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

Vicente González Ruiz - Depto Informática - UAL

March 13, 2022

# 1. Description

Let's start our understanding of InterCom with a simple program that inputs digital audio (through the ADC of your computer) and, as soon as possible, outputs the audio (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 the latency of this system. 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.

2. Download the Python module wire3.py with:

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

   This module implements the algorithm:

```
while True:
    chunk = sound_card.read(chunk_size = 1024)
# (1)
    sound_card.write(chunk)  # (2)
```

   where (1) captures 1024 frames from the ADC, and (2) plays the chunk of frames. 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 and you are not using a headset, check first that the output volumen of your speakers is not too high, otherwise you could involuntary "couple" the speaker and the mic(rophone) of your computer, producing a loud and annoying tonal sound. In order to mitigate this effect, you can also control the 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 through clicking in the speaker icon (situated in the top-right corner of your screen) of the Xfce window manager.

4. OK, run the module with:

```
pip install sounddevice  # Only once
python wire3.py
```

   Stop (killing) the module by clicking the CTRL- and c-keys (CTRL+c), simultaneously.

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

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

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

(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 noisy 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 line-output of your soundcard to the input 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 the file `test.wav`.

(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 also the feedback of such sound. It's important the sound to be short (a few miliseconds) in order to visually recognize it and it's replicas.

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

(g) Kill `python wire3.pt` with CTRL+c.

(h) Load the sound file into Audacity:

```
audacity test.wav
```

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

(j) Select (using the mouse) the region that contains your sound and a replica.

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

(l) Measure the time between the ocurrence of the hit (of the nail) and the recording of its first replica produced by the speaker-to-the-mic feedback. This time is the real latency of your computer runing `wire3.py`.

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

```
# CHUNK_SIZE    real
32              ...
64              ...
```

```
128                 ...
256                 ...
512                 ...
1024                ...
2048                ...
4096                ...
8192                ...
```

with the real (practical) latency.

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_ch
```

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

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

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_chu

9. Which seems to be the minimal practical (real) latency (the latency obtained idealllly when $\texttt{CHUNK\_SIZE} = 1$ ... however, notice that depending on your computer, this chunk size can be too small, overwhelming the CPU) in your computer? Justify your answers.

# 3. Timming

You should reach this milestone at most in one week.

# 4. Deliverables

Describe your experimental setup (CPU processor, RAM, soundcard chipset, etc.), plot the data, and answer the question enunciated in the Step 9.

# 5. Resources

[1] M. Walker. python-sounddevice.