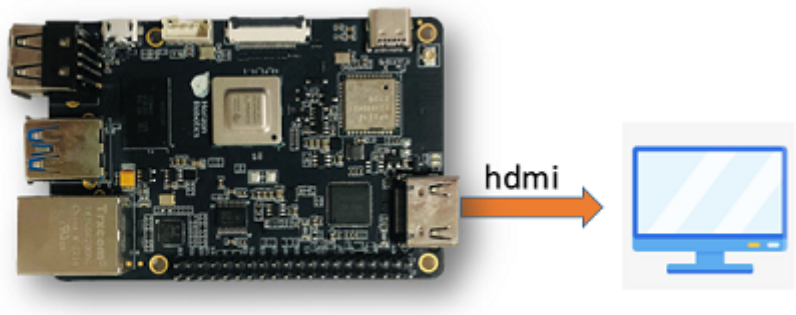
RTSP streaming decoding

This example `rtsp2display` implements the function of pulling `rtsp` code stream, decoding, and outputting video images through `HDMI`. Users can preview the screen through the display.

The example flow chart is as follows.



- **Environment preparation:**
  - Connect the development board and the monitor via an HDMI cable
  - Power on the development board and log in via the command line
  - Prepare the `rtsp` stream as the input source and use the system's preset streaming service. This service will process the `1080P_test.h264` video file into an rtsp stream with the URL `rtsp://127.0.0.1/1080P_test.h264`. Users can start the streaming service with the following command:

```
cd /app/pydev_demo/08_decode_rtsp_stream/
root@ubuntu:/app/pydev_demo/08_decode_rtsp_stream# sudo ./live555MediaServer
&
```

- **How to run:** The sample code is provided in source code form and needs to be compiled and run using the `make` command.

  The steps are as follows:

```
sunrise@ubuntu:~$ cd /app/cdev_demo/rtsp2display
sunrise@ubuntu:/app/cdev_demo/rtsp2display$ sudo make #Some warning messages
may be printed, ignore them
sunrise@ubuntu:/app/cdev_demo/decode2display$ sudo ./rtsp2display -i
rtsp://127.0.0.1/1080P_test.h264 -t tcp
```

Parameter configuration:

- -i: stream url address
- -t: transmission type, optional tcp / udp
- **Expected effect:** After the program runs correctly, the video screen will be output through the `HDMI` interface of the development board, and the user can preview the video screen through the display. The running log is as follows.

```
sunrise@ubuntu:/app/cdev_demo/rtsp2display$ sudo ./rtsp2display -i
rtsp://127.0.0.1/1080P_test.h264 -t tcp
avformat_open_input ok!
avformat_find_stream_info ok!
Input #0, rtsp, from 'rtsp://127.0.0.1/1080P_test.h264':
  Metadata:
    title           : H.264 Video, streamed by the LIVE555 Media Server
    comment         : 1080P_test.h264
  Duration: N/A, start: 0.040000, bitrate: N/A
  Stream #0:0: Video: h264 (High), yuv420p(progressive), 1920x1080 [SAR 1:1
DAR 16:9], 25 fps, 25 tbr, 90k tbn, 50 tbc
av_dump_format ok!
rtsp_w:1920,rtsp_h:1080
display_w:1920,dispaly_h:1080
... omission ...
sp_open_vps success!
Created new framebuffer: fb_id=76 for dma_buf_fd=18
```

Notes:

- When using the UDP protocol to transmit the code stream, the screen may be distorted due to network packet loss. In this case, you can switch to TCP protocol transmission to solve the problem.