Zerocam

Scripte:

#/bin/bash

raspistill -t 86000000 -tl 10000 --rotation 90 -q 80 -o /home/pi/Photo/image%04d.jpg

---

#/bin/bash

raspistill -t 0 -tl 10000 -w 1920 -h 1080 --annotate 12 --rotation 90 -o /home/pi/Photo/image%04d.jpg

---

rc.local:

```
#!/bin/sh  -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
#aktivieren...
#/home/pi/10sec-still_FullHD.sh
#/home/pi/camstream.sh
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

exit 0
```

Streaming: http://zerocam2:8080/?action=stream

/bin/bash

v4l2-ctl -c compression_quality=85
v4l2-ctl -c brightness=55
v4l2-ctl -c rotate=90
v4l2-ctl -c contrast=30
v4l2-ctl -c saturation=20
v4l2-ctl -c scharpness=60
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 1280x720 -f 25" -o "output_http.so -w /usr/local/share/mjpg-streamer/www -p 8080"

Anleitung:

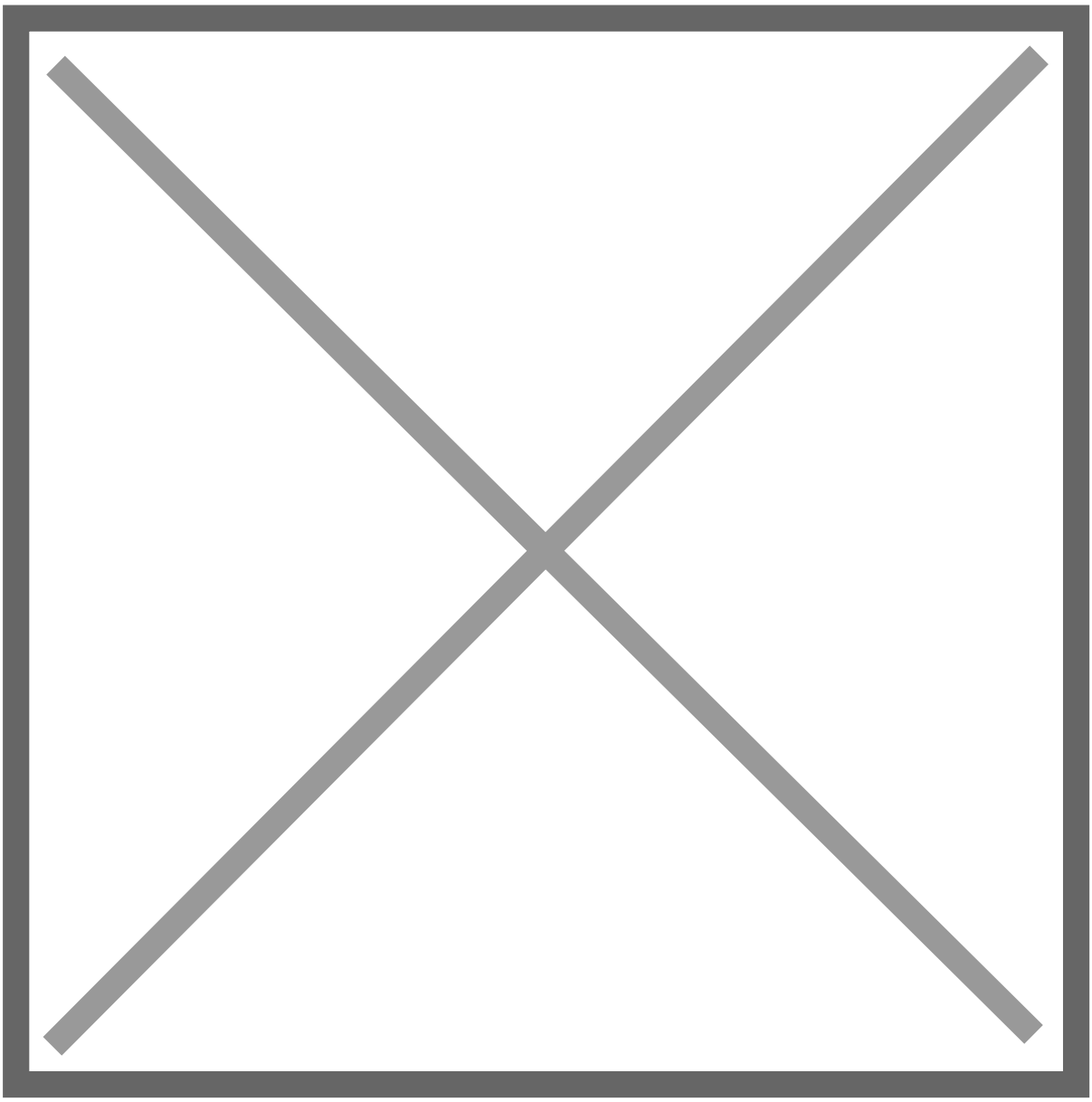
Let’s test by trying to run mjpg-streamer against the input_uvc.so plugin, since I”m not using a Raspi-Camera, but a USB one instead. (**See notes at the bottom about Raspberry Pi Bullseye and Raspi-Cameras**). I’ll also output using the http plugin. So my testing command line looks like this:

```
/usr/local/bin/mjpg_streamer -i "input_uvc.so -f 15 -r 640x480" \
 -o "output_http.so -w /usr/local/share/mjpg-streamer/www"
```

Executing that line gives us a dump of data – here’s the relevant info:

```
MJPG Streamer Version: git rev: 5a6e0a2db163e6ae9461552b59079870d0959340
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: 15
 i: Format............: JPEG
 i: TV-Norm...........: DEFAULT
 o: www-folder-path......: /usr/local/share/mjpg-streamer/www/
 o: HTTP TCP port........: 8080
 o: HTTP Listen Address..: (null)
 o: username:password....: disabled
 o: commands.............: enabled
```

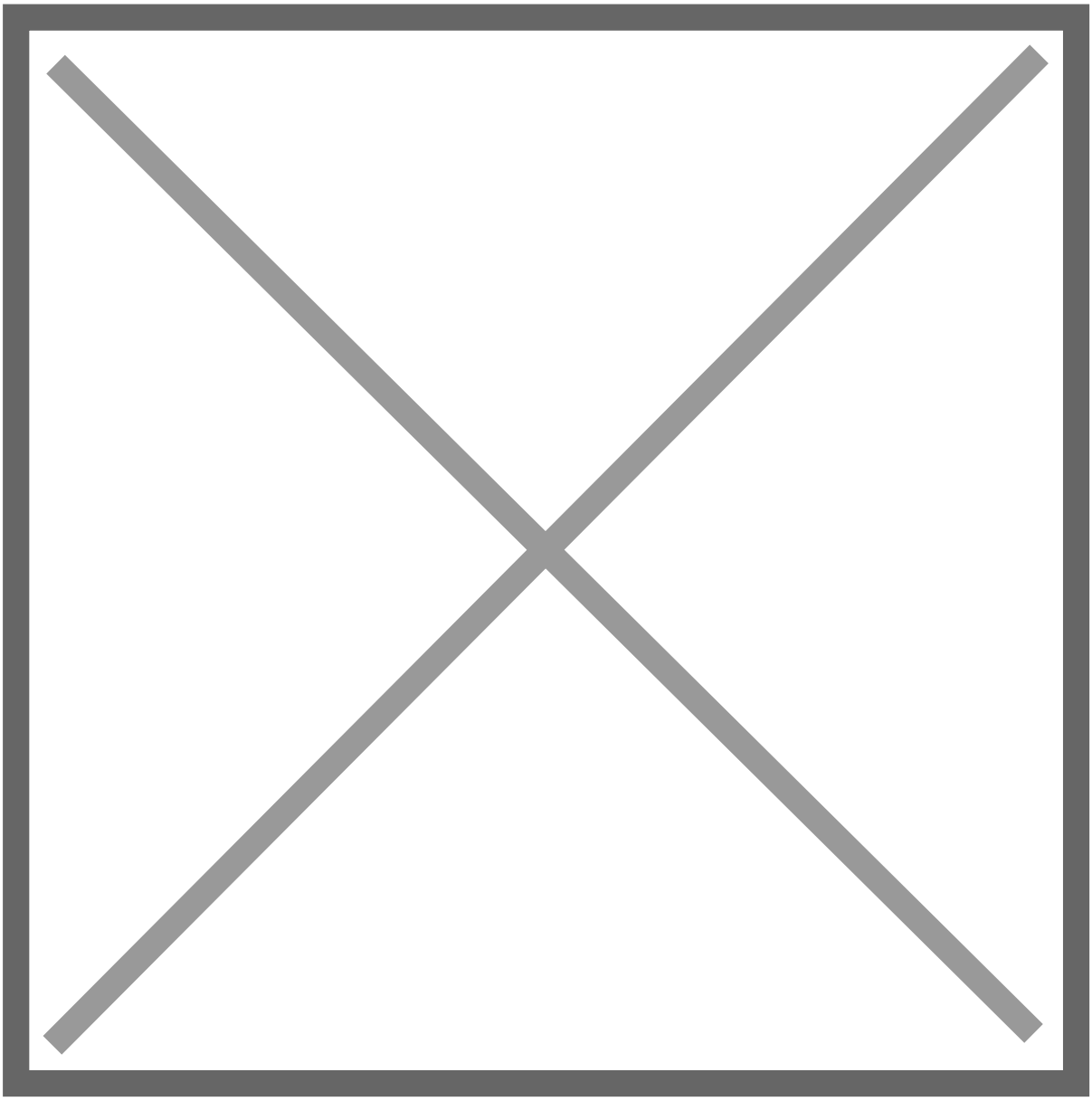
Our key variables to tweak are **-f for frame rate**, and **-r for resolution**. I’ll change those later, but for a quick test I head on out to my browser and point it at the www server at http://cam-pi-zero.local:8080/



I always feel like… somebody’s watching me…

Not only do we see the web interface, but a snapshot of the web cam, which was, clearly, laying down on a workbench behind me while I tried this out.

Validating the stream worked (by clicking ‘Stream’) and turning around, I could see myself moving:



Hello you... come here often?

You can now hit CTRL-C in your SSH window (or terminal) and quit the stream. After this point, I could delete the build files I downloaded.

Tweaking configuration

I wanted to run my camera at its intended resolution and at least a frame rate of 30fps. To do that I modified my command line:

```
#30fps, 1080 HD
/usr/local/bin/mjpg_streamer -i "input_uvc.so -f 30 -r 1920x1080" \
 -o "output_http.so -w /usr/local/share/mjpg-streamer/www"
```

I then loaded my own stream up in VLC by opening the direct network path to the stream:

http://cam-pi-zero.local:8080?action=stream

That worked, and I get about a 1-2 second delay from Pi to VLC. So I wouldn't use this with audio feeds unless I was planning on creating a sync delay. I'm using this to monitor 3D prints, or watch birds outside, so my use case does **not** demand low latency. Is it exactly 30 frames per second? No…. NO it's not.

At 1280×720, it seemed closer, but at 1920×1080, when set at 30fps, it definitely wasn't even close. I'd guess more like 15.

The PI wasn't maxed out on CPU, but it could just be the nature of USB 2.0 at this point. I'm not sure how I can tell if it's overloaded there or not, but, once again, this is a light monitoring video stream, I don't care too much about latency. It could be WiFi as well. Someday I may dig in a bit more and find out where the bottleneck is.

I think you could probably choose either 1920×1080 15fps or 1280×720 30fps routes and be okay.

Be careful about frames and resolution – Watch your CPU, but also make sure you don't see messages about fps being 'coerced' down or up. Your mileage may vary, and you may have to tweak format, fps, and resolution settings further, check out the mjpg-streamer headquarters on their git hub page (see references down below). You can also see the different modes available to your camera by using this command:

```
1 v4l2-ctl --list-formats-ext
```

Running On Startup

I liked Jacob Salmela's script – all I did was change it for my command line on the stop and restart section for my resolution and frame-rate. Save the following script by doing a sudo vi:

```
1 sudo vi /etc/init.d/livestream.sh
```

```
#!/bin/sh
# /etc/init.d/livestream.sh
### BEGIN INIT INFO
# Provides:          livestream.sh
# Required-Start:    $network
# Required-Stop:     $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: mjpg_streamer for webcam
# Description:       Streams /dev/video0 to
http://IP/?action=stream
### END INIT INFO
f_message(){
        echo "[+] $1"
}

# Carry out specific functions when asked to by the
system
case "$1" in
        start)
                f_message "Starting mjpg_streamer"
                /usr/local/bin/mjpg_streamer -b -i
"input_uvc.so -f 15 -r 1920x1080" -o "output_http.so -
w /usr/local/share/mjpg-streamer/www"
                sleep 2
                f_message "mjpg_streamer started"
                ;;
        stop)
                f_message "Stopping mjpg_streamer…"
                killall mjpg_streamer
                f_message "mjpg_streamer stopped"
                ;;
        restart)
                f_message "Restarting daemon:
mjpg_streamer"
                killall mjpg_streamer
                /usr/local/bin/mjpg_streamer -b -i
"input_uvc.so -f 15 -r 1920x1080" -o "output_http.so -
w /usr/local/share/mjpg-streamer/www"
                sleep 2
                f_message "Restarted daemon:
mjpg_streamer"
                ;;
        status)
                pid=`ps -A | grep mjpg_streamer |
grep -v "grep" | grep -v mjpg_streamer. |
awk '{print $1}' | head -n 1`
                if [ -n "$pid" ];
                then
                        f_message "mjpg_streamer is
running with pid ${pid}"
                        f_message "mjpg_streamer was
started with the following command line"
                        cat /proc/${pid}/cmdline
; echo ""
                else
                        f_message "Could not find
mjpg_streamer running"
                fi
                ;;
        *)
                f_message "Usage: $0
{start|stop|status|restart}"
                exit 1
                ;;
esac
exit 0
```

Then we enable this on startup by performing:

```
sudo chmod 755 /etc/init.d/livestream.sh
sudo update-rc.d livestream.sh defaults
```

Now you can reboot your pi, or use commands like ‘sudo service livestream start’ I rebooted my pi, and my fed was running, and running a status command on my service yields:

```
pi@cam-pi-zero:~ $ sudo service livestream status
● livestream.service - LSB: mjpg_streamer for webcam
   Loaded: loaded (/etc/init.d/livestream.sh; generated)
   Active: active (running) since Thu 2020-05-14 11:46:09 CDT; 2min 51s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 411 ExecStart=/etc/init.d/livestream.sh start (code=exited, status=0/SUCCESS)
   Memory: 2.2M
   CGroup: /system.slice/livestream.service
           └─416 /usr/local/bin/mjpg_streamer -b -i input_uvc.so -f 15 -r 1920x1080 -o output_http.so

May 14 11:46:07 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: TV-Norm...........: DEFAUL
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: www-folder-path......: /us
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: HTTP TCP port........: 808
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: HTTP Listen Address..: (nu
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: username:password....: dis
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: commands............: ena
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: starting input plugin inpu
May 14 11:46:08 cam-pi-zero.local mjpg_streamer[416]: MJPG-streamer [416]: starting output plugin: ou
May 14 11:46:09 cam-pi-zero.local livestream.sh[411]: [+] mjpg_streamer started
May 14 11:46:09 cam-pi-zero.local systemd[1]: Started LSB: mjpg_streamer for webcam.
```

Conclusion

Using the Raspberry Pi for video streams is good enough if we’re looking for low frame rate monitoring without audio. I’ve yet to find anything that really gives me the frame rate and audio (regardless of latency) that a standard USB webcam directly into my OBS machine would give. Maybe if NDI ever makes it on to the raspberry Pi, or if there is ever SLDP support.

Have you tried streaming across the network on a Raspberry Pi with a USB webcam? Did you fare better than I? Let me know!

Nevertheless, with that, I'm done! I can now embed this into OBS via the VLC media source or anything that can handle an HTTP Motion JPEG video stream. I won't have audio, but that's okay for what I'm using this for.

Raspi-Cameras (libcamera is in, raspivid is out in Bullseye)

Mjpg-Streamer won't have the raspicam input module anymore if you compile on Bullseye, because I believe the headers don't exist as the Bullseye version is going the way of libcamera. While I was able to hack around that and build it, I got HORRIBLE artifacts using a raspberry pi camera with the h264 codec and libcamera. I tried switching codecs to MJPEG instead of H264 and had less artifacts, but worse frame rate. So for now I decided to go with gstreamer instead of mjpg-streamer at 1280×720 for a nice 30fps solution.

For me, what worked was to enable the legacy raspi-camera support (so we're not using libcamera, we're using v4l2).

Here are the rough steps:

- Started out with Bullseye, latest install with sudo apt update/sudo apt upgrade executed.
- Edit /boot/config.txt and change the following:
  - start_x=1 //add this line
  - gpu_mem=128 //probably no higher
  - #camera_auto_detect=1 //Comment this line out
- Restart
- Installs:

```
1
2
```

```
sudo apt-get install gstreamer1.0-tools gstreamer1.0-plugins-base \
    gstreamer1.0-plugins-good gstreamer1.0-plugins-bad
```

Then execute this on the command line:

```
1
```

```
/usr/bin/gst-launch-1.0 v4l2src ! video/x-h264,width=1280,height=720,framerate=30/1 ! h264parse config-interval=1 ! matroskamux streamable=true ! tcpserversink host=::0 port=9000 sync=false sync-method=2
```

Once that's up and running, I ran VLC against it using :network-caching=250 and a URL of tcp://<yourhostname>:9000 and was able to get a decent stream. That's about the extent I've been pushing the raspi-cameras as I typically use the USB side. I did notice it was nice and fluid, along with less power consumption by a 100 milliamps or so, so I think I'll use this as a portable battery powered wireless camera to use around the house on streams. Hmmm, I may need to look at getting a microphone on here somehow and streaming audio with gstreamer as well.

Revision #2
Created 2023-01-15 02:45:06 UTC by willi
Updated 2024-11-02 15:46:57 UTC by willi