

Hardwarepraktikum Internet-Technologien

Task 6: Simple digital signal processing



Julius-Maximilians-Universität Würzburg

Chair of Computer Science III

A project report submitted by **group 11**

Enrique Mesonero Ronco

Manuel Calvo Martín

Pablo E. Ortega Ureña

Contents

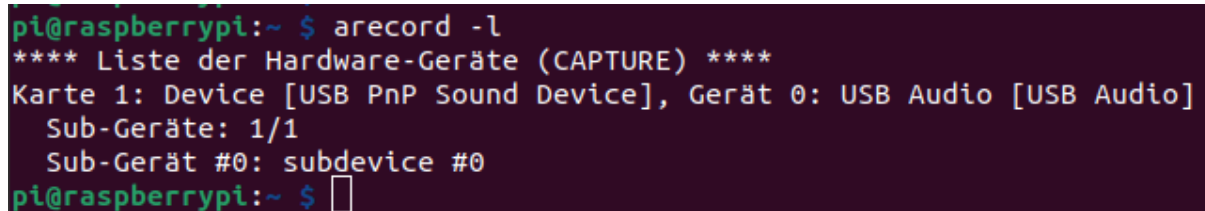
6. Single digital signal processing	3
6.2. Audio signal recording	3
6.3. Reading audio data in Python	4
6.4. Fourier transform and bandpass	4

6. Single digital signal processing

6.2. Audio signal recording

First of all, to perform the exercise we consult the arecord manual to familiarize ourselves with the command.

Then, we query the card and the unit ID with the arecord -l command.



```
pi@raspberrypi:~ $ arecord -l
**** Liste der Hardware-Geräte (CAPTURE) ****
Karte 1: Device [USB PnP Sound Device], Gerät 0: USB Audio [USB Audio]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
pi@raspberrypi:~ $
```

Figure 1

Human voice: 4000 Hz, sampling frequency 8000 Hz, based on Nyquist's theorem.

Men have deeper voices than women (different frequency, men less frequency than women).

At least 8000 Hz sampling frequency, since what we record is human voice.
Usual sampling frequency, 48 kHz, maximum sampling frequency.

Typical sampling frequency on CD: 44100 kHz.

Looking at the arecord manual, 16 Bits format, Little Endian, 44100 Hz.

Format to record: -f S16_LE -c1 -r44100 -d 5 -D hw:1,0 recording.wav
(-format 16BIT Little Endian mono channel frequency 44100 Hz
For the duration it is 5 sec, with -d.)

```
pi@raspberrypi:~ $ arecord -f S16_LE -c1 -r 44100 -d 5 -D hw:1,0 grabacion.wav
Aufnahme: WAVE 'grabacion.wav' : Signed 16 bit Little Endian, Rate: 44100 Hz, mono
pi@raspberrypi:~ $ ls
Bookshelf  Documents  grabacion.wav  Pictures  TCPclient.py  Videos
Desktop    Downloads  Music          Public    Templates
pi@raspberrypi:~ $ aplay grabacion.wav
Wiedergabe: WAVE 'grabacion.wav' : Signed 16 bit Little Endian, Rate: 44100 Hz, mono
```

Figure 2

We listened to the audio and it was very low, so we turned up the audio levels on the microphone since the speaker levels were already at maximum.

For the next section, we tried a sampling rate of 8000 Hz, and it recorded for less than 1 second, even though we set it to 5 seconds.

```
pi@raspberrypi:~ $ arecord -f S24_LE -c1 -r 44100 -d 5 -D hw:1,0 grabacionALTA24B.wav
Aufnahme: WAVE 'grabacionALTA24B.wav' : Signed 24 bit Little Endian, Rate: 44100 Hz, mono
arecord: set_params:1339: Sample-Format nicht unterstuetzt
Available formats:
- S16_LE
```

Figure 3

We also tried with 16000 Hz but it also cuts. After checking the recordings, they were still been recorded at 44100 Hz. A solution was provided in the forum, using this line: `ox input.wav -r <SampleRate> -c 1 -b <Bits> output.wavs`

We converted one of our recordings and then we confirmed that the lowest understandable sample rate was 8000 Hz

The converted files are in a separate folder : *(differentSampleRatesAndBits)*

6.3. Reading audio data in Python

We make the .py and analyze one of our recordings. We see that the number of elements of the array is 220500, that divided by 44100, gives us the 5 seconds.

```
pi@raspberrypi:~ $ python wav_analysis.py
220500
220500
[ 4.27246094e-04  3.05175781e-04  3.96728516e-04 ... -2.13623047e-04
 -3.05175781e-05  3.05175781e-05]
```

Figure 4

We multiply and normalize to stay in the $[-1,+1]$ range. We get the same signal shape, but with higher amplitude. We convert the new signal to a .wav file and we hear the difference, it is much louder.

The pdf files containing the plot, along with the recordings, are in *grabacionALTA_Sample* folder.

6.4. Fourier transform and bandpass

We perform the Fourier Transform to our signal. We plot the frequency domain for both signals, then we try to filter the amplified signal using a low-pass filter but we are not able to obtain a good result. All of our tries sound like it is underwater or canned when we convert the array using the inverse transform and writing a new .wav file.

Again, all of the plots and the resultant .wav file is in the *grabacionALTA_Sample* folder.

We also analyze two files from the Forrest Gump movie, obtaining similar results. The resultant files are on the *gump_Sample* and *virus_Sample* folders.