

**pulsecast**

Using PulseAudio with Single Board Computers to Power a Whole Home Audio Solution

# Agenda

1. The Problem
2. Linux Sound Architecture and PulseAudio
3. Recipe for pulsecast System using Raspberry Pi with HifiBerry

# The Problem

I have several stereos in a house that already has Ethernet in every room.

I would like to be able to direct the same audio to all speakers in the house, without loss of audio quality and without noticeable delays between speakers.

It would also be nice to keep using my current music player, Clementine, as well as being able to use any streaming services that I might subscribe to (only a handful of which are integrated into Clementine).

Speaker notes

**My problem has been solved**

Over the years, by a lot of products in a lot of different ways. If only I could find the answers I was sure it had been solved several times before in Linux/OSS.

**Prior Solution**

1. Create and save play lists from a music player.
2. Feed the play lists to icecast from ezstream.
3. Connect a tablet or computer to each stereo and play the stream via browser.



## Issues with Prior Solution

- Need for extra steps of saving play list and then invoking ezstream.
- All items on play-list needed to be in same format.
- Any changes required restarting the stream.
- Each time the stream started/restarted had to visit each stereo.

# Google ChromeCast Audio

Speaker notes

A friend persuaded me to look at ChromeCast Audio.

# Pro

- Inexpensive (about \$30)
- Based on DLNA
- Grouping feature to simultaneously cast to ChromeCast Audio devices

## Speaker notes

The ChromeCast Audio is a different product than the regular ChromeCast having a combined TOS optical / headphone jack port and the regular ChromeCasts don't support the Audio extensions.

# Con

- CCA has been discontinued
- Best OS Support was Android
- Some Features Required Google Home App
- On other platforms only supported player was the Chrome Browser
- Sound quality similar to built in audio on other devices
- Frequent dropouts even over Ethernet (unacceptable)

# Unofficial Support

- pulseaudio-dlna (outside project, not part of pulseaudio)
- PulseAudio AVAHI support
- VLC added ChromeCast support in version 3.0

## Speaker notes

pulseaudio-dlna wasn't reliable and had an annoying artifact that when my player switched tracks that there was a very audible artifact.

Pulseaudio has built in avahi support and I would look at configuring that directly in pulse if I had further interest in the ChromeCast, but the audio dropouts were unacceptable.

I had already moved on before the vlc feature was available.



## It did support an external DAC

The ChromeCasts had an optical audio out which uses a non-standard variant of a Toslink cable that you can feed a Digital-Analog Converter with a Tos input.

Speaker notes

But the standard current DAC interface is usb.

I never tested it with a DAC, but it probably didn't fix the dropouts.

The CCA was a great attempt at an inexpensive consumer solution. But at less than the price of a decent soundcard they may have been too ambitious on cost.



# Multi-Room Wired

There are a lot of possible configurations here, and many systems out there in this space.







# Advantages

- Reliable, especially the simpler octopus configurations
- Been around a long time

# Disadvantages



- Requires running lots of new dedicated wire (expensive)
- Future Reconfiguration would require pulling more wire again
- Off the shelf systems are costly and at risk of obsolescence

## Speaker notes

The system pictured costs \$1500 plus installation, requires controllers on the walls of your zones (which also work as intercoms), and claims integration with the likes of Alexa. If the company goes out of business or stops supporting this generation of product that phone app may go away, ditto for replacement parts. Which means in 5-10 years you're probably upgrading to a new one.

# Sonos

It is even more proprietary than the ChromeCast and extremely expensive as in it can cost thousands of dollars to implement.



# Whole House FM

Doesn't require wires or any network infrastructure.

This solution is cheap and would work, but has limitations.



## Speaker notes

You know that annoying problem where some recordings are made to sound louder than others, with transmitting over FM this problem is magnified because you have to set the system for the loudest source material, but then the dynamic range limitations of FM mean that your quieter material is going to be degraded, unless you have a live DJ or automatic compression system monitoring and fixing your levels constantly. Consumer FM is also limited to an extremely low power output, which also results in a lower signal quality.

# DLNA / AVAHI

DLNA and Avahi are a set of standards for multi-media devices to communicate, worthy of a talk in their own right.

## Speaker notes

The ChromeCast was based on these protocols. Most DLNA implementations only support one source to one destination at a time, the grouping extension implemented by the ChromeCast Audio is a major feature of the devices.

There are several projects to make a Raspberry Pi or something similar act as a DLNA device for sending media to for rendering. They all look to be fairly small and I never got as far as testing them.

Volumio and RuneAudio are open source Media Center distributions that will run on low end devices like Pi.

The grouping feature of the CCA appears to be unique and innovative, the other options did not appear to have anything like it.

#### Speaker notes

I found that while the projects support DLNA devices as destinations, they have no support for grouping targets, which limits them to stream to one set of speakers at a time.

Rather than have to dig into the DLNA/AVAHI protocol I decided to move on.

# VLC and ffmpeg

Both VLC and ffmpeg have the ability to stream content via RTP. Unfortunately, their built in streaming is limited to single files without support for play-lists.



# Internet Radio DJ Software

There is software for running an Internet radio station, but the open source options I glanced over did not appear to be simple to install and configure.

## Speaker notes

Once you get your stream going your player devices would just have a loop to check for the stream and start playing it with a command line audio player.

**Why can't we just capture to a virtual sound-card  
and stream from there?**

In search of that answer I started to look at the Linux Sound Architecture. OSS and ALSA are too basic to do that on their own.

PulseAudio supports both RTP streaming between computers running it and DLNA / AVAHI.

#### Speaker notes

In fact the same author wrote the standard Linux AVAHI support as wrote Pulse.

This gets us to ...

**Part II**

**Linux Sound Architecture and  
PulseAudio**

# An Introduction to Linux Sound Architecture

- Open Sound System (OSS)
- Enlightened Sound Daemon (ESD)
- Advanced Linux Sound Architecture (ALSA)
- Jack
- PulseAudio

## Speaker notes

ESD is a dead project at this point, and Linux Kernel support for OSS has been deprecated although the project is still releasing updates and used in BSD.

# ALSA

The Advanced Linux Sound Architecture (ALSA) provides audio and MIDI functionality to the Linux operating system. ALSA has the following significant features:

- Efficient support for all types of audio interfaces, from consumer sound cards to professional multichannel audio interfaces.
- Fully modularized sound drivers.
- SMP and thread-safe design.
- User space library (alsa-lib) to simplify application programming and provide higher level functionality.
- Support for the older Open Sound System (OSS) API, providing binary compatibility for most OSS programs.

#### Speaker notes

ALSA primarily acts as the driver layer, as does legacy OSS if present.

PulseAudio and Jack are Sound Servers both of which work closely together with ALSA



# What is a Sound Server?

The Sound Server accepts sound input from source and redirects it to a sink (sound card etc). This architecture was developed to manage multiple inputs and outputs on a system.

## Speaker notes

Without an intermediate layer multiple applications could be directly sending to the sound card at the same time without any mixing or contention policy.

# Jack

Jack is targeted to Audio Recording and Mixing Applications implementing a virtual audio Patch Panel. Jack and Pulse are able to coexist and run at the same time.

# PulseAudio

Pulse provides API and hardware abstraction, including networked hardware. Most distribution's Desktop environments use Pulse and it generally provides a working out of the box setup.

To work its' magic Pulse interfaces with existing ALSA and OSS hardware drivers, and provides emulation of Enlightened Sound Daemon.

- User Space Daemon
- Control and Configuration from Command Line, Configuration Files, and Graphical Utilities.
- Created by Lennart Poettering.

The diagram on the next slide comes courtesy of rudd-o.com, the original is at <https://rudd-o.com/uploads/images/how-pulseaudio-works/pulseaudio-diagram.png/view>

#### Speaker notes

Pulse is run as a user space Daemon, typically when you log into a graphical environment, it is started for you. You can also start it manually for a non-graphical environment.

Pulse can be controlled from a GUI or its command line utilities. When configured from a GUI it writes to configuration files in a different location and format than when configured by editing config files.

If you didn't already know, the creator of Pulse was Lennart Poettering.

## Speaker notes

As the diagram demonstrates being a sound server is complicated.

Reminder, use view image to display entire image.

# PulseAudio Terminology

## **Server:**

the PulseAudio Daemon. Since it runs in user space each user gets their own instance of the daemon.

## **Client:**

the application sending data to Pulse via one of its several supported APIs or emulations.

### **Sources:**

generate sound, this could be clients or hardware such as a microphone.

### **Sink:**

the device generating the sound, generally this is handled by ALSA.



## **Front End:**

Applications that interact directly with Pulse, such as configuration utilities and sound mixer applets.

## **Modules:**

Pulse is modular, splitting functionality into optional modules and allowing for extension of functionality by developers outside the Pulse project.

### Speaker notes

Running in user space each user gets their own pulseaudio instance. Desktop computers are generally single user so this isn't usually a problem. System Mode (runs as a system process) did not work for me.

# Communicating with PulseAudio

On your desktop you'll mostly use pavucontrol, paprefs and your desktop environments audio management tools to interact with pulse. However, for diagnostics and for working on the headless pulsecast you'll need pacmd and pactl.

```
pacmd info
## pactl utility provides a briefer summary
pactl info
pacmd list-(sinks|modules|cards|sources|clients|sink-inputs)
pactl list (sinks|modules|cards|sources|clients|sink-inputs)
pacmd dump
```

## Speaker notes

pacmd is the current favored command line client. pactl is still supported.

```
pi@berry2:~ $ pactl list sink-inputs
Sink Input #0
  Driver: module-rtp-recv.c
  Owner Module: 8
  Client: n/a
  Sink: 0
  Sample Specification: s16be 2ch 44062Hz
  Channel Map: front-left,front-right
  Format: pcm, format.sample_format = "\"s16be\"" format.rate = "44100" format.channels = "2"
  Corked: no
  Mute: no
  Volume: front-left: 65536 / 100% / 0.00 dB, front-right: 65536 / 100% / 0.00 dB
         balance 0.00
  Buffer Latency: 258372 usec
  Sink Latency: 239144 usec
  Resample method: speex-fixed-1
  Properties:
    media.role = "stream"
    media.name = "RTP Stream (PulseAudio RTP Stream on mypc)"
    rtp.session = "PulseAudio RTP Stream on mypc"
    rtp.origin = "user 936190379 0 IN IP4 192.168.1.11"
    rtp.payload = "10"
    module-stream-restore.id = "sink-input-by-media-role:stream"
```

# Part III

## pulsecast

# A Recipe for Whole Home Audio with PulseAudio and Raspberry Pi

### Speaker notes

This is a recipe, I could have chosen a Beagle Bone or in Intel/AMD based SBC, or lots of other hardware.

As an aside my experience has found regular a Intel/AMD Linux PC to have less sync drift over time compared to my RPi based devices, it is likely that Intel/AMD single board computers would share this benefit, but they will drive your cost up. The drift issue was easy enough to fix by adding a cron job to restart Pulseaudio at a time I was unlikely to be listening.

# Linux to Linux Pulse

Make sure all of the needed packages are installed.

On Debian/Ubuntu/Mint these are

- pulseaudio
- pulseaudio-utils
- pulseaudio-module-gconf
- pavucontrol

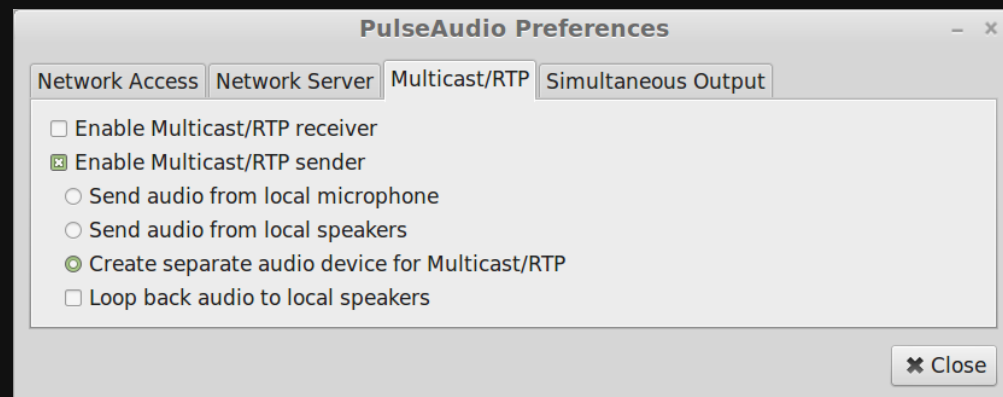
## Speaker notes

If you have two boxes already running Linux this is a great way to start.

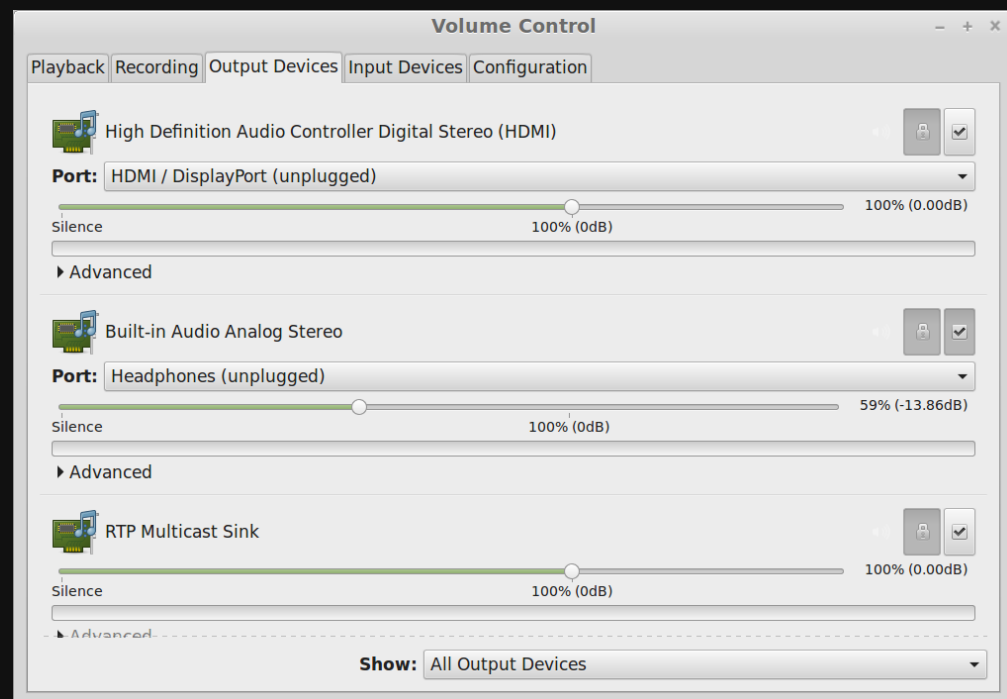
The config files generated by the GUI are in a different location and format than the ones we'll be using for the headless configuration.

## Enable Sending on the Source Computer

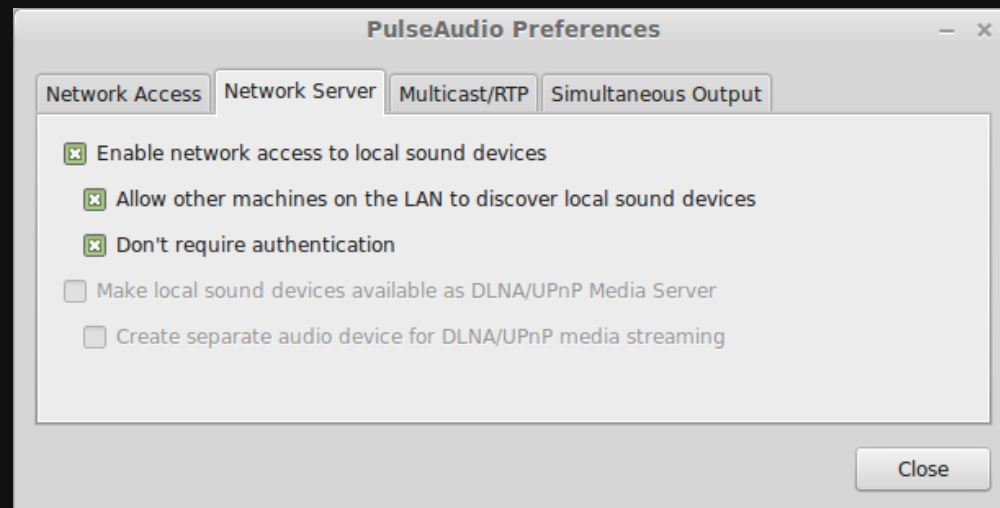
Two Clicks is all it takes in PulseAudio Preferences (paprefs).



# The New Output Device in PulseAudio Volume Control

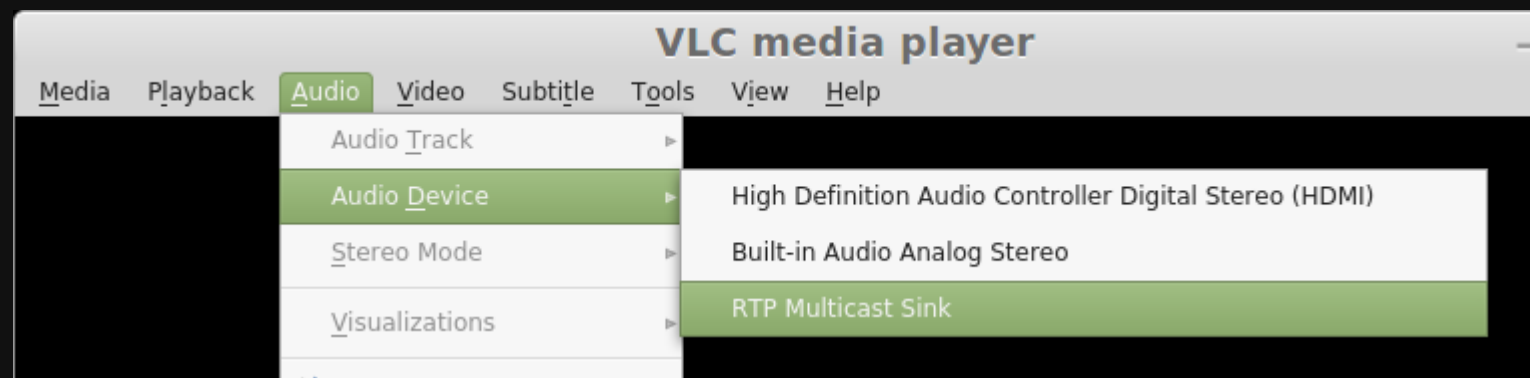


## A few more clicks on the Receiving Side in PulseAudio Preferences (paprefs)





# Switching Output in VLC



## Speaker notes

That's it for getting the music you play with VLC to play on the other computer's speakers.

This is really nice in that you can put only your music player on the Virtual SoundCard we created and leave impolite browsers to blast out the tiny speakers on your PC.

Note that while you can stream the audio portion of a video from VLC there is a slight buffer delay which will make it not work so great.

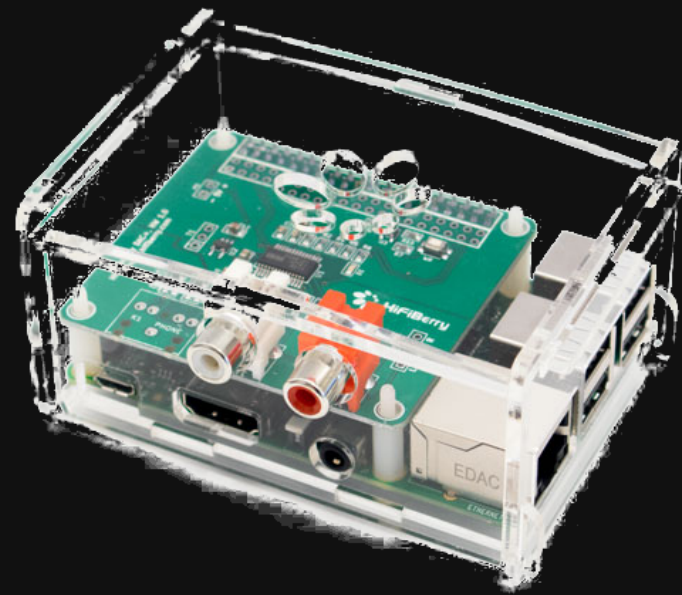
If you want to see the configuration values for a setup you made with gui tools

```
$ ls ~/.gconf/system/pulseaudio/modules  
%gconf.xml  remote-access  rtp-recv  rtp-send  zeroconf-discover
```

each folder contains a **%gconf.xml** file.

Entries in **%gconf.xml** are the same configuration directives we'll see later, but wrapped in an xml tag.

```
<stringvalue>module-native-protocol-tcp</stringvalue>
```



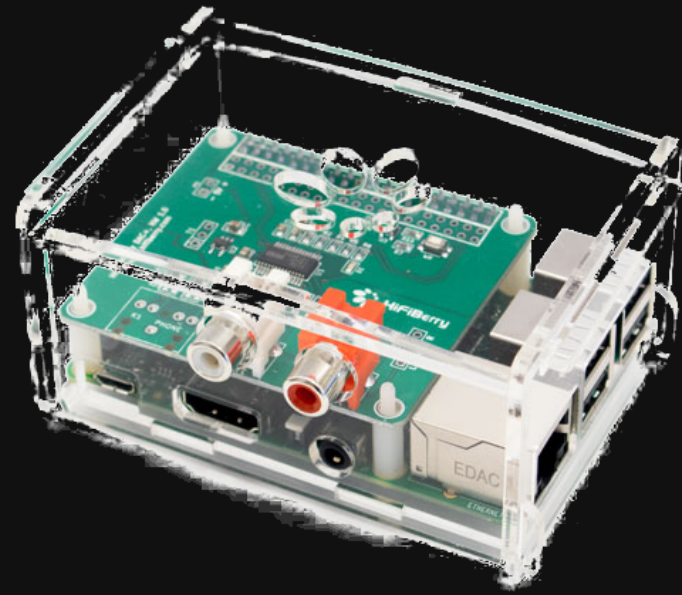
# pulsecast Device Using Raspberry Pi

## What You'll Need

- Raspberry Pi

### Speaker notes

When I originally put this together the latest model was Pi 3 B, currently (June 2019) Pi 3 Model B+ is available.



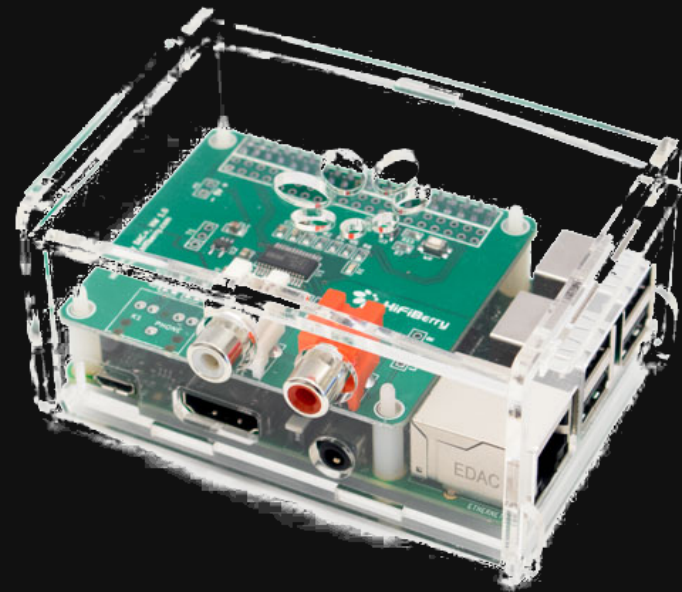
# pulsecast Device Using Raspberry Pi

## What You'll Need

- Raspberry Pi
- Micro SD Card

### Speaker notes

My total installation size is about 1.2 GB, you could use an SD card as small as 4GB.



# pulsecast Device Using Raspberry Pi

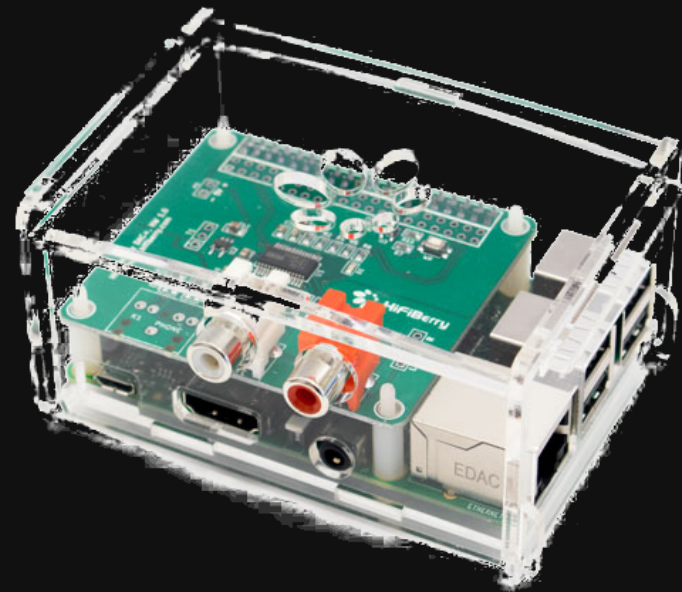
## What You'll Need

- Raspberry Pi

### Speaker notes

This is a popular sound card for the Pi and works in many other Single Board Computers. I recommend/used the DAC+ or the DAC+ Pro for this project. Both have RCA jacks for output (which is what you want for a stereo), gold plated on the Pro. Both use the same Burr Brown chip-set (a lot of audiophiles prefer their chips to other brands). The Pro also has its own clock generation. I would have used all Pros but they were out of stock after I built my first unit.

The other HiFiBerry products use a less expensive DAC chip, including the Amp product which will let you directly power small speakers without needing an Amplifier (Stereo Receiver).



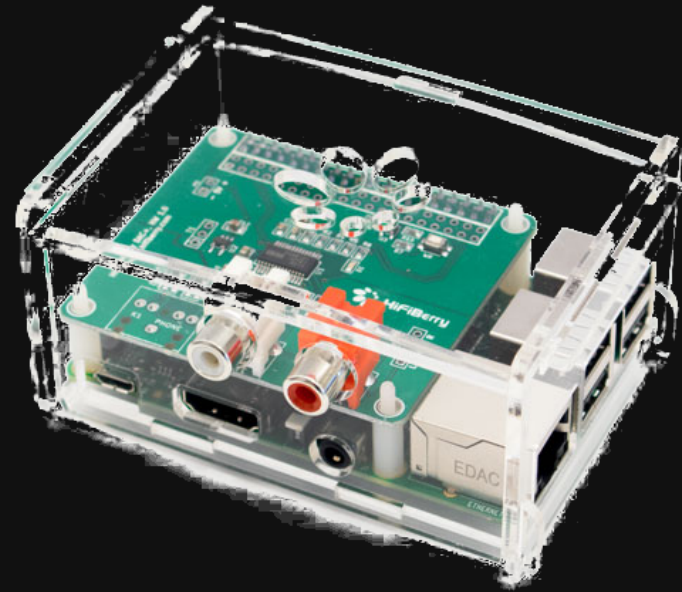
# pulsecast Device Using Raspberry Pi

## What You'll Need

- Raspberry Pi
- Micro SD Card
- HiFi Berry SoundCard

### Speaker notes

You will need a special case designed to accommodate the board you're using and it's RCA jacks.



# pulsecast Device Using Raspberry Pi

## What You'll Need

- Raspberry Pi
- Micro SD Card
- HiFi Berry SoundCard

[Go for HiFi Berry](#)

### Speaker notes

Raspbian is the most popular distribution for Single Board Computers and I use Debian based distributions regularly. NOOBS makes makes it easy for first timers to setup, but does require hooking up a keyboard, mouse and monitor.



# Quick Pi Setup



1. Format MicroSD as FAT32. (I use gparted)
2. download & extract NOOBS
3. copy the extracted NOOBS to the newly formatted MicroSD
4. Attach peripherals and network, boot Pi with NOOBS MicroSD installed.
5. Install minimal Raspbian (we don't want the gui).
6. Switch keyboard to US during install or your keyboard will be mapped wrong.
7. Reboot when the installation is complete.

Speaker notes

This is a quick setup overview for novices.

## 8. login

- user: pi
- password: raspberry

9. sudo ifconfig to get MAC and IP address.

10. `sudo raspi-config` to configure your pi.

- Change password.
- Set hostname.
- Choose Interfacing Options, from sub menu SSH, and enable SSH access.

11. Either reserve the ip address in DHCP or set a static ip address.

12. Now is a good time to update the system.

- `sudo apt-get update`

#### Speaker notes

You will need the device to be at a fixed ip address for future connection to it.

Add an entry in your DNS if you have a DNS server.

# Enable the Soundcard

Edit /boot/config.txt (must be root).

```
## dtparam=audio=on ## disables built in sound

## load hifiberry
dtoverlay=hifiberry-dacplus

## disable wifi and bluetooth if you're not using them.
dtoverlay=pi3-disable-wifi
dtoverlay=pi3-disable-bt
```

- `sudo halt -p`
- Unplug your Pi.
- Remove the USB devices you used for setting up NOOBS.
- Power the Pi on again.

#### Speaker notes

All communications with my devices will be over Ethernet, so for security I disable the unneeded io interfaces.

To avoid resource contention on the shared USB/Ethernet bus disconnect all USB devices.

# Install PulseAudio

SSH back into the device to begin setting up pulseaudio.

```
sudo dpkg -i pulseaudio pulseaudio-utils screen
```

Add the following scripts

```
/usr/local/sbin/startpulse.sh

#!/bin/bash
while [ 1 ]; do
    ## start does not return error if daemon already loaded
    /usr/bin/pulseaudio --start --disallow-module-loading
    sleep 60
done

/usr/local/sbin/reloadpulse.sh

#!/bin/bash
## kill all pulse processes
/usr/bin/pkill pulseaudio
/usr/bin/pulseaudio --start --disallow-module-loading
```

# Configure Pulse

```
ls /etc/pulse
client.conf daemon.conf default.pa system.pa
```

These are read in the order daemon, default, client.

```
daemon.conf
```

```
default-fragments = 8
default-fragment-size-msec = 10
deferred-volume-safety-margin-usec = 8000
```

```
default.pa
```

```
load-module module-native-protocol-tcp auth-ip-acl=127.0.0.1;192.168.1.111
load-module module-rtp-recv
```

## Speaker notes

System.pa is only read if pulse is started in system mode, which is not recommended. And in my experience didn't work.

auth-ip-acl is optional, if you don't set it any host on your network can connect.

I also made changes to daemon.conf to match the defaults for debian desktop

# Fix a Minor Issue

I found that over time the delay on the various pulsecasts would drift, where this did not appear to be a problem with a desktop pc running linux.

Just add to the pi user's crontab the following line:

```
00 6 * * * /usr/bin/pkill pulseaudio
```







## Speaker notes

pkill won't kill startpulse.sh, but will kill all pulse daemons, startpulse is looping infinitely and starting pulse every 60 seconds.



- Files
- Playlists
- Internet
- Devices
- Song i...
- Artist ...

Feel So Good  
 Crazy Miranda  
 Pretty As You Feel  
 Wild Turkey  
 Law Man  
 Rock And Roll Island  
 Third Week In The Chelsea  
 Never Argue With A German If You'...  
 Thunk  
 War Movie  
 Bless Its Pointed Little Head (1996)  
 Clergy  
 3/5's Of A Mile In 10 Seconds  
 Somebody To Love  
 Fat Angel  
 Rock Me Baby  
 The Other Side Of This Life  
 It's No Secret  
 Plastic Fantastic Lover  
 Turn Out The Lights  
 Bear Melt  
 Crown of Creation  
 Live At The Fillmore East  
 Live At The Monterey Festival

Spirit of the Wood	Galaxy C...	Weapon of...	2:16	Spirit of...	🎵
Stairs Prayer	Galaxy C...	Weapon of...	5:31	Stairs Pray...	🎵
We 3 Kings	Galaxy C...	Weapon of...	2:48	We3Kings....	🎵
▶ 1 She Has Funny Cars	Jefferson A...	Surrealisti...	3:13	01-She Ha...	🎵
2 Somebody to Love	Jefferson A...	Surrealisti...	3:01	02-Someb...	🎵
3 My Best Friend	Jefferson A...	Surrealisti...	3:04	03-My Bes...	🎵
4 Today	Jefferson A...	Surrealisti...	3:02	04-Today....	🎵
5 Comin' Back to Me	Jefferson A...	Surrealisti...	5:24	05-Comin'...	🎵
6 3/5 of a Mile in 10 Seco...	Jefferson A...	Surrealisti...	3:45	06-3_5 of ...	🎵
7 D.C.B.A.-25	Jefferson A...	Surrealisti...	2:39	07-D.C.B....	🎵
8 How Do You Feel	Jefferson A...	Surrealisti...	3:34	08-How D...	🎵
9 Embryonic Journey	Jefferson A...	Surrealisti...	1:55	09-Embry...	🎵
10 White Rabbit	Jefferson A...	Surrealisti...	2:33	10-White ...	🎵
11 Plastic Fantastic Lover	Jefferson A...	Surrealisti...	2:38	11-Plastic ...	🎵



# Clementine

I've been using this open source cross platform player for years. It is very good at managing local collections, and has integration for several streaming services and comes pre-configured with a lot of Internet radio stations.

In 2016 they even released an Android Remote Control Application.

## Speaker notes

The project is in C++ and while there are active commits it doesn't appear to have enough maintainers, so for example it is severely lagging in adding support for streaming services.

# More Options and the Future

Snapcast is a recent project which could be used in place of PulseAudio which has a time synch mechanism, but does not yet consider itself stable and is not packaged.

I've been looking at some MPD (Music Player Daemon) alternatives, Nuvola and Mopidy, because they support a variety of client front ends and support more streaming services.

## Speaker notes

I've been collecting music for 40 years, and have a huge collection. Last Month Quboz finally started accepting US subscribers. There are two things that are special about Quboz: they were a pioneer of offering higher quality streaming and they have a deeper classical catalog than any of the services that originated in the US.

## PulseAudio Documentation

<https://gavv.github.io/blog/pulseaudio-under-the-hood/>

<https://www.freedesktop.org/wiki/Software/PulseAudio/>

<https://wiki.archlinux.org/index.php/PulseAudio>

## Clementine

<https://www.clementine-player.org/>

This Slide Deck ©2018, 2019 John Karr

```
Reveal.initialize({ // ... width: "100%", height: "100%", margin: 0, minScale: 1, maxScale: 1 });
```