## SDK分类

### Mobile-sdk

用于开发手机app，实现飞行控制、数据回传、相机云台控制、图像回传、获取电池遥控器信息。



### UX SDk

与mobile sdk相似，同样用于开发app。

显示相机提要的实时视图

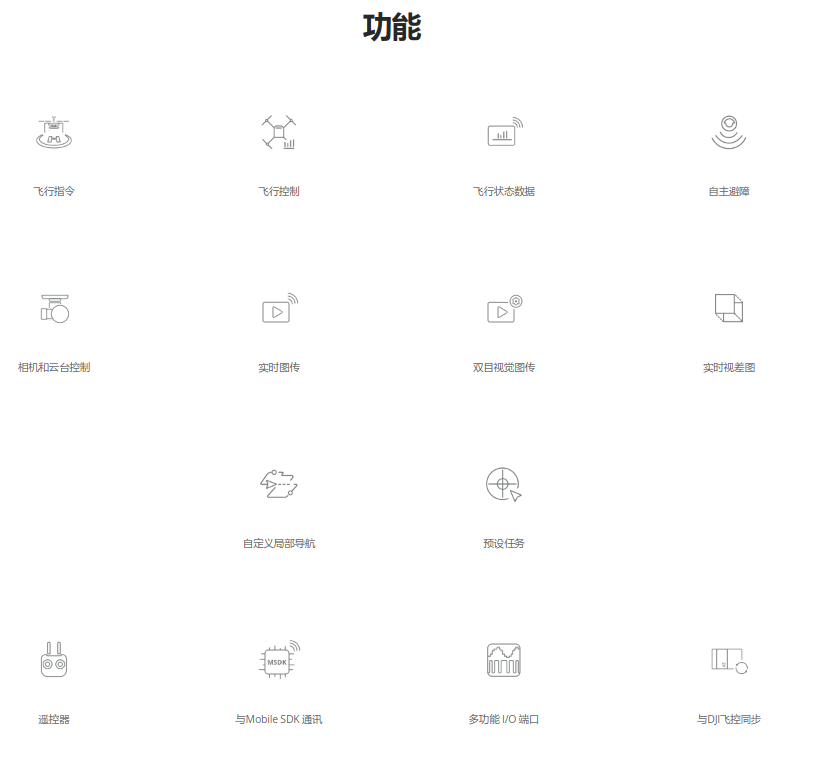
显示产品状态（飞机遥测，电池电量，信号强度等）

允许用户查看和更改产品设置

具有自动起飞，着陆，回家等基本功能。

### Onboard SDK

通过与机身上的USB或者串口相连，实现飞行控制、数据回传、云台控制、避障、飞行数据状态等等。具体如下图



使用OSDK 开发的无人机自动控制程序，借助机载计算机的强大算力，不仅能够实现对无人机以及无人机负载设备的自动控制，还能实现图像识别、物体追踪及深度感知等高级应用。

### PAYLOAD SDK

通过X-Port 标准云台和SkyPort V2转接环与机器连接，方便开发者利用DJI 无人机上如电源、通讯链路及状态信息等资源，开发出可挂载在DJI 无人机上的负载设备。主要用于相机设备，通过app控制负载设备。

Windows SDK

与mobile sdk 类似，只是跑在Windows平台上的应用，具体功能如下图所示：



## Onboard SDK使用

### 编译ffmpeg

SDK需要用到ffmpeg 所以需要下载链接 <https://github.com/FFmpeg/FFmpeg.git>

修改configure 文件

SLIBNAME\_WITH\_MAJOR='$(SLIBNAME).$(LIBMAJOR)'

LIB\_INSTALL\_EXTRA\_CMD='$$(RANLIB)"$(LIBDIR)/$(LIBNAME)"'

SLIB\_INSTALL\_NAME='$(SLIBNAME\_WITH\_VERSION)'

SLIB\_INSTALL\_LINKS='$(SLIBNAME\_WITH\_MAJOR)$(SLIBNAME)'

修改为：

SLIBNAME\_WITH\_MAJOR='$(SLIBPREF)$(FULLNAME)-$(LIBMAJOR)$(SLIBSUF)'

LIB\_INSTALL\_EXTRA\_CMD='$$(RANLIB)"$(LIBDIR)/$(LIBNAME)"'

SLIB\_INSTALL\_NAME='$(SLIBNAME\_WITH\_MAJOR)'

SLIB\_INSTALL\_LINKS='$(SLIBNAME)'

新建build.sh 填写

#!/bin/bash

make clean

./configure --target-os=linux \

--disable-doc \

--enable-shared \

--disable-static \

--disable-yasm \

--disable-symver \

--enable-gpl \

--disable-ffmpeg \

--disable-ffplay \

--disable-ffprobe \

--disable-ffserver \

--disable-doc \

--disable-symver \

--enable-cross-compile \

--extra-cflags="-Os -fpic $ADDI\_CFLAGS" \

--extra-ldflags="$ADDI\_LDFLAGS" \

$ADDITIONAL\_CONFIGURE\_FLAG

make clean

make -j4

make install

再给予这个文件可执行权限 chmod 777 build.sh

然后执行 ./build.sh 等待编译完成

提示没有权限则 执行sudo make install

### 安装 opencv

sudo apt-get install opencv

### SDK下载

进入github [GitHub - dji-sdk/Onboard-SDK at 4.1](https://github.com/dji-sdk/Onboard-SDK/tree/4.1) 建议使用代理下载。

### 编译

执行

mkdir build

cd build

cmake ../

提示报错 FFmpeg-n4.4/libavcodec/opusdec.c:345：对‘swr\_is\_initialized’未定义的引用

需要引用相关的静态库文件 找到Onboard-SDK/osdk-core/advanced-sensing/ori-osdk-core/cmake-modules/FindFFMPEG.cmake

在

SET( FFMPEG\_BASIC\_LIBRARIES

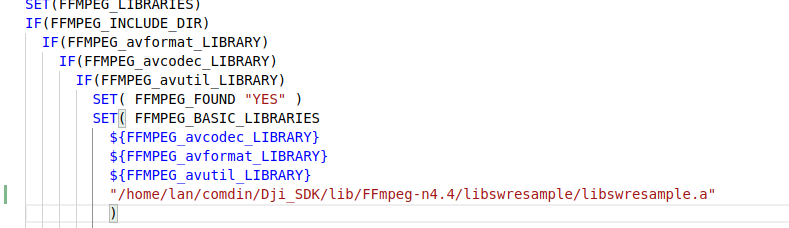
${FFMPEG\_avcodec\_LIBRARY}

${FFMPEG\_avformat\_LIBRARY}

${FFMPEG\_avutil\_LIBRARY}

)

语句下修改成



最后添加的路径为静态库文件所在的文件夹。同样需要修改Onboard-SDK/osdk-core/cmake-modules/FindFFMPEG.cmake

这两个文件是一样的，都会执行，具体使用是哪一个暂时不清楚。

再次cmake ../

然后make -j4

这样程序就能生成 例程在 build/bin下面。

### 运行例程

在大疆官网注册用户，在用户中心 创建应用程序，获取应用ID 和应用秘钥。打开

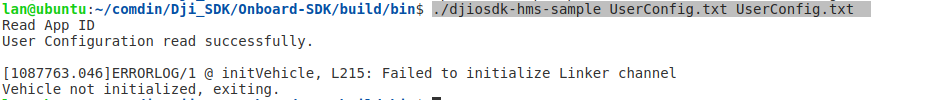
Onboard-SDK/sample/platform/linux/common/UserConfig.txt 填入id以及秘钥



复制该文件至bin下面 cp ../../sample/platform/linux/common/UserConfig.txt ./

运行例程

./djiosdk-hms-sample UserConfig.txt UserConfig.txt



这样程序就运行成功了

## 在arm上运行

### 交叉编译libusb

下载libusb https://github.com/libusb/libusb.git

解压后进入目录 执行 ./autogen.sh 时目录下生成 configure 文件 再执行

./configure --with-pic --prefix=/home/lan/comdin/Dji\_SDK/lib/libusb-1.0.24/usbtest --build=i686-linux --host=arm-linux CC= "arm-linux-gnueabihf-gcc " CXX="arm-gnueabihf-5.4.1/bin/arm-linux-gnueabihf-g++ " --prefix=/home/lan/comdin/Dji\_SDK/lib/libusb-1.0.24/usbtest --enable-shared=no --enable-static=yes --disable-udev

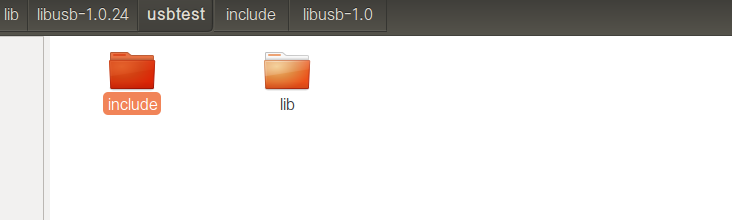
然后

make -j4

make install

等待全部执行完成之后在usbtest目录下会有相应的

库文件和头文件



拷贝至交叉编译器下的目录sudo cp usbtest/include/\* /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include

sudo cp usbtest/lib/\*.a /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/lib

### 交叉编译eudev

下载eudev <https://github.com/gentoo/eudev.git>

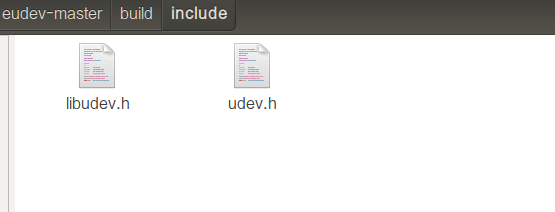
解压进入源码目录 运行 ./autogen.sh 生成configure 文件

执行 ./configure --host=arm-linux --prefix=/home/lan/comdin/Dji\_SDK/lib/eudev-master/build CC=arm-linux-gnueabihf-gcc AR=arm-linux-gnueabihf-ar --disable-blkid --disable-kmod

再make -j4

make install即可生成 libudev.a

拷贝build/include目录下的两个文件



至 交叉编译目录下/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include

具体路径由交叉编译器决定

拷贝build/lib目录下的libudev.a文件至/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/lib

### 3、交叉编译FFmpeg

进入目录执行

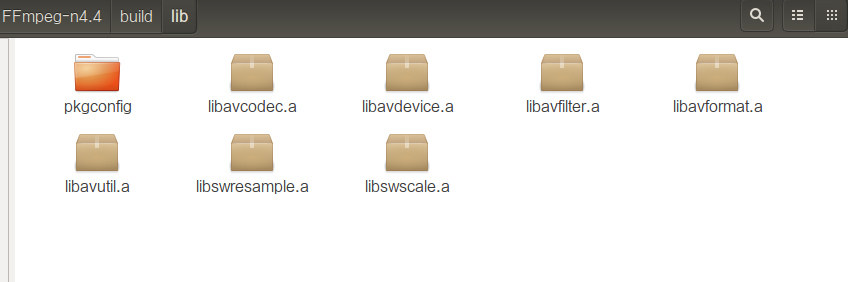
./configure --prefix=/home/lan/comdin/Dji\_SDK/lib/FFmpeg-n4.4/build cc=arm-linux-gnueabihf-gcc ar=arm-linux-gnueabihf-ar arch=arm --enable-cross-compile

修改ffbuild目录下的config.mak文件

打开文件搜索STRIP 将STRIP=strip 修改成 STRIP=arm-linux-strip保存

执行make -j4 等待编译完成之后 make install

在build文件目录下会生成我们需要的文件



拷贝文件

sudo cp -r build/include/\* /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include

sudo cp -r build/lib/\*.a /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/lib/

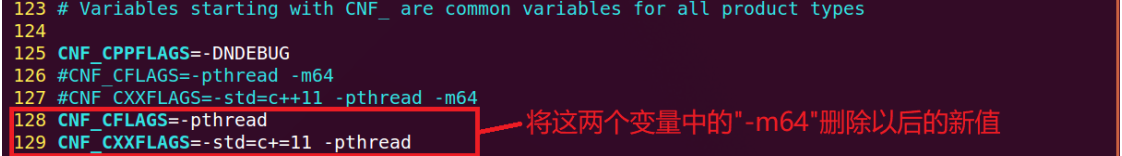
### 交叉编译openssl

Openssl3.0始终无法交叉编译成，转而使用openssl-1.1.1

/config shared no-asm --prefix=/home/lan/linux/tool/openssl

配置成功以后会生成 Makefile，打开 Makefile，找到所有包含“-m64”的内容，一共两处

分别为变量 CNF\_CFLAGS 和 CNF\_CXXFLAGS，将这两个变量中的“-m64” 删除掉.



Makefile 修改好以后使用如下命令编译并安装 libopenssl：  
make CROSS\_COMPILE=arm-linux-gnueabihf- -j12  
make install

复制库文件

lan@ubuntu:~/linux/tool/openssl$ sudo cp -r include/\* /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include

lan@ubuntu:~/linux/tool/openssl$ sudo cp -r lib/\*.a /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/lib/

### 5、交叉编译libz

下载 <https://github.com/madler/zlib/archive/refs/tags/v1.2.11.tar.gz>

执行export CC=arm-linux-gnueabihf-gcc

./configure --prefix=build

make

make install

目录build下会生成相关的库文件



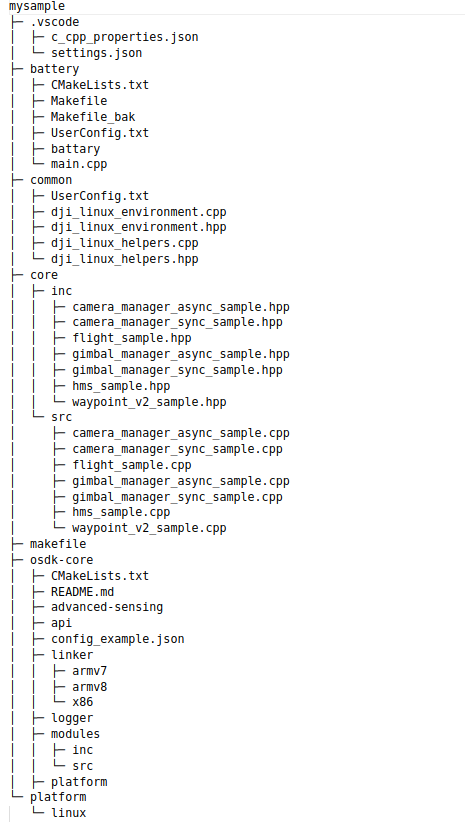
拷贝

lan@ubuntu:~/comdin/Dji\_SDK/lib/zlib-1.2.11/build$ sudo cp include/\* /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include

lan@ubuntu:~/comdin/Dji\_SDK/lib/zlib-1.2.11/build$ sudo cp lib/libz.a /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/lib

### 6、单独拿出battery例程

复制sample/lplatform/inux下的battery、common、整个platform（不包括stm32），复制整个osdk-core，复制sample下的core。最终的目录结构如下



因目录文件过多不显示一些次级文件夹。为了方便，下面这个文件就是整理过的，也有makefile，makefile需要根据情况修改引用目录，在battery下也有一个makefile，用于生成可执行文件。以及整合所有依赖。

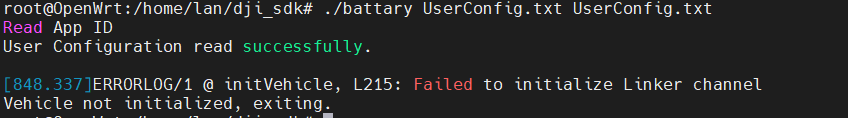


Make 再make battery 之后拷贝battery下的battery可执行文件至目标机器。

还需要拷贝一个标准的c++动态链接库libstdc++.so.6，在交叉编译工具下搜索该文件，指向libstdc++.so.6.0.20这个文件，直接拷贝这个文件至目标机器的/usr/lib下修改文件名为libstdc++.so.6。再拷贝上文提及过的UserConfig.txt文件至目标机器。修改一下执行权限并运行

chmod 777 battary

./battary UserConfig.txt UserConfig.txt



这样交叉编译就成功了，这里没有运行成功是因为没有连接usb设备。也就是没有大疆的无人机设备。

例程使用方法

编译

在主目录下进行make 注意修改主目录下的makefile 将编译器修改为需要的编译器、默认情况下是arm-linux-gnueabihf-gcc 如需在x86环境下使用程序 添加参数 TARGET\_ARCH=X86

make -j4

需要某个例程的话比如

make battery

暂时就两个例程 battery 和h264 如需增加，在platform目录下拷贝一份出来，或者不拷贝出来也可以，在该例子下从battery复制Makefile到该例子下面，进入Makefile, 129行修改一下main.o为该例子的main.o,生成文件自己命名,写法如下：

@$(CXX) ../build/Objects/mytest/main.o $(ALL\_FILES) -pthread -g -fPIC -g -o mytest

再打开主目录下的makefile，新建一个目标比如：

flight-control:

make -c flight-control

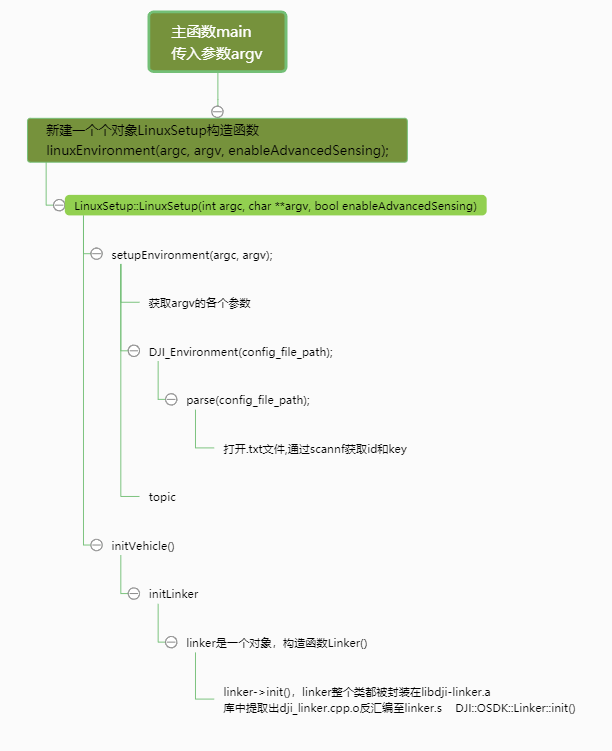
PHONY +=battery clean h264 flight-control

再执行

make flight-control

## 程序分析

### 1、启动获取ID信息



### 2、获取遥感数据

订阅一些数据需要使用调用vehicle->subscribe->initPackageFromTopicList(

pkgIndex, topicSize, topicList, enableTimestamp, freq); 对topiclist进行初始化，告诉设备需要什么数据。函数原型：

DataSubscription::initPackageFromTopicList(int packageID, int numberOfTopics,

TopicName\* topicList,

bool sendTimeStamp, uint16\_t freq)

This is the interface for the end user to generate a package for subscription.

参数:

packageID – The ID of package it'll generate

numberOfTopics:

topicList – List of Topic Names to subscribe in the package

sendTimeStamp – Note that timestamp is the time of package transmission, not data acquisition from sensor.

Freq 频率

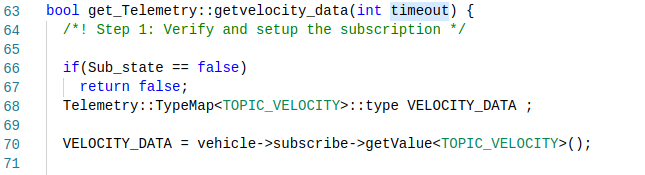
例如订阅速度

1、订阅设置

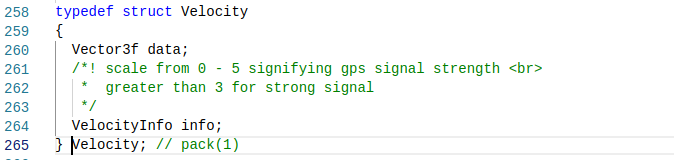


setUpSubscription的参数pkgIndex是订阅的ID，freq订阅频率，这里被设置为10，也就是每秒10次；topicList订阅类型的数组，因为可以一次订阅很多数据，所以是个数组，其中TOPIC\_VELOCITY这个宏定义就是我们需要的速度；topicSize有多少个订阅的内容；timeout超时事件。

2、获取订阅信息



68行定义一个结构体，类型为Telemetry::TypeMap<TOPIC\_VELOCITY>::type，这是模板将

VELOCITY\_DATA定义为Velocity类型

70行获取速度信息 在VELOCITY\_DATA中含有xyz三个方向的速度，需要合成才能是最终的速度。

### 3、视频数据获取

大疆的视频流传输至遥控器使用DJI FPV 数字图传系统、延迟很低。DJI FPV 数字图传系统由三部分组成遥控器、头戴显示、DJI FPV 图传天空端。这一套方案只适合控制者使用，不支持图像传输至服务器。因此要将数据传输到远程服务器则必须经过移动网络4g或者5g网络，这里的方案可采用FPV->onbord设备->4G模块->远程服务器。

FPV->onbord设备的延迟很低（官方远程FPV图传28ms[数据来源](https://www.dji.com/cn/fpv)），主要的延迟来自于4G网络。

在文档中提到：M210 系列无人机支持获取FPV 相机和I 号云台上相机的H.264 码流或RGB 图像；M300 RTK 无人机支持获取FPV 相机和所有云台相机的H.264 码流；其中的FPV 相机相机其实就是FPV图传系统中的天空端。在没有图传系统的支持下只能使用云台相机的H.264码流。

获取M210 系列和M300 RTK 无人机上相机H.264 码流的流程如下所示：

1、使用获取相机H.264 码流的功能前，请开发者根据实际的使用需要先实现liveViewSampleCb函数，用于获取并处理相机H.264 码流。

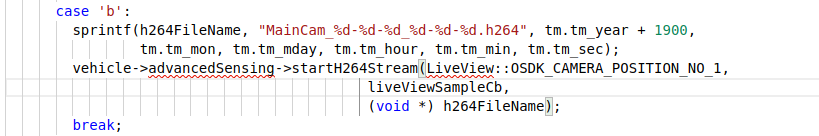
2、调用startH264Stream()接口，指定所需获取码流的相机、接收相机H.264 码流的回调函数和用户信息；

3、开启无人机和机载计算机，运行使用基于OSDK 开发的应用程序，此时无人机将会向机载计算机推送H.264 码流；

4、机载计算机接收到H.264 码流的数据后，将触发（作为入参传入开发者设置的回调函数中）基于OSDK 开发的应用程序；

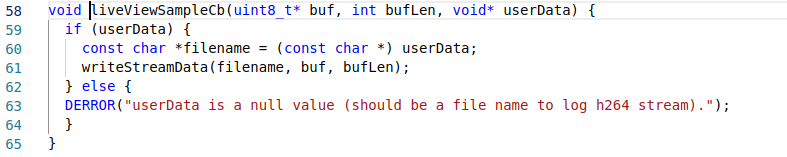
5、开发者根据实际需求设计的函数liveViewSampleCb在获取相机H.264 码流后，将对所获得的H.264 码流执行存储、解码及转发等相应的操作。

具体看下面的代码



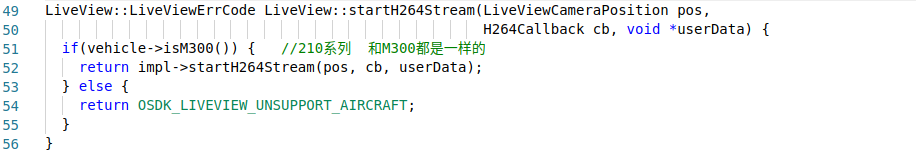
第二行以时间创建一个文件名，第四行开始获取H264码流，获取到之后执行回调函数

liveViewSampleCb函数为自定义如下：

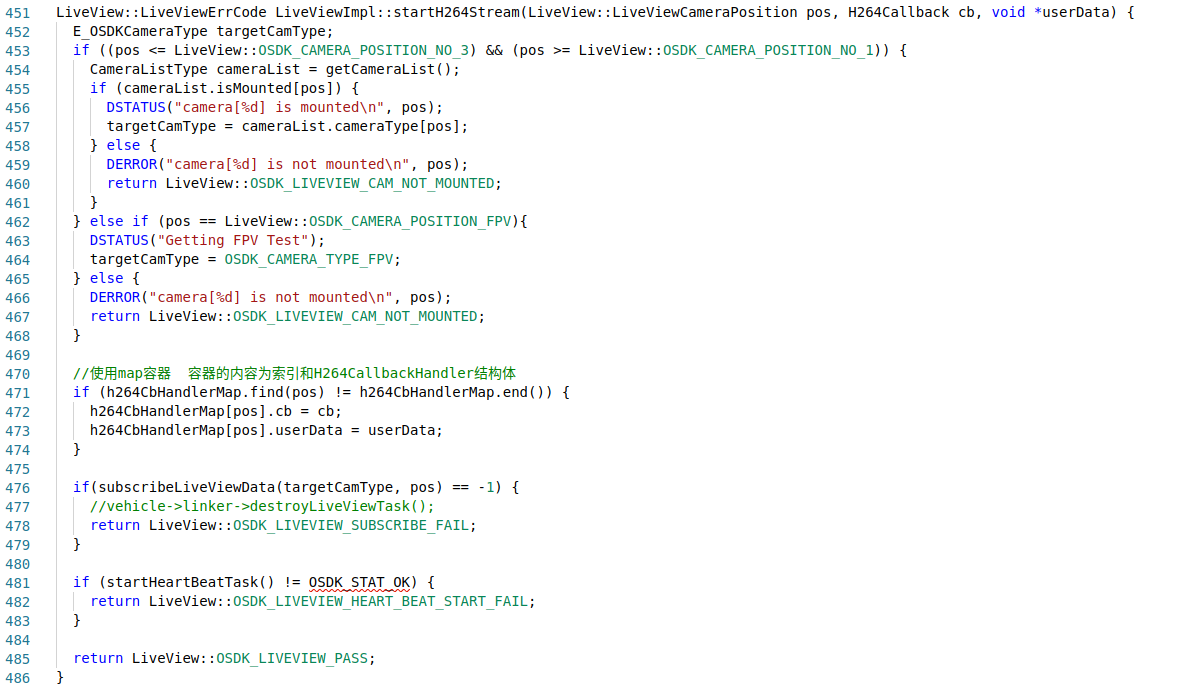


获取到之后就可以写入文件，或者直接转发。

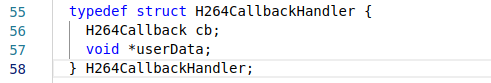
再来看startH264Stream函数



进入startH264Stream



这个函数的功能是订阅视频流、创建一个名为heartBeatTask 的一秒心跳任务。h264CbHandlerMap，这是个结构体内容为



也就是回调函数和用户传入的数据就是一开始的文件名。

在476行if(subscribeLiveViewData(targetCamType, pos) == -1) {

//vehicle->linker->destroyLiveViewTask();

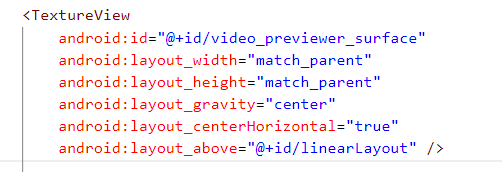
return LiveView::OSDK\_LIVEVIEW\_SUBSCRIBE\_FAIL;

}

订阅视频数据，通过pos定位是哪一个相机。

## Android 视频获取程序分析

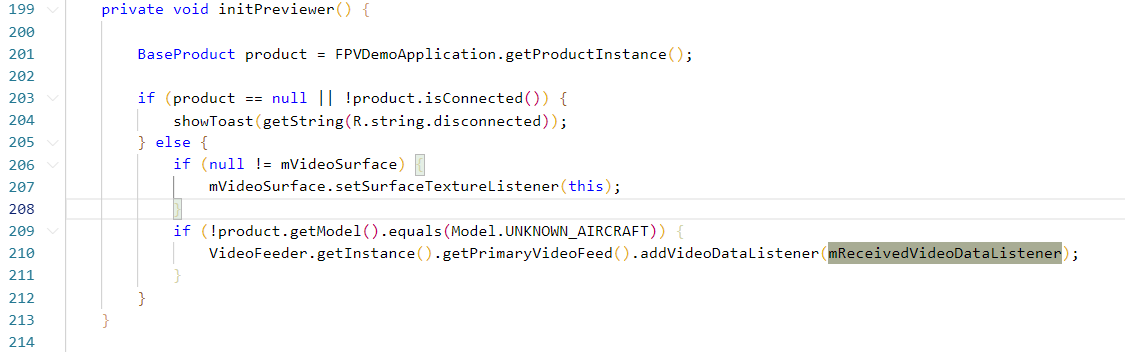
在activity\_main.xml中创建一个元素用来显示视频解码后的数据



然后获取该ui元素的id

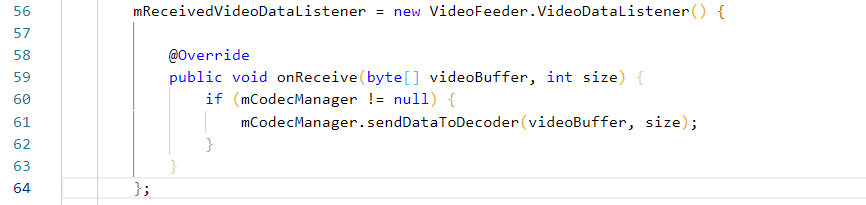
mVideoSurface = (TextureView)findViewById(R.id.video\_previewer\_surface)

设置 mReceivedVideoDataListener为大疆摄像头接口数据回调，

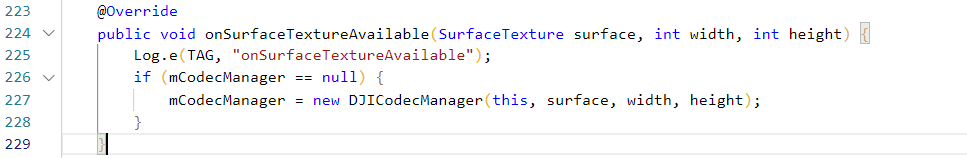


当摄像头有数据过来时将会调用mReceivedVideoDataListener中的onReceive

：



数据将会发送到mCodecManager上进行解码mCodecManager的定义如下：



解码后的数据在mCodecManager中处理，处理过程是在大疆写好的sdk中，看不到过程，处理后的视频数据放入某个源中mVideoSurface会自动获取该源更新显示，

因为MainActivity extends Activity implements SurfaceTextureListener接口因此activity中需要重写如下方法

# 

至此完成了视频数据的接收和播放，设备和手机之间采用wifi连接，wifi为无人机的热点或者遥控器的热点、属于局域网。

# 架构构思

