How to calibrate with an oscilloscope?

1. connect the one channel of the oscilloscope to the aux output of the computer (of course the one you will use for the experiment)

2. connect and plug the photodiode to another channel. Place the photodiode (a little white device on the screen where the visual stimulus will appear. Hold it in position (you can stick it or just press it). Make sure the receptor (a little glass hole) is centered on the target.

3. calibrate the oscilloscope:

    a. set the optimal noise-to-signal threshold so that only the target stimulus will trigger the recording

    b. select which channel is the trigger. For example, if you set the audio channel as trigger the recording will start only if the audio stimulus breaks the audio threshold, no matter what happens in the video channel.

4. start the recording (single sequence for one event only, continuous run for many events)

5. run the stimuli

How to read the results

The oscilloscope will record the event or events starting from the trigger, displayed as two colored waveforms on screen. You can now choose the time resolution at which the waves are displayed, and you can move horizontally along the recording. You can also move separately the different waves on screen (up-down). You will have a clear, graphic representation of the time order of the events: onset and offset of each one, together with their intensity.

If you want to enhance a weak signal you can amplify it from its channel control panel.

Time precision in Psychopy

By its default setting Psychopy has highly precise visual output and terrible audio output. This is well known and people discuss it on forums. "Terrible" means that audio latency wanders between 150-250 ms and duration of stimulation can vary up to more/less 100 ms than what you wrote on the script.

Here some tips to achieve a reasonable precision in audio output.

Choose the right library

By now (version 1.82.01) Psychopy has two audio libraries: Pygame and Pyo. By no means Pyo is more accurate, but Pygame is set as default. To set Pyo as default go on Settings>audio library and invert the order of the libraries, from:

[u'pygame', u'pyo']

to:

[u'pyo', u'pygame']

This will solve half of your problems, bringing the differential audio/video latency to a consistent (more/less 10ms) 40 ms. Same for duration accuracy. The other half comes from the fact that Pyo is still a bugged library which gives a lot of compatibility problems. For what I've seen so far, when running an experiment with Pyo the "Esc" command to stop the experiment may not work. Instead, the program would get stuck and you will have to shut it down from the Task Manager (Ctrl+Alt+Canc). Otherwise, it may close and send you an error message. There might be a solution to this bug and I'm looking for it in the forums.

Choose the right audio driver

Again, the default setting is PrimarySounds. You'd better change it to ASIO, which is supposed to be more precise. The ASIO4all driver, if compatible with our audio card, is supposed to be the best and we might consider installing it. As compared to the other possible drivers (PrimarySounds and Audigy), ASIO combined with Pyo is not more precise but more reliable, which is very important cause it gives you a known latency that you can correct.

Set times in frames

Frames are more accurate than seconds, which are always a rounding. So go over your script and GUI and set times, where possible, in frames (mind frame rate!!). So, for example, at 144Hz refresh you're your times will be:

1s = 144 frames

-40 ms = -5 frames (audio latency)

200 ms = 29 frames

700 ms = 101 frames

Although the best strategy would be to ask Psychopy to calculate the equivalent time in frames from time in seconds by taking a sample of frame rate every routine (frame rate is slightly oscillating) this may overload the system and generate an even bigger latency! So it's better to set the times directly in frames.

You will see that in the GUI (builder view) the audio duration cannot be set in frames, so you have to keep it in seconds: you can only set onset time in frames. Also, temporal properties of visual stimuli, as measured with the oscilloscope, are highly accurate both in frames and in seconds so all this is crucial for audio only.

Play audio files, don't produce them!

Another useful trick is to produce (matlab, python,...) a .wav file with the audio you want to play and the duration it must be, rather than defining it in psychopy: producing and stopping it is more energy (and time!) consuming than simply playing a recorded file from beginning to end. Our test session showed perfect duration accuracy for .wav files, as opposed to the +- 15 ms of produced waveforms.

So what's the best performance Psychopy can give you?

With the tricks I've discovered so far I achieved a consistent latency of 40ms (+- 15ms) with less than one frame variance (around 6 ms), which can be efficiently corrected on the code, and a perfect duration accuracy on .wav files, as measured both on my experiment which includes time variable and on a test experiment (much simpler with fixed numerical times). **So, you will achieve an inaccuracy of maximu more/less 15 ms for the temporal properties of your audio stimuli.**

Mind the machine!

All the above is true for our machine (Display1). A test session at Ahissar's lab, for example, showed that Pyo there doesn't always run, or if it runs it is with Audigy driver only and with 500ms latency!!