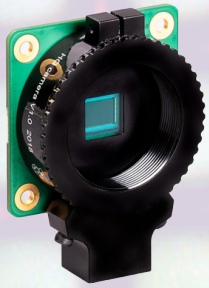


La caméra Raspberry Pi HQ en action

prise de vue et diffusion vidéo en continu avec un Raspberry Pi 4

Mathias Claußen (labo d'Elektor)

Nous avons déjà vu l'utilisation du Raspberry Pi 4 comme caméra réseau. Cette fois, nous montrons comment configurer une caméra pour diffuser en continu de la vidéo et prendre des photos. Bien sûr, ce serait moins cher avec un dispositif prêt à l'emploi, mais où serait le plaisir ? Qu'apprendriez-vous ?

Si vous cherchez une webcam bon marché, à base de RPi et prête à l'emploi, ne lisez pas cet article. Mais si voulez créer quelque chose et acquérir de nouvelles connaissances, poursuivez ! Ce projet de caméra est basé sur un RPi 4B, une caméra RPi HQ, un objectif, un écran TFT et un trépied récupéré sur un autre montage. Pour le boîtier, j'ai choisi un modèle d'Adafruit avec un joli style rétro ; du coup le résultat ressemble plus à un appareil photo. Le RPi 4B est le modèle disponible le plus récent. La **figure 1** montre l'installation en action.

De la lumière physique à l'image numérique

Avant de décrire l'assemblage de la caméra, intéressons-nous à la théorie du traitement des images et de la production des données. Pour comprendre ce que la caméra voit et ce que vous obtenez à la fin, quelques étapes de traitement et de conditionnement du signal sont nécessaires. La première étape est la partie optique.

La possibilité d'obtenir une image nette à une distance donnée dépendra des objectifs utilisés. Ils ont également un impact sur les images que vous capturerez ensuite, car les lentilles à l'intérieur de ces objectifs doivent guider la lumière incidente vers votre capteur d'images. Si elles sont mal agencées ou de mauvaise qualité, cela peut provoquer une distorsion de l'image. Ici, nous avons choisi un objectif de 16 mm, marqué 10 MP. Si vous vous demandez pourquoi un objectif peut avoir une résolution en mégapixels (MP), lisez entre autres [1]. Cela signifie également qu'un objectif de 10 MP ne pourra jamais faire qu'un capteur d'image produise plus de pixels que ce qui est spécifié. D'ailleurs, nous effectuerons plus loin une comparaison avec l'objectif à 3 MP de 6 mm. Choisissez l'objectif en fonction de votre application.

Outre la partie optique, le capteur d'image lui-même joue un rôle important. Beaucoup de gens pensent que plus sa résolution de capture est élevée, meilleur il est. Si vous avez le choix, vous opterez sans doute pour un capteur à 18 MP plutôt qu'un modèle à 12 MP, car vous obtiendrez 6 MP de plus en résolution. Mais pour les capteurs d'images, une deuxième dimension entre en jeu : leur taille. Plus un capteur d'image est petit pour un nombre de MP donné, moins il y a de lumière disponible par pixel. En outre, plus ces capteurs sont petits, plus il est probable que les pixels voisins puissent s'influencer mutuellement, non seulement à cause de la lumière, mais aussi à cause des effets thermiques au sein de la puce. Cela signifie que la surface utilisée par un capteur peut influencer la qualité de l'image que vous obtenez, tout comme le nombre de mégapixels déclarés.

Noir, blanc et beaucoup de nuances de gris

Pourquoi parler de noir, blanc et nuances de gris alors qu'on souhaite des images en couleur ? Vous devez vous intéresser à ce que le capteur peut voir et ce qu'il va délivrer à notre système. Vous avez peut-être déjà joué avec des photodiodes qui réagissent à la lumière incidente par un changement de résistance. Sur une puce d'appareil photo, le

principe est le même, mais nous avons des millions de ces diodes, densément serrées les unes contre les autres. Chacune de ces diodes n'est capable de distinguer que la quantité de lumière qui la frappe à un endroit donné. La couleur impliquée n'est pas prise en compte, de sorte que la puce ne peut reconnaître que le noir et le blanc, avec de nombreuses nuances de gris entre les deux. Ce qui revient à dire qu'à la base la puce est achromate.

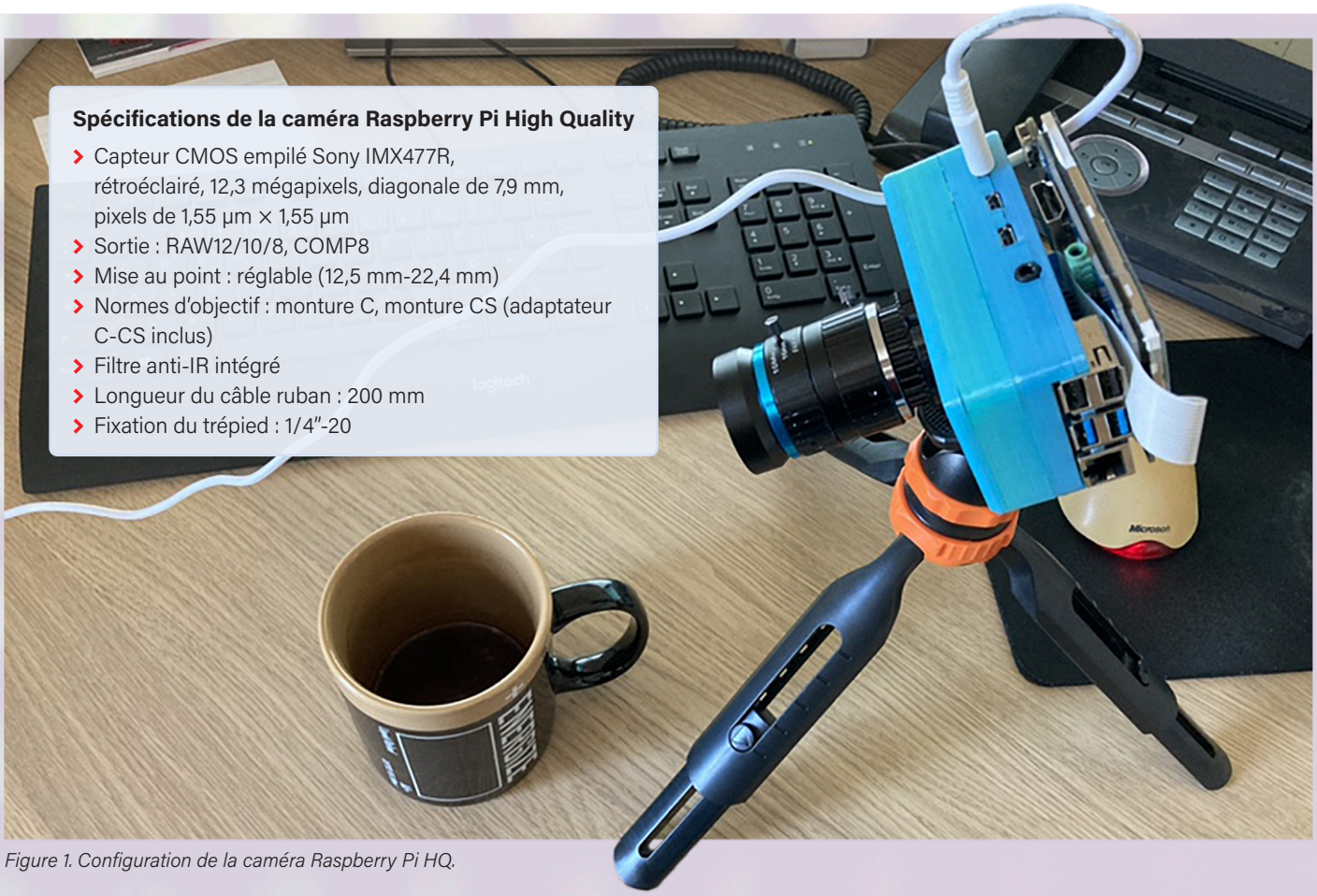
S'il faut détecter les trois principales composantes de couleur d'une image – rouge, bleu et vert – on peut ajouter un filtre optique. Ce filtre est ajouté à chaque diode ou, pour simplifier la formulation, à chaque pixel brut. Cela signifie que nous avons maintenant des millions de pixels bruts qui peuvent indiquer un niveau individuel pour la couleur à laquelle ils sont dédiés. On pourrait supposer qu'ils sont répartis sur la puce avec des motifs rouge, bleu, vert. En fait, la caméra dispose de 50% de pixels bruts verts sur son capteur, le rouge et le bleu représentant chacun 25%, selon la matrice de filtres de Bayer. Pour une explication plus détaillée, consultez l'article de Wikipédia en [2]. Le capteur doit échantillonner chacun de ces pixels bruts et produire une image RVB appropriée à partir de ces pixels. Le capteur d'image lui-même est aujourd'hui plus qu'un simple convertisseur analogique-numérique, il participe au traitement de l'image.

Transfert d'images

Lorsque le capteur d'image a préparé les informations entrantes, il faut les transmettre à un hôte qui peut poursuivre le traitement. Le moyen de transfert le plus simple possible consiste à utiliser un bus parallèle et à émettre les données à chaque top d'horloge. Bien que simple et utilisée sur les plus petites UC embarquées, cette méthode est limitée. Déplacer de nombreux bits en parallèle à vitesses élevées peut être un défi, notamment pour la synchronisation et l'intégrité des données sur de longues distances, et le protocole peut être propre à chaque module de caméra.

L'Alliance MIPI (*Mobile Industry Processor Interface*) a défini l'interface série pour caméra (CSI, *Camera Serial Interface*) afin de permettre l'interchangeabilité des modules de caméra. La spécification CSI-2 définit une connexion de données série entre la caméra et l'hôte, constituée de plusieurs voies de données. Pour le RPi 4B, seules deux voies de données sont acheminées vers le connecteur de la caméra, ce qui donne un débit de 3 Gbits/s, chaque voie pouvant transférer 1,5 Gbits/s. La documentation MIPI CSI-2 n'étant pas librement accessible, cela explique que le pilote de l'interface de la caméra sur le RPi 4B se cache dans un micrologiciel propriétaire fonctionnant sur le GPU.

Si nous faisons quelques calculs, le transfert d'images de plus de 1920 × 1080 pixels avec 24 bits de couleur (huit bits par canal) se traduira par 49,77 Mbits par image. Avec une bande passante de 3 Gbits/s, cela donne un transfert de 60 images par seconde. En théorie, cela signifie que pour une résolution de 2560 × 1440 pixels et 24 bits de couleur, on pourrait transférer un peu plus de 30 images par



Spécifications de la caméra Raspberry Pi High Quality

- Capteur CMOS empilé Sony IMX477R, rétroéclairé, 12,3 mégapixels, diagonale de 7,9 mm, pixels de 1,55 µm × 1,55 µm
- Sortie : RAW12/10/8, COMP8
- Mise au point : réglable (12,5 mm-22,4 mm)
- Normes d'objectif : monture C, monture CS (adaptateur C-CS inclus)
- Filtre anti-IR intégré
- Longueur du câble ruban : 200 mm
- Fixation du trépied : 1/4"-20

Figure 1. Configuration de la caméra Raspberry Pi HQ.

seconde. Malheureusement, cette résolution ne peut pas être utilisée pour la diffusion vidéo. S'il s'agit d'images fixes, la bande passante est moins problématique, même à la résolution maximale de 12 MP qu'offrira le module caméra Raspberry Pi HQ.

Un kit de construction de caméra

Considérons les ingrédients nécessaires à la construction d'une caméra. Au cœur de l'assemblage se trouve la caméra Raspberry Pi HQ qui présente une résolution de 12 MP (**fig. 2** et **3**). Reportez-vous à ses spécifications dans l'encadré « Spécifications de la caméra Raspberry Pi HQ ». Nous utilisons l'objectif à monture C de 16 mm (**fig. 4**) et l'objectif à monture CS de 6 mm (**fig. 5**). Les deux sont disponibles dans la boutique Elektor [3][4].

Un Raspberry Pi 4 traite les images (**fig. 6)**. Toutes les pièces sont maintenues dans un boîtier imprimable en 3D d'Adafruit. Note : même s'il est imprimé en 3D, vous avez besoin de quelques vis. Pour plus de détails sur le boîtier, consultez [5]. Les pièces sont présentées dans la **figure 7**.

L'écran TFT de 3,5 pouces (**fig. 8)** et le trépied (**fig. 9**) sont issus d'anciens projets. La **figure 10** montre la caméra assemblée. Voilà un joli kit de construction pour votre propre caméra.

Nous avons un objectif à monture C et un objectif à monture CS. N'oubliez pas d'ajouter ou de retirer l'adaptateur de monture CS à monture C selon l'objectif que vous utilisez. Si vous avez du mal à faire la mise au point, il se peut que vous ayez installé l'adaptateur alors que vous utilisez un objectif à monture CS. L'adaptateur illustré à la **figure 11** doit être retiré pour les objectifs à monture CS. Son omission ou son utilisation incorrecte peuvent entraîner des images floues.

C'est presque dans la boîte

Une fois tout monté, une version récente de Raspberry Pi OS est installée sur une carte SD. Comme un écran est connecté à des ports HDMI, aucune étape supplémentaire n'est nécessaire pour obtenir une image sur celui-ci, il suffit d'un câble approprié. Si vous utilisez un écran de 3,5 pouces moins cher avec une simple connexion SPI, vous devrez peut-être modifier certains fichiers dans le système d'exploitation du RPi. Si vous voulez acheter un nouvel écran, le fait de disposer d'une entrée HDMI vous dispense de cette modification et vous devriez recevoir directement une sortie d'image.

Une fois la configuration initiale effectuée par Raspberry Pi OS, il faut activer le module caméra. Vous pouvez utiliser *Configuration du Raspberry Pi* à cette fin. Sous l'onglet *Interfaces*, activez *Caméra*. Pour commander le système par le réseau, vous pouvez également cocher *SSH* et *VNC*. Après un redémarrage, la caméra devrait être prête à l'emploi.

Effectuez un premier test avec un terminal. Tapez `raspistill -fp -s` pour obtenir un aperçu de ce que voit la caméra. Si vous utilisez VNC

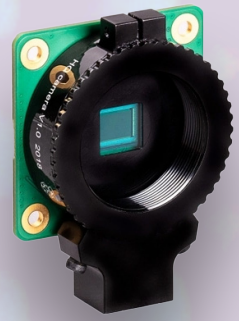


Figure 2. Module caméra Raspberry Pi HQ (source : www.raspberrypi.org).

Figure 3. Option de montage de la caméra Raspberry Pi HQ (source : www.raspberrypi.org).

Figure 4. Objectif de 16 mm pour le module caméra Raspberry Pi HQ.

Figure 5. Objectif de 6 mm pour le module caméra Raspberry Pi HQ.

pour accéder à distance à votre bureau, sachez que vous ne verrez pas la sortie de la caméra dans la visionneuse VNC. Si vous souhaitez que l'aperçu de la caméra soit visible, modifiez les paramètres du serveur VNC comme indiqué à la **figure 12**.

Pour prendre une photo et l'enregistrer sous forme d'image, tapez `raspistill -q 100 -o img.jpg`. Un fichier *img.jpg* sera créé et la qualité sera fixée à 100 pour la compression.

Première séance d'essais

Maintenant que la caméra fonctionne, prenons quelques premières images de test. Nous utiliserons d'abord l'objectif de 6 mm (fig 5). La **figure 13** montre une page de notes A5 avec des lignes droites. La **figure 14** montre un gros plan de cette feuille et vous pouvez voir que l'objectif fonctionne comme une sorte d'optique *fisheye*. Toute la lumière incidente est maintenant transformée en une image unie, mais qui a l'aspect d'une petite sphère, alors que la feuille se trouve à 13 cm de l'objectif. On trouve les spécifications techniques de cet objectif en [4].

Avec l'objectif de 16 mm, nous utilisons la même feuille de notes A5. La **figure 15** montre que l'image n'est visiblement pas déformée comme avec l'objectif de 6 mm. Les spécifications de l'objectif se trouvent en [3].

L'objectif ne participe que pour moitié à la qualité de l'image. Le module caméra Raspberry Pi HQ contribue à la deuxième moitié (fig. 3). Au cœur du module, on trouve un capteur Sony IMX477 [6].

Ceci n'est pas un appareil photo

Avec l'objectif et le RPi 4B, on a tout le matériel de base que l'on trouverait également dans un appareil photo. Ce qui est plus rare de nos jours, c'est l'absence de mise au point automatique pour l'objectif et le module caméra. C'est à vous de régler la mise au point. De même, si vous êtes habitué aux appareils photo, vous remarquerez rapidement l'absence de logiciel. Le RPi n'est pas, du point de vue logiciel, un véritable appareil photo de remplacement. La correction d'image, le filtrage et les autres paramètres que l'on trouve sur les logiciels d'appareil photo ne sont pas présents. C'est donc à vous d'écrire un logiciel de caméra pour travailler avec. Pour l'instant, je ne me concentre pas sur le développement de logiciels pour la caméra, mais je pourrais le faire dans un avenir proche.

Diffusion vidéo en continu

Bien qu'il ne s'agisse pas d'un appareil photo classique, il y ressemble. Mais qu'en est-il de la diffusion de vidéo ? Le RPi peut diffuser des vidéos, mais nous sommes limités à 1080p avec 30 images par seconde (FPS). Néanmoins, avoir une caméra qui peut être connectée via le Wi-Fi semble être une bonne chose. De plus, nous pouvons maîtriser le type de logiciel utilisé et sécuriser le RPi.

Pour la diffusion en continu, il y a plusieurs options logicielles. J'ai choisi µStreamer du projet pi-kvm [7]. Si vous voulez faire la même chose, vous devez suivre le guide pour compiler le logiciel avec le support d'OpenMAX. OpenMAX vous permet d'utiliser l'accélération matérielle

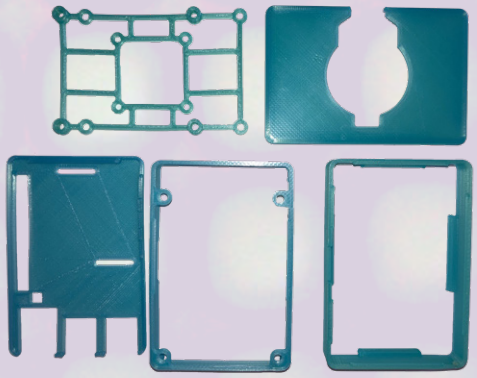


Figure 6. Pièces de base pour la caméra Raspberry Pi.

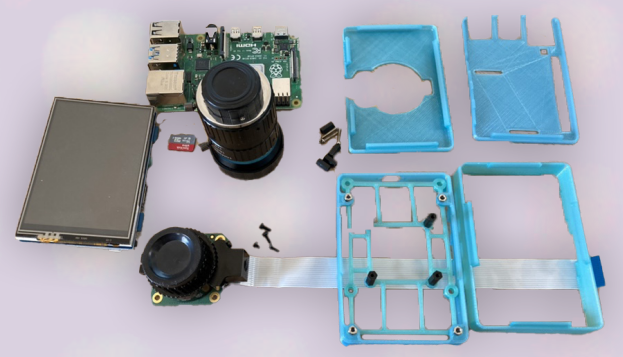


Figure 7. Pièces du kit de construction de la caméra.

Figure 8. Module TFT en option.

Figure 9. Trépied en option.

Figure 10. Caméra assemblée.

pour les tâches multimédias comme la compression d'images, de sorte qu'un flux MJPEG peut être produit sans que l'UC ait à faire tout le travail.

Clonez le dépôt avec `git clone --depth=1 https://github.com/pikvm/ustreamer` et installez les librairies requises sur le RPi avec `sudo apt install libevent-dev libjpeg8-dev libbsd-dev libraspberrypi-dev` pour mettre tous les éléments en place. Maintenant, sur un terminal, vous devez entrer dans le répertoire *µstreamer* et taper `make OMX=1` pour commencer la compilation. Cela ne prendra qu'un court instant jusqu'à ce qu'un exécutable soit prêt.

Pour lancer un flux, vous pouvez utiliser `./ustreamer --format=YUYV --encoder=OMX --workers=3 --host=0.0.0.0 --port=8080 -r 1920x1088 -q 90 -f 30`. Cette commande met en place un flux MJPEG de 1920 × 1080 pixels à 30 FPS. Si vous accédez maintenant à votre RPi avec un navigateur via le port 8080, vous verrez une page web comme à la **figure 16**. Si nous accédons maintenant au flux, la fréquence d'images sera très faible, environ 3 à 4 FPS, mais vous obtiendrez une bonne qualité de vidéo.

Les pilotes et leurs modes

Le RPi fournit, pour tous les modules caméra, un pilote V4L2 qui permet d'écrire des applications pour l'entrée vidéo ou audio en utilisant une interface standard. Cela facilite l'utilisation de la caméra avec le RPi et permet de recourir à un large éventail de logiciels, mais il y a un problème. Tant que nous demandons au pilote de fournir une résolution de 1270 × 720 pixels ou moins, il exploite le module caméra connecté en mode vidéo. Cela signifie que nous obtenons jusqu'à 60 FPS à 1270 × 720 pixels. C'est bien si vous branchez un petit capteur bon marché pour avoir une sorte de caméra de surveillance. Pour obtenir une résolution plus élevée (par ex. 1920 × 1080 pixels), le pilote passe en mode photo. Dans ce cas, il fait fonctionner le capteur comme un appareil photo et effectue tous les traitements d'image, les corrections de luminosité et le post-traitement comme s'il prenait des photos. Il en résulte un taux de rafraîchissement très faible, mais une image de bonne qualité.

Pour contourner ce problème, nous devons décharger le pilote V4L2 avec `sudo rmmod bcm2835-v4l2` et le recharger avec `sudo modprobe bcm2835-v4l2 max_video_width=4056 max_video_height=3040`. Cela force l'utilisation du mode vidéo pour toutes les résolutions que le RPi prend en charge. Si nous démarrons maintenant µStreamer avec `./ustreamer --format=YUYV --encoder=OMX --workers=3 --host=0.0.0.0 --port=8080 -r 1920x1088 -q 90 -f 30`, le nombre d'images par seconde est plus élevé. Si vous ouvrez à nouveau le flux, vous pouvez également accéder aux statistiques et voir que la cadence est maintenant d'environ 20 FPS, selon votre navigateur.

Latence et plus d'images

Un inconvénient de la méthode ci-dessus est le retard introduit dans le flux. L'ensemble du processus, depuis le RPi jusqu'à votre navigateur,

Figure 11. Bague d'adaptation CS.

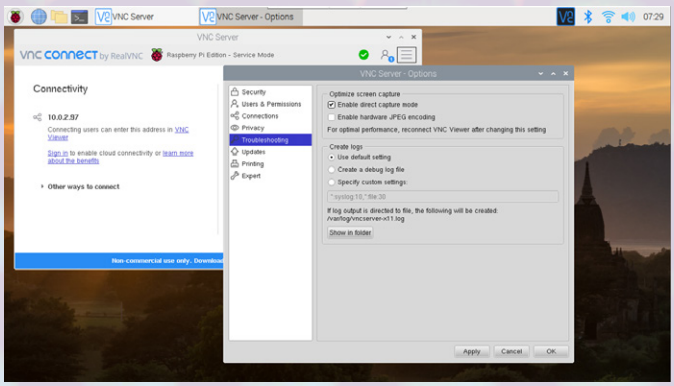


Figure 12. Paramètres VNC pour visualiser la sortie de la caméra.

Figure 13. Feuille de notes A5 avec des lignes droites.

Figure 14. Vue rapprochée avec un objectif de 6 mm.

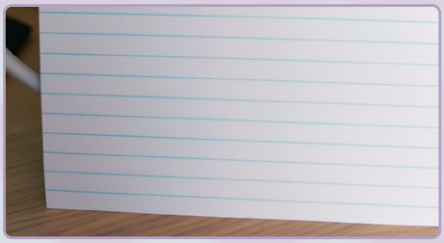


Figure 15. Feuille A5 capturée avec un objectif de 16 mm.

dure environ une seconde. Si vous faites de la diffusion en continu ou avez l'intention d'en faire, ce délai peut être très désagréable. Le réduire dépend du RPi lui-même, du temps qu'il faut pour capturer et envoyer l'image sur le réseau, et aussi du temps que prennent le décodage et l'affichage de l'image sur le périphérique de lecture. Une solution pour augmenter la cadence est d'utiliser *raspivid* avec `raspivid -t 0 -w 1920 -h 1080 -fps 30 -l -o tcp://0.0.0.0:5000`. Cela permet à un lecteur vidéo comme VLC de se connecter à votre Raspberry Pi et de lire le flux vidéo, toujours avec une latence d'environ une seconde.

Diffusion en continu RTSP avec H.264

Outre la diffusion en continu MJPEG ou l'utilisation de *raspivid* avec TCP, nous pouvons utiliser RTSP. RTSP est un protocole de diffusion en temps réel, conçu pour contrôler la diffusion de données sur un réseau. Il orchestre l'échange de données entre les clients et le serveur de diffusion. Comme RTSP ne gère que l'échange de données, il est également nécessaire de disposer d'un outil capable de produire un flux de données compressé que nous pouvons transporter. C'est là que UV4L (*User space Video4Linux*) entre en jeu. Il permet d'utiliser l'entrée caméra du RPi 4B en combinaison avec l'encodeur H.264 inclus dans le RPi 4B pour produire un flux de données compressé H.264.

Cette configuration permet d'obtenir un flux vidéo (sans audio) avec un retard inférieur à une seconde à 6 Mbit/s et une utilisation de l'UC d'environ 10%. Pour mesurer le délai, la caméra a été dirigée vers un moniteur affichant l'heure locale et un lecteur VLC lisant la vidéo a été placé au-dessus. Voir les **figures 17** et **18**. Nous pouvons calculer un délai de 670 ms pour le traitement complet du signal. Ce n'est toujours pas parfait, mais c'est beaucoup plus exploitable. La **figure 19** montre les détails du flux produit par cette configuration.

Comment configurer cela ? Tout d'abord, vous devez installer UV4L sur votre RPi 4B en suivant le guide [8]. En plus de l'UV4L de base, vous devez également installer les extensions avec `sudo apt-get install uv4l-raspicam-extras`. C'est la première partie pour avoir un périphérique vidéo qui peut sortir un flux H.264. La configuration sera faite dans une étape ultérieure.

La partie suivante est le v4l2rtspserver [9]. Clonez le dépôt avec `git clone https://github.com/mpromonet/v4l2rtspserver.git` sur votre RPi 4B. Installez CMake avec `sudo apt install cmake` pour pouvoir compiler le v4l2rtspserver. Utilisez un terminal et entrez dans le v4l2rtspserver qui vient d'être créé. Tapez `cmake . && make` pour compiler le code. Si tout fonctionne comme prévu, tapez `sudo make install`.

En se basant sur le post du forum [10], nous allons raccourcir un peu le guide. Dans un terminal, tapez `sudo nano /etc/uv4l/uv4l-raspicam.conf` et cherchez la ligne `encoding`. Changez-la en `encoding = h264`. Recherchez également `width` et `height`. Changez-les en `width = 1296` et `height = 972`. Ensuite, recherchez `framerate`, mettez-le à `framerate = 30`, et enregistrez vos modifications.

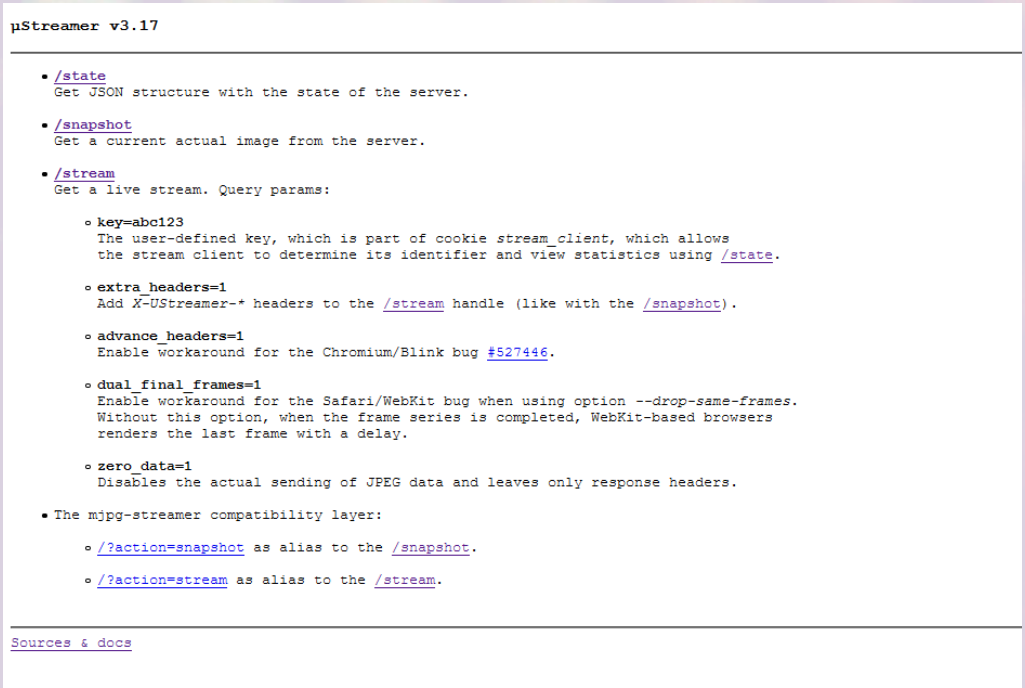


Figure 16. Interface web de µStreamer.

Vous vous demandez peut-être pourquoi nous n'optons pas pour la Full HD (1920 × 1080 pixels). Cela correspondrait à un facteur de forme de 16/9. Comme le capteur nous donne une image 4/3, nous perdrions certaines parties de l'image avec ce choix. S'il nous faut du 16/9, nous pourrons le faire plus tard.

Ensuite, il faut éditer un script pour charger l'uv4l pour la caméra Raspberry Pi HQ au démarrage. Dans un terminal, tapez `sudo nano /etc/systemd/system/uv4l_raspicam.service` et remplacez la ligne `ExecStart` par `ExecStart=/usr/bin/uv4l -f -k --sched-fifo --mem-lock --config-file=/etc/uv4l/uv4l-raspicam.conf --driver raspicam --driver-config-file=/etc/uv4l/uv4l-raspicam.conf --enable-server off`. Enregistrez le fichier et redémarrez le RPi 4B. Pour diffuser maintenant le flux de votre caméra, ouvrez un terminal et tapez `export LD_PRELOAD=/usr/lib/uv4l/uv4lext/armv6l/libuv4lext.so`. Sinon, le v4lrtspserver ne se chargera pas et se terminera avec un message d'erreur. Pour démarrer un flux de test, tapez dans un terminal `v4l2rtspserver -F 30 -H 960 -W 1280 -P 8555 /dev/video0` et le v4lrtspserver devrait démarrer et afficher une fenêtre de prévisualisation sur votre bureau.

Pour afficher le flux vidéo, vous pouvez utiliser VLC [11] et ouvrir un flux réseau. Comme URL, utilisez `rtsp://<vous.raspberry.pi.ip>:8555/unicast`. Vérifiez sous *Afficher plus d'options* que la *Mise en cache* est réglée sur un maximum de 100 ms pour une meilleure performance en temps réel. Appuyez sur *Lire* et vous devriez voir le flux de votre caméra.

USB OTG

Puisque le transport de la vidéo sur une connexion réseau peut introduire des retards, pourquoi ne pas utiliser l'USB à la place ? Le RPi 4B+ offre un support USB OTG, ce qui lui permet d'agir comme un périphérique USB lorsqu'il est connecté à un PC. Actuellement, la partie USB nécessaire pour transformer le RPi 4 en webcam présente encore quelques bogues [12], c'est pourquoi nous ne l'utiliserons pas dans cet article. De plus, cette configuration nécessitera un câble en Y pour séparer l'alimentation et les données du port USB, qui doit actuellement être réalisé à la main. Donc pas de sortie vidéo stable à faible latence, du moins avec le RPi 4. Il n'en serait pas de même pour le RPi Zero W, comme le démontre la *webcam Show-me* [13].

Action !

La caméra assemblée (fig. 1) est posée sur mon bureau. Elle capture des images, les enregistre et diffuse des vidéos. Le capteur d'image, associé aux objectifs à monture C et CS, produit une excellente image en mode photo et lors de la capture de vidéos. L'association de ce matériel avec le logiciel et le pilote me laisse des sentiments mitigés. Le RPi 4B est limité en matière de codage vidéo et de mise en œuvre de l'interface de la caméra.

En écrivant cet article, j'ai eu quelques idées pour améliorer ou utiliser des logiciels existants. La première serait d'implémenter une interface utilisateur fonctionnelle pour de la photo de base avec un aperçu en direct et un enregistrement des images. Dans le meilleur des cas, celle-ci fonctionnerait également sur les réseaux avec une

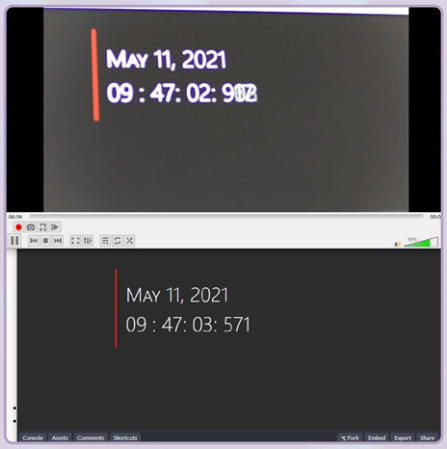


Figure 17. Première mesure du retard du flux.

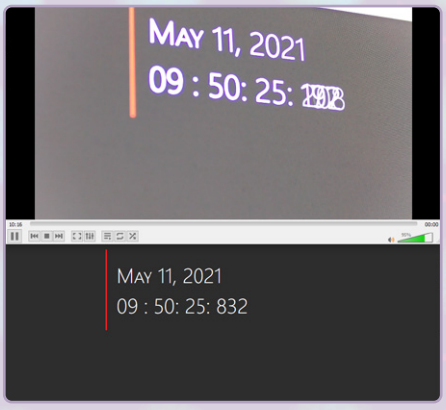


Figure 18. Deuxième mesure du retard du flux.

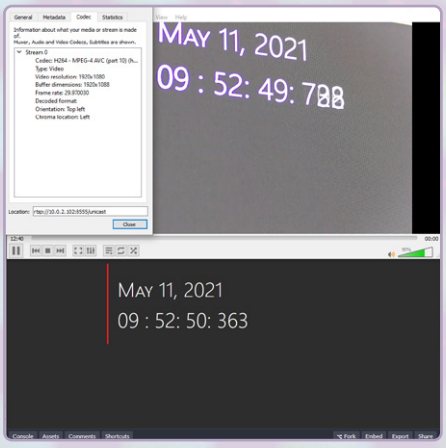


Figure 19. Paramètres du flux.

latence inférieure à une seconde. En outre, il faudrait examiner plus en profondeur l'aspect USB OTG, car le matériel du RPi 4B en est capable. Une fois encore, si cela avait fonctionné dès le déballage, où serait le plaisir ? Les meilleures leçons à tirer proviennent des défis qu'offre un projet.

(200582-04)

Contributeurs

Texte et photos : **Mathias Claußen**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Harmen Heida**
Traduction : **Denis Lafourcade**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- **Objectif de 16 mm à monture C (10 MP) pour module caméra Raspberry Pi HQ**
  www.elektor.fr/16-mm-c-mount-lens-10-mp-for-raspberry-pi-hq-camera-module
- **Objectif de 6 mm à monture CS (3 MP) pour module caméra Raspberry Pi HQ**
  www.elektor.fr/6-mm-cs-mount-lens-3-mp-for-raspberry-pi-hq-camera-module
- **Module caméra Raspberry Pi HQ**
  www.elektor.fr/raspberry-pi-high-quality-camera-module
- **Raspberry Pi 4B (2 Go de RAM)**
  www.elektor.fr/raspberry-pi-4-b-2-gb-ram
- **Raspberry Pi 4B (4 Go de RAM)**
  www.elektor.fr/raspberry-pi-4-b-4-gb-ram
- **Imprimante 3D i3 Mega-S d'Anycubic (kit)**
  www.elektor.fr/anycubic-i3-mega-s-3d-printer-kit

LIENS

[1] Mégapixels et capteur photographique : https://fr.wikipedia.org/wiki/Capteur_photographique
[2] Matrice de filtres de Bayer : https://fr.wikipedia.org/wiki/Matrice_de_filtres_color%C3%A9s
[3] Objectif de 16 mm, boutique Elektor : www.elektor.fr/16-mm-c-mount-lens-10-mp-for-raspberry-pi-hq-camera-module
[4] Objectif de 6 mm, boutique Elektor : www.elektor.fr/6-mm-cs-mount-lens-3-mp-for-raspberry-pi-hq-camera-module
[5] Boîtier en impression 3D : https://learn.adafruit.com/raspberry-pi-hq-camera-case
[6] Informations sur le capteur IMX477 : www.sony-semicon.co.jp/products/common/pdf/IMX477-AACK_Flyer.pdf
[7] µStreamer sur GitHub : https://github.com/pikvm/ustreamer
[8] Installation d'UV4L : https://www.linux-projects.org/uv4l/installation/
[9] v4l2rtspserver sur GitHub : https://github.com/mpromonet/v4l2rtspserver
[10] Message du forum sur la diffusion en continu H.264 à faible latence : www.bensoftware.com/forum/discussion/3254/raspberry-pi-h264-rtsp-low-latency-camera-instructions/p1
[11] Lecteur VLC : https://www.videolan.org/
[12] Limites du « gadget » UVC pour Raspberry Pi 4B : https://www.raspberrypi.org/forums/viewtopic.php?t=294140
[13] Webcam Show-me sur GitHub : https://github.com/showmewebcam/showmewebcam