Fcauldade de Ciências e Tecnolofia da Universidade Nova de Lisboa

Relatório do 1º Trabalho de laboratório

Electrónica III

Professor João Pedro Oliveira

Filipe Perestrelo nº 39656

Jaime Teyller nº 35805

Índice

[Introdução 3](#_Toc447661066)

[Tarefa 1 4](#_Toc447661067)

[Tarefa 2 6](#_Toc447661068)

[Tarefa 3 11](#_Toc447661069)

[Conclusões 14](#_Toc447661070)

# Introdução

Este trabalho teve como objectivo caracterizar um ADC (Conversor Analogico-Digital) no seu funcionamento estático e dinâmico. Para tal foram realizadas 3 tarefas onde se usou 3 ferramentas distintas complementares: MatLab, LTSpice e uma Plataforma de microcontrolador (Arduino).

Na primeira tarefa, foi criado um sinal em MatLab, com um certo ruído e depois procedeu-se aos cálculos da relação sinal-ruído (SNR), bem como do número efectivo de bits (ENOB) de um ADC com 8 bits de codificação de sinal.

Na segunda tarefa, utilizou-se um sinal em rampa para se calcular as distorções não líneares: Diferencial (DNL) e Integral (INL). Depois, foi usado um sinal sinusoidal para que, à semelhança da primeira tarefa, se calculasse o SNR e os ENOB.

Na terceira tarefa, foram produzidos dois sinais sinusoidais a duas frequências diferentes, que puderam ser visíveis num osciloscópio. Através da plataforma Arduino e do software associado a essa plataforma foi possível ler valores que foram introduzidos no MatLab para os cálculos do SNR e dos ENOB.

# Tarefa 1

Para a realização da primeira tarefa, foi então criado um sinal sinusoidal de amplitude 1V, centrado em 0V, de frequência 10MHz. Foi introduzido a potência um ruído de jitter através da expressão:

onde A representa a amplitude (1V) do sinal, fin representa a frequência do sinal (10MHz) e σjitter representa o desvio padrão deste ruído que foi definido com o valor de 15x10-12.

Foi adicionado também um ruído térmico, através de um ciclo que colocava valores aleatórios que seriam acrescentados ao sinal (fig.1).

for j = 1: navg

time\_real = time + randn(1, np)\*jitter\_noise;

vin = Ain\*sin(2\*pi\*fin\*time\_real) + randn(1, np)\*vin\_noise;

dout = round(vin./Vlsb);

doutf = fft(dout.\*blackman(max(size(dout)))');

doutfp = (doutf.\*conj(doutf))/np^2;

doutp = doutp + doutfp;

end

Fig.1 – Ciclo que acrescenta ao sinal um ruído térmico

Assim, fez-se a quantização do sinal (fig.2) e a posterior transformada de fourier do sinal ponto a ponto, que resultou no espectro deste sinal na frequência (fig.3).



Fig.2 – Sinal de entrada Vin (a azul) e posterior quantização (a vermelho)



Fig.3 – Transformada de Fourier do sinal com ruído

Após estas operações, já haviam condições para se calcular o SNR e os ENOB. O SNR foi calculado através da seguinte expressão:

sendo que Ps representa a potência do sinal dada por:

Pnq representa a potência do ruído de quantização dada por:

onde Vref é a amplitude do sinal (1V) e N representa o número de bits do ADC (8 bits).

Assim, o SNR teve o valor de 49,5621.

A partir do SNR calculou-se os ENOB pela expressão:

O valor do ENOB foi de 8.0191 bits pelo que teve de ser arredondado para 8, visto tratar-se de um número inteiro.

Calculou-se ainda o valor da relação sinal-ruído e distorção (SNDR) pela expressão:

Verificou-se então que o valor de SNDR não variou muito do valor de SNR (como seria de esperar) pois este valor fixou-se em 50.0352dB.

# Tarefa 2

Para a realização da segunda tarefa, começou-se por abrir o esquemático do circuito que incluia um ADC no LTSpice (fig.4).

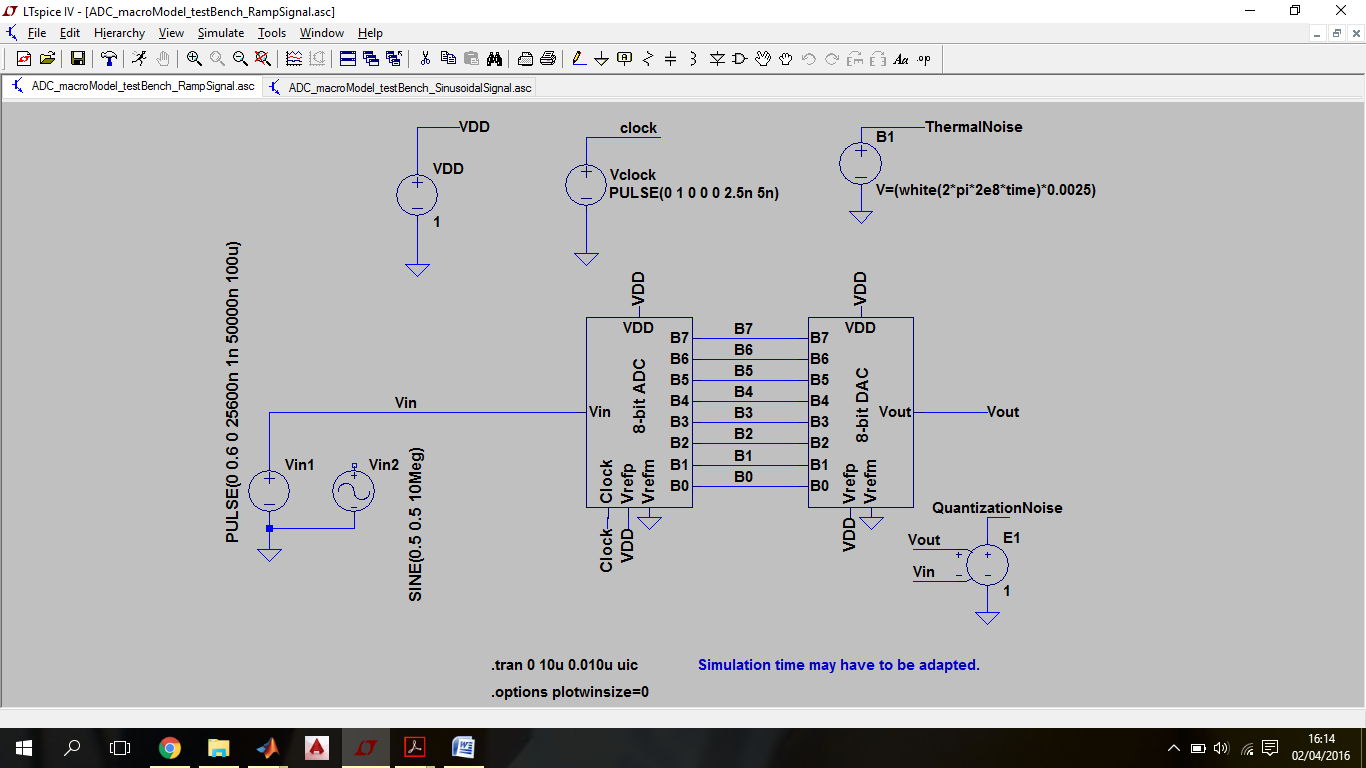


Fig.4 – Esquemático no LTSpice

Foi pedido para simular os dois sinais de entrada V1 e V2 de modo a obter o erro do sinal rampa (fig.5) e do sinal sinusoidal (fig.6).

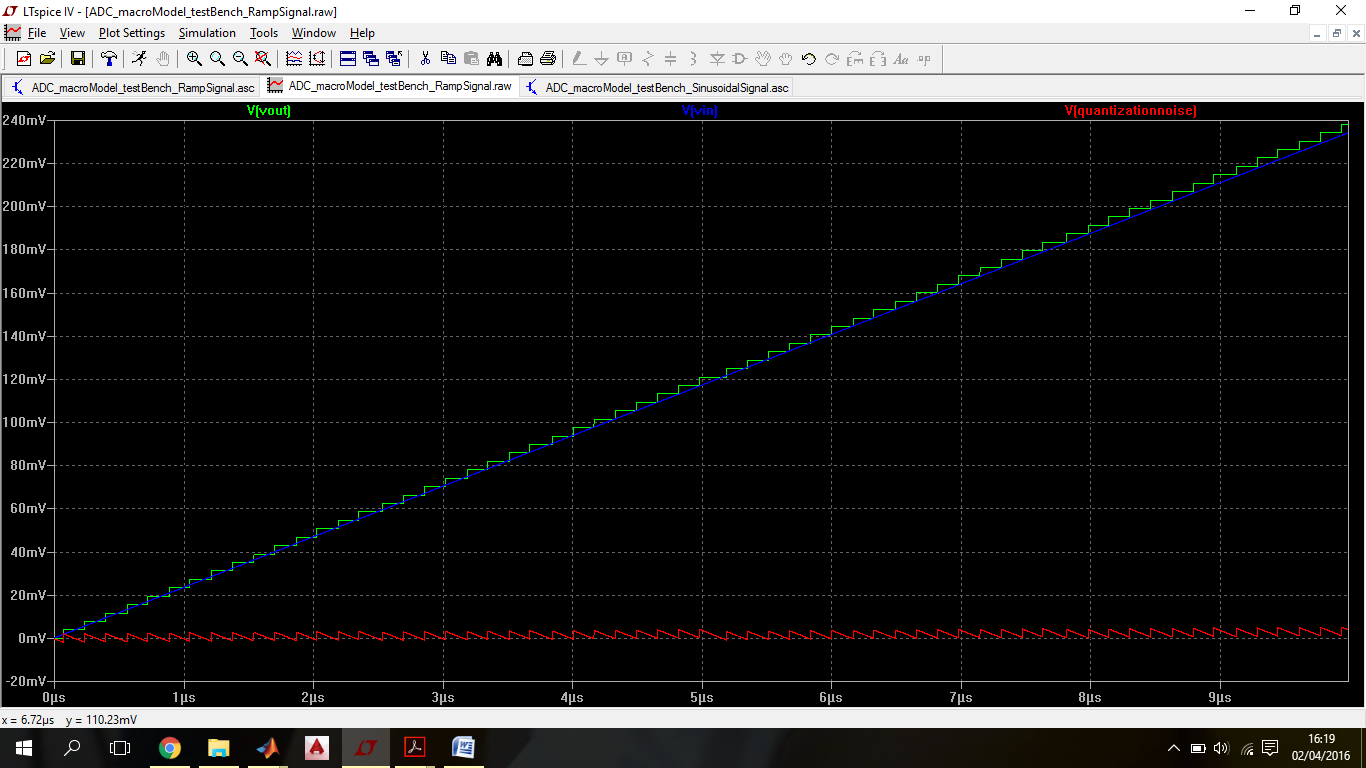


Fig.5 – Simulação do sinal rampa (azul), saída quantizada (verde) do sinal de erro (vermelho)

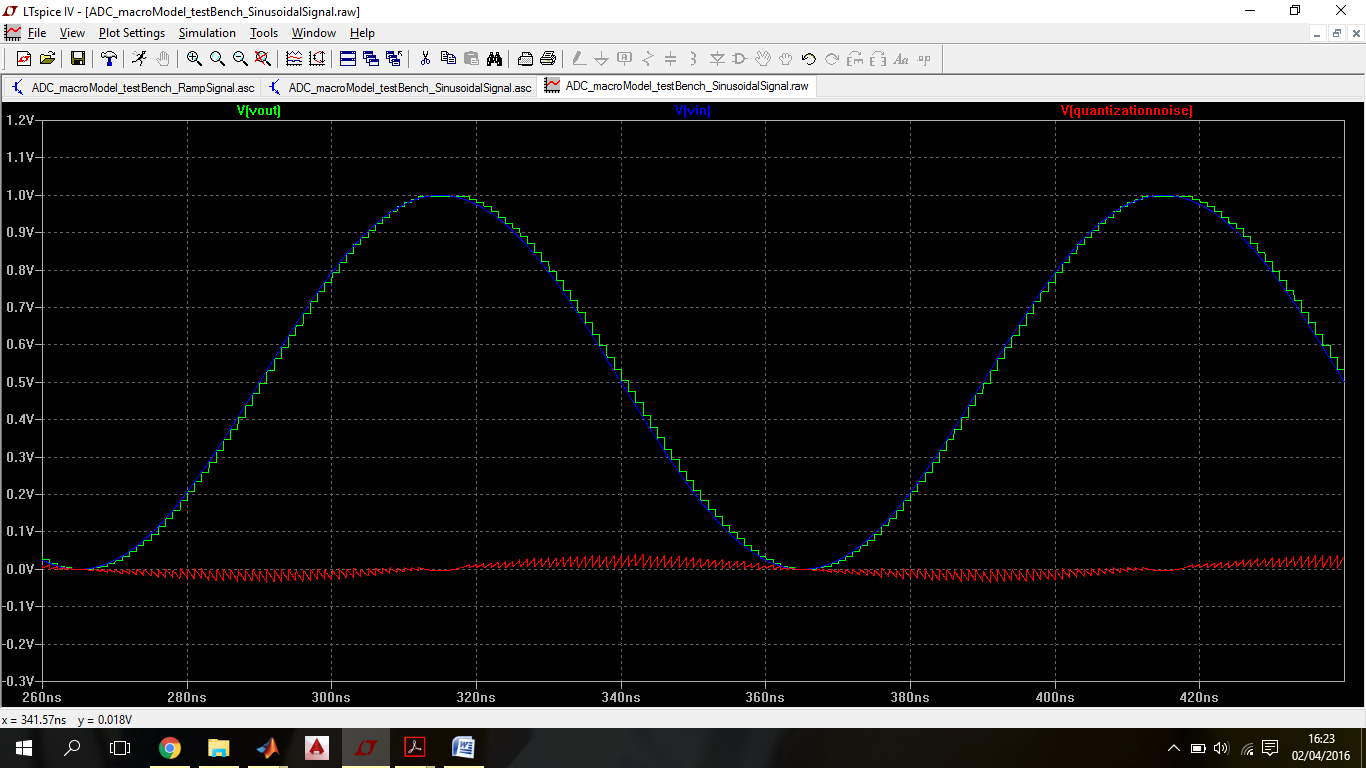


Fig.6 – Simulação do sinal sinusoidal (azul), saída quantizada (verde) do sinal de erro (vermelho)

A partir do sinal rampa foi possível observar a sua INL (fig.7)

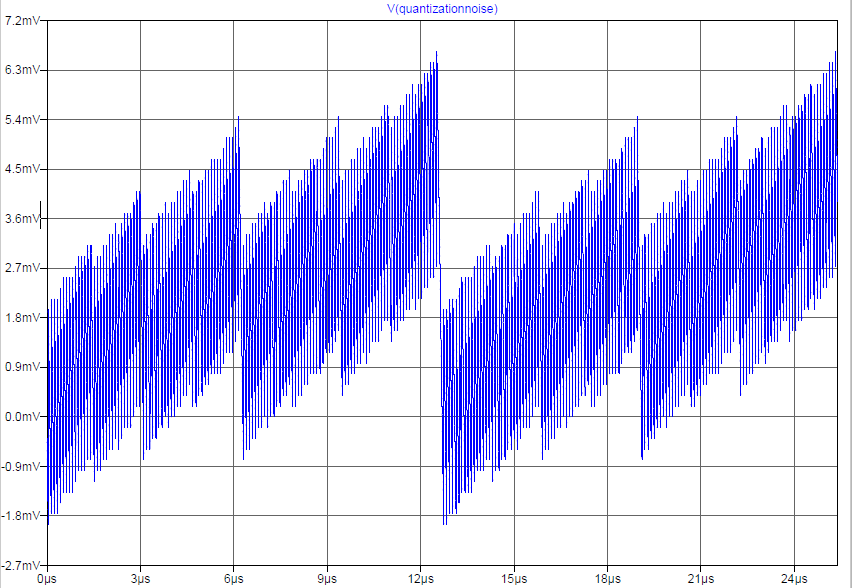


Fig.7 – INL do sinal rampa

É possível reparar que a INL é dada pelo erro de quantização e que não apresenta uma variação muito elevada.

Após estas simulações procedeu-se à utilização da ferramenta FFT (Fast Fourier Transform) para se obter o espectro do sinal sinusoidal na frequência (fig.8).

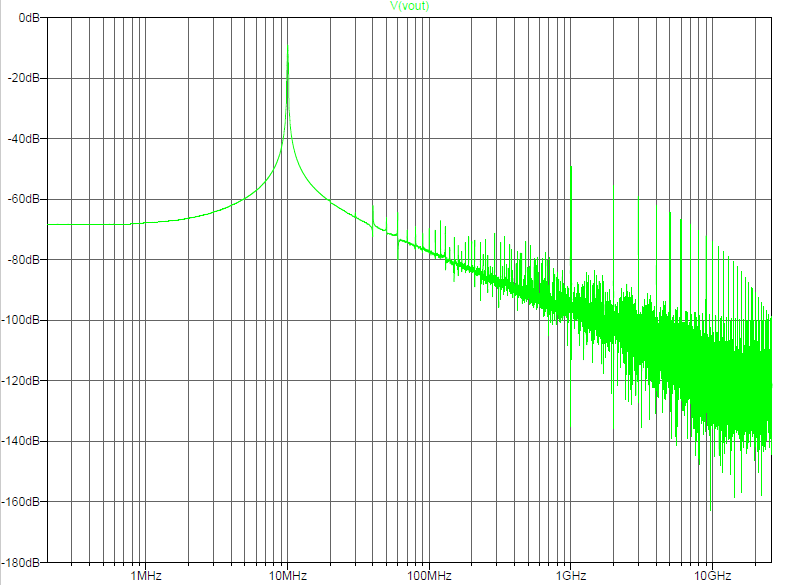


Fig.8 – Espectro do sinal sinusoidal na frequência

Como se pode concluir logo à partida, o primeiro pico do espectro encontra-se na frequência de 10MHz, como seria de esperar visto que a frequência fundamental do sinal sinusoidal era de 10MHz.

Com a implementação do código em MatLab, foi estimado o valor de SNR para 33 e o valor de ENOB para 5 bits, o que representa uma incompatibilidade do ADC. Isto pode dever-se a erros de cálculo, ou até mesmo da não anulação completa da componente DC do sinal, bem como o facto do ADC não ser ideal.

De seguida procedeu-se à implementação do mesmo sinal, mas desta vez com uma componente de simulação de erro térmico ao qual se colocou uma amplitude de 0.085 que teve um certo efeito de distorção no sinal (fig.9) e obteve-se um sinal de saída que incluiu a quantização do erro de térmico presente (fig.10).

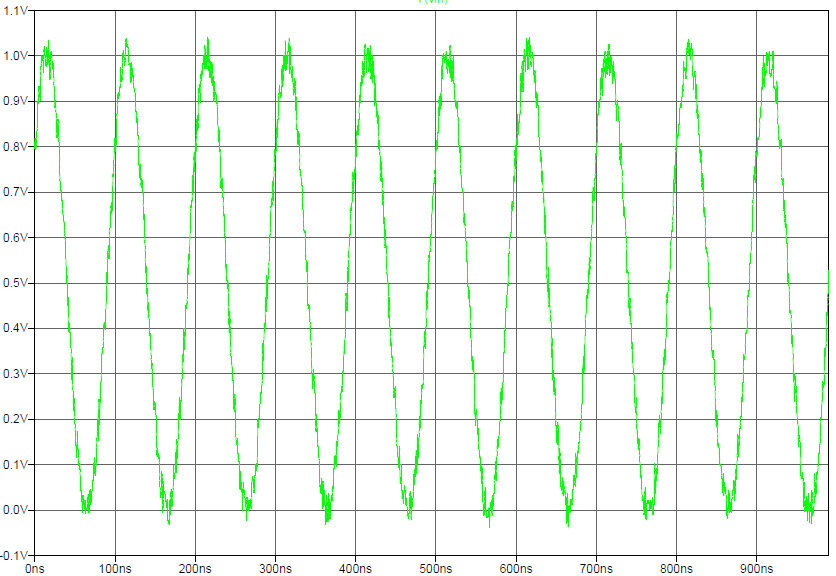


Fig.9 – Sinal de entrada com a simulação de ruído térmico

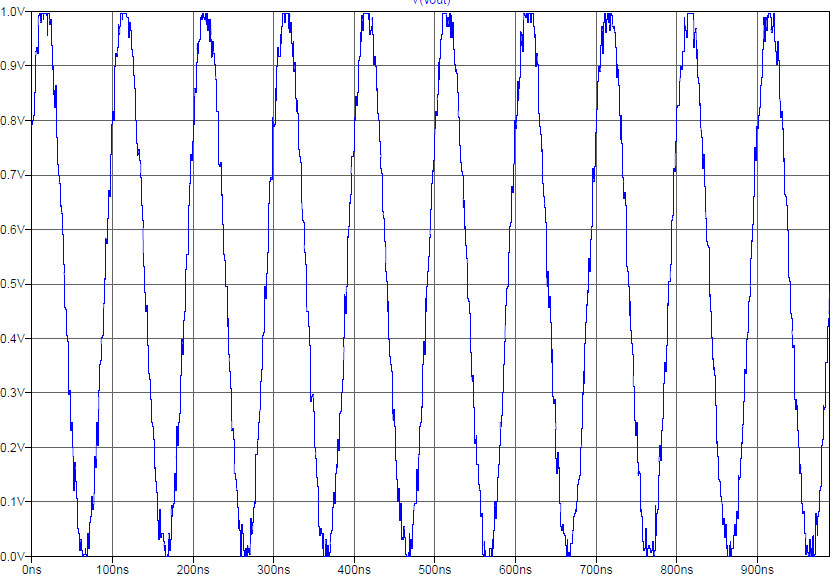


Fig.10 – Sinal de saída do sinal sinusoidal incluindo o erro térmico

Procedeu-se então à verificação da transformada de Fourier do sinal de saída de modo a verificar o seu espectro na frequência (fig.11).

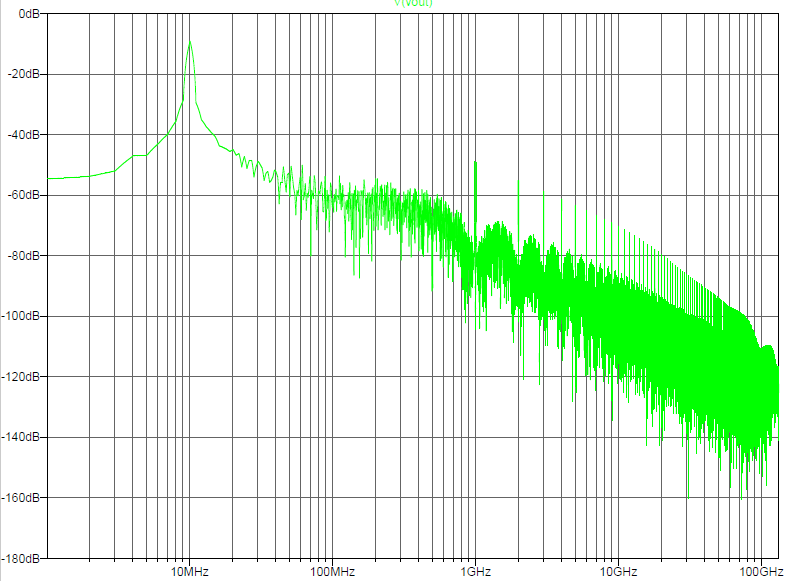


Fig. 11 – Transformada de Fourier do sinal sinusoidal com ruído térmico quantizado

Neste caso, foi possível concluir que a harmónica que tem mais impacto no sinal, continua a ser a da frequência original (10MHz), mas começa a haver uma maior distorção antes da frequência de 1GHz, devido à presença do ruído térmico. Isto leva a concluir que quanto mais ruído foi “injectado” no sinal de entrada, mais degradada será a sua saída.

Procedeu-se de seguida à exportação dos valores deste espectro para o MatLab de forma a poder-se calcular o SNR, bem como o ENOB. Os resultados evidenciaram, e bem, que a relação sinal-ruído desceu, ou seja, o sinal ficou degradado. O SNR teve o valor de 32 e, por conseguinte, o ENOB teve o valor de 5, o que foi de acordo com aquilo que já se estava à espera.

# Tarefa 3

Procedeu-se à implementação do sinal sinusoidal na plataforma Arduíno. Implementou-se primeiro um sinal com uma frequência de 10Hz (fig.12) e depois um outro sinal com uma frequência de 5kHz (fig.13). Os valores foram importados para o MatLab, onde se pôde observar os gráficos.



Fig.12 – Sinal sinusoidal implentado no Arduíno com frequência de 10Hz



Fig.13 – Sinal sinusoidal implentado no Arduíno com frequência de 5kHz

Como pode ser evidente, o sinal sinusoidal de frequência 5kHz apresenta um certo aliasing no início da sua implementação, bem como uma maior distorção do sinal. Isto acontece devido ao facto de 5kHz ser uma frequência muito elevada para a plataforma Arduino.

De seguida, com os valores importados de cada sinal, começou-se por aplicar a transformada de Fourier do sinal de 10Hz (fig.14).



Fig.14 – Transformada de Fourier do sinal de frequência 10Hz

A partir desta transformada, foi possível obter os cálculos que se pretendia. Assim obteve-se o valor de SNR de 31.2, o valor de SFDR (Spurious Free Dynamic Range) de 8.9 e o valor de THD (Total Harmonic Distortion) de 28.8, sendo que o SFDR se trata da distância entre as duas harmónicas mais influentes e o THD é dado pela expressão:

onde PDn representa a potência do ruído e Ps a potência do sinal.

Para fazer estes cálculos todos foi preciso tratar o sinal através de um algoritmo num script do MatLab (fig.15).

Harm1 = max(doutfp)

Harm2 = max(doutfp(doutfp<max(doutfp)))

Pn1=sum(doutfp)-Harm1

SNR=10\*log10(Harm1/Pn1)

SFDR = Harm1-Harm2

for i=1:length(doutfp)

if doutfp(i) == Harm1

doutfp(i) = 0;

end

if doutfp(i) == Harm2

doutfp(i) = 0;

end

end

THD = 10\*log10(sum(doutfp)/Harm1+Harm2)

Fig.15 – Parte do código utilizado em MatLab para o cálculo de SNR, SFDR e THD.

Os valores obtidos indicam que, na prática, os erros reais são muito mais influentes do que na sua idealização inicial. Assim, é possível explicar que os resultados estão afastados do que se realmente se queria, porque existem ruídos que não são possíveis de simular, ou por erros da experiência.

Para o segundo sinal, verifica-se que este não cumpre o critério de Nyquist, logo existe à partida um não cumprimento dos requisitos e, portanto não valeu a pena seguir em diante. Neste caso verifica-se, que se se colocarem os dois sinais no mesmo gráfico (fig.16), observa-se que a frequência de um sinal que cumpre o critério (de 10Hz) tem uma frequência maior do que um sinal que não cumpre (5kHz), mas como tal é impossível, não se pode analisar o sinal como deve ser.



Fig-16 – Os dois sinais sobrepostos: a vermelho o de frequência 5kHz e a azul o de frequência 10Hz

# Conclusões

Após a realização das três tarefas, a maior conclusão que se pode extrair é a de que na prática, as idealizações nem sempre correspondem à realidade, devido à existencia de erros.

Na primeira tarefa, conclui-se que consoante a frequência de amostragem, o sinal pode, ou não, saturar. Conclui-se ainda que o número efectivo de bits está de acordo com o que se quer. O SDNR supera o valor do SNR por pouco devido à existência de ruído.

Na segunda tarefa, conclui-se que o ADC não é ideal, porque o SNR tem um valor mais baixo do que o que se estava à espera, o que faz com que o número de bits decresça em 3 situando-se nos 5 bits. Com o adicionar de ruído térmico, verifica-se que o SNR vai ser mais pequeno e, portanto existirá uma maior degradação do sinal, bem como da sua quantização.

Na terceira tarefa, conclui-se que se um sinal não cumpre o critério de Nyquist, começa a haver aliasing do sinal, como aconteceu com o sinal de 5kHz de frequência. Para o sinal de 10Hz de frequência, conclui-se que o seu SNR é muito baixo comparado com o que se tinha idealizado. Isto deve-se ao facto de haverem erros na quantização.