Archphile Manual
V.0.11.2

(minimum requirements – Archphile 0.99.72 alpha)

Authored by Mike Andonov
info@archphile.org

Last edited: 20/05/2018

Table Of Contents

Introduction

Archphile (http://archphile.org) is an **ArchlinuxARM** (**https://archlinuxarm.org**) based Linux distribution with additonal packages and configuration in order to transform your board to a **k.i.s.s.** (keep it simple, stupid!) computer transport.

It currently supports all Armv7/Armv8 Raspberry Pi and Odroid C2 devices.

Archphile uses **MPD** (**M**usic **P**layer **D**aemon):

https://www.musicpd.org

MPD is a media player that acts as a network server and needs a client to use it with.

Archphile uses **YMPD** (https://ympd.org) MPD client by default. However you can use it with any other available MPD client.

Below you will find most of the information needed in order to sucessfully configure your board with Archphile.

Please note that you will need to **connect via ssh** in order to apply all the required changes.

Finding the IP address for the first time

In order to connect to your Archphile board, you need to find its IP address. This can be done by using an android/ios app like **fing** (https://www.fing.io), or via the web interface of your router.

If your system supports **bonjour/avahi**, then you can use **http://archphile.local** in order to access the web interface or to connect via SSH (with putty – please see below).

After connecting for the first time, it’s strongly suggested to set up a static network IP.

Connect Via SSH

Connecting via ssh is pretty easy. If you use Windows you will need **putty:**

https://www.putty.org

Linux and Mac os X users can simply use a **terminal application** and connect (*assumming that Archphile uses the ip 192.168.1.142*) using the following command:

```
ssh root@192.168.1.142
```

File editing with nano

Another very useful tool is **nano** editor. Archphile configuration is based on plain text file editing. For this purpose you will need an editor. For example, let's say you want to edit **/etc/mpd.conf** file. All you need to do is:

```
nano /etc/mpd.conf
```

When you finish editing the file, press **CTRL + X**, then press **Y** and **ENTER** and that's it. Your edited file is now saved and you 're ready to go.

Systemd services

Most of the programs found in archhpile are handled via systemd services. Some useful commands are:

```
systemctl enable|reenable|disable|start|stop|restart <blabla>
```

where **<blabla>** can be mpd, ympd, upmpdcli, shairport-sync, roonbridge, etc..

1.0 System Configuration

1.1 Root Password

The default password for **root** is **archphile.** It is strongly recommended that you change this password the first time you boot Archphile. This can be done using the following command:

```
passwd
```

1.2 Timezone and NTP server configuration

Timezone/NTP default settings are set up for the country I live in. If you want to change them you will need to do the following:

```
rm /etc/localtime
```

and then link to the appropriate time zone.

```
ln -s /usr/share/zoneinfo/Europe/Athens /etc/localtime
```

The above command is just an example. If you want to explore all available locations, please use the following command:

```
ls -la /usr/share/zoneinfo
```

In order to change NTP servers, you will need to edit the following file:

```
rm /etc/ntp.conf
```

Note: Modifying the time zone and NTP servers is completely optional.

2.0 Network configuration

Archphile default network configuration uses DHCP. It's strongly recommended to **change to static IP**. In order to do this you must edit the following file:

```
nano /etc/netctl/archphile-network
```

The active configuration by default (for DHCP) is the following:

```
Description='A basic dhcp ethernet connection'
Interface=eth0
Connection=ethernet
IP=dhcp
ExecUpPost='/usr/bin/ntpd -gq || true'
```

Lets assume that the ip of your router is 192.168.1.1 and you want to use the static IP 192.168.1.142 for the Archphile board. The first step is to comment the DHCP section. Now, uncomment the static IP network and change the IPs to match your setup:

```
#Description='A basic dhcp ethernet connection'
#Interface=eth0
#Connection=ethernet
#IP=dhcp
#ExecUpPost='/usr/bin/ntpd -gq || true'

Description='A basic ethernet connection with static ip'
Interface=eth0
Connection=ethernet
IP=static
Address=('192.168.1.142/24')
Gateway='192.168.1.1'
ExecUpPost='/usr/bin/ntpd -gq || true'
```

The last step is to reenable archphile-network:

```
netctl reenable archphile-network
```

Now reboot with:

```
systemctl reboot
```

Note: The use of a static IP means that the network configuration is dependent on the router. If you plan to use your Archphile board with another router than the one you've set it up for, make sure that the new one supports the same IP address range. A good idea is to revert to DHCP configuration, boot with the new router and change to static IP again at a later stage.

3.0 NAS Configuration

3.1 Samba Shares

Archphile supports samba/cifs/nfs by default. However there is no automated tool to configure the NAS. The only needed action is the editing of **/etc/fstab**:

```
nano /etc/fstab
```

Let’s assume that you use a NAS with a static IP 192.168.1.150. The directory served is **music** and it is configured as public. All you need to do for this case is to edit fstab and put all this information in the “Public Share” example:

```
//192.168.1.150/music /mnt/nas-samba cifs
vers=1.0,guest,ro,uid=mpd,gid=audio,iocharset=utf8,nolock,noauto,x-
systemd.automount,x-systemd.device-timeout=10,sec=ntlm
```

If this share was not public, you would need to use the next example found in fstab and put your username and password:

```
//192.168.1.150/music /mnt/nas-samba cifs
vers=1.0,username=yourusername,password=yourpassword,ro,uid=mpd,gid=audio,iocharset
=utf8,nolock,noauto,x-systemd.automount,x-systemd.device-timeout=10,sec=ntlm
```

If you compare this line with the one found in fstab, you will notice that **rsize** and **wsize** are missing. I suggest you to do the same, unless you know what you are doing and you want to bybass the default settings.

A reboot is needed for the changes to take effect:

```
systemctl reboot
```

3.2 NFS Shares

For a NAS using NFS and a static IP of 192.168.1.150, the fstab section will be:

```
192.168.1.150:/music   /mnt/nas-nfs  nfs4    ro,noauto,x-systemd.automount,x-
systemd.device-timeout=10,rsize=8192,wsize=8192
```

A reboot is needed for the changes to take effect:

```
systemctl reboot
```

3.3 USB Disk Sharing

If you use a usb disk with your board, you can use Archphile in order to serve it on your local network. All you need to do is enable Samba server:

```
systemctl start smbd nmbd
systemctl enable smbd nmbd
```

If you want to stop and disable this feature at a later stage, you can use the following commands:

```
systemctl stop smbd nmbd
systemctl disable smbd nmbd
```

4.0 MPD

4.1 Packages

Archphile offers various MPD (Music Player Daemon) packages:

- mpd-archphile

this is the default Archphile package. It has most of the features needed for the average user. The major difference with other packages is that it has **ffmpeg** support, so that you can listen to alac or other “exotic” files.

- mpd-archphile-minimal

this is a stripped down version of the default package. Amongst others, it does not provide ffmpeg support. This is the package of my preference.

- mpd-archphile-sacd

This package offers an MPD **fork** with SACD ISO support.

All Archphile image come with **mpd-archphile**. If you want to install any of the rest you must do the following:

```
pacman -Sy mpd-archphile-minimal
systemctl reenable mpd
systemctl restart mpd && archphile-optimize
```

or

```
pacman -Sy mpd-archphile-sacd
systemctl reenable mpd
systemctl restart mpd && archphile-optimize
```

in order to re-install the officiall package:

```
pacman -Sy mpd-archphile
systemctl reenable mpd
systemctl restart mpd && archphile-optimize
```

4.2 Additional file extensions support (alac etc..)

The default MPD package (mpd-archphile) is built with **ffmpeg** support. This means that MPD can support additional file types like for example **alac**. All you need to do is enable ffmpeg in **/etc/mpd.conf**.

```
nano /etc/mpd.conf
```

You will find the following section:

```
decoder {
plugin “ffmpeg”
enabled “no”
}
```

Change the no to yes:

```
decoder {
plugin “ffmpeg”
enabled “yes”
}
```

and restart MPD:

```
systemctl restart mpd
```

4.3 Software/Hardware mixer

Archphile comes with all mixers disabled by default in order to ensure Bit Perfect playback. However if you know what you are doing and you want to experiment, you can enable them in **/etc/mpd.conf**.

First you need to edit **/etc/mpd.conf**:

```
nano /etc/mpd.conf
```

and change the following line:

```
mixer_type  “disabled
```

to either

```
mixer_type  “hardware”
```

or

```
mixer_type  “software”
```

and restart MPD:

```
systemctl restart mpd && archphile-optimize
```

Note: Not all DACs support a hardware mixer (you can check with alsamixer for that..). In additon changing the hardware or software volume from 0dB to anything else, means that the result will stop being Bit-Perfect (please read the **Appendix**).

4.4 Resampling

MPD is built with SoX (http://sox.sourceforge.net/SoX/Resampling) resampler support and if you want to upsample/downsample, you will need to edit **/etc/mpd.conf**:

```
nano /etc/mpd.conf
```

Then, find this section:

```
#resampler {
#  plugin "soxr"
#  quality "very high"
#}
```

and enable it:

```
resampler {
 plugin "soxr"
 quality "very high"
}
```

Next, go to **audio_output** section and find these lines:

```
#format "192000:24:2"
```

```
#format "*:24:*"
```

Let’s assume that you want to upsample everything to **24/192**. You will just need to enable the first line.

```
format "192000:24:2"
```

If you want to upsample everything to **32/96**, you will need to modify and enable the first line:

```
format "96000:32:2"
```

If you want to upsample everything to **32 bit**, you will need to modify and enable the second line:

```
#format "*:32:*"
```

Save the new **/etc/mpd.conf** file and restart MPD with:

```
systemctl restart mpd && archphile-optimize
```

4.5 MPD and DSD

MPD in Archphile is set up with DoP disabled by default. This means that the reproduction of DSD files will be done via Native DSD (if the DAC supports it in Linux) or via a DSD to PCM conversion.

If you know that your DAC supports DoP, you can edit **/etc/mpd.conf**:

```
nano /etc/mpd.conf
```

find the line:

```
#dop   "yes"
```

and enable it:

```
dop   "yes"
```

Now restart MPD:

```
systemctl restart mpd && archphile-optimize
```

Most of the users get confused with all these Native DSD, DoP etc. terms. Please make sure that you read Appendix **Chapter B. DoP vs Native DSD vs DSD to PCM conversion and MPD**.

4.6 Library Auto-Update

MPD database has an option for auto-update. This means that MPD can scan for changes in library and add new files automatically. This is disabled by default.If you want to enable it you need to edit /etc/mpd.conf:

```
nano /etc/mpd.conf
```

and change:

```
auto_update “no”
```

to:

```
auto_update “yes”
```

and then restart MPD:

```
systemctl restart mpd && archphile-optimize
```

4.7 Backup/Restore of Music Library Database

If you want to transfer your MPD library through different Archphile installations, or simply keep a backup of it, you can use some pretty simple scripts included: **dbackup and drestore.**

In order to backup the library,type:

```
dbackup
```

If you intend to use the same Archphile installation and you just want to restore the MPD library database, type:

```
drestore
```

If you want to transfer your library database to a new installation, after using **dbackup,** use an application that supports **SFTP file transfer** like **filezilla** or **winscp**:

https://filezilla-project.org
https://winscp.net

and copy **mpd.db** file from **/opt/dbackup** directory.

Now copy the new Archphile image on an sd card, boot to your new installation and place the previously backuped **mpd.db** file in **/opt/dbackup**.

Now it’s time to restore:

```
drestore
```

and you ‘re ready to use your old library with MPD.

Note: the systemd restart command of MPD has been enriched in order to include an additional command. Everytime MPD restarts, **archphile-optimze** script needs to run in order for tasksets and CPU affinity to be re-applied. So, **whenever** you need to restart MPD, the command will be:

```
systemctl restart mpd && archphile-optimize
```

5.0 UPNP/DLNA Support

Archphile supports **upnp/dlna** protocol using **upmpdcli:**

https://www.lesbonscomptes.com/upmpdcli

With upmpdcli, Archphile acts as a upnp/dlna **renderer** that can be used with various applications, like for example **bubbleupnp**.

Upmpdcli is not enabled by default. In order to start it you will need the following command:

```
systemctl start upmpdcli
```

If you want to enable it so that it runs after every boot:

```
systemctl enable upmpdcli
```

If you later change your mind and you want to disable it, you can use the following command:

```
systemctl disable upmpdcli
```

6.0 Airplay Support

Archpile supports **Airplay** with the use of **shairport-sync:**

https://github.com/mikebrady/shairport-sync

shairport-sync is not enabled by default. In order to start it you will need the following command:

```
systemctl start shairport-sync
```

If you want to enable it so that it runs after every boot:

```
systemctl enable shairport-sync
```

If you later change your mind and you want to disable it, you can use the following command:

```
systemctl disable shairport-sync
```

7.0 Spotify Support

Archphile supports **Spotify**, using a **librespot** fork:

https://github.com/librespot-org

librespot works with the official Spotify applications, where Archphile will **appear as a supported device**. In order to configure librespot you will need to edit the following file:

```
nano /etc/librespot.conf
```

and put your **username** and **password**.

librespot is not enabled by default. In order to start it you will need the following command:

```
systemctl start librespot
```

If you want to enable it so that it runs after every boot:

```
systemctl enable librespot
```

If you later change your mind and you want to disable it, you can use the following command:

```
systemctl disable librespot
```

8.0 Roon Support

Archphile can be used as a **Roon bridge.** The needed software **is not open source** and this is the reason that **it's not installed by default**. If you want to use Archphile as a Roon Bridge you will need to install it:

For the Raspberry Pi you will need the following command:

```
pacman -Sy roonbridge-archpile-arm7
```

For the Odroid C2 you will need:

```
pacman -Sy roonbridge-archpile-arm8
```

Afther the installation, **roonbridge** will not be enabled by default. In order to start it you will need the following command:

```
systemctl start roonbridge
```

If you want to enable it so that it runs after every boot:

```
systemctl enable roonbridge
```

If you later change your mind and you want to disable it, you can use the following command:

```
systemctl disable roonbridge
```

9.0 Android Remote Control

In order to control various Archphile services/applications/states, it's highly suggested to create a custom SSH remote control **(after setting up a static IP!)**. This can be done with an app like **Raspberry SSH Custom Buttons** or **Raspi SSH**. There you can create buttons for the majority of the systemd services described in previous sections. Below you will find examples of such buttons you can create:

– **Start Spotify**: add a button with this name and in the field of the command put **systemctl start librespot**

– **Stop Spotify**: add a button with this name and in the field of the command put **systemctl stop librespot**

– **Start Upnp**: add a button with this name and in the field of the command put **systemctl start upmpdcli**

– **Stop Upnp**: add a button with this name and in the field of the command put **systemctl stop upmpdcli**

– **Start Shairport-sync**: add a button with this name and in the field of the command put **systemctl start shairport-sync**

– **Stop Shairport-sync**: add a button with this name and in the field of the command put **systemctl stop shairport-sync**

You can do the same for **roonbridge**.

In addition you can create some **MPD related** commands:

- **Restart MPD**: add a button with this name and in the field of the command put **systemctl restart mpd** (or even better, **systemctl restart mpd && archphile-optimize**)

– **Update MPD library**: add a button with this name and in the field of the command put **mpc update**

– **Stop Music**: add a button with this name and in the field of the command put **mpc stop**
– **Play Music**: add a button with this name and in the field of the command put **mpc play**

Last but not least, you can create buttons for **shutdown** and **reboot**:

– **Shutdown**: add a button with this name and in the field of the command put **systemctl poweroff**

– **Reboot**: add a button with this name and in the field of the command put **systemctl reboot**

The advantage of this SSH remote is that you can control most of the functions of Archphile, without the need of terminal access. In addition **you won't need to enable any service you don't need to run every time Archphile boots**. For example, you will only use (with the stasrt button) **Spotify** when you want. When you don't use it, librespot will not be enabled and it will not get any system resource.

10.0 Archphile and I2s DACs for the Raspberry Pi

The use of I2s DACs in Archphile is pretty simple and straight-forward. The only step needed is the editing of **/boot/config.txt**:

```
nano /boot/config.txt
```

There you will find a list of (disabled) ovelays for various I2S DACs. Let's assume you want to enable your **Mamboberry Dac**. You need to find this section:

```
# Hifiberry and Mamboberry Dacs
#dtoverlay=hifiberry-dac
```

enable the overlay:

```
# Hifiberry and Mamboberry Dacs
dtoverlay=hifiberry-dac
```

and reboot:

```
systemctl reboot
```

11.0 Archphile Optimization

11.1 Odroid C2 optimizations

Archphile for Odroid C2 comes with various optimizations enabled by default. Below you will find some useful information about the most important ones.

11.1.1 CPU Cores Isolation

The Odroid C2 Archphile image comes with **cores 1,2 and 3** isolated by default.

Core 2 is used only by MPD, **cores 3 and 4** are used for **USB** and **ethernet interrupts**. The remaining processes are handled by **core 0**.

The core isolation is achieved via the editing of **/boot/boot.txt** file (plus the **mkscr** command):

```
nano /boot/boot.txt
```

There you will find the appropriate configuration already applied with the isolcpus option:

```
setenv bootargs "console=ttyAML0,115200n8 root=PARTUUID=${uuid} isolcpus=1,2,3 rw
rootwait earlycon"
```

The redirection of MPD so that it uses the previously isolated **core 1** is handled via a command in **archphile-optimize** script:

```
nano usr/bin/archphile-optimize
```

There you will find the following enabled command:

```
taskset -c -a -p 1 $(pidof mpd)
```

Please note that **you don’t have to do anything from your side**. All the information above is just given for you to be fully aware on what’s already configured on your system.

11.1.2 CPU Affinity

By default **USB inrerrupts** are redirected to **core 4** and **ethernet interrupts** to **core 3**.The script including the appropriate commands is **/usr/bin/irq-archphile**.

The above script is then called by **archphile-optimize** script which runs at every boot. If you edit it:

```
nano /usr/bin/archphile-optimize
```

you will find the already enabled section:

```
# IRQ affinity optimization - Do not apply it if you have a Raspberry Pi!!!
#
#/usr/bin/irq-archphile
```

Again, **you don’t have to do anything** as **everything is already applied**.

11.1.3 Odroid C2 optimizations mpd-archphile-sacd

As already mentioned, you have the option to install **mpd-archphile-sacd**, an **MPD fork** that supports **SACD ISOs**. As **DTS decoding** is very **CPU hungry**, if you decide to use this MPD fork, you will need to **disable** CPU isolation and MPD redirection .

At firstyou will need to edit **/boot/boot.txt:**

```
nano /boot/boot.txt
```

Then find the line that includes isolcpus command:

```
setenv bootargs "console=ttyAML0,115200n8 root=PARTUUID=${uuid} isolcpus=1,2,3 rw
rootwait earlycon"
```

and get rid of it:

```
setenv bootargs "console=ttyAML0,115200n8 root=PARTUUID=${uuid} rw rootwait
earlycon"
```

In addition, give the following two commands:

```
cd /boot
```

```
./mkscr
```

Then edit **archphile-optimize** script:

```
nano /usr/bin/archphile-optimize
```

find this command:

```
taskset -c -a -p 1 $(pidof mpd)
```

and disable it:

```
#taskset -c -a -p 1 $(pidof mpd)
```

Finally, reboot:

```
systemctl reboot
```

11.2 Raspberry Pi optimizations

11.2.1 CPU Cores Isolation

The Raspberry Pi Archphile image comes with **cores 0 and 1** are isolated by default.

Core 0 handles all interupts (that unfortunately cannot be redirected on RPI) and **core 1** is only used by MPD. Everything else is handled by **cores 2 and 3**.

The core isolation is achieved via the editing of **/boot/comdline.txt** file:

```
nano /boot/cmdline.txt
```

There you will find the appropriate configuration already applied with the **isolcpus** option:

```
root=/dev/mmcblk0p2 rw rootwait console=ttyAMA0,115200 console=tty1 selinux=0
isolcpus=0,1 plymouth.enable=0 smsc95xx.turbo_mode=N dwc_otg.lpm_enable=0
kgdboc=ttyAMA0,115200 elevator=noop
```

The redirection of MPD so that it uses the **core 1** is handled via a command in **archphile-optimize** script:

```
nano /usr/bin/archphile-optimize
```

There you will find the following enabled command:

```
taskset -c -a -p 1 $(pidof mpd)
```

Please note that **you don't have to do anything from your side**. All the information above is given for you to be fully aware on what's configured on your system.

11.2.2 Wireless, Bluetooth and HDMI

Wireless and bluetooth are disabled by default. If you want to enable them, you will need to edit **/boot/config.txt**:

```
nano /boot/config.txt
```

11.2.2.1 Wireless

If you want to enable wireless, you will need to find the following line:

```
dtoverlay=pi3-disable-wifi
```

and disable it:

```
#dtoverlay=pi3-disable-wifi
```

Please note that in order to use a wireless, you will neet to set a country code.

This can be done in **archphile-optimize** script:

```
nano /usr/bin/archphile-optimize
```

find the line below:

```
#iw reg set GR
```

change to your country's code (you will have to google for wireless country codes) and enable. For example, for **Norway** the line will be:

```
iw reg set NO
```

Save and reboot:

```
systemctl reboot
```

11.2.2.2 Bluetooth

If you want to enable bluetooth you will need to find the following line:

```
dtoverlay=pi3-disable-bt
```

and disable it:

```
#dtoverlay=pi3-disable-bt
```

Finally, reboot:

```
systemctl reboot
```

11.2.2.3 HDMI

HDMI is enabled by default. If you want to disable it, you can edit **archphile-optimize** script:

```
nano /usr/bin/archphile-optimize
```

find the section below

```
# Disable HDMI for RPI
#/opt/vc/bin/tvservice -o
```

and enable it:

```
# Disable HDMI for RPI
/opt/vc/bin/tvservice -o
```

Finally, reboot:

```
systemctl reboot
```

11.2.3 Raspberry Pi optimizations mpd-archphile-sacd

As already mentioned, you have the option to install **mpd-archphile-sacd**, an **MPD fork** that supports **SACD ISOs**. As **DTS decoding** is very **CPU hungry**, if you decide to use this MPD fork, you will need to **disable** CPU isolation and MPD redirection.

Firstly you will need to edit **/boot/cmdline.txt:**

```
nano /boot/boot.txt
```

disable the line that includes the **isoclcpus** command:

```
#setenv bootargs "console=ttyAML0,115200n8 root=PARTUUID=${uuid} isolcpus=1,2,3 rw
rootwait earlycon"
```

and enable the line under “Default Configuration” title:

```
root=/dev/mmcblk0p2 rw rootwait console=ttyAMA0,115200 console=tty1 selinux=0
plymouth.enable=0 smsc95xx.turbo_mode=N dwc_otg.lpm_enable=0 kgdboc=ttyAMA0,115200
elevator=noop
```

Then, edit **archphile-optimize** script:

```
nano /usr/bin/archphile-optimize
```

find the following command:

```
taskset -c -a -p 1 $(pidof mpd)
```

and disable it:

```
#taskset -c -a -p 1 $(pidof mpd)
```

Finally, reboot:

```
systemctl reboot
```

11.3 Generic Optimizations

If you never use **YMPD** and you prefer another MPD client, you can disable it with:

```
systemctl disable ympd
```

If you later change your mind and you want to enable it:

```
systemctl enable ympd
```

If you are sure that you don’t need **avahi** for bonjour network discovery, you can disable it with:

```
systemctl disable avahi-daemon
```

If you later change your mind and you want to enable it:

```
systemctl enable avahi-daemon
```

If you don’t have any USB disk/stick attached on your Archphile board and you don’t plan to use one, you can disable the usb auto-mounter (udevil):

```
systemctl disable devmon@root
```

If you later change your mind and you want to enable it:

```
systemctl enable devmon@root
```

12.0 Real Life Examples

In this section I will analyze some of the most frequent user configurations in order to show you **how easy it is** to get started with Archphile.

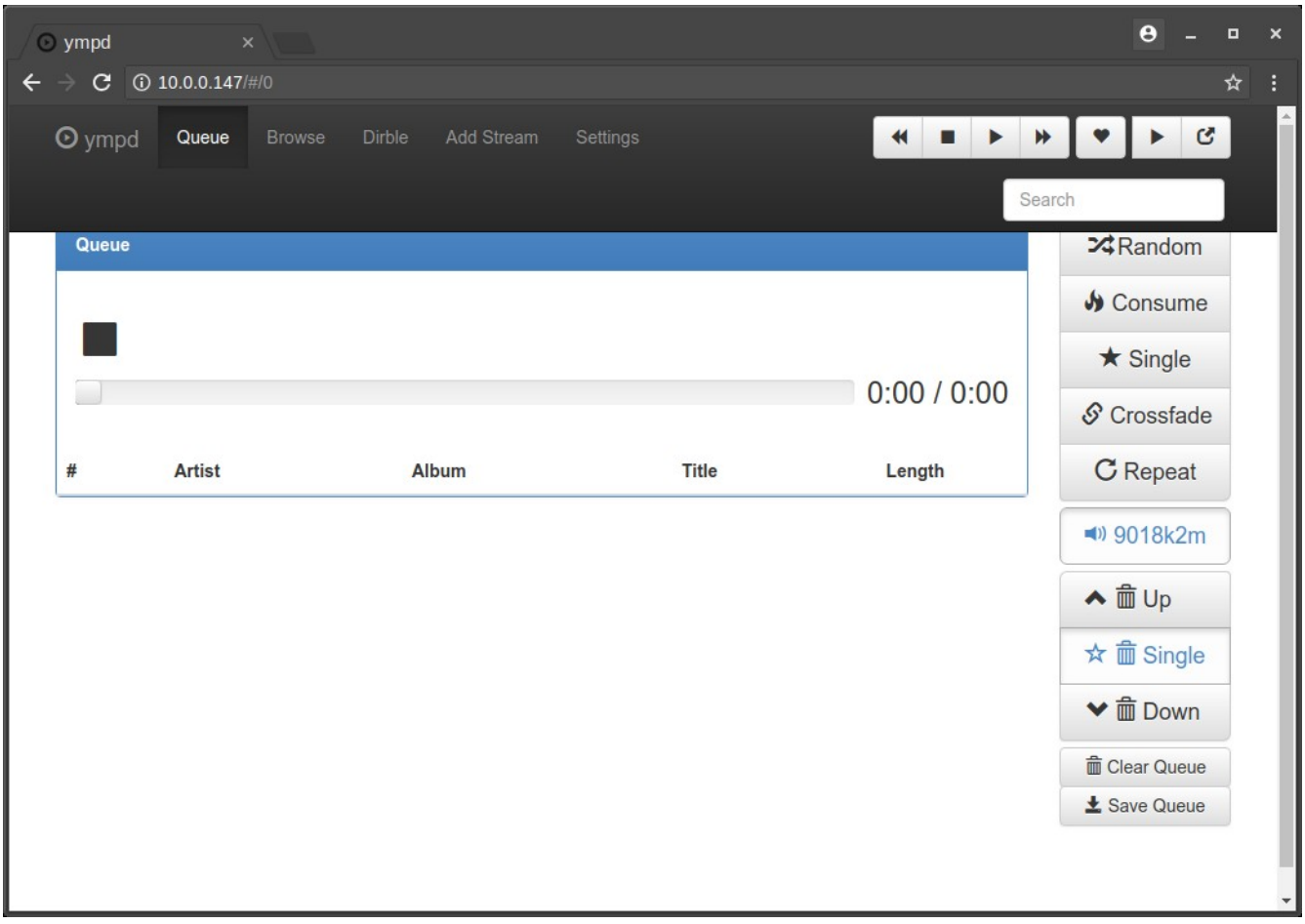
12.1 Simple use with a USB disk or stick

Let’s assume that you have a USB disk or stick with your music files and that this device is plugged on your Archphile board.

The first thing that you have to do is find the IP address of the Archphile device. As I already mentioned in the introduction of this manual, in order to do this, the simplest way is to use **fing** application for android or ios.

Now that you found the IP, you are ready to use your board!

Just open a web browser and type the IP you just found. What you will immediately see is the default Archphile MPD client, named **YMPD**:



If your os supports **bonjour/avahi**, instead of using the IP, you can access the board just using the following address:

archphile.local

The next thing that you have to do is **start updating the MPD database**, so that MPD is able to scan and later use your files. In order to do this, just go to YMPD **“Settings”** and press the button **“Update Database”.** After a while, your files will start appearing in the **“Browse”** section of YMPD.

Please note that depending on the size of the music library it might take from seconds to hours for the library scanning to complete.

When this procedure is over you can start browsing your library through YMPD and add the files you want to listen to YMPD **queue**.

Although you can keep using Archphile like it is now, seting a static IP is almost necessary. This way you will ensure that you will always know how to access your transport and in addition you will be able to use an SSH android remote.

So the remaining steps to complete Archphile setup are:

- Follow the instructions of **Chapter 2.0 (Network Configuration)** in order to set a static IP
- Follow the instrucitons of **Chapter 9.0 (Android Remote Control)** in order set up an SSH remote

- As a last **optional** step you can install an MPD client (MPD Remote, M.A.L.P, MPDroid etc..) on your smartphone to use in addition to YMPD.

12.2 Simple use with a NAS

This scenario is almost identical to the one above. The only extra step you have to do is to follow the instructions from **Chapter 3.0 (Nas Configuration)** and configure your NAS.

After you reboot, you will need to go to YMPD **“Settings”** and press the button **“Update Database”.** After a while, your files will start appearing in the **“Browse”** section of YMPD.

13.0 Frequently Asked Questions

- Is really Archphile a new Linux distribution?

No, Archphile is not a new distribution. It's just ArchlinuxARM (https://archlinuxarm.org) with lots pre-installed packages (from both ArchlinuxARM and Archphile), plus many additional tweaks and configuration.

Please note, that although being an ArchlinuxARM distribution, Archphile is not a rolling release distro. Updating the system with pacman will 99,99% break most of the additional Archphile packages already installed.

- I want to buy Archphile. How much does it cost?

Archphile is a completely free/open source software. You don't have to pay anything in order to download and use it. However if you are satisfied with this project, you can help it be kept alive by donating, using the paypal or BTC information on the right side of Archphile website (http://archphile.org).

- I want to see your code/packages/scripts right now!

All right! All you have to do is visit my github page:

https://github.com/archphile

- What are the benefits of using Archphile?

Archphile is mainly targeted to Raspberry Pi and Odroid C2 owners who want to use their devices as music transports and don't have the knowledge or the time to spend in order to configure a Linux distribution for this purpose.

- I am using Volumio/Runeaudio/Moodeaudio. Why should I change and use Archphile?

I don't care what you are currently using. As long as you are satisfied with your current distribution, stick on it and listen to your music.

- Come on dude! Give me a reason to prefer Archphile over the other distributions!

Ok. The philosophy behind archphile is to **keep it simple**. You will configure the system once and forget it.

Archphile doesn't use any resource hungry web servers, interpreters like php or databases like mysql and it's configured so that it uses the less possible resources.

This, doesn't mean that Archphile sounds better, but that it's usually more stable comparing to a bloated alternative.

- Yes, but configuring Archphile is so difficult! YMPD does not have any system configuration settings. It even doesn't have a shutdown button!

Yes configuring Archphile via "terminal file editing" might seem difficult for newbies and this is the reason I wrote this manual.

Regarding YMPD, No, it does not have any OS confuguration setting or shutdown button, because it's just an MPD client. Its job is to control MPD and nothing more.

- Do you really use Archphile as a transport for your hifi or do you just play around with it?

The reason for keep developing Archphile is because I am a user of this software. I use an Odroid C2 along with a DIY 9018k2m DAC and my flacs are served via a diy USB NAS (DD-WRT Router with a USB disk on it) using SAMBA.

- You describe Archphile as an "audiophile" distribution. What do you mean with this term? Is it that good?

The term audiophile is usually used in order to describe people enthusiastic about high fidelity sound reproduction.
I don't state that using Archphile with a Raspberry Pi or an Odroid C2 is a high quality hi-fi or hi-end solution. To be honest with you I really don't care about it! Archphile is a distribution for people that want to get the best out of these tiny boards with regards to audio reproduction and just start listening to music.

- I see that you focus very much on DSD capabilities, offering an extra custom MPD fork package. Why are you so much interested in this format? It it so good?

To be honest, I don't like and I don't listen to DSD files. In my opinion, everything that has to do with DSD is just a hype.

I believe that a good redbook file (16/44.1) should be enough for a superb sound result. The reason I play around with DSD, creating custom packages etc. is because I am a geek and I like playing around with stuff like that.

- I want the X or Y feature to be included. Please can you do this?

No!

- I have a question and I need to contact you. How can I contact you?

Please send an email to info@archphile.org, but please, before you send me an email, Read The Fucking Manual!

Appendix

A. Bit Perfect Playback

Bit Perfect playback is a phrase used to describe a setup that plays back music as-is, keeping the bit depth and sampling rate untouched, avoiding any format convsersion or any software volume modification.

In order to verify what is actually being sent to the DAC, you should use the following command:

```
cat /proc/asound/card*/pcm*p/sub*/hw_params
```

Below you will find some output examples of this command:

- playing a “Redbook” 16/44.1 flac:

```
access: RW_INTERLEAVED
format: S16_LE
subformat: STD
channels: 2
rate: 44100 (44100/1)
period_size: 5513
buffer_size: 22050
```

- playing a 24/96 flac:

```
access: RW_INTERLEAVED
format: S24_LE
subformat: STD
channels: 2
rate: 96000 (96000/1)
period_size: 12000
buffer_size: 48000
```

- playing a DSD64 dsf using DoP Mode:

```
access: RW_INTERLEAVED
format: S24_LE
subformat: STD
channels: 2
rate: 176400 (176400/1)
period_size: 22050
buffer_size: 882000
```

- playing a DSD64 dsf using Native DSD Mode:

```
access: RW_INTERLEAVED
format: DSD_U32_BE
subformat: STD
channels: 2
rate: 88200 (88200/1)
period_size: 11025
buffer_size: 44100
```

Note: There are cases where no resampling takes place, but you keep seing 24 or 32 bit in the format section even if the file is 16 bit.
This is usually not a problem. Many DACs only support 24 or 32 bit, so MPD sends a the actual bit depth plus some “zeros” to the DAC in order to reach the supported bit depth of the DAC. This procedure is still Bit-Perfect.

The easiest way to identify the capabilities of a DAC is to use the following command (when MPD doesn’t play any file):

```
alsacap
```

For example, the result for my current DAC is the following:

```
*** Scanning for playback devices ***
Card 0, ID `D20', name `DIYINHK USB Audio 2.0'
  Device 0, ID `USB Audio', name `USB Audio', 1 subdevices (1 available)
    2 channels, sampling rate 44100..384000 Hz
    Sample formats: S16_LE, S32_LE, SPECIAL, DSD_U32_BE
      Subdevice 0, name `subdevice #0'
```

Here we can see that the dac supports from **44.1Khz to 384Khz**. With regards to bit depths, it supports **16 bit and 32 bit**. So when MPD plays a 24 bit flac, 32 bit are sent to my DAC.

Last but not least, **DSD_U32_BE** means that this DAC supports **Native DSD** in Linux.

B. DoP vs Native DSD vs DSD to PCM conversion and MPD

DSD files can be played by MPD using three different software/hardware methods:

- DoP

Using DoP, the DSD stream is packed into a PCM data stream and transmitted to the DAC. The latter reassembles the oroginal DSD data stream completely unchanged.

In order to use the DoP protocol:

1. the DAC chip of your DAC device must support DSD
2. the USB receiver of your DAC must support DoP
3. DoP has to be enabled in **/etc/mpd.conf** (see **Chapter 4.5 MPD and DSD**)

Using the command mentioned in Appendix **Chapter A**:

```
cat /proc/asound/card*/pcm*p/sub*/hw_params
```

the result for DSD64 is:

```
access: RW_INTERLEAVED
format: S24_LE
subformat: STD
channels: 2
rate: 176400 (176400/1)
period_size: 22050
buffer_size: 882000
```

We can see that the DSD stream is packed in a PCM **24/176.4** data stream.

Note: As we already mentioned, not all DACs support 16/24/32 bit, so it’s normal if you see a stream of 32/176.4 (this is what I see with both my DACs). These 32 bits is 24 bit plus “zeros” in order to reach the supported bit depth.

- Native DSD

Using Native DSD, the DSD stream is transmitted to the USB receiver unmodified.

In order to use Native DSD:

1. the DAC chip of your DAC must support DSD
2. the USB receiver of your DAC must support Native DSD
3. the DAC/USB receiver must be supported in Linux Kernel
4. DoP has to be disabled in **/etc/mpd.conf** (see **Chapter 4.5 MPD and DSD**)

Using the same command:

```
cat /proc/asound/card*/pcm*p/sub*/hw_params
```

the result for DSD64 is:

```
access: RW_INTERLEAVED
format: DSD_U32_BE
subformat: STD
channels: 2
rate: 88200 (88200/1)
period_size: 11025
buffer_size: 44100
```

It’s clear from the format **DSD_U32_BE** that this stream is using Native DSD.

Many DAC owners expect their DAC to use Native DSD under Linux because they read it on their DAC’s hadrware specs. Unfortunately a specific entry for each DAC/USB receiver is needed in a Linux Kernel file and this makes things very complicated.

- DSD to PCM conversion

Using DSD to PCM conversion before the stream leaves the transport is the failsafe option.

In order to use DSD to PCM conversion:

- The DAC does not support DSD

or

- The DAC does not support Native DSD
- DoP option must be disabled in **/etc/mpd.conf**

Note: DoP can be disabled in two different ways:

1. the DoP line is inactive or absent:

```
#dop   "yes"
```

2. the DoP line is active and contains a **no** instead of a **yes**:

```
dop   "no"
```

The resut of 1 and 2 is the same.

If you are interested in this topic, please make sure you read my blog post below:

http://thepenguin.eu/2017-12-22-mpd-and-dsd-files

Information about this manual

This manual can be downloaded from Archphile website. The url will always be:

http://archphile.org/manual/archphile-manual.pdf

An .odt file will always be available in the following github url:

https://github.com/archphile/manual/blob/master/archphile-manual.odt

Please make sure to always have the latest version (mentioned in the first page of this PDF).

I would like to give special thanks to Michael Psyrras for helping me with corrections and additional ideas during the writing of this manual.

If you found any mistake and you would like to let me know about it, please send an email to info@archphile.org.