

# Set up a Linux Audio DSP UAC2 Rpi3A+ Gadget

Version:

23/10/14 – Part II/V E.WiFi Options have been rewritten to meet the changes of the Debian bookworm release.

Topic:

Following Part I of this HowTo, it will take approx. 30 minutes to set up a basic UAC2 gadget. Be aware that despite starting from the "lite" version of the Raspberry Pi OS, the resulting system will include many standard OS features not really needed for an audio gadget setup. This quick and standard setup will perform flawlessly and will satisfy most needs of most users.

Part II of this HowTo will deal with some advanced tweaking of the OS. Part III deals with CPU tweaking options potentially useful for audio. Part IV contains a detailed description of a more complex setup including all goodies mentioned before in Parts II and III. Part V is a collection of script listings, diagnostic commands and other useful stuff. In section V there also is a complete description of a highly optimized UAC2 gadget.

System Actuality:

Raspberry Pi OS imager 1.7.5 / Debian Bookworm V.6.1 / CamillaDSP V.2

Ingredients:

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

Rpi3A+ vs. Rpi4/5:

In this above combination along with a power-economic Rpi3A+, the resulting UAC2 Gadget can easily be operated by connecting it from its single USB-A port to another single host USB port for both data transfer and power supply. The overall processing resources of a Rpi3A+ are plenty enough even to allow CPU clock frequency throttling. Instead, resorting to a more power hungry Rpi4/5 will imply the need of a dedicated power supply.

About the formatting of this HowTo:

# Code is formatted in this liberation mono typeset

Text and code snippets formatted *LikeThis* is either *UserDefined*, *SystemSpecific* or *ChangingOverTime*, like *HomeDirectoryTrees* or *ReleaseVersions*. You will have to adapt it to your system and needs.

For further improvement's sake, please ... :

Feedback is welcomed, and probably best posted into p.hofman's the very informative diyaudio rpi4 otg thread for further general discussion: <https://www.diyaudio.com/community/threads/linux-usb-audio-gadget-rpi4-otg.342070/>

Disclaimer:

DSP might destroy your audio equipment and damage your auditory system by accidental excessive loudness if not correctly set up. Proceed at you own risk.

# La Direttissima

Barebone UAC2 gadget:

## 1. **/boot/config.txt**

```
$ sudo vi /boot/config.txt  
→ add this statement: dtoverlay=dwc2,dr_mode=peripheral
```

## 2. **/home/pi/gadget.sh**

```
$ touch /home/pi/gadget.sh  
$ chmod u+x /home/pi/gadget.sh  
→ copy and paste scripting code from part IV/L1  
$ sudo /home/pi/gadget.sh
```

Autostart the UAC2 gadget mode at boot:

## **/lib/systemd/system/gadget.service**

```
$ sudo touch /lib/systemd/system/gadget.service  
→ copy and paste scripting code from part IV/L3a  
$ sudo systemctl enable /lib/systemd/system/gadget.service
```

Autostart CamillaDSP mode at boot:

## **/lib/systemd/system/gdsp.service**

```
$ sudo touch /lib/systemd/system/gdsp.service  
→ copy and paste scripting code from part IV/L3b  
$ sudo systemctl enable /lib/systemd/system/gdsp.service
```

```
$ sudo reboot now
```

# Content

## Part I/V - The basic DSP UAC2 gadget setup

- A. Install and update the OS
- B. Setup the UAC2 mode
  - B.1. Set up the UDC and the DWC2 kernel module
  - B.2. Create the audio gadget functionality
- C. CamillaDSP
  - C.1. Install CamillaDSP (CDSP) on your system
  - C.2. Configure CDSP to capture from the gadget's USB\_input
  - C.3. First functional system test
- D. Make things work together and autostart on bootup
  - D.1. UAC2 gadget mode service
  - D.2. Audio processing service
  - D.3. Make start the gadget.service and the gdsp.service at host boot

## Part II/V - General (non Audio) OS tweaks and options

- E. WiFi options
  - E.1. Set up iwd instead of WPA-supPLICANT
- F. Programs - uninstall the useless ones
- G. SD card - reduce it's dynamic wear
  - G.1. Write temp and dynamic files into a RAM disk instead onto the SD card
  - G.2. Disable swapping to a file
  - G.3. Minimize journaling
  - G.4. Disable journaling on the ext4 file system
- H. OS functionality - reduce the overall workload
  - H.1. Kernel modules - the good, the bad, and the uglies ...
  - H.2. Systemd units - services, sockets, timers, targets ...

## Part III/V - Audio related tweaks and options

- I. CPU settings and allocations
  - I.1. Set the CPU to a constant frequency
  - I.2. Run audio programs (such as CDSP) on specific and optionally isolated CPU(s)

## Part IV/V - A real world functional system setup

- K. The testbed / system description
- L. The scripts
  - L.1. Gadget initialization script
  - L.2. Two Service init files to start the functionality at bootup

## Part V/V - Addendum

- X. Listings
  - X.1. Programs
  - X.2. Kernel modules
  - X.3. Systemd units
- Y. Debug and system checks
- Z. Useful links

# Part I/V: The basic DSP UAC2 gadget 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 current available version of the Raspberry Pi Imager. Enable WiFi and SSH for the management of the Pi as a headless host. For this HowTo, a standard user name "pi" is assumed. Then startup the Raspberry Pi, log into it and run the update process.

Download from the Raspberry Pi OS website:

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

**apt**

run this command to update the OS

```
$ sudo apt update
$ sudo apt upgrade
$ sudo reboot now
```

## B. Setup the audio UAC2 / OTG

Maybe an appealing in-depth lecture to this topic first:

<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://www.collabora.com/news-and-blog/blog/2019/03/27/modern-usb-gadget-on-linux-and-how-to-integrate-it-with-systemd-part-2/>

### B.1.: Set up the UDC and the DWC2 kernel module

**/boot/config.txt**

Edit the RaspberryOS configuration file

The config.txt must contain this line:

```
dtoverlay=dwc2,dr_mode=peripheral
```

For a minimal headless approach, the config.txt might contain only these 3 lines:

```
arm_64bit=1
dtoverlay=dwc2,dr_mode=peripheral
gpu_mem=16
```

To preserve Rpi onboard audio devices functional instead, eventually keep any or both of these two lines:

```
dtoverlay=vc4-kms-v3d (enables the vc4hdmi device)
dtparam=audio=on (enables the Headphones device)
```

Checks:

```
$ lsmod | grep dwc2
```

This must now show the dwc2 and the roles module

```
$ ls -R /sys/class/udc/*
```

Must now show the UDC (USB device controller) with it's specific directory tree

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

The Headphones and the vc4hdmi audio devices might figure or might not figure

At this point, the gadget itself is not configured yet. Therefore it does not appear as an alsa soundcard device.

## B.2.: Create the audio gadget functionality

The gadget is set up by using the up-to-date configfs interface (the g\_audio approach is deprecated)

- 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/pi/gadget.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 part IV/L1 of this HowTo. Running this script file will initialize the gadget. You may adapt some parameters for your own settings and needs.

As for all shell files, in order to run this file as a command, it must be declared as executable

```
$ chmod u+x /home/pi/gadget.sh
```

Then run this file as root to create the UAC2Gadget

```
$ sudo /home/pi/gadget.sh
```

Checks:

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

This must show the UAC2Gadget after the script has been run

```
$ aplay -l
```

Show the resulting playback audio cards

```
$ arecord -l
```

Show the resulting capture audio cards

```
$ ls -R /sys/kernel/config/usb_gadget/*
```

Must show the active configuration of the usb\_gadget

Reference - must read:

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

Reference - for the complete set of functions see:

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

Reference - further reading:

<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 (CDSP) forever ...

From now on in the further text notes, “CamillaDSP” will be abbreviated as “CDSP”

### C.1.: Install CamillaDSP on your system

#### **wget**

run this command to download the CDSP tar

```
$ wget https://github.com/Henquist/camilladsp/releases/download/v2.0.1/camilladsp-Linux-aarch64.tar.gz
```

v2.0.1 refers to the CDSP version to download

#### **apt**

run this command to install bsdtar

```
$ sudo apt install libarchive-tools
```

#### **bsdtar**

run this command to extract and install CDSP

```
$ bsdtar -xf camilladsp-Linux-aarch64.tar.gz
$ sudo mv camilladsp /usr/local/bin
```

#### **mkdir**

run this command to set up a basic sub-directory tree for the CDSP auxiliary files. This is optional.

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

Check:

```
$ camilladsp
```

### C.2.: Configure CDSP to capture from the gadget's USB\_input

/home/pi/\_camilladsp/configs/CDSP\_Config.yml

edit the CDSP configuration file

Make the values of the CDSP configuration file match with the gadget functions values of the /home/pi/gadget.sh init script. For e.g. c\_rate=44100, c\_chmask=3, c\_ssize=2 might translate into ...

```
$ vi /home/pi/_camilladsp/configs/CDSP_Config.yml
...
devices
  capture_samplerate: 44100
  ...
  capture:
    channels: 2 # (chmask=3)
    device: hw:CARD=UAC2Gadget,DEV=0
    format: S16_LE # (c_ssize=2)
    type: Alsa
  playback:
    ...
...
```

### C.3. First functional system test

At this point, a basic functional system is already set up. Test it for it's correct, intended function.

```
$ sudo reboot now
```

```
$ sudo /home/pi/gadget.sh
$ camilladsp /home/pi/camilladsp/configs/CDSP_Config.yml
```

If the system does not behave as expected, some debugging might be appropriate before going further.

### **/home/pi/gdsp.sh**

Create this file

GDSP stands for GadgetDSP. For convenience, the audio startup process might be implemented into a shell script to allow for some extras. An example of such (a bit more elaborate file) is described in part IV/L2 of this HowTo.

## **D. Make things work together and autostart on bootup**

### **D.1. UAC2 gadget mode service**

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

#### **/usr/lib/systemd/system/gadget.service**

create this new file

```
$ sudo vi /usr/lib/systemd/system/gadget.service
...
[Service]
ExecStart=/home/pi/gadget.sh
Group=root
User=root
...
```

To keep this section slim, the ready-to-paste script code for this file is found in part IV/L3 of this HowTo, named/labeled as gadget.service. You may adapt some parameters for your own settings and needs.

Check:

```
$ sudo systemctl start gadget.service
```

This must have the same effect as starting gadget.sh as root - It must create and configure the UAC2 gadget

### **D.2. Audio processing service**

Set up a service to start the gadget audio processing at boot.

#### **/usr/lib/systemd/system/gdsp.service**

create this new file

```
$ sudo vi /usr/lib/systemd/system/gdsp.service
...
[Service]
ExecStart=/home/pi/gdsp.sh
Group=root
User=root
...
```

To keep this section slim, the ready-to-paste script code for this file is found in part V/L3 of this HowTo, named/labeled as gdsp.service. You may adapt some parameters for your own settings and needs.

Make sure that the audio processing does start after the gadget init service has set up the audio gadget: CDSP has to find the gadget's functional audio device. Otherwise, it will crash. Therefore, inside of the /home/pi/gdsp.sh start script there first is a check for the correct functioning of the gadget device before then starting up the audio processing by CDSP. See Part IV/L2 for this kind of solution.

Check:

```
$ sudo systemctl start gdsp.service
```

This must have the same effect as starting gdsp.sh as root - It must start the DSP processing.

### D.3. Make start the gadget.service and the gdsp.service at host boot

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. First check both your /home/pi/gadget.sh and your /home/pi/gdsp.sh shell scripts by test-starting them inside a root console, and then control the outcome.

Then, if both scripts run flawlessly, manually test-start the gadget.service and the gdsp.service by

```
$ sudo systemctl start gadget.service  
$ sudo systemctl start gdsp.service
```

If these two services has proven to be functional, enable them at boot

```
$ sudo systemctl enable foo.service
```

If something goes wrong and for debug purpose, you may disable the boot-time startup behavior by ...

```
$ sudo systemctl disable foo.service
```

Running services can be stopped by ...

```
$ sudo systemctl stop foo.service
```



# Part II/V: General (non Audio) OS tweaks and options

Content:

- E. WiFi options
- F. Programs - uninstall the useless ones
- G. SD card - reduce it's wear
- H. OS features - reduce the workload

## E. WiFi options

### E.1. Set up iwd instead of WPA-supPLICANT

WPA-supPLICANT depends on many dependencies and occupies 35MB of disk space in total. IWD instead is much leaner.

See <https://wiki.debian.org/NetworkManager/iwd>

#### Prepare iwd

```
$ sudo apt install iwd
$ iwctl
- station list
- station wlan0 scan
- station wlan0 get-networks
- exit
```

This will create the /var/lib/iwd path for the iwd configuration file

**/var/lib/iwd/dhccpd.YourNetworkSSID.psk**

create a matching configuration file

```
$ sudo vi /var/lib/iwd/YourNetworkSSID.psk
...
[Security]
Passphrase=YourPassphrase
...
```

Type the passphrase as is and not within quotes: E.g. like this, and not “like this”

**/etc/NetworkManager/conf.d/iwd.conf**

Make NetworkManager use iwd as backend

```
$ sudo vi /etc/NetworkManager/conf.d/iwd.conf
...
[device]
wifi.backend=iwd
...
```

#### Enable iwd, disable WPA-supPLICANT and reboot

```
$ sudo systemctl enable iwd
$ sudo systemctl disable wpa_supplicant
$ sudo systemctl stop NetworkManager
```

The connection will get lost by the last command. Make the Pi reboot to establish a new ssh connection.

Check:

```
$ systemctl | grep iwd
$ systemctl | grep wpa_supplicant
```

#### Get rid of the WPA-supPLICANT installation

```
$ sudo apt purge wpasupplicant
```

```
$ sudo apt autoremove
$ sudo rm -rf /etc/wpa_supplicant
```

### Use the iwd inbuilt DHCP client instead of the regular isc-dhcp-client

```
/etc/iwd/main.conf
$ sudo vi /etc/iwd/main.conf
...
[General]
EnableNetworkConfiguration=True
...

$ sudo apt remove isc-dhcp-client
$ sudo rm -rf /etc/dhcp
```

## F. Uninstall useless programs (useless for your specific audio use)

Uninstalled programs do free space and will never be upgraded any longer. So this manages the SD card. Uninstalled programs don't appear as running services either. Make your choice which programs you will get rid of.

```
apt
run apt to uninstall some programs

$ sudo apt purge foo foo foo foo
$ sudo apt autoremove
```

For a list of candidates for removal see part V/X.1. of this HowTo

You may also replace some programs by leaner versions, such as vim-tiny instead of vim-common.

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.

## G. Reduce the wear of the SD Card

Every write process wears the SD Card. Therefore, minimizing writes will lead to a longer life of the SD card.

### G.1. Write temp and dynamic files into a RAM disk instead onto the SD card

```
/etc/fstab
append several lines to this file

$ sudo vi /etc/fstab
...
tmpfs /tmp          tmpfs noatime,nodev,nosuid,size=5M 0 0
tmpfs /var/lock       tmpfs noatime,nodev,nosuid,size=5M 0 0
tmpfs /var/log       tmpfs noatime,nodev,nosuid,size=50M 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

### G.2. Disable swapping to a file

dphys-swapfile

run this command

```
$ sudo dphys-swapfile swapoff
```

### G.3. Minimize journaling

**/etc/systemd/journald.conf**

edit this file to minimize journaling

```
$ sudo vi /etc/systemd/journald.conf
...
Storage=none
MaxLevelStore=emerg
MaxLevelSyslog=emerg
MaxLevelKMsg=emerg
MaxLevelConsole=emerg
MaxLevelWall=emerg
ReadKMsg=no
Audit=no
...
```

### G.4. Disable journaling on the ext4 file system

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

```
$ sudo tune2fs -O ^has_journal /dev/mmcblk0p2
```

Remark:

-O like Oscar, not -0 like Zero

Check:

```
$ sudo debugfs -R features /dev/mmcblk0p2 | grep has_journal
```

## H. The OS - reduce it's overall workload

A fresh install of Raspberry Pi OS lite 64Bit comes along with some 60 kernel modules loaded and some 140 active systemd units while 30 of them running at steady-state conditions. Most of these resources are not needed for the purpose of USB gadget audio DSP. Two options of unloading/stopping these programs are described - a dynamic one, and a static one. The dynamic approach will demand some processing time at every startup, but offers some flexibility and preserves the full functionality of the OS for maintenance. Instead, in the static approach, the modules may be statically prevented from loading at boot time.

In this HowTo, some lists of trash/keep candidates are already set up. These lists may need some individualization on other system. Any list of the trash or keep candidates may be iteratively gained by a trial-and-error method: Have your system run the application, and then one by one unload modules and disable services. If the application crashes, then you disabled a module or service (unit) to keep. Reset and go on trying ...

### H.1. Kernel modules - the good, the bad, and the uglies ...

#### H.1.a. The dynamic method

At boot, all standard modules get loaded. Then, the unwanted/unneeded kernel modules get unloaded within a shell script file before audio processing starts.

**/home/pi/gdsp.sh**

The dynamic method is implemented within a function named `os_kernel_trim()`

See part IV/L2 of this HowTo for the details.

Central part is the declaration of list of modules to keep. See part V/X.2.b. of this HowTo for a "keep" list. This list is declared as variables in the head section of the coding of /home/pi/gdsp.sh. Also all other variables are declared there in the head section.

Then, in the middle part of the script, where the functions are defined, there is the function

```
function os_krnl_trim()
```

This function implements the functionality to unload all loaded modules which do not figure in this list.

### H.1.b. The static method

The static approach is to avoid loading the kernel modules at boot time as part of the (static) system configuration.

**/etc/modprobe.d/blacklist.conf**

create this new file

```
$ sudo vi /etc/modprobe.d/blacklist.conf
...
install foo /bin/false
install foo /bin/false
...
```

See part V/X.2.a. of this HowTo for a "trash" list of **modules to exclude** from being loaded

### H.1.c. Rpi onboard sound device related kernel modules

Depending on enabling or disabling the onboard sound hardware, you may include or exclude the following two kernel modules:

Sound jack connector (Headphones):

snd\_bcm2835

HDMI sound output:

snd\_soc\_hdmi\_codec

## H.2. Systemd units - services, sockets, timers, targets ...

### H.2.a. The dynamic method

For the systemd units, there is a good starting point: While the system is functional in steady-state, any running unit is a candidate for the category which is to be kept running:

```
$ systemctl | grep "loaded active running" | cut -d"." -f1
```

Then you may compare this list against all units in "active" state

```
$ systemctl | grep "loaded active" | cut -d"." -f1
```

You may then stop all "loaded active" units not being in "loaded active running" state. This already does a very good job, without any good-bad-ugly list. If you need further tuning, you may declare units to keep and units to stop.

**/home/pi/gdsp.sh**

The dynamic method is implemented within a function named os\_sysd\_trim()

See part IV/L2 of this HowTo for the details.

### **H.2.b. The static approach**

The static approach is to avoid starting the services at boot time as part of the (static) system configuration, by masking the service.

```
$ sudo systemctl mask foo.service (=byebye forever ... unless you unmask them again)
```

or

```
$ sudo systemctl disable foo.service
```

See part V/X.3. of this HowTo for a list of services to stop from being started

Checks:

```
$ systemctl | grep "loaded active running"
```

```
$ systemctl | grep "loaded active"
```

# Part III/V: Audio related tweaks and options

Content:

I: CPU settings

## I. CPU settings and allocations

### I.1. 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**

run this command

```
$ sudo apt install linux-cpupower
```

**/home/pi/gdsp.sh**

add some scripting

```
#!/bin/sh
...
### Variables declaration part of script
...
# CPU frequency min/max declaration
CPU_FREQ_LO=600000 # values in kHz!
CPU_FREQ_HI=1200000
...
### Functions part of the script
...
function CPU_freq_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
}
...
### Main program flow part of the script
...
host_CPU_set & # Must be run in background as a daemon
sleep 0.1
camilladsp config.yml
...
```

Checks:

```
$ sudo cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
```

```
$ sudo 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.

Remark II:

The actual implementation is a slightly different and a bit more elaborate, but follows the same principle. Have a look at the function `cpu_freq_set` in Part IV/L2 of this HowTo

## I.2. Run audio programs (such as CDSP) on specific and optionally isolated CPU(s)

The following examples do isolate CPU 1,2 and 3 to independently run all threads of specified audio programs on these specific CPUs. Instead, let e.g. CPU 0 handle all IRQs

### I.2.a. Assign programs to specific CPUs

**/home/pi/gdsp.sh**

edit this file, use the `taskset` command to assign program and CPUs

```
#!/bin/sh
...
sudo taskset -c 1-3 camilladsp configfile ...
...
```

Comment:

This assigns all threads of CDSP to CPUs 1,2 and 3. The system scheduler may still assign any other program to these CPUs. To have CDSP 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.

### I.2.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 text line:

```
$ sudo vi /boot/cmdline.txt
...
... irqaffinity=0 nohz=on isolcpus=1-3 nohz_full=1-3 rcu_nocbs=1-3
...
```

Checks:

```
$ sudo cat /sys/devices/system/cpu/isolated
```

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

### I.2.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 and threads between all available CPU resources. Some audio programs such as CDSP and also MPD run theirs 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, three CPUs (1, 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 CPU 1, 2 and 3. Furthermore, in the above example, all IRQ handling is affined to CPU0.

# Part IV/V: A real-world system setup

## K. The Testbed

The hardware consists of a Raspberry Pi 3A+ with a HifiBerry Digi2-Pro (HW 2.2) board. Data connection and power supply is both performed via one and the same USB\_A to USB\_A cable attached to the host computer. There is no additional supply through the USB\_B connector needed.

The Raspberry Pi OS lite installation is best performed by the Raspberry Pi Imager, SSH and WiFi enabled for a headless system inside of an accessible WiFi zone.

Modifications on standard OS files:

### **/boot/cmdline.txt**

contains the kernel isolation statements described in part III of this setup

```
... irqaffinity=0 nohz=on isolcpus=1-3 nohz_full=1-3 rcu_nocbs=1-3
```

### **/boot/config.txt**

contains nothing but

```
arm_64bit=1
dtoverlay=dwc2,dr_mode=peripheral
gpu_mem=16
```

New and application specific files:

### **/home/pi/gadget.sh**

content see below

### **/home/pi/gdsp.sh**

content see below

### **/usr/lib/systemd/gadget.service**

content see below

### **/usr/lib/systemd/gdsp.service**

content see below

Two directories for the DSP functionality:

### **home/pi/camilladsp/coeffs**

contains the fir filters files

### **home/pi/camilladsp/configs**

contains the CDSP configuration file(s)



## L. The Scripts

### L.1. /home/pi/gadget.sh - the gadget init script

You may/must adapt the 10 red parameter values for your own settings and needs.

```
#!/bin/sh
#####
# gadget.sh
# Audio Gadget setup
# RpiOS (Debian) / RPi 3+
#####
#
# Daihedz
# 23/06/10/ 00:00

### Settings ###

# 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 values are standard
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="DIYAudio" # adapt as you like
STRG_PRODUCT="UAC2Gadget4448" # adapt as you like
STRG_SERIALNUMBER="000001" # adapt as you like

# Configuration(s) - likely/optionally to be adapted
CONFIGURATION_CNF_1="UAC2Config4448" # 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/manufacturer
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
```

## L.2. The systemd service init scripts

### L.2.a. `/usr/lib/systemd/gadget.service` - the gadget init autostart service file

You may/must adapt the 2 red parameter values for your own settings and needs.

```
[Unit]
Description=UAC2 Gadget init
After=multi-user.target

[Service]
ExecStart=/home/pi/gadget.sh
Group=root
User=root

[Install]
WantedBy=multi-user.target
```

### L.2.b. `/usr/lib/systemd/gdsp.service` - the gadget DSP start script autostart service file

You may/must adapt the 2 red parameter values for your own settings and needs. You may e.g. want to change the statement `ExecStart` to `ExecStart=camilladsp camillaconfigfile.yml` for a direct start of CDSP.

```
[Unit]
Description=CamillaDSP audio processing starter
After=multi-user.target

[Service]
CPUSchedulingPolicy=fifo
CPUSchedulingPriority=10
ExecStart=/home/pi/gdsp.sh
Group=root
User=root

[Install]
WantedBy=multi-user.target
```

# Part V/V: Addendum

## X. Lists of items

Take care: All these lists may be either incomplete, or contain units not belonging into them.

### X.1. Programs to optionally remove

As described in part II/F:

avahi-daemon bash-completion bluez bluez-firmware ca-certificates dphys-swapfile eject file gcc gettext-base gnupg libcamera-apps-lite man-db manpages ncurses-term ppp tasksel usbutils zip

Comment: Getting rid of all these above listed programs and theirs dependencies will free more than 150MB of disk space. Running services drop by half from approx. 30 to approx. 15.

### X.2. Kernel modules

As described in part II, section H.1.

#### X.2.a Kernel modules to eventually unload

```
### 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
```

```
### Tuttifrutti
aes_arm64 aes_generic af_alg algif_aead algif_hash algif_skcipher backlight bluetooth bcm2835_mmal_vchiq btbcm
btqca btsdio btrl cbc ccm ctr cmac drm drm_panel_orientation fuse garp gcm gf128mul ghash_generic hmac
ip_tablesip6 libaes llc md4 md5 raspberrypi_hwmon stp uio uio_pdrv_genirq vc_sm_cma vc4 x_tables
```

#### X.2.b Kernel modules to eventually keep loaded

```
### basic OS modules
roles i2c_bcm2835 libcomposite regmap_i2c
```

```
### gadget related modules
dwc2 usb_f_uac2 u_audio
```

```
### network modules
brcmfmac brcmutil cfg80211 rfkill
```

```
### audio/sound modules
snd snd_bcm2835 snd_compress snd_pcm snd_pcm_dmaengine snd_soc_bcm2835_i2s snd_soc_core snd_timer
```

```
### SPDIF-out modules (HifiBerry Digi2 pro)
snd_soc_rpi_wm8804_soundcard snd_soc_wm8804 snd_soc_wm8804_i2c
```

### X.3. Systemd units (services, sockets, timers) not to start

As described in part II, section H.2.

```
### services to stop
apt-daily apt-daily-upgrade avahi-daemon console-setup cron dhphys-swapfile ifupdown-pre keyboard-setup kmod-
static-nodes man-db ModemManager raspi-config rng-tools-debian rpi-eeeprom-update rsyslog systemd-journal-flush
```

systemd-journald systemd-random-seed systemd-remount-fs systemd-timesyncd systemd-udevd systemd-udev-trigger  
systemd-update-utmp triggerhappy

### sockets to stop

e2scrub\_all fstrim logrotate systemd-fsckd systemd-initctl systemd-journald systemd-journald-audit systemd-journald-dev-log systemd-rfkill systemd-tmpfiles-dmp systemd-udevd-control systemd-udevd-kernel

### targets to stop

cryptsetup nfs-client sockets swap time-set time-sync

### timers to stop

apt-daily apt-daily-upgrade e2scrub\_all fstrim logrotate systemd-tmpfiles-clean systemd-tmpfiles-dmp

## Y. Useful system checks

### Y.1. The UAC2 gadget functionality

```
$ 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

```
$ sudo cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
$ sudo cat /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
$ sudo cat /sys/devices/system/cpu/isolated
```

...

### Y.2314298 The little red LED there

```
$ isitlit
```

## Z. Useful Links

### Basics/Theory

<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>

### Practical audio gadget setup

<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://www.collabora.com/news-and-blog/blog/2019/03/27/modern-usb-gadget-on-linux-and-how-to-integrate-it-with-systemd-part-2/>

### Software/Programs

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

<https://github.com/Henquist/camilladsp/releases/download/v2.0.1/camilladsp-linux-aarch64.tar.gz>

### Forums further discussion

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

### Rpi configuration

[https://www.raspberrypi.com/documentation/computers/config\\_txt.html](https://www.raspberrypi.com/documentation/computers/config_txt.html)

Special thanks to H.Enquist (CamillaDSP) and P.Hofman (ALSA/Linux expertise) and to many others having contributed

15.10.2023

Daihedz, a poster of diyaudio.com 's