How-To Guide

This guide goes over loading, decoding, and playing MP3 vs WAV sounds on the BeagleY-AI embedded linux board using ALSA and libmpg123. The guide from previous years is slightly outdated and does not properly install the Libmpg123 library on the BeagleY-AI..

Prerequisites

- GCC, make, CMake, C environment
- Have Zen Hat overlay installed on the embedded device (follow the Zen Cape Audio Guide for more information)
- ALSA library is installed (follow the Zen Cape Audio Guide for more information)

WAV vs MP3

Some key differences between the audio formats.

Reading Data:

WAV: No decoding needed (PCM data found after offset 44).

MP3: Decoded with libmpg123 library. Metadata accessible by the ID3v1 standard.

Compression:

WAV: Is an uncompressed and lossless format storing pulse-code modulation (PCM).

MP3: Is a compressed and lossy format (compressed with MPEG)

File Size:

WAV: Files are much larger because they are lossless

MP3: Files are normally smaller because of their lossy quality

Understanding PCM Data

See the lecture video for understanding PCM data representation:
https://www.youtube.com/watch?v=OiWvHj8j4RE

When trying to include both WAV files and MP3 files, you need to be careful about calculating the superposition of PCM data; the two file types may have a different number of channels. For example, the WAV files used in A3 have mono audio while MP3 files generally support stereo audio.

Based on the number of channels, the number of samples per frame will be different. In mono audio 1 frame corresponds to 1 sample. In stereo audio, 1 frame consists of both the left and right samples.

Mono Representation:

PCM: [S1, S2, S3,..., Sn]

Frame 1= S1, Frame 2= S2, … Frame n = Sn

Stereo Representation:

PCM: [L1, R1, L2, R2, L3, R3, …, Ln, Rn]

Frame 1= [L1, R1], Frame 2= [L2, R2], … Frame n = [Ln, Rn]

To add mono channel PCM data to dual channel data, you simply need to add the mono sample to both L1 and R1.

PCM Combined: [L1+S1, R1+S1, L2+S2, R2+S2, L3+S3, L3+S3, … ,Ln + Sn, Rn + Sn]

Installation For WAV

No additional libraries needed.

Installation For MP3

On the host, install the cross compiler for the libmpg123-dev library:

```
(host) $ sudo apt-get update
(host) $ sudo apt install libmpg123-dev:arm64
```

On the target, install the libmpg123-0 library:

```
(bbg) $ sudo apt-get update
(bbg) $ sudo apt install libmpg123-0
```

Ensure .so files are installed and present (note that version numbers may differ):

```
(bbg) $ ls /usr/lib/aarch64-linux-gnu/libmpg*
libmpg.so.0 libmpg.so.0.47.0
```

Playing Audio using C

This code was modified from [this student guide] and the A3 template.
The example code below is a simple example that does the following:

1. Show how to store WAV files to play properly when ALSA is configured for stereo.
2. Provide an example on how to decode, store, and play MP3 files.

It is divided into 3 sections. Configuring ALSA to play dual channel, loading and playing WAV files, and loading and playing MP3 files.

```
#include <alsa/asoundlib.h>
#include <alloca.h>
#include <mpg123.h>

#define SAMPLE_RATE 44100
#define NUM_CHANNELS 2

#define SAMPLE_SIZE (sizeof(short))

// Store data of a single wave file read into memory.
// Space is dynamically allocated; must be freed correctly!
typedef struct {
 int numSamples;
 short *pData;
} wavedata_t;

int main()
{
 // configure PCM to dual channel
-----------------------------------------------------------------
 snd_pcm_t *pcmHandle;

 // Opens the PCM output
 int err = snd_pcm_open(&pcmHandle, "default", SND_PCM_STREAM_PLAYBACK, 0);
 if (err < 0) {
   printf("Playback open error: %s\n", snd_strerror(err));
   exit(EXIT_FAILURE);
 }

 snd_pcm_set_params(pcmHandle,
         SND_PCM_FORMAT_S16_LE,
         SND_PCM_ACCESS_RW_INTERLEAVED,
         NUM_CHANNELS,
         SAMPLE_RATE,
         1,
         50000);
```

```
 // Read, store, and play WAV file
-----------------------------------------------------------
 const int PCM_DATA_OFFSET = 44;
 char* fileName = "wave-files/never_gonna.wav";
 wavedata_t wavedata;

 // Open the WAV file
 FILE *file = fopen(fileName, "r");
 if (file == NULL) {
   fprintf(stderr, "ERROR: Unable to open file %s.\n", fileName);
   exit(EXIT_FAILURE);
 }

 // Get file size
 fseek(file, 0, SEEK_END);
 int sizeInBytes = ftell(file) - PCM_DATA_OFFSET;
 wavedata.numSamples = sizeInBytes / SAMPLE_SIZE;

 // Search to the start of the data in the file
 fseek(file, PCM_DATA_OFFSET, SEEK_SET);

 // Allocate space to hold all PCM data
 wavedata.pData = malloc(sizeInBytes);
 if (wavedata.pData == 0) {
   fprintf(stderr, "ERROR: Unable to allocate %d bytes for file %s.\n",
       sizeInBytes, fileName);
   exit(EXIT_FAILURE);
 }

 // Read PCM data from wave file into memory
 int samplesRead = fread(wavedata.pData, SAMPLE_SIZE, wavedata.numSamples, file);
 if (samplesRead != wavedata.numSamples) {
   fprintf(stderr, "ERROR: Unable to read %d samples from file %s (read %d).\n",
wavedata.numSamples, fileName, samplesRead);
   exit(EXIT_FAILURE);
 }

 // Duplicate the elements of the array
 // e.g. [1,2,3,4] -> [1,1,2,2,3,3,4,4]
 wavedata_t stereoWavedata;
 stereoWavedata.numSamples = wavedata.numSamples*2;
 stereoWavedata.pData = malloc(sizeInBytes*2);

 for (int i=0; i<wavedata.numSamples; i++)
 {
   stereoWavedata.pData[2*i] = wavedata.pData[i];
   stereoWavedata.pData[2*i+1] = wavedata.pData[i];
```

```
 }
 // Play WAV audio
 snd_pcm_prepare(pcmHandle);
 snd_pcm_writei(pcmHandle, stereoWavedata.pData, stereoWavedata.numSamples /
NUM_CHANNELS / sizeof(short));

 // Read, store, and play MP3 file
------------------------------------------------------------
 mpg123_init();

 mpg123_handle *mp3Handle = mpg123_new(NULL, NULL);
 unsigned char* mp3Buffer;
 size_t mp3BufferSize;

 char* mp3Filename = "mp3-files/never_gonna.mp3";
 if (mpg123_open(mp3Handle, mp3Filename) != MPG123_OK)
 {
   perror("Failed to open MP3 file");
   return 0;
 }

 mp3BufferSize = mpg123_outblock(mp3Handle);
 mp3Buffer = (unsigned char*)malloc(mp3BufferSize * sizeof(unsigned char));

 // Gets the information. Sample code does not use the information
 int numChannels, encoding;
 unsigned int bitRate;
 mpg123_getformat(mp3Handle, (long*)&bitRate, &numChannels, &encoding);

 // Play MP3 Audio
 size_t bytesRead = 0;
 while (mpg123_read(mp3Handle, mp3Buffer, mp3BufferSize, &bytesRead) == MPG123_OK)
 {
   snd_pcm_prepare(pcmHandle);
   snd_pcm_writei(pcmHandle, mp3Buffer, bytesRead / NUM_CHANNELS / sizeof(short));
 }

 // Cleanup
------------------------------------------------------------
 free(wavedata.pData);
 free(stereoWavedata.pData);
 free(mp3Buffer);
 snd_pcm_close(pcmHandle);
 mpg123_close(mp3Handle);
 mpg123_delete(mp3Handle);
 return 0;
}
```

Compiling

Ensure that there is a linker flag in your CMakeLists.txt that points to the libmpg123 library.

```
add_compile_options(-lmpg123)
add_link_options(-lmpg123)
```

Make sure that your project makefile links the ALSA library (follow the Zen Cape Audio Guide for more information).

```
# ALSA support
find_package(ALSA REQUIRED)
target_link_libraries(music_player LINK_PRIVATE asound)
```

If you are referencing local audio files in your program, it may be a reasonable idea to copy them to the build directory.
In the following example, the program is set to read audio files from the wave-files directory:

```
# Copy the WAV folder to NFS
add_custom_command(TARGET music_player POST_BUILD
  COMMAND "${CMAKE_COMMAND}" -E copy_directory
     "${CMAKE_SOURCE_DIR}/wave-files"
     "~/cmpt433/public/myApps/wave-files"
  COMMENT "Copying WAV files to public NFS directory")
```

Troubleshooting

- No Sound Playing - Make sure an audio device is plugged in to the audio jack. There is no built-in speaker on the BeagleY-AI.
- WAV file playing at 2 times speed - Need to duplicate the mono channel PCM data
    - E.g. [1,2,3,4] -> [1,1,2,2,3,3,4,4]
- MP3 file playing at half speed - Likely configured the ALSA audio device to play using mono audio. Change the configuration to stereo audio.
- mpg123_read() != MPG123_OK on first call.
    - You can either ignore the first call to mpg123_read() and continue reading or
    - Call the mpg123_getformat() function before mpg123_read()

Additional Notes

Libmpg123 supports extracting metadata (see Extra Information from this guide)