediaScannerConnection.OnScanCompletedListener() {  
 @Override  
 **public void** onScanCompleted(String path, Uri uri) {  
 Log.*i*(**"ExternalStorage"**, **"Scanned "** + path + **":"**);  
 Log.*i*(**"ExternalStorage"**, **"-> uri="** + uri);  
 }  
  
 });  
  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
  
 **thermalImageView**.animate().setDuration(50).scaleY(0).withEndAction((**new** Runnable() {  
 **public void** run() {  
 **thermalImageView**.animate().setDuration(50).scaleY(1);  
 }  
 }));  
 }  
 });  
 }  
 }).start();  
  
 }  
 *//向指定地址的另一终端发送该帧图像*  
 **if** (**streamSocket** != **null** && **streamSocket**.isConnected()){  
 **try** {  
 *// 发送PNG图像* **final** OutputStream outputStream = **streamSocket**.getOutputStream();  
 **final** ByteArrayOutputStream bufferStream = **new** ByteArrayOutputStream();  
  
 **thermalBitmap**.compress(Bitmap.CompressFormat.***WEBP***, 100, bufferStream);  
 bufferStream.flush();  
 (**new** Thread() {  
 @Override  
 **public void** run() {  
 **super**.run();  
 **try** {**byte**[] headerBytes = ByteBuffer.*allocate*((Integer.***SIZE*** + Short.***SIZE***) / 8).putInt(bufferStream.size()).putShort((**short**)**deviceRotation**).array();  
 **synchronized** (**streamSocket**) {  
 outputStream.write(headerBytes);  
 bufferStream.writeTo(outputStream);  
 outputStream.flush();  
 }  
 bufferStream.close();  
  
  
 } **catch** (IOException ex) {  
 Log.*e*(**"STREAM"**, **"Error sending frame: "** + ex.toString());  
 }  
 }  
 }).start();  
 } **catch** (Exception ex){  
 Log.*e*(**"STREAM"**, **"Error creating PNG: "**+ex.getMessage());  
  
 } }  
 }

4实现结果

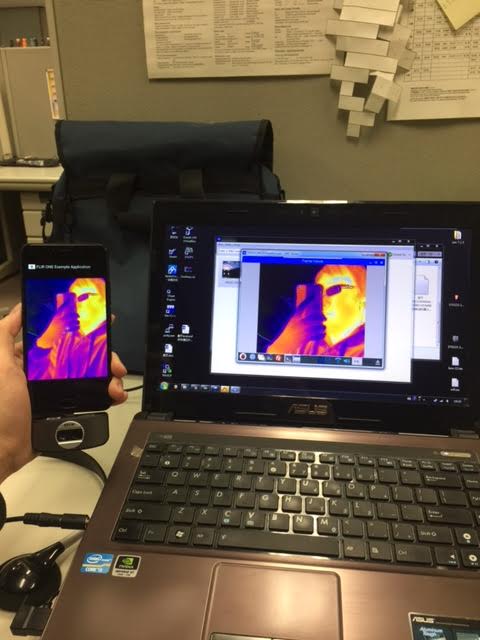
在移动平台端，使用可见光模式拍摄的图像，图像渲染模式VisibleAlignedRGBA8888Image：



在移动平台端，使用红外模式拍摄的热成像，图像渲染模式BlendedMSXRGBA8888Image：



于移动平台上得到的热成像通过无线局域网，使用图像传输方式，在PC客户端显示：



5结语

参考文献