

# Set up a Linux Audio DSP UAC2 Rpi3A+ Gadget

Version:

23/03/29

All this is work under progress, and quite a bit has changed since the last version of this HowTo. The actual software versions provide better, simpler and more robust functions out of the box, and so some workarounds are no longer needed. Linux Kernel is now V.6.1 and therefore finally fully functional for an audio gadget. Some welcomed feedback and infos from the internet has again been included into this version - thank you to all having commented the former versions and especially to p.hofman and h.enquist posting on diyaudio.com. This version partially allows also a more dynamic approach than the previous one, in setting some configuration by executable shell script files if possible, instead of statically modifying a set of configuration files. In this logic, it avoids e.g. the use of the rigid and deprecated g\_audio kernel module in favor of resorting to the more versatile and modern configfs interface.

Topic:

It will take approx. 30 minutes to set up a basic UAC2 gadget following part I of this HowTo. Note that the resulting system will come along also with all these standard OS features not really needed for an audio-gadget-only setup. So the emerging gadget is still a bit "overweight", despite starting from the Raspberry Pi OS Lite (64-bit) variant. Be clear that this basic setup will completely satisfy the needs of most users and that in certain cases it might not be appropriate to further change anything. This said, an optional, more extensive second part of this HowTo will deal with some tweaking options and modifications to make the setup a bit leaner. It's a part II for all these who like to remove ballast and add some additional polish, bells and whistles, and this is why this How-To is split in two parts: To keep things as short and as simple as possible in part I for all pragmatically oriented users.

System Actuality:

Raspberry Pi OS imager 1.7.4 / Bullseye V.6.1 after regular full upgrade / CamillaDSP V.1.0.3

Ingredients:

Rpi3A+, Raspberry Pi OS Lite (64Bit), CamillaDSP.

Warning:

Some suggestions may not be really elegant, maybe stupid, and some better solutions may exist. Feedback welcomed!

About the formatting of this HowTo:

# Code is formatted in this typeset

# Code formatted LikeThis might have to be manually edited. It is either UserDefined, SystemSpecific or ChangingOverTime, like HomeDirectoryTrees or ReleaseVersions.

Disclaimer:

Take utmost care. DSP might damage or destroy your equipment if not correctly set up. Electricity might be lethal, such as music at > 200dB SPL. And remember, life as such is life threatening. Proceed at your own risk, death is lethal ...

# Content

Content of Part I/III - The basics:

- A. Install and update the basic OS
- B. Setup the UAC2 mode
- C. CamillaDSP
- D. Make things work together

Content of Part II/III - Tweaks and options

- E. Pimp your audio
- F. Lean Ux - an attempt to get the whole system a wee bit leaner, lighter, faster

Content of Part III/III Addendum:

- X. Scripts
- Y. Debug and system checks
- Z. Useful links

## **Part I/III: The Basic Setup**

A: The basic Raspberry OS

B. Setup the audio UAC2 / OTG

B1: Set up the UDC and the DWC2 kernel module

B2: Create the audio gadget functionality

C: CamillaDSP forever ...

C1: Install CamillaDSP (CDSP) on your system

C2: Configure CDSP to capture from the gadget's USB\_input

D. Make things work together

D.1. Gadget mode autostart

D.2. CamillaDSP autostart

D.3. Enabling these autostart services

## **Part II/III: Tweaks, Slimdowns and optional Goodies**

E. Pimp your audio

E.1. Disable the default RaspberryPi onboard audio functionality if not needed

E.2. Set the CPU to a constant frequency

E.3. Run audio programs (such as CDSP) on specific and optionally isolated CPU(s)

E.3.a. Assign programs to specific CPUs

E.3.b. CPU isolation

E.3.c. Pros and Cons

F. Lean Ux - an attempt to get the system a wee bit leaner, lighter, faster

F.1. Save your SD card

F.1.a. Write temp files into a RAM disk instead onto the SD card

F.1.b. Disable swapping to a file

F.1.c. Minimize journaling

F.1.d. Disable journaling on the ext4 file system

F.1.e. Uninstall useless programs (useless in terms of useless for your specific audio needs and use)

F2. Beef up your WiFi

F.2.a. Make the dhcp startup some seconds faster

F.2.b. Set up iwctl instead of (outdated and bloated) wpa-suplicant

F.3. Kernel modules - the good, the bad, and the uglies ...

F.3.a. Static no-loading and/or dynamic removing unwanted kernel modules methods

F.3.b. First - Kernel modules better to keep (the good)

F.3.c. Modules not needed for specific audio use

F.4. Services

### **Part III/III Addendum:**

X. Scripts and code snippets

X.1. Gadget init script

X.2. CamillaDSP start script

X.3. Toggle-switch the gadget between peripheral mode and host mode

Y. Useful system checks

Y.1. The UAC2 gadget

Y.2. CamillaDSP

Y.3. Services and Modules

Y.4. WiFi

Y.5. CPU

Z. Useful links

# Part I/III: The Basic Setup

Content:

- A. Install and update the basic OS
- B. Setup the UAC2 mode
- C. CamillaDSP
- D. Make things work together

## A: The basic Raspberry OS

As a very first step, install Raspberry Pi OS Lite (64-bit) onto a SD card, best by using the latest version of the Raspberry Pi Imager. Then startup the Raspberry Pi with it and run the update process.

Download:

[https://downloads.raspberrypi.org/imager/imager\\_latest.exe](https://downloads.raspberrypi.org/imager/imager_latest.exe)

**apt** - update the OS

```
# apt update  
# apt full-upgrade  
# reboot now
```

Comment:

Any kernel version < V.5.18 is flawed for audio gadget use. Actually (23/04) the imager still installs a flawed kernel V.5.15 in terms of audio gadget use. The additional update/upgrade process instead downloads and installs V.6.1.. Therefore, better do not omit the update procedure until a more recent kernel will be installed straight away.

## B. Setup the audio UAC2 / OTG

### B1: Set up the UDC and the DWC2 kernel module

**/boot/config.txt** - append to this file.  
# vi /boot/config.txt  
dtoverlay=dwc2,dr\_mode=peripheral

Check:  
\$ lsmod | grep dwc2  
must now show the dwc2 and the roles module

Check:  
\$ ls -R /sys/class/udc/\*  
Must now show the UDC (USB device controller) with it's specific directory tree

Check:  
\$ ls -R /sys/kernel/config/usb\_gadget/\*  
Must show the active configuration of the usb\_gadget

### B2: Create the audio gadget functionality

The gadget is set up by using the configs interface such as described in  
[https://www.kernel.org/doc/html/latest/usb/gadget\\_configs.html](https://www.kernel.org/doc/html/latest/usb/gadget_configs.html)

Basically ...

- the audio gadget will be declared
- the configuration(s) will be declared
- the functions will be declared
- the functions will be associated with their configurations
- the gadget will be enabled

**/home/yourdirectory/gadget\_init.sh** - create an executable shell script file for all the appropriate code  
# ...  
# ...

To keep this section slim, the ready-to-paste script code for this file is found in section X at the end of this document.

Gadget initialization:  
Running this executable shell script file will initialize the gadget.

Check:  
\$ cat /proc/asound/cards  
Must show the UAC2Gadget after the script has been run

Check:  
\$ aplay -l  
Show the resulting playback audio cards

Check:  
\$ arecord -l  
Show the resulting caption audio cards

References:  
<https://www.kernel.org/doc/html/v6.1/usb/gadget-testing.html>  
v6.1 refers to the actual linux kernel version running on the gadget

## C: CamillaDSP forever ...

### C1: Install CamillaDSP (CDSP) on your system

#### wget

Download the CDSP tar

```
$ wget
https://github.com/Henquist/camilladsp/releases/download/v1.0.3/camilladsp-
linux-aarch64.tar.gz
v1.0.3 refers to the CDSP version to download
```

#### apt

install bsdtar

```
# apt install libarchive-tools
```

#### bsdtar

Extract and install CDSP

```
$ bsdtar -xf camilladsp-linux-aarch64.tar.gz
# mv camilladsp /usr/local/bin
```

#### mkdir (optional and as a suggestion)

Set up a basic sub-directory tree for the CamillaDSP auxiliary files

```
$ cd /home/YourDirectory
$ mkdir _camilladsp
$ mkdir _camilladsp/coeffs
$ mkdir _camilladsp/configs
```

### C2: Configure CDSP to capture from the gadget's USB\_input

#### /home/YourDir/CDSP\_Dir/ConfigsDir/CDSP\_Config.yml

Make the values of the CDSP configuration file match with the gadget functions values of the gadget init script

```
$ vi /home/YourDir/CDSP_Dir/ConfigsDir/CDSP_Config.yml
```

```
...
devices
  capture_samplerate: nnnnn -->> has to match c_srate
  ...
  capture:
    channels: n -->> has to match (decoded masked) value of c_chmask
    device: hw:CARD=xyz -->> has to match input device name from $arecord -l
    format: Snn_LE -->> has to match c_ssize (xform bits to bytes)
    type: Alsa
  playback:
    ...
...
```

For e.g. c\_srate=44100, c\_chmask=3, c\_ssize=2, this might transform into ...

```
...
devices
  capture_samplerate: 44100
  ...
  capture:
    channels: 2
    device: hw:CARD=UAC2Gadget,DEV=0
    format: S16_LE
    type: Alsa
  playback:
    ...
...
```

## D. Make things work together

### D.1. Gadget mode autostart

Set up a service to initialize the audio gadget device(s) at boot

```
/usr/lib/systemd/system/gadget-init.service - create this new file
# vi /usr/lib/systemd/system/gadget-init.service

[Unit]
Description=Gadget Boot Starter
After=multi-user.target
[Service]
ExecStart=/home/YourDir/YourGadgetInitScriptName.sh
Group=root
User=root
[Install]
WantedBy=multi-user.target
```

### D.2. CamillaDSP autostart

Same goes for CamillaDSP. Make shure that CamillaDSP does start after the gadget init service has set up the audio gadget, because CamillaDSP has to find the gadget's functional audio device. Therefore, e.g. a delay function might be provided inside of the CamillaDSP start script. See section X.2. for this solution.

```
/usr/lib/systemd/system/camilladsp-start.service - create this new file
# vi /usr/lib/systemd/system/camilladsp-start.service

[Unit]
Description=CamillaDSP Boot Starter
After=multi-user.target
[Service]
CPUSchedulingPolicy=fifo
CPUSchedulingPriority=10
ExecStart=/home/YourDir/YourCamillaDSPStartScript.sh
Group=root
User=root
[Install]
WantedBy=multi-user.target
```

### D.3. Enabling these autostart services

You may enable these services at startup as one of the very last steps, e.g. after having tested the full functionality of your basic setup.

Fist check both your gadget init shell script and your CDSP startup shell script

Then, if your shell scripts run flawlessly, manually check the startup service by

```
# systemctl start AnyName.service
```

Then, if your startup services has proven to be functional, enable the startup services at boot

```
# systemctl enable AnyName.service
```

If something is wrong afterwards, for debug purpose disable the service autostart by ...

```
# systemctl disable AnyName.service
```

A running service can be stopped by ...

```
# systemctl stop AnyName.service
```

## Part II/III: Tweaks, Slimdowns and optional Goodies

Content:

E. Pimp your audio

F. Lean Ux - an attempt to get the whole system a wee bit leaner, lighter, faster

### E. Pimp your audio

#### E.1. Disable the default RaspberryPi onboard audio functionality if not needed

Apply these changes to disable the Headphones and the vc4hdmi audio devices which by standard are enabled.

**/boot/config.txt** - edit this file not to load some audio related modules

```
# vi /boot/config.txt
# dtoverlay=vc4-kms-v3d (comment this one out to disable the vc4hdmi device)
# dtparam=audio=on (comment this one out to disable the Headphones device)
```

Check:

```
$ cat /proc/asound/cards
```

The Headphones and the vc4hdmi audio devices should no longer figure

Comment:

For audio use only, you can comment everyting out in the /boot/config.txt except of

- arm\_64bit=1
- the sections for the Rpi 4

#### E.2. Set the CPU to a constant frequency

The regular ondemand CPU frequency switching governor may cause xruns during playback. This is because CPU frequency switching may come along with some ms of latency. Therefore, instead allow uncontrolled dynamic switching by the ondemand governor, better resort to the constant frequency performance and powersave governors to precisely toggle between high and low CPU demands.

**apt**

```
# apt install linux-cpupower
```

**/home/yourdirectory/camilladsp\_start.sh** - add some scripting

```
#!/bin/sh
#####
# camilladsp_start.sh
# RpiOS (Debian) / RPi 3+
#####
...
# CPU frequency min/max declaration
CPU_FREQ_LO=600000 # values in kHz!
CPU_FREQ_HI=1200000
...
function host_CPU_set() {
    # CPU frequency min/max values
    sudo cpupower frequency-set --min $CPU_FREQ_LO --max $CPU_FREQ_HI
    # Set CPU frequency to constant max value
    sudo cpupower frequency-set --governor performance
    sleep 0.5
    # Set CPU frequency to constant min value
    sudo cpupower frequency-set --governor powersave
}
```



```
...
host_CPU_set & # Must be run in background as a daemon
sleep 0.1
camilladsp configfile ...
...
```

Check:

```
# cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
# cat /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
```

Remark:

Starting a program might cause a high CPU load. Therefore for initializing, switch for a short time to a high CPU frequency, and then reduce the frequency for a more economic steady-state. As the ondemand scheduler is intended to do and would do. The above approach does exactly this, but then keeps the steady-state lower frequency to a constant value.

### E.3. Run audio programs (such as CDSP) on specific and optionally isolated CPU(s)

The following examples do isolate CPU 2 and 3 to independently run all threads of specified audio programs on these specific CPUs. Instead, let e.g. CPU 0 handle all IRQs

#### E.3.a. Assign programs to specific CPUs

**/home/yourdirectory/camilladsp\_start.sh** - add the taskset command to assign program and cpus

```
#!/bin/sh
#####
# camilladsp_start.sh
# RpiOS (Debian) / RPi 3+
#####
...
sudo taskset -c 2-3 camilladsp configfile ...
...
```

Comment:

This assigns all threads of CamillaDSP to CPUs 2 and 3. The system scheduler may still assign any other program to these CPUs. To have CamillaDSP run as only programs on these CPUs, you will have to isolate these CPUs. See section E.3.b.

Comment II:

See section E.3.c.

#### E.3.b. CPU isolation

This best works together with the CPU assignment method described in the section E.3.a. above

**/boot/cmdline.txt** - add some instructions to the existing textline:

```
# vi /boot/cmdline.txt
... irqaffinity=0 nohz=on isolcpus=2-3 nohz_full=2-3 rcu_nocbs=2-3
```

Check:

```
# cat /sys/devices/system/cpu/isolated
$ ps H -q $(pidof -s camilladsp) -o 'pid,tid,cls,rtprio,comm,pcpu,psr'
```

### E.3.c. Pros and Cons

There are pro's, con's and compromises in terms of CPU assignment and isolation, as CPU isolation might not only be beneficial. The regular system scheduler is very effective in distributing all load between all available CPU resources. Some audio programs such as CamillaDSP (CDSP) and also MPD run their process as several threads, and by standard all or them being more or less evenly distributed between all CPUs resources. Such minimizing the individual load per CPU, this allows the CPU to be operated at a lower frequency. In practice and as in one of the tested setups, CDSP runs as 3 threads and starts another 3 to 4 child threads. Now, squeezing all these threads onto a single CPU might indeed not be an optimal solution. The CPU frequency will need to be highish, or worst case the total load will even get too high for a single CPU to handle it. As a workaround and as described in the above setup, two CPUs (2 and 3) might be isolated and assigned to all of the threads of CDSP and it's child threads. This approach still isolates the audio processing while letting the scheduler balance the load between CPU2 and CPU3. Furthermore, in the above example, all IRQ handling is affined to CPU0.

## F. Lean Ux - an attempt to get the system a wee bit leaner, lighter, faster

These propositions are on a slightly slippery terrain. They work, but there might be better, simpler and more radical approaches.

### F.1. Save your SD card

#### F.1.a. - write temp files into a RAM disk instead onto the SD card

```
/etc/fstab - append several lines to this file
# vi /etc/fstab
tmpfs    /tmp      tmpfs    noatime,nodev,nosuid,size=25M 0 0
tmpfs    /var/lock tmpfs    noatime,nodev,nosuid,size=5M  0 0
tmpfs    /var/log  tmpfs    noatime,nodev,nosuid,size=5M  0 0
tmpfs    /var/run  tmpfs    noatime,nodev,nosuid,size=5M  0 0
tmpfs    /var/tmp  tmpfs    noatime,nodev,nosuid,size=5M  0 0
```

Comment:

In the same logic, do not set up a swap partition in fstab

#### F.1.b. - disable swapping to a file

```
dphys-swapfile
# dphys-swapfile swapoff
```

#### F.1.c. - minimize journaling

```
/etc/systemd/journald.conf - edit this file to minimize journaling
# vi /etc/systemd/journald.conf
Storage=none
SystemMaxUse=45M
RuntimeMaxUse=45M
ForwardToConsole=yes
TTYPath=/dev/null
MaxLevelStore=emerg
MaxLevelSyslog=emerg
MaxLevelKMsg=emerg
MaxLevelConsole=emerg
MaxLevelWall=emerg
ReadKMsg=no
Audit=no
```

#### F.1.d. - disable journaling on the ext4 file system

First insert the sd-card onto a suitable linux host and then, on the host's console ...

```
# tune2fs -O ^has_journal /dev/mmcblk0p2
```

Remark:

-O like Oscar, not -0 like Zero

Check:

```
# debugfs -R features /dev/mmcblk0p2 | grep has_journal
```

#### F.1.e. - uninstall useless programs (useless in terms of useless for your specific audio use)

Uninstalled programs do free space and will never be upgraded any longer. So this manages the SD card. Make your choice which programs you will get rid of.

**apt**

```
# apt purge xyz xyz xyz xyz xyz xyz xyz xyz xyz  
# apt autoremove
```

xyz such as ...

avahi-daemon

bash-completion

bind9-host

bluez

bluez-firmware

ca-certificates

dphys-swapfile

eject

file

gcc

gettext-base

gnupg

libcamera-apps-lite

libcamera0

man-db

manpages

ncurses-term

python3-libcamera

python3-v4l2

tasksel

usbutils

vim-common

zip

reinstall some leaner packages

```
# apt install xyz xyz xyz xyz
```

xyz such as ...

vim-tiny

Comment:

This might be an ongoing process. Every time an update/upgrade process will be performed, unexpected programs may appear in the update list you do not really want to keep.

## F2. Beef up your WiFi

### F.2.a. - make the dhcp startup some seconds faster

**/etc/dhcpd.conf** - append 'noarp' to this file to make dhcpd startup faster  
# vi /etc/dhcpd.conf  
noarp

### F.2.b. - set up iwd instead of (outdated and bloated) wpa-supPLICANT

#### Prepare iwd to start after the next boot

```
# apt install iwd
# systemctl enable iwd.service
# systemctl start iwd.service
# iwctl
- device list
- station wlan0 scan
- station wlan0 get-networks - your network should figure on the list
- exit
# vi /var/lib/iwd/YourNetworkSSID.psk - create a matching configuration file
[Security]
Passphrase=YourPassphrase
```

#### Disable wpa\_supplicant

```
# systemctl disable wpa_supplicant
# reboot now - system will reboot and establish a network connection through iwd
```

Check:

```
# systemctl | grep iwd
```

#### Get rid of wpa\_supplicant

```
# apt purge wpasupplicant
# apt autoremove
```

Comment:

As long as wpasupplicant is running, iwd will not set up a /var/lib/iwd/YourNetworkSSID.psk file. Therefore, first manually prepare a psk file for iwd to log into your network, then disable wpasupplicant, then reboot.

Comment II:

You will be astonished about all the files deleted when wpasupplicant is cleaned away, in total amounting to 35 MB disk space.

## F.3. Kernel modules - the good, the bad, and the uglies ...

Raspberry Pi OS Lite has a bunch of kernel modules loaded which are not needed for a specific audio gadget use. These unneeded modules may be excluded from loading statically, or be unloaded (and also reloaded) dynamically.

### F.3.a. Static no-loading and dynamic removing unwanted kernel modules methods

static no-loading procedure within a configuration file...

```
/etc/modprobe.d/blacklist.conf - create this new file
# vi /etc/modprobe.d/blacklist.conf
install xyz /bin/false
```

```
install xyz /bin/false
install xyz /bin/false
install xyz /bin/false
...
```

dynamic unloading of kernel modules within a shell script file ...

**/home/yourdirectory/camilladsp\_start.sh** - add some scripting

```
#!/bin/sh
#####
# camilladsp_start.sh
# RpiOS (Debian) / RPi 3+
#####
...
ulist="xyz xyz xyz xyz xyz xyz xyz xyz"
llist="xyz xyz xyz xyz"
# Unload the kernel modules before audio use
for m in $ulist; do
    modprobe -rf $m
done
...
# Then start your audio
camilladsp configfile ...
...
# After audio use, some kernel modules may be re-loaded again
for m in $llist; do
    modprobe $m
done
```

### **F.3.b. First - Kernel modules better to keep (the good)**

First a list of modules which seem not certain if it is appropriate to get rid off.

```
aes_arm64
aes_generic
af_alg
algif_aead
algif_hash
algif_skcipher
brcmfmac
brcmutil
cfg80211
cmac
gf128mul
ghash_generic
i2c_bcm2835
libaes
libcomposite
md4
md5
regmap_i2c
rfkill
snd
snd_bcm2835
snd_compress
snd_pcm
snd_pcm_dmaengine
snd_soc_bcm2835_i2s
snd_soc_core
```

snd\_soc\_rpi\_wm8804\_soundcard  
snd\_soc\_wm8804  
snd\_soc\_wm8804\_i2c  
snd\_timer  
u\_audio  
usb\_f\_uac2

### **F.3.c. Modules not needed for specific audio use**

These xyz's seem safe not to load or to unload

### sound related  
snd\_bcm2835  
snd\_soc\_hdmi\_codec

### video related  
bcm2835\_codec  
bcm2835\_isp  
bcm2835\_v4l2  
mc  
videobuf2\_bcm2835\_v4l2  
videobuf2\_common  
videobuf2\_dma\_contig  
videobuf2\_memops  
videobuf2\_vmalloc  
videobuf2\_v4l2  
videodev  
v4l2\_mem2mem

### tutti frutti  
backlight  
bluetooth  
bcm2835\_mmal\_vchiq  
btbcm  
btqca  
btsdio  
btrl  
drm  
drm\_panel\_orientation  
fuse  
garp  
ip\_tables  
ipv6  
llc  
stp  
uio  
uio\_pdrv\_genirq  
vc\_sm\_cma  
vc4  
x\_tables

## F.4. Services

### systemctl

Have as few services as possible active in the system

```
# systemctl mask xyz (=byebye forever ...)
```

xyz such as ...

apt-daily.service

apt-daily.timer

apt-daily-upgrade.service

apt-daily-upgrade.timer

avahi-daemon.service

console-setup.service

cron.service

dphys-swapfile.service

getty@tty1.service

keyboard-setup.service

man-db.timer

ModemManager

rng-tools-debian.service

rsyslog.service

systemd-journal-flush.service ??

systemd-journald.service

systemd-journald.socket

systemd-journald-audit.socket

systemd-journald-dev-log.socket

triggerhappy.service

triggerhappy.socket

Check:

```
$ systemctl | grep running
```

```
$ systemctl | grep active
```

Comment:

Some of these services cannot be inactivated by

```
# systemctl disable xyz.service
```

Therefore use of the irreversible

```
# systemctl mask xyz.service
```

instead

Some services may better be dynamically stopped on demand from within a script

**/home/yourdirectory/camilladsp\_start.sh** - add some scripting

```
#!/bin/sh
```

```
#####
```

```
# camilladsp_start.sh
```

```
# RpiOS (Debian) / RPi 3+
```

```
#####
```

```
...
```

```
slist="xyz xyz xyz xyz xyz xyz xyz xyz"
```

```
# Deactivate the services before audio use
```

```
for s in $slist; do
```

```
    systemctl stop $s.service
```

```
done
```

```
...
```

```
# Then start your audio
```

```
camilladsp configfile ...
```

```
...
```

xyz such as ...  
dhcpcd  
ifupdown-pre  
raspi-config  
rpi-eeprom-update  
systemd-modules-load  
systemd-random-seed  
systemd-timesyncd

## Part III/III: Addendum

### X. Scripts and code snippets

#### X.1. Gadget init script

Adapt the settings to match you setup and your preferences.

For the complete set of functions see:

<https://www.kernel.org/doc/Documentation/ABI/testing/configfs-usb-gadget-uac2>

For the basic procedure see:

[https://www.kernel.org/doc/html/latest/usb/gadget\\_configfs.html](https://www.kernel.org/doc/html/latest/usb/gadget_configfs.html)

**/home/yourdirectory/gadget\_init.sh** - create an executable shell file for all the following code

```
#!/bin/sh
#####
# gadget_init.sh
# RpiOS (Debian) / RPi 3+
#####

### Settings ###

# !!! The settings of the Directories and Functions sections must be adapted !!!

# Directories - !!! must be adapted !!!
CONFIGFS_ROOT=/sys/kernel/config # adapt to your machine
GDG_DIRNAME="audio-basic" # adapt as you like

# Basics - better not to be changed, because of the standard nature of the values set
BCD_DEVICE=0x0100 # v.1.0.0
BCD_USB=0x0200 # USB2
ID_VENDOR=0x1d6b # Linux Foundation
ID_PRODUCT=0x0104 # 0x0104 for Multi Functional Gadget / 0x0101 for Audio Gadget

# Strings - likely/optionally to be adapted (except the first one)
STRG_LANGUAGE=0x409 # no need to adapt - 0x409 is a standard value (for US English)
STRG_MANUFACTURER="My_Own_Setup_Gallery" # adapt as you like
STRG_PRODUCT="Audio_Gadget_44/48" # adapt as you like
STRG_SERIALNUMBER="000001" # adapt as you like

# Configuration(s) - likely/optionally to be adapted
CONFIGURATION_CNF_1="44/48_Basic" # adapt as you like

# Functions - !!! must be adapted to the audio format(s) to be processed !!!
AUDIO_CHANNEL_MASK_CAPTURE=3 # 1=Left 2=Right 3=Stereo 0=disables the device
AUDIO_CHANNEL_MASK_PLAYBACK=3
AUDIO_SAMPLE_RATES_CAPTURE=44100,48000
AUDIO_SAMPLE_RATES_PLAYBACK=44100,48000
AUDIO_SAMPLE_SIZE_CAPTURE=2 # 1 for S8LE / 2 for S16LE / 3 for S24LE / 4 for S32LE
AUDIO_SAMPLE_SIZE_PLAYBACK=2

### Load the required kernel modules (and ev. overlays) ###
```



```

# libcomposite
modprobe libcomposite

### create the gadget ###

# create the gadget directory and change into it
cd "${CONFIGFS_ROOT}"/usb_gadget
mkdir -p $GDG_DIRNAME
cd $GDG_DIRNAME

# basics
echo $BCD_DEVICE > bcdDevice
echo $BCD_USB > bcdUSB
echo $ID_VENDOR > idVendor
echo $ID_PRODUCT > idProduct

# strings
mkdir -p strings/$STRG_LANGUAGE
echo $STRG_SERIALNUMBER > strings/$STRG_LANGUAGE/serialnumber
echo $STRG_MANUFACTURER > strings/$STRG_LANGUAGE/manufacture
echo $STRG_PRODUCT > strings/$STRG_LANGUAGE/product

# configuration(s)
mkdir configs/c.1 # index mandatory for every configuration
mkdir -p configs/c.1/strings/$STRG_LANGUAGE
echo $CONFIGURATION_CNF_1 > configs/c.1/strings/$STRG_LANGUAGE/configuration

# functions
mkdir -p functions/uac2.usb0
echo $AUDIO_CHANNEL_MASK_CAPTURE > functions/uac2.usb0/c_chmask
echo $AUDIO_SAMPLE_RATES_CAPTURE > functions/uac2.usb0/c_srate
echo $AUDIO_SAMPLE_SIZE_CAPTURE > functions/uac2.usb0/c_ssize
echo $AUDIO_CHANNEL_MASK_PLAYBACK > functions/uac2.usb0/p_chmask
echo $AUDIO_SAMPLE_RATES_PLAYBACK > functions/uac2.usb0/p_srate
echo $AUDIO_SAMPLE_SIZE_PLAYBACK > functions/uac2.usb0/p_ssize

# associate functions to configurations
ln -s functions/uac2.usb0 configs/c.1/

# enable the gadget
ls /sys/class/udc > UDC

```

## X.2. CamillaDSP start script

As both the gadget init script and the CamillaDSP start script will be run after the same systemd target, you must ensure that CamillaDSP is starting after the gadget audio device is fully functional. Therefore, you may insure the correct timing at the script level. Something like this:

**/home/yourdirectory/camilladsp\_start.sh**

```

#!/bin/sh
#####
# camilladsp_start.sh
# RpiOS (Debian) / RPi 3+
#####

...
GDGT_SNDCARD="UAC2Gadget" # Must match the name of the gadget soundcard device
...

...
# Make shure the gadget soundcard device is functional before starting CamillaDSP
while true; do
    grep $GDGT_SNDCARD /proc/asound/cards
    if [ $? = 1 ]; then
        continue
    fi
done

```

```
...
```

```
...  
# And now start CamillaDSP  
camilladsp ...  
...
```

### X.3. Toggle-switch the gadget between peripheral mode and host mode

This one is interesting, but untested. For a discussion of this basic principle see <https://forums.raspberrypi.com/viewtopic.php?t=246348#p1622016>

**/home/yourdirectory/camilladsp\_start.sh** - add some scripting

```
#!/bin/sh  
#####  
# camilladsp_start.sh  
# RpiOS (Debian) / RPi 3+  
#####  
...  
Switch to peripheral mode:  
...  
# modprobe -r dwc2  
# dtoverlay -r dwc2  
# dtoverlay dwc2 dr_mode=peripheral  
# modprobe dwc2  
...  
# Then start your audio  
camilladsp configfile ...  
...  
Switch back to host mode:  
...  
# modprobe -r dwc2  
# dtoverlay -r dwc2  
# dtoverlay dwc2 dr_mode=host  
# modprobe dwc2  
...
```

## Y. Useful system checks

### Y.1. The UAC2 gadget

```
$ lsmod | grep dwc2  
$ ls -R /sys/class/udc/*  
$ cat /proc/asound/cards  
$ aplay -l  
$ arecord -l  
$ ls -R /sys/kernel/config/usb_gadget/*
```

### Y.2. CamillaDSP

```
$ pidof camilladsp  
$ ps -q $(pidof -s camilladsp) -eLo 'pid,tid,cls,rtprio,comm,pcpu,psr'
```

### Y.3. Services and Modules

```
$ systemctl | grep running  
$ systemctl | grep active  
  
$ lsmod
```

## Y.4. WiFi

```
$ ip route  
$ systemctl | grep iwd
```

## Y.5. CPU

```
# cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq  
# cat /sys/devices/system/cpu/cpufreq/policy0/scaling_governor  
# cat /sys/devices/system/cpu/isolated  
  
...
```

## Y.2314298 The little red LED there

```
# isitlit
```

## Z. Useful Links

<https://www.kernel.org/doc/Documentation/ABI/testing/configfs-usb-gadget-uac2>  
[https://www.kernel.org/doc/html/latest/usb/gadget\\_configfs.html](https://www.kernel.org/doc/html/latest/usb/gadget_configfs.html)  
<https://www.kernel.org/doc/html/v6.1/usb/gadget-testing.html>

<https://www.diyaudio.com/community/threads/linux-usb-audio-gadget-rpi4-otg.342070/>  
<https://www.collabora.com/news-and-blog/blog/2019/02/18/modern-usb-gadget-on-linux-and-how-to-integrate-it-with-systemd-part-1/>

[https://downloads.raspberrypi.org/imager/imager\\_latest.exe](https://downloads.raspberrypi.org/imager/imager_latest.exe)  
<https://github.com/Henquist/camilladsp/releases/download/v1.0.3/camilladsp-linux-aarch64.tar.gz>

<https://www.diyaudio.com/community/threads/camilladsp-cross-platform-iir-and-fir-engine-for-crossovers-room-correction-etc.349818/>

29.3.2023

Daihedz, a poster of diyaudio.com