SWANSEA UNIVERSITY

Audio Switch

The design, development and results of a bespoke audio switch.

by
Gary Freegard
Department of Psychology
24/01/2013

The Requirement.

This design came about due to the particular requirements of a researcher. This was to produce a 7 millisecond pulse of sound in either the left or right ear, at either the start or end of a visual presentation, see figure 1 and 2 below.

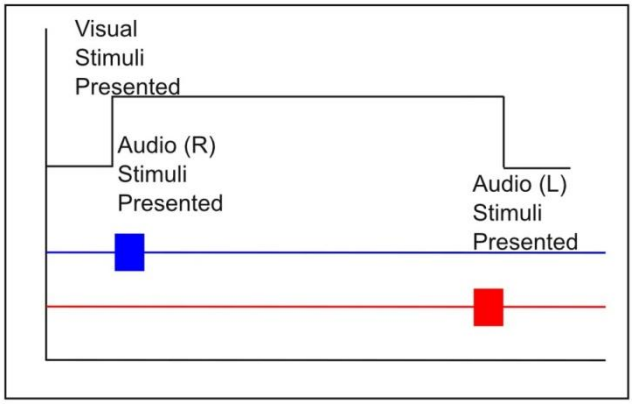

figure 1

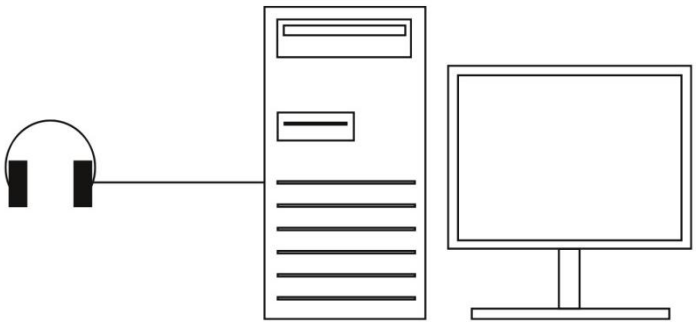

figure 2

During an investigation into the timings of this particular visual/audio presentation, it became apparent that the hardware and software being used was not capable to deliver the audio signal at the correct time.

The Problem.

The delay from the activation of the audio stimuli and the actual output was approximately 90 milliseconds, measured on a Dell Precision 390(Windows XP) with onboard sound card and using Opensesame as the presentation software. A similar test was carried out on another PC (Dell Optiplex 755) using Opensesame, and this was able produce better results of approximately of 34 milliseconds (using onboard sound card), taken from the average in table 1, and 60 milliseconds (using separate sound card), though these still do not meet the requirement.

The statistical results are shown in table 1, these are in milliseconds, n = 50;

Min	28.004
Max	41.837
Average	34.720
Stdev	3.576

table 1

It was found that the test code could be modified to compensate for the 34 milliseconds delay (average in table 1) and the statistical results are shown in table 2 below, and are in milliseconds, n = 50;

Min	-3.553
Max	7.933
Average	2.481
Stdev	3.304

table 2

The negative indicates activation of the sound prior to the presentation of the visual stimuli.

Clearly this compensation gives a better result though still not ideal.

Below is the code used to compensate for the delay.

```
1  win.flip()                  waits for vertical sync
2  io.Out32(0x378,3)           output on parallel port
3  right_sound.play()          play sound
4  self.sleep(soundDelay-8)    wait 26 millisecond
5  win.flip()                  waits for vertical sync
6  fixdot.draw()               draws visual stimuli
```

The method used to compensate for these delays relies on the use of sleep command; unfortunately this command, does not accurately or precisely produce the correct delay due to the operating system capabilities. So the variation shown in table 1 is caused by a combination of operating system and hardware latencies.

For comparison a similar test was carried out using E-Prime, the results are shown in table 3, below.

Min	8.377
Max	14.829
Average	11.567
StDev	1.496

table 3

The test did not include any compensation for any average delay, as this would have reduced the average. Comparing tables 1 and 3 one can deduce that timing of the E-Prime audio system is significantly better than the Opensesame system.

The Solution.

Since there was no option to use E-Prime because of the researchers requirements, I decided to look at the possibility of using an external audio switch that would be controlled by the PC.

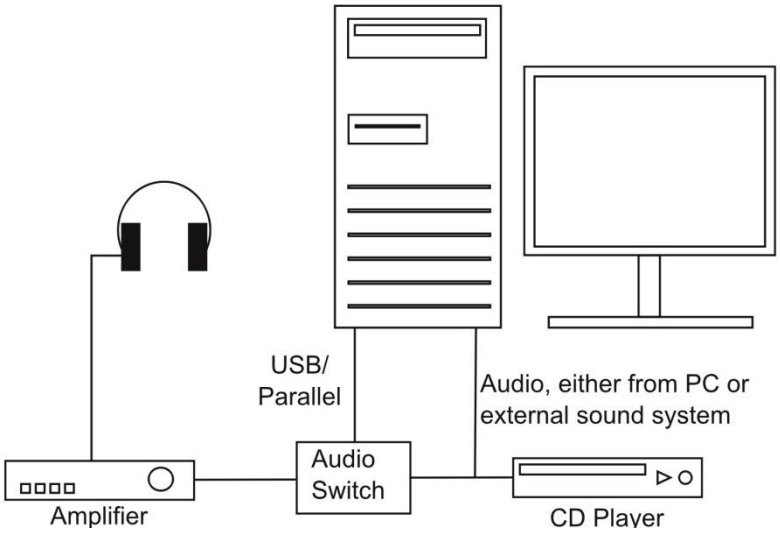

figure 3

A simple CMOS multiplexer was selected for testing. Unfortunately this was not ideal because the audio channels could not be selected separately but it was successful for proving the concept. This multiplexer was connected to the test PC and was controlled using the output from the parallel port.
A different multiplexer was sourced that was capable of selecting each audio channel separately, thus allowing for greater flexibility. At the same time it was decided to add USB connectivity by incorporating a microcontroller. The reasons for this are:

The parallel port has disappeared from laptops and is disappearing from desktops; it is being replaced with USB.

It allows the device to have multiple methods of operation, which the user can select with a bespoke program, or even during the test using the presentation software.

Table 4 below shows the results when using the parallel port, and the timings are in microseconds (a thousandth of a millisecond); n = 50.

Min	53.000
Max	313.000
Average	177.667
StDev	74.304

table 4

This solution provides an average/accuracy of less than 0.2 milliseconds.
Table 5 below shows the results when using USB (Serial), and the timings are in microseconds (a thousandth of a millisecond).

Min	87.000
Max	1100.000
Average	626.875
StDev	294.057

table 5

This solution provides average/accuracy of approximately 0.6 milliseconds.

The benefit of using a microcontroller with USB connectivity is the ability to add and select different functionality. This ability has been used by adding two modes of operation, these are:

Follow: the audio switch will mirror the condition of the parallel port i.e. if the parallel port is on then the audio channel is on, when off then the audio is off.

Pulse: the audio switch is enabled after a predetermined fixed delay for a predetermined fixed duration. Both the delay and duration is user defined , they can be set from 0 to 65000 microseconds and 1 to 9999 milliseconds , respectively, via USB.

The last selected mode, delay and duration is stored even after power is removed, so that on power up the last mode will be used.
When the unit is in pulse mode the duration of the pulse is significantly more accurate than is achieved when using Opensesame or E-Prime, the pulse duration was seen to vary from 2 to 10 milliseconds for what is meant to be a 7 millisecond pulse whilst with the audio switch this will produce a 7 millisecond pulse consistently (within 1 per cent). The delay is used to compensate for screen latencies; this is the time difference between when the screen is refreshed in the PC to when this screen is actually shown.

All the tests of the audio switch were carried out using an external sound source, a CD playing a 3.5KHz stereo tone.
It is possible to use the test PC to produce the required audio but if this is done within the presentation software then the inherent delay must be taken into account i.e. using Opensesame then the sound playback must be started at least 42 millisecond (table 1 max)prior to use.

Actual Results

An optical sensor was used to measure the monitor screen latencies, see figure 4 for positions
Table 6 shows the latency from parallel port to CRT in the top position on the monitor whilst table 7 shows the latency for the centre position. The tables are in milliseconds and n=50.

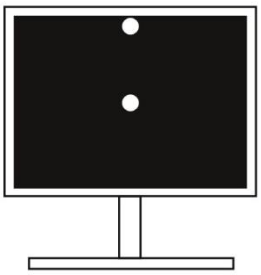

figure 4

Min	9.444
Max	9.500
Average	9.484
StDev	0.009

table 6

Min	13.801
Max	13.831
Average	13.819
StDev	0.008

table 7

Figure 5, below, shows the data collected from a test using the audio switch and a CRT monitor (LG Flatron F900P 1280 x 1024 @ 100Hz). The delay was set to 13.8 milliseconds and duration to 7 milliseconds. The green line indicates the state of the parallel port, which is triggered when the screen is refreshed. The red line shows the state of the visual stimuli on the screen (at the centre). In this setup there is a 13.81(table 7 average) millisecond delay between PC screen refresh and monitor screen refresh, this is made up of 5 milliseconds as the point of measurement was the screen centre and since the screen is refreshed every 10 milliseconds it takes half of this to refresh to the centre of the screen, and 9.48(table 6 average) milliseconds caused by hardware (CRT or PC). The blue line is the audio output, which is delayed to start just as the CRT screen centre is refreshed.

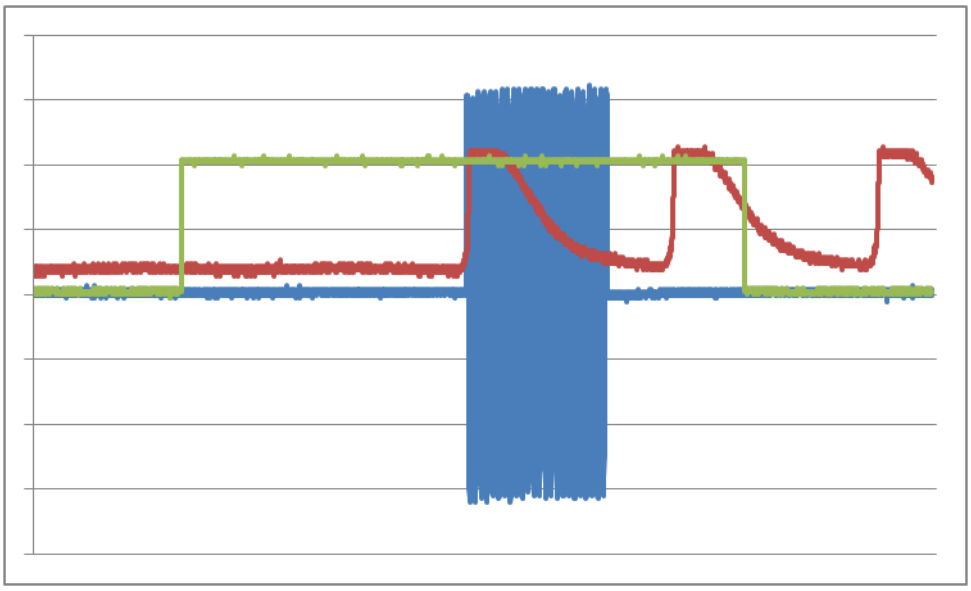

figure 5

Conclusion.

This audio switch offers the user the ability to produce an audio tone, within 1.1 milliseconds of the required start time and with a duration which is ±0.5% of the require delay and duration (when in pulse mode).

	Audio Switch			Openseame	
	Parallel	USB	E-Prime	Comp	Uncomp
Average	0.178	0.627	16.314	2.481	34.720
StDev	0.074	0.294	1.441	3.304	3.576

table 8

Table 8 shows the average (accuracy) and standard deviation (precision), of all the tests carried out.

	Audio Switch			Openseame	
	Parallel	USB	E-Prime	Comp	Uncomp
Average	1	3.53	91.82	13.96	195.42
StDev	1	3.96	19.40	44.47	48.12

table 9

Table 9 shows the magnitude of difference when the method is compared against the audio switch (parallel port), i.e. the audio switch (parallel) is almost 200 times more accurate and almost 50 times more precise than using Opensesame (uncompensated).

It can clearly been seen that using this method significantly improves the temporal resolution of audio playback, over the standard methods used for testing.

The Future.

The inclusion of a headphone amplifier see figure 6 below.

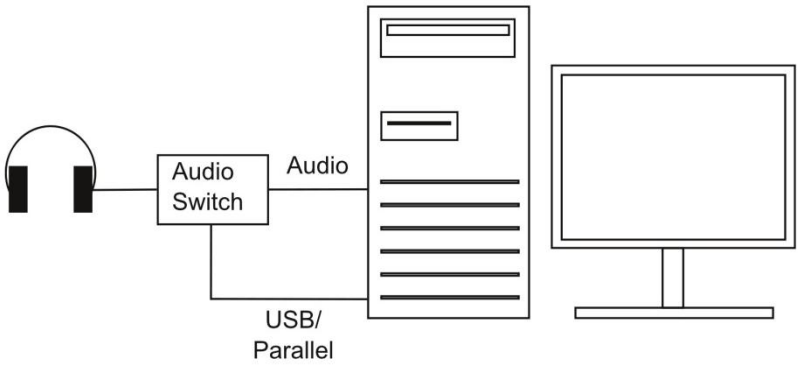

figure 6

Modification of the microcontroller software to allow the connection and control by either USB or parallel together, currently when the USB is connected then all control is through this.

Possible other modifications:

Different delays/durations for left and right channels, selectable by use of different trigger codes i.e. parallel port has 255 different combinations.

Audio switch activation by optical sensor, for use when there is no parallel port and USB is not accurate enough.

A predetermined pattern of activation of left /right channels, triggered by parallel, USB or optical i.e. left activation for 10ms, delay 200ms then right activation for 20ms.