Bearbeitung von MP3-Dateien unter Linux

Es werden folgende Programme verwendet:

- lame: Meist Bestandteil der Linux-Distribution, muss oft nachinstalliert werden.
- aplay: Bestandteil der Linux-Distribution.
- hexdump: Bestandteil der Linux-Distribution.
- gnuplot: Bestandteil der Linux-Distribution.
- play.py: Im Mp3Kit enthalten.
- sin_triangle.py: Im Mp3Kit enthalten.

MP3-Tools

16-Bit Rohdaten mit 44.1 kHz Samplerate in mp3 umwandeln

Die Datei inputdat.bin enthält die Samples in binärer Form. Pro Sample werden 2 16-Bit Worte im little-endian Format verwendet. Die 2 Worte enthalten die Werte für den linken und rechten Kanal:

```
lame -r -s 44.1 --bitwidth 16 --signed --little-endian -b 128 <inputdat.bin> <outputdat.mp3>
```

mp3 in Rohdaten umwandeln

```
lame --decode <inputdat.mp3> <outputdat.bin>
```

16-Bit Rohdaten mit 44.1 kHz Samplerate abspielen

```
aplay -t raw  -c 2 -f S16_LE -r 44100 <dateiname>
```

16-Bit Rohdaten in eine C-Datei umwandeln

```
hexdump -v -e '16/1 "0x%02x," "\n"' <dateiname>
```

16-Bit Rohdaten in eine Textdatei mit drei Spalten umwandeln

Format der Textdatei:

1. Spalte: Byte-Offset innerhalt der Rohdaten.
2. Spalte: Samples des linken Kanals.
3. Spalte: Samples des rechten Kanals.

Ofs ist die Anzahl der Bytes, die am Anfang der Rohdaten übersprungen werden sollen. Count ist die Anzahl Bytes, die für die Ausgabe verwendet werden soll.

```
hexdump -s <Ofs> -n <Count> -v -e '"%8_ad " 2/2 "%8d " "\n"' <dateiname>
```

Ver- und Entzerrung, Daten für Flash-Baustein aufbereiten

Kennline plotten

```
python play.py plot
```

Datei verkleinern:

```
python play.py  transfer <inputdatei> <outputdatei> <MaxAnzahlSamples>
```

Verzerrungskennline anwenden

```
python play.py  verzerr <inputdatei> <outputdatei> <MaxAnzahlSamples>
```

mit Entzerrer:

```
python play.py  entzerr <inputdatei> <outputdatei> <MaxAnzahlSamples>
```

Datei mit Sinus und Dreieck-Signal im Rohformat erzeugen (Ergebnis in sin_triangle.bin, Spieldauer 100 sec):

```
python sin_triangle.py
```

Eine mp3-Datei mit einer Laenge von 2 MByte hat bei einer Bitrate von 128 KBit/s eine Spieldauer von 2 MByte * 8 Bit/Byte / 128 kBit/sec = 128 sec

Bei einer Samplerate von 44.1 kHz werden dann 128sec * 44.1kHz = 5.6 Millionen Samples benoetigt.

Im rohen Format ergibt das eine Dateigroesse von etwa 5.6 Mio Sample * 2 Kanaele * 2 Byte = 21.4 MBytes.

Also: oben fuer MaxAnzahlSamples 5000000 einsetzen, dann gibt es eine Datei mit ca. 2 MByte mp3-Daten!

Vorhandene mp3-Datei (muss Bitrate von 128kbit/s und Samplerate von 44.1kHz haben) anpassen:

```
lame --decode <inputdat.mp3> raw.dat
python play.pay transfer raw.dat rawsmaller.dat 5000000
lame -r -s 44.1 --bitwidth 16 --signed --little-endian -b 128 rawsmaller.dat <outputdat.mp3>
```

Vorhandene mp3-Datei (muss Bitrate von 128kbit/s und Samplerate von 44.1kHz haben) verzerren:

```
lame --decode <inputdat.mp3> raw.dat
python play.pay verzerr raw.dat rawsmaller.dat 5000000
lame -r -s 44.1 --bitwidth 16 --signed --little-endian -b 128 rawsmaller.dat <outputdat.mp3>
```

mp3-Datei mit 440 Hz Sinus und Dreiecksignal erzeugen

```
python sin_triangle.py
lame -r -s 44.1 --bitwidth 16 --signed --little-endian -b 128 sin_triangle.bin <outputdat.mp3>
```