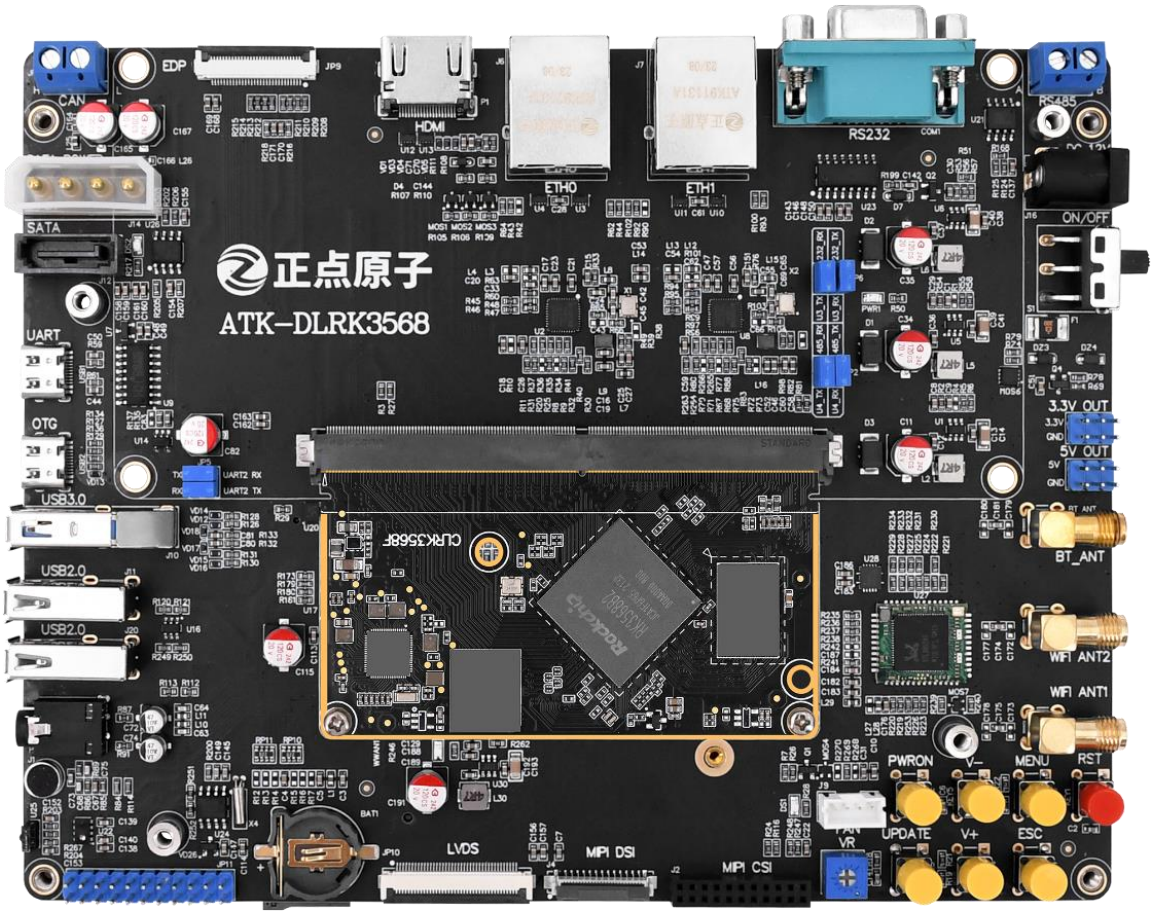
ATK-DLRK3568_RTMP 推流手册 V1.1



正点原子公司名称 ：广州市星翼电子科技有限公司
原子哥在线教学平台：www.yuanzige.com
开源电子网 / 论坛 ：http://www.openedv.com/forum.php
正点原子淘宝店铺 ：https://openedv.taobao.com
正点原子官方网站 ：www.alientek.com
正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 linux 团队	正点原子 linux 团队	2023.07.31

目录

前言

免责声明

本文档所提及的产品规格和使用说明仅供参考，如有内容更新，恕不另行通知；除非有特殊约定，本文档仅作为产品指导，所作陈述均不构成任何形式的担保。

为了得到最新版本的产品信息，请用户定时访问正点原子资料下载中心或者与淘宝正点原子旗舰店客服联系索取。感谢您的包容与支持。

第一章 RTMP 推流之视频监控

目前常见的视频监控和视频直播都是使用了 RTMP、RTSP、HLS、MPEG-DASH、 WebRTC 流媒体传输协议等。

RTSP（Real-Time Streaming Protocol）：实时流传输协议，用于控制媒体服务器上的实时流传输，支持音频和视频的传输。RTSP 常用于视频监控系统中，在客户端和服务器之间建立起媒体传输的连接和控制通道。

RTMP（Real-Time Messaging Protocol）：实时消息传输协议，最初由 Adobe 开发，用于音频、视频和数据的传输。RTMP 常用于视频直播系统中，通过将实时视频流传输到媒体服务器，实现低延迟的实时视频传输。

HLS（HTTP Live Streaming）：基于 HTTP 的流媒体传输协议，由苹果公司推出，用于将多媒体文件切片并通过 HTTP 协议进行传输。HLS 适用于各种平台和设备，并且具有自适应流媒体的特性，能够根据网络条件和设备能力进行动态调整。

MPEG-DASH（Dynamic Adaptive Streaming over HTTP）：基于 HTTP 的动态自适应流媒体传输协议，由 ISO MPEG 组织制定。MPEG-DASH 将视频和音频分段，并且根据设备和网络条件选择合适的片段进行传输，以提供更好的观看体验。

WebRTC（Web Real-Time Communication）：Web 实时通信技术，用于在 Web 浏览器之间实现实时音视频通信。WebRTC 支持点对点的传输，可以直接在浏览器中进行视频监控和视频直播，无需额外的插件或软件。

本章我们将向大家介绍如何通过 FFmpeg+Nginx、使用 RTMP 推流实现视频监控或直播。

1.1 RTMP 的工作原理

本章我们将先介绍 RTMP 的工作原理：

1. RTMP 连接建立：客户端通过 RTMP 协议与媒体服务器建立 TCP 连接，默认使用端口号 1935。建立连接后，客户端和服务器之间可以进行握手，确保双方的兼容性和通信的安全性。

2. 握手过程：RTMP 握手分为简单握手和复杂握手两个阶段。

- 简单握手：客户端发送简单握手请求给服务器，服务器返回简单握手响应。这一过程主要用于验证客户端和服务器之间的握手信息，确保双方能够正常通信。

- 复杂握手：复杂握手是通过 Diffie-Hellman 密钥交换协议进行的。客户端和服务器交换密钥信息，生成会话密钥，用于后续的数据加密与解密。

3. 媒体数据传输：握手成功后，客户端和服务器之间开始进行音频、视频和数据的实时传输。

- 发送数据：客户端将音视频数据封装成 RTMP 消息，并发送给服务器。RTMP 消息包括消息头和消息体，消息头中包含了时间戳、消息长度等信息。

- 接收数据：服务器接收 RTMP 消息，并根据消息头中的时间戳等信息进行解析和处理。服务器可以将接收到的数据保存、转发、处理等。

4. 控制命令交互：RTMP 还支持控制命令的交互，用于控制媒体传输和会话管理等。

- 客户端发送控制命令给服务器，如播放、暂停、停止等操作。

- 服务器接收控制命令，并根据命令执行相应的操作，如播放、暂停、切换等。

通过以上的步骤，RTMP 协议实现了音频、视频和数据的实时传输和交互。它在实时直播、视频聊天等场景中被广泛应用，尤其擅长处理实时性要求较高的音视频数据。

1.2 视频监控简介

本章我们将使用 RTMP 流媒体服务来实现视频监控，RTMP 流媒体服务框架图如下所示：

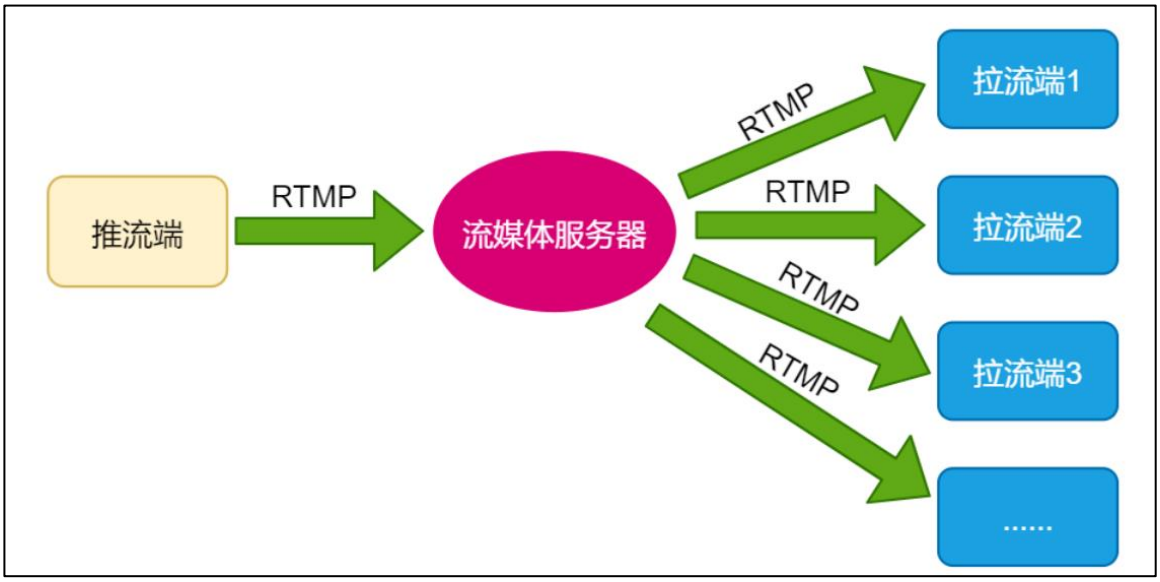


图 1.2.1 流媒体服务

推流端负责将视频数据通过 RTMP 流媒体协议传输给 RTMP 流媒体服务器，拉流端可以从流媒体服务器中通过 RTMP 协议获取到视频数据；而流媒体服务器负责接收推流端的视频数据、当有客户端（拉流端）想要获取视频数据时再将其发给相对应的客户端。

所以从上图可知，要想要实现 RTMP 视频监控，必须要有这三部分：推流客户端、流媒体服务器以及拉流客户端。那这些需要我们自己去实现吗？当然不需要，譬如推流我们可以使用 FFmpeg 来做，流媒体服务器则使用 Nginx 来搭建，而拉流则可以用 VLC 播放器来实现。

1.3 Nginx 流媒体服务器

这里我们选择在开发板搭建流媒体服务器，并且推流端也是开发板。原子团队已经把板子出厂系统移植好了 Nginx，并且板子在启动进入系统时会自动启动 Nginx，也就是启动流媒体服务器，所以板子启动之后本身就已经是一台流媒体服务器。

启动开发板，进入系统后执行 nginx -V 命令验证，如下图所示：

```
root@ATK-DLRK356X:/# nginx -V
nginx version: nginx/1.12.2
built by gcc 10.3.0 (Buildroot 2018.02-rc3-ge2c114c7)
configure arguments: --crossbuild=Linux::aarch64 --with-cc=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/
host/bin/aarch64-buildroot-linux-gnu-gcc --with-cpp=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/bi
n/aarch64-buildroot-linux-gnu-gcc --with-ld-opt= --with-ipv6 --prefix=/usr --conf-path=/etc/nginx/nginx.conf --sbin-path
=/usr/sbin/nginx --pid-path=/var/run/nginx.pid --lock-path=/var/run/lock/nginx.lock --user=www-data --group=www-data --e
rror-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --http-client-body-temp-path=/var/tmp/n
ginx/client-body --http-proxy-temp-path=/var/tmp/nginx/proxy --http-fastcgi-temp-path=/var/tmp/nginx/fastcgi --http-scgi
-temp-path=/var/tmp/nginx/scgi --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi --with-pcre --without-select_module --without
-poll_module --without-http-cache --with-cc-opt='-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 -Os'
root@ATK-DLRK356X:/#
```

图 1.3.1 查看 nginx 版本信息

从上图提供的信息，可以得知 nginx 的版本是 1.12.2，交叉编译器的版本为 10.3.0，nginx 版本的发行时间等。

执行 nginx -h 查看帮助信息：

```
root@ATK-DLRK356X:/# nginx -h
nginx version: nginx/1.12.2
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: /usr/)
  -c filename   : set configuration file (default: /etc/nginx/nginx.conf)
  -g directives : set global directives out of configuration file

root@ATK-DLRK356X:/#
```

图 1.3.2 查看 nginx 帮助信息

从上图提供的信息，可以得知：

-? 或 -h：显示帮助信息并退出。

-v 或 -V：显示 NGINX 版本信息。

-t：检查 NGINX 配置文件的语法有效性。

-T：检查 NGINX 配置文件的语法有效性，并显示配置文件的详细信息。

-q：在检查配置文件时输出简短的错误信息。

前面已经说了，出厂系统中 nginx 服务是自动开启的，可以通过 ps 命令查看到：

```
ps -aux
```

```
root       631  1.1  3.4 366720 69732 ?        Sl   16:35   0:00 /usr/bin/weston
root       632  0.0  0.1   7212  3532 ?        S    16:35   0:00 /usr/sbin/wpa_s
root       638  0.0  0.0      0     0 ?        I<   16:35   0:00 [kbase_event]
root       654  0.8  1.1  56540 23344 ?        S    16:35   0:00 /usr/libexec/we
root       655  0.7  1.4  63308 29388 ?        S    16:35   0:00 /usr/libexec/we
root       658  3.2  8.3 1103060 168832 ?      Sl   16:35   0:01 /opt/apps/1080p
root       662  0.0  0.0   2420   112 ?        Ss   16:35   0:00 /usr/sbin/droph
root       666  0.0  0.0  14144   824 ?        Ss   16:35   0:00 nginx: master p
www-data   667  0.0  0.1  14536  3028 ?        S    16:35   0:00 nginx: worker p
www-data   669  0.0  0.1  14320  2860 ?        S    16:35   0:00 nginx: cache ma
root       706  0.0  0.0 234624  1584 ?        Sl   16:35   0:00 /usr/bin/adbd
root       712  0.0  0.0      0     0 ?        I<   16:35   0:00 [kbase_event]
root       724  0.0  0.0      0     0 ?        I<   16:35   0:00 [kworker/u9:2-k
root       728  0.0  0.0      0     0 ?        S    16:35   0:00 [irq/84-dwc3]
root       732  0.0  0.0   2568   284 ?        S    16:35   0:00 /usr/sbin/ptp4l
root       735  0.0  0.0   2528   332 ?        S    16:35   0:00 /usr/sbin/phc2s
root       739  0.0  0.1   3252  2168 ?        S    16:35   0:00 /bin/sh /usr/bi
root       744  0.0  0.0   1940   104 ?        Ss   16:35   0:00 input-event-dae
root       745  0.0  0.1   6496  3112 ttyFIQ0   Ss   16:35   0:00 -/bin/sh
root       782  4.1  0.1 283752  2136 ?        Sl   16:35   0:02 rknn_server
root       807  0.0  0.0   5780  1860 ttyFIQ0   R+   16:36   0:00 ps -aux
root@ATK-DLRK356X:/#
```

图 1.3.3 查看 nginx 服务

此时我们打开电脑浏览器，输入开发板的 IP 地址，可以输入 ifconfig 查看，如下图所示：

注：笔者是开发板与电脑都是直连路由器，如果是其他连接方式，可以参考一下开发板光盘 A 盘-基础资料\10、用户手册\03、辅助文档《【正点原子】Linux 网络环境搭建手册 V1.0.pdf》

```
root@ATK-DLRK356X:/# ifconfig
eth0      Link encap:Ethernet  HWaddr B6:1E:86:01:F7:7F
          inet addr:192.168.6.64  Bcast:192.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::b41e:86ff:fe01:f77f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:313 errors:0 dropped:0 overruns:0 frame:0
          TX packets:168 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:160734 (156.9 KiB)  TX bytes:15319 (14.9 KiB)
          Interrupt:38

eth1      Link encap:Ethernet  HWaddr B2:1E:86:01:F7:7F
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:49

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:120 errors:0 dropped:0 overruns:0 frame:0
          TX packets:120 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9460 (9.2 KiB)  TX bytes:9460 (9.2 KiB)

root@ATK-DLRK356X:/#
```

图 1.3.4 开发板 IP 信息

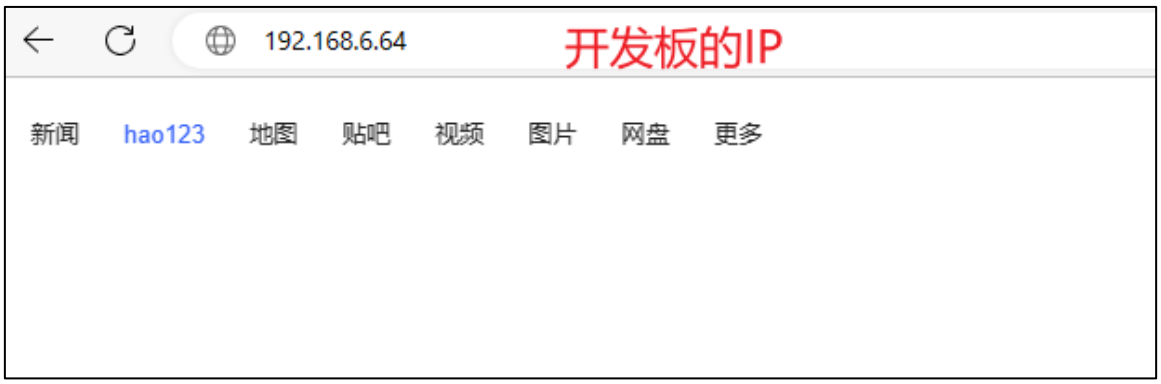


图 1.3.5 输入开发板 IP 地址

按下回车，如下所示：

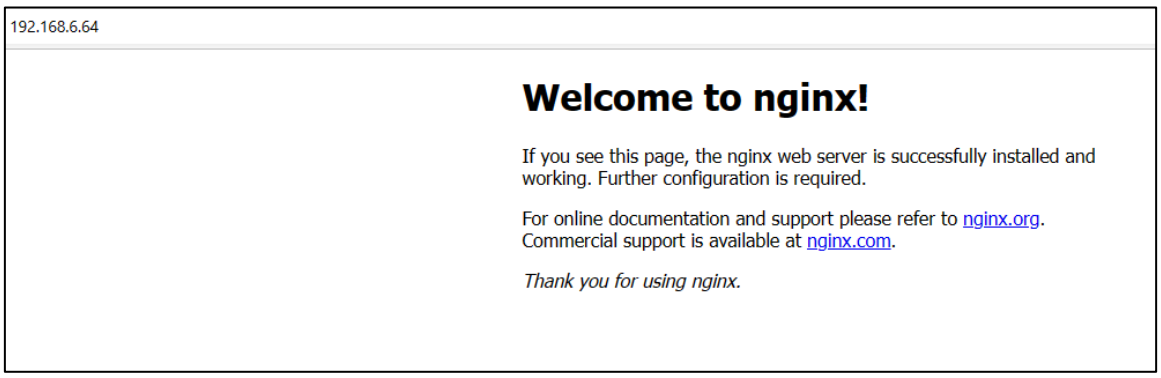


图 1.3.6 welcome to nginx

如果显示出上面的这个页面，表示我们的 nginx 已经正常工作了。

接下来，剩下最重要的一步了，就是对 nginx 进行配置，因为后续我们会使用到 FFmpeg 进行推流，将视频流通过 RTMP 推给 nginx 流媒体服务器。打开 nginx 的配置文件 /etc/nginx/nginx.conf,添加如下内容：

```
rtmp{
        server{
                listen 1935;            #监听端口，若被占用，可以修改
                chunk_size 4096;        #上传 flv 文件块大小
        application live{
                live on;                #开启 live
                hls on;                 #开启 hls
                hls_path /tmp/hls;      #rtmp 推流请求路径，文件存放路径
                hls_fragment 5s;        #每个 TS 文件包括 5s 的视频内容
        }
}
}
```

如下所示：

```
events {
    worker_connections  1024;
}

rtmp{
        server{
                listen 1935;
                chunk_size 4096;
        application live{
                live on;
                hls on;
                hls_path /tmp/hls;
                hls_fragment 5s;
        }
}
}

http {
    include       mime.types;
```

图 1.3.7 添加内容

添加完成之后保存退出并重启开发板即可！

1.4 FFmpeg 推流

接下来我们使用 FFmpeg 进行推流，将视频流数据通过 RTMP 推流给 nginx 流媒体服务器，执行如下命令进行推流：

```
ffmpeg -re -i /opt/apps/src/media/movies/梵天神器片段1080p.mp4 -c:av copy -f flv rtmp://127.0.0.1/live/mytest
```

简单地介绍一下这些参数，首先-i 表示输入视频数据，这里我们使用了一个 mp4 视频文；rtmp://127.0.0.1/live/mytest 表示将视频流通过 RTMP 推给流媒体服务器，这里因为我们的服务器和推流端都是开发板，所以这个 IP 地址 127.0.0.1 指的就是本机的流媒体服务器。

```
root@ATK-DLRK356X:/# ffmpeg -re -i /userdata/apps/src/media/movies/梵天神器片段1080p.mp4 -c:av copy -f flv rtmp://127.0.0.1/live/mytest
ffmpeg version 4.1.3 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 10.3.0 (Buildroot 2018.02-rc3-g064e70a2)
  configuration: --enable-cross-compile --cross-prefix=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/bin/aarch64-buildroot-linux-gnu- --sysroot=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/aarch64-buildroot-linux-gnu/sysroot --host-cc=/usr/bin/gcc --arch=aarch64 --target-os=linux --disable-stripping --pkg-config=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/bin/pkg-config --disable-static --enable-shared --prefix=/usr --enable-avfilter --enable-version3 --enable-logging --enable-optimizations --disable-extra-warnings --enable-avdevice --enable-avcodec --enable-avformat --enable-network --disable-gray --enable-swscale-alpha --enable-dct --enable-fft --enable-mdct --enable-rdft --disable-crystalhd --disable-dxva2 --enable-runtime-cpudetect --disable-hardcoded-tables --disable-mipsdsp --disable-mipsdspr2 --disable-msa --enable-hwaccels --disable-cuda --disable-cuvid --disable-nvenc --disable-avisynth --disable-frei0r --disable-libopencore-amrnb --disable-libopencore-amrwb --disable-libdc1394 --disable-libgsm --disable-libilbc --disable-libvo-amrwbenc --disable-symver --disable-doc --enable-gpl --enable-nonfree --disable-debug --disable-small --enable-ffmpeg --enable-ffplay --enable-avresample --disable-ffprobe --disable-postproc --enable-swscale --enable-librga --enable-indevs --enable-alsa --enable-outdevs --enable-pthreads --enable-zlib --enable-bzlib --disable-libfdk-aac --disable-libcdio --enable-gnutls --disable-openssl --enable-libdrm --disable-libopenh264 --enable-libvorbis --enable-muxer=ogg --enable-encoder=libvorbis --disable-vaapi --disable-vdpau --enable-rkmpp --enable-libdrm --disable-decoder=h264_v4l2m2m --disable-decoder=vp8_v4l2m2m --disable-decoder=mpeg2_v4l2m2m --disable-decoder=mpeg4_v4l2m2m --disable-encoder=h264_v4l2m2m --disable-encoder=vp8_v4l2m2m --disable-encoder=mpeg2_v4l2m2m --disable-encoder=mpeg4_v4l2m2m --disable-mmal --disable-omx --disable-omx-rpi --disable-libopencv --disable-libopus --disable-libvpx --disable-libass --disable-libbluray --disable-libmfx --enable-librtmp --enable-libmp3lame --disable-libmodplug --disable-libspeex --enable-libtheora --disable-libwavpack --disable-iconv --enable-libfreetype --enable-fontconfig --disable-libopenjpeg --enable-libx264 --enable-libx265 --disable-x86asm --disable-mmx --disable-sse --disable-sse2 --disable-sse3 --disable-ssse3 --disable-sse4 --disable-sse42 --disable-avx --disable-avx2 --disable-armv6 --disable-armv6t2 --enable-vfp --enable-neon --disable-altivec --extra-libs=-latomic --enable-pic --cpu=cortex-a55
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
[mov,mp4,m4a,3gp,3g2,mj2 @ 0x31a9e970] stream 0, timescale not set
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '/userdata/apps/src/media/movies/梵天神器片段1080p.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
```

图 1.4.1 FFmpeg 推流

由上图得知，FFmpeg 已经成功推流了，现在我们可以进行拉流了，可以将我们的 Windows 主机作为拉流端，使用 VLC 软件进行拉流，VLC 软件可以到开发板光盘 A 盘-基础资料\04、软件目录下找到，大家可以自行安装好。安装好之后打开 VLC，如下图所示：

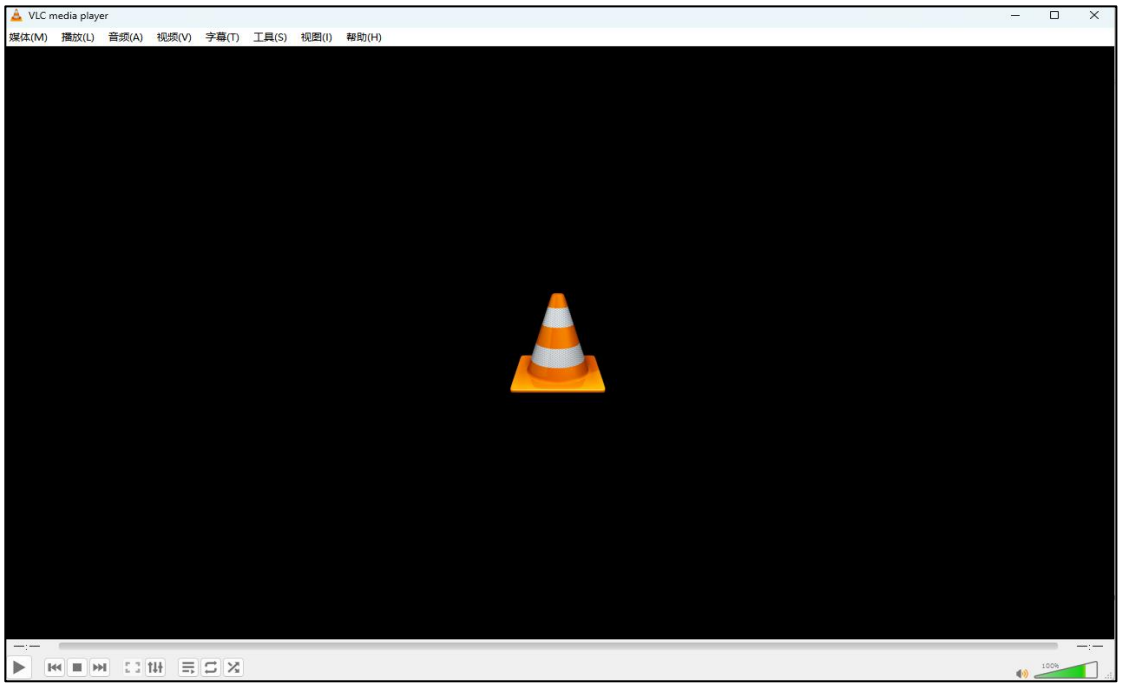


图 1.4.2 VLC 软件

点击左上<媒体> → <打开网络串流> 如下图所示：

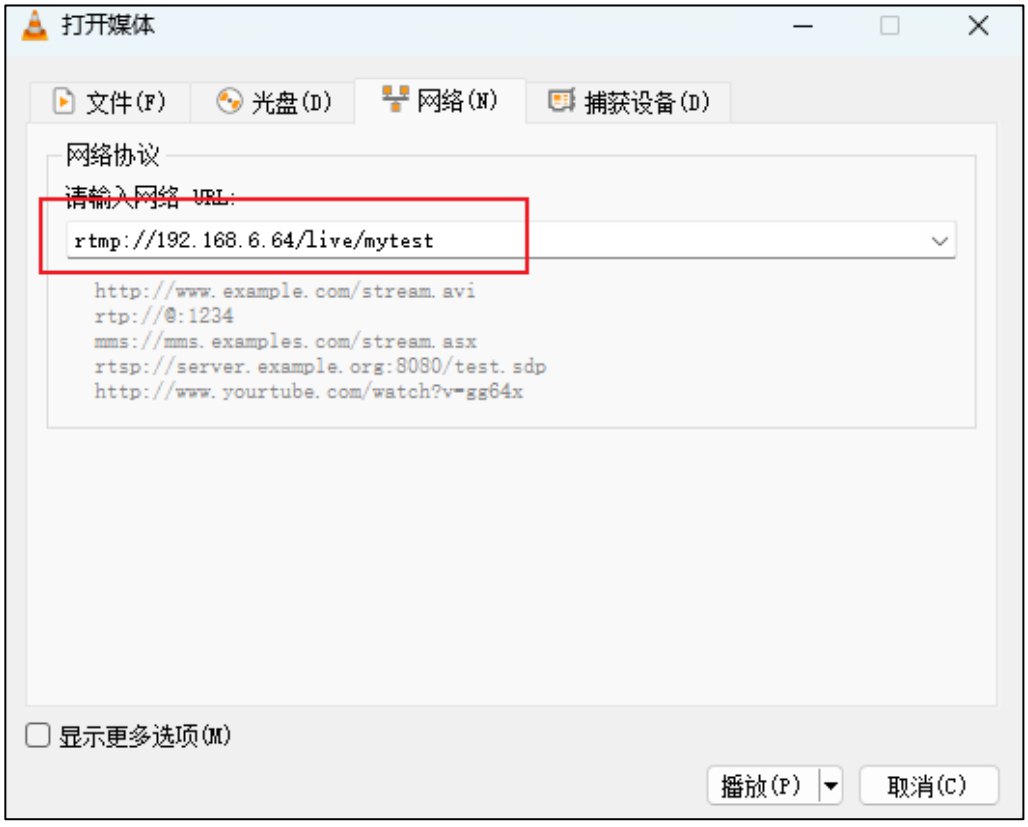


图 1.4.3 网络串流

输入流媒体服务器的 IP 地址以及路径，笔记使用的开发板的 IP 地址为 192.168.6.64（图 1.3.4），点击<播放>既可从 RTMP 流媒体服务器拉取视频数据进行播放，如下所示：

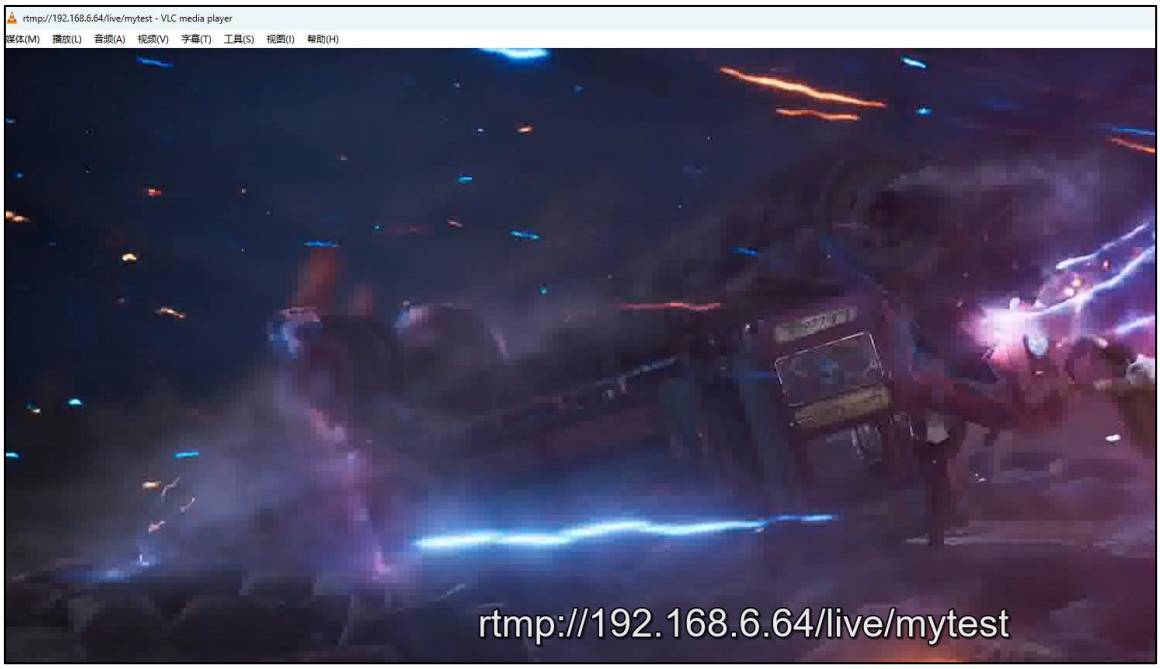


图 1.4.4 VLC 拉流播放视频

视频播放端卡顿和画质低是属于正常现象，这个主要是因为计算解码时间过长，网络波动等问题导致音视频卡顿。这个就需要专业的音视频人员去设置合理的配置参数，合理的帧率、码率、优化编解码算法等方法。

接下来我们再使用 USB 摄像头进行测试命令执行之后，接着在 Windows 下使用 VLC 拉流播放摄像头采集到的画面，实现视频摄像头监控。有读者可能会问，为什么不使用 MIPI 摄像头，这里笔者初步的判断因为 FFmpeg 不支持 MIPI 协议。

使用 FFMpeg 采集摄像头视频数据将其发送给 nginx 流媒体服务器，执行以下命令：

```
ffmpeg -f v4l2 -video_size 640x480 -framerate 30 -i /dev/video9 -q 10 -f flv rtmp://127.0.0.1/live/mytest
```

```
root@ATK-DLRK356X:/# ffmpeg -f v4l2 -video_size 640x480 -framerate 30 -i /dev/video9 -q 10 -f flv rtmp://127.0.0.1/live/
mytest
ffmpeg version 4.1.3 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 10.3.0 (Buildroot 2018.02-rc3-g064e70a2)
  configuration: --enable-cross-compile --cross-prefix=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host
/bin/aarch64-buildroot-linux-gnu- --sysroot=/home/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/aarch64-bu
ildroot-linux-gnu/sysroot --host-cc=/usr/bin/gcc --arch=aarch64 --target-os=linux --disable-stripping --pkg-config=/home
/alientek/ATK-DLRK3568/buildroot/output/rockchip_rk3568/host/bin/pkg-config --disable-static --enable-shared --prefix=/u
sr --enable-avfilter --enable-version3 --enable-logging --enable-optimizations --disable-extra-warnings --enable-avdevic
e --enable-avcodec --enable-avformat --enable-network --disable-gray --enable-swscale-alpha --enable-dct --enable-fft --
enable-mdct --enable-rdft --disable-crystalhd --disable-dxva2 --enable-runtime-cpudetect --disable-hardcoded-tables --di
sable-mipsdsp --disable-mipsdspr2 --disable-msa --enable-hwaccels --disable-cuda --disable-cuvid --disable-nvenc --disab
le-avisynth --disable-frei0r --disable-libopencore-amrnb --disable-libopencore-amrwb --disable-libdc1394 --disable-libgs
m --disable-libilbc --disable-libvo-amrwbenc --disable-symver --disable-doc --enable-gpl --enable-nonfree --disable-debu
g --disable-small --enable-ffmpeg --enable-ffplay --enable-avresample --disable-ffprobe --disable-postproc --enable-swsc
ale --enable-librga --enable-indevs --enable-alsa --enable-outdevs --enable-pthreads --enable-zlib --enable-bzlib --disa
ble-libfdk-aac --disable-libcdio --enable-gnutls --disable-openssl --enable-libdrm --disable-libopenh264 --enable-libvor
bis --enable-muxer=ogg --enable-encoder=libvorbis --disable-vaapi --disable-vdpau --enable-rkmpp --enable-libdrm --disab
le-decoder=h264_v4l2m2m --disable-decoder=vp8_v4l2m2m --disable-decoder=mpeg2_v4l2m2m --disable-decoder=mpeg4_v4l2m2m --
disable-encoder=h264_v4l2m2m --disable-encoder=vp8_v4l2m2m --disable-encoder=mpeg2_v4l2m2m --disable-encoder=mpeg4_v4l2m
2m --disable-mmal --disable-omx --disable-omx-rpi --disable-libopencv --disable-libopus --disable-libvpx --disable-libas
s --disable-libbluray --disable-libmfx --enable-librtmp --enable-libmp3lame --disable-libmodplug --disable-libspeex --en
able-libtheora --disable-libwavpack --disable-iconv --enable-libfreetype --enable-fontconfig --disable-libopenjpeg --ena
ble-libx264 --enable-libx265 --disable-x86asm --disable-mmx --disable-sse --disable-sse2 --disable-sse3 --disable-ssse3
--disable-sse4 --disable-sse42 --disable-avx --disable-avx2 --disable-armv6 --disable-armv6t2 --enable-vfp --enable-neon
 --disable-altivec --extra-libs=-latomic --enable-pic --cpu=cortex-a55
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
Input #0, video4linux2,v4l2, from '/dev/video9':
  Duration: N/A, start: 1943.414466, bitrate: 147456 kb/s
    Stream #0:0: Video: rawvideo (YUY2 / 0x32595559), yuyv422, 640x480, 147456 kb/s, 30 fps, 30 tbr, 1000k tbn, 1000k tb
c
Stream mapping:
  Stream #0:0 -> #0:0 (rawvideo (native) -> flv1 (flv))
Press [q] to stop, [?] for help
Output #0, flv, to 'rtmp://127.0.0.1/live/mytest':
  Metadata:
    encoder         : Lavf58.20.100
    Stream #0:0: Video: flv1 (flv) ([2][0][0][0] / 0x0002), yuv420p, 640x480, q=2-31, 200 kb/s, 30 fps, 1k tbn, 30 tbc
    Metadata:
      encoder         : Lavc58.35.100 flv
    Side data:
      cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: -1
frame=  108 fps= 30 q=10.0 size=     325kB time=00:00:03.60 bitrate= 739.5kbits/s speed=1.01x
```

图 1.4.5 摄像头监控

图 1.4.6 VLC 拉流播放摄像头采集到的画面

测试发现，延迟太高了，导致开发板当前采集的画面与 VLC 播放到的画面并不同步，笔者实测大概有 5、6 秒的延时，笔者认为是 FFmpeg 内部进行了很多处理，譬如对视频、音视频的处理、算法的处理，导致会耗费相当大的时间。

本章的内容就到此结束