DESENVOLVIMENTO E APLICAÇÃO O DE UM AMPLIFICADOR LOCK-IN BASEADO EM DSP

Rafael Astuto Arouche Nunes

PROJETO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Aprovada por:

_____
Prof. José Luiz da Silva Neto, Ph.D.
(Orientador)

_____
Prof. Marcelo Portes de Albuquerque, Ph.D.
(Co-Orientador)

_____
Prof. Sergio Sami Hazan, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2009

Dedicatória

Quando pensei em para quem dedicar esta obra, não tive dúvidas, seria para meu irmão! Bruno Astuto Arouche Nunes, ou melhor, apenas Bruno! Sempre comigo, sempre me ajudando, me apoiando, ao meu lado em todos os momentos importantes da minha vida. E durante meu projeto final não poderia ser diferente.

Quero dedicar todo este feito à você Bruno, que mais do que um irmão mais velho é meu melhor amigo, um companheiro e tanto!

Obrigado por tudo que você é na minha vida, te amo!

Agradecimentos

Primeiramente agradeço à Deus.

Agradeço de forma extremamente especial aos meus pais Marcos Arouche Nunes e Miriam Cristina Astuto por todo apoio fornecido durante minha jornada acadêmica. Sem eles nada do que sou e nenhuma conquista em minha vida seria possível. Sempre estiveram ao meu lado me apoiando e me incentivando, mas ao mesmo tempo me guiando e me mostrando o melhor caminho em todas minhas decisões. Esta obra só foi possível graças à existência de pais maravilhosos como estes, que com toda certeza do mundo posso afirmar e glorificar: melhores não há!

É com imenso orgulho que reforço de maneira carinhosa e especial um agradecimento ao meu pai, que nunca me disse o que fazer, mas sempre me mostrou o melhor caminho a seguir. Obrigado pai, essa obra é para você!

Com grande amor, agradeço e dedico este projeto também à minha namorada Daniela que durante estes quase sete anos juntos, sempre me apoiou e esteve ao meu lado em todas as situações, nos momentos bons e ruins, nas horas de alegria e de tristeza. Amor, que esta seja apenas a primeira vitória juntos, pois ao lado de Deus tenho certeza de que muitas conquistas ainda iremos alcançar! Você também faz parte deste projeto! Sem sua ajuda incondicional nada disso teria se realizado! Te amo!

Agradeço ao meu orientador de iniciação tecnológica, Marcelo Portes de Albuquerque, que esteve comigo durante quase quatro anos me apoiando e muito me ensinando. De fato um mestre, ou melhor, um doutor, com sabedoria ímpar e carisma incomum na área. Espero ainda poder realizar muitas conquistas ao lado deste, que mais do que meu professor, tornou-se um grande amigo.

Agradeço ao professor Zé Luiz que acreditou no potencial do projeto do Lock-In e me deu um voto de confiança a aceitá-lo como tema do meu projeto final.

Obrigado Shigueo pela ajuda durante o projeto final na fabricação dos sensores ultra-som.

Obrigado ao CNPq pela ajuda financeira fornecida durante quase quatro anos de bolsa de iniciação tecnológica no CBPF.

Obrigado a UFRJ pelos seis anos de ensino de extrema qualidade.

Índice

Lista de Figuras

Lista de Tabelas

Introdução

Tema

Este trabalho trata sobre o estudo de Processadores Digitais de Sinais (DSP), visando aprofundar-se em suas aplicações, a fim de realizar experiências científicas e construção de instrumentos eletrônicos de medição, com o intuito de por em prática teorias matemáticas e físicas previamente estudadas.

Para realização deste trabalho foi de suma importância a utilização de conhecimentos teóricos provindos das áreas de Instrumentação, Eletrônica, Física e Matemática, os quais contribuíram para elaboração dos algoritmos utilizados no trabalho.

A primeira etapa do trabalho se deu na escolha do DSP mais adequado para realização do projeto. O protótipo elaborado foi feito baseado na plataforma do *kit* de desenvolvimento EP2S60 da *Altera Devices*, descrito com mais detalhes no Capítulo 2.

No segundo estágio foi elaborado um algoritmo para simular o funcionamento de um *Lock-In*. O intuito deste algoritmo foi de mostrar como o *Lock-In* consegue observar sinais imersos em ruídos e extrair com precisão sua fase e magnitude. Com o intuito de ilustrar seu funcionamento também foram elaborados simuladores gráficos em MATLAB.

No terceiro estágio do trabalho foi implementado no DSP EP2S60 um protótipo de *Lock-In* baseado nos algoritmos anteriormente desenvolvidos. Este protótipo será detalhadamente descrito no capítulo 2.

Justificativa

Tem sido crescente a demanda por equipamentos portáteis (como GPS, celulares e instrumentos de medição), de fácil manuseio e que tenham uma interface simples com o usuário. A utilização de DSP procura unir todos esses preceitos de forma a se tornar uma ferramenta de trabalho das mais completas atualmente.

Com suas mais diversas possibilidades de uso, nas mais abrangentes áreas, os processadores digitais de sinais são muito versáteis, podendo se enquadrar em quaisquer tipos de projetos. Vários ramos da indústria civil e militar utilizam DSP em seus equipamentos. Realizar projetos de pesquisa e aprendizado utilizando estes processadores é fundamental para se desenvolver tecnologia, o que torna o DSP uma ferramenta estratégica nos laboratórios das universidades de todo o mundo.

Objetivos

Este trabalho tem por objetivo a construção de um protótipo de um Amplificador *Lock-in* (descrito com detalhes no Capítulo 1), e sua validação através de experiências físicas. A elaboração do projeto é baseada na teoria de sinais, que define com precisão todo o processo de transformação e detecção do sinal realizado por um *Lock-in*. Essa detecção consiste na técnica de medida de um sinal de baixa amplitude corrompido por ruído (aleatório e periódico), baseado na detecção síncrona, que corresponde ao processo de demodulação síncrona (Figura 1), explicado no Capítulo 1.

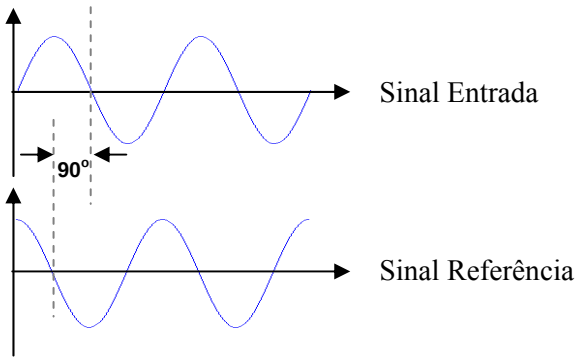


Figura 1: Princípio da detecção síncrona (ou demodulação) para medir o atraso de um sinal de referência em relação a um sinal de um sistema físico.

A construção de um *Lock-in* pode ser feita utilizando-se princípios e técnicas da eletrônica clássica (analógica ou digital) em conjunto com DSPs e softwares específicos, que irão controlar e manipular os conversores analógico-digitais (ADC) e digital-analógicos (DAC).

Neste trabalho, foi desenvolvimento um protótipo de um Amplificador *Lock-in* baseado em *kits* DSP de empresas comerciais. O processo abrange desde a pesquisa pelo equipamento adequado, até o desenvolvimento de algoritmos de testes e simuladores.

Metodologia

O início da elaboração do protótipo do Amplificador *Lock-in* se deu desenvolvendo a teoria matemática e física necessária à aplicação. Baseado nisso, foram criados diversos algoritmos, implementados no *software* matemático MATLAB, a fim de realizar sua validação.

A comprovação teórica foi importante para a criação de simuladores, que proporcionaram uma interface gráfica para melhor visualização e entendimento do processo físico envolvido. Com isto, foi possível desenvolver um protótipo, com o intuito de validar os resultados obtidos nas simulações.

Descrição

O Capítulo 1 traz à luz as teorias que sustentam todo funcionamento de um *Lock-In*, bem como seus fundamentos matemáticos e físicos. São apresentadas diferenças entre um *Lock-In* digital e um analógico, focando-se em retratar as características do detector sensível de fase, o PSD.

No segundo capítulo deste trabalho são mostradas as etapas de elaboração de um simulador *Lock-In* feito em Matlab, inclusive com interface gráfica. Neste capitulo o usuário consegue, manipulando o simulador, analisar de que forma um ruído interfere na detecção de fase e magnitude, bem como saber qual a quantidade mínima de pontos por período seriam necessários obter do sinal de entrada, a fim de se conseguir uma boa precisão nos cálculos, para uma eventual experiência.

O capítulo 3 apresenta a implementação de um protótipo de *Lock-In* feito em interface DSP através do kit de desenvolvimento Altera Stratix® II EP2S60. Todo processo foi estruturado através da comunicação da placa com o software Matlab, que trabalhando em conjunto com sua IDE Quartus II, possibilitou a construção do protótipo.

No quarto capítulo são apresentados resultados de medições reais feitas com o DSP programado com o algoritmo do *Lock-In*. Os testes foram feitos de forma a comprovar a robustez do código, que trabalhando com sinais de alta freqüência forneceu medidas muito precisas, com baixo desvio padrão.

O quinto capítulo apresenta uma aplicação prática para o *Lock-In* implementado no kit da Altera. Através da utilização de sensores de ultra-som de 40 kHz de freqüência de ressonância, montou-se um esquema para medição de espessura de amostras de alumínio.

No Apêndice A é apresentado o algoritmo base do *Lock-In* e no Apêndice B o código fonte do simulador feito no Matlab. No Apêndice C serão abordados os princípios e teorias a respeito dos Processadores Digitais de Sinal (DSP), seu histórico, princípio de funcionamento, aplicações industriais e científicas e como pode ser utilizado para desenvolvimento de projetos. É mostrada também a estrutura interna de um DSP e alguns exemplos de programas que foram implementados nos ambientes de desenvolvimento de projetos.

Capítulo 1

O Amplificador Lock-In

1.1 – Introdução

O Processamento de Sinais, de uma forma geral, consiste na análise de sinais, extraindo informações que possam torná-los úteis para alguma aplicação específica. Esta é uma área que trabalha essencialmente com a modelagem matemática destes sinais, tornando-se a base para toda teoria desenvolvida na realização de seu processamento, seja apenas para sua compreensão, seja para construção de equipamentos ou instrumentos, que são de grande valia para verificação, validação ou detecção de fenômenos físicos.

Dentre as inúmeras possibilidades de trabalho, podemos citar o processamento de sinais de baixa amplitude. Isso se torna especialmente interessante quando se deseja realizar análises das propriedades elétricas ou magnéticas de algum material. Para isso é imprescindível a utilização de um Amplificador *Lock-In* para realização de medidas do sinal através da amostra

O Amplificador *Lock-In*, ou somente *Lock-In*, é um instrumento de grande importância em qualquer laboratório onde se realizam medidas físicas, isso por que este instrumento tem grande facilidade de detectar sinais imersos em ruídos, conseguindo com precisão detectar sua magnitude e fase.

A detecção de fase feita pelo *Lock*-In é feita através da técnica de Detecção Sensível de Fase (PSD)[1]. Esta técnica corresponde à principal função do amplificador, captando apenas o sinal de interesse para a medida, suprimindo o efeito de ruídos ou interferência de componentes ativos.

[1] PSD, do inglês *Phase Sensitive Detector*.

Para funcionar corretamente o PSD deve estar programado para funcionar em uma determinada banda de freqüência, ou seja, reconhecer apenas o sinal de interesse para a medida, eliminando freqüências indesejadas. Isto é feito fornecendo-se ao detector uma tensão de referência de mesma freqüência e com uma fase fixa relacionada ao sinal de entrada. O uso deste sinal de referência assegura que o instrumento irá rastrear qualquer mudança na freqüência do sinal que está sendo analisado. Desta característica que se deriva o nome do aparelho. Mais detalhes sobre o PSD serão tratados na seção 1.4 deste Capítulo.

Podemos entender melhor o funcionamento do *Lock-In* através da Figura 1.1 abaixo, onde percebemos como o sinal de entrada provindo de um sistema físico é analisado dentro do equipamento, fornecendo como saída sua medida de fase ou magnitude. O sinal do sistema físico passa primeiro por um ganho, pois geralmente o equipamento é utilizado para medição de sinais de baixíssima amplitude. Após isso se realiza a operação de multiplicação e acumulação dentro do PSD, fornecendo o resultado de fase desejado.

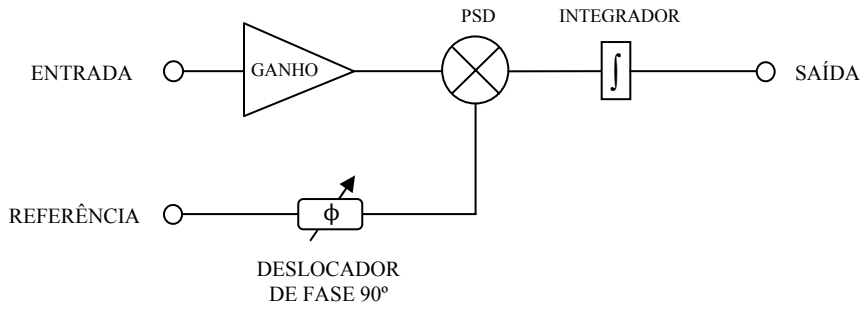


Figura 1.1 – Esquemático simplificado de um Amplificador *Lock-In*.

Através do esquema descrito acima percebemos que um *Lock-In* é essencialmente composto por três partes principais: o deslocamento do sinal de referencia em 90º, o PSD que realizará os cálculos matemáticos e o integrador. A fundamentação matemática detalhada das

etapas de cada uma destas partes será descrita com maior precisão na seção 1.3 deste Capítulo.

1.2 – Lock-In Digital

A criação de um *Lock-In* digital a partir do uso de DSP's fez com que a tarefa de implementar a teoria para criação do equipamento se tornasse muito mais fácil e rápida. Utilizar um DSP visa aumentar a precisão do equipamento, tendo em vista que por ser digital, se torna menos susceptível à interferências externas, principalmente em seus componentes ativos e passivos, como resistores, capacitores e indutores.

Na Figura 1.2 é apresentado um esquema completo de construção de um *Lock-In* digital comercial e analisá-lo a fim de compreender melhor sua estrutura física principal. Percebe-se da figura os principais componentes encontrados no desenvolvimento deste trabalho, como as duas entradas de sinal ("reference input" e "signal input"), o ganho do amplificador da entrada ("input amplifier"), o deslocador de fase ("phase shifter"), a geração da tabela com os valores de seno e cosseno do sinal de referência ("look-up table"), os multiplicadores em fase e em quadratura, correspondentes ao PSD e por fim as saídas de magnitude e fase.

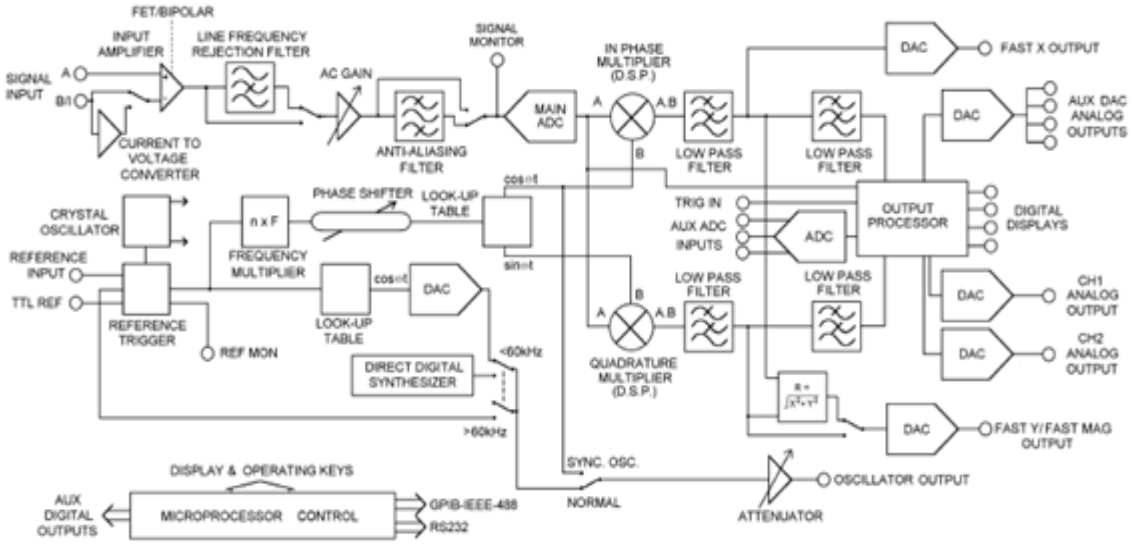


Figura 1.2 – Diagrama completo de construção de um *Lock-In* digital comercial.

Fonte: Signal Recovery [6]

1.3– Detecção Sensível à Fase

Para que haja o entendimento do funcionamento do *Lock-In* é necessário uma compreensão das operações realizadas internamente. O amplificador possui duas entradas, uma de referência e outra para o sinal a ser analisado. Desta forma ocorre uma multiplicação de ambos os sinais, cujo resultado fornece as componentes para o cálculo da fase e magnitude do sinal do sistema físico.

Supõe-se inicialmente um sinal senoidal de referência $V_{ref} = A.cos(\omega t)$, que pode ser gerado internamente pelo *Lock-In* ou fornecido externamente por outro equipamento, e um sinal de entrada $V_{in} = B.cos(\omega t + \theta)$, de mesma freqüência, porem com uma defasagem em relação à referência de θ. O processo de detecção consiste em multiplicar os dois sinais, em um processo de multiplicação e acumulação, gerando dois valores numéricos distintos, que serão usados nos cálculos de magnitude e fase, mostrados a seguir:

$$V_{PSD} = A.cos(\omega_o t).B.cos(\omega_o t + \theta)$$

$$= AB.\cos(\omega_o t).[cos(\omega_o t).cos(\theta) - sin(\omega_o t).sin(\theta)]$$

$$= AB.[cos^2(\omega_o t).cos(\theta) - cos(\omega_o t).sin(\omega_o t).sin(\theta)]$$

$$= AB.\{[{}^1/_2 + {}^1/_2\, cos(2\omega_o t)].cos(\theta) - {}^1/_2\, sin(2\omega_o t).sin(\theta)\}$$

$$= {}^1/_2\, AB.cos(\theta) + {}^1/_2\, AB.[cos(2\omega_o t).cos(\theta) - sin(2\omega_o t).sin(\theta)]$$

$$V_{PSD} = {}^1/_2\, AB.cos(\theta) + {}^1/_2\, AB.cos(2\omega_o t + \theta) \quad \text{Eq. 1}$$

Seja U_1 o valor médio de V_{PSD} para $T = {}^{2\pi}/\omega_o$:

$$U_1 = \frac{1}{T}\int_0^T V_{PSD}\,dt$$

$$= \frac{1}{T}\int_0^T \frac{AB}{2}cos(\theta)dt + \frac{1}{T}\int_0^T \frac{AB}{2}cos(2\omega_o t + \theta)dt$$

$$U_1 = \frac{AB}{2}cos(\theta) \quad \text{Eq. 2}$$

Analogamente podemos multiplicar o sinal de entrada V_{in} por $V_{ref_{90}}$, que corresponde ao sinal de referência defasado de 90 graus[2]. Sendo assim obtemos o seguinte valor para a saída do PSD:

$$V_{PSD} = A.sin(\omega t).B.cos(\omega t + \theta)$$

$$= AB.\sin(\omega t).[cos(\omega t).cos(\theta) - sin(\omega t).sin(\theta)]$$

$$= AB.[cos(\omega t).sin(\omega t).cos(\theta) - sin^2(\omega t).sin(\theta)] \quad \text{Eq. 3}$$

Sabendo-se que: $sin^2(\omega t) = \frac{[cos(2\omega t) - 1]}{2}$ Eq. 4

$$cos(\omega t).sin(\omega t) = \frac{sin(2\omega t)}{2} \quad \text{Eq. 5}$$

[2] Correspondente à *"look-up table"* indicada na Figura 3.2.1.

Substituindo Eq. 4 e Eq. 5 em Eq. 3, obtêm-se:

$$V_{PSD} = {}^{1}/_{2}\, AB.sin(\theta) + {}^{1}/_{2}\, AB.sin(2\omega t + \theta) \quad \text{Eq. 6}$$

Seja U_2 o valor médio de V_{PSD}:

$$U_2 = \frac{1}{T}\int_0^T V_{PSD}\,dt$$

$$= \frac{1}{T}\int_0^T \frac{AB}{2} sin(\theta)dt + \frac{1}{T}\int_0^T \frac{AB}{2} sin(2\omega t + \theta)dt$$

$$U_2 = \frac{AB}{2} sin(\theta) \quad \text{Eq. 7}$$

Desta forma obtemos os dois valores, U_1 e U_2 que são a chave para o cálculo da fase e da magnitude, no *Lock-In*. É interessante ainda destacar dois pontos importantes com relação à diferença de fase entre o sinal de entrada e a referência. Caso a defasagem entre os dois sinais seja de zero grau ($\theta = 0°$) teremos então que $U_1 = \frac{AB}{2}$ e $U_2 = 0$.

De forma similar podemos destacar o comportamento do PSD quando a defasagem entre os dois sinais é de 90 graus ($\theta = 90°$). Teremos então que $U_1 = 0$e $U_2 = \frac{AB}{2}$. Neste caso em especial o resultado da multiplicação no PSD, com o sinal defasado, é um sinal de saída com freqüência igual ao dobro da do sinal de entrada, porem com valor médio igual a zero. Esta situação fica mais bem ilustrada na Figura 1.3 abaixo.

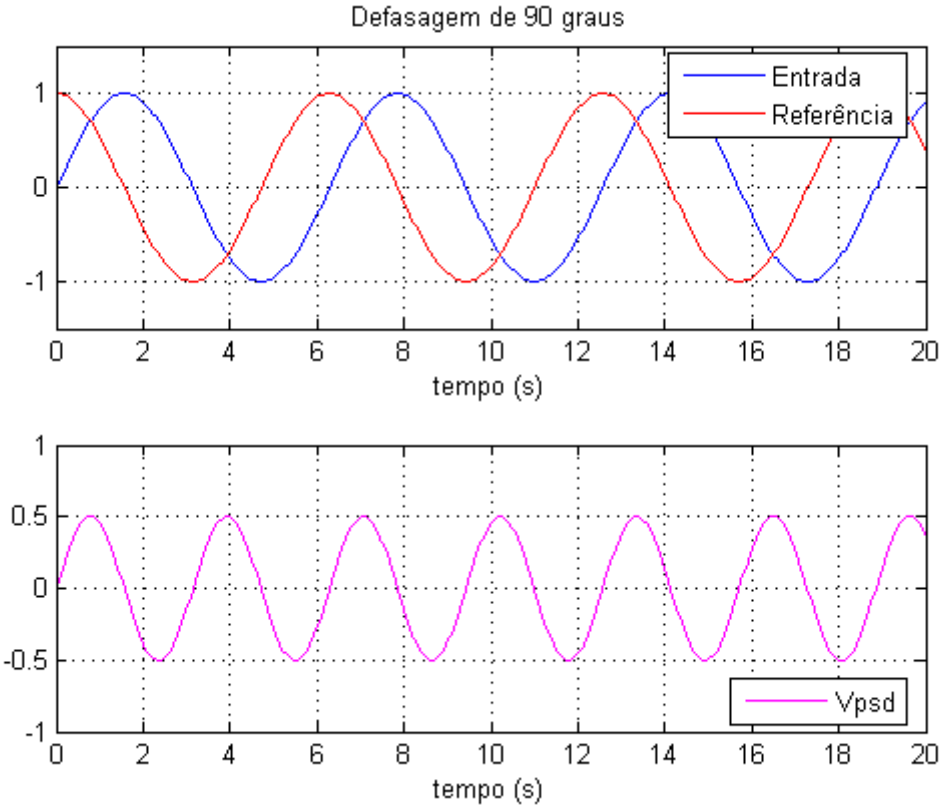


Figura 1.3 – Sinal de entrada defasado de 90 graus do sinal de referência, resultando em $U_1 = 0$ e $U_2 = \frac{AB}{2}$. O sinal de saída do PSD fica com freqüência igual ao dobro da do sinal de entrada, todavia com um valor médio nulo.

Desta forma podemos perceber que o nível médio de saída do sinal do PSD depende não somente das amplitudes do sinal de entrada e da referência, mas também da defasagem entre os dois sinais. Para determinar a amplitude do sinal de entrada podemos obter o nível médio do sinal demodulado, correspondente ao sinal multiplicado, fixando o valor da amplitude do sinal de referência, para que este tenha um deslocamento de fase igual a zero em relação ao sinal de entrada.

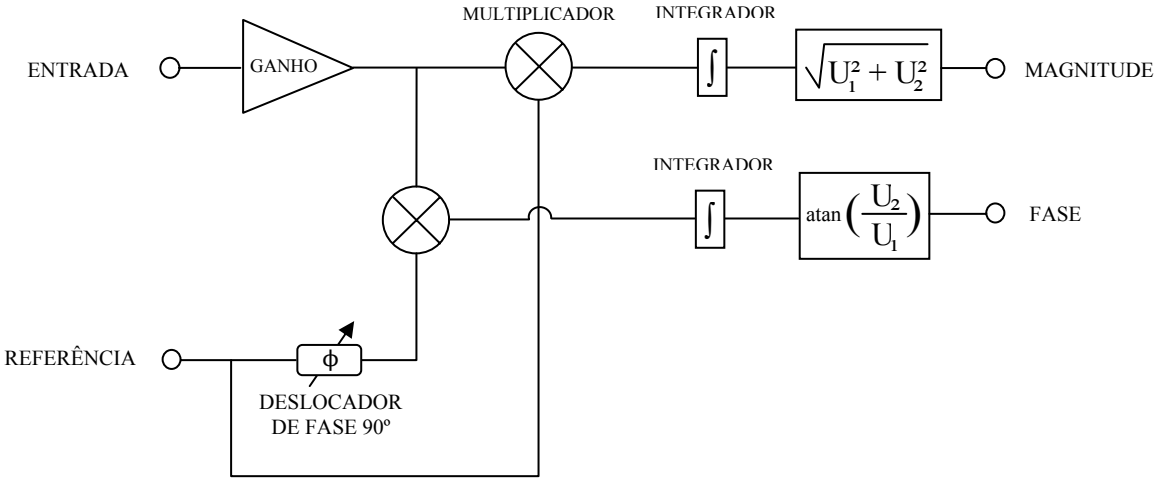
Toda esta análise feita anteriormente é plausível quando tratamos de sinais livres de ruído, o que não se aplica quando o objetivo é a construção de um protótipo real de um Amplificador *Lock-In*. Os ruídos provenientes do sinal de entrada podem não conter uma freqüência fixa, muito menos um deslocamento de fase fixo em relação à referência.

A presença de ruídos não causa diferença na variação do nível DC do sinal de saída do PSD. As componentes das perturbações do sinal de entrada que são próximas à sua freqüência

irão resultar em sinais com freqüências muito mais baixas na saída do demodulador. Para eliminá-las pode ajustar um filtro passa-baixa para trabalhar com uma freqüência de corte inferior à do sinal de entrada, trabalhando em conjunto com o PSD.

Com todos os cálculos previamente definidos, podemos reordenar o diagrama de funcionamento do *Lock-In*, incluindo desta vez as componentes U_1 e U_2, que resultarão nas saídas de fase e magnitude do sinal do sistema físico. Tal representação pode ser vista na Figura 1.4.

Figura 1.4 – Esquemático completo de um Amplificador *Lock-In*. O sinal de referência é multiplicado duas vezes para geração das componentes U_1 e U_2, cada uma destas multiplicações feitas com sinais defasados de 90 graus entre si. Após isso ocorre o cálculo de magnitude e fase, sendo as duas saídas do equipamento.



Capítulo 2

Simulando um *Lock-In*

2.1–Algoritmo Base

Todo o desenvolvimento deste trabalho se iniciou a partir da criação de um algoritmo, pelo qual foi possível analisar e entender melhor o funcionamento do *Lock-In*. A partir da teoria definida no Capítulo 1 criou-se uma rotina matemática para modelar a estrutura interna de um *Lock-In*, composto pelo sinal de referência e por um sinal de entrada simulando um sistema físico ruidoso.

Como demonstrado anteriormente, o *Lock-In* tem um forte potencial de detectar o sinal de entrada imerso em ruído, inclusive onde a Relação Sinal-Ruído (SNR) aproxima-se de zero dB. Isso ocorre devido ao uso de um integrador que atua como um filtro eliminando freqüências indesejadas. Tendo em mente essas características, desenvolveu-se uma rotina de cálculos que incluía um sinal proveniente de um sistema físico fictício extremamente ruidoso e um sinal de referência de mesma faixa de freqüência.

A seguir são mostradas as variáveis de entrada do algoritmo desenvolvido para o simulador, com o intuito de comprovar a capacidade do *Lock-In* em medir sinais mesmo em um ambiente altamente ruidoso. Nesta simulação o usuário pode definir a freqüência de amostragem do sinal, ou seja, a quantidade de amostras lidas por segundo, através da definição de variáveis de entrada. O algoritmo completo encontra-se em anexo no final deste documento, no Apêndice A.

Variáveis de Entrada do Algoritmo			
FAmos	-	Freqüência de amostragem do sinal de entrada	[Hz]
NPObs	-	Número de períodos observados	
NPts	-	Quantidade de pontos por período	
SNR	-	Relação Sinal-Ruído	[dB]
Fase	-	Fase do sinal do Sistema-Físico	[º]
Mag	-	Magnitude do sinal do Sistema-Físico	[V]

Tabela 2.1 – Significado das variáveis de entrada do programa.

2.2 – Definição da Experiência

O *Lock-In* foi simulado para realizar medições com freqüência de referência de 1 MHz e amplitude de referência de 1 V_{pp}. O sistema físico simulado teve amplitude fixada no valor de 50 $mV_{pp,}$ fase de 55º e níveis de ruído com intensidades variando de 0.1 mV_{pp} até 1 mV_{PP} com freqüência de 200 kHz. As constantes de tempo de simulação (número de períodos observados do sinal) variaram entre 1, 5 e 10 períodos. Na Figura 2.1 podemos encontrar uma

demonstração gráfica do sinal de referência e do sinal do sistema físico. Na Tabela 2.2 a seguir encontram-se resumidos os valores de simulação do Lock-In.

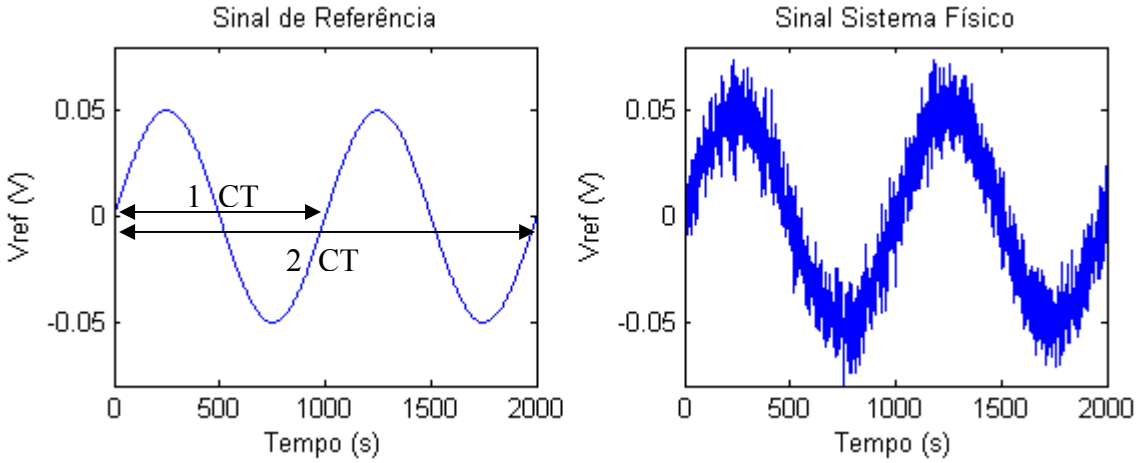


Figura 2.1 – Demonstração do sinal de referência e do sinal do sistema físico.

Sinal de Referência		
Magnitude	(V_{pp})	1.0
Freqüência	(kHz)	200

Sistema Físico		
Magnitude	(mV_{pp})	50
Fase	(º)	55
Ruído	(mV_{pp})	0.1 - 1

Tabela 2.2 – Valores de entrada para simulação do amplificador *Lock-In*, a fim de validar a teoria de redução de ruídos. Inicialmente utilizou-se ruído com 0.1 mV_{pp} finalizando a experiência com ruído de 1 mV_{pp}.

A partir desta simulação, obteve-se uma seqüência de resultados interessante. Na Tabela 2.3 podemos observar de que forma o *Lock-In* se comporta de acordo com a quantidade amostras observadas para a integração do sinal. Podemos perceber que quanto maior a freqüência de amostragem, maior a resolução do sinal, prova disso é a Relação Sinal-Ruído, que aumenta de acordo com a precisão do cálculo da magnitude. Vale lembrar que a freqüência de amostragem, definida por FAmos, é igual ao produto entre o número de pontos do sinal e sua freqüência.

$$N_{pontos} = \frac{FAmos}{f_{entrada}}$$

CT	FAmos (x10^6)	SNR (dB)	Fase (º)	Mag (V)
1	10	26.465	61.19	0.44
1	20	29.759	64.38	0.46
1	50	32.235	57.72	0.51
1	100	38.546	55.99	0.52
5	10	34.447	57.15	0.50
5	20	36.876	54.42	0.49
5	50	40.076	53.86	0.49
5	100	44.431	55.78	0.50
10	10	36.493	55.61	0.50
10	20	39.722	55.51	0.50
10	50	45.076	54.16	0.50
10	100	46.153	54.51	0.49

Tabela 2.3 – Resultados obtidos com nível de ruído de 0.1 mV_{pp}.

Outro ponto interessante a se observar é a variação do resultado com relação ao aumento da constante de tempo, ou seja, a quantidade de períodos que entram na janela de integração do PSD. Quanto maior a quantidade de períodos, maior a quantidade de amostras observadas e conseqüentemente uma maior precisão no cálculo de magnitude e fase.

Para fins de visualização, vamos agora mostrar resultados obtidos quando variamos o nível de ruído para dois valores distintos. Primeiramente um teste com ruído em 0.5 mV_{pp}, numa faixa mediana de perturbação e depois uma simulação com um sinal de entrada altamente ruidoso, de 1 mV_{pp}.

CT	FAmos (x10^6)	SNR (dB)	Fase (º)	Mag (V)
1	10	13.176	51.74	0.37
1	20	15.411	49.98	0.62
1	50	19.719	64.27	0.52
1	100	21.872	46.36	0.44
5	10	18.840	62.80	0.55
5	20	21.633	69.63	0.40
5	50	25.551	50.94	0.56

5	100	27.559	46.83	0.48
10	10	23.275	44.27	0.47
10	20	24.649	65.37	0.56
10	50	27.902	51.48	0.53
10	100	34.202	53.92	0.51

Tabela 2.4 – Resultados obtidos com nível de ruído de 0.5 mV_{pp}.

CT	FAmos ($x10^6$)	SNR (dB)	Fase (º)	Mag (V)
1	10	10.191	66.55	0.72
1	20	10.719	55.16	0.37
1	50	14.417	30.26	1.10
1	100	16.271	63.23	0.66
5	10	12.826	66.48	0.79
5	20	16.981	81.49	0.77
5	50	18.914	55.34	0.56
5	100	25.000	53.43	0.59
10	10	17.269	29.73	0.56
10	20	17.833	65.76	0.42
10	50	23.631	59.21	0.48
10	100	24.183	51.13	0.50

Tabela 2.5 – Resultados obtidos com nível de ruído de 1 mV_{pp}.

Percebemos na Figura 2.2 que quanto maior a Freqüência de Amostragem do sinal de entrada, ou seja, quanto mais pontos são utilizados para o cálculo, maior a precisão do resultado, pois se consegue minimizar os efeitos do ruído. Aliado a uma maior quantidade de períodos integrados, esta precisão aumenta ainda mais.

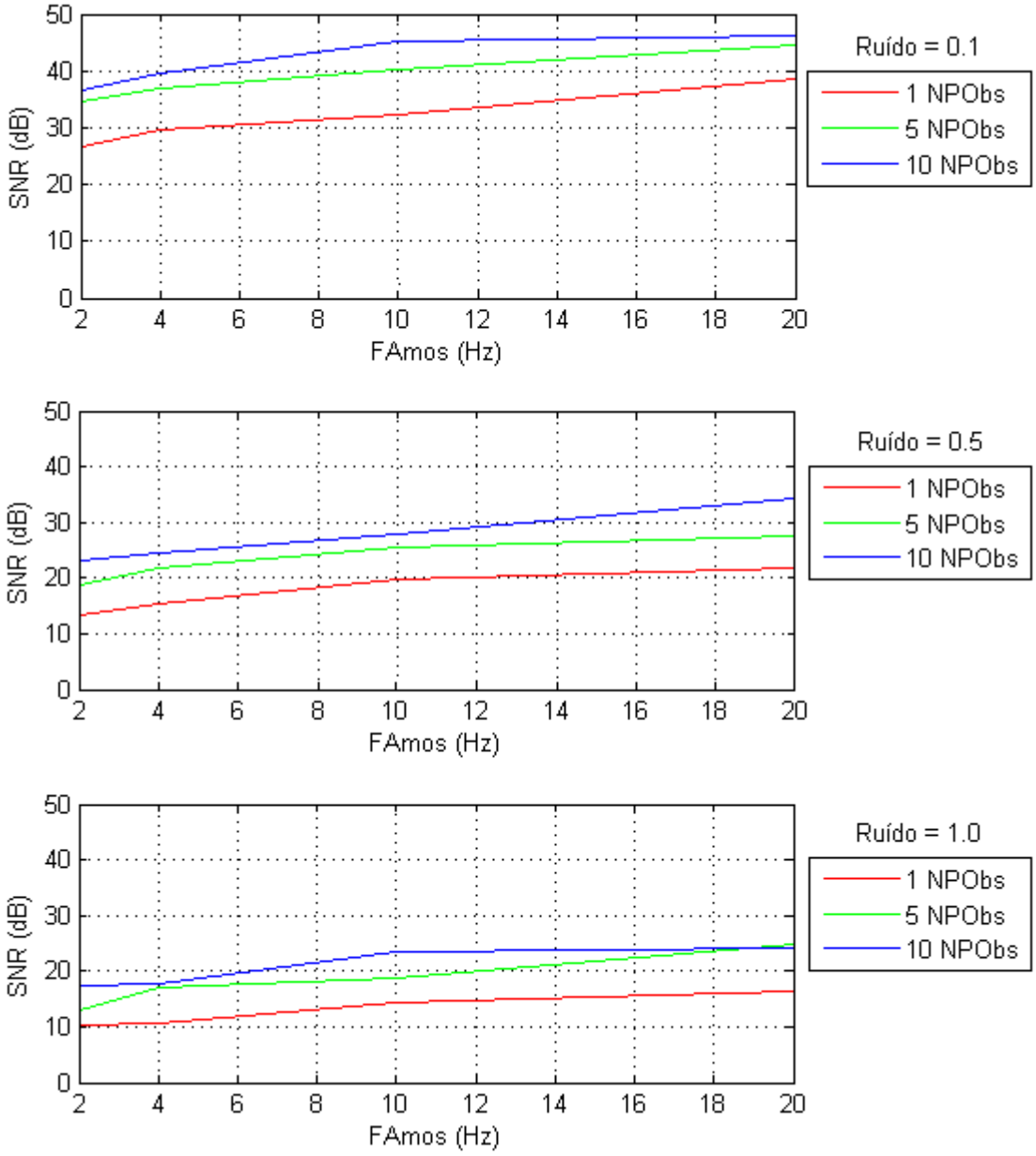


Figura 2.2 – Gráfico comparativo da Relação Sinal-Ruído entre vários tipos de sinais de entrada, com ruídos variando entre 0.1 mV_{pp} e 1.0 mV_{pp}. Quanto maior a relação entre a freqüência de amostragem e o numero de períodos observados, melhor a SNR, ou seja, menos o ruído interfere no cálculo do *Lock-In*.

Os resultados apresentados na Tabela 2.5 mostram um nível de ruído de 0.5 mV_{pp}. Para obtenção do valor da magnitude não houve muita discrepância, mas no valor de fase houve relativa diferença entre o valor desejado e o calculado. A fase só conseguiu estabilizar perto do valor de entrada com um numero maior de períodos observados.

Na Tabela 2.6 vemos um caso crítico, onde o sinal de entrada está completamente imerso em ruídos. Neste caso para conseguir fazer a leitura eficiente da fase e magnitude é necessário uma maior quantidade de períodos integrados no cálculo de U_1 e U_2. Podemos perceber também que se mantendo um numero fixo de períodos observados e variando apenas o numero de pontos do sinal de entrada, a diferença de valores é grande. Por exemplo: com 5 períodos integrados e 50 amostras observadas obteve-se um valor bem diferença do esperado. Apenas com uma quantidade de amostras cinco vezes maior, de 250, é que se conseguiu um valor significativo em relação ao original.

Esta experiência é importante para mostrar que o *Lock-In* é um dispositivo adaptativo, ou seja, se o sinal de entrada do sistema físico é um sinal com baixa taxa de ruído, pode-se diminuir a quantidade de pontos analisados, o que aumenta a rapidez do equipamento. Porém se faz necessário uma maior precisão ou se o sinal de entrada é extremamente ruidoso, podemos aumentar a quantidade de pontos integrados, o que diminui os efeitos das perturbações.

2.3– Desenvolvimento de um Simulador

Com o intuito de demonstrar graficamente o funcionamento do algoritmo, foi desenvolvido em *Matlab* um simulador iterativo, onde o usuário pode definir todas as características do sistema envolvido, definindo as variáveis de entrada de uma amostra, bem como o nível de ruído no qual esta amostra se encontra.

O interessante neste simulador é verificar como o *Lock-In* se comporta entre o momento transiente, antes da janela de integração estar completa com o sinal e o instante em que entra em regime permanente, conseguindo manter praticamente constantes os valores de magnitude e fase.

Na Figura 2.3 podemos encontrar uma imagem da janela do simulador, feito através da ferramenta gráfica *Guide*, presente do *Matlab*, versão 7.0 R14. O código fonte deste programa encontra-se em anexo no Apêndice B, no final deste documento.

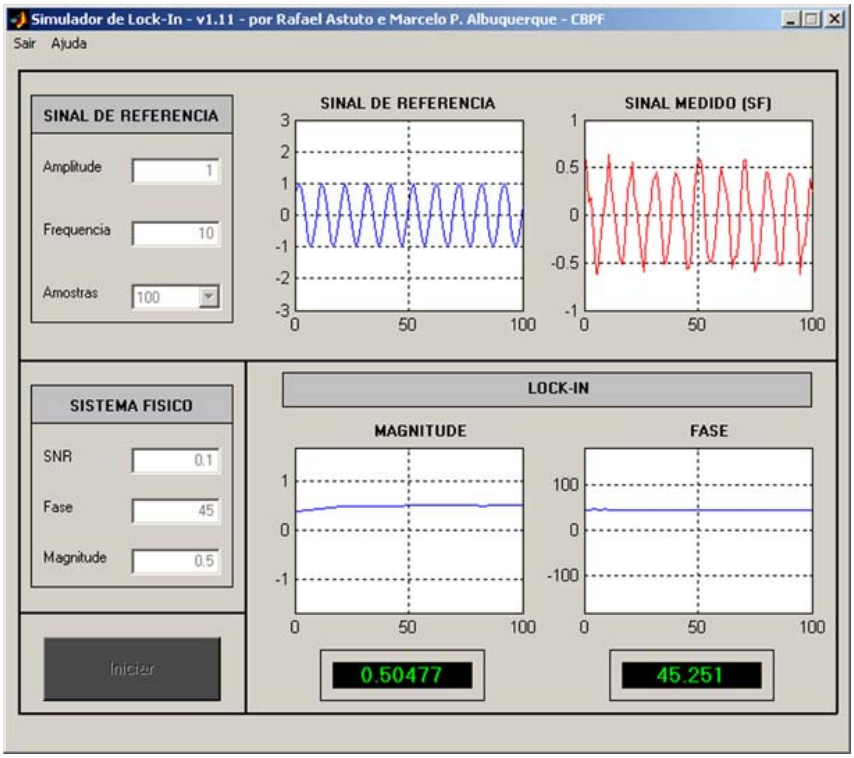


Figura 2.3 – Imagem de tela do simulador de *Lock-In*. Como entrada o usuário pode definir o valor da amplitude, freqüência e o número de amostras do sinal de referência, além de poder configurar o nível de ruído através do valor da Relação Sinal-Ruído (SNR), da fase e da magnitude do sinal provindo do sistema físico.

De acordo com a Figura 2.3 podemos perceber o desempenho do simulador em calcular o valor de magnitude e fase com extrema precisão. No cálculo da amplitude do sinal do sistema físico o simulador forneceu o resultado com um erro de aproximadamente 0,9% e para a fase com um erro de aproximadamente 0,5%, se enquadrando perfeitamente nos padrões matemáticos toleráveis.

2.4– Análise de Resultados

A partir dos gráficos abaixo podemos perceber como o Algoritmo Base do *Lock-In* se aproxima fielmente ao resultado esperado. Foram feitas uma série de simulações, onde se variou o nível de ruído do sinal de entrada, o número de períodos observados e a quantidade de pontos observados em cada período.

Esta simulação foi feita com o intuito de mostrar o quanto o algoritmo é preciso em relação ao cálculo de magnitude e fase. Mesmo com níveis de ruído altos o *Lock-In* conseguiu achar o resultado muito próximo do esperado. Todos os testes simularam um sistema com um sinal de um sistema físico de amplitude 0.5V e fase de 55º.

$$f(t) = 0.5\sin(\omega t + 55^o)$$

Na Figura 2.4 podemos analisar dois testes feitos com ruído de 0.1 mV_{pp}. O primeiro resultado mostra o histograma dos resultados obtidos com apenas 1 período observado, variando-se de 10 a 1000 pontos por período. Neste caso a distribuição mostrou-se bem concentrada, próxima de 55º, com o centro coincidindo à curva gaussiana, onde 65% das amostras convergiram para o resultado esperado.

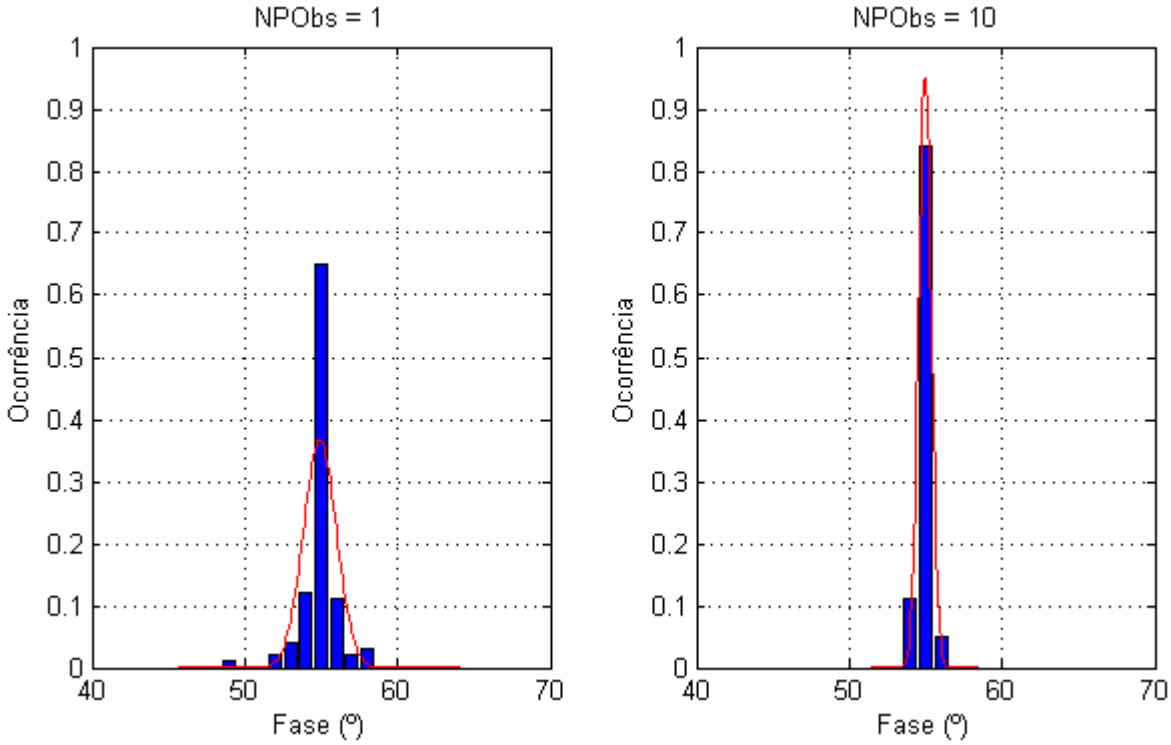


Figura 2.4 – Distribuição dos resultados do cálculo de fase com nível de ruído de 0.1 mV_{pp}, com 1 e 10 períodos observados para uma fase de 55º.

No segundo teste observamos que o maior número de períodos observados fez com que houvesse uma maior concentração ao redor do valor correto da fase, de 55º. Apesar da maior precisão, o fato do nível de ruído ser muito baixo, executar o algoritmo com menos períodos observados e menos pontos torna a rotina mais rápida, melhorando assim a resposta do DSP na hora de realizar os cálculos, caso fosse implementado.

Na Figura 2.5 percebemos que praticamente 100% dos resultados obtidos aproximaram-se do valor indicado de 0.5V tanto para 1 período observado, quanto para 10. Este resultado era esperando, tendo em vista o baixo nível de ruído, facilitando o cálculo da magnitude, que fica menos sujeita à variações de resultado, devido às perturbações.

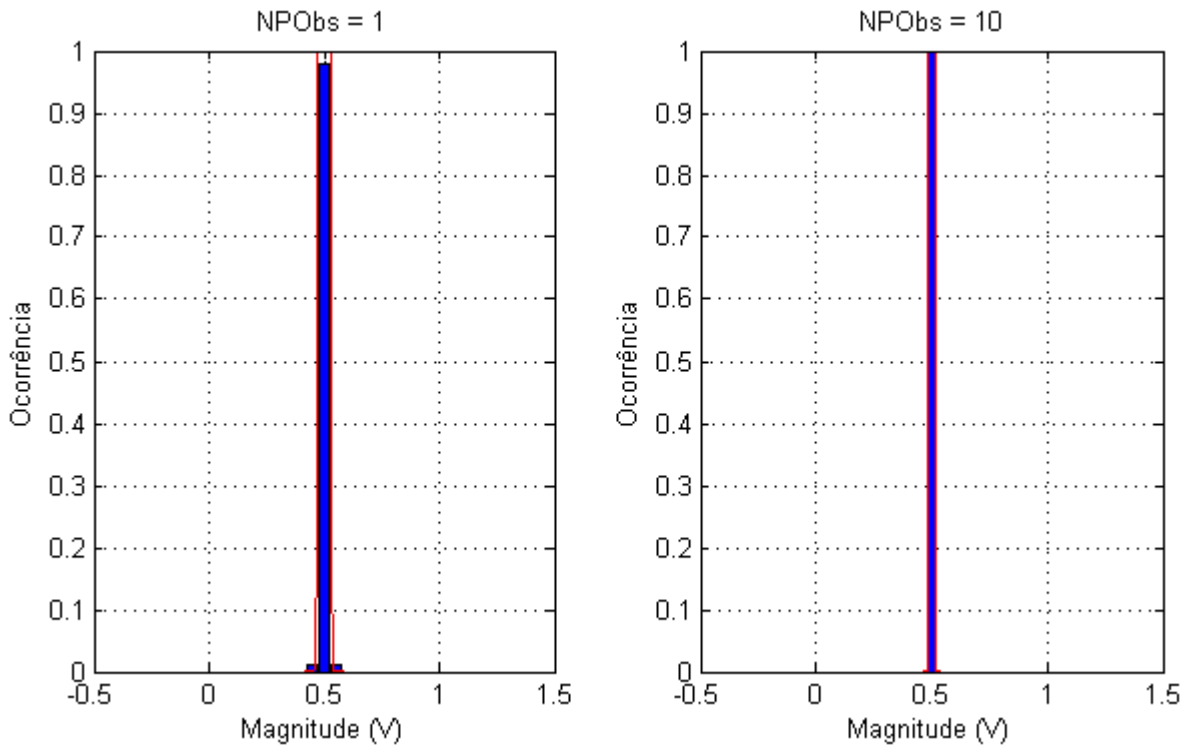


Figura 2.5 – Distribuição dos resultados do cálculo de magnitude com nível de ruído de 0.1 mV_{pp}, com 1 e 10 períodos observados para uma amplitude de 0.5V.

Na Figura 2.4.3 simulamos uma situação onde o nível de ruído corrompe por completo o sinal de entrada. Neste caso o valor da fase fica bastante comprometido, fazendo-se necessário uma maior quantidade de períodos observados para compensar a baixa precisão do sinal do sistema físico. A execução da rotina com 10 períodos integrados fez com que fosse diminuído o efeito alto nível de perturbação.

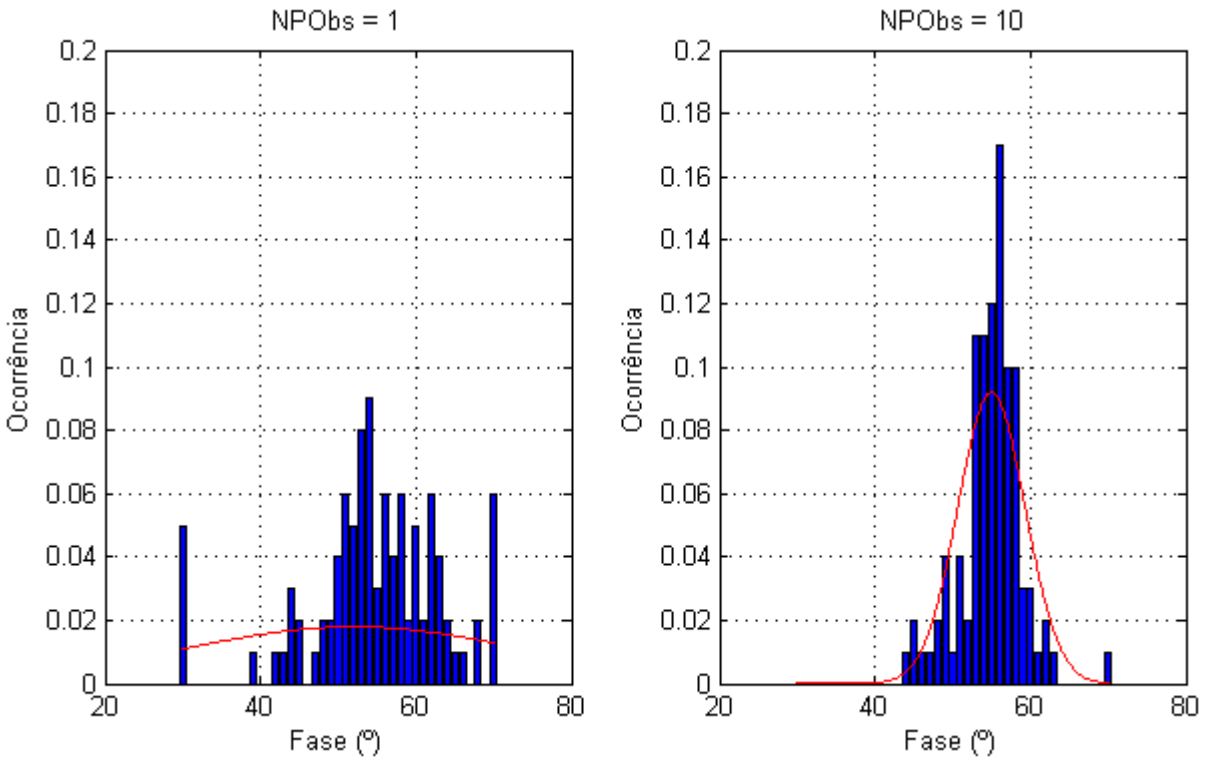


Figura 2.6 – Distribuição dos resultados do cálculo de fase com nível de ruído 1 mV_{pp}, com 1 e 10 períodos observados para fase de 55º. O alto nível de perturbação aliado com uma baixa amostragem do sinal, fez com que a resolução para 1 período de observação fosse muito ruim. Neste caso apenas 10% das amostras se aproximaram do valor desejado, o que não aconteceu quando aumentamos para 10 períodos, onde a gaussiana retrata a maior precisão no cálculo, com aproximadamente 30% das amostras próximas do centro.

Na Figura 2.7 retratamos o comportamento do *Lock-In* para o cálculo de magnitude com um nível de ruído 1 mV_{pp}. Nesta ambiente altamente ruidoso a simulação mostrou que mesmo assim o algoritmo foi robusto, sofrendo pouca variação no resultado. Com 1 período de observação, encontramos mais de 70% das amostras com o valor próximo a 0.5V, e com 10 períodos de observação em torno de 80% das amostras ficaram próximas ao valor desejado.

Para um ambiente muito ruidoso, representado por um nível de 1 mV_{pp} de ruído, o sistema se comportou se forma pouco precisa com apenas 1 período observado, obtendo maior sucesso nos cálculos quando integrados 10 períodos do sinal de entrada. Podemos observar também a maior sensibilidade do *Lock-In* para obter os resultados de fase, onde a variância foi muito maior.

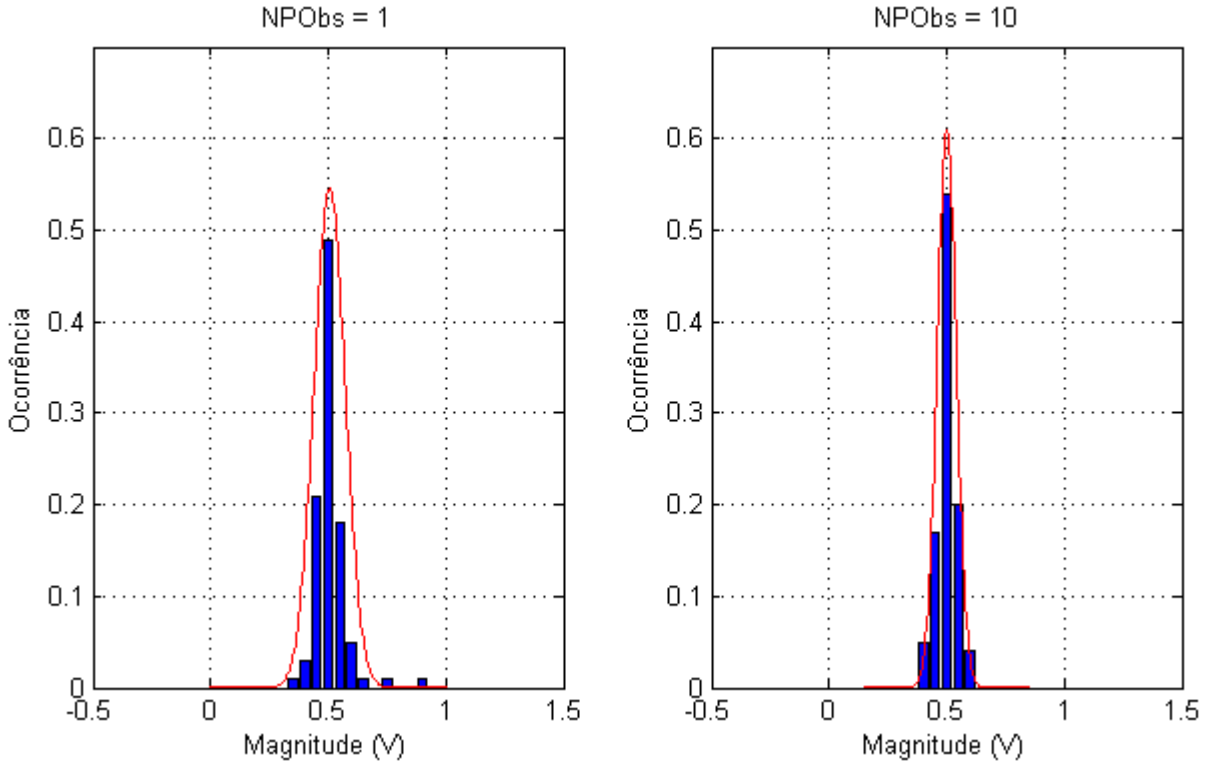


Figura 2.8 – Distribuição dos resultados do cálculo de magnitude com nível de ruído 1 mV_{pp}. O alto nível de perturbação não foi suficiente para desviar o padrão de resultado do algoritmo.

A fim de validar ainda mais o processo estatístico confrontamos o algoritmo a um ambiente ainda mais hostil, com nível de ruído 2 mV_{pp}, extremamente alto, porém submetido a uma análise mais criteriosa, com 10 e 20 períodos de observação, contendo cada período entre 500 e 1000 pontos.

Os resultados obtidos para o cálculo de fase, na Figura 2.9 abaixo, reforçam a alta sensibilidade do *Lock-In* para obtenção da fase do sistema físico. Mesmo com as amostras tendendo a uma distribuição normal próxima do valor de 55º, não houve grande acúmulo de resultados próximos deste valor, que variaram entre 50º e 65º.

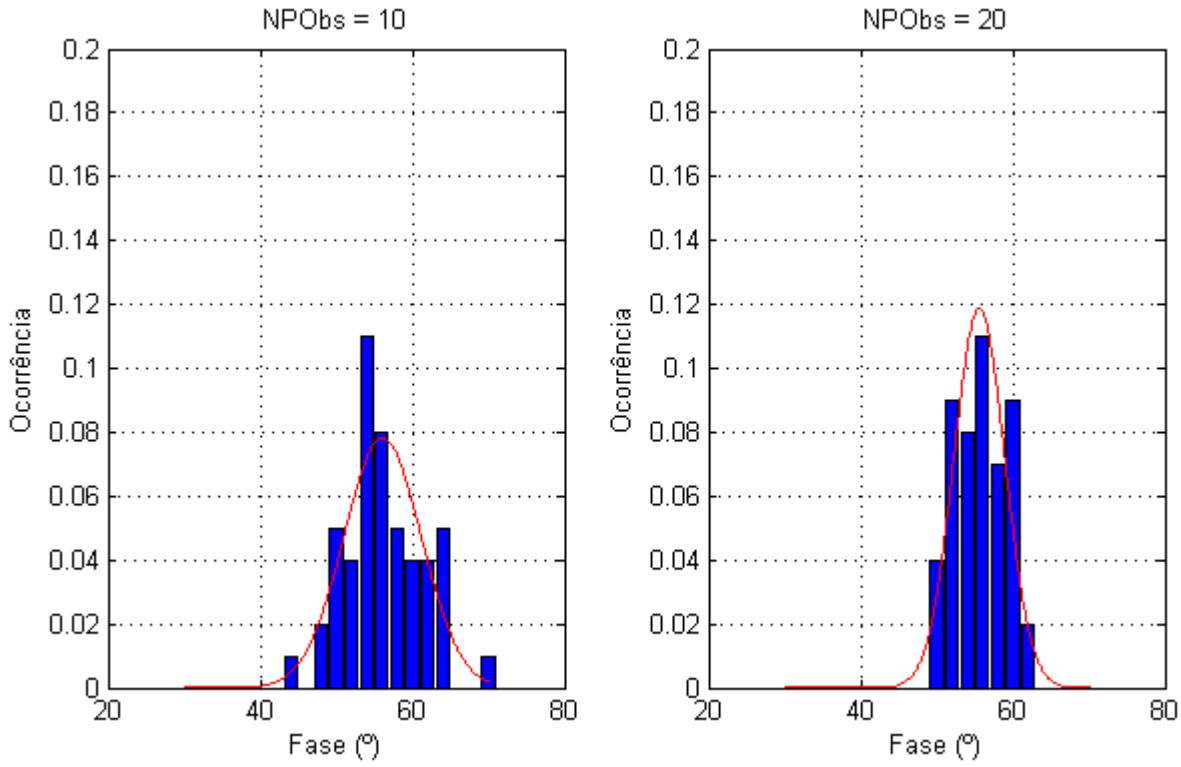


Figura 2.9 – Histograma dos resultados de fase com nível de ruído 2 mV_{pp}.

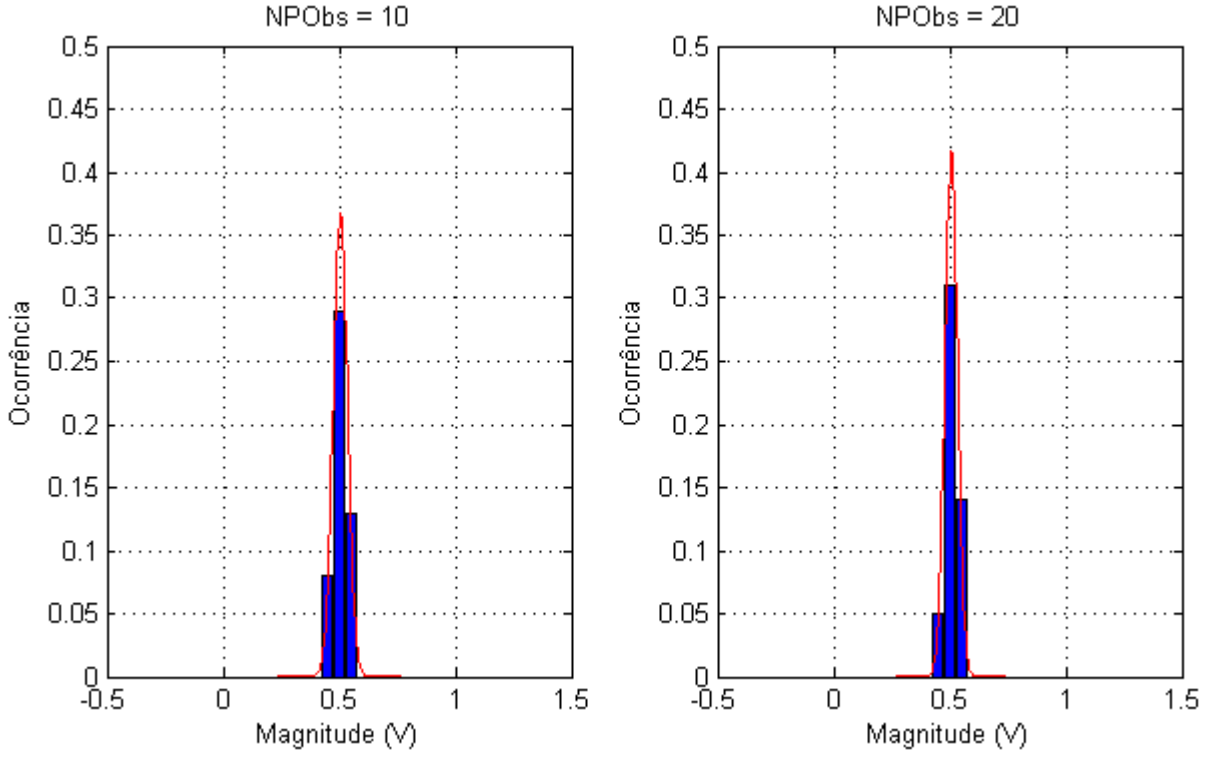


Figura 2.10 – Histograma dos resultados de magnitude com ruído de 2 mV_{pp}.

A Figura 2.10 mostra os resultados obtidos para o cálculo de magnitude do sinal do mesmo sistema com nível de ruído 2 mV_{pp}. Percebemos que o *Lock-In* se comportou de forma bastante homogênea, conseguindo obter valores próximos da amplitude original em quase todas as amostras.

2.5– Comportamento do Desvio Padrão das Medidas

Com os dados de magnitude e fase são traçados gráficos que relacionam o desvio padrão das amostras com a constante de tempo (CT). Neste gráfico, mostrado na Figura 2.11 abaixo, podemos perceber que o *Lock-In* varia exponencialmente com relação com CT, ou seja, quanto maior o número de períodos observados, maior a precisão.

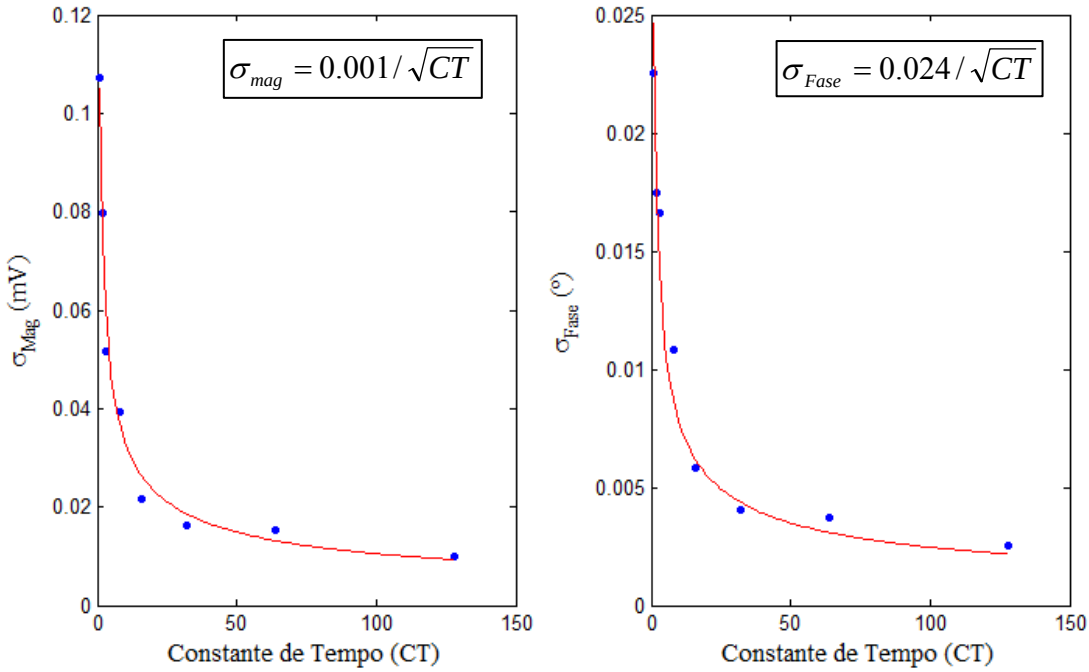


Figura 2.11 – Desvio padrão das medições de magnitude e fase com simulação de ruído de 0.5 mV_{PP} e F_{ref} = 0.5 MHz. Os gráficos mostram que o comportamento exponencial varia inversamente com a raiz quadrada da constante de tempo de integração.

2.6 – Conclusões

Os resultados das simulações mostram como o *Lock-In* se comporta em ambientes altamente ruidosos. Para níveis de ruído baixos, com sinal praticamente limpo, conseguiu-se obter os valores de magnitude e fase com poucas amostras, observando-se apenas 1 período do sinal de entrada. Porém isso não se repetiu ao aumentarmos o nível de perturbação do sinal. Ao elevarmos os níveis de ruído para 1 mV_{pp} e 2 mV_{pp}, representando situações altamente distorcidas, o *Lock-In* precisou de um número maior de amostras e de períodos observados para conseguir analisar a amplitude e o deslocamento de fase.

Outro ponto interessante a ser destacado foi o fato do *Lock-In* ser muito mais sensível no cálculo de fase do que em magnitude. Podemos perceber pelos histogramas anteriores que para o cálculo de fase foram necessários mais pontos, ao passo que para o valor de magnitude, com poucas amostras o resultado já se estabelecia, mostrando o quanto o *Lock-In* é capaz de fazer redução de ruídos, eliminando-os, a fim de fornecer um resultado bastante preciso.

Capítulo 3

DSP Altera Stratix II EP2S60

3.1 – Altera Stratix® II EP2S60 DSP

A família Stratix II é a segunda geração de DSP's da Altera, atualmente em sua quarta geração. Este modelo é uma plataforma programável de alto desempenho que possui interface inclusive com outros DSP's de outros fabricantes, como da *Texas Instruments* e *Analog Devices*. Possui conversores A/D e D/A de alta velocidade, bem como interruptores programáveis e displays de sete segmentos. Abaixo segue uma lista detalhada de todos os componentes e características físicas do kit de desenvolvimento EP2S60:

- Conversores:
  - Dois conversores A/D 12-bits 125 MHz
  - Dois conversores D/A 14-bits 165 MHz
  - Três conversores D/A 8-bits para saída VGA (180 mpx/s)
  - Um codec de 96 kHz stéreo para saída de áudio
- Memória:
  - 1 MB de SRAM (10 ns) assíncrona com barramento de 32-bit
  - 16 MB de memória flash com barramento de 8-bit
  - 32 MB de memória SDRAM com barramento de a 64-bit
  - Conector CompactFlash suportando modos de acesso ATA e IDE
- Configurações
  - Configuração on-board de 16 MB de memória flash (EPM7256 MAX)
  - Configuração por download dos dados usando USB
- Sistema de Expansão
  - Dois conectores para placa filha com Conversores A/D
  - Dois conectores de expansão e prototipagem.

- Outros itens:
  - Display duplo de sete segmentos
  - Quatro interruptores programáveis
  - Uma entrada RS-232 (conector fêmea de 9-pin)
  - Interface Ethernet 10/100 Mbps
  - Oito LED's programáveis
  - Oscilador 100-MHz (em soquete)
  - Sistema de ventilação ativo (*cooler*)

Na Figura 3.1 abaixo podemos identificar estes componentes individualmente e ter uma visão geral da arquitetura da placa.

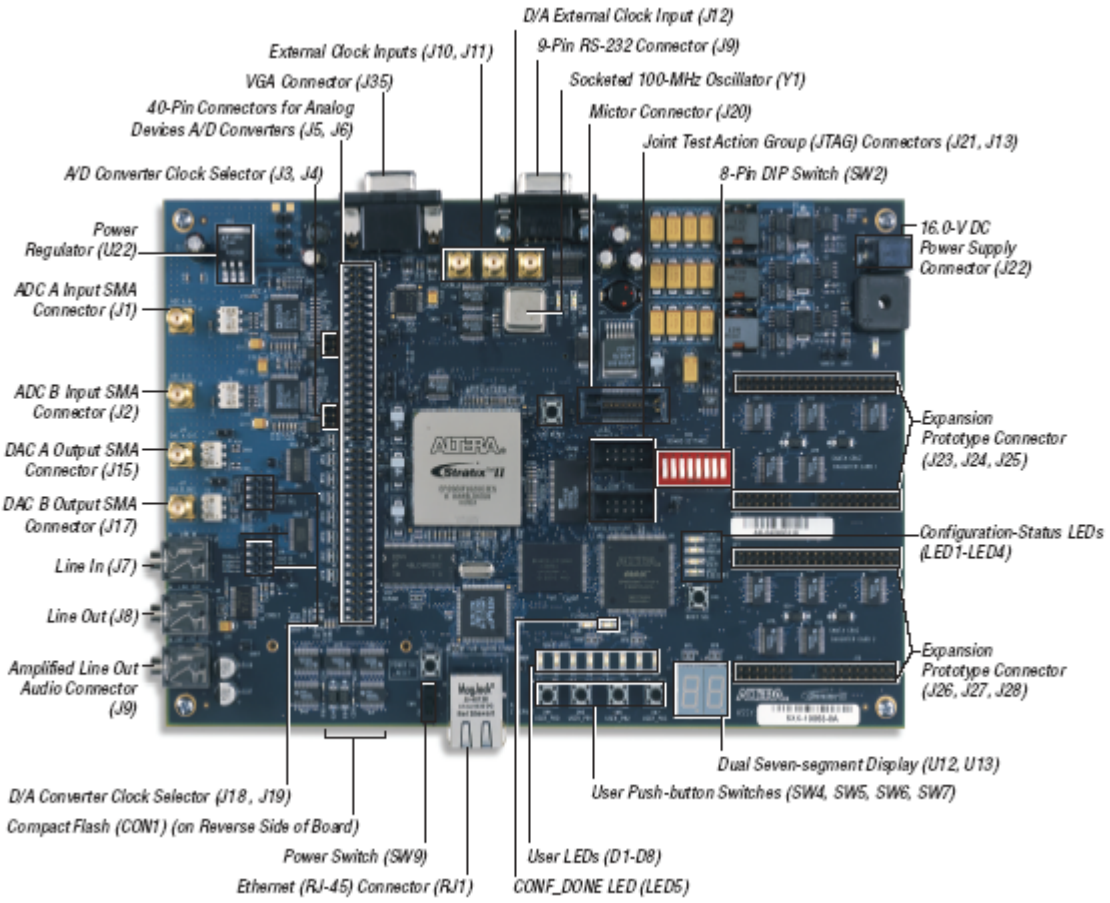


Figura 3.1 – Esquemático da arquitetura do DSP EP2S60.

Fonte: Data Sheet DS-S29804

3.1.1 – Descrições gerais do kit

O kit de desenvolvimento Altera Stratix II EP2S60 DSP possui uma série de interfaces e de componentes adaptados ao desenvolvimento de aplicações para Processamento Digital de Sinais. Nesta seção, na Tabela 3.1, serão mostrados os principais componentes da placa, detalhando-os e descrevendo-os, para melhor entendimento de seu funcionamento. Na Tabela 3.1.2 são mostradas as interfaces de debug e de expansão da placa.

Interface	Tipo	Designação	Descrição
Stratix II	FPGA	U18	EP2S60 StratixII.
Dispositivo MAX	PLD	U10	EPM7256ETC144.
Conversores A/D	E/S	U1 e U2	Conversores 12bits 125 MHz.
Conversores D/A	E/S	U14 e U15	Conversores 14bits 165 MHz.
01 MB SRAM	Memória	U43 e U44	1 MB, 10ns, SRAM assíncrona (32-bit).
16 MB Memória Flash	Memória	U17	16 MB de memória flash (8-bit).
32 MB SDRAM	Memória	U39 e U40	32 MB de memória SDRAM (64-bit).
Conectores SMA (*clock*)	Entrada	J10, J11 e 12	Conectores SMA para entrada de clock externo com terminadores de 50 W.
Display de 7 segmentos	Display	U12 e U13	Display duplo de sete segmentos.
Interruptores	E/S	SW4, SW5, SW6 e SW7.	Quatro interruptores que podem ser definidos como lógica de entrada pelo usuário.
LEDs	Display	D1 – D8	Oito LEDs que podem ter suas funções definidas pelo usuário.
Power On-LED	Display	LED7	Aceso quando o kit está alimentado.
LED de configuração	Display	LED5	Aceso quando a configuração foi carregada com sucesso no dispositivo Stratix II.
Conector RS232	E/S	J29	Conector DB9, configurado como uma porta serial DTE. As voltagens da interface são convertidas para sinais de 3.3V para o dispositivo Stratix II que deve ser configurado para receber e enviar dados seriais.
Oscilador 100 MHz	Clock	Y1	Oscilador em soquete na placa.
Fonte de Alimentação 16VDC	Entrada	J22	Adaptador de 110-240VAC para 16VDC.
Conector JTAG	E/S	J21	Conector JTAG usado para configurar o dispositivo Stratix II diretamente.
Configuração do controlador do JTAG	E/S	J13	Conector JTAG usado para configurar o controlador de configuração.
Conversor D/A VGA	E/S	U45	Saída do conversor D/A de 8 bits, (180 megapixels/s) para VGA.
Áudio CODEC	E/S	U5	CODEC de áudio 96 kHz estéreo.

Tabela 3.1 – Descrição geral dos componentes do kit Stratix II EP2S60 DSP.

Interface	Tipo	Designação	Descrição
Conector Analog Devices	Expansão	J20	Conector Mictor para 33-pins no dispositivo Stratix II (32 dados de sinais e um sinal de clock) para uso como um analisador lógico externo.
Conector TI-EVM	Expansão	J31, J33	Interface ao TI-EVM (no sentido reverso a placa do circuito)
Conectores de prototipagem	Expansão	J23-J28	O kit tem duas interfaces para a placa filha com conectores de 74-pins (Estes pinos são usados para I/O genérico). Estes conectores são denominados como “Santa Cruz Daughter Card 1“ e “Santa Cruz Daughter Card 2”.
Conector Mictor	E/S	J20	Conector Mictor para 33-pins no dispositivo Stratix II (32 dados de sinais e um sinal de clock) para uso como um analisador lógico externo.

Tabela 3.2 – Descrição geral das interfaces de debug e de expansão do kit Altera Stratix II EP2S60 DSP.

3.2– USB Blaster

O USB Blaster é um cabo de dados JTAG que faz a comunicação da placa com o computador, possibilitando o download de programas e a configuração o DSP. Este tipo de conector é compatível com a maioria dos DSPs feitos pela Altera, inclusive a família Stratix II, sendo usado neste trabalho.

O cabo de dados USB Blaster necessita de algumas configurações específicas de alimentação, geralmente padronizadas nos computadores e nos DSPs da Altera:

- 5.0V de alimentação para o cabo USB
- Entre 1.5V e 5.0V nos pinos de conexão da placa

A interface de comunicação é compatível com Windows XP/2000 e o driver para configuração acompanha o CD de instalação do cabo. É muito importante conectar o cabo apenas após instalar o driver no sistema operacional, pois caso seja conectado antes, o computador ao tentar detectar automaticamente o dispositivo, pode conflitar com o driver original. Na Figura 3.2 pode-se visualizar uma foto do cabo JTAG.

Figura 3.2 – Foto do cabo de dados JTAG USB Blaster, utilizado para comunicação entre o computador e o DSP EP2S60. Este cabo ainda pode ser utilizado para outras famílias de DSP da Altera.

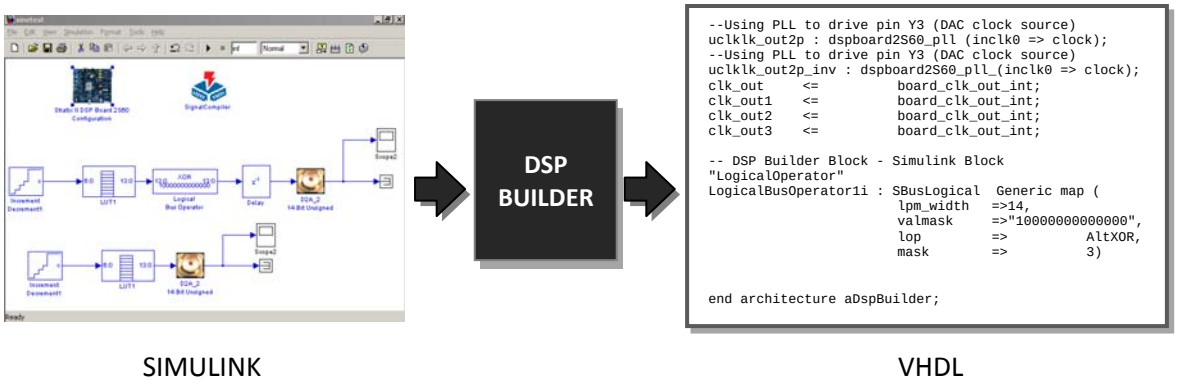
Fonte: Altera Devices [7]

Com o cabo devidamente instalado, pode-se iniciar a programação do DSP. Ao conectá-lo à placa deve-se selecionar na IDE Quartus II qual interface se deseja utilizar para fazer a comunicação com o DSP, nesta hora escolhe-se a opção *"USB-Blaster [USB-0]"*. Neste momento, sempre que ocorrer uma compilação no DSP, o programa automaticamente detecta o dispositivo e realiza a transferência de dados para a placa.

3.3 – DSP Builder

Para realizar a programação no DSP existem duas possibilidades: uma é usar o Quartus II e programá-lo via linha de código ou usando blocos lógicos pré-definidos, a outra é usar a interface desenvolvida pela Altera para trabalhar com o Matlab Simulink. Esta interface chama-se DSP Builder.

O DSP Builder é uma tecnologia criada para converter o diagrama de blocos do Simulink em linguagem HDL, gravada no DSP. Com isso tornou-se muito mais fácil e rápido a elaboração de rotinas e programas no DSP EP2S60. Através da Figura 3.3 pode-se entender melhor como ocorre o funcionamento do DSP Builder.



SIMULINK VHDL

Figura 3.3 – Esquema de conversão do DSP Builder. Ao elaborar um código no Simulink (.mdl) o programa converte os blocos em linguagem VHDL, que pode ser entendida pelo DSP, que realizará a rotina desejada.

3.4– Utilizando o kit Stratix II EP2S60

A utilização do kit DSP EP2S60 é muito simples, porém requer alguns cuidados. Para obter a interface de comunicação com a placa é preciso fazer a instalação de uma série de programas, entre eles: Quartus II (IDE), MegaCore IP Library (bibliotecas e funções compatíveis com o Simulink) e DSP Builder (conversão do código de programação para VHDL).

A fonte de alimentação do kit é única de 16V DC, ligada diretamente no conector J22. Todos os componentes subseqüentes que necessitam de tensões abaixo da de entrada (1.2V, 3.3V e 5V) são automaticamente alimentados. Apesar de possuir um *cooler* sobre o processador principal, o mesmo esquenta muito, assim como os conversores A/D e D/A. É recomendável trabalhar em um ambiente devidamente refrigerado e desligar o kit da chave seletora caso não esteja sendo usado.

Um aspecto negativo do kit é sua programação volátil, ou seja, toda vez que a placa for desligada e religada ela deve ser re-configurada, caso contrário carrega automaticamente sua configuração padrão de fábrica. Caso seja necessária uma aplicação que deva estar gravada sempre que a placa for ligada deve-se optar pela configuração não-volátil do kit. Neste caso deve-se utilizar o controlador de configurações U10 e a memória flash.

O controlador Max não perde seus dados de configuração quando a placa está desligada, programando o dispositivo Stratix II (U18) com dados da memória flash (U17) assim que o sistema é iniciado. O software Quartus II tem a capacidade de gerar arquivos hexadecimais, que podem ser armazenados na memória flash como arquivos de configuração. O usuário pode gerenciar até quatro arquivos de configurações diferentes: três gerados pelo usuário e uma configuração de fábrica. O usuário pode selecionar qual programa será carregado no Stratix II definindo os DIP switches de SW2.

Na Figura 3.4 abaixo podemos encontrar um diagrama funcional com todas as principais características da placa.

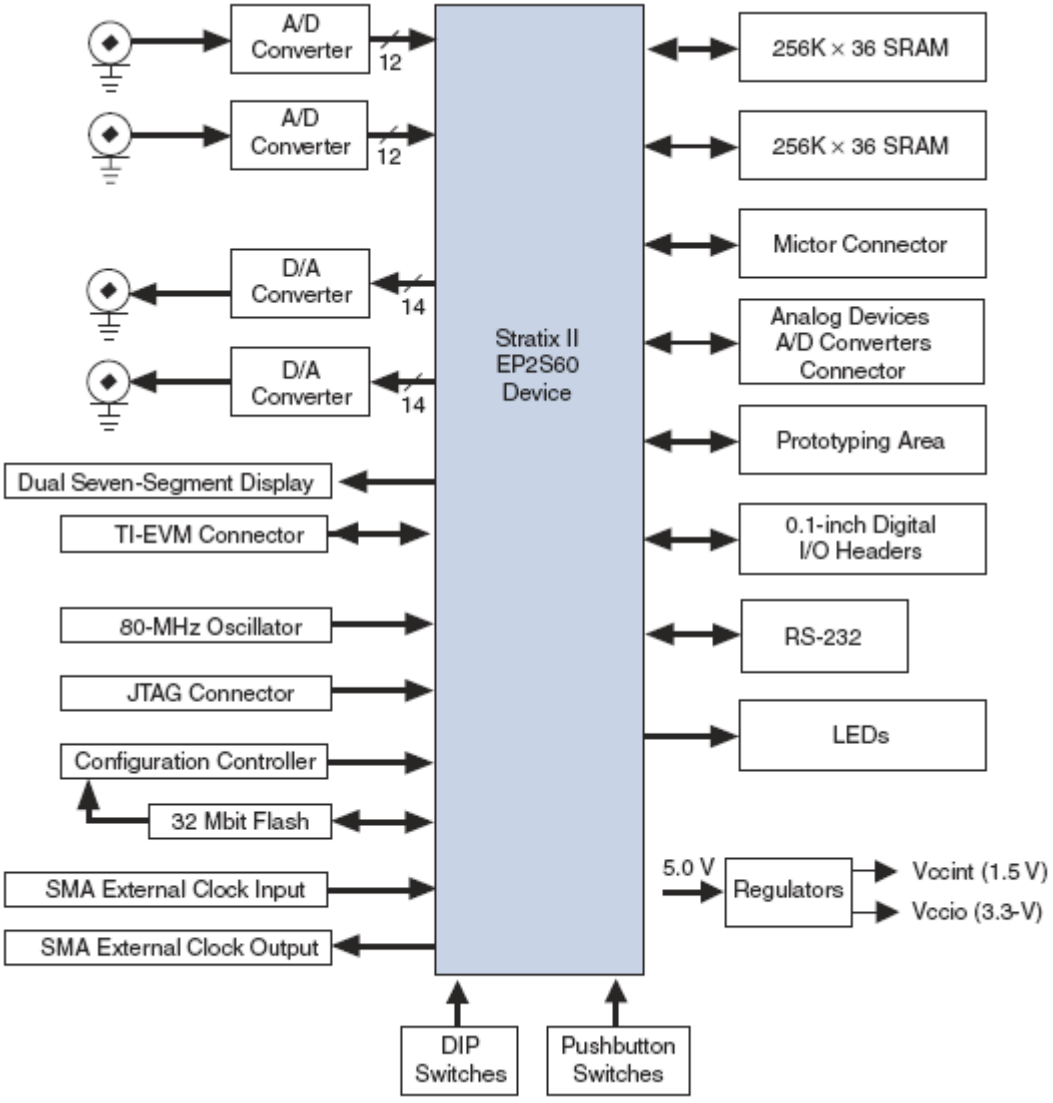


Figura 3.4 – Diagrama de blocos funcional do kit DSP EP2S60.

Fonte: Data Sheet DS-S29804

3.4.1 – Conversores A/D

O kit Stratix II EP2S60 possui dois conversores A/D de 12-bits cada, com a capacidade de gerenciar até 125 milhões de amostras por segundo (MSPS[3]). Circuito de entrada do conversor A/D utiliza um transformador de acoplamento com freqüência de corte inferior em 3-dB de aproximadamente 1 MHz. O sinal de clock que alimenta o conversor A/D pode ser configurado internamente ou externamente. Este sinal pode ser fornecido pelo dispositivo Stratix II, ou por meio do conector de entrada de um clock externo, ou pelo clock de referência de 100 MHz da própria placa.

3.4.2 – Conversor D/A

O kit Stratix II EP2S60 DSP tem dois conversores D/A, com as seguintes características: amostragem de 14-bits com freqüência máxima de 165 milhões de amostras por segundos (MSPS) e saída analógica limitada com codificação dos dados no formato *unsigned integer*.

O sinal de clock do conversor D/A é obtido direto do dispositivo Stratix II. A Figura 3.5 apresenta o circuito após o conversor D/A. O chip DAC904, consiste de uma fonte de corrente com valor máximo de 20 mA e a saída do conversor é aterrada por meio de um resistor de 51Ω. Além disso, encontramos um capacitor de 27 pF em paralelo com o resistor de saída resultando em um filtro passa baixa, de pólo único, com freqüência de corte superior de 230MHz. A saída é obtida no conector SMA.

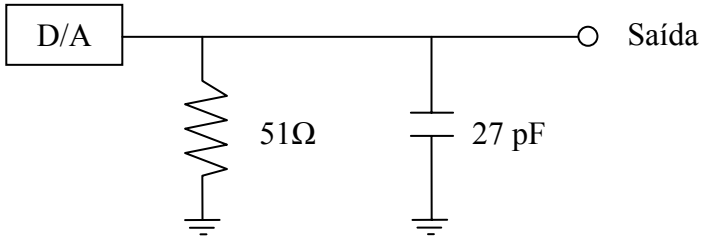


Figura 3.5 – Filtro passa-baixa após conversor D/A do kit EP2S60.

[3] MSPS – million samples per second

Capítulo 4

Resultados

4.1– Interface com Matlab

O Matlab foi de grande importância no desenvolvimento do protótipo de *Lock-In* baseado nos algoritmos apresentados no Capítulo 2 e no Apêndice A. Através do Simulink foi possível criar um ambiente de comunicação com o DSP, a fim de coletar dados e manipulá-los em rotinas dentro do próprio Matlab, com rapidez e simplicidade.

A instalação do DSP Builder e das funções do MegaCore nas bibliotecas do Simulink permitiram a criação de um programa baseado nas componentes da placa. Todos os principais componentes como interruptores, LED's, conversores, memória entre outros, encontram-se representados como blocos configuráveis. Desta forma foi possível desenvolver o corpo da rotina do *Lock-In*.

4.2 – Resultados

4.2.1 – Medições de Magnitude

Para o teste de magnitude foram introduzidos no conversor A/D dois sinais idênticos, de mesma amplitude e freqüência, que foram medidos pelo DSP. A placa, programada com o algoritmo do *Lock-In*, fez as medições de magnitude e fase, que neste caso (sinais idênticos) foi igual a zero.

Como primeiro teste foi introduzido um sinal com amplitude de 1,38V, fase zero e 1 MHz de freqüência. Os resultados das medições podem ser conferidos na Figura 4.1.

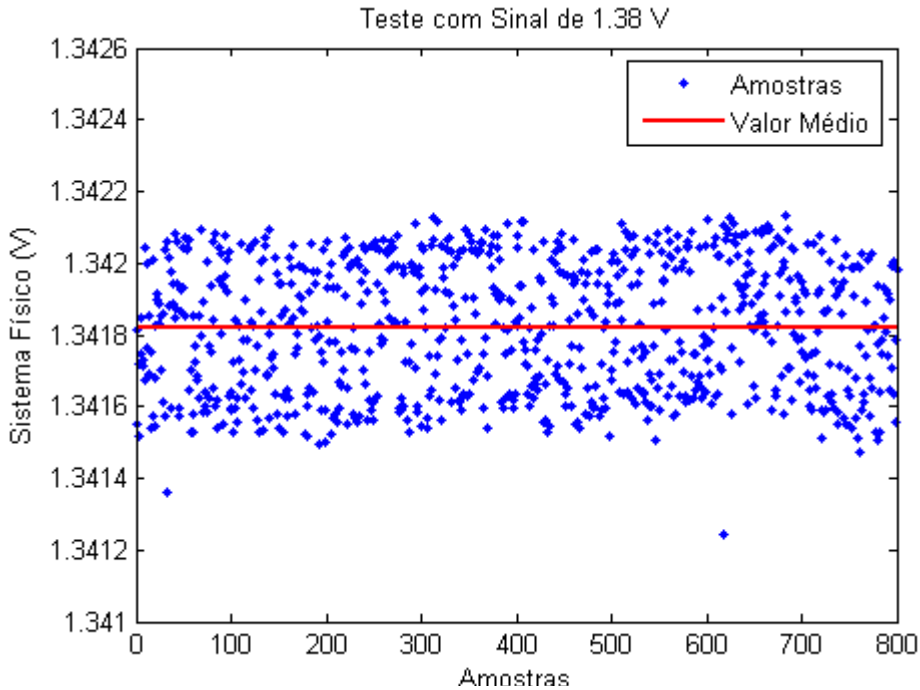


Figura 4.1 – Em vermelho o valor médio obtido, em torno de 1,3418V, o que representa um erro de menos de 3% em relação ao sinal original.

Em um segundo teste foi introduzido um sinal de menor amplitude com entrada de 689 mV e precisão de 8% em relação ao sinal original, como pode ser comprovado na Figura 4.2.

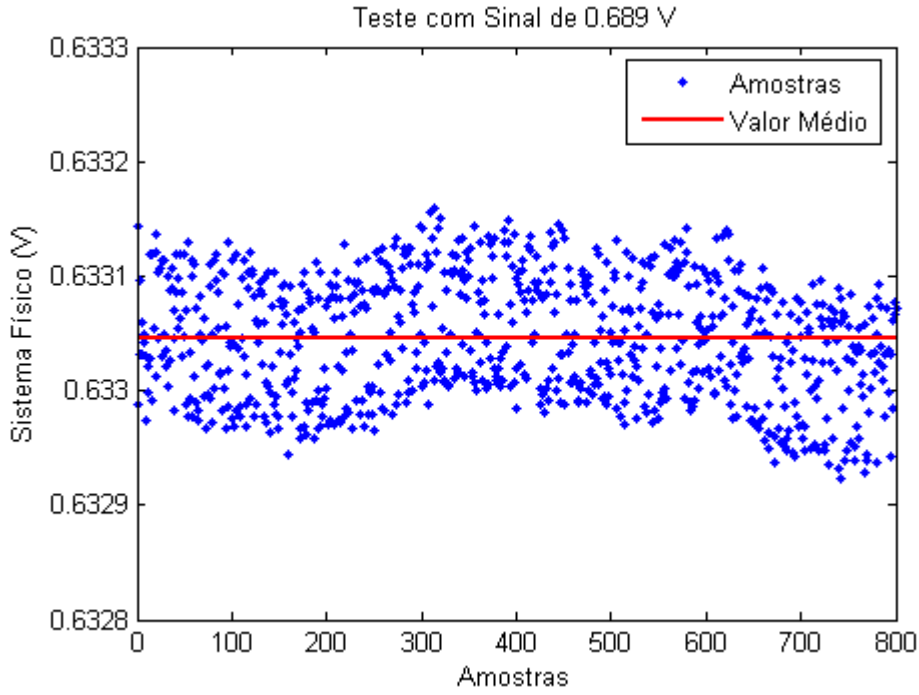


Figura 4.2 – Entrada de 0.689V com erro na faixa de 8%.

Baixando-se ainda mais a amplitude do sinal, desta vez para 355 mV o *Lock-In* ainda conseguiu manter boa precisão na medição, mantendo a mesma faixa de erro de 3% em relação ao sinal original, como pode ser comprovado na Figura 4.3.

Por fim introduziu-se um sinal extremamente baixo, com 224 mV, o menor sinal que a fonte utilizada conseguia emitir. Neste momento houve uma maior discrepância em relação ao sinal original, ou pela imprecisão da própria fonte (que estava em seu limite inferior) ou pelo próprio *Lock-In* que não conseguiu efetuar corretamente a leitura devido a maior susceptibilidade do sinal a ruídos nesta faixa de amplitude, como pode ser observado na Figura 4.4.

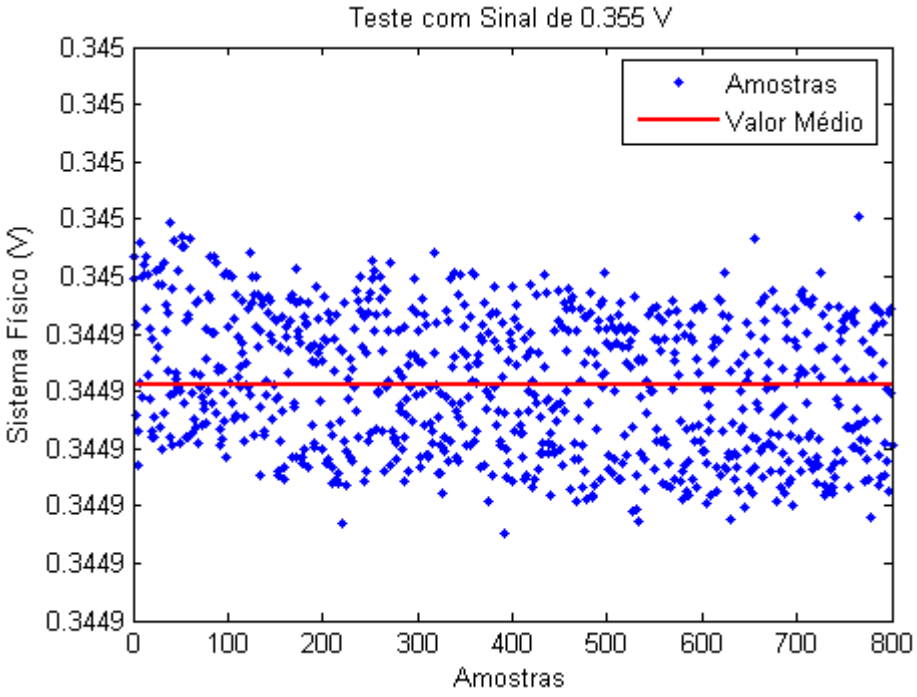


Figura 4.3 – Distribuição de 800 amostras com entrada de 355 mV

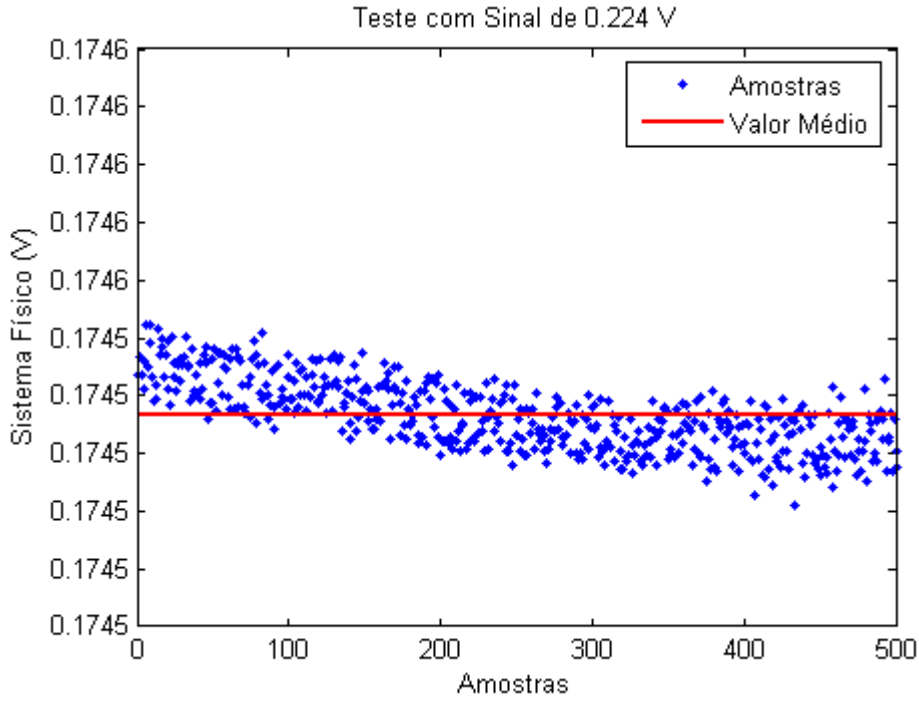


Figura 4.4 – Distribuição de 500 amostras com entrada de 224 mV. A presença de ruídos ou a imprecisão da fonte para baixos sinais foram fatores que resultaram em um erro de 28%.

Na Tabela 4.1 podemos encontrar um detalhamento dos resultados mapeados para os diversos tipos de sinais testados. Podemos perceber que quanto menor a amplitude do sinal, maior o desvio padrão, deixando a medição com menor precisão. Isso pode se dever ao fato da maior interferência de ruídos externos, à baixa sensibilidade do conversor para este nível de amplitude ou ainda à imprecisão da fonte para baixos sinais, tendo em vista que com 224 mV já se encontrava em seu limite inferior.

Entrada (V)	Lock-In (V)	Precisão	Desvio Padrão (V)
2,671	2,6354	1,35 %	0,01
2,212	2,1780	1,56 %	0,01
1,931	1,8909	2,12 %	0,01
1,569	1,5399	1,89 %	0,01
1,380	1,3410	2,84 %	0,02
0,689	0,6570	4,87 %	0,01
0,512	0,4973	2,95 %	0,02
0,465	0,4450	4,50 %	0,001
0,355	0,3449	4,36 %	0,001
0,224	0,1745	28,33 %	0,001

Tabela 4.1 – Resultados obtidos de medições de magnitude pelo *Lock-In*.

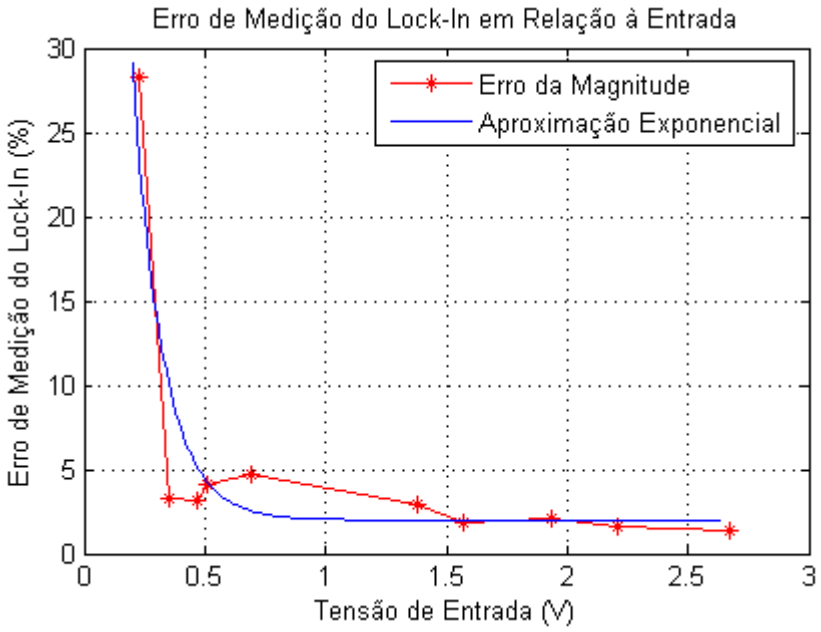


Figura 4.5 – Em vermelho a relação entre a tensão de entrada do sistema físico experimental e o erro na medição do *Lock-In*. Em azul uma aproximação exponencial do erro.

Com estes dados conseguimos traçar um perfil da resposta do *Lock-In* em relação ao erro da medida. Com a ajuda do Matlab foi possível definir uma equação que resulta no erro de medição do sinal de entrada de acordo com o nível de amplitude, sendo assim ser possível prever a precisão de uma medição. Na Figura 4.5 acima encontramos um gráfico desta relação.

O erro da medição da magnitude do *Lock-In* pôde ser aproximado por uma regressão exponencial. Sendo assim, pode-se conseguir analisar antes mesmo de se introduzir um sinal de entrada, qual será seu erro de medição que foi definido na equação de erro abaixo:

$$e(\%V) = 90.e^{-8V} + 2$$

4.2.2 – Medições de Fase

Analogamente às medições de magnitude foram feitos testes para medição de fase. Para esta situação o sinal era replicado nas duas entradas do *Lock-In*, com a finalidade de obterem-se medições de diferença de fase iguais a zero. Para isso foram feitas 2000 medições consecutivas, que foram demonstradas na Figura 4.6 abaixo.

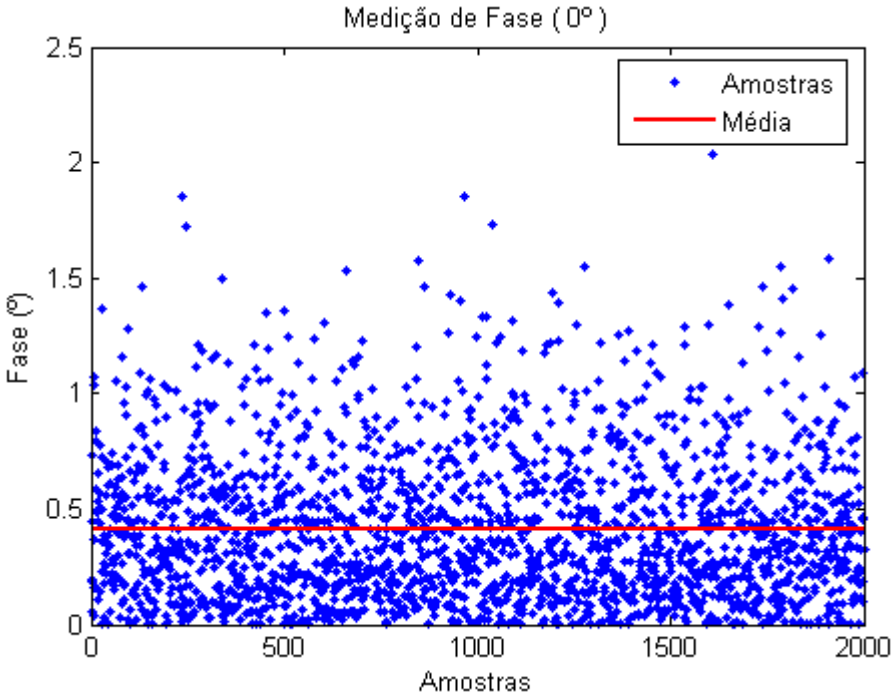


Figura 4.6 – Dispersão das amostras calculadas pelo *Lock-In* para sinais em fase. O valor médio das amostras foi de 0,48º com um desvio padrão de 0,32 graus. A precisão foi em torno de 3% do valor inicial.

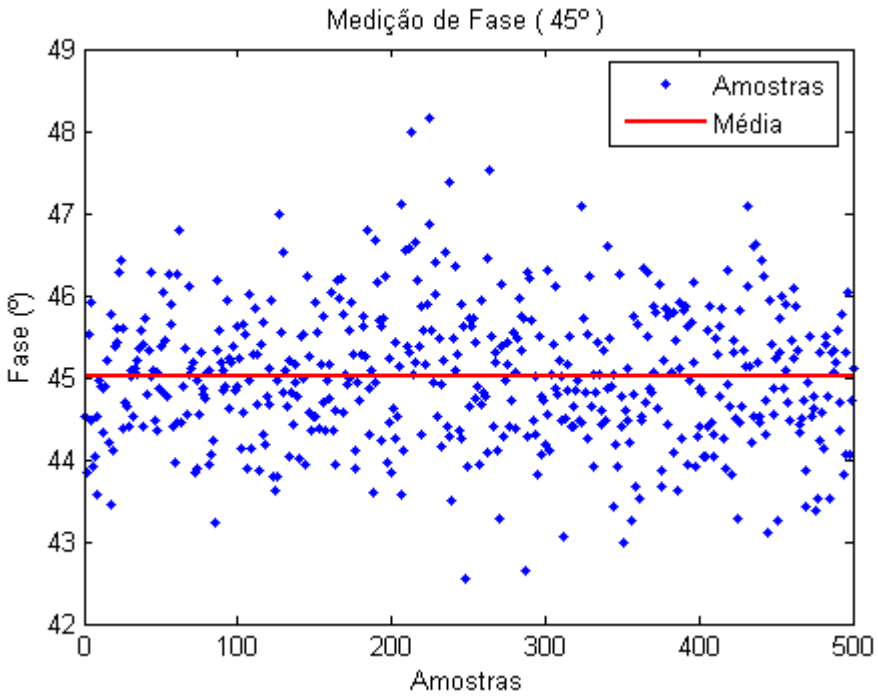


Figura 4.7 – Dispersão das amostras para uma defasagem de 45 graus. A precisão deste cálculo foi de 4,49% com um desvio padrão de 0,83 graus.

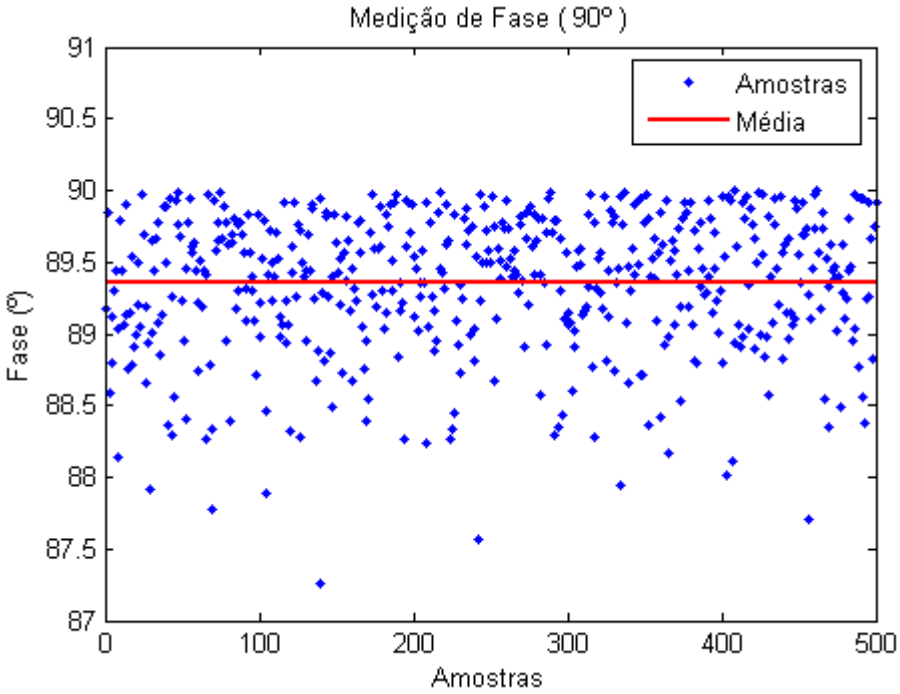


Figura 4.8 – Dispersão das amostras para uma defasagem de 90 graus. A precisão deste cálculo foi de 8,92% com um desvio padrão de 0.49 graus.

Foram realizados dois outros testes a fim de validar a precisão do *Lock-In* na medição de fase. A Figura 4.7 mostra a dispersão das amostras para uma variação de fase de 45 graus e a Figura 4.8 para uma defasagem de 90 graus em relação ao sinal de referência.

Os resultados obtidos com os testes encontram-se na Tabela 4.2 abaixo, focando na precisão dos cálculos, a média dos valores encontrados e o desvio padrão de todas as amostras analisadas.

Defasagem (º)	Precisão	Média (º)	Desvio Padrão (º)
0	3,15 %	0,41	0,32
45	4,49 %	45,02	0,83
90	8,92 %	89,35	0,49

Tabela 4.2 – Resultado dos três testes feitos para medição de fase.

Capítulo 5

Utilização de Ultra-som Para Medição de Espessura de Amostras de Alumínio

5.1 – O Experimento

Nesta parte do trabalho foram utilizados sensores para emissão de ondas de ultra-som com a finalidade de fazer medições de espessura de amostras de alumínio com muita precisão. Em suma o equipamento assemelha-se a um micrômetro, pois consegue medir pequenas espessuras, da ordem de décimos de milímetros.

Os dois sensores possuem freqüência de ressonância na faixa de 40 kHz, como pode ser demonstrado no gráfico da Figura 5.2. Para obter este valor colocaram-se ambos os sensores um em frente ao outro, com distância mínima, próxima de zero mm, como demonstrado na ilustração da Figura 5.1 abaixo.

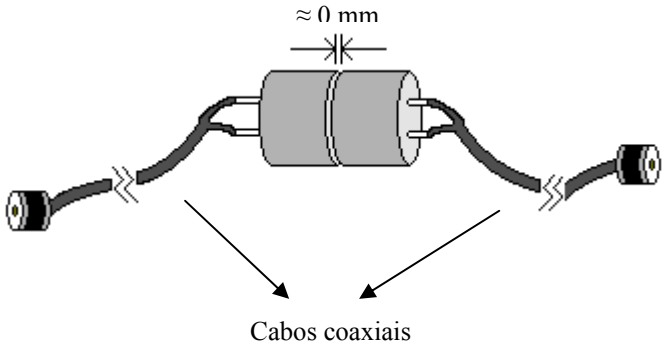


Figura 5.1 – Esquema de montagem para estimação da freqüência de ressonância. Foram soldados aos terminais dos sensores cabos coaxiais, a fim de conectar o emissor a uma fonte AC e o receptor em um osciloscópio. A distância entre ambos pode ser considerada desprezível.

Com os sensores devidamente posicionados, fixou-se a saída da fonte em $20V_{pp}$ (máximo da fonte), conectando-a ao emissor. Este valor de tensão foi escolhido, pois a perda de energia, mesmo com uma distância pequena, é muito grande, fazendo com que a tensão lida pelo receptor seja da ordem de um décimo da de entrada.

Variando-se a freqüência da fonte de 10 kHz em passos de 1 a 5 kHz, dependendo da precisão obtida. Até a faixa de 33 kHz a tensão lida pelo receptor foi muito baixa, da ordem de 1 mV_{pp}. Após esse valor, pequenas variações de freqüência começaram a ser mais sentidas pelo receptor. Chegou-se então em um valor máximo de tensão em exatos 40 kHz. Valores acima ou abaixo deste, fazem a leitura do sinal cair drasticamente, perdendo muita resolução.

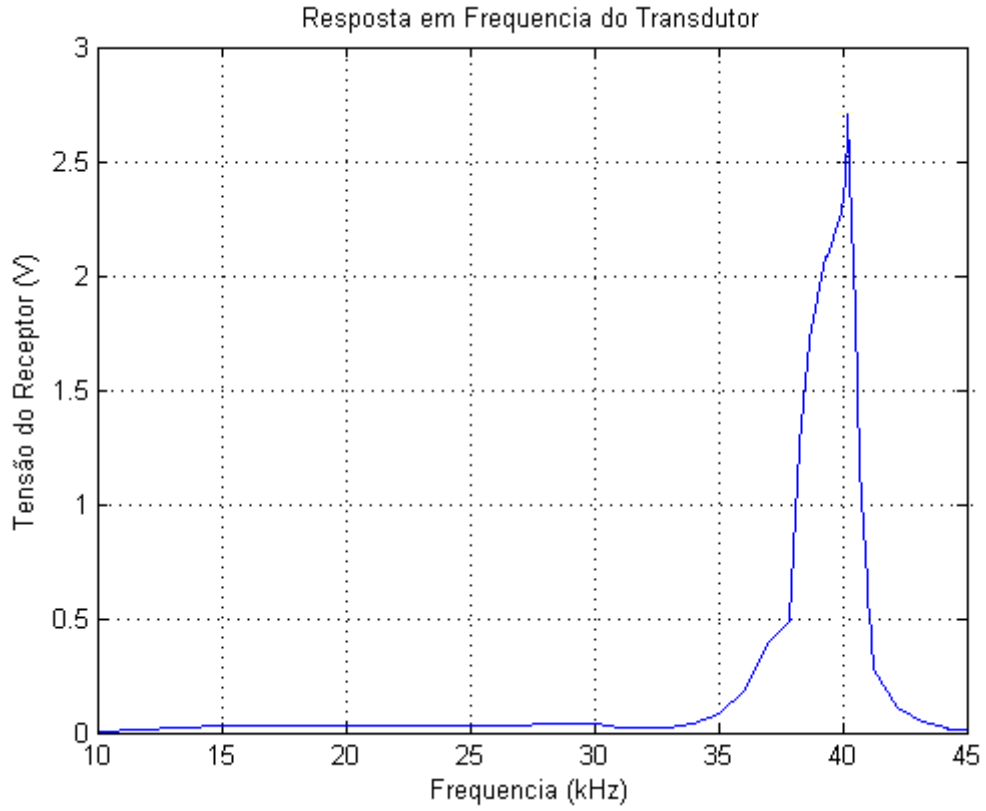


Figura 5.2 – Gráfico de resposta em freqüência do transdutor. A leitura do receptor obteve a máxima tensão com um sinal de 40 kHz.

Com este valor, fixou-se a fonte em 40 kHz e $20V_{pp}$ para todas as medições. Os sensores foram presos em um torno mecânico manual[4], com o transdutor receptor na ponta fixa e o emissor na parte móvel, como ilustrado na Figura 5.3 abaixo.

[4] Também conhecido como morsa.

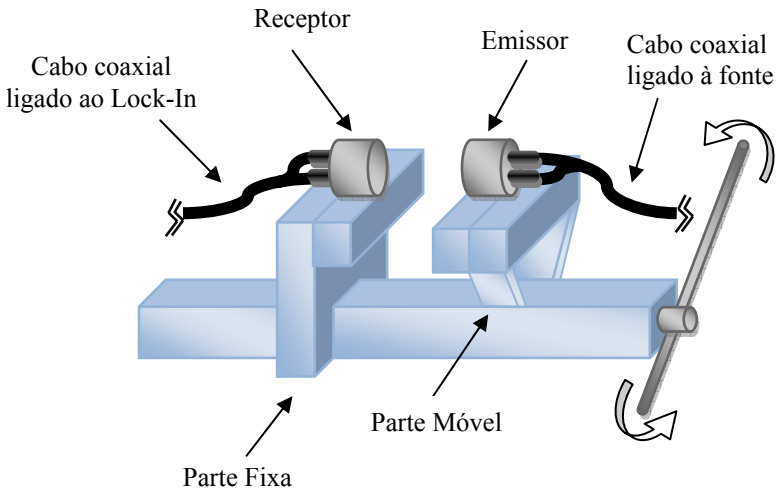


Figura 5.3 – Esquema de montagem dos sensores. Utilizando-se um torno manual, colocou-se o receptor na parte fixa e o emissor, conectado à fonte, na parte móvel. Variando-se a distância através da manivela, consegue-se com o Lock-In, medir a diferença de fase entre os dois sinais.

A partir da montagem dos sensores iniciou-se a medição de amostras de alumínio, que foram usinadas com espessuras variadas, desde 1 mm até 10 mm, em intervalos de 1 mm cada. As amostras tinham as duas faces bem polidas e fora feitas com um diâmetro maior do que dos sensores.

Com a utilização de uma pequena quantidade gel de ultra-som, o mesmo utilizado em exames médicos, prendeu-se a amostra entre sensores com uma leve pressão do torno, apenas para firmá-las. O gel foi usado para melhorar a transferência de energia do sinal entre os sensores e a placa de alumínio, funcionando como um condutor para a onda mecânica.

Para se chegar à espessura do material foi necessária uma simples medição de fase entre o sinal emitido e o sinal recebido pelo sensor. Essa diferença de fase, em pequenas distâncias, é diretamente proporcional à espessura da amostra. Sabendo-se a velocidade de propagação do som no alumínio, a freqüência do sinal e a fase do sinal é possível calcular de imediato a espessura da amostra.

Com a freqüência do sinal da fonte e a velocidade do som no alumínio ou em qualquer outro material que se deseja trabalhar, sabe-se o comprimento de onda:

$$v = \lambda . f \qquad \text{Eq. 1}$$

$$v_{Al} = 4420\, m/s$$

$$f = 40\, kHz$$

$$\lambda_{Al} = \frac{f}{v_{Al}} \qquad \text{Eq. 2}$$

$$\lambda_{Al} = \frac{4420}{40000} = 0{,}1105\, m$$

Com o comprimento de onda em mãos sabemos que um período, ou seja, 360º, correspondem a 0,1105 m. Através de simples proporção chega-se que para cada 1º têm-se o equivalente a 0,30694 mm. Com isso podemos tabelar os valores teóricos da diferença de fase de acordo com a espessura de cada amostra, conforme a Tabela 5.1 abaixo:

Espessura da Amostra	Diferença de Fase Teórica
1 mm	3,26 º
2 mm	6,52 º
3 mm	9,77 º
4 mm	13,03 º
5 mm	16,29 º
6 mm	19,55 º
7 mm	22,81 º
8 mm	26,06 º
9 mm	29,32 º
10 mm	32,58 º

Tabela 5.1 – Com o valor inicial de 1º = 0,30694 mm, tabelou-se os valores teóricos de diferença de fase de todas as amostras, a fim de saber com antecedência se o valor medido pelo Lock-In está ou não de acordo com valores pré estabelecidos de precisão e margens de erro.

A seguir encontram-se algumas fotos tiradas a partir da montagem real do esquema de medição. Na Figura 5.4 pode-se observar os dois sensores de ultra-som e uma amostra de alumínio entre eles sendo medida. Na Figura 5.5 encontra-se a foto detalhada da montagem dos sensores na ponta do cabo coaxial. Por fim na Figura 5.6 encontramos uma foto do DSP conectado aos dois sensores fazendo a leitura dos sinais.

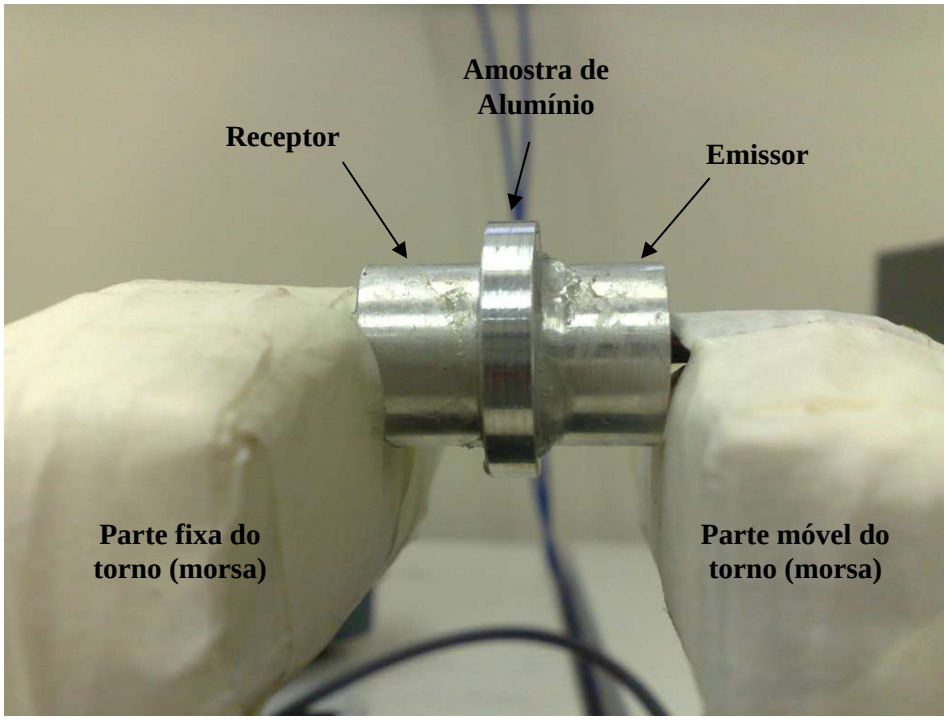


Figura 5.4 – Foto do esquema de montagem dos sensores.

Figura 5.5 – Foto dos sensores soldados na ponta de um cabo coaxial.

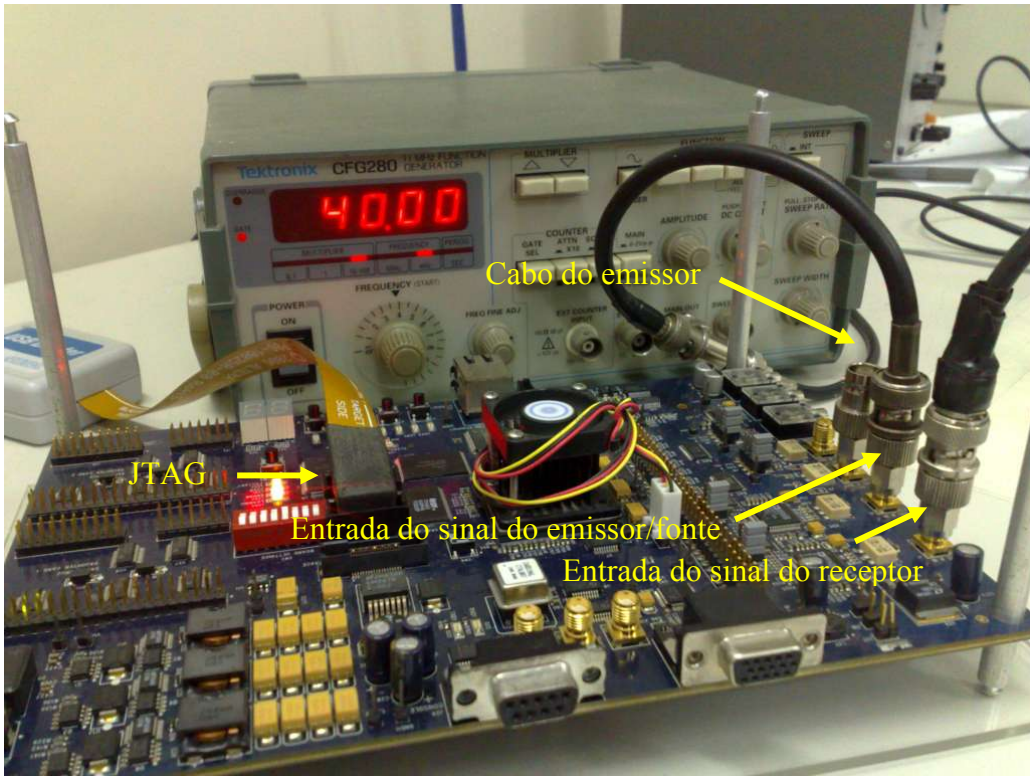


Figura 5.6 – Esquema de montagem no DSP. Entrada dos sinais vindos dos sensores e conexão da placa através de interface JTAG com o computador.

5.2– Resultados das Medições

A partir da montagem anteriormente descrita, foram feitas consecutivas medições com cada amostra de alumínio. A seqüência de medição deu-se através de uma rotina implementada no MATLAB, que possui uma função de leitura dos dados provindos do DSP pela conexão JTAG.

Desta maneira foi possível rodar, com uma mesma amostra, sucessivas medições, a fim de realizar uma análise probabilística dos resultados. Para isso foi necessário fixar a amostra no torno, com uma pequena quantidade de gel de clínico próprio para ultra-som. Com a amostra levemente pressionada entre os sensores, rodou-se a rotina dentro de um loop, controlando a quantidade de medições que seriam feitas.

Os resultados obtidos a partir de 100 medições por amostra, encontram-se organizados na Tabela 5.2 abaixo. Nota-se que apesar do erro relativamente alto, porém aceitável, o desvio padrão da fase e da medição da espessura da foi baixo.

Amostra (mm)	Diferença de Fase Média	Espessura Média (mm)	Erro Fase	Erro Espessura	Desvio Padrão Fase	Desvio Padrão Espessura (mm)
1	3,71°	1,13	13,69 %	-11,67 %	0,266°	0,082
2	6,97°	2,13	6,84 %	-6,33 %	0,275°	0,084
3	10,25°	3,14	4,77 %	-4,55 %	0,284°	0,087
4	15,11°	4,63	15,91 %	-13,23 %	1,290°	0,388
5	18,37°	5,64	12,73 %	-10,97 %	1,330°	0,410
6	21,63°	6,64	10,61 %	-9,37 %	1,263°	0,413
7	25,93°	7,96	13,64 %	-11,63 %	1,894°	0,581
8	29,19°	8,96	11,93 %	-10,38 %	1,642°	0,506
9	32,45°	9,96	10,61 %	-9,37 %	1,655°	0,504
10	35,71°	10,96	9,55 %	-8,54 %	1,671°	0,512

Tabela 5.2 – Tabela com os resultados das medições com as amostras de alumínio. Os resultados mostram uma boa precisão, dadas às condições do ambiente de medição, sujeito a grande quantidade de ruídos. O erro percentual tanto de fase quanto de espessura ficou dentro de uma faixa média de 9 a 11%. A indicação de valores médios da tabela dá-se à média das 100 medições por amostra feitas, a fim de realizar-se o cálculo de desvio padrão.

Para calcular a espessura da amostra realizou-se uma simples correlação entre o ângulo de fase com o valor da espessura de 1° de defasagem, previamente definida.

$$d = \beta \, . x_o \qquad \text{Eq. 3}$$

Onde:

- β → diferença de fase entre os dois sinais
- x_o → espessura de uma amostra corresponde a 1° de defasagem, tendo sido calculada anteriormente, no valor de 0,30694 mm

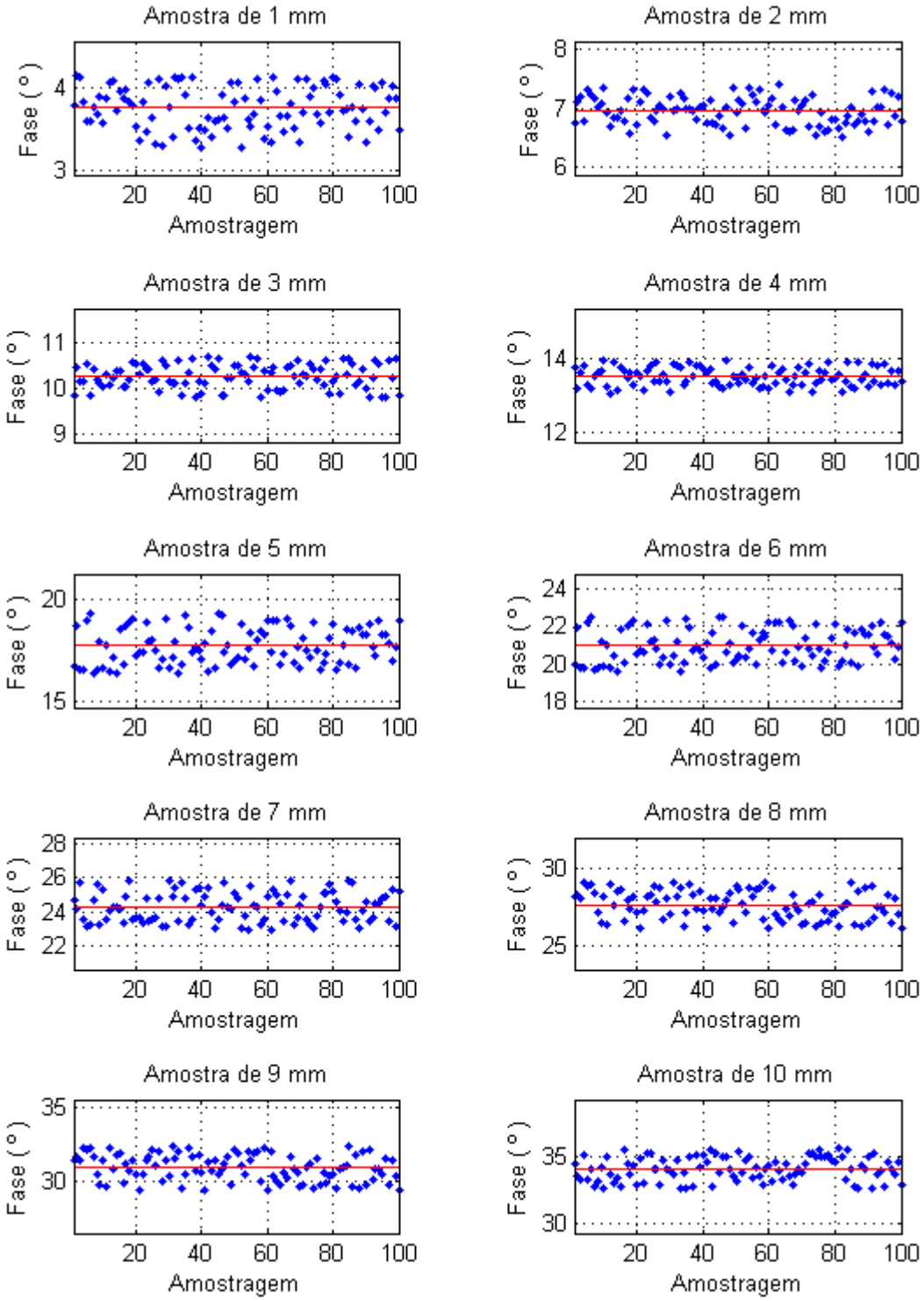


Figura 5.7 – Gráfico com as medições das amostras de alumínio de 1 a 10mm. Foram realizadas 100 medições para cada amostra, sendo assim, ser possível calcular o desvio padrão e o erro de cada medição.

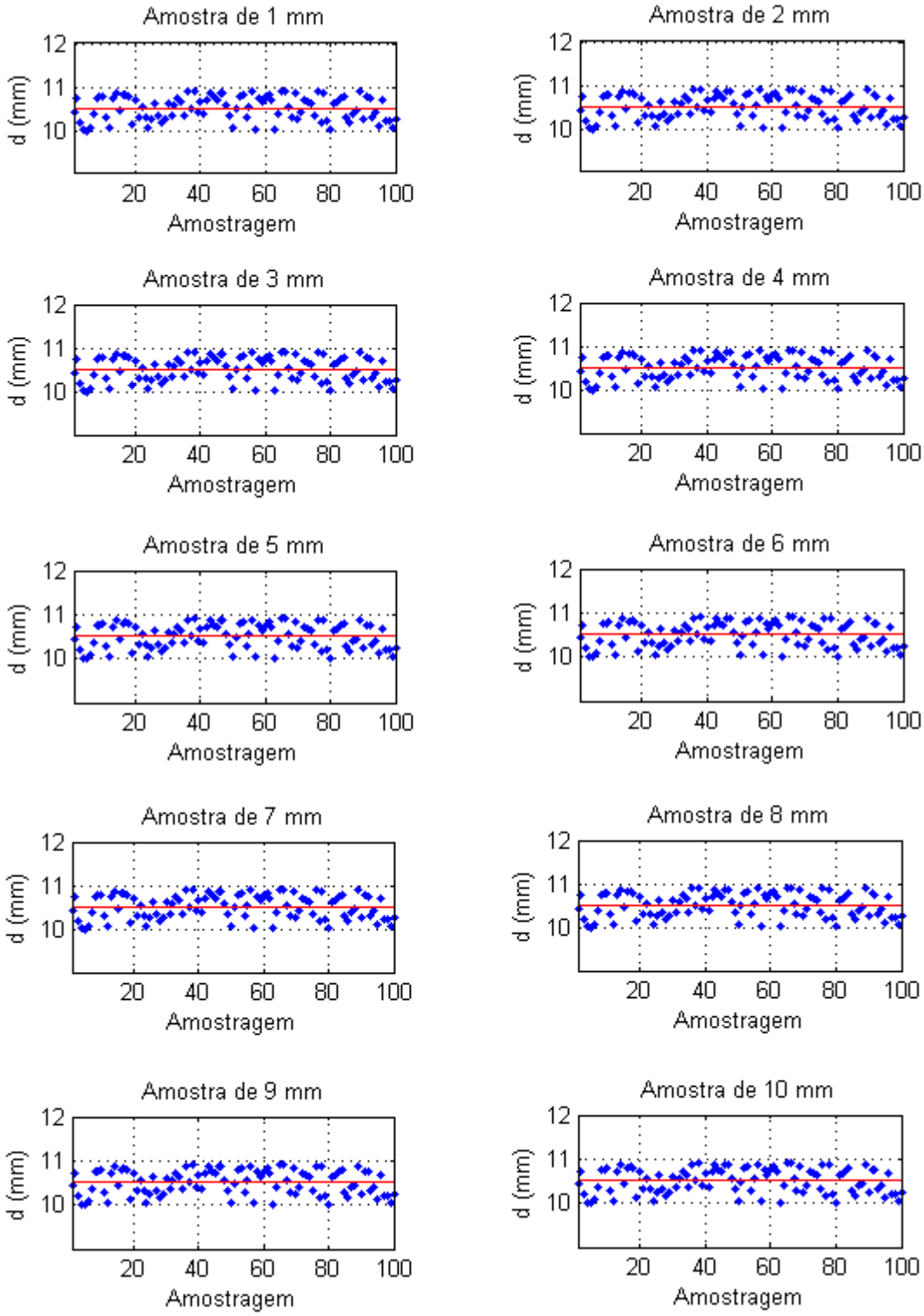


Figura 5.8 – Gráfico com os valores da espessura (d) de cada amostra. Valores calculados a partir das medições de fase, onde $d = \beta \,.\, 0{,}30694$, sendo β é o valor de fase calculado pelo Lock-In e 0,30694 é o valor correspondente a espessura de uma amostra causaria para ter defasagem de 1º.

5.3– Conclusões

A partir dos resultados obtidos e calculados pelo Lock-In, foi possível perceber que houve uma ótima resposta do equipamento com relação aos dados inicialmente previstos. Com erros de medição na faixa de 10% para fase e para a espessura, pode-se dizer que o resultado final da experiência foi satisfatória, conseguindo indicar com boa precisão o valor das amostras.

Para melhorar ainda mais estes resultados seria preciso a utilização de cabos coaxiais de melhor blindagem, além de uma conexão com o sensor deita através de algum dispositivo de encaixe, com "*borns*" onde os sensores ficariam firmemente presos. Outro fator que contribuiu para a faixa de erro obtida foi o fato das medições terem sido feitas no ar. Por se tratar de uma onda mecânica, o som se propaga melhor em meios mais densos, senso assim, uma futura montagem poderia incluir uma bacia de água quadrada, por exemplo, onde os sensores ficariam posicionados de maneia oposta. Após isso se mediria a diferença de fase sem amostra, apenas com a água, e então com a amostra. A diferença entre os dois valores corresponderia à diferença de fase do som no alumínio.

De uma forma geral pode-se concluir então que o experimento foi de fato bem realizado e os resultados dentro do previsto, validando assim a teoria utilizada para modelar o Lock-In dentro DSP.

Apêndice A

Algoritmo Base para o *Lock-In*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%    CENTRO BRASILEIRO DE PESQUISAS FÍSICAS (CBPF) – CAT                 %%%
%%%    UNIVERSIDADE FEDERAL DO RIO DE JANEIRO (UFRJ) – DEE                 %%%
%%%                                                                        %%%
%%%    Desenvolvido por :                                                  %%%
%%%      - Rafael Astuto Arouche Nunes                                     %%%
%%%      - Marcelo P. Albuquerque (orientador)                             %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% OBS: Algoritmo desenvolvido utilizando-se o programa Matlab 7.0 R14
clear all; close all; clc       % limpando possiveis variaveis ja existentes
loop = 0;                       % variavel de controle do loop infinito

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%     ENTRADAS DE CONTROLE DO LOCK-IN     %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A  = 1;                             % Amplitude do sinal de referencia
fref = 2;                           % Frequencia angular Hz

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%     ENTRADAS DO SISTEMA FISICO     %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MagNoise = 0.1;                     % Intensidade do ruido do sistema fisico
FaseAux     = 55;                   % Fase do sistema fisico
MagSistema  = 0.7;                  % Magnitude do sistema fisico

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%     INICIO DA SIMULACAO DE DETECCAO SINCRONA %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TamanhoJanelaObservacao = 2;       % Numero de periodos observados
T = 1/fref;                        % Periodo do sinal de referencia
N = 1000;                          % Pontos em um periodo do sinal de referencia
famos = fref*N;

Delta_t = 1/famos;                                 % Periodo de amostragem
NumAmostrasObs=TamanhoJanelaObservacao*T/Delta_t   % Numero de Pontos Observados

MagM = 0;
FaseM = 0;
JanelaIntegracao=zeros(1,NumAmostrasObs);
```

```
41      t=(0:Delta_t:TamanhoJanelaObservacao * T-Delta_t);
42      CoefSin = sin(2*pi*fref*t);            % Projecao Seno do sinal de referencia
43      CoefCos = cos(2*pi*fref*t);            % Projecao Cosseno do sinal de referencia
44
45      t=0;
46      while (loop ==0)
47          VIn = MagSistema * sin(2*pi*fref*t + FaseAux*pi/180) + MagNoise*randn(1,1);
48          JanelaIntegracao(1:NumAmostrasObs-1) = JanelaIntegracao(2:NumAmostrasObs);
49          JanelaIntegracao(NumAmostrasObs)=VIn;
50          Temp=CoefSin(1);
51          CoefSin(1:NumAmostrasObs-1) = CoefSin(2:NumAmostrasObs);
52          CoefSin(NumAmostrasObs)=Temp;
53
54          Temp=CoefCos(1);
55          CoefCos(1:NumAmostrasObs-1) = CoefCos(2:NumAmostrasObs);
56          CoefCos(NumAmostrasObs)=Temp;
57
58          U1 = sum(JanelaIntegracao .* CoefSin);
59          U2 = sum(JanelaIntegracao .* CoefCos);
60
61          MagM = 2*sqrt(U1^2 + U2^2)/ NumAmostrasObs;
62          FaseM = atan(U2/U1)*180/pi;
63
64          fprintf('\nt= %4.5f    |   Mag= %4.9f   |   Fase= %4.9f ',t, MagM, FaseM);
65          t = t + Delta_t;
66      end
67
68      %%% FIM DO CÓDIGO %%%
```

Apêndice B

Código Fonte para o Simulador de *Lock-In*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%    CENTRO BRASILEIRO DE PESQUISAS FÍSICAS (CBPF) – CAT                %%%
%%%    UNIVERSIDADE FEDERAL DO RIO DE JANEIRO (UFRJ) – DEE                %%%
%%%                                                                       %%%
%%%    Desenvolvido por :                                                 %%%
%%%      - Rafael Astuto Arouche Nunes                                    %%%
%%%      - Marcelo P. Albuquerque (orientador)                            %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function MainGuide_OpeningFcn(hObject, eventdata, handles, varargin)
%****************** Versao 1.8 ******************%
Compilacao=6.09032103;
set(handles.textVersao,'String',sprintf('Compilacao %1.8f',Compilacao));
%***************************************************%

function pushbuttonIniciar_Callback(hObject, eventdata, handles)
set(handles.textFlag,'String',sprintf('1'));
FlagPrint=0;                                           % 1 = ON / 0= OFF

Aamp = get(handles.editAmp,'String');
fFreq = get(handles.editFreq,'String');
SNRaux  = get(handles.editInt,'String');
FaseSist  = get(handles.editFase,'String');
MagSist  = get(handles.editMag,'String');

A=str2double(Aamp);                                     % amplitude
f =str2double(fFreq);                                   % frequencia angular
SNR =str2double(SNRaux);
FaseSistema =str2double(FaseSist);
MagSistema =str2double(MagSist);

IntensidadeNoise = (A/sqrt(2) * (1/(sqrt(exp((SNR/10)*log(1))))));

[okCampos, okFase , FaseSistema, A, f, IntensidadeNoise, MagSistema] = ChecaCampos
(FaseSist, MagSist, SNRaux, fFreq, Aamp)

if okCampos == 1 & okFase == 1
    set(handles.pushbuttonIniciar,'Enable','off');
    set(handles.editAmostras,'Enable','off');
    set(handles.editAmp,'Enable','off');
```

```
041         set(handles.editFreq,'Enable','off');
042         set(handles.editInt,'Enable','off');
043         set(handles.editFase,'Enable','off');
044         set(handles.editMag,'Enable','off');
045     else
046         return;
047     end
048 
049     FlagString = get(handles.textFlag,'String');
050     FlagAmostras = get(handles.flagAmostras,'String');
051     N = str2double(FlagAmostras);
052 
053     if (FlagPrint==1)
054         fprintf('\nCentro Brasileiro de Pesquisas Fisicas - CBPF Brasil');
055         fprintf('\n------------------------------------------------------------');
056         fprintf('\n\tNumero de amostras                  : (%d)', N);
057         fprintf('\n\tAmplitude do Sinal                  : (%d)', A);
058         fprintf('\n\tFrequencia do Sinal (Hz)            : (%d)', f);
059         fprintf('\n\tSNR                                 : (%f)', SNR);
060         fprintf('\n\tAtraso do Sinal pelo Sistema (graus) : (%f)', FaseSistema);
061         fprintf('\n\tAtenuaçao do Sinal pelo Sistema     : (%f)', MagSistema);
062         fprintf('\n------------------------------------------------------------');
063     end
064 
065     t=1;
066     n = 1:N;                        % tempo discreto (vetor abscissa)
067     SinalRefCos=zeros(1,N);
068     SinalRefSin=zeros(1,N);
069     SinalSF=zeros(1,N);
070     Mag=zeros(1,N);
071     Fase=zeros(1,N);
072 
073     while (FlagString == '1')
074             FlagString = get(handles.textFlag,'String');
075             SinalRefCos(1:N-1)=SinalRefCos(2:N);
076             SinalRefSin(1:N-1)=SinalRefSin(2:N);
077             [SinalRefCos(N), SinalRefSin(N)] = GenPointSinalRef(t, A, f, N, FlagPrint);
078             SinalSF(1:N-1)=SinalSF(2:N);
079             [SinalSF(N),Noise(N)] = MedeSinalSistemaFisico(t, A, f, N, MagSistema,...
080                                                            FaseSistema, SNR, FlagPrint);
081             [Mag(N), Fase(N)] = LockInMatlab(SinalRefCos, SinalRefSin, SinalSF, N,...
082                                              FlagPrint);
083             Mag(1:N-1)=Mag(2:N);
084             Fase(1:N-1)=Fase(2:N);
085 
086             if A > MagSistema
087                 in = A                  % escala do plot do sinal de referencia e ruido
088             else in = MagSistema;
089             end
090             mg = MagSistema*3 + 0.2;    % escala do plot da magnitude
```

```
091             % plot sinal referencia
092             axes(handles.axesSRef);
093             plot(n,SinalRefCos);
094             axis([0  N  -3  3]);
095             grid on;
096 
097             % plot sinal referencia
098             axes(handles.axesSF);
099             plot(n,SinalSF, 'r');
100             axis([0  N  -in  in]);
101             grid on;
102 
103             % plot magnitude
104             axes(handles.axesMag);
105             plot(n,Mag);
106             axis([0  N  -mg  mg]);
107             grid on;
108 
109             % plot fase
110             axes(handles.axesFase);
111             plot(n,-Fase);
112             axis([0  N  -180  180]);
113             grid on;
114 
115             MagError(1:N-1)=Mag(2:N);
116             FaseError(1:N-1)=Fase(2:N);
117             set(handles.textMag,'String',sprintf('%4.5f',Mag(N)));
118             set(handles.textFase,'String',sprintf('%3.3f',-Fase(N)));
119             t = t + 1;
120             drawnow
121         end
122 
123     function editAmostras_Callback(hObject, eventdata, handles)
124     val = get(hObject,'Value');
125     switch val
126         case 1
127             set(handles.flagOsc,'String',sprintf('1000'));
128         case 2
129             set(handles.flagOsc,'String',sprintf('2000'));
130         case 3
131             set(handles.flagOsc,'String',sprintf('1000'));
132         case 4
133             set(handles.flagOsc,'String',sprintf('5000'));
134         case 5
135             set(handles.flagOsc,'String',sprintf('10000'));
136         end
137 
138     %%% FIM DO CÓDIGO %%%
```

Apêndice C

O DSP

C.1 – Introdução

O processamento digital de sinais, da sigla em inglês DSP (*"Digital Signal Processing"*) consiste em uma metodologia de análise de sinais do mundo real. Estes sinais têm representação através de uma seqüência de números, utilizando ferramentas matemáticas, podendo assim realizar transformações ou extrair informações desses sinais, como mostrado na Figura C.1.

O mundo real consiste de uma infinidade de sinais analógicos, que não são ou não podem ser entendidos na linguagem de máquina. Nem a natureza nem os seres humanos utilizam sinais digitais para se comunicar no dia a dia. Para que computadores, celulares e quaisquer outros dispositivos eletrônicos possam trabalhar, é necessário que haja um processamento destes sinais analógicos, porém na forma digital.

Para melhor compreensão do que significa processamento digital de sinais é necessário primeiro que entendamos o que são, de onde vem e do que consistem os sinais analógicos. Estes sinais abrangem variáveis presentes no dia a dia, tais como o som, luz, temperatura, pressão, entre outros. Todas estas informações são-nos enviadas constantemente e analisadas por nosso cérebro na forma de impulsos elétricos, que compõe também uma forma de sinal analógico.

O sinal digital é justamente uma representação numérica destes sinais analógicos. E para que haja essa representação, são utilizados conversores analógico-digitais (ADC), que processam estes sinais transformando-os numa seqüência de 0s e 1s, podendo assim ser analisados por computadores e programas.

Após a análise computacional dos dados, pode-se ainda reenviar estas informações para o mundo real, utilizando conversores digital-analógicos (DAC), que tem a função de analisar uma determinada seqüência binária e transformá-la num sinal analógico.

Figura C.1 – Exemplo de como ocorre a conversão dos sinais tanto na forma analógico-digital quanto na forma digital-analógica, em um processo de gravação e reprodução de voz.

Fonte: CBPF/CAT [1]

Na Figura C.1 vemos o ADC (conversor analógico-digital) recebendo o sinal de voz vinda do microfone e realizando a conversão para transformá-lo em um sinal digital, podendo assim ser analisado pelo DSP. Após a análise feita pelo processador, ocorre a conversão inversa, onde o DAC (conversor digital-analógico) recebe o sinal digital e convertendo-o para a forma analógica, enviado o sinal para os auto-falantes.

C.2 – O Começo

No início da década de 80, as principais empresas de componentes eletrônicos como *Altera Corporation, Texas Instruments* e *Motorola*, começavam o desenvolvimento de uma ferramenta que se tornaria em pouco tempo o coração de muitos equipamentos: o DSP.

O Processador Digital de Sinais foi idealizado com a finalidade de criar-se um microprocessador com uma arquitetura desenvolvida especificamente para operações que requeressem processamento de sinais específicos, principalmente áudio e vídeo.

Atualmente, após anos de evolução tecnológica dos componentes e com técnicas de fabricação mais avançadas, os DSP's são produtos que englobam, em um único *kit* de desenvolvimento, tecnologia suficiente para realizar praticamente qualquer tipo de processamento e análise de dados.

O DSP é, acima de tudo, um dispositivo programável, que possui uma rotina própria de programação. Cada empresa fabrica não somente o *chip* propriamente dito, mas também seu ambiente de desenvolvimento ou IDE[5], que é a plataforma de programação dos códigos e instruções que irão utilizar os conversores AD e DA[6] para processamento dos sinais.

Os processadores digitais de sinal possuem duas formas de processamento distintas: *real-time* e *offline*. As aplicações em tempo real são muito utilizadas em DSP's devido à sua alta capacidade de processamento. Desta forma, aplicações onde o *delay* não é tolerável utilizam largamente DSP's como plataforma de processamento de sinais. Um bom exemplo disto são os celulares e aparelhos de comunicação, que permitem uma comunicação contínua, onde ambos os lados podem falar simultaneamente, sem interrupções.

C.3 – Aplicações

O DSP é atualmente utilizado em larga escala em diversos dispositivos eletrônicos, como celulares, computadores, aparelhos de vídeo, modems, entre outros. Um bom exemplo de aplicação são os televisores de LCD e plasma, que ao utilizar processadores digitais de sinal embutidos em seus circuitos eletrônicos, conseguem, hoje em dia, fornecer alta fidelidade de imagem e som, a um custo cada vez menor. Sendo assim, o DSP é um dispositivo presente em diversas áreas da indústria, tais como médica, científica, automotiva e principalmente militar.

C.3.1 – Freios Anti-Travamento (ABS)

A miniaturização dos componentes, aliada com o avanço em pesquisa de segurança automotiva, faz com que os DSP's se façam cada vez mais presentes nos automóveis hoje em dia. Sistemas de freios antitravamento (ABS), distribuição eletrônica de frenagem (EBD), *air-bags* cada vez mais rápidos e eficientes, computadores de bordo com mais funções e mais adaptativos ao estilo de condução, são apenas alguns exemplos de sistemas que utilizam o DSP como base para processamento de sinais provindos dos sensores.

[5] Integrated Development Environment ou Ambiente Integrado de Desenvolvimento
[6] Analógico-digital e Digital-Analógico

Na Figura C.2 podemos ver de forma detalhada como o Módulo de Controle (que contem o DSP) trabalha para realizar o funcionamento do ABS, controlando a frenagem das rodas sem que o motorista perca o controle do carro. O sensor de velocidade posicionado nas rodas envia constantemente sinais para o módulo de controle do carro, composto por um processador digital de sinais, que faz a leitura e interpreta em qual situação o freio foi acionado. Em caso de frenagem brusca o módulo envia um sinal para a bomba hidráulica que realiza o travamento controlado das rodas.

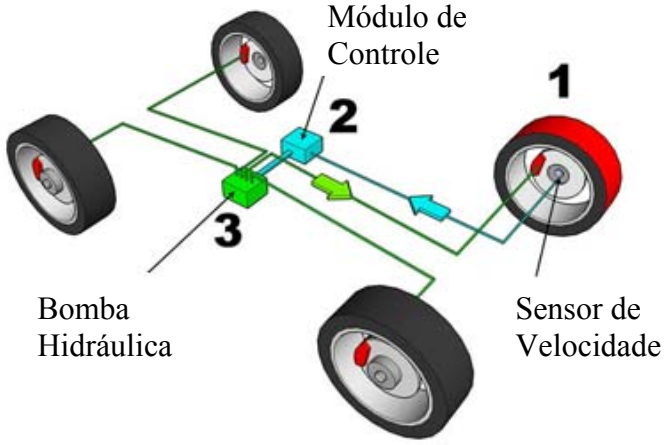


Figura C.2 – Esquemático de representação do funcionamento de um ABS.

Fonte: drivingfast.net [2]

A evolução dos DSP's faz com que as novas gerações de freios ABS sejam mais eficientes, mais precisas e com tempo de resposta menor. Sendo assim, o DSP é um equipamento fundamental nos carros modernos, tornando-se o coração do automóvel, controlando todas suas funções.

C.3.1 – Compressão e Descompressão de Sinal

Outra aplicação comumente feita utilizando-se DSP's é a compressão e descompressão de sinais. Câmeras multimídia, como *webcams*, permitem que pessoas possam enviar e receber imagens em tempo real, sem que haja interrupção do sinal. Também encontramos esta tecnologia em sistemas de áudio que utilizam CD ou DVD. O DSP é utilizado para realizar uma complexa rotina de detecção e correção de *"raw data"* (registro sem informação), diretamente a partir da leitura do disco.

C.3.2 – Filtros Digitais

Filtros digitais têm como principal função remover partes indesejadas de um sinal, como ruídos ou harmônicos, porém pode funcionar também como um seletor de freqüências, podendo ser passa-baixa[7], passa-alta[8] ou passa-faixa[9]. A Figura C.3 mostra um esquema simples onde o filtro funciona como uma função que recebe como parâmetro um sinal e possui como saída o sinal filtrado.

Figura C.3 – O sinal original, contendo ruídos ou harmônicos, passa pelo filtro, que faz a seleção de qual parte será processada, enviando o sinal filtrado para o DSP.

Ao receber o sinal original, o conversor analógico-digital (ADC) realiza a digitalização da amostra, que é então lida pelo DSP. Por ser digital, o filtro pode ser configurado para trabalhar em ordens mais elevadas, com relativa facilidade. Após ser digitalizado, o sinal, agora filtrado, é manipulado por algoritmos e rotinas específicas definidas pelo usuário, sendo então enviado para o conversor digital-analógico (DAC) na saída.

Esta operação pode ser observada na Figura C.4, que mostra de forma esquemática o caminho percorrido pelo sinal, desde sua entrada no conversor AD, até a saída pelo conversor DA, passando pela análise digital feita no DSP.

[7] O filtro passa-baixa permite a passagem de baixas freqüências, atenuando a amplitude de freqüências acima da freqüência de corte $f_c = \frac{1}{2\pi RC}$. É composto basicamente de um circuito RC, com tensão de saída sob o capacitor.
[8] O filtro passa-alta permite a passagem de altas freqüências, atenuando a amplitude de freqüências abaixo da freqüência de corte. É composto basicamente de um circuito RC, com tensão de saída sob o resistor.
[9] O filtro passa-faixa permite a passagem de freqüências compreendidas entre dois limites desejáveis, eliminando ou atenuando as freqüências acima ou abaixo das freqüências de corte. É composto basicamente de um circuito LRC, com tensão de saída sob o resistor.

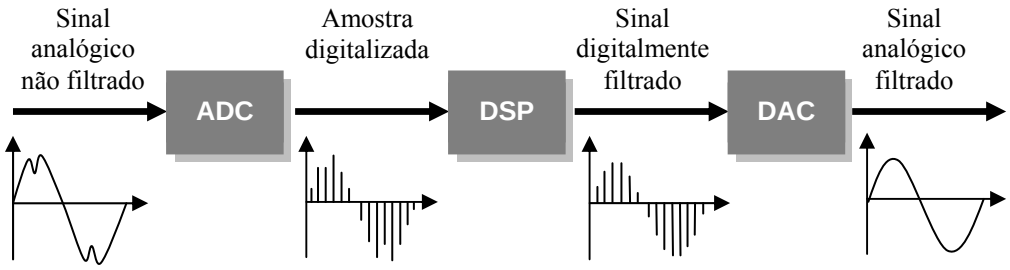


Figura C.4 – Representação esquemática do caminho do sinal desde a entrada no conversor AD, passando pelo DSP, que realiza da manipulação deste, passando pelo conversor DA na saída.

Fonte: dsptutor.freeuk.com [3]

Diferentemente dos filtros analógicos, que utilizam componentes ativos, passivos e amplificadores operacionais em sua composição, os filtros digitais são mais precisos e versáteis, pois utilizam um microprocessador para realização dos cálculos a partir do sinal de entrada.

Outro ponto favorável aos filtros digitais implementados a partir de um DSP está o fato destes serem programáveis. Desta forma é possível criar rotinas de armazenamento dos dados enviados pelos conversores na memória (que variam de acordo com a família do processador utilizado), além cálculos em tempo real, o que é muito útil quando se trabalha com instrumentação.

Por ser digital, este tipo de filtro não está sujeito à perturbações resultantes das características físicas de seus componentes, como capacitores e resistores, principalmente no que diz respeito à temperatura.

Em contraparte aos filtros analógicos, os filtros digitais possuem uma larga escala de trabalho com sinais de baixa freqüência, porém alguns DSP's hoje em dia suportam níveis de freqüência extremamente altos, da ordem de megahertz. Esta característica torna o DSP uma ferramenta adaptável que, dependendo do modelo utilizado e de sua capacidade de processamento, pode se adequar ao tipo de sinal de entrada.

C.4 – Desenvolvimentos de Projetos

Por se tratar de um dispositivo programável, o DSP necessita de uma interface de controle própria, a IDE. O desenvolvimento de projetos, independente do fabricante ou do modelo do DSP, se dá através destas IDE's, que em sua maioria utilizam linguagens de programação comuns, como *Assembly, C* e *C++*.

Cada *kit* de DSP é composto pelo *software* indicado pelo fabricante para realização dos projetos. Estes aplicativos foram moldados para trabalhar com funções pré-determinadas de cada modelo de DSP compatível. Desta forma o usuário utiliza a interface de programação, monta o projeto e compila o código no DSP, seja por comunicação USB ou JTAG[10].

A fim de mencionar exemplos de IDE's dos principais fabricantes do mercado podemos citar o *Code Composer Studio* (CCS) da *Texas Instruments* (Figura C.6), o VisualDSP++ da *Analog Devices* (Figura C.5) e o Quartus II da *Altera Devices*, utilizado no projeto.

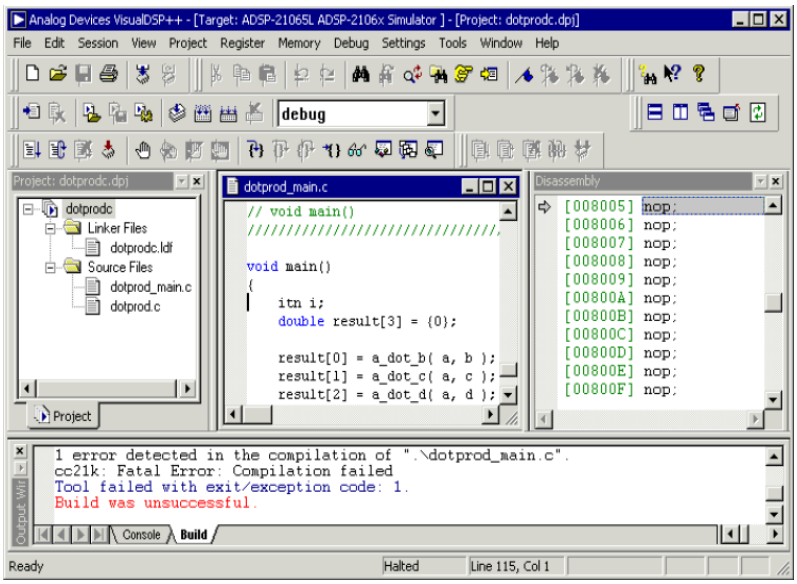


Figura C.5 – IDE VisualDSP++ da *Analog Devices*. Aceita as principais linguagens de programação como *Assembly, C* e *C++*.

[10] JTAG (Joint Test Action Group) é o nome dado ao sistema de comunicação feito para controlar pinos de um circuito integrado. Foi homologado pela norma IEEE Std. 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture-Description.

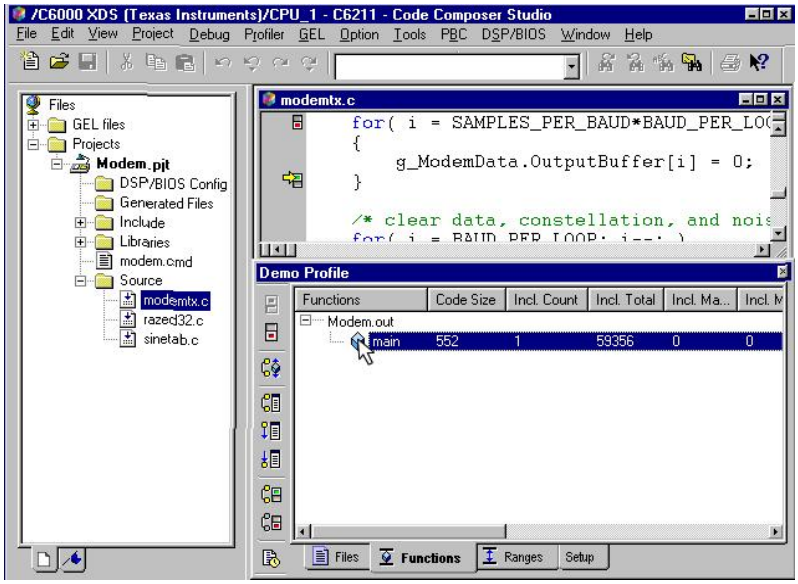


Figura C.6 – IDE *Code Composer Studio* da *Texas Instruments*. Tem como principal base de programação *C* e *C++*, possuindo interface com o *software* matemático *Matlab*[11] e *Matlab Simulink*, garantindo a esta IDE ótima versatilidade e facilidade na hora de montar projetos com DSP.

O desenvolvimento de um projeto utilizando-se um DSP envolve três principais etapas: pesquisa, simulação e emulação. A etapa de pesquisa consiste basicamente em entender o problema a ser estudado, as características do projeto, como será sua interface e com que rapidez e robustez deve ser implementado. Nesta etapa ainda são introduzidas informações de como o DSP em questão trabalha, se realiza cálculos em ponto fixo ou ponto flutuante. Característica essa que pode ser crucial na escolha do DSP mais adequado.

Realizar simulações é uma etapa primordial para o sucesso do projeto. É através desta etapa que modela-se inicialmente o projeto, definindo-se as variáveis de entrada e saída, bem como a quantidade de memória a ser utilizada ou ainda com que rapidez os cálculos estão sendo feitos. A criação de simuladores ou rotinas de testes antes da programação no DSP facilita e agiliza o projeto, na medida em que se consegue descobrir *bugs* e trechos do código sem funcionalidade. Nesta etapa consegue-se "enxugar" bastante o código inicial, deixando a rotina mais rápida e eficiente.

[11] Outros fabricantes de DSP fornecem plataformas compatíveis com o Matlab. Motorola, Texas Instruments e Altera Devices são as principais, sendo que as duas primeiras geralmente trazem bibliotecas pré-instaladas com as versões mais recentes do Matlab Simulink.

Na fase de emulação são realizados testes já dentro de seu ambiente de desenvolvimento. Programando-se o DSP através de suas rotinas pré-definidas, análises mais detalhadas do código são realizadas, a fim de deixar o projeto com maior precisão de cálculos, menor erro no resultado desejado e com menor uso de memória possível.

A escolha do melhor DSP para o projeto não envolve apenas sua capacidade de processamento, mas também como será realizada sua compilação. Algumas empresas oferecem ótimos equipamentos, porém com uma IDE relativamente complicada. Tem sido muito comum hoje em dia a criação de ambientes gráficos de desenvolvimento. Esta nova forma de programar o DSP deixa a tarefa de criação do projeto muito mais simples, rápida e fácil de entender.

Toda etapa de compilação e programação dá-se através de uma via de comunicação, em geral USB ou JTAG. Cada fabricante define como irá ocorrer a programação do *hardware*, que possuem características próprias, devido à pinagem dos componentes e à compatibilidade de versões dos equipamentos. Na Figura C.7 podemos observar dois modelos de conectores JTAG disponíveis no mercado.

(a) (b)

Figura C.7 – (a) JTAG da *Texas Instruments* C2000™ Series XDS510LC. (b) JTAG da *Analog Devices* USB-Blaster™, compatível com as famílias de FPGA Stratix, Cyclone, MAX e FLEX 10K. Este modelo de JTAG foi utilizado no *kit* EP2S60 durante o projeto de construção do protótipo do Amplificador *Lock-In*.

C.5 – A Estrutura de um DSP

Cada Processador Digital de Sinais tem sua característica física própria, porém todos têm algo em comum por trás. Compartilhando a mesma estrutura básica, os DSP's tornaram-se uma grande ferramenta hoje em dia. Simples, funcional e de fácil manuseio.

Um DSP comercial é basicamente composto pelas seguintes estruturas: processador, memória, unidades aritméticas, portas de comunicação e conversores. Outros componentes trabalham ao redor suportando ou complementando o *hardware* em outras funções, seja para deixar o DSP mais rápido ou com maior capacidade de processamento ou armazenamento de dados.

Na Figura C.8 podemos encontrar um diagrama simplificado do DSP da *Analog Devices*, modelo ADSP-21160M. Em destaque encontram-se os principais blocos da placa: o processador principal (contendo os registradores e unidades aritméticas lógicas), memória e portas de comunicação.

As Unidades Aritméticas Lógicas são responsáveis por realizar operações aritméticas e lógicas, tanto em ponto fixo como em ponto flutuante, dependendo do modelo do DSP em questão.

As unidades de DMA[12] são importantes para realizarem tarefas de forma independente ao processador principal, ou seja, podem fazer a comunicação entre a memória e as portas seriais sem interromper alguma rotina que esteja eventualmente sendo rodada.

Alguns modelos encontrados no mercado fornecem ainda suporte ao processamento de áudio e vídeo. Para tal possuem embutidas na placa principal, ou em placas filhas, conversores A/D e D/A, como podem ser observados na Figura C.9.

[12] Direct Memory Access.

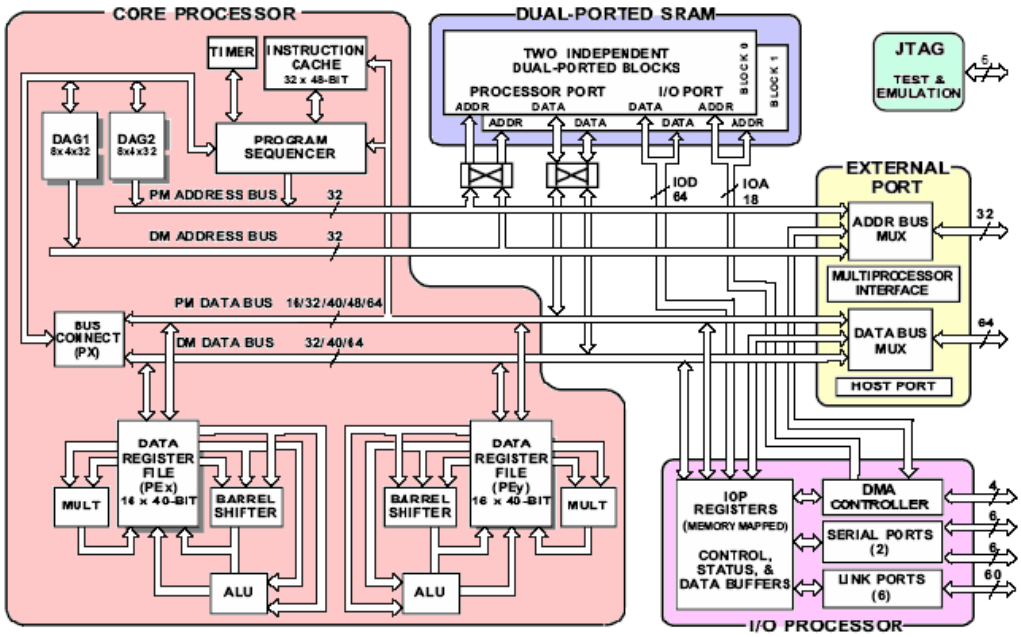
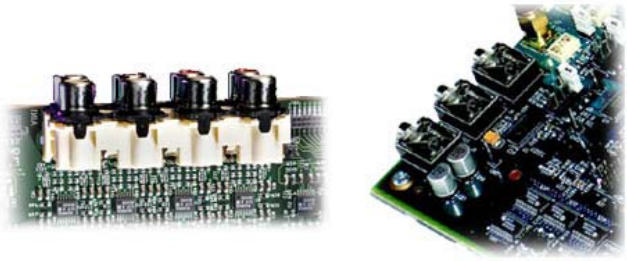


Figura C.8 – Diagrama de blocos da estrutura básica do processador digital de sinais ADSP-21160M da *Analog Devices*. Em destaque os principais grupos do DSP.

Fonte: Analog Devices [4]



(a) (b)

Figura C.9 – (a) Conversor de vídeo do *kit* de DSP ADSP-BF533 da *Analog Devices*. (b) Conversor de áudio do DSP EP2S60 da *Altera Devices.*

C.6 – Exemplos

Com a finalidade de mostrar a versatilidade e funcionalidade dos DSP's, serão mostrados agora alguns exemplos práticos de programas, funções e projetos que podem ser implementados nos *kits* de desenvolvimento.

O primeiro exemplo resulta de um código teste do DSP ADSP-21160M, compilado durante o aprendizado sobre o *kit*, realizado no LPS/UFRJ[13], durante o ano de 2005. Consta de uma rotina para realização do cálculo de convolução.

```
 1  void main()
 2  {
 3          InitializeSineTable( Table, sizeof(Table) );
 4          GenerateInputPulse( Table, Input, sizeof(Table) );
 5          CalculateOutputPulse( Input, sizeof(Input), Impulse,
 6                                sizeof(Impulse), Output );
 7          exit( 0 );
 8  }
 9
10  void InitializeSineTable(float Table[], size_t nSize)
11  {
12          const float RADIANS = 0.017453292;
13
14          for ( int i=0; i<nSize; i++ )
15          {
16                  Table[i] = sin ( RADIANS * i );
17          }
18  }
19
20  void GenerateImpulseCoeffs (const float Table[], float Impulse[], size_t nSize)
21  {
22          for ( int i=0; i<nSize; i++ )
23          {
24                  Impulse[i] = Table[(i*10)];
25          }
26  }
27
28  void CalculateOutputPulse(const float Input[], size_t nInputSize,
29                            const float Impulse[], size_t nImpulseSize,
30                           float Output[])
31  {
32          for( int i=0; i<nInputSize; i++ )
33          {
34                  for( int j=0; j<nImpulseSize; j++ )
35                  {
36                          Output[i+j] = Output[i+j] + (Input[i] * Impulse[j]);
37                  }
38          }
```

Tabela C.1 – Rotina de cálculo de convolução implementada no DSP ADSP-21160M através da IDE VisualDSP++ da *Analog Devices*. Todo o código foi feito em linguagem C.

O exemplo a seguir foi escrito por Rodrigo C. Torres e representa digitalmente um sintetizador de guitarra que fornece dois efeitos: distorção e eco. As ações são controladas por botões de interrupção localizados na placa do DSP, cujos LED's correspondentes são acesos, indicando qual efeito está sendo utilizado.

[13] Laboratório de Processamento de Sinais da COPPE/UFRJ.

Estas interrupções, chamadas de IRQ, são funções ativadas cada vez que o usuário aperta o botão na placa ou o programa chama a rotina de execução da mesma. No caso do sintetizador foram usadas duas interrupções. O primeiro botão ao ser pressionado ativa a função de distorção, já o segundo botão ativa o efeito de eco. Na Figura C.10 podemos entender melhor como funciona esta relação.

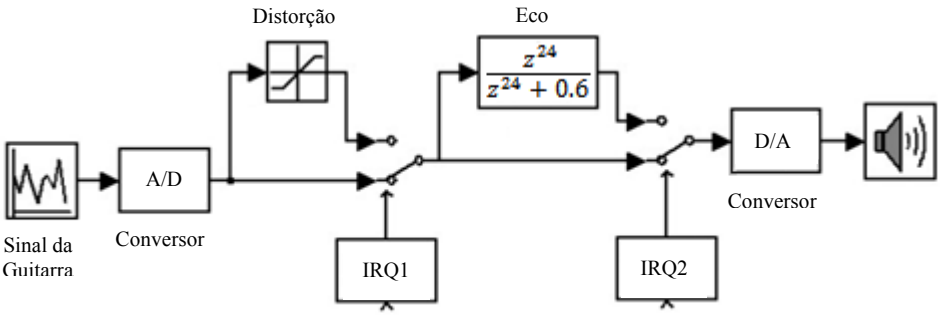


Figura C.10 – Esquemático do sintetizador de guitarra implementado no DSP ADSP-21160M. O chaveamento ilustrado não ocorre fisicamente, mas sim digitalmente dentro do código. Cada vez que uma interrupção é ativada, esta chama uma função, executando-a, no caso ou de distorção ou de eco.

Fonte: Processadores Digitais de Sinais e suas Aplicações [5]

O código da Tabela C.2 demonstra como as funções do IRQ são chamadas dentro do DSP. Nas linhas 1 e 10 são definidas as funções de distorção e eco, respectivamente. A linha 37 mostra a função "`CSound codec`", onde estão definidos os parâmetros de configuração dos conversores da placa. A função "`interrupt`" mostrada nas linhas 43 e 44 retornam às funções definidas anteriormente, fazendo o efeito desejado.

```
 1  void distortion()
 2  {
 3          static float top = 15000000.0;
 4          left *= (left < 0) ? log(-1*left) : log(left);
 5          if (left > top) left = top;
 6          else if (left < -top) left = -top;
 7          left *= 2;
 8  }
 9
10  void echo()
11  {
12          left = *pos = left + GAIN*(*pos);
13          pos = (float *) circptr(pos, 1, buffer, DELAY);
14  }
15
16
17  void intDistortion(int x)
18  {
19          distEnabled = !distEnabled;
20          set_flag(SET_FLAG0, TGL_FLAG);
21  }
22
23  void intEcho(int x)
24  {
25          echoEnabled = !echoEnabled;
26          set_flag(SET_FLAG1, TGL_FLAG);
27          if (echoEnabled)
28          {
29                  pos = buffer;
30                  #pragma SIMD_for
31                  for (int i=0; i<DELAY; i++) buffer[i] = 0.0;
32          }
33  }
34
35  void main()
36  {
37          CSound codec;
38          distEnabled = echoEnabled = false;
39          set_flag (SET_FLAG0, CLR_FLAG);
40          set_flag (SET_FLAG1, CLR_FLAG);
41          set_flag (SET_FLAG2, CLR_FLAG);
42          asm("bit set MODE2 IRQ0E | IRQ1E | IRQ2E;")
43          interrupt (SIG_IRQ0, intDistortion);
44          interrupt (SIG_IRQ1, intEcho);
45          while (true)
46          {
47                  asm ("idle;");
48                  codec.getSamples (left, right);         // input do ADC
49                  if (distEnabled) distortion();
50                  if (echoEnabled) echo();
51                  codec.setSamples (left, left);          // output do DAC
52          }
53  }
```

Tabela C.2 – Código do sintetizador de guitarra implementado no DSP ADSP-21160M. A rotina foi desenvolvida e compilada usando-se a IDE VisualDSP++ da *Analog Devices*.

Referências Bibliográficas

[1] NUNES, Rafael A. A., ALBUQUERQUE, Marcelo P., ALBUQUERQUE, Márcio P.. **"Introdução a Processadores de Sinais Digitais - DSP."** Nota Técnica, CBPF-NT-001/2006, Fevereiro de 2006

[2] *http://drivingfast.net/*

[3] *http://dsptutor.freeuk.com*

[4] **"ADSP-21160 Analog Devices Data Sheet"**, ADSP-21160, Abril de 2006 *http://www.analog.com/static/imported-files/data_sheets/ADSP-21160M.pdf*

[5] TORRES, Rodrigo C., **"Processadores Digitais de Sinais e suas Aplicações"**. Abril de 2004.

[6] **"Signal Recovery Digital Lock-In Amplifier"**. *http://www.signalrecovery.com/*