LESSON C-6

WORKING WITH AUDIO

Audio can add another level to a project, whether it's playing audio through the speaker of a Halloween decoration, or playing your own recorded voice through a mobile robot. This lesson will teach you about the audio capabilities of the Raspberry Pi and how they can be used to add audio to a project.

Amplifiers

Back in Level B, you learned about using different pulse width modulation (PWM) frequencies to create different sounds with a Piezo-electric speaker. In addition to frequency, audio signals have another very important attribute called amplitude. Amplitude is the measurement of how large, or how many volts tall an audio signal is, and this value determines how much "work" the signal can do in driving a speaker.

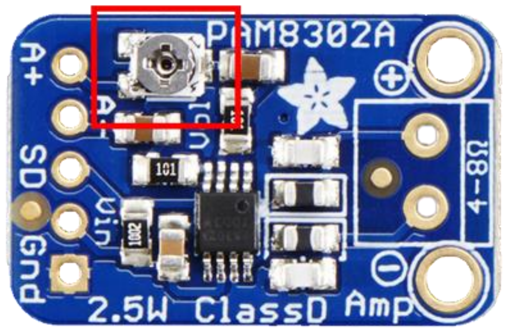
A small speaker, like a Piezo, should be driven by an audio signal with a very small amplitude, as the speaker element in a Piezo is very tiny and cannot move very far.

A larger speaker, like the speaker in a car stereo, needs a much larger amplitude signal to drive the speaker element and produce sound. The larger the speaker, the larger the drive signal needed for that speaker.

The device used to increase the amplitude of a signal is called an amplifier. An amplifier takes the input of a small signal, and outputs a much larger signal that is capable of driving larger speakers.

Amplifier Gain

The amount of amplification that is applied inside an amplifier is called gain. Too much gain in the amplifier and the output signal will overdrive the speaker, resulting in loud, distorted sound. Not enough gain in the amplifier will result in the speaker being under-driven, resulting in very low or no sound coming out of the speaker.



Some larger amplifiers have fixed gain levels that cannot be adjusted, but smaller amplifiers will often have adjustable gain that can be controlled by a potentiometer on the circuit board of the amplifier.

The photo above is the amplifier included in your kit. The highlighted potentiometer is used to adjust the gain:

- Counter-Clockwise: Lower gain, output level will be lower or more quiet.
- Clockwise: Higher gain, output level will be higher or more loud, full gain will likely be far too much for small speakers and could damage them.

In order to get the best sound quality, you might need to make adjustments to the input level using the volume control on your Pi and the output level using the gain adjustment on the amplifier. These two settings will work together to get a good output level for your speaker.

Distortion:

Distortion occurs when a drive signal is too large for a connected speaker, and the resulting sound is said to be distorted because it is not a true representation of the intended sound. The sound of distortion can often be attributed to a voice coming through a bullhorn, but it is also used as an effect for music featuring electric guitars.

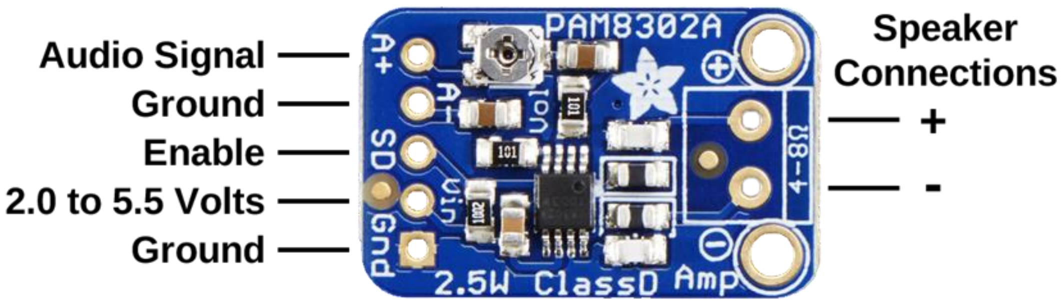
The level of distortion will vary with how badly the speaker is being overdriven. Slight distortion of a signal may only result in slightly degraded audio output, while heavier distortion might lead to audio that is completely unusable, and the possibility of damaging the speaker.

The PAM8302 Amplifier

The amplifier included in your kit is built around the PAM8302A chipset. It's capable of delivering 2.5 Watts of power into a speaker. Here are the full specifications:

- Input Power – 2.0V to 5.5V
- Compatible Speakers – 4 to 8 Ohm impedance. If connecting your amplifier to a speaker other than the one supplied in your kit, ensure the impedance rating is in the 4-8 Ohm range. Speakers with ratings outside this range can damage the amplifier.

The pinout for the PAM8302A Amplifier is shown below:



Board Label	Description
A+	Positive audio input signal
A-	Negative audio signal if available or ground
Enable	Apply ground switch amplifier into low power (off) mode, or leave unconnected and amplifier will always be on
Vin	Voltage In — Range can be 2.0V to 5.5V
Gnd	Ground connection
Speaker +	Positive Speaker Connection
Speaker -	Negative Speaker Connection

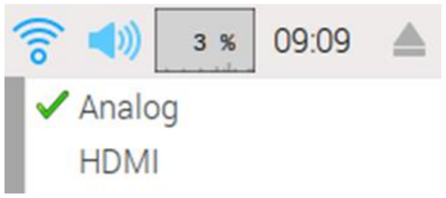
The speaker connections are made using bare speaker wire and the screw-terminal connector, while the other signals can be accessed using the standard breadboard-compatible header.

Audio Output Hardware

The Raspberry Pi model 3B from your kit includes a 3.5mm jack that can be used to output audio signals. This type of connection is commonly referred to as a "headphone" jack as it's normally used to connect headphones to a device. This port is located between the Ethernet and HDMI connectors on the Pi.

The Pi can also output audio using the HDMI cable. This can be useful if you have a TV or computer monitor with built-in speakers and an HDMI connection. Running audio over the HDMI connection allows you to run both audio and video in a single cable, eliminating the need for a separate, dedicated audio cable.

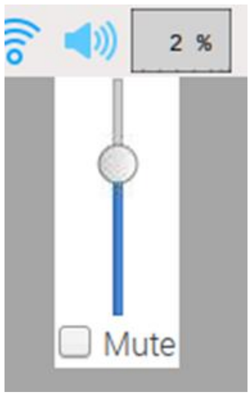
You can switch between these two outputs by right-clicking on the audio icon in the upper-right menu bar and selecting the option that you would like.



- Analog: This selection will output audio on the 3.5 mm audio jack on the side of the Pi.

- HDMI: This selection will output audio on the HDMI connector, so it can be received by an HDMI connected device with speakers.

The output volume level can be adjusted by left-clicking the audio icon and adjusting the slider up and down. The top of the slider is 100% volume and the bottom is 0%. There is also a Mute checkbox that will automatically make the output level 0%.



If you're ever having trouble with sound not coming out of your Pi when you expect it to, check the settings <u>in this order</u>:

- Mute is not checked

- Volume slider is near 100%

Output device matches your connected device, Analog for a device connected to the 3.5mm audio jack and HDMI for audio capable devices connected to the HDMI port.

Also, do not assume that all HDMI devices have speakers. HDMI connected TVs will have internal speakers, but most HDMI computer monitors do not have internal speakers. When in doubt, check the specifications of your device to determine its audio capabilities and required connections.

Raspberry Pi Audio Output Quality:

Instead of having dedicated audio processing circuitry, the audio output on a Raspberry Pi runs using software PWM. This results in lower quality audio output, but this lack of extra circuitry helps to keep the Pi as small as possible. The audio output quality is acceptable for voices or sound effects, but the lower quality would likely be very noticeable when trying to use the Pi to play music.

USB Audio Output Devices

One way to get better quality output audio from a Raspberry Pi is to use an external USB sound card. These devices have their own devoted circuitry to create audio signals, which will often result in higher quality audio. Some of the drawbacks related to these types of devices are driver and software support. Just because you can plug a USB device into your Raspberry Pi doesn’t mean the Raspbian OS will know how to talk to that device.

A list of supported devices and any known issues with their driver support can be found here:

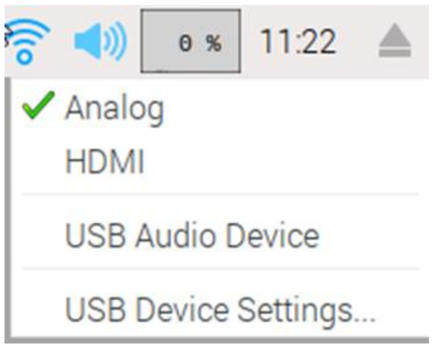
https://elinux.org/RPi_VerifiedPeripherals#USB_Sound_Cards

If you plan to attempt to use an external USB audio device with your Pi, make sure you consult this list to ensure that the hardware is supported and that you fully understand the software modifications that will be required in order to make your device work.

The USB Audio Adapter included in your kit uses the GeneralPlus chipset and driver. While this device is not called out specifically in the list of supported devices, this audio adapter adheres to USB audio standards, so it falls under the Class compliant USB Sound Cards at the bottom of the list. Using a device that conforms to USB audio standards will ensure the best possible driver and software support, as support for this type of audio device is built directly into Raspbian (and other Linux distributions).

A USB audio device should only be plugged into a Raspberry Pi when the Pi is powered off. This will allow the Pi to detect the audio device at boot and load any necessary drivers so the device can be used for audio output once the Pi has fully booted up.

If an external USB audio device is used, additional options will appear when right-clicking on the audio icon in the upper-right menu bar.



If your USB audio device supports audio output, then selecting **`USB Audio Device`** from the list will output audio using that device, and not Analog or HDMI. If your device does not support output audio, then this option will not be present in the list.

Clicking on **`USB device settings`** will allow you configure audio settings related to the USB audio device. These settings will be covered later in the section on USB Audio Input Settings.

Audio on GPIO Pins

You’ve previously learned about sending audio out through the 3.5mm jack and HDMI, but there is another method available for getting audio out of a GPIO pin. The left and right audio channels coming out of the 3.5mm jack are actually connected to GPIO40 and GPIO45. You've never heard of GPIO pin numbers this high because these are not user-accessible pins. These GPIO pins can only be used internally by the board to send audio out of the 3.5mm jack.

The Pi uses some additional GPIO modes named ALT0 and ALT5 for playing right and left audio channels, ALT0 for the right, and ALT5 for the left. Since you have so much control over the GPIO pins on the Raspberry Pi, you can actually use software to connect GPIO pins from our breadboard to the ALT0 and ALT5 channels. That way whenever audio is played through the 3.5mm audio jack, it will also be played through the GPIO pin.

There are some limitations on which GPIO pins you can use for this type of operation:

Channel	GPIO Mode	GPIO Pins
Right	ALT0	GPIO12 or GPIO13
Left	ALT5	GPIO18 or GPIO13

By setting the mode for GPIO18 to ALT5, you can obtain the left channel of any audio played through the 3.5mm audio jack. This can be done by using the following WiringPi command on the CLI:

```
gpio -g mode 18 ALT5
```

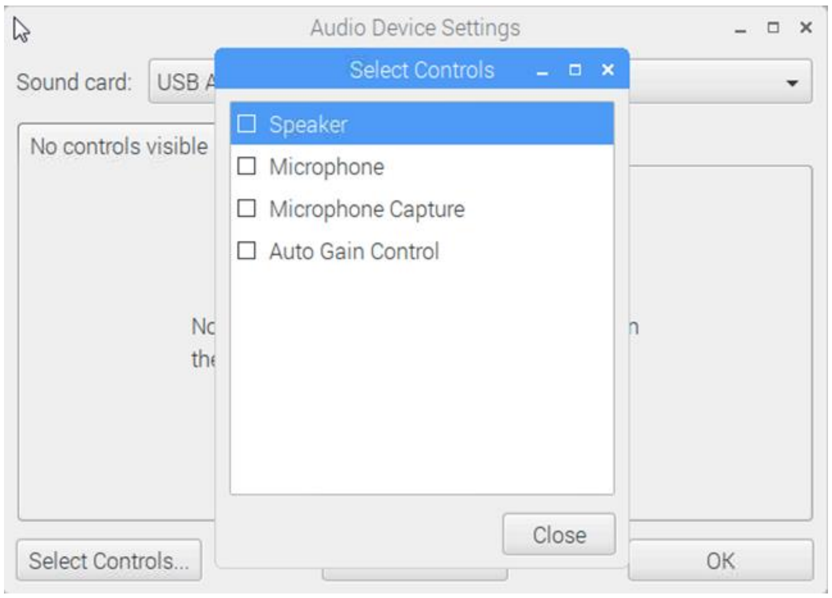
Audio Input Hardware

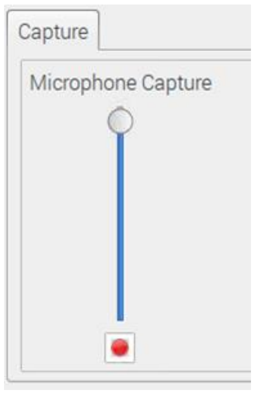
The Raspberry Pi does not have built-in capability to allow for audio input like a microphone. If desired, this functionality can be added by using a USB Audio Device.

USB Audio Input Devices

Most USB audio devices will have a 3.5mm output or headphone jack, but some will also have another 3.5mm input or microphone jack that can be used to record audio input. While audio output level can be controlled by the volume slider, microphone input level can also be adjusted by right clicking on the audio icon in the upper-right menu bar and selecting **`USB Device Settings`** from the menu. Using the dropdown menu at the top of the settings window, select **`USB Audio Device (Alsa Mixer)`**. This will ensure that you're looking at the settings for your USB audio device.

Initially, there will not be any settings available as the settings that are available are configured in the **`Select Controls`** area. After clicking on the **`Select Controls`** button, you will be able to select which audio controls you would like to view in the main panel by adding a check box next to each control:





The Speaker and Microphone settings are not needed as they just set the output volume level for those 3.5mm ports on the USB audio device, and that setting can be accessed in the top menu bar of the desktop. Clicking the box next to **`Microphone Capture`** and then closing the window with the **`Close`** button will allow this control to be accessed in the main control window.

The **`Capture`** tab will now contain the **`Microphone Capture`** level slider.

The position of this slider will determine how much gain will be applied to the signal coming in from the microphone when being recorded. Louder sounds coming into the microphone will need a lower capture level on the slider, while quieter sounds requiring more gain, will need a higher capture level on the slider to ensure they get captured with good quality.

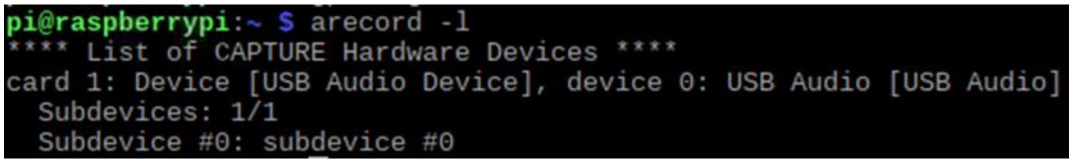
The red button below the slider will mute the microphone input, disabling all input from the microphone. When the button is red, the microphone is live and can be recorded. When the button is light gray, the microphone input is muted, and no sound can be recorded. The mute function can be toggled by clicking on this button.

Software Audio Functions

Recording and playing audio from the CLI is fairly simple with a couple of commands called **`arecord`** and **`aplay`**. As the names suggest, **`arecord`** is used to record audio, and **`aplay`** is used to play audio files.

The Arecord Command

In order to record audio, you will need to figure out the location of your USB capture device. This location will consist of a card and device number. Luckily, **`arecord`** has a handy option called **`-l`** (the letter l is for list) that can be used to list all available recording devices on the system. The full command is **`arecord -l`** and the output looks something like this:



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

There was only one capture device found on this system and its location is card 1, device 0. When you supply this location as an option later when recording with **`arecord`**, you will use the format **`--device=hw1,0`** to specify that you would like to record input from this device.

Here are some of the options that must be specified when using the **`arecord`** command:

`--device=hw1,0`	Device location as seen above
`--format S16_LE`	Recorded format will be signed 16-bit audio
`--rate 48000`	Sample rate of 48000 Hz (48KHz)
`-c1`	Record only one channel, also known as mono
`test.wav`	Filename to use for recorded file

Here are some of the options that must be specified when using the **`arecord`** command:

`--device=hw1,0`	Device location as seen above
`--format S16_LE`	Recorded format will be signed 16-bit audio
`--rate 48000`	Sample rate of 48000 Hz (48KHz)
`-c1`	Record only one channel, also known as mono
`test.wav`	Filename to use for recorded file

Here is the command and all the options assembled into a single command:

```
sudo arecord --device=hw:1,0 --format S16_LE --rate 48000 -c1 test.wav
```

Running this command in the directory **`/home/pi`** will save a new file named **`test.wav`** to the **`/home/pi`** folder. The recording will start as soon as this command is executed and a message with recording information will be printed to the console. When you're finished with your recording you can use the CTRL-C keyboard shortcut to end the program.

The Aplay Command

The **aplay** command also has a -l option to list all available playback devices, but a card or device number is not required to be specified when running the **aplay** command. If no device is specified as an option, then **aplay** will play sound through whatever output device is selected in the upper-right status bar. This makes it a lot simpler to run the **aplay** command:

```
sudo aplay test.wav
```

This command will play the file named **test.wav** from the current working directory, through the currently selected output device. You could also specify the full file path so you can play sounds that are located in another folder. This would allow you to be in **/home/pi** but play a sound file that's located in **/home/pi/Desktop**:

```
sudo aplay /home/pi/Desktop/hello.wav
```

You could run this command from any directory in the system, and **aplay** would still play **hello.wav** from your Desktop.

The OS Module

The **os** module in Python allows you to run CLI commands from within a Python program. This can be helpful when you want to do something in a Python program that can only be done on the CLI.

Importing the **os** module can be done just like all other modules:

```
import os
```

This must be done at the beginning of your program so the **os** module will be available throughout the rest of the program.

The os module has many functions, but you will mainly be focusing on its **`os.system()`** function. This is the function that allows commands to run as if they were typed into the CLI:

```
os.system("some_command")
```

Anything placed between the double-quotation marks will run as if typed into the CLI. In this case **`some_command`** would attempt to run on the CLI. Let's look at some more useful examples:

```
os.system("aplay /home/pi/test.wav")
```

The command above will use the **`aplay`** command to play **`test.wav`** from the **`/home/pi`** directory.

You can also string multiple **`os.system()`** commands together:

```
os.system("gpio -g mode 18 ALT5")

os.system("aplay /home/pi/Desktop/test.wav")

os.system("gpio -g mode 18 in")
```

In the program above, a WiringPi command is being used to switch GPIO18 to a PWM pin that will contain the left audio channel, aplay will play **`test.wav`** from the Desktop, and another WiringPi command is switching GPIO18 back to its default state as an input.

As you can see from the example above, the ability to execute CLI commands from within Python programs definitely opens up some new possibilities to explore in upcoming projects.