Raspberry Pi operating system includes the Advanced Linux Sound Architecture (ALSA). Two of the command line methods available in ALSA are **arecord** and **aplay**. These will record and playback audio in wave format but as far as I know, not MP3 format or any other audio format. There may be the possibility to record in MP3 format with a plug-in, but this paper will cover the software without plug-ins.

ALSA is part of the Linux operating system and it appears that Raspbian versions have limited audio formats available. As far as I can determine, it is only possible to record in one format, signed 16-bit Little Endian (S16_LE) wave. Furthermore, the sampling rate is restricted to 44,100 Hz. This is a good format for recording good quality music but is overkill for recording voice.

Sound sampled at a rate of 44,100 Hz results in 44,100 samples per second of the sound. If the format is 16 bit, then each sample is 2 bytes in size (there are 8 bits per byte). Therefore, one second of sound sampled at 16 bit with 44,100 Hz will have a file size of 2 X 44,100 = 88,200 bytes. There are 1024 bytes in one kilobyte (KB) and 1024 kilobytes in one megabyte (MB) and 1,048,576 bytes in one megabyte. Thus, one second of sound at 16 bit 44,100 Hz will contain 88,200 bytes / 1,048, 576 bytes per MB = 0.084 MB per second. Then 12 seconds of audio would have a file size of 0.084 X 12 = 1.008 MB. A sound file of 12 seconds would be about one megabyte in size, which is quite large.

We are dealing with two problems here: **1)** the wave format is not a compressed format and **2)** the sampling rate of 16 bit 44,100 Hz is much more than we need to get good quality audio for voice only (no music). The MP3 format is a compressed format and results in much smaller file sizes. This would be optimal when we want to transfer audio files over the Internet. Also, fairly good voice can be done at 16 bit 8,000 Hz, which yields files about one-fifth the size of 44,100 Hz. Ultimately, for doing voice over the Internet, we need a more compressed audio format than wave 16 bit 44,100 Hz. But since that is the format we are stuck with in ALSA, we will start learning RPi audio with this format.

GEAR has USB microphones and USB speakers that can be used to record and playback audio. However, these are not easy plug and play devices on the RPi, compared to a PC. After reading through this lesson you will have an appreciation for the complexity that arises in using the USB devices.

To learn more about the **arecord** method, we can simply type the command name in the terminal, which will display the options available. Below is a record of issuing the command with an edited partial list of the options of interest for this lesson.

```
pi@GEAR1~ $ arecord

Usage: arecord [OPTION]... [FILE]...

    -l, --list-devices     list all soundcards and digital audio devices
    -D, --device=NAME      select PCM by name
    -c, --channels=#       channels
    -f, --format=FORMAT    sample format (case insensitive) [in this case must use --format S16_LE]
    -r, --rate=#           sample rate [must use 44100 Hz, if you specify other, it will default to 44100]
```

We will start by looking at the first option to use with the command, which will yield further important information: **`-l (lower case L)`** to list devices available on RPi for recording:

```
arecord -l
```

Below is the output of the GEAR1 RPi when above command was issued:

```
pi@GEAR1:~ $ arecord -l

 **** List of CAPTURE Hardware Devices ****
 card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
```

From the above we learn that the RPi has only one audio capture device (microphone). It happens to be a USB device. Also, very important, it has been assigned as card 1.

Another ALSA method we need to cover is **amixer**. We will issue the amixer command with the following options:

`amixer -c1 info`

Remember that our USB microphone is card 1, so to list information about it, we must specify **`-c1`** in the command followed by **`info`**.

```
pi@GEAR1:~ $ amixer -c1 info

Card hw:1 'Device'/'C-Media Electronics Inc. USB PnP Sound Device at usb-3f980000.usb-1.4, full spe'
  Mixer name     : 'USB Mixer'
  Components     : 'USB8086:0808'
  Controls      : 4
  Simple ctrls  : 2
```

From the above we learn that the mixer that is part of the USB microphone is designated card hw:1. This is an important piece of information we need when issuing the record command. We also learn that the microphone mixer contains four controls. We need to issue the amixer command with another option to learn more about the controls:

```
pi@GEAR1:~ $ amixer -c1 controls

numid=2,iface=MIXER,name='Mic Capture Switch'
numid=3,iface=MIXER,name='Mic Capture Volume'
numid=4,iface=MIXER,name='Auto Gain Control'
numid=1,iface=PCM,name='Capture Channel Map'
```

From the above we learn that the volume control for the microphone has a number ID of 3 (numid=3). We can learn more about that control by issuing the command below:

```
pi@GEAR1:~ $ amixer -c1 cget numid=3

numid=3,iface=MIXER,name='Mic Capture Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=16,step=0
  : values=0
  | dBminmax-min=0.00dB,max=23.81dB
```

From the above we learn that the volume input for the microphone can be set from a minimum value of 0 to a maximum value of 16. Also note that currently the input is set to a value of 0 (values=0). When I first started working with this microphone I thought it was not a very good device. Now I know that when you connect the USB microphone to the RPi, it is initially set for the lowest input volume. So, let's fix that with the next command:

```
pi@GEAR1:~ $ amixer -c1 cset numid=3 16

numid=3,iface=MIXER,name='Mic Capture Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=16,step=0
  : values=16
  | dBminmax-min=0.00dB,max=23.81dB
```

Now we see that microphone input volume is set at the maximum level. It will now yield good volume when making recordings. It is also possible to adjust the volume with audio controls on the RPi desktop, which I will cover at the end of this lesson.

Now we have all the required information we need to make a recording (and the microphone is adjusted properly). The command below will make a recording:

```
arecord --device=hw:1 --format S16_LE --rate 4100 -c1 jeff.wav
```

Let's look at the options included above. First, **`--device=hw:1`**. When we issued the **`amixer -c1 info`** command we learned that the microphone had been assigned a device name of **`hw:1`**. The next option is: `--format S16_LE,` which is the only format available, but it must be specified. Recall that I indicated that the rate must be 44100 Hz but I have specified a much lower rate of 4100. We will see what happens with this below. Next, we have the option -c1 which indicates the number of channels (c1 is mono and c2 is stereo, we only have one microphone so recording must be mono). Lastly, we specify the file name for the recording (jeff.wav). If you don't specify a directory path, the file will be placed in the pi directory.

```
pi@GEAR1:~ $ arecord --device=hw:1 --format S16_LE --rate 4100 -c1 jeff.wav

Recording WAVE 'jeff.wav' : Signed 16 bit Little Endian, Rate 4100 Hz, Mono
Warning: rate is not accurate (requested = 4100Hz, got = 44100Hz)
         please, try the plug plugin
^CAborted by signal Interrupt...
```

Notice that my rate setting of 4100 was not accepted. Rather, the rate was set at 44100 Hz. I tried various rates to get smaller file sizes, but alas, all values lower than 44100 resulted in an accepted rate of 44100 Hz.

To terminate the recording just press ctrl + C. The last line of the command output displayed this as I pressed the keyboard combination to halt the recording.

Now let's see how to play it back with **aplay**. First, we must gather information about the playback device (the USB speakers). Let's see what some of the important options are for the aplay method.

```
pi@GEAR1:~ $ aplay

Usage: aplay [OPTION]... [FILE]...
-l, --list-devices      list all soundcards and digital audio devices
-L, --list-pcms         list device names
-D, --device=NAME       select PCM by name
```

We need information about available playback devices and also the names of those devices.

```
pi@GEAR1:~ $ aplay -l

**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835
IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Device_1 [USB2.0 Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

From the above we can see that there are two playback devices available. The one assigned to card 0 is the audio device that is part of the RPi chipset (bcm2835). We don't want to play the audio back through this device because we have not connected a speaker to it. The one we want is the USB2.0 device, which we see has the card number of 2.

We need an additional piece of information which is gained with the following command:

```
pi@GEAR1:~ $ aplay -L

null
    Discard all samples (playback) or generate zero samples (capture)
default
sysdefault:CARD=ALSA
    bcm2835 ALSA, bcm2835 ALSA
    Default Audio Device
dmix:CARD=ALSA,DEV=0
    bcm2835 ALSA, bcm2835 ALSA
    Direct sample mixing device
dmix:CARD=ALSA,DEV=1
    bcm2835 ALSA, bcm2835 IEC958/HDMI
    Direct sample mixing device
dsnoop:CARD=ALSA,DEV=0
    bcm2835 ALSA, bcm2835 ALSA
    Direct sample snooping device
dsnoop:CARD=ALSA,DEV=1
    bcm2835 ALSA, bcm2835 IEC958/HDMI
    Direct sample snooping device
hw:CARD=ALSA,DEV=0
    bcm2835 ALSA, bcm2835 ALSA
    Direct hardware device without any conversions
hw:CARD=ALSA,DEV=1
    bcm2835 ALSA, bcm2835 IEC958/HDMI
    Direct hardware device without any conversions
plughw:CARD=ALSA,DEV=0
    bcm2835 ALSA, bcm2835 ALSA
    Hardware device with all software conversions
plughw:CARD=ALSA,DEV=1
    bcm2835 ALSA, bcm2835 IEC958/HDMI
    Hardware device with all software conversions
sysdefault:CARD=Device_1
    USB2.0 Device, USB Audio
    Default Audio Device
front:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    Front speakers
surround21:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    2.1 Surround output to Front and Subwoofer speakers
surround40:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    4.0 Surround output to Front and Rear speakers
surround41:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    4.1 Surround output to Front, Rear and Subwoofer speakers
surround50:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    5.0 Surround output to Front, Center and Rear speakers
surround51:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    5.1 Surround output to Front, Center, Rear and Subwoofer speakers
surround71:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    7.1 Surround output to Front, Center, Side, Rear and Woofer
speakers
iec958:CARD=Device_1,DEV=0
```

```
    USB2.0 Device, USB Audio
    IEC958 (S/PDIF) Digital Audio Output
dmix:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    Direct sample mixing device
dsnoop:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    Direct sample snooping device
hw:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    Direct hardware device without any conversions
plughw:CARD=Device_1,DEV=0
    USB2.0 Device, USB Audio
    Hardware device with all software conversions
```

Well, that was a whole lot of information. The only piece we need is `sysdefault:CARD=Device_1`, which you may notice is the first line that is not part of the `bcm2835` chipset. This is the device name that we need to include in the **aplay** command to ensure that the sound plays back through the USB speakers

Before we do the playback, perhaps we should check the volume setting of the USB speakers. Remember that the USB speakers are assigned as card 2 (-c2).

```
pi@GEAR1:~ $ amixer -c2 info
```

```
Card hw:2 'Device_1'/'Generic USB2.0 Device at usb-3f980000.usb-1.2,
full speed'
  Mixer name     : 'USB Mixer'
  Components     : 'USB1908:2070'
  Controls      : 3
  Simple ctrls  : 1
```

The mixer for the USB speakers has 3 controls. Let's learn more about those controls.

```
pi@GEAR1:~ $ amixer -c2 controls
```

```
numid=2,iface=MIXER,name='PCM Playback Switch'
numid=3,iface=MIXER,name='PCM Playback Volume'
numid=1,iface=PCM,name='Playback Channel Map'
```

We will be interested in checking the volume control, which is numid=3. Let's see how it is currently set.

```
pi@GEAR1:~ $ amixer -c2 cget numid=3
```

```
numid=3,iface=MIXER,name='PCM Playback Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=255,step=0
  : values=255
  | dBminmax-min=-128.00dB,max=-127.00dB
```

The volume can be set to values between 0 and 255. Currently the volume is set to maximum (values=255). Well. that might be a bit loud, but we can leave it there for now. You can also adjust the volume using the desktop controls. The command below will result in playing of the file jeff.wav at maximum volume.

```
amixer -c2 cset numid=3 255; aplay -D sysdefault:CARD=Device_1 jeff.wav
```

Below is the terminal output when issuing the command.

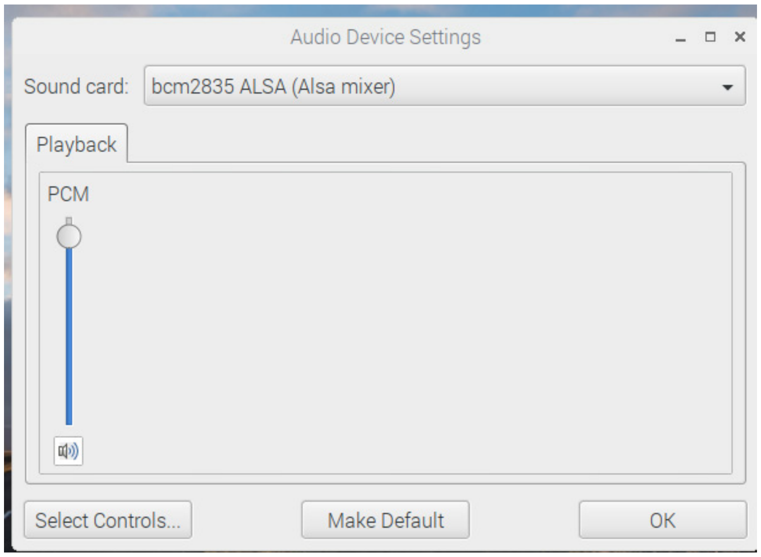
```
pi@GEAR1:~ $ amixer -c2 cset numid=3 255; aplay -D sysdefault:CARD=Device_1 jeff.wav

numid=3,iface=MIXER,name='PCM Playback Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=255,step=0
  : values=255
  | dBminmax-min=-128.00dB,max=-127.00dB
Playing WAVE 'jeff.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
```

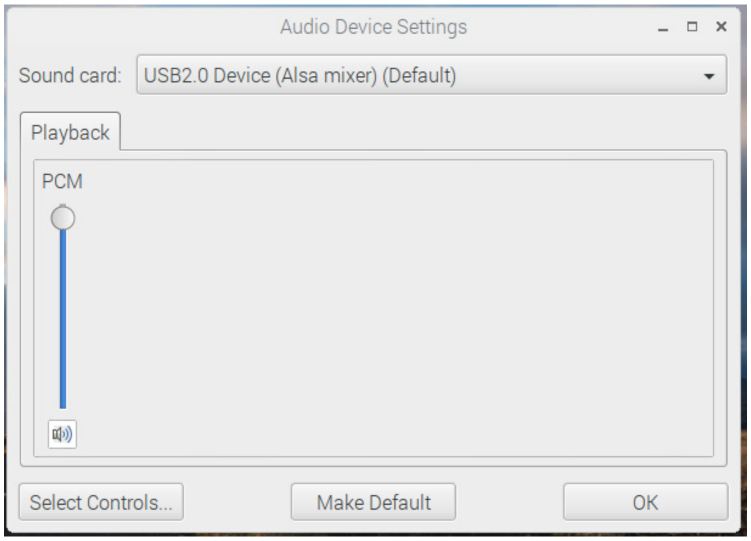
I would be the first to admit that the above procedures seem a bit tedious. It would be much easier to download a copy of Audacity to do your recordings and playback from the RPi desktop. And with Audacity you can record in MP3 format at lower sampling rates, which result in much smaller file sizes. Nevertheless, by going through this lesson, you have learned how to gain some information about the sound devices that are connected to the RPi. That will probably come in handy when you work further with audio on the RPi.

Adjusting Audio Devices from the Desktop

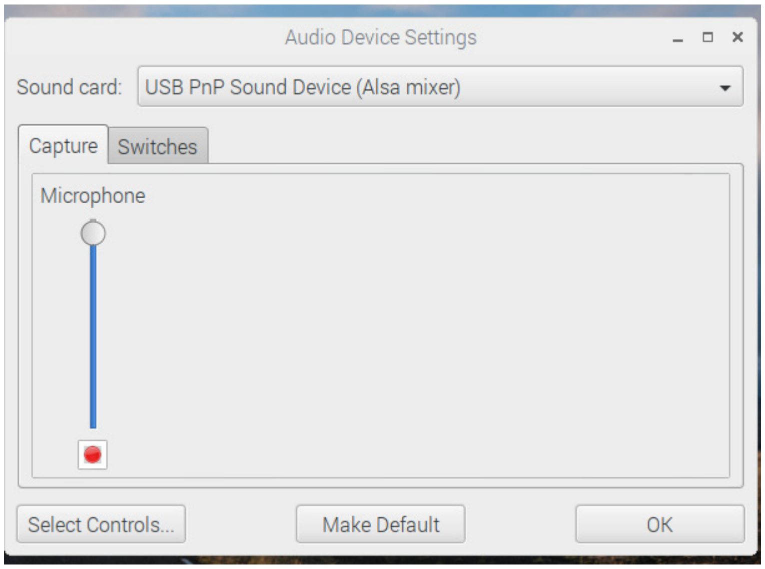
By learning to use the command line methods for audio, you have gained some technical knowledge. In some instances, you might find it more convenient to make adjustments from the desktop. To access the desktop controls, open the RPi menu and select **Preferences/Audio Device Settings**. The Audio Device Settings dialog will open. In the image to the right the controls for the built-in sound card of RPi (bcm2835) are displayed. To select another device, open the drop-down labeled **Sound card** at the top of the dialog box. Once you have the desired device selected, you need to click on the **Select Controls** button and select the controls you wish to view. In the case of the bcm2835, there is only one control, volume of playback. Here you can adjust the slider to change the volume of this playback device.


Sound card: bcm2835 ALSA (Alsa mixer)

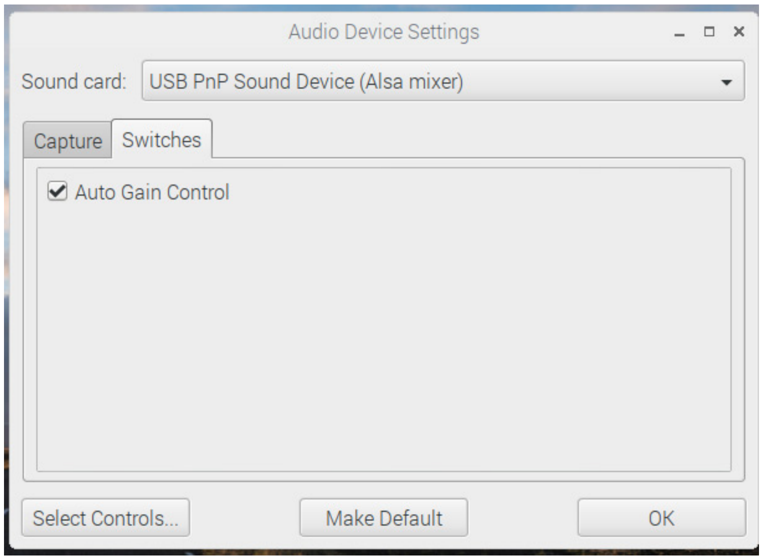
The dialog box to the right is for the USB speaker. It also has a volume control for playback that can be adjusted.



The dialog box to the right is for the USB microphone. In this case I have set two controls to be displayed (volume and automatic gain control). The **Capture** tab is selected, which contains the volume input slider. It is adjusted in this case to maximum volume input.



Here I have selected the **Switches** tab for the microphone dialog box. It has a check box for the Auto Gain Control. When AGC is on, the microphone will attempt to adjust the input volume setting to match the level of sound it is receiving. You can try this on and off to see if it makes any difference (I have not tried using it in off position). Perhaps, if the input sound is too high, with AGC on, it will auto-adjust the volume down (I am not sure).



Sound card: USB PnP Sound Device (Alsa mixer)

[✓] Auto Gain Control