Kamera-Live-Streaming mit HLS/DASH

**Das HLS- und das DASH-Protokoll werden zum Streamen von Videos im Internet verwendet und ersetzen das alte RTMP-Protokoll. Diese Methoden übertragen ein Video in kleinen Stücken mit unterschiedlichen Qualitätsstufen über Standard-HTTP-Webserver. Dieser Beitrag zeigt, wie man FFmpeg verwendet, um eine Wiedergabeliste von Videoblöcken von einer Kamera zu generieren und einen Live-Stream von einem einfachen Webserver bereitzustellen, der auf Apache, NGINX oder Python HTTP läuft. Die Leistung und Verzögerung von HLS/DASH-Streaming werden ebenfalls gemessen.**

[#raspberry-pi](https://www.codeinsideout.com/tags/#raspberry-pi)[#camera](https://www.codeinsideout.com/tags/#camera)[#stream](https://www.codeinsideout.com/tags/#stream)[#hls](https://www.codeinsideout.com/tags/#hls)[#dash](https://www.codeinsideout.com/tags/#dash)[#python](https://www.codeinsideout.com/tags/#python)

Letzte Aktualisierung: 04.11.2021

[stream\_ffmpeg\_hls](https://github.com/vuquangtrong/pi_streaming)

Verzögerung beim Streaming im HLS-Protokoll

[Big Buck Bunny-Film](https://peach.blender.org/) , © 2008, Blender Foundation

1. Installieren Sie FFmpeg[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/" \l "1-install-ffmpeg" \o "Permanenter Link)

Installieren Sie FFmpeg aus dem Paketmanager:

sudo apt install -y ffmpeg

Oder kompilieren Sie eine ausführbare Datei, indem Sie der Anleitung zum [manuellen Kompilieren von FFmpeg folgen.](https://www.codeinsideout.com/blog/pi/compile-ffmpeg/)

2. HLS vs. MPEG-DASH[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#2-hls-vs-mpeg-dash)

Ein Streaming-Protokoll ist eine Art von Technologie, die entwickelt wurde, um Videodateien über das Internet zu transportieren. In der Vergangenheit wurden Online-Videos hauptsächlich über das RTMP-Protokoll bereitgestellt. RTMP ist ein Flash-basierter Standard, der auch heute noch zum Senden von Videos von einem Encoder an eine Online-Videoplattform verwendet wird. Das RTMP wurde langsam durch das HLS- und MPEG-DASH-Protokoll ersetzt.

2.1. HLS[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#21-hls)

HLS ist die Abkürzung für *HTTP Live Streaming* . Ursprünglich von Apple entwickelt, bestand der Zweck von HLS darin, das iPhone in die Lage zu versetzen, auf Livestreams zuzugreifen. HLS kann Videos wiedergeben, die mit den Codecs H.264 oder HEVC/H.265 codiert sind.

Wie der Name schon sagt, liefert HLS Inhalte über Standard-HTTP-Webserver. Das bedeutet, dass für die Bereitstellung von HLS-Inhalten keine spezielle Infrastruktur benötigt wird. Jeder Standard-Webserver oder CDN funktioniert. Darüber hinaus ist es mit diesem Protokoll weniger wahrscheinlich, dass Inhalte von Firewalls blockiert werden, was ein Plus ist.

Wie es funktioniert, wird das Video in 10-Sekunden-Segmente zerhackt. Die Latenz für die Zustellung liegt in der Regel im Bereich von 45 Sekunden. Mit einigen angewendeten Einstellungen kann die Verzögerung auf 3-5 Sekunden reduziert werden. Dieses Protokoll enthält auch mehrere andere integrierte Funktionen. Beispielsweise ist HLS ein Protokoll mit adaptiver Bitrate. Das bedeutet, dass das Clientgerät und der Server die Internetgeschwindigkeit des Benutzers dynamisch erkennen und die Videoqualität entsprechend anpassen.

2.2. MPEG-DASH[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#22-mpeg-dash)

Als neuerer Standard ist MPEG-DASH ein aufstrebender Konkurrent von HLS. Dieses Protokoll wurde als Reaktion auf die Fragmentierung des Video-Streaming-Marktes entwickelt. Zu dieser Zeit konkurrierte Apples HLS mit mehreren anderen Streaming-Protokollen. Das Ergebnis war ungewiss, was Standardorganisationen dazu veranlasste, MPEG-DASH als alternatives, vereinheitlichendes Streaming-Protokoll zu entwickeln.

MPEG-DASH ist ein Open-Source-Standard. Wie das HLS-Streaming-Protokoll ist MPEG-DASH ein Videoverfahren mit adaptiver Bitrate. Es unterstützt auch Werbung, und die Technologie dafür entwickelt sich schnell weiter. MPEG-DASH wird jedoch nicht vom mobilen Safari-Browser unterstützt. HLS ist einfach viel umfassender kompatibel als MPEG-DASH.

3. Webserver einrichten[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#3-setup-web-server)

Da HLS und MPEG-DASH HTTP-basierte Protokolle sind, muss kein spezieller Webserver installiert werden, sondern lediglich ein einfacher Webserver, der Video-Chunk-Dateien bereitstellen kann.

3.1. Apache[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#31-apache)

Apache ist eine beliebte Webserver-Anwendung, die auf dem Raspberry Pi installiert werden kann, damit er Webseiten bereitstellen kann. Apache allein kann HTML-Dateien über HTTP bereitstellen und mit zusätzlichen Modulen dynamische Webseiten mithilfe von Skriptsprachen wie PHP bereitstellen.

Apaches Designarchitektur:

* Prozessgesteuerter Ansatz
* Erstellt für jede Anfrage einen neuen Thread.

sudo apt install -y apache2

Standardmäßig legt Apache eine Test-HTML-Datei im Webordner ab /var/www/html/. Diese Standardwebseite wird unter [http://localhost](http://localhost/) oder auf der IP-Adresse des Pi selbst bereitgestellt.

3.2. NGINX[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#32-nginx)

NGINX (ausgesprochen *engine x* ) ist eine beliebte leichte Webserveranwendung, die auf dem Raspberry Pi installiert werden kann, damit er Webseiten bereitstellen kann. Wie Apache kann NGINX HTML-Dateien über HTTP bereitstellen und mit zusätzlichen Modulen dynamische Webseiten mit Skriptsprachen wie PHP bereitstellen.

Die Designarchitektur von NGINX:

* Ereignisgesteuerter Ansatz
* Verarbeitet mehrere Anfragen innerhalb eines Threads

Nginx kann mit begrenzten Ressourcen sehr schnell und umfassend arbeiten.

sudo apt install -y nginx

Ähnlich wie Apache bedient NGINX auch Webseiten in /var/www/html/. Gehen Sie zu <http://localhost/etc/nginx/sites-available> , um die Links der Website anzuzeigen.

Wie der Artikel [Apache Vs NGINX – Welcher ist der beste Webserver?](https://serverguy.com/comparison/apache-vs-nginx)Wie bereits erwähnt, sollte es mit NGINX gehen, wenn statische Webseiten mit hohem Datenverkehr (Anfragen) bereitgestellt werden.

4. Live-Video streamen[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#4-stream-live-video)

MPEG-DASH und HLS erstellen beide Wiedergabelistendateien, deren Inhalt eine Liste von Videoblöcken ist. ffmpegkann von der Kamera lesen und Videoblöcke schreiben sowie die Wiedergabeliste aktualisieren.

Um die SD-Karte zu beschleunigen und zu schützen, ist es besser, Videoblöcke in den RAM-Speicher zu schreiben. Wenn die Anzahl der Chunks hoch wird, löschen Sie die alten, um mehr Platz zu bekommen.

Machen wir es zuerst mit HLS!

4.1. Erstellen Sie Videoblöcke[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#41-create-video-chunks)

Erstellen Sie einen neuen Ordner im gemeinsamen Speicher:

mkdir -p /dev/shm/hls

und erstellen Sie einen Softlink zum Webordner:

ln -s /dev/shm/hls /var/www/html/hls

ffmpegZum Erstellen einer HLS-Wiedergabeliste verwenden :

* -input\_format h264 -i /dev/video0: Eingabe von /dev/video0(Pi-Kamera) mit V4L2 H264-Format (weitere Informationen finden Sie unter [V4L2 H264-Direktkopie](https://www.codeinsideout.com/blog/pi/set-up-camera/#v4l2-h264-direct-copy-mp4) )
* -c:v copy: H264-Video direkt vom V4L2-Treiber verwenden
* -f hls: Ausgabe im HLS-Format
* -hls\_time 1: Videoblöcke werden in 1-Sekunden-Segmenten gespeichert
* -hls\_list\_size 30: Wiedergabeliste hat 30 Segmente
* -hls\_flags delete\_segments: Segmente löschen, die nicht in der Wiedergabeliste enthalten sind
* /dev/shm/hls/live.m3u8: Der Speicherort der Playlist-Datei und der Videosegmente

ffmpeg -y \

-input\_format h264 -i /dev/video0 \

-c:v copy \

-f hls \

-hls\_time 1 \

-hls\_list\_size 30 \

-hls\_flags delete\_segments \

/dev/shm/hls/live.m3u8

4.2. Verwenden Sie HLS-Streaming[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#42-use-hls-streaming)

Ein JavaScript namens [hls.js](https://github.com/video-dev/hls.js) wird zum Abspielen des HLS-Streams verwendet.

/var/www/html/hls.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>HLS Live Stream</title>

</head>

<body>

<h1>HLS Live Stream</h1>

<script src="hls.js"></script>

<video id="video" controls autoplay></video>

<script>

var video = document.getElementById("video");

var videoSrc = "hls/live.m3u8";

// First check for native browser HLS support

if (video.canPlayType("application/vnd.apple.mpegurl")) {

video.src = videoSrc;

}

// If no native HLS support, check if hls.js is supported

else if (Hls.isSupported()) {

var hls = new Hls();

hls.loadSource(videoSrc);

hls.attachMedia(video);

}

</script>

</body>

</html>

**HLS-Leistung**

HLS lässt sich gut über HTTP streamen, hat aber eine große Verzögerung. Bei der Auflösung 1024x768 @ 30fps, 1-Sekunden-Segmente, zeigt es immer noch aVerzögerung von ~10 Sekunden.

Die HLS-Testseite <https://hls-js.netlify.app/demo> kann die Leistung einer HLS-Wiedergabeliste messen. Wenn Sie einen lokalen Server testen, muss er CORS in den Servereinstellungen aktivieren oder die CORS-Entsperrungserweiterung verwenden.

4.3. Verwenden Sie MPEG-DASH-Streaming[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#43-use-mpeg-dash-streaming)

DASH ist dasselbe wie HLS, der Unterschied liegt im Playlist-Format und im Container der Segmente.

Erstellen Sie einen neuen Ordner im gemeinsamen Speicher:

mkdir -p /dev/shm/dash

und erstellen Sie einen Softlink zum Webordner:

ln -s /dev/shm/dash /var/www/html/dash

ffmpegZum Erstellen einer DASH-Wiedergabeliste verwenden :

* -f dash: DASH-Format
* -seg\_duration 1: Segmentgröße ist 1 Sekunde
* -streaming 1: Streaming aktiviert
* -window\_size 30 -remove\_at\_exit 1: Playlist hat 30 Chunks, Chunks beim Beenden löschen
* /dev/shm/dash/live.mpd: Playlist-Datei

ffmpeg -y \

-input\_format h264 \

-i /dev/video0 \

-c:v copy \

-f dash \

-seg\_duration 1 \

-streaming 1 \

-window\_size 30 -remove\_at\_exit 1 \

/dev/shm/dash/live.mpd

Wechseln Sie schließlich zu [Dash.js](https://github.com/Dash-Industry-Forum/dash.js) , um den DASH-Stream abzuspielen:

/var/www/html/dash.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>MPEG-DASH Live Stream</title>

</head>

<body>

<h1>MPEG-DASH Live Stream</h1>

<script src="dash.all.min.js"></script>

<video id="videoPlayer" controls></video>

<script>

(function () {

var url = "dash/live.mpd";

var player = dashjs.MediaPlayer().create();

player.initialize(

document.querySelector("#videoPlayer"),

url,

true

);

player.updateSettings({

streaming: {

lowLatencyEnabled: true,

liveDelay: 2,

liveCatchup: {

minDrift: 0.05,

playbackRate: 1,

latencyThreshold: 30,

},

},

});

})();

</script>

</body>

</html>

**MPEG-DASH-Leistung**

MPEG-DASH erreichen kann~3 Sekunden Verzögerung, was viel besser ist als HLS. Von einem Echtzeit-Livestream ist man allerdings noch weit entfernt.

**Geringe Wartezeit**

HLS und MPEG-DASH unterstützen Low-Latency-Streaming, müssen aber in Server und Encoder konfiguriert werden. Dieses Thema wird später behandelt.

5. Python-Webserver[⚓︎](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/#5-python-web-server)

Der Python-HTTP-Server http.server.SimpleHTTPRequestHandlerkann auch HLS-Dateien streamen. Hier sind die notwendigen Dateien aufgelistet, um den HLS-Streaming-Server mit Python auszuführen:

**index.html**

Verwenden Sie [hls.js](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/stream_ffmpeg_hls/hls.js) , um HLS Stream abzuspielen. Es gibt zusätzliche HLS-Konfigurationen: js var config = Hls.DefaultConfig; config.liveSyncDurationCount = 1; config.startFragPrefetch = true; console.log(config); var hls = new Hls(config);

**hls.js**

HLS-Stream-Player, geschrieben in JavaScript für das Web

**server.py**

Dies implementiert einen einfachen HTTP-Request-Handler basierend aufSimpleHTTPRequestHandler

**run.sh**

Dieses Skript erstellt einen temporären Ordner im gemeinsamen Speicher, um Videosegmente zu speichern. Dann wird es ausgeführt ffmpeg, um die Kamera zu lesen und Videoblöcke zu schreiben. Schließlich ruft es [server.py](https://www.codeinsideout.com/blog/pi/stream-ffmpeg-hls-dash/stream_ffmpeg_hls/server.py) auf, um das Web zu bedienen.

Wird verwendet ffmpeg, um sowohl HLS- als auch DASH-Segmente zu generieren, mit einigen Optionen zur Reduzierung der Latenz.

ffmpeg -y \

-input\_format h264 \

-f video4linux2 \

-framerate 25 \

-use\_wallclock\_as\_timestamps 1 \

-i /dev/video0 \

-c:v copy \

-f dash \

-ldash 1 \

-seg\_duration 1 \

-frag\_duration 1 \

-streaming 1 \

-window\_size 30 -remove\_at\_exit 1 \

-strict experimental -lhls 1 \

-hls\_playlist 1 -hls\_master\_name live.m3u8 \

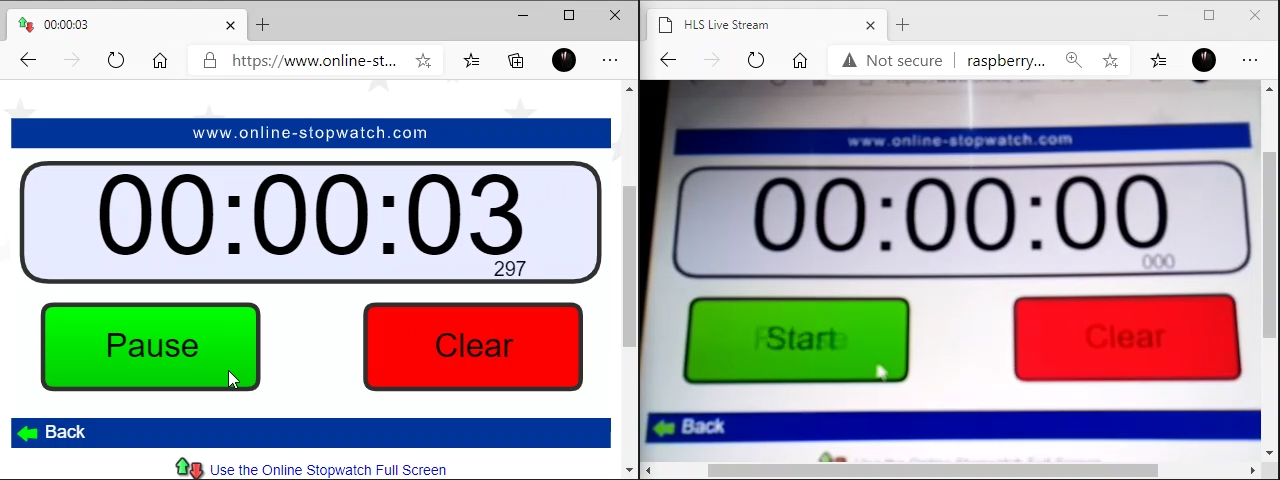
-utc\_timing\_url https://time.akamai.com/?iso \

-write\_prft 1 \

-target\_latency 1 \

/dev/shm/hls/live.mpd &

Das Ergebnis ist erwartungsgemäß nicht gut, da es noch dazu kommt3,3 Sekunden Verzögerungin einem LAN.

Verzögerung beim HLS-Streaming