Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Sistema autônomo de análise de métricas da rede elétrica implementado com DsPIC.

Autor:

_____

Cássio Netto de Carvalho

Orientador:

_____

Prof. Fernando Antônio Pinto Barúqui, D. Sc.

Examinador:

_____

Prof. Joarez Bastos Monteiro, D. Sc.

Examinador:

_____

Prof. Carlos Fernando Teodósio Soares, D. Sc.

DEL

Julho de 2013

Sistema autônomo de análise de métricas da rede elétrica implementado com DsPIC.

Cássio Netto de Carvalho

TRABALHO ORIENTADO PELO PROFESSOR FERNANDO ANTÔNIO PINTO BARUQUI DO DEPARTAMENTO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA APROVAÇÃO NA DISCIPLINA PROJETO DE GRADUAÇÃO.

Rio de Janeiro - Brasil

Julho de 2013

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

DEDICATÓRIA

Aos meus pais Valmir Monteiro de Carvalho e Lígia Conceição Carioly Netto de Carvalho.

AGRADECIMENTO

Primeiramente agradeço aos meus pais, que dentro de suas condições e escolhas, me deram a estrutura necessária para concluir a graduação em uma das melhores instituições de ensino do país. A toda minha família que sempre esteve presente ajudando de alguma maneira nos momentos difíceis, e sempre acreditando no meu potencial e nos meus sonhos.

Aos amigos, que por vezes entenderam minha ausência nesses longos anos de graduação. Aqueles que me ajudaram diretamente e indiretamente em cada disciplina cursada seja com um material, uma explicação, um conselho ou motivação.

Agradeço especialmente aos amigos Giovani Ferreira, Maurício Rodrigues, Manuela Lima, Leandro Borges, Pedro Esteves, Marcel Mello, Trevor Dobbin, Camilla Gueiros, Diego Santos Wanderley, Christian Belga e Fernando Henrique Figueiredo Nunes, que em inúmeras situações tiveram papel decisivo no meu sucesso durante o curso.

A todo corpo docente do DEL (*Departamento de Engenharia Eletrônica e de Computação)* pelos conhecimentos passados, pelas horas de dedicação, por toda paciência e boa vontade em ensinar mesmo com todas as dificuldades e limitações da nossa realidade.

Ao meu orientador neste projeto Prof. Fernando Antônio Pinto Barúqui por toda ajuda, motivação, orientação e conhecimento passado.

A PRV Tech e principalmente aos amigos Roberto Dias, Pedro Dagola e Victor Campos, por terem disponibilizado o laboratório, pelo tempo e conhecimento que dispuseram para me ajudar em algumas etapas deste projeto.

Agradeço ao Sr. Aleksandar Nikolic, que junto à Mikroelektronika cedeu a licença do compilador *MikroC for dsPic* ao DEL para o desenvolvimento desse projeto.

Ao povo brasileiro que contribuiu de maneira significativa à minha formação, seja ela social ou intelectual.

Agradeço a Deus por me proporcionar tudo isso.

RESUMO

Este trabalho tem por finalidade o desenvolvimento de uma ferramenta autônoma para a medição do consumo de energia em $Ampère_{RMS}$-Hora (Ah) e da distorção harmônica total (THD) de um determinado sinal da rede elétrica. A intenção do projeto foi criar um circuito que se utiliza de dsPICs para o cálculo e processamento dos dados, e os disponibiliza em um display de LCD.

Este documento detalha a fundo cada etapa do desenvolvimento, montagem e validação dos resultados adquiridos.

Palavras-Chave: dsPIC, Rede Elétrica, Ah, RMS, THD.

ABSTRACT

The main goal of this work is the development of a portable tool for the measurement of the power consumption in Ampère-Hour (Ah), and the Total Harmonica Distortion (THD) of a signal from the power electric network. The objective has been the design of a circuit based on dsPICs for the signal processing, which presents the results on a LCD panel.

Key-words: dsPIC, power electric network, Ah, RMS, THD

SIGLAS

UFRJ – Universidade Federal do Rio de Janeiro

RMS – *Root Mean Square*

THD – *Total Harmonic Distortion*

dsPICs – Microprocessadores com modulo DSP nativo

DFT – *Discrete Fourier Transform*

ADC – *Analog to Digital Converter*

Ah – *Ampère$_{RMS}$-Hora*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Tema

Um dos grandes tópicos da atualidade é a geração de energia e o uso racional da mesma. O estudo de energias alternativas é algo em que a maioria dos países e instituições privadas está investindo grandes esforços com o intuito de encontrar a melhor solução em substituição às atuais formas de geração e armazenamento de energia.

Junto dessa preocupação vem a necessidade de conscientização da população mundial em termos de usabilidade e desperdício desse recurso tão indispensável hoje para a vida humana. Assim, a tecnologia vem como meio de ajudar nessa tarefa e viabilizar caminhos para monitoração e validação desse uso doméstico ou industrial. Este projeto lida com duas importantes métricas em termos de monitoração da Rede Elétrica, que são: o consumo de energia e a distorção harmônica.

1.2 – Delimitação

Com objetivo de nortear futuras aplicações no que diz respeito ao cálculo e monitoração do consumo de energia por uma determinada carga ligada a rede elétrica, este projeto foi desenvolvido tendo como base física elementos como o microcontrolador e o sensor de corrente por efeito Hall.

Isso nos traz uma limitação que é dada pela resolução do sensor de corrente. Este nos permite medir correntes com limites máximos de 30 A. Porém, por esse limite ser uma corrente muito alta, o circuito não será utilizado nesse limite nos testes.

1.3 – Justificativa

A capacidade de gerar energia está diretamente relacionada com o crescimento do país e sua capacidade industrial. Por isso, é importante o planejamento de uma malha energética que consiga efetivamente alimentar a demanda existente em todo o país.

Isso se torna um problema, já que os recursos naturais para a geração de eletricidade são limitados, e para aumentar a capacidade das usinas existentes, ou até a construção de outras, causa um grande impacto ao meio ambiente.

Cada vez mais se percebe que dentro de algum tempo teremos que usar a eletricidade de maneira mais consciente para que não ocorram desperdícios. Infelizmente ainda há uma considerável quantidade de energia que é ilegalmente utilizada, normalmente furtada com ligações clandestinas e muitas vezes não apropriada.

Esse desvio de energia causa sobrecarga na rede elétrica que inicialmente foi planejada para certa demanda bem inferior a que realmente dela é utilizada. Além da sobrecarga a malha energética, há ainda a questão financeira. O custo da energia desviada deve ser coberto pelos consumidores regulares, o que faz com que na tarifa esteja incluída essa diferença.

Fiscalizar a energia gasta por cada domicílio, quanto à fraude, pode esbarrar em alguns aspectos legais, e prejudicar o planejamento do crescimento do fornecimento elétrico. Porém o advento de um instrumento capaz de monitorar essa energia fora dos limites físicos da residência ou indústria pode ser de grande ajuda para estimar o consumo em um determinado intervalo de tempo.

O projeto aqui descrito pode ser utilizado com esse intuito. Calculando a corrente *RMS*, multiplicando-a pelo intervalo de tempo da medida, e sabendo a tensão fornecida, que é conhecida, podemos então calcular a energia consumida.

Outra informação importante é a THD, que mostra o quanto a rede elétrica está sendo perturbada por harmônicos por conta de uma determinada carga. Monitorar a distorção harmônica da corrente se mostra fundamental visto os danos que ela pode trazer para um sistema. Como forma de exemplificar problemas práticos que os harmônicos podem causar, lista-se:

- Aumento de tensão pode diminuir a vida útil de motores, isolações, fios e cabos.

- Ocasionar queimas, falhas e desligamentos.
- Aumento das perdas nos estatores e rotores de máquinas rotativas, causando superaquecimento danoso às máquinas.
- Aquecimento em reatores de lâmpadas fluorescentes
- Aparecimento de ressonâncias entre capacitores para correção de fator de potência e o restante do sistema, causando sobretensões e sobrecorrentes que podem causar sérios danos ao sistema.

1.4 – Objetivos

O principal objetivo deste projeto é nortear futuras aplicações implementadas com tecnologia mais avançada, mas que tenha a mesma finalidade de calcular corrente *RMS* e *THD*.

Este projeto ajudou a observar o tratamento dos métodos de cálculo e a análise de possíveis problemas envolvendo limitações físicas e de resolução de resultados, delimitando assim seu projeto e limitações de uso.

1.5 – Metodologia

O desenvolvimento do projeto seguiu algumas etapas definidas buscando as melhores opções de ferramentas e métodos para alcançar o melhor desempenho e precisão das medidas calculadas.

O primeiro passo foi a escolha do microcontrolador. A princípio, pela natureza dos cálculos, mais precisamente a THD, o microcontrolador com um módulo DSP se mostrou uma ferramenta interessante para facilitar o desenvolvimento do software. Com isso, dentro da família dsPIC da Microchip, foi escolhido aquele de maior capacidade de processamento e amostragem de sinais analógicos. As principais características que levaram à escolha do dsPIC33FJ16GS502 serão detalhadas no Capítulo 3.

Após aquisição e estudo do microcontrolador, a fase de desenvolvimento do software iniciou-se. Neste momento, foi importante implementar a comunicação com o *display* LCD, visando a futura validação do código de cálculo das métricas.

Nesta fase o maior desafio foi escolher os melhores métodos matemáticos que calculassem os valores da corrente RMS e da THD, respeitando as limitações técnicas

do microcontrolador, mais precisamente a restrição da memória do dsPIC. Essas escolhas serão estudadas mais profundamente no Capítulo 2.

De posse do hardware e do software, foi confeccionada a placa de circuito impresso, entrando assim na fase final do projeto: a validação dos resultados. Alguns ajustes finos foram necessários, visto que a montagem na protoboard mostrou-se muito susceptível à influência de ruídos.

Nos testes finais, foram montados cenários com resultados conhecidos, ainda com auxílio de outros equipamentos acurados para a validação do projeto. Esses resultados serão apresentados ao final deste documento.

1.6 – Descrição

No capítulo 2 serão explicadas algumas características do sinal de entrada, assim como o tratamento do mesmo antes que se possa inserí-lo na entrada do microcontrolador. Ainda no Capítulo 2, os algoritmos de cálculo dos valores RMS e THD serão detalhados passo a passo, a fim de mostrar ao leitor cada método matemático escolhido para o cálculo das métricas.

O Capítulo 3 apresenta as características gerais do dsPIC escolhido. Detalhes sobre o módulo ADC e UART podem ser encontrados, respectivamente, nas sessões 3.2 e 3.3. Há ainda uma breve introdução à estrutura do programa principal, a interação com o usuário e informações sobre o controle e características físicas do *display*.

Os resultados são apresentados no Capítulo 4. Nele será explicitado como foram feitos os testes e a validação dos valores obtidos com o circuito.

A conclusão será apresentada no Capítulo 5.

Capítulo 2

Circuitos

2.1 – Sinais de Entrada

Consideramos como sendo o sinal de entrada, um sinal de tensão senoidal resultante da Rede Elétrica. Este sinal é regido pela seguinte função

$$v_{REDE}(t) = 127\sqrt{2}\sin(2\pi.60.t + \theta)$$

Para que o microcontrolador possa amostrar o sinal corretamente teremos que usar alguns artifícios para tratar o sinal e ele ser usado pelo dsPIC.

2.2 – Tratamento dos Sinais de Entrada

Através de uma análise prévia do circuito observou-se que seria necessário o uso de mais memória do que o dsPIC possui. Assim decidiu-se usar dois dsPICs, pela facilidade de já se ter o microcontrolador e não necessitar compra de novos componentes, para o cálculo da THD e da Corrente RMS-hora. Sendo assim, teremos um dsPIC dedicado para o cálculo da *THD* e outro para o cálculo da Corrente RMS-hora. Como ambos os cálculos são relativos à corrente, utilizaremos apenas um método para tratar o sinal. Assim, apenas será preciso conectar as entradas dos dois dsPICs no mesmo ponto para receber o sinal tratado.

Até o momento, temos uma carga ligada na rede elétrica e um sinal de corrente que é consequência dessa ligação. Porém, precisamos transformar esse sinal de corrente em um sinal de tensão proporcional para que o dsPIC consiga amostrar, processar e calcular as métricas.

Para isso, usaremos um sensor de corrente linear por efeito Hall com tensão de *offset* ligado em série com a carga.

- **Sensor Hall –** O sensor de corrente por efeito Hall é um transdutor que varia sua voltagem de saída em resposta ao campo magnético criado pela condução da corrente elétrica. A tensão de saída respeita a seguinte equação:

$$v = \frac{Rh.I.B}{T}$$

Sendo *Rh* o coeficiente Hall, constante para uma temperatura fixa em um determinado material, *I* a corrente que cruza o material, *B* a densidade do fluxo magnético, *T* a temperatura de operação e *v* a tensão resultante e proporcional à corrente *I*.

No caso do sensor de corrente usado [1], a saída tem a resolução de 0 a 5V, proporcional à corrente que passa por ele. Sendo a corrente nula, ou não havendo carga ligada ao sensor, a saída do mesmo apresenta a tensão de 2,5V. Na figura 2.1 podemos ver a função de cada pino do sensor. Uma aplicação típica, para o cálculo da corrente que passa por ele, é apresentada na Figura 2.2.

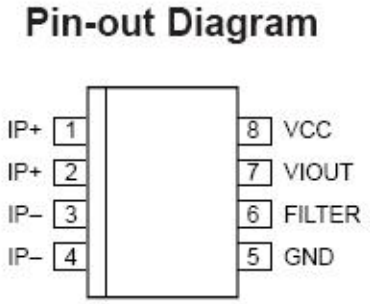

Figura 2.1 – Pinagem do Sensor Hall.
Fonte: Sensor Datasheet.

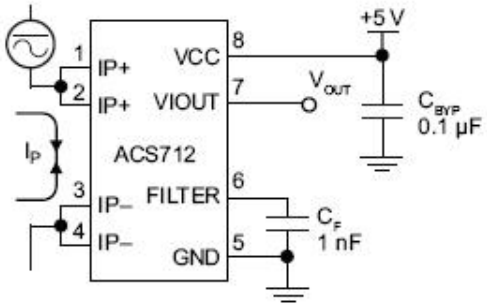

Figura 2.2 – Aplicação típica do Sensor Hall
Fonte: Sensor Datasheet.

Com isso, temos agora um sinal de tensão proporcional à variação da corrente. No entanto, a variação de tensão na entrada do dsPIC deve estar dentro do intervalo de 0 até V_{CC}, ou seja, o hardware não tem resolução para tensões negativas, o que se configura um problema para os ciclos negativos do sinal de tensão resultante. Para resolver esse problema ajustaremos a tensão de *offset* na saída do sensor de corrente sem carga como o valor da metade do V_{CC} (1,65 V). Primeiramente foi usado um *trimpot* multivolta, porém a instabilidade mecânica do mesmo fez com que se optasse por uma associação de resistores com precisão de 1% entre a saída do sensor e a entrada do dsPic. Assim, as disparidades nos valores poderiam ser somente corrigidas pelo software. Na Figura 2.3 podemos ver como foi feita a ligação do *trimpot* multivolta, onde posteriormente foi colocado um divisor resistivo de alta precisão.

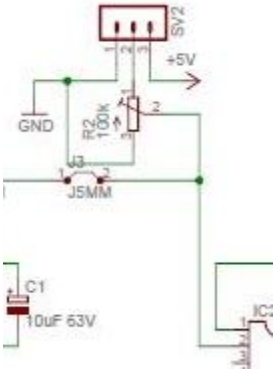

Figura 2.3 – *Trimpot* usado para abaixar a tensão de offset.

Após o uso desse artifício para o controle do offset, bastou-se subtrair 512 – metade da escala de 10 bits – para que se pudesse calcular somente os valores do sinal sem o nível DC do sensor de corrente. E assim, garantimos que qualquer sinal aplicado ao dsPIC tenha valor máximo de 3.3V e mínimo de 0.

2.3 – Cálculo do valor RMS e RMS-hora

Quando alimentamos uma carga ou circuito com uma tensão variável, em forma de senóide, por exemplo, a energia transmitida não é constante como em uma fonte de tensão contínua. Como forma de facilitar a medida dessa tensão variável, usa-se o valor RMS (*Root Mean Square*) para se obter um “valor eficaz” que equivale ao valor de uma fonte contínua que forneceria ao circuito a mesma potência elétrica.

É possível calcular o valor RMS [2] de um sinal de tensão contínuo com a seguinte equação

$$V_{RMS} = \sqrt{\frac{1}{T}\int_{to}^{to+T} v(t)^2 dt}$$

Sendo *v*(*t*) uma tensão senoidal, podemos simplificar a definição de RMS da seguinte maneira

$$V_{RMS} = \frac{V_{máx}}{\sqrt{2}}$$

Porém, apesar de o sinal de entrada do circuito ser contínuo, o módulo ADC (*Analog to Digital Converter*) do dsPIC transformará esse sinal analógico em digital, amostrando-o diversas vezes. Assim, na prática os dados que teremos para processar e calcular o RMS serão valores discretos referentes ao sinal contínuo da entrada.

Para o cálculo da frequência de amostragem vamos considerar que o sinal que será amostrado seja limitado em 5 kHz.

Sabemos que o período do sinal de 60 Hz é

$$T = \frac{1}{60} = 16{,}6\, ms$$

e assim meio período é 8,3 ms.

Para se ter 200 amostras por período serão obtidas amostras a cada 85 us, número inteiro que pode ser programado no timer do microcontrolador, o que nos dá uma frequência de amostragem de

$$f_{am} = \frac{1}{85\text{x}10^{-6}} \cong 11{,}764\, kHz$$

suficiente para amostrar um sinal de 5 Khz, segundo o critério de Nyquist.

De posse da frequência de amostragem, é preciso um algoritmo que implemente a equação do cálculo do valor RMS de uma maneira que o dsPIC consiga processar corretamente.

Deve-se então estipular um número de iterações que seja múltiplo de 200 para que seja obtido um número inteiro de períodos amostrados.

Usando o método de integração numérica por trapézios, a equação de V_{RMS} foi calculada como descrito a seguir.

$$(1)\ SUM = \frac{(X_n)^2 + (X_{n+1})^2}{2} + SUM$$

$$(2)\ X_n = X_{n+1}$$

$$(3)\ X_{n+1} = nova\ amostra$$

Onde X_n e X_{n+1} são, respectivamente, a amostra anterior e a atual e *SUM* é o somatório das médias. Esse bloco do algoritmo se repete até que a última amostra seja obtida. Ao final de cada ciclo de amostragem o resultado do X_{RMS} será

$$X_{RMS} = \sqrt{\frac{SUM}{N_{iterações}}}$$

Todo esse processo é executado ininterruptamente até que o usuário interfira. Sendo assim, para maior estabilidade do resultado final do cálculo e para amenizar algum tipo de erro ou interferência causada ou pelo sinal de entrada ou por ruídos, o valor que será mostrado no *display* é uma média móvel dos valores calculados pelo algoritmo acima.

A média móvel, usada com o intuito de ter um valor mais estável no tempo, amenizando alguma descontinuidade no sinal ou valor afetado por ruído, foi implementada da maneira descrita abaixo.

A partir de um vetor com 50 posições, inicialmente nulo, preenchemos o vetor de maneira que a última posição sempre receba o último valor RMS calculado.

$$(1)\ vetor = [x_0, x_1, \dots, x_{49}]$$

$$(2)\ x_{49} \leftarrow \text{último valor RMS calculado}$$

Na próxima iteração, o valor da posição x_{j+1} passa para a posição x_j, o valor de x_0 é descartado e x_{49} recebe um novo valor, que é sempre o último cálculo do RMS. Ao final de cada interação calculamos a média móvel como sendo

$$(3)\quad média\ móvel = \frac{[x_0, x_1, \ldots, x_{49}]}{50}$$

Apesar de necessitar de um tempo de processamento em torno de 3 segundos para o valor final convergir, esse artifício nos garante uma menor variação do valor mostrado no *display*.

Tendo em mãos toda a estrutura de cálculo da corrente RMS, o cálculo do RMS-hora é relativamente simples. Usando a função da interrupção externa do dsPIC, dedicou-se um PIC da família 12F675 [3] para gerar uma onda quadrada de 1Hz e servir como base de tempo da interrupção.

Com isso, a cada 1s, uma interrupção é acionada e dentro dela o valor atual do RMS instantâneo é dividido por 3600. O resultado dessa divisão é acumulado em uma variável e mostrado no *display*. Como o resultado é armazenado em uma variável de tipo *float*, consideramos que o circuito não ficará ligado tempo suficiente para que haja *overflow* do valor calculado.

2.4 – Cálculo da THD

Como vimos no início do capítulo 2, o sinal de entrada é composto por uma tensão senoidal com frequência de 60 Hz. Porém, para analisar mais profundamente qual a forma de onda da corrente, devemos lembrar a Lei de Ohm [4]

$$V(j\omega) = Z(j\omega).I(j\omega)$$

Onde $Z(j\omega)$ é a impedância da carga ligada ao circuito, $V(j\omega)$ é a tensão e $I(j\omega)$ é a corrente, todos no domínio da frequência. É fácil perceber então que a corrente é dada por

$$I(j\omega) = \frac{V(j\omega)}{Z(j\omega)}$$

Concluímos, assim, que a forma de onda da corrente está diretamente ligada à impedância, ou seja, ao tipo de carga que o circuito possui. No caso de uma carga linear, a forma de onda da corrente será uma senoide de mesma frequência, podendo ter fase e amplitude modificadas.

Infelizmente na vida real e prática dificilmente encontraremos uma carga linear. De forma geral, a impedância depende da frequência e do valor instantâneo da corrente. Então, toda vez que ligamos um eletrodoméstico introduzimos harmônicos na forma de onda da corrente pela natureza não linear da carga, e defasamos o sinal devido à sua reatância.

Esse efeito muitas vezes é indesejado em certas aplicações, por isso o cálculo da THD é um método eficaz para analisar a influência dos harmônicos em um sinal elétrico.

A THD [2] nada mais é do que a medida da distorção harmônica, e é definida pela seguinte equação

$$THD(\%) = \sqrt{\left(\frac{P_{total} - P_1}{P_1}\right)} * 100$$

Onde P_{total} é a potência de todas as componentes harmônicas do sinal incluindo a componente fundamental P_1. Podemos perceber que a THD é a raiz quadrada da razão entre o somatório das potências das frequências harmônicas e a potência da frequência fundamental. A THD é normalmente apresentada como porcentagem.

Como no caso do RMS, precisamos de um algoritmo para o cálculo da THD que o microcontrolador consiga executar usando seus recursos de processamento e memória disponíveis. Nesse ponto, a escolha do método de cálculo foi complexa, pois a memória do dsPIC não suporta vetores com grande número de posições. Isso inviabilizou o uso do cálculo através do método por DFT (*Discrete Fourier Transform*) com resolução aceitável para o projeto.

Como um meio de solucionar o problema da limitação da memória, optou-se por calcular a THD pela série de Fourier. No dsPIC dedicado para o cálculo da THD, foi implementado um algoritmo para calcular os coeficientes da série de Fourier do sinal pelo método da integral. O uso desse método só foi possível graças à frequência da rede

elétrica ser extremamente estável e apresentar uma variação mínima, quando observada em grandes janelas de tempo. Isso implica que, para aplicações instantâneas ou com variações lentas de tempo, podemos considerar a frequência como constante.

Verifica-se que qualquer sinal periódico pode ser representado como um somatório de senos e cossenos, como indica a equação da Série de Fourier

$$f(t) = X_0 + \sum_{n=1}^{N} A_n \sin(n\omega t) + B_n \cos(n\omega t)$$

onde X_0 é a componente contínua do sinal.

Através da Série de Fourier, podemos calcular a amplitude da componente fundamental, em $n = 1$, e, consequentemente, a potência P_1.

Escolhendo esse método para o desenvolvimento do algoritmo, há apenas um detalhe a observar no que se diz respeito à memória. Deve-se inserir os valores discretos pré-armazenados do seno e cosseno da fundamental, nesse caso 60Hz. Esses valores foram obtidos com a ajuda do *software* para aplicações matemáticas Matlab.

Teremos então 3 vetores: o seno, o cosseno e o sinal amostrado. Para uma resolução satisfatória, dentro das limitações da memória do dsPIC, foram escolhidos vetores com N = 100 posições. Sendo esses

- $SIN[n]$ → Vetor com as posições do seno com frequência da fundamental
- $COS[n]$ → Vetor com as posições do cosseno com frequência da fundamental
- $X[n]$ → Vetor onde serão armazenadas as amostras do sinal analógico

Sendo os vetores *SIN[n]* e *COS[n]* valores constantes e pré-carregados na memória de programa, o único vetor a ser preenchido em tempo real pelo dsPIC é o *X[N]*, que será alimentado com os valores de amostragem do sinal de entrada.

Esse processo é executado de maneira simples, apenas amostrando e armazenando o valor digital na próxima posição do vetor *X[N]*, com o valor de *n* variando de 0 a 99.

Porém, para garantir que o dsPIC comece a amostrar o sinal quando o mesmo estiver com um valor próximo de zero, indicando que está no início de um novo ciclo, foi pensado um detector que analisa aproximadamente o momento em que o sinal sai da parte negativa e entra em sua parte positiva. A partir desse determinado momento, o dsPIC começa a alimentar o vetor *X[N]*. No caso da THD, para um vetor de 100 posições, obteve-se uma amostra a cada 160 us já que um período de 60 Hz dura aproximadamente 16,6 ms.

O próximo passo é definir, por meio desses vetores, os valores dos coeficientes *A* e *B*. Iniciando os coeficientes com o valor 0, usamos a seguinte iteração, com *n* variando de 0 a 99:

$$A_{n+1} = \frac{(\mathrm{SIN}[n] * X[n] + \mathrm{SIN}[n+1] * X[n+1])}{2} + A_n$$

$$B_{n+1} = \frac{(\mathrm{COS}[n] * X[n] + \mathrm{COS}[n+1] * X[n+1])}{2} + B_n$$

Após todas as iterações, temos um valor real para os coeficientes. Agora usamos as seguintes identidades para calcular os valores a serem usados para o cálculo da THD.

$$A = \frac{2A}{N-1} \; ; \; B = \frac{2B}{N-1}$$

Precisamos agora calcular a potência média do sinal amostrado, que agora é conhecido como sendo os valores de *X[n]*. Faremos agora mais 100 iterações, sempre variando *n* de 0 a 99, que nos dará a integral da potência do sinal.

$$P_{total} = \frac{(X[n]^2 + X[n+1]^2)}{2} + P_{total}$$

Terminado esse cálculo, usaremos a seguinte equação para obter o valor da potência média.

$$P_{med} = \frac{P_{total}}{N-1}$$

Temos agora todos os resultados necessários para que a THD possa ser determinada. Assim, aplicamos a equação

$$THD(\%) = \sqrt{\left(\frac{2(P_{med} - P_{noise})}{A^2 + B^2}\right) - 1} * 100$$

Onde P_{noise} é a potência de um possível ruído de fundo inserido no circuito, seja por sua estrutura física ou alguma causa externa, e pode alterar a escala do resultado final. Esta pequena alteração na fórmula do cálculo da THD melhora significativamente a precisão da medida. Para o cálculo de P_{noise} o circuito foi ligado com o sinal fundamental, e usando o valor que foi mostrado no *display* LCD, calculou-se a potência do ruído de fundo tal que, se subtraído, zerasse o valor do *display*.

Assim como no cálculo da corrente RMS, antes do resultado da THD ser transmitido para o outro dsPIC, a fim de ser mostrado no *display*, uma média móvel é aplicada para estabilizar e proteger o valor final de algum tipo de erro causado por ruído ou erro na obtenção de dados.

Capítulo 3

Microcontrolador

3.1 – Características Gerais do dsPIC

O módulo de processamento é usado para o controle de toda a parte inteligente do projeto. É onde efetivamente os dados são usados para o cálculo das métricas, isso inclui a amostragem do sinal analógico, a execução dos algoritmos, a transmissão do resultado e por fim o controle do *display*.

Esse módulo é composto por dois dsPICs da Microchip, modelo dsPIC33FJ16GS502 [5]. Sendo um responsável pelo cálculo da corrente RMS e controle do *display*, e o outro pelo cálculo da THD. A comunicação entre ambos é feita através do módulo UART nativo.

Abaixo, na figura 3.1 e na tabela 3.1, podemos ver, respectivamente, a pinagem com a função de cada entrada ou saída do dsPIC e suas principais características.

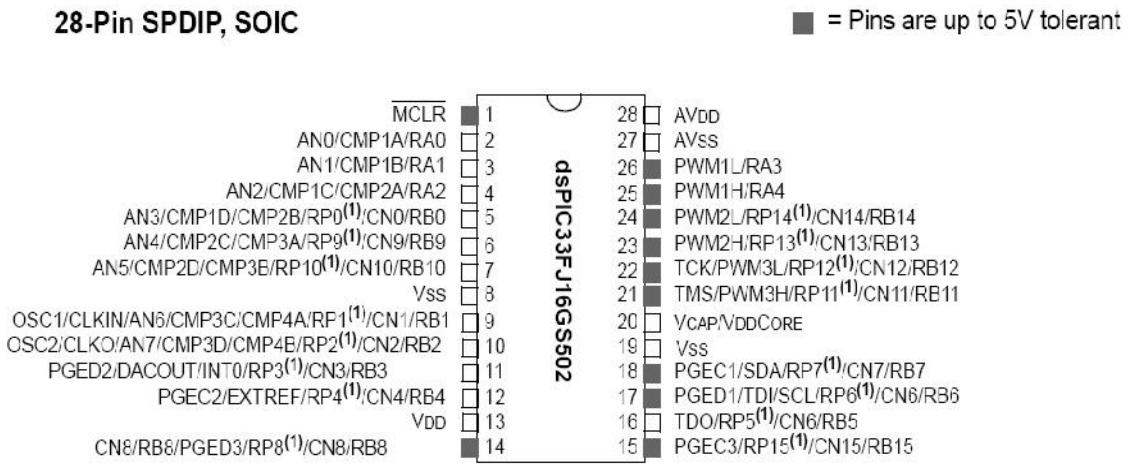

Figura 3.1 – Pinagem do dsPIC.
Fonte: dsPIC33FJ16GS502 Datasheet.

Tabela 3.1 – Principais características do microcontrolador
Fonte: dsPIC33FJ16GS502 Datasheet.

Device	Pins	Program Flash Memory (Kbytes)	RAM (Bytes)	Remappable Peripherals									DAC Output	I^2C™	ADC			I/O Pins	Packages
				Remappable Pins	16-bit Timer	Input Capture	Output Compare	UART	SPI	PWM(2)	Analog Comparator	External Interrupts(3)			SARs	Sample and Hold (S&H) Circuit	Analog-to-Digital Inputs		
dsPIC33FJ16GS502	28	16	2K	16	3	2	2	1	1	4x2(1)	4	3	1	1	2	6	8	21	SPDIP SOIC QFN-S

3.2 – Módulo ADC

Uma das partes mais importantes do módulo de processamento é o conversor analógico-digital *(*ADC – *Analog to Digital Converter).* É nele que o sinal de entrada é amostrado para que os cálculos sejam possíveis.

O ADC é composto de um *sample-hold* dedicado para as entradas analógicas pares, um *sample-hold* compartilhado pelas entradas ímpares, dois módulos SAR (*Successive Approximation Registers*), sendo um para entradas pares e outro para as entradas ímpares, e um *buffer* com resolução de 10 bits. Com esta configuração, as entradas pares e ímpares são convertidas em paralelo alcançando uma taxa de conversão de 4MHz.

A seguir podemos ver o esquemático do ADC.

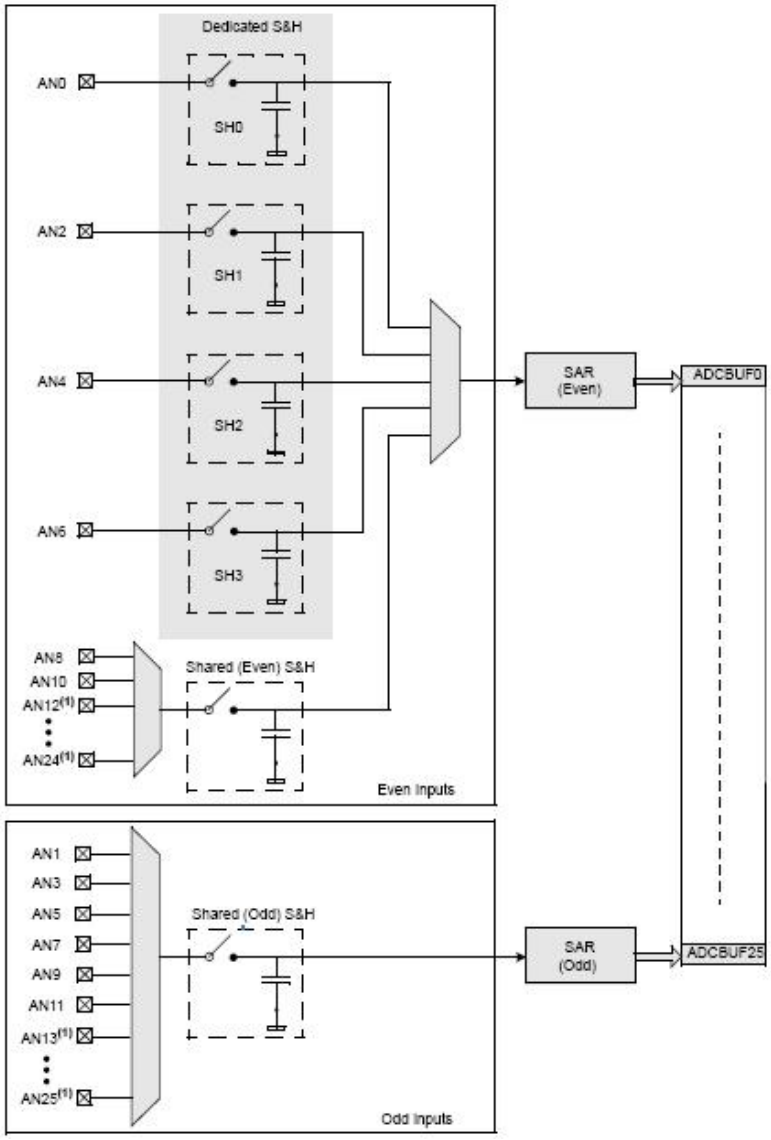

Figura 3.2 – Módulo ADC.
Fonte: dsPIC33FJ16GS502 Datasheet.

3.3 – Módulo UART

Como vimos na sessão 2.4, para calcular a THD precisamos de alguns vetores com diversas posições, demandando mais memória do que o dsPIC possui. Para resolver este problema, usou-se dois dsPICs, sendo um deles dedicado somente para a THD, o que configura outro problema, já que apenas um deles irá controlar o *display*. Assim será usado o módulo UART - *Universal Asynchronousreceiver Transmitter* para transmitir o resultado de um dsPIC para o controlador do *display*.

Nesse projeto em particular usou-se a seguinte configuração:

- Sem paridade, 8-bit por transmissão;
- Modo de baixa velocidade;
- *Baud rate* – 9600
- *Auto-baud* desabilitado

Uma particularidade desse dsPIC é que alguns pinos devem ser logicamente mapeados. Sendo esse o caso do módulo de transmissão, não havendo pinos dedicados à TX e RX, deve-se configurá-los nos respectivos registradores de seleção de pinos mapeáveis. Essa característica é interessante em termos de organização do *layout* do circuito.

Fisicamente, é apenas necessário conectar o pino TX de um dsPIC com o pino RX do outro. É importante frisar que as configurações UART de ambos os microcontroladores devem ser exatamente iguais, caso contrário a comunicação entre eles não será bem sucedida.

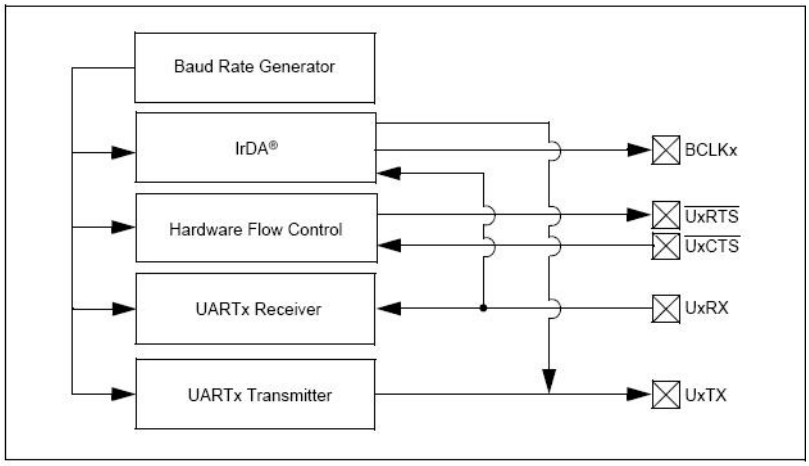

Figura 3.3 – Diagrama de blocos simplificado - UART.
Fonte: dsPIC33FJ16GS502 Datasheet.

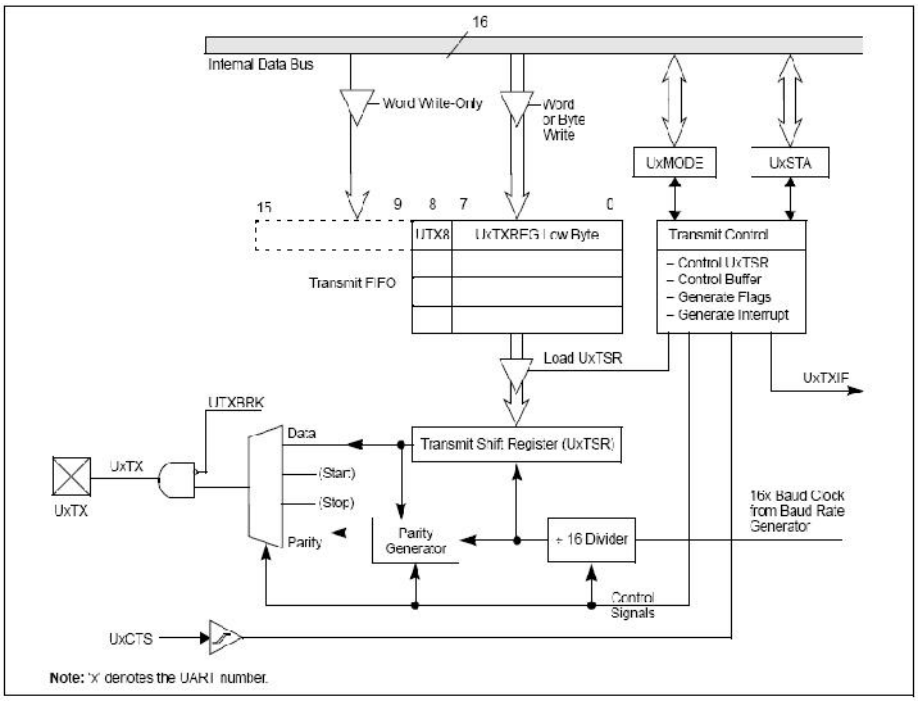

Figura 3.4 – Diagrama de blocos do transmissor - TX.
Fonte: dsPIC33FJ16GS502 Datasheet.

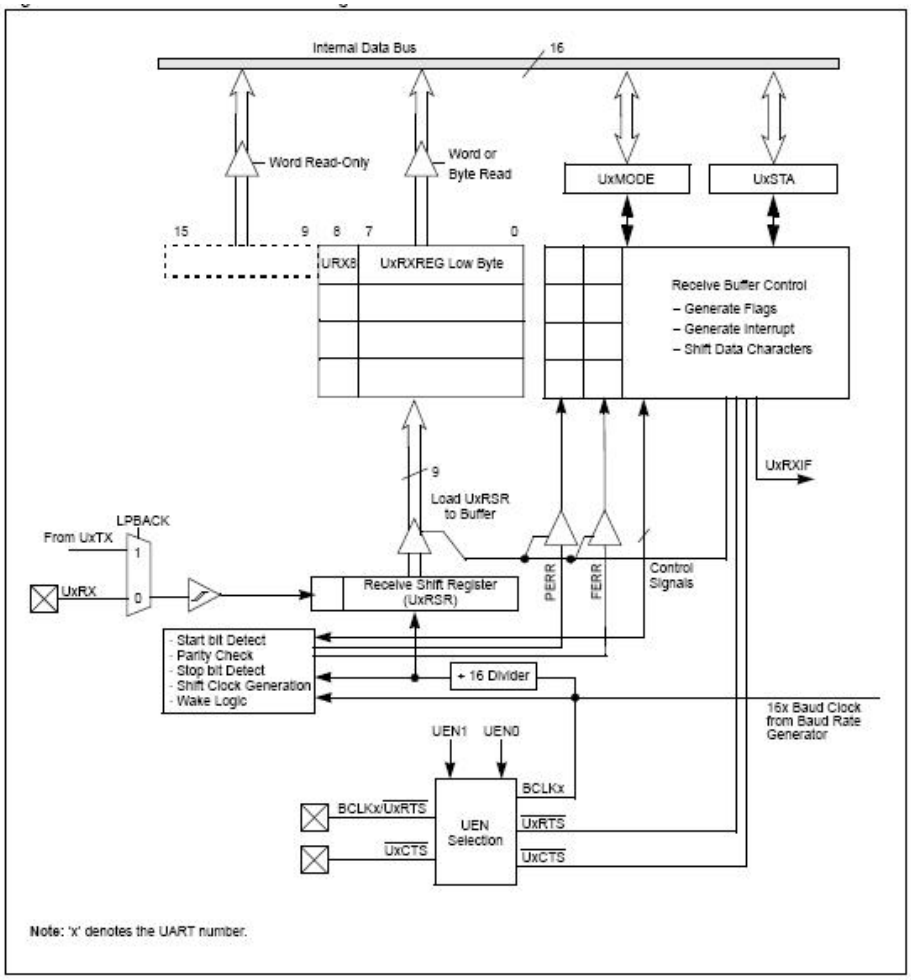

Figura 3.5 – Diagrama de blocos do receptor - RX.
Fonte: dsPIC33FJ16GS502 Datasheet.

3.4 – Programa Principal

Podemos dividir o projeto em *hardware* e *software*, sendo o *hardware* a estrutura física (placa de circuito impresso, microcontroladores, oscilador, resistores, etc) e o *software* toda a parte lógica da programação necessária para controlar os dsPICs.

Todo o projeto foi desenvolvido em linguagem C usando o compilador *mikroC PRO Advanced C Compiler for dsPIC MCU*, cuja licença para a utilização foi doada pela empresa ao Departamento de Eletrônica e de Computação (DEL).

O programa principal contempla a interface com o usuário. Para a navegação, um botão deve ser acionado, alterando as opções em uma estrutura *switch-case*. Sendo uma estrutura sequencial, primeiro será calculada a corrente RMS, passando para a corrente RMS-hora e por fim a medida da THD.

3.5 – Controle do *Display*

Após todos os cálculos efetuados pelo módulo de processamento, é necessário mostrar os valores e resultados obtidos. Pra isso, foi usado um *display LCD – Liquid Crystal Display* com 16 colunas e 2 linhas.

Para o controle do *display*, usou-se uma biblioteca nativa do compilador que necessita de três parâmetros: em qual linha será escrito o texto, em qual coluna se iniciará o texto e o texto propriamente dito. Deve-se também configurar os pinos de saída do dsPIC que serão responsáveis por controlar o *display* e enviar os dados para o mesmo.

Podemos ver na Figura 3.6 a pinagem do *display*.

Figura 3.6 – Pinagem Display 16x2.

Capítulo 4

Resultados

4.1 – *Hardware*

Após a fase de desenvolvimento do projeto, foi projetada e fabricada a placa de circuito impresso para que os testes finais pudessem ser executados. Assim, após a criação do layout e roteamento a placa foi produzida seguindo os layouts mostrados abaixo.

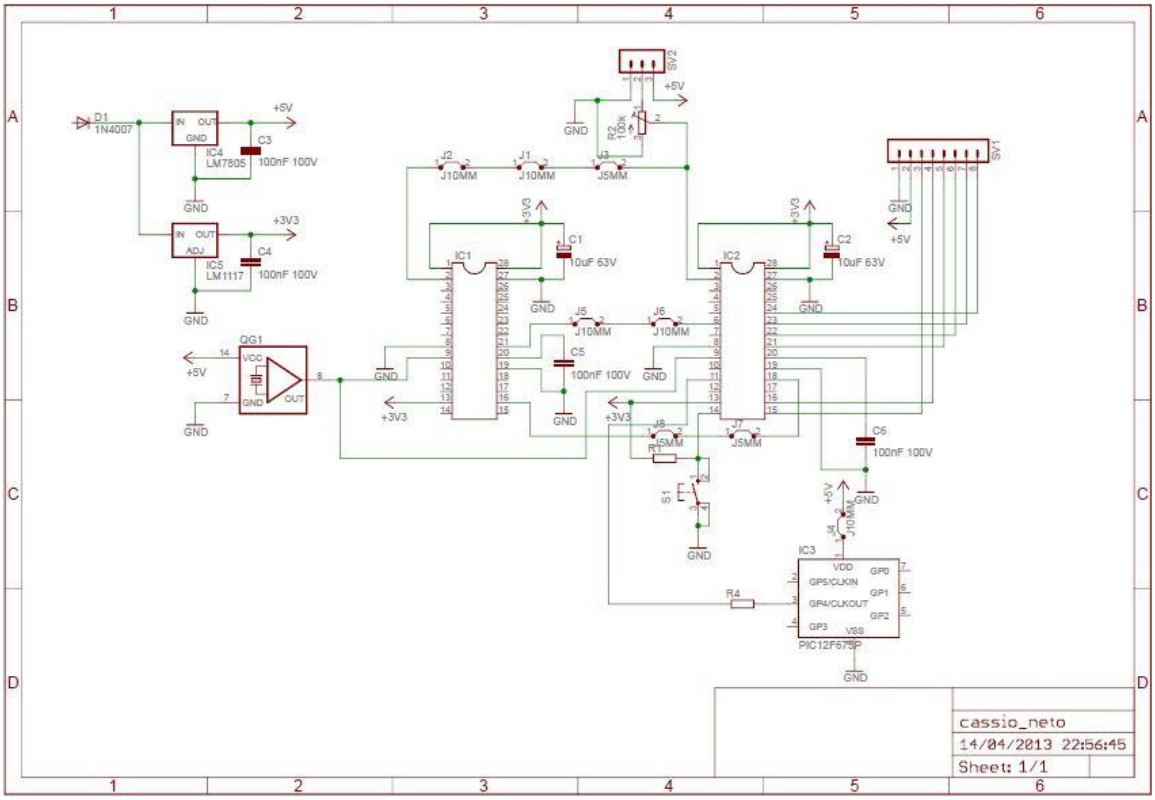

Figura 4.1 – Esquemático do Circuito.

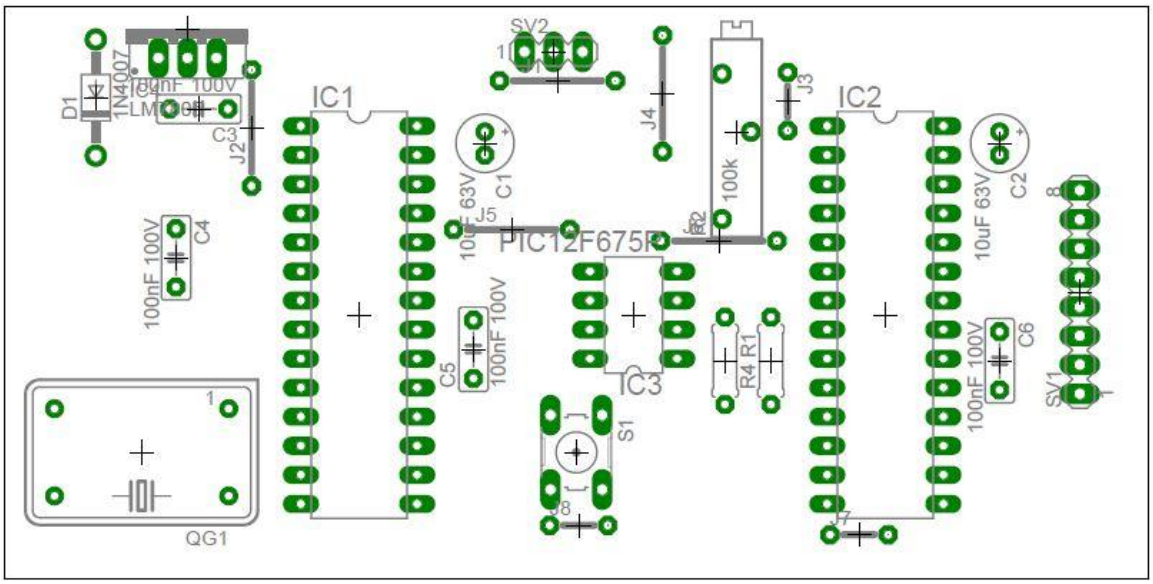

Figura 4.2 – Esquemático da Montagem.

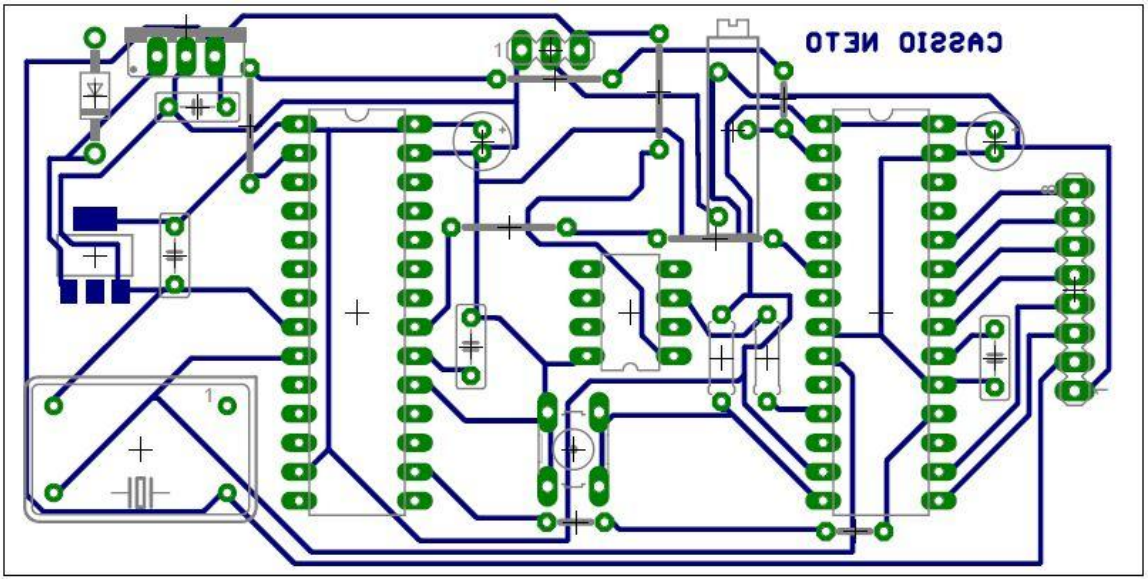

Figura 4.3 – Esquemático de Produção.

4.2 – Resultados RMS

Para concluir o projeto e validar o resultado das métricas calculadas pelo microcontrolador, foi necessária a montagem de um cenário de testes. Foi usado uma associação de lâmpadas de filamento ligadas em 220V em série com um Variac (dispositivo utilizado para variar a tensão AC), que por sua vez estava ligado em série com um multímetro de bancada. Todo esse circuito ligado pelo sensor de corrente do projeto, como mostra o diagrama de blocos abaixo.

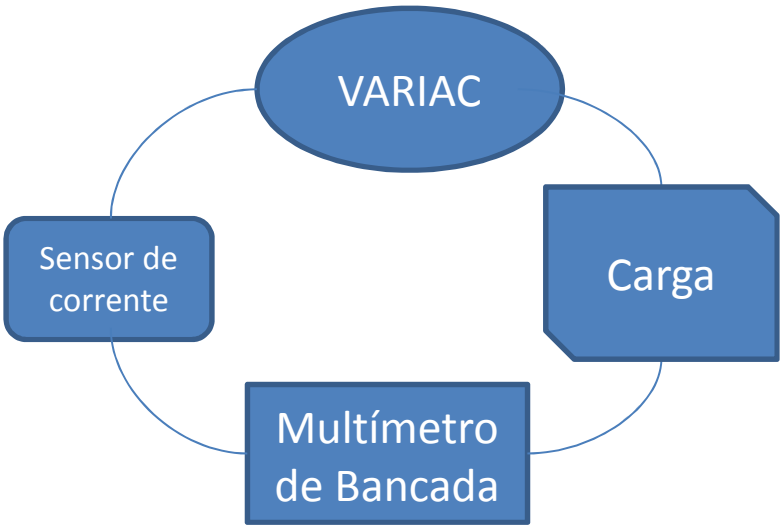

Figura 4.4 – Diagrama de blocos do cenário de teste do RMS.

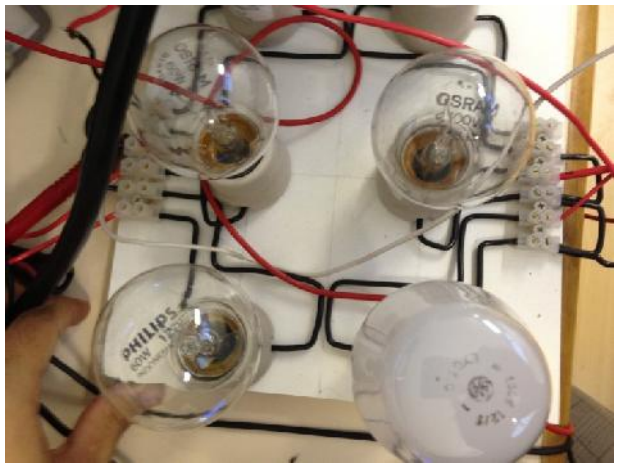

Figura 4.5 – Associação usada como carga para os testes.

Após montado o cenário de testes, pudemos começar a validar os valores calculados pelo dsPIC. Variando a tensão no Variac e observando a mudança na corrente no multímetro de bancada, foi possível comparar o resultado da leitura no multímetro e o mostrado no *display* do projeto.

Abaixo vemos alguns resultados coletados durante os testes. As figuras mostram o valor da corrente RMS calculada pelo multímetro de bancada e pelo microcontrolador mostrado no *display* LCD.

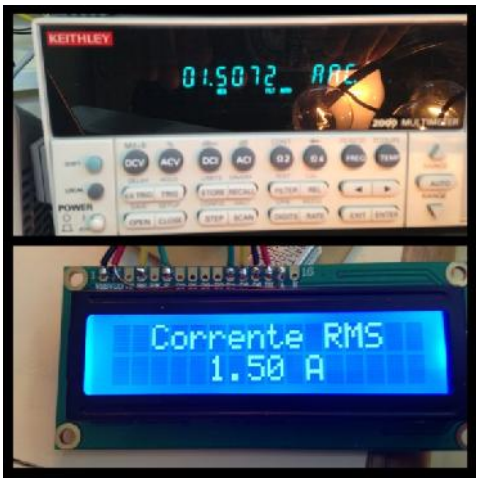

Figura 4.6 – Teste RMS 1,5 A.

Figura 4.7 – Teste RMS 2 A.

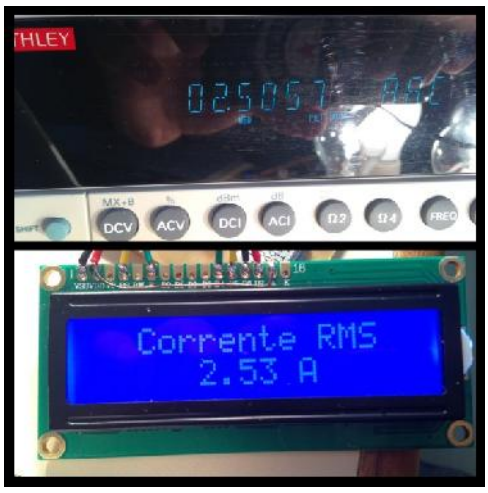

Figura 4.8 – Teste RMS 2,5 A.

Figura 4.9 – Teste RMS 1,58 A.

Figura 4.10 – Teste RMS 500 mA.

Figura 4.11 – Teste RMS 900 mA.

Nos testes, variou-se a corrente de 0 até 2,5 A a fim de não ultrapassar o máximo valor suportado pela carga. Observou-se que um problema de precisão nos resultados. Isso foi causado por uma limitação do *hardware*, que fez com que a combinação da

sensibilidade do sensor de corrente com o conversor ADC, de apenas 10 bits, trouxesse uma incerteza na segunda casa decimal.

Nos testes, medições abaixo de 1 A não apresentaram valores confiáveis, devido à baixa amplitude do sinal que não era suficiente para o módulo ADC executar uma leitura correta. Os resultados mostrados pelo *display* com correntes entre 1 e 2 A corresponderam fielmente aos valores mostrados no multímetro até a segunda casa. De 2 até 2,5 A, a segunda casa decimal passou a não ser mais confiável.

Em uma conta rápida, podemos verificar o porquê da segunda casa decimal apresentar tal incerteza.

Sendo a alimentação do dsPIC de 3.3V, e o módulo ADC de 10 bits, temos que

$$\frac{3{,}3}{1024} = 3{,}22\ x\ 10^{-3}$$

Esse valor nos mostra que a cada 3,22 mV variando na entrada do dsPIC, 1 bit varia no valor convertido. Tendo isso em mãos e sabendo que a sensibilidade do sensor de corrente é de 66 mV/A, é fácil perceber que uma variação de 0,01 A na entrada do sensor causará uma variação de 0,66 mV na saída do sensor, que é, na verdade, a entrada do dsPIC. Logo, esse valor é inferior ao mínimo reconhecido pelo conversor AD (3,22 mV > 0,66 mV). Isso nos mostra que apenas será possível garantir, com esse *hardware* escolhido para o desenvolvimento deste projeto aqui descrito, a primeira casa decimal tendo como erro a segunda casa.

Para o teste e validação do RMS- hora, o circuito de testes foi deixado ligado por 40 minutos. Foi montada uma tabela com valor teórico calculado pela mesma fórmula implementada no microcontrolador, e depois observado os valores de 10 em 10 minutos para serem comparados com a tabela. Os valores observados encontram-se na Tabela 4.1.

Tabela 4.1 – Valores coletados no teste de RMS/hora

Valor Teórico (A)	Valor Display (A)	Diferença (A)	Tempo (Min)
0,402325	0,46	-0,05768	20
0,603991	0,69	-0,08601	30
0,805658	0,92	-0,11434	40

4.3 – Resutados THD

Para validar o valor da THD, outro cenário de testes foi preciso. Isso se deu pela baixa amplitude do sinal de saída do sensor nas correntes testadas, que interferiu no funcionamento do algoritmo.

Assim, para a validação do algoritmo e dos valores por ele calculados foi necessário um gerador de sinais com dois canais, para que se pudesse compor sinais harmônicos em cima da fundamental e comparar com o valor calculado no Matlab.

O sinal fundamental foi uma onda senoidal com frequência de 60 Hz, *offset* de 2.5 V e amplitude de 5 V. Um outro sinal com frequência de 120 Hz e *offset* de 2.5 V foi adicionado ao sinal fundamental, e variando a sua amplitude foi-se validando os resultados calculados pelo microcontrolador.

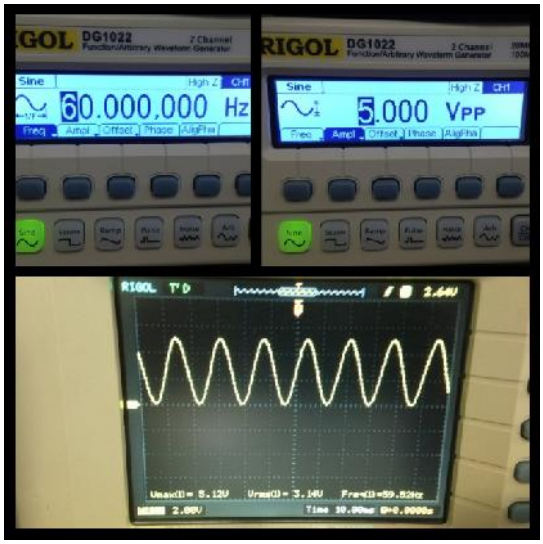

Figura 4.12 – Sinal Fundamental

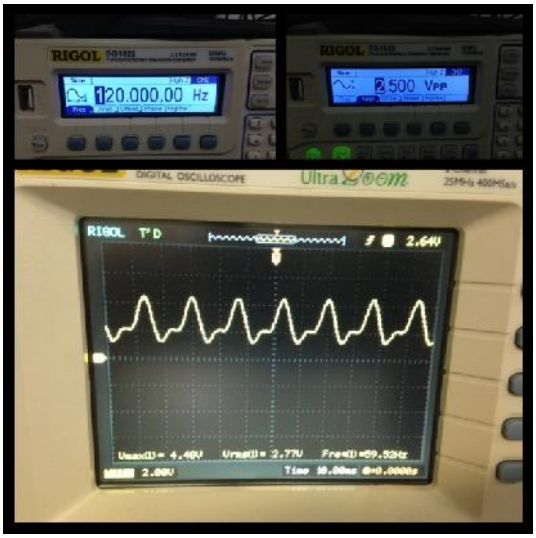

Figura 4.13 – Sinal composto pelo harmônico de 120 Hz.

Figura 4.14 – Teste THD Valor teórico 50%.

Figura 4.15 – Teste THD Valor teórico 60%.

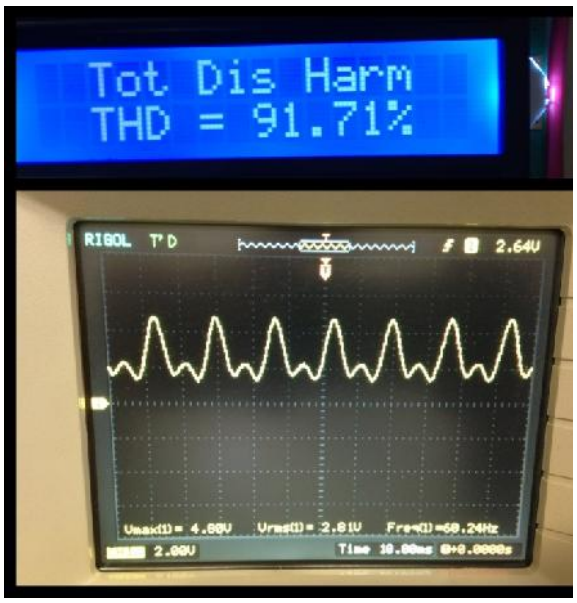

Figura 4.16 – Teste THD Valor teórico 90%.

Figura 4.17 – Teste THD Valor teórico 100%.

Tabela 4.2 – Resultados consolidados dos Testes.

Corrente RMS				THD			
Valor Multímetro (A)	Valor Medido (A)	Diferença (A)	Erro (%)	Valor Teórico (%)	Valor Medido (%)	Diferença (%)	Erro (%)
0,52	0,50	0,02	4%	50	50,01	-0,01	0%
0,92	0,90	0,02	2%	60	59,28	0,72	1%
1,50	1,50	0	0%	90	91,71	-1,71	2%
1,58	1,58	0	0%	100	101,83	-1,83	2%
2,00	2,02	-0,02	1%				
2,50	2,53	-0,03	1%				

Capítulo 5

Conclusão

No mundo da eletrônica digital há sempre limitações, tanto de *hardware* quanto de *software*. Sendo que estes devem sempre andar em conjunto, buscando a otimização de recursos para um resultado final desejado ou aceitável. Quando se trabalha com métricas, seja ela de qual natureza for, o ideal é a máxima precisão com muitas casas decimais e uma incerteza tal que possa ser desconsiderada.

Infelizmente, tudo tem seu preço. Um *hardware* de menor custo leva a uma precisão menor e pode não ser útil dependendo da aplicação. Por outro lado um *hardware* de maior precisão pode ser um investimento tão alto que não valha a pena seu uso. Tudo isso deve ser considerado quando se desenvolve um projeto que visa precisão. Microcontroladores são bem versáteis para usos genéricos, mas podem ser um problema quando se quer uma aplicação mais focada.

Neste projeto foi usado um dsPIC considerado de alta capacidade de processamento, mas mesmo assim não foi possível alcançar uma precisão de mais de uma casa decimal na leitura. Vimos que as escolhas de software foram bem sucedidas e que funcionaram como esperado.

As limitações encontradas e observadas no decorrer do desenvolvimento deste projeto foram exclusivamente limitações do *hardware* escolhido. A fim de deixar em aberto o estudo de um sistema autônomo para o cálculo da corrente RMS e da THD de um sinal, tendo como norte todas as etapas descritas neste documento, pode-se concluir que há algumas ações a serem tomadas a fim de melhorar o desempenho dos algoritmos usados. Essas ações serão listadas na sessão Trabalhos Futuros que visa sugerir ao leitor algumas novas alternativas para resolução de problemas encontrados que influenciaram significativamente a precisão dos resultados finais.

Apesar dos resultados observados nos testes, o estudo inicial de um sistema capaz de calcular corrente RMS e THD de um dado sinal foi bem sucedido. Muitos problemas puderam ser mapeados e mais conhecimento adquirido sobre esse assunto. Podendo assim servir de norte para futuras aplicações que visem ao cálculo das mesmas métricas.

Trabalhos Futuros

A fim de nortear trabalhos futuros que visem reproduzir todo o projeto descrito neste documento, deixo as seguintes sugestões para ajudar a aprimorar os resultados finais calculados:

- Usar um filtro anti-aliasing para garantir a banda de frequência do sinal amostrado.
- Módulo amplificador que ajuste automaticamente o valor do ganho de acordo com a amplitude do sinal de entrada.
- Usar outro canal do conversor AD para obter o valor do *offset* na entrada e subtrair esse valor instantâneo dentro do software, garantindo que a variação do nível DC que não pertence ao sinal seja corretamente desconsiderada no resultado final.
- A escolha de um microcontrolador com um módulo ADC maior de 10 bits, de preferência o maior número de bits disponível no mercado, ajudará a detectar menores variações de tensões na entrada e assim aumentará a precisão da medida.
- Observar o valor da alimentação do microcontrolador ou referência do módulo ADC para que este seja igual ao valor máximo da escala do sensor de corrente escolhido. Assim, nenhum artifício será necessário para ajustar valores diferentes de escala.
- O uso de um sensor de corrente mais sensível à variações de corrente pode ser um fator que aumente a precisão das medidas calculadas.

Seguindo essas sugestões, julga-se que o resultado final será mais preciso do que os observados nos testes descritos no Capítulo 4.

Bibliografia

[1] ALLEGRO MICROSYSTEMS, INC, **0712.pdf**, Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1kVRMS Isolation and a Low-Resistance Current Conductor, 2009. Arquivo baixado. 637kb. Acrobat Reader. Disponível em: < http://www.allegromicro.com/en/Products/Part_Numbers/0712/0712.pdf >.

[2] FILHO, S. de M. **Medição de Energia Elétrica.** 7 ed. Rio de Janeiro: Guanabara Dois, 1997.

[3] MICROCHIP, **41190G.pdf**, 8-Pin, Flash-Based 8-Bit CMOS Microcontrollers**.** Arquivo baixado. 1.424Mb. Acrobat Reader. Disponível em: < http://ww1.microchip.com/downloads/en/DeviceDoc/41190G.pdf >.

[4] SEDRA, SMITH, Microelectronics Circuits. Makron Books, 2000.

[5] MICROCHIP, **70318F.pdf**, 16-bit Digital Signal Controllers (up to 16 KB Flash and up to 2 KB SRAM) with High-Speed PWM, ADC, and Comparators. Arquivo baixado. 5.126Mb. Acrobat Reader. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/70318F.pdf >.

[6] SOUZA, AMADEU, Programação em C para o dsPIC, Fundamentos. Ensino Profissional, 2008.

[7] SILVA, R. L. **Medidor de Energia Elétrica de Baixo Custo com Interface Serial Compatível com a NBR14522.** Trabalho de conclusão, CEFET, Florianópolis, 2007.

Apêndice A

Código Fonte do dsPIC I (RMS)

```
/*
Codigo do dsPIC1 - Calcula IRMS e mostra os valores para o Display Lcd 16x2
OFICIAL
FIN = 40Mhz (Ressonador)
FCY = FOSC/2 = 20Mhz
*/

#define TAMANHO 50
#define N_Am    100
#define BAUDRATE 9600
#define BRGVAL ((20000000/BAUDRATE)/16)-1
#define RMSHORA 1/3600

// Definicao das variaveis
int g, trigger, switch_var;
float rms_hora, rmshoratot;
char dadochr2[5];
char dado[10] = {0};

// Definicao das funcoes
void       init_ADC(void); // Funcao de configuracao do modulo ADC
void       init_config(void); // Funcao de configuracao das portas
float      IRMS_calc(void); // Funcao de calculo do IRMS
void       IRMS(void); // Funcao de execucao do IRMS

// Definicao das variaveis a serem usadas pela biblioteca do LCD
sbit LCD_RS at LATB15_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB11_bit;
sbit LCD_D5 at LATB12_bit;
sbit LCD_D6 at LATB13_bit;
sbit LCD_D7 at LATB14_bit;

sbit LCD_RS_Direction at TRISB15_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB11_bit;
sbit LCD_D5_Direction at TRISB12_bit;
```

```
sbit LCD_D6_Direction at TRISB13_bit;
sbit LCD_D7_Direction at TRISB14_bit;

/********** CONFIGURACAO DAS PORTAS**********/
void init_config(void){
  ADPCFGbits.PCFG0=0;      // AN0 como entrada analogica
  ADPCFGbits.PCFG1=0;      // AN1 como entrada analogica

  TRISB=0; // PORTB como saida
  TRISBbits.TRISB8 = 1; // RB8 como entrada -> Acionamento do botao
  TRISBbits.TRISB3 = 1; //RB3 como entrada -> Interrupção Externa

  Unlock_IOLOCK();  // Habilitar mudancas de PPS
  //PPS_Mapping(6, _INPUT, _T1CK);
  //PPS_Mapping(0, _INPUT, _INT1);
  PPS_Mapping(7, _INPUT, _U1RX); // RP7 como RX
  PPS_Mapping(9, _OUTPUT, _U1TX); // RP9 como TX
  Lock_IOLOCK();    // Desabilitar mudancas de PPS
}
/*************** CONFIGURACAO DO MODULO ADC ******************/
void init_ADC(void){
  ADCONbits.ADON=0; // Desabilita o processo de conversao
  ADCONbits.ORDER=0; // Ordem do ADC par-impar
  ADCONbits.ADCS=0b001; // Divisor Fvco/2 => No caso o clock de 20 Mhz
  ADCONbits.FORM=0; // Formato de saida inteiro
  ADCONbits.ASYNCSAMP=0; // SampleHold sempre amostrando ate receber o
trigger da conversao
  ADSTATbits.P0RDY=0;  // Limpa o bit de status do Par 0
  ADCPC0bits.TRGSRC0=0b00001; // Seleciona o trigger individual por software

  ADCONbits.ADON=1; // Inicializa o processo de conversao
}

/************ CONFIGURACAO DA INTERRUPCAO *****************/
void init_Interrupt(void){
     INTCON2bits.INT0EP = 0;
     IPC0bits.INT0IP = 0x07;
     IFS0bits.INT0IF = 0;
     IEC0bits.INT0IE = 1;
}
```

```
/************************ IRMS ****************************/
void IRMS(void){

  char dadochr[5];
  int i,j;
  float vetor[TAMANHO], media_rms;

  for(i=0;i<TAMANHO;i++){
  vetor[i]=0;
  }

  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Out(1,3, "Corrente RMS");

  while(1){
   media_rms = 0;

   for(j=0;j<TAMANHO-1;j++){
   vetor[j]=vetor[j+1];
   }
   vetor[TAMANHO-1] = IRMS_calc();
   for(j=0;j<TAMANHO;j++){
   media_rms += vetor[j];
   }
   media_rms = media_rms/TAMANHO;
   media_rms = media_rms - 0.09;

   sprintf(dadochr, "%.2f A", media_rms);
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,3, "Corrente RMS");
   Lcd_Out(2,6,dadochr);
   delay_ms(100);

   if(Button(&PORTB, RB8, 8, 0)){
     switch_var++;
   break; }
  }
}

/************************ IRMS_calc ****************************/
float IRMS_calc(void){

   int z;
   int i,j, x;
```

```
    float soma, somatorio = 0;
    float valorant, buffer, W, AUX, K, II, rms, rms_final;

    for(z=0;z<1071;z++){

      ADCPC0bits.SWTRG0=1;
      while(ADSTATbits.P0RDY=0)
      continue;
      x = ADCBUF0;
      W = x - 491;
      K = W*30;
      II = K/375;
      //II = II;

      AUX = II*II;
      soma = somatorio + AUX;
      somatorio = soma;
      delay_us(85);
     }

      rms = somatorio/1071;
      rms_final = sqrt(rms);

      return rms_final;
}
/********************** IRMS_Hora *********************************/
void IRMS_hora(void){

   char dadochr[5];

   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,5, "RMS/hora");

   while(1){

    rms_hora = IRMS_calc();

    sprintf(dadochr, "%.3f A", rmshoratot);
    Lcd_Out(2,6,dadochr);

    if(Button(&PORTB, RB8, 8, 0)){
    switch_var++;
    IEC0bits.INT0IE = 0;
    break;
```

```
    }
   }
}

/************** CONFIGURACAO UART ************************/
void init_uart(void){

   U1MODEbits.STSEL = 0; // 1-stop bit
   U1MODEbits.PDSEL = 0; // No Parity, 8-data bits
   U1MODEbits.ABAUD = 0; // Auto-Baud Disabled
   U1MODEbits.BRGH = 0; // Low Speed mode
   U1BRG = BRGVAL; // BAUD Rate Setting for 9600
   U1MODEbits.UARTEN = 1; // Enable UART
   U1STAbits.UTXEN = 1; // Enable UART Tx

}

/******************** PROGRAMA PRINCIPAL ***************************/
void Int0Int() iv IVT_ADDR_INT0INTERRUPT ics ICS_OFF {

  rmshoratot = rmshoratot + (rms_hora * RMSHORA);
  IFS0bits.INT0IF = 0;
}
void main(){

   init_config(); // Setup
   init_uart();

   Delay_ms(100);

   Lcd_Init();          // Inicializa o display
   Lcd_Cmd(_LCD_CLEAR);       // Limpa o display
   Lcd_Cmd(_LCD_CURSOR_OFF);  // Desliga cursor

   init_ADC(); // ADC Setup

   Lcd_Out(1,3, "Projeto Final");
   Lcd_Out(2,4, "DEL - UFRJ");
   Delay_ms(3000);
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,6, "Autor");
   Lcd_Out(2,2, "Cassio Carvalho");
   Delay_ms(3000);
```

```
Lcd_Cmd(_LCD_CLEAR);
Lcd_Out(1,2, "Aperte o botao");
Lcd_Out(2,3, "para iniciar.");
while(!Button(&PORTB, RB8, 8, 0)){}

switch_var = 0;
rmshoratot = 0;
rms_hora = 0;

while(1){

 switch(switch_var){

   case 0:
       lcd_cmd(_LCD_CLEAR);
       lcd_out(1,5, "Corrente");
       lcd_out(2,7, "RMS");
       delay_ms(3000);

       IRMS();

   case 1:
       lcd_cmd(_LCD_CLEAR);
       lcd_out(1,5, "Corrente");
       lcd_out(2,3, "RMS por Hora");
       delay_ms(3000);

       init_Interrupt();

       IRMS_hora();

   case 2:
       lcd_cmd(_LCD_CLEAR);
       lcd_out(1,7, "THD");
       delay_ms(3000);

       while(1){
       if(U1STAbits.OERR == 1)
        {
        U1STAbits.OERR = 0;
        continue;
        }

        trigger = 1;
```

```
            while(trigger == 1){
            while(U1STAbits.URXDA == 0){continue;}
            if(U1RXREG == 'i'){
            g = 0;
            trigger = 0;
            }
            }
            while(trigger == 0){
            while(U1STAbits.URXDA == 0){continue;}
            dado[g] = U1RXREG;
            g++;
            if(g == 4){
            trigger = 1;
            }
            }
           lcd_out(1,3, "Tot Dis Harm");
           lcd_out(2,3, "THD = ");
           lcd_out_cp(dado);
           lcd_out_cp("%");
           delay_ms(200);

           if(Button(&PORTB, RB8, 8, 0)){
            switch_var = 0;
           break; }

           }
     }
    }
}
```

Apêndice B

Código Fonte do dsPIC II (THD)

```
/*
Codigo do dsPIC2 - Calcula o THD e exporta os valores para o Display Lcd 16x2
OFICIAL
FIN = 40Mhz (Ressonador)
FCY = FOSC/2 = 20Mhz
*/

#define TAMANHO 20
#define N_Am    100
#define BAUDRATE 9600
#define BRGVAL ((20000000/BAUDRATE)/16)-1
#include "constantes.h"

// Definicao das variaveis
int             trigger, num, i;
float           num2;
char            dado[4] = {0};

// Definicao das variaveis a serem usadas pela biblioteca do LCD
sbit LCD_RS at LATB15_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB11_bit;
sbit LCD_D5 at LATB12_bit;
sbit LCD_D6 at LATB13_bit;
sbit LCD_D7 at LATB14_bit;

sbit LCD_RS_Direction at TRISB15_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB11_bit;
sbit LCD_D5_Direction at TRISB12_bit;
sbit LCD_D6_Direction at TRISB13_bit;
sbit LCD_D7_Direction at TRISB14_bit;

/********************* Total Harmonic Distortion ************************/
void THD(void){
    int   m, k, i, j;
    float coef_A, coef_B, Pmed, THD, X[N_Am], media_thd, vetor_thd[TAMANHO];
    char dado[5];

    coef_A = 0;
    coef_B = 0;
    Pmed = 0;
    media_thd = 0;
```

```
    // Coletar o sinal de entrada
    for(m=0; m<N_Am; m++){
     ADCPC0bits.SWTRG0=1;
     while(ADSTATbits.P0RDY=0)
     continue;
     X[m] = ADCBUF0;
     delay_us(161);
    }

    for(m=0;m<N_Am;m++){
      X[m] = X[m] - 512;
      X[m] = (X[m]/1024)*3.3;
    }

    // Calculo dos coeficientes da componente fundamental da serie de Fourier
    for(m=0; m<N_Am-1; m++){
       coef_A = (SENO[m]*X[m]+SENO[m+1]*X[m+1])/2+coef_A;  // Aqui entra os
coeficientes gerados pelo Matlab
       coef_B = (COSSENO[m]*X[m]+COSSENO[m+1]*X[m+1])/2+coef_B; //
    }

    coef_A = 2*coef_A/(N_Am-1);
    coef_B = 2*coef_B/(N_Am-1);

    // Calculo da Potencia media

    for(k=0; k<N_Am-1; k++){
       Pmed = (X[k]*X[k]+X[k+1]*X[k+1])/2+Pmed;
    }

    Pmed = Pmed/(N_Am-1);

    // Calculo do THD

    THD = sqrt(2*(Pmed-195e-4)/(coef_A*coef_A+coef_B*coef_B)-1)*100;

     for(j=0;j<TAMANHO-1;j++){
     vetor_thd[j]=vetor_thd[j+1];
     }

     vetor_thd[TAMANHO-1] = THD;

     for(j=0;j<TAMANHO;j++){
     media_thd += vetor_thd[j];
     }
     media_thd = media_thd/TAMANHO;

     sprintf(dado, "%.2f", media_thd);
```

```
    while(U1STAbits.UTXBF == 1){}
     U1TXREG = 'i';
     for(i=0;i<4;i++){
     while(U1STAbits.UTXBF == 1){}
     U1TXREG = dado[i];
     }
     U1TXREG = 'f';
     while(U1STAbits.UTXBF == 1){}
}

/******************** CONFIGURACAO UART ***************************/
void init_uart(void){

   U1MODEbits.STSEL = 0; // 1-stop bit
   U1MODEbits.PDSEL = 0; // No Parity, 8-data bits
   U1MODEbits.ABAUD = 0; // Auto-Baud Disabled
   U1MODEbits.BRGH = 0; // Low Speed mode
   U1BRG = BRGVAL; // BAUD Rate Setting for 9600
   U1MODEbits.UARTEN = 1; // Enable UART
   U1STAbits.UTXEN = 1; // Enable UART Tx

}
/******************** MODULO ADC ***************************/
void init_ADC(void){

  ADCONbits.ADON=0; // Desabilita o processo de conversao
  ADCONbits.ORDER=0; // Ordem do ADC par-impar
  ADCONbits.ADCS=0b001; // Divisor Fvco/2 => No caso o clock de 20 Mhz
  ADCONbits.FORM=0; // Formato de saida inteiro
  ADCONbits.ASYNCSAMP=0; // SampleHold sempre amostrando ate receber o
trigger da conversao
  ADSTATbits.P0RDY=0;  // Limpa o bit de status do Par 0
  ADCPC0bits.TRGSRC0=0b00001; // Seleciona o trigger individual por software

  ADCONbits.ADON=1; // Inicializa o processo de conversao

}
/******************** CONFIGURACAO DAS PORTAS ********************/
void init_config(void){

  ADPCFGbits.PCFG0=0;      // AN0 como entrada analogica
  ADPCFGbits.PCFG1=0;      // AN1 como entrada analogica

  TRISB=0; // PORTB como saida

  Unlock_IOLOCK();  // Habilitar mudancas de PPS
  PPS_Mapping(11, _INPUT, _U1RX); // RP11 como RX
  PPS_Mapping(5, _OUTPUT, _U1TX); // RP5 como TX
  Lock_IOLOCK();    // Desabilitar mudancas de PPS
}
```

```
/****************** PROGRAMA PRINCIPAL ************************/
void main(){

  init_config(); // Setup
  init_uart();


   Delay_ms(100);

   init_ADC(); // ADC Setup

   while(1){

   trigger = 1;

     while(trigger == 1){
     ADCPC0bits.SWTRG0=1;
     while(ADSTATbits.P0RDY=0)
     continue;
     delay_us(10);
     if(ADCBUF0 < 512){
     trigger = 0;
     }
     }
     while(trigger == 0){
     ADCPC0bits.SWTRG0=1;
     while(ADSTATbits.P0RDY=0)
     continue;
     delay_us(10);
     if(ADCBUF0 > 512){
     trigger = 1;
     }
     }
     THD();

   }

}
```