



Adafruit I2S MEMS Microphone Breakout

Created by lady ada



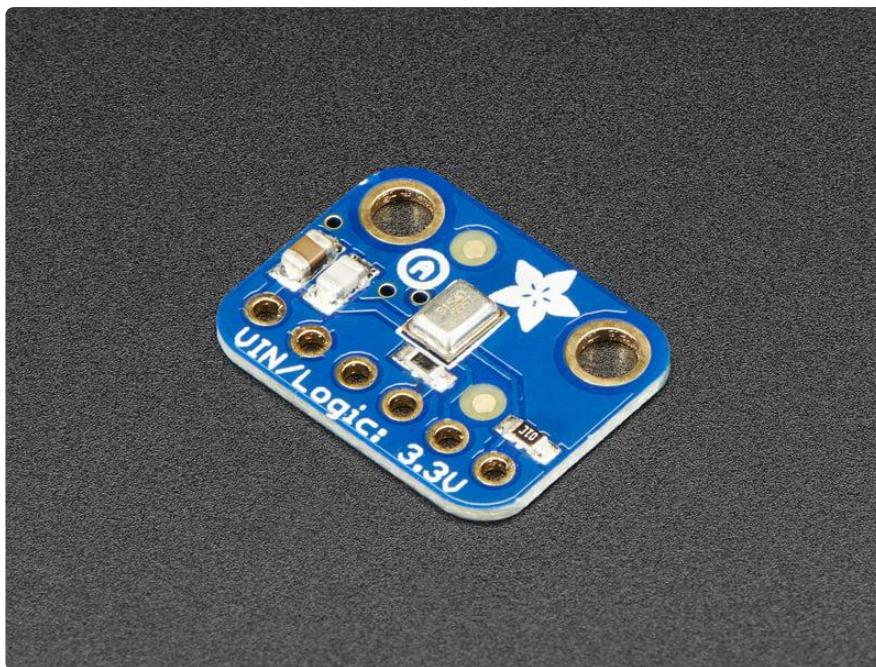
<https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout>

Last updated on 2023-08-29 03:19:54 PM EDT

Table of Contents

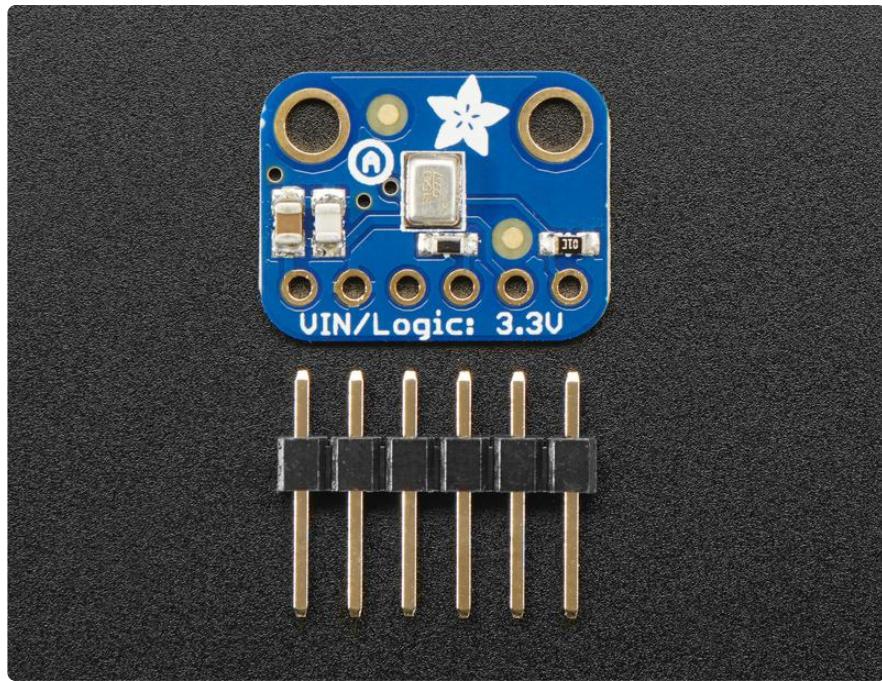
Overview	3
Assembly	5
• Prepare the header strip:	
• Add the breakout board:	
• And Solder!	
Pinouts	6
• Power Pins	
• I2S Data Pins	
Arduino Wiring & Test	8
• Wiring	
• I2S Library	
• VU Meter Demo	
• ArduinoSound Library	
Raspberry Pi Wiring & Test	14
• Wiring For Mono Mic	
• Wiring For Stereo Mic	
• Install Raspbian on an SD Card	
• Update the Pi	
• Python and Pip	
• Installer Script	
• Run the Script	
• Test & Record!	
• Test Playback	
• Manual Load	
• Rebuilding After System Update	
• Adding Volume Control	
Old Kernel Install Method	23
• Wiring For Mono Mic	
• Wiring For Stereo Mic	
• Raspberry Pi i2s Configuration	
• Kernel Compiling	
• Prepare to Compile the i2s module	
• Auto-load the module on startup	
• Test & Record!	
• Test Playback	
• Adding Volume control	
Downloads	33
• Files	
• Schematic & Fab Print	

Overview

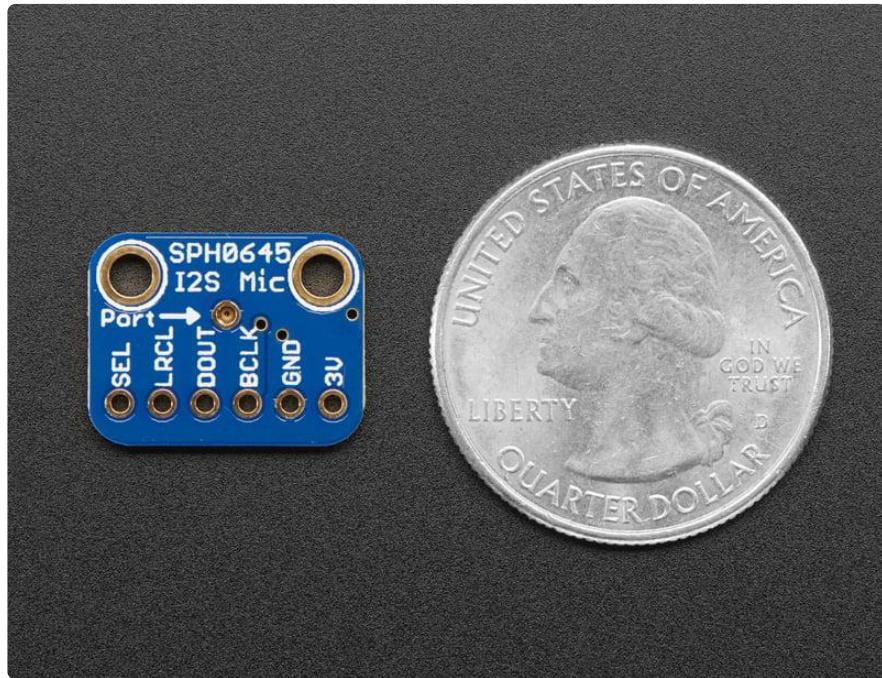


For many microcontrollers, [adding audio input is easy with one of our analog microphone breakouts](#) (<http://adafru.it/1063>). But as you get to bigger and better microcontrollers and microcomputers, you'll find that you don't always have an analog input, or maybe you want to avoid the noise that 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.



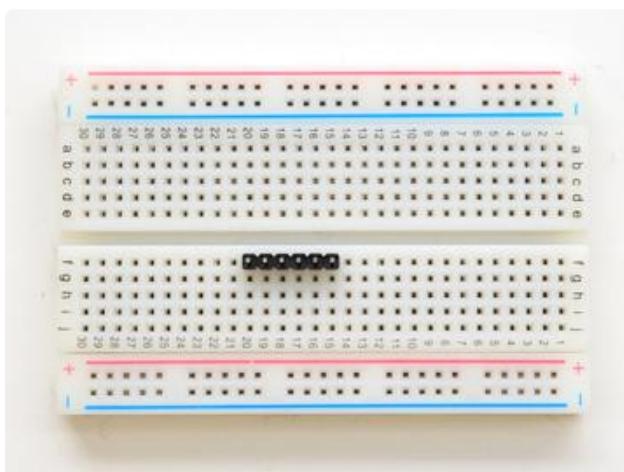
This I2S MEMS microphone is bottom ported, so make sure you have the hole in the bottom facing out towards the sounds you want to read. It's a 1.6-3.3V device only, so not for use with 5V logic (it's really unlikely you'd have a 5V-logic device with I2S anyways). Many beginner microcontroller boards 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

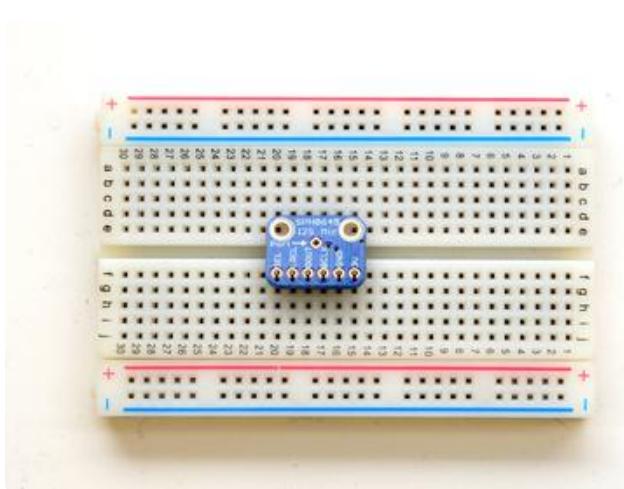
Assembly is really easy, you can use straight or 'right-angle' style headers to attach to the PCB. We'll be using the plain straight headers included

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:

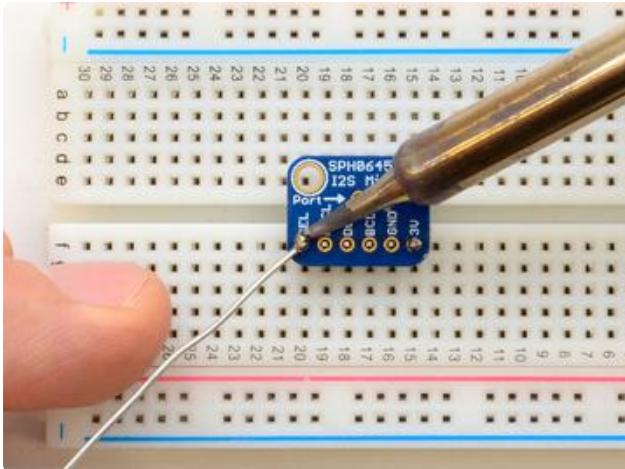
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down



Add the breakout board:

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

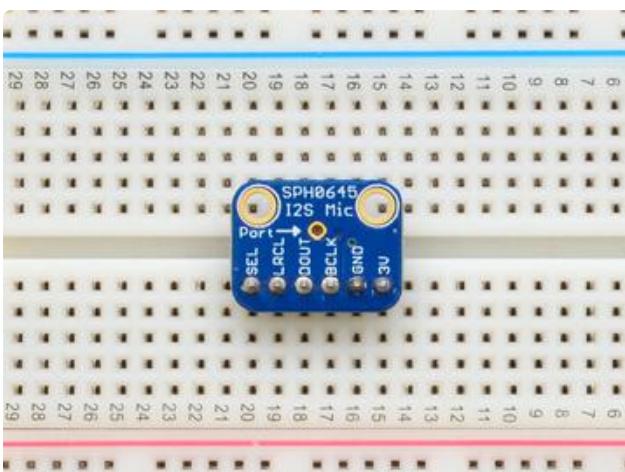
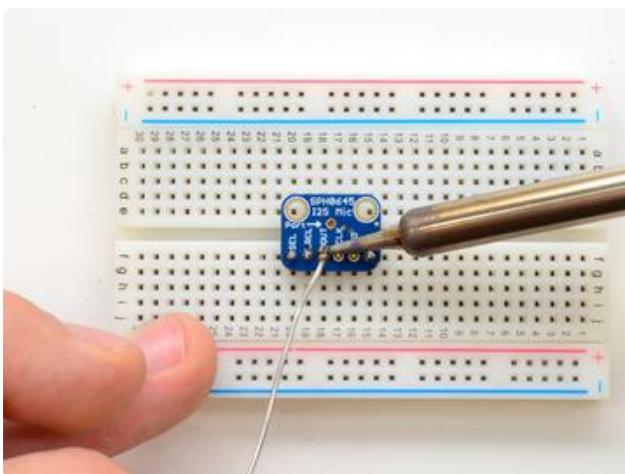
Make sure the side with the components is face down, as shown in the photos in this guide!



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 \(\)](#)).



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

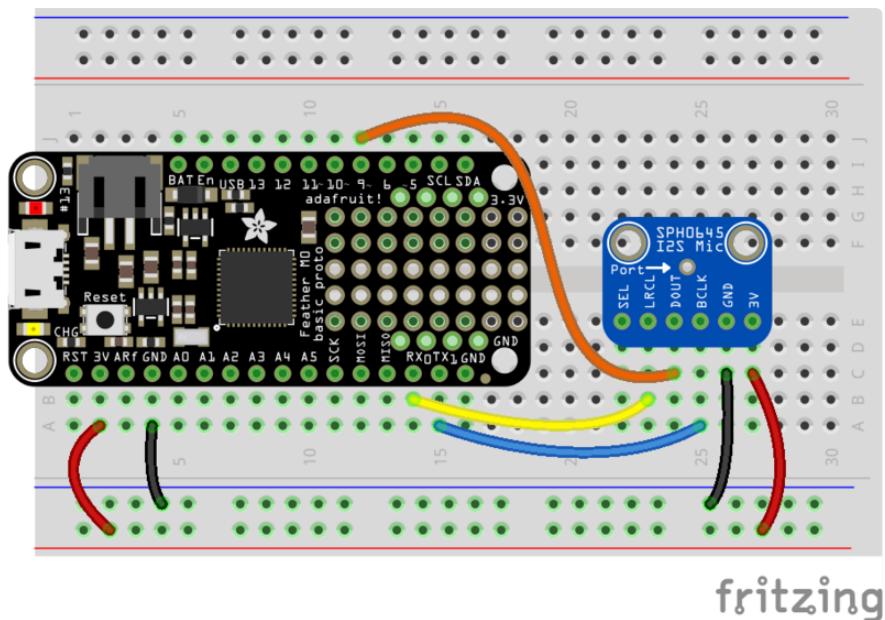
Remember, the I2S microphone requires an I2S peripheral and won't work with chips that don't support it in hardware! For this example we'll use a Feather M0, but you can also use an Arduino Zero.

Wiring

For Feather M0, Arduuino 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

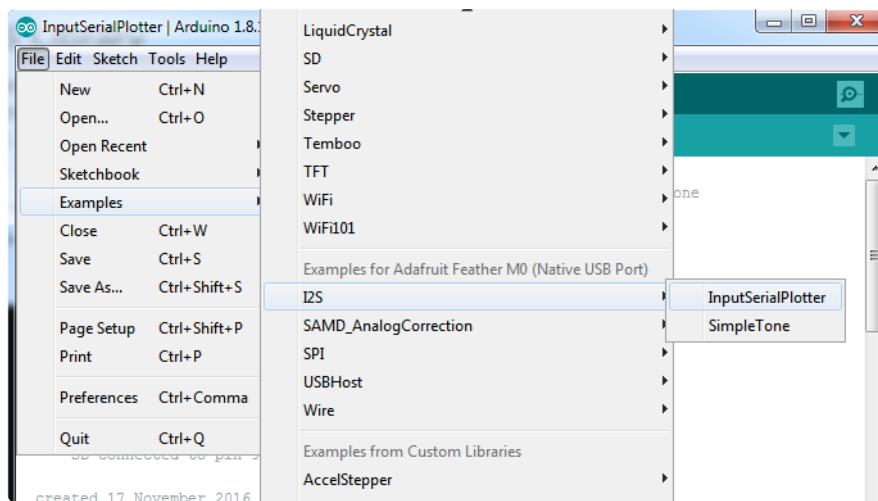


i2sfeatherm0.fzz

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 I2S 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 -> 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 <I2S.h>

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;
  }
}
```

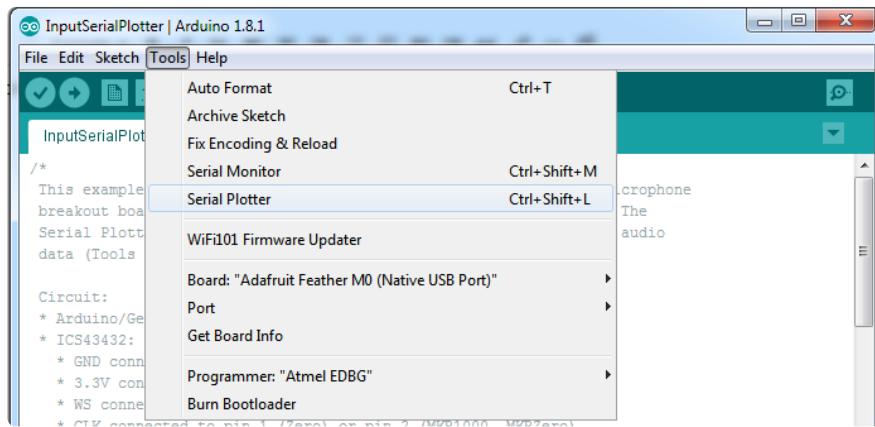
```

}
// convert to 18 bit signed
sample &gt;&gt;= 14;

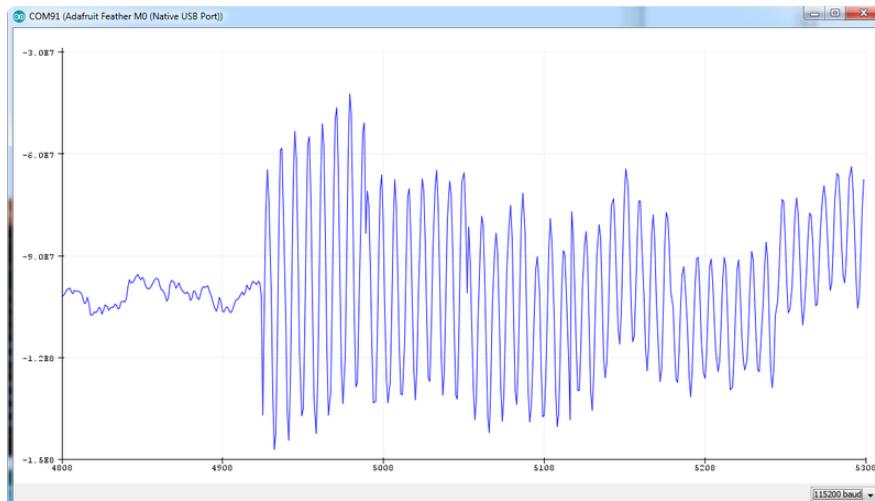
// if it's non-zero print value to serial
Serial.println(sample);
}

```

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 -> 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 <I2S.h>

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

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

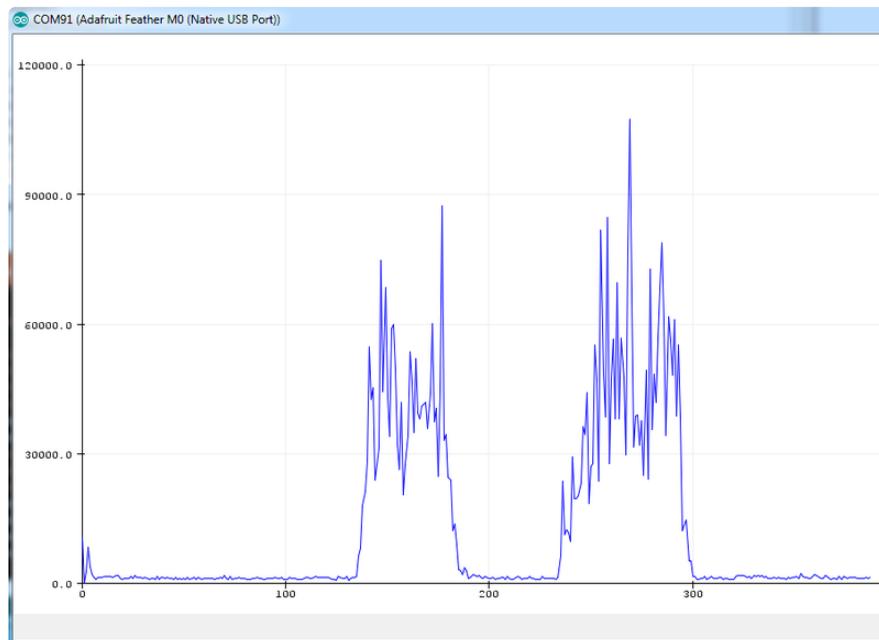
    // subtract it from all samples to get a 'normalized' output
    for (int i=0; i<SAMPLES; i++) {
        samples[i] -= meanval;
        //Serial.println(samples[i]);
    }

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

```

```
    }
    Serial.println(maxsample - minsample);
}
```

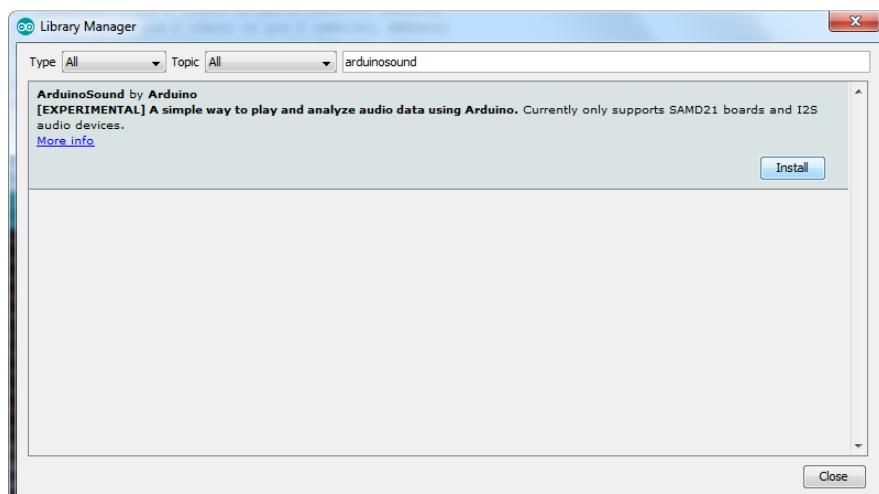
Open up the serial plotter to see how making noises will create peaks!



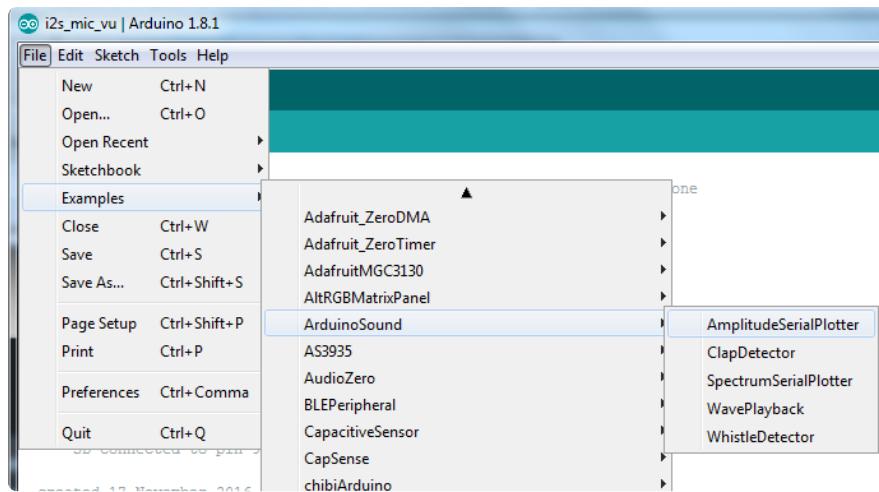
ArduinoSound Library

For most uses, it's better to have a higher-level library for managing sound. The [ArduinoSound](#) library works with I2S mics and can do filtering, amplitude detection, etc!

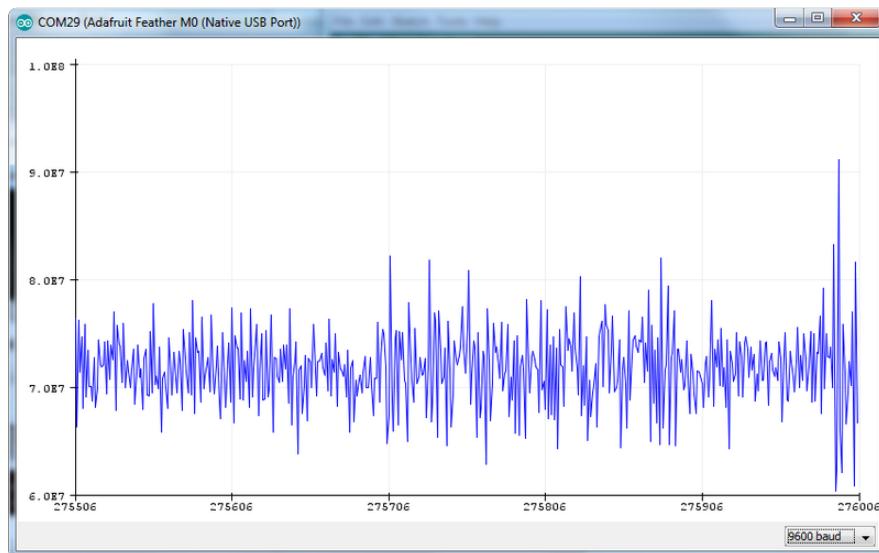
Install it using the Arduino library manager



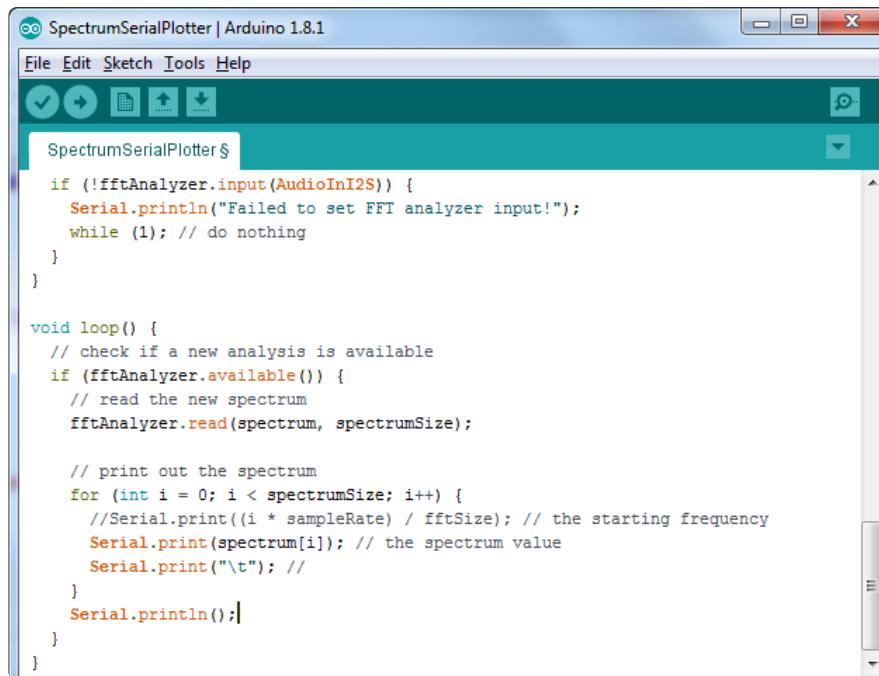
Various examples come with the library, check them out in the File->Examples->ArduinoSound sub menu



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:

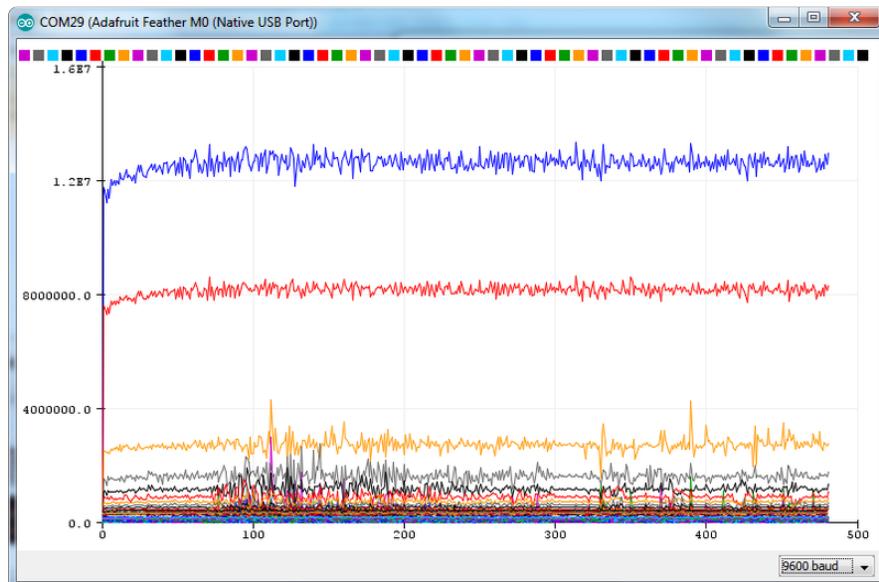


The screenshot shows the Arduino IDE interface with the title bar "SpectrumSerialPlotter | Arduino 1.8.1". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for back, forward, upload, and download. The main code editor window contains the following C++ code:

```
if (!fftAnalyzer.input(AudioInI2S)) {
    Serial.println("Failed to set FFT analyzer input!");
    while (1); // do nothing
}

void loop() {
    // check if a new analysis is available
    if (fftAnalyzer.available()) {
        // read the new spectrum
        fftAnalyzer.read(spectrum, spectrumSize);

        // print out the spectrum
        for (int i = 0; i < spectrumSize; i++) {
            //Serial.print((i * sampleRate) / fftSize); // the starting frequency
            Serial.print(spectrum[i]); // the spectrum value
            Serial.print("\t");
        }
        Serial.println();
    }
}
```



Raspberry Pi Wiring & Test

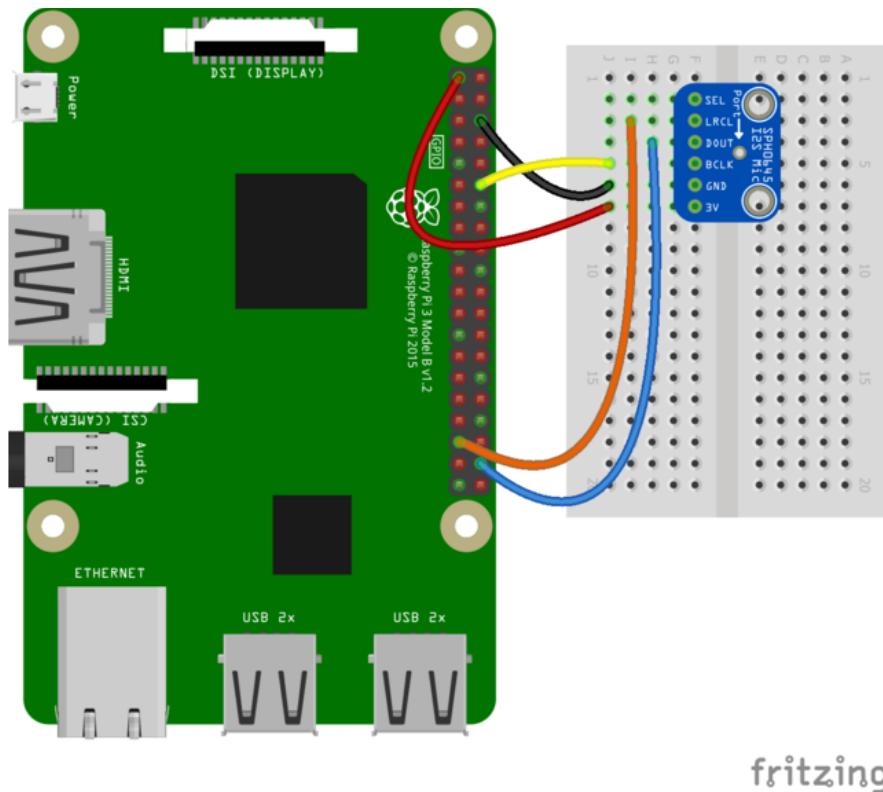
You can add mono or stereo I2S microphones to your Raspberry Pi, too! This will work with any Raspberry with a 2x20 GPIO connector.

This guide is largely based on this [great git repo \(\)](#), which we forked for use here. Also useful was the information provided in this [forum post \(\)](#).

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)

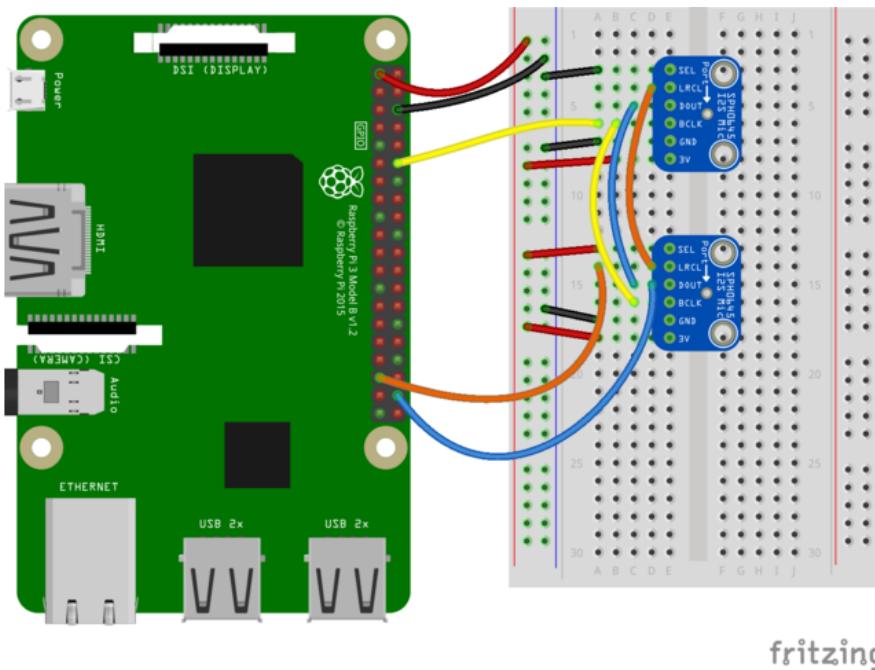


fritzing

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



fritzing

Be sure to follow all these steps in the order shown to insure the I2S mic module is built against the proper kernel headers.

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 \(\)](#) and follow [these instructions to install the OS image to the SD card \(\)](#).

Update the Pi

Run these commands:

```
sudo apt-get -y update
sudo apt-get -y upgrade
```

and then REBOOT THE PI.

Python and Pip

Run the following command to make sure you have Python 3 and Pip installed:

```
sudo apt install python3-pip
```

Installer Script

This script will do all the work for you. Here are the commands to run. Make sure your Pi has network access, it needs to download the software!

```
cd ~  
sudo pip3 install --upgrade adafruit-python-shell  
wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/  
master/i2smic.py  
sudo python3 i2smic.py
```

```
pi@raspberrypi:~ $ cd ~  
pi@raspberrypi:~ $ sudo pip3 install --upgrade adafruit-python-shell  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting adafruit-python-shell  
  Downloading https://www.piwheels.org/simple/adafruit-python-shell/adafruit_python_shell-1.1.0  
-py3-none-any.whl  
Requirement already satisfied, skipping upgrade: clint in /usr/local/lib/python3.7/dist-packages  
(from adafruit-python-shell) (0.5.1)  
Requirement already satisfied, skipping upgrade: Adafruit-PlatformDetect in /usr/local/lib/python3.7/dist-packages  
(from adafruit-python-shell) (2.17.0)  
Requirement already satisfied, skipping upgrade: args in /usr/local/lib/python3.7/dist-packages  
(from clint->adafruit-python-shell) (0.1.0)  
Installing collected packages: adafruit-python-shell  
  Found existing installation: adafruit-python-shell 1.0.1  
    Uninstalling adafruit-python-shell-1.0.1:  
      Successfully uninstalled adafruit-python-shell-1.0.1  
Successfully installed adafruit-python-shell-1.1.0  
pi@raspberrypi:~ $ wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scri  
pts/master/i2smic.py  
--2020-09-21 16:16:06-- https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scri  
pts/master/i2smic.py  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.68.133  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.68.133|:443... conn  
ected.  
HTTP request sent, awaiting response... 200 OK  
Length: 1741 (1.7K) [text/plain]  
Saving to: 'i2smic.py.2'  
  
i2smic.py.2          100%[=====]  1.70K  --.-KB/s   in 0.001s  
  
2020-09-21 16:16:07 (2.88 MB/s) - ‘i2smic.py.2’ saved [1741/1741]  
pi@raspberrypi:~ $ sudo python3 i2smic.py
```

Run the Script

Once you run the script, you will be presented with options for configuration.

This script downloads and installs
I2S microphone support.

RASPBERRY_PI_3B_PLUS detected.

Auto load module at boot? [y/n] █

Auto Load at Boot

The Pi model should be automatically
detected.

If you want the I2S mic module support to
be loaded at boot, select Yes here.
Otherwise you'll have to manually install
the module each time you want to use it.

```

raspberrypi-kernel-headers is already the newest version (1.20200902-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
fatal: destination path 'Raspberry-Pi-Installer-Scripts' already exists and is not an empty directory.

make -C /lib/modules/5.4.51-v7+/build M=/home/pi/Raspberry-Pi-Installer-Scripts/i2s_mic_
module clean
make[1]: Entering directory '/usr/src/linux-headers-5.4.51-v7+'
make[1]: Leaving directory '/usr/src/linux-headers-5.4.51-v7+'
make -C /lib/modules/5.4.51-v7+/build M=/home/pi/Raspberry-Pi-Installer-Scripts/i2s_mic_
module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.51-v7+'
CC [M] /home/pi/Raspberry-Pi-Installer-Scripts/i2s_mic_module/snd-i2smic-rpi.mod.o
LD [M] /home/pi/Raspberry-Pi-Installer-Scripts/i2s_mic_module/snd-i2smic-rpi.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.51-v7+'
rm -f /lib/modules/5.4.51-v7+/kernel/sound/drivers/snd-i2smic-rpi.ko
install -m644 -b -D snd-i2smic-rpi.ko /lib/modules/5.4.51-v7+/kernel/sound/drivers/snd-i2smic-rpi.ko
depmod -a
DONE.

Settings take effect on next boot.

REBOOT NOW? [y/N]

```

Reboot

Since the script runs a system update and installs some other needed support software, it may take a while to run. Just let it go. When it's finished, you should be prompted with an option to reboot.

You need to reboot for the settings to take effect.

Test & Record!

OK, that probably took a bit of time, but now you are ready to rock!

Use the following command to list the available input devices:

```
arecord -l
```

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

```

pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: snd_rpi_i2s_card [snd_rpi_i2s_card], device 0: simple-card_codec_link snd_soc-dummy-dai-0 [simple-card_codec_link snd_soc-dummy-dai-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
pi@raspberrypi:~ $

```

Note the card number. In the screen shot above it is `0`. 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:0 -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:0 -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:

```
pi@raspberrypi: ~
@raspberrypi:~ $ arecord -D plughw:0 -c2 -r 48000 -f S32_LE -t wav -V stereo -v
file_stereo.wav
Recording WAVE 'file_stereo.wav' : Signed 32 bit Little Endian, Rate 48000 Hz, Stereo
Plug PCM: Hardware PCM card 0 'snd_rpi_i2s_card' device 0 subdevice 0
Its setup is:
stream      : CAPTURE
access      : RW_INTERLEAVED
format      : S32_LE
subformat   : STD
channels    : 2
rate        : 48000
exact rate  : 48000 (48000/1)
msbits      : 32
buffer_size : 24000
period_size : 6000
period_time : 125000
tstamp_mode : NONE
tstamp_type : MONOTONIC
period_step : 1
avail_min   : 6000
period_event: 0
start_threshold : 1
stop_threshold  : 24000
silence_threshold: 0
silence_size : 0
boundary     : 1572864000
appl_ptr     : 0
hw_ptr       : 0
+
##### 35%|36%##### +
```

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
```

Manual Load

If you chose not to auto load the module at boot, then you will need to manually load it with the following command:

```
sudo modprobe snd-i2smic-rpi rpi_platform_generation=PI_SEL
```

Replace **PI_SEL** with the appropriate value for your Pi Model:

- 0 = Pi Zero or Zero W
- 1 = Pi 2 or 3
- 2 = Pi 4

You can use the `arecord -l` command to verify it loaded. Example on a Pi 3:



```
pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
pi@raspberrypi:~ $ sudo modprobe snd-i2smic-rpi rpi_platform_generation=1
pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: sndrpipi2scard [snd_rpi_i2s_card], device 0: simple-card_codec_link snd-soc-dummy-dai-0 [simple-card_codec_link snd-soc-dummy-dai-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi:~ $
```

Rebuilding After System Update

If you later run a system update and find that the I2S mic no longer works, you may need to rebuild and reinstall the driver. To do so, simply run the install script again:

```
sudo ./i2smic.sh
```

It may complain about things, like git, already being installed. But you can generally ignore those error messages.

If you've deleted the repo directory since the original install, then just reclone it.

Adding Volume Control

You can add volume control to your mic via alsamixer and alsoundrc config. ([Hat tip to RickTracer \(\)](#)). To do so, create and edit a file .alsoundrc in your home directory.

```
nano ~/.alsoundrc
```

and put the following in:

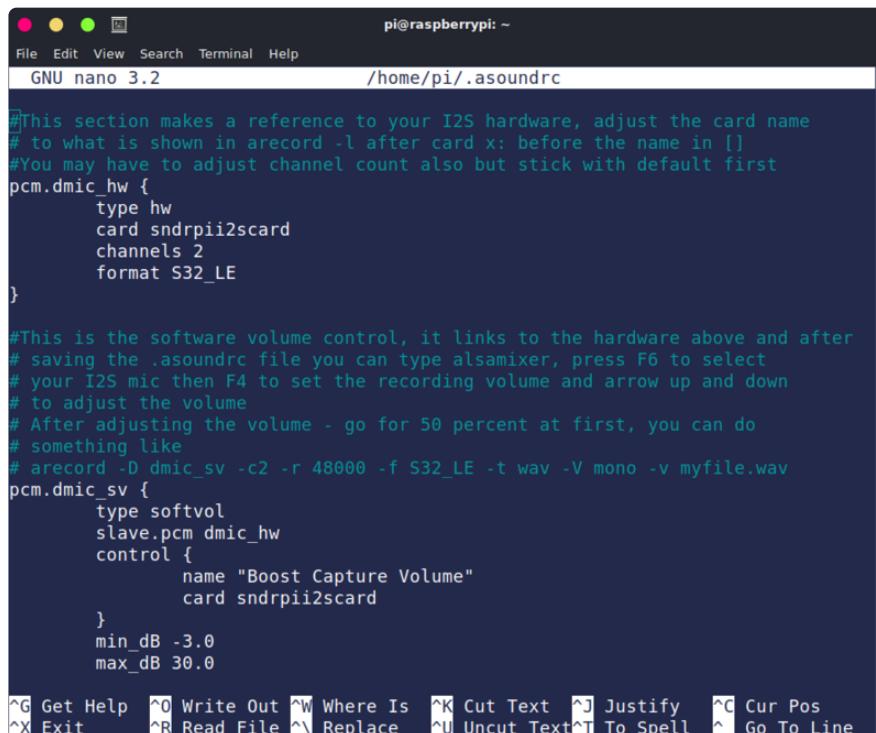
```
#This section makes a reference to your I2S hardware, adjust the card name
# to what is shown in arecord -l after card x: before the name in []
#You may have to adjust channel count also but stick with default first
pcm.dmic_hw {
    type hw
    card sndrpipi2scard
    channels 2
    format S32_LE
}

#This is the software volume control, it links to the hardware above and after
# saving the .alsoundrc 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 sndrpipi2scard
}
min_dB -3.0
max_dB 30.0
}

```



```

pi@raspberrypi: ~
File Edit View Search Terminal Help
GNU nano 3.2          /home/pi/.asoundrc

#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 sndrpipi2scard
    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 sndrpipi2scard
    }
    min_dB -3.0
    max_dB 30.0
}

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit  ^R Read File  ^Y Replace  ^U Uncut Text  ^T To Spell  ^L Go To Line

```

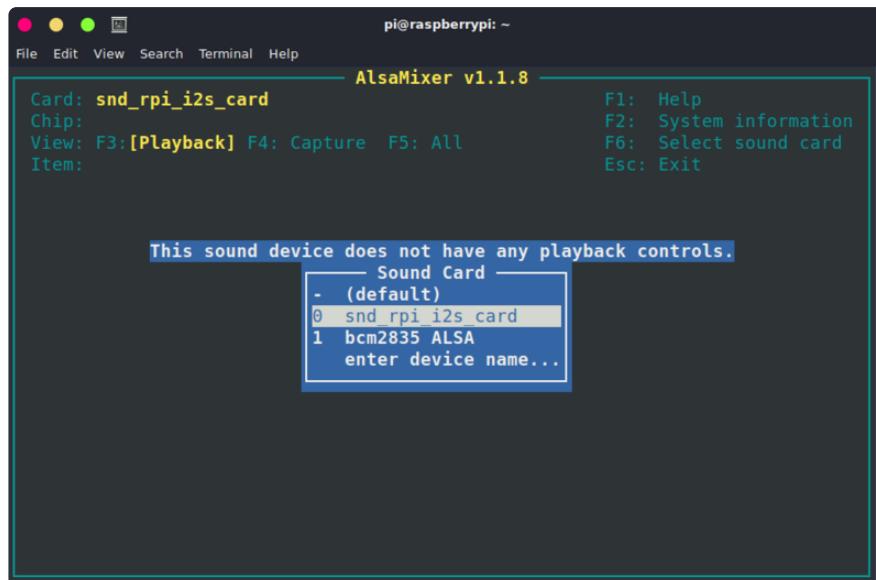
Now before you can change the volume you need to use the device once (this is an alsamixer thing)

Run:

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

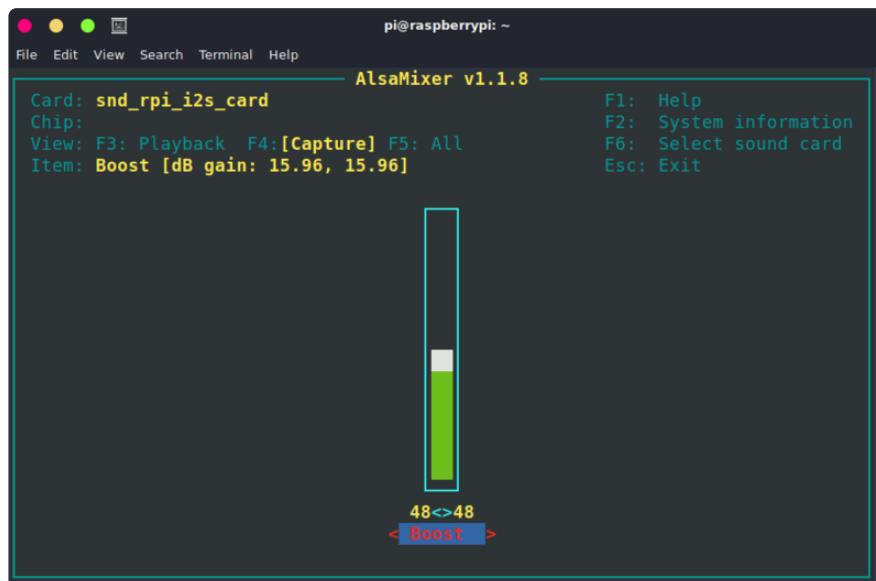
And cancel with ^C once it starts recording.

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
```

Old Kernel Install Method

This technique is not suggested, see previous page for a better working method!!!

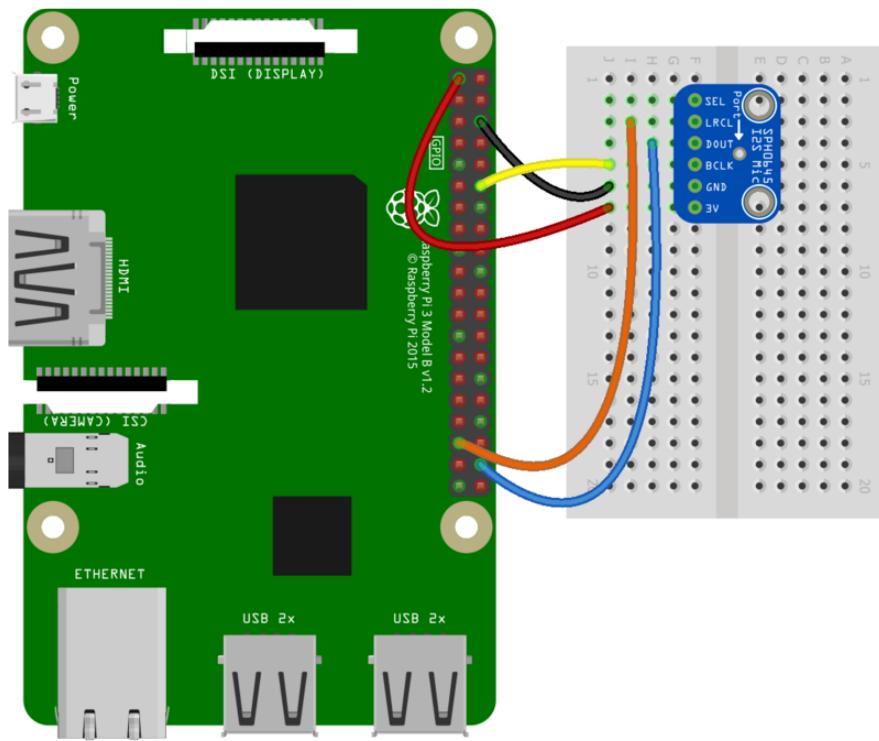
You can add mono or stereo I2S microphones to your Raspberry Pi, too!

This will work with Raspberry Pi B+, 2, 3, Zero and any other 2x20-connector-Pi

This guide is largely based on this great git repo [https://github.com/nejohnson2/rpi-i2s \(\)](https://github.com/nejohnson2/rpi-i2s)

Wiring For Mono Mic

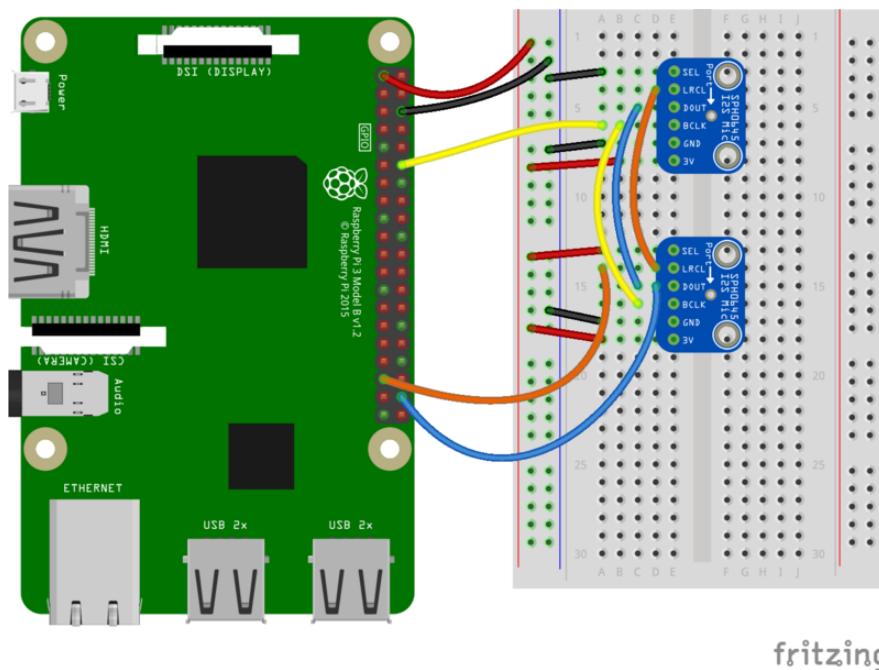
- Mic 3V - Pi 3.3v
- Mic Gnd - Pi Gnd
- Mic SEL - Pi Gnd (this is used for channel selection. Connect to 3.3 or GND)
- Mic BCLK - BCM 18 (pin 12)
- Mic LRCL - BCM 19 (pin 35)
- Mic DOUT - BCM 20 (pin 38)



fritzing

Pi i2s Mono Fritzing File

Wiring For Stereo Mic



Pi i2s Stereo Fritzing File

Raspberry Pi i2s Configuration

Start by logging into your Raspberry Pi via a terminal, we recommend ssh so you can copy + paste the many commands.

Turn on i2s support by editing /boot/config.txt with:

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

Uncomment #dtparam=i2s=on

A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the contents of the "/boot/config.txt" file. The file contains configuration parameters for the kernel, including comments about enabling hardware interfaces like I2C, SPI, and the LIRC module. It also includes a section for audio support with the "snd_bcm2835" module.

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

# 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

^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
```

Next, we'll make sure sound support is enabled in the kernel with:

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

Add **snd-bcm2835** on its own line, to the modules file as shown below

A screenshot of a terminal window titled "pi@raspberrypi: ~". The window shows the contents of the "/etc/modules" file. It contains a comment explaining the purpose of the file and a list of kernel modules to load at boot time. The "snd-bcm2835" module is listed on a new line, indicating it has been added.

```
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.

i2c-dev
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
```

Now reboot your pi with:

```
sudo reboot
```

Once rebooted, re-log in.

Enter the following to confirm the modules are loaded

```
lsmod | grep snd
```

```
pi@raspberrypi:~ $ lsmod | grep snd
snd_soc_bcm2835_i2s    6354  0
snd_soc_core          125949  1 snd_soc_bcm2835_i2s
snd_pcm_dmaengine     3391  1 snd_soc_core
snd_bcm2835           20447  0
snd_pcm                75762  3 snd_bcm2835,snd_soc_core,snd_pcm_dmaengine
snd_timer              19288  1 snd_pcm
snd                   51908  4 snd_bcm2835,snd_soc_core,snd_timer,snd_pcm
pi@raspberrypi:~ $ |
```

Kernel Compiling

Ok now its time for the fun part! You'll manually compile in i2s support.

Start by updating your Pi:

```
sudo apt-get update
sudo apt-get install rpi-update
sudo rpi-update
```

Then reboot!

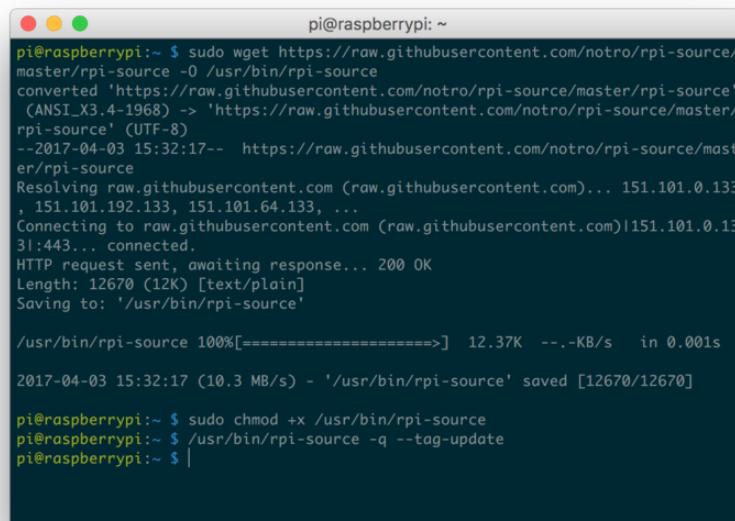
Install the compilation dependencies:

```
sudo apt-get install git bc libncurses5-dev bison flex libssl-dev
```

Download kernel source & compile:

```
sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source -O /usr/bin/rpi-source
sudo chmod +x /usr/bin/rpi-source
/usr/bin/rpi-source -q --tag-update
rpi-source --skip-gcc
```

On a Pi 3 this will take many many minutes, so don't worry if its taking 15 minutes. On a Pi Zero it can take an hour or longer!



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo wget https://raw.githubusercontent.com/notro/rpi-source/
master/rpi-source -O /usr/bin/rpi-source
converted 'https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source'
(ANSI_X3.4-1968) --> 'https://raw.githubusercontent.com/notro/rpi-source/master/
rpi-source' (UTF-8)
--2017-04-03 15:32:17-- https://raw.githubusercontent.com/notro/rpi-source/master/
rpi-source
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133
, 151.101.192.133, 151.101.64.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443...
HTTP request sent, awaiting response... 200 OK
Length: 12670 (12K) [text/plain]
Saving to: '/usr/bin/rpi-source'

/usr/bin/rpi-source 100%[=====] 12.37K --.-KB/s in 0.001s
2017-04-03 15:32:17 (10.3 MB/s) - '/usr/bin/rpi-source' saved [12670/12670]

pi@raspberrypi:~ $ sudo chmod +x /usr/bin/rpi-source
pi@raspberrypi:~ $ /usr/bin/rpi-source -q --tag-update
pi@raspberrypi:~ $ |
```

If the script pauses at this prompt:

```
Code coverage for fuzzing (KCOV) [N/y/?] (NEW)
```

Just press enter to accept the default and continue.

Prepare to Compile the i2s module

Now you're ready to compile i2s support:

```
sudo mount -t debugfs debugfs /sys/kernel/debug
```

This may already be done - mount: debugfs is already mounted - in which case keep going

If you are using Pi 3 or Pi 2 - make sure the module name is **3f203000.i2s**

If you are using Pi Zero - the module name is **20203000.i2s**

Run `rpi-i2s-audio`

Download the module, written by [Paul Creaser \(\)](#)

```
git clone https://github.com/PaulCreaser/rpi-i2s-audio  
cd rpi-i2s-audio
```

Pi Zero Only

If you are using a Raspberry Pi Zero, edit my_loader.c with nano my_loader.c and change the two lines

```
.platform = "3f203000.i2s",
```

and

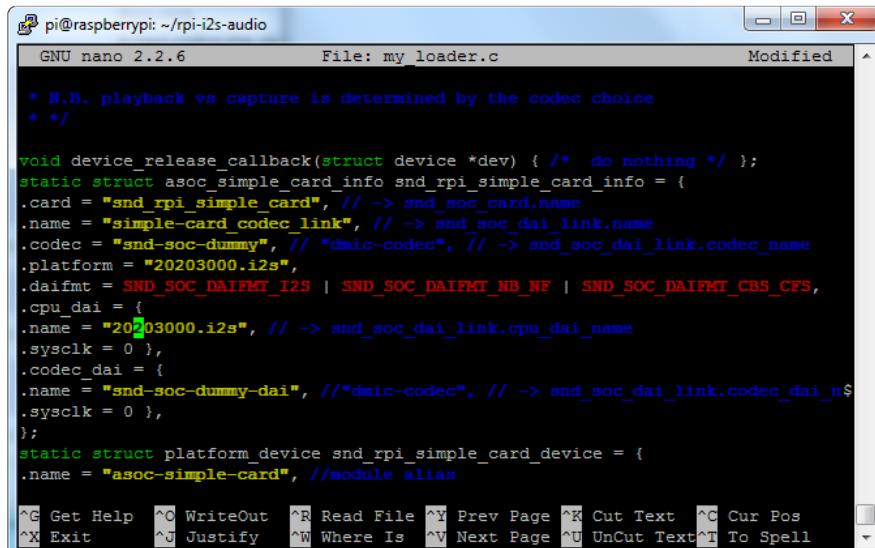
```
.name = "3f203000.i2s",
```

with

```
.platform = "20203000.i2s",
```

and

```
.name = "20203000.i2s",
```



```
pi@raspberrypi: ~/rpi-i2s-audio  
GNU nano 2.2.6          File: my_loader.c          Modified  
  
* N.B. playback vs capture is determined by the codec choice  
* */  
  
void device_release_callback(struct device *dev) { /* do nothing */ };  
static struct asoc_simple_card_info snd_rpi_simple_card_info = {  
.card = "snd_rpi_simple_card", // -> snd_soc_card.name  
.name = "simple-card codec link", // -> snd_soc_dai_link.name  
.codec = "snd-soc-dummy", // "dmic-codec", // -> snd_soc_dai_link.codec_name  
.platform = "20203000.i2s",  
.daifmt = SND_SOC_DAIFMT_I2S | SND_SOC_DAIFMT_NB_NF | SND_SOC_DAIFMT_CBS_CFS,  
.cpu_dai = {  
.name = "20203000.i2s", // -> snd_soc_dai_link.cpu_dai_name  
.sysclk = 0 },  
.codec_dai = {  
.name = "snd-soc-dummy-dai", // "dmic-codec", // -> snd_soc_dai_link.codec_dai_name  
.sysclk = 0 },  
};  
static struct platform_device snd_rpi_simple_card_device = {  
.name = "asoc-simple-card", //module alias
```

If you aren't using a Pi Zero, continue on!

Compile the module with

```
make -C /lib/modules/$(uname -r )/build M=$(pwd) modules  
sudo insmod my_loader.ko
```

Verify that the module was loaded:

```
lsmod | grep my_loader  
dmesg | tail
```

```
pi@raspberrypi:~/rpi-i2s-audio  
pi@raspberrypi:~/rpi-i2s-audio $ sudo insmod my_loader.ko  
pi@raspberrypi:~/rpi-i2s-audio $ lsmod | grep my_loader  
my_loader           1978    0  
pi@raspberrypi:~/rpi-i2s-audio $ dmesg | tail  
[ 13.135773] Adding 102396k swap on /var/swap. Priority:-1 extents:4 across:2  
41660k SSFS  
[ 13.416699] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready  
[ 13.546055] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready  
[ 13.551157] asix 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0x45E1  
[ 103.633295] random: crng init done  
[ 157.000466] my_loader: loading out-of-tree module taints kernel.  
[ 157.015313] request module load 'bcm2708-dmaengine': 0  
[ 157.027280] register platform device 'asoc-simple-card': 0  
[ 157.034692] Hello World :)  
[ 157.090003] asoc-simple-card asoc-simple-card.0: snd-soc-dummy-dai <-> 202030  
00.i2s mapping ok  
pi@raspberrypi:~/rpi-i2s-audio $
```

Note that on the Pi 2/3 you'll see `asoc-simple-card asoc-simple-card.0: snd-soc-dummy-dai <-> 3F203000.i2s mapping ok` on the last line and on Pi Zero you'll see `asoc-simple-card asoc-simple-card.0: snd-soc-dummy-dai <-> 20203000.i2s mapping ok`

Auto-load the module on startup

Now you can set it up so the module is loaded every time you boot the Pi

```
sudo cp my_loader.ko /lib/modules/$(uname -r)  
echo 'my_loader' | sudo tee --append /etc/modules &gt; /dev/null  
sudo depmod -a  
sudo modprobe my_loader
```

And reboot!

```
sudo reboot
```

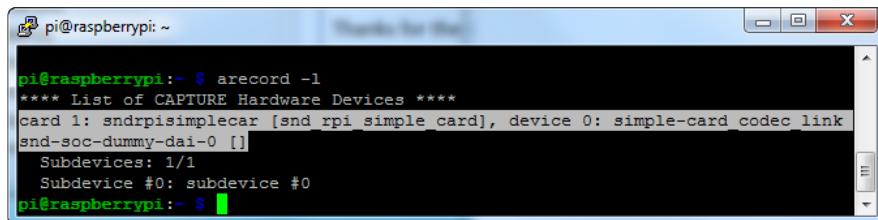
Test & Record!

OK that was a lot of effort but now you are ready to rock!

Use the following command to list the available input devices:

```
arecord -l
```

you should see a `snd_rpi_simple_card`



```
pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: snd_rpi_simplecar [snd_rpi_simple_card], device 0: simple-card_codec_link
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi:~ $
```

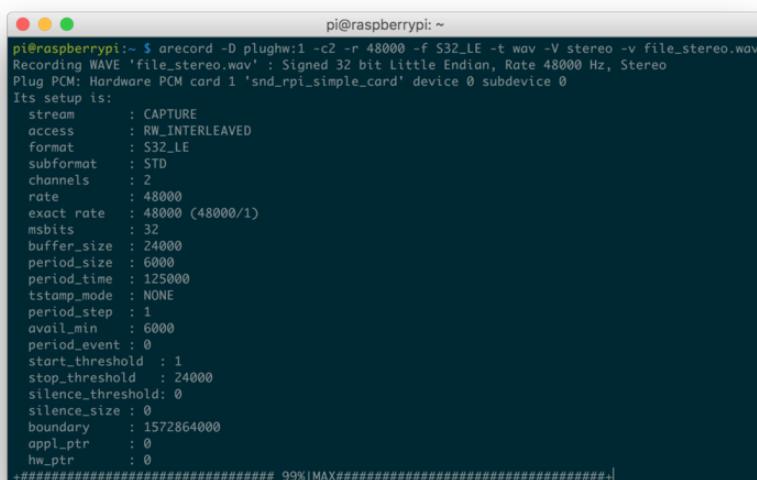
You can record a wav file in mono with this command:

```
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



```
pi@raspberrypi:~ $ arecord -D plughw:1 -c2 -r 48000 -f S32_LE -t wav -V stereo -v file_stereo.wav
Recording WAVE 'file_stereo.wav' : Signed 32 bit Little Endian, Rate 48000 Hz, Stereo
Plug PCM: Hardware PCM card 1 'snd_rpi_simple_card' device 0 subdevice 0
Its setup is:
  stream      : CAPTURE
  access      : RW_INTERLEAVED
  format      : S32_LE
  subformat   : STD
  channels    : 2
  rate        : 48000
  exact rate  : 48000 (48000/1)
  msbits      : 32
  buffer_size : 24000
  period_size : 6000
  period_time : 125000
  tstamp_mode : NONE
  period_step : 1
  avail_min   : 6000
  period_event : 0
  start_threshold : 1
  stop_threshold : 24000
  silence_threshold: 0
  silence_size : 0
  boundary    : 1572864000
  appl_ptr    : 0
  hw_ptr      : 0
+#####99%|MAX#####
pi@raspberrypi:~ $
```

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 :), just insert your Pi's IP address below:

```
scp pi@<local-ip>:/home/pi/file.wav ~/Desktop/file.wav
```

Adding Volume control

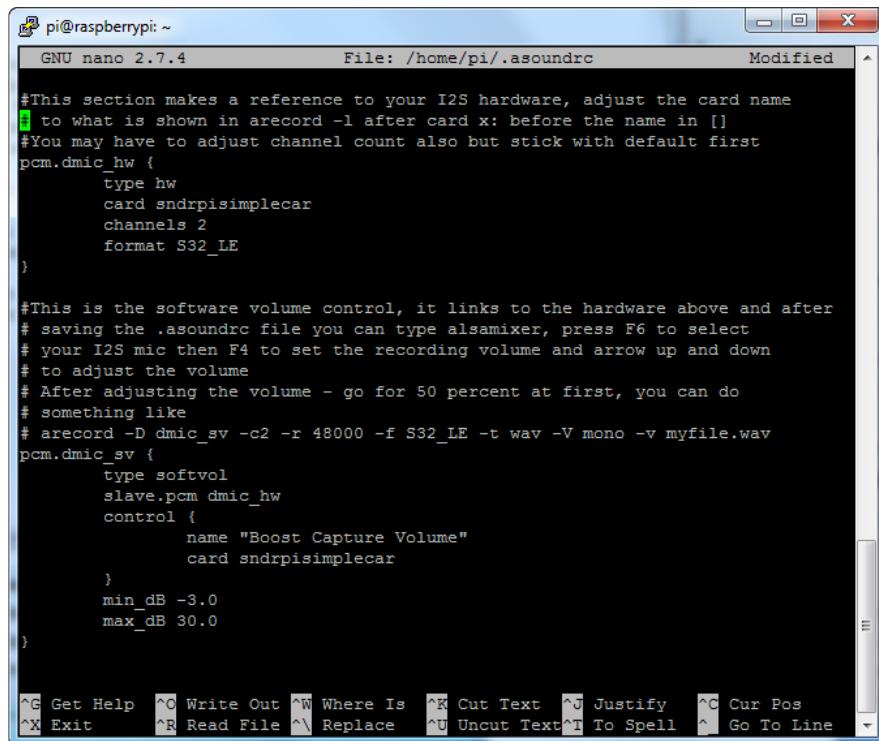
You can add volume control to your mine via alsamixer and alsa config. ([Hat tip to RickTracer \(\)](#))

Run `sudo 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 sndrpisimplecar
    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 sndrpisimplecar
    }
    min_dB -3.0
    max_dB 30.0
}
```



```

pi@raspberrypi: ~
GNU nano 2.7.4          File: /home/pi/.asoundrc          Modified
#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 sndrpisimplecar
    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 sndrpisimplecar
    }
    min_dB -3.0
    max_dB 30.0
}

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^N Replace ^U Uncut Text ^T To Spell ^L Go To Line

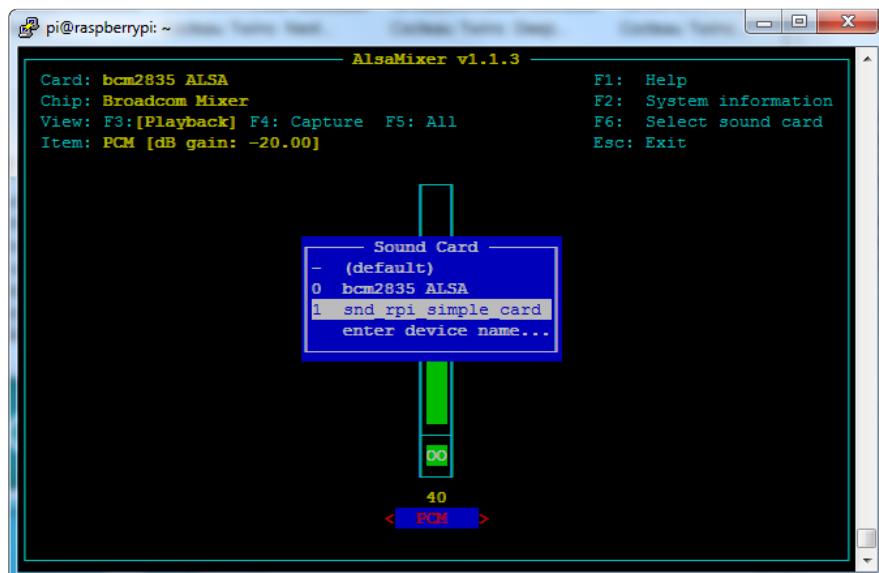
Now before you can change the volume you need to use the device once (this is an alsamixer thing)

Run

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

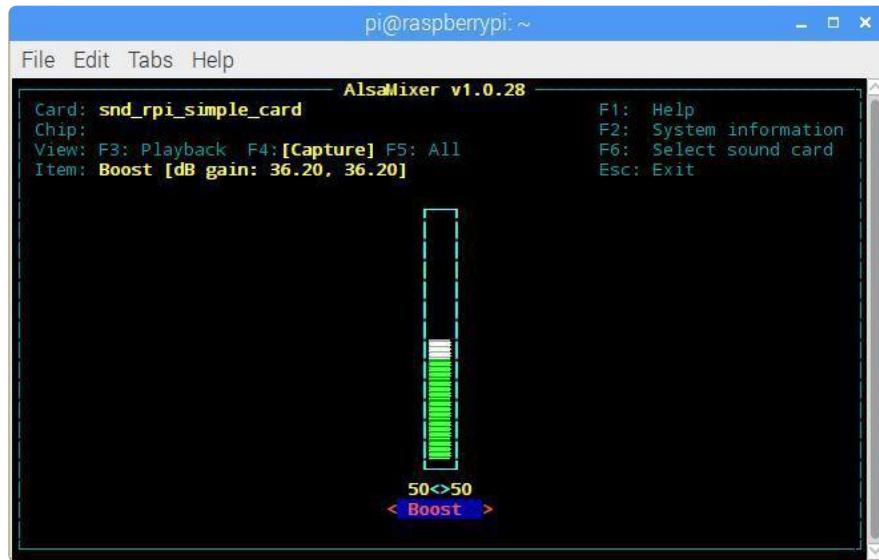
And cancel with ^C once it starts recording.

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



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

Press F5 to change the volume.



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

- [EagleCAD PCB Files on GitHub \(\)](#)
- [Fritzing object in the Adafruit Fritzing library \(\)](#)

Schematic & Fab Print

