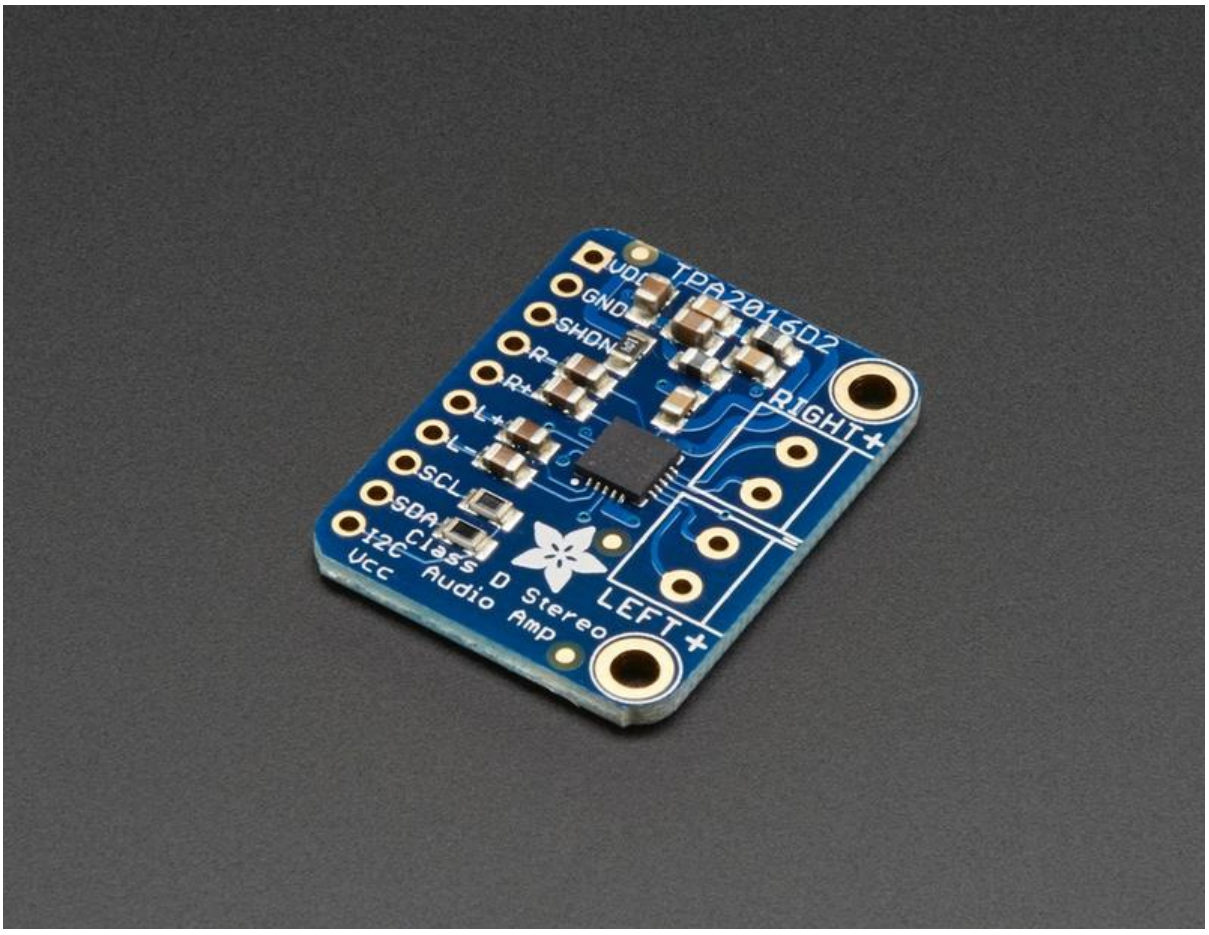
Adafruit TPA2016 2.8W AGC Stereo Audio Amplifier

Created by lady ada

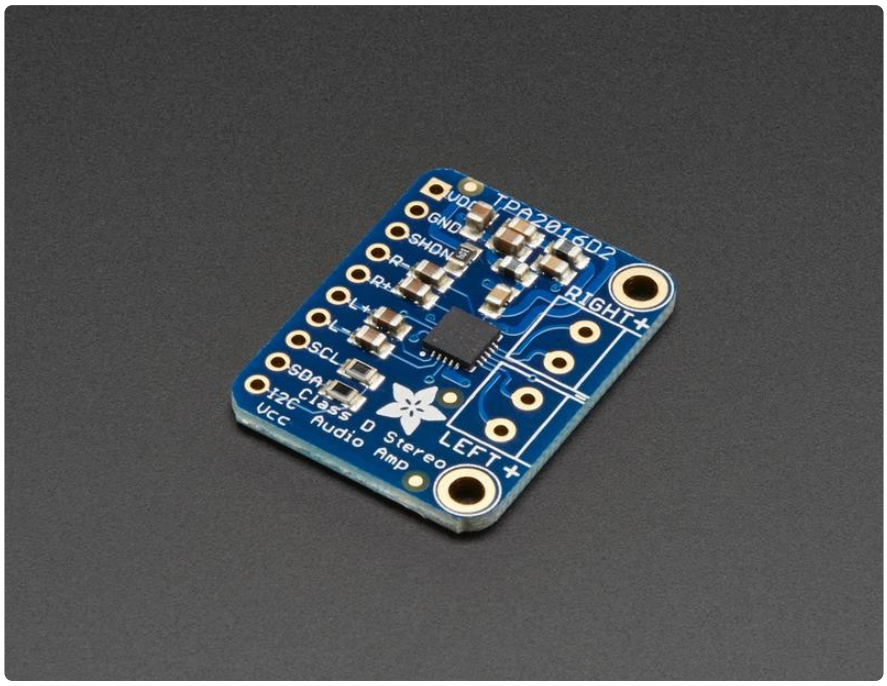


https://learn.adafruit.com/adafruit-tpa2016-2-8w-agc-stereo-audio-amplifier

Last updated on 2022-12-01 02:08:10 PM EST

Table of Contents

Overview



This incredibly small stereo amplifier is surprisingly powerful. It is able to deliver 2 x 2.8W channels into 4 ohm impedance speakers (@ 10% THD) and it has a i2c control interface as well as an AGC (automatic gain control) system to keep your audio from clipping or distorting.

If you don't want to use I2C to control it, it does start up on with 6dB gain by default and the AGC set up for most music playing. We do suggest using it with a microcontroller to configure it, however, since its quite powerful. Settings are not stored in the chip, so you'll need to adjust any gain & AGC amplification settings every time the amp is powered up.

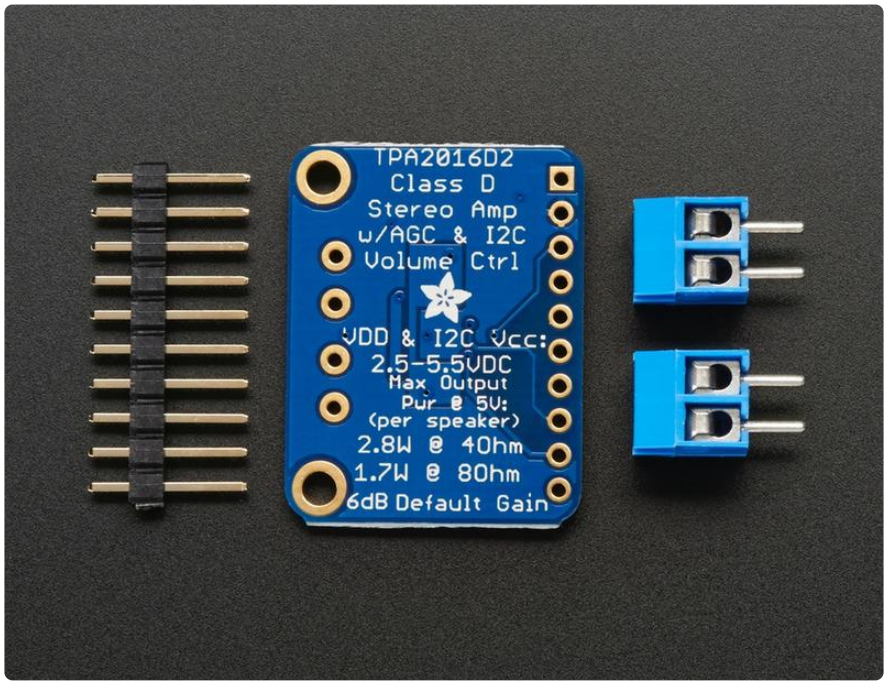
Inside the miniature chip is a class D controller, able to run from 2.7V-5.5VDC. Since the amp is a class D, it's incredibly efficient (89% efficient when driving an 8Ω speaker at 1.5 Watt) - making it perfect for portable and battery-powered projects. It has built in thermal and over-current protection but we could barely tell if it got hot. This board is a welcome upgrade to basic "LM386" amps!

Our Arduino library will let you set the AGC configuration (you can also just turn it off), max gain, and turn on/off the left & right channels all over I2C!

adafruit_products_1712front_LRG.jpg ()

Specifications

- Output Power: 2.8W at 4Ω, 10% THD, 1.7W at 8Ω, 10% THD, with 5V Supply
- PSRR: 80 dB, 5ms startup time
- Designed for use without an output filter, when wires are kept at under 2"-4" long
- I2C interface pins for setting gain, AGC configuration parameters, etc. See the Software page for more details
- Selectable gain from -28dB to 30dB
- Excellent click-and-pop suppression
- Thermal shutdown protection
- Shutdown pin for power saving mode.
- Low current draw: 3.5mA quiescent and 0.2uA in shutdown mode



What is a Class D Amplifier?

A Class D Amplifier uses PWM to generate high-frequency square waves with a duty-cycle proportional to the voltage level of the input audio signal. By minimizing the transition time between fully on and fully off, the MOSFET drivers are able to operate at a very high efficiency. Class D amplifiers such as this one typically operate at over 90% efficiency, compared to efficiencies of 50% or less for typical class AB amplifiers.

The high frequency square-wave component of the output signal is filtered by the inductance of the speaker voice coil, leaving only the amplified audio signal.

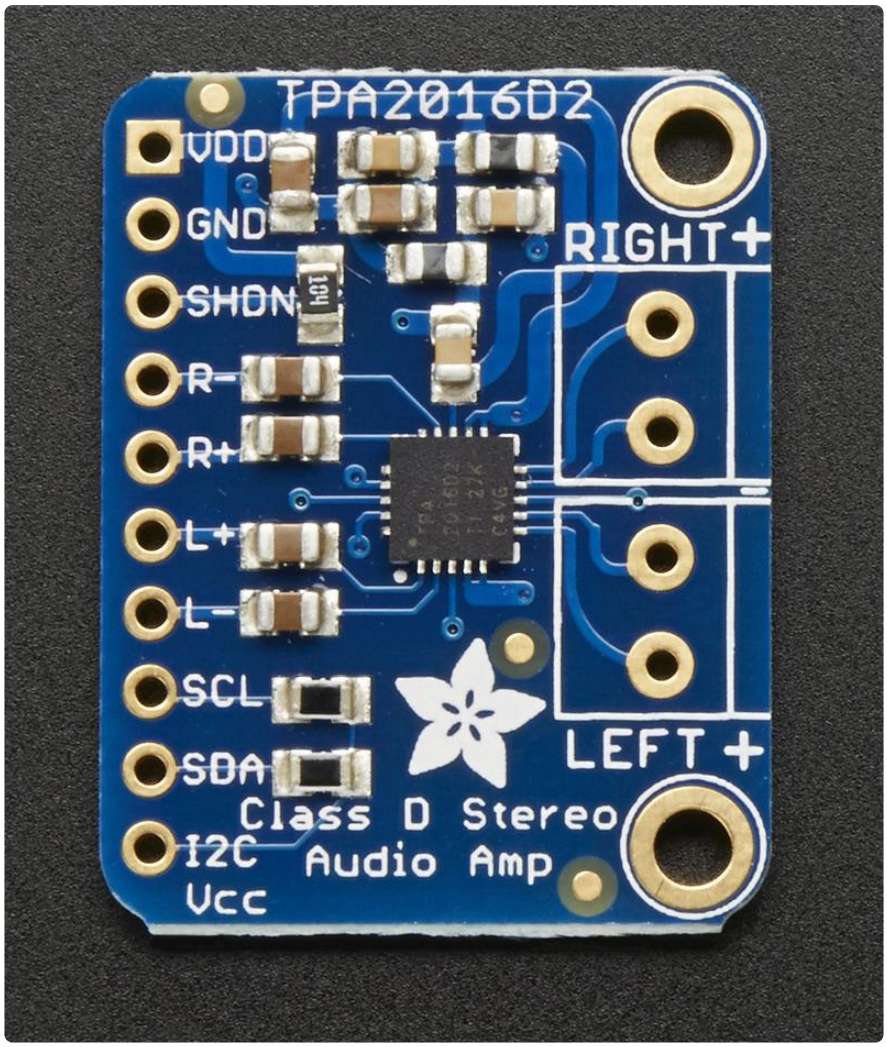
Other Audio amps available at Adafruit

We have a few choices of audio amplifiers, here's how you can compare them

- MAX98306 (http://adafru.it/987) - This class D audio amplifier has selectable gains of 6dB, 9dB, 12dB, 15dB and 18dB that you can choose with a jumper. It can do up to 3.7W into 3Ω, 2.8W into 4Ω and 1.7W into 8Ω. However, you cannot shut down each speaker separately. Its a good choice where you don't mind setting the gain with a jumper and if you do not need to ever turn off only one audio channel and you do not need more than 18dB. Its an excellent amplifier that can driver down to 3Ω speakers
  Like the TS2012, it has differential inputs, bridge tied outputs, and can run from 2.7V to 5.5V

- TS2012 (http://adafru.it/1552) - This class D audio amplifier has selectable gains of 6dB, 12dB, 18dB and 24dB that you can choose with a jumper (the MAX98306 goes up to 18dB only). It can do up to 2.8W into 4Ω and 1.7W into 8Ω. It cannot drive 3Ω. You can shut down each channel separately. Setting the gain is easy on the onboard DIP switches. Its a good choice where you don't need to drive 3Ω speakers or if you ever want to turn off only one audio channel. If you need 24dB gain this amp can do it.
  Like the MAX98306, it has differential inputs, bridge tied outputs, and can run from 2.7V to 5.5V

Pinouts



Using the TS2016 audio amplifier is pretty easy - the power, control and input pins are on the left. On the right are the speaker outputs.

The power and control pins are on a 0.1" spaced header. The speaker outputs are for 3.5mm spaced terminal blocks (included)

Speaker Outputs

On the right there are the 4 output speaker pins. These outputs are Bridge Tied Load which means you must connect the speakers directly - do not try to connect these outputs to another amplifier! Use any 4 to 8 ohm speakers. Lower resistance means you'll be able to get louder audio (2.8W max into 4 ohm, 1.7W max into 8 ohm). You'll want to make sure your speakers have a wattage rating higher than the max power, so make sure your 4 ohm speakers are 3W+ and your 8 ohm speakers are 2W+.

Otherwise you risk blowing out the speakers or otherwise damaging them with too much power. If you're really needing to driver smaller speakers, you can use the output limiting capability of the chip, see the Software page for details

Power Pins

Starting at the top of the left side, there are two power pins, VDD and GND that are used for powering the chip. These should be 2.7V to 5.5VDC. There's no polarity protection so make sure you get the wires in the right polarity! Higher voltages will give you more power so if you want that full 2.8W you need to give it 5 VDC.

Shutdown Control Pins

You can turn off the amplifier with the SHDN pin. Unlike the TS2012, this chip doesn't have separate pins for turning off each channel (left/right) from the breakout. Instead, if you want to turn off the left or right channel only, you'll need to do it via the i2c interface, see the Software page for details.
Connect this pin to ground to turn off the amplifier completely, By default this pin has pullups to VDD so both channels are on by default!

Audio input pins

There are four pins for the audio input signal. For right channel, R+ and R- and for left channel, L+ and L-. These are differential inputs. If you are connecting this to a device with differential outputs. just connect the + and - pins as indicated on that device's outputs. If there is only one differential reference, connect L- and R- together and tie that to your differential reference. If you are using every-day single-ended audio signal, connect L- and R- to ground, and L+ and R+ to your signal.

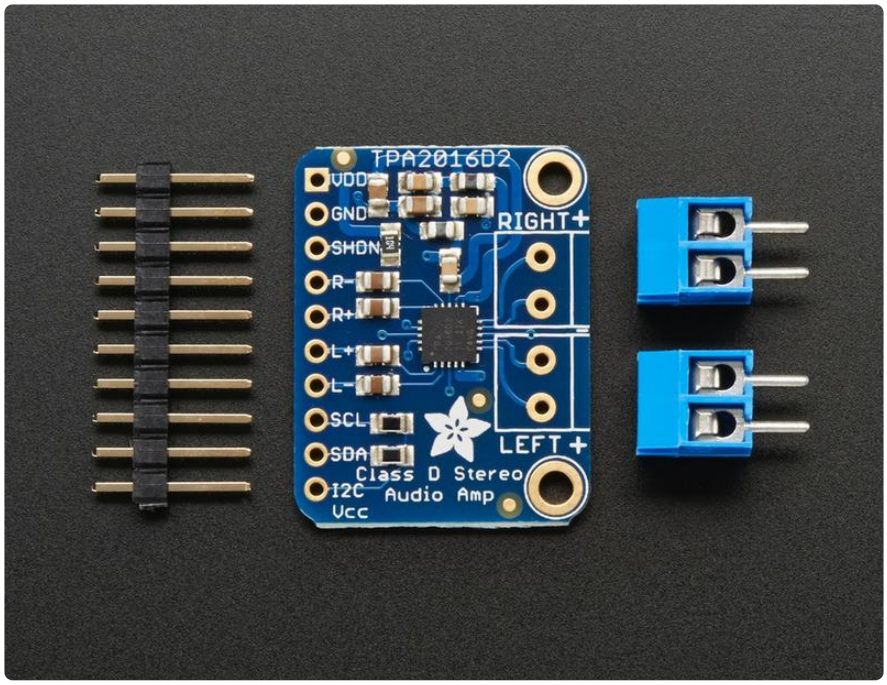
There are 1uF input series capacitors on all four pins so it is OK if your signal does not have audio bypass caps. If your signal does have audio bypass caps, you do not need to remove them, just keep them in.

I2C Control pins

The special thing about this amplifier is that instead of switches, jumpers or potentiometer for setting the gain, there's an i2c interface. This interface can be connected to just about any kind of microcontroller & we have example code for Arduino. To use, connect the SCL and SDA pins to your microcontroller I2C host pins. 10K pull-up resistors are built into the board, you do not need to add these.

You also have to connect the I2C VCC pin to your logic voltage level. If you're using a 3V microcontroller, connect it to 3V. If you're using a 5V microcontroller, connect it to 5V. This is so you can have 5V powering the amplifier but 3V to keep the levels happy for your lower voltage micro.

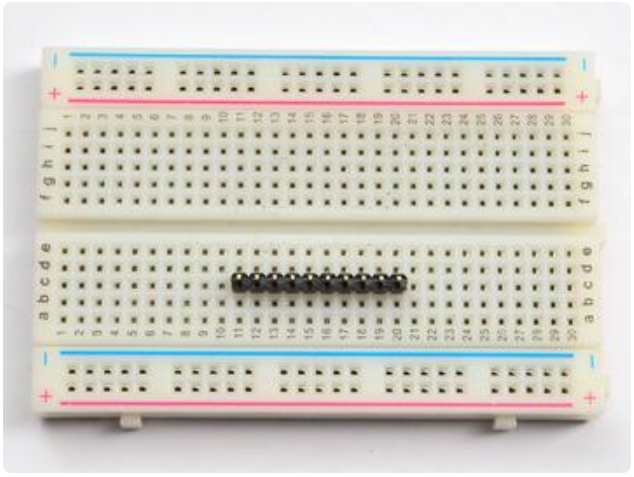
Assembly



Now that you have your amp board in hand, its time to solder & wire it up.

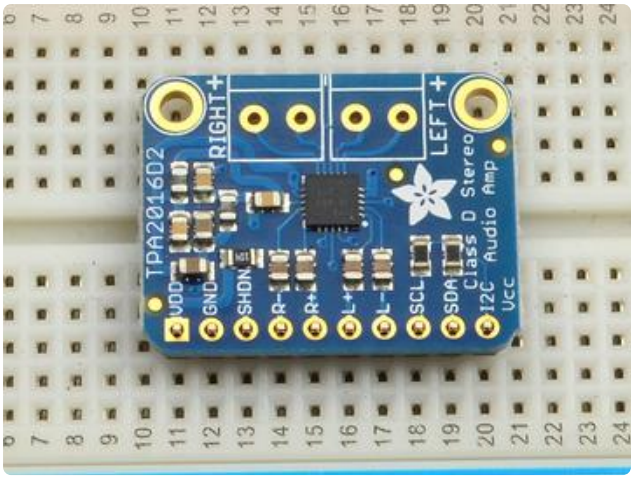
Header assembly

This step is optional if you plan to solder wires directly to the TPA2016 breakout. However, if you plan to use the amplifier in a breadboard or connect the speakers to the terminal blocks, you'll want to follow these steps! Soldering iron and solder are required, if you're new to soldering, check out our tutorial here ()

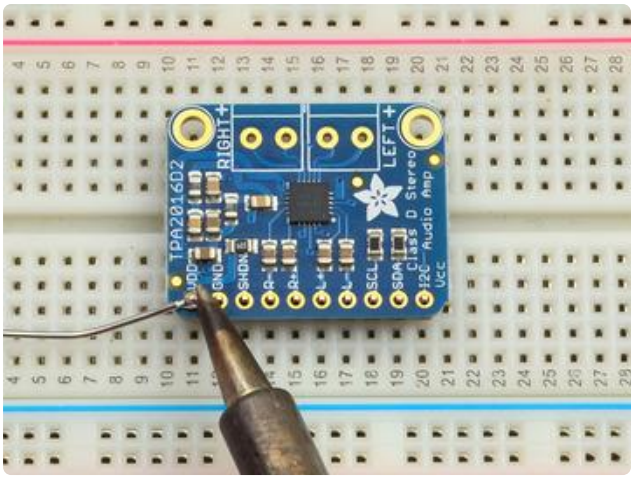


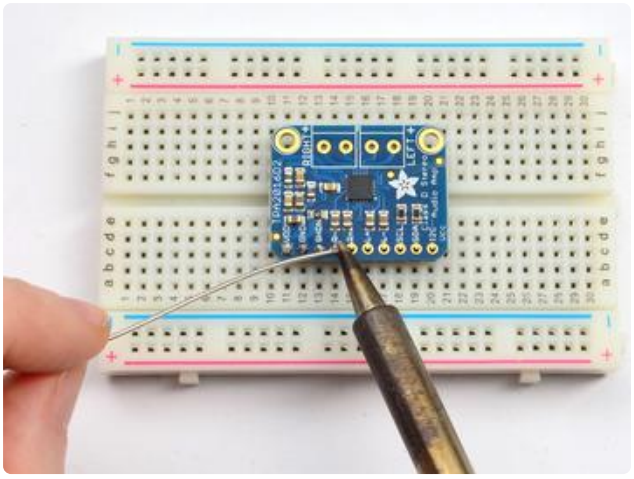
Start by finding the 0.1" male header that came with your kit and breaking it if necessary into a 10-pin long piece. You can use a pair of pliers or diagonall cutters to trim the header down to 10 pins long.

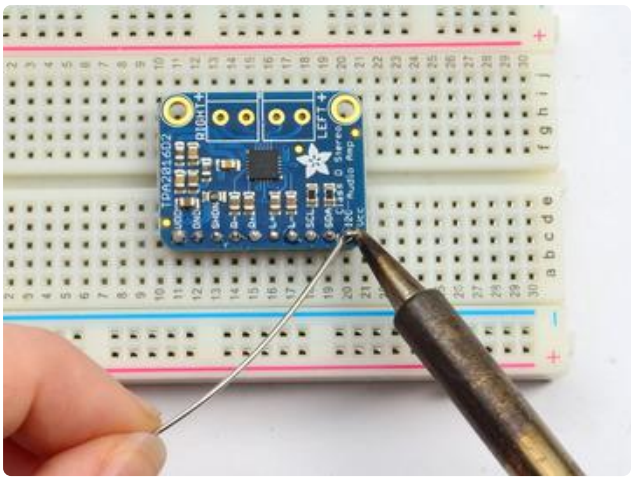
Place the long ends into a breadboard as shown.



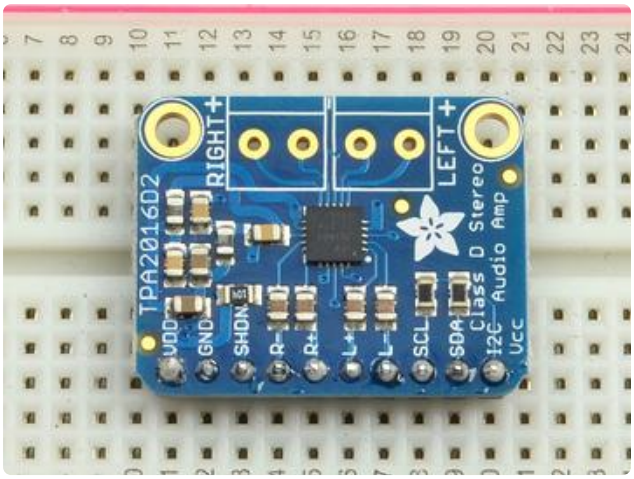
Now put the TS2012 on top so that the short ends of the pins stick thru the control and power pins.





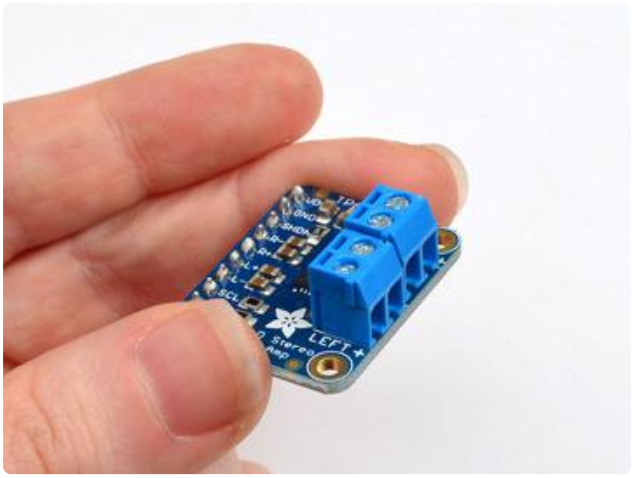


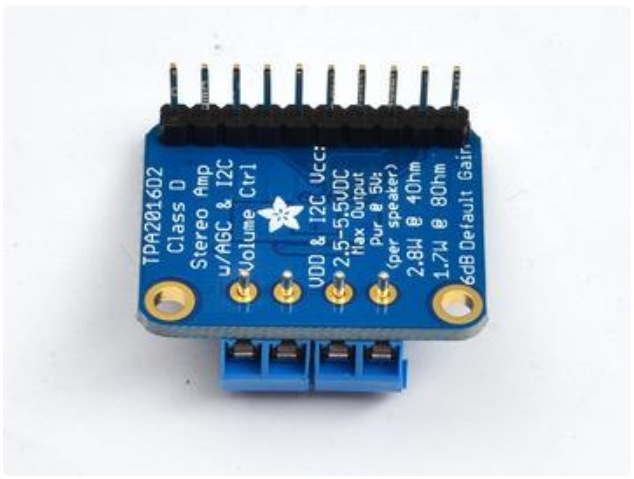
Solder them up! Solder each pin one by one, making sure you have a good connection for each pin and pad, nice shiny joints.



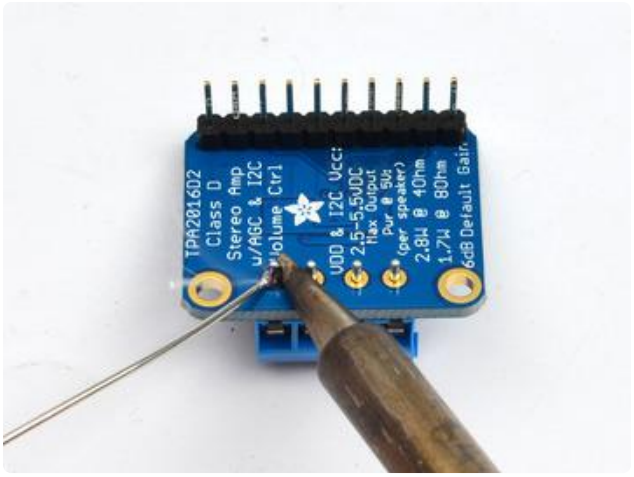
Check your work when done, make sure there are no bridged connections or cold solder joints.

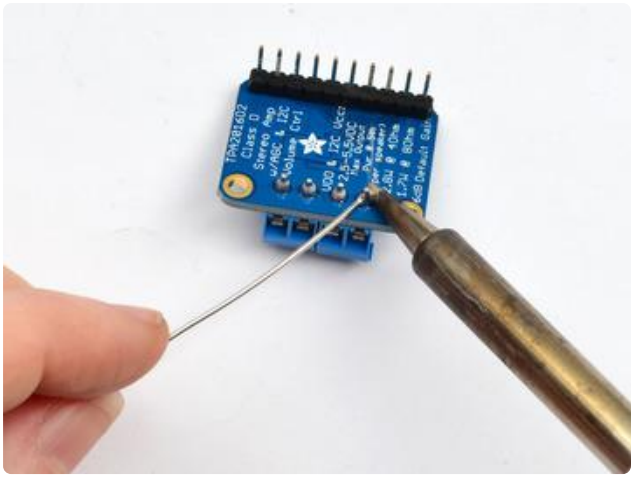
Speaker Terminal Blocks





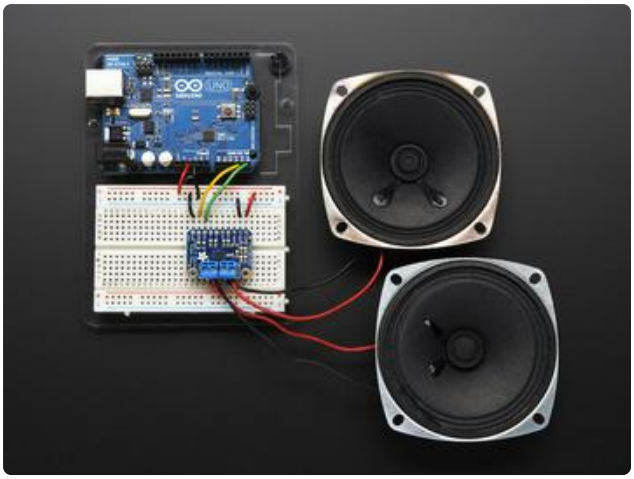
If you want to use the terminal blocks, place each one so the holes are pointing out.





Solder the four connections. they are bigger than most header so use plenty of solder and heat.

Check to make sure you have good connections



That's it! Now you can plug the amplifier into a breadboard and use a #1 Phillips or flat screwdriver to open and close the terminal blocks.

Wiring

You can perform a couple tests on the amplifier without hooking up the i2c pins - it'll be fixed at 6dB and the default AGC is on but you should be able to hear audio through speakers

Power

You can power the amplifier from 3-5V DC. If you want to use a wall adapter, a 2.1mm terminal block is handy (http://adafru.it/368)

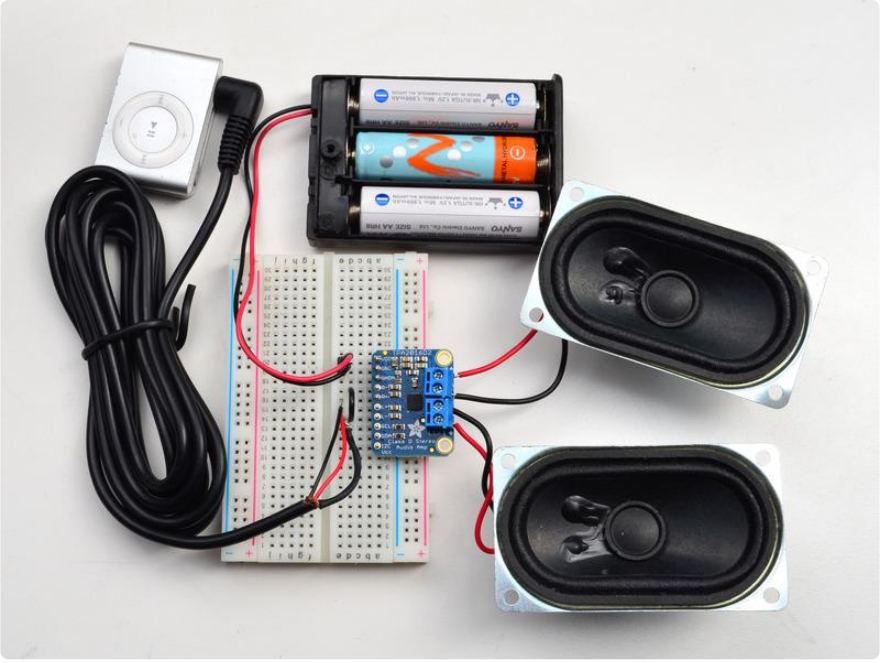
or you can use a 3xAA battery holder as we do in the photos below

Connecting Audio

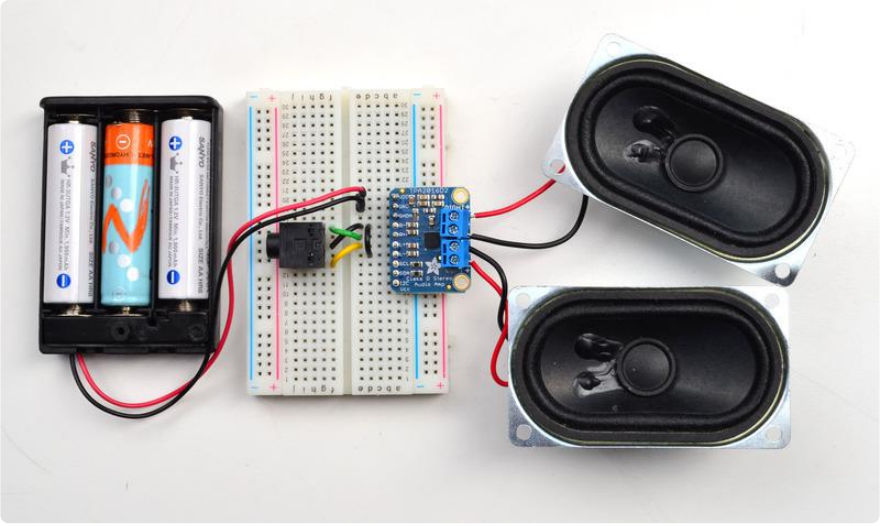
Testing the amplifier is really easy - no firmware or special configuration required to get a little audio out. In these images we'll assume you're using a breadboard but the wiring is the same if you soldered connections directly.

The easiest way to test your amplifier is to connect a 3.5mm audio pigtail cable to the audio inputs. Connect Right to R+ and Left to L+ then since a 3.5mm audio cable is single-ended, connect the ground wire to both R- and L-

You can connect the power to the board via either a breadboard rail with 3-5VDC on it (say from an Arduino power supply or battery pack)



If you want to plug in a 3.5mm jack into the breadboard instead of having the cable pigtail, you can wire up a stereo jack as shown below. Tie ground to R- and L- and left and right to L+ and R+



With just an audio source you can sort of see how the AGC system works - the amp will try to 'normalize' the volume to be constant

---

Arduino

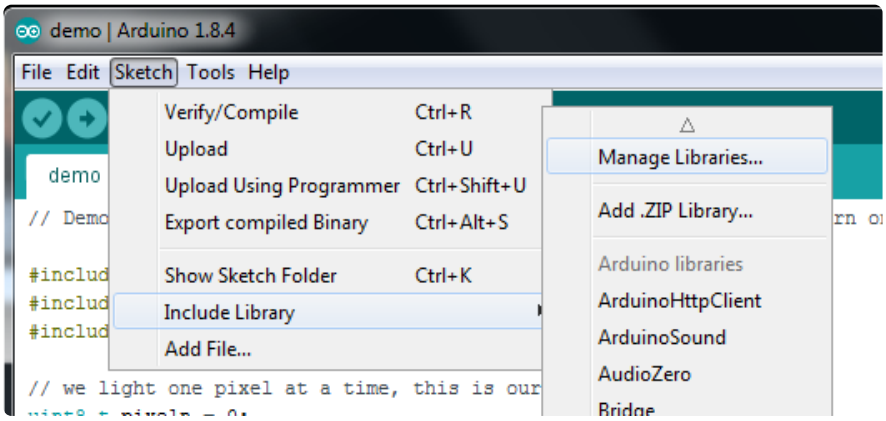
Using with Arduino

Its super easy to use this stereo amplifier board with an Arduino thanks to the great Adafruit library. Once you've installed the library you can connect the TPA2016 via I2C your Arduino, it will work with any kind or flavor. If you're using a different kind of microcontroller, the library is a good reference to help you port the code over.

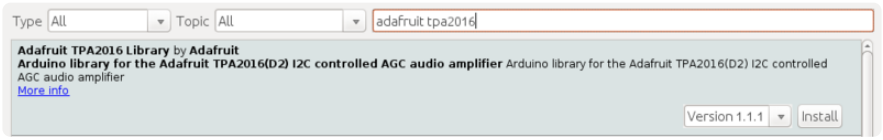
Download the library

First up, we'll download the Arduino library from the Arduino library manager.

Open up the Arduino library manager:



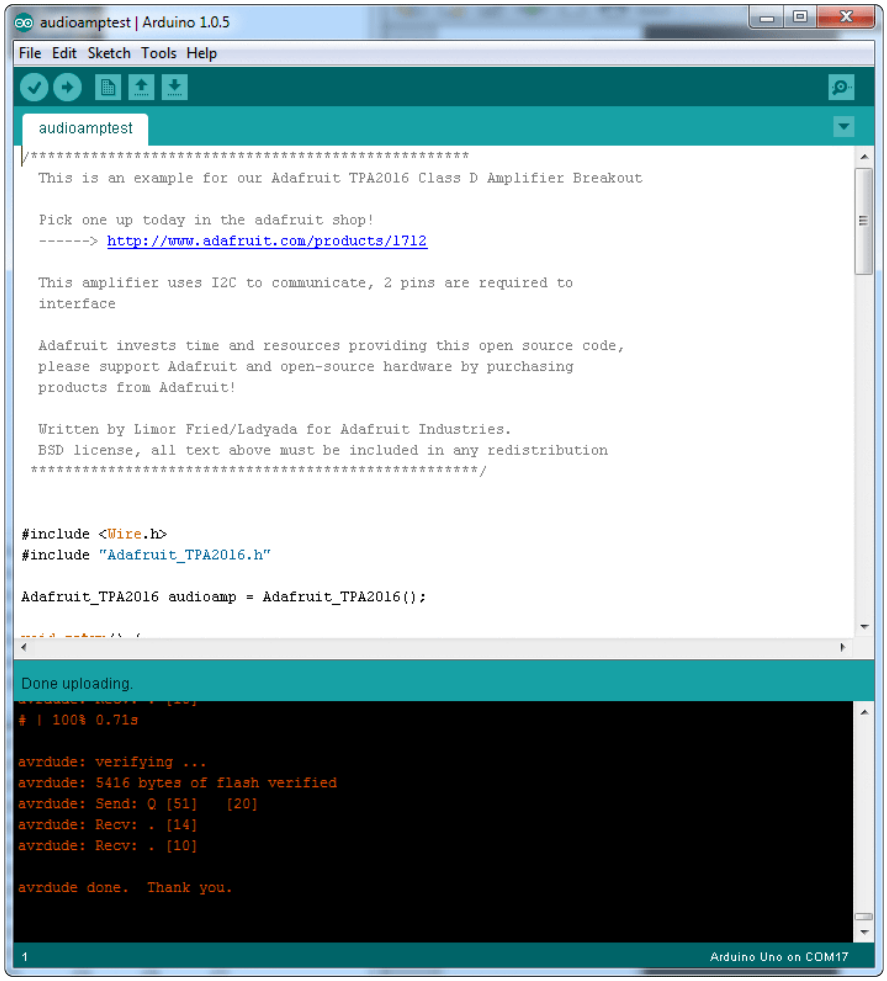
Search for the Adafruit TPA2016 library and install it



We also have a great tutorial on Arduino library installation at: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use ()

Run Test Sketch

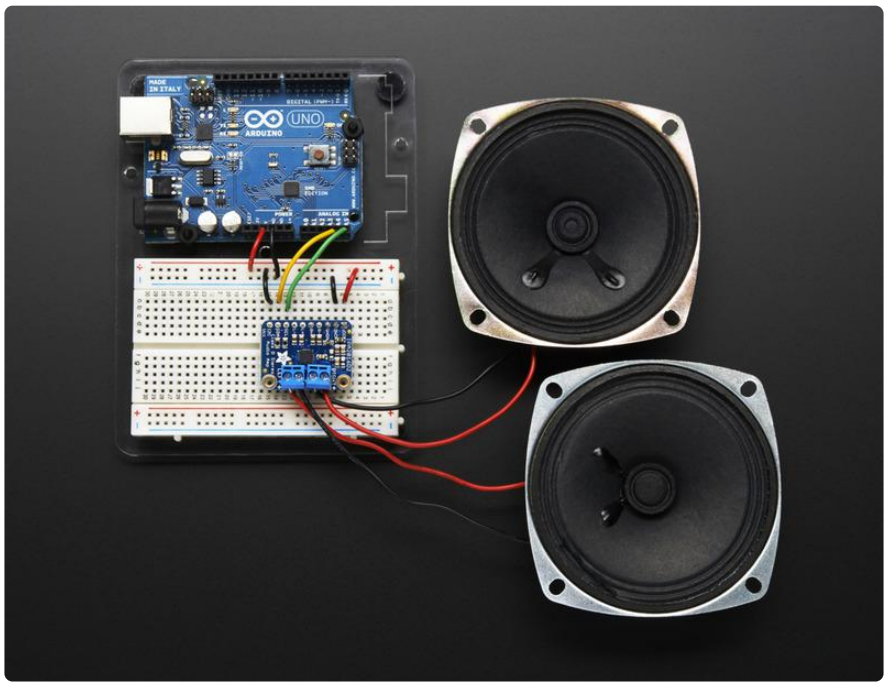
After you've restarted, you should be able to load up the File->Examples->Adafruit_TPA2016->audioamp sketch



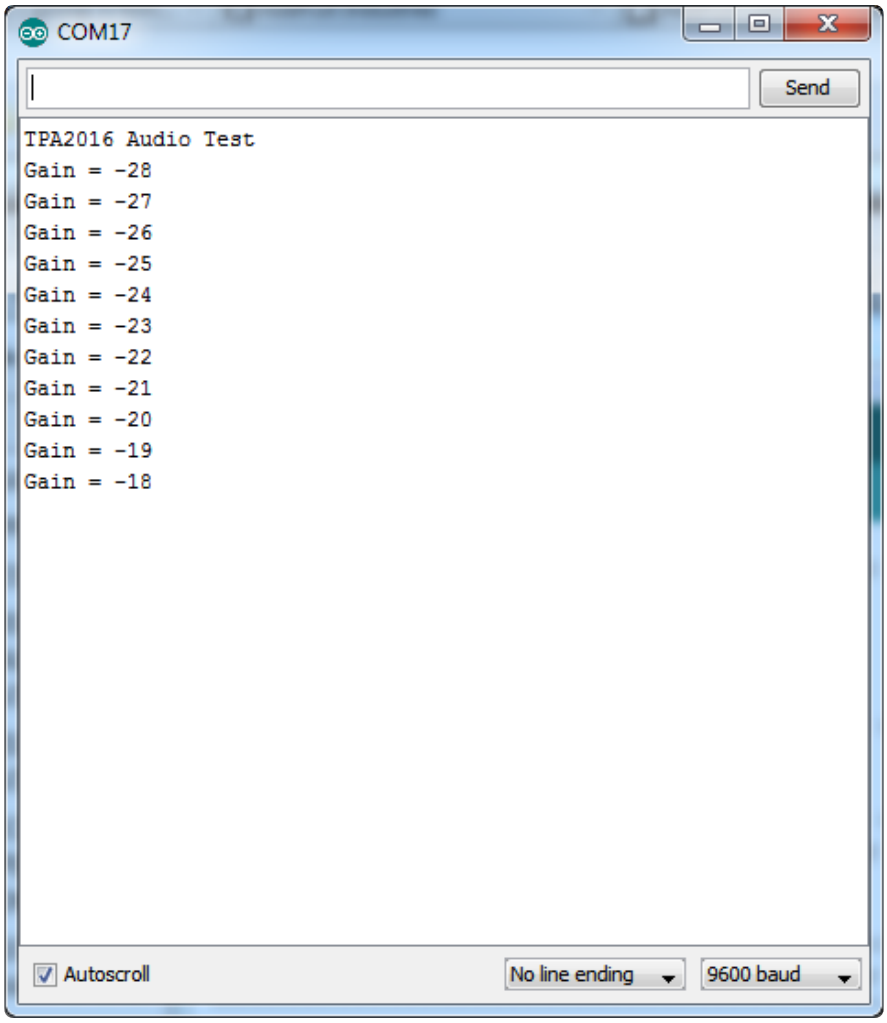
Wire up the sensor as shown, to I2C. You'll want to wire up an audio source using a pigtail cable or a audio jack as shown. Connect the GND pin to ground, VDD and I2C Vcc pin to 5V and connect SDA to your Arduino's SDA pin, SCL to SCL pin

- On UNO/Duemilanove/etc, SDA == Analog 4, SCL == Analog 5
- On Leonardo/Micro, SDA == Digital 2, SCL == Digital 3

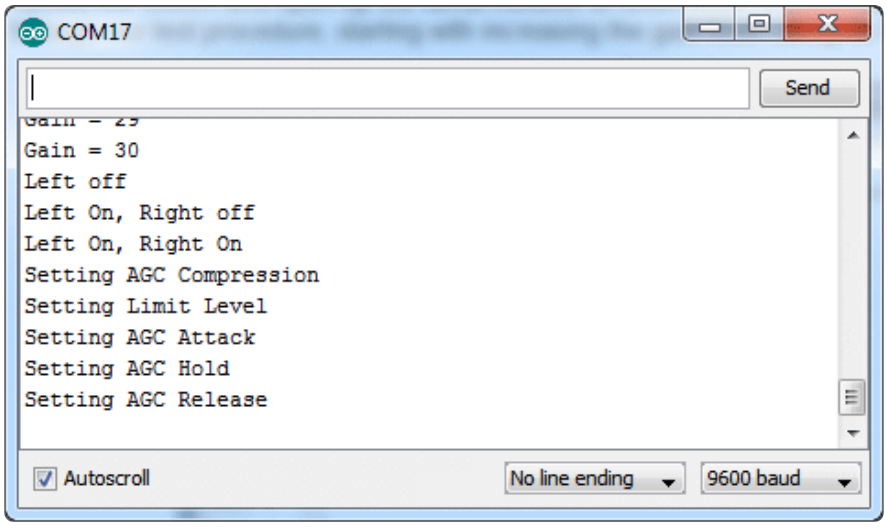
- On Mega/ADK/Due, SDA == Digital 20, SCL == Digital 21



Upload the sketch and open up the serial console at 9600 baud. You should see that the Arduino go through our test procedure, starting with increasing the gain from -28 up to 30 dB.



The test code will also toggle each of the speakers on and off (left channel, right channel) and set up the AGC parameters.



Pretty much if you can hear the gain increase in the beginning and then each speaker turn on & off then the test has completed successfully!

Now you can configure the amplifier settings on your own using the library

procedures.

Adafruit TPA2016 Library reference

To use the library, you'll need to have something like these lines in the beginning of your code, they'll include the I2C Wire library and the Adafruit library as well. Then it will create the audio amplifier object. There's no way to change the I2C address on this device.

```
#include &lt;Wire.h&gt;
#include "Adafruit_TPA2016.h"

Adafruit_TPA2016 audioamp = Adafruit_TPA2016();
```

Then somewhere in your setup() function, start the amplifier interface with

```
audioamp.begin();
```

The amplifier starts 'on' by default and with 6dB gain so running begin() won't change that.

Channel Control

Now that you're talking to the amp, you can do stuff like turn on and off the right and left channels with

```
audioamp.enableChannel(rightchannel, leftchannel);
```

for example, to turn off the left channel and keep the right one on, use

```
audioamp.enableChannel(true, false);
```

Both channels are on by default

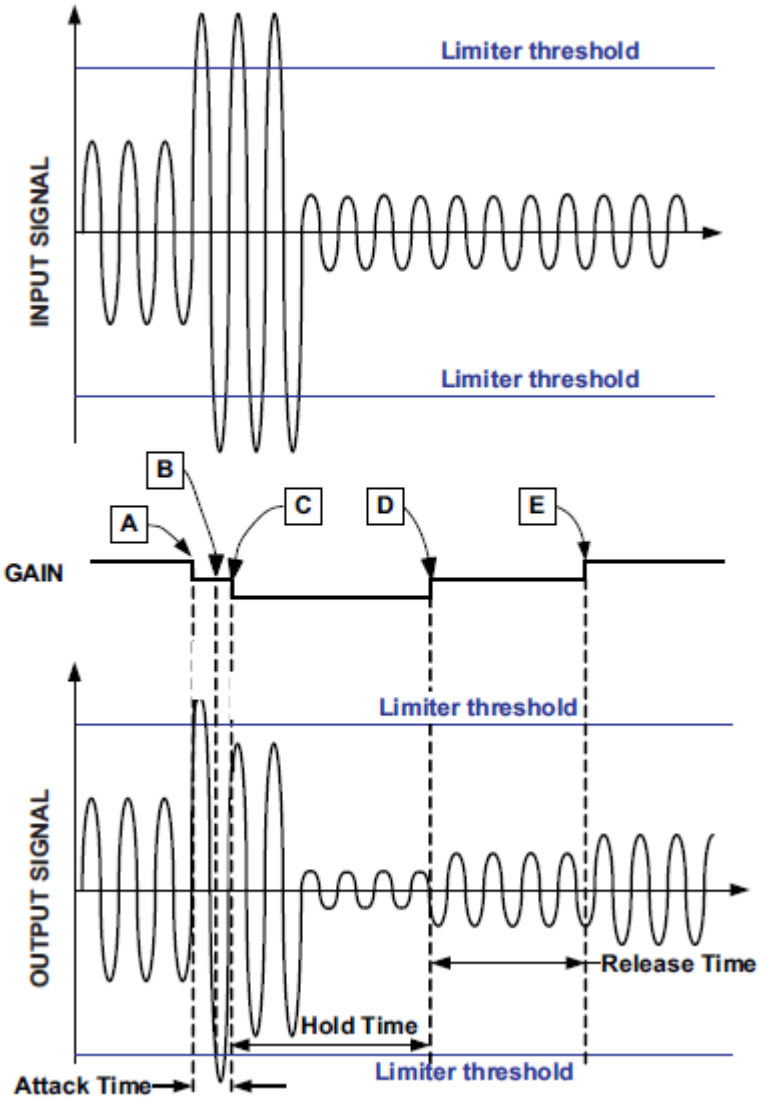
Gain Control

You can control the max gain that the amplifier will use, this is even with the AGC on. So if you select 20dB for example, the AGC won't automatically amplify any higher than 20dB.

You can set the gain with:

```
audioamp.setGain(gain);
```

Where gain is from -28 (dB) to 30 (dB)



Limiter Settings

The limiter is what helps avoid overdriving speakers if you have a fixed Wattage you want to stay under. By default it is On, you can turn it off with

```
audioamp.setLimitLevelOn();
```

and back off with

```
audioamp.setLimitLevelOff();
```

You can set the maximum voltage gain with

```
audioamp.setLimitLevel(limit);
```

where limit ranges from 0 (-6.5dBv) to 31 (9dBV) see page 22 of the Datasheet for more details.

AGC configuration settings

You can set up the AGC for how it will react to audio level changes. If you're an audio geek you can use the tables on page 23-24 of the datasheet to set up the AGC exactly how you like. If you aren't sure what settings to use, check page 19 of the datasheet which has some basic guidance.

You can start by setting the AGC compression ratio. This tells the amplifier how much to automatically tweak the gain to make loud and soft sounds about the same volume. For example, spoken word should have high compression since you want to have it all about the same volume. Adjustments are made by calling

```
audioamp.setAGCCompression(compression);
```

Where the compression setting is TPA2016_AGC_OFF (1:1 compression), TPA2016_AGC_2 (1:2 compression), TPA2016_AGC_4 (1:4 compression), or TPA2016_AGC_8 (1:8 compression).

For example, to set the attack, use:

```
audioamp.setAttackControl(attackvalue);
```

where attackvalue ranges from 0-31
To set the hold value, use:

```
audioamp.setHoldControl(holdvalue);
```

Where holdvalue is between 0-31
and for the AGC release, use:

```
audioamp.setReleaseControl(releasevalue);
```

Where releasevalue is also between 0-31

Pages 23-24 of the datasheet will tell you how to convert those values to ms/dB numbers.

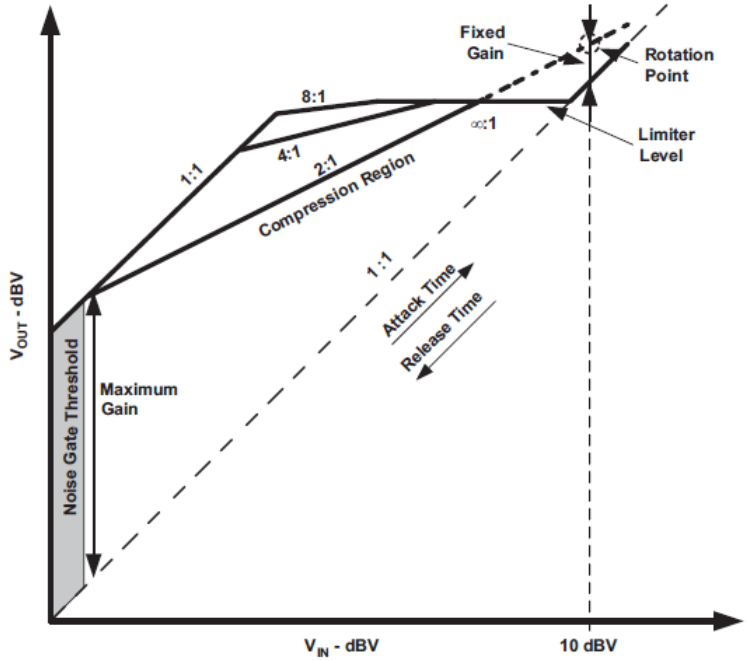


Figure 35. Output Signal vs. Input Signal State Diagram

Python & CircuitPython

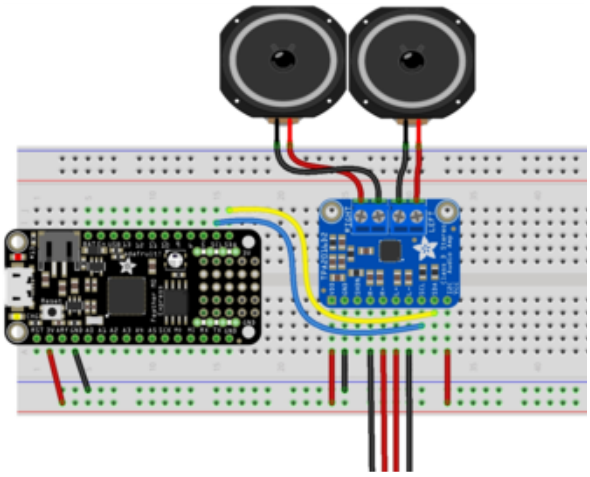
It's easy to use the TPA2016 breakout with Python or CircuitPython and the Adafruit CircuitPython TPA2016 () module. This module allows you to easily write Python code that controls gain, power and more using the breakout.

You can use this breakout with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

CircuitPython Microcontroller Wiring

First wire up a TPA2016 to your board exactly as shown on the previous pages for Arduino. This breakout uses I2C.

Here's an example of wiring a Feather M0 to the sensor with I2C:

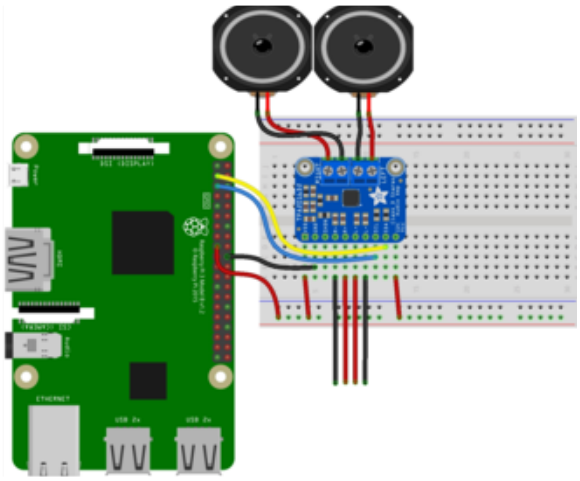


Board 3V to breakout VDD
Board 3V to breakout I2C VDD
Board GND to breakout GND
Board SCL to breakout SCL
Board SDA to breakout SDA
Breakout R+/R- to audio input R+/R-
Breakout L+/L- to audio input L+/L-
Breakout screw terminal R+/R- to Speaker R+/R-
Breakout screw terminal L+/L- to Speaker L+/L-

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired with I2C:



Pi 3.3V to breakout VDD
Pi 3.3V to breakout I2C VDD
Pi GND to breakout GND
Pi SCL to breakout SCL
Pi SDA to breakout SDA
Breakout R+/R- to audio input R+/R-
Breakout L+/L- to audio input L+/L-
Breakout screw terminal R+/R- to Speaker R+/R-
Breakout screw terminal L+/L- to Speaker L+/L-

CircuitPython Installation of TPA2016 Library

You'll need to install the Adafruit CircuitPython TPA2016 () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (). Our CircuitPython starter guide has a great page on how to install the library bundle ().

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- adafruit_tpa2016.mpy
- adafruit_bus_device
- adafruit_register

Before continuing make sure your board's lib folder or root filesystem has the adafruit_tpa2016.mpy, adafruit_bus_device, and adafruit_register files and folders copied over.

Next connect to the board's serial REPL () so you are at the CircuitPython >>> prompt.

Python Installation of TPA2016 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready ()!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-tpa2016`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the breakout we'll initialize it, and set the gain and more from the board's Python REPL.

Since you're using an I2C connection run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import busio
import board
import adafruit_tpa2016

i2c = busio.I2C(board.SCL, board.SDA)
tpa = adafruit_tpa2016.TPA2016(i2c)
```

Now you're ready to manipulate sound with the breakout using any of these properties:

- `fixed_gain` - The fixed gain of the amplifier in dB. If compression is enabled, fixed gain is adjustable from `-28` to `30`. If compression is disabled, fixed gain is adjustable from `0` to `30`.
- `max_gain` - The max gain in dB. Must be between `18` and `30`.
- `amplifier_shutdown` - Amplifier shutdown. Amplifier is disabled if `True`. Defaults to `False`. If `True`, device is in software shutdown, e.g. control, bias and oscillator are inactive.
- `speaker_enable_l` - Enables left speaker. Defaults to enabled. Set to `False` to disable.
- `speaker_enable_r` - Enables right speaker. Defaults to enabled. Set to `False` to disable.

For example to set `max_gain` and `fixed_gain`:

```
tpa.max_gain = 28
tpa.fixed_gain = -16
```

To shutdown the amplifier:

```
tpa.amplifier_shutdown = True
```

To disable the left speaker:

```
tpa.speaker_enable_l = False
```

This demonstrates some of the capabilities of this breakout.

To use real-time compression, the driver and breakout include the ability to set `compression_ratio`, `attack_time`, `release_time` and more. For more information and examples, check out the documentation ().

That's all there is to using the TPA2016 with CircuitPython!

Full Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import busio
import board
import adafruit_tpa2016

i2c = busio.I2C(board.SCL, board.SDA)
tpa = adafruit_tpa2016.TPA2016(i2c)

tpa.fixed_gain = -16
```

Python Docs

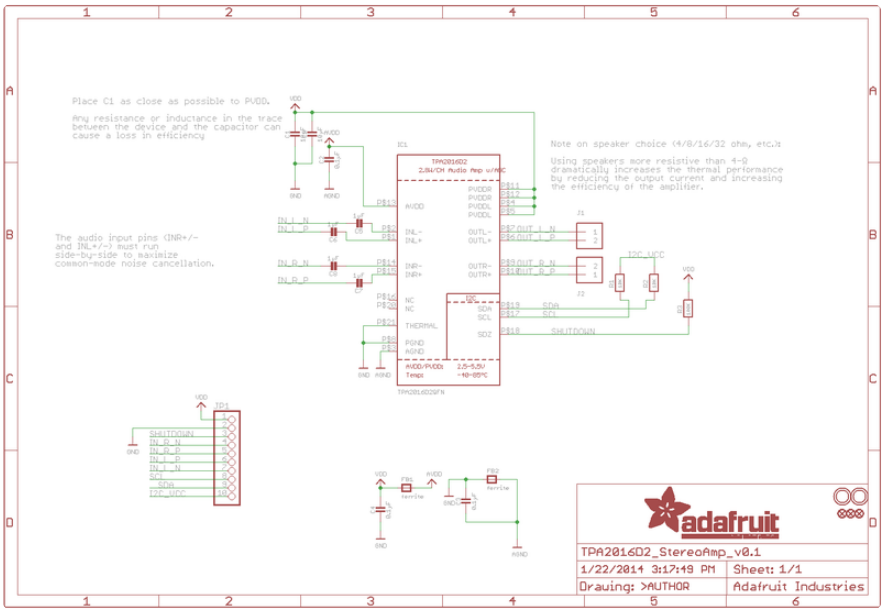
Python Docs ()

Downloads

Datasheets

- TPA2016 Datasheet ()
- Fritzing object in Adafruit Fritzing library ()
- EagleCAD PCB files on GitHub ()

Schematic



Dimensions