UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

ATUALIZAÇÃO DE HARDWARE E FIRMWARE DO PROTEGEMED

Júlio César dos Santos

Passo Fundo

2017

UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

ATUALIZAÇÃO DE HARDWARE E FIRMWARE DO PROTEGEMED

Júlio César dos Santos

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Prof. Dr. Luiz Eduardo Schardong Spalding

Coorientador: Prof. Dr. Marcelo Trindade Rebonatto

Passo Fundo

2017

CIP – Catalogação na Publicação

S237a Santos, Julio Cesar
Atualização de hardware e firmware do Protegemed / Julio Cesar Santos. – 2017.
73 f. : il. color. ; 30 cm.

Orientador: Prof. Dr. Luiz Eduardo Schardong Spalding.
Coorientador: Prof. Dr. Marcelo Trindade Rebonatto.
Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 201.

1. Protegemed. 2. Programas de computador. 3. Dispositivos eletrônicos. 4. Hardware. I. Spalding, Luiz Eduardo Schardong, orientador. II. Rebonatto, Marcelo Trindade, coorientador. III. Título.

CDU: 004

Catalogação: Bibliotecária Marciéli de Oliveira - CRB 10/2113

UPF Universidade de Passo Fundo | PPGCA
Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

JÚLIO CÉSAR DOS SANTOS

Aos quatorze dias do mês de dezembro do ano de dois mil e dezessete, às 14 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Estudo e implementação para atualização de hardware e firmware do Protegemed com placas texas tm4c1294"**, de autoria de Júlio César dos Santos, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Luiz Eduardo Schardong Spalding, Marcelo Trindade Rebonatto, Carlos Amaral Hölbig e Andréa Teresa Riccio Barbosa. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Luiz Eduardo Schardong Spalding - UPF
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Marcelo Trindade Rebonatto - UPF
(Coorientador)

Prof. Dr. Carlos Amaral Hölbig - UPF
(Avaliador Interno)

Profa. Dra. Andréa Teresa Riccio Barbosa - UFMS
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

ATUALIZAÇÃO DE HARDWARE E FIRMWARE DO PROTEGEMED

RESUMO

Durante as intervenções cirúrgicas as equipes médicas são auxiliadas por equipamentos eletromédicos. Sua utilização contribui para melhorar os procedimentos operatórios, entretanto, expõem os pacientes a alguns riscos, dentre estes, os eventos de microchoque. O Protegemed permite o monitoramento e a análise em tempo real de risco de tais eventos, contribuindo para a segurança dos pacientes. A arquitetura inicial do Protegemed vem sendo aprimorada desde o ano de 2004, quando utilizava um microcontrolador de 8 bits, em 2017 é utilizado um microcontrolador ARM Cortex M3 de 32 bits com recursos de memória RAM praticamente esgotados. Devido a necessidade de implementações de novas funcionalidades que permitam, por exemplo, a análise da tensão elétrica em conjunto com a corrente de alimentação aplicadas aos equipamentos médicos, uma nova plataforma deve ser utilizada. Para determinar esta nova versão do Protegemed uma pesquisa foi realizada entre as diversas plataformas de desenvolvimento disponíveis atualmente no mercado, em especial, microcontroladores e computadores de única placa. Com base nas novas especificações do projeto testes foram realizados em uma plataforma de cada família, onde o resultado foi a escolha do microcontrolador ARM Cortex-M4F TM4C. Após essa definição, a implementação detalhada do novo *firmware* é apresentada juntamente com os resultados comparativos entre as versões anteriores e a TM4C.

Palavras-Chave: Microchoque, Protegemed, TM4C .

HARDWARE AND FIRMWARE UPDATE IN THE PROTEGEMED SYSTEM

ABSTRACT

During surgical interventions, medical teams are assisted by electromedical equipment. Its use contributes to improve operative procedures, however, exposing patients to some risks, among them, microchoke events. Protegemed allows the monitoring and real-time risk analysis of such events, contributing to patient safety. The Protegemed initial architecture has been improved since 2004, when its using an 8-bit microcontroller, in 2017 a 32-bit ARM Cortex M3 microcontroller with nearly exhausted RAM resources is used. Due to the need for implementations of new functionalities that allow, for example, analysis of the electrical voltage in conjunction with the supply current applied to medical equipment, a new platform must be used. To determine this new version of Protegemed a research was conducted among the various development platforms currently available in the market, especially microcontrollers and single board computers. Based on the new project specifications tests were carried out on a platform of each family, where the result was the choice of ARM Cortex-M4F TM4C microcontroller. After this definition, detailed implementation of the new firmware is presented along with the comparative results between previous versions and TM4C.

Keywords: Microchoke, Protegmed, TM4C.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

EEM. – Equipamento eletromédico

ddp. – Diferença de potêncial

fem. – Força eletromotriz

V. – Volt

CC. – Corrente contínua

CA. – Corrente alternada

A. – Ampere

J. – Joule

W. – Potência elétrica ($1W = 1J/s$)

T. – Período

f. – Frequência

Hz. – Hertz (ciclos por segundo)

FO. – forma de onda

s. – Segundo

y_p. – Amplitude de pico

y_{pp}. – Amplitude de pico a pico

rms. – Valor eficaz (do inglês Root mean square, raiz média quadrática)

DTFT. – Discrete time Fourier transfor (Transformada de Fourier de tempo discreto)

DFT. – Discrete Fourier Transform (Transformada discreta de Fourier)

DSP. – Digital signal processing (Processamento digital de sinais)

FFT. – Fast Fourier transform (Transformada rápida de Fourier)

LSB. – Least Significant bit (bit menos significativo)

T_s. – Período de Amostragem

f_s. – Frequência de amostragem

SBC. – Single Board Computers (Computadores de única placa)

μC. – Microcontrolador

SO. – Sistema Operacional

E/S. – Entrada e saída

MSP. – Mixed Signal Processors

RTOS. – *Real time operating system* (sistema operacional de tempo real)

FIFO. – Fist in fisrt out (primeiro a entrar é o primeiro a sair)

MSPS. – Mega Sample per Second (Milhões de amostra por segundo)

HTML. – HyperText Markup Language

DA. – Digital para analógico

DMA. – Direct memory access (acesso direto a memória)

CCS. – Code Composer Studio (Ambiente de desenvolvimento da TM4C)

IDE. – Integrated development environment (ambiente de desenvolvimento integrado)

TI. – Texas Instruments

csv. – comma-separated values

TI-RTOS. – Texas Instruments Real Time Operating System

API. – *Application Programming Interface*

ROV. – RTOS Object View

PE. – Protection earth (condutor de aterramento/proteção)

TC. – Transformador de corrente

RC. – Resistor Capacitor

CMSIS DSP. – *Cortex Microcontroller Software Interface Standard for Digital Signal Proccesing*

NDK. – Network Development Kit

HAL. – Hardware Adaptation Layer (camada de abstração de *hardware*)

HTTP. – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)

SUMÁRIO

1. INTRODUÇÃO

O Protegemed vem sendo aperfeiçoado desde o ano de 2004 e consiste de um equipamento eletrônico e um software que são utilizados em uma sala de cirurgia para detectar quando há um problema elétrico que possa colocar em risco a saúde do paciente. Seu funcionamento será detalhado na seção 2.3 deste texto. Inicialmente, o Protegmed utilizava um microcontrolador (μC) de 8 *bits* que monitorava uma única tomada onde eram conectados os equipamentos eletromédicos (EEM). Naquele projeto inicial, devido as características do μC, eram necessários vários circuitos periféricos externos, como por exemplo, memória *RAM*, conversor Analógico Digital (AD) e relógio em tempo real (*RTC - Real Time Clock*)[1]. A comunicação utilizada era baseada no protocolo serial RS232, que possui baixa imunidade a ruído e taxa de comunicação dependente da distância entre os dispositivos [2]. O μC era responsável pelo envio dos dados coletados pelo hardware de condicionamento de sinal, não realizando nenhuma manipulação adicional. Para inspecionar os dados coletados, utilizava-se um software instalado em um computador pessoal (*PC - Personall Computer*), este era responsável pela análise dos sinais no domínio da frequência, através da transformada rápida de Fourier *(FFT - Fast Fourier Transform)*, e pela apresentação gráfica dos resultados.

Com a versão Protegemed2 (versão 2010), melhoras significativas ocorreram no hardware de supervisão, a adoção de um μC da plataforma ARM, de 32bits, proporcionou aumento substancial no número de periféricos integrados no próprio μC (conversor analógico/digital (conversor A/D), RTC, modulação por largura de pulso (PWM), ethernet, barramento *SPI* entre outros). A comunicação passou a utilizar a arquitetura *ethernet*, os cálculos da FFT são realizados no próprio μC e o número de tomadas monitoradas cresceu de uma para quatro [3].

A versão atual (versão 2012) utiliza o NXP LPC1768 (*mbed*), monitora 3 tomadas e incorpora identificação do EEM conectado através da tecnologia de identificação por radiofrequência (do inglês *Radio-Frequency IDentification* RFID). Esta versão encontra-se embarcada no interior do painel de gases do Hospital São Vicente de Paulo (HSVP), Passo Fundo [3]. Neste texto será denominada simplesmente como *mbed*, em referência ao μC utilizado.

Em um sistema embarcado, ao contrário de *desktop* ou *notebook*, os recurso de memória são limitados e, geralmente, não expansíveis [4]. No *mbed* a memória RAM disponível é de 64Kb, e está sendo utilizada quase que totalmente, podendo ser a causa de algumas falhas que resultam no travamento da execução do *firmware*. Tal travamento necessita de um *reset* manual e portanto o deslocamento de um técnico para realizar o serviço. Além da limitação de memória, também existem limitações no *hardware* referente ao número de entradas analógicas disponíveis. Essa limitação obriga a utilização de dois *mbed* por painel de tomadas monitorado, além da instalação de uma *switch* de rede.

O Protegemed tem sido objeto de pesquisas no campo da segurança dos EEM, tal característica exige constantes atualizações de *hardware* e *firmware* conforme exposto acima. No mercado atual existem diversas opções, que vão desde microcontroladores a sistemas embarcados com sis-

tema operacional já instalado. Tal diversidade dificulta a definição de um padrão, tanto de *hardware* quanto de *software*, específico para sistemas embarcados [5].

Desta forma, o problema de pesquisa é descobrir se existe no mercado alguma plataforma de prototipagem rápida com características de *hardware* que se enquadre nas especificações da atualização necessária para o Protegmed e que seja possível construir, a partir desta plataforma, uma versão comercial e financeiramente viável.

Considerando este problema, o objetivo é escolher uma plataforma e testá-la para obter o atendimento integral de todas a funcionalidades implementadas na versão mbed, além de aumentar a disponibilidade de memória RAM, melhorar os recursos na aquisição de dados e proporcionar a atualização de firmware de forma remota. A plataforma escolhida também precisa ter disponibilidade comercial e fornecer ferramentas de desenvolvimento como compiladores e depuradores [6].

O presente trabalho apresenta no capítulo 2 uma revisão sobre grandezas elétricas e ferramentas matemáticas, bem como faz uma rápida abordagem sobre o Protegemed. No capítulo **??**, apresenta algumas das principais plataformas disponíveis no mercado atualmente, define especificações para a escolha da nova plataforma e apresenta as ferramentas e materiais escolhidos para confeccionar o protótipo. No capítulo 5 são apresentados os testes realizados com o protótipo, que é chamado, a partir daquele momento, de Protegemed *TM4C* ou simplesmente *TM4C*. No capítulo 6 são apresentadas as conclusões e as sugestões para trabalhos futuros.

2. REVISÃO DE LITERATURA

2.1 ELETRICIDADE E ANÁLISE DE CIRCUITOS

Neste capitulo são apresentados alguns conceitos básicos de eletricidade. Considerações sobre valor eficaz e transformada de Fourier são descritas por sua relevância ao projeto do Protegemed.

2.1.1 Tensão, Corrente, Potência e Frequência

A tensão elétrica, também chamada de diferença de potencial (ddp) ou ainda força eletromotriz (fem), pode ser definida como a capacidade de se realizar trabalho quando os elétrons forem forçados a se movimentar [7]. Sua unidade de medida é o volt (V) .

A corrente elétrica, ou fluxo de elétrons, é originada pela tensão. Esta, pode ser classificada como contínua (CC), quando os elétrons se movem apenas em uma direção, ou alternada (CA),quando existe um movimento alternado no fluxo de elétrons [7]. A unidade de medida da corrente elétrica é o ampere (A).

A taxa com a qual a energia, expressa em joule , é gerada ou utilizada, é denominada potência elétrica e é medida em watts (W)[1]. Quando uma bateria fornece corrente a uma carga, a energia gerada pela bateria é consumida pela carga. A potência elétrica instantânea é obtida pelo produto da tensão e da corrente sobre esta carga, equação 1 [8].

$$P = V \cdot I \quad [W] \tag{1}$$

A tensão CC geralmente, mas não exclusivamente, é obtida através de processos eletroquímicos. Como exemplo pode-se citar pilhas e baterias. A tensão CA, por sua vez, tem como princípio o eletromagnetismo e é obtida através de geradores. Devido ao movimento mecânico rotacional do rotor do gerador, a tensão CA tem formato senoidal, equação 2.

$$V = V_p \cdot cos(\omega t + \phi) \quad [V] \tag{2}$$

Onde: Vp representa a amplitude da tensão, ω a frequência angular em rad/s variando em função do tempo t em segundos e ϕ o deslocamento de fase em radianos.

O tempo necessário para a tensão CA completar um ciclo é chamado de período (T) e é expresso em segundos . O número de ciclos completos em um segundo é denominado frequência (f) e tem como unidade de medida o Hertz (Hz) . A frequência é calculada a partir do período pela equação 3.

[1] $1\frac{J}{s} = 1W$

$$f = \frac{1}{T} \quad [Hz] \tag{3}$$

A frequência também pode ser expressa em função do ângulo conforme a equação 4.

$$\omega = 2\pi \cdot f \quad [rad/s] \tag{4}$$

A velocidade do rotor e o número de pólos do gerador definem a frequência da tensão gerada[9]. No Brasil a frequência da rede de distribuição é de 60Hz ou 377rad/s.

2.1.2 Forma de Onda

Sinais elétricos podem ser representados graficamente em função do tempo através de uma forma de onda (FO) [9]. Muitos sinais elétricos são periódicos, isto é, se repetem a cada T segundos [10]. Um exemplo de FO periódica é mostrado na Figura 1. Quando analisamos FOs de sinais periódicos alguns parâmetros básicos precisam ser definidos.

- Período (T): Intervalo de tempo em segundos (s) para completar um ciclo.
- Frequência (f): Quantidade de ciclos completos em um segundo, expressa na unidade Hertz (Hz).
- Amplitude de pico (y_p) : Valor máximo, valor de pico, da FO em relação ao eixo horizontal (valor zero).
- Amplitude pico a pico (y_{pp}) : Diferença entre o máximo positivo e o máximo negativo da FO.

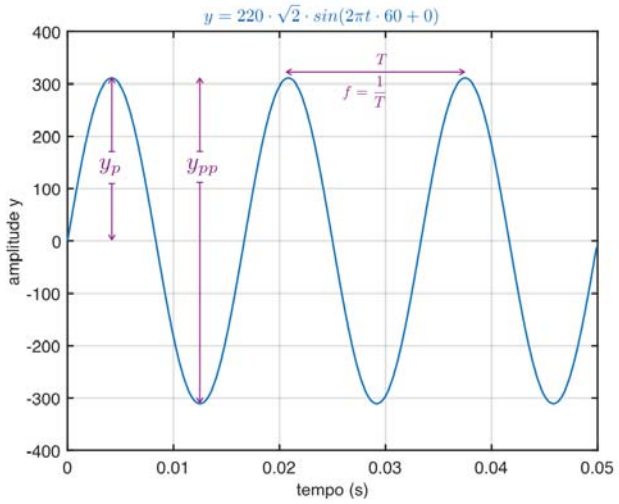


Figura 1. Forma de onda periódica

Esses parâmetros são válidos tanto para a tensão quando para a corrente, desta foram, deve-se substituir a variável dependente y por v ou i. Existem diversas FOs periódicas não senoidais, como

por exemplo dente de serra, triangular e quadrada, destas, algumas podem ou não apresentar parcela negativa.

2.1.3 Valor Eficaz (rms)

O valor eficaz, ou rms, de uma tensão elétrica CA (ou corrente) é uma constante que representa o valor necessário para entregar a mesma potência que uma fonte de tensão CC(ou corrente) entregaria a uma carga resistiva[11]. O termo rms é a abreviatura do inglês *root mean square* (rms) , ou, raiz média quadrática. Neste texto trataremos o valor eficaz como rms. Matematicamente, pode-se expressar a tensão rms para o sinal senoidal da equação 2 através da equação 5 [10].

$$V_{rms} = \sqrt{\frac{1}{T}\int_{t_0}^{t_0+T} V_p^2 \cdot cos^2(\omega t + \phi)dt} \quad [V_{rms}] \tag{5}$$

Quando a tensão é puramente senoidal, pode-se simplificar os termos da equação 5 na forma da equação 6.

$$V_{rms} = \frac{V_p}{\sqrt{2}} \quad [V_{rms}] \tag{6}$$

As equações 5 e 6 são validas para sinais contínuos e periódicos, para sinais discretos o valor rms é obtido conforme a equação 7.

$$Vrms = \sqrt{\frac{1}{N}\sum_{n=1}^{N} |V_n^2|} \quad [V_{rms}] \tag{7}$$

O vetor V contém N valores discretos de tensão, o valor rms é obtido extraindo a raiz quadrada da soma dos quadrados dos elementos individuais divididos pelo número total de elementos presentes em V [12].

No Protegmed os valores rms de corrente, fase e diferencial, são utilizados na detecção de eventos de liga, desliga, início de fuga e fim de fuga. Na versão em desenvolvimento, são calculados também os valores rms da tensão e da corrente no condutor de aterramento.

2.1.4 Métodos de Fourier

Em 1822 o matemático Jean Baptiste Joseph Fourier publicou um trabalho sobre a teoria matemática de condução de calor. Este trabalho deu origem a famosa série de Fourier [11]. Fourier afirma que uma função periódica pode ser representada por uma soma infinita de funções seno ou cosseno de frequências múltiplas [10].

Uma função periódica pode ser representada pela série de Fourier conforme a equação 8[11]

.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \, cos \, n\omega_0 t \; + \; b_n \, sen \, n\omega_0 t) \tag{8}$$

onde: $f(t)$ é a função periódica, a_0, a_n e b_n são os coeficientes de Fourier da função $f(t)$ e ω_0 é a frequência fundamental[2] da função $f(t)$.

Os coeficientes de Fourier são obtidos através das equações 9, 10 e 11 [11].

$$a_0 = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) dt \tag{9}$$

$$a_k = \frac{2}{T} \int_{t_0}^{t_0+T} cos \, k\omega_0 t \; dt \tag{10}$$

$$b_k = \frac{2}{T} \int_{t_0}^{t_0+T} sen \, k\omega_0 t \; dt \tag{11}$$

onde: T é o período da função $f(t)$ e k representa o k-ésimo coeficiente na sequência de números inteiros 1, 2, 3... Na figura 2 é mostrado um exemplo de uma FO dente de serra e sua reconstrução usando as dez primeiras somas da série de Fourier.

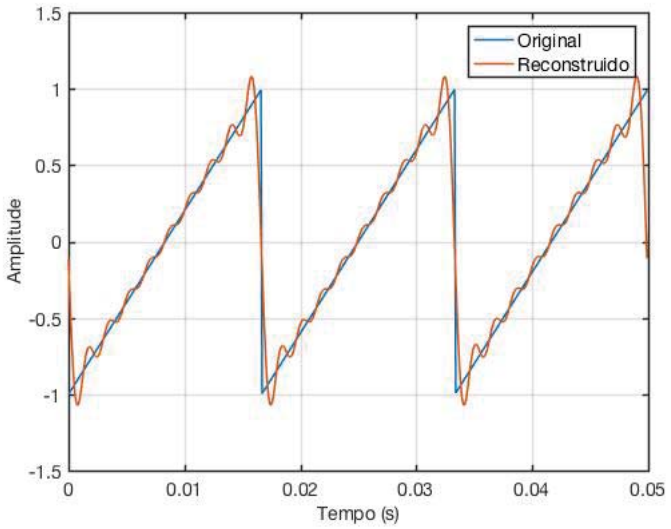


Figura 2. Sinal original e sua reconstrução usando a série de Fourier.

A utilização de mais termos na soma da série leva, de forma obvia, a uma reconstrução mais próxima do sinal original.

[2] ω_0 pode ser substituído por $2\pi/T$ ou $2\pi \cdot f$ quando f é expresso em Hertz

2.1.5 Transformada Discreta de Fourier

A transformada de Fourier converte um sinal no domínio do tempo para o domínio da frequência e a transformada inversa de Fourier converte um sinal no domínio da frequência para o domínio do tempo [13]. Um sinal pode ser tanto contínuo quanto discreto, periódico ou aperiódico, a partir destas características pode-se classificar o termo transformada de Fourier em quatro tipos [14].

- Transformada de Fourier: Sinal contínuo e aperiódico. Ex.: Curva de Gauss e exponenciais decrescentes.
- Séries de Fourier: Sinal contínuo e periódico. Ex.: onda senoidal, onda quadra, onda triangular e dente de serra (como a apresentada na figura 2).
- Transformada de Fourier de tempo discreto (DTFT) : Sinal discreto e aperiódico. Ex.: Sinais definidos entre o infinito positivo e negativo que não se repetem de forma periódica.
- Transformada discreta de Fourier(DFT): Sinal discreto e periódico. Ex.: Sinais definidos entre o infinito positivo e negativo que se repetem de forma periódica.

A transformada utilizada no processamento digital de sinais (DSP) é a DFT. Para um sinal discreto $x(nT)$ a DFT é obtida aplicando a equação 12 [14].

$$X(k) = \sum_{n=0}^{N-1} x(n)W^{nk} \qquad k = 0, 1, \ldots, N-1 \tag{12}$$

Onde: T é o período de amostragem de $x(n)$, N é número de amostras do vetor x e W representa o fator de fase, equação 13.

$$W = e^{-j2\pi/N} \tag{13}$$

A equação 12 pode ser escrita para $k = 0, 1, \ldots, N-1$ pela equação 14.

$$X(k) = x(0) + x(1)W^{k} + x(2)W^{2k} + \cdots + x(N-1)W^{(N-1)k} \tag{14}$$

A equação 14 representa uma matriz $N \times N$, uma vez que $X(k)$ precisa calcular N valores para k. A complexidade computacional da DFT é portanto N^2 [13]. A transformada rápida de Fourier (FFT) é um algoritmo eficiente para calcular a DFT, esta utiliza a periodicidade, equação 15, e a simetria, equação 16, de W para reduzir a complexidade de N^2 para $N \log N$ [15].

$$W^{k+N} = W^{k} \tag{15}$$

$$W^{k+N/2} = -W^{k} \tag{16}$$

Como a FFT possui menor complexidade computacional comparada a DFT, este algoritmo é amplamente utilizado em ambientes embarcados.

2.2 PROCESSAMENTO DIGITAL DE SINAIS

O processamento digital de sinais (DSP) é diferenciado de outras áreas da ciência da computação pelo tipo único de dados que utiliza, sinais. A origem destes sinais podem ser ondas sonoras, imagens, sensores pressão, velocidade, altitude, etc. O DSP utiliza algoritmos matemáticos para manipular tais sinais com um determinado propósito [14]. No Protegemed os valores de tensão, corrente e fuga são tratados pelo *firmware* com o propósito de identificar eventos específicos. Neste capítulo, serão abordados os conceitos fundamentais utilizados em um sistema de aquisição de dados.

2.2.1 Quantização

A maioria dos sinais encontrados na natureza são contínuos com amplitude variando em função do tempo. Para que tais sinais possam ser processados na forma digital, estes devem passar pelo processo de quantização . A quantização converte uma sinal de contínuo para discreto [16]. Para qualquer amostra de um sinal digitalizado o erro de quantização é de ½ LSB (Least Significant bit - bit menos significativo) . Para um conversor AD com N bits de resolução e V_{ref} volts de referência, o erro de quantização em volts pode ser obtido através da equação 17.

$$Q_{err} = \frac{V_{ref}}{2 \cdot 2^N} \quad [V] \tag{17}$$

Desta forma, o sinal digitalizado é o resultado do sinal amostrado somado ao erro de quantização. Na maioria dos casos, o erro de quantização resulta em ruído randômico uniformemente distribuído entre $\pm 1/2$LSB, com média zero e desvio padrão de $1/\sqrt{12}$LSB (≈ 0.29LSB). Assim, um conversor AD de 8 bits adiciona um ruído de $0,29/256 \approx 1/900$ do valor de fundo de escala, enquanto que um conversor AD de 12 bits adiciona $0,29/4096 \approx 1/14000$. Ou seja, aumentando a resolução do conversor AD aumentamos também a precisão do sinal amostrado[14].

2.2.2 Amostragem

Amostras do sinal analógico são tomadas em intervalos de tempo fixo. Este intervalo é chamado período de amostragem (T_s) . A frequência de amostragem (f_s) é o inverso do T_s. O teorema da amostragem, ou teorema da amostragem de Shannon-Nyquist, declara que um sinal é adequadamente amostrado somente se este não possuir frequências maiores que a metade da frequência de amostragem [14]. Na prática, isso significa que a f_s deve ser no mínimo duas vezes a máxima frequência, presente no sinal (B), para que a reconstrução possa ser realizada de forma adequada. Afirmação expressa pela equação 18.

$$f_s > 2B \tag{18}$$

A relação $f_s/2$ é conhecida como frequência de Nyquist. Quando o sinal amostrado apresenta frequências maiores que a frequência de Nyquist, ocorre o fenômeno de *aliasing*. Na figura 3 são apresentados quatro sinais com frequências distintas e mesmo período de amostragem ($f_s = 1600Hz$).

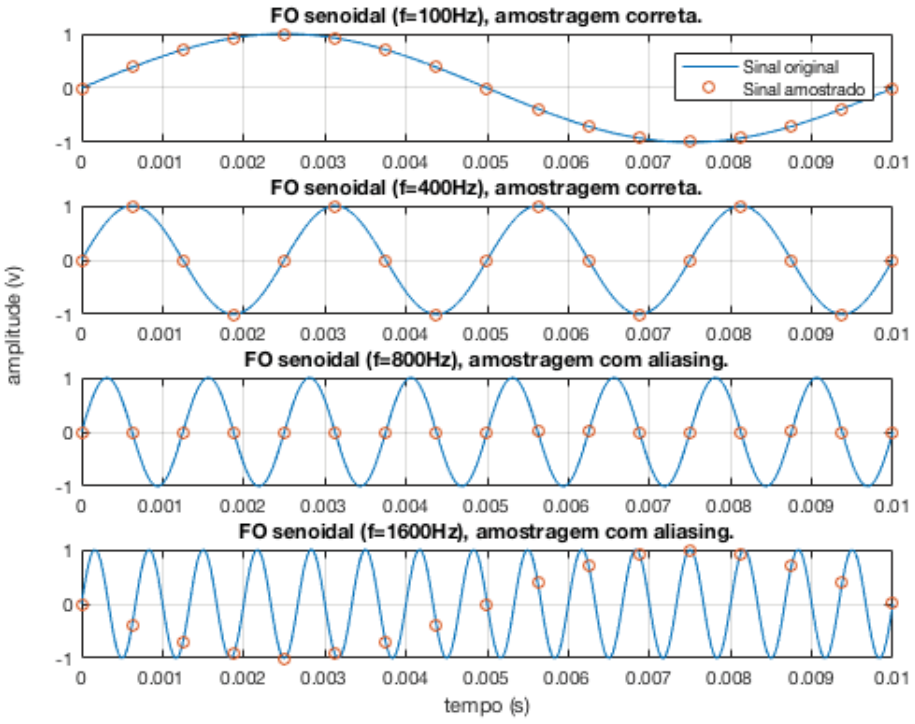


Figura 3. Efeito de *aliasing* na amostragem de sinais [17].

É possível observar que, conforme a frequência do sinal de entrada se aproxima da frequência de Nyquist ($f_s/2$), o perfil da onda amostrada se degrada, isto é, em uma eventual reconstrução do sinal no domínio do tempo, este não reproduzirá fielmente sua forma original. Exatamente em $f_s/2$, o sinal amostrado é nulo, o que obviamente não é verdadeiro para o sinal de entrada. Acima de $f_s/2$, o sinal amostrado possui frequência menor que o sinal de entrada. Apesar do teorema da amostragem declarar que f_s deve ser pelo menos duas vezes maior que a maior frequência presente no sinal de entrada, na prática, uma relação de 10 vezes é necessária para a reconstrução apropriada do sinal amostrado [17].

2.3 O SISTEMA PROTEGEMED

O sistema Protegemed é um produto eletrônico que une *hardware* e *software* no monitoramento das correntes de fuga dos equipamentos eletromédicos (EEM). Os eventos de correntes de fuga, que neste texto são caracterizadas como fuga de corrente elétrica para um caminho impróprio (seja através do paciente ou dos equipamentos e instalações), são capturados durante o procedimento

cirúrgico e podem representar risco a saúde do paciente e da equipe médica. O Protegemed oferece uma solução capaz de detectar e informar a ocorrência de tais eventos sem contudo influenciar no funcionamento do EEM. Os valores capturados pelo *hardware* são enviados a um servidor e armazenados em um banco de dados. Uma interface com o usuário permite várias análises e tomadas de decisão.

2.3.1 Visão geral do Protegemed

O circuito que fornece alimentação a sala de cirurgia é fornecido no sistema de aterramento TN-S, caracterizado pela separação dos condutores neutro e terra. Este circuito é direcionado ao quadro de disjuntores, ao transformador de separação e a um dispositivo de supervisão de isolamento. Deste ponto em diante a configuração de aterramento é modificada para o sistema IT-Médico [1]. O circuito de alimentação é então direcionado ao painel de tomadas e gases do centro cirúrgico onde o Protegemed encontra-se embarcado [3]. Uma visão geral da aplicação é apresentada na Figura 4

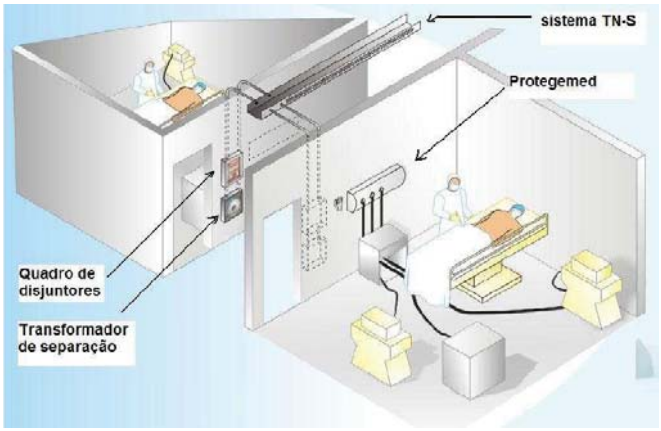


Figura 4. Visão geral do Sistema Protegemed [1].

Os EEMs são conectados às tomadas localizadas no painel de gases, estas tomadas são monitoradas pelo Protegemed. O monitoramento consiste basicamente em três etapas: aquisição, análise e comunicação.

2.3.2 Etapa de Aquisição

Na etapa de aquisição, os eventos de fuga são capturados durante o procedimento cirúrgico. Isto é feito monitorando a corrente diferencial dos condutores da tomada que alimenta o EEM.

Para isso utiliza-se um transformador de corrente (TC) na configuração diferencial. Nesta configuração ambos os condutores de alimentação, $F1$ e $F2$, passam no interior do TC caracterizando assim o enrolamento primário do transformador. A Figura 5 apresenta a configuração diferencial do TC onde o EEM é representado pela impedância Z.

No enrolamento secundário é ligado a uma carga resistiva R, sendo que a tensão sobre esta carga é proporcional a diferença entre as correntes de alimentação e é lida através de Vd. Em um caso ideal a tensão induzida ao secundário é nula, pois a corrente de alimentação $I1$ deve ser igual a corrente de retorno $I2$. Um desequilíbrio entre as correntes $I1$ e $I2$ caracteriza uma fuga de corrente para um caminho impróprio.

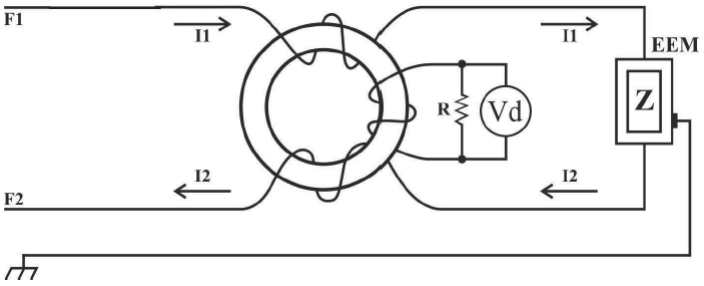


Figura 5. Transformador de Corrente na configuração diferencial [1].

O valor lido na resistência de carga R do TC é condicionado por um amplificador de instrumentação e inserido no canal de conversão AD do μC. Esta aquisição caracteriza-se por um conjunto de 256 valores discretos, com resolução de 12 bits, de uma forma de onda (FO) da corrente diferencial em um ciclo de 60Hz.

Um segundo TC é utilizado para monitorar o valor eficaz e a FO da corrente de alimentação do EEM. Neste caso, o tempo de utilização (tempo em que o EEM fica ligado) e a FO são utilizados no acompanhamento de sua vida útil.

Para determinar qual EEM está conectado a tomada que está sendo monitorada, um circuito de identificação por rádio frequência (RFID – Radio Frequency Identification) é utilizado. O receptor está instalado no entorno da tomada, enquanto o transmissor é posicionado no plugue de conexão do EEM. Na Figura 6 é apresentado o diagrama de blocos para o *hardware* do sistema Protegemed.

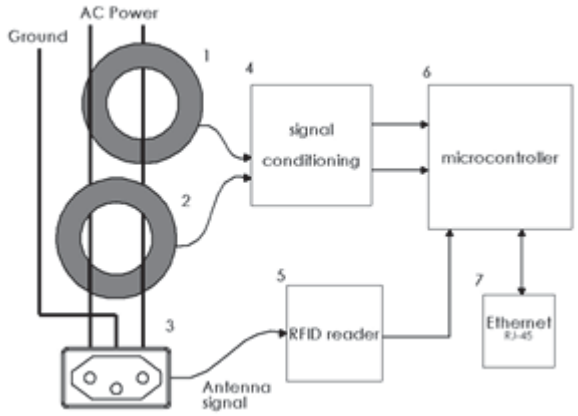


Figura 6. Diagrama de blocos do sistema Protegemed [1].

Os dados coletados pelo condicionador de sinal e pelo circuito de *RFID* são então analisados pelo *firmware* do μC.

2.3.3 Etapa de Análise

Na análise, os dados lidos pelo AD são processados para se obter o valor eficaz da corrente diferencial e então comparados com limites estabelecidos previamente, por exemplo, se este valor excedeu 50 microamperes (μA) um evento de fuga é gerado.

O tempo de uso do EEM é obtido pelo registro de eventos de “LIGA” (corrente de alimentação acima de 0,1 A, por exemplo) e “DESLIGA” do EEM. Para isso, cada EEM possui um cadastro, onde constam, além do tipo de EEM e modelo, as FOs de referência, como no momento de sua compra e a cada ano ou a cada manutenção preventiva realizada. Isto é importante porque os valores eficazes das correntes diferencial e alimentação e as suas FOs de referência são comparados com os valores atuais de corrente eficaz e FO. Pretende-se com isto verificar se é possível determinar o grau de deterioração do isolamento do EEM.

2.3.4 Etapa de Comunicação

Atualmente esta etapa ocorre somente quando:

a) Um evento de fuga é detectado.

b) Um evento de LIGA é detectado.

c) Um evento de DESLIGA é detectado.

d) Evento de final de fuga.

Em cada um destes casos, os dados e a identificação do EEM são agrupados para a transmissão através da porta *ethernet*. Estes dados são gravados no banco de dados do PC responsável pelo monitoramento.

2.3.5 Situação Atual

Atualmente, o Protegemed encontra-se embarcado no interior do painel de gases da sala 1 do Centro Cirúrgico do Hospital São Vicente de Paulo (HSVP) de Passo Fundo, RS. Neste painel são instaladas 6 tomadas a serem monitoradas, cada tomada necessita de dois canais analógicos, um para o sinal de corrente diferencial e outro para a corrente de alimentação, totalizando 12 canais analógicos por painel conforme apresentado na Figura 7.

O μC utilizado atualmente é um *mbed* NXP LPC1768 [3]. Cada *mbed* possui 6 canais AD, portanto dois deles são necessários para cada painel, além disso, um *switch* de rede é utilizado para a conexão dos kits à rede *ethernet* do hospital. As atualizações de *FW* são realizadas pela porta USB do *mbed* sendo necessário abrir o painel de gases para realizar as atualizações de *firmware*. Tal tarefa

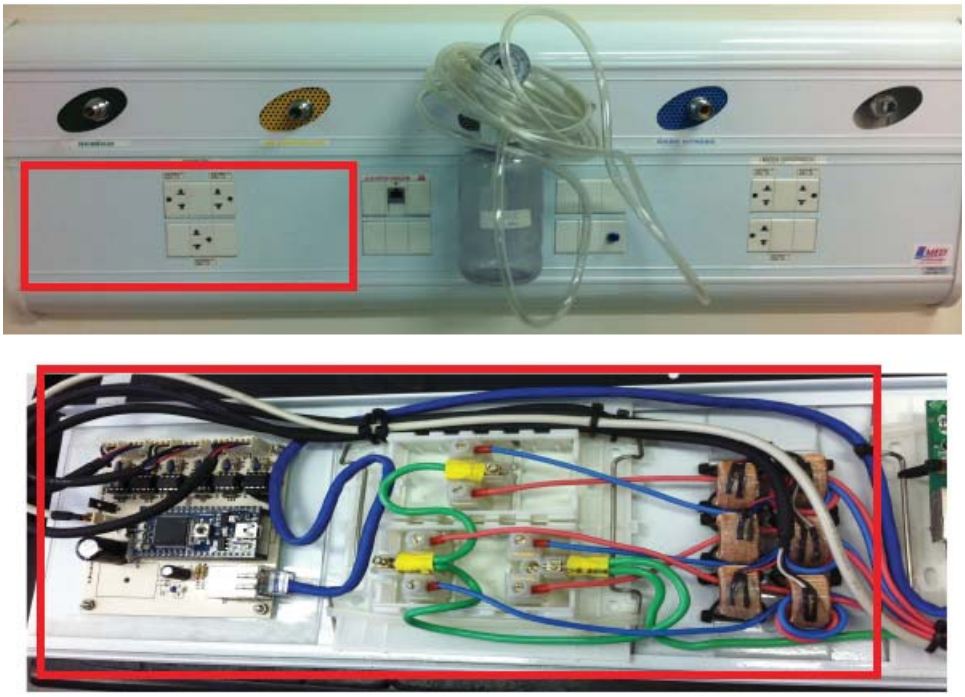


Figura 7. Detalhe do sistema Protegemed embarcado no painel de gases do HSVP [3].

requer, além de tempo, acesso ao Centro Cirúrgico, o que somente é possível mediante a solicitação prévia e agendamento.

Em eventuais falhas no *firmware* de controle um *reset* manual se faz necessário, o que atualmente é feito desligando a alimentação do módulo de controle, para isso é necessário deslocamento de um técnico até o quadro de alimentação.

3. ESPECIFICAÇÕES PARA A ATUALIZAÇÃO

Com base nas características apresentadas, foram definidas as especificações básicas para a escolha de uma nova plataforma de hardware, tais especificações são listadas abaixo:

- Mínimo de 12 canais analógicos: Com 12 canais AD é possível monitorar um painel de gases completo (possui 6 tomadas ao todo) e ainda eliminar o *switch* de rede atualmente utilizado. Entretanto, para as novas funcionalidades são desejáveis mais canais. Formas de se conseguir mais canais AD serão abordadas nas seções posteriores.

- Interface *ethernet* integrada (Possível *Wi-Fi*): A versão atual já conta com comunicação *ethernet*, um possível *upgrade* para isso seria a utilização de uma interface *wireless*, o que possibilitaria a eliminação da conexão física do cabo de rede.

- Memória *RAM* $\geq$ 128KB: A memória disponível na versão atual é de 64kb (utilizando a memória estendida) [3]. A medida que mais *threads* são adicionadas ao *firmware* atual, a necessidade de memória *RAM* aumenta, chegando ao ponto de estar quase que totalmente esgotada atualmente.

- Monitoração da Tensão Aplicada ao EEM: Esta é uma função adicional ainda não existente na versão atual. Este monitoramento permitiria uma melhor analise na determinação da vida útil do EEM. Esta é uma pesquisa em andamento no PPGCA.

- Eliminação do *switch* de rede: Na versão atual, dois dispositivos Protegemed são necessários para monitorar um único painel, limitação imposta pelo número de A/D's do *mbed*. Com o aumento no número de A/D's por dispositivo, a *switch* poderia ser eliminada.

- Custo por painel menor que o atual: Redução do custo e da complexidade de montagem, sempre é uma característica desejável em sistemas embarcados.

- Comunicação Bidirecional: A definição de novos valores limites de corrente de fuga para cada EEM, deve ser possível forma remota.

- Atualização de *firmware* remota: Atualmente é necessária uma conexão física com o módulo *mbed*, através da porta USB, quando é necessário atualizar o *firmware*. A nova plataforma fará isto de forma remota.

- Memória de programa capaz de atender futuras implementações: Novas funcionalidades implicam diretamente na utilização de mais memória de programa. Diante disso, o tamanho da memória de programa deve ser necessariamente maior na nova plataforma.

A partir destas especificações, foram pesquisadas plataformas disponíveis no mercado capaz de substituírem o sistema atual e suportar novas funcionalidades.

3.1 ALTERNATIVAS DE PLATAFORMAS

A solução tradicional para o desenvolvimento de produtos com eletrônica embarcada (*Embedded System*) é a aplicação de Microprocessadores/Microcontroladores [18]. Atualmente uma ampla gama *Single Board Computers* (*SBC*'s ou computadores de única placa) estão disponíveis no mercado, principalmente pela grande demanda dos dispositivos móveis [19]. Para a atualização de um sistema já existente, como o Protegemed, uma pesquisa que reúna as duas possibilidades se faz necessária. A seguir serão apresentados alguns dos dispositivos, Microcontroladores (μC) e SBC's, disponíveis no mercado, abordando suas características e diferenças.

3.1.1 *Single Board Computer - SBC*

Um SBC possui todas as funcionalidades de um computador completo montado em uma única placa de circuito impresso. Atualmente são baseados, em sua maioria, na arquitetura ARM Cortex-A. Sua principal característica é a presença de um Sistema Operacional (SO) embarcado, combinando interfaces eletrônicas de baixo nível, como pinos de entrada/saída (E/S) , e barramentos de comunicação [20].

A partir das especifícações expostas na subseção 3, foram selecionadas algumas plataformas para análise. Dentre estas, na Tabela 1 são listadas suas principais características relevantes ao projeto.

Tabela 1. Lista de Plataformas SBC.

Plataforma	Núcleos	Clock (GHz)	RAM (MB)	ADs	Wi-fi	Tamanho (mm)	Custo (US$) [3]
Banana-Pi	8	1.8	2048	-	Sim	92×60	74,00
Beaglebone Black	1	1	512	8	Não	86.4×53.3	55,00
Intel Galileo Gen 2	1	0,4	256	6	Não	124×72	45,00
Odroid C1+	4	1.5	1024	2	Não	85×56	40,00
Raspberry Pi 3 Model B	4	1.2GHz	1024	-	Sim	85.6×56.5	35,00
Udoo	2	1GHz	1024	12	Sim	110×85	115,00

Todos os SBCs avaliados possuem porta de comunicação *ethernet* e memória *RAM* superior a especificação inicial, entretanto, apresentam baixo número de canais ADs integrados.

A placa Udoo Dual Lite atende ao requisito mínimo de canais AD definido nas especificações, entretanto, não deixa nenhum outro canal AD livre para futuras implementações. Nenhuma das demais plataformas atendem a este requisito de forma nativa, porém, a utilização destas ainda é possível com a instalação de CIs conversores AD externos ligados a um barramento de comunicação, SPI ou I2C, presentes em todos os SBCs listados.

A utilização de AD externos pode representar uma perda de desempenho no software de controle, pois este deve dedicar tempo de processamento a tarefa de leitura sequencial dos dados

[3] Custo em junho de 2016.

convertidos pelos ADs. Duas destas plataformas possuem características especiais de hardware que permitem sua aplicação sem grande impacto ao *software* de controle e serão detalhadas a seguir.

3.1.1.1 *BeagleBone Black*

Este *SBC* possui tamanho e custo reduzido, grande número de pinos de interface, cartão SD, memória *embedded Multi Media Card (eMMC)* interna e 7 conversores AD integrados conforme apresentado na Figura 8. Trata-se da união de um *software* de alto nível (SO) com eletrônica de baixo nível. Diversas distribuições de SO's estão disponíveis para instalação nesta plataforma, incluindo Linux Ubuntu e Debian além de *Windows Embedded Compact* e CE 6.0 [21].

A presença de SO Linux facilita muito as tarefas de comunicação, processamento e armazenamento dos dados. Possibilita grande flexibilidade na escolha de linguagens de programação, como *Phyton*, C, C++, Java e JavaScript. Entretanto, o uso de Linux para alguns tipos de aplicação em *Real Time* (Tempo Real), como aquisição de dados por exemplo, não é recomendada, seu *kernel* por padrão, não é pré-emptivo, isto é, após o processador iniciar a execução de um código do *kernel* ele não pode ser interrompido [20].

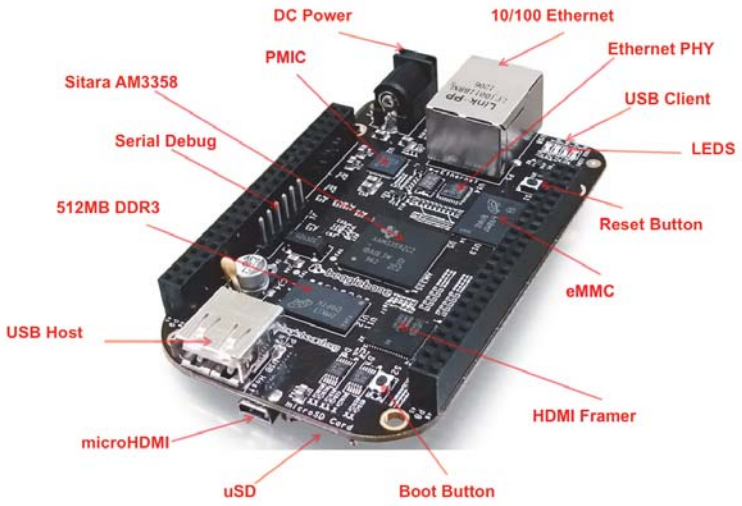


Figura 8. Beaglebone Black [21].

Uma possível solução para um sistema de aquisição em tempo real, é apresentada pelo Prof. Derek Molloy [20], sua implementação está baseada na utilização da Programable Real-time Unit Sub-system (PRUSS), presente no interior do μP principal AM3358. As unidades *PRUSS* são na verdade dois μC de 32-bits com arquitetura *RISC*, são executadas de forma independente do sistema operacional e possuem memória dedicada e *clock* próprio de 200MHz. Um diagrama esquemático da solução adota é apresentada na Figura 9.

Com esta solução é possível realizar as aquisições de forma independente do sistema operacional e transferi-las para uma área de memória, previamente determinada, acessível tanto pelo sistema operacional quanto pelas PRUSS. Como duas unidades *PRUSS* estão presentes no *Beaglebone Black* a utilização de dois conversores externos é possível, podendo assim atingir 16 canais AD

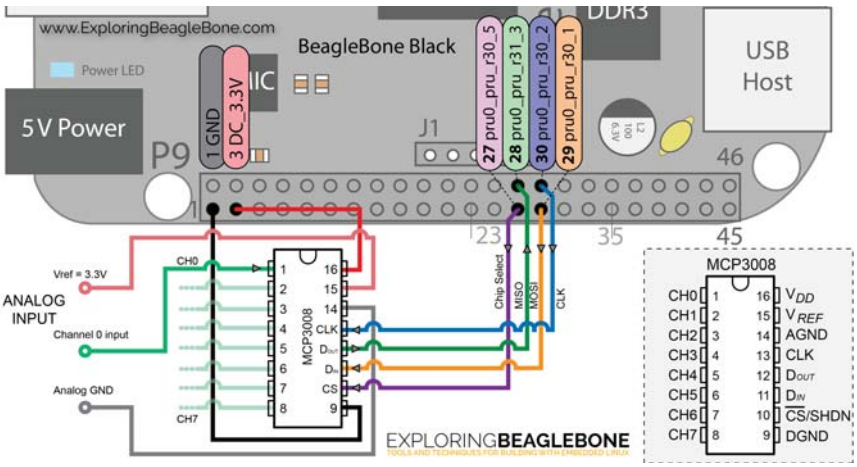


Figura 9. AD externo com *Beaglebone Black e PRUSS* [22].

sem impacto nenhum ao processamento do sistema operacional. A modificação do CI MCP3208 por outro, por exemplo ADS7883, pode elevar a resolução de conversão de 10 para 12 bits.

3.1.1.2 *Udoo Dual lite*

Assim como o *Beaglebone Black* a *Udoo* possui uma grande gama de SO's disponíveis para instalação, incluindo uma distribuição *Android*. Compartilha também as vantagens de possuir um SO embarcado. Entretanto, dentre os *SBCs* a *Udoo* possui o maior custo de aquisição, todavia é a única a possuir o número mínimo de conversores AD, 12 no total, de forma nativa. A presenta do módulo de comunicação *Wi-Fi* torna possível eliminar a conexão *ethernet* que utiliza cabos. Uma visão geral da *Udoo* é apresentada na Figura 10.

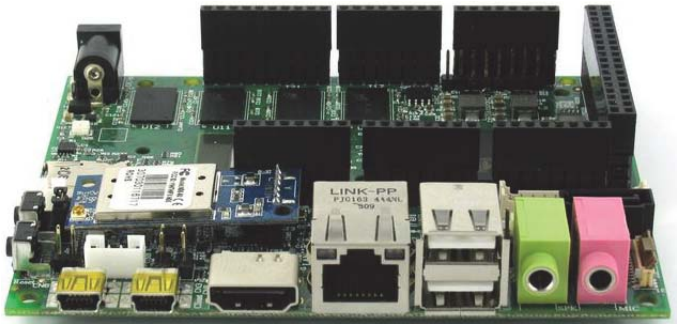


Figura 10. Udoo Dual Lite [23].

Assim como no caso do *Beaglebone Black*, a *Udoo* apresenta uma característica de *hardware* que torna o sistema de aquisição em tempo real possível, sem contudo impactar no sistema operacional. A *Udoo* é na verdade um sistema multiplataforma integrada em uma única placa, detalhe exposto na Figura 11.

Estão presentes na *Udoo* dois μCs da plataforma ARM, sendo o Freescale i.MX6Dual (ARM Cortex-A9), responsável pelo sistema operacional e o Atmel SAM3X8E (ARM Cortex-M3) que tem seus pinos dispostos de modo a ser compatível com qualquer periférico utilizado pelo Arduino Due.

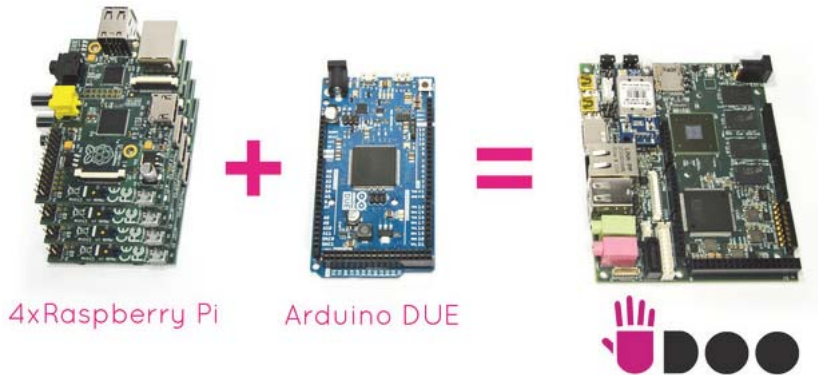


Figura 11. Característica multiplataforma da *Udoo Dual Lite* [23].

Os dois rodam de forma independente, com velocidades de *clock* diferentes, compartilhando alguns pinos de entrada e saída. A comunicação entre eles ocorre através de uma interface serial integrada [23].

Por se tratar de um sistema multiplataforma, é possível dividir a carga de processamento entre aquisição e tratamento de dados. Desta forma o SAM3X8E pode ser encarregado da aquisição, através dos canais ADs disponíveis, e o envio destes ao i.MX6Dual que realizará o tratamento e envio dos dados através da interface de rede.

3.1.2 *Mixed Signal Processor - MSP*

Também construídos a partir da arquitetura ARM eles incorporam o processamento de sinais analógicos e digitais em um único encapsulamento. *MSPs* não possuem SO embarcado [4], entretanto, são fornecidas pelo fabricante, bibliotecas de software com características de Sistema Operacional de Tempo Real (RTOS) . Tais bibliotecas possibilitam acesso aos periféricos de entrada e saída, criação de *tasks* e *threads* com um certo nível de abstração do *hardware* envolvido [17].

Adotando critério semelhante ao utilizado nos SBCs, a Tabela 2 foi elaborada listando algumas plataformas disponíveis no mercado e que integram o μC em um *kit* de avaliação.

Tabela 2. Lista de Plataformas MSP.

Plataforma	*Clock (MHz)*	*RAM (KB)*	ADs	*ethernet*	Tamanho (mm)	Custo (US$) [4]
TM4C1294	120	256	20	Sim	56×124	19,99
FRDM-64KF	120	256	4	Sim	81×53	35,00
NUCLEO-F401RE	100	128	16	Não	80×69	12,74
LPCXpresso4337	204	136	7	Não	123×59	26,25

O número de ADs supera o mínimo especificado apenas em duas plataformas, TM4C1294 da Texas Instruments[10] e NUCLEO-F401RE da STM, porém a segunda não possui porta *ethernet* integrada, problema que pode ser contornado utilizando algum tipo de conversor serial para *ethernet*. A desvantagem deste tipo de conversores é a limitação da velocidade de transmissão dos pacotes de

[4] Custo em junho 2017

dados, já que o canal serial opera a uma taxa máxima de 115.200 *bytes* por segundo (bps). As demais plataformas não atendem o mínimo número de canais AD e não serão discutidas.

3.1.2.1 ST NUCLEO-F401RE

O kit de desenvolvimento NUCLEO-F401RE, apresentado na Figura 12, é fabricado pela *ST Semiconductor* e dispõe de uma grande quantidade de periféricos integrados. Em destaque um bom número de conversores AD, 16 canais de 12bits no total.

Figura 12. Visão geral do Nucleo-F401RE [24].

Entretanto, não existe nenhuma interface de *ethernet* nativa neste *kit*, isso representa um ponto negativo importante no que diz respeito a utilização desta plataforma no Protegemed. Para contornar o problema de conexão, é possível utilizar um *shield* de comunicação, *ethernet* ou *wi-fi*, produzidos para Arduino™, já que este kit possui compatibilidade com alguns destes.

Dentre os pontos positivos, pode-se citar a compatibilidade com o ambiente de desenvolvimento *ARMmbed* utilizado atualmente. Isso poderia representar uma rápida migração do *firmware* atual para o novo, já que compartilham do mesmo nível de abstração de *hardware*.

3.1.2.2 *Texas Instruments TM4C1294XL*

O *kit* de desenvolvimento TM4C1294 (EK-TM4C1294XL) é uma plataforma de avaliação de baixo custo para μCs baseados no ARM Cortex-TM4. O TM4C utiliza o μC TM4C1294NCPDT que possui *ethernet* MAC e PHY 10/100 embutidos, USB 2.0, módulo de hibernação, módulo *PWM* e uma

grande gama de periféricos para comunicação serial (I2C, UART, SPI, CAN). O TM4C conta ainda com dois botões e quatro LEDs de usuário, botões de *reset* e de saída de hibernação.

Os pinos de interface do μC estão disponíveis em quatro conectores dispostos nas bordas da placa, estes conectores permitem a instalação de módulos de hardware adicionais chamados pelo fabricante de *Booster Packs*. Alguns exemplos de *Booster Packs* são módulos de comunicação *Wi-Fi* e *Drivers* de motor de passo. Além disso, é possível soldar conectores na sua borda inferior o que facilita a utilização do TM4C em *proto boards* para a rápida montagem de protótipos. Na Figura 13 é apresentada uma visão geral da placa e uma listagem dos sinais disponíveis no conector inferior.

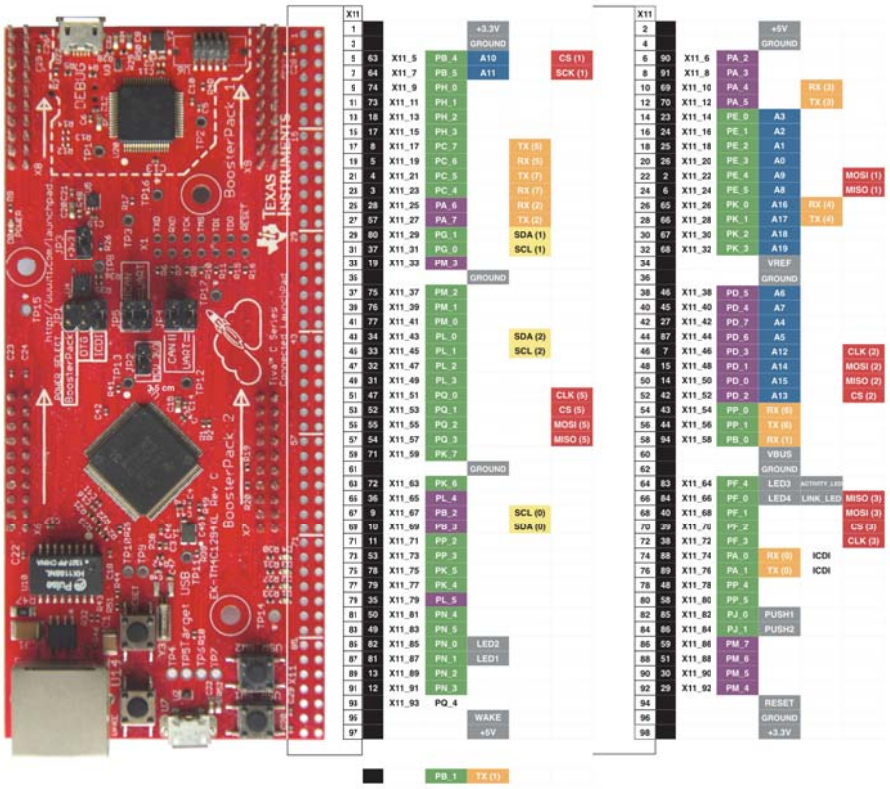


Figura 13. Pinos de entrada e saída disponíveis na *TM4C1294XL* [25].

O fato do TM4C possuir porta *ethernet* integrada, facilita sua conexão a internet e por consequência sua utilização em projetos que envolvem internet da coisas (IOT). Através da porta *ethernet* também é possível a atualização do *firmware* de forma remota, algo muito desejado no caso de sua aplicação no Protegemed. O número de canais AD presentes nesta placa, 20 no total, são suficientes para a versão atual além de comportar futuras implementações de aquisição de dados.

A programação do software neste μC pode ser realizada tanto em C quanto em C++, diversos compiladores estão disponíveis para a criação do código, inclusive um serviço baseado em nuvem chamado *CCS Cloud* [26]. A comunidade de desenvolvedores dispõem de *forums* para discussão e troca de informações durante a resolução de problemas. A *TM4C* dispõe ainda de uma interface de *Debug* que possibilita por exemplo, a execução passo a passo, o monitoramento de variáveis e a inserção de *breakpoints* no *firmware* que está sendo testado.

3.2 DEFINIÇÃO DA PLATAFORMA

Dentre os SBCs avaliados, a Udoo Dual Lite foi selecionada, visto que é a única que atende as especificações mínimas de forma nativa, sendo ainda possível instalar expansões para acomodar novos recursos de *hardware*. Seu sistema multiplataforma, com dois μCs, permite a operação de aquisição de dados de forma independente das tarefas de processamento e comunicação. A utilização de sistema operacional Linux facilita a comunicação, gerenciamento, armazenamento de dados. A presença de uma interface *Wi-Fi* possibilita a eliminação da rede cabeada utilizada atualmente. A grande quantidade de memória *RAM* e o clock de 1GHz são pontos favoráveis na *Udoo*, por fim o μC com dois núcleos de processamento pode, no futuro, ser alvo de aplicações que utilizam o conceito de programação concorrente.

A utilização de um SBC pode representar uma grande mudança, na velocidade e no desenvolvimento, de novos recursos para o Protegmed. Entretanto, a necessidade de componentes externos e o alto custo de aquisição da *Udoo*, são fatores que devem ser levados em conta.

A utilização do MSP TM4C1294 atende as especificações inicias e deixa boa margem para futuras atualizações, a presença da porta *ethernet* nativa, atualização do *firmware* através da rede e o grande número de canais AD, que superam com boa margem as especificações iniciais, qualificam esta placa como possível nova plataforma para o sistema Protegemed. A programação concorrente, neste caso, pode ser explorada com a utilização do controlador de *Direct Memory Access* (DMA - em português Controlador de acesso direto a memória) presente na *TM4C*.

Na Tabela 3 são comparadas algumas características da plataforma atual com as plataformas selecionadas.

Tabela 3. Comparativo entre plataforma atual e futura.

Plataforma	*Clock*	*RAM*	ADs	*ethernet*	Tamanho (mm)	Custo (US$) [5]
mbed LPC1768	96 MHz	32KB [6]	6	Sim	54×26	49,00
TM4C1294XL	120 MHz	256KB	18	Sim	56×124	19,99
Udoo Dual	1GHz [7]	1GB	12	Sim	110×85	115,00

Os custos apresentados para cada placa não incluem impostos e outras possíveis despesas, como envio e tarifas extras.

3.2.1 Conversor A/D - *TM4C* e *Udoo*

A aquisição de dados é uma etapa fundamental no Protegmed. A manipulação das amostras capturadas pelo conversor A/D é realizada pelo μC. Por esse motivo, a comparação correta entre *TM4C* e *Udoo* deve ser direcionada ao μC SAM3X8 (ARM Cortex-M3) e não iMx6Dual (ARM Cortex-A9). Para que seja possível inserir, e manipular, mais canais analógicos, é necessário comparar as

[5] Custo em junho de 2017.
[6] Expansível até 64KB [3].
[7] SAM3X8 roda em 84MHz[23]

especificações do módulo conversor A/D de cada plataforma[27][28]. Na tabela 4 estão listadas as especificações mais relevantes ao projeto.

Tabela 4. Comparativo do conversor A/D entre *TM4C* e *Udoo*

Descrição	*TM4C*	*Udoo*
Resolução (bits)	12	12
Número de canais	20	12 [8]
Número de conversores	2	1
Taxa de conversão (MSPS)[9]	2	1
Acesso por DMA	Sim	Sim
Hardware Sample Averaging hardware	Sim	Não
Seleção de canal	*Sample Sequencer*	Multiplexador

Normalmente, os μCs possuem somente um conversor A/D conectado a vários canais. A seleção do canal a ser utilizado é realizada através de um multiplexador programável por *software*. Desta forma, para cada conversão, é necessário antes selecionar o canal que será utilizado para só então iniciar a conversão, esta topologia é utilizada pelo *mbed* e pelo μC da *Udoo*[27].

Na TM4C estão presentes dois conversores A/D , estes conversores compartilham o acesso a todos os canais disponíveis, entretanto, cada conversor tem seu próprio controlador. Isto significa, que é possível, utilizar os dois conversores para um mesmo canal, elevando assim a taxa de conversão de 2MSPS para 4MSPS. A seleção dos canais a serem utilizados, é realizada previamente através da programação dos *sample sequencers*. Os *sample sequencers* são responsáveis pelo controle de amostragem e captura de dados. Estes, armazenam o resultado da conversão em memórias do tipo FIFO [28]. Desta forma, o *software* não necessita acessar o multiplexador para selecionar o canal utilizado, este processo é realizado por *hardware* pelo *sample sequencer*.

Outra característica relevante na *TM4C* é a presença do *Hardware Sample Averaging*. Sua utilização, permite que cada amostra armazenada no *sample sequencer*, seja o resultado da média de até 64 amostras. Na figura 14 é apresentado um exemplo de utilização do *Hardware Sample Averaging*.

Neste exemplo, o valor do *Hardware Sample Averaging* é igual a 4. Um sample sequencer, com profundidade de memória de 4×32bits, está programado para gerar uma interrupção quando a segunda posição for preenchida. Assim, a primeira posição contém o resultado da média das quatro primeiras amostras e a segunda posição, a média das quatro amostras seguintes.

O custo da utilização do *Hardware Sample Averaging* é o decréscimo proporcional na taxa de conversão, ou seja, para uma média de 16 amostras, a taxa de conversão é reduzida 16 vezes.

[8] SAM3X8 possui 15 canais, entretanto, no projeto da Udoo apenas 12 estão disponíveis para utilização [29]

[9] Mega Sample per Second (Milhões de amostra por segundo)

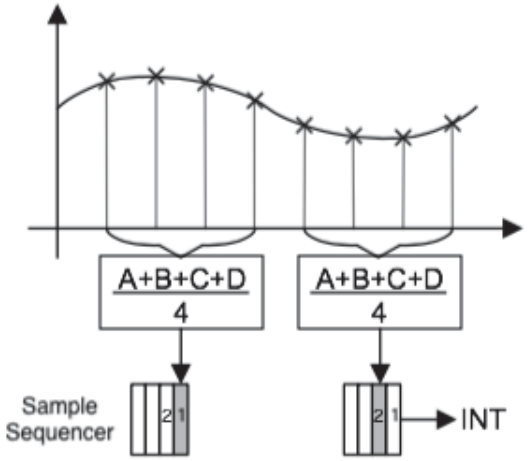


Figura 14. Exemplo de utilização do *Hardware Sample Averaging*, extraído e modificado de [28]

.

3.2.2 Testes com a *UDOO*

O teste realizado consiste em implementar um sistema de aquisição de dados utilizando o μC SAM3x8E, processá-los com o μC i.MX6, e disponibilizar os resultados em uma página HTML . Na Figura 15 é apresentado o diagrama de blocos do teste proposto.

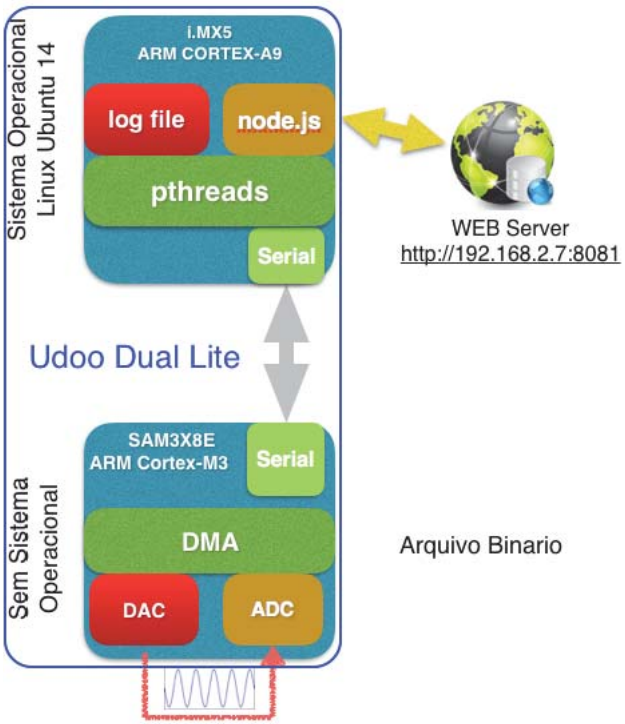


Figura 15. Diagrama de blocos do teste proposto.

Os objetivos do teste são:

- Validar a aquisição de dados no sistema multiplataforma.
- Utilizar os softwares de programação oferecidos pela *Udoo*.
- Utilizar a transferência de dados por DMA.
- Verificar potenciais problemas durante seu desenvolvimento.

Para este teste será utilizado apenas um canal analógico. O vetor de amostra deverá conter 256 valores discretos de uma FO de 60Hz, isso resulta em uma frequência de amostragem de $f_s = 256 \cdot 60Hz = 15360Hz$. Entretanto, para o envio dos dados através da interface serial, este vetor será reduzido utilizando a média aritmética para cada quatro amostras, resultando em um tamanho final de $256/4 = 64$ amostras. A resolução do conversor A/D é de 12bits e a tensão de referência utilizada é de 3,3V. Neste caso o erro de quantização obtido através da equação 17 é calculado pela equação 19.

$$Q_{err} = \frac{3,3}{2 \cdot 2^{12}} \approx 402\mu V \tag{19}$$

3.2.2.1 Formas de Onda utilizadas

Para validar os resultados, foram criadas FO conhecidas com auxílio do *software Matlab*, tais FOs serão recriadas pelo μC da *Udoo* através do conversor digital analógico (DA) . No total foram utilizadas três FOs, $y1$, $y2$ e $y3$, compostas cada por um conjunto de 255 valores discretos. Para isso, foram utilizadas as equações 20, 21 e 22 respectivamente.

$$y1(t) = 0.8 \cdot sin(2\pi \cdot 60 \cdot t); \tag{20}$$

$$y2(t) = 0.5 \cdot sin(2\pi \cdot 60 \cdot t) + 0.3 \cdot sin(2pi \cdot 120 \cdot t) + 0.05 \cdot sin(2pi \cdot 180 \cdot t); \tag{21}$$

$$y3(t) = 0.5 \cdot sin(2pi \cdot 60 \cdot t) + 0.1 \cdot sin(2pi \cdot 720 \cdot t); \tag{22}$$

Os sinais de teste foram normalizados para inteiros positivos, variando entre 0 e 4095 (12 bits de resolução), através da equação $y_k[n] = ceil(y_k[n] \cdot 2048) + 2047$. Os resultados foram armazenados em forma de tabela, estes servem como referência para recriar os sinais no espaço de tempo contínuo através do conversor DA.

Essencialmente, estes sinais são a representação de uma senoide pura, $y1$, uma senoide com duas componentes harmônicas (2ª e 3ª harmônicas) em $y2$, e uma senoide com um único componente harmônico de 3ª ordem. A inclusão das harmônicas foi realizada para validar os cálculos realizados nos sinais após a amostragem. Um dos tratamentos dado ao sinal é aplicação da FFT, onde se espera a reprodução dos mesmos resultados calculados com o *Matlab*.

3.2.2.2 Estrutura do *firwmare* de teste - *Udoo*

Quando se deseja gerar um sinal contínuo a partir de dados discretos, contidos em tabelas por exemplo, uma das possibilidades é a criação de um *loop* que deve percorrer todas as posições desta tabela e transferir o valor encontrado para o conversor DA. Quando a última posição for alcançada o *loop* deve ser reiniciado e processo continua indefinidamente. O problema com esta abordagem é que, necessariamente, todas as operações de leitura e escrita devem ser realizadas explicitamente pelo *software* de controle, isto é, todas as informações são transferidas da tabela para a memória da CPU e da CPU para o DAC [4].

Situação semelhante ocorre quando é necessária a amostragem de um sinal contínuo através do conversor A/D. Para que o sinal amostrado seja consistente, é mandatório garantir a leitura do conversor A/D em intervalos de tempo pré-definidos, sob pena da ocorrência de erros na leitura e má qualidade nos dados amostrados. Uma abordagem para a solução deste problema é a utilização de um *timer* que gera uma interrupção a cada período de amostragem, então, a cada interrupção uma amostra deve ser lida do conversor A/D e armazenada em um *buffer* para processamento posterior.

Para transferir dados entre duas áreas de memória, sem a necessidade de intervenção da CPU, pode-se utilizar um recurso conhecido com acesso direto à memória (DMA) , este permite a transferência entre áreas de memória sem a intervenção da CPU. Desta forma, pode-se programar o controlador de DMA para realizar a geração do sinal, transferindo dados da tabela (contida na memória) para o conversor D/A. Uma intervenção da CPU só é necessária quando o final da tabela é alcançado (fim da transferência), neste momento, a CPU é responsável por reinicializar o índice de leitura e reiniciar o controlador de DMA. Testes mostram que mesmo em pequenos blocos de dados é mais vantajoso utilizar a transferência de dados através de DMA do que por *software* [17].

No caso desta aplicação o recurso de DMA foi utilizado para ambas as situações, geração do sinal de teste e amostragem do sinal gerado. A utilização do controlador de DMA é relativamente difícil de ser implementada, muito pela necessidade do conhecimento de registradores específicos, descritos em folhas de dados que facilmente superam 1000 páginas.

3.2.2.3 *Softwares* e linguagens utilizados

Como a *Udoo* é um sistema multiplataforma, são necessárias duas abordagens distintas no que diz respeito a programação. A primeira, para o caso do sistema de aquisição implementado no μC SAM3X8, foi realizada utilizando o software de programação do Arduino [30]. O FW se resume a um grande *loop* principal seguido por funções de interrupção específicas, sem contar com o suporte de um SO. No caso deste FW, o loop principal fica sempre aguardando um novo dado ser disponibilizado pela interrupção do DMA, quando isso ocorre, os dados são enviados através da interface serial para o μC i.MX6.

Ao contrário do SAM38XE o i.MX6 possui um SO embarcado. Para o teste foi utilizada a distribuição Linux Ubuntu 14. A presença de um SO simplifica o desenvolvimento de aplicações de

alto nível, como por exemplo *web servers* e banco de dados [20]. A tarefa atribuída ao i.MX6 é a recepção e o tratamento dos sinais amostrados pelo SAM3X8E.

O problema do produtor/consumidor descreve um exemplo clássico de sincronização entre processos [31]. Neste caso o papel do produtor é desempenhado pelo SAM3X8E, que envia mensagens a cada $16.67ms$, já o consumidor, i.MX6 recebe, armazena em um *buffer* intermediário, e disponibiliza os dados para manipulação.

O software de comunicação implementado, escrito totalmente em C, utiliza a biblioteca *pthread* em conjunto semáforos para o tratamento da seção crítica e controle do *buffer*. Além disso, este software analisa os dados recebidos através do cálculo do valor rms e da FFT. Então, dois arquivos são gerados, um que contém os dados brutos e a identificação da amostra recebida, e outro que contém um *log* de todas as mensagens recebidas. O arquivo de *log* é composto pela data e hora em que foi recebido, a identificação da amostra, o valor rms normalizado em Volts e as amplitudes da frequência que ultrapassam um valor mínimo pré definido (Ex.: Amplitude $>$ 0.09V).

O arquivo contendo os dados brutos, é utilizado pelo servidor html, escrito em *node.js*, para visualização gráfica no navegador. A sequência de funcionamento se dá como segue, o servidor atende a requisições através do endereço 192.128.2.7:8081, quando uma nova amostra é solicitada através do módulo *socket.io* o servidor executa o *software* de comunicação, lê os valores brutos recebidos e os transmite ao navegador também utilizando *socket.io*. Um *script* na página HTML é o responsável pelo normalização e apresentação na forma gráfica dos valores lidos.

3.2.2.4 Resultados obtidos com *Udoo*

A primeira observação se refere a qualidade do sinal gerado. Com o auxilio do osciloscópio, foi possível verificar um sinal sem distorções ou *dead band* (descontinuidade nas transições de posições da tabela de valores). Na Figura 16 são mostradas 4 imagens para ilustrar esta afirmação, na parte superior estão contidas imagens do sinal $y1(t)$ e $y2(t)$, ambos capturados pelo osciloscópio. Processo semelhante foi realizado através da interface *web*, visualmente é possível observar uma boa similaridade entre as formas de onda do osciloscópio e as mostradas no navegador.

Outra análise pode ser realizada observando as componentes harmônicas presentes no sinal testado, como tais componentes são conhecidas, pois foram determinadas pelas equações 20, 21 e 22. É possível confrontar os valores calculados com os resultados obtidos no *Matlab*, o resultado pode ser visto na Figura 17.

Os valores correspondentes ao cálculo de rms não tiveram variação, Udoo $0.456V$ e Matlab $0.456V$, no caso da análise de componentes harmônicos, as maiores amplitudes foram atingidas nas mesmas frequências (como esperado) com pouca variação no valor calculado, isso pode ter ocorrido devido aos arredondamentos aplicados nos diferentes tamanhos de variáveis utilizadas, na Udoo 32 bits e no Matlab 64 bits.

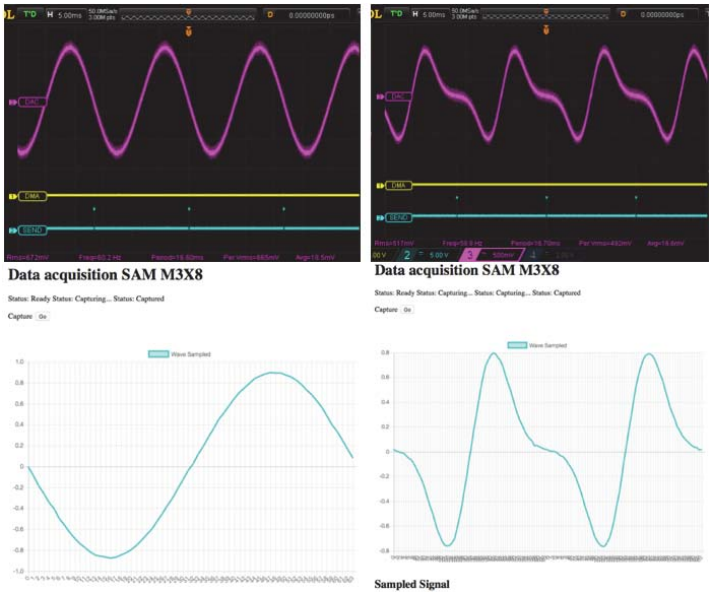


Figura 16. Formas de onda. Osciloscópio na parte superior e navegador *web* na parte inferior.

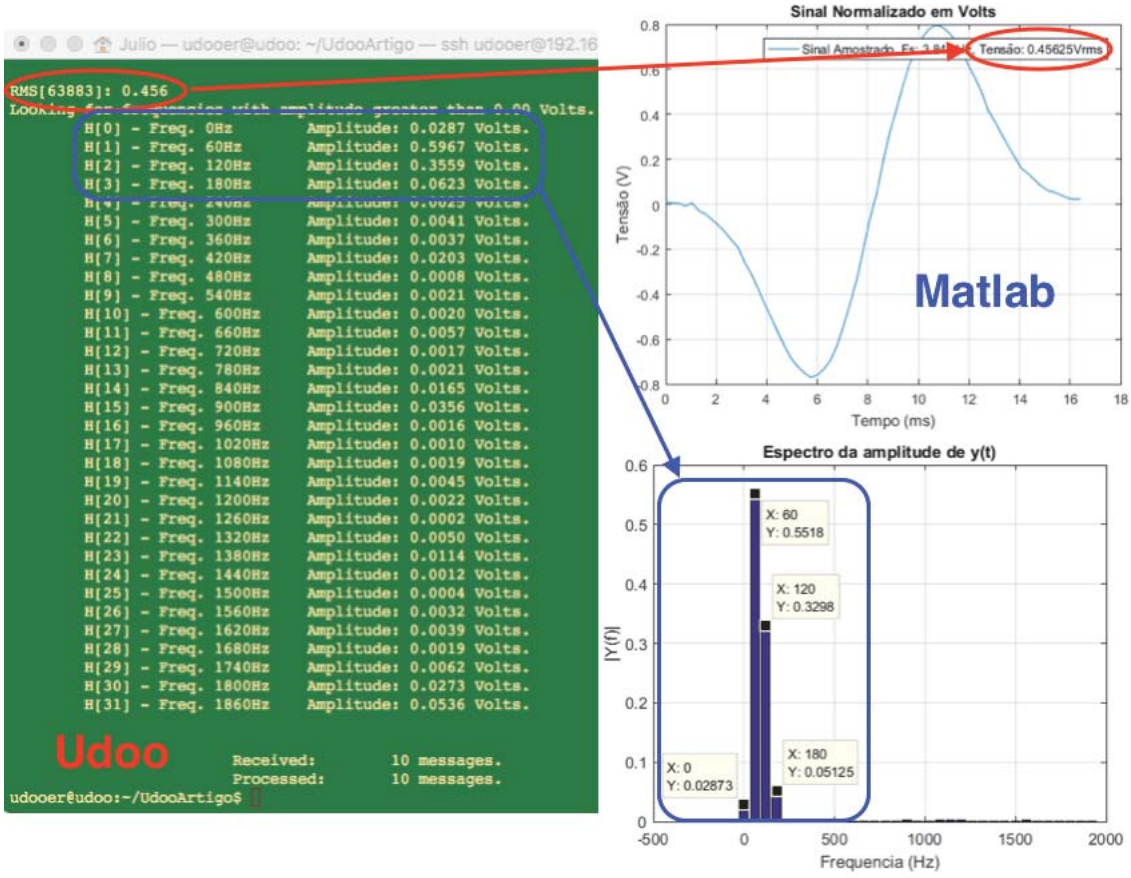


Figura 17. Comparação dos os valores obtido durante a aquisição com a Udoo e calculados com o Matlab.

3.2.2.5 Discussão dos resultados - *Udoo*

Apesar de se utilizar o ambiente de programação do Arduino, famoso por sua simplicidade, a programação do DMA não tem qualquer suporte de bibliotecas prontas, esta deve ser realizada

utilizando registradores específicos do SAM3X8. Outro ponto negativo deste ambiente de desenvolvimento é ausência de Debug. Não é possível, por exemplo, inserir *breakpoints* (pontos de parada) no FW ou executá-lo passo a passo.

O ambiente de desenvolvimento com SO embarcado se mostrou de fácil utilização, muito devido ao nível de abstração que o SO proporciona. A utilização de linguagens de alto nível, como o *node.js*, simplificou muito o desenvolvimento do *software* de comunicação.

A aquisição e dados foi realizada sem erros e a comunicação entre processadores se mostrou confiável. Os testes de comunicação realizados, utilizando a interface *wi-fi*, mostraram instabilidade do sinal em distancias maiores de 4 metros. Quando utilizado a interface *ethernet* não ocorreram problemas.

3.2.3 Testes com a *TM4C*

Na *TM4C* o teste proposto é a amostragem simultânea de 16 canais analógicos. O sinal de teste, obtido com um gerador de funções, é aplicado aos canais 0 e 9 (AIN0 e AIN9), os demais canais não estão conectados a nenhum sinal específico. Na figura 18 é mostrada o diagrama de blocos do teste proposto.

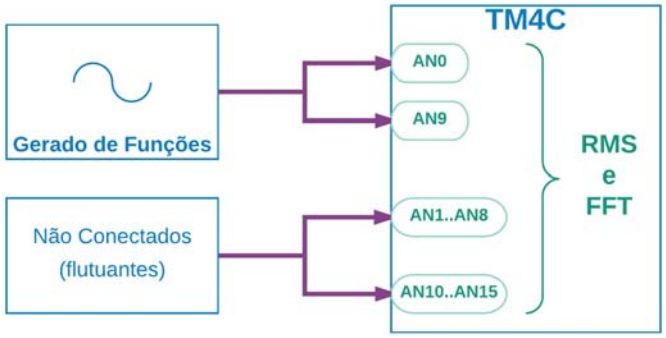


Figura 18. Diagrama blocos do teste realizado com a TM4C

Para todos os canais amostrados são calculados o valor rms e a FFT . Os objetivos deste teste são:

- Utilizar o ambiente de desenvolvimento Code Composer Studio (CCS).
- Medir o tempo necessário para realizar a amostragem e os cálculos.
- Validar os resultados com *Matlab*.

Assim como no teste com a *Udoo*, o vetor de amostra deverá conter 256 valores discretos de uma FO de 60Hz, resultando em uma frequência de amostragem de $f_s = 256 \cdot 60Hz = 15360Hz$. O número total de vetores é igual ao número de canais amostrados, quinze no total. A resolução do

conversor A/D é de 12bits e a tensão de referência utilizada é de 3,3V. O erro de quantização é o mesmo obtido pela equação 19, aproximadamente $402\mu V$.

3.2.3.1 Ambiente de desenvolvimento

O *Code Composer Studio* (CCS) é um ambiente de desenvolvimento integrado (IDE) que dá suporte aos μCs fabricados pela *Texas Instruments* (TI) utilizando compiladores otimizados para as linguagens C e C++ [32]. Sua interface é inteiramente baseada na conhecida IDE Eclipse [33], combinada com multiplas ferramentas de *debug* fornecidas pela TI.

O ambiente de edição suporta recursos que exigem uma visão global de todos os arquivos declarados no projeto. Com isso, é possível encontrar todos os locais e arquivos em que uma determinada função é chamada dentro do projeto. Para que isso seja possível, o CCS mantém um índice contendo informações semânticas do projeto [34].

A partir de sua IDE, é possível acessar o *link* para o *Resource Explorer* (vasta biblioteca online mantida pela TI) exibido de forma integrada ao ambiente de edição. Uma vez aberto, pode-se navegar por uma grande quantidade de códigos de exemplo organizados por plataforma de desenvolvimento, importá-los para o projeto atual, pesquisar *datasheets* e notas de aplicação, além de instalar novos compiladores e *softwares* de apoio.

Talvez o melhor recurso do CCS seja o ambiente de *debug* apresentado na figura 19. Através deste, é possível inserir *breakpoints*, rodar o FW passo a passo, inspecionar e modificar variáveis, exibir um conjunto de dados graficamente, calcular a FFT de um sinal amostrado e ainda exportar para um arquivo utilizando o formato *comma-separated values* (csv) .

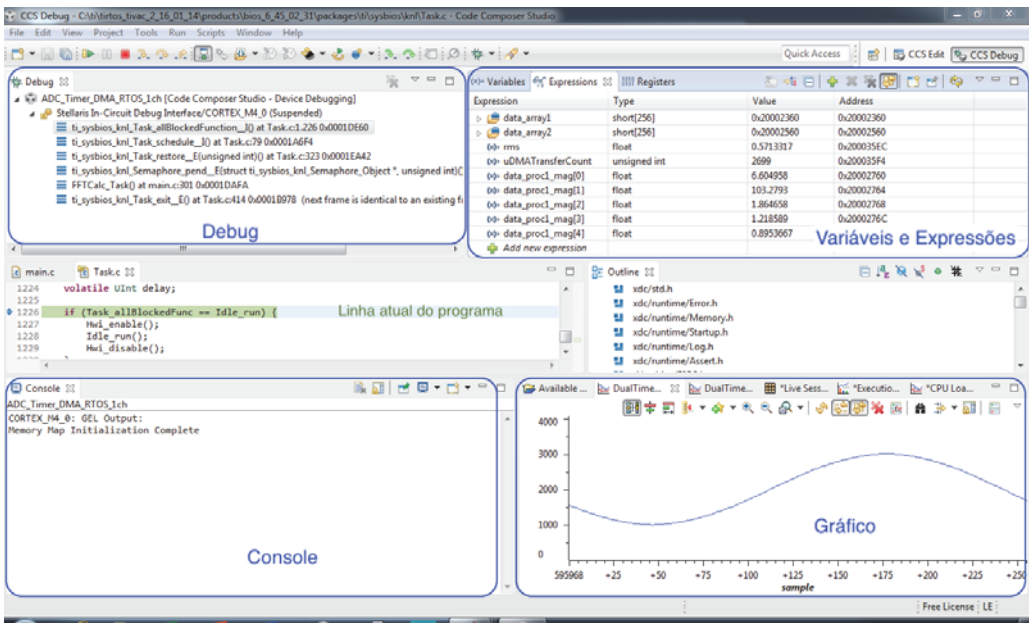


Figura 19. Visão geral do ambiente de debug do CCS

Os recursos de *debug* permitem analisar o resultado do processo de aquisição de dados, forma de onda e FFT , sem a necessidade de um *software* A/D adicional. No campo de variáveis

e expressões pode-se inserir funções matemáticas básicas na variável observada. Desta forma, é possível ajustar o ganho do canal analógico enquanto o FW está em execução.

3.2.3.2 Estrutura do *firwmare* de teste - *TM4C*

O FW proposto para o teste está baseado na utilização do DMA, dos *sample sequencers* e do *Hardware Sample Averaging* (recursos descritos na seção 3.2.1). Como a *TM4C* possui dois conversores A/D , com taxa de conversão de 2MSPS cada, dos 16 canais a serem amostrados, 8 estão conectados no primeiro conversor (AD0) e outros 8 no segundo (AD1).

O valor de *Hardware Sample Averaging* foi definido em 16 vezes, ou seja, cada amostra é o resultado da média de 16 amostras. Com a utilização deste, a f_s é reduzida em 16 vezes, $2\text{MSPS}/16 = 125\text{KSPS}$, como são utilizados 8 canais, a f_s de cada canal é de no máximo $125\text{KSPS}/8 = 15625\text{SPS}$ ou 15625Hz. Este valor atende ao especificado para o teste $f_s = 15360\text{Hz}$ ($15625\text{Hz} > 15360\text{HZ}$).

As amostras são lidas de forma contínua, ou seja, nenhum ciclo do sinal deve ser ignorado, para isso, uma possível solução é a utilização de um *buffer* circular [35]. Dentre os métodos de funcionamento do controlador de DMA, um é denominado *ping/pong*, neste método, são indicadas duas variáveis utilizadas como *buffer* circular, enquanto uma está sendo utilizada pelo DMA, a outra está disponível ao FW. O DMA pode transferir dados em incrementos programáveis de 1 à 1024 passos binários. Na prática isso significa que cada transferência, do conversor A/D para o DMA por exemplo, pode conter no máximo 1024 amostras. Na figura 20 é mostrado o diagrama para o sistema de aquisição.

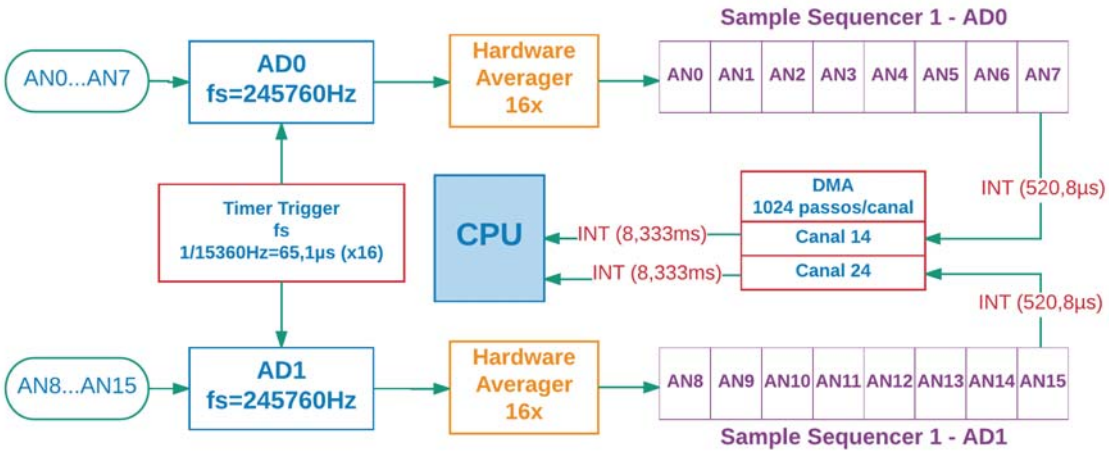


Figura 20. Diagrama de blocos aquisição de dados utilizados na TM4C

.

A interrupção do DMA para o conversor A/D 0 (também para o conversor A/D 1), contém 1024 amostras dos canais AN0, AN1,. . . ,AN7 mapeados no *sample sequencer 1*, resultando em $1024/8 = 127$ amostras por canal. O vetor de amostras a ser processado deve conter 256 valores. A FO de teste

possui frequência de 60Hz, de acordo com a equação 3 o período pode ser calculado pela equação 23.

$$T = \frac{1}{60} = 16,666mS \tag{23}$$

Como são transferidos apenas 127 amostras por canal em cada interrupção do DMA, são necessárias duas interrupções, por conversor A/D , para que sejam transferidas 256 amostras. Desta forma, pode-se afirmar que o DMA gera interrupções a cada 8,333mS e um bloco completo de novos dados a cada 16,666ms.

As amostras do *sample sequencer* são lidas pelo DMA e armazenadas no primeiro *buffer* circular. Em cada interrupção do DMA, os dados contidos no *buffer* circular, devem ser separados e reagrupados de forma a representarem as amostras de cada canal em um vetor $V_{ANx}[1 \times 256]$ $(x = 0, 1, 2, \ldots, 15)$. Para cada vetor V_{ANx} o valor rms é calculado seguido da FFT .

3.2.3.3 Resultados obtidos com *TM4C*

Para investigar o intervalo de tempo em cada interrupção do DMA, cálculo do valor rms e da FFT , é utilizado um método de *debug* baseado na utilização de E/S e de um osciloscópio[16]. Na figura 21 é apresentado o resultado dos períodos de execução para o teste realizado.

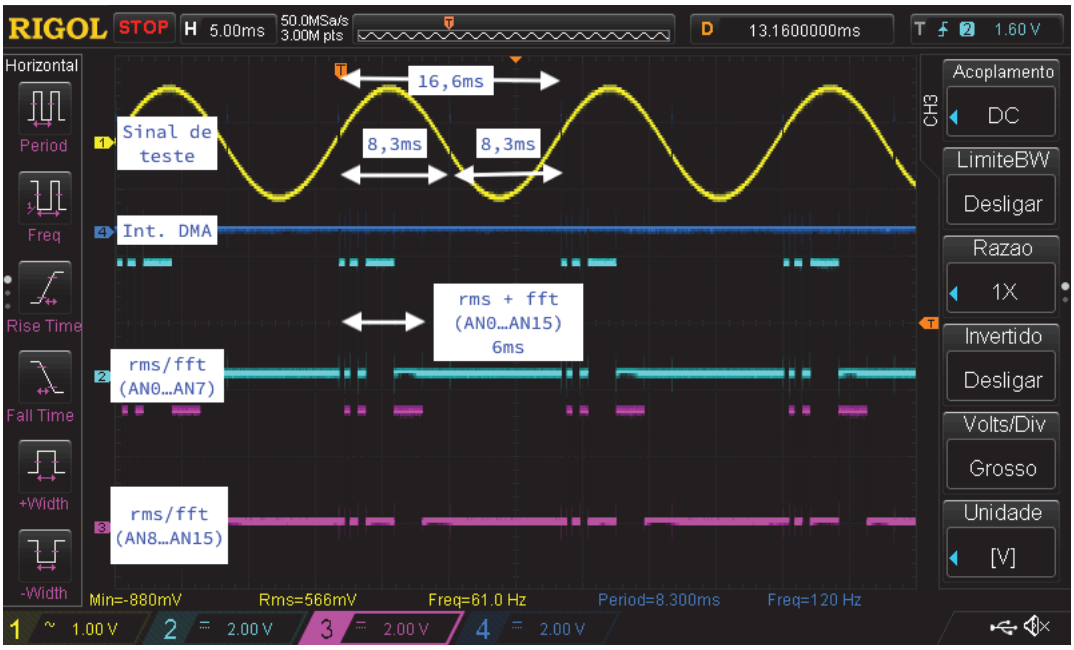


Figura 21. Tela do osciloscópio mostrando o debug de *hardware* utilizando os pinos de entrada e saída para verificar os tempos de execução de cada *task*

Neste caso, são escolhidos pinos de E/S configurados como saída, estes pinos são então setados para nível lógico baixo antes do inicio do *loop* principal do FW. No início de cada função, um pino especifico é setado em nível lógico alto, ao final desta função, este é novamente setado em nível

lógico baixo. Com o auxilio de um osciloscópio, é realizada a análise do intervalo de tempo em que este pino ficou em nível lógico alto, ou seja, o tempo em que a função ficou em execução.

A interrupção do DMA (Int. DMA), ocorre a cada 8,3ms com duração entre 10 e 15μs (não visível na figura 21). Quando duas interrupções ocorrem, são realizados os cálculos de rms e FFT para os canais AN0 até AN15. O tempo total de 6ms, inclui a transferência de valores do *buffer* circular para os vetores de processamento.

Com o auxilio do ambiente de *debug* do CCS, foram exportados o resultados das amostras do canal analógico AN0 para análise com o Matlab. Na figura 22 é apresentado o resultado desta análise.

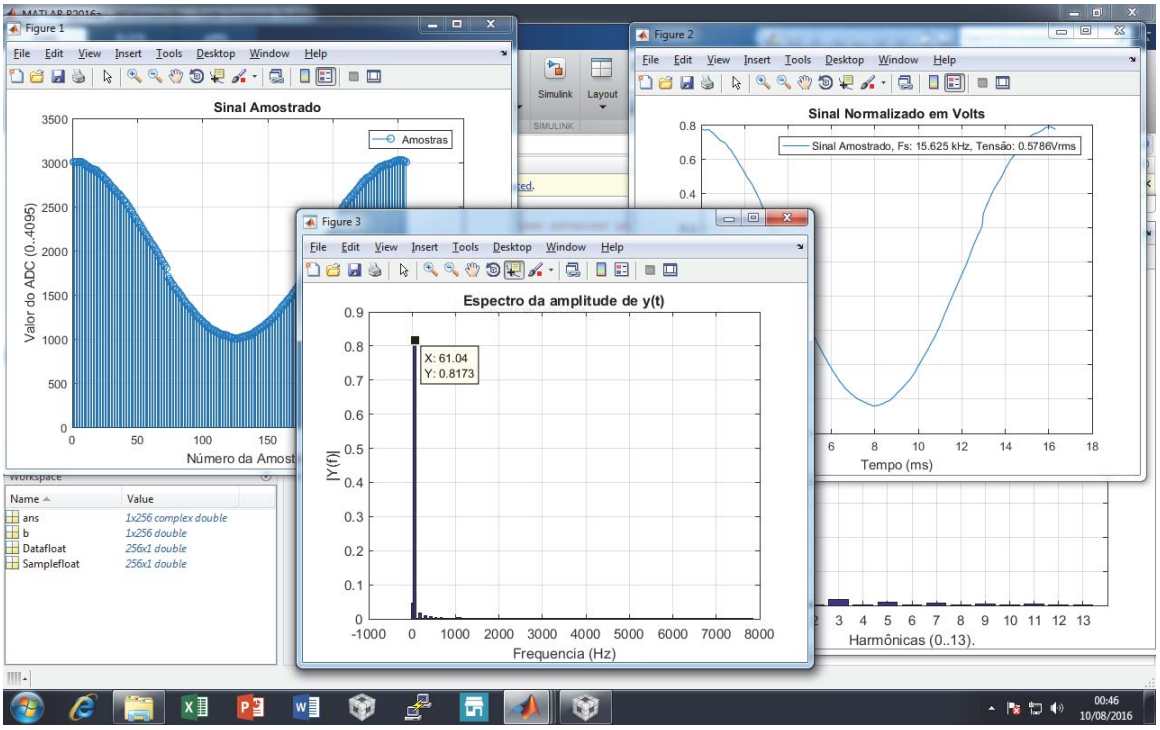


Figura 22. Análise de amostras canal AN0 com o Matlab

A tensão rms, calculada com as amostras exportadas, foi de 0,5786V, na figura 21, o valor lido pelo osciloscópio[10] foi de 0,566V, isso representa um erro de $(1-(0,566/0,5786))\cdot 100 = 2,17\%$. Como esperado, na análise de harmônicas através da FFT , as amostras tiveram sua maior amplitude na frequência de 60Hz.

3.2.3.4 Discussão dos resultados - *TM4C*

A utilização do ambiente de desenvolvimento do CCS se mostrou fácil e intuitiva, os recursos de *debug* facilitam muito a depuração do FW e a análise dos valores amostrados.

Quanto aos tempos de execução, o uso do DMA possibilita a realização da tarefa de aquisição com um mínimo de interferência da CPU. O tempo total necessário para realizar o cálculo de rms e FFT

[10]Não calibrado pela Rede Brasileira de Calibração (RBC)

foi de 6ms, estes cálculos são realizados a cada 16,6ms, isso significa que restam aproximadamente 10ms para desempenhar tarefas de análise ou comunicação.

Os resultados calculados com o Matlab, baseado nas amostras importadas do CCS, tiveram um erro próximo de 2%, como o ambiente de teste foi montado em *proto-board*, problemas de conexão ou ruído de sinal podem ter ocorrido. O erro de quantização do A/D pode também ter colaborado para este resultado, entretanto, um teste em um ambiente mais A/D equado poderá reduzir o erro obtido atualmente.

3.2.4 Avaliação dos Testes

Para auxiliar na definição da escolha da nova plataforma, parâmetros baseados no FW atual (tarefa de aquisição, análise e comunicação), no custo de aquisição e no ambiente de desenvolvimento foram criados. Cada um destes parâmetros recebeu peso igual, sendo o total igual a 100. Na tabela 5 é apresentado o resultado desta comparação.

Tabela 5. Comparativo final Udoo e TM4C

Parâmetro	Peso	*TM4C*	*Udoo*
Aquisição de dados	20	20	10
Análise e cálculos	20	18	20
Comunicação	20	10	20
Custo	20	20	5
Ambiente de Desenvolvimento	20	15	15
Total	100	83	70

Na tarefa de aquisição de dados, a *TM4C* tem ampla vantagem, a presença do *Hardware Sample Averaging* e dos *sample sequencers*, diminuem a complexidade e melhoram a qualidade das amostras na aquisição de dados. Além disso, a *TM4C* possui 20 canais analógicos, frente aos 12 da *Udoo*, contanto ainda com 2 conversores A/D que tem o dobro da taxa de amostragem do conversor presente na *Udoo*.

Para a tarefa de análise e cálculos, a *Udoo* conta com uma leve vantagem, muito pela possibilidade de realização dos cálculos utilizando o μC i.MX6 (OS Linux embarcado). Entretanto, a *TM4C* obteve um bom desempenho, precisando de apenas 6ms para calcular o valor rms e FFT de 16 canais analógicos.

A *Udoo* possui conexão *wi-fi* e *ethernet*, enquanto que a *TM4C* apenas *ethernet*, por essa razão, esta obteve maior pontuação neste parâmetro.

O custo de aquisição é um ponto muito negativo na *Udoo*, este é duas vezes maior que o custo do *mbed*, plataforma utilizada atualmente. Com relação a *TM4C*, o custo de $19,99 se refere a uma plataforma de avaliação, não permitida para uso comercial. Por esse motivo, é necessário o desenvolvimento de um projeto próprio utilizando o μC TM4C1294ncpdt, o custo deste μC é de $14,93 [36]. Ainda não há orçamentos para a fabricação desta placa, todos os testes foram realizados com a plataforma de avaliação (TM4C1294XL).

O ambiente de desenvolvimento é excelente em ambas as plataforma. Na *TM4C* o destaque fica no ambiente de *debug*, na *Udoo* a possibilidade da utilização de *softwares* de alto nível como *java* e *javascript*.

Tendo como base os resultados comentados, optou-se pela utilização da *TM4C* como nova plataforma de *hardware* do Protegemed.

4. IMPLEMENTAÇÃO DE *FIRMWARE* NA *TM4C*

Neste capítulo são descritos alguns dos recursos presentes no RTOS utilizado, considerações sobre os sensores de corrente, apresentação do diagrama de blocos para o protótipo proposto, além de comentários sobre cada etapa implementada no *firmware*.

4.1 SISTEMA OPERACIONAL DE TEMPO REAL - TI-RTOS

No desenvolvimento de sistemas embarcados a complexidade das tarefas tem exigido cada vez mais a utilização de um RTOS. Escrever códigos em um único *loop* com algumas chamadas de funções ou interrupções de hardware, torna complexa a análise, modificação e depuração do FW [17].

Na versão de FW que está em desenvolvimento é utilizado o TI-RTOS , este oferece *kernel* de tempo real multitarefas, além de componentes de *middleware* e *drivers* de dispositivos [37]. Posto de forma simples, o TI-RTOS é uma biblioteca na qual pode-se adicionar módulos para desempenhar tarefas comuns como, gerenciamento de memória, análise em tempo real, escalonamento (de *threads*) e sincronização (tendo uma *thread* enviando sinal para outra). O benefício disso, é que estes módulos foram desenvolvidos, testados e validados pela TI, desta forma, o usuário (programador) pode focar no FW em desenvolvimento.

As principais características do *kernel* do TI-RTOS são [38]:

- Preemptivo: Isso significa que a *thread* com maior prioridade sempre é executada primeiro. O *time-slicing* não é suportado de forma nativa (pode ser implementado).
- Orientado a eventos: Quaisquer interrupções, ou chamadas de usuário configuradas pelo kernel, são acessadas através do escalonador.
- Baseado em objetos: Todas as *Application Programming Interface* (API) operam em seus próprios objetos, portanto, quando um objeto é modificado os outros não são alterados.
- Determinístico: O escalonador funciona atualizando a fila de eventos de forma que toda a troca de contexto leve o mesmo número de ciclos para ser executada[11].

São suportados quatro tipos de *threads*, listadas na tabela 6 em ordem de prioridade de execução, assim, *Hwi* tem a maior prioridade enquanto que *Idle* a menor.

A criação das *threads* pode ocorrer tanto em tempo de projeto quanto em tempo de execução. Para auxiliar na configuração dos módulos e *threads*, pode ser utilizado tanto o método de configuração por *script* quanto o ambiente gráfico, na figura 23 é mostrado o ambiente gráfico de configuração com uma visão geral dos módulos utilizados no protótipo Protegmed TM4C.

[11] Funções de alocação dinâmica de memória, como *malloc*, não são deterministicas

Tabela 6. Tipos de *threads* suportadas pelo *kernel* do TI-RTOS

Tipo de *thread*	Abreviatura	Característica
Interrupções de *Hardware*	Hwi	Iniciada por evento de *hardware*
Interrupções de *Software*	Swi	Iniciada por chamada de *software*
Tarefas	Task	Normalmente bloqueadas/desbloqueadas por semáforos
Processo em segundo plano	Idle	Em execução quando as demais *threads* não estiverem.

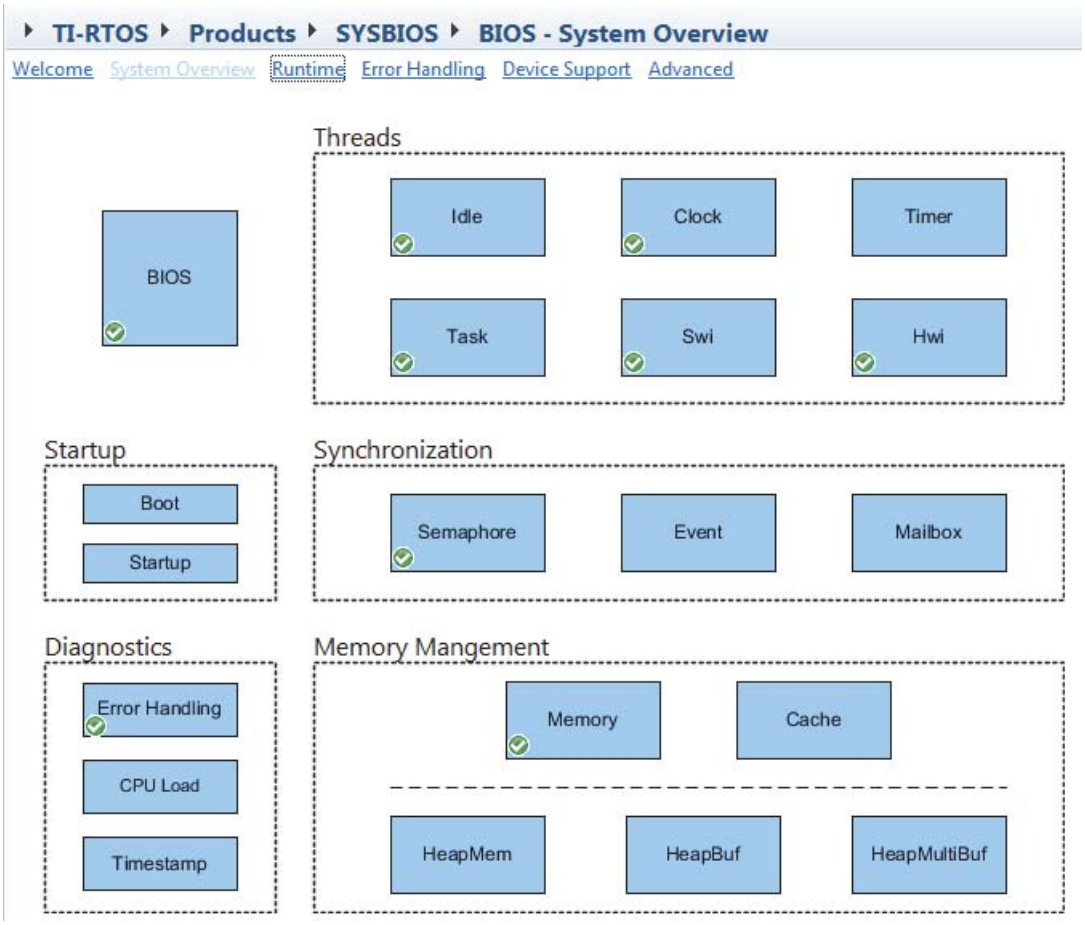


Figura 23. Visão geral dos módulos do RTOS utilizado no Protegmed

Por ser um *kernel* completamente configurável, existe uma certa complexidade no início de sua utilização, um exemplo disto, é a necessidade de especificar qual deve ser tamanho da pilha para qualquer *thread* criada, a escolha de um valor pequeno provoca, obviamente, um estouro de pilha, enquanto que um valor demasiado grande, significa o desperdício de memória RAM.

Para auxiliar nesta questão, o ambiente de *debug* do CCS dispõe de uma ferramenta chamada RTOS Object View (ROV) . Através desta, pode-se, entre outros recursos, verificar qual o pico máximo de utilização da pilha para cada *thread* criada, desta forma, pode-se escolher o valor que melhor descreve a necessidade de cada *thread*, garantindo assim uma boa utilização dos recursos disponíveis.

Quando o módulo de instrumentação é utilizado, o ambiente de *debug* do CCS oferece ferramentas para análise gráfica, estatística, status e *benchmark* das *threads* criadas com o TI-RTOS. Através destas, é possível avaliar o fluxo de execução do escalonador, a percentagem de utilização da

CPU, o comportamento preemptivo das *threads*, o número de chamadas, a concorrência entre outras funções. Como o módulo de instrumentação utiliza *buffers* circulares e funções embutidas dentro do FW do usuário, seu uso consome recursos de memória RAM e de programa.

Neste trabalho, o módulo de instrumentação é utilizado para realizar o *benchmark* utilizado na comparação entre a versão mbed e (*TM4C*).

4.2 SENSOR DE CORRENTE

Para a aquisição dos sinais de corrente de alimentação, corrente diferencial, corrente do condutor de aterramento (PE) e tensão, são utilizados sensores de corrente elétrica do tipo transformador de corrente (TC) . Estes sensores funcionam com o mesmo princípio dos transformadores, onde, a relação de corrente entre o primário e o secundário, é dada pela relação do número de espiras. São do tipo não invasivo, isto é, não interrompem o circuito onde estão inseridos, e oferecem excelente isolação galvânica, desta forma, podem ser utilizados em altas correntes, entretanto, não são capazes de medir corrente contínua [39].

Um dos problemas deste tipo de sensor é possibilidade de saturação do núcleo magnético quando operados fora da faixa especificada. Quando utilizados em baixas correntes, apresentam resposta não linear, na amplitude e na fase. Por esse motivo, cada sensor deve ser calibrado individualmente. A calibração fornece os valores do ganho de corrente versus frequência e ângulo de fase versus frequência (apêndice A).

No Protegemed, três configurações de sensores são utilizadas, todas com o mesmo núcleo toroidal magnético (nanocristalino) e número de espiras no secundário. A diferença reside no resistor de carga do secundário(R_L), resistor de ganho de amplificação (R_G) e número de espiras do primário (N_1). Cada uma destas modificações produzem deferentes correntes máximas (I_max). A tabela 7 especifica os resistores e o número de espiras para cada configuração.

Tabela 7. Configuração dos sensores utilizados no Protegmed

Função	R_L (Ω)	R_G (Ω)	N_1	I_{max}
Corrente de alimentação	1	5k6	1	5,2Arms
Corrente diferencial	180	480	3	1,1mA
Tensão alimentação [12]	180	560	1	3,2mA
Corrente de fuga	180	480	1	3,2mA

Na tese de dissertação de mestrado de Figueiredo [40], surgiu a necessidade de um sensor de corrente que pudesse ser conectado em μCs com diferentes tensões de referência no conversor A/D ($V_{ref} = 5,0V$ ou $3,3V$). Desta forma, foram fabricados pela empresa Elomed sensores de corrente com o circuito de condicionamento de sinal integrado ao toróide, mostrados na figura 24.

Neste circuito, a tensão de saída está referenciada a tensão de alimentação do sensor, assim, se a alimentação for de 5V a saída excursiona de 0 à 5V (quando a alimentação é 3,3V a saída vária

[12] O resistor entre fase e neutro é de 220kΩ. 220Vrms=1mArms

de 0 à 3,3V). Além disso, a saída conta com um filtro analógico passa baixa de primeira ordem do tipo RC (Resistor Capacitor) . Tal filtro tem a finalidade de atenuar ruídos elétricos e de atuar como filtro anti-aliasing (fenômeno descrito na seção 2.2.2).

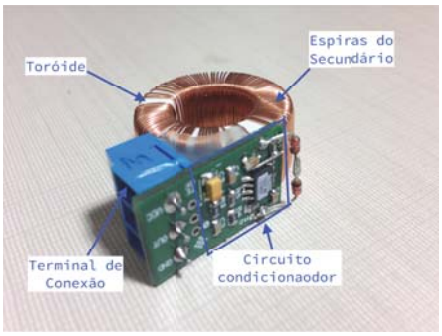


Figura 24. Sensor de corrente SC5A

No Protegemed, são utilizadas as primeiras doze componentes harmônicas da frequência fundamental (60Hz), portanto, a máxima frequência de interesse é $f_{max} = 60\text{Hz} \cdot 12 = 720Hz$. O filtro utilizado no condicionador de sinal tem frequência de canto $f_{-3dB} = 4,8\text{kHz}$ não interferindo na frequência de interesse.

Para fins de aliasing, a frequência de amostragem, antes do circuito de *Hardware Sample Averaging*, é de $f_s = 16 \cdot 15360 = 245760\text{Hz}$. Neste caso, a frequência de Nyquist é de aproximadamente 122kHz, isto equivale a 2,5 décadas acima da frequência de canto $(122kHz/4.8kHz = 25,41)$. Em um filtro de primeira ordem, a atenuação é de -20dB por década, desta forma, o filtro em questão atenua -50dB na frequência de Nyquist.

4.3 DIAGRAMA DE BLOCOS DO PROJETO

Conforme discutido na seção 2.3.5, a implementação atual do Protegmed utiliza dois μC mbed para cada painel de tomada, limitação imposta pelo número de canais analógicos disponíveis.

A versão *TM4C* deve monitorar todas as tomadas presentes no painel, duas tensões de alimentação e a corrente no condutor de proteção. Como todas as tomadas serão monitoradas com apenas um μC , não existe mais a necessidade do *switch* de rede. Na figura 25 é apresentado o diagrama de blocos para a versão do Protegmed utilizando a *TM4C*.

No total são utilizados 15 canais analógicos restando ainda outros 5 canais para futuras implementações. Em cada canal analógico está conectado um sensor de corrente com função específica, corrente de alimentação, diferencial ou de fuga para o condutor de aterramento. Para a medição da tensão de alimentação, um resistor de 220kΩ é utilizado como carga e ligado ao condutor de fase e de neutro do circuito. Esta é uma medição indireta de tensão através da corrente circulante no resistor de 220kΩ. No diagrama de bloco apresentado na figura 25 não é mostrada a alimentação do μCs. A versão (*mbed*) também possui a identificação do EEM conectado através de RFID, na versão *TM4C* este recurso não esta integrado ao FW, entretanto, foi testado como um módulo separado.

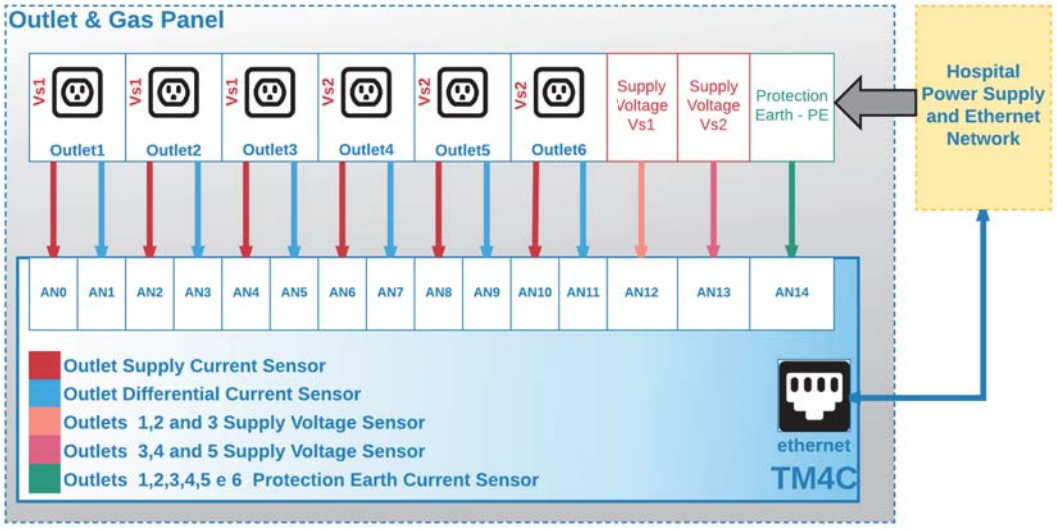


Figura 25. Diagrama de blocos da versão TM4C

4.4 AQUISIÇÃO DE DADOS

Para a tarefa de aquisição de dados, é utilizado o mesmo diagrama apresentado na figura 20, com essa estrutura, a CPU é interrompida somente quando um novo bloco de amostras estiver disponível, toda a tarefa de amostragem é realizada através do DMA.

O período de amostragem é ajustado em um dos temporizadores presentes na TM4C, este, é programado para realizar contagem ascendente com recarga automática ao final do período. A interrupção do temporizador da inicio ao processo de conversão do A/D, definindo assim a frequência de amostragem (f_s). Para o Protegemed, 256 amostras ($n_{samples}$) de uma FO de 60Hz (f_{base}) deve ser utilizada. A frequência de amostragem pode ser obtida pela equação 24.

$$f_s = n_{samples} \cdot f_{base} = 256 \cdot 60Hz \; = \; 15.360Hz \tag{24}$$

O valor de contagem e da recarga ($Tmr_{cnt}calc$) utilizada no temporizador, é definida pela razão entre o *clock* da CPU (CPU_{clock}) e a frequência de amostragem desejada (f_s). Resultado apresentado na equação 25.

$$Tmr_{cnt}calc = \frac{CPU_{clock}}{f_s} = \frac{120.000.000Hz}{15.360Hz} = 7.812,5 \tag{25}$$

Como o valor calculado ($Tmr_{cnt}calc$) não é um número inteiro, o resultado é truncado para 7812 contagens ($Tmr_{cnt}real$). Um incremento de contagem acontece a cada transição no clock da CPU. A frequência de cada interrupção do temporizador é obtida com a equação 26

$$Tmr_{f_s} = \frac{CPU_{clock}}{Tmr_{cnt}real} = \frac{120.000.000}{7812} = 15.360,9831Hz \tag{26}$$

O erro entre a frequência de amostragem (f_s) desejada e a frequência de amostragem real ($Tmr_{cnt}real$) programada no temporizador é calculada conforme a equação 27.

$$err_{f_s} = \left(1 - \frac{f_s}{Tmr_{cnt}calc}\right) \times 100 = \frac{15.360Hz}{15.360,9831Hz} = 0,0064\% \quad (27)$$

Para melhorar a qualidade dos dados amostrados é utilizado um valor de 16 vezes no *Hardware Sample Averaging*. Na prática isso significa que a frequência de amostragem antes do *Hardware Sample Averaging* é 16 vezes maior.

A cada interrupção do DMA, é necessário copiar os dados do *buffer* circular para a variável de processamento. Na implementação do FW foram definidas duas *threads* do tipo Hwi, uma para o conversor AD0 e outra para o conversor AD1. Em cada uma delas, uma instrução *Swi_post* faz a chamada para uma *thread* do tipo Swi onde os dados do *buffer* circular são transferidos para suas respectivas variáveis de processamento. A utilização destes tipos de *threads*, Hwi e Swi, garante a execução sobre outras *threads* criadas, Task ou Idle, desta forma, a aquisição de dados tem sempre prioridade sobre qualquer outra tarefa do FW. Na figura 26 o fluxograma de execução para as *threads* Hwi e Swi é mostrado. O fluxo de execução é o mesmo para o conversor ADC0 e ADC1.

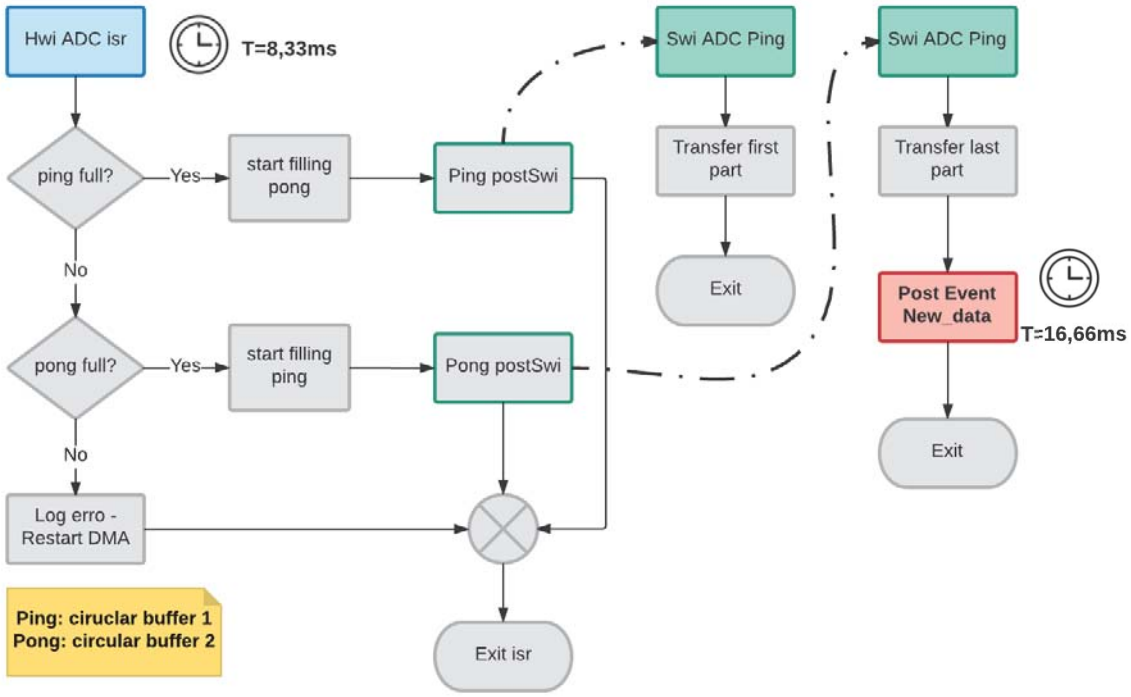


Figura 26. Fluxograma para *thread* Hwi e Swi da etapa de aquisição

O bloco de dados amostrados é organizado em uma matriz $m \times n$, sendo m o número do canal analógico amostrado e n o número da amostras. Por exemplo, *ADCChannel[7][100]* refere-se ao canal analógico 7 e a amostra número 100. Para o FW da *TM4C* foi definida a matriz *ADCChannel[15][256]*, ou seja, 15 canais analógicos com 256 amostras cada.

4.5 ANÁLISE DOS DADOS CAPTURADOS PELOS SENSORES DE CORRENTE

Nesta etapa os dados amostrados, provenientes dos sensores de corrente, devem ser normalizados e analisados. O processo de normalização consiste em remover o deslocamento de tensão adicionado ao canal analógico e aplicar o ganho específico do sensor de corrente utilizado. Para remover o deslocamento de tensão, a média das amostras é subtraída de cada valor amostrado, em seguida o ganho do sensor é aplicado. Este processo deve ser executado em cada canal analógico utilizado.

Após a normalização o valor rms é calculado e comparado a constantes previamente estabelecidas. Este conjunto de constantes são armazenados em uma memória não volátil (EEPROM), ou seja, mantém os dados durante a perda de alimentação. O valor de cada constante representa o limiar para a detecção de um evento específico, por exemplo, uma corrente de 1,0Arms medida no canal analógico 0 deve iniciar o evento de liga do EEM conectado.

A análise dos dados provenientes dos sensores produz os seguintes eventos:

1. Evento de corrente de alimentação:
   (a) Se o valor rms está acima do limiar estabelecido é gerado o evento de liga.
   (b) Se o estado anterior era ligado e valor rms estiver abaixo do limiar estabelecido, é gerado um evento de desliga.
2. Evento de corrente diferencial:
   (a) Se o valor rms está acima do limiar estabelecido é gerado um evento de início de fuga.
   (b) Se o estado anterior era início de fuga e valor rms estiver abaixo do limiar estabelecido, é gerado um evento de fim de fuga.

Sempre que um evento é gerado a FFT deve ser calculada. O resultado do cálculo da FFT é um vetor V com 256 posições ou 128 pares de números complexo na forma $V_{re} + jV_{im}$, representado os componentes harmônicos da frequência fundamental [41]. A frequência fundamental f_{H1} para Protegemed pode ser obtida através da equação 28, onde N representa o número de elementos do vetor V e f_s a frequência de amostragem.

$$f_{H1} = \frac{f_s}{N} = \frac{15.360Hz}{256} = 60Hz \tag{28}$$

Desta forma, o intervalo de frequência compreende 0Hz, componente CC, até $128 \cdot 60Hz = 7.680Hz$. Como no Protegmed utiliza-se apenas as 12 primeiras harmônicas para a reconstrução do sinal, a maior harmônica enviada é $12 \cdot 60Hz = 720Hz$. Na reconstrução do sinal, realizada na interface de usuário do PC, é calculado o espectro da amplitude, equação 29, e a fase de V, equação 30, para cada componente harmônico enviado.

$$|V| = \sqrt{V_{re}^2 + V_{im}^2} \tag{29}$$

$$\angle V = \tan\left(\frac{V_{im}}{V_{re}}\right)^{-1} \tag{30}$$

Assim, o sinal (V) pode ser reconstruído (V_r) utilizando a equação 31.

$$V_r = \sum_{n=1}^{12} |V_{[n]}| \cdot \cos(2\pi n \cdot t + \angle V_{[n]}) \tag{31}$$

Apesar de ser possível codificar facilmente a equação 7 (rms discreto), o mesmo não acontece com relação a FFT (equação 12) . Para realizar os cálculos de rms e FFT é utilizada a biblioteca CMSIS DSP. Esta biblioteca foi desenvolvida pela ARM, possui 61 funções matemáticas de uso comum no processamento digital de sinais sendo otimizada para processadores Cortex-M [42]. Sua licença permite a aplicação em projetos comerciais ou não comerciais. Apesar desta biblioteca não estar disponível de forma nativa no ambiente de desenvolvimento CCS, é possível compilar uma versão compatível com os processadores baseados em ARM fabricados pela TI [43]. Na figura 27 é apresentado um fragmento do código, que usa a biblioteca CMSIS DSP, para normalizar e calcular o valor rms do grupo de tomadas presentes no painel.

```
/* process all outlets */
for(i=0;i<OUTLET_COUNTER;i++)
{
    // ADC offset
    arm_offset_f32(outlet[i].phaseWave, -ptgmSettings.channel[i].channel_offset , outlet[i].phaseWave, SAMPLE_FRAME);
    arm_offset_f32(outlet[i].diffWave, -ptgmSettings.channel[i+1].channel_offset , outlet[i].diffWave, SAMPLE_FRAME);
    // ADC scale
    arm_scale_f32(outlet[i].phaseWave, ptgmSettings.channel[i].channel_gain , outlet[i].phaseWave, SAMPLE_FRAME);
    arm_scale_f32(outlet[i].diffWave, ptgmSettings.channel[i+1].channel_gain , outlet[i].diffWave, SAMPLE_FRAME);
    // calculate RMS
    arm_rms_f32(outlet[i].phaseWave, SAMPLE_FRAME, &outlet[i].phaseRMS);
    arm_rms_f32(outlet[i].diffWave, SAMPLE_FRAME, &outlet[i].diffRMS);
}
```

Figura 27. Fragmento de código utilizando a biblioteca CMSIS DSP

As instruções da biblioteca utilizam ponteiros na manipulação das variáveis de entrada e saída, o que significa que não a duplicidade de variáveis durante sua execução. A descrição de cada função é simples e intuitiva, sempre iniciam com o nome do fabricante/desenvolvedor (*arm*) seguindo da funcionalidade (*offset*, *scale*, *rms*) e do tipo de dado que manipulam (neste caso, ponto flutuante com 32 bits - *f32*).

No desenvolvimento do FW da *TM4C* o cálculo do valor rms é realizado por uma *thread* do tipo Task. Este tipo de *thread* tem estrutura semelhante a um mini programa *main* escrito em C, ou seja, inicializam variáveis antes de entrarem em um laço infinito (ou não). A diferença reside no fato da execução desta *thread* ser controlada pelo escalonador. Tal controle se dá por meio de semáforos ou eventos.

Uma *thread* denominada *capture Task* é responsável pela normalização e cálculo do valor rms, quando finalizada desbloqueia a execução da *thread event Task*, responsável pela detecção dos eventos acima citados (liga, desliga, fuga...). Quando um evento é detectado, a FFT é calculada, o resultado é colocado na fila de mensagens e um semáforo sinaliza desbloqueio da *thread* responsável pelo envio dos dados ao servidor. Na figura 28 é apresentado o fluxograma de execução desta etapa.

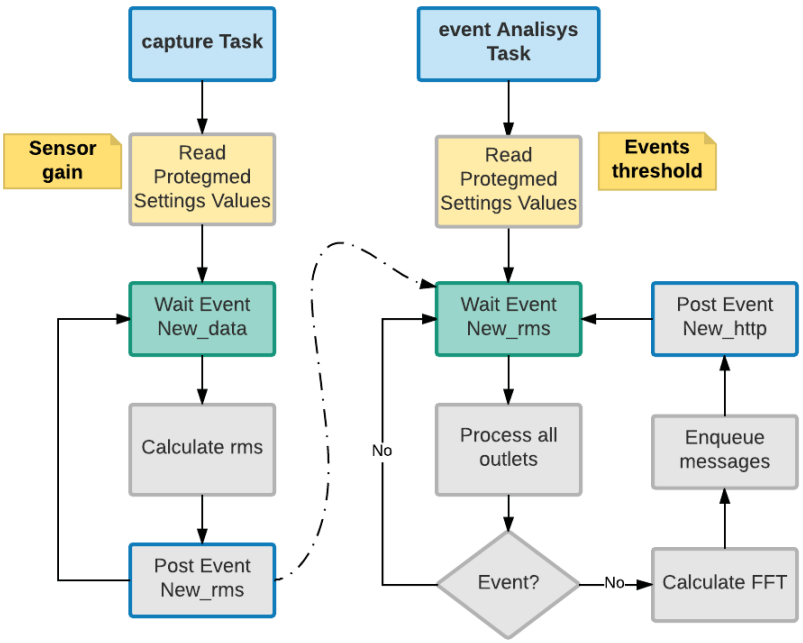


Figura 28. Fluxograma de execução das *threads* de análise

Como referência pode-se comparar a evolução das versões de *hardware* baseadas em ARM utilizadas no Protegemed. Na tabela 8 são listadas algumas características de *hardware* e de processamento.

Tabela 8. Evolução das versões do Protegemed

Elemento	AT91SAM7X256	LPC1768	TM4C1294
Ano da versão	2010	2012	2017
Frequência de operação (Clock – MHz)	48	96	120
Memória Flash (Kb)	256	512	1024
Memória SRAM (Kb)	64	32	256
Número de tomadas supervisionadas	4	3	6
Tempo de cálculo do RMS por tomada	3ms	711μs	105,2μs
Tempo de cálculo da FFT por tomada	50ms	11,5ms	833μs
Custo (US $)	16,24	49,99	19,99

Entre a versão atual, LCP1768 (*mbed*), e a versão protótipo, TM4C, é possível observar que mesmo o *clock* aumentando apenas 25%, a velocidade de processamento no cálculo de rms foi reduzido em quase 7 vezes e o processamento da FFT em aproximadamente 14 vezes. Tais diferenças são atribuídas ao aumento de performance, entre o Cortex-M3 (*mbed*) e o Cortex-M4F (*TM4C*), a

presença de um *hardware* de ponto flutuante (FPU do inglês *Floating Point Unit*) e a utilização da biblioteca CMSIS DSP[42].

4.6 COMUNICAÇÃO COM O SERVIDOR DO PROTEGMED

Quando um dos eventos descritos em 4.5 ocorrer, um bloco de dados, contendo informações sobre sobre tal evento, é enviado ao servidor do Protegmed. Algumas considerações sobre os protocolos e modelos de comunicação utilizados pelo Protegmed são descritos abaixo.

A comunicação em rede pode ser representada em camadas, onde cada uma delas trata uma parcela da informação. As camadas são organizadas de forma que o nível mais elevado apresenta maior abstração que o anterior. Como cada camada é empilhada sobre a outra, é comum referir-se a isso como pilha TCP/IP. Dois modelos de referência se destacam, o modelo *Open System Interconnection*(OSI), que utiliza 7 camadas, e o modelo TCP/IP, que utiliza 4 camadas [44]. Na figura 29 são apresentados os modelos OSI e TCP/IP.

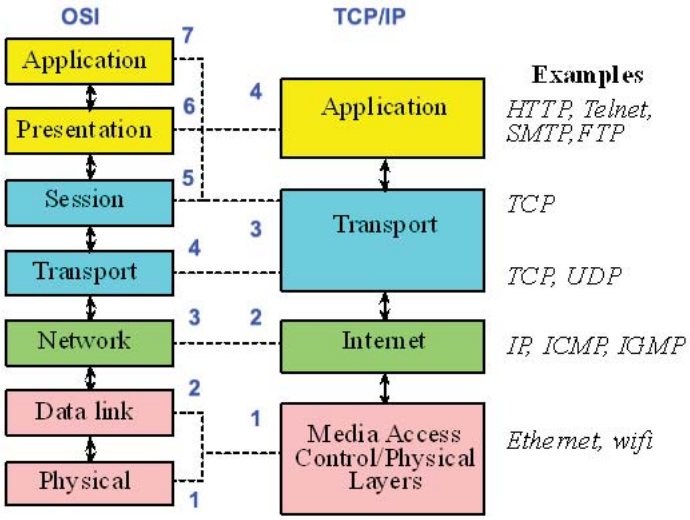


Figura 29. Modelo OSI e TCP/IP, adaptado e modificado de [16]

O Protocolo de Controle de Transmissão (TCP - do inglês *Transmission Control Protocol*) é responsável pelo transporte dos dados, garantindo a entrega de maneira confiável e livre de erros. O Protocolo de Internet (IP - do inglês *Internet Protocol*) é responsável pelo endereçamento e o roteamento, oferecendo uma identificação única a cada dispositivo conectado a uma rede(endereço de IP). O conjunto de protocolos TCP/IP permite que dispositivos com arquiteturas distintas, executando *softwares* diferentes possam comunicar-se entre si. A utilização do protocolo TCP/IP oferece diversos benefícios como, padronização, interconectividade e roteamento [45].

A comunicação, através da rede *ethernet*, é inteiramente gerenciada pelo TI-RTOS, para seu desenvolvimento foi utilizado o Network Development Kit (NDK) fornecido pela TI. O NDK implementa a pilha TCP/IP com utilização reduzida de memória RAM, além disso, o NDK é isolado do sistema operacional nativo e do *hardware* de baixo nível por interfaces de programação abstraídas. O sistema operacional é abstraído por uma camada de adaptação do sistema operacional (SO) e o *hardware*

através da biblioteca de camada de adaptação de hardware (HAL - do inglês Hardware Adaptation Layer). Essas bibliotecas são utilizadas para interagir com *kernel* do TI-RTOS e com os periféricos (*hardware*) do sistema[46].

A comunicação entre o mbed e o servidor do Protegemed utiliza o protocolo de Transferência de Hipertexto (HTTP), quarta camada no modelo TCP/IP, e o método POST para realizar a transmissão de dados entre eles. Em uma dissertação de mestrado, Schmitz[47], propôs a utilização do protocolo *WebSocket* como solução para o problema da comunicação bidirecional (sem a implementação de um servidor *web* no mbed). Na versão TM4C o protocolo *WebSocket* não é implementado, entretanto, um servidor TCP/IP e um cliente HTTP são utilizados. Na figura 30 é apresentado o fluxograma executado pelas *threads* de comunicação.

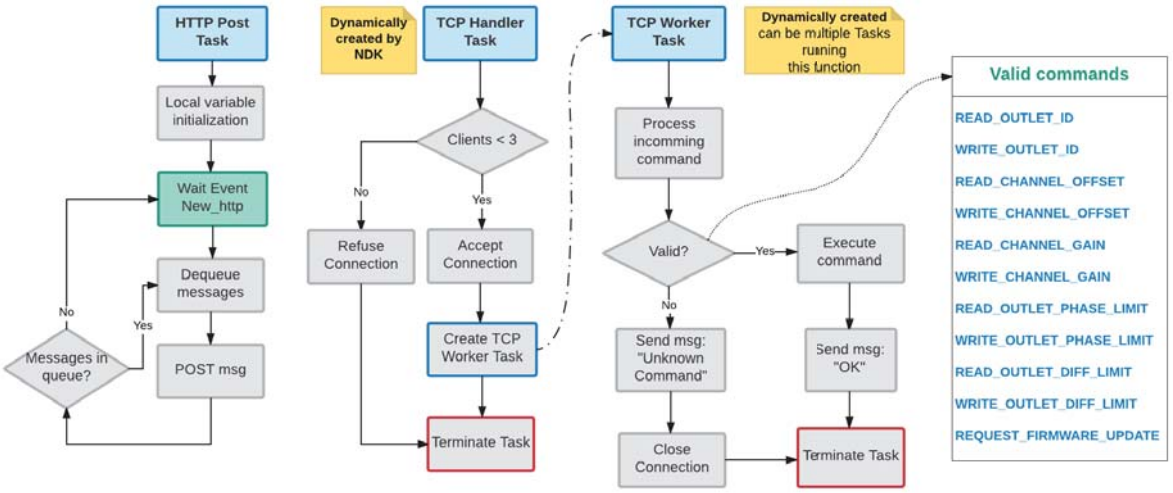


Figura 30. Fluxograma das *threads* de comunicação

As *threads TCP Hanlder* e *TCP Worker* são do tipo Task e foram criadas em tempo de execução. A *thread HTTP POST* é executado enquanto houverem mensagens na fila.

4.6.1 Atualização remota de *firmware*

Na versão mbed o FW somente era atualizado de forma local, ou seja, o binário contendo o novo FW era transferido através da porta USB do PC. Para a versão a TM4C o *bootloader* presente na memória *ROM* é utilizado. O *bootloader* é um pequeno código responsável pelo gerenciamento da transferência do FW de um servidor para a memória *flash* (memória de programa) do μC[48].

Na implementação realizada, para que um novo FW seja transferido, um comando específico foi definido, este comando deve iniciar com o caracter 'U' seguido pelo número de MAC do μC. Quando isso ocorre, todas as interrupções são desabilitadas, o μC é reinicializado e o *bootloader* é executado.

No *bootloader*, o protocolo de rede *bootstrap* (BOOTP) é utilizado, este é um antecessor do protocolo DHCP e é usado para descobrir o endereço IP do cliente, o endereço IP do servidor e o

nome do arquivo de FW a ser transferido. O BOOTP usa pacotes UDP/IP para a comunicação entre cliente e servidor (*bootloader* atua como o cliente).

O *bootloader* envia, utilizado o protocolo BOOTP, uma mensagem em *broadcast*, quando o servidor recebe a requisição BOOTP, ele responde informando ao *bootloader* o endereço IP do cliente (*TM4C*), o endereço IP do servidor e o nome do arquivo de FW. Uma vez que esta resposta é recebida, o protocolo BOOTP é concluído e transferência do FW iniciada utilizando o protocolo Trivial File Transfer Protocol (TFTP)[48]. O TFTP é um protocolo simples para transferência de arquivos, portanto, não dispõe dos recursos comumente encontrados na protocolo FTP regular, como, listagem de diretórios e autenticação de usuários[49]. Assim que a transferência é concluída, o *bootloader* reinicializa o μC e o novo FW é executado.

A TI fornece o *software LM flash programmer* que atua como servidor para o *bootloader*[50]. Este *software* foi utilizado nos testes de atualização realizados na TM4C. Na figura 31 é apresentado a tela de configuração do *LM Flash Programmer*

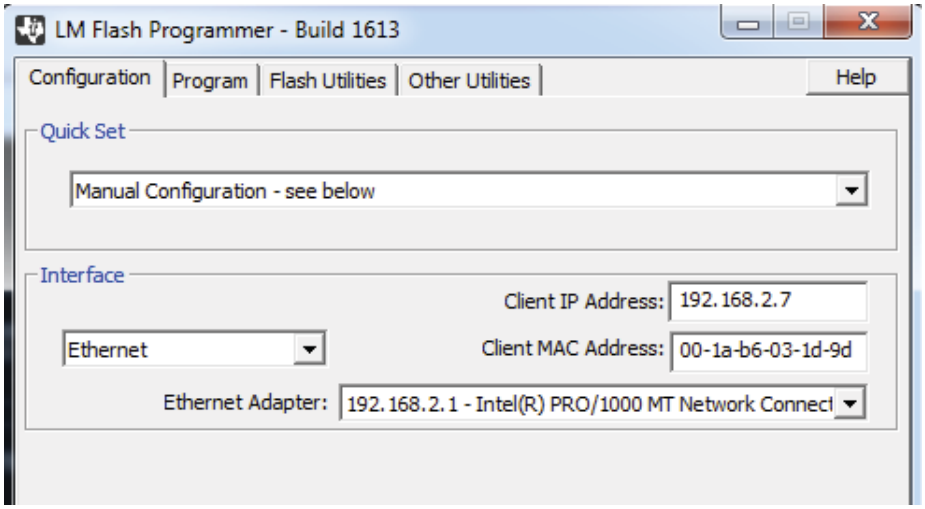


Figura 31. Tela de configuração do *LM Flash Programmer*

Algumas considerações devem ser feitas a respeito da atualização de FW através da rede *ethernet*. Durante a transferência do arquivo, se a conexão com a rede for perdida, o processo de transferência é reinicializado assim que a conexão for novamente estabelecida. Entretanto, se houver perda de alimentação durante a transferência, o FW presente na memória *flash* estará incompleto e por consequência não será executado. Neste caso, uma transferência local deve ser realizada.

5. RESULTADOS

Os resultados aqui apresentados estão organizados de forma modular, assim, na seção 5.1 são mostrados os testes realizados no cálculo do valor rms, espectro de frequência e o ambiente utilizado para sua realização. Na seção 5.3 a *TM4C* é conectada ao servidor do Protegmed, através da rede *ethernet*, para verificar a captura de eventos e a correta reconstrução das formas de onda adquiridas. O atualização de FW de forma remota é apresentada na seção 5.5 e seu funcionamento investigado e avaliado. Na última parte, seção 5.4, o resultado do teste realizado para a deteção de uma *tag* de RFID é apresentado.

5.1 VALIDAÇÃO DOS DADOS AMOSTRADOS

O objetivo dos testes, rms e espectro de frequência, é verificar o funcionamento do FW e dos conversores A/D. Para isso, foi utilizado um sistema capaz de gerar FOs com amplitude e componentes harmônicos conhecidos. Desta forma, pode-se comparar os valores teóricos com o resultado obtido no experimento prático.

5.2 AMBIENTE DE TESTE UTILIZADO

Quatro FOs foram criadas com componentes harmônicos e amplitudes conhecidas. Para esta tarefa, foi utilizado o *Matlab/Simulik Embedded Coder* como ferramenta de programação e simulação em conjunto com um μC da série C2000, o μC F28069M (fabricado pela TI).

O *Embedded Coder* permite que um modelo criado no *Matlab/Simulink* seja compilado em um código executável no μC F28069M. Desta forma, é possível simular o funcionamento do modelo (FW) antes de transferi-lo ao μC.

As FOs são reconstruídas, a partir de dados discretos armazenados em tabelas, estes dados, tem origem em quatro equações que descrevem as FOs em função do tempo. A reconstrução no ambiente físico é realizada pelos conversores D/A presentes no μC F28069M.

A utilização do *Matlab/Simulink* como ferramenta de programação permite que as FOs criadas para o teste, possam ser avaliadas ainda no ambiente de desenvolvimento. Desta forma, o valor rms e espectro de frequência foram determinados antes de sua reconstrução física.

Na figura 32 são apresentadas as quatro formas de onda utilizadas no teste. A primeira delas, figura 32(a), é composta da harmônica fundamental (60Hz) somada a outras duas harmônicas de ordem impar. Na figura 32(b) a harmônica fundamental é somada a outras três componentes também de ordem ímpar. Uma onda senoidal pura é apresentada na figura 32(c) e uma onda senoidal com ruído branco na figura 32(d).

Figura 32. Formas de onda utilizadas durante os testes de aquisição

Para verificar a detecção de eventos, teste realizado na seção 5.3, no FW da TM4C, as amplitudes das FOs de teste precisam ser alteradas dinamicamente, tal modificação pode ser realizada enviando um comando específico, através da interface serial, ao μC F28069M. Na figura 33 é mostrada a estrutura de conexão para os testes realizados.

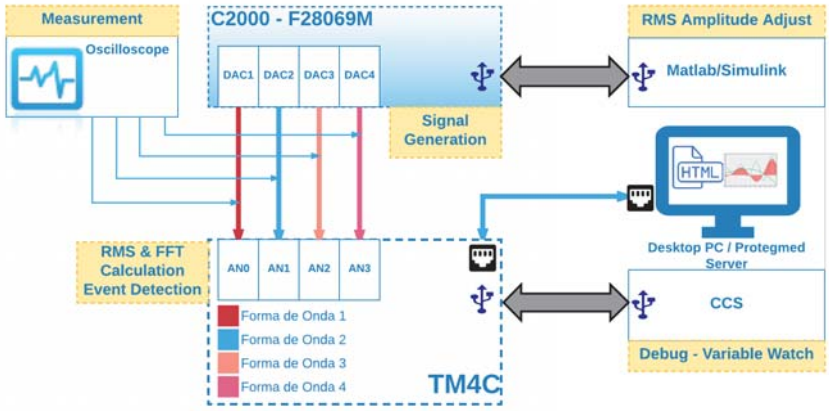


Figura 33. Ambiente de teste para aquisição de dados

5.2.1 Valor rms

Neste teste, as amplitudes das FOs são mantidas constantes e o valor rms é lido no osciloscópio e no ambiente de *debug* da CCS. Os valores de tensão rms e amplitude presentes nos gráficos da figura 32(a) , 32(b), 32(c) e 32(d) são calculados com base na simulação realizada no ambiente do *Matlab/Simulink*. Na tabela 9, coluna Simulink, os dados de simulação são apresentados seguidos dos valores medidos (osciloscópio) e calculados (*TM4C*). Exceto para o valor simulado, os dados apresentados na tabela 9 são o resultado da média de 20 amostras.

Tabela 9. Teste de aquisição de dados - Tensão rms

FO	Tensão rms (V)			erro(%) TM4C/Osc.
	Simulink	Osciloscópio	TM4C	
1	0,706	0,702	0,690	-1,71
2	0,362	0,356	0,348	-2,47
3	0,932	0,920	0,923	0,32
4	0,761	0,738	0,736	-0,27

O erro relativo entre o valor calculado pela *TM4C* e medido pelo osciloscópio é obtido através da equação 32.

$$\delta_{erro} = \left(\frac{TM4C}{Osciloscpio} - 1 \right) \times 100 \qquad (\%) \tag{32}$$

5.2.2 Espectro de frequências

O espectro de frequência para as FOs de teste foi calculado pelo *Matlab/Simulink* e pelo FW da TM4C. Na tabela 10 é mostrado o resultado destes cálculos.

Tabela 10. Teste de aquisição de dados - FFT

FO	Frequência (Hz)	Amplitude (V)		TM4C/Sim.(%)
		Simulink	TM4C	
1	60	0,8262	0,8064	-2,39
	180	0,4935	0,4873	-1,25
	420	0,2644	0,2479	-6,24
2	60	0,3323	0,3551	6,86
	180	0,2655	0,2631	-0,90
	420	0,1664	0,1560	-6,25
	540	0,2298	0,2120	-7,74
3	60	1,3160	1,3025	-1,02
4	60	1,0650	1,0376	-2,57

O erro relativo entre o valor simulado pelo *Matlab/Simulink* e pela *TM4C* é obtido através da equação 33.

$$\sigma_{erro} = \left(\frac{TM4C}{Simulink} - 1 \right) \times 100 \qquad (\%) \tag{33}$$

5.2.3 Discussão dos resultados para o teste rms e FFT

Para o teste de cálculo do valor rms, o maior erro relativo foi encontrado na FO2 seguida pela FO1. Nestas FOs estão inseridas, de forma proposital, componentes harmônicos que distorcem a FO senoidal. A influência de tal distorção harmônica, pode ter contribuído para a elevação do erro apresentado no cálculo realizado pelo FW da *TM4C*. Tal afirmação é corroborada pelos valores encontrados para as FOs 3 e 4, livres de harmônicas, ambas com erro relativo inferior a 0,5%. Fatores como má conexão e temperatura também podem contribuir para o aumento do erro.

Na análise do espectro de frequência, calculado pelo FW da TM4C e simulado pelo *Matlab/-Simulink*, os picos de amplitude coincidiram com a frequência de origem. O maior erro relativo ocorreu na FO2 em sua nona harmônica.

O ambiente de teste utilizou duas placas de desenvolvimento conectadas entre si através de condutores de prototipagem rápida, alimentadas exclusivamente pela porta USB do PC. Tal situação podem levar a flutuações na alimentação dos μCs e por consequência influenciar na performance da amostragem e dos cálculos. De maneira geral os resultados foram capazes de reproduzir os valor simulados e medidos, entretanto, quando da montagem em placa customizada, os testes devem ser repetidos a fim de verificar a melhora de desempenho.

5.3 INTEGRAÇÃO COM SERVIDOR DO PROTEGEMED

O servidor do Protegemd é executado em uma máquina virtual e recebe dados, pela rede *ethernet*, através do método POST do protocolo HTTP. No ambiente de teste a máquina virtual está instalada em um PC que funciona como servidor. A visualização dos dados capturados é realizada pela interface *web* disponibilizada pelo servidor.

Para o teste, as FOs geradas pelo μC F28069M são ligadas a TM4C conforme apresentado na figura 33. Os eventos de liga/desliga são obtidos elevando a amplitude da FO de teste através de um modelo do *Matlab/Simulink*. Este modelo envia os novos valores de amplitude através da interface serial para o μC F28069M.

Quando a amplitude da FO de teste é elevada, o FW da *TM4C* interpreta isso como um evento de liga, assim, envia uma mensagem ao servidor do Protegemed contendo, entre outras informações, um conjunto com 24 valores que representam o par de números complexos obtidos pelo cálculo da FFT.

Estes valores são utilizados na reconstrução da FO que é apresentada no ambiente *web* do Protegemed. Na figura 34(a) é mostrado o resultado da aquisição da FO1 no navegador *web*. Quando compara com a FO gerada pelo *Matlab/simulink* e, posteriormente, pelo μC F28069M, percebeu-se que a FO mostrada no ambiente *web* estava espelhada. Tal espelhamento pode ser causado por um erro no FW da *TM4C* ou por alguma falha na reconstrução da FO realizada pelo servidor do Protegmed. Com auxilio de um osciloscópio foi possível confirmar o espelhamento, imagem da tela do osciloscópio na figura 34(b).

(a) (b)

Figura 34. Captura com Protegemed *WEB* da FO1

Para verificar o funcionamento do FW da *TM4C* os dados de uma amostra bruta, 256 valores discretos, foram exportados para o formato *csv* e avaliados no ambiente do *Matlab*. Para essa avaliação, um pequeno *script*(apêndice B) foi criado. Este *script* recebe como entrada a frequência de amostragem e o vetor amostrado, como saída, apresenta três gráficos contento a FO original (baseado nos dados brutos), o espectro da frequência e a reconstrução da FO a partir das 12 primeiras harmônicas do sinal sob teste, neste caso FO1. A reconstrução da FO no *script* é realizada aplicando as equações 29, 30 e 31. O gráfico gerado por este *script* é apresentado na Figura 35.

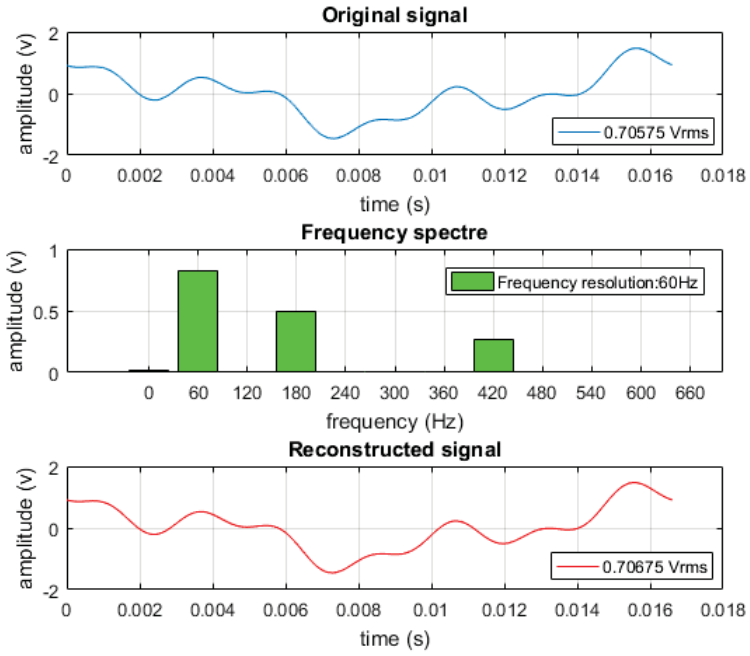


Figura 35. Resultado da reconstrução de amostra bruta com o *Matlab/Simulink*

O gráfico confirma que os dados amostrados pela *TM4C* estão corretos e que a reconstrução utilizando as 12 primeiras harmônicas é realizada corretamente pelo *script* do *Matlab*.

Para corrigir este problema foi necessário realizar uma modificação na função do servidor responsável pela reconstrução da FO a partir de suas componentes harmônicas. Após correção a FO de teste foi corretamente exibida no Protegemed *web*, figura 36.

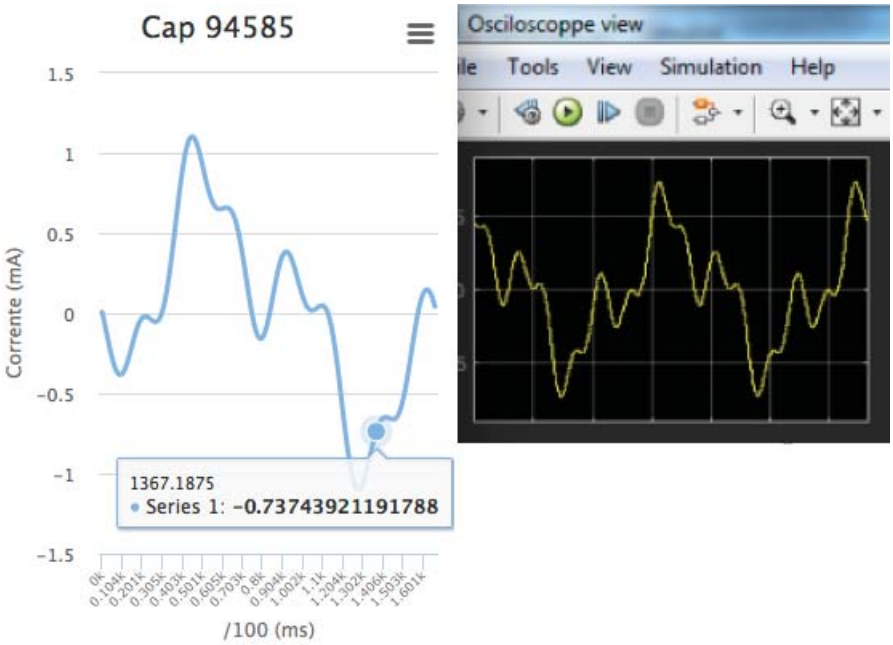


Figura 36. Captura da FO de teste corrigida

5.4 UTILIZAÇÃO DE RFID

Apesar de não estar integrado ao protótipo construído com a TM4C, uma versão para teste de identificação de EEM utilizando RFID foi implementada. Neste teste uma ambiente simples é montado conforme a figura 37.

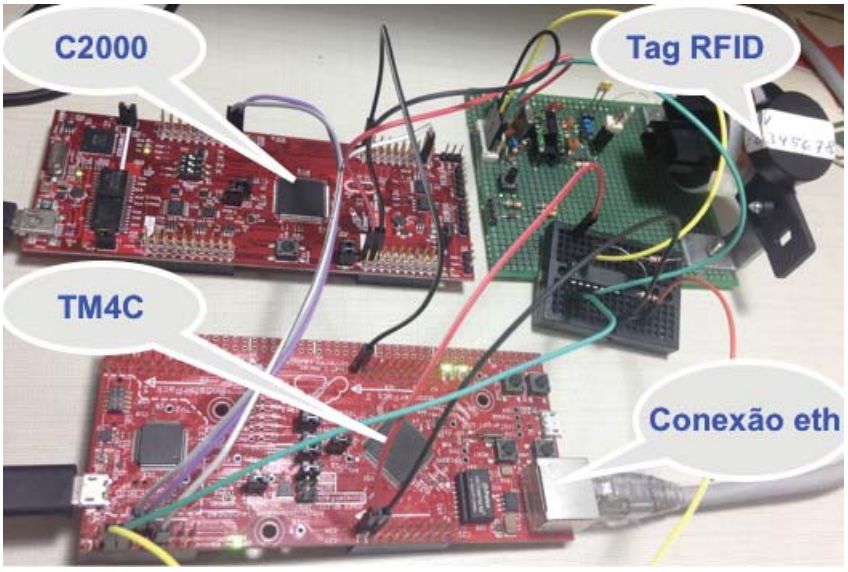


Figura 37. Teste de identificação utilizando RFID

Através da rede *ethernet* um comando de leitura da *tag* é solicito. Uma *thread* implementada no FW realiza a leitura e envio do número da *tag* identificada. Na figura 38 é mostrado a tela do *software* de comunicação utilizado no teste de *RFID*.

Figura 38. Software de comunicação utilizado no teste de RFID

5.5 TESTE DE ATUALIZAÇÃO DE *FIRMWARE* ATRAVÉS DA REDE

Para testar a atualização do FW da versão protótipo, na etapa de comunicação, um comando de início de atualização foi inserido. Caso a TM4C receba o caracter 'U' seguido de seu número de MAC, todas as interrupções são desabilitadas, o μC é reinicializado e o *bootloader* é executado. O funcionamento do *bootloader* é descrito na seção 4.6.1. O *software LM Flash Programmer* é utilizado como servidor de atualização e *software Packet Sender*[51] como interface TCP/IP para solicitar a atualização. Através do *software Wireshark*[52] é possível observar a sequência de acontecimentos na rede de comunicação durante o processo de atualização. Na figura 39 é exibida a captura de tela do *Wireshark* durante a atualização.

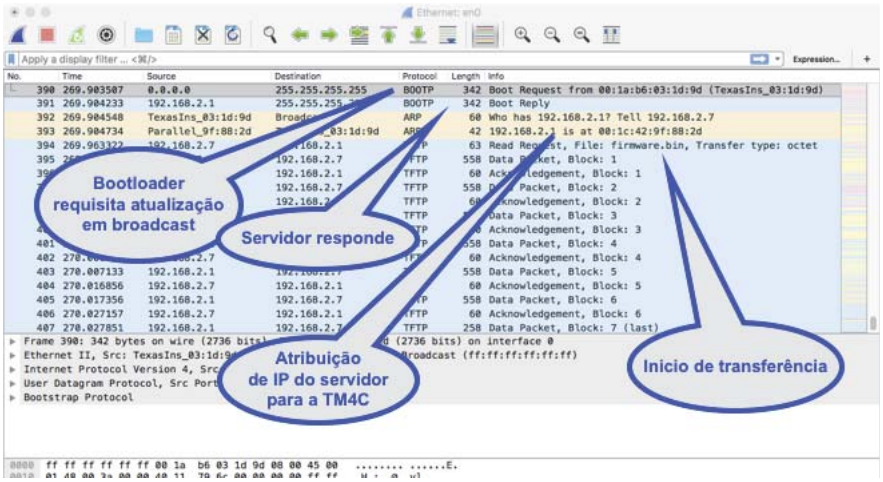


Figura 39. Captura do processo de atualização com *Wireshark*

6. CONSIDERAÇÕES FINAIS

Durante a pesquisa realizada verificou-se a existência de dois segmentos distintos, um utilizando μCs e outro SBCs. Através das especificações do projeto, foram selecionadas uma plataforma de cada segmento para testes de *hardware* e *firmware*. Avaliadas as características, a escolha foi o μC TM4C como nova plataforma para o Protegemed.

A implementação do firmware na *TM4C* se deu de forma modular e gradual, sendo o objetivo principal comprovar suas funcionalidades e desempenho quando comparado ao μC utilizado atualmente. A utilização de memória RAM foi de aproximadamente 65% para os módulos testados, restando ainda 35% para futuras implementações. A TM4C, ARM Cortex-M4F, proporciona uma evolução significativa nas tarefas relacionadas a aquisição e processamento de dados, dentre as quais destacam-se:

- A qualidade das amostrados é melhorada pela utilização do *Hardware Sample Averaging*.
- A carga de processamento na CPU é reduzida com a utilização do DMA.
- Os cálculos de rms e FFT são acelerados pela presença de um *hardware* de ponto flutuante e pela utilização da biblioteca CMSIS DSP.
- A amostragem ocorre de forma contínua, todos os ciclos da FO são monitorados.

Os canais analógicos adicionais permitem que a tensão elétrica aplicada ao EEM e a corrente no condutor de aterramento possam ser monitoradas. Tais recursos foram sugeridos por Concetta [53], integrante do grupo de pesquisa do Protegmed na Itália, em sua dissertação de mestrado. Durante o mestrado em eficiência energética Lima[54] verificou que sobre-tensões na alimentação do EEM poderiam levar a um falso evento de fuga no Protegemed. Através da análise da corrente no condutor de aterramento e da tensão de alimentação, será possível verificar tal situação.

Durante os testes de integração com o servidor da plataforma atual, foi identificado, no servidor, uma pequena falha no processo de reconstrução da FO a partir de seus componentes harmônicos. Após correção as ondas capturadas com a *TM4C* foram fielmente reproduzidas no ambiente *web* do Protegemed.

A comunicação bidirecional foi realizada através de um servidor TCP/IP, implementado na *TM4C*, com protocolo de comunicação dedicado. A atualização do FW através da rede *ethernet* proporciona grande flexibilidade em um ambiente de acesso restrito como a sala de cirurgia do hospital. A possiblidade de modificar o firwmare e transferi-lo de forma remota, sem a necessidade de um deslocamento físico até o local, representa também velocidade na correção de falhas e inserções de novas funcionalidades. Embora a perda de alimentação possa ser um problema se ocorrer durante o processo de atualização, medidas para contornar ou minimizar esse acontecimento estão sendo estudadas.

6.1 TRABALHOS FUTUROS

Durante o processo de testes com capturas de FOs de cargas distintas, observaram-se algumas discrepâncias entre a FO reconstruída a partir de componentes harmônicos e a FO exibida em um osciloscópio.

O método utilizado atualmente consiste em extrair, através da FFT, os componentes harmônicos presentes na FO. Doze destas harmônicas são enviadas, através da rede para o servidor, onde a FO é reconstruída. Entretanto, em sinais provenientes de chaveamento eletrônico, como lâmpadas led ou equipamentos com fontes chaveadas, tal reconstrução é comprometida.

Desta forma, uma versão que envia dados brutos (dados capturados pelos sensores de correntes e não tratados pelo firwmare) e não seus componentes harmônicos foi desenvolvida e será utilizada no trabalho de um outro mestrando do grupo de pesquisa.

Para a versão Protegmed TM4C pode-se listar algumas melhorias que poderão motivar o surgimento de novos trabalhos.

- Armazenamento de eventos em mídia local quando a rede de comunicação estiver indisponível.
- Otimizar a troca de mensagens com o servidor evitando o envio de dados redundantes (como por exemplo o valor de calibração e *offset* do sensor de corrente).
- Desenvolvimento de uma placa customizada utilizando o μC TM4C1294ncpd.
- Pesquisar/desenvolver um sensor de corrente capaz de medir CC com ajuste automático de ganho.
- Desenvolver e certificar uma versão de firmware que opere em tempo real.
- Integrar a identificação do EEM por RFID no firmware de usuário (atualmente o mbed utiliza um segundo μC para esta tarefa).
- Desenvolver um firmware que atue como monitor/medidor de energia (consumo, fator de potência), fazendo uso dos sensores já instalados.
- Implementar o protocolo *websockets* para a comunicação entre servidor e TM4C.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SPALDING, L. E. S. *Método Para Detectar o Risco de Microchoque Através da Supervisão da Corrente Diferencial em Equipamentos Eletromédicos Durante Procedimento Cirúrgico*. 131 p. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2009.

[2] HOROWITZ, P.; HILL, W. *The Art of Electronics*. 3. ed. New York: Cambridge University Press; 3 edition (April 9, 2015), 2015. 1219 p.

[3] REBONATTO, M. T. *Métodos para Análise de Correntes Elétricas de Equipamentos Eletromédicos em Procedimentos Cirúrgicos e Detecção de Periculosidade aos Pacientes*. 115p p. Tese (Doutorado) — Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2015.

[4] VALVANO, J. W. *Embedded Systems: Introduction to Arm® Cortex(TM)-M Microcontrollers*. 5. ed. Texas: CreateSpace Independent Publishing Platform, 2015. v. 1. 508 p.

[5] LIGGESMEYER, M. T. P. Trends in embedded software engineering. *IEEE Software*, v. 26, n. 3, p. 19–25, 2009.

[6] PARAI BANASREE DAS, G. D. M. K. An overview of microcontroller unit: From proper selection to specific application. *International Journal of Soft Computing Engineering*, v. 2, n. 6, p. 220–231, 2013.

[7] GUSSOW, M. *Eletricidade básica*. 2. ed. Porto Alegre: Schaum, 2009.

[8] SINGH, Y.; VERMA, M. *Fundamentals of Electrical Engineering*. 1. ed. New Delhi: Laxmi Publications Pvt Limited, 2010.

[9] BIRD, J. *Electrical Circuit Theory and Technology*. 2. ed. Oxford: Newnes, 2001.

[10] NILSSON, S. A. R. J. W. *Circuitos Elétricos*. 8. ed. São Paulo: Pearson, 2009.

[11] JOHNSON JOHN L HILBURN, J. R. J. D. E. *Fundamentos de Análise de Circuitos Elétricos*. 4. ed. Rio de Janeiro: Prentice-Hall, 1994.

[12] INC., M. *Matlab Documentation - Root mean square level*. 2017. Disponível em: <https://www.mathworks.com/help/signal/ref/rms.html>. Acesso em: 12 Nov. 2017.

[13] CHASSAING, R. *DSP Applications Using C and the TMS320C6x DSK*. 1. ed. New York: Wiley, 2002.

[14] SMITH, S. W. *Digital Signal Processing - A Pratical Guide for Engineers and Scientists*. 666. ed. Burlington: Newnes, 2003. 238 p.

[15] HEIDEMAN, M.; JOHNSON, D.; BURRUS, C. Gauss and the history of the fast fourier transform. *IEEE ASSP Magazine*, v. 1, n. 4, p. 14–21, October 1984.

[16] VALVANO, J. W. *Real-Time Interfacing to Arm® Cortex(TM)-M Microcontrollers*. 3. ed. Texas: CreateSpace Independent Publishing Platform, 2013. v. 3. 600 p.

[17] VALVANO, J. W. *Real-Time Operating System for Arm® Cortex(TM)-M Microcontrollers*. 2. ed. Texas: CreateSpace Independent Publishing Platform, 2014. v. 3. 448 p.

[18] LEWIS, D. W. *Fundamentals of Embedded Software with the Arm® Cortex-M3*. 2. ed. London: Pearson, 2013. 238 p.

[19] MOLLOY, D. *Exploring Raspberry PI: Interfacing to the Real World With Embedded Linux*. 1. ed. Dublin: Willey, 2016. 693 p.

[20] MOLLOY, D. *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux*. 1. ed. Dublin: Wiley, 2014. 600 p.

[21] KRIDNER, J. *BeagleBone: open-hardware expandable computer*. 2016. Disponível em: <http://beagleboard.org/support/bone101>. Acesso em: 30 Jun. 2016.

[22] MOLLOY, D. *Real-Time Beaglebone Interface*. 2014. Disponível em: <http://exploringbeaglebone.com/chapter13/prettyPhoto>. Acesso em: 30 Jun. 2016.

[23] UDOO. *Udoo Docs*. 2014. Disponível em: <http://www.udoo.org/docs/Introduction/Introduction.html>. Acesso em: 30 Jun. 2016.

[24] ARMMBED. *ARMmbed Boards*. 2016. Disponível em: <https://developer.mbed.org/platforms/ST-Nucleo-F401RE/>. Acesso em: 31 Out. 2016.

[25] INSTRUMENTS, T. *TI Connected Launchpads*. 2014. Disponível em: <http://www.ti.com/ww/en/launchpad/launchpads-connected-ek-tm4c1294xl.html>. Acesso em: 30 Jun. 2016.

[26] INSTRUMENTS, T. *TI Cloud Tools*. 2015. Disponível em: <https://dev.ti.com>. Acesso em: 24 Out. 2016.

[27] ATMEL. *SAM3X / SAM3A Series Datasheet*. 2016. Disponível em: <http://www.atmel.com/ru/ru/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf>. Acesso em: Nov. 22, 2016.

[28] INSTRUMENTS, T. *Tiva TM4C1294NCPDT Microcontroller Datasheet*. 2014. Disponível em: <http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf>. Acesso em: Nov. 9, 2017.

[29] UDOO. *Udoo Schematic i.MX6Q Open Source Module Rev.D*. 2013. Disponível em: <http://download.udoo.org/files/schematics/UDOO_REV_D_schematics.pdf>. Acesso em: Nov. 9, 2017.

[30] ARDUINO. *Download the Arduino Software*. 2016. Disponível em: <https://www.arduino.cc/en/Main/Software>. Acesso em: Nov. 21, 2016.

[31] DOMEIKA, M. *Software Development for Embedded Multi-core Systems: A Practical Guide Using Embedded Intel Architecture*. [S.l.]: Elsevier Science, 2011. ISBN 9780080558585.

[32] INSTRUMENTS, T. *Code Composer Studio (CCS) Integrated Development Environment (IDE)*. 2017. Disponível em: <http://www.ti.com/tool/CCSTUDIO#Technical%20Documents>. Acesso em: Nov. 10, 2017.

[33] ECLIPSE. *Eclipse Site*. 2017. Disponível em: <http://www.eclipse.org>. Acesso em: Nov. 17, 2017.

[34] RIDGE, N. *C++ Language Support in Eclipse CDT*. 2017. Disponível em: <https://www.eclipse.org/community/eclipse_newsletter/2017/april/article3.php>. Acesso em: Nov. 17, 2017.

[35] TOULSON, R.; WILMSHURST, T. *Fast and Effective Embedded Systems Design - Applying the ARM mbed*. 1. ed. Oxford: Newnes, 2012. 379 p.

[36] STORE, T. *TI Store - TM4C12994NCPDTI3*. 2017. Disponível em: <https://store.ti.com/TM4C1294NCPDTI3.aspx>. Acesso em: 20 Nov. 2017.

[37] INSTRUMENTS, T. *SYS/BIOS (TI-RTOS Kernel) User's Guide*. 2017. Disponível em: <http://www.ti.com/lit/ug/spruex3t/spruex3t.pdf>. Acesso em: 9 Nov. 2017.

[38] WILBUR, E. *Intro to the TI-RTOS Kernel Workshop - Student Guide*. Dallas, 2015. 344 p.

[39] LEPKOWSKI, J. *Motor Control Sensor Feedback Circuits*. 2003. Disponível em: <http://www.udoo.org/docs/Introduction/Introduction.html>. Acesso em: 15 Nov. 2017.

[40] FIGUEIREDO, J. A. O. de. *Aquisição de grandezas elétricas: definição de arquitetura, método e validação em protótipo*. 102 p. Tese (Doutorado) — PPGCA, Passo Fundo, 2016.

[41] AL.], W. Y. Y. . [et. *MATLAB/Simulink for digital signal processing*. 1. ed. Estados Unidos: The Mathworks, 2012. 486 p.

[42] MARTIN, T. *The designer's guide to the Cortex-M processor family: a tutorial approach*. 1. ed. Oxford: Elsiver, 2013. 312 p.

[43] ASHARA, A. *Using the CMSIS DSP Library in Code Composer Studio for TM4C MCUs*. 2015. Disponível em: <http://www.ti.com/lit/an/spma041g/spma041g.pdf>. Acesso em: 9 Nov. 2017.

[44] TANENBAUM, A. S. *Redes de computadores*. 5. ed. [S.l.]: Pearson, 2011. 600 p.

[45] FALL, W. R. S. K. R. *TCP/IP Illustrated*. 2. ed. [S.l.]: Addison-Wesley, 2011.

[46] INSTRUMENS, T. *TI Network Developer's Kit (NDK) v2.25*. 2016. Disponível em: <http://www.ti.com/lit/ug/spru523j/spru523j.pdf>. Acesso em: Nov. 10, 2017.

[47] SHCMITZ, M. A. *Comunicação bidirecional para a plataforma embarcada do Protegemed*. Passo Fundo, 2017. 79 p.

[48] INSTRUMENTS, T. *TivaWare Boot Loader: USER'S GUIDE*. 2016. Disponível em: <http://www.ti.com/lit/ug/spmu301d/spmu301d.pdf>. Acesso em: 27 Out. 2016.

[49] SOLLINS, K. R. *THE TFTP PROTOCOL (REVISION 2)*. 1992. Disponível em: <https://tools.ietf.org/html/rfc1350>. Acesso em: 27 Out. 2016.

[50] INSTRUMENTS, T. *Flash Programmer, GUI and Command Line*. 2016. Disponível em: <http://www.ti.com/tool/LMFLASHPROGRAMMER>. Acesso em: 15 Jun. 2016.

[51] NAGLE, D. *Download Packet Sender*. 2017. Disponível em: <https://packetsender.com/download>. Acesso em: 25 Out. 2017.

[52] FOUNDATION, W. *Download Wireshark*. 2017. Disponível em: <https://www.wireshark.org/#download>. Acesso em: 27 Out. 2017.

[53] CONCETTA, B. *Validazione Secondo Normativa Tecnica Vigente Di Un Prototipo Di Unita' Di Alimentazione Per Uso Medico*. Roma, 2014. 88 p.

[54] LIMA, D. L. *Análise Da Qualidade E Segurança Elétrica Em Sala Cirúrgica*. Campo Grande, 2017. 152 p.

APÊNDICE A – TABELA DE CALIBRAÇÃO SENSOR ELOMED SC5A

SensorId: *MC5-2010/44898* Imax: 5,2Arms

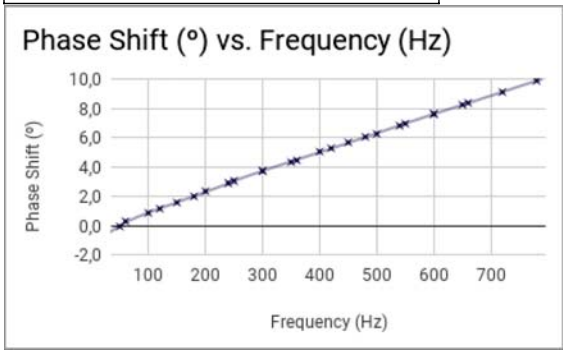
Test Temperature (°C): 25.3

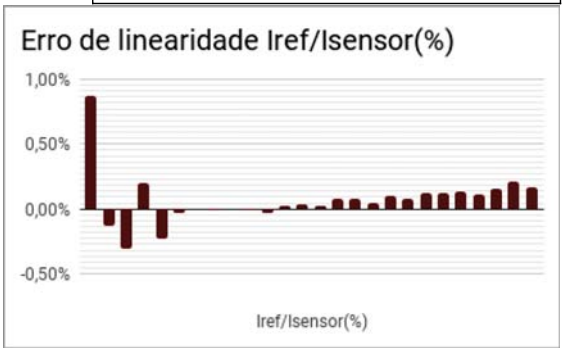
Frequency Response to 3Arms

Frequency (Hz)	Phase Shift (°)	Amplitude Gain (Iref/Isensor)	% Gain
60	0,3	5,390000	0,00%
120	1,2	5,380000	-0,19%
180	2,0	5,380000	-0,19%
240	2,9	5,380000	-0,19%
300	3,7	5,390000	0,00%
360	4,5	5,390000	0,00%
420	5,3	5,400000	0,19%
480	6,1	5,410000	0,37%
540	6,8	5,410000	0,37%
600	7,6	5,420000	0,56%
660	8,4	5,430000	0,74%
720	9,1	5,440000	0,93%
780	9,9	5,450000	1,11%
50	0,0	5,400000	0,00%
100	0,9	5,380000	-0,37%
150	1,6	5,380000	-0,37%
200	2,4	5,380000	-0,37%
250	3,1	5,390000	-0,19%
300	3,7	5,390000	-0,19%
350	4,3	5,390000	-0,19%
400	5,1	5,400000	0,00%
450	5,7	5,400000	0,00%
500	6,3	5,410000	0,19%
550	7,0	5,410000	0,19%
600	7,6	5,420000	0,37%
650	8,3	5,430000	0,56%

Current Linearity to 60Hz

Frequency (Hz)	Iref (Arms)	Isensor (Arms)	Iref/Isensor (%)
60	0,165475	0,166939	0,88%
60	0,342490	0,342044	-0,13%
60	0,394940	0,393723	-0,31%
60	0,735645	0,737082	0,19%
60	0,898065	0,896028	-0,23%
60	1,113175	1,112857	-0,03%
60	1,292440	1,292484	0,00%
60	1,446220	1,446094	-0,01%
60	1,641490	1,641600	0,01%
60	1,828075	1,827862	-0,01%
60	1,987360	1,986819	-0,03%
60	2,046565	2,047073	0,02%
60	2,400205	2,400959	0,03%
60	2,588280	2,588979	0,03%
60	2,766960	2,769102	0,08%
60	2,961205	2,963600	0,08%
60	3,129800	3,131369	0,05%
60	3,360335	3,363727	0,10%
60	3,555290	3,558149	0,08%
60	3,770430	3,775059	0,12%
60	3,948095	3,952827	0,12%
60	4,013400	4,018638	0,13%
60	4,392770	4,397900	0,12%
60	4,614340	4,621505	0,16%
60	4,818770	4,828901	0,21%
60	5,041770	5,050134	0,17%





APÊNDICE B – FUNÇÃO UTILIZADA NA ANÁLISE E RECONSTRUÇÃO FOS

```
function [ X ] = fft_analisys( Fs, signal )
```

Signal reconstruction based on harmonics

Eng. Júlio C. dos Santos

Input: Fs: Sample frequency signal: Original signal vector Output: r_signal: Reconstructed signal

Reconstruction equation: r_singal = dc_component + amplitude * cos(2*pi*f + phi);

Calculate fft

```
T=1/Fs;             % sample periode
L=length(signal);   % frame size
t=(0:L-1)*T;        % time vector
X=fft(signal);      % take the fft
A= abs(X/(L/2));    % amplitude
phi = angle(X);     % angle
A=A(1:L/2+1);       % half of frequency spectre
f=Fs*(0:(L/2))/L;   % frequency vector
% Reconstruct wave based on harmonics defined by Max_harmonic;
Max_harmonic = 60*12;       % 12nd harmonic of 60Hz (720)
p = linspace(0,2*pi,L);     % points in signal
r_signal=A(1);              % DC component
for j=2 : 1 : ( Max_harmonic /(Fs/L))
    r_signal = r_signal + ( A(j) * (cos((j-1)* p + phi(j))));
end
% Plot results
subplot(3,1,1);     % original signal plot
plot(t,signal);
title('Original signal');
xlabel('time (s)');
ylabel('amplitude (v)');
legend(strcat(num2str(rms(signal)),' Vrms'));
grid;
subplot(3,1,2);      % frequency vs amplitude
bar(f(1:( Max_harmonic /(Fs/L))),A(1:( Max_harmonic /(Fs/L))),'g');
title('Frequency spectre');
xlabel('frequency (Hz)');
ylabel('amplitude (v)');
grid;
legend(strcat('Frequency resolution: ', num2str(Fs/L),'Hz'));
subplot(3,1,3);
plot(t,r_signal,'r');   % reconstructed signal
title('Reconstructed signal');
xlabel('time (s)');
ylabel('amplitude (v)');
legend(strcat(num2str(rms(r_signal)),' Vrms'));
grid

end
```