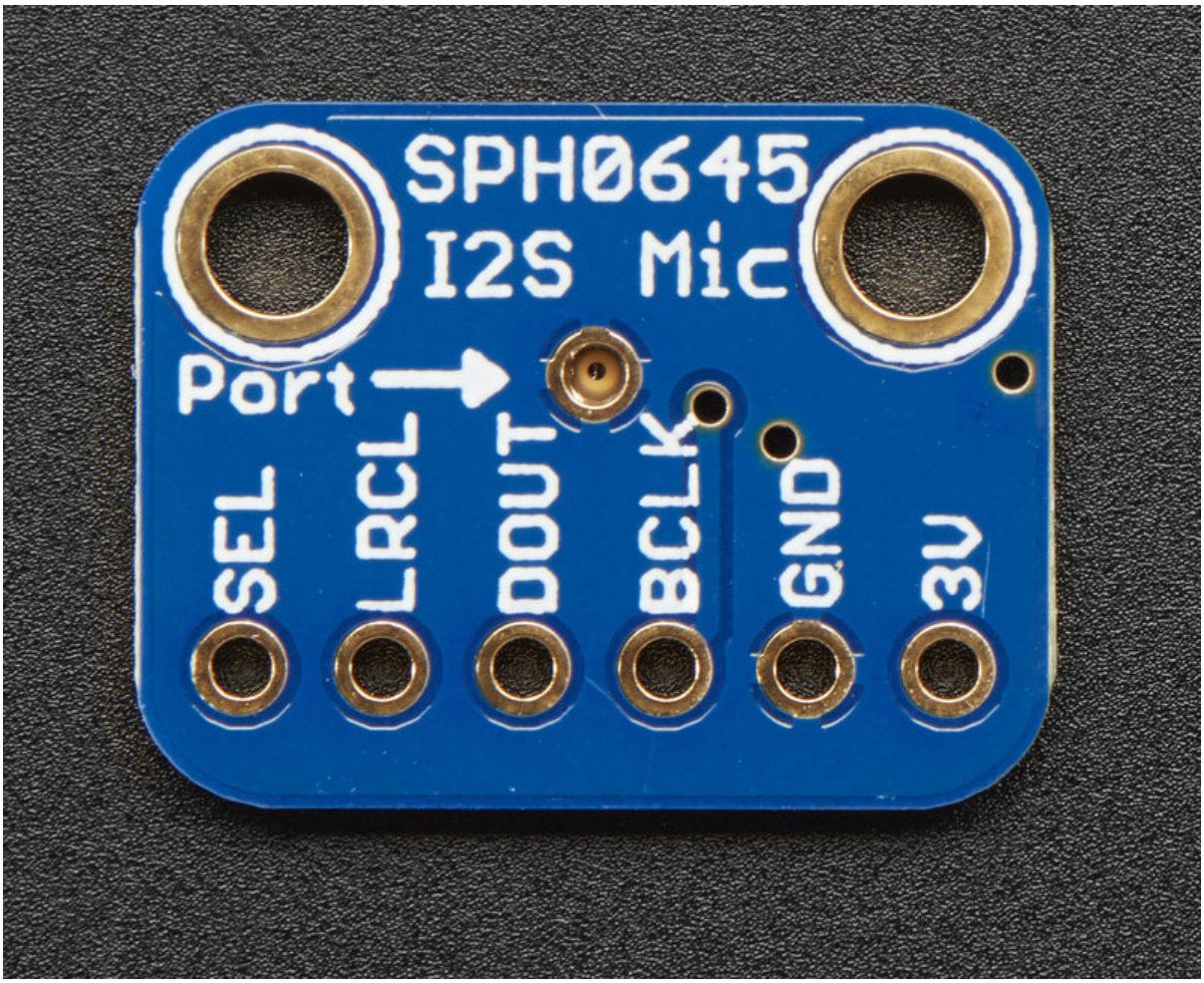
Adafruit I2S MEMS Microphone Breakout

Created by lady ada

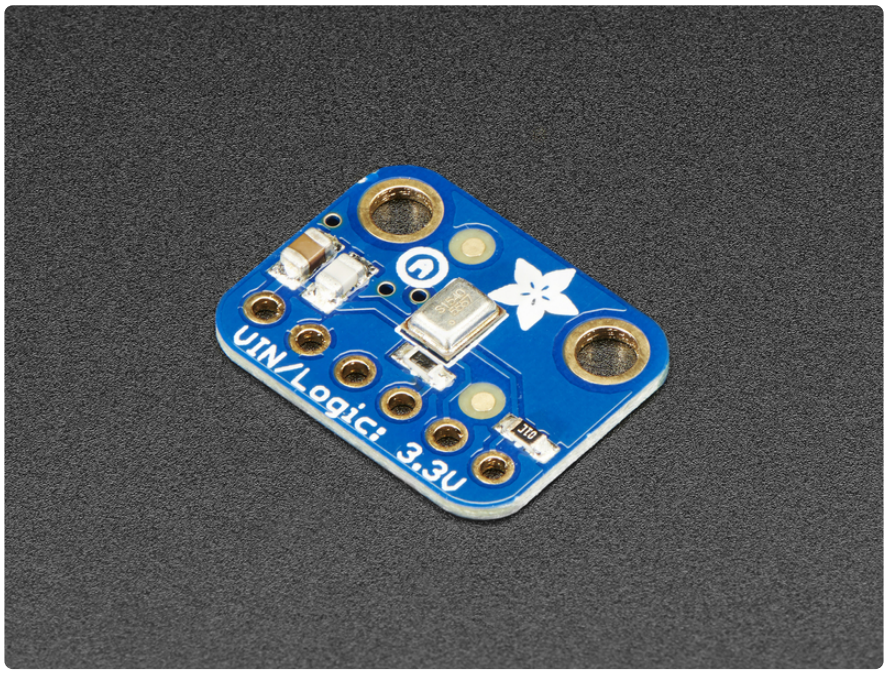


https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout

Last updated on 2024-10-19 01:39:19 PM EDT

Table of Contents

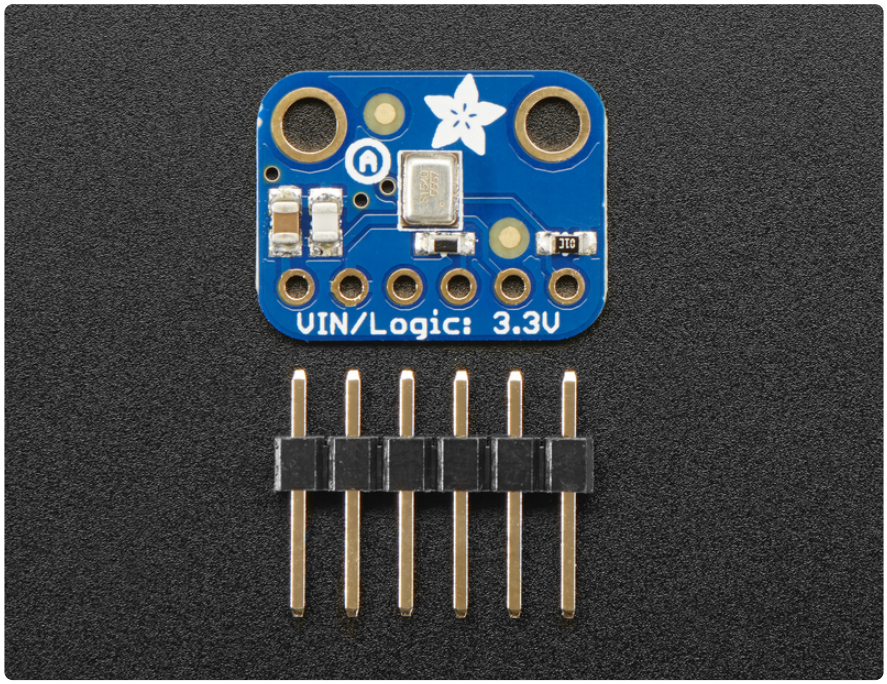
Overview



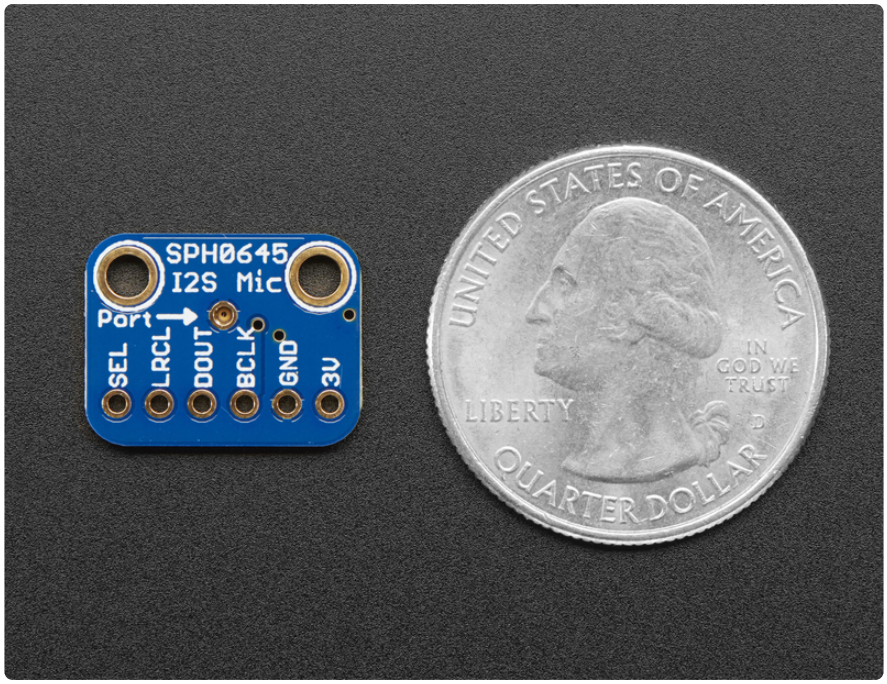
For many microcontrollers, adding audio input is easy with one of our analog microphone breakouts (http://adafru.it/1063). But as you get to bigger and better microcontrollers and microcomputers, you'll find that you don't always have an analog input, or maybe you want to avoid the noise that can seep in with an analog mic system. Once you get past 8-bit micros, you will often find an **I2S** peripheral, that can take digital audio data in! That's where this **I2S Microphone Breakout** comes in.

Instead of an analog output, there are three digital pins: Clock, Data and Word-Select. When connected to your microcontroller/computer, the 'I2S Main' will drive the clock

and word-select pins at a high frequency and read out the data from the microphone. No analog conversion required!



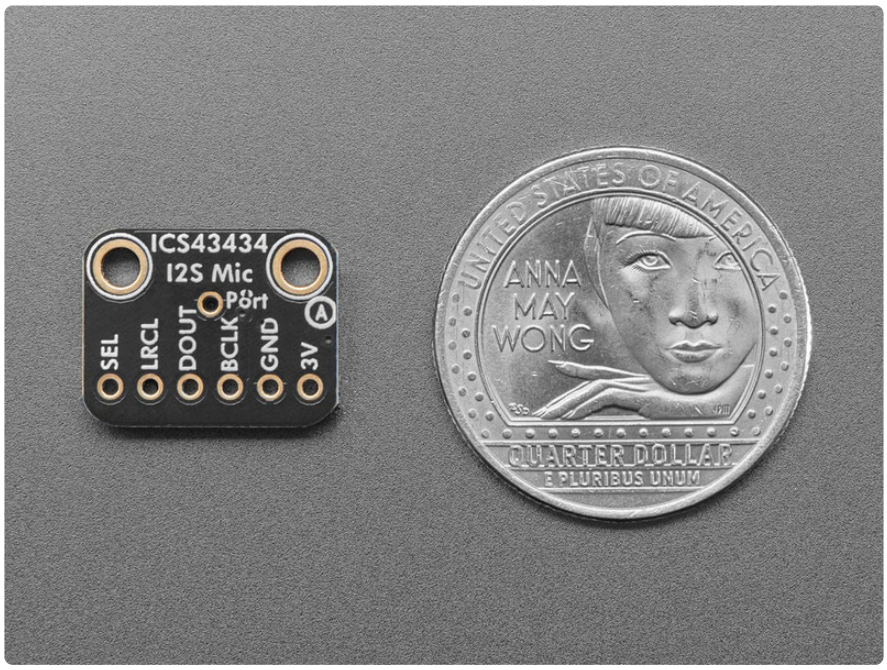
The microphone is a single mono element. You can select whether you want it to be on the Left or Right channel by connecting the Select pin to power or ground. If you have two microphones, you can set them up to be stereo by sharing the Clock, WS and Data lines but having one with Select to ground, and one with Select to high voltage.



This I2S MEMS microphone is bottom ported, so make sure you have the hole in the bottom facing out towards the sounds you want to read. It's a 1.6-3.3V device only, so

not for use with 5V logic (its really unlikely you'd have a 5V-logic device with I2S anyways). Many beginner microcontroller boards don't have I2S, so make sure its a supported interface before you try to wire it up!

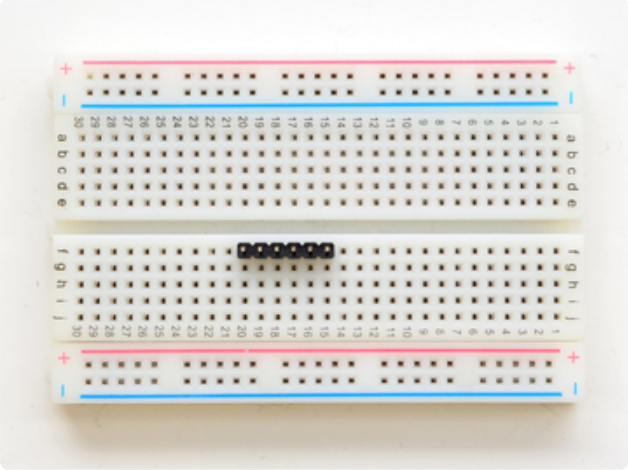
This microphone is best used with microcontrollers or computers that have **hardware I2S peripheral support** such as the Cortex M-series chips like the Arduino Zero, Feather M0, or single-board computers like the Raspberry Pi.



Assembly

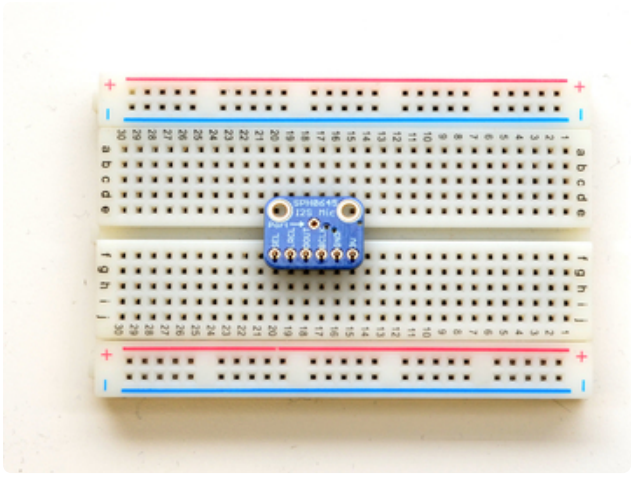
Assembly is really easy, you can use straight or 'right-angle' style headers to attach to the PCB. We'll be using the plain straight headers included

The board comes with all surface-mount components pre-soldered. The included header strip can be soldered on for convenient use on a breadboard or with 0.1" connectors. You can also skip this step and solder on wires.



Prepare the header strip:

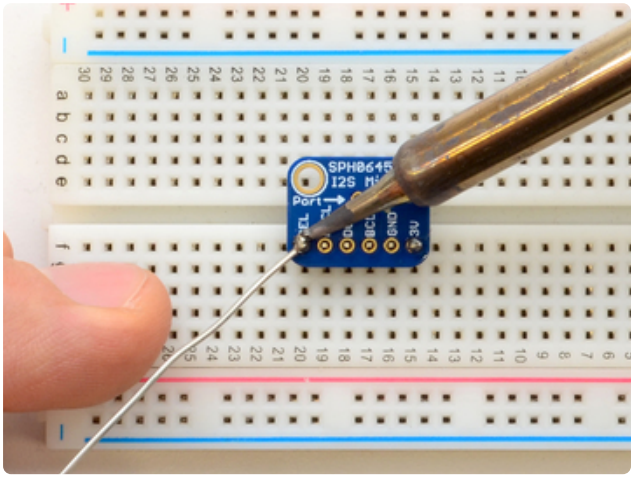
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

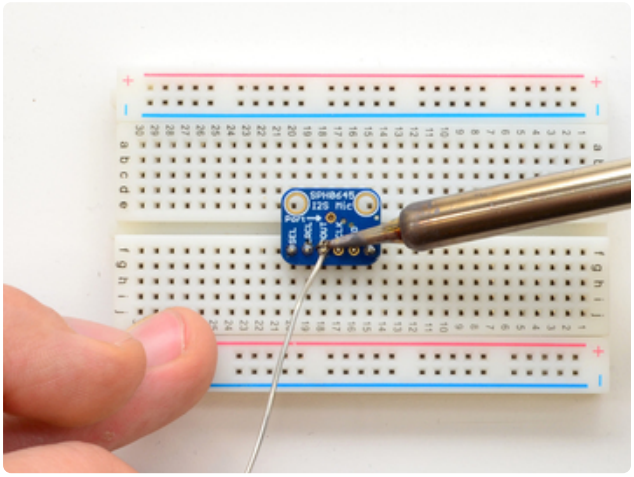


Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads

Make sure the side with the components is face down, as shown in the photos in this guide!

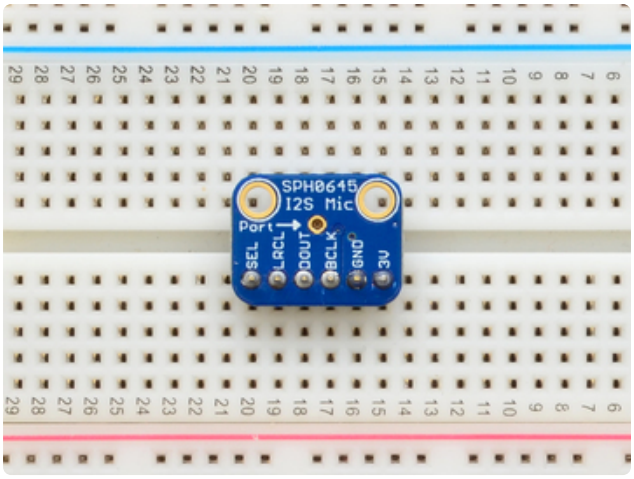




And Solder!

Be sure to solder all 5 pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)).

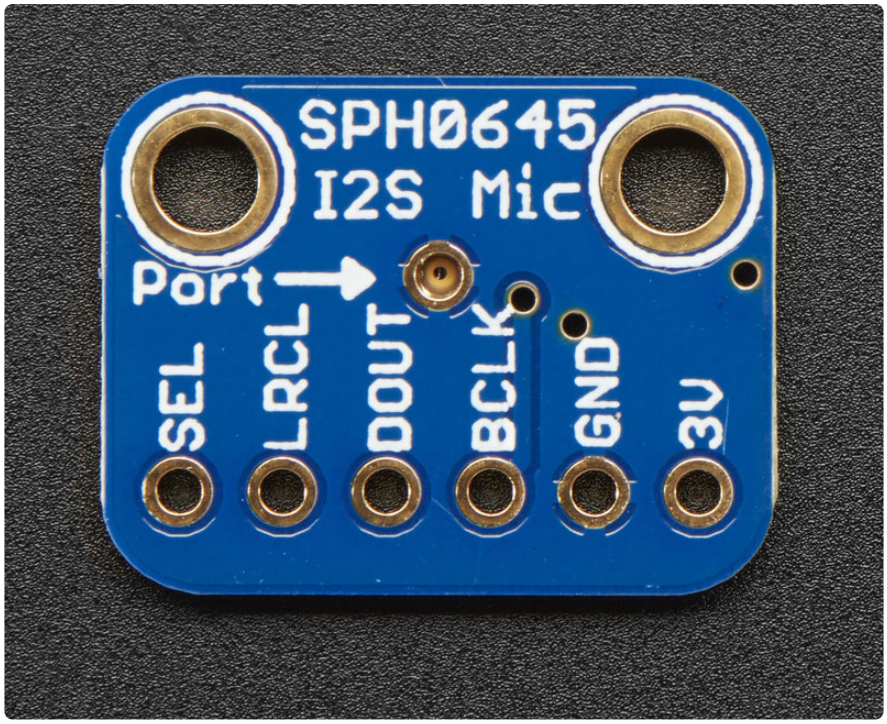


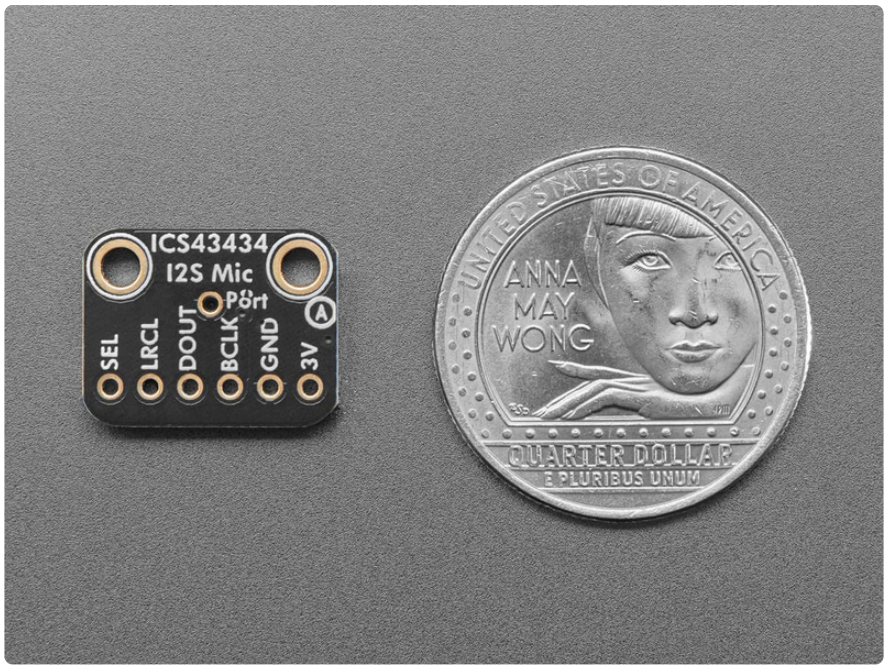
You're done! Check your solder joints visually and continue onto the next steps

---

Pinouts

Unlike most of our breakouts, this sensor has the detection element on the **bottom** of the PCB, so we expect you to solder it 'upside down' with the sensor package on the bottom and the port on top!





Power Pins

- **3V** - this is the power in pin. Technically it can be powered from as low as 1.6V to 3.6V but you'll need to make sure your logic level matches!
- **GND** - power and data ground

I2S Data Pins

- **BCLK** - the bit clock, also known as the data clock or just 'clock' - comes from the I2S main to tell the microphone its time to transmit data. This should run at 2-4 MHz but we've found you can often run it a little slower and it'll work fine
- **DOUT** - the data output from the mic!
- **LRCLK** - the left/right clock, also known as **WS** (word select), this tells the mic when to start transmitting. When the **LRCLK** is low, the left channel will transmit. When LRCLK is high, the right channel will transmit.
- **SEL** - the channel select pin. By default this pin is low, so that it will transmit on the left channel mono. If you connect this to high logic voltage, the microphone will instantly start transmitting on the right channel.

Arduino Wiring & Test

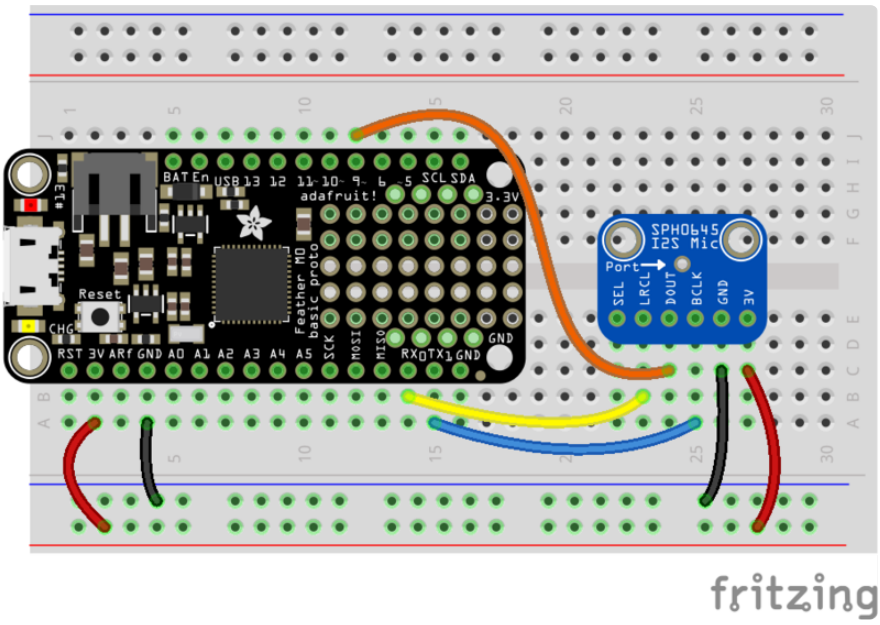
Remember, the I2S microphone requires an I2S peripheral and won't work with chips that don't support it in hardware! For this example we'll use a Feather M0, but you can also use an Arduino Zero.

Wiring

For Feather M0, Ardruino Zero and friends, use the following wiring:

- **GND** connected GND
- **3.3V** connected 3.3V (Feather, Zero) or VCC (MKR1000, MKRZero)
- **LRCLK** / WS connected to pin 0 (Feather, Zero) or pin 3 (MKR1000, MKRZero)
- **BCLK** connected to pin 1 (Feather, Zero) or pin 2 (MKR1000, MKRZero)
- **Data** /SD connected to pin 9 (Zero) or pin A6 (MKR1000, MKRZero

You can leave **Select** disconnected

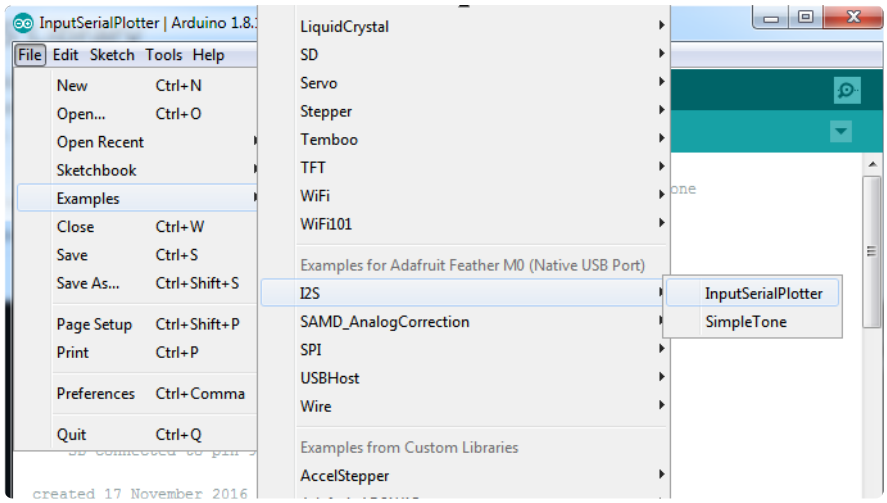


i2sfeatherm0.fzz

https://adafru.it/uya

I2S Library

Luckily, there's a nice little I2S library already written for Arduinos based on the SAMD processor. Make sure you have the most recent Arduino IDE and SAMD core. Then select the board you're using (e.g. Adafruit Feather M0) and you'll see the I**2S** library examples show up in the pulldown menu

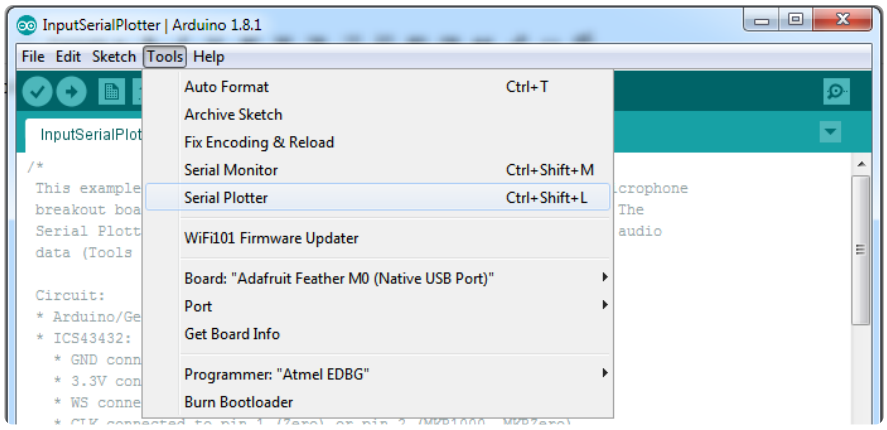


You could try the InputPlotter demo but this code is higher resolution:

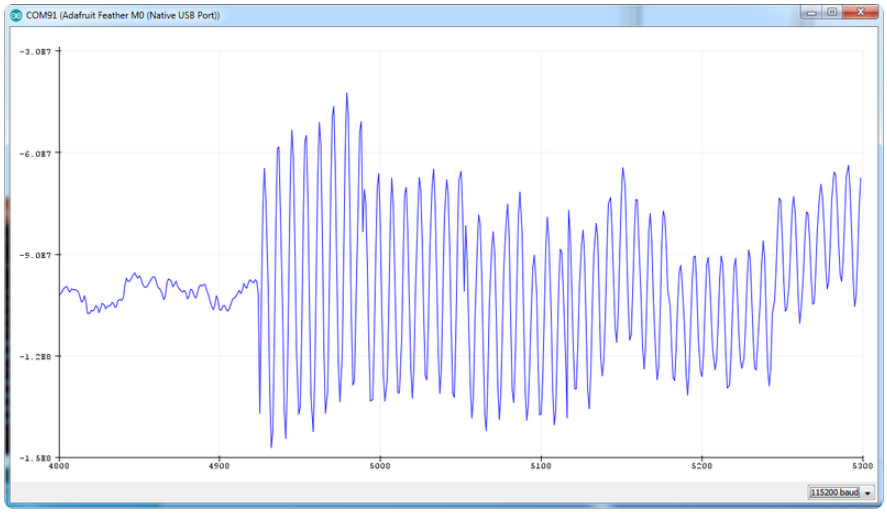
```
/*
 This example reads audio data from an I2S microphone
 breakout board, and prints out the samples to the Serial console. The
 Serial Plotter built into the Arduino IDE can be used to plot the audio
 data (Tools -&gt; Serial Plotter)

 Circuit:
 * Arduino/Genuino Zero, MKRZero or MKR1000 board
   * GND connected GND
   * 3.3V connected 3.3V (Zero) or VCC (MKR1000, MKRZero)
   * WS connected to pin 0 (Zero) or pin 3 (MKR1000, MKRZero)
   * CLK connected to pin 1 (Zero) or pin 2 (MKR1000, MKRZero)
   * SD connected to pin 9 (Zero) or pin A6 (MKR1000, MKRZero)

 created 17 November 2016
 by Sandeep Mistry
 */

#include &lt;I2S.h&gt;

void setup() {
  // Open serial communications and wait for port to open:
  // A baud rate of 115200 is used instead of 9600 for a faster data rate
  // on non-native USB ports
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // start I2S at 16 kHz with 32-bits per sample
  if (!I2S.begin(I2S_PHILIPS_MODE, 16000, 32)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }
}

void loop() {
  // read a sample
  int sample = I2S.read();

  if ((sample == 0) || (sample == -1) ) {
    return;
  }
  // convert to 18 bit signed
  sample &gt;&gt;= 14;
```

```
  // if it's non-zero print value to serial
  Serial.println(sample);
}
```

Upload to your Arduino Zero/Feather wired up as above, and open up the **Serial Plotter**



Try blowing or whistling at the sensor to see response in real time



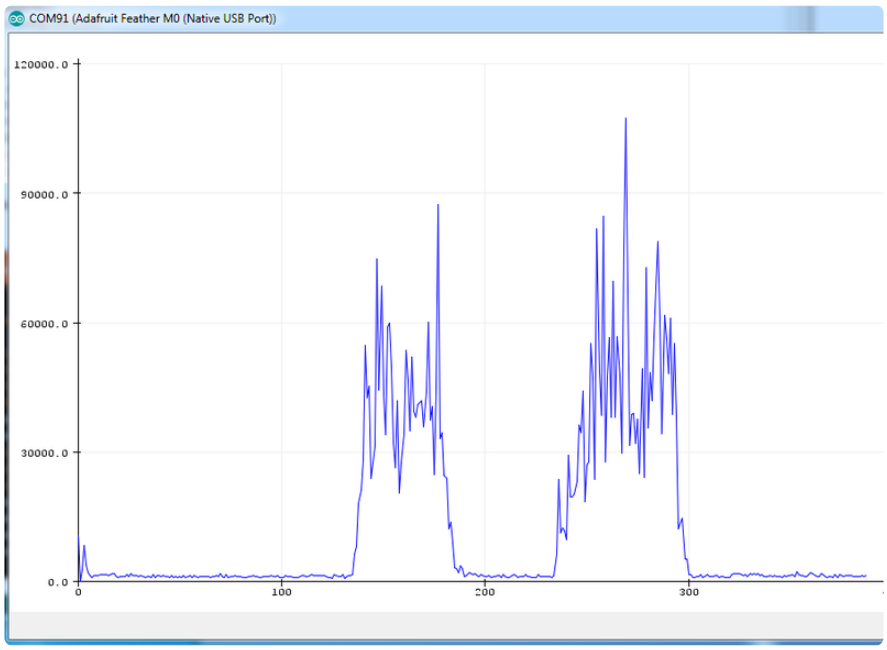
VU Meter Demo

Often times you don't want the actual audio data but the overall "sound pressure level". This example will take a bunch of samples, normalize the data to be around 0, then give you the maximum difference between the waveforms for a 'volume graph'

```
/*
 This example reads audio data from an Invensense's ICS43432 I2S microphone
 breakout board, and prints out the samples to the Serial console. The
 Serial Plotter built into the Arduino IDE can be used to plot the audio
 data (Tools -&gt; Serial Plotter)

 Circuit:
 * Arduino/Genuino Zero, MKRZero or MKR1000 board
 * ICS43432:
   * GND connected GND
   * 3.3V connected 3.3V (Zero) or VCC (MKR1000, MKRZero)
```

```
   * WS connected to pin 0 (Zero) or pin 3 (MKR1000, MKRZero)
   * CLK connected to pin 1 (Zero) or pin 2 (MKR1000, MKRZero)
   * SD connected to pin 9 (Zero) or pin A6 (MKR1000, MKRZero)

 created 17 November 2016
 by Sandeep Mistry
 */

#include &lt;I2S.h&gt;

void setup() {
  // Open serial communications and wait for port to open:
  // A baud rate of 115200 is used instead of 9600 for a faster data rate
  // on non-native USB ports
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // start I2S at 16 kHz with 32-bits per sample
  if (!I2S.begin(I2S_PHILIPS_MODE, 16000, 32)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }
}

#define SAMPLES 128 // make it a power of two for best DMA performance

void loop() {
  // read a bunch of samples:
  int samples[SAMPLES];

  for (int i=0; i&lt;SAMPLES; i++) {
    int sample = 0;
    while ((sample == 0) || (sample == -1) ) {
      sample = I2S.read();
    }
    // convert to 18 bit signed
    sample &gt;&gt;= 14;
    samples[i] = sample;
  }

  // ok we hvae the samples, get the mean (avg)
  float meanval = 0;
  for (int i=0; i&lt;SAMPLES; i++) {
    meanval += samples[i];
  }
  meanval /= SAMPLES;
  //Serial.print("# average: " ); Serial.println(meanval);

  // subtract it from all sapmles to get a 'normalized' output
  for (int i=0; i&lt;SAMPLES; i++) {
    samples[i] -= meanval;
    //Serial.println(samples[i]);
  }

  // find the 'peak to peak' max
  float maxsample, minsample;
  minsample = 100000;
  maxsample = -100000;
  for (int i=0; i&lt;SAMPLES; i++) {
    minsample = min(minsample, samples[i]);
    maxsample = max(maxsample, samples[i]);
  }
  Serial.println(maxsample - minsample);
}
```

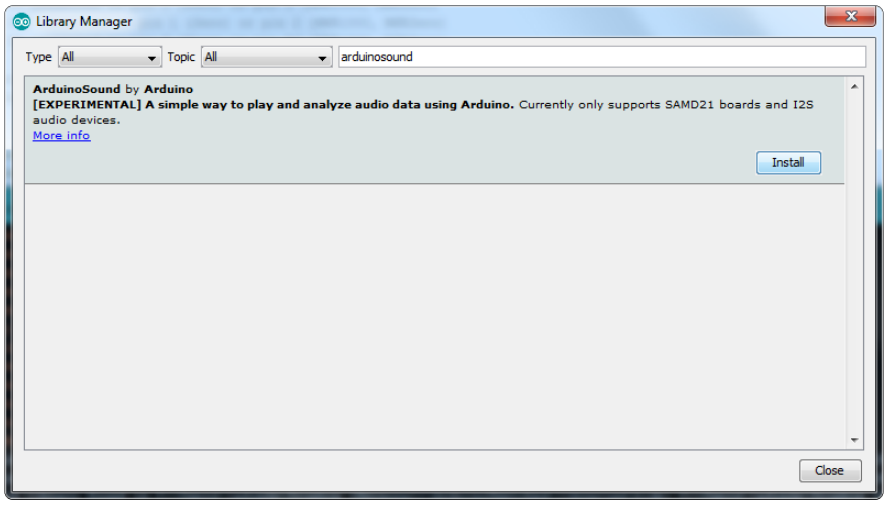
Open up the serial plotter to see how making noises will create peaks!



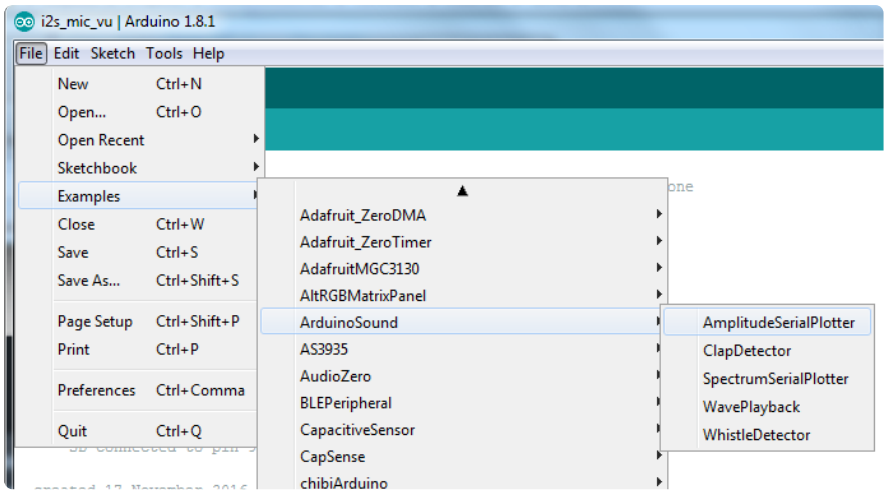
ArduinoSound Library

For most uses, its better to have a higher-level library for managing sound. The `ArduinoSound` library works with I2S mics and can do filtering, amplitude detection, etc!

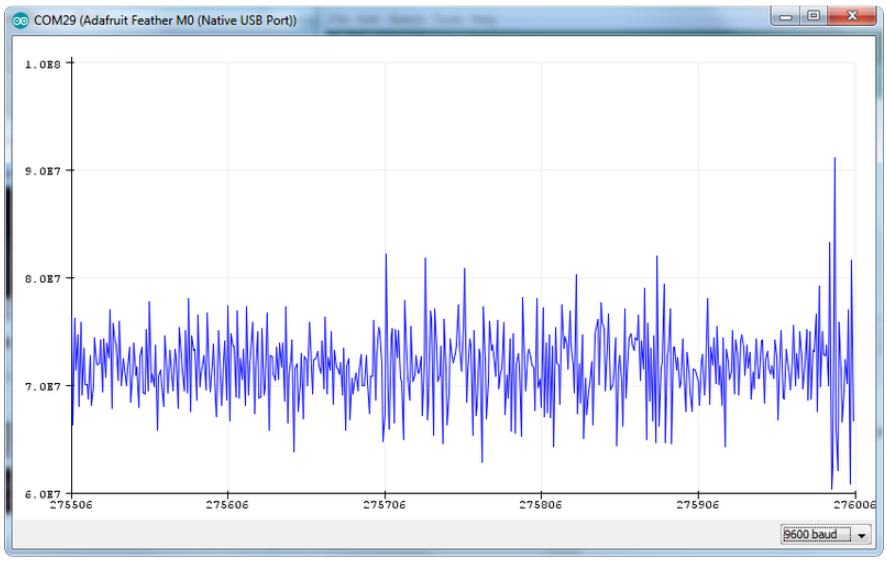
Install it using the Arduino library manager



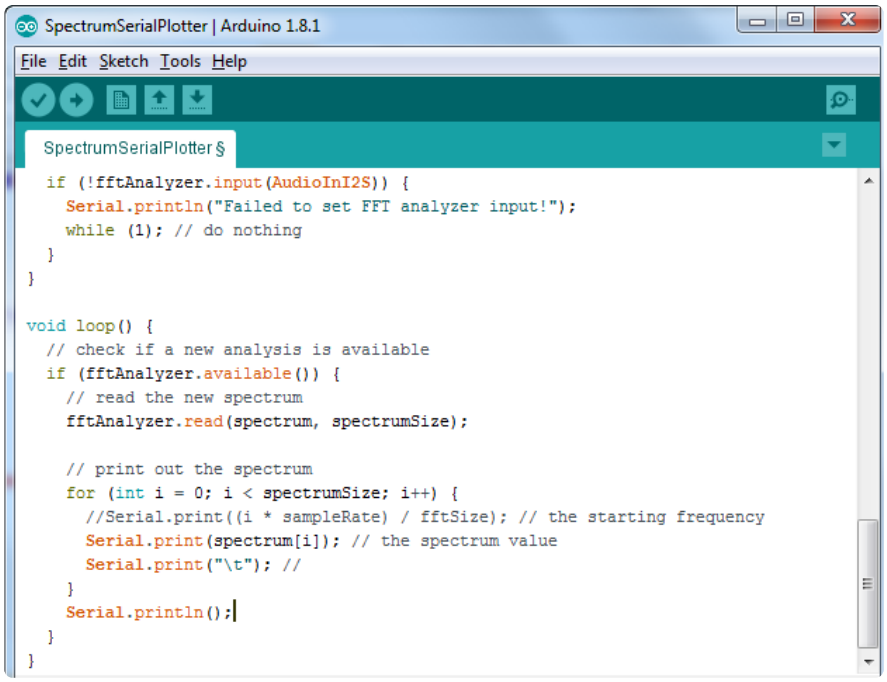
Various examples come with the library, check them out in the File->Examples->ArduinoSound sub menu

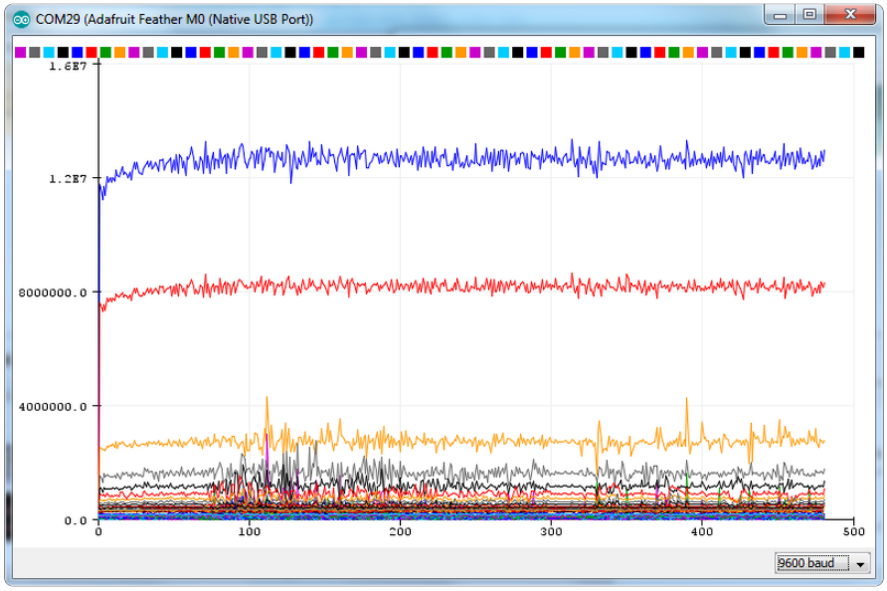


For example, amplitude Serial plotter will do basic amplitude plotting:



You can also do FFT spectral diagramming using SpectrumSerialPlotter. We made a small change to the example so that all 128 bins are plotted:





Raspberry Pi Wiring & Test

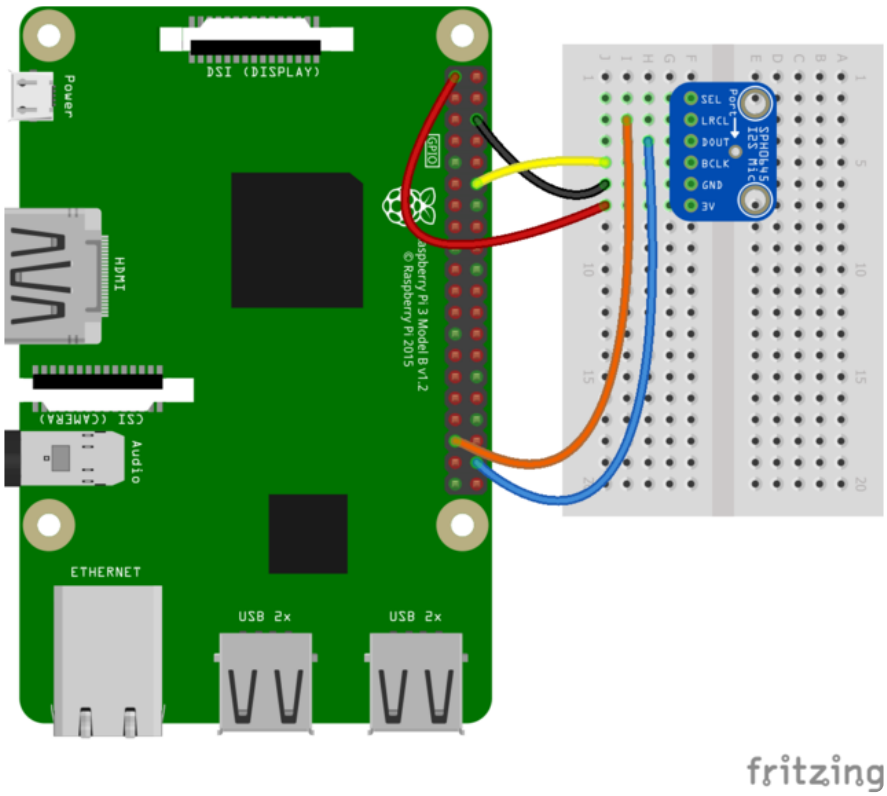
The driver for the microphone works with Raspberry Pi OS Bookworm and Bullseye. This has been tested with both 32-bit and 64-bit Lite releases.

You can add mono or stereo I2S microphones to your Raspberry Pi, too! This will work with any Raspberry with a 2x20 GPIO connector.

This guide is largely based on this great git repo (https://adafru.it/KBh), which we forked for use here. Also useful was the information provided in this forum post (https://adafru.it/fEw).

Wiring For Mono Mic

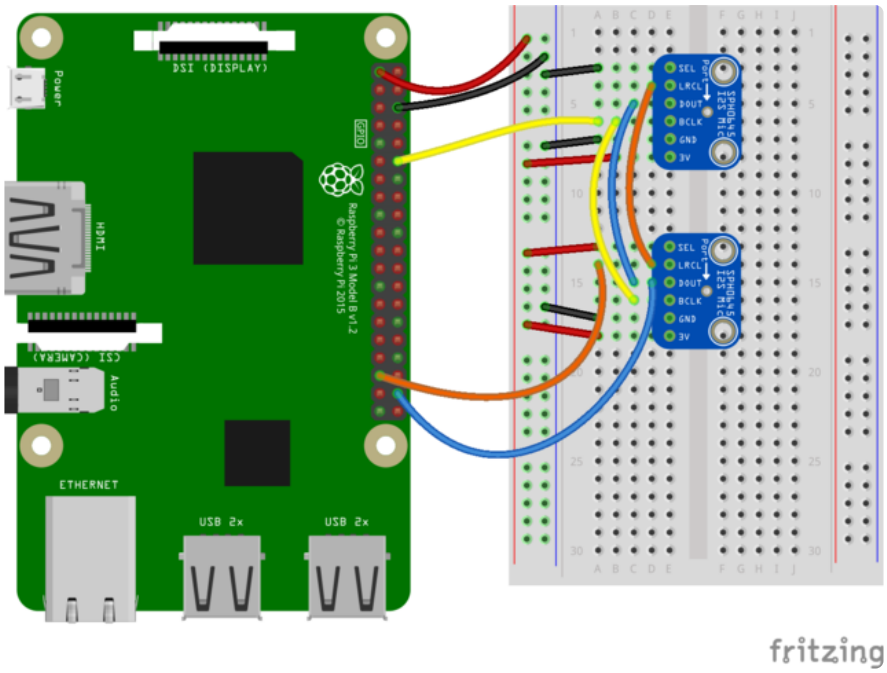
- Mic 3V to Pi 3.3V
- Mic GND to Pi GND
- Mic SEL to Pi GND (this is used for channel selection, connect to either 3.3V or GND)
- Mic BCLK to BCM 18 (pin 12)
- Mic DOUT to BCM 20 (pin 38)
- Mic LRCL to BCM 19 (pin 35)



Wiring For Stereo Mic

Connect both mics as above except for the SEL pin.

- Left Mic SEL to Pi GND
- Right Mic SEL to Pi 3.3V



Install Raspbian on an SD Card

You'll need to start with Raspbian or Raspbian Lite. Get the latest version from the Raspberry Pi Download page (https://adafru.it/fQi) and follow these instructions to install the OS image to the SD card (https://adafru.it/jd0).

Update config.txt

The following line needs to be added to config.txt.

```
dtoverlay=googlevoicehat-soundcard
```

Starting with the PiOS Bookworm config.txt is located in:

```
/boot/firmware/config.txt
```

Earlier PiOS releases such as Bullseye have the file located in:

```
/boot/config.txt
```

That is it! reboot your Pi.

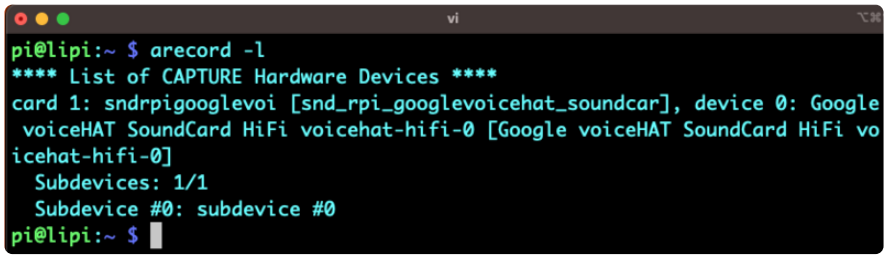
```
sudo reboot
```

Test & Record!

Use the following command to list the available input devices:

```
arecord -l
```

You should see a card entry with information similar to this:



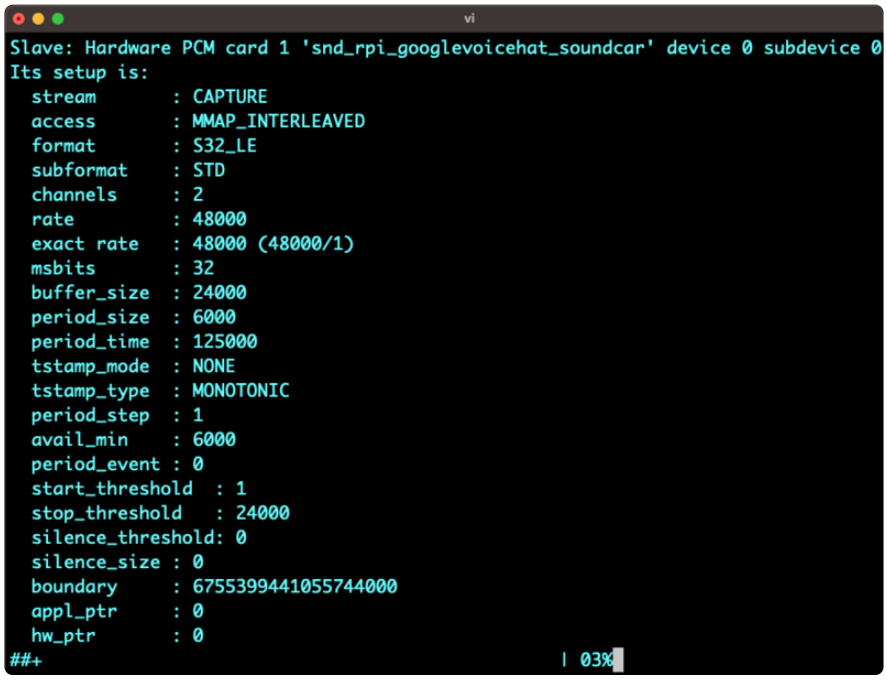
Note the card number. In the screen shot above it is `1`. You can record a wav file in mono with this command (change the `-plughw` parameter to match the card number from above):

```
arecord -D plughw:1 -c1 -r 48000 -f S32_LE -t wav -V mono -v file.wav
```

Or, if you have two i2s mics installed, record in stereo with this command:

```
arecord -D plughw:1 -c2 -r 48000 -f S32_LE -t wav -V stereo -v file_stereo.wav
```

If all is working correctly, you should see the VU meter react at the bottom of the terminal window:



Test Playback

If you have speakers hooked up to the Pi, you can play the file back directly on the device:

```
aplay file.wav
```

Or, you can copy it over to your computer for playback :), make sure SSH is enabled and then:

```
scp pi@raspberrypi:file.wav ~/Desktop/file.wav
```

Volume Control is not supported with the googlevoicehat-soundcard.

High Frequency Recording

The googlevoicehat-soundcard module is great in that it comes precompiled and just works. This is what most people should use. It will be at a fixed 48 kHz.

If you want to record at higher frequencies checkout the rpi-i2s-audio overlay (https://adafru.it/1a9d). There is a report (https://adafru.it/1a9e) of successful recording up to 96 kHz.

tip credit: forum user @grondeau

Adding Volume Control

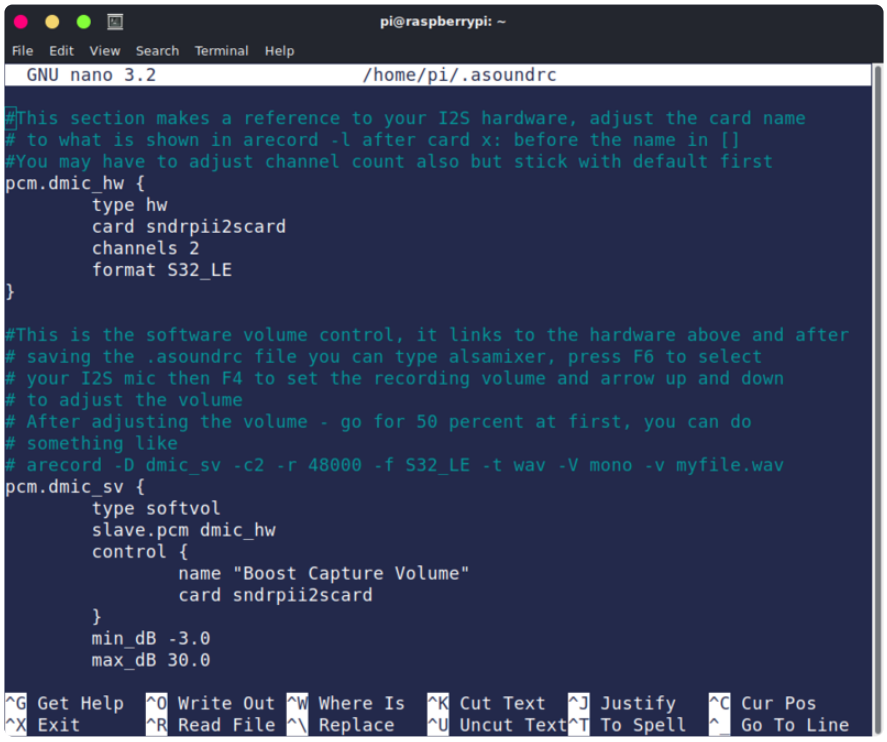
You can add volume control to your mic via **alsamixer** and alsa config. (Hat tip to RickTracer (https://adafru.it/doW)). To do so, create and edit a file **.asoundrc** in your home directory.

```
nano ~/.asoundrc
```

and put the following in:

```
#This section makes a reference to your I2S hardware, adjust the card name
# to what is shown in arecord -l after card x: before the name in []
#You may have to adjust channel count also but stick with default first
pcm.dmic_hw {
    type hw
    card sndrpii2scard
    channels 2
    format S32_LE
}

#This is the software volume control, it links to the hardware above and after
# saving the .asoundrc file you can type alsamixer, press F6 to select
# your I2S mic then F4 to set the recording volume and arrow up and down
# to adjust the volume
# After adjusting the volume - go for 50 percent at first, you can do
# something like
```

```
# arecord -D dmic_sv -c2 -r 48000 -f S32_LE -t wav -V mono -v myfile.wav
pcm.dmic_sv {
    type softvol
    slave.pcm dmic_hw
    control {
        name "Boost Capture Volume"
        card sndrpii2scard
    }
    min_dB -3.0
    max_dB 30.0
}
```



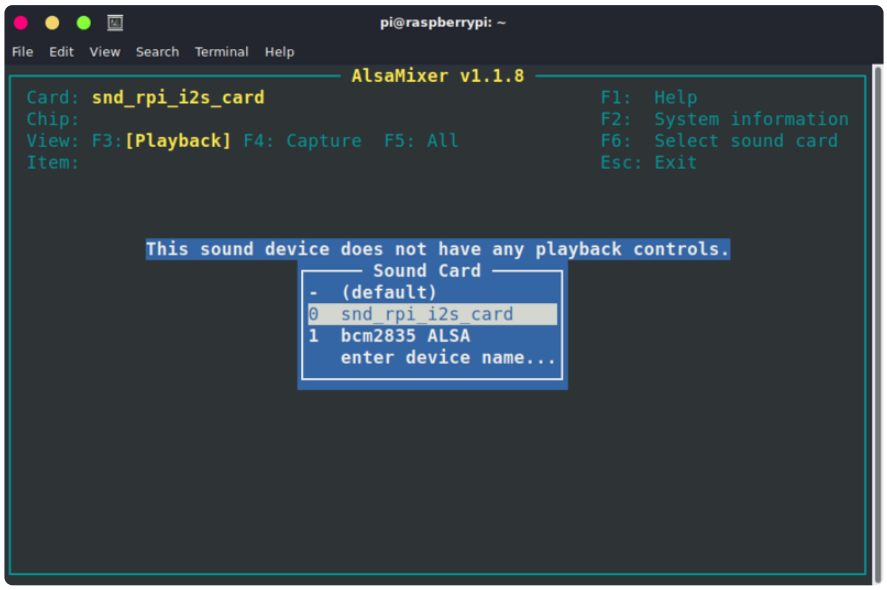
Now before you can change the volume you need to use the device once (this is an alsa thing)

Run:

```
arecord -D dmic_sv -c2 -r 44100 -f S32_LE -t wav -V mono -v file.wav
```

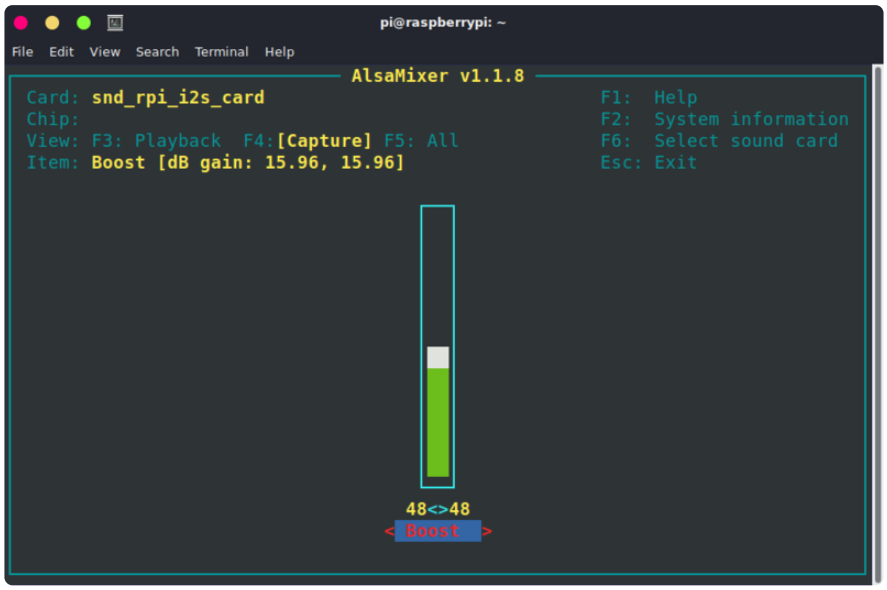
And cancel with ^C once it starts recording.

Now you can run **alsamixer** - press **F6** and select the I2S sound card



It will complain there are no playback controls (because its for recording only).

Press **F4** to switch to **Capture** mode and you should be able to adjust the volume with up/down arrow keys.



Then you can record with the i2c mic device using:

```
arecord -D dmic_sv -c2 -r 48000 -f S32_LE -t wav -V mono -v recording.wav
```

and playback with:

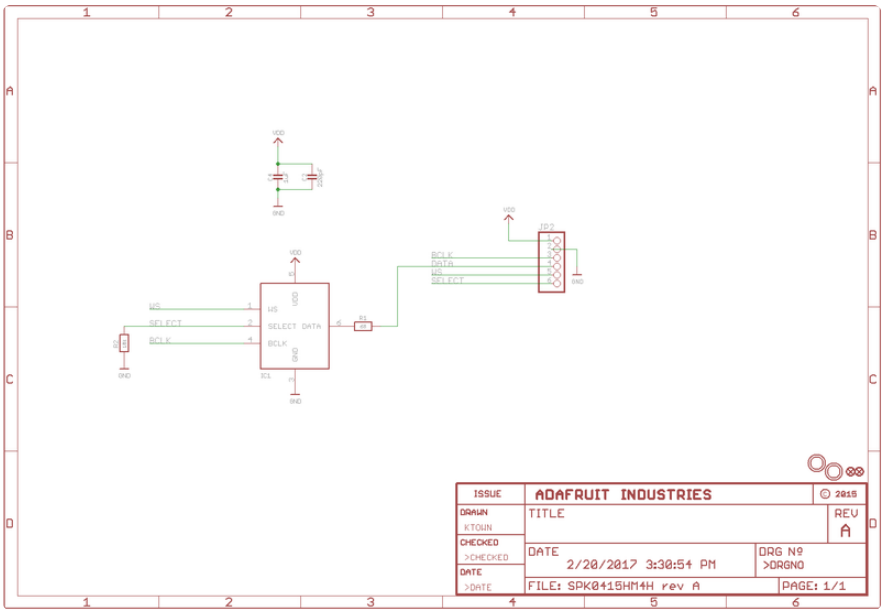
```
aplay recording.wav
```

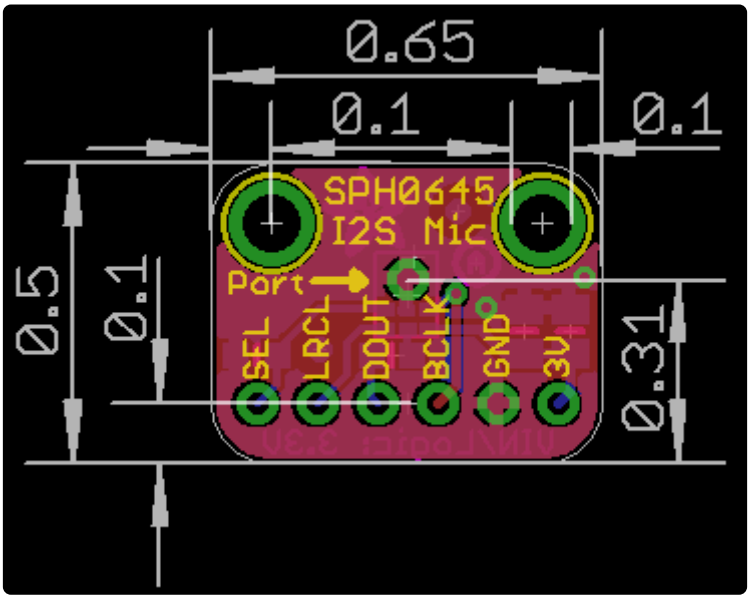
Downloads

Files

- SPH0645 EagleCAD PCB Files on GitHub (https://adafru.it/uyb)
- ICS-43434 EagleCAD PCB Files on GitHub (https://adafru.it/1a7S)
- Fritzing object in the Adafruit Fritzing library (https://adafru.it/aP3)

Schematic & Fab Print - SPH0645





Schematic and Fab Print ICS-43434