Discrete-Time Signal Processing
Online Filtering on Embedded Hardware

Oliver FUNK, Johan JANSEN VAN VUUREN, James CUSHWAY
FNKOLI001, JNSJOH014, CSHJAM001

EEE4001F
Class of 2017
University of Cape Town
South Africa

May 2017

1 Introduction

Filtering forms an integral part of signal processing. It involves the manipulation of signals using some process, which can be described mathematically, to produce altered versions of said signals. This process can reveal information contained in the signals that was not easily detectable before or produce desired alterations to the signal.

Filtering often needs to be performed in real time, due to the time constraints posed by systems making use of the filtering[7]. While analogue filters naturally perform real time filtering, there are several delays associated with digital filtering. The delays are mainly associated with sampling of the signal, and the calculations that need be performed for the digital filtering of the signal. These effects can be mitigated somewhat by means of a higher clock speed, and efficient calculations to reduce the time complexity of the filtering algorithm, such as the fast Fourier transform (FFT).

To implement a digital filter, analogue input signals must be sampled and digitised using an ADC. The sampled data is then fed to and processed by a digital device. Finally, the output of the processing is sent to a DAC where it is converted back to an analogue signal.

This project aims to investigate several topics relating to FIR filters. Several FIR filter design techniques will be used to synthesize highpass and lowpass filters using MATLABs FDAtool. The simulated and actual response of each filter will be compared, for signals in the audible range, limiting the frequencies of interest to the 20 Hz - 20 kHz range.

The STM32F4 discovery microcontroller will be used to implement the various digital filters used throughout the project.

2 Digital filtering techniques

Digital filtering has two broad categories of filters, finite impulse response (FIR) filters and infinite impulse response (IIR) filters. FIR filters are those which have a finite impulse response, while IIR filters have an infinite impulse response. This project deals only with FIR filters and thus only FIR filter design techniques will be discussed. There are numerous design techniques that exist for the implementation of FIR filters. The chosen designs for this project are Equiripple, Least-squares and a Hamming window filter [4], as these are the some of the most popular filtering techniques used in modern digital signal processing.

Equiripple filtering makes use of ParksMcClellan algorithm [5]. The ParksMcClellan algorithm is an iterative approach to digital filter design whereby the algorithm is supplied with a desired frequency response (pass and stop band), a weighting function for a maximum allowable error and a filter order (N). The algorithm then finds a set of N+1 coefficients that minimises the maximum deviation from the ideal response. The algorithm achieves this by choosing an initial extremal set of frequencies (maximum and minimum points within the stop and pass band) and then performs polynomial interpolation on the given points, yielding an N+1 order polynomial of frequencies. The algorithm then computes an error function, $E(\omega)$, between the given response and desired response and a minimum error value, δ, calculated from the difference between the obtained and desired extremal set values, depicted in figure 1 below.

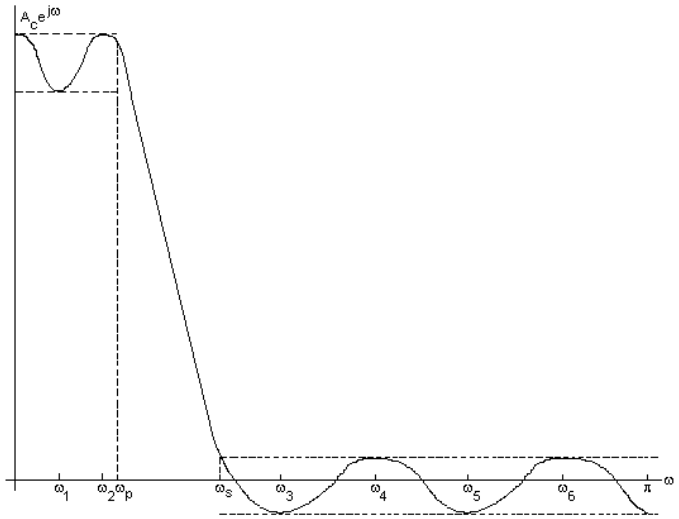

Figure 1: Error calculation between given and desired frequency response

The image depicts the pass band and stop band frequencies, ω_p and ω_s respectively. The local maxima of the error function, $max_\omega|E(\omega)|$, is then computed, and if $max_\omega|E(\omega)| > \delta$, the extremal set of frequencies is updated by picking new frequencies where the local maxima of $E(\omega)$ lies. This is repeated until $max_\omega|E(\omega)| < \delta$. Finally, the filter coefficients are obtained by use of an inverse discrete Fourier transform.

The least squares filtering technique [1] is similar to the Equiripple technique in that it too uses an algorithm to find filter coefficients that produce a minimum error for a desired response from given input parameters. The least squares technique is also an iterative approach that begins with a set of initial filter weights, and an error function calculated. However, thereafter the algorithms begin to differ. The least squares method makes use of the gradient descent algorithm, a first order iterative optimization algorithm which finds a local minima of a function. The error function is a multi-variable function of the filter weights, and thus examining the derivative of the error function depicts how to alter each weight in order to reduce the error, by moving down or up its slope to decrease or increase the error respectively. Gradient descent makes use of this by altering each frequency in small incremental steps to move down the slope of the derivative error function in attempt to find the local minima, depicted in figure 2 below.

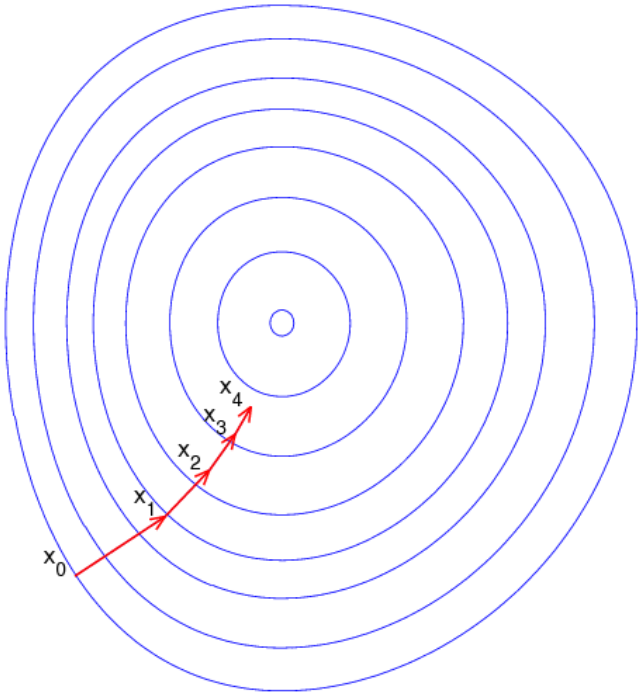

Figure 2: Gradient descent along an error function

The error function is shown as a contour plot, and the incremental steps shown slowly moving towards the minima of the function. Once the algorithm reaches the minima (a gradient of 0), the algorithm is complete and the filter tap weights are returned.

The final digital filtering technique, the windowing method, is unlike either of the above-mentioned techniques as it is not an algorithm. It makes use of an IIR filter designed to meet the frequency specifications. The IIR filter is then truncated by multiplication of a window function in the time domain. The result is a convolution in the frequency domain with the frequency spectrum of the filter and window function. If the window function has a narrower side lobe, the frequency response of the truncated IIR filter remains closer to its original ideal response. The window function used in this project is known as the Hamming window, shown in figure 3 below.

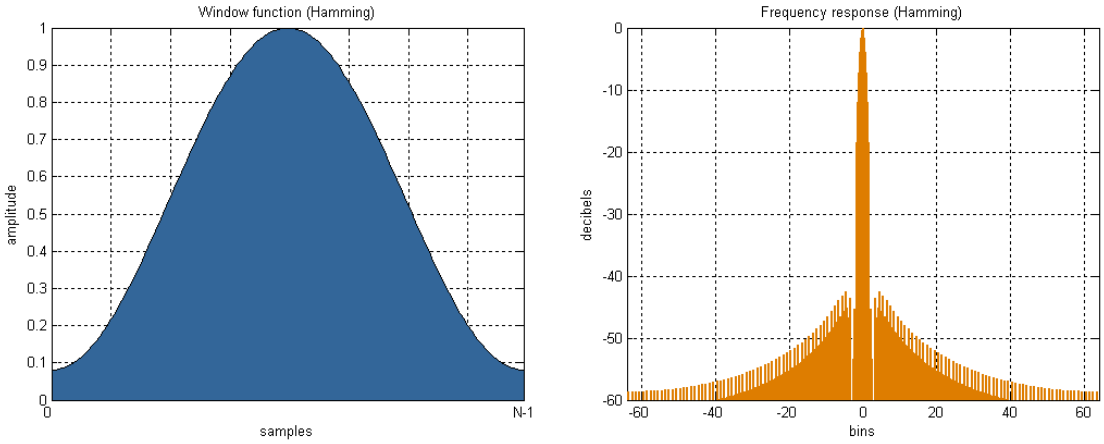

Figure 3: Gradient descent along an error function

The hamming window is optimized to yield very low side-lobes. Lower side-lobes allows for less frequency leakage from more present frequencies which allows for a better amplitude resolution in the final windowed filter.

All filters will be designed using MATLAB's FDA filter design tool, which takes in parameters such as filter order, stop band and pass bad frequencies and sampling frequency, and produces an filter coefficients that best matches the desired filter.

3 Methodology

3.1 Procedure followed

The ADC and DAC of the STM32F4 Discovery were initially set up [6] and tested to ensure they were working correctly.

A sampling frequency of 44kHz was specified for the ADC, which is more than twice the maximum frequency of interest, i.e. 20kHz, to avoid aliasing.

A discrete time domain convolution algorithm was implemented and tested using the given h[n] filter.

Using MATLAB, the theoretical response of the given h[n] filter was found. The filter was then tested using varying sinusoidal inputs and its response compared to the simulated response.

The FDAtool in MATLAB was then used to synthesize the FIR filter coefficients for six different filters, a lowpass and highpass for each of the following design techniques:

1. Equiripple design
2. Least Squares design
3. Hamming Window design

The simulated and actual responses of the filters were then compared..

3.2 Equations used

Some fundamental equations used throughout the project are stated here.

The discrete time convolution equation for a filter h with the response y to some input x is given by:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{1}$$

If h is causal then $h[n] = 0 \; \forall \; n < 0$[6]. Additionally, if it is an FIR filter, with length q+1, then $h[n] = 0 \; \forall \; n > q$, then the convolution equation can be simplified to:

$$y[n] = \sum_{k=0}^{q} h[k]x[n-k] = h[0]x[n] + h[1]x[n-1] + ... + h[q]x[n-q]$$

Thus, the values of h from 0 to q completely specify the filter and q is thus the order of the filter.

The discrete time Fourier transform for a discrete signal $x[n] = 0, 1, \ldots, N-1$ where $x[n] = 0 \; \forall \; n < 0$, is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi \frac{k}{N})n} \tag{2}$$

This formula is also called the discrete Fourier transform (or DFT for short).

4 Test Filter

An FIR filter was given, specified by:

$$h_v = [0.0387 \; -0.0874 \; -0.0463 \; 0.0288 \; 0.0202 \; -0.0591 \; -0.0248 \; 0.1023 \; 0.0248$$
$$-0.3171 \; 0.4749 \; -0.3171 \; 0.0248 \; 0.1023 \; -0.0248 \; -0.0591 \; 0.0202 \; 0.0288 \; -0.0463 \; -0.0874 \; 0.0387]$$

In figure 4, the magnitude and phase of the filter was found for different frequencies. This was achieved using a discrete time Fourier transform (see equation (2)).

As one can see, this is a highpass filter with a linear phase response in the passband. One can also see the repeating side lobes, which are characteristic of digital filters.

4.1 Filter Testing

Sinusoidal inputs were filtered using the test filter. Figure 5 shows the response of the filter to three frequencies (11.5, 12.8 and 13.6 kHz). The FFT of the first two images doesn't convey much information as the resolution was very low. However, the resolution was increased for the final response so that the fourier transform could be observed properly. Lower and high frequencies than those shown were also tested but there was very minimal response to frequencies of 10.5 kHz and less. As the frequency of the inputs was swept between 11.5 to 20 kHz, the output peak-to-peak voltage increased and decreased periodically. This is due to the slight oscillation in the pass band of the filter (which can be seen in the simulated response).

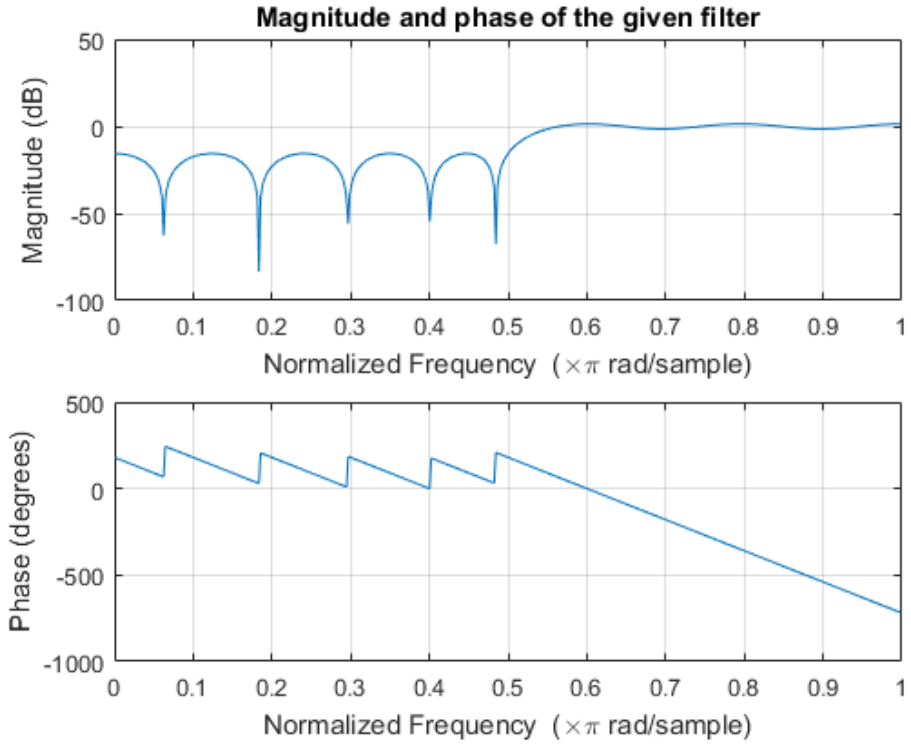

Figure 4: Magnitude and phase of the given filter

4.2 Filter analysis

From the above responses it is evident that the filter acted as expected. The frequencies in the stop band were well attenuated and in contrast, the attenuation of the frequencies in the pass band was minimal.

5 Comparison Of Filter Design Methods

Using the MATLAB FDAtool [2], three design methods, namely Equiripple, Least-squares and a Hamming Window, were employed to design a lowpass, bandpass and highpass filter.

Table 1 shows the stop and pass band specifications for the filters. These bands were chosen based on the defined audio ranges in music

Table 1: Frequency specifications for the stop and pass bands of the filters

Filter Type	f_{pass} [Hz]	f_{stop} [Hz]	f_{crit} [Hz]
Lowpass	400	600	500
Highpass	5000	4000	4000

These frequencies were selected based on the audio spectrum of music [8], summarized in table 2. The lowpass filter specifications should pass frequencies in sub-bass to low midrange regions. This range of frequencies is generally viewed as the bass sounding range. The highpass filter specifications should pass frequencies in presence to the brilliance regions. The resulting signals should sound higher pitched, contain more treble. The treble sound is the opposite to the bass sound.

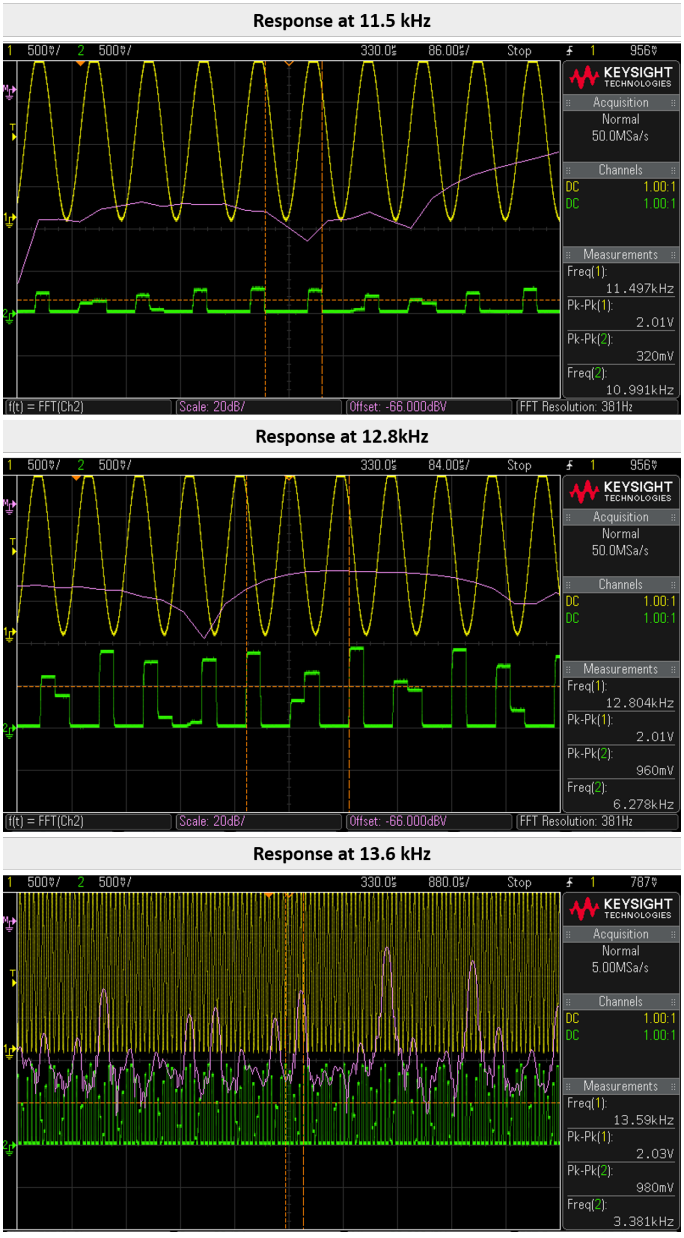

Figure 5: Actual responses of the test filter at various frequencies, including the FFT spectrum of the output signal

Table 2: Summary of defined audio ranges for music

Frequency Range	Frequency Values
Sub-bass	20 to 60 Hz
Bass	60 to 250 Hz
Low midrange	250 to 500 Hz
Midrange	500 Hz to 2 kHz
Upper midrange	2 to 4 kHz
Presence	4 to 6 kHz
Brilliance	6 to 20 kHz

5.1 Simulated Frequency Response of the Filters

After designing the lowpass and highpass filters using the aforementioned deign methods in the MATLAB FDAtool, the magnitude and phase response of each filter was found, as shown in figures 6 and 7. In the pass band region for each filter, the phase response is linear, which means the filters should not distorted the input signal too severely after filter it.

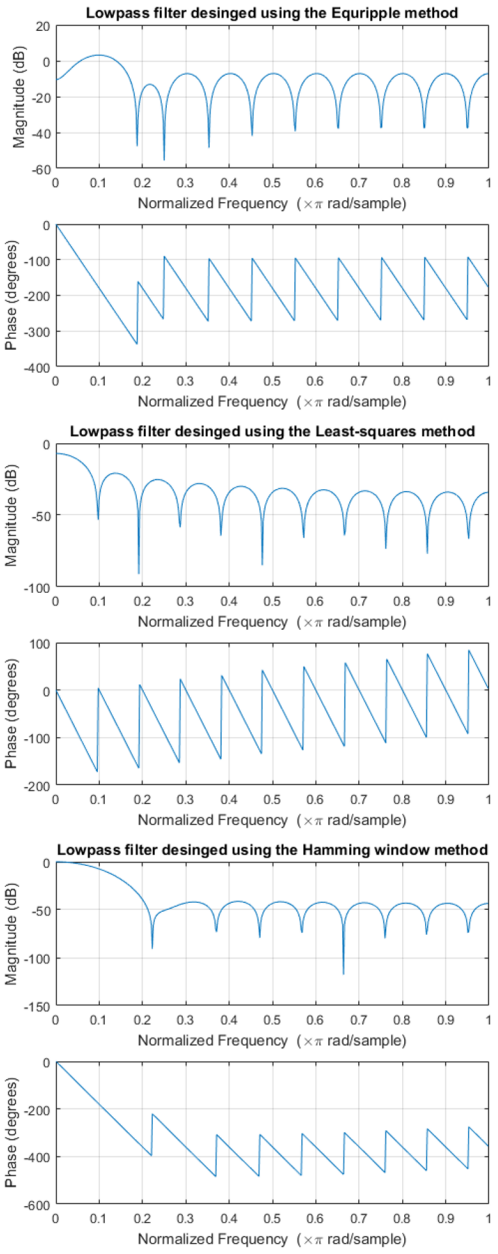

Figure 6: The lowpass filter magnitude and phase plots for each design method

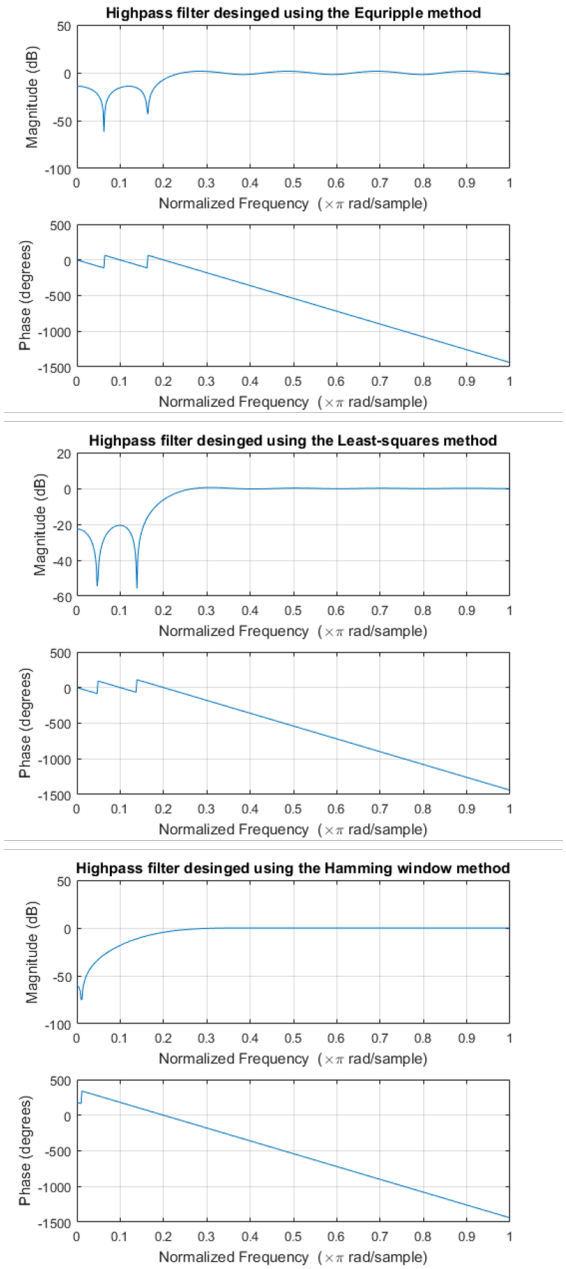

Figure 7: The highpass filter magnitude and phase plots for each design method

5.2 Actual Response of the Filters

The filters were tested with Gaussian noise as opposed to several sinusoids, as the filtering of noise gives a clear depiction of whether high frequencies are attenuated or allowed to pass through. Furthermore, since Gaussian noise contains such a large range of frequencies [3], it acts as a crude impulse response for the filter. Thus, analysing the Fourier transform of the noise responses of each filter allows for a rough comparison of the simulated filter responses.
When analysing the Fourier transforms of each implemented filter, it can be seen that a range of 0-10KHz was used for each, as this makes up the full range of frequency pass and stop bands for each filter. However, as can be seen in the response below, the Equiripple filter needed a range of 0-50KHz as it only began to attenuate frequencies at about 30 kHz.

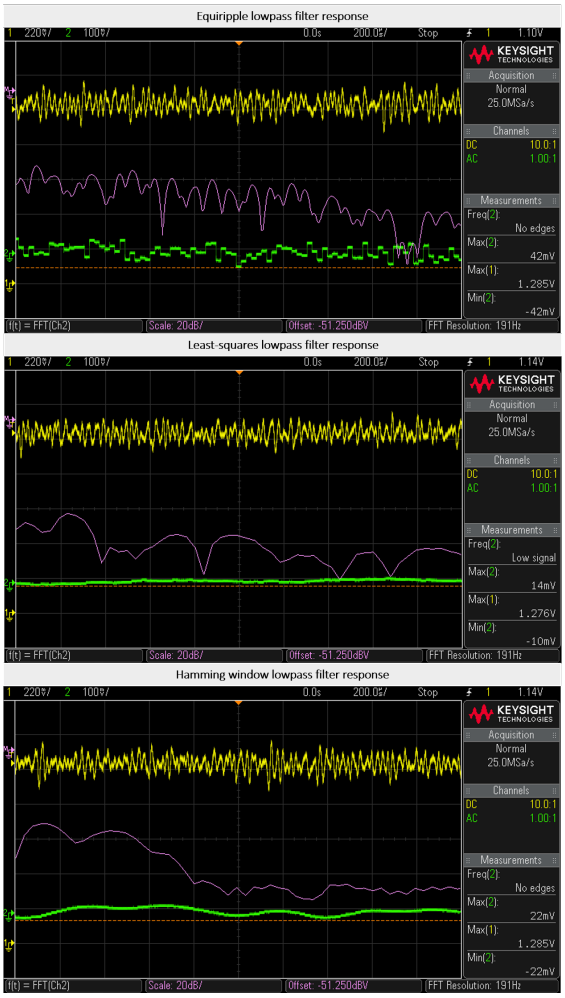

Figure 8: The responses of the implemented low-pass filters to Gaussian noise input signals. ***Yellow*** - input signal, ***Green*** - output signal, ***Purple*** - FFT of the output signal

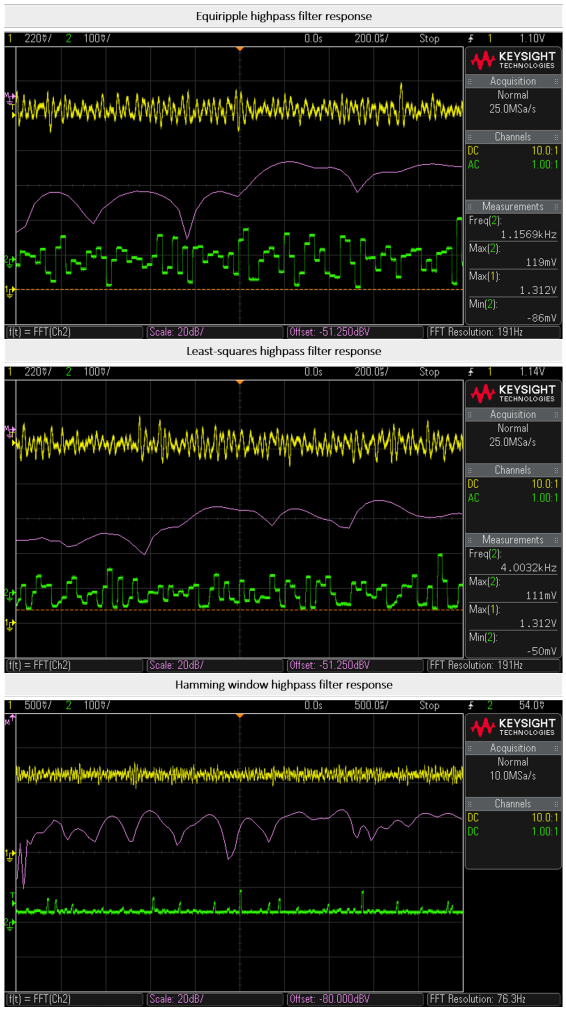

Figure 9: The responses of the implemented highpass filters to Gaussian noise input signals. ***Yellow*** - input signal, ***Green*** - output signal, ***Purple*** - FFT of the output signal

5.3 Discussion And Analysis Of filters

The Equiripple lowpass filter performed poorly and only began attenuating frequencies of 30 kHz and upward, which does not meet the specifications for the lowpass filters. On the other hand, the Least-squares and Hamming Window lowpass filters performed well. The actual simulation responses were similar in nature, with the Hamming window having a larger first lobe than the Least-squares filter. Both filters attenuated the high frequency components of the input noise signal. The Hamming window filter attenuated the higher frequencies the best overall, while passing the lower frequencies well.

All the highpass filters performed well. They attenuated the lower frequency components of the input signal, whilst passing the higher frequency components. The responses of the Equiripple and Hamming window resemble the simulated responses of the filters. One can see the two low frequency side-lobes of the Equiripple filter and then a rise to the pass band, where there is some rippling too and for the Hamming window, the low frequency components started very diminished and rise with a logarithmic trend to the pass band, which matches the trend in the simulated response. However, the Hamming windows contained too much rippling in the stop band and thus performed the worse out of all three highpass filters. The passband of the Least-squares highpass filter was a bit too large, passing too many low frequency components and thus did not meet the specification. The Equiripple highpass filter performed the best overall, attenuating enough of the low frequency components of the input signal.

6 Conclusion

The DAC and ADC on the STM32F4 were set up correctly, and were able to operate at 44KHz. Thereafter, the given h[n] filter was implemented on the system, and was found to be a high pass filter. The filter worked fairly well, attenuating most low frequencies.

Three different design techniques, namely equiripple, least squares and hamming window, were used on MATLAB's FDA tool to design highpass and lowpass filters. Thereafter the filters were implemented on the system, tested and analyzed. It was found that the equiripple lowpass filter did not perform very well, only attenuating frequencies after 30KHz, extremely far out of the specification zone. The system was checked several times to ensure that the poor performance was not due to an incorrect implementation. The behaviour of the filter was seen by the high frequencies that it allowed to pass through.

All other filters were seen to function correctly, the lowpass filter attenuating the high frequencies, and passing through low frequencies, and the highpass filters attenuating low frequencies and allowing the high frequencies to pass through. The crude fourier transforms of the filters also all matched with their simulated counterparts, demonstrating that they were correctly implemented and worked as expected.

References

[1] Documentation - FIRLS, https://www.mathworks.com/help/signal/ref/firls.html.

[2] Filter Design and Analysis using FDATool of MATLAB , https://users.encs.concordia.ca/~blynch/elec442lab5.pdf.

[3] GAUSSIAN NOISE or GAUSSIAN PROBABILITY DISTRIBUTION, http://www.sfu.ca/sonic-studio/handbook/Gaussian_Noise.html.

[4] Hamming Window, https://www.dsprelated.com/freebooks/sasp/Hamming_Window.html.

[5] Introduction to DSP - filtering: design by equiripple method, http://www.bores.com/courses/intro/filters/4_equi.htm.

[6] Online filtering on embedded hardware, http://www.dip.ee.uct.ac.za/~nicolls/lectures/eee4001f/projects/project09.pdf.

[7] Real Time FIR Digital Filters, http://www.eas.uccs.edu/~mwickert/ece5655/lecture_notes/ece5655_chap7.pdf.

[8] Tech Stuff - Frequency Ranges, http://www.zytrax.com/tech/audio/audio.html.