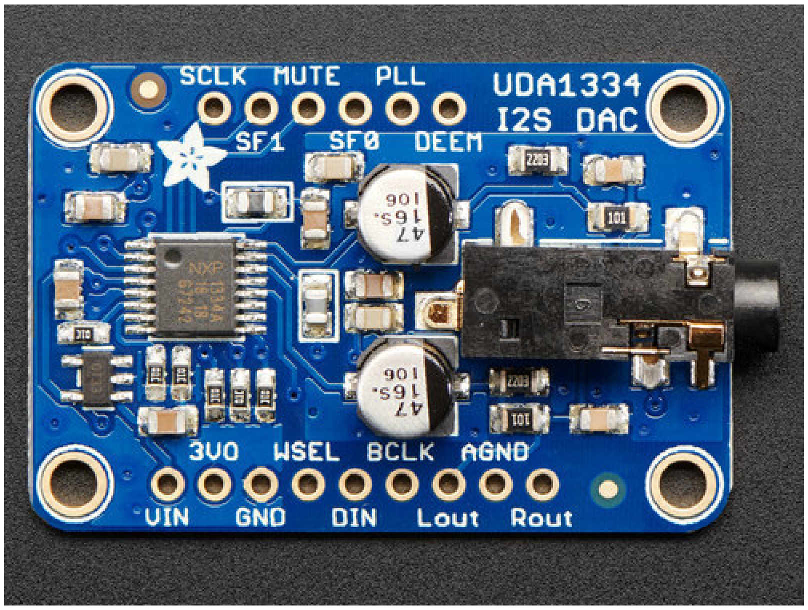
adafruit learning system

Adafruit I2S Stereo Decoder - UDA1334A

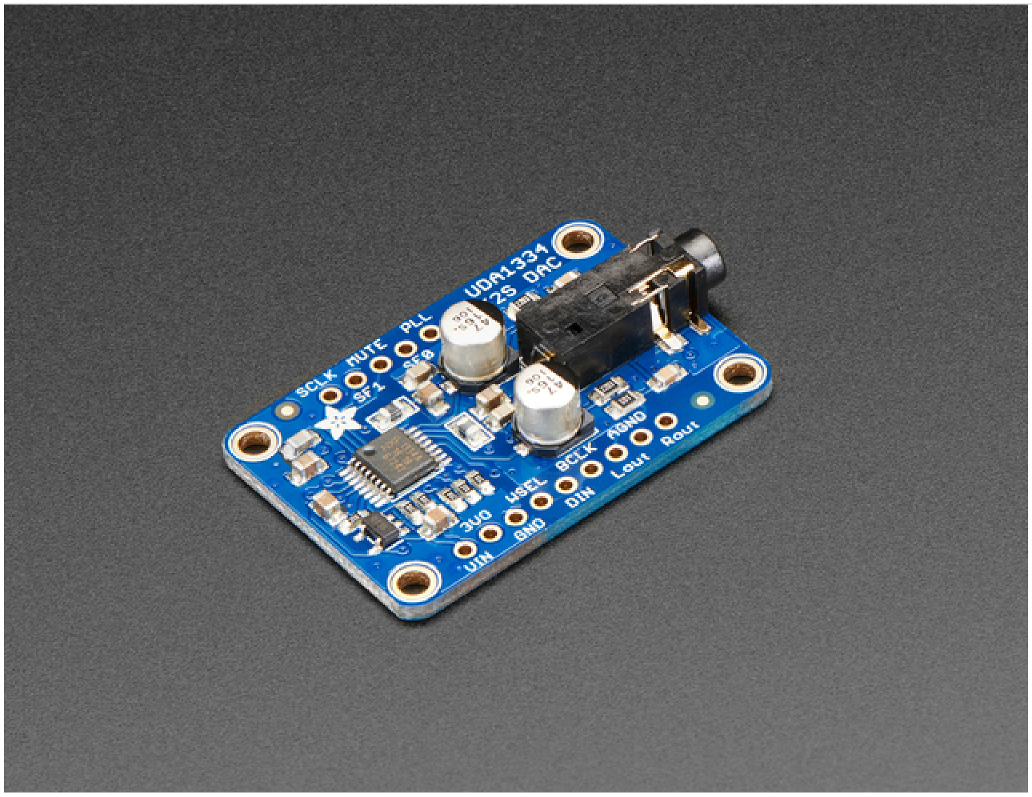
Created by lady ada



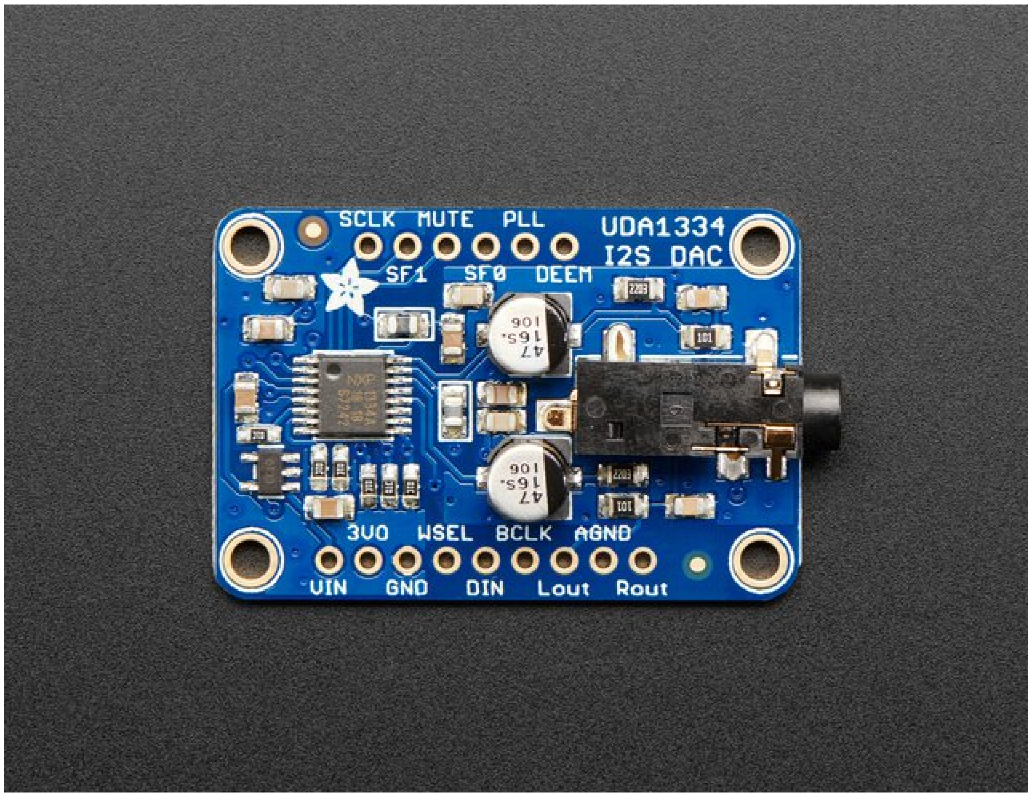
Last updated on 2017-11-27 10:57:29 PM UTC

Guide Contents

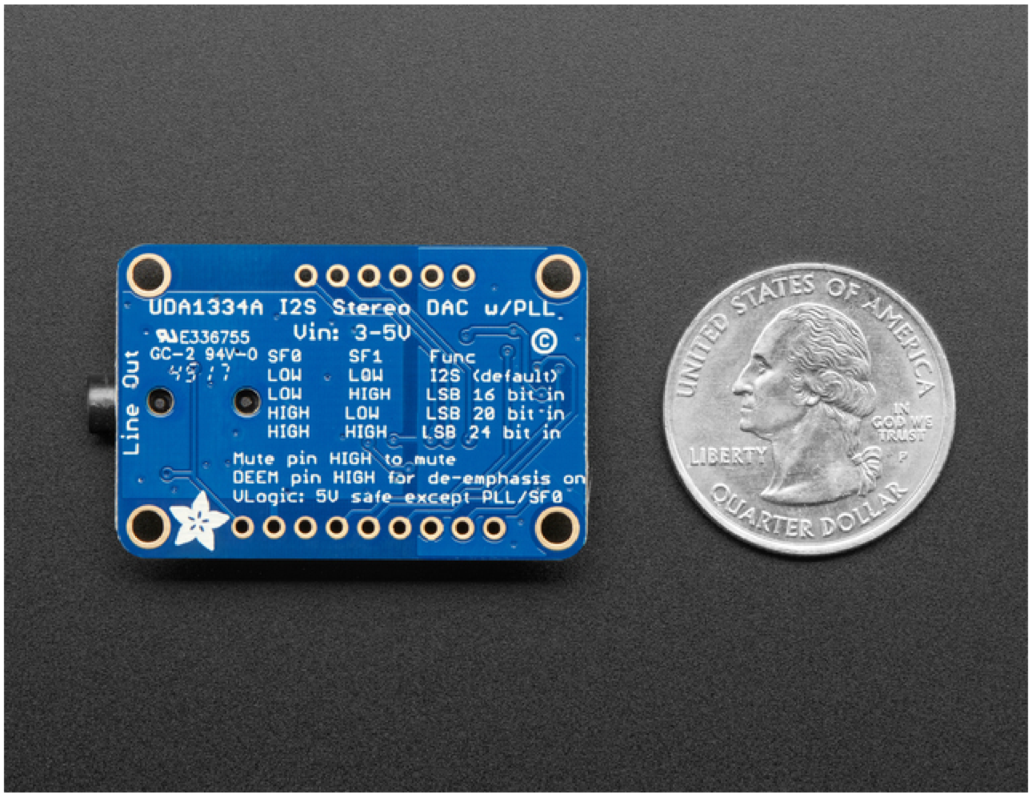
Overview



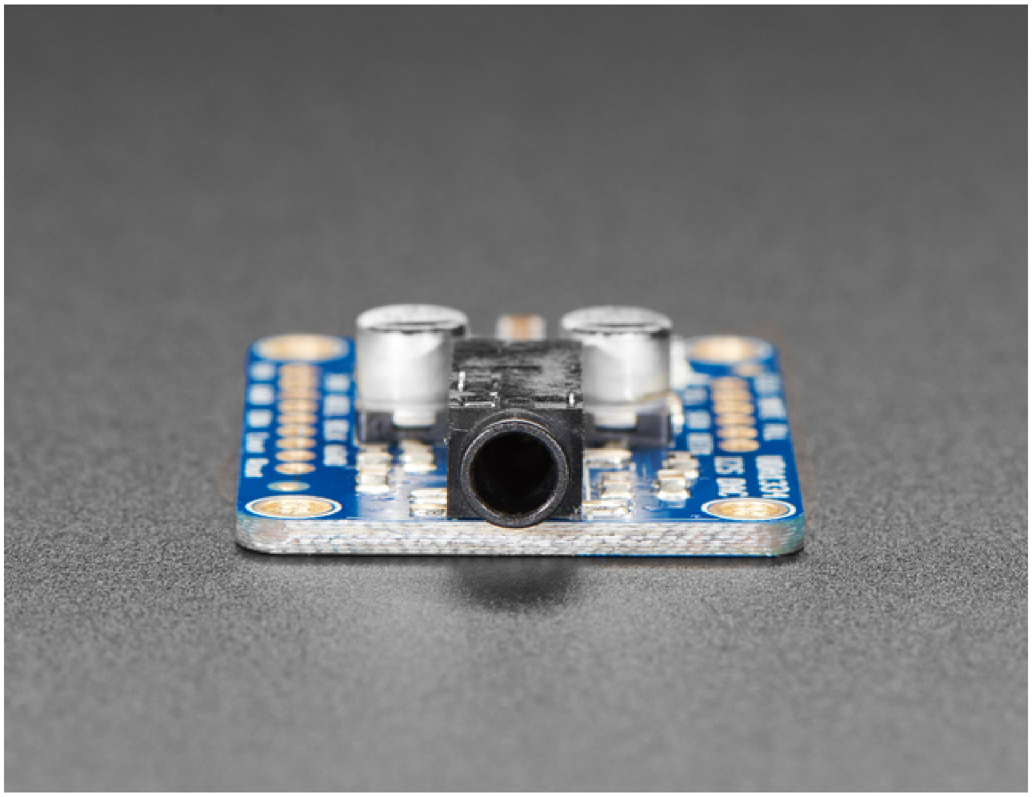
This fully-featured UDA1334A I2S Stereo DAC breakout is a perfect match for any I2S-output audio interface. It's affordable but sounds great! The NXP UDA1334A is a jack-of-all-I2S-trades: you can use 3.3V - 5V logic levels (a rarity), and can process multiple different formats by setting two pins to high or low. The DAC will process data immediately, and give you a clear, analog, stereo line level output. It's even cool with MCLK-less I2S interfaces such as the Raspberry Pi (which it's ideal for) - a built in PLL will generate the proper clock from the incoming signal.



For inputs, you can use classic I2S (the default) or 16-bit, 20-bit or 24-bit left justified data. You can set it up to take an input system/master clock but we default-set it to just generate it for you, so you only need to connect Data In, Word Select (Left/Right Clock) and Bit Clock lines. If you want, there's a mute pin and a de-emphasis filter you can turn on.

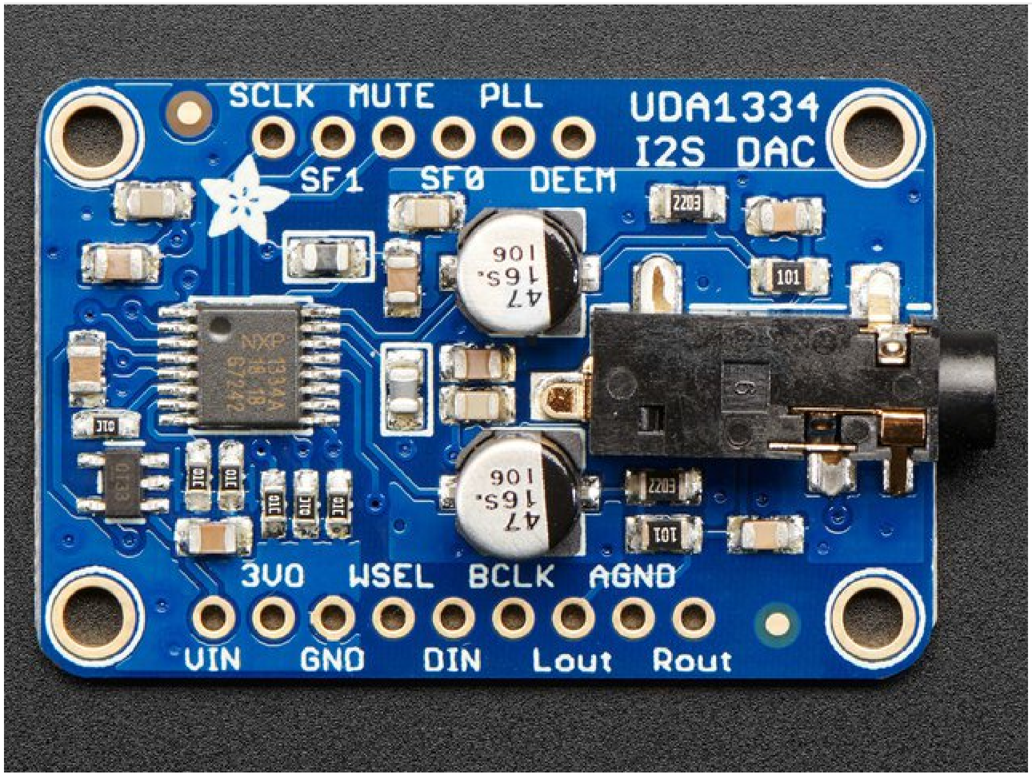


We put in plenty of ferrite beads, a low-dropout regulator, and the recommended band-pass filter so you get a very nice clean output. With a sine-wave generator we swept through 20-20KHz and saw no attenuation or distortion. Plug into either the 3.5mm stereo headphone jack or the breadboard-friendly pads. We think you'll be pleased with this DAC!



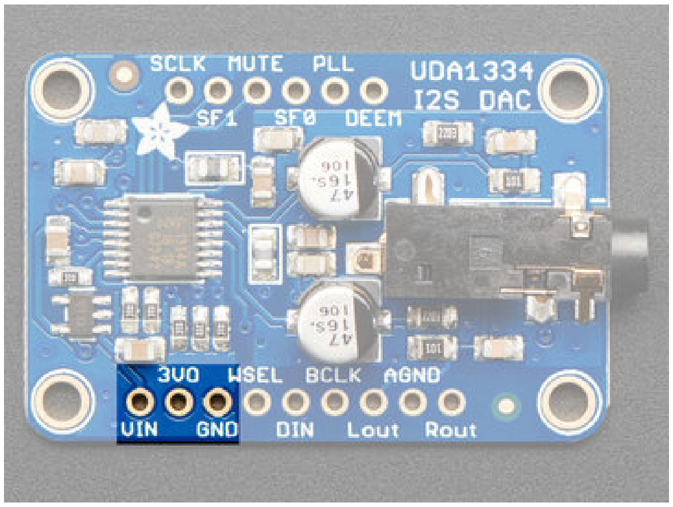
Each order comes with one I2S Stereo DAC breakout and some header you can solder on.

Pinouts



The UDA1334A is an **I2S** amplifier - it does not use analog inputs, it only has digital audio input support! Don't confuse I2S with I2C, I2S is a sound protocol whereas I2C is for small amounts of data.

Power Pins

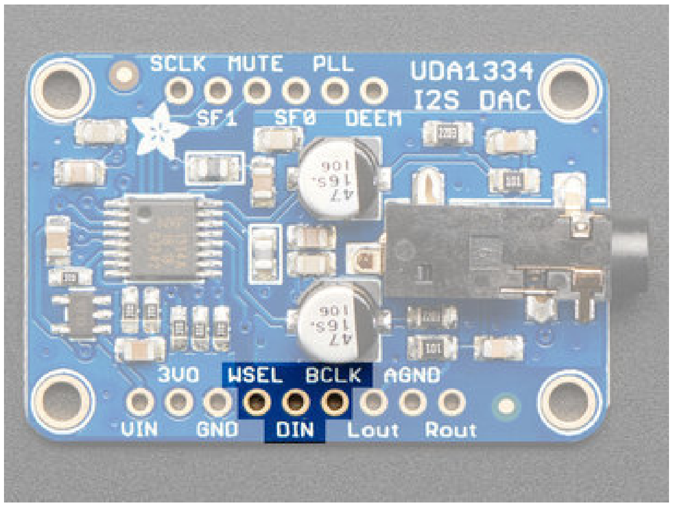


The UDA1334A requires 3.3V power but can take 3-5V level logic on nearly all pins.

You can provide 3-5V power on the **VIN** pin and **GND** and the built in regulator will generate a nice clean 3.3V supplier on **3VO**ut.

Use the quietest power supply for Vin, we do filter the power supply, but the quieter the better!

I2S Pins



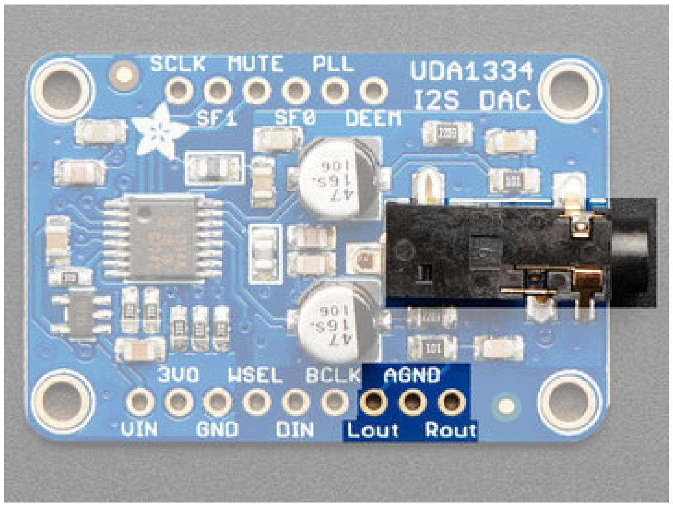
Three pins are used for stereo I2S data in. These pins are required!

These can be 3.3-5V logic

- **WSEL** (Word Select or Left/Right Clock) - this is the pin that tells the DAC when the data is for the left channel and when its for the right channel
- **DIN** (Data In) - This is the pin that has the actual data coming in, both left and right data are sent on this pin, the WSEL pin indicates when left or right is being transmitted
- **BCLK** (Bit Clock) - This is the pin that tells the amplifier when to read data on the data pin.

MCLK is not required to use this DAC, if you have an MCLK pin on your audio source, leave it disconnected.

Audio Outputs



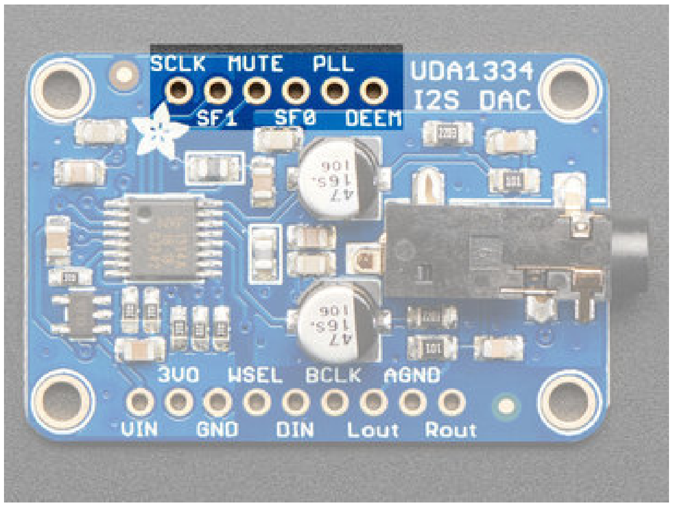
The exciting part! This is where your line level audio comes out. We put big 47uF blocking capacitors on the output so you can connect this to any stereo system. **AGND** is a clean analog ground signal that we recommend using as your analog reference, you'll get a cleaner signal.

Note that this DAC was intended for use with a separate amplifier and is rated for a 3 KΩ load. However, we've found you *can* plug in 32Ω headphones and the output is current-limited so it won't damage the DAC but you will get distortions. (Powered headphones won't have this issue)

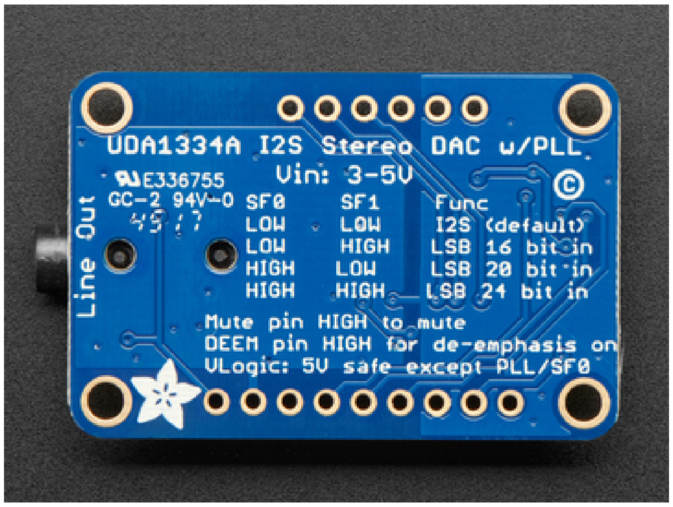
Optional Control Pins

There are some extra configuration pins if you want to use them. They are not required for 99% of usage with an Arduino or Teensy or Raspberry Pi. But you never know! So they are there for you. **PLL** and **SF0** are 3.3V logic only, the other pins are 3-5V safe.

Most of the pins have to do with changing the setup from audio mode to video mode. If you happen to want video-mode, for synchronizing with NTSC/PAL, check the datasheet - we haven't used it for that purpose.



- **SCLK (Sys Clock)** - Optional 27 MHz 'video mode' ssytem clock input - by default we generate the sysclock from the WS clock in 'audio mode' But the UDA can also take a oscillator input on this pin
- **Mute** - Setting this pin High will mute the output
- **De-Em**phasis - In audio mode (which is the default), can be used to add a de-emphasis filter. In video mode, where the system clock is generated from an oscillator, this is the clock output.
- **PLL** - sets the PLL mode, by default pulled low for Audio. Can be pulled high or set to ~1.6V to set PAL or NTSC video frequency



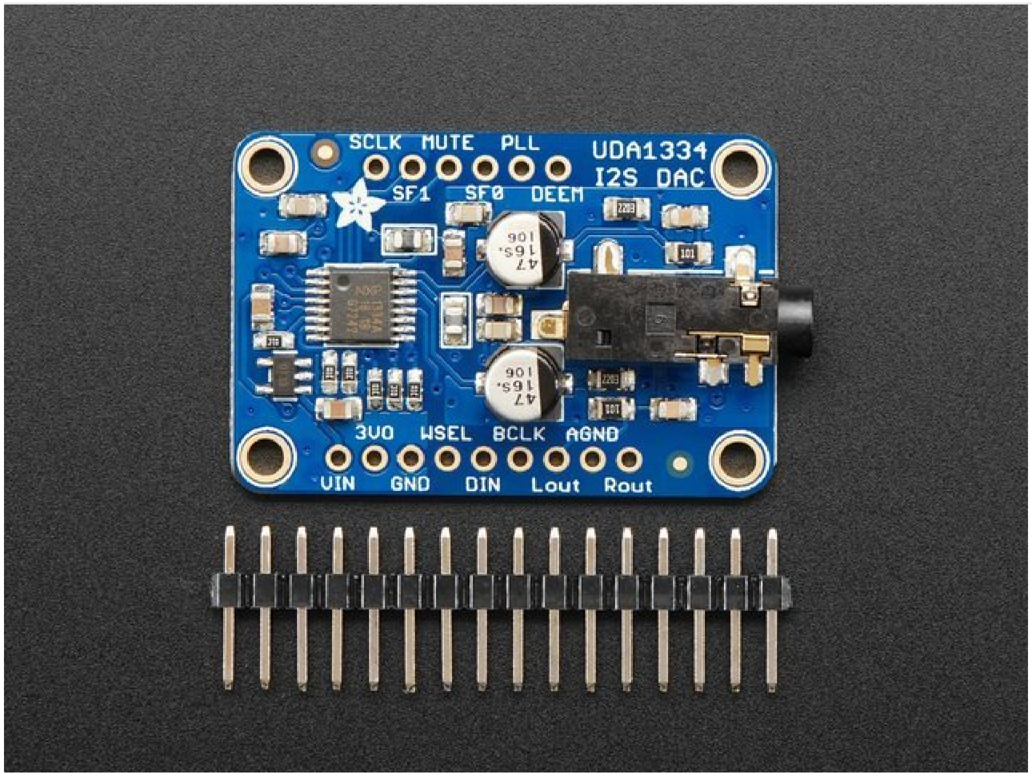
SF0	SF1	Func
LOW	LOW	I2S (default)
LOW	HIGH	LSB 16 bit in
HIGH	LOW	LSB 20 bit in
HIGH	HIGH	LSB 24 bit in

Mute pin HIGH to mute
DEEM pin HIGH for de-emphasis on
VLogic: 5V safe except PLL/SF0

SF0 and **SF1** are used to set the input data format. By default both are pulled Low for I2S but you can change them around for alternate formats.

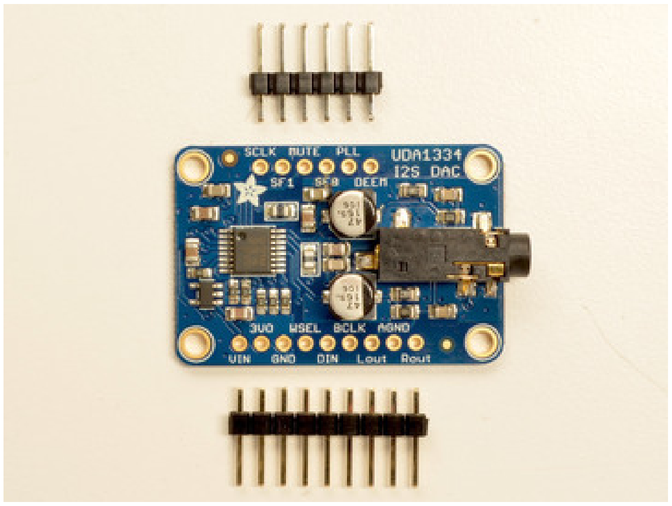
See the back of the PCB for a quick reference

Assembly

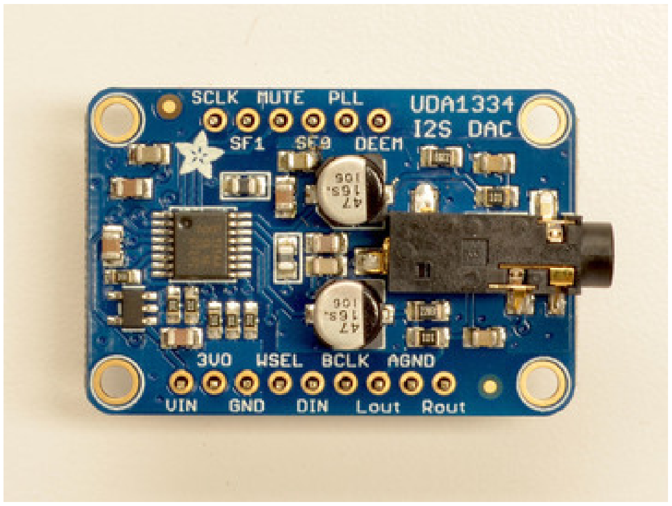


Installing Standard Headers

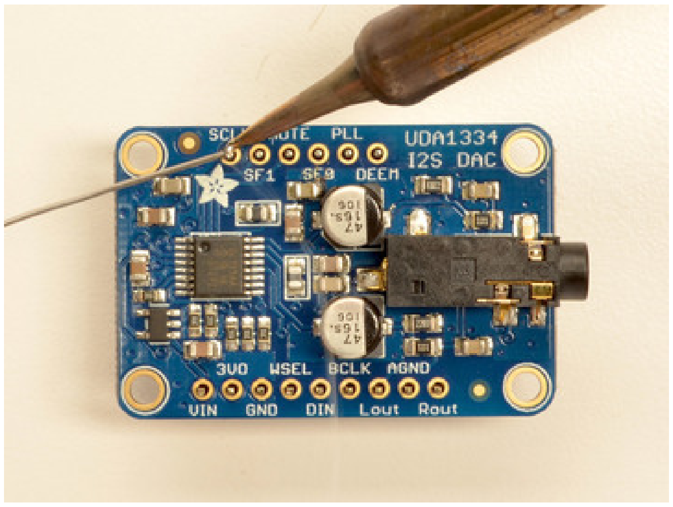
The shield comes with 0.1" standard header.



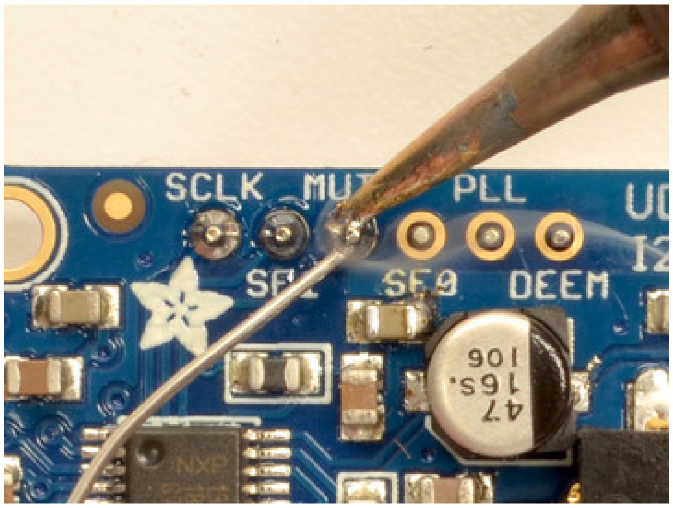
Break apart the 0.1" header into 6 and 9-pin long pieces and slip the short ends into the holes in the board

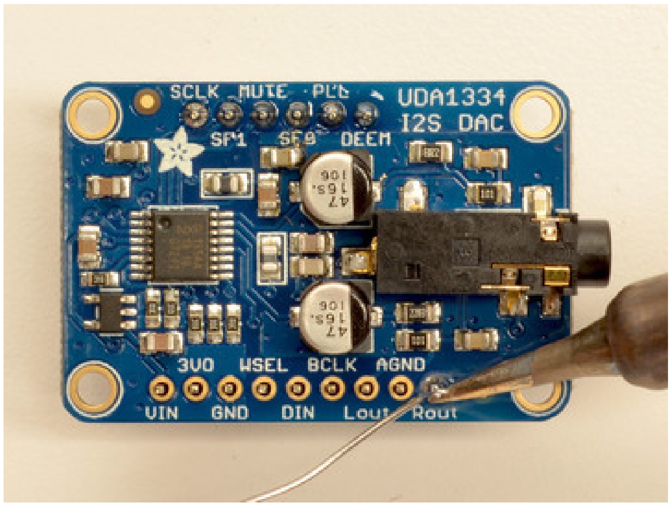


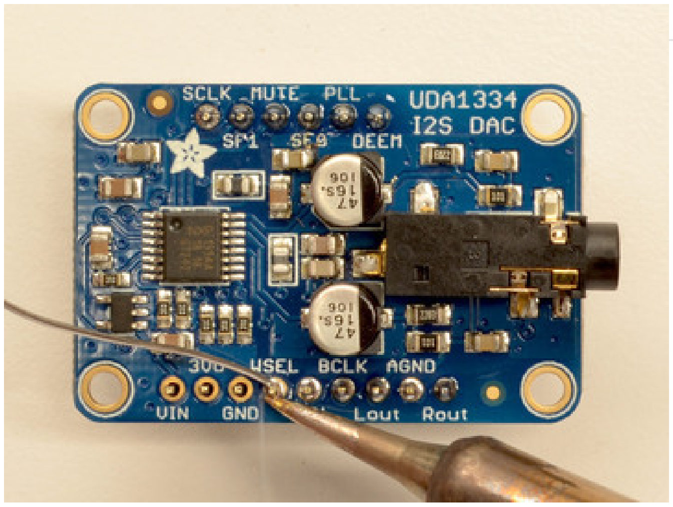
Make sure that all of the short parts of the header are sticking through the two sets of pads on either side of the board

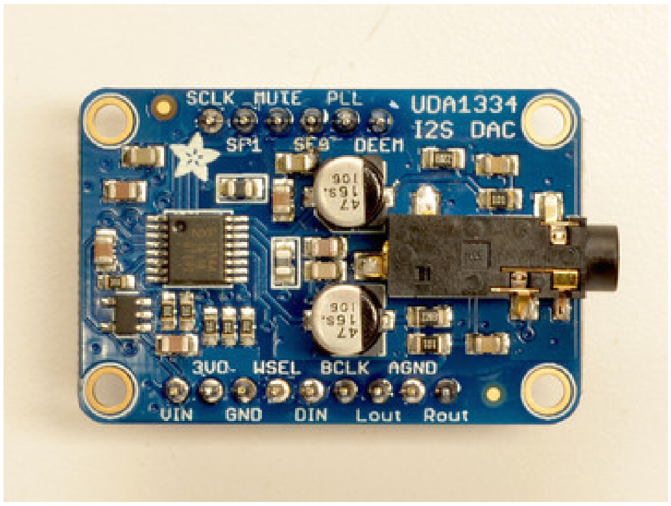


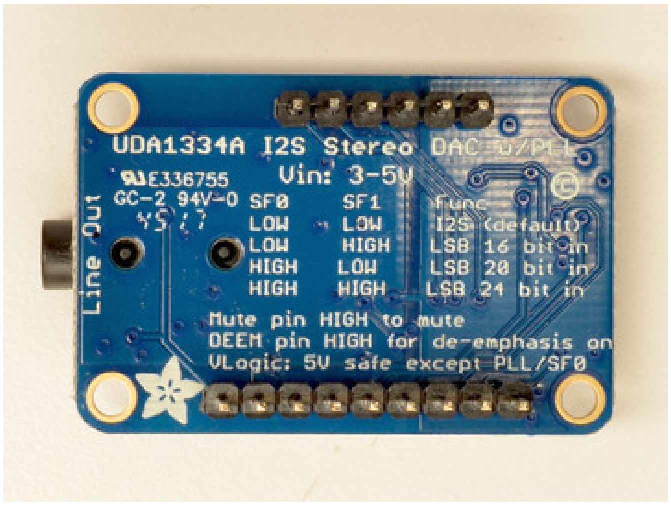
Solder each one of the pins into the board to make a secure connection


SCLK
MUTE
PLL
SF1
SF0
DEEM


SCLK MUTE PLL
UDA1334
I2S DAC
SF1 SF0 DEEM
3VO WSEL BCLK AGND
VIN GND DIN Lout Rout


SCLK MUTE PLL
UDA1334
I2S DAC
SF1 SF0 DEEM
3VO WSEL BCLK AGND
VIN GND Lout Rout





That's it! Move on to next page for wiring information

Raspberry Pi Wiring

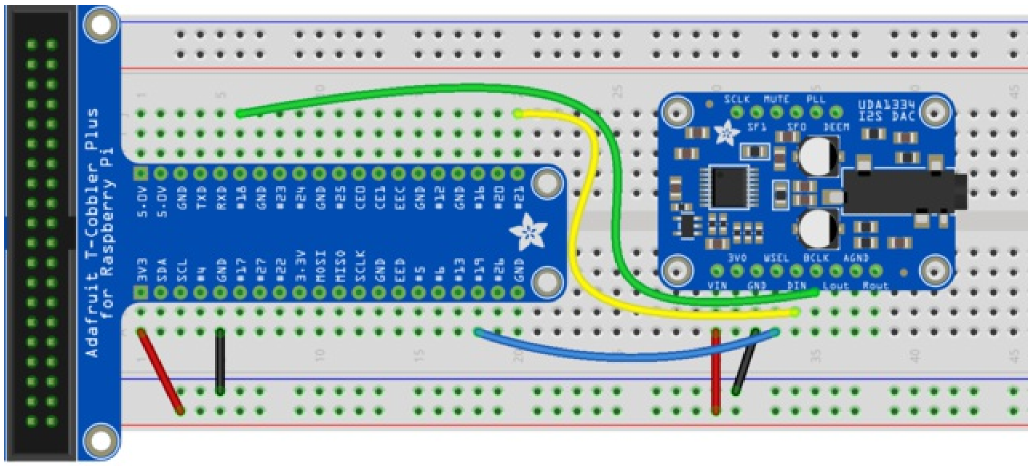
if you have a Raspberry Pi and you want higher quality audio than the headphone jack can provide, I2S is a good option! You only use 3 pins, and since its a pure-digital output, there can be less noise and interference.

This board works very well with boards that *don't* have audio like the Pi Zero and is the easiest way to get quality audio out

This technique will work with any Raspberry Pi with the 2x20 connector. Older Pi 1's with a 2x13 connector do not bring out the I2S pins as easily

Connect:

- **Amp Vin** to Raspbery Pi **3V** or **5V**
- **Amp GND** to Raspbery Pi **GND**
- **Amp DIN** to Raspbery Pi **#21**
- **Amp BCLK** to Raspbery Pi **#18**
- **Amp LRCLK** to Raspbery Pi **#19**


Adafruit T-Cobbler Plus for Raspberry Pi
5.0V 5.0V GND TXD RXD #18 GND #23 #24 GND #25 CE0 CE1 EEC GND #12 GND #16 #20 #21
3V3 SDA SCL #4 GND #17 #27 #22 3.3V MOSI MISO SCLK GND EED #5 #6 #13 #19 #26 GND
SCLK MUTE PLL
SF1 SFO DEEM
UDA1334 I2S DAC
3VO WSEL BCLK AGND
VIN GND DIN Lout Rout

Pi + UDA Fritzing

https://adafru.it/A9T

Raspberry Pi Setup

At this time, Jessie Raspbery Pi kernel does not support mono audio out of the I2S interface, you can only play stereo, so any mono audio files may need conversion to stereo!

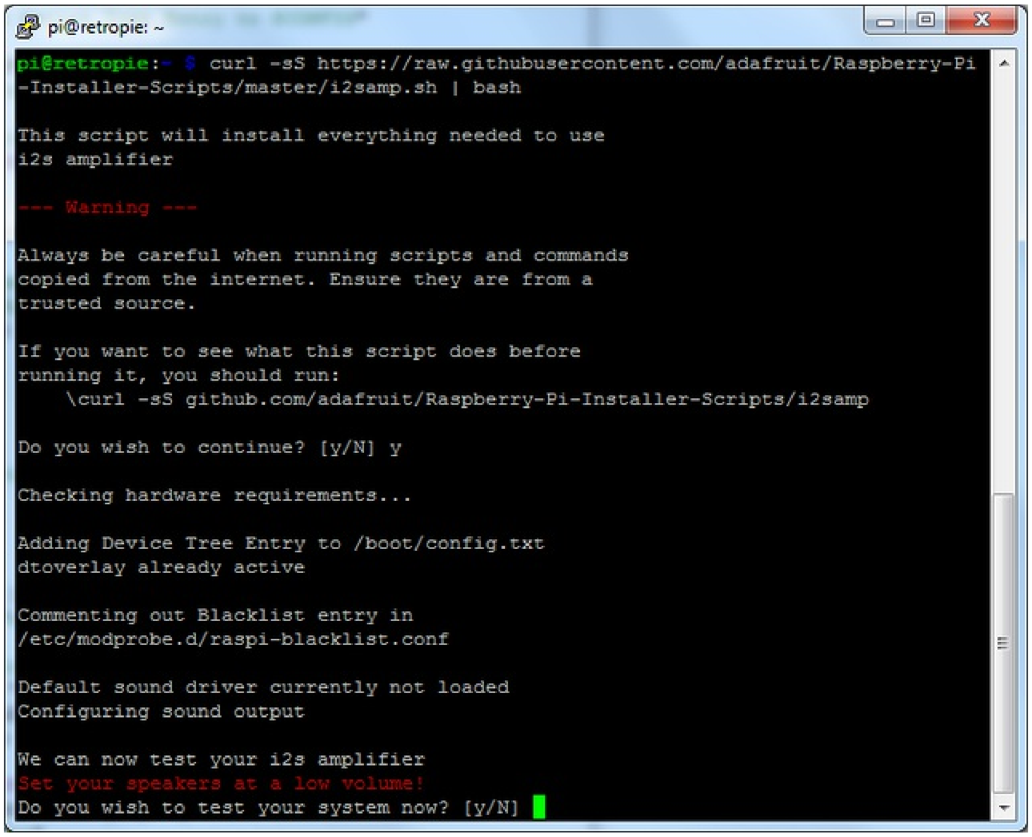
Fast Install

Luckily its quite easy to install support for I2S DACs on Raspbian Jessie.

These instructions are totally cribbed from the PhatDAC instructions at the lovely folks at Pimoroni!

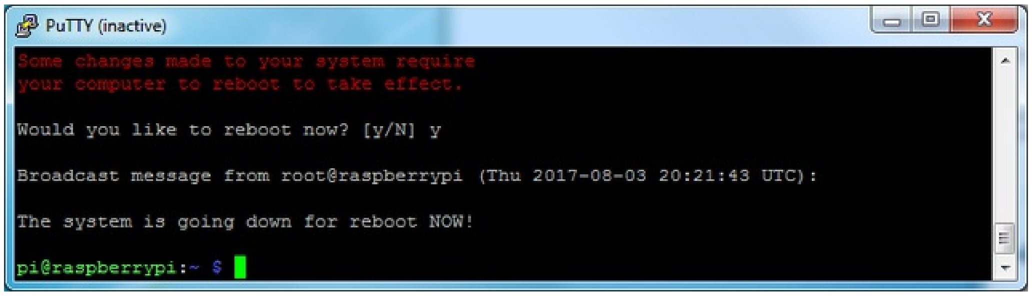
Run the following from your Raspberry Pi with Internet connectivity:

```
curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash
```



```
pi@retropie: ~
pi@retropie:~ $ curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi
-Installer-Scripts/master/i2samp.sh | bash

This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay already active

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Default sound driver currently not loaded
Configuring sound output

We can now test your i2s amplifier
Set your speakers at a low volume!
Do you wish to test your system now? [y/N]
```

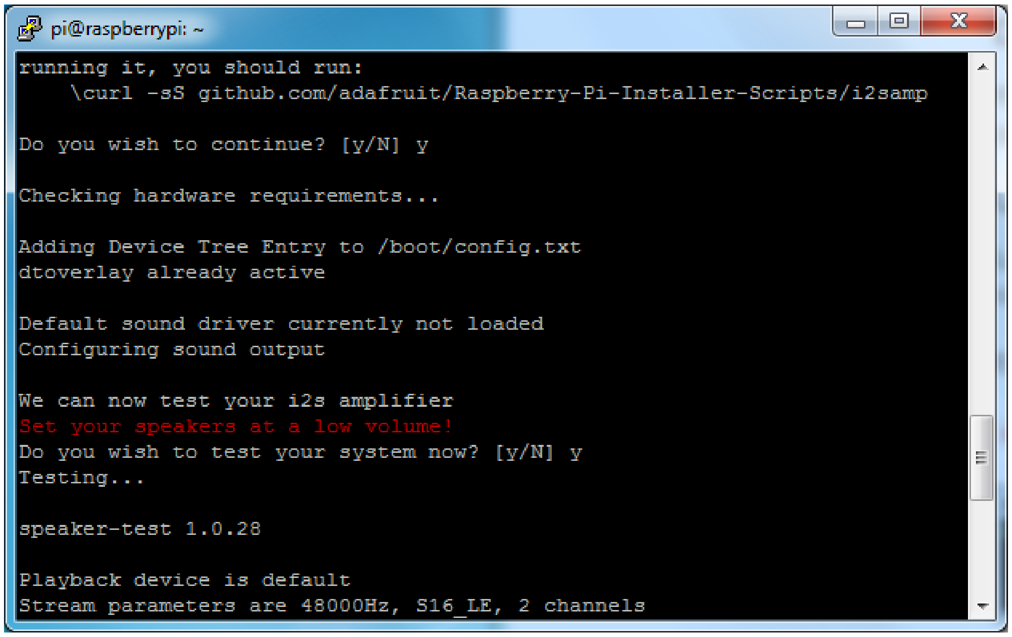
You will need to reboot once installed.



```
PuTTY (inactive)
Some changes made to your system require
your computer to reboot to take effect.

Would you like to reboot now? [y/N] y

Broadcast message from root@raspberrypi (Thu 2017-08-03 20:21:43 UTC):

The system is going down for reboot NOW!

pi@raspberrypi:~ $
```

You must reboot to enable the speaker hardware!

After rebooting, log back in and re-run the script again...It will ask you if you want to test the speaker. Say yes and listen for audio to come out of your speakers...



```
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay already active

Default sound driver currently not loaded
Configuring sound output

We can now test your i2s amplifier
Set your speakers at a low volume!
Do you wish to test your system now? [y/N] y
Testing...

speaker-test 1.0.28

Playback device is default
Stream parameters are 48000Hz, S16_LE, 2 channels
```

In order to have volume control appear in Raspbian desktop or Retropie you must reboot a second time after doing the speaker test, with **sudo reboot**

You can then go to the next page on testing and optimizing your setup. Skip the rest of this page on **Detailed Installation** if the script worked for you!

Detailed Install

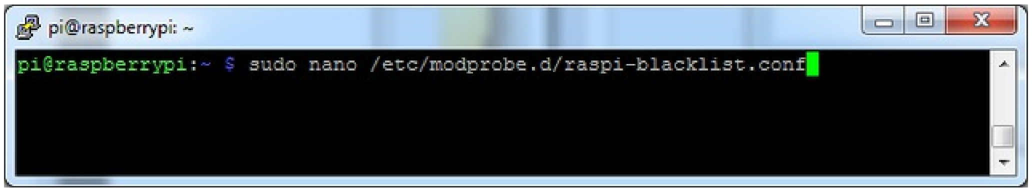
If, for some reason, you can't just run the script and you want to go through the install by hand - here's all the steps!

Update /etc/modprobe.d (if it exists)

Log into your Pi and get into a serial console (either via a console cable, the TV console, RXVT, or what have you)

Edit the raspi blacklist with

`sudo nano /etc/modprobe.d/raspi-blacklist.conf`

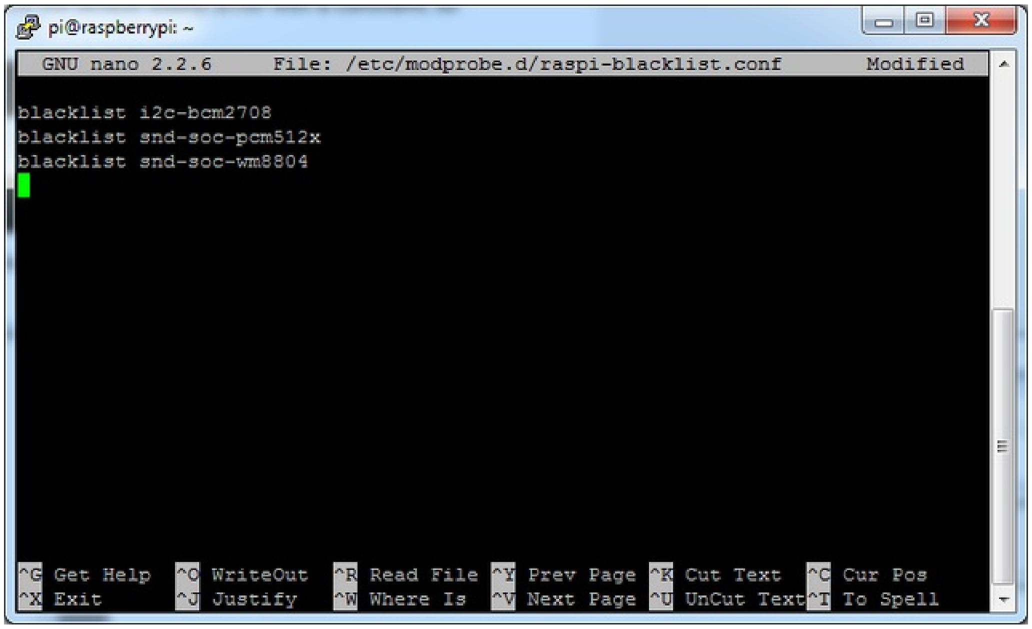


```
pi@raspberrypi:~ $ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

If the file is empty, just skip this step

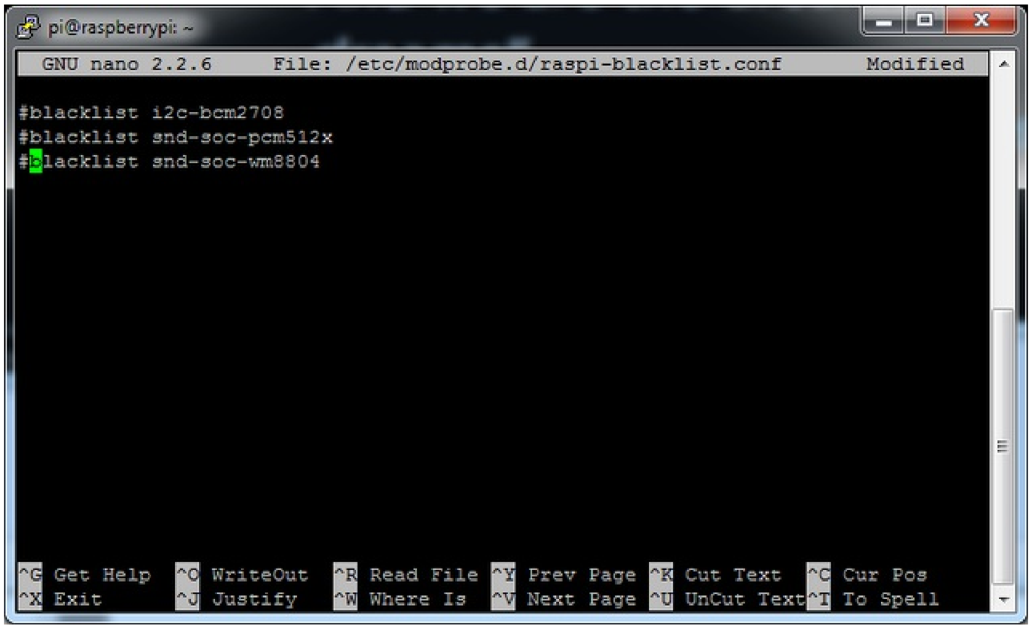
However, if you see the following lines:

```
blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804
```



```
GNU nano 2.2.6      File: /etc/modprobe.d/raspi-blacklist.conf        Modified

blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Update the lines by putting a # before each line



```
GNU nano 2.2.6      File: /etc/modprobe.d/raspi-blacklist.conf        Modified

#blacklist i2c-bcm2708
#blacklist snd-soc-pcm512x
#blacklist snd-soc-wm8804

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Save by typing **Control-X Y <return>**

Disable headphone audio (if it's set)

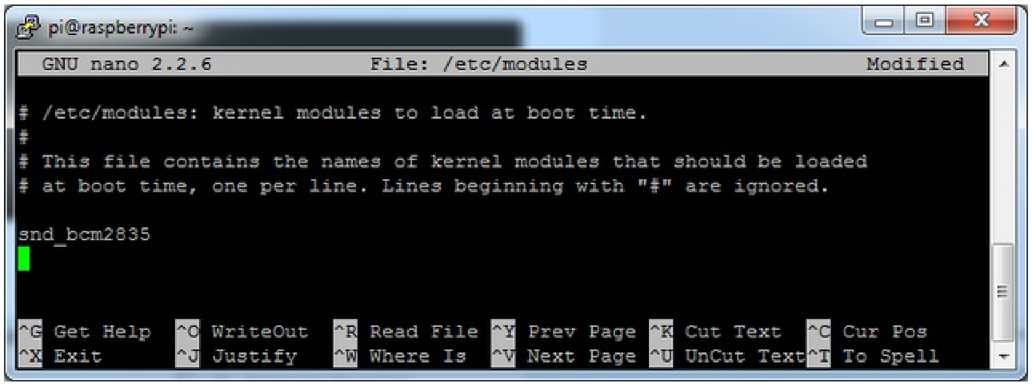
Edit the raspi modules list with

```
sudo nano /etc/modules
```

If the file is empty, just skip this step

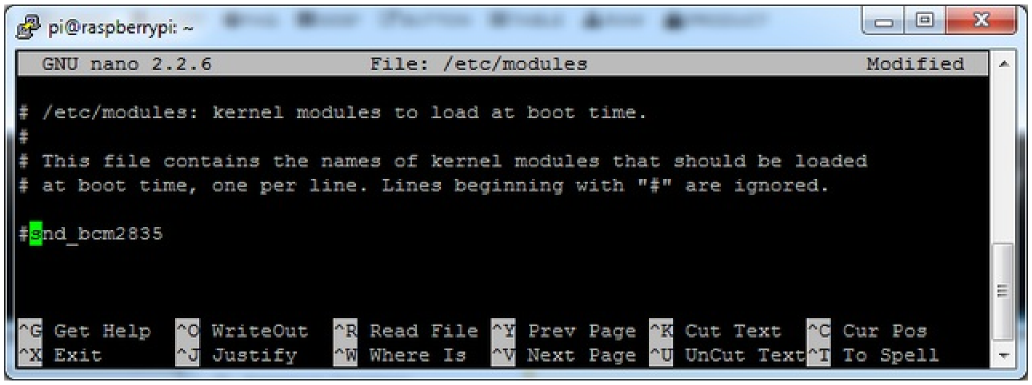
However, if you see the following line:

`snd_bcm2835`



```
pi@raspberrypi: ~
  GNU nano 2.2.6              File: /etc/modules                         Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

snd_bcm2835


^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

Put a # in front of it



```
pi@raspberrypi: ~
  GNU nano 2.2.6              File: /etc/modules                         Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

#snd_bcm2835


^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

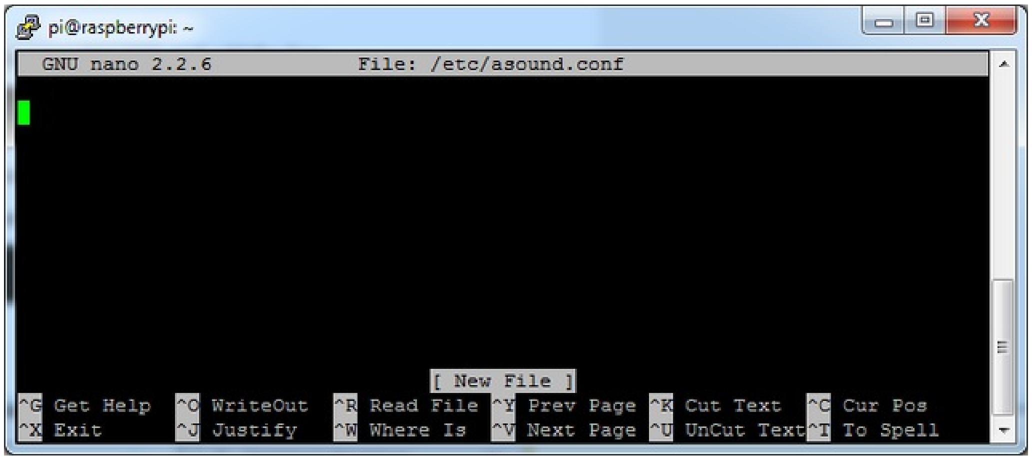
and save with **Control-X Y <return>**

Create asound.conf file

Edit the raspi modules list with

`sudo nano /etc/asound.conf`

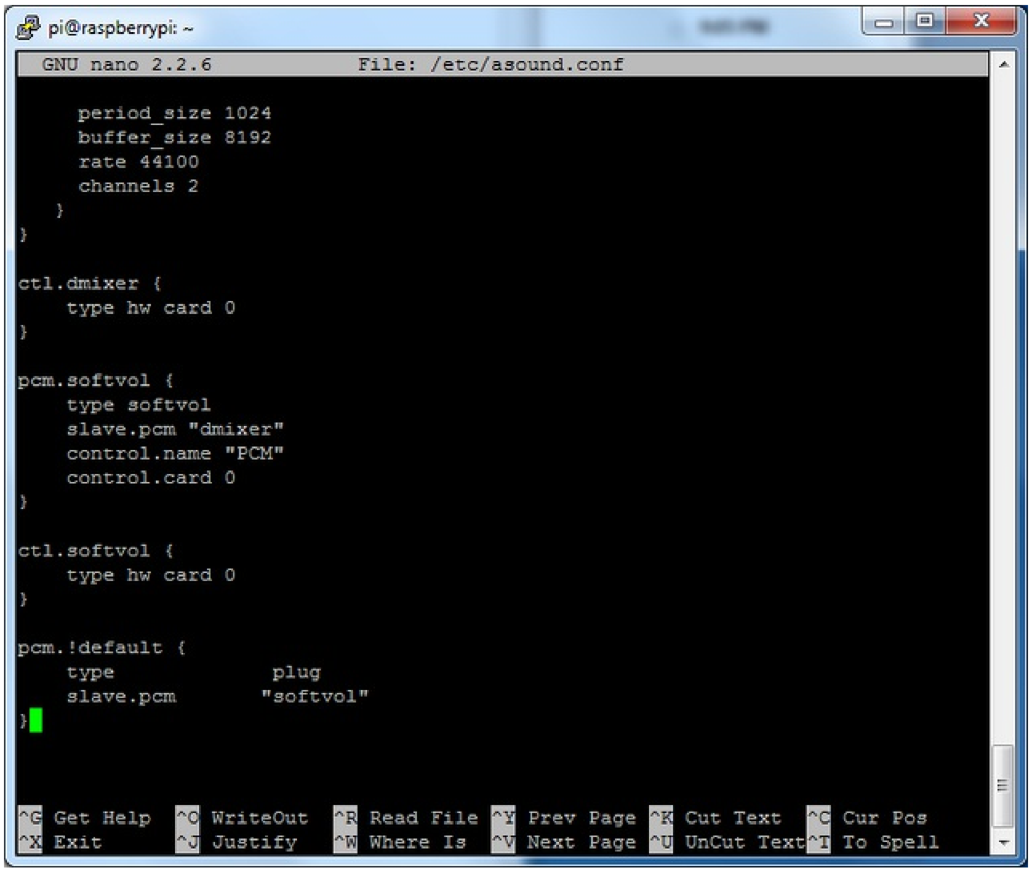
This file ought to be blank!



```
  GNU nano 2.2.6              File: /etc/asound.conf

                                  [ New File ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Copy and paste the following text into the file

```
pcm.speakerbonnet {
   type hw card 0
}

pcm.dmixer {
   type dmix
   ipc_key 1024
   ipc_perm 0666
   slave {
     pcm "speakerbonnet"
     period_time 0
     period_size 1024
     buffer_size 8192
     rate 44100
     channels 2
   }
}

ctl.dmixer {
    type hw card 0
}

pcm.softvol {
    type softvol
    slave.pcm "dmixer"
    control.name "PCM"
    control.card 0
}

ctl.softvol {
    type hw card 0
}

pcm.!default {
    type             plug
    slave.pcm       "softvol"
}
```



```
pi@raspberrypi: ~
  GNU nano 2.2.6            File: /etc/asound.conf

     period_size 1024
     buffer_size 8192
     rate 44100
     channels 2
   }
}

ctl.dmixer {
    type hw card 0
}

pcm.softvol {
    type softvol
    slave.pcm "dmixer"
    control.name "PCM"
    control.card 0
}

ctl.softvol {
    type hw card 0
}

pcm.!default {
    type             plug
    slave.pcm       "softvol"
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

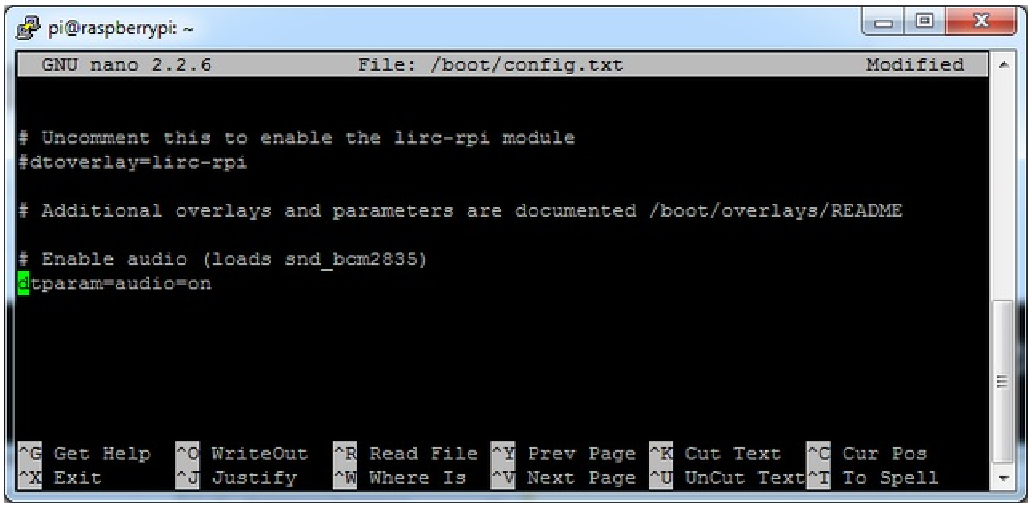
Save the file as usual

Add Device Tree Overlay

Edit your Pi configuration file with

`sudo nano /boot/config.txt`

And scroll down to the bottom. If you see a line that says: `dtparam=audio=on`

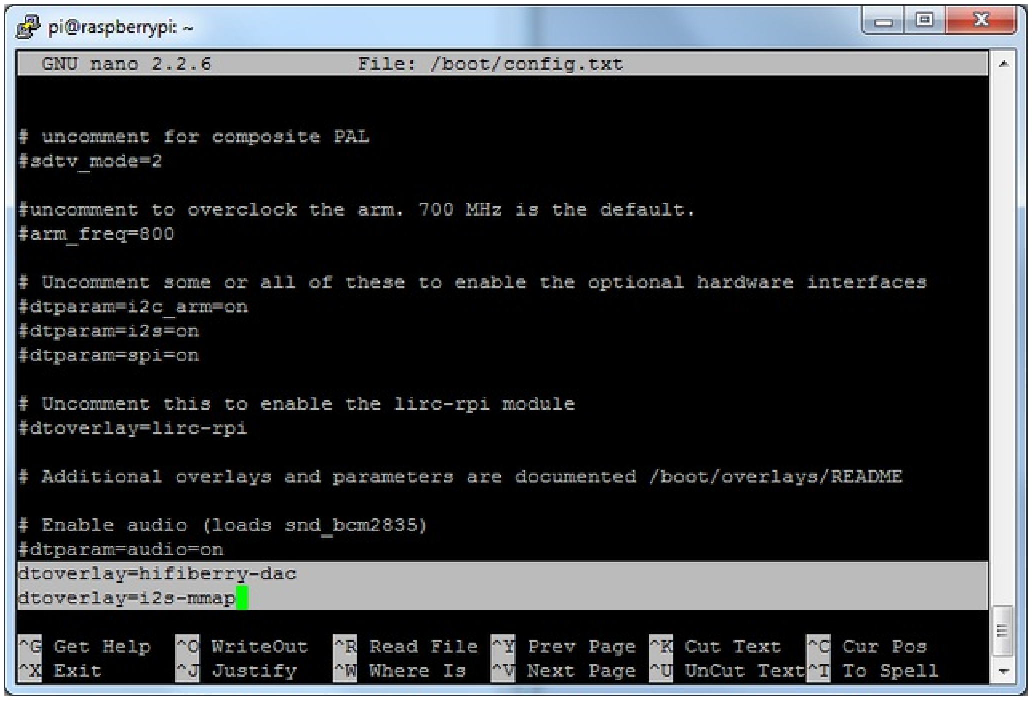


```
pi@raspberrypi: ~
  GNU nano 2.2.6            File: /boot/config.txt                    Modified


# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on




^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Disable it by putting a # in front.

Then add:
`dtoverlay=hifiberry-dac`
`dtoverlay=i2s-mmap`

on the next line. Save the file.



pi@raspberrypi: ~

```
  GNU nano 2.2.6              File: /boot/config.txt


# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Reboot your Pi with **sudo reboot**

Raspberry Pi Test

Speaker Tests!

OK you can use whatever software you like to play audio but if you'd like to test the speaker output, here's some quick commands that will let you verify your amp and speaker are working as they should!

Simple white noise speaker test

Run `speaker-test -c2` to generate white noise out of the speaker, alternating left and right. Since the I2S amp merges left and right channels, you'll hear continuous white noise

Simple WAV speaker test

Once you've got something coming out, try to play an audio file with **speaker-test** (for WAV files, not MP3)

```
speaker-test -c2 --test=wav -w /usr/share/sounds/alsa/Front_Center.wav
```

You'll hear audio coming from left and right alternating speakers

Simple MP3 speaker test

If you want to play a stream of music, you can try

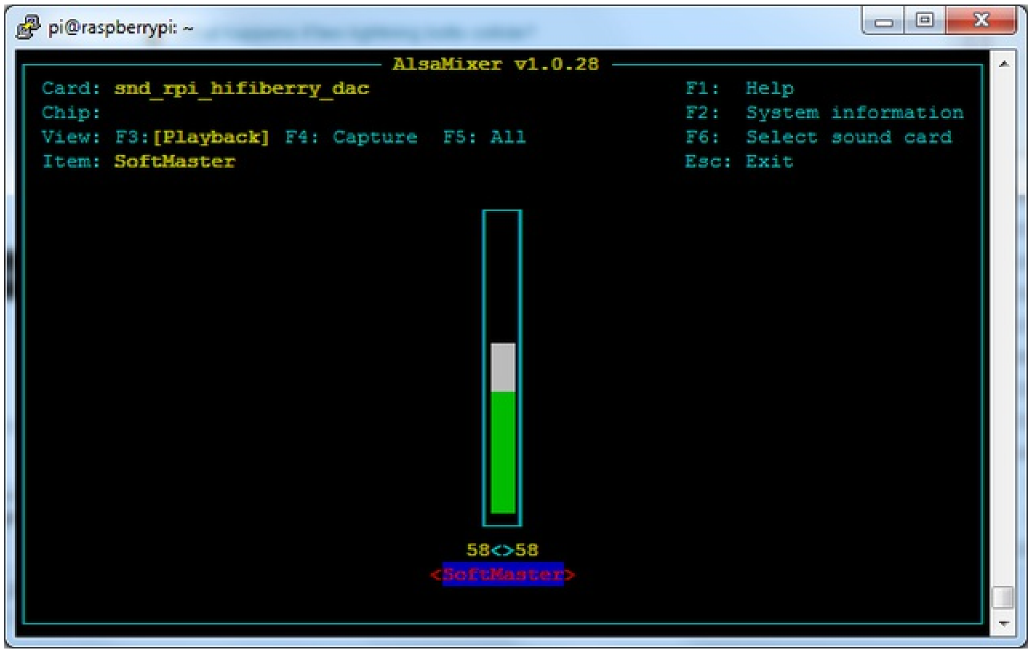
```
sudo apt-get install -y mpg123
mpg123 http://ice1.somafm.com/u80s-128-mp3
```

If you want to play MP3's on command, check out this tutorial which covers how to set that up

At this time, Jessie Raspbery Pi kernel **does not support mono audio** out of the I2S interface, **you can only play stereo**, so any mono audio files may need conversion to stereo!

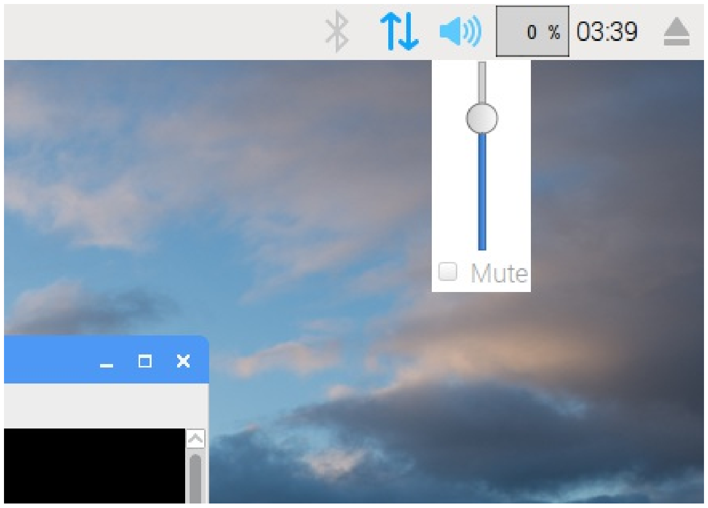
Volume adjustment

Many programs like PyGame and Sonic Pi have volume control within the application. For other programs you can set the volume using the command line tool called **alsamixer**. Just type alsamixer in and then use the up/down arrows to set the volume. Press Escape once its set



```
pi@raspberrypi: ~
                         AlsaMixer v1.0.28
 Card: snd_rpi_hifiberry_dac                          F1:  Help
 Chip:                                                F2:  System information
 View: F3:[Playback] F4: Capture  F5: All             F6:  Select sound card
 Item: SoftMaster                                     Esc: Exit

                              58<>58
                           <SoftMaster>
```

In Raspbian PIXEL you can set the volume using the menu item control. If it has an X through it, try restarting the Pi (you have to restart twice after install to get PIXEL to recognize the volume control



Pi I2S Tweaks

Reducing popping

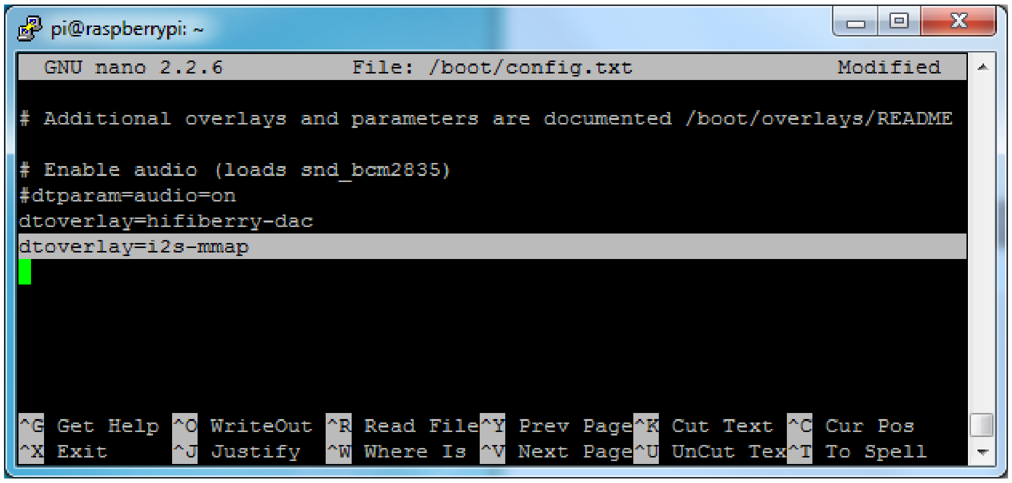
For people who followed our original installation instructions with the simple alsa config, they may find that the I2S audio pops when playing new audio.

The workaround is to use a software mixer to output a fixed sample rate to the I2S device so the bit clock does not change. I use ALSA so I configured **dmixer** and I no longer have any pops or clicks. Note that the RaspPi I2S driver does not support **dmixer** by default and you must follow these instructions provided to add it. Continue on for step-by-step on how to enable it!

Step 1

Start by modify **/boot/config.txt** to add `dtoverlay=i2s-mmap`

Run **sudo nano /boot/config.txt** and add the text to the bottom like so:



```
  GNU nano 2.2.6            File: /boot/config.txt                  Modified

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Save and exit.

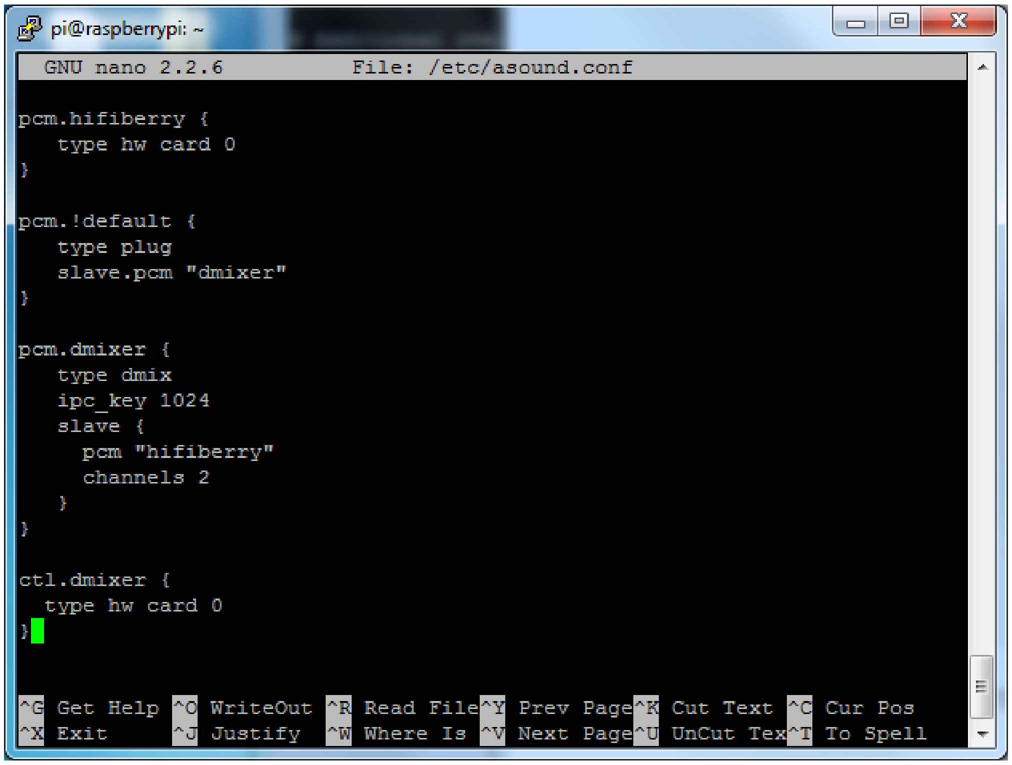
Then change **/etc/asound.conf** to:

```
pcm.speakerbonnet {
   type hw card 0
}

pcm.!default {
   type plug
   slave.pcm "dmixer"
}

pcm.dmixer {
   type dmix
   ipc_key 1024
   ipc_perm 0666
   slave {
     pcm "speakerbonnet"
     period_time 0
     period_size 1024
     buffer_size 8192
     rate 44100
     channels 2
   }
}

ctl.dmixer {
  type hw card 0
}
```

By running `sudo nano /etc/asound.conf`

This creates a PCM device called speakerbonnet which is connected to the hardware I2S device. Then we make a new 'dmix' device (`type dmix`) called `pcm.dmixer` . We give it a unique Inter Process Communication key (`ipc_key 1024`) and permissions that are world-read-writeable `(ipc_perm 0666`) The mixer will control the hardware pcm device speakerbonnet (pcm "speakerbonnet") and has a buffer set up so its nice and fast. The communication buffer is set up so there's no delays (`period_time 0` , `period_size 1024` and `buffer_size 8192` work well). The default mixed rate is 44.1khz stereo (`rate 44100 channels 2`)

Finally we set up a control interface but it ended up working best to just put in the hardware device here - `ctl.dmixer { type hw card 0 }`



```
  GNU nano 2.2.6             File: /etc/asound.conf

pcm.hifiberry {
   type hw card 0
}

pcm.!default {
   type plug
   slave.pcm "dmixer"
}

pcm.dmixer {
   type dmix
   ipc_key 1024
   slave {
     pcm "hifiberry"
     channels 2
   }
}

ctl.dmixer {
  type hw card 0
}


^G Get Help ^O WriteOut ^R Read File^Y Prev Page^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

Save and exit. Then reboot the Pi to enable the mixer. Also, while it will *greatly* reduce popping, you still may get one once in a while - especially when first playing audio!

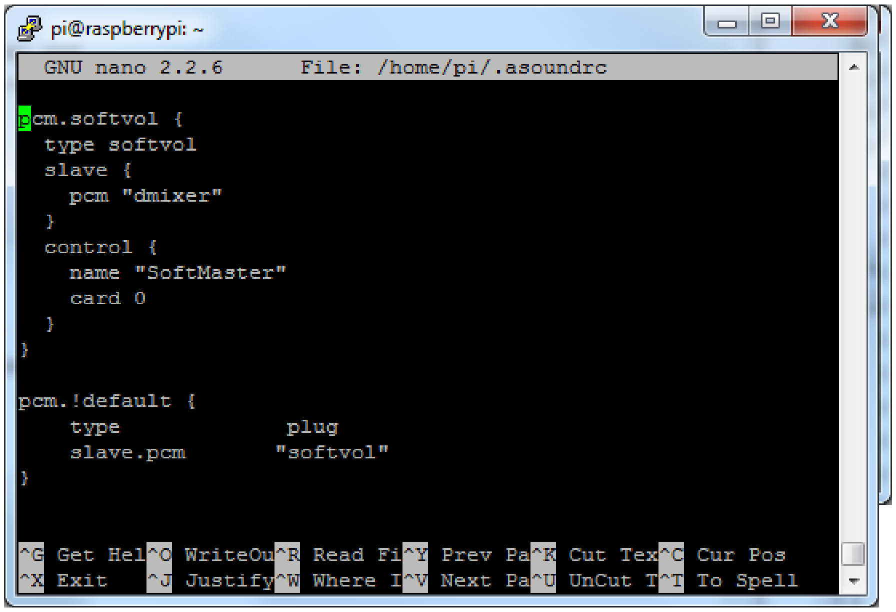
Add software volume control

The basic I2S chipset used here does not have software control built in. So we have to 'trick' the Pi into creating a software volume control. Luckily, its not hard once you know how to do it.

Create a new audio config file in **~/.asoundrc** with `nano ~/.asoundrc` and inside put the following text:

```
pcm.speakerbonnet {
   type hw card 0
}

pcm.dmixer {
   type dmix
   ipc_key 1024
   ipc_perm 0666
   slave {
     pcm "speakerbonnet"
     period_time 0
     period_size 1024
     buffer_size 8192
     rate 44100
     channels 2
   }
}

ctl.dmixer {
    type hw card 0
}

pcm.softvol {
    type softvol
    slave.pcm "dmixer"
    control.name "PCM"
    control.card 0
}

ctl.softvol {
    type hw card 0
}

pcm.!default {
    type             plug
    slave.pcm       "softvol"
}
```

This assumes you set up the dmixer for no-popping above!



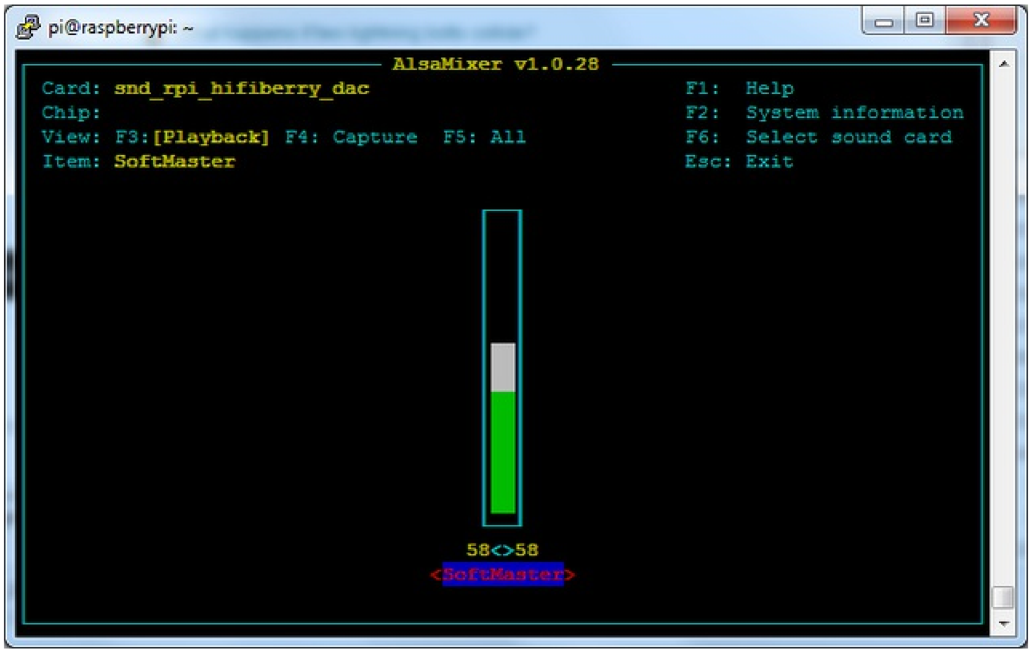
```
  GNU nano 2.2.6        File: /home/pi/.asoundrc

pcm.softvol {
  type softvol
  slave {
    pcm "dmixer"
  }
  control {
    name "SoftMaster"
    card 0
  }
}

pcm.!default {
    type             plug
    slave.pcm       "softvol"
}


^G Get Hel^O WriteOu^R Read Fi^Y Prev Pa^K Cut Tex^C Cur Pos
^X Exit   ^J Justify^W Where I^V Next Pa^U UnCut T^T To Spell
```

Save and exit

Now, here's the trick, you have to reboot, then play some audio through alsa, then reboot to get the alsamixer to sync up right:

```
speaker-test -c2 --test=wav -w /usr/share/sounds/alsa/Front_Center.wav
```

Then you can type **alsamixer** to control the volume with the 'classic' alsa mixing interface



```
                          AlsaMixer v1.0.28
 Card: snd_rpi_hifiberry_dac                        F1:  Help
 Chip:                                              F2:  System information
 View: F3:[Playback] F4: Capture  F5: All           F6:  Select sound card
 Item: SoftMaster                                   Esc: Exit

                               58<>58
                            <SoftMaster>
```

Just press the up and down arrows to set the volume, and ESC to quit

Play Audio with PyGame

You can use **mpg123** for basic testing but it's a little clumsy for use where you want to dynamically change the volume or have an interactive program. For more powerful audio playback we suggest using PyGame to playback a variety of audio formats (MP3 included!)

Install PyGame

Start by installing pygame support, you'll need to open up a console on your Pi with network access and run:

```
sudo apt-get install python-pygame
```

Next, download this pygame example zip to your Pi

Click to download PyGame example code & sample mp3s

https://adafru.it/wbp

On the command line, run

`wget https://cdn-learn.adafruit.com/assets/assets/000/041/506/original/pygame_example.zip`

`unzip pygame_example.zip`

Run Demo

Inside the zip is an example called **pygameMP3.py**

This example will playback all MP3's within the script's folder. To demonstrate that you can also adjust the volume within pygame, the second argument is the volume for playback. Specify a volume to playback with a command line argument between 0.0 and 1.0

For example here is how to play at 75% volume:

```
python pygameMP3.py 0.75
```

Here's the code if you have your own mp3s!

```
''' pg_midi_sound101.py
play midi music files (also mp3 files) using pygame
tested with Python273/331 and pygame192 by vegaseat
'''
#code modified by James DeVito from here: https://www.daniweb.com/programming/software-development/code/4


#!/usr/bin/python

import sys
import pygame as pg
import os
import time
```

```
def play_music(music_file):
    '''
    stream music with mixer.music module in blocking manner
    this will stream the sound from disk while playing
    '''
    clock = pg.time.Clock()
    try:
        pg.mixer.music.load(music_file)
        print("Music file {} loaded!".format(music_file))
    except pygame.error:
        print("File {} not found! {}".format(music_file, pg.get_error()))
        return

    pg.mixer.music.play()

    # If you want to fade in the audio...
    # for x in range(0,100):
    #     pg.mixer.music.set_volume(float(x)/100.0)
    #     time.sleep(.0075)
    # # check if playback has finished
    while pg.mixer.music.get_busy():
        clock.tick(30)


freq = 44100    # audio CD quality
bitsize = -16   # unsigned 16 bit
channels = 2    # 1 is mono, 2 is stereo
buffer = 2048   # number of samples (experiment to get right sound)
pg.mixer.init(freq, bitsize, channels, buffer)


if len(sys.argv) > 1:

    try:
        user_volume = float(sys.argv[1])
    except ValueError:
        print "Volume argument invalid. Please use a float (0.0 - 1.0)"
        pg.mixer.music.fadeout(1000)
        pg.mixer.music.stop()
        raise SystemExit

    print("Playing at volume: " + str(user_volume)+ "\n")
    pg.mixer.music.set_volume(user_volume)
    mp3s = []
    for file in os.listdir("."):
        if file.endswith(".mp3"):
            mp3s.append(file)

    print mp3s

    for x in mp3s:
        try:
            play_music(x)
            time.sleep(.25)
        except KeyboardInterrupt:
            # if user hits Ctrl/C then exit
            # (works only in console mode)
            pg.mixer.music.fadeout(1000)
            pg.mixer.music.stop()
            raise SystemExit
else:
    print("Please specify volume as a float! (0.0 - 1.0)")
```

Arduino Wiring & Test

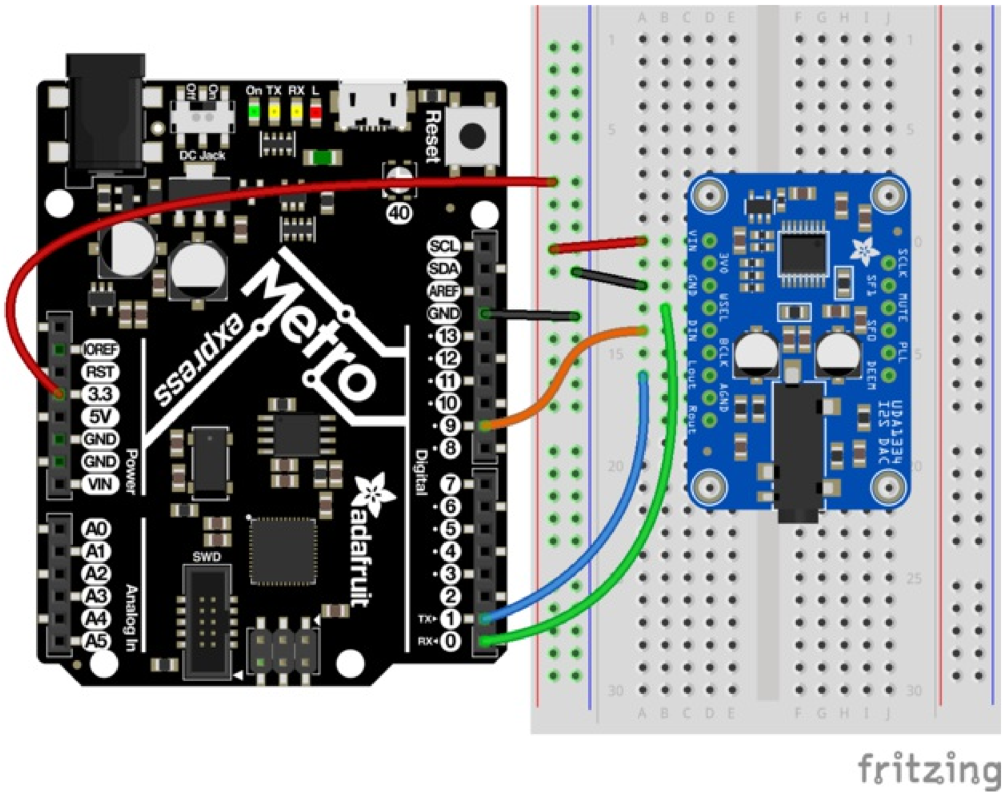
The classic ATmega328P-based Arduino's like the UNO and Metro 328 don't have I2S interfaces, so you *can't* use this breakout with them

But the newer ATSAMD21-based boards like the Zero, Metro M0, Feather M0 can! (Note, Gemma M0 & Trinket M0 do not have I2S pins available).

You may want to check your board documentation to determine where the I2S interface is. We'll use the Metro M0 pinout:

- **LRCLK / WSEL** on Digital #0
- **BCLK** on Digital #1
- **DATA IN** on Digital #9

Then power from 3.3V + Ground as usual:



We can then run this demo which will generate a sine wave on both left and right channels. Try tweaking the sample rate, amplitude and frequency, see how the quality and volume changes!

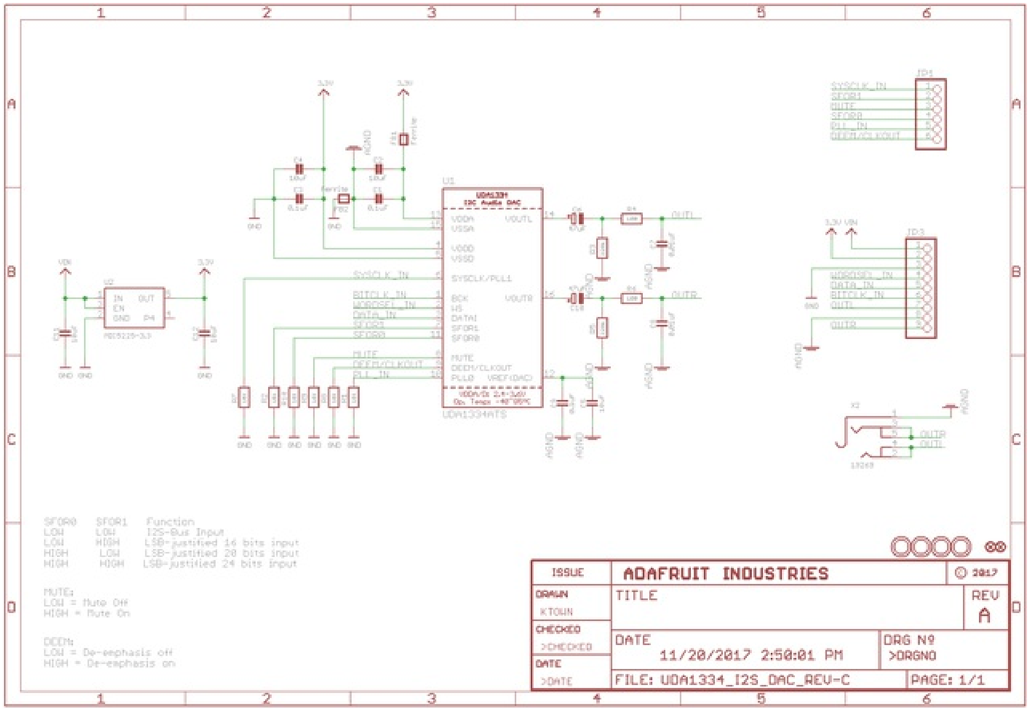
```
/*
 This example generates a sine wave based tone at a specified frequency
 and sample rate. Then outputs the data using the I2S interface.

 Public Domain
*/

#include <I2S.h>

#define FREQUENCY    440  // frequency of sine wave in Hz
#define AMPLITUDE  10000  // amplitude of sine wave
#define SAMPLERATE 44100  // sample rate in Hz

int16_t sinetable[SAMPLERATE / FREQUENCY];
uint32_t sample = 0;

#define PI 3.14159265

void setup() {
  Serial.begin(115200);
  Serial.println("I2S sine wave tone");

  // start I2S at the sample rate with 16-bits per sample
  if (!I2S.begin(I2S_PHILIPS_MODE, SAMPLERATE, 16)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }

  // fill in sine wave table
  for (uint16_t s=0; s < (SAMPLERATE / FREQUENCY); s++) {
    sinetable[s] = sin(2.0 * PI * s / (SAMPLERATE/FREQUENCY)) * AMPLITUDE;
  }
}


void loop() {
  if (sample == (SAMPLERATE / FREQUENCY)) {
    sample = 0;
  }

  // write the same sample twice, once for left and once for the right channel
  I2S.write((int16_t) sinetable[sample]);  // We'll just have same tone on both!
  I2S.write((int16_t) sinetable[sample]);

  // increment the counter for the next sample in the sine wave table
  sample++;
}
```

Downloads

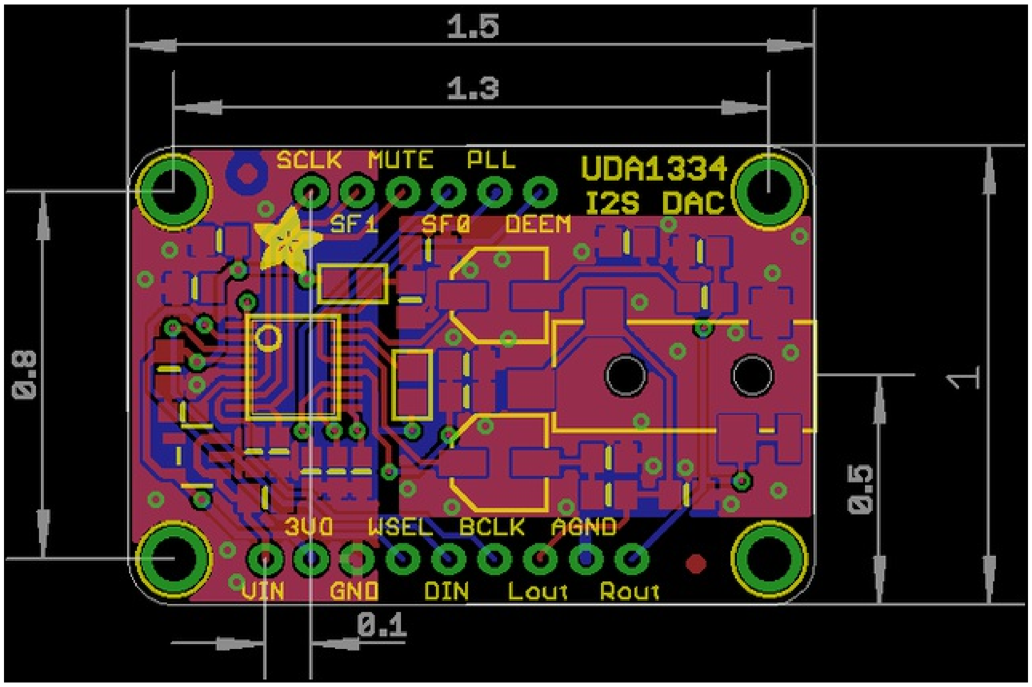
Files

- EagleCAD PCB Files
- Fritzing object in Adafruit Fritzing library
- UDA1334A Datasheet

Schematic & Fabrication Print



ADAFRUIT INDUSTRIES
TITLE
DATE 11/20/2017 2:50:01 PM
FILE: UDA1334_I2S_DAC_REV-C
PAGE: 1/1
UDA1334ATS



1.5
1.3
0.8
1
0.5
0.1
UDA1334
I2S DAC
SCLK
MUTE
PLL
SF1
SF0
DEEM
3V0
WSEL
BCLK
AGND
VIN
GND
DIN
Lout
Rout