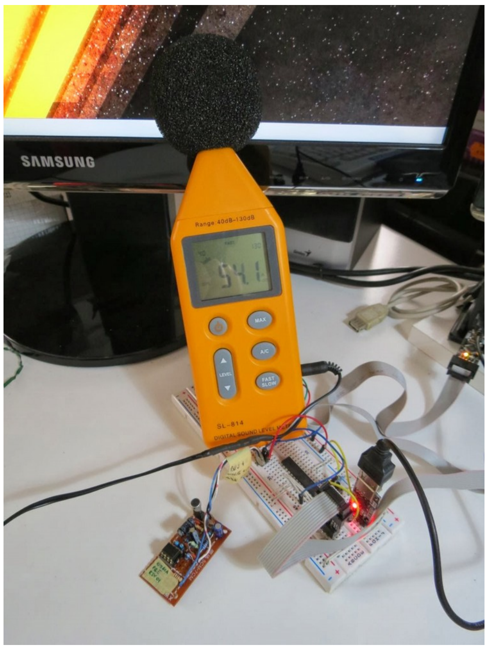
A simple Sound Pressure Level Meter (SPL) dB audio meter using AVR ATmega

Updated to version 02.

What is proposed here is a SPL db meter using and AVR Atmega micro.



A **sound level meter** or sound meter is an instrument which measures sound pressure level. Sound pressure level (SPL) or sound level is a logarithmic measure of the effective sound pressure of a sound relative to a reference value. It is measured in decibels (dB) above a standard reference level. The commonly used reference sound pressure in air is = 20 µPa (rms) which is usually considered the threshold of human hearing. Keep in mind that 1 pascal will equal an SPL of 94 dB. Because the frequency response of human hearing changes with amplitude, a weighting have been established for measuring sound pressure. Usually the A-weighting curve is used. A weighting curve is a graph of gain across the frequency range (10Hz to 20kHz).

SPL level is defined as

$$SPL\_db = 20\log_{10}\left(\frac{p\_rms}{p\_ref}\right)$$

given *p_rms* as the sound pressure measured, and *p_ref* as the reference sound pressure.

Once we have got the RMS value of the signal (*actualRMS*), we can transform it to *SPLdb* using this formula:

$$SPL\_db = 20\log_{10}\left(\frac{actualRMS}{refRMS}\right) + refSPLdb$$

given *refRMS* as the reference RMS value for the input board at a know *refSPLdb* SPLdb level.

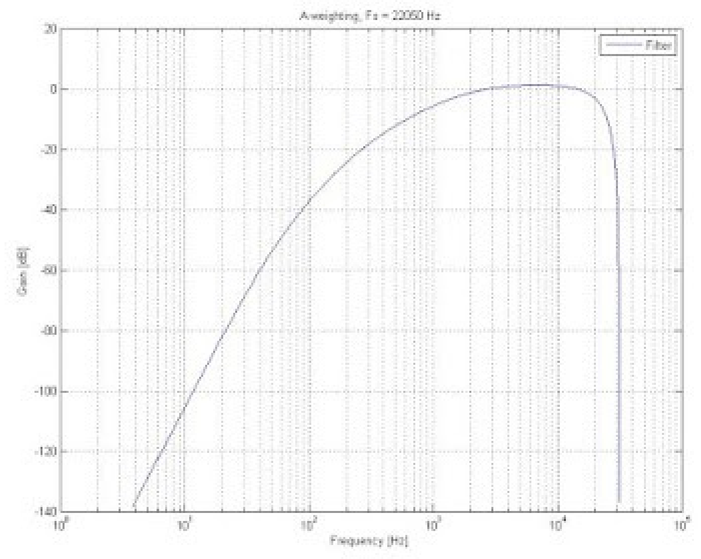
To compute SPL measurements, the meters loop is:

1. collects N samples
2. do FFT for the *N* samples collected, the signal is now transformed in the frequency domain
3. apply A-weighting (in freq domain)
4. get magnitude of the signal
5. get RMS value of the signal
6. apply a time-weight filter to RMS value
7. compute the SPL using the RMS value
8. output data

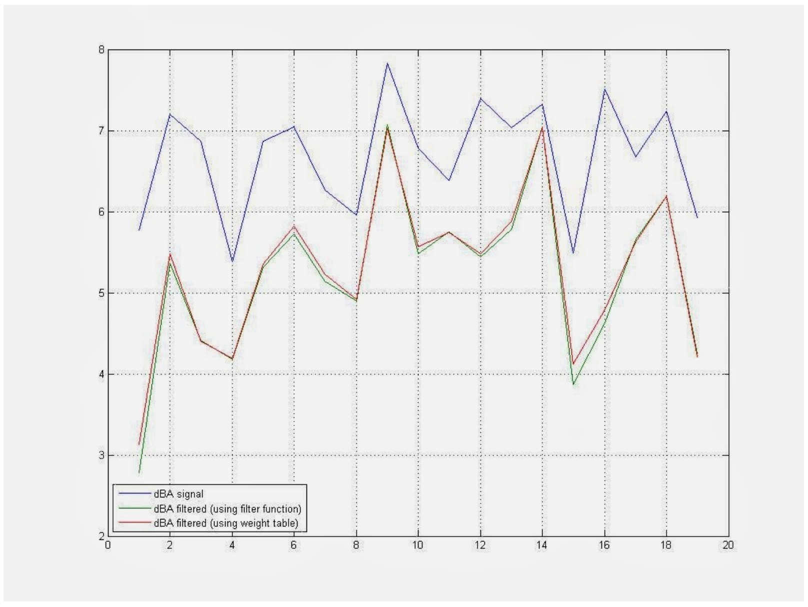
Every sample is collected at a fixed time, a timer interrupt impose this timing, this is because we need to know the sampler frequency, to built filters and output signal magnitude.
Runnig @16Mhz i'm able to collect samples at almost 22050Hz.

For FFT i've used **Radix-4 FFT library** you can find it
here http://davidegironi.blogspot.it/2013/06/avr-atmega-audio-input-rma-using-fft.html.

The method to weighting the signal proposed here just use a **weight table** that contains the weight of the signal in the frequency domain, this table shoud be FFT size/2, because we can retain frequencies below Nyquist rate.



A matlab script to check this filter method is used.
The picture above is the frequency reponse graph of the **A-Weighting** curve used.
The picture below compares the values of SPL for the unfiltered signal (blue), A-weighting filtered signal using the matlab "*filter*" function (green), A-weighting filtered signal using the weight table method (red).
As you can see, matlab native "*filter*" function and weight table method almost response in the same way.



Once we have got the A-weighted signal we can compute the RMS value for the signal.
To obtain RMS value of the AC current of the audio signal processed, from which you can get the SPL db value, we use the **Parseval's theorem for FFT** so for fft size of N (look at the Radix-4 FFT library link above for further information)

$$RMS\_freq = \sqrt{\frac{2}{N^2} \sum_{i=0}^{N/2} ReX[k]^2 + ImX[k]^2}$$

An SPL meter also needs the signal to be time-weighted. **Time-weightings** have been internationally standardised, 'S' (1 s) originally called Slow, 'F' (125 ms originally called Fast and 'I' (35 ms) originally called Impulse.
A time-weight is a **RC filter, exponential averaging** is equivalent to a simple RC filter

$$Y = (1 - alpha) * Y + alpha * Ynew$$

Y is the RMS signal global, and *Ynew* is the RMS signal computed every step.
alpha effects the filter speed, it has to be a number between 1 and 0.
We can define *alpha* so that it depends on the sample frequency *Fs* and the windows size *W*. A good estimation could be

$$alpha = \frac{\frac{W}{1000}}{\frac{Fs}{N} + \frac{W}{1000}}$$

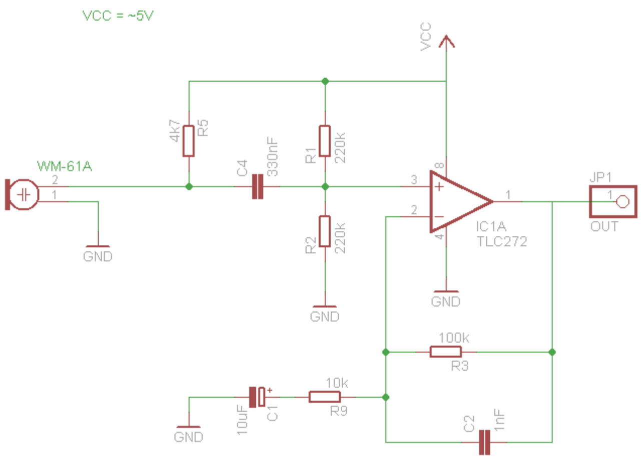
In my code, i've use another estimation for *alpha*, it seams to me better to consider the number of cycles needed to collect the *N* samples, i've call this number *fsc*, said this, my *alpha* was estimated as

$$alpha = \frac{\frac{W}{1000}}{fsc + \frac{W}{1000}}$$

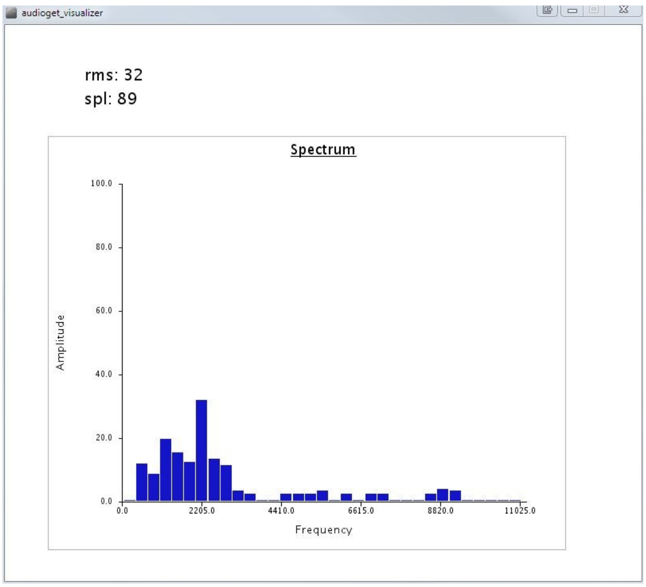
Now that the signal is windowed, we can compute SPL using the above definition:

$$SPL\_db = 20\log_{10}(\frac{actualRMS}{refRMS}) + refSPLdb$$

The **input microphone preamp board** proposed here is a derivation of the John Conover WM-61A preamp. The reference is the schematic-6 found here: [Using the Panasonic WM61A as a Measurement Microphone (John Conover)]().
This board is stated 306uV RMS at ~32db.
Using an ebay SPL meter, and a scope, i've found that my board implementation should be 315uV at ~32db (which is next to 306uV as stated in the referecence audio board schematics).



Two [http://processing.org/]() scripts are provided as a **viewer** for the data.
The screenshot below shows one of them, it also show you the spectrum of the input signal.


audioget_visualizer

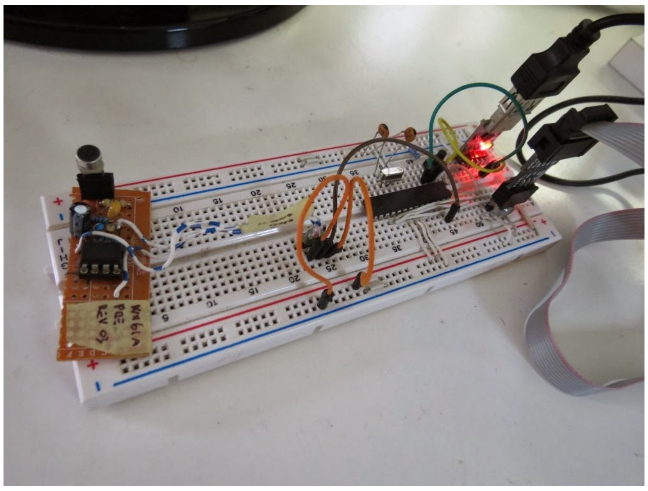
rms: 32
spl: 89

I'm not a DSP expert, so method proposed here can be improved, for example it is possible to implement the A-weighting filter in time domain, becuse it can be used as a IIR filter, this will prevent the use of the FFT computation for the signal, saving microcontroller resources.

Also, if you want to help me, send me feedback if you:

- you've got any better input board circuit
- you've found bugs in my software implementation

This project is build on an ATmega8 running at 16Mhz, compiled with avrgcc, using Eclipse IDE



ChangeLog

- 02b: main.c strict compile error fixed, function audioget_getrmsval now called the right way (thanks to *Emmanuel Pierre* for reporting this bug)
- 02: weighting function bug fix, weighting now applied on both the real and imaginary part of the fft (thanks to *Xiaofeng* for reporting this bug).
- 01: first version.

Code

- avr_splmeter_02b.zip
- ~~avr_splmeter_02.zip~~
- ~~avr_splmeter_01.zip~~

Notes

- read risk disclaimer
- excuse my bad english