PLATAFORMA DE TESTES PARA CONVERSORES ANALÓGICO-DIGITAIS

Eduardo Vilela Pinto dos Anjos

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Fernando Antônio Pinto Barúqui

Rio de Janeiro
Março de 2014

PLATAFORMA DE TESTES PARA CONVERSORES ANALÓGICO-DIGITAIS

Eduardo Vilela Pinto dos Anjos

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO.

Examinado por:

Prof. Fernando Antônio Pinto Barúqui, D.Sc.

Prof. Carlos Fernando Teodósio Soares, D.Sc.

Prof. Jose Gabriel Rodriguez Carneiro Gomes, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2014

Vilela Pinto dos Anjos, Eduardo

Plataforma de Testes para Conversores Analógico-Digitais/Eduardo Vilela Pinto dos Anjos. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2014.

XII, 125 p.: il.; 29, 7cm.

Orientador: Fernando Antônio Pinto Barúqui

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Eletrônica e de Computação, 2014.

Referências Bibliográficas: p. 75 – 75.

1. Conversor Analógico-Digital. 2. Método dos Histogramas. 3. Teste da FFT. I. Antônio Pinto Barúqui, Fernando. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Eletrônica e de Computação. III. Título.

Agradecimentos

Primeiramente, gostaria de agradecer a minha mãe, Maria Verônica Vilela Pinto, por todo o carinho e suporte que me foi dado, não só durante a graduação, mas em toda minha vida. Por toda força e amor nos momentos difíceis, e toda ajuda e cobrança, quando necessário, para o alcance dos objetivos traçados. Nada teria sido possível sem ela.

Ao meu pai, Jackson Gonçalves dos Anjos, por sua amizade e por seus conselhos e palavras de sabedoria, que levarei por toda minha vida.

Ao meu irmão Henrique Vilela Pinto dos Anjos, por sua amizade e brincadeiras cotidianas.

To Kanako Matsuyama, for always being there for me, even if she is on the other side of the globe. Thank you for making the last couple years of my life the best ones I ever had. Your support makes me stronger each and every day.

Ao meu professor e orientador Fernando Antônio Pinto Barúqui, por estar sempre a disposição para tirar minhas dúvidas, e por todo o apoio e paciência durante o desenvolvimento deste trabalho. Além disso, agradeço pela oportunidade que me foi dada, e por tudo o que me ensinou.

Aos professores do Departamento de Engenharia Eletrônica e de Computação, por fornecerem um ensino de excelência e estarem sempre disponíveis para ajudar e sanar dúvidas de todos os alunos. Em especial aos professores Carlos Fernando Teodósio, José Gabriel Gomes, Antônio Petraglia, Jomar Gozzi, Eduardo Nunes e ao professor e coordenador Casé.

To the Santa Clara University, for the amazing year I spent there as an exchange student. It was the best experience of my life. Specially for Andrea Muillemberg for all the support during my stay, and to Dr. Sally Wood and Dr. Shoba Krishnan for the amazing classes and advices.

À todos os amigos e companheiros que pude conhecer ao longo de toda a faculdade, que tornaram essa caminhada mais leve e divertida. Muito obrigado pela companhia em todas as matérias e trabalhos feitos na graduação. Em especial aos amigos Ignácio Ricart e Paulo Victor Vidal, pelo time que formamos em praticamente todos os trabalhos em grupo.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

Plataforma de Testes para Conversores Analógico-Digitais

Eduardo Vilela Pinto dos Anjos

Março/2014

Orientador: Fernando Antônio Pinto Barúqui

Curso: Engenharia Eletrônica e de Computação

Conversores Analógico-Digitais são hoje os circuitos de sinais mistos mais produzidos e estudados no mundo. Com isso, é de suma importância validar o funcionamento destes conversores, utilizando diversos testes de bancada, que normalmente são bastante exaustivos. Isto nos leva a procurar uma plataforma que execute estes testes de forma simples e direta. Este trabalho tem como objetivo apresentar uma plataforma de testes automática para conversores analógico-digitais, que seja fácil de usar e gere resultados confiáveis. Os testes escolhidos para serem executados foram o método dos histogramas e o teste da FFT. A plataforma possui um hardware feito com microcontrolador para executar a aquisição de amostras e um software implementado com MATLAB, utilizando-se o ambiente de desenvolvimento de interface gráfica com o usuário (GUIDE). O trabalho é concluído apresentando resultados de testes em conversores já testados anteriormente, e comparando os resultados fornecidos com os obtidos pela plataforma, podemos comprovar o funcionamento da mesma.

Palavras-chave: Conversor Analógico-Digital, ADC, Teste de ADC, Método dos Histogramas, Teste da FFT.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

TEST PLATAFORM FOR ANALOG-TO-DIGITAL CONVERTERS

Eduardo Vilela Pinto dos Anjos

March/2014

Advisor: Fernando Antônio Pinto Barúqui

Course: Electronic Engineering

Nowadays, Analog-to-Digital Converters are the most produced and studied circuits in the world. Thus, it's very important to validate the operation of these converters, using benchmark tests that usually are very exhausting to perform. This issue leads to a search for a platform that executes these tests on a straightforward manner. In this work, we present an automatic testing platform for analog-to-digital converters, which is easy to use and generate reliable results. The tests chosen to be performed were the Histogram Method and the FFT Testing. The platform has a hardware made with a microcontroller to make the sample acquisition and software implemented with MATLAB, using the graphic user interface development environment (GUIDE). This work is concluded presenting results from tests in converters that had been tested before, and by comparing the provided results with the ones made with the platform, we can proof that the platform works.

Keywords: Analog-to-Digital Converters, ADC, ADC Testing, Histogram Method, FFT Testing.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Tema

O tema do trabalho é a caracterização dos parâmetros de conversores analógico-digitais. Pretendemos medir diversas figuras de mérito, tanto estáticas quanto dinâmicas, como Offset, Não-Linearidade Integral (INL) e Diferencial (DNL), Erro de Ganho, Relação Sinal-Ruido (SNR) e Sinal-Ruído-Distorção (SINAD), Número Efetivo de Bits (ENOB), Distorção Harmônica Total (THD) e Faixa Dinâmica Livre de Espúrios (SFDR).

Este tema se enquadra na área de eletrônica, mais especificamente, na parte de circuitos eletrônicos e instrumentação.

1.2 Delimitação

O objeto do estudo são os conversores analógico-digitais de baixa velocidade, os chamados de taxa de Nyquist, pois além de seres os conversores que iremos testar, conversores muito velozes necessitam de outras estratégias de teste, assim como equipamentos mais robustos. Portanto, outros tipos de conversores estão fora do escopo deste projeto.

1.3 Justificativa

O uso de conversores de dados, analógico-digital ou digital-analógico, vem crescendo cada vez mais na indústria. Isso tem incentivado o aumento das pesquisas na área de projetos de conversores analógico-digitais. A validação do funcionamento dos conversores decorre de testes realizados em bancada para determinação das figuras de mérito amplamente empregadas pela indústria. Estes testes são exaustivos e demandam um longo tempo, o que justifica a utilização de um sistema automatizado para sua realização.

Os métodos de testes geralmente utilizados também envolvem caríssimos equipamentos e montagens complicadas e confusas. Portanto, é de interesse comum uma solução que simplifique este processo.

Na nossa universidade, o Programa de Engenharia Elétrica, mais especificamente os laboratórios de microeletrônica, estão desenvolvendo projetos de pesquisa em conversores analógico-digitais mas não possuem nenhum equipamento automatizado para fazer a validação. Isto justifica a elaboração deste sistema automático.

Além disso, a elaboração desta plataforma é um projeto multi-disciplinar, que envolve conhecimentos de programação, circuitos eletrônicos e microeletrônicos, além de instrumentação e processamento de sinais. É uma excelente forma de empregar todo o conhecimento obtido ao decorrer do curso de Engenharia Eletrônica e de Computação, sendo uma excelente opção para Projeto Final de Graduação.

1.4 Objetivos

O objetivo geral deste projeto é desenvolver uma plataforma de testes para conversores analógico-digitais na faixa de frequência de Nyquist. Pretendemos criar um sistema que possua circuitos eletrônicos que controlam o conversor e executam a aquisição dos dados, assim como uma interface de software que permita uma interação com o usuário, deixando a plataforma mais fácil de se operar. Os resultados obtidos serão apresentados em forma de gráficos e tabelas numéricas, que serão

armazenadas para análise posterior.

1.5 Metodologia

Este trabalho será feito em duas partes, uma de *hardware* e outra de *software*.

A parte de *hardware* será feita com um controlador PIC. Este controlador será responsável por gerar os sinais que controlam o conversor, assim como irá armazenar os dados em sua memória e fará a interação com o computador. Esta interação será feita através do barramento USB, que é muito simples de se usar e está presente na maioria dos controladores PIC.

Os circuitos que geram sinais de entrada serão equipamentos de alta qualidade, pois a criação de tais circuitos poderiam interferir na qualidade das medições, o que vai contra os padrões do IEEE. Estes equipamentos serão controlados por barramentos certificados pelo IEEE.

A parte de *software* será feita em MATLAB, utilizando o *toolbox* GUIDE para criar a interface gráfica. O programa contará com uma interface gráfica que permitirá um setup inicial para que o aparato de testes tome as decisões necessárias para fazer a medição. Ele também irá obter os dados e gerar gráficos importantes, assim como figuras de mérito relevantes, como o número efetivo de bits (ENOB).

A plataforma irá executar dois testes: O primeiro irá medir os coeficientes estáticos utilizando o método dos histogramas, que é um método estatístico para obter os níveis de transição do conversor. Estes valores serão plotados, gerando a curva característica do conversor, assim como curvas de linearidade e diversos parâmetros, como o *offset*. O segundo teste irá medir os coeficientes dinâmicos, usando um gerador de sinais externo como entrada. A saída será armazenada por uma quantidade de tempo suficiente para se realizar a FFT do sinal, que será plotada, e retornar basicamente todos os parâmetros dinâmicos desejados.

1.6 Descrição

No Capítulo 2 serão apresentados os conceitos básicos de conversores analógico-digitais.

No Capítulo 3 serão apresentadas as figuras de mérito relevantes e como testá-las.

O Capítulo 4 apresenta o *hardware* utilizado na plataforma de testes.

O Capítulo 5 apresenta a interface gráfica da plataforma de testes.

Os testes e resultados serão apresentados no Capítulo 6.

Na conclusão, iremos comparar os resultados obtidos e esperados, e assim validar a funcionalidade da plataforma.

Capítulo 2

Conversores Analógico-Digitais

A conversão de dados é um processo comumente utilizado na indústria, pois permite uma troca de informação entre o mundo analógico e o digital. Ao mesmo tempo em que é mais fácil de se processar dados e sinais digitalmente, as grandezas do mundo real são analógicas, tais como temperatura, deslocamento, pressão, etc. Isto faz com que esta ponte entre os dois meios seja muito procurada.

Neste capítulo serão analisados os conceitos básicos de conversores de dados, bem como serão caracterizados conversores ideais e apontados os diversos problemas da conversão de dados na prática. Serão apresentados os sistemas de conversão de dados, e iremos detalhar cada bloco deste complexo sistema.

2.1 Conversores Ideais

Conversores analógico-digitais (também chamados de conversores A/D ou simplesmente ADC) ideais funcionam como uma caixa preta, onde recebem como informação uma voltagem de entrada V_{in} e uma referência V_{ref}, e geram como saída uma palavra binária B_{out}, como pode ser visto na Figura 2.1.

O conversor funciona de forma que a razão entre V_{in} e V_{ref} define B_{out}, onde V_{in} é sempre menor que V_{ref}. Isto significa que a palavra binária B_{out} representa um número menor que 1. Sendo N o número de bits do conversor A/D, podemos modelar B_{out} pela Eq. 2.1, como em [1]:

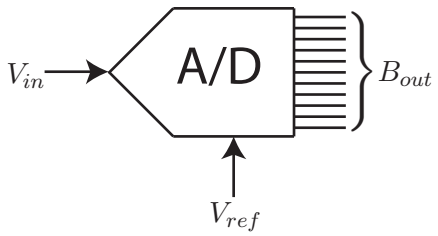


Figura 2.1: Conversor Analógico-Digital Ideal.

$$B_{out} = b_1 2^{-1} + b_2 2^{-2} + \ldots + b_N 2^{-N} \tag{2.1}$$

Os valores $b_1, b_2, \ldots, b_N$ são os bits do código B_{out}. Como estes bits representam um valor binário menor que 1, temos que b_1 é o bit mais significativo (MSB), e b_N é o bit menos significativo (LSB) de B_{out}.

Como a razão $\frac{V_{in}}{V_{ref}}$ contínua define a saída B_{out} discreta, podemos dizer que $\frac{V_{in}}{V_{ref}}$ está entre duas palavras binárias adjacentes, uma representando o valor n e outra o valor $n+1$. Ou seja, temos:

$$B_n < \frac{V_{in}}{V_{ref}} < B_{n+1} \tag{2.2}$$

Num ADC ideal, B_{out} será o valor binário que mais se aproxima da razão. Podemos notar que, como B_n e B_{n+1} são códigos adjacentes, eles diferem por apenas um LSB, cujo valor é:

$$LSB = 2^{-N} = \frac{1}{2^N} \tag{2.3}$$

Aplicando a definição da equação 2.3, podemos reescrever a equação 2.2 como:

$$B_{out} - \frac{1}{2}LSB < \frac{V_{in}}{V_{ref}} < B_{out} + \frac{1}{2}LSB \tag{2.4}$$

Manipulando algebricamente, temos:

$$\frac{V_{in}}{V_{ref}} - \frac{1}{2}LSB < B_{out} < \frac{V_{in}}{V_{ref}} + \frac{1}{2}LSB \tag{2.5}$$

Multiplicando todos os termos por V_{ref}, obtemos:

$$V_{in} - \frac{1}{2}LSB \times V_{ref} < B_{out} \times V_{ref} < V_{in} + \frac{1}{2}LSB \times V_{ref} \tag{2.6}$$

Pode-se definir uma grandeza muito útil, V_{LSB}, que é a menor mudança que deve ser feita na entrada de forma a alterar o código da saída. Como a menor mudança na saída é um LSB, então V_{LSB} é o valor de entrada que muda a saída em 1 LSB [1]. Como o maior valor de entrada é V_{ref}, temos:

$$\frac{V_{LSB}}{V_{ref}} = LSB \rightarrow V_{LSB} = V_{ref} \times LSB = \frac{V_{ref}}{2^N} \tag{2.7}$$

Aplicando o conceito do V_{LSB} à equação 2.6, resulta em:

$$V_{in} - \frac{1}{2}V_{LSB} < B_{out} \times V_{ref} < V_{in} + \frac{1}{2}V_{LSB} \tag{2.8}$$

Reescrevendo a equação 2.8 de uma forma mais compacta:

$$B_{out} \times V_{ref} = V_{in} + q_e, \quad onde\, q_e \leq \pm\frac{1}{2}V_{LSB} \tag{2.9}$$

Com o resultado da equação 2.9, pode-se observar que como o ADC possui saída digital, discreta, para entradas analógicas, contínuas, diversos valores na entrada do conversor podem exibir o mesmo código na saída. Isto significa que mesmo a conversão analógica-digital ideal introduz um erro, chamado de erro de quantização, que será apresentado em breve.

A partir da equação 2.9 é possível traçar um gráfico entre V_{in} e B_{out}, visto na Figura 2.2. Este gráfico é conhecido como a Curva Característica do conversor A/D. Por simplicidade, na Figura 2.2 é mostrada a curva ideal para um conversor de apenas 3 bits.

Algumas propriedades desta curva são importantes de destacar. O código da saída é zero para valores de entrada entre 0 e $\frac{1}{2}V_{LSB}$. Quando a entrada é de $\frac{1}{2}V_{LSB}$, o valor de saída fica entre 0 ou 1, ou seja, há uma indecisão de qual valor

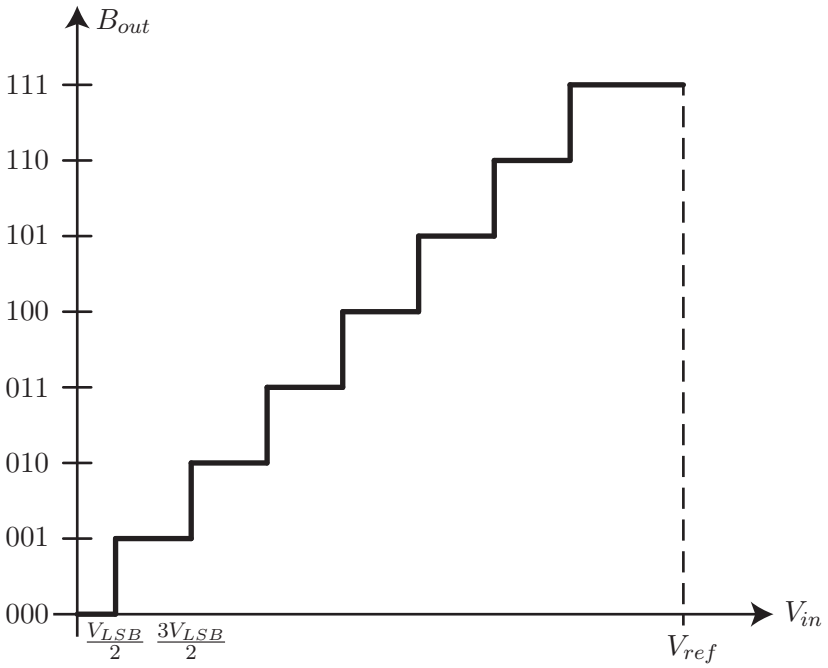


Figura 2.2: Curva característica de um ADC de 3 bits.

assumir. Esse ponto é modelado idealmente de forma que metade das vezes a saída é 0 e na outra metade é 1. Estes pontos de indecisão são chamados de **Níveis de Transição**, pois são onde a transição de um código para o outro acontece. Pode-se observar também que a curva possui um formato de escada, onde cada 'degrau' possui largura de V_{LSB}.

2.2 Sistemas de Conversão Analógico-Digital

O conversor analógico-digital ideal apresentado é uma abstração de um sistema que é muito mais complexo. Ele é na verdade um grande sistema, que pode ser decomposto em vários blocos com diferentes funções e peculiaridades.

A conversão analógica-digital pode ser decomposta em basicamente quatro blocos: Filtro *anti-aliasing*, Amostrador, Quantizador e Codificador [2]. O diagrama de blocos pode ser visto na Figura 2.3.

O filtro *anti-aliasing*, como o próprio nome diz, executa uma filtragem no sinal de entrada que evita o efeito de *aliasing*, sobre o qual será falado em breve. Basicamente, este bloco limita a banda do sinal, para que o processo de amostragem seja

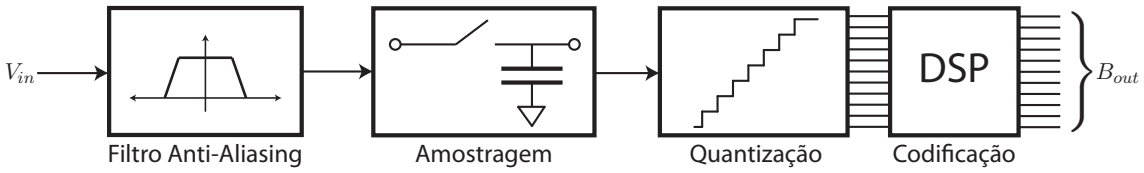


Figura 2.3: Sistema de Conversão Analógico-Digital.

mais seguro.

O bloco de amostragem tem como função obter uma amostra do sinal contínuo, e manter o valor desta amostra constante durante todo o processo de conversão. Isto significa que a escala de tempo do sinal de entrada fica discretizada, adicionando fatores muito interessantes para o sistema. O processo de amostragem será discutido em detalhes na Seção 2.3.

O processo de quantização é análogo ao de amostragem, porém para os valores de amplitude. O quantizador analisa o valor do sinal de entrada e o compara com uma escala interna, semelhante à curva característica do conversor A/D da Figura 2.2. Esta escala é dividida em degraus (não necessariamente de mesmo tamanho), onde cada degrau está relacionado a uma combinação binária. O quantizador então é o responsável por mapear o sinal de entrada nos códigos binários. É neste bloco que vai se introduzir o erro de quantização, que será analisado na Seção 2.4

Apesar do quantizador já fornecer uma saída binária, muitas vezes é desejado que esta saída esteja formatada em uma codificação específica - como, por exemplo, Código Gray - diferente da entregue pelo bloco de quantização. Esta adequação é feita por um Codificador que executa operações matemáticas na palavra digital, modificando-a como desejado.

2.3 Amostragem

A amostragem é o processo que transforma sinais contínuos em sinais discretos no tempo. Idealmente, o Amostrador pode ser modelado como na Figura 2.4.

Nesta modelagem, baseada em [3, pp. 181-191], o sinal de entrada é multiplicado

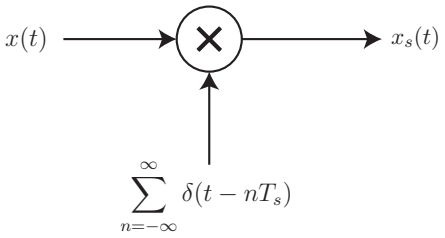


Figura 2.4: Modelagem de um Amostrador Ideal.

por uma sequência de impulsos distanciados por um periodo de tempo fixo, o período de amostragem T_s, gerando o sinal amostrado $x_s(t)$. Assim, tem-se:

$$x_s(t) = x(t) \times \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \rightarrow x[n] \tag{2.10}$$

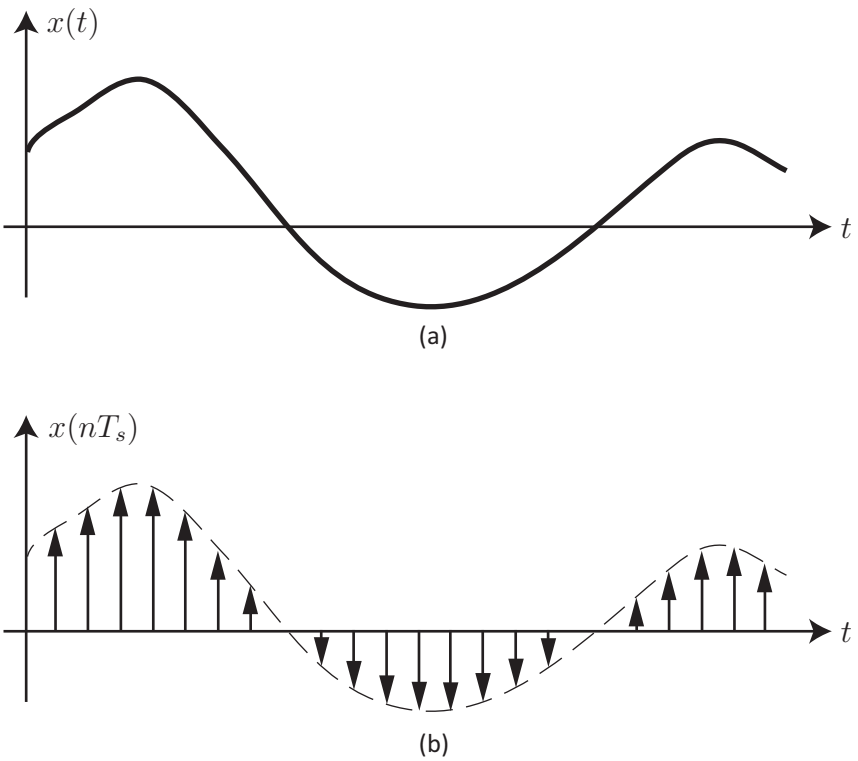


Figura 2.5: (a) Sinal de entrada original, (b) Sinal de entrada multiplicado pelo trem de impulsos.

O efeito desta modelagem pode ser observado na Figura 2.5. Uma análise importante que deve ser feita é o comportamento do sinal amostrado no domínio da frequência, utilizando a transformada de Fourier deste sinal. Para isto, tem-se:

$$X_s(\jmath\Omega) = \mathcal{F}\{x_s(t)\} = \mathcal{F}\left\{x(t) \times \sum_{n=-\infty}^{\infty} \delta(t - nT_s)\right\} \tag{2.11}$$

O produto de dois sinais no domínio do tempo resulta na convolução destes dois sinais no domínio da frequência, corrigido por um fator multiplicativo de $\frac{1}{2\pi}$. Assim, a equação 2.11 pode ser simplificada da seguinte forma:

$$X_s(\jmath\Omega) = \frac{1}{2\pi}\mathcal{F}\{x(t)\} * \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} \delta(t - nT_s)\right\} \tag{2.12}$$

A transformada de Fourier de um trem de impulsos no domínio do tempo é:

$$\mathcal{F}\left\{\sum_{n=-\infty}^{\infty} \delta(t - nT_s)\right\} = \frac{2\pi}{T_s}\sum_{k=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi k}{T_s}\right) \tag{2.13}$$

Utilizando $\Omega = \frac{2\pi}{T}$, pode-se definir a frequência de amostragem como $\Omega_S = \frac{2\pi}{T_s}$. Supondo que a transformada de Fourier do sinal de entrada $x(t)$ seja $X(\jmath\Omega)$, pode-se simplificar a equação 2.12, obtendo:

$$X_s(\jmath\Omega) = \frac{1}{2\pi}X(\jmath\Omega) * \left[\Omega_S \sum_{k=-\infty}^{\infty} \delta\left(\Omega - k\Omega_S\right)\right] \tag{2.14}$$

Resolvendo esta equação, tem-se:

$$\begin{aligned} X_s(\jmath\Omega) &= \frac{\Omega_S}{2\pi}\int_{-\infty}^{\infty} X(\jmath\Omega) \sum_{k=-\infty}^{\infty} \delta\left(\Omega - k\Omega_S - \theta\right) d\theta \\ &= \frac{\Omega_S}{2\pi}\sum_{k=-\infty}^{\infty}\int_{-\infty}^{\infty} X(\jmath\Omega)\delta\left(\Omega - k\Omega_S - \theta\right) d\theta \\ &= \frac{\Omega_S}{2\pi}\sum_{k=-\infty}^{\infty} X\left[\jmath(\Omega - k\Omega_S)\right] \end{aligned} \tag{2.15}$$

O resultado final do desenvolvimento anterior pode ser simplificado como:

$$X_s(\jmath\Omega) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X\left[\jmath(\Omega - k\Omega_S)\right] \tag{2.16}$$

A equação 2.16 é muito importante, pois significa, na prática, que o espectro do sinal amostrado é o espectro do sinal original, repetido a cada Ω_S. Isto implica que

a frequência de amostragem Ω_S deve ser cuidadosamente escolhida, e deve levar em conta a largura de banda do sinal de entrada. A relação entre Ω_S e a banda do sinal de entrada foi introduzida pelo teorema de Nyquist-Shannon, também conhecido como teorema da amostragem, que é um dos teoremas mais importantes da história do processamento de sinais.

O teorema de Nyquist-Shannon estabelece o seguinte: Seja um sinal de entrada $x(t)$ que possui uma banda limitada por uma frequência Ω_N, ou seja, $|X(j\Omega)|= 0, \forall \quad |\Omega| > |\Omega_N|$, como mostrado na Figura 2.6. A frequência na qual este sinal $x(t)$ deve ser amostrado deve ser tal que

$$\Omega_S \geq 2 \times \Omega_N \tag{2.17}$$

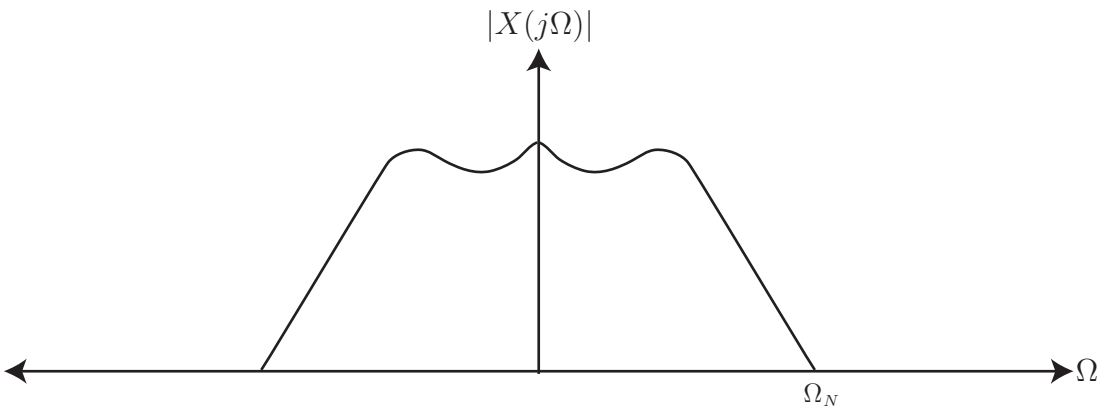


Figura 2.6: Módulo do Espectro de Frequência do Sinal de Entrada.

A frequência Ω_N é conhecida como a *frequência de Nyquist*, e o produto $2\Omega_N$ é chamado de *Taxa de Nyquist*. Com este teorema, podemos observar três tipos de amostragem: Amostragem à Taxa de Nyquist, Subamostragem e Superamostragem. Cada tipo será detalhado a seguir.

2.3.1 Amostragem à Taxa de Nyquist

Este tipo de amostragem acontece quando a frequência de amostragem Ω_S respeita o teorema de Nyquist, ou seja, $\Omega_S > 2\Omega_N$. O processo pode ser visto na Figura 2.7.

Como a frequência de amostragem é maior que a taxa de Nyquist, as réplicas do

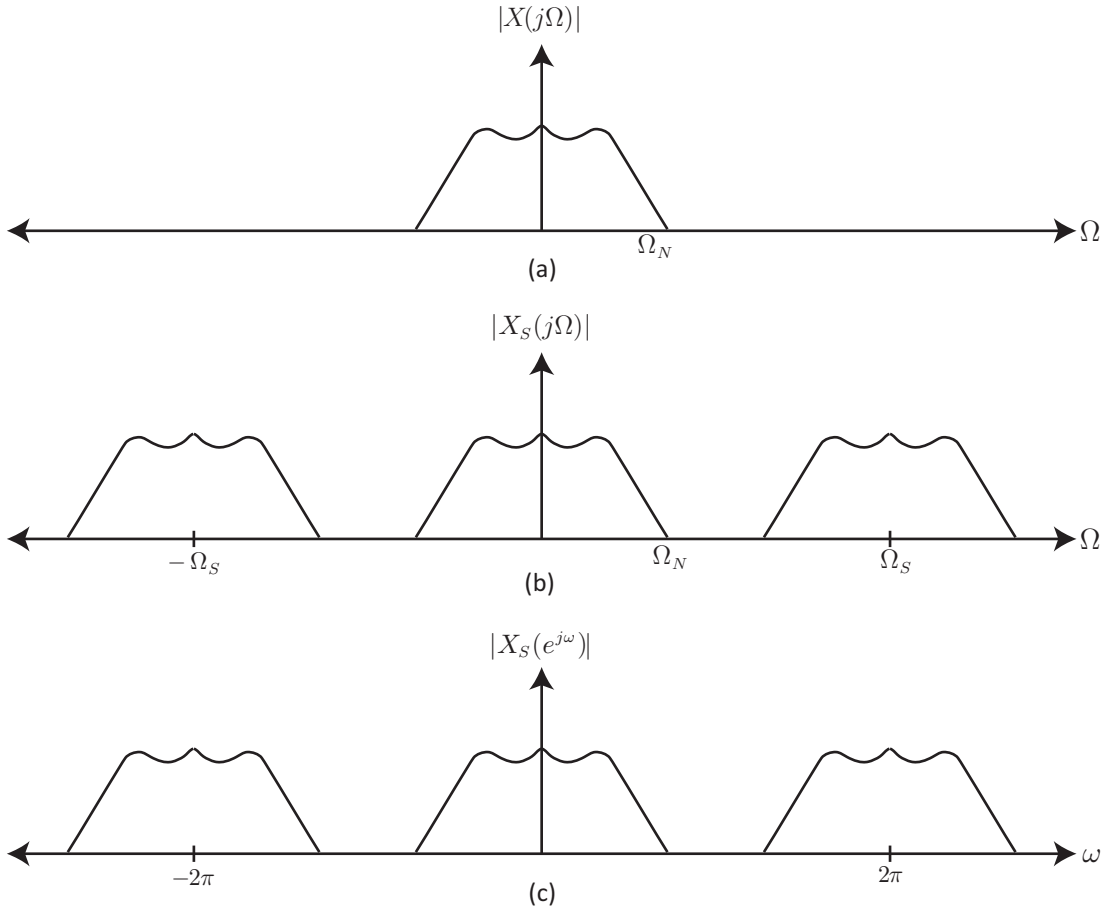


Figura 2.7: (a) Espectro do Sinal Original. (b) Espectro do Sinal Amostrado à Taxa de Nyquist. (c) Espectro do Sinal Discreto, com a frequência normalizada.

sinal separadas por Ω_S não encostam umas nas outras e, portanto, não modificam a banda do sinal. Neste sistema é possível a recuperação da cópia original, utilizando um filtro passa-baixas com frequência de corte em Ω_N.

Os conversores que possuem este modo de amostragem são chamados de **Conversores de Taxa de Nyquist**. São considerados deste tipo os conversores com taxa de amostragem entre $2\Omega_N$ e $10\Omega_N$. Quando a taxa de amostragem está acima destes valores, consideramos o conversor de sobreamostragem.

2.3.2 Subamostragem

A subamostragem (ou *undersampling*, em Inglês) ocorre quando a taxa de amostragem é menor que a taxa de Nyquist, ou seja $\Omega_S < 2\Omega_N$ [2, pp. 10-12]. O processo é retratado na Figura 2.8.

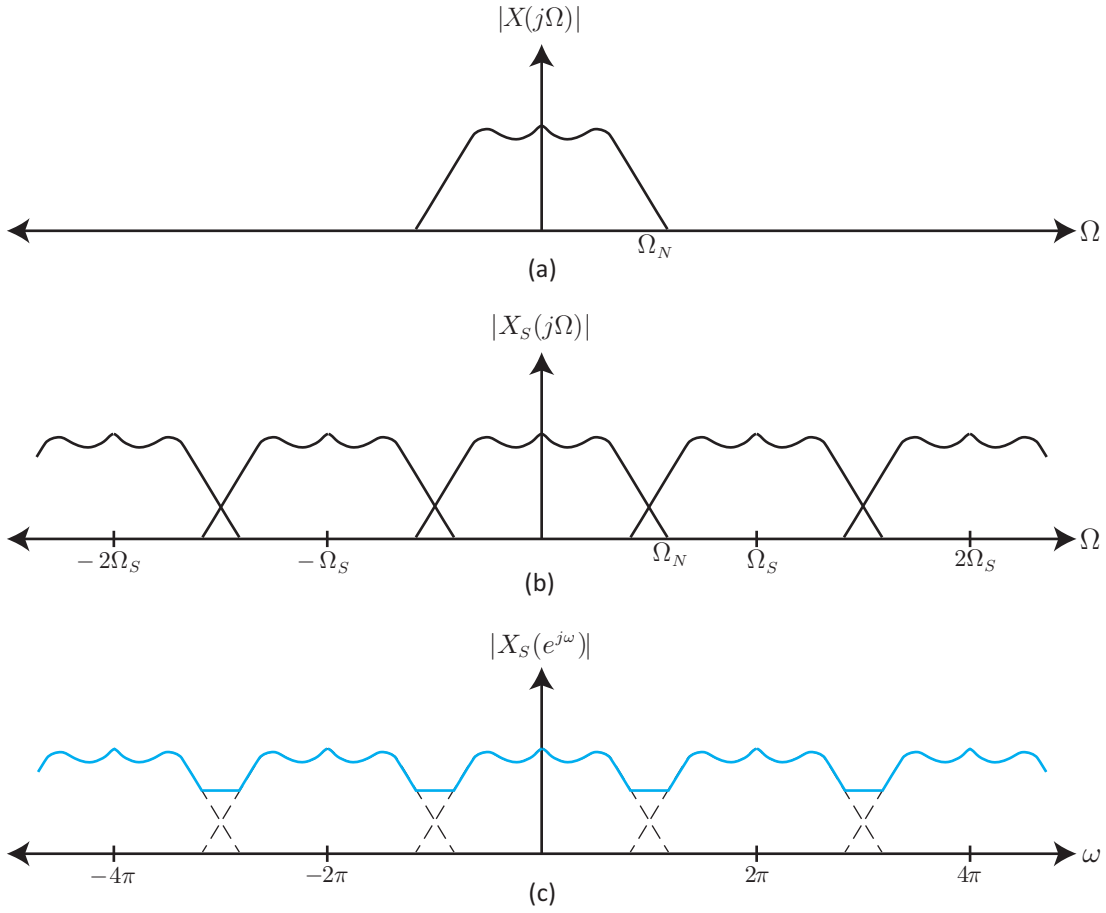


Figura 2.8: (a) Espectro do Sinal Original. (b) Espectro do Sinal Subamostrado. (c) Espectro do Sinal Discreto com o efeito do Aliasing.

Neste caso, diferente do anterior, as réplicas separadas por Ω_S tocam umas nas outras. As partes que se tocam se somam, formando um sinal contínuo no espectro de frequência. Este processo caracteriza o *Aliasing*, que é um fator bastante indesejado, já que, uma vez contaminado com o *Aliasing*, é impossível recuperar a banda original do sinal.

O *Aliasing* é um dos grandes problemas dos sistemas discretos, tanto é que é colocado um filtro *Anti-Aliasing* na entrada do sistema de conversão analógico-digital, limitando a banda do sinal de entrada.

2.3.3 Sobreamostragem

A sobreamostragem (ou *oversampling*, em Inglês) ocorre quando a taxa de amostragem é muito maior que a taxa de Nyquist, ou seja, $\Omega_S \gg 2\Omega_N$ [4, pp. 10-12].

Normalmente considera-se muito maior quando Ω_S é cerca de uma ordem de grandeza maior que Ω_N. O processo de sobreamostragem não é muito diferente do de amostragem à taxa de Nyquist, apenas aumenta o espaço entre as réplicas do sinal original. O efeito, então, é semelhante ao observado na Figura 2.7.

A sobreamostragem é muito usada em conversores Sigma-Delta, que são conversores altamente dependentes de uma Relação Sinal-Ruido alta. Aumentando a frequência de amostragem, aumenta-se também a relação sinal-ruído, elevando o número efetivo de bits (ENOB) destes conversores que, na prática, possuem 1 ou 2 bits.

Conversores com este tipo de amostragem são chamados de **Conversores de Sobreamostragem**. Por possuirem alta velocidade de conversão, já que $\Omega_S \gg 2\Omega_N$, não estão presentes no escopo deste projeto, já que requerem outro tipo de estratégia para serem testados.

2.4 Quantização

O processo de quantização é responsável por transformar os valores de amplitude contínuos do sinal em valores discretos. Como mencionado na Seção 2.2, o Quantizador mapeia o sinal de entrada nos códigos binários, de forma não-linear. Assim, o Quantizador pode ser interpretado como um sistema não-linear que transforma os valores do sinal de entrada $x[n]$ em valores quantizados $\hat{x}[n]$ [4, pp. 7-10]. Matematicamente, tem-se

$$\hat{x}[n] = Q(x[n]) \tag{2.18}$$

onde $Q(\bullet)$ é a operação de quantização.

Por simplicidade matemática, será suposto um bloco de quantização com as seguintes características:

- Uniforme, ou seja, seus degraus são igualmente espaçados.

- Simétrico, que opera entre limites de tensão simétricos, que serão chamados de A.

- Arrendonda para o valor quantizado mais próximo, ao invés de truncar ou arrendondar para cima.

Com estas suposições, uma curva característica pode ser traçada, como visto na Figura 2.9.

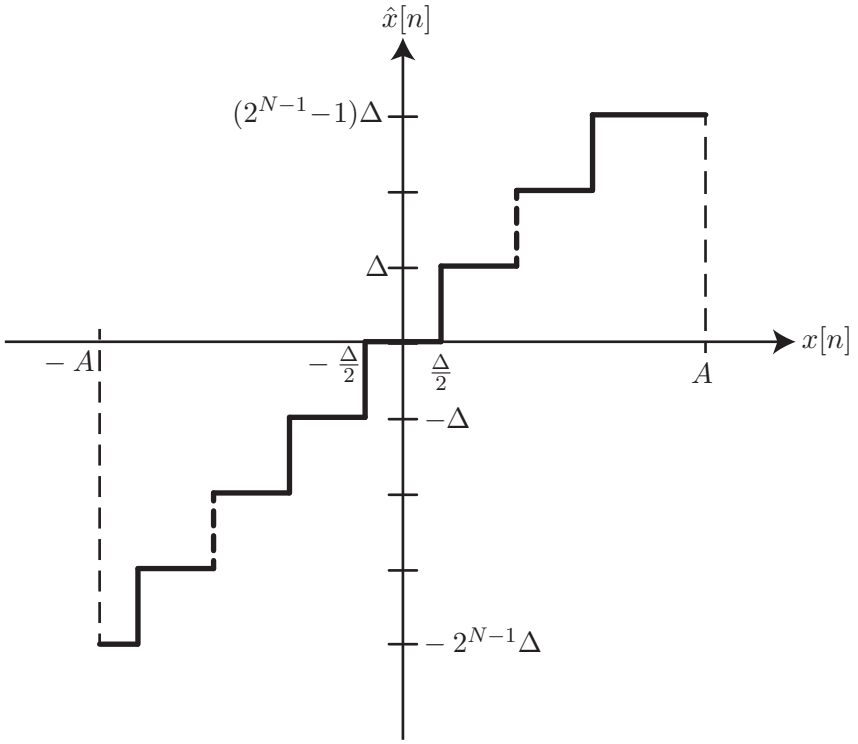


Figura 2.9: Curva Característica do Quantizador.

Alguns fatores desta curvas podem ser observados. A curva de mapeamento do quantizador é dividida em diversos degraus, que são chamados de **Degraus de Quantização**. A curva possui 2^N degraus, cada um com largura Δ, onde N é o número de bits do quantizador. Como os degraus ocupam a escala de $-A$ até A, o valor de Δ é

$$\Delta = \frac{A-(-A)}{2^N} = \frac{2A}{2^N} \tag{2.19}$$

Outro fator importante que pode ser observado na curva é que há uma quantidade finita de valores de $\hat{x}[n]$ usados para representar $x[n]$, que pode assumir qualquer

valor entre $-A$ e A. Pode-se concluir, então, que a operação $Q(\bullet)$ introduz um erro no sinal, que é chamado de erro de quantização.

2.4.1 Erro de Quantização

O bloco de quantização pode ser interpretado como na Figura 2.10, onde $e[n]$ é o erro de quantização [3, pp. 243-247].

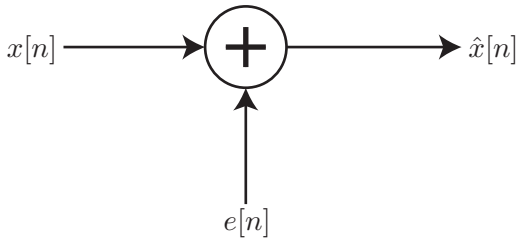


Figura 2.10: Modelo utilizado para o Quantizador.

Assim, por definição

$$e[n] = \hat{x}[n] - x[n] \tag{2.20}$$

Esta interpretação do quantizador é bastante útil caso o erro seja conhecido de antemão. Como nem sempre isto é possível, utiliza-se um método estatístico para a análise do erro de quantização. Esta análise pode ser vista em maiores detalhes em [3].

O modelo para análise é chamado de Pseudo-Ruído de Quantização (PQN, da sigla em Inglês) que assume que o erro de quantização se comporta como um *ruído branco uniforme aditivo*. O modelo assume que $e[n]$ é *descorrelacionado* com $x[n]$, o que é bastante aceito para um número de bits não muito baixo. Por conta desta análise, muitas vezes o erro de quantização é chamado de ruído de quantização na literatura.

Para o Quantizador que está sendo considerado, pode-se notar que o erro varia entre

$$-\frac{\Delta}{2} \leq e[n] < \frac{\Delta}{2} \tag{2.21}$$

e como é uma variável uniforme, a PDF (função de densidade de probabilidade) do erro pode ser definida como na Figura 2.11, onde

$$p_e(e) = \begin{cases} \frac{1}{\Delta}, & -\frac{\Delta}{2} \leq e[n] < \frac{\Delta}{2} \\ 0, & \text{para demais valores.} \end{cases} \tag{2.22}$$

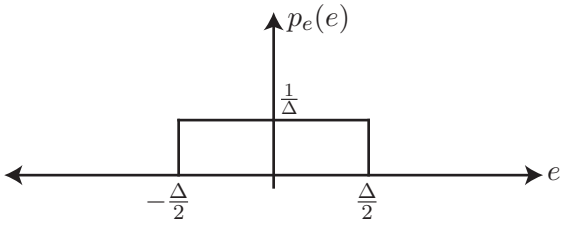


Figura 2.11: PDF do Erro de Quantização.

Com a PDF do sinal, a análise estatística pode ser feita ao obter a média e a variância do ruído de quantização.

$$\overline{e} = E[e] = \int_{-\infty}^{\infty} e \cdot p_e(e) de = \frac{1}{\Delta} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e \cdot de = 0 \tag{2.23}$$

$$\sigma_e^2 = E[e^2] = \int_{-\infty}^{\infty} e^2 \cdot p_e(e) de = \frac{1}{\Delta} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2 \cdot de = \frac{1}{\Delta} \left[\frac{e^3}{3} \right]_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = \frac{\Delta^2}{12} \tag{2.24}$$

Como o ruído é considerado branco, a sua autocorrelação é dada por

$$\phi_{ee}[m] = \sigma_e^2 \delta[m] \tag{2.25}$$

Ao obter a autocorrelação, o próximo passo é obter a densidade espectral de potência do ruído:

$$\Phi_{ee}(e^{j\omega}) = \sum_{m=-\infty}^{\infty} \phi_{ee}[m] e^{-j\omega m} = \sigma_e^2 \tag{2.26}$$

O resultado é um ruído com densidade espectral de potência constante no intervalo $[-\pi, \pi)$. A potência média do ruído é dada por

$$P_{ee}(e^{\jmath\omega}) = \phi_{ee}[0] = \sigma_e^2 = \frac{\Delta^2}{12} = \frac{4A^2}{12 \cdot 2^{2N}} \tag{2.27}$$

Assim, pode-se ver que quanto maior o número de bits, menor a potência média do ruído de quantização. Em outras palavras, aumentar o número de bits do Quantizador significa um ter um efeito do erro de quantização menor, que torna o Quantizador mais eficiente. Além disso, este resultado está diretamente relacionado à obtenção no valor teórico do Número Efetivo de Bits, que será feito no próximo capítulo.

Capítulo 3

Parâmetros e Testes

Existem inúmeros tipos de conversores no mercado, onde cada conversor possui distintas características: arquitetura, faixa de atuação, velocidade, entre outras. Tais diferenças tornam cada conversor adequado para alguma aplicação. Porém, conversores para mesmas aplicações são diferenciados por muitos parâmetros de medidas, que indicam a qualidade de sua performance - e consequentemente seu preço.

Os parâmetros se dividem em dois grupos: estáticos, que são obtidos através da curva característica do conversor; dinâmicos, que descrevem o comportamento frequencial do ADC.

Neste capítulo serão apresentados os parâmetros estáticos e dinâmicos relevantes para o projeto, bem como os testes para que estes parâmetros possam ser obtidos.

3.1 Especificações Gerais

Quando um conversor Analógico-Digital é projetado, alguns parâmetros já são definidos, pois são especificações nas quais o projeto é baseado.

3.1.1 Resolução

É o numero de bits que um conversor A/D utiliza para representar o valor analógico em sua entrada. É convencionalmente denotada como N.

3.1.2 Faixa de Tensão de Entrada

São os limites de voltagem de entrada no qual o conversor atua. Também chamado de *Faixa Nominal*.

3.1.3 Taxa de Conversão *(Throughput)*

É a capacidade de conversão do ADC. Geralmente é dada em Amostras por Segundo (SPS, na sigla em Inglês).

3.1.4 Faixa Dinâmica

É a relação entre o maior valor e o menor valor de entrada do conversor. É bastante encontrado em decibéis (dB). O menor valor é limitado pelo ruído do circuito, enquanto o maior é limitado pela tensão de alimentação.

3.2 Parâmetros Estáticos

A função de transferência de um conversor A/D é definida por sua curva característica, onde se relacionam entrada analógica e saída digital. Numa curva característica ideal, todos os Degraus de Quantização possuem mesmo tamanho, como visto no Capítulo 2.

Numa curva real, entretanto, existem diversas imprecisões, que são mensuradas por estes parâmetros estáticos. Existem mais parâmetros que os que serão mostrados nesta seção. Porém, serão apresentados os mais relevantes na literatura e ao projeto em questão.

3.2.1 Voltagem de Offset

O *offset* em um conversor A/D é definido como o deslocamento feito pela curva característica no plano $A \times D$. Como vimos anteriormente, a saída de um ADC ideal deve ser zero apenas quando a entrada analógica está entre 0 e $\frac{V_{LSB}}{2}$. Na prática, o zero na saída pode permanecer para tensões maiores ou menores que

estas, ou seja, o nível de transição entre a saída binária 0 e 1 pode variar. Esta variação é chamada de *offset*. Seu efeito pode ser observado na Figura 3.1.

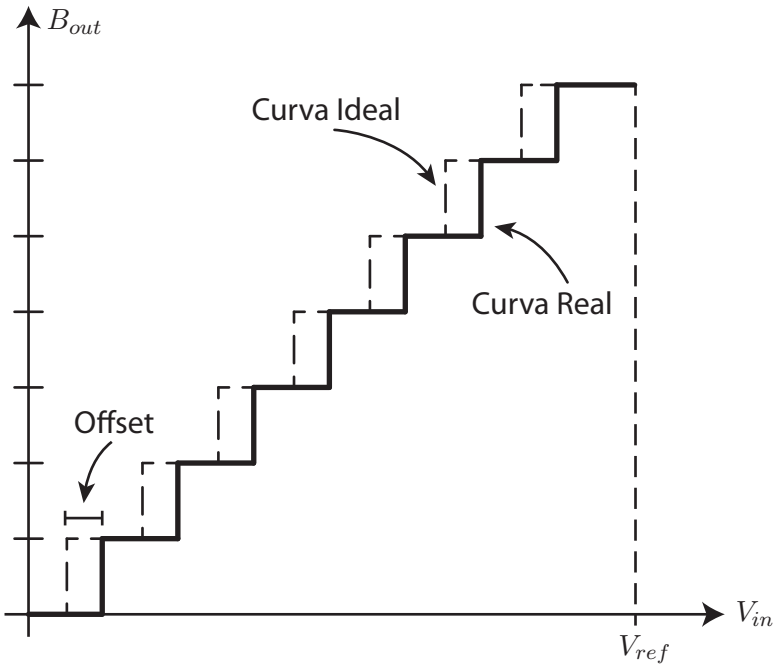


Figura 3.1: Efeito da Voltagem de *Offset* na curva característica.

3.2.2 Erro de Ganho

É a diferença entre a inclinação da curva característica ideal e a curva real do sistema. Tal inclinação pode ser obtida ao interpolar a curva com um reta, que vai do zero ao limite máximo da escala. É importante observar que o erro de ganho é obtido desconsiderando o *offset*, o que faz a reta partir do zero, e a curva ideal é definida com ganho unitário. O efeito do erro de ganho na função de transferência do conversor pode ser visto na Figura 3.2.

3.2.3 Erro Diferencial de Não-Linearidade (DNL)

É a variação no tamanho dos degraus analógicos do conversor. Para o ADC ideal, todos os degraus são uniformes, de tamanho V_{LSB}. Na prática, estes degraus tem tamanhos distintos, e a diferença entre o tamanho do degrau real e o degrau ideal é chamada de DNL.

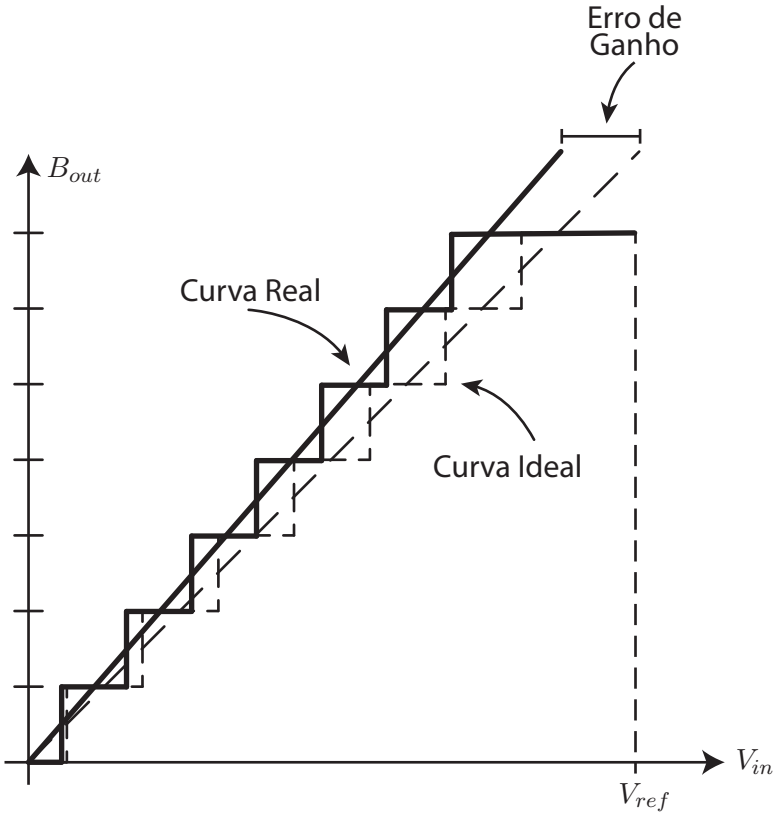


Figura 3.2: Efeito do Erro de Ganho na curva característica.

A DNL pode ser apresentada de duas formas: ou como um gráfico, que relaciona essa diferença no degrau com o código de saída que este degrau representa, como é visto na figura 3.3 ou pode ser apresentada também como o maior valor absoluto deste gráfico. Na literatura acadêmica e em *datasheets*, a DNL é encontrada das duas formas. É interessante notar que a DNL é normalmente dada em unidades de V_{LSB}.

Seja Δ o tamanho ideal do degrau, e $\Delta_r(k)$ o tamanho real do degrau que representa o código k, ou seja, a diferença entre os níveis de transição entre os códigos $\{k, k-1\}$ e entre $\{k+1, k\}$. A DNL(k), ou seja, a DNL do código k, é definida por:

$$\text{DNL}(k) = \frac{\Delta_r(k) - \Delta}{\Delta} \tag{3.1}$$

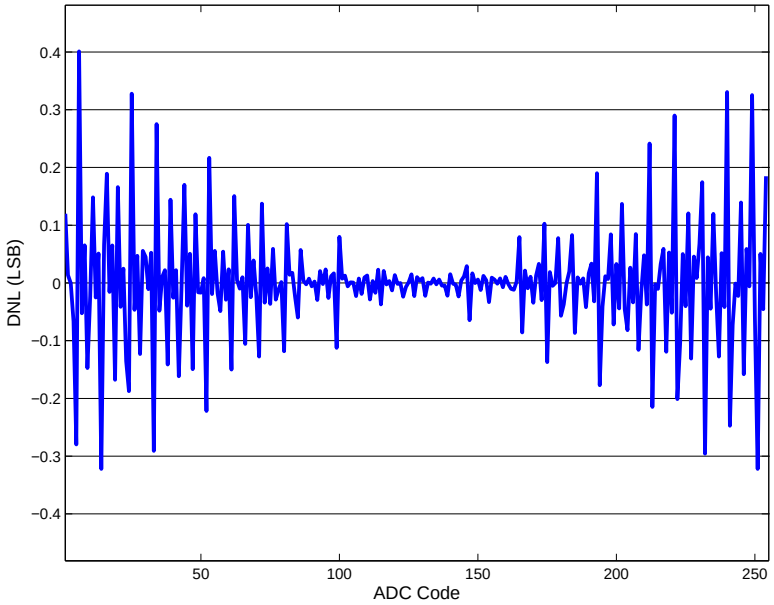


Figura 3.3: Gráfico da DNL vs. o Código Binário na Saída.

O valor absoluto da DNL é:

$$\text{DNL} = \max\{|DNL(k)|\} \tag{3.2}$$

3.2.4 Erro Integral de Não-Linearidade (INL)

A diferença entre a reta de interpolação e a função de transferência do conversor analógico-digital define a INL. Existem dois modos de se obter a reta de interpolação para o cálculo da INL:

- **Reta Ideal:** Utiliza-se a reta que vai do zero ao fundo de escala do conversor.
- **Regressão Linear:** Utiliza-se a reta que melhor se encaixa nos níveis de transição do ADC. A reta obtida neste método é a que mais se aproxima da Função de Transferência do conversor, pois leva em consideração erros de ganho e *offset*.

Destes dois metodos, o segundo é o mais utilizado e aceito na literatura, pois fornece resultados mais confiáveis e melhores, reduzindo a INL máxima. Com isso,

precisamos encontrar o Ganho e o *Offset* para definir a INL. Para isso, utilizamos a equação (72) de [5], que é definida por

$$G \times T[k] + V_{off} + \epsilon[k] = \Delta(k-1) + T[1] \tag{3.3}$$

onde $T[k]$ é o nível de transição entre os códigos binários k e $k-1$, o que implica que $T[1]$ é o nível de transição entre 1 e 0. Δ é o tamanho ideal do degrau de quantização, G é o ganho, V_{off} é o *offset* e $\epsilon[k]$ é o erro residual correspondente ao código k.

G e V_{off} podem ser encontrados utilizando o método dos mínimos quadrados, que tem o intuito de minimizar a soma dos quadrados dos resíduos $\epsilon[k]$. Estes valores já foram calculados em [5, pp. 43-44], para valores gerais de $T[k]$. Assim, temos para o ganho

$$G = \frac{\Delta(2^N-1)\cdot\left(\sum_{k=1}^{2^N-1} kT[k] - 2^{(N-1)}\sum_{k=1}^{2^N-1} T[k]\right)}{(2^N-1)\sum_{k=1}^{2^N-1} T^2[k] - \left(\sum_{k=1}^{2^N-1} T[k]\right)^2} \tag{3.4}$$

e para o *offset*

$$V_{off} = T[1] + \Delta(2^{(N-1)}-1) - \frac{G}{2^N-1}\sum_{k=1}^{2^N-1} T[k] \tag{3.5}$$

A não-linearidade integral pode ser definida, então, utilizando os resíduos, resultando em

$$\mathrm{INL}[k] = \frac{\epsilon[k]}{\Delta} \tag{3.6}$$

onde $\mathrm{INL}[k]$ é a INL do código k, dada em unidades de V_{LSB}. Assim como a DNL, a INL também é exibida em forma de gráfico, que relaciona o erro de não-linearidade com o respectivo código, além de exibir seu máximo valor, que é:

$$\mathrm{INL} = \max\{|INL[k]|\} \tag{3.7}$$

3.2.5 Falha de Códigos

Uma falha de código ocorre quando certo valor digital que idealmente deveria aparecer na saída não é exibido. Uma condição suficiente para que não haja falha de código é que a DNL seja maior que -1 para todos os códigos, já que a DNL é -1 quando o tamanho do degrau é zero, ou seja, não há degrau para aquele código. O efeito da falha de código pode ser visto na Figura 3.4. Pode-se notar que não há nenhum degrau para representar o código k na imagem.

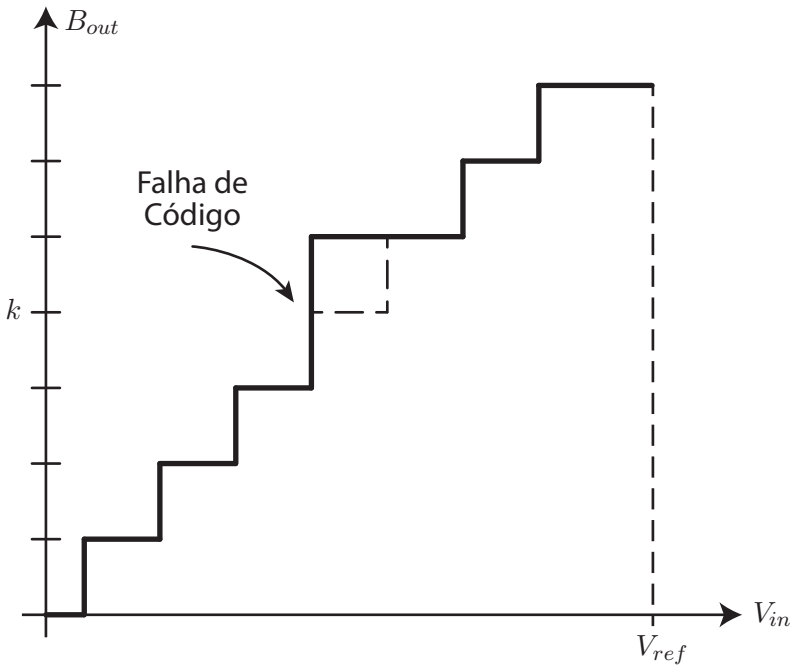


Figura 3.4: Efeito da Falha de Código na curva característica.

3.3 Parâmetros Dinâmicos

Além de analisar a curva característica do conversor, é importante também identificar o comportamento do conversor no domínio da frequência. A resposta em frequência do conversor e a velocidade de resposta definem os parâmetros de medidas dinâmicos. A performance do conversor é bastante influenciada pela faixa de frequência na qual o sinal de entrada se encontra, o que tornam estes parâmetros fundamentais para caracterizar o conversor.

3.3.1 Relação Sinal-Ruido (SNR)

É a razão entre a potência média do sinal de entrada e a potência média do ruido gerado pelo conversor, equivalente na entrada, tanto pelo circuito, quanto pelo ruído de quantização. A SNR leva em conta apenas a faixa de frequência de Nyquist, ou seja, entre 0 e $\frac{f_s}{2}$. O valor da SNR é normalmente dado em decibéis (dB), ou seja:

$$\text{SNR}\Big|_{dB} = 10\log_{10}\left(\frac{\overline{P_s}}{\overline{P_n}}\right) \tag{3.8}$$

O valor da SNR pode ser estimado, considerando um conversor ideal. Como visto anteriormente, apesar de ser ideal, o conversor analógico-digital é afetado por um erro de quantização, que é interpretado como ruído. A potência deste ruído foi calculada na seção 2.4, e é dada por

$$\overline{P_n} = \frac{\Delta^2}{12} \tag{3.9}$$

onde delta é o tamanho de degrau de quantização. Considerando um sinal de entrada senoidal de frequência ω, de amplitude A, que ocupa toda a faixa dinâmica, a potência média deste sinal pode ser facilmente calculada.

$$\overline{P_s} = \frac{1}{T}\int_{-T/2}^{T/2}[A\sin(\omega t)]^2 dt = \frac{A^2}{T}\left[\frac{t}{2}\right]_{-T/2}^{T/2} = \frac{A^2}{2} \tag{3.10}$$

Como toda a faixa dinâmica do conversor possui 2^N degraus, onde N é o número de bits, e A é metade deste valor (sinal vai de -A a A), temos que

$$A = \frac{2^N\Delta}{2} \tag{3.11}$$

e aplicando a equação 3.10, tem-se:

$$\overline{P_s} = \frac{2^{2N}\Delta^2}{8} \tag{3.12}$$

Com os valores de $\overline{P_s}$ e $\overline{P_n}$, podemos obter a SNR teórica:

$$\text{SNR}\Big|_{dB} = 10\log_{10}\left(\frac{\frac{2^{2N}\Delta^2}{8}}{\frac{\Delta^2}{12}}\right) = 10\log_{10}\left(2^{2N}\frac{3}{2}\right) \tag{3.13}$$

Simplificando o logaritmo, temos:

$$\text{SNR}\Big|_{dB} = 6.02 \times N + 1.76 \tag{3.14}$$

Este resultado impõe um limite teórico para o valor da SNR, já que em conversores reais, diversos erros como o DNL e o INL, aumentam a potência do ruído. O resultado também relaciona o número de bits com a SNR, que é importante para a definição do número efetivo de bits.

3.3.2 Relação Sinal-Ruido-e-Distorção (SINAD)

É bastante semelhante a SNR, porém, é levado em conta a potência dada pelos harmônicos do sinal de entrada, que também são componentes indesejadas no espectro do conversor. Assim, a SINAD é definida pela razão entre a Potência média do Sinal e a soma das Potências médias do ruído e dos harmônicos.

$$\text{SINAD}\Big|_{dB} = 10\log_{10}\left(\frac{\overline{P_s}}{\overline{P_d} + \overline{P_n}}\right) \tag{3.15}$$

3.3.3 Número Efetivo de Bits (ENOB)

É uma medida alternativa da relação sinal-ruído. Neste caso, os valores são dados em bits. Utilizando a equação 3.14, e isolando o número de bits N, tem-se:

$$\text{ENOB} = \frac{\text{SNR}|_{dB} - 1.76}{6.02} \tag{3.16}$$

Considerando um sistema de conversão mais realista, utiliza-se a SINAD no lugar da SNR. A definição do ENOB fica da seguinte forma:

$$\text{ENOB} = \frac{\text{SINAD}|_{dB} - 1.76}{6.02} \tag{3.17}$$

3.3.4 Distorção Harmônica Total (THD)

A distorção harmônica Total (THD) é uma medida que relaciona a quantidade de harmônicos produzidos pelo conversor com o sinal de entrada do conversor. A THD é definida como a razão entre o valor RMS dos harmônicos e o valor RMS do sinal de entrada. Assim, temos

$$\text{THD} = \frac{\sqrt{\sum_{i=2}^{n} V_i^2}}{\sqrt{V_s^2}} \tag{3.18}$$

onde V_i é a amplitude do i-ésimo harmônico e V_s é a amplitude do sinal. Como a potência é proporcional ao quadrado da amplitude, ou seja $P_n \propto V_n^2$, a THD pode ser reescrita da seguinte forma

$$\text{THD} = \sqrt{\frac{\sum_{i=2}^{n} P_i}{P_s}} \tag{3.19}$$

que em outros termos diz que a THD é a raiz quadrada da razão entre a soma das Potências dos harmônicos e a Potência do Sinal. Como a THD pode ser dada também em decibéis, a equação 3.19 pode ser reescrita como:

$$\text{THD}\Big|_{dB} = 10\log_{10}\left(\frac{\sum_{i=2}^{n} P_i}{P_s}\right) \tag{3.20}$$

3.3.5 Faixa Dinâmica Livre de Espúrios (SFDR)

A SFDR é um parâmetro importante para sistemas de comunicação, e é definido como a raiz quadrada da razão entre a Potência do sinal e a maior Potência dentre a Potência dos harmônicos. O pico causado no espectro pelos harmônicos são chamados de **espúrios**, e o pico de maior valor é o maior espúrio do espectro. Assim, a Faixa Dinâmica Livre de Espúrios, é definida por

$$\text{SFDR}\Big|_{dB} = 10\log_{10}\left(\frac{P_s}{max[P_i]}\right) \tag{3.21}$$

onde P_i é a potência do i-ésimo harmônico e P_s é a potência do sinal. A SFRD é

proporcional à amplitude do sinal, que se estiver num nível muito baixo pode ser suprimida por este espúrio no espectro.

3.4 Método dos Histogramas

O método dos histogramas foi o teste escolhido para obter os parâmetros estáticos dos conversores Analógico-Digitais neste projeto. Os resultados que serão apresentados aqui foram obtidos das referências [6] e [7]. Especialmente o trabalho apresentado por Blair em [6] é excelente e sua leitura é bastante recomendada pois esta seção é totalmente baseada neste.

O princípio do método é bastante simples: são feitas diversas conversões de um sinal senoidal com mesma frequência e amplitude e cria-se um histograma com as amostras convertidas pelo ADC. A partir deste histograma, são obtidos, estatisticamente, os níveis de transição do dispositivo em teste. Com os níveis de transição, todas as figuras de mérito estáticas do conversor podem ser obtidas.

Apesar de ser bastante simples, diversos parâmetros devem ser calculados para que o método funcione e garanta sua precisão, pois como é um método estatístico, possui um fator de incerteza. Porém, esta incerteza pode ser tão pequena quanto desejada [6], o que torna método bastante robusto. Outro fator que aumenta a robustez do método é o fato de se utilizar ondas senoidais como entrada, que são obtidas com muito mais precisão que ondas triangulares, que costumavam ser a escolha para este tipo de teste.

Algumas definições devem ser feitas para o cálculo do método. A notação utilizada aqui é semelhante a utilizada por Blair. Tem-se:

- $T[k]$ é o nível de transição entre os códigos k e $k-1$
- $W[k]$ é o largura do degrau do código k, ou seja,

$$W[k] = T[k+1] - T[k]$$

- N é o número de bits do conversor
- V é a faixa nominal reduzida, que é a diferença entre os niveis de transição mais extremos, ou seja, $V = T[2^N - 1] - T[1]$
- Q_{avg} é o tamanho médio dos degraus de quantização, cujo valor é

$$Q_{avg} = \frac{V}{2^N - 2}$$

Com todas estas definições, as equação obtida para a DNL pode ser adaptada, obtendo

$$\mathrm{DNL}[k] = \frac{W[k] - Q_{avg}}{Q_{avg}} \tag{3.22}$$

que é diferente da definição dada por Blair em [6], pois o erro de ganho não foi incluído na definição utilizada neste trabalho [2]. Já o Ganho, *Offset* e a INL continuam a ser obtidos através das equações dadas na seção 3.2.4.

3.4.1 Cálculo dos Níveis de Transição

O cálculo de $T[k]$ é feito a partir do histograma das amostras do ADC. O conversor executa diversas pequenas conversões, de modo a garantir que a fase das amostras seja uniformemente distribuida entre 0 e 2π [6]. Ao todo serão feitas R conversões com M amostras, obtendo um número total de S amostras, onde

$$S = R \cdot M \tag{3.23}$$

O sinal a ser amostrado é da forma

$$v(t) = A\sin(\omega t + \phi) + d \tag{3.24}$$

onde A, ω e d, neste projeto, são conhecidos. O fato destes valores serem desconhecidos nao afeta a INL e a DNL, somente ocasionariam erros no Ganho e no *Offset*

[6].

Após a obtenção das S amostras, elas são distribuídas num histograma. Este histograma possui uma classe para cada valor digital do conversor, ou seja, possui 2^N classes. Seja $h[i]$ o número de amostras na classe i, define-se, então, $ch[k]$ como

$$ch[k] = \sum_{i=0}^{k} h[i] \tag{3.25}$$

que é o número total de amostras nas classes com valor menor que k. Os níveis de transição são então estimados [6],[7] por

$$T[k] = d - A\cos\left(\frac{\pi ch[k-1]}{S}\right) \tag{3.26}$$

que serão utilizados para o cálculo de todos os parâmetros estáticos deste projeto.

Para garantir que a estimativa dos valores de transição seja precisa, é necessário o cálculo de alguns parâmetros-chave para este teste. Eles serão apresentados nas próximas sub-seções.

3.4.2 Tolerância e Incerteza

Em [6], estas são as duas quantidades utilizadas no método para descrever sua precisão estatística. O valor B é a Tolerância do resultado e u é a Incerteza deste resultado. Estes valores podem ser definidos em relação aos níveis de transição $T[k]$ ou em relação à largura dos degraus $W[k]$. Como os níveis de transição são utilizados diretamente para o cálculo da INL e a largura dos degraus para o cálculo da DNL, dizemos que a tolerância e a incerteza são definidos em relação à INL (quando for para $T[k]$) ou à DNL (quando for para $W[k]$).

Podemos definir a tolerância e incerteza em relação à DNL. Seja W_R a largura do degrau real e W_M a largura do degrau medida pelo método. Pode-se dizer que a largura dos degraus é medida com uma certeza de $1-u$ (incerteza de u) quando há

uma probabilidade maior ou igual a $1-u$ de que:

$$\frac{W_R}{1+B} \leq W_M \leq W_R(1+B) \tag{3.27}$$

A definição em relação a INL é semelhante. Seja T_R o nível de transição real e T_M o nível de transição medido. Pode-se dizer que o nível de transição é medido com uma certeza de $1-u$ (incerteza de u) quando há uma probabilidade maior ou igual a $1-u$ de que:

$$T_R - B \cdot Q_{avg} \leq T_M \leq T_R + B \cdot Q_{avg} \tag{3.28}$$

Sabendo a influência da tolerância e da incerteza nas medidas, pode-se definir os valores para ambos. É importante notar que o valor para cada uma delas é dado por quem está executando o teste, ou seja, o usuário. Valores muito pequenos para cada um dos parâmetros resulta em um número mínimo de amostras muito elevado.

3.4.3 Frequência da Senóide de Teste

A frequência da onda senoidal de teste deve ser cuidadosamente calculada, para garantir que a fase da primeira amostra do sinal seja uniformemente e randomicamente distribuída entre 0 e 2π. Para isso, deve obrigatoriamente haver um numero inteiro de ciclos numa mesma conversão, e o número de amostras deve ser relativamente primo ao número de ciclos [6], [5]. Para isso, é feito o seguinte procedimento: é escolhido um número de ciclos D e um número de amostras M para cada conversão, onde $M+1$ é um número natural multiplo de D, ou seja

$$\frac{M+1}{D} = k, \quad \text{onde } k \in \mathbb{N} \tag{3.29}$$

A frequência do sinal de testes ,f_{in}, é dada por

$$\frac{f_{in}}{f_s} = \frac{D}{M} \tag{3.30}$$

onde f_s é a frequência de amostragem do sinal de entrada.

Com esses cálculos, poderiamos escolher M tão grande quanto quisessemos, para reduzir o número de conversões mínima. Porém, existe um número máximo de amostras que pode ser utilizado, que possui uma relação com a precisão da frequência dada pelo instrumento que irá gerar a senóide. Seja ρ a razão entre a frequência do sinal e a frequência de amostragem e $\Delta\rho$ a imprecisão desta razão. Estes valores devem satisfazer a seguinte condição [6]:

$$\frac{\Delta\rho}{\rho} \leq \begin{cases} \frac{1}{4(D-1)M} & D > 1, \\ \frac{1}{4M} & D = 1. \end{cases} \tag{3.31}$$

Pode-se notar que, para valores maiores de D e M, há uma necessidade de uma precisão cada vez maior nas frequências. É importante escolher o valor máximo para M, já que isto reduz o número máximo de conversões que devem ser feitas. Este valor é facilmente obtido a partir da escolha de um D qualquer, utilizando a equação 3.31.

3.4.4 Tensão de *Overdrive*

É um pequeno aumento na voltagem de pico da senóide de entrada para que haja o controle da influência do erro na senóide de entrada no resultado. Segundo Blair, em [6], o erro recorrente de ruído na senóide de entrada é maior perto dos picos da mesma. Com esse overdrive, é possível deixar estes erros longe das medidas, sem contar que o efeito do ruído pode ser tão pequeno quanto desejamos, deixando o *overdrive* tão grande quanto necessário.

O *overdrive* é proporcional à quantidade de ruído na senóide de entrada. Na ausência de ruído, o *overdrive* deve ser suficiente para que haja sempre uma amostra nos picos do sinal [5].

Para obter a tolerância B nas larguras dos degraus, o *overdrive* é dado por [6]

$$V_{OD} \geq \sigma \times \max\left\{3, \sqrt{\frac{3}{2B}}\right\} \tag{3.32}$$

onde σ é o nível de ruído do circuito e do sinal de entrada somados. Vale ressaltar que neste ruído não é incluído o ruído de quantização. Para obter a tolerância B nos níveis de transição, o *overdrive* é dado por

$$V_{OD} \geq \sigma \times \max\left\{2, \frac{\sigma 2^N}{VB}\right\} \tag{3.33}$$

Estes valores de V_{OD} são suficientes para deixar o erro devido ao ruído menor que $B/3$, que quando somados aos erros estatísticos, podem ser descartados.

3.4.5 Número Mínimo de Conversões

Este parâmetro é provavelmente o mais importante que deve ser observado. Ele depende de diversos fatores, entre eles a tolerância e a incerteza da medida. Este valor é dado pela equação (12) em [6], cujo valor é

$$R = C\left[\frac{2^{N-1}K_u}{B}\right]^2 \left[\alpha\frac{\pi}{M}\right] \cdot \left\{1.13\left(\frac{\sigma*}{V}\right) + 0.2\left[\alpha\frac{\pi}{M}\right]\right\} \tag{3.34}$$

onde:

R é o Número mínimo de conversões,

C é igual a 1 quando a Tolerância é definida para a INL e 2 para a DNL,

M é o número de amostras por conversão,

α é igual a $1 + 2V_{OD}/V$,

$\sigma*$ é o valor RMS do rúido quando a Tolerância é definida para a INL,

$\sigma*$ é o $\min[\sigma, Q/1.13]$ quando a Tolerância é definida para a DNL,

K_u é igual a $Z_{u/2}$ para obter a tolerância desejada em níveis de transição ou largura de degraus individuais,

K_u é igual a $Z_{N,u/2}$ para obter a tolerância desejada em níveis de transição ou largura de degraus no pior caso.

Os outros parâmetros seguem a nomenclatura dada durante esta seção inteira. Por se tratar de uma equação muito complexa, não iremos derivá-la. Caso seja de interesse do leitor, recomenda-se a leitura de [6, seção V]. Os valores de $Z_{N,u/2}$ são dados por

$$Z_{N,u/2} = \sqrt{2}\,\mathrm{erfc}^{-1}[1-(1-u)^{2^{-N}}] \tag{3.35}$$

e os de $Z_{u/2}$ são os mesmos, porém para $N = 0$.

Podemos observar alguns fatos da equação 3.34. Primeiramente, podemos ver que seu valor é bastante elevado. Isto porque o termo no primeiros colchetes é muito grande, pois como B é menor que 1, e 2^{N-1} pode ficar muito grande, dependendo do número de *bits* do conversor. O termo no segundo colchete vai ficando menor quando o número de amostras em uma conversão é maior. Por isso que é interessante usar o número M maior possível, como vimos anteriormente. Os termos entre as chaves são interessante, pois quando o ruído é muito grande, o primeiro termo domina. A medida que o ruído vai diminuindo, o segundo termo domina este valor, que é inversamente proporcional ao número de amostras, podendo gerar, inclusive, um fator $1/M^2$ caso o ruído seja aproximadamente zero.

Como o número mínimo de conversões é bastante elevado, o número de amostras total, dado por $S = R \cdot M$ fica muito grande. Nos testes que fizemos, para um conversor de 10 bits, usando tolerância $B = 0.1$ e incerteza de 1% na DNL, o número de amostras foi de mais de 3 milhões. Isto faz que a plataforma demore um pouco para realizar todo o teste.

3.5 Teste da FFT

A Transformada Rápida de Fourier (FFT, da sigla em inglês) é uma ferramenta muito poderosa para a análise dos parâmetros dinâmicos. Por se tratar de um algoritmo amplamente conhecido e utilizado, sua implementação é bastante direta,

e como utilizaremos o MATLAB, ele já possui uma função que executa a FFT implementada.

A FFT mapeia o sinal de entrada senoidal discreto no tempo e retorna um sinal no domínio da frequência. Assim, é possível separar a componente pura do sinal e as componentes de ruído e distorção. Ao separá-las, é possível determinar a potência de cada componente, o que é a base do cálculo de todos os parâmetros dinâmicos. Pode-se notar que o teste é bastante descomplicado. No entanto, devem ser tomados alguns cuidados no cálculo da frequência e número de pontos do sinal.

O sinal que resulta da FFT é dividido em espaços discretos no domínio da frequência. Estes espaços são proporcionais ao número de pontos da FFT, que representam frequências desde DC até a frequência de amostragem f_s do sinal de entrada [3]. Por conta da simetria e da periodicidade da FFT, a frequência do sinal de entrada deve ser menor que a metade da frequência de amostragem, pelo teorema de Nyquist-Shannon.

Para o cálculo da frequência, desejamos que ela se localize no centro do seu espaço na FFT. Seja uma FFT feita com N_{FFT} pontos e com frequência de amostragem f_s. Cada espaço na frequência tem seu centro localizado nos seguintes pontos, que são possíveis frequências para o sinal de entrada

$$f_k = \frac{f_s}{N_{FFT}} \cdot k \ , \qquad \text{onde } 0 < k \leq \frac{N_{FFT}}{2} \tag{3.36}$$

onde k obviamente não pode ser 0 pois torna o sinal DC. Apesar de todas estas frequências serem possíveis, para que o teste não seja afetado por erros dinâmicos e *aliasing*, deve ser escolhido um valor para k muito menor que $\frac{N_{FFT}}{2}$. Assim, a frequência do sinal de teste possui a seguinte critério de escolha

$$f_k = \frac{f_s}{N_{FFT}} \cdot k \ , \qquad \text{onde } 0 < k \ll \frac{N_{FFT}}{2} \tag{3.37}$$

O número de pontos N_{FFT} a ser utilizado fica a critério do usuário, já que um menor número de pontos deixa o teste mais rápido porém menos preciso. A única

restrição é que o número de pontos seja uma potência de 2, que é um requisito do algoritmo da FFT [3].

Capítulo 4

Hardware da Plataforma de Teste

O *hardware* da plataforma de testes é o responsável pela interface entre o *software* de análise de dados e o Conversor Analógico-Digital. Sua principal função é fazer a aquisição de dados do ADC e entregar os resultados da conversão para o computador. Ele também deve saber o momento exato que a aquisição deve ser feita, para que não haja erro na medida. Todas estas tarefas devem ser feitas por um sistema simples e que possa ser utilizado por diferentes tipos de computadores sem muito esforço.

Neste capítulo serão apresentadas as escolhas de projeto feitas para alcançar as especificações, além do circuito final do projeto, o desenho do circuito impresso e o produto final.

4.1 Projeto do Hardware

O projeto do*hardware* da plataforma foi feito com duas premissas: ser simples e confiável. Para isso, optamos por utilizar um microcontrolador, que seria o grande cérebro do projeto, e reduziria o número de componentes discretos utilizados no circuito. Assim, a fase de projeto consistiu em fazer a escolha dos componentes e dos tipos de conversores à serem medidos, já que o trabalho mais pesado será feito pelo *firmware* do microcontrolador.

4.1.1 Escolha do Microcontrolador

O microcontrolador, também denotado por MCU, é o componente mais importante do *hardware*. Ele deve ser capaz de fazer a aquisição de dados na velocidade em que o conversor A/D entrega sua saída, e ser confiável a ponto de não prejudicar a medida que foi feita. Os dados devem ser entregues ao computador a partir de uma porta USB, que é encontrada atualmente em todos os computadores, mais frequentemente que portas seriais.

O componente escolhido para a aquisição de dados foi o **PIC18F4550**, da Microchip. Este PIC possui um número elevado de pinos, 40, onde 35 são pinos de entrada/saida (I/O, da sigla de Inglês) [8]. Este CI também já foi utilizado por outros projetos no laboratório. Portanto, algumas unidades do PIC podiam ser utilizadas sem acréscimo de custo ao projeto.

Inicialmente, o *hardware* foi planejado com um sistema de geração de fases de controle automático. Porém, após observarmos diversos ADCs disponíveis no mercado, notamos que muitos deles tem padrão de fases de controle completamente diferentes uns dos outros, com uma característica comum de possuir uma fase que indica o momento em que a leitura da saída deve ser executada. Por isso, foi feita a opção por deixar a cargo do usuário a geração das fases de controle, já que é uma tarefa simples, porém bastante específica para cada conversor, o que seria bastante complicado de se generalizar. O importante é enviar para o microcontrolador principal esta fase de leitura, já que, sem ela, o MCU não consegue saber quando a saída efetivamente mudou.

4.1.2 Comunicação USB

A comunicação USB será utilizada no *hardware* para enviar a informação obtida do ADC para o computador. Esta informação será processada pelo MATLAB, que será o programa responsável por este interfaceamento.

Existem diversos protocolos para que a comunicação USB possa ser feita. Dentre eles, há três principais [9], que são implementáveis no MCU principal do projeto:

Classe de Dispositivo de Comunicação (CDC)

É o protocolo mais simples. Ele emula uma porta serial através de um *driver* que deve ser instalado. Sua grande vantagem é a simplicidade de implementação de *software*, já que grandes *softwares* comerciais, como MATLAB e LABVIEW possuem funções bem símples para a comunicação serial. O ponto negativo fica por conta da instalação do *driver*, que não é feita automaticamente como em dispositivos *plug-and-play*. Ele também é um protocolo bastante antigo, pois surgiu na época em que as portas seriais estavam sendo substituídas pela USB. Portanto, não há tantas atualizações para ele.

Dispositivos de Interface Humana (HID)

Este protocolo é o mais utilizado pela comodidade de não necessitar de uma instalação de *driver*, tornando o dispositivo mais portátil. Porém, sua implementação em *software* é muito mais complexa, já que devem ser utilizadas bibliotecas externas em alguns *softwares*, dentre eles o MATLAB, que foi o escolhido neste projeto. Isto não é uma tarefa simples, aumentando a complexidade de uso deste protocolo que, além disso, tem uma velocidade baixa de transmissão, de 64kBytes/s.

Dispositivos de Armazenamento de Massa (MSC)

A classe de dispositivos de armazenamento em massa é a mais utilizada quando se deseja uma transferência de arquivos de modo bi-direcional [9]. Este protocolo consegue atingir a maior velocidade de transferência que, no PIC utilizado no projeto, chega a 12Mbit/s, conhecida como *full-speed*, que é bastante superior comparado com a velocidade dos outros. No entanto, para atingir tal velocidade é necessário o desenvolvimento de um *driver* específico para a aplicação na qual será usado. Isto requer um conhecimento bastante elevado da comunicação USB para algo que não é tão primordial para o projeto.

A conexão USB neste projeto será feita utilizando o protocolo CDC, por conta da simplicidade de implementação do *software*, já que depois da instalação do *driver*

não há muita vantagem em se utilizar o protocolo HID. Assim, a comunicação USB será feita através de uma porta serial emulada pelo microcontrolador.

A velocidade da CDC é bem maior que a do HID, pois ela trabalha com transferências em *bulk*, como o MSC, mas fica abaixo da velocidade deste, pois possui muitos dados extras para garantir uma conexão efetiva. Nos testes feitos, pode-se observar que o protocolo CDC consegue entregar uma velocidade acima de 1Mbit/s, o que é suficiente para esta plataforma.

4.1.3 Especificaçoes dos Conversores a serem Testados

No mercado, existem diversos conversores, com diversas especificações diferentes. Elaborar uma plataforma de testes que consiga medir parâmetros para todos os conversores é quase que impossível. Portanto, devem ser feitas escolhas para que se possa testar a maior quantidade de conversores possíveis, além de focar principalmente nos que são feitos pelos laboratórios do PEE.

A primeira característica escolhida foi a de testar conversores com a saída paralela. Assim, deve-se destinar um pino de I/O para cada *bit* na saída do conversor. Como há um número limite de pinos, deve haver um número máximo de *bits* nos dispositivos a serem testados, que foi definido como 16 *bits*. Assim, foram destinados 16 pinos de I/O para a aquisição dos *bits* da saída do ADC.

Outra limitação que será imposta é na taxa de conversão do dispositivo em teste. Como a comunicação USB consegue transmitir acima de 1Mbit/s, iremos limitar o *throughput* dos conversores nesta velocidade.

No teste de conversores de alta velocidade, o mais difícil de ser feito é justamente a aquisição de dados tão rápida quanto o *throughput* do ADC. Esta tarefa geralmente é realizada por um equipamento chamado Analisador Lógico, que é como um grande osciloscópio, porém ele analisa apenas valores digitais. Seu grande ponto forte é a quantidade de *bits* que consegue adquirir em altíssima velocidade, o que torna este equipamento extremamente caro [2].

4.1.4 Gerador de Sinais

Um componente importante para que os testes sejam feitos com bastante precisão é utilizar um gerador de sinais de qualidade para produzir as formas de onda dos testes. Como optamos por métodos de testes que utilizam senóides, que são ondas bastante simples de serem geradas com boa precisão, a escolha deste equipamento não precisa ser tão restrita.

Como o laboratório possui um gerador de forma arbitrária de alta qualidade, o AGF3552 da Tektronix, iremos utilizá-lo. Este equipamento possui uma característica muito boa, que é uma conexão USB com o computador, para ser controlado por um protocolo específico de equipamentos de bancada, chamado de VISA (Virtual Instrument Software Arquitecture), o qual o MATLAB consegue utilizar mediante à instalação de um *plug-in* disponibilizado de graça pela página da Tektronix.

Esse controle remoto do equipamento nos permite selecionar frequências e amplitudes para as senóides de forma bastante precisa, e sem interferência do usuário, reduzindo a possibilidade de erros nas medidas feitas pela plataforma.

4.2 Sistema de Testes da Plataforma

Com base nas escolhas de projeto, um diagrama de blocos pode ser esboçado para representar todo o sistema da plataforma. Este diagrama pode ser visto na figura 4.1.

Como pode ser observado, todos os elementos em discussão estão presentes na figura. Primeiro, observa-se o computador conectando tanto ao gerador, quanto ao Circuito de Aquisição pela USB, cada um com seu protocolo, VISA e CDC, respectivamente. O PC envia informações ao gerador, para configurar a senóide de entrada no dispositivo em teste (DUT, da sigla em inglês) e também ao Circuito de Aquisição, que irá controlar a quantidade de amostras a serem analisadas.

O Circuito de Aquisição também deve receber um sinal, que indique quando a aquisição deve ser executada, ou seja, quando a saída do ADC deve ser lida. Esta fase de leitura é enviada pelo controle do DUT, que pode ser interno ou externo

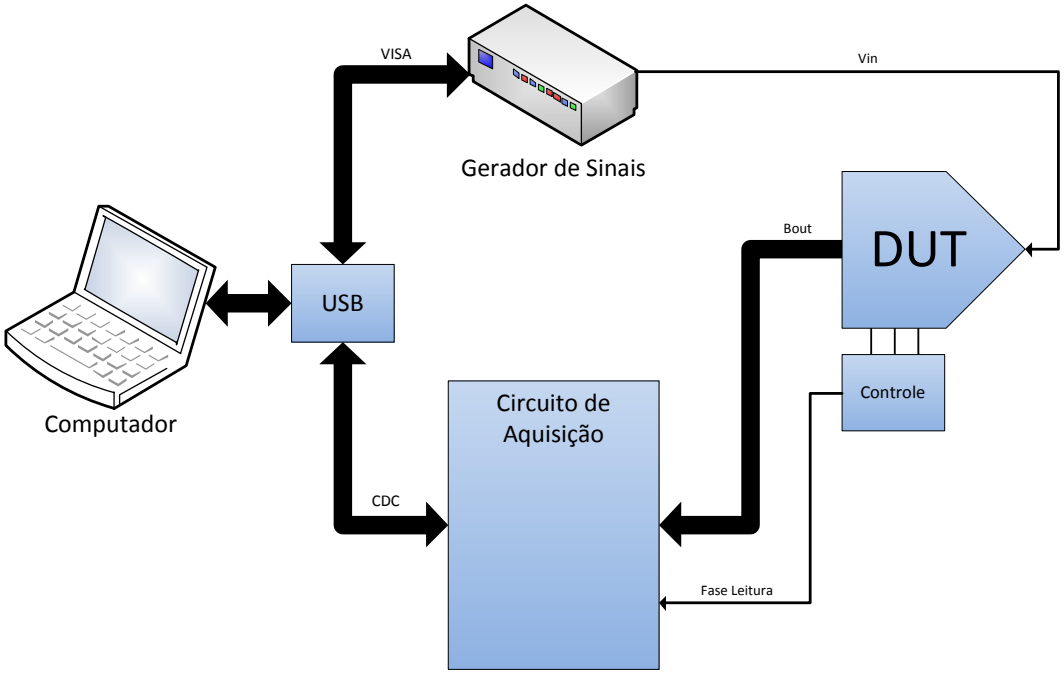


Figura 4.1: Diagrama de Blocos da Plataforma de Testes.

ao conversor, que para a plataforma é indiferente. Após a aquisição da amostra, o circuito envia estes resultados em tempo real ao computador, que irá salvar e analisar os dados.

4.3 Circuito de Aquisição

Para finalizar o sistema de testes, deve-se especificar o Circuito de Aquisição que será utilizado. Este circuito é basicamente o PIC18F4550 com alguns elementos externos especificados no *datasheet*. O circuito de aquisição ficou estabelecido como na Figura 4.2. Os valores Bn representam o *bit* de aquisição n.

O circuito é bastante simples, e gira em torno do PIC. Primeiramente temos as portas binárias de entrada (saídas do ADC) que, apesar de distribuídas entre os pinos, foram escolhidas por ocuparem os mesmos registradores, o que tornaria o código mais veloz. Os *bits* B0-B7 ficarão armazenados no registrador **PORTD** do PIC e os B8-B15 no **PORTB**. Há também um pino para receber a fase de leitura, que foi designado para o pino 3 (A1).

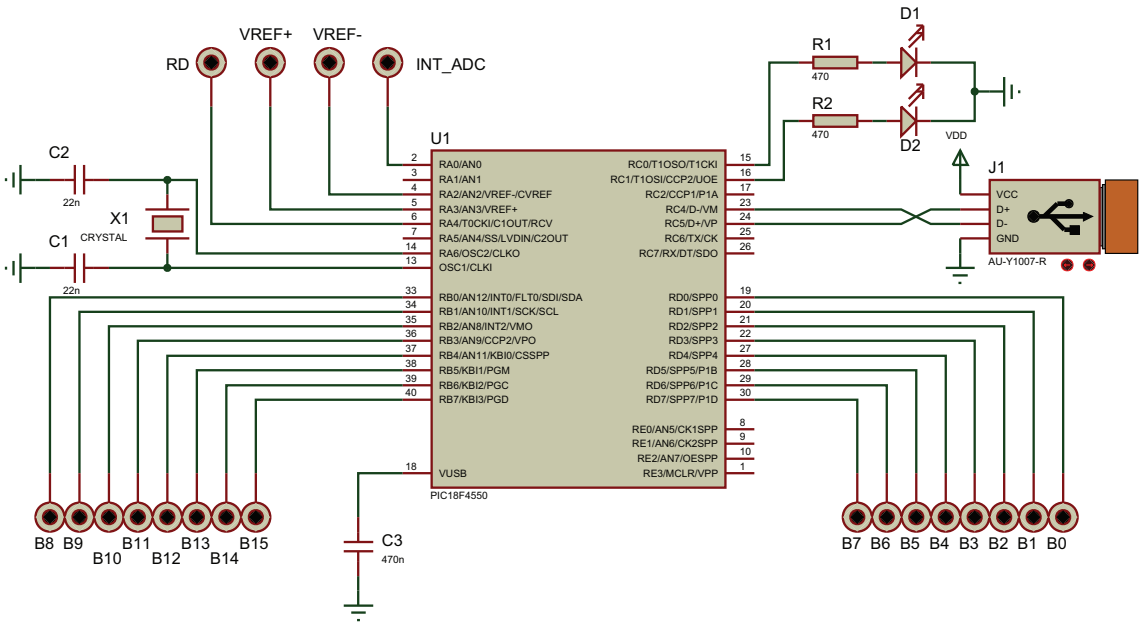


Figura 4.2: Esquemático do Circuito de Aquisição.

Para que o PIC funcione, ele deve ser alimentado por alguma fonte. Na plataforma, esta alimentação é fornecida pela porta USB, cuja voltagem é de aproximadamente 5 V. Porém, o protocolo de transmissão USB utiliza o 0 V como nível lógico baixo e 3.3 V como alto, que é uma voltagem diferente da alimentação. Para contornar este problema, o PIC tem um regulador interno de 3.3 V que é usado na transmissão USB.

Como o PIC utiliza 5 V de alimentação, isto acaba impondo uma limitação na voltagem de nível lógico alto da saída do dispostivo em teste. Segundo o *Datasheet* em [8], o MCU só reconhece como nível lógico alto voltagens maiores que 2 V. Abaixo disso, é necessário algum dispositivo para elevar esta voltagem de saída do DUT.

O cristal no circuito é de 20 MHz, e está conectado aos pinos 13 e 14, reservados para o oscilador externo. Ele é necessário para que o *clock* da USB seja o mais preciso possível. Internamente, é configurada uma PLL para transformar os 20 MHz em 48, que é o requisito de *clock* do barramento USB. C1 e C2 são dois capacitores ligados ao terra que compõem a malha de defasagem junto com o cristal. O capacitor C3 é uma recomendação para que a conexão USB seja mais estável também [8].

A porta USB deve conectar-se aos pinos 23 e 24, separados especialmente para esta interface. Esta porta é do tipo B. Foram colocados também dois LEDs para

que o usuário possa ver o *status* do circuito. Quando o LED vermelho (D2) acende, não há nenhum teste sendo executado, mas há alimetação ao circuito. Já o LED verde (D1) significa que há um teste sendo executado.

Reservamos também o pino 2, que é um canal para o ADC interno do PIC. Com isso, podemos utilizar este conversor interno para testar o *software* que foi feito de forma mais rápida. Neste conversor interno, é possível configurar uma fonte externa para ser a referência do ADC. Esta voltagem deve ser localizada nos pinos 4 e 5. Ao utilizar o ADC interno, a aquisição é muito menos suscetível a erros de interfaceamento. Como seu propósito é para debugar o circuito, ela não será configurável, ou seja, suas especificações serão instaladas na *firmware* do PIC.

Apesar de já estar esquematizado, o circuito não possui funcionalidade nenhuma ainda, pois para isso é necessária a criação do *firmware* do microcontrolador, ou seja, do programa interno que diz ao PIC que passos seguir, e em qual ordem. Sem este programa, o circuito não faz aquisição alguma.

4.3.1 Placa de Circuito Impresso

Baseado no esquemático da Figura 4.2, podemos projetar um *layout* de placa de circuito impresso para o Circuito de Aquisição da plataforma. Para o desenho da placa, foi utilizado o *software* Cadsoft Eagle, por ser um *software* com biblioteca bastante extensa e possuirmos maior prática utilizando-o. O resultado final é exibido na Figura 4.3.

O circuito elaborado ficou bem compacto, utilizando apenas um *jumper*. Vale destacar a malha de terra, dando uma captação maior de ruídos para que haja menor interferência nas medidas. No *design* do circuito, só utilizamos a camada de baixo da placa. Com isso, pôde-se utilizar uma produção artesanal do circuito, ao invés de encomendá-lo em uma fábrica.

O resultado final da placa de circuito impresso, após todo o processo de produção artesanal completo e toda a solda dos materiais, é mostrado na Figura 4.4. Podemos observar alguns pinos de entrada no topo da placa. Estes pinos facilitam a conexão

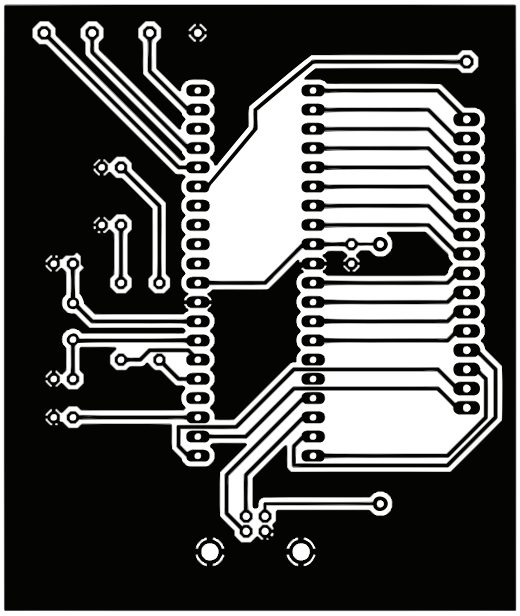


Figura 4.3: Layout do Circuito de Aquisição.

de plugues tipo Jacaré, que são os utilizados no laboratório. Do lado direito da placa tem-se o conector de entrada paralelo. Foi feita uma pequena placa para facilitar o interfaceamento, onde na parte inferior estão localizados pinos de engate com o espaçamento correto para uma *protoboard*. Os pinos se encontram em ordem de LSB para MSB, de baixo para cima. O fio em marrom, relativamente separado do conector é o responsável pelo interfaceamento da fase de leitura. Na parte inferior da placa localiza-se o conector USB fêmea tipo B. À direita do PIC podemos notar os componentes periféricos, como os LEDs e o cristal de 20 MHz.

4.4 Sistema de Testes Completo

Com a placa do circuito de aquisição pronta, todas as peças da plataforma podem ser conectadas como esquematizado na Figura 4.1. O resultado pode ser visto na Figura 4.5. Na figura, estava sendo testado o conversor interno do PIC de aquisição. Portanto, não há nenhum conversor conectado no barramento paralelo.

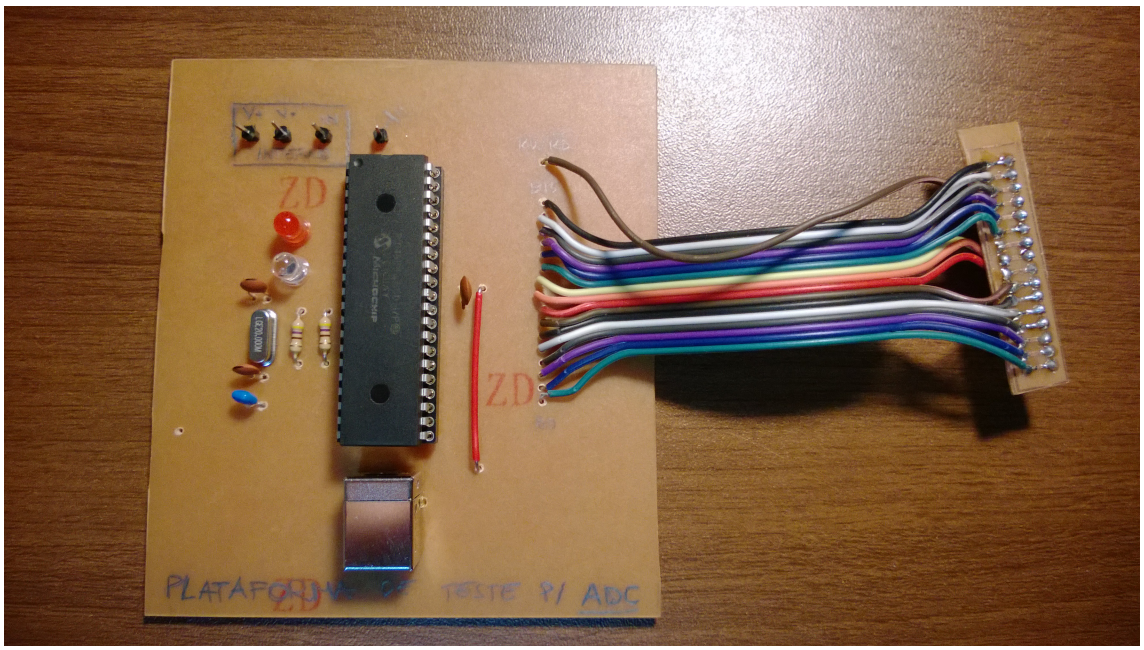


Figura 4.4: Placa de Circuito Impresso Finalizada.

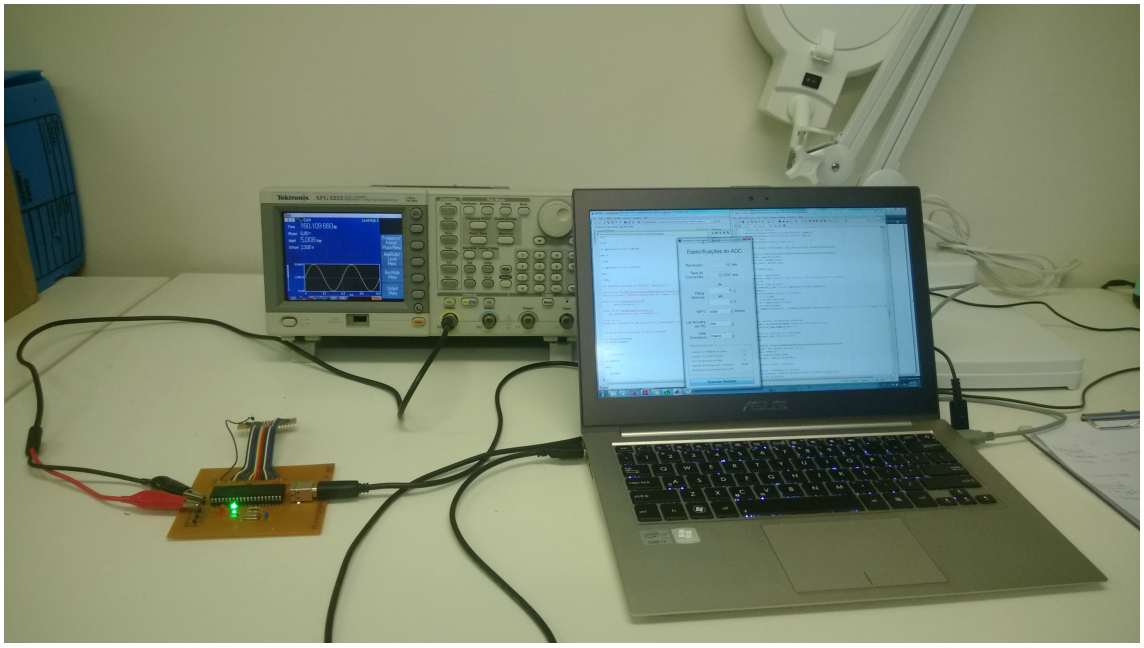


Figura 4.5: Sistema de Testes completo.

Capítulo 5

Software da Plataforma de Teste

O *software* da plataforma de testes é o centro de todo o projeto. É este *software* que, além de direcionar ao microcontrolador PIC o que fazer, através do *firmware*, irá executar a interface USB com o MCU, adquirir as amostras em tempo real, e depois analisá-las de forma eficiente. Após a análise, o *software* deve salvar os parâmetros calculados para que possam ser usados para publicações científicas.

O *software* foi dividido em três partes: O *firmware* do PIC, o Programa de Aquisição de Dados, e o Programa de Análise de Resultados. Optamos por separar o Programa de Aquisição do de Análise pois assim é possível fazer a análise das mesmas amostras mais de uma vez. Um diagrama de blocos pode ser esboçado, mostrando como todas as peças deste *software* se encaixam. Este diagrama é mostrado na Figura 5.1. Podemos observar que o programa do PIC e o *software* de Aquisição trabalham simultaneamente, e os dados são transmitidos em tempo real.

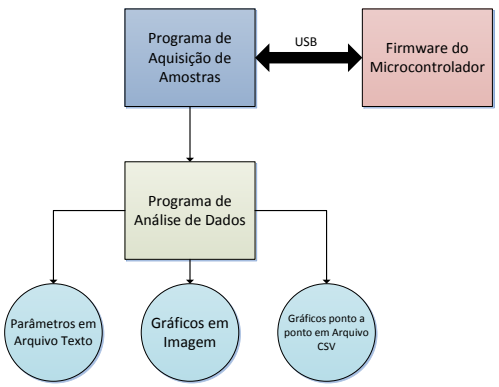


Figura 5.1: Diagrama de Blocos do *Software*

Pretendemos mostrar neste capítulo a escolha da linguagem de programação para cada *software*, além do funcionamento de cada parte em detalhes. Serão exibidos fluxogramas de funcionamento e imagens das interfaces gráficas.

5.1 Linguagens e Ferramentas de Programação

As ferramentas de programação que foram definidas para este projeto tiveram como critério de escolha a facilidade de programação. O microcontrolador PIC é programado, por padrão, em linguagem C, porém existem diferentes compiladores que podem ser usados para programá-lo. Optamos pelo compilador da CCS, que possui diversas funções e bibliotecas já implementadas para o usuário. Isto facilita muito tarefas complicadas, como implementar uma conexão USB entre o PIC e o computador.

A ferramenta que foi escolhida para os outros dois programas foi o MATLAB. Ele é um programa muito utilizado pelo autor do projeto, portanto, seria mais simples de fazer um *software* para executar as tarefas desejadas. O MATLAB é convencionalemte uma ferramenta que executa os programas através da interpretação de *scripts*, em formato '.m'. Isto torna mais complicado uma reutilização do programa por outros usuários, já que a entrada de informações feita pelo usuário é feita diretamente no *script*, o que necessitaria uma leitura e entendimento do código-fonte do programa, que é uma tarefa bastante complexa.

Para contornar este problema, o MATLAB possui um módulo que facilita muito a criação de interface gráfica. Este módulo pode ser acessado através do comando *guide*, sigla em inglês para Ambiente de Desenvolvimento de Interface Gráfica do Usuário. Com este módulo, conseguimos incluir tudo que é desejado para a plataforma utilizando o MATLAB. Além disso, podemos utilizar caixas de texto e botões, que tornam a entrada de dados muito mais intuitiva.

5.2 *Firmware* do Microcontrolador

O *firmware* do microcontrolador é um programa que comanda tudo que é feito pelo PIC. Ele irá ler e interpretar os parâmetros de teste enviados pelo usuário, bem como fazer a aquisição dos dados do conversor que está sendo testado. Após a leitura desses dados, o MCU irá enviá-los para o computador, que irá interpretá-los.

O algoritmo implementado no *firmware* é ilustrado na Figura 5.2. Iremos descrever os detalhes desta implementação.

O programa se inicia no exato momento em que a plataforma de testes é alimentada por uma fonte. Neste caso, a fonte é a própria porta USB, que possui um módulo de alimentação de 5 V. Ao conectar a plataforma ao computador, através de sua porta USB, inicia-se a configuração do PIC. Esta etapa é importante, pois indica ao Microcontrolador quais são os seus blocos internos, chamados de periféricos, que irão ser utilizados pela plataforma. Além disso, os configuramos para funcionar do modo desejado, como, por exemplo, dizer quais portas são entradas e quais são saídas.

Feito isso, é iniciada, de fato, a conexão USB. No momento anterior, só foi usada a alimentação da porta, e não a comunicação. O primeiro passo é aguardar a enumeração da plataforma pelo computador, ou seja, aguardar a identificação do computador que se conectou à plataforma de testes. Isto é feito pela leitura de um identificador, um número no protocolo CDC. Este número associa a plataforma ao seu driver. Com a enumeração pronta, já pode ser estabelecida a conexão com o Programa de Aquisição. Enquanto não há conexão, não há aquisição de dados por parte da Plataforma.

Com a conexão já estabelecida, o LED verde se acende, indicando que os testes foram iniciados. O Programa de Aquisição então envia alguns parâmetros que foram calculados, e são pertinentes para o comportamento do PIC. Os valores R, M, e N_{FFT} indicam quantas amostras serão adquiridas e o parâmetro N quantos bits serão enviados pela porta. O mais importante deles, no entanto, é o indicador de qual conversor será utilizado, um conversor externo ou o conversor interno do PIC.

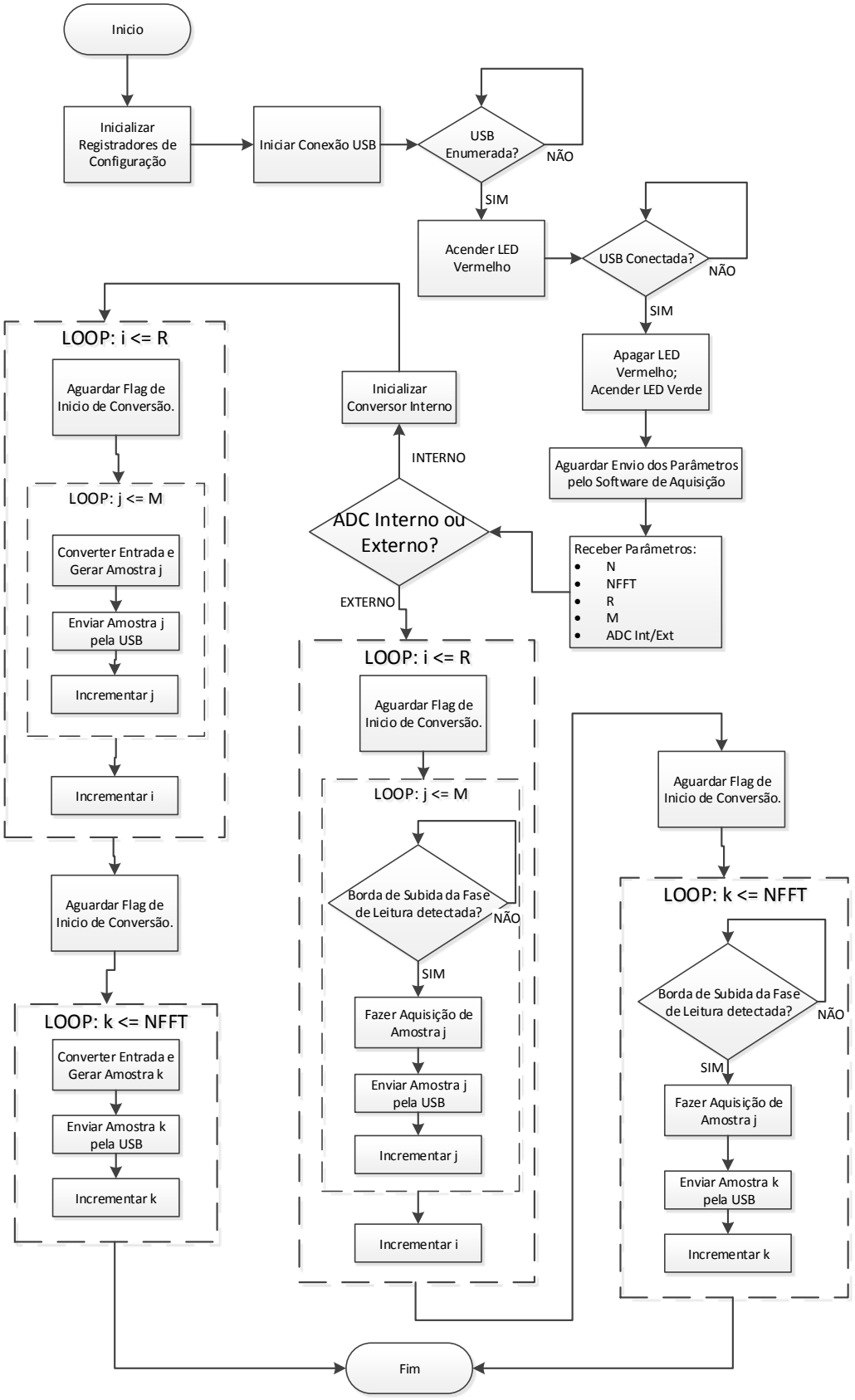


Figura 5.2: Fluxograma do Firmware do Microcontrolador

A partir desta informação, o *firmware* indica ao PIC qual é a porta que deve ser utilizada para a leitura das amostras.

Caso o conversor interno seja escolhido para o teste, devemos configurar o conversor para que funcione no pino certo. As voltagens de referência também foram estabelecidas externamente ao PIC, e passadas ao ADC por dois pinos externos, e não as próprias referências do PIC. Após configurado, podemos iniciar a conversão e aquisição de amostras. Para que a frequência de conversão seja constante, utilizamos um dos *timers* do MCU para indicar quando iniciar a conversão. Utilizamos as rotinas de interrupção para deixar o código mais elegante e eficiente. Cada amostra que é convertida é enviada diretamente após sua aquisição. Isto é feito por diversas vezes até que são adquiridas todas as amostras desejadas.

Se a escolha do usuário for por testar um conversor externo, então o PIC utiliza a porta da fase de leitura. O programa detecta a borda de subida da fase de leitura do conversor, que é a indicação de que a aquisição deve ser feita. Com a amostra adquirida, ela é enviada para o computador pela USB. Este processo é repetido para todas as amostras desejadas.

Podemos reparar que tanto para o conversor interno, quanto para o externo, foi colocado um flag de inicio de conversão, que tem como função manter sincronizado o *firmware* e o Programa de Aquisição. Caso um seja capaz de ler mais rápido que o outro consegue enviar, isto possibilita o esvaziamento do *buffer* de transmissão, evitando uma possível perda de dados.

Com o *firmware* pronto, toda a interface física já está acertada, faltando apenas a parte de software no MATLAB, que será o responsável por dar sentido aos dados obtidos pela plataforma.

5.3 Programa de Aquisição de Amostras

O programa de Aquisição de Amostras é o responsável, como o nome propriamente diz, pela Aquisição de Amostras do Conversor A/D em teste. Ele é o responsável por conectar-se ao circuito de aquisição através da porta USB, obter as amostras que

estão sendo geradas pelo ADC e as salvá-las em um arquivo que será posteriormente aberto pelo programa de Análise de Dados. Ele também se conecta ao gerador de sinais e envia os parâmetros de configuração da senóide de entrada. É neste *software* que o usuário também insere os dados referentes ao conversor que será testado, e há uma verificação para saber se é possível fazer o teste do dispositivo desejado.

A interface gráfica é mostrada na Figura 5.3. Ela possui alguns campos de entrada de dados e um botão principal, que inicia as medidas. Os dados que são pedidos pela interface são informações vitais, que indicam à plataforma como prosseguir no teste. A interface também possui alguns parâmetros que facilitam as medidas, como indicar se a leitura das amostras deve ser no nível lógico alto ou baixo do sinal de leitura. O painel de *Status* de Aquisição indica como está sendo o progresso do teste, para que o usuário não fique esperando sem saber o que está acontecendo.

O algoritmo executado por este *software* é ilustrado pelo fluxograma na Figura 5.4. O fluxograma indica cada etapa do processo sem muitos detalhes, os quais iremos descrever.

O programa inicia ao se abrir o arquivo "PT_ADC_Analysis.fig"na pasta de arquivos do projeto. Como foi dito, abre-se uma interface muito simples, para que o usuário digite as especificações do conversor. É importante que as informações sejam corretas, pois irão influenciar no cálculo dos parâmetros de teste. Nesta janela gráfica, é possível escolher se será medido um conversor externo ao PIC de aquisição ou o conversor interno deste PIC.

Ao selecionar o ADC interno, as configurações são automaticamente modificadas para se adequarem ao conversor interno do PIC. A velocidade de conversão utilizando nosso *firmware* foi medida experimentalmente, com valor de 29.0865kHz. Sua resolução é de 10 bits. A partir desta modificação, não é possivel alterar estes dados, a não ser que o usuário mude a seleção de volta para o conversor externo. Isto nos permite uma verificação de *software* mais veloz e precisa. Como configuramos a voltagem de referência para a conversão como sendo indicada externamente, o usuário pode alterar este parâmetro.

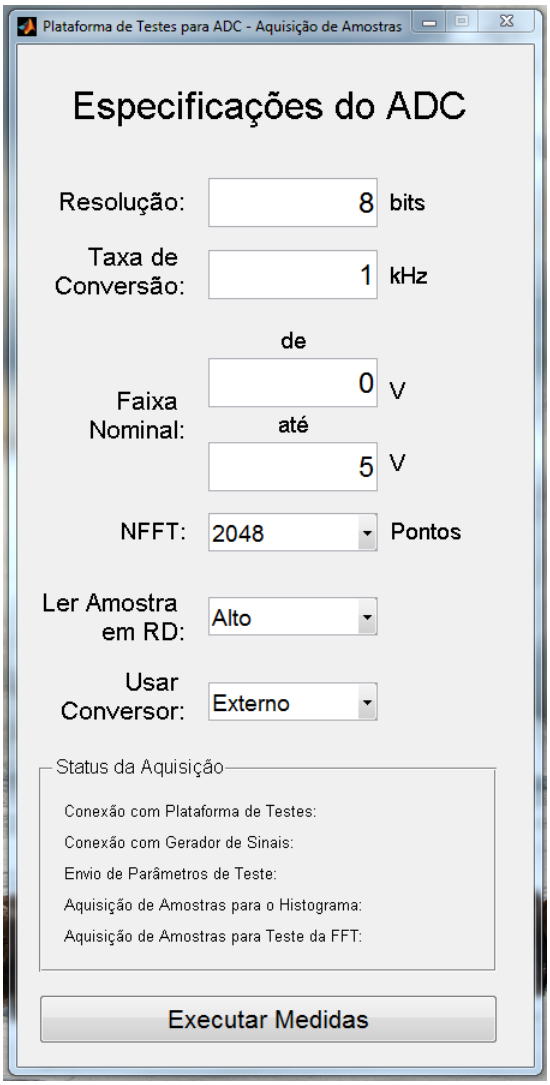


Figura 5.3: Interface Gráfica do Programa de Aquisição de Amostras.

Após inserir todos os dados em relação ao conversor, o usuário inicia a aquisição de amostras, ao pressionar o botão "Executar Medidas". A primeira coisa que o programa faz é observar se as configurações enviadas pelo usuário respeitam os limites de operação da Plataforma de Testes, que são:

Resolução: de 1 a 16 bits,

Frequência de Conversão: até 60kHz,

Faixa Nominal: de -10V até 10V.

Como os outros parâmetros são selecionáveis por uma *dropbox*, eles sempre serão válidos. A limitação em resolução ocorre por conta das portas do PIC, que são

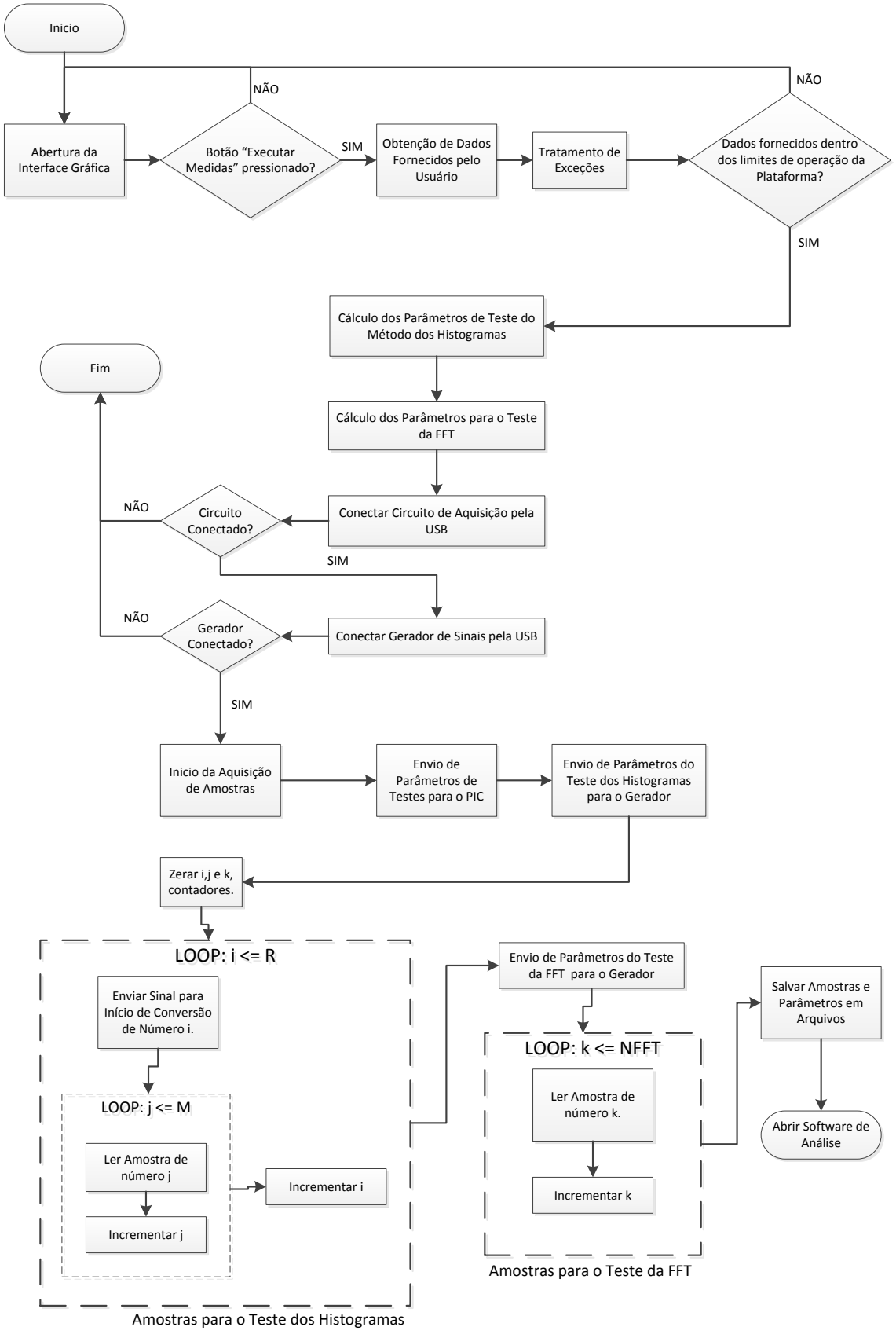


Figura 5.4: Fluxograma do Programa de Aquisição de Amostras

apenas 16 destinadas para a aquisição. A limitação na frequência de conversão ocorre por conta da comunicação USB, que ficaria instável se trabalhássemos acima desta frequência. Já a da faixa nominal fica por conta do Gerador de Sinais, que consegue apenas gerar ondas de -10V até 10V.

Caso os dados fornecidos estejam nos conformes, é iniciado o cálculo dos parâmetros de teste, tanto para o método dos histogramas, quanto para o teste da FFT. O processo para os parâmetros do método dos histogramas foi demonstrado na seção 3.4. Escolhemos uma Tolerância de 0.1 e uma Incerteza de 1% em relação a DNL para os cálculos. Superestimamos o nível de ruído do sistema, que colocamos em 1 mV. A precisão na frequência de entrada é dada pelo gerador de sinais, cujo valor é de 1 μHz [10]. Juntando estas especificações com os dados de entrada do usuário, todos os parâmetros são calculados, dentre eles

R = número de conversões à serem feitas,

M = número de amostras por conversão.

Estes parâmetros são os utilizados nos *loops* da Figura 5.6.

Os parâmetros do teste da FFT são bem mais simples, como mostramos na secão 3.5. Só há o cálculo da frequência de teste, já que o número de amostras é escolhido pelo usuário.

A próxima etapa consiste na conexão do programa com os equipamentos periféricos, o Circuito de Aquisição e o Gerador de Sinais. O Circuito de Aquisição é conectado ao MATLAB utilizando um objeto serial. Ao criar-se o objeto, é tentada a conexão, que caso não funcione acarreta no fechamento do programa. Uma mensagem de alerta aparecerá na tela do usuário neste caso. Para a conexão com o Gerador de Sinais, é criado um objeto VISA no MATLAB. Para isto, é necessária a instalação de um *plug-in* fornecido pela Textronix gratuitamente. Esta instalação é demonstrada no manual de uso da plataforma, no Apêndice A. Caso a conexão não seja feita, o programa também é automaticamente fechado e uma mensagem de alerta aparece ao usuário.

Com a conexão feita, os parâmetros de testes são enviados para o PIC e para o Gerador de Sinais, e a aquisição é iniciada.

As amostras para o método dos histogramas são adquiridas primeiro, e cada inicio de conversão é indicado por um *flag* enviado ao PIC. É feito desta forma para que não haja uma sobrecarga do *buffer* de transmissão USB. Assim, a cada final de conversão só é iniciada outra quando o buffer de transmissão é esvaziado. Em cada conversão são adquiridas M amostras, e são feitas R conversões, obtendo um total de $R \cdot M$ amostras.

Após esta aquisição, são obtidas as amostras para o teste da FFT. O Gerador de Sinais recebe outros parâmetros, já que os sinais de entrada são diferentes para cada teste. São então adquiridas NFFT amostras, valor muito menor que o do teste dos histogramas.

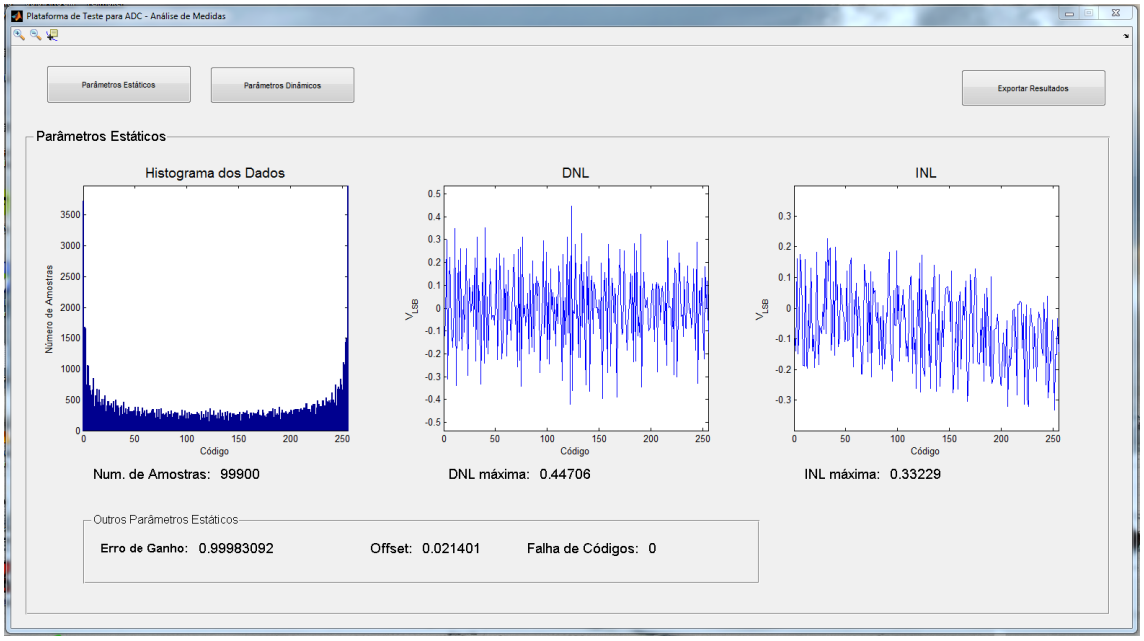
Ao fim, as amostras são exportadas em um arquivo estilo ".csv", com os valores separados por vírgula. Com os arquivos salvos, o programa de Aquisição de Dados chama a abertura do programa de Análise de Dados.

5.4 Programa de Análise de Dados

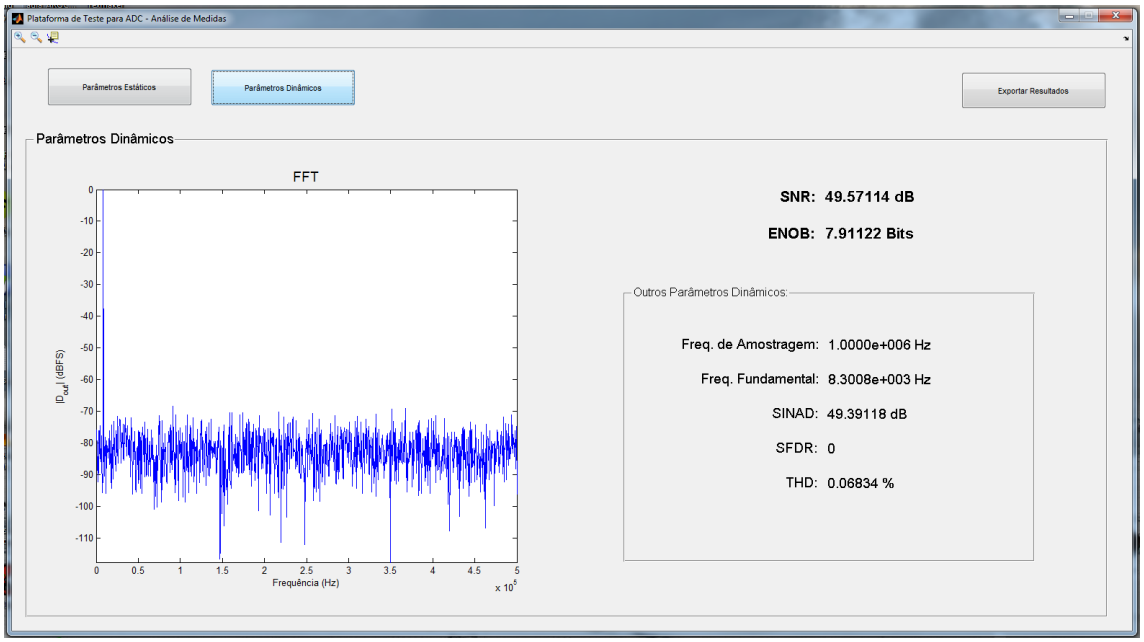
O programa de Análise de Dados é automaticamente aberto após a Aquisição de Amostras ser finalizada. Ele lê os arquivos gerados pelo programa anterior e obtém o cálculo de todos os parâmetros estáticos e dinâmicos, que foram mostrados no Capítulo 3. Após os cálculos, todos estas figuras de mérito são exibidas de uma forma bastante conveniente. Optamos por fazer dois programas separados pois deste modo pode-se analisar os dados novamente sem ter que executar uma nova aquisição.

A interface gráfica deste *software* é mostrada na Figura 5.5. Podemos notar que é uma interface pouco poluída visualmente, e entrega todos os dados de maneira muito simples. Os parâmetros dinâmicos e estáticos foram separados em uma espécie de aba improvisada, facilitando a visualização dos dados agrupados por tipos. Além dos parâmetros numéricos, também são demonstrados os gráficos da DNL e INL. A FFT também é demonstrada, em dBfs (decibéis de escala cheia). As abas improvisadas

são acionadas pelos botões "Parâmetros Estáticos"e "Parâmetros Dinâmicos", o que deixa a navegação muito fácil. Além disso, há o botão "Exportar Resultados", que gera um arquivo texto com os parâmetros numéricos e figuras ".eps"com os gráficos. Os gráficos também serão exportados em pontos numéricos.



(a)



(b)

Figura 5.5: Interface Gráfica do Programa de Análise de Dados. (a) Exibindo os Parâmetros Estáticos. (b) Exibindo os Parâmetros Dinâmicos.

Todo o algoritmo executado por este *software* é ilustrado pelo fluxograma da Fi-

gura 5.6. Como nos fluxogramas anteriores, ele ilustra o que cada passo do programa faz por alto, mas iremos descrever cada etapa em detalhes.

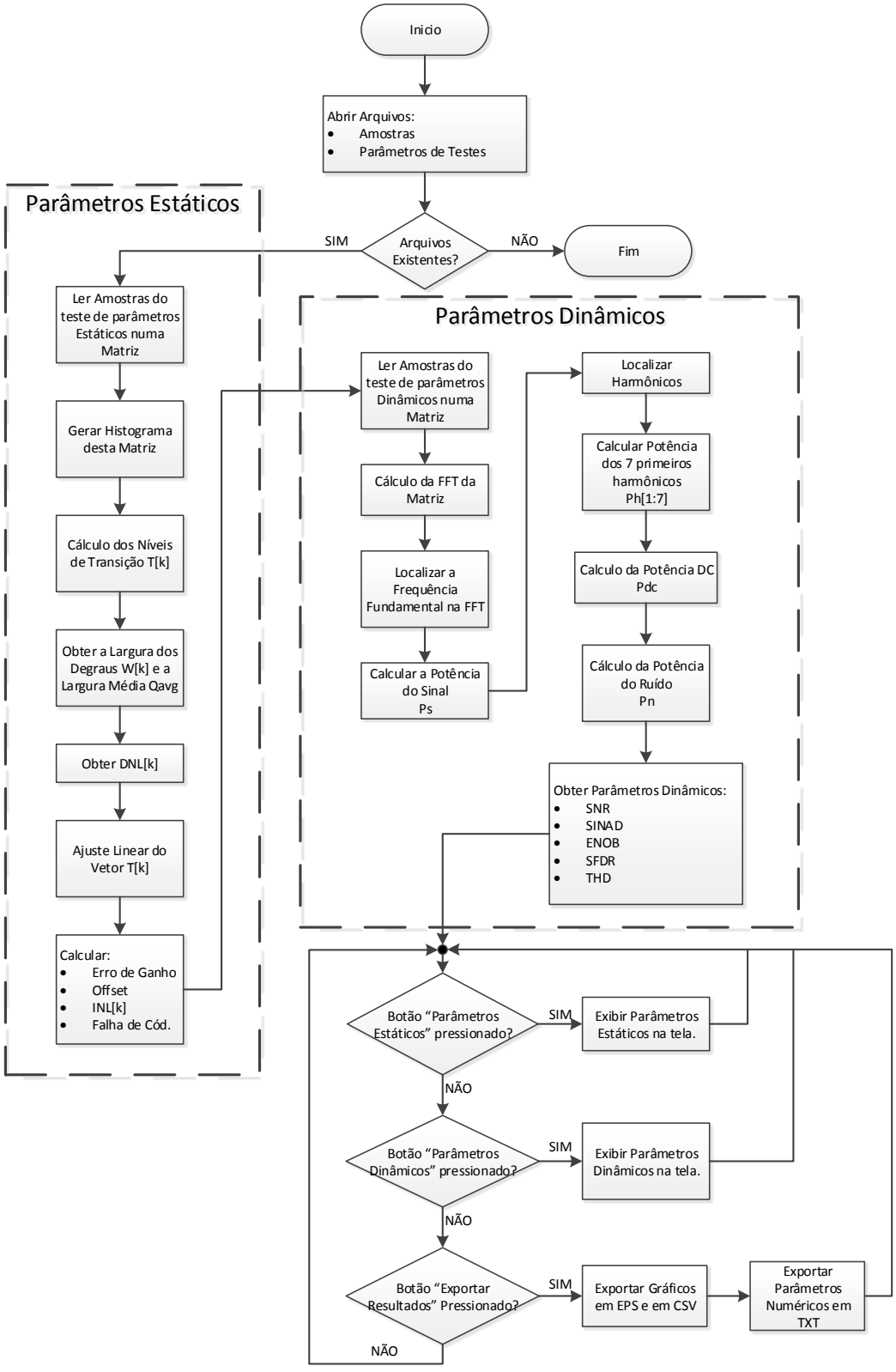


Figura 5.6: Fluxograma do Programa de Análise de Dados.

A janela de interface gráfica do MATLAB abre com uma função de inicio, onde se carregam todos os parâmetros. É nesta função que se calculam todos os parâmetros e gráficos. Ao se iniciar, a função procura pelos arquivos gerados pelo programa de Aquisição de Dados. Se os arquivos não estiverem funcionando, o programa fecha sozinho, e uma mensagem de alerta é enviada ao usuário. Porém, como na maioria das vezes o próprio programa de aquisição invoca a abertura da Análise de Dados, isto é pouco provável de acontecer. Segurança nunca é demais.

O programa, então, começa pelos parâmetros estáticos. Ele abre o arquivo com as amostras e as lê direto num *array*. Este *array* é, então, distribuído em um histograma, gerando uma classe para cada saída binária. A partir deste histograma, podemos fazer o cálculo dos níveis de transição, o vetor $T[k]$, como foi mostrado na seção 3.4. Com este vetor, já é possível calcular todas as figuras de mérito estáticas.

O processo utilizado segue a ordem do fluxograma. Primeiro calcula-se o tamanho dos degraus $W[k]$ e logo o tamanho médio dos degraus Q_{avg}. Com isso, já temos todos os dados para o cálculo do vetor $DNL[k]$. O MATLAB possui um comando que nos entrega o valor máximo desta DNL. Utilizando o vetor $T[k]$, obtemos o erro de ganho G, o *offset* V_{off} e a $INL[k]$, utilizando a equações apresentadas na seção 3.2.4. Também retiramos o número de falhas de códigos, observando o tamanho do degrau dos códigos igual a zero. Assim, se finaliza o cálculo e o arquivo é fechado, iniciando com os parâmetros dinâmicos.

As figuras de mérito dinâmicas seguem o mesmo processo. Abre-se o arquivo e a leitura de todos os pontos é feita para um *array*. É feita, então, a FFT deste *array*, gerando a composição em frequência do sinal. Dela, retiramos só a primeira metade que corresponde à banda entre DC e a metade da frequência de amostragem. Então, procuramos o ponto com maior amplitude (excluindo o DC) que corresponde ao sinal de entrada. Retiramos, então, a potência desta componente, a potência do sinal P_s. Sabendo a localização da fundamental, podemos obter os harmônicos, gerando um vetor com a potência de cada harmônico. A potência DC é mais simples, pois se localiza na primeira componente. Já a potência do ruído é a potência total da FFT

subtraída dos outros valores, P_s, P_h e P_{DC}.

Com todas as potências separadas, podemos encontrar todas as figuras de mérito descritas na seção 3.3, que são: SNR, SINAD, ENOB, THD e SFDR. Também colocamos a FFT em dBfs, deslocando a maior componente (a fundamental) para 0 dB, demonstrando o nível de ruído do circuito. Para o código que retira os dados da FFT, algumas idéias foram inspiradas em [11].

Após obter todos os parâmetros e gráficos, os mesmos são exibidos na interface gráfica do programa. Caso o usuário esteja satisfeito com a medição, ele pode exportar os resultados utilizando o botão "Exportar Resultados", no canto superior direito da tela. Ao clicar no botão, o programa pede que o usuário entre com um identificador para a medida, que será o nome da pasta aberta com os resultados. Nesta pasta serão exportados os gráficos da DNL, INL, FFT e o Histograma em arquivos EPS, que são o formato padrão para publicações científicas. Além disso, o gráfico exportado possui legendas em inglês, que também é o padrão de artigos. Os gráficos também são exportados em um arquivo CSV, com os valores de ponto-a-ponto dos gráficos. Por fim, os parâmetros numéricos que foram calculados também serão exportados em um arquivo texto, uma espécie de relatório, com todos os parâmetros obtidos nas medidas.

Capítulo 6

Resultados Experimentais

Com a plataforma pronta para o uso, é muito importante testá-la para averiguarmos se a mesma está funcionando corretamente. Para isto, iremos testar conversores Analógico-Digitais já existentes no mercado, que possuam alguma informação do fabricante referente aos parâmetros de medida que estão sendo testados.

Realizaremos três testes: num primeiro momento iremos testar o conversor interno do PIC do circuito de aquisição da plataforma de testes. Depois testaremos um PIC externo com duas frequências de conversão diferentes, para observarmos as diferenças nos parâmetros obtidos.

Neste capítulo, iremos apresentar todas as figuras de mérito que estão sendo testadas pela plataforma, bem como tabelas comparativas entre os resultados obtidos e os dados do fabricante. Desta forma, pretendemos comprovar o funcionamento da plataforma.

6.1 Conversor Interno do PIC18F4550

O primeiro conversor que escolhemos para teste foi o conversor interno do microcontrolador do circuito de aquisição. É uma escolha natural, pois o interfaceamento é bem mais simples, e nos ajuda a testar se os programas de Aquisição de Amostras e de Análise de Dados estão funcionando corretamente. Após todo o aparato de testes estar funcionando corretamente, pudemos iniciar as medidas.

Para demonstrar que as medidas da plataforma são confiáveis, procuramos demonstrar a repetibilidade dos resultados. Como sabemos, o teste dos histogramas é estatístico e, portanto, possui um erro atrelado à medida. Podemos deixar este erro tão pequeno quanto desejado, especificando no cálculo de parâmetros para a realização do teste, como foi explicado no capítulo 3.4. Para isso, foram feitas cinco medições diferentes do conversor interno, utilizando-se os mesmos parâmetros de teste. Apesar de cinco ser um número pequeno, uma variação pequena nos resultados destas amostras é um bom indicador de que estão sendo gerados resultados satisfatórios. Com os resultados obtidos, podemos traçar os gráficos e encontrar os valores médios destas medições e, assim, observar se os valores obtidos nas cinco medidas extrapolaram este erro estatístico.

Para iniciarmos o teste, precisávamos obter alguns parâmetros de teste que deveriam ser incluídos no Programa de Aquisição de Amostras. O número de *bits* do conversor interno, que apesar de chegar a 10, foi limitado em 8 para que não houvesse uma alteração na frequência de conversão. Isto estava acontecendo devido à latência da conexão, que era o dobro para enviar 10 *bits*, já que, para isso, deve-se enviar 2 *bytes*, contra apenas 1 *byte* do conversor de 8 *bits*. A frequência de conversão foi, então, fixada em 4.8943 kHz, valor obtido experimentalmente, através de um pequeno pulso colocado em uma saída do circuito toda vez que se iniciava uma conversão. O último valor foi a faixa nominal do conversor, de 0 V (V_{ref-}) a 3.3 V (V_{ref+}) indicada por pinos externos.

Como o objetivo de se testar o conversor interno é puramente de verificação da plataforma, e não para obter as informações do conversor, não se teve a necessidade de explorar o desempenho do Conversor A/D interno ao máximo. Assim, as especificações do ADC interno do PIC18F4550 ficaram como na Tabela 6.1.

Com estes valores, já é possível fazer as medições. Após obter todas as cinco medidas, exportamos os resultados de cada uma delas, e utilizamos todos os relatórios gerados pelo Programa de Análise de Dados para obter a média das medidas. Os resultados das medidas, assim como a média das medidas estão na Tabela 6.2.

Resolução (Bits)	8
Taxa de Conversão (kHz)	4.8943
V_{ref-} (V)	0
V_{ref+} (V)	3.3
N_{FFT} (Pontos)	4096

Tabela 6.1: Especificações do Conversor Interno do PIC18F4550.

Parâmetros	Medida 1	Medida 2	Medida 3	Medida 4	Medida 5	Média
DNL (LSB)	0.39028	0.37625	0.42972	0.39995	0.40874	0.40099
INL (LSB)	0.34713	0.31956	0.36337	0.34262	0.37467	0.34947
Ganho	1.00086	1.00091	1.00087	1.00092	1.00091	1.00089
Offset (LSB)	-0.00184	-0.00172	-0.00176	-0.00200	-0.00195	-0.00185
Falha de Códigos	0	0	0	0	0	0
SNR (dB)	47.46765	47.39784	47.46401	46.82884	47.73520	47.37871
SINAD (dB)	47.24971	47.18115	47.06271	46.60766	47.46617	47.11348
ENOB (dB)	7.55553	7.54414	7.52447	7.44888	7.59148	7.53290
THD (%)	0.09602	0.09651	0.13175	0.10413	0.10375	0.10643
SFDR (dB)	64.69194	65.74394	61.70221	64.34999	65.55450	64.40852

Tabela 6.2: Resultado das cinco Medidas e da Média das Medidas do Conversor Interno do PIC18F4550.

Podemos observar que as medidas variam muito pouco entre sí, o que demonstra que a medida consegue ser bastante confiável. Nos parâmetros obtidos pelo teste dos histogramas, os valores se mantiveram dentro da Tolerância de 10% adotada para a medida. Já os fatores obtidos através do teste da FFT tiveram uma variação bastante pequena, que provavelmente foi causada por algum ruído na onda de entrada. Ainda assim, a variação no número efetivo de *bits* (ENOB) dentre as cinco medidas foi de ≈ 0.1.

Como o *software* de Análise de Dados exporta os gráficos ponto-a-ponto, fizemos uma média dos gráficos das cinco medidas e plotamos os resultados, que podem ser vistos da Figura 6.1.

Finalmente, podemos comparar o resultado obtido pela plataforma com o fornecido pelo *Datasheet* do fabricante. Esta comparação pode ser vista na Tabela 6.3. Podemos observar que as médias das medidas se encontram dentro do especificado pelo fabricante com bastante folga. Especificamos a variação da DNL, já que especificamos a tolerância em relação a ela. Mesmo no pior caso, a DNL ficaria um pouco acima da metade do valor máximo especificado pelo fabricante, mostrando que o

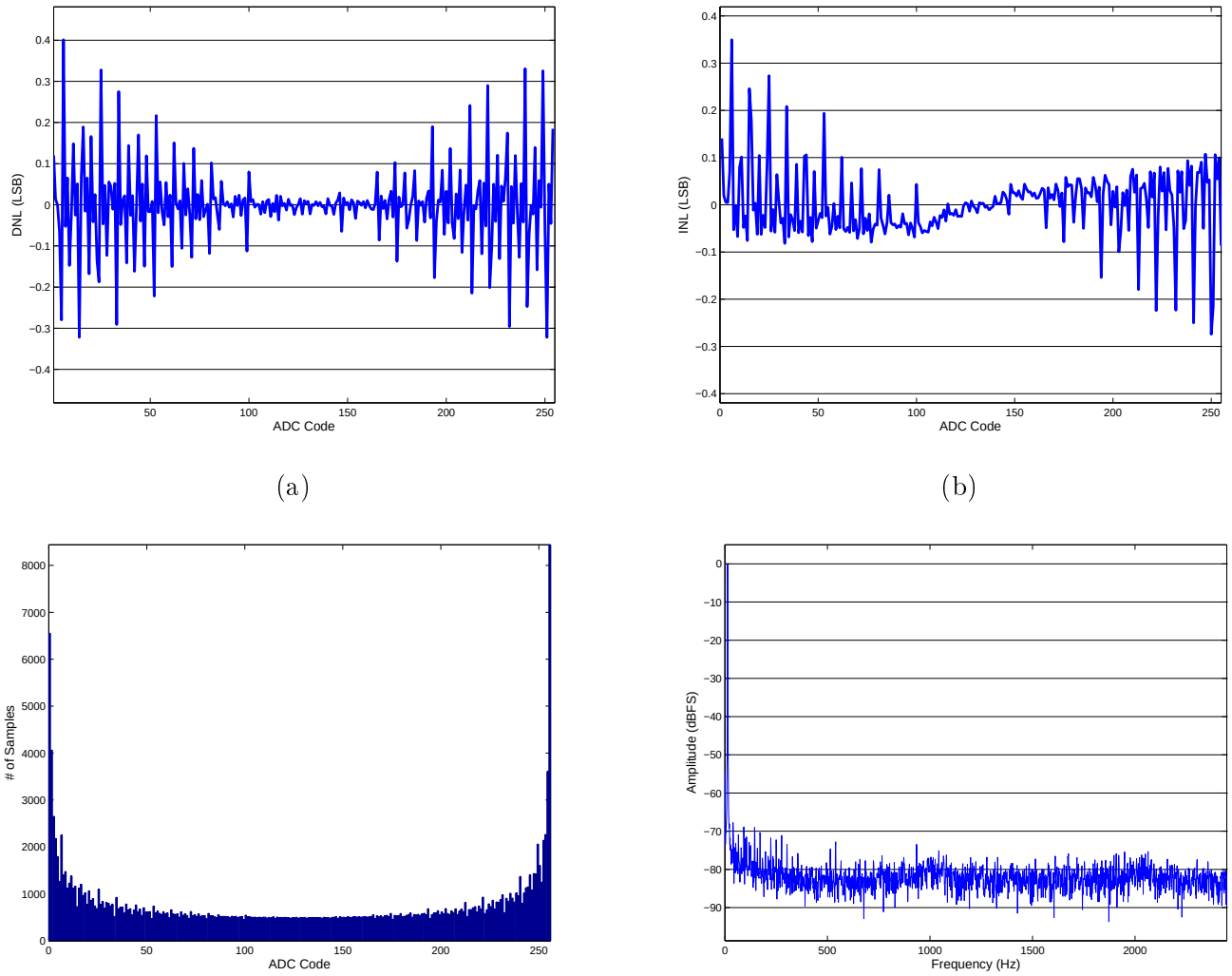


Figura 6.1: Figuras de Mérito do Conversor Interno do Circuito de Aquisição. (a) DNL. (b) INL. (c) Histograma. (d) FFT.

resultado obtido foi bastante satisfatório. Infelizmente, não consta no *Datasheet* [8] os valores especificados pelo fabricante para os parâmetros dinâmicos. Porém, como estavamos utilizando um conversor de 8 bits e obtivemos um número efetivo de *bits* médio de 7.53290, podemos observar que o nosso resultado chegou bastante próximo do valor teórico da medida, o que era o esperado.

Parâmetros	Média das Medidas	Fabricante
DNL (LSB)	0.40099 ± 0.1	$\leq \pm 1$
INL (LSB)	0.34947	$\leq \pm 1$
Ganho	1.00089	≤ 1.0128
Offset (LSB)	-0.00185	$\leq \pm 0.0193$
Falha de Códigos	0	0
SNR (dB)	47.37871	ND
SINAD (dB)	47.11348	ND
ENOB (dB)	7.53290	ND
THD (%)	0.10643	ND
SFDR (dB)	64.40852	ND

Tabela 6.3: Tabela comparativa entre os Valores Médios Medidos e os Entregues Pelo Fabricante. ND - Não determinado.

6.2 Conversor Externo do PIC18F4550

Para testarmos a aquisição de amostras externas, optamos por configurar um PIC18F4550, praticamente idêntico ao PIC do circuito de aquisição como puramente um conversor Analógico-Digital. Para fixarmos a velocidade de conversão, utilizamos um dos contadores embutidos no circuito do microcontrolador, e a rotina de interrupção acionada por ele. O interessante desta configuração é que, com ela, podemos regular a frequência de conversão do ADC apenas mudando parâmetros deste contador.

Do mesmo modo que na medida anterior, procuramos demonstrar a repetibilidade dos resultados. Assim, também foram realizadas cinco medições diferentes do conversor e, no fim, obtivemos a média destas medições.

Neste teste, configuramos dois Microcontroladores em distintos circuitos integrados, onde ambos são PIC18F4550, com duas frequências de conversão diferentes:

5 kHz e 10 kHz. A frequência real varia um pouco, pois após o timer acionar a interrupção, é executada uma pequena rotina, que demanda um pouco de tempo. Assim, medimos a real frequência de conversão de ambos com um frequencímetro, obtendo 4.78028 kHz e 9.15785 kHz, respectivamente. Ambos conversores foram configurados para utilizar uma fonte externa de 5 Volts como alimentação, e usar a própria alimentação como referências de conversão. Antes das medidas, averiguamos com o multímetro qual era exatamente a voltagem recebida por cada um dos conversores. As especificações finais ficaram como na Tabela 6.4.

Especificações	**Conversor 1**	**Conversor 2**
Resolução (Bits)	8	8
Taxa de Conversão (kHz)	4.78028	9.15785
V_{ref-} (V)	0	0
V_{ref+} (V)	5.0757	5.0713
N_{FFT} (Pontos)	4096	4096

Tabela 6.4: Especificações dos Conversores Externos 1 e 2.

Com todas as especificações em mãos, pudemos executar as cinco medidas para cada conversor. Exportamos os resultados de cada uma delas e utilizamos estes resultados para obter uma medida média para cada um dos conversores. O resultado das medidas para o Conversor 1 estão mostrados na Tabela 6.5 e suas figuras de mérito na Figura 6.2. Os resultados do Conversor 2 estão na Tabela e as figuras de mérito na Figura 6.3.

Parâmetros	Medida 1	Medida 2	Medida 3	Medida 4	Medida 5	Média
DNL (LSB)	0.38951	0.41512	0.36516	0.42007	0.40727	0.39943
INL (LSB)	0.41184	0.42707	0.41412	0.42926	0.42626	0.42171
Ganho	0.99980	0.99971	0.99967	0.99965	0.99966	0.99970
Offset (LSB)	-0.00383	-0.00371	-0.00364	-0.00360	-0.00379	-0.00371
Falha de Códigos	0	0	0	0	0	0
SNR (dB)	47.77193	48.77734	48.46312	47.90780	47.86378	48.15679
SINAD (dB)	47.18119	48.25214	47.86654	47.35729	47.25017	47.58147
ENOB (dB)	7.54414	7.72203	7.65798	7.57339	7.55560	7.61063
THD (%)	0.15601	0.13052	0.14483	0.14791	0.15754	0.14736
SFDR (dB)	57.47130	59.98070	58.67665	58.17930	57.68386	58.39836

Tabela 6.5: Resultado das cinco Medidas e da Média das Medidas do Conversor 1.

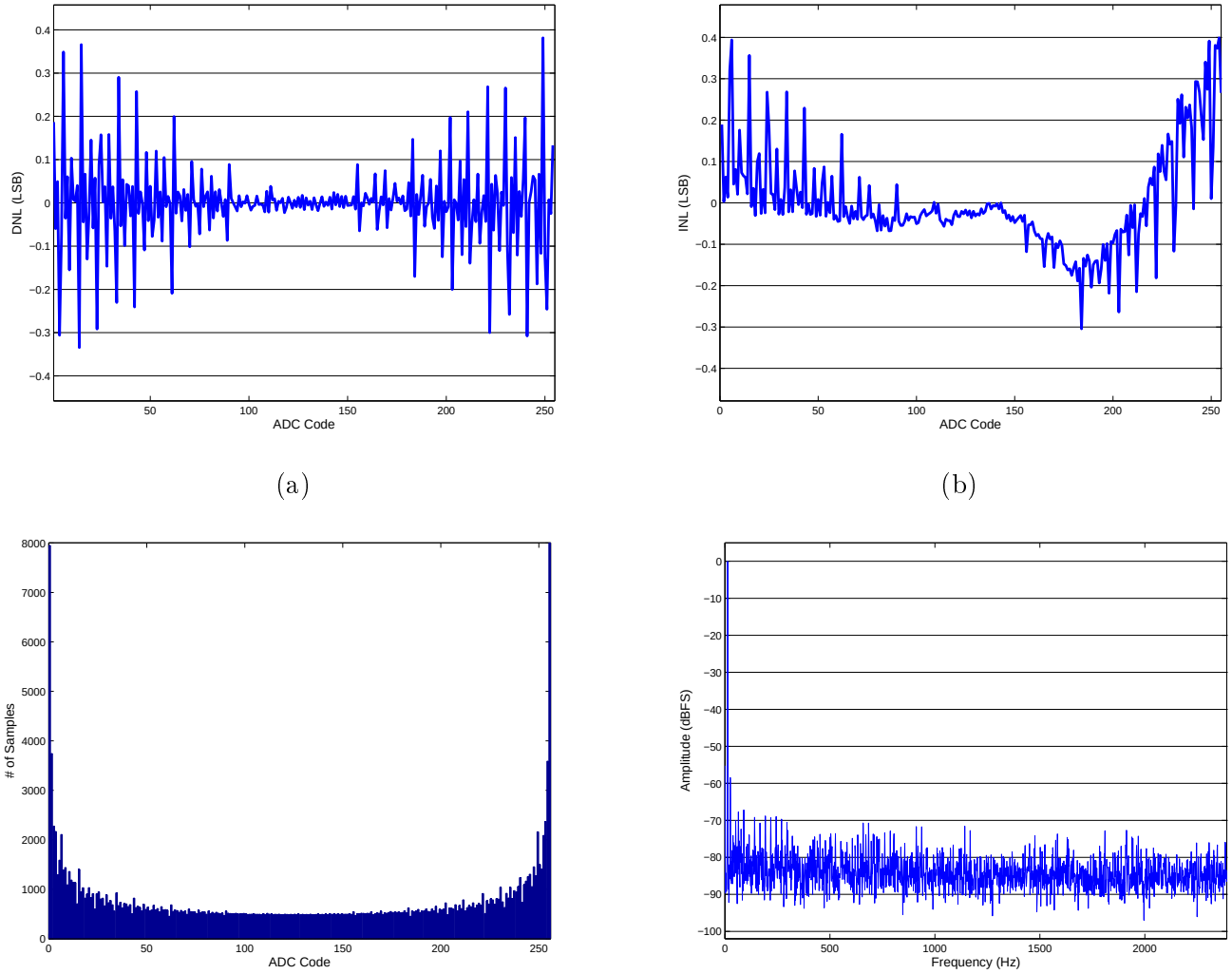


Figura 6.2: Figuras de Mérito do Conversor Externo 1. (a) DNL. (b) INL. (c) Histograma. (d) FFT.

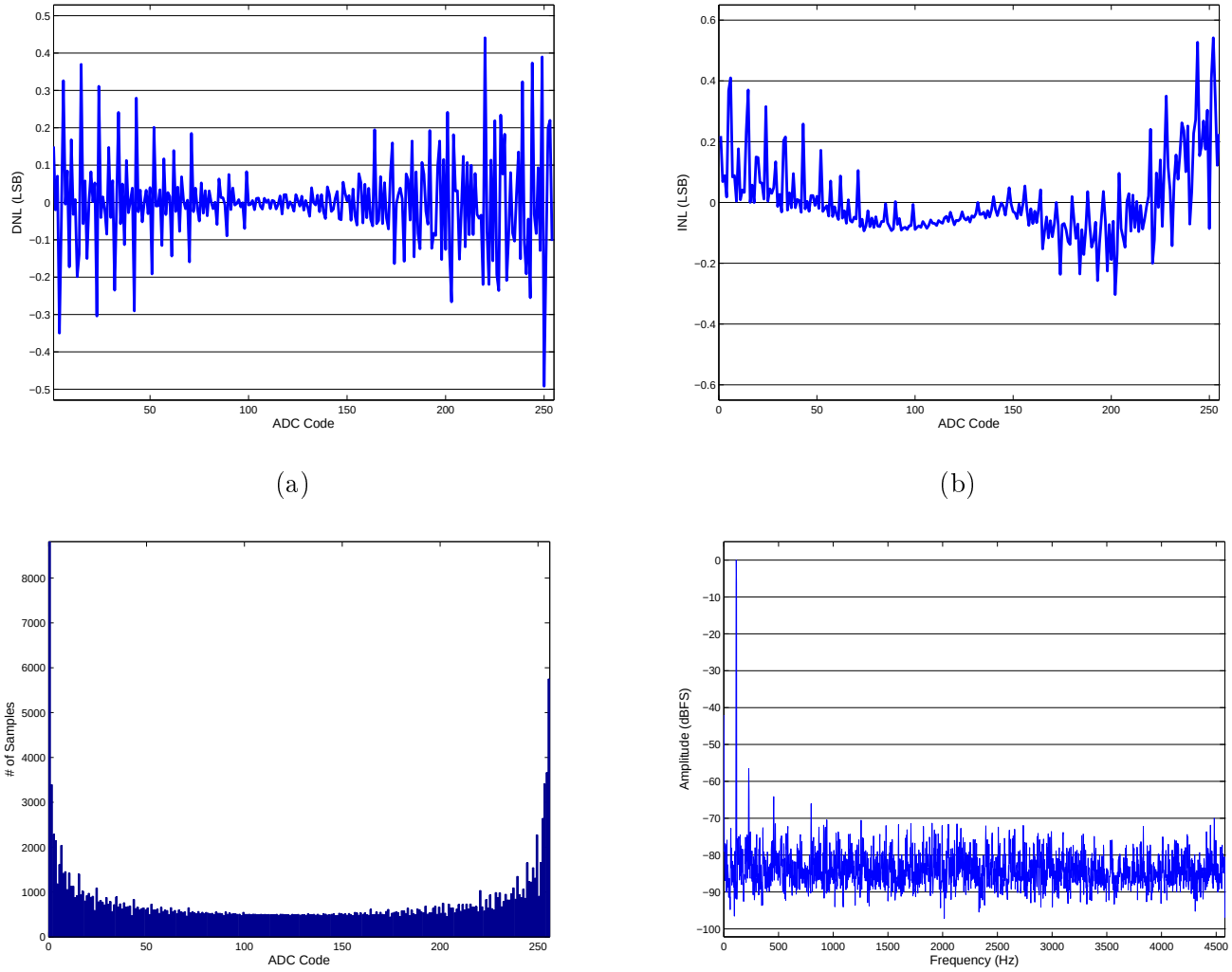


Figura 6.3: Figuras de Mérito do Conversor Externo 2. (a) DNL. (b) INL. (c) Histograma. (d) FFT.

Parâmetros	Medida 1	Medida 2	Medida 3	Medida 4	Medida 5	Média
DNL (LSB)	0.49787	0.47919	0.50839	0.48466	0.48978	0.49198
INL (LSB)	0.55065	0.53896	0.54626	0.53954	0.54052	0.54319
Ganho	0.99995	1.00000	0.99993	0.99998	0.99999	0.99997
Offset (LSB)	-0.00431	-0.00432	-0.00422	-0.00432	-0.00431	-0.00430
Falha de Códigos	0	0	0	0	0	0
SNR (dB)	45.95828	45.93415	45.78704	45.93931	46.02997	45.92975
SINAD (dB)	45.45067	45.46677	45.28118	45.43117	45.53397	45.43275
ENOB (dB)	7.25671	7.25939	7.22856	7.25347	7.27055	7.25374
THD (%)	0.17733	0.17023	0.18053	0.17781	0.17373	0.17592
SFDR (dB)	56.62230	56.90674	56.23826	56.27462	56.59480	56.52735

Tabela 6.6: Resultado das cinco Medidas e da Média das Medidas do Conversor 2.

Com todas as medidas já obtidas, podemos finalmente fazer um quadro comparativo entre os conversores e o fabricante, que é o mesmo para ambos. Podemos também comparar os dois conversores entre sí, e observar o efeito de uma frequência de conversão mais elevada. Estas comparações podem ser feitas através da Tabela 6.7.

Parâmetros	**Conversor 1 - Média**	**Conversor 2 - Média**	**Fabricante**
DNL (LSB)	0.39943 ± 0.1	0.49198 ± 0.1	$\leq \pm 1$
INL (LSB)	0.42171	0.54319	$\leq \pm 1$
Ganho	0.99970	0.99997	≤ 1.0128
Offset (LSB)	-0.00371	-0.00430	$\leq \pm 0.0193$
Falha de Códigos	0	0	0
SNR (dB)	48.15679	45.92975	ND
SINAD (dB)	47.58147	45.43275	ND
ENOB (dB)	7.61063	7.25374	ND
THD (%)	0.14736	0.17592	ND
SFDR (dB)	58.39836	56.52735	ND

Tabela 6.7: Tabela comparativa entre os Valores Médios Medidos nos dois conversores e os Entregues Pelo Fabricante. ND - Não determinado.

A primeira observação que pode ser feita é em relação ao fabricante. Podemos notar que ambas as medidas respeitam perfeitamente os valores limites dado pelo fabricante, o que demonstra uma certa precisão no método de medição utilizado pela plataforma. Outro fator interessante na tabela é que as medidas obtidas do Conversor 1 foram melhores que as do Conversor 2. Isto se deve à escolha de frequência, onde uma é praticamente o dobro da outra. Como o tempo de conversão no PIC é igual, mudaria entre um e outro o tempo de aquisição da amostra, que

é muito mais longo no Conversor 1. Isto faz com que a precisão da conversão seja maior, gerando figuras de mérito melhores. Esta é provavelmente a razão pela qual o fabricante não fornece os parâmetros dinâmicos em seu *datasheet*, pois como é possível alterar esta frequência de conversão, estes parâmetros iriam variar bastante, pois dependem diretamente desta frequência. Ainda assim é possível observar que as medidas dinâmicas foram razoáveis, pois o número efetivo de *bits* se aproxima bastante do número teórico de bits, que é 8 para ambos os conversores.

Algo importante de ser mencionado é o tempo total para que a medição seja feita por completo. Claramente, este valor depende do número de amostras que serão adquiridas e da taxa de conversão do dispositivo em teste. Para os testes apresentados nesta seção o tempo total de teste ficou entre 5 e 10 minutos para cada medição. Este tempo é bastante razoável, levando em conta que o conversor possui um *throughput* bastante baixo.

Capítulo 7

Conclusão

O objetivo deste trabalho foi projetar uma plataforma de testes para Conversores Analógico-Digitais que fosse bastante simples de se utilizar e realizasse medidas confiáveis. Com isso, pretendíamos deixar automatizado o processo de medição de conversores, para que possa ser utilizado em futuros circuitos integrados fabricados na nossa universidade.

Podemos concluir que o objetivo foi cumprido com êxito. A plataforma foi projetada utilizando métodos recomendados pelo IEEE, o que certifica as medidas feitas por ela. Seu uso é tão simples quanto apertar um botão que faz todos os cálculos dos parâmetros de teste e o próprio *software* configura o equipamento para medida. Um circuito de aquisição também foi projetado, facilitando o interfacemento do dispositivo em teste com a plataforma. Após a medição, é possível exportar os resultados em formatos prontos para serem publicados em artigos acadêmicos.

As medições feitas pela plataforma comprovaram o seu funcionamento, pois comparando medidas de conversores já existentes no mercado com as informações dadas pelo *datasheet* dos mesmos, pudemos observar que as medidas feitas pela plataforma foram muito bem sucedidas.

A maior dificuldade do projeto foi na velocidade de aquisição e transmissão das amostras obtidas, pois a plataforma fica limitada pela velocidade do microcontrolador de processar e enviar dados, o que só consegue ser melhorado utilizando um componente melhor, e consequentemente, mais caro. Porém, para as especificações

que desejávamos, o microcontrolador utilizado demonstrou ser o suficiente.

7.1 Melhorias e Trabalhos Futuros

Esperamos utilizar esta plataforma no futuro para medir conversores que já foram fabricados e estão em processo de projeto na nossa universidade. Com isso, podemos fazer algumas melhorias caso os novos projetos exijam características diferentes da plataforma.

Uma idéia que surgiu para que houvesse uma aquisição em frequências maiores foi a utilização de uma memória FIFO, que trabalha com frequências muito altas e tem um espaço de armazenamento suficiente para os testes, o que aumentaria bastante a frequência máxima de conversão suportada pela plataforma.

Outra seria incluir direto na plataforma um cálculo para que seja exibido o resultado médio de um número de amostras definido pelo usuário, deixando o sistema ainda mais robusto.

Para trabalhos futuros, seria interessante introduzir alguns outros testes para obter alguns outros parâmetros que ficaram de fora do escopo da plataforma, mas que são pertinentes para certas topologias de amplificadores, como, por exemplo, a imprecisão da amostragem (*Jiiter*) e a Intermodulação de Dois Tons (IMD2).

Referências Bibliográficas

[1] JOHNS, D. A., MARTIN, K. *Analog Integrated Circuit Design.* John Wiley & Sons, Inc, 1997.

[2] MALOBERTI, F. *Data Converters.* 1 ed. The Netherlands, Springer, 2007.

[3] OPPENHEIN, A. V., SCHAFER, R. W. *Discrete-Time Signal Processing.* 3 ed. EUA, Pearson Education Inc., 2009.

[4] VAN DE PLASSCHE, R. *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters.* 2 ed. The Netherlands, Kluwer Academic Publishers, 2003.

[5] WORKING GROUP OF THE IEEE INSTRUMENTATION AND MEASUREMENT SOCIETY. *IEEE Std 1241-2000: IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters.* IEEE, 2001.

[6] BLAIR, J. "Histogram Measurement of ADC Nonlinearities Using Sine Waves", *IEEE Transactions on Instrumentation and Measurement*, v. 43, n. 3, pp. 373–383, jun. 1994.

[7] DOERNBERG, J., LEE, H.-S., HODGES, D. A. "Full-Speed Testing of A/D Converters", *IEEE Journal of Solid-State Circuits*, v. sc-19, n. 6, pp. 820–827, dez. 1984.

[8] MICROCHIP. "PIC18F2455/2550/4455/4550 Datasheet". 2006.

[9] AXELSON, J. *USB Complete: Everything You Need to Develop USB Peripherals.* Lakeview Research LLC, 2005.

[10] TEXTRONIX, I. "AFG3000 Series Arbitrary Function Generators Programmer Manual". 2004.

[11] JUAN, Z., DENGFENG, J., FANGYU, L. "Research and Realization of Autmatic Test System for ADC Based on Matlab". In: *Image Analysis and Signal Processing, 2009. IASP 2009*, Abril 2009.

Apêndice A

Manual da Plataforma de Testes

Neste Apêndice, iremos apresentar um Manual para o uso da plataforma de testes. Iremos apresentar como configurar o MATLAB para o uso do *software*, e um guia para a instalação do *driver* da plataforma.

A.1 Preparação do MATLAB

Para que a plafatorma funcione com todas as suas funções, é de suma importância a instalação do protocolo VISA, acrônimo em inglês para Arquitetura de *Software* para Instrumentos Virtuais, que é nada mais que uma espécie de linguagem de comunicação padronizada para Instrumentos de medição.

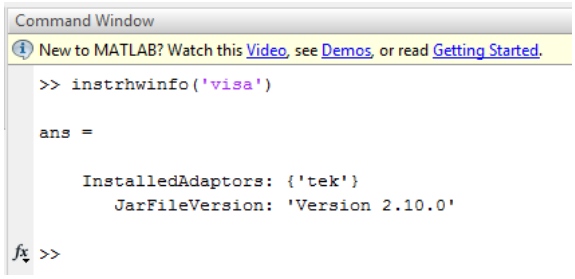
Este protocolo possui diferentes implementações para diferentes fabricantes. O MATLAB funciona com as implementações dos fabricantes Agilent, National Instruments e Tektronix. Como a plataforma de testes está configurada para usar o gerador de sinais AFG3252 da Tektronix, optamos pela instalação da implementação da Tektronix. O uso de alguma outra implementação é possível, porém não foram testadas as outras implementações.

O arquivo de instalação pode ser obtido gratuitamente na página da Tektronix, mais precisamente na página `http://www.tek.com/oscilloscope/tds7054-software/tekvisa-connectivity-software-v400`. Para obter o arquivo, é necessário cadastro no *site* da Tektronix. A instalação deste pacote é

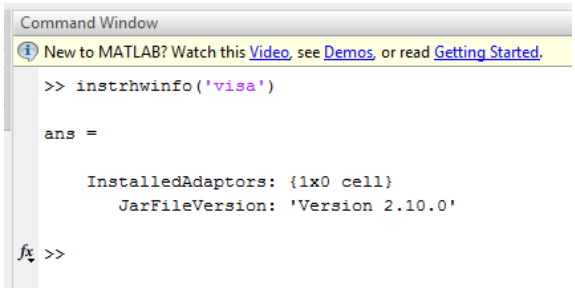
muito direta, apenas deve-se seguir o guia de instalação do arquivo.

É importante ressaltar que a implementação do VISA da Tektronix somente funciona no MATLAB de 32 bits. Apesar de instalado, caso o usuário esteja utilizando a versão de 64 bits do MATLAB, o gerador de sinais não será identificado pelo *software* de Aquisição.

Para saber se a instalação foi bem sucedida, o usuário pode digitar o código *instrhwinfo('visa')*. Caso o pacote esteja instalado, o resultado obtido é o da Figura A.1a. Caso a instalação não seja bem sucedida, o resultado do comando é o da Figura A.1b.



(a)



(b)

Figura A.1: Resultado do comando *instrhwinfo('visa')* caso (a) VISA instalado com sucesso. (b) VISA não instalado.

A.2 Guia de Uso Passo-a-Passo

Para facilitar o usuário, elaboramos este guia passo-a-passo para utilizar a Plataforma de Testes para Conversores Analógico-Digitais proposta neste trabalho. Para prosseguir com este guia, é necessário que o usuário possua os Arquivos do CD do trabalho.

Passo 1

Conecte o Circuito de Aquisição à USB do seu computador. Caso não seja a primeira utilização, pule para o Passo 6. Se for a primeira utilização do dispositivo, deve ser feita a instalação do *driver*. O Windows tentará fazer a instalação automática, porém sem sucesso. No canto direito inferior da tela do usuário, irá aparecer uma mensagem do Windows indicando que o driver não foi encontrado.

Passo 2

Abra o **Gerenciador de Dispositivos**, que pode ser acessado através do Painel de Controle do Windows. Irá aparecer um dispositivo com o nome **"SERIAL DEMO"**, em destaque pois não foi feita a instalação do *driver*, como na Figura A.2. Clique com o botão direito no dispositivo e selecione a opção **"Atualizar Driver..."**.

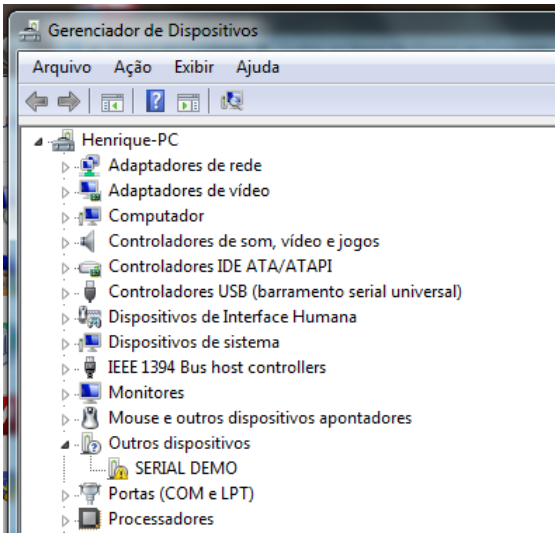


Figura A.2: Gerenciador de Dispositivos do Windows, com a plataforma sem o *driver* instalado.

Passo 3

Uma nova janela será aberta, semelhante a da Figura A.3. Selecione a opção "Procurar *software* de *driver* no computador".

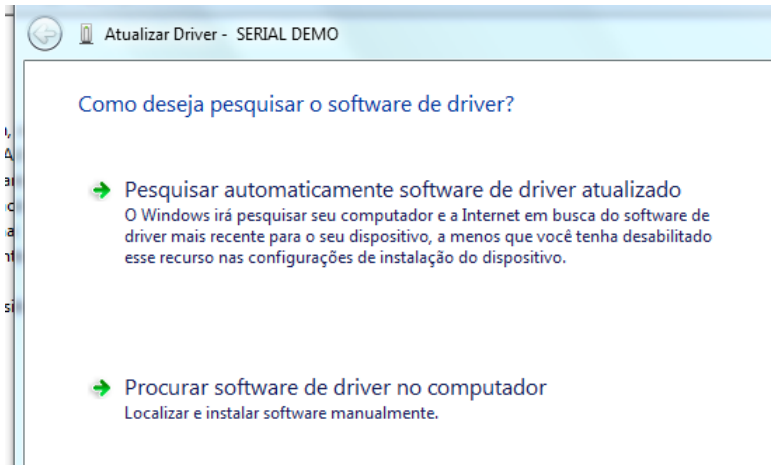


Figura A.3: Primeira janela do Instalador do *Driver*.

Passo 4

Na próxima janela, o Windows pedirá que você escolha a pasta que contem o *driver*. Selecione a pasta **"Driver de Instalação"** nos Arquivos do CD da Plataforma de Teste e aperte OK. O sistema operacional irá iniciar então a instalação do *driver*. Ao aparecer a Figura A.4, clique em "Instalar".

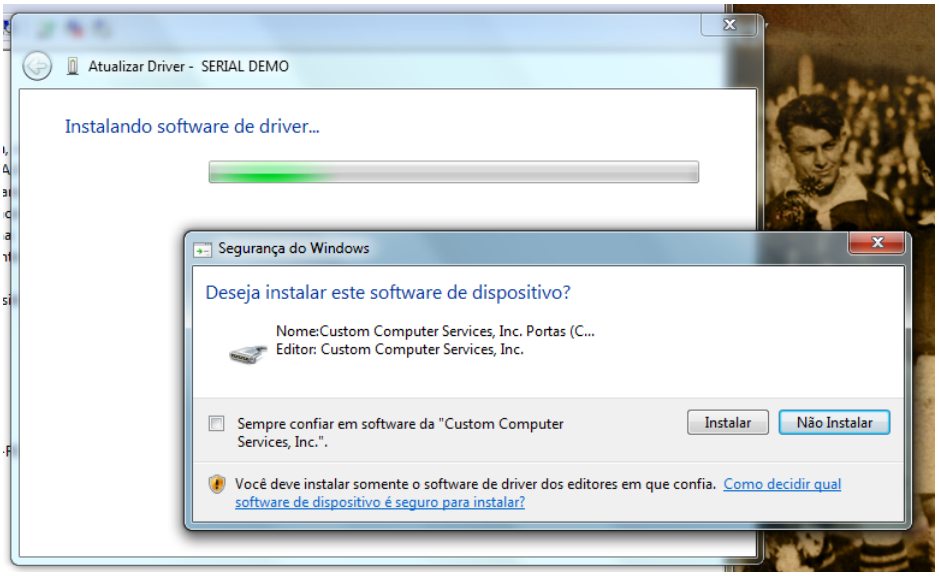


Figura A.4: Segunda janela do Instalador do *Driver*.

Passo 5

Ao finalizar a instalação, o **Gerenciador de Dispositivos** irá ter removido o **SERIAL DEMO** e irá aparecer um novo dispositivo, que pode ser visto em **"Portas**

(COM e LPT)", sob o nome de **USB to UART (COMX)**, onde o número X é automaticamente escolhido pelo computador. Para o exemplo da Figura A.5, a porta escolhida foi a COM3.

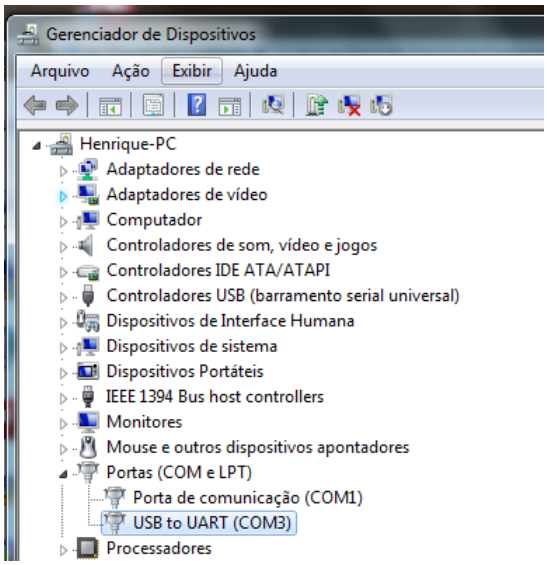


Figura A.5: Gerenciador de Dispositivos do Windows, com o driver já instalado.

Passo 6

Com o *driver* já instalado, pode-se iniciar a aquisição de dados. Certifique que o gerador de sinais também esteja conectado à USB. Para iniciar, vá para os Arquivos do CD e abra a pasta **"Software da Plataforma"** e clique no arquivo **"Software_Aquisicao.m"**. Clique em Executar para abrir a interface gráfica, já demonstrada na Figura A.6.

Passo 7

Inclua todos os dados do seu conversor e pressione o botão Executar Medidas. Aguarde a conclusão da aquisição de amostras. O progresso pode ser visto no painel de *Status* da Aquisição.

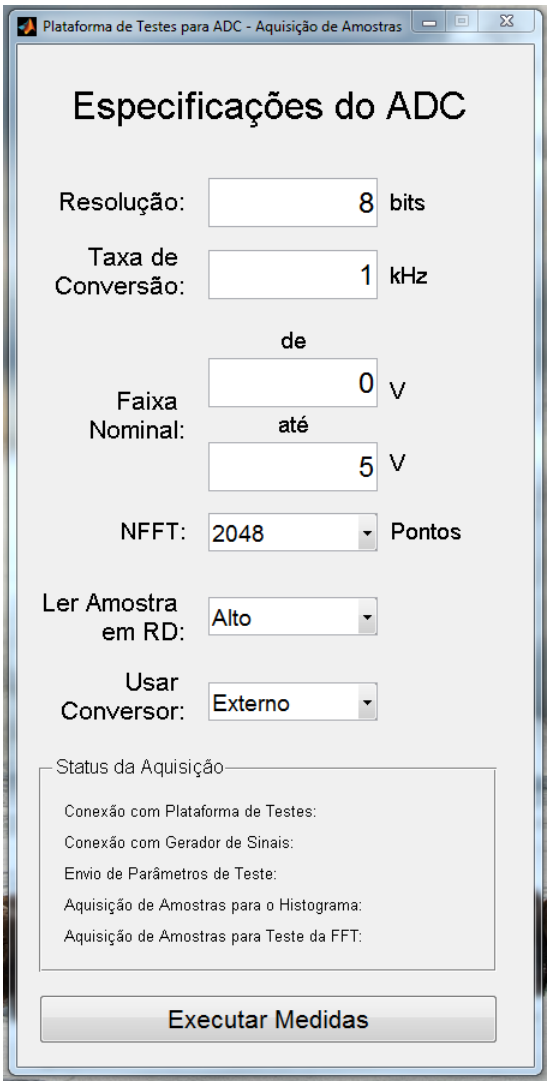


Figura A.6: Interface Gráfica do Programa de Aquisição de Amostras.

Passo 8

Ao finalizar a aquisição, o *software* de Análise será aberto *automaticamente*, gerando a interface da Figura A.7. Caso o resultado da medição for satisfatório, o usuário pode exporta-lo para a mesma pasta que se localiza o *Software* da Plataforma, clicando no botão de Exportar Resultados. Caso contrário, a medição pode ser repetida facilmente. É necessário porém que o circuito de aquisição seja desconectado e conectado à plataforma, para resetar o *firmware*. Feito isso, volta-se ao Passo 6 deste guia para serem feitas novas medidas.

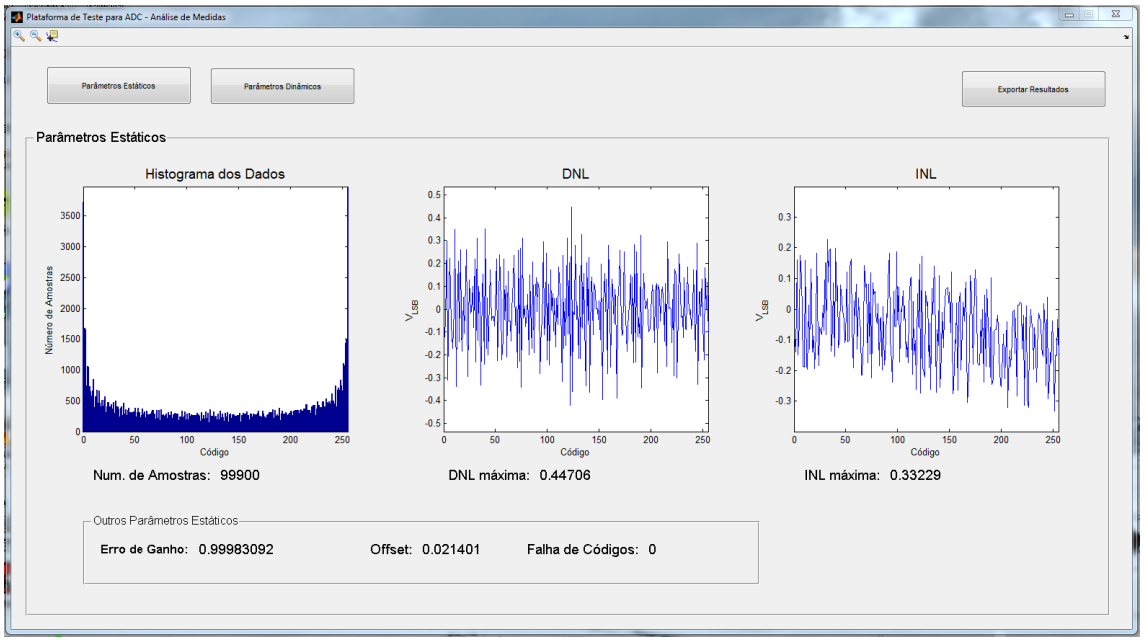


Figura A.7: Interface Gráfica do Programa de Aquisição de Amostras.

Passo 9

Caso o usuário deseje utilizar o *software* de Análise sem que haja a aquisição de dados, ou seja, utilizando um arquivo com amostras feito anteriormente pelo programa de Aquisição de Amostras, basta clicar no arquivo **"Software_Analise.m"** e executá-lo pelo MATLAB, que todas as figuras de mérito serão calculadas automaticamente na abertura do programa.

Apêndice B

Código-Fonte dos Programas no MATLAB

Neste Apêndice são apresentados os códigos-fonte dos programas feitos no MATLAB.

B.1 *Software* de Aquisição de Amostras

```
%      Executes on button press in b_Measurement.
function b_Measurement_Callback(hObject, eventdata, handles)
% hObject    handle to b_Measurement (see GCBO)
% eventdata  reserved - to be defined in a future version of
   MATLAB
% handles    structure with handles and user data (see GUIDATA)

%#####################################################
%
%   Aquisiçao de Dados - Plataforma de Teste de ADC
%
%   Ao pressionar este botão, o usuário inicia
%   a bateria de testes. O programa será dividido em passos.
%   1o - Cálculo de parâmetros de acordo com os dados
```

```
%    entregues pelo usuário. Deve se observar se os parâmetros
%    estao dentro dos conformes.
%   N <= 16, Faixa Nominal <= 10V , Fs < 60k
%    2o   Conexão com PIC e Gerador de Sinais, e envio
%    de parâmetros
%
%#########################################################



%#########################################################################
%    Leitura dos Valores de Entrada entregue pelo usuário
%#########################################################################

% N é a resolução. Deve estar entre 1 e 16
N = str2double(get(handles.t_Resolution,'String'));
if ((N > 16)||(N<1))
    h = errordlg(['O número de bits deve ser menor que 16.'...
                  ' Para mais detalhes, consultar Manual.']);
    uiwait(h)
    return
end

%fConv é a frequencia de conversão. A USB limita em 60kHz.
fConv = str2double(get(handles.t_FreqConv,'String'));
% if (fConv > 60)
%     h = errordlg(['Esta frequência de conversão não é suportada
    pela plataforma.' ...
%                   'Para mais detalhes, consultar Manual.']);
%     uiwait(h)
%     return
```

```
% end

% dynRange é a faixa nominal. Deve ter uma dif. de 5V no máximo.
dynRangeMin = str2double(get(handles.t_DynRangeMin,'String'));
dynRangeMax = str2double(get(handles.t_DynRangeMax,'String'));
if (dynRangeMax dynRangeMin > 10)
    h = errordlg(['Esta faixa nominal não é suportada pela
   plataforma.' ...
                  'Para mais detalhes, consultar Manual.']);
    uiwait(h)
    return
end

% NFFT é o número de pontos da FFT. Não necessita de verificação.
val = get(handles.drop_NFFT,'Value');
string_list = get(handles.drop_NFFT,'String');
selected_string = string_list{val};
NFFT = str2double(selected_string);

% O usuário deve informar se leitura é feita em high
% ou low da fase de leitura
val = get(handles.drop_RD,'Value');
switch val
    case 1
        seletor_RD = 'HIGH';
    case 2
        seletor_RD = 'LOW';
end

% O usuário deve informar se o conversor a ser usado
% é o interno ou o externo.
```

```
val = get(handles.drop_IntExt,'Value');
switch val
    case 1
        seletor_ADC = 'EXT';
    case 2
        seletor_ADC = 'INT';
end

%#########################################################
%
%       Cálculo de Parâmetros para Aquisição.
%
%       Parâmetros Estáticos:
%       B   u   R   M   fs_est   Vod
%
%       Parâmetros Dinâmicos
%       fs_din
%
%#########################################################

% Escolheremos os parâmetros de acordo com os valores
% desejados no trabalho escrito, onde B é 10% na DNL.
B = 0.10; % Tolerancia
u = 0.01; % Incerteza
v = 1   u; % Certeza
A = (dynRangeMax   dynRangeMin)/2;
d = (dynRangeMax + dynRangeMin)/2;
Q = 2*A/(2^N); % Degrau de Quant. Teórico
sigma = 1e 5; % é o ruído do circuito. Ele está sendo
   superestimado por Qn/2
% O overdrive usado é o para a DNL.
```

```
V_OD = sigma*(max(3,sqrt(3/2/B)));

%
%   Frequencia do sinal de entrada
%
D = 3;
delta_rho = 1e 6; % O gerador de sinais utilizado tem precisão de
    1uHz
% f_input = fConv*D/M;
%ro = f_input/fConv;

M_max = (sqrt(D/4/(D 1)/delta_rho));
k = floor((M_max + 1)/D);
M = k*D    1;

f_input = fConv*D/M;
display(M);
%
%   Número de conversões mínima.
%

% Calculando o parâmetro indicado no paper
Z_u_by2 = sqrt(2)*erfcinv(1    v^(2));
Z_Nu_by2 = sqrt(2)*erfcinv(1    v^(2^N)); % Para o pior caso.

% Utilizando a formula em Blair, eq. (12)
C = 2; % Para DNL
Ku = Z_Nu_by2; % Para o pior caso

V = 2*(A); % Usando valor ideal.
alfa = 1 + 2*V_OD/V;
```

```
sigma_star = min(sigma,Q/1.13); % Para a DNL

R_min = C*((2^(N 1)*Ku/B)^2)*(alfa*pi/M)*((1.13*sigma_star/V)
    +0.2*(alfa*pi/M));



R = ceil(1.1*R_min); % Por enquanto, usar R menor para testes

Vmin_h = d    V/2 + Q    V_OD;
Vmax_h = d + V/2    Q + V_OD;
% Parâmetros para serem enviados:
% R,M    PIC
% f_input, Vmax_h e Vmin_h    Gerador

%
%   Parâmetros de Teste dinâmicos
%

B_width = fConv/(NFFT);
k_bin = 51;
f_input_fft = k_bin*B_width;
Vmin_fft = d    V/2 + Q;
Vmax_fft = d + V/2    Q;

% Parâmetros a serem enviados:
% NFFT    PIC
% f_input_fft, Amin, Amax, sem OD    Gerador

%%%%%%%%%%%%%%% PARÂMETROS SALVOS %%%%%%%%%%%%%%%
fName = mfilename;
```

```
Path = mfilename('fullpath');
fileP = strrep(Path,fName,'param.txt');
%   Padrão de Escrita:
%   Parametro<space>Valor\r\n
%   Ordem dos Parâmetros: N, FreqConv, Faixa Din Min e Max
param_File = fopen(fileP ,'w');
fprintf(param_File,'N %i\r\n',N);
fprintf(param_File,'Freq_Conv %3.12f\r\n',fConv); %em kHz
fprintf(param_File,'Faixa_Din_Min %3.12f\r\n',dynRangeMin);
fprintf(param_File,'Faixa_Din_Max %3.12f\r\n',dynRangeMax);
fprintf(param_File,'NFFT %i\r\n',NFFT);
fprintf(param_File,'%s\r\n',seletor_ADC);
fprintf(param_File,'################################ HIST
    ######################\r\n');
% B, u, V_OD, M, R, f_input, Vmin_h, Vmax_h
fprintf(param_File,'B %3.12f\r\n',B);
fprintf(param_File,'u %3.12f\r\n',u);
fprintf(param_File,'V_OD %3.12f\r\n',V_OD);
fprintf(param_File,'M %3.12f\r\n',M);
fprintf(param_File,'R %3.12f\r\n',R);
fprintf(param_File,'f_input %3.12f\r\n',f_input);
fprintf(param_File,'Vmin_h %3.12f\r\n',Vmin_h);
fprintf(param_File,'Vmax_h %3.12f\r\n',Vmax_h);
% FFT: f_input_fft, Vmin_fft, Vmax_fft
fprintf(param_File,'################################ FFT
    ######################\r\n');
fprintf(param_File,'f_input_fft %3.12f\r\n',f_input_fft);
fprintf(param_File,'Vmin_fft %3.12f\r\n',Vmin_fft);
fprintf(param_File,'Vmax_fft %3.12f\r\n',Vmax_fft);
fclose(param_File);
```

```
%#################################################
%
%       Conexão com instrumentos:
%       >Plataforma de Testes   serial
%       >Gerador de Sinais    visa
%       >Caso nao seja conectado algum dos
%       dois, deve se cancelar função
%       >Ver se há algum instrumento em instrfind.
%       Se sim, fclose e delete.
%
%#################################################

% Deletando os outros instrumentos na memória do
% MATLAB, evitando conflito
if(isempty(instrfind)==0)
    fclose(instrfind);
end
delete(instrfind); % Apagando os instrumentos da memória do
   matlab

serial_info = instrhwinfo('serial');
if(isempty(serial_info.ObjectConstructorName))
    h = errordlg(['A plataforma de testes não foi encontrada.'...
                  ' Certifique se de que ela esteja conectada a
    USB.']);
    uiwait(h)
    return
end
p_Teste = eval(serial_info.ObjectConstructorName{1});
set(p_Teste, 'BaudRate', 921600);
set(p_Teste, 'DataBits', 8, 'StopBits', 1);
```

```
set(p_Teste, 'Terminator', 'LF', 'Parity', 'none');
set(p_Teste, 'FlowControl', 'none');

% Tentar abrir objeto. Caso falhe, avisa ao usuário
try
    fopen(p_Teste);
catch err
    %display(err.identifier)
    if(strcmp(err.identifier,'MATLAB:serial:fopen:opfailed'))
        h = errordlg(['A plataforma de testes não não pode ser
    conectada.'...
                      ' Certifique se de que ela esteja
    conectada a USB.']);
        uiwait(h)
        close(gcbf);
        return
    end
end

set(handles.t_OK1,'Visible','on');
drawnow;
% Criando objeto do gerador de sinais.

visa_info = instrhwinfo('visa','tek');

if(isempty(visa_info.ObjectConstructorName))
    h = errordlg(['A gerador de sinais não foi encontrado.'...
                  ' Certifique se de que ele esteja conectada a
    USB.']);
    uiwait(h)
    return
```

```
end

g_Sinais = eval(visa_info.ObjectConstructorName{1});

% Tentar abrir objeto. Caso falhe, avisa ao usuário
try
    fopen(g_Sinais);
catch err
    display(err.identifier)
    if(strcmp(err.identifier,'instrument:fopen:opfailed'))
        h = errordlg(['Não foi possível conectar ao gerador de
   sinais.'...
                      'Tente novamente.']);
        uiwait(h)
        return
    end
end

set(handles.t_OK2,'Visible','on');

%#########################################################
%
%       Aquisição de Amostras
%
%#########################################################


%%% PRIMEIRO MANDAM SE OS PARÂMETROS AO PIC %%%%%%
% Dos parâmetros do usuario: N, NFFT, INT/EXT e HIGH/LOW
fprintf(p_Teste,'%d\r\n',N);
fprintf(p_Teste,'%d\r\n',NFFT);
```

```
fprintf(p_Teste,'%s\r\n',seletor_RD); % 'HIGH' ou 'LOW'
fprintf(p_Teste,'%s\r\n',seletor_ADC); % 'INT' ou 'EXT'
% Dos parâmetros para os hist:
fprintf(p_Teste,'%d\r\n',R);
fprintf(p_Teste,'%d\r\n',M);

% Parâmetro importante para aquisição.
if(N <= 8)
    int_size = 'uint8';
else
    int_size = 'uint16';
end


%%% RECEBE CONFIRMAÇÃO DO PIC PARA PROSEGUIR %%%%%%%%%%
code_conf = fscanf(p_Teste,'%s');
if(strcmp(code_conf,'OK') == 0)
    h = errordlg(['A plataforma não pode ler os parâmetros.'...
                  'Começe denovo.']);
    uiwait(h);
    fclose(instrfind);
    close(gcbf);
    return
end
set(handles.t_OK3,'Visible','on');
drawnow;
%%% CONFIGURAR GERADOR DE SINAIS PARA TESTE DOS HIST %%%%

% Selecionar impedância alta
fprintf(g_Sinais,'OUTPut1:IMPedance INFinity');
% Configurar frequência de entrada
```

```
fprintf(g_Sinais,'SOURce1:FREQUENCY:FIXED %3.8fKHz',f_input);
% Amplitude da senóide
fprintf(g_Sinais,'SOURce1:VOLTage:HIGH %fV',Vmax_h);
fprintf(g_Sinais,'SOURce1:VOLTage:LOW %fV',Vmin_h);
% Ligar o Canal
fprintf(g_Sinais,'OUTPut1:STATE ON');

%%% APÓS GERADOR CONFIGURADO, INICIAR AQUISIÇÃO %%%%%%%%
pause(1);

%%% ESTÁTICA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Buff_Size = max(factor(M));
if(Buff_Size < M/Buff_Size)
    Buff_Size = M/Buff_Size;
end
display(Buff_Size);
j_max = M/Buff_Size;

samples_est = zeros(M,R);
set(handles.t_OK4,'Visible','on','String','0%','ForegroundColor',
    'black');
for i = 1:R
    % Enviar flag de start
    set(handles.t_OK4,'String',sprintf('%3.2f%%',i/R*100));
    drawnow;
    fprintf(p_Teste,'%s\r\n','ST_HIST');
    for j = 1:j_max
        % Iniciar aquisição
        a = (Buff_Size)*(j-1)+1;
        b = Buff_Size*j;
        samples_est(a:b,i) = fread(p_Teste,Buff_Size,int_size);
```

```
    end
    x = p_Teste.BytesAvailable;
    if(x)
        display(x)
    end
end

% Gravar arquivo com amostras estáticas
file_Samp = strrep(Path,fName,'amostras_histograma.dat');
dlmwrite(file_Samp,samples_est,',');

% Imprimir na tela que foi finalizada
set(handles.t_OK4,'Visible','on');
drawnow;

%%% DINÂMICA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Buff_Size = 64; %Buffer multiplo de 2.
N_points = NFFT*3;
i_max = N_points/Buff_Size;
samples_din = zeros(1,N_points);

% Configurar gerador novamente:

% Configurar frequência de entrada
fprintf(g_Sinais,'SOURce1:FREQUENCY:FIXED %3.8fKHz',f_input_fft);
% Amplitude da senóide
fprintf(g_Sinais,'SOURce1:VOLTage:HIGH %fV',Vmax_fft);
fprintf(g_Sinais,'SOURce1:VOLTage:LOW %fV',Vmin_fft);
```

```
% Enviar flag de start
fprintf(p_Teste,'%s\r\n','ST_FFT');
for i = 1:i_max
    % Iniciar aquisição
    a = (Buff_Size)*(i-1)+1;
    b = Buff_Size*i;
    samples_din(a:b) = fread(p_Teste,Buff_Size,int_size);
end
file_Din = strrep(Path,fName,'amostras_fft.dat');
dlmwrite(file_Din,samples_din,',');
% Imprimir na tela que foi finalizada
set(handles.t_OK5,'Visible','on');
drawnow;

%%% FECHAR COMUNICAÇÃO COM OBJETOS
fclose(p_Teste);
fclose(g_Sinais);

% ABRIR PROGRAMA DE ANÁLISE
PT_ADC_Analysis;
close(gcbf);
```

PT_ADC_Aquisition.m

B.2 *Software* de Análise de Dados

```
%     Executes just before PT_ADC_Analysis is made visible.
function PT_ADC_Analysis_OpeningFcn(hObject, eventdata, handles,
    varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved  to be defined in a future version of
    MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PT_ADC_Analysis (see
    VARARGIN)

% Choose default command line output for PT_ADC_Analysis
handles.output = hObject;

%#####################################################
%
%       Função de abertura do Programa de Analise de Dados
%
%#####################################################

% Identificação dos arquivos

fName = mfilename;
Path = mfilename('fullpath');
fileP = strrep(Path,fName,'param.txt');
arquivoHistograma = strrep(Path,fName,'amostras_histograma.dat');
arquivoFFT = strrep(Path,fName,'amostras_fft.dat');

fid = fopen(fileP,'r');
if (fid == 1)
    h = errordlg(['Não foi possível encontrar o arquivo de Parâ
    metros.'...
                  ' Certifique se que o arquivo está na pasta e
    tente novamente']);
    uiwait(h)
    close(gcf);
```

```
    return
end

line = textscan(fid,'%s %n',5);
textscan(fid,'%s',1);
textscan(fid,'%s %s %s',1);
param_Hist = textscan(fid,'%s %n',8);
textscan(fid,'%s %s %s',1);
param_FFT = textscan(fid,'%s %n',3);
fclose(fid);

% Com o arquivo lido, podemos atribuir os parâmetros de teste

% Parametros gerais
N = line{2}(1);
fs = line{2}(2)*10^3;
Amax_input = line{2}(4);
Amin_input = line{2}(3);
NFFT_txt = line{2}(5);

% Parametros do histograma
for i = 1:8
    eval([param_Hist{1}{i} '=' num2str(param_Hist{2}(i),'%3.12f')
    ';']);
end

% Parametros da FFT
for i = 1:3;
    eval([param_FFT{1}{i} '=' num2str(param_FFT{2}(i),'%3.12f') '
   ;']);
end
```

```
%#################################################################
%
%       ANÁLISE DE PARÂMETROS ESTÁTICOS
%
%    Nesta seção, analisaremos os parametros estáticos
%    utilizando o método dos histogramas. Encontramos
%    o Ganho, Offset, DNL, INL e Falha de Códigos.
%
%#################################################################

Amax = Vmax_h;
Amin = Vmin_h;

TwoToNminus1 = 2^N 1;
TwoToN = 2^N;

try
    samples_bin = dlmread(arquivoHistograma,',');
catch err
    display(err.identifier)
    if(strcmp(err.identifier,'MATLAB:dlmread:FileNotOpened'))
        h = errordlg(['O arquivo com as amostras do histograma nã
    o foi encontrado.'...
                      ' Certifique se que o arquivo está na
    pasta e tente novamente']);
        uiwait(h)
        close(gcf);
        return
    end
```

```
end


%%%%%%%%%%%% CÁLCULO DO HISTOGRAMA %%%%%%%%%%%%

H = hist(samples_bin,0:TwoToNminus1);
H_final = zeros(1,TwoToNminus1);
for i=1:2^N
    H_final(i) = sum(H(i,:));
end

% Agora estimamos os níveis de transição.
S = numel(samples_bin);
Testimate = zeros(1,TwoToNminus1);
ch = 0;
d = (Amax+Amin)/2;
A = (Amax Amin);
for i=1:TwoToNminus1
    ch = H_final(i)+ch;
    Testimate(i) = d    A/2*cos(pi*ch/S);
end

% Exibindo o número de amostras
set(handles.t_numSamples,'String',int2str(S));

%%%%%%%%%%%%%%%% CALCULO DA DNL %%%%%%%%%%%%%%%%%%%

% W é a largura dos degraus.
W = Testimate(2:TwoToNminus1)  Testimate(1:TwoToNminus1 1);
% V é a escala nominal reduzida
V = Testimate(TwoToNminus1)    Testimate(1);
```

```
% Qavg é o tamanho médio dos degraus.
Qavg = V/(2^N 2);
% DNL é calculada como na formula apresentada no trabalho
DNL = (W    Qavg)/Qavg;

%%%%%%%%%%%%%%%%%%%% CÁLCULO DA INL %%%%%%%%%%%%%%

% Utilizando o 1242 2000, obtemos ganho e offset com eq (73) e
    (74)
% Encontrando os somatórios
sum_Tk = sum(Testimate);
sum_k_Tk = sum((1:TwoToNminus1).*(Testimate));
sum_Tk_2 = sum(Testimate.^2);

% Ganho é dado por:
Gain = Qavg*TwoToNminus1*(sum_k_Tk    2^(N 1)*sum_Tk)/((
    TwoToNminus1*sum_Tk_2) (sum_Tk^2));

% Offset é dado por
Offset = Testimate(1) + Qavg*(2^(N 1)    1)    Gain/TwoToNminus1*
    sum_Tk;

% Calculando a INL
residue = Qavg*(0:(TwoToNminus1 1)) + Testimate(1)    Gain*
    Testimate    Offset;
INL = residue/Qavg;
max_INL = max(abs(INL));

% Calculando falha de códigos
Missing_Codes = 0;
epslon = 1e 20;
```

```
for  i=1:TwoToNminus1 1
     if(DNL(i) < ( 1 + B + epslon)) % erro numérico
          Missing_Codes = Missing_Codes + 1;
     end
end

% Com todos estes cálculos, os parâmetros estaticos foram todos
    calculados.
% Enviando os parâmetros que serão exportados para o Workspace
% DNL, INL, Histograma, Ganho, Offset, Falha de Códigos

setappdata(0,'DNL',DNL);
setappdata(0,'INL',INL);
setappdata(0,'Hist',H_final);
setappdata(0,'G',Gain);
setappdata(0,'Vos',Offset);
setappdata(0,'MissCodes',Missing_Codes);

%%%%%%%%%%%%%%% EXIBINDO OS DADOS %%%%%%%%%%%%%%%%%%%

% Exibindo ganho, offset e falha de código
set(handles.t_Gain,'String',num2str(Gain,8));
set(handles.t_Offset,'String',num2str(Offset));
set(handles.t_MissingCode,'String',num2str(Missing_Codes));

% Plotando o histograma dos dados
axes(handles.histogramAxes);
bar(0:TwoToNminus1,H_final);
axis([ 1  TwoToN  0  max(H_final)]);
xlabel('Código');
ylabel('Número de Amostras');
```

```
title('Histograma dos Dados','FontSize',14);

%Plotando a DNL
axes(handles.dnlAxes);
plot(DNL);
max_DNL = max(abs(DNL));
axis([0 TwoToNminus1 1.2*max_DNL 1.2*max_DNL]);
xlabel('Código');
ylabel('V_{LSB}');
title('DNL','FontSize',14);

% Exibindo o DNL máximo
set(handles.maxDNL_text,'String',num2str(max_DNL));

%Plotando a INL
axes(handles.inlAxes);
plot(INL);
axis([0 TwoToNminus1 1.2*max_INL 1.2*max_INL]);
xlabel('Código');
ylabel('V_{LSB}');
title('INL','FontSize',14);

% Exibindo o DNL máximo
set(handles.t_maxINL,'String',num2str(max_INL));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           ANÁLISE DE PARÂMETROS DINÂMICOS
%
%    Nesta parte, iremos utilizar o método da FFT para
%    encontrar diversos paramêtros, como SNR, SINAD,
```

```
%    SNFD e ENOB.
%
%#########################################################################

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TENTATIVA DE ABERTURA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
try
     amostras_fft = dlmread(arquivoFFT,',');
catch err
     display(err.identifier)
     if(strcmp(err.identifier,'MATLAB:dlmread:FileNotOpened'))
          h = errordlg(['O arquivo com as amostras do teste da FFT
    não foi encontrado.'...
                        ' Certifique se que o arquivo está na
    pasta e tente novamente']);
          uiwait(h)
          close(gcf);
          return
     end
end

NFFT = numel(amostras_fft)/3; % = NFFT_txt;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONDICIONAMENTO DO SINAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Vamos iniciar a condicionar o sinal para fazer a análise da FFT
    .
```

```
% Primeiro, iremos transformar um sinal de 0 a 2^N 1 para de  1 a
    1.
cod_max = TwoToNminus1;
sinal_fft = (amostras_fft    cod_max/2)/(cod_max/2);

% Agora o sinal_fft vai de  1 a 1. Iremos então procurar as
   passagens pelo
% zero, para obter um sinal que comece no zero e termine no zero,
    aprox.

passagemZero = 0;

for i=1:numel(sinal_fft) 1
    if(sinal_fft(i) < 0)
        if(sinal_fft(i+1) > 0)
            passagemZero = [passagemZero i+1];
        end
    end
end

num_PZero = numel(passagemZero);

% Vamos pegar uma parte aproximadamente no meio da amostragem.
inicio_fft = passagemZero(ceil(num_PZero/3));

input_fft = sinal_fft(inicio_fft:(inicio_fft+NFFT 1));

% Como é bastante dificil obter um sinal perfeitamente encaixado
   na FFT,
% iremos multiplicá lo por uma janela de Hanning
Din = input_fft.*hanning(NFFT).';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALCULO DA FFT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Dout = fft(Din)/NFFT;
Dout_abs = 20*log10(abs(Dout(1:NFFT/2+1)));
Dout_plot = Dout_abs max(Dout_abs); % em dBfs
Fgrid = fs/2*linspace(0,1,NFFT/2+1);
Pow_Spectrum = abs(Dout(1:NFFT/2+1)).*abs(Dout(1:NFFT/2+1));

% Obtendo os parâmetros: fs  freq. de amostragem

% Procurando a frequencia com maior amplitude  a fundamental

maxdB = max(Dout_abs);
fSignal_bin = find(Dout_abs == maxdB);
fSignal = (fSignal_bin 1)/NFFT*fs; % 1 pois o MATLAB inicia com
    o indice 1.

% fSignal é a frequencia fundamental, e a bin é o ponto da FFT.

span_s = min(5,NFFT*0.005);
span_h = 2;

% A potencia do sinal é a integral do espectro do sinal. Aqui é
    considerado
% 0.5% de cada lado do ponto máximo como sinal.
Pow_Signal = sum(Pow_Spectrum(fSignal_bin span_s:fSignal_bin+
    span_s));
```

```
% A potencia DC é 0.5% de cada lado da Bin 1
Pow_DC = sum(Pow_Spectrum(1));

% A potencia dos harmonicos define a potencia das distorçoes. Pra
    isso
% devemos localizar os harmonicos.

harmFreq = [];
harmPow = [];

for harmNumber = 1:10 %1o harm   fundamental
    harmBin = harmNumber.*(fSignal_bin 1) +1; %Localizaçao de
   cada harmonico
    if harmBin > NFFT/2 % Rebatido pelo aliasing
        harmBin = NFFT   harmBin;
    end
    tone = (harmBin 1)/NFFT*fs;
    harmFreq = [harmFreq tone];
    harmPow = [harmPow sum(Pow_Spectrum(harmBin 1:harmBin+1))];
end

% A potencia da distorção eh então a soma de todas menos a
   fundamental
Pow_Dist = sum(harmPow(2:max(harmNumber)));

% A potencia do ruido é todo o resto
Pow_Noise = sum(Pow_Spectrum(1:NFFT/2+1))   Pow_DC   Pow_Signal;%
      Pow_Dist;

% Calculando o harmonico com maior potência
harmPowMax = max(harmPow(2:max(harmNumber)));
```

```
% Com esses valores, sao calculados alguns parâmetros

SNR = 10*log10(Pow_Signal/Pow_Noise);
SINAD = 10*log10(Pow_Signal/(Pow_Noise+Pow_Dist));
ENOB = (SINAD 10*log10(3/2))/(20*log10(2));
THD = sqrt(Pow_Dist/Pow_Signal);
SFDR = 10*log10(harmPow(1)/harmPowMax);

% Com todos estes cálculos, os parâmetros estaticos foram todos
    calculados.
% Enviando os parâmetros que serão exportados para o Workspace
% FFT, fs, fSignal, SNR, SINAD, ENOB, NFFT, THD, SFDR

setappdata(0,'Dout',Dout);
setappdata(0,'fs',fs);
setappdata(0,'fin',fSignal);
setappdata(0,'SNR',SNR);
setappdata(0,'SINAD',SINAD);
setappdata(0,'THD',THD);
setappdata(0,'ENOB',ENOB);
setappdata(0,'SFDR',SFDR);
setappdata(0,'NFFT',NFFT);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EXIBINDO OS PARÂMETROS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plotando FFT
axes(handles.snrAxes);
plot(Fgrid,Dout_plot);
```

```
fs_2 = max(Fgrid);
axis([0 fs_2 min(Dout_plot) max(Dout_plot)]);
xlabel('Frequência (Hz)');
ylabel('|D_{out}| (dBFS)');
title('FFT','FontSize',14);
set(gca,'XTick',0:fs_2/4:fs_2 )
set(gca,'XTickLabel',{'0','fs/8','fs/4','3fs/8','fs/2'})

% Exibindo parâmetros
if (fs> 1000)
    fs_display = fs/1000;
    set(handles.t_fs,'String',sprintf('%3.5f kHz',fs_display));
else
    fs_display = fs;
    set(handles.t_fs,'String',sprintf('%3.5f Hz',fs_display));
end

if (fSignal > 1000)
    f_display = fSignal/1000;
    set(handles.t_fSignal,'String',sprintf('%3.5f kHz',f_display)
    );
else
    f_display = fSignal;
    set(handles.t_fSignal,'String',sprintf('%3.5f Hz',f_display))
    ;
end

set(handles.t_SNR,'String',strcat(num2str(SNR,'%3.5f'),' dB'));
set(handles.t_SINAD,'String',strcat(num2str(SINAD,'%3.5f'),' dB')
    );
set(handles.t_SFDR,'String',sprintf('%3.5f dB',SFDR));
```

```
set(handles.t_ENOB,'String',strcat(num2str(ENOB,'%3.5f'),' Bits')
    );
set(handles.t_THD,'String',strcat(num2str(THD*100,'%3.5f'),' %'))
    ;
set(handles.t_NFFT,'String',sprintf('%i Pontos',NFFT));

% Exibindo apenas o painel de param. estaticos
set(handles.dynamicPanel,'Visible','off');
set(handles.staticPanel,'Visible','on');

%     Executes on button press in exportButton.
function exportButton_Callback(hObject, eventdata, handles)
% hObject    handle to exportButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
   MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Obter identificador para os resultados
prompt = 'Entre com um identificador para as medidas:';
dlg_title = 'Exportar Resultados';
num_lines = 1;
def = {'ADC Interno 18F'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
identifier = answer{1};

% Criar pasta para colocar arquivos

fName = mfilename;
Path = mfilename('fullpath');
pasta_resultados = strrep(Path,fName,'Resultados\');
```

```
hora_medida = clock;
str_data = sprintf('%i %02i %02i  %02ih%02im%02is',hora_medida(1),
    hora_medida(2),...
                        hora_medida(3),hora_medida(4),hora_medida(5),
    round(hora_medida(6)));

pasta_export = [pasta_resultados str_data '    ' identifier];

mkdir(pasta_export);

% Obtendo os parâmetros em background
INL = getappdata(0,'INL');
DNL = getappdata(0,'DNL');
Hist = getappdata(0,'Hist');
G = getappdata(0,'G');
Vos = getappdata(0,'Vos');
MissCodes = getappdata(0,'MissCodes');
Dout = getappdata(0,'Dout');
fs = getappdata(0,'fs');
fin = getappdata(0,'fin');
SNR = getappdata(0,'SNR');
SINAD = getappdata(0,'SINAD');
THD = getappdata(0,'THD');
ENOB = getappdata(0,'ENOB');
SFDR = getappdata(0,'SFDR');
NFFT = getappdata(0,'NFFT');


N = log2(numel(Hist));
max_INL = max(INL);
max_DNL = max(DNL);
```

```
% PLOTANDO E EXPORTANDO %%%%%%%%%%%%%%%%%%%%

h = figure(1);
set(h,'Visible','off');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Exportando o gráfico
plot(INL);
axis([0 numel(INL) 1.2*max_INL 1.2*max_INL]);
set(gca,'YGrid','on','GridLineStyle',' ');

xlabel('ADC Code','FontSize',12);
ylabel('INL (LSB)','FontSize',12);

%saveas(h,'INL.eps','epsc');
INL_path = [pasta_export '\' 'INL.eps'];
print(h,'-depsc',INL_path);

% Exportando ponto a ponto
dlmwrite([pasta_export '\' 'INL.dat'],INL,',');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DNL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Exportando o gráfico
plot(DNL);
```

```
axis([0 numel(DNL)  1.2*max_DNL 1.2*max_DNL]);
set(gca,'YGrid','on','GridLineStyle',' ');

xlabel('ADC Code','FontSize',12);
ylabel('DNL (LSB)','FontSize',12);

DNL_path = [pasta_export '\' 'DNL.eps'];
print(h,' depsc',DNL_path);

% Exportando ponto a ponto
dlmwrite([pasta_export '\' 'DNL.dat'],DNL,',');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HISTOGRAMA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Exportando o gráfico
bar(0.5:2^N 0.5 ,Hist);
axis([0 2^N 0 max(Hist)]);

xlabel('ADC Code','FontSize',12);
ylabel('# of Samples','FontSize',12);

Hist_path = [pasta_export '\' 'Histograma.eps'];
print(h,' depsc',Hist_path);

% Exportando ponto a ponto
dlmwrite([pasta_export '\' 'Histograma.dat'],Hist,',');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FFT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Fgrid = fs/2*linspace(0,1,NFFT/2+1);
Dout_abs = 20*log10(abs(Dout(1:NFFT/2+1)));
Dout_plot = Dout_abs max(Dout_abs); % em dBfs

plot(Fgrid,Dout_plot);
fs_2 = max(Fgrid);
axis([0 fs_2 min(Dout_plot) max(Dout_plot) 10]);
set(gca,'YGrid','on','GridLineStyle',' ');

xlabel('Frequency (Hz)','FontSize',12);
ylabel('Amplitude (dBFS)','FontSize',12);

FFT_path = [pasta_export '\' 'FFT.eps'];
print(h,' depsc',FFT_path);

% Exportando ponto a ponto
a = [Fgrid.' Dout_plot.'];
dlmwrite([pasta_export '\' 'FFT.dat'],a,'delimiter',',','
    precision','%.6f','newline','pc');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ARQUIVO RELATÓRIO %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Criando um txt com todos os parâmetros

report_path = [pasta_export '\' 'Report.txt'];

report_file = fopen(report_path,'w');
```

```
if (report_file == -1)
    h = errordlg('Não foi possível criar o relatório de dados.');
    uiwait(h)
    fclose(report_file);
    close(gcf);
    return
end

% Imprimindo um cabeçalho
fprintf(report_file,[    '
   ##############################################\r\n'...
                        '#   Relatório da Plataforma de Teste
    #\r\n'...
                        '#   Autor: Eduardo V. P. dos Anjos
    #\r\n'...
                        '
   ##############################################\r\n\r\n']);
fprintf(report_file,'Medida feita em %02i/%02i/%i às %02i:%02i
   :%02i\r\n\r\n',hora_medida(1),hora_medida(2),...
                    hora_medida(3),hora_medida(4),hora_medida(5),
   round(hora_medida(6)));

fprintf(report_file,'### Parâmetros Estáticos ##################\
   r\n\r\n');
fprintf(report_file,'DNL (máxima): %3.12f LSB\r\n',max_DNL);
fprintf(report_file,'INL (máxima): %3.12f LSB\r\n',max_INL);
fprintf(report_file,'Ganho: %3.12f\r\n',G);
fprintf(report_file,'Offset: %3.12f V\r\n',Vos);
fprintf(report_file,'Falha de Códigos: %01i\r\n\r\n',MissCodes);
fprintf(report_file,'### Parâmetros Dinâmicos ##################\
   r\n\r\n');
```

```
fprintf(report_file,'FFT feita com %i pontos\r\n',NFFT);
fprintf(report_file,'Freq. de Amostragem: %3.12f Hz\r\n',fs);
fprintf(report_file,'Freq. da Fundamental: %3.12f Hz\r\n',fin);
fprintf(report_file,'SNR: %3.12f dB\r\n',SNR);
fprintf(report_file,'SINAD: %3.12f dB\r\n',SINAD);
fprintf(report_file,'ENOB: %3.12f Bits\r\n',ENOB);
fprintf(report_file,'THD: %3.12f %%\r\n',THD);
fprintf(report_file,'SFDR: %3.12f dB\r\n\r\n',SFDR);

fclose(report_file);

h = msgbox('Os resultados foram exportados com sucesso!','Aviso')
    ;
uiwait(h);
```

PT_ADC_Analysis.m

Apêndice C

Código-Fonte do Firmware do Microcontrolador

Neste Apêndice são apresentados os códigos-fonte dos programas feitos no MATLAB.

C.1 *Firmware* do Microcontrolador

```
//        pt_firmware.c
//
//  Este arquivo é o firmware da plataforma de testes para
//  conversores AD, um projeto final do curso de
//  Engenharia Eletrônica e de Computação.
//
//  Autor: Eduardo Vilela Pinto dos Anjos
//
//  Título de Projeto: Plataforma de Testes para Conversores
//                     Analógico Digitais
//

#include <18F4550.h>
```

```
#device PASS_STRINGS = IN_RAM
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,
    VREGEN,NOMCLR


#use delay(clock=48000000)

//Definindo alguns pinos especiais.
#define GREEN_LED       PIN_C0  // LED Verde
#define RED_LED         PIN_C1  // LED Vermelho
#define RD_PHASE        PIN_A4  // Fase de Leitura
#define ADC_INTERNAL    PIN_A0  // Pino do conversor interno.
#define ADC_VrefP       PIN_A3
#define ADC_VrefM       PIN_A2




#include <stdlib.h>

#include <usb_cdc.h>

#include <string.h>

#include <math.h>

// Funçao da usb_cdc.h que foi modificada por conta de retornar
    os caracteres recebidos
// toda vez que recebia uma string, ocupando o buffer
#byte TRISB = 0xF93
#byte PORTB = 0xF81
```

```
#byte TRISD = 0xF93
#byte PORTD = 0xF81 // 8 bits menos significativos
#byte TRISA = 0xF92

// PARA O CONVERSOR INTERNO
    ////////////////////////////////////////////////////
#byte ADCON0 = 0xFC2
#byte ADCON1 = 0xFC1
#byte ADCON2 = 0xFC0
#byte ADRESH = 0xFC4
#byte ADRESL = 0xFC3

#bit RD_PHASE_INT = ADCON0.1


void get_string_PT(char* s, unsigned int max) {
    unsigned int len;
    char c;
      max;
    len=0;
    do {
      c=usb_cdc_getc();
      if(c==8) {   // Backspace
          if(len>0) {
            len   ;
          }
      } else if ((c>=' ')&&(c<='~'))
        if(len<max) {
          s[len++]=c;
        }
    } while(c!=13);
```

```
    s[len]=0;
}

void PT_Init(void){
    // Configurando registradores de I/O
    TRISB = 0xFF;
    TRISD = 0xFF;
    TRISA = 0x0F;

}

void Internal_ADC_Init(void){
    TRISA = 0xFF;
    ADCON0 = 0b00000001;
    ADCON1 = 0b00111011; // GND  VDD  Bits analógicos de A0 a A3
    ;
    ADCON2 = 0b00010110;
}

int leitura,envio;
unsigned int8 saida = 0;
unsigned int16 teste = 0;
unsigned long j;


#INT_AD
void AD_ISR(void){
    clear_interrupt(int_ad);
    leitura = 1;
}
```

```
#INT_TIMER1
void TIMER1_isr(void){
    set_timer1(63136);
    RD_PHASE_INT= 1;
    output_low(pin_a5);
    output_high(pin_a5);
    delay_us(1);
    output_low(pin_a5);
}

#INT_EXT2
void EXT2_isr(void){
    output_high(pin_b5);
    printf(usb_cdc_putc_fast,"%c",input_d());
    j++;
    output_low(pin_b5);
}

void main(void)
{
    char buffer[10];
    char seletor_RD[5];
    char seletor_ADC[5];
    signed int N;
    signed long NFFT,M,N_points;
    unsigned long long R,i; // R pode ser muito grande!

    //PT_Init();
    usb_cdc_init();
    usb_init();
    output_low(GREEN_LED);
```

```
output_low(RED_LED);

enable_interrupts(global);
//enable_interrupts(int_ad);
//enable_interrupts(int_timer1);
leitura = 0;
envio = 0;
while(!usb_enumerated()){output_high(RED_LED);}

while(!usb_cdc_connected()){}

output_high(GREEN_LED);
output_low(RED_LED);

while(!usb_cdc_kbhit()){}

// Obtendo Parâmetros de Teste
////////////////////////////////////////////////////////
// Ordem:
// N, NFFT, seletor_ADC, seletor_RD, R, M

get_string_PT(buffer,10); // Primeiro recebe N
N = atoi(buffer);
get_string_PT(buffer,10); // 2o eh NFFT
NFFT = atol(buffer);
get_string_PT(buffer,10); // Seletor de fase de leitura. Ou
high, ou low.
strncpy(seletor_RD,buffer,5);
get_string_PT(buffer,10); // Escolhe entre ADC interno e
externo
strncpy(seletor_ADC,buffer,5);
```

```
 get_string_PT(buffer,10); // R é o número de conv.
 R = atoi32(buffer);
 get_string_PT(buffer,10); // M o de amostras por conv.
 M = atol(buffer);

 // Indicar que tudo foi recebido e começar os trabalhos.
 printf(usb_cdc_putc,"OK\r\n");

if(strcmp(seletor_ADC,"INT") == 0){
  Internal_ADC_Init();

    // Executar primeiro o teste dos histogramas
  for(i=0; i<R; i++){
    get_string_PT(buffer,10);
    enable_interrupts(int_timer1);
    set_timer1(63136);
    if(strcmp(buffer,"ST_HIST") == 0){}
    for(j=0;j<M;){
      if(leitura){
      printf(usb_cdc_putc,"%c",ADRESH);
      j++;
      leitura = 0; // Conversão foi lida
      }
    }
    disable_interrupts(int_timer1);
  }

    // Depois o da FFT.
    N_points = 3*NFFT;
    get_string_PT(buffer,10);
    if(strcmp(buffer,"ST_FFT") == 0){
```

```
        output_high(RED_LED);
    }
    enable_interrupts(int_timer1);
    set_timer1(63136);
    for(j=0;j<N_points;){
        //RD_PHASE_INT = 1;
        //while(RD_PHASE_INT == 1){}
        //printf(usb_cdc_putc,"%c%c",ADRESL,ADRESH);
        if(leitura){
            printf(usb_cdc_putc,"%c",ADRESH);
            j++;
            leitura = 0; // Conversão foi lida
        }
    }
    disable_interrupts(int_timer1);
}
else if(strcmp(seletor_ADC,"EXT") == 0){
    // Primeiro histograma
    for(i=0; i<R; i++){
        get_string_PT(buffer,10);
        j = 0;
        if(strcmp(buffer,"ST_HIST") == 0){
            enable_interrupts(INT_EXT2_L2H);
              // usb_cdc_init(); ou usb_init();
        }
        while(j<M) {}
        disable_interrupts(INT_EXT2_L2H);
    }
    // Depois FFT
    N_points = 3*NFFT;
    get_string_PT(buffer,10);
```

```
        j = 0;
        if(strcmp(buffer,"ST_FFT") == 0){
            enable_interrupts(INT_EXT2_L2H);
        }
        while(j<N_points) {}
        disable_interrupts(INT_EXT2_L2H);
      }
    else{
      // Para debugar. Não pode entrar aqui.
      }

     while(TRUE){}
}
```

PT_Firmware.c