**C++项目Qt视频、音频部分整理**

QT是一个跨平台的C++图形化界面应用程序框架，凭借其优良的跨平台特性、良好的封装机制、丰富的API、大量的开发文档等优点，得到了很多软件开发者的青睐。Qt的多媒体技术发展到现在，使用简单、功能灵活，已经支持的跨平台开发，支持QML以获得更多资源，它为开发者提供了一系列丰富的接口，使开发者能够轻松便利的使用平台、多媒体回放以及操作摄像机和收音机设备的多媒体技术。

目前适应市场要求，我们对C++的项目做了一次更新，新增Qt视频、音频部分，此次更新新增udp协议。

**第一部分 UDP协议**

1. **UDP协议**

QUdpSocket类可以用来发送和接收UDP数据包。UDP是一种不可靠的，面向数据报的协议。一些应用层的协议使用UDP，因为它比TCP更加小巧轻便。采用UDP，数据是以包(数据报)的形式从一个主机发送到另一个主机的。这里并没有连接的概念，而且如果UDP没有被传递成功，它不会像发送者报告任何错误。

1. **UDP广播**

广播使用广播地址255.255.255.255，将消息发送到在同一广播网络上的每个主机值得强调的是：本地广播信息是不会被路由器转发。当然这是十分容易理解的，因为如果路由器转发了广播信息，那么势必会引起网络瘫痪。这也是为什么IP协议的设计者故意没有定义互联网范围的广播机制。

广播地址通常用于在网络游戏中处于同一本地网络的玩家之间交流状态信息等。

其实广播顾名思义，就是想局域网内所有的人说话，但是广播还是要指明接收者的端口号的，因为不可能接受者的所有端口都来收听广播。

1. **UDP组播：组播IP地址：224.0.0.0---239.255.255.255**

多播，也称为“组播”，将网络中同一业务类型主机进行了逻辑上的分组，进行数据收发的时候其数据仅仅在同一分组中进行，其他的主机没有加入此分组不能收发对应的数据。

　　在广域网上广播的时候，其中的交换机和路由器只向需要获取数据的主机复制并转发数据。主机可以向路由器请求加入或退出某个组，网络中的路由器和交换机有选择地复制并传输数据，将数据仅仅传输给组内的主机。多播的这种功能，可以一次将数据发送到多个主机，又能保证不影响其他不需要（未加入组）的主机的其他通信。

相对于传统的一对一的单播，多播具有如下的优点：

　　1、具有同种业务的主机加入同一数据流，共享同一通道，节省了带宽和服务器的优点，具有广播的优点而又没有广播所需要的带宽。

　　2、服务器的总带宽不受客户端带宽的限制。由于组播协议由接收者的需求来确定是否进行数据流的转发，所以服务器端的带宽是常量，与客户端的数量无关。

3、与单播一样，多播是允许在广域网即Internet上进行传输的，而广播仅仅在同一局域网上才能进行。

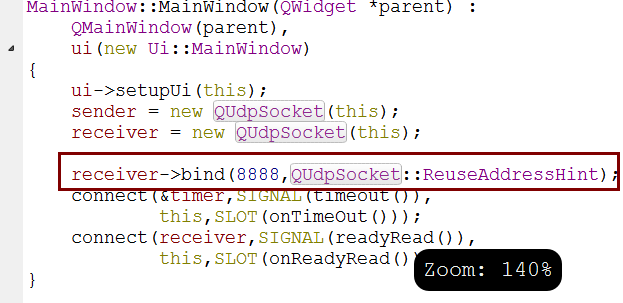
组播的缺点：

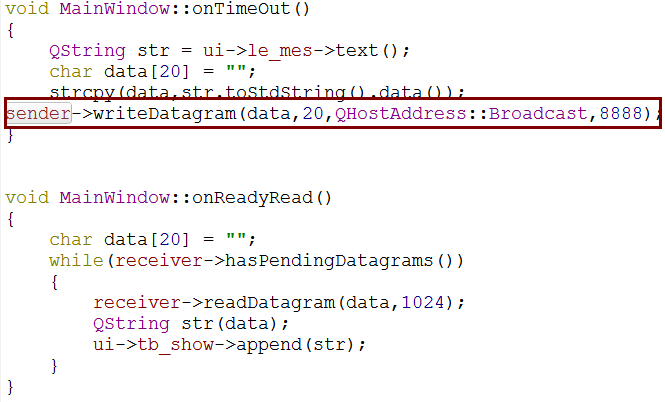
　　1、多播与单播相比没有纠错机制，当发生错误的时候难以弥补，但是可以在应用层来实现此种功能。

　　2、多播的网络支持存在缺陷，需要路由器及网络协议栈的支持。

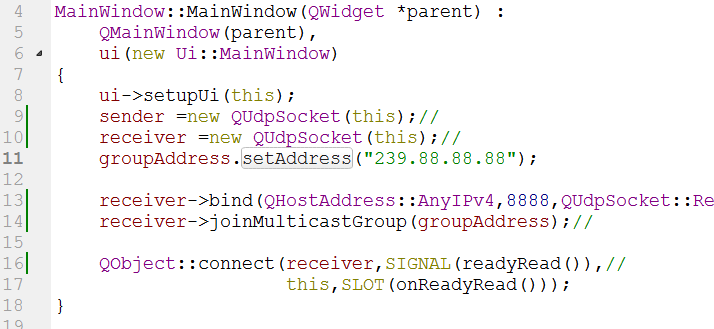
3、多播的应用主要有网上视频、网上会议等。

1. **UDP广播**





1. **UDP组播**



**第二部分 视频部分**

1. **视频编码**

音视频编码技术就是采用特定的压缩技术，将某个音频或者视频格式的文件格式转换成另一种音频或者视频格式文件的方式。

视频的编码格式有很多，目前最为常用的事MPEG系列和H.26X系列。其中MPEG系列是由ISO所指定而发布的视频、音频、数据的压缩标准。H.264系列是由国际电信联盟（ITU-T）开发的视频压缩编码标准，包括H.261、H.262、H.263++、H.264，其中H.264具有更高的数据压缩比，在同等图像质量下，H.264的数据压缩率比MEPG-2高2倍~3倍，比MEPG-4提高了近30%。经过H.264压缩的视频数据，提高了存储性能，在网络传输过程中占用的宽带更少更经济，极大地抑制了由于摄像机噪声导致的图像失真，背景流动现象，使图像质量更加清晰。

1. **基于Qt多媒体技术的音视频采集和编码实现**
2. 多媒体模块的导入

Qt5.0版本中的多媒体模块提供了一组丰富的QML类型和C++类以满足多媒体开发的需要，另外还提供了必要的API接口来访问照相机和收音机设备。

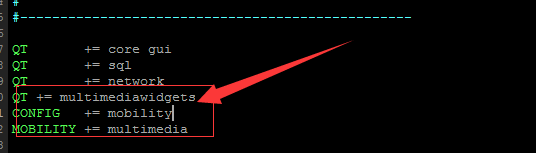
要在Qt5.0以上的版本中使用多媒体模块，首先应该在工程中的pro文件进行如下设置：

CONFIG += mobility

MOBILITY += multimedia

QT += multimediawidgets

像图片一样设置：



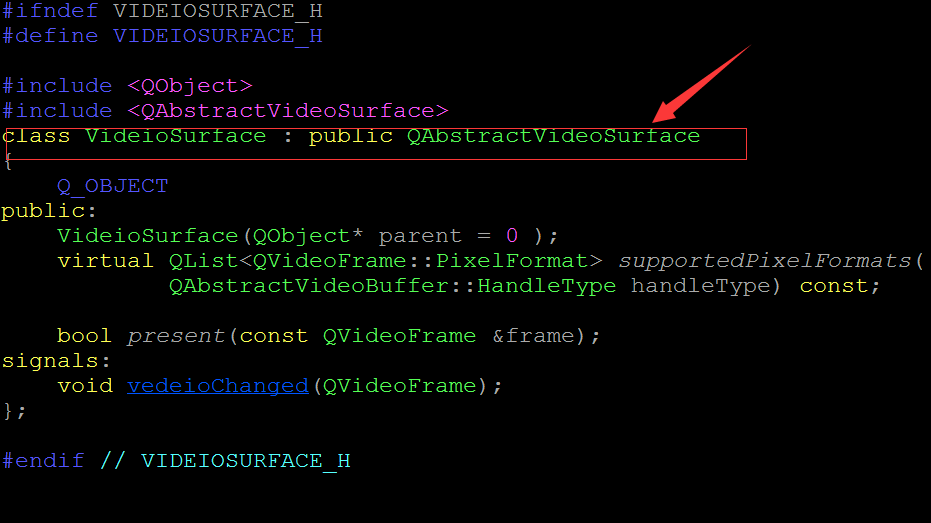
只有这样设置之后才能使用Qt的多媒体库。

1. FFMpeg头文件以及动态库的添加

把编译好的FFmpegSDK文件放到一个文件夹内，首先在工程中设置文件路径以及需要的库文件，在工程中的文件包含使用的头文件，注意的是，因为FFMepg库的接口都是C函数，头文件也没有extren “C”的声明，而Qt是C++编译环境，所以使用文投建时，需要添加extern “C”。

1. 摄像机视频的采集和编码

Qt中的QCamera类为系统的摄像机设备提供了接口，它和QVideoWidget搭配使用用来显示摄像机视频图像，搭配QMediaRecoder可以保存摄像机视频，搭配QCameraimageCapture可以抓拍摄像机图像。我们现在所介绍的视频的采集方法，除了需要实时显示摄像机图像之外，还需要同步获取摄像机视频源数据，所以引入了QAbstractVideoSurface这个类。在工程中添加VideoSurface，该类继承自QAstractVideoSurface这个基类。如下图所示：

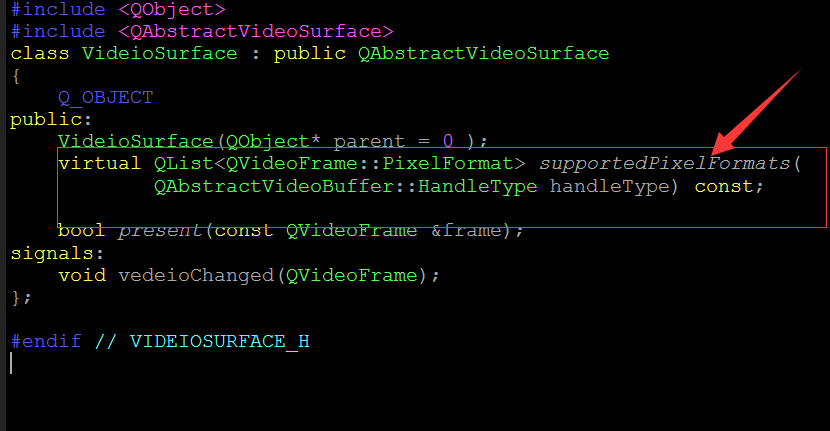


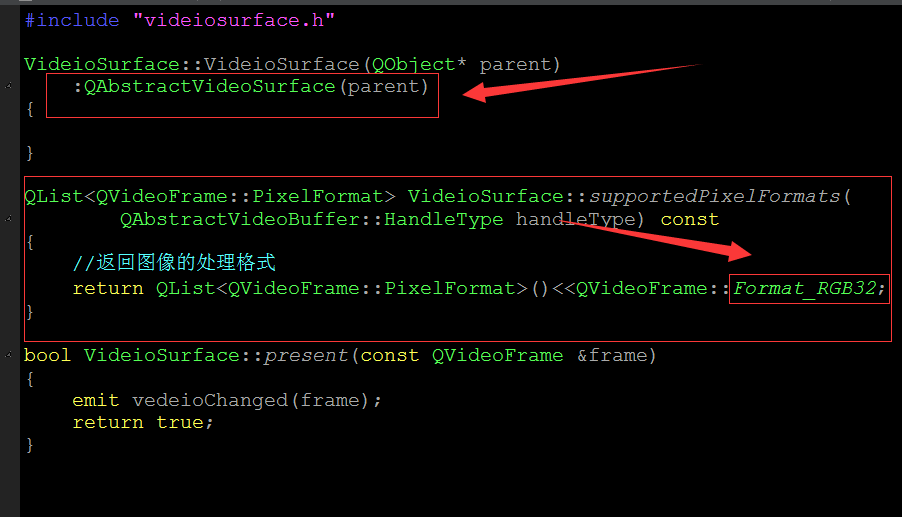
首先，我们按照下面步骤来：

1. .重载虚基函数

QList<QVideoFrame::PixelFormat>supportedPixelFormats(QAbstractVideoBuffer::HandleType = QAbstractVideoBuffer::NoHandle)const;

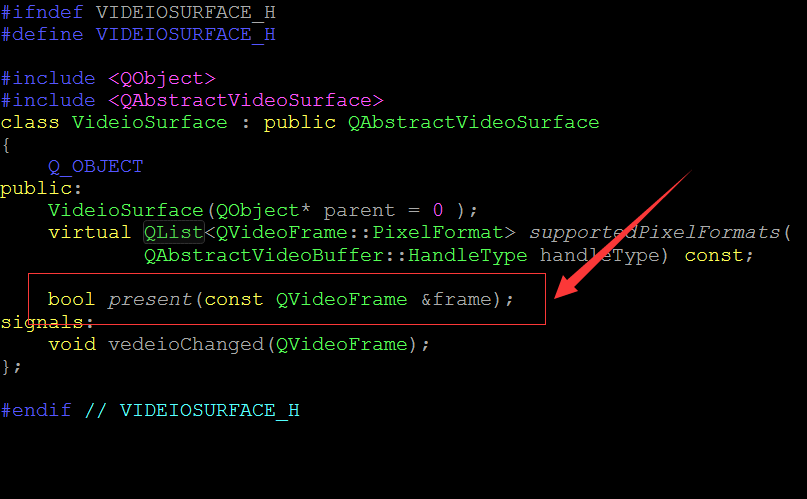
用于设置程序支持的原始视频格式，如下图所示：





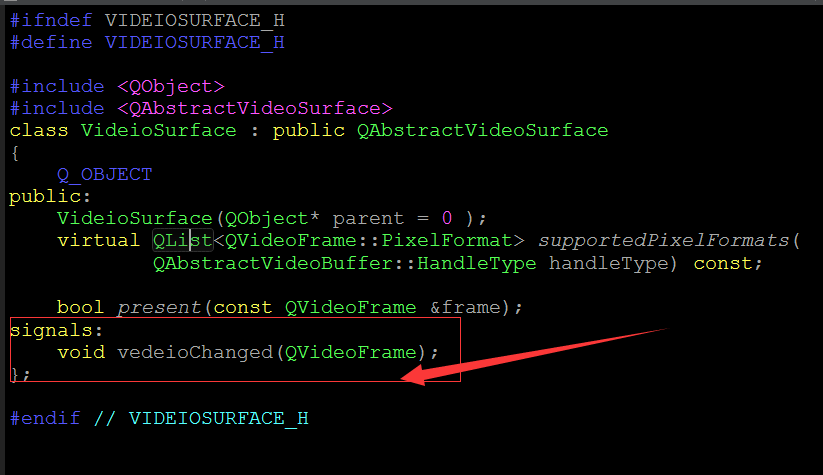
这里注意一下：window下转换的图片格式是Format\_RGB32,苹果格式是Format\_ARGB32。

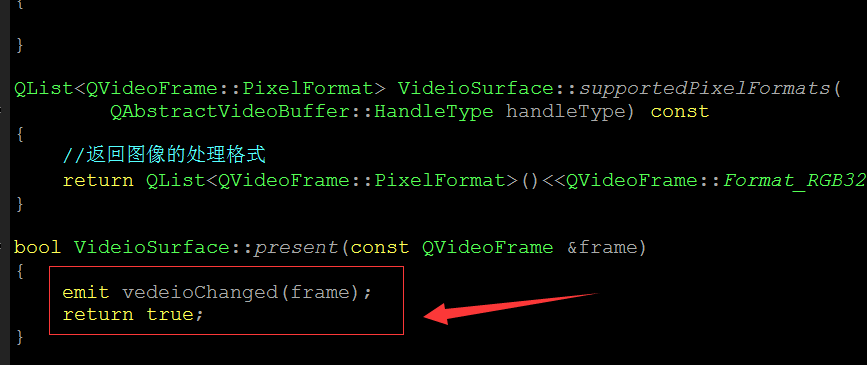
1. 重载基类的bool present(const QVideoFrame& frame)函数，用于获取当前帧的视频源数据，如下图：



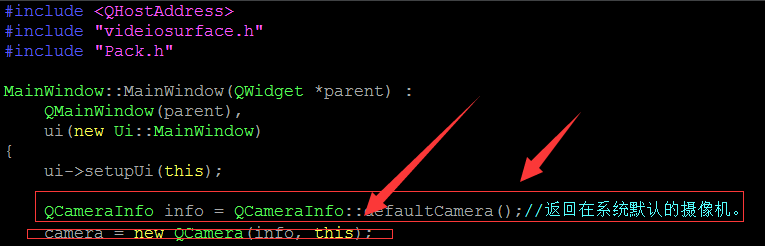


1. 在VideoSurface类中添加一个自定义信号videoChanged信号，我们在present函数中发送该信号。如下图所示：





4）使用QCameraInfo::defaultCamera();返回在系统默认的摄像机，创建摄像机对象，在相应的槽函数中使用start跟stop函数打开和关闭摄像头信息。如下图所示：





1. 刚刚在第二步中我们看到VideoSurface中present函数发送了一个信号到主窗口，那么我们主窗口自定义一个槽函数来实现捕捉摄像头所扑捉到的视频图片信息，并通过UDP

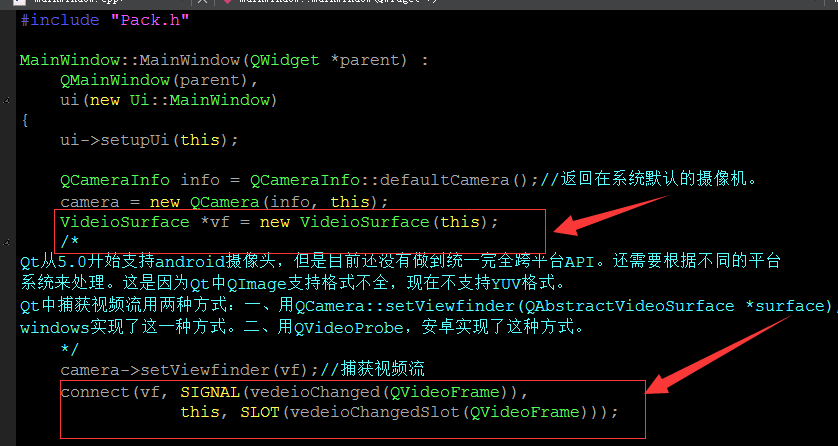
发送出去。首先我们创建VideoSurface的对象，用QCamera::setViewfinder(QAbstractVideoSurface \*surface)这种方式捕获视频流。这里注意：Qt从5.0开始支持android摄像头，但是目前还没有做到统一完全跨平台API。还需要根据不同的平台系统来处理。这是因为Qt中QImage支持格式不全，现在不支持YUV格式。Qt中捕获视频流用两种方式：

一、用QCamera::setViewfinder(QAbstractVideoSurface \*surface),

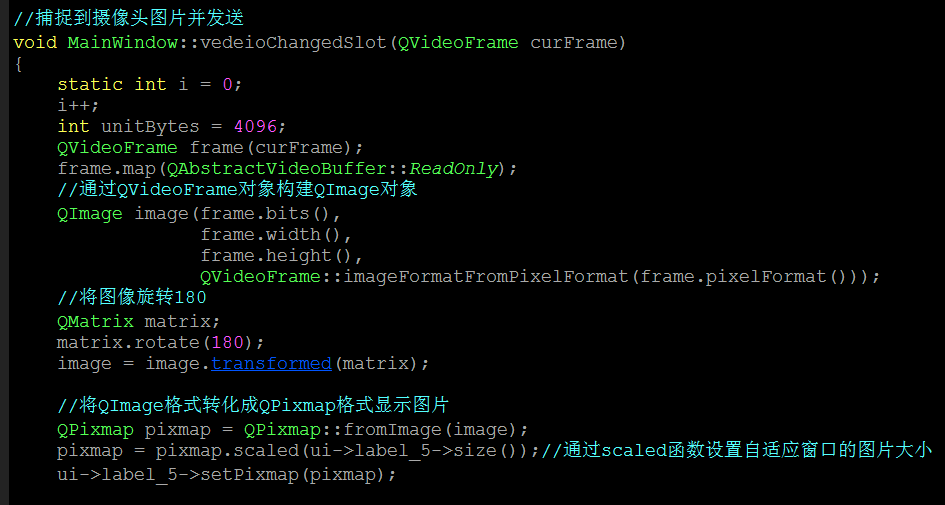
windows实现了这一种方式。二、用QVideoProbe，安卓实现了这种方式。如下图所示：



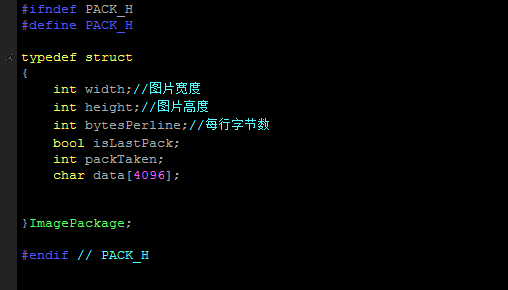
接下来我们再连接我们的自定义槽函数，如下图所示：

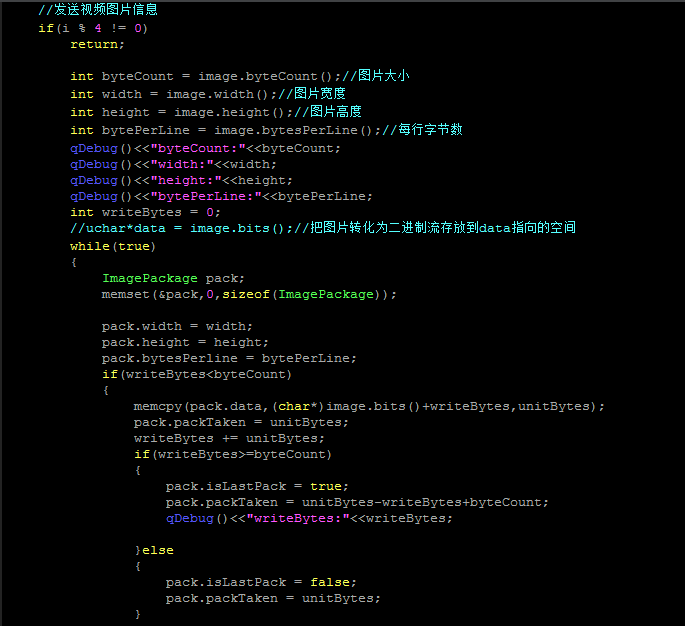


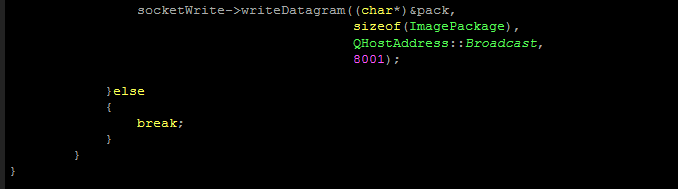
1. 我们再自定义的槽函数中首先获取摄像头扑捉的图片信息，将它显示在自己的窗口上，这里首先通过QVideoFrame对象构造QImage对象。接下来再将扑捉的图片通过QMatrix创建对象matrix，调用rotate函数将图片旋转180度，最后将QImage的图片格式转化成QPixmap格式，并使用scaled函数设置自适应的窗口图片大小，将图片显示在窗口部件中。如下图所示：



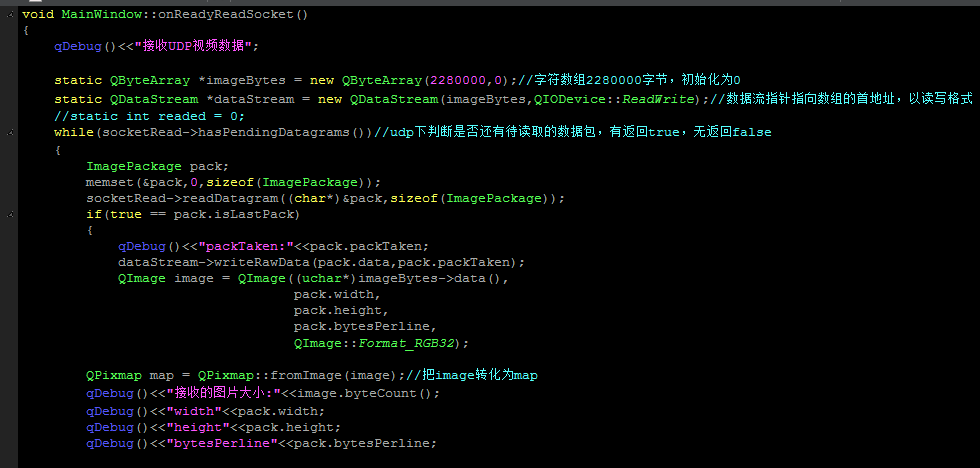
1. 接下来我们发送视频图片信息，获取图片宽度、高度、每行字节数、并将图片转化成二进制流，发送出去。如下图所示：

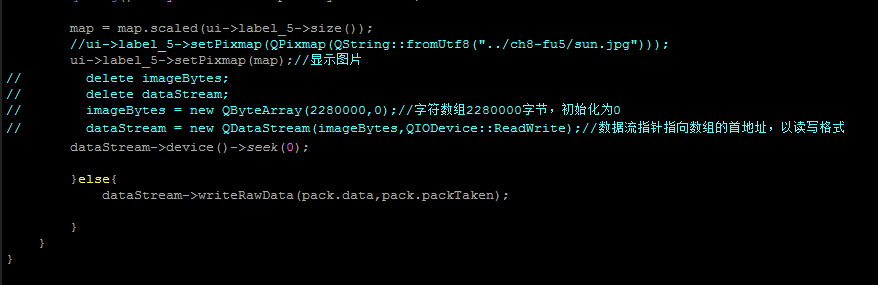






1. 最后一步，UDP接收视频图片信息，并显示在窗口部件上，具体操作如下图所示：





1. 注意：最后项目过程中，可能昨晚出现接收到的视频图片信息很模糊甚至看不清到底是什么，这里有三种解决方法供参考：

第一种方法：放到线程中去做视频的发送与接收

第二种方法：设置图片大小，将图片大小设置小一点

第三种方法：调整发送帧率。

**第三部分 音频部分**

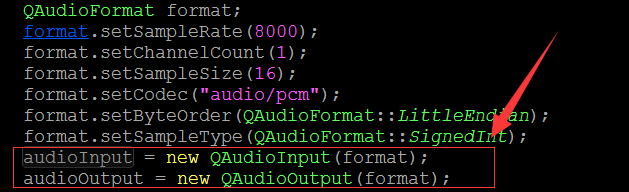
1. **音频编码**

G.711是由国际电信联盟（ITU-T）制定的音频编码方式，又称ITU-TG.711，目前有两种编码方式：A-law以及Mu-law，Mu-law主要应用于北美跟日本，A-law主要用于欧洲和世界其他地区。

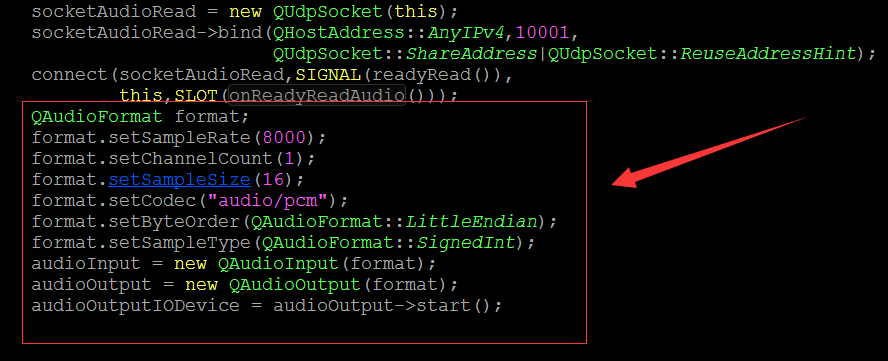
1. **基于Qt多媒体技术的音频的采集和编码实现**

Qt中的QAudioInput类可以进行本地的麦克风、收音机等设备的音频采集，而QAudioOutput类可以实现把本地的音频数据输出到音频输出设备，实现音频的播放功能。具体步骤如下：

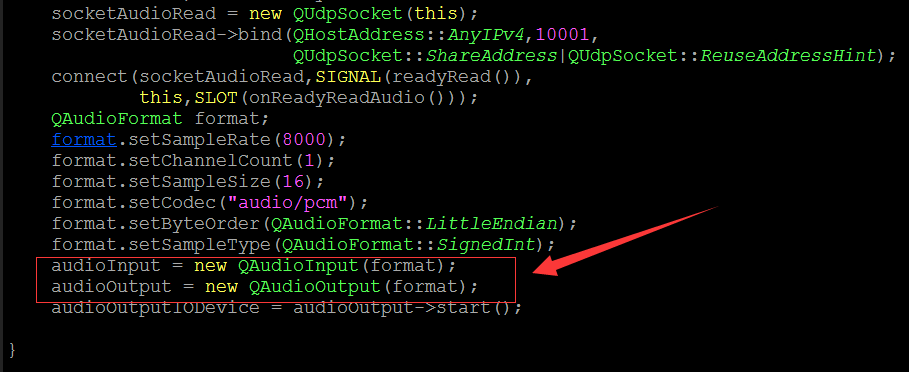
1. 获取声音的输入输出设备，可以直接获取默认的设备，也可以在程序中查找声音的输入输出设备，然后指定，我们现在直接获取默认的设备，调用QAudioDevice::defaultInputDevice()方法得到QAudioDeviceInfo的指针audioInputDevice；调用QAudioDeviceInfo::defaultOutDevice()方法得到QAudioDeviceInfo的指针audioOutputDevice.



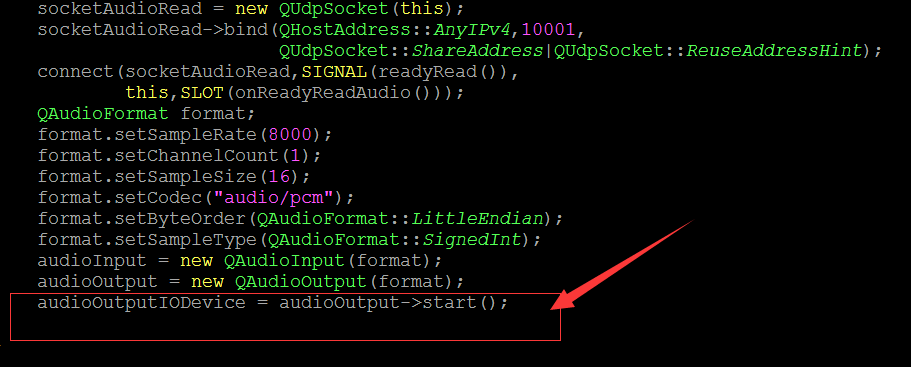
1. 设置声音QAudioFormat格式，QAudioInput和QAudioOutput对象的QAudioFormat格式应该一致，否则播放的声音与输入的声音不同。新建一个QAudioFormat对象format，采样率(SampleRate)：800；信道数(ChannelCount)：1；样本大小(SampleSize):16;编码方式(Codec):”audio/pcm”;字节序(ByteOrder):QAudioFormat::LittleEndian；采样类型(SampleType):QAudioFormat::signdInt.如下图所示：

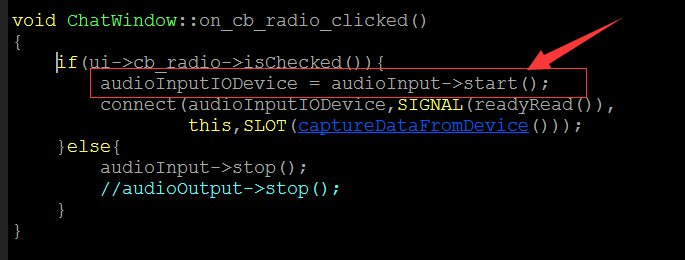


1. 分别按照上述的声音输入输出设备和声音的参数信息创建QAudioInput对象audioInput和QAudioOutput对象audioOutput。

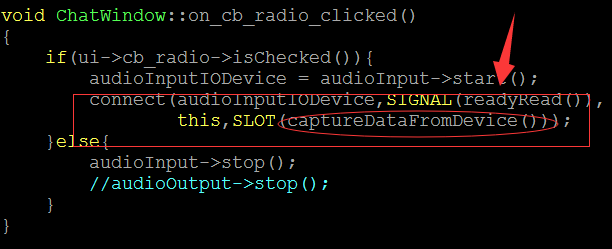


1. 调用audioInput和audioOutput的start接口，打开声音输入输出设备，将分别返回QIODevice指针audioInputIODevice和audioOutputIODevice。

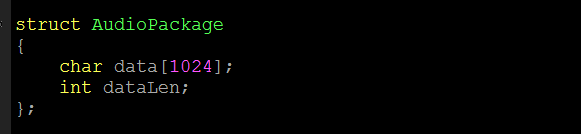


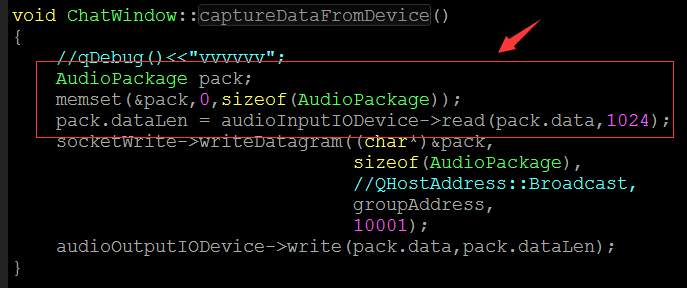


1. 当声音输入设备采集到声音数据后，将触发audioInputIODevice的readyRead()信号，所以将此信号与该类captureDataFromDevice()函数连接，即可在此函数内处理得到的原始数据。

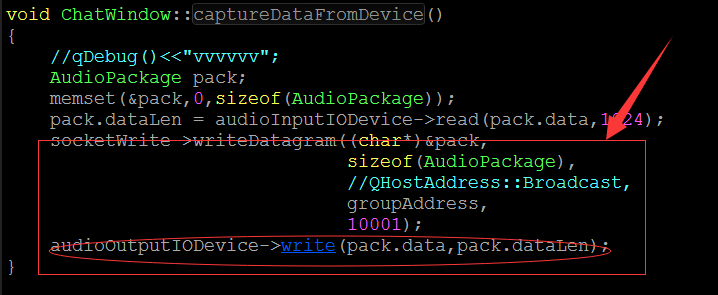


1. 在captureDataFromDevice()槽函数中调用audioInputIODevice的read函数得到原始的音频数据。如下图所示：





1. 声音播放：把得到的原始音频数据写入缓存，然后调用audioOutputDevice()的write函数把缓存的数据写入声音输出设备即可听到声音。如下图所示：



1. 原始音频数据采用的G.711编码方式进行编码，就可以获得压缩后的音频数据，即可进行网络传输。

