Raspberry Pi Audio

Overview

Edit this on GitHub

Raspberry Pi Audio Boards bring high quality audio to your existing hi-fi or Raspberry Pi-based equipment and projects. We offer four different Hardware Attached on Top (HAT) options that will fit any Raspberry Pi using the 40-pin GPIO header.

Each board has a specific purpose and set of features. The highest audio quality playback is available from our DAC PRO, DAC+ and DigiAMP+ boards, which support up to full HD audio (192kHz); while the Codec Zero supports up to HD audio (96kHz) and includes a built-in microphone, making it ideal for compact projects.

Features at a glance

	Line out	Balanced out	Stereo speakers	Mono speaker	Headphones	Aux in	Aux out	Ext mic	Built-in mic
DAC Pro	✓	✓			✓				
DAC+	✓				✓				
DigiAmp+			✓						
Codec Zero				✓		✓	✓	✓	✓

Line out
: A double phono/RCA connector, normally red and white in colour. This output is a variable analogue signal (0-2V RMS) and can connect to your existing hi-fi (pre-amp or amplifier), or can be used to drive active speakers which have their own amplifier built in.

Balanced out
: An XLR connector, normally a three-pin male connector. This is used in a studio set-up, and in some high-end hi-fi systems. It can also be used to drive active monitor speakers like those used at clubs or on stage directed towards the DJ or performers.

Stereo speakers
: Two sets of screw terminals for 2×25W speakers. These are for traditional hi-fi speakers without built-in amplification. These are known as passive speakers.

Mono speaker

A screw terminal for a single 1.2W speaker, as found in a transistor radio or similar.

Headphones

A 3.5mm jack socket delivering stereo audio for a set of headphones. The headphone amplifiers on the Raspberry Pi DAC boards can drive up to 80/90Ω impedance headphones.

Aux in

A double Phono/RCA connector or 3.5mm socket. Accepts analogue audio in up to 1V RMS. This can be used to record audio from a variable analogue source such as a mobile phone, MP3 player or similar.

Aux out

A double Phono/RCA connector or 3.5mm socket. Delivers analogue audio out up to 1V RMS. This can be used to feed audio into an amplifier at a reduced volume compared to Line out.

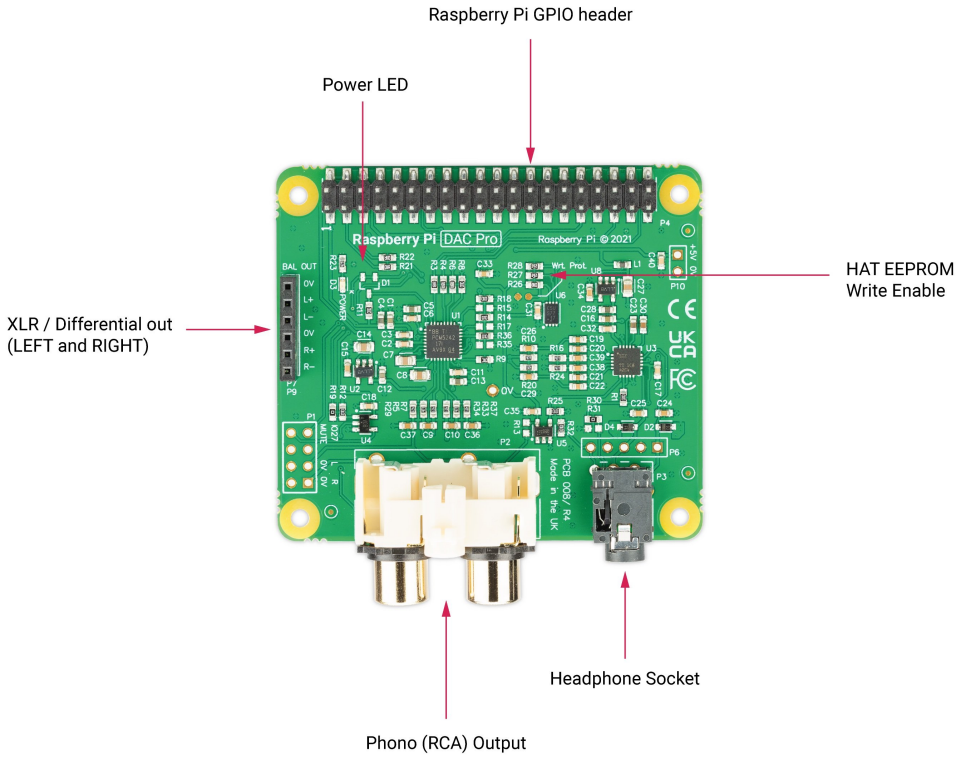
Ext mic

A 3.5mm socket for use with an external electret microphone. The built-in MEMS microphone on the Codec Zero is automatically disabled when the external Mic in connector is used.

Raspberry Pi DAC Pro

Edit this on GitHub

The Raspberry Pi DAC Pro HAT is our highest-fidelity digital to analogue converter (DAC).



With the Texas Instruments PCM5242, the DAC Pro provides outstanding signal-to-noise ratio (SNR) and supports balanced/differential output in parallel to phono/RCA line-level output. It also includes a dedicated headphone amplifier. The DAC Pro is powered by a Raspberry Pi through the GPIO header.

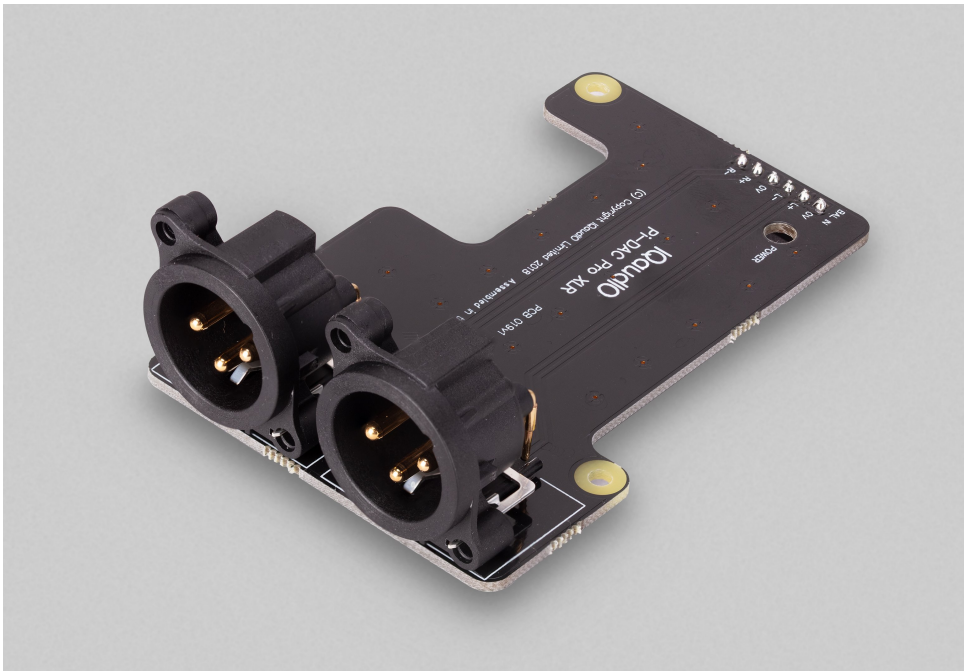
As part of the DAC Pro, two three-pin headers (P7/P9) are exposed above the Raspberry Pi's USB and Ethernet ports for use by the optional XLR board, allowing differential/balanced output.

Pinouts

P1	Analogue out (0-2V RMS), carries GPIO27, MUTE signal (headphone detect), left and right audio and left and right ground.
P6	Headphone socket signals (1: LEFT, 2: GROUND, 3: RIGHT, 4: GROUND, 5: DETECT).
P7/9	Differential (0-4V RMS) output (P7: LEFT, P9: RIGHT).
P10	Alternative 5V input, powering Raspberry Pi in parallel.

Optional XLR Board

The Pi-DAC PRO exposes a 6 pin header used by the optional XLR board to provide Differential / Balanced output exposed by XLR sockets above the Pi's USB/Ethernet ports.

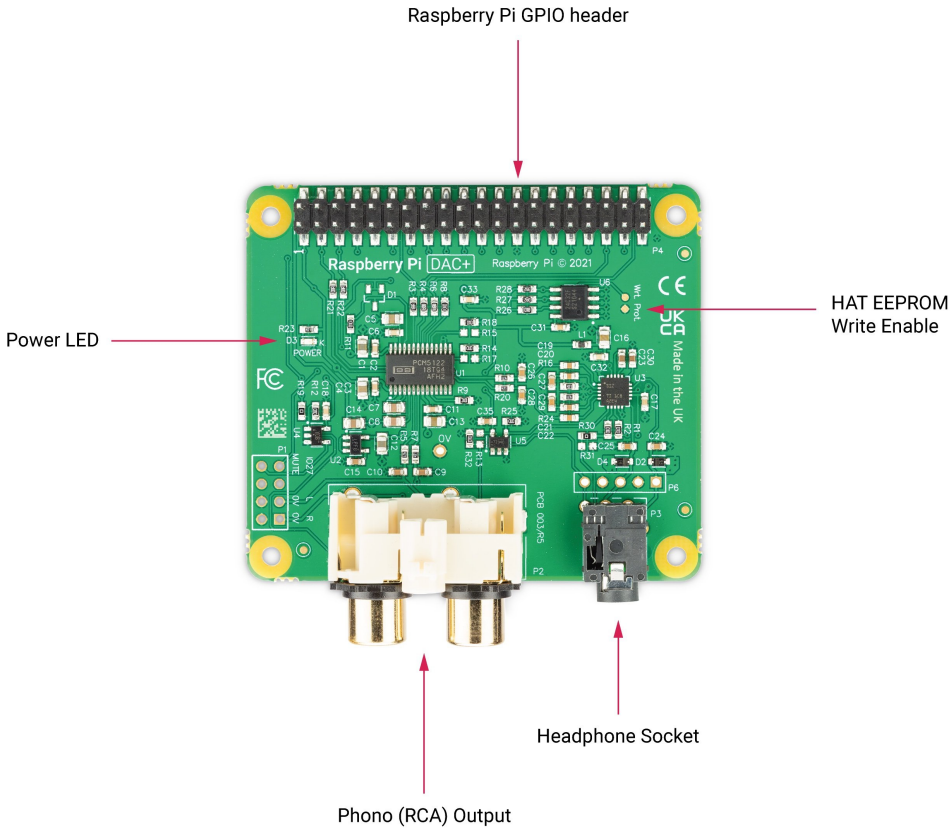


An XLR connector is used in Studio and some hi-end hifi systems. It can also be used to drive ACTIVE "monitor" speakers as used at discos or on stage.

Raspberry Pi DAC+

Edit this on GitHub

Raspberry Pi DAC+ is a high-resolution audio output HAT that provides 24-bit 192kHz digital audio output.



A Texas Instruments PCM5122 is used in the DAC+ to deliver analogue audio to the phono connectors of the device. It also supports a dedicated headphone amplifier and is powered via the Raspberry Pi through the GPIO header.

Pinouts

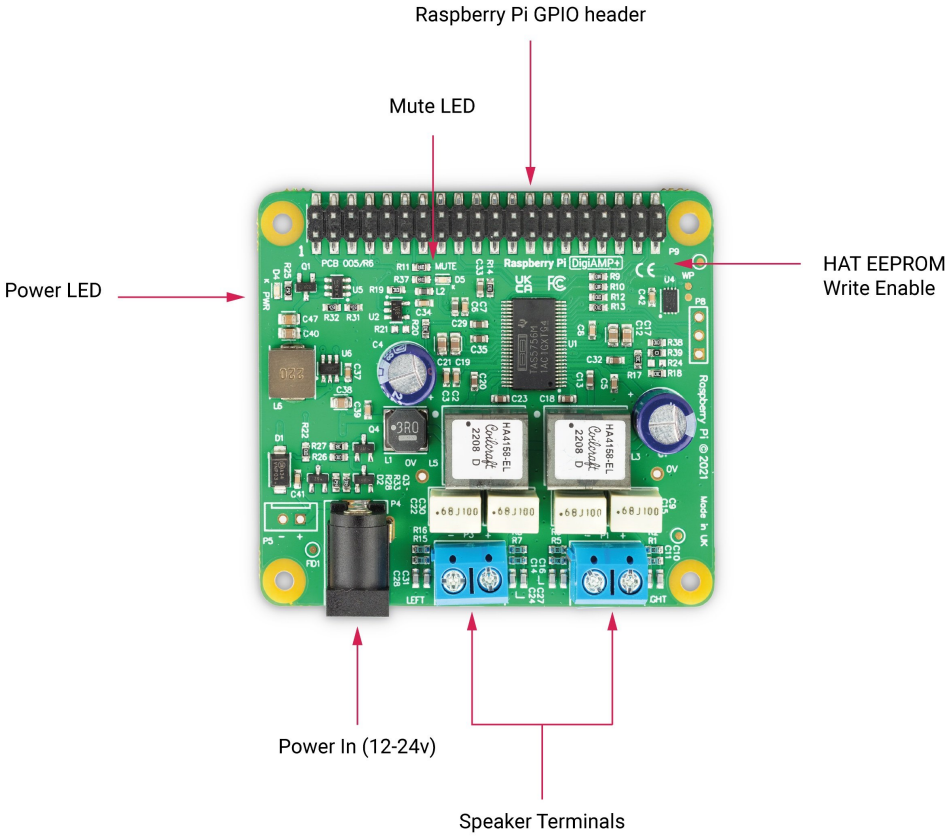
P1	Analogue out (0-2V RMS), carries GPIO27, MUTE signal (headphone detect), left and right audio and left and right ground.
P6	Headphone socket signals (pin1: LEFT, 2:GROUND, 3: RIGHT, 4:GROUND, 5:DETECT).

Raspberry Pi DigiAMP+

Edit this on GitHub

With Raspberry Pi DigiAMP+, you can connect 2 passive stereo speakers up to 35W with variable output, making it ideal for use in Raspberry Pi-based hi-fi systems.

DigiAMP+ uses the Texas Instruments TAS5756M PowerDAC and must be powered from an external supply. It requires a 12-24V DC power source (the XP Power VEC65US19 power supply is recommended).



DigiAMP+'s power in barrel connector is 5.5mm x 2.5mm.

At power-on, the amplifier is muted by default (the mute LED is illuminated). Software is responsible for the mute state and LED control (Raspberry Pi GPIO22).

DigiAMP+ is designed to provide power to the Raspberry Pi and DigiAMP+ together in parallel, delivering 5.1V at 2.5amp to the Raspberry Pi through the GPIO header.

WARNING

> Do not apply power to the Raspberry Pi's own power input when using DigiAMP+.

Pinouts

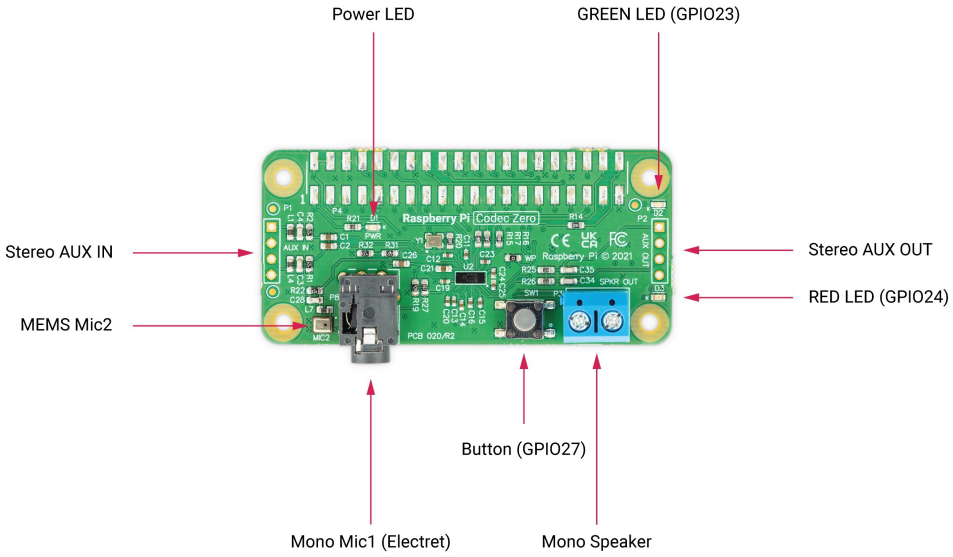
P5	Alternative power input for hard wired installations (polarity must be observed).
P8	TAS5756m Internal GPIO1/2/3

Raspberry Pi Codec Zero

Edit this on GitHub

Raspberry Pi Codec Zero is a Raspberry Pi Zero-sized audio HAT. It delivers bi-directional digital audio signals (I2S) between a Raspberry Pi and the Codec Zero's on-board Dialog Semiconductor DA7212 codec. The Codec Zero supports a range of input and output devices.

- High performance 24-bit audio codec
- Supports common audio sample rates between 8-96kHz
- Built in micro-electro-mechanical (MEMS) microphone (Mic2)
- Mono electret microphone (Mic2 left)
- Automatic MEMS disabling on Mic2 insert detect
- Supports additional (no fit) mono electret microphone (Mic1 right)
- Stereo auxiliary input channel (AUX IN) - PHONO/RCA connectors
- Stereo auxiliary output channel (Headphone/AUX OUT)
- Flexible analogue and digital mixing paths
- Digital signal processors (DSP) for automatic level control (ALC)
- Five-band EQ
- Mono line-out/mini speaker driver: 1.2W @ 5V, THD<10%, R=8Ω

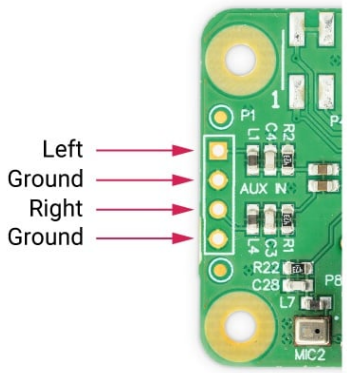


The Codec Zero includes an EEPROM which can be used for auto-configuration of the Linux environment if necessary. It has an integrated MEMS microphone, and can be used with stereo microphone input via a 3.5mm socket and a mono speaker (1.2W/8Ω).

In addition to the green (GPIO23) and red (GPIO24) LEDs, a tactile programmable button (GPIO27) is also provided.

Pinouts

P1/2	Support external PHONO/RCA sockets if needed. P1: AUX IN, P2: AUX OUT.
P1	Pin 1 is square.



Codec Zero is an ideal design starting point for small-scale projects such as walkie-talkies, smart doorbells, vintage radio hacks, or smart speakers.

Configuration

Edit this on GitHub

A pre-programmed EEPROM is included on all Raspberry Pi audio boards. Raspberry Pi audio boards are designed to be plug-and-play; Raspberry Pi OS is able to automatically detect and configure itself. In Raspberry Pi OS, right-clicking on the audio settings in the top right-hand corner of your screen will allow you to switch between the on-board audio settings and the HAT audio settings:

There are a number of third-party audio software applications available for Raspberry Pi that will support the plug-and-play feature of our audio boards. Often these are used headless. They can be controlled via a PC or Mac application, or by a web server installed on Raspberry Pi, with interaction through a webpage.

If you need to configure Raspberry Pi OS yourself, perhaps if you're running a headless system of your own and don't have the option of control via the GUI, you will need to make your Raspberry Pi audio board the primary audio device in Raspberry Pi OS, disabling the Raspberry Pi's on-board audio device. This is done by editing the `/boot/firmware/config.txt` file. Using a Terminal session connected to your Raspberry Pi via SSH, run the following command to edit the file:

```
$ sudo nano /boot/firmware/config.txt
```

Find the **`dtparam=audio=on`** line in the file and comment it out by placing a # symbol at the start of the line. Anything written after the # symbol in any given line will be disregarded by the program. Your **`/boot/firmware/config.txt`** file should now contain the following entry:

```
#dtparam=audio=on
```

Press CTRL+X, then Y and Enter to save, followed by a reboot of your Raspberry Pi in order for the settings to take effect.

```
$ sudo reboot
```

Alternatively, the `/boot/firmware/config.txt` file can be edited directly onto the Raspberry Pi's microSD card, inserted into your usual computer. Using the default file manager, open the `/boot/firmware/` volume on the card and edit the `config.txt` file using an appropriate text editor, then save the file, eject the microSD card and reinsert it back into your Raspberry Pi.

Attaching the HAT

The Raspberry Pi audio boards attach to the Raspberry Pi's 40-pin header. They are designed to be supported on the Raspberry Pi using the supplied circuit board standoffs and screws. No soldering is required on the Raspberry Pi audio boards for normal operation unless you are using hardwired connections for specific connectors such as XLR (External Line Return) connections on the DAC Pro.

All the necessary mounting hardware including spacers, screws and connectors is provided. The PCB spacers should be screwed, finger-tight only, to the Raspberry Pi before adding the audio board. The remaining screws should then be screwed into the spacers from above.

Hardware versions

There are multiple versions of the audio cards, and the version that you possess determines the actions required to configure it. Older IQaudIO-marked boards (black PCB) are electrically equivalent to the Raspberry Pi-branded boards (green PCB) but have different EEPROM contents. The following command can be used to confirm which version you have:

```
$ grep -a . /proc/device-tree/hat/*
```

If the vendor string says "Raspberry Pi Ltd." then no further action is needed (but see below for the extra Codec Zero configuration). If it says "IQaudIO Limited www.iqaudio.com" then you will need the additional config.txt settings outlined below. If it says "No such file or directory" then the HAT is not being detected, but these config.txt settings may still make it work.

```
# Some magic to prevent the normal HAT overlay from being loaded
dtoverlay=
# And then choose one of the following, according to the model:
dtoverlay=rpi-codeczero
dtoverlay=rpi-dacplus
dtoverlay=rpi-dacpro
dtoverlay=rpi-digiampplus
```

Extra Codec Zero configuration

The Raspberry Pi Codec Zero board uses the Dialog Semiconductor DA7212 codec. This allows the recording of audio from the built-in MEMS microphone, from stereo phono sockets (AUX IN) or two mono external electret microphones. Playback is through stereo phono sockets (AUX OUT) or a mono speaker connector.

Each input and output device has its own mixer, allowing the audio levels and volume to be adjusted independently. Within the codec itself, other mixers and switches exist to allow the output to be mixed to a single mono channel for single-speaker output. Signals may also be inverted; there is a five-band equaliser to adjust certain frequency bands. These settings can be controlled interactively, using AlsaMixer, or programmatically.

Both the AUX IN and AUX OUT are 1V RMS. It may be necessary to adjust the AUX IN's mixer to ensure that the input signal doesn't saturate the ADC. Similarly, the output mixers can be to be adjusted to get the best possible output.

Preconfigured scripts (loadable ALSA settings) are available on GitHub, offering:

- Mono MEMS mic recording, mono speaker playback
- Mono MEMS mic recording, mono AUX OUT playback
- Stereo AUX IN recording, stereo AUX OUT playback
- Stereo MIC1/MIC2 recording, stereo AUX OUT playback

The Codec Zero needs to know which of these input and output settings are being used each time the Raspberry Pi powers on. Using a Terminal session on your Raspberry Pi, run the following command to download the scripts:

```
$ git clone https://github.com/raspberrypi/Pi-Codec.git
```

If git is not installed, run the following command to install it:

```
$ sudo apt install git
```

The following command will set your device to use the on-board MEMS microphone and output for speaker playback:

```
$ sudo alsactl restore -f /home/pi/Pi-Codec/Codec_Zero_OnboardMIC_record
_and_SPK_playback.state
```

In order for your project to operate with your required settings when it is powered on, edit the `/etc/rc.local` file. The contents of this file are run at the end of every boot process, so it is ideal for this purpose. Edit the file:

```
$ sudo nano /etc/rc.local
```

Add the chosen script command above the exit 0 line and then Ctrl X, Y and Enter to save. The file should now look similar to this depending on your chosen setting:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

sudo alsactl restore -f /home/pi/Pi-Codec/Codec_Zero_OnboardMIC_record_a
nd_SPK_playback.state

exit 0
```

Ctrl X, Y and Enter to save and reboot your device for the settings to take effect:

```
$ sudo reboot
```

If you are using your Raspberry Pi and Codec Zero in a headless environment, there is one final step required to make the Codec Zero the default audio device without access to the GUI audio settings on the desktop. We need to create a small file in your home folder:

```
$ sudo nano .asoundrc
```

Add the following to the file:

```
pcm.!default {
        type hw
        card Zero
}
```

Ctrl X, Y and Enter to save, and reboot once more to complete the configuration:

```
$ sudo reboot
```

Muting and unmuting the DigiAMP+

The DigiAMP+ mute state is toggled by GPIO22 on Raspberry Pi. The latest audio device tree supports the unmute of the DigiAMP+ through additional parameters.

Firstly a "one-shot" unmute when kernel module loads.

For Raspberry Pi boards:

```
dtoverlay=rpi-digiampplus,unmute_amp
```

For IQaudIO boards:

```
dtoverlay=iqaudio-digiampplus,unmute_amp
```

Unmute the amp when an ALSA device is opened by a client. Mute, with a five-second delay when the ALSA device is closed. (Reopening the device within the five-second close window will cancel mute.)

For Raspberry Pi boards:

```
dtoverlay=rpi-digiampplus,auto_mute_amp
```

For IQaudIO boards:

```
dtoverlay=iqaudio-digiampplus,auto_mute_amp
```

If you do not want to control the mute state through the device tree, you can also script your own solution.

The amp will start up muted. To unmute the amp:

```
$ sudo sh -c "echo 22 > /sys/class/gpio/export"
$ sudo sh -c "echo out >/sys/class/gpio/gpio22/direction"
$ sudo sh -c "echo 1 >/sys/class/gpio/gpio22/value"
```

to mute the amp once more:

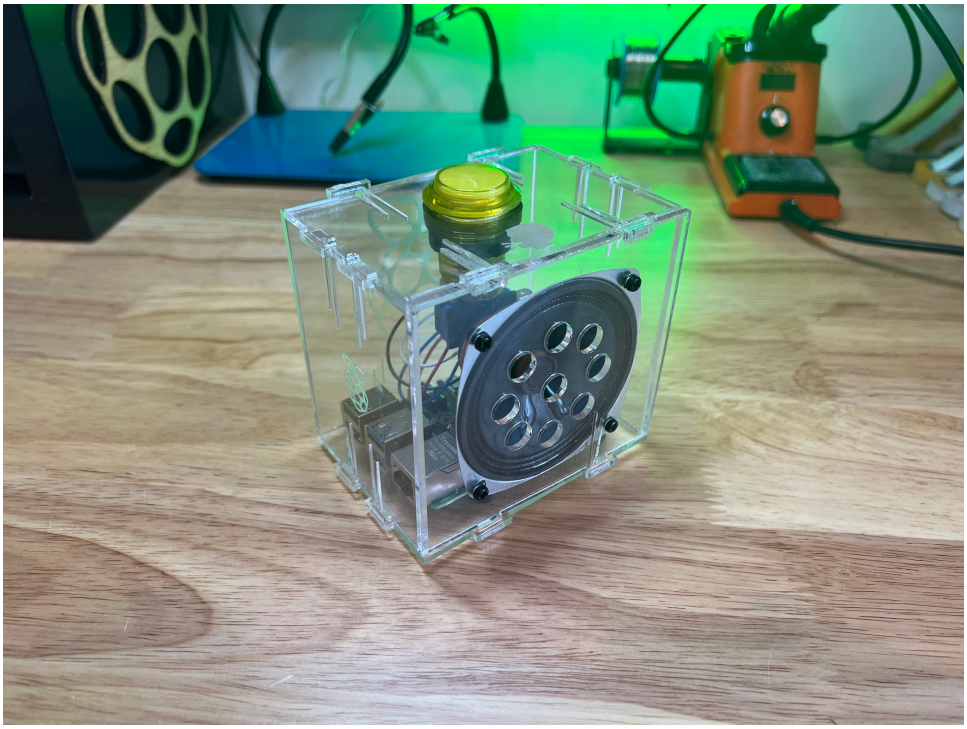
```
$ sudo sh -c "echo 0 >/sys/class/gpio/gpio22/value"
```

Getting started

Edit this on GitHub

Creating a toy chatter box

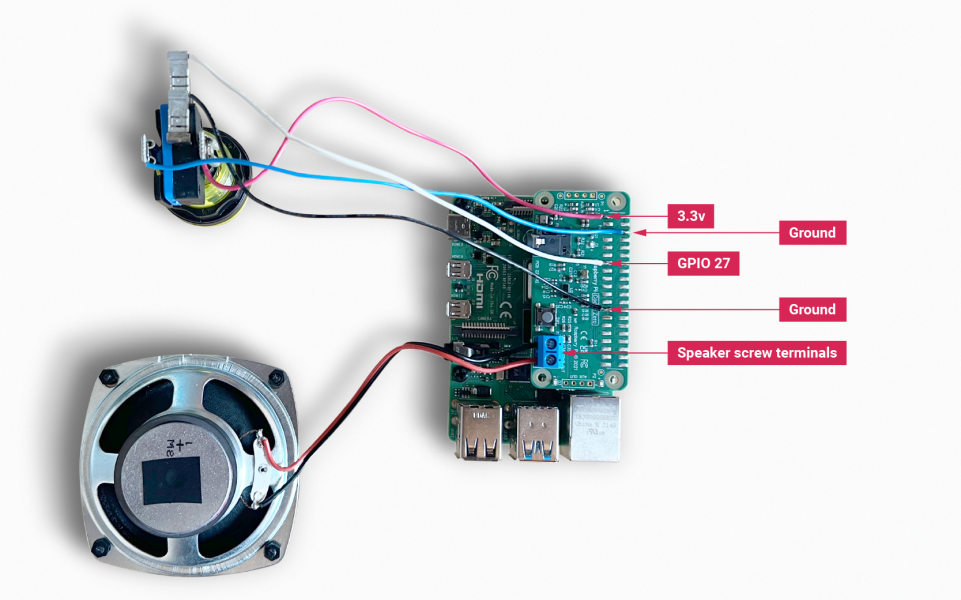
As an example of what Raspberry Pi Audio Boards can do, let's walk through the creation of a toy chatter box. Its on-board microphone, programmable button and speaker driver make the Codec Zero an ideal choice for this application.



A random pre-recorded five-second audio clip will be played when the button is pressed. After holding for ten seconds, a notifying burp sound will be emitted, after which a new five-second clip will be recorded. Holding the button down for more than 20 seconds will play a second burp sound, and then erase all previous recordings.

Hardware and wiring

For this project, any small passive speaker should be sufficient. We're using one available here, which handles 5W of power at 4Ω. We have also used an illuminated momentary push button, and a laser-cut box to house all the components; but both are entirely optional. This example will work just using the Codec Zero's on-board button, which is pre-wired to GPIO 27. (Alternatively, you can use any momentary push button, such as those available here.)



Use a small flat-head screwdriver to attach your speaker to the screw terminals. For the additional push button, solder the button wires directly to the Codec Zero pads as indicated, using GPIO pin 27 and Ground for the switch, and +3.3V and Ground for the LED, if necessary.

Setting up your Raspberry Pi

In this example, we are using Raspberry Pi OS Lite. Refer to our guide on installing Raspberry Pi OS for more details.

Make sure that you update your operating system before proceeding and follow the instructions provided for Codec Zero configuration, including the commands to enable the on-board microphone and speaker output.

Programming your Raspberry Pi

Open a shell — for instance by connecting via SSH — on your Raspberry Pi and run the following to create our Python script:

```
$ sudo nano chatter_box.py
```

Adding the following to the file:

```
#!/usr/bin/env python3
from gpiozero import Button
from signal import pause
import time
import random
import os
from datetime import datetime
```

```
# Print current date

date = datetime.now().strftime("%d_%m_%Y-%H:%M:%S")
print(f"{date}")

# Make sure that the 'sounds' folder exists, and if it does not, create
it

path = '/home/pi/sounds'

isExist = os.path.exists(path)

if not isExist:
  os.makedirs(path)
  print("The new directory is created!")
  os.system('chmod 777 -R /home/pi/sounds')

# Download a 'burp' sound if it does not already exist

burp = '/home/pi/burp.wav'

isExist = os.path.exists(burp)
if not isExist:
  os.system('wget http://rpf.io/burp -O burp.wav')
  print("Burp sound downloaded!")

# Setup button functions - Pin 27 = Button hold time 10 seconds.

button = Button(27, hold_time=10)

def pressed():
    global press_time
    press_time = time.time()
    print("Pressed at %s" % (press_time));

def released():
    release_time = time.time()
    pressed_for = release_time - press_time
    print("Released at %s after %.2f seconds" % (release_time, pressed_f
or))
    if pressed_for < button.hold_time:
        print("This is a short press")
        randomfile = random.choice(os.listdir("/home/pi/sounds/"))
        file = '/home/pi/sounds/' + randomfile
        os.system('aplay ' + file)
    elif pressed_for > 20:
        os.system('aplay ' + burp)
        print("Erasing all recorded sounds")
        os.system('rm /home/pi/sounds/*');

def held():
    print("This is a long press")
    os.system('aplay ' + burp)
    os.system('arecord --format S16_LE --duration=5 --rate 48000 -c2 /ho
me/pi/sounds/$(date +"%d_%m_%Y-%H_%M_%S")_voice.m4a');

button.when_pressed = pressed
button.when_released = released
button.when_held = held

pause()
```

Ctrl X, Y and Enter to save. To make the script executable, type the following:

```
$ sudo chmod +x chatter_box.py
```

Enter the following to create a crontab daemon that will automatically start the script each time the device is powered on:

```
$ crontab -e
```

You will be asked to select an editor; we recommend you use `nano`. Select it by entering the corresponding number, and press Enter to continue. The following line should be added to the bottom of the file:

```
@reboot python /home/pi/chatter_box.py
```

Ctrl X, Y and Enter to save, then reboot your device.

Operating your device

The final step is to ensure that everything is operating as expected. Press the button and release it when you hear the burp. The recording will now begin for a period of five seconds. Once you have released the button, press it briefly again to hear the recording. Repeat this process as many times as you wish, and your sounds will be played at random. You can delete all recordings by pressing and holding the button, keeping the button pressed during the first burp and recording process, and releasing it after at least 20 seconds, at which point you will hear another burp sound confirming that the recordings have been deleted.

Next steps

Upgrades! It is always fun to upgrade a project, so why not add some additional features, such as an LED that will illuminate when recording? This project has all the parts required to make your own version of a Google intelligent speaker system, or you may want to consider building a second device that can be used to create a pair of walkie-talkies that are capable of transferring audio files over a network via SSH.

Hardware information

Edit this on GitHub

Hardware information:

- PCB screws are all M2.5.
- PCB standoffs (for case) are 5mm male/female.
- PCB standoffs (for Raspberry Pi to audio boards) are 9mm female/female.
- PCB standoffs (for XLR to DAC PRO) are 8mm female/male.
- PCB standoffs (for the official Raspberry Pi 7-inch display) are 5mm male/female.
- The rotary encoders we have used and tested are the Alpha three-pin rotary encoder RE160F-40E3-20A-24P, the ALPS EC12E2430804 (RS: 729-5848), and the Bourns ECW0JB24-AC0006L (RS: 263-2839).
- The barrel connector used for powering the DigiAMP+ is 2.5mmID, 5.5mmOD, 11mm.
- The DigiAMP+ is designed to operate with a 12V to 24V, 3A supply such as the XPPower VEC65US19 or similar.
- The DigiAMP+ uses CamdenBoss two-part connectors. Those fitted to the PCB are CTBP9350/2AO.
- The speaker terminal used on the Codec Zero will accept wires of between 14~26 AWG (wire of max 1.6mm in diameter).

GPIO usage

Raspberry Pi audio boards take advantage of a number of pins on the GPIO header in order to operate successfully. Some of these pins are solely for the use of the board, and some can be shared with other peripherals, sensors, etc.

The following Raspberry Pi GPIO pins will be used by the audio boards:

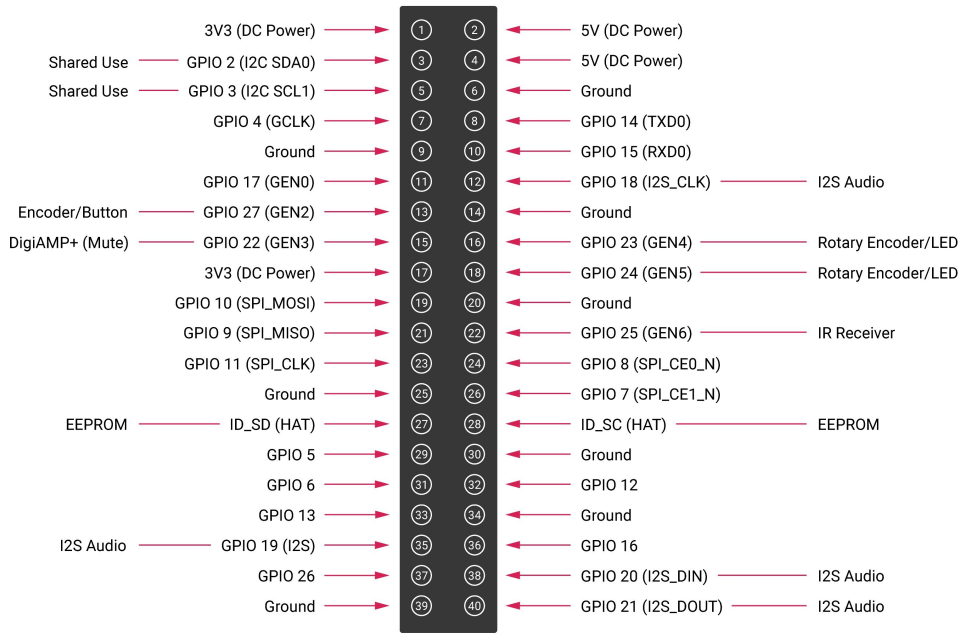
- All power pins

- All ground pins
- GPIO 2/3 (I2C)
- GPIO 18/19/20/21 (I2S)

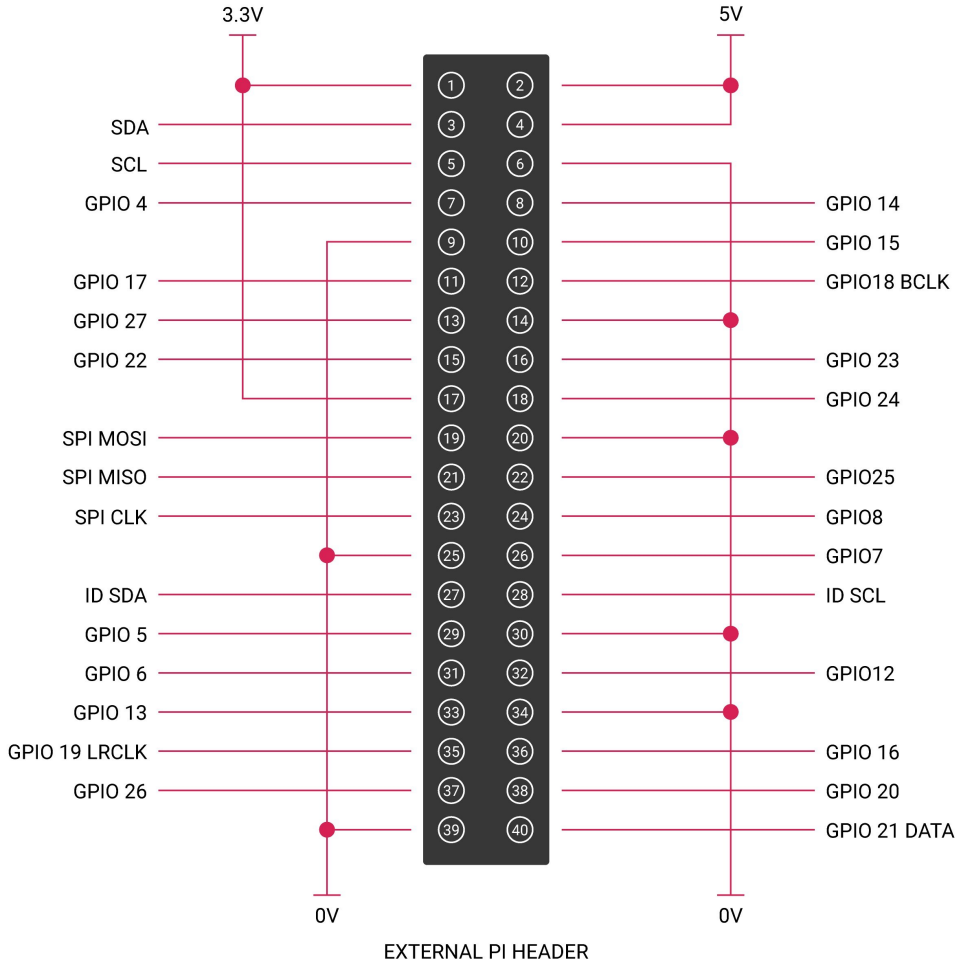
If appropriate then the following are also used:

- GPIO 22 (DigiAMP+ mute/unmute support)
- GPIO 23/24 for rotary encoder (physical volume control) or status LED (Codec Zero)
- GPIO 25 for the IR Sensor
- GPIO 27 for the rotary encoder push switch/Codec Zero switch

DAC PRO, DAC+, DigiAMP+, Codec Zero



The DAC PRO, DAC+ and DigiAMP+ re-expose the Raspberry Pi signals, allowing additional sensors and peripherals to be added easily. Please note that some signals are for exclusive use (I2S and EEPROM) by some of our boards; others such as I2C can be shared across multiple boards.



Saving AlsaMixer settings

To store the AlsaMixer settings, add the following at the command line:

```
$ sudo alsactl store
```

You can save the current state to a file, then reload that state at startup.

To save:

```
$ sudo alsactl store -f /home/pi/usecase.state
```

To restore a saved file:

```
$ sudo alsactl restore -f /home/pi/usecase.state
```

MPD-based audio with volume control

To allow Music Player Daemon (MPD)-based audio software to control the audio board's built in volume, the file `/etc/mpd.conf` may need to be changed to support the correct AlsaMixer name.

This can be achieved by ensuring the 'Audio output' section of `/etc/mpd.conf` has the 'mixer_control' line. Below is an example for the Texas Instruments-based boards (DAC PRO/DAC+/DigiAMP+):

```
audio_output {
    type "alsa"
    name "ALSA Device"
    mixer_control "Digital"
}
```

Updating your firmware

Edit this on GitHub

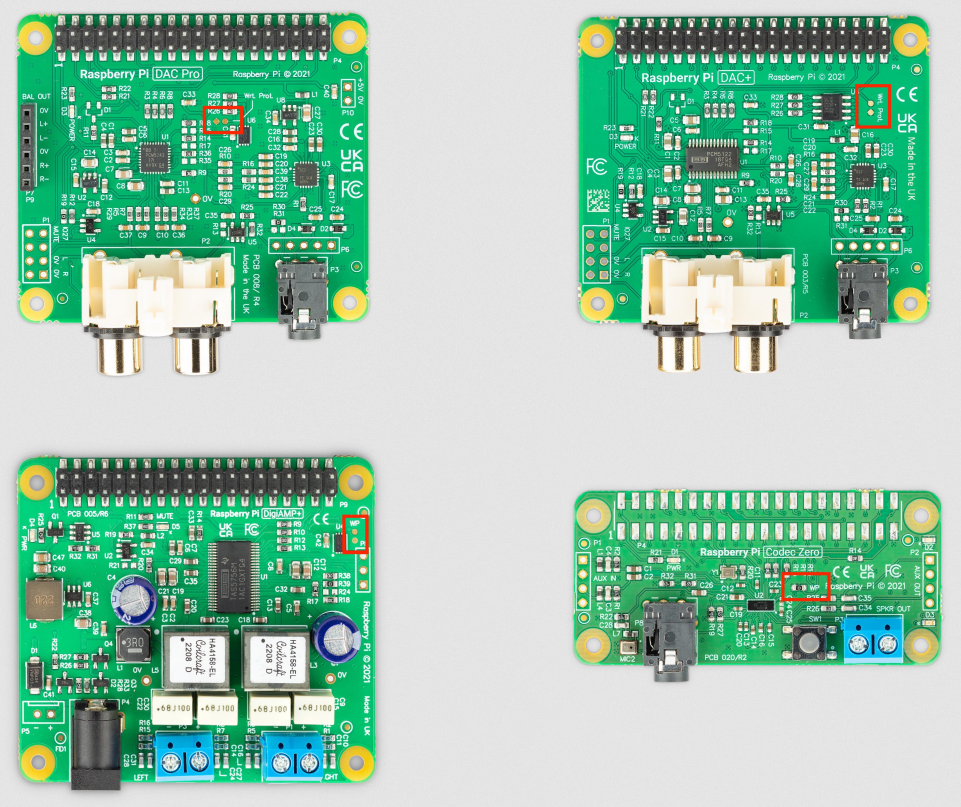
Raspberry Pi Audio Boards use an EEPROM that contains information that is used by the host Raspberry Pi device to select the appropriate driver at boot time. This information is programmed into the EEPROM during manufacture. There are some circumstances where the end user may wish to update the EEPROM contents: this can be done from the command line.

IMPORTANT

Before proceeding, you should update the Raspberry Pi OS running on your Raspberry Pi to the latest version.

The EEPROM write-protect link

During the programming process you will need to connect the two pads shown in the red box with a wire to pull down the EEPROM write-protect link.



NOTE

In some cases the two pads may already have a 0Ω resistor fitted to bridge the write-protect link, as illustrated in the picture of the Codec Zero board above.

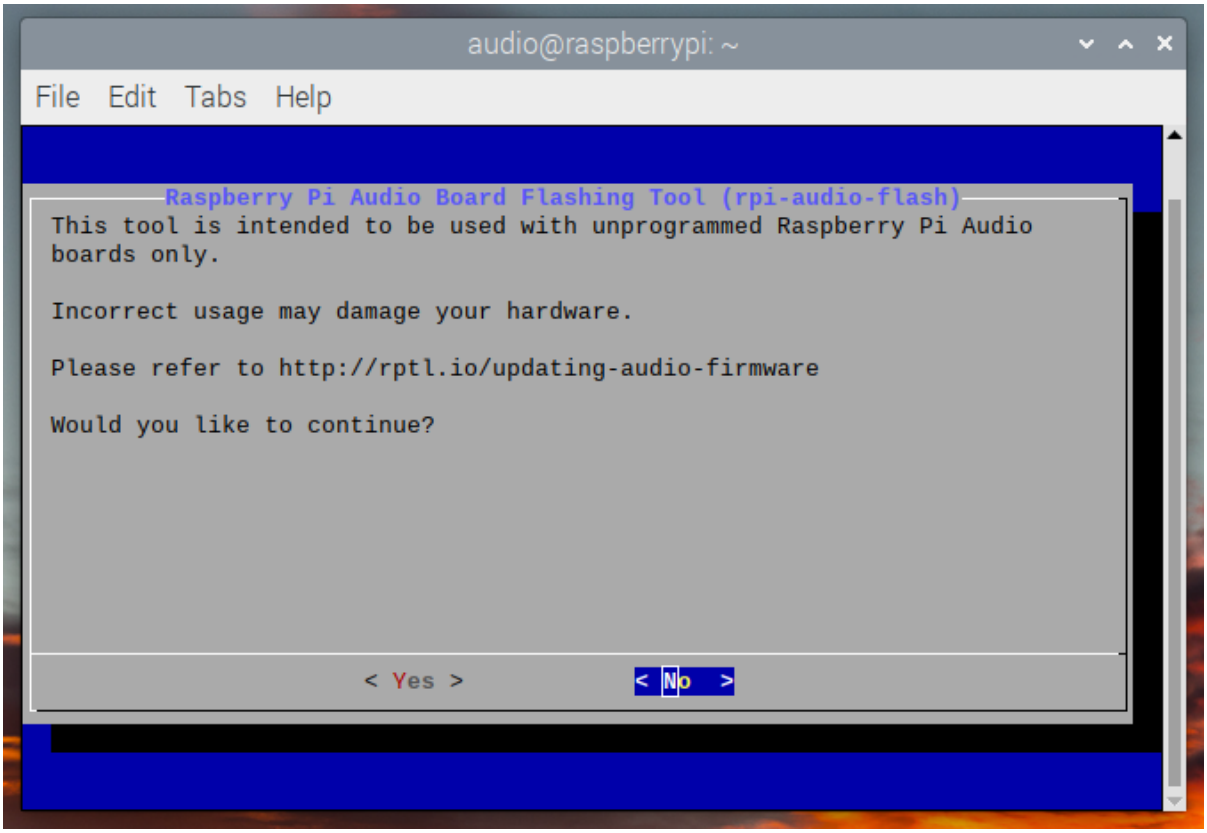
EEPROM Programming

Once the write-protect line has been pulled down, the EEPROM can be programmed.

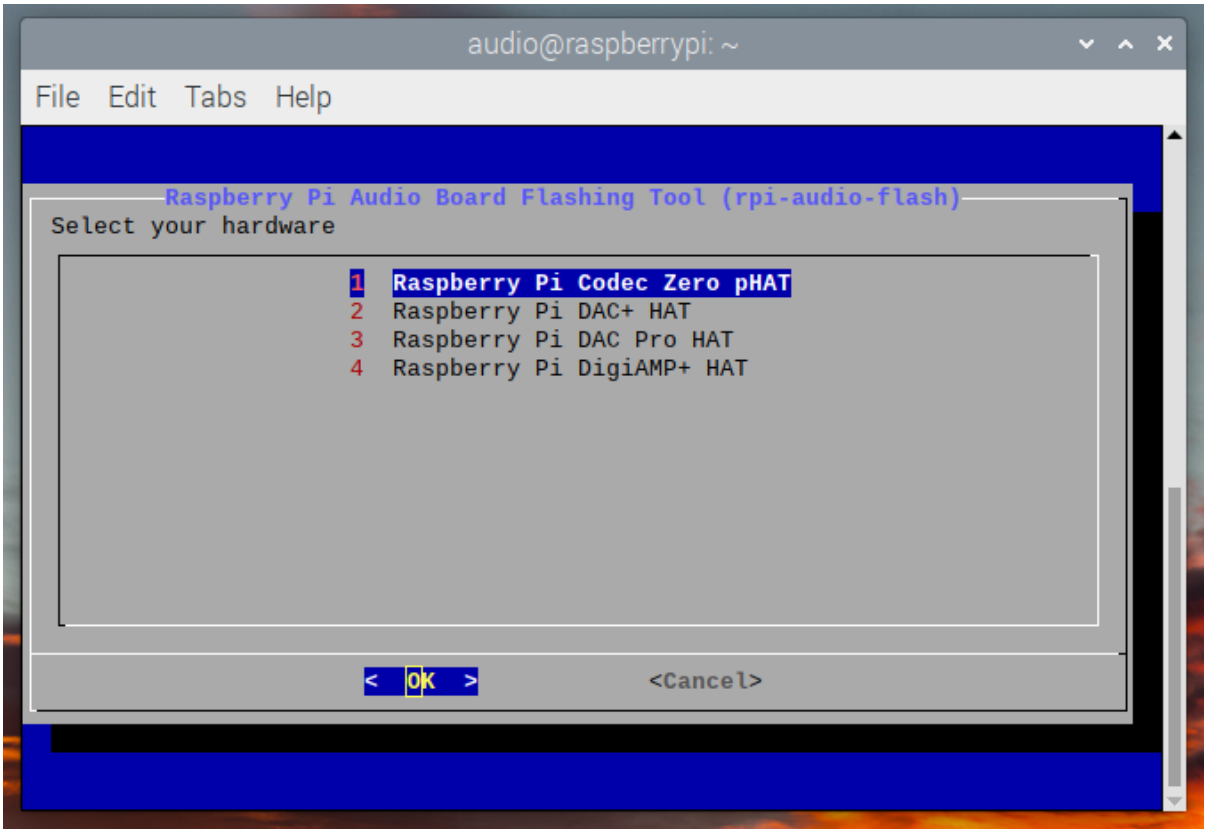
You should first install the utilities and then run the programmer. Open up a terminal window and type the following:

```
$ sudo apt update
$ sudo apt install rpi-audio-utils
$ sudo rpi-audio-flash
```

After starting you will be presented with a warning screen.



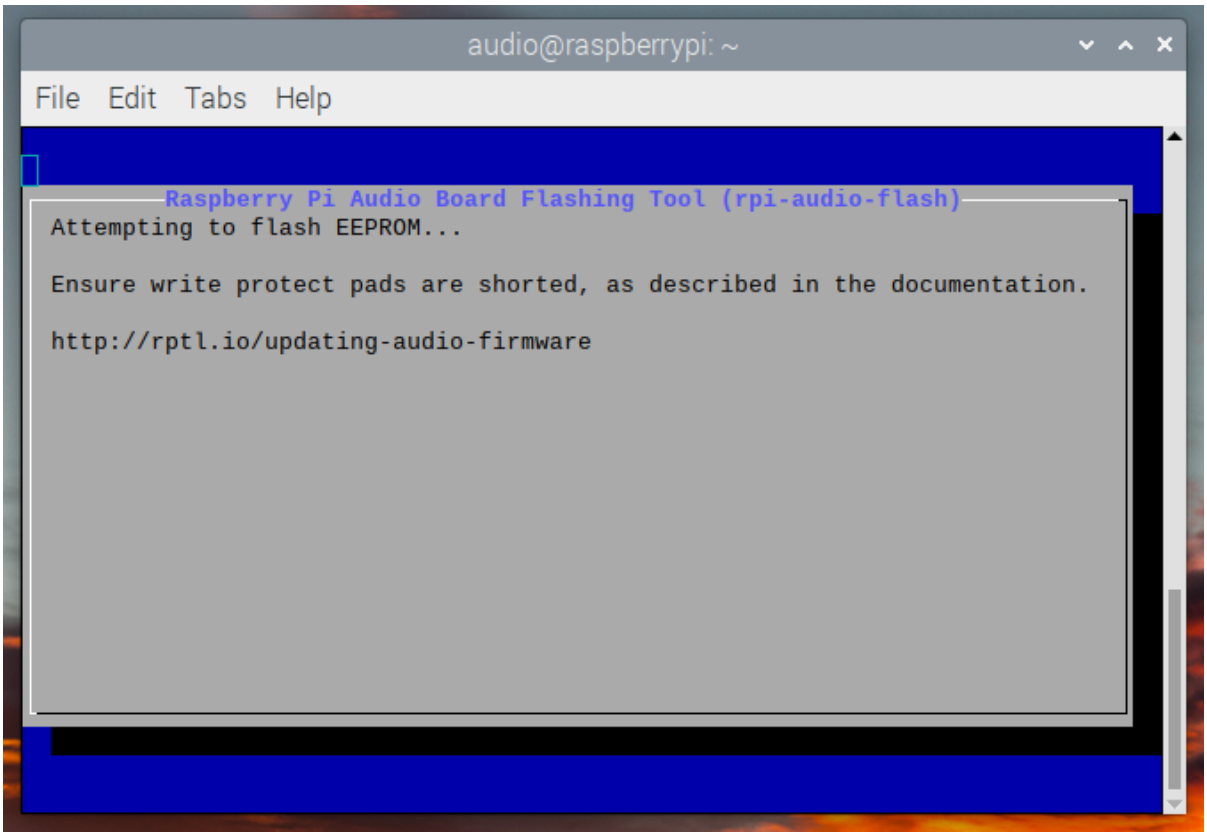
Selecting "Yes" to proceed will present you with a menu allowing you to select your hardware.



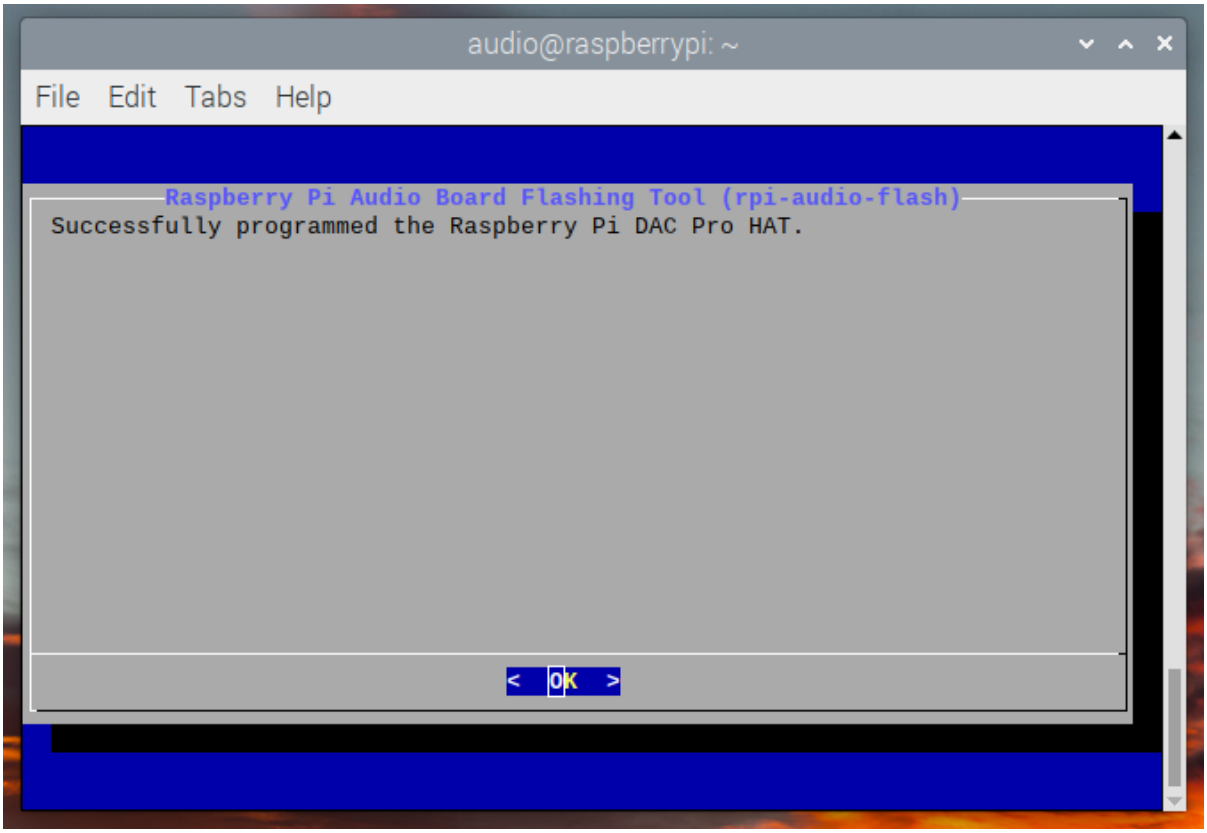
NOTE

If no HAT is present, or if the connected HAT is not a Raspberry Pi Audio board, you will be presented with an error screen. If the firmware has already been updated on the board, a message will be displayed informing you that you do not have to continue.

After selecting the correct hardware a screen will display while the new firmware is flashed to the HAT.



Afterwards a screen will display telling you that the new firmware has installed.



NOTE

If the firmware fails to install correctly, an error screen will be displayed. In the first instance you should remove and reseat the HAT board and try flashing the firmware again.