# Tecnologías Multimedia - Study Guide - Milestone 4: "wiring" the ADC with the DAC and measuring latencies

Vicente González Ruiz - Depto Informática - UAL

September 28, 2020

# 1. Description

Let's start our development of InterCom with a simple program that inputs digital audio (through the ADC of your computer) and, as soon as possible, outputs it (through the DAC). We are going to use a minimal "wiring" program, that when it is run and we feed back the audio output with the audio input of our computer, allows us to measure its latency. To handle the audio hardware we use `sounddevice` [1], that is wrapper for the PortAudio library.

# 2. What you have to do?

1. Refresh some (probably high-school) ideas about The Sound, The Human Auditory System, and The Human Sound Perception. Notice that we are able to perceive sounds whose frequency ranges between 20 Hz and 20 KHz (approximately), and that a speaker is basically a microphone, and viceversa.

2. Download the Python module wire3.py with:

```
sudo apt install curl
curl https://raw.githubusercontent.com/Tecno
```

   This module implements the algorithm:

$$\begin{cases} \texttt{CHUNK\_SIZE} \leftarrow 1024 \\ \textbf{while true} \begin{cases} \texttt{chunk} \leftarrow \texttt{stream.read}(\texttt{CHUNK\_SIZE}) \\ \texttt{stream.write}(\texttt{chunk}) \end{cases} \end{cases}$$

   where (1) captures `CHUNK_SIZE` frames from the sound card, and (2) plays the chunk of frames `chunk`. In `sounddevice` a frame is

a collection of one or more samples (typically, a frame is a single sample if the number of channels is 1, or two samples if the number of channels is 2).

3. If you want to run this module right now, check first that the output volumen of your speakers is not too high, otherwise you could "couple" the speaker and the mic(rophone) of your computer, producing a loud and annoying sound. In order to mitigate this effect, you can also control the record gain of your mic (if the gain is 0, no feedback between the speaker and the mic will be possible). In Xubuntu, these controls are available by clicking in the speaker icon (situated in the top right corner of your screen) of the Xfce window manager.

4. OK, run it (remember to activate first the virtual environment `tm` before using Python), with:

```
pip install sounddevice
python wire3.py
```

5. Now, lets compute, experimentally, the `wire3.py` latency.

(a) First, we need the tools SoX, Audacity and wire3.py:

```
sudo apt install sox
sudo apt install audacity
sudo apt install curl
curl https://raw.githubusercontent.com/T
```

(b) Run:

```
pip install matplotlib
python plot_input.py
```

and check that the gain of the mic does not produce clipping during the sound recording.

(c) In a terminal, run:

```
python wire3.py
```

while you control the output volume of the speakers to produce a decaying coupling effect between both devices (the speaker(s) and the mic). If your desktop has not these transducers, we can

use a male-to-male jack audio cable and connect the input and the output of your sound card.

(d) In a different terminal (keep `python wire3.py` running), run:

```
sox -t alsa default test.wav
```

to save the ADC's output to a file.

(e) While `sox` is recording, produce some short sound (for example, hit your laptop or your micro with one or your nails). Do this at least a couple of times more, to be sure that you record the sound and the feedback of such sound. It's important the sound to be short (a few miliseconds) in order to recognize it and it's replicas.

(f) Stop `sox` by pressing the CTRL and C keys (CTRL+C) at the same time. This kills `sox`.

(g) Stop `python wire3.pt` with CTRL+C.

(h) Load the sound file into Audacity:

```
audacity test.wav
```

(i) Localize the first one of your nail-hitting-the-mic sounds in the audio track.

(j) Select the region with your sound and a replica using our mouse.

(k) Use the zoom to selection buttom to zoom-in the selected area.

(l) Measure the time between the ocurrence of the sound and it's first replica. This time is the real latency of your computer runing `wire3.py`.

(m) Modify the constant `CHUNK_SIZE` and repeat this process, starting at the Step 5c. Create an ASCII file (`latency_vs_chunk_size.txt`) with the content (use tabulators to space the columns):

```
# CHUNK_SIZE    real
32              ...
64              ...
128             ...
256             ...
512             ...
```

```
1024            ...
2048            ...
4096            ...
8192            ...
```

6. At this point, we know the real latency of `wire3.py` as a function of `CHUNK_SIZE`. Plot the file `latency_vs_chunk_size.txt` with:

```
sudo apt install gnuplot
echo "set terminal pdf; set output 'latency_vs
```

7. Let's compute now the buffering latency of a chunk (the chunk-time). If `sampling_rate` is the number of frames per second, it holds that:

$$\texttt{buffering\_latency} = \texttt{CHUNK\_SIZE}/\texttt{sampling\_rate} \quad (1)$$

   Add these calculations to `latency_vs_chunk_size.txt` using a third column (remember to use TABs).

```
# CHUNK_SIZE    real    minimal
:               :       :
```

8. Plot both latencies:

```
echo "set terminal pdf; set output 'latency_vs_
```

9. Which seems to be the minimal practical latency (the latency obtained when `CHUNK_SIZE` $= 1$) in your computer? Justify your answers.

# 3. Timming

You should reach this milestone at most in one week.

# 4. Deliverables

Answer the question enunciated in the Step 9.

# 5. Resources

[1] python-sounddevice.