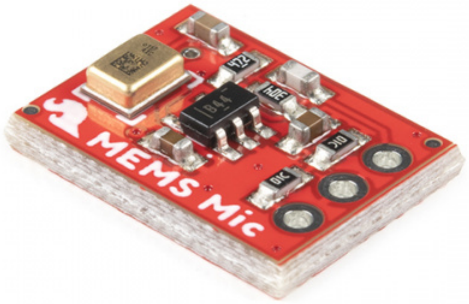
Analog MEMS Microphone Breakout - SPH8878LR5H-1 Hookup Guide

Introduction

Note: This tutorial covers the latest version of the SparkFun Analog MEMS Microphone Breakout (BOB-19389). We designed the updated version as a drop-in replacement so users with the previous versions of this breakout board (BOB-9868 or BOB-18011) can follow along with this tutorial. For specific details regarding the microphone ICs, refer to the Documents tab on their product pages or the previous release of this Hookup Guide:

RETIRED MEMS MICROPHONE HOOKUP GUIDE

The SparkFun Analog MEMS Microphone Breakout - SPH8878LR5H-1 is a simple and easy-to-use microphone for a variety of sound-sensing projects. The on-board microphone is a low-power, omnidirectional microphone with an analog output. It works for both near and long-range uses and is particularly good for portable applications due to its low power consumption. Possible applications include: smartphones, digital video cameras, and keeping an "ear" on your pets while you're away.



SparkFun Analog MEMS Microphone Breakout - SPH8878LR5H-1

BOB-19389

Read this guide to get an overview of the breakout board and how to use it, including its technical specifications, how to hook it up to a microcontroller, and example code to get started!

Required Materials

You'll need these items along with the MEMS Microphone Breakout to follow along with this tutorial. First up, you'll want a microcontroller to power the microphone and monitor its output:

SparkFun Thing Plus - ESP32-S2 WROOM
WRL-17743

SparkFun RedBoard Qwiic
DEV-15123

SparkFun RedBoard Turbo - SAMD21 Development Board
DEV-14812

SparkFun Qwiic Micro - SAMD21 Development Board
DEV-15423

Building a circuit using this breakout requires some assembly and soldering. You may already have a few of these items but if not, the tools and hardware below help with that assembly:

Hook-Up Wire - Assortment (Stranded, 22 AWG)
PRT-11375

Break Away Headers - Straight
PRT-00116

Soldering Iron - 60W (Adjustable Temperature)
TOL-14456

Solder Lead Free - 15-gram Tube
TOL-09163

Recommended Reading

To successfully use the SparkFun MEMS microphone breakout board, you'll need to be familiar with Arduino microcontrollers, analog (aka ADC) input, and sound waves. For folks new to these topics, check out the following resources to get a feel for the concepts and verbiage used throughout this tutorial.

What is an Arduino?

What is this 'Arduino' thing anyway? This tutorials dives into what an Arduino is and along with Arduino projects and widgets.

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

Analog vs. Digital

RedBoard Qwiic Hookup Guide

This tutorial covers the concept of analog and digital signals, as they relate to electronics.

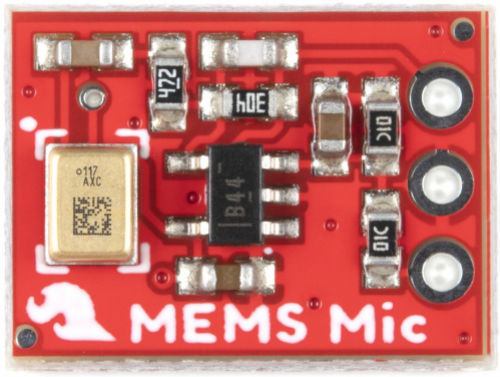
This tutorial covers the basic functionality of the RedBoard Qwiic. This tutorial also covers how to get started blinking an LED and using the Qwiic system.

Hardware Overview

The SparkFun Analog MEMS Microphone Breakout uses the SPH8878LR5H-1 microphone and amplifies the signal with an OPA344 OpAmp. Let's take a closer look at the SPH8878LR5H-1 and the other hardware on the board.

SPH8878LR5H-1 Microphone

The SPH8878LR5H-1 microphone from Knowles Electronics is a bottom port analog microphone that supports both single-ended and differential modes.

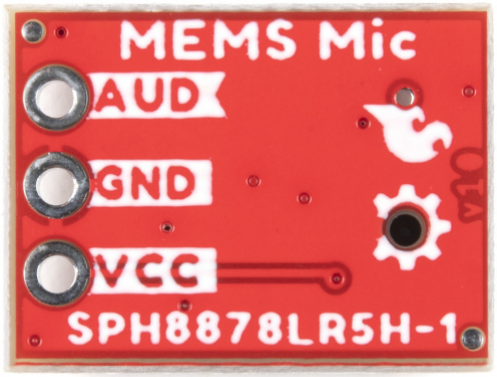


We opted for a single-ended output design on this breakout so it can act as a drop-in replacement for users with previous versions of the SparkFun MEMS microphone breakouts. The Left Out- pin is connected to a test point so savvy users who wish to use this microphone in differential mode can tap into that signal with some careful soldering.

The table below outlines some relevant specifications of the SPH8878LR5H-1. For a complete technical overview of the microphone, refer to the datasheet.

Parameter	Min	Typ	Max	Units	Notes
Sensitivity	-45	-44	-43	dBV/Pa	94 dB SPL @ 1kHz, Single-Ended Mode
Signal-to-Noise Ratio ("SNR")	-	66	-	dBV/Pa	94 dB SPL @ 1kHz, A-weighted Single-Ended Mode
Frequency Range	7	-	36	Hz(min)/kHz(max)	
Acoustic Overload Point	-	134	-	dB SPL	

The microphone receives audio input from the bottom of the board. The board breaks out the power pins (VCC and Ground) and the audio output (AUD) from the microphone:



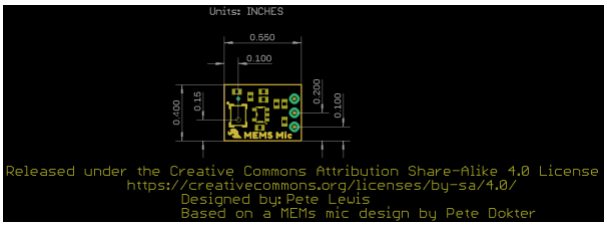
- **AUD** - Audio signal output.
- **VCC** - Voltage input (**2.3V to 3.6V**). Supply current of about 265µA.
- **GND** - Ground.

OpAmp

The SparkFun breakout board includes an OPA344 operational amplifier with a gain of *64 and a frequency response range of 7.2Hz-19.7KHz. The amplifier's AUD output floats at one-half VCC when the mic detects no sound. When held at arms length and talked into, the amplifier will produce a peak-to-peak output of just about 200 mV.

Board Dimensions

The board measures 0.50"x 0.40" (12.70mm x 10.16mm).



Hardware Assembly

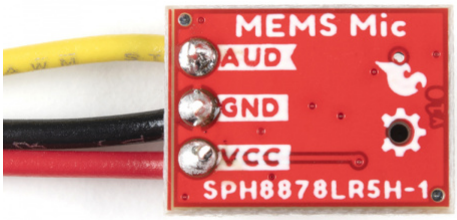
Now that we're familiar with the microphone breakout, let's connect it to a microcontroller and monitor some sound!

Microphone Breakout Connections

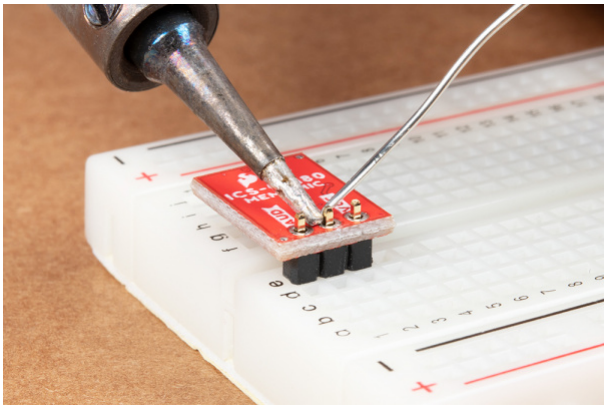
For a permanent connection, we recommend soldering three wires (or headers) to the PTHs on the breakout. We opted for soldering wires to the PTH connectors for a quick permanent connection to the breakout. For a temporary connection during prototyping, you can use IC hooks like these.

We recommend using the following colors of wire to easily distinguish the signals but you can always select a different color if you prefer (or do not have the colors used available).

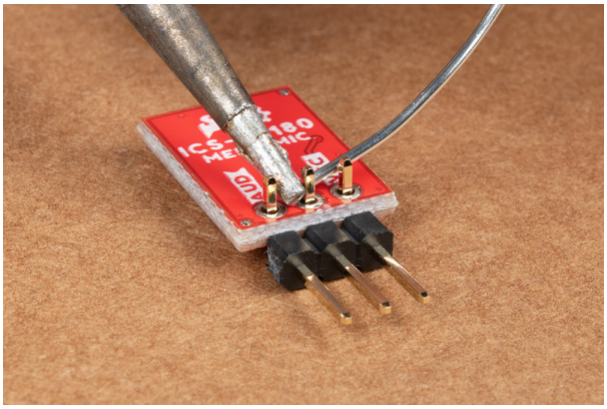
- **Red** for VCC
- **Black** for GND
- **Yellow** (or some other color not Red or Black) for AUD



Note: You can use any connection as explained above to connect. If you decide to solder straight header pins, we recommend inserting the straight header pin's tail from the top of the board so that the audio input for the microphone is facing away from a surface. However, depending on your application, you can also insert the pins on the side as well. For a low profile application, you will want to use right angle header pins.



Straight header pins being soldered to MEMS microphone.



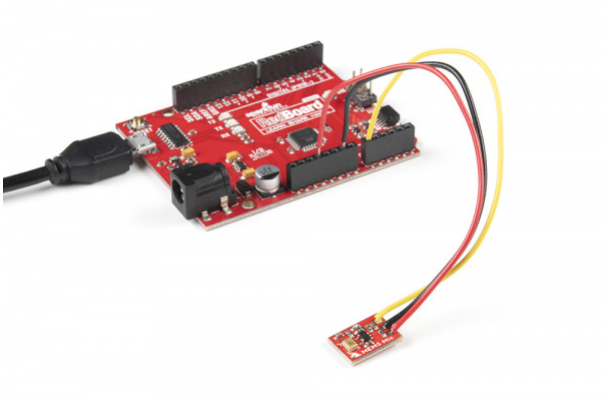
Right angle header pins being soldered to MEMS microphone.

Connecting to a Microcontroller

Next up we'll connect the breakout to a microcontroller we can use to monitor the audio signal output. For this tutorial, we used a SparkFun RedBoard Qwiic. Make the following connections between the breakout and RedBoard Qwiic (or whichever microcontroller you choose):

RedBoard/Arduino	MEMS Microphone
A0	AUD
GND	GND
3.3V	VCC

The completed circuit should look something like the photo below:



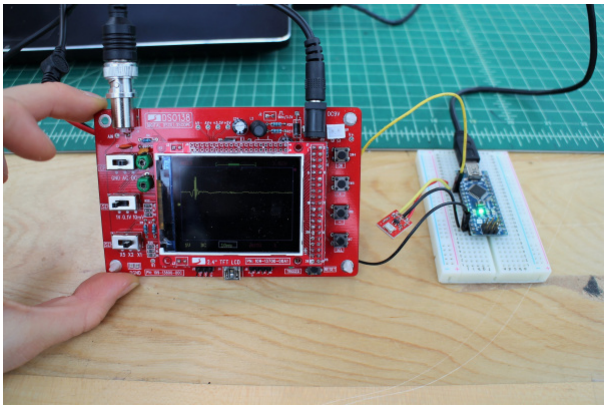
Read on to the next section for Arduino example code to monitor sound volume with the microphone breakout.

Arduino Software Example

> **Note:** If this is your first time using Arduino IDE or board add-on, please review the following tutorials.
>
> - Installing the Arduino IDE
> - Installing Board Definitions in the Arduino IDE

Interpreting the Audio Output Signal

The SPH8878LR5H-1 signal output is a varying voltage. When all is quiet, the AUD output floats at one-half the power supply voltage. For example, with a 3.3V power supply, the AUD output will be about 1.65V. In the photo below, the yellow marker on the left side of the oscilloscope screen marks the zero axis for the voltage (aka V = 0). The pulse is the AUD output of a finger snap close to the mic.



Converting ADC to Voltage

The microcontroller analog (ADC) input converts our audio signal into an integer. The range of possible ADC values depends on which microcontroller you are using. For an Arduino microcontroller with an ATmega328P, the analog resolution is 10-bits. This range is between 0 and 1023, so the resolution of our ADC measurement is 1024. To convert our analog measurement into a voltage, we use the following equation:

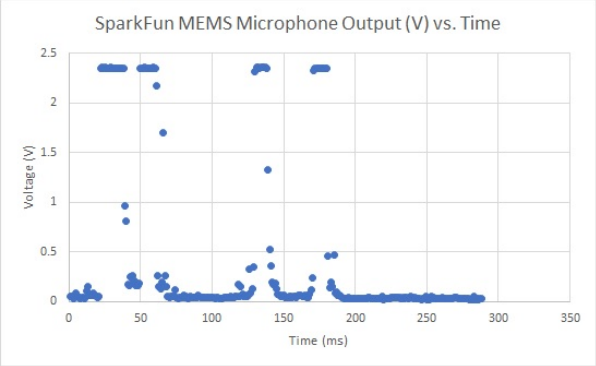
$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

In our case, the *ADC Resolution* is 1024, and the *System Voltage* 3.3 V. We'll need to add this equation in our code to convert our *ADC Reading* into a voltage.

But Wait, What Are We Actually Measuring??

For many applications that deal with sound (which is a wave), we're mostly interested in the **amplitude** of the signal. In general, and for the sake of simplicity, a larger amplitude means a louder sound, and a smaller amplitude means a quieter sound (and the sound wave frequency roughly corresponds to pitch). Knowing the amplitude of our audio signal allows us to build a sound visualizer, a volume unit ("VU") meter, set a volume threshold trigger, and other cool and useful projects!

To find the audio signal amplitude, take a bunch of measurements in a small time frame (e.g. 50 ms, the lowest frequency a human can hear). Find the minimum and maximum readings in this time frame and subtract the two to get the peak-to-peak amplitude. We can leave it at that, or divide the peak-to-peak amplitude by a factor of two to get the wave amplitude. We can use the ADC integer value, or convert this into voltage as described above.



Example Code

Note: For a simple test to see if your microphone is working, try using the example below! Select your Arduino board, COM port, and hit the upload button.

```
/*************************
  Simple Example Sketch for the SparkFun MEMS Microphone Breakout Board

*************************/

// Connect the MEMS AUD output to the Arduino A0 pin
int mic = A0;

// Variable to hold analog values from mic
int micOut;

void setup() {
  Serial.begin(9600);

}

void loop() {
  // read the input on analog pin 0:
  micOut = analogRead(mic);

  // print out the value you read:
  Serial.println(micOut);

}
```

Open the Serial Monitor or Serial Plotter to view the output. Snap, clap, or speak into the microphone and observe the readings. The raw value will be higher as the microphone picks up louder sounds. For a more refined example, check out the example code below! You can also view the raw output in the example code below but it requires a little bit more effort.

Below is a simple example sketch to get you started with the MEMS microphone breakout board. You can find the code in the GitHub repo as well. The code, written for an Arduino microcontroller, includes a conversion equation from the ADC Reading to voltage, a function to find the audio signal peak-to-peak amplitude, and a simple VU Meter that outputs to the Arduino Serial Monitor. For a more visual output, you can also use the Serial Plotter.

Be sure to read the comments in the code to understand how it works and to adapt it to fit your needs. Select your Arduino board, COM port, and hit the upload button.

```
/**************************
 * Example Sketch for the SparkFun MEMS Microphone Breakout Board
 * Written by jenfoxbot <jenfoxbot@gmail.com>
 * Code is open-source, beer/coffee-ware license.
 */

// Connect the MEMS AUD output to the Arduino A0 pin
int mic = A0;

// Variables to find the peak-to-peak amplitude of AUD output
const int sampleTime = 50;
int micOut;

//previous VU value
int preValue = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
   int micOutput = findPTPAmp();
   VUMeter(micOutput);
}


// Find the Peak-to-Peak Amplitude Function
int findPTPAmp(){
// Time variables to find the peak-to-peak amplitude
   unsigned long startTime= millis();  // Start of sample window
   unsigned int PTPAmp = 0;

// Signal variables to find the peak-to-peak amplitude
   unsigned int maxAmp = 0;
   unsigned int minAmp = 1023;

// Find the max and min of the mic output within the 50 ms timeframe
   while(millis() - startTime < sampleTime)
   {
      micOut = analogRead(mic);
      if( micOut < 1023) //prevent erroneous readings
      {
        if (micOut > maxAmp)
        {
          maxAmp = micOut; //save only the max reading
        }
        else if (micOut < minAmp)
        {
          minAmp = micOut; //save only the min reading
        }
      }
   }
```

```
  PTPAmp = maxAmp - minAmp; // (max amp) - (min amp) = peak-to-peak amplitude
  double micOut_Volts = (PTPAmp * 3.3) / 1024; // Convert ADC into voltage

  //Uncomment this line for help debugging (be sure to also comment out the VUMeter function)
  //Serial.println(PTPAmp);

  //Return the PTP amplitude to use in the soundLevel function.
  // You can also return the micOut_Volts if you prefer to use the voltage level.
  return PTPAmp;
}

// Volume Unit Meter function: map the PTP amplitude to a volume unit between 0 and 10.
int VUMeter(int micAmp){

  // Map the mic peak-to-peak amplitude to a volume unit between 0 and 10.
   // Amplitude is used instead of voltage to give a larger (and more accurate) range for the ma
p function.
   // This is just one way to do this -- test out different approaches!
  int fill = map(micAmp, 23, 750, 0, 10);

  // Only print the volume unit value if it changes from previous value
  while(fill != preValue)
  {
    Serial.println(fill);
    preValue = fill;
  }
}
```

Resources and Going Further

Now that you've connected your MEMS microphone breakout, it's time to incorporate it into your own project! For more information on the board, check out the resources below:

- Datasheet (SPH8878LR5H-1)
- Schematic
- Eagle Files
- Board Dimensions
- Hardware GitHub Repository
- Example Code GitHub Repository

If you run into trouble getting, or understanding, an audio signal output from the MEMS mic breakout board, try using a multimeter and/or an oscilloscope to measure the voltage output of the signal in quiet and loud settings. If you're still stuck, check out our forums and we'll help you troubleshoot.

After you've read in the MEMS microphone and have a good handle on the signal output, you're ready to start using it for practical microphone applications! Here are a few ideas to get you started:

1. Build a music visualizer! Here's a sample sketch for the music visualizer shown in the SparkFun Simple Sketches example.

2. Record sounds and play them back! You'll also need a speaker, an amplifier transistor, some pushbuttons, and some code. Here's an open-source mbed example. (The example was initially written for the ADMP401 but should work just fine with the latest release).
3. Make a sound-reactive EL Wire costume and replace the Sound Detector with the MEMS Microphone!
4. Make a Bark Back Pet Monitor with a Raspberry Pi to record the sound levels in your home, upload the data MQTT, and trigger an audio player to when the volume reaches a threshold.

Or check out these other audio related tutorials below.

Sound Detector Hookup Guide

The Sound Detector is a microphone with a binary output. This guide explains how it works and how you can use it in your projects.

WAV Trigger Hookup Guide V11

An introduction to being able to trigger music and sound effects based on buttons, sensors, or switches using the WAV Trigger board.

Hackers in Residence: The Sound Visualizer Pt. 2

An addition to a previous project, this time using a PC and a custom Java app to create your own music

Papa Soundie Audio Player Hookup Guide

Add sound effects to your project, prop or costume with Papa Soundie Audio Player.

visualizer using a RGB LED matrix.