# Table of Contents

**Overview**

**Pinouts**
- Power Pins
- Audio Input
- Audio Output
- STEMMA Connector
- Terminal Block
- Volume
- Power LED and Jumper

**CircuitPython**
- Wiring
- Example Code
- Code Configuration
- Code Walkthrough

**CircuitPython Docs**

**Arduino**
- Wiring
- Example Code

**Arduino Docs**

**Downloads**
- Files
- Schematic and Fab Print
Overview

The Adafruit STEMMA Audio Amp is a super small mono amplifier is surprisingly powerful - able to deliver up to 2.5 Watts into 4-8 ohm impedance speakers. Inside the miniature chip is a class D controller, able to run from 2.0V-5.5VDC. Since the amp is a class D, it's very efficient (over 90% efficient when driving an 8Ω speaker at over half a Watt) - making it perfect for portable and battery-powered projects. It has built in thermal and over-current protection but we could barely tell it got hot. There's even a volume trim pot so you can adjust the volume on the board down from the default 24dB gain. This board is a welcome upgrade to basic "LM386" amps!
This board is a simplified version of our breadboard-friendly Adafruit Mono 2.5W Class D Audio Amplifier board. This version is designed to be ‘plug and play’ - with no soldering required. You can input the audio via the JST-PH 3 Pin connector - such as a pin header cable or one with alligator clips. The trade-off is that this board doesn't have differential inputs, instead the audio inputs are referenced to the power/signal ground wire.

The input to the amplifier goes through 1.0uF capacitors so you can have DC. The output is "Bridge Tied" - that means the output pins connect directly to the speaker pins, no connection to ground. The output is a high frequency 250KHz square wave PWM that is then 'averaged out' by the speaker coil - the high frequencies are not heard. All the above means that you can't connect the output into another amplifier, it should drive the speakers directly.
The board comes fully assembled and tested. We also include header to plug it into a breadboard and a 3.5mm screw-terminal blocks so you can easily attach/detach your speaker. You will be ready to rock in 15 minutes! Speaker is not included, use any 4 ohm or greater impedance speaker.

- Output Power: 2.5W at 4Ω, 10% THD, 1.5W at 8Ω, 10% THD, with 5.5V Supply
- 50dB PSRR at 1KHz
- Filterless design, with ferrite bead + capacitors on output.
- Fixed 24dB gain, onboard trim potentiometer for adjusting input volume.
- Thermal and short-circuit/over-current protection.
- Low current draw: 4mA quiescent and 0.5mA in shutdown (due to pullup resistor on SD pin)

Note: The terminal block included with your product may be blue or black.

---

**Pinouts**
Power Pins

These pins are available both through the STEMMA connector, and as through-hole pads.

- VIN - This is the power pin. To power the board, provide 3V-5VDC.
- GND - This is common ground for power.

Audio Input

This pin is available both through the STEMMA connector, and as a through-hole pad.

- SIGNAL - This is the audio input pin. Connect to a line level (~1Vpp) voltage audio signal to send audio to the amp. There is capacitive AC coupling so you can have a DC offset.

Audio Output

- VO+ - Connect the power wire on a compatible 4-8Ω speaker to this pin.
- VO- - Connect the ground wire on a compatible 4-8Ω speaker to this pin.

Bridge-Tied Load outputs mean you cannot connect these to another amplifier - they must connect to a speaker directly!

STEMMA Connector

The STEMMA connector is located on the left side of the front of the board, below the PAM8302 on the board silk. You can use a STEMMA JST PH 2mm 3-pin cable to connect the amp to your microcontroller board, for example [this cable](https://example.com) to connect using a breadboard.

The STEMMA connector has the following pins:

- SIGNAL (white wire) - Audio input pin - line level AC or DC coupled input
- VIN (red wire) - Power pin, 3-5VDC
- GND (black wire) - Ground pin.
Terminal Block

The terminal block is located on the right side of the front of the board, between the + and - on the board silk. Your terminal block may be blue or black.

The block has the following terminals:

- - - This terminal is towards the top of the board, and is located next to the - symbol on the board silk. It is the same as the VO- pin. Connect the ground wire on your speaker to this terminal.
+ - This terminal is towards the bottom of the board and is located next to the + symbol on the board silk. It is the same as the VO+ pin. Connect the power wire on your speaker to this terminal.

Volume

The silver trimpot sitting top-center on the front of the board, labeled Volume, allows you to adjust the volume on the board down from the default 24dB gain.

Power LED and Jumper

- Power LED - On the front of the board, below the STEMMA connector on the left, is the power LED, labeled ON. It is a green LED.
- LED jumper - In the lower right corner on the back of the board is a jumper for the power LED. If you wish to disable the power LED, simply cut the trace on this jumper. To enable it again, solder the pads back together.

CircuitPython

It's easy to use the STEMMA Audio Amp with CircuitPython and the built-in audiopwmio module. This module allows you to play a simple tune through the speaker connected to the amp.

This page uses the Feather RP2040 to demonstrate.

Wiring

Connect up the amp exactly as follows.
This diagram shows wiring it up using the STEMMA connector and the speaker terminal block.

Amp to Feather:

Feather A0 to STEMMA Signal (white wire)
Feather 3.3V to STEMMA VIN (red wire)
Feather GND to STEMMA GND (black wire)

Speaker to amp:

- side of terminal to speaker - (black wire)
+ side of terminal to speaker + (red wire)

This diagram shows wiring it up using a breadboard.

Feather to amp:

Feather GND to breakout GND (black wire)
Feather 3.3V to breakout VIN (red wire)
Feather A0 to breakout Signal (white wire)

Speaker to amp:

Speaker - to breakout VO- (black wire)
Speaker + to breakout VO+ (red wire)

Example Code

To use with CircuitPython, you simply need to update code.py on your CIRCUITPY drive with the example code. No external libraries are needed.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download code.py file in a zip file. Extract the contents of the zip file, and copy code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the following.
```python
import time
import array
import math
import board
from audiocore import RawSample
from audiopwmio import PWMAudioOut as AudioOut

tone_volume = 0.1
# The tones are provided as a frequency in Hz. You can change the current tones or add your own to make a new tune. Follow the format with commas between values.
tone_frequency = [784, 880, 698, 349, 523]

audio = AudioOut(board.A0)
while True:
    # Play each tone in succession.
    for frequency in tone_frequency:
        # Compute the sine wave for the current frequency.
        length = 8000 // frequency
        sine_wave = array.array("H", [0] * length)
        for index in range(length):
            sine_wave[index] = int((1 + math.sin(math.pi * 2 * index / length)) * tone_volume * (2 ** 15 - 1))

        sine_wave_sample = RawSample(sine_wave)

        # Play the current frequency.
        audio.play(sine_wave_sample, loop=True)
        time.sleep(0.5)
        audio.stop()
        time.sleep(1)

    # All done playing all tones; start over from the beginning.
```

Once you have copied code.py to your CIRCUITPY drive, you should hear a series of five tones, 0.5 seconds each, with a 1 second delay between. These tones play in a loop. That's all there is to using the STEMMA Audio amp to play a short tune!

### Code Configuration

There are two things you can configure in this demo.

©Adafruit Industries
The first is tone volume. It defaults to 0.1. You can increase this value to increase the volume of the tones.

```
tone_volume = 0.1
```

The second is the tone frequencies. There are five frequencies, in Hz, provided in a specific order. You can change the Hz of any of these values to change one of the five tones. You can also add more values, following the same format with commas between them, to create a longer tune.

```
tone_frequency = [784, 880, 698, 349, 523]
```

### Code Walkthrough

The whole demo put together is conceptually quite short.

Following imports and configuration, you create an `audio` object on pin A0.

```
audio = AudioOut(board.A0)
```

Inside the `while` loop, there is a `for` loop. This nested loop runs once for each tone frequency, and then repeats.

```
while True:
    for frequency in tone_frequency:
```

Inside the `for` loop, you first compute the sine wave for the current frequency.

```
length = 8000 // frequency
sine_wave = array.array("H", [0] * length)
for index in range(length):
    sine_wave[index] = int((1 + math.sin(math.pi * 2 * index / length)) * tone_volume * (2 ** 15 - 1))
sine_wave_sample = RawSample(sine_wave)
```

At the end of the `for` loop, you play the computed sine wave for the current frequency for 0.5 seconds. Then you stop playing for 1 second.

```
audio.play(sine_wave_sample, loop=True)
time.sleep(0.5)
audio.stop()
time.sleep(1)
```
When the code reaches the end of the 1 second silence, it begins the for loop again for the next frequency in the cycle.

CircuitPython Docs

Arduino

Using the Adafruit STEMMA Audio Amp with Arduino is as easy as wiring up the amp to an RP2040 microcontroller, and running the provided example code. No libraries are necessary! This page uses the Feather RP2040 to demonstrate.

Wiring

Connect the audio amp to a Feather RP2040 exactly as shown below.

You can wire it up using the STEMMA port and the speaker terminal block.

Amp to Feather:

- Feather A0 to STEMMA Signal (white wire)
- Feather 3.3V to STEMMA VIN (red wire)
- Feather GND to STEMMA GND (black wire)

Speaker to amp:

- - side of terminal to speaker - (black wire)
- + side of terminal to speaker + (red wire)

Alternatively, you can use a solderless breadboard.
Example Code

// SPDX-FileCopyrightText: 2023 Kattni Rembor for Adafruit Industries
// SPDX-FileCopyrightText: Earle F. Philhower, III
//
// SPDX-License-Identifier: MIT

/*
   This example plays a tune through a mono amplifier using a simple sine wave.

   Released to the public domain by Earle F. Philhower, III
   <earlephilhower@yahoo.com>

   Adapted from stereo original example 2023 by Kattni Rembor
*/

#include <PWMAudio.h>
PWMAudio pwm(0, true); // GP0 = left, GP1 = right
const int freq = 48000; // Output frequency for PWM
int16_t mono = 0;

const int notes[] = { 784, 880, 698, 349, 523 };
const int dly[] =   { 400, 500, 700, 500, 1000 };
const int noteCnt = sizeof(notes) / sizeof(notes[0]);
int freqMono = 1;

double sineTable[128]; // Precompute sine wave in 128 steps
unsigned int cnt = 0;

void cb() {
  while (pwm.availableForWrite()) {
    double now = ((double)cnt) / (double)freq;
    int freqScale = freqMono << 7; // Prescale by 128 to avoid FP math later on
    pwm.write((int16_t)(mono * sineTable[(int)(now * freqScale) & 127]));
    cnt++;
  }
}

void setup() {
  // Set up sine table for waveform generation
  for (int i = 0; i < 128; i++) {
    sineTable[i] = sin(i * 2.0 * 3.14159 / 128.0);
  }
}
void loop() {
  delay(1000);
  mono = 0;
  Serial.println("loop");
  for (int i = 0; i < noteCnt; i++) {
    freqMono = notes[i];
    mono = 5000;
    delay(dly[i]);
  }
  mono = 0;
  delay(3000);
}

Once you've successfully uploaded the sketch to your Feather RP2040, you'll hear a series of tones played through the speaker. That's your short tune!